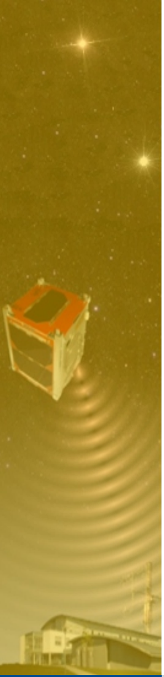
Institut für Informatik
Lehrstuhl für Robotik und Telematik
Prof. Dr. K. Schilling



Würzburger Forschungsberichte
in Robotik und Telematik

Uni Wuerzburg Research Notes
in Robotics and Telematics

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Band 4

Florian Zeiger

# Internet Protocol based networking of mobile robots

## Die Schriftenreihe

wird vom Lehrstuhl für Informatik VII: Robotik und Telematik der Universität Würzburg herausgegeben und präsentiert innovative Forschung aus den Bereichen der Robotik und der Telematik.

Die Kombination fortgeschrittener Informationsverarbeitungsmethoden mit Verfahren der Regelungstechnik eröffnet hier interessante Forschungs- und Anwendungsperspektiven. Es werden dabei folgende interdisziplinäre Aufgabenschwerpunkte bearbeitet:

- Robotik und Mechatronik: Kombination von Informatik, Elektronik, Mechanik, Sensorik, Regelungs- und Steuerungstechnik, um Roboter adaptiv und flexibel ihrer Arbeitsumgebung anzupassen.
- Telematik: Integration von Telekommunikation, Informatik und Steuerungstechnik, um Dienstleistungen an entfernten Standorten zu erbringen.

Anwendungsschwerpunkte sind u.a. mobile Roboter, Tele-Robotik, Raumfahrtsysteme und Medizin-Robotik.

Lehrstuhl Informatik VII
Robotik und Telematik
Am Hubland
D-97074 Wuerzburg

Tel.: +49 (0) 931 - 31 - 86678
Fax: +49 (0) 931 - 31 - 86679

schi@informatik.uni-wuerzburg.de
http://www7.informatik.uni-wuerzburg.de

ISSN (Internet): 1868-7474
ISSN (Print): 1868-7466
eISBN: 978-3-923959-59-4

### Zitation dieser Publikation

ZEIGER, F. (2010). Internet Protocol based networking of mobile robots. Schriftenreihe Würzburger Forschungsberichte in Robotik und Telematik, Band 4. Würzburg: Universität Würzburg.

# Internet Protocol based Networking of Mobile Robots

Dissertation zur Erlangung des
naturwissenschaftlichen Doktorgrades
der Bayerischen Julius–Maximilians–Universität Würzburg

vorgelegt von

**Florian Zeiger**

aus

Würzburg

Würzburg 2011

Eingereicht am: 07.04.2010
bei der Fakultät für Mathematik und Informatik
1. Gutachter: Prof. Dr. Klaus. Schilling
2. Gutachter: Prof. Seong-Lyun Kim, PhD
Tag der mündlichen Prüfung: 01.02.2011

# Danksagung

Die vorliegende Arbeit ist das Ergebnis aus 6 Jahren Forschung im Bereich der Fernsteuerung und Vernetzung von mobilen Robotern. Natürlich wäre die Arbeit in dieser Form nicht ohne die Unterstützung, die ich von Familie, Freunden und Kollegen erfahren habe möglich gewesen.

Zuerst bedanke ich mich bei meiner Frau Melanie, die mich immer unterstützt und mir den Rücken frei hält. Sie musste während der Erstellung der Arbeit viele einsame Abende verbringen.

Auch meinem Doktorvater Professor Dr. Klaus Schilling danke ich für die Möglichkeit an seinem Lehrstuhl eine Promotion durchzuführen. Zum einen konnte ich als langjähriger Mitarbeiter des Lehrstuhls an vielen spannenden Projekten, wie z.B. UWE-1 und EU-India, mitarbeiten, zum anderen konnte ich mich aktiv in der Aufbauphase des "Zentrum für Telematik e.V." einbringen. So war es mir möglich durch meine Arbeit viel Erfahrung in nationalen und internationalen Forschungsprojekten zu sammeln. Meinem Zweitgutachter Professor Seong-Lyun Kim danke ich, dass er sich die Zeit genommen hat und sich mit meiner Arbeit beschäftigt hat, um ein Gutachten zu erstellen. Meinen Prüfern Professor Dr. Rainer Kolla und Professor Dr.-Ing. Phuoc Tran-Gia danke ich dafür, dass sie gerne als Prüfer zu meiner Disputation bereit standen und eine problemlose und rasche Terminfindung unterstützten.

Meinem guten Freund Markus Sauer danke ich ebenfalls. Während unserer langen Zeit als Kollegen hat sich eine tiefe Freundschaft entwickelt und ich danke ihm für die vielen gemeinsamen Abende, Urlaube, Diskussionen und Ratschläge privater als auch fachlicher Art. Auch haben wir viele gemeinsame Forschungsarbeiten durchgeführt und konnten gemeinsam an der Gründung des "Zentrum für Telematik e.V." mitwirken.

Während meines Studiums konnte ich bei Praktika, Diplomarbeit und meiner Arbeit als studentische Hilfskraft sehr viel von Dr. Dirk Staehle, Prof. Dr. Michael Menth und Dr. Kenji Leibnitz lernen und danke ihnen dafür, dass sie mich dem Arbeiten in Forschung und Lehre näher brachten.

Dr. Rastin Pries danke ich zum einen für die fachlichen Diskussionen und Rückmeldungen, zum anderen auch als einen sehr guten Freund mit dem ich schon mehrere tausend Ski-Kilometer zurückgelegt habe.

Bei meinen Praktikanten, Studienarbeitern und Diplomanden Bernhard Herbst, Marco Schmidt, Jens Bardutzky, Christian Herrmann, Daniel Eck, Christian Selbach, Nikolaus Krämer, Thomas Fentzahn und Benjamin Ruderisch möchte ich mich für ihren Einsatz und die Projektarbeit recht herzlich bedanken. Oft konnten unsere gemeinsamen Forschungsarbeiten auch auf anerkannten internationalen Konferenzen publiziert werden. Viele dieser ehemaligen Studenten sind mittlerweile zu Kollegen geworden und tragen zu einem sehr guten Teamwork am Lehrstuhl und am "Zentrum für Telematik e.V." bei.

Für die sehr kompetente fachliche Zusammenarbeit, das sehr gute Arbeitsklima im Kollegenkreis und die immer vorhandene Hilfsbereitschaft im Team möchte ich mich bei meinen Kollegen am "Zentrum für Telematik e.V." Maximilian Drentschew, Markus Krauß, Eberhard von Deuster, Ulrike Walter, bei den Kollegen am Lehrstuhl Marco Schmidt, Christian Herrmann, Daniel Eck, Robin Heß, Heidi Frankenberger, Edith Reuter, Anette Zillenbiller, und auch bei den ehemaligen Kollegen Lothar Stolz, Dr. Frauke Driewer, Dr. Peter Hokayem, Professor Dušan Stipanović, Dr. Lei Ma, Dr. Martin Hess und Rajesh Shankar bedanken. Rajesh hat mich sehr bei meiner Arbeit im EU-India Projekt unterstützt und ich danke ihm auch für die vielen Eindrücke und Erfahrungen, die ich in Indien gewinnen konnte.

Meinen Freunden Dr. Robert Henjes, Dr. Andreas Binzenhöfer, Dr. Tobias Hoßfeld und Dr. Simon Oechsner möchte ich für die zahlreichen Abende mit fachlichen und auch nicht so fachlichen Diskussionen und die vielen Freizeitaktivitäten die wir zusammen bestritten haben danken.

Natürlich trug auch mein privates Umfeld zum erfolgreichen Abschluss der Promotion bei und ich möchte mich bei meinen Eltern und meiner ganzen Familie für die jahrelange Unterstützung bedanken.

# Contents

# 1 Introduction

Nowadays, robots are common in many civil areas in order to relieve and support the work of humans, or to improve the security of human workers in dangerous environments. Thereby, semi-autonomous, as well as teleoperated robots are used. In the last years, challenging application scenarios have turned out to reach the performance limits of the robot or the human operator. Relief can be achieved by coordinating multiple robots in order to share the workload and to solve the task jointly. Other approaches aim on reducing the workload of human operators by new and innovative user interfaces. Nevertheless, all solutions require a reliable communication which supports different application and task specific traffic characteristics. The underlying communication establishes the base for all kinds of cooperation, coordination, or teleoperation between human entities and robots. Robots without the ability to communicate are not able to develop full service capabilities, and communication is also required independent from the level of autonomy. In addition, communication should be transparent for the users – developers, as well as operators – and should minimize installation, maintenance, and configuration efforts. The objective for all scenarios is the development of a suitable and reliable communication between human operators and robots in order to allow for exchange of stored sensor data and commands. Nowadays, Internet protocol (IP) based communication is supported by many technologies. Thus, it enables the integration of teams with humans and robots on this level. Nevertheless, the performance of these IP communication links differ a lot according to the transported traffic and the underlying technologies like wireless LAN, Universal Mobile Telecommunication System (UMTS), or Internet. Link analyses for each of these technologies with respect to the needs of networked robotic scenarios are essential in order to implement effective and reliable communication between humans and robots. Considering the original motivation and requirements leading to the development of technologies like wireless LAN, mobile communication via UMTS, or the Internet clearly shows the importance to investigate existing communication technologies while simultaneously taking into account the different requirements and constraints from the area of networked robotics.

This work investigates mobile robot teleoperation using three major technologies (wireless LAN, UMTS, and Internet). For this application, the link behavior is analyzed and results are used to provide tailored solutions for integrating IP based communication on top of wireless LAN, UMTS, and Internet into networked robotic applications. Demonstrators with real hardware are presented for each of the investigated areas.

## 1.1 Contribution

The contribution of this thesis is focused on teleoperation of mobile robots via IP networks. This monograph addresses the issue in three areas: enabling mobile robot teleoperation by the example of the remote laboratory with real hardware experiments at the University of Würzburg, tuning of ad-hoc routing protocols to enable mobile robot teleoperation via mobile ad-hoc networks, and enabling mobile robot teleoperation via 3G telecommunication networks.

First, a requirement analysis and an evaluation of existing technologies are given, and together with a link analysis with respect to mobile robot teleoperation via the Internet, a design for a remote laboratory with learning units for mobile robots is developed. The proposed architecture is capable of hosting several hardware experiments and provides mechanisms for student access, tutor interfaces, user management, system maintenance, security issues, and reliable connection to robotics hardware over low bandwidth links. The proposed architecture is realized in the context of the tele-laboratory of the *Department of Robotics and Telematics* of the University of Würzburg. In addition, an extension of this tele-laboratory architecture is developed in order to provide an integrated and world wide accessible tele-laboratory in robotics, whereas hardware units can be placed at distant locations all over the world. The developed system supports resuming of sessions and an adaptive use of the bandwidth which is a significant advantage compared to the existing state of the art developments in this area. With respect to networked control, the given link analysis allows for the example setup of a Quanser hardware experiment to control the rotation, and respectively the position of a wheel via network.

Second, four different ad-hoc routing protocols are evaluated with respect to typical scenarios from networked robotics. The analysis of these ad-hoc routing protocols with default parameter settings in real hardware tests showed the potential to improve the protocol performance in the test scenarios. Relevant protocol parameters are identified and their effects on the routing protocol behavior are evaluated. The results are used to enable mobile robot teleoperation in mobile ad-hoc networks with multi-hop communication between remote operator and networked mobile robot hardware. Also additional mechanisms for improving the performance of the complete telerobotic system are developed, implemented, and evaluated. The findings are demonstrated with several selected application examples. By the example of an unmanned small size helicopter, local autonomy features are developed and allow for an integration of the unmanned aerial vehicle into a wireless IP based ad-hoc network. These local autonomy functions keep the robotic system always in a safe operation mode and thus, the telerobotic system can cope also with suddenly occurring communication failures. A further application example presents a navigation and exploration task, where the mobile robot transmits an onboard video stream while being teleoperated via a wireless ad-hoc network. To enable a smooth video transmission, a traffic shaping mechanism is developed which adaptively adjusts the video quality based on network feedback. This mechanism is of particular importance when different types of traffic are present in the network and when video transmission is realized via multiple communication relays.

The third area focuses on communication of robotic systems for teleoperation via 3G telecommunication networks (UMTS). A detailed analysis of the communication link considers packet inter-arrival times, round trip times, one-way delays, and packet loss. Two mobile robot systems which differ in their communication architecture are used in demonstration setups to show the performance of these different implementations while using a 3G telecommunication link for the command and data link. Based on the measured characteristics, guidelines for implementing and parameterizing teleoperation architectures for mobile robots in such environments are given.

These three different subject areas are selected as they cover the currently most important technologies which will be used for mobile robot teleoperation in the near future.

## 1.2 Outline

This monograph is structured as follows. The three principal topics are represented by one chapter each, and the structures of these chapters are similar. Starting with the enabling technologies and an overview of the state-of-the-art, the scientific contributions follow, and the chapters conclude with the presentation of representative application examples of the specific research area.

Chapter 2 gives a design for an international remote laboratory with mobile robot hardware experiments. The content of this tele-laboratory is world wide accessible via the Internet. Enabling technologies like Internet, Ethernet, and IP based communication are presented, followed by an investigation of currently existing remote experiments and the used technologies like frameworks for dynamic web content generation, virtual reality methods, and rich Internet applications. A link analysis is presented which allows for a tele-laboratory architecture providing experiments also for low bandwidth access. This is followed by a detailed description of the modularized architecture of the tele-laboratory which is developed in the frame of this thesis. Finally, it is shown, how a networked control experiment can be realized based on knowledge of the underlying communication link and Section 2.4 gives a discussion of the results.

Chapter 3 investigates the application of ad-hoc routing protocols in the area of networked robotics. After highlighting important aspects of the underlying technology wireless LAN IEEE 802.11 and the ad-hoc routing protocols which are analyzed, the description of the test scenario is given. The protocol analysis first uses default parameter settings in a typical robotics scenario and gives detailed insights on the round trip times, the packet loss, and the rerouting time for this type of setup. Also relevant parameters for performance improvements are identified and the gained knowledge is applied to more complex scenarios including multi-hop communication. Chapter 3 presents two application scenarios, the integration of an unmanned helicopter into an IP based ad-hoc network, and a navigation and exploration task with mobile robots which are connected via a wireless ad-hoc network. Using these application examples, the positive effects of the developed mechanisms on the teleoperation capabilities are demonstrated. Finally, a discussion of the results for setting up mobile wireless ad-hoc networks with mobile robots is given in Section 3.4.

Chapter 4 elaborates teleoperation of mobile robots via UMTS connections.

Starting with a brief presentation of important mechanisms of this technology, a possibility for measuring one-way delays in the following test setups is developed. Section 4.2 explains the test scenario setup and gives a link analysis evaluating packet inter-arrival times, round trip times, one-way delays, and packet loss for different traffic streams and in different setups which are typical in the area of networked robotics. Finally, an application of mobile robot teleoperation via UMTS is presented by the example of two different mobile robots with different communication architectures. Chapter 4 concludes with a discussion of the results, guidelines, and a delay estimation for setting up mobile robot teleoperation systems via UMTS.

The results of this monograph are summarized and future directions are discussed in Chapter 5.

# 2 Teleoperation of Robotic Systems via Internet

Remote laboratories are a key aspect in engineering education in order to practice theoretical knowledge acquired in lectures also with hardware equipment. Mobile robots provide a motivating basis for hardware experiments in kinematics, dynamics and control topics. To set up an internationally available tele-laboratory which provides experiments with real hardware, a broad spectrum of challenges related to user management, data security aspects, safe teleoperation of hardware, 24 hours continuous availability, hardware interfaces, constraints related to the available link quality, fault detection, and damage prevention have to be mastered. On one hand, interfaces for the locally available hardware system have to be designed to allow for seamless teleoperation via Internet from any location worldwide. On the other hand a control protocol between users and hardware is developed to provide asynchronous command and data flow. Therefore, intensive studies of the present communication link are necessary to get the relevant parameters for designing the communication protocols and the hardware system architecture appropriately.

## 2.1 Enabling Technologies

The following paragraphs give a short introduction into the enabling technologies for the implementation of modern remote learning units of mobile robots. This includes a summary of currently used networking technologies for Local Area Networks and the Internet, as well as some details on telerobotics and a short summary of the state-of the art in tele-laboratories. More details on the addressed topics are given in the referenced literature of each paragraph.

### 2.1.1 Local Area Networks

The development of Local Area Networks (LANs) started in the seventies by Xerox PARC[1]. The first approach of Metcalf and Boggs [26] was only slightly modified and in 1983, the IEEE 802.3 standard for Carrier Sense Multiple Access/Collision Detection (CSMA/CD) was developed. But IEEE 802.3 is not the one and only standard for LANs. Also Token Bus IEEE 802.4 and Token Ring IEEE 802.5 were well established standards for Local Area Networks.

#### Ethernet

The early development stages of Ethernet started with *Thick Ethernet* which was also known as *10Base5*. The data rate of *10Base5* is specified for 10 Mbit/s, and a segment size of 500 m with a maximum of 100 nodes is supported. *Thin Ethernet* (*10Base2*) followed with a segment size of 185 m and a maximum of 30 network nodes. Compared to *10Base5*, the *10Base2* Ethernet was cheaper and much easier to install due to more flexible cables. Later, *10Base-T* was developed which uses twisted pair cables and allows for a segment size of 100 m with a maximum of 1024 nodes. Here, all the nodes are connected via hubs. *10Base-F* uses fiber optics as physical media and has a segment size of 2000 m with a maximum of 1024 nodes. *10Base-T* and *10Base-F* supported data rates of 10 Mbit/s. Within these standards, also *Switched Ethernet* is located as it is an improvement of *Ethernet* with respect to the efficiency of the communication channel. Initially, the nodes of an *Ethernet* network were connected via hubs. Thus, all nodes were located inside the same collision domain, which reduced the possible data throughput drastically when the nuber of nodes is increased, as a hub always forwards data packets to all output ports. As a consequence, switches were introduced in order to reduce the size of the collision domain, as they send packets only to the relevant output port. Nowadays, hubs, as well as switches are well known options for connecting network nodes.

#### Fast-Ethernet

1992 started the development of the *Fast Ethernet* technology to allow for higher data rates. In 1995, *Fast Ethernet* IEEE 802.3u was published as an extension

[1] Xerox Palo Alto Research Center

for the IEEE 802.3 standard with the definitions for *100Base-T*, *100Base-T2*, *100Base-T4*, and *100Base-TX*. These definitions differ in the type of the used cables and provide data rates of 100 Mbit/s in half duplex, and in full duplex mode. Nowadays, *100Base-TX* is the standard implementation for 100 MBit/s Ethernet, and network nodes can be connected via hubs or switches. *Fast Ethernet* is also defined for fiber optics by *100Base-FX* which supports a segment size of 2000 m with 100 MBit/s full duplex.

### Gigabit-Ethernet

In 1995, the development of a faster Ethernet Standard was initiated and 1998 *Gigabit-Ethernet* was published within the IEEE 802.3z standard. *1000Base-T* supports a segment size of 100 m, and the nodes are connected via hubs or switches. Data transmission via fiber optics is supported by *1000Base-SX* with a segment size of 550 m, and by *1000Base-LX* with a segment size of 5000 m. Currently, new standards for data rates of 10 GBit/s are in the development phase. IEEE 802.3ae defines *10 Gigabit-Ethernet* for data transmission via fiber optics and IEEE 802.3an and IEEE 802.3ak are standards for *10 Gigabit-Ethernet* copper cables.

| Type | max. segment size | |
|---|---|---|
| *10GBase-LX4* | 240 m - 300 m | wavelength division multiplexing |
| *10GBase-LW4* | 240 m - 300 m<br>10 km | wavelength division multiplexing, multi-mode cabling (300 m), single-mode fiber (10 km) |
| *10GBase-SR* | 26 m, 82 m, 300 m | short range multi-mode fiber cabling |
| *10GBase-LR* | 10 km | single-mode fiber, wave length 1310 nm |
| *10GBase-ER* | 40 km | single-mode fiber, wave length 1550 nm |
| *10GBase-SW* | 26 m, 82 m, 300 m | compatible to SONET and STM-64 equipment |
| *10GBase-LW* | 10 km | compatible to SONET and STM-64 equipment |
| *10GBase-EW* | 40 km | compatible to SONET and STM-64 equipment |
| *10GBase-CX4* | 15 m | |
| *10GBase-T* | 50 m - 100 m | CAT6e cable (50 m) and CAT7 (100 m) |

Table 2.1: Different types of *10 Gigabit-Ethernet*.

Also for *10 Gigabit-Ehternet* definitions exist for supporting a different segment size and a different type of wiring (cf. Table 2.1). For fiber optics *10GBase-LX4*, *10GBase-LW4*, *10GBase-SR*, *10GBase-LR*, *10GBase-ER*, *10GBase-SW*, *10GBase-LW*, and *10GBase-EW* are defined, whereas *10GBase-CX4* and *10GBase-T* include descriptions for wiring via copper cables.

### 2.1.2 Internet

The Internet is in many ways exceptional. It is a system which was not planned by anyone and which has no centralized control instance. Also the enormous success and the fast growth is unique. In the meantime, several books and publications addressed the history of the Internet. [27] gives detailed historical information on the Internet. This section will give a brief summary of the history and the origin of the Internet and will further elaborate some technical aspects which are relevant for the implementation of remote laboratories. The content of the following paragraph is based on the information in [27] and [28].

The hour of birth of an idea which describes a network based of packet switching was on the 1967 ACM SIGOPS Symposium on Operating System Principles, where Roberts [29] proposed the idea of this system. The implementation of the system developed on the basis of the idea of Roberts was later known as the ARPANET. In December 1968, ARPA (Advanced Research Projects Agency)[2] commissioned BBN[3] for the development of a network which should allow for sending electronic messages to network nodes in distant locations. Thus, a first network of four nodes was established in December 1969 - the ARPANET - which connected the University of Utah, the Stanford Research Institute (SRI), the University of California, Los Angeles (UCLA), and the University of California, Santa Barbara (UCSB). ARPANET grew very fast, and after three years, in 1972, ARPANET consists of about 34 nodes. The increasing size of ARPANET also showed the necessity of a protocol for flow control in order to provide a suitable data flow in such a large (at this time, ARPANET was the largest existing computer network) network. This can also be considered as the birth of the Transmission Control Protocol / Internet Protocol (TCP/IP). In the eighties, more and more LANs were connected to the ARPANET and the Domain Naming System

[2]ARPA was founded in 1958; currently known as DARPA - Defense Advanced Research Projects Agency after renaming in 1996

[3]BBN - a consulting company from Cambridge, Massachusetts

(DNS) was introduced in order to provide a unique addressing of all the hosts. Already at the end of the seventies the positive effect of the ARPANET on research was noticed by the U.S. National Science Foundation (NSF) and the NSFNET-Backbone network was realized. Regional networks and LANs were connected via Network Access Points (NAP) to this NSFNET-Backbone network with the Transmission Control Protocol / Internet Protocol suite being the big commonality of all these connected networks. Thus, the successful design of NSFNET was often taken as archetype for research networks established in the nineties. In the early days of the Internet, the majority of the users were researchers or government organizations. The development of the World Wide Web was the initial start of all the nowadays known services which can be currently used by everyone as soon as an Internet connection is available, and which made the Internet known to everybody. Although the name *Internet* suggests one homogeneous network,

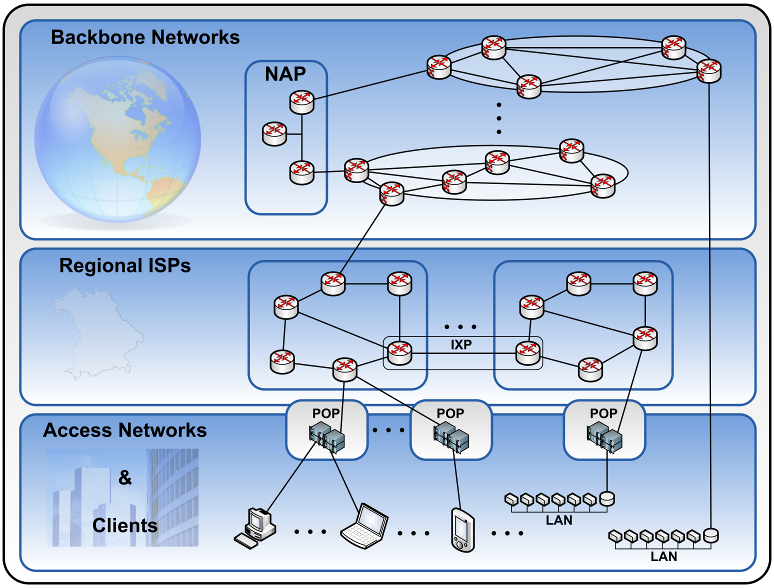


Figure 2.1: The Internet.

the architecture of the Internet is much more complex (cf. Figure 2.1) due to its origin described above. It consists of different interconnected networks, and also many different technologies for physical data transmission, as well as for logical data transmission (different protocols) are in use. The core of the complete architecture are backbone networks. These backbone networks have high capacities and they are connected via Network Access Points (NAP). Regional Internet service providers (ISP) are connected to backbone networks. Internet exchange points (IXP) connect different ISP networks in order to reduce the transit traffic to their upstream network providers. Regional ISPs are connected to access networks via a point of presence (POP). Usually, access networks use technologies like DSL or dial-in lines to establish connections to client computers of local are networks (e.g. company networks).

As already mentioned above, also different technologies are present inside all these networks. Since the early days of the Internet, technologies like X.25, frame relay, and *Asynchronous Transfer Mode* (ATM) are used. X.25 was one of the first protocols suites developed for data transmission between computers via the telephone network. X.25 data packets consisted of a three Byte header and up to 128 Byte payload data while supporting a maximum data rate of 19200 Bit/s. In the eighties, the frame relay technology replaced X.25. Frame Relay transmits data via virtual circuits and supports higher data rates than X.25. Data packets consist of a one Byte flag, two to four Byte address data, variable length of payload data, two Byte frame check sequence, and again a one Byte flag. A much better known data transmission technique is *Asynchronous Transfer Mode* which was developed in the beginning of the nineties. ATM also uses virtual circuits and it transmits data in frames with a fixed length of 53 Byte (5 Byte header and 48 Byte payload). Usually, data rates of 155 Mbit/s or 622 Mbit/s are supported. The installation of ATM was quite expensive. But nevertheless, it was a widely used technology whose relevance stared to decrease just a few years ago. In future, ATM will most probably be replaced by Ethernet based networks.

The availability of a world wide accessible Internet opened up new perspectives for distributed services, and the remote laboratories presented in this chapter are also one of these new services. For the implementation of Internet services of any kind, it is always necessary to know the capabilities and the limitations of the underlying network. Recently, many researchers worked on the characterization of Internet traffic with respect to different applications and services e.g. www and http traffic, voice over IP, file sharing, or video on-demand. The challenges in implementing remote laboratories which are accessible and usable via the Internet are described in Section 2.1.4.

### 2.1.3 Telerobotics

The development of telerobotics and telerobotic control schemes was initiated by the need for performing operations at remote sites, where human beings were not able to work (e.g. due to safety reasons). Also the reduction of costs and time saving due to the automation of processes promoted the efforts of developing capable telerobotic systems. In the beginning of the nineties, first telerobotic approaches were developed [30] [31], and in these scenarios, the robots are often limited in their autonomy. Usually, this drawback is compensated by a human operator, whereas the robot performs low level instructions to fulfill its task at the remote site, and the human operator is responsible for higher control issues like planning and perception. A lot of detailed information on all aspects of telerobotics is given in [32]. Basically, [33] introduced a classification of telerobotic control schemes which are still valid today:

- **Direct continuous control:** Also known as master/slave control, whereas the remote robot follows the inputs of the controller.

- **Shared continuous control:** Here, control is placed at a higher level, and thus, the remote device may vary from its planned course if it encounters a problem or an obstacle.

- **Discrete command control:** The controller of the remote device can carry out discrete commands autonomously, which requires a higher level of capabilities at the controller to perform these actions without the help of a human operator.

- **Supervisory control:** The remote robot can operate autonomously and interacts only in case of unexpected situations with the human operator.

- **Learning control:** The remote device is able to learn from human inputs and sensor data in order to generate a behavior for similar situations. In case such a similar situation occurs in future, the robot can react autonomously without requesting the help of the human operator.

Now, the characteristics of the underlying communication channel are very important aspects, which directly influence the decision which of the above presented telerobotic control schemes can be applied. In case of rather unpredictable, variable, and sometimes large delays as they are often present in Internet connections, the use of continuous control might not be a good choice. Usually, this approach causes severe problems in such a scenario, due to the influence of the delays (cf. [34]) might lead to instabilities.

In 1998 and 1999, [35], [36] and [37] realized systems applying shared continuous control via short high bandwidth connections. As soon as the closed loop control is done locally - as it is done usually for discrete command control, supervisory control, and learning control - the approach is even more robust on the delay effects. The team of Goldberg (cf. [30] [38] [39]) developed several robot systems which were teleoperated via Internet. The first system which allowed the remote users to view and alter a real world environment using telerobotics via the Internet was the the *Mercury Project*. It was online from September 1994 until March 1995 and users had to excavate artifacts buried in a sand-filled terrarium. The Raiders Robot was one of the first robots being remote operated via the web browser. Later, in 1995, the University of Southern California developed the *Telegarden*, which was active until August 2004. This system allowed the user for interaction with a remote garden with real living plants. In 1999, Carnegie Mellon University (CMU) developed the *Xavier* robot (cf. [40]). It was a mobile robot and the research emphasis of this project was on the local intelligence of the robot and on autonomy and supervisory control aspects. Also in Switzerland, several researchers were active at the end of the nineties and developed a number of robots teleoperated via the Internet (cf. [41] [42]). An example is the *Khep* robot of the project *RobOnWeb*. In 1997, also the continuous control scheme was applied for controlling a robot with force feedback via the Internet (cf. [43]). The research focus of this project was on the delay induced by the Internet. Here, also the packet loss was identified as a serious problem. A combination of supervisory control and shared continuous control was analyzed and evaluated in [36] and [37]. A robot was teleoperated via UDP and the research of this project was focused on a network model for variable delay. A model-based system was developed by [44], whereas the research goal was to allow an intuitive and easy way of programming a robot and using a 3D model of the robot's environment. The teleoperation of this system was demonstrated in 1996 between Montreal and Vancouver.

The remote operation of robots was also a very promising approach in Space Science, and thus, also NASA[4] emphasized this topic. [45], [46], [47] developed a web interface (WITS - web interface for telescience) for the remote operation of rovers in distant locations - e.g. on other planets like Mars. This project provided also a knowledge base for the Mars Pathfinder mission in 1997 and also for the 2003 and 2005 missions to Mars. Also the DLR[5] is active in these research areas (cf. [48] [49] [50] [51]) and contributes to current missions e.g. TECSAS, where the grasping of a tumbling target satellite is addressed.

Based on the experiences of the above telerobots, the development of complete tele-laboratories was the subsequent step ahead for the researcher community.

### 2.1.4 Tele-Laboratories

In the initial stage of tele-education via TV in the sixties, there was no way to implement experiments remotely as bidirectional communication between teacher and student was not possible. Therefore, presence phases in specific laboratories had to be organized for the students. The situation only changed during the nineties, when increased capabilities of the Internet enabled access to equipment via the Internet and interactions between students and hardware were possible. Nevertheless, challenging implementation problems in the tele-education context had to be solved before complete educational units consisting of lectures and experimental components could be realized [52]. While mainly techniques to teleoperate hardware via Internet were investigated in the beginning [53], [54], [55], [56], as late as after 2000 whole educational units have been developed [57], [58], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71]. Currently, several systems are operated successfully. [72] proposed a framework using Matlab/Simulink and Labview to implement experiments in the area of automation and control. In [73] a system for programming robots remotely is presented. Hardware can be programmed to participate in games, in order to accomplish an objective in a remote environment. Since the mid-nineties research is focused on tele-experiments with real hardware, addressing in particular mobile robots [19], [74], [75], [20], [76], [4]. Mobile robots offer a motivating, interdis-

[4]National Aeronautics and Space Administration

[5]Deutsches Zentrum für Luft- und Raumfahrt

ciplinary field, where theoretical background can be directly transferred to experiments with interesting industrial applications [74]. During the last years, several e-learning systems were developed. Usually, these systems use their own content management system which decreases the level of interoperability between these systems. The recent research focus in this area is set on using service oriented architectures in tele-education systems. Several approaches are discussed, like the implementation of architectures based on web services [77] [78] or multi-agent approaches in service oriented architectures [79]. Using service oriented architectures allows a modularized large-scale interoperability for different kinds of e-learning systems. Also incorporating the spirit of the Semantic Web into tele-education systems is investigated, which aims on a seamless semantic understanding of the learning content.

### Design and Architectures of Remote Laboratories for Mobile Robots

During recent developments in the area of tele-laboratories with real hardware experiments, several architecture design approaches were used. In [80] access to a Linux based PC is granted to the user and remote access is provided via the remote control software VNC. This Internet enabled PC is connected to the hardware and the user can directly use the hardware via the VNC remote desktop. For this approach a lot of security relevant aspects must be considered in order to prevent the hardware from damages or to disable a misapplication of the PC. In [81], a classical client-server approach is applied. Here, virtual reality methods are used to enable simple remote operation of a mobile robot. [82] describes a hardware experiment which is provided to students over the Internet. [83] also implemented a client-server approach in combination with .NET and Macromedia Flash technologies. In [84], the client-server design is combined with a Matlab/Simulink server, and hardware from Quanser is connected via the WinCon real time system. In 2003/2004, the BMBF [6] funded the nationwide tele-laboratory in Germany LearNet which hosts several experiments with real hardware. This project also used a client-server architecture for each experiment [85] and has an independent user management for each experiment.
About ten years ago the possibilities for the generation of dynamic web content was only possible with a few technologies (e.f. common gateway interface

[6]German Federal Ministry of Research and Technology - Bundesministerium für Bildung und Forschung

(CGI), or using Perl), and the capabilities of these systems were limited. Nowadays, state of the art techniques for the generation of dynamic web content like PHP, complete frameworks with template engines (e.g. smarty[7]) or rich Internet applications like Adobe Flash, Java$^{TM}$WebStart, JavaFX, or Microsoft Silverlight allow for high level of interactivity and flexibility. These technologies provide a good basis to implement highly interactive remote learning units.

## 2.2 Remote Learning Units of Mobile Robots

This section presents a tele-laboratory which provides several experiments with real hardware being accessible via Internet, whereas the experiments are distributed all over the world. The design of the tele-laboraoty in Würzburg is driven by the above mentioned aspects. A twenty-four hour availability seven days a week combined with a minimum of administration effort with respect to user management, hardware management, tutorials, and learning material, and also a minimum of maintenance effort, which means design of a robust hardware unit, autonomous prevention of hardware damage, are the prerequisites for a successful tele-laboratory. In addition, the worldwide availability and connectivity via Internet plays also a key role, as a reliable communication between user and hardware provides a stable base for real hardware teleoperation experiments offered to a world wide user community of students accessing mobile robots via Internet. Thus, the communication link must fulfill minimum requirements which are described in the following sections in order to allow control of the hardware. The successful application of a client-server model in the above mentioned projects also made it to be a suitable choice for the tele-laboratory of Würzburg [20] [21] which is presented in this work.

To handle the variable delays mentioned in the previous paragraph, asynchronous teleoperation is supported by the back-end architecture described in the next sections. Also the developed front end which was developed to provide the required interactivity between user and remote hardware is presented. The components of the tele-laboratory which were developed for this work are currently integrated into the project "International Virtual Laboratory on Mechatronics" [3] which was funded by the European Union[8]. Within this European Union project,

[7]http://www.smarty.net/ (02.09.2010)

[8]European Union Cross Cultural Program ECCP

international partners form Spain (Universidad Carlos III de Madrid) and India (Anna University, Chennai and Thiagarajar College of Engineering, Madurai) developed world wide distributed hardware experiments which were integrated into the architecture developed in the frame of this thesis. All the developed learning units are accessible via Internet, and the interaction between users and hardware is realized over specially designed web based front. The objective of this European Union funded project was the integration of different hardware experiments in the area of mechatronics contributed by the project partners. These distributed learning units are accessible via a common web-portal following standards like Java$^{TM}$Webstart and finally access to this interactive services are provided to students via the Internet [2].

## 2.2.1 Architecture of the Hardware Segment

The analysis of several other tele-laboratory implementations [83] [80] [84] [81] [85] (cf. Section 2.1.4), and the experiences made with these implementations approved the decision to use a classical server-client approach for connecting the hardware to the Internet. Figure 2.2 shows the components of the tele-laboratory which is currently running at the University of Würzburg. To keep the architecture modular and scalable, the technical experiment components like robot control server, control client, and hardware are strictly separated from the administration part like student web portal, database, web-server, and administration interface. Basically, each hardware experiment component has its own dedicated control server to support the different hardware interfaces, and to allow for a distribution of the hardware experiments to different locations so that also a world-wide distribution – which is an objective of the presented tele-laboratory – is supported.

#### Robot Control client

The client itself must guarantee platform independence, give an interface to control mobile robots, and provide feedback about hardware and communication link – also in case of communication restrictions like unreliable links or low bandwidth environments. The initial access to the robot control client is managed by a web-portal which provides access to tutorials for the corresponding experiments, time reservation for hardware experiments, online multiple choice exams,

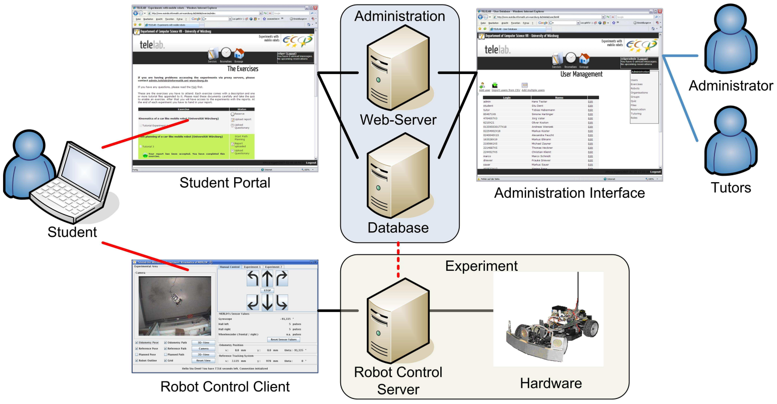
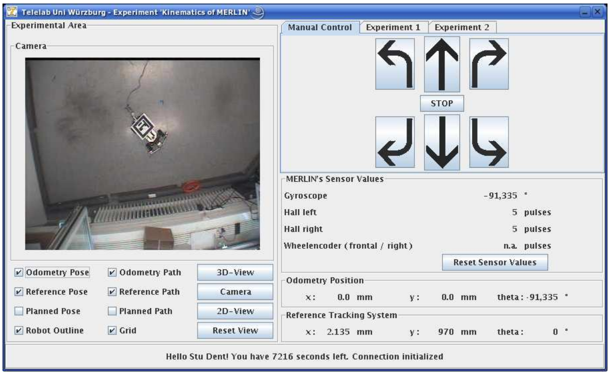


Figure 2.2: Components of the tele-laboratory architecture at the University of Würzburg.

a possibility to upload experiment reports, and personalized views of the user's progress for each experiment [3]. Access to hardware requires a valid verification and time reservation for a user. After successful verification of the user, access to the robot control client (cf. Figure 2.3) is granted. The robot control client program itself is a standalone application using the Java$^{TM}$ Web Start Technology. Thus, all Java$^{TM}$ supporting platforms can be used as client computer. Starting the client invokes an authentication procedure which checks the database for a valid user registration, and a valid time slot for the user and the corresponding experiment. This authentication mechanism is based on the http-protocol and is not connecting directly to the database (cf. Figure 2.4). Furthermore, an http-request with information about the current session is sent to the web server. The web server replies to this request by sending data in XML-format (cf. Listing 2.1). This reply is mainly structured in two parts: the user section and the reservation section. The user section contains detailed user information like name, last name, and course information. The reservation section contains valid reservations for the requesting user with start and end time stamps in the unix time stamp format. The client itself processes this reservation data and decides whether a valid time slot is available. If the time slot reservation is valid, the server accepts the authentication request of the client which is identified by its *SessionID*. After successful



MERLIN's Sensor Values

Gyroscope -91,335 °
Hall left 5 pulses
Hall right 5 pulses
Wheelencoder (frontal / right) n.a. pulses

Odometry Position
x: 0.0 mm y: 0.0 mm theta: -91,335 °

Reference Tracking System
x: 2.135 mm y: 970 mm theta: 0 °

✓ Odometry Pose ✓ Odometry Path
✓ Reference Pose ✓ Reference Path
☐ Planned Pose ☐ Planned Path
✓ Robot Outline ✓ Grid

Hello Stu Dent! You have 7216 seconds left. Connection initialized

Figure 2.3: Robot Control Client

authentication, the client connects via TCP/IP connection to the robot control server using a dedicated port. The experiment area is equipped with a camera and a live video stream of the experiment is available. With respect to link quality and available bandwidth, the frame rate and quality of the video must be scalable. Also, for the universal and world wide usage of the experiments a minimum of administration effort for the network (e.g. firewalls ...) must be achieved. Thus, the video stream is encapsulated by the http-protocol (cf. Figure 2.4) in order to avoid special firewall configurations.

```
<?xml version="1.0"?>
<!DOCTYPE reservation-info SYSTEM "reserv-xchng.dtd">
<reservation-info>
        <server-status>
                <time>122211212</time>
        </server-status>
        <user>
                <firstname>John</firstname>
                <lastname>Doe</lastname>
                <group>1</group>
                <organisation>Uni-Wuerzburg</organisation>
                <matrikel>1234567</matrikel>
        </user>
        <reservation>
                <resource>1</resource>
```

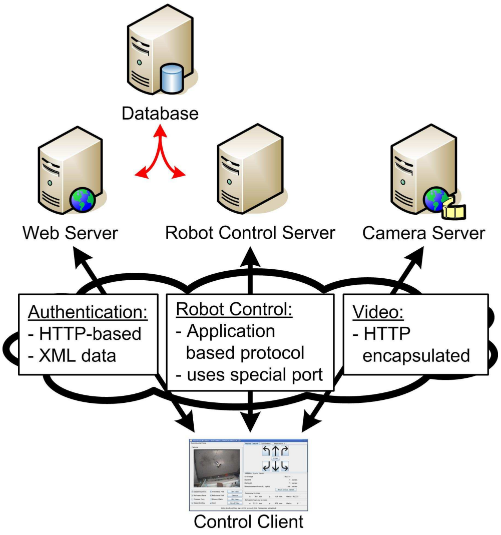


Figure 2.4: Communication scheme of the control client program.

```
                <start-date>1131670123</start-date>
                <end-date>1131672233</end-date>
        </reservation>
</reservation-info>
```

Listing 2.1: Reply of the web server in XML format.

The designed experiment uses http on standard port 80 for all its traffic, including authentication data and camera picture. To keep the client platform independent, it is based on the Java$^{TM}$ Web Start technology. Once the client is started, the current version is downloaded and stored in the cache memory of the client PC. Any further start checks for updates and – if available – installs them.

### Robot Control Server

Within the presented architecture, the central unit of each tele-experiment is the robot control server. This server has several tasks and the hardware as well as in the software structure will be described in this chapter.

### Specific tasks of the server

The robot control server is the key component in the scope of this specific tele-experiment and has to perform important tasks in the experiment procedure. The first group of tasks is related to administration, like authentication and adherence to time schedules. The authentication is a very crucial point, as the user is directly connected via the robot control client to the experiment without any explicit intermediate authentication entity. This means, the server has to check again that only valid users have access to the experiment. Furthermore, only a certain time slot is reserved for any user, which means that the adherence to the time schedule has to be guaranteed by the server.
The second group of tasks concerns the operation and execution of the experiment itself. This involves several subtasks, like steering the robot component of the experiment, data acquisition from the different sensors and components, data exchange with the user, logging of the experiment procedure, and fault detection. This broad variety of tasks requires a stable and reliable server component.

### Software Layer

The whole software is implemented in the Java$^{TM}$ programming language and is therefore operating system independent and in the presented setup, a Linux operating system (SuSe) provides the necessary basic functionality. To guarantee the modularity and extensibility of the system, configuration files are stored in the XML format. This is very important to be independent from the used robot and hardware architecture and for providing extensibility and modularity. The server acts as an interface between the user, who is connected via Internet, and the robot, which has to be controlled from the user to perform the experiment (cf. Figure 2.5). To grant the access to the experiment only to valid users, the server

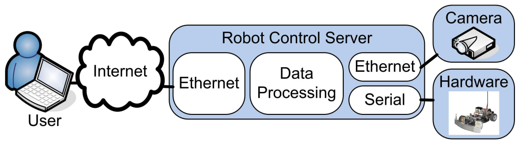


Figure 2.5: Robot control server.

has to check for each user for valid reservation of the robot resource. This validation is performed through the same interface as described in section 2.2.1, which is provided by the administrative entity. It enables the server to crosscheck the reserved time slots with the main database. This procedure compares the ID of the user with his reserved time slots represented in the authentication XML-data (cf. Listing 2.1). The interface to the Internet is implemented with standard socket communication over TCP, and the contents of the robot control client are wrapped in standard HTML to enable easy access to the experiment over a normal Internet browser. The communication to the robot is realized with an especially designed communication protocol for the MERLIN robot. This communication protocol is used to exchange sensor and command data between the robot and the server via serial connection. Therefore, the Java$^{TM}$ Communications package was integrated in the server. Another essential task of the server is the data processing of different data streams, as it is not possible (and desirable) to transmit all sensor data obtained during the experiment to the user (i.e. camera stream, sensor data, positioning data, acknowledgments). Therefore, the server collects all information and decides, according to a predefined profile, which data has to be relayed to the user. In that way it is possible to regulate the amount of data which is exchanged between server and user – for example if only a low data rate is available for the user. The profiles for this selection are also XML formatted and can be extended easily. To enable a 24 hour operation of the experiment it is very essential to run the server without continuous monitoring of a human operator. Therefore, an error detection was implemented to inform the operator via email if any error occurs during operation. This significantly reduces the maintenance effort for the experiments.

#### Hardware Layer

The server hardware contains two processors (Intel Pentium 4 with 3.40 GHz) and has 1 GB of main memory. This provides a sufficient amount of computing power to perform the different tasks mentioned in the previous section. The server is the physical interface between the Internet and the tele-experiment with its robot hardware. It has two ethernet connections, one for the connection to the Internet, the other for an internal network, which provides the server with camera and positioning data. The robot is connected with a standard serial connection (see Figure 2.5). This topology is very useful, as the server has to combine all the information from the different components to relay experiment specific information to the user. The server acts as a central entity of the experiment procedure, but is independent of the administration which schedules the students (see Figure 2.7, [3]). The main requirement to this hardware architecture is stability and reliability, as it has to run 24 hours a day for experiments.

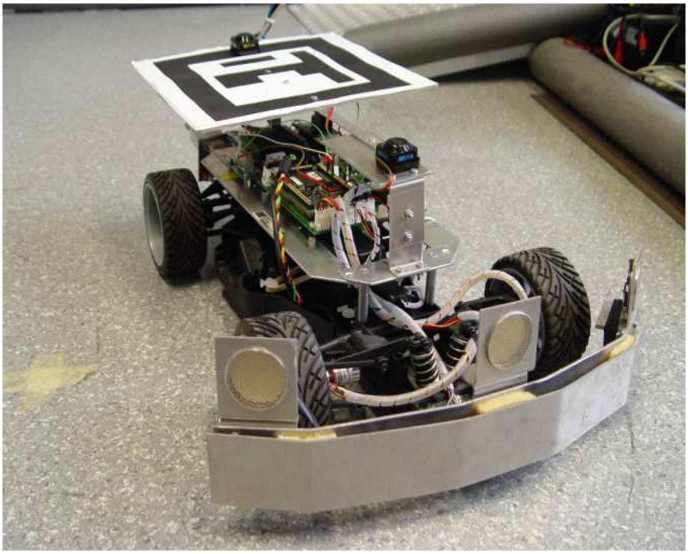


Figure 2.6: The MERLIN robot for the tele-laboratory in Würzburg.

### 2.2.2 Robot Hardware

The robot connected to the control server is a car like mobile robot based on the indoor version of the MERLIN (Mobile Experimental Robot for Locomotion and Intelligent Navigation) robot platform [55] (see Fig. 2.6). It can be remotely controlled from the user via the Internet. The core of the robot consists of a 80C167 microprocessor, which is used for data acquisition from the sensors and the communication with the server. Connected sensors are ultrasonic sensors, gyroscope and a wheel encoder. Thus, the MERLIN robot can be used for a broad spectrum of experiments, ranging from kinematic calculations to obstacle avoidance and path planning [74]. The MERLIN vehicle is based on Ackermann steering, and therefore has non-holonomic constraints. A more detailed technical description can be found in [86]. The purpose of the experiments is the demonstratoin of the kinematic correlations of the robot and the comparison with the real movements depending on characteristic parameters. Thus, the students gain basic knowledge in kinematics on real hardware via the Internet.

### 2.2.3 Mobile Robot Tele-Education Experiments in Würzburg

The experiments implemented in Würzburg combine the application of telematcis methods in parallel to the knowledge learned for performing the experiments, and thus, extend the education of engineers, as well as students of natural sciences with a qualified practical education. Basically, two different mobile robot designs are used for these experiments, the differential drive robot TOM robot (Tele-Operated Machine, see [87]) and the car-like MERLIN robot. In contrast to many other remote controlled robots in the Internet, the University of Würzburg provides seven interactive learning units related to mobile robots. These experiments are accessible via the web portal of the Virtual University of Bavaria[9]. The experiments related to the kinematics of mobile robots introduce the students to the basics of the holonomic and non-holonomic mobile robots. The focus of these experiments is set on the problems experienced during the work with real hardware. In contrast to ideal simulation models, real hardware and real sensor measurement data is always influenced by external sources of errors like friction, inertia, and sensor or actuator deviations.
Furthermore, the students are introduced to the differential drive mobile robot TOM and the car-like mobile robot MERLIN to analyze the different kinematics constraints caused by the different mechanical constructions. For the TOM robot, the kinematics and inverse kinematics (calculate the control commands to move the robot from pose A to pose B.) is covered with an experiment. MERLIN explains the functionality of the commonly used steering mechanism in automobiles - the Ackerman steering. Furthermore, the students learn how to handle the behavior of the robot in a mathematical way and how to deal with the inverse kinematics problem for this non-holonomic robot. Finally, these experiments explain some basic procedures to deal with real sensor data and systematic errors. Therefore, a simulation of an idealized model is given to the students. After important hardware parameters are determined, the ideal simulation of a movement

[9]http://www.vhb.org/ - Virtuelle Hochschule Bayern (09.02.2010)

is compared to the real path traveled by the MERLIN robot in order to analyze sources for the deviations (e.g. friction, inertia, or systematic errors).

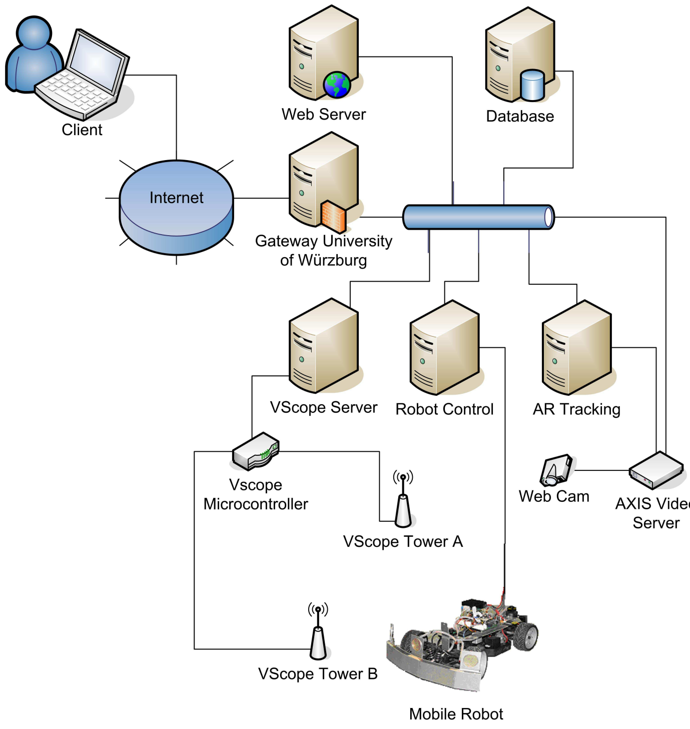


Figure 2.7: Example setup of the MERLIN experiment at the University of Würzburg.

The setup which is described in the preceding sections of this work is used for these integrated remote-experiments (cf. Figure 2.7). The administrative parts like user management, hardware booking, and user validation are implemented on the web server and database component. The web server also hosts the robot control client. The hardware segment at least consists of a robot control server and the mobile robot. Figure 2.7 shows the tele-laboratory for the MERLIN robot which is enhanced by a VScope server and the ARTracking server which provides real time position information and camera/video control for the mobile robot during the experiment.

The second main topic of the remote experiments is the motor-controller synthesis for mobile robots. Usually, mobile robots are driven by direct current (DC) electrical motors. Thus, the control of DC motors is a key issue while analyzing the dynamics and kinematics of mobile robots. The experiments for this topic are dealing with the basics of DC motors, the PWM (pulse-width modulation), and the mechanical and driving system of MERLIN and TOM. Furthermore, an introduction to control theory is given, including the theory and methods for control system analysis and synthesis. The focus of the TOM experiments is set on the control of both electrical motors, and the objective is the determination of the transfer function of an open-loop controller, a closed loop controller, and the step response of the system. Finally, the designed PID controllers for both robots are tuned using the Ziegler-Nichols rules.

In the following, a short description of experiments from the users view is given. Before the students are allowed to access the hardware, a registration for the experiment web portal is required. This experiment portal provides online access to a tutorial and background information for each experiment. Each tutorial gives the information required for a compulsory multiple-choice test which has to be passed by the students in order to sign in for real hardware access. Finally, the tutorial provides some questions which should be answered in an experiment report. After the student hast booked a time slot for experiments with the robot assigned, he can start the client software through the web-interface. This client is based on modern Java$^{TM}$web-technologies, and is transferred directly from the web-browser to the local machine of the student. One of the first tele-experiments is related to path planning for a mobile robot and contains several subtasks, starting with the determination of the position error after several simple movements. The student experiences in this experiment important aspects of real hardware operation. In the next subtask, the obtained knowledge is used to solve a more complex path planning problem, e.g. combining simple movements to a complex path and minimizing the accumulated position error. The tasks of this experiment can only be fulfilled by a strong interactivity between user and robot, and the complete system is specially designed in order to prevent from data loss and the failure of the complete experiment in case of a communication link break down. If the connection to the robot is interrupted due to problems with the Internet connection, the experiment can be captured again by simply reconnecting the client. The robot will resend its current sensor and position values to the client and the user can continue the experiment. The Java applet also provides the possibility for downloading the measured sensor data for further data processing and analysis. This is necessary for answering the questions for the experiment report. The experiment report can be uploaded to the portal where it can be accessed by a tutor for correction and evaluation. The portal also provides information for the students about their status and their results of each experiment. Other experiments of the tele-laboratory are related to control engineering, for example PID controller tuning. As mentioned above, a control loop directly between the user and the robot is not allowed by the system and in the present case, it is not required anyway. Therefore, the student has to parametrize the PID controller in the robot control server and observe the behavior of the robot. The server reports the sensor reply from the robot and enables in that way an evaluation of different control parameters. The stability of the communication is also in that case not critical, as the robot control server ensures the correct operation of the robot and retrieving the necessary data. The application of the tele-laboratory in the lectures and exercises at the University of Würzburg involves students regularly in the experiments and guarantees a steady improvement of the experiments. The presented tele-laboratory concept proved to be very successful in the curriculum of the students and is an excellent alternative to classical experiments related only to simulations. Additionally, the flexibility of the presented architecture also allows for an extension in terms of the number and type of experiments, and in terms of accessibility and availability of experiments.

### 2.2.4 Integration of World Wide Distributed Remote Experiments

This section describes improvements of the above presented architecture and shows how this system can be used for providing world wide access to hardware experiments which can also be located at different places. Within the European Union funded project "International Virtual Laboratory on Mechatronics", the architecture proposed in the previous section is extended in order to realize a world wide distributed tele-laboratory which provides experiments with real hardware. To get a first idea of the present communication channel characteristics, the measurement of delay times and their distributions were performed for these long distance communication via Internet [2]. The following sections give details on the results of these measurements and describe the experiment content provided by the "International Virtual Laboratory on Mechatronics".

### Using European Tele-Experiments from Asia

To get an idea of the communication link properties which are present in international tele-laboratories, measurements of delay times and their distribution were performed. This section presents the results of these measurements between Asia and the remote-laboratory in Germany. Figure 2.8 shows the probability distribu-

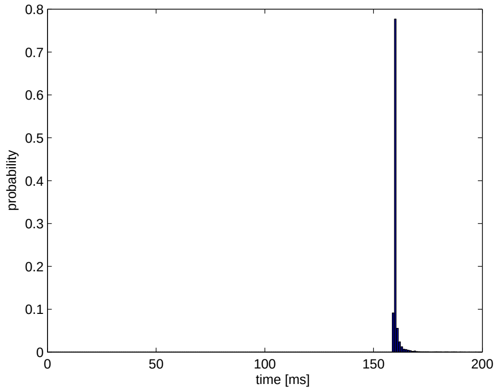
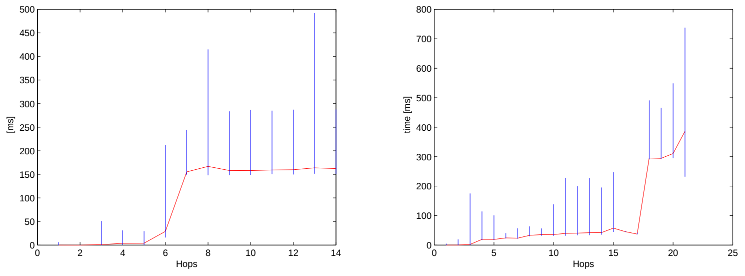


Figure 2.8: Probability distribution of delays between Würzburg (Germany) and Tianjin University (China).

tion of measured delays between University of Würzburg (Germany) and Tianjin University (China) which were taken at one day. The majority of the measured delay times results in a round trip time of about 160 ms. Thus, this would be an acceptable delay and is not affecting the usability significantly. In Figure 2.9b, where the delay of the communication link to India is measured, the response time is about 390 ms which is much more than twice the roundtrip time encountered for a tele-experiment setup inside Europe. Such a high delay for the teleoperation of hardware might be noticeable for the experiment user in case of designing control loops or in case of direct teleoperation, and thus must somehow



(a) China (b) India

Figure 2.9: Traceroute Measurements.

be considered during the design process of an experiment.

A further measurement which provides details of the used route from Würzburg to Tianjin is presented in Figure 2.9a, measurements related to the route from Würzburg to Chennai (India) are presented in Figure 2.9b. The values are the result of several *traceroute* measurements carried out within a time span of two weeks, whereas a single measurement was taken 5 times a day with a duration of 15 minutes. The graph shows the average value of the response time of each hop of the used route through the Internet. In addition, the vertical lines show the minimum and maximum values which were measured for the corresponding node. For the interpretation of the displayed data, some more detailed information must be known. The used *traceroute* implementation sends several packets to the destination IP address and increments the time-to-live (TTL) by one for each packet starting from TTL=1. Each receiving host decrements the TTL by one. A router receiving a packet which should be forwarded and has a TTL=1 discards this packet and replies with Internet Control Message Protocol (ICMP) reply number 11 (*time-to-live exceeded in transit*). In case the destination host receives the packet with TTL=1, the reply would be ICMP reply number 3 (*Destination Unreachable*). Thus, the sending host collects the IP addresses of all hosts for the specific route. Now, the result could be influenced by several things. Firewalls, ip-tunneling, load balancing of network segments, network address translation, or even erroneous IP stack implementations might lead to a discovered route which does not correspond to the real one. Also the measured

| Route to China | | Route to India | |
|---|---|---|---|
| **hop no.** | **location** | **hop no.** | **location** |
| 1,2,3 | Würzburg, GER | 1,2,3 | Würzburg, GER |
| 4,5 | Frankfurt, GER | 4,5 | Frankfurt, GER |
| 6 | Athens, GRE | 6,7 | Hamburg, GER |
| 7,8 | Beijing, CHN | 8,9,10,11 | Frankfurt, GER |
| 9,10 | Guangzhou, CHN | 12,13 | Amsterdam, NED |
| 11,12 | Beijing, CHN | 14,15,16,17 | London, GBR |
| 13,14 | Tianjin, CHN | 18,19,20 | Bombay, IND |
| | | 21,22 | Chennai, IND |

Table 2.2: Locations of nodes for the routes to China and India.

response time can behave strange. In case ICMP traffic is prioritized lower than other traffic, replies can be delayed in an unpredictable way. Thus, the effects displayed in Figures 2.9a and 2.9b can be explained easily. The average response time of hop 8 in Figure 2.9a is higher than the measured average response time of hop 9. This is a result of different priorities of ICMP traffic in the router configuration. A similar behavior can also be observed between node 20 and 21 in Figure 2.9b, where the minimum response time of hop 21 is smaller than the minimum response time of hop 20. Nevertheless, both figures give a good view of the composition of the resulting overall delay, and allow also a more detailed analysis of the properties of the link with respect to the realization of world wide accessible tele-experiments.

As shown in Table 2.2, hops 1 - 6 of Figure 2.9a are located inside Europe. Here, the response times are very low. The long distance connection to China induces the first larger delay of about 140 ms. For the remaining hops, the average response time stays almost constant at about 160 ms. Nevertheless, as it can be seen from the plotted maximum values of the measurements, hops 7 - 14 induce a larger variability of the delays.

For the measurement to India, a different behavior is observed. Here, hops 1 - 17 are located in Europe (cf. Table 2.2 and Figure 2.9b). In contrast to the connection to China, already hops 10 - 17 are responsible for a variability of the delays. As expected, the connection between Europe and India induced the largest part (about 250 ms) of the overall delay, which is about 390 ms.

Thus, at least a delay of 160 ms is present for an Internet connection to China, and to India, a delay of at least 390 ms has to be expected. In general, the present delays will be visible for the user in case of direct teleoperation, and it has to be considered in case of implementing even very simple control mechanisms via this communication link. Investigating the behavior of the last few hops which are all located inside China or India (cf. Figures 2.9a and 2.9b), another aspect is visible which has to be considered. These Figures show the measured minimum and maximum values of these response times, and here, a large variance is observed. These variabilities of the response times are very difficult to handle. With respect to direct teleoperation, where the user reacts directly on the feedback he received from the robot by means of a video stream or sensor data, a high variability of the responses will cause an uncomfortable feeling for the user in some ways. First, the received video has a variable frame rate which is very hard for a human operator to understand in order to gain knowledge about the situation of the vehicle to be teleoperated. Second, the control commands given by the human operator are based on delayed information about the vehicle. Both aspects lead to overreaction on the human side by the means of generating control commands, and the teleoperated vehicle will behave unintentionally. Realizing a complete control loop under these conditions is also very difficult.

#### Implemented Experiments

Within the project European Universities and Indian Universities provide hardware experiments which are hosted via the portal developed in Würzburg [19]. Each project partner in Europe and India hosts two different hardware experiments in the area of mobile robotics, control engineering, path planning, as well as robotic manipulators. In addition to the previously presented tele-laboratory architecture, also two experiments were developed for the University of Würzburg in the frame of this thesis work. Each experiment is designed for a 4-hour period and address the following specific topics:

- Modeling of a vehicle with Ackermann-steering and identification of crucial kinematic parameters: A robot with Ackermann-steering has to follow certain non-holonomic kinematic constraints. A key objective of this experiment is the mathematical modeling of the robot behavior during movements.

- Motor control for a mobile robot: Often it is necessary to move a mobile robot with a defined constant velocity. Different types of surface conditions or uphill and downhill slopes make this a rather difficult task and require the implementation of control mechanisms.

- Path planning for a non-holonomic vehicle: With respect to the non-holonomic constraints mentioned above, path planning for this type of robots is not a simple task. For this experiment several path planning methods are implemented and compared in different environments.

- Modeling the kinematics of a differential drive rover: A differential drive mobile robot has one axis with two powered wheels. These wheels can rotate with different angular velocities and also in different directions. This ability allows a high mobility and causes different kinematic constraints compared to a mobile robot with an Ackermann-steering. The experiment deals with the mathematical description of the differential drive.

- Design of a speed control for a differential drive rover: For a differential drive mobile robot it is very important to have a reliable velocity control for these two powered wheels, as already a small difference in the rotation speeds will result in a change of the moving direction.

- Mathematical modeling of the robot control and application of tuning rules for a PID controller: Control engineering provides a lot of methods for tuning control mechanisms. This remote experiment is used for an application of these tuning mechanisms. The results are visualized with the real mobile robot hardware and the effects of different parameter settings of the control mechanisms are simple to understand. These hardware experiments provide a good portal for applying theoretical approaches to real hardware.

- Implementation of algorithms for obstacle avoidance, navigation, and path planning: Path planning in a static environment is already addressed in a separate experiment. As soon as dynamic environments with moving obstacles (e.g. people or other robots) are considered, a dynamic re-planning is required.

The first experiment of the University of Würzburg is on the kinematics of the indoor MERLIN robot. This experiment should help to understand indispensable basics of the mobile robots' kinematics. It provides the background for the remote kinematics experiment of the virtual laboratory at Würzburg. Exercises are constructed in order to show to the student most of the non-idealities of the real hardware. Obtained data can be easily used for accuracy improvement by high precision mathematical model building. The second experiment of Würzburg covers pathplanning of a car-like mobile robot. The experiment uses the non-holonomic indoor MERLIN robot with an Ackerman-steering and demonstrates the problems of the inverse kinematics of this kind of mobile robot. Basic motions are combined to primitive maneuvers, like three-point-turn and sidewise shifting. The next step is the combination of primitive maneuvers to complex maneuvers for achieving a certain configuration. This experiment shows how to combine primitive maneuvers in order to solve the inverse kinematics problem, and gives a detailed description of these maneuvers. These experiments are designed ac-

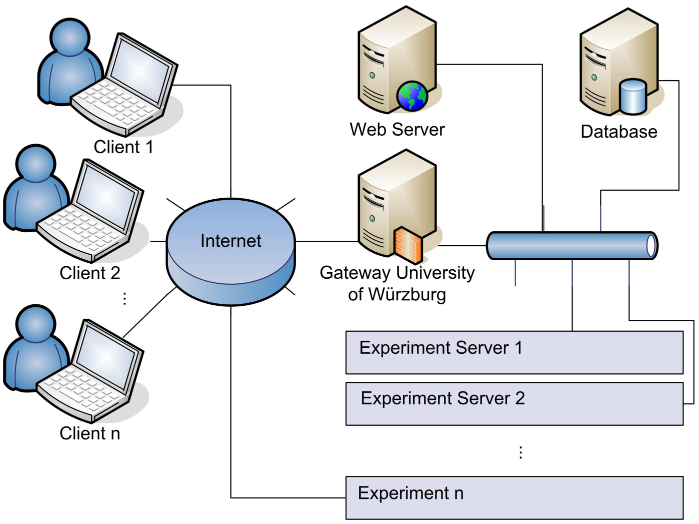


Figure 2.10: The integration of distributed remote experiments

cording to the previously mentioned guidelines and requirements (see also [22], [19], [3]) in order to provide robust teleoperation features via Internet. Each of the experiments uses an *experiment server* which can be either a dedicated PC or a logical instance of a program hosted by an application server. The setup of such a scenario is depicted in Figure 2.10, and provides the high flexibility which is required for such a kind of distributed tele-laboratoy. Thus, the international tele-laboratory partners in Europe (Universidad Carlos III de Madrid, Spain[10]) and India (Department of Mechanical Engineering of Anna University in Chennai, and Department of Computer Science and Engineering of the Thiagarajar College of Engineering in Madurai) can integrate own experiments by setting up an experiment server which supports the protocol developed within this work (cf. Section 2.2.1) in order to communicate with the administrative components of the tele-laboratory system.

Universidad Carlos III de Madrid developed an experiment on events detection and an experiment on topological navigation. The goal of the events detection experiment is an analysis how a door is observed by a laser sensor and to build an algorithm that detects the door from the data information. Here, the student will be able to move the robot in order to observe the image produced in the measurements of a laser telemeter in different places close to the door. With this information, the student should develop and test an algorithm that reliably detects the door. Students will be able to have contact with real sensors and to notice the difficulty of the information extraction from the sensor readings. The second experiment addresses topological navigation and the door detector algorithm developed in the previous experiment is used to perform a simple navigation task such as moving to a door and cross it. Different positions and different algorithms will be available during the experiment and allow for a validation of the developed concepts. Students will be able to have contact with real navigation problems related to the sensor system.

The first experiment of the Department of Mechanical Engineering of Anna University deals with motor control. Stepper motors are used as feed drives in the selected CNC machines, and by controlling the number of pulses, the slide position and pulse frequency for the velocity of the slide are achieved. Control of spindle speed is achieved by varying the voltage supplied to the DC motors of the spindle drive. The students have access on a tutorial which explains the required theoretical approaches. They write a program using G-Codes and M-Codes, execute them through the web and analyze the results. The second experiment of

[10]http://roboticslab.uc3m.es/ (02.09.2010)

Anna University is on a remote controlled robot arm. By controlling the movements of various joints and end effectors, the robot arm can move according to a program written by the student in the programming language C. The user is trained to operate the robot in order to perform various tasks such as pick and place, sorting of parts, or palletizing. The robot arm can be teleoperated either by own created programs or via a specially designed graphical user interface.

The Department of Computer Science and Engineering of the Thiagarajar College of Engineering developed an experiment for tracing the boundary of a wall and another one for a repeatability analysis of a mobile robot using computer vision. For the first experiment, the objective is to trace the boundary of the wall with a mobile robot using different navigation algorithms. During the experiment, the students are taught to write a program for enabling the robot to fulfill its task. The objective of the second experiment is an evaluation of the repeatability and accuracy of a mobile robot using computer vision technique. This experiment setup consists of one mobile robot and one high-resolution pre-calibrated camera which allows for interesting experiments on error propagation.

The successful realization of these different experiments above show, that the tele-laboratory approach developed in this thesis work is a powerful architecture which can be used for providing world wide access to distance learning units for interdisciplinary topics whereas the experiments itself can also be located all over the world.

## 2.3 Control of Quanser SRV02 Experiment over the Internet

The Quanser SRV02 position control experiment consists of a motor which is connected to a wheel via a gear. The wheel is also connected to an encoder which records the turn ratio of the wheel with a certain resolution. The objective of this experiment is the position control of the wheel. Usually, the used experiment hardware is connected directly to the PC with the control algorithm via a special hardware interface card which generates the voltage for the motor and connects the sensors to the PC. The control algorithm can be implemented in Matlab or C-code, and access to the hardware is provided by special program libraries from Quanser. In this work, the experiment is used in a modified hardware setup which is displayed in Figure 2.11. Here, the hardware component is connected to the controller component via an IP/Ethernet based network. The communication through the network induces delays between hardware and control PC which needs to be considered in order to design a stable and robust controller. The experiment hardware (motor) is connected to the hardware PC via the hardware interface and an interface card which allows for reading sensor data and setting the actuator inputs. The hardware PC can determine the position (angle of rotation) of the wheel. This data is now transmitted to the control PC via the interposed network with a defined sampling rate. The control PC is running an application with a control algorithm which is used to control the position of the wheel. Based on the received sensor data, the controller generates a control signal and sends it back to the hardware PC via the network. Now, the hardware PC applies a voltage to the motor which corresponds to the received control signal.

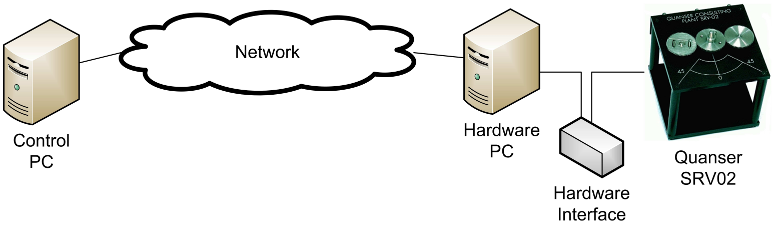


Figure 2.11: Modified hardware setup for the Quanser SRV02 Experiment.

### 2.3.1 Theory and Mathematical Background

As first step, the mathematical model of the above described hardware equipment is derived according to the technical data provided by the tutorial of the Quanser SRV02 Experiment. In Figure 2.12, the electric components of the motor are displayed. With armature circuit input voltage $V_m(t)$, armature resistance $R_m$, armature inductance $L_m$, armature circuit current $I_m(t)$, motor back-emf voltage $E_{emf}(t)$, motor shaft position $\Phi_m(t)$, and the torque generated by the motor $T_m(t)$ the following calculations can be done.

Applying Kirchhoff's voltage law leads to

$$V_m - R_m I_m - L_M \frac{dI_m}{dt} - E_{emf} = 0$$

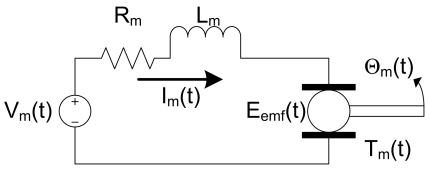


Figure 2.12: Electric components of the motor according to Quanser data sheet and technical specification.

The motor inductance can be neglegted as $L_m << R_m$. This results in

$$I_m = \frac{V_m - E_{emf}}{R_m}$$

It is known, that the back-emf of the motor is proportional to the motor shaft velocity $\omega_m$ with ($\omega_m = \dot{\Theta}$), leading to

$$I_m = \frac{V_m - K_m\dot{\Theta}_m}{R_m} \tag{2.1}$$

Newton's 2nd law of motion helps to characterize the mechanical aspects of the setup and results in the following equation with the motor moment of inertia $J_m$, the torque applied at the load $T_l$, the gear ratio between motor and load $K_g$, and the gearbox efficiency $\eta_g$.

$$J_m\ddot{\Theta}_m = T_m - \frac{T_l}{\eta_g K_g} \tag{2.2}$$

Applying the 2nd law of motion at the load of the motor (with $B_{eq}$ being the viscous damping coefficient seen at the output) gives

$$J_l\ddot{\Theta}_l = T_l - B_{eq}\dot{\Theta}_l \tag{2.3}$$

The combination of Equations 2.2 and 2.3 leads to:

$$J_l\ddot{\Theta}_l = \eta_g K_g T_m - \eta_g K_g J_m \ddot{\Theta}_m - b_{eq}\dot{\Theta}_l \tag{2.4}$$

It is known, that $\Theta_m = K_g\Theta_l$ and $T_m = \eta_m K_l I_m$ ($\eta_m$ is the motor efficiency), and thus Equation 2.4 can be rewritten.

$$J_l\ddot{\Theta}_l + \eta_g K_g^2 J_m \ddot{\Theta}_l + B_{eq}\dot{\Theta}_l = \eta_g\eta_m K_g K_t I_m \tag{2.5}$$

The the electrical properties of Equation 2.1 can be combined with the mechanical aspects of Equation 2.5 which results in the transfer function of the system:

$$\frac{\Theta_l(s)}{V_m(s)} = \frac{\eta_g \eta_m K_t K_g}{J_{eq} R_m s^2 + (B_{eq} R_m + \eta_g \eta_m K_m K_t K_g^2) s} \tag{2.6}$$

In Equation 2.6 $J_{eq} = J_l + \eta_g J_m K_g^2$ can be considered as the equivalent moment of inertia of the motor system as it is seen at the output. With

$$\begin{aligned} b_1 &= \eta_g \eta_m K_t K_g \\ a_1 &= J_{eq} R_m \\ a_2 &= B_{eq} R_m + \eta_g \eta_m K_m K_t K_g^2 \end{aligned}$$

Equation 2.6 can be rewritten in the following transfer function:

$$\frac{\Theta}{v_s} = \frac{b_1}{a_1 s^2 + a_2 s} \tag{2.7}$$

The system of Equation 2.7 can be represented by the following differential equation:

$$a_1 \ddot{\Theta} + a_2 \dot{\Theta} = b_1 v_s \tag{2.8}$$

The transformation into the state space can be done as follows:

$$\begin{aligned} x_1 &= \Theta \\ x_2 &= \dot{\Theta} \end{aligned}$$

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= \ddot{\Theta} = \frac{-a_2}{a_1}\dot{\Theta} + \frac{b_1}{a_1} v_s = \frac{-a_2}{a_1} x_2 + \frac{b_1}{a_1} v_s \end{aligned}$$

$$A = \begin{pmatrix} 0 & 1 \\ 0 & \frac{-a_2}{a_1} \end{pmatrix} \tag{2.9}$$

$$B = \begin{pmatrix} 0 \\ \frac{b_1}{a_1} \end{pmatrix} \tag{2.10}$$

$$\begin{pmatrix} \dot{x_1} \\ \dot{x_2} \end{pmatrix} = A \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + B v_s \tag{2.11}$$

### 2.3.2 Implementation of the Controller

#### Direct Connection between Controller and Hardware

The standard setup of the Quanser SRV02 experiment uses a special interface card in the PC to send commands to the hardware module and for capturing sensor data. Program libraries allow to include access to the hardware in third party software and real time ability is provided. In a first step, a controller is implemented locally on the same PC which is connected to the hardware module. The functioning of the control software is shown in Figure 2.13. The `init` procedure reads the program parameters `frequency` and `gain` and initializes the variables `gain`, `count`, and `voltage`. After the hardware module is initialized, it starts sending encoder values with the given `frequency` and accepts commands. These commands are applied immediately at the actuator. The main program has a loop which consists of three steps: waiting for encoder data from the hardware, compute a new voltage according to a control law, and send the new voltage to the hardware. In this case, where the controller is implemented locally, a proportional controller is applied to generate a new output, in order to reach the goal position `command`. The value of `voltage` is calculated based on the input variable `count` which corresponds to the counter values of the encoder:

$$voltage = -gain \cdot \left(command - \left(count \cdot \frac{360}{4096}\right)\right); \qquad (2.12)$$

This simple control law can now be tuned in order to achieve defines responses of the system (for more details please refer to the tutorial of the Quanser SRV02 Experiment), whereas the controller and the system are connected locally without using a communication link which induces delays.

#### IP Connection between Controller and Hardware

In order to allow for controlling the above described hardware equipment from a distant location, the software functioning is changed as shown in Figures 2.14 and 2.15. The software is divided into two main parts – the server, and the client. Both components communicate via sockets over an IP based Ethernet connection or Internet.

Figure 2.14 shows the algorithm of the server. The `init` procedure reads the program parameters `frequency` and `sampling_time`, and initializes the global

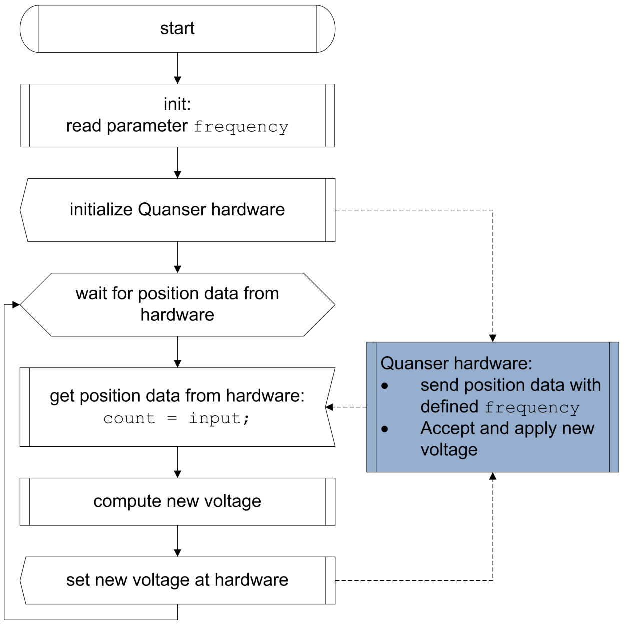

start
init:
read parameter frequency
initialize Quanser hardware
wait for position data from hardware
get position data from hardware:
count = input;
compute new voltage
set new voltage at hardware
Quanser hardware:
• send position data with defined frequency
• Accept and apply new voltage

Figure 2.13: Algorithm for local control of Quanser equipment.

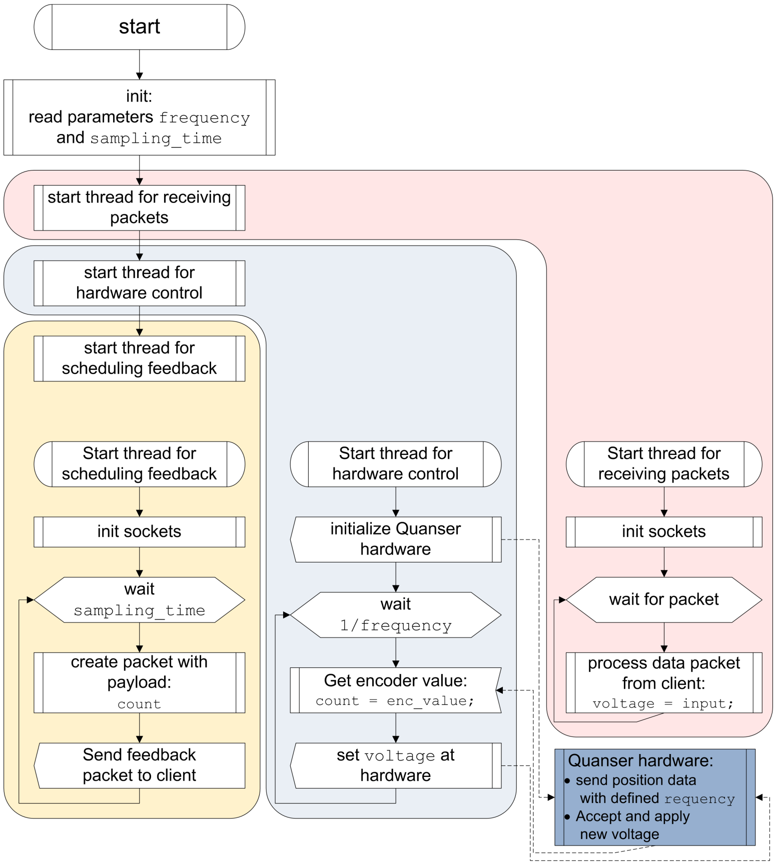

start
init:
read parameters frequency
and sampling_time
start thread for receiving packets
start thread for hardware control
start thread for scheduling feedback
Start thread for scheduling feedback
init sockets
wait sampling_time
create packet with payload: count
Send feedback packet to client
Start thread for hardware control
initialize Quanser hardware
wait 1/frequency
Get encoder value: count = enc_value;
set voltage at hardware
Start thread for receiving packets
init sockets
wait for packet
process data packet from client: voltage = input;
Quanser hardware:
• send position data with defined requency
• Accept and apply new voltage

Figure 2.14: Algorithm for control server of Quanser equipment.

variables `voltage` and `count`. In the next step, three threads are started: for receiving packets, for interfacing the hardware, and for scheduling feedback packets. The thread for receiving packets initializes a socket and waits for incoming packets. As soon as a packet from the client arrives, it is processed, and the payload data is written to the global program variable `voltage`. The thread for scheduling the feedback also starts with initializing a socket. Then, it enters a loop for sending a packet with the current value of `count` as payload each `sampling_time` milliseconds. Thus, packets are sent with a frequency of $\frac{1}{sampling\_time}$. The hardware control thread initializes the hardware module, sends the current values of `voltage` and reads the current encoder values to `count` with a frequency of 1000Hz.

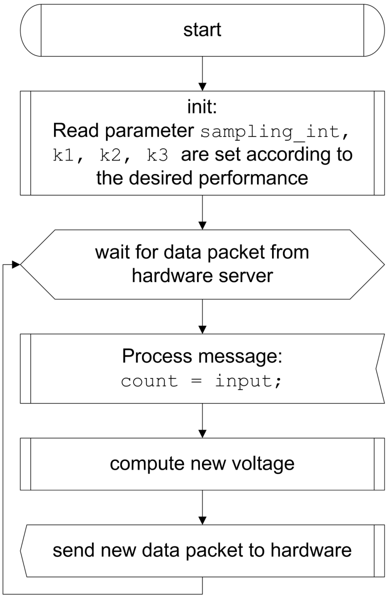

start
init:
Read parameter sampling_int,
k1, k2, k3 are set according to
the desired performance
wait for data packet from
hardware server
Process message:
count = input;
compute new voltage
send new data packet to hardware

Figure 2.15: Algorithm for control client of Quanser equipment.

The functioning of the client is shown in Figure 2.15. The program starts with reading parameter `sampling_int` and parameters for the controller (e.g. `gain`, `k1`, `k2`, or `k3`) depending on the setup. After finishing the `init` procedure, the program waits for packets arriving from the server. These packets have the encoder value of the hardware as payload, and as soon as the packet arrives a new voltage is computed according to the control law which is applied. This new voltage is sent back to the server immediately. Now, the control law as described for the local controller (Equation 2.12) cannot bes used in this networked setup as the communication link over the IP based network induces disturbances which cannot be compensated by the controller.

### 2.3.3 A Discrete-Time System with Time Delay

The above mentioned calculations are dealing with an ideal mathematical model without considering time delays of the system. The experiment which is described in this chapter uses a packet switching based communication channel which induces a time delay, as command packets are transmitted from the control algorithm to the hardware and sensor data is sent back from the hardware to the control algorithm. Now, this system will be investigated further. The data is processed with sampling period *h* and the delay which is induced by the communication channel is $\tau$. For this discrete time system the approach from [88] (p.48) can be used:

**Case 1:** $\tau \leq h$
The time delay is less than the sampling period and the continuous time system is described by

$$\dot{x} = Ax(t) + Bu(t-\tau) \tag{2.13}$$

The behavior of the signal $u(t)$ and the delayed signal is shown in Figure 2.16. For the duration of the sampling period $u(t)$, the delayed signal is constant, whereas it will change between the sampling instants (cf. Figure 2.16). According to the approach in [88], the sampling of the continuous time system from Equation 2.13 will result in

$$x(kh+h) = \Phi x(kh) + \Gamma_0 u(kh) + \Gamma_1 u(kh-h) \tag{2.14}$$

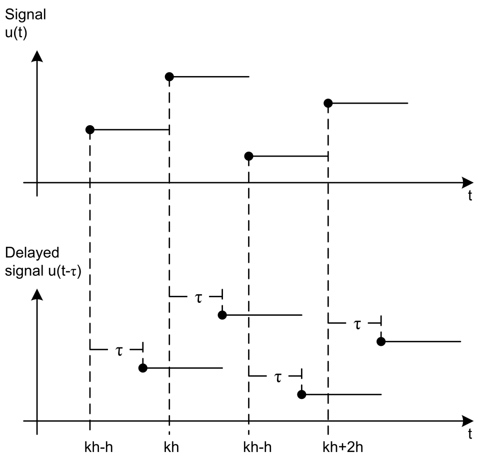


Figure 2.16: Behavior of signal $u(t)$ and the delayed signal (c.f.[88]).

with

$$\Phi = e^{Ah} \tag{2.15}$$

$$\Gamma_0 = \int_0^{h-\tau} a^{As} dsB \tag{2.16}$$

$$\Gamma_1 = e^{A(h-\tau)} \int_0^{\tau} a^{As} dsB \tag{2.17}$$

The state space model can be written as

$$\underbrace{\begin{pmatrix} x(kh+h) \\ u(kh) \end{pmatrix}}_{Z(k+1)} = \underbrace{\begin{bmatrix} \Phi & \Gamma_1 \\ 0 & 0 \end{bmatrix}}_{\tilde{\Phi}} \underbrace{\begin{pmatrix} x(kh) \\ u(kh-h) \end{pmatrix}}_{Z(k)} + \underbrace{\begin{bmatrix} \Gamma_0 \\ I \end{bmatrix}}_{\tilde{\Gamma}} \underbrace{u(kh)}_{u(k)} \tag{2.18}$$

State $u(kh)$ is used to keep track of the input, and $u(kh-h)$ holds the past values of the input. The above equation can be rewritten as

$$Z(k+1) = \tilde{\Phi}Z(k) + \tilde{\Gamma}u(k)$$

Now, $K$ can be designed for

$$u(k) = KZ(k) = (k_1\ k_2\ k_3)\begin{pmatrix} x_1(k) \\ x_2(k) \\ u(k-1) \end{pmatrix} \tag{2.19}$$

so that $(\tilde{\Phi} + \tilde{\Gamma}K)$ is stable.

**Case 2:** $\tau > h$
For time delays $\tau$ longer than the sampling period $h$, the previous analysis must be changed and the modified approach of [88] can be used. With

$$\tau = (d-1)h + \tau';\ 0 < \tau' \le h$$

and $d$ as integer value, the following equation is obtained (c.f. [88]).

$$x(kh+h) = \Phi x(kh) + \Gamma_0 u(kh-dh+h) + \Gamma_1 u(kh-dh)$$

where

$$\Phi = e^{Ah} \tag{2.20}$$

$$\Gamma_0 = \int_0^{h-\tau'} a^{As} ds B \tag{2.21}$$

$$\Gamma_1 = e^{A(h-\tau')} \int_0^{\tau'} a^{As} ds B \tag{2.22}$$

The state-space representation is given by

$$\begin{bmatrix} x(kh+h) \\ u(kh-dh+h) \\ \vdots \\ u(kh-h) \\ u(kh) \end{bmatrix} = \underbrace{\begin{bmatrix} \Phi & \Gamma_1 & \Gamma_0 & \cdots & 0 \\ 0 & 0 & I & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & I \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}}_{\tilde{\Phi}} \begin{bmatrix} x(kh) \\ u(kh-dh) \\ \vdots \\ u(kh-2h) \\ u(kh-h) \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ I \end{bmatrix}}_{\tilde{\Gamma}} u(kh) \tag{2.23}$$

### 2.3.4 Experiment Setup and Evaluation

As shown in the previous section, the system can be described by Equation 2.11, whereas the matrices $A$ and $B$ can be determined according to Equations 2.9 and 2.10. The manual of the Quanser SRV02 hardware provides the values for the constants which are used in these equations (c.f. Table 2.3). Thus, the values for $a_1$, $a_2$, and $b_1$ can be calculated.

| **Symbol** | **Description** | **Nominal SI Value** |
|---|---|---|
| $\eta_g$ | gearbox efficiency | 0.9 |
| $\eta_m$ | motor efficiency | 0.69 |
| $K_t$ | motor-torque constant | 0.00767 |
| $K_g$ | gear ration between motor and wheel | 70 |
| $J_{eq}$ | equivalent moment of inertia at the load | $2.0 \cdot 10^{-3}$ |
| $R_m$ | armature resistance | 2.6 |
| $B_{eq}$ | equivalent viscous damping coefficient | $4.0 \cdot 10^{-3}$ |
| $K_m$ | back-emf constant | 0.00767 |

Table 2.3: Coefficients of the used hardware (from the Qanser SRV02 manual).

$$\begin{aligned}
b_1 &= \eta_g \eta_m K_t K_g = 0.3334149 \\
a_1 &= J_{eq} R_m = 0.0052 \\
a_2 &= B_{eq} R_m + \eta_g \eta_m K_m K_t K_g^2 = 0.0104 + 0.17901045981 = 0.1894
\end{aligned}$$

The values for $a_1$, $a_2$, and $b_1$ are now used for matrices $A$ and $B$ according to Equations 2.9 and 2.10:

$$A = \begin{pmatrix} 0 & 1 \\ 0 & \frac{-a_2}{a_1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & -36.4251 \end{pmatrix}$$

$$B = \begin{pmatrix} 0 \\ 64.1182 \end{pmatrix}$$

As a next step, the communication delays between hardware and control PC are analyzed. Therefore, the measurements of round trip times are evaluated in order to get the average and the maximum delay of the present communication link. This communication delay analysis now allows for an estimation of the delays which occur and which have to be compensated by the system. Depending on the present time delays Equations 2.15, 2.16, 2.17, or Equations 2.20, 2.21, 2.22 are used for the discrete-time matrices. This leads to the discrete-time augmented system according to Equation 2.18 in case the time delay is smaller than the sampling interval ($\tau \leq h$), or Equation 2.23 for the time delay being larger than the sampling time ($\tau > h$). The poles are placed, so that the systems follows the constraints of the hardware in terms of avoiding the generation of voltages being out of range. The design results in the following closed loop system:

$$\begin{aligned} Z(k+1) &= \tilde{\Phi}\, Z(k) + \tilde{\Gamma}\, K\, Z(k) \\ &= (\tilde{\Phi} + \tilde{\Gamma} K)\, Z(k) \end{aligned}$$

In order to achieve the system behavior displayed in Figure 2.17, state feedback matrix $K$ for the present system is chosen:

$$K = \begin{pmatrix} k_1 & k_2 & k_3 \end{pmatrix} = \begin{pmatrix} -0.3660 & 0.0698 & 0.2927 \end{pmatrix}$$

The real world implementation of the above designed controller follows the ar-

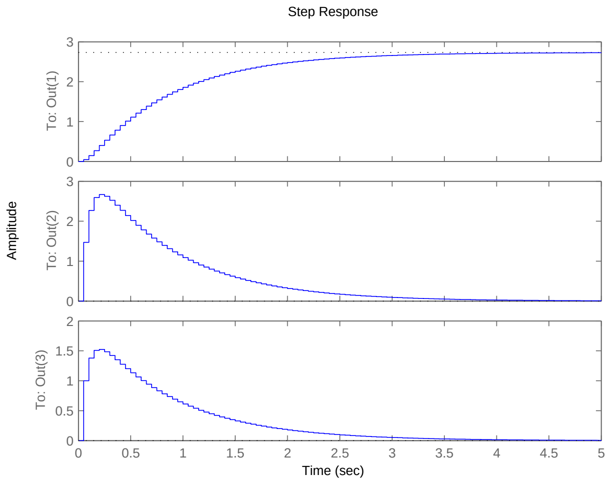


Figure 2.17: Step response of the designed controller.

chitecture displayed in Figures 2.14 and 2.15. The system is running with a sampling rate of 20 Hz, and $\dot{\Theta}$ and $\ddot{\Theta}$ are measured with the same frequency. For the experiment, the system starts in the *zero position*. The desired goal position of the wheel is increased by $45^\circ$ at 6.3 seconds, 9 seconds, and at 11.8 seconds of experiment time. At 14.2 seconds the goal position is reseted to the initial *zero position*. The behavior of this real system controlled via Internet is shown in Figure 2.18. Figure 2.18 shows that the controller for the real hardware system can

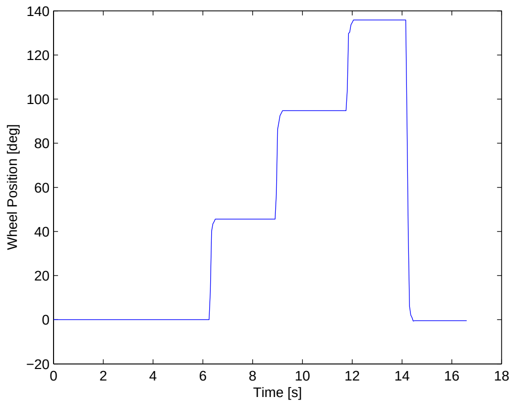


Figure 2.18: Real system behavior when controlled via Internet.

be parametrized so that the system can be controlled via Internet.

## 2.4 Discussion of the Results

This chapter mainly focuses on the design and implementation of a tele-laboratory with real hardware experiments. Starting with a requirement analysis and an analysis of the state of the art several approaches for this type of tele-education units are presented. Early developments in this area used Matlab/Simulink and Labview experiments or connected remote controllable robots, while the approach developed in this work aimed on self-contained learning units with interactively interfaced hardware. In principle, different architectures would be possible to achieve this goal, but of course each approach has its own advantages and disadvantages. Classical client-server approaches, web service architectures, and multi-agent approaches in service oriented architectures are discussed. The objective of the tele-laboratory which was developed in the frame of this work is a modularized large-scale interoperability for different kinds of hardware experiments. Different visualization and data handling and processing technologies can be used. Methods from the area of virtual reality, dynamic web content generation (e.g. PHP, Smarty), .NET, Macromedia Flash, Matlab/Simulink, Quanser WinCon, and also rich Internet applications like Adobe Flash, Java Web-Start, JavaFX, and Microsoft Silverlight provide a large set of functionalities to achieve the goal of interactive tele-learning units with mobile robots. The objective of the first part of this thesis is the development of a remote laboratory which provides tele-experiments with real mobile robot hardware. These experiments should be accessible via the Internet from all over the world, and in addition, also the experiment hardware itself should be places at different countries and on different continents. Therefore, an architecture was developed which allows for an integration of distributed servers connected to hardware which host different learning units while simultaneously providing a comprehensive user management with all the necessary features and modules. Communication between mobile robot and client application is implemented asynchronously, as commands are transmitted to the hardware server and execution takes place on the hardware. Here, also security mechanisms are implemented to prevent the hardware from damage. This design supports Internet communication via low bandwidth links, as well as communication via partially disconnected links. Additionally, the implemented system allows users to resume an interrupted experiment session. Also interactivity is provided together with the feature of being able to resume an experiment without loosing experiment data after communication breaks down completely. This feature turns out to be one of the most important functionalities during our tests where mobile robot hardware was teleoperated from remote locations in India. A summary of the service capability of the implemented remote laboratory *Tele-Experiments with Mobile Robots* is given in Table 2.4. This table shows a comparison of well known and successfully realized tele-laboratories developed in the scientific community in the past 10 years. All of these remote laboratories are designed to be available 24 hours each day. Nevertheless, the *Integrated Remote Laboratory* of the University of Illinois and the laboratory of the *Blekinge Institute of Technology* are online only during the period of the corresponding lectures. Common categories like *Client*, *Type*, *Interaction*, *Location*, *Status*, *Interface*, and *Networking* are used for the comparison according to the classification in [89]. *Client* describes the type of the client software (e.g. web-intereface or Java client) and shows required third party software (e.g. Matlab or Lab View). Of course, all remote laboratories are accessible via the Internet, but nevertheless, different client implementations exist. The majority of the remote laboratories in Table 2.4 use web-interfaces with dynamically created content. In addition, some laboratories like the *Microelectronics WebLab* of the MIT and the *Control Laboratory* of the Oregon State University provide Java applets. For some of the remote laboratories of the comparison (e.g. the *Integrated Remote Laboratory* of the University of Illinois, the *Automatic Control Telelab* of the University of Siena, and the *Remote Robotics and Control Lab* of the University of Maribor) also expensive third party software (e.g. Matlab or Lab View) is required. The *Tele-Experiments with Mobile Robots* of the University of Würzburg provide a Java WebStart based application for the interaction between hardware and user. Category *Type* describes the supported experiment type and distinguishes between remote hardware and remote simulation. Remote hardware means, that real hardware entities can be accessed and teleoperated, whereas remote simulations just provide access to simulated entities which are usually represented by a piece of software. All remote laboratories listed in Table 2.4 support remote hardware experiments. In addition, *NetLab* of the University of South Australia, the *Automatic Control Telelab* of the University of Siena, the *Remote Robotics and Control Lab* of the University of Maribor, the remote laboratory of the *Blekinge Institute of Technology* and the *Tele-Experiments with Mobile Robots* of the University of Würzburg also host remote simulations. The type of *Interaction* can be implemented as batch processing or as interactive. For batch processing, a command set is transferred to the remote hardware and after a successful data transmission, the commands are processed. Interactive means a real interaction between user and hardware is present which is necessary for the desired hands-on experience of the user. The *Microelectronics WebLab* of the MIT, the *Web Shaker* of UC San Diego, the *Automatic Control Telelab* of the University of Siena, and the *Remote Robotics and Control Lab* of the University of Maribor use batch processing. All other remotes experiments of Table 2.4 provide a more interactive interface. Column *Location* distinguishes between local experiments and distributed experiments. A remote laboratory with local experiments hosts one or more experiments which are placed only at one location. Remote experiments which host experiments of different Universities, cities, or countries support distributed experiments. Only two remote laboratories listed in Table 2.4 (*LearNet* and *Tele-Experiments with Mobile Robots* of the University of Würzburg) support distributed experiments, whereas the remote laboratory *Tele-Experiments with Mobile Robots* is additionally designed to host world wide distributed experiments. *Status* indicates whether a remote laboratory is currently not running (offline) or active. Five of the remote laboratories are offline (the *Telegarden* of MIT, the *Control Laboratory* of Oregon State University, the *Integrated Remote Laboratory* of the University of Illinois, the *Automatic Control Telelab* of the University of Siena, and the remote laboratory of the *Blekinge Institute of Technology*). *LearNet* is partially offline, which means that some of the originally hosted experiments are not available anymore. All other listed remote laboratories are currently online and accessible, whereas often a registration is mandatory. Category *Interface* describes elements of the user interface which are available in addition to the sensor data which is common for all the laboratories listed in Table 2.4. Usually, a video stream is provided for all experiments. The *Integrated Remote Laboratory* of the University of Illinois and the *Remote Robotics and Control Lab* of the University of Maribor uses Lab View to display sensor data. In addition to video and sensor data, the *Tele-Experiments with Mobile Robots* of the University of Würzburg also provides a mixed reality interface which allows the use of low bandwidth connections. The last column *Networking* shows features of the remote laboratories with respect to the implemented networking. Here, only the *Integrated Remote Laboratory* of the University of Illinois and *Tele-Experiments with Mobile Robots* of the University of Würzburg support resuming of sessions and the *Tele-Experiments with Mobile Robots* additionally supports low bandwidth and partially disconnected links. Of course, Table 2.4 shows only a selection of all remote labs which were developed during the last 10 years. The tele-laboratories of Table 2.4 are selected according to their implemented features and their high publicity they reached within the scientific community.

In addition to the remote laboratory, it is also shown in Section 2.3 how detailed knowledge about the communication channel allows for the setup of a system which is controlled via the network using approaches for the control of discrete time systems. The designed and implemented tele-laboratory architecture (see Section 2.2) was used and developed during several projects (e.g. "Tele-Experimente mit Mobilen Robotern - Virtuelle Hochschule Bayern" or "International Virtual Laboratory on Mechatronics - EU-ECCP") and is currently used in teaching at the Department of Robotics and Telematics at the University of Würzburg.

Table 2.4: Comparison of Remote Laboratories.

| Name | Client | Type | Interaction | Location | Status | Interface | Networking |
|---|---|---|---|---|---|---|---|
| Microelectronics WebLab (iLab), MIT | Java applet | Remote | Batch | local exp. | Active | Dedicated interface | n/a |
| Telegarden, MIT | Web-Interface | Remote | Interactive | local exp. | Offline | Video | n/a |
| The Distributed Robotics Garden, MIT | | Remote | Interactive | local exp. | Active | Video | n/a |
| Control Laboratory, Oregon State University | Java applet | Remote | Interactive | local exp. | Offline | Video | n/a |
| Web Shaker, UC San Diego | Web-Interface | Remote | Batch | local exp. | Active | Video | n/a |
| Integrated Remote Laboratory, University of Illinois | Web-Interface & Lab-View | Remote | Interactive | local exp. | Offline | Lab-View & video | resuming sessions |
| Automatic Control Telelab, University of Siena | Web-Interface & Matlab | Remote sim. & remote | Batch | local exp. | Offline | Video | n/a |
| Remote Robotics and Control Lab, University of Maribor | Web-Interface & Lab-View | Remote & sim. | Batch | local exp. | Active | Lab-View & video | n/a |
| NetLab, University of South Australia | Web-Interface | Remote & sim. | Interactive | local exp. | Active | Video | n/a |
| Blekinge Institute of Technology, Sweden | Web-Interface | Remote & sim. | Interactive | local exp. | Offline | Video | n/a |
| Telelaboratory of University of Jaume I | Web-Interface | Remote | Interactive | local exp. | Active | Video | n/a |
| LearNet, 8 Universities in Germany | Web-Interface | Remote | Interactive | distrib. exp. | part. offline | Video | n/a |
| Tele-Experiments with Mobile Robots, University of Würzburg | Web-based & Java WebStart | Remote & sim. | Interactive | int. distrib. exp. | Active | Video and Mixed Reality | Adaptive to bandwidth & resuming sessions |

# 3 Ad-Hoc Networks of Mobile Robots

Ad-hoc capabilities are key features of modern mobile robot networks, as a maximum of flexibility in terms of the supported network topology is achieved. This chapter starts with an overview of enabling technologies like wireless communications, wireless LAN and ad-hoc routing protocols. The theoretical focus is set on a detailed evaluation of four well known ad-hoc routing protocols (AODV, DSR, OLSR, and BATMAN) with respect to the use in networks of mobile robots. The obtained results are used to improve the performance of mobile robot teleoperation and coordination in both application scenarios given at the end of this chapter.

## 3.1 Enabling Technologies

### 3.1.1 Wireless Communication

Nowadays, wireless communications became an important part of daily life. Mobile phones are widely spread and provide connectivity for voice communication and data communication via the world wide web. Also wireless networks of PCs or Notebooks are quite common, and wireless networks based on IEEE 802.11 wireless LAN (WLAN) can be set up everywhere by nearly everyone. This chapter starts with a short introduction of several application scenarios which are possible with nowadays available WLAN hardware and in the second part, important aspects of the IEEE 802.11 standard for wireless communications are described. For many businessmen WLAN became a indispensable technology for daily life. Often WLAN access is available at airports, hotels, and train stations, and sometimes even in trains and airplanes. Now, it is easily possible to combine access to the Internet, to e-mails, or to business data together with a mobility aspect. Earlier this ideas were summarized by the concept of a "mobile office". Of course, WLAN is only one aspect of mobile wireless broadband communication. Fur-

ther aspects with respect to 3G communication networks and Universal Mobile Telecommunications System (UMTS) are given in Chapter 4.
WLAN is also a technology which allows for building a fast and inexpensive infrastructure in case networks based on wired infrastructure are not suitable or not available. Thus, often PCs inside office buildings are connected via WLAN or even buildings itself can be connected over a distance of up to some kilometers via a dedicated WLAN using directional antennas.
As setting up WLAN networks is fast, inexpensive, and easy, it might be also a choice for establishing voice and data communication networks in case of emergency situations. Natural disasters like earth quakes or tsunamis usually destroy the present communication infrastructure like the base stations for mobile phones or even the telephone cables, and rescue teams must set up their own communication network. Currently, rescue teams can easily setup voice communication via their standard radio devices, but broadband data communication cannot be handled with current radio equipment of rescue teams. Here, WLAN networks might jump in and fill the gap.
A new direction of the application of WLAN networks is the area of car-to-car (Car 2 Car) communication. Here, the focus is set on three objectives: safety applications, traffic engineering, and entertainment, whereas each area has its own requirements and priorities for the communication channel. Safety applications need low latencies, low bandwidth, and data loss has to be avoided. Traffic engineering aims on distant distribution of information and delay tolerant applications while supporting broadcast communication with a limited data loss. In the area of entertainment, a large amount of data should be transferred by audio and video streams which causes the highest need for bandwidth and requires quality of service for the data transfer.
The European Union and national science funding institutions have also identified the high potential of this research area, and currently several important related research projects are running:

- "FleetNet"[1] (2000-2003) funded by the BMBF;
- "CarTALK" (2001-2004) funded by the EU (based on FleetNET);
- "NoW - Network on Wheels"[2] (2004-2008) funded by BMBF;

[1] http://www.et2.tu-harburg.de/fleetnet/ (15.10.2008)
[2] http://www.network-on-wheels.de/ (09.02.2010)

- "AKTIV - Adaptive und kooperative Technologien für den intelligenten Verkehr"[3] (2006-2010) funded by BMBF;
- "WILLWARN" (2005-2007)[4] funded by the EU (a sub-project of PReVENT[5]);

In 2007, the Rhine-Main area was selected as test area for the pilot project "SIM-TD - Sichere intelligente Mobilität - Testfeld Deutschland"[6]. An EU-wide initiative, the "Car 2 Car Communication Consortium" [7] has its main focus on establishing an European industry standard for Car 2 Car communication based on WLAN. All the presented projects are using IP communication based on sub-standards of IEEE 802.11 (802.11 a/b/g/p) or UMTS (for details on UMTS please refer to Chapter 4).

A key issue of efficient wireless communication is an effective access on the transmission media. Media Access Control (MAC) is a collection of mechanisms which are necessary to coordinate access of users on a communication medium. With respect to the ISO/OSI model, these mechanisms are located at the data link layer (layer 2) and can be sub-divided into data link control (DLC) and logical link control (LLC). For wireless networks, the "Carrier Sense Multiple Access with Collision Detection" (CSMA/CD) mechanism which is well known from IEEE 802.3 based networks (Ethernet) cannot be used. The signal quality of wireless networks depends on many different influences (e.g. the distance between sender and receiver, obstacles, disturbances). Thus, it might happen that a sender detects a free medium and starts sending, but the reception of the signal is disturbed. Also the collision detection for the sending node is not so easy as it is in wired networks. With respect to radio networks, the MAC mechanisms must also be able to handle the typical problems like hidden nodes, exposed nodes, and the near-far problem.

The hidden node situation is shown in Figure 3.1a. In this situation, node A and node B, as well as node B and node C are in communication range. Direct communication between node A and C is not possible. In case node A has data which

[3]http://www.aktiv-online.org/ (09.02.2010)

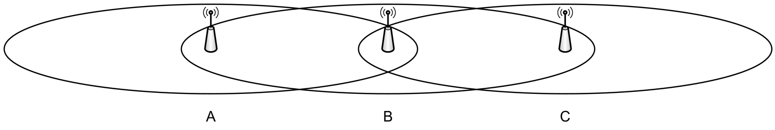
[4]http://www.prevent-ip.org/en/prevent_subprojects/safe_speed_and_safe_following/willwarn/ (09.02.2010)

[5]http://www.prevent-ip.org/en/home.htm (09.02.2010)

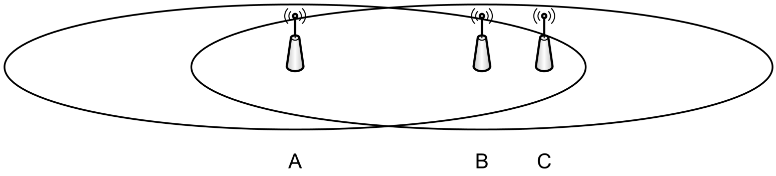
[6]http://www.cvisproject.org/en/links/sim-td.htm (09.02.2010)

[7]http://www.car-to-car.org/ (09.02.2010)

should be transmitted to node B, A will listen on the medium and as soon it is free, node A starts sending data. At the same time, node C cannot recognize the data exchange from node A to node B and can also send data to node B which will disturb and interrupt both transmissions (from A to B and from C to B) without any possibility for node A and for node C of sensing this failure.



(a) Hidden node and exposed node situation.



(b) Near far problem.

Figure 3.1: Typical situation which might cause problems for wireless communication.

An exposed node situation can also be explained by Figure 3.1a. It exists in case node B sends data to node A and node C also wants to send data to an additional node which is neither in communication range to node B, nor in communication range to node A. In this case node C will delay sending data as long as the medium is busy due to an active node B. As node A is not within the range of node C, this delay is not necessary as simultaneous sending of node B and node C cannot disturb the data transmission to the corresponding receiving nodes.

The near-far problem is shown in Figure 3.1b. It happens in case receiver C is closer to transmitter B than to transmitter A. If node A and node B start transmitting data to node C simultaneously at an equal sending power, the receiver gets a much better signal from the closer transmitter (node B) due to the signal of one transmitter being the noise of the other transmitter's signal. As the near-far problem is more relevant for systems based on the *code division multiple access* (CDMA) mechanism, more details on this issue are given in Chapter 4.

### 3.1.2 Wireless Local Area Networks (WLAN)

#### IEEE 802.11 Standard

The IEEE 802.11 standard [90] describes a currently world wide used family of radio networks for which a large variety of products exist also in the consumer market. IEEE 802.11 standardizes a physical layer (ISO/OSI layer 1) and a media access control layer (ISO/OSI layer 2) which are necessary to cope with the special requirements of wireless communication which are already mentioned in the previous section. For higher layers the the common interface of the IEEE 802 family is used in order to allow for a seamless interoperability. Objectives of this standard are a robust wireless data communication and a world wide use with respect to the used radio frequencies. In general, the proposed system architecture supports two different modes: infrastructure based and ad-hoc. In the infrastructure mode an access point is used to coordinate the media access of each associated station. The ad-hoc mode has no central unit which coordinates the media access, and additionally multi-hop communication should be supported. For detailed explanation of all IEEE 802.11 functionalities please refer to [28].

#### Media Access Control for IEEE 802.11

As already mentioned in Section 3.1.1, media access control is one key issue with respect to efficient wireless data transfer and IEEE 802.11 specifies three mechanisms to cope this challenge. The following paragraphs give a short summary of the most important functionalities of the IEEE 802.11 MAC-layer in order to get a clear view of the later presented problems which are solved in the frame of this work in order to provide seamless operating networks of mobile robot based on wireless LAN. An extensive summary of the IEEE 802.11 standard is given in [28] and all details are specified in [90].
A *Distributed Coordination Function* (DCF) provides a compulsory mechanism based on *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA) which has to be supported. An optional mechanism to solve the problem of hidden nodes can also be implemented. The *Point Coordination Function* (PCF) provides an optional mechanism for a contention-free centralized data transfer. For all mechanisms the following parameters are relevant:

- DCF Inter-Frame Spacing (DIFS): This parameter represents the lowest priority for media access. The duration $t_{DIFS}$ is defined as $t_{DIFS} = t_{SIFS} +$

$2 \cdot t_{st}$ with $t_{SIFS}$ being the duration of SIFS and $t_{st}$ being the duration of the slot time. DIFS is used for the asynchronous data transfer.

- PCF Inter-Frame Spacing (PIFS): The PIFS interval is used by access points and specifies the duration for which an access point has to wait until it can poll stations. As the duration of PIFS is smaller than the duration of DIFS, the priority of PIFS is higher than the priority of DIFS, and thus, an access point can start its activity earlier than a station. The duration of $t_{PIFS}$ is defined as $t_{PIFS} = t_{SIFS} + t_{st}$.

- Short Inter-Frame Spacing (SIFS): SIFS is the shortest waiting time and thus gives the highest priority for media access. Usually control packets are sent with this priority in order to ensure the network maintenance while blocking traffic of lower priority. For available WLAN hardware, the duration of SIFS is 10 $\mu$s.

- slot time: The duration of the slot time is defined according to the signal propagation delay between sender and receiver, the processing time of the radio electronics, and other physical parameters. Usually, the a slot has a duration of 20 $\mu$s for WLAN.

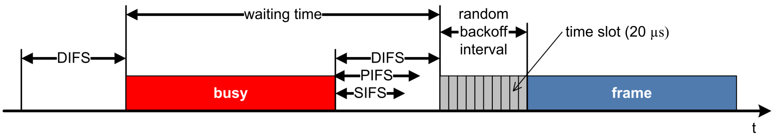


Figure 3.2: Timing of IEEE 802.11 CSMA/CA (according to [28]).

The interactions of all of the above mentioned parameters are shown in Figure 3.2. As the implementation of each IEEE 802.11 media access is based on the "carrier sense multiple access with collision avoidance" mechanism (CSMA/CA), it will be shortly described in the following paragraphs. CSMA/CA is a decentralized media access mechanism, and the key feature for avoiding collisions is a backoff algorithm. In case a device has data available to be sent, it has to wait until the media is not busy for the duration of $t_{DIFS}$ before the data transmission is started. If the media is active in case a device wants to send data, it has to wait for $t_{DIFS}$, and afterwards, the contention phase (contention window) is entered. For the contention phase, the device selects a random backoff time, which corresponds to a random time slot in the contention phase and the overall waiting time is calculated. Now, the device is waiting and for each elapsed time slot, the corresponding time is subtracted from the calculated waiting time. As soon as the waiting time is equal to zero, the device can start the sending procedure. In case the media becomes busy while the device is waiting, it stops decrementing the waiting time until the media is free for the duration of $t_{DIFS}$. Then, the old waiting time is again decremented by the duration of elapsing time slots. As described above, the device starts sending as soon as a waiting time equal to zero is reached. As this mechanism respects the already elapsed waiting time of a device, a kind of fairness is introduced. Nevertheless, the described basic mechanism has some drawbacks in case of a very low or a very high load. In order to reduce the probability of different devices selecting the same backoff interval (probability depends on the length of the contention window) a mechanism called *Exponential Backoff* is introduced. First, a media access procedure starts with a small size of the contention window $CW_{min} = 7$. As soon as a collision is detected, the contention window size is doubled until the maximum of $CW_{max} = 255$ is reached. To solve the problem of hidden nodes, the RTS/CTS extension is implemented

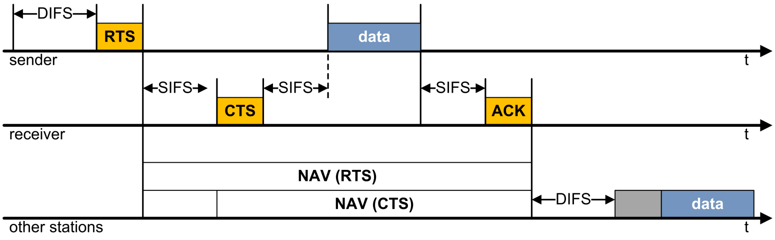


Figure 3.3: RTS/CTS to solve hidden node problem (according to [28]).

(cf. Figure 3.3). After a device is allowed to send data according to the above described CSMA/CA mechanism, it does not start sending data immediately. Instead, a *Request-To-Send* (RTS) packet is sent. This packet contains the receiver and the duration including the acknowledgment of the upcoming data transfer. Each station which receives this packet stores the planned data transfer duration in its *Net Allocation Vector* (NAV) which represents the earliest time the media

will be free for this station. The receiver of the planned data transfer waits for the duration of $t_{SIFS}$ and afterwards, it sends a *Clear-To-Send* (CTS) packet back to the sender. The CTS packet also contains the duration of the planned data transfer, and each station which receives this packet adjusts its NAV. The CTS packet can also be received by stations which are hidden for the sender but in range of the receiver. After the sender received the CTS packet it waits for $t_{SIFS}$ and starts the data transfer. Finally, after the receiver has all data it sends an acknowledgment to the sender the data transfer is completed. The RTS/CTS extension reduces the probability of collisions as collisions can only occur in the beginning of a sending period in case other stations also send RTS packets.
For more details on additional extensions and functionalities of the IEEE 802.11 standard please refer to [90].
The following subsections give a short overview of the most common sub-standards of the IEEE 802.11 family which are currently usable, and for which commercial off-the-shelf hardware is available.

#### IEEE 802.11a Standard

The IEEE 802.11a standard provides data rates up to 54 Mbit/s and uses the 5 GHz band. It supports *Dynamic Frequency Selection* (DFS) and *Transmit Power Control* (TPC) [90] which is required for the used sending power and the sending schemes. Depending on the regulation of different countries, different frequency ranges of the 5 GHz band are coupled to a maximum allowed sending power.

#### IEEE 802.11b Standard

The IEEE 802.11 standard is extended by IEEE 802.11b to provide a "higher-speed physical extension" in the 2.4 GHz frequency band. Basically, this extension is the definition of a new physical layer, while the already above presented MAC mechanisms of IEEE 802.11 are retained. IEEE 802.11b supports data rates of 11, 5.5, 2, or 1 MBit/s with a maximum usable payload bandwidth of about 6 MBit/s. More details are given in [91] and [90].

#### IEEE 802.11g Standard

IEEE 802.11g supports data rates higher than 20 MBit/s and uses also the 2.4 GHz frequency band. These higher data rates compared to IEEE 802.11b while using the same frequency band are achieved by new modulation schemes, improved error correction mechanisms, and orthogonal frequency division multiplex (OFDM) (for more details on OFDM please refer to [92]). IEEE 802.11g is also backwards compatible to IEEE 802.11b. Nowadays, available IEEE 802.11g products support data rates of 54 MBit/s.

Within the IEEE 802.11 standard much more extensions exist, and some of them are still in the development phase. For a summary of some sub-standards please refer to [28], and detailed information on all currently available sub standards is given in [90].

### 3.1.3 Wireless Mobile Ad-Hoc Networks and Ad-Hoc Routing

During the last years the topic of wireless mobile ad-hoc networks (MANETs) became very popular and many routing algorithms for this type of network were developed. MANETs consist of several devices/nodes which can establish radio communication between each other at any time, and a device can act as communication relay to forward data if necessary to establish a communication link to distant nodes. In this case, ad-hoc routing mechanisms are used to provide connectivity on a logical layer and data is forwarded from one device to the next network node until the destination is reached. Ad-hoc networks are characterized by a decentralized organization and their ability for autonomous configuration and autonomous setup. In addition, also changes in the network topology due to node mobility, node failure, temporarily non-operational nodes, or new nodes joining the network are supported. The first network of this type was the ALOHA-Net [93] which was set up in the seventies (funded by DARPA) to create a computer network using low-cost amateur radio equipment. Nowadays, the new generations of MANETs are used to interconnect user devices like PDAs, mobile phones, or notebooks, and for establishing Internet connections of mobile devices. The decentralized architecture of MANETs allow for their application in case the loss of a network node should not be a danger for the entire network. Also scenarios which require a variable number of participating network nodes and an autonomous configuration are supported. Due to the easy and fast setup of MANETs a reliable communication infrastructure can be established very quickly. With respect to their application wireless ad-hoc networks can further be classified in wireless mesh networks, wireless sensor networks, and mobile ad-hoc networks.

A wireless mesh network usually consists of several radio nodes where all nodes are static without any mobility. In addition, redundant communication links are available within the network which keeps the network operating in case single nodes are temporarily not working.

Wireless sensor networks are set up by distributed autonomous sensor nodes which monitor its environment. Usually, each sensor node is equipped with a radio transceiver supporting energy efficient communication (e.g. ZigBee based on IEEE 802.15.4 or Bluetooth/IEEE 802.15.1). Common applications of wireless sensor networks are area monitoring, environmental monitoring, home automation, healthcare applications, and traffic monitoring and control.

A very common and wide spread technology for the realization of mobile ad-hoc networks is IEEE 802.11. Latest developments in the application of mobile ad-hoc networks aimed towards Vehicular Ad-Hoc Networks (VANET) [94] and Intelligent Vehicular Ad-Hoc Networks [95]. Here, an effective communication between vehicles with dynamic mobility and also roadside equipment is established in order to provide data exchange like traffic information, danger warnings or even media communication or tracking services [96]. This decentralized design of these wireless ad-hoc networks makes them suitable for many different applications. The minimal configuration effort combined with adaptive ad-hoc routing protocols, which enable mobility and a flexible deployment of nodes, allow these wireless ad-hoc networks for providing connectivity and communication even in situations when no pre-installed communication infrastructure is available anymore (e.g. disaster and emergency situations).

The flexibility and the support of highly dynamic network structures which are mentioned in the previous paragraphs are enabled by complex ad-hoc routing mechanisms. As there is no centralized route maintenance entity inside the entire network, each single node has to discover the relevant paths to its required destinations in a relatively short time. The nodes also have to detect the changes in the network topology and they have to react autonomously on these changes. Of course, these functionalities should be implemented with a minimum usage of data traffic for route maintenance. Here, the traditional routing mechanisms which are known from the Internet are not useful as they are based on central route maintenance entities inside the network and the above mentioned mobility of nodes is not supported. Ad-hoc routing protocols can further be classified by means of their functionality:

**Position based** or **geographical routing**: This protocol type determines the most suitable path between source and destination based on position information. Thereby, position data can be obtained from localization systems like GPS. Examples for this protocol type are the Location-Aided Routing protocol (LAR) [97] or the Blind Geographic Routing (BGR) [98].

**Topology based routing**: Here, the information about the network topology is determined by sending and receiving special data packets. Subclasses are proactive and reactive ad-hoc routing protocols.

**Proactive routing**: Topology discovery is done periodically which allows for a fast detection of topology changes. The information gathered by a network node is stored in a local routing table. But as network maintenance data is always transmitted - even if no payload data should be transmitted - a certain load of the network is always present. Besides the advantages of a fast reaction on topology changes due to already discovered routes when data is ready to be sent, also some disadvantages exist. The continuously transmitted traffic for network maintenance might have negative effects on the usable bandwidth or the energy consumption of the nodes. An example of a well known pro-active routing protocol is Optimized Link State Routing (OLSR) [99].

**Reactive routing**: In contrast to the above described pro-active mechanisms, the reactive routing protocols discover required routes only on-demand. Thus, the data traffic for route maintenance is reduced to a minimum. Of course, sending payload data can only be started after the route discovery to the destination node is completed which will introduce some small delays. Well known reactive ad-hoc routing protocols are Ad-hoc On-demand Distance Vector (AODV) [100] and Dynamic Source Routing (DSR) [101].

**Hybrid approaches**: Hybrid ad-hoc routing protocols try to combine the advantages of reactive and pro-active protocols. Examples for this type are Hybrid Routing Protocol for Large Scale Mobile Ad Hoc Networks with Mobile Backbones (HRPLS) [102] or Temporally-Ordered Routing Algorithm routing protocol (TORA) [103].

There are some more subgroups like power aware routing protocols, multicast routing protocols, geographical multicast protocols, and some more protocols which are not classified. All in all, more than 80 ad-hoc routing protocols were developed during the last years with a lot of different objectives and application areas, and thus, also different strengths and weaknesses. The following sections will briefly summarizes four popular ad-hoc routing protocols - AODV, DSR, OLSR, and BATMAN - which are further used in this work for routing in ad-hoc networks of mobile robots. For all of these protocols, implementations for real world use are existing as a stable running version.

### Ad-hoc On-demand Distance Vector (AODV)

The AODV routing protocol [104] [100] [105] is a reactive routing protocol and searches for a route on-demand. In case a certain node is part of an active route, *Hello* messages are used to obtain the route status. These *Hello* messages are broadcasted periodically to all neighbors. If a neighbor does not send a *Hello* message within a specified time a link loss is detected and the node is deleted from the routing table. In addition, a *Route Error* message (RRER) is generated. To discover a route to an unknown destination, a *Route Request* (RREQ) message is broadcasted. Each intermediate node which is not the destination and without a route to the destination receiving a RREQ broadcasts this RREQ further. In case the RREQ is received more than once, only the first reception will result in a broadcast. To avoid uncontrolled dissemination of RREQ messages, each RREQ has a certain time to live (TTL) after which it is discarded. When the destination receives a RREQ message, a *Route Reply* (RREP) message is generated and sent back to the source in unicast hop by hop fashion along the route which was determined by the RREQ message. After generating a RREP message, the RREQ message is discarded at this node. As the RREP propagates, each intermediate node creates a route to the destination. After the source receives the RREP, it records the route to the destination and begins sending data. In case the source receives multiple RREPs, the route with the shortest hop count is chosen. The status of each route is maintained in the local routing table and timers are used to determine link failures which will result in the creation of *Route Error* messages (RERR). More detailed information on AODV is given in [100].
In the referred test scenarios of this work, AODV-UU version 0.9.5 from Uppsala University (Sweden) is used [8]. Table 3.1 shows the most important parameters of AODV. For some of them, the names are self-explanatory, and sometimes some fixed relations between some parameter values should be met. Usually, AODV

[8]http://core.it.uu.se/core/index.php/AODV-UU (09.02.2010)

Table 3.1: Parameters for AODV (according to [100]).

| Parameter Name: | Default Value |
|---|---|
| ACTIVE_ROUTE_TIMEOUT | 3000 ms |
| ALLOWED_HELLO_LOSS | 2 |
| BLACKLIST_TIMEOUT | RREQ_RETRIES · NET_TRAVERSAL_TIME |
| DELETE_PERIOD | see description |
| HELLO_INTERVAL | 1000 ms (hard coded) |
| LOCAL_ADD_TTL | 2 |
| MAX_REPAIR_TTL | 0.3 * NET_DIAMETER |
| MIN_REPAIR_TTL | see description |
| MY_ROUTE_TIMEOUT | 2 * ACTIVE_ROUTE_TIMEOUT |
| NET_DIAMETER | 35 |
| NET_TRAVERSAL_TIME | 2 * NODE_TRAVERSAL_TIME · NET_DIAMETER |
| NEXT_HOP_WAIT | NODE_TRAVERSAL_TIME + 10 |
| NODE_TRAVERSAL_TIME | 40 ms |
| PATH_DISCOVERY_TIME | 2 · NET_TRAVERSAL_TIME |
| RERR_RATELIMIT | 10 |
| RING_TRAVERSAL_TIME | 2 · NODE_TRAVERSAL_TIME · (TTL_VALUE + TIMEOUT_BUFFER) |
| RREQ_RETRIES | 2 |
| RREQ_RATELIMIT | 10 |
| TIMEOUT_BUFFER | 2 |
| TTL_START | 1 |
| TTL_INCREMENT | 2 |
| TTL_THRESHOLD | 7 |
| TTL_VALUE | see description |

is based on sending *Hello* messages with a given rate determined by parameter HELLO_INTERVAL. This parameter is important for the performance of the protocol, but unfortunately, it is hard coded in the protocol code. In case the code is changed in order to allow different values for HELLO_INTERVAL, the current value of HELLO_INTERVAL should be advertised throughout the network by appending a HELLO_INTERVAL extension to the RREP message. The value of MIN_REPAIR_TTL should be equal to the last known hop count to the destination. In case of using *Hello* messages, it is mandatory that the value for ACTIVE_ROUTE_TIMEOUT is greater than ALLOWED_HELLO_LOSS · HELLO_INTERVAL. ACTIVE_ROUTE_TIMEOUT should be greater than 10000 ms when link layer feedback is used for detecting link failures. The TTL_VALUE corresponds to the value of the TTL field of the IP header during the expanded ring search. The value for TIMEOUT_BUFFER is configurable in order to adjust the timeout probability for delayed RREP packets. With TIMEOUT_BUFFER=0, the functionality of this buffer is deactivated. DELETE_PERIOD provides an upper bound for the time for which node A can have neighbor node B as active next hop for a destination node D, while node B has already invalidated its route to node D. After DELETE_PERIOD is over, node B can delete the already invalidated route to D. This parameter is closely correlated to the underlying link layer, and in case HELLO messages are used, the value of DELETE_PERIOD has to be greater than or equal to ALLOWED_HELLO_LOSS · HELLO_INTERVAL. In case of using link layer feedback, DELETE_PERIOD has to be greater than or equal to ACTIVE_ROUTE_TIMEOUT. Parameter NET_DIAMETER gives the maximum number of hops between two nodes of the network. NODE_TRAVERSAL_TIME gives an estimation of the average one-hop traversal time of packets, including queuing and processing delays. The AODV documentation also propose TTL_START = 2 in case of using *Hello* messages and the value for BLACKLIST_TIMEOUT should be increased if an expanded ring search is used. According to the specification, a value of BLACKLIST_TIMEOUT = ((TTL_THRESHOLD - _TTL_START) / TTL_INCREMENT) + 1 + RREQ_RETRIES · NET_TRAVERSAL_TIME is recommended, in order to allow for additional route discovery attempts.

### Dynamic Source Routing (DSR)

DSR is also a reactive ad-hoc routing protocol which works similar to AODV without using *Hello* messages for route maintenance. However, it uses source routing [106]. DSR allows the network to be completely self-organizing and self-configuring, without the need of any existing network infrastructure or administration. DSR does not use any periodic routing advertisement, link status sensing, or neighbor detection packets, and does not rely on these functions from any underlying protocols in the network. DSR is composed of two main mechanisms that work together to allow the discovery and maintenance of source routes in the ad-hoc network. In case source node (S) wants to send data to an unknown destination host (D), S initiates the Route Discovery mechanism. S broadcasts a Route Request message which identifies the source and destination of the Route Discovery to all neighbors. A Route Request also contains a record listing the address of each intermediate node which was forwarding this particular copy of the Route Request. A node which receives this Route Request without being the destination looks up for a source route to the requested destination in its route cache. Without any source route present in its own route cache, the node appends its own address to the route record and broadcasts the Route Request message. In case this request message was received more than once, it is simply discarded. As soon as the Route Request message arrives at the desired destination D, a Route Reply message to S is created which contains an accumulated route record of the Route Request. After S receives this Route Reply, it caches the corresponding route in its route cache and S is ready to transmit data. Of course, there exist mechanisms to omit flooding of the network with Route Requests. A hop limit was introduced and every time a Route Request is forwarded, the hop limit is decremented by one. As soon as it reaches zero, the request is discarded. Also mechanisms for avoiding infinite recursion of Route Discoveries are implemented. A more detailed description of this protocol is given in [107] [101]. For the presented work DSR-UU version 0.2 from Uppsala Univeristy (Sweden) [9] is used. Table 3.2 shows the parameters of the DSR protocol. DISCOVERY_HOP_LIMIT is a counter which is sent in a route discovery packet and it is decremented each time a node is passed. As soon as it reaches 0, the packet will be discarded. When a node receives a route request, the an-

[9]http://core.it.uu.se/core/index.php/DSR-UU (09.02.2010)

Table 3.2: Parameters for DSR (according to [101]).

| Parameter Name: | Default Value |
|---|---|
| DISCOVERY_HOP_LIMIT | 255 hops |
| BROADCAST_JITTER | 10 ms |
| ROUTE_CACHE_TIMEOUT | 300 s |
| SEND_BUFFER_TIMEOUT | 30 s |
| REQUEST_TABLE_SIZE | 64 nodes |
| REQUEST_TABLE_IDS | 16 identifiers |
| MAX_REQUEST_REXMT | 16 retransmissions |
| MAX_REQUEST_PERIOD | 10 s |
| REQUEST_PERIOD | 500 ms |
| NONPROP_REQUEST_TIMEOUT | 30 ms |
| REXMT_BUFFER_SIZE | 50 packets |
| MAINT_HOLDOFF_TIME | 250 ms |
| MAX_MAINT_REXMT | 2 retransmissions |
| TRY_PASSIVE_ACKS | 1 |
| PASSIVE_ACK_TIMEOUT | 100 ms |
| GRAT_REPLY_HOLDOFF | 1 s |
| MAX_SALVAGE_COUNT | 15 |

swer, a route reply message (RREP), is delayed for a random time duration of a length between 0 and BROADCAST_JITTER. ROUTE_CACHE_TIMEOUT limits the time of storing information of unused routes in the route cache and if a route is not used for this period, the corresponding route entry is deleted in the route cache. SEND_BUFFER limits the buffer queue size for packets which could not be sent as the route is not known yet. As soon as a packet exceeds its SEND_BUFFER_TIMEOUT it is deleted in the buffer. Each node stores own route requests (RREQ) and forwarded RREQs of other nodes in its request table. The size of the request table is given by parameter REQUEST_TABLE_SIZE. Additionally, a node also stores an identifier and a target address of the most recent RREQs in a cache of size REQUEST_TABLE_IDS. In case of unsuccessful RREQs, a node will send a new RREQ after REQUEST_PERIOD. According to [101] it is recommended that the time between route discovery attempts for the same destination is doubled until MAX_REQUEST_REXMT is reached. A request is discarded after reaching MAX_REQUEST_REXMT. In case neighbor nodes do not propagate a request and thus, no replies will be received. A request is marked as unsuccessful after timeout NONPROP_REQUEST_TIMEOUT is exceeded. While packets are forwarded, each node has to confirm the reachability of the next hop node at least once during interval MAINT_HOLDOFF_TIME. In case of not receiving confirmation, a number of MAX_MAINT_REXMT acknowledgment requests is sent an if these are all unsuccessful, the link to the next hop is considered broken. These confirmations can be sent for more packets at once, and therefore not so frequently. Parameter REXMT_BUFFER_SIZE determines the maximum number of unconfirmed packets which are allowed. In case no confirmation is received within MAINT_HOLDOFF_TIME, all unconfirmed packets have to be considered as being not sent. Each time a packet is forwarded, the node sends this packet without requesting a network layer acknowledgment and expects a passive acknowledgment within PASSIVE_ACK_TIMEOUT. If no passive acknowledgment is received within this timeout, the packet is retransmitted TRY_PASSIVE_ACKS times without requesting network layer acknowledgments. If all of these transmissions do not result in a reply, a network layer feedback is requested for all remaining transmission attempts of this packet. It can happen that an intermediate node which is forwarding a packet has information about a shorter route to the destination node. Then, a gratuitous route reply is transmitted to the sending source node of the packet. A source node which receives this gratuitous route reply stores it to its gratuitous route reply table until timeout GRAT_REPLY_HOLDOFF is reached. It can also happen that a node notices that a source route is not valid any more but another route to the destination is known. Thus, the node can salvage the packet but this is quite expensive for the network. These costs are limited by MAX_SALVAGE_COUNT and when this threshold is exceeded, the source has to discover a new route to the destination.

### Optimized Link State routing (OLSR)

OLSR is a table-driven, pro-active routing protocol for mobile ad-hoc networks. It uses hop-by-hop routing – each node uses its local information to route packets. OLSR minimizes the overhead from flooding of control traffic by using only selected nodes – called Multipoint Relays (MPR) – to retransmit control messages. Each node in the network selects a set of nodes in its neighborhood which may retransmit its messages. This set of selected neighbor nodes is called the MPR set of that node. The neighbors of node N which are not in its MPR set, receive and process broadcast messages but will not retransmit broadcast messages received from node N. The MPR set is selected such that it covers all 2-hop nodes. That means every node in the 2-hop neighborhood of N must have a link to the MPRs of N. OLSR continuously maintains routes to all destinations in the network and it is suitable for a large set of nodes communicating with each other.
To distribute link and neighborhood information, *HELLO* messages are exchanged periodically. These messages are also used for link sensing and for checking the connectivity. Thus, the network topology is discovered and disseminated through the network, which allows the route calculation. More details on OLSR are given in [99]. The referred scenario tests of the present work are performed with OLSR version 0.5.3 [10]. The OLSR protocol uses many configuration parameters and Table 3.3 lists only the most important and relevant parameters for the analysis. The following paragraph will give a short explanation of these parameters. Parameter HELLO_INTERVAL specifies the rate for sending *HELLO* messages which are used for gathering neighbor information, link states, and link quality. The update frequency of node and link information is defined by REFRESH_INTERVAL. In case of losing the connection to a neighbor

[10] http://www.olsr.org/ (09.02.2010)

Table 3.3: Parameters for OLSR (sccording to [99]).

| Parameter Name: | Default Value |
|---|---|
| HELLO_INTERVAL | 2 s |
| HELLO_VALIDITY | 6 s |
| REFRESH_INTERVAL | 2 s |
| TC_INTERVAL | 5 s |
| TC_VALIDITY | 15 s |
| MID_INTERVAL | TC_INTERVAL |
| HNA_INTERVAL | TC_INTERVAL |
| NEIGHBOR_HOLD_TIME | 3 · REFRESH_INTERVAL |
| TOP_HOLD_TIME | 3 · TC_INTERVAL |
| DUP_HOLD_TIME | 30 s |
| MID_HOLD_TIME | 3 · MID_INTERVAL |
| HNA_HOLD_TIME | 3 · HNA_INTERVAL |
| HYST_THRESHOLD_HIGH | 0.8 |
| HYST_THRESHOLD_LOW | 0.3 |
| HYST_SCALING | 0.5 |
| TC_REDUNDANCY | 0 |
| MPR_COVERAGE | 1 |
| MAXJITTER | HELLO_INTERVAL / 4 |
| WILLINGNESS | dyn. calc. |
| LINK_QUALITY_LEVEL | 2 |
| LINK_QUALITY_WIN_SIZE | 10 |
| POLLRATE | 0.05 s |

node, this link will still be known as active for the time period of length NEIGHBOR_HOLD_TIME. In order to build up topology information, the MPR sends out *topology control* messages and the information gathering has to be completed within a time interval of length TC_INTERVAL. TOP_HOLD_TIME is used in case a link set of a node becomes empty. Then, empty topology control messages are sent for a duration of TOP_HOLD_TIME. TC_REDUNDANCY specifies the level of information details. In case of TC_REDUNDANCY $= 0$, only information about the MPR is sent. For TC_REDUNDANCY $= 3$ also information of each connected neighbor is communicated. MID_INTERVAL is the timeout for completing the multiple interface definition procedure, whereas this mechanisms is not required for nodes with only one interface. MID_HOLD_TIME specifies the time for which a multiple interface description is valid. In case a node is connected to hosts and networks, this node should advertise a *Host And Network* (HNA) message periodically every HNA_HOLD_TIME. DUP_HOLD_TIME gives the time period for which messages which are received multiple times are considered as duplicates. The parameters HYST_THRESHOLD_HIGH, HYST_THRESHOLD_LOW, and HYST_SCALING are used for sensing the link quality. The values of these parameters have to be in interval $[0,1]$ and the condition HYST_THRESHOLD_LOW $<$ HYST_THRESHOLD_HIGH must be true. OLSR uses an internal parameter *willingness* to determine the data forwarding behavior of a node. Default settings are stored in parameter WILL_DEFAULT and the values must be in the interval $[0,7]$, whereas 7 corresponds to always forwarding data, and 0 corresponds to never forwarding data. In case of high node mobility, the neighbors will change frequently and the nodes can increase the value for MPR COVERAGE ($> 0$) to ensure that two-hop neighbors are in range. Increasing this value will dramatically increase the protocol overhead traffic, as more control messages will be emitted and packet collisions will occur with a higher probability. Using parameter MESSAGE_INTERVAL, whereas MESSAGE_INTERVAL is a value between 0 and MAXJITTER, can regulate the message sending behavior of each node. For more details on all parameters please refer to [99].

#### BATMAN

BATMAN (Better approach to mobile ad-hoc networking) is an innovative approach to ad-hoc routing. Unlike other algorithms that exist right now, BATMAN does not calculate routes. It continuously detects and maintains the routes by receiving and broadcasting packets from other nodes. Instead of discovering the complete route to a destination node, BATMAN only identifies the best single-hop neighbor and sends a message to this neighbor. These messages contain the source address, a sequence number, and a time-to-live (TTL) value that is decremented by 1 every time before the packet is broadcasted. A message with a TTL value of zero is dropped. The sequence number of these messages is of particular importance for the BATMAN algorithm. As a source numbers its messages, each node knows whether a message is received the first time or repeatedly. The algorithm of the BATMAN message handling for nodes with a single interface is shown in Listing 3.1. This algorithm works only correct if none of the participating nodes in the mesh has more than one interface. Otherwise, DIRECT_NEIGHBOR and IDF has to be supported. Listing 3.2 shows the message handling for nodes with multiple interfaces.

```
A packet arrives:
 1.) If BATMAN VERSION is different:
                drop it;
 2.) If packet is a re-broadcast of own OGM:
                Mark OG as BDNB and drop it;
 3.) If OGM has UDF:
                drop it;
 4.) If received via BDNB and is not DUPLICATE:
                Update RANKING of PNTOG;
 5.) Re-broadcast packet if:
        1.) is from DIRECT\_NEIGHBOR\_IF and
                1.) is BDNB and BNTOG:
                        Re-broadcast;
                2.) (is BDNB and not BNTOG) or UDNB:
                        Re-broadcast with UDF;
        2.) is not from DIRECT\_NEIGHBOR\_IF and BDNB and BNTOG and
                1.) is not DUPLICATE:
                        Re-broadcast;
                2.) is BUPLICATE and TTL==BEST\_OG\_TTL:
                        Re-broadcast;
```

Listing 3.1: BATMAN message handling for nodes with single interface.

```
A packet arrives:
1.) If BATMAN VERSION is different:
                drop it;
2.) If packet is a re-broadcast of own OGM and:
                1.) if IDF is set and OGM IP == rx interface IP:
                        Mark OG as BDNB for this interface and drop it;
3.) If OGM has UDF:
                drop it;
4.) If received via BDNB and is not DUPLICATE:
                Update RANKING of PNTOG;
5.) Re-broadcast packet if:
        1.) is from DIRECT\_NEIGHBOR\_IF (pft src IP == OGM IP) and:
                1.) is BDNB and BNTOG:
                        Re-broadcast on all interfaces and with IDF
                        on receive interface;
                2.) (is BDNB and not BNTOG) or UDNB:
                        Re-broadcast with IDF and UDF on received
                        interface only;
        2.) is not from DIRECT\_NEIGHBOR\_IF and BDNB and BNTOG and
                1.) is not DUPLICATE:
                        Re-broadcast on all interfaces without IDF;
                2.) is BUPLICATE and TTL==BEST\_OG\_TTL:
                        Re-broadcast on all interfaces without IDF;
```

Listing 3.2: BATMAN message handling for nodes with multiple interfaces.

More details on BATMAN are given in [11]. In the presented test scenarios, BATMAN version 0.2 is used. Most of the parameters of BATMAN given in Table 3.4 are hard coded in the protocol. *Originator messages* (OGMs) are used by nodes to broadcast their existence to other nodes and to measure the link quality to neighbor nodes. BNTOG is a tuple which hold the best neighbor to a given originator. Each OGM which is received via a dedicated interface directly from the interface which initiating the sending leads to a tagging of this interface as DIRECT_NEIGHBOR_IF. This can easily be detected by comparing the originator message IP address and the source IP of the packet. Each OGM, and correspondingly the OGM-SQN tuple OSQT (where SQN is the sequence number of an OGM) which is received via a bidirectional neighbor (BDNB) without an unidirectional link flag (UDF), is tagged with a time stamp indicating the time of reception. An OGM is considered as DUPLICATE in case a corresponding

[11] https://www.open-mesh.net/batman/ (09.02.2010)

OSQT already exists for this OGM. The indirect link flag (IDF) has to be set in those OGMs which have to be re-broadcasted and which were received from a DIRECT_NEIGHBOR_IF, or which will be re-broadcasted via the interface where they have been received from. A potential neighbor to an originator (PNTOG) is generated by non-DUPLICATE OSQTs from a particular neighbor. Parameter OSQT_TIME_FRAME gives the time interval in which OGM-OSQT messages are maintained and considered for ranking the PNTOG. OSQT_FRAME is suggested as alternative for OSQT_TIME_FRAME and consists of the sequence number range within OGMs which are used for ranking the PNTOG. The value for the BEST_KNOWN_TTL is deduced from OSQTs which are not DUPLICATE and received via the best neighbor originator BNTOG within the current OSQT_TIME_FRAME or the last OSQT_FRAME.

Table 3.4: Parameters for BATMAN.

| Parameter Name: | Default Value |
|---|---|
| BNTOG | best neighbor to originator |
| BDNB | bidirectional neighbor |
| BEST_KNOWN_TTL | best known TTL to a given BNTOG |
| DEV-IF | network device |
| DIRECT_NEIGHBOR_IF | OGM received via dedicated interface |
| DUPLICATE | OGM already has a record in corresponding OSQT |
| IDF | direct link flag |
| PNTOG | potential neighbor to originator |
| OGM | originator message |
| SQN | sequence number of an OGM |
| OSQT | OGM SQN tuple |
| OSQT_TIME_FRAME | time frame for ranking OGMs |
| OSQT_FRAME | suggested alternative for OSQT_TIME_FRAME |
| VERSION | lowest version compatible |
| UDF | unidirectional link flag |
| UDNB | unidirectional neighbor |

### 3.1.4 Evaluation of Ad-Hoc Routing Protocols

Ad-hoc routing protocols are designed for scenarios where networks need to support a high flexibility in terms of mobility and network topology. To cover this broad range of possible setups and situations, many different types of ad-hoc routing protocols were developed according to different design philosophies. However, this diversity also complicates meaningful comparisons and evaluations. Currently, three main approaches for ad-hoc routing protocol analysis and evaluation are established in the scientific community. First, protocol evaluation can be based on analysis using parameters like time complexity or communication complexity. The second method uses protocol simulations [108] [104]. Well known tools for these protocol simulations are OPNET[12], Network Simulator[13], or GloMoSim [109]. Of course, all these simulations results are strongly depending on the conditions and parameter settings of the simulation and also on the used simulation tool itself. A third method for evaluation is the implementation of the protocols and the setup of a real world test scenario. This method can easily show the real world performance of a protocol but for the evaluation for a large number of protocols, the implementation effort might be too high. Nevertheless, each of the above described methods need a useful metric in order to allow for a meaningful performance evaluation [110] [12].

### 3.1.5 Networked Robotics and Ad-Hoc Networks of Mobile Robots

This section gives a brief overview of different communication approaches used in the area of mobile robots during the last years. The typical teleoperation scenario in the past aimed on simplicity, and had the objective to guarantee reliable teleoperation of one mobile robot by a dedicated control station. The robot, as well as the control station use special equipment which provides a communication link only between these two entities, and a further integration of additional network nodes is not foreseen. Examples for the successful application of this approach are given in [111], where a rover is remotely controlled via a dedicated wireless link while navigating in an outdoor environment. In [112] an autonomous unmanned helicopter is connected to the ground station via a dedicated

[12]http://www.opnet.com (09.02.2010)
[13]http://www.isi.edu/nsnam/ns/ (09.02.2010)

link based on the DECT technology[14]. Nowadays, mobile robots are often inte-

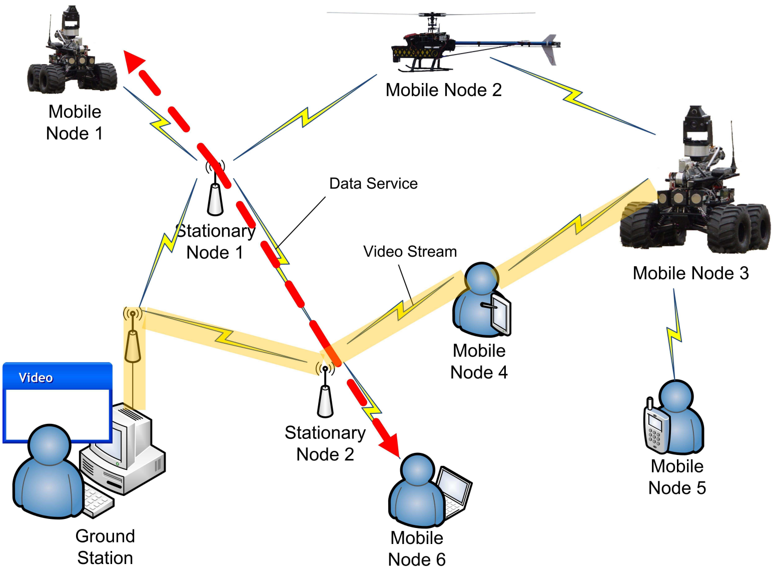


Figure 3.4: Future scenarios for mobile robot ad-hoc networks.

grated into existing IP based networks to enhance interoperability and increase their data distribution capabilities. In current research projects, several network topologies – starting form wireless LANs using infrastructure mode and several access points up to wireless ad-hoc networks using different ad-hoc routing protocols – were used [113], [114], [115], [116], [117]. Often, these networks use standard IP based communication together with WLAN. An advantage of WLAN is the availability of a relatively high bandwidth and the high flexibility in integrating new protocols or extending features of available protocol implementations. As currently the cooperation of several vehicles is very important, challenging problems like nodes acting autonomously as communication relay, a highly dynamic and variable network topology (some network nodes may leave or join the network at any time), routing problems, and several data streams and sources with different bandwidth requirements have to be solved. Often, ad-hoc

[14]ETSI-Standard EN300175

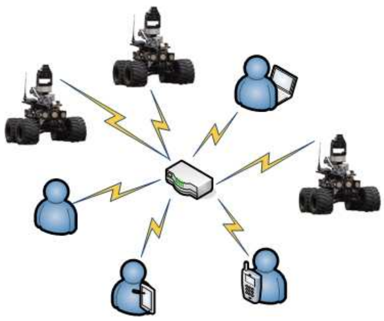
capabilities must be present in current scenarios.

As described above, the wireless communication equipment is very sensitive to external influences and disturbances with respect to reliable transmission of data. Often, a direct line-of-sight is required to maintain the communication link. In case of extending the communication range also into areas which are in the communication shadow behind an obstacle, the use of communication relays is currently a common approach. In [118], some communication relays are placed in case the link quality decreases, and thus, the range and reliability is increased. Nevertheless, in this case the topology of the multi-hop network is decided in advance and configuration is done static before the relay nodes are deployed. Thus, a later change of the logical network topology is not foreseen, and in this kind of setup on demand rerouting is not possible. In case the stationary communication relays should be replaced by mobile nodes which can act either as active part of a mobile robot team or just as communication relay if necessary, dynamic topologies are coming into play. The capability of maintaining connectivity in a network with dynamic topology can be achieved by applying ad-hoc routing mechanisms. As already mentioned in the previous sections, there are many existing mechanisms acting according to different topology update principles.

These mechanisms allow for a scenario which is displayed in Figure 3.4. This scenario is based on a network which supports dynamic topologies and the integration of different network nodes (human, robots, sensors, stationary communication relays,...) by a common standard in combination with a transparent communication between all network nodes. This scenario is currently very interesting with respect to the application in the area of networked robotics as it supports one-to-one, one-to-many, many-to-one, and any-to-any communication on a logical layer. This type of flexible communication is transparent for the user and it is a must for nowadays networked robot scenarios, where a seamless networking of all nodes is a prerequisite to accomplish a task successfully. Currently, it can be easily realized with existing hardware devices and technologies like IEEE 802.11 Wireless LAN which is well known. Nevertheless, some challenges must be coped for a successful application of this technology in the field of networked robotics. In these scenarios often video streams, sensor data, and commands to the robots are transmitted. Thus, a mixture of different types of traffic (delay sensitive, packet loss sensitive, real time,...) is present in the network at the same time. This might also require additional mechanisms on application layer like in

[9], where a traffic shaping algorithm is presented. Also an appropriate parameter setting of the used protocols will increase the performance of the network and will enable the interconnection of robots and humans via a common network. The following chapters will describe how these complex networks can be set up easily with standard WLAN equipment which is currently available.

The application of WLAN for networked robots is a common approach since the last years. In 2000, a small mobile robot was controlled via WLAN [119]. In 2003, Ollero integrated an access point into an unmanned helicopter system [114] to interconnect flying network nodes and a ground segment which was later extended to a network connecting three unmanned aerial vehicles (UAVs) [120]. In this case, the scenario did not require fast changing network topologies and ad-hoc capabilities of the network which allows for an access point as coordinating entity for media access. This architecture is shown in Figure 3.5a. A different, much more flexible scenario is depicted in Figure 3.5b. Here, the network consists of stationary nodes, mobile robots (unmanned ground vehicles and unmanned aerial vehicles), and human team members. In addition to the previously mentioned scenario, several network nodes are mobile which requests the support of dynamic topologies. It is also possible that new nodes will join, or existing nodes will leave the network at any time during the assigned mission. Inside the network each node competes for the medium access in its transmission range (cf. Section 3.1.2).


(a) Infrastructure based setup.

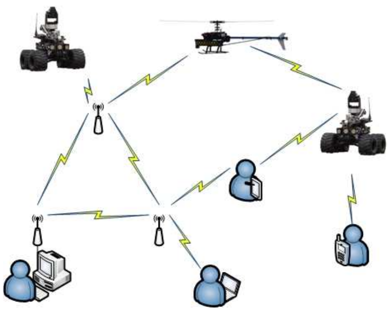
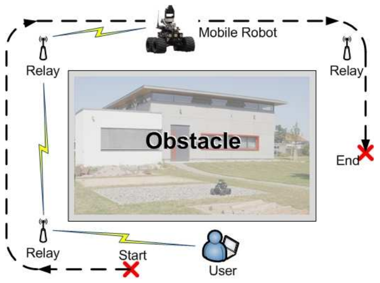
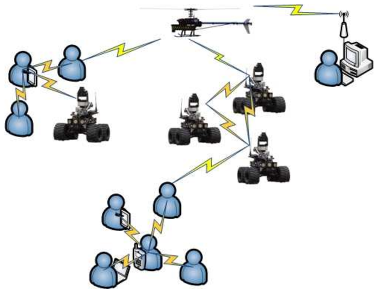

(b) Ad-Hoc network Setup.

Figure 3.5: Different topologies for wireless LAN.

In Figure 3.6a a typical scenario from the area of mobile robotics – especially for remote exploration tasks – is shown. Objective is the teleoperation and navigation of a mobile robot via wireless communication link. The communication infrastructure is set up on demand and in the test setup of Figure 3.6a an obstacle blocks a direct communication link to an area which should be explored by the robot. To provide seamless communication between the operator and the mobile robot, several nodes act as communication relays on-demand. The capability of using some nodes as communication relay is provided by an ad-hoc routing protocol. A more advanced scenario where the communication via relay nodes and the ad-hoc communication inside the team/swarm is combined is shown in Figure 3.6b. Here, several teams of humans and robots, or even heterogeneous teams are connected, whereas single dedicated robots may act as relay or even a team can act as communication relay. In this scenario a highly integrated network infrastructure is required which provides a transparent communication layer for the users. Also in this scenario, several types of traffic (e.g. video, voice, sensor data, or commands) with different characteristics and priorities must be transported via these communication links on-demand.



(a) Test scenario for ad-hoc routing protocols.



(b) Network of heterogeneous teams.

Figure 3.6: Typical network topologies for networked robots.

### Teleoperation with Local Autonomy

Teleoperation of mobile robots always imply security matters. On the one hand, data integrity must be assured to prevent deliberate misuse of the robot and mech-

anisms for security aspects with respect to data security, user management, prevention of misuse or intrusions, and encryption of connections must be implemented. Often, research prototypes or testbeds are not including these aspects, but with respect to industrial applications the presence data security is inevitable. Nowadays, many proprietary solutions exist but governmental organizations (e.g. BSI[15]) supports the standardization and recommend the use of IEEE 802.11i / 802.1x in industrial and commercial solutions. Besides these aspects, also the operation of mobile robots like an unmanned aerial vehicle (e.g. a helicopter) or an unmanned ground vehicle holds a risk in terms of endangering or injuring persons or damaging the environment. As soon as a wireless link is incorporated into the teleoperation or control, the potential loss of the communication link must be considered. Several approaches exist to prevent the robots from being a risk for the living or non-living environment.
As previously mentioned, the design of teleoperation mechanisms via wireless communication links must always be able to cope with packet loss, high jitter, or even a communication drop out. One possibility to allow teleoperation via lossy communication links is a combination of local autonomy which takes over the control of the most critical and necessary subsystems of the mobile robot in case communication is lost, and a teleoperation interface which allows the user an appropriate command of the robot while communication is available.

With respect to UGVs, the Outdoor MERLIN of University of Wuerzburg follows an approach which is similar in some aspects [121]. The MERLIN robot carries several different sensors for obstacle detection which allows a more powerful local autonomy. Thus, in case of a bad link or lost communication, MERLIN can still perform some of its tasks [111].

### Generated Traffic

Usually, the traffic transmitted via a multi-hop network of mobile robots has a broad spectrum in its characteristics. The network might be used for exchanging packets for control loops, commands, different kinds of sensor data, or in heterogeneous teams even voice. The traffic which is usually generated by controllers consists in rather small packets sent with a high sampling rate (e.g.$>$ 100Hz). Commands sent by users are often transmitted with a much smaller frequency.

[15] Bundesamt fuer Sicherheit in der Informationstechnik in Germany

Table 3.5: Sensor Packets.

| Type | Size (bytes) | Interval (ms) |
|---|---|---|
| Communication Status | 6 | 500 |
| Link Status | 8 | 500 |
| Orientation | 17 | 300 |
| GPS 1 | 15 | 1000 |
| GPS 2 | 22 | 1000 |
| GPS 3 | 6 | 1000 |
| GPS 4 | 17 | 1000 |
| Ultrasonic | 13 | 500 |
| Motor Status | 17 | 1000 |
| Energy Status | 21 | 1000 |

With respect to sensor data feedback, the characteristics of the generated packet stream depends on the kind of sensors which are used. The range lasts from only a few bytes (e.g. a single value for representing a measured velocity) up to large streams with a bandwidth of more than 3 Mbit/sec (e.g. video data). Besides the required bandwidth of the corresponding data flow, also the sampling rates of data affects the overall performance of the 802.11 link. In case of packets for control mechanisms or video transmissions a rather constant packet inter-arrival time with a small jitter (variance of inter-arrival times) is desirable.

The following sections give an analysis of ad-hoc routing protocols with respect to teleoperation of mobile robots. Therefore, the MERLIN robot is used for a navigation and exploration task while the data flow between robot and control PC is recorded and analyzed. The MERLIN Control Protocol is used as command protocol for the robot. It is located at the application layer of the ISO/OSI model. During the tests, several packet types are exchanged between robot and control PC. The command packet has a payload of 13 bytes. These command packets are sent with a frequency of 10 Hz. The robot itself generates 10 different packets (cf. Table 3.5).

For further details on the MERLIN Control Protocol refer to [111] and [121].

As transport protocol, UDP is used which will additionally add 8 bytes for the UDP header. Thus, the complete communication fully complies with the ISO/OSI reference model.

## 3.2 Ad-Hoc Routing Protocols for Mobile Robot Teleoperation

The application of WLAN in real world scenarios is getting more and more popular, as now stable versions of ad-hoc routing protocol implementations exist. This section analyzes four ad-hoc routing protocols with respect to mobile robot teleoperation and some of the results are published in [11]. From the existing implementations of ad-hoc routing protocols, Ad-hoc On-demand Distance Vector (AODV), Dynamic Source Routing (DSR), Optimized Link State Routing (OLSR), and Better Approach to Mobile Ad-Hoc Networking (BATMAN) are selected for the tests.

### 3.2.1 Test Setup and Evaluation

To allow for a characterization of the ad-hoc routing protocols for mobile robot teleoperation applications, the analysis of the duration for the rerouting is important. For the performed tests, several mobile nodes were used. One node is a PC for the operator. Up to 4 MERLIN robots (standard version) were used as stationary communication relay nodes, and one Outdoor MERLIN was used (cf. Figure 3.7) as teleoperated mobile node. All MERLIN robots have a C167 micro

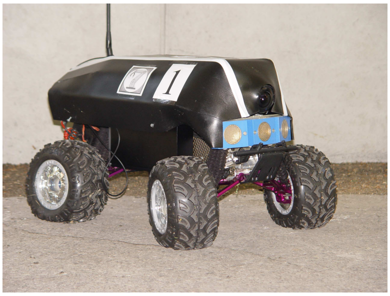


Figure 3.7: The teleoperated OutdoorMERLIN robot.

controller for low-level operations and sensor data processing, as well as a PC-104 for more complex and computationally more intensive tasks. The PC-104 uses a Linux operating system and all nodes are equipped with 802.11b standard WLAN equipment (Atheros chip). For steering the mobile robot, the operator's PC is running an application which generates command packets of a size between 6 and 22 bytes of payload. These packets are sent via UDP over the wireless network to the mobile robot. The onboard software of the mobile robot generates a UDP packet stream of packets with variable size containing the sensor data.

The scenario is set up in a way that the rerouting procedure will start with the mobile robot being at a certain location. Therefore, a large building is used as obstacle (cf. Figure 3.8). Relay nodes are placed at the corners of the building, such that they have always the neighbor nodes at the next and previous corner of the building within their communication range. Thus, the rerouting procedure can be initiated at dedicated locations as soon as the mobile robot is moved out of the line-of-sight of one node, the rerouting procedure is initialized. This scenario represents a worst case in terms of link redundancy, as only one route between operator PC and mobile robot is available without any redundant backup route existing. Relevant measurement categories are the packet loss and the duration of a communication drop-out during rerouting.

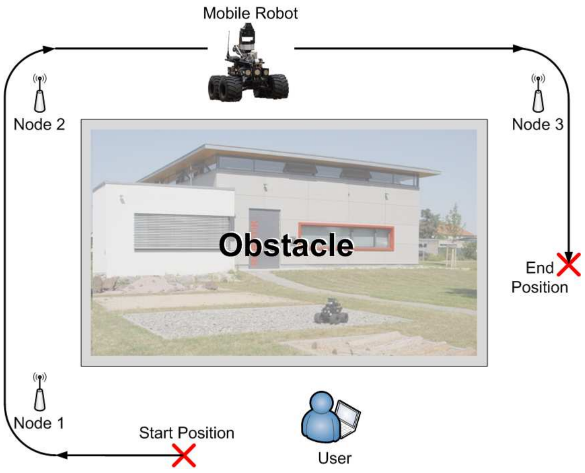


Figure 3.8: Test Setup.

### 3.2.2 Performance of Ad-Hoc Routing Protocols with Default Settings

In a first step, the four well known ad-hoc routing protocols AODV, OLSR, DSR, and BATMAN are used with standard parameter settings and the round trip time (rtt) of packets, the packet loss, and the duration of rerouting is analyzed in the scenario described in Section 3.2.1. Therefore, a direct communication link between the mobile robot and the control PC is established, and the robot is driven around a corner of a building which causes a communication break down after losing the line-of-sight. Now, the tested ad-hoc routing protocols have to reestablish the communication link via an additional mobile robot acting as communication relay node. To receive the required amount of measurement data for evaluation, several test runs are carried out. The time which is required for rerouting is the observed measurement category. To get an idea of the reactions of the different ad-hoc routing mechanisms, the behavior of a representative test run of each protocol is shown in the next paragraphs. Finally a summary of the overall results is given.

#### Performance of BATMAN with Default Parameter Settings

At first, each node is configured with the BATMAN protocol. In Figure 3.9, the round trip times of the command packets for this a run are displayed. This test runs shows a behavior which was observed at all test runs of this protocol at default parameter settings. At around 45 seconds of test time, a direct line-of-sight connection to the controller is not possible anymore and the communication protocol has to include one more hop (node 1). After losing of the line-of-sight, a communication drop-out is recognized and the robot stops. For more than 50 seconds, the BATMAN protocol is not able to find a new route between controller and robot. This result occurred for all test runs of BATMAN performed with standard parameter settings.

#### Performance of AODV with Default Parameter Settings

In Figure 3.10, the round trip times of the control packets during the test of AODV are shown. At 35 seconds, a direct communication between controller and robot is not possible anymore. Surprisingly, AODV reestablishes a route via node 2 (Robot→N2→N1→PC) and 19.8 seconds later, communication continues. This

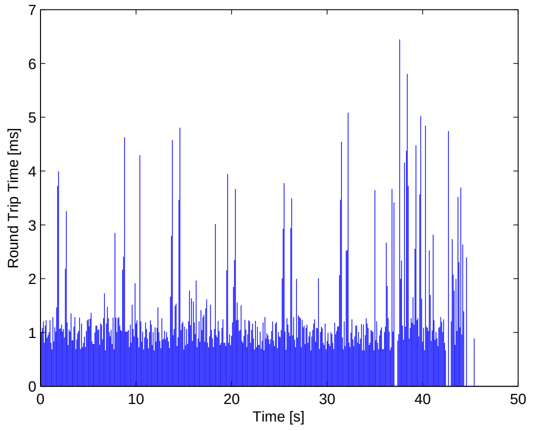


Figure 3.9: Round trip times for BATMAN with default parameter settings.

behavior of selecting not the shortest route with respect to the number of intermediate nodes is observed at approximately one fourth of all the performed AODV tests. At 67 seconds, the line-of-sight between the robot and node 2 is lost after moving the robot around the next corner of the obstacle. Two short communication drop-outs appear within the next 10 seconds and a new route via node 3 is used. At 80 seconds of test time, the robot returns on the same way to the start position. On this way back the number and length of the communication drop-outs are negligible, as the AODV protocol on the mobile robot has the required routes already in its route cache.

### Performance of OLSR with Default Parameter Settings

The round trip times of the command packets during the OLSR test are given in Figure 3.11. After 71.5 seconds, the line-of-sight between robot and controller is lost. 10.1 seconds later, the connection via node 2 is established. At 94.5 seconds test time, the direct communication between node 1 and the robot is lost. 14.7 seconds later, a route via node 2 is active. At about 125 seconds test time, the communication is forced to include node 3. During the communication via 4 hops, several packets are lost and the variance of the round trip time is signifi-

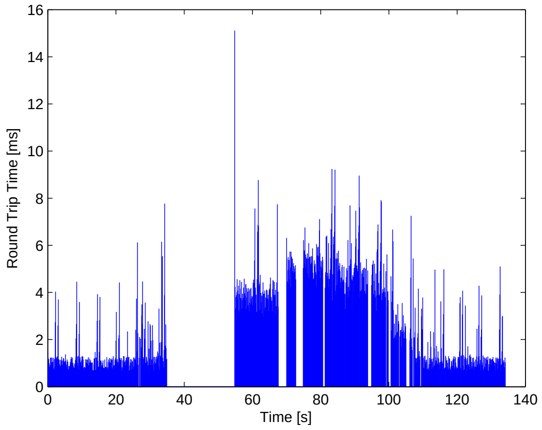


Figure 3.10: Round trip times for AODV with default parameter settings.

cantly higher than during the communication via links with a smaller amount of hops. On the robot's way back to the controller, node 2 is not used anymore, as OLSR establishes a route via node 1 after losing the line-of-sight to node 3.

### DSR

Figure 3.12 shows the command packet round trip times of the DSR test run. Between 35.2 seconds and 37.6 seconds test time, communication via node 1 is established. At 49.6 seconds, direct communication between node 1 and the robot is lost. 2.5 seconds later, node 2 is included into the route, and communication is reestablished. At 62.3 seconds test time, DSR establishes a new route via node 3 within 2.7 seconds. On the way back, only very few packets are lost as the route re-establishing works fast and reliable.

### Packet Loss & Time for Rerouting

These four examples of test runs give a good impression of the different behavior of the evaluated ad-hoc routing mechanisms. Unfortunately, BATMAN is unable to re-establish a communication link over one or more intermediate nodes. Thus,

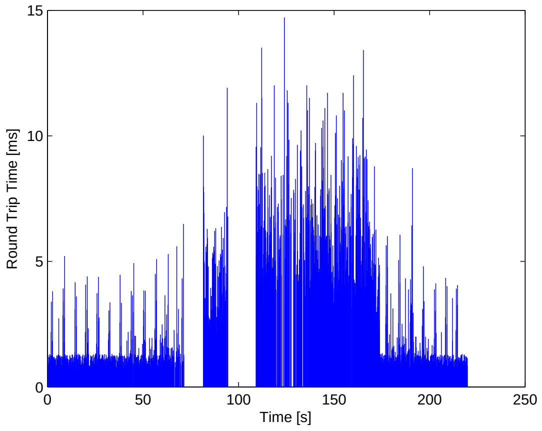


Figure 3.11: Round trip times for OLSR with default parameter settings.

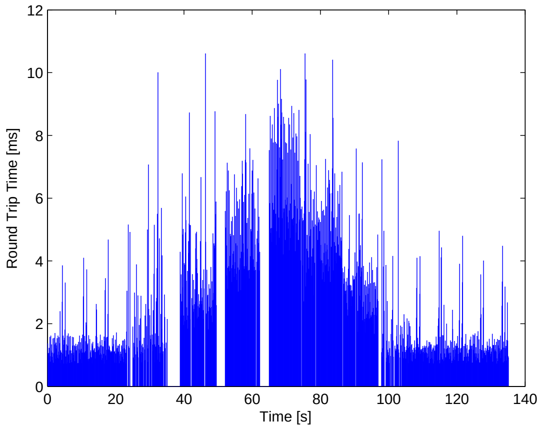


Figure 3.12: Round trip times for DSR.

Table 3.6: Packet loss & times for route re-establishing.

| Protocol | Packet loss during test run | Time for rerouting min. | Time for rerouting max. |
|---|---|---|---|
| AODV | 29.2% | $2.1s$ | $> 30s$ |
| OLSR | 14.2% | $10.1s$ | $> 30s$ |
| DSR | 11.2% | $2.4s$ | $2.7s$ |

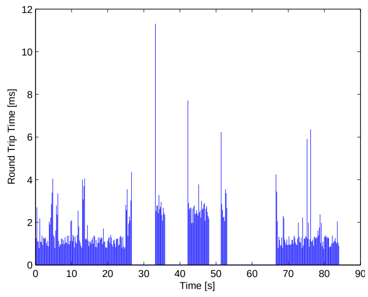
it is not included into table 3.6 which shows the observed minimum and the maximum of the time which is required by each protocol to reestablish a route is given. The value for the maximum time for rerouting of ($> 30s$) for AODV and OLSR means that there are test runs in which a complete communication loss appeared. The values of the time for rerouting given in Table 3.6 are only taken from the robots way from controller to node 3. On the way back, measurements are not used as re-establishing the new route works much faster due to the required information being already present in the route cache. This table clearly shows, that among the four tested protocols configured with the default parameter settings, DSR performs best in the tested scenario. Table 3.6 also shows the average packet loss for all test runs. The mobile robot always started at the same start location and is driven via remote control to the goal position which is also the same for all test runs. Also the driven path, and thus, the areas for initiating the rerouting procedure are identical in each of the tests which allows for comparing the test runs. The packet loss in Table 3.6 is now computed over the complete test duration on the way from the start location to the goal position and the average value of each protocol is given. Also here, DSR gives the best results compared to AODV and OLSR.

### 3.2.3 Protocol Parameter Tuning

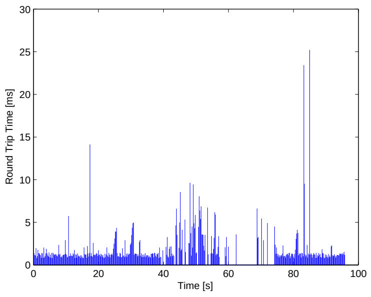
As using the default parameter settings results in quite unsatisfactory results for mobile robot teleoperation scenarios, where the time for rerouting should be as small as possible, parameter tuning of the evaluated ad-hoc routing mechanisms is investigated. Each routing protocol has different parameters which will have different effects when changed. For the following tests, the setup is similar to the test setup for the default parameter settings described in the previous section.

### AODV

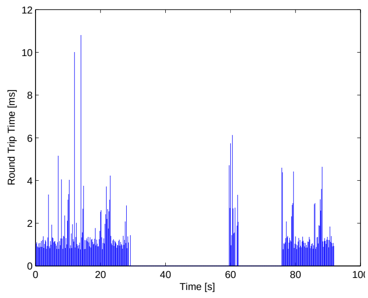
AODV has numerous parameters which can can be tuned in order to increase the performance, and Figure 3.13 gives a comparison of representative test runs with different parameter settings. Therefore, tests were performed with the FORCE_GRATOUITOUS flag enabled which should result in a faster route discovery as intermediate nodes are allowed to send a reply on RREQs if the requested destination is known. A result of a test run with the FORCE_GRATOUITOUS flag enabled is shown in Figure 3.13b. Also tests with link layer feedback being enabled are performed. Enabling link layer feedback allows AODV to interpret link layer messages in order to identify link failures earlier. Figure 3.13d shows the result for *receive 2 hellos from host before treating as neighbor* enabled. This setting should prevent from setting a node as neighbor immediately after receiving only one hello message. Thus, a node must be reachable for a longer period of time before it is being treated as neighbor. This procedure might result in more stable links. AODV can also be used with sending *Hello* messages only when forwarding data. This will reduce the routing protocol overhead to the minimum, but might have a negative effect on the response of the routing protocol on network topology changes. The results for this test run are shown in Figure 3.13e. In comparison to the default settings of AODV (cf. Figure 3.13a) the following conclusions can be drawn. Enabling the FORCE_GRATOUITOUS flag does not lead to the desired effect of reducing the rerouting time. In contrast, the performance really decreased significantly as no multi-hop communication could be established. Also the test with enabling the link layer feedback does not improve the performance. For the test run with AODV waiting for the reception of two hello messages from a host before treating this host as neighbor, it is expected that the rerouting requires a little bit more time, but finally, the link should be more stable. As expected, the change of this parameter results in a slower rerouting procedure but unfortunately, the link cannot be establishes stable. The tests for using optional *Hello* messages (cf. Figure 3.13e) delivered the expected results and the performance is really poor. All these results lead to a test setup with a combination of changed parameter settings. Therefore, AODV is used with the FORCE_GRATOUITOUS flag set and waiting for the reception of two hello messages from a host before treating this host as neighbor. The



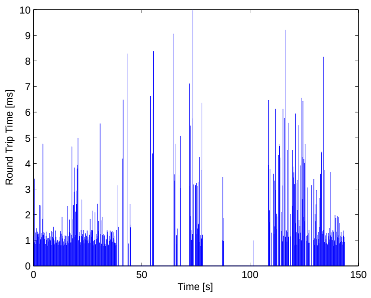
(a) Default settings.



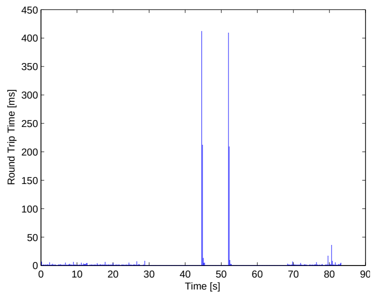
(b) *Force gratuitous* enabled.



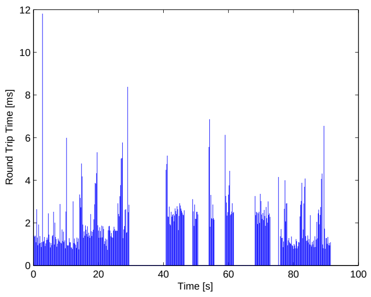
(c) *Link layer feedback* enabled.



(d) *Accept neighbor after 2 HELLOs* enabled.



(e) Sending HELLO messages only when forwarding data.



(f) Combination of tuned parameters.

Figure 3.13: Round trip times of AODV with different parameter settings.

results are shown in Figure 3.13f. As changing a single parameter without positive results, the expectations for the combined tuning are quite low. Nevertheless, AODV can establish a multi-hop communication via one hop, but the durations of communication drop outs are relatively long and not acceptable for mobile robot teleoperation. A summary of the results is shown in Table 3.7, whereas only minimum and maximum durations for route re-establishing are shown. In this case, the calculation of an average duration for route re-establishing is not meaningful as many test runs were unsuccessful.

Table 3.7: Times for route re-establishing for AODV with tuned parameters.

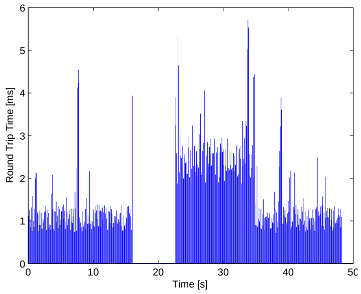
| Parameter Setting: | min. time: | max. time: |
|---|---|---|
| Default | 2.4s | $>$ 30s |
| Receive 2 HELLO messages before treating node as neighbor | 2.2s | 11.8s |
| Enable local repair | 3.8s | 15.4s |
| Link layer feedback | 3.6s | 28.8s |
| Combined | 2.2s | 16.8s |

### OLSR

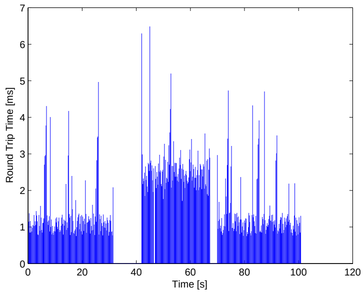
According to the recommendation in the OLSR specification[16], the OLSR configuration file is changed with USE_HYSTERESIS being disabled and parameter LINK_QUALITY_LEVEL $= 2$. The following Figures 3.14 and 3.15 show the comparison of several OLSR test runs with different parameter settings. In Figure 3.14a, results are displayed for OLSR with default settings and it can be observed that OLSR can reestablish a stable route after 12 ms. For all tests a minimum duration for rerouting of 12 s, an average of 13.24 s, and a maximum of 16.6 s is observed. In Figure 3.14b results are shown for parameter WILLINGNESS$= 7$. Thus, each node acts as relay for all data which is received. For all test runs with this parameter setting, the required time for rerouting is not shortened and no

[16]http://downloads.open-mesh.net/misc/olsrd/olsrd.conf.example-german-comments, visited December 2008

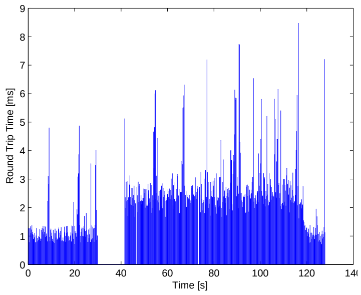
improvement is observed. Quite contrary to the expected behavior, with a measured average drop-out length of 39.08 s, a minimum of 10.0 s, and a maximum of 68.8 s, the protocol performance is even decreased significantly. For the next test run LINK_QUALITY_WIN_SIZE is changed from 10 to 100 according to recommendations of the OLSR developers. LINK_QUALITY_WIN_SIZE measures statistics about link stability and link availability. Of course, the full capability of this parameter will be more significant in a network with a higher grade of meshing. In the test scenario, only the time of initiating the rerouting might be influenced. The results for this parameter setting is shown in Figure 3.14c. For the tests with this parameter settings, an average drop-out time of 12.44 s, a minimum of 9.8 s, and a maximum of 15.4 s is measured, which gives no significant difference to the protocol behavior with default settings. A more important parameter is HELLO_INTERVAL which is decreased from 2.0 seconds to 0.5 seconds. The corresponding results are shown in Figure 3.14d. Thus, new neighbor nodes are detected faster and a route re-establishing should work faster. On the other hand, protocol overhead is increased by this parameter adjustment. It will not effect the relatively small test setup, but for bigger networks which a much higher number of nodes, this might cause problems. For this setting, the average drop-out time is 14.72 seconds, the minimum is measured at 11.0 seconds, and the maximum is at 19.4 seconds which means that no change in performance is achieved but the overhead is increased. Only changing this parameter has no immediate effect as HELLO_VALIDITY might preponderate the impact of HELLO_INTERVAL. Results for changing parameter HELLO_VALIDITY are displyed in Figure 3.15a. For these test runs, HELLO_VALIDITY is reduced from 6.0 seconds to 3.0 seconds which results in updating neighbor twice as fast as before. Again, the improvement of the performance is lower than expected with an average drop-out time of 9.56 seconds, a minimum of 7.6 seconds, and a maximum of 11.0 seconds. Also TC_INTERVAL seems to be a relevant parameter as it is responsible for the exchange of topology information. For the results shown in Figure 3.15b, TC_INTERVAL is reduced from 5.0 seconds to 2.5 seconds. Thus, topology changes should be discovered twice as fast and rerouting should be faster. The measured average drop-out time for this setup is 11.64 seconds, the minimum is at 10.6 seconds, and the maximum is at 12.8 seconds which is not the expected significant change compared to the default settings. Reducing TC_VALIDITY_TIME from 15.0 seconds to 5.0 seconds (cf. Figure 3.15c) re-



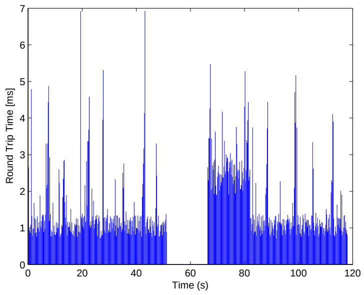
(a) Default parameter settings.



(b) WILLINGNESS= 7.



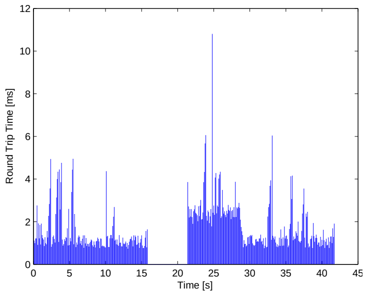
(c) LINK_QUALITY_WIN_SIZE= 100.



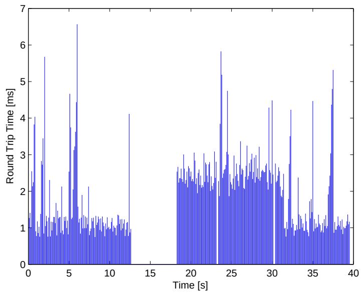
(d) HELLO_INTERVAL= 0.5 s.

Figure 3.14: Round trip times for OLSR with different parameter settings.

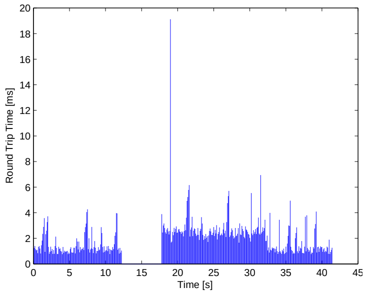
sults in an average drop-out duration of 10.4 seconds, a minimum of 7.8 seconds, and a maximum of 12.0 seconds. Changing only single parameters did not result in the desired performance improvement. Finally, these results lead to a test setup where several parameters are tuned simultaneously. Figure 3.15d shows the result for HELLO_INTERVAL= 0.5 seconds, HELLO_VALIDITY= 1.5 seconds, TC_INTERVAL= 2.5 seconds and TC_VALIDITY=5.0 seconds. For this setup, the average drop-out duration is 5.96 seconds with a minimum of 5.2 seconds and a maximum of 7.4 seconds. This is an improvement of the performance compared to the results of the test runs with default parameter settings (cf. Table 3.8).



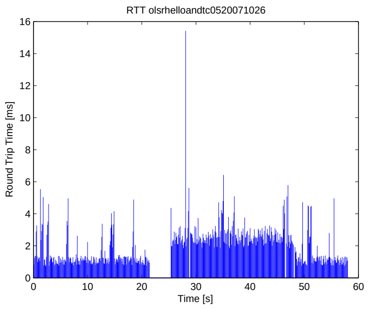
(a) HELLO_VALIDITY= 3.0 s.



(b) TC_INTERVAL= 2.5 s.



(c) TC_VALIDITY= 5.0 s.



(d) HELLO_INTERVAL= 0.5 seconds, HELLO_VALIDITY= 1.5 seconds, TC_INTERVAL= 2.5 seconds, and TC_VALIDITY=5.0 seconds.

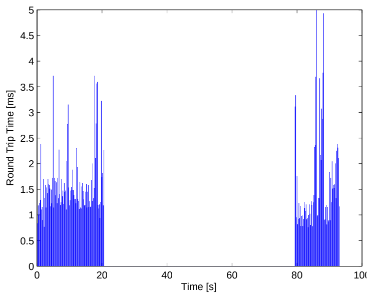
Figure 3.15: Round trip times for OLSR with different parameter settings.

Table 3.8: Times for route re-establishing for OLSR with tuned parameters.

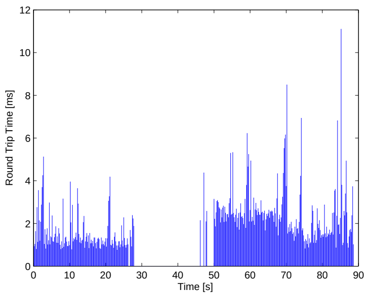
| Parameter Setting: | min. time: | avg. time: | max. time: |
|---|---|---|---|
| Default | 12.0s | 13.24s | 16.6s |
| WILLINGNESS= 7 | 10.0s | 39.08s | 68.6s |
| LINK_QUALITY_WIN_SIZE= 100 | 9.8s | 12.44s | 15.4s |
| HELLO_INTERVAL= 0.5 | 11.0s | 14.72s | 19.4s |
| HELLO_VALIDITY= 3.0 | 7.6s | 9.56s | 11.0s |
| TC_INTERVAL= 2.5 | 10.6s | 11.64s | 12.8s |
| TC_VALIDITY= 5.0 | 7.8s | 10.4s | 12.0s |
| Combined | 5.2s | 5.96s | 7.4s |

#### BATMAN

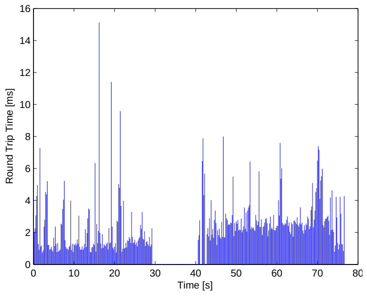
As documented in the previous section, and as described in [11], all BATMAN test runs with standard parameter settings ended up in a communication loss. and the BATMAN protocol with default parameter settings could not succeed in the present scenario setup. Figure 3.16a and Figure 3.16b show this unacceptable protocol behavior with communication losses for more than one minute and more than 20 seconds. For the test with these settings BATMAN showed a very unpredictable behavior, and in cases where rerouting was successful the measured minimum duration for a communication loss is 6.0 seconds, the maximum duration is 58.2 seconds, and the average is 34.4 seconds. Out of the very small number of variable parameters for BATMAN, ORIGINATOR_INTERVAL seems to be a suitable parameter for optimization. ORIGINATOR_INTERVAL defines the time to wait before the batman daemon sends the next message (default value is 1000 milliseconds). Now, the value for ORIGINATOR_INTERVAL is decreased which should result in a faster response on topology changes by the routing protocol. The results for ORIGINATOR_INTERVAL= 250 ms are shown in Figure 3.16c. Of course, this parameter setting increases the routing protocol overhead by factor four but also allows for a faster and more reliable route recovery. During these tests, no complete loss of the communication is observed. The minimum time for route re-establishing is 11.2 seconds, the maximum is 21.4 seconds, and the average is at 12.76 seconds. For the results of Figure 3.16d the ORIGINA-



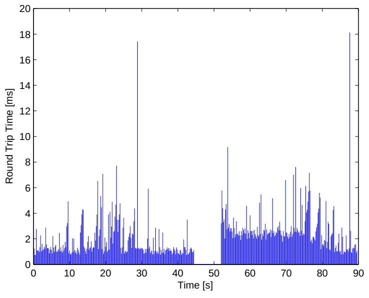
(a) Default settings - communication loss for 1 minute.



(b) Default settings - communication loss for more than 20 seconds.



(c) ORIGINATOR_INTERVAL= 250 ms.



(d) ORIGINATOR_INTERVAL= 125 ms.

Figure 3.16: Round trip times for BATMAN with different parameter settings.

TOR_INTERVAL is reduced to 125 milliseconds which again raises the routing protocol overhead by four. Now, the network reacts again more reliable with an average drop-out duration of 13.4, a minimum of 5.4 seconds, and a maximum of 37.6 seconds. This parameter setting improved the performance of BATMAN significantly (cf. Table 3.9). Unfortunately, this improvement is achieved on very high costs for the network as the routing protocol overhead is increased tremendously. The present scenario is relatively small and has no redundant routes. In this case the increase of protocol maintenance traffic does not endanger the stability and does not reduce the available throughput significantly. But in case of larger networks, a setting of ORIGINATOR_INTERVAL$=$ 125 ms should be used carefully. Nevertheless, it is not possible to get results comparable to DSR, and the duration of the rerouting procedure takes quite long compared to the requirements of direct teleoperation of mobile robots.

Table 3.9: Times for route re-establishing for BATMAN with tuned parameters.

| Parameter Setting: | min. time: | avg. time: | max. time: |
|---|---|---|---|
| ORIGINATOR_INTERVAL$=$ 1000 | 6.0s | 34.4s | $>$ 60s |
| ORIGINATOR_INTERVAL$=$ 250 | 11.2s | 12.76s | 21.4s |
| ORIGINATOR_INTERVAL$=$ 125 | 5.4s | 13.4s | $>$ 30s |

### Summary

The previous paragraphs give an overview and a comparison of the four ad-hoc routing protocols AODV, OLSR, DSR, and BATMAN in a scenario where a direct communication link between mobile robot and control PC is lost after driving around an obstacle. The communication link has to be re-established via one relay node. It is shown, that some tuned parameter settings are useful to increase the performance of the protocol and the duration of the communication losses can be shortened. Unfortunately, it is not possible to find a suitable setup for AODV which provides a reliable rerouting. For OLSR, the rerouting duration is improved by a combined change of the parameters HELLO_INTREVAL, HELLO_VALIDITY, TC_INTERVAL, and TC_VALIDITY. In the initial tests, the BATMAN protocol could not reestablish a link by including one more communication relay node. With the new parameter settings, the rerouting procedure of the BATMAN protocol works fast and reliable. Nevertheless, the improvements of all these protocols are achieved by generating some more network maintenance traffic. Of, course, this aspect must always be considered and this trade-off between the available bandwidth and topology updates will be more important with a growing number of network nodes. It was shown, that parameter tuning improves the behavior and the reliability of AODV, OLSR, and BATMAN significantly in the given worst case scenario. The advance of the performance of DSR due to advantages of the protocol design philosophy with respect to the given worst case scenarios could not be balanced.

### 3.2.4 Multi-hop Communication

The scenario used for the tests in Section 3.2.3 is relatively small and the focus is set on setting up a communication link via only one relay node. This defined setup provides the required frame for analyzing the behavior of the different routing protocols and for the characterization of the rerouting procedure in more complex scenarios. It is shown, that an appropriate parameter setting definitely increases the routing performance. But in parallel to the increased routing performance, the protocol overhead is increased. Thus, a scenario with more nodes is investigated in a next step. Therefore, the already mentioned test setup of Figure 3.8 is used and communication takes place via more than only one relay node. The parameter setting corresponds to the above identified settings, where the corresponding ad-hoc routing protocol shows the best performance. The following paragraphs describe the results of these tests and detailed information on one representative test run is presented. The results displayed in the tables include all test runs which have been carried out often enough to allow for this statistical analysis. Some of the results were also published in [13] and [1].

#### AODV

In this scenario, AODV is used with the modified parameter setting with FORCE_GRATUITOUS and LOCAL_REPAIR enabled and *treating node as neighbor after receiving 2 HELLO messages*. Again, the performance of AODV is very disappointing and a reliable rerouting is not possible for this test setup. Figure 3.17 shows the protocol behavior and at a test time shortly after 60 seconds, the communication was lost. As this result is not very convincing, an additional test is done with the default parameter settings. Also here, the results are quite disappointing as it is not possible to communicate via one hop (cf. Figure 3.17). To exclude environmental disturbances on the used radio frequency or malfunctions of the hardware, additional verification tests were done immediately after a failure of AODV was recognized. First, each node sends ping packets to its neighbor nodes without ad-hoc protocols activated to check the proper functionality of the hardware setup and the radio link. Second, DSR with default setting was used in the multi-hop setup. These verification tests were all successful. Thus, the problem of the poor performance of AODV is somehow caused by the used protocol implementation itself.

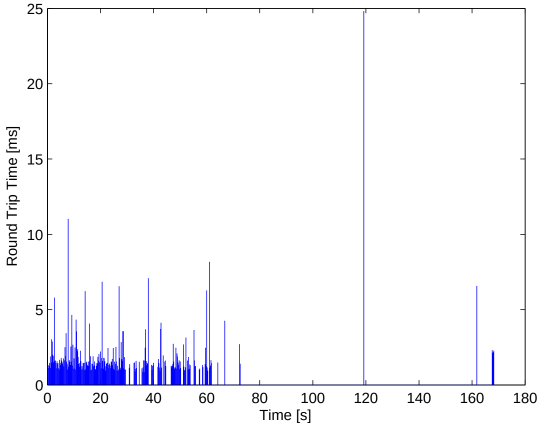


Figure 3.17: Behavior of AODV with FORCE_GRATUITOUS and LOCAL_REPAIR enabled and *treating node as neighbor after 2 HELLO messages* are received in the multi-hop scenario.

## OLSR

Figure 3.18 shows the OLSR protocol behavior in the multi-hop scenario with HELLO_INTERVAL= 0.5, HELLO_VALIDITY= 1.5, TC_INTERVAL= 2.5, and TC_VALIDITY= 5.0. At around 43.3 seconds of test time, a communication loss occurred with a duration of 5.0 seconds. Afterwards, a connection is set up via one hop. Until the next rerouting procedure is initiated at about 62.6 seconds, several packets were lost. Re-establishing a link via two hops took 13.0 seconds, and at about 90.4 seconds of test time, communication is set up via three relay nodes. While driving the robot back on the same way, the rerouting is much faster. For all OLSR experiments, rerouting has an average duration of 9.5 seconds, a minimum duration of 2.4 seconds, and a maximum duration of 21.6 seconds. Compared to the OLSR behavior with default settings, the performance is increased. Of course, a little bit more protocol overhead traffic is generated which does not matter in the present scenario as it is relatively small with respect to the number of participating nodes.

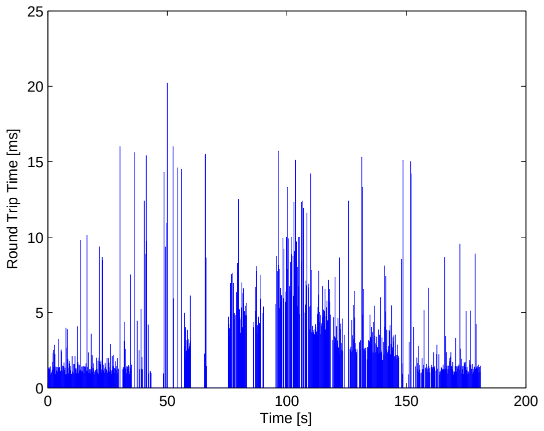


Figure 3.18: OLSR behavior with HELLO_INTERVAL= 0.5, HELLO_VALIDITY= 1.5, TC_INTERVAL= 2.5, and TC_VALIDITY= 5.0.

### BATMAN

The results for the behavior of the BATMAN protocol in the multi-hop scenario are shown in Figure 3.19. The first communication drop out and packet loss occurs at 40.8 seconds of test time with a very short duration of 0.8 seconds. A route is re-established via one relay node. At 54.0 seconds of test time a second hop is included after a rerouting procedure with a duration of 13.4 seconds. A communication via three relay nodes is established within 6.4 seconds after a communication loss at 73.0 seconds of test time. Again, the rerouting is much faster on the way back. For all these test runs, an average rerouting time of 9.35 seconds, a minimum duration of 0.8 seconds, and a maximum duration of 26.2 seconds is measured.

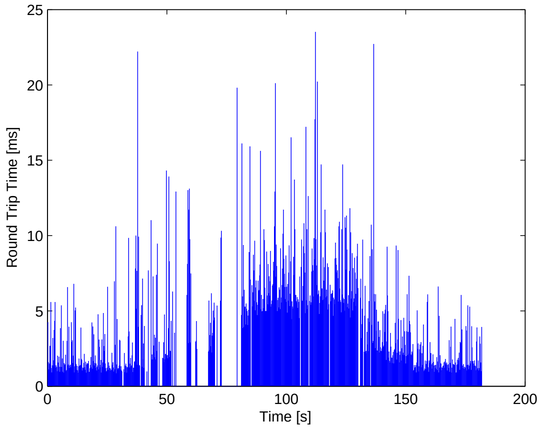


Figure 3.19: BATMAN behavior with ORIGINATOR_INTERVAL= 125ms.

## DSR

Figure 3.20 shows the results for the DSR protocol in the multi-hop scenario. The first rerouting is at 38.0 seconds of test time and it takes 2.0 seconds to re-establish the connection. The next communication drop out is at 66.6 seconds of test time and has a length of 11.8 seconds. The connection over three hops is established after a communication loss with a duration of 10.2 seconds starting at 114.0 seconds of test time. For all tests, an average rerouting duration of 12.14 seconds, a minimum of 1.0 seconds, and a maximum of 40.4 seconds is measured.

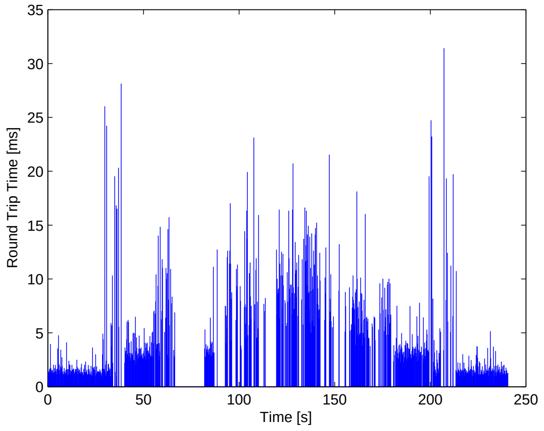


Figure 3.20: DSR round trip times in the multi-hop scenario.

### 3.2.5 Conclusion

Factors with major influence on the performance of the ad-hoc routing protocol are the protocol design philosophy and the protocol type. This issue should be kept in mind when the results of the previous sections are reflected. Thus, different protocols have an inherent behavior while handling the worst case scenario which is evaluated in this work. The difficulties of the current scenario have a stronger effect on AODV, OLSR, and BATMAN than on DSR (cf. [1]). The observed behavior of AODV might either be a consequence of the *Blacklisting* mechanism which is integrated into the protocol, or the result of weak points in the implementation. Furthermore, it is shown that protocol parameter tuning improves the performance of OLSR and BATMAN. Another key impact factor on the behavior of the complete wireless ad-hoc network is the packet size distribution of the traffic streams. The MERLIN robot generates a large number of relatively small packets. This is a big challenge for the underlying WLAN MAC layer with respect on multi-hop communication. In this case, the relay nodes are forced to use the MAC mechanism frequently while simultaneously receiving a lot of small packets via this half-duplex link. Thus, the collision probability of packets increases significantly which leads to a reduced throughput. As a result,

users are confronted with high packet loss ratios, high delays, and more communication losses due to long periods for route re-establishing in ad-hoc routing scenarios. To solve this problem, the MERLIN robot should summarize as many payload data as possible in larger packets which will immediately reduce the packet collision probability.
Another aspect of dealing with partially unstable links is the application of approaches for delay and disruption tolerant networks (DTN). Here, mechanisms still allow robustness in case of intermittent disconnections. [122] presents an architecture of an overlay network for asynchronous message forwarding in environments with limited end-to-end connectivity. [123] proposes the MaxProp protocol which is optimized for the routing of messages in a delay tolerant network.

Table 3.10: Summary of protocol tests.

| Protocol: | min. time: | avg. time: | max. time: |
|---|---|---|---|
| AODV | – | $-$s | $>$ 60s |
| OLSR | 2.4s | 9.5s | 21.6s |
| BATMAN | 0.8s | 9.35s | 26.2s |
| DSR | 12.14s | 1.0s | 40.4s |

## 3.3 Application: Teleoperation via Ad-Hoc Networks

This Section gives application examples of wireless ad-hoc networks which were realized in the frame of this work. Section 3.3.1 describes how a unmanned small size helicopter can be integrated into an IP based network, in order to allow teleoperation and data acquisition via this network while simultaneously providing a save operation of the helicopter. In Section 3.3.2 a MERLIN robot is teleoperated via a multi-hop network in order to accomplish a navigation and exploration task. While the robot is teleoperated, sensor and video data is transmitted back to the user via the ad-hoc network. During the mission, the robot moves in an outdoor area with different obstacles blocking the radio communication which induces a dynamic into the network topology. In this chapter, also useful traffic shaping mechanisms for transmitting a video stream in multi-hop networks is given.

### 3.3.1 Integrating a Small Size Helicopter into an IP based Ad-Hoc Network

#### System Design

One example using this approach is the helicopter of the University of Wuerzburg [15]. Onboard the helicopter, a state machine is taking care of the modes of operation and it is interacting with the status of the communication link (cf. Figure 3.21). While receiving commands, the helicopter is in normal operation modes and can directly be operated by the user. In case a communication link failure is detected, the state machine switches to a safe mode of operation - a stationary hovering without changing position and orientation. After re-establishing the communication link, the status of the helicopter and the control PC must be synchronized to provide valid data to the user. The helicopter control protocol presented in [14] is used in an environment which allows any to any communication between all network nodes like mobile robots, UAVs, or humans with a PC or PDA. Therefore, a WLAN running in ad-hoc mode is used. Each robot and UAV is equipped with a PC architecture and a standard TCP/IP and UDP/IP protocol stack. The helicopter flight control has a simple finite state machine (FSM) implemented which provides the features mentioned above. The following paragraph gives a short overview of this FSM onboard the helicopter. The FSM of the control software has five states the helicopter can use: *Not Initialized, Locked Hover, Unlocked Hover, Joystick Mode*, and *Task Mode* (see Figure 3.21). After starting the onboard software, the helicopter enters the *Not Initialized* state. In this state, no commands for the helicopter will be accepted due to a missing connection to a control PC. After the initialization procedure, the helicopter enters the *Locked Hover* mode and a connection to an available control station is established. In the *Locked Hover* mode, the helicopter keeps its position and altitude and waits for the *unlock-command* from the control PC to enter the *Unlocked Hover* mode. During the *Unlocked Hover* mode, the helicopter holds its pose and waits for commands to enter either the *Joystick Mode* or the *Task Mode*. If the helicopter is in *Joystick Mode, Task Mode*, or *Unlocked Hover* mode and the connection to the control PC is lost, it immediately switches to the *Locked Hover* mode.

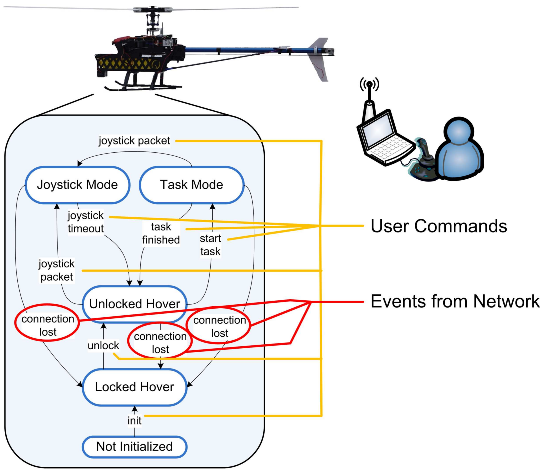


Figure 3.21: State machine of the helicopter interacting with the communication link.

### Autonomy

The application of autonomy mechanisms in an adequate approach to allow for teleoperation capabilities of complex systems at remote locations. According to the application scenario and the used system, autonomy features are used to prevent the system from damages, to bridge communication delays and losses, or to operate the system always in a defined state. [124] presents the design of a rover to explore the surface of Mars. Thereby, the rover is operated in unknown and unstructured terrain. The system which is described in [124] allows the teleoperation of the mobile system even with the long communication delays which are present for this environment (between 4 and 21 minutes for one-way communication). This is achieved by a partition of the system in three main components: tasks, macrocommands, and the scheduling of low level commands. [125] reviews requirements for autonomous operation of spacecrafts for deep space missions in a poorly known environment. In the context of [125], autonomy is understood by the capabilities of spacecrafts. Thus, it is focused on two important issues of this application: meeting the mission performance requirements for a specified period of time without external support and optimizing the mission product within the given constraints. According to [126], the main emphasizes of earth orbiting satellites with respect to autonomy are on failure detection and the corresponding recovery actions, reducing periods of degraded operation modes, and reacquisition of communication links to the ground station after disturbances. With respect to spacecraft operation on deep space missions, the above framework is extended to cope with situations which are typical for these missions. Here, usually large transmission delays are present, ground control interaction is not allowed during critical mission phases like fly-bys or atmospheric descents, the environment is unstructured and only partially know, human interactions at time critical situations are impossible, and there are limited capabilities of the spacecraft with respect to power, mass space, and the harsh environmental conditions. Thus, the fail-safe-mode reactions of earth orbiting spacecrafts are extended with nominal functions to be autonomous in order to allow for interplanetary missions. [125] also describes in details, how adaptive control concepts provide a solution for mission critical like descent and landing on a comet and drilling into the surface in order to collect samples from the comet. [127] presents Internet based ground operations for Mars lander and rover missions. Also here, autonomy features are implemented to allow for the remote operation of the mobile system via long distances. The scientific main contribution of [127] is on the so-called *web interface for telescience*. It is an integrated environment for operations at the central operations location. Also collaboration by geographically distributed scientists and public outreach is supported. This web interface can also be used as primary operations tool for the visualization of downlink data and the generation of command sequences for the robotic arm and the robotic arm camera which is mounted on the rover. The main components of this web interface system are a database, a server, and several clients. The database stores downlink data and uplink sequence information. The server is responsible for the communication between the database and the clients. All clients are connected to the system via the Internet. Of course, Internet based mission operations demand for security mechanisms. Therefore, the system uses SSL and Triple-DES-EDE3 encryption. The above presented implementations use different approaches and autonomy mechanisms to keep the remote systems in a defined operational status.

With respect to the presented implementation for the integration of a small size helicopter into an IP based wireless network, the on board autonomy mechanisms closely interact with the status of the communication channel. Basically, the on board functionalities of the helicopter provide stable flight conditions in each of the four states *joystick mode*, *task mode*, *unlocked hover*, and *locked hover*. In Figure 3.21, the points of interaction between the helicopter's state machine and the communication link are designated by *Events from Network*. In *joystick mode*, the on board system waits for simple commands for movements which allows also inexperienced users to fly the helicopter. The *task mode* allows the helicopter to be commanded for a complete task e.g. flying on a predefined route to reach given waypoints. With respect to these two modes, autonomy means the provision of low level control algorithms to allow for a set of basic flight patterns. If the system is in the *unlocked hover* or the *locked hover* mode, control algorithms keep the helicopter hovering safely at the current position. These points of interaction between communication link and the system trigger the activation of the *locked hover* mode. In this case, the communication loss is considered to be too long, or the packet losses are considered to be too high to allow a safe operation of the helicopter. A special application protocol for commanding the helicopter was developed at the University of Würzburg which allows for a secure communication between ground control stations and the helicopter and which is able to detect disturbances of the communication in time. Thus, shortfalls of communication are compensated by the implemented autonomy feartures.

### 3.3.2 Improving video transmission for Teleoperation

Video streams still play a major role in all applications where a mobile robot is controlled by humans. Hereby it makes no difference which level of teleoperation is applied – supervisory control, direct teleoperation or anything in between. The video is often the most important element to reach and maintain situational awareness and common ground between human and robots. Depending on the task of the human, different parameters of the video stream are important. In case of humans monitoring the robot or searching objects with the help of the robot, the resolution is one of the most important parameters. For direct teleoperation (possibly with assistance systems) the resolution is not the major performance criteria. Here, high frame rates, a constant frame inter-arrival time, and small constant delays are much more important for the performance of the human operating the machine remotely. If the video gets stuck while a human is steering a mobile robot, in most cases the human has to stop the robot until the frame rate recovers. With respect to the communication delays during teleoperation, the human teleoperator can adapt his behavior to a certain delay of the feedback. But if this delay is varying (changing inter-arrival times of the video) the operator is not able to compensate this anymore, which leads in most cases to wrong and imprecise steering commands. Dependent on the specific tasks, it has to be ensured that the parameters for the video stream are still at the required or possible level when the teleoperation is realized over an ad-hoc network. The focus of this section is set on the effective transmission of a video via a wireless ad-hoc network and some parts of the following issues are published in [9] and [10].

#### Shaping a video stream

In the above mentioned scenarios, the available throughput of a route via a wireless multi-hop network is a highly dynamic parameter which depends on many environmental influences and affects the quality of the application significantly. The throughput of a wireless node can be decreased due to different reasons. In case intermediate nodes of a route are also part of a route which has to transport other bandwidth intensive flows, the available bandwidth must be shared between all present data streams passing this node, which will reduce the available bandwidth for the video link. Furthermore, also a decreasing link quality will reduce the effective bandwidth and increase the packet loss probability. If the network is not reacting to traffic overload at a specific node, this will lead to unpredictable packet loss at this point, and delays at the different receivers are the consequence. For the teleoperation scenario the effect will be, that the video stream will get randomly stuck, due to the packet loss. Most probably, the operator will get confused and will stop the robot.

To set up the test scenario where a node is used for more than one bandwidth intensive traffic flow, four nodes are used (cf. Figure 3.22). All nodes are located such that they are in direct communication range. During the tests, defined additional UDP traffic is generated between node 3 and node 4 while the investigated video stream is transmitted via UDP from the mobile robot to the user's PC via node 3. The generated UDP traffic is used to reach certain load levels at intermediate node 3. As in this scenario, node 3 and node 4 are in communication range to all other nodes, this setup will also cause interferences at the physical layer (cf. 3.1.2). To provide best repeatability of the tests, all nodes are stationary. Only the additional traffic between node 3 and node 4 will be varied according to a defined profile. Measured categories are the packet loss and the packet inter-arrival times.

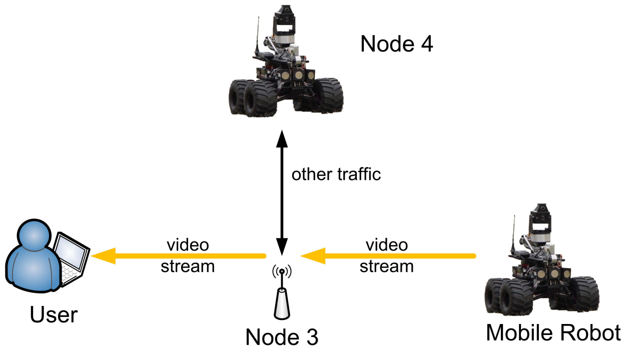


Figure 3.22: The test setup for multiple data streams at one node.

These categories are measured while the amount of additionally generated traffic is increased. As reference test, video transmissions of constant target quality are used and compared to the packet loss of the transmission with adaptive quality.

The proposed mechanism mainly consists of two parts: the *network feedback*, and the *adaptive adjustment of the video quality*. The mechanism is used for a simple admission control of the video source and intends to provide the best possible video image quality considering the current state of the link. The objective is an efficient use of the available bandwidth without overloading any node of the route with video traffic. Thus, it is not used to increase the link quality directly but uses the available resources most efficient and reliable for the operators' video stream.

### Network Feedback

The network feedback is responsible to transmit the status of a node to the video source. Therefore, nodes of the network host a small client program at the application layer. This client application is listening in promiscuous mode at layer 3 of the ISO/OSI model (IP-layer) and measures the utilization of the wireless link. All kinds of traffic are monitored: incoming and outgoing packets, packets for forwarding, and packets with other nodes in range as destination – basically all traffic which causes the radio link of this node to be busy. The network feedback client sends UDP packets with an adjustable frequency (in the test setup 10 Hz) and 8 bytes as payload to the video-source if it is a used hop in the video stream route between video-source and receiving node. This payload is used to indicate the status of the corresponding node, either *normal operation* or *overload situation*. In the beginning, each node is in the *normal operation* mode. As soon as a certain utilization of the supported bandwidth is exceeded, the status of this node switches to *overload situation*. Important parameters for the network feedback clients are the feedback frequency $f$ and the threshold for status determination $d$. In case $f$ is too high, too much feedback traffic is generated which degrades the performance of the network, as too many small packets with a high sending frequency will have a very bad effect on 802.11b WLAN and will significantly decrease the throughput. Thus, the generated feedback traffic should be limited depending on the interpretation rate of the video adjustment mechanism and the selected load window for the wireless nodes. Often it is also not necessary to run a feedback client on each network node. For setting parameter $d$, it should be considered, that $d$ specifies the percentage of the nominal bandwidth (e.g. for 802.11b this would be 11 Mbit/sec) which can be used without switching to the *overload situation* state. The feedback clients measures packets on layer 3, where the maximum available bandwidth corresponds to the *goodput* of the wireless link which is about 75% of the nominal link bandwidth (e.g. for 802.11b this would be 75% of 11 Mbit/sec).

As the proposed mechanism is used within a network where a link failure can occur at any time, the measurement and signaling mechanism must be active. Thus, link failures and link re-establishing can be monitored reliably. As the mechanism for video quality adaptation performs best with a feedback frequency of $f = 10$ Hz (according to the presented scenario), the generated measurement traffic has a bandwidth of less than 0.003 Mbit/sec per measurement node. To set parameter $d$, the *goodput* of about 7 to 7.5 Mbit/sec (for an 11 Mbit/sec WLAN link) must be considered. In order to allow a reaction on potential overload situations while providing the user a video stream with a bandwidth of 1 to 1.5 Mbit/sec for the best quality, $d$ is set to 50.

### Adaptive Video Quality

The Video Quality in the presented system is adapted according to the current state of the ad-hoc route for the video transfer. The adaption mechanism receives all status packages from the nodes between two received frames from the image source, interprets these packages and selects the quality for the next frames with a combination of previous status data and the current state. To reduce oscillating behavior in quality switching near the selected load limit of the nodes, a kind of inertia mechanism for the adaptation process is integrated. The implemented inertia mechanism guarantees not to change the image quality whenever a status of a node changes. It is possible to set a certain number (cf. Algorithm 1 in [9], min/max of *inertia_counter*) of receiving same successive route load states until the quality is changed. In the current test setup, four different video qualities are used at a frame rate of 11 frames per second each. Table 3.11 shows the average size of one image for the corresponding image quality level. A higher

Table 3.11: Average size of one image per quality level.

| Quality | minimum | low | medium | high |
|---|---|---|---|---|
| Size (kbyte) | 15 | 26 | 34 | 47 |

number of different quality scales would also be possible. In the current test setup a minimum of $-3$ and a maximum of 3 are selected for the *inertia_counter*. With this value the mechanism reacts in the worst case after six frames with subsequent overload states and in average after three frames. This keeps the load caused by the video traffic on the different nodes in a certain defined window around the selected threshold for overload state. In combination with parameter $d$ of the above described feedback mechanism, the quality adjustment intervenes as soon as a node exceeds a radio link utilization of more than approx. 78% ($\approx$ 50% of nominal bandwidth). This prevents the node from reaching a utilization of 100% of the available maximum throughput which would result in a high packet loss rate due to an increasing number of packet collisions.

The proposed mechanism is tested in a real outdoor environment with a wireless ad-hoc network of four nodes. One is the PC of the operator, one is an Outdoor MERLIN, and two intermediate nodes are MERLIN robots (indoor version). Figure 3.23 shows the detailed system setup. All MERLIN robots have a C167 microcontroller for low-level operations and sensor data processing, as well as a PC-104 for more complex and computationally more intensive tasks. The PC-104 uses a Linux operating system and all nodes are equipped with 802.11b standard WLAN equipment (Atheros chip).

To grab the video from an analog camera (approx. 65 degree field of view) an Axis video server is used. It can grab the video from up to four cameras with a resolution of 768x576 pixels. Dependent on the configuration and connected clients, a frame rate of up to 25 images per second can be provided either as MJPEG or MPEG4 over a TCP/IP connection. For the described tests the PC-104 is connected over a cross-link cable to the Ethernet interface of the video server. As nothing else is connected to this Ethernet interface of the PC-104 it can be exclusively used for the video traffic. For the presented tests four MJPEG video streams with full resolution and four different compression rates are provided by the onboard PC. MJPEG as video compression was selected, as MPEG4 compression takes a significant longer time on the Axis server what causes a significant delay in the video stream. Also a loss of a packet during transmission of MPEG4 streams to the robot might lead to longer set of distorted images because compared to MJPEG not all frames of the stream contain the full image information needed. In case of the investigated scenario, the MJPEG frames are transmitted via UDP protocol from the onboard PC to the operator's PC.

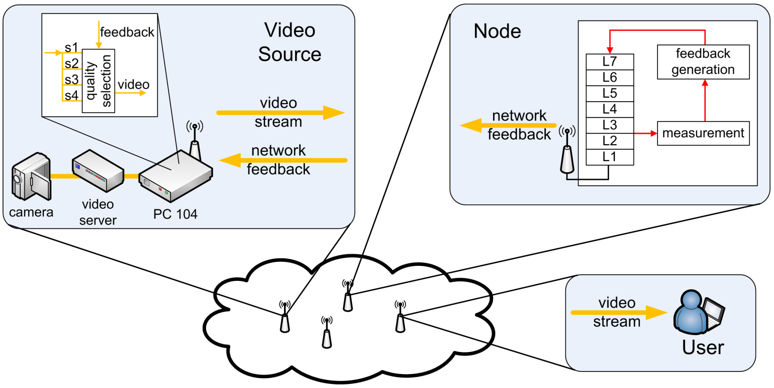


Figure 3.23: Test system setup.

### Network Feedback in Overload Situations

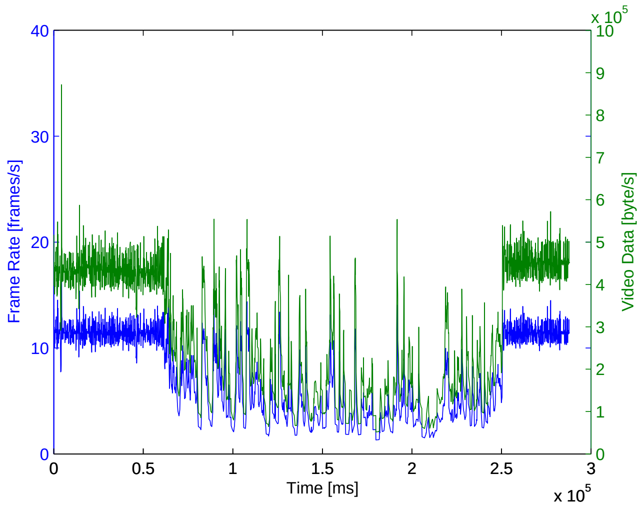
To characterize the behavior of the network in a overload situation, in a first step, a reference scenario was set up and measured. Therefore, no network feedback mechanism is used, and a mobile robot generates a video stream which is sent to the PC of the operator. Between node 4 and node 3, additional traffic is generated during the different test phases according to Table 3.12 to reach a defined load at intermediate node 3.

Table 3.12: Generated additional traffic.

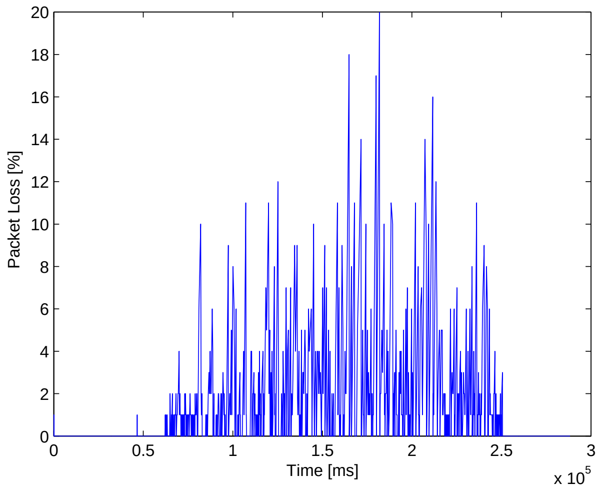
| Phase | generated additional traffic (Mbit/sec) |
|---|---|
| 1 | 0 |
| 2 | 3,2 |
| 3 | 4 |
| 4 | 4,8 |
| 5 | 5,6 |
| 6 | 6,4 |
| 7 | 7,2 |
| 8 | 8 |
| 9 | 8,8 |

The results of this reference test are shown in Figure 3.24a. The x-axis shows the test time in milliseconds. The left y-axis describes the received frame rate in frames per second (fps) and the right y-axis displays the received video data rate in bytes per second (bps) at the receiving node (operator's PC). The test started with no additional traffic being generated. Successively, more and more additional traffic is generated by switching to the next phase each 20 seconds according to Table 3.12. After 200 seconds of test time, the additionally generated traffic is reduced by switching back one phase each 10 seconds. In the beginning of the test – during phase 1 up to the end of phase 3 – the received frame rate is about 11 fps. After switching to phase 4 at about 60 seconds, the received video frame rate decreases significantly. The received frame rate between 100 and 200 seconds drops to 2 – 3 fps while node 3 is overloaded. After the additionally generated traffic is reduced, the received frame rate recovered to 11 fps. Increasing the additional traffic forces node 3 to an overload situation. As the bandwidth used by the video stream cannot be adapted to the new situation, a packet loss of the video data is inevitable which is shown in Figure 3.24b. The y-axis shows the number of lost packets vs. the test time on the x-axis.

Another measured category is the frame inter-arrival time of the video stream. This is a quite sensitive aspect, as a large jitter (variance of the frame inter-arrival



(a) Framerate and traffic.



(b) Packet loss.

Figure 3.24: Behavior in an overload situation without network feedback.

time) is very irritating for the operator due to a very unsteady motion of the video image. Without additional traffic, the frame inter-arrival time is smaller than 100 ms with a variance close to 0 (cf. Figure 3.25) what corresponds to the average frame rate of 11 fps. After 60 seconds and an additionally generated traffic of 4.8 Mbit/sec, the frame inter arrival time increases to more than 400 ms with a variance of more than 10000 which indicates an unacceptable video for the operator.

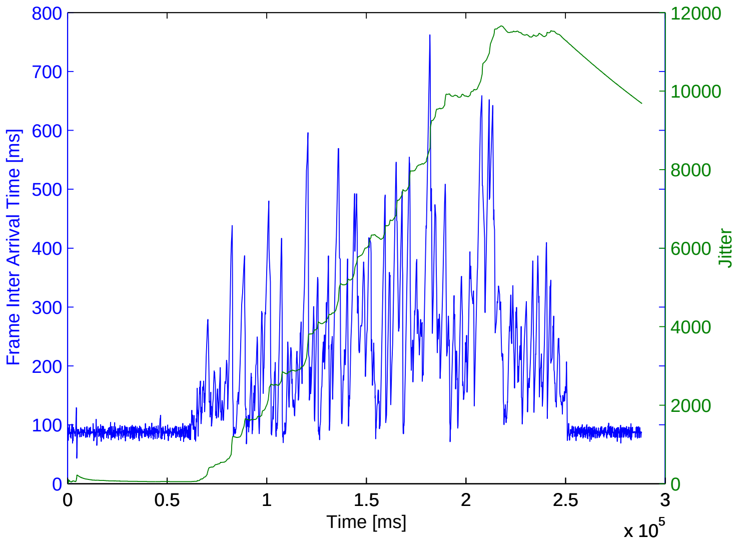
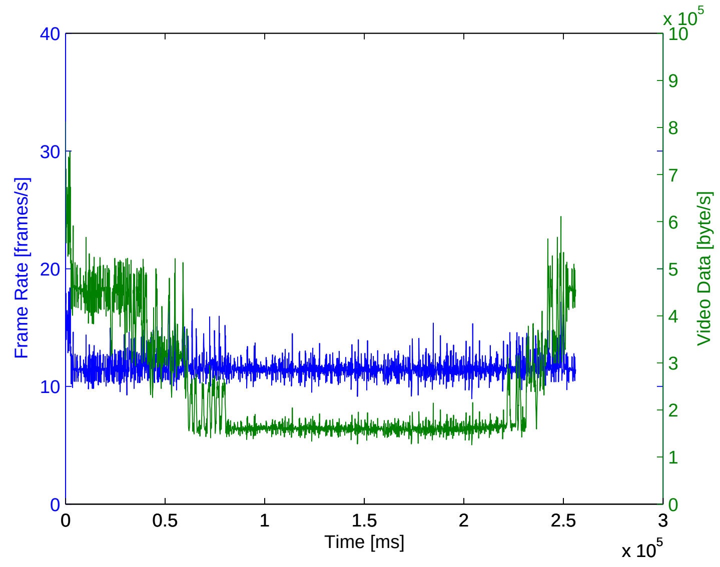
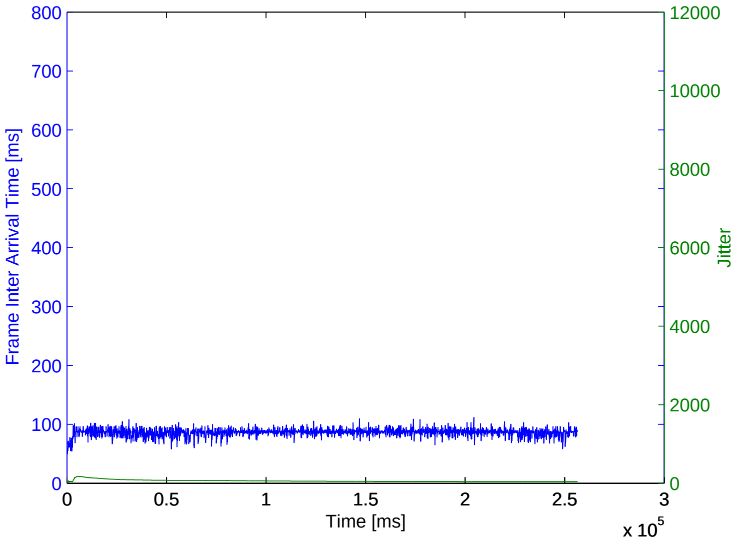


Figure 3.25: Frame inter arrival time without network feedback.

The same test setup is used again – now with the network feedback and adaptive quality mechanism, which should improve the observed behavior (cf. Figure 3.23). In Figure 3.26a, the frame rate and the video data rate is shown while using an adaptive video quality together with the network feedback mechanism. In the beginning, without additional traffic, the mobile robot generates a video stream of about 450000 bytes/sec. During the test, the additionally generated traffic is increased similar to the test described above. The implemented mechanism takes care that the video source reduces its generated video traffic to about 300000 as soon as phase 3 (with an additional load of 4 Mbit/sec) is entered. Increasing the additional load at node 3 to more than 4.8 Mbit/sec results again in a reduction of the video traffic (180000 bytes/sec). During the complete test run, the frame rate stays almost constantly at 11 fps as the adaptive video bandwidth reduction avoids the loss of video traffic. Also the frame inter arrival time stays constantly below 100 ms with a jitter of almost 0 (cf. Figure 3.26b).



(a) Framerate and traffic.



(b) Frame inter arrival time.

Figure 3.26: Behavior in an overload situation with network feedback.

#### Network Feedback for the MERLIN Robot System

Now, the described mechanism is used in an ad-hoc network. It has to take care that the video stream is delivered via different transmission relay nodes and dynamic routes to the operator with a suitable quality. The scenario is set up in a way, such that several nodes of the network are included into the route between the operator's PC and the teleoperated mobile robot. Figure 3.27 shows how the different mobile robots acting as potential communication nodes were placed in a real outdoor scenario for the described tests. The three potential communication nodes are positioned around a small hill in a way, that each node covers a certain reception area overlapping with the reception areas of the two nearest neighbors. The small hill in the center prevents direct communication between the mobile robot and the control PC of the user when line of sight is lost. So a direct communication between the operator station and a node behind the hill (either mobile robot 2 or the teleoperated mobile robot) and between mobile robot 1 and 3 is not possible anymore. To guarantee these test constraints also the transmit power for each node was reduced to 10mW additionally. Data exchange between the MERLIN robot and the control system MRCS is implemented via UDP and TCP. Sensor data is transmitted using UDP, and the video stream is sent from the mobile robot to the control PC via TCP. For the test, the teleoperated mobile robot is controlled via joystick along the path as shown in Figure 3.27. The selected path and environment requires that the routing of the robot's communication link is changed according to the current position of the moving robot and the respective possible links to other communication nodes in the scenario. As soon as the teleoperated mobile robot enters the communication shadow of the small hill and no direct link is possible anymore, it has to communicate via one or more of the other mobile robots to establish a communication link to the operator station. In order to achieve this dynamic routing and to keep keep the communication link usable, ad-hoc routing protocols are applied and combined with the network feedback concept shown this section. In the previous section, ad-hoc routing protocols for mobile robot teleoperation are investigated. The required routing protocol parameter tuning is described in [13] and [11]. Based on these results the BATMAN protocol is used for the current scenarios with tuned parameter setting in order to provide the required performance. In the current scenario, only the mobile node (here: teleoperated mobile robot) generates a network feedback. There is no need to use the network feedback of all the relay nodes, as all relay nodes in the scenario are stationary (link quality is not varying significantly) and the relay nodes do not create additional traffic besides the video data of the robot and the command and sensor data exchange between operator and mobile robot. While the mobile robot is moving, the network feedback mecha-

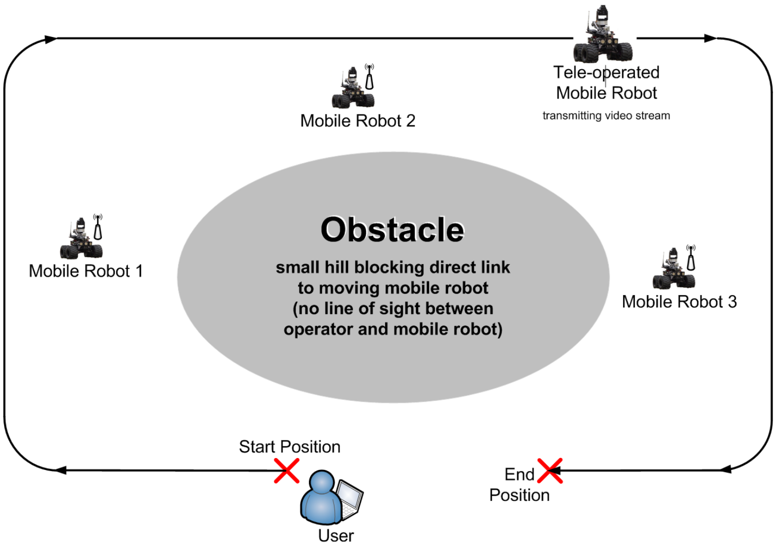

Tele-operated Mobile Robot
transmitting video stream
Mobile Robot 2
Obstacle
small hill blocking direct link to moving mobile robot (no line of sight between operator and mobile robot)
Mobile Robot 1
Mobile Robot 3
Start Position
End Position
User

Figure 3.27: Setup of navigation test scenario with the MERLIN system.

nism is continuously monitoring the communication links and provides input for the video adaptation mechanism. With respect to the parameter settings, it has to be taken into account that several network topologies may occur which will affect the behavior of the wireless network significantly. As the communication via multiple hops implies variable bandwidth limitations e.g. due to hidden nodes, setting of parameter $d$ (threshold for maximum link utilization) of the network feedback mechanism must be done carefully. In this work, the video source is the main traffic source of the entire network. Due to the steady flow of delay sensitive packets, a conservative setting of $d = 25\%$ is chosen as this also considers the half duplex characteristics of the WLAN links and hidden node situations of the scenario.

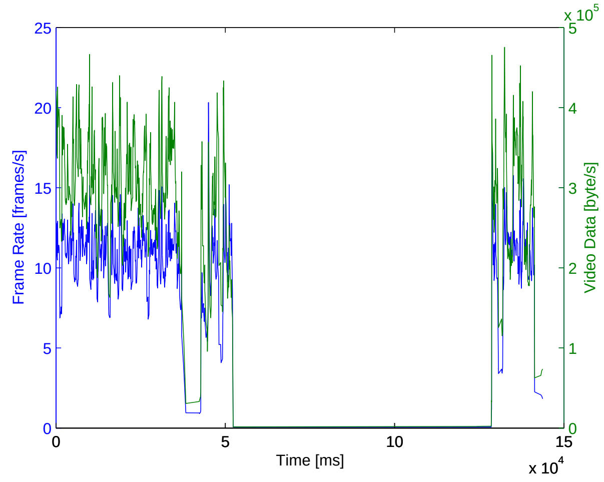
In Section 3.1.4, difficulties of a proper setup and evaluation of scenarios with wireless ad-hoc networks are already discussed. The biggest challenge is the design of a scenario which provides a base for comparable test runs. The used radio link might be disturbed by many external influences. Besides, parameters like link quality or signal strength which can be an indicator for external disturbances, there might still be an external disturbance which cannot be detected and characterized so easily. Thus, the influence of existing error sources on the test runs must be minimized or at least considered in the evaluation. In general, two methods are available for the evaluation. The first possibility is, that many test are performed in the different scenarios with the setup to be investigated. Then the settings are changed and again many test runs are performed in the same scenarios as before. Thus, a trend between the two present behaviors might be observed. The second possibility is the repetition of many test runs in one meaningful scenario. After tuning the parameters, the test runs must be repeated with the new settings. Of course, in both cases all tests runs must be performed contemporary in order to minimize environmental changes (e.g. weather). These two methods do not directly allow a quantitative evaluation, but a relative evaluation and the determination of a trend of the behavior due to changes in the parameter settings of the observed mechanisms are possible and often enough to analyze the investigated system. This work uses the second method. For this work numerous single test runs were performed. During a test run, the mobile robot is teleoperated via wireless ad-hoc network while driving around the hill. To allow the operator for proper teleoperation, a video stream and sensor data are transmitted from the mobile robot to the operator. As the small hill blocks the line-of-sight communication between operator and mobile robot, several communication relay nodes will be included on-demand. During a test run, the round trip time of packets between operator and mobile robot, the frame inter arrival time, the frame rate, as well as the packet loss are measured.

### Results

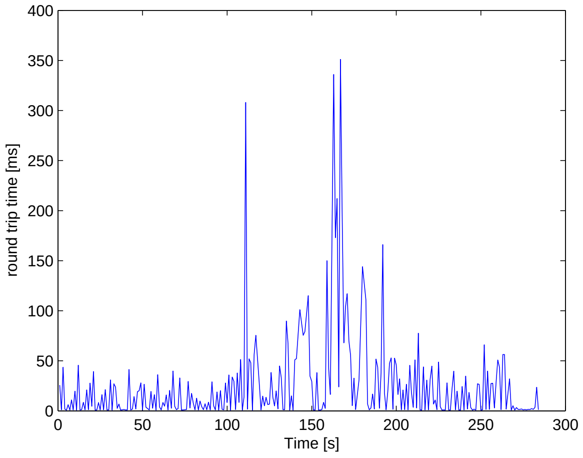
While a test run is performed, several data flows are present inside the network. The operator sends control and command packets from the control PC to the mobile robot, whereas sensor data is transmitted in the opposite direction. The most bandwidth consuming data stream is the video image. All data is transmitted via the UDP protocol. In the investigated scenario, no active measurement and signaling traffic of the feedback mechanism is generated as the feedback client program is running on the mobile robot which also acts as video source. Thus, the feedback is directly sent to the quality adaptation mechanism without stressing the wireless communication. In the following paragraphs the results of one representative test run are explained exemplarily – of course much more tests were performed in order to draw the presented conclusions based on a large amount of data which allows for a statistical evaluation.

The video stream behavior of a representative test run without network feedback is displayed in Figure 3.28a. The left y-axis of Figure 3.28a shows the video frame rate in frames per second and the right y-axis the video data in bytes. On the x-axis, the experiment time is plotted in milliseconds. In the beginning of the test, the operator is provided with a video frame rate of more than 10 frames per second. At around 40 seconds of test time, a short break down of the video link for about 5 seconds is detected. A later analysis turned out, that at this time, the direct line-of-sight communication between operator PC and mobile robot was lost and a new route was set up via a relay node. Shortly after 50 seconds the video transmission failed again and could not be reestablished. A later analysis showed that an additional communication relay node was included, and the route is established from the operator's PC via two relay nodes to the mobile robot. Due to the half duplex characteristics of the link, the available bandwidth decreased significantly what lead to a complete overload of the network link by the video traffic. Thus, the video link, as well as the command link broke down and the robot stopped. The observed link failure for such a long time results also from the ad-hoc routing protocol being not able to maintain the topology changes as also the signaling traffic for routing updates was lost during the overload phase. This behavior now should be avoided by the use of the network feedback mechanism.

In Figure 3.28b the round trip time (rtt) of ping packets between operator PC and mobile robot is plotted on the y-axis in milliseconds while the test is performed with active network feedback mechanism. It can be seen that at approximately 110 seconds test time the rtt increases and shows relatively high peaks at about 160 seconds of experiment time. After 200 seconds, the rtt decreases again to values below 60 milliseconds. The comparison with the routing tables for these times show, that shortly after 100 seconds experiment time, the first relay node was included into the communication link and at about 140 seconds the



(a) Frame rate and received amount of video data at the receiver.



(b) Round trip times.

Figure 3.28: Examples for multi-hop communication without network feedback.

second relay node joined. In parallel, the packet loss is plotted for the same test run in Figure 3.29. Between 140 seconds and 240 seconds of experiment time the graph shows several short periods in which packets are lost. The majority of these packet losses occur due to the two rerouting procedures while driving the robot in areas without direct line-of-sight communication with the formerly associated communication relay node. Of course, also some of the packets might be lost due to a very high link load while communicating via multiple hops. The following figures will give more details on this. Figure 3.30a shows the frame inter-arrival

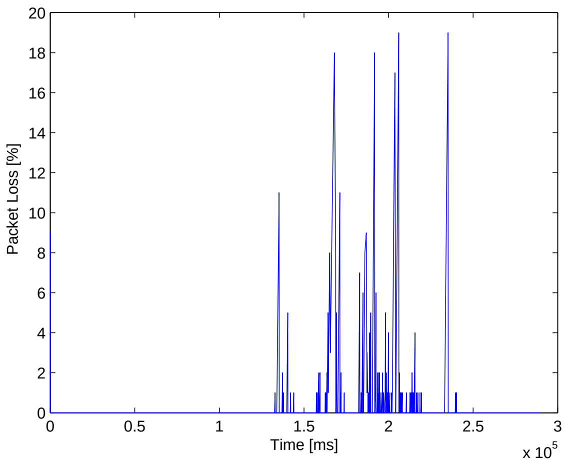
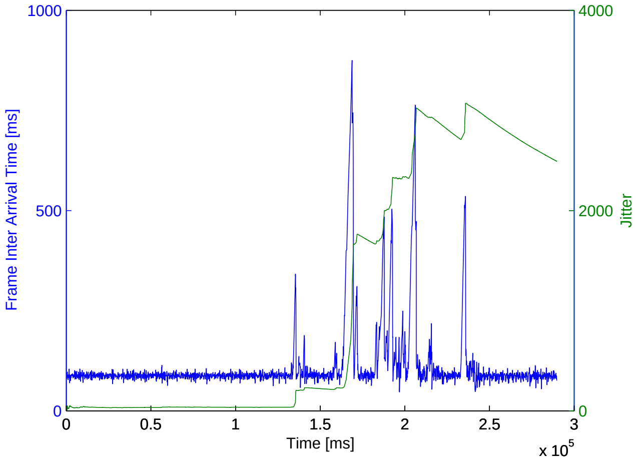
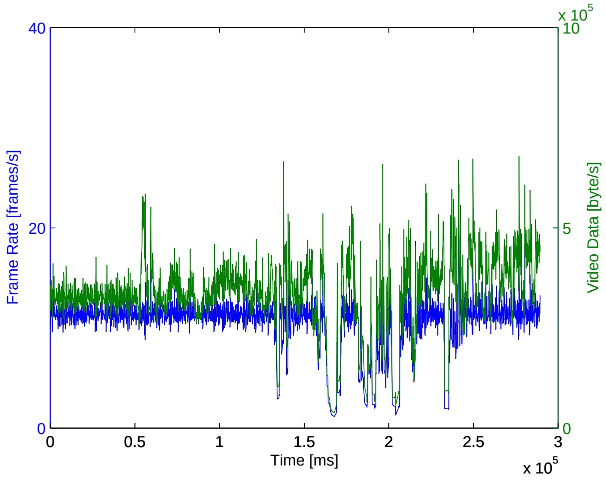


Figure 3.29: Example of packet loss for test with the MERLIN system.

time and the variance of the frame inter-arrival time (jitter) at the receiver and Figure 3.30b displays the frame rate and amount of transmitted video data at the operator’s PC while a representative test is performed with active network feedback mechanism. From the beginning of the test run until approximately 130 seconds of the experiment time the graph of Figure 3.30a oscillates around a value of about 90 milliseconds for the frame inter-arrival time. At 130 s, 160 s, and between 180 and 200 s of experiment time several peaks of up to 850 milliseconds are detected. This observation corresponds to the graph of Figure 3.30b, where the received frame rate of the video is displayed on the left y-axis and the received amount of video data is displayed on the right y-axis. From the beginning of the test until about 160 seconds of test time the video frame rate keeps constant at 11 frames per second while the transmitted amount of data varies during the rerouting at 130 seconds of experiment time. Between 160 and 240 seconds of test time, the frame rate varies due to changes in the link quality during this period of time (unstable links because of handovers). In contrast to the previous tests without network feedback, the video stream did not break down. The network feedback mechanism takes care that the transmitted video data traffic does not exceed the capabilities of the complete link - also while communicating via several relay nodes. The stable load condition inside the network also allows the ad-hoc routing protocol to reconfigure the routing tables of the network nodes in time, as the video link capacity is limited autonomously before the rerouting procedure is initiated in order to reserve bandwidth for signaling and maintenance traffic. This is possible as the proposed mechanism additionally monitors the really available link capacity of WLAN which is reduced as the link quality decreases. Thus, the network feedback mechanism prevents the network from being overloaded before a handover is started due to limiting the video traffic also in case of a low link quality to the currently associated network node.



(a) Frame inter-arrival time and jitter of video data at the receiver.



(b) Frame rate and received amount of video data at the receiver.

Figure 3.30: Examples of activated network feedback mechanism for test with the MERLIN system.

### Network Feedback for Player Framework and Pioneer Robot

In order to verify the proposed approach with other teleoperation system which are using different data traffic characteristics, also tests with the Player framework and a Pioneer 3AT[17] robot are done. Therefore, again a navigation task is given to the user and the mobile robot has to be teleoperated via a multi hop network. The ad-hoc network consists of three stationary nodes, one mobile robot (see Figure 3.31), and the ad-hoc capabilities are again provided with the BATMAN routing protocol. The tests are set up in an indoor environment and the robot has to be navigated through rooms and corridors to a given waypoint and then, back to the starting position and tests are carried out by several test candidates (users). Each user has to accomplish two test runs, one without traffic shaping mechanism and one with traffic shaping enabled. For each of these test runs, the same user interface is used and command data, as well as sensor data is transmitted between Player and the user interface via TCP and for the traffic shaping setups, the video is transmitted via UDP. The traffic mechanism is only applied on the video data which is sufficient as this type of data uses the largest amount of the totally used bandwidth (see Figures 3.32 and 3.33). The effects of the traffic shaping mechanism are now comprised by measuring the time which is required by the users for accomplishing the task. In addition, it is also monitored

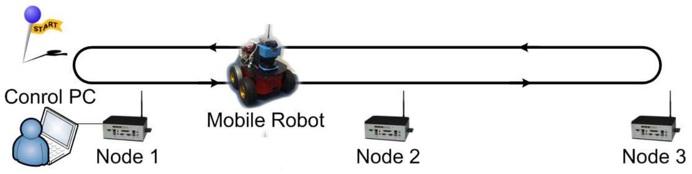


Figure 3.31: Setup for tests with Player and Pioneer 3AT system.

how often users had to stop the mobile robot due to disturbances in the network connection. The measurement data shown in Figures 3.32 and 3.33 is acquired at node 1 which is directly connected to the control PC. While the mobile robot is navigated to the given waypoint, the ad-hoc network includes also node 2 and node 3 into the route between mobile robot and control PC on demand .

[17]http://www.activrobots.com/ (09.02.2010)

### Results

Figure 3.32 shows the bandwidth usage for the test without traffic shaping. Here, the sensor data stream, the video stream, and the control commands are transmitted with TCP and no additional optimization of the usage of the communication link is activated. As soon as multi-hop communication is initiated, the throughput is reduced by the flow control of the used protocol stack (this can be seen at approx. 60 seconds of test time in Figure 3.32). At around 80 seconds of test time, the generated traffic exceeds the capacity of the ad-hoc network and the throughput is tremendously reduced. These negative effects of packet losses and collisions are directly tangible for the user for about 30 seconds and reduce the teleoperation capabilities significantly. The user has to stop the robot several times in order to avoid collisions with obstacles. At 120 seconds of test time, the throughput recovers and teleoperation is possible again. Of course, also situations can occur where the generated traffic together with the effects of packet losses lead to a situation with a complete and unrecoverable loss of communication. Usually, re-establishing the communication link is possible due to a timeout of the video stream which stops video data transmission automatically. Thus, the video stream can be reconnected manually by the user as soon as the reduced network load allowed the sensor and command data communication to recover.

Figure 3.33 shows now the test with traffic shaping enabled and it can be seen clearly that the shaping mechanism prevents a total overload of the network capacity. As a result the number and the length of the periods where the user has to stop the mobile robot is reduced. Comparing all test runs of both test setups, the result is as follows. The average duration of all tests without traffic shaping was 3.04 minutes which was longer than the duration of the tests with traffic shaping (average duration of 2.75 minutes). It is also observed, that the robot has to be stopped much more often when being teleoperated without traffic shaping (avg. 3 stops). With traffic shaping activated, the average number of necessary stops is reduced to 1.2. In summary, the positive effects of the traffic shaping mechanism are clearly shown and the occurrence and duration of periods with complete and unrecoverable loss of communication are reduced.

On one side, this mechanism has also a small drawback as the quality of some video images is reduced. As long as enough network capacity is available, only some single images of the complete video are injected with a lower quality. Of course, in situations where the network load is very high with respect to the avail-

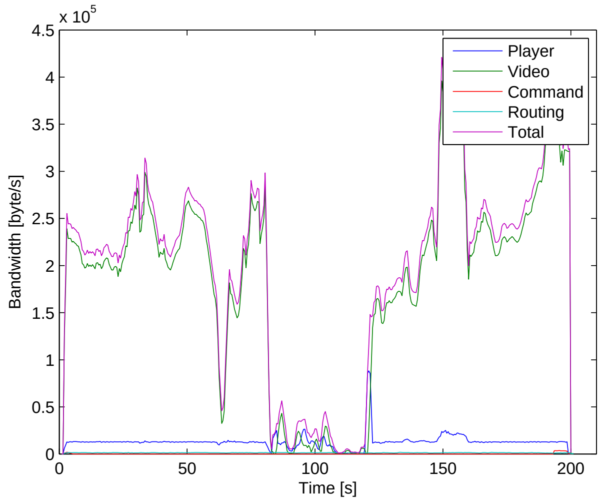


Figure 3.32: Example of bandwidth utilization of the Player and Pioneer 3AT system without traffic shaping.

able capacity, more and more video images are transmitted with a lower quality, or even the quality of the complete video stream is reduced. Nevertheless, also the lowest image quality still allows for a seamless mobile robot teleoperation. On the other side, this mechanism can be used for allowing teleoperation capabilities with onboard video data in multi-hop networks where standard systems cause a tremendous network overload and as a consequence, a communication loss. In addition, also the required time for task completion is reduced by approximately 9%, and the number of required stops is even reduced by 40%. These advantages preponderate the sometimes lower image quality.

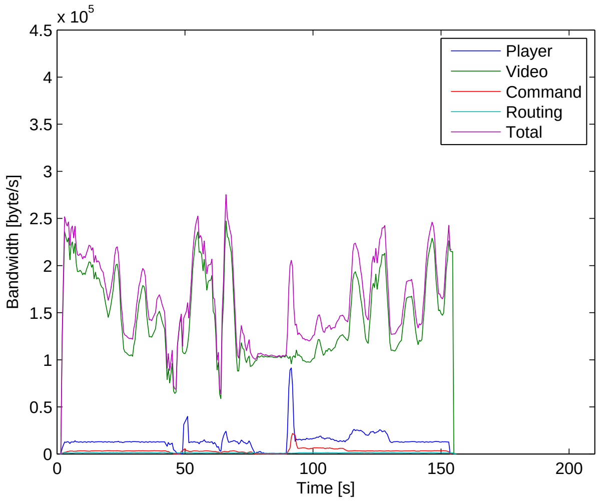


Figure 3.33: Example of bandwidth utilization of the Player and Pioneer 3AT system with traffic shaping.

### 3.3.3 Conclusion

This section describes several successful applications of mobile robot tele-operation over ad-hoc networks. Using the example of the small size helicopter of the University of Würzburg (cf. Section 3.3.1), the UAV is integrated into an IP based ad-hoc network. Here, also the system design of the UAV allows for different operation modes, each with different requirements for the communication link. Thus, this system can also work when it is used in only partially connected networks. The traffic shaping mechanism of Section 3.3.2 shows very good results in two different situations. In case of an overload situation at a network node, the video stream is shaped in a special in order to avoid or delay connection losses. The same mechanism is also used in a wireless network in combination with multi-hop communication. The test evaluations of Section 3.3.2 show, that teleoperation capabilities of the operator are improved, as the jitter of the video stream data packets is minimized and the video could be transmitted with a constant frame rate. It is shown, that the parameter tuning proposed in Section 3.2 in combination with the mechanisms described in Section 3.3.2 allows for a significant improvement of mobile robot teleoperation via wireless ad-hoc networks. Test evaluations also proved, that the assigned navigation tasks can be accomplished significantly faster when the traffic shaping mechanism is used. Also the number of required stops of the mobile robot is reduced.

## 3.4 Discussion of the Results

This second part of the thesis is focused on setting up wireless ad-hoc networks of mobile robots. The performance of four well known ad-hoc routing protocols (AODV, OLSR, DSR, BATMAN) is analyzed in a scenario which is typical for current applications of networked mobile robots. This scenario is a worst case setup for the routing protocols with respect to redundant routes and unstable links. By the example of this worst case setup, the limits of the protocols' functionalities are shown when the routing protocols are used with default parameter settings (cf. Section 3.2.2). Of course, these default parameter settings are optimized for communication networks with more nodes and a higher grade of meshing than in the present worst case topology, and thus, these ad-hoc routing protocols have to be applied carefully at the presented networked robotic scenario. This topology is also used for a study on parameter tuning (see Section 3.2.3) with the objective of increasing the performance and robustness of the four analyzed ad-hoc routing protocols. The relevant parameters are identified and measurement categories like the round trip times, the packet loss, and the duration of the rerouting procedure are used for this characterization. It is shown, that an appropriate parameter setting improves the protocol performance significantly (cf. Table 3.10) in the investigated worst case scenario. This successful parameter tuning is also applied in a setup which gives a detailed analysis of the ad-hoc routing protocol behavior in multi-hop communication scenarios. The tests also showed some disadvantages of some of the analyzed protocols' working principles as for example the *Blacklisting* algorithm for unstable links of AODV. It is also demonstrated by the example of the BATMAN routing protocol, how parameter tuning helps to make a protocol working reliable also in this type of network topology. In addition to this protocol analysis, all results are applied to different example applications for networked mobile robots. It is shown, how the command and data links of a miniature UAV can be integrated into an IP based ad-hoc network. Here, also local autonomy features are used to keep the miniature helicopter always in safe operating conditions - also in case the communication link fails completely for a longer period of time (see Section 3.3.1). This implemented application example demonstrates how shortfalls of the communication link like communication losses or packet losses can be compensated by autonomy mechanisms. In Section 3.3.2 a wireless ad-hoc network of mobile robots is used for a navigation and exploration task is shown. Here, also a mechanism for adaptive video transmission based on network feedback is presented. This mechanism shapes the video traffic by adaptively adjusting the video quality based on the current load and link status of the network. Thus, the quality of the teleoperation, and consequently the capabilities of the remote operator, are increased. This chapter shows how currently existing ad-hoc routing protocols can be parameterized in order to be successfully applied in networked robot scenarios and how the performance of the complete telerobotic system can be improved by the integration of communication into the telematics system (e.g. additionally implementing traffic shaping mechanisms) and local autonomy features. These achievements are also proven in real hardware tests of different systems of mobile robots.

# 4 Mobile Robot Teleoperation using 3G Telecommunication Technologies

The upcoming high-bandwidth networks for mobile phones and mobile Internet like Universal Mobile Telecommunication System (UMTS) offer a new potential technology for mobile robot teleoperation. Up to now, the coverage of these networks has increased in a way that at least all bigger cities have access to broadband networks. This everywhere availability in large areas is a major advantage for each telematic application compared to a solution where infrastructure initially has to be built up, and where maintenance effort is necessary. Besides the operation of mobile robots, this includes applications like tele-maintenance, tele-support, or tele-monitoring of industrial facilities. These applications in general demand also these high-bandwidth communication links. Sometimes, an additional own infrastructure as proposed in Chapter 3 might be not be necessary, feasible, or possible, and thus, the high-bandwidth telephone networks provide therefore an interesting alternative. The design of these networks in a way that they provide a seamless transition between different communication cells, the scheduling concepts for resource allocation for the different users, and the ability to work also indoors are also very promising issues for robotics. In particular, the application area of service robotics can largely benefit from these characteristics. On the other hand, the mobile phone networks like UMTS are designed for different purposes and under different constraints. Therefore, it is important to investigate the critical parameters of a communication technology like UMTS in order to adjust the available communication parameters on the application layer and to realize the optimum usage of this technology for the application.
The use of this technology in the context of connecting hardware was also investigated in the area of car-2-car communication [128] and vehicular ad-hoc networks (VANETs) [129]. Nevertheless, these investigations show, that the area of networked robotics might also be a suitable application area for the UMTS

technology, but further investigations of the communication link are necessary as the requirements of networked robots differ from the requirements of the already investigated areas.

The European mobile phone system of the second generation (2G) "Global System for Mobile Communications" (GSM), which was developed in the nineties, was mainly designed for voice communication. For these 2G systems, extensions like High Speed Circuit Switched Data (HSCSD), General Packet Radio Service (GPRS), or Enhanced Data Rates for GSM Evolution (EDGE) were introduced in order to enabled data communication with a bandwidth of 115.2 kbit/s (HSCSD), 171.2 kbit/s (GPRS), and 384 kbit/s (EDGE) (cf. [28]). Currently, the European mobile telephone systems are upgraded to the third generation system *Universal Mobile Telecommunications System*. These systems allow for mobile high bandwidth data and voice communication. Telecommunication providers now expand the coverage of their broadband services from urban areas towards highways and rural areas. The large-area availability of this wireless data service has a high potential for newly developed services, and it is also a promising technology to be used for mobile robot teleoperation.

Nevertheless, before UMTS can be seamlessly integrated in existing frameworks for networked robots, several analyses with respect to the data transmission have to be done. Besides the available bandwidth, also characteristics like the one-way delay, the round-trip times, the packet inter-arrival times, and the packet loss are important and have a big influence on the communication among networked mobile robots or between mobile robots and the user.

This chapter starts with a brief introduction of the enabling technology UMTS and explains methods for measurements and analysis of the data traffic which are relevant in the following scenarios. Further, the description of an evaluation scenario and the corresponding test setup is given which is followed by the evaluation of the measurements and test results. The chapter concludes with a summary and a discussion of the results.

## 4.1 Enabling Technologies

### 4.1.1 Universal Mobile Telecommunications System (UMTS)

The Universal Mobile Telecommunication System (UMTS) standard is a new mobile communication standard of the third generation (3G) of mobile communication systems. The UMTS standard complies to the international IMT-2000 specification, but nevertheless it is not a worldwide unified system. Currently, European countries are working on establishing UMTS in metropolitan areas. Pivotal for the development of a new communication standard was the objective of a more efficient utilization of the limited number of available frequencies and the need of higher data rates for end users. This need for a higher bandwidth in mobile environments came up as the Internet became more and more popular, and as also mobile Internet access was promoted. As traditional GSM based cellular phone systems were primarily designed for voice communication, they were not able to provide appropriate data rates for the emerging mobile data services. Thus, UMTS was developed to fulfill requirements like efficient utilization of the resources (the number of available frequencies is limited and has frequency allocation has to be done carefully in order to use these limited resources efficiently), support of packet switched and circuit switched communication, support of different data rates, support of variable bit rate traffic and adaptive Quality of Service (QoS) according to the current link status, Variable cell radius, flexible management of radio resources, and the support of QoS for different services. In general, the requirements for 3G mobile communication systems are the support of bit rates up to 2 Mbps, the support of variable bit rates to offer bandwidth on demand, multiplexing services with different quality requirements like voice, video, and data on one single connection, support of variable delay requirements, quality requirements from 10% frame error rate to $10^{-6}$ bit error rate, support of coexistence of second- and third-generation systems, compatibility and inter-system handovers between second- and third-generation systems, support of asymmetric uplink and downlink data transmission, and a high spectrum efficiency.

To fulfill these numerous requirements, UMTS is developed in several consecutive phases and each system release realizes more of the requirements. The final stage of UMTS fulfills the above mentioned requirements and has capabilities like the support of high bit rates (theoretically up to 2 Mbps in 3rd Generation Partnership Project (3GPP) Release 99, up to 14.4 Mbps in 3GPP Release 5, and up to 28.8 Mbps in 3GPP Release 7), low delays and packet round trip times below 100 ms (Release 5) or below 50 ms (Release 6), mobility for packet data, support of QoS, simultaneous voice and data capability, and compatibility to GSM and GPRS systems. Thus, a wide set of services is supported.

## Services

In general, according to [130], the services supported by UMST can be divided into circuit switched services, packet switched services, content to person services, business connectivity, and location services. The term circuit switched services embraces the classical voice services, Adaptive Multi-Rate (AMR) voice, wideband AMR voice and video. Also a multimedia architecture for circuit switched connections was recommended by the ITU-T with the H.324M system which should support H.263, as well as MPEG-4 video codecs. Packet switched services include messaging using short messaging service (SMS), multimedia messaging service (MMS), audio messaging, instant messaging, video sharing and mobile email. Also a push-to-talk over cellular (PoC) service is envisioned which should allow unidirectional speech transmission to a defined group of receivers. In addition, also voice over IP (VoIP) services and the support of multiplayer games are packet switched services. Traditional browsing, audio and video streaming, and content download can be summarized by the term content to person services. Here, typically the applications transfer data very asymmetric and for user convenience the delays and download times should be low. Services in the area of business connectivity allow access to company intranet or the synchronization of business data from mobile offices, notebooks, PDAs, or smart phones. Here, also end-to-end security mechanisms like Virtual Private Networks (VPN) are supported, and as usually a lot of data is transmitted, the download and upload performance with respect to the round trip time is important. Location services is a relatively new developing area of services. In the context of mobile communication, the user should be supplied with information relevant for his current location. Smart phones will be equipped with GPS sensors and thus, information like maps, city guides, traffic information, or a weather forecast can be obtained with respect to the user's position. Also, the combination of location information combined with communication capabilities can be use the other way around. It can be also possible to give the user a better information about his position via the communication channel in case the GPS signal is not good enough to get a position fix.

### System Architecture

To realize the objectives of UMTS, the system is based on a new design of the network infrastructure which can be grouped into the UMTS Terrestrial Radio Access Network (UTRAN), the Core Network (CN), and the User Equipment (UE) (cf. Figure 4.1). All radio related issues are handled by the UTRAN. The connection to external networks, all switching and routing is done by the CN. These three functional elements consist of several components which are shown in Figure 4.1.

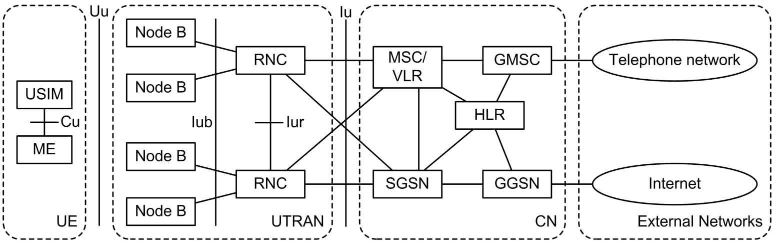


Figure 4.1: UMTS architecture (according to [130]).

In a first implementation step, updates of existing GSM network infrastructure allow an inexpensive introduction of this new system as some existing GSM network components can still be used for UMTS. Within the Core Network, circuit switched data (telephone calls) coming from the UTRAN is forwarded to a Mobile Switching Center (MSC), and via the Gateway Mobile Switching Center (GMSC) it reaches the telephone network, and thus, its destination. In case of packet switched data from the UTRAN, Serving GPRS Support Nodes (SGSN) forward the data via the Gateway GPRS Support Node (GGSN) to the Internet. The Home Location Register (HLR) is a database containing the users' service profiles like allowed services or allowed roaming areas. The MSC, SGSN, and GGSN components already exist for the GSM networks and are used for UMTS after a small upgrade. The Radio Network Controller (RNC) is responsible for a set of Node B modules which represent the radio resources. In general, the RNC is the service access point for all supported services in its cells. RNC and Node

B are components which are specially developed for UMTS networks. The real innovation of UMTS in contrast to GSM is the use of the Wideband Code Division Multiple Access (W-CDMA) technology which has currently emerged as the most widely used air interface for third generation wireless communication systems. Of course, CDMA was already used for the cellular phone system in the USA since the beginning of the 1990ies while in Europe, the GSM technology is based on Time Division Multiple Access (TDMA). W-CDMA allows for a simultaneous communication of several users using the same frequency, each without interfering the communication of someone else. Therefore, each data stream is spreaded using a unique defined spreading code before it is sent. The receiving node knows the spreading code and can despread the received data stream in order to extract the original data. As each of the used spreading codes are orthogonal, the spreaded signal cannot interfere the other spreaded signals and also cannot be interfered by other spreaded signals. The supported data rate depends on the length of the spreading code and can also be adapted during a transmission. This technology also requires a power control of the sending units in order to handle the near-far problem. Otherwise, it would be possible for a node to predominate the signal of another node which might cause a failure of the CDMA mechanism. For more details on these mechanisms please refer to [130]. Currently, basic UTMS networks support data rates of 384 kbit/s for the downlink and 64kbit/s for the uplink. Since 2006 the High Speed Downlink Packet Access (HSDPA) and High Speed Uplink Packet Access (HSUPA) extension is implemented which supports now data rates of 7.2 Mbit/s for the downlink and 3.6 Mbit/s for the uplink. Of course, several aspects have an influence on the data rates which are realized with available hardware. Important for the maximum supported data rate are e.g the distance between the device and the base station or the number of users which are currently active in a cell. Another aspect is the HSDPA-Category of the user equipment. Usually, currently up-to-date equipment which can be purchased has HSDPA-Category 6 which corresponds to maximum data rate of 3.6 Mbit/s, or HSDPA-Category 8 which corresponds to maximum data rate of 7.2 Mbit/s. The main objective of the second HSDPA extension is the reduction of the round trip times. Here, the management of HSDPA connections is shifted from the RNC to the Node B which allows for round trip times between 50 ms and 100 ms. Also, the possible packet inter-departure time is reduced by HSDPA from 10 ms (UMTS) to 2 ms. This section about the UMTS technology is only a short summary of the most important mechanisms which are relevant for the following sections on mobile robot teleoperation via UMTS. For further reading on details of UMTS and WCDMA please refer to [130].

### 4.1.2 Measurement of One-Way Delays

The measurement and verification of end-to-end or one-way data traffic characteristics is not only a current research topic in the area of networked robotics. Since the nineties, where the popularity of the Internet dramatically increased, measurement methodologies for traffic characteristics were implemented and verified. In [131], special *DAG monitoring cards*[1] with synchronized clocks are used and the one-way delays are measured for data streams which are generated according to common traffic models. [132] uses special hardware with atomic clocks synchronized via GPS. According to the authors, this setup achieves an accuracy of 10 $\mu s$. [133] measures the single-hop packet delay for routers of the Sprint[2] IP backbone network. Specially equipped routers allow for measuring the processing time of the packets at the router itself and provide an accuracy of about $6-10$ $\mu s$. Finally, detailed analyses of the queuing delay distributions are given. In [134] a special toolset for measuring one-way delays, the *CM Toolset*, is presented. Here, a computational correction is used which gives, according to the authors, an accuracy of about $1-10$ $ms$. In [135], Paxson investigated end-to-end Internet packet dynamics and describes the used methodologies in [136]. Even today this topic is still under research as shown in [137].
With respect to the evaluation scenario described in this chapter, where networked mobile robots communicate via UMTS, the measurement of one-way delays is also relevant. In situations, where important high priority messages are exchanged between the mobile nodes, or if the data feedback from a mobile robot has to keep defined constraints. This is often the case while a mobile robot is teleoperated, as the one-way delay characteristics must be known in order to seamlessly integrate the communication channel into the underlying robot control framework. One big challenge of one-way delay measurements is the correct interpretation of the different timestamps of the measured data packets, which is directly influenced of the different clocks at each involved measurement node. As the expected delays are in the area of 50 milliseconds or more, the aimed

[1] Endace Measurement Systems, http://www.endace.com (09.02.2010)
[2] https://www.sprint.net (09.02.2010)

accuracy of the measurement should be at least in the magnitude of one millisecond in order to allow for a meaningful interpretation of the data with respect to the application described in this work. The requested measurement accuracy has high demands on the synchronization, resolution, the accuracy, and the skew of the clocks.
For the explanation of the terms clock synchronization, clock resolution, clock accuracy, and clock skew the nomenclature of [136] and [138] is used.
**Offset:** The offset $O$ is defined as time difference between the time of the clock $t_c$ and a reference time $t_r$: $O = t_c - t_r$. Often, a "true" time according to national standards is used as reference time $t_r$. In Germany, the "true" reference time is provided by the "Physikalisch Technische Bundesamt"[3].
**Clock synchronization:** A clock is synchronized to a given time reference, if the time offset between clock and reference time is below a given threshold $d$, $|O| \leq d$. Usually, this threshold is defined by the application. A common method for clock synchronization of different PCs is the Network Time Protocol (NTP) [138] which exchanges packets with a time source to estimate the current time. Thereby, the accuracy of NTP strongly depends on the used operating system and the quality of the used network connection. With respect to the current problem, the roles of the network as infrastructure for time synchronization and as object to be analyzed turns out to be a drawback, as effects inside the network which are object of the current analysis potentially effect also the quality of the clock synchronization. Another problem of synchronizing different clocks of different PCs over a longer period of time is, that PC clocks usually have a clock drift and a clock skew which results in an offset between the formerly synchronized clocks.
**Clock resolution:** The clock resolution is the smallest time interval by which the clock is updated. Nevertheless, a small resolution does not guarantee a high accuracy of the clock.
**Clock accuracy:** The accuracy of the clock at a given time $t$ is given by the current offset $O(t)$ and the clock is accurate if $O(t)$ is zero.
**Clock skew and drift:** According to [138], the skew of a clock at a particular moment is given by the frequency difference between the clock itself and national standards which is equal to the first derivative of its offset with respect to the "true" time. It is also said that in reality, clocks usually have some variation in its skew given by the second derivative of the clock offset with respect to the

[3]http://www.ptb.de (09.02.2010)

"true" time, which is called drift [138].

### Measurement Methods

The objective of the one-way delay measurements of this chapter is a characterization of the communication link with respect to the requirements of mobile robot teleoperation. When a mobile robot is teleoperated, several data streams of different characteristics and requirements are sent via the same communication link. For some data packets, a minimum of delay is desirable, or a maximum delay for reliable transmission is required (e.g. important commands with respect to security issues). Also for the transmission of sensor data or video, a known link characteristic allows for efficient traffic shaping which will increase the link quality. To get meaningful results for one-way delay measurements, the clock synchronization of the involved measurement nodes is required. As already mentioned above, all these clocks have a drift which is not predictable. Thus, the synchronization of the clocks has to be repeated from time to time. The regular synchronization of the clocks has on one hand the advantage, that the offset between both clocks is set to zero at particular time $t_{syn}(i)$. But on the other hand, it might also cause some problems considering the measurement results, as it is shown in Figure 4.2. To give a short explanation of the meaning of Figure 4.2 let us assume a measurement setup of two PCs which are sending data packets from one PC to the other one and the one-way delay of these packets should be measured. The real propagation delay of the data packets is exactly 50 ms – of course, in reality this will not behave exactly like this, but for understanding the problem these assumptions can be made. Figure 4.2 is basically divided into three parts. The topmost graph shows the offsets of two clocks $c1$ and $c2$ of different PCs with respect to a "true" time in milliseconds. In the example of Figure 4.2, both clock offsets $O_{c1}$ and $O_{c2}$ are positive which means that both clocks are progressing faster than the "true" time. It is also observed, that the offset changes over the time (cf. the gradient of the green and the red graph in Figure 4.2 change over the time). This variable offset shows drift of the time of each clock. Now, both clocks are synchronized with the "true" time at $t_{sync}(i)$, which will set the offset to zero.

The two graphs in the center of the figure describe the situation of the packet transmission. The upper x-axis shows the "true" time-axis of PC 1 and the lower x-axis shows the "true" time-axis of PC 2. At 50 ms of "true" time at PC 1, a

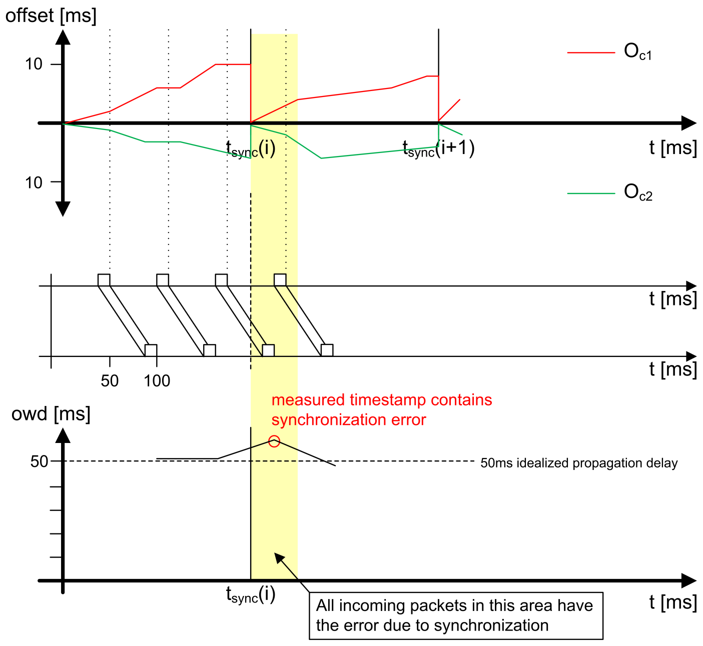

offset [ms]
10
10
$t_{sync}(i)$
$t_{sync}(i+1)$
t [ms]
$O_{c1}$
$O_{c2}$
t [ms]
t [ms]
50
100
owd [ms]
50
measured timestamp contains synchronization error
50ms idealized propagation delay
$t_{sync}(i)$
All incoming packets in this area have the error due to synchronization
t [ms]

Figure 4.2: Measurement error due to synchronization.

packet is transmitted to the second PC and the real one-way delay of this packet in this example is assumed to be exactly 50 ms. Subsequently, additional packets are sent from PC 1 to PC 2 with a given packet inter-departure time of 50 ms. The third and bottom graph, shows the measured one-way delay (OWD) of the packets at PC 2 in milliseconds. Note that the OWD can only be measured after the packet is completely received at PC 2. The x-axis of all the different graphs of Figure 4.2 are aligned and have the same scale. The measured OWD of packet *n* now can be described by the following equation:

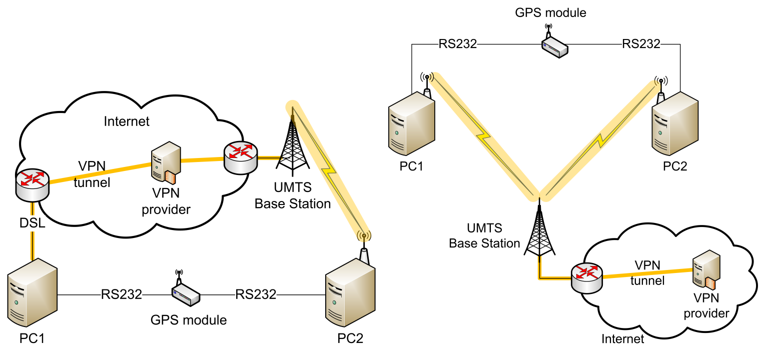
$$OWD(n) = O_{c1}(t_{tx}(n)) - O_{c2}(t_{rx}(n)) + \Delta t_p \tag{4.1}$$

OWD(n) is the measured one-way delay of packet n, $O_{c1}(t_{tx}(n))$ is the offset of clock 1 at the particular time of transmission $t_{tx}$ of packet $n$ with respect to "true" time, $O_{c2}(t_{rx}(n))$ is the offset of clock 2 at the particular time of reception $t_{rx}(n)$ of packet $n$, and $\Delta t_p$ is the propagation delay of packet $n$. In the example of Figure 4.2 the measured OWD values for the packets 1, 2 and 4 are between 51 ms and 48 ms, and for the third packet OWD(3) is at 60 ms. Here, the measurement error due to synchronization is induced. The sender and the receiver clock is synchronized with the "true" time while packet 3 is traveling from PC 1 to PC 2. Thus, the time stamp for sending $t_{tx}(3)$ is taken before the synchronization time $t_{sync}(i)$, and the time stamp at the reception of the packet at PC 2 $t_{rx}(3)$ is taken after the synchronization when the offset at PC2 is very small due to the synchronization. Of course, this error can be much higher in real world setups as it depends strongly on the characteristics of the clock drifts of each clock which is involved in the measurement. For the application to the real world measurement scenario, a packet whose sending time stamp $t_{tx}(n)$ is set before a particular synchronization time $t_{sync}(i)$, and whose receiving time stamp $t_{rx}(n)$ is set after $t_{sync}(i)$, cannot be used for data evaluation, as afterwards the correction of the induced measurement error due to synchronization cannot be corrected anymore. For the measurements presented in this chapter, the clocks are synchronized each second, and for measuring only the relatively small intervals of less than 300 milliseconds following after each particular synchronization time are used. This minimizes the errors caused by unpredictable and variable clock drifts, as the offsets increase less if the elapsed time interval is short. Also the problem of the induced error due to clock synchronization is solved, as no relevant packets are en-route during clock synchronization.

### Measurement Setup

Basically, two different measurement scenarios are used for the presented analysis. In the first scenario, data traffic between a PC and a mobile node is evaluated (cf. Figure 4.3a). Here, the PC is connected to the Internet via a DSL dial-up connection, and the mobile node is connected to the Internet via UMTS. In the second scenario (cf. Figure 4.3b), two mobile nodes are connected via UMTS. As they are located close to each other, they are even in the same UMTS cell. As currently UMTS providers allow access to own services only via dedicated vpn gateways, the communication in both scenarios uses a vpn tunnel. Of course a vpn tunnel increases the data traffic as additional information is added each payload data packet. Thus, also delays are induced – on the one hand due to the increased data volume, on the other hand also due to processing times inside the vpn system itself. Nevertheless, as the use of a vpn is necessary in case dedicated services should be used, it is thoroughly necessary to perform measurements in this real world setup. In the following sections the term one-way delay always includes all delays which are created by all the different used technologies and the measurements really reflect the situation of an existing system which is purchasable on the market.

For both scenarios of Figure 4.3, the measurement program itself is the same and



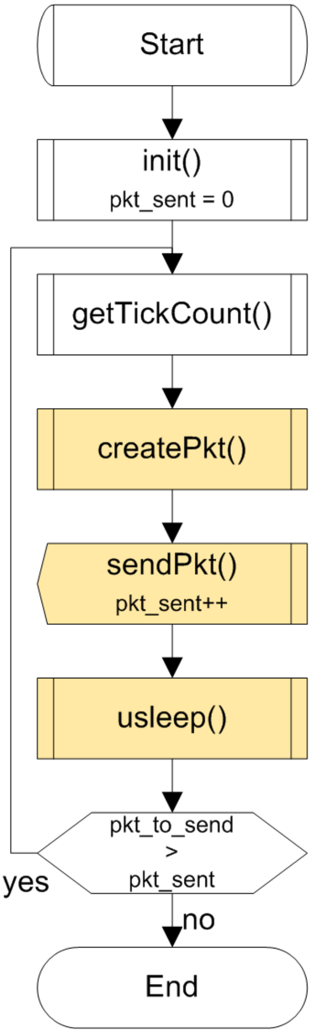
(a) Measurements between PC and UMTS node via DSL.

(b) Measurements between two UMTS nodes.

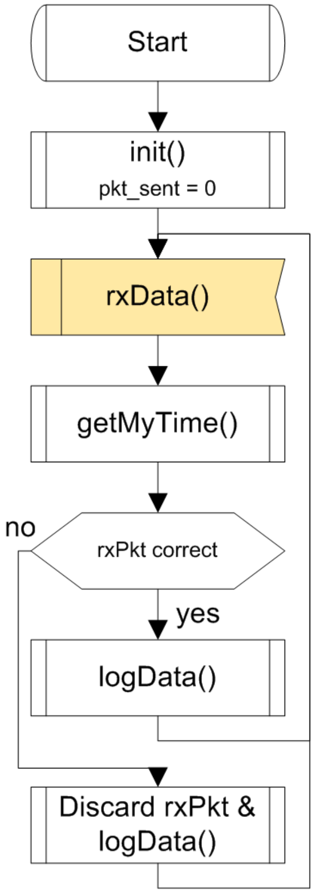
Figure 4.3: UMTS measurement scenarios.

the GPS time is used as reference time source. Both measurement PCs are connected to the same GPS module using a RS232 interface and both PCs receive the current GPS time simultaneously each $\Delta t_{sync}$ milliseconds (default value is 1000 ms). The measurement program consists of two parts: a client program which actively sends data packets, and a server application receiving data packets from the client. As the client carries out active measurements, the generated traffic stream corresponds to the data stream of the application scenario to be analyzed: the remote control of a mobile robot. The server application also evaluates the timestamps and stores the experiment data. The functioning of both programs is clearly presented in Figure 4.4. The client synchronizes with the GPS time source each second and sends data packets with a given *packet size* and a given *packet inter-departure time* to the client. In order to avoid the above described error due to time synchronization, packets are marked for evaluation only during the time interval $\Delta t_m = t_{sync} + 150\ ms$, with $t_{sync}$ being the particular time of clock synchronization. All other packets which are sent are just used to keep the communication link in the desired load condition. To get an accurate time value also in between the particular clock synchronization times, the *system tick counter* is used. The resolution of the system tick counter depends on the underlying operating system. For the following measurements, the operating system provides a system tick counter resolution of 1 *ns*. Detailed functions of the client application are shown in Figure 4.4a. After the initialization process function is finished `getTickCount()` is used to get the current *system tick count* before starting to create and send the packet. In the next step the packet is created with the given size containing the current timestamp. In case it is sent within interval $\Delta t_m$, it is also marked for evaluation. Function `sendPacket()` increments the `pkt_sent` counter by one and sends the packet. Finally, the client program waits for the remaining packet inter-departure time which can be calculated by the system tick count acquired before the packet was created and the packet inter-departure time given at program start. As long as `pkt_sent` is smaller than the given number of packets to be sent, the program continues sending data as described above.

The functioning of the server application is shown in Figure 4.4b. The time is synchronized also every second and it uses the same mechanism like the client application of using the *system tick counter* for calculating the accurate time between particular clock synchronization times. After initialization, the program waits for measurement packets from the client. As soon as a packet is received,



(a) Client application.



(b) Server application.

Figure 4.4: Simplified SDL diagrams of the measurement programs.

the current time is calculated. In case the packet is received correct, the departure timestamp of the packet is stored together with the time of the reception and afterwards, the program returns to waiting for a new packet to arrive. In case the packet transmission is incorrect, it is considered as packet loss.
In Figure 4.4 all functions which potentially influence the measured time values are colored yellow. With respect to the client application, the loop for sending packets should be repeated with a constant rate given at program start. As this loop contains the functions `createPkt()`, `sendPkt()`, and `usleep()`, the complete processing time of these three functions should be equal to the packet inter-departure time $t_{pidt}$ given at program start. Therefore, the system tick counter is used to estimate the time elapsed while the packet is created and sent and `usleep()` will wait for the period $t_{usleep()} = t_{pidt} - t_{createPkt()} - t_{sendPkt()}$. The delay induced by processing buffers of protocol stacks while sending or receiving data occurs in the functions `sendPkt()` and `rxData()`. This delay must not be excluded from the measurement as it also occurs in the real communication process between two nodes and thus, it belongs to the one-way delay which is objective of the measurements. A detailed analysis of the above presented setup shows, that this method suits very well for measuring the one-way delays in the following scenarios. The one-way delays are expected to be in the magnitude of several 10 milliseconds.

## 4.2 Link Analysis of UMTS for Mobile Robot Teleoperation

This section gives now a detailed analysis of the communication channel characteristics for a mobile robot being teleoperated via UMTS It starts with a description of the experiment setup. Then, packet inter-arrival times, round trip times, one-way delays, and packet loss are characterized in different setups which are typical for telerobotics. Finally, a discussion of the results is given.

### 4.2.1 Scenario Setup

The focus of this section is set on teleoperation of a mobile robot via UMTS communication link and the characterization of the communication link in order to allow for a seamless teleoperation of the robot. Therefore, three different test setups are analyzed:

- The connection between one robot connected to UMTS and a PC connected to the Internet via DSL (16 MBit/s) (Mode 1).
- The connection of two mobile robots via UMTS (Mode 2). In this case, only the two robots generate traffic.
- The connection of two mobile robots via UMTS while a third UMTS node is transmitting a large amount of data to the Internet (Mode 3).

In a first step, the data for teleoperation is generated without a connection to real robot hardware. The mobile clients are represented by mini PCs which are connected to the Internet via a USB UMTS device (Huawei 3G modem). This device supports HSDPA and HSUPA broadband data transmission besides the normal UMTS mode and it is used for all tests presented in this work. Currently, all UMTS access providers do not provide public IP addresses, and usually proprietary services are blocked. Therefore, each UMTS node joins a virtual private network using openvpn[4] (cf. Figure 4.5). Thus, a physical, and a hardware component is present (cf. left part of Figure 4.5), and a logical overlay is set on top of this (cf. right side of Figure 4.5) which enables easy addressing of each mobile node. Usually, the UMTS specification promises a reliable packet transmission every 20 milliseconds. To get an idea of the link behavior, for each of the three above mentioned modes, data streams of different sizes are generated. As measurement categories, the packet inter-arrival time is analyzed as this is an important criterion for video and sensor data transmission. The round trip time (rtt) is investigated as this two-way delay is also very important in case the mobile robot is teleoperated directly by a human operator. For the packet inter-arrival time analysis, data is transmitted only one-way and the packet inter-arrival time is plotted at the destination node. Therefore, data is generated with a packet size of 2048 bytes and a packet inter-departure time of 10, 20, 50, and 100 milliseconds are generated. This results in packet streams of 20, 40, 100, and 200 kbytes per second. The round trip time measurements use ICMP ping packets with the size set to 2040 bytes of payload and 8 bytes ICMP header. Here, the send intervals for ping packets are also set to 10, 20, 50, and 100 ms which generates data streams comparable to the above described tests for the packet inter-arrival time.

[4]http://openvpn.net (09.02.2010)

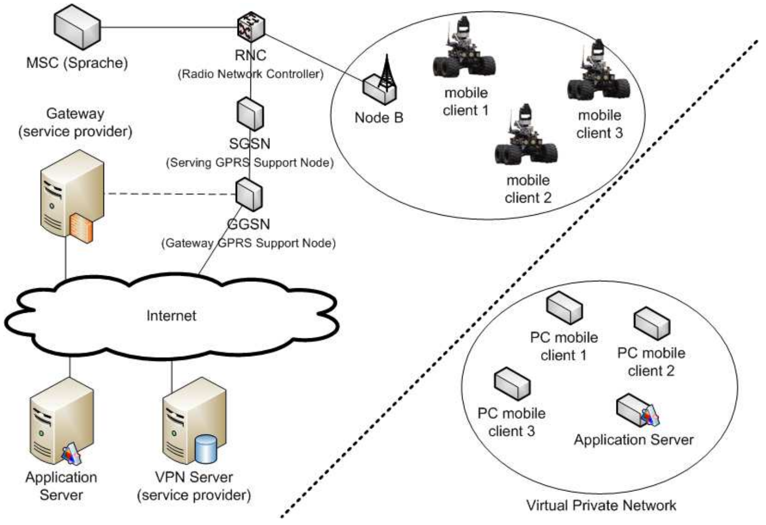


Figure 4.5: Connecting mobile robots via UMTS.

Of course, the data stream for the round trip time tests are transmitted in both directions – from the source to the destination robot and back.

### 4.2.2 Packet Inter-Arrival Time

For this analysis, the packet inter-arrival time at the destination node is measured. Also the number of packets is counted. The results are displayed in histograms with a bin size of 2 milliseconds. The smallest bin holds all values between 0 and its own value, and the bin with the highest value holds also all larger values up to infinity. The x-axis shows the packet inter-arrival time in seconds and the y-axis shows the relative frequency of occurrence. As the data set of the measurements contains enough data to prove that the resulting distribution is stable, the relative frequency of occurrence can be considered as equal to the probability of occurrence of the corresponding packet inter-arrival time. Results of the following sections were partially published in [7].

### Connection between Internet PC and UMTS node

For this setup, the packet inter-arrival time is measured in both directions for packet streams of 20, 100, and 200 kbytes/sec. Figure 4.6 shows the results for the 20 kbytes/sec stream. The packet inter-departure time for this stream is 100 milliseconds. The upper subplot of Figure 4.6 shows, that more than 50% of the packets arrive with an inter-arrival time of 100 ms. Approximately, another 10% arrive with an inter-arrival time of 90 ms and 110 ms respectively. For the opposite direction – from UMTS node to Internet PC (cf. lower subplot of Figure 4.6)– the result is completely different. Here, the packet inter-arrival time is closely distributed around the expected packet value of 100 milliseconds.

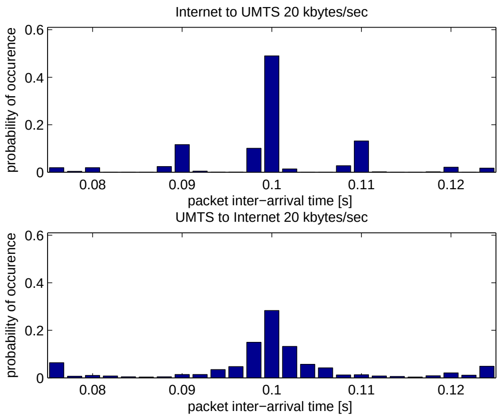


Figure 4.6: Packet inter-arrival time between Internet PC and UMTS node and a data bandwidth of 20 kbytes/sec.

For the stream with a bit rate of 100 kbytes/sec, the results between the both investigated transmission directions vary significantly (see Figure 4.7). At the UMTS node, more than 50% of the transmitted packets arrive with an inter-arrival-time of 20 ms and two other peaks with 10% each are visible at about 10 ms and 30 ms. For the packet inter-arrival time for the transmission direction from UMTS node to Internet PC, only about 13% of the packets have an inter-arrival time of 20 ms. Here, also small peaks at 10ms and 30 ms ($>$ 6% each), as well as a high amount of packets with 5 ms or less ($>$ 13%) and with 40 ms or more ($>$ 7%) are present.

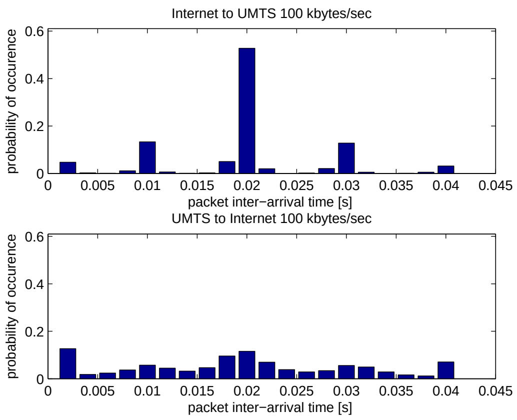


Figure 4.7: Packet inter-arrival time between Internet PC and UMTS node and a data bandwidth of 100 kbytes/sec.

Also for the 200 kbytes/sec stream, the results for both directions are quite different (cf. Figure 4.8). For the transmission from Internet PC to UMTS node more than 40% of the packets have an inter-arrival time of 10 ms which corresponds to the present packet inter-departure time. More than 20% of the packets have an inter-arrival time of 4 ms and less. For the other transmission direction – from UMTS to internet PC – only about 20% of the packets arrive with the expected inter-arrival time of 10 ms. A second peak of about 18% is present for an inter-arrival time of 4 ms and less. Most of the other packets are distributed to inter-arrival times between 4 ms and 36 ms.

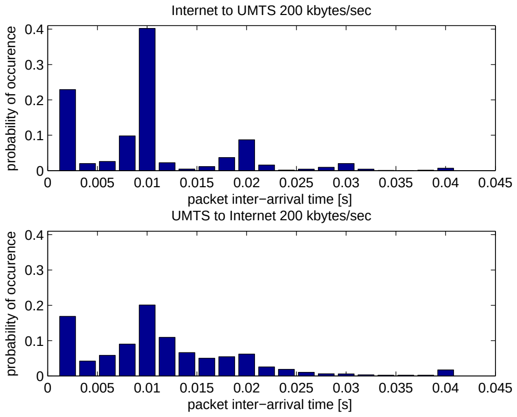


Figure 4.8: Packet inter-arrival time between Internet PC and UMTS node and a data bandwidth of 200 kbytes/sec.

### UMTS to UMTS connection

This section shows the results of transmissions between two UMTS nodes. Each upper subplot of Figures 4.9, 4.10, and 4.11 show the packet inter-arrival times when traffic is transmitted only between the two involved nodes. The lower subplots of these Figures show the inter-arrival time of the packets while a third UMTS node transmits a large file to an Internet PC. Thus, this data stream must share the UMTS cell capacity with the measured data stream.

Figure 4.9 shows the results for the 20 kbytes/sec stream. Here, both setups show similar results. The majority of the packets has an inter-arrival time of 100 ms (33% without additional traffic and 28% with additional traffic) and almost all other inter-arrival times are distributed in 10 ms intervals at 80 ms, 90 ms, 110 ms, and 120 ms. For this stream, the additionally generated traffic reduces the amount of the packets with an inter-arrival time of 100 ms for approximately 5%.

In Figure 4.10 the results are displayed for the 100 kbytes/sec stream with a packet inter-departure time of 20 ms. In both situations, almost 20% of the packets have an inter-arrival time of 20 ms. In case of no additionally generated traffic, more than 55% of the packets arrive with an inter-arrival time of less than

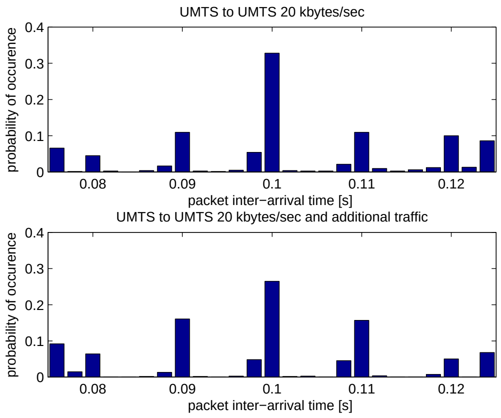


Figure 4.9: Packet inter-arrival time between UMTS nodes and a data bandwidth of 20 kbytes/sec.

10 ms. In case additional traffic is generated, only about 42% of the packets have an inter-arrival time of 10 ms or less. The remaining packet inter-arrival times are distributed at peaks around 30 ms and 40 ms.

The results of the 200 kbytes/sec stream displayed in Figure 4.11 are similar to the above described observations for the 100 kbytes/sec stream. The additionally generated traffic reduces the amount of packets at the expected inter-arrival time of 10 ms for about 5%, and the remaining packets are again located at the bins with 10 ms inter-bin distance and inter-arrival times of less than 4 ms, 20 ms, 30 ms, and 40 ms.

Finally, the effective receiving bit rates of the payload data is shown in Tables 4.1 and 4.2. Table 4.1 shows the results of the test runs between UMTS node and Internet PC and Table 4.2 shows the results of the test between two UMTS nodes without additionally generated traffic (Mode 2) and between two UMTS nodes with additionally generated traffic (Mode 3). These two tables give an idea of the present packet loss during the test runs. Surprisingly good results are achieved for all data streams during the transmission from Internet PC to UMTS node. In the opposite direction, an increased packet loss is observed for the 200 kbytes/sec stream. For Mode 2 and Mode 3, where two UMTS nodes communicated with

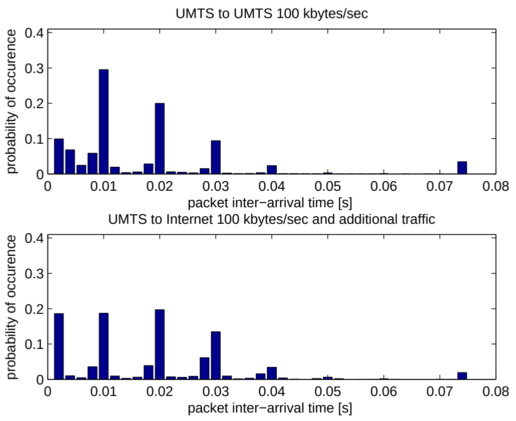


Figure 4.10: Packet inter-arrival time between UMTS nodes and a data bandwidth of 100 kbytes/sec.

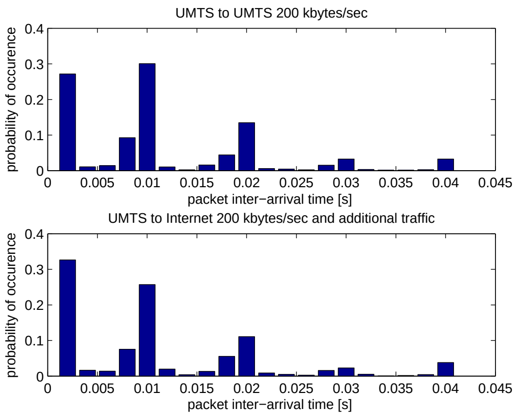


Figure 4.11: Packet inter-arrival time between UMTS nodes and a data bandwidth of 200 kbytes/sec.

each other, the packet loss was significantly higher.

| sending data rate | receiving at UMTS node | receiving at Internet PC |
|---|---|---|
| 20 kbytes/s | 19.20 kbytes/s | 19.58 kbytes/s |
| 100 kbytes/s | 98.09 kbytes/s | 92.62 kbytes/s |
| 200 kbytes/s | 194.33 kbytes/s | 162.82 kbytes/s |

Table 4.1: Resulting effective payload bit rates at the receiving node.

| sending data rate | receiving Mode 2 | receiving Mode 3 |
|---|---|---|
| 20 kbytes/s | 16.19 kbytes/s | 19.31 kbytes/s |
| 100 kbytes/s | 94.03 kbytes/s | 92.53 kbytes/s |
| 200 kbytes/s | 141.80 kbytes/s | 125.61 kbytes/s |

Table 4.2: Resulting effective payload bit rates at the receiving node.

### 4.2.3 Round Trip Times

Figure 4.12 shows the measured round trip times for the generated packet streams. Mode 1 corresponds to the Internet-UMTS node scenario, Mode 2 is the transmission between two UMTS nodes without additionally generated traffic and for Mode 3, the data exchange of two UMTS nodes is measured while a third node transmits additional data to an Internet PC. These box-plots show the median of the measured values (horizontal line) and the box shows 50% of the values bounded by the lower 25% quartile and the upper 75% quartile. Furthermore, lines indicate the most extreme values within 1.5 times the interquartile range from the ends of the box and extreme values lying outside this borders are marked with crosses. For the three streams with 20 kbyte/s, 50% of the measured rtt values are distributed very close to the corresponding median. A similar observation can be made for the 40 kbyte/s stream in Mode 1 and Mode 3, and for the 100 kbyte/s stream in the Mode 1 scenario. As expected, the largest variations appear for the high bandwidth streams with 200 kbyte/s and the 100 kbyte stream in Mode 3 with the additionally generated traffic. In general, often 50% of the measured rtt values of each test run are located quite close to the corresponding median. Only few extreme values are measured below the lower 25% quartile border and sometimes, very high rtt values are measured above the upper border of the 75% quartile (e.g. for the 100 kbyte/s stream in Mode 3). As the confidence intervals with respect to the median values of these measurements do not overlap at all, the distributions of these measured data rows can be considered as significantly different. Thus, the later used application to mobile robotics must consider these high round trip times which occur sometimes and must be able to handle this large variability.

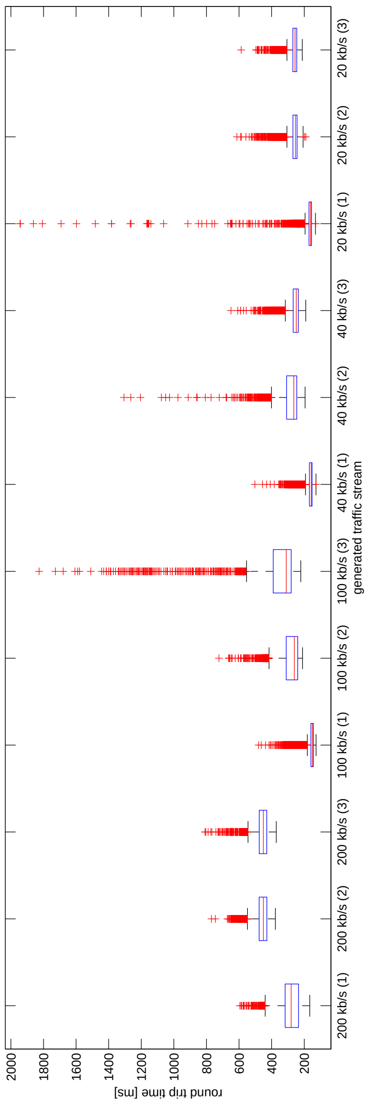


Figure 4.12: Round trip times in milliseconds (Mode (1): Internet-UMTS; Mode (2): UMTS-UMTS; Mode (3): UMTS-UMTS and additional traffic).

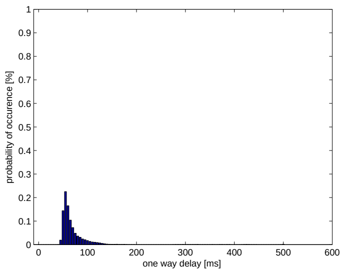
### 4.2.4 One-Way Delay for Data Streams

The one-way delay is measured using the method explained in Section 4.1.2. Therefore, six traffic streams which are typical for robotics applications (cf. Table 4.3) are generated with a different packet inter-departure time (pidt) for each of these streams. Each packet has a payload of 100 bytes. Additionally, headers of IP, UDP, and the VPN increase the complete packet size to 161 bytes. As already

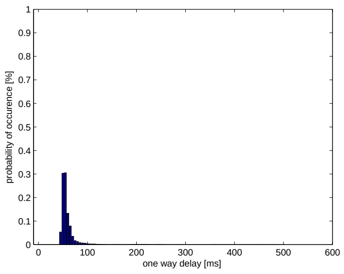
| stream ID | pidt [ms] | bandwidth | application |
|---|---|---|---|
| A | 2 | 80.5 $\frac{kbyte}{sec}$ | real time sensor data 500 Hz |
| B | 5 | 32.2 $\frac{kbyte}{sec}$ | real time sensor data 200 Hz |
| C | 10 | 16.1 $\frac{kbyte}{sec}$ | real time sensor data 100 Hz |
| D | 20 | 9.05 $\frac{kbyte}{sec}$ | real time sensor data 50 Hz |
| E | 50 | 3.22 $\frac{kbyte}{sec}$ | manual robot teleoperation |
| F | 100 | 1.61 $\frac{kbyte}{sec}$ | manual robot teleoperation |

Table 4.3: Generated traffic streams for the measurements.

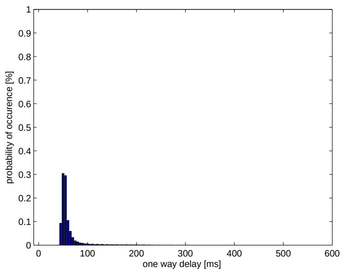
mentioned in Section 4.1.2, the measurement method does not use the complete interval between the particular clock synchronization times for the measurement in order to avoid errors due to packets being en-route during clock synchronization. The following setup measures only packets which are sent within the first 150 ms after a particular clock synchronization time. The other packets which are not taken into account for measurement, as they are sent more than 150 ms after clock synchronization, are only used to keep the communication link in the desired load status. For all measurements 1000 packets are marked for evaluation, and thus, for each test run 66667 packets $((1000\ ms/150\ ms) \cdot 10000)$ are measured. The OWD is evaluated in three scenarios: both communication directions between mobile node and Internet PC (cf. Figure 4.3a), and the communication between two mobile nodes (cf. Figure 4.3b). The results for the OWD measurements for the direction from *Internet PC to mobile node* are shown in Figure 4.13. The x-axis specifies the measured OWD in milliseconds, and the y-axis gives the relative frequency, and thus, the probability of occurrence of the corresponding OWD value. To get a clear diagram of the distribution of the measured one-way-delays, each histogram plot in Figure 4.13 uses bin sizes of 5 ms for the OWD values. At first view, apart from stream F, the resulting OWD measurements for



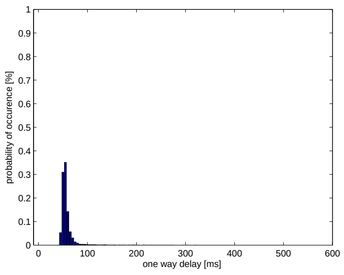
(a) Packet inter-departure time 2 ms



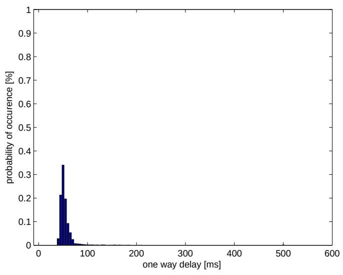
(b) Packet inter-departure time 5 ms



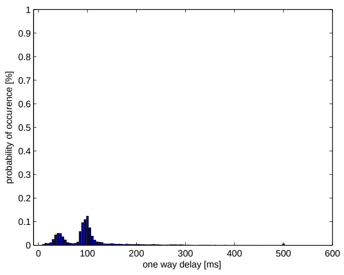
(c) Packet inter-departure time 10 ms



(d) Packet inter-departure time 20 ms



(e) Packet inter-departure time 50 ms



(f) Packet inter-departure time 100 ms

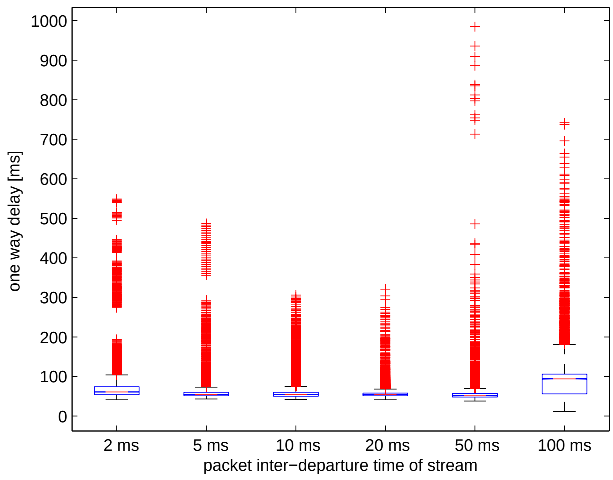
Figure 4.13: UMTS one-way delay from Internet PC to mobile node.

the different data streams show similar characteristics. The histograms of the distributions rise very fast in the beginning, and the majority of the measured OWD values of each stream are in the bins containing values between 50 ms and 75 ms. In the developing of the graph, all distributions show a small tail which is significantly decreased for bin sizes higher than 100 *ms*. An analysis of important statistical parameters like the minimum, maximum, mean, and the median values of the OWD measurements for each stream gives a more detailed view on the OWD distributions (cf. Table 4.4). The minimum values are between 11 *ms* (stream F) and 43 *ms* (stream B) which is quite similar for all tested data streams, whereas characteristics like maximum and mean values differ much more. The maximum values are in the range from 306 *ms* (stream C) up to 985 *ms* (stream E), and the mean values are in the range from 56.71 *ms* (stream E) up to 100.34 *ms* (stream F). Consistent with the observations in Figure 4.13, the median values of the streams are in the same range from 51 *ms* (stream E) to 94 *ms* (stream F).

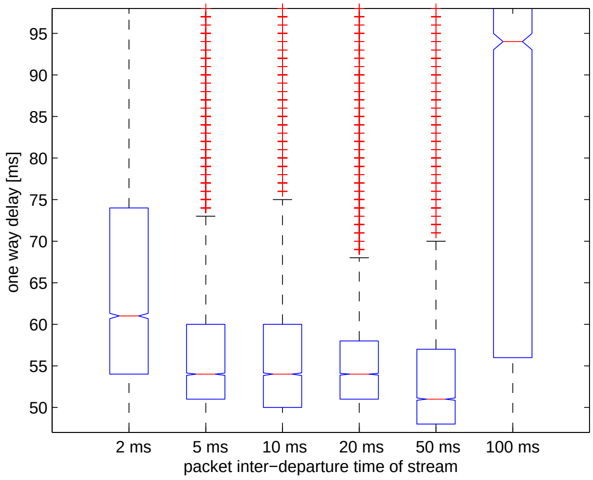
| inter-departure time | 2 ms | 5 ms | 10 ms | 20 ms | 50 ms | 100 ms |
|---|---|---|---|---|---|---|
| min. [ms] | 41 | 43 | 42 | 41 | 38 | 11 |
| max. [ms] | 549 | 487 | 306 | 321 | 985 | 742 |
| mean [ms] | 72.52 | 61.05 | 61.00 | 57.69 | 56.71 | 100.34 |
| median [ms] | 61 | 54 | 54 | 54 | 51 | 94 |

Table 4.4: Statistical data for the OWD measurements Internet PC to mobile node.

The pretended similarities of the distributions shown in Figure 4.13 might lead to the conclusion, that the packet inter-departure time of the packets of each data stream does not significantly influence the resulting OWD. To give an answer on this question, the distributions of the streams are compared using boxplots including the 95%-confidence interval around the median. In case the confidence intervals around the median value of each distribution do not overlap, the distributions differ significantly with respect to the median. Figure 4.14 shows these boxplots for the streams of Figure 4.13. Figure 4.14a shows the complete boxplots for the six data streams, including also the outliers. As the values of these outliers are relatively large, the details around the median values are shown in Figure 4.14b, which enlarges the area of interest of Figure 4.14a. The notches



(a) Boxplots for all generated streams.



(b) Boxplots for all generated streams (enlarged area).

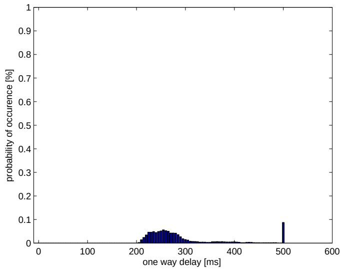
Figure 4.14: Boxplots for UMTS one-way delay from Internet PC to mobile node.

of the boxes show the size of the 95%-confidence interval of the corresponding median value, and it is obvious, that these confidence intervals of the different OWD measurements do not overlap. Thus, the distributions of the OWD measurements of the streams can be considered as significantly different. The size of the notches gives the size of the confidence interval, and Figure 4.14b shows also, that the size of the confidence interval is quite small, which is an indicator for taking enough measurement values into account for evaluation in order to provide a representative test.

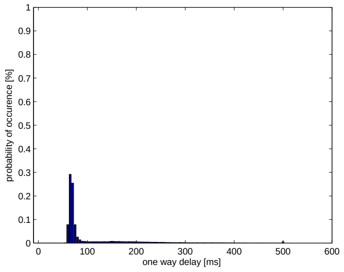
Nevertheless, with respect to the application of mobile robotics, the results are important. The minimum delay investigated for this communication direction is about 11 – 43 *ms*. The maximum delay which is measured has no really guaranteed upper bound, and outliers occur with values up to 985 *ms*. These measurements include only successfully transmitted data packets. The packet loss which occurred during these experiments will be analyzed in the following section.

In addition to the results above, now the opposite direction of sending data is investigated in order to get a complete analysis of the communication channel capabilities. Now, the mobile node creates data, and the Internet PC is the receiving node. These measurements use the same data streams as the above measurements (cf. Table 4.3). Figure 4.15 shows the UMTS one-way delay from *mobile node to Internet PC*. Compared to the one-way delays from *Internet PC to mobile node* (shown in Figure 4.13), the results of the measurements for the direction from *mobile node to Internet PC* (see Figure 4.15) have completely different characteristics. The most peculiar results are obtained from the measurement of stream A (Figure 4.15a). These measured OWD values show a quite broad distribution compared to the other results of Figure 4.15, and for stream A no clear peak of the OWD values exist. For the streams B to F, the OWD distributions show a similar behavior. But investigating details of the statistical data (cf. Table 4.5) shows a difference of about 10 *ms* between the minimum values of streams B,C and D,E. Also the differences in the maximum, the mean, and median values raise the suspicion that the OWD distributions are different and ask for a more detailed evaluation.

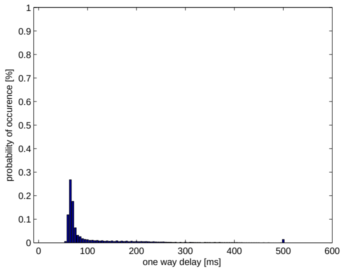
Therefore, again boxplots with the 95%-confidence interval around the median are used for the evaluation of the OWD distributions. Figure 4.16a shows the complete boxplots for all measurements, whereas Figure 4.16b displays an enlarged section around the median values and the corresponding confidence in-



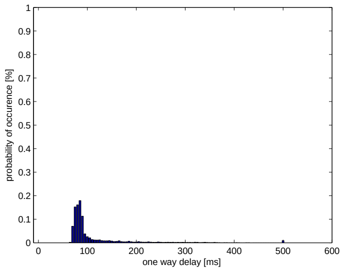
(a) Packet inter-departure time 2 ms



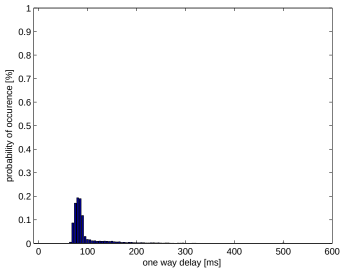
(b) Packet inter-departure time 5 ms



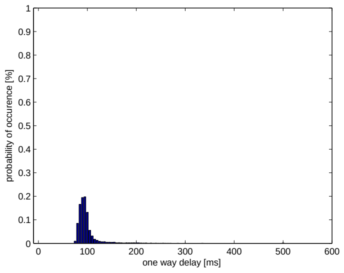
(c) Packet inter-departure time 10 ms



(d) Packet inter-departure time 20 ms

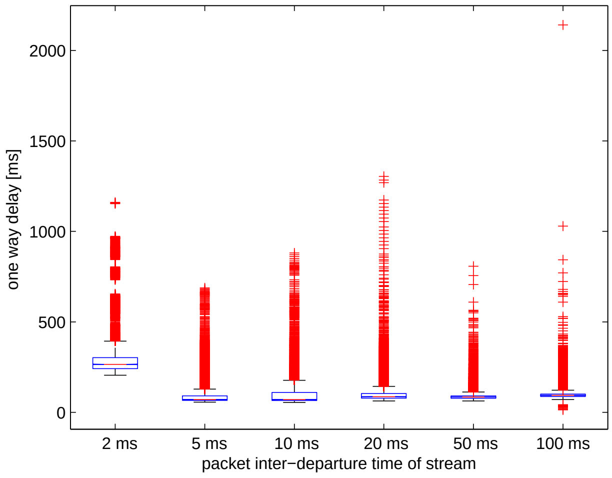


(e) Packet inter-departure time 50 ms

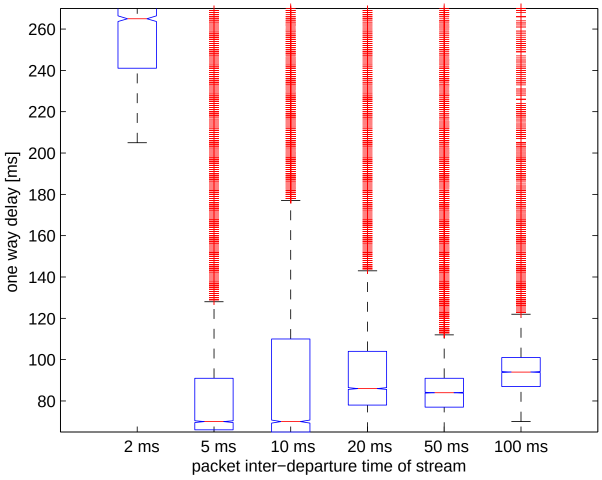


(f) Packet inter-departure time 100 ms

Figure 4.15: UMTS one-way delay from mobile node to Internet PC.



(a) Boxplots for all generated streams.



(b) Boxplots for all generated streams (enlarged area).

Figure 4.16: Boxplots for UMTS one-way delay from mobile node to Internet PC.

| inter-departure time | 2 ms | 5 ms | 10 ms | 20 ms | 50 ms | 100 ms |
|---|---|---|---|---|---|---|
| min. [ms] | 205 | 56 | 54 | 63 | 63 | 13 |
| max. [ms] | 1161 | 688 | 881 | 1304 | 807 | 2142 |
| mean [ms] | 311.79 | 103.23 | 110.88 | 114.94 | 98.65 | 102.98 |
| median [ms] | 265 | 70 | 70 | 86 | 84 | 94 |

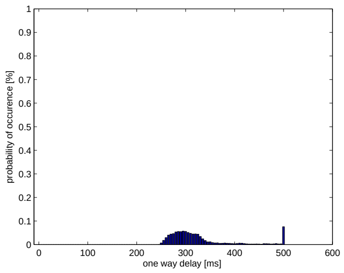
Table 4.5: Statistical data for the OWD measurements from mobile node to Internet PC.

tervals. As already expected due to the distributions shown in Figure 4.15, the median value of data stream A (2 *ms* packet inter-departure time) has a much higher value of 265 *ms* compared to the other streams. Also the confidence intervals of the OWD measurement value distributions do not overlap, which leads again to the conclusion, that also for this communication direction the distributions of the OWD values differ significantly.

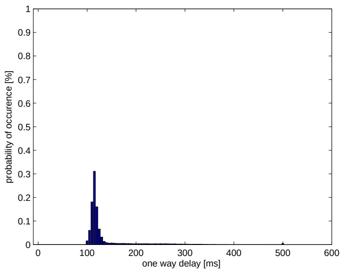
The above investigated communication directions are both using the identical physical medias. The difference between these two setups is just the direction in which the packets are sent. Now, the setup is changed in order to analyze the link characteristics for the connection *between two mobile nodes*. Both mobile nodes are connected to the Internet via UMTS link and a VPN connection is established as shown in Figure 4.3b. Again, the six different data streams (see Table 4.3) are transmitted.

Figure 4.17 shows the distributions of the OWD values for all these streams measured for the communication *between two UMTS mobile nodes*. Again, stream A shows a quite broad distribution of the OWD values without a clear peak. The distributions of the other streams show a clear peak with differences in their amplitude. For all data streams of this scenario, the measured OWD values are higher than the measured values of the scenarios of UMTS node and Internet PC. Comparing the statistical data of Table 4.6 and the data of the previous scenarios (cf. Tables 4.4 and 4.5) shows, that for the OWD between two UMTS mobile nodes the measured minimum (11 – 247 *ms*), maximum (674 – 1832 *ms*), mean (133.96 – 337.06 *ms*), and median values (111 – 306 *ms*) are significantly higher than the corresponding values of the previous measurement setups.

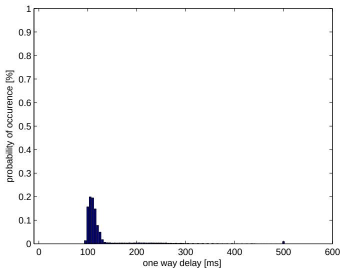
Also for this scenario, boxplots with a 95% confidence interval are used to



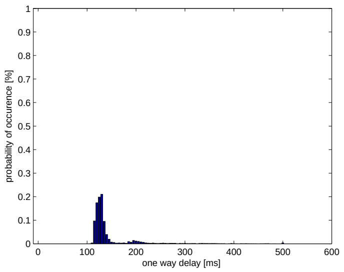
(a) Packet inter-departure time 2 ms



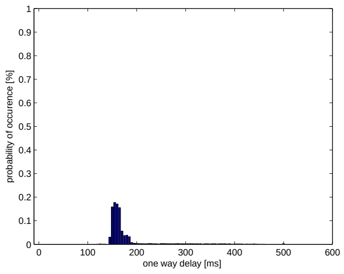
(b) Packet inter-departure time 5 ms



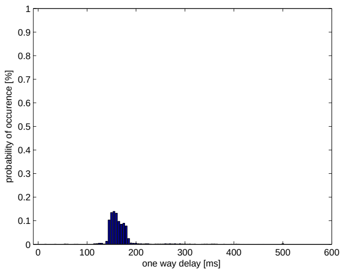
(c) Packet inter-departure time 10 ms



(d) Packet inter-departure time 20 ms



(e) Packet inter-departure time 50 ms



(f) Packet inter-departure time 100 ms

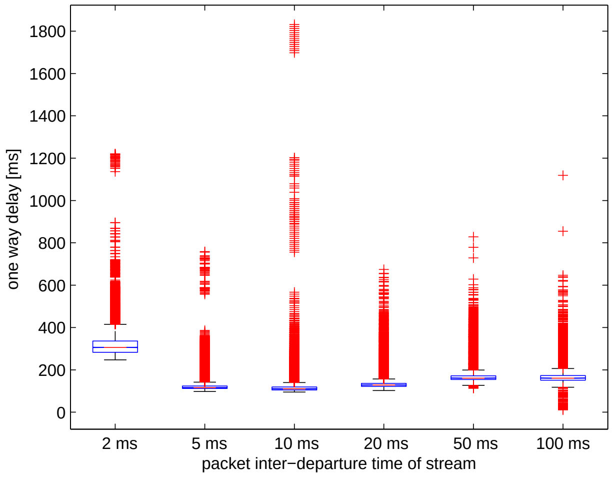
Figure 4.17: UMTS one-way delay from mobile node to mobile node.

| inter-departure time | 2 ms | 5 ms | 10 ms | 20 ms | 50 ms | 100 ms |
|---|---|---|---|---|---|---|
| min. [ms] | 247 | 98 | 95 | 102 | 113 | 19 |
| max. [ms] | 1220 | 759 | 1832 | 674 | 828 | 1119 |
| mean [ms] | 337.06 | 133.96 | 135.01 | 145.98 | 180.57 | 169.15 |
| median [ms] | 306 | 116 | 111 | 128 | 161 | 161 |

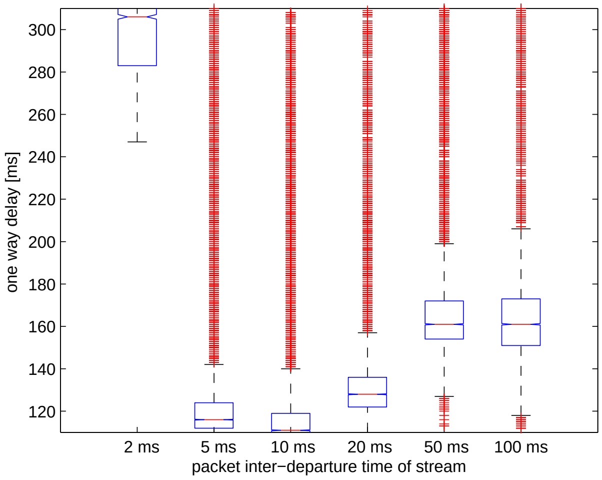
Table 4.6: Statistical data for the OWD measurements from mobile node to mobile node.

compare the OWD distributions (see Figure 4.18). Figure 4.18a shows the boxplots including the outliers for streams A to F, and Figure 4.18b shows the an enlarged area around the confidence intervals of each stream. Again, the confidence intervals of the OWD value distribution of each stream do not overlap, and the OWD distributions can be considered as significantly different.

A direct comparison of the results of all three different measurement setups is given in Figure 4.19, where the corresponding cumulative distribution functions (CDFs) of the OWD measurement value distributions are shown. The x-axis show the OWD in milliseconds, and the y-axis give the number of packets (in percent) having an equal or less OWD than the corresponding x-value. Figure 4.19a shows the CDFs of the results of the *Internet PC to mobile node* setup, Figure 4.19b shows the CDFs of the OWD distributions of the *mobile node to Internet PC* direction, and in Figure 4.19c, the CDFs for the results of the *mobile node to mobile node* communication scenario are given. In general, all displayed graphs have a high slope in the beginning, and at a particular point, the slope decreases very fast. This area, where the slope is decreasing fast, is of special interest, as the location of this area is a descriptor of the performance of the corresponding data stream. The more this area is in the left upper part of the figure, the better is the performance of the corresponding stream. Comparing the subfigures of Figure 4.19, it can be observed, that the location of this particular area differs from scenario to scenario. For the scenario which investigates the communication direction from *Internet PC to mobile node*, 90% of the packets of the streams B, C, D, E have an OWD of less than 77 *ms*. For stream A, 90% of the packets have an OWD of less than 100 *ms*, and for stream F the OWD for 90% of the packets is significantly higher with 161 *ms*. For all streams, 70% of the packets have an

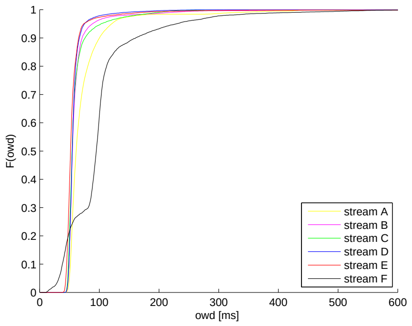


(a) Boxplots for all generated streams.

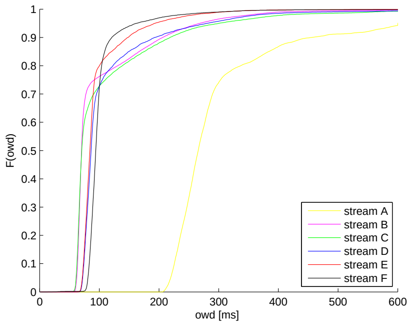


(b) Boxplots for all generated streams (enlarged area).

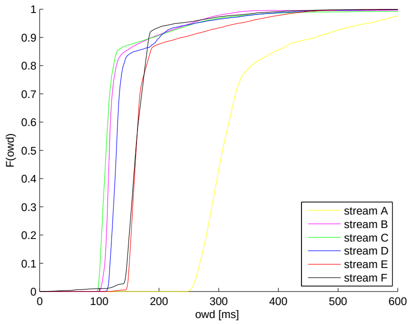
Figure 4.18: Boxplots for UMTS one-way delay from mobile node to mobile node.



(a) Internet PC to mobile node.



(b) Mobile node to Internet PC.



(c) Mobile node to mobile node.

Figure 4.19: Cumulative distribution function for different OWD measurement scenarios and different data streams.

OWD of less than 100 *ms*. Above the 80% threshold, the slopes of the graphs of streams A, B, C, D, and E decrease quickly and the best performance of this setup is achieved for stream F. The results for the direction from *mobile node to Internet PC* are shown in Figure 4.19b. The cumulative distribution functions of the measurements between two mobile nodes (cf. Figure 4.19c) can be split in three groups: group one with stream A, group two with streams B, C, D, and group three with streams E and F. The graph of stream A contains the highest OWD values, and for 80% of the packets an OWD of up to 365 *ms* has to be expected. For the streams B,C, and D, 80% of the packets have an OWD of less than 141 *ms*, and for the streams E and F 80% of the packets arrive with an OWD of less than 177 *ms*. It is also seen, that in the range between 86% and 90% of the packets on the y-axis, group two and group three are merged and the CDF of stream A still differs significantly from all other streams. Nevertheless, with

| Stream | Internet PC to mobile node | Mobile node to Internet PC | Mobile node to mobile node |
|---|---|---|---|
| A | $< 101$ *ms* | $< 457$ *ms* | $< 463$ *ms* |
| B | $< 72$ *ms* | $< 207$ *ms* | $< 189$ *ms* |
| C | $< 77$ *ms* | $< 219$ *ms* | $< 190$ *ms* |
| D | $< 67$ *ms* | $< 193$ *ms* | $< 202$ *ms* |
| E | $< 66$ *ms* | $< 148$ *ms* | $< 239$ *ms* |
| F | $< 161$ *ms* | $< 121$ *ms* | $< 184$ *ms* |

Table 4.7: Measured OWD for 90% of the packets.

respect to teleoperation of mobile robots experiences from experiments and implementations have shown, that a useful indicator is the threshold for 90% of the packets. A short summary of these results is given in Table 4.7, and it is observed that for this 90% threshold stream F performed best for the scenarios with the communication from *mobile node to Internet PC* and with the communication between two mobile nodes. In case data is sent from *Internet PC to mobile node*, stream F performs worst. In general, the smallest OWD values are measured for the Internet to mobile node scenario, and the largest OWD is observed in the *mobile node to mobile node* setup.

### 4.2.5 Packet Loss for Data Streams

In addition to the one-way delay evaluations of the previous section now the packet loss for these above situations are analyzed. The results are shown in Table 4.8. In general, for each of the three scenarios, a smaller packet inter-departure time results in a higher packet loss ratio. For all three measurement setups, stream A with 2 *ms* packet inter-departure time has the highest packet loss, and the largest packet loss ratio of 36.8% occurs in the scenario with the communication between two mobile nodes. Comparing streams B, C, D, and E on scenario level shows the highest packet loss level for the *mobile node to Internet PC* scenario, whereas the performance of streams B, C, D, and E for the scenarios *Internet PC to mobile node* and *mobile node to mobile node* shows only minor differences with respect to the packet loss. Within a particular scenario, streams B to E have the smallest packet loss ratio.

| inter-departure time | Internet PC to mobile node | Mobile node to Internet PC | Mobile node to mobile node |
|---|---|---|---|
| 2 ms (Stream A) | 3.8% | 15% | 36.8% |
| 5 ms (Stream B) | $< 0.1\%$ | 9% | 0.9% |
| 10 ms (Stream C) | 0.4% | 10% | 0.2% |
| 20 ms (Stream D) | $< 0.1\%$ | 7% | 0.1% |
| 50 ms (Stream E) | 0.1% | 2% | $< 0.1\%$ |
| 100 ms (Stream F) | 1.9% | 7% | 0.14% |

Table 4.8: Packet loss during OWD measurements.

### 4.2.6 One-Way Delay an Packet Loss for Single Messages

The previous sections investigated the OWD of data streams with packet inter-departure times between 2 and 100 milliseconds. This section is focused on the analysis of the OWD which is present for the transmission of single messages. Especially the areas of networked robotics and mobile robotics provide numerous applications for sending of single messages. An example for such a scenario is the periodical but relatively seldom transmission of important status messages. The bandwidth allocation for a UMTS data link depends on provider specific implementations and has a major effect on the applications in the area of mobile robotics. Thus, a detailed investigation of the link behavior is necessary in order to realize mobile robot teleoperation and networked mobile robots via UMTS links. The following experiment uses the same setup than the experiments in the previous sections. Each message contains 100 bytes of payload and the messages have a packet inter-departure time of 60 seconds. The experiment and respec-

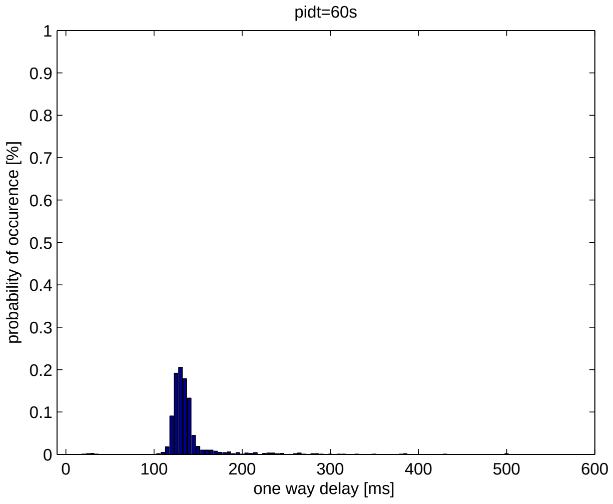


Figure 4.20: OWD for single messages sent over UMTS.

tively the measurements were done for a 24 hour period. Results of this experiments are shown in Figure 4.20 and Table 4.9 gives the relevant statistical data for the measured OWD values. The OWD mean value of 139.18 milliseconds and the OWD median value of 132 milliseconds is comparable to the mean and median values of the data streams with 10 and 20 milliseconds of packet inter-departure time. The minimum value of the single message experiment is relatively small and with 22 ms it is in the range of the data stream with 100 ms packet inter-departure time.

The packet loss for these measurements is very small. The test duration is 24 hours and each 60 seconds, a message is sent. Only 10 packets were lost during the above described experiment which corresponds to a packet loss of 0.69%.

| | |
|---|---|
| min. value | 22 |
| max. value | 682 |
| mean value | 139.18 |
| median | 132 |

Table 4.9: Statistical data for single messages sent via UMTS link.

### 4.2.7 Reflections on Transport Layer Behavior

The aspects of mobility and wireless communication are a big advantage which can be used in many application areas. Nevertheless, the mobility of nodes and wireless communication has also a big influence on the design requirements of all protocol layers. Usually, the mobility aspect of radio networks has already to be supported in the lowest layers (e.g. physical layer and data link layer of the OS-I/ISO model). But just implementing mechanisms for robust communication like error correction or packet retransmissions on these lower layers is not sufficient to provide connectivity and mobility to the upper layers, to the application, or to the users. In order to provide connectivity and mobility to applications in a transparent way, usually addressing mechanisms are implemented on the network layer (layer 3) and flow control is covered by the network layer (layer 4). Thus, also the well known protocols like UDP or TCP, which are often used on layer 4, have to support mobility aspects. On the one hand, the communication of applications can be based on connections as it is usually done by using TCP connections. On the other hand, also connectionless packet based communication can be used (e.g. UDP). Usually, connectionless protocols like UDP have a simple design and can easily be used in mobile networks. Of course, in this case, reliable packet transmission from source to destination is not guaranteed by the transport layer. Connection oriented protocols like TCP are much more complicated and do not work properly in mobile scenarios without additional mechanisms. In order to enable TCP and its flow control mechanisms to be used in mobile wireless networks, several mechanisms were developed in the past in order to allow the use of TCP in mobile radio networks. In cable networks packet loss usually results from link overload. These overload situations cause congestion at network nodes, and accordingly, the overloaded nodes will discard data packets. Thus, traditional TCP interprets packet loss as network overload and reduces the throughput in order to recover. Mechanisms like TCP slow-start, fast retransmit, or fast recovery (for more details please refer to [139]) enable TCP to react on the above described congestion situations and packet losses. All these mechanisms always assume, that the reason for the packet loss is an overload situation at one or more network nodes. In contrast to cable networks, the main reason for packet loss in wireless networks is usually the radio link, as it is much more susceptible to interference than cable connections. Thus, the bit error rate for data transmission via radio link is by orders of magnitude higher than the bit error rate while transmitting data via cable connections. Mechanisms on the physical layer and data link layer of wireless connections (e.g. forward error correction or packet retransmissions) can diminish the effect of physical interferences. Unfortunately, timing problems for protocols on higher layers will occur as the default protocol parameters are designed for the application in highly reliable cable networks and not for the application in mobile wireless environments. Despite the already described problems due to interferences on the radio link, also problems due to mobility aspects can arise, as a high grade of mobility results in a low link persistence with frequent changes of the network topology. In order to use TCP on the transport layer of mobile radio networks, new protocol versions of TCP with several extension were developed in the past. These extensions enable TCP to minimize some of the above described problems while being compatible to older existing TCP implementations. Among the recently developed approaches indirect TCP (I-TCP) (see [140]) is one of the oldest extensions. Here, the network is segmented by a *foreign agent* into a mobile segment running I-TCP, and a static network using traditional TCP. The segmentation of the network is a considerable disadvantage of I-TCP, as the end-to-end flow control is relayed by the *foreign agent* in a non-transparent way. Snooping TCP works transparent for the application and connects mobile nodes and static network nodes via an agent which is buffering the data. For more details please refer to [141] and [142]. For networks with intermittent connectivity mobile TCP (M-TCP) was developed (for more details please refer to [143]). Unfortunately, M-TCP requires an update of network hardware, as well as partial adjustments of the used protocols. The integration of the *fast retransmit / fast recovery* mechanisms into TCP (cf. [144]) prevents the activation of the *slow-start* mechanism in case of roaming, and due to the *selective acknowledgments* mechanism only corrupted or lost data packets are retransmitted [145]. For the application of TCP in 3G telecommunication networks several suggestions and recommendations were made [146]. In the past also suggestions were published with respect to the size of the *maximum transmission unit* (MTU) [147], *selective acknowledgments, limited transmit* [148], and *explicit congestion notification* [149]. Recently, also proxy-based architectures were developed to be used in mobile networks [150], whereas the capabilities of these performance enhancing proxies in combination with IP security mechanisms are limited. To improve the performance in networks with limited bandwidth RFC 3150 (*End-to-end Performance Implications of Slow Links*) [151] gives recommendations with respect to MTU and header compression, and RFC 3155 (*End-to-end Performance Implications of Links with Errors*) [152] lists recommendations for using TCP via wireless links.

In summary, the above described approaches can improve the performance of TCP in some special situations, but nevertheless, there is no existing universal approach to enable full TCP performance in all kinds and scenarios of mobile wireless networks. Thus, also the conclusion of [28], that the idea of a byte-stream oriented flow control protocol like TCP does not really fit with the ephemerality of links in wireless networks, can be understood. With respect to the application to mobile robotics, often real-time traffic has to be supported. In this case, TCP would not be the best choice, as retransmissions of lost packets would just restore obsolete data. Recently, TCP was also used in robotics applications in order to receive acknowledgments for successfully transmitted packets. But just using TCP to benefit from acknowledgments is not the real idea of this protocol as the main reason for using TCP is the implemented flow control. Thus, TCP should only be applied in case the provided flow control is really required.

### 4.2.8 Summary

In order to get a first impression of the link behavior, measurements of the packet inter-arrival times were done for three different setups. Data streams of 20, 100, and 200 kbytes/sec were measured for the communication from *UMTS device to Internet PC*, from *Internet PC to UMTS device*, and from *UMTS device to UMTS device*. For all measurements of the transmission direction towards UMTS devices, the packet inter-arrival times are clustered in slots each 10 milliseconds. The measurements of the opposite communication direction do not show this separation of the packet inter-arrival into time slots.

The above mentioned data streams with 20, 100, and 200 kbytes/sec were also used for round trip time measurements. The generated diagram gives an overview

of the round trip times which have to be expected in the present scenarios and also indications for the round trip time distribution of each stream in the corresponding scenario are given. Resulting round trip times strongly depend on the scenario and the data rate of the transmitted data stream and the mean values are in the range of 130 ms to 460 ms.
The three different setups were also used for the measurement of the one-way delay. The data streams which were generated for these experiments differ from the above generated streams in terms of packet size and packet inter-departure times. Here, the packet size is constant for all data streams and the packet inter-departure times are varied (2, 5, 10, 20, and 100 milliseconds). The one-way delay was analyzed for these different data streams in three the different scenarios. It was shown how these different data stream characteristics affect the one-way delays of the data packets of each stream. It could be clearly shown how the Internet data transmission influences the measured one-way delay times at the receiving node. The measured one-way delays are distributed relatively broad in case of transmitting packets from UMTS devices to the Internet PC with a packet inter-departure time of 2 ms, and one-way delays between 200 ms and 310 ms for the scenario *UMTS to Internet PC*), and respectively between 250 ms and 350 ms for the *UMTS to UMTS* scenario are measured. The distribution of the measured one-way delays of the same data stream in opposite transmission direction shows a completely different behavior and a clear peak at about 70 ms is visible. The one-way delays for all data streams in the *UMTS device to UMTS device* scenario were higher (all mean values above 130 ms) than in the other two scenarios, explicitly for the data stream with 2 ms packet inter-departure time. The analysis of the packet loss showed, that the packet loss is relatively small for the data streams with a packet inter-departure time of at least 5 ms, and with packet inter-departure times of 2 ms, the packet loss increases. Comparing the three scenarios shows that the highest packet loss of 36.9% occurs for the scenario *UMTS to Internet*. It was also observed, that the one-way delay for the data stream with 2 ms packet inter-departure time was much smaller for the transmission direction from Internet PC to UMTS device than in the opposite direction. Typical sensor data packets with a small packet size should not be sent with a packet inter-departure time smaller than 5 ms in order to keep the packet loss ratio and one-way delays low. For packet inter-departure times between 5 ms and 100 ms, one-way delays between 56 ms and 170ms have to be expected (depending on the scenario).

In addition to the data streams, also the one-way delay and the packet loss for the transmission of single messages was investigated. This experiment results in a very small packet loss of about 0.69% and the measured one-way delays have a mean value of about 139 ms.
Now, these results provide a basis to design systems which are able to provide reliable mobile robot teleoperation via links with the above mentioned one-way delay, round trip time, and packet loss characteristics. Details on such an application example are given in the next section.

## 4.3 Application: Mobile Robot Teleoperation via UMTS

This section provides detailed insights into different experiments in which mobile robots are controlled remotely via UMTS. Two different mobile robot platforms are used for these real world tests as the systems differ in the implementation of the communication between robot hardware and control PC. One platform uses the well known Player software framework [153] and transmits sensor data via TCP, the other platform was developed by the robotics team at the University of Würzburg and uses UDP for sensor data transmission.

### 4.3.1 Setup and Requirements

For these application scenarios, a Pioneer 3-AT robot and a MERLIN robot are teleoperated. The mobile robots are connected to a Laptop via UMTS (cf. setup in Figure 4.5). The mobile robot platform has a mini PC with a 1200MHz processor and 1GB RAM. The Pioneer Robot is equipped with a network video camera AXIS 221 and the MERLIN robot has an AXIS 213 pan-tilt-zoom camera. The cameras are connected to one of the network interfaces of the onboard PC and a motion JPEG video stream is forwarded via port 80 to the laptop. For the experiment, different resolutions of the video streams are used and the frame rate is limited to 7 frames per second for the setup without HSUPA, and 14 frames per second for the experiments with HSUPA respectively.

The Pioneer robot is interfaced with the Player software version 2.0.4. which provides communication between the client and the hardware. The client application is a Java program which provides a video image of the mobile robot's onboard camera and displays sensor data like obstacles which are detected by the

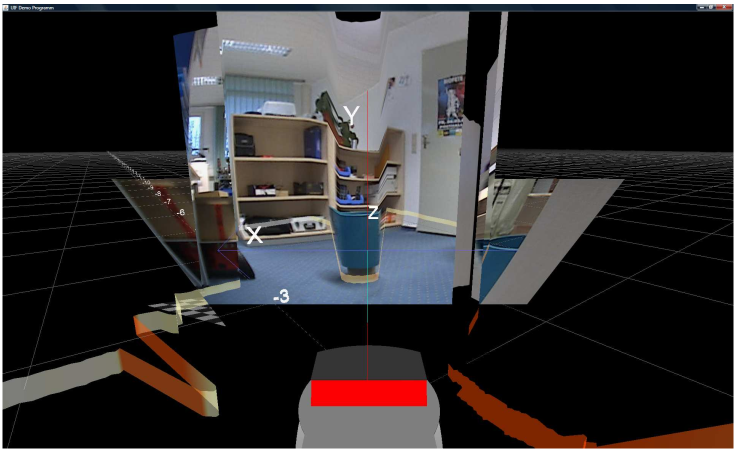


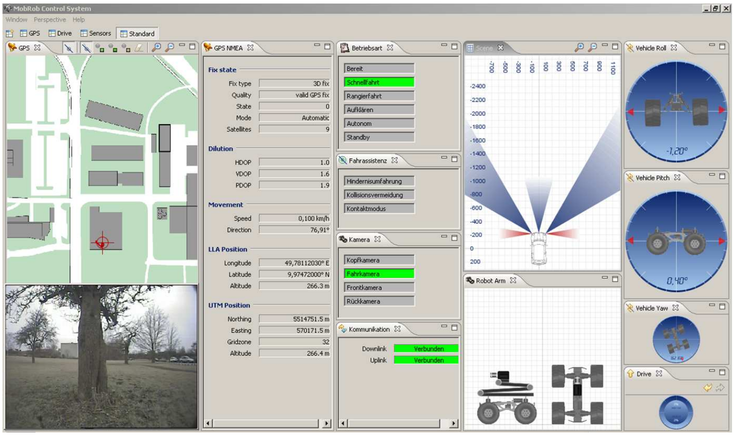
Figure 4.21: Mixed Reality Software Framework (cf. [154]).

laser scanner (see Figure 4.21). The client program sends the control commands with a bit rate of 1.2 kbytes/sec over a TCP connection, and the video stream is also transmitted via TCP.

The MERLIN robot is controlled via the MRCS system (cf. Figure 4.22) which was developed by the University of Würzburg. For more information on the MRCS please refer to [111]. MERLIN sends a motion JPEG video stream and sensor data of the ultrasonic sensors. MERLIN Robot and MRCS exchange sensor data and command data with UDP, and the video stream is transmitted via TCP.

For both mobile robot systems, the UMTS connection of the service provider Vodafone[5] is used which supports the bandwidths shown in Table 4.10. The robots, as well as the control clients connect to an OpenVPN server, and the communication between each robot and its client is realized within this virtual private network.

[5]http://www.vodafone.de/ (09.02.2010)



**Fix state**

| | |
|---|---|
| Fix type | 3D fix |
| Quality | valid GPS fix |
| State | 0 |
| Mode | Automatic |
| Satellites | 9 |

**Dilution**

| | |
|---|---|
| HDOP | 1.0 |
| VDOP | 1.6 |
| PDOP | 1.9 |

**Movement**

| | |
|---|---|
| Speed | 0,100 km/h |
| Direction | 76,91° |

**LLA Position**

| | |
|---|---|
| Longitude | 49,78112030° E |
| Latitude | 9,97472000° N |
| Altitude | 266.3 m |

**UTM Position**

| | |
|---|---|
| Northing | 5514751.5 m |
| Easting | 570171.5 m |
| Gridzone | 32 |
| Altitude | 266.4 m |

| | |
|---|---|
| Downlink | Verbunden |
| Uplink | Verbunden |

Figure 4.22: MRCS for remote controlling the MERLIN robot (cf. [111]).

| | downlink | uplink |
|---|---|---|
| UMTS | 384 KBit/s | 64 KBit/s |
| HSDPA stage 1 | 1.8 MBit/s | 384 KBit/s |
| HSDPA stage 2 and HSUPA | 3.6 MBit/s | 1.8 MBit/s |
| HSDPA stage 3 and HSUPA | 7.2 MBit/s | 3.6 MBit/s |

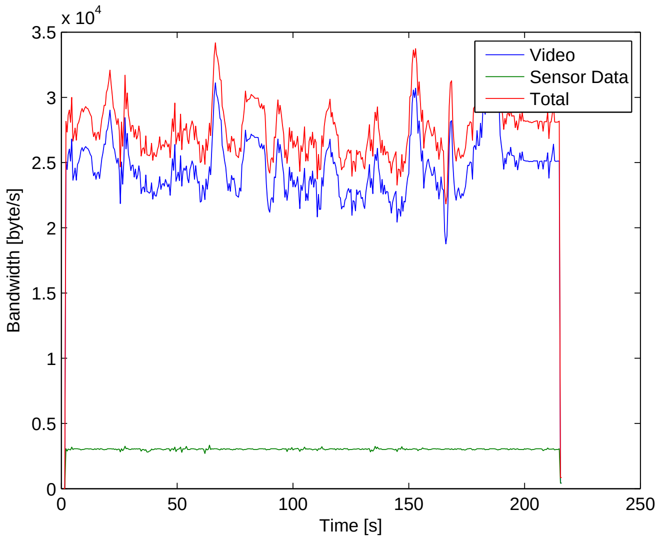
Table 4.10: UMTS uplink and downlink bandwidths of Vodafone (September 2009).

### 4.3.2 UMTS uplink without HSUPA

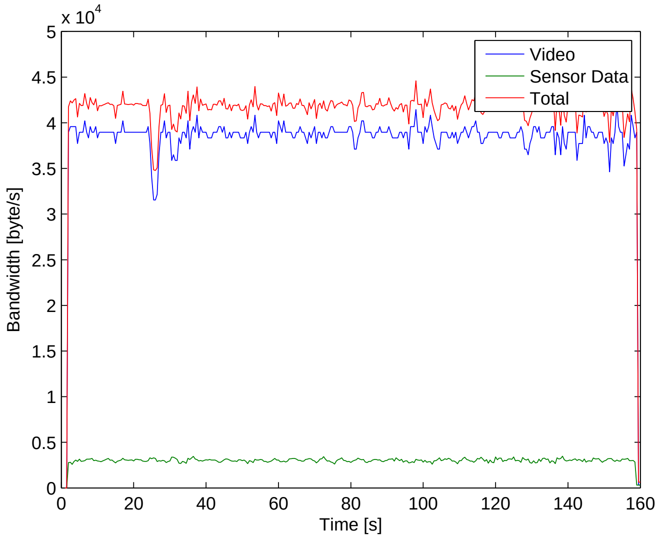
With UMTS stage 1, HSUPA is not available and an uplink bandwidth of only 384 KBis/s is available. For these scenarios, the video streams of the MERLIN robot are configured with a resolution of 178x144 and 356x288. The Pioneer robot is set up for a video transmission with a resolution of 160x120 and 240x180. Each video stream is limited to a maximum of 7 frames per second in order to avoid an apparent overload of the link.

Figure 4.23 shows the sensor data and video traffic on the uplink from the MERLIN robot to the control PC. Figure 4.23a shows the used uplink bandwidth for a video stream with a resolution of 178x144, and Figure 4.23b shows the data for a video stream with a resolution of 356x288. In both cases, also sensor data in terms of status information about the robot and distance information from four ultra sonic sensors is transmitted from the mobile robot to the control PC. The video data is transmitted via TCP, and UDP is used to send the sensor data packets. As it is shown in the Figures, the sensor data stream is using an almost constant bit rate of about 2.8 up to 3.0 kbyte/s. With the smaller resolution of 178x144, the video data requires a variable bandwidth between 18 and 35 kbyte/s. For this setup, only about 73% of the available uplink bandwidth is used, and the delay of the video stream which is experienced by the user is insignificant. In the second setup of the MERLIN robot, the higher video resolution of 356x288 results in an overall link utilization of 35 up to 43 kbyte/s which corresponds to approximately 90% of the available uplink bandwidth. Here, the user experiences a noticeable delay of the video stream but comfortable teleoperation is still possible. The transmission of a video stream with higher resolutions or higher frame rates will overload the uplink channel and results in significant delays of the video stream.

The results for the Pioneer robot scenario are shown in Figure 4.24. The Pioneer robot is sending sensor data from the laser range finder and video data to the control PC in both of the displayed setups. All sensor data is transmitted within the Player framework and a TCP connection is used. The data transmission of the laser range finder requires a bandwidth of about 12 kbyte/s. In case of the transmission of the video stream with a lower resolution of 160x120 (Figure 4.24a), an overall bandwidth of about 28 kbyte/s is used, and for the video with the size of 240x180 (Figure 4.24b), an overall uplink bandwidth of approximately 38 kbyte/s is required. The delays of the video stream for the 160x120 resolution are not noticeable, and for the 240x180 resolution a small delay of the video image is



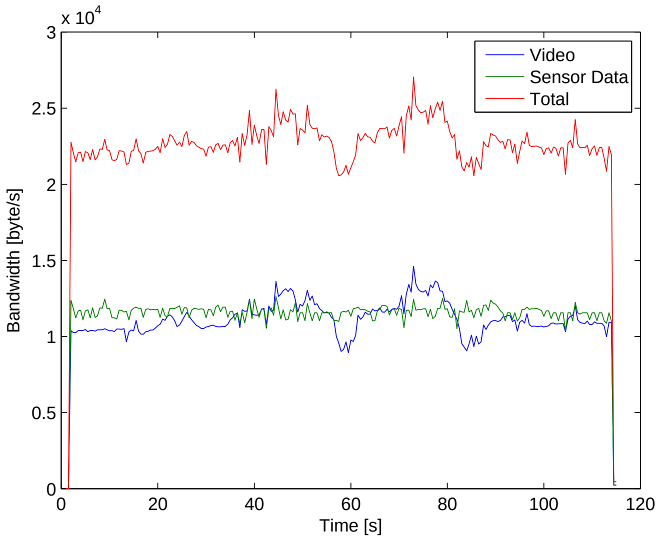
(a) Video resolution of 178x144 at 7fps



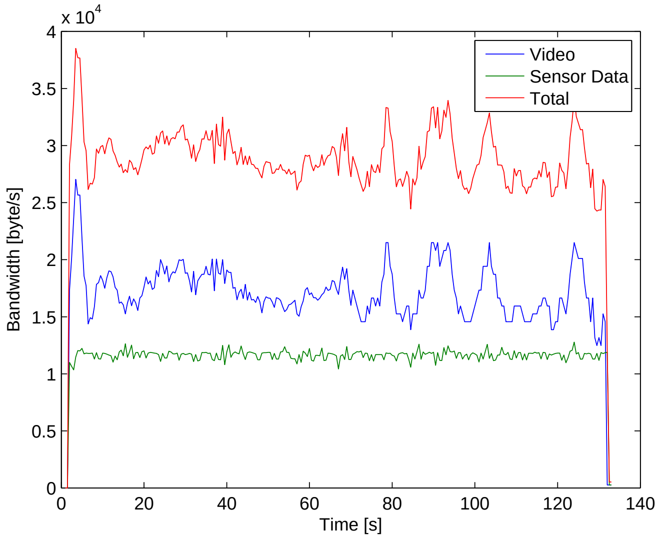
(b) Video resolution of 356x288 at 7fps

Figure 4.23: MERLIN uplink with standard UMTS.

present but does not influence the teleoperation of the mobile robot in a negative way. Using higher video resolutions or higher frame rates of the video stream without additional mechanisms will obviously exceed the available uplink bandwidth and thus, these settings are not investigated, as the resulting transmission delays are not acceptable for teleoperation anymore.



(a) Video resolution of 160x120 at 7fps

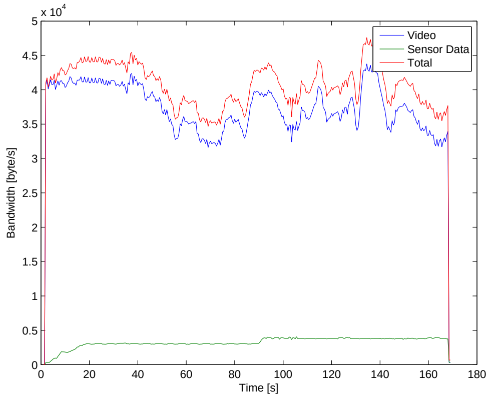


(b) Video resolution of 240x180 at 7fps

Figure 4.24: Pioneer uplink with standard UMTS.

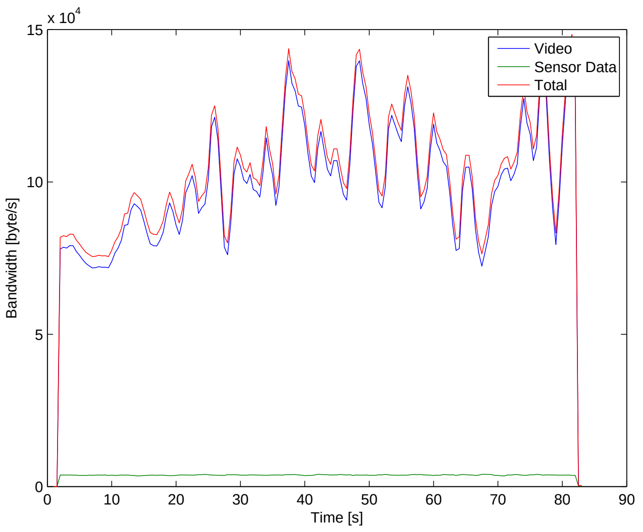
### 4.3.3 UMTS uplink with HSUPA

The above section uses a standard UMTS uplink without the support of HSUPA. Thus, the upload bandwidth in the previous setup is limited to 384 KBit/s which is one of the major limitations with respect to the increase of teleoperation capabilities. Now, UMTS is configured with HSUPA which allows for a theoretical maximum upload bandwidth of 1.8 MBit/s. For this setup, the MERLIN robot is configured for three different setups and sends video data either with a resolution of 192x144, 384x288, or 768x576 with 14 frames per seconds. Thus, MERLIN sends video data with different resolutions and data of the ultra sonic sensors and system status information. The amount of traffic which is generated by the sensor and status data is similar to the corresponding previous setup (2.8 - 3.0 kbyte/s). The overall bandwidth requirement for the 192x144 video stream varies between 30 kbyte/s and 48 kbyte/s (see Figure 4.25a). For the video resolution of 384x288, an upload capacity between 70 kbyte/s and 145 kbyte/s is necessary (cf. Figure 4.25b), and for the video resolution of 768x576, an overall uplink bandwidth between 80 kbyte/s and 160 kbyte/s is required. Thus, in the setup with the video stream of the high resolution of 768x576, approximately 71% of the theoretical maximum uplink bandwith is used, and for all these configurations, the delay of the video stream did not limit the teleoperation capabilities of the mobile robot.

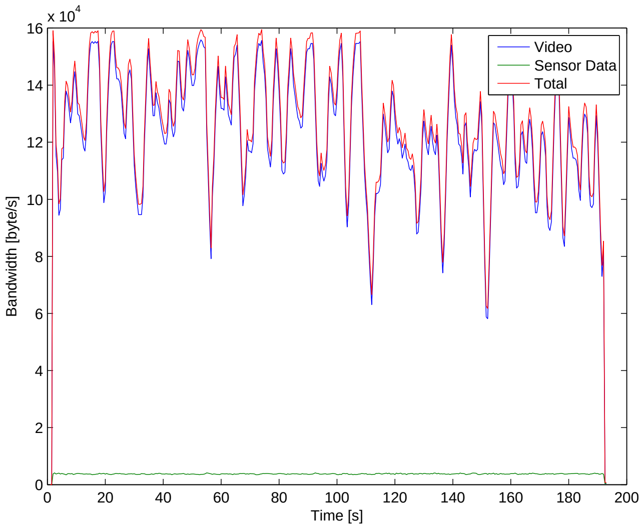


(a) 192x144 at 14fps

Figure 4.25: MERLIN uplink with HSUPA.



(b) 384x288 at 14fps



(c) 768x576 at 14fps

Figure 4.25: MERLIN uplink with HSUPA (cont.).

## 4.4 Discussion of the Results

The results obtained in the test runs with the artificially generated command traffic (see Section 4.2) give a detailed overview of the channel behavior. As soon as the broadband communication mode is used, data is delivered at the mobile robot and at the operator's PC constantly. A noticeable outcome are the frequent packet inter-arrival times at 10 ms and multiples of 10 ms with a noticeable peak at the 20 ms inter-arrival times. This behavior is similar in all investigated scenarios. Also for the transmission of single packets (cf. Section 4.2.6), the performance of UMTS was highly satisfying and showed different behavior compared to previous analyses of older UMTS versions. The measured packet loss during all tests, the observed round trip times, and their distributions provide the base for the implemented mobile robot teleoperation test scenarios.

The investigated setup without HSUPA clearly shows, that the video image resolution and frame rate has to be limited in order to achieve a comfortable robot teleoperation. For the Pioneer robot, a maximum resolution of 240x180 can be sent together with the sensor data of the laser range finder and for the MERLIN robot, the video resolution can be set to 356x288. The difference in the supported video resolution results from different types of transmission protocols (UDP vs. TCP) and the different size of the data sets of the sensors. The delay $\Delta t$ of the transmission of a video frame which has to be expected can be estimated by the following equation:

$$\Delta t = t_{cam} + OWD + \frac{n}{s_{uplink}} + \frac{n}{MTU} \cdot PIAT \qquad (4.2)$$

$t_{cam}$ is the processing time of the video camera to generate the video frame and is in the order of magnitude of some 10 milliseconds. OWD is the one-way delay for data transmission via UMTS link. The duration of the transmission depends on the characteristics of the data stream, as well as on the amount of data and the link utilization respectively. This parameter is investigated in details in Section 4.2.4. For the present data streams, one-way delays between 121 ms and 207 ms have to be expected according to the distributions given in Section 4.2.4 (cf. Table 4.7). $n$ gives the frame size in bytes, and $s_{uplink}$ is the uplink bandwidth in byte/s. In case of the used motion JPEG stream the size of each video frame always depends on the content of the image and the given image quality. Both aspects directly influence the resulting compression ratio. The approximate frame sizes for the used video streams are given in Table 4.11. The Maximum Transmission

| resolution | frame size |
|---|---|
| MERLIN, 178x144 | 2.5 - 4.4 kbyte |
| MERLIN, 356x288 | 4.8 - 5.8 kbyte |
| MERLIN, 160x120 | 1.3 - 2.1 kbyte |
| MERLIN, 240x180 | 1.7 - 3.6 kbyte |

Table 4.11: Approximate frame sizes of the used video streams.

Unit (MTU) is defined by the used protocol layers and specifies maximum packet size which can be transmitted without fragmentation. As the scenario setups use virtual private networks in addition to UDP or TCP, an effective payload on application level of 1237 bytes per packet is achieved. PIAT is the packet inter-arrival time and depends also on the data characteristics and the amount of data which is sent. A detailed analysis of this parameter is given in Section 4.2.2. The packet inter-arrival time is in the range of 2 ms to 25 ms, whereas more than 80% of the packets have an inter-arrival time of less than 15 ms.
An upper bound of the delay of the transmission of a single frame cannot be given as also packet losses might occur. But nevertheless, with Equation 4.2 and the values from above, an estimation of the minimum delay can be achieved:

$$50ms + 121ms + \frac{3kbyte}{48kbyte} \cdot 1000ms + \left\lceil \frac{3kbyte}{1237byte} \right\rceil \cdot 10ms > 263ms$$

The above equation gives just an estimation of the minimum delay which has to be expected. The one-way delay, as well as the packet inter-arrival time follow different distributions which also depend on the characteristics of the transported traffic (see Sections 4.2.4 and 4.2.2). Also the frame sizes of the video stream are variable depending on the content of the images. Figure 4.26 shows the packet inter-arrival times of a video stream of the above application scenario. The video source generates packets which are larger than the maximum payload at application layer. Thus, a video frame is fragmented and the send buffers are continuously filled which leads to continuously sending packets. On the receiver's side, the peaks of the packet inter-arrival times each 10 ms are also visible which complies to the results of Section 4.2.2. Another aspect which should always be

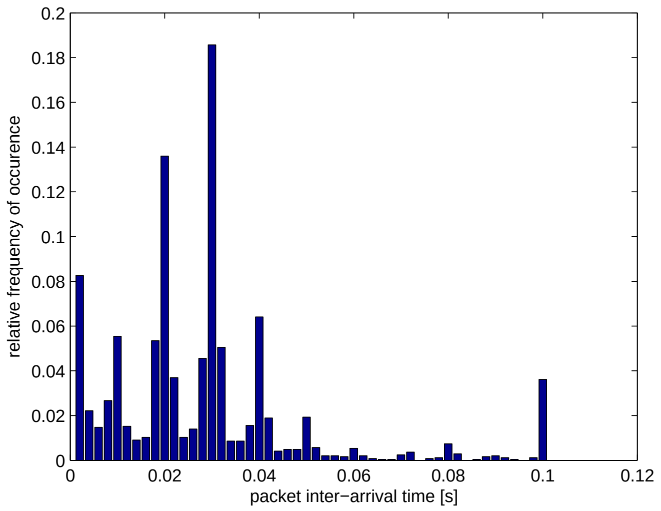


Figure 4.26: Packet inter-arrival of the video data coming from the mobile robot.

kept in mind is the simultaneous use of TCP and UDP which is done in the setup with the MERLIN robot. In case of an overload situation of the link, the UDP data stream will dramatically decrease the throughput of the TCP stream which will lead to a very high delay of the video stream (cf. Section 4.2.7). In order to maximize the link utilization with respect to the teleoperation capabilities of the telerobotic system, the mechanisms which are presented in Section 3.3.2 can also be used here. Using the HSUPA technology allows the video transmission with a much higher resolution and quality. Nevertheless, in case of exceeding the link capacity, the considerations with respect to simultaneously using TCP and UDP are also valid.

The measurements lead to the conclusion, that a defined traffic shaping is a suitable approach to use the characteristics of the UMTS link more efficient and to increase the quality of teleoperation (e.g. better video quality or less packet loss). Approaches for this idea are already mentioned in another context (e.g. using network feedback - see Section 3.3.2) and showed to be useful techniques [9][10][6]. The above presented results showed clearly that UMTS is a well suited communication technology for the teleoperation of mobile robots to allow broadband communication between multiple hardware and human entities, and will be in the research focus of networked robotics in near future.

# 5 Conclusions and Future Directions

Nowadays, many different communication technologies are available for the different application scenarios of networked robotic systems. However, there is still research effort needed to gain knowledge how architectures and protocol stacks can be dovetailed in order to achieve maximum performance of the complete telerobotic system. In this monograph, teleoperation of mobile robots via IP based communication networks is investigated in three application environments with different communication technologies: *teleoperation of robotic systems via Internet, wireless ad-hoc networks of mobile robots,* and *mobile robot teleoperation using UMTS*. The achieved results are demonstrated with real hardware setups for each of the used technologies.

Chapter 2 presents the design and architecture for remote learning units of mobile robots and several developed experiments. The experiments cover topics like modeling of a vehicle with non-holonomic constraints, path planning for a mobile robot, kinematics of a differential drive robot, speed control for a differential drive robot, design of a PID controller for speed control of a mobile robot, algorithms for obstacle avoidance, and navigation and path planning. Section 2.2 presents an architecture which is developed in the frame of this work in order to allow for the setup of real hardware experiments, whereas the operated hardware itself can be placed at distant locations spread all over the world. Section 2.3 describes an experiment setup, where a plant is connected via Internet communication to a controller. Using the example of the Quanser SRV02 hardware experiment, a controller is designed which allows for controlling this system via Internet. Real hardware experiments are performed and demonstrate the successful realization of the controller. Access to the learning units is offered world wide via the Internet, and the presented remote laboratory architecture also supports different quality levels of for the communication link. Thus, the experiments can be used via high-speed broadband Internet connections, as well as via old-fashioned modem dial-up connections. Integrated modules for security, user management, scheduling, and resource management allow a flexible scaling in terms of the number of users and the number of provided learning units. The key issue for successful international tele-laboratories is the platform independence which is guaranteed by the used Java WebStart technology and the user interface which is accessible through the web portal. The architecture which was developed in the frame of this work was realized and implemented within the European Union funded project *International Virtual Laboratory on Mechatronics*. In order to achieve the combination of interactivity and teleoperation via low bandwidth links several approaches were implemented. With respect to the structure of the implemented experiments, the experiment design aims on low bandwidth requirements. Nevertheless, also interactivity via low bandwidth links is supported by data compression and virtualization techniques. Bandwidth intensive sensor data (e.g. from PMD cameras or laser range finders) can be compressed. Also the transmitted video stream can be substituted by a sensor data stream while the video image at the teleoperation interface is replaced by a virtual reality environment which is augmented by sensor data. The developed architecture is syntonized to the communication link characteristics and the requirements which result from the corresponding hardware experiments, including also administrative aspects of remote laboratories. It supports all of the above presented mechanisms and allows students to resume experiments after a communication loss without loosing experiment data. Thus, it was possible to perform tests under real operating conditions, where mobile robots located in Germany were teleoperated successfully form India and vice versa. These tests showed the advantages of supporting resuming experiment sessions and the adaptive use of bandwidth by the developed tele-laboratory system and until today the remote laboratory of the University of Würzburg is successfully used in education. The experiences showed, that the proposed architecture provides a very reliable and scalable system for tele-education in robotics which can be used via Internet from everywhere.

Chapter 3 shows how to setup wireless ad-hoc networks for mobile robot communication. An analysis of different ad-hoc routing protocols in defined networked robotic scenarios allows for a parameter tuning of the corresponding ad-hoc routing protocols. Relevant parameters are identified and their working principles are investigated. The investigated setups are representative for networked robotic scenarios but they are also worst case scenarios with respect to the protocol design philosophies. It is demonstrated that appropriate parameter tuning improves the network performance in these situations. Also the average duration of communication losses is reduced. The protocol behavior and the influence of different protocol parameters are also analyzed in scenarios with multi-hop communication. All results and the proposed improvements are demonstrated with several example applications. The example of teleoperation of mobile robots via wireless ad-hoc networks consists of two applications: the integration of a small size helicopter into an IP based ad-hoc network, and mobile robot teleoperation for a navigation and exploration task. The integration of a small size UAV into a mobile ad-hoc networks demonstrates the interaction of mechanisms for local autonomy on application layer and the network behavior. An architecture which allows for commanding the UAV also in partially disconnected networks is described. Thus, local autonomy features allow for bridging communication gaps which might occur unpredictable. The navigation and exploration task uses wireless multi-hop communication while the robot is remote controlled. Mechanisms like a traffic shaping algorithm and network feedback are implemented. Thus, an adaptive video quality and a jitter reduction of the video data packets are achieved, and the bandwidth requirements of the video stream are adjusted on demand based on the link quality and load status which is derived from the network feedback. These mechanisms are tested and evaluated in real world operating conditions and it is shown, that the required network performance can be provided. The performance improvements with respect to the teleoperation capabilities of the telematic system are measured by the time required for task completion and the required number of stops of the mobile robot which are initiated by the user. The developed mechanisms reduce both significantly, the time duration for task completion, as well as the number of required stops. It is shown that the performance of the analyzed ad-hoc routing protocols in the presented worst case scenario can be improved by appropriate parameter tuning. In combination with additionally implemented features like traffic shaping mechanisms or local autonomy features, it is demonstrated, how the performance of the complete telerobotic system can be improved.

The idea of investigating the UMTS technology for telerobotic scenarios in Chapter 4 of this work is also motivated by the ideas presented in publications with respect to the car-to-car communication via wireless ad-hoc networks and via mobile telephone networks. Existing investigations and simulation studies showed unfavorable behavior of UMTS when being used in scenarios typical for networked robotics (e.g. for transmitting single messages containing important information). Often, an unacceptable delay or packet loss was observed in past work. Nevertheless, none of these existing analyses provided information detailed enough to decide how to apply the UMTS technology for mobile robot teleoperation or for other networked robotic scenarios. The work presented in this monograph specifically addresses this application area and derived important knowledge for setting up telerobotic systems. In the frame of this work, a real hardware testbed was installed using real communication hardware, as well as real mobile robot hardware in order to provide most realistic behavior of the analyzed telerobotic system. Also different communication setups (e.g. UMTS node to UMTS node, or UMTS node to Internet PC) are investigated. The analysis of packet inter-arrival times, one-way delays, packet losses, and round trip times for data streams, as well as for single message transmissions provided meaningful results and extend the knowledge from existing work. The gained link characteristics directly lead to guidelines to setup telerobotic systems and provide an estimation of the delay which has to be accepted for this type of setup. These results are verified in demonstrations of teleoperation scenarios with two different mobile robot teleoperation systems with different communication architectures. The traffic shaping and local autonomy mechanisms which are developed in the frame of Chapter 3 can also be used in the context of mobile robot teleoperation via UMTS.

In the course of this monograph, three important areas of mobile robot communication are analyzed. Mechanisms on application level are developed, protocol parameters are evaluated, and link analyses are performed in order to seamlessly integrate telerobotic systems and the communication link. The gained knowledge in the area of *mobile robot teleoperation via Internet* allows for the implementation of an international tele-laboratory which provides access to distributed hardware experiments also in low bandwidth environments. In contrast to existing implementations, the developed system supports resuming of sessions, as well as the adaptation to bandwidth constraints. It is also shown by an example, how the link analysis contributes to the successful realization of reliable networked control algorithms. In future work, educational content can now be developed based on the results of this work and it can be integrated into the developed architecture. The results in the area of *ad-hoc networks of mobile robots* extend existing expertise regarding the application of existing ad-hoc routing protocols in networked robotic scenarios. As originally, the development of routing protocols for wireless ad-hoc protocols was driven by premises from telecommunication networks, the scenarios of networked mobile robots which are analyzed in this work provide new impulses. The acquired results of the protocol parameter tuning can positively influence the development of new ad-hoc routing protocols and can also be considered in the further development of existing ad-hoc routing mechanisms. The results in the area of *teleoperation of mobile robots via UMTS* present important information about the link behavior of this technology in telerobotic applications. Also here, the analyzed applications have different demands on the communication than the services initially considered during the planning and development phase of UMTS. The elaborated findings can be applied to different application areas like car-to-car communication, tele-education, networking of industrial plants and machines, as well as emergency management and search and rescue scenarios. Another very interesting and new technological approach is the development of small satellites in order to setup ad-hoc networks in space. Results of the present work can be used to derive methods and algorithms which can be applied in future realizations of these space-based ad-hoc networks.

# Bibliography of the Author

**— Book Chapters —**

[1] F. Zeiger, N. Krämer, M. Sauer, and K. Schilling, *Springer Lecture Notes in Electrical Engineering - Informatics in Control, Automation and Robotics*, ch. Mobile Robot Teleoperation via Wireless Multihop Networks - Parameter Tuning of Protocols and Real World Application Scenarios. Springer, 2009.

**— Journals —**

[2] F. Zeiger, M. Schmidt, and K. Schilling, "Remote Experiments with Mobile Robot Hardware via Internet at limited Link Capacity," *IEEE Transactions on Industrial Electronics*, vol. 56, pp. 4798–4805, December 2009.

[3] F. Zeiger, M. Schmidt, and K. Schilling, "Die Realisierung von Robotik und Mechatronik Tele-Experimenten via Internet für internationale Studenten," *Global Journal of Engineering. Education*, vol. 10, no. 3, pp. 245–256, 2006.

[4] F. Zeiger and K. Schilling, "Remote Laboratories on Mobile Robotics," *IEEE Computer Society - Technical Committee on Learning Technology (LTTC)*, vol. 7, July 2005.

**— Conference Papers —**

[5] F. Zeiger, F. Kempf, and K. Schilling, "UMTS One Way Delay Characterization for Mobile Robot Teleoperation," in *Proceedings of the 41st International Symposium on Robotics (ISR) and 6th German Conference on Robotics (ROBOTIK)*, 2010.

[6] F. Zeiger, M. Sauer, L. Stolz, and K. Schilling, "Integrating Teams of Mobile Robots into Wireless Ad-hoc Networks," in *Proceedings of IFAC Workshop on Networked Robotics (NetRob)*, (Golden, USA, CO), October 2009.

[7] F. Zeiger, M. Sauer, L. Stolz, and K. Schilling, "Teleoperation of a Mobile Robot via UMTS Link," in *Proceedings of the 6th International Conference on Informatics, Automation and Robotics (ICINCO)*, (Milan, Italy), July 2009.

[8] M. Sauer, F. Zeiger, and K. Schilling, "A Simulation Setup for Communication Hardware in the Loop Experiments," in *Proceedings of the 6th International Conference on Informatics, Automation and Robotics (ICINCO)*, (Milan, Italy), July 2009.

[9] F. Zeiger, M. Sauer, and K. Schilling, "Video Transmission with Adaptive Quality based on Network Feedback for Mobile Robot Teleoperation in Wireless Multi-Hop Networks," in *Proceedings of the 5th International Conference on Informatics, Automation and Robotics (ICINCO)*, (Funchal, Portugal), May 2008.

[10] F. Zeiger, M. Sauer, and K. Schilling, "Intelligent Shaping of a Video Stream for Mobile Robot Teleoperation via Multihop Networks in Real-World Scenarios," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Nice, France), September 2008.

[11] F. Zeiger, N. Krämer, and K. Schilling, "Commanding Mobile Robots via Wireless Ad-Hoc Networks - A Comparison of Four Ad-Hoc Routing Protocol Implementations," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (Pasadena, USA, CA), May 2008.

[12] F. Zeiger, N. Krämer, M. Sauer, and K. Schilling, "Challenges in Realizing Ad-Hoc Networks based on Wireless LAN with Mobile Robots," in *Proceedings of the Workshop on Wireless Multihop Communications in Networked Robotics (WMCNR)*, (Berlin,Germany), April 2008.

[13] F. Zeiger, N. Krämer, and K. Schilling, “Parameter Tuning of Routing Protocols to Improve the Performance of Mobile Robot Teleoperation via Wireless Ad-Hoc Networks,” in *Proceedings of the 5th International Conference on Informatics, Automation and Robotics (ICINCO)*, (Funchal, Portugal), May 2008.

[14] F. Zeiger, C. Selbach, B. Ruderisch, and K. Schilling, “An Application Protocol to Integrate a Small Size Helicopter into an IP based Ad-Hoc Network,” in *Proceedings of the International Conference on Robot Communication and Coordination (ROBOCOMM)*, (Athens, Greece), October 2007.

[15] C. Herrmann, F. Zeiger, L. Ma, C. Selbach, and K. Schilling, “Design and test of an autonomous helicopter for multi-vehicle cooperation,” in *Proceedings of the IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*, (Toulouse, France), September 2007.

[16] F. Zeiger, M. Schmidt, and K. Schilling, “A Flexible Extension for Pico-Satellite Communication Based on Orbit Operation Results of UWE-1,” in *Proceedings of the 57th International Astronautical Congress*, (Valencia, Spain), October 2006.

[17] M. Schmidt, F. Zeiger, and K. Schilling, “Design and Implementation of In Orbit Experiments for the Pico Satellite UWE-1,” in *Proceedings of the 57th International Astronautical Congress (IAC)*, (Valencia, Spain), October 2006.

[18] G. Zysko, F. Zeiger, K. Schilling, and M. Sauer, “Remote Laboratory Experiments Addressing Path Planning For Mobile Robots,” in *Proceedings of the 2nd International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, (Barcelona, Spain), pp. 431–434, September 2005.

[19] F. Zeiger and K. Schilling, “Remote Laboratories for Mobile Robots,” in *Proceedings of the 6th International Workshop on Research and Education in Mechatronics (REM)*, (Annecy, France), July 2005.

[20] F. Zeiger and K. Schilling, “Experiments with Mobile Robots in Tele-Education,” in *Proceedings of the 3. Deutsche e-learning Fachtagung Informatik (DeLFI)*, (Rostock, Germany), September 2005.

[21] K. Schilling and F. Zeiger, “Experiments with remote Equipment in Tele-Education,” in *Proceedings of the 2nd Joint Workshop of Cognition and Learning through Media-Communication for Advanced e-Learning 2005, Sophia University*, (Tokyo, Japan), pp. 217–220, 2005.

[22] M. Sauer, F. Zeiger, F. Driewer, and K. Schilling, “Remote control on mobile robots in low bandwidth environments,” in *Proceedings of the 2nd International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, (Barcelona, Spain), pp. 163–168, September 2005.

[23] B. Herbst, F. Zeiger, M. Schmidt, and K. Schilling, “UWE-1: A Pico-Satellite to Test Telecommunication Protocols,” in *Proceedings of the 56th International Astronautical Congress (IAC)*, (Fukuoka, Japan), October 2005.

[24] Y. Aoki, R. Barza, F. Zeiger, B. Herbst, and K. Schilling, “The CubeSat Project at the University of Würzburg - The Mission and System Design,” in *Proceedings of the Space Technology Education Conference*, (Aalborg, Denmark), April 2005.

[25] M. Menth, J. Milbrandt, and F. Zeiger, “Elastic Token Bucket - A Traffic Characterization for Time-Limited Bursty Traffic,” in *Proceedings of the 12th GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB) together with 3rd Polish-German Teletraffic Symposium (PGTS)*, 2004.

## General References

[26] R. M. Metcalfe and D. R. Boggs, “Ethernet: Distributed packet switching for local computer networks,” *Communications of the ACM*, vol. 19, pp. 395–404, July 1976.

[27] J. Naughton, *Brief History of the Future: Origins of the Internet*. Orion, London; 2nd Revised edition, October 2000.

[28] J. Schiller, *Mobilkommunikation*. Addison-Weseley, Pearson Education Limited, 2003.

[29] L. G. Roberts, "Multiple computer networks and intercomputer communication," in *Proceedings of the first ACM symposium on Operating System Principles*, pp. 3.1–3.6, 1967.

[30] K. Goldberg, M. Mascha, S. Gentner, N. Rothenberg, C. Sutter, and J. Wiegley, "Desktop tele-operation via the world wide web," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 654–663, 1995.

[31] K. Taylor and J. Trevelyan, "Australia's telerobot on the web," in *Proceedings of the 26th International Symposium on Industrial Robotics*, pp. 39–44, October 1995.

[32] T. B. Sheridan, *Telerobotics, Automation and Human Supervisory Control*. The MIT Press, 1992.

[33] L. Conway, R. A. Volz, and M. W. Walker, "Teleautonomous systems: Projecting and coordinating intelligent action at a distance," *IEEE Transactions on Robotics and Automation*, vol. 6, pp. 146–158, April 1990.

[34] T. B. Sheridan, "Space teleoperation through time delay:review and prognosis," *IEEE Transactions on Robotics and Automation*, vol. 9, pp. 592–606, October 1993.

[35] T. Tzyn-Jong and K. Brady, "A framework for the control of time-delayed telerobotic systems," in *Proceedings of Symposium on Robot Control*, vol. 2, pp. 599–604, September 1998.

[36] K. Brady and T. Tzyn-Jong, "Internet-based remote teleoperation," in *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*, 1998.

[37] K. J. Brady and T. Tzyn-Jong, "Intelligent remote teleoperation," *IEEE Potentials*, vol. 18, pp. 14–16, August-September 1999.

[38] K. Goldberg, M. Mascha, S. Gentner, J. Rossman, N. Rothenberg, C. Sutter, and J. Wiegley, "Beyond the web: manipulating the real world," in *In Second International World-Wide Web Conference: Mosaic and the Web*, vol. 28, pp. 209–219, 1995.

[39] K. Goldberg, B. Chena, R. Solomon, S. Bui, B. Farzin, J. Heitler, D. Poon, and G. Smith, "Collaborative teleoperation via the internet," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2000.

[40] R. Simmons, J. Fernandez, R. Goodwin, S. Koenig, and J. O'Sullivan, "Xavier: An autonomous mobile robot on the web," *IEEE Robotics and Automation Magazine*, 1999.

[41] O. Michel, P. Saucy, and F. Mondada, "KhepOnTheWeb: An experimental demonstrator in telerobotics and virtual reality," in *In Proceedings of International Conference on Virtual Systems and MultiMedia VSMM*, pp. 90–98, 1997.

[42] R. Siegwart and P. Saucy, "Interacting Mobile Robots on the Web," in *Proceedings of IEEE International Conference on Robotics and Automation*, 1999.

[43] R. Oboe and P. Fiorini, "Issues on internet-based teleoperation," in *Proceedings of International IFAC Symposium on Robot Control SYROCO*, September 1997.

[44] J. E. Lloyd, J. S. Beis, D. K. Pai, and D. G. Lowe, "Model-based telerobotics with vision," in *Proceedings of International Conference on Robotics and Automation*, vol. 2, pp. 1297–1304, April 1997.

[45] P. G. Backes, G. K. Tharp, and K. S. Tso, "The web interface for telescience (WITS)," in *Proceedings of International Conference on Robotics and Automation*, vol. 1, pp. 411–417, April 1997.

[46] P. G. Backes, K. S. Tso, and G. K. Tharp, "Mars pathfinder mission internet-based operations using WITS," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, pp. 284–291, 1998.

[47] P. G. Backes, G. Rabideau, K. S. Tso, and S. Chien, “Automated planning and scheduling for planetary rover distributed operations,” in *Proceedings of International Conference on Robotics and Automation*, vol. 2, pp. 984–991, May 1999.

[48] G. Hirzinger, K. Landzettel, B. Brunner, I. Schaefer, M. Fischer, M. Grebenstein, N. Sporer, J. Schott, M. Schedl, and C. Deutrich, “DLR’s Robotics Lab - Recent Developments in Space Robotics,” in *Proceedings of the Fifth International Symposium Artificial Intelligence, Robotics and Automation in Space ISAIRAS*, p. 25, 1999.

[49] G. Hirzinger, “Mechatronics and telerobotics - key technologies for a new robot generation in space and terrestrial applications,” in *Proceedings of the ICARCV The Sixth International Conference on Control, Automation, Robotics and Vision, Singapore, Dec. 5-8*, 2000.

[50] G. Hirzinger, B. Brunner, K. Landzettel, N. Sporer, J. Butterfaß, and M. Schedl, “Space robotics-dlr’s telerobotic concepts, lightweight arms and articulated hands,” *Autonomous Robots*, vol. 14, pp. 127–145, 2003.

[51] C. Preusche, “Haptics and Telerobotics,” in *Proceedings of the IST 2006 - Haptex Workshop*, 2006.

[52] S. Dormido (ed.), “Internet Based Control Education,” in *Proceedings of the IFAC Workshop Madrid*, 2001.

[53] K. Goldberg and R. Siegwart, *Beyond Webcams - An Introduction To Online Robots*. MIT Press, 2001.

[54] K. Schilling and H. Roth, “Mini-robots for teaching in telematics and control,” in *Proceedings EAEEIE 1998*, 1998.

[55] K. Schilling, H. Roth, and R. Lieb, “Remote Control of a Mars Rover via Internet to Support Education in Control and Teleoperations,” *Acta Astronautica*, vol. 50, pp. 173–178, 2002.

[56] K. Schilling and H. Roth, “Mobile robots for education in telematics, control and mechatronics,” in *Proceedings IFAC World Congress 1999, Section Control Education*, pp. 211–215, 1999.

[57] K. Schilling, H. Roth, and O. Rösch, “Fernsteuerung und Tele-Sensorik für mobile Roboter in Raumfahrt, Industrie und Ausbildung,” *at - Automatisierungstechnik*, vol. 8/2001, pp. 366–372, 2001.

[58] K. Schilling and H. Roth, eds., *Proceedings of International Workshop Tele-Education in Mechatronics Based on Virtual Laboratories, Weingarten*, 2001.

[59] J. B. Weinberg and X. Yu, “Special Issue "Robotics in Education": Low-Cost Platforms for Teaching Integrated Systems,” *IEEE Robotics and Automation Magazine*, vol. 10, No. 2, 2003.

[60] X. Yu and J. B. Weinberg, “Special Issue "Robotics in Education": New Platforms and Environments,” *IEEE Robotics and Automation Magazine*, vol. 10, No. 3, 2003.

[61] A. Casals and A. Grau, eds., *Proceedings of 1st Workshop on Robotics Education and Training, Weingarten*, 2001.

[62] D. Schmid, G. Gruhler, and A. Fearns, eds., *eLearning*. Verlag Europa-Lehrmittel, 2003.

[63] L. Greenwald and J. Kopena, “Mobile Robot Labs,” *IEEE Robotics and Automation Magazine*, vol. 10, No 2, June 2003.

[64] S. Kariya, “Online Education Expands and Evolves,” *IEEE Spectrum*, vol. 40, No. 5, 2003.

[65] “Special issue on internet and online robots for telemanipulation,” in *Int. Journal of Robotic Systems* (R. Marin, P. Sanz, and K. Schilling, eds.), vol. 22, 2005.

[66] S. H. Patel and T. Sobh, “An experiment in distance engineering education,” *IEEE Robotics & Automation Magazine*, vol. Vol. 2, Nr.3, pp. 91–98, 2006.

[67] A. Bicchi, A. Caiti, L. Pallottino, and G. Tonietti, “Online Robotic Experiments for TeleEducation at the University of Pisa,” *Int. Journal of Robotic Systems*, vol. 22 (4), pp. 217 – 230, 2005.

[68] K.-B. Sim, K.-S. Byun, and F. Harashima, "Internet Based Tele-operation of an Intelligent Robot with Optimal 2-Layer Fuzzy Controller," *IEEE Transactions on Industrial Electronics*, vol. 53 (4), pp. 1362 – 1372, August 2006.

[69] K. Schilling, T. Adami, and R. D. Irwin, "A Virtual Laboratory for Space Systems Engineering Experiments," in *Proceedings 15 th IFAC Symposium on Automatic Control in Aerospace, Bologna*, pp. 326–331, 2001.

[70] R. Marin, P. J. Sanz, P. Nebot, and R. Wirz, "A Multimodal Interface to Control a Robot Arm via Web: A Case Study on Remote Programming," *IEEE Transactions on Industrial Electronics*, vol. 52 (6), pp. 1506– 1520, December 2005.

[71] W. E. White, M. Zywno, W. Brimley, and D. O. Northwood, "Integrating New and Traditional Pedagogies for Effective Interactive Learning in Engineering Education," in *Proceedings 12th Canadian Conference on Engineering Education*, pp. 223–228, 2001.

[72] D. Hercog, B. Gergic, S. Uran, and K. Jezernik, "A DSP-Based Remote Control Laboratory," *IEEE Transactions on Industrial Electronics*, vol. 54 (6), pp. pp. 3057–3068, Dec 2007.

[73] J. Fernández, R. M. Prades, and R. W. Gonzalez, "Online competitions: An open space to improve the learning process," *IEEE Transactions on Industrial Electronics*, vol. 54, No.6, pp. pp. 3086–3093, Dec 2007.

[74] K. Schilling, H. Roth, and O. Rösch, "Mobile Mini-Robots for Engineering Education," *Global Journal of Engineering Education*, vol. 6, pp. 79–84, 2002.

[75] G. Zysko, R. Barza, K. Schilling, L. Ma, and F. Driewer, "Remote Experiments on Kinematics and Control of Mobile Robots," in *Proceedings of 5th Symposium on Intelligent Autonomous Vehicles*, 2004.

[76] G. Zysko, R. Barza, and K. Schilling, "TeleLab using non-holonomic car-like Mobile Robot," in *Proceedings of IFAC Workshop Grenoble*, 2004.

[77] G. Vossen and P. Westerkamp, “E-learning as a web service,” in *7th International Database Engineering and Applications Symposium (IDEAS 2003)*, pp. 242–249, 2003.

[78] X. Qiu and A. Jooloor, “Web service architecture for e-learning,” *Journal of Systemics, Cybernetics and Informatics, Special Issue for EISTA 2004 International Conference on Education and Information Systems: Technologies and Applications*, vol. Volume 3, Issue 5, p. 25, 2006.

[79] F. Lin, P. Holt, S. Leung, and Q. Li, “A multiagent and service-oriented architecture for developing adaptive e-learning systems,” *International Journal Cont. Engineering Education and Lifelong Learning*, vol. Vol. 16, Nos. 1/2, pp. pp 77–91, 2006.

[80] R. Leiner, “Tele-experiments via internet a new approach for distance education,” in *11th Mediterranean Electrotechnical Conference, MELECON, 2002*, pp. 538–541, 2002.

[81] A. Hopp, D. Schulz, W. Burgard, A. B. Cremers, and D. Fellner, “Virtual reality visualization of distributed tele-experiments,” in *Proc. of 1998 IEEE Industrial Electronics Conference (IECON98), Aachen Germany*, 1998.

[82] K. Schilling, H. Roth, and C. Spilca, “A tele-experiment on rover motor control via internet,” *Journal of Robotic Systems*, vol. 22 (3), pp. 123–130, March 2005.

[83] K. Zakova, “Experiments with inverted pendulum: from simulations to remote control,” *International Journal of Education and Information Technologies*, vol. 1 (3), pp. 142–147, 2007.

[84] T. F. Junge and C. Schmid, “Web-based remote experimentation using a laboratory-scale optical tracker,” in *American Control Conference 2000, Chicago, Illinois, USA*, June 2000.

[85] H. Brandtstädter, J. Schneider, and F. Freyberger, “Hardware and software components for a new internet-based multimodal tele-control experiment with haptic sensation,” in *Proceedings of EuroHaptics 2004, Munich, Germany*, June 2004.

[86] K. Schilling and Q. Meng, “The MERLIN vehicles for outdoor applications,” in *Proceedings SPIE Unmanned Ground Vehicle Technology IV, Orlando*, 2002.

[87] A. Khamis, M. P. Vernet, and K. Schilling, “A remote experiment on motor control of mobile robots,” in *Proceedings of the 10th Mediterranean Conference on Control and Automation*, 2002.

[88] K. J. Åström and B. Wittenmark, *Computer Controlled Sytsems - Theory and Design*. Prentice Hall Information and System Sciences Series, 1997.

[89] L. Gomes and S. Bogosyan, “Current trends in remote laboratories,” *IEEE Transactions on Industrial Electronics*, vol. 56, pp. 4744–4756, December 2009.

[90] IEEE Standard for Information Technology, “IEEE 802.11, IEEE Standard for Information Technology - Telecommunications and information exchange between systems - Local and Metropolitan Networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications,” tech. rep., The Institute of Electrical and Electronics Engineers, IEEE, 1999.

[91] K. Pahlavan and P. Krishnamurthy, *Principles of Wireless Networks - A Unified Approach*. Prentice Hall, January 2002.

[92] R. W. Chang, “Synthesis of band-limited orthogonal signals for multichannel data transmission,” *Bell System Technical Journal*, vol. 46, pp. 1775–1796, 1966.

[93] F. Kuo and N. Abramson, “Some advances in radio communications for computers," The Aloha System,” Tech. Rep. B73-1, Univ. of Hawaii, Honolulu, Hawaii, March 1973.

[94] W. Franz, “Car-to-Car Communication for Active Safety and Other Applications,” in *ATA EL Conference, Parma, Italy*, June 2004.

[95] H. Füßler, S. Schnaufer, M. Transier, and W. Effelsberg, “Vehicular Ad-Hoc Networks: From Vision to Reality and Back,” in *4th IEEE/IFIP Annual Conference on Wireless On demand Network Systems and Services (WONS ’07), Obergurgl, Austria*, January 2007.

[96] A. Boukerche, H. A. B. F. Oliveira, E. F. Nakamura, and A. A. F. Loureiro, “Vehicular Ad Hoc Networks: A New Challenge for Localization-Based Systems,” *Computer Communications*, vol. 31 (12), pp. 2838–2849, 2008.

[97] Y.-B. Ko and N. H. Vaidya, “Location-aided routing (LAR) in mobile ad hoc networks,” in *4th annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 66–75, 1998.

[98] M. Witt and V. Turau, “BGR: Blind Geographic Routing for Sensor Networks,” in *Proceedings of the Third International Workshop on Intelligent Solutions in Embedded Systems*, pp. 51 – 61, May 2005.

[99] P. Jacquet and T. Clausen, “Optimized Link State Routing Protocol (OLSR).” IETF RFC 3626, October 2003.

[100] C. E. Perkins, E. M. Belding-Royer, and S. Das, “Ad hoc On-Demand Distance Vector (AODV) Routing.” IETF RFC 3561, July 2003.

[101] D. B. Johnson, D. A. Maltz, and J. Broch, “The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4.” IETF RFC 4728, February 2007.

[102] A. Pandey, M. N. Ahmed, N. Kumar, and P. Gupta, *Lecture Notes in Computer Science - High Performance Computing (HIPC)*, vol. 4297/2006, ch. A Hybrid Routing Scheme for Mobile Ad Hoc Networks with Mobile Backbones, pp. 411–423. Springer Berlin / Heidelberg, 2006.

[103] V. Park and S. Corson, “Temporally-Ordered Routing Algorithm (TORA).” Version 1, Functional Specification, Internet Draft, IETF MANET Working Group, June 2001.

[104] C. Perkins, E. Royer, S. Das, and M. Marina, “Performance comparison of two on-demand routing protocols for ad hoc networks,” *IEEE Personal Communications*, vol. 8, no. 1, pp. 16–28, 2001.

[105] I. D. Chakeres and E. M. Belding-Royer, “AODV Routing Protocol Implementation Design,” in *Proceedings of the International Workshop on Wireless Ad hoc Networking (WWAN), Tokyo, Japan*, pp. 698–703, March 2004.

[106] D. I. Program, “Internet Protocol Specification.” IETF RFC 791, September 1981.

[107] D. B. Johnson and D. A. Maltz, “Dynamic source routing in ad hoc wireless networks,” *Mobile Computing*, vol. Vol. 353, 1996.

[108] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, “A performance comparison of multi-hop wireless ad hoc network routing protocols,” in *Mobile Computing and Networking*, pp. 85–97, 1998.

[109] R. Bagrodia, R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, B. Park, and H. Song, “Parsec: A Parallel Simulation Environment for Complex Systems,” *IEEE Computer*, vol. 31(10), pp. 77–85, October 1998.

[110] J. Macker, “Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations.” RFC 2501, January 1999.

[111] D. Eck, M. Stahl, and K. Schilling, “The Small Outdoor Rover MERLIN and its Assistance System for Tele-Operations,” in *Proceedings of International Conference on Field and Service Robotics, Chamonix, France*, 2007.

[112] M. Musial, U. W. Brandenburg, and G. Hommel, “Success of an Inexpensive System Design: The Flying Robot MARVIN,” in *16th Int. Unmanned Air Vehicle System Conference (UAVs)*, 2001.

[113] T. Chung, L. Cremean, W. B. Dunbar, Z. Jin, E. Klavins, D. Moore, A. Tiwari, D. van Gogh, and StephenWaydo, “A platform for cooperative and coordinated control of multiple vehicles,” in *3rd Conference on Cooperative Control and Optimization*, 2002.

[114] A. Ollero, J. Alcazar, F. Cuesta, F. Lopez-Pichaco, and C. Nogales, “Helicopter Teleoperation for Aerial Monitoring in the COMETS Multi-UAV System,” in *3rd IARP Workshop on Service, Assistive and Personal Robots, Madrid (Spain)*, 2003.

[115] R. Vidal, O. Shakernia, H. J. Kim, H. Shim, and S. Sastry, "Multi-agent probabilistic pursuit-evasion games with unmanned ground and aerial vehicles," *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 662–669, October 2002.

[116] M. Musial, G. Hommel, U. W. Brandenburg, E. Berg, M. Christmann, C. Fleischer, C. Reinicke, V. Remuß, S. Rönnecke, and A. Wege, "MARVIN - Technische Universität Berlin's Flying Robot Competing at the IARC'99," *Fachgespräche Autonome Mobile Systeme AMS 99*, vol. 15, pp. 324–333, 1999.

[117] V. Remuß and M. Musial, "Communication System for Cooperative Mobile Robots using Ad-Hoc Networks," in *5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, 2004.

[118] N. Pezeshkian, H. G. Nguyen, and A. Burmeister, "Unmanned ground vehicle radio relay deployment system for non-line-of-sight operations," in *Proceedings of the 13th IASTED International Conference on Robotics and Applications, August 29-31, Würzburg, Germany, RA2007*, 2007.

[119] A. Winfield and O. Holland, "The application of wireless local area network technology to the control of mobile robots," *Microprocessors and Microsystems*, vol. 23, pp. 597–607, 2000.

[120] A. Ollero, G. Hommel, J. Gancet, L. Gutierrez, D. Viegas, J. Wiklund, and M. Gonzalez, "COMETS: A Multiple Heterogeneous UAV System," in *IEEE International Workshop on Safety, Security, and Rescue Robotics SSRR, Bonn, Germany*, 2004.

[121] M. Frank, S.Busch, P. Dietz, and K. Schilling, "Teleoperations of a Mobile Outdoor Robot with Adjustable Autonomy," in *Proceedings of the 2nd International Conference on Machine Intelligence,ACIDCA-ICMI 2005, Tozeur, Tunisia*, 2005.

[122] K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internets," in *Proceedings of the ACM SIGCOMM 2003*, 2003.

[123] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, “MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks,” in *Proc. IEEE INFOCOM*, April 2006.

[124] K. Schilling, L. Richter, M. Bernasconi, C. Jungius, and C. Garcia-Marirrodriga, “Operations and control of the mobile instrument deployment device on the surface of mars,” *Control Engineering Practice*, vol. 5, pp. 837–844, 1997.

[125] K. Schilling, J. de Lafontaine, and H. Roth, “Autonomy capabilities of European deep space probes,” *Autonomous Robots*, vol. 3, pp. 19–30, 1996.

[126] W. Wimmer, “A survey on applied on board autonomy for mission control of ESA satellites,” in *IFAC Workshop Proceedings*, 1984.

[127] P. G. Backes, K. S. Tso, J. S. Norris, and G. K. Thrarp, *Beyond Webcams - An Introduction to Online Robot*, ch. Internet-based Ground Operations for Mars Lander and Rover Missions, pp. 227–240. The MIT Press, 2002.

[128] C. Wewetzer, M. Caliskan, K. Meier, and A. Luebke, “Experimental Evaluation of UMTS and Wireless LAN for Inter-Vehicle Communication,” in *7th International Conference on ITS, Telecommunications, ITST 2007*, pp. 1–6, 2007.

[129] A. Ebner, H. Rohling, and L. Wischhof, “Performance of UTRA TDD ad-hoc and IEEE 802.11b in vehicular environments,” in *Proc. IEEE 57th Vehicular Technology Conference Spring, Jeju, 2003*, 2003.

[130] H. Holma and A. Toskala, eds., *WCDMA for UMTS: HSPA Evolution and LTE*. John Wiley & Sons, Ltd, fourth ed., September 2007.

[131] D. Constantinescu, P. Carlsson, A. Popescu, and A. A. Nilsson, “Measurement of one-way internet packet delay,” in *17th Nordic Teletraffic Seminar - NTS17*, 2004.

[132] T. Iwama, A. Kaneko, A. Machizawa, and H. Toriyama, “Real-time measurement of one-way delay in the internet environment,” in *Asia-Pacific Workshop on Time and Frequency (ATF2004)*, 2004.

[133] K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, and C. Diot, "Measurement and analysis of single-hop delay on an ip backbone network," in *IEEE Journal on Selected Areas in Communications*, p. 2003, 2003.

[134] U. Hofmann, T. Pfeiffenberger, and B. Hechenleitner, "One-way-delay measurements with CM toolset," in *IEEE International Conference on Performance, Computing, and Communications IPCCC*, pp. 41–47, 2000.

[135] V. Paxson, "End-to-end internet packet dynamics," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 4, pp. 139–152, 1997.

[136] V. Paxson, *Measurements and analysis of end-to-end Internet dynamics*. PhD thesis, University of California at Berkeley, Berkeley, CA, USA, 1998.

[137] A. Binzenhöfer, D. Schlosser, K. Tutschku, and M. Fiedler, "An autonomic approach to verify end-to-end communication quality," in *Tenth IFIP-IEEE International Symposium on Integrated Network Management (IM 2007)*, (Munich, Germany), may 2007.

[138] D. Mills, "RFC 1305: Network Time Protocol (Version 3): Specification, Implementation and Analysis," tech. rep., Network Information Center, SRI International, Menlo Park, CA, March 1992.

[139] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*. Addison Wesley Pub Co Inc, 2009.

[140] A. Bakre and B. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," in *15th International Conference on Distributed Computing Systems (ICDCS)*, pp. 136–143, 1995.

[141] H. Balakrishnan, S. Seshan, and R. H.Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *Wirel. Netw.*, vol. 1, no. 4, pp. 469–481, 1995.

[142] E. A. Brewer, Y. H. Katz, Y. Chawathe, S. D. Gribble, T. Hodes, G. Nguyen, M. Stemm, and T. Henderson, "A network architecture for heterogeneous mobile computing," *IEEE Personal Communications*, vol. 5, pp. 8–24, 1998.

[143] K. Brown and S. Singh, “M-TCP: TCP for Mobile Cellular Networks,” *ACM SIGCOMM Computer Communication Review*, vol. 27, pp. 19–43, October 1997.

[144] R. Caceres and L. Iftode, “Improving the performance of reliable transport protocols in mobilecomputing environments,” *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 850 – 857, June 1995.

[145] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, “TCP Selective Acknowledgment Options.” IETF RFC 2018, October 1996.

[146] H. Inamura, G. Montenegro, R. Ludwig, A. Gurtov, and F. Khafizov, “TCP over Second (2.5G) and Third (3G) Generation Wireless Networks.” IETF RFC 3481, February 2003.

[147] V. Jacobson, R. Braden, and D. Borman, “TCP Extensions for High Performance.” IETF RFC 1323, May 1992.

[148] M. Allman, H. Balakrishnan, and S. Floyd, “Enhancing TCP’s Loss Recovery Using Limited Transmit.” IETF RFC 3042, January 2001.

[149] K. Ramakrishnan, S. Floyd, and D. Black, “The Addition of Explicit Congestion Notification (ECN) to IP.” IETF RFC 3168, September 2001.

[150] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, “Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations.” IETF RFC 3135, June 2001.

[151] S. Dawkins, G. Montenegro, M. Kojo, and V. Magret, “End-to-end Performance Implications of Slow Links.” IETF RFC 3150, July 2001.

[152] S. Dawkins, G. Montenegro, M. . Kojo, V. Magret, and N. Vaidya, “End-to-end Performance Implications of Links with Errors.” IETF RFC 3155, August 2001.

[153] B. Gerkey, R. T. Vaughan, and A. Howard, “The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems,” in *11th International Conference on Advanced Robotics (ICAR)*, (Coimbra, Portugal), pp. 317–323, June 2003.

[154] M. Sauer, *Mixed-Reality for Enhanced Robot Teleoperation*, vol. 5 of *Schriftenreihe Würzburger Forschungsberichte in Robotik und Telematik*. Am Hubland, 97074 Würzburg: Universität Würzburg, 2010.