25-02-2021

In-Band Network Telemetry Tests in NREN Networks

Grant Agreement No.:	856726
Work Package	WP6
Task Item:	Task 1
Document Type:	White Paper
Dissemination Level:	PU (Public)
Lead Partner:	GARR
Document ID:	GN4-3-21-2866285
Authors:	Damian Parniewicz (PSNC), Tomáš Martínek (CESNET), Federico Pederzolli (FBK), Damu Ding (FBK), Mauro Campanella (GARR), Ivana Golub (PSNC), Tim Chown (Jisc)

The research leading to these results has received funding from the European Union’s Horizon 2020 research and innovation programme under Grant Agreement No. 856726 (GN4-3).

Abstract
This document reports the conclusions drawn and lessons learned from the study and implementation of In-Band Network Telemetry (INT) carried out by the DPP sub-task of WP6 T1.

Table of Contents

Table of Figures

Table of Tables

Executive Summary

The Data Plane Programming (DPP) sub-task in the Network Technology Evolution task (Task 1) of the Network Technologies and Services Development work package (WP6) of the GN4-3 project is assessing the applicability and usefulness of DPP technologies in two use cases of interest to the GÉANT NREN community: DDoS detection and mitigation; and high-precision network monitoring using In-Band Network Telemetry (INT).

The intermediate results of the data plane programming work on DDoS detection and mitigation were presented at the Performance Management Workshop in March 2020 [DPP_PMW].

This document focuses on the experiences gathered through running INT in a geographically distributed testbed spanning three sites and deployed on top of NREN production networks. Testing is performed by adding an INT header to each packet in medium-capacity UDP flows between the sites. The added INT header contains packet-specific flow and time quantities that are analysed at the destination. The document details the design choices taken, the operational challenges faced and the initial results, which demonstrate the potential for high-precision monitoring through the use of INT.

The activity utilises programmable hardware (Tofino-based switches and FPGA-based SmartNICs), and required the setting up of a simple, high-performance data collection and presentation system based on the open source tools InfluxDB and Grafana.

The results presented here are considered the first step in preparation for future high-capacity network monitoring and debugging systems. The system set up can be extended by a software-based INT measurement tool and Data Plane Development Kit (DPDK), which will allow the use of off-the-shelf computing equipment without specialised hardware.

1 In-Band Network Telemetry (INT) Overview

In-Band Network Telemetry (INT) is a network monitoring framework based on a specification of the P4 language consortium [INT v2.1] [P4 Specs] that can provide highly detailed information on network behaviour through inserting a small amount of information directly inside packets passing through network devices where INT functionality is enabled, essentially adding probing functionality to potentially every packet, including customer traffic. This makes INT a very powerful debugging protocol, capable of measuring and recording the `experience' of each tagged packet sent in the network.

The basic workflow of INT is depicted in Figure 1.1.

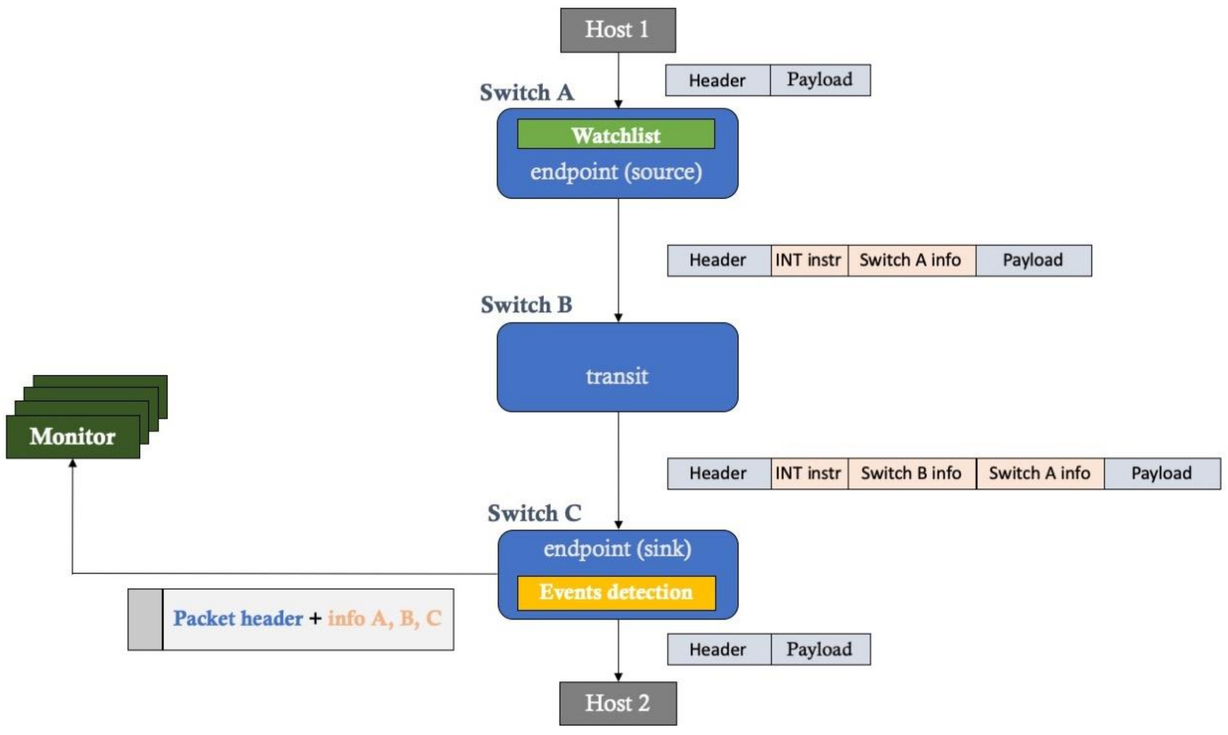


Figure 1.1: INT network switch roles and information flow [P4docs]

The first INT-enabled switch adding information to the packets of a monitored flow is called the INT source, while the destination INT switch is called the INT sink. The INT sink removes all INT headers and information (called INT metadata) from packets and passes INT information to the INT monitoring system. Intermediate INT nodes – called INT transit nodes – may add, read, modify and delete specific INT headers.

Each INT-enabled switch can add the following list of standard network-specific information, and in principle any other type of information, to a packet:

- *Node ID*: ID of the INT-enabled network node adding the new metadata information to the packet.
- *Ingress Interface ID*: local ID of the port where the packet was received.
- *Egress Interface ID*: local ID of the port the packet will be sent out of.
- *Ingress Timestamp*: local node time when the packet was received.
- *Egress Timestamp*: local node time when the packet left the node.
- *Hop Latency*: time taken for the packet to traverse that specific node.
- *Egress Port TX Link Utilization*: current utilisation of the egress port via which packet will be sent out.
- *Queue Occupancy*: ID of the queue in which the packet was temporarily stored, and measurement of the occupancy of that queue.
- *Buffer Occupancy*: ID of the buffer in which the packet was temporarily stored and measurement of that buffer's occupancy.

Which information is actually added to each packet depends on the INT monitoring configuration. Upon the network control plane request to monitor a flow, the INT source node sets the *instruction map* field of the main INT header. The instruction map is a set of bits coding to which metadata information must be added by INT nodes.

The current INT specifications do not mandate a specific header placement in the packet structure and packet tagging; however, they detail possible choices compatible with the most common packet formats, e.g.: INT over IPv4/GRE, INT over UDP/TCP, etc. Note that, since INT adds additional bytes to each packet, care has to be taken to avoid exceeding the MTU along the path and to ensure that packet processing by 'standard', i.e. non-INT-enabled nodes is not impacted. As a norm, NRENs offer complete transparency and neutrality in their backbones, as well as an MTU of about 9K bytes. Since edge MTU is usually only 1500 bytes, most constraints come from source or destination sites, either because of smaller MTUs or because of packet filtering, in particular firewalling.

In-band Network Telemetry enables detailed network debugging and performance analysis. It allows a network operator to:

- Observe flow dynamics at high-resolution:
  - changes in flow characteristics
  - microbursts
  - high-resolution measurement of packet transmission delay and packet delay variation
  - higher than expected latency for end-to-end flows.
- Identify flows causing congestion:
  - high-resolution node queue/buffer dynamics
  - non-zero or increasing packet drops.
- Detect events about flow path change:
  - observing new port IDs, new node IDs in the flow.

Packets with INT data can seamlessly traverse non-INT nodes because, in this implementation, INT headers are placed after Layer 4 headers. When the network is composed of both INT-enabled and non-INT-enabled switches only partial mesh information can be collected. In the simplest case, INT nodes can be added only on the borders of the network, in which case they can still provide high-resolution insights into end-to-end or domain-to-domain flow dynamics.

At this time, INT capabilities are under development for only a small set of network devices (mostly FPGA and Tofino-based), such as the NoviFlow switch [NoviFlow], for which the INT implementation is based on Intel/Barefoot's reference INT implementation for Tofino chips [Intel Tofino]. Broadcom [Broadcom] are also implementing INT for their programmable chipset families, which will be offered as part of the Broadcom+ Network Visibility product.

In-band Network Telemetry is currently being tested and used in the following Research & Education networks:

- ESnet in the US:
  - deployed FPGA-based INT nodes connected to standard ESnet switches in several locations in the US;
  - INT headers are added only to duplicates of the original packets reduced in size by payload removal, which are forwarded along the same path as original packets;
  - INT is used to debug and perform research on the network behaviour and its traffic.
- AmLight in Brazil:
  - deployed NoviFlow switches with INT capabilities;
  - INT used to monitor traffic for astronomy users.
- SURF in the Netherlands:
  - deployed using Tofino-based switches and FPGA cards;
  - to increase performance of INT collector, Remote Direct Memory Access (RDMA) technology was utilised;
  - used for research activities around IPv6 path tracking, IPv6 switch latency measurement.

For more information on the ESnet and AMLight tests, see the presentations in The First Telemetry and Big Data Workshop organised by the GN4-3 project [TBDW].

2 Current Experimentation on INT Implementation in GN4-3

2.1 Distributed INT Testbed

The DPP team in GN4-3 WP6 created a distributed testbed for testing an in-house P4 implementation of In-Band Network Telemetry. The tests carried out are designed to measure the performance characteristics of the production network and analyse the infrastructure requirements to best gather and visualise INT data. The testbed does not (and is not intended to) mimic exactly how INT technology could be deployed in GÉANT's and the NRENs' production networks but is instead targeted at validating INT functionalities and capabilities for advanced monitoring. Practical deployment details will depend on the specific use case and the granularity of information requested, which may involve for example a specific flow being selected or generated and monitored end-to-end. The INT information may be generated at the two ends only, or information can be added at each intermediate hop, provided it is INT enabled.

The testbed, depicted in Figure 2.1, is composed of three sites located respectively in the NRENs in the Czech Republic (CESNET) and Poland (PSNC) and in the FBK research institute in Italy. The nodes have been configured to be visible using public IP addresses, as initial tests showed that encryption or tunneling seriously reduced their performance. All sites are interconnected using the production connectivity of the three networks, so the INT packets cross standard switches/routers.

INT source and transit nodes are located both in FBK and PSNC. FBK INT nodes are EdgeCore Wedge switches, while PSNC is currently using a virtual INT environment based on bmv2 software switches [P4bm] deployed in Mininet and adapted for bmv2 [P4app]. The PSNC site is used primarily for the functional verification of the INT implementation in the P4 language and collects and analyses INT metadata. FBK generates a constant stream of UDP packets using the [iperf] tool running on 'Server1,' connected to EdgeCore Wedge#3 which implements INT source functionality. Wedge#2 is slated to work as an INT transit node in the future, but it currently behaves as a regular, non-INT switch, while `Server2' acts as a router and provides NATted connectivity to the Internet. In PSNC, host H1 generates UDP packets using a python script employing the [Scapy] library and sends them as a separate stream to CESNET via the bmv#1 node, as shown in Figure 2.1.

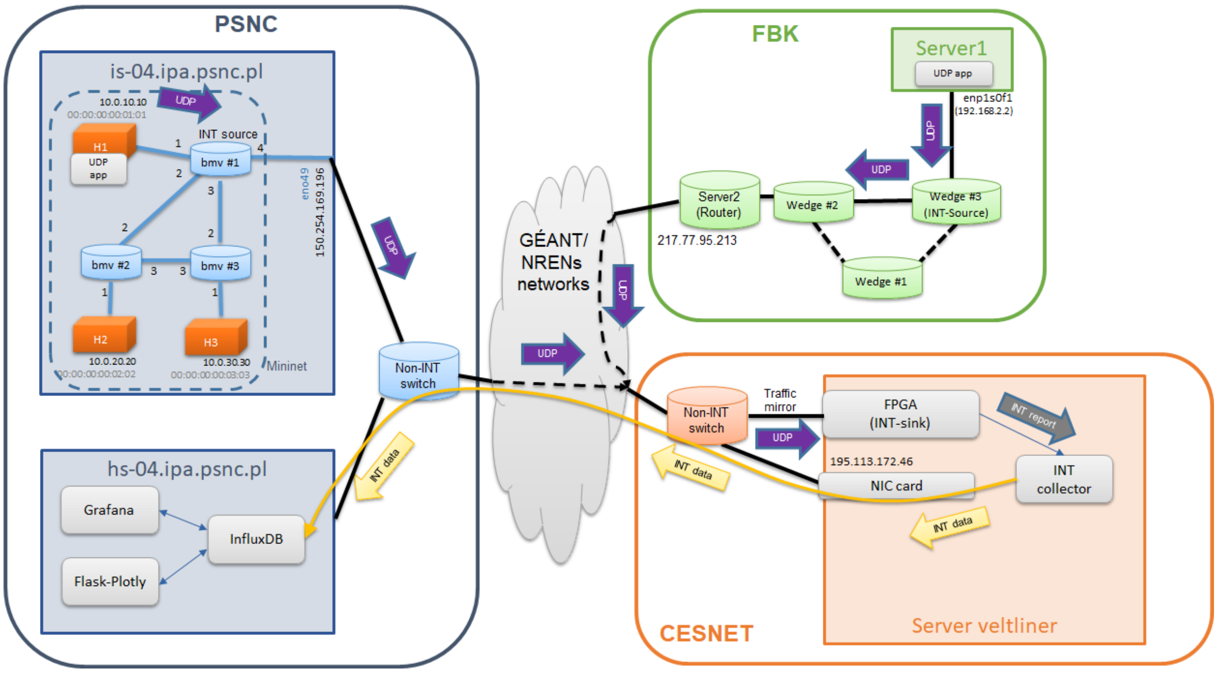


Figure 2.1: Distributed GN4-3 testbed for development and testing of In-Band Telemetry

The INT sink node in CESNET is responsible for extracting INT telemetry data and sending it to the Influx database deployed in PSNC. In PSNC, the INT data visualisation tools are based on [Grafana] and used for period presentation of hours-to-microseconds graph resolution and Plotly-based visualisation [Plotly] for packet-level nanosecond graph visualisation.

At the time of writing, flows of up to about 260k UDP packets per second were tested and successfully presented using the INT collection and visualisation system developed. More details are presented in the sections below on 'Large Data Collection and Presentation' and 'INT Testbed: Issues Encountered and Lessons Learned'. Experiments using TCP-based flows are planned for the near future.

2.2 INT Headers Format and Content

The DPP team uses the specification of the INT header and INT metadata presented in Figure 2.2 and Figure 2.3. In the experiments performed, the INT protocol carries information (in the form of INT metadata) about the identifier of the switch, identifiers of ingress and egress ports where each packet was received and then forwarded, and ingress and egress 64-bit timestamps (compiled using local clocks) for when each packet was received and transmitted, respectively. Note that, due to hardware limitations, FBK's wedges can only utilise a 48-bit timestamp, which cannot easily be synchronised with a global clock (it can therefore be used for jitter but not very precise delay calculations); this issue is discussed again in section 4 which lists lessons learned.

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Int Type   |      Rsvd     |      Len      |    Dscp   |Rs.|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver|Rep|C|E|   Rsvd  | Ins Cnt |    Max Hops   |   Total Hops  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Instruction Mask       |              Rsvd             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                      INT node #1 metadata                     +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                      INT node #2 metadata                     +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                              ...                              +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                      INT node #N metadata                     +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Next Proto  |           Dest Port           |Pad|    Dscp   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2.2: Structure of INT header and INT metadata

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Switch Id                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Ingress Port Id        |         Egress Port Id        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                       Ingress Timestamp                       +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                        Egress Timestamp                       +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2.3: Structure of INT node metadata header

2.3 INT Header Location in the Packet

For the tests described in this document, an arbitrary choice was made to place INT tags and headers in the payload region of standard IP packets, after the IP and UDP or TCP headers, and to tag the packet with a DSCP value equal to 0x20 (the DSCP value used for INT must be decided and configured by the network operator) that signals its nature (i.e., to identify the packet as containing INT metadata to INT-enabled nodes further along its path). A filter is configured in INT nodes to ensure that only packets with a known combination of tags are processed in the INT pipeline. This solution, together with a specific DSCP value to distinguish INT packets from regular traffic, minimises the chance of issues with middleboxes analysing headers above the IP headers allowing standard routing/switching and firewalling to work without modifications.

To cater to various INT deployment requirements, alternative approaches in the placement of INT headers have been defined in the INT specification v0.5 [P4 Specs] and are listed in Table 1.

Location	Description
INT over IPv4/GRE	INT headers are carried between the GRE header and the encapsulated GRE payload. The existence of INT is indicated by the GRE protocol type value reserved for INT.
INT over TCP/UDP (chosen approach by DPP working group)	A shim header is inserted following the TCP/UDP header. INT Headers are carried between this shim header and the TCP/UDP payload. Since v2.0, the spec also supports an option to insert a new UDP header (followed by INT headers) before the existing L4 header. This approach doesn’t rely on any tunneling/virtualisation mechanism and is versatile to apply INT to both native and virtualised traffic. The existence of an INT header is indicated by the presence of a specific DSCP value and/or presence of a specific destination port value. Additionally, 8 bytes of probe marker can be added just after the TCP/UDP header, especially when no DSCP value or destination port can be reserved for INT.
INT over VXLAN	VXLAN generic protocol extensions are used to carry INT headers between the VXLAN header and the encapsulated VXLAN payload. The existence of INT is indicated by the reserved value of the Next Protocol field of the VXLAN header (field defined by VXLAN extension [VXLAN-GPE]).
INT over Geneve	Geneve is an extensible tunneling framework, allowing Geneve options to be defined for INT headers. The existence of INT is indicated by the GENEVE option class identified by code 0x0103.

Table 2.1: Possible INT header location options

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 +                                                               +
 |                                                               |
 +                                                               +
 |                      IPv4 header (20 bytes)                   |
 +                                                               +
 |                                                               |
 +                                                               +
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 +                       UDP header (8 bytes)                    +
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 +                                                               +
 |                      INT header (12 bytes)                    |
 +                                                               +
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 +                                                               +
 |                                                               |
 +                                                               +
 |                INT nodes metadata (N x 24 bytes)              |
 +                                                               +
 |                                                               |
 +                                                               +
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                    INT tail header (4 bytes)                  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 |                          Packet payload                       |
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2.4: Main INT header placement in a packet.

The entire INT header, whose placement is reported in Figure 2.4, is composed of 12 bytes of initial INT header and 4 bytes of tail INT header. Additional INT node metadata headers are added between these two, one after another, by each intermediate node supporting INT.

In the case of the experiments currently performed by the DPP team, a single INT node metadata is 24 bytes long and contains switch id, ingress and egress port ids, and timestamps. The calculations of additional bytes added to the packet by the INT node are presented in Table 2.2. However, depending on the INT measurement configuration, this size calculation can change. The exact size of all INT headers in a packet depends on the bits encoded in the *instruction map* field of the main INT header.

Number of INT nodes	Additional bytes in a packet
1	12 + 24 + 4 = 40 bytes
2	12 + 2 x 24 + 4 = 64 bytes
3	12 + 3 x 24 + 4 = 88 bytes
4	12 + 4 x 24 + 4 = 112 bytes

Table 2.2: INT header and metadata sizes vs. number of INT nodes on the path (DPP settings)

The presence of INT data after TCP/UDP headers is indicated by the combination of source and destination IP addresses, DSCP value, and destination port values. The original DSCP and port values are copied to corresponding INT tail header fields for recovery in the INT sink node.

2.4 Packet Processing on the INT Sink Node in CESNET

As shown in Figure 2.5, within CESNET's premises all incoming packets sent to the public IP address of the INT sink node are mirrored through a network test access point (tap) (to enable testing of different interface speeds and a simple mirroring function) to the physical port of an FPGA card located in the same server.

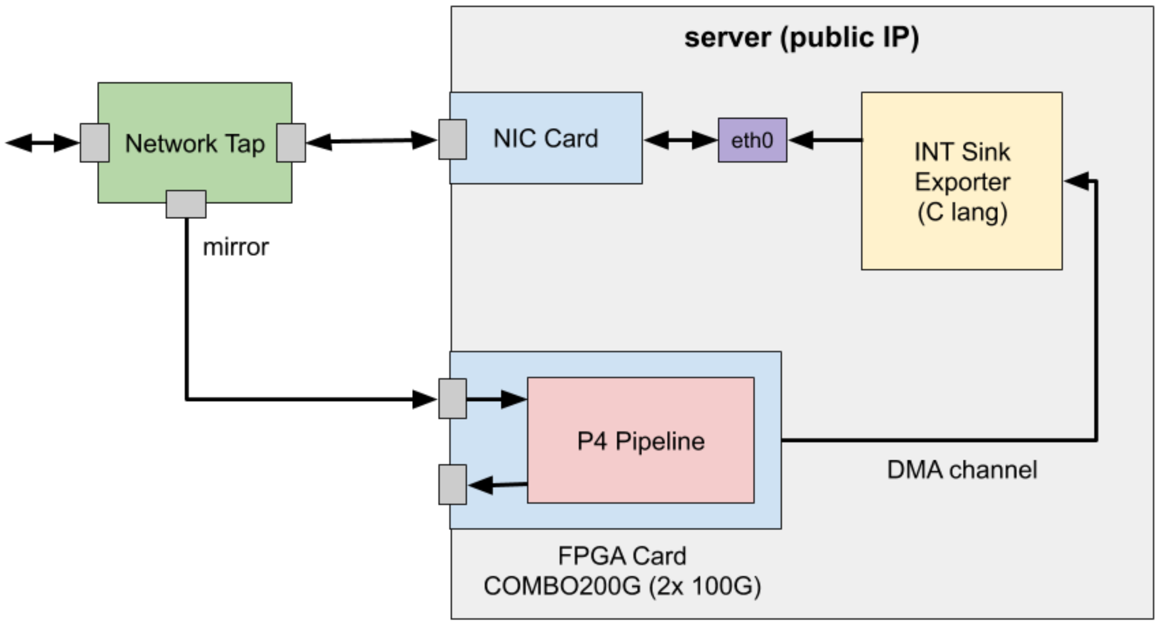


Figure 2.5: Current architecture of CESNET's INT sink node

The FPGA implements a P4 pipeline, which identifies packets with an INT header in incoming traffic and extracts from them the information required for export to the database (see section `Exporting

INT Data to the INT Database'). Since the communication with the database is currently realised through VPN and TCP connections, the extracted information cannot be sent directly from the FPGA, because such a complex function cannot easily be described through the P4 language and implemented in the available FPGA resources. Instead, it is sent through the DMA channel to the software part of the application, which converts the incoming data to the InfluxDB line protocol format [InfluxDB_lp] and sends it via a secure line to the influx database (see Figure 2.5).

The FPGA technology employed allows processing of incoming traffic at wire speeds of up to 100Gbps It also assigns a hardware timestamp to each incoming packet to accurately measure characteristics such as delay or jitter (this is a significant issue with some NICs and switches: some only support timestamps on one of Tx/Rx, others do not allow to easily synchronise the local hardware clock with a remote, global clock). The processing of the incoming packet itself is controlled through a program written in the P4 language, which is automatically converted into the hardware description language (VHDL) and the configuration bitstream of the FPGA chip.

The limitations of the FPGA solution are its higher purchase price and more limited programming capabilities (e.g., a limited number of items in M/A tables or registers). Therefore, the possibility is being explored of automatically compiling the P4 INT sink program into a DPDK application running on off-the-shelf computing equipment equipped with commodity network interfaces. This solution leads to lower CAPEX and greater flexibility in supporting P4 programs, but also to a lower performance ceiling, which is estimated to be in the order of 10 to 50 Gbps.

It is also planned to upgrade the INT sink node architecture. Currently, the node is connected as a passive monitoring probe through a network TAP, without the ability to affect traffic to the server, whereas in the future the plan is to design the INT sink node as an active element with the ability to extract INT records directly from flows going to the endpoint server.

3 Large Data Collection and Presentation

3.1 Exporting INT Data to the INT Database

The INT exporter is a software component that reads INT reports from the INT sink node and exports the desired information to the INT database using the database protocol (e.g., the Influx line protocol in the case of InfluxDB, an example of which is given in Figure 3.1). INT data is buffered and sent to the INT database in batches containing 40k INT reports. Each batch of INT data is sent using a TCP connection initialised by the InfluxDB client library.

```
int_telemetry,srcip=217.77.95.213,dstip=195.113.172.46,srcp=54474,dstp=1700
0
origts=78325098270654,dstts=1605187313616122575,seq=1031173654,delay=160510
8988517851921,sink_jitter=128891,reordering=0 1605187313616122575
```

Figure 3.1: A single INT data entity (217 bytes) exported in Influxdb line protocol format

Most attributes exported to the database are just copies of information carried by INT packets. However, for performance reasons, some precomputation is done within the INT collector, resulting in three additional attributes (the last three attributes presented in Table 3.1 below).

Attribute	Description
srcip	Source IP copied from the IPv4 header of the INT packet.
dstip	Destination IP copied from the IPv4 header of the INT packet.
sctp	Source port copied from the UDP header of the INT packet.
dstp	Destination port copied from the UDP header of the INT packet.
seq	For UDP flow, it is the value of the INT sink counter incremented for each incoming packet (in future will be used for coping the TCP sequence number).
srcts	64-bit timestamp copied from the first INT node metadata (INT source node ingress timestamp).

Attribute	Description
dstts	64-bit timestamp copied from the last INT node metadata (INT sink node egress timestamp).
delay	The difference between ‘dstts’ and ‘srcts’. If INT nodes are well synchronised with a global clock, then this value represents packet transmission delay. In case of weak synchronisation or lack of synchronisation, it can be used for the calculation of packet delay variation.
sink_jitter	Inter-packet delay at INT sink node (difference of ‘dstts’ between consecutive packages in the monitored flow).
reordering	Indicator if packet reorder happened (difference of seq’ between consecutive packages in the monitored flow). It will be meaningful when measuring TCP flows. Currently set always to zero.

Table 3.1: INT data attributes exported from the INT collector to the INT database

3.2 INT Information Collection in the InfluxDB Database

The INT database is used to collect all INT data exported by the INT collector located in CESNET. It permits verification of the correctness of the received INT data. Note that storing each INT report is only possible for relatively low INT packet rates. Supporting high rates requires significant investment in infrastructure.

In the distributed testbed described in this document, the INT database is deployed in PSNC using an InfluxDB docker container, which was chosen due to its high performance on both writes and queries of time series data, nanosecond timestamp precision, effective time-series data compression and very rich functionality for time series manipulation.

InfluxDB is deployed on an HP ProLiant DL 380 Gen9 server equipped with 160GB RAM and 1.2TB SSD disk storage, which allows storage of up to a few days’ worth of INT data (the data retention time is managed by the Influx retention policy set according to the amount of INT data ingested in InfluxDB). InfluxDB is configured to receive INT data on TCP port 8086.

INT data attributes that identify a flow (4-tuple flow: src IP, dst IP, src port, dst port) are saved as Influx tag fields, and the rest of the attributes are Influx field values. Additionally, each INT data entry contains a timestamp value, which is a copy of the ‘dstts’ value (required for the proper sequencing of data points within the INT visualisation). The initial size of each INT data element is presented in Table 3.2.

Entry element name	Entry element type	Element size in a single entry
srcip	Influx tag	Not part of entry size
dstip	Influx tag	Not part of entry size
sctp	Influx tag	Not part of entry size
dstp	Influx tag	Not part of entry size
seq	Influx value	uint type - 8 bytes
srcts	Influx value	uint type - 8 bytes
dstts	Influx value	uint type - 8 bytes
delay	Influx value	uint type - 8 bytes
sink_jitter	Influx value	uint type - 8 bytes
reordering	Influx value	uint type - 8 bytes
timestamp	Influx timestamp	uint type - 8 bytes

Table 3.2: INT data entry elements in InfluxDB database

The above table suggests that the size of a single entry in the INT database is 7*8=56 bytes plus additional bytes for a pointer to an influx measurement entry (defined by the field tag values). However, thanks to the great compression capabilities of InfluxDB, the stored size of each entry is much smaller. For example, 24 hours of INT measurement of a UDP flow between FBK and CESNET with a rate of 5400 packets per second results in the INT database disk usage increasing by only 12GB (see Figure 3.2). During this time, the database collected 466.5 million entries within a single influx measurement. In this specific case, a single INT data entry size requires 0.027 bytes of additional disk space (12GB / 466.5 millions). This high compression rate of the large amount of data in InfluxDB is possible because some INT data values either change slowly or are constant (e.g., switch and traversed port IDs). More information about InfluxDB data storing and compression is available in the medium.com [DataSeries].

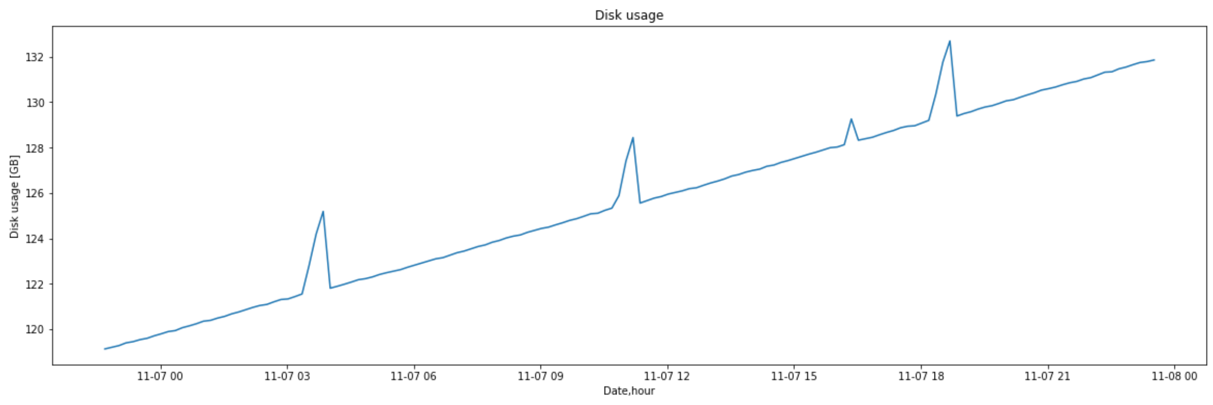


Figure 3.2: INT database disk usage for 24 hours when collecting 5400 INT data entries per second

A series of standalone performance tests of the INT database in PSNC were performed via python scripts deployed on another local server. These measured the performance of a single InfluxDB instance without taking into account the performance of other elements of the INT infrastructure. Figure 3.3 and Figure 3.4 below present the result of these tests:

- The INT database can, using a single TCP connection, collect all INT reports without losses up to about 100K INT entries per second.
- Better performance can be achieved using multiple parallel TCP connections, but maximum write performance – about 1.2M reports per second (85K per TCP client) – without losses was achieved using just 14 parallel TCP connections.
- Maximum query performance is about 100-120K INT data entries per second.

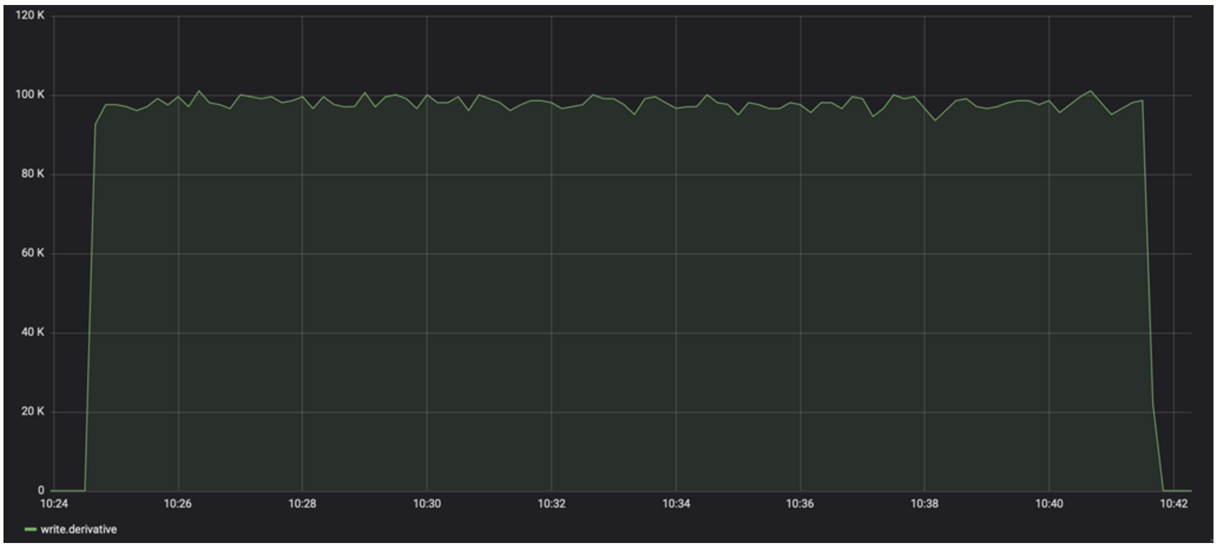


Figure 3.3: INT database a single client write performance performed within PSNC testbed (100 million INT data entries)

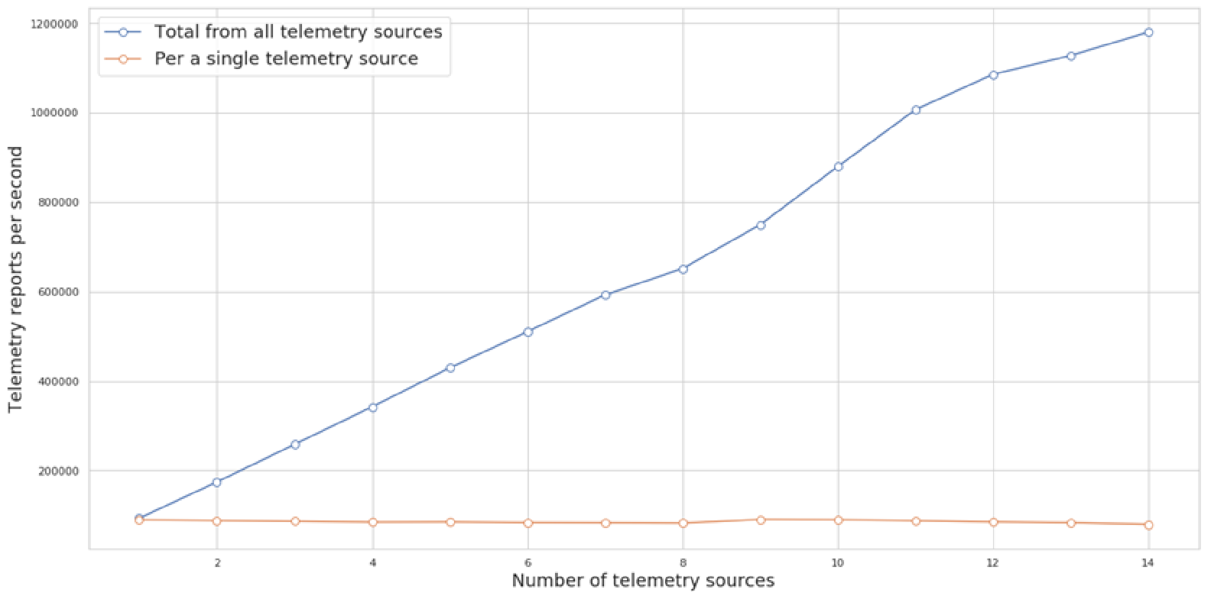


Figure 3.4: Total measured write performance of the INT database within PSNC testbed

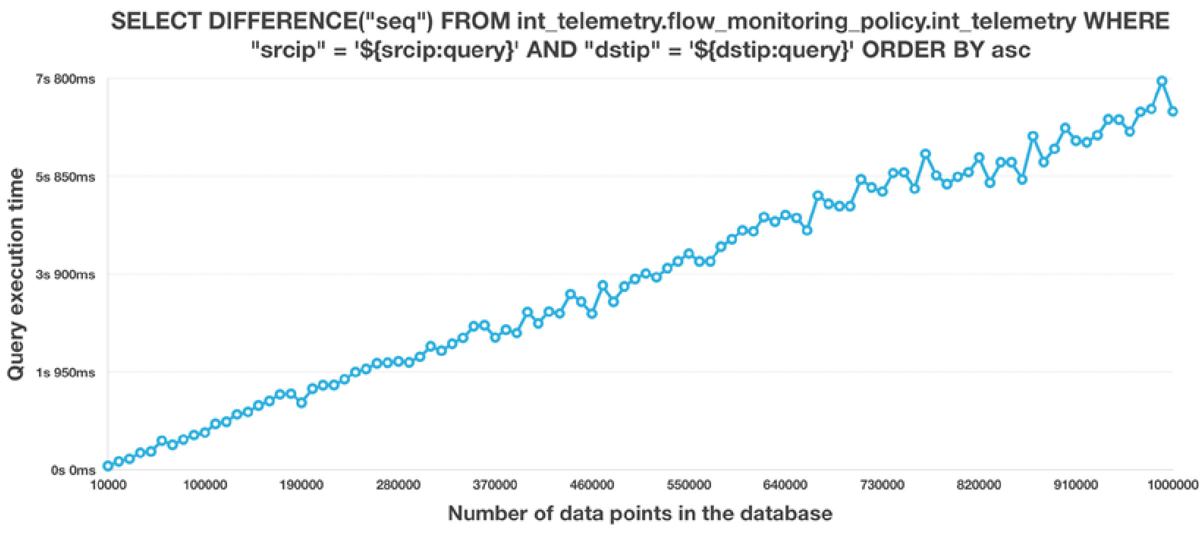


Figure 3.5: INT database read execution time for a selected query and queried data size

During INT tests, when measuring the performance of exporting INT reports from the INT sink located in CESNET to the database deployed in PSNC, numerous small slowdowns were experienced due to packet loss between the two sites. The maximum measured performance, shown in Figure 3.6, was 312K INT entries per second (about 530 Mbps of INT data), exported using 20 parallel TCP connections.

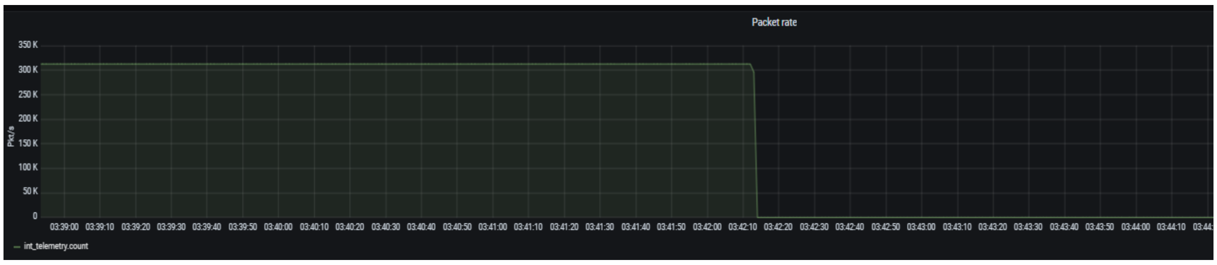


Figure 3.6: Maximum measured performance of an INT sink exporting a single flow's data to InfluxDB (312K packets per second)

3.3 INT Data Visualisation

Depending on the desired time resolution, the collected INT data can be visualised using two different tools:

- [Grafana] – an existing open-source visualisation tool useful for presenting INT data covering long time scales.
- [Plotly] – an in-house product for presenting INT data at the packet-level resolution.

Grafana presents time-averaged data from the INT database in real-time. This visualisation is designed to monitor INT data for periods ranging from seconds up to several hours. The Grafana dashboard for INT telemetry displays the 'INT sink and INT source timestamps difference', 'packet inter-arrival time' and 'reordering' attributes stored in the INT database and also the number of packets in the monitored flow. If the clocks of INT-enabled nodes are properly synchronised, the 'INT sink and INT source timestamps difference' will be the real path delay experienced by the packets. The dashboard, shown in Figures 3.7-3.9, works well for visualising intervals between about 10 seconds and a few hours.

In the following plots the scale for absolute delay in some sub-graphs is for illustration purposes of the behaviour and does not correspond to the effective end-to-end time delay. This is due to the synchronisation issues encountered and detailed in section 4.

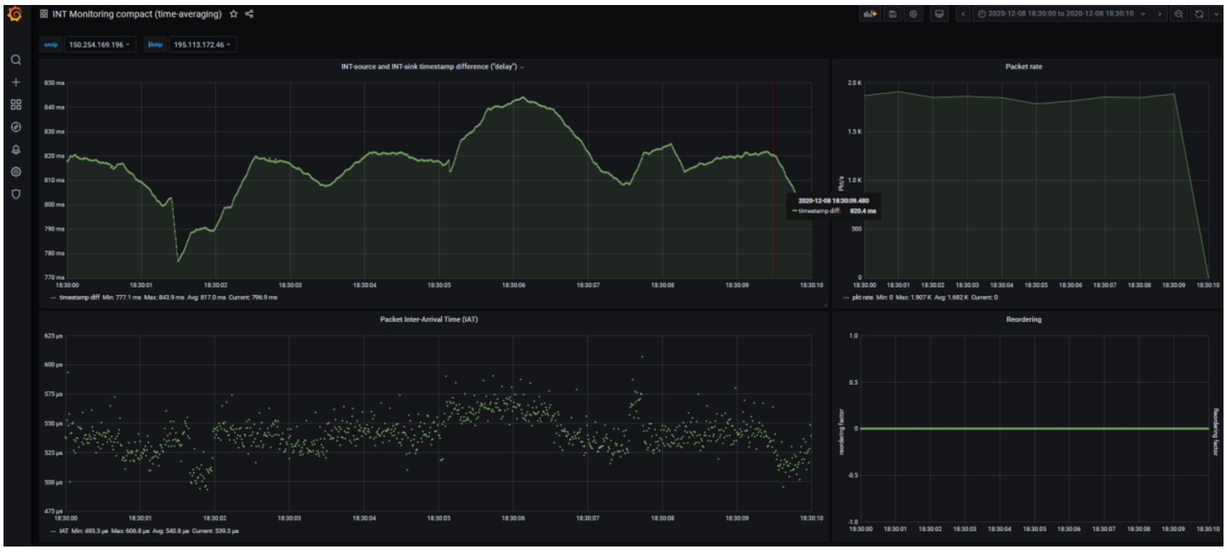

INT Monitoring compact (time-averaging)
2020-12-08 18:30:00 to 2020-12-08 18:30:10
srcip 150.254.169.196
dstip 195.113.172.46
INT-source and INT-sink timestamp difference ("delay")
2020-12-08 18:30:09.480
timestamp diff: 820.4 ms
timestamp diff Min: 777.1 ms Max: 843.9 ms Avg: 817.0 ms Current: 796.9 ms
Packet rate
Pkt/s
pkt rate Min: 0 Max: 1.907 K Avg: 1.682 K Current: 0
Packet Inter-Arrival Time (IAT)
IAT Min: 493.3 µs Max: 606.8 µs Avg: 540.8 µs Current: 539.3 µs
Reordering
reordering factor

Figure 3.7: Grafana visualisation of 10 seconds of the INT monitored flow from PSNC to CESNET (absolute delay scale not real)

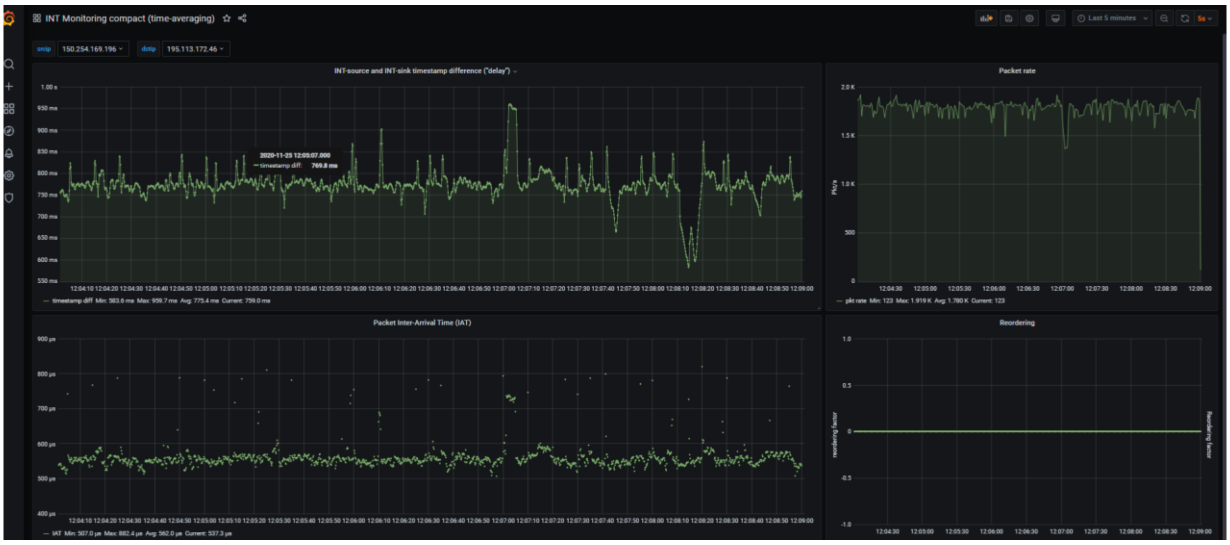
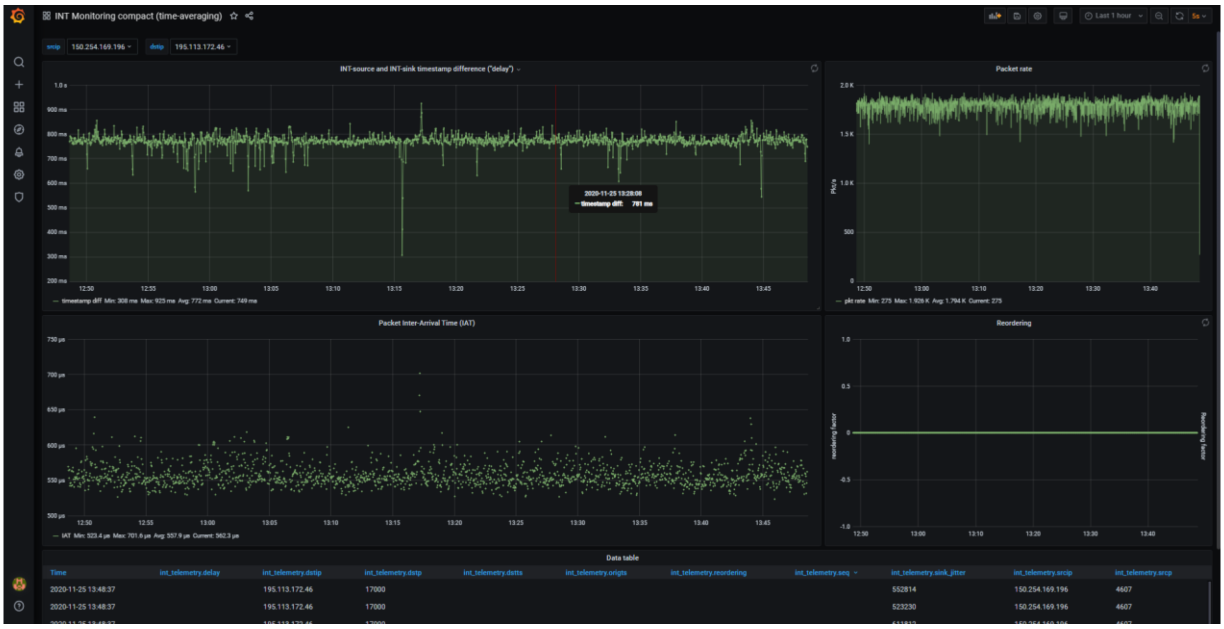

INT Monitoring compact (time-averaging)
srcip 150.254.169.196
dstip 195.113.172.46
Last 5 minutes
INT-source and INT-sink timestamp difference ("delay")
Packet rate
Packet Inter-Arrival Time (IAT)
Reordering
timestamp diff Min: 583.6 ms Max: 959.7 ms Avg: 775.4 ms Current: 759.0 ms
pkt rate Min: 123 Max: 1.919 K Avg: 1.780 K Current: 123
IAT Min: 507.0 µs Max: 882.4 µs Avg: 562.0 µs Current: 537.3 µs

Figure 3.8: Grafana visualisation of 5 minutes of the INT monitored flow from PSNC to CESNET (absolute delay scale not real)


INT Monitoring compact (time-averaging)
srcip 150.254.169.196
dstip 195.113.172.46
Last 1 hour
INT-source and INT-sink timestamp difference ("delay")
Packet rate
Packet Inter-Arrival Time (IAT)
Reordering
timestamp diff Min: 308 ms Max: 925 ms Avg: 772 ms Current: 749 ms
pkt rate Min: 275 Max: 1.926 K Avg: 1.794 K Current: 275
IAT Min: 523.4 µs Max: 701.6 µs Avg: 557.9 µs Current: 562.3 µs
Data table

Time	int_telemetry.delay	int_telemetry.dstip	int_telemetry.dstp	int_telemetry.dstts	int_telemetry.origts	int_telemetry.reordering	int_telemetry.seq	int_telemetry.sink_jitter	int_telemetry.srcip	int_telemetry.srcp
2020-11-25 13:48:37		195.113.172.46	17000					552814	150.254.169.196	4607
2020-11-25 13:48:37		195.113.172.46	17000					523230	150.254.169.196	4607

Figure 3.9: Grafana visualisation of 1 hour of the INT monitored flow from PSNC to CESNET (absolute delay scale not real)

The Plotly visualisation tool is used to create graphs without any aggregation: data from each packet is shown, with a time resolution of 1ns. As such, Plotly graphs can contain hundreds of millions of points, and cannot be generated in real time. They are made available via an HTTP Flask server [Flask] and as a python script. Currently, the generation of four types of graphs is supported: timestamps difference, timestamps differences histogram, inter-arrival time and inter-arrival time histogram, examples of which are shown in Figures 3.10 to 3.14.

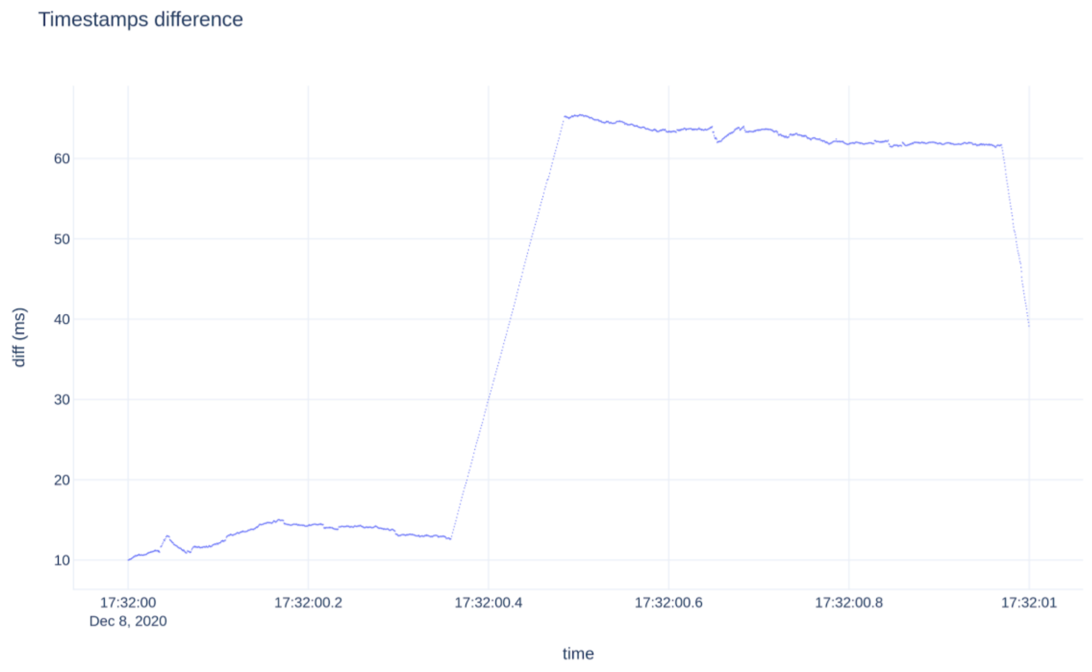


Figure 3.10: Plotly visualisation of INT sink and INT source timestamps difference during 1s for the flow from PSNC to CESNET (1.8K packets/s)

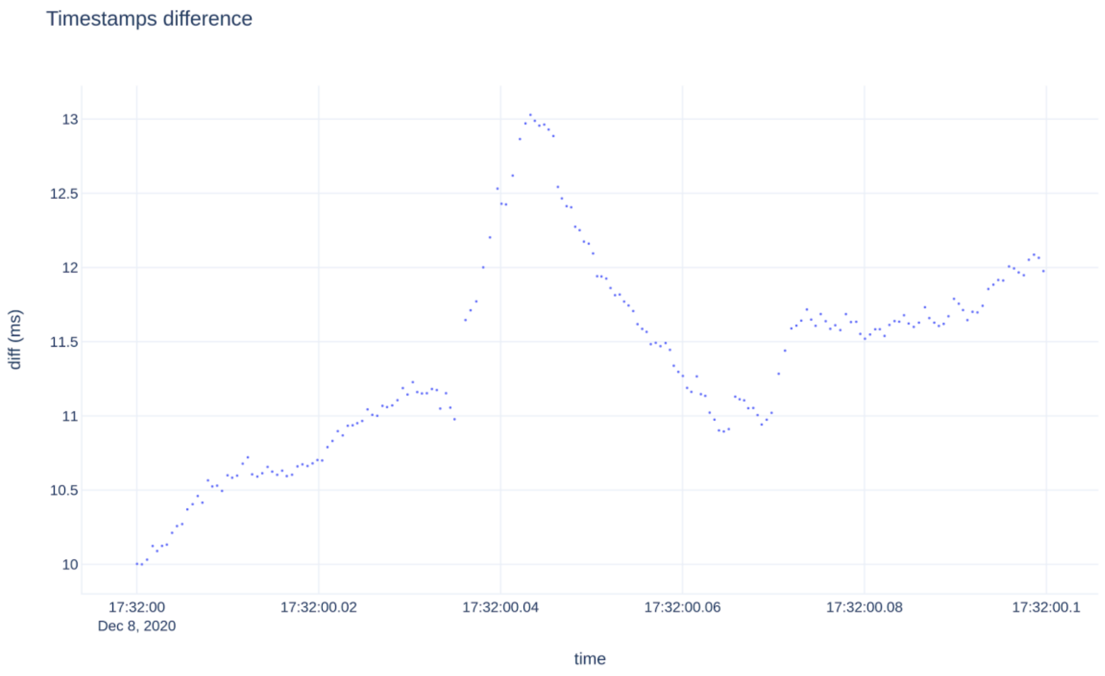


Figure 3.11: Plotly visualisation of INT sink and INT source timestamps difference during 100ms for the flow from PSNC to CESNET (1.8K packets/s)

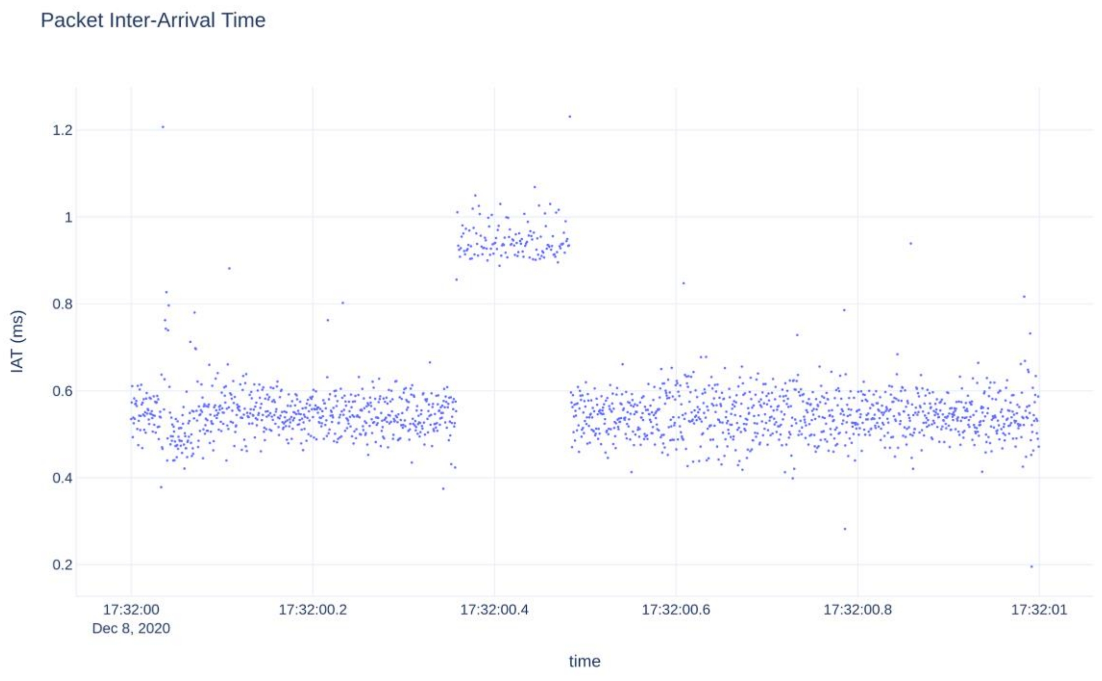


Figure 3.12: Plotly visualisation of packet inter-arrival time during 1s for the flow from PSNC to CESNET (1.8K packets/s)

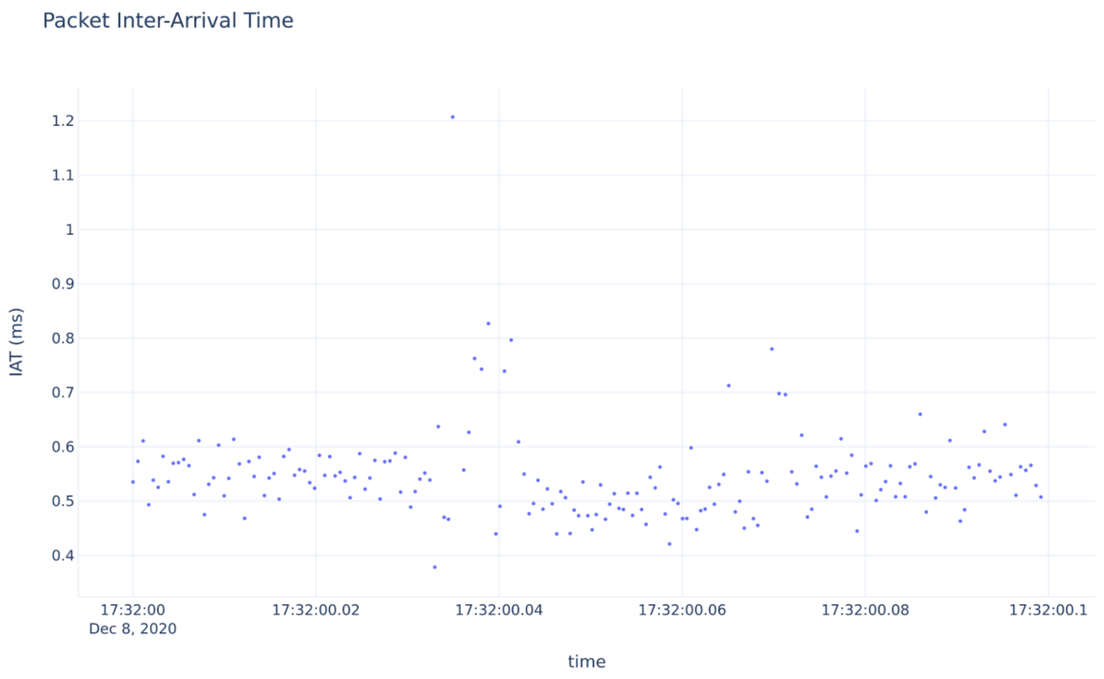


Figure 3.13: Plotly visualisation of packet delay variation during 100ms for the flow from PSNC to CESNET (1.8K packets/s)

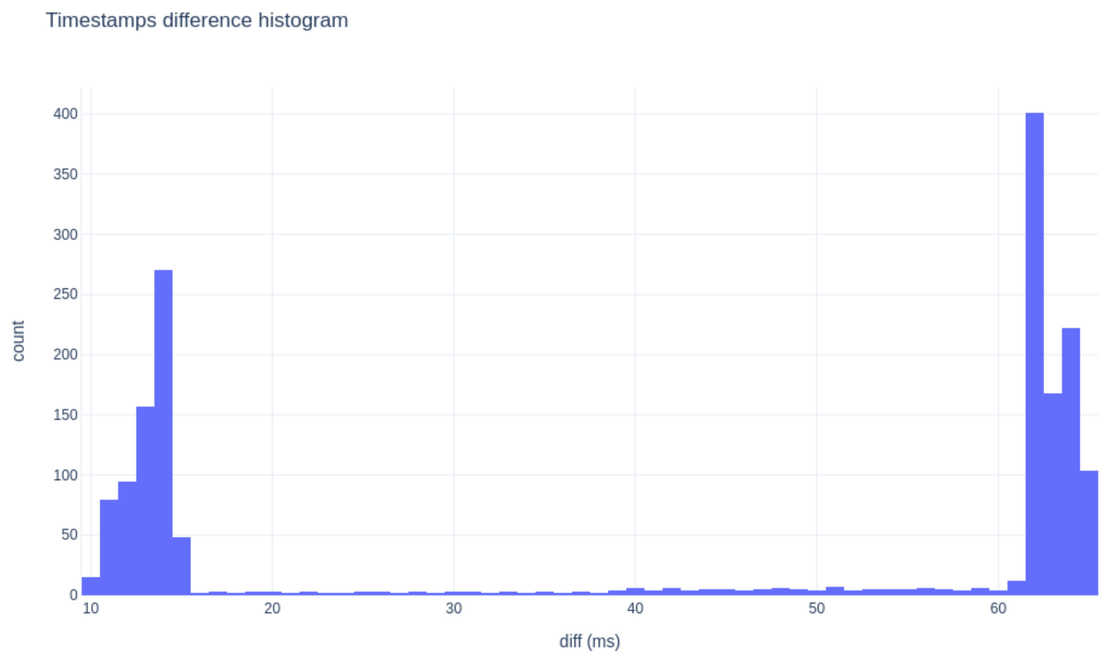


Figure 3.14: Timestamps difference histogram for 1s of the INT monitored flow from PSNC to CESNET (1.8K packets/s)

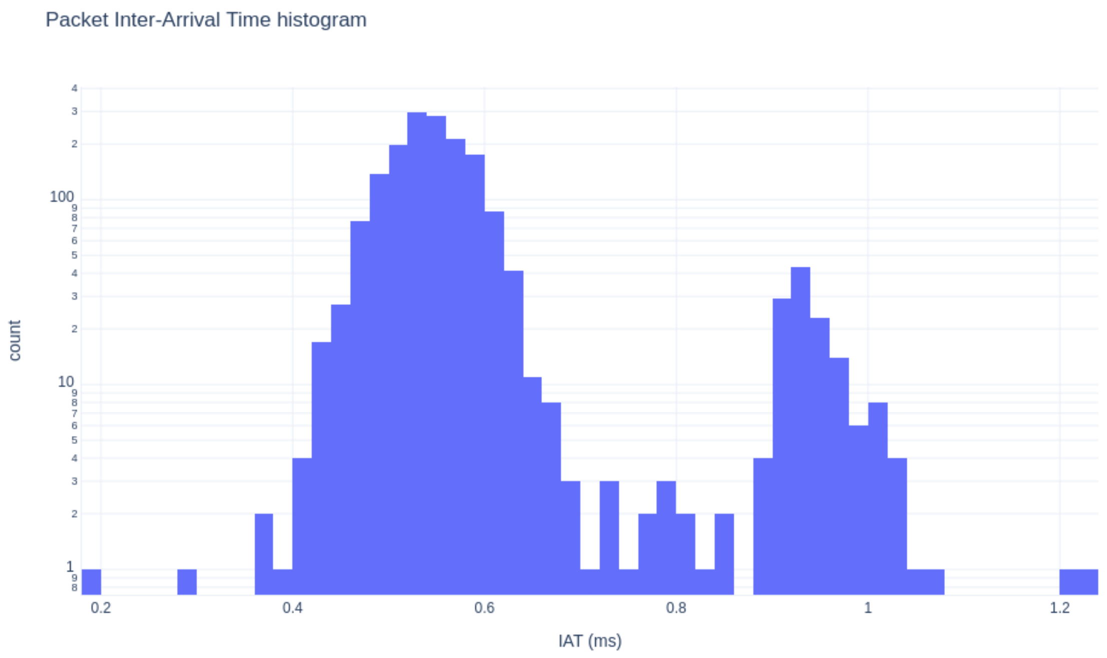


Figure 3.15: Packet inter-arrival histogram for 1s of the INT monitored flow from PSNC to CESNET (1.8K packets/s)

The reported plots are just an example of packet behaviour in a production network. The traffic monitored at microsecond time granularity shows a long tail for IPDV, as well as in the absolute value of the e2e delay. Some diagrams (e.g., Figure 3.11 and Figure 3.12) exhibit patterns that may be interpreted as short-lived queueing (up to many milliseconds). A delay in the order of milliseconds may also be due to the sending node, rather than the switches and the network itself. More examples, with finer time granularity, are given in the following section.

The time granularity chosen for the graphs is close to single-packet transmission time and inter-packet time delay. The servers are connected at 1Gbps which implies that the time to transmit a single packet is about 1.1 milliseconds, while the inter-packet delay is about 5.5ms at 1.8 Kpackets per second and 3.8ms at 260 Kpackets per second, as displayed by the measurements. A more detailed computation is provided in Table 3.3 below.

The current DPP INT monitoring infrastructure allows gathering of up to 260K INT reports per second, which is adequate to monitor 1Gbps flows with minimum size UDP packets.

Flow speed	Only 64B packets (+20B interpacket gap)	Only 1518B packets (+20B interpacket gap)
1Gbps	1.49M packets/s	81.3K packets/s
10Gbps	14.9M packets/s	813K packets/s
100Gbps	149M packets/s	8.13M packets/s

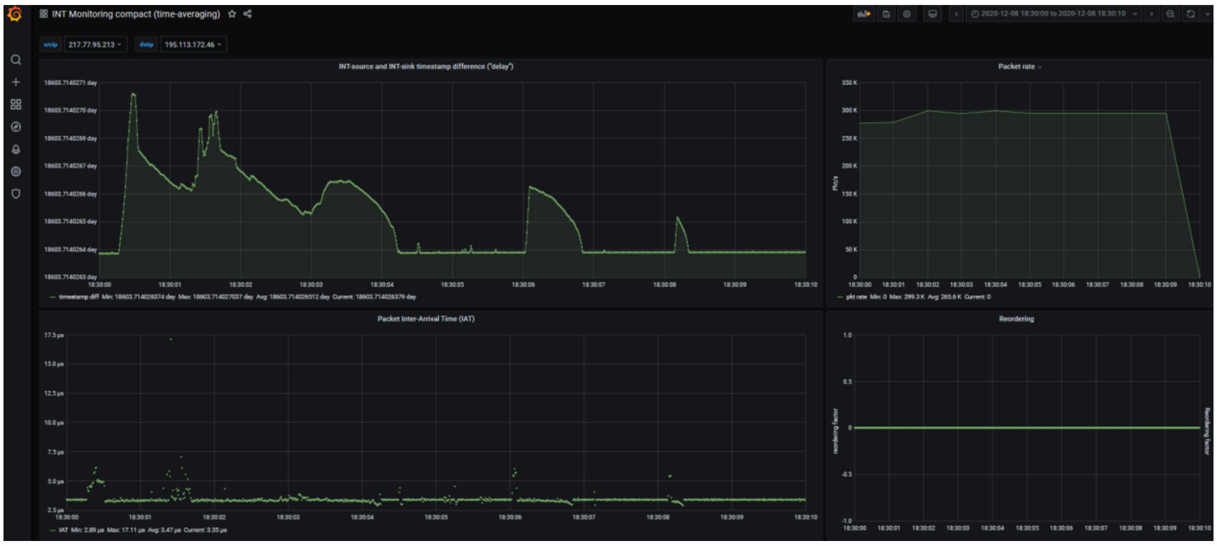
Table 3.3: Number of packets for different [flow rates]

3.4 INT Telemetry Best Performance Results Achieved for a Single Measured Flow

The largest flow that was tested was a UDP packet transmission from FBK to CESNET, sent at a rate of 300K packets per second. However, this flow rate was too high to be stored in a prepared single node instance of the InfluxDB database located in PSNC. Packet rates up to about 260K packets per second can still be collected in that database. The measured data was visualised using both Grafana and Plotly graphs. At such a high rate, Grafana and the INT data exporting process at the sink were the first two elements of this infrastructure to exhibit some issues.

Displaying Grafana graphs was problematic because of long response times for multiple parallel queries, which requires the processing of about a million INT data points in InfluxDB.

High-resolution Plotly graphs were generated for 1s, 100ms, and 10ms periods. Although these graphs visualise every single packet in the database, they contain gaps (see Figures 3.17 and 3.18). Most probably, these gaps are the results of some dropped UDP packets carrying INT headers between FBK and CESNET. Further problems are listed in section 4 which discusses issued encountered.



— timestamp diff Min: 18603.714026374 day Max: 18603.714027037 day Avg: 18603.714026512 day Current: 18603.714026379 day

— pkt rate Min: 0 Max: 299.3 K Avg: 265.6 K Current: 0

— IAT Min: 2.89 µs Max: 17.11 µs Avg: 3.47 µs Current: 3.35 µs

Figure 3.16: Biggest monitored flow (260k packets per second) visualised in Grafana (10 seconds) (absolute delay scale not real)

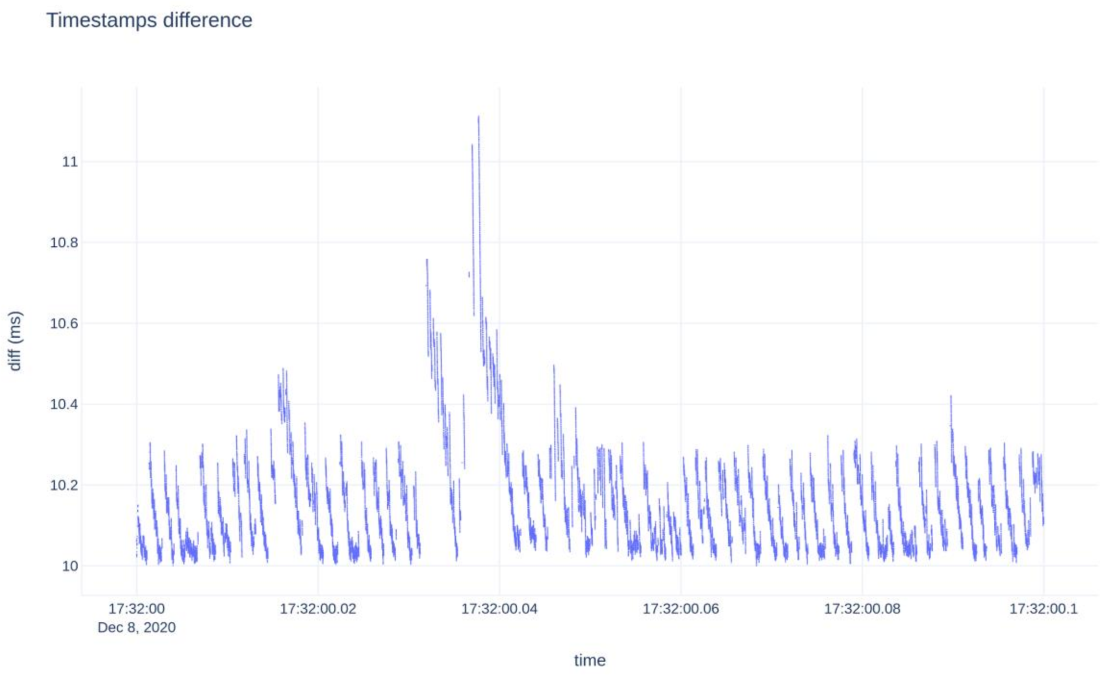


Figure 3.17: Plotly visualisation of INT sink and INT source timestamps difference for 100ms interval (260K packets/s)

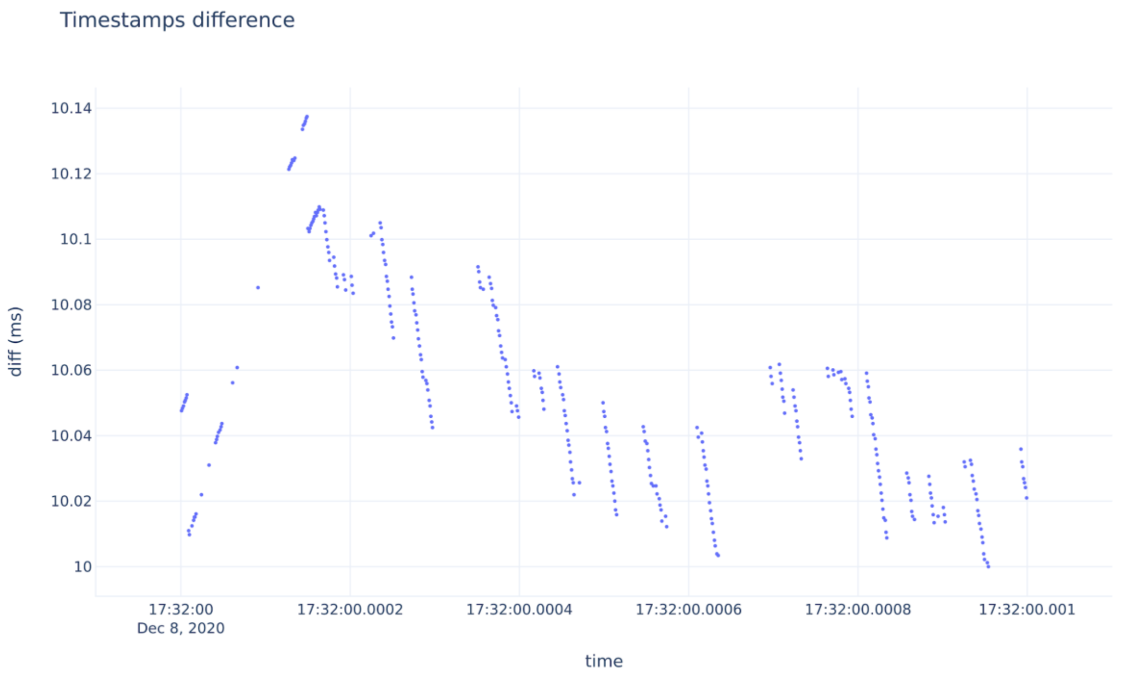


Figure 3.18: Plotly visualisation of INT sink and INT source timestamps difference for 1ms interval (260K packets/s)

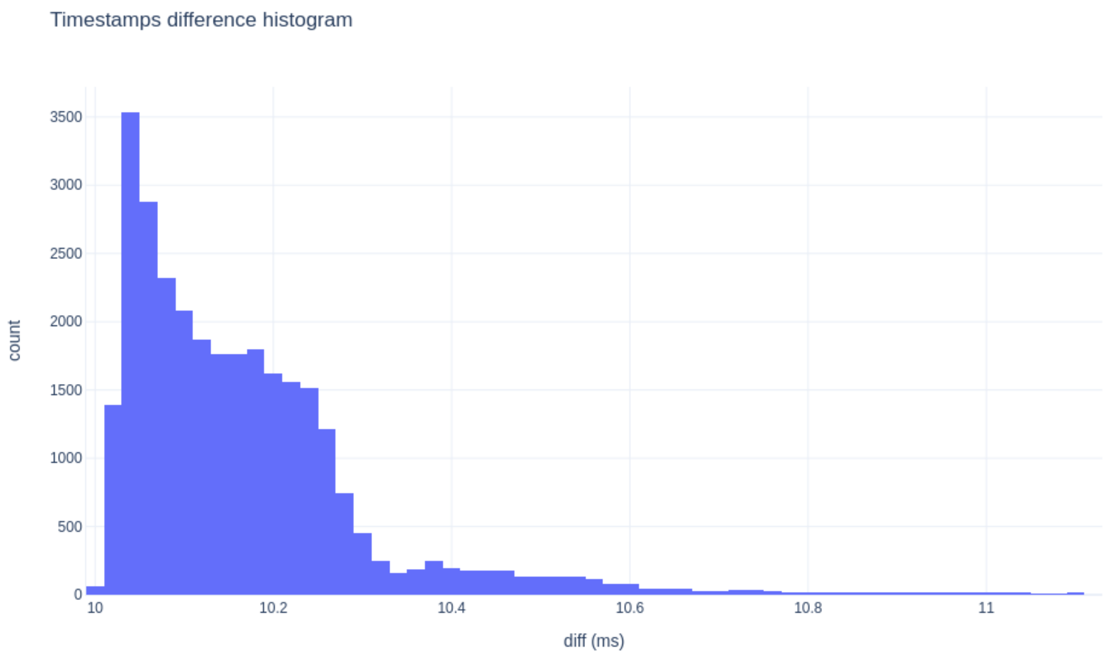


Figure 3.19: Plotly visualisation of timestamps difference histogram for 1s interval (260K packets/s)

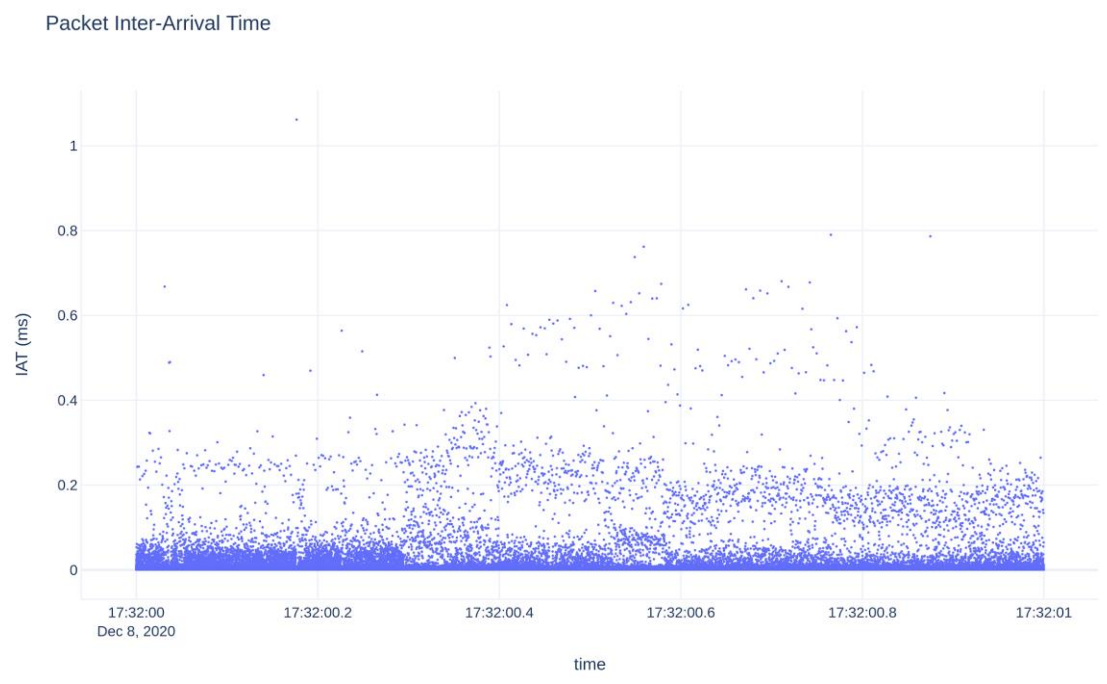


Figure 3.20: Plotly visualisation of packet inter-arrival time for 1s interval (260K packets/s)

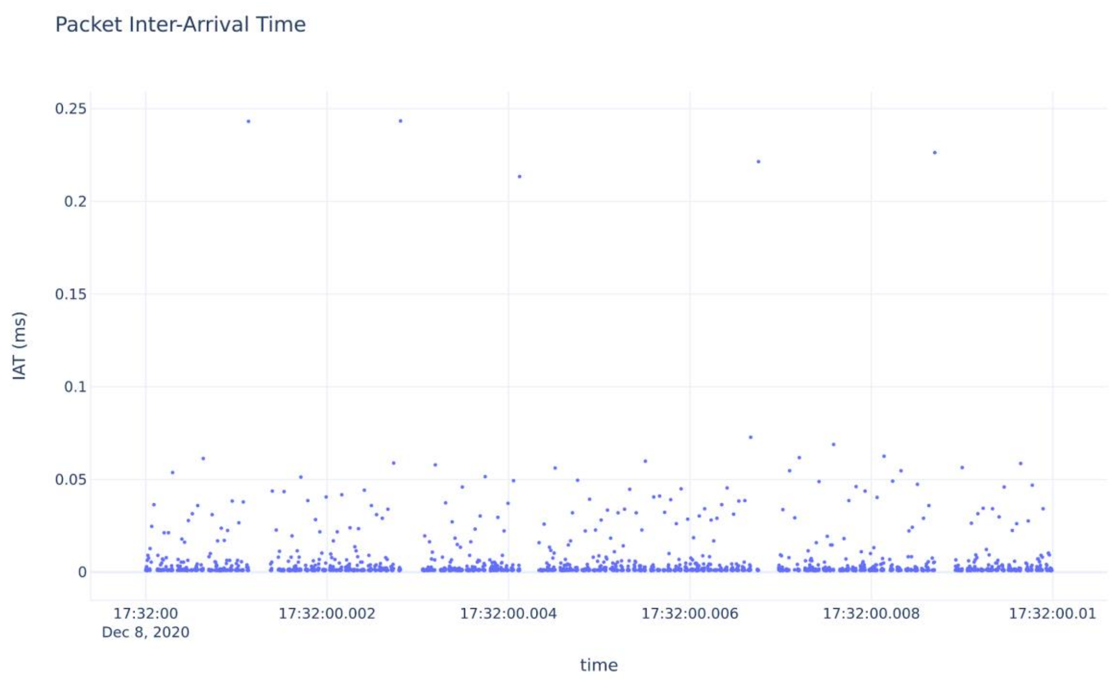


Figure 3.21: Plotly visualisation of packet inter-arrival time for a 10ms interval (260K packets/s)

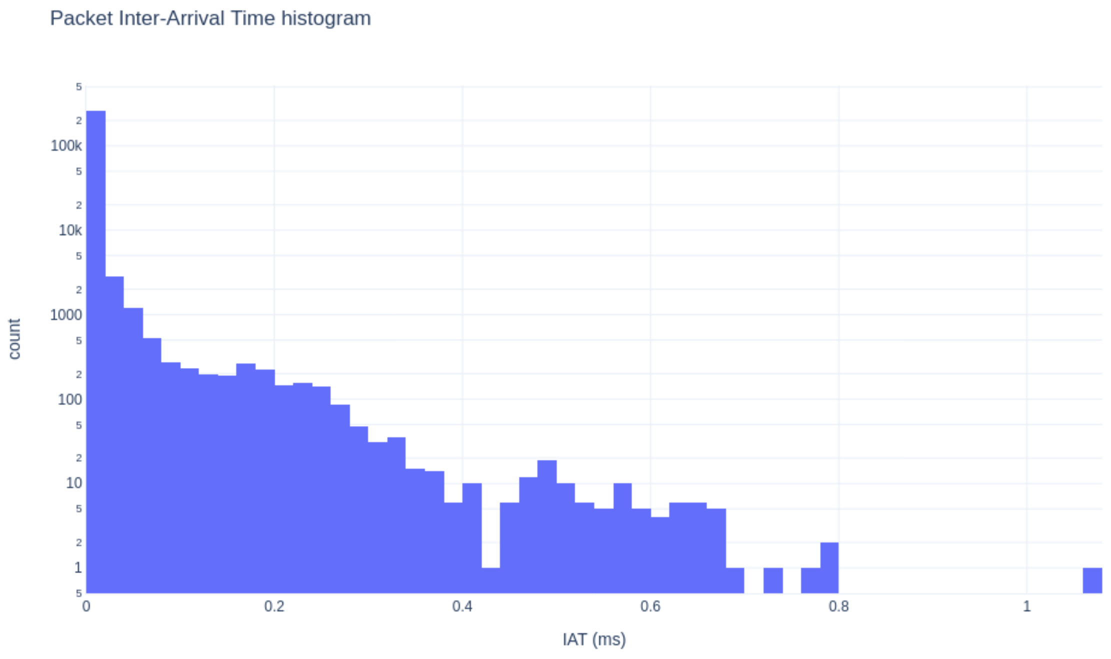


Figure 3.22: Plotly visualisation of packet inter-arrival time histogram for a 1s interval (260K packets/s)

4 INT Testbed: Issues Encountered and Lessons Learned

The following issues were encountered while running the experiments described above in the INT testbed spanning the three sites.

1. <u>Wedge EdgeCore Tofino chip timestamps synchronisation with global clock</u>

The timestamp in Tofino is only 48 bits with 1ns resolution, which means that the time is reset approximately every 3 days. Tofino chips use an internal clock and, given the hardware version of the line card, its synchronisation to any external time has not yet been easily achieved. Due to this complexity the INT visualisation plots show delay values that are very large and not correct. However, packet inter-arrival INT measurements are accurate.

Another constraint to consider when creating P4 programs is that Tofino chips do allow complex computations on headers to maintain line rate, such as multiplication to convert the time expressed in nanoseconds to other time units (e.g., microseconds). This constraint has been overcome by standardising the time unit to nanoseconds in the INT experiments performed.

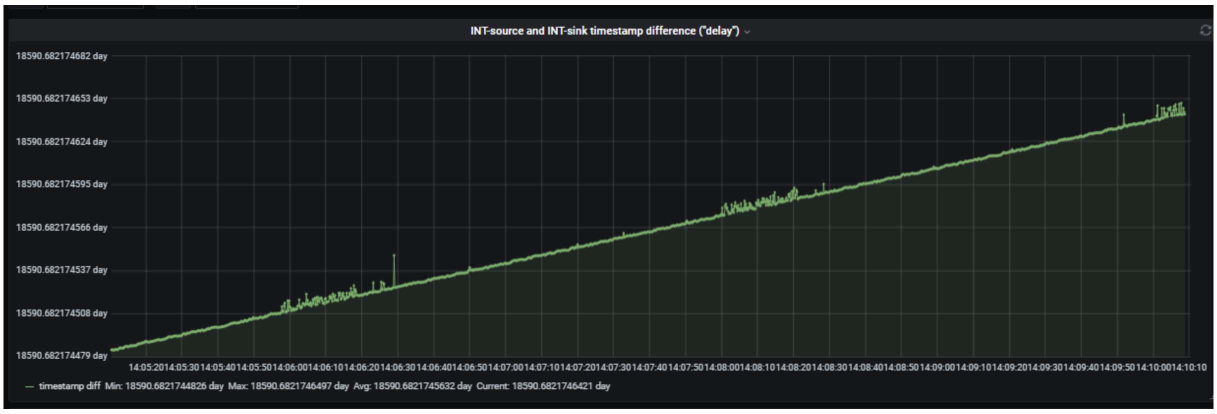


Figure 4.1: Visible effect of lack of clock synchronisation in the Wedge EgdeCore switch

2. <u>PSNC virtual INT environment low performance and bad stability</u>

INT source and INT transit nodes in PSNC are based on bmv2 virtual switches and hosts running in a Mininet environment. This setup allowed the generation and transmission of up to about 2000 UDP packets per second. Currently, generated packets are 70 bytes long (with 6 bytes of UDP payload). The Mininet environment proved to be unstable at these rates, crashing and requiring daily reboots. However, this lack of stability allowed significant testing and debugging of INT capabilities, confirming that BMv2 is a valid tool for initial INT validation. The testbed will be evolved by adding two Arista

7170 switches in PSNC, which are based on Barefoot Tofino chips. The availability of HW based switches will permit testing at significantly higher rates and capacity.

3. <u>Incorrect identification of non-INT packets as INT packets</u>

In the first version of the INT sink node, only those packets that contained UDP as the transport protocol and the value 0x20 in the DSCP entry of the IPv4 header were identified as INT packets. However, this identification rule proved to be insufficient as these conditions are often randomly met by non-INT packets. After adding a check on the Switch ID entry of the INT metadata header, this problem was temporarily solved with a final rule that filters incoming packets based on the source IP address, with the above conditions applied only to flows coming from predefined servers (FBK and PSNC). The solution is to add probe bytes for the INT header (see chapter about INT location in the packet).

4. <u>Performance of exporting INT data to InfluxDB database</u>

While the measured maximum write performance of the InfluxDB database using a single TCP connection was measured to be 85K-100K INT data entries per second, the actual maximum write performance using the INT collector located in CESNET was only 15K INT data entries per second. The DPP team investigated the cause of such a large difference, finding that it was due to the use of an encrypted VPN tunnel between CESNET and PSNC to carry the extracted INT reports. After removing this, switching to direct Internet access to InfluxDB, a single TCP connection write performance increased to 64K INT data entries per second. The additional write performance increase was achieved by using multiple parallel TCP writers, allowing sending up to 260K INT data entries per second to the Influx database.

5. <u>Low performance and low resolution of the INT data visualisation in Grafana</u>

Initially, the content of INT headers was exported to the INT database. Calculation of packet latency or jitter was done at the time of Grafana queries to InfluxDB. However, such queries prevented the use of the time aggregation functionality of Grafana graph panels, which resulted in very poor performance of the INT data presentation for long time windows. This problem was drastically mitigated by making the required computation of latency and jitter values directly inside the INT sink node.

The second problem in Grafana was related to the unavailability of microsecond and nanosecond resolutions for time series graphs, meaning packet-level INT data couldn't be properly displayed in the Grafana dashboard. This problem was solved by developing an additional tool for INT visualisation based on the Plotly library.

A third problem is the low performance of the dashboard, related to the insufficient performance of InfluxDB read queries compared to the write rates of measurement results under high-volume data ingested to the InfluxDB. For example, a query to visualise all the data for a 250Kpps flow for a short period of just one second requires about 50 seconds to execute. This implies a low read performance, of about 5K data points per second, which greatly affects the visualisation performance of a large set of data points of a flow. Most probably, this low read throughput was caused by large compression of INT data resulting in costly read operations. Currently, all INT data is stored in InfluxDB in order to allow full debugging and validation of INT data. In the future, different technologies can be investigated to process/reduce the high-volume stream of INT data, reducing the need to store all INT headers data.

5 Interoperability Considerations

Several versions of the In-Band Network Telemetry specification have been published [P4 Specs]: 0.5, 1.0, 2.0, and the latest version 2.1. Some of the earlier versions are still in use, for example in the popular SDN controller [ONOS], which uses an INT implementation that is not fully compatible with version 1.0 of the specifications.

The IETF draft standard for In-situ [OAM] also provides an alternative to the INT specification. In-situ OAM exhibits many differences in terms of protocol coding in comparison to INT. However, INT v2.1 and In-situ OAM are converging with respect to the functionalities they offer.

The DPP team has developed an INT implementation compatible with the ONOS INT implementation (the team initially planned to use ONOS INT for its Wedge EdgeCore switches, but that proved unfeasible due to incompatibilities between the Tofino SDK and the required virtualisation environment). Additionally, the DPP team also implemented INT version 2.0, but this version has yet to be fully tested in the testbed.

The different versions of INT differ in field encoding details but offer the same overall functionality. Switching between different INT versions requires changing the P4 INT implementation in the switches but does not affect other elements of the INT monitoring system (i.e., INT exporter, INT database and INT visualisation).

In future upgrades, the DPP team plans to implement INT v2.1, where Domain Namespace IDs were added. The usage of Domain Namespace IDs allows more freedom in how to decode and interpret INT metadata. This will permit different INT implementations to interoperate in multi-domain and multi-technology networks.

Note that the P4 implementation presented does not currently support IPsec, and even if this were to be supported in the future, it could only work in domains that do not implement IPsec integrity checking where INT is used. This is because the additional payload inserted (and later transparently removed) would make the Integrity Check Value in the IPsec Authentication Header of packets temporarily invalid until the INT headers are removed.

6 Conclusions and Future Work

The DPP team has performed extensive In-Band Network Telemetry measurement and visualisation of UDP flows, with packet rates up to 260K packets per second. Addition and removal of INT headers were performed using an EdgeCore Wedge Tofino switch and FPGA-based network card respectively. INT data was gathered in InfluxDB and presented using both Grafana and a Plotly-based tool, the latter of which can display up to nanosecond resolution INT data with thousands of INT reports collected per second.

The experiments which generated 260K INT reports per second demonstrated that each packet in a fully loaded 1Gbps flows with minimum size UDP packets can be monitored using INT.

The performances achieved so far are dependent on the local LAN lines capacity constraint at 1Gbps and the use of BMv2.

The use of INT in every packet represents an extreme use case and will probably not be needed for many of the expected use cases. However, should higher speeds be required, several techniques may be considered to apply INT to 10Gbps or even 100Gbps flows, including:

- In-data plane reduction of the number of INT reports by skipping INT data insertion in a packet if the INT data is very similar to data inserted in the previous packet.
- In-data plane generation of INT data aggregates, histograms and events when a such final monitoring output is required by users/applications of the monitoring system.
- Intermediate processing of INT data using message broker systems (e.g. Kafka and Kafka Streams) allowing the reduction of the amount of data to be stored in the INT database.
- Using InfluxDB only to store INT data that for example represents "large" variations with respect to the expected value, and data received from low-level INT data reduction/aggregation.
- Using machine learning algorithms to analyse INT data.

References

[Broadcom]	https://www.broadcom.com/
[DataSeries]	https://medium.com/dataseries/analysis-of-the-storage-mechanism-in-influxdb-b84d686f3697
[DPP_PMW]	https://wiki.geant.org/download/attachments/133765749/200305-PerfMonWS-Zagreb_WP6_T1_DPP_Campanella_v2.pptx
[Flask]	https://flask.palletsprojects.com/en/1.1.x/
[flow rates]	http://rickardnobel.se/actual-throughput-on-gigabit-ethernet/
[Grafana]	https://grafana.com/
[InfluxDB_lp]	https://docs.influxdata.com/influxdb/v1.8/write_protocols/line_protocol_reference/
[Intel Tofino]	Tofino chip. From Barefoot, now Intel https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/tofino-series/tofino.html
[INT v2.1]	In-Band Network Telemetry (INT) Dataplane Specification Version 2.1, May 2020 https://github.com/p4lang/p4-applications/blob/master/docs/INT_v2_1.pdf
[iperf]	https://iperf.fr/
[Noviflow]	Networking software company https://noviflow.com/
[OAM]	https://datatracker.ietf.org/doc/draft-ietf-ippm-ioam-data/
[ONOS]	https://opennetworking.org/onos/
[P4 Specs]	https://p4.org/specs/
[P4app]	https://github.com/p4lang/p4app
[P4bm]	https://github.com/p4lang/behavioral-model
[P4docs]	https://github.com/p4lang/p4-applications/tree/master/docs
[Plotly]	https://plotly.com/python/
[Scapy]	https://github.com/secdev/scapy/
[TBDW]	The First Telemetry and Big Data Workshop, 10th November 2020, https://wiki.geant.org/display/PUB/Telemetry+and+Big+Data+Workshop
[VXLAN-GPE]	https://tools.ietf.org/html/draft-ietf-nvo3-vxlan-gpe-10

Glossary

CAPEX	Capital expenditure
DPDK	Data Plane Development Kit
DPP	Data Plane Programming
DSCP	Differentiated Services Code Point
FBK	Fondazione Bruno Kessler
FPGA	Field Programmable Gate Array
GRE	Generic Routing Encapsulation
INT	In-Band Network Telemetry
IPDV	IP Packet Delay Variation
NIC	Network Interface Controller
RDMA	Remote Direct Memory Access
SDK	Software Development Kit
Tap	Test access point
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VXLAN	Virtual Extensible LAN protocol