Network Coding for Reliable Data Dissemination in Wireless Sensor Networks

Dissertation zur Erlangung des
naturwissenschaftlichen Doktorgrades
der Julius-Maximilians-Universität Würzburg

vorgelegt von

Isabel Madeleine Runge

Würzburg 2022

Network Coding for Reliable Data Dissemination in Wireless Sensor Networks

Dissertation zur Erlangung des
naturwissenschaftlichen Doktorgrades
der Julius-Maximilians-Universität Würzburg

vorgelegt von

Isabel Madeleine Runge

aus
Wertheim

Würzburg 2022

Eingereicht am: 15.09.2021
bei der Fakultät für Mathematik und Informatik
1. Gutachter: Prof. Dr. Reiner Kolla
2. Gutachter: Prof. Dr. Jörg Nolte
Tag der mündlichen Prüfung: 21.03.2022

Abstract

The application of Wireless Sensor Networks (WSNs) with a large number of tiny, cost-efficient, battery-powered sensor nodes that are able to communicate directly with each other poses many challenges. Due to the large number of communicating objects and despite a used CSMA/CA MAC protocol, there may be many signal collisions. In addition, WSNs frequently operate under harsh conditions and nodes are often prone to failure, for example, due to a depleted battery or unreliable components. Thus, nodes or even large parts of the network can fail. These aspects lead to reliable data dissemination and data storage being a key issue. Therefore, these issues are addressed herein while keeping latency low, throughput high, and energy consumption reduced. Furthermore, simplicity as well as robustness to changes in conditions are essential here. In order to achieve these aims, a certain amount of redundancy has to be included. This can be realized, for example, by using network coding. Existing approaches, however, often only perform well under certain conditions or for a specific scenario, have to perform a time-consuming initialization, require complex calculations, or do not provide the possibility of early decoding. Therefore, we developed a network coding procedure called Broadcast Growth Codes (BCGC) for reliable data dissemination, which performs well under a broad range of diverse conditions. These can be a high probability of signal collisions, any degree of nodes' mobility, a large number of nodes, or occurring node failures, for example. BCGC do not require complex initialization and only use simple XOR operations for encoding and decoding. Furthermore, decoding can be started as soon as a first packet/codeword has been received. Evaluations by using an in-house implemented network simulator as well as a real-world testbed showed that BCGC enhance reliability and enable to retrieve data dependably despite an unreliable network. In terms of latency, throughput, and energy consumption, depending on the conditions and the procedure being compared, BCGC can achieve the same performance or even outperform existing procedures significantly while being robust to changes in conditions and allowing low complexity of the nodes as well as early decoding.

Kurzfassung

Der Einsatz von drahtlosen Sensornetzen (Wireless Sensor Networks, WSNs) mit einer Vielzahl hochintegrierter, kostengünstiger und batteriebetriebener Sensorknoten, die direkt miteinander kommunizieren können, birgt viele Herausforderungen. Aufgrund der großen Anzahl von kommunizierenden Objekten kann es trotz eines verwendeten CSMA/CA MAC Protokolls zu vielen Signalkollisionen kommen. Darüber hinaus arbeiten WSNs häufig unter rauen Bedingungen und die Knoten sind oft anfällig für Ausfälle, z.B. aufgrund aufgebrauchter Energiekapazität oder defekter Komponenten. Infolgedessen können einzelne Knoten oder auch große Teile des Netzes ausfallen. Diese Aspekte führen dazu, dass zuverlässige Datenverteilung und Datenhaltung von entscheidender Bedeutung sind und folglich im Rahmen dieser Arbeit adressiert werden. Gleichzeitig soll die Latenz niedrig, der Durchsatz hoch und der Energieverbrauch möglichst gering gehalten werden. Des Weiteren sind eine geringe Komplexität sowie Robustheit gegenüber veränderten Bedingungen wesentlich. Um diese Ziele zu erreichen, ist ein gewisses Maß an Redundanz nötig. Dies kann beispielsweise durch die Verwendung von Netzwerkkodierung realisiert werden. Bestehende Ansätze liefern jedoch oft nur unter bestimmten Bedingungen oder für ein spezifisches Szenario gute Performanz-Ergebnisse, müssen aufwändig initialisiert werden, benötigen komplexe Berechnungen oder bieten keine Möglichkeit für frühzeitige Dekodierung. Daher haben wir ein als Broadcast Growth Codes (BCGC) bezeichnetes Netzwerkkodierungsverfahren für zuverlässige Datenverteilung entwickelt, welches unter einem breiten Spektrum unterschiedlicher Bedingungen gute Ergebnisse erzielt. Zu diesen Bedingungen gehören zum Beispiel eine hohe Wahrscheinlichkeit von Signalkollisionen, ein beliebiger Grad an Knotenmobilität, eine große Knotenanzahl oder das Auftreten von Knotenausfällen. BCGC benötigen keine komplexe Initialisierung und verwenden nur einfache XOR-Operationen für Kodierung und Dekodierung. Darüber hinaus kann mit der Dekodierung bereits begonnen werden, sobald ein erstes Paket/Codewort empfangen wurde. Evaluationen mit einem eigens implementierten Netzwerksimulator sowie einem realen Testbed haben gezeigt, dass BCGC ermöglichen, Daten trotz eines unzuverlässigen Netzwerks zuverlässig zu erhalten. In Bezug auf Latenz, Durchsatz und Energieverbrauch können BCGC, je nach Bedingungen und verglichenem Verfahren, vergleichbare Ergebnisse wie bestehende Verfahren erzielen oder diese sogar deutlich übertreffen, während sie gleichzeitig robust gegenüber veränderten Bedingungen sind, eine geringe Komplexität der Knoten erlauben sowie eine frühzeitige Dekodierung ermöglichen.

Danksagung

Diese Dissertationsarbeit ist im Rahmen meiner Arbeit als wissenschaftliche Mitarbeiterin am Lehrstuhl für Technische Informatik der Julius-Maximilians-Universität Würzburg sowie während meiner Elternzeit entstanden. Das Forschen und Lehren an diesem Lehrstuhl hat mir große Freude bereitet, daher möchte ich mich hiermit als erstes bei meinem Doktorvater, Prof. Dr. Reiner Kolla, bedanken. Prof. Dr. Kolla hat sich immer sehr viel Zeit für mich genommen, stand mir immer beratend zur Seite und hat mich durch seine langjährige Erfahrung mit wertvollen Tipps und Inspirationen bei meiner Forschung und dem Erstellen dieser Dissertationsarbeit unterstützt.

Außerdem möchte ich mich hiermit bei Prof. Dr. Jörg Nolte bedanken für seine Ideen und Anregungen zu dieser Arbeit während unseren Gesprächen und seine Bereitschaft diese Arbeit zu begutachten.

Des Weiteren möchte ich mich bei den aktuellen und ehemaligen Mitarbeitern, studentischen Hilfskräften und unserer Sekretärin Frau Martina Lehrmann für die gute Zusammenarbeit und schöne Arbeitsatmosphäre am Lehrstuhl bedanken.

Außerdem bedanke ich mich bei meinem Studienkollegen und guten Freund Thomas für das Korrekturlesen und auf Verständlichkeit Überprüfen des für den Leser vermutlich herausforderndsten Kapitels.

Besonders bedanken möchte ich mich auch bei meinen Eltern und meiner Schwester, die mir während meiner gesamten Promotionszeit immer motivierend und unterstützend zur Seite standen.

Ein besonderer Dank gilt außerdem meinem Ehemann Armin, der diese Arbeit bis ins Detail Korrektur gelesen, wertvolle konstruktive Kritik geäußert und mich in jeglicher Hinsicht immer unterstützt und mir den Rücken freigehalten hat.

Mein allergrößter Dank geht jedoch an meine kleine Tochter Leonie, welche ab ihrem dritten Lebensmonat mit mir an dieser Dissertationsarbeit geschrieben hat. Sie hat mir beigebracht effizienter zu arbeiten, hatte aber dennoch unendliche Geduld mit mir und hat sich die ein oder andere Seite dieser Arbeit auch interessiert vorlesen lassen. Tausend Dank, Leonie und Armin, dass ihr mich diese Dissertationsarbeit zu Ende schreiben habt lassen und die wenige Freizeit die ich hatte so wundervoll gestaltet habt!

Contents

Chapter 1

Introduction

Contents

1.1 Motivation

Today, distributed wireless systems can consist of an increasingly larger number of highly integrated, increasingly smaller, communicating and cooperating objects. Technology scaling has enabled more and more computing power, memory capacity, sensors, and communication infrastructure to be built into the smallest devices. Often, these devices are battery-powered, cost-efficient, and no bigger than a few millimeters. Systems consisting of such devices which are able to collect sensor data in a distributed manner over time, process data, and communicate with each other via wireless transmissions are called *Wireless Sensor Networks* (WSNs), see for example [AV10; WB10; For16]. The corresponding devices are called *(sensor) nodes*. Nodes that are assigned to gather original data generated by the other nodes of the network are also called sinks. In general, applications of WSNs range from healthcare [Pat+12], agriculture and environment monitoring [Kwo+12; OS12], industrial monitoring [Lu+05], logistics [Xia+16], military [BQW16], smart grid [Fad+15], to rare event detection [HSR16]. Furthermore, a WSN can also be part of an Internet of Things (IoT) system or be incorporated into a scenario of Industry 4.0. Sensor nodes are often prone to failure and their computing power, energy, and memory are typically limited. Therefore, single nodes or even large parts of the network can fail, for example, due to a depleted battery or due to mechanical defects caused by a deployment in a harsh environment or unreliable components in general. In addition, a network consisting of several hundred or thousand densely deployed nodes can be prone to signal collisions and, thus, packet loss. As a consequence, reliability of certain single transmissions or entire devices can not be guaranteed. Reliable data storage and data dissemination is, therefore, essential in WSNs, especially in the case of a large number of communicating devices. In this thesis, reliability is translated into the following objectives:

- appropriate redundancy depending on the state of the network should be used.
- single points of failure and highly congested areas should be avoided.
- data should be disseminated/collected without relying on the successful reception of specific packets.
- as much original data as possible should be reconstructed and stored in each sink at any considered point in time, even in the beginning, i.e. early decoding should be possible.
- each sink should have to receive as little data as possible in order to be able to gather and reconstruct a respective amount of original data.

In addition to the important aim to ensure reliability, latency should be low, throughput should be high, and energy consumption should be reduced. Furthermore, simplicity of encoding and decoding as well as robustness to changes in conditions are essential here. Therefore, these aims should be achieved in the case of any realistic payload size, unreliable radio transmissions with a small but also with a large amount of packet loss, large or very large networks, irregular initial topology, any pattern or degree of node failure, and regardless of the degree of nodes' mobility.

Reliability here means meeting the stated aims under these challenging diverse conditions. To achieve those aims despite a possibly demanding situation, a certain amount of redundancy is necessary. This can be realized and, thus, reliable data storage and data dissemination can be enabled, for example, by using network coding. Existing approaches often perform well either in highly dynamic or in completely static scenarios, are restricted to a small maximum number of nodes, perform a time-consuming initialization process or preliminary simulations, assume lossless transmissions, require perfect information about all neighboring nodes, do not provide the possibility of early decoding, or require complex calculations, for example.

Therefore, we developed a network coding procedure called Broadcast Growth Codes (BCGC) for reliable data dissemination, which performs well under the diverse conditions described above and is presented in this thesis. In general, BCGC provide a certain form of temporal, spatial, and information redundancy. Furthermore, they address all mentioned aspects of reliability, reduce latency, increase throughput, and reduce energy consumption while not requiring complex initialization, using simple XOR operations for encoding and decoding, and being robust to changes in conditions.

1.2 Thesis Outline

This thesis is structured as follows: First, basic principles of network coding which are relevant for the understanding of the remaining chapters are presented in Chapter 2. This includes relevant terminology, the fundamental procedure of network coding, a classification of network coding, well-known basic examples, and a brief survey of the history of network coding. Furthermore, the differences between source coding, channel coding, and network coding as well as the relation between network coding and routing are also explained in Chapter 2. This introductory chapter on network coding basics closes with an overview of the main benefits of network coding.

Chapter 3 then represents the framework for subsequent considerations and analyses. There, the assumed system setup, a definition of the addressed goals, and a description of possible application scenarios of BCGC, our developed network coding procedure, are presented. One possible choice for a specification of the communication on physical and MAC layer for BCGC is the IEEE Std 802.15.4. As it is used as a basis here, general information on the IEEE Std 802.15.4, some of which are relevant for following parameter decisions, is given in this chapter. However, BCGC are not restricted to this standard. Therefore, alternatives to the IEEE Std 802.15.4 are also listed in this chapter. Finally, the key performance metrics which are evaluated in a later chapter are presented in Chapter 3.

The subsequent Chapter 4 contains related work concerning coding schemes and data dissemination. In this context, NoCoding/Forwarding, which does not use any encoding in the strict sense, is introduced first. After that, the frequently used coding scheme of random linear network coding (RLNC) and the coding procedure called Growth Codes (GC), which forms the basis for BCGC, are presented comprehensively. This chapter closes with an overview of related work concerning network coding based data dissemination, which includes the distribution process and distributed storage.

The main scientific contributions presented in this thesis can be found in the following chapters. In Chapter 5, our developed network coding procedure, BCGC, is introduced in detail. For this purpose, the fundamental procedure of BCGC is presented first. Then, the two important BCGC process parameters, the degree of codewords and the composition of codewords, are comprehensively described. Finally, the differences between BCGC and the network coding procedures GC and RLNC are discussed.

A description of the used simulation environment and the results of a comprehensive simulative evaluation of BCGC can, then, be found in Chapters 6 and 7. Therefore, the implemented radio transmission schemes, MAC protocols, initial topology, mobility models, patterns for node failures, and used energy consumption model are presented first in Chapter 6. Furthermore, the used default configuration of BCGC process parameters and also of relevant parameters relating to simulation settings are explained in Chapter 6.

Chapter 7 then contains the core of the simulation based evaluation of BCGC. This evaluation chapter is divided into two parts: The first part is an evaluation of BCGC process parameters, such as the degree of codewords for transmission and the composition of codewords. The evaluation of the degree of codewords contains an evaluation of different degree functions and an evaluation of different upper bounds for the used codeword degree. The evaluation concerning the composition of codewords consists of an evaluation of an approach with specific symbol selection and an evaluation of an approach with a requested desired symbol. The second part of the evaluation chapter presents an evaluation of BCGC using different simulation settings, including a comparison with NoCoding/Forwarding and the existing network coding procedures RLNC and GC.

After the simulation based evaluation, a brief evaluation of BCGC on a testbed is presented in Chapter 8. Therefore, the used testbed platform including the chosen settings for the testbed based evaluation are described first. Then, the results from the testbed are shown and compared to the corresponding results from the simulator.

This thesis closes with a conclusion and an outlook to future work in Chapter 9.

Chapter 2

Network Coding Basics

Contents

The procedure of Broadcast Growth Codes (BCGC), which was developed within the scope of this thesis, belongs to the field of network coding. Therefore, the basic principles of network coding which are relevant for the understanding of the remaining chapters are presented first. Network coding is an extensive research area, which has influences from many different disciplines. Approaches originate in information theory, graph theory, linear algebra, linear optimization, combinatorics, coding theory, and many more. A comprehensive introduction to this topic can be found in the following books, see [HL08; Yeu08; MS12]. For a compact overview and tutorial on network coding, see [FLW06; DIG10; MNT11] and [CW07; FS+07; FS+08], respectively.

This chapter is structured as follows: First, relevant terminology is introduced in Section 2.1. Then, the fundamental procedure of network coding is explained, and a classification of network coding is presented in Section 2.2.1. This is complemented by a description of well-known basic examples, and a brief survey of the history of network coding in Sections 2.2.2 and 2.2.3. Subsequently, the differences between source coding, channel coding, and network coding are highlighted in Section 2.3. After that, the relation between network coding and routing is explained, and an appropriate routing procedure for the analysis of BCGC is suggested in Section 2.4. This introductory chapter on network coding basics finally closes with an overview of the main benefits of network coding in Section 2.5.

2.1 Relevant Terminology

Here, a network of n homogeneous sensor nodes with the ability to generate, store, and combine data is considered. Data which is generated by a node will be called the node's own *symbol*. It is assumed that symbols of all nodes have the same size. In the case that generated symbols have different sizes, a normalization would be necessary. If there are n nodes in a network, there will be n distinct original symbols. Using wireless communication, a node is able to transmit or receive data packets. Depending on the specifications of the applied procedure, a node may combine, i.e. encode, original symbols and generates a so-called *codeword*. This codeword is subsequently transmitted in a data packet. Each packet contains one codeword as payload. Necessary information for decoding is included in the packet header. A codeword can be *decoded*, and the resulting original symbols are then called *reconstructed*. A corresponding graphical representation of the described process can be seen in Figure 2.1.

The so-called *degree* d of a codeword indicates the number of symbols which have been combined for this codeword. If s_1, s_2, s_3, s_4 are original symbols and encoding is done by bitwise XOR operations of the respective symbols, an example for a codeword is $s_1 \oplus s_3 := (s_1 \wedge \overline{s_3}) \vee (\overline{s_1} \wedge s_3)$[1]. The XOR operation, which is denoted by $\oplus$, corresponds to an add operation in $GF(2)$[2], i.e. a binary addition is performed. The *distance* of a received codeword is the number of symbols which are included in the codeword but have not been reconstructed by the considered node yet. For example, a distance-1-codeword of degree d consists of $d-1$ symbols which have been reconstructed and 1 symbol which has not been reconstructed yet. If a node has already only reconstructed s_1 and s_2, then $s_1 \oplus s_3$ is a distance-1-codeword for this node. If encoding is done by bitwise XOR operations, a codeword can be *reduced* if a symbol is added via XOR which is already contained as XOR is self-inverse. This can be seen by using the subsequent transformation:

$$s_1 \oplus s_1 = (s_1 \wedge \overline{s_1}) \vee (\overline{s_1} \wedge s_1) = 0 \vee 0 = 0.$$

The following example shows the described facts with actual bit values. Let $s_1 = 0101$

[1] $\overline{s} = \neg s$, i.e. negation is used here.

[2] $GF(2)$ denotes the so-called Galois field with 2 elements. For mathematical details on Galois fields, see Section 4.2.4.

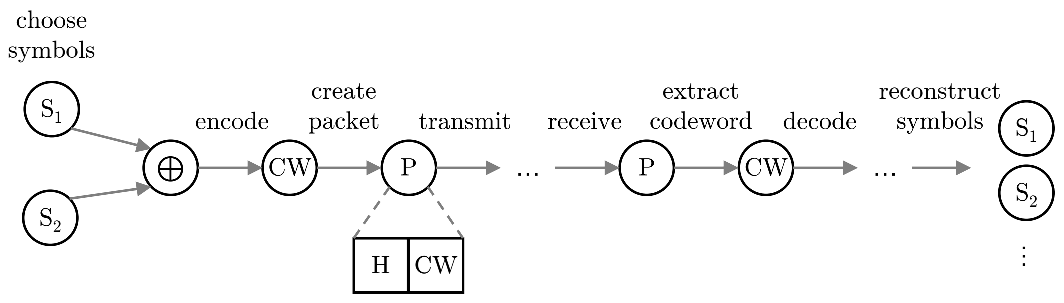


Fig. 2.1: Graphical representation of a network coding based transmission process

and $s_3 = 1100$, then

$$s_1 \oplus s_3 = \begin{array}{r} 0101 \\ \oplus\ 1100 \\ \hline 1001 \end{array}.$$

For decoding, if $s_1 \oplus s_3$ is received and s_1 is already known, s_3 can be reconstructed via:

$$(s_1 \oplus s_3) \oplus s_1 = \begin{array}{r} 1001 \\ \oplus\ 0101 \\ \hline 1100 \end{array} = s_3.$$

A node which generates sensor data is also called a *source node*. Furthermore, each node which is on the path between source node and destination is called an *intermediate node* for this specific transmission. A node assigned to reconstruct and gather original symbols is called a *sink*. There can be one dedicated sink or multiple sinks in a network.

With regard to transmissions, there are *unicast*, *multicast*, and *broadcast* transmissions. If a node overhears a packet which is not addressed to it, the node discards this packet. Due to the broadcast feature of wireless transmissions, this often occurs in wireless scenarios. In the case of a unicast transmission, a node transmits data to exactly one neighboring node. In contrast, a node transmits identical data to multiple neighbors at the same time in the case of a multicast transmission. Broadcast can be considered a special case of multicast where a node sends identical data to all neighboring nodes at the same time. However, these terms should not be confused with the terms unicast session/scenario/network and multicast session/scenario/network as they appear in subsequent network coding examples and also in some related work. A *unicast session/scenario* in network coding literature is a scenario with one source node which aims to transmit data, possibly via several intermediate nodes, to one final destination. A corresponding network is called a *unicast network*. If there are, for example, two independent source nodes, which aim to transmit data to one destination each, it is two independent unicast sessions. In a *multicast session/scenario* in network coding literature, there is one source node and several final destinations. Again, a corresponding network is called a *multicast network*. A *broadcast session/scenario* is a special case of a multicast session/scenario where all nodes of the network are final destinations. In addition, there are the terms *many-to-many broadcast*, *many-to-many communication*, or *multi-source multi-sink problem*. All of these denote a scenario where there are multiple source nodes and multiple sinks. This means, more than one source node generates data which should eventually reach the multiple sinks. A multicast network with multiple independent multicast sessions is, thus, considered. *All-to-all broadcast/communication* is a special case of many-to-many broadcast/communication where each node of the network is a source node and a sink.

The following sections and chapters refer to a packet based network. This means, packets are considered for network transmission, see e.g. [HL08]. Packets are generally sent from one or more source nodes via intermediate nodes to one or more sinks.

After having defined relevant terminology, an introduction to network coding follows.

2.2 Introduction to Network Coding

In this introductory section, the fundamental procedure of network coding is described, and a classification of network coding is presented. According to this classification, the BCGC procedure is classified based on its features. As a complement, well-known basic examples for network coding scenarios are given. Furthermore, the history of network coding is briefly surveyed.

2.2.1 Procedure of Network Coding

With traditional *store-and-forward* or *routing*, data received by an intermediate node is subsequently forwarded to neighboring nodes without modification, see for example [HL08; Yeu08; DIG10; MNT11]. If *network coding* is used in addition to a routing scheme, previously received data is encoded or re-encoded according to the chosen network coding procedure before further transmission. This means, received packets are not just forwarded by intermediate nodes, as they are in the case of routing without network coding, but stored, modified, and then transmitted. A more detailed description of the relation between network coding and traditional routing can be found in Section 2.4. The XOR operation is the most simple coding function which can be used for network coding. Furthermore, encoding can be performed at a source node and at intermediate nodes. However, network coding should not be confused with source coding or channel coding. Therefore, a distinction between network coding, source coding, and channel coding follows in Section 2.3. As already mentioned in Section 2.1, encoded data, i.e. the created codeword, is transmitted as the payload of a packet if network coding is applied. To be able to decode encoded data again, all relevant information for decoding has to be added to the packet header.

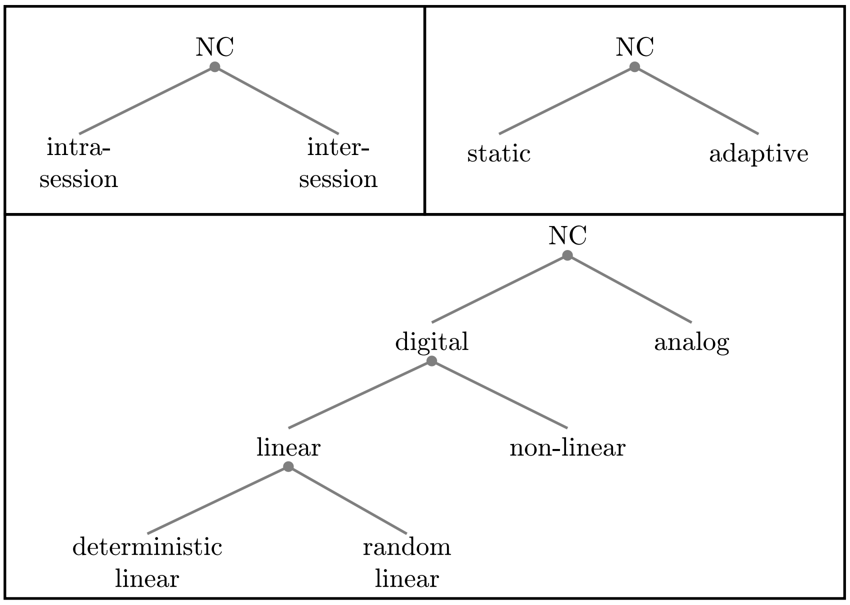


Fig. 2.2: Classification of network coding procedures

There are different approaches to classify network coding procedures. Some of them are very special or rarely used in literature. The most common categories are presented in the following, and a corresponding overview is shown in Figure 2.2.

A first option to classify network coding procedures is the distinction between *inter-session* and *intra-session* network coding. Corresponding definitions can also be found in standard literature, such as [HL08; KKR12]. In the case of inter-session network coding, data from different sessions, i.e. from different source-destination(s)-tuples, can be combined. These different sessions can intersect at an intermediate node, which then has the possibility to encode data. In the case of inter-session network coding, intermediate nodes often first decode, recompose, and then encode packets instead of simply re-encoding them [KKR12]. Here, an operation which is commonly used to combine data is XOR. Inter-session network coding is typically applied in multiple unicast scenarios or multiple multicast scenarios, where usually different sessions overlap. A graphical representation of an example of inter-session network coding is depicted in Figure 2.3. Inter-session network coding is used, among other things, to reduce the number of required transmissions and thereby increases throughput. Instead of transmitting data from different sessions one after the other, inter-session network coding combines, i.e. encodes, data from overlapping sessions. Thus, data of different sessions is transmitted simultaneously, for example, through a bottleneck. For decoding beyond the bottleneck, the fact that data moves through the network on various paths

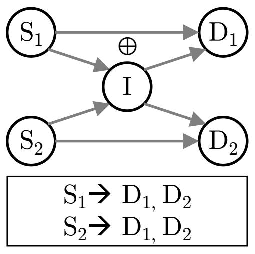


Fig. 2.3: Example of inter-session network coding scenario

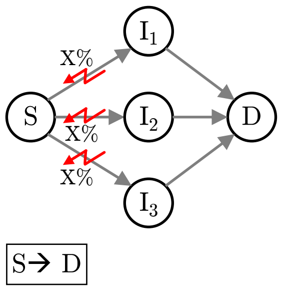


Fig. 2.4: Example of intra-session network coding scenario

is exploited. The node(s) on the other side of the bottleneck may have received a certain amount of the data which had been combined for the packet, either before or simultaneously via a different path. In the case of an encoding with XOR, the node can then use this data to try to decode the current packet and possibly reconstruct a new symbol. Prominent examples for inter-session network coding are *COPE* [Kat+05; Kat+06; Kat+08], the XOR based procedure of [WCK+05], and the random linear network coding[3] (RLNC) based procedure of [FWL08].

For intra-session network coding, only data from the same session, i.e. from the same source-destination(s)-tuples, can be coded together. This means, data which has the same destination(s) can be coded together, and decoding is typically performed at the destination or destinations [KKR12]. Often, intra-session network coding is RLNC based and re-encoding is used, instead of decoding with subsequent encoding. Intra-session network coding is usually applied in single lossy unicast scenarios or single multicast scenarios, where data is transmitted via several hops. In single multicast scenarios, links between nodes can be lossy, but do not have to be lossy to benefit from network coding. A graphical illustration of an example of intra-session network coding can be seen in Figure 2.4. Intra-session network coding is used, among other things, to increase reliability without the need for feedback messages or other coordination traffic. For

[3]See Section 4.2 for more details on RLNC.

this purpose, intermediate nodes send combinations of previously received packets and, thus, compensate if individual packets got lost or will get lost. For example, in the case of RLNC, all data can be reconstructed by the destination(s) as soon as enough linearly independent packets have been received. It is not necessary to retransmit specific individual packets that did not arrive at a destination. Instead, by combining previously received data in each intermediate node, packets are useful for multiple destinations of the considered session. It does not matter which packets were lost on the path and which exact packets arrived at the respective destination. Thus, erasures, which may occur at each hop, are compensated. Intra-session network coding is used, for example, for the very first network coding approach [Ahl+00], for the famous linear network coding propositions of [LYC03], for the first RLNC results [Ho+03a], or for the RLNC based coding procedure called *MORE* [Cha+07].

However, there are also network coding approaches which cannot be assigned to pure inter-session or pure intra-session network coding, but are a combination of both. Examples are [SMR11; Hua+14].

A second option to classify network coding procedures is the distinction between *static* and *adaptive* network coding. For static network coding, encoding is executed independently of, for example, the current state of the network, the state of the sinks, or the environment of the nodes which are currently performing the encoding. The exact coding procedure can be completely predetermined before the algorithm starts and before any data is sent through the network. For example, for linear network coding based approaches, the coefficients for each node can already be chosen in advance. In the case of adaptive network coding, encoding is adapted to the current situation. For example, for an encoding which is currently being executed by a node, it is possible to take into account how much data has already been reconstructed by the sinks. Certain parameters of the respective coding function are determined during the course of the algorithm.

A third option to classify network coding procedures is the distinction between *digital* network coding and *analog* network coding. Digital network coding can be further divided into *linear* and *non-linear* procedures. Depending on the type of coefficient determination, there is *random* linear network coding and *deterministic* linear network coding. For digital network coding, encoding is performed on packet level. The payload of a packet in bit form is considered as a codeword, and information from the packet header is used for decoding. In the case of a linear network coding procedure, a codeword is the result of a linear combination of individual symbols or codewords. It should be noted that the XOR operation is a special form of a linear combination. The coefficients used for a linear combination can be stored in the packet header to enable decoding. Coefficients can be chosen randomly. In this case, the corresponding procedure is called random linear network coding. Alternatively, coefficients can be chosen specifically. The associated procedure is then called deterministic linear network coding. Details on linear network coding and especially on random linear network coding are described in Section 4.2. If encoding is done in a different way than by linear combination and if codewords also cannot be generated by linear combination, the procedure is called non-linear network coding. Non-linear network coding is addressed, for example, in [LL04a; DFZ05; KTT09].

For analog network coding, encoding is performed on analog physical signals instead of being based on packets. Therefore, analog network coding is also called physical-layer network coding. In the case that two nodes transmit simultaneously and the packets collide, these packets are added on signal level. If a receiver knows the content of one of the collided packets, it can extract the corresponding signal from the interfered signals and obtains the remaining signal. Thus, interference is used to improve throughput. These and other details can be found, for example, in [ZLL06; KGK07; KKR12]. Analog network coding was first presented in [ZLL06]. A first implementation of analog network coding can be found in [KGK07]. In the following sections and chapters, however, packet based networks will be addressed. This means, only packet based approaches will be considered and reasoning will be done on packet level. For this reason, analog network coding will not be dealt with in subsequent sections.

BCGC, which are presented here, can be classified as inter-session network coding. Intermediate nodes can combine data regardless of which session the data originates from. Furthermore, BCGC belong to the category of adaptive network codes as encoding is adapted to the current situation. The procedure of BCGC is a digital network coding procedure and represents a certain form of deterministic linear network codes.

2.2.2 Basic Examples of Network Coding Scenarios

In the following, three basic examples are presented in which the application of network coding is beneficial with regard to the required number of transmissions. In general, there are unicast and multicast sessions, and transmissions can be lossless or lossy. Here, a single unicast/multicast session or multiple independent unicast/multicast sessions are considered. The application of network coding can bring advantages only in the case of a single lossy unicast session, multiple independent unicast sessions, a single multicast session, or multiple independent multicast sessions. These multicast and multiple unicast sessions can be lossless or lossy. An overview of all options and in which cases network coding can be beneficial is shown in Table 2.1. The following three examples should cover the mentioned options and represent basic examples of network coding scenarios. Since the process and the advantages of a network coding procedure do not differ fundamentally between multiple independent lossless and lossy unicast sessions or between a single and multiple multicast sessions, only one example each is presented in both cases. Thus, the first example considers a single lossy unicast session, the second example considers two independent lossless unicast sessions, and the third example considers a single lossless multicast session. Looking at the first two examples, the difference between intra-session and inter-session network coding can also be seen.

	lossless	lossy
single unicast session		+
multiple independent unicast sessions	+	+
single multicast session	+	+
multiple independent multicast sessions	+	+

Table 2.1: Overview of cases in which network coding can be beneficial

In all examples, an abstract approach is used and only the payload of the packets is taken into account. Furthermore, time is slotted, and it is assumed that only one packet can be transmitted over each link per time slot.

The first example, which considers a single lossy unicast session, is a chain of nodes connected via point-to-point links. This example is depicted in Figure 2.5 and shows how a network coding procedure combines data from within a session, i.e. intra-session network coding is performed. Nodes are able to transmit data according to the directed links in the finite graph in Figure 2.5. There is one source node S, an intermediate node I, and one destination node D. Node I should be able to receive data from the source node S, and the destination D should be able to receive data from I. No direct communication between source node S and destination D is possible. Within one time slot, it is assumed that a node can either send or receive a packet. However, links are assumed to be lossy. The source node S now wants to transmit its data a and b via the intermediate node I to the destination D. This type of scenario is considered as a single unicast session in network coding terminology as there is one source node which aims to transmit data to one final destination. Without network coding, it is necessary to transmit data a and b separately from node I to D. It is assumed that without network coding, node I transmits received data alternately. If, for example, the transmission does not arrive at D in the fourth time slot, at least two further transmissions are necessary without network coding until all data has arrived. This means, destination D has all data after six transmissions at the earliest. With network coding, it is possible to send an XOR combination of a and b in the fifth time slot. Thus, D can decode b from this combination and has all data after only one additional transmission. With network coding, one transmission and one time slot less is required than without network coding. Hence, this example shows how in a single lossy unicast session the required number of transmissions can be reduced by using network coding. This first scenario, is a simple example of intra-session network coding. A basic example of inter-session network coding now follows.

The second example, which addresses two independent lossless unicast sessions, is a network of three nodes which transmit data via broadcast transmissions, as depicted in Figure 2.6. This example shows how network coding combines data from different sessions, i.e. inter-session network coding is performed. Nodes are able to transmit

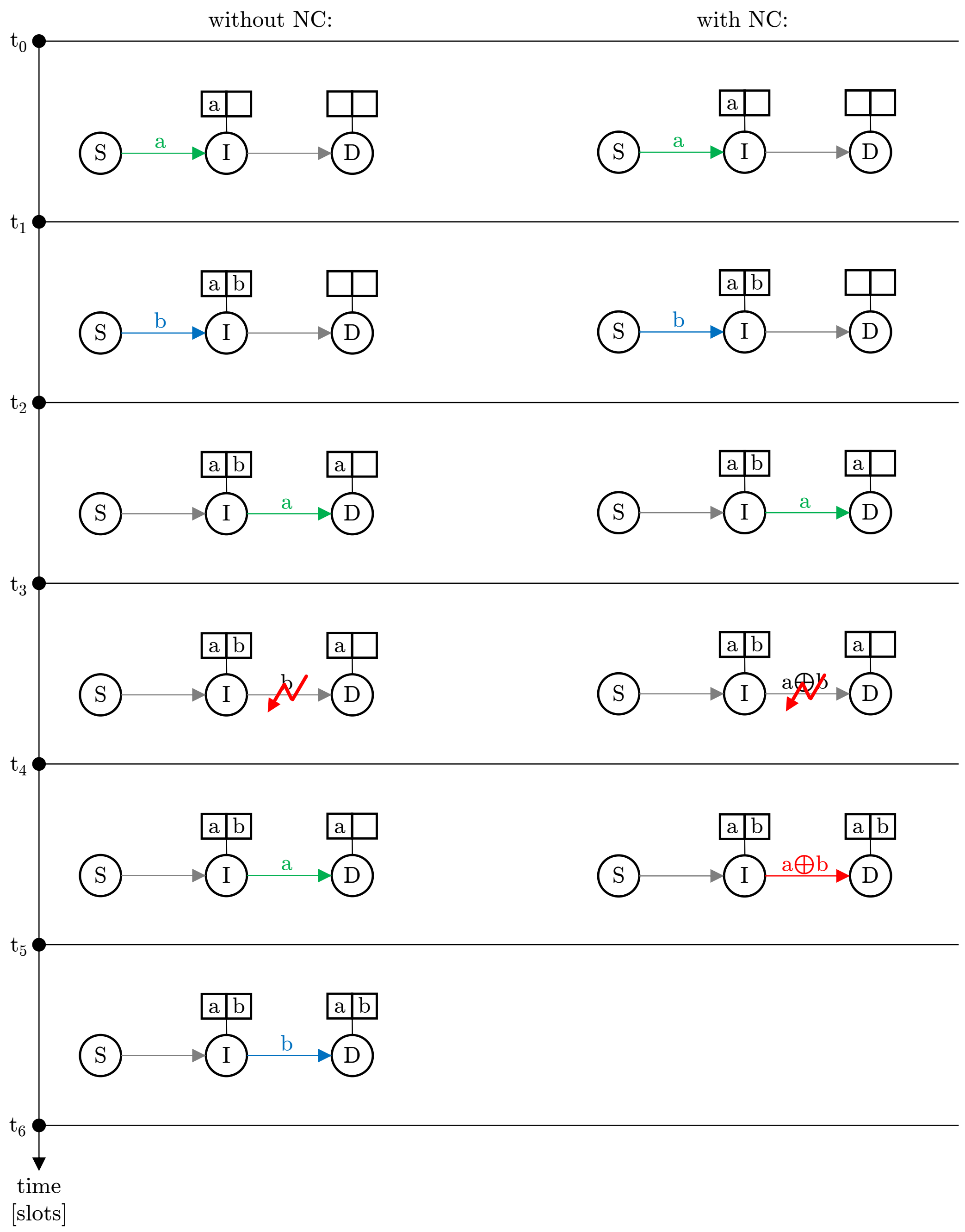


Fig. 2.5: Example of network coding advantage for a lossy unicast session

data reliably according to the directed links in the finite graph. This means, lossless, reliable links are considered. In [WCK+05], this example was first used in the context of network coding [KKR12]. Let nodes A and B be able to receive data reliably from node I. Furthermore, node I should be able to receive data reliably from nodes A and B. No direct communication between nodes A and B shall be possible. Within one time slot, it is assumed that a node can either send or receive a packet. There can only be one broadcast transmission per time slot in this considered network if collisions shall be avoided. This means, nodes A and B cannot transmit data simultaneously to node I. Node A now wants to send its data a via the intermediate node I to node B, and B wants to send its data b to node A. Hence, there are two independent source nodes which aim to transmit data to one final destination each. This type of scenario is considered as two unicast sessions in network coding terminology. After node I has received data a from node A and data b from node B, without network coding, it can transmit w.l.o.g. data a in a third time slot. Subsequently, node I has to transmit data b in a fourth time slot so that finally both nodes, A and B, have the desired data. Now, consider the case that network coding is applied and the XOR operation is used as coding function. With network coding, the example differs from the variant without network coding from the third time slot onward. In the case of network coding, node I can now combine the previously received data a and b and transmit all data in only one packet as $a \oplus b$ to A and B. Since XOR is self-inverse, nodes A and B can use their own data to extract their respective missing data from the package. Thus, without network coding, four transmissions and four time slots are necessary in the considered example and only three with network coding. This example shows how in two independent lossless unicast sessions the required number of transmissions can be reduced by using network coding.

The third example, which represents a single lossless multicast session, considers the wired, so-called butterfly network for multicast. In the context of network coding, it was first used for illustration in [Ahl+00]. Since then, the butterfly example has appeared in numerous introductions to network coding, such as in [FS+07; HL08; Ksc12]. Besides the original version, there are also different variants of the butterfly network. For example, a modification with two source nodes instead of one, a butterfly with wireless links, or a combination of these two which consists of only three nodes in total [HL08; Yeu08]. However, the original, wired butterfly will be considered in the following as it is the most famous example in the context of network coding. A graphical representation of this example is depicted in Figure 2.7. Here, nodes are able to transmit data reliably according to the directed links in the finite graph. There is one source node S, which generates and transmits two distinct packets a and b via intermediate nodes to the destinations D_1 and D_2. This type of scenario is considered as a multicast session/scenario in network coding terminology as there is one source node and several final destinations. Figure 2.7 shows how data flows through the network over time for both the case without network coding and with network coding. It is assumed that only one packet can be transmitted over each link per time slot. Without network coding,

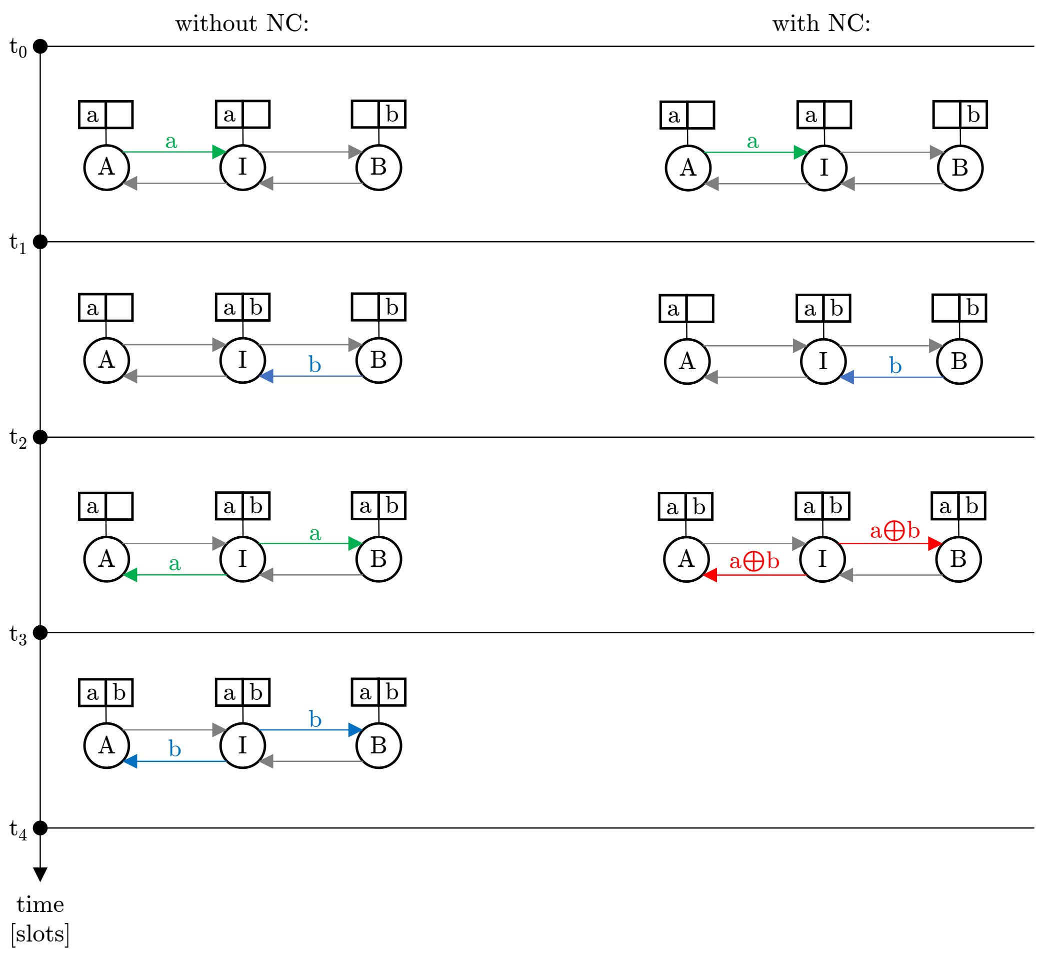


Fig. 2.6: Example of network coding advantage for two independent lossless unicast sessions

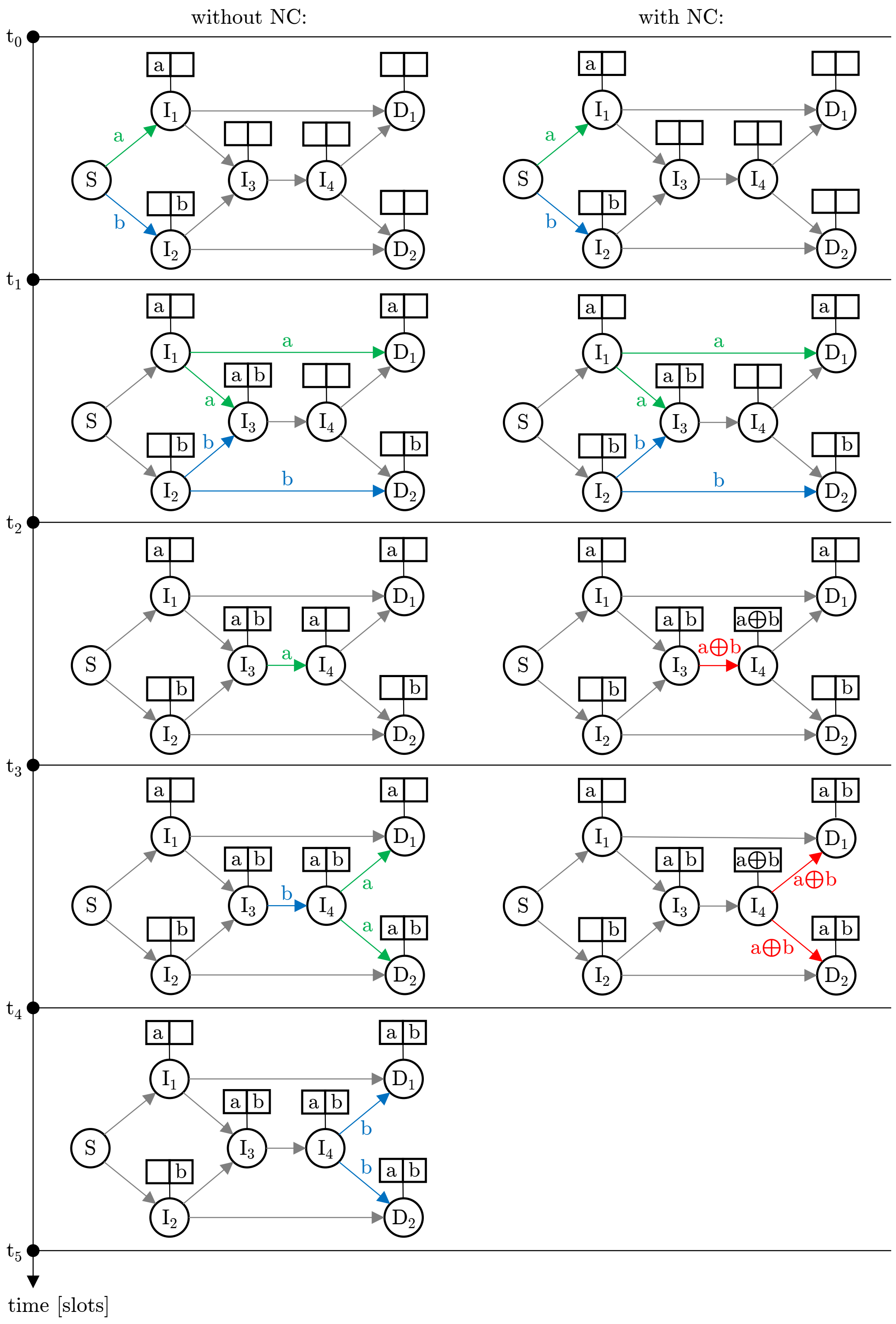


Fig. 2.7: Example of network coding advantage for a lossless multicast session

there is a bottleneck between the intermediate nodes I_3 and I_4. Either packet a or packet b can be transmitted within the third time slot. Without loss of generality, it is assumed that node I_3 chooses packet a for transmission first and transmits packet b in the subsequent time slot. Therefore, packet a is received by D_1 and D_2 after four time slots, and packet b is only received by destination D_2. After five time slots, packet b is finally also received by D_1, and both destinations have all data. In the case of network coding, the intermediate node I_3 can calculate an XOR combination of packets a and b. Having received a and $a \oplus b$, destination D_1 can reconstruct packet b. The same applies to destination D_2 with b and $a \oplus b$. After already four time slots both destinations have all data. Hence, with network coding one transmission less is necessary than without network coding. The number of transmissions which is required to transfer all data from the source node to the destinations is reduced. Thus, throughput is increased by using network coding in the considered multicast scenario.

The three presented examples show scenarios in which the application of network coding is beneficial referring to the required number of transmissions and the throughput. More details about the advantages of network coding can be found in Section 2.5.

2.2.3 History of Network Coding

The notion of network coding can originally be ascribed to Ahlswede et al. in [Ahl+00]. There, a wired point-to-point communication network with only one source node is considered. Information is transmitted from this single source node via intermediate nodes to several destination nodes. Thus, a multicast scenario/network is considered, as defined in Section 2.1. Ahlswede et al. generalize the *max-flow min-cut theorem*, which is proved in [FF56], [DF56], and [EFS56], for network information flow [Yeu11]. According to [EFS56; FS+07; Yeu08], the principle idea of the max-flow min-cut theorem is the following: Consider a connected, finite, directed graph $G = (V, E)$, consisting of a set of nodes V and a set of edges $E \subset (V \times V)$. Assume there is one source node and one or several sinks in V. A *cut* in this graph G is a set of edges $E' \subset E$ without which the source node and at least one of the sinks would no longer be connected. The *value of a cut* is the sum of the respective edges' capacities. However, only the capacities of edges which are directed toward the sink are considered. Then, the *min-cut* of G is the smallest value among all cuts' values. Furthermore, the *max-flow* is the maximum amount of data which can be transmitted from the source node to each sink per time slot, or unit of time in general. The max-flow min-cut theorem then states that the max-flow equals the min-cut. As the max-flow/min-cut value is an upper bound, it is also referred to as the *max-flow bound*, *min-cut bound*, *maximum capacity bound*, or *multicast capacity*. One main contribution of Ahlswede et al. in [Ahl+00] is to show that this max-flow bound cannot generally be reached by traditional routing in a multicast scenario but can be reached by additionally employing network coding. Thus, Ahlswede et al. demonstrate that, for a single multicast session, it is not optimal if data is only forwarded to neighboring nodes and routed to the destinations. Instead, data should be encoded at intermediate nodes. They conclude that using network coding saves bandwidth and improves throughput.

Li et al. investigate in [LYC03] how fast information can be spread from one source node to all nodes in a multicast network using network coding. They show that linear network coding is sufficient to achieve the optimal multicast rate, i.e. the max-flow bound, and even the individual max-flow bound at each node. This is also proved in [KM02; KM03] by using an algebraic approach with matrices instead of a vector space framework as in [LYC03]. Furthermore, Koetter and Médard extend the results of Li et al. to arbitrary wireline networks in [KM02; KM03]. In [KM03], the authors continue and develop the ideas of [KM02], which refers to a preprint of [LYC03]. Using an algebraic approach, the authors of [KM03] show that encoding and decoding can be performed in polynomial time in the case of linear network codes. This is also proved independently in [JCJ03] and [SET03], where additionally a polynomial time algorithm is developed. This algorithm determines the functions that are used by the nodes for encoding so that the max-flow bound is achieved in a multicast network with one source node. This means, it determines the coefficients at each node for linear network coding so that the sinks are able to receive and decode data at a rate according to the max-flow bound. In [Ho+03a; Ho+06], this is modified by Ho et al. to random linear network coding (RLNC), which asymptotically achieves the max-flow bound in a multi-source multicast network. There, coefficients are chosen randomly by each node independently, which allows a distributed approach. Further details on RLNC can be found in Section 4.2. An early practical implementation and simulation of network coding in real packet based networks with e.g. delays, losses, and node failures can be found in [CWJ03]. The presented approach does not need any knowledge about the topology or encoding and decoding functions, and it does not rely on synchronous transmissions. In [Lun+04], Lun et al. present a random linear network coding approach. They show that it can reach the maximum throughput rate for single unicast or single multicast sessions in a lossy wireline or wireless network if packets arrive at an appropriate rate. Reaching the maximum throughput rate means achieving the max-flow bound. The results of [Lun+04] are extended in [Lun+05a] to more general conditions regarding the packets' arrival process. The throughput advantage of network coding over routing in a single multicast scenario is demonstrated in [Ahl+00]. The corresponding statement was also shown for a scenario with multiple unicast sessions in [LL04b; WCK+05; HK05], for example. The advantages of wireless networks are considered in the context of network coding, for example, by Deb et al. in [Deb+05]. In [Kat+05], Katti et al. present a first implementation of network coding in wireless networks. The developed opportunistic network coding approach called *COPE* supports multiple unicast sessions. Applications for network coding in wireless networks are analyzed and presented by Fragouli et al. in [FWL06]. There, the focus is on the benefits of network coding in terms of energy efficiency. Since then, a wide variety of research has been done in the field of network coding. Different performance metrics and many specific applications have been studied.

After having introduced network coding in general, the differences between network coding, source coding, and channel coding will be presented in the next section.

2.3 Distinction between Source Coding, Channel Coding, and Network Coding

When transmitting data over a transmission channel from source to sink, techniques such as source coding, channel coding, and network coding can be applied. Each of these techniques has different objectives and can be considered orthogonal to the other two approaches. Hence, a combination of the three techniques is also possible. Source coding and channel coding are established methods, which have been studied and used for many years.

The purpose of *source coding* is that a source node has to transmit less data and still all original data of this source node can be reconstructed by the sink(s) after reception. With source coding, data is compressed in a source node, e.g. by removing redundancies in the raw data. For encoding, only data which was generated by this source node itself is used. Furthermore, a source node is the only type of node which performs source coding. After having received a packet, intermediate nodes only forward this packet to the respective destination(s). Decoding is performed exclusively by the sink(s). Depending on the prerequisites, lossless or lossy source coding can be used, i.e. complete recovery of the original data by a sink is generally possible or not possible. In the following, the focus will not be on source coding, since this approach is orthogonal to the network coding procedure presented here, and can be applied additionally. For more detailed information on source coding, please refer to [Skl+01; AM12; Gra12], for example.

In contrast to source coding, *channel coding* systematically adds useful redundancy in order to compensate for transmission errors in the transmission channel. These can be bit errors or so-called *erasures*, i.e. entire packets are lost. The purpose of channel coding is to improve the reliability with which data from a source node reaches the sink(s) in the case of an error-prone or lossy channel. Like source coding, channel coding is performed exclusively by the source node(s). For encoding, only data which was generated by the respective source node is used. Again, intermediate nodes only forward encoded packets, and decoding takes place in the sink(s). General information on channel coding can be found, for example, in [Skl+01; RL09; Fri13; Bos13].

A graphical representation of the classification of channel coding procedures, which are presented in the following, can be seen in Figure 2.8. In general, it is possible to choose *Reverse Error Correction* (REC), *Forward Error Correction* (FEC), or a hybrid approach from both as channel coding. REC procedures contain procedures for error detection, thus, so-called *error-detecting codes* are employed. If an error is detected, a new transmission of the respective packet can be requested. Here, the procedure

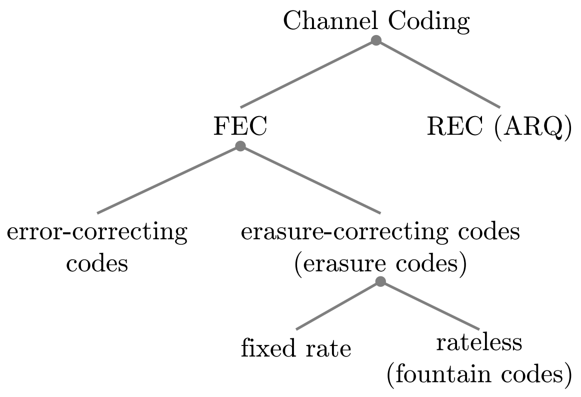

Channel Coding
FEC
REC (ARQ)
error-correcting codes
erasure-correcting codes (erasure codes)
fixed rate
rateless (fountain codes)

Fig. 2.8: Classification of channel coding techniques

of REC is therefore also called *Automatic Repeat reQuest* (ARQ). Depending on the scenario, retransmission can be requested per link or on an end-to-end basis. Examples of error-detecting procedures that can be used in the context of ARQ are procedures which use checksums, such as cyclic redundancy check (CRC) codes, or which add parity bits to detect errors. Details on the respective procedures and further ARQ strategies can be found, for example, in [BGH92; LW03; TW11]. In the case of lossy wireless links or collision-prone systems, ARQ may not be optimal as feedback messages can also be lost. ARQ may also be not convenient in a multicast scenario as there can be a lot of feedback messages in the network which are of interest for only a few nodes.

An alternative to REC/ARQ is FEC. However, a combination of the two approaches, ARQ and FEC, is also possible. For FEC, a source node adds useful redundant information to its transmissions. According to the considered layer or application, redundancy can be added at different levels, such as at bit level or at packet level. Thus, redundancy can be added within each packet only referring to this packet or across packets. Here, across packets means that different original symbols, which without encoding would have been packed into different packets as payload, can be combined in packets for transmission. Then, there are procedures which only transmit this kind of encoded packets and procedures which transmit non-encoded packets plus additional encoded packets. For the latter procedures, the term encoded packets is used in the following for all transmitted packets due to the more compact formulation. By adding redundancy as described, transmission errors which are introduced by the channel can be compensated at the respective sink(s). With FEC, no request for retransmission is sent. Instead, the procedure is only information redundancy based.

The considered transmission errors which should be compensated with FEC can be bit errors within a packet or erasures, i.e. whole packets that got lost. Accordingly, FEC procedures are distinguished between error-correcting codes and erasure-correcting codes. There are multiple schemes of *error-correcting codes*, which can correct bit errors.

Some of the best-known examples of error-correcting codes are Hamming codes, Reed-Solomon codes, Bose-Chaudhuri-Hocquenghem (BCH) codes, or Low-Density Parity Check (LDPC) codes. More detailed information about these coding schemes can be found in literature, such as in [MS77; CC81; TW11; Bos13].

In the following, it is assumed that error detection was performed in advance and packets which contain bit errors were corrected or dropped. Therefore, only non-corrupted packets or erasures, i.e. lost packets, are considered here. The focus in the following is, thus, on so-called *erasure-correcting codes* or *erasure codes*. Erasure codes are a special form of FEC which compensate erasures, i.e. lost packets. To this end, a source node adds redundancy to its transmissions.

As already described above, redundancy can be included by combining different original symbols in a packet for transmission, which without encoding would have been packed into different packets. Depending on the procedure, only such encoded packets with this kind of 'mixed' payload or non-encoded packets and additionally a certain amount of such encoded packets can be sent. In the following, there will not be a distinction between non-encoded and encoded packets in this context. All transmitted packets will be called encoded packets. Let k be the number of original symbols which were generated by a considered source node. This means, k is the number of original data packets which would be sent by this source node without encoding according to the definitions in Section 2.1. Furthermore, let m be the number of encoded packets sent by the source node, i.e. m is the number of generated codewords. After m encoded packets have been sent, some may be lost due to erasures so that only k' encoded packets arrive at the sink(s). The respective sink will then attempt to reconstruct the original k symbols from the k' received encoded packets. Once a certain amount of encoded packets has been received by a sink, this sink is able to reconstruct all original symbols which were generated by the source node. Thus, erasures can be compensated as it is not necessary to receive all transmitted encoded packets in order to recover all original symbols.

For erasure codes, there are *fixed rate* and *rateless codes*. Here, the so-called *code rate* denotes the ratio k/m. With fixed rate erasure codes, only a predefined fixed number of codewords can be generated from a given amount of original symbols. This means that the predefined code rate remains constant. The already mentioned Hamming codes, Reed-Solomon codes, or BCH codes, for example, can be employed for fixed rate erasure codes. Further well-known fixed rate erasure codes are tornado codes [Lub+01]. By contrast, with rateless erasure codes, an infinite number of encoded packets can be generated from a given amount of original data. The code rate, thus, varies over time. With rateless erasure codes, an endless sequence of encoded packets can be generated, just as a fountain which emits water endlessly. For this reason, end-to-end rateless erasure codes are also called *fountain codes*. Examples of fountain codes are Luby Transform (LT) codes [Lub02], Online codes [May02], or Raptor codes [Sho06]. These and further details on erasure codes, fountain codes, LT codes, Raptor codes etc. can be found e.g. in [Riz97; Bye+98; Mit04; Mac05; RL09].

In the case of *network coding*, encoding is performed not only by source nodes but also by intermediate nodes, unlike source coding or channel coding. Network coding is, thus, a distributed coding scheme. Furthermore, an intermediate node can encode data of different sources with each other with inter-session network coding. In contrast to source coding, network coding does not compress data, but adds redundancy, as in channel coding. Since network coding has this essential common feature with channel coding, different channel coding variants are distinguished from network coding in the following. The order of the following comparison corresponds to the order which has already been used for the classification of channel coding.

First, ARQ is considered. There are network coding approaches which use some form of feedback like ARQ. These procedures can rather be considered as a combination of network coding and ARQ. Examples can be found in [Ngu+08; LLL08; KWH11]. This mentioned feedback can be used for coordination, for example, in order to decide which node should finally forward received data, which data should be combined for retransmission so that as many neighbors as possible benefit, or when to stop transmitting. Second, unlike the channel coding procedure of error-correcting codes, network coding does not correct bit errors. Again, a combination of network coding and error-correcting codes is possible. Third, erasure codes are considered. Network coding, especially intra-flow network coding, is most comparable to the channel coding procedure of erasure codes. Like these, network coding can be used to compensate for erasures, among other things. However, network coding is also used to increase throughput, which is apparent in the case of a multicast scenario or bottlenecks, for example. Network coding can, in general, achieve a higher throughput than rateless erasure codes [Lun+04]. Furthermore, since network coding adds redundancy at each intermediate node, unlike end-to-end erasure codes, the probability of failure does not increase with each hop.

To summarize, source coding, channel coding, and network coding are three completely different approaches, each with its own prerequisites and intentions. However, it is possible to combine the single techniques with each other. Approaches for joint channel coding and network coding, or for a combination of source coding and network coding are presented, for example, in [Tho08; Guo+12; Yan+14; Hua+14; Ho+04; Wu+05].

In this section, the differences between source coding, channel coding, and network coding have been highlighted. The relation between network coding and routing will be explained in the subsequent section. Furthermore, an appropriate routing procedure for the analysis of BCGC will be discussed.

2.4 Relation between Network Coding and Routing

Network coding and routing can be considered as two orthogonal approaches. An underlying routing scheme can be complemented by the application of network coding. Here, simple forwarding procedures where data is transmitted, for example, to one or multiple random neighbors, to all neighbors, or to a neighboring node with a certain probability are also regarded as routing procedures. In the following, a distinction is, therefore, made between network coding added to an underlying routing scheme, which for the sake of brevity is also only referred to as network coding, and routing without network coding.

In the case of *network coding*, data is modified by intermediate nodes before retransmission. By contrast, with traditional *store-and-forward* or *routing* without network coding, data remains unchanged at intermediate nodes. In the case of routing, data is stored at the nodes and subsequently forwarded toward the respective destination. Depending on the concrete routing mechanism, a path from the considered intermediate node to the sink might be determined, and packets can then be routed along it. Processing of data is only performed by the respective sink. A corresponding definition of routing is also used in common literature, such as [KW07; Yeu08; MWM09; DIG10].

Unlike other wireless networks, WSNs have particular demands, so routing procedures had to be adapted or newly developed for WSNs [AK04; KW07; FRS09; PNV13]. In WSNs, the topology can change continuously due to usually unreliable wireless links or as nodes may fail, become inactive, or be added to the network later. Furthermore, nodes often do not know the topology of the network in advance. In addition, nodes usually have restricted resources concerning computational power and energy, for example. Overall, WSNs are very application-specific, and each scenario has its own constraints. Routing methods that can be used specifically for WSNs will, therefore, be considered in the following.

In general, there are many different routing protocols for WSNs. They differ in the scheme according to which data is forwarded. An intermediate node has to decide to which neighboring node data should be forwarded so that it finally reaches its destination, i.e. the sink(s). Often, routing tables are used for this decision [KW07]. A routing table is then set up and maintained as part of the routing procedure. The routing table usually contains paths to other nodes and their respective costs, which depend on the criteria that have to be considered. Costs can be, for example, the number of hops to the respective node or the energy consumption for the respective route. However, there are also some routing procedures that operate without a routing table or specific routing scheme. Basic examples are *flooding* and *gossiping*, which can be used for routing at WSNs, but have not been developed specifically for WSNs. With flooding and gossiping, packets propagate randomly through the network and are not routed along a previously calculated path. No specific routing technique is used. As declared at the beginning of this section, flooding and gossiping are nevertheless assigned to routing procedures here, as they specify how data is forwarded in the network.

Flooding is the simplest form of forwarding data. A node transmits its packet to all direct neighbors via broadcast. These neighbors store the received packet and forward it to all of their respective neighbors subsequently. This process is repeated until the packet reaches its destination or other limits, such as the maximum number of hops, are reached. So, no routing table is used, and there is no need to determine to which neighbors a packet should be forwarded. A description of flooding can also be found in [LKW09; SSS10; TW11; ZNK12], for example. Another basic form of forwarding data is gossiping, see for example [HKB99; KW07; Sha+09; Dim+10b]. In contrast to flooding, received packets are stored and subsequently forwarded to only one randomly selected neighbor, instead of being forwarded to all neighbors. This is repeated until the respective packet has arrived at its destination or until a predefined limit is reached. Gossiping can be considered as a form of randomized flooding. Flooding and gossiping are two extreme types of forwarding which do not use a specific routing scheme or routing table. Approaches that lie between flooding and gossiping are, for example, controlled flooding, probabilistic routing, or opportunistic routing. In the case of controlled flooding, a packet is transmitted to a subset of neighbors via a topology-control algorithm [KW07]. If probabilistic routing is used, a packet is forwarded only with a certain probability, see e.g. [HHL02; WL05]. With opportunistic routing [BM04], each node which overhears a transmission forwards the received packet if it is closer to the destination. Further details about variants of flooding and gossiping in WSNs can be found, for example, in [KW07].

There is a considerable number of routing protocols for WSNs and also plenty of different ways to classify them. Often, the categories are not disjoint, and routing procedures can be assigned to several categories. A frequently used classification is proposed in [PNV13], which extends the well-known categorization of [AK04]. An overview of the classification of routing protocols in WSNs which is presented in the following is depicted in Figure 2.9. In [PNV13], routing protocols are distinguished by (1) network structure, (2) communication model, (3) whether a procedure is topology-based, and (4) whether it is a reliable routing procedure. These four classes can be further subdivided. First, based on their network structure, routing protocols can be classified as flat or hierarchical. Second, routing procedures can be categorized according to their communication model as query-based, coherent-based, and negotiation-based. Third, topology-based routing protocols can be location-based and mobile agent-based. Fourth, reliable routing schemes can be classified into multipath-based and QoS-based. Furthermore, routing procedures can be differentiated between pro-active, re-active, and a hybrid version of these two. This classification is made according to the time of route calculation and routing table creation. Pro-active routing protocols are also called table-driven, and re-active routing methods are also referred to as demand-driven or source-initiated in literature. Moreover, routing protocols can be data-centric or address-centric [SSS10], depending on how data is requested by a sink node. A description of

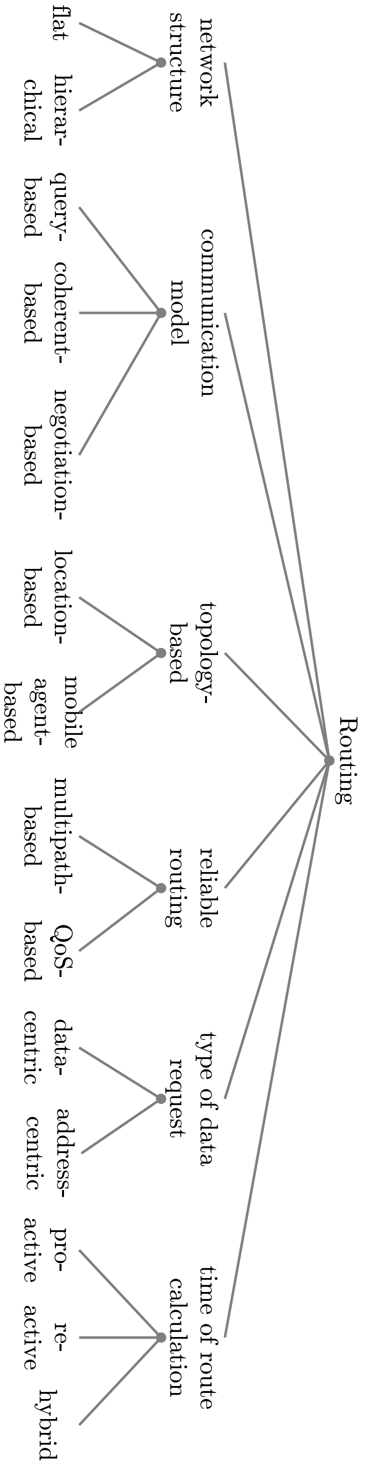


Fig. 2.9: Classification of routing protocols for WSNs

the single categories is omitted here but can be found, for example, in [SSS10; PNV13]. For more details on classification and routing procedures for WSNs in general, see e.g. [AK04; SSS10; Li+11; PNV13].

The additional application of network coding provides, especially in wireless networks, many advantages over traditional routing without network coding, see Section 2.5. For this reason, it is advantageous to use a network coding procedure on top of an existing routing protocol in the respective scenarios. If both approaches are considered independently of each other, as is the case here, the respective routing method is called *coding-oblivious routing*. Alternatively, a routing procedure can also take into account coding opportunities when selecting a route in order to improve routing and increase performance. This form of routing is called *coding-aware routing* and can be found, for example, in [WDC07; ZL09; SRB10; Che+15]. Furthermore, it is also possible to supplement a routing procedure with network coding to compensate for the disadvantages of the respective routing procedure. This is used, for example, for network coding based opportunistic routing. There, an opportunistic routing procedure is complemented by intra-session network coding. This means, intermediate nodes, which overheard a packet, perform network coding and then transmit encoded packets. This way, instead of many identical packets, more linearly independent packets are forwarded and less coordination traffic is required. Examples are [Cha+07; LLL08; KWH11; KKW12].

In the following chapters, the impacts of the developed BCGC network coding procedure will be studied. For this purpose, it is necessary to choose a routing scheme which specifies how packets are forwarded through the network. BCGC are applied in addition to the chosen routing method. Here, a preferably simple coding-oblivious routing procedure should be assumed as basis, which, however, can easily be exchanged. An analysis of which routing procedure can be supplemented by BCGC and which advantages the respective combination brings is left for future work. In large-scale mobile, lossy wireless networks, routing tables are big and are subject to change and, thus, pose a challenge. In a mobile environment with constantly changing topology, routing tables with information on neighboring nodes, connectivity and paths have to be updated frequently. For this purpose, information has to be exchanged and paths have to be recalculated, which can be complex and expensive, especially if there are many nodes in the network. Unreliable wireless networks in particular represent an additional challenge for the maintenance and reliability of a routing table. The addressed system setup which is described in Section 3.1 includes a large number of nodes and has computing power and energy constraints. Thus, table-based routing procedures and, in general, computationally intense versions of routing are not applicable. The considered routing protocol has to cope with a constantly changing topology. Furthermore, coordination traffic should be avoided, since there are already many communicating nodes in the network and bidirectional communication is not reliably possible. Moreover, the broadcast nature of wireless transmissions should be exploited, i.e. transmitted, overheard packets should not be discarded. In addition, the selection of the concrete routing procedure should avoid having to compare different probabilities for the transmission of packets. According to the system setup in Section 3.1, there are many different original symbols in the system due to the many source nodes. This makes an analysis of the appropriate probabilities complex.

Since a simple, common routing procedure should be used as basis for the application of BCGC, a form of flooding is chosen in the following. Here, packets reach a destination randomly instead of being routed along a previously determined path. This way, however, a path from source to destination can be used which is not guaranteed to be optimal with respect to chosen performance metrics. Furthermore, it is also possible that the max-flow bound is not achieved. Besides this variant of routing where data is simply broadcast to all neighbors, no further routing procedure is considered here.

In this section, the relation between network coding and traditional routing has been described and a routing procedure for the analysis of BCGC has been suggested. Benefits of the application of network coding will be presented next to conclude this introductory chapter on network coding basics.

2.5 Benefits and Drawbacks of Network Coding

In this section, advantages and disadvantages of applying network coding in addition to an underlying routing scheme instead of using only traditional routing, which does not encode data at intermediate nodes, are presented. For the sake of a more compact formulation, the terms network coding versus routing are used for this purpose.

A first benefit of network coding is the potentially increased efficiency concerning throughput in comparison to conventional routing, see Section 2.2.2 or [KM02; CWJ03; CFS06; GR05]. With network coding, combinations of data are sent. Due to its design, an encoded packet can contain, in a sense, information from multiple non-encoded packets. Therefore, different sinks may obtain different, missing data from the same encoded packet. Thus, more data may be retrieved from a certain amount of transmitted packets, which means an increase in throughput. As already mentioned in Section 2.2.3, the optimal multicast rate, i.e. the max-flow/min-cut bound, can be achieved in a multicast scenario by using network coding [Ahl+00]. The optimal rate indicates the maximum number of bits/packets which can be transmitted from the source to all destinations per time unit. This optimal rate is also called the capacity of the network. In multicast networks, linear network coding suffices to reach this rate [LYC03]. For linear network coding, it is possible to determine the coefficients, which are used for encoding at the nodes so that the optimal multicast rate is achieved, by a polynomial time algorithm [KM03; Jag+05]. However, even RLNC, where coefficients are chosen randomly, is asymptotically optimal in a wireless multicast network, as discussed, for example, in [Deb+05]. By contrast, it is generally not possible to reach this rate with traditional routing in a multicast scenario [Ahl+00]. In addition, finding an optimal routing solution is NP-hard, see for example [Jag+05; HL08]. To conclude, using network coding can improve throughput and can possibly enable to achieve the optimal multicast rate, i.e. the optimal throughput. The butterfly network, see Section 2.2.2, is one of the best-known examples of the throughput advantage of coding over routing in a multicast scenario of a wireline network. Via corresponding manipulations of the butterfly network, as discussed for example in [HL08; Yeu08], it can be seen that this advantage does not only apply to multicast or wired networks. The throughput advantage of network coding over routing is not only valid for a single multicast session, as considered in [Ahl+00]. It is also valid e.g. for a single unicast session in a lossy network, or for multiple unicast sessions, see Section 2.2.2 or [LMK05; LL04b; WCK+05; HK05].

Another benefit of network coding is the improved energy efficiency compared to routing, see [WCK05; WFL05; CW07; FWL08]. The required number of transmissions can be used as an approximation of the energy consumption as those two metrics are proportional according to, for example, [WFL05; CW07; FWL08]. However, more precise values for the respective energy consumption of a transmission between each two considered nodes can also be used. Furthermore, minimizing the total amount of energy which is necessary to transmit a bit/packet from a source to each destination in a single multicast scenario is denoted as solving the problem of minimum-energy multicast. As already mentioned, in the case of network coding, an encoded packet can contain, in a sense, information from multiple non-encoded packets, and different data might be obtained from the same encoded packet. Therefore, less transmissions might be required to transfer all data to all destinations, and thus less energy will be consumed. In wireless networks with broadcast transmissions, the problem of minimum-energy multicast can be solved in polynomial time for linear network coding [WCK05; Lun+05b] and is NP-hard for traditional routing [WNE00; Lia02; WCK05]. Being solvable in polynomial time means that the coefficients can be determined by a polynomial time algorithm so that the minimum-energy per packet is achieved. In case an optimal routing solution is found in a multicast scenario, it requires potentially more energy per bit/packet than an optimal network coding solution [WCK05; LMK05]. In the special case of a wireless broadcast scenario, each node is a source node and aims to transmit data to all other nodes of the network. For a wireless broadcast scenario, the minimum-energy problem of routing is NP-complete [ČHE02; Lia02]. If topology changes continuously and nodes do not have any information on the network, network coding outperforms traditional routing concerning the required energy per bit/packet especially in a wireless broadcast scenario [WL05; FWL06; FWL08]. To conclude, energy can be saved by using network coding.

There are many scenarios where the application of network coding has an advantage over traditional routing concerning throughput or energy efficiency. However, there are conditions under which linear network coding is not sufficient in a non-single-multicast scenario. Examples can be found e.g. in [LL04a; DFZ05]. A general overview of cases in which traditional routing can be applied, cases in which linear network coding is sufficient, and cases in which non-linear network coding may have to be used is provided by [LL04a]. For the broadcast scenario assumed here in Section 3.1, linear network coding is sufficient according to [LL04a]. For a single multicast session, there is an explicit max-flow/min-cut representation, see [Ahl+00]. However, this is much more complex for multiple independent multicast sessions, i.e. a multi-source multi-sink scenario as addressed in Section 3.1, according to [Ahl+00; YYZ07].

A further benefit of an additional application of network coding over conventional routing refers to the so-called delay or system latency. Here, delay denotes the time needed to transfer all data from source to destination(s). Network coding can reduce the delay, see for example [CW07; DIG10], due to the lower number of transmissions required, which has already been discussed. The reduced delay can also be seen in the examples of Section 2.2.2.

Another advantage is the fact that network coding enhances reliability. This means, it increases the resilience of the system to packet losses due to link or node failures, see for example [WL05; Fra+07; GTK08]. It should be pointed out that throughput, energy efficiency, and reliability are inherently linked. With network coding, there is spatial redundancy as data is encoded into different packets during the process which are then transmitted. Especially in wireless networks, transmissions have a broadcast character by nature, which can be exploited. In a single transmission, multiple copies of one packet are generated and sent to neighboring nodes in the case of multicast transmissions. Thus, identical packets or data are stored in several separate nodes. In this way, data can reach a destination within various packets, and packets can reach a destination along different, redundant paths. Moreover, when routing and ARQ is used and a packet is lost, the retransmission of exactly this packet is requested. With network coding, on the contrary, feedback and retransmission of lost packets is completely avoided as it is not relevant that each packet or a specific packet arrives at its destination. As soon as a sufficient number of packets has arrived at the destination, all original data can be reconstructed there. Link failures can, therefore, be tolerated up to a certain extent. However, certain conditions are imposed on the packets depending on the respective network coding procedure. For RLNC, for example, a sufficient number of linearly independent packet is necessary. Especially in a lossy wireless network, it is beneficial not to be dependent on a special packet as retransmission is highly inefficient.

In addition to efficiency, reduced delay and enhanced reliability, network coding inherently provides a certain degree of security, which is another advantage of network coding. For example, if only single packets are overheard by a passive eavesdropper, the eavesdropper is usually not able to reconstruct the contained data. Especially with RLNC, the eavesdropper needs a certain minimum number of packets to get any data at all. In the case of BCGC, an eavesdropper would have to be listening from an early stage of the algorithm and receive a suitable combination of packets. Alternatively it would have to use decoding methods like those for RLNC and then eavesdrop a certain minimum number of packets to get any data at all. Generally, it is always possible to add e.g. separate encryption mechanisms to the packets. Further information on security in network coding can be found in [BN05; HL08; CC11; JL12], for example.

These listed advantages of the application of network coding are apparent especially in lossy wireless networks with dynamically changing topology, if a decentralized approach is required, or if no information about the network as a whole is available.

Besides these mentioned benefits, there are also several disadvantages associated with the use of network coding in addition to an underlying routing scheme. For network coding, for later decoding of received codewords, it is necessary to add information to each packet regarding which symbols have been combined in its contained codeword. However, the amount of information required for decoding, and thus the size of the packet header, varies depending on the implemented network coding procedure. Besides necessary information for decoding, a packet header may also have to contain certain additional information for the recipient. For example, nodes may have to add a concrete number, or concrete IDs of desired symbols, or a whole list of reconstructed symbols to the header of their packets for transmission. This information is then used by the recipient to decide, for example, which or how many symbols to combine for a new codeword. Depending on the used network coding method, this additional information in the header can be of different size, but it is also possible that no additional information is included. In general, when using network coding, the size of the packet header is increased, but the payload size remains unchanged. This means that the overhead within a packet becomes larger and consequently the entire packet. As a result, network coding causes larger packets to be transmitted, but may require significantly fewer packets, as discussed in the section on benefits. This tradeoff and its influence on the performance will be examined in more detail in the later evaluation chapter.

Another disadvantage of network coding is the encoding and decoding effort. Depending on the network coding procedure, the complexity of encoding and decoding varies greatly. For encoding, symbols may have to be selected randomly or in a predetermined way and then be combined as determined by the procedure. This can be less or more complex depending on the specifications. Decoding can be simple, but can also require computationally demanding algorithms such as Gaussian elimination. To conclude, in the case of network coding, encoding and decoding cause additional effort, but network coding may require significantly fewer packets to be generated and sent, which in turn can reduce the overall effort.

Chapter 3

System Setup and Performance Metrics

Contents

This chapter starts with a presentation of the assumed system setup, a definition of the addressed goals, and a description of possible application scenarios of our developed network coding procedure in Sections 3.1.1 to 3.1.3. After that, general information on the considered standard, the IEEE Std 802.15.4, is presented in Section 3.2.1. In this regard, we also provide a brief introduction to aspects of the standard which are relevant for following design and parameter decisions in Section 3.2.2. Alternatives to the IEEE Std 802.15.4 are mentioned to complete this issue in Section 3.2.3. Furthermore, the key performance metrics which will be evaluated in a later chapter are presented here in Section 3.3. The topics which are discussed in this chapter represent the framework for our following considerations and analyses.

3.1 System Setup and Desired Goals

In this section, the addressed system setup to which the following chapters refer is introduced first. Based on this system setup, a description of goals and an overview of how these are achieved by using our developed network coding procedure called Broadcast Growth Codes (BCGC) follows. This section closes with possible application scenarios of BCGC.

3.1.1 Assumed System Setup

We are addressing a system which is a large-scale, mobile WSN with hundreds or thousands of battery-powered nodes. Furthermore, it is assumed that nodes have a limited transmission range. In particular, direct communication between an arbitrary node and a central sink or an external agent is not guaranteed. Communication between neighboring nodes should therefore be possible. For this reason, a mesh network topology is assumed. This means, there is no restriction on how nodes are arranged to each other and nodes can transmit data to all nodes within their transmission range. An advantage of a mesh is that there are often several paths between two considered nodes instead of only one path. This fact enhances the robustness of the network and enables more reliable communication. In the case of a static network, the considered network has to be connected but does not have to be complete, i.e. it does not have to be fully connected. Hence, the mesh topology does not have to be a full mesh. In the case of a dynamic network, there must not be any parts of the network that are isolated during the entire dissemination process.

It is assumed that each node generates sensor data, transmits this data, and aims to gather all data which was generated by the other nodes of the network. Nodes which reconstruct and gather original data from other nodes are called sinks, as defined in Section 2.1. Hence, all nodes are assumed to be sinks here, so a broadcast network with multiple independent broadcast sessions is considered. To conclude, all-to-all communication is addressed, which is a special case of many-to-many communication, see Section 2.1. According to Section 2.1, a multi-source multi-sink problem is, thus, regarded here. In the end, all current data could be retrieved from an arbitrary node, for example by an external agent, as all nodes are sinks. This contributes to reliable data retrieval. Alternatively, it is also possible to get all data at an earlier stage by querying several sinks instead of only one sink. The number of sinks which have to be queried to get all data depends on how early the data is requested.

A network with hundreds or thousands of participants can be prone to link failures due to signal collisions. On top of that, reflections or strong signal attenuation can occur if there are many obstacles which are difficult to penetrate in the vicinity of the nodes. Therefore, we are assuming a lossy wireless network with unreliable transmissions. We do not require real-time conditions and do not consider time-critical scenarios. Thus, there is no need for deterministic communication, guaranteed time of reception, or reliability of certain single transmissions. Links may fail and be restored at a later time, which changes neighborhood connections among nodes. For neighboring nodes, there is no difference if nodes move away, switch to sleep mode in the context of so-called duty cycling, are destroyed, or if connecting links fail. For this reason, the terms 'mobile' and 'mobility' will refer to all of these options and denote changed connections in general. We are addressing a system in which the nodes' positions can change. Performance, however, should not rely on a certain degree of nodes' mobility. Besides that, there are no restrictions concerning the physical arrangement of nodes.

Furthermore, it is assumed that there are only low-cost sensor nodes in the network, which have limited computing power and can only perform low-complexity calculations. However, a sufficiently dimensioned data memory is assumed. This is a viable assumption, as even small wireless sensor nodes can be equipped with low-cost external flash memory. Especially in harsh environments where nodes can be easily destroyed or in large-scale networks with a huge number of nodes, it is beneficial to have low-cost sensor nodes. Additionally, nodes are assumed to transmit data via broadcast to all direct neighbors, which is inherent to wireless transmissions.

A deterministic form of routing can be difficult to accomplish in a large-scale, mobile network, see Section 2.4, and will therefore not be considered here. As discussed in Section 2.4, limited computing power is another reason to refrain from computationally intense versions of routing. Since a simple, common routing procedure should be employed here instead, a form of flooding will be used as routing protocol. Thus, data will reach sinks randomly by using broadcast transmissions. For a detailed explanation of this choice of routing method and an overview of routing in general see Section 2.4.

Furthermore, the addressed system should be able to start operating without any complex initialization process or configuration in advance. In this way, the system can be easily installed, and further nodes can also be added later. The used communication protocol should be simple to implement, and it should be possible to implement it in a distributed manner.

For the sake of simplicity, we will consider a system which generates sensor data only once and then distributes this data over the entire network. This can also be extended to several consecutive data collections and distributions, which are referred to as *generations* in the following. Since we are only looking at one data collection and subsequent distribution, nodes will not be idle listening for a long time but will be busy distributing and receiving data. Therefore, no duty cycling should be performed within one generation. This means, nodes should be permanently active within this considered generation and should not switch to sleep mode. Within a generation, they would only miss transmissions instead of saving energy when switching to sleep mode. If there are many collisions, nodes may transmit data less frequently but should never switch off the receiver unit. Once the data distribution of a generation is completed, nodes could go to sleep and save energy until the next data collection. We assume that data has to be collected only rarely and, therefore, duty cycling will be profitable in general, i.e. it helps to save energy, if several generations are considered. More details will be omitted at this point and left to future work. It is assumed that the amount of data collected by a sink per generation is not very large, i.e. no voice or video data will be considered. Thus, packets, which contain sensor data, will not become large even without fragmentation. Furthermore, we do not want to restrict whether data is transferred to the Internet or not. This means, the considered WSN can be part of an IoT system but does not have to be.

We also assume that the considered system can be separated from the rest of the world and can be autonomous. Thus, it does not have to be integrated into an existing infrastructure in a complex way but is operable on its own. This increases the security of the system and simplifies its installation.

Having introduced the assumed system setup, desired goals and how these can be achieved by employing BCGC will now be described.

3.1.2 Desired Goals and Approaches for Achieving these Goals

Our main focus is to ensure reliability of a network despite possible node failures, unreliable transmissions, and uncertain radio connectivity in general. Here, reliability means that all desired original data reaches each sink in the end, i.e. reliable communication between source nodes and sinks is ensured. Depending on the scenario, it may only be required that all data can be accessed by an external agent if a sufficient number of sinks is queried at any given time. Thus, reliability implies that in the end all original data can be retrieved from the network. To achieve this goal despite challenging conditions, a certain amount of redundancy is necessary. However, unnecessary transmissions should be avoided. Data should be distributed with few, short transmissions so that all desired data is received reliably and early, i.e. latency is low, few signal collisions occur in total, and little energy is consumed. Furthermore, throughput should be high. To sum up, the addressed goals, ordered by priority and starting with the most important goal, are:

- reliability,
- low latency,
- high throughput, and
- reduced energy consumption.

In order to achieve these goals, the system setup was chosen accordingly, which is described in the following. Among other things, it was decided that data is reconstructed and gathered at different nodes. A single point of failure is, thus, avoided. According to the considered system setup in Section 3.1.1, data is even gathered by all nodes, which means all nodes are sinks. Consequently, it is possible to retrieve all current original data at an arbitrary node in the end. This spatial redundancy of data increases the reliability of the network, for example, in case of node failure. In addition, the area around a single sink is not heavily stressed since there is neither only one sink nor is systematic routing used, as mentioned in Section 3.1.1. In addition, packets are sent via broadcast to all neighbors and, thus, the contained data is multiplied. Therefore, data is available multiple times in the network already at an early stage which enhances robustness against failures and, hence, ensures reliability. At a certain extend, node failure or an isolation of parts of the network due to link failure does not pose a problem. However, it should be avoided to transmit too much redundancy so that the required number of packets does not increase significantly.

As all nodes are assumed to act as sinks, it is possible to retrieve data from different nodes in order to finally have all current original data. It is not necessary that any of the queried nodes has already reconstructed all data. Hence, all current original data can be retrieved earlier than if there is only one dedicated sink or if at least one random sink has to have all data. This reduces the latency of the system, so also less energy is consumed as all components have to be active for a shorter time. In general, all symbols can either be retrieved by querying several sinks at an early stage or even only one arbitrary sink later. This is advantageous especially for a mobile network or for a network which is susceptible to failure.

In addition to the suitably chosen system setup, the addressed aims can be achieved by using BCGC, our developed network coding procedure. This correlation is explained next. In the case of BCGC, a transmitted packet/codeword consists of a combination of distinct symbols. A node not only uses its own symbol but also combines previously received symbols, which corresponds to the definition of a network coding procedure. This has the effect that each node transmits a symbol several times in different compositions, encoded by using the network coding procedure BCGC. As a consequence, it is not important to receive one specific transmission. It is, thus, no problem to miss single packets. Further transmissions will follow, and eventually a considered sink has gathered all original data. This temporal redundancy increases the reliability of the network and the nodes' data distribution/collection process despite considering a lossy WSN. Thus, robust communication is enhanced by not requiring each packet. However, each packet which is received by a node can be beneficial. After having considered the goal of reliability, latency will be focused next.

Latency should be low, i.e. all sinks should reconstruct and gather all original data quickly. Thereby, it is possible to retrieve data earlier, for example by an external agent querying different nodes. In the case of BCGC, a sink can start decoding and trying to reconstruct original symbols as soon as the first codeword has been received. Thus, in contrast to RLNC, which is a widely used network coding procedure, see Section 4.2, it is not necessary to wait for a certain number of linearly independent packets/codewords and be only able to decode everything at once. Decoded symbols are then stored in the respective sink for BCGC. So, with BCGC, an external agent can easily identify after how many queried nodes he has already retrieved all original symbols and does not have to query any further nodes. It is not necessary to check after each node whether there is a sufficient number of linearly independent packets/codewords among the collected packets, as it would be necessary for RLNC. In addition, if nodes fail at an early stage so that not all symbols have reached sinks, BCGC can be used to reconstruct at least a certain fraction of all symbols since decoding of received codewords is started immediately after reception. This improves the reliability of the system, at least with regard to the symbols that are still present in the system and have not been lost due to node failure. With RLNC, by contrast, no codewords can be decoded and, thus, no symbol can be reconstructed under these circumstances. Thus, by using BCGC, quick and robust data retrieval, for example by an external agent, is enabled, especially in a failure-prone network.

To achieve the goal of low latency, sinks should, among other things, aim to reconstruct many original symbols with any number of received packets. The more symbols can be obtained from each considered number of received packets, the less transmissions/received packets are necessary in total to gather all original data. By reducing the required number of transmissions, not only the algorithm will terminate earlier but also some signal collisions are avoided, reliability is enhanced, energy consumption is reduced, and throughput is improved. Thus, all of the mentioned goals are addressed. Here, the reduction of the required number of packets is achieved by using the network coding procedure BCGC. Compared to transmissions of non-encoded data, network coding in general reduces the required number of transmissions, see Chapter 2. With BCGC, data is combined favorably to avoid many transmissions/receptions of redundant packets, which will be described in detail in Chapter 5. This specific combination also increases the probability that packets will provide immediate benefit to the receiving nodes. Hence, in the case of BCGC, a codeword is specifically composed before broadcasts are performed in order to avoid too much redundancy. In addition, as mentioned before, BCGC provide a certain form of redundancy across packets so that reliability can be ensured without greatly increasing the required number of packets and, thus, without increasing latency. Furthermore, if a large-scale WSN with hundreds or thousands of nodes is considered, BCGC packets are shorter than, for example, RLNC packets. This means, with BCGC, the transmission time of a single packet is shorter. Thus, less collisions occur and the data collection process can be completed more quickly. Summing up, latency is reduced by requiring less packets and by using shorter packets. More details on the procedure of BCGC will follow in Chapter 5.

To conclude, with the mentioned aspects, the procedure of BCGC enables reliable communication and robust data retrieval, despite the network being undependable. Furthermore, latency is reduced, throughput is increased, and less energy is consumed.

After having described the assumed system setup and the addressed goals of BCGC, possible application scenarios of BCGC will be presented next.

3.1.3 Application Scenarios of BCGC

In the following, some examples will be presented for which the application of BCGC can be beneficial. However, BCGC are not restricted to the herein presented specific scenarios. First, rather general use cases are explained, then more concrete scenarios are described. Besides BCGC, NoCoding and RLNC are possible methods that are briefly discussed here for applicability. Details on NoCoding and RLNC can be found in Chapter 4.

3.1.3.1 Robust Communication

A first possible application scenario for BCGC is to enable robust communication and data retrieval in general, for example in a WSN or IoT surrounding with challenging conditions. The addressed system setup should be the same as described in Section 3.1.1. In case of a large number of communicating devices, many signal collisions might be caused, making it difficult to guarantee the successful reception of specific packets. Especially in an additionally harsh environment or at high load of individual nodes/parts of the network, node failures can frequently occur. How in this scenario reliability can be ensured by using BCGC has already been presented in detail in Section 3.1.2. In short, BCGC enable robust communication because of the following three aspects. Due to the temporal redundancy of symbols when using BCGC, it is not required to receive each packet successfully and it is no problem to miss single packets. In addition, even in case of early node failure and a resulting loss of some of the symbols, the symbols still present in the system can be reliably reconstructed in a sink due to the early start of decoding in BCGC. Furthermore, BCGC provide redundancy across packets by combining specifically chosen symbols in a codeword. Therefore, reliability can be ensured without greatly increasing the required number of packets and latency. Reasons for not using related work such as RLNC in this scenario are, for example, the large packets that result from RLNC if a large number of nodes is involved. In addition, the fact that nothing can be decoded if not a sufficient number of packets/codewords has been received is also a reason against RLNC under the described conditions. Moreover, the high decoding effort with RLNC can be a problem. NoCoding, on the other hand, is not well suited because of the very large number of packets required.

3.1.3.2 Route Discovery Phase of a Routing Protocol

Another possible, rather general application of BCGC is in the route discovery phase of a routing protocol in a large-scale WSN. The position of each node has to be communicated to every other node of the network so that every node can determine an optimal route to each node before the system starts its actual work. In this scenario, nodes correspond to devices which generate sensor data and transmit this data via broadcast transmissions to their direct neighbors. According to the system setup in Section 3.1, it is assumed that nodes have a limited transmission range so that communication with a central gateway is not possible for all nodes. However, communication with neighboring nodes is feasible. As assumed in Section 3.1, nodes are placed randomly and connected via mesh topology so that each node can communicate with its direct neighbors but not necessarily with all nodes of the network. Furthermore, nodes are supposed to have just been placed so that they particularly do not know their direct neighbors yet. Nodes should immediately start sending without a long initialization/synchronization and, at least for this initial phase, transmit at random points in time, i.e. without any form of scheduling. Consequently, many collisions occur if many participants are in close proximity to each other. For this scenario, it is assumed that collisions and, therefore, temporary link failures can occur but that nodes do not move so that a route discovery phase is only necessary at the beginning or in certain time intervals. After the route discovery phase, nodes should immediately start communicating and perform their actual task.

Challenges of this scenario are 1) the high data traffic in the case of a large number of nodes, i.e. the large amount of different node information that has to be sent to each node, 2) the large number of resulting collisions, 3) the fact that the route discovery phase should be fast, and 4) the fact that all data has to be reliably available in all nodes at the end. Thus, the following two aspects are the goals of this scenario. 1) The route discovery phase should be fast, which means it should have low latency. 2) Additionally, it should be reliable, i.e. the information from each node should reach every other node successfully, despite possibly high collision rates.

To achieve low latency, it is advantageous if packets are small, few packets have to be received and, therefore, also fewer collisions occur. Especially in a large-scale system, RLNC packets are often large and in the case of NoCoding, many packets have to be received as many redundant packets arrive. For many nodes in the network, the size of BCGC packets is much closer to NoCoding packets than to RLNC packets. Regarding the required number of packets, BCGC are much closer to RLNC due to the specific packet composition adapted to the current state of the system. Neighboring nodes receive few redundant packets and many packets that they can immediately use because of the specific composition in BCGC. Due to shorter packets and a smaller number of required packets, it is possible to achieve a lower latency with BCGC here. Furthermore, BCGC allow to start decoding and, thus, processing of the nodes' data as soon as the first packet has been received. This means, nodes can start immediately with first calculations for route discovery in the case of BCGC. In contrast, with RLNC, it is necessary to wait for a certain number of received packets before being able to decode anything. Concerning reliability, it is advantageous if not every packet has to arrive, i.e. individual packets can be lost. It is also beneficial referring to reliability if packets contain a combination of data from several nodes so that the information from each node is available several times in the network. BCGC fulfill both of these desirable aspects. Moreover, it is also an advantage concerning reliability if packets are small, few packets are needed, fewer collisions occur, and lower latency is required. Consequently, BCGC are well suited for the application scenario described here.

3.1.3.3 Industrial Internet of Things

After general scenarios have been treated, more concrete scenarios will be considered next. A possible use case of BCGC belongs to the Industrial Internet of Things. Here, we consider a production line which is repeatedly modified, i.e. movement is performed at discrete points in time. This modification can contain reconfiguration of machines and can be performed, for example, up to 400 times a year. Reasons for a modification can be changing production orders, for which a different configuration of machines is required in each case. For monitoring, machines can be retrofitted with sensor nodes. Depending on the size of the production line and what kinds of sensor values should be determined, the number of nodes can vary greatly. One hundred nodes to several thousands of nodes are possible. For such sensor nodes, using power line or wired communication can be difficult to realize. Cables can be under heavy strain in an industrial environment and, thus, break easily. On top of that, some parts of a machine may not practically be equipped with cables. Therefore, sensor nodes are often battery operated and use wireless communication. Due to the large number of distributed nodes and the challenging environment, a mesh topology, where nodes are connected to nodes within transmission range, but not to all nodes, can be beneficial. It is assumed that, due to the harsh environment, no synchronized communication is possible, so transmissions at random points in time and resulting collisions are considered. Furthermore, the big amount of impenetrable or reflective obstacles in an industrial environment can affect the radio range. As a consequence, direct communication with a central gateway may be difficult to realize. A node in the immediate vicinity is more likely to be successfully reached. The aim of industrial monitoring can be anomaly detection. To this end, vibrations can be measured in order to recognize changes at an early stage, for instance. In this way, sensors identify if general maintenance or repair work should be performed in order to avoid an expensive unexpected, long machine downtime due to defects. It is possible that certain sensor data, for example data concerning maintenance of wear parts, will be monitored by external companies. In this case, it is advantageous if sensor nodes do not have to be integrated into the company's infrastructure. Instead, they should represent an independent network that can be quickly set up or expanded. Furthermore, production in general can be monitored for quality assurance. Thus, it can be determined and documented whether a certain predefined quality standard is adhered to. Noncompliance can be detected early in order to avoid large defect batches. In addition, the performance of a single machine or of a sequence of machines can be analyzed, and processes can subsequently be improved. Especially if a production line is repeatedly modified, it can be useful to check the performance regularly and, if necessary, to change suitable parameters. Different configurations or arrangements of machines can also be compared in this way. If the production line is modified, some or all sensor nodes will move with the respective machine or even have to be installed at a different part of the machine. Thus, nodes do not move continuously but at discrete times. However, an industrial environment can be harsh for sensor nodes and the communication between these nodes. There may be dust, vibrations, reflections on the floor, walls, or machines, or in general many materials which are difficult for radio to penetrate. Additionally, the more nodes are communicating, the higher the probability of signal collisions is. The resulting node or link failures show a similar behavior as if nodes were moving. Especially in a rough industrial environment, it can be beneficial to retrieve current sensor data at several distributed places in order to avoid a single point of failure. The nodes' data should be gathered quickly even after having modified the production line. Therefore, it can be useful if it is possible to query an arbitrary node to receive all or a big amount of generated sensor data of the network. For this reason, all current data should be gathered in each node. Otherwise, a previously selected node could later possibly be at a place that is no longer optimal or even inappropriate for fast data collection. On top of that, data retrieval should be performed without complex, time-consuming initialization.

To sum up, the challenges of this described IoT application scenario are 1) the potentially large number of nodes and, thus, again the high data traffic between the nodes, 2) in addition to the resulting collisions, the harsh environment which leads to the transmission range being negatively affected or even to nodes failing, and 3) the goal to reconstruct data as fast as possible in all nodes in order to ensure a quick data retrieval at any time. However, it may not be necessary for all original data to be present in each node, all but a certain percentage may be sufficient. In addition, there might be a certain redundancy of sensor data among the sensor nodes which should ensure a reliable monitoring of the production line. Therefore, depending on the amount of redundancy installed, it is not necessary to have the data of each sensor. Having all data reliably everywhere is, thus, not the primary goal. In this case, not only one node but a small number of nodes will be queried at the end if all data of the network has to be retrieved nevertheless. Thus, the most important goal in this considered scenario is that data is available quickly.

As already described in detail for the more general scenario of route discovery in a routing protocol, BCGC are particularly well suited to ensure low latency. This is achieved by using short packets and requiring a small number of packets to be able to reconstruct a certain amount of original data. In addition, due to the early start of decoding in BCGC, an external agent can quickly query a sufficient number of nodes at any time and rapidly collect all or the desired amount of original symbols. Thus, several nodes can be queried at an early stage and one or a few random nodes at a later stage in order to obtain the desired amount of data. Consequently, the procedure of BCGC is particularly suitable for collecting data in this described use case. NoCoding and RLNC are not as well suited as BCGC for the same reasons as mentioned in the scenarios above. NoCoding requires many packets, and RLNC uses large packets in the case of a large number of nodes. In addition, RLNC requires the same number of linearly independent received packets as there are different original symbols in the network in order to be able to decode anything. An external agent is not able to reconstruct any symbol before that. In the case of BCGC, the external agent can stop querying nodes as soon as it has collected the desired amount of reconstructed symbols from the nodes. Especially at an early stage of the algorithm and if not all data is requested this can be much earlier than with RLNC. In particular, the latter aspect of early decoding is a significant advantage of BCGC in this use case.

3.1.3.4 Monitoring in the Medical Field

The last presented, concrete scenario is from the medical field. Here, we are considering a hospital or a mobile medical clinic which has been quickly set up in a state of emergency. Positions and availability of, for example, beds, medical material or medical equipment are now to be observed by a network of sensor nodes. Medical equipment can be heart defibrillator, lung ventilator, ECG machine, or medical monitors in general. Since many of these objects are mobile and can be moved, the sensor nodes attached to them also move relatively to their surroundings. Thus, we are considering a large number of more than 1000 nodes which can move continuously. In this scenario, a complex, time-consuming initialization should be avoided so that the system is quickly ready for use immediately after installation. Furthermore, it is assumed that, especially due to the very large number of mobile nodes and the quick setup, no scheduling is used for communication and nodes transmit at random points in time. Thus, many signal collisions can occur. On top of that, there is a multitude of obstacles and possibly massive walls in proximity to the nodes, which can lead to strong signal attenuation and reflections. In addition, node failures are also common due to heavy use of the objects which are equipped with sensors. Here, low-cost sensor nodes with low computing power should be used. Besides, a central gateway that collects all current data which was generated by the sensor nodes is not suitable in this scenario. It would not be directly reachable for every node due to the limited transmission range. Furthermore, a central gateway would represent a single point of failure. Additionally, the area around the gateway would be highly stressed and, thus, prone to failure. It is not necessary for each node to be connected to every other node, but it is sufficient if some parts of the network are connected via only few nodes. This means, nodes can be placed in the corridor and, thus, connect the nodes from the individual rooms. However, the network as a whole should be connected. Nodes are assumed to use broadcast transmissions, which are inherent in wireless communication, to all direct neighbors within their transmission range.

The challenges of this presented application scenario are 1) the very large number of nodes and, thus, the information of the many nodes that have to be transmitted to many other nodes, 2) the many collisions in the dense parts of the network, 3) the existing sparse parts of the network which are only connected to the rest of the network via one or a few nodes, 4) the fact that all data should reach each node reliably, and 5) the fact that requested information should be available early. Thus, the aim here is that all nodes can reliably and quickly gather all current sensor data which was generated by the nodes of the network so that a querying agent reliably receives its desired data.

Thus, it is possible to query any conveniently located node and obtain all current sensor data of the network from it. It may not be convenient or would take a longer time to query multiple nodes. In this way, location and availability of a sought-after object can be determined. However, decoding should be started early by each node as it should be possible to read out received symbols from a considered node early to get the desired information as quickly as possible. A querying agent is usually interested in the information of one or more special nodes. So, as soon as the respective symbols are reconstructed in a queried node, the agent has been successful. Especially in the case of existing sparse parts of the network, it can take a long time until all information is available in a considered node. For this reason, it would be problematic if, like with RLNC, only everything could be decoded at once. Therefore, early decoding of received packets/codewords is crucial here, which can be fulfilled very well by BCGC but not by RLNC. Thus, the mentioned goals of reliability, low latency, and early decoding can be better achieved by using BCGC than by NoCoding or RLNC, as already explained in previous paragraphs.

3.2 Considered Standard: IEEE Std 802.15.4

The IEEE Std 802.15.4 is one possible choice for a specification of the communication on physical and MAC layer for the developed network coding procedure BCGC and matches the addressed system setup. However, BCGC are not restricted to the IEEE Std 802.15.4. When using an alternative specification, certain fundamentals, such as packet structure and maximum packet size, can be different. Permitted number of transmissions per time interval, modulation technology, frequency, and, thus, data rate, or range can also vary. On top of that, the maximum possible number of devices, the cost for deployment and single devices, as well as license fees, reliability of transmission, power consumption, and latency can change. The procedure of BCGC, however, remains unchanged. Even though different parameter values will be optimal.

As a consequence, only general information on IEEE Std 802.15.4 and aspects which are relevant for BCGC will be presented in the following. An additional detailed description of the standard is omitted. Alternatives to IEEE Std 802.15.4 are mentioned at the end of this section.

3.2.1 General Information on IEEE Std 802.15.4

The Institute of Electrical and Electronics Engineers (IEEE) is an organization which, among other things, develops standards in the technological field. Project 802 provides standards relating to local and metropolitan area networks (LAN/MAN), see [DAm19]. More details on IEEE 802 wireless systems can also be found, for instance, in [WMB07]. Examples of well-known IEEE 802 standards are the IEEE Std 802.3 for Ethernet or the IEEE Std 802.11 for Wireless Local Area Network (WLAN), see [IEE18] or [IEE16a], respectively. In the context of WSNs, the IEEE 802.15 Wireless Personal Area Network (WPAN) standard series can be applied. In particular, the IEEE Std 802.15.4 for Low-Rate Wireless Networks, see [IEE16b], is an often used standard, according to [Alf19]. We will refer to the IEEE Std 802.15.4 [IEE16b] in subsequent chapters and, therefore, address relevant aspects in the following. This standard specifies network communication for the physical layer and the MAC sublayer of the ISO Open Systems Interconnection (OSI) model [ISO94]. Details on the OSI model are omitted here but can be found, for example, in [Ala14; Bla06; Shi01]. The IEEE Std 802.15.4 addresses low cost, low complexity, low power consumption, low data rate systems, see [IEE16b]. These are typical constraints for large-scale WSNs and will be briefly explained in the following.

WSNs, especially in the field of IoT, often consist of a big amount of nodes and are battery operated. If a system contains many nodes, it is desirable that these nodes are inexpensive. This can be achieved, among other things, by using nodes which only have low computing power. Therefore, low complexity calculations are preferable, which additionally saves energy. If a system runs on battery power and the battery cannot be exchanged or even not recharged, low energy consumption is key. Therefore, IEEE Std 802.15.4 provides the option to use duty cycling, i.e. nodes can alternate between an active mode and an inactive sleep mode. Another possibility to save energy is to use a low data rate.

The IEEE Std 802.15.4 only focuses on the two lowest layers of the OSI model. Therefore, this standard can be supplemented, for example, by ZigBee or IPv6 over Low Power Wireless Personal Area Network (6LoWPAN). A detailed overview of ZigBee/IEEE 802.15.4 and 6LoWPAN can be found, for example, in [Zig15; Zig19; Erg04; Bar+07] and [GC19; KMS+19; ML08], respectively.

Beginning with IEEE Std 802.15.4-2003 [IEE03], there are different versions and amendments of this standard. A survey of these variants of IEEE Std 802.15.4 can be found, for example, in [RN19]. IEEE Std 802.15.4-2015 is the latest revision of the standard, which was published in 2016, but some transceivers in existing testbeds still relate to older versions. The Atmel AT86RF231 transceiver [Atm09], for example, which is used in nodes of the FIT IoT-LAB testbed, see Chapter 8 and [Adj+15b], is compliant to IEEE Std 802.15.4-2006. For reasons of fair comparison with the results of the testbed, we will only be using aspects of IEEE Std 802.15.4-2015 in subsequent simulations which already existed in 802.15.4-2006.

3.2.2 Considered PHY and Packet Size Specifications for Data Packets in IEEE Std 802.15.4

In IEEE Std 802.15.4, there are several options concerning the used frequency. We decided to choose the 2.4 GHz industrial, scientific, and medical (ISM) band. The ISM band is license-free, available worldwide, and has been considered by the standard since the first version in 2003 [IEE03]. In this frequency band, IEEE Std 802.15.4 provides, among others, an O-QPSK modulation. O-QPSK is a digital modulation method which performs offset quadrature phase shift keying. Details on O-QPSK can be found e.g. in [IEE16b], or [Che04]. According to [IEE16b], a PHY activates and deactivates the transceiver, determines if the channel is busy in the case of CSMA-CA, and transmits and receives data, for example. Only O-QPSK has been available for the 2.4 GHz PHY since the first version of the standard, see [IEE03]. On top of that, the Atmel AT86RF231 transceiver [Atm09], which is deployed at the FIT IoT-LAB testbed and, therefore, is considered in Chapter 8, uses O-QPSK. For these reasons, we will refer to the 2.4 GHz O-QPSK PHY here. However, there are many other PHYs with a different frequency or modulation method to choose from in IEEE Std 802.15.4. Consequently, a different data rate, transmission range, or packet reception rate, for example, can result. An appropriate PHY has to be chosen depending on the considered scenario and prerequisites.

In the following, frame format specifications for data packets in IEEE Std 802.15.4 will be presented. Some fields of the standardized frame are optional and will be chosen suitably to the mentioned assumptions in Section 3.1. The specified data frame format in IEEE Std 802.15.4-2006 [IEE06], IEEE Std 802.15.4-2011 [IEE11] and IEEE Std 802.15.4-2015 [IEE16b], for example, differs in a few fields. As already explained in Section 3.2.1, we will refer to IEEE Std 802.15.4-2015 but only use aspects which already existed in IEEE Std 802.15.4-2006. Concerning frame format, the differences between IEEE Std 802.15.4-2006 and the latest version of the standard will be noted in the following description.

In IEEE Std 802.15.4, there are two options for the address format, which influence the size of a data packet. A specified address in IEEE Std 802.15.4 can either be a 64-bit extended universal identifier (EUI-64) or a 16-bit short address. The EUI-64 is globally unique and generated as described in [IEE14]. Since 16-bit addresses are sufficient for

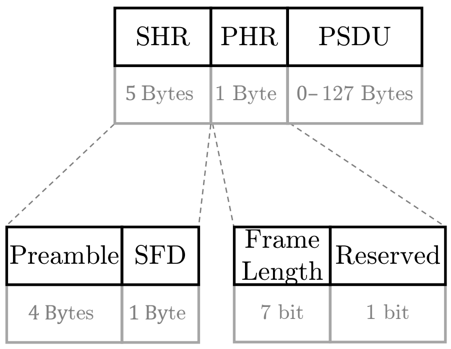


Fig. 3.1: PHY Packet format

the considered system setup, we will use the 16-bit short address format. According to [GMS15], short addresses are frequently used in limited scenarios, see for example [SB16; Lee05; Zen+08].

All following aspects and values refer to data packets. Furthermore, the O-QPSK PHY, which is responsible for data transmission on physical layer, is assumed when indicating the respective number of Bytes. In IEEE Std 802.15.4, see [IEE16b], physical layer packets are restricted to a maximum of 133 Bytes when considering 2.4 GHz with O-QPSK and a data rate of 250 kbit/s. Therefore, the maximum transmission time of a data packet is $\frac{133\,\text{Bytes}}{250\,\text{kbit/s}} = 4.256\,\text{ms}$. When using O-QPSK, a physical layer packet contains an overhead of 6 Bytes and $0-127$ Bytes of payload. Considering physical and MAC layer, a data packet which fulfills the mentioned assumptions has an overhead of 17 Bytes, which will be explained next. Thus, 116 Bytes remain for overhead and payload in upper OSI layers.

In IEEE Std 802.15.4, see [IEE16b], Physical Layer Packets (PHY Packets) have the following structure: Packets start with a Synchronization Header (SHR), followed by a Physical Header (PHR), and a Physical Payload (PHY Payload). The PHY Payload is also called Physical Service Data Unit (PSDU). The SHR consists of a Preamble Sequence (4 Bytes when considering 2.4 GHz with O-QPSK) and a Start of Frame Delimiter (SFD) (1 Byte when considering 2.4 GHz with O-QPSK) in order to indicate the start of an arriving packet. The PHR only contains the Frame Length/Reserved Field (1 Byte) to specify the length of the following PSDU. Only 7 bits of the PHR are used to determine the length of the PSDU in Bytes, so the PSDU length is $0-127$ Bytes. As a consequence, a PHY Packet is restricted to a maximum of 133 Bytes and has an overhead of 6 Bytes. The structure of a PHY Packet, as described here, can also be seen in Figure 3.1.

On MAC layer, the PSDU, is called MAC frame. Depending on its type, a frame has a specific structure. We only consider data packets, so the following indicated sizes refer to data frames. The MAC frame is divided into the MAC Header (MHR), the MAC

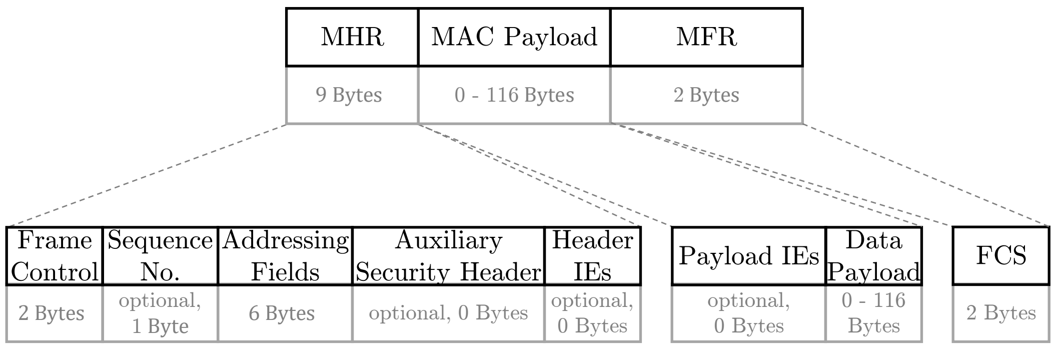


Fig. 3.2: MAC frame format for data frames

Payload and the MAC Footer (MFR). For a data frame which is also compatible to IEEE Std 802.15.4-2006, the MHR consists of the Frame Control Field (2 Bytes), the Sequence Number (1 Byte), Addressing Fields, and an optional Auxiliary Security Header. In IEEE Std 802.15.4-2015, the Sequence Number is also optional and can be suppressed. However, in IEEE Std 802.15.4-2006 and IEEE Std 802.15.4-2011 this suppression has not been possible yet. Therefore, the Sequence Number will not be omitted in the following. For our purpose, we are using 2-Byte short addresses. Then, the Addressing fields contain the Destination Address (2 Bytes), the Source PAN Identifier (2 Bytes) and the Source Address (2 Bytes). As we do not want to include additional security mechanisms, yet, the Auxiliary Security Header is omitted in the following. In contrast to IEEE Std 802.15.4-2006 and IEEE Std 802.15.4-2011, IEEE Std 802.15.4-2015 provides the option to add one or more Information Elements (IEs) to a data packet. In order to include IEs, Header IEs have to be added to the MHR, and Payload IEs to the MAC Payload. In order to be compatible with IEEE Std 802.15.4-2006, we will not use IEs in the following.

The MFR contains the Frame Check Sequence (FCS) (2 Bytes when considering 2.4 GHz with O-QPSK). In total, the MAC frame has an overhead of 11 Bytes. As it is restricted to $0-127$ Bytes, at most 116 Bytes remain for the MAC Payload. The structure of a MAC frame, as described here, can additionally be seen in Figure 3.2.

Considering physical and MAC layer, a data packet has an overhead of 17 Bytes, and 116 Bytes remain for overhead and payload in upper OSI layers. Thus, 116 Bytes are available for the BCGC frame, which will be discussed in Section 5.4 in more detail.

3.2.3 Alternatives to IEEE Std 802.15.4

In the field of WSNs and IoT, there are various alternatives to IEEE Std 802.15.4 and technologies which supplement this standard. Popular specifications relating to short-range wireless communication are Bluetooth Low Energy (LE) [Blu19a], Bluetooth Mesh [Blu19b], ZigBee PRO [Zig15], Z-Wave [Z-W20], and Thread [Bru19], for example. ZigBee PRO and Thread are build upon IEEE 802.15.4. Furthermore, there are certain standards which were developed especially for the field of the Industrial Internet of Things. Examples are WirelessHART [Int16], ISA100.11a [Int11], or WIA-PA [Int15]. Concerning physical layer, they are based on IEEE 802.15.4 but differ on MAC layer in order to meet the requirements of an industrial application. However, ZigBee PRO is also frequently used in Industry 4.0.

In the field of Wireless Local Area Networks (WLAN), there are several different specifications, see [IEE16a]. Here, for example, Wi-Fi 6 (IEEE 802.11ax) [IEE21], which follows the widely-used Wi-Fi 4 (IEEE 802.11n) [IEE09] and Wi-Fi 5 (IEEE 802.11ac) [IEE13], can be mentioned. A low-power WLAN standard with longer range is Wi-Fi HaLow (IEEE 802.11ah) [IEE17].

Concerning long-range communication, there are cellular technologies, such as 4G/5G [3GP20b], and Low Power Wide Area Network (LPWAN) specifications. Prominent examples from the field of LPWAN are Narrow-Band IoT (NB-IoT) [3GP20a], Sigfox [Sig19], and LoRaWAN [LoR18].

For a detailed comparison of the mentioned specifications, see e.g. [GK15; WJ16; ARH16; RKS17]. Depending on the considered scenario and prerequisites, an alternative to IEEE Std 802.15.4 might be preferable. For BCGC, especially specifications which can be used in a mesh network apply, for example, Bluetooth Mesh, ZigBee PRO, or WirelessHART.

3.3 Performance Metrics

In this section, the performance metrics which are relevant for the addressed system setup, see Section 3.1, are introduced. These presented metrics will be considered in subsequent evaluations.

3.3.1 Required Number of Packets

An important performance metric is the *required number of packets* (RNP) which has to be received by a sink to reconstruct and gather a certain number of distinct original symbols. Here, the RNP to reconstruct all symbols is the most relevant value. Depending on the percentage x of distinct original symbols which should be reconstructed, the acronym RNP_x is used. In addition to the average of the sinks' values, the minimum and maximum value can also be interesting.

In general, the RNP is relevant as it enables to compare different approaches directly with each other without the respective packet sizes having an influence on the result. Neither the probability of signal collision nor the transmission time is contained as only the required number of successfully received packets is counted. Thus, the RNP provides an indication of the quality of an approach's successfully transmitted packets. On top of that, the reliability of the system is inversely proportional to the RNP. A lower RNP means that less transmissions are necessary to receive a certain number of useful symbols. Therefore, less transmissions can be affected by collisions, which means the reliability of the nodes' data distribution/collection process will be enhanced. Furthermore, the RNP influences the duration of the process and also the energy costs.

In general, the RNP can be indicated as direct value or as percentage of the number n of existing symbols, which means $\frac{RNP \cdot 100\%}{n}$. This value is greater than or equal to 100% as RNP $\geq n$.

Another relevant performance metric in the context of RNP is the so-called *RNP gain*. It is defined as the ratio of the RNP without a certain modification to the RNP with this modification. If the RNP gain relating to non-coding vs. coding procedure is considered, the term *coding gain* can also be used instead, see for example [Kat+06; Li+07].

3.3.2 Node Latency/System Latency

Another performance metric is the time until a sink has completed the algorithm successfully, which means until a sink has reconstructed and gathered all distinct original symbols. This metric is called *time until finished* or *node latency*. The maximum value of this metric, the *system latency*, is the time until the last sink has reconstructed all symbols, i.e. the time until all sinks have reconstructed and gathered all original symbols.

The time until finished/latency is a relevant performance metric as it allows a realistic comparison of different approaches. Real circumstances, e.g. the actual transmission time of a packet, the probability of collision, the used MAC protocol, and, thus, the duration of the entire process, are considered. This performance metric reflects the quality of an approach as a whole. As a result, latency represents a good complement to the RNP. On top of that, latency is also inversely proportional to the reliability of the system. A shorter time until finished or lower latency means that the system can only be affected by collisions for a shorter period of time, for example. Thus, the system is more reliable if the latency is lower.

3.3.3 Number of Queried Nodes

The *number of queried nodes* represents the number of randomly chosen nodes which have to be queried to retrieve all distinct original symbols at a certain fixed time during the simulation.

Depending on the scenario, the number of queried nodes can be an important performance metric. In some scenarios, all original symbols should be available as early as possible and not necessarily in one single sink or in all sinks.

3.3.4 Packet Reception Rate

The *Packet Reception Rate* (PRR) is the ratio of packets which were received by a node to packets which were sent to the considered node, i.e. which could have been received. Packets do not reach a node due to 1) signal collisions, 2) if the node is not within the transmission range during the whole transmission, or 3) if the node is not in receive mode.

By contrast, the *collision rate* (coll) refers to the packets which could have reached a considered node but were not successfully received by the node in the actual simulation. Here, if the node is not within the transmission range during the whole transmission or if a node is not in receive mode and a packet arrives, this transmission is also considered a collision as the packet cannot be received. Thus, $PRR = 1 - coll$.

The PRR is a metric which gives information about the suitability of the chosen MAC protocol and the quality of the links in the system, i.e. the dependability of transmissions. It is influenced, for example, by the transmission time of a packet, i.a. by the packet size. The indication of the PRR can be used when performance results of a certain procedure are presented. Especially for low PRR, the performance of a procedure can be of interest.

As described in Section 3.2.2, the frequently used 2.4 GHz ISM band is considered here. Therefore, in a real world scenario, the PRR can also be influenced by the presence of transmissions by other technologies, such as Wi-Fi or Bluetooth, which also operate on 2.4 GHz. The interference from these techniques on IEEE Std 802.15.4 has been comprehensively studied in literature, see [GHZ12; NZN16; Pet+07]. Depending on the distance of an IEEE Std 802.15.4-device to a Wi-Fi access point, for example, the reliability of transmissions can be severely affected, see [GHZ12].

3.3.5 Throughput

The term *data rate* is defined as the maximum possible number of bits that theoretically can be transmitted by a node per time unit, see for example [SM09]. By contrast, *throughput* is the amount of useful data which is actually received by a node per time unit. This is a common differentiation which can often be found in literature, see [SM09; Sin14; Dea12]. The data rate does not consider signal collisions and assumes a node receives packets all the time without any break in between packets. On top of that, it assumes a packet only contains useful information, i.e. no protocol overhead is sent, no redundant data is transmitted, and so on. Therefore, the determined throughput is always less than or equal to the specified data rate. According to Section 3.2, a data rate of 250 kbit/s will be assumed in the following. Depending on the respective layer, a different throughput will be determined. Considering a physical layer packet according to IEEE Std 802.15.4, see Section 3.2.2, the overhead is 6 Bytes and the payload is

$0-127$ Bytes. Thus, the throughput on physical layer is:

$$\mathit{throughput}_{PHY} \leq \frac{127}{133} \cdot 250\,\text{kbit/s} \approx 239\,\text{kbit/s}.$$

On MAC layer, the packet's overhead is 11 Bytes and the payload is $0-116$ Bytes. For further explanation, see Section 3.2.2. Therefore, the throughput on MAC layer is:

$$\mathit{throughput}_{MAC} \leq \frac{116}{127} \cdot \frac{127}{133} \cdot 250\,\text{kbit/s} = \frac{116}{133} \cdot 250\,\text{kbit/s} \approx 218\,\text{kbit/s}.$$

In general, the smaller the overhead is, the larger the throughput is. As already explained in Section 3.2.2, the maximum possible size of a BCGC frame is 116 Bytes. A lower bound for a BCGC packet overhead is calculated as follows. Here, *deg* is the size of the field which contains the used codeword degree, *maxDeg* is the maximum codeword degree to use, and *IDs* is the size of the field with the IDs of the symbols that were combined for the codeword. Furthermore, *desDeg* is the size of the field which contains the desired codeword degree. A more detailed description of *deg*, *maxDeg*, *IDs*, *desDeg*, and their respective value can be found in Section 5.4. Hence,

$$\begin{aligned}
\mathit{overhead}_{BCGC} &\geq \mathit{deg} + \mathit{IDs} + \mathit{desDeg} = \\
&= \lceil \log_2 \mathit{maxDeg} \rceil \text{bits} + (1 \cdot 2)\text{Bytes} + \lceil \log_2 \mathit{maxDeg} \rceil \text{bits} = \\
&= 2\text{Bytes} + 2 \cdot \lceil \log_2 2 \rceil \text{bits} = \\
&= 2\text{Bytes} + 2\text{bits} \Rightarrow 3\text{Bytes}.
\end{aligned}$$

Here, the smallest reasonable maximum degree of $\mathit{maxDeg} = 2$ and a codeword degree of 1 is used, which causes the lowest possible overhead. In this case, up to $116-3$ Bytes $= 113$ Bytes of payload can be included in a BCGC data packet. Thus, the throughput for BCGC is:

$$\mathit{throughput}_{BCGC} \leq \frac{113}{116} \cdot \frac{116}{133} \cdot 250\,\text{kbit/s} = \frac{113}{133} \cdot 250\,\text{kbit/s} \approx 212\,\text{kbit/s}.$$

The throughput which is determined during following simulations will be called the *effective throughput* to distinguish from the upper bounds which have been introduced here. The effective throughput is calculated as follows:

$$\mathit{throughput}_{\mathit{eff}} = \frac{n \cdot pl}{\mathit{time}}\,[\text{kbit/s}]\,.$$

Here, n is the number of distinct original symbols, which were generated by the nodes of the network, and pl is the payload size, which is the size of one original symbol. Furthermore, *time* is the measured average time which was necessary to reconstruct all n original symbols by a sink. This means, n and pl are fixed, and *time* results from

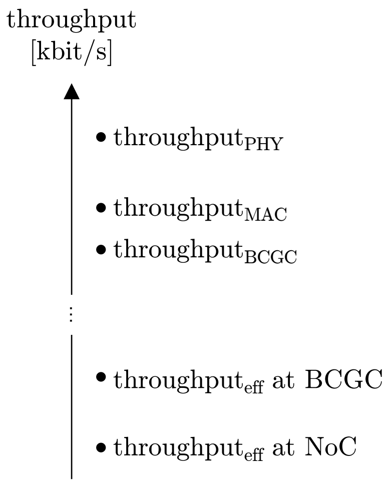

throughput [kbit/s]
throughput$_{\text{PHY}}$
throughput$_{\text{MAC}}$
throughput$_{\text{BCGC}}$
throughput$_{\text{eff}}$ at BCGC
throughput$_{\text{eff}}$ at NoC

Fig. 3.3: Upper bounds of throughput and example for effective throughput for BCGC and NoCoding/Forwarding

simulation. The size of useful data in BCGC is $n \cdot pl$ as only n distinct original symbols exist, so it is not possible to get further relevant data from received codewords. Each symbol has a size of pl. Therefore, the effective throughput is the amount of useful data which was received by a sink within the measured time. Thus, it indicates the amount of useful data which was received per time unit. One of the differences between the effective throughput and the hypothetical upper bound of the throughput in BCGC is the fact that a node does not only receive data but also transmits data. Here, sending and receiving is performed consecutively, and for the effective throughput only received data is relevant. On top of that, redundant packets are also received, and there are collisions, which means some packets cannot be received successfully by a node. All of those aspects reduce the effective throughput as the time until a sink has reconstructed all n original symbols is used for the formula.

Figure 3.3 illustrates the different throughput options. In this figure, an example of an effective throughput for BCGC and for NoCoding/Forwarding, which means no encoding is performed, is added. Details on NoCoding can be found in Section 4.1. The corresponding simulation used a payload of 320 bits and a high degree of mobility.

A further relevant performance metric in the context of throughput is the so-called *throughput gain*. It is defined as the ratio of the throughput with a certain modification to the throughput without this modification. In the case of the mentioned example with 320 bit payload, the throughput gain from the use of BCGC compared to the use of NoCoding was about $\frac{5}{1} = 5$.

3.3.6 Energy Consumption

Another performance metric which will be taken into consideration in Sections 7.1 and 7.2 is *energy consumption*. Our main focus is reliability, see Section 3.1, so energy efficiency is only secondary but will also be considered for the sake of completeness. In general, energy efficiency is relevant as small sensor nodes in large-scale WSNs or IoT settings often run on battery power. In some cases, batteries cannot be recharged and also not exchanged, especially if nodes are deployed in harsh, extreme, or dangerous environments.

The amount of energy which is necessary for generating sensor data can vary considerably depending on the specific application and the type of sensors used, see [Rag+02; Ana+09; Gar10]. For example, passive sensors, such as temperature sensors, consume significantly less energy than active sensors, such as radar sensors. Furthermore, also the amount of energy consumed by data processing can vary greatly depending on the type of instructions, implementation of the core, or technology used, for example. It is difficult to make a general statement concerning sensing or data processing. In literature, energy efficiency is often directly derived from the required number of transmissions until a certain amount or all original data has been received by all sinks, see for example [WFL05; FWL08; ZNK12]. Energy efficiency concerning sensing or computation can each be considered as orthogonal to energy efficiency concerning communication. Here, we will thus only focus on communication energy consumption.

The detailed energy model which is used to derive the energy consumption of the simulated approaches in Sections 7.1 and 7.2 can be found in Section 6.1.7. For the testbed based evaluation in Chapter 8, the actual power consumption of each node was measured during the experiments and, then, energy consumption could be calculated. The measurement was performed as provided by the used FIT IoT-LAB testbed platform and as described, for example, in [Adj+15a; Fam+14].

Depending on the technology used, transmitting, receiving, and waiting for packets (idle listening) has a different impact on the total energy consumption. In general, however, it can be said that if, for example, the total number of transmissions/RNP is lower while the packet size remains constant, the time nodes send/receive/wait for packets can also be reduced. In this way, energy consumption will be decreased. Thus, concerning transmissions, options to save energy are to reduce the number of transmissions or the size of data packets. This can be done, for example, by decreasing the total amount of data which has to be sent or by reducing the size of a single symbol of generated raw sensor data, respectively. Sensor data is often spatially and temporally correlated, see [VAA04], especially if nodes are deployed closely. Therefore, data compression can reduce the total amount of data for transmission and also the size of a data packet. A survey of different data compression procedures for Wireless Sensor Networks can be found, for example, in [RBD13; Sri+12; KL05]. The amount of data for transmission can also be decreased by reducing the sample rate at which sensor data is generated so that less raw data is sampled in each node. Furthermore, it is also possible to send only significant data. In this case each node has to identify relevant data. However, in the system setup in Section 3.1, it is assumed for simplicity that sensor data is generated only exactly once per node. Another possibility to reduce the number of transmissions is to combine data by using network coding, see [FLW06]. A network coding procedure, BCGC, is presented and investigated in the following chapters.

Besides the listed examples, there are plenty other techniques to save energy, see [Ana+09; RBC14; Wan+17]. Depending on the considered scenario or constraints, a different approach or combination of approaches to reduce energy consumption will be applicable. Some of the mentioned examples can, for example, increase latency, do not transmit all original data, or reduce the density of the network significantly. Each of these effects can be tolerable for some scenarios and not tolerable for others.

Chapter 4

Related Work Coding Schemes and Data Dissemination Approaches

Contents

In this chapter, related work concerning coding schemes and data dissemination is presented. First, the procedure of NoCoding is explained in Section 4.1, which does not use any encoding in the strict sense but is important to compare due to its simplicity. After that, the frequently used coding scheme of random linear network coding (RLNC) is introduced in Section 4.2. Subsequently, the coding procedure called Growth Codes, which forms the basis for BCGC, is described in detail in Section 4.3. This chapter finally closes with a section on related work concerning network coding based data dissemination in Section 4.4, which includes the distribution process and distributed storage.

As defined in Section 2.1, data which is generated by a source node is called a *symbol*. For the following descriptions, it is assumed that there are n distinct original symbols in the network which should eventually be received and reconstructed by each sink. Links can be lossy, but this does not influence the results in the following as only the packets arriving at a sink are considered. It is determined to what extent these received packets contribute to reconstruct the n original symbols when using a respective coding scheme.

4.1 NoCoding/Forwarding

In this section, the procedure of NoCoding (NoC) is introduced. NoCoding is also called Forwarding as data is only forwarded without any modification. The principle of NoC is explained first. Subsequently, stochastic correlations in NoC are discussed which can later be compared with the results of the other coding schemes. Then, the structure of data packets in NoC is presented. This section closes with a brief description of the benefits and limitations of NoC.

4.1.1 Principle of NoCoding

In the case of NoCoding, a codeword consists of exactly one original symbol in a non-encoded form. This means, a symbol is sent as payload in a packet without being encoded or combined with other symbols. Here, the term ‘codeword’ is used for the sake of uniform formulation.

After a packet was received by an intermediate node, the contained payload, which consists of exactly one symbol, is stored by the node together with the ID of the symbol as a (symbol, ID)-pair. The ID, which can be retrieved from the packet header, is necessary to be able to identify the symbol. If a node has to generate a new packet for transmission, it randomly chooses a symbol from its storage and uses this as the payload of the new packet. Here, not the most recently received symbol is used, but a random previously received symbol is chosen. The corresponding ID is included in the packet header. In principle, NoC equals the selected routing procedure, see Section 2.4, without further encoding as received data is forwarded to the neighbors without any modifications.

As described in the system setup in Section 3.1, the aim of each sink is to have gathered and stored all original symbols in the end. For this purpose, it is not sufficient to receive n arbitrary packets. Instead, for each symbol, a packet has to be received which contains exactly this symbol as payload. If only one symbol is still missing, a packet with this last symbol as payload has to be received. Each different packet is not helpful and has to be discarded.

4.1.2 Stochastic Correlations

Formulas for the so-called Coupon Collector's problem can be used to determine how many packets have to arrive on average at a sink in order to gather all or a certain number of distinct original symbols. In the original Coupon Collector's problem, which is described, for example, in [MR95], there are n different coupons which can eventually be collected. Packages, which consist of c random coupons, have to be bought for this purpose. Each coupon has the same probability to be included in a package. It should be determined how many packages have to be bought on average to collect all n coupons if no coupon has been collected before. This scenario can be transferred to NoC for the case of $c = 1$ as each received NoC packet only contains one original symbol.

Assume that each of the n existing distinct symbols has the same probability to be contained in a packet which is received by a sink. Let p_r be the probability that a received packet contains an original symbol which has not been received by the considered sink yet if r distinct original symbols have already been collected and stored by this sink. According to the formula in a so-called Laplace-Experiment, see for example [Hen08], it can be seen that

$$p_r = \frac{n-r}{n}.$$

This can be concluded as there are n different symbols which can be included in a packet, where $n-r$ of those possibilities yield a new symbol. This formula for p_r can also be found in [MR95] for the Coupon Collector's problem. Assume r distinct original symbols have already been collected and stored in a considered sink. Let X_r then be a discrete random variable denoting the number of packets which have to be received until a new symbol is collected. Transferred from [MR95] to our scenario, the expected number of packets which have to arrive at a sink in order to gather a new symbol is

$$E[X_r] = \frac{1}{p_r} = \frac{n}{n-r}.$$

In the following, it is assumed that a sink does not have any original symbol at the beginning even though the node which is directly connected to this sink stored its own symbol at initialization. Thus, to gather a first symbol, an expected number of $E[X_0] = \frac{n}{n} = 1$ packet has to be received by a sink. For a second symbol, $n-1$ symbols are still missing, so $E[X_1] = \frac{n}{n-1}$, and so on. In total, an expected number of

$$E[\#packets] = \sum_{r=0}^{n-1} \frac{n}{n-r} = n \cdot \sum_{r=0}^{n-1} \frac{1}{n-r} = n \cdot \sum_{i=1}^{n} \frac{1}{i} = n \cdot H_n \tag{4.1}$$

packets has to arrive at a sink in order to collect and store all n distinct original symbols. Here, the n-th harmonic number H_n is contained. An analog to Equation (4.1) can also be found in [MR95] for the Coupon Collector's problem. The expected value $E[\#packets] = n \cdot H_n$ is bounded by $E[\#packets] < n \cdot \left(ln(n) + \gamma + \frac{1}{2n} \right)$, with the Euler–Mascheroni constant γ, see for example [PS72; You91]. Hence,

$$E[\#packets] \in O(n \cdot \log(n)), \tag{4.2}$$

which means that a sink has to receive $O(n \cdot \log(n))$ packets until it has finally collected all n distinct original symbols.

4.1.3 Data Packets in NoC

An NoC packet contains one non-encoded, original symbol as payload. If the size of an original symbol is b bits, the payload of an NoC packet consists of $\lceil b/8 \rceil$ Bytes. Here, a representation in Bytes is used for comparability with the results from the testbed. In order to be able to identify the contained symbol, the corresponding ID has to be included in the packet header. Assuming a 16-bit short address as explained in Section 3.2.2, the size of an NoC header is only 2 Bytes. To sum up, an NoC frame contains:

- the ID of the contained symbol (2 Bytes) and
- one symbol (payload) ($\lceil b/8 \rceil$ Bytes).

Therefore, the total NoC frame overhead is 2 Bytes, and the relative overhead is $2/(\lceil b/8 \rceil)$. The structure of an NoC frame, which represents the MAC Data Payload of Figure 3.2, is additionally depicted in Figure 4.1. Please note that the proportions in the figure do not reflect the actual size ratios.

4.1.4 Benefits and Limitations of NoC

At the end of this section on NoCoding, benefits and limitations are briefly described. One advantage of NoC is that it requires a very small packet header. Thus, NoC packets and consequently the transmission times are short. This is advantageous especially if there are many communicating participants in the system. Such a scenario is prone to packet collisions or congestion and less collisions occur if all packets are shorter. In

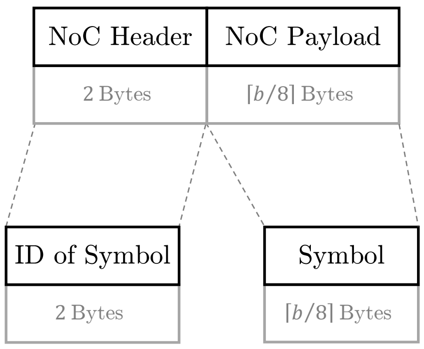


Fig. 4.1: NoC frame format

general, latency can be reduced if transmission times are shorter. A further advantage of NoC is the ease of implementation. It is not necessary to implement complex encoding or decoding. Instead, a symbol can be included in a packet without modification and be read directly from a packet. Furthermore, a node does not have to perform computationally complex encoding or recoding in order to create a packet for transmission. A sink does not have to perform complex decoding either. Thus, the computational effort in the nodes is low with NoC.

However, there are also crucial drawbacks with NoC. If a node receives a packet from a neighbor, then at least this neighbor already knows the contained symbol. If additionally both nodes have a common neighbor, then this neighbor also knows the symbol. So if the node sends this symbol later in a packet, it is redundant and, thus, useless for the neighbor from whom it originally received the symbol and for all common neighbors. Furthermore, for each symbol, a considered sink has to receive a packet which contains exactly this symbol as payload in order to be able to gather all distinct original symbols in the end. In the beginning, many received packets contain a new symbol for a sink as no or only few symbols have already been stored by this sink. At some point, however, more and more redundant packets arrive, and especially waiting for the last missing symbol can take a very long time. Thus, a large number of redundant packets is received, which are unnecessary and waste resources. As many redundant packets arrive with NoC, many packets have to be received in total at a sink in order to gather all original symbols. In particular, in a system with many participants which transmit packets, it is desirable to send as few redundant packets as possible as unnecessary transmissions only increase the probability of collisions, energy consumption, and latency.

In summary, NoC packet headers are short but a large number of packets has to be received at a sink in order to finally gather all original symbols. The advantages of short packets are, therefore, possibly overlaid by the disadvantages of the big number of packets, depending on the payload size.

4.2 Random Linear Network Coding

Random linear network coding (RLNC) was introduced by Ho et al. in [Ho+03a]. In the following, the principle of RLNC is described first. Then, basics on Galois fields, on which RLNC is based, as well as details on encoding and decoding in RLNC are presented. Subsequently, possible choices for the Galois field are discussed and the structure of RLNC data packets is explained. Finally, benefits and limitations of RLNC are described in detail.

4.2.1 Principle of RLNC

RLNC represents a form of linear network coding, which means a codeword is the result of a linear combination, where coding coefficients are chosen randomly.

Intermediate nodes store the contained payload of received packets, i.e. the respective codeword, and relevant parts of the header. They calculate a linear combination of all codewords in their memory in order to create a new codeword for transmission. If only individual symbols are used for the linear combination, the procedure is called *encoding*. For RLNC, it is also possible to calculate a linear combination of codewords instead of using only individual symbols. In this case, the encoding procedure is also called *recoding* or *re-encoding*. A generated codeword is subsequently transmitted as the payload of a packet with all relevant information for decoding in the packet header. This information consists of the coefficients which were used for the linear combination. The vector of these coefficients is called the *encoding vector* of the linear combination. In the case of RLNC, sink nodes are the only nodes which perform decoding.

The procedure of RLNC always refers to a predefined Galois field $GF(q)$. Both symbols and coefficients, and therefore also codewords, are based on $GF(q)$. For this reason, basics on Galois fields are presented in the following before details on encoding and decoding in RLNC are described.

4.2.2 Basics on Galois Fields

A Galois field $GF(q) = (F, +_F, *_F)$ is also called a finite field. It denotes a set F of $q \in \mathbb{N}$ elements with two operations on this set, addition $+_F$ and multiplication $*_F$ so that $(F, +_F)$ and $(F \setminus \{0\}, *_F)$ form Abelian groups. Here, 0 denotes the identity element of the addition $+_F$. Furthermore, the distributive law is valid in a Galois field $GF(q)$. Thus, multiplication is distributive over addition, which means $\forall a, b, c \in F \quad a *_F (b +_F c) = (a *_F b) +_F (a *_F c)$. An Abelian group $(F, +_F)$ is a set of elements F and an operation $+_F$ on this set with the following properties:

(i) Commutativity, i.e. $\forall a, b \in F \quad a +_F b = b +_F a$,

(ii) Closure, i.e. $\forall a, b \in F \quad a +_F b \in F$,

(iii) Associativity, i.e. $\forall a, b, c \in F \quad (a +_F b) +_F c = a +_F (b +_F c)$,

(iv) Identity, i.e. $\exists 0 \in F\ \forall a \in F \quad a +_F 0 = 0 +_F a = a$, and

(v) Invertibility, i.e. $\forall a \in F\ \exists a' \in F \quad a +_F a' = 0$.

The respective axioms can be formulated analogously for $(F \setminus \{0\}, *_F)$. The identity element 1 of the multiplication $*_F$ should then be used instead of the identity element 0 of the addition $+_F$. The two operations, addition $+_F$ and multiplication $*_F$, have to be defined appropriately to the respective set F so that the field axioms are fulfilled. It can be said that addition and multiplication are performed over F. Subtraction is defined as addition with the additive inverse, and division is performed as multiplication with the multiplicative inverse.

The number of elements q of a Galois field $GF(q)$ is always a prime power, i.e. $q = p^m$ with a prime number p and $m \in \mathbb{N}$. Furthermore, up to isomorphism, there is one and only one Galois field with exactly q elements. The binary field $GF(2)$ is the simplest Galois field. It only consists of the elements 0 and 1. There is no Galois field with fewer elements. For addition and multiplication in $GF(2)$, modulo-2 addition and multiplication are used. Computing in a Galois field larger than $GF(2)$ can be done by direct calculation modulo a suitable irreducible polynomial. Alternatively, lookup tables can be used to retrieve the solution of a product or sum instead of calculating it directly. However, for large Galois fields, lookup tables become big, require a corresponding amount of storage space, and searching for entries takes time.

These definitions and propositions as well as further information about Galois fields can be found in [Ksc12] or in common literature on linear algebra, see for example [Beu13; Kow13].

4.2.3 Encoding and Decoding for RLNC

After having introduced Galois fields, encoding/recoding and decoding in RLNC will be briefly presented next. In this context, the term of *linear independence* is relevant and is, therefore, explained in advance. Vectors are called *linearly independent* if the zero vector can only be generated by a linear combination of these vectors where all coefficients are 0. Otherwise, the vectors are called *linearly dependent*. Considering vectors of the dimension n, i.e. vectors with n elements, only a maximum number of n vectors can be linearly independent. More than n vectors are always linearly dependent. These and further details about linearly independent vectors can be found in any common literature on linear algebra, such as [Beu13; Kow13]. For the sake of brevity, the expression ‘linearly independent packets’ will be used in the following sections on RLNC and refers to the encoding vectors contained in the packets.

Let $x_1, x_2, ..., x_n$ with $n \in \mathbb{N}$ and $x_i \in (GF(q))^l$ for $i = 1, ..., n$ with $l \in \mathbb{N}$ be the original symbols which have been generated and should eventually arrive at each sink. Each symbol is a vector of the dimension l, and $GF(q)$ is the chosen Galois field. Furthermore, let $w_1, w_2, ..., w_K$ with $K \in \mathbb{N}$ and $w_j \in (GF(q))^l$ for $j = 1, ..., K$ be the symbols or codewords which were received and stored by a considered intermediate node. This node can then create a new codeword for transmission via the linear combination

$$y = \sum_{j=1}^{K} \alpha_j \cdot w_j. \tag{4.3}$$

Here, coefficients $\alpha_j \in GF(q)$ for $j = 1, ..., K$ are chosen uniformly at random from the Galois field $GF(q)$ [Ho+03a]. The choice of the coefficients is performed by each node independently. Addition and multiplication are performed over $GF(q)$. The encoding according to Equation (4.3) is a recoding if not only original symbols but also codewords are used for the linear combination. Recoding can be performed without any previous decoding. A codeword for transmission, however, has to be specified with respect to the original symbols $x_1, x_2, ..., x_n$ in order to enable decoding at a sink. Therefore, a transformation of the coefficients α_j for $j = 1, ..., K$ is necessary before a codeword can be sent as payload together with the coefficients in the packet header. The final linear combination then is

$$y = \sum_{i=1}^{n} c_i \cdot x_i, \tag{4.4}$$

with $c_i \in GF(q)$ for $i = 1, ..., n$. Depending on the dimension l of a codeword, the linear combination in Equation (4.4) can also be represented as a multiplication of vectors or matrices: $y = (c_1, ..., c_n) \cdot (x_1, ..., x_n)^T$. The vector $(c_1, ..., c_n)$ of the used coefficients is called the *encoding vector* of y. This encoding vector is transmitted together with the respective codeword y in a packet. If this packet is received by a node, the codeword is stored together with the encoding vector as a pair and subsequently used for recoding. For the transformation of the coefficients from Equation (4.3) to Equation (4.4), the randomly chosen coefficient α_j and the respective encoding vector $(c_1^j, ..., c_n^j)$ of the codeword or symbol w_j is used for each $j = 1, ..., K$. The mentioned encoding vector $(c_1^j, ..., c_n^j)$ refers to the creation of w_j, i.e. $w_j = \sum_{i=1}^{n} c_i^j \cdot x_i$. More details on the transformation can be found, for example, in [DMC06]. If w_j is an original symbol, all but one component of the corresponding encoding vector $(c_1^j, ..., c_n^j)$ of w_j are 0. The remaining component is 1, so the 'codeword' w_j is the same as the original symbol. An intermediate node can, therefore, perform encoding/recoding regardless of the type of packets it previously received and has stored. A newly created codeword y is transmitted together with its final encoding vector in a packet.

The dimension l of an original symbol depends on the size of the symbol and the used Galois field $GF(q)$, based on which RLNC is performed. Let the size of a symbol be b bits. An element of $GF(q)$ can have a value between 0 and $q - 1$ and can, thus, represent up to $\log_2 q$ bits. Therefore, a vector of the dimension $l = \lceil \frac{b}{\log_2 q} \rceil$ is required to represent a symbol of b bits. For $\lceil \frac{b}{\log_2 q} \rceil > 1$, the considered symbol is a row vector as the possible range of numbers is too big to be represented by a single element. The coefficients of the linear combination are scalars from $GF(q)$ and can, hence, be represented by exactly one element of $GF(q)$ each. They have a value between 0 and $q-1$ and, thus, require $\log_2 q$ bits each. As can be seen in Equation (4.4), a codeword contains several original symbols. However, the combination of these symbols is a linear combination and not a concatenation. As the coefficients are scalars from the Galois field $GF(q)$, the codeword y has the same size as any original symbol x_i. If the symbols are row vectors, the codewords are row vectors of the same dimension. Therefore, symbols and codewords are vectors of length l over the Galois field $GF(q)$. For the sake of simplicity, it is assumed in the following that $l = 1$, i.e. $x_i \in GF(q)$ for $i = 1, ..., K$. The extension to $l > 1$, however, is straightforward.

After encoding has been described, decoding will be considered next. A received codeword and the corresponding encoding vector, i.e. the coefficients of the linear combination, can be interpreted as a linear equation. Here, the unknowns are the n original symbols. If there are n original symbols, at least n equations, i.e. (codeword, encoding vector)-pairs, have to be received to be able to solve the system of linear equations and reconstruct all original symbols. Each sink gathers received packets and stores the contained codeword and corresponding encoding vector row by row in a so-called extended *decoding matrix*. As soon as n linearly independent packets have been received by a considered sink, i.e. the respective encoding vectors are linearly independent, the decoding matrix has full rank. This means, the decoding matrix is invertible and, thus, the system of linear equations is solvable. The sink can then start decoding by matrix inversion via Gaussian elimination and reconstruct the n original symbols. A packet which is linearly dependent on previously received packets does not contribute to solving the system of linear equations since no new information is contained. A linearly dependent packet does not increase the rank of the decoding matrix. Therefore, n linearly independent packets have to be received, which implies that more than n packets may be needed. For RLNC, it is not relevant which specific packets are received as long as they are linearly independent. At least n packets are necessary, and any n linearly independent packets are sufficient to reconstruct all original data. In the optimal case, exactly n received packets are needed to reconstruct the n original symbols. More information on the probability to receive n linearly independent packets and on the expected number of packets which have to be received to be able decode can be found in Section 4.2.4.

As decoding corresponds to solving a system of linear equations, which is typically done via Gaussian elimination, it generally has a complexity of $0(n^3)$, as stated, for example, in [TB97].

The calculation of a codeword should be based on a Galois field. Otherwise, the linear combination could result in packets with very large payload as codewords may become large. Also the randomly chosen coefficients could become large, which results in a large packet header. The choice of the size of the Galois field is crucial for the performance of RLNC, which will be discussed in the following.

4.2.4 Galois Field Size Analysis and Stochastic Correlations

For RLNC, the chosen Galois field $GF(q)$ should not be too small in order not to receive many linearly dependent packets, which have to be discarded. Packets which are linearly dependent on a sink's already stored packets do not provide new information for this sink and do not bring any benefit for the reconstruction of the original symbols. In the case of linearly dependent packets, more than the minimum number n of packets have to be received by the considered sink to reconstruct the n original symbols. This means, the multicast capacity is not achieved. However, the probability that there are n linearly independent packets among n received packets, i.e. that n packets are sufficient, approaches 1 with increasing value of q. This will be shown in the following.

The larger the Galois field, the higher the probability is that a received packet is linearly independent of the already stored packets in the considered sink. If there are $n-1$ stored packets and a further packet is received, the probability that the n vectors are linearly independent is

$$p_{n-1} = \frac{q^n - q^{n-1}}{q^n} = \frac{q-1}{q} = 1 - \frac{1}{q}. \tag{4.5}$$

This can easily be proved and, for example, can also be extracted from a derivation in [TBF11]. For any number j of previously stored packets with $j \leq n-1$, there are at least as many possibilities for a linearly independent incoming packet as in the case of $n-1$ stored packets, which can easily be shown. Therefore, the probability p_j that a newly received vector is linearly independent of the previously stored j vectors is

$$p_j \geq 1 - \frac{1}{q}. \tag{4.6}$$

With $GF(q)$, the probability $p_{dec}(q,n,n)$ to have n linearly independent packets among n received packets can be calculated via

$$p_{dec}(q,n,n) = \prod_{i=1}^{n}(1 - \frac{1}{q^i}) = \prod_{i=1}^{n} \frac{q^i - 1}{q^i}, \tag{4.7}$$

see for example [TBF11]. This formula can be derived by multiplying the probability that the first packet is linearly independent, i.e. not the zero vector, by the probability that the next received packet is linearly independent of the first vector. Then, for each subsequent packet, the probability that a received packet is linearly independent of all previously stored packets is multiplied. This product ends at the probability that a received packet is linearly independent of the $n-1$ linearly independent packets which have already been stored. These single probabilities can be bounded according to Equation (4.6) so that with Equation (4.7) it can be concluded that

$$p_{dec}(q,n,n) \geq (1-\frac{1}{q})^n, \tag{4.8}$$

which can also be derived from [Ho+03a; Ho+03b; Ho+06] for one receiver. This means, the probability that a sink is able to reconstruct all n original symbols if it has received n packets is at least $(1-\frac{1}{q})^n$.

It can be seen that $p_{dec}(q,n,n)$ increases with the size q of the chosen Galois field as there are more possibilities to randomly pick the coefficients. The probability $p_{dec}(q,n,n)$ can become arbitrarily close to 1 by choosing a sufficiently large size q of the Galois field, which can be seen from Equation (4.7) and is also shown in [Ho+06], for example. According to [TBF11], the choice of n does not have a significant impact on the probability $p_{dec}(q,n,n)$, which is valid for larger n. As shown in [TBF11], this probability is close to 1 even for small values of q, such as $q = 32 = 2^4$. For example, for $q = 2^1$, $q = 2^4$, or $q = 2^8$ and $n = 256$ the probability to be able to reconstruct the n original symbols after having received n packets is $p_{dec}(q,n,n) \approx 0.288788$, $p_{dec}(q,n,n) \approx 0.933595$, or $p_{dec}(q,n,n) \approx 0.996078$, respectively.

Furthermore, the probability $p_{dec}(q,n,k)$ to have n linearly independent packets among a fixed number of $k \geq n$ received packets, which is also called the *decoding probability*, can be calculated via

$$p_{dec}(q,n,k) = \prod_{i=0}^{n-1}(1-\frac{1}{q^{k-i}}) = \prod_{i=0}^{n-1}\frac{q^{k-i}-1}{q^{k-i}}, \tag{4.9}$$

see for example [TBF11]. Equation (4.7) is a special case of Equation (4.9) with $k = n$.

As for Equation (4.7), the fact that the choice of n does not make a big difference to the results is also true for the more general Equation (4.9). Moreover, for a fixed, arbitrary number $k \geq n$ of received packets, the decoding probability $p_{dec}(q,n,k)$ is higher the bigger q is [TBF11]. However, the larger k is, the smaller the difference in $p_{dec}(q,n,k)$ is as all probabilities $p_{dec}(q,n,k)$ approach the value 1 as k increases, which can be seen on the plots in [TBF11]. Since Equation (4.7) is a special case of Equation (4.9), the above conclusion also applies to $p_{dec}(q,n,n)$.

This means, with a larger q, less packets are needed to achieve the same decoding probability $p_{dec}(q,n,k)$ and $p_{dec}(q,n,n)$. Thus, probably less packets have to be received to be able to reconstruct all n original symbols if the size q of the Galois field is bigger.

The expected number of packets $E[\#packets]$ which have to be received to have n linearly independent packets, i.e. to be able to reconstruct the n original symbols, is

$$E[\#packets] = E[k] = \sum_{i=1}^{n}\frac{1}{1-(1/q)^i} = \sum_{i=1}^{n}\frac{q^i}{q^i-1}, \tag{4.10}$$

see for example [EOM06]. The larger q is, the closer $E[\#packets]$ is to n, which can be seen from Equation (4.10) or found in, for example, [EOM06; LMS09]. The expected value $E[\#packets]$ can become arbitrary close to n if q is accordingly large. Thus, if q is large enough, it is sufficient to receive n packets on average in order to be able to reconstruct all n original symbols [EOM06]. In the optimal case, exactly n packets are needed, which corresponds to achieving the multicast capacity. However, this optimum cannot be reached with randomly chosen coefficients in practice as the probability for linearly dependent packets is never 0. According to [LMS09], the expected value is upper bounded by

$$E[\#packets] \leq min\{n \cdot \frac{q}{q-1}, n+1+\frac{1-q^{-n+1}}{q-1}\}, \tag{4.11}$$

which is again bounded by $n+2$. Thus, the following applies to the expected value:

$$n \leq E[\#packets] \leq n+2, \tag{4.12}$$

see [LMS09]. This means, between n and $n+2$ packets have to be received on average in order to reconstruct the n original symbols. The field size q and the number of original symbols n do influence the expected value $E[\#packets]$. However, no matter how large q or n are, only a maximum number of 2 additional packets have to be received on average in order to reconstruct the original symbols, see Equation (4.12). The larger n is, the less these extra packets account for on a percentage basis, so the extra packets can be considered negligible for large n. From Equation (4.12), it can be concluded that $E[\#packets] \in \Omega(n)$ and $E[\#packets] \in O(n)$. Thus,

$$E[\#packets] \in \Theta(n), \tag{4.13}$$

which means that a sink has to receive $\Theta(n)$ packets until it is finally able to reconstruct all n original symbols.

However, reality can deviate significantly from the expected value. According to the system setup in Section 3.1, each node knows only its own symbol at the beginning. Even in general, usually, not all symbols are stored in each intermediate node. Therefore, an intermediate node is not able to choose all n coefficients at random with uniform distribution as some of the coefficients can only have the value 0. This increases the probability for linearly dependent packets. Another reason for the deviation from the expected value can be the following situation. Nodes may need to calculate new linear combinations for the next transmissions even if they have not received any new information since generating the previous packets. Especially if only few packets containing few symbols each were received before, this often leads to the reception of several linearly dependent packets at the considered sink. Thus, accordingly more than n packets have to be received to reconstruct the n original symbols.

To conclude, coefficients have to be chosen at random from a sufficiently large Galois field. Thereby, the probability that a packet which is received by a sink is linearly independent of the already stored packets in this sink is high. Furthermore, also the decoding probability $p_{dec}(q,n,k)$ to have n linearly independent packets among a fixed number of $k \geq n$ received packets is then close to 1. The probability $p_{dec}(q,n,n)$ to be able to reconstruct the n original symbols after having received n packets is also close to 1 if q is chosen sufficiently large. Thus, probably less packets have to be received to be able to reconstruct all n original symbols.

However, the larger the Galois field is, the more complex encoding and decoding is. This is due to, for example, multiplication modulo of an irreducible polynomial and, in particular, matrix inversion. For $GF(2^8)$, for example, lookup tables can be used for addition and multiplication, which is beneficial especially for computationally weak sensor nodes. Lookup tables, however, require storage space in the nodes and searching within tables also takes time. In $GF(2^8)$, the storage space for one lookup table is about $q^2 \cdot \log_2 q$ bit $\approx$ 66 kBytes, which still can be reduced. In a Galois field $GF(q)$ with $q = 2^{16}$ or bigger, the storage space for lookup tables would be more than 8 GBytes. Moreover, searching for values in these big tables would also take a long time so that a direct calculation is preferable despite the computing effort. With a larger Galois field, fewer packets are needed to reconstruct all symbols, but a larger Galois field also implies that coefficients are bigger. Therefore, the respective encoding vector in the packet header, i.e. the overhead, is bigger and, thus, also the size of each packet. As a consequence, more storage space is needed to store a certain number of packets in a node and the transmission time of each packet is increased.

Thus, when choosing the Galois field, a compromise must be made between computational complexity/storage space for lookup tables and packet size on the one hand and probability of linearly independent packets and, hence, RNP on the other. However, it is not necessary to choose a very large Galois field in order to achieve a high decoding probability. Already with $GF(2^8)$ the decoding probability $p_{dec}(q,n,n)$ is almost 1, for example, for $n = 256$, see Equation (4.9) and [TBF11]. Thus, with a bigger Galois field there is hardly any improvement possible. According to literature, see for example [FLW06; CW07; Méd+14], the Galois field $GF(2^8)$ suffices in most cases, and often even $GF(2)$ [Méd+14]. Typical Galois field sizes are $q = 2^1$ or $q = 2^8$ as in these cases exactly 1 bit or 1 Byte, respectively, are needed for the representation of a coefficient. Thus, in the following chapters, RLNC is considered where coefficients are chosen uniformly at random over the Galois field $GF(2)$ or $GF(2^8)$.

4.2.5 Data Packets in RLNC

As already mentioned, the choice of the Galois field influences the size of a packet in RLNC. For the following values, it is assumed that there are n distinct original symbols, which were generated and should eventually be reconstructed by each sink. An RLNC

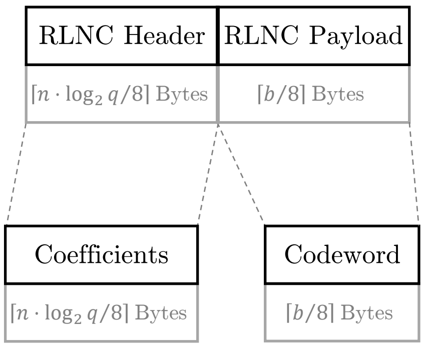

RLNC Header
RLNC Payload
$\lceil n \cdot \log_2 q/8 \rceil$ Bytes
$\lceil b/8 \rceil$ Bytes
Coefficients
Codeword
$\lceil n \cdot \log_2 q/8 \rceil$ Bytes
$\lceil b/8 \rceil$ Bytes

Fig. 4.2: RLNC frame format

packet contains the codeword, which was calculated via linear combination and should now be sent, as payload. If the size of an original symbol is b bits, the payload of an RLNC packet has $\lceil b/8 \rceil$ Bytes as a codeword has the same size as an original symbol. Here, a representation in Bytes is used. The corresponding coefficients of the linear combination are included in the packet header to enable decoding at a sink. An RLNC packet header then contains all n coefficients, i.e. $\lceil (n \cdot \log_2 q)/8 \rceil$ Bytes for $GF(q)$. To sum up, an RLNC frame contains:

- the n coefficients ($\lceil (n \cdot \log_2 q)/8 \rceil$ Bytes) and
- one codeword (payload) ($\lceil b/8 \rceil$ Bytes).

The total RLNC frame overhead is $\lceil (n \cdot \log_2 q)/8 \rceil$ Bytes, and the relative overhead is $(n \cdot \log_2 q)/b$. For example, the total overhead of an RLNC frame is $\lceil (n \cdot 1)/8 \rceil$ Bytes if using the Galois Field $GF(2)$ and $\lceil (n \cdot 8)/8 \rceil$ Bytes $= n$ Bytes for $GF(2^8)$. Thus, in the case of $n = 256$, it is 32 Bytes or 256 Bytes, respectively. With $n = 1024$, the total RLNC frame overhead is already 128 Bytes or 1024 Bytes, respectively. This means, for a large-scale system of n nodes where each node generates a symbol, as addressed in Section 3.1, RLNC easily exceeds the limit of 116 Bytes given in IEEE Std 802.15.4 if no variant of header compression is used. The structure of an RLNC frame, which would represent the MAC Data Payload of Figure 3.2, is depicted in Figure 4.2. Again, the proportions in the figure do not reflect the actual size ratios.

Besides the transmission time, the packet size also influences the storage space required in a node. For recoding, each intermediate node has to store a certain amount of received (codeword, encoding vector)-pairs. In the following, the (codeword, encoding vector)-pair contained in a packet will itself be referred to as a packet for simplicity. Depending on the implementation, it is possible to store, for example, all received packets, all received linearly independent packets, a fixed number of received packets, or a fixed number of linear combinations of received packets. No matter which of these variants is chosen, the storage space required in the node is always proportional to the number of symbols n and strongly influenced by the chosen Galois field.

4.2.6 Benefits and Limitations of RLNC

The application of RLNC has many benefits, which is why this coding technique is widely used. A first advantage of RLNC is that no knowledge of neighboring nodes or previous transmissions is necessary as coefficients are chosen randomly. Due to the random, independent choice of the coefficients, a distributed implementation is easily possible and no coordination between nodes is required. In addition, no adjustments are needed for a dynamic scenario because of the random selection of coefficients. RLNC is well suited to dynamically changing networks.

Furthermore, RLNC possesses inherent security as it is usually not possible to reconstruct any part of the original data unless n linearly independent packets have been received. This means that eavesdroppers usually cannot retrieve any information.

Another advantage of RLNC is the fact that recoding can be done without previous decoding. In this way, new linear combinations can be generated and new data can be integrated into codewords without the need for computationally complex decoding at intermediate nodes. This increases the probability that a packet is linearly independent of the packets that have already been received by the considered sink. Only sinks need to perform decoding to retrieve the original symbols.

To reconstruct all n original symbols, RLNC requires exactly n linearly independent packets. The bigger the chosen Galois field is, the closer the expected number of required packets is to n and the higher the probability is that n packets are sufficient. As shown, for example, in [Ho+06], RLNC theoretically achieves the max-flow bound in a multi-source multicast network for a sufficiently large Galois field. However, even with a small Galois field and despite randomly chosen coefficients, only a few more than n packets are needed in expectation, which is a fourth, crucial benefit of RLNC. In the case of a lossy network, this reduces latency and increases energy efficiency as less packets have to be sent/received with RLNC. Reliability is also increased as packets are allowed to get lost and reconstruction is still successful as long as any n linearly independent packets are received by a considered sink. However, for example, a certain high degree of mobility is required in a realistic system to achieve that $E[\#packets] \approx n$ is valid for the expected number of required packets, even in the case of a bigger q. In general, Equation (4.10) is only valid for a considered sink if all coefficients in the received packets had been chosen at random with uniform distribution. In reality, however, a neighbor can only calculate and send a linear combination of the symbols which are available in its stored received packets. Therefore, depending on the situation, reality can deviate greatly from Equation (4.10) and significantly more packets may have to be received by a sink. Equation (4.10) can at most be approximately valid if there is, for example, a lot of mobility in the system and, thus, the probabilities of the symbols and, hence, of the coefficients in received packets are distributed more equally.

Besides the advantages of RLNC, there are also some critical drawbacks. For example, the header of a packet may be very large, depending on the selected Galois field and the number n of nodes/distinct original symbols in the network. All n coefficients which are used in the linear combination have to be listed in the header if no additional header compression is used. These coefficients are necessary to decode the respective codeword. The size of a coefficient is, as described in Section 4.2.4, dependent on the chosen Galois field. Therefore, the RLNC packet header does not scale well with increasing number of nodes in the system. Additionally, packet size is constant within the algorithm as always all n coefficients have to be included in the header. This encoding vector overhead can be neglected if the payload is big enough. However, if a packet should meet the requirements of IEEE Std 802.15.4, see Section 3.2, the available space is quickly filled by the header only. Especially if there are many distinct original symbols, i.e. a large-scale system is considered, and coefficients exceed a certain size, there is little or even no space left for payload. For example, in the case of $n = 1024$ and $GF(2)$, the coefficients require 1024 bits = 128 Bytes in the header. Larger packets imply a longer transmission time, which can lead to delays, i.e. an increased latency, a higher probability of collision, and more energy is consumed by the transmission of a packet. Furthermore, larger packets require more storage space at the intermediate nodes. Thus, the advantage of the smaller number of required packets may be outweighed by the disadvantage of larger packets. Especially if, e.g. due to size restrictions, only a small payload is possible compared to the overhead, RLNC has performance issues.

A second disadvantage of RLNC is the computational complexity for encoding and decoding, which depends on the selected Galois field. Small and cost-efficient mobile sensor nodes often do not support complex calculations. Algorithms for processing and distributing data, therefore, should be simple. In particular, decoding in RLNC is a challenge for small nodes as an expensive Gaussian elimination has to be performed here for the general case. Complex calculations can be replaced by lookup tables, but these in turn require storage space and time to read the results from the tables. Due to the additional challenge of large packets, RLNC might not be applicable for large-scale systems of small, low-cost sensor nodes with low processing power.

A third disadvantage with regard to RLNC is the fact that at the beginning nothing is decoded for a while and then everything is decoded at once. This results in a step curve for the number of original symbols reconstructed at a given time. In certain application scenarios, however, it may not be relevant to reconstruct all original symbols in a sink, but with RLNC only all data or no data at all can be reconstructed. In particular, with RLNC, it is not possible to reconstruct some of the symbols in advance. Especially in the case of node failure or early switching to sleep mode, for example, sometimes less than n linearly independent packets arrive at a sink in the end. Then, this sink is not able to reconstruct any original symbol.

For differences between RLNC and BCGC, see Section 5.6. Further comprehensive information on RLNC can be found, for example, in [HL08; MS12].

4.3 Growth Codes

Growth Codes (GC) form an important basis for BCGC, and were developed by Kamra et al. in [Kam+06b]. Technical details and proofs were published in [Kam+06a]. In the following, GC are described according to [Kam+06b; Kam+06a]. However, additional details which cannot be found in the original article but are needed for BCGC are also presented here.

There are certain features which are similar for GC and the well-known Luby Transform (LT) codes [Lub02]. These are 1) encoding is performed via XOR, which is also the case e.g. for RLNC with $GF(2)$, 2) codewords are composed/re-encoded according to a proposed degree, which is more a term than a concept and can be applied to various coding schemes, 3) the degree is based on a so-called degree distribution, where for LT codes an ideal/robust soliton distribution is used and for GC the GC degree distribution. Especially the last aspect makes a significant difference between the two encoding schemes. In the case of LT codes, the degree of a codeword for transmission is chosen according to the probabilities associated with the ideal/robust soliton distribution. This means that the degrees are not specifically ordered but only the distribution is prescribed. In contrast, for GC, codewords of degree 1 are generated in the beginning, codewords of degree 2 are generated later, etc. In the course of the algorithm, the codeword degree is monotonically increasing, which is a key characteristic of GC and the reason for its name.

4.3.1 Principle of Growth Codes

GC are a special form of network coding. They were developed for distributed data collection and data storage in highly dynamic, wireless sensor networks (WSNs). The aim of GC is to keep data dependably despite possible node failures. If there are n nodes in a network, each node generates its own symbol, so there will be n distinct original symbols. In [Kam+06b; Kam+06a], there is only one dedicated sink, which aims to gather and reconstruct these symbols. Dependability is achieved by replicating each node's own symbol and spreading those replications across the network. In the case of GC, a codeword consists of an XOR-combination of different symbols. Under certain circumstances, an intermediate node's own symbol is added to a codeword which will be transmitted within the next packet. Therefore, the number of symbols in a codeword grows continuously. Packets are transmitted via bidirectional data exchange with a randomly selected neighboring node. Eventually, all n symbols, which were generated by the n nodes of the network, should be received and reconstructed by the single sink.

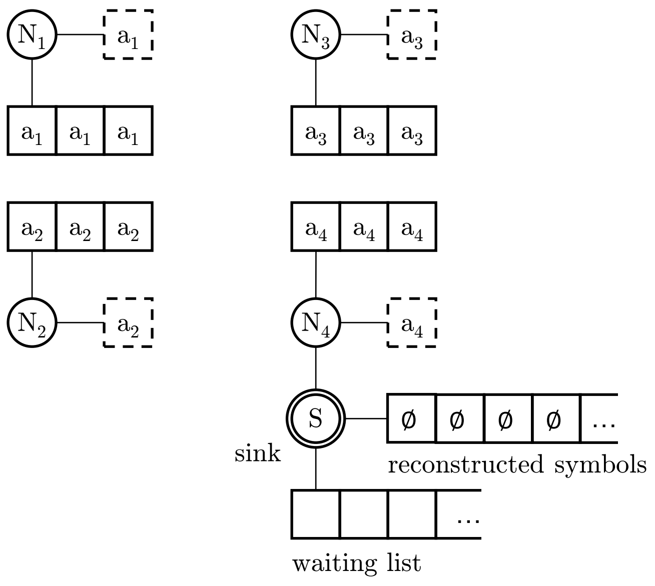


Fig. 4.3: System at initialization

The aim of GC is to be able to reconstruct as many originally generated symbols as possible at any time, i.e. after having received any number of packets. In the following, the phrases 'receiving a packet which contains a certain codeword' and 'receiving a codeword' are used interchangeably since each packet contains exactly one codeword.

For GC, nodes only have to know their current direct neighbors for communication. They neither have to know the topology of the whole network nor the direction of the sink.

4.3.2 Initialization at Growth Codes

Each of the n nodes generates its own symbol and stores this symbol permanently in a separate storage space in the node. For initialization, a node's own symbol is additionally stored in each storage space of the node's storage which is used for data exchange. These copies of the node's own symbol are considered as codewords which have the degree $d = 1$ as they only contain one symbol. According to [Kam+06b; Kam+06a], ten storage spaces in each node are sufficient for data exchange in a network of 500 nodes, for example. An example for the nodes' storage content at the beginning of the GC procedure can be seen in Figure 4.3.

Algorithm 1 Coordination of sending and receiving

```
 1: procedure COORDINATESENDRECEIVE
 2:     while 1 do
 3:         while notMySlot do
 4:             if receiveRequest then                    // if request for data exchange is received
 5:                 node ← getRequestingNode()            // request for data exchange received from node
 6:                 isInitiator ← false                   // node is initiator of data exchange
 7:                 reencodeAndExchange(node)             // data exchange, see Algorithm 3
 8:                 receiveRequest ← false
 9:             end if
10:         end while
11:         isInitiator ← true                            // considered node itself is initiator of data exchange
12:         node ← chooseRandomNeighbor()
13:         initiateBidirectionalDataExchange(node)
14:         reencodeAndExchange(node)                     // data exchange, see Algorithm 3
15:     end while
16: end procedure
```

Algorithm 2 Reception and storage of a packet/codeword (CW)

```
1: procedure RECEIVEANDSTORE(no)
2:     packet ← getPacketFromNeighbor()
3:     cw ← packet.getCw()                                // extract contained CW
4:     storeCw(cw, no)          // store CW in storage space number no, see Algorithm 3 for no
5: end procedure
```

4.3.3 Encoding and Data Exchange with Growth Codes

In [Kam+06b; Kam+06a], time is divided into rounds. Each round, a node initiates one bidirectional data exchange with a randomly selected neighbor. For this purpose, the node chooses and recodes a codeword of its storage, selects a random neighbor, and exchanges the chosen codeword with the codeword which is offered by the neighbor. The coordination of sending and receiving is described more formally in Algorithm 1.

For the basic GC principle, there is only one sink in the network, which is directly connected to a random regular node. The sink receives each data packet which was received by the corresponding node in the current round, and tries to decode the contained codeword. In the case of GC, the sink is the only node which decodes codewords and aims to reconstruct as many original symbols as possible at any time. Regular nodes only recode a codeword directly before exchanging it, and store received codewords without transforming them. Algorithm 2 gives an abstract description of the reception and storage of a codeword at a regular node.

In terms of being able to decode optimally, a perfect source setting is assumed in [Kam+06b; Kam+06a]. This means, each symbol has the same probability of being included in a codeword for transmission. On top of that, it is assumed that the degree of each codeword which reaches the sink fulfills the determined optimal degree. In [Kam+06b; Kam+06a], it was shown that the decodability at the sink in a perfect source setting is optimal if the degree of a codeword which is transmitted to the sink is determined according to the so-called GC degree distribution.

In a distribution, each possible value of a random variable is assigned a certain probability. Thus, a distribution is defined by the system of assigned probabilities. This proposition and more details can be found, for example, in [Hen08].

In the case of the GC degree distribution, the value of the considered random variable is the degree of a codeword. So each degree d should be assigned a probability. The GC degree distribution in [Kam+06b; Kam+06a] is based on maximizing the probability $p_{Dist1}(r,d)$. Here, $p_{Dist1}(r,d)$ denotes the probability that an incoming codeword of degree d contains exactly one symbol which has not been reconstructed yet if r symbols have already been reconstructed. This means, the probability that a received codeword can immediately be decoded is maximized.

In a second distribution, a hypergeometric distribution, see for example [Hen08], the value of the random variable Y is the number of symbols in a codeword which have not been reconstructed yet. Each symbol of an incoming codeword can either be already reconstructed or not reconstructed yet by the sink. In a distance-1-codeword, exactly one symbol has not been reconstructed yet by definition. Consequently, the relevant value of the random variable Y for calculating the probability $p_{Dist1}(r,d)$ is $Y = 1$ as one symbol has not been reconstructed. A symbol can be considered to be drawn from one of two distinct sets, while unordered sampling without replacement is used. The probability $p_{Dist1}(r,d)$ is, hence, calculated according to the probability which is associated to $Y = 1$ in the case of the hypergeometric distribution. Therefore, the following formula is used, where n is the total number of different symbols existing in the network, and r is the number of already reconstructed symbols:

$$p_{Dist1}(r,d) = \frac{\binom{r}{d-1}\binom{n-r}{1}}{\binom{n}{d}} \tag{4.14}$$

In [Kam+06b; Kam+06a], the probability $p_{Dist1}(r,d)$ is also calculated according to Equation (4.14). The degree d which maximizes $p_{Dist1}(r,d)$ for a respective number r of reconstructed symbols in Equation (4.14) is optimal for an incoming codeword at the sink. Thus, all codewords which will be transmitted by any node after the sink has reconstructed r symbols should have the degree d until the degree $d + 1$ is determined to be optimal for decoding. The optimal degree for decoding is used as the current proposed degree of the system. In [Kam+06a], it is shown that the optimal degree is monotonically increasing with the number of reconstructed symbols. At the beginning, degree 1 is optimal. Later, when the sink has reconstructed more than $\frac{n-1}{2}$ original

Algorithm 3 Recoding and exchange of a packet/codeword (CW)

```
 1: procedure REENCODEANDEXCHANGE(node)
 2:     os ← getOwnSymbol()                        // respective node's own symbol
 3:     d ← getCurrentProposedDegree()             // degree according to hard-coded transition points
 4:     cw, no ← chooseRandomCw()                  // choose random CW from storage, and get its storage space no.
 5:     if getDegree(cw) < d then                  // if proposed degree not fulfilled yet
 6:         if !cw.contains(os) then               // if own symbol not included yet
 7:             cw ← cw ⊕ os                       // add own symbol
 8:         end if
 9:     end if
10:     packet ← createPacket(cw)                  // add header and create packet for transmission
11:     if isInitiator then                        // if considered node initiated data exchange, see Algorithm 1 for isInitiator
12:         transmitPacket(packet, node)           // transmit packet to chosen neighbor
13:     end if
14:     receiveAndStore(no)                        // receive packet from neighbor, store in storage space, see Algorithm 2
15:     if !isInitiator then                       // if considered node did not initiate data exchange, see Algorithm 1 for isInitiator
16:         transmitPacket(packet, node)           // transmit packet to chosen neighbor
17:     end if
18: end procedure
```

symbols, degree 2 is optimal, and so on. Therefore, the considered codes are called 'Growth Codes'. Further details on the degree distribution and relevant approximations follow in Section 4.3.6.

The GC degree distribution in [Kam+06b; Kam+06a] is realized by appropriately selected degree transition points, which are hard-coded into all nodes in advance. Transition points represent points in time when the proposed degree for all codewords which will be transmitted subsequently is incremented by 1. If the degree is increased too early, the sink will probably not be able to decode the respective codewords and not reconstruct the contained symbols immediately. This means, the aim of GC, to be able to reconstruct as many original symbols as possible at any time, is not achieved. If the degree is increased too late, the sink will receive more completely redundant codewords, which represents unnecessary transmissions.

When generating a codeword for transmission, a node chooses a random codeword of its storage for data exchange and checks its degree. If this degree is lower than the system's proposed degree of that time and its own symbol is not included, it recodes the codeword by combining the codeword with its symbol via XOR. The codeword remains unchanged if it already contains the node's own symbol or if its degree already fits the proposed degree. This process of recoding, and the exchange of a codeword is additionally described in Algorithm 3.

An example for a first bidirectional data exchange is depicted in Figure 4.4. Here, the proposed degree is $d = 1$.

Figure 4.5 shows an example for a data exchange when the sink has already recon-

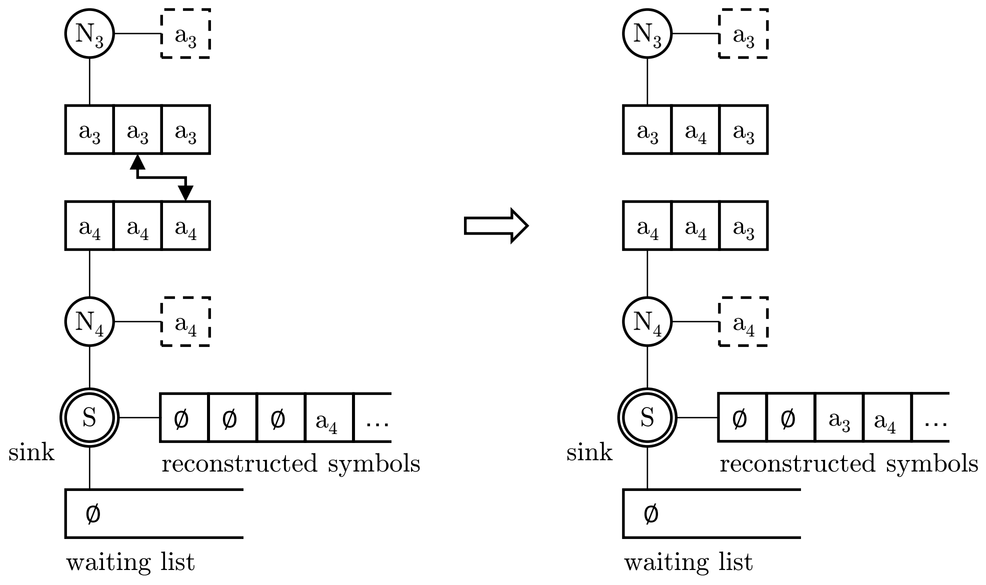


Fig. 4.4: Example of first data exchange in the system (before ⇒ after), $d = 1$

structed some symbols and the proposed degree is $d = 2$.

4.3.4 Data Packets in Growth Codes

The payload of a data packet in the case of GC solely consists of the codeword which was chosen for transmission. If the size of an original symbol is b bits, a codeword also has b bits and, thus, the payload of a GC packet consists of $\lceil b/8 \rceil$ Bytes. The associated header indicates which symbols have been combined via XOR to form the respective codeword. This information is necessary for decoding. In [Kam+06b], it is assumed that each node possesses a unique ID and transfers this ID to its own symbol. There are n different symbols in a network of n nodes, so the authors of [Kam+06b] assume that each ID has $\log_2 n$ bits. For realistic implementations of other network coding procedures, a 16-bit short address is usually used as ID, which is also described in Section 3.2.2. For this reason, an ID should have 16 bits at GC in the case of a comparison with related work. In this section, however, GC are described in accordance with the authors in [Kam+06b]. In [Kam+06b], two different methods are applied to specify the symbols which are contained in a codeword: a log format and a bit format.

In the case of log format, the ID of each symbol which is included in the codeword is added to the header. In the case of bit format, there are n bits to indicate the symbols which were combined for the codeword. Each bit is assigned to a certain symbol. If a symbol is included in the codeword, the respective bit in the header is set to 1, otherwise

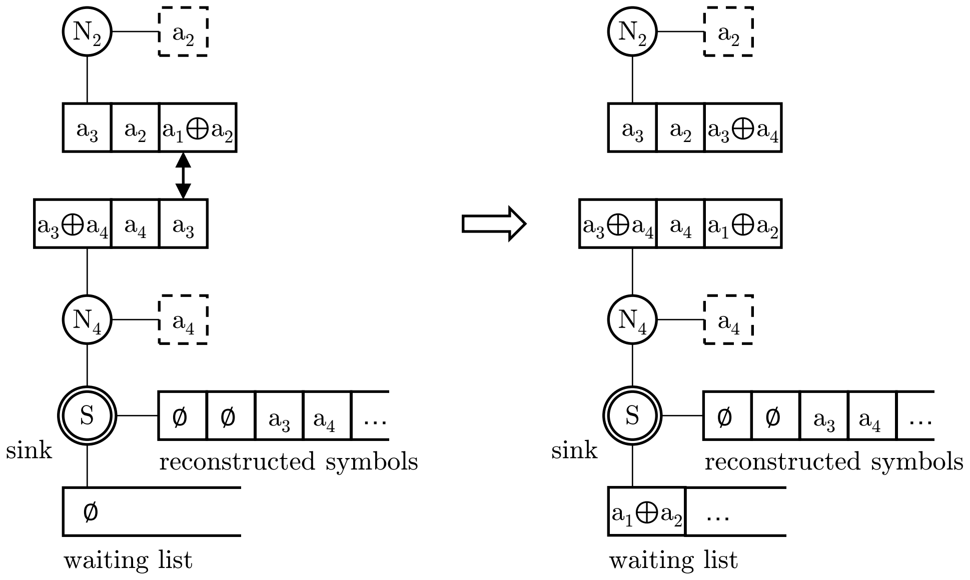


Fig. 4.5: Example of data exchange later on (before ⇒ after), $d = 2$

it is set to 0. For this format, n bits are necessary in a network of n nodes. Depending on the number of symbols which are combined in the codeword, i.e. the degree d of the codeword, either log format or bit format provides a shorter header and is applied consequently. While $d \cdot \log_2 n < n$, log format is shorter, and if $d \cdot \log_2 n \geq n$, bit format is used. In [Kam+06b], the first bit of the first Byte in the header indicates whether log format or bit format is applied. In log format, the remaining seven bits of this Byte contain the degree d of the codeword, which is thus limited by 128. This indication is necessary to be able to identify the position of the following IDs. In bit format, the remaining seven bits are ignored. All in all, a GC frame contains:

- one bit to indicate the used frame format (1 bit),
- the degree of the contained codeword/left blank (7 bits),
- the IDs of the contained symbols/bits to indicate the contained symbols ($min\{\lceil (d \cdot \log_2 n)/8 \rceil, \lceil n/8 \rceil\}$ Bytes or $min\{(d \cdot 16)/8, \lceil n/8 \rceil\}$ Bytes), and
- one codeword (payload) ($\lceil b/8 \rceil$ Bytes).

Thus, there are $1 + min\{\lceil (d \cdot \log_2 n)/8 \rceil, \lceil n/8 \rceil\}$ Bytes or $1 + min\{d \cdot 2, \lceil n/8 \rceil\}$ Bytes total overhead in the case of GC, and the relative overhead is:
$(1 + min\{\lceil (d \cdot \log_2 n)/8 \rceil, \lceil n/8 \rceil\})/\lceil b/8 \rceil$ or $(1 + min\{d \cdot 2, \lceil n/8 \rceil\})/\lceil b/8 \rceil$.

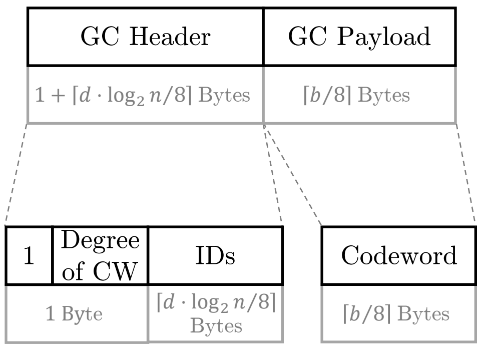

GC Header
GC Payload
$1 + \lceil d \cdot \log_2 n/8 \rceil$ Bytes
$\lceil b/8 \rceil$ Bytes
1
Degree of CW
IDs
Codeword
1 Byte
$\lceil d \cdot \log_2 n/8 \rceil$ Bytes
$\lceil b/8 \rceil$ Bytes

Fig. 4.6: GC frame structure in log format

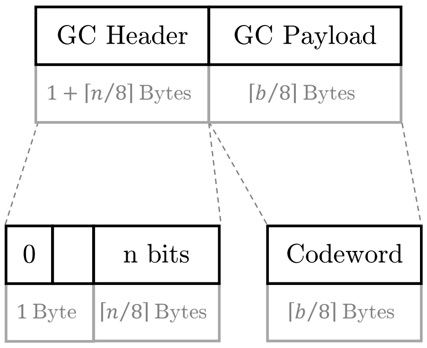

GC Header
GC Payload
$1 + \lceil n/8 \rceil$ Bytes
$\lceil b/8 \rceil$ Bytes
0
n bits
Codeword
1 Byte
$\lceil n/8 \rceil$ Bytes
$\lceil b/8 \rceil$ Bytes

Fig. 4.7: GC frame structure in bit format

To sum up this section, the following Figures 4.6 and 4.7 show the two options of the general structure of a GC frame, which represents the MAC Data Payload of Figure 3.2. Please note that the proportions in the figures do not reflect the actual size ratios. The actual ratio depends on the number n of original symbols, the degree d of a codeword, and the size of the codeword, i.e. the size of a single symbol.

4.3.5 Decoding at Growth Codes

Decoding starts immediately after having received a packet, i.e. a codeword in the considered sink. For decoding, all symbols in the codeword which have already been reconstructed are removed from the received codeword via XOR. This is done by combining the codeword via XOR with the already known symbols which are included in the codeword as XOR is self-inverse. If the remaining codeword only contains exactly one symbol, the codeword has been decoded, and the remaining symbol is stored as

Algorithm 4 Reception and decoding of a packet/codeword (CW) in the sink

```
procedure SINKRECEIVEANDDECODE(packet)
    cw ← packet.getCw()                                   // extract contained CW
    listIDs ← packet.getContainedSymbols()                // extract IDs
    for all ID ∈ listIDs do
        if storageS.contains(ID) then                     // if symbol already reconstructed
            cw ← (cw ⊕ storageS.getSymbol(ID))            // remove symbol from CW
            packet.setCw(cw)                              // update CW in packet
            packet.updateHeader()                         // update IDs of contained symbols
        else                                              // if symbol not yet reconstructed
            numInnovativeSymbols++
            innovSymbolID ← ID
        end if
    end for
    if numInnovativeSymbols > 1 then                      // not possible to decode CW yet
        waitingList.add(packet)                           // store updated CW in waiting list
    end if
    if numInnovativeSymbols = 1 then                      // CW decoded
        storageS.add(cw)                                  // contained symbol reconstructed
        checkWaitingList(innovSymbolID)                   // remove reconstructed symbol from CWs in waiting list
    end if
    if allSymbolsReconstructed then
        setNodeFinished()
    end if
end procedure
```

being reconstructed. If the remaining codeword contains more than one symbol, it is added to the sink's so-called waiting list. The algorithm terminates when the sink has reconstructed every original symbol of the network. A more formal description of the reception and decoding of a codeword in the sink follows in Algorithm 4.

4.3.6 Stochastic Correlations and Details on the Degree Distribution

In the following, the optimal degree of an incoming codeword, which depends on the number of already reconstructed symbols, is discussed first. Then, the expected number of codewords which have to be received to reconstruct a certain number or all original symbols is determined. Finally, it is explained in detail how the GC degree distribution is implemented.

4.3.6.1 Optimal Degree of a Codeword

The aim of GC is to reconstruct as many original symbols as possible in the sink with each number of received packets/codewords. It is, thus, optimal if the sink can immediately decode an incoming codeword. This means, the probability $p_{Dist1}(r,d)$ to receive a distance-1-codeword at the sink should be maximized, as already described in Section 4.3.3. To this end, if the sink has reconstructed r symbols, the proposed degree of codewords for transmission should be increased from d to $d+1$ while

$$r \geq \left\lfloor \frac{dn-1}{d+1} \right\rfloor + 1 =: R_d + 1, \text{ for } d \in [1;n-1], \tag{4.15}$$

which will be shown in the following. De facto, the degree should be increased as long as Equation (4.15) is fulfilled and r has already reached or exceeded the respective transition point R_d+1. This means, for a considered number of already reconstructed symbols r, the proposed degree should be increased as long as a higher degree gives a higher probability to receive a distance-1-codeword. The fundamental approach of R_d is based on [Kam+06a]. For the derivation of R_d, it is assumed that all original symbols have the same probability to be included in a codeword, so Equation (4.14) can be used. The following calculation then determines R_d:

$$\begin{aligned}
& p_{Dist1}(r,d+1) > p_{Dist1}(r,d), && \text{for } d \in [1;n-1] \\
\Leftrightarrow\ & \frac{\binom{r}{d}\binom{n-r}{1}}{\binom{n}{d+1}} > \frac{\binom{r}{d-1}\binom{n-r}{1}}{\binom{n}{d}} \\
\Leftrightarrow\ & \frac{d+1}{d} > \frac{n-d}{r-d+1} \\
\Leftrightarrow\ & r > \frac{dn-d^2+(d-1)(d+1)}{d+1} \\
\Leftrightarrow\ & r > \frac{dn-1}{d+1}.
\end{aligned}$$

If there are $r=\frac{dn-1}{d+1}$ reconstructed symbols in the sink, the probabilities $p_{Dist1}(r,d+1)$ and $p_{Dist1}(r,d)$ are identical. In this case, degree d should be used for codewords as packets of degree d can be shorter than those of degree $d+1$. Thus, until and including $r=\left\lfloor \frac{dn-1}{d+1} \right\rfloor =: R_d$ symbols have been reconstructed by the sink, the current proposed degree of codewords for transmission should be d. Here, the floor function can be included as $r \in \mathbb{N}$. As soon as r reaches $r=R_d+1$, the proposed degree should be increased to at least $d+1$ as the probability $p_{Dist1}(r,d+1)$ to immediately decode an incoming codeword is then higher than $p_{Dist1}(r,d)$.

In contrast to the conclusion in [Kam+06b; Kam+06a], Equation (4.15) implies that codewords of degree $\leq d$ have to be received to reconstruct the first R_d+1 original symbols, not only R_d symbols. The reason for this is that the transition from d to $d+1$ takes place directly after $r=R_d+1$ symbols were reconstructed. This issue will change, among others, the lower and the upper bound of the summation when calculating the number of codewords which are necessary to reconstruct a certain number of symbols. Therefore, Equation (4.18) and Equation (4.19) were adapted respectively.

According to Equation (4.15), degree $d = 1$ should be used for codewords for transmission as long as $r \leq \left\lfloor \frac{n-1}{2} \right\rfloor := R_1$. After $r = R_1 + 1$ symbols were reconstructed, the proposed degree should be increased and degree $d = 2$ should be used while $r \leq \left\lfloor \frac{2n-1}{3} \right\rfloor := R_2$, and so on.

As degree $d = 1$, for example, should be used until $r = \left\lfloor \frac{n-1}{2} \right\rfloor$ and $r \leq n$, not every possible degree between 1 and n can exist due to the pigeonhole principle. A degree $d+1$ is omitted if and only if $R_d = \left\lfloor \frac{dn-1}{d+1} \right\rfloor = \left\lfloor \frac{(d+1)n-1}{(d+1)+1} \right\rfloor = R_{d+1}$ for $d \in [1; n-2]$. This can be proved by using the following aspects: If $r > R_d$ then $p_{Dist1}(r, d+1) > p_{Dist1}(r, d)$, and if $r > R_{d+1}$ then $p_{Dist1}(r, d+2) > p_{Dist1}(r, d+1)$. Thus, if $r > R_d = R_{d+1}$ then $p_{Dist1}(r, d+2) > p_{Dist1}(r, d+1) > p_{Dist1}(r, d)$, so it is directly transitioned from degree d to $d+2$, and degree $d+1$ is omitted. Even if this instance is not considered in [Kam+06b; Kam+06a], the derived Equation (4.18) for the number K_{d+1} of codewords of degree $\leq (d+1)$ which have to be received still applies. In the case of an omitted degree $d+1$, the upper bound of the summation for K_{d+1} is lower than the lower bound as $R_{d+1} < R_d + 1$, so $K_{d+1} = K_d + 0$ in Equation (4.19). This means zero codewords of degree $d+1$ are added to the codewords of degree $\leq d$. Thus, the formula remains valid even if certain degrees are omitted.

4.3.6.2 Expected Value of the Required Number of Packets/Codewords

For degree-1-codewords, basically the same propositions apply as for NoC in Section 4.1.2 since degree-1-codewords are non-encoded symbols. Thus, formulas for the Coupon Collector's problem could be used to determine how many degree-1-codewords have to arrive at the sink in order to reconstruct a certain number of symbols. The Coupon Collector's scenario can be transferred to GC for the case of $c = 1$ as only at most one original symbol can be extracted from a codeword in the case of GC. In a codeword of degree d, $d - 1$ of the included symbols have to be already reconstructed by the sink to decode the codeword and reconstruct the respective symbol. However, the formulas of the Coupon Collector's problem can only be applied to determine how many degree-1-codewords have to be received to reconstruct a certain number of symbols. Degree-1-codewords are proposed until $R_1 + 1 = \left\lfloor \frac{n-1}{2} \right\rfloor + 1$ symbols have been reconstructed by the sink, cf. Equation (4.15). For codewords of degree ≥ 2, the situation is more complex and cannot be reduced to the Coupon Collector's problem. Therefore, a more general approach will be used here. However, the Coupon Collector's formulas, which only apply to degree 1 and are presented in Section 4.1.2, can also be derived from the more general approach.

As already defined before, an incoming codeword of degree d is a distance-m-

codeword for a considered sink, if it contains exactly m symbols which have not been reconstructed by the sink yet. A degree-d-codeword with $d \geq 2$ can be a distance-0-codeword, a distance-1-codeword, or a codeword of distance > 1. Receiving a distance-0-codeword is the worst case as this codeword is redundant and has to be rejected. A codeword of distance > 1 could be decoded later, but could also turn into a distance-0-codeword instead. For the sake of simplicity, it is assumed that there is no waiting list, so codewords with a distance > 1 will be rejected in the following as they do not have an immediate benefit, and they still could become redundant. A distance-1-codeword is optimal as it can immediately be decoded. By receiving such a codeword, the sink can reconstruct one further symbol. It has to be determined how many packets/codewords have to be received to reconstruct a certain number of symbols.

For the following formulas, incoming codewords which have a distance > 1 are considered to be rejected. Hence, a codeword provides one further reconstructed symbol if and only if the received codeword is a distance-1-codeword. This assumption will be relaxed after the first fundamental formulas. Additionally, it is assumed that solely degree-d-codewords arrive at the sink, and that all n existing distinct symbols have the same probability to be contained in a codeword. Under these circumstances, the probability that an incoming codeword of degree d can immediately be decoded is

$$p_r := p_{Dist1}(r,d) = \frac{\binom{r}{d-1}\binom{n-r}{1}}{\binom{n}{d}},$$

with r reconstructed symbols, see Equation (4.14). Thus, $p_{Dist1}(r,d)$ is the probability to reconstruct one further symbol when receiving a codeword of degree d. In the case of an incoming degree-1-codeword and r already reconstructed symbols, the probability to reconstruct one further symbol is $p_{Dist1}(r,1) = \frac{n-r}{n}$. Here, one further symbol can be reconstructed if and only if one of the $n-r$ missing symbols arrives at the sink in this degree-1-codeword.

A codeword can either be a distance-1-codeword with probability $p_{Dist1}(r,d)$ or no distance-1-codeword with probability $(1-p_{Dist1}(r,d))$. Thus, receiving a codeword can be interpreted as Bernoulli-Experiment, which is defined, for example, in [Hen08]. A codeword can be decoded and one further symbol is reconstructed if and only if the received codeword is a distance-1-codeword. Assume r symbols have already been reconstructed. Let X_r then be a discrete random variable denoting the number of codewords which have to be received until one further symbol can be reconstructed, i.e. until the next distance-1-codeword is received. This means that Bernoulli-Experiments are executed until a distance-1-codeword is finally received. Hence, each of the X_r, with $r \in [0; n-1]$, is geometrically distributed with parameter $p = p_{Dist1}(r,d)$. Details on the geometric distribution as well as on associated probabilities and the expected value can be found, for example, in [Hen08]. For the geometrically distributed random variable X_r, each possible number of received codewords m is assigned the probability

$$p(m) = P(X_r = m) = (1 - p_{Dist1}(r,d))^{m-1} \cdot p_{Dist1}(r,d).$$

This formula contains that the first $m-1$ codewords are no distance-1-codewords and the codeword number m is a distance-1-codeword. According to [Hen08], the expected value for X_r is

$$E[X_r] = \frac{1}{p_{Dist1}(r,d)} = \frac{\binom{n}{d}}{\binom{r}{d-1}\binom{n-r}{1}} \tag{4.16}$$

in the given context. This means, an expected number of $E[X_r]$ packets/codewords has to be received to reconstruct one further symbol if r symbols have already been reconstructed by the sink. Concerning the general statement, Equation (4.16) also appears in a proof in [Kam+06a], but with no mathematical derivation beforehand. In [Kam+06b; Kam+06a], it is assumed that the sink has not reconstructed any symbol at the beginning even though the node which is directly connected to the sink has reconstructed its own symbol at initialization. Therefore, the following formulas start at $r = 0$. To calculate how many packets/codewords $E[\#packets]$ have to be received in total on average in order to reconstruct all n original symbols, the expected value $E[X_r]$ for every possible r has to be summed up, i.e.

$$E[\#packets] = \sum_{r=0}^{n-1} E[X_r]. \tag{4.17}$$

Depending on r, a certain degree d is proposed according to Equation (4.15) and, thus, used to determine the corresponding $E[X_r]$. For $r = 0, ..., R_1$, with $R_1 = \left\lfloor \frac{n-1}{2} \right\rfloor$, the respective expected value $E[X_r]$ refers to degree-1-codewords, see Equation (4.15). Using degree $d = 1$ in Equation (4.16), gives the expected value

$$E[X_r]_{(d=1)} = \frac{1}{p_{Dist1}(r,1)} = \frac{\binom{n}{1}}{\binom{r}{0}\binom{n-r}{1}} = \frac{n}{n-r}.$$

Thus, an expected number of $E[X_0] = \frac{n}{n} = 1$ degree-1-codeword has to be received by the sink to reconstruct the first symbol. In order to reconstruct the last symbol of the $R_1 + 1 = \left\lfloor \frac{n-1}{2} \right\rfloor + 1$ symbols which are associated with degree-1-codewords, an expected number of $E[X_{R_1}] = \frac{n}{n-R_1}$, which is less than 2, degree-1-codewords is necessary. In total, an expected number of

$$K_1 := \sum_{r=0}^{R_1} E[X_r]_{(d=1)} = \sum_{r=0}^{R_1} \frac{n}{n-r} \tag{4.18}$$

degree-1-codewords has to arrive at the sink in order to reconstruct the first $R_1 + 1$ symbols. An analog to Equation (4.18) can also be found in [MR95] for the Coupon Collector's problem. The authors of GC state Equation (4.18) in [Kam+06b; Kam+06a] with a different upper bound of the summation, as already explained at the beginning of this section. When $R_1 + 1$ symbols have been reconstructed by the sink, which means symbol number $R_1 + 2$ will be reconstructed next, the degree of incoming codewords should increase from 1 to 2, see Equation (4.15). The degree should remain 2 until $R_2 + 1 = \left\lfloor \frac{2n-1}{3} \right\rfloor + 1$ symbols have been reconstructed. As in [Kam+06a], for degree $d \geq 2$, the term 'expected number' will be replaced by 'expected maximum number' in the following because codewords with a distance > 1 are not rejected when applying the actual algorithm. Codewords which cannot be decoded immediately are stored in a waiting list. Later on, each of these codewords can provide one further reconstructed symbol. This means, less codewords have to be received to reconstruct a certain number of original symbols.

In general, an expected maximum number of

$$K_d := K_{d-1} + \sum_{r=R_{d-1}+1}^{R_d} \frac{\binom{n}{d}}{\binom{r}{d-1}\binom{n-r}{1}} \tag{4.19}$$

packets/codewords has to be received to reconstruct $R_d + 1 \leq n$ original symbols, with $d \in [2;n]$, $R_d := \left\lfloor \frac{dn-1}{d+1} \right\rfloor$ of Equation (4.15), and $K_1 := \sum_{r=0}^{R_1} \frac{n}{n-r}$ of Equation (4.18). This means, for each possible number of reconstructed symbols between $R_{d-1} + 1$ and R_d, the expected maximum number of degree-d-codewords which has to be received to reconstruct another symbol is added to K_{d-1}. An expected maximum number of K_{d-1} of the K_d codewords has a degree $\leq (d-1)$, and an expected maximum number of $K_d - K_{d-1}$ codewords are degree-d-codewords. Equation (4.19) and a corresponding proof can also be found in [Kam+06b; Kam+06a] for GC, but with a different lower and upper bound of the summation, as already mentioned at the beginning of this section. Equation (4.19) can also be applied for degree-n-codewords, i.e. $d = n$. Here, a symbolic $R_n := n - 1$ has to be introduced, which is not explicitly addressed in [Kam+06b; Kam+06a]. By definition, R_n would be the number of reconstructed symbols after which degree n is transitioned to degree $n + 1$, which does not exist in a system of n original symbols. In general, degree-n-codewords should be received when at least $R_{n-1} + 1 = n - 1$ symbols have been reconstructed, and until $R_n = n - 1$ symbols have been reconstructed. This means, when $R_n = n - 1$ symbols have been reconstructed, a degree-n-codeword is received and, thereby, n symbols will be reconstructed. Subsequently the degree would increase to $n + 1$ if the algorithm did not terminate after n symbols have been reconstructed. The lower and the upper bound of Equation (4.19) is $n - 1$ when calculating K_n, so only one addend is used. Furthermore, $\frac{\binom{n}{d}}{\binom{r}{d-1}\binom{n-r}{1}} \overset{r=n-1,d=n}{=}$ $\frac{\binom{n}{n}}{\binom{n-1}{n-1}\binom{n-(n-1)}{1}} = 1$. To conclude, an expected maximum number of

$$E[\#packets] = K_n = K_{n-1} + 1 \tag{4.20}$$

packets/codewords has to be received, including K_{n-1} codewords of degree $\leq n-1$ and 1 degree-n-codeword, in order to reconstruct all n original symbols. Here, K_n and K_{n-1} are determined according to Equation (4.19).

4.3.6.3 Degree Distribution for Growth Codes

The GC degree distribution is based on maximizing the probability $p_{Dist1}(r,d)$ that a degree-d-codeword arriving at the sink is a distance-1-codeword for the sink. This means that the codeword contains exactly one symbol which has not been reconstructed yet and, thus, can immediately be decoded. For the GC degree distribution, the value of the considered random variable X is the degree of a codeword. Hence, each possible degree d is assigned a certain probability $P(X = d)$ and, when generating a codeword, the degree should be chosen according to these probabilities.

One option to maximize $p_{Dist1}(r,d)$ is to use Equation (4.19) as a base due to its construction. Depending on the number of already reconstructed symbols, the degree with the highest probability $p_{Dist1}(r,d)$ is proposed. For each degree, the expected maximum number of codewords which have to be received is subsequently calculated by Equation (4.19). Consequently, the relative frequency of degree-d-codewords in Equation (4.19) can be applied to determine which probability should be assigned to degree d. In order to create a k-th codeword, the authors of [Kam+06b; Kam+06a] suggest:

$$\bar{\pi}(k): \qquad P(X = d) = max\left(0, min\left(\frac{K_d - K_{d-1}}{k}, \frac{k - K_{d-1}}{k}\right)\right), \tag{4.21}$$

where

$$K_d = K_{d-1} + \sum_{r=R_{d-1}+1}^{R_d} \frac{\binom{n}{d}}{\binom{r}{d-1}\binom{n-r}{1}},$$
$$K_1 = \sum_{r=0}^{R_1} \frac{n}{n-r}.$$

Equation (4.21) means that the probability $P(X = d)$ to choose a degree d for the k-th codeword, which will be created next, is evaluated for every possible degree $d \in [1; n]$. Here, Equation (4.19) and Equation (4.15) are used to calculate K_d and R_d, respectively. The max function in Equation (4.21) gives 0 if $K_{d-1} > k$, which means that less codewords k were and are created than the expected maximum number of degree $d-1$. Thus, no codewords of degree $\geq d$ should be used, so $P(X = d) = 0$. If $K_{d-1} < k$, the second term applies, which contains the calculation of a min function and will be described in the following. The min function in Equation (4.21) uses the second term if $k < K_d$. In this case, less codewords k were and are created than the expected maximum number K_d of codewords of degree $\leq d$ which has to be received by the sink in general. This means, an expected maximum number of only $k - K_{d-1}$ degree-d-codewords should be used. If $k > K_d$, an expected maximum number of $K_d - K_{d-1}$ degree-d-codewords should be created, so the min function uses the first term, and also codewords of degree $> d$ will be considered later. In order to get the probability $P(X = d)$ for a degree-d-codeword, the relative frequency of degree-d-codewords compared to the number k of created codewords is used by dividing the calculated number by k in Equation (4.21).

According to [Kam+06b; Kam+06a], it is optimal for decoding at the sink if the degree of incoming codewords is monotonically increasing. Therefore, the degree of codewords for transmission should not be determined according to a probability, but has to be deterministic, start with degree 1 and increase step by step. Even though the authors of [Kam+06b; Kam+06a] declare to apply the degree distribution according to Equation (4.21), they do not use it in the strict sense. Instead, they utilize appropriately selected degree transition points, which are points in time when the proposed degree for all codewords which will be created next is increased by 1. These degree transition points are hard-coded into all nodes in advance, and are based on Equation (4.19). The reason why degree transition points which are chosen according to Equation (4.19) can be determined in advance will be explained in the following. For Growth Codes, it is assumed that each node performs one bidirectional data exchange with a randomly chosen neighbor per round, i.e. it receives and transmits one packet/codeword per round. As a node counts the number of elapsed rounds, it can estimate how many packets/codewords the sink has received in the case of a round-based, collision-free approach. In reality, more codewords could have been transmitted by a node/received by the sink as a node can be chosen for data exchange more than once, so less rounds might be needed until the sink has reconstructed a respective number of symbols. After K_d rounds, with $d \in [1; n-1]$, i.e. after at least K_d codewords have been transmitted/received, the degree should increase from d to $d+1$. For each possible number of reconstructed symbols r, the optimal degree d can be calculated by using Equation (4.15). Then, $p_{Dist_1}(r, d)$ can be determined for each r via Equation (4.14). With this information, the expected maximum number of codewords which has to be received can be determined for each r by using Equation (4.16). Finally, K_d sums up the expected maximum number of codewords for each number of reconstructed symbols r at which a degree $\leq d$ is proposed. As K_d, with $d \in [1; n-1]$, uses only variables which are known or can be calculated, each K_d can be determined in advance. Therefore, the set of $\{K_d | d \in [1; n-1]\}$ can be applied as degree transition points which are determined in advance, and each K_d represents the number of rounds when the proposed degree is increased from d to $d+1$.

Assume that a node only transmits exactly one codeword per round and a sink only receives exactly one codeword per round accordingly. If the degree transition points are chosen according to K_d in Equation (4.19), the node will transmit K_1 degree-1-codewords, $K_2 - K_1$ degree-2-codewords, $K_3 - K_2$ degree-3-codewords, and so on until $K_n - K_{n-1}$ degree-n-codewords. This means, K_n codewords are transmitted in total. Assuming a round-based, collision-free approach, the sink then receives this number of codewords from the respective degree. In the following, the approach based on degree transition points will be compared to the option to choose the degree of each codeword which will be transmitted according to the probabilities of the GC degree distribution, see Equation (4.21). In order to make the comparison as fair as possible, it is assumed that also exactly $k = K_n$ codewords will be transmitted when using the GC degree distribution. For both approaches, it is possible that the sink is already able to reconstructed all original symbols before the considered node has transmitted all of the K_n symbols. However, this possibility will be neglected for the sake of simplicity both for the approach based on degree transition points and for the approach which uses the GC degree distribution. Under these assumptions, there are the following probabilities for the respective degrees at the GC degree distribution:

$$
\begin{aligned}
P(X=1) &= \frac{K_1}{K_n}, \\
P(X=2) &= \frac{K_2 - K_1}{K_n}, \\
P(X=3) &= \frac{K_3 - K_2}{K_n}, \ldots, \\
P(X=n) &= \frac{K_n - K_{n-1}}{K_n}.
\end{aligned}
$$

If K_n codewords are created independently of each other and according to these probabilities, the expected value for the number of codewords of a degree d is calculated as follows. Each created codeword can be a degree-d-codeword or no degree-d-codeword, and the order of the created codewords is neglected. Thus, the expected value $E[\#(\textit{deg-d-CWs})]$ for the number of created degree-d-codewords is determined according to the expected value in a binomial distribution with parameters $n = K_n$ and $p = p(d)$. Hence, $E[\#(\textit{deg-d-CWs})] = n \cdot p = K_n \cdot p(d)$, see for example [Hen08]. This gives the following expected values:

$$\begin{aligned}
E[\#(\textit{deg-1-CWs})] &= K_n \cdot \frac{K_1}{K_n} = K_1, \\
E[\#(\textit{deg-2-CWs})] &= K_n \cdot \frac{K_2 - K_1}{K_n} = K_2 - K_1, \\
E[\#(\textit{deg-3-CWs})] &= K_n \cdot \frac{K_3 - K_2}{K_n} = K_3 - K_2, \ldots, \\
E[\#(\textit{deg-n-CWs})] &= K_n \cdot \frac{K_n - K_{n-1}}{K_n} = K_n - K_{n-1}.
\end{aligned}$$

To conclude, K_1 degree-1-codewords, $K_2 - K_1$ degree-2-codewords, $K_3 - K_2$ degree-3-codewords, etc., and $K_n - K_{n-1}$ degree-n-codewords are created on average if K_n codewords are created in total. Thus, the same number of codewords of each degree is transmitted if the GC degree distribution of Equation (4.21) is applied as if the degree transition points according to K_d in Equation (4.19) are used. However, the order of the chosen degrees can be different for these approaches.

4.3.7 Benefits and Limitations of Growth Codes

One benefit of GC is that almost no knowledge about the network topology or the position of the sink is necessary. Only direct neighbors have to be known. Thus, the algorithm can start immediately without a complex initialization process, and adapts well to topology changes. Furthermore, GC can be implemented easily, even in a distributed setting.

The advantage of GC which will be described next refers to routing. In the case of GC, multiple replications of each symbol are spread across the network, and data exchanges with only random neighbors are sufficient so that all original symbols reach the sink eventually. It is not necessary to send data directly from each node to the sink. This means, no routing table has to be generated or maintained, and the area around the sink is not congested or highly stressed.

GC claim to lose no or only few symbols in the case of node failures. This is ensured by replicating each node's own symbol. These replications will be combined with other nodes' symbols in codewords and spread across the network. If there are C storage spaces for data exchange in each node, a node's own symbol will be C-times in this storage at the beginning, and will be transmitted later on. On top of that, a node's own symbol is added to each codeword which is sent by the node if it is not included and the proposed degree is not reached yet. This means a node's own symbol is available in many codewords which are stored in different nodes' storage for data exchange. By adding this amount of redundancy of each node's own symbol to the network, the probability to lose data in the case of node failures is reduced. Nevertheless, the network is not spammed with an infinite number of equal symbols as there are only C storage spaces for received codewords in each node.

Another benefit of GC is that reconstruction at the sink can already be started after having received only few codewords. It is not necessary to wait until a certain number of codewords has been received. By using the GC degree distribution, GC aim to reconstruct as many symbols as possible directly after reception.

Concerning drawbacks, GC use unicast-transmissions and do not take advantage of the multicast potential of radio communication. Additionally, they require bidirectional data exchange, which is difficult to realize in large-scale mobile WSNs.

Besides that, the degree transition points are not adapted to the sink's state but are chosen and hard-coded in advance as nodes do not know how many symbols r have already been reconstructed by the sink. Without knowledge of this number r, it is not possible to determine the optimal degree for the sink via Equation (4.15). Usually, the node density in the network, the degree of nodes' mobility, and the probability of link or node failures influence the optimal points in time for the degree transition points crucially but are not necessarily known in advance. If, as for GC, the number of elapsed rounds is used to estimate the number of codewords received by the sink, and this estimation is then used to indirectly estimate the number of reconstructed symbols in the sink in order to determine the optimal degree for this number of reconstructed symbols, then some accuracy is lost. This is a big disadvantage as the optimal choice of the transition points, i.e. of the degree, is important in order to be able to decode as much as possible directly after reception and reconstruct a new symbol and, thus, for the performance of the GC procedure. The optimal points in time to transition from d to $d+1$ should be identified. Equation (4.19), according to which the degree transition points in GC are determined, gives the expected maximum number of codewords which has to be received, not the expected number. If the GC degree distribution was used, the situation would be the same. Additionally, it is assumed that each node transmits and receives exactly one codeword per round at GC. This means, it is assumed that also the sink receives exactly one codeword per round. Therefore, the number of elapsed rounds is used to specify the degree transition points via Equation (4.19) in GC. In reality, however, more codewords could have been received per round in a round-based, collision-free system as a node can be chosen for data exchange more than once. Estimating the number of codewords received by the sink by the number of elapsed rounds is, thus, not necessarily accurate. After a considered number of rounds K_d, more than K_d codewords may have already been received by the sink. If the degree is increased only after K_d rounds, it may be increased later than Equation (4.19) suggests. Since GC use the estimated number of received codewords by the sink and use Equation (4.19) to estimate indirectly the number of reconstructed symbols in the sink for Equation (4.15) to specify the current proposed degree, the resulting proposed degree can be either too high or too low. To be able to calculate the current optimal degree for the sink via Equation (4.15), the actual number of reconstructed symbols r in the sink is necessary. Here, Equation (4.15) gives the optimal degree concerning the probability p_{Dist1} to reconstruct one further symbol but cannot be applied for GC, as already explained. If the degree is lower than the optimum, more distance-0-codewords arrive, which are redundant and, hence, represent the worst case. Therefore, more codewords are necessary to reconstruct the same number of symbols than if the degree was chosen optimally.

GC do not perform very well in static scenarios or scenarios with a low degree of nodes' mobility. In those cases, the GC degree distribution, which is realized by hard-coded degree transition points, can usually not be fulfilled. If the degree changes according to the degree transition points, the degree may already be lower than the optimum, as already explained in the paragraph before. If even those transition points are not fulfilled, the situation is even worse. Because of the little or not changing neighborhood, the same codewords arrive at a node again and again. The node's own symbol can only be added to each codeword once. After that, the degree of the codeword cannot be increased anymore by adding the node's own symbol. If a chosen codeword for transmission does not fulfill the proposed degree and the difference is even bigger than 1, a node can only increase the degree by at most one nevertheless. The degree of created codewords grows too slowly in scenarios with a low degree of nodes' mobility consequently. Thus, a highly dynamic system is a requirement for good performance of the GC algorithm.

Furthermore, for GC, it is assumed that all symbols have the same probability to be contained in a considered codeword, i.e. that the symbols for a codeword are uniformly drawn at random. In reality, this assumption usually does not hold. A node can only use codewords for transmission which were received before and are still stored in its storage. In addition, in the case of GC, a node does not compose a new codeword, it can only add its own symbol under certain conditions. Especially in a scenario with a low degree of nodes' mobility, there are symbols which are received within many codewords by a considered node and others which are received for the first time at a very late stage. This influences the performance of the GC algorithm as it takes a long time until the last symbols are reconstructed. This means more codewords have to be received to reconstruct a certain number of symbols than Equation (4.19) claims. If symbols cannot be considered to be uniformly drawn at random, the probability $p_{Dist1}(r, d)$ cannot be calculated according to Equation (4.14) as necessary prerequisites are not fulfilled. If the probability to receive a certain symbol in a considered node was known for each symbol, $p_{Dist1}(r, d)$ could be determined with a different, more complex formula. However, those probabilities cannot be known in advance. In reality, especially in a scenario with a low degree of nodes' mobility, a higher degree has to be used to maximize the actual $p_{Dist1}(r, d)$ than if symbols are uniformly drawn at random. In this case, more symbols should be included in a codeword to increase the probability that symbols which have not been received by the neighbor yet are used. Consequently, the degree d is increased too slowly if the degree transition points are set according to Equation (4.19). Thus, without adjustments of the degree, more codewords than the hypothetical optimum will have to be received. In a highly dynamic system, symbols are spread across the network and the assumption that symbols are uniformly drawn at random can be considered to hold. Thus, the formulas are valid in this case.

To conclude, it is important to keep in mind that a perfect source scenario is considered in the case of GC. In particular, all symbols are assumed to have the same probability to be contained in a codeword. Further assumptions include that each round exactly one codeword is received and this codeword fulfills the proposed degree according to the degree transition points. Only under these assumptions, the following proposition is true: GC reconstruct as many symbols as possible with any number of received codewords.

4.4 Network Coding Based Data Dissemination Approaches

After common coding schemes have been described in detail, related work concerning data dissemination will be presented next. In the following, only distributed approaches without central control are considered. Moreover, all of the addressed scenarios have multiple sources/multiple original symbols, i.e. a single message broadcast scenario is not regarded, and multiple sinks/storage nodes are assumed. Overall, the presented approaches have the same basic requirements as assumed in Section 3.1. Data dissemination approaches which are compliant with the system setup and the core ideas of Section 3.1 can aim for reliable communication but also for permanent distributed storage of data in the network. Thus, the distribution process itself as well as distributed storage will be considered in the following. However, the BCGC procedure presented here can be assigned to data dissemination focusing on reliable communication and not to distributed storage approaches. For this reason, only a brief introduction of network coding based distributed storage follows, which concludes with a differentiation between BCGC and distributed storage.

4.4.1 Distributed Storage Based on Network Coding

Here, the term distributed storage refers to distributed storage based on network coding. Distributed storage approaches that use the idea of network coding can be, for example, random linear coding based as in [Ace+05; DPR05; DPR06], or LT based such as in [LLL07; AKS08a; AKS08b]. In addition, there are also repair strategies for distributed storage, such as those presented in [Dim+10a; Dim+11], to further improve data persistence in the case of node failures.

With distributed storage, not all original data has to be stored in all storage nodes. Instead, all original data has to be kept reliably in the system so that it is possible to retrieve all data when it is requested, even in case of node failures. A distributed storage approach is, thus, about distributing data among the nodes in such a way that all original data can be reconstructed with high probability at the time when it is demanded. Therefore, depending on the approach, a fixed number of nodes or as few nodes as possible have to be queried to gather and finally reconstruct all original data. This also means that a system is considered without a present sink in the true sense. In the following, only distributed storage methods are considered which are based on a coding scheme comparable to network coding. Storage nodes encode received data if it is specified, using the encoding method provided by the distributed storage approach, and then store and/or forward the resulting codeword. Unlike BCGC, storage nodes keep received or generated codewords permanently in their storage and do not decode. Thus, storage nodes store codewords, not original data. In order to be able to gather and recover all original data when querying a certain number of nodes at any later point in time, redundancy of data across nodes is necessary. The redundancy management strategies, which in the cases considered here additionally include encoding, and distribution mechanisms used for this purpose are given by the respective distributed storage approach. These strategies can vary greatly from one approach to another. For example, which and how many packets/symbols are used to generate a codeword can be determined differently. Moreover, the operation which is used for encoding as well as the maximum number of stored codewords per storage node can be different depending on the approach. Often, only exactly one codeword is stored per storage node. Concerning distribution mechanisms, data can move randomly through the network, e.g. by random walk via always a random neighbor, or be routed directly to a storage node, and push or pull methods can be applied to distribute data, for example.

Overall, distributed storage approaches differ from BCGC primarily in that each node permanently stores a codeword or a certain small number of codewords and not original data as with BCGC. Often, codewords are generated by a node from received packets and stored codewords for its own storage, instead of being used only for forwarding to neighboring nodes. The goal of distributed storage is to generate and distribute codewords to the different storage nodes in such a way that, later on, a certain number of arbitrary storage nodes can be queried to reliably gather all original data. In the case of distributed storage, storage nodes do not decode. Only after completion of the distribution process, storage for an arbitrary period of time, and a later query of several arbitrary nodes, the gathered codewords are decoded. Especially with erasure code based approaches and one stored codeword per node, as in [DPR06; LLL07; AKS08a], about as many nodes have to be queried as there are original symbols in order to retrieve all data. With BCGC, on the other hand, all desired original data should be able to be reconstructed after a fast distribution process and few packet transmissions in any node at the end, or at an earlier point in time in a few nodes. Usually, significantly fewer nodes are required to be queried here than with distributed storage approaches. Thus, a BCGC sink should be able to decode as many codewords as possible from a given number of received packets/codewords and reconstruct as many symbols as possible in order to store them subsequently. This means that with BCGC, or also e.g. with RLNC, what happens in one node is what happens in distributed storage approaches across all nodes, which is a crucial difference between distributed storage approaches and BCGC. With BCGC, the focus is on reliable communication and not on permanent distributed storage of data.

4.4.2 Data Dissemination Focusing on Communication

After related work concerning distributed storage has been briefly introduced, network coding based data dissemination with focus on the distribution process itself, i.e. on communication, will be presented next. Again, only distributed approaches for data dissemination without central control are regarded.

Some research on data dissemination that focuses on communication considers a **one-to-all** communication scenario. This means, it is assumed that there is only one source node which generates sensor data and all nodes of the network aim to receive, reconstruct, and store all generated data. Well-known research addressing this type of scenario can be found, for example, in [CWJ03; GR05], which are based on RLNC. In [CWJ03], an early practical implementation and simulation of RLNC in real packet based networks is presented by Chou et al. In [GR05], Gkantsidis and Rodriguez use RLNC in the general context of content distribution. In contrast to the conditions assumed here in Section 3.1 for BCGC, Gkantsidis and Rodriguez additionally use bidirectional communication in order to decide whether the created packet of a certain node should be requested. A major difference to BCGC is that in [CWJ03] and [GR05], symbols for a codeword are chosen randomly, which is typical for RLNC based approaches. Furthermore, as already mentioned, a one-to-all communication scenario is considered in [CWJ03; GR05]. Further approaches which also address a one-to-all communication scenario but are more similar to BCGC as they combine codewords specifically are presented by Keller et al. and Sadeghi et al. in [KDF08; STK09], for example. There, the choice of symbols for a codeword is, however, based on perfect feedback from neighboring nodes. Nodes use their perfect knowledge of which symbols their neighbors are missing for the composition of a next codeword. Such reliable knowledge is rarely available in real scenarios and is not compliant with the system setup in Section 3.1. Furthermore, in [KDF08; STK09], a star topology is assumed, which implies that only the source node performs network coding and sends its own symbols in suitable combinations directly to all nodes. Nodes do not receive packets from neighboring nodes except from the source node. This corresponds to the definition of channel coding rather than network coding. All mentioned approaches in this paragraph, which are presented in [CWJ03; GR05; KDF08; STK09], have in common that they consider a one-to-all communication scenario.

Research that addresses **many-to-many** communication will be presented in the remainder of this section since scenarios as similar as possible to those assumed for BCGC in Section 3.1 should be discussed here. A many-to-many communication scenario represents a multi-source multi-sink problem, as introduced in Section 2.1. This kind of scenario does not specify the type of physical layer transmissions, so unicast, multicast, or broadcast transmissions are possible and can be chosen separately. All-to-all communication, which is addressed here in Section 3.1, is a special case of many-to-many communication and will, thus, be focused in this section.

Overall, only procedures and analyses are presented here that assume a similar system setup as described in Section 3.1.

Also, this section does not discuss routing schemes as they are presented separately and distinguished from network coding in Section 2.4. Related work, which deals with the concrete path that packets take through the network, is, thus, not considered here. Instead, only related work in relation to the distribution of data via network coding is presented. This means, procedures are regarded that specify which data is contained in the individual packets, i.e. which data is combined for a respective codeword, not procedures that focus on the path of the packets. Furthermore, procedures which send only one symbol per packet, i.e. which correspond to a form of NoCoding, are not considered here. Instead, only network coding procedures, which combine several previously received symbols in one packet, are presented.

In short, only related work on data dissemination which addresses a many-to-many communication scenario, i.e. a multi-source multi-sink problem, and which uses a network coding based approach will be considered here. The related work presented in the following is divided into 1) research on network coding which analyzes the performance of existing network coding schemes and 2) concrete, developed network coding approaches.

4.4.2.1 Analyses on Network Coding for Data Dissemination

The widely used method of RLNC was first introduced by Ho et al. in [Ho+03a]. To generate a codeword for transmission, a linear combination of all symbols available in the considered node is calculated in the case of RLNC. The coefficients of the linear combination are chosen uniformly at random from the given Galois field. Thus, the number and the concrete choice of symbols used for a codeword is random, in contrast to the specific composition of codewords in the case of BCGC. This deterministic and purposeful composition adapted to the current situation is an essential aspect of BCGC. In [Ho+03a], the authors provide a lower bound on the probability of success for RLNC in a multicast network. Additionally, they show that their given upper bound on error probability decreases exponentially with increasing basis of the Galois field. Overall, Ho et al. show benefits of RLNC in comparison to routing without network coding. In [Ho+06], a more detailed follow-up research paper by the authors of [Ho+03a], it is additionally shown that RLNC asymptotically achieves the max-flow bound in a multi-source multicast network, i.e. a many-to-many communication scenario is considered. More information on RLNC can be found separately in Section 4.2 since RLNC is a fundamental, frequently used network coding scheme. Even though a network coding procedure in the true sense is presented in [Ho+03a], this research paper is mentioned in this first paragraph since all subsequently listed research containing analyses is based on RLNC.

In the area of random linear network coding based approaches which address a many-to-many communication scenario for data dissemination, [DMC04] and its follow-up [DMC05; DMC06] of Deb et al. are among the most well-known research articles. There, the special case of a many-to-all communication scenario is assumed. However, the all-to-all communication scenario, as assumed for BCGC, is also considered as a worst-case scenario. In [DMC04; DMC05; DMC06], Deb et al. examine uncoordinated information dissemination based on RLNC, which is in contrast to the specific composition of codewords in BCGC. Unlike for BCGC in Section 3.1, in [DMC04; DMC05; DMC06], a complete graph is assumed and, additionally, each node sends a generated packet to only one random neighbor per round. This means, there is no broadcast transmission to all neighboring nodes. In [DMC04; DMC05; DMC06], this transmission model is called *random phone call model* and corresponds to gossiping introduced in Section 2.4. Also, unlike in Section 3.1, a static network is assumed for the presented analysis. As a consequence, the system setup differs in significant aspects from the assumptions made in Section 3.1. Thus, the values presented in [DMC04; DMC05; DMC06] cannot be directly compared with our results. However, the considered figure of merit is the time it takes for all nodes to have all distinct original symbols, which is also one of the performance metrics for BCGC, the system latency. In [DMC04; DMC05; DMC06], the overall usefulness of RLNC for gossiping is analyzed theoretically and by simulation.

Mosk-Aoyama et al. and Borokhovich et al. address in [MS06; BAL10; BAL14] an all-to-all communication scenario and theoretically analyze how long it takes in this context until all nodes have everything when RLNC is used. Thus, the same general scenario is considered as for BCGC in Section 3.1, but with RLNC instead of a specific composition of codewords. The following assumptions also differ from those in Section 3.1, so the presented results of Mosk-Aoyama et al. and Borokhovich et al. cannot be directly applied to BCGC. In [MS06; BAL10; BAL14], a general static network is assumed, which can be an arbitrary connected network and does not have to be complete. Furthermore, unicast transmissions are considered which are additionally assumed to be lossless. In general, Mosk-Aoyama et al. analyze in [MS06] how the topology affects the performance of RLNC in the addressed scenario. Borokhovich et al. provide in [BAL10; BAL14] new tight bounds on latency and required number of transmissions.

In [Hae11; Hae16], Haeupler addresses an all-to-all communication scenario for data dissemination, as assumed in Section 3.1 for BCGC. Overall, the research articles [Hae11; Hae16] are similar to the already mentioned research of Mosk-Aoyama et al. and Borokhovich et al. concerning assumptions, used network coding scheme, and considered performance metrics. However, in addition to the tight bounds for an all-to-all communication scenario, bounds for the more general case of many-to-all communication are also provided in [Hae11; Hae16]. Moreover, in [Hae11; Hae16], not only static but also arbitrary mobile networks are considered. Furthermore, there are analyses for unicast transmissions but also for broadcast transmissions with asynchronous communication, exactly as assumed for BCGC in Section 3.1. However, Haeupler provides in [Hae11; Hae16] only a theoretical analysis of data dissemination where RLNC is assumed to be used as network coding scheme.

After related work which primarily contains analyses on all-to-all data dissemination via RLNC has been presented, research focusing on specific, developed network coding approaches now follows. Since BCGC represent a concrete network coding procedure, a comparison with related work from this category is particularly relevant.

4.4.2.2 Network Coding Procedures for Data Dissemination

Information dissemination in a static network as well as in an always connected mobile network is addressed by Haeupler and Karger in [HK11]. There, k symbols, which were generated in a distributed manner in the network, are to be sent to each of the n nodes of the network. For $k = n$ and the assumption that each node can only generate one symbol at most, this would correspond to an all-to-all communication scenario. The authors, therefore, consider a more general case, which they call k-token dissemination. Overall, the goal in [HK11] is to distribute all k distinct symbols to all nodes in as few rounds as possible, i.e. as fast as possible. Keeping system latency as low as possible is also one of the goals of BCGC. For the dissemination process, a network coding method shall be employed in [HK11]. In contrast to the deterministic BCGC network coding scheme with specific composition of codewords, RLNC is chosen as network coding procedure in [HK11]. Unlike for BCGC, it is also assumed that the k symbols do not have an ID in advance. For this reason, the authors in [HK11] start the dissemination algorithm with indexing. Here, each symbol is assigned a distinct ID between 1 and k, which is referred to as k-indexing in [HK11]. In the case of BCGC, by contrast, it is assumed that each node generates exactly one symbol and that nodes have a distinct ID which they then transfer to their generated symbol. Thus, there is no indexing in BCGC. The k-indexing is an essential part of the approach from [HK11] and orthogonal to the processes in BCGC. After k-indexing, the distribution of all k symbols to all nodes follows. For this, there is a theoretical analysis of how long the dissemination takes with and without RLNC in the case of lossless transmissions when each node sends a packet to all of its neighbors every round. Both the lossless transmissions and the round-based approach are contrary to the assumptions for BCGC. All in all, Haeupler and Karger present in [HK11] their own algorithms based on RLNC for k-token dissemination and provide approximations for the system latency. As already mentioned, a separate discussion of RLNC can be found in Section 4.2.

Fragouli et al. and Widmer et al. consider in [FWL05; WL05; FWL06; FWL08] all-to-all data dissemination like BCGC but use standard RLNC with $GF(2^8)$ for this purpose.

Thus, there is no specific selection of symbols for the generation of a codeword. In [FWL05; WL05; FWL06; FWL08], Fragouli et al. and Widmer et al. present a general theoretical analysis of how the use of RLNC in the all-to-all communication scenario improves energy efficiency as measured by the number of transmissions. In addition, the authors investigate the performance difference between probabilistic routing with and without RLNC under more realistic conditions via simulation. Without network coding, an innovative received packet/symbol is subsequently forwarded with a certain probability or a certain number of times when using the presented form of probabilistic routing. With network coding, a corresponding number of new packets/codewords is generated for transmission. Performance criteria are packet delivery ratio, i.e. the proportion of packets which can be decoded at a sink, system latency, and required number of transmissions. The difference in performance due to a particular choice of the so-called *forwarding factor*, which influences the probability and frequency of transmissions via broadcast for probabilistic routing with and without network coding, is examined in particular. In [FWL05; WL05; FWL06; FWL08], the focus is not primarily on the development/study of a network coding scheme itself but rather on how often a new codeword should be generated and sent after an innovative symbol has been received, which is determined by the used forwarding factor. The authors of [FWL05; WL05; FWL08] additionally also consider multiple generations of data, i.e. each source node produces multiple data sets instead of only one as assumed for BCGC in Section 3.1. These two aspects generally represent an orthogonal approach to our considerations. For example, the form of flooding assumed in Sections 2.4 and 3.1 could be replaced by the forwarding algorithm presented in [FWL05; WL05; FWL06; FWL08], with appropriate analyses for a suitable forwarding factor and certain modifications.

Another research paper with the goal of all-to-all data dissemination but which is also mainly focused on the transmission process rather than on the network coding scheme itself is [MHZ17] by Mager et al. There, RLNC with $GF(2)$ is used as network coding procedure, but with two minor modifications: First, as soon as an innovative symbol arrives, it is always added to the next codeword for transmission, and second, also a node's own symbol is always added to the codeword until this symbol has been received three times. However, a reason or an analysis why the value three was chosen here cannot be found in [MHZ17]. The further composition of codewords corresponds to standard RLNC. Main contribution of [MHZ17] is the realization of synchronous transmissions while using RLNC and having the goal to receive as many packets as possible per time unit. For this purpose, the so-called capture effect is made use of. However, the procedure presented in [MHZ17] is still a digital network coding approach. It is not assigned to analog network coding, since encoding is performed during the creation of packets and, thus, before transmission, and since codewords are not created by mixing signals. The adaptive transmission policy with synchronous transmissions presented by Mager et al. in [MHZ17] is orthogonal to the BCGC approach and could also be added to it. For this purpose, RLNC in [MHZ17] would have to be replaced by

BCGC.

Further research which addresses an all-to-all communication scenario for data dissemination is introduced in [BDF19] by Bromberg et al. There, a variant of RLNC is used as network coding procedure, instead of specifically composing codewords as in BCGC. Furthermore, contrary to the assumptions in Section 3.1 for BCGC, nodes are assumed to communicate via physical layer unicast transmissions in [BDF19]. The focus in [BDF19] is on handling multiple generations, which exceeds the abstracted assumptions from Section 3.1 and will, thus, be considered only in future work.

After related work based on RLNC has been presented, approaches which can be assigned to deterministic linear network coding and are more similar to BCGC with respect to the composition of codewords follow.

One of the most well-known approaches from the field of data dissemination which is more similar to BCGC, is introduced in [Kat+06; Kat+08] by Katti et al. Like BCGC, the presented procedure is a form of deterministic linear network coding and an adaptive network coding scheme. Furthermore, as in the case of BCGC, XOR operations are used for encoding and codewords are specifically composed. Also, the mentioned goal in [Kat+06; Kat+08] is to maximize the throughput by combining as many different symbols as possible in one codeword, i.e. in one packet. For the composition of codewords, state information of the neighboring nodes is used, similar to BCGC. This state information is achieved in [Kat+06; Kat+08], among other things, by complete reception reports, which are appended to each packet or are additionally sent as separate packets. Each node has to inform its neighbors regularly which exact symbols it has already reconstructed. Thus, in [Kat+06; Kat+08], a node has to know all of its neighbors, store the received exact information to each of its neighbors in respective tables and maintain the corresponding information. Additionally, nodes use knowledge or compute guesses from the sender/receiver information of received packets in combination with link quality information, which they obtain from the routing protocol and store for each link. Then, for each reconstructed received symbol, they determine and store with which probability which of the neighbors has already reconstructed the considered symbol. In the case of BCGC, this extensive, exact knowledge about the state of each neighbor or link quality information is not necessary and, in particular, a node does not even have to know all of its neighbors. In [Kat+06; Kat+08], only so many symbols are combined for each packet that all neighbors can decode the contained codeword immediately with certainty or with high probability. Unlike BCGC, the ability to decode immediately is the most important criterion, and a received packet whose contained codeword cannot be decoded immediately is discarded. The approach of Katti et al. can be assigned to inter-session network coding, like BCGC and all related work presented in this section before. However, it does not consider an all-to-all communication scenario but multiple independent unicast sessions. Overall, the approach of Katti et al. is designed specifically for unicast sessions. Due to the similarity in terms of specific codeword composition, use of XOR, and prominence of this research paper in the field of data dissemination, it is still included here. In [Kat+06; Kat+08], packets to be sent and tables to be stored can quickly become very large and the communication required to maintain these can be costly, especially if there are many different original symbols or nodes in the network. Due to the fact that a node always has to know all of its neighbors, has to determine and store which symbol has already been reconstructed with which probability by which neighbor, and has to know the link qualities, the approach presented in [Kat+06; Kat+08] is not suitable for large-scale, mobile networks. This, however, is one of the main assumptions in Section 3.1.

Further research on data dissemination based on deterministic linear network coding which addresses an all-to-all communication scenario, can be found in [Li+07]. There, the goal of the authors Li et al. is to minimize the number of transmissions for data dissemination. The presented approach belongs like BCGC to adaptive network coding and uses information about the state of the neighborhood to create codewords. Codewords are specifically composed using XOR so that all neighbors can immediately decode a received codeword. Unlike with BCGC, it is checked for each symbol if all neighbors can still immediately decode the codeword with the considered symbol before the symbol is added to this codeword. For this purpose, a node stores for each node from its 2-hop-neighborhood which symbols the respective node has already reconstructed. This information is not necessary for BCGC. Especially for large-scale networks with many different original symbols, these tables can quickly become very large, the associated information has to be exchanged, which can also become costly, and the maintenance of the tables can become complex. Furthermore, the approach presented in [Li+07] requires that the underlying routing procedure is deterministic, i.e. it is not possible to use any simple routing procedure as in BCGC. Deterministic routing, however, poses a challenge especially in large-scale, mobile networks as assumed for BCGC.

Another research paper on data dissemination in an all-to-all communication scenario is [Cos+08] by Costa et al. The presented procedure can be assigned to deterministic linear network coding and codewords are specifically composed by XOR combinations of symbols, as in BCGC. This composition, also like in BCGC, depends on the current state of the neighboring nodes, i.e. adaptive network coding is applied. However, the goal of the approach is not to maximize throughput by composing a codeword specifically to contain new symbols for as many nodes as possible as for BCGC. Instead, codewords are generated in a way that they can be decoded immediately by as many nodes as possible. This may lead to a reduction in throughput. Unlike BCGC, the approach of Costa et al. assumes perfect information about all neighboring nodes, i.e. neighboring nodes reliably inform the node under consideration which symbols they have already reconstructed. The node stores this information for each of its neighbors. The approach in [Cos+08] requires knowledge of all neighboring nodes and maintains a table for each neighbor with the symbols the respective neighbor has already reconstructed. Unlike with BCGC, it is checked for each symbol whether this symbol reduces the number of nodes at which the new codeword with the considered symbol is immediately decodable before adding it to the previously assembled codeword. A considered symbol is added if and only if this is not the case. With each additional symbol, a codeword has to be immediately decodable for the same number of neighbors as before or for more neighbors than before. For the approach in [Cos+08], perfect knowledge about the neighborhood and its state is assumed. However, collecting, storing, and maintaining this knowledge is problematic in large-scale, mobile networks with many original symbols. It is, thus, not suitable for the assumed system setup in Section 3.1. Moreover, the primary goal addressed in [Cos+08] is different from the goals of BCGC, and the fact that throughput may be reduced is inconsistent with the goals formulated for BCGC in Section 3.1.

Two last research articles on all-to-all data dissemination presented here are [Mun+07; Mun+08]. The approaches by Munaretto et al. can be assigned to adaptive, linear network coding, as BCGC. Also, as in BCGC, the XOR operation is used for encoding. The aim in [Mun+07] is to have to send as few packets/codewords as possible until all nodes have reconstructed all original symbols. In [Mun+08], the mentioned goal is to reconstruct as many original symbols as possible with each number of received codewords. In contrast to the previously listed related work, but as with BCGC, the authors of [Mun+07; Mun+08] choose the degree of a codeword specifically and adapted to the state of the system. When a node generates a codeword according to the specifications in [Mun+07; Mun+08], it chooses the degree depending on the number r of symbols already reconstructed in the node itself. For BCGC, by contrast, the number of reconstructed symbols in the neighboring nodes is considered, since these nodes should be able to decode the generated codeword after reception. Furthermore, in [Mun+07] a heuristic in the form of degree $d = r/13$ and $r/5$ is used without further analysis, justification, or indication of how the degree should be determined for a number of nodes other than $N = 100$. Therefore, the approach presented in [Mun+07] is difficult to apply to the scenarios considered here in Section 3.1 without further knowledge. In [Mun+08], there is a more detailed analysis of the degree than in [Mun+07]. However, in [Mun+08], idealized model conditions are used and, for each possible degree, a simulation is performed in which only codewords of this degree are generated. After these preliminary simulations, the simulation/degree that was best at a considered number of reconstructed symbols is chosen during the actual algorithm when the respective number of symbols has been reconstructed. The degree is, thus, determined offline under abstracted conditions and only partially adapted to the actual situation during the algorithm. In addition, this approach ignores the influence of the previous course of the respective simulation when choosing a degree, which is why the real course can differ from the optimal course obtained in the previous simulations. If the scenario is fundamentally changed, e.g. a different number of nodes is used, new preliminary simulations are necessary. In [Mun+07; Mun+08], it is assumed that all linearly independent received codewords are stored in the considered node and can be used for later encoding. In contrast to BCGC and the previously presented approaches from the field of deterministic network coding, in [Mun+07; Mun+08], symbols are not specifically selected for a codeword once the degree is determined. Instead, random codewords from the node's storage are combined via XOR until the predetermined degree is fulfilled. Thus, the approaches presented in [Mun+07; Mun+08] do not belong to pure deterministic linear network coding since symbols are chosen at random. However, there is a specification of the number of symbols to be used, which is why the procedures are no pure random linear network coding method but rather a hybrid of both. The approaches of [Mun+07; Mun+08] are closer to RLNC and will, therefore, be regarded as RLNC-based approaches here. For decoding, the authors of [Mun+07; Mun+08] propose Gaussian elimination, which may not be suitable under the assumptions made in Section 3.1. Moreover, preliminary simulations are problematic if, as assumed in Section 3.1, the algorithm should be able to start without time-consuming initialization phase and if, in particular, the network and the constraints are unknown in advance or may change during the course of the algorithm. For this reason, the approach presented in [Mun+08] is not suitable for the chosen conditions and possible application scenarios introduced in Section 3.1 for BCGC. In addition to the presented procedures in [Mun+07; Mun+08], the authors also discuss a buffer management system for continuous data gathering, i.e. they consider multiple generations. This aspect is orthogonal to BCGC and is left for future work.

Concluding this section, a tabular overview of the presented related work compared to BCGC follows in Table 4.1.

Research	Analysis/ procedure	x-to-x comm.	RLNC-based/ deterministic	Specific comp. of CW	Specific choice of CW degree	Required knowledge	Transmission
[CWJ03]	analysis	one-to-all	RLNC-based	-	-	no knowledge	n/a
[GR05]	procedure	one-to-all	RLNC-based	-	-	no knowledge	unicast
[KDF08]	procedure	one-to-all	deterministic	+	-	perfect knowledge	broadcast
[STK09]	procedure	one-to-all	deterministic	+	-	perfect knowledge	broadcast
[Ho+03a; Ho+06]	procedure	many-to-many	RLNC-based	-	-	no knowledge	n/a
[DMC04; DMC05; DMC06]	analysis	many-to-all	RLNC-based	-	-	no knowledge	unicast
[MS06]	analysis	all-to-all	RLNC-based	-	-	no knowledge	unicast
[BAL10; BAL14]	analysis	all-to-all	RLNC-based	-	-	no knowledge	unicast
[Hae11; Hae16]	analysis	all-to-all	RLNC-based	-	-	no knowledge	unicast+broadcast
[HK11]	procedure	many-to-all	RLNC-based	-	-	no knowledge	broadcast
[FWL05; WL05; FWL06; FWL08]	procedure	all-to-all	RLNC-based	-	-	no knowledge	broadcast
[MHZ17]	procedure	all-to-all	RLNC-based	-	-	no knowledge	broadcast
[BDF19]	procedure	all-to-all	RLNC-based	-	-	no knowledge	unicast
[Kat+06; Kat+08]	procedure	one-to-one	deterministic	+	-	perfect knowledge	pseudo-broadcast
[Li+07]	procedure	all-to-all	deterministic	+	-	perfect knowledge	broadcast
[Cos+08]	procedure	all-to-all	deterministic	+	-	perfect knowledge	broadcast
[Mun+07; Mun+08]	procedure	all-to-all	deterministic	-	+	no knowledge	broadcast
BCGC	procedure	all-to-all	deterministic	+	+	no knowledge	broadcast

Table 4.1: Related Work Data Dissemination

Chapter 5

Procedure of Broadcast Growth Codes

Contents

In this chapter, Broadcast Growth Codes (BCGC) are introduced in detail. First, the fundamental procedure of BCGC is presented in Section 5.1. Then, the two essential BCGC process parameters, the degree of codewords and the composition of codewords, are comprehensively described in Sections 5.2 and 5.3, respectively. Subsequently, the structure of a BCGC data packet is specified in Section 5.4. To highlight the advances of BCGC, the differences of BCGC to the network coding procedures Growth Codes and random linear network coding are discussed at the end of this chapter in Sections 5.5 and 5.6, respectively.

5.1 Fundamental Procedure of BCGC

The BCGC procedure presented here uses the assumptions made in Section 3.1 and is based on our following publications: [RK18; RK17; GK17; GHK16; GHK15]. Among others, it is assumed that each node generates a so-called symbol and aims to gather all distinct original symbols which were generated by the other nodes of the network. A multi-source multi-sink problem is, thus, regarded here. Each node is able to create codewords, decode received packets/codewords and stores the reconstructed original symbols. Therefore, it possesses a storage for reconstructed symbols and a so-called waiting list for received codewords which have not been completely decoded, yet. Furthermore, nodes are assumed to transmit packets via physical layer broadcast transmissions to all direct neighbors, as stated in Section 3.1. The time of sending and receiving of a packet is then realized according to the chosen MAC protocol. At initialization each node has only reconstructed its own symbol and, therefore, only this symbol is stored in the node's storage for reconstructed symbols. The BCGC scheme developed in this thesis is a network coding procedure for data dissemination based on Growth Codes, see Section 4.3. It combines, i.e. encodes, own and previously received data and then forwards a combination of these in the form of a codeword. For BCGC, as for GC, a codeword of degree d consists of an XOR combination of d distinct symbols and is sent as the payload in a BCGC packet. In the following, as in Section 4.3 on GC, the phrases 'transmitting/receiving a packet which contains a certain codeword' and 'transmitting/receiving a codeword' are used interchangeably since each packet contains exactly one codeword. The information necessary for encoding and decoding in BCGC is included in the packet header. A detailed description of the structure of a BCGC packet can be found in Section 5.4. Encoding and decoding by a node is done for BCGC, as for GC, using XOR only. This means that for decoding, in particular, no computationally expensive Gaussian elimination is performed to comply with the constraints from Section 3.1. In order to decode a codeword of degree d immediately after reception and to reconstruct a new symbol, exactly $d-1$ of the d symbols contained in the codeword have to be already known, i.e. reconstructed, by the receiving node. This means that a received codeword can be decoded immediately if it is a distance-1-codeword, see Section 2.1, for the considered node. If the received codeword is a distance->1-codeword for the node, the reduced codeword remaining after the decoding attempt is stored in the node's waiting list for possible later decoding. Unlike GC, however, BCGC do not consider only one sink, but each node is a sink, decodes received codewords, and should eventually reconstruct and store all distinct original symbols, see Section 3.1. Algorithm 5 presents an overview of decoding at BCGC. Since decoding in a node at BCGC is analogous to decoding in the sink at GC, see Section 4.3.5 for more details on decoding.

The goal addressed by BCGC is to ensure resilient communication and reliable data dissemination. Especially in a lossy, unreliable, large-scale WSN, where packets can

Algorithm 5 Reception and decoding of a packet/codeword (CW) in a node

```
1: procedure RECEIVEANDDECODE(packet)
2:     /*relevant for determining the degree of next CW to be sent, see Section 5.2*/
3:     if $d_{desiredMin} >$ packet.getDesiredDegree() then
4:         $d_{desiredMin} \leftarrow$ packet.getDesiredDegree()          // min. of neighbors' desired degree
5:     end if
6:     /*relevant for symbol selection for next CW to be sent, see Section 5.3.2*/
7:     if packet.getDesiredID() $! = -1$ then
8:         requestedSymbolList.add(packet.getDesiredID())          // ID of requested desired symbol
9:     end if
10:    cw $\leftarrow$ packet.getCw()          // extract contained CW
11:    listIDs $\leftarrow$ packet.getContainedSymbols()          // extract IDs
12:    for all ID $\in$ listIDs do
13:        if storageR.contains(ID) then          // if symbol already reconstructed
14:            cw $\leftarrow$ (cw $\oplus$ storageR.getSymbol(ID))          // remove symbol from CW
15:            storageR.increaseNumReceived(ID)          // relevant for symbol selection, see Section 5.3.1
16:            packet.setCw(cw)          // update CW in packet
17:            packet.updateHeader()          // update IDs of contained symbols
18:        else          // if symbol not yet reconstructed
19:            numInnovativeSymbols $++$
20:            innovSymbolID $\leftarrow$ ID
21:        end if
22:    end for
23:    if numInnovativeSymbols $> 1$ then          // not possible to decode CW yet
24:        waitingList.add(packet)          // store updated CW in waiting list
25:    end if
26:    if numInnovativeSymbols $= 1$ then          // CW decoded
27:        storageR.add(cw)          // contained symbol reconstructed
28:        storageR.increaseNumReceived(innovSymbolID)          // relevant for symbol selection, see Section 5.3.1
29:        checkWaitingList(innovSymbolID)          // remove reconstructed symbol from CWs in waiting list
30:    end if
31:    if allSymbolsReconstructed then
32:        setNodeFinished()
33:    end if
34: end procedure
```

collide and nodes can fail, it can be challenging to enable robust communication and retrieve data dependably. Each sink should eventually have all relevant data without any of the n distinct originally generated symbols being lost. Therefore, the data dissemination/collection process should be performed as bandwidth efficient as possible to enhance reliability. Each sink node should, thus, gather and reconstruct any number of distinct original symbols as quickly as possible. For this purpose, unnecessary redundant packets/codewords should be avoided. This also increases throughput and reduces energy consumption. Furthermore, it should not be a problem if single or many packets do not arrive. More details about the desired goals of BCGC can be found in Section 3.1.2.

To achieve these mentioned goals, BCGC have two essential parameters: First, the degree of a codeword to be created for transmission, i.e. how many symbols should be combined in a codeword via XOR. Second, the concrete symbol selection for this codeword, i.e. which specific symbols should be combined. The degree and the composition of codewords for transmission are discussed in more detail in Section 5.2 and Section 5.3, respectively. In both aspects, BCGC differ in significant ways from GC. These differences will be mentioned where relevant in the following and summarized separately in Section 5.5.

Codeword Degree In the case of BCGC, when a node creates a new packet for transmission, it determines its current desired degree for a packet to be received next and adds this desired degree to the packet header. The node's desired degree is determined, using a predefined so-called degree function, depending on the current state of the node, i.e. depending on how many symbols it has already reconstructed. Here, the number of symbols already reconstructed by the node is taken into account since the node can only use these symbols for decoding a received codeword. The more symbols are already reconstructed in a node, the higher is the probability that the node can immediately decode a received codeword of degree d. The degree function should, therefore, enable a node to desire a degree so that the node is likely to be able to decode a codeword of this degree immediately upon receipt and reconstruct a further symbol. The node's desired degree d should, thus, have such a value that a received degree-d-codeword is likely to represent a distance-1-codeword for this node. More details about the degree function follow in Section 5.2. Before creating a new codeword for transmission, a node considers all desired degrees received since its last packet transmission. For the new codeword, the node then uses the minimum of the desired degrees sent by its neighbors, starting at a degree of 1. In this way, no node is left behind. Nodes at the edge of the considered area are surrounded by a lower node density than nodes on the inner part of the area and have, thus, less neighbors sending a packet to them within a considered period of time. Thus, by using the minimum of the desired degrees, especially nodes at the edge or nodes which join the network later can also be finished since they do not only receive codewords with a degree which is too high. If it is not the goal that all nodes reconstruct each number of symbols as quickly as possible, as assumed here in Section 3.1, but for example that only one node should have all symbols as fast as possible, a different strategy may be more appropriate. Then, it could be advantageous to use, for example, the average or the maximum of the received desired degrees, but this will not be pursued here since it does not fit the assumptions made in Section 3.1.

Symbol Selection for Codewords In addition to a suitably chosen degree, a node can also specifically compose the content of a codeword for transmission in the case of BCGC. This means it can specifically select the symbols that are combined for the codeword. Like the degree, these symbols should be chosen in such a way that neighbors receiving the packet are likely to be able to decode the contained codeword immediately and reconstruct a new symbol. In the case of GC, a node adds only its own symbol to a codeword randomly chosen from its storage for received codewords before transmission. In other approaches which, like BCGC, also choose the codeword degree specifically, such as [Mun+07; Mun+08], only symbols/codewords randomly chosen from the node's storage for received codewords are combined. With BCGC, on the other hand, codewords are specifically composed. Thus, there is also no storage for received codewords. Instead, a node in BCGC, like the single sink in GC, possesses a storage for reconstructed symbols. The BCGC node then selects symbols for a new codeword from this storage of reconstructed symbols, and thus can specifically decide on both degree and symbol selection. Consequently, a predetermined degree is always fulfilled with BCGC, as opposed to GC. Furthermore, in contrast to GC, BCGC do not always add a node's own symbol in order to avoid that neighboring nodes receive the node's own symbol repeatedly with almost every packet.

For an overview of the creation and transmission of a packet/codeword by a node, see Algorithm 6. After the fundamental procedure of BCGC has been briefly introduced, essential aspects concerning the degree and symbol selection of a codeword will be explained in more detail next.

Algorithm 6 Creation and transmission of a packet/codeword (CW) by a node

```
 1: procedure CREATEANDSEND
 2:     $d_{desired}$ ← getOwnDesiredDegree()        // node's desired degree, see Section 5.2
 3:     /*if request for desired symbol is enabled and all but one symbol are reconstructed in the node*/
 4:     if $booleanIdRequest$ & $storageR$.size() = ($numSymbols$ − 1) then
 5:         $id_{desired}$ ← getOwnDesiredSymbol()        // node's desired symbol, see Section 5.3.2
 6:     else
 7:         $id_{desired}$ ← −1        // no desired symbol
 8:     end if
 9:     $d$ ← getCurrentProposedDegree()        // degree of CW to be sent, see Section 5.2
10:     /*composition of CW to be sent, according to Section 5.3*/
11:     $tmpList$ ← chooseSymbols($d$,$requestedSymbolList$)
12:     $storageR$.increaseNumSent($tmpList$)        // relevant for symbol selection, see Section 5.3.1
13:     $cw$ ← createCw($tmpList$)        // via XOR of chosen symbols
14:     $packet$ ← createPacket($cw$,$d_{desired}$,$id_{desired}$)        // according to Section 5.4
15:     transmitPacket($packet$)        // send packet to all neighbors
16: end procedure
```

5.2 Degree of Codewords for Transmission

The degree of a codeword for transmission can be determined for BCGC by a combination of two approaches: the degree function and an upper bound for the codeword degree. These two aspects will be described in detail in the following.

5.2.1 GC Degree Function

To specify the desired degree of a node, a degree function is used which matches the node's state. This provides a degree that is suitable for the node depending on the number of symbols which have already been reconstructed by the node. The degree is determined so that the considered node is likely to decode a codeword with this degree immediately upon reception and to reconstruct a further symbol. This means, the degree should theoretically maximize the probability for a distance-1-codeword. For the following formulas, it is assumed that all symbols have the same probability to be included in a codeword. The probability that a degree-d-codeword received by a considered node is a distance-1-codeword for this node can then be calculated by:

$$p_{Dist1}(r,d) = \frac{\binom{r}{d-1}\binom{n-r}{1}}{\binom{n}{d}}, \tag{5.1}$$

where n is the number of distinct symbols existing in the network, and r is the number of already reconstructed symbols in the considered node. Thus, $p_{Dist1}(r,d)$ denotes the probability that an incoming codeword of degree d contains exactly one symbol which has not been reconstructed yet if r symbols have already been reconstructed and all symbols have the same probability of being included in a codeword. In other words, $p_{Dist1}(r,d)$ is the probability that a received codeword can immediately be decoded if the assumptions of the formula are fulfilled. Equation (5.1) has already been presented as Equation (4.14) in Section 4.3.3, where it is used as the basis for determining the codeword degree in Growth Codes. The stochastic background of Equation (5.1) has also already been explained in detail in Section 4.3.3. Therefore, only essential aspects are mentioned here.

BCGC as well as GC aim to need as few received codewords as possible to be able to reconstruct a certain number of distinct original symbols, i.e. as much as possible should be decodable immediately after reception. Hence, the degree d which maximizes $p_{Dist1}(r,d)$ in Equation (5.1), with r already reconstructed symbols in the considered node, is theoretically optimal for a codeword arriving at a sink. As mentioned in Section 4.3.3 and in more detail in Section 4.3.6.1, the optimal degree is monotonically increasing with the number r of reconstructed symbols, starting at degree 1. Thus, the desired degree of a node should also be monotonically increasing for BCGC.

However, for GC, the maximization of the probability $p_{Dist1}(r,d)$ from Equation (5.1) is not directly implemented by the nodes or the single sink. Furthermore, a corresponding degree depending on the number of reconstructed symbols in the sink is not used. Instead, an average value K_d is calculated via Equation (4.19) in Section 4.3.6.2, based on the probabilities $p_{Dist1}(r,d)$ and under the assumption that all symbols have the same probability to be included in a codeword. This value K_d indicates the expected (maximum) number of degree-$\leq d$-codewords which has to be received by the sink. The original Growth Codes in [Kam+06b; Kam+06a] assume that each node performs exactly one bidirectional data exchange with a randomly chosen neighbor per round. The approximate number of rounds needed to reconstruct a respective number of symbols is, therefore, estimated by K_d and hard-coded into all nodes in advance as fixed, so-called degree transition points in [Kam+06b; Kam+06a]. There, degree transition points represent points in time when the proposed degree for all codewords which will be transmitted subsequently is incremented by 1. For a respective number of reconstructed symbols in the sink, the corresponding optimal degree according to Equation (5.1) is, thus, not certainly used. Instead, at each considered point in time, each node indirectly estimates via K_d how many symbols are probably already reconstructed in the sink and choose a degree matching this estimation. In order to meet the assumptions in Section 3.1, bidirectional data exchange has to be replaced by broadcast transmissions to all direct neighbors in the case of GC. Then, the sink's required number of received packets has to be estimated directly by K_d instead of using the respective number of rounds. Nevertheless, this is still only a rough estimation of the optimal degree for the sink since Equation (4.19) is used. In addition, nodes only estimate the number of received packets at the sink by using their own number of received packets. Then, they

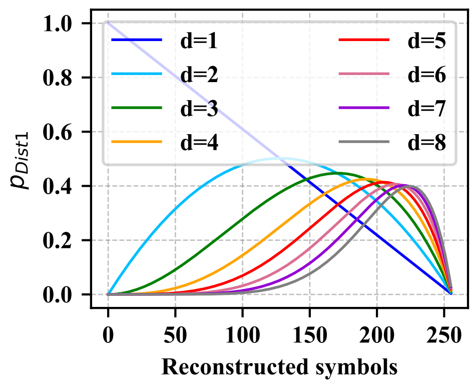


Fig. 5.1: Probability $p_{Dist1}(r,d)$ for degree d

determine via K_d the degree for the codeword they are currently creating. Hence, they still estimate the number of reconstructed symbols in the sink. For more details on the exact implementation of the degree transition points approach in GC, which is called GC degree distribution in [Kam+06b; Kam+06a], see Section 4.3.6.3.

BCGC, on the other hand, use Equation (5.1) directly as the basis for determining the degree of codewords for transmission, without performing any further estimation. This is possible for two reasons: First, a node in BCGC knows how many symbols it has already reconstructed, i.e. is able to compute its own optimal desired degree with respect to Equation (5.1). Second, the concept of adding a desired degree to each packet has been implemented at BCGC. The function used to determine the degree which maximizes Equation (5.1) is referred to as the 'GC degree function' in the following. Here, the maximization of $p_{Dist1}(r,d)$ from Equation (5.1) is performed analogous to the description in Section 4.3.6.1. Thus, until and including

$$\left\lfloor \frac{dn-1}{d+1} \right\rfloor =: R_d \tag{5.2}$$

symbols have been reconstructed by a considered node, its desired degree for a next received codeword should be $\leq d$, for $d \in [1; n-1]$. As soon as $r = R_d + 1$, the desired degree should be increased to at least $d+1$. Here, the term 'at least' is used since degrees can also be skipped, which is described in detail in Section 4.3.6.1. A degree $d+1$ is omitted if and only if $R_d = R_{d+1}$. In summary, Equation (5.2) implies that codewords of degree $\leq d$ should be received to reconstruct the first $R_d + 1$ distinct original symbols, starting at $d = 1$. Figure 5.1 shows the described aspects graphically for $n = 256$ nodes by plotting the probability $p_{Dist1}(r,d)$ according to Equation (5.1) for each degree d listed in the legend. When maximizing Equation (5.1), the degree d which has the highest probability $p_{Dist1}(r,d)$ for the current number r of reconstructed symbols in a considered node is used as this node's desired degree. The GC degree function is,

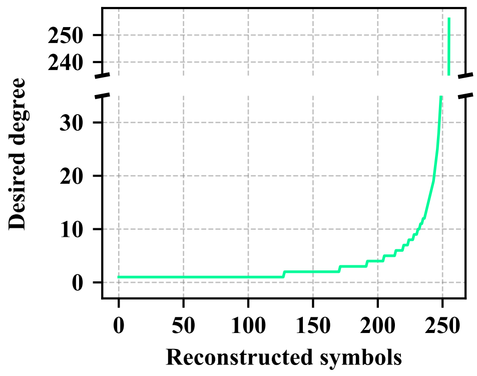


Fig. 5.2: Desired degree for respective number of reconstructed symbols

thus, an envelope of the plotted $p_{Dist1}(r,d)$-curves. The result of this maximization, i.e. the GC degree function described here, can additionally be seen for $n = 256$ nodes in Figure 5.2. Figure 5.2 shows a node's desired degree for a packet received next, which is added to the header of the node's current packet, depending on the number r of reconstructed symbols in the node. The respective degree transition points can be recognized in Figure 5.2 in the form of stair steps.

Using Equation (5.1), the expected (maximum) number of codewords which has to be received to reconstruct a respective number of symbols can be determined for each degree. This has already been explained in detail in Section 4.3.6.2. For codewords of degree $d \geq 2$, the term 'maximum/at most' had to be added here, as already for GC in Section 4.3.6.2, since Equation (4.19) does not consider that received distance->1-codewords are not discarded. Instead, these codewords are stored in partly decoded form in the node's waiting list. With each newly reconstructed symbol, a further attempt is made to decode this codeword. More details about this issue can be found in Section 4.3.6.2. Furthermore, all mentioned results are only valid under the assumption that all symbols have the same probability to be included in a codeword. Otherwise, the probability $p_{Dist1}(r,d)$ cannot be calculated via Equation (5.1).

For GC, the expected (maximum) number K_d of codewords of degree $\leq d$ to be received is calculated for a certain number r of reconstructed symbols only at the degree transition points, i.e. at $r = R_d + 1$, for $d \in [1;n-1]$. This is due to the fact that GC estimate and realize the degree transition in this way. There, the degree transitions $r = R_d + 1$ are used since until this number of reconstructed symbols, only a degree $\leq d$ should be received and a higher degree should be received afterwards, see Equation (5.2). The calculation of the values K_d is comprehensively described in

Section 4.3.6.2, including stochastic background information, and is executed via

$$K_1 = \sum_{\substack{[GC:]r=0 \\ [BCGC:]r=1}}^{R_1} \frac{n}{n-r} \tag{5.3}$$

and

$$K_d = K_{d-1} + \sum_{r=R_{d-1}+1}^{R_d} \frac{\binom{n}{d}}{\binom{r}{d-1}\binom{n-r}{1}}, \tag{5.4}$$

with $d \in [2; n]$ and $R_d := \left\lfloor \frac{dn-1}{d+1} \right\rfloor$ of Equation (5.2), see also Equation (4.18) and Equation (4.19). As already explained in Section 4.3.6.2, a symbolic $R_n := n-1$ is introduced here. In Equation (5.4), the calculation of the expected (maximum) number K_d of packets/codewords which has to be received by a node for a certain number of reconstructed symbols is executed only for the degree transition points, as for GC in Section 4.3.6.2. However, this calculation can also be executed for any other number of reconstructed symbols, taking into account the used degree in each case. In particular, the expected (maximum) number K_n of packets/codewords which has to be received in order to reconstruct all n distinct original symbols can also be calculated. Therefore, the effects of the use of different degrees for a fixed number of already reconstructed symbols can be compared with each other for example. This is used particularly in Section 5.2.2. For BCGC, Equation (5.3) starts at $r = 1$, instead of $r = 0$ as for GC, since each node has already stored its own symbol as reconstructed symbol at initialization. Figure 5.3 shows the expected (maximum) number $\Delta K_d = K_d - K_{d-1}$ of required codewords of degree d for $n = 256$ nodes. For this purpose, only the results for the degree transition points are calculated and the results for lower degrees are subtracted in each case.

5.2.2 Delayed Increase of Codeword Degree

For the GC degree function, only the required number of received packets/codewords to reconstruct a certain number of symbols is considered and minimized. The degree is increased if an increase is advantageous for the required number of packets. However, according to Section 3.1 the resulting latency has to be considered as well. Latency is not only influenced by the required number of received packets but also by the transmission time of a packet. Therefore, stochastic estimations combined with the packet transmission time for a packet of a given degree are used next. These estimate whether an indicated degree increase is beneficial also under consideration of the new package size. This approach is relevant and, thus, applied only in the case of dynamic packet sizes. Here, the expected maximum number of packets/codewords that would still have to be received if only the previous degree d was used is calculated. Then, it is

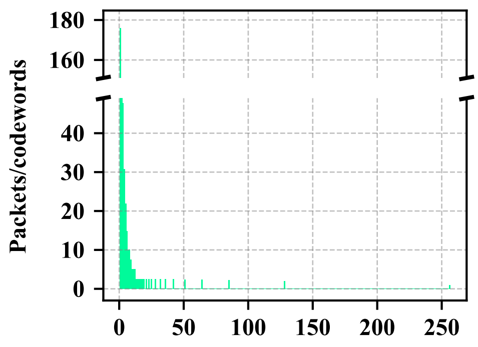


Fig. 5.3: Expected (max.) number ΔK_d of CWs to be received of respective degree

multiplied by the transmission time of such a degree-d-packet for each degree increase from d to $d+1$ specified by the used degree function. The result is compared with the corresponding expected maximum number of packets/codewords of degree $d+1$ multiplied by the transmission time of a degree-$(d+1)$-packet. An increase in the degree which is specified by the degree function should only be executed if a higher degree $d+1$ provides the better result in this comparison. Otherwise, the degree increase should be delayed and be checked again at the next number of reconstructed symbols. The degree function will then again suggest a higher degree than the current degree. In short, estimations are used which delay the increase of the degree if this is advantageous. The stochastic approach presented here can be applied to any underlying degree function. It only requires the current number of reconstructed symbols at which a degree transition is now to be checked and the currently used degree. In the following, the presented stochastic approach will be applied to the GC degree function. If the degree of a packet is increased by 1, a further ID is added to the packet header, i.e. the packet size is increased by 2 Bytes, see Section 5.4. At a data rate of 250 kbit/s, this increases the transmission time of a packet by 0.064 ms. For the estimation of the required number of received packets, Equation (5.6), which is described next, is used. The following Equations (5.5) and (5.6) were derived in the same way as Equations (4.16) and (4.17), but adapted to the situation considered here. Let X_r be a discrete random variable denoting the number of packets/degree-d-codewords which have to be received by a node until one further symbol can be reconstructed, i.e. until the next distance-1-codeword has been received by the considered node. Each of the X_r, with $r \in [0; n-1]$, is geometrically distributed with parameter $p = p_{Dist1}(r,d)$. Then, the expected value for X_r is

$$E[X_r] = \frac{1}{p_{Dist1}(r,d)} = \frac{\binom{n}{d}}{\binom{r}{d-1}\binom{n-r}{1}}, \tag{5.5}$$

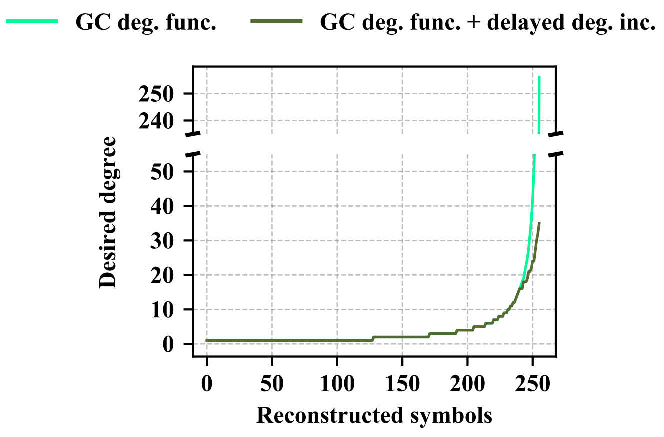


Fig. 5.4: GC degree function + delayed degree increase

if all symbols have the same probability to be contained in a packet/codeword. Thus, here, an expected number of $E[X_r]$ packets/codewords has to be received to reconstruct one further symbol if r symbols have already been reconstructed by the considered node. To calculate how many packets/codewords $E[\#packets]$ have to be received on average in order to reconstruct all remaining $n-\hat{r}$ original symbols, the expected value $E[X_r]$ for every possible remaining $\hat{r}$ has to be summed up, i.e.

$$E[\#packets] = \sum_{r=\hat{r}}^{n-1} E[X_r] \overset{eq.\ (5.5)}{=} \sum_{r=\hat{r}}^{n-1} \frac{\binom{n}{d}}{\binom{r}{d-1}\binom{n-r}{1}}. \tag{5.6}$$

For more detailed stochastic background information, see Section 4.3.6.2. In the stochastic approach described here, the value $E[\#packets]$ is calculated via Equation (5.6) for d and for $d+1$. The respective value is, then, multiplied by the associated packet transmission time. Subsequently, the results are compared. Then, it can be decided whether the desired degree at the point under consideration should be increased from d to $d+1$ or whether a degree increase should be delayed until later. Here, codewords of distance > 1 are not discarded like codewords of distance 0 but are stored in the waiting list with the option to be decoded later. Therefore, again, the expected value calculated via Equation (5.6) represents the expected maximum number of required packets/codewords to be received.

This stochastic approach results in the degree function which is depicted in Figure 5.4 when applied to the GC degree function with $n = 256$ nodes/symbols. It can be seen that the desired degree for a codeword to be received next is increased with a delay especially when there are already many reconstructed symbols in the considered node, i.e. in the later course of the data dissemination process. Furthermore, it can be seen that the degree does not increase to the maximum possible degree, which would be 256 in the considered case. Thus, in order to achieve a lower latency, it can be reasonable not to use the maximum possible degree. This insight is implemented in Section 5.2.4 by limiting the degree and evaluated in Section 7.1.1.2.

5.2.3 Degree Functions with a Greater Slope

If all symbols had the same probability of being included in a codeword, the GC degree function could be used to determine the optimal degree to be received next depending on the current number of reconstructed symbols in the considered node. If this optimal degree was then used for all packets sent to the node, the node could reconstruct as many symbols as possible immediately after reception as the optimal degree maximizes $p_{Dist1}(r,d)$ in Equation (5.1) per definition. Furthermore, Equation (5.4) could be used to calculate the expected (maximum) number of packets/codewords which has to be received. If dynamic packet sizes are used and not only the required number of packets but also the latency should be considered, the approach from Section 5.2.2 could also be used to estimate when and finally to which extent the codeword degree should be increased. In a realistic system, however, the probabilities of the symbols used to create a codeword are not uniformly distributed. A node can use only those symbols for the creation of a codeword which it has already reconstructed. Thus, especially in a static network or in a system with a low degree of nodes' mobility, symbols from the immediate surroundings are received very frequently in the beginning. Symbols from distant nodes are received much later. In the case of a low degree of nodes' mobility and a not extremely high node density in the network, only the same symbols are contained in codewords repeatedly, especially in the beginning. Therefore, many redundant packets/codewords are received after the first few symbols have been reconstructed. As a consequence, the probabilities $p_{Dist1}(r,d)$ calculated via Equation (5.1) do not necessarily equal the real probabilities. Let r be a considered small number of reconstructed symbols and d a degree previously determined by maximizing Equation (5.1), i.e. by using the GC degree function. Then, the actual probability $p_{Dist1}(r,d)$ is lower than the probability calculated via Equation (5.1). In addition, if a degree d given by the GC degree function is used, the real probability of receiving a degree-d-codeword as a redundant distance-0-codeword is higher than the probability $p_{Dist0}(r,d)$ computed analogously to Equation (5.1). This means, if there are only few reconstructed symbols in a considered node and if the neighboring nodes use the GC degree function to choose their respective desired degrees, the probability that the node creates a packet that contains a distance-0-codeword for a considered neighbor is increased compared to an idealized model scenario. For a higher degree $\hat{d}$ than the degree d recommended by the GC degree function at a considered number of reconstructed symbols, the actual probability $p_{Dist1}(r,\hat{d})$ may be larger than $p_{Dist1}(r,d)$ in these cases. The GC degree function, thus, may provide a degree d which is too low for the respective number of reconstructed symbols. Increasing the degree earlier than recommended by the GC degree function may be beneficial. In order to decrease the probability of receiving a distance-0-codeword and increase the probability of receiving a distance-1-codeword instead, the node can increase its desired degree compared to the degree specified by the GC degree function. This means, the node can use a degree function with a greater slope than the GC degree function. If the degree is chosen according to the GC degree function or, as for GC, according to the expected maximum number of received codewords via Equation (5.4), more redundant, unnecessary codewords arrive than predicted. Therefore, usually more codewords have to be received in reality to reconstruct a certain number of symbols than specified by Equation (5.4). Thus, the estimation for K_d via Equation (5.4) is also not valid, and the decision whether a degree should be increased with a delay using the approach from Section 5.2.2 is not reliable. However, the higher the degree of mobility or the higher the density of the network, the better the mixing of the distinct symbols. Then, the symbols' probabilities to be included in a codeword are more uniform, which means Equations (5.1) and (5.4) and the stochastic estimations from Section 5.2.2 become closer to reality. In addition, the increasing number of reconstructed symbols in a node supports this effect since exactly these symbols are available for the creation of a codeword.

Depending on the dynamics, density, and especially the number of reconstructed symbols in a considered node, weaker or stronger deviations from the GC degree function should be made. Thus, the desired degree should be increased earlier than suggested by the GC degree function. In the case of a deviation from the GC degree function and an earlier increase of the desired degree to reduce the probability of receiving a distance-0-packet, the degree should not be increased too early. Otherwise, the degree of a received codeword which fulfills the node's desired degree will be too high for the respective number of reconstructed symbols. Thus, the received codeword will be a distance->1-codeword for the considered receiving node. Received codewords with a distance > 1 cannot be decoded immediately, and it is, hence, not possible to reconstruct a new symbol directly after the reception of the respective codeword. This contradicts the goal of reconstructing a given number of symbols by a node with as few transmissions/received codewords as possible. In the worst case, only codewords with a certain distance > 1 were and are received by a considered node due to their too high degrees. Then, decoding via XOR is not possible and decoding in the considered node comes to a complete standstill. Assume a node is missing exactly 2 of 256 existing symbols, for example. If the node then requests a desired degree of 256 and receives corresponding codewords from its neighbors, these codewords will always have a distance of 2 for the considered node. As a consequence, the node will never be able to decode these codewords, no matter which symbols are included. Thus, deviating from the GC degree function can reduce the probability of receiving a distance-0-packet, but increasing the degree too early reduces the decodability at the receiver.

In the later course of a data dissemination process, when many symbols have already been reconstructed in a considered node, all symbols have approximately the same probability of being included in a codeword by the node. Equations (5.1) and (5.4), therefore, reflect reality again more and more accurately. The degree function should, thus, transition back to the the GC degree function, which then maximizes the probability of receiving distance-1-codewords and minimizes the required number of received packets to reconstruct a certain number of symbols. However, based on the insights from Section 5.2.2 and the fact that latency should also be taken into account here, it is to be expected that not using the maximum possible degree can provide better results than a pure transition to the GC degree function. This aspect will be addressed in more detail in the following separate Section 5.2.4 and will be evaluated in Section 7.1.1.2.

Finding a common solution for all nodes is a challenge and does not necessarily have to be possible. Due to the dynamic dependencies within the system, a solution can be good for one node and due to existing interdependencies, however, worsen the state of another node. Additionally, for any chosen scenario, especially for any degree of nodes' mobility or node density in the network, and any considered performance metric, a different optimum degree function is possible. For this reason, the evaluations in Section 7.1.1.1 analyze the impact of different slopes of the degree function using the default simulation settings specified in Section 6.2.2. Among these, the degree function which performs best under the considered conditions is then used for subsequent evaluations.

Overall, the degree function used here for BCGC is adapted to the current state of the network by each node requesting a desired degree for a next codeword to be received based on the symbols it has already reconstructed. The associated neighbors then send codewords of the most suitable degree to the requests they have received. However, the symbols reconstructed in a node are strongly influenced by the observed density and dynamics in the node's surroundings, as described earlier. In contrast to the GC approach in [Kam+06b; Kam+06a], the degree of codewords for transmission is dynamically adapted to the situation and not set in advance using fixed transition points only depending on the number of elapsed rounds.

5.2.4 Upper Bound for Codeword Degree

Here, the IEEE Std 802.15.4 [IEE06], which is described in detail in Section 3.2, is chosen for the specification of the communication on physical and MAC layer. In order to meet the specifications of IEEE Std 802.15.4 concerning packet sizes, see Section 3.2.2, the BCGC frame must not exceed 116 Bytes. For details on the BCGC frame structure, see Section 5.4. If the GC degree function or a degree function with a greater slope which at some point transitions back to the GC degree function is used, the desired degree is between 1 and n in the case of n nodes/symbols. However, with for example $n = 256$ nodes/symbols, a packet which contains a codeword of degree n and necessary information for decoding would exceed the IEEE Std 802.15.4 packet size specifications.

For this reason, it is necessary to limit the maximum possible degree depending on the used payload size by a suitably chosen upper bound *maxDeg*. This not only avoids large packets with codewords of very high degree but also reduces the size of packets with codewords of lower degree as certain fields in the packet header become shorter. With a payload size of, for example, 20 Bytes and $n = 256$ nodes/symbols, a maximum codeword degree of 46 can be used to comply with the packet size restrictions of IEEE Std 802.15.4, see Section 5.4.

Generally, it is possible to use static or dynamic packet sizes. Some applications require static packet sizes, for others applications, dynamic packet sizes can be used. When using static packet sizes, a codeword packet is filled with zeros to meet a specified fixed size. In contrast, only relevant information is included in a packet in the case of dynamic packet sizes, which may result in smaller packets. Here, the size of a created BCGC packet depends on the degree of the contained codeword. In general, if the packet size is smaller, the transmission time will be shorter. Thus, smaller packets have a lower probability of collision, and enable more successful transmissions per considered time interval. Therefore, an algorithm may terminate earlier, and better results concerning further performance metrics may be achieved. If a smaller upper bound *maxDeg* is chosen, packets become shorter. Especially with static packet sizes, this can have a big impact since in this case, packets have the same fixed size from the very beginning. For dynamic packet sizes, the difference is also noticeable, but only once the bound *maxDeg* would be exceeded in an approach without the presence of an upper bound. For the evaluations in Sections 7.1 and 7.2, the focus is on dynamic packet sizes and static packet sizes are only used if necessary.

From the moment the degree would exceed *maxDeg* without the bound, fewer distance-1-codewords and more redundant distance-0-codewords will be received by limiting the degree by *maxDeg* than without a deviation from the ‘optimal’ degree function. Therefore, more packets are needed to reconstruct the remaining respective numbers of symbols if an upper bound *maxDeg* is used. An estimation of the required number of packets can be calculated using Equation (5.6) from Section 5.2.2, assuming that all symbols have the same probability of being included in a codeword. Without a bounded degree and after having reconstructed $n-1$ symbols, the considered node’s desired degree is n, see Equation (5.2). If this is fulfilled by the neighboring nodes, i.e. a codeword of degree n is sent, only exactly this one more codeword has to be received to be able to reconstruct the last symbol. By contrast, in the case of for example $n = 256$ and $maxDeg = 46$, an expected maximum number of about 6 packets/codewords still has to be received for the last symbol, according to Equation (5.6).

Especially towards the end of the data dissemination process, all symbols have an approximately equal probability to be contained in a codeword. Thus, Equations (5.1) and (5.4) and the stochastic estimation from Section 5.2.2 reflect reality more and more accurately. When almost all symbols have been reconstructed by each node, the GC degree function returns a degree which theoretically minimizes the required number of packets/codewords to be received. If the codeword degree is limited by *maxDeg*, more packets may be needed, but packets become shorter. This aspect has already been considered in Section 5.2.2. There, an approach is presented which suggests to increase the degree only if the estimated required number of packets of the new degree multiplied by the new packet transmission time yields a better result than without the degree increase. It can, thus, be advantageous to limit the degree not only in order to comply with the packet size specifications but also with regard to latency. In Section 5.2.2, an example of 256 nodes/symbols and a payload size of 20 Bytes is considered. There, a correspondingly calculated estimate suggests that only a maximum degree of 35 should be used if the lowest possible system/node latency is to be achieved, see Figure 5.4.

In general, the given payload determines the range within which *maxDeg* can be chosen. To sum up, the smaller *maxDeg* is chosen, the smaller the packet sizes become but the more packets have to be received to reconstruct a certain number of symbols. An optimal upper bound *maxDeg* with respect to latency will, thus, be somewhere in between for a considered scenario. In addition, especially in the case of collision-prone transmissions, the use of $maxDeg < n$ can improve not only the latency but also reduce the number of sent packets. Depending on the addressed performance metric, a different *maxDeg* may be found to be optimal. Overall, results will differ depending on the payload and given conditions, such as the degree of nodes' mobility due to the different speed of spreading the symbols across the network. For this reason, the general effects of different bounds *maxDeg* will be evaluated in Section 7.1.1.2 under the default settings described in Section 6.2.2.

5.3 Composition of Codewords for Transmission

Especially in the case of a low degree of nodes' mobility or a low node density, the n existing distinct symbols do not have the same probability to be included in a codeword in the beginning of a data dissemination process. Therefore, more distance-0-codewords and fewer distance-1-codewords are received than theoretically expected, for example, when using the GC degree function. In addition, in the case of a limitation of the degree by $maxDeg < n$, which is suggested in Sections 5.2.2 and 5.2.4, the 'optimal' degree given by the respective degree function will not be fulfilled towards the end of the data dissemination process. Even though this can reduce latency, more codewords may have to be received since more distance-0-codewords and less distance-1-codewords are received than if the degree was fulfilled. In order to be able to reconstruct the last symbol, in particular, more packets/codewords have to be received than without the use of an upper bound for the codeword degree. For these reasons, it may be reasonable not to compose codewords randomly but to select symbols specifically. In this way, rare symbols are spread more quickly through the network and fewer packets/codewords are required to reconstruct a given number of symbols. Regarding the symbol selection, there are two approaches in BCGC, which are described in the following.

5.3.1 Specific Symbol Selection

In the case of GC, a node adds only its own symbol to a codeword randomly chosen from its storage before transmission. As a consequence, however, the proposed codeword degree for transmission cannot always be fulfilled with GC, since only the node's own symbol is added at most. For other approaches such as [Mun+07; Mun+08], symbols/codewords randomly chosen from the node's storage for received codewords are combined for a codeword to be transmitted. In the case of BCGC, on the other hand, a node can specifically compose a codeword for transmission in addition to an appropriately chosen degree. This means, the symbols that are combined for the codeword are specifically chosen. In this way, a degree determined for a codeword to be created next is always met with BCGC. The intention of this approach is to preferentially forward rare symbols so that they propagate through the network more quickly. Like the degree, the symbols should be chosen in such a way that neighboring nodes receiving the packet are likely to decode the contained codeword immediately after reception and to reconstruct a further new symbol from this codeword. Codewords should, therefore, be composed in such a way that they represent a distance-1-codeword for the neighbors with high probability. In general, for BCGC, it is not reasonable to use d for a neighbor probably profitable symbols in a packet/codeword for $d \geq 2$. The neighbor will, then, most likely not be able to decode this codeword immediately since it probably represents a distance->1-codeword for the node. This can also be seen in Section 7.1.2.1.

In the case of BCGC, a node can reconstruct at most 1 symbol by receiving a packet and decoding the contained codeword. For this reason, exactly one symbol is selected specifically here and all other symbols are chosen randomly when creating a codeword. Furthermore, the degree given by the suggested degree function is taken into account. Among all reconstructed symbols available for the creation of a new codeword, the symbol with the highest probability of being profitable for the neighbors is determined as the specifically selected symbol. Here, a symbol being profitable for a node means it has not been reconstructed by the considered node yet. For this purpose, each node collects information about how often it has already forwarded or received a considered reconstructed symbol. If a node re-transmitted a symbol before, it assumes that its neighbors are more likely to have already reconstructed this symbol than a symbol it has not re-transmitted yet. If the symbol was not generated by the node itself, the node originally received the symbol from at least one of its neighbors in the case of a static network or a low degree of nodes' mobility. Thus, at least one of the neighbors will probably know this symbol nevertheless. However, the expected number of neighbors who already know this symbol will be less than if the node already transmitted the symbol via broadcast. For this reason, the node chooses a symbol which it has never sent or a symbol which it has sent least often. If this applies to more than one symbol, the node chooses the symbol it has received least often.

By specifically adding one symbol as described before, rare symbols are more likely to be integrated into a codeword than symbols which are already frequently present in the vicinity of the considered node. Especially when only few symbols have already been reconstructed in the nodes and, thus, only few symbols are available for the creation of a new codeword, it is profitable to specifically select a rare symbol. In this case, the individual symbols have an unequal probability to be included in a codeword. Thus, the probability to send a redundant distance-0-codeword to the neighbors in the case of random symbol selection and a not highly dynamic system is increased compared to idealized model conditions. If codewords are specifically composed, fewer redundant packets/codewords are sent and rare symbols propagate through the network more quickly. However, since not all neighbors will still need the one specifically selected symbol, it is beneficial to choose the codeword degree appropriately.

With the chosen degree function as described in Section 5.2, there is no deterministic adaptation to the current degree of the nodes' mobility or the node density in the network. These two properties of the network influence strongly how evenly the probability of the symbols to be contained in a codeword is distributed, especially if nodes have reconstructed only few symbols each. Thus, they decide to what extent a degree given by a degree function which only takes into account the number of already reconstructed symbols in a node is suitable. In order to mitigate a possibly uneven distribution of the probabilities of the symbols, codewords are specifically composed by preferentially selecting one rare symbol at a time in the approach presented here. Thus, this approach is introduced to compensate for a low degree of nodes' mobility or low node density. In the case of a static network or a low degree of nodes' mobility, a node can gather information about the neighborhood and then create specific packets for the neighboring nodes. In the case of a highly dynamic system, information about the neighborhood does not provide any benefit since the neighborhood may already be completely different by the time the next packet is created. If symbols are specifically chosen in order to be suitable for the neighbors and the neighborhood changes, there is no relation between the collected information and the neighboring nodes. Consequently, the specifically selected symbols are as profitable as randomly chosen symbols and, therefore, do not cause any damage. In the case of a dynamic system, however, the probability of the symbols to be included in a codeword is more evenly distributed already early in the dissemination process. As a consequence, the given degree function alone, which is designed for random symbol selection with approximately equally probable symbols, ensures that the probability of distance-1-codewords arriving at a node is maximized. In the case of a static network or a low degree of nodes' mobility or a not highly dynamic network with a low node density, however, specifically selected symbols are profitable. Particularly in this situation, the probabilities of the symbols are not evenly distributed so that especially rare symbols take a long time to traverse the network. This aspect is now compensated by the specific composition of codewords for transmission.

In addition, especially in the case of a limitation of the degree by a chosen upper bound *maxDeg* $< n$, it is also beneficial to integrate a specific symbol into the codeword instead of using only random symbols. This bound is only reached in the later course of the dissemination process, so that all symbols then already have a similar probability of being included in a codeword. However, due to the limit of the degree, a lower degree is used than recommended for an 'optimal' required number of received packets. For this reason, in the case of random symbol selection, more distance-0-codewords would be received than without the reduction of the degree, see also Section 5.2.4. This can also be improved by a specific symbol selection as described here.

Due to the dynamic interdependence of degree function, symbol selection, and environmental conditions, these are difficult to separate analytically and are, therefore, studied via evaluations in Sections 7.1 and 7.2. An evaluation of the impact of a random symbol selection, a specific selection of exactly one symbol as described here, and a specific selection of all used symbols is presented in Section 7.1.2.1. In principle, it would be possible to try to compensate for the uneven symbol selection and the resulting high percentage of received distance-0-codewords by simply adjusting the degree function and selecting symbols randomly. However, this degree function can have to be quite different depending on conditions such as the degree of nodes' mobility or the node density and, thus, can even have to vary for nodes within a network. Therefore, an optimum may only be found for a specific considered situation. This optimum is, then, not necessarily optimal for other conditions. For this reason, a second complementary approach was chosen here to compensate for the non-uniform symbol selection, the specific codeword composition. Using this approach, rare symbols are used preferentially, thus, aiming for a more uniform symbol selection. However, only estimates can be made here which symbols might be profitable for the neighbors. For a low degree of nodes' mobility, this estimation will be profitable but is, however, still only an estimation and will not apply for each neighbor. Thus, it is reasonable to have chosen a suitable degree function as a basis. The higher the degree of nodes' mobility, the more the estimations can deviate from reality and, thus, be equally well or poorly suited as randomly chosen symbols. In this case, the chosen degree function is crucial.

5.3.2 Requested Desired Symbol

Besides the proactive approach described in Section 5.3.1 to specifically compose a codeword, BCGC also provides the possibility to use a reactive approach concerning the symbol selection. Instead of estimating which symbols are likely to be beneficial to the neighbors, this second approach involves responding to a request for a desired symbol from the neighbors. If a node is missing only one symbol and a limitation of the degree by *maxDeg* $< n$ is used, with BCGC, a node can request the last symbol it has not yet reconstructed. For this purpose, the node adds the ID of the missing symbol to its packet header. However, this increases the packet size by 2 Bytes and, thus, the packet transmission time by 0.064 ms. A neighbor sends the most frequently requested symbol since its last packet transmission which it has already reconstructed as one of the d symbols in its next codeword/packet. As soon as the node's request is fulfilled by a neighbor, the node only needs exactly this packet/codeword to be able to reconstruct its last missing symbol, just as if it had received a degree n codeword. Often, nodes within a node's vicinity are missing the same symbols as the considered node. Thus, if a neighbor combines all received requested symbols in one packet, this may be worse for nodes which are still missing more than one symbol as the codeword then probably represents a distance->1-codeword for them. If only one of the desired symbols is used for the codeword, the probability that those nodes can decode the codeword and reconstruct the other node's desired symbol as well is higher. If the goal was that only one node or few nodes should be completed as fast as possible, combining all desired symbols could be beneficial. However, since all nodes should be able to decode a received packet with high probability immediately after reception here, the previously described approach was focused.

For this described approach of requesting a desired symbol, larger packets are needed to request a symbol. On the other hand, the approach can reduce a node's required number of received packets/codewords to reconstruct all original symbols. The original goal of this approach was to accelerate the reconstruction of the last missing symbol of a node, if a degree restriction by *maxDeg* is used, in order to improve the system latency. However, even without using a desired symbol, with for example $maxDeg = 46$ and 256 symbols, only an expected maximum number of about 6 packets/codewords has to be received to reconstruct the last symbol, see Section 5.2.4. Saving those 5 packets corresponds to only about 2 percent of the absolute minimum required number of received packets. Therefore, in this case, there is only a small potential for improvement by using a desired symbol. More details are shown in the evaluation in Section 7.1.2.2. The properties of the network itself have a much greater impact on system latency than the last symbol to be reconstructed, see Section 7.1.1.

5.4 Data Packets in BCGC

As payload, a BCGC packet contains the created codeword, which is an XOR combination of the symbols selected for the current packet. Here, a representation in Bytes is used. If the size of an original symbol is b bits, the payload of a BCGC packet, i.e. the codeword, has $\lceil b/8 \rceil$ Bytes since XOR retains the size. In order to be able to identify which symbols have been combined in the codeword, i.e. to be able to decode the codeword, the IDs of the symbols combined via XOR have to be specified in the packet header. Here, IDs are assumed to be 16-bit short addresses as explained in Section 3.2.2. In the case of BCGC, the degree of a codeword is limited by *maxDeg*, so a maximum number of *maxDeg* different symbols can be included in a codeword. Depending on the number of used symbols, the listing of the IDs, therefore, requires up to $maxDeg \cdot 2$ Bytes. In the case of dynamic packet sizes, the packet header must contain the codeword degree in order to be able to determine the size of the ID field. This requires $\lceil \log_2 maxDeg \rceil$ bits, with $maxDeg \geq 2$. Furthermore, the node which generates the packet adds its current desired codeword degree to the packet header. Due to the degree limitation by *maxDeg*, $\lceil \log_2 maxDeg \rceil$ bits are necessary for this purpose as $maxDeg \geq 2$. In addition, the ID of the node's requested symbol is added to the packet header when the node is missing only one symbol and if the approach of a requested desired symbol is used. In this case, additional 2 Bytes have to be included in the header later on in the simulation. The bit/Byte values given here refer to the usage of dynamic packet sizes. In the case of static packet sizes, the packet degree can be omitted in the packet header. However, the specification of the contained symbols always needs exactly $maxDeg \cdot 2$ Bytes when using static packet sizes. In addition, it is necessary to take into account the required space of 2 Bytes in the header for the ID of the node's requested symbol from the very beginning if the approach of a requested desired symbol is used. Assuming a 16-bit short address, a BCGC frame, thus, contains:

- the used codeword degree if dynamic packet sizes are assumed ($\lceil \log_2 maxDeg \rceil$ or 0 bits),
- the IDs of the symbols which have been combined in the codeword ($\leq maxDeg \cdot 2$ Bytes),
- the desired codeword degree of the node ($\lceil \log_2 maxDeg \rceil$ bits),
- the ID of the node's requested symbol if the approach of a requested desired symbol is used, which is only sent when the node still has to reconstruct one symbol or in the case of a static packet size (2 or 0 Bytes), and
- one codeword (payload) ($\lceil b/8 \rceil$ Bytes).

The total packet size has to be rounded up to the next Byte since the PHY payload can only be specified in Bytes according to IEEE Std 802.15.4 [IEE16b], as already explained in Section 3.2.2. In the case of dynamic packet sizes, the total BCGC frame overhead is between $2 + \lceil 2 \cdot \lceil \log_2 maxDeg \rceil / 8 \rceil$ Bytes and $maxDeg \cdot 2 + \lceil 2 \cdot \lceil \log_2 maxDeg \rceil / 8 \rceil + 2$ Bytes, depending on the degree $d \in [1; maxDeg]$ of the contained codeword and if the approach of a requested desired symbol is used. Assuming the smallest reasonable maximum degree of $maxDeg = 2$ and a codeword degree of 1, which causes the lowest possible overhead, the total BCGC frame overhead is $1 \cdot 2 + \lceil 2 \cdot \lceil \log_2 2 \rceil / 8 \rceil = 3$ Bytes. Therefore, the minimum total BCGC frame overhead is 3 Bytes, regardless of the number of nodes/symbols in the system. The relative overhead is between $(1 \cdot 2 + \lceil 2 \cdot \lceil \log_2 maxDeg \rceil / 8 \rceil) / \lceil b/8 \rceil$ and $(maxDeg \cdot 2 + \lceil 2 \cdot \lceil \log_2 maxDeg \rceil / 8 \rceil + 2) / \lceil b/8 \rceil$. In

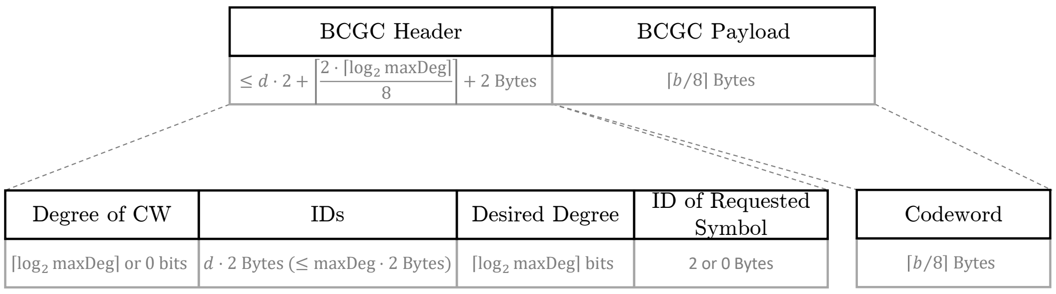

BCGC Header
BCGC Payload
$\leq d \cdot 2 + \left\lceil \frac{2 \cdot \lceil \log_2 \text{maxDeg} \rceil}{8} \right\rceil + 2$ Bytes
$\lceil b/8 \rceil$ Bytes
Degree of CW
IDs
Desired Degree
ID of Requested Symbol
Codeword
$\lceil \log_2 \text{maxDeg} \rceil$ or 0 bits
$d \cdot 2$ Bytes ($\leq$ maxDeg $\cdot$ 2 Bytes)
$\lceil \log_2 \text{maxDeg} \rceil$ bits
2 or 0 Bytes
$\lceil b/8 \rceil$ Bytes

Fig. 5.5: BCGC frame format

the case of *maxDeg* $= 46$, at the beginning of the BCGC process, i.e. with a codeword degree of 1, there are 4 Bytes of total BCGC frame overhead. For *maxDeg* $= 46$ and a codeword degree of 6, which is used until about 1/3 of 256 existing symbols have been reconstructed, the overhead is 14 Bytes. If less than 3% of the 256 symbols are missing and *maxDeg* $= 46$, the maximum degree is used as codeword degree. Then, the total BCGC frame overhead is 94 Bytes, and 96 Bytes if only one symbol has to be reconstructed and the approach of a requested desired symbol is used. However, such a high degree is used only for a few packets. Crucial about BCGC packets is that the overhead does not increase linearly with the number of nodes/symbols n as it does for RLNC. Instead, the overhead is limited via the maximum degree *maxDeg*, which can be chosen arbitrarily between 2 and n. For this reason, given packet size constraints can easily be met with BCGC. If a lower maximum degree is chosen, packets are significantly smaller, but more packets have to be received to reconstruct all n symbols.

The structure of a BCGC frame, which contains a degree-d-codeword and represents the MAC Data Payload of Figure 3.2, is depicted in Figure 5.5. Please note that the proportions in the figure do not reflect the actual size ratios here.

Considering physical and MAC layer, an IEEE Std 802.15.4 data packet has an overhead of 17 Bytes, see Section 3.2.2. A BCGC packet consists of the physical and MAC layer overhead and the BCGC frame. Thus, a packet's total overhead is between $17 + 1 \cdot 2 + \lceil 2 \cdot \lceil \log_2 \mathit{maxDeg} \rceil / 8 \rceil$ Bytes and $17 + \mathit{maxDeg} \cdot 2 + \lceil 2 \cdot \lceil \log_2 \mathit{maxDeg} \rceil / 8 \rceil + 2$ Bytes in the case of dynamic packet sizes if the approach of a requested desired symbol is used. Since the minimum BCGC frame overhead is 3 Bytes, a packet's minimum total overhead is 20 Bytes if physical and MAC layer overhead and the BCGC frame are considered. This means, IEEE Std 802.15.4 BCGC packets are at least 20 Bytes in size. With a data rate of 250 kbit/s, the transmission time of a 20-Byte-packet is $\frac{20\,\text{Bytes}}{250\,\text{kbit/s}} = 0.64\,\text{ms}$. An IEEE Std 802.15.4 PHY Packet is restricted to a maximum of 133 Bytes, see Section 3.2.2. When considering a data rate of 250 kbit/s, the maximum transmission time of an IEEE Std 802.15.4 data packet, i.e. the transmission time of a

133-Byte-packet, is $\frac{133\,\text{Bytes}}{250\,\text{kbit/s}} = 4.256\,\text{ms}$.

Due to the 17 Bytes of overhead considering physical and MAC layer, 116 Bytes are available for the BCGC frame to stay within the specifications of the standard. Of these 116 Bytes between $1 \cdot 2 + \lceil 2 \cdot \lceil \log_2 \mathit{maxDeg} \rceil / 8 \rceil$ Bytes and $\mathit{maxDeg} \cdot 2 + \lceil 2 \cdot \lceil \log_2 \mathit{maxDeg} \rceil / 8 \rceil + 2$ Bytes are needed for the BCGC header in the case of dynamic packet sizes if the approach of a requested desired symbol is used. The remaining Bytes can be used for payload or be omitted.

Discussion of Usefulness For BCGC, a maximum degree $\mathit{maxDeg} = 1$ will not be considered. In this case, symbols would only be sent in a non-encoded form as with NoCoding but would have the BCGC overhead instead of the shorter NoC overhead. If only degree-1-codewords are to be sent throughout a simulation run, NoC should, thus, be preferred. A *maxDeg* of at least 2 is, therefore, considered in the case of BCGC. This means, codewords consist of one symbol for a short time at the very beginning and of at least two symbols during the remaining course of a simulation run. In principle, it would be possible to send d NoC packets one after the other instead of one degree-d BCGC packet. Sending only a degree-d BCGC packet instead of d NoC packets may save Bytes on the IEEE Std 802.15.4 packet header but mainly saves on the payload/codeword. In the case of d NoC packets, d times a payload (a symbol) is sent. By contrast, the payload is sent only once and still includes all d symbols in the case of a degree-d BCGC packet. The size of the BCGC payload is the same as the size of one NoC payload since the result of an XOR combination of symbols has the same size as a single symbol. The gain of coding therefore refers to the payload/codeword. However, with the BCGC packet, some additional header information is needed as described in Section 5.4. Here, the specification of the included IDs is the same for d NoC packets as for one degree-d BCGC packet. As a consequence, no Bytes are saved with respect to the IDs in the header. In the following, it shall be determined until which payload size it only makes sense to use concatenated NoC packets. This is the case if d NoC packets have together a smaller size than one corresponding BCGC packet. These d NoC packets also provide in principle up to d different symbols, instead of only 1 symbol, and no additional encoding and decoding effort is required. Beyond this value, BCGC could be beneficial. When considering d IEEE Std 802.15.4 NoC packets and a degree-d IEEE Std 802.15.4 BCGC packet, it can be derived from Section 4.1.3 and Section 5.4 that

$$1 \text{ BCGC packet} \overset{!}{<} d \text{ NoC packets} \tag{5.7}$$

$$\Leftrightarrow \lceil 2 \cdot \lceil \log_2 \mathit{maxDeg} \rceil / 8 \rceil \text{ Bytes} \overset{!}{<} (d-1) \cdot (17 + \lceil b/8 \rceil) \text{ Bytes}, \tag{5.8}$$

with 17 Bytes of IEEE Std 802.15.4 packet overhead considering physical and MAC layer and a payload size of $\lceil b/8 \rceil$ Bytes. With a theoretical payload of 0 Bytes, the maximum possible degree that still fits into an IEEE Std 802.15.4 packet is 56. Considering a larger *maxDeg* than 56 is, therefore, not necessary here. A lower *maxDeg* results in a lower BCGC overhead. In this case, *d* NoC packets already become larger than the corresponding degree-*d* BCGC packet, which means BCGC can possibly be beneficial, at a lower payload. Thus, using $maxDeg = 56$, Equation (5.7) can be transformed to

$$2 \text{ Bytes} \overset{!}{<} (d-1)\cdot(17+\lceil b/8\rceil) \text{ Bytes.} \tag{5.9}$$

This equation is always fulfilled, which means that a concatenation of *d* whole NoC packets does not have to be preferred to a degree-*d* BCGC packet. Alternatively, it would also be possible to place several NoC frames one after the other within one IEEE Std 802.15.4 packet so that the IEEE overhead does not affect the comparison. For a fair comparison, the 'packet degree' then also has to be specified for the concatenated NoC frames, which requires additional $\lceil\lceil\log_2 maxDeg\rceil/8\rceil$ Bytes in the header. Equations (5.7) and (5.9) then transform into

$$\underbrace{\lceil 2\cdot\lceil\log_2 maxDeg\rceil/8\rceil \text{ Bytes}}_{\leq 2 \text{ Bytes}} \overset{!}{<} (d-1)\cdot\lceil b/8\rceil + \lceil\lceil\log_2 maxDeg\rceil/8\rceil \text{ Bytes.} \tag{5.10}$$

The degree of generated codewords grows to *maxDeg* during a simulation run and $maxDeg \geq 2$, so the average degree sent does not remain at 1. Therefore, only degree $d \geq 2$ codewords are discussed here. It can easily be seen from Equation (5.10) that the higher the degree *d* of a BCGC codeword is, the smaller is the payload above which *d* concatenated NoC frames are no longer smaller than a corresponding degree-*d* BCGC frame i.e. above which BCGC can possibly be advantageous. The overhead of *d* concatenated NoC frames is $d \cdot 2 + \lceil\lceil\log_2 maxDeg\rceil/8\rceil$ Bytes. For $maxDeg = 56$ and only degree-2 BCGC frames, the overhead of two concatenated NoC frames is 5 Bytes and the BCGC frame overhead is 6 Bytes. Already from a payload size of 1 Byte, two concatenated NoC frames reach or exceed the size of one degree-2 BCGC frame. Moreover, the degree of codewords continues to grow if $maxDeg = 56$ is used. For higher degrees, the payload size from which *d* concatenated NoC frames reach or exceed the size of one degree-*d* BCGC frame is less than 1 Byte. These results can also be seen from Equation (5.10). Only in the case of $maxDeg = 2$, codewords do not grow beyond a degree of 2. As mentioned before, for a lower *maxDeg*, the payload size is smaller up to which *d* concatenated NoC frames are shorter or equal in length to a degree-*d* BCGC frame. For $maxDeg = 2$ and degree-2-packets, however, two concatenated NoC frames are already always bigger than one degree-*d* BCGC frame, regardless of the payload size. Due to the payload and the constraints on packet size, typically a lower *maxDeg* than 56 and usually also higher codeword degrees than degree 2 are used for BCGC. Consequently, the size of a corresponding number of *d* concatenated NoC frames easily exceeds the size of one degree-*d* BCGC frame. Mostly, there is no payload for which this is not the case, so BCGC can in principle always be useful. Furthermore, in scenarios as addressed in Section 3.1, usually no payload sizes smaller than 1 Byte are used so Equation (5.10) is mostly satisfied. Thus, BCGC can, in principle, be useful without any drastic restriction. For a very small payload and restricted conditions, it will be better to put d NoC frames one after the other instead of using a degree-d BCGC frame. However, once the payload is realistically sized, d or more NoC frames usually do not fit in the place of a BCGC frame in a packet due to the size. The degree d of a BCGC packet is chosen to maximize the probability that a packet/codeword is a distance-1-packet for the neighbors receiving it. If less than d symbols are used, i.e. less than d NoC frames are concatenated, the probability that a receiver does not yet know exactly one of the contained symbols is lower than for the corresponding degree-d BCGC packet. At the same time, the probability that a neighbor receiving the packet already knows all of the included symbols so that the packet is an unnecessary, redundant distance-0-packet is higher than for the corresponding degree-d BCGC packet. Overall, it might be beneficial to send $k \leq d$ concatenated NoC frames that just fit in the BCGC packet than k individual NoC packets each with its IEEE Std 802.15.4 header. However, this would be a variant of NoC. In addition, more packets of the size of a BCGC packet would be required since the specified degree d usually cannot be fulfilled. The performance of BCGC compared to NoC, especially regarding the required number of received packets to reconstruct a certain number of symbols, is evaluated in more detail in Section 7.2.

5.5 Differences between BCGC and Growth Codes

In the following, differences between BCGC and GC are briefly summarized. Concerning packet size, BCGC and GC differ only in the header size. In contrast to BCGC, GC no longer list the individual IDs of the symbols contained in the respective codeword in the header once a certain codeword degree has been reached. Instead, each of the n distinct original symbols is assigned a bit in the header in advance which then indicates whether the associated symbol is contained in the codeword. Since GC, unlike BCGC, do not limit the degree, this listing requires exactly n bits for n nodes/distinct symbols in the system. As a consequence, GC packet sizes do not scale well with increasing number of nodes and GC packets can become very large. BCGC packet headers, on the other hand, are independent of the number of nodes due to the limitation of the degree by *maxDeg* and, thus, scale very well. As mentioned above, for GC, nodes use certain bits in the header to indicate which symbol is included in the respective codeword and which is not. With such an indication, a new node and, thus, a new symbol cannot be added to the network without additional effort. With BCGC, however, it is no problem if a new node is added later in the data dissemination process, assuming that the ID of the new symbol is unique.

Unlike BCGC, GC only consider exactly one sink, which should eventually reconstruct and store all distinct original symbols. Furthermore, GC are originally based on rounds, and it is assumed that each node transmits/receives exactly one packet/codeword per round. For data exchange, the original GC only use unicast transmissions and assume bidirectional data exchange. By contrast, BCGC perform physical layer broadcast transmissions to all direct neighbors and bidirectional data exchange is not required. Concerning the used MAC protocol, BCGC is not restricted to round based approaches and multiple nodes can transmit at the same time.

In the case of GC, the sink has not yet reconstructed any symbol at the beginning. At initialization, each node stores its own symbol in each storage space of its storage which is used for data exchange. For the scenario considered here, the authors of GC suggest to use ten storage spaces in each node, i.e. the own symbol of a node is stored ten times per node. At initialization in the case of BCGC, each node has reconstructed its own symbol and, therefore, only this symbol is stored in the node's storage for reconstructed symbols once.

For GC, the proposed degree for a codeword to be sent next starts at 1 and is always increased by 1 after K_d rounds for $d = 1, ..., n-1$. These degree transition points are hard-coded into all nodes in advance, were set under idealized assumptions, and, in particular, are not adapted to the used network or the state of the sink. In the case of a low degree of nodes' mobility and a not extremely high node density, only the same symbols are contained in received codewords repeatedly in the beginning of a data dissemination process. Therefore, many redundant packets/codewords are received especially in the beginning. It is, thus, not possible to estimate before the start of the dissemination process after how many rounds which number of symbols will be reconstructed by the sink in order to choose a respective optimal codeword degree for the sink. The number of elapsed rounds is, therefore, only a rough estimate of the number of reconstructed symbols and can deviate greatly from the actual value. The optimal degree to be received next at the respective point in time can, thus, also be a completely different one. By contrast, in the case of BCGC, the desired degree of a node, and hence the degree sent by a neighbor, is determined depending on the current situation, i.e. on the current actual number of symbols reconstructed in the considered node.

Furthermore, in the case of GC, the GC degree function is used as a basis for the proposed degree using the mentioned estimation. By contrast, for BCGC, a degree function with a greater initial slope than the GC degree function is used. This degree function is applied to compensate for the uneven distribution of the probabilities of the symbols to be included in a codeword in reality.

In the case of GC, the proposed degree starts at 1 and increases to the maximum possible degree n if there are n nodes/distinct original symbols in the system. With BCGC, on the other hand, the degree can be limited by a suitably chosen upper bound in order to take into account not only the required number of received packets/codewords but also the transmission time of the respective packets. Therefore, for BCGC, system latency is considered when choosing the most suitable degree, but is not taken into account when determining the degree in the case of GC. Moreover, a GC packet header has to indicate for each symbol whether it is included in the associated codeword or not included. As a consequence, GC packets, in contrast to BCGC packets, become larger and larger with increasing number of nodes/symbols n, since GC do not provide for limiting the codeword degree.

In addition, in the case of GC, a node adds only its own symbol to a codeword randomly chosen from its storage for received codewords for data exchange. A proposed degree is, thus, not always fulfilled. Furthermore, neighboring nodes receive the node's own symbol with almost every packet in the case of a not highly dynamic system. Whereas with BCGC, one symbol is selected specifically and all others are selected randomly from the node's storage of reconstructed symbols for the creation of a new codeword for transmission. This means, a codeword is specifically composed and the predetermined degree is always fulfilled with BCGC. Thus, BCGC can specifically decide on both degree and symbol selection and takes into account the current state of the sink(s).

Due to the differences mentioned here, significantly more distance-0-codewords and fewer distance-1-codewords are received with GC than with BCGC. As a consequence, significantly more packets/codewords are required when using GC. Especially in static networks or in the case of a low degree of nodes' mobility and a not extremely high node density, this can be observed. GC were specifically designed for highly dynamic systems, which is a requirement for good performance of the GC algorithm. By contrast, BCGC are well suited to static and dynamic systems due to their design.

After the differences between BCGC and GC have been listed, the differences between BCGC and RLNC will be considered next.

5.6 Differences between BCGC and RLNC

In the following, differences between the procedure and features of BCGC and RLNC will be briefly presented. The payload size is identical for RLNC and BCGC as, in both cases, a codeword is as large as one original symbol. However, the size of the header is different. If there are n distinct original symbols, n coefficients have to be included in a packet header for RLNC, which require a total of $n \cdot \log_2 q$ bits in the case of $GF(q)$. This means, the RLNC packet size depends on the chosen Galois field $GF(q)$ and the number of nodes/distinct original symbols. The bigger the Galois field is or, especially, the more nodes exist in the network, the larger an RLNC packet header becomes. RLNC packets have the same size throughout the entire data dissemination process. By contrast, in the case of BCGC, only the IDs of actually used symbols have to be included in the packet header. This number of IDs, the codeword degree, can additionally be limited by an upper bound. In this case, BCGC packets start with a codeword of degree 1, i.e. are very small, and then grow with regard to the codeword degree only up to the chosen upper bound. The size of the BCGC header then depends on this chosen upper bound and not on the absolute number of nodes. Thus, the BCGC header scales much better than the RLNC header as the number of nodes in the network increases. In general, a larger packet size implies a longer transmission time. Thus, it takes longer to transmit a certain number of packets one after the other, and there may be a delay compared to a procedure with shorter packet sizes. Furthermore, a longer transmission time also implies a higher probability of collisions. Especially if there are many participants in the system who want to transmit packets, it is advantageous if packets are shorter so that the probability of collisions is lower.

For RLNC, a coefficient belonging to a respective original symbol has a predetermined position in the packet header. As a consequence, a new node/symbol can only be added to the network with additional effort in the case of RLNC. For BCGC, on the other hand, the IDs of the used symbols are listed in the header, so that it is easy to add new nodes/symbols to the system.

In the case of RLNC, original symbols and codewords can be used for encoding. Thus, recoding is performed in intermediate nodes. In contrast, BCGC first decode and then completely reassemble codewords in intermediate nodes. However, in the considered system setup in Section 3.1, it is assumed that each node aims to gather all distinct original symbols. Therefore, decoding in intermediate nodes is not unnecessary but required.

Decoding in BCGC starts when a first packet/codeword is received. It is not necessary to wait for linearly independent packets. By contrast, in RLNC, decoding is started only when n linearly independent packets have been received. Then, all n original symbols are reconstructed at once, unlike for BCGC which reconstructs symbols from the beginning. Until n linearly independent packets are received, no symbol can be reconstructed in the case of RLNC.

For BCGC, encoding and decoding is generally less complex than for RLNC as only XOR operations are used instead of modulo calculation with an irreducible polynomial and Gaussian elimination. In general, the XOR operation in BCGC seems to be identical to linear network coding with the Galois field $GF(2)$. However, BCGC does not choose coefficients randomly like RLNC, but specifically. This would correspond to intentionally setting the coefficients to 1 or 0 so that certain symbols are included or omitted, respectively. Furthermore, in the case of BCGC, a certain degree is used, i.e. only a certain number of symbols are combined with each other. A degree of 1 is used at the beginning and then the degree increases in the course of the data dissemination process for BCGC. The low degrees are necessary for the decodability in the case of BCGC because, unlike RLNC, no Gaussian elimination is performed but only decoding via XOR. The speed at which the degree is increased depends on the used degree function and the state of the network. By contrast, for RLNC, the ‘degree’ can be maximal, i.e. all symbols which were received in codewords before can be used for each following linear combination. But coefficients can also be 0 by chance depending on the chosen Galois field. The number of actually combined symbols corresponds to the number of coefficients which are not equal to 0. However, the larger the Galois field, the lower the probability that a coefficient will have a value of 0. In general, all previously received symbols and codewords of the node are used to create a new RLNC codeword, i.e. symbols are not specifically selected. The Galois field, then, decides on the coefficients and, thus, on the probability of redundancy. When using a larger Galois field, fewer packets are sent/received that are linearly dependent from the previously received packets, which corresponds to receiving less redundancy. Since initially only a few symbols are known by an intermediate node, the 'degree' is small at first and then becomes larger as the number of received RLNC packets increases. However, the number of original symbols which can be combined and, thus, are combined grows fast, especially in the case of a mobile network where often new symbols are contained in the received packets/codewords. An RLNC packet, or more precisely the corresponding linear combination, therefore quickly contains significantly more symbols than would be the case with BCGC. An RLNC packet, therefore, has a lower probability of being a redundant distance-0-packet for a receiver than a BCGC packet if the contained BCGC codeword was composed only of random symbols. However, in order to compensate for the increased redundancy due to the lower codeword degree in the case of BCGC, redundancy is actively avoided by specifically selecting symbols in BCGC. To conclude, depending on the chosen Galois field and the number of nodes/distinct original symbols, RLNC may create larger packets than BCGC. However, fewer RLNC-packets may be required to reconstruct all original symbols in a considered sink. In addition, the larger the Galois field is, the less linearly dependent packets, i.e. unnecessary packets, are received. Thus, even fewer packets have to be received to reconstruct all n original symbols. However, at least n packets are always required for RLNC. Which of the two procedures, BCGC or RLNC, is eventually better for which performance metric and, in particular, under which settings is investigated by evaluations in Section 7.2.

Chapter 6

Simulation Environment

Contents

In this chapter, the simulation environment and the used default configuration for subsequent evaluations is described comprehensively. The description of the simulation environment in Section 6.1 includes implemented radio transmission schemes, MAC protocols, initial topology, mobility models, patterns for node failures, and used energy consumption model in Sections 6.1.2 to 6.1.7. Here, the concept is applied that all but one parameter is chosen according to a given default configuration and the parameter which is evaluated is variable. For this purpose, the default configuration of BCGC process parameters and of relevant parameters relating to simulation settings are listed and explained in Sections 6.1.8, 6.2.1, and 6.2.2.

6.1 Description of Simulation Environment

The implemented network simulator models the behavior of n homogeneous nodes that generate sensor data, create codewords and packets, transmit and receive packets, process the data from received packets and decode contained codewords, and store received data. For this purpose, wireless communication between direct neighbors is simulated, including different radio transmission schemes and protocols which control the nodes' access to the shared medium. Starting from an initially set topology, movement of the nodes is also implemented and can be enabled. How these individual aspects are realized exactly and which options are available in each case is described here. Thus, in this section, relevant details of the implemented network simulator are presented. However, existing network simulators are listed first, including reasons for using an in-house implemented network simulator.

6.1.1 Benefits of Own Network Simulator

For the evaluation of BCGC a network simulator was used. However, BCGC were also run on a testbed, which is described in more detail in Chapter 8. A simulator is generally useful before and in addition to an implementation on real nodes/a testbed for the following reasons: First, to our knowledge, there are no testbeds with several hundred or thousand homogeneous nodes communicating with each other which, in particular, are also mobile and can adopt different degrees of mobility and movement patterns. Thus, very large-scale networks of this kind can hardly be studied in a testbed so far. Second, the position of nodes in a testbed is predetermined to a certain extent, so that the density and topology cannot be changed arbitrarily and, in particular, nodes cannot be placed randomly. If a certain large number of nodes is to be used, nodes in the testbed are often very close to each other. Thus, even with the minimum adjustable transmit power, only a densely occupied network can be considered. Furthermore, if only some nodes within a certain area are selected for the own experiment in order to reduce the node density, other experiments may be performed in parallel on the nodes located between the used nodes. In this case, many unexpected and hard to predict signal collisions can occur. Third, the availability of nodes in a testbed is limited in time and a timely availability of nodes is not guaranteed. Reasons for this can be, for example, extensive updates or maintenance work by the operator or experiments by other users taking a long time. Planning and reservation are necessary and nodes are not always available in the desired number or physical arrangement, especially not repeatedly in direct succession. Fourth, as in any real-world system, environmental influences can occur which affect the results and their comparability.

To conclude, the use of a network simulator is beneficial in order to analyze a considered procedure extensively in isolation and to compare its performance with other procedures without environmental/hardware influences and under all possible desired conditions.

In general, there are multiple widely-used existing network simulators, such as ns-3 [Hen+08; Pro21], MiXiM (OMNeT++) [Köp+08; Vik21], COOJA [Öst+07], GloMoSim/QualNet [ZBG98; SCA21], or NetSim [TET21].

Most existing network simulators are very complex and contain significantly more features than needed here. Furthermore, if at all possible, some modules would have to be modified or newly implemented to simulate exactly the conditions described in Section 3.1 or assumed for the network coding procedures used here. In addition, when using an existing network simulator, specific network coding procedures, such as Growth Codes, would still have to be implemented. Therefore, for the following evaluations, our in-house developed network simulator is used. It considers all requirements addressed in Section 3.1 and is specifically designed for the use of network coding procedures. Thus, an additional network coding procedure, such as BCGC, can be implemented and evaluated easily.

6.1.2 Implemented Radio Transmission Schemes

In our in-house developed simulator described here, two different options for a radio transmission scheme are implemented. These model signal propagation from a transmitter to each possible receiver through the wireless medium. The first scheme is simple and very abstract, but not realistic. The second scheme models a more realistic radio propagation which is, however, more complex and parameter values have to be set appropriately to the addressed environment.

One parameter value which has to be selected in advance for both radio transmission models is the transmit power. Concerning the transmit power, it is to be noted that a higher transmit power can increase the transmission range. However, it also increases the energy consumption of each transmission and can lead to more collisions, especially in a network with many participants as assumed in Section 3.1. Generally, it is possible to choose a fixed transmit power or to adjust it dynamically. There are many different approaches to adapt the transmit power dynamically depending on the considered application and requirements, see for example [Lin+16; Nar+02; Kub+03]. This topic is an extensive research area which is, however, outside the scope of this thesis.

6.1.2.1 Circular Transmission Area

The first implemented radio transmission scheme has a circular transmission area. There, each node has the same fixed transmission range and it is assumed that every node within the transmission range of another node can receive this node's transmissions. Furthermore, it is assumed that the received transmit power is constant within the entire circular area, no matter where the considered receiver is located. Thus, there is no signal attenuation and the transmission area is circular. However, in order to receive a packet successfully, the considered receiver has to be within the transmission range of the transmitting node for the entire duration of this packet transmission. Thus, the receiving node must not move out of or into this area after the start and before the end of the transmission. In addition, the receiver has to be active and must never be in sleep mode during the entire transmission. Otherwise, it will not receive the considered packet successfully. Depending on the chosen MAC protocol, nodes can transmit simultaneously. Therefore, the considered receiving node must also not receive any other transmission from another node or transmit a packet itself during the duration of the considered packet transmission. Otherwise, the packets will collide and the considered node will not receive any of the packets successfully.

6.1.2.2 Irregular Transmission Area Using Log-Normal Shadowing Path Loss Model

The second implemented radio transmission scheme is based on the so-called log-normal shadowing radio propagation model and causes an irregular transmission area. Furthermore, interference by other simultaneous transmissions is also taken into account. In contrast to the transmission scheme with circular transmission area, it is not the case that the full transmitted power is received at every position within a given circular area around the transmitter and nothing is received outside this circular area. Instead, in the second model presented here, the strength of the radio signal decreases with distance from the transmitter. The so-called *path loss* then denotes the reduction in signal strength along the path from the transmitter to the receiver and is dependent on the distance d. It is specified as the ratio of the transmitted power P_t to the received power $P_r(d)$, or in decibel representation as the difference between $P_t\,[dBm]$ and $P_r(d)\,[dBm]$. The decrease in the power of a signal is generally referred to as signal attenuation.

The received signal strength of a transmitted signal can be influenced by various factors. The first factor considered here in more detail is the free space path loss, which even occurs in vacuum without any obstacles, i.e. with direct line-of-sight, and increases with distance. Considering only free space path loss, one possible option to model radio propagation is via the so-called *Friis free space equation*:

$$P_r(d)\,[mW] = \frac{P_t G_t G_r \lambda^2}{(4\pi d)^2 L}, \tag{6.1}$$

which originates from [Fri46]. This equation in the form used here can be found in [Rap96]. The Friis free space equation gives the received power $P_r(d)$ in milliwatts at a distance d in meters from the transmitter in free space with unobstructed path between transmitter and receiver. Furthermore, only a single direct line of sight path is considered for Equation (6.1). Here, P_t denotes the transmit power in milliwatts, G_t the transmitter antenna gain, G_r the receiver antenna gain, λ the wavelength in meters, and L the system loss factor which is not related to propagation. According to

Equation (6.1), the received signal strength decreases quadratically with distance. With $G_t = G_r = L = 1$, the often used simplified Friis free space equation results:

$$P_r(d)[mW] = P_t \cdot \left(\frac{\lambda}{4\pi d}\right)^2. \tag{6.2}$$

The free space path loss can then be expressed in logarithmic representation in decibels as:

$$\begin{aligned} PL_{FS}(d)[dB] &= 10 \cdot \log_{10}\left(\frac{P_t}{P_r(d)}\right) \overset{eq.\ (6.2)}{=} \\ &= 10 \cdot \log_{10}\left(\left(\frac{4\pi d}{\lambda}\right)^2\right) = \\ &= 10 \cdot 2 \cdot \log_{10}\left(\frac{4\pi d}{\lambda}\right), \end{aligned} \tag{6.3}$$

which can also be found in [Rap96]. Here, the value 2 corresponds to the so-called *path loss exponent n*. In decibel representation, the free space path loss is the difference between the transmitted power and the received power, i.e.

$$PL_{FS}(d)[dB] = P_t[dBm] - P_r(d)[dBm]. \tag{6.4}$$

Equation (6.4) can be derived from $PL_{FS}(d)[dB] = 10 \cdot \log_{10}\left(\frac{P_t[mW]}{P_r(d)[mW]}\right)$. Since the value $P_r(d)$ cannot be calculated with Equation (6.2) for distance $d = 0$, path loss models often use a reference distance d_0 [Rap96]. The reference distance d_0 should be chosen accordingly so that all possible positions of receivers are usually further away from the transmitter. For indoor scenarios, typically, $d_0 = 1\,\text{m}$ is chosen, see for example [Rap96; Gar10]. At $d = d_0$, it is easily possible to calculate the value $P_r(d_0)$ or to use an empirically measured and determined value for $P_r(d_0)$. Substituting $d = d_0 \cdot \frac{d}{d_0}$ into Equation (6.2) yields $P_r(d) = P_r(d_0) \cdot \left(\frac{d_0}{d}\right)^2$ for the received power in free space for $d \geq d_0$, see also [Rap96]. More generally, $P_r(d) = P_r(d_0) \cdot \left(\frac{d_0}{d}\right)^n$ for $d \geq d_0$ with path loss exponent n, see [Gar10].

The free space path loss model is simple and requires only little computational effort. In reality, however, the path loss of a radio transmission is generally more complex since the abstracted model conditions usually do not apply. Often, there are obstacles on the direct path between transmitter and receiver as well as in the surrounding area. These can cause, for example, shadowing, reflection, refraction, scattering, or diffraction of the radio signal, which affect the received signal strength and the path(s) of the radio waves.

In the following, in addition to the free space path loss, which has already been discussed, *shadowing* will be considered in more detail as a second factor affecting the received power of a transmitted signal. Shadowing of a transmitted radio signal is caused by obstacles on the path between transmitter and receiver. In this case, radio waves cannot reach the receiver on a direct path, i.e. via line-of-sight, but only on an indirect path. The effect of shadowing is difficult to describe analytically. Therefore, empirical models which use a combination of analytical calculations and empirical techniques were developed for path loss estimation when shadowing is considered. The second radio propagation model which is implemented in the simulator and introduced here is an example of a path loss model which takes shadowing into account. It is called the *log-normal shadowing model* and is described in the following. The average path loss over all possible path loss values at a considered distance d with $d \geq d_0$ including shadowing effects is given by:

$$\overline{PL}(d)\,[dB] = PL_{FS}(d_0) + 10 \cdot n \cdot \log_{10}\left(\frac{d}{d_0}\right), \tag{6.5}$$

see [Rap96]. Equation (6.5) uses the free space path loss $PL_{FS}(d_0)$ determined via Equation (6.3) at the reference distance d_0 and the path loss exponent n. Instead of using Equation (6.3), $PL_{FS}(d_0)$ can also be determined by empirical measurements. The radio propagation model which only considers the average path loss from Equation (6.5) is called the *log-distance path loss model*, see for example [Rap96].

In empirical studies, for example by the authors of [CMN84; Ber87], it was observed that the actual path loss $PL_{shadow}(d)$ at a distance d including shadowing effects can be considered as a random variable, where the logarithmic representation of this random variable is normally distributed about the mean value $\overline{PL}(d)$ from Equation (6.5). The corresponding distribution is, therefore, called *log-normal distribution* and the corresponding radio propagation model is called the *log-normal shadowing model*. The difference between the actual path loss $PL_{shadow}(d)$ and the average path loss $\overline{PL}(d)$ at a distance d is a zero-mean log-normally distributed random variable χ_σ. It is due to random shadow effects caused by obstacles on the direct path between transmitter and receiver. The actual path loss $PL_{shadow}(d)$ in decibel representation can, therefore, be determined for $d \geq d_0$ via the following formula for the log-normal shadowing model:

$$PL_{shadow}(d)\,[dB] = \overline{PL}(d) + \chi_\sigma = PL_{FS}(d_0) + 10 \cdot n \cdot \log_{10}\left(\frac{d}{d_0}\right) + \chi_\sigma, \tag{6.6}$$

see [Rap96]. Equation (6.6) uses a zero-mean log-normally distributed random variable χ_σ which has a standard deviation of σ. Depending on the specific scenario, n and σ have to be chosen appropriately. The more suitable n and σ are chosen for the considered scenario, the more accurate the model is. Empirical studies have been carried out for this purpose and corresponding suggestions for values of n and σ can be found, for example, in [Rap96]. In Equation (6.6), the reference path loss $PL_{FS}(d_0)$ is the free space path loss calculated via Equation (6.3) at the reference distance d_0, but can also be determined by empirical measurements at distance d_0. Equation (6.6) takes shadowing into account, which generally occurs in reality. Therefore, using Equation (6.6), the received signal strength at a considered location can be estimated more accurately than with Equation (6.5). The estimated received power for the log-normal shadowing model can then be determined via:

$$P_r(d)\,[dBm] = P_t\,[dBm] - PL_{shadow}(d)\,[dB]. \tag{6.7}$$

It can be derived that the received signal strength (in mW representation) decreases with a power of n with distance. Furthermore, it can be seen that two nodes at the same distance d from a considered transmitter do not generally receive the transmitted signal with equal signal strength in this radio propagation model due to the random variable χ_σ in Equation (6.6). Hence, the transmission area in the log-normal shadowing model is irregular instead of circular as in the free space path loss model. This corresponds to a more realistic radio propagation since, without the assumption of a free space scenario, there are usually obstacles on the direct path which can influence signal propagation. In general, the log-normal shadowing model is a widely used radio propagation model, as also mentioned in, for example, [Rap96; Sar+03; CT11; Dez+15]. As stated in [Rap96], in indoor scenarios, path loss follows the log-normal shadowing path loss model described in Equation (6.6). According to [Rap96; ZK07], the model can be applied for indoor and outdoor scenarios.

However, there are many other radio propagation models which estimate the path loss that occurs and the received signal strength at a considered location besides the ones mentioned here. Depending on the specific environment and conditions, very different radio propagation models have been developed. Some models are designed specifically for a particular scenario. A survey of further radio propagation models can be found, for example, in [Rap02; Sar+03; WMB07; Gar10].

In the second transmission scheme implemented in the simulator, the log-normal shadowing model is used to estimate the path loss and, thus, the received signal strength at a given distance. Since it is possible in the simulator to choose a MAC protocol where nodes can transmit simultaneously, a node may receive several transmissions at the same time, each with a possibly different signal strength. For this reason, at a given time, not only the received signal strength of one transmitted signal is taken into account, but the received signal strength of each transmitted signal that reaches a certain minimum threshold $P_{interferenceMin}$ of signal strength. From this information, the signal with the strongest signal strength P_r can be determined. The other remaining signals are considered as interference with signal strength $P_{interference}$. The ratio between the strength P_r of the strongest signal and the interference $P_{interference}$ is called the *signal-to-interference ratio* (SIR), i.e. $SIR = \frac{P_r}{P_{interference}}$. Then, the strongest signal is received if and only if 1) it is at least as strong as the threshold for successful reception, the receiver sensitivity $P_{receiveMin}$, for the entire transmission time and 2) the SIR is at least as large as a certain predetermined threshold SIR_{min}. In decibel representation, the strongest signal has to be stronger than the strength of all other detected signals by at least SIR_{min} since $SIR[dB] = P_r\,[dBm] - P_{interference}\,[dBm]$. It should be noted that signals which would not be successfully received as a separate signal by the considered receiver as they do not reach the threshold $P_{receiveMin}$ can still be considered as interference if they are at least as strong as $P_{interferenceMin}$.

In the simulator, only signals sent by nodes of the considered network are modeled. In reality, however, and thus also in the testbed from Chapter 8, also signals originating from other devices/networks that transmit on the same frequency can be received. These are then treated as interference as well. Furthermore, in the simulator, only the signal strength of the second strongest signal is used for $P_{interference}$ for reasons of abstraction. In summary, a packet transmitted via a signal with signal strength P_r is successfully received if and only if $P_r \geq P_{receiveMin}$ and $SIR \geq SIR_{min}$ hold for the entire duration of the transmission. The fact that received signal strengths can change over time, in particular also during an ongoing transmission, the fact that new signals can arrive, or that ongoing signal receptions may no longer be perceived is taken into account in the approach presented here. In general, SIR should not be confused with the so-called *signal-to-noise ratio* (SNR). SNR exclusively considers noise, such as thermal noise, which is permanently present even without any transmissions instead of considering interference by other signals. It is determined via $SNR = \frac{P_r}{P_{noise}}$, where P_r is the strongest received signal. Often, the *signal-to-interference-and-noise ratio* ($SINR$) is also used, which takes into account both interference and noise and, thus, gives the ratio of the strongest received signal P_r to $P_{noise} + P_{interference}$, i.e. $SINR = \frac{P_r}{P_{noise}+P_{interference}}$, see for example [WMB07]. Since the focus here should be on possible collisions with other existing transmissions, a SIR threshold based signal reception model is considered.

A packet transmission is counted as a collision at a considered receiver if the received signal is strong enough at some point during the transmission to be successfully received, i.e. the received signal strength is above the receiver sensitivity $P_{receiveMin}$, and if one of the following two aspects is true. First, the signal arrives at the receiver as the strongest signal but the corresponding SIR is below SIR_{min} at some point during the transmission time. Second, the signal is not the strongest signal arriving at the receiver. If a transmission is counted as a collision, the considered receiver does not receive the respective packet successfully.

To sum up, the second radio transmission model implemented in the simulator includes log-normal shadowing path loss from Equation (6.6). This means, it uses the log-normal shadowing radio propagation model in order to estimate the received signal strength at a considered location via Equation (6.7). The received signal strength decreases with the distance to the transmitter and an irregular transmission area is caused as random shadowing is considered in addition to the general path loss. Furthermore, interference from other simultaneously transmitted signals is taken into account here by considering the signal-to-interference ratio. The scheme described here provides a more realistic radio propagation than the scheme with circular transmission area described first, but is more complex. Furthermore, certain parameters have to be set appropriately to the concrete addressed environment and scenario. For the choice of the parameter values, it is necessary to specify a concrete scenario. Then, the obtained simulation result only apply to the addressed circumstances. Since the simplified Friis free space equation from Equation (6.2) is used as a basis here, transmitter antenna gain G_t, receiver antenna gain G_r, and system loss factor L are chosen by default as $G_t = G_r = L = 1$. Other parameters, such as transmit power P_t, receiver sensitivity $P_{receiveMin}$, minimum receivable noise/interference power $P_{interferenceMin}$, and minimum signal-to-interference ratio SIR_{min}, have to be chosen according to the specific conditions.

6.1.3 Implemented MAC Protocols

In the simulator, various options of Medium Access Control (MAC) protocols are implemented. A MAC protocol generally controls access to the communication medium. Depending on the number of participants, this is necessary since unregulated access can lead to many collisions and, thus, makes communication more difficult and wastes energy. In general, MAC protocols can be classified as contention based protocols, where collisions can occur and are handled in a predefined way, or schedule based protocols, which are usually collision-free, see for example [KW07; Ana+09; YB09]. In pure contention based MAC protocols, nodes are not synchronized and not scheduled, but usually access the medium randomly. Typical examples of contention based MAC protocols are ALOHA [Abr70] and Carrier Sense Multiple Access (CSMA) [KT75]. In pure ALOHA, a node transmits a generated packet immediately and, then, waits for an acknowledgment of successful reception. If no acknowledgment arrives within a given period of time, the node assumes a collision and chooses a new random point in time to retransmit the packet. In CSMA, a node checks the medium for ongoing transmissions, i.e. it performs *carrier sense*, before sending a packet. It then defers its packet transmission if the channel is busy, or transmits its packet if the channel is available. Variants of both of these contention based MAC protocols are implemented in the simulator and are, thus, described here. A well-known schedule based MAC protocol, which is the basis for another MAC protocol implemented in the simulator, is Time Division Multiple Access (TDMA). For TDMA, time is divided into slots of equal length and each node is assigned a time slot in which it can transmit its packet without causing a collision. As described in Section 3.3.6, energy efficiency is relevant in large-scale WSNs or IoT settings. A widely used approach to save energy is to employ *duty cycling*, for example in the used MAC protocol. This means, the radio module is switched on and off alternately in order to reduce idle listening and, thus, save energy. However, due to the assumptions and goals in Section 3.1, duty cycling is not integrated in the default settings of the implemented MAC protocols, as also reasoned in Section 6.1.7. Examples of energy efficient MAC protocols which use duty cycling are TRAMA [ROG03], T-MAC [VL03], X-MAC [Bue+06], or GoMacH [ZS17]. Of these mentioned MAC protocols, TRAMA is schedule based, all others are contention based.

In the following, the contention based MAC protocols implemented in the simulator, in which nodes access the medium randomly and collisions can be caused, are presented first. Then, the implemented schedule based MAC protocol is presented, in which nodes are synchronized and no signal collisions caused by simultaneous transmissions occur.

6.1.3.1 CSMA/CA Based MAC Protocol According to IEEE Std 802.15.4

A first implemented contention based MAC protocol where collisions can occur is based on the principle of Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) as presented in the IEEE Std 802.15.4. Here, the unslotted variant of the CSMA/CA algorithm included in the standard is used instead of the slotted option in order to avoid the central control required for synchronization and to avoid an additional overhead of further coordination, such as using beacons.

The unslotted CSMA/CA algorithm from IEEE Std 802.15.4 [IEE16b] is executed by each node individually as follows. Initially, a delay (backoff) with a randomly chosen length from the interval $\left[0; 2^{BE}-1\right]BP$ is executed for desynchronization of the nodes. Therefore, systematic collisions are avoided. Here, BE is the so-called *backoff exponent* and affects the maximum possible length of a current backoff. It is initialized with the default value *macMinBE*. Furthermore, BP denotes the length of a so-called *backoff period*. A backoff period corresponds to the time needed to transmit 20 symbols, where one symbol consists of 4 bits. At a data rate of 250 kbit/s, the transmission of one symbol takes 16 µs. Thus, a backoff period BP takes $20 \cdot 16 = 320$ µs. After the initial backoff for desynchronization, a so-called *Clear Channel Assessment* (CCA) is performed, which checks the shared transmission medium for ongoing transmissions. Accordingly, the result is that the channel is perceived as available, i.e. idle, or busy. If the medium is identified as available, the considered node changes to transmit mode and, then, starts transmitting its current packet. If the medium is found to be busy, the so-called *number of backoffs* NB is increased by 1. The number of backoffs NB indicates how many backoffs have already been performed for the current packet to be sent. It is initiated with 0 and must not exceed a given maximum value *macMaxCSMAbackoffs*. If the maximum value *macMaxCSMAbackoffs* would be exceeded, the current packet is discarded, i.e. the attempt to transmit is canceled and the transmission process is skipped. If the channel is identified as busy and $NB \leq$ *macMaxCSMAbackoffs*, the backoff exponent BE is also increased by 1. Here, BE is upper bounded by the default value *macMaxBE*, which means that BE is not increased anymore if the value *macMaxBE* is reached. Subsequently, a random backoff from the interval $\left[0; 2^{BE}-1\right]BP$ is waited again. This is followed by another CCA, etc. The procedure described here for choosing the backoff length is also called *binary exponential backoff* (BEB) scheme. By increasing BE, the interval

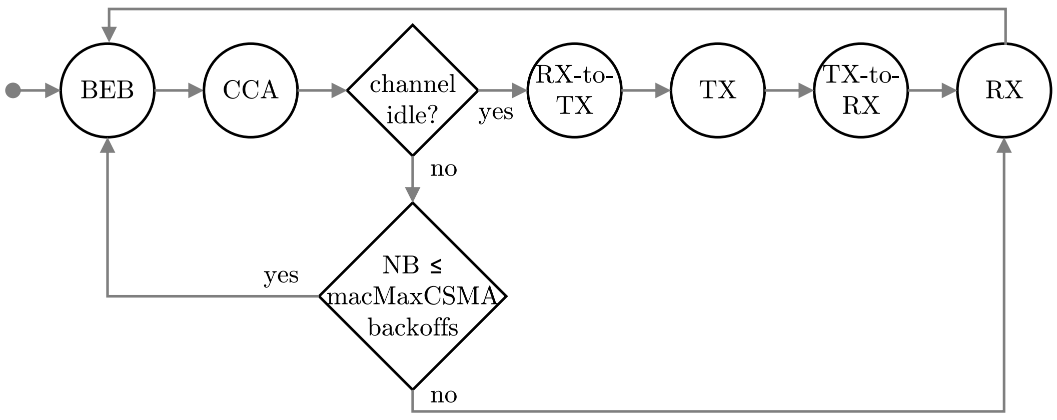


Fig. 6.1: State diagram of implemented CSMA/CA based MAC protocol in default setting

from which the delay length is randomly chosen increases. In this way, it is tried to avoid finding the channel busy again during CCA. Hereby, the access to the medium/the transmission behavior is adapted to the network load without having to use a central control. By performing CCA and using backoffs if necessary, the strategy of collision avoidance is pursued. Energy is saved by avoiding collisions and, thus, unnecessary transmissions. More details about the implemented CSMA/CA algorithm can be found in IEEE Std 802.15.4 [IEE16b].

In the simulator, the CSMA/CA based MAC protocol is implemented as a final state machine with the states BEB, CCA, RX-to-TX, TX, TX-to-RX, and RX. An optional $SLEEP$ state can also be enabled here. The MAC protocol is realized for each node independently from the other nodes as depicted in Figure 6.1. In BEB state, a binary exponential backoff is performed according to IEEE Std 802.15.4, as described earlier. In the next state, CCA, the medium is checked for ongoing transmissions. In case the channel is detected as busy, the node switches again to BEB state if the maximum number *macMaxCSMAbackoffs* of backoffs has not been exceeded yet. If *macMaxCSMAbackoffs* has been exceeded, the node skips sending and switches directly to receive mode or to sleep mode if the optional $SLEEP$ state is included. In case the medium is identified as available during CCA, the node switches to RX-to-TX state. The state RX-to-TX corresponds to the transceiver's turnaround from receive to transmit. In this state, a node also creates the next packet to be sent because, in this way, the most current information can be used. Afterwards, the node is in the transmit mode, i.e. in the state TX. In this state, the node sends its packet via a single broadcast to all of its neighbors without waiting for an acknowledgment since no bidirectional communication is assumed in Section 3.1. The duration t_{TX} of the state TX corresponds to the duration of the transmission of the respective packet. This is followed by an optional $SLEEP$ state, in which the transceiver is switched off for a specified time t_{SLEEP}. In the subsequent $TX\text{-}to\text{-}RX$ state, the node switches from transmitting to receiving. Then, in RX state, the considered node is in receive mode, i.e. it listens to the medium for incoming packets and receives arriving packets. The node stays in RX state for a given time t_{RX}. If there is an ongoing collision-free transmission to the node when t_{RX} ends, the state is only left as soon as this transmission is finished.

As already mentioned at the beginning of Section 6.1.3, there is no duty cycling in the default setting of the described unslotted CSMA/CA based MAC protocol. This means, the energy-saving $SLEEP$ state is not used here, which is reasoned in more detail in Section 6.1.7.

The default values for *macMinBE*, *macMaxBE*, and *macMaxCSMAbackoffs* in IEEE Std 802.15.4 [IEE16b] are 3, 5, and 4, respectively. Thus, in the case of the default value $BE = macMinBE = 3$, the interval $\left[0; 2^{BE} - 1\right] BP$ starts at a length of $7 \cdot 20 \cdot 16\,\mu s = 2240\,\mu s$. With $BE = macMaxBE = 5$, the interval is $31 \cdot 20 \cdot 16\,\mu s = 9920\,\mu s$. According to IEEE Std 802.15.4 [IEE16b], the time required to perform CCA detection is 8 symbol periods, which means $8 \cdot 16\,\mu s = 128\,\mu s$. Furthermore, RX-to-TX or TX-to-RX turnaround time is up to 12 symbol periods in IEEE Std 802.15.4 [IEE16b], i.e. $12 \cdot 16\,\mu s = 192\,\mu s$. In addition, the default value of t_{RX} is assumed to be 20 ms here. In our evaluations of the time in which a node is in RX state, this value provided the best results in simulations of BCGC, as shown in the thesis [Wis18] for payloads relevant for us. This thesis was conducted as part of this work.

There are several research papers which analyze the influence of the parameters *macMinBE*, *macMaxBE*, and *macMaxCSMAbackoffs* [RG09] or adapt *macMinBE* to the state of the network [KCK06]. Others optimize the presented backoff strategy in general, which includes that they increase/reduce BE differently than suggested in IEEE Std 802.15.4 [Ha+07] or choose a different backoff interval [Ha+07; Lee+14; KGM11]. In [Wis18], the BCGC algorithm is executed using the CSMA/CA based MAC protocol as presented here. There, the default values of IEEE Std 802.15.4 are used for *macMinBE*, *macMaxBE*, and *macMaxCSMAbackoffs* and the results are compared with the results obtained with modified values of these parameters. Moreover, CSMA/CA based MAC protocols with optimized backoff strategy similar to those presented in [Ha+07] and in [Lee+14] are also used for comparison in [Wis18]. Since all results do not differ significantly from those obtained with the default values, the default values from the IEEE Std 802.15.4 are used for *macMinBE*, *macMaxBE*, and *macMaxCSMAbackoffs* here, as suggested in [Wis18].

Despite performing CCA, however, packet collisions are possible with CSMA/CA. Reasons for this can be that, first, one node is performing CCA while another node is also in CCA state or performing a turnaround but has not yet started transmission. Thus, both nodes have observed a clear channel and their transmission process eventually overlaps which causes packet collisions. Secondly, the so-called *hidden-node problem* can occur. There, the channel for one node appears to be available but a neighboring node is receiving a packet from another node that is out of range of the first node. If the first node then starts its transmission, the two packets will collide at the neighboring node. Thirdly, it is possible that nodes move away during a transmission so that the received signal becomes too weak and the packet cannot be received completely. It is also possible that other receiving/transmitting nodes occur or move into an ongoing transmission so that they cannot receive the complete packet.

6.1.3.2 ALOHA Based MAC Protocol with Random Transmission and no Carrier Sense

Besides contention based MAC protocols which check the medium for ongoing transmissions, there is also the option of not using carrier sense with contention based protocols. In the next implemented variant of a contention based MAC protocol, each node chooses a random start time for its next transmission and starts transmitting when the respective point in time is reached regardless of the state of the medium. As a consequence, there are significantly more collisions than in comparable approaches with carrier sense. This described MAC protocol is similar to the pure ALOHA protocol and is, therefore, referred to as ALOHA based here.

6.1.3.3 TDMA Based MAC Protocol

Another implemented MAC protocol is Time Division Multiple Access (TDMA) based. In contrast to the two implemented MAC protocols described above, it is not contention based but schedule based. Here, time is divided into slots of equal length. Each node is assigned a time slot in which it can transmit without causing collisions. In the simulator, time is divided into rounds where a round is defined as the time needed for all nodes to transmit one after the other. Each node then is assigned a fixed transmission slot within each round. Nodes are synchronized and, therefore, no signal collisions occur. Thus, the MAC protocol described here is also referred to as a round based/slotted, collision-free MAC protocol when contrasted with the other implemented random access, collision-based MAC protocols. The property of the TDMA based protocol to be collision-free refers only to signal collisions. Collisions caused by the fact that a node is not within the transmission range during the whole transmission or by the fact that a node is not in receive mode can still occur.

6.1.4 Implemented Options for the Initial Topology

Before starting a simulation run, an initial position on the simulation area has to be chosen for each node one after the other. It is assumed that no two nodes can choose the same position. In the case of a static network, each node remains at the chosen initial position during the entire simulation run. In a mobile network, depending on the selected mobility model and speed, nodes can quickly move far away from the chosen starting position so that the initial topology has only little influence on the results. Here, the term ‘topology’ refers to the physical arrangement of the nodes on the simulation area and does not specify which nodes can actually communicate with each other. The communication topology has already been set in Section 3.1 as a mesh topology, which means that each node can transmit data to all nodes within its transmission range, and is not discussed in this section. Generally, there are numerous options for the initial topology concerning physical arrangement. There can be spatial constraints or no spatial constraints except for the fact that nodes have to be placed on the simulation area.

6.1.4.1 Topology with Spatial Constraints

A first possible topology with spatial constraints is the so-called *grid topology*. There, nodes are arranged in a regular grid. During the development of BCGC, a grid topology was initially chosen for static scenarios due to its regular structure where each node which is not in the boundary region has the same number of neighbors. In the simulator, it is still possible to use a grid topology, however, for the evaluations in Sections 7.1 and 7.2, a different default initial topology is chosen for static networks.

Another deterministic topology with spatial constraints is the *line topology*. In a line topology, nodes are arranged along a line, for example at a given equidistant distance. An extension of this topology is the arrangement of nodes along certain given paths, such as the border of a rectangle. This type of topology is suitable, for example, to model corridors and rooms of a hospital, which is one of the possible application scenarios of BCGC described in Section 3.1.3. In the simulator, it is used as initial topology for the Pathway mobility model.

Besides deterministic options, a *random-based topology with spatial constraints* can also be chosen as initial topology. For example, nodes could be randomly placed on the simulation area using certain weights based on geographic constraints. There, predefined locations and their surroundings could be chosen as initial position by a node with a certain given probability each and, thus, with higher probability than other locations. These so-called points of interest or cluster center could represent e.g. special buildings, sights/attractions, or rooms in a scenario with people. This type of topology is similar to human behavior and is used as initial topology, for example, in the SMOOTH mobility model, which is introduced later and also implemented in the simulator.

All these variants for the initial topology can be close to reality depending on the scenario, but can also be very application specific so that results of the respective simulations might only be valid for the chosen concrete scenario.

6.1.4.2 Random Topology with no Spatial Constraints

Another random-based topology, however, without spatial constraints, is the so-called *random topology*. There, nodes are randomly placed on the simulation area. It is the simplest random-based topology with no constraints and is, thus, chosen as initial topology for the evaluations in Sections 7.1 and 7.2 for both static and mobile scenarios.

6.1.5 Implemented Mobility Models

According to Section 3.1, it is assumed that nodes are mobile, which means nodes can change their position on the simulation area and the topology changes. However, for neighboring nodes, there is no difference if nodes move away, if nodes switch off, or if all packets collide and do not arrive, for example. Consequently, the mobility model is not the only factor that determines whether packets are received or not. However, in the case of a mobile scenario, the chosen mobility model affects packet reception and the path of the packets through the network in general. Depending on the respective mobility model, the speed of the nodes can also be selected in addition to the motion pattern. Speed is usually chosen uniformly at random from a given interval by each node individually. Therefore, heterogeneous modeling of velocities is possible. In the following, mobility models which are implemented in the simulator are briefly described. A more detailed overview of these and other mobility models can be found, for example, in [CBD02; BH04; AGM08; RS11]. Except for the time-based Random Walk model, the introduced mobility models were mainly implemented in the simulator as part of a master thesis, see [Stö17]. Further implementation details can, therefore, be found in [Stö17]. In general, there are synthetic mobility models and there are trace-driven models, which are based on real-world traces. Among the synthetic models, according to [BH04], there are random based mobility models, models with temporal dependency, models with spatial dependency, and models with geographic restriction. Using the classification from [BH04], random based models do not have temporal or spatial dependencies and no constraints in general. For models with temporal dependency, the past motion of a node has influence on its current and future motion. Spatial dependency means that the movement of a node depends on its surroundings, i.e. on the movement of other nodes. If the mobility model contains a geographic restriction, nodes are only allowed to move on certain parts of the simulation area. In the simulator, a mobility model is implemented from each of these listed classes. The implemented models are briefly introduced below.

6.1.5.1 Random Waypoint Model

A first mobility model which is implemented in the simulator is the so-called *Random Waypoint mobility model*. This model is commonly used by related work due to its simplicity. According to [BH04], it can be assigned to random based mobility models. The original model is from the authors of [JM96]. In the Random Waypoint model, each node first waits a random period of time. Then it chooses a random destination position on the simulation area and a random speed from a given interval and moves at that speed from its current position to its chosen destination. After that, the node waits again for a random period of time, chooses a new random destination and speed, and so on. Due to its design, in the course of a simulation run, an accumulation of nodes emerges in the middle part of the simulation area and only few nodes are in the boundary area, see [BRS03; BRS04]. This changed density in certain areas was observed in executed simulations and influenced the system latency negatively due to the single poorly connected nodes in the boundary area. In addition, the Random Waypoint mobility model suffers from the so-called speed decay problem. This problem of the Random Waypoint model is described and analyzed in detail, for example, in [YLN03]. Nodes with distant destinations and slow speeds become more frequent over time. As a consequence, the average speed of the nodes decreases as the simulation run progresses, which is denoted as speed decay problem. Therefore, the Random Waypoint mobility model would further worsen the results of simulations with settings/procedures that cause a higher system latency and would distort the overall comparison of individual settings/procedures. It is, thus, not suitable for the evaluations here in Sections 7.1 and 7.2.

6.1.5.2 Time-Based Random Walk Model

Another frequently used mobility model is the so-called *time-based Random Walk model*. The time-based Random Walk model is a random based mobility model according to [BH04]. Each node chooses a random direction and a random speed from a given interval and maintains both for a predetermined period of time. Then, the node again chooses a random direction and a random speed, etc. The lower the maximum speed, i.e. the upper interval limit for the speed, or the shorter the predetermined period of time is set, the more the nodes move only locally. Depending on the application scenario, however, this can be close to reality if nodes move less frequently across the entire simulation area than they stay in a locally limited space. Furthermore, if a node reaches the boundary, a so-called bouncing rule is applied. This rule can specify, for example, that a node is reflected at the boundary or that a node disappears at the boundary and reappears in the middle or on the other side of the simulation area. To avoid strong accumulations in the center of the simulation area and to model a realistic behavior, it is assumed that nodes are reflected at the boundary here. Over time, this can lead to accumulations at the boundary, which, however, were not observed in the executed simulations in Sections 7.1 and 7.2. A single simulation run was always completed before accumulations became apparent. A disadvantage of the time-based Random Walk mobility model is that this model is memoryless so that movements are executed independently from the previous movements. As a consequence, a strong change of direction or speed can occur repeatedly. Regarding human motion, this type of movement may seem unnatural, but for machines, for example, this mobility pattern may not be untypical. Advantages of the Random Walk model are that it is simple to implement and that no complex calculations have to be performed. Furthermore, no information about previous movements has to be stored to determine a next movement and there are no further constraints. Moreover, by using the Random Walk mobility model, nodes are distributed uniformly at random across the simulation area at any later point in time within a simulation run, see [BRS04]. Thus, the mobility model is comparatively easy to analyze. These listed benefits also apply to the Random Waypoint model. Due to their simplicity, the Random Walk and the Random Waypoint mobility model are often used by related work to model motion in a simplified way, see [BH04]. However, an important advantage of the Random Walk model and the reason why it is preferred to the Random Waypoint model here is that there is no speed decay problem with Random Walk. Instead, the average speed of the nodes remains at a constant level throughout a simulation run. In general, there is a tradeoff between realistic mobility pattern and simplicity. As a compromise between these two opposing properties and due to the other characteristics mentioned here, the time-based Random Walk mobility model is used for the evaluations in Sections 7.1 and 7.2.

6.1.5.3 Gauss-Markov Mobility Model

Unlike the two mobility models described before, the so-called *Gauss-Markov mobility model* presented next does not have abrupt changes in direction or speed, but produces smoother movement pattern. This is realized by the previous motion having an influence on the future motion. The Gauss-Markov mobility model can, therefore, be assigned to the class of models with temporal dependency. It was developed in [LH99; Tol99; LH03]. In the Gauss-Markov mobility model, a node chooses a direction and a speed and maintains them for a given period of time. It then computes a new direction and a new speed using the previous direction/speed, a constant desired average value for direction/speed, and a random variable from a Gaussian distribution. Furthermore, these three values from the calculation of the next direction/speed are each weighted with a factor, which has to be chosen depending on the application scenario. This factor allows to obtain motion patterns ranging from a totally random motion to always the exact same motion as before and, thus, a linear motion over the simulation surface. The factor determines how strongly the next motion depends on the previous one and reflects the degree of randomness in the movement. With the Gauss-Markov model, a more natural human motion is modeled than with the Random Walk or Random Waypoint mobility model. However, more complex calculations are necessary and the desired average values and the factor have to be chosen appropriately. This would require further analysis to obtain a motion pattern that fits the application scenario quite well. In simulations performed with the Gauss-Markov mobility model in [Stö17], even with an intermediate degree of dependence on the previous motion, similar results were obtained for BCGC as with the Random Walk model when using the same speed and node density in both models. Therefore, the simpler Random Walk mobility model, which contains fewer parameters, is used for the evaluation in Sections 7.1 and 7.2.

6.1.5.4 Pathway Mobility Model

Another implemented mobility model is the so-called *Pathway mobility model*, which originates in [Tia+02]. It belongs to the class of models with geographic restriction, see [BH04], as nodes are only allowed to move on previously defined paths. This mobility model was implemented for the simulator as it particularly fits the application scenario of a hospital with its predefined corridors and rooms, which is described in Section 3.1. In the Pathway mobility model, a graph with vertices and edges is given according to the addressed application scenario. At initialization, each node chooses a random vertex of the graph as its starting position. Then, each node chooses another random vertex as its destination vertex and a random speed from a given interval. Subsequently, each node moves from its current position at its chosen speed over a sequence of edges of the graph to its destination vertex. This sequence of edges should represent the shortest path between the current position and the destination and is, therefore, calculated using Dijkstra's algorithm [Dij59]. Nodes move only on edges of the graph. When the destination is reached, the considered node pauses for a random period of time and then chooses a new random destination vertex and a new random speed, etc. In the implementation, it was ensured that the node density in the Pathway mobility model does not deviate significantly from the node density observed in the other implemented mobility models in order to enable a fair comparison with the Random Walk mobility model. To model the application scenario of a hospital, a cyclic graph in the form of a rectangle was chosen. From this rectangle, at given vertices, edges go perpendicularly inwards and outwards with vertices at their ends representing the rooms. Edges should correspond to corridors and entrances to the rooms. In executed simulations in [Stö17], similar results for BCGC were obtained with this graph using the Pathway mobility model as with Random Walk. For this comparison, the same speed and node density were used in both models. For the Pathway mobility model, the underlying graph has to be designed according to the considered scenario and, thus, causes a realistic movement pattern for this specific scenario. However, the results of the simulation may only be valid for the specific chosen graph. For this reason and in order to avoid having to specify a concrete graph, the Random Walk model is preferred to the Pathway mobility model for the evaluations in Sections 7.1 and 7.2.

6.1.5.5 SMOOTH Mobility Model

The also implemented *SMOOTH mobility model* has been developed by the authors of [MCN11] and is a variant of the SLAW mobility model of [Lee+09]. It can be assigned to the class of models with temporal dependency, models with spatial dependency, or trace driven models. Nodes visit places they have visited before more often than places they have never been to (model with temporal dependency). Places which are visited by many other nodes are preferred by a node (model with spatial dependency). Furthermore, the SMOOTH mobility model was developed on the basis of findings from real-world traces (trace driven model). According to [Lee+09], the characteristics of human movement derived from traces include the properties that more short than long distances are covered, breaks are rather short than long, and contact times between people are also rather short. Moreover, in [Lee+09], it is deduced that the length of these distances/times follows a truncated power law distribution. In addition, people tend to move within an individually limited area and move more often to locations favored by many others. At initialization, a given number of cluster centers are randomly placed on the simulation area. Afterwards, each cluster center is assigned a certain weight, which influences the number of nodes that will later choose their starting position in this cluster. The probability distribution of the weights should be a truncated power law distribution so that there are few large and many small clusters. Then, the nodes randomly choose a cluster center, considering the individual cluster weights, and are placed at a random location in the surroundings of their chosen cluster center. After initialization, each node has to choose a random destination point on the simulation area. For this purpose, a node decides whether to choose a place it has already visited or a new place. However, the probability to choose a new place is the lower the more places the node has already visited before. If the node decides to visit a new place, it is more likely that it chooses a nearby place than a more distant one. Again, the probability distribution follows a truncated power law. If the node decides to visit a place it has already visited, it chooses among all of these places a certain place with a probability proportional to the number of times it has visited this place. The speed with which the node then moves to the chosen destination is constant. It is determined based on a truncated power law distribution, depending on the distance of the chosen destination from the starting position of this movement. To a more distant destination, a higher speed is chosen than to a closer destination. If a destination is reached, the node waits for a random period of time, whereby again, following a truncated power law distribution, it is more likely to wait for a short period of time. Afterwards, the node chooses a new destination again, etc. In this way, many short and few long movements result. In order to obtain similar results for BCGC in the SMOOTH mobility model as in the Random Walk model, the average number of neighboring nodes in all parts of the simulation area was generated as equally as possible in comparative simulations in [Stö17]. For this purpose, a particularly large number of clusters of the same size were generated. Nevertheless, the node density did not become as uniform as with the Random Walk mobility model, which resulted in a slightly higher latency for BCGC with SMOOTH. However, this artificial generation of a uniform node density does not correspond to the core idea of the SMOOTH mobility model. For this reason, simulations with fewer cluster centers and individual larger clusters were also executed in [Stö17]. There, a significantly higher system latency was observed. The SMOOTH mobility model represents real human movements and uses only characteristics from real-world traces instead of the traces themselves. If traces were to be used directly, they would have to come from the same or a similar scenario that is to be simulated. This would require a sufficiently large amount of data, and simulation parameters could hardly be varied. The SMOOTH mobility model offers much more flexibility and any number of simulations can be run. However, it is very complex to generate a suitable model, i.e. to define cluster centers appropriately, to determine weights adequately, to specify the truncated power law distributions suitably, and to set the many parameters. In addition, the SMOOTH mobility model is very scenario specific as soon as cluster centers etc. are set. The results are then only valid for the addressed scenario since the node density in the individual parts of the simulation area is strongly tailored to the scenario. For these reasons, the SMOOTH mobility model is not used for the evaluations in Sections 7.1 and 7.2.

6.1.6 Implemented Patterns for Node Failures

Nodes in a sensor network or in an IoT environment can fail. The goal for any considered procedure, such as BCGC or RLNC, is to collect all original data despite the assumed form of node failures. In case of node failures, all data should be reconstructed and stored in all still active nodes. Even the symbols generated by the nodes that are no longer operating should not be lost. This means, all symbols generated in the network should be collected by each node which is still operating. However, if the node failure occurs very early in the dissemination process, some symbols may no longer be present in the network. In this case, those symbols that still exist in the system should be reconstructed in all remaining nodes. In general, node failure can be temporary or permanent. Temporary node failures and link failures caused by e.g. packet collisions or nodes running out of the transmission range of another node during a transmission hardly differ. Since link failures are already taken into account in the simulator, temporary node failures are, thus, neglected and only permanent node failures are additionally considered here. In the simulator, it is possible to run simulations without node failures or to choose between different types of permanent node failures. Depending on the concrete application scenario, different patterns of node failures are realistic. For the evaluations here, it is not crucial to have as realistic node failures as possible. Instead, rather drastic node failures should be simulated in order to observe to what extent a considered procedure can cope with these.

6.1.6.1 Random Node Failures

The simplest form of node failures which is implemented in the simulator are the so-called *random node failures*. There, half of the nodes fail randomly at a given point in time. The proportion of nodes that fail could also be varied. However, a node failure of 50% is already a quite extreme scenario. Furthermore, nodes fail independently of their position on the simulation area and independently of the behavior of other nodes. This type of node failures reduces the node density on the entire simulation area and each node has fewer neighboring nodes on average than before.

6.1.6.2 Exponentially Distributed Node Failures

Another implemented form of node failures are *exponentially distributed node failures*. In this variant, nodes within a circular region around the center of the simulation area fail at a given point in time. The radius of this circular area can be varied so that a correspondingly modified number of node failures is caused. However, in the model with exponentially distributed node failures, it is not the case that 100% of the nodes within a circular area around the center fail and no node fails outside this area. Instead, a so-called critical distance to the center of failure can be chosen up to which all nodes should fail. Beyond this range, the probability that a considered node fails decreases exponentially with increasing distance from the center. In addition, there is a tuning parameter with which node failure can be extended to a certain area or even to the boundary of the simulation area, still with fewer and fewer nodes failing as the distance from the center increases. To conclude, node failure can affect an arbitrarily large area previously chosen without causing all nodes to fail in that area. Especially in a static scenario, many nodes in a substantial part of the simulation area fail and are not replaced. Thus, for the remainder of the respective simulation run, there are fewer possible paths between nodes located on different sides of the failure region than before nodes failed. The number of remaining paths depends on the length of the radius of the circular area and on the critical distance. In a dynamic scenario, over time, the failure region is reoccupied by nodes that are still active. Thus, at some point in time, nodes are again located on the entire simulation area, but the network then has reduced density.

6.1.6.3 Sequential Node Failure

In the previously considered patterns of node failures, failures only occur at a single fixed point in time during a simulation run. In the case of so-called *sequential node failure*, which is also implemented in the simulator, one node after the other fails. Nodes fail in regular time intervals until the simulation run is finished or until there are no active nodes left. This means, there are gradually fewer and fewer active nodes and the node density on the simulation area decreases continuously. However, during the

State	Supply current	Power consumption
TX (at $P_t = 0\,\text{dBm}$)	11.6 mA	$11.6\,\text{mA} \cdot 3.0\,\text{V} = 34.8\,\text{mW}$
RX, CCA	12.3 mA	$12.3\,\text{mA} \cdot 3.0\,\text{V} = 36.9\,\text{mW}$
BEB	0.4 mA	$0.4\,\text{mA} \cdot 3.0\,\text{V} = 1.2\,\text{mW}$
$TX\text{-}to\text{-}RX, RX\text{-}to\text{-}TX$	0.4 mA	$0.4\,\text{mA} \cdot 3.0\,\text{V} = 1.2\,\text{mW}$
$SLEEP$	0.02 µA	$0.02\,\mu\text{A} \cdot 3.0\,\text{V} = 0.000\,06\,\text{mW}$

Table 6.1: Current consumption/power consumption specifications in respective state

execution of procedures with higher latency, more node failures will eventually occur than when using procedures with lower latency.

6.1.7 Energy Consumption Modeling

Energy is consumed in sensing, in communication, and also in data processing, for example, see [Gar10]. As already explained in detail in Section 3.3.6, the focus here shall, however, be on communication. The specified energy consumption, therefore, refers to the communication energy consumption of the nodes' radio transceivers. For this purpose, the different states of a transceiver are considered. A description of the used states and transitions can be found in Section 6.1.3. To derive the energy consumption in a considered simulation run, the total time spent in each state is first multiplied by the power consumption in the respective state. Then, the resulting energy consumption in the single states is summed up for the total energy consumption. The used approximated values for supply current/power consumption in transmit state TX, receive state RX, backoff state BEB, CCA state, turnaround state $TX\text{-}to\text{-}RX$ or $RX\text{-}to\text{-}TX$, and $SLEEP$ state can be found in Table 6.1. Here, a supply voltage of $V_{DD} = 3.0\,\text{V}$ is assumed. The values for supply current and supply voltage are taken from the data sheet of the AT86RF231 transceiver, see [Atm09], which is also used in the testbed in Chapter 8. They can, however, be easily replaced by the values of a different transceiver. It should be noted that the energy consumption in transmit state TX depends on the used transmit power P_t. According to Section 6.2, P_t is set to 0 dBm by default here.

A commonly used approach to save energy is to employ *duty cycling*, which is also included as an option in IEEE Std 802.15.4. There, nodes are repeatedly set to an energy-saving $SLEEP$ state. The so-called *duty cycle* is the active part of a node, which should be kept as short as possible to save energy. At the same time, however, latency should not increase unreasonably. Duty cycling can be beneficial if there are long periods without transmissions so that nodes would perform idle listening for a long time, which consumes a similar amount of energy as if they were actually receiving data. However, if there are a lot of transmissions at any time and each sink should quickly receive all desired data, as is the case here due to the assumptions and goals in Section 3.1, nodes should not enter a $SLEEP$ state. Otherwise, they would miss many useful transmissions so that only the latency would increase and, thus, hardly any energy could be saved. If multiple generations are considered and a node knows that it and its neighbors have already received all desired data of the current generation, the node could enter a $SLEEP$ state until the next generation, etc. However, considering multiple generations is beyond the scope and will be left for future work. Therefore, a $SLEEP$ state is not used in the default setting here, and there is no duty cycling. More details on duty cycling in WSNs are omitted here but can be found, for example, in [Ana+09; Car+13; Wan+17].

6.1.8 Parameters Relating to Simulation Settings

This section lists parameters which are relevant for the execution of the simulation and also summarizes the previous subsections. The first parameter relating to simulation settings is the number of nodes which make up the network. Another essential simulation setting is the size of the payload which has to be transmitted in a data packet. Here, only the specifications of IEEE Std 802.15.4 concerning packet sizes, see Section 3.2.2, have to be met. Furthermore, the radio transmission scheme has to be chosen. It specifies the general shape of the transmission area, determines the received signal strength at a considered location, and decides whether a packet will be received successfully in case of interference due to other simultaneously arriving signals. The transmission range of a node is not specified directly but results indirectly from the transmission scheme, the values set there, and the surroundings of the considered node. For more details on implemented radio transmission schemes, see Section 6.1.2. On top of that, a suitable MAC protocol has to be selected for wireless data communication in the simulation. There are schedule based MAC protocols, which are usually collision-free, and contention based MAC protocols, where collisions can occur. In the case of a contention based MAC protocol, it is possible to use carrier sense and check the medium for ongoing transmissions before transmitting a packet, or to omit carrier sense. All these options are implemented in the simulator and are described in more detail in Section 6.1.3. Another parameter relating to simulation settings is the size of the simulation area. Together with the number of nodes and the transmission range of a node, it determines the general node density in the network. Since the transmission range results from given circumstances and conditions, the size of the simulation area or the number of nodes should be varied in order to change the node density. Moreover, the initial topology of the network, i.e. the initial position of the nodes on the simulation area, must be decided on. The term initial topology should, however, not be confused with the communication topology, which is specified here as mesh topology according to the system setup in Section 5.6. A description of the implemented options for the initial topology of the network can be found in Section 6.1.4. In the case of a mobile network, different mobility models are possible. These models decide on the movement pattern of the individual nodes. Some of the mobility models already specify an initial topology. More details on the implemented mobility models can be found in Section 6.1.5. In addition to the mobility model, a range for the possible speed of the individual nodes has to be chosen if the considered network is assumed to be mobile. Another parameter concerning simulation settings is the decision if node failures are considered separately or as an inherent part of topology changes/packet collisions. In general, node failures can be transient or permanent. They can occur at some time or regularly, and at random places or in contiguous areas, for example. For details on implemented pattern for node failures, see Section 6.1.6.

6.2 Default Configuration of Parameters

In this section, the default configuration of BCGC process parameters and of parameters relating to simulation settings will be listed and explained. Concerning subsequent simulations, we have set the guidelines that all but one parameter will be chosen according to the default configuration. The parameter which is evaluated in the respective section will be variable. In this way, the impact of the variable parameter can be investigated. However, the process parameters in particular influence each other. Therefore, it is possible that a different combination of process parameter values provides better results in a considered simulation setting than the optimal values determined individually. Furthermore, the individual parameters relating to simulation settings may have a differently strong influence on the performance of a considered procedure and can also have opposing effects. Thus, no general statement can be made from the individual results for every possible combination of parameter values which relate to simulation settings. Evaluating each reasonable combination of values of BCGC process parameters and values of parameters relating to simulation settings can be considered as a multidimensional optimization problem with constraints. Finding those minima/maxima for a considered performance metric would also be viable but represents a different approach.

6.2.1 Default Configuration of BCGC Process Parameters

For the evaluation of BCGC process parameters in Section 7.1, the default simulation settings as described in Section 6.2.2 are used. Depending on the chosen settings of the simulation, different values may be suitable as default values for the process parameters and also different values of the process parameters may provide the best results concerning performance metrics. The default configuration of relevant BCGC parameters which will be used for subsequent evaluations, is listed in the following. More details on the parameters themselves can be found in Sections 5.2 and 5.3.

One of the two main categories of BCGC process parameters is the degree of a codeword for transmission, which is determined using a degree function. For the evaluation of the degree function in Section 7.1.1, the Growth Codes degree function is used as a basis. Then, the best performing considered degree function is used for the remaining evaluations. For the evaluation of the degree function in Section 7.1.1, no limitation of the degree is used as a basis to be able to see the entire effects of a respective degree function without cutting it off early and possibly distorting the results. However, for further evaluations, the maximum admissible upper bound for the codeword degree, $maxDeg = 46$, is used with which the packet size restrictions by IEEE Std 802.15.4 can still be met with the selected default payload size, see Section 6.2.2, if not stated otherwise.

In general, packet sizes can be static or be dynamic, which means they can change in the course of a simulation run. Here, dynamic packet sizes are chosen as default configuration as a contention based MAC protocol without the need of synchronization is used as default MAC protocol. If synchronization does not have to be ensured and static packet sizes are not required due to a certain application, dynamic packet sizes should be preferred. They produce smaller packets, which means shorter transmission times and, thus, better results concerning performance.

Besides the degree of a codeword, the other major BCGC parameter is the selection of the symbols which will be included in a next codeword for transmission. An unfavorable specific symbol selection may distort the evaluations. Furthermore, the stated formulas in Section 5.2 are only valid if symbols are chosen randomly. Hence, for the previously mentioned evaluations, the default setting is to choose symbols randomly from the symbols already reconstructed in the considered node. For the remaining evaluations in Section 7.2, however, the specific symbol composition of codewords is used, which is found to be the best performing variant in Section 7.1.2. There, the first symbol is specifically selected based on certain counters in order to be as suitable as possible for the neighboring nodes, and only the other symbols are chosen randomly. The use of a desired symbol is investigated in Section 7.1.2.2 but not applied in further evaluations due to the not noticeable improvement by this approach.

6.2.2 Default Simulation Settings

In the following, the default configuration of parameters relating to simulation settings is described. Concerning the number of nodes n, the default value is set to $n = 256$ since a large-scale, however realistic, network should be modeled here. On top of that, it is assumed that all of the nodes are sinks as default configuration, so there are 256 sinks, which aim to gather all original symbols eventually. Therefore, the choice of the position of a sink does not influence the results. Furthermore, a single point of failure is avoided as there is not only one sink in the system. With regard to payload size, 20 Bytes of payload is assumed as default value here. This also corresponds to the reference payload size used for calculations in IEEE Std 802.15.4, see [IEE06], and is within the typical payload range for IEEE Std 802.15.4 applications. Concerning the radio transmission scheme, the more realistic radio propagation model based on log-normal shadowing path loss with consideration of the signal-to-interference ratio is used as default setting. It causes an irregular transmission area. More details on this used radio transmission scheme can be found in Section 6.1.2.2. The parameters transmit power P_t, receiver sensitivity $P_{receiveMin}$, minimum receivable noise/interference power $P_{interferenceMin}$, minimum signal-to-interference ratio SIR_{min}, path loss exponent n, and standard deviation σ, which have to be specified for this model, are chosen as follows: In the simulator, a default transmit power of $P_t = 0\,\text{dBm} = 1\,\text{mW}$ is used for the evaluations in Sections 7.1 and 7.2, which is within the typical range for the transmit power of IEEE Std 802.15.4 devices, see [IEE16b]. Concerning receiver sensitivity, $P_{receiveMin} = -85\,\text{dBm}$ is chosen as default value here. According to IEEE Std 802.15.4 in [IEE16b], a device which is compliant with the standard has to have a sensitivity of $-85\,\text{dBm}$ or better. Furthermore, a default value of $P_{interferenceMin} = -90\,\text{dBm}$ is assumed for the threshold $P_{interferenceMin}$ for the evaluations in Sections 7.1 and 7.2. For the minimum signal-to-interference ratio SIR_{min}, the value $SIR_{min} = 5\,\text{dB}$ is used as default value here. According to [Pet+06], IEEE Std 802.15.4 compliant devices using the 2.4 GHz PHY with O-QPSK have a bit error rate of less than 10^{-8} in the case of $SIR = 5\,\text{dB}$. The path loss exponent $n = 4.5$ is chosen as default value here, which is within the range specified in [Rap96] for the case ‘obstructed in building’ and provides an average maximum transmission range of about 10 m. A different choice of n may result in a significantly different transmission range, so n should always be chosen appropriately for the considered scenario. The maximum transmission range can be calculated from $P_r(d) - P_{receiveMin} = 0$ with Equation (6.7) by solving the equation for d. With the default standard deviation $\sigma = 1$, a maximum transmission range of about 10.5 m is considered and will be used as default maximum transmission range in the following. The simulator originally considered units of length instead of meters and a maximum transmission range of 6.83 units of length, which now corresponds to about 10.5 m here. Concerning the MAC protocol of the default configuration, the often used CSMA/CA MAC protocol from IEEE Std 802.15.4 is assumed. It is a contention based MAC protocol which performs carrier sense. However, collisions can still occur, for example, due to the hidden node problem. More details on this protocol can be found in Section 6.1.3.1. Here, the time t_{RX} in which a node is in receive mode, is assumed to be 20 ms by default, which is also reasoned in Section 6.1.3.1. Another parameter relating to simulation settings is the size of the simulation area. Here, a square simulation area with a side length of 64 units of length is chosen as default size. This corresponds to a side length of about 100 meters. The chosen default configuration of the initial topology is the random topology, which is described in detail in Section 6.1.4.2. However, the initial network has to be connected in order to achieve the goal of having all data in all nodes at the end in the case of a static network. For this reason, the network may be rebuilt several times at the beginning of a simulation run until a connected initial network has been created.

Under the default settings described here, an average of about 8 neighbors per node can be observed. Thus, neither a very dense nor a very sparse network is assumed.

On top of that, a mobile network which uses the time-based Random Walk mobility model is considered. This mobility model is also frequently used by related work. For details and more precise reasons for this choice and against the choice of a different implemented mobility model as default model, see Section 6.1.5. Here, the range for the possible speed of the individual nodes is chosen as $(0.75; 1.5]\,m/s$, which represents the speed of a human who walks at slow to medium pace.

For the default configuration, a scenario without separate node failures is assumed for the evaluations in Sections 7.1 and 7.2. However, link failures caused by packet collisions, or by nodes moving out of the transmission range of another node during a transmission, etc. are inherently included due to the chosen default transmission scheme, MAC protocol, and mobility model, for example.

Chapter 7

Simulation Based Evaluation of Broadcast Growth Codes

Contents

In this chapter, a comprehensive simulation based evaluation of BCGC is presented. This evaluation consists of an extensive evaluation of BCGC process parameters, such as the degree of codewords for transmission and the composition of codewords in Section 7.1. Concerning the degree of codewords, the use of different degree functions and different upper bounds for the codeword degree are evaluated in Section 7.1.1. With regard to the composition of codewords, specific symbol selection and requesting/sending a desired symbol are evaluated in Section 7.1.2. After the evaluation of BCGC process parameters, an evaluation of BCGC using different simulation settings can be found in the final section of this chapter in Section 7.2. This includes a comparison with related work, such as NoCoding/Forwarding, random linear network coding, and Growth Codes.

7.1 Evaluation of BCGC Process Parameters

For the evaluation of BCGC process parameters, the default simulation settings presented in Section 6.2.2 are used. Depending on the chosen settings of the simulation, different configurations of the process parameters may provide the best results in terms of a considered performance metric. Here, only one BCGC process parameter is varied at a time for the evaluation while all others are fixed and use their respective default configuration from Section 6.2.1. As there is a mutual influence between the BCGC process parameters, a different combination of parameter configurations may provide better results in a considered simulation setting than the optimal values determined individually. Nevertheless, not every possible combination of parameters is evaluated here to find optima in the considered specific default simulation setting as already explained in Section 6.2. Instead, the impact of each varied parameter shall be investigated individually.

In the following, parameters related to the degree of a codeword generated via BCGC, i.e. how many symbols are combined in a packet/codeword, are studied first. In this context, options for the used degree function and different additional upper bounds for the degree are considered. Subsequently, parameters are analyzed which refer to the specific selection of symbols, i.e. which exact symbols are used for a codeword. Here, the overall symbol selection is considered but also the use of a desired symbol. The goal of the following evaluations is to analyze the individual parameters of BCGC and their impact on the most important performance metrics presented in Section 3.3.

All plotted values here are average values of multiple seed-based simulation runs. For each seed, the average of the sinks' values is calculated if not stated otherwise. Some figures contain a broken axis. In this case, the two parts of the axis can have different scaling for clarity. For example, one part of the axis may be zoomed in to show details and the other part may be zoomed out for a better overview. Important figures which are shown repeatedly are the plotted values for

- the number of completed nodes,
- the number of sent/received packets,
- the RNP,
- the percentage of received distance-0-codewords,
- the percentage of received distance-1-codewords, and
- the percentage of received distance->1-codewords.

Concerning figures containing the number of completed nodes, the average time at which a respective number of nodes is completed, i.e. has reconstructed all distinct original symbols, is plotted. The figures with graphs for sent/received packets show the average number of performed broadcast transmissions to all direct neighbors, i.e. the number of packets sent by a node via broadcast transmission, until a respective point in time as sent packets. Furthermore, they show the average number of single packets received by a node until a respective point in time as received packets. For this purpose, a node counts its number of sent/received packets until the last node of the network is finished, not only until the node itself is finished. It is important to note that the 'sent'-/'received'-curves include only a few seeds or even only one seed towards the end. For this reason, these curves do not have a smooth course towards the end for some figures. In addition, the plotted values on the x-axis of a 'sent'-/'received'-figure, therefore, also differ from the values on the x-axis of a 'completed'-figure, which indicates the average time. In order to be able to distinguish the value of the entirety of the seeds from the value of a few seeds or one seed in a 'sent'-/'received'-curve, the minimum, median, and maximum of the seeds are also marked on the curve. For this purpose, the minimum/median/maximum time at which a seed was completed, i.e. the respective last node was finished, is used. Then, the corresponding average number of finally sent/received packets in the associated seed is marked at this point in time. The three markers do not have to be on the graph as they refer to only one seed each and do not represent an average value over all seeds like the graph itself. Furthermore, the median is marked instead of the average value in order to illustrate the position of the values of the seeds and to reduce the influence of individual extreme values. Figures which present the RNP show the average number of packets which had to be received to be able to reconstruct a respective number of distinct original symbols. With regard to figures that indicate the percentage of received distance-0-codewords/distance-1-codewords/distance->1-codewords, the average percentage of received distance-0-codewords/distance-1-codewords/distance->1-codewords within a certain predefined period of time before the time at which a respective average number of distinct original symbols has been reconstructed is plotted. This means that the time has been normalized for all seeds and all graphs used in the respective figure. Normalization was necessary because each seed and especially the seeds of different settings took different amounts of time to finish. The representation with a regular time-axis would be confusing and the graphs difficult to compare with each other. The x-axis now shows the time at which a respective average number of symbols has been reconstructed in a node. As a consequence, however, only in about the first two thirds of the x-axis all nodes of a network are taken into account for the calculation of the average percentage of received distance-0-codewords/distance-1-codewords/distance->1-codewords. Towards the end, only a few nodes and at the very end even only one node are considered. For example, an average value of 256 reconstructed symbols can only be reached if the last node of the considered seed has reconstructed all 256 symbols in a system with 256 distinct symbols. In addition, this last node is missing only a few or even only one symbol towards the end so that the graphs fluctuate strongly in the end. When looking at this type of figure, the focus should, therefore, be on the first two thirds of the x-axis, since this area contains information about the entire network and not only about a few remaining nodes.

In addition to the plotted values, tables show an overview of relevant performance metrics. Here, 'RNP_{100}', 'Node Latency/System Latency', 'PRR', '$\text{Throughput}_{\mathit{eff}}$', and 'Energy Consumption' are used as defined in Sections 3.3.1, 3.3.2, and 3.3.4 to 3.3.6. The indicated energy consumption refers to the whole network and is calculated according to Section 6.1.7. Furthermore, 'Time 95% of Nodes Completed' indicates the average time at which 95% of all nodes in the network are finished, i.e. have reconstructed all distinct original symbols. 'Sent Packets' is the average number of packets sent by a node as broadcast transmission until this node is finished. This value differs from the plotted graphs for sent packets, which show the number of sent packets until the whole system is finished. The value given here is, therefore, a supplement to the plotted values and cannot be read from the corresponding graph. Besides these listed average values, the standard deviation σ of a respective average value over all seeds and the absolute minimum/maximum value among all respective values of all seeds is also given for certain performance metrics.

7.1.1 Evaluation of Degree of Codewords for Transmission

The first BCGC process parameter, which will be evaluated here, is the degree of a codeword for transmission generated via BCGC. As already mentioned in Section 3.1, the degree d of a codeword indicates the number of symbols which have been combined for this codeword. In order to decode a degree-d-codeword and reconstruct exactly one of the symbols which are included in the codeword, $d-1$ of the contained symbols have to be already reconstructed by the considered node. For this reason, an appropriate degree of incoming codewords is crucial for optimal decoding and, thus, will be evaluated here. If the degree of incoming codewords is too high in regard to the number of the node's reconstructed symbols, many codewords cannot be decoded immediately after reception. They have to be stored in the node's waiting list instead and, thus, decoding is delayed. Therefore, more packets/codewords have to be received to reconstruct a certain number of symbols. However, if the degree of incoming codewords is too low, many received packets/codewords will be redundant, which means that all included symbols have already been reconstructed by the considered receiving node. The reception of a redundant codeword represents an unnecessary reception and should be avoided for this reason. It only increases the number of received packets/codewords without reconstructing a further symbol. A higher codeword degree reduces the probability of receiving a redundant packet/distance-0-codeword but also reduces the probability of immediate decodability of the received packet/codeword. Further theoretical details on the degree of a codeword for transmission in the case of BCGC can be found in Section 5.2. As described in Section 5.2.1, for BCGC, a degree function is used to determine a node's desired degree for a packet/codeword received next depending on its current number of reconstructed symbols. When a node creates a packet/codeword, the codeword degree is specified using the received desired degrees of its neighboring nodes. However, the quality of the chosen degree correlates strongly with the quality of the underlying degree function, which will, therefore, be studied in Section 7.1.1.1. On top of that, the nodes' desired degree and, thus, the used codeword degree can be limited in BCGC in order to have shorter packets and comply with the maximum packet sizes specified by IEEE Std 802.15.4, see Section 3.2.2. This aspect is evaluated in Section 7.1.1.2.

7.1.1.1 Evaluation of Degree Function

In the following, options for the applied degree function, i.e. a node's desired degree to be received next for a respective number of reconstructed symbols, are evaluated.

GC Degree Function vs. Delayed Degree Increase First, the GC degree function, as described in Section 5.2.1, is used. It maximizes the probability $p_{Dist1}(r, d)$ from Equation (5.1) to receive a packet/codeword which can be decoded immediately after reception if all symbols have the same probability to be included in a codeword. However, as described in detail in Sections 5.2.1 and 5.2.2, the GC degree function only considers the required number of received packets/codewords, but not the size of the respective packets. For this reason, it is proposed in Section 5.2.2 to use certain stochastic estimates of the resulting latency and delay the increase of the desired degree if a degree increase does not improve the estimated latency. The resulting delayed degree function is already shown in Figure 5.4 in Section 5.2.2 together with the original GC degree function. The differences to the GC degree function in terms of performance can be seen in Figures 7.1 and 7.2. Since the GC degree function is designed to keep the required number of received packets (RNP) as low as possible, which minimizes both energy consumption and latency for packets of given, fixed length, the RNP is compared in Figure 7.2. A lower RNP also ensures that the overall number of collisions is reduced in the case of a collision-prone MAC protocol. It shall be shown to what extent the approach with a delayed degree leads to a worse RNP. Furthermore, it is criticized here that the GC degree function does not take latency into account in the case of dynamic packet sizes or static packet sizes and a non-predefined packet length. As this issue is addressed by the described approach of delayed degree increase using stochastic estimations, the resulting latency is compared with the latency resulting from the original GC degree function in Figure 7.1.

In the following, the results are interpreted:
With the original GC degree function, the average degree of a received packet was 5.9. By contrast, the average degree of a received packet was 3.5 when using a delayed degree increase. This is due to the fact that with a delayed degree increase, lower degrees are

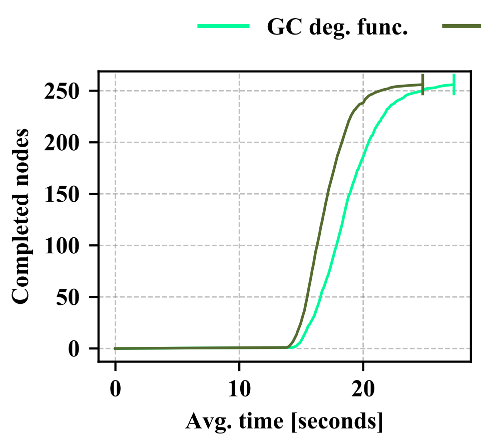


Fig. 7.1: Completed nodes

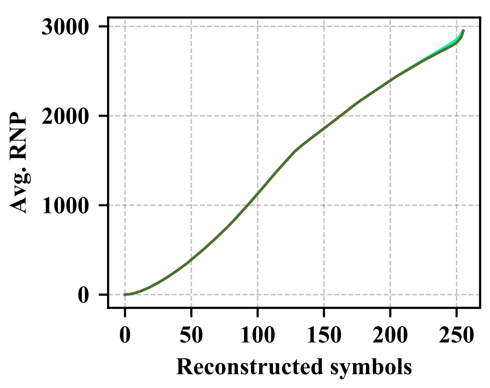


Fig. 7.2: RNP

used towards the end of the dissemination process than with the original GC degree function, as shown in Figure 5.4. By using lower degrees towards the end, packets are shorter from this point on. Consequently, more packets can then be sent/received within a considered period of time if using the default MAC protocol. As shown in Figure 7.2, in the considered default scenario, the average number of received packets needed to reconstruct a respective number of symbols does not differ noticeably in the two approaches and cannot be distinguished from stochastic fluctuations. These aspects together lead to the fact that latency is lower with a delayed degree increase using stochastic estimations compared to using a degree according the original GC degree function, see Figure 7.1. Thus, not using the maximum possible degree may not be a significant disadvantage in terms of RNP. It may also result in node and system latency being lower than with the standard GC degree function. As a result, approaches that limit the degree should be pursued and are, therefore, analyzed in more detail in Section 7.1.1.2.

A tabular summary of the RNP_{100}, latency, and further performance metrics for the GC degree function compared to a delayed degree increase can be found in Table 7.1.

GC Degree Function vs. Degree Functions with a Greater Slope Besides the fact that the degree function should not increase as high as specified by the GC degree function towards the end of the dissemination process, another deviation from the original GC degree function is discussed in Section 5.2. In realistic systems, especially in the beginning of the data dissemination process, symbols have a strongly different probability to be included in a codeword. Therefore, if using a degree d given by the GC degree function, the probability to receive a distance-0-codeword will be increased and the probability to receive a distance-1-codeword will be decreased compared to an

Performance metric	GC deg.func.	GC deg.func. + delayed deg.inc.
RNP_{100} $\pm\sigma$ (min, max)	2948 ± 154 (768, 6323)	2952 ± 161 (659, 5397)
Node latency $\pm\sigma$ $[s]$ (min)	18.7 ± 1.9 (12.3)	17.0 ± 1.1 (11.3)
System latency $\pm\sigma$ $[s]$	27.3 ± 4.2	24.8 ± 4.9
Time 95% nodes completed $[s]$	23.3	20.4
PRR	$1 - 0.318$	$1 - 0.322$
$\text{Throughput}_{\mathit{eff}}$ $[bit/s]$	2205	2413
Sent packets $\pm\sigma$	674 ± 40	660 ± 39
Energy consumption $[J]$	0.8255	0.7841

Table 7.1: Performance metrics for GC degree function vs. delayed degree increase

idealized scenario with equal probabilities. Therefore, the required number of received packets/codewords (RNP) is higher than would be expected under idealized conditions. Increasing the degree earlier than specified by the original GC degree function may be beneficial concerning the required number of packets in these cases and, thus, also concerning latency. For this reason, different degree functions are investigated here with a greater slope in the beginning than the slope of the GC degree function. At a later stage within the data dissemination process, when many symbols have already been reconstructed in the nodes, all symbols have approximately the same probability of being included in a codeword. Therefore, at a later point in time, all considered degree functions transition back to the GC degree function. In general, the original GC degree function maximizes the probability $p_{Dist1}(r,d)$ for uniformly distributed probabilities and provides a theoretically optimal degree for immediate decoding/reconstruction, see Section 5.2.1. The search space to find an optimal degree function is very large and a resulting optimal solution might then be valid only for the considered scenario. Thus, the influence of different slopes on the individual performance metrics is studied in the following instead. For this purpose, the GC degree function, a degree function with a slightly larger slope (slope incr. low), a degree function with a significantly larger slope (slope incr. med.), and an aggressive degree function with a very steep slope (slope incr. high) are used. The respective desired degree for a considered number of reconstructed symbols can be found in Figure 7.3 for the individual degree functions.

The goal of the GC degree function is to minimize the RNP. However, this is not fulfilled under the present conditions which deviate from the ideal scenario and should be improved by a greater slope. Therefore, in Figure 7.5, the RNP resulting from the use of the GC degree function is compared with the RNP resulting from the other degree functions with a greater slope. The RNP depends directly on the percentage of received

Performance metric	slope incr. low	slope incr. med.	slope incr. high
RNP_{100} $\pm\sigma$ (min, max)	2114 ± 118 (558, 5045)	1700 ± 98 (452, 3999)	2190 ± 456 (500, 96652)
Node latency $\pm\sigma$ $[s]$ (min)	13.7 ± 0.9 (8.3)	12.0 ± 1.3 (6.8)	16.1 ± 3.0 (4.7)
System latency $\pm\sigma$ $[s]$	23.8 ± 5.2	21.2 ± 4.8	289.5 ± 158.2
Time 95% nodes completed $[s]$	17.7	16.5	34.4
PRR	$1 - 0.323$	$1 - 0.329$	$1 - 0.343$
$\text{Throughput}_{\mathit{eff}}$ $[bit/s]$	3009	3432	2634
Sent packets $\pm\sigma$	493 ± 28	413 ± 24	567 ± 114
Energy consumption $[J]$	0.7063	0.6226	8.5191

Table 7.2: Performance metrics for degree functions with a greater slope

distance-0-codewords, distance-1-codewords, and distance->1-codewords, so these are shown in Figures 7.6 to 7.8, respectively. A change in the slope of the degree function primarily changes the distance with which codewords arrive at the receiver so that in Figures 7.6 to 7.8 the direct influence of the degree function can be observed. Since latency is a key performance metric of BCGC, which also determines energy consumption, the resulting latency for each degree function is shown in Figure 7.4. Latency depends on the RNP_{100}, but also on the packet size, i.e. how many packets can be sent/received per considered period of time. For this reason, the temporal development of the number of sent/received packets is shown in Figure 7.10. The lower the y-value of a curve, the fewer packets have been sent/received until the respective point in time. For the used default CSMA/CA MAC protocol, the default simulation setting, and the BCGC procedure, a decrease in the slope of a 'sent' / 'received' curve during the dissemination process means that packets became larger, i.e. that the codeword degree was increased. Figure 7.9 shows how many packets of which degree have been received with a respective degree function to separately visualize and emphasize how big the received packets have been. Please note that the graphs show a peak at degree $d = 46$ since $\mathit{maxDeg} = 46$ is used here as an upper bound for the codeword degree. An overview of these mentioned values and further important performance metrics of BCGC for the considered degree functions with a greater slope than the GC degree function can additionally be found in Table 7.2. For the corresponding values of the GC degree function, see Table 7.1.

In the following, the results are interpreted:
When using the original GC degree function, the average degree of a received codeword was 5.9. With the degree function with a slightly larger slope (slope incr. low), or significantly larger slope (slope incr. med.), or very steep slope (slope incr. high), the resulting average degree was 7.0, 10.3, or 11.6, respectively. The following identified

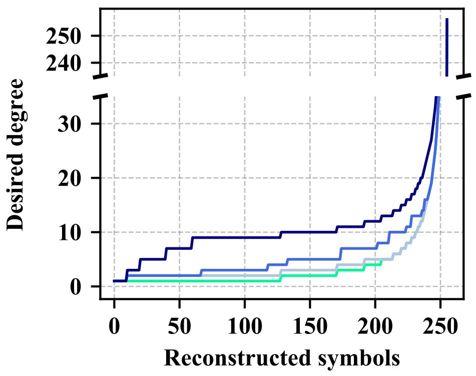


Fig. 7.3: Desired degree for next CW

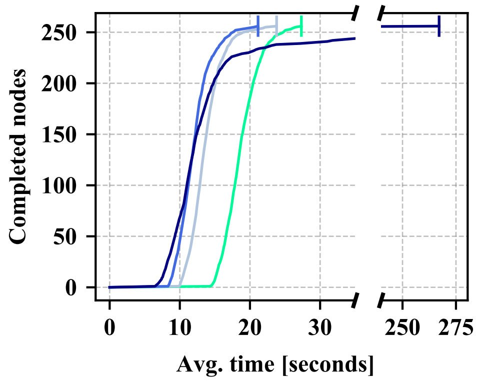


Fig. 7.4: Completed nodes

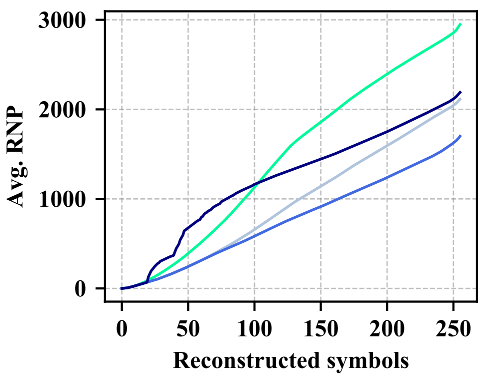


Fig. 7.5: RNP

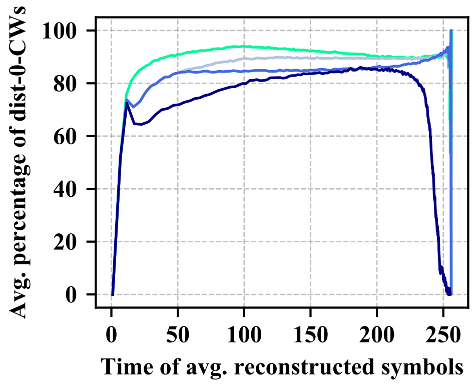


Fig. 7.6: Received dist-0-CWs

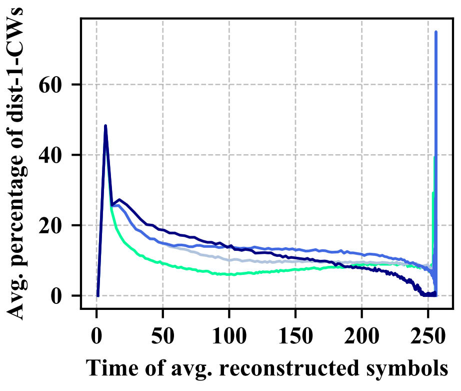


Fig. 7.7: Received dist-1-CWs

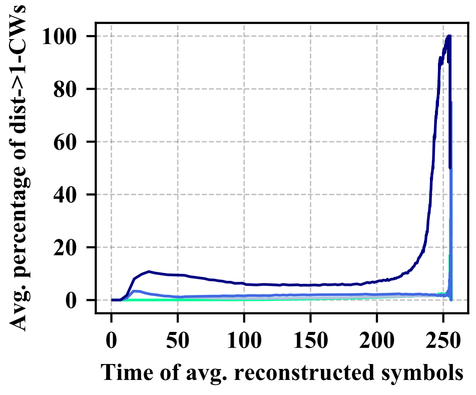


Fig. 7.8: Received dist->1-CWs

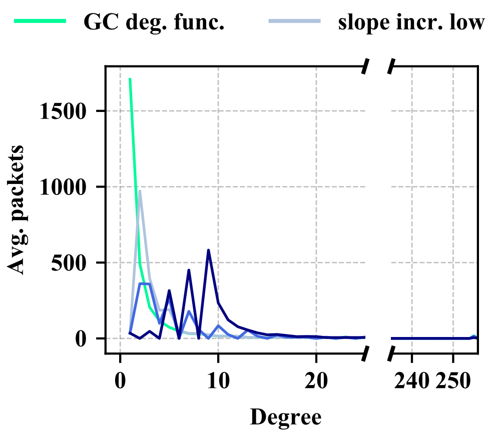


Fig. 7.9: Received packets of degree *d*

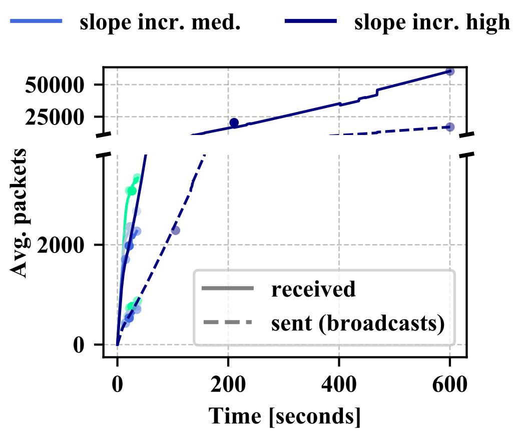


Fig. 7.10: Sent/received packets

correlations apply specifically to a system with a low degree of mobility and a not extremely high node density, as is the case in the default simulation settings used here, see Section 6.2.2. The overall very high percentage of received distance-0-codewords can be explained as follows: First, only symbols which have already been reconstructed by the node creating the codeword can be used for a codeword and second, symbols are selected randomly. The lower/higher starting value of the curve which shows the percentage of distance-0-codewords/distance-1-codewords is due to the fact that a node initially has reconstructed only its own symbol. Therefore, among the first received packets/codewords, there are many codewords which are distance-1-codewords for the node. However, the probability of the symbols to be included in a codeword is not equally distributed in the beginning. Thus, earlier than expected, for example, for the Coupon Collector's problem, more and more redundant codewords and fewer distance-1-codewords are received, unless the degree is increased appropriately. As a consequence, the percentage of distance-0-codewords increases strongly and the percentage of distance-1-codewords decreases accordingly. Symbols have a strongly unequal probability to be included in a codeword in the beginning of a data dissemination process. Therefore, the formulas on which the GC degree function is based deviate significantly from reality. Hence, the degree given by the GC degree function does not certainly maximize the probability that a received packet of this degree is a distance-1-codeword, but is too low. Codewords can only be composed of the symbols which have already been reconstructed by the respective node, and all reconstructed symbols are chosen with equal probability for this purpose. As a consequence, often, one of the proportionally many reconstructed symbols originating from the vicinity of the node is chosen. These symbols have already been received/transmitted many times and have also been reconstructed by all neighbors with high probability. Rarely, one of the proportionally few reconstructed symbols is chosen which have just been reconstructed by the node and, thus, have only rarely/never been re-transmitted. However, especially these rare symbols are interesting for the neighbors to receive since they are more likely to have not yet been reconstructed by the neighbors. With a higher codeword degree, the probability that these rare symbols are also included in a codeword is higher due to the larger number of symbols which are used in a codeword. Therefore, when using degree functions with a greater slope, the percentage of received distance-1-codewords and distance->1-codewords is higher than for the original GC degree function, see Figures 7.7 and 7.8. In the case of a higher codeword degree, more symbols are chosen for a codeword and these are chosen randomly here. For this reason, degree functions with a greater slope than the GC degree function cause a lower percentage of received redundant distance-0-codewords. This can also be seen in Figure 7.6. With each further reconstructed symbol the distribution of the probabilities of the symbols to be included in a codeword becomes more equal. Hence, the formulas on which the GC degree function is based correspond more and more to reality. This means that the GC degree function provides a progressively more suitable degree in terms of maximizing the probability of receiving a distance-1-codeword. As a consequence, the percentage of distance-0-codewords/distance-1-codewords decreases/increases by itself after a certain number of reconstructed symbols when using the GC degree function. The other considered degree functions transition to the GC degree function towards the end of the dissemination process. Thus, for a given number of reconstructed symbols, they return a similar or the same degree as the GC degree function. Consequently, they then deviate only slightly from the GC degree function in terms of the percentage of received distance-0-codewords and distance-1-codewords, see Figures 7.6 and 7.7.

The lower the density in the network and the lower the degree of nodes' mobility, the higher is the probability to receive redundant distance-0-codewords for a considered number of reconstructed symbols r and a degree d given by the GC degree function. Thus, the more the degree function should deviate from the GC degree function in the beginning and the more the degree should be increased. However, in the case of a very high degree of mobility or density, symbols quickly have a similar probability of being included in a codeword so that a calculation of the probability $p_{Dist1}(r,d)$ via Equation (5.1) is valid. Then, the probability $p_{Dist1}(r,d)$ can be maximized using Equation (5.1), i.e. the GC degree function can be applied. For more theoretical details on these relations, see Section 5.2.1. Evaluations for scenarios with different degrees of mobility can be found separately in Section 7.2.4.

As already mentioned, if using degree functions with a greater slope, the percentage of received distance-1-codewords is higher than for the original GC degree function, see Figure 7.7. Furthermore, the percentage of received redundant distance-0-codewords is lower, see Figure 7.6. However, if the slope is chosen too high, the percentage of distance-0-codewords decreases but the percentage of distance-1-codewords does not increase accordingly, as can be seen for the degree function with a very steep slope (slope incr. high). This can be explained by the fact that more distance->1-codewords arrive, see Figure 7.8. However, a higher percentage of incoming distance->1-codewords can lead to the decoding process being delayed or stalled. This can be seen, for example, in Figure 7.5 for the very aggressive degree function.

In general, the resulting RNP can be derived from the percentage of received distance-1-codewords. However, also codewords from the waiting list, i.e. codewords which previously arrived as distance->1-codewords, can be decoded. Thus, further symbols are reconstructed without having to receive additional codewords. For this reason, the RNP value can be lower, i.e. better, although the percentage of received distance-1-codewords is lower, i.e. worse, than with a different degree function. A flattening of the RNP curve in the course of the dissemination process can also be explained by the reconstruction of symbols via the waiting list, which can be clearly seen, for example, for the very aggressive degree function in Figure 7.5. In Table 7.2, the average value of the performance metric $\mathrm{RNP}_{100} \pm\sigma$ is indicated with the standard deviation σ of this average value over all seeds. In the case of the degree function with a slightly larger slope (slope incr. low), the degree function with a significantly larger slope (slope incr. med.), and the aggressive degree function with a very steep slope (slope incr. high), the value for $\mathrm{RNP}_{100} \pm\sigma$ is $2114 \pm 118, 1700 \pm 98$, or 2190 ± 456, respectively. However, if considering the average value of the standard deviations of all seeds $\hat{\sigma}$ instead, $\hat{\sigma}$ is $460, 397$, or even 3719, respectively. This shows that the resulting average value RNP_{100} does not differ strongly between the individual seeds. By contrast, the value within a seed, i.e. the value for the individual nodes in a considered network, differs significantly. In particular, for the aggressive degree function, there is an extreme difference between the individual nodes. The last nodes had to receive a significantly higher number of packets to be able to reconstruct all distinct original symbols than the first ones. As also can be seen in Tables 7.1 and 7.2, the RNP gain of the degree function with a slightly larger slope (slope incr. low), or a significantly larger slope (slope incr. med.), or a very steep slope (slope incr. high) compared to the original GC degree function is $\frac{2948}{2114} \approx 1.39$, $\frac{2948}{1700} \approx 1.73$, or $\frac{2948}{2190} \approx 1.35$, respectively. Furthermore, the corresponding throughput gain is $\frac{3009}{2205} \approx 1.36$, $\frac{3432}{2205} \approx 1.56$, or $\frac{2634}{2205} \approx 1.19$, respectively, see Tables 7.1 and 7.2. For the definitions of the term 'RNP gain' and 'throughput gain', see Section 3.3. Concerning energy consumption, by far the most energy was consumed using the degree function with a very steep slope, which even performed worse than the original GC degree function, and the least energy using the degree function with a significantly larger slope, see Tables 7.1 and 7.2.

Regarding latency, it can be seen that, for example, the degree function with a very steep slope (slope incr. high) performs worst although the RNP_{100} value is better than with the GC degree function. Generally, due to a higher desired degree in the case of the degree function with a very steep slope, larger packets are used. Thus, fewer packets are sent/received in a considered period of time, as can be seen in Figures 7.9

and 7.10. How many packets of which degree have to be received, i.e. how big packets are, influences latency but also energy consumption, see Table 7.2. Simulations show that if combining these two aspects, latency is finally best for the degree function with a significantly larger slope (slope incr. med.).

As described in Section 3.1.2, the RNP as well as latency, throughput, and energy consumption are relevant here. However, the goal here is not to find the global optimum for the considered example situation, but the basic mechanisms should be studied. For this reason, the degree function with a significantly larger slope than the original GC degree function (slope incr. med.) is used as the degree function of BCGC for further evaluations. This degree function has the biggest RNP gain and throughput gain, the lowest energy consumption, and performed best in terms of latency compared to the other presented degree functions, see Tables 7.1 and 7.2. Although this degree function is not necessarily optimal, it had the best overall performance among the considered degree functions and will, therefore, be used in the following for BCGC.

Long-Tail Problem The time course of the completed nodes shows that many nodes are completed quickly and a few take significantly longer, see Figure 7.4. This effect can also be seen when comparing the values of ‘Time 95% nodes completed’ with ‘System latency’ in Table 7.2. There are two fundamental reasons for this behavior, which are explained in the following.

First, at the beginning, each node has reconstructed only its own symbol, i.e. each symbol exists only exactly once in the network. The so-called long-tail problem arises, for example, from the fact that nodes at the boundary area have a longer average distance, counted in hops, and in particular a longer maximum distance to all other nodes in the network than nodes on the inner part. Thus, in static systems or systems with a low degree of nodes’ mobility, symbols have to traverse a longer average distance to reach the nodes at the boundary area. This distance is increased by the current link error rate, i.e. by occurring collisions, per hop and, thus, the effect is further amplified. Therefore, it theoretically takes longer for all symbols to arrive at the few nodes at the boundary area than at the many nodes in the interior due to the distance to be covered. In addition, only the symbols that have already been received and reconstructed by a node creating a codeword can be combined in this node’s new codewords. Furthermore, in the case of BCGC, a node cannot use all symbols it has already reconstructed for one packet/codeword but only a certain number of symbols since it has to take into account the desired degrees of its neighbors. Using these degrees to determine the degree of a codeword for transmission is necessary for the decodability of the packet/codeword in the neighboring nodes. Without using these degrees given by the neighbors’ degree function, all reconstructed symbols could be sent to the neighbors much faster. However, a respective received codeword could probably not be decoded immediately, or even never be decoded. Thus, the symbols would be present in codewords in the neighbors’ waiting lists, but not be reconstructed. Such a procedure would, therefore, only be reasonable if, for example, Gaussian elimination is used for decoding, which cannot be used here due to the addressed system setup in Section 3.1.1. For more details on this issue, see Section 5.2. If nodes are located/moving on a plane, as assumed in Section 3.1, the observed number of nodes in the vicinity of a node as well as the node density usually differ for each node. At the beginning of a data dissemination process, there are typically fewer distinct original symbols in the vicinity of a node located at the boundary area than for a node in the inner part. In addition, in the case of a static network or a low degree of nodes' mobility, a node at the edge receives fewer packets/codewords per period of time as it has fewer direct neighbors which are sending packets via broadcast transmissions. Thus, it takes more time for a node at the edge to receive a certain number of packets/codewords and, hence, to be able to reconstruct a certain number of symbols. Assuming that a node at the edge transmits at the same time intervals as nodes in the inner part of the area which is occupied by nodes, the node at the edge will transmit the same symbols more often, given the same number of reconstructed symbols and an unchanged degree. Nodes in the vicinity of a node at the edge often already know all of the symbols which have been sent several times by the considered node. As a consequence, they receive many distance-0-codewords from this node. This leads to the fact that a node at the boundary area receives more redundant codewords than a node in the inner part of the area, especially if the node and its surrounding nodes have reconstructed only a few symbols. For these reasons, in addition to the higher maximum distance to other nodes in the network, it takes even longer for nodes at the edge to receive/reconstruct all distinct original symbols than it does for nodes in the interior. For nodes with a low node density in their surroundings, the situation is the same as for the nodes at the edge described here.

The extent of the long-tail problem is strongly dependent on the initial positioning of the nodes, especially in the case of a static network or a low degree of nodes' mobility and a not extremely high node density. This topology then determines the node density in the vicinity of a node and the maximum distance of a node to all other nodes of the network counted in hops. Single badly connected, distant nodes can strongly influence the average values such as node latency and, in particular, system latency. For this reason, a simulative evaluation with multiple seeds is reasonable. However, some nodes will always be left behind by the others in a random placement of the nodes on a plane since there will always be nodes located at the edge. If, in the case of a static network, a grid topology was used instead of a random topology, the long-tail problem would be less obvious since all nodes, except for the nodes at the boundary region, would be surrounded by the same node density. Thus, there are no nodes with extremely many neighbors and, in particular, no very poorly connected nodes. Another factor that mitigates the long-tail problem is the nodes' mobility. Mobility of nodes ensures that symbols are distributed more quickly in the network and that the maximum distance to be covered by each symbol, measured in hops, becomes smaller. The higher the degree of mobility, the faster symbols are spread across the network. The long-tail problem is most severe in static networks, where poorly connected nodes remain poorly connected and there is no mobility to support the distribution of symbols. For the simulation results shown here, a low degree of nodes' mobility is assumed, as described in Section 6.2.2. Other degrees of mobility as well as a static network are considered separately in Section 7.2.4.

Second, another main cause of the long-tail problem is the fact that nodes at the edge have reconstructed only little while nodes in the neighborhood of these nodes have already reconstructed everything. If a node has already reconstructed all distinct original symbols and generates a new codeword, all symbols have the same probability to be included in this codeword. In this case, it would be optimal for the neighbors of such a node to use the GC degree function for determining their current desired degree. Any deviation from the degree given by the GC degree function will result in a lower probability of receiving an immediately decodable distance-1-codeword. Furthermore, the formulas from Equations (5.1), (5.5), and (5.6) can be used without restriction since the assumptions for the formulas are fulfilled. However, since symbols have a strongly varying probability to be contained in a codeword in the beginning of a dissemination process, degree functions with a greater slope are evaluated here. In particular, the aggressive degree function with a very steep slope returns a much higher degree than the GC degree function. Therefore, the probability of a received distance-1-codeword is reduced as soon as all symbols have the same probability to be included. Then, more distance->1-codewords will arrive and more packets/codewords have to be received to reconstruct a further symbol. In the worst case, the number of reconstructed symbols in the few remaining nodes remains the same over a long period of time and only the waiting list in the respective nodes is increased. This implies that it takes a long time until another node is completed. Especially with the aggressive degree function, this phenomenon can be observed clearly under the default simulation settings. For example, if a node has only reconstructed 62 symbols, its desired degree is 9 with the aggressive degree function with a very steep slope. If the neighbors have already reconstructed all original symbols, the node should optimally have a desired degree of 1, as is the case with the GC degree function. At a desired degree of 1, the node would only have to receive 1.3 packets/codewords in expectation for one more reconstructed symbol, as can be determined via Equation (5.5). However, if the node uses a desired degree of 9 instead, it has to receive an expected maximum number of 17209 packets/codewords to be able to reconstruct another symbol. This explains the shape of the curve for the aggressive degree function in Figure 7.4, i.e. the significant long-tail problem. This problem described here arises especially if there is a strong difference between the number of reconstructed symbols in almost all nodes compared to the number of reconstructed symbols in a few other nodes. This difference in the number of reconstructed symbols is especially evident in the case of a static network or a system with a low degree of nodes' mobility, not extremely high density and random

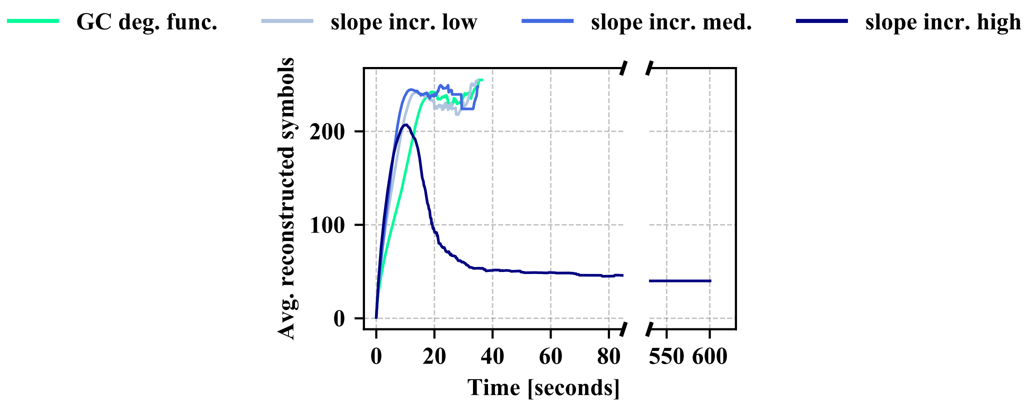


Fig. 7.11: Reconstructed symbols in non-completed nodes

placement of the nodes, as already described before. Figure 7.11 shows the average number of reconstructed symbols in a non-completed node at a considered point in time. It is important to note that the plotted graph includes only a few seeds or even only one seed towards the end due to its definition. For this reason, the graph fluctuates towards the end. In Figure 7.11, the difference in the number of reconstructed symbols can be clearly seen for the aggressive degree function if considering the number of completed nodes shown in Figure 7.4. Please note that the degree function with a significantly larger slope (slope incr. med.), which is used for BCGC, already reduces the long-tail problem significantly compared to the situation with the aggressive degree function, see Figure 7.4. In general, the problem is stronger the more the degree function deviates from the original GC degree function. A significant deviation from the GC degree function can, therefore, accelerate the time at which most nodes are completed, since fewer redundant packets/codewords are received. However, the time at which the last nodes finish may become much worse, so that the average values will also be worse.

When investigating the long-tail problem, it can be seen in Figure 7.11 that nodes which end up taking significantly longer than the other nodes do not wait for a few last symbols in the case of the aggressive degree function with a very steep slope (slope incr. high). Instead, these nodes are still missing a large number of distinct original symbols. The long-tail problem of the aggressive degree function shown in Figure 7.4 is, therefore, not due to the challenge of reconstructing the few last symbols or the very last symbol in each node, especially since there is no limitation of the degree yet. If $n-1$ symbols have been reconstructed in a considered node and, thus, a desired degree of n is used, an expected number of only one more packet/codeword has to be received to reconstruct the last symbol once the desired degree n is fulfilled.

To mitigate the long-tail problem, the goal should be to distribute all symbols as quickly as possible throughout the entire network, for example by limiting the degree or by preferentially forwarding new symbols in particular. For this reason, different upper bounds for the codeword degree are evaluated in Section 7.1.1.2. Furthermore, preferentially forwarding new symbols is addressed here by the presented approach with specific symbol selection, which is, therefore, evaluated in Section 7.1.2.

In general, the aspects of the long-tail problem which are inherent to the network topology cannot be completely eliminated. Nodes at the edge will always receive less packets per considered time interval and symbols will always have to traverse a longer average distance to reach these nodes in the case of a low degree of nodes' movement.

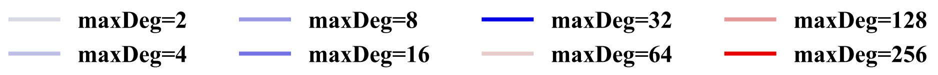
In the following, the limitation of the degree, which is explained in Section 5.2.4, is evaluated next. Limiting the degree also addresses the long-tail problem. It will lead to avoid choosing a degree which is much too high towards the end of the dissemination process, uses smaller packet sizes, and aims to distribute all symbols to all nodes quickly.

7.1.1.2 Evaluation of Upper Bound for Codeword Degree

In the following, different upper bounds *maxDeg* for the codeword degree are evaluated. Background information and details on the limitation of the degree can be found in Section 5.2.4. Since the default settings from Section 6.2.1 are considered here, only evaluations with dynamic packet sizes are executed in the following. The effect of limiting the degree would be even more significant with static packet sizes than with dynamic packet sizes. The reason for this is that with static packet sizes and a limitation of the degree, the packet sizes are smaller from the beginning than without an upper bound or with a higher upper bound. With dynamic packet sizes, on the other hand, the packet sizes increase with increasing degree. As a consequence, the limitation of the degree is noticeable only for the packets that have to be sent/received to reconstruct the last few symbols. The more the degree is limited, i.e. the lower *maxDeg* is chosen, the more packets/codewords have to be received to reconstruct a certain number of symbols in a node since usually more redundant codewords are received than without a limitation. At the same time, however, packets and, thus, the transmission time of packets becomes shorter. Since the reduced transmission time is more noticeable with static packet sizes than with dynamic packet sizes, even a lower *maxDeg* would be optimal in the case of static packet sizes. However, the fundamental impact of a limitation of the degree is the same for static and dynamic packet sizes. In general, simulations have to be executed to determine which upper bound *maxDeg* is optimal in terms of, for example, latency or energy consumption. Depending on the considered simulation settings, such as payload size or degree of nodes' mobility, and focused performance metric, a different upper bound may be optimal. Therefore, only the impact of different options for the upper bound *maxDeg* are investigated here under the default settings described in Section 6.2.2. It is not a goal to find the optimal upper bound for the considered default simulation settings. As an upper bound, *maxDeg*, the values $2, 4, 8, 16, 32, 64, 128$, and 256 are examined in the following for a system with $n = 256$ nodes/distinct original symbols. In the case of a lower upper bound and dynamic packet sizes, packets are affected by the degree limit much earlier than with a higher upper bound. Thus, significantly more packets are affected and not only the last packets to be received. As a consequence, a more significant difference in RNP_{100} or latency is to be expected between different lower upper bounds than between different higher upper bounds. Therefore, more lower than higher upper bounds *maxDeg* are considered here. For the default payload size of 20 Bytes, only a maximum degree of $maxDeg = 46$ is possible in order not to exceed the maximum packet size specified by IEEE Std 802.15.4. An upper bound $maxDeg = 256$ corresponds to not using any degree restriction. In this case, the used degree function is then exactly the selected degree function with a significantly larger slope (slope incr. med.) from Section 7.1.1.1. The resulting degree functions for the *maxDeg*-values mentioned above are depicted in Figure 7.12.

Here, the required number of received packets/codewords to reconstruct a certain number of distinct original symbols, the percentage of received distance-0-codewords, and the percentage of received distance-1-codewords is show in Figures 7.14 to 7.16 for the different used upper bounds. These values are especially interesting since they are directly affected by a degree restriction. The more the degree is restricted, the higher percentage of redundant distance-0-codewords/lower percentage of distance-1-codewords is expected to be received and, thus, the more packets are expected to have to be received to reconstruct a respective number of distinct original symbols. The corresponding required number of received packets/codewords can also be estimated via Equation (5.6). With a lower upper bound, more packets are expected to be needed, but the individual packets are shorter. In the case of shorter packets, more packets can be sent/received per time interval with the default MAC protocol used here, see Sections 6.1.3.1 and 6.2.2. This can be observed in Figure 7.18 for the used upper bounds $maxDeg = 2, 4, 8, 16, 32, 64, 128, 256$. The exact distribution of the packet lengths of the received packets can be derived from Figure 7.17 for each of these upper bounds. There, it can be seen how many packets of which degree, i.e. which length, had to be received in the end. A reduction of the latency, which can be achieved due to these correlations, was one of the main reasons for which the degree should be limited by an upper bound. Since packets of different lengths are used depending on the chosen upper bound, latency cannot be derived directly from the RNP_{100}. A shorter packet length can result in a lower latency even with a higher RNP_{100}. Latency is, therefore, shown separately in Figure 7.13. An overview of the resulting values of relevant performance metrics for $maxDeg = 2, 4, 8, 16, 32, 64, 128, 256$ can additionally be found in Table 7.3.

In the following, the results are interpreted:
By limiting the codeword degree to $maxDeg = 2, 4, 8, 16, 32, 64, 128$, or 256, the average degree of a received codeword was 2.0, 3.6, 5.4, 6.4, 7.2, 8.1, 9.1, or 10.3, respectively. If an upper bound *maxDeg* is used, a node's desired degree and, thus, the used codeword degrees are bounded above by *maxDeg*. Therefore, less symbols are combined in a



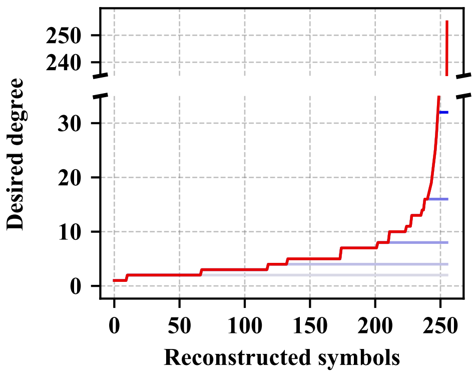


Fig. 7.12: Desired degree for next CW

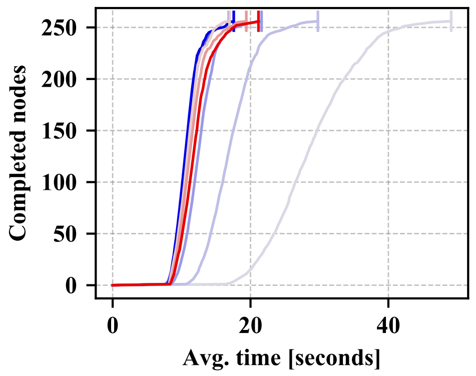


Fig. 7.13: Completed nodes

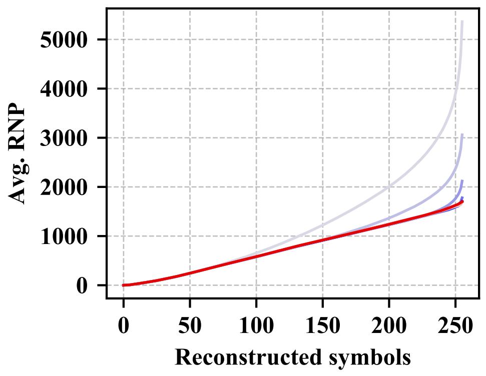


Fig. 7.14: RNP

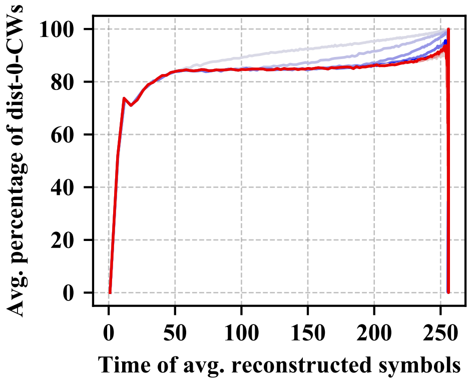


Fig. 7.15: Received dist-0-CWs

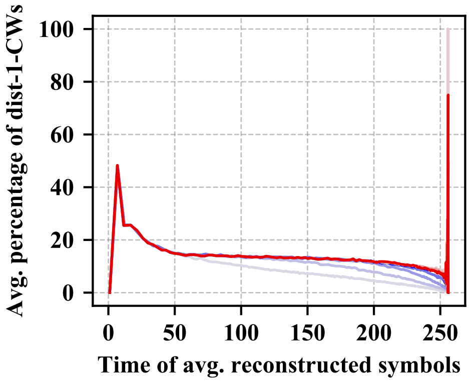


Fig. 7.16: Received dist-1-CWs

Performance metric	*maxDeg* = 2	*maxDeg* = 4	*maxDeg* = 8	*maxDeg* = 16
RNP_{100} $\pm\sigma$ (min, max)	5363 ± 427 (1842, 10858)	3061 ± 284 (803, 7427)	2119 ± 179 (498, 4602)	1779 ± 117 (464, 4203)
Node latency $\pm\sigma$ $[s]$ (min)	28.9 ± 2.2 (13.4)	17.1 ± 1.5 (8.5)	12.6 ± 1.0 (6.7)	10.9 ± 0.7 (6.1)
System latency $\pm\sigma$ $[s]$	49.1 ± 6.4	29.8 ± 6.7	21.6 ± 3.7	16.9 ± 2.1
Time 95% nodes completed $[s]$	39.3	22.9	16.6	14.3
PRR	$1 - 0.316$	$1 - 0.322$	$1 - 0.331$	$1 - 0.332$
$\text{Throughput}_{\mathit{eff}}$ $[bit/s]$	1427	2406	3261	3763
Sent packets $\pm\sigma$	1152 ± 90	677 ± 60	488 ± 40	417 ± 26
Energy consumption $[J]$	1.6012	0.9656	0.6948	0.5369

Performance metric	*maxDeg* = 32	*maxDeg* = 64	*maxDeg* = 128	*maxDeg* = 256
RNP_{100} $\pm\sigma$ (min, max)	1706 ± 88 (429, 3924)	1684 ± 101 (414, 3528)	1694 ± 102 (482, 4271)	1700 ± 98 (452, 3999)
Node latency $\pm\sigma$ $[s]$ (min) $[s]$	10.8 ± 0.6 (6.9)	11.0 ± 1.1 (7.0)	11.5 ± 1.1 (6.8)	12.0 ± 1.3 (6.8)
System latency $\pm\sigma$ $[s]$	17.6 ± 2.4	17.0 ± 3.2	19.4 ± 4.9	21.2 ± 4.8
Time 95% nodes completed $[s]$	13.9	14.4	15.8	16.5
PRR	$1 - 0.334$	$1 - 0.331$	$1 - 0.331$	$1 - 0.329$
$\text{Throughput}_{\mathit{eff}}$ $[bit/s]$	3822	3748	3596	3432
Sent packets $\pm\sigma$	404 ± 22	404 ± 30	409 ± 26	413 ± 24
Energy consumption $[J]$	0.5495	0.5197	0.5783	0.6226

Table 7.3: Performance metrics for BCGC degree function with different upper bounds *maxDeg*

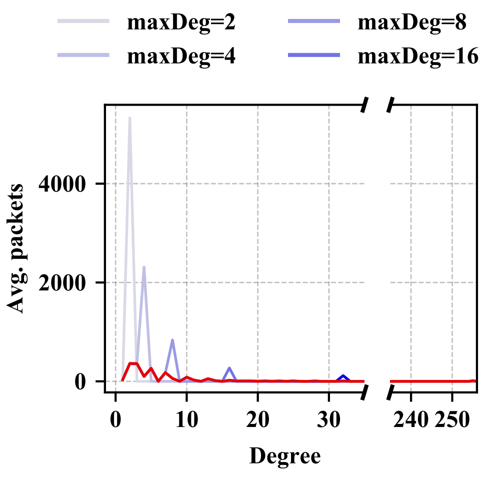


Fig. 7.17: Received packets of degree *d*

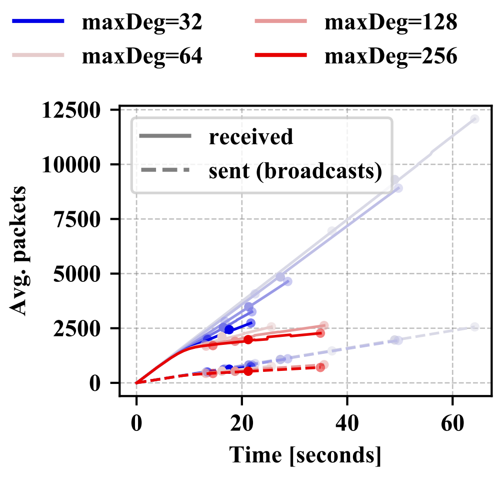


Fig. 7.18: Sent/received packets

codeword in the case of a lower *maxDeg* as soon as a certain number of symbols has been reconstructed and the respective degree *maxDeg* would have been exceeded with a higher *maxDeg* or with no limitation of the degree. The less symbols are combined in a codeword the higher is the probability that this codeword represents a distance-0-codeword for a receiver. Thus, the percentage of received redundant distance-0-codewords is increased the lower *maxDeg* is chosen, see Figure 7.15. The percentage of received distance-1-codewords is decreased accordingly, as shown in Figure 7.16. Starting from the number of reconstructed symbols above which the percentage of received distance-0-codewords is increased for a respective used *maxDeg*, the RNP increases, see Figure 7.14. Overall, each node has to receive additional packets depending on the chosen upper bound *maxDeg*. It can be seen in Figure 7.14 that a strong limitation of the degree, by e.g. $maxDeg = 2$ or 4, under the default simulation settings from Section 6.2.2 leads to a significantly higher RNP than without limiting the degree, i.e. with $maxDeg = 256$. In the case of $maxDeg = 2$ or 4, the RNP differs compared to the use of no limitation of the degree starting from about half or 2/3 of the number of symbols to be reconstructed, respectively. With a higher upper bound for the codeword degree, the difference in RNP is less and less significant. In the case of a higher upper bound than $maxDeg = 2$ or 4, the difference becomes apparent only at the last symbols to be reconstructed. Thus, a difference in RNP compared to no limitation of the degree can be seen in Figure 7.14 only up to $maxDeg = 8$ or 16. However, even with no limitation of the degree, i.e. with $maxDeg = 256$, the slope of the RNP graph increases slightly in the end. This means, more packets/codewords have to be received to reconstruct the last few symbols than for the symbols before. This can be recognized more easily in Figure 7.5 for the degree function with a significantly larger slope than the original GC degree function (slope incr. med.), which is used here, than in Figure 7.14 due to the scaling of the axes. In Figure 7.5, it can also be seen that the increased slope of the RNP graph in the end does not only occur with the degree function used here but with all compared degree functions from Section 7.1.1.1. The increase of the slope of the RNP curve towards the end is, thus, not due to the limitation of the degree provided that *maxDeg* is higher than a certain value. In Table 7.3, the difference in RNP_{100} can still be recognized up to $maxDeg = 16$. For a higher upper bound *maxDeg*, the difference in RNP_{100} is so small that it is difficult to distinguish from statistical fluctuations. One reason for this small difference is that the difference in the required number of received packets/codewords affects only the very last symbols that have to be reconstructed. With, for example, a degree limitation by $maxDeg = 32$, there is a difference to no limitation of the degree only from about 250 of 256 symbols to be reconstructed. In addition, with an upper bound of $maxDeg = 2, 4, 8, 16, 32, 64, 128$, or 256 and $n = 256$ nodes/distinct original symbols, an expected (maximum) number of only $128, 46, 32, 16, 8, 4, 2$, or 1 packets/codewords has to be received even to reconstruct the most difficult to reconstruct last symbol, respectively. This can be calculated using Equation (5.6) from Section 5.2.2, assuming that all symbols have the same probability of being included in a codeword. It can, thus, be seen that even with a very drastic restriction of the degree to $maxDeg = 2$ in comparison to the RNP without a limitation of the degree, not extremely many additional packets have to be received to reconstruct the last symbol. In the case of $maxDeg = 32$, the last symbol only requires an expected maximum number of 8 received packets/codewords instead of 1 packet/codeword as with no upper bound for the codeword degree. The second to last symbol and earlier symbols can each be reconstructed with even fewer additional received packets. Hence, there is no significant difference regarding RNP to no limitation of the degree in this case. The results presented here refer to BCGC with the degree function with a significantly larger slope than the original GC degree function (slope incr. med.). Overall, the RNP gain of the use of $maxDeg = 2, 4, 8, 16, 32, 64$, or 128 compared to no limitation of the degree, i.e. $maxDeg = 256$, was $\frac{1700}{5363} \approx 0.32$, $\frac{1700}{3061} \approx 0.56$, $\frac{1700}{2119} \approx 0.80$, $\frac{1700}{1779} \approx 0.96$, $\frac{1700}{1706} \approx 1.00$, $\frac{1700}{1684} \approx 1.01$, or $\frac{1700}{1694} \approx 1.00$, respectively, see Table 7.3. This means the RNP_{100} is worse for $maxDeg = 2, 4$, and 8 than for $maxDeg = 256$ and approximately identical for $maxDeg = 16, 32, 64$, and 128. Thus, as expected, there is no RNP gain by limiting the degree compared to using the degree function with a significantly larger slope than the original GC degree function (slope incr. med.) without any limitation of the degree. However, if *maxDeg* is not chosen extremely low, there is only little or even no difference in RNP_{100} compared to no limitation of the degree. For a different degree of nodes' mobility than the default setting assumed here, the simulation results for the respective upper bounds will differ from the results presented here. For example, the lowest upper bound with which no difference in RNP can be recognized compared to the RNP in the case of no limitation of the degree may be different. However, the general behavior described here remains the same. A detailed evaluation of BCGC using different degrees of nodes' mobility is presented separately in Section 7.2.4. As can be seen in Figure 7.17, a low upper bound *maxDeg* causes that significantly more packets have to be received than without a limitation of the degree. However, these many packets are shorter than in the case of a higher upper bound. The lower transmission time of the respective shorter packets can also be identified from Figure 7.18. There, it can be seen that the lower *maxDeg* is, the more packets are sent per considered time interval. For a payload size other than the default payload size of 20 Bytes, these differences can be more/less significant if the relative packet overhead is higher/lower. Different payload sizes are considered separately in Section 7.2.1.

The data distribution/collection process by the sinks themselves should be performed with few transmissions of preferably small packets to enhance reliability. Overall, with $maxDeg = 32$ or higher, the RNP and thus the RNP_{100} is not noticeably increased compared to not using any limitation of the degree. At the same time, packets are shorter and, therefore, the smaller *maxDeg* is chosen the more packets can be sent/received per considered time interval with the default MAC protocol. Combining these two aspects, it follows for the default simulation setting that an upper bound *maxDeg* between 32 and 64 has the lowest resulting latency, which is shown in Figure 7.13. In this case, if at all, only slightly more but shorter packets have to be received than without a limitation of the degree. Therefore, each node is completed, i.e. has reconstructed all distinct original symbols, faster. The time difference between the completion of the first and the last node in the network is, thus, also shorter since the transmission time and collision rate are reduced. Therefore, distant or poorly connected nodes can, then, be reached more quickly. In the case of $maxDeg = 2$ or 4, the difference in RNP compared to use of no limitation of the degree can already be recognized from about half or 2/3 of the number of symbols to be reconstructed, respectively. Thus, the difference accumulates so that RNP_{100} is significantly higher than with $maxDeg = 256$. This significant difference in RNP_{100} cannot be compensated by the smaller packet size. As a consequence, each node needs significantly longer to finish than without a limitation of the degree. In addition, the difference between the time at which the first node is completed and the time at which the last node is completed is increased. Therefore, it can be seen in Figure 7.13 that the limitation of the degree by $maxDeg = 2$ or 4 causes the latency-related curve to flatten considerably, resulting in a higher node latency and ultimately a higher system latency. For a lower *maxDeg*, latency is dominated by the significantly higher RNP_{100} and for a higher *maxDeg* by the larger packets. Thus, the optimum concerning node and system latency is expected to be found between $maxDeg = 16$ and $maxDeg = 64$ here. With $maxDeg = 2, 4, 8, 16, 32, 64$, or 128, the corresponding throughput gain was $\frac{1427}{3432} \approx 0.42$, $\frac{2406}{3432} \approx 0.70$, $\frac{3261}{3432} \approx 0.95$, $\frac{3763}{3432} \approx 1.10$, $\frac{3822}{3432} \approx 1.11$, $\frac{3748}{3432} \approx 1.09$, or $\frac{3596}{3432} \approx 1.05$, respectively, see Table 7.3. In addition, the energy consumption of the system was 1.6012J, 0.9656J, 0.6948J, 0.5369J, 0.5495J, 0.5197J, or 0.5783J in the case of $maxDeg = 2, 4, 8, 16, 32, 64$, or 128. In contrast, by using the degree function with a significantly larger slope than the original GC degree function (slope incr. med.) without limiting the degree, i.e. with *maxDeg* $= 256$, the energy consumption was 0.6226J. Thus, there was no improvement in throughput$_{eff}$ or energy consumption with *maxDeg* $= 2, 4$, and 8. The optimum for both performance metrics is in the range between *maxDeg* $= 16$ and *maxDeg* $= 64$. For subsequent evaluations, an upper bound of *maxDeg* $= 46$ is used for the default settings. This upper bound is within the range of possible values for *maxDeg* which achieved the best results here. In addition, it is the maximum upper bound with which the packet size specifications of IEEE Std 802.15.4 can still be met under the default simulation settings with the option to still add a desired symbol. By using *maxDeg* $= 46$, the average degree of a received codeword was 7.6.

The so-called long-tail problem has already been considered and explained in Section 7.1.1.1. There, many nodes finished quickly and a few took a very long time. However, the long-tail problem was caused by the network itself there and also by the use of a much steeper degree function than the original GC degree function. Here, not the very aggressive degree function but the degree function with a significantly larger slope than the original GC degree function (slope incr. med.) from Section 7.1.1.1 is used as a basis. With this degree function, the long-tail problem is not as severe as with the aggressive degree function since the number of reconstructed symbols in the nodes does not vary as much, see Section 7.1.1.1. In addition, an upper bound for the codeword degree is used here so that the degree could only be too low in the end. By limiting the degree by a suitably chosen *maxDeg*, each node can be completed earlier. Then, also the difference between the time at which the first node is completed and the time at which the last node is completed is shorter. However, the inherent properties of the network remain so that a few nodes still take longer than the other nodes of the network, see Figure 7.13. This means, the long-tail problem still exists if the degree is limited by an upper bound *maxDeg*. In addition, the long-tail problem can be more significant if a very low upper bound, such as *maxDeg* $= 2$ or 4 here, is used so that especially many packets have to be received and have to reach distant or badly connected nodes. In this case the time difference between the completion of the first and the last node in the network is increased and the curve of the completed nodes is flattened considerably, see Figure 7.13. Moreover, the long-tail problem can also be slightly increased if very long packets have to be received due to a very large upper bound *maxDeg* or no limitation of the degree. However, here, the long-tail problem is in no case as prominent as in Section 7.1.1.1 with the very aggressive degree function.

To conclude, limiting the maximum possible degree by *maxDeg* can improve the results concerning latency, throughput, and energy consumption. Furthermore, a slight delay due to the long-tail problem can still be observed regardless of whether a suitable *maxDeg* or no upper bound for the codeword degree is used.

7.1.2 Evaluation of Composition of Codewords for Transmission

Besides the degree of created codewords, also the choice of the symbols which are included in these codewords is an important process parameter of BCGC and influences the performance crucially. Evaluations concerning the symbol selection are, therefore, presented in the following. If all symbols of a received codeword have already been reconstructed in a considered node, the codeword is redundant and the reception of the codeword was unnecessary. Symbols which were often chosen for being included in a codeword have probably already been reconstructed in the neighboring nodes. Thus, receiving those symbols again does not bring any benefit for the neighbors. By contrast, if some symbols are only rarely selected for transmission, it takes a long time until they have been received by each node of the network. In general, symbols can be chosen randomly or selected specifically. For BCGC, symbols are selected depending on their probability to be useful for neighboring nodes. For this purpose, counters are used which indicate how often a respective reconstructed symbol has already been sent or received. In order to improve performance, symbols are selected as evenly as possible in BCGC, which means rare symbols are preferred to ones which seem to be more frequent. If specific symbol selection is used, rare symbols are expected to propagate through the network more quickly and reach each node earlier. As a consequence, nodes will be able to reconstruct all distinct original symbols more quickly, i.e. node latency and system latency will be improved, and, hence, the long-tail problem is addressed. These aspects are shown in more detail in the following evaluations. In this context, there is also the option to request a certain desired symbol in addition to the specific selection of symbols in the case of BCGC. Further theoretical details on the composition of codewords in BCGC have already been described in Section 5.3. In Section 7.1.2.1, the option to select symbols specifically will be evaluated in comparison to randomly chosen symbols. An additional request/use of a desired symbol will be evaluated separately in Section 7.1.2.2.

7.1.2.1 Evaluation of Specific Symbol Selection

In terms of codeword composition, three approaches are compared in the following: the random choice of each symbol (all random), the specific selection of one symbol (1 selected, BCGC-default), and the specific selection of all symbols (all selected).

Using specific symbol selection compared to random symbol selection will immediately change the percentage of received distance-0-codewords, distance-1-codewords, and distance->1-codewords. Therefore, corresponding results are shown in Figures 7.22 to 7.24. If codewords are more appropriately composed for the receiver than in a compared approach, the percentage of received distance-1-codewords is larger. The more redundant symbols are used for the codewords for transmission, the higher is the percentage of received distance-0-codewords. However, if there are many unknown

Performance Metric	all random	1 selected (BCGC)	all selected
$RNP_{100} \pm\sigma$ (min, max)	1689 ± 97 (418, 4278)	827 ± 85 (277, 3213)	1553 ± 510 (374, 8278)
Node Latency $\pm\sigma$ $[s]$ (min)	10.8 ± 0.8 (6.9)	5.5 ± 0.9 (3.2)	11.6 ± 6.4 (3.1)
System Latency $\pm\sigma$ $[s]$	16.7 ± 2.8	11.4 ± 2.9	35.5 ± 14.3
Time 95% of Nodes Completed $[s]$	14.1	8.0	23.4
PRR	$1-0.332$	$1-0.341$	$1-0.355$
$\text{Throughput}_{\mathit{eff}}$ $[bit/s]$	3805	7562	3915
Sent Packets	402 ± 26	202 ± 27	403 ± 172
Energy Consumption $[J]$	0.5190	0.3488	1.0677

Table 7.4: Performance metrics for different codeword composition strategies and $maxDeg = 46$

symbols for the receiver in the codewords, the percentage of received distance->1-codewords is larger. The goal of using specific symbol selection is to have to receive fewer packets/codewords to reconstruct a respective number of symbols than with a random composition of codewords. However, received distance->1-codewords may later become distance-1-codewords in the considered node's waiting list and, thus, provide a further reconstructed symbol. For this reason, the received percentage of distance-1-codewords alone does not indicate the RNP. The required number of received packets/codewords to reconstruct a respective number of distinct original symbols is, therefore, compared separately in Figure 7.21 for the three different approaches. Based on this graph, latency can be concluded. However, this is not easy to deduce if the slope is very different for two curves to be compared and especially if the slope changes significantly in the course of a curve. A greater slope in the later course of the dissemination process, i.e. when many symbols have been reconstructed in a node, does not only imply a greater increase in the number of packets which has to be received. It also implies an increasing number of larger packets in particular. As a result, such a change in the slope of the RNP-related curve has a significant impact on latency. Node and system latency are, thus, considerably more affected than is directly apparent with the RNP shown in Figure 7.21. For this reason, latency is plotted separately in Figure 7.20 and, for clarity, the distribution of received packet sizes can additionally be derived from Figure 7.19.

In the following, the results are interpreted:
In the case of specific symbol selection of exactly one symbol, a rare symbol is more likely to be added to a codeword than if no specific symbol selection is used. Rare symbols are, thus, preferentially forwarded by specific codeword composition using a specifically

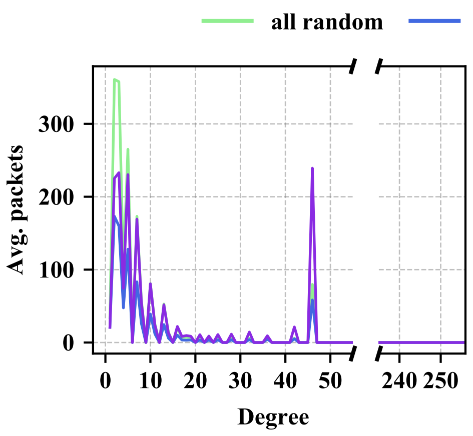


Fig. 7.19: Received packets of degree d

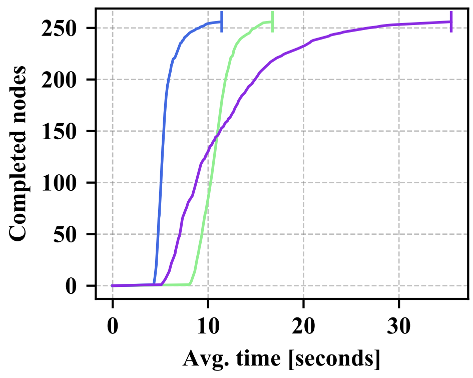


Fig. 7.20: Completed nodes

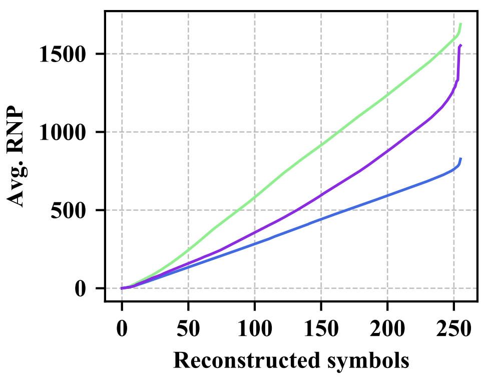


Fig. 7.21: RNP

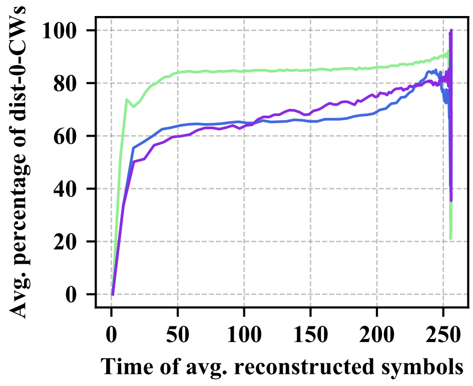


Fig. 7.22: Received dist-0-CWs

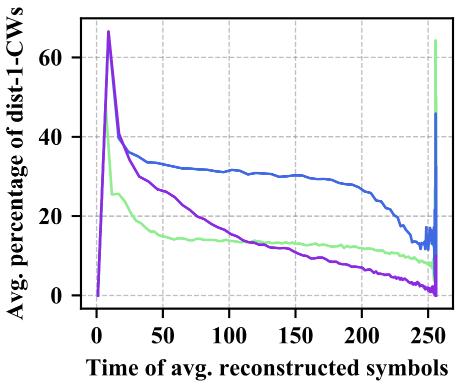


Fig. 7.23: Received dist-1-CWs

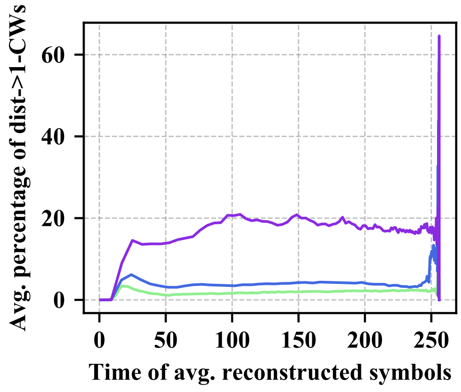


Fig. 7.24: Received dist->1-CWs

selected symbol. In this way, the uneven symbol selection discussed in Section 5.2.3 is mitigated from the beginning. The very high percentage of received distance-0-codewords due to uneven symbol selection is reduced compared to the approach using random symbol selection, see Figure 7.22. At the same time, the percentage of received distance-1-codewords or the percentage of received distance->1-codewords is increased, see Figures 7.23 and 7.24. A distance->1-codeword may be decodable at a later time and, thus, may also be beneficial. By specifically selecting a symbol, currently rare symbols are distributed through the network more quickly than with random symbol selection. Thus, overall, symbols are selected/forwarded more uniformly and, therefore, all symbols reach each node more quickly. As a consequence, each node has to receive fewer packets/codewords to reconstruct a respective number of symbols if exactly one symbol is specifically selected instead of randomly choosing all symbols, see Figure 7.21. Overall, the approach with the specific selection of exactly one symbol (1 selected, BCGC-default) provides the best results concerning RNP and in particular concerning RNP_{100}, see Figure 7.21 and Table 7.4. The average degree of a received packet was 7.6, 8.8, and 13.3 for the random choice of each symbol (all random), the specific selection of exactly one symbol (1 selected, BCGC-default), and the specific selection of all symbols (all selected), respectively. With random symbol selection, especially the number of received degree-1-codewords and degree-2-codewords is increased in comparison to specific symbol selection, see Figure 7.19. This is due to the fact that more codewords of these degrees have to be received until a respective number of symbols has been reconstructed in the case of random symbol selection. Furthermore, as can be seen in Figure 7.19, fewer packets have to be received from each degree in the case of selecting exactly one symbol specifically. As a consequence, node and system latency, which can be derived from Figure 7.20 or found in Table 7.4, are lower if exactly one symbol is selected specifically than with random symbol selection.

A specific symbol selection, i.e. an approach in which exactly one symbol or all symbols are specifically selected, leads to less distance-0-codewords being received than with random symbol selection, see Figure 7.22. However, if all symbols are selected specifically, the percentage of received distance->1-codewords is significantly higher than if only one symbol is selected specifically or if all symbols are selected randomly, see Figure 7.24. This is due to the fact that, in this case, too many symbols are combined in each codeword which are probably still unknown to the receiver. The approach with exactly one specifically selected symbol always yields the largest percentage of received distance-1-codewords. The approach of specifically selecting all symbols yields a higher percentage of received distance-1-codewords in the first part of the data dissemination process than the approach in which all symbols are randomly chosen and a lower percentage in the second half. However, decoding can also be performed via the waiting list and, thus, distance->1-codewords can also be profitable. Therefore, it can be seen in Table 7.4 with respect to the RNP_{100} that selecting all symbols specifically is better than choosing all symbols randomly. However, in Figure 7.21 it can be noticed that to reconstruct the last few symbols in a node, significantly more packets/codewords have to be received than for the other symbols in the case of selecting all symbols specifically. This behavior does neither occur with the specific selection of exactly one symbol (1 selected, BCGC-default) nor with the random choice of each symbol (all random). The increased slope of the RNP-curve for the approach with which all symbols are selected specifically (all selected) leads to the fact that each node has to receive a lot of packets with this approach especially at the end. These packets are then additionally relatively large, see Figure 7.19. Thus, each node takes longer to finish if all symbols are specifically selected. In addition, the time difference between the completion of the first and the last node of the network is increased as more packets have to be received per hop. It, thus, takes correspondingly longer to traverse the distance to distant or poorly connected nodes. This explains the flatter shape of the curve for this approach in Figure 7.20. Therefore, node and system latency are lower compared to the approach with randomly chosen symbols despite the fact that the RNP is better most of the time.

Overall, with respect to the RNP, it can be observed that with all three considered approaches, more symbols have to be received to reconstruct the last few or the last symbol, see Figure 7.21. This is only partly caused by the limitation of the degree to $maxDeg = 46$ in the case of random symbol selection, see Section 7.1.1.2. It is mainly inherent to the network topology as few symbols have to overcome a larger distance until they arrive at a respective considered node. Towards the end, in Figure 7.21, a similar behavior can be seen for the specific selection of exactly one symbol as for the random symbol selection. This described phenomenon cannot be completely eliminated by specific symbol selection since symbols generated by distant nodes always have to overcome a greater distance than symbols from the vicinity. However, as already explained, by using the specific selection of one symbol, each symbol, i.e. in particular also rare symbols, can reach a considered node with less transmissions than if random symbols are used. Therefore, the RNP and, thus, node and system latency are better with the specific selection of one symbol than with random symbol selection, see Figures 7.20 and 7.21. A few nodes will still take longer to reconstruct a certain number of symbols or all symbols than the many other nodes due to their position in the network and their degree of connectivity. These individual nodes will always have a greater average and maximum distance to the origin of the symbols than other nodes. Therefore, symbols will require more hops to reach these respective nodes. This is inherent to the network. The resulting RNP gain of BCGC with specific selection of exactly one symbol compared to random symbol selection was $\frac{1689}{827} \approx 2.04$, see Table 7.4. Compared to the procedure with the original GC degree function from Section 7.1.1.1, the obtained RNP gain was even $\frac{2948}{827} \approx 3.56$, see Table 7.4. Furthermore, the throughput gain compared to random symbol selection and compared to the original GC degree function was $\frac{7562}{3805} \approx 1.99$ and $\frac{7562}{2205} \approx 3.43$, respectively. In addition, an energy consumption of 0.3488J was observed for BCGC with specific selection of exactly one symbol, 0.5190J for random symbol selection, and 0.8255J for the original GC degree function from Section 7.1.1.1.

To conclude, the use of specific symbol selection of exactly one symbol provides better results concerning all considered performance metrics than BCGC with random symbol selection under the default conditions used here. Moreover, the results are considerably better than those of the initial variant of BCGC with the original GC degree function from Section 7.1.1.1. For the simulation results shown here, the default simulation settings described in Section 6.2.2 were used. Other settings may lead to different results, as explained in previous sections. In particular, a high degree of mobility may change the results concerning symbol selection. In this case, gathered information about the neighborhood may become invalid. Thus, the specific selection of a symbol may no longer represent an improvement compared to the random choice of symbols. The impact of different degrees of speed on BCGC is, therefore, investigated separately in Section 7.2.4. Section 7.2.4 also analyzes whether the specific symbol selection can even be a disadvantage in the case of a dynamic system. This is, however, not the case. Instead, BCGC with specific symbol selection perform even better in a dynamic simulation setting than the approach with a random choice of symbols, as finally shown and explained in Section 7.2.4. In a static network, gathered information has the highest validity and is, therefore, always beneficial to use. Specific symbol selection is, thus, an essential part of the BCGC procedure which improves its performance significantly.

7.1.2.2 Evaluation of Requested Desired Symbol

In the case of BCGC, there is the option to request a desired symbol when only one symbol is still missing in addition to the specific symbol selection. In the following, evaluations concerning requesting/sending a desired symbol are presented. The goal of this approach is to reduce the aspect of the long-tail problem which can be amplified by limiting the degree to *maxDeg*, see Sections 5.2.4 and 7.1.1.2. This means, the utilization of a desired symbol should in particular reduce the required number of received packets/codewords to reconstruct the last missing symbol in a node and, thus, possibly also reduce node and system latency. Packets are 2 Bytes larger, i.e. have a 0.064 ms longer transmission time, if a desired symbol is requested. Therefore, latency cannot be derived from the RNP alone and is, hence, considered separately in addition to the RNP in Figure 7.25. The required number of received packets/codewords to reconstruct a respective symbol is shown in Figure 7.26. An overview of the resulting values of further performance metrics when comparing BCGC without vs. with a desired symbol can be found in Table 7.5. Without limiting the degree to $maxDeg < n$, if only one symbol is still missing, a degree-n-codeword is requested/sent. Hence, it is not reasonable to request/add a respective desired symbol as a degree-n-codeword already contains all existing distinct original symbols. In Section 7.1.1.2, it is concluded that the aspect of the long-tail problem which is amplified by a limitation of the degree by *maxDeg* has only small extents under the default conditions considered here. With

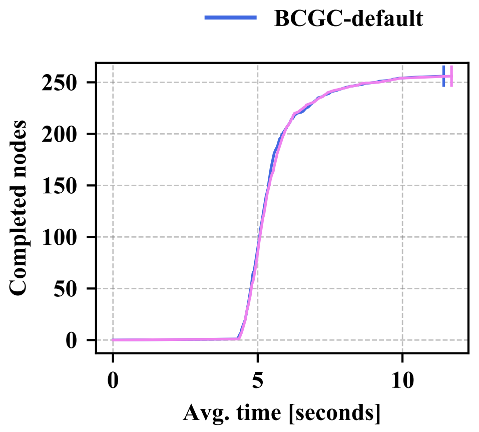


Fig. 7.25: Completed nodes

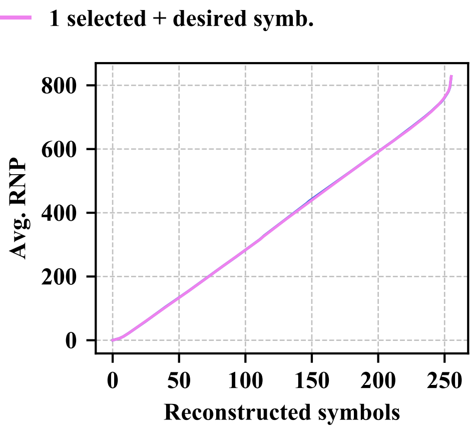


Fig. 7.26: RNP

Performance Metric	w/o desired symb. (BCGC)	with desired symb.
RNP_{100} $\pm\sigma$ (min, max)	827 ± 85 (277, 3213)	828 ± 94 (304, 3229)
Node Latency $\pm\sigma$ $[s]$ (min)	5.5 ± 0.9 (3.2)	5.6 ± 1.1 (3.1)
System Latency $\pm\sigma$ $[s]$	11.4 ± 2.9	11.7 ± 3.1
Time 95% of Nodes Completed $[s]$	8.0	8.0
PRR	$1 - 0.341$	$1 - 0.342$
$\text{Throughput}_{\mathit{eff}}$ $[bit/s]$	7562	7559
Sent Packets	202 ± 27	202 ± 31
Energy Consumption $[J]$	0.3488	0.3567

Table 7.5: Performance metrics for BCGC without vs. with desired symbol

the default upper bound *maxDeg* = 46 and 256 symbols, only an expected maximum number of about 6 packets/codewords has to be received to reconstruct the last missing symbol in a considered node. Therefore, there is only little room for improvement with the default value for *maxDeg*. With a maximum degree of *maxDeg* = 2 or 4 and $n = 256$ nodes/symbols, an expected maximum number of 128 or 46 received packets/codewords is needed to reconstruct the last symbol. Thus, there is more potential for improvement than in the case of the default value *maxDeg* = 46. Other circumstances, such as a different upper bound *maxDeg* or a different payload size, can, therefore, lead to other results and decisions. The impact of different payload sizes is evaluated separately in Section 7.2.1.

In the following, the results are interpreted:
For the following results, the default simulation setting as described in Section 6.2.2 is used. Simulations have shown that the RNP, i.e. the required number of received packets/codewords to reconstruct a respective number of symbols, may be slightly better if a desired symbol is used than without the usage of a desired symbol, see Figure 7.26. However, with the default settings, the difference is extremely small so that it is difficult to distinguish from statistical fluctuations. A possible minor improvement in the shape of the curve earlier than at $r = 255$ can be explained by the fact that if a node has not yet reconstructed a certain symbol, often nodes in its neighborhood have not reconstructed this symbol either. Thus, a sent desired symbol is not only beneficial for the node requesting this symbol. It may also be beneficial for other neighboring nodes which are still missing more than one symbol and also still need this specific symbol. These neighbors may, then, need fewer received packets/codewords to reconstruct a further symbol than without having received this desired symbol of the other node. However, as with any other specifically selected symbol, some neighbors may already know the symbol. If a node requests a desired symbol and one of its neighbors creates a codeword considering the received desired symbol, this symbol will be used instead of the respective previously specifically selected symbol. This way, generating an increased amount of distance->1-codewords for the neighbors which are still missing more than one symbol is avoided. The difference in performance is, thus, not as big as if the approach with a desired symbol had been compared to the approach with random symbol selection instead of specific symbol selection. Regarding the required number of received packets/codewords to reconstruct the last symbol, see Figure 7.26, no improvement distinguishable from statistical fluctuations could be detected by using a desired symbol. Overall, under the considered conditions, no improvement in the RNP_{100} can be observed when using a desired symbol instead of no desired symbol, see Table 7.5. Therefore, the RNP gain is $\frac{827}{828} \approx 1.00$, which means there is no noticeable gain. The throughput gain behaves analogously. In addition, each transmitted packet with a desired symbol has a larger transmission time. As a result, in the end, no improvement in node latency or system latency is achieved by using a desired symbol, see Figure 7.25 and Table 7.5. In particular, system latency is

even worsened by requesting a desired symbol, as shown in Figure 7.25 and Table 7.5. Since the use of a desired symbol does not noticeably improve any of the considered performance metrics and in some cases even worsens these, BCGC do not use a desired symbol in the following.

7.2 Evaluation of BCGC Using Different Simulation Settings

In this section, BCGC are evaluated using different simulation settings and compared with related work such as NoCoding, random linear network coding, and Growth Codes. For the evaluation of BCGC under varied simulation settings, the default configuration of BCGC process parameters is used, as described in Section 6.2.1. As declared in Section 6.2, only one parameter relating to simulation settings is varied at a time while all others use their respective default value, see Section 6.2.2. Thus, the impact of each simulation parameter can be investigated individually. However, a different simulation setting, i.e. a different combination of parameter values, can lead to different results since the individual parameters may have a differently strong influence on the performance of a considered procedure and can even have opposing effects. To evaluate the addressed network coding procedures under each reasonable combination of parameter values in relation to the simulation settings is beyond the scope of this thesis, as already explained in Section 6.2.

The goal of the following evaluations is to show that BCGC can be used for small but also for large payload sizes, that they can cope well with unreliable communication, and that they are even suitable for a large-scale network with many nodes. Furthermore, it shall be demonstrated that BCGC do not require a certain minimum degree of mobility, but can be used in a network with no or only a low degree of mobility as well as in a highly dynamic system. In addition, it shall be shown that even in very unreliable networks, where large parts of the network fail at an early stage, BCGC ensure that all distinct original symbols still present in the network arrive at all nodes and can eventually be reconstructed. Overall, the evaluation analyzes whether BCGC disseminate data reliably under the most diverse conditions so that finally all original data can be retrieved. Therefore, the data dissemination should be performed with few transmissions of preferably small packets in order to enable each sink node to reconstruct any number of distinct original symbols quickly without unnecessary energy consumption.

For this evaluation, NoC and RLNC are implemented as described in Chapter 4. An NoC frame has an overhead of 2 Bytes, see Section 4.1.3. Adding the overhead of an IEEE Std 802.15.4 data packet, see Section 3.2.2, the total NoC packet overhead is $2 + 17 = 19$ Bytes. In the case of 20 Bytes of payload, the size of an NoC packet is 39 Bytes. When considering a data rate of 250 kbit/s, the transmission time of this NoC packet is 1.248 ms.

RLNC which uses the Galois Field $GF(2^8)$ does not meet the packet size constraints of IEEE Std 802.15.4 for $n = 256$ nodes/symbols. Therefore, RLNC based on $GF(2)$ is considered here. For more details on the choice of the Galois Field, see Section 4.2.4. With $n = 256$ nodes/symbols, the overhead of an RLNC frame is 32 Bytes if using the Galois Field $GF(2)$, see Section 4.2.5. Thus, the total RLNC packet overhead is $32 + 17 = 49$ Bytes, including the overhead of an IEEE Std 802.15.4 data packet. Together with a payload of 20 Bytes, this results in an RLNC packet size of 69 Bytes. At a data rate of 250 kbit/s, this corresponds to a transmission time of 2.208 ms for an RLNC packet with 20 Bytes of payload.

Unlike NoC and RLNC, GC was implemented for this comparative evaluation only based on the description from Chapter 4, but with some necessary modifications to comply with the system setup from Section 3.1 and for a fair comparison with the other procedures. First, as described in Section 4.3.4, GC use two different methods to specify the symbols which are contained in a codeword, and always choose the method which produces smaller packet sizes. For BCGC and NoC, a 16-bit short address is used as ID here. For this reason, an ID should also have 16 bits in the case of GC for a fair comparison, instead of $\log_2 n$ bits as assumed by [Kam+06b], see Section 4.3.4. For the following simulation results, in the case of static packet sizes, GC use the bit format which is described in Section 4.3.4 since this representation is shorter than using 16-bit short IDs when considering 256 nodes. In the case of dynamic packet sizes, 16-bit short IDs are used for packets with a codeword of degree $d \leq 15$ and bit format is used for $d \geq 16$. Second, unidirectional broadcast transmissions to all direct neighbors are assumed for BCGC to fulfill the system setup from Section 3.1, instead of bidirectional data exchanges with unicast transmissions as in the original GC procedure. However, this causes that some additional adjustments have to be made. By using broadcast transmissions, no longer only one packet per round is received in the sink node. In the case of a CSMA MAC protocol, for example, the concept of rounds is also no longer used. Thus, to still follow the idea behind Equation (4.19), the degree transition points K_d have to be tied to the number of received packets instead of the number of elapsed rounds since no longer only one packet per round is received in the sink. The number of packets received in a node is an estimate of the number of packets received in the sink and this, in turn, is to estimate the number of symbols which have already been reconstructed by the sink. Due to this relationship, the most accurate value, the number of reconstructed symbols in the sink, is used here, and by then maximizing Equation (4.14), the optimal degree for the sink under the assumptions made for the equation is used as the proposed degree. This means, the GC degree function determines the proposed degree here. Another adjustment which is necessary due to the use of broadcast transmissions is that, at initialization, a considered node's own symbol is only stored in the node's separate storage space. It is not additionally stored in each storage space of the node's storage for data exchange as described in Section 4.3.2 for the original GC. Otherwise, due to the broadcast transmissions, a node's own symbol would be sent as a codeword more often than intended by the original GC. Also, no duplicate received codewords are added to a node's storage for data exchange, which occurs more often with broadcast transmissions compared to unicast transmissions. Results are only improved by this change as it reduces multiple transmissions of the same codewords. GC perform worse in terms of average values with a random placement of the single sink on the simulation area in the case of a static network or a low degree of nodes' mobility in comparison to BCGC. Therefore, the sink is placed in the central part of the simulation area for the following simulations. This also represents a change that improves the final results compared to the original GC algorithm. Therefore, the used adjusted GC perform better than the original GC in any scenario and can be used as an upper bound. For reasons of simplification, the term GC will be used for the adjusted GC in the following. Considering dynamic packet sizes, a GC frame has an overhead of between $1+2=3$ Bytes for a packet with a degree-$d=1$-codeword and $1+32=33$ Bytes for a degree-$d \geq 16$-codeword and $n=256$ nodes/symbols, see Section 4.3.4. Assuming static packet sizes, the GC frame overhead is, thus, 33 Bytes. If not stated otherwise, dynamic packet sizes are used for GC here. This results in a GC packet overhead between $3+17=20$ Bytes and $33+17=50$ Bytes. Hence, in the case of 20 Bytes of payload, the size of a GC packet is between 40 Bytes and 70 Bytes. With a data rate of 250 kbit/s, the transmission time of this GC packet is between 1.28 ms and 2.24 ms.

As described in Section 5.4, a BCGC frame overhead is between $1 \cdot 2 + \lceil 2 \cdot \lceil \log_2 46 \rceil / 8 \rceil =$ 4 Bytes and $46 \cdot 2 + \lceil 2 \cdot \lceil \log_2 46 \rceil / 8 \rceil = 94$ Bytes in the case of dynamic packet sizes, a maximum degree of *maxDeg* $= 46$, and with no desired symbol. Together with the overhead of an IEEE Std 802.15.4 data packet, this results in a BCGC packet overhead of between $4+17=21$ Bytes and $94+17=111$ Bytes. In the case of 20 Bytes of payload, the size of this BCGC packet is between 41 Bytes and 131 Bytes. At a data rate of 250 kbit/s, this corresponds to a transmission time of between 1.312 ms and 4.192 ms. For the calculation of the maximum BCGC packet size, the maximum possible *maxDeg* was used, at which for a payload of 20 Bytes the maximum packet size of 133 Bytes specified by IEEE Std 802.15.4 is still fulfilled also if a desired symbol would have been used.

All plotted values shown in the following are average values of multiple seed-based simulation runs. For each seed, the average of the sinks' values is calculated if not stated otherwise.

7.2.1 Varied Payload Size

In the following, different payload sizes are considered: A very small payload size of 1 Byte, the default payload size of 20 Bytes, as well as $40, 60, 80,$ and 100 Bytes. Significantly bigger payload sizes do not comply with the maximum packet size specifications of IEEE Std 802.15.4, where the size of a PHY Packet is restricted to 133 Bytes, see

	Packet overhead	overhead depending on num. of nodes/distinct original symbols n
NoC	19 Bytes	-
RLNC for $n = 256$	49 Bytes	+
GC for $n = 256$	$20 - 50$ Bytes	+
BCGC with $maxDeg = 6$	$20 - 30$ Bytes	-
BCGC with $maxDeg = 46$	$21 - 111$ Bytes	-

Table 7.6: Overview of packet overhead in the case of NoC, RLNC, GC, and BCGC

Section 3.2.2.

The total RLNC packet overhead is 49 Bytes in the case of $n = 256$ nodes/symbols, including the overhead of an IEEE Std 802.15.4 data packet. Hence, 100 Bytes of payload exceed the maximum allowed packet size specified by IEEE Std 802.15.4. For GC with $n = 256$, the minimum GC packet overhead is 20 Bytes and the maximum packet overhead occurring during the data dissemination process is 50 Bytes. Thus, GC with 100 Bytes of payload also exceed the specified limit of 133 Bytes. For comparison with BCGC, however, the respective simulations are still executed. By contrast, BCGC with a suitably chosen maximum degree *maxDeg* comply with the specifications of the standard even with a payload size of 100 Bytes. In the case of 100 Bytes of payload, the maximum possible upper bound *maxDeg* is 6. With $maxDeg = 6$, the BCGC packet overhead is only between 20 and 30 Bytes. In the case of the default upper bound $maxDeg = 46$, the BCGC packet overhead is between 21 and 111 Bytes. Due to the small NoC packet overhead of 19 Bytes, there are no problems with 100 Bytes of payload in the case of NoC.

The larger the payload size, the less impact a larger packet header has. With a very large payload, the size of the packet header is negligible. Due to the packet size restrictions in IEEE Std 802.15.4, however, only a payload size of up to 100 Bytes is admissible here. If a different specification is used, significantly larger payload sizes may be permitted and the effects described below may then be even more apparent. The header sizes of the considered procedures differ more or less significantly depending on the number of used nodes/symbols n. In addition, the header sizes of GC and BCGC also increase during the dissemination process. There, the codeword degree starts at 1 and monotonically increases up to n for n nodes/symbols or up to *maxDeg*, respectively. For an overview of the packet overhead in the case of NoC, RLNC, GC, and BCGC, see Table 7.6. For $n = 256$ nodes/symbols the relative difference of the header sizes of the procedures to be compared is not very large. Therefore, primarily the procedures themselves decide about the performance. With a chosen $maxDeg = 46$, BCGC even have the largest header among the compared procedures towards the end of the dissemination process. This large packet size is, however, only used for few packets. Moreover, *maxDeg* $= 46$ is the maximum permissible upper bound for the codeword degree when considering the default payload size of 20 Bytes with which the IEEE Std 802.15.4 packet size specifications can still be met. Respective packets are, therefore, by definition large and close to the specified limit of 133 Bytes. For other possible upper bounds, such as the used *maxDeg* $= 6$, the maximum occurring packet size is significantly smaller. In contrast to BCGC and NoC, the packet header for RLNC and GC grows with the number of nodes/symbols in the system so that it can become considerably larger. For example, with $n = 1024$ nodes and $GF(2)$, the RLNC header is already $17 + 128 = 145$ Bytes and the GC header is up to $17 + 1 + 128 = 146$ Bytes. In this case, the performance of the respective RLNC or GC procedure will be significantly worse than the performance results shown here. Furthermore, in particular, the packet size specifications of IEEE Std 802.15.4 are then no longer met.

In the following, as described in Section 6.2.2, the default number of $n = 256$ nodes/symbols is considered. Therefore, mainly the procedures, and thus e.g. the required number of received packets, and not the packet header make the difference. For this reason, the impact of the respective considered payload size is evaluated for the individual procedures with respect to the relevant performance metrics for the default setting. Furthermore, it is evaluated how the procedures, i.e. NoC, RLNC, GC, and BCGC, compare to each other under these conditions. A larger number of nodes or a significantly larger payload size can lead to different results, but the basic principles shown here remain valid.

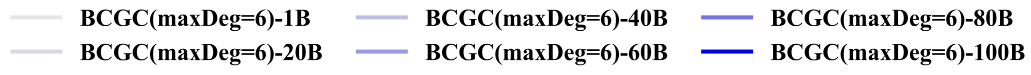
In general, a larger payload size or packet header, i.e. a larger packet, means that fewer packets can be sent per considered time interval when using the default CSMA/CA MAC protocol from Sections 6.1.3.1 and 6.2.2. Thus, it takes longer for a node to send/receive a certain number of packets. To compare this aspect for different payload sizes and for different procedures using the same payload size, the number of packets sent by a node via broadcast transmission and the number of single packets received by a node are plotted in Figures 7.28, 7.33, 7.35, and 7.37. Due to the changing time at which packets are sent depending on the used payload, the required number of received packets to reconstruct a respective number of symbols can change, especially for BCGC. Therefore, the RNP is shown for BCGC in Figure 7.29. Caused by the changed number of sent/received packets and the possibly different RNP, both node latency and system latency may also be different. These are essential performance metrics and are, therefore, presented in Figures 7.27, 7.32, 7.34, and 7.36. For a tabular overview of the values of relevant performance metrics resulting from varying the payload size, see Tables 7.7 to 7.10. A comprehensive description of all values plotted in figures and parameters given in tables can be found at the beginning of Section 7.1. Packet headers of NoC, RLNC, GC, and BCGC can be similar in size, as for example with $n = 256$ nodes/symbols and the added IEEE Std 802.15.4 header of 17 Bytes. They also can be negligible with a correspondingly large payload. In this case, the RNP_{100} primarily determines the resulting latency and energy consumption. As a consequence, a procedure with a low

RNP_{100} will perform better than a procedure with a high RNP_{100}. With a smaller payload size, even a small difference in header size becomes more relevant, so that the resulting packet size together with the RNP_{100} is decisive, for example, for latency.

In contrast to the other compared procedures, in the case of BCGC, the procedure itself is also influenced by the used payload size. There, depending on the payload size, a different upper bound *maxDeg* for the codeword degree is possible so that the packet size specifications by IEEE Std 802.15.4 are still adhered to. The changes that occur due to a chosen different upper bound have already been examined in detail in Section 7.1.1.2. For 100 Bytes of payload, the largest possible upper bound is $maxDeg = 6$ if the option to still add a desirable symbol should be retained. If a different specification is assumed as a basis, a higher *maxDeg* may be possible.

For a respective considered payload size, the maximum possible upper bound for the codeword degree does not necessarily have to provide the best results in terms of latency, throughput, and energy consumption. However, the corresponding RNP_{100} will be lower than if a lower upper bound were used, as already discussed in detail in Sections 5.2.4 and 7.1.1.2. As already concluded in Section 7.1.1.2, for each used payload size, a different upper bound *maxDeg* may be optimal for a respective considered performance metric. The higher *maxDeg* is chosen, the lower is the required number of received packets to reconstruct a respective number of symbols in a node. In addition to the RNP_{100}, the respective packet transmission time also affects latency and, thus, energy consumption. However, the larger the payload size, the less influence a chosen *maxDeg*, in percentage terms, has on the packet size and, hence, on the transmission time of a packet. As a consequence, it is to be expected that with a larger payload size a higher *maxDeg* is optimal in terms of latency than with a smaller payload. However, due to the packet size restrictions imposed by IEEE Std 802.15.4, only an upper bound of up to $maxDeg = 6$ is possible for a payload size of 100 Bytes, for example. Since it is not the goal to find an optimal upper bound for each considered payload size and relevant performance metric, no separate evaluation of the upper bound for each payload size is performed here. Instead, $maxDeg = 6$ is chosen in order to use the same upper bound for all considered payload sizes and, thus, to be able to see the impact of the payload size itself without having additional influences by different upper bounds.

In the following, the results are interpreted:
With increasing payload size, packets become larger. In Figures 7.28, 7.33, 7.35, and 7.37, it can be seen for all compared procedures that nodes send less frequently if packets are larger and, therefore, fewer packets are received by a node per considered time interval. These less frequent transmissions are due to the underlying default MAC protocol, which is described in detail in Section 6.1.3.1. If a different MAC protocol were used, the points in time at which packets are sent by a node may be maintained and instead the collision rate may be considerably increased. Different MAC protocols are, therefore, considered in Section 7.2.2. In Figures 7.28, 7.33, 7.35, and 7.37, the plotted values of the ‘sent’ curve refer to broadcast transmissions of a node to all



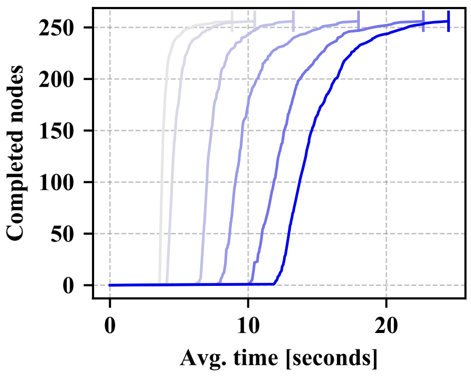


Fig. 7.27: Completed nodes

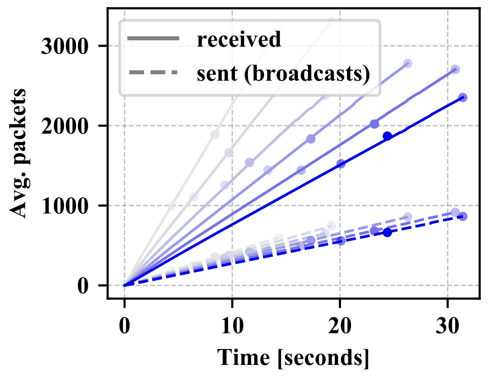


Fig. 7.28: Sent/received packets

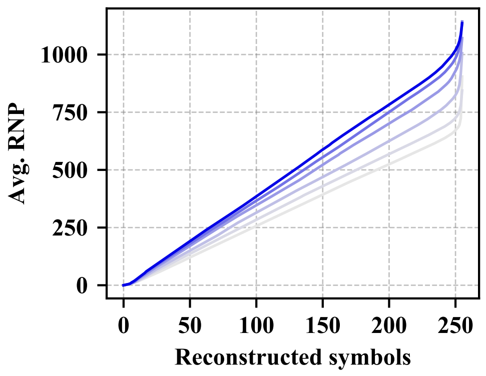


Fig. 7.29: RNP

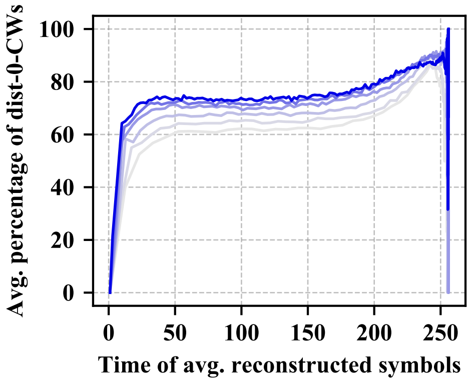


Fig. 7.30: Received dist-0-CWs

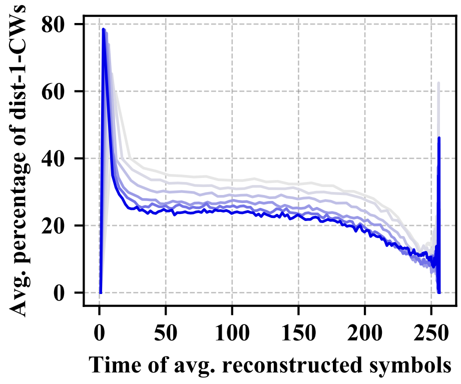


Fig. 7.31: Received dist-1-CWs

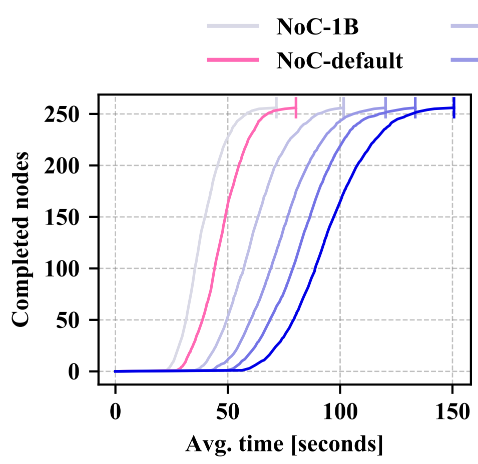


Fig. 7.32: Completed nodes

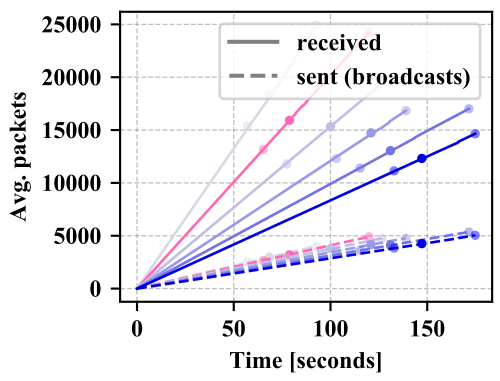


Fig. 7.33: Sent/received packets

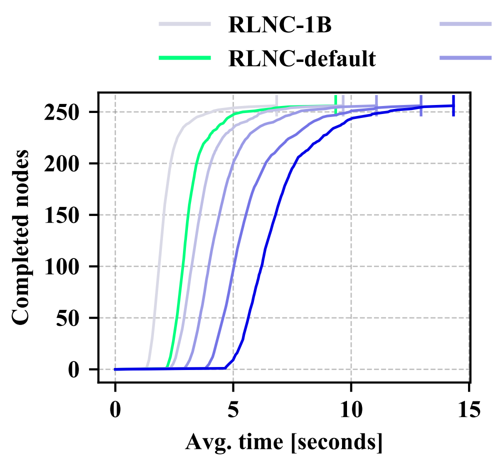


Fig. 7.34: Completed nodes

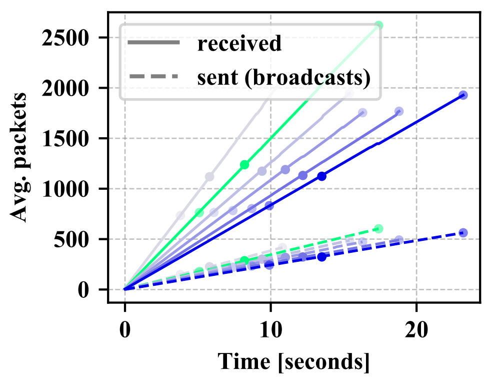


Fig. 7.35: Sent/received packets

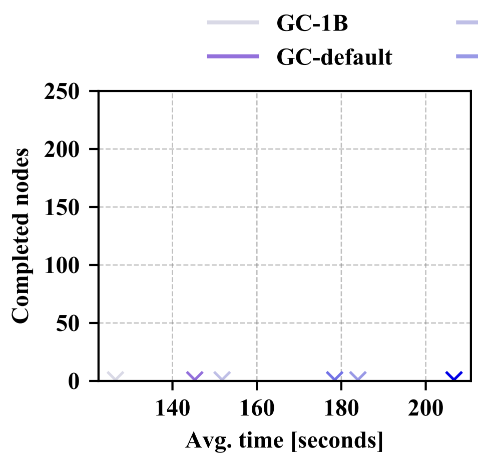


Fig. 7.36: Completed nodes

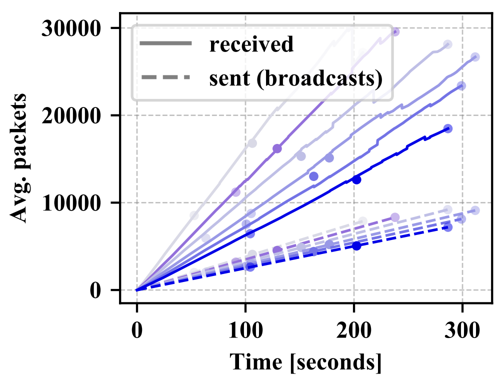


Fig. 7.37: Sent/received packets

Performance Metric	BCGC-1 B	BCGC-20 B	BCGC-40 B	BCGC-60 B	BCGC-80 B	BCGC-100 B
RNP$_{100}$ $\pm\sigma$ (min, max)	904 ± 364 (285, 2588)	844 ± 105 (295, 1948)	1006 ± 185 (293, 2311)	1071 ± 64 (349, 2991)	1143 ± 133 (409, 3553)	1135 ± 55 (455, 3011)
Node Latency $\pm\sigma$ $[s]$ (min)	4.0 ± 1.6 (2.5)	4.9 ± 0.6 (3.3)	7.5 ± 1.5 (4.4)	9.9 ± 0.5 (6.4)	12.8 ± 1.6 (8.4)	14.9 ± 0.6 (10.4)
System Latency $\pm\sigma$ $[s]$	8.8 ± 3.1	10.5 ± 3.4	13.3 ± 2.8	18.0 ± 3.6	22.7 ± 3.8	24.5 ± 3.1
Time 95% of Nodes Completed $[s]$	5.2	6.6	10.1	14.2	16.9	20.2
PRR	$1-0.243$	$1-0.326$	$1-0.372$	$1-0.392$	$1-0.400$	$1-0.402$
Throughput$_{eff}$ $[bit/s]$	559	8510	11136	12423	12953	13790
Sent Packets	168 ± 67	190 ± 23	270 ± 51	326 ± 18	384 ± 43	413 ± 20
Energy Consumption $[J]$	0.2933	0.3385	0.4163	0.5511	0.6795	0.7215

Table 7.7: Performance metrics for BCGC with *maxDeg* $= 6$ and varied payload size

Performance Metric	NoC-1 B	NoC-20 B (default)	NoC-40 B	NoC-60 B	NoC-80 B	NoC-100 B
RNP_{100} $\pm\sigma$ (min, max)	10152 ± 1048 (3860, 24680)	9489 ± 483 (3742, 17881)	9427 ± 901 (3398, 21269)	8978 ± 543 (3655, 18326)	8294 ± 528 (3997, 17113)	7918 ± 336 (3318, 14549)
Node Latency $\pm\sigma$ $[s]$ (min)	39.0 ± 3.7 (18.0)	47.5 ± 2.5 (23.7)	61.6 ± 5.6 (28.0)	74.1 ± 4.4 (35.9)	83.5 ± 4.9 (43.3)	94.6 ± 3.8 (41.9)
System Latency $\pm\sigma$ $[s]$	71.6 ± 11.0	80.4 ± 13.1	101.6 ± 15.8	120.2 ± 10.2	133.4 ± 14.8	150.7 ± 12.8
Time 95% of Nodes Completed $[s]$	54.9	64.4	86.6	100.6	110.1	124.9
PRR	$1-0.195$	$1-0.302$	$1-0.364$	$1-0.394$	$1-0.404$	$1-0.407$
$\text{Throughput}_{\mathit{eff}}$ $[bit/s]$	53	865	1340	1663	1968	2169
Sent Packets	1699 ± 163	1943 ± 101	2320 ± 213	2558 ± 153	2635 ± 154	2731 ± 111
Energy Consumption $[J]$	2.4054	2.6410	3.2470	3.7464	4.0609	4.4958

Table 7.8: Performance metrics for NoC with varied payload size

Performance Metric	RLNC-1 B	RLNC-20 B (default)	RLNC-40 B	RLNC-60 B	RLNC-80 B	RLNC-100 B
RNP_{100} $\pm\sigma$ (min, max)	382 ± 19 (257, 1405)	460 ± 92 (257, 1893)	445 ± 28 (258, 1434)	473 ± 48 (264, 1761)	521 ± 91 (267, 1790)	565 ± 121 (266, 1784)
Node Latency $\pm\sigma$ $[s]$ (min)	2.1 ± 0.1 (1.2)	3.2 ± 0.6 (1.6)	3.7 ± 0.2 (1.9)	4.5 ± 0.4 (2.4)	5.7 ± 1.0 (3.0)	6.9 ± 1.5 (3.4)
System Latency $\pm\sigma$ $[s]$	6.8 ± 2.3	9.3 ± 3.6	9.7 ± 2.6	11.1 ± 2.8	13.0 ± 3.1	14.3 ± 3.4
Time 95% of Nodes Completed $[s]$	3.4	4.8	5.8	7.1	8.7	10.1
PRR	$1 - 0.386$	$1 - 0.457$	$1 - 0.491$	$1 - 0.516$	$1 - 0.535$	$1 - 0.547$
$\mathrm{Throughput}_{\mathit{eff}}$ $[bit/s]$	968	13179	22505	27605	29216	30709
Sent Packets	81 ± 3	110 ± 20	115 ± 7	130 ± 12	151 ± 24	169 ± 34
Energy Consumption $[J]$	0.2168	0.2841	0.2849	0.3180	0.3621	0.3933

Table 7.9: Performance metrics for RLNC with varied payload size

Performance Metric	GC-1 B	GC-20 B (default)	GC-40 B	GC-60 B	GC-80 B	GC-100 B
$\text{RNP}_{100} \pm \sigma$ (min, max)	20221 ± 8030 (8540, 34675)	18420 ± 6023 (11218, 29566)	15766 ± 5526 (5933, 28135)	15878 ± 5136 (8766, 26682)	13240 ± 4742 (7546, 23383)	13306 ± 3564 (6452, 18485)
Node Latency $\pm\sigma$ $[s]$ (min)	126.5 ± 51.9 (52.8)	145.3 ± 48.2 (91.2)	151.8 ± 53.6 (63.6)	183.9 ± 55.1 (105.2)	178.4 ± 57.6 (100.7)	206.7 ± 52.1 (104.4)
System Latency $\pm\sigma$ $[s]$	126.5 ± 51.9	145.3 ± 48.2	151.8 ± 53.6	183.9 ± 55.1	178.4 ± 57.6	206.7 ± 52.1
Time 95% of Nodes Completed $[s]$	126.5	145.3	151.8	183.9	178.4	206.7
PRR	$1 - 0.342$	$1 - 0.379$	$1 - 0.394$	$1 - 0.399$	$1 - 0.400$	$1 - 0.399$
$\text{Throughput}_{\mathit{eff}}$ $[bit/s]$	19	310	610	725	1009	1067
Sent Packets	4864 ± 2073	5094 ± 1739	4806 ± 1772	5503 ± 1670	4912 ± 1840	5145 ± 1339
Energy Consumption $[J]$	4.0556	4.5382	4.6307	5.4860	5.2377	5.9688

Table 7.10: Performance metrics for GC with varied payload size

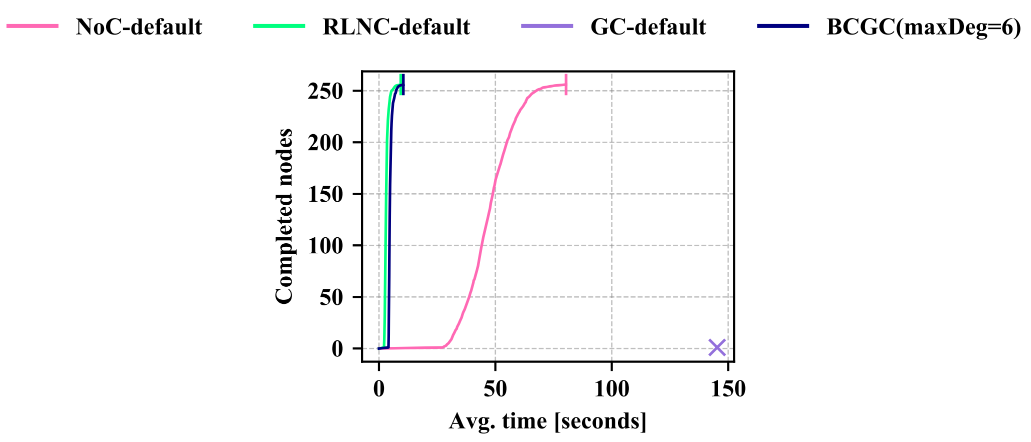


Fig. 7.38: Completed nodes for 20 Bytes payload

direct neighbors. By contrast, the values of the 'received' curve refer to single packets received by a node. Therefore, the curves for 'sent' and 'received' change their slope to a different extend. With a smaller payload size, more packets are sent by a node in a considered time interval. Therefore, sent and received packets contain/use more up-to-date information in the case of smaller packets than with larger packets due to the time delay caused by larger packets. As a result, the neighboring nodes receive more suitable packets/codewords in terms of degree and, in particular, in terms of symbol selection. Thus, the percentage of received distance-0-codewords is lower and the percentage of distance-1-codewords is correspondingly higher in the case of a smaller payload size, see Figures 7.30 and 7.31. As a consequence, the required number of received packets/codewords to reconstruct a respective number of symbols changes with the payload size in the case of BCGC, see Figure 7.29, due to the different points in time at which packets are sent. For BCGC, the RNP for each number of reconstructed symbols is lower with a smaller payload size. This cannot be recognized for the other compared procedures since they do not use any information about the neighborhood.

If the payload size is smaller, more packets are received per time interval and in the case of BCGC, also the RNP_{100} is lower. Therefore, the respective considered procedure has a lower node latency and system latency if the payload size is smaller, which can be seen in Figures 7.27, 7.32, 7.34, and 7.36. In general, the larger the payload size, the lower the slope of the latency-related curves, see Figures 7.27, 7.32, 7.34, and 7.36. This means that nodes in the network finish at increasingly different times as the payload size becomes bigger. This is due to the fact that nodes need a different number of received packets/codewords depending on their position in the network in order to reconstruct all symbols, see Section 7.1.1.1. In addition, symbols also have to overcome a different maximum distance to reach these nodes, which has already been explained in detail in

Section 7.1.1.1. A larger payload size and, thus, larger packets amplify both aspects accordingly, leading to flatter curves.

Figure 7.38 shows how the considered procedures behave in a direct comparison for a payload size of 20 Bytes. There, the important performance metric of latency is shown. For a comparison when using one of the other payload sizes, the corresponding separate plots can be compared with each other. When comparing the number of sent/received packets by a node for all applied procedures, for example for 20 Bytes, the ratio between the packet sizes of the respective procedures can be derived. BCGC have a similarly good performance as RLNC. However, node latency and system latency are slightly better for RLNC than for BCGC. Reasons for this are that the RNP_{100} is lower for RLNC and the packet headers are similar in percentage size for the considered $n = 256$ nodes/symbols, especially due to the 17-Byte-IEEE Std 802.15.4 header. The RNP in the case of RLNC is better than in the case of BCGC as symbols can be distributed faster in the network if a node always sends a large portion of the symbols it has already received. However, sending as many encoded symbols as possible in each packet is only beneficial if for example Gaussian elimination can be used for decoding, as for RLNC. In the case of BCGC, however, it is necessary to use a certain predetermined codeword degree for the sake of decodability as Gaussian elimination is not used here to comply with the constraints from Section 3.1. If more than $n = 256$ nodes were considered, the RLNC packet header would quickly become much larger than the BCGC packet header since RLNC does not scale well with respect to the number of nodes. In this case, RLNC would quickly be worse than BCGC in terms of node latency and system latency. A separate evaluation with a larger number of nodes can be found in Section 7.2.3. NoC performs considerably worse than BCGC in terms of latency since it has a considerably higher RNP_{100} due to the Coupon Collector's problem. Furthermore, the difference in header size is not significant, especially if *maxDeg* is chosen appropriately. Thus, the RNP_{100} has the main impact on node and system latency. For the considered default simulation settings with 20 Bytes of payload, the coding gain of BCGC with $maxDeg = 46$ compared to NoC is $\frac{9489}{827} \approx 11.47$, see Tables 7.4 and 7.8. Furthermore, the throughput gain compared to NoC is $\frac{7562}{865} \approx 8.74$. For BCGC with $maxDeg = 6$, as used for the evaluation with varied payload size, the coding gain compared to NoC is $\frac{9489}{844} \approx 11.24$ and the throughput gain is even $\frac{8510}{865} \approx 9.84$, see Tables 7.7 and 7.8. For a definition of the terms 'coding gain' and 'throughput gain', see Sections 3.3.1 and 3.3.5. Like NoC, GC also have a considerably worse latency than BCGC since they have a similar header size as BCGC and a significantly higher RNP_{100}. The higher RNP_{100} for GC can be explained by the chosen non-suitable degree for the default simulation setting with a low degree of nodes' mobility and by the fact that there is no specific symbol selection in GC. GC are designed for highly dynamic systems. These fulfill the properties of a perfect source scenario assumed for GC the better the higher the dynamics. Only in the case of such a scenario, GC can perform well. Therefore, the default simulation settings, see Section 6.2.2, are not suitable for GC. BCGC will always outperform GC in all aspects when using these settings. For this reason, GC are only considered for the evaluation with varied degree of nodes' mobility in Section 7.2.4 and are omitted for the other evaluations. Thus, only NoCoding and the commonly used related work, RLNC, are taken into account for comparison in the following.

7.2.2 Varied MAC Protocol

For the following evaluation, three different MAC protocols are used: two contention based MAC protocols, in which nodes access the medium randomly and collisions can be caused, and a schedule based MAC protocol, in which nodes are synchronized and scheduled so that no signal collisions occur. The first two protocols are a CSMA/CA based MAC protocol according to IEEE Std 802.15.4, see Section 6.1.3.1, and an ALOHA based MAC protocol with random transmission and no carrier sense, see Section 6.1.3.2. The third considered protocol is a TDMA based MAC protocol, see Section 6.1.3.3.

The choice of the MAC protocol influences the reliability of a transmission and the time delay between a first transmission attempt of a considered packet and its actual transmission. Packet collisions occur with both, the CSMA/CA based MAC protocol and the ALOHA based MAC protocol. However, under similar conditions, significantly more collisions can occur with the ALOHA based protocol as it does not perform carrier sense before transmission. For the CSMA/CA based MAC protocol, the duration of a cycle with the states *BEB*, *CCA*, *RX-to-TX*, *TX*, *TX-to-RX*, and *RX*, see Sections 6.1.3.1 and 6.2.2, within which a first transmission attempt is made is approximately 24 ms for a BCGC packet with a degree-1-codeword and 20 Bytes of payload. With a degree-46-codeword, the duration is approximately 27 ms. For this calculation, the duration of the individual states as indicated in Section 6.1.3.1 and the transmission time of a respective BCGC packet were used. In the case of an NoC or RLNC packet, the duration is about 24 ms or 25 ms, respectively. Using the ALOHA based MAC protocol, the time at which the next packet is to be sent by a considered node is randomly chosen from a predefined interval. The states *RX-to-TX*, *TX*, *TX-to-RX*, and *RX* existing in the ALOHA based protocol then lie within this interval I. Thus, the choice of the interval length determines the collision rate. For the following evaluation, an interval length of $|I| = 26.5$ ms is used. This interval length was chosen to be between the interval length as for a first transmission attempt with CSMA/CA for the smallest possible BCGC packet and for the largest possible BCGC packet for the default configuration. The conditions for the CSMA/CA based MAC protocol and the ALOHA based MAC protocol which are considered here for comparison are therefore similar. With the implemented TDMA based MAC protocol, there are no packet collisions caused by signal collisions since each node is assigned a fixed transmission time slot of equal length and only one node transmits at a time.

The more collision-prone a MAC protocol is, the greater the challenge when using large packets. Larger packets are generally more affected by the collision rate and effects, such as increased latency, are thus amplified per required transmission/hop. The packet size can be large on the one hand due to the payload size but on the other hand also due to a large header. In the following, however, the default simulation settings from Section 6.2.2 are used, i.e. the payload size is small and the header sizes do not differ greatly since only $n = 256$ nodes are considered. For RLNC, the header size increases with the number of nodes/distinct symbols in the system. Therefore, when using the same interval length for all compared procedures and a larger number of nodes, the collision rate for RLNC increases significantly with the number of nodes when using the ALOHA based protocol. As a consequence, the performance of RLNC would quickly be worse than that of BCGC if a larger number of nodes is considered, especially with the ALOHA based MAC protocol.

For NoC and RLNC, the used MAC protocol and, in particular, the collision rate do not change any aspect of the respective procedure itself. However, for BCGC, estimates are made about which symbols have not been received by neighbors yet. The accuracy of such estimations may, hence, be influenced by the collision rate. It is, therefore, relevant to evaluate whether BCGC can cope with a higher collision rate. With a lower collision rate, transmitted packets are more likely to arrive at each neighbor, so that counters for the specific symbol selection are less distorted. Thus, more accurate information is used for the creation of packets in BCGC. Then, less packets/codewords may have to be received to reconstruct a respective number of symbols. Therefore, the RNP is compared for the different MAC protocols in Figure 7.41 for BCGC. Furthermore, the MAC protocol determines the time interval between a node's first transmission attempt for a considered packet and the actual transmission of this packet. With the ALOHA based and the TDMA based MAC protocol, transmission takes place immediately as soon as an attempt to transmit is made. Then, packets are received by the neighbors with a certain probability, depending on the collision rate, for example. By contrast, with the CSMA/CA MAC protocol, there are several transmission attempts of the same packet if the channel is occupied at the time of the respective attempt to transmit the considered packet. As a consequence, there is a delay between the creation of a packet at the time of the first attempt to transmit this packet and the actual transmission/possible reception of this packet. For NoC and RLNC this does not cause any difference for the procedure itself. However, for BCGC, this does have an impact on the procedure. If a packet is sent after a certain delay, as with the CSMA/CA MAC protocol, the packet does not contain the most recent information. It may, thus, be less suitable for the neighboring nodes than a packet composed at the current time would be. Consequently, the delay may lead to an increase in the RNP. The extent to which a particular MAC protocol, and in particular such a delay, influences the RNP in BCGC is therefore analyzed in the following evaluations and shown in Figure 7.41.

In addition to the RNP, the number of received packets per considered time interval also affects the important performance metrics of node and system latency. The number

Performance Metric	BCGC-CSMA/CA (default)	BCGC-ALOHA	BCGC-TDMA
$\mathrm{RNP}_{100} \pm\sigma$ (min, max)	827 ± 85 (277, 3213)	815 ± 58 (276, 2925)	781 ± 153 (264, 2345)
Node Latency $\pm\sigma$ $[s]$ (min)	5.5 ± 0.9 (3.2)	6.8 ± 0.7 (4.2)	106.9 ± 18.7 (69.9)
System Latency $\pm\sigma$ $[s]$	11.4 ± 2.9	13.0 ± 3.4	241.1 ± 47.7
Time 95% of Node Completed $[s]$	8.0	9.0	155.2
PRR	$1 - 0.341$	$1 - 0.615$	-
$\mathrm{Throughput}_{eff}$ $[bit/s]$	7562	6044	391
Sent Packets	202 ± 27	261 ± 26	99 ± 17
Energy Consumption $[J]$	0.3488	0.4243	8.8901

Table 7.11: Performance metrics for BCGC with varied MAC protocol

of received packets per considered time interval and, thus, latency depends on the number of sent packets per time interval and on the collision rate. These can be different for each MAC protocols considered here. Therefore, in Figures 7.40, 7.43, and 7.45 and Figures 7.39, 7.42, and 7.44 the sent/received packets and the latency for the different MAC protocols are compared for a respective procedure. A higher number of sent packets per considered time interval can result from the use of the ALOHA based MAC protocol instead of the CSMA/CA based protocol, for example. Alternatively, this can also be achieved with the ALOHA based protocol by using a smaller time interval instead of the time interval specified here. For the same RNP, the resulting latency can be better with a higher number of transmitted packets per considered time interval as long as the collision rate is not significantly increased. Latency is only improved if the number of received packets is increased. With a higher number of transmitted packets per considered time interval, the collision rate can also be very high so that the resulting number of received packets per interval is deceased. In this case, latency becomes worse compared to a larger chosen time interval. However, the goal here is not to find an optimal interval length for BCGC, RLNC, or NoC for the ALOHA based MAC protocol. Instead, the results of a considered procedure are to be compared for the ALOHA based protocol with the chosen interval length $|I|$, for the CSMA/CA based MAC protocol and for the TDMA based MAC protocol. In addition, NoC, RLNC, and BCGC shall be compared against each other when using the ALOHA based MAC protocol and the TDMA based MAC protocol. A comparison with the CSMA/CA based MAC protocol can already be found in Figure 7.38. For a tabular overview of the values of relevant performance metrics resulting from varying the used MAC protocol, see Tables 7.11 to 7.13.

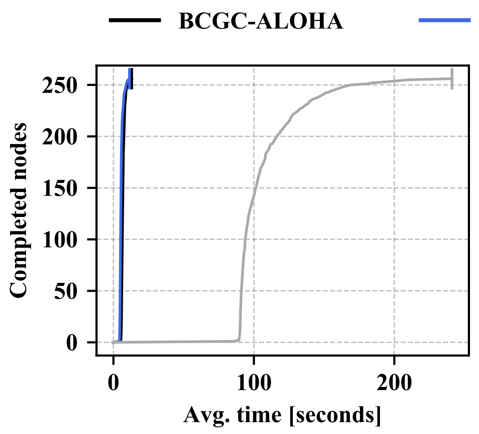


Fig. 7.39: Completed nodes

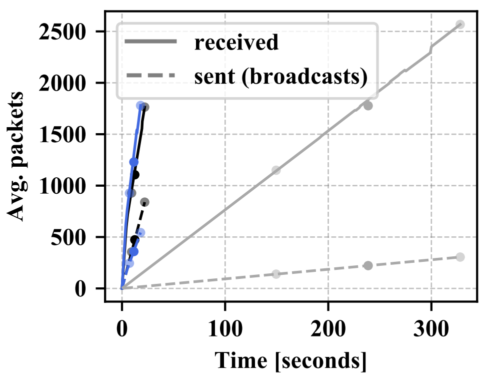


Fig. 7.40: Sent/received packets

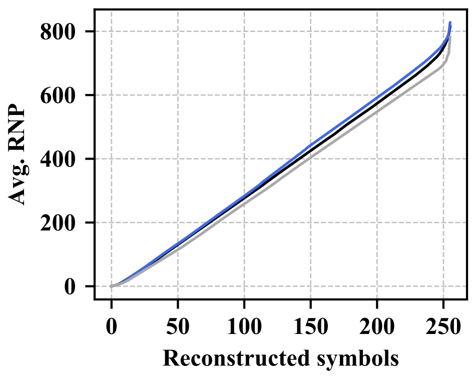


Fig. 7.41: RNP

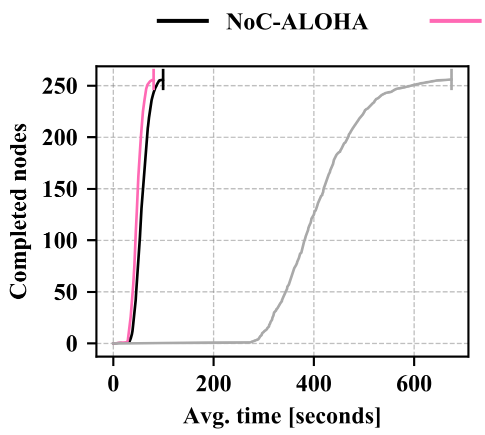


Fig. 7.42: Completed nodes

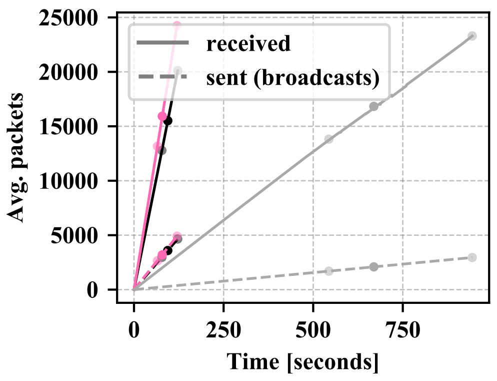


Fig. 7.43: Sent/received packets

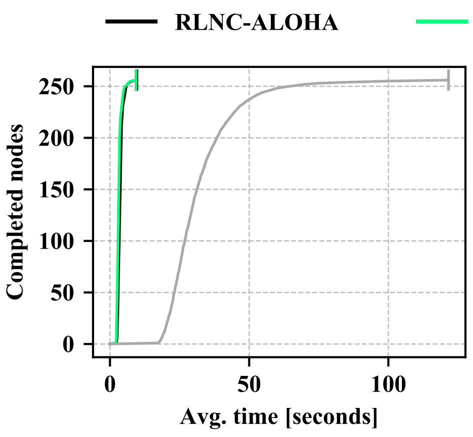


Fig. 7.44: Completed nodes

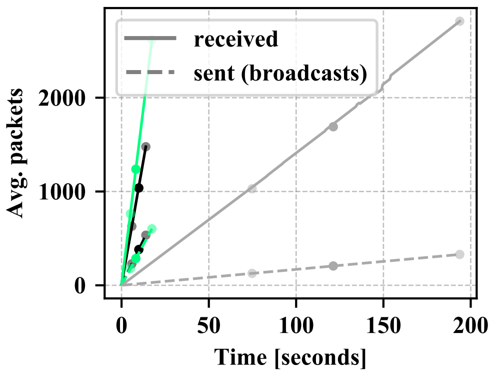


Fig. 7.45: Sent/received packets

Performance Metric	NoC-CSMA/CA (default)	NoC-ALOHA	NoC-TDMA
RNP_{100} $\pm\sigma$ (min, max)	9489 ± 483 (3742, 17881)	9275 ± 1008 (3501, 21771)	10184 ± 828 (2830, 21429)
Node Latency $\pm\sigma$ $[s]$ (min)	47.5 ± 2.5 (23.7)	57.3 ± 5.9 (20.7)	411.5 ± 32.7 (234.6)
System Latency $\pm\sigma$ $[s]$	80.4 ± 13.1	99.3 ± 14.1	674.5 ± 101.4
Time 95% of Nodes Completed $[s]$	64.4	80.9	554.7
PRR	$1 - 0.302$	$1 - 0.482$	-
$Throughput_{eff}$ $[bit/s]$	865	721	100
Sent Packets	1943 ± 101	2185 ± 225	1286 ± 102
Energy Consumption $[J]$	2.6410	3.0903	24.8560

Table 7.12: Performance metrics for NoC with varied MAC protocol

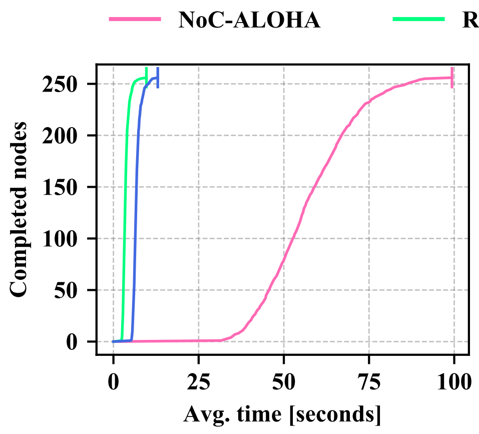


Fig. 7.46: Completed nodes

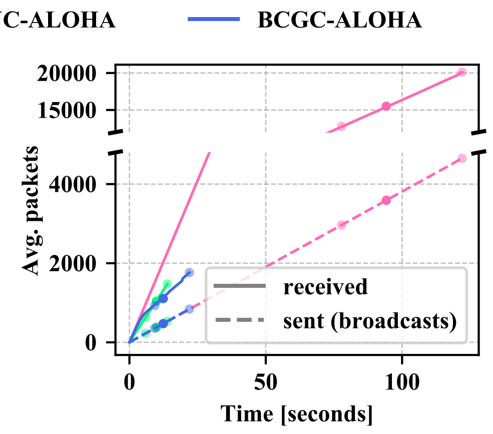


Fig. 7.47: Sent/received packets

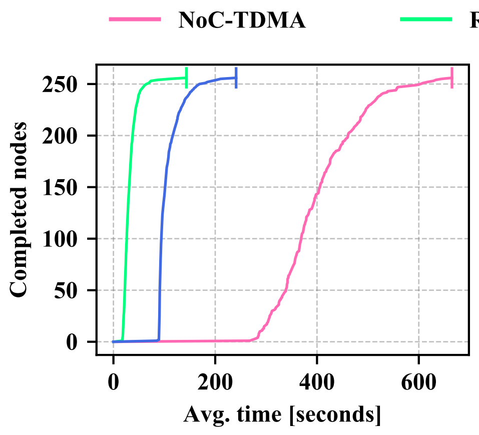


Fig. 7.48: Completed nodes

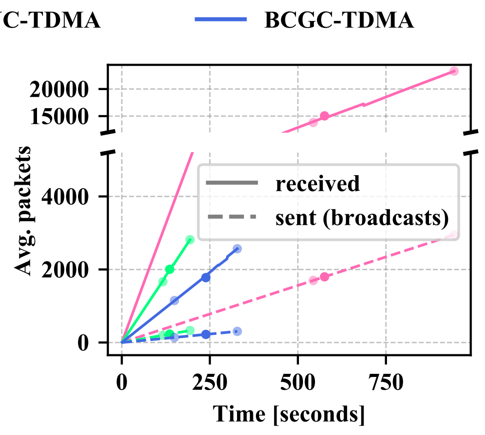


Fig. 7.49: Sent/received packets

Performance Metric	RLNC-CSMA/CA (default)	RLNC-ALOHA	RLNC-TDMA
RNP_{100} $\pm\sigma$ (min, max)	460 ± 92 (257, 1893)	369 ± 11 (257, 895)	402 ± 28 (262, 1044)
Node Latency $\pm\sigma$ $[s]$ (min)	3.2 ± 0.6 (1.6)	3.6 ± 0.1 (2.2)	32.5 ± 1.8 (15.3)
System Latency $\pm\sigma$ $[s]$	9.3 ± 3.6	9.7 ± 2.3	121.6 ± 32.4
Time 95% of Nodes Completed $[s]$	4.8	5.4	54.9
PRR	$1-0.457$	$1-0.671$	-
Throughput_{eff} $[bit/s]$	13179	11360	1265
Sent Packets	110 ± 20	138 ± 4	55 ± 3
Energy Consumption $[J]$	0.2841	0.3623	4.4835

Table 7.13: Performance metrics for RLNC with varied MAC protocol

In the following, the results are interpreted:
In contrast to the CSMA/CA based MAC protocol, no carrier sense is performed with the ALOHA based protocol and packet transmissions are not postponed/skipped. Therefore, the ALOHA based MAC protocol which uses the interval length $|I|$, i.e. similar conditions to the CSMA/CA protocol, has a significantly higher collision rate than the CSMA/CA based protocol for each considered procedure. The TDMA based MAC protocol has the lowest collision rate as only one node is transmitting at a time, i.e. there are no concurrent transmissions. The differences in collision rate can be recognized in Figures 7.40, 7.43, and 7.45 by comparing the number of sent packets with the number of received packets at any point in time in each case. According to the increase of the collision rate, more packets have to be sent to receive the same number of packets/codewords and, in particular, until a receiver can reconstruct all distinct original symbols

Since the MAC protocol does not change the procedure for NoC and RLNC, no change in the required number of received packets to reconstruct a respective number of symbols is to be expected. In contrast, in the case of BCGC, the MAC protocol influences the procedure itself by the respective collision rate and by the time between the creation of a packet and the actual transmission/possible reception of this packet. For BCGC, the collision rate increases in the course of the data dissemination process for the ALOHA based MAC protocol and for the CSMA/CA based protocol due to the increasing BCGC packet size. In addition, the TDMA based MAC protocol has a lower collision rate than the CSMA/CA based protocol, which in turn has a lower collision rate than the ALOHA based protocol. The number of times a respective symbol has been sent/received is taken into account for the composition of codewords in the case of BCGC. With a lower collision rate, transmitted packets are more likely to arrive and, thus, more accurate information is used for the creation of BCGC packets. As a consequence, less packets/codewords have to be received to reconstruct a respective number of symbols in the case of a lower collision rate, which can be seen when comparing the ALOHA based protocol with the TDMA based protocol in Figure 7.41. Despite the significantly higher collision rate for the ALOHA based protocol, the difference in the RNP is not particularly large. When composing a codeword, a symbol which has never or least frequently been transmitted by the node is specifically selected for the codeword. The resulting ranking of the symbols will deviate from reality by an increased collision rate only if the collision rate is so high that a packet does not arrive at any neighbor. If there are several symbols which were sent equally rarely by the node, then these are sorted in a second stage according to the number of times they were received. This is influenced by the collision rate and, thus, explains the only slightly different RNP between the TDMA based and the ALOHA based protocol. When using the default CSMA/CA MAC protocol, another aspect that influences the results is the postponement of the transmission of a packet in the case of a busy channel. A packet is created directly before the first attempt to transmit this packet. With the ALOHA based MAC protocol and the TDMA based protocol, the packet is sent immediately and was, therefore, created using the latest information at the time of transmission. By contrast, with the CSMA/CA MAC protocol, the packet is sent after a certain delay if the medium is found to be busy. In the case of a delay, a transmitted packet does not contain the most recent information. Thus, it may be less suitable for the respective receiver than a packet composed at the actual time of transmission would have been. As a consequence, each delay can increase the RNP. For this reason and due to the higher collision rate when using the CSMA/CA based MAC protocol, it can be seen in Figure 7.41 that the TDMA based MAC protocol has a lower RNP than the CSMA/CA based protocol. Despite the higher collision rate for the ALOHA based MAC protocol than for the CSMA/CA based protocol, the RNP is lower for the ALOHA based protocol. Thus, it can be seen that the impact of an increased collision rate on the RNP, i.e. less accurate counters, is smaller than the impact of a delay in the packet transmission, i.e. outdated information. Especially in the course of the dissemination process, when BCGC packets become larger and the frequency and duration of delays in sending a created packet increase, the difference in RNP is more noticeable. Overall, however, neither the very high collision rate nor the delays lead to a significant change in the RNP for BCGC. To conclude, even with a very high collision rate or frequent/longer delays in the packet transmission, the RNP does not change significantly in the case of BCGC. BCGC can, thus, also be used in the case of low reliability of the individual transmissions. In particular, despite the accumulated information, BCGC does not require more reliability than NoC.

In the case of NoC, the slope of the 'sent-curve' is slightly higher for the CSMA/CA based protocol than for the ALOHA based MAC protocol, see Figure 7.43. For RLNC and BCGC, this is exactly the opposite, see Figures 7.40 and 7.45. This is due to the fact that for NoC, $|I|$ is slightly larger for the ALOHA based MAC protocol than the duration of a cycle for the CSMA/CA protocol, which is influenced by the exact packet size. In addition, with the small NoC packets and the time t_{RX}, for which a node is in receive mode, see Section 6.2.2, the channel is less frequently occupied at the first transmission attempt of a considered packet. Therefore, a transmission is less likely to be postponed/skipped than with larger packets. Thus, slightly more packets are sent per considered period of time with the CSMA/CA based protocol than with the ALOHA based MAC protocol, as shown in Figure 7.43. RLNC packets, on the other hand, are larger so that a packet transmission is more frequently postponed/skipped with the CSMA/CA based protocol. The ALOHA based MAC protocol does not check the medium for ongoing transmissions and postpones transmissions if necessary but transmits as soon as the randomly chosen time at which the next packet is to be sent is reached. Therefore, fewer RLNC packets are sent per considered time interval with the CSMA/CA based MAC protocol than with the ALOHA based protocol, which can be seen in Figure 7.45. With BCGC, packets are initially very small and grow in size during the dissemination process. Therefore, in the beginning, the duration of a cycle for the CSMA/CA protocol is slightly shorter than $|I|$ and becomes slightly larger than $|I|$ in the course of the dissemination process. In addition, with BCGC using the CSMA/CA MAC protocol, packet transmissions are postponed/skipped more frequently the larger packets become. Therefore, in the case of BCGC, the slope of the 'sent-curve' for the CSMA/CA based protocol even decreases during the dissemination process, see Figure 7.40. Thus, with the ALOHA based protocol more packets are sent per considered time interval than with the CSMA/CA based protocol. When using the TDMA based MAC protocol, the least number of packets are sent per period of time, as only one node after the other sends its packet, so there are no simultaneous transmissions, see Figures 7.40, 7.43, and 7.45

In principle, more packets could be received per considered time interval if more packets are sent. However, the collision rate with the ALOHA based MAC protocol under the conditions considered here is significantly higher than with the CSMA/CA based protocol. Therefore, fewer packets are received per considered period of time with ALOHA than with CSMA/CA, see Figures 7.40, 7.43, and 7.45. For BCGC, a change in the slope of the 'received-curve' can be seen, since less packets are sent and additionally more collisions occur in the course of the data dissemination process due to the increasing packet size. The lower number of received packets per considered time interval when using the ALOHA based MAC protocol results in node and system latency being higher with the ALOHA based MAC protocol than with the CSMA/CA based MAC protocol, see Figures 7.39, 7.42, and 7.44. With the TDMA based MAC protocol, significantly less packets are sent per considered time interval and the associated collision rate is also correspondingly lower. However, in the end, fewer packets are received per time interval than with the ALOHA based and the CSMA/CA based protocol due to the very low number of packets sent per considered time interval. As a consequence, node latency and system latency are significantly increased by not using concurrent transmissions, as with the TDMA based MAC protocol, see Figures 7.39, 7.42, and 7.44. In addition, the same fixed length of all time slots in the TDMA based protocol has a negative impact on the performance of BCGC. When using the TDMA based MAC protocol, the size of a slot has to be adapted to the maximum occurring packet size of the considered procedure. This eliminates a major advantage of BCGC: packets can be small for a long time in the beginning and only become larger towards the end of a data dissemination process. When using the implemented TDMA based MAC protocol, it can therefore be reasonable to choose a lower maximum possible degree *maxDeg* so that the length of a time slot is shorter. However, this would require a separate evaluation since a lower *maxDeg* causes that more packets may have to be received. With the other two MAC protocols, the advantage of BCGC of initially short packets remains. Furthermore, a slightly lower slope in the latency-related curves for the ALOHA based MAC protocol than for the CSMA/CA based protocol can be seen, especially for NoC in Figure 7.42. This is caused by the fact that the differences that exist for all nodes due to their respective position in the network are amplified by the lower number of received packets per time interval for each node. A similar correlation has already been recognized and explained for the different payload sizes in Section 7.2.1. There, the slope of the latency-related curves was lower the larger the payload size was. In contrast to NoC, the difference between the slopes for the ALOHA based MAC protocol and for the CSMA/CA based protocol is not noticeable in the case of RLNC and BCGC. For the TDMA based MAC protocol compared to the ALOHA based and the CSMA/CA based protocol, this difference is clearly visible for all procedures, see Figures 7.39, 7.42, and 7.44.

If NoC, RLNC, and BCGC are compared with each other using the collision-prone ALOHA-based MAC protocol, it can be seen in Figure 7.47 that all procedures send approximately the same number of packets per considered period of time as all procedures use the same interval length $|I|$. Due to the smaller packet sizes for NoC, NoC causes significantly fewer collisions than BCGC or RLNC and, thus, more packets are received per time interval with NoC. However, due to the Coupon Collector's problem in the case of NoC, the corresponding RNP_{100} is significantly larger than for BCGC or RLNC. As a consequence, node latency and system latency are significantly worse for NoC than for BCGC and RLNC, see Figure 7.46. BCGC packets are initially small and the packet size increases in the course of the dissemination process. Thus, for the considered number of nodes/symbols $n = 256$ and the corresponding RLNC header size, more packets per time interval are received in the beginning with BCGC than with RLNC and fewer packets are received later. However, here, the difference in RNP_{100} between BCGC and RLNC is too high so that the only initially higher number of received packets per time interval with BCGC cannot compensate for this difference. Thus, node and system latency for RLNC are better than for BCGC, see Figure 7.46. However, with a larger n and a, thus, quickly significantly larger RLNC header size, the collision rate for RLNC can easily exceed the influence of the RNP_{100} difference if an equal interval length $|I|$ is used for all procedures. In this case, the latency would become lower for BCGC than for RLNC.

If an own suitable interval length $|I|$ is used for each procedure, the interval length would be correspondingly larger for RLNC in the case of a larger number of nodes/larger RLNC packets. Then, fewer packets could be sent and, thus, received per considered period of time with RLNC than with BCGC. This can quickly increase the latency for RLNC compared to BCGC. These changes due to a larger number of nodes/number of distinct original symbols will be encountered in both the ALOHA based MAC protocol and the CSMA/CA based MAC protocol. For the CSMA/CA MAC protocol and RLNC, either the collision rate would be significantly increased or the time t_{RX}, for which a node is in receive mode, would have to be increased in the case of a larger number of nodes. Increasing t_{RX} implies that correspondingly fewer packets are sent/received per considered time interval. A comparison of NoC, RLNC, and BCGC using the CSMA/CA based MAC protocol can already be found in Figure 7.38. When comparing NoC, RLNC, and BCGC for the TDMA based MAC protocol, it can be seen that the smaller the packets are, the more packets can be sent per time interval, see Figure 7.49. For BCGC, the duration of a TDMA-round, i.e. the time needed for one node after the other to send a packet, has been adjusted to the maximum packet size using the default upper bound $maxDeg = 46$. Thus, BCGC has the largest packets among the considered procedures. A lower chosen *maxDeg* would reduce the duration of a round and, thus, improve latency for BCGC. However, the default values should be used here rather than looking for an optimum. With a larger number of nodes/symbols than the default value $n = 256$, RLNC packets quickly become significantly larger so that the duration of a round is correspondingly longer and, thus, latency is significantly higher. Due to the respective packet sizes, more packets are sent and, thus, received per time interval with NoC than with RLNC and again more packets with RLNC than with BCGC. However, the value RNP_{100} is significantly higher for NoC than for RLNC or BCGC and the higher number of received packets per considered period of time cannot compensate this difference in the case of $n = 256$ nodes. Therefore, the latency in the case of NoC is worse than the latency for RLNC or BCGC when using the TDMA based MAC protocol, see Figure 7.48. With the default number of nodes $n = 256$ and the default upper bound $maxDeg = 46$, the latency for BCGC is worse than for RLNC when using the TDMA based protocol, as shown in Figure 7.48. This is due to the fact that with BCGC both RNP_{100} is higher and fewer packets are received per considered time interval.

7.2.3 Varied Number of Nodes/Symbols

For the following evaluation of BCGC, the results obtained by using the default number of nodes, $n = 256$, are compared with those obtained by using a much higher number of nodes, $n = 1024$. In the case of BCGC, the considered number of nodes does not change the size of the packet header. Instead, the size of the BCGC packet header is determined by the chosen upper bound for the codeword degree, *maxDeg*. For a larger number of nodes, a different upper bound may be more appropriate than the optimal upper bound determined for a smaller number of nodes. Here, however, it is not the aim to determine optimal upper bounds but to observe how the BCGC procedure behaves in principle with a larger number of nodes than the default value. For this reason, the default upper bound *maxDeg* $= 46$ is used here, which still complies with the packet size specifications of IEEE Std 802.15.4 for the used default payload size of 20 Bytes. The larger number of nodes increases the node density in the network compared to the default number of nodes. The original GC degree function assumes that all n symbols are always available for the creation of a codeword in a node and, then, determines the current optimal degree according to the formulas from Section 5.2.1, which depend on n. Therefore, the GC degree function adapts itself to a changed given number of nodes n and increases the degree with a larger number of distinct original symbols n accordingly later. This corresponds indirectly to an adaptation to the changed node density since with a higher node density, the degree also has to be increased correspondingly later. The used BCGC degree function adapts analogously to the GC degree function to the changed number of nodes and, thus, also to the changed node density. However, no optimal degree function for a considered node density is to be searched for here. Instead, only the BCGC procedure itself is to be evaluated with a different number of nodes and, thus, a different node density. In general, it shall be shown that BCGC provide good results even with a large number of nodes and have significant advantages over the frequently used RLNC in this case.

Unlike for NoC and BCGC, the number of nodes/distinct original symbols changes the size of the RLNC packet header. This means, RLNC packets become larger so that it takes longer to transmit a single RLNC packet in the case of $n = 1024$ nodes than in the case of $n = 256$. As a consequence, in the case of RLNC, significantly more transmissions may have to be postponed with the default MAC protocol with $n = 1024$ nodes than with $n = 256$ nodes. Thus, less packets are sent per considered time interval. Moreover, more collisions may be caused. For the $n = 1024$ nodes considered here as an alternative to the default number of nodes, an RLNC header based on $GF(2)$ is already $n/8 + 17 = 145$ Bytes in size. RLNC can, therefore, not comply with the packet size restrictions of the IEEE Std 802.15.4 with this number of nodes, regardless of the chosen payload size. With the default payload size of 20 Bytes and $n = 1024$ nodes, an RLNC packet has a size of 165 Bytes. In general, RLNC does not scale well with the number of nodes/distinct original symbols and is practically not applicable for the $n = 1024$ nodes considered here. Nevertheless, RLNC will be used for comparison with BCGC in the following.

Due to the higher node density in the case of $n = 1024$, it is expected that the medium is found to be busy more often. Thus, more transmissions have to be postponed. Therefore, fewer packets can be sent per considered time interval with $n = 1024$ than with $n = 256$ due to the changed node density. In addition, more packet transmissions may collide in the case of a higher node density. With $n = 1024$ instead of $n = 256$, each node is expected to have more neighbors. Hence, if the number of postponements and

Performance Metric	BCGC-256 (default)	BCGC-1024
RNP_{100} $\pm\sigma$ (min, max)	827 ± 85 (277, 3213)	2361 ± 91 (1647, 3576)
Node Latency $\pm\sigma$ $[s]$ (min)	5.5 ± 0.9 (3.2)	15.9 ± 1.4 (11.8)
System Latency $\pm\sigma$ $[s]$	11.4 ± 2.9	17.3 ± 0.8
Time 95% of Nodes Completed $[s]$	8.0	16.5
PRR	$1-0.341$	$1-0.427$
$\text{Throughput}_{\mathit{eff}}$ $[bit/s]$	7562	10411
Sent Packets	202 ± 27	329 ± 17
Energy Consumption $[J]$	0.3488	0.4378

Table 7.14: Performance metrics for BCGC with varied number of nodes

the number of signal collisions are not significantly increased, nodes may receive more packets per considered time interval with $n = 1024$ than with $n = 256$. However, the number of received packets can also be reduced if significantly fewer packets are sent per time interval due to postponements or too many collisions are caused. Therefore, the average number of packets sent/received by a node is shown in Figures 7.51, 7.53, and 7.55.

A larger number of nodes implies that more distinct original symbols have to be reconstructed in each node. Thus, more packets/codewords have to be received to be able to reconstruct this increased number of symbols. As a consequence, the time required for a node to be completed may increase for each of the considered procedures, but especially for RLNC with its larger packets. However, the possibly increased number of received packets per considered time interval may in turn improve latency. Therefore, the resulting node latency and system latency for the different number of nodes is shown in Figures 7.50, 7.52, and 7.54 for NoC, RLNC, and BCGC. A tabular overview of the values of relevant performance metrics resulting from varying the number of nodes/distinct original symbols in the system can be found in Tables 7.14 to 7.16.

In the following, the results are interpreted:
For all compared procedures, Figures 7.51, 7.53, and 7.55 show that fewer packets are sent as broadcast transmission to all direct neighbors per considered time interval for $n = 1024$ due to the increased node density. For RLNC, the difference is even more noticeable as a result of the additional larger packet sizes. The average number of received packets in a node per considered time interval is larger for $n = 1024$ than for $n = 256$ in the case of NoC and BCGC. This can be explained by the significantly increased number of neighbors, the only small reduction in the number of sent packets, and the not greatly increased collision rate with the larger number of nodes/higher

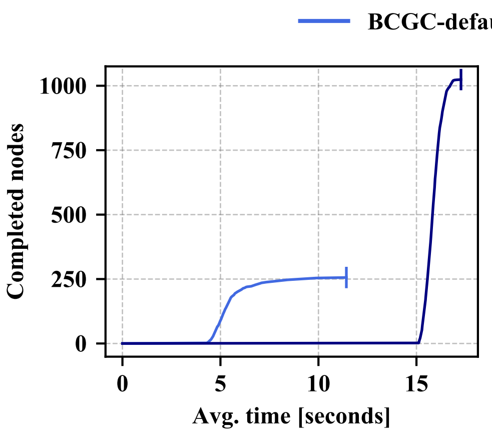


Fig. 7.50: Completed nodes

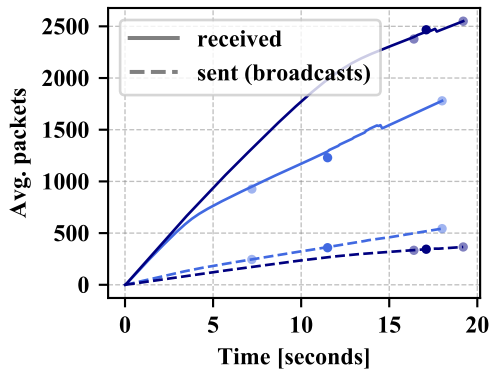


Fig. 7.51: Sent/received packets

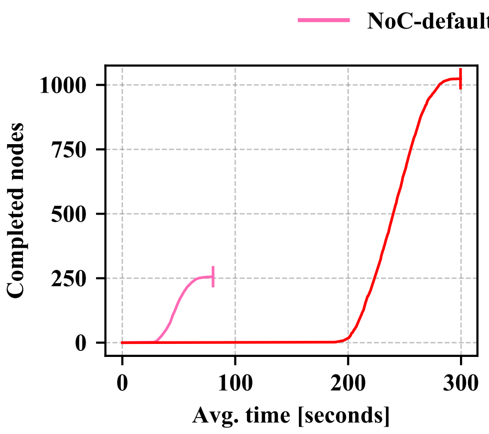


Fig. 7.52: Completed nodes

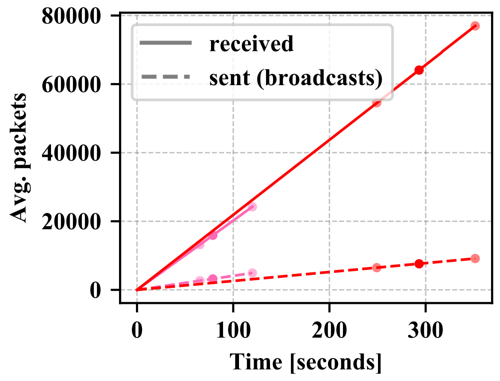


Fig. 7.53: Sent/received packets

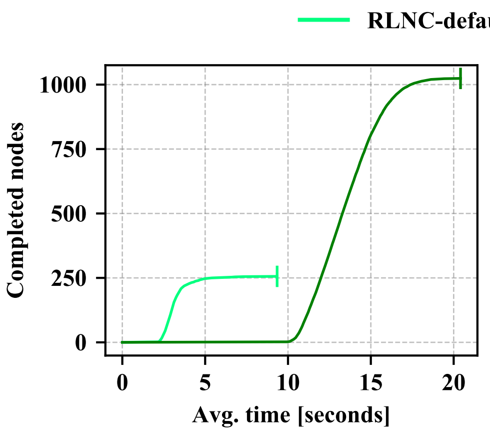


Fig. 7.54: Completed nodes

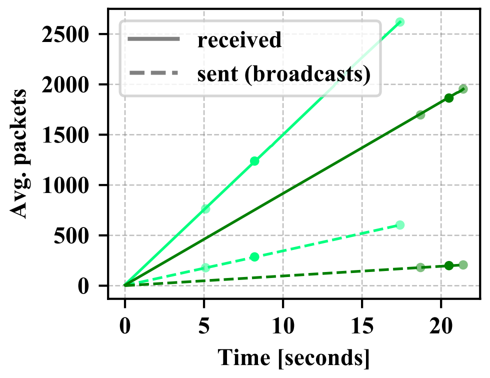


Fig. 7.55: Sent/received packets

Performance Metric	NoC-256 (default)	NoC-1024
RNP_{100} $\pm\sigma$ (min, max)	9489 ± 483 (3742, 17881)	52632 ± 9018 (27987, 79948)
Node Latency $\pm\sigma$ $[s]$ (min)	47.5 ± 2.5 (23.7)	240.4 ± 41.3 (131.6)
System Latency $\pm\sigma$ $[s]$	80.4 ± 13.1	299.5 ± 34.9
Time 95% of Nodes Completed $[s]$	64.4	276.4
PRR	$1 - 0.302$	$1 - 0.428$
$\text{Throughput}_{\mathit{eff}}$ $[bit/s]$	865	700
Sent Packets	1943 ± 101	6251 ± 1072
Energy Consumption $[J]$	2.6410	7.9767

Table 7.15: Performance metrics for NoC with varied number of nodes

Performance Metric	RLNC-256 (default)	RLNC-1024
RNP_{100} $\pm\sigma$ (min, max)	460 ± 92 (257, 1893)	1222 ± 15 (1024, 1965)
Node Latency $\pm\sigma$ $[s]$ (min)	3.2 ± 0.6 (1.6)	13.5 ± 0.1 (9.7)
System Latency $\pm\sigma$ $[s]$	9.3 ± 3.6	20.4 ± 1.0
Time 95% of Nodes Completed $[s]$	4.8	16.8
PRR	$1 - 0.457$	$1 - 0.698$
$\text{Throughput}_{\mathit{eff}}$ $[bit/s]$	13179	12145
Sent Packets	110 ± 20	130 ± 1
Energy Consumption $[J]$	0.2841	0.461

Table 7.16: Performance metrics for RLNC with varied number of nodes

density, see Figures 7.51 and 7.53. Due to the larger RLNC packet sizes for $n = 1024$ compared to $n = 256$ and the additional higher node density, significantly fewer RLNC packets are sent per considered time interval and a higher collision rate is caused. In contrast to NoC and BCGC, this cannot be completely compensated for in the case of RLNC by the larger number of neighbors due to the big difference. As a consequence, for RLNC, the number of received packets per considered time interval is reduced for $n = 1024$ nodes compared to $n = 256$ nodes, see Figure 7.55.

As already described, in the case of $n = 1024$ nodes, more symbols have to be reconstructed by each node and, thus, more packets have to be received per node to be completed than with $n = 256$. This cannot be completely compensated here even by an increased number of received packets per considered time interval, as is the case for NoC and BCGC. For RLNC, the situation is further worsened by the reduced number of received packets per considered time interval. Therefore, the node and system latency is higher for the larger number of nodes for all compared procedures under the considered default simulation setting and with the default payload size of 20 Bytes, see Figures 7.50, 7.52, and 7.54. Due to the lower number of received packets per time interval, the node and system latency is significantly increased in the case of RLNC by the larger number of nodes. As a consequence, latency for RLNC is worse than for BCGC despite the lower RNP_{100} in the case of RLNC, as shown in Figures 7.50 and 7.54. The larger the number of nodes, the stronger this effect will be since RLNC packets do not scale well. RLNC packet sizes grow with the number of nodes/distinct original symbols in the system. For NoC, the inherent Coupon Collector's problem is amplified by the larger number of nodes, so that significantly more packets have to be received to reconstruct all distinct original symbols than in the case of $n = 256$ nodes. As a result, the latency for a network with $n = 1024$ nodes is eventually much worse for NoC than for RLNC and especially than for BCGC.

Regarding the long-tail problem, for BCGC, the time at which the first node finishes and the time at which the last node finishes are much closer together for $n = 1024$ nodes than for $n = 256$ nodes, see Figure 7.50. In the case of BCGC, the latency related curve has a greater slope for the larger number of nodes, and there are not a few nodes that take significantly longer than the other nodes. The increased node density in the network ensures that there are no nodes that are particularly poorly connected. This is even the case for the nodes at the edge. Thus, nodes finish more evenly and the long-tail problem is mitigated. However, as is the case for RLNC or NoC, there may be other factors that worsen the result again and, thus, flatten the slope of the latency-related curve. These can be, for example, a reduced number of received packets per time interval, as for RLNC, or an increased RNP_{100}, as for NoC. Especially BCGC benefit from the increased node density since, then, specifically selected symbols can be useful for a larger number of nodes. In addition, here, the number of received packets per considered time interval is larger with $n = 1024$ for BCGC, unlike for RLNC, without having to receive as many packets as in the case of NoC. Due to the larger number of symbols to be reconstructed in the case of $n = 1024$ nodes, more BCGC packets have to be received than with $n = 256$. However, nodes can receive/reconstruct more BCGC packets/symbols per considered time interval due to the increased node density. Therefore, latency is not increased to the same extent for BCGC as the number of nodes, see Figure 7.50.

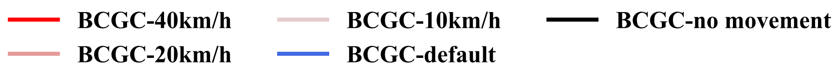
7.2.4 Varied Degree of Mobility

In the following evaluation, different degrees of mobility are considered. A mobile network which uses the time-based Random Walk mobility model is assumed in the case of mobile nodes, as specified in Section 6.2.2 as default setting. Further mobility models which are implemented in the simulator and described in Section 6.1.5 were analyzed in detail in the master thesis [Stö17]. There, similar results were obtained for BCGC with all implemented mobility models. Thus, only the default, time-based Random Walk mobility model or no movement of the nodes are considered for evaluation in Sections 7.1 and 7.2.

Here, the impact of five different degrees of mobility on NoC, RLNC, GC, and BCGC are compared: First, a static network with no movement of the nodes. Second, a slow to medium speed of $0.75 - 1.5m/s \approx 2.5 - 5km/h$, which represents the speed of a human who walks at slow to medium pace and is used as default speed here, see Section 6.2.2. A speed of $0.75 - 1.5m/s$ means that each node chooses its current speed randomly from the interval $(0.75; 1.5]\,m/s$. Third, a high speed of $1.5 - 3m/s \approx 5 - 10km/h$, which corresponds to walking at a quick pace or jogging. Fourth, a very high, but for the addressed scenarios in Section 3.1.3 still realistic speed of $10 - 20km/h$, which is the speed of a non-throttled forklift or a bike at slow to medium pace. Fifth, an extremely high, for the scenarios in Section 3.1.3 not realistic speed of $20 - 40km/h$, which may, however, be realistic for other scenarios. For the graphical representation of the simulation results, only the maximum speed is indicated in the respective legend for the sake of shorter notation.

An evaluation of the influence of different degrees of mobility is of interest since nodes' movement accelerates the distribution of the symbols in the network. The higher the degree of mobility, the faster the distribution of a considered symbol through the network will be if symbols are randomly selected for retransmission, as with NoC, RLNC, and GC. In this case, each node will be completed earlier. Thus, node and system latency will be improved, which is, therefore, compared in Figures 7.56 and 7.63 to 7.65 for a respective procedure for each considered degree of mobility. By contrast, in the case of BCGC, symbols are not randomly selected for retransmission. Instead, counters are used for the specific symbol selection, which are less accurate the higher the degree of mobility is. For this reason, it is especially relevant to perform simulations of BCGC with a varied degree of mobility. If codewords are composed less appropriately, the required number of received packets to reconstruct a respective number of distinct original symbols is increased and, thus, latency becomes worse. Therefore, it should be evaluated whether a higher degree of mobility causes an increase of the RNP and, hence, of the latency. These are, thus, plotted in Figure 7.57 and Figure 7.56 for BCGC. In particular, it should be analyzed in this context whether the specific symbol selection is disadvantageous in the case of a higher degree of mobility. For this purpose, the BCGC procedure with specific symbol selection and the BCGC procedure with random symbols are compared in Figures 7.61 and 7.62 for the highest speed considered. In addition, in the case of BCGC, the slope of the degree function is increased compared to the GC degree function. The reason for this is to compensate for the fact that more redundant distance-0-codewords and fewer distance-1-codewords are received than assumed for the original GC degree function. Especially in the beginning of the dissemination process, only a small number of distinct original symbols is available in the vicinity of a node. Only these can be used for the creation of a packet/codeword. As a consequence, the number of received redundant packets/codewords is increased in comparison to a situation where all distinct original symbols are available for the creation of a codeword, as explained in detail in Section 5.2.1. The faster the nodes move, the more distinct original symbols are present in the vicinity of each node. Therefore, the less the slope of the degree function has to be increased compared to the original GC degree function. However, in the beginning of the data dissemination process, contrary to what is assumed for the GC degree function, all symbols can still not be present in each node. For this reason, a certain increase of the slope of the degree function compared to the original GC degree function will be beneficial even in the case of a high degree of mobility. Here, however, it is not the goal to determine an optimal slope for each considered degree of mobility. Instead, the impact of the different considered degrees of mobility on the chosen default degree function should be analyzed. Therefore, the percentage of received distance-0-codewords/distance-1-codewords/distance->1-codewords is plotted in Figures 7.58 to 7.60. If the percentage of received distance-0-codewords/distance-1-codewords decreases/increases with increasing speed, this is due to the better distribution and, thus, availability of the symbols for the codeword composition. The improved availability of distinct original symbols together with a then possibly too aggressive degree function increases the percentage of received distance->1-codewords. However, if the used default degree function is too aggressive for a higher degree of mobility, the percentage of distance->1-codewords will increase and at the same time the percentage of distance-1-codewords will decrease instead of the distance-0-codewords.

In the following, the results are interpreted:
For BCGC, NoC, GC, and RLNC, it can be seen in Figures 7.56 and 7.63 to 7.65 that a higher speed leads to a reduction in node and system latency. This is solely due to the faster distribution of the symbols in the network. Since the number of sent/received packets per considered time interval is not significantly affected by the variation of the chosen speed, mainly the respective RNP causes a change in latency. Furthermore, it can be observed in Figures 7.56 and 7.63 to 7.65 that the difference between the time



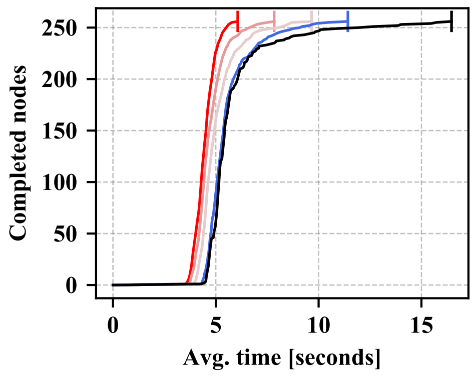


Fig. 7.56: Completed nodes

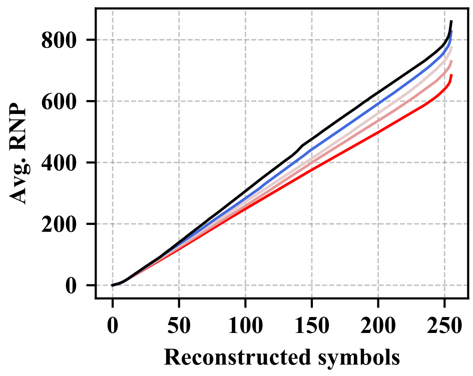


Fig. 7.57: RNP

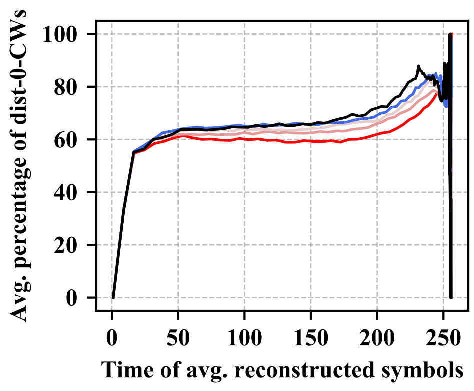


Fig. 7.58: Received dist-0-CWs

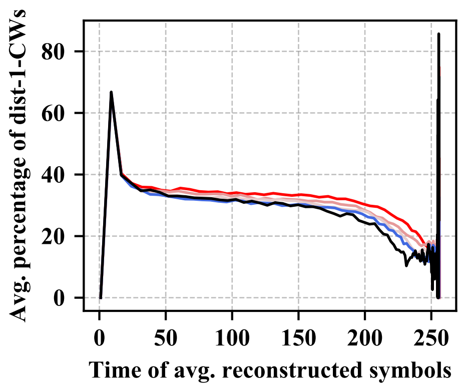


Fig. 7.59: Received dist-1-CWs

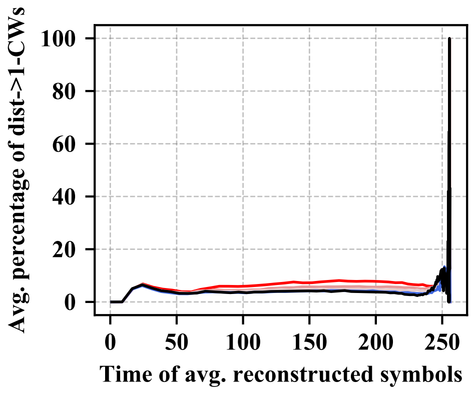


Fig. 7.60: Received dist->1-CWs

Performance Metric	BCGC-0km/h	BCGC-5km/h (default)	BCGC-10km/h	BCGC-20km/h	BCGC-40km/h
RNP$_{100}$ $\pm\sigma$ (min, max)	859 ± 95 (305, 3393)	827 ± 85 (277, 3213)	774 ± 33 (315, 2379)	729 ± 33 (331, 1549)	684 ± 32 (352, 1128)
Node Latency $\pm\sigma$ $[s]$ (min)	5.8 ± 0.9 (3.1)	5.5 ± 0.9 (3.2)	5.0 ± 0.2 (3.5)	4.7 ± 0.2 (3.4)	4.5 ± 0.3 (2.9)
System Latency $\pm\sigma$ $[s]$	16.5 ± 5.5	11.4 ± 2.9	9.7 ± 2.4	7.8 ± 1.1	6.1 ± 0.5
Time 95% of Nodes Completed $[s]$	9.6	8.0	6.8	6.1	5.3
PRR	$1 - 0.351$	$1 - 0.341$	$1 - 0.338$	$1 - 0.333$	$1 - 0.326$
Throughput$_{eff}$ $[bit/s]$	7166	7562	8133	8710	9237
Sent Packets	213 ± 27	202 ± 27	186 ± 7	174 ± 7	164 ± 9
Energy Consumption $[J]$	0.4985	0.3488	0.2964	0.2419	0.1887

Table 7.17: Performance metrics for BCGC with varied degree of mobility

Performance Metric	NoC-0km/h	NoC-5km/h (default)	NoC-10km/h	NoC-20km/h	NoC-40km/h
RNP$_{100}$ $\pm\sigma$ (min, max)	n/a n/a	9489 ± 483 (3742, 17881)	7838 ± 892 (3421, 16000)	5817 ± 691 (2935, 11719)	4368 ± 1117 (2135, 9489)
Node Latency $\pm\sigma$ $[s]$ (min)	n/a n/a	47.5 ± 2.5 (23.7)	39.2 ± 4.5 (19.0)	29.0 ± 3.5 (17.1)	21.7 ± 5.5 (10.8)
System Latency $\pm\sigma$ $[s]$	n/a	80.4 ± 13.1	60.4 ± 9.0	41.7 ± 5.8	30.2 ± 6.1
Time 95% of Nodes Completed $[s]$	n/a	64.4	52.2	37.0	26.3
PRR	n/a	$1 - 0.302$	$1 - 0.301$	$1 - 0.299$	$1 - 0.299$
Throughput$_{eff}$ $[bit/s]$	n/a	865	1057	1432	1972
Sent Packets	n/a	1943 ± 101	1603 ± 183	1183 ± 142	885 ± 226
Energy Consumption $[J]$	n/a	2.6410	1.9851	1.3709	0.9896

Table 7.18: Performance metrics for NoC with varied degree of mobility

Performance Metric	RLNC-0km/h	RLNC-5km/h (default)	RLNC-10km/h	RLNC-20km/h	RLNC-40km/h
RNP_{100} $\pm\sigma$ (min, max)	503 ± 133 (259, 5910)	460 ± 92 (257, 1893)	414 ± 58 (258, 1012)	387 ± 11 (265, 1216)	367 ± 11 (262, 783)
Node Latency $\pm\sigma$ $[s]$ (min)	3.5 ± 0.8 (1.6)	3.2 ± 0.6 (1.6)	2.9 ± 0.4 (1.6)	2.7 ± 0.1 (1.6)	2.5 ± 0.1 (1.7)
System Latency $\pm\sigma$ $[s]$	14.8 ± 8.5	9.3 ± 3.6	6.5 ± 1.4	5.5 ± 1.1	4.6 ± 0.6
Time 95% of Nodes Completed $[s]$	7.7	4.8	4.1	3.8	3.4
PRR	$1 - 0.453$	$1 - 0.457$	$1 - 0.453$	$1 - 0.453$	$1 - 0.451$
Throughput_{eff} $[bit/s]$	12210	13179	14477	15299	16226
Sent Packets	122 ± 30	110 ± 20	98 ± 13	92 ± 2	86 ± 3
Energy Consumption $[J]$	0.4492	0.2841	0.1989	0.1679	0.1408

Table 7.19: Performance metrics for RLNC with varied degree of mobility

Performance Metric	GC-0km/h	GC-5km/h (default)	GC-10km/h	GC-20km/h	GC-40km/h
$\text{RNP}_{100} \pm\sigma$ (min, max)	n/a n/a	18420 ± 6023 (11218, 29566)	8544 ± 1957 (5375, 11886)	5096 ± 735 (3825, 5936)	2900 ± 242 (2322, 3172)
Node Latency $\pm\sigma$ $[s]$ (min)	n/a n/a	145.3 ± 48.2 (91.2)	68.0 ± 14.7 (44.5)	39.8 ± 5.8 (29.9)	23.0 ± 2.0 (19.3)
System Latency $\pm\sigma$ $[s]$	n/a	145.3 ± 48.2	68.0 ± 14.7	39.8 ± 5.8	23.0 ± 2.0
Time 95% of Nodes Completed $[s]$	n/a	145.3	68.0	39.8	23.0
PRR	n/a	$1 - 0.379$	$1 - 0.377$	$1 - 0.377$	$1 - 0.375$
$\text{Throughput}_{\mathit{eff}}$ $[bit/s]$	n/a	310	630	1050	1794
Sent Packets	n/a	5094 ± 1739	2380 ± 583	1411 ± 249	818 ± 89
Energy Consumption $[J]$	n/a	4.5382	2.1218	1.2419	0.7169

Table 7.20: Performance metrics for GC with varied degree of mobility

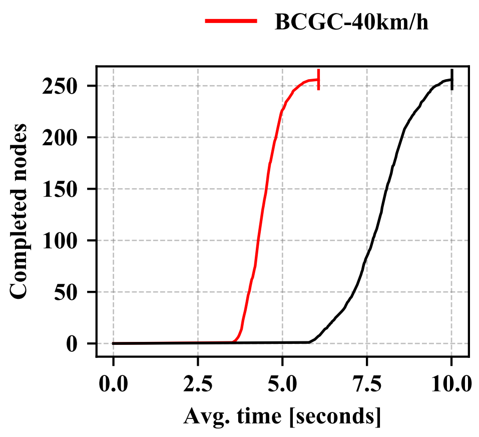


Fig. 7.61: Completed nodes

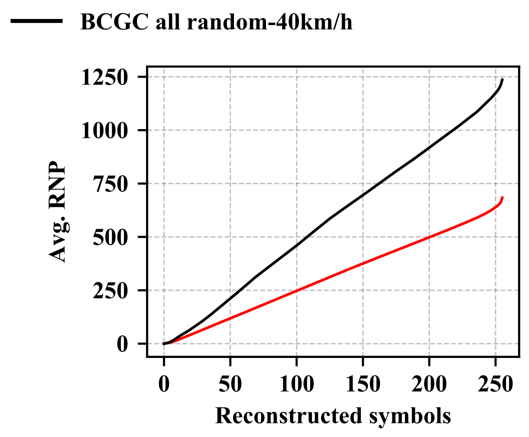


Fig. 7.62: RNP

at which the first node finishes and the time at which the last node in the network finishes is smaller the higher the degree of mobility is. This means that the aspect of the long-tail problem that many nodes are completed quickly and a few nodes take a very long time becomes less significant as the speed increases. This is due to the fact that with increasing speed, poorly connected nodes are poorly connected for a decreasing amount of time. Nodes at the boundary area receive less packets per considered time interval, receive more redundant packets, and have the largest distance to all other nodes in the network among all nodes. As a consequence, they take longer to be completed. However, the higher the degree of mobility, the shorter the time nodes remain at the boundary area if the default mobility model is used, and consequently the less prominent the long-tail problem. On the other hand, the less the nodes move, the more significant is the long-tail problem for each of the considered procedures. More general details about the long-tail problem are already explained in Section 7.1.1.1.

For BCGC, it should be analyzed separately whether a high degree of mobility is disadvantageous for the procedure due to the specific symbol selection. Regarding the RNP, however, it can be observed in Figure 7.57 that a higher speed improves the RNP and, thus, also the latency, see Figure 7.56. This is due to the fact that the higher the speed of the nodes, the more often there are nodes with new symbols in the vicinity of a considered node. It is, therefore, less likely to receive a distance-0-codeword and more likely to receive a distance-1-codeword or a distance->1-codeword from its neighbors, which can also be seen in Figures 7.58 to 7.60. In principle, it can be assumed that the specific symbol selection provides a greater improvement in the case of a static network or a network with a low degree of nodes' mobility than in the case of a highly dynamic system. However, as can be seen in Figures 7.61 and 7.62, even at a very high speed, both RNP and, thus, latency are improved by specific symbol selection

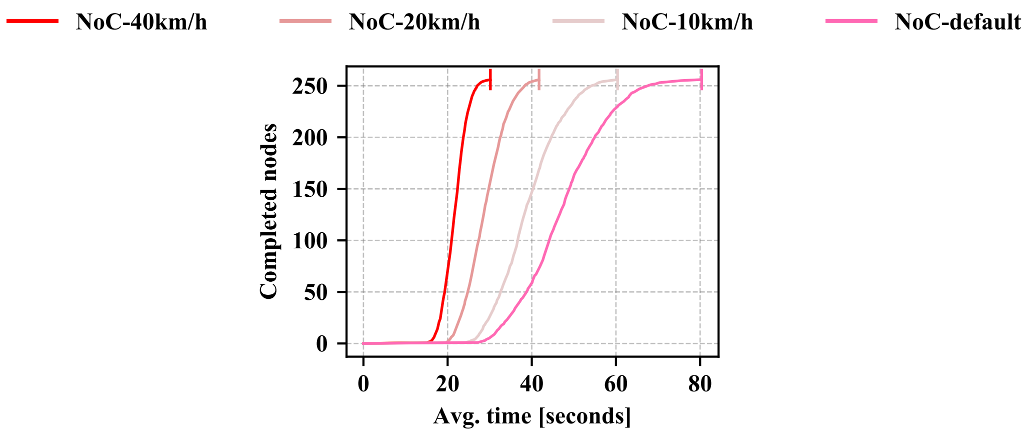


Fig. 7.63: Completed nodes

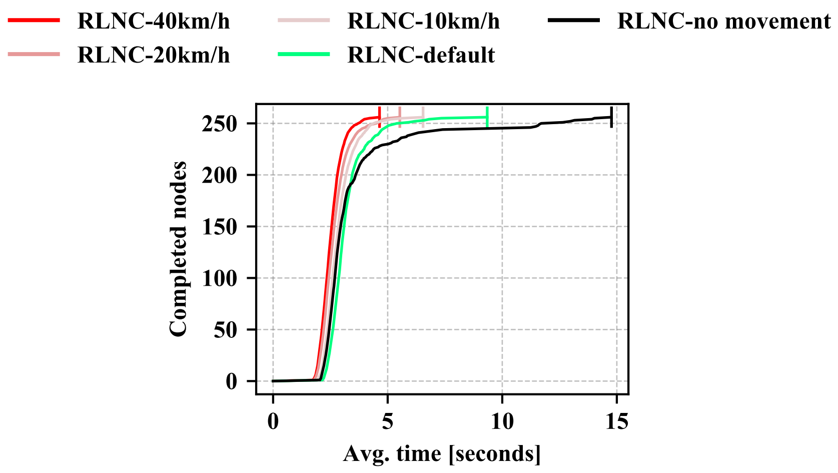


Fig. 7.64: Completed nodes

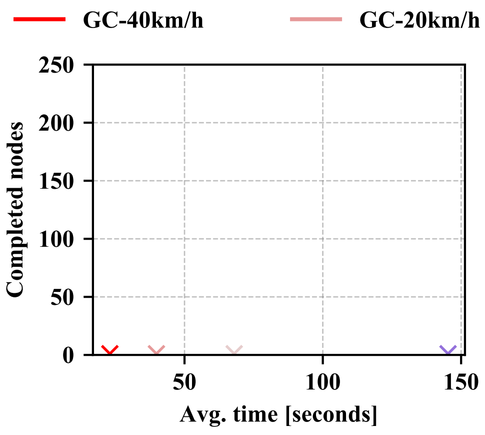


Fig. 7.65: Completed nodes

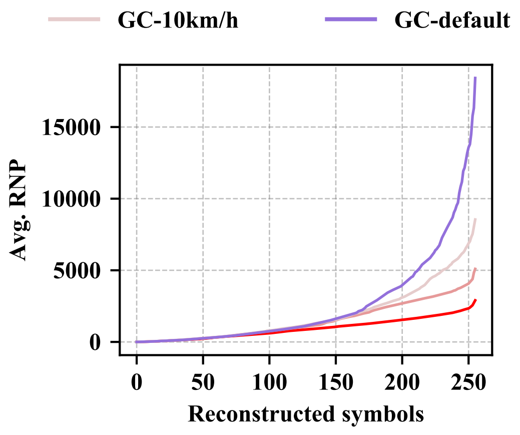


Fig. 7.66: RNP

compared to random symbol selection. This can be explained as follows. With specific symbol selection, only one symbol is specifically selected, the symbol which is most likely to be beneficial to possible receivers according to the used counters. Often, most nodes in the vicinity of a node are missing similar symbols. Consequently, even at very high speed, nodes move only a certain distance away between the transmission of two packets. Hence, gathered information about profitable symbols still has a certain validity. Therefore, even with a high degree of mobility, specific symbol selection yields better results than random symbol selection. For more details on specific symbol selection, see Sections 5.3.1 and 7.1.2.1. In summary, specific symbol selection is advantageous for BCGC even in the case of a highly dynamic system, and BCGC are suitable for any degree of mobility.

In Figure 7.60, it can be seen that with the used degree function, the percentage of received distance->1-codewords increases slightly with increasing speed. However, the increase in the percentage of distance->1-codewords is only very small. At the same time, there is no decrease but an increase in the percentage of received distance-1-codewords the higher the degree of mobility is, see Figure 7.59. Moreover, the percentage of received distance-0-codewords decreases at the same time, see Figure 7.58. The default BCGC degree function with $maxDeg = 46$ is, therefore, not too aggressive even with a high degree of mobility and the resulting more uniform distribution of the probabilities of the symbols to be included in a codeword. Instead, the default BCGC degree function even provides better results the more dynamic the system is. A dynamic adaptation of the degree function to the prevailing dynamics recognized by each node for its surrounding area is investigated to some extend for the initial version of BCGC in [Stö17]. However, an optimization of the BCGC degree function depending on the respective observed dynamics is beyond the scope of this thesis.

BCGC are in a similar range as RLNC with respect to latency at each considered degree of mobility with the default number of nodes $n = 256$. However, at a larger number of nodes/symbols, BCGC represent a significant improvement over RLNC, which is evaluated separately in Section 7.2.3. By contrast, BCGC represent a significant improvement compared to NoC at any degree of mobility.

Unlike BCGC, GC are not designed for systems with a low degree of nodes' mobility and perform poorly in such systems, as already shown in Section 7.2.1. As explained theoretically in Section 4.3, GC require a high degree of mobility since only in this case the assumption made for the GC degree function that all symbols have the same probability to be included in a codeword is at least approximately true. Due to the more even availability of the symbols with increasing speed, the GC degree function then fulfills its purpose to minimize the required number of received packets to reconstruct a respective number of original symbols increasingly better. This means, the higher the degree of mobility the better the RNP for GC, see Figure 7.66. For GC, a very strong improvement of the RNP can be seen as the speed of the nodes increases. This also leads to a significant improvement of the latency, see Figure 7.65, so that GC has a lower latency than NoC in the case of a high degree of mobility. However, even with a highly dynamic system, the latency of BCGC is significantly better than the latency of GC, see Figure 7.56. Reasons for this are that the slope of the GC degree function is too low. Moreover, in the case of GC, a node can only add its own symbol to a codeword so that many redundant packets/codewords are received. For further reasons, see Section 5.5. As can be seen in Figure 7.65, in the case of GC, the last node needs about $150\,\mathrm{s} = 2.5\,\mathrm{min}$ to finish when using the default degree of mobility. In the case of no movement of the nodes, the simulation was stopped after a simulated time of 20 min without GC having been terminated. Thus, corresponding results are omitted in the plots and tables here as GC perform so much worse than BCGC in the case of a static network that the exact values are no longer relevant. For NoC, the same applies in the case of no movement of the nodes.

7.2.5 Varied Degree of Node Failures

In this last section of the evaluation, the performance of BCGC is analyzed with node failures compared to a situation without node failures and compared with NoC and RLNC. In the case of node failures, exponentially distributed node failures are assumed, see Section 6.1.6. The critical distance to the center up to which all nodes should fail is chosen so that 10% of the simulation area is 100% affected. The area mainly affected by node failure is set to 20% of the simulation area, and there are also node failures outside this region, as described in Section 6.1.6. In total, 95% of the node failures is within the chosen 20% radius, 5% is outside. Furthermore, it is assumed that node failure occurs after 1.5 s or after 5 s. This corresponds to almost 25% of the time at which nodes are completed on average or to approximately the time at which nodes are completed, respectively. Overall, drastic node failures in a critical part of the simulation area are simulated here. Due to the low degree of mobility used by default, the resulting gap without active nodes in a substantial part of the simulation area is not quickly reoccupied by nodes that are still active. It is not the goal here to compare different patterns of node failure. Instead, it is intended to show that BCGC can still be functional under a drastic form of node failure and ensure that all distinct original symbols still present in the network can be reconstructed by all still operating nodes. It shall, hence, be demonstrated that BCGC enable a reliable data dissemination process even if large parts of the network fail at an early stage. Furthermore, this will also be analyzed for related work procedures such as NoC and RLNC.

In particular, node latency is given in Figures 7.67, 7.69, and 7.70 to show how drastic node failures affect this important performance metric. Further performance metrics can be found in Tables 7.21 to 7.23. Latency is determined by the RNP_{100} and the number of packets received per considered time interval. Therefore, for BCGC, the RNP_{100} is shown in Table 7.21 and the number of sent/received packets in Figure 7.68. Symbols originating from one side of the region affected by node failure may reach

Performance Metric	BCGC (default)	BCGC-with node failures at $t = 5\,\text{s}$	BCGC-with node failures at $t = 1.5\,\text{s}$
$\text{RNP}_{100} \pm\sigma$ (min, max)	827 ± 85 (277, 3213)	834 ± 83 (277, 1954)	1026 ± 156 (292, 2674)
Node Latency $\pm\sigma$ $[s]$ (min)	5.5 ± 0.9 (3.2)	5.6 ± 1.0 (3.5)	7.1 ± 1.6 (4.0)
System Latency $\pm\sigma$ $[s]$	11.4 ± 2.9	11.5 ± 3.0	12.4 ± 3.0
PRR	$1 - 0.341$	$1 - 0.340$	$1 - 0.330$
$\text{Throughput}_{\mathit{eff}}$ $[bit/s]$	7562	7459	5944
Sent Packets	202 ± 27	206 ± 28	261 ± 51
Energy Consumption $[J]$	0.3488	0.3544	0.3868

Table 7.21: Performance metrics for BCGC w/o vs. with node failures

nodes on the opposite side via fewer different paths than in the case without node failure. Thus, more packets may have to be received to reconstruct, in particular, all distinct original symbols in a node. This means the RNP may be affected by node failure. In addition, occurring node failures reduce the node density in the network. With the used default MAC protocol, nodes are then less likely to find the medium occupied and can, therefore, send more frequently. Furthermore, there may be less packet collisions due to simultaneous transmissions in the case of a lower node density. As a consequence of these two aspects, the number of sent/received packets per considered time interval can also be different in the case of node failures. A main goal of BCGC mentioned in Section 3.1.2 is reliability. For this reason, it is desirable to have to visit/query as few nodes as possible at any time to gather all original data. The corresponding plot is, therefore, shown in Figure 7.71. The fewer random nodes have to be queried at any time, the more robust the system is against node failure and the faster all data can be gathered by an external agent, for example. Overall, it is also an indication of how evenly each symbol is already spread across the network.

In the following, the results are interpreted:
As can be seen in Figure 7.67 and Table 7.21, even severe node failures of about 20% of the nodes in the substantial central part of the simulation area early in the data dissemination process did not pose a problem for BCGC. Due to the occurring node failures, a larger required number of packets, RNP_{100}, had to be received in order to reconstruct all distinct original symbols in a node in the case of BCGC, see Table 7.21. The earlier the node failure occurs, the fewer nodes are already finished at that point in time, i.e. have an RNP_{100}^{node} which has not been influenced by node failures. Thus, the more packets have to be received in a node on average, i.e. the higher the RNP_{100} is. In addition, the failure region is in the central part of the simulation area. In this

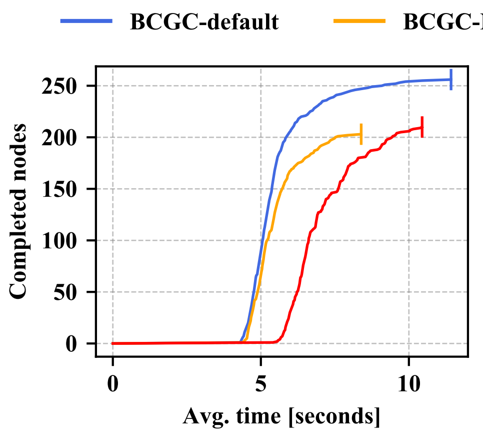


Fig. 7.67: Completed nodes

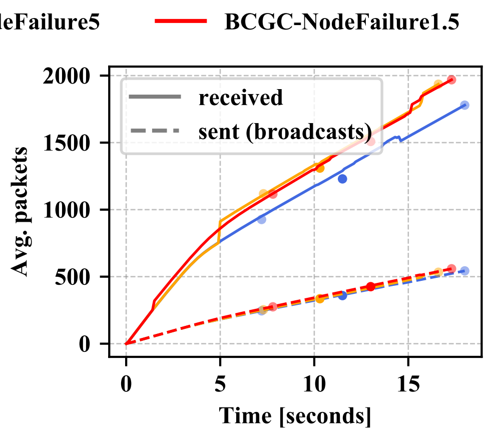


Fig. 7.68: Sent/received packets

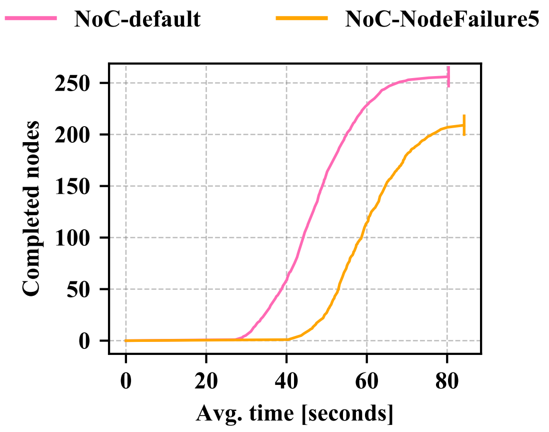


Fig. 7.69: Completed nodes

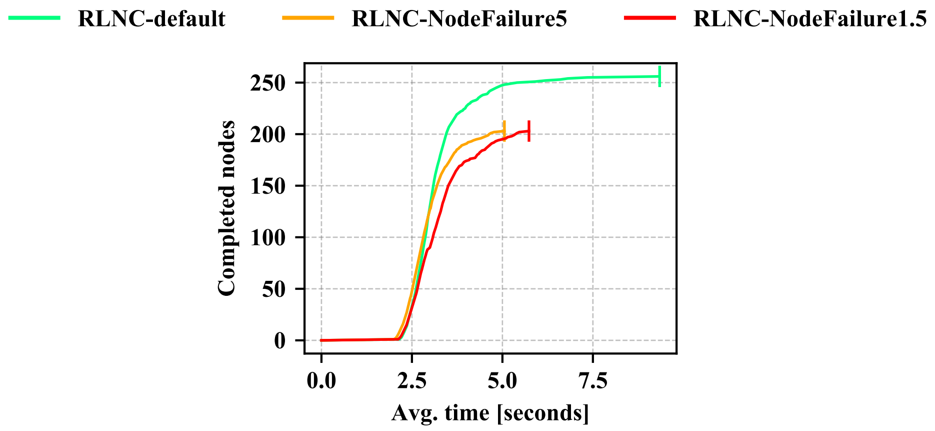


Fig. 7.70: Completed nodes

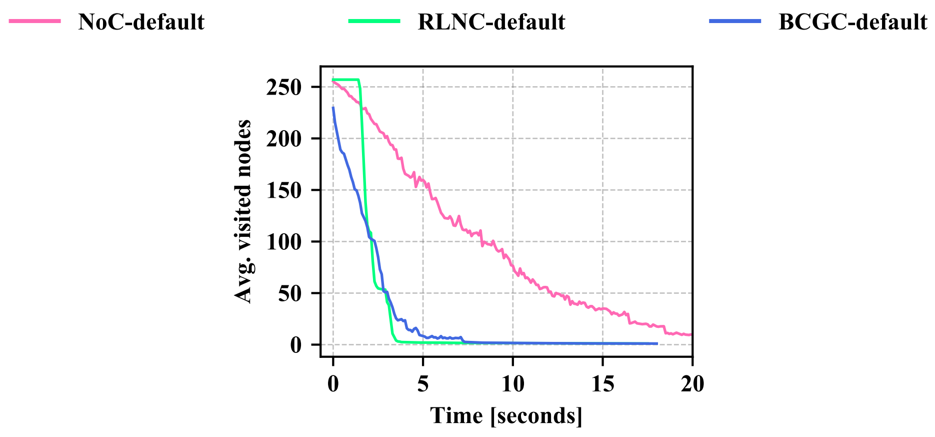


Fig. 7.71: Visited nodes in the case of default simulation settings without node failures

Performance Metric	NoC (default) (default)	NoC-with node failures at $t = 5\,\mathrm{s}$	NoC-with node failures at $t = 1.5\,\mathrm{s}$
RNP_{100} $\pm\sigma$ (min, max)	9489 ± 483 (3742, 17881)	11602 ± 683 (3674, 21142)	n/a n/a
Node Latency $\pm\sigma$ $[s]$ (min)	47.5 ± 2.5 (23.7)	60.5 ± 3.5 (34.8)	n/a n/a
System Latency $\pm\sigma$ $[s]$	80.4 ± 13.1	94.3 ± 14.9	n/a
PRR	$1 - 0.302$	$1 - 0.280$	n/a
$\mathrm{Throughput}_{\mathit{eff}}$ $[bit/s]$	865	679	n/a
Sent Packets	1943 ± 101	2510 ± 146	n/a
Energy Consumption $[J]$	2.6410	3.1367	n/a

Table 7.22: Performance metrics for NoC w/o vs. with node failures

Performance Metric	RLNC (default) (default)	RLNC-with node failures at $t = 5\,\mathrm{s}$	RLNC-with node failures at $t = 1.5\,\mathrm{s}$
RNP_{100} $\pm\sigma$ (min, max)	460 ± 92 (257, 1893)	440 ± 57 (259, 1408)	483 ± 87 (257, 1633)
Node Latency $\pm\sigma$ $[s]$ (min)	3.2 ± 0.6 (1.6)	3.1 ± 0.4 (1.6)	3.4 ± 0.6 (1.6)
System Latency $\pm\sigma$ $[s]$	9.3 ± 3.6	8.4 ± 2.2	9.2 ± 2.5
PRR	$1 - 0.457$	$1 - 0.452$	$1 - 0.440$
$\mathrm{Throughput}_{eff}$ $[bit/s]$	13179	13519	12395
Sent Packets	110 ± 20	106 ± 14	118 ± 21
Energy Consumption $[J]$	0.2841	0.2567	0.2827

Table 7.23: Performance metrics for RLNC w/o vs. with node failures

area, nodes typically have reconstructed the most symbols at a considered point time and can forward all of these symbols to their neighbors. If these nodes fail, this is no longer possible. Furthermore, as soon as node failures have occurred, symbols have to traverse more hops to reach and be reconstructed by nodes on the other side of the failure region. This is due to the lower node density and missing connections in the center. This consequently increases the RNP, the required number of received packets to reconstruct a respective number of distinct original symbols, from the time of node failure. Furthermore, in the case of node failure, the average node density in the network is reduced. As a consequence, the medium is found occupied less often and due to the additionally fewer simultaneous packet transmissions, the collision rate $coll := 1 - PRR$ will be reduced. This can be seen for BCGC in Table 7.21. Therefore, with the used default MAC protocol and the considered degree and pattern of node failure, slightly more packets were sent and more packets were received per considered time interval by a node after the occurrence of node failures, as can be seen in Figure 7.68. However, if node failures reduce the density significantly, an increased number of received packets per time interval may no longer be observed. This can especially be the case for a different pattern of node failures, such as the pattern of random node failure where each node fails with $x\%$ probability. An increased number of received packets per considered time interval compensates for an increased required number of received packets to a certain extend. However, it was not possible to completely compensate for this increased number here. Therefore, node and system latency are increased in the case of node failure, as shown in Table 7.21. In Figure 7.67, the result appears to be different at first glance. Here, however, the average value was plotted only up to the lowest number of nodes still existing in the system among all seeds in order to avoid any distortion of the respective graph due to the decreasing number of available seeds. From the course of the graph, however, it can already be seen that in the case of node failures and especially in the case of early node failures, a respective number of completed nodes was reached at a later point in time. With regard to node and system latency, the values from Table 7.21 should be considered. Furthermore, Figure 7.67 shows that at the time of node failure at 5 s, some nodes were already completed. Then, the process of completing further nodes was slowed down due to the occurring node failures. In the case of node failure after 1.5 s, no node was finished at the time of the occurring node failures, so also the completion of the first nodes was delayed. Nevertheless, even under these drastic node failures, all symbols still existing in the system could be gathered and reconstructed in all remaining nodes in the case of BCGC.

Unlike BCGC, NoC was only able to cope with the considered node failures which occurred after 5 s under favorable conditions. In the case of node failures after 1.5 s, for all seeds, there were some symbols which were no longer present in the system as they had not yet been distributed well enough in the network with NoC. In this case, the simulation was stopped because the nodes could not have been able to reconstruct all distinct original symbols anymore. With node failures occurring after 5 s, this was also the case for several seeds for NoC. The plotted graphs, therefore, only contain those seeds for NoC for which all distinct original symbols were still in the network after the node failure. In the case of node failures after 1.5 s, in each seed no node could be completed in NoC. Therefore, no values are given for this degree of node failure in the plots or in the table. In contrast, when using BCGC, symbols were already sufficiently distributed over the network in the considered scenarios. In this case, the application of BCGC allowed the reconstruction of all distinct original symbols in all remaining active nodes and improved the reliability compared to the use of pure forwarding without encoding of data, i.e. NoC.

As with BCGC, in RLNC, symbols were well distributed across the network at an early stage. For this reason, all remaining active nodes could reconstruct all distinct original symbols in the case of node failures after both 5 s and 1.5 s. In general, RLNC and BCGC were able to cope well with early node failure in a substantial part of the network, unlike NoC.

How well symbols in decoded/reconstructed form were distributed over the network at a respective time in the case of no node failure can be seen in Figure 7.71. There, the average number of randomly chosen nodes which had to be visited/queried to retrieve all distinct original symbols at a respective time is plotted. These respective values were obtained by executing 25 random collection processes for each seed at fixed times and then averaging the resulting numbers of queried nodes per considered point in time. Then, the average is calculated over all seeds. Thus, Figure 7.71 shows the number of queried nodes as described in Section 3.3.3, which is an important performance metric and helpful for explaining the results in this section. Here, for each random collection process, reconstructed symbols are gathered instead of codewords as a collection process should be performed quickly and without complex computation. Especially in the case of RLNC, it would otherwise be necessary to check for each packet/codeword whether it is linearly independent of the packets/codewords which have already been collected. Alternatively, regular computationally expensive decoding attempts would have to be made during the collection process to determine the actual number of nodes to be visited/queried. Both would be computationally intensive. Therefore, in the case of RLNC, random nodes have to be visited to retrieve all distinct original symbols until a first node is queried which has already reconstructed all symbols. This is due to the fact that in RLNC, nodes can decode received packets/codewords only after they have received n linearly independent packets/codewords in the case of n distinct original symbols in the system. Before that, nodes do not have any reconstructed symbols. For example, if five nodes had to be queried for RLNC, it means that the first four queried nodes had not reconstructed anything and the last one was finished. In BCGC and NoC, on the other hand, nodes have already stored reconstructed symbols from the beginning and, thus, each queried node contributes to the collection process.

With 20% node failure, i.e. approximately 200 of 256 nodes remain active, the limit up to which a particular process could have been completed at all is where the curve under consideration lies below about $y = 200$ in Figure 7.71. Above this limit, more nodes had to be visited/queried randomly than there would still have been active nodes in the network in the case of 20% node failure. If at a considered point in time more than 200 nodes had to be queried/visited according to the graph to gather all distinct original symbols and then all nodes fail except for 200 nodes, it is very likely that some symbols are no longer in the system. This is the reason why in the case of NoC and node failures after 1.5 s, not all symbols could be reconstructed in all remaining nodes and the algorithm was stopped. In the case of exponentially distributed node failures, the probability that symbols got lost is even higher than can be deduced from Figure 7.71. The reason for this is that a connected region fails and all symbols from this area have to be already present outside the failure area when the node failure occurs in order not to lose those symbols. For exponentially distributed node failures, the result, therefore, also depends on how well symbols from the central part of the network have already been distributed to the surroundings. For a procedure which distributes data more quickly, the degradation compared to Figure 7.71 is less severe than for procedures in which data is distributed more slowly. Under the default conditions, it is therefore to be expected that for RLNC, due to its lower latency than for BCGC, the proportional deterioration of the situation is less noticeable. Under conditions other than the default, e.g. with a higher number of nodes, where RLNC has a worse latency, the situation will worsen more for RLNC. With NoC an even more significant worsening of the situation than shown in Figure 7.71 is expected.

In Figure 7.71, it can be seen that BCGC has advantages over RLNC in the case of random node failures at an early point in time. Then, more random nodes can fail in the case of BCGC than in the case of RLNC without loosing at least one symbol. This is due to the already described fact that for BCGC, codewords are decoded and symbols reconstructed from the beginning. This is also the case for NoC. However, due to the Coupon Collector's problem with NoC, the rate at which symbols are reconstructed with NoC is significantly worse than with BCGC. Therefore, usually significantly more nodes have to be queried than with BCGC.

Concerning node failure, BCGC have a further crucial advantage over RLNC. This advantage is related to the fact that with RLNC, a node is either able to reconstruct everything or nothing whereas with BCGC, symbols are reconstructed gradually. In the case of node failure, there may be symbols which are no longer in the system, i.e. are not contained in any stored codeword/packet/reconstructed symbol. In general, it may be the case that there are no longer n linearly independent stored packets/codewords in the system after node failure. Then, nodes with RLNC will never be able to receive n linearly independent packets and, thus, will never begin decoding. As a consequence, all nodes will never have reconstructed anything with RLNC in this case. With BCGC, on the other hand, all remaining nodes start decoding immediately after having received a first packet/codeword. For RLNC, this partial decoding would be possible only by a modification.

To conclude, even in unreliable networks, where large parts of the network fail at an early stage, BCGC ensure that all distinct original symbols still present in the network and being decodable can be gathered and reconstructed by all remaining nodes. This is possible as every node aims to reconstruct as much original data as possible at any considered point in time and, in particular, due to the ability of BCGC to perform early decoding.

Chapter 8

Testbed based Evaluation

Contents

For the evaluation of BCGC process parameters and of BCGC using different simulation settings in Sections 7.1 and 7.2, an in-house developed network simulator was used. However, a simulator abstracts reality to a certain extent and results are, therefore, not exact. Some parameters, such as energy consumption, can only be estimated very roughly as no real hardware is used. The benefits of using a simulator have already been discussed in Section 6.1.1. For the reasons mentioned there, the main evaluations of BCGC were carried out by means of a simulator. However, in order to obtain more accurate results and to verify the results from the simulator, a supplementary evaluation of BCGC on a real testbed is reasonable, which therefore follows here. For this purpose, general information on the used testbed environment is first presented in Section 8.1. Then, the used settings for the testbed and the required changes of the settings for the simulator are described in Section 8.2. Finally, results from the testbed are presented and compared to the corresponding results from the simulator in Section 8.3.

8.1 Description of Testbed Environment

For the following evaluation, the FIT IoT-LAB testbed [Adj+15b] is used. It is an open-access remote testbed consisting of more than 2700 nodes, including more than 100 mobile nodes. These nodes are distributed over 6 locations in France. Each node provided in the testbed contains a so-called *Open Node* (ON), a *Gateway* (GW), and a *Control Node* (CN). An ON can be accessed and programmed for the experiments. By contrast, a CN cannot be accessed for experiments, but coordinates the reprogramming of the associated ON, and monitors the power consumption of this ON, for example. In the remainder of this chapter, the term 'node' refers to the respective ON. In the testbed, there are different types of nodes, the largest proportion being the so-called M3 sensor nodes, which were therefore used for the experiments presented in the following. M3 nodes are also the type of nodes with the largest number of available nodes at one site and the type of nodes with the largest number of nodes that can communicate directly with each other via radio. M3 sensor nodes are equipped with a 32-bit ARM Cortex M3 micro-controller[4] (STM32F103REY) and an ATMEL AT86RF231 radio chip, which is compliant to IEEE Std 802.15.4 and uses the 2.4 GHz ISM frequency band. Furthermore, these nodes are static nodes and provide four types of sensors, a light sensor, a pressure and temperature sensor, an accelerometer/magnetometer, and a gyroscope.

8.2 Testbed and Simulator Settings

The evaluation of BCGC and comparison with related work using the FIT IoT-LAB testbed platform was mainly performed as part of a master thesis, see [Ome18], which was conducted as part of this work. Therefore, the results presented in Figures 8.1 and 8.2 and Tables 8.1 to 8.3 regarding the testbed are based on [Ome18]. For the evaluation presented in the following, experiments with 81 M3 sensor nodes were executed. This number of nodes was the maximum number of nodes which were able to communicate directly with each other via radio communication and were repeatedly available at the time of the experiments. According to the addressed system setup in Section 3.1.1, a considered static network shall be connected but does not have to be fully connected. This condition is, thus, fulfilled for the 81 selected static nodes.

Since the 81 selected nodes were located very close to each other and on two parallel lines, the transmit power was reduced to the lowest possible transmit power of the used transceiver, $P_t = -17\,\text{dBm}$, to avoid that each node could reach every other node. Otherwise, the network would have been fully connected and, thus, have a very high node density, which contradicts the assumed system setup in Section 3.1.1. For the remaining settings of the transceiver, the default values described in [Atm09] and Section 6.2.2 were used. With these settings, an average number of 24 neighbors was encountered during the executed experiments on the testbed, see [Ome18]. Except for the transmit power, the same default values are also used for the evaluations in Sections 7.1 and 7.2 with the simulator.

Concerning the MAC protocol, the unslotted CSMA/CA based MAC protocol according to IEEE Std 802.15.4 was chosen for the experiments in [Ome18], which is also used as default MAC protocol in the simulator and described in Section 6.1.3.1. In [Ome18], the maximum number of backoffs *macMaxCSMAbackoffs* that are allowed to be performed was set to *macMaxCSMAbackoffs* $= 0$ for the experiments. Furthermore, the time t_{RX} in

[4]7.2 MHz, 64 kB RAM, 256 kB ROM, 512 kB Flash

which a node is in receive mode was set to $t_{RX} = 15 \cdot t_p$. Here, t_p is the transmission time of the packet to be sent next by the considered node. This means, t_{RX} depends on the current packet size and may, thus, be different for different procedures and in the case of dynamic packet sizes, additionally in the course of the dissemination process. Except for *macMaxCSMAbackoffs* and t_{RX}, the default values given in IEEE Std 802.15.4 were used for all other parameters related to the MAC protocol. Furthermore, nodes did not perform duty cycling and, in particular, were not repeatedly set to an energy-saving $SLEEP$ state for the experiments shown below. As described in detail in Section 6.1.7, using duty cycling is not beneficial for the addressed system setup here as all data is to be distributed quickly and nodes do not perform idle listening for a long time. In [Ome18], also experiments using a $SLEEP$ state were executed which reduced energy consumption in some cases but always increased latency.

In [Ome18], a payload size of 2 Bytes was used for the experiments on the testbed. Furthermore, 20 simulation runs were performed for the results in [Ome18] and are included in the average values given below.

For the experiments on the testbed, NoC, RLNC, and BCGC were implemented according to the schemes described in Sections 4.1 and 4.2 and Chapter 5, respectively, as applications on top of the operating system RIOT. RLNC was implemented based on $GF(2)$, as also used for the evaluations in Sections 7.1 and 7.2. However, in contrast to the descriptions in Sections 4.1.3, 4.2.5, and 5.4 and the used header sizes for the simulative evaluations in Sections 7.1 and 7.2, the header sizes of NoC, RLNC, and BCGC were extended by 3, 73, and 2 Bytes, respectively, for the evaluation on the testbed with 81 nodes. These extensions were required for the sake of implementation simplicity. See [Ome18] for further details. This extreme extension of the RLNC header was used for the coefficients and could be avoided if 8 coefficients are stored in one Byte and the respective relevant bit is extracted by masking, for example. For RLNC based on $GF(8)$, however, the mentioned additional header size is required in any case. With this new header size, RLNC can only be applied to a system with a maximum of 111 nodes with a small payload size of 2 Bytes without exceeding the packet size requirements of IEEE Std 802.15.4. If these packet size requirements are not met, packets are not sent in the FIT IoT-LAB testbed with the considered configurations of the nodes. However, the 81 nodes used here did not pose a problem for RLNC with the stated settings. For BCGC, an upper bound of *maxDeg* $= 54$ was used for the following results of the experiments on the testbed. This is the maximum upper bound for BCGC with which the packet sizes requirements of IEEE Std 802.15.4 are still fulfilled under the conditions considered here. Furthermore, the approach of a desired symbol, as described in Section 5.3.2, was used for BCGC for the following testbed experiments. Further implementation details can be found in [Ome18].

In order to be able to compare the results of the testbed with those of the simulator in the following, some of the default values used in the simulator had to be changed. First, the number of nodes/distinct original symbols had to be set to 81. Second, concerning the size of the simulation area, the side length of the square simulation area had to be changed to 19 units of length in order to create a situation where a node under the given conditions had on average 24 neighbors. With this change, all other settings of the radio transmission scheme in the simulator could be left at the default values. Only the supply current in transmit state TX had to be adapted separately as $P_t = -17\,\text{dBm}$ instead of $P_t = 0\,\text{dBm}$ was used as transmit power for the experiments on the testbed. The new supply current was, therefore, 7.4 mA, see [Atm09]. This value is required for the estimation of the energy consumption by the simulator. Third, for the implemented MAC protocol in the simulator, *macMaxCSMAbackoffs* $= 0$ and $t_{RX} = 15 \cdot t_p$ had to be used instead of the respective default values. Fourth, in order to compensate for the different header sizes in the simulator compared to the testbed, a phantom payload of a corresponding size was added to each packet in the simulator for the subsequent comparison. Fifth, instead of the default setting, a static network was used in the simulator for the following results in order to simulate similar conditions as in the testbed. Sixth, BCGC were simulated with the approach of requesting a desired symbol as described in Section 5.3.2.

8.3 Comparison to Simulation based Evaluation

In the following, the results of the experiments on the testbed are presented. The respective values are based on [Ome18]. Hence, the results presented here are chosen accordingly. This means, for the testbed, only two figures are shown, the average number of packets sent/received by a node until a respective point in time, see Figures 8.1 and 8.2. In these, the difference of the implemented procedures, NoC, RLNC, and BCGC, during the course of the respective experiment can be recognized. Since the number of sent and received packets differed significantly from each other, especially for NoC, they are shown in two different figures, unlike for the simulator. For the graphs, the average value over all seeds and for each seed the average value over all nodes was calculated for each considered point in time. However, the values of the individual seeds differed greatly in the experiments on the testbed. For this reason, the slope of the graphs decreases towards the end when only a few remaining seeds have an influence on the average value and these seeds have a lower value. Furthermore, regarding the sent packets, in Tables 8.1 to 8.3, a different type of value is given than in Sections 7.1 and 7.2 for the simulation based evaluation. Here, it is indicated how many packets a node had to send on average until all nodes were finished. Previously, on the other hand, the average number of packets a node had to send until this node itself was finished was specified in the respective tables. In the following, however, the number of packets a node had to send until the network was finished is also given for the results of the simulator at 'Sent Packets'. All other performance metrics in Tables 8.1 to 8.3 correspond to the descriptions in Section 7.1.

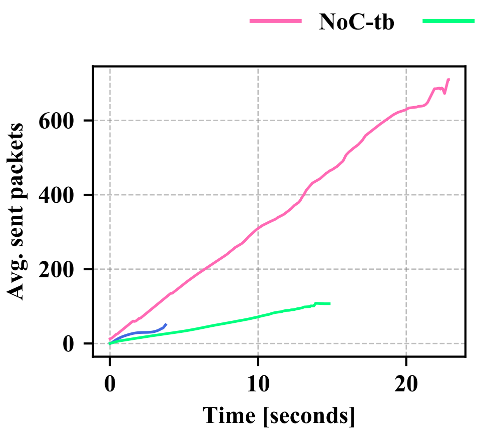


Fig. 8.1: Sent packets

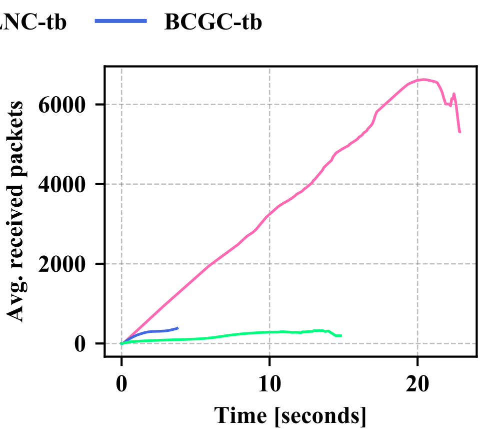


Fig. 8.2: Received packets

For the testbed based evaluation, the actual power consumption of each node was measured during the experiments and, then, the corresponding energy consumption of the network was calculated. This measurement of the power consumption refers to the Open Nodes, which run the experiments, and are executed by the Control Nodes. The measurement was performed as provided by the used FIT IoT-LAB testbed platform and as described, for example, in [Adj+15a; Fam+14]. Here, the used sampling period for the power consumption was 65.95 ms, see [Ome18]. The energy consumption specified for the testbed refers not only to the dissemination process and the associated computing in the nodes, but to the entire experiment on the nodes. In addition, a duration had to be defined in advance, for which the experiment is executed on the nodes. The power consumption of the nodes is, then, also measured for this entire duration, which thus influences the average value of the energy consumption. This duration was set to 2 min for all executed experiments in [Ome18]. In the simulator, the energy consumption is estimated. However, only the energy consumption concerning communication, i.e. caused by the radio chip, is considered and not concerning sensing or data processing, for example, as in the real testbed. Therefore, the values from the simulator are very low and differ significantly from the measured energy consumption of the testbed.

Overall, it can be seen in Figures 8.1 to 8.4 and Tables 8.1 and 8.3 that the results of the testbed and the simulator are of a similar order of magnitude for BCGC and also for RLNC, except for the energy consumption, which has been explained before. Slight differences in the results are due to small differences in the actual realization, such as the positions of the nodes and the actual implementation. Furthermore, for the experiments on the testbed, a first version of BCGC was used instead of the final version, which is presented here and used for the simulator. Therefore, in particular, the values for the RNP_{100}, the node latency, and the system latency would be lower if the

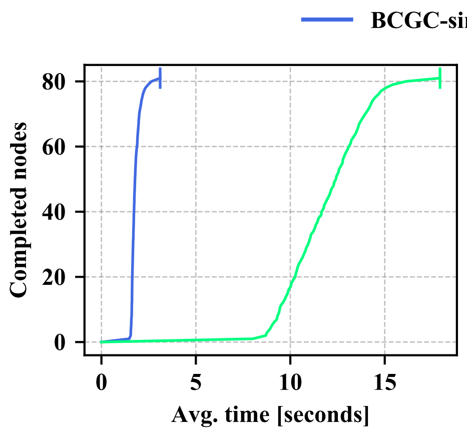


Fig. 8.3: Completed nodes

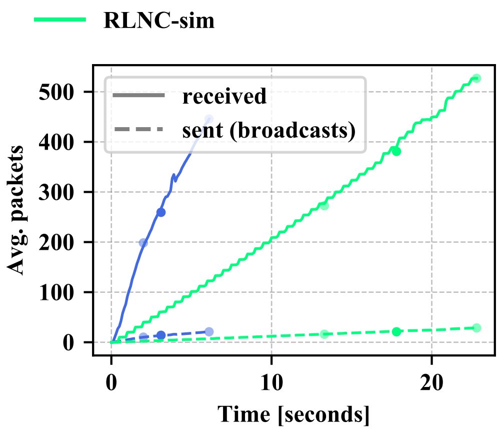


Fig. 8.4: Sent/received packets

Performance Metric	BCGC-simulator	BCGC-testbed
RNP_{100}	168	204
Node Latency $[s]$	1.8	1.0
System Latency $[s]$	3.1	3.2
PRR	$1-0.192$	$1-0.574$
Sent Packets	14	30
Energy Consumption $[J]$	0.034	13.08

Table 8.1: Performance metrics for BCGC with simulator vs. testbed

Performance Metric	NoC-simulator	NoC-testbed
RNP_{100}	n/a	2647
Node Latency $[s]$	n/a	7.9
System Latency $[s]$	n/a	10.9
PRR	n/a	$1-0.557$
Sent Packets	n/a	284
Energy Consumption $[J]$	n/a	13.11

Table 8.2: Performance metrics for NoC with simulator vs. testbed

Performance Metric	RLNC-simulator	RLNC-testbed
RNP_{100}	240	119
Node Latency $[s]$	11.8	6.2
System Latency $[s]$	17.9	8.7
PRR	$1-0.26$	$1-0.798$
Sent Packets	22	57
Energy Consumption $[J]$	0.6658	13.29

Table 8.3: Performance metrics for RLNC with simulator vs. testbed

final version of BCGC had been used for the testbed based experiment. As was already the case in Section 7.2.4, the simulation of NoC in a static network did not terminate within the given period of time. For this reason, results are omitted here. However, it can be recognized from a general comparison of the results of the testbed here and those of the simulator in Section 7.2 that the behavior of NoC in comparison to BCGC was similar. Concerning the results of the experiments on the testbed, it can be seen that node and system latency are significantly better for BCGC than for NoC or RLNC, see Tables 8.1 to 8.3. Due to the significantly larger header size for RLNC than for BCGC and the considered computational effort, the performance of BCGC concerning latency was significantly better than for RLNC in the testbed, see Figure 8.3 and Tables 8.1 and 8.3. If the RLNC header used in the testbed were not so large, as described in Section 8.2, the difference would be smaller. However, if a larger number of nodes is used, the difference will be more significant again. This is due to the fact that RLNC does not scale well with the number of nodes concerning packet size but also concerning computational effort due to the used Gaussian elimination. The measured energy consumption in the testbed was dominated by the very long run-time of the experiment, which was chosen to be 2 min for BCGC, NoC, and RLNC. Thus, the results regarding energy consumption in the testbed are not very informative.

To conclude, it is shown that BCGC are also functional in a real implementation and that the simulation results are plausible.

Chapter 9

Concluding Remarks

Contents

In this chapter, the main contributions of this thesis and the key characteristics of the herein developed network coding procedure are summarized and future work is addressed.

9.1 Contributions of this Thesis

The key contributions presented in this thesis are:

1. the development of the network coding procedure BCGC for reliable data dissemination in WSNs.
2. the implementation of a corresponding network simulator which supports among others a variety of network coding procedures, radio transmission schemes, MAC protocols, mobility models, and patterns for node failures.
3. the comprehensive evaluation of BCGC and related work by both simulations and in a real world testbed.

Broadcast Growth Codes (BCGC) were designed to enable a high degree of reliability at low latency, high throughput, and reduced energy consumption. Furthermore, BCGC are characterized by their simplicity and robustness. These key performance metrics are translated into the following procedural objectives:

- appropriate redundancy depending on the state of the network, i.e. the codeword degree of a packet/codeword to be sent next and the symbol selection for this codeword should be appropriate.

- early decoding, i.e. every node should be able to reconstruct as much original data as possible at any considered point in time and, in particular, also at an early stage.
- simple encoding and decoding to enable the use of low-cost sensor nodes which may have limited computing power.
- zero configuration, i.e. nodes should immediately start collecting, processing, and transmitting sensor data without complex initialization.

These objectives should be achieved while being agnostic to network size, payload sizes, collision rates, different patterns and degrees of node failures and should also not be dependent on a certain degree of nodes' mobility.

To this end, among others, dynamic packet sizes are used in the case of BCGC. Packet sizes are small in the beginning and grow with the used codeword degree d. However, packet sizes are bounded by an appropriately chosen upper bound for the codeword degree. This means, they do not grow linearly with the number of nodes in the system. Using dynamic packet sizes and appropriately limiting the codeword degree reduces latency and energy consumption, increases throughput, and results in more robustness and reliability. Furthermore, for encoding and decoding only simple XOR operations are used in the case of BCGC. In addition to simplicity, this also allows to start decoding packets/codewords immediately after the reception of a first packet, i.e. at a very early stage.

On the one hand, the aim to use an appropriate amount of redundancy concerning the degree of codewords means to combine enough symbols d for one packet/codeword to avoid the reception of a large number of unnecessary redundant packets. On the other hand, it also means not to combine too many symbols so that packets/codewords can still be decoded as soon as possible by the receiver and packets do not become too large at the same time. Those aspects aim to minimize the RNP, the required number of received packets to reconstruct a respective number of distinct original symbols in a considered node while keeping packet sizes short. Moreover, the used codeword degree d is adapted to the current state of the neighboring nodes and, thus, of the network in general. This is realized by the BCGC degree function, which determines a node's desired degree for a codeword to be received next depending on the number of symbols actually reconstructed in this node. For the creation of a codeword, the minimum of the received desired degrees of the neighbors is then used as the codeword degree d. In this way, no nodes are left behind so that the goal of having gathered and reconstructed as many distinct original symbols as possible in a node at any considered point in time can be met. Furthermore, the degree function was modified in order to fulfill the mentioned aims and to ensure to use an appropriate degree of redundancy. In this context, the BCGC degree function compensates for the uneven distribution of the probabilities of the original symbols to be included in a codeword in reality. In addition, the degree function does not optimize only the required number of received packets but also considers the resulting packet sizes and, thus, the resulting latency, throughput, and energy consumption.

Furthermore, using appropriate redundancy also refers to the symbol selection. When creating packets/codewords, symbols should be selected in such a way that a receiver does not depend on the successful reception of a concrete packet/codeword. Moreover, on the one hand, not too many symbols which are probably already known by the receivers should be combined for a codeword if d symbols are to be combined in order to avoid the reception of a large number of unnecessary redundant packets. On the other hand, also not too few probably already known symbols and too many probably still unknown symbols should be combined so that codewords can still be decoded as soon as possible using only simple XOR operations. In general, BCGC use a pro-active symbol selection, which means codewords are composed specifically in order to forward preferentially rare symbols and use an appropriate amount of redundancy.

To conclude, using appropriate redundancy for BCGC ensures that early decoding is possible, thus, contributes to an enhancement of reliability, increases the throughput, and reduces latency as well as energy consumption.

Through simulation based evaluations and evaluations performed in a testbed, it was shown that BCGC fulfill the mentioned aims even under challenging conditions. BCGC are robust to changes in the BCGC process parameters or simulation settings and enable reliable data dissemination. In terms of latency, throughput, and energy consumption, BCGC can achieve the same performance or even outperform existing procedures while allowing low complexity of the nodes as well as early decoding. For the considered default simulation settings, the coding gain of BCGC with the default configuration of the BCGC process parameters compared to no encoding of data for transmission, i.e. compared to pure forwarding of data, was almost 12 and the throughput gain almost 9. The RNP gain compared to RLNC was only about 0.6 and the throughput gain also only about 0.6. However, RLNC generally requires significantly more computational effort for decoding using Gaussian elimination and early decoding is not possible. Furthermore, the size of RLNC packet headers grows linearly with the number of nodes/symbols in the system and becomes very large in the case of a large number of nodes. Compared to GC, the RNP gain of BCGC was about 22 and the throughput gain about 24 for the considered default simulation settings. Thus, BCGC outperformed GC, which served as a basis for BCGC, significantly.

9.2 Future Work

It was shown that, even with a non-optimized default configuration of the BCGC process parameters, BCGC perform well in comparison to related work for the default settings as well as for variations of these settings. Furthermore, BCGC were robust against changes in the BCGC process parameters. However, it was not a goal here to find an optimal combination for the configurations of these BCGC process parameters. Instead, the impact of each parameter was investigated individually. Therefore, a first aspect which is an option for future work is to find an optimal configuration of the BCGC process parameters for a specific use case and to extend this analysis to further use cases.

In addition, the approach of dynamically adapting the BCGC process parameters to the perceived dynamics could be pursued. In this regard, an optimization of the BCGC degree function depending on the respective observed degree of nodes' mobility or node density is an option for future work.

Furthermore, the behavior of BCGC could be investigated if, instead of specifying the number of nodes n, n is estimated and then an estimate of n is used for BCGC. In this case, the effects of an incorrect estimation of n in BCGC could be analyzed as well as how much accuracy is necessary to meet certain given performance criteria.

Here, it is assumed that nodes generate sensor data only once and then distribute this data over the entire network. Therefore, another aspect suggested for future work is the consideration of multiple generations of data for BCGC, i.e. each node generates sensor data repeatedly. In this case, it may be beneficial to perform duty cycling in between generations.

Concerning the used routing method, a form of flooding is assumed as basis for BCGC here. This simple routing procedure can easily be exchanged. An analysis of which routing procedure can be supplemented by BCGC and which advantages the respective combination brings is, therefore, an option for future work. In this context, BCGC could be implemented on top of a more advanced routing protocol or even be extended to some form of routing-aware coding.

Bibliography

[3GP20a] 3GPP. **Release 13**. 2016 (accessed February 20, 2020). URL: `https://www.3gpp.org/release-13` (cit. on p. 49).

[3GP20b] 3GPP. **Release 15**. 2018 (accessed February 20, 2020). URL: `https://www.3gpp.org/release-15` (cit. on p. 49).

[Abr70] N. Abramson. **The ALOHA System: Another Alternative for Computer Communications**. In: *Proceedings of the November 17-19, 1970, Fall Joint Computer Conference*. Nov. 1970, pp. 281–285 (cit. on p. 143).

[Ace+05] S. Acedanski, S. Deb, M. Médard, and R. Koetter. **How good is Random Linear Coding Based Distributed Networked Storage**. In: *Workshop on Network Coding, Theory and Applications*. 2005, pp. 1–6 (cit. on p. 93).

[Adj+15a] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noël, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, et al. **FIT IoT-LAB: A Large Scale Open Experimental IoT Testbed**. In: *2nd World Forum on Internet of Things (WF-IoT)*. IEEE. Dec. 2015, pp. 459–464. DOI: `10.1109/WF-IoT.2015.7389098` (cit. on pp. 54, 247).

[Adj+15b] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne. **FIT IoT-LAB: A Large Scale Open Experimental IoT Testbed**. In: *IEEE World Forum on Internet of Things (IEEE WF-IoT)*. Milan, Italy, Dec. 2015. URL: `https://hal.inria.fr/hal-01213938` (cit. on pp. 46, 243).

[AGM08] N. Aschenbruck, E. Gerhards-Padilla, and P. Martini. **A Survey on Mobility Models for Performance Analysis in Tactical Mobile Networks**. In: *Journal of Telecommunications and Information Technology* (2008), pp. 54–61 (cit. on p. 149).

[Ahl+00] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung. **Network Information Flow**. In: *IEEE Transactions on Information Theory* 46.4 (2000), pp. 1204–1216 (cit. on pp. 11, 15, 18, 19, 28–30).

[AK04] J. N. Al-Karaki and A. E. Kamal. **Routing Techniques in Wireless Sensor Networks: A Survey**. In: *IEEE Wireless Communications* 11.6 (2004), pp. 6–28 (cit. on pp. 24, 25, 27).

[AKS08a] S. A. Aly, Z. Kong, and E. Soljanin. **Fountain Codes Based Distributed Storage Algorithms for Large-Scale Wireless Sensor Networks**. In: *International Conference on Information Processing in Sensor Networks (ipsn 2008)*. IEEE. 2008, pp. 171–182 (cit. on pp. 93, 94).

[AKS08b] S. A. Aly, Z. Kong, and E. Soljanin. **Raptor Codes Based Distributed Storage Algorithms for Wireless Sensor Networks**. In: *2008 IEEE International Symposium on Information Theory*. IEEE. 2008, pp. 2051–2055 (cit. on p. 93).

[Ala14] M. Alani. **Guide to OSI and TCP/IP Models**. Jan. 2014. DOI: `10.1007/978-3-319-05152-9` (cit. on p. 45).

[Alf19] R. L. Alfvin. **IEEE 802.15 Working Group on Wireless Specialty Networks**. 2018 (accessed September 9, 2019). URL: `http://www.ieee802.org/15/about.html` (cit. on p. 45).

[AM12] J. B. Anderson and S. Mohan. **Source and Channel Coding: An Algorithmic Approach**. Vol. 150. Springer Science & Business Media, 2012 (cit. on p. 20).

[Ana+09] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella. **Energy Conservation in Wireless Sensor Networks: A Survey**. In: *Ad Hoc Networks* 7.3 (2009), pp. 537–568 (cit. on pp. 54, 55, 143, 157).

[ARH16] N. Ahmed, H. Rahman, and M. I. Hussain. **A Comparison of 802.11ah and 802.15.4 for IoT**. In: *Ict Express* 2.3 (2016), pp. 100–102 (cit. on p. 49).

[Atm09] Atmel. **AVR Low Power 2.4 GHz Transceiver for ZigBee, IEEE 802.15.4, 6LoWPAN, RF4CE, SP100, WirelessHART, and ISM Applications, AT86RF231**. Sept. 2009 (cit. on pp. 46, 156, 244, 246).

[AV10] I. F. Akyildiz and M. C. Vuran. **Wireless Sensor Networks**. Vol. 4. John Wiley & Sons, 2010 (cit. on p. 1).

[BAL10] M. Borokhovich, C. Avin, and Z. Lotker. **Tight Bounds for Algebraic Gossip on Graphs**. In: *IEEE International Symposium on Information Theory*. IEEE. 2010, pp. 1758–1762 (cit. on pp. 97, 104).

[BAL14] M. Borokhovich, C. Avin, and Z. Lotker. **Bounds for Algebraic Gossip on Graphs**. In: *Random Structures & Algorithms* 45.2 (2014), pp. 185–217 (cit. on pp. 97, 104).

[Bar+07] P. Baronti, P. Pillai, V. W. Chook, S. Chessa, A. Gotta, and Y. F. Hu. **Wireless Sensor Networks: A Survey on the State of the Art and the 802.15.4 and ZigBee Standards**. In: *Computer Communications* 30.7 (2007), 1655–1695 (cit. on p. 45).

[BDF19] Y.-D. Bromberg, Q. Dufour, and D. Frey. **Multisource Rumor Spreading with Network Coding**. In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. IEEE. 2019, pp. 2359–2367 (cit. on pp. 100, 104).

[Ber87] R. Bernhardt. **Macroscopic Diversity in Frequency Reuse Radio Systems**. In: *IEEE Journal on Selected Areas in Communications* 5.5 (1987), pp. 862–870 (cit. on p. 140).

[Beu13] A. Beutelspacher. **Lineare Algebra: Eine Einführung in die Wissenschaft der Vektoren, Abbildungen und Matrizen**. Springer-Verlag, 2013 (cit. on p. 63).

[BGH92] D. P. Bertsekas, R. G. Gallager, and P. Humblet. **Data Networks**. Vol. 2. Prentice-Hall International New Jersey, 1992 (cit. on p. 21).

[BH04] F. Bai and A. Helmy. **A Survey of Mobility Models in Wireless Adhoc Networks**. In: (2004) (cit. on pp. 149–152).

[Bla06] A. G. Blank. **TCP/IP Foundations**. Wiley, Feb. 2006 (cit. on p. 45).

[Blu19a] Bluetooth-SIG. **Bluetooth Core Specification Version 5.2**. Dec. 2019 (cit. on p. 49).

[Blu19b] Bluetooth-SIG. **Mesh Profile Specification Version 1.0.1**. Jan. 2019 (cit. on p. 49).

[BM04] S. Biswas and R. Morris. **Opportunistic Routing in Multi-Hop Wireless Networks**. In: *ACM SIGCOMM Computer Communication Review* 34.1 (2004), pp. 69–74 (cit. on p. 25).

[BN05] K. Bhattad and K. R. Narayanan. **Weakly Secure Network Coding**. In: *1st Workshop on Network Coding, Theory, and Applications (NetCod)* (Apr. 2005) (cit. on p. 31).

[Bos13] M. Bossert. **Kanalcodierung**. Walter de Gruyter, 2013 (cit. on pp. 20, 22).

[BQW16] M. G. Ball, B. Qela, and S. Wesolkowski. **A Review of the Use of Computational Intelligence in the Design of Military Surveillance Networks**. In: *Recent Advances in Computational Intelligence in Defense and Security*. Springer, 2016, pp. 663–693 (cit. on p. 1).

[BRS03] C. Bettstetter, G. Resta, and P. Santi. **The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks**. In: *IEEE Transactions on Mobile Computing* 2.3 (2003), pp. 257–269 (cit. on p. 150).

[BRS04] D. M. Blough, G. Resta, and P. Santi. **A Statistical Analysis of the Long-Run Node Spatial Distribution in Mobile Ad Hoc Networks**. In: *Wireless Networks* 10.5 (2004), pp. 543–554 (cit. on pp. 150, 151).

[Bru19] C. W. Bruno Johnson Alin Lazar. **Thread 1.2 Base Features**. Tech. rep. Thread Group, June 2019 (cit. on p. 49).

[Bue+06] M. Buettner, G. V. Yee, E. Anderson, and R. Han. **X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks**. In: *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*. ACM. 2006, pp. 307–320 (cit. on p. 144).

[Bye+98] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. **A Digital Fountain Approach to Reliable Distribution of Bulk Data**. In: *ACM SIGCOMM Computer Communication Review* 28.4 (1998), pp. 56–67 (cit. on p. 22).

[Car+13] R. C. Carrano, D. Passos, L. C. Magalhaes, and C. V. Albuquerque. **Survey and Taxonomy of Duty Cycling Mechanisms in Wireless Sensor Networks**. In: *IEEE Communications Surveys & Tutorials* 16.1 (2013), pp. 181–194 (cit. on p. 157).

[CBD02] T. Camp, J. Boleng, and V. Davies. **A Survey of Mobility Models for Ad Hoc Network Research**. In: *Wireless Communications and Mobile Computing* 2.5 (2002), pp. 483–502 (cit. on p. 149).

[CC11] N. Cai and T. Chan. **Theory of Secure Network Soding**. In: *Proceedings of the IEEE* 99.3 (2011), pp. 421–437 (cit. on p. 31).

[CC81] G. C. Clark Jr and J. B. Cain. **Error-Correction Coding for Digital Communications**. Plenum Press, 1981 (cit. on p. 22).

[CFS06] C. Chekuri, C. Fragouli, and E. Soljanin. **On Average Throughput and Alphabet Size in Network Coding**. In: *IEEE Transactions on Information Theory* 52.6 (2006), pp. 2410–2424 (cit. on p. 28).

[Cha+07] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. **Trading Structure for Randomness in Wireless Opportunistic Routing**. In: *ACM SIGCOMM Computer Communication Review* 37.4 (2007), pp. 169–180 (cit. on pp. 11, 27).

[Che+15] J. Chen, K. He, R. Du, M. Zheng, Y. Xiang, and Q. Yuan. **Dominating Set and Network Coding-Based Routing in Wireless Mesh Networks**. In: *IEEE Transactions on Parallel and Distributed Systems* 26.2 (Feb. 2015), pp. 423–433. DOI: `10.1109/tpds.2013.303` (cit. on p. 27).

[ČHE02] M. Čagalj, J.-P. Hubaux, and C. Enz. **Minimum-Energy Broadcast in All-Wireless Networks: NP-Completeness and Distribution Issues**. In: *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*. 2002, pp. 172–182 (cit. on p. 29).

[Che04] W. K. Chen. **The Electrical Engineering Handbook**. Elsevier, 2004 (cit. on p. 46).

[CMN84] D. C. Cox, R. R. Murray, and A. Norris. **800-MHz Attenuation Measured In and Around Suburban Houses**. In: *AT&T Bell Laboratories Technical Journal* 63.6 (1984), pp. 921–954 (cit. on p. 140).

[Cos+08] R. A. Costa, D. Munaretto, J. Widmer, and J. Barros. **Informed Network Coding for Minimum Decoding Delay**. In: *5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems*. IEEE. 2008, pp. 80–91 (cit. on pp. 101, 102, 104).

[CT11] Y. Chen and A. Terzis. **On the Implications of the Log-Normal Path Loss Model: An Efficient Method to Deploy and Move Sensor Motes**. In: *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*. 2011, pp. 26–39 (cit. on p. 141).

[CW07] P. A. Chou and Y. Wu. **Network Coding for the Internet and Wireless Networks**. In: *IEEE Signal Processing Magazine* 24.5 (2007), pp. 77–85 (cit. on pp. 5, 29, 30, 69).

[CWJ03] P. Chou, Y. Wu, and K. Jain. **Practical Network Coding**. In: *Proceedings of the Annual Allerton Conference on Communication, Control, and Computing*. Vol. 41. 1. Oct. 2003, pp. 40–49 (cit. on pp. 19, 28, 95, 104).

[DAm19] J. D'Ambrosia. **IEEE 802 LAN/MAN Standards Committee**. 2019 (accessed September 20, 2019). URL: `http://ieee802.org/` (cit. on p. 45).

[Dea12] T. Dean. **Network+ Guide to Networks**. Cengage Learning, 2012 (cit. on p. 51).

[Deb+05] S. Deb, M. Effros, T. Ho, D. R. Karger, R. Koetter, D. S. Lun, M. Médard, and N. Ratnakar. **Network Coding for Wireless Applications: A Brief Tutorial**. In: IWWAN. 2005 (cit. on pp. 19, 28).

[Dez+15] B. Dezfouli, M. Radi, S. Abd Razak, T. Hwee-Pink, and K. A. Bakar. **Modeling Low-Power Wireless Communications**. In: *Journal of Network and Computer Applications* 51 (2015), pp. 102–126 (cit. on p. 141).

[DF56] G. Dantzig and D. Fulkerson. **On the Max-Flow Min-Cut Theorem of Networks**. In: *Linear Inequalities, Ann. Math. Studies* 38 (1956) (cit. on p. 18).

[DFZ05] R. Dougherty, C. Freiling, and K. Zeger. **Insufficiency of Linear Coding in Network Information Flow**. In: *IEEE Transactions on Information Theory* 51.8 (2005), pp. 2745–2759 (cit. on pp. 12, 30).

[DIG10] M. Di Renzo, M. Iezzi, and F. Graziosi. **Beyond Routing via Network Coding: An Overview of Fundamental Information-Theoretic Results**. In: *21st Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*. IEEE. 2010, pp. 2745–2750 (cit. on pp. 5, 8, 24, 30).

[Dij59] E. W. Dijkstra. **A Note on Two Problems in Connexion with Graphs**. In: *Numerische Mathematik* 1.1 (1959), pp. 269–271 (cit. on p. 152).

[Dim+10a] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran. **Network Coding for Distributed Storage Systems**. In: *IEEE transactions on information theory* 56.9 (2010), pp. 4539–4551 (cit. on p. 93).

[Dim+10b] A. G. Dimakis, S. Kar, J. M. Moura, M. G. Rabbat, and A. Scaglione. **Gossip Algorithms for Distributed Signal Processing**. In: *Proceedings of the IEEE* 98.11 (2010), pp. 1847–1864 (cit. on p. 25).

[Dim+11] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh. **A Survey on Network Codes for Distributed Storage**. In: *Proceedings of the IEEE* 99.3 (2011), pp. 476–489 (cit. on p. 93).

[DMC04] S. Deb, M. Médard, and C. Choute. **Algebraic Gossip: A Network Coding Approach to Optimal Multiple Rumor Mongering**. In: *Proceedings of the Annual Allerton Conference on Communication, Control, and Computing*. Vol. 42. IEEE. 2004 (cit. on pp. 97, 104).

[DMC05] S. Deb, M. Médard, and C. Choute. **On Random Network Coding Based Information Dissemination**. In: *Proceedings. International Symposium on Information Theory (ISIT)*. IEEE. 2005, pp. 278–282 (cit. on pp. 97, 104).

[DMC06] S. Deb, M. Médard, and C. Choute. **Algebraic Gossip: A Network Coding Approach to Optimal Multiple Rumor Mongering**. In: *IEEE Transactions on Information Theory* 52.6 (2006), pp. 2486–2507 (cit. on pp. 64, 97, 104).

[DPR05] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran. **Ubiquitous Access to Distributed Data in Large-Scale Sensor Networks Through Decentralized Erasure Codes**. In: *IPSN 2005. 4th International Symposium on Information Processing in Sensor Networks*. IEEE. 2005, pp. 111–117 (cit. on p. 93).

[DPR06] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran. **Decentralized Erasure Codes for Distributed Networked Storage**. In: *IEEE Transactions on Information Theory* 52.6 (2006), pp. 2809–2816 (cit. on pp. 93, 94).

[EFS56] P. Elias, A. Feinstein, and C. Shannon. **A Note on the Maximum Flow Through a Network**. In: *IRE Transactions on Information Theory* 2.4 (1956), pp. 117–119 (cit. on p. 18).

[EOM06] A. Eryilmaz, A. Ozdaglar, and M. Medard. **On Delay Performance Gains from Network Coding**. In: *40th Annual Conference on Information Sciences and Systems*. IEEE. 2006, pp. 864–870 (cit. on p. 68).

[Erg04] S. C. Ergen. **ZigBee/IEEE 802.15.4 Summary**. In: *UC Berkeley, September* 10 (2004), p. 17 (cit. on p. 45).

[Fad+15] E. Fadel, V. Gungor, L. Nassef, N. Akkari, M. A. Maik, S. Almasri, and I. F. Akyildiz. **A Survey on Wireless Sensor Networks for Smart Grid**. In: *Computer Communications* 71 (2015), pp. 22–33 (cit. on p. 1).

[Fam+14] O. Fambon, E. Fleury, G. Harter, R. Pissard-Gibollet, and F. Saint-Marcel. **FIT IoT-LAB Tutorial: Hands-On Practice With a Very Large Scale Testbed Tool for the Internet of Things**. In: *10èmes journées francophones Mobilité et Ubiquité, UbiMob2014* (2014) (cit. on pp. 54, 247).

[FF56] L. Ford and D. Fulkerson. **Maximal Flow Through a Network**. In: *Canadian Journal of Mathematics* 8 (1956), pp. 399–404 (cit. on p. 18).

[FLW06] C. Fragouli, J.-Y. Le Boudec, and J. Widmer. **Network Coding: An Instant Primer**. In: *ACM SIGCOMM Computer Communication Review* 36.1 (2006), pp. 63–68 (cit. on pp. 5, 55, 69).

[For16] A. Forster. **Introduction to Wireless Sensor Networks**. John Wiley & Sons, 2016 (cit. on p. 1).

[Fra+07] C. Fragouli, D. Katabi, A. Markopoulou, M. Medard, and H. Rahul. **Wireless Network Coding: Opportunities & Challenges**. In: *IEEE Military Communications Conference (MILCOM)*. IEEE. 2007, pp. 1–8 (cit. on p. 30).

[Fri13] B. Friedrichs. **Kanalcodierung: Grundlagen und Anwendungen in modernen Kommunikationssystemen**. Springer-Verlag, 2013 (cit. on p. 20).

[Fri46] H. T. Friis. **A Note on a Simple Transmission Formula**. In: *Proceedings of the IRE* 34.5 (1946), pp. 254–256 (cit. on p. 138).

[FRS09] H. Frey, S. Rührup, and I. Stojmenović. **Routing in Wireless Sensor Networks**. In: *Guide to Wireless Sensor Networks*. Springer London, 2009, pp. 81–111. DOI: `10.1007/978-1-84882-218-4_4` (cit. on p. 24).

[FS+07] C. Fragouli, E. Soljanin, et al. **Network Coding Fundamentals**. In: *Foundations and Trends® in Networking* 2.1 (2007), pp. 1–133 (cit. on pp. 5, 15, 18).

[FS+08] C. Fragouli, E. Soljanin, et al. **Network Coding Applications**. In: *Foundations and Trends® in Networking* 2.2 (2008), pp. 135–269 (cit. on p. 5).

[FWL05] C. Fragouli, J. Widmer, and J.-Y. Le Boudec. **A Network Coding Approach to Energy Efficient Broadcasting: From Theory to Practice**. Tech. rep. 2005 (cit. on pp. 98, 99, 104).

[FWL06] C. Fragouli, J. Widmer, and J.-Y. Le Boudec. **On the Benefits of Network Coding for Wireless Applications**. In: *4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*. IEEE. 2006, pp. 1–6 (cit. on pp. 19, 29, 98, 99, 104).

[FWL08] C. Fragouli, J. Widmer, and J.-Y. Le Boudec. **Efficient Broadcasting Using Network Coding**. In: *IEEE/ACM Transactions on Networking* 16.2 (2008), pp. 450–463 (cit. on pp. 10, 29, 54, 98, 99, 104).

[Gar10] V. Garg. **Wireless Communications & Networking**. Elsevier, 2010 (cit. on pp. 54, 139, 141, 156).

[GC19] J. H. G. Montenegro N. Kushalnagar and D. Culler. **Transmission of IPv6 Packets over IEEE 802.15.4 Networks**. RFC 4944. RFC Editor, Sept. 2007 (accessed November 21, 2019). DOI: `10.17487/RFC4944`. URL: `https://www.rfc-editor.org/info/rfc4944` (cit. on p. 45).

[GHK15] I. M. Grimm, M. van gen Hassend, and R. Kolla. **Ausfalltolerante Datenhaltung durch Multicast Growth Codes**. In: *14. GI/ITG KuVS Fachgespräch Sensornetze* (Sept. 2015), pp. 49–52 (cit. on p. 106).

[GHK16] I. M. Grimm, M. van gen Hassend, and R. Kolla. **Zuverlässige Datenhaltung in unzuverlässigen Netzen**. In: *15. GI/ITG KuVS Fachgespräch Sensornetze* (Sept. 2016), pp. 22–26 (cit. on p. 106).

[GHZ12] W. Guo, W. M. Healy, and M. Zhou. **Impacts of 2.4-GHz ISM Band Interference on IEEE 802.15.4 Wireless Sensor Network Reliability in Buildings**. In: *IEEE Transactions on Instrumentation and Measurement* 61.9 (2012), pp. 2533–2544 (cit. on p. 51).

[GK15] R. Gessler and T. Krause. **Wireless-Netzwerke für den Nahbereich**. Verlag Springer, 2015 (cit. on p. 49).

[GK17] I. M. Grimm and R. Kolla. **Einfluss der Gradverteilung bei Multicast Growth Codes**. In: *16. GI/ITG KuVS Fachgespräch Sensornetze* (Sept. 2017), pp. 17–21 (cit. on p. 106).

[GMS15] J. Granjal, E. Monteiro, and J. S. Silva. **Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues**. In: *IEEE Communications Surveys & Tutorials* 17.3 (2015), pp. 1294–1312 (cit. on p. 47).

[GR05] C. Gkantsidis and P. R. Rodriguez. **Network Coding for Large Scale Content Distribution**. In: *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies*. Vol. 4. IEEE. 2005, pp. 2235–2245 (cit. on pp. 28, 95, 104).

[Gra12] R. M. Gray. **Source Coding Theory**. Vol. 83. Springer Science & Business Media, 2012 (cit. on p. 20).

[GTK08] M. Ghaderi, D. Towsley, and J. Kurose. **Reliability Gain of Network Coding in Lossy Wireless Networks**. In: *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*. IEEE. 2008, pp. 2171–2179 (cit. on p. 30).

[Guo+12] Z. Guo, J. Huang, B. Wang, S. Zhou, J.-H. Cui, and P. Willett. **A Practical Joint Network-Channel Coding Scheme for Reliable Communication in Wireless Networks**. In: *IEEE Transactions on Wireless Communications* 11.6 (2012), pp. 2084–2094 (cit. on p. 23).

[Ha+07] J. Y. Ha, T. H. Kim, H. S. Park, S. Choi, and W. H. Kwon. **An Enhanced CSMA-CA Algorithm for IEEE 802.15.4 LR-WPANs**. In: *IEEE Communications Letters* 11.5 (2007), pp. 461–463 (cit. on p. 146).

[Hae11] B. Haeupler. **Analyzing Network Coding Gossip Made Easy**. In: *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*. 2011, pp. 293–302 (cit. on pp. 97, 98, 104).

[Hae16] B. Haeupler. **Analyzing Network Coding (Gossip) Made Easy**. In: *Journal of the ACM (JACM)* 63.3 (2016), pp. 1–22 (cit. on pp. 97, 98, 104).

[Hen+08] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena. **Network Simulations with the ns-3 Simulator**. In: *SIGCOMM Demonstration* 14.14 (2008), p. 527 (cit. on p. 137).

[Hen08] N. Henze. **Stochastik für Einsteiger: eine Einführung in die faszinierende Welt des Zufalls**. Springer-Verlag, 2008 (cit. on pp. 59, 76, 84, 85, 89).

[HHL02] Z. J. Haas, J. Y. Halpern, and L. Li. **Gossip-Based Ad Hoc Routing**. In: *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies*. Vol. 3. IEEE. 2002, pp. 1707–1716 (cit. on p. 25).

[HK05] T. Ho and R. Koetter. **Online Incremental Network Coding for Multiple Unicasts**. In: *DIMACS Working Group on Network Coding* (2005), pp. 26–28 (cit. on pp. 19, 29).

[HK11] B. Haeupler and D. Karger. **Faster Information Dissemination in Dynamic Networks via Network Coding**. In: *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*. 2011, pp. 381–390 (cit. on pp. 98, 104).

[HKB99] W. R. Heinzelman, J. Kulik, and H. Balakrishnan. **Adaptive Protocols for Information Dissemination in Wireless Sensor Networks**. In: *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*. 1999, pp. 174–185 (cit. on p. 25).

[HL08] T. Ho and D. Lun. **Network Coding: An Introduction**. Cambridge University Press, 2008 (cit. on pp. 5, 8, 9, 15, 29, 31, 73).

[Ho+03a] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros. **The Benefits of Coding over Routing in a Randomized Setting**. In: *Proceedings of the IEEE International Symposium on Information Theory*. IEEE. 2003, p. 442 (cit. on pp. 11, 19, 62, 64, 67, 96, 97, 104).

[Ho+03b] T. Ho, M. Medard, J. Shi, M. Effros, and D. R. Karger. **On Randomized Network Coding**. In: *Proceedings of the Annual Allerton Conference on Communication, Control, and Computing*. Vol. 41. 1. Citeseer. 2003, pp. 11–20 (cit. on p. 67).

[Ho+04] T. Ho, M. Médard, M. Effros, R. Koetter, and D. Karger. **Network Coding for Correlated Sources**. In: CISS. 2004 (cit. on p. 23).

[Ho+06] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong. **A Random Linear Network Coding Approach to Multicast**. In: *IEEE Transactions on Information Theory* 52.10 (2006), pp. 4413–4430 (cit. on pp. 19, 67, 71, 96, 104).

[HSR16] D. C. Harrison, W. K. Seah, and R. Rayudu. **Rare Event Detection and Propagation in Wireless Sensor Networks**. In: *ACM Computing Surveys (CSUR)* 48.4 (2016), p. 58 (cit. on p. 1).

[Hua+14] Q. Huang, K. Sun, X. Li, and D. O. Wu. **Just Fun: A Joint Fountain Coding and Network Coding Approach to Loss-Tolerant Information Spreading**. In: *Proceedings of the 15th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. ACM. 2014, pp. 83–92 (cit. on pp. 11, 23).

[IEE03] IEEE. **IEEE Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 15: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPAN)**. In: *IEEE Std 802.15.4-2003* (Oct. 2003), pp. 1–680. DOI: 10.1109/IEEESTD.2003.94389 (cit. on pp. 45, 46).

[IEE06] IEEE. **IEEE Standard for Information Technology - Local and Metropolitan Area Networks - Specific Requirements - Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)**. In: *IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003)* (Sept. 2006), pp. 1–320 (cit. on pp. 46, 119, 159).

[IEE09] IEEE. **IEEE Standard for Information Technology - Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput**. In: *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as Amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)* (Oct. 2009), pp. 1–565. DOI: 10.1109/IEEESTD.2009.5307322 (cit. on p. 49).

[IEE11] IEEE. **IEEE Standard for Local and Metropolitan Area Networks - Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)**. In: *IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)* (Sept. 2011), pp. 1–314. DOI: 10.1109/IEEESTD.2011.6012487 (cit. on p. 46).

[IEE13] IEEE. **IEEE Standard for Information Technology - Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz**. In: *IEEE Std 802.11ac-2013 (Amendment to IEEE Std 802.11-2012, as Amended by IEEE Std 802.11ae-2012, IEEE Std 802.11aa-2012, and IEEE Std 802.11ad-2012)* (Dec. 2013), pp. 1–425. DOI: 10.1109/IEEESTD.2013.6687187 (cit. on p. 49).

[IEE14] IEEE. **IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture**. In: *IEEE Std 802-2014 (Revision to IEEE Std 802-2001)* (June 2014), pp. 1–74. DOI: 10.1109/IEEESTD.2014.6847097 (cit. on p. 46).

[IEE16a] IEEE. **IEEE Standard for Information Technology — Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks — Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications**. In: *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)* (Dec. 2016), pp. 1–3534. DOI: 10.1109/IEEESTD.2016.7786995 (cit. on pp. 45, 49).

[IEE16b] IEEE. **IEEE Standard for Low-Rate Wireless Networks**. In: *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)* (Apr. 2016), pp. 1–709. DOI: 10.1109/IEEESTD.2016.7460875 (cit. on pp. 45–47, 126, 144–146, 160).

[IEE17] IEEE. **IEEE Standard for Information Technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Sub 1 GHz License Exempt Operation**. In: *IEEE Std 802.11ah-2016 (Amendment to IEEE Std 802.11-2016, as amended by IEEE Std 802.11ai-2016)* (May 2017), pp. 1–594. DOI: 10.1109/IEEESTD.2017.7920364 (cit. on p. 49).

[IEE18] IEEE. **IEEE Standard for Ethernet**. In: *IEEE Std 802.3-2018 (Revision of IEEE Std 802.3-2015)* (Aug. 2018), pp. 1–5600. DOI: 10.1109/IEEESTD.2018.8457469 (cit. on p. 45).

[IEE21] IEEE. **IEEE Standard for Information Technology - Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Enhancements for High-Efficiency WLAN**. In: *IEEE Std 802.11ax-2021 (Amendment to IEEE Std 802.11-2020)* (2021), pp. 1–767. DOI: 10.1109/IEEESTD.2021.9442429 (cit. on p. 49).

[Int11] International-Society-of-Automation-(ISA). **ANSI/ISA-100.11a-2011 Wireless Systems for Industrial Automation: Process Control and Related Applications**. 2011 (cit. on p. 49).

[Int15] International-Electrotechnical-Commission-(IEC). **IEC 62601:2015 Industrial Networks - Wireless Communication Network and Communication Profiles - WIA-PA**. 2015 (cit. on p. 49).

[Int16] International-Electrotechnical-Commission-(IEC). **IEC 62591:2016 Industrial Networks - Wireless Communication Network and Communication Profiles - WirelessHART**. 2016 (cit. on p. 49).

[ISO94] ISO/IEC. **ISO/IEC 7498-1:1994, Information Technology — Open Systems Interconnection — Basic Reference Model: The Basic Model**. Beuth, 1994 (cit. on p. 45).

[Jag+05] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. M. Tolhuizen. **Polynomial Time Algorithms for Multicast Network Code Construction**. In: *IEEE Transactions on Information Theory* 51.6 (2005), pp. 1973–1982 (cit. on pp. 28, 29).

[JCJ03] S. Jaggi, P. A. Chou, and K. Jain. **Low Complexity Algebraic Multicast Network Codes**. In: *IEEE International Symposium on Information Theory. Proceedings.* IEEE. 2003, pp. 368–368 (cit. on p. 19).

[JL12] S. Jaggi and M. Langberg. **Secure Network Coding**. In: *Network Coding*. Elsevier, 2012, pp. 183–215. DOI: 10.1016/b978-0-12-380918-6.00007-x (cit. on p. 31).

[JM96] D. B. Johnson and D. A. Maltz. **Dynamic Source Routing in Ad Hoc Wireless Networks**. In: *Mobile computing*. Springer, 1996, pp. 153–181 (cit. on p. 150).

[Kam+06a] A. Kamra, J. Feldman, V. Misra, and D. Rubenstein. **Data Persistence for Zero-Configuration Sensor Networks**. In: *ACM Special Interest Group on Data Communications (SIGCOMM)*. 2006 (cit. on pp. 73–77, 82, 83, 85–88, 111, 112, 119).

[Kam+06b] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein. **Growth Codes: Maximizing Sensor Network Data Persistence**. In: *ACM SIGCOMM Computer Communication Review*. Vol. 36. 4. ACM. 2006, pp. 255–266 (cit. on pp. 73–79, 82, 83, 85–88, 111, 112, 119, 196).

[Kat+05] S. Katti, D. Katabi, W. Hu, H. S. Rahul, and M. Médard. **The Importance of Being Opportunistic: Practical Network Coding for Wireless Environments**. In: *Proceedings of the Annual Allerton Conference on Communication, Control, and Computing*. Vol. 43. 2005 (cit. on pp. 10, 19).

[Kat+06] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. **XORs in the Air: Practical Wireless Network Coding**. In: *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. 2006, pp. 243–254 (cit. on pp. 10, 50, 100, 101, 104).

[Kat+08] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. **XORs in the Air: Practical Wireless Network Coding**. In: *IEEE/ACM Transactions on Networking* 16.3 (2008), pp. 497–510 (cit. on pp. 10, 100, 101, 104).

[KCK06] J.-G. Ko, Y.-H. Cho, and H. Kim. **Performance Evaluation of IEEE 802.15. 4 MAC with Different Backoff Ranges in Wireless Sensor Networks**. In: *10th IEEE Singapore International Conference on Communication Systems*. IEEE. 2006, pp. 1–5 (cit. on p. 146).

[KDF08] L. Keller, E. Drinea, and C. Fragouli. **Online Broadcasting with Network Coding**. In: *4th Workshop on Network Coding, Theory and Applications*. IEEE. 2008, pp. 1–6 (cit. on pp. 95, 104).

[KGK07] S. Katti, S. Gollakota, and D. Katabi. **Embracing Wireless Interference: Analog Network Coding**. In: *ACM SIGCOMM Computer Communication Review* 37.4 (2007), pp. 397–408 (cit. on p. 12).

[KGM11] M. Khanafer, M. Guennoun, and H. T. Mouftah. **An Efficient Adaptive Backoff Algorithm for Wireless Sensor Networks**. In: *IEEE Global Telecommunications Conference (GLOBECOM)*. IEEE. 2011, pp. 1–6 (cit. on p. 146).

[KKR12] D. Katabi, S. Katti, and H. Rahul. **Harnessing Network Coding in Wireless Systems**. In: *Network Coding*. Elsevier, 2012, pp. 39–60. DOI: `10.1016/b978-0-12-380918-6.00002-0`. URL: `https://doi.org/10.1016%2Fb978-0-12-380918-6.00002-0` (cit. on pp. 9, 10, 12, 15).

[KKW12] A. Khreishah, I. M. Khalil, and J. Wu. **Distributed Network Coding-Based Opportunistic Routing for Multicast**. In: *Proceedings of the 13th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. 2012, pp. 115–124 (cit. on p. 27).

[KL05] N. Kimura and S. Latifi. **A Survey on Data Compression in Wireless Sensor Networks**. In: *International Conference on Information Technology: Coding and Computing (ITCC'05)*. Vol. 2. IEEE. 2005, pp. 8–13 (cit. on p. 54).

[KM02] R. Koetter and M. Médard. **Beyond Routing: An Algebraic Approach to Network Coding**. In: *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies*. Vol. 1. IEEE. 2002, pp. 122–130 (cit. on pp. 19, 28).

[KM03] R. Koetter and M. Médard. **An Algebraic Approach to Network Coding**. In: *IEEE/ACM Transactions on Networking (TON)* 11.5 (2003), pp. 782–795 (cit. on pp. 19, 28).

[KMS+19] N. Kushalnagar, G. Montenegro, C. Schumacher, et al. **IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals**. RFC 4919. RFC Editor, Aug. 2007 (accessed November 21, 2019). DOI: `10.17487/RFC4919`. URL: `https://www.rfc-editor.org/info/rfc4919` (cit. on p. 45).

[Köp+08] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. K. Haneveld, T. E. Parker, O. W. Visser, H. S. Lichte, and S. Valentin. **Simulating Wireless and Mobile Networks in OMNeT++ the MiXiM vision**. In: *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*. 2008, pp. 1–8 (cit. on p. 137).

[Kow13] H.-J. Kowalsky. **Lineare Algebra**. Walter de Gruyter, 2013 (cit. on p. 63).

[Ksc12] F. R. Kschischang. **An Introduction to Network Coding**. In: *Network Coding*. Academic Press, 2012, pp. 1–37. DOI: `10.1016/b978-0-12-380918-6.00001-9` (cit. on pp. 15, 63).

[KT75] L. Kleinrock and F. Tobagi. **Packet Switching in Radio Channels: Part I - Carrier Sense Multiple-Access Modes and their Throughput-Delay Characteristics**. In: *IEEE Transactions on Communications* 23.12 (1975), pp. 1400–1416 (cit. on p. 143).

[KTT09] O. Kosut, L. Tong, and D. Tse. **Nonlinear Network Coding is Necessary to Combat General Byzantine Attacks**. In: *Proceedings of the Annual Allerton Conference on Communication, Control, and Computing*. Vol. 47. IEEE. 2009, pp. 593–599 (cit. on p. 12).

[Kub+03] M. Kubisch, H. Karl, A. Wolisz, L. C. Zhong, and J. Rabaey. **Distributed Algorithms for Transmission Power Control in Wireless Sensor Networks**. In: *IEEE Wireless Communications and Networking (WCNC)*. Vol. 1. IEEE. 2003, pp. 558–563 (cit. on p. 137).

[KW07] H. Karl and A. Willig. **Protocols and Architectures for Wireless Wensor Networks**. John Wiley & Sons, 2007 (cit. on pp. 24, 25, 143).

[KWH11] D. Koutsonikolas, C.-C. Wang, and Y. C. Hu. **Efficient Network-Coding-Based Opportunistic Routing Through Cumulative coded Acknowledgments**. In: *IEEE/ACM Transactions on Networking* 19.5 (2011), 1368–1381 (cit. on pp. 23, 27).

[Kwo+12] K. H. Kwong, T.-T. Wu, H. G. Goh, K. Sasloglou, B. Stephen, I. Glover, C. Shen, W. Du, C. Michie, and I. Andonovic. **Practical Considerations for Wireless Sensor Networks in Cattle Monitoring Applications**. In: *Computers and Electronics in Agriculture* 81 (2012), pp. 33–44 (cit. on p. 1).

[Lee+09] K. Lee, S. Hong, S. J. Kim, I. Rhee, and S. Chong. **Slaw: A New Mobility Model for Human Walks**. In: *IEEE INFOCOM*. IEEE. 2009, pp. 855–863 (cit. on p. 153).

[Lee+14] S.-Y. Lee, Y.-S. Shin, K.-W. Lee, and J.-S. Ahn. **Performance Analysis of Extended Non-Overlapping Binary Exponential Backoff Algorithm over IEEE 802.15.4**. In: *Telecommunication Systems* 55.1 (2014), pp. 39–46 (cit. on p. 146).

[Lee05] J.-S. Lee. **An Experiment on Performance Study of IEEE 802.15.4 Wireless Networks**. In: *IEEE Conference on Emerging Technologies and Factory Automation*. Vol. 2. IEEE. 2005, 8–pp (cit. on p. 47).

[LH03] B. Liang and Z. J. Haas. **Predictive Distance-Based Mobility Management for Multidimensional PCS Networks**. In: *IEEE/ACM Transactions On Networking* 11.5 (2003), pp. 718–732 (cit. on p. 151).

[LH99] B. Liang and Z. J. Haas. **Predictive Distance-Based Mobility Management for PCS Networks**. In: *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. 18th Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)*. Vol. 3. IEEE. 1999, pp. 1377–1384 (cit. on p. 151).

[Li+07] L. Li, R. Ramjee, M. Buddhikot, and S. Miller. **Network Coding-Based Broadcast in Mobile Ad-Hoc Networks**. In: *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*. IEEE. 2007, pp. 1739–1747 (cit. on pp. 50, 101, 104).

[Li+11] C. Li, H. Zhang, B. Hao, and J. Li. **A Survey on Routing Protocols for Large-Scale Wireless Sensor Networks**. In: *Sensors* 11.4 (Mar. 2011), pp. 3498–3526. DOI: 10.3390/s110403498 (cit. on p. 27).

[Lia02] W. Liang. **Constructing Minimum-Energy Broadcast Trees in Wireless Ad Hoc Networks**. In: *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing*. 2002, pp. 112–122 (cit. on p. 29).

[Lin+16] S. Lin, F. Miao, J. Zhang, G. Zhou, L. Gu, T. He, J. A. Stankovic, S. Son, and G. J. Pappas. **ATPC: Adaptive Transmission Power Control for Wireless Sensor Networks**. In: *ACM Transactions on Sensor Networks (TOSN)* 12.1 (2016), p. 6 (cit. on p. 137).

[LKW09] X.-H. Lin, Y.-K. Kwok, and H. Wang. **Energy-Efficient Resource Management Techniques in Wireless Sensor Networks**. In: *Guide to Wireless Sensor Networks*. Springer London, 2009, pp. 439–468. DOI: 10.1007/978-1-84882-218-4_17 (cit. on p. 25).

[LL04a] A. R. Lehman and E. Lehman. **Complexity Classification of Network Information Flow Problems**. In: *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics. 2004, pp. 142–150 (cit. on pp. 12, 30).

[LL04b] Z. Li and B. Li. **Network Coding: The Case of Multiple Unicast Sessions**. In: *Allerton Conference on Communications*. Vol. 16. 8. 2004 (cit. on pp. 19, 29).

[LLL07] Y. Lin, B. Liang, and B. Li. **Data Persistence in Large-Scale Sensor Networks with Decentralized Fountain Codes**. In: *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*. IEEE. 2007, pp. 1658–1666 (cit. on pp. 93, 94).

[LLL08] Y. Lin, B. Li, and B. Liang. **CodeOR: Opportunistic Routing in Wireless Mesh Networks with Segmented Network Coding**. In: *IEEE International Conference on Network Protocols*. IEEE. 2008, pp. 13–22 (cit. on pp. 23, 27).

[LMK05] D. S. Lun, M. Médard, and R. Koetter. **Efficient Operation of Wireless Packet Networks Using Network Coding**. Vol. 5. International Workshop on Convergent Technologies (IWCT), 2005 (cit. on p. 29).

[LMS09] D. E. Lucani, M. Médard, and M. Stojanovic. **Random Linear Network Coding for Time-Division Duplexing: Field Size Considerations**. In: *Proceedings of the Global Communications Conference (GLOBECOM)*. IEEE, Dec. 2009, pp. 1–6. DOI: `10.1109/GLOCOM.2009.5425257`. URL: `https://doi.org/10.1109/GLOCOM.2009.5425257` (cit. on p. 68).

[LoR18] LoRa-Alliance. **LoRaWAN 1.0.3 Specification**. 2018 (cit. on p. 49).

[Lu+05] B. Lu, L. Wu, T. G. Habetler, R. G. Harley, and J. A. Gutierrez. **On the Application of Wireless Sensor Networks in Condition Monitoring and Energy Usage Evaluation for Electric Machines**. In: *31st Annual Conference of IEEE Industrial Electronics Society (IECON)*. IEEE. 2005, 6–pp (cit. on p. 1).

[Lub+01] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman. **Efficient Erasure Correcting Codes**. In: *IEEE Transactions on Information Theory* 47.2 (2001), pp. 569–584 (cit. on p. 22).

[Lub02] M. Luby. **LT Codes**. In: *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2002, pp. 271–280 (cit. on pp. 22, 73).

[Lun+04] D. S. Lun, M. Médard, R. Koetter, and M. Effros. **On Coding for Reliable Communication over Packet Networks**. In: *Proceedings of the Annual Allerton Conference on Communication, Control, and Computing*. Vol. 42. Oct. 2004 (cit. on pp. 19, 23).

[Lun+05a] D. S. Lun, M. Médard, R. Koetter, and M. Effros. **Further Results on Coding for Reliable Communication over Packet Networks**. In: *Proceedings. International Symposium on Information Theory (ISIT)*. IEEE. 2005, pp. 1848–1852 (cit. on p. 19).

[Lun+05b] D. S. Lun, N. Ratnakar, R. Koetter, M. Médard, E. Ahmed, and H. Lee. **Achieving Minimum-Cost Multicast: A Decentralized Approach Based on Network Coding**. In: *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies*. Vol. 3. IEEE. 2005, pp. 1607–1617 (cit. on p. 29).

[LW03] A. Leon-Garcia and I. Widjaja. **Communication Networks**. McGraw-Hill, Inc., 2003 (cit. on p. 21).

[LYC03] S.-Y. Li, R. W. Yeung, and N. Cai. **Linear Network Coding**. In: *IEEE Transactions on Information Theory* 49.2 (2003), pp. 371–381 (cit. on pp. 11, 19, 28).

[Mac05] D. J. MacKay. **Fountain Codes**. In: *Communications, IEE Proceedings*. Vol. 152. 6. IET. 2005, pp. 1062–1068 (cit. on p. 22).

[May02] P. Maymounkov. **Online Codes**. Tech. rep. Technical Report, New York University, 2002 (cit. on p. 22).

[MCN11] A. Munjal, T. Camp, and W. C. Navidi. **SMOOTH: A Simple Way to Model Human Mobility**. In: *Proceedings of the 14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. 2011, pp. 351–360 (cit. on p. 153).

[Méd+14] M. Médard, F. H. Fitzek, M.-J. Montpetit, and C. Rosenberg. **Network Coding Mythbusting: Why it is not about butterflies anymore**. In: *IEEE Communications Magazine* 52.7 (2014), pp. 177–183 (cit. on p. 69).

[MHZ17] F. Mager, C. Herrmann, and M. Zimmerling. **One for All, All for One: Toward Efficient Many-to-Many Broadcast in Dynamic Wireless Networks**. In: *Proceedings of the 4th ACM Workshop on Hot Topics in Wireless*. 2017, pp. 19–23 (cit. on pp. 99, 104).

[Mit04] M. Mitzenmacher. **Digital Fountains: A Survey and Look Forward**. In: *Information Theory Workshop*. IEEE. 2004, pp. 271–276 (cit. on p. 22).

[ML08] X. Ma and W. Luo. **The Analysis of 6LoWPAN Technology**. In: *IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*. Vol. 1. IEEE. 2008, pp. 963–966 (cit. on p. 45).

[MNT11] T. Matsuda, T. Noguchi, and T. Takine. **Survey of Network Coding and its Applications**. In: *IEICE Transactions on Communications* 94.3 (2011), pp. 698–717 (cit. on pp. 5, 8).

[MR95] R. Motwani and P. Raghavan. **Randomized Algorithms**. Cambridge university press, 1995 (cit. on pp. 59, 60, 86).

[MS06] D. Mosk-Aoyama and D. Shah. **Information Dissemination via Network Coding**. In: *IEEE International Symposium on Information Theory*. IEEE. 2006, pp. 1748–1752 (cit. on pp. 97, 104).

[MS12] M. Médard and A. Sprintson. **Network Coding: Fundamentals and Applications**. Academic Press, 2012 (cit. on pp. 5, 73).

[MS77] F. J. MacWilliams and N. J. A. Sloane. **The Theory of Error-Correcting Codes**. Vol. 16. Elsevier, 1977 (cit. on p. 22).

[Mun+07] D. Munaretto, J. Widmer, M. Rossi, and M. Zorzi. **Network Coding Strategies for Data Persistence in Static and Mobile Sensor Networks**. In: *5th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks and Workshops (WiOpt)*. IEEE. 2007, pp. 1–8 (cit. on pp. 102–104, 109, 122).

[Mun+08] D. Munaretto, J. Widmer, M. Rossi, and M. Zorzi. **Resilient Coding Algorithms for Sensor Network Data Persistence**. In: *Wireless Sensor Networks*. Springer, 2008, pp. 156–170 (cit. on pp. 102–104, 109, 122).

[MWM09] S. Misra, I. Woungang, and S. C. Misra. **Guide to Wireless Sensor Networks**. Springer London, 2009. DOI: 10.1007/978-1-84882-218-4 (cit. on p. 24).

[Nar+02] S. Narayanaswamy, V. Kawadia, R. S. Sreenivas, and P. Kumar. **Power Control in Ad-Hoc Networks: Theory, Architecture, Algorithm and Implementation of the COMPOW Protocol**. In: *European Wireless Conference*. Vol. 2002. Florence, Italy. 2002, p. 156162 (cit. on p. 137).

[Ngu+08] D. Nguyen, T. Tran, T. Nguyen, and B. Bose. **Wireless Broadcast Using Network Coding**. In: *IEEE Transactions on Vehicular Technology* 58.2 (2008), pp. 914–925 (cit. on p. 23).

[NZN16] R. Natarajan, P. Zand, and M. Nabi. **Analysis of Coexistence between IEEE 802.15.4, BLE and IEEE 802.11 in the 2.4 GHz ISM band**. In: *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*. IEEE. 2016, pp. 6025–6032 (cit. on p. 51).

[Ome18] J. Omeliyanenko. **Untersuchung der Energieeffizienz der MCGC-Netzwerkkodierung im IoT-Testbed**. Unpublished master thesis. Sept. 2018 (cit. on pp. 244–247).

[OS12] M. F. Othman and K. Shazali. **Wireless Sensor Network Applications: A Study in Environment Monitoring System**. In: *Procedia Engineering* 41 (2012), pp. 1204–1210 (cit. on p. 1).

[Öst+07] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. **Cross-Level Simulation in COOJA**. In: *Proceedings of the 4th European Conference on Wireless Sensor Networks (EWSN)*. 2007 (cit. on p. 137).

[Pat+12] S. Patel, H. Park, P. Bonato, L. Chan, and M. Rodgers. **A Review of Wearable Sensors and Systems with Application in Rehabilitation**. In: *Journal of Neuroengineering and Rehabilitation* 9.1 (2012), p. 1 (cit. on p. 1).

[Pet+06] M. Petrova, J. Riihijarvi, P. Mahonen, and S. Labella. **Performance Study of IEEE 802.15.4 using Measurements and Simulations**. In: *IEEE Wireless Communications and Networking Conference (WCNC)*. Vol. 1. IEEE. 2006, pp. 487–492 (cit. on p. 160).

[Pet+07] M. Petrova, L. Wu, P. Mahonen, and J. Riihijarvi. **Interference Measurements on Performance Degradation between Colocated IEEE 802.11g/n and IEEE 802.15.4 Networks**. In: *6th International Conference on Networking (ICN'07)*. IEEE. 2007, pp. 93–93 (cit. on p. 51).

[PNV13] N. A. Pantazis, S. A. Nikolidakis, and D. D. Vergados. **Energy-efficient Routing Protocols in Wireless Sensor Networks: A Survey**. In: *IEEE Communications Surveys & Tutorials* 15.2 (2013), pp. 551–591 (cit. on pp. 24, 25, 27).

[Pro21] ns-3 Projekt. **ns-3 | a discrete-event network simulator for internet systems**. 2021 (accessed June 15, 2021). URL: `https://www.nsnam.org/` (cit. on p. 137).

[PS72] G. Pólya and G. Szegő. **Problems and Theorems in Analysis**. Springer, 1972 (cit. on p. 60).

[Rag+02] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava. **Energy-Aware Wireless Microsensor Networks**. In: *IEEE Signal Processing Magazine* 19.2 (2002), pp. 40–50 (cit. on p. 54).

[Rap02] T. S. Rappaport. **Wireless Communications: Principles and Practice**. 2nd ed. Prentice-Hall New Jersey, 2002 (cit. on p. 141).

[Rap96] T. S. Rappaport. **Wireless Communications: Principles and Practice**. Prentice-Hall New Jersey, 1996 (cit. on pp. 138–141, 160).

[RBC14] T. Rault, A. Bouabdallah, and Y. Challal. **Energy Efficiency in Wireless Sensor Networks: A Top-Down Survey**. In: *Computer Networks* 67 (2014), pp. 104–122 (cit. on p. 55).

[RBD13] M. A. Razzaque, C. Bleakley, and S. Dobson. **Compression in Wireless Sensor Networks: A Survey and Comparative Evaluation**. In: *ACM Transactions on Sensor Networks (TOSN)* 10.1 (2013), pp. 1–44 (cit. on p. 54).

[RG09] D. Rohm and M. Goyal. **Dynamic Backoff for IEEE 802.15. 4 Beaconless Networks**. In: *IEEE Mini-Grants (National Science Foundation under Grant No. 0442313), University of Wisconsin Milwaukee, Milwaukee, WI* 53201 (2009) (cit. on p. 146).

[Riz97] L. Rizzo. **Effective Erasure Codes for Reliable Computer Communication Protocols**. In: *ACM SIGCOMM Computer Communication Review* 27.2 (1997), pp. 24–36 (cit. on p. 22).

[RK17] I. M. Runge and R. Kolla. **MCGC: A Network Coding Approach for Reliable Large-Scale Wireless Networks**. In: *Proceedings of the First ACM International Workshop on the Engineering of Reliable, Robust, and Secure Embedded Wireless Sensing Systems*. Nov. 2017, pp. 16–23 (cit. on p. 106).

[RK18] I. M. Runge and R. Kolla. **Extensions at Broadcast Growth Codes**. In: *17. GI/ITG KuVS Fachgespräch Sensornetze* (Sept. 2018), p. 48 (cit. on p. 106).

[RKS17] U. Raza, P. Kulkarni, and M. Sooriyabandara. **Low Power Wide Area Networks: An Overview**. In: *IEEE Communications Surveys & Tutorials* 19.2 (2017), pp. 855–873 (cit. on p. 49).

[RL09] W. Ryan and S. Lin. **Channel Codes: Classical and Modern**. Cambridge University Press, 2009 (cit. on pp. 20, 22).

[RN19] A. G. Ramonet and T. Noguchi. **IEEE 802.15.4 Historical Evolution and Trends**. In: *21st International Conference on Advanced Communication Technology (ICACT)*. IEEE. 2019, pp. 351–359 (cit. on p. 46).

[ROG03] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves. **Energy-Efficient Collision-Free Medium Access Control for Wireless Sensor Networks**. In: *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*. 2003, pp. 181–192 (cit. on p. 144).

[RS11] A. Ribeiro and R. Sofia. **A Survey on Mobility Models for Wireless Networks**. In: (2011) (cit. on p. 149).

[Sar+03] T. K. Sarkar, Z. Ji, K. Kim, A. Medouri, and M. Salazar-Palma. **A Survey of Various Propagation Models for Mobile Communication**. In: *IEEE Antennas and Propagation Magazine* 45.3 (2003), pp. 51–82 (cit. on p. 141).

[SB16] E. Silveira and S. Bonho. **Temperature Monitoring through Wireless Sensor Network Using an 802.15.4/802.11 Gateway**. In: *IFAC-PapersOnLine* 49.30 (2016), pp. 120–125 (cit. on p. 47).

[SCA21] SCALABLE-Network-Technologies-Inc. **QualNet Network Simulation Software**. 2021 (accessed June 15, 2021). URL: `https://www.scalable-networks.com/products/qualnet-network-simulation-software/` (cit. on p. 137).

[SET03] P. Sanders, S. Egner, and L. Tolhuizen. **Polynomial Time Algorithms for Network Information Flow**. In: *Proceedings of the 15th Annual ACM Symposium on Parallel Algorithms and Architectures*. 2003, pp. 286–294 (cit. on p. 19).

[Sha+09] D. Shah et al. **Gossip algorithms**. In: *Foundations and Trends® in Networking* 3.1 (2009), pp. 1–125 (cit. on p. 25).

[Shi01] B. L. Shinder. **Computer Networking Essentials**. Pearson Education, 2001 (cit. on p. 45).

[Sho06] A. Shokrollahi. **Raptor Codes**. In: *IEEE Transactions on Information Theory* 52.6 (2006), pp. 2551–2567 (cit. on p. 22).

[Sig19] Sigfox. **Sigfox Connected Objects: Radio Specifications**. Nov. 2019 (cit. on p. 49).

[Sin14] B. Singh. **Data Communications and Computer Networks**. PHI Learning Pvt. Ltd., 2014 (cit. on p. 51).

[Skl+01] B. Sklar et al. **Digital Communications: Fundamentals and Applications**. 2001 (cit. on p. 20).

[SM09] H. Sack and C. Meinel. **Digitale Kommunikation**. Springer Berlin Heidelberg, 2009 (cit. on p. 51).

[SMR11] H. Seferoglu, A. Markopoulou, and K. K. Ramakrishnan. **I2NC: Intra- and Inter-Session Network Coding for Unicast Flows in Wireless Networks**. In: *Proceedings IEEE INFOCOM*. IEEE. 2011, pp. 1035–1043 (cit. on p. 11).

[SRB10] S. Sengupta, S. Rayanchu, and S. Banerjee. **Network Coding-Aware Routing in Wireless Networks**. In: *IEEE/ACM Transactions on Networking* 18.4 (2010), pp. 1158–1170 (cit. on p. 27).

[Sri+12] T. Srisooksai, K. Keamarungsi, P. Lamsrichan, and K. Araki. **Practical Data Compression in Wireless Sensor Networks: A Survey**. In: *Journal of Network and Computer Applications* 35.1 (2012), pp. 37–59 (cit. on p. 54).

[SSS10] S. K. Singh, M. Singh, and D. Singh. **Routing Protocols in Wireless Sensor Networks - A Survey**. In: *International Journal of Computer Science & Engineering Survey* 1.2 (Nov. 2010), pp. 63–83. DOI: `10.5121/ijcses.2010.1206` (cit. on pp. 25, 27).

[STK09] P. Sadeghi, D. Traskov, and R. Koetter. **Adaptive Network Coding for Broadcast Channels**. In: *Workshop on Network Coding, Theory, and Applications*. IEEE. 2009, pp. 80–85 (cit. on pp. 95, 104).

[Stö17] M. Stölzle. **Reaktive Anpassung der MCGC Netzwerkkodierung an die vorliegende Dynamik eines WSNs**. Unpublished master thesis. Sept. 2017 (cit. on pp. 149, 152–154, 225, 234).

[TB97] L. N. Trefethen and D. Bau III. **Numerical Linear Algebra**. Vol. 50. Siam, 1997 (cit. on p. 65).

[TBF11] O. Trullols-Cruces, J. M. Barcelo-Ordinas, and M. Fiore. **Exact Decoding Probability under Random Linear Network Coding**. In: *IEEE Communications Letters* 15.1 (2011), pp. 67–69 (cit. on pp. 66, 67, 69).

[TET21] TETCOS-LLP. **NetSim Standard**. 2021 (accessed June 15, 2021). URL: `https://www.tetcos.com/netsim-std.html` (cit. on p. 137).

[Tho08] R. Thobaben. **Joint Network/Channel Coding for Multi-User Hybrid-ARQ**. In: *7th International ITG Conference on Source and Channel Coding*. VDE. 2008, pp. 1–6 (cit. on p. 23).

[Tia+02] J. Tian, J. Hahner, C. Becker, I. Stepanov, and K. Rothermel. **Graph-Based Mobility Model for Mobile Ad Hoc Network Simulation**. In: *Proceedings of the 35th Annual Simulation Symposium*. IEEE. 2002, pp. 337–344 (cit. on p. 152).

[Tol99] V. Tolety. **Load Reduction in Ad Hoc Networks Using Mobile Servers**. In: (1999) (cit. on p. 151).

[TW11] A. S. Tanenbaum and D. J. Wetherall. **Computer Networks**. 5th ed. Prentice Hall, 2011 (cit. on pp. 21, 22, 25).

[VAA04] M. C. Vuran, Ö. B. Akan, and I. F. Akyildiz. **Spatio-Temporal Correlation: Theory and Applications for Wireless Sensor Networks**. In: *Computer Networks* 45.3 (2004), pp. 245–259 (cit. on p. 54).

[Vik21] A. Viklund. **MiXiM project**. 2011 (accessed June 15, 2021). URL: `http://mixim.sourceforge.net/` (cit. on p. 137).

[VL03] T. Van Dam and K. Langendoen. **An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks**. In: *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*. 2003, pp. 171–180 (cit. on p. 144).

[Wan+17] Z. Wang, M. Feng, T. Miao, W. Jiang, and J. Shen. **Energy Efficient MAC Protocol for Wireless Sensor Networks: A Survey**. In: *Cloud Computing and Security*. Ed. by X. Sun, H.-C. Chao, X. You, and E. Bertino. Cham: Springer International Publishing, 2017, pp. 422–429 (cit. on pp. 55, 157).

[WB10] Q. Wang and I. Balasingham. **Wireless Sensor Networks - An Introduction**. In: *Wireless Sensor Networks: Application-Centric Design* (2010), pp. 1–14 (cit. on p. 1).

[WCK+05] Y. Wu, P. A. Chou, S.-Y. Kung, et al. **Information Exchange in Wireless Networks with Network Coding and Physical-Layer Broadcast**. Tech. rep. MSR-TR-2004, 2005 (cit. on pp. 10, 15, 19, 29).

[WCK05] Y. Wu, P. A. Chou, and S.-Y. Kung. **Minimum-Energy Multicast in Mobile Ad Hoc Networks using Network Coding**. In: *IEEE Transactions on Communications* 53.11 (2005), pp. 1906–1918 (cit. on p. 29).

[WDC07] Y. Wu, S. M. Das, and R. Chandra. **Routing With a Markovian Metric to Promote Local Mixing**. In: *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*. IEEE. 2007, pp. 2381–2385 (cit. on p. 27).

[WFL05] J. Widmer, C. Fragouli, and J.-Y. Le Boudec. **Low-Complexity Energy-Efficient Broadcasting in Wireless Ad-Hoc Networks Using Network Coding**. In: *In Proceedings*. LCA-CONF-2005-016. 2005 (cit. on pp. 29, 54).

[Wis18] A. Wischnewski. **Selbstdrosselung in Funknetzen**. Unpublished bachelor thesis. Aug. 2018 (cit. on p. 146).

[WJ16] Q. Wang and J. Jiang. **Comparative Examination on Architecture and Protocol of Industrial Wireless Sensor Network Standards**. In: *IEEE Communications Surveys & Tutorials* 18.3 (2016), pp. 2197–2219 (cit. on p. 49).

[WL05] J. Widmer and J.-Y. Le Boudec. **Network Coding for Efficient Communication in Extreme Networks**. In: *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-Tolerant Networking*. 2005, pp. 284–291 (cit. on pp. 25, 29, 30, 98, 99, 104).

[WMB07] B. H. Walke, S. Mangold, and L. Berlemann. **IEEE 802 Wireless Systems: Protocols, Multi-Hop Mesh/Relaying, Performance and Spectrum Coexistence**. John Wiley & Sons, 2007 (cit. on pp. 45, 141, 142).

[WNE00] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides. **On the Construction of Energy-Efficient Broadcast and Multicast Trees in Wireless Networks**. In: *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. 19th Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064)*. Vol. 2. IEEE. 2000, pp. 585–594 (cit. on p. 29).

[Wu+05] Y. Wu, V. Stankovic, Z. Xiong, and S.-Y. Kung. **On Practical Design for Joint Distributed Source and Network Coding**. In: *1st Workshop on Network Coding, Theory, and Applications*. Citeseer. 2005 (cit. on p. 23).

[Xia+16] X. Xiao, Q. He, Z. Fu, M. Xu, and X. Zhang. **Applying CS and WSN Methods for Improving Efficiency of Frozen and Chilled Aquatic Products Monitoring System in Cold Chain Logistics**. In: *Food Control* 60 (2016), pp. 656–666 (cit. on p. 1).

[Yan+14] S. Yang, R. W. Yeung, J. H. Cheung, and H. H. Yin. **BATS: Network Coding in Action**. In: *Proceedings of the Annual Allerton Conference on Communication, Control, and Computing*. Vol. 52. IEEE. 2014, pp. 1204–1211 (cit. on p. 23).

[YB09] B. Yahya and J. Ben-Othman. **Towards a Classification of Energy Aware MAC Protocols for Wireless Sensor Networks**. In: *Wireless Communications and Mobile Computing* 9.12 (2009), pp. 1572–1607 (cit. on p. 143).

[Yeu08] R. W. Yeung. **Information Theory and Network Coding**. Springer Science & Business Media, 2008 (cit. on pp. 5, 8, 15, 18, 24, 29).

[Yeu11] R. W. Yeung. **Network Coding: A Historical Perspective**. In: *Proceedings of the IEEE* 99.3 (2011), pp. 366–371 (cit. on p. 18).

[YLN03] J. Yoon, M. Liu, and B. Noble. **Random Waypoint Considered Harmful**. In: *IEEE INFOCOM 2003. 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*. Vol. 2. IEEE. 2003, pp. 1312–1321 (cit. on p. 150).

[You91] R. M. Young. **75.9 Euler's Constant**. In: *The Mathematical Gazette* 75.472 (1991), pp. 187–190 (cit. on p. 60).

[YYZ07] X. Yan, R. W. Yeung, and Z. Zhang. **The Capacity Region for Multi-Source Multi-Sink Network Coding**. In: *IEEE International Symposium on Information Theory*. IEEE. 2007, pp. 116–120 (cit. on p. 30).

[Z-W20] Z-Wave-Alliance. **Z-Wave Specification 2020C**. 2020 (cit. on p. 49).

[ZBG98] X. Zeng, R. Bagrodia, and M. Gerla. **GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks**. In: *Proceedings of the 12th Workshop on Parallel and Distributed Simulation PADS'98 (Cat. No. 98TB100233)*. IEEE. 1998, pp. 154–161 (cit. on p. 137).

[Zen+08] K. Zen, D. Habibi, A. Rassau, and I. Ahmad. **Performance Evaluation of IEEE 802.15.4 for Mobile Sensor Networks**. In: *5th IFIP International Conference on Wireless and Optical Communications Networks (WOCN'08)*. IEEE. 2008, pp. 1–5 (cit. on p. 47).

[Zig15] ZigBee-Alliance. **ZigBee Specification**. 2015 (cit. on pp. 45, 49).

[Zig19] ZigBee-Alliance. **Low-power, low-cost, low-complexity networking for the Internet of Things**. 2019 (accessed November 21, 2019). URL: `https://zigbee.org/zigbee-for-developers/network-specifications/` (cit. on p. 45).

[ZK07] M. Z. Zamalloa and B. Krishnamachari. **An Analysis of Unreliability and Asymmetry in Low-Power Wireless Links**. In: *ACM Transactions on Sensor Networks (TOSN)* 3.2 (2007) (cit. on p. 141).

[ZL09] X. Zhang and B. Li. **Optimized Multipath Network Coding in Lossy Wireless Networks**. In: *IEEE Journal on Selected Areas in Communications* 27.5 (2009), pp. 622–634 (cit. on p. 27).

[ZLL06] S. Zhang, S. C. Liew, and P. P. Lam. **Hot Topic: Physical-Layer Network Coding**. In: *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*. 2006, pp. 358–365 (cit. on p. 12).

[ZNK12] X. Zhang, G. Neglia, and J. Kurose. **Network Coding in Disruption Tolerant Networks**. In: *Network Coding*. Elsevier, 2012, pp. 267–308. DOI: `10.1016/b978-0-12-380918-6.00010-x`. URL: `https://doi.org/10.1016%2Fb978-0-12-380918-6.00010-x` (cit. on pp. 25, 54).

[ZS17] S. Zhuo and Y.-Q. Song. **GoMacH: A Traffic Adaptive Multi-channel MAC Protocol for IoT**. In: *Proceedings of the 42nd Conference on Local Computer Networks (LCN)*. IEEE. 2017, pp. 489–497 (cit. on p. 144).