Using P4-INT on Tofino for measuring device performance characteristics in a network lab

Sadok Mehdi Mazigh, Marcel Beausencourt, Max Julius Bode, Thomas Scheffler
Hochschule für Technik und Wirtschaft Berlin, Germany
{Sadok.Mazigh, Marcel.Beausencourt, Max.Bode}@student.htw-berlin.de, thomas.scheffler@htw-berlin.de

***Abstract*—This paper presents a prototypical implementation of the In-band Network Telemetry (INT) specification in P4 and demonstrates a use case, where a Tofino Switch is used to measure device and network performance in a lab setting. This work is based on research activities in the area of P4 data plane programming conducted at the network lab of the Hochschule für Technik und Wirtschaft Berlin.**

I. Introduction

Fine grained performance measurements of networks and network devices usually require dedicated test equipment that is not always available and expensive to purchase and maintain [1]. In this paper we introduce an approach that uses a P4 programmable Tofino-Switch which implements a subset of the P4 In-band Network Telemetry (INT) specification [2] to measure packet processing delays in network equipment.

The P4 programming language [3] facilitates the direct manipulation of data packets in the processing pipeline of a network processor or ASIC. Current hardware implementations (e.g. Intel Tofino) support 100Gbit/s network interfaces, so that high-speed packet-manipulation and exact measurements become possible in a relatively inexpensive network setup.

The P4 Switch can be placed in the data-path of the *device under test* (DUT) and inserts hardware-generated timestamps with nanosecond precision directly into the data packets. These packets are then processed by the DUT and reappear at the Switch for a second measurement. This paper concentrates on the feasibility and methodology of such measurements based on the INT framework and not on the detailed characterization of the Intel Tofino chip.

The remainder of the paper starts with a short introduction of INT (Section II) and related work (Section III). It is then followed by a presentation of our P4 implementation (Section IV) measurements and network setup are discussed in (Section V). Finally, a conclusion is drawn and outlook on future work is given (Section VI).

II. The INT framework

In-Band Network Telemetry (INT) is one of the main applications developed by the P4 Applications Working Group[1]. INT is a framework that can monitor the network state without the involvement of the control plane. This tool allows real-time monitoring of line-rate packets (>100 Gbit/s) and thus enables more accurate insight into the data path processing.

[1]https://opennetworking.org/news-and-events/blog/announcing-the-p4-applications-working-group/

It is a challenging task to track packets at high data rates in real time, because the data is very difficult to process by conventional computers and applications (usually only individual packets are selected by sub-sampling). Generating telemetry data directly from the data plane can therefore provide a much better insight into network performance and health.

INT packets consist of an additional header (INT header) for controlling the telemetry function in network devices. This is followed by metadata (INT Metadata) such as *Switch ID*, *hop latency*, *queue wait times*, *ingress port ID* or *link usage* [4]. INT headers are generated by INT sources and inserted into a data packet. The INT packet is forwarded along the transit, with the INT nodes collecting internal status information and then adding it as INT metadata to the packet stack. This end-to-end monitoring data is later collected/removed by the INT sink and sent to a centralised controller for evaluation.

The INT framework supports different data acquisition and collection methods as well as different header placements to facilitate flexible deployment. In this work our P4 switch will only be used as an INT source and transit hop and the collected data is later extracted manually from the packet stream.

III. Related Work

In Situ Operations, Administration, and Maintenance (IOAM) [5] is an important framework introduced by the Internet Engineering Task Force (IETF). The programming of the protocols is different compared to INT, but their capabilities are very similar.

In-band Flow Analyzer is a framework defined by the IFA [6], which is implemented in a number of chipsets from the company Broadcom. The main differences between IFA and other methods are:

- The use of IFA headers as an IP option reduces the implementation effort compared to INT over TCP/UDP.
- IFA headers allow traversal through firewalls and gateways.
- The forwarding of the telemetry metadata via the transit nodes allows the monitoring of longer paths in case the MTU value is exceeded.

P4STA [7] is a load generation and measurement framework. P4STA uses hardware-generated timestamps which are stored and forwarded in TCP option fields.

P4 offers several advantages for the implementation of OAM operations, especially by defining the protocols and how headers are processed. Since it is a requirement in INT to capture the

This document is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0):
http://creativecommons.org/licenses/by-sa/4.0 This CC license does not apply to third party material (attributed to another source) in this publication.

```
header int_shim_h {
        bit<16> len;
        bit<16> port;}
header int_header_h {
        bit<16> hop_ml;
        bit<8> nhop;
        bit<8> remaining;}
header int_metadata_h {
   bit<1> bos;
   bit<7> swid;
   bit<8> ig_port;
   bit<8> eg_port;
   bit<8> reserved_1;
   bit<48> eg_mac_tstamp;
   bit<16> empty;
   bit<48> ig_mac_tstamp;
   bit<16> reserved_2;}
```

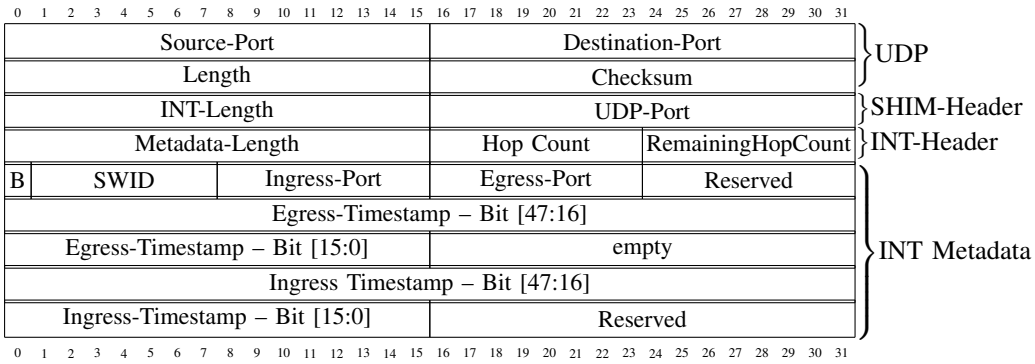

Figure 1: P4 header definition and format of our *INT over TCP/UDP* data packets

timestamp at the input and output of the switch, these should be captured as early as possible from the ingress port and as late as possible at the egress port. This allows for the measurement of delays and delay variations introduced by the network device itself, as well as measurements along the whole packet path.

In our work we show that the P4 capabilities paired with the flexibility and maturity of the INT framework can provide similar results as the other approaches but have the benefit of an open and extensible implementation.

IV. P4 Implementation

The INT specification was introduced by the P4 consortium in 2015 and since then has gone through different revisions, the latest being version 2.1 [2]. This specification defines strategies for the implementations of the framework, ranging from INT-XD (where no packet modifications are supported), over INT-MX (where packets only carry instructions for the switches) to INT-MD (which supports instructions and metadata inside of network packets). Our implementation focuses on the INT-MD version of the INT specification, which allows source and transit nodes to add additional metadata to network packets for conducting passive performance measurements on normal network traffic.

INT-MD is the mode that requires the most packet modification, while minimising the overhead at the monitoring system to compile reports from multiple nodes. P4 allows INT methods to be adapted to different formats to encapsulate different states and INT metadata to be transmitted in different payload formats.

A. INT metadata format

In our prototypical implementation we selected a subset of the *INT over TCP/UDP* specification and focused exclusively on UDP payloads. We currently do not support sending specific instructions to the INT nodes. All processing is predefined by the P4 code. The participating INT nodes simply insert Ingress and Egress timestamps as the packets enter or leave the MAC of the switch, together with the physical switch port numbers.

INT packets are identified by a specific UDP Destination Port. The switch therefore needs to modify the original port number and a shim header is added to the packet, which preserves the original port number, so that it can be later re-inserted at the INT sink. INT metadata headers are subsequently added at each hop. They are inserted between this header and the UDP/TCP payload up until a `RemainingHopCount` or the MTU limit of the link is reached. The most recent header can be found at the top of this header stack. Figure 1 shows a graphical representation of our current header format.

B. Source Code

The P4 source code for the specification of the implemented INT header structure is shown on the left side of Figure 1. It shows how easy it is to define such structures in P4 and adapt them to new purposes or to conduct experiments. There will be just one `int_shim_h` and one `int_header_h` per packet, but several `int_metadata_h` headers can be inserted along the packet path, up to the specified `remaining` hop count or until the link MTU limit has been reached. The most recent metadata header is inserted at the top and pushes down the older headers, thus creating a stack of metadata. The last metadata header is marked by the `bos` (bottom of stack) bit.

P4 supports this insertion procedure and also allows for the necessary modifications of checksum and length fields in the packet. The Ingress MAC timestamp is directly available to P4, because it is automatically provided by Tofino as part of the packet metadata information for each incoming packet. We can therefore insert this information directly into the corresponding header field by P4 code.

The procedure to include an Egress MAC timestamp in the INT header is a bit more complicated. By the time the packet reaches the Egress MAC, all P4 processing has already stopped and the P4 programmer can no longer access or execute any

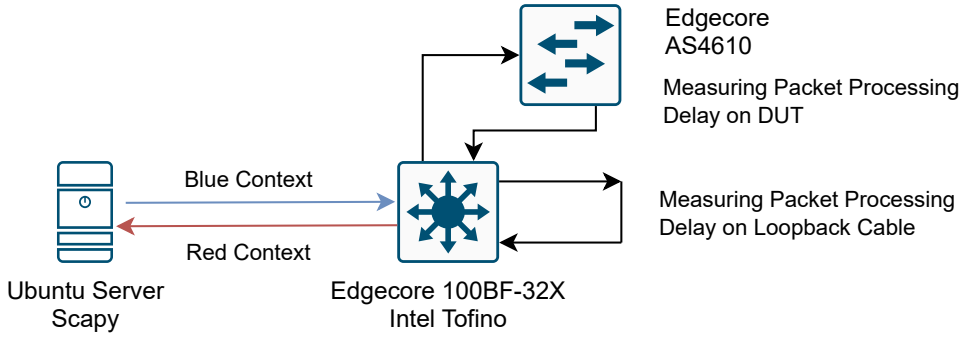

Figure 2: Network setup for INT measurements in the network lab

code on the packet directly. However, the Tofino chip supports automatic timestamp insertion and UDP checksum updates even this late in the packet pipeline. In order to make this work, the P4 programmer needs to insert a virtual PTP header into the packet, which is never sent over the wire and its only purpose is to carry the instruction for the time stamper on where to insert the timestamp into the packet. Unfortunately this stamper requires a different field format from the Ingress MAC, so that we have to leave 16 bit of empty space between both timestamps in our packet.

V. First Measurements

Based on our implementation we conducted a few measurements to establish a baseline and explore the capabilities of P4 INT. Figure 2 shows the topology of our network lab. A Ubuntu server is connected to a P4 switch, so that test packets can be sent to the switch. The switch then forwards the test packets to the DUT and inserts the INT metadata.

Our Ubuntu server uses two different network namespaces (*blue* and *red*) to distinguish outgoing and incoming test traffic. Test packets were created using Scapy[2] and there was no additional load on the network devices. Figure 3 shows the insertion of INT headers by the P4 switch, as packets enter and re-enter the switch. A normal UDP packet enters the switch from the *blue* network namespace. The P4 switch acts as an INT source and inserts the necessary INT headers together with a first INT metadata header containing the timestamps. When the P4 switch sees the packet a second time (after it has looped back from the DUT), it adds another INT metadata header containing an updated timestamp.

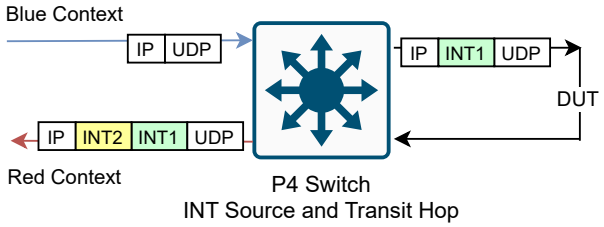

Figure 3: INT Packetflow and marking by P4-capable switch

We collected both Ingress and Egress MAC timestamps for each processed packet. This allows us to get some data on in-device processing, as well as measure delay occurring in the network. The switch currently does not act as an INT sink. The packet containing the full INT stack is forwarded via the *red* network namespace to the Ubuntu server for capture and processing. All our tests were run in a steady state, where all tables were fully established and no MAC learning had to take place.

The first test was a simple loopback test. Packets from the egress port of the P4 switch were directly looped back via an optical cable to an ingress port on the same switch. For this, the 100 Gbit/s port was configured as 4x25 Gbit/s and an optical breakout cable was used to create the loop.

In a second test, the port was configured as 40 Gbit/s and a 4x10 Gbit/s breakout cable was used to connect to an Edgecore AS4610 switch running BISDN Linux[3]. Packets were switched on the AS4610 in a dedicated VLAN and reappeared immediately on the P4 switch. Table I shows the aggregated results from these measurements. We also plotted the delay distribution for 30 test packets in both test cases in Figure 4.

The measurement via the Loopback link showed a round-trip delay (Egress MAC to Ingress MAC) of around 107ns. This measurement is in line with those reported by other researchers [7]. When we included the AS4610 in the packet path, the roundtrip delay increased to 2.1µs. Knowing this baseline allows us to accurately measure the processing delay of external network components with very high precision. We measured processing delay of the packet pipeline in our switch (adding the INT Headers into the packet and switching to a fixed port) with a mean of 615.72 ns and a standard deviation of 16.3 ns. All test packets were generated and processed using Scapy. The INT headers were added by the Tofino P4 switch.

Table I: Measured packet forwarding delays

	Mean delay in ns	Std. Dev. in ns
Loopback	106.96	1.06
AS4610	2128.96	16.58

[2]https://scapy.net/

[3]https://www.bisdn-linux.de/

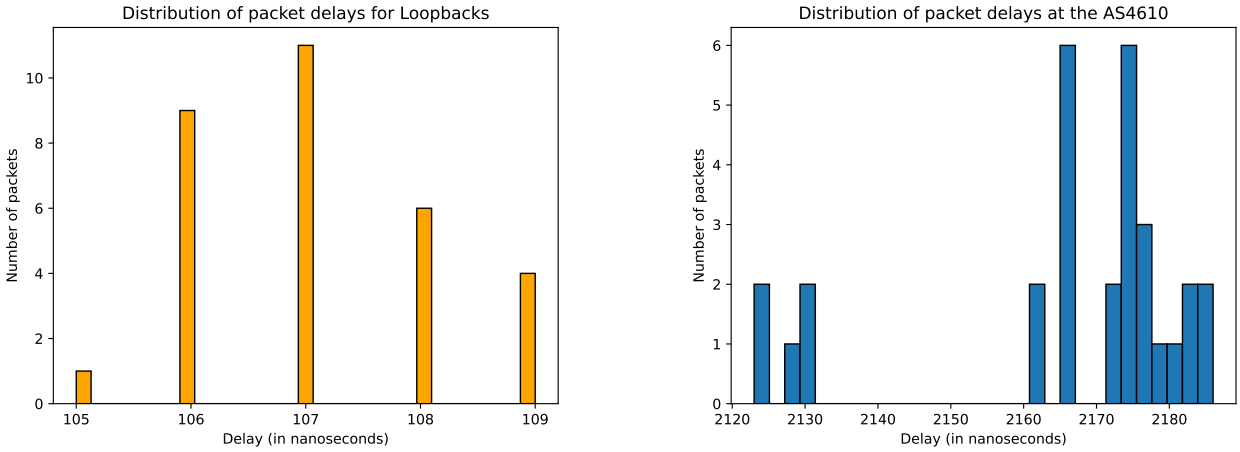

Figure 4: Distribution of measured delay values for 30 test data packets

For debugging purposes Wirehark[4] was used. Wireshark is a powerful tool for network analysis offering a broad range of dissectors for protocols out of the box. Our implementation required custom dissectors, which were implemented using LUA.

VI. Conclusion and Outlook

Our work showed that an INT P4 implementation is possible and can be used to effectively measure delay parameters in networks and network devices with high precision.

Additional measurements and test are needed to get a better estimate of the precision and reproducibility of our results. Packet delays in *store-and-forward* switches are highly dependent on packet size and interface speed. Our UDP INT packet has a size of 106 Byte after the first measurement. This results in a serialisation delay on a 10 Gbit/s interface of 84.8 ns and 33.92 ns on a 25 Gbit/s interface. We want to find out, if these theoretical values can also be validated using practical measurements. We also have not yet made any measurements with the 100 Gbit/s line rate supported by the switch.

For our tests we had no incentive to implement an INT packet sink. The packets carry the INT header all the way to the destination. This allows us to collect and process the headers at the destination host in our lab. For passive measurements in a real network this solution would break all network protocols. So we need to develop an INT sink, together with functional telemetry analysis, to deploy this solution outside of our lab.

Acknowledgment

This work was partly supported by the Institut für Angewandte Forschung (IFAF) Berlin under the project NetTraffic-P4[5].

[4]https://www.wireshark.org/

[5]https://www.ifaf-berlin.de/projekte/nettraffic-p4/

References

[1] G. Antichi, M. Shahbaz, Y. Geng, N. Zilberman, A. Covington, M. Bruyere, N. Mckeown, N. Feamster, B. Felderman, M. Blott, A. W. Moore, and P. Owezarski, "OSNT: Open Source Network Tester," *IEEE Network*, vol. 28, no. 5, pp. 6–12, 2014.

[2] P4.org, "In-Band Network Telemetry (INT) Dataplane Specification, Version 2.1," 05 2020. [Online]. Available: https://github.com/p4lang/p4-applications/blob/master/docs/INT_v2_1.pdf

[3] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, jul 2014. [Online]. Available: https://doi.org/10.1145/2656877.2656890

[4] P. Manzanares-Lopez, J. Muñoz-Gea, and J. Malgosa, "Passive in-band network telemetry systems: The potential of programmable data plane on network-wide telemetry," *IEEE Access*, vol. PP, p. 4, 01 2021.

[5] F. Brockners, S. Bhandari, D. Bernier, and T. Mizrahi, "In Situ Operations, Administration, and Maintenance (IOAM) Deployment," RFC 9378, Apr. 2023. [Online]. Available: https://www.rfc-editor.org/info/rfc9378

[6] J. Kumar, S. Anubolu, J. Lemon, R. Manur, H. Holbrook, A. Ghanwani, D. Cai, H. Ou, Y. Li, and X. Wang, "Inband Flow Analyzer," Internet Engineering Task Force, Internet-Draft draft-kumar-ippm-ifa-05, Aug. 2022, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-kumar-ippm-ifa/05/

[7] R. Kundel, F. Siegmund, J. Blendin, A. Rizk, and B. Koldehofe, "P4STA: High Performance Packet Timestamping with Programmable Packet Processors," in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, 2020, pp. 1–9.