Towards monitoring hybrid next-generation software-defined and service provider MPLS networks

Zaballa, Eder Ollora ; Franco, David; Thomsen, Signe Erdman; Higuero, Marivi; Wessing, Henrik; Berger, Michael Stübert

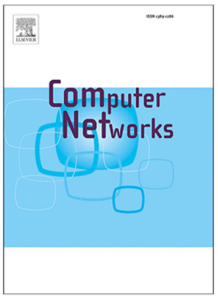
Published in:
Computer Networks

Link to article, DOI:
10.1016/j.comnet.2021.107960

Publication date:
2021

Document Version
Peer reviewed version

Journal Pre-proof



Towards monitoring hybrid next-generation software-defined and service provider MPLS networks

Eder Ollora Zaballa, David Franco, Signe Erdman Thomsen, Marivi Higuero, Henrik Wessing, Michael S. Berger

PII: S1389-1286(21)00094-3
DOI: https://doi.org/10.1016/j.comnet.2021.107960
Reference: COMPNW 107960

To appear in: *Computer Networks*

Received date : 14 July 2020
Revised date : 16 February 2021
Accepted date : 22 February 2021

Please cite this article as: E.O. Zaballa, D. Franco, S.E. Thomsen et al., Towards monitoring hybrid next-generation software-defined and service provider MPLS networks, *Computer Networks* (2021), doi: https://doi.org/10.1016/j.comnet.2021.107960.

Towards Monitoring Hybrid Next-Generation Software-Defined and Service Provider MPLS Networks

Eder Ollora Zaballa[a], David Franco[b], Signe Erdman Thomsen[a], Marivi Higuero[b], Henrik Wessing[a], Michael S. Berger[a]

[a]*DTU Fotonik, Technical University of Denmark, Kongens Lyngby, Denmark*
[b]*Department of Communications Engineering, University of the Basque Country, Bilbao, Spain*

Abstract

The market prediction for network deployments has positioned Software-Defined Networks (SDN) as the first of the options for changing local, transport, or cloud networks. Since the OpenFlow protocol gained traction and evolved in the last versions, the possibilities for expanding network capabilities to deploy custom services have risen considerably. With next-generation SDN (NGS), flexibility has grown as data plane programming languages, such as P4, and Data-Control Plane Interface (DCPI) protocols like P4Runtime have appeared. Furthermore, the ability to program the data plane has opened the possibilities to develop new network telemetry approaches, such as In-band Network Telemetry (INT). A transition to partially incorporated SDN, also known as hybrid SDN, often involves considerable complexity, especially when legacy devices implement non-open standards and protocols. Therefore, incorporating programmable SDN devices and deploying network telemetry protocols on top of existing legacy devices is still challenging. This research focuses on deploying and integrating the INT protocol using programmable P4 switches over a hybrid SDN network. We describe and implement the required control plane applications and data plane configuration, and discuss the constraints that need to be managed so that P4 programmable switches can interact with the rest of the MPLS legacy devices. In this sense, we discuss P4 switch placement alternatives to maximize their performance and usability in a hybrid SDN network. Then, we validate the INT-based monitoring system by ensuring traffic forwarding using several INT header placements. In these tests, we demonstrate the feasibility of merging P4 switches running INT traffic and legacy devices, presenting the requirements to accomplish hybrid next-generation SDN (NGS) architectures. Besides, we provide new monitoring features, such as MPLS label verification, and we also use telemetry data to feed-back traffic forwarding applications for traffic engineering purposes. We finally show the time that packets spend in the pipeline comparing different parsing and actions performed in different cases.

Keywords: Hybrid SDN, P4, In-band Network Telemetry, monitoring, MPLS, next-generation SDN

1. Introduction

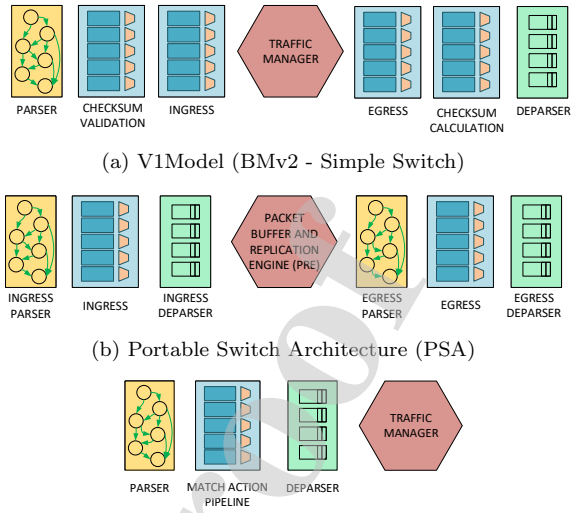
In recent years, Internet Service Providers (ISPs) have witnessed an increasing demand for capacity from their core network devices. This fact has become even more noticeable during the period of lockdown that occurred in many countries due to the COVID-19 pandemic. According to a recent study published by the New

Email addresses: eoza@fotonik.dtu.dk (Eder Ollora Zaballa), david.franco@ehu.eus (David Franco), setthomsen@gmail.com (Signe Erdman Thomsen), marivi.higuero@ehu.eus (Marivi Higuero), hewe@fotonik.dtu.dk (Henrik Wessing), msbe@fotonik.dtu.dk (Michael S. Berger)

York Times [1], content providers like Netflix and video streaming platforms like Zoom have grown in terms of their daily sessions, by 16% and 200%, respectively. This unprecedented increment in traffic has impacted the performance of monolithic legacy core network architectures, which are often based on Multiprotocol Label Switching (MPLS), and has revealed the widely understood necessity of evolving towards next-generation core networks.

In this regard, Software-Defined Networking (SDN) and Network Function Virtualization (NFV) are promising technologies that have been proposed to overcome many limitations of the static architectures deployed in the current networks. NFV brings the concept of virtualization to network services, and SDN provides the necessary abstractions to make the network dynamic and programmable. NFV, which was initially conceived as a Virtual Machine (VM)-based virtualization, is now evolving to a lightweight container-based approach [2] where services are built to avoid the performance issues introduced by the hypervisor layer. Moreover, the Function as a Service (FaaS) paradigm is rising as a near-future architecture for NFV [3]. It decomposes network services into several lightweight functions that are instantiated on-demand.

Alternatively, SDN is widely used in high-performance, hyperscale data centers (e.g., Google, Facebook, Amazon, etc.). The commercialization of cost-effective OpenFlow-based devices has also allowed many organizations to optimize their critical applications. However, the OpenFlow protocol's possibilities are bound to the OpenFlow version that the forwarding device implements. Next-generation SDN (NGS) is designed to overcome such limitations by allowing programmable data planes and the implementation of custom packet processing pipelines (see Figure 1). P4 [4] has emerged as the new *de facto* data plane programming language, allowing developers to define which headers a switch will parse, which fields and headers are used to match on tables, and which actions the switch will perform when table matches happen. Among various



(a) V1Model (BMv2 - Simple Switch)

(b) Portable Switch Architecture (PSA)

(c) SimpleSumeSwitch Architecture (NetFPGA SUME)

Figure 1: (a) V1Model architecture used in the BMv2 Simple Switch target. (b) The Portable Switch Architecture (PSA) is the specification of a target architecture published by the P4.org Architecture Working Group. (c) The SimpleSumeSwitch P4 architecture for the NetFPGA SUME.

options, the control plane protocol for P4-based devices (i.e., P4 switches) is based on P4Runtime [5], a control plane specification that enables runtime control of P4 switches.

Programmable data planes yield new network monitoring approaches that enable the attachment of telemetry information to user data packets (i.e., in-band telemetry). In-band Network Telemetry (INT) [6] is an implementation of such an approach that leverages P4 and P4 switches to provide telemetry reports without the involvement of the control plane. P4-based in-band telemetry can be applied to improve the monitoring system (e.g., by avoiding probe packets) and to develop tailored monitoring functionalities for a legacy MPLS network. From an ISP perspective, the complete migration of a legacy architecture to an NGS network is costly in terms of Capital Expenditures (CAPEX) and Operational Expenditures (OPEX). A transition to partially incorporated SDN, also known as hybrid SDN, is presented here as an alternative to avoid incurring an excessive expense. Therefore, deploying an effective network monitoring system on top of the

existing legacy devices and programmable SDN devices, hereby named hybrid NGS monitoring, is still a challenge.

In this paper, we develop an INT-based monitoring system for hybrid NGS networks based on MPLS. We provide the control plane design to achieve an operational integration of the P4 switches within the MPLS network and the design of the data plane processing pipeline to implement INT-based monitoring. Additionally, several monitoring scenarios are tested using physical programmable devices and emulation tools to validate the developed monitoring features.

The following are the main contributions of this work:

- Provide new monitoring features and traffic engineering techniques based on INT and programmable data planes (Sections 5 & 6).
- Describe the constraints that need to be managed to integrate P4 switches within an MPLS network (Section 3).
- Discuss P4 switch placement alternatives to optimize the performance of hybrid SDN networks (Section 3.4).
- Implement the required control plane applications and data plane configuration (Section 4).
- Validate and test the performance of the INT-based monitoring system for a hybrid SDN-MPLS scenario (Section 6).

The remainder of the paper is organized as follows. Section 2 provides an overview of the current literature, and Section 3 explains the basic concepts of INT and describes the main constraints related to the integration of MPLS and P4 in a hybrid SDN (hybrid NGS). After that, the hybrid NGS monitoring system is described in Section 4, including the control and data plane of P4 devices in the MPLS network. Then, Section 5 introduces different monitoring features and scenarios that are later tested in Section 6, which shows the results obtained from emulating the aforementioned scenarios. Finally, Section 7 depicts the main conclusions of this work.

2. Related work

This section reviews relevant work that includes topics like MPLS, P4, SDN, and INT (i.e., telemetry). Finding a significant amount of work that provides all these features together is complicated, as NGS features were developed only recently. However, this section tries to present studies that include two or more of the above subjects and points out how they differ from the current study. Therefore, related work will be organized into sections that focus on various combinations of the aforementioned subjects.

2.1. SDN and MPLS

Related work that leverages SDN and MPLS has primarily focused on providing better traffic engineering, resource allocation, and network management. Bahnasse et al. [7] designed a dynamic model of bandwidth allocation using hybrid SDN. The authors demonstrated that their model provides better results than the Maximum Allocation Model (MAM) and Russian Doll Model (RDM) for Internet Control Message Protocol (ICMP) and Voice over Internet Protocol (VoIP) tests, whereas the Hypertext Transfer Protocol (HTTP) response was between MAM and RDM for most tests. A subsequent paper by Bahnasse et al. [8] also presented a new method for managing MPLS VPNs using a hybrid SDN model. The authors demonstrated their idea's feasibility and showed a particular time improvement with the deployment of MPLS Virtual Private Networks (VPNs) using their model compared with the manual method. Tajiki et al. [9] formulated a heuristic algorithm to improve the performance of flow rerouting and Label Switch Path (LSP) recreation. The simulation, conducted in MATLAB, showed that their scheme prevents the congestion of shortest-path-based schemes, increasing network throughput. Similarly, average link utilization is higher but also path length (which should not affect the average latency values as path selection implements delay constraints based on path length). While the aforementioned studies have tried to integrate SDN into legacy networks, they did not integrate virtual or physical

devices that support OpenFlow or P4Runtime. Our study includes physical P4-programmable devices and determines the requirements for integrating this with virtual devices running Cisco IOS images.

2.2. P4 and hybrid SDN

Studies that integrate P4/P4Runtime and hybrid SDN are not abundant. The works that combine P4 and hybrid SDN focus on security and paradigms that establish path optimization. Feng et al. [10] proposed a security mechanism that first exchanges critical switches in a topology (placement problem). It also attracts traffic to P4 programmable devices and either forwards traffic or sends packets to the control plane for further inspection. Martinez-Yelmo et al. [11] designed and tested a hybrid software switch that supports a forwarding algorithm based on a centralized control plane with a distributed approach based on the ARP-Path protocol. Their results show the ARP-P4 scheme to be effective after ARP-Path and ECMP P4Runtime tests. However, the performance does not meet expectations, primarily that of Behavioral Model 2 (BMv2). While these cases are interesting for hybrid SDN, the available research lacks significant studies, including telemetry, P4, and hybrid SDN. No study was found that incorporated the three aforementioned topics and that included physical P4 switches and commercial legacy routers that test MPLS traffic.

2.3. Network monitoring approaches

Network monitoring studies for SDN and non-SDN networks are abundant, although the amount of work focusing on hybrid SDN is limited. Indeed, Abushagur et al. [12] point out the insufficient amount of work that concerns monitoring and hybrid SDN. The authors review the existing monitoring approaches for traditional and SDN and point out the importance of SDN controllers for polling SDN switches and peer with legacy routers to compile a global link utilization image. Regarding traditional networks, the authors reviewed the research based on active Network Tomography (NT), which estimates the information of individual links by monitoring a subset of the entire network. Focusing on SDN networks, the authors reviewed the current work on optimal polling schemes [13] and flow-level monitoring applications using flow sampling methods [14]. Finally, Abushagur et al. proposed a mechanism that adhered to the study's proposed research direction after analyzing the related work. Their proposal included SDN switches as ingress and egress nodes of a Segment Routing-based (SR) network and legacy routers that paired with the SDN controller to complete the rest of the nodes. The proposal did not include any testing mechanism or implementation but instead proposed a research direction to follow. The related work and the proposed mechanism describe a similar approach to in-band telemetry protocols and programmable data planes for hybrid SDN networks.

2.3.1. Analogous approaches to INT

Traditional network monitoring has relied on statistics, and probe packets such as ICMP echo requests/replies to assess the status of the network. Recent innovations provide greater insight into network behavior by generating detailed telemetry metadata reports that can be recorded into the data packet itself, generating so-called in-band telemetry. The term "INT" has often been used to refer to in-band data plane telemetry in general and is not limited to In-band Network Telemetry. For example, MPLS INT (MINT) [15], defined by the Internet Engineering Task Force (IETF), is a telemetry mechanism that records flow specific information from end stations and/or switches across an MPLS network. It employs a per-hop method to collect data, and it can perform localized end-to-end analytic operations on the data. Similarly, INT for 6TiSCH [16] offers a flexible monitoring solution with minimal resource consumption and communication overhead that enables 6TiSCH networks to collect a wide range of per-packet and per-hop monitoring information. The mechanism leverages the space remaining in the IEEE 802.15.4e frames and does not affect network behavior and performance for piggybacking telemetry information onto the data packets. Other specifications are

oriented toward defining a monitoring solution that is valid for any Internet network. For example, In-situ Operations, Administration, and Maintenance (IOAM) [17] [18], promoted by several companies, is an implementation that includes specific operational and telemetry information in data packets. IOAM complements other out-of-band OAM mechanisms that use dedicated probe packets to convey OAM information. This mechanism is not oriented toward deployment across the whole Internet, but for supervised domains or subnetworks within a single administration. There are also specifications focused on the definition of generic telemetry frameworks for the implementation of in-band telemetry. This is the case for In-situ Flow Information Telemetry [19], which defines a framework for applying existing and emerging in-band telemetry techniques, such as the IOAM (as mentioned earlier) or INT to enable the collection of performance information from the network. Finally, In-band Network-Wide Telemetry is a framework based on INT [20] that aims to extend the monitoring function to the entire network, rather than a portion of the network. It decouples the network monitoring system into a routing mechanism and a routing path generation policy and uses a path planning policy to generate non-overlapped INT paths that cover the entire network.

2.3.2. SDN, P4, and INT

In recent years, INT has been mainly studied along with topics like P4 and SDN. This is because P4 can leverage INT in programmable data planes, and SDN can play a role either in monitoring or in traffic engineering. In this sense, N.V.Tu et al. [21], and Hyun et al. [22] designed and implemented an INT architecture for the Open Network Operating System (ONOS) SDN controller. The authors also discussed the use cases for and limitations of deploying an ONOS-based INT monitoring system. N.V.Tu et al. later published a paper about a high-performance eXpress Data Path (XDP)-based INT collector (INTCollector). The merged conclusion of the three papers above is that integrating INT management in an SDN network is useful for monitoring traffic within SDN boundaries. However, the increase in CPU and resource cost for processing and monitoring many flows could require sampling the monitored information or offloading some analysis to P4 switches or NICs. The INTCollector tackles the overhead of CPU usage when processing INT report packets. Shie et al. [23] also demonstrated an implementation of the INT protocol for metro networks. Their research aimed to collect per-packet telemetry information to enforce an adaptive QoS for the same packets that experienced different packet-treatment levels in previous network steps. In other words, a packet that experienced high levels of delay from the access nodes receives a better treatment closer to the network core. The authors avoided using an SDN controller, which guaranteed an improved packet treatment in the data plane. At the same time, they pointed out that the analysis of telemetry statistics is useful for further optimizations using the SDN controller.

As a result of the analysis presented, we observe that existing studies have not yet merged P4, SDN, MPLS, and telemetry. We believe ours is the first paper to integrate the INT protocol with next-generation SDN devices and MPLS legacy routers.

3. Discussion about hybrid NGS monitoring

This section first describes the main elements in an INT scenario as well as the available monitoring modes. Then, a discussion is presented concerning the location of the INT header so that traffic with appended telemetry can be appropriately forwarded in an MPLS network. Moreover, other relevant topics, such as the limitations of the telemetry header size, switch placement, and time synchronization, are also discussed in this section.

INT is a framework designed by the P4 consortium to promote the collection and reporting of network monitoring information by the data plane without requiring the intervention of the control plane. INT packets contain header fields that define what state information needs to be

collected and written into the packet as it transits the network. The INT architecture definition is generic, and hence it enables several high-level applications such as network troubleshooting, advanced congestion control, advanced routing, and network data plane verification. With INT, apart from reducing the overhead and going deeper into the details obtained by the telemetry system (e.g., hop-by-hop behavior), the generation of telemetry reports can be triggered by various events, such as flow monitoring, queue congestion, and packet drops.

Switches that implement INT can be categorized into three types (as seen in Figure 2): the INT source, the INT transit, and the INT sink. The INT source inserts the telemetry headers in the proper position, which varies depending on the traffic type (virtualization, TCP/UDP, ESP, etc.). This topic is further developed in the next section. The INT transit is responsible for checking the instruction bitmap and inserting telemetry information along with the metadata stack. Finally, the INT sink clones the packet and performs two tasks. First, the INT sink removes any INT headers from the packet and forwards the packet to the destination, with the same structure as it had when it entered the telemetry network domain. Second, with the cloned packet, the INT sink generates a telemetry report to send to the INT collector, which is generally a server that processes the information and stores the telemetry data. In some cases, the INT sink can do some initial calculations and evaluations in reaction to urgent events. Otherwise, the collector can be used for this purpose as well as for producing statistics based on both real-time and historical data.

3.1. INT monitoring modes

According to the latest INT Dataplane Specification [6], INT switches can monitor packets using three different modes. The specification defines the INT-XD (INT eXport Data), INT-MX (INT eMbed Instruct(X)ions), and INT-MD (INT eMbed Data) application modes. INT-XD refers to the previously defined postcard mode. INT switches export telemetry data when a packet matches on data plane tables (tracking tables). This mode does not require modifying packets that traverse switches, but an additional packet (a report) is generated per switch that exports telemetry data. In INT-MX mode, the INT source switch embeds instructions in the packet, and the switches along the path export the telemetry data following a mode like the one described for INT-XD. The packet includes the instructions in the INT header but does not carry any metadata. Finally, INT-MD mode allows instructions and data to be embedded in the packet that traverses the network (hop-by-hop INT).

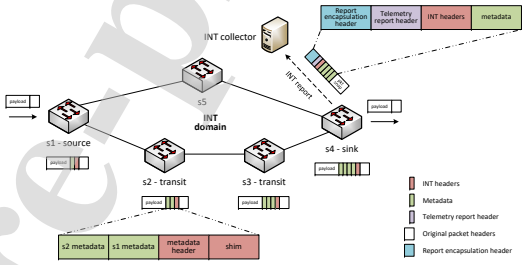


Figure 2: INT source, transit and sink working under INT-MD mode placing telemetry headers between TCP header and TCP payload.

This paper focuses on the INT-MD application mode applied to networks that can process MPLS packets within ISP networks. INT-MD mode requires that the INT headers be embedded in the packet. Our work focuses on INT-MD because it provides some benefits in terms of limiting switch-to-controller and switch-to-collector communication and generates a reduced number of INT reports compared to INT-XD and INT-MX. However, one of the main disadvantages of using INT-MD mode is that the packet size grows as it traverses via INT transit devices, making it possible to surpass the Maximum Transfer Unit (MTU) limit.

This issue can be addressed in the following ways:

- Modify the MTU of the link between the INT source and sinks to a value higher than the MTU of the preceding networks. The INT spec recommends *per-hop metadata length **

*4 * INT hop count + fixed INT header (12 bytes)* to prevent the egress MTU from being exceeded due to INT metadata insertion. Although the most typical MTU value is 1500 bytes, the so-called Jumbo Ethernet frames carry up to 9000 bytes of payload, which reduces the CPU usage and the overhead. Modifying the MTU has some side effects like increasing the delay, which is minimized in fiber-based deployments.

- Set the M bit in the INT header to prevent other hops from adding more INT metadata when the MTU length is reached. This should trigger a backup mechanism in which every hop generates its own telemetry report (similar to postcard mode).

- Involve INT source/transit switches in the path MTU discovery process so that they can report a conservative MTU value. This causes an increment in the overhead, as more packets will be necessary for the same amount of user data.

- INT transit switches can perform IPv4 fragmentation to overcome the egress MTU limitation. However, this should be avoided as it can dramatically downgrade the performance of applications. Moreover, IPv6 packets cannot be fragmented at intermediate hops.

The exact telemetry header location for INT-MD is intentionally not specified by the INT specification [6] authors. The headers can be inserted as payload or as an option for many, if not all, encapsulation types. The following section provides a detailed description of the possible header locations, and the locations suggested in this paper.

3.2. Discussion about INT header location

The INT specification [6] suggests locating INT headers over TCP/UDP, GRE (IPv4 and Ethernet), VXLAN GPE, and Geneve. While the exact location of telemetry headers is not specified, Figure 3 shows the suggested locations for telemetry headers (INT shim, INT metadata header, and the INT metadata stack).

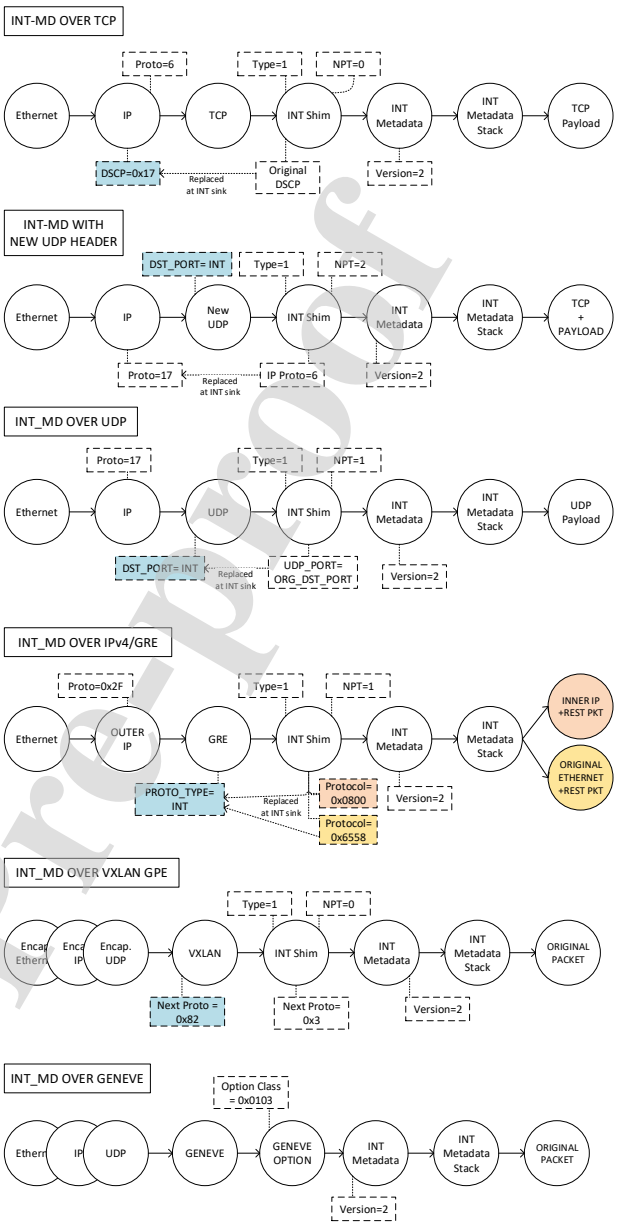


Figure 3: Various possibilities for including INT-MD mode headers in different types of traffic.

The specification suggests that for TCP traffic, the IP header's DSCP value 0x17 can be used to indicate the presence of INT headers after the TCP header. When UDP is the transport protocol of the current packet being monitored, the INT source switch could change the destination port for a well-known port number that signals to transit and sink switches that the current packet has INT headers. Furthermore, the specification

also suggests that a new UDP header can be introduced as an INT marker (using the same port number criteria as for the previous case), even when the original packet has a TCP header (which will follow the new UDP and INT headers).

In terms of overlay protocols, the specification describes the use of INT over IPv4/GRE, VXLAN GPE, and GENEVE. In the case of GRE, INT is placed behind the GRE header and uses its protocol type field to mark INT's presence. INT's shim header keeps the original value of the protocol type, which varies if the following header is IPv4/IPv6 or Ethernet (Transparent Ethernet Bridging). In the case of VXLAN GPE, a new field named "Next Protocol" (compared with VXLAN) is used to hold a value that suggests the presence of INT, which can be found between the VXLAN and the original payload. Similar to the case of GRE, INT shim header holds the original "Next protocol" value. Finally, for the case of GENEVE, the INT headers are placed behind a GENEVE option specifically used for INT.

Whereas the INT specification suggests the INT header location considering different types of traffic, the MPLS networks can vary significantly. The MINT specification draft [15] does not follow the suggestions of the INT specification, and it proposes a different location. In our proposal, we follow both [6] [15] specification suggestions and a new header locations for ESP traffic. Having a unique header placement is not possible in most cases, and therefore, the following suggestions show locations that can overcome most of the limitations.

First, following the original work by Thomsen [24], and MINT [15], the first proposal suggests including telemetry headers between the MPLS and IP headers. The MINT draft proposes the use of two special-purpose labels to indicate the presence of telemetry headers. However, it is also possible to use a 64-bit probe marker that can also indicate the telemetry header presence. While the marker presence can conflict with the subsequent fields or payload, the likelihood of this happening is low [6]. We finally propose the use of an unassigned MPLS label value (e.g., 4–6 values are unused) that can signal the presence of telemetry. In this way, we save 4 bytes compared with the MINT or INT proposals. However, this unassigned label would only be compliant within the local MPLS domain, assuming tests show proper traffic forwarding using legacy routers.

While a location between MPLS and IP has been proven to work [24], the IP and further header displacement could become a problem in some corner cases. While this approach would also accommodate encrypted IPsec traffic, load balancing functionalities that require IP and TCP/UDP header hashing would fail. Such a case, though, could succeed with the use of MPLS entropy labels. Furthermore, the possible overhead for an MPLS SR case requires further MINT label and entropy label addition. This case would also demand an ability to push/pop deeper label stacks. Finally, tunnel decapsulation for Penultimate Hop Popping (PHP) LSR routers would remove MPLS labels but keep the telemetry headers between the egress Ethernet and IP headers, which would disable any IP forwarding for the PHP and subsequent routers.

To this end, our work also suggests and tests the use of INT headers as suggested in the INT specification [6] for IPv4 and TCP/UDP packets. The location between the TCP header and the TCP payload (similarly for UDP) for the INT headers and metadata stack will likely circumvent most of the limitations encountered by locating INT between MPLS and IP. Most legacy routers do not require parsing ahead of TCP and, therefore, having telemetry headers between TCP and the payload would work in most cases. However, this approach will not work when packets are encrypted with a protocol such as IPsec; in that case, INT over MPLS will still work, despite its disadvantages.

3.3. Limitations on telemetry size for INT-MD over TCP/UDP

As INT-MD is the focus of this paper, this section analyzes the limitations of including INT-MD headers on packets traversing the service provider networks. As described in Equation 1, to accommodate all telemetry headers, the packet's max-

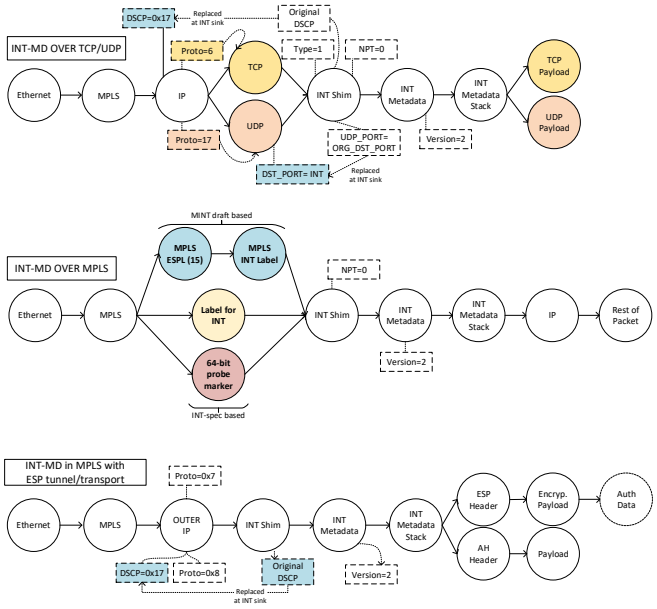


Figure 4: Proposed header location to integrate INT with MPLS networks.

imum size (P) when received by the INT source is highly dependent on the total number of INT hops (H). A packet traversing from INT source to INT sink will first increase its size by 16 bytes due to the INT shim and INT metadata headers, and it will also increase by $4 \cdot (H-1)$ or $8 \cdot (H-1)$ bytes, with H representing the number of hops and T representing 4 or 8 bytes, depending on the instructions and hop metadata length.

$$\begin{aligned} P + INT_S + INT_M + T \cdot (H-1) &\leq MTU \\ P + 4 + 12 + T \cdot (H-1) &\leq MTU \\ P &\leq MTU - 16 - T \cdot (H-1) \end{aligned} \quad (1)$$

where:

MTU = Maximum Transfer Unit
P = Original packet size
INT_S = INT shim header size
INT_M = INT metadata header size
T = Telemetry metadata size per hop
H = Number of INT hops

While the telemetry report is not included in the above equation, it is important to consider the packet size limitations. The maximum number of bytes from the original packet that can be appended to a telemetry report is significantly more limited than an INT-MD packet. This is due to the fact that the INT telemetry report needs new outer Ethernet, IPv4 or IPv6, UDP and Telemetry Report headers. However, it is not always necessary to encapsulate the complete original packet in the report as it can be truncated (as opposed to INT-MD).

3.4. Switch placement

In this paper, a mixed approach (edge and transit) for SDN switch placement is proposed. The architecture consists of P4 switches placed as Label Edge Routers (LERs) or Provider Edge (PE) routers in sites that aggregate a high number of customers. Ideally, a provider should try upgrading as many edge devices as possible to support P4, although attaining such an accomplishment requires gradual deployment. To match the INT domain with the MPLS domain, we consider that a minimum setup requires having at least two P4 LERs at both edges of the network. In this way, the monitoring information can be collected end-to-end and with consideration for having the full MPLS domain. If only one P4 LER is configured, INT monitoring capabilities are reduced to a portion of the MPLS network. All in all, with the use of P4 and INT, a provider can track performance at the edge devices and track edge-to-edge packet-based telemetry.

Furthermore, the architecture expects to have several P4 switches working as core network Label Switch Router (LSR) or Provider (P) router. Particularly at topological positions where LSRs have multiple links and have an influence on multiple path forwarding. The target is to optimize the network's performance by using the minimum number of P4 switches for the Hybrid NGS monitoring system. The topology shown in Figure 5 shows two P4 switches placed as LERs, and another as an LSR, with redundant paths between them. Replacing a legacy router at an appropriate LSR topology location enables SDN controllers to command P4 switches to redirect traffic via one of several possible paths to reach the same endpoint. P4 switch tables can be programmed at runtime to apply forwarding rules after analyzing the timestamps of packets forwarded via multi-

ple paths. This enables new MPLS-TE procedures that take advantage of telemetry to make forwarding decisions. While a gradual replacement of legacy LSRs by P4 switches is required, a minimal configuration only requires two P4 LERs, so setting up at least one P4 LSR is suggested so that advanced monitoring strategies can be implemented (e.g., path verification). In fact, the greater the number LSRs deployed, the greater the granularity will be provided by the monitoring system, which allows for more precise telemetry.

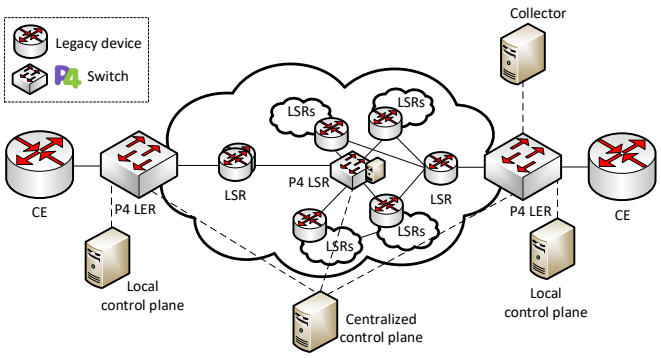


Figure 5: Example of P4 switch integration in a generic MPLS network.

3.5. Timestamping and time synchronization

P4 switches can leverage timestamps to measure the actual end-to-end latency of customer packets without inserting probes. Indeed, probes may not receive the same treatment from programmable and legacy routers (lack of same headers, lack of a particular payload, inability to reproduce the same header values, etc.). Therefore, employing specific user data packets provides a detailed view of the treatment of these packets.

Timestamping requires precise time synchronization, which is generally obtained with the use of Global Positioning System (GPS) modules embedded in commercial P4 devices. Network-dependent alternatives such as Precise Time Protocol (PTP) can also be used, as long as it is understood that they produce a sync error due to their dependency on network behavior (e.g., PTP server placement). However, this error can be considered negligible (if measured in nanoseconds or microseconds) compared with the measured end-to-end latency (which is generally measured in terms of milliseconds).

Finally, it is worth mentioning that in this work, no synchronization mechanism is needed as we use an emulated architecture that allows us to set timestamps using the same physical P4 target.

4. Description of the hybrid NGS monitoring integration

This section first explains the ISP use case to build a hybrid NGS network, and then it describes control and data plane designs for the P4 switches to implement INT in a hybrid SDN network based on MPLS. Topics such as the control plane integration and data plane offloading of various network functions are addressed in the following subsections.

4.1. Internet Service Provider use case

Often, ISPs have a considerable number of MPLS routers that have no telemetry capabilities or that cannot be commanded by an SDN controller using OpenFlow or P4Runtime. The cost of deploying a complete NGS network is too high when the devices in the network involved can be counted in tens or hundreds. Deploying a new type of architecture requires a considerable amount of know-how within the engineering team too. Moreover, the available switches with an Application-Specific Integrated Circuit (ASIC) that can be programmed using P4 support high data rates compared with average legacy enterprise routers. In terms of cost, replacing many routers can result in high expenditure and therefore the inclusion of P4 switches must be planned and gradually deployed for value-added use cases (as analyzed in the previous section).

In our research, the experiments included Cisco MPLS access routers used to connect enterprise and service provider networks. The MPLS implementation of these routers, the one leveraged with the programmable ASIC and a custom control plane demonstrates that the implementation of this study is compatible with devices running in provider networks.

4.2. Control plane design and integration

To integrate P4 switches in MPLS networks, the control plane must include the same applications and process protocols that legacy routers do. The actual application deployment may vary from network to network. Therefore, this section presents an overview of the requirements in terms of control plane applications that must be included to provide a basic MPLS integration. It is worth noting that a more complex MPLS configuration might lead to developing additional features in hybrid NGS control planes.

Generally, MPLS networks run an Interior Gateway Protocol (IGP). Several IGP protocols like Open Shortest Path First (OSPF), Intermediate System to Intermediate System (IS-IS), or Routing Information Protocol Version 2 (RIPv2) are applied as IGP protocols for MPLS networks. The use of IGP protocols is important because they retrofit the label distribution scheme by providing a route for the internal network (among other functions). As MPLS networks differ in their features and functionalities (RSVP-TE implementation, proprietary IGP protocols, EGP support, etc.), the SDN control plane may also need to implement the BGP protocol, especially PE routers. Figure 6, shown below, describes a general view of the necessary protocols and data plane implementations to be included by the P4 data plane and its centralized and/or local control plane.

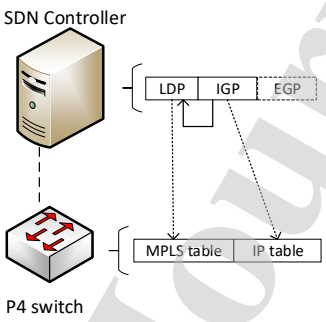


Figure 6: Required minimum control and data plane configuration to enable hybrid SDN and legacy MPLS routers.

In our deployment, the legacy routers are manually configured (i.e., interface modes, IP routing, MPLS rules, etc.). Therefore, the P4 switch control plane only needs to support some of the LDP session messages. The complete message exchange is shown in Figure 7. In our deployment, we incorporate a centralized control plane and a local control plane to partition the applications. The deployment keeps MPLS forwarding, IP forwarding, and INT applications in the centralized control plane. However, the local control plane incorporates the LDP session application (TCP session for Initialization Messages, Keep Alive Messages, Address Message, etc.). Moreover, the deployment also accommodates both the Address Resolution Protocol (ARP) responder and the continuous LDP Hello message process, both locally and offloaded to the data plane. Whether the local control plane or the offloaded version is used for testing is has no repercussion with results in Section 6.

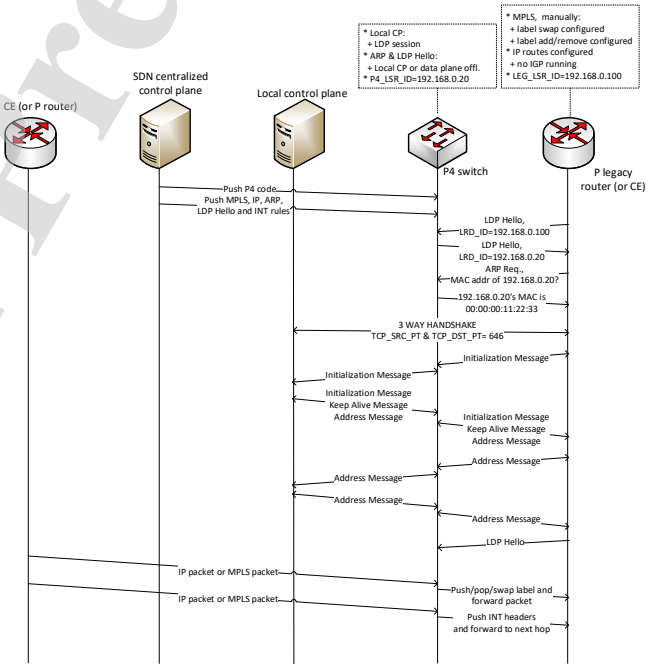


Figure 7: P4 control and data plane message exchange to integrate with legacy routers.

4.3. Data plane design and offloading

The advantage of programming the data plane is that tables can be organized and programmed as required. To follow the design of MPLS networks, developers need to include MPLS and IP

tables to forward traffic according to control plane rules and the traffic types traversing the switch. Additionally, one or more INT tables are also needed to indicate the switches when to push INT headers, add telemetry to the stack, and finally pop headers and generate reports.

Despite the mandatory tables for traffic forwarding and INT implementations, using P4 devices offers the possibility to offload functions to the data plane to implement several network functions. To ease the task of adapting every application needed in the local or centralized control plane, some functions can be offloaded to P4. For instance, the ARP responses and the periodic LDP Hello message responses can be offloaded. Generally, the control plane for P4 programmable devices needs an ARP responder, either as a Proxy-ARP application in the centralized control plane or as a standalone local application. The ARP application is needed before the P4 switches, and legacy routers start an LDP session. When ARP is used as an offloaded app, the centralized control plane pushes a minimum set of rules so that ARP requests are answered by the data plane for the appropriate requested IPs, and the LDP Hellos are sent with the correct LSR IDs. Figure 8 below shows how the apps are deployed in each control plane, and how the ARP and LDP Hellos can be offloaded to the data plane.

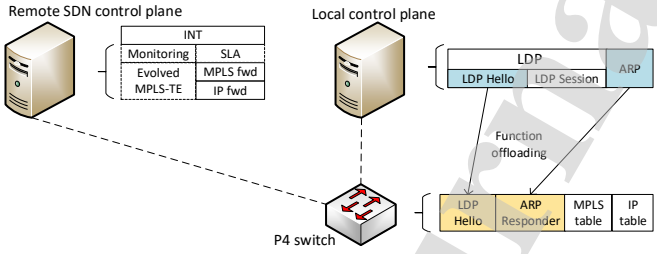


Figure 8: App distribution for centralized and local control plane. Function offloading is shown for ARP and LDP Hello messages.

While the control plane is freed from answering ARP requests or LDP Hello messages, one of the disadvantages of offloading the LDP Hello messages to the data plane is that if the local control plane fails (the one in charge of LDP sessions and distributing MPLS labels in actual deployments), then a legacy router might interpret that the router is still working properly. This could happen because ARP responses and LDP Hello messages continue to be sent even if the control plane fails, as long as rules persist in the tables.

5. Monitoring feature description and verification

This section presents several use cases for hybrid NGS monitoring system operating on top of a legacy MPLS network. The cases presented in this section demand switch placements that require LERs or specific LSRs to be replaced with P4 switches. In most cases, devices that perform as INT sink require to be programmable switches (like a P4 LER) if no other INT compatible device is found in the path (such as a P4 LSR). In any other case, P4 switches can still be used as LERs performing Ultimate Hop Popping (UHP), and legacy routers can be PHP LSRs.

5.1. MPLS label verification

One of the advantages of appending telemetry headers to packets in hybrid NGS networks is that new instructions (in the instruction bitmap) can be assigned to custom functions. In this case, the network will monitor the path labels, providing a toolset for MPLS fault management. Because it is a hybrid NGS network and the network includes P4 devices, the number of labels that can be carried is limited to the links in which P4 devices are present, and MPLS headers are used.

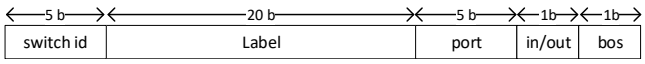


Figure 9: New telemetry header: MPLS label verification metadata that is inserted into the metadata stack.

To accommodate this case, a new telemetry (metadata) header is created. There is no such header in the previous specifications that combines the information included in the label verification header. This new header includes the switch ID, MPLS label, ingress/egress port, in/out bit marker, and a bottom-of-stack bit.

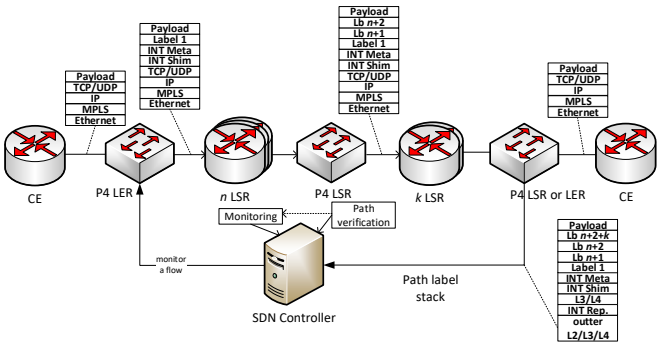


Figure 10: MPLS label verification mode to ensure the configuration matches the expect behavior.

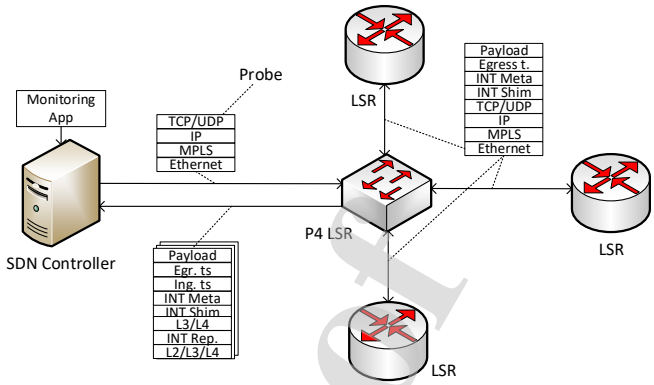


Figure 11: Telemetry mode to test next-hop(s) pipeline performance.

With this header, any P4 switch in the LSP, commanded by the SDN controller, can include in/out MPLS labels.

This use case is convenient when service providers need to verify the path that a packet traverses, and the use of the hybrid SDN network can reveal partial MPLS label information. Along with the ports used to forward the traffic, service providers can compare this information with the expected LSPs.

5.2. Next-hop(s) legacy pipeline monitoring

In this case, active probes and INT headers are combined in the same packet. A P4 LSR with multiple paths and legacy devices as next hops can forward a probe that has purposely been crafted so that a legacy next-hop sends it back to the same P4 LSR with appropriate information in the INT stack. This process involves adding a timestamp to the metadata stack of the probe packet when it is sent to legacy device (egress timestamping) and also when it is received (ingress timestamping). Subtracting egress and ingress timestamps show an approximate pipeline latency (processing time) at the legacy device, ignoring the propagation time. This feature is useful to measure with certain accuracy the pipeline of legacy devices, for instance, to detect congestion in queues.

5.3. Path latency verification

In this scenario, a service agreement that tries to establish forwarding based on paths with lowest latency can be verified. The INT sink switches can monitor the end-to-end time that packets experience and notify the INT collector when any service agreement is broken. The customer Service Level Agreement (SLA) monitoring application can determine that when a particular percentage of packets experience higher latency than the one agreed and determine that an SLA might have been broken. This could be an interesting service for customers who would like to actively monitor service agreements.

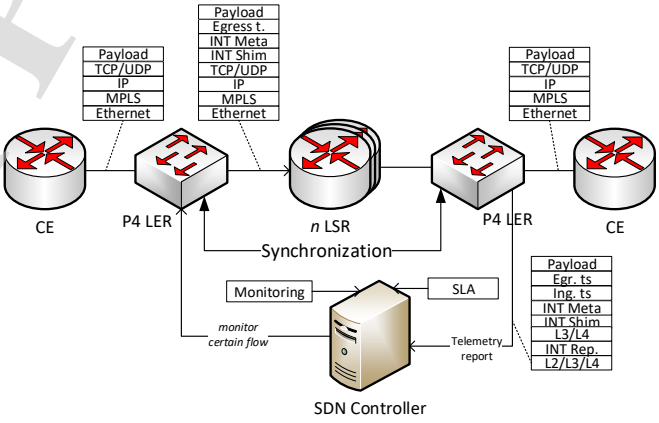


Figure 12: Verification mode to confirm that service agreements are fulfilled.

5.4. Traffic engineering

Telemetry data can be used to make optimized forwarding decisions and implement advanced Traffic Engineering (TE) techniques. As an example, a multi-path traffic monitoring system can help a hypothetical evolved MPLS-TE (eMPLS-TE) mechanism to provide a path with

the lowest possible latency. In this case, a P4 LSR switch between both LERs replicates (a few) packets via multiple paths. At the endpoint, and following the network in Figure 13, the LER discards two out of the three packets but still sends three reports to the INT collector. When several packets have been monitored, the eMPLS-TE will be able to build a short-term end-to-end latency picture and decide, if necessary, to change the P4 LSR's default forwarding path. Additionally, probe packets or packets from different customers could also provide end-to-end latency information in additional paths without replicating packets.

Besides end-to-end latency, measuring network devices' interface utilization allows having a near real-time image of link utilization in the network. P4 LERs and LSRs report the received/transmitted packet rates at every interface, which means that all the interfaces connected to those devices are supervised. The eMPLS-TE mechanism is thus able to compute the available link capacity ($AvailableLinkUsage = TotalLinkCapacity - CurrentLinkUsage$) in order to provision paths according to the required bandwidth stated in the SLA for each customer service.

Finally, buffer and queue occupancy statistics provide an overview of the congestion of the network, which provides feedback to the eMLPS-TE mechanism so it can reroute packets through alternative MPLS paths for congestion control.

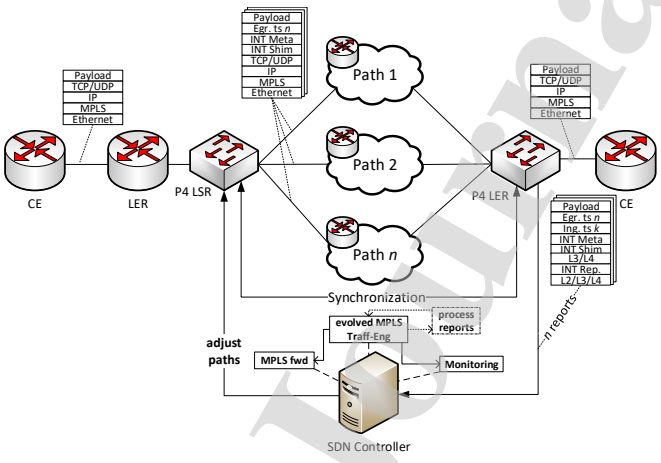


Figure 13: Multi-path latency measurement to retrofit an eMPLS-TE mechanism to route via the path with lowest latency.

6. Monitoring validation and discussion

In this section, several tests are performed to validate INT-MD integration in MPLS traffic and test the novel monitoring features described in the previous section.

6.1. Test environment

The HPE ProLiant DL380 Gen9 Server was used to emulate the different scenarios. The server includes an Intel Xeon CPU E5-2620 v4 running at 2.10GHz and 4x16GB Single Rank x4 DDR4-2400 registered memory kits. The physical P4 NIC was deployed using a Netronome P4 SmartNIC ISA-4000-10-2-2 (PB Agilio CX 2x10GbE). The emulator used in the tests was GNS3 v2.2.10, and the legacy routers were emulated in this environment. The emulated routers were the Cisco 3725 that run Cisco IOS 3700 Software (C3725-ADVENTERPRISEK9-M), Version 12.4(15)T14. The tests in Sections 6.2 and 6.3 make use of the P4 SmartNIC and the server. The tests in Section 6.4 use the HPE server providing the emulation of the network and the legacy routers and integrate the Edgecore Wedge 100BF-32X programmable switch. The Cisco switches (running on the GNS3 emulator) and the programmable switch are connected using an Intel XL710-BM1 Quad-Port 10G SFP+ Ethernet network interface card. Our testing scenarios and developed code used in this paper have been shared in a public repository [25] containing the following main contributions:

- The Programmer Studio control plane and the $P4_{14}$ source code for the Netronome SmartNIC.
- The Python Barefoot Runtime (BFRT) rules and the $P4_{16}$ source code for the Edgecore Wedge switch.
- The LDP control plane and ARP control plane (when not offloaded to the data plane) for both targets.
- The source code and commands to test each of the cases presented in Section 6.4.1.

Localization	Forwarding	Possible marker 1 size	Possible marker 2 size	5-tuple LB support	PHP legacy supp.	UHP legacy supp.
Between MPLS and IP	Yes	64 bits (2 x MPLS hds)	32bits (1 MPLS hd)	No*	No	No
Between TCP/UDP and Payload	Yes	0 bits (DSCP/UDP DST PORT)	64-bit (new UDP or probe)	Yes	Yes	Yes
Between IP and ESP	Yes	0 bits (DSCP/UDP DST PORT)	64-bit (new UDP†or probe)	N/A‡	Yes	Yes

* While 5-tuple load balancing is not supported, MPLS entropy labels can be used to circumvent this problem.
† An specification draft proposes to include UDP header for ESP traffic in order to enable load balancing. This would not count as extra size.
‡ Would never support 5-tuple load balancing but current routers support defining hash fields to use.

Table 1: Comparison of major features for each header placement that were verified to allow forwarding using legacy MPLS routers

Figure 14 shows how the emulated legacy routers communicate with the physical P4 switch. The emulation environment offers the possibility to connect the emulated legacy routers (Cisco 3725) with the physical server's network interfaces, offering a bridge between the host computer's virtual/physical interfaces and the emulator. Some of the network interfaces on the host server were exposed by the P4 SmartNIC's virtual interfaces, which also offers two physical ports. In this way, the SmartNIC offers the possibility to emulate up to 32 interfaces (plus the two physical ones), as if it were a P4 switch.

The centralized control plane is managed via the SmartNIC Programming Studio application, while the local control plane uses one of the virtual interfaces. Because the same switch is used to represent different switches in the topology, each local control plane application (as many P4 switches are added to the emulator) must be attached to a different virtual interface. In Figure 14, the ARP and LDP Hello messages are located in the local control plane to show the difference compared with Figure 7, which implement offloaded and local control plane apps, respectively. As mentioned, the location of the applications is inconsequential for testing purposes but offers new possibilities for offloading further applications to the data plane.

6.2. *Monitoring system validation and telemetry header placement verification*

To demonstrate that INT-MD can be used in provider networks, the proposed header locations need to be tested to check if the traffic can be forwarded as expected. This test's main goal is to prove the feasibility of INT-MD for the provider's traffic and test if the legacy routers can forward

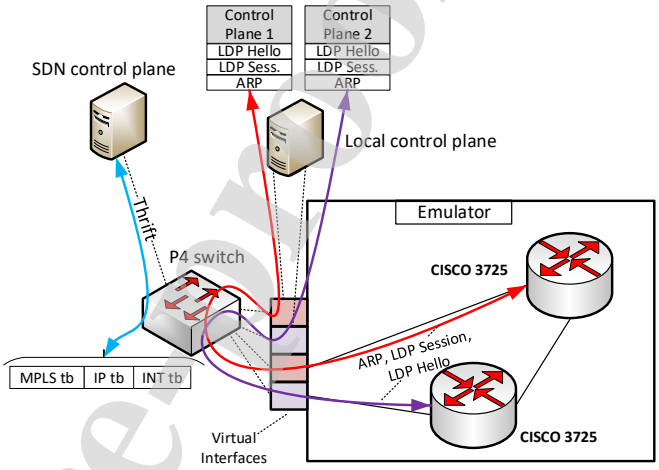


Figure 14: Message flow between legacy routers and the physical switch and control planes.

the traffic.

The linear topology used for the tests is shown in Figure 15. The devices in this topology represent an LSP that comprises P4 and legacy routers. The LERs and the central LSR are P4 switches, as suggested in Section 3.4.

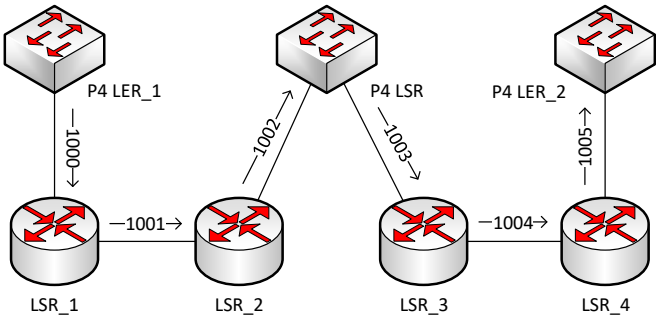


Figure 15: Linear testing topology.

The experiments confirmed that telemetry headers can be placed and forwarded using any of the proposed header locations: between MPLS and IP, between TCP and TCP Payload, and

between IP and ESP. In every test, the legacy routers were able to swap labels and forward the packets to the next hop. We also tested whether the P4 LER switch could push MPLS labels and INT-MD headers and metadata too. The following P4 switches also inserted the necessary telemetry data. Figure 16 shows the bytes of the packet content at the sink P4 LER switch (LER_2) before being encapsulated as a report.

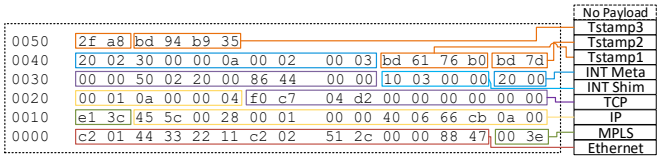


Figure 16: INT-MD and ingress timestamps over IPv4/TCP Wireshark packet capture at P4 LER_2 before encapsulation into an INT report.

The tests demonstrate that IP and TCP headers are not necessary for legacy routers to perform MPLS-based forwarding. However, we found that IP and TCP/UDP fields can be used for different purposes in current provider MPLS networks. Therefore, table 1 summarizes the supported and unsupported features that each placement provides, and that may lead a provider to choose one placement or another.

As a general consideration, the placement that is most beneficial for provider traffic is INT-MD over TCP/UDP and payload. This is because placing the header between MPLS and IP prevents providers from using load balancing techniques that require IP and TCP/UDP fields. Still, using entropy labels can solve the problem presented for INT-MD over MPLS. However, this requires an additional MPLS header. Added to the possible pair of MPLS headers used as markers, forwarding devices (legacy or programmable) require the ability to parse four MPLS headers, which may be close to the maximum number of MPLS headers that P4 switches and legacy routers can parse. While it has been tested to work with a P4 LER and legacy LSRs, using INT-MD over MPLS is discouraged because this method does not comply with how legacy PHP or UHP routers operate. When MPLS headers are popped, INT-MD headers remain between the Ethernet and IP, preventing IPv4/IPv6 forwarding by legacy LERs or PHP LSRs.

To address the case of having encrypted packets, the use of telemetry headers between IP and ESP headers, as shown in Figure 4 is recommended (INT-MD over TCP/UDP does apply). In this case, placing telemetry headers after IP would likely be safe as intermediate devices would probably interpret INT headers and the metadata stack to be part of ESP. If any intermediate system needs to process the SPI and the sequence number, then placing telemetry headers after the first 64 bits of ESP would be a solution.

6.3. MPLS label verification

After the header placement is decided, there are several use cases to test due to the practicality of using INT-MD-based telemetry headers in the traffic of provider networks. As explained in the previous section, this study created a new INT monitoring header for MPLS networks that was not proposed in other studies so far. Since a packet can carry all the MPLS labels (traversing in and out of a P4 device) in the same packet, it is beneficial for tracking the labels used in hybrid NGS and MPLS networks. The P4 code shown in Listing 1 adds the nth label verification telemetry header. It is important to mention that this use case is different from the ones mentioned in the INT standard, and unassigned instruction bits must be used to accommodate this use case (e.g., using reserved bits from the instruction bitmap).

```
action add_out_label(my_id){
    add_header(int_label_ver[n]);
    modify_field(int_label_ver[n].id, my_id);
    modify_field(int_label_ver[n].label, mpls.label);
    modify_field(int_label_ver[n].port, 0);
    modify_field(int_label_ver[n].in_out, 1);
    modify_field(int_label_ver[n].bos, 1);
    subtract(int_label_ver[n].port,
     standardmetadata.egress_port, 0x300);
    modify_field(int_label_ver[n-1].bos, 0);
}
```

Listing 1: $P4_{14}$ code to add the nth label verification header to the stack.

The traffic capture in Figure 17 shows the correspondence of headers and bytes of the packet. In our case, the switches in the path were able to capture four labels in total. The use of a P4 LSR helps in this case because using more links increases the probability of monitoring two links.

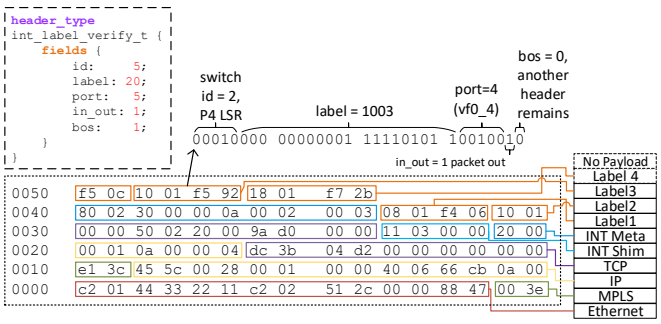


Figure 17: INT-MD and MPLS label verification headers at P4 LER_2

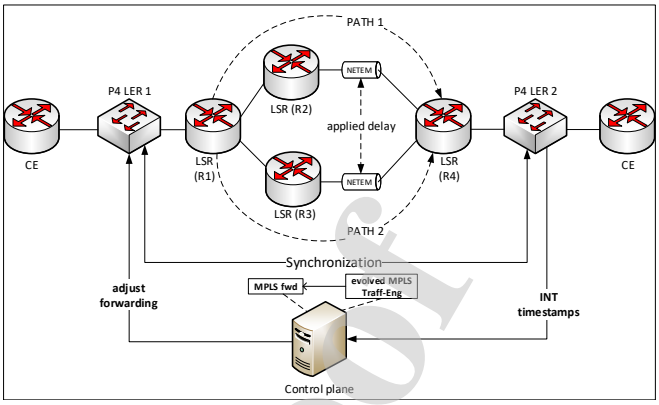


Figure 18: Double-path logical topology with P4 switches as LERs to test end-to-end timestamps

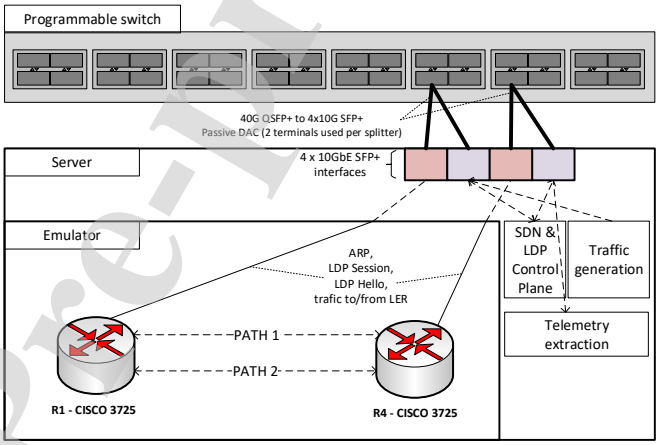


Figure 19: Physical connection of the emulator and the P4 switch

Moreover, the P4 code of the MPLS label verification header named "int_label_verify_t" is explained. Each field is assigned to the proper bits of one of the "int_label_verify" headers. The current header refers to P4 LSR (switch 2), which added the output label 1003 and the port.

6.4. Path latency and traffic engineering

In Section 5.3 and 5.4, we described how the telemetry can be used to account for end-to-end path latency and help apply traffic engineering rules. Our test scenario deployed a single Edgecore Wedge 100BF-32X programmable switch along with the GNS3 emulator on the server. The switch and the emulator were connected using 4x10 GbE SFP+ interfaces. These interfaces served as a passthrough between the Cisco routers and the physical switch. As seen in Figure 18, the deployed switch works as the first and last hop MPLS router (LER), inserting and deleting MPLS labels (UHP) and telemetry headers and values. Figure 19 represents the physical connection of the different components for building the logical topology seen in Figure 18.

In this test case, a single programmable switch is used. Therefore, it is unnecessary to use any synchronization protocol. The global time counter used on the same switch is sufficient to record timestamps and calculate the end-to-end latency. Because the counter rolls over after a determined time, the first switch's timestamp value needs to be lower than the last switch's timestamp value (else, the rollover needs to be taken into account when processing timestamps). If several switches are deployed to measure the same behavior, a supported synchronization protocol needs to be deployed (e.g., PTP) or other supported standards such as the GPS. Depending on the network conditions and the chosen synchronization, the expected accuracy may vary, which can complicate precise latency measurements. Still, achieving millisecond precision is plausible, and the tests shown in this section will adjust the forwarding path when the time difference between paths reaches tenths of milliseconds. It is still possible to forward traffic using different paths if microsecond precision is achieved. Still, decisions that trigger such network forwarding changes should first measure a considerable number of packets suffering from increased latency and still expect variable latency values that accommodate the jitter.

Figure 20 shows two flows being tested in the MPLS network. The first flow (Flow 1, blue) flows via *PATH 1* as stated in Figure 18. The second flow (Flow 2, red) is forwarded via *PATH 2*. PATH 1 experiences an additional 40 ms of latency (+5 ms jitter) compared with PATH 2 from the beginning of the tests. The additional latency is enforced at each path using the NetEm utility at GNS3. From the beginning until marker ①, both flows traverse their assigned paths.

When the control plane processes the latency of both flow paths and the traffic engineering application is activated, it decides to change the MPLS labels at the first P4 LER, changing Flow 1's packets to PATH 2. From marker ① to marker ②, both flows experience a similar end-to-end latency. At marker, ②, the latency along PATH 2 is purposely modified so that both flows now experience an additional 55 ms (+2 ms jitter) of latency.

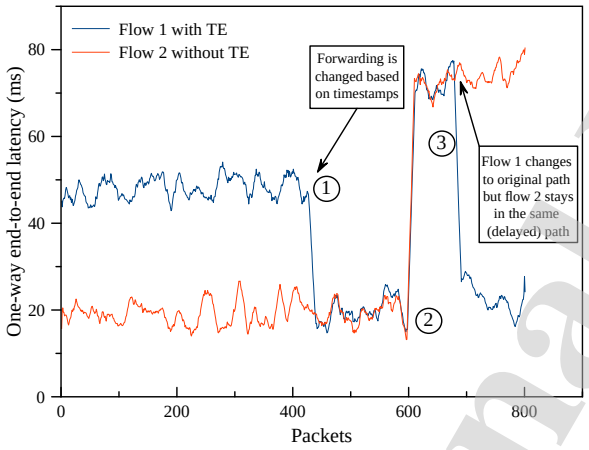


Figure 20: End-to-end latency results for two different flows

Between marker ② and marker ③, the initial latency imposed on PATH 1 is reduced to 5 ms (+1 ms of jitter), and because Flow 1's traffic is adjustable, the first P4 LER changes the MPLS tags and uses the original path, considerably reducing the experienced end-to-end latency. Although we had not decided when to switch paths, it is important to note that triggering a change on the network, such as a path change, should modify the state of as few devices as possible and should not be done frequently (i.e., paths should not change every few seconds but only when a second path is expected to offer better resources for a fair amount of time).

6.4.1. MPLS and INT effect on a programmable pipeline

This section compares different cases that do and do not perform INT processing to demonstrate the overhead of performing telemetry tasks on a programmable switch. In our tests, the P4 switch functions as an L3 router (Case A), as a regular MPLS LER (Case B), an MPLS LER and INT source (Case C), as an MPLS LSR and INT transit (Case D), and finally as an MPLS LER and INT sink (Case E). In all cases (see Table 2 and Table 3), the packets included a 1000 byte payload, and the time spent on the pipeline was measured as the difference between the egress MAC timestamp and the ingress MAC timestamp. For each case, the packet sent to the switch was custom-crafted with the appropriate headers. For instance, when the switch is tested as an MPLS LSR and INT transit (Case D), the packet already includes the appropriate MPLS and INT headers when it is parsed and processed by the switch.

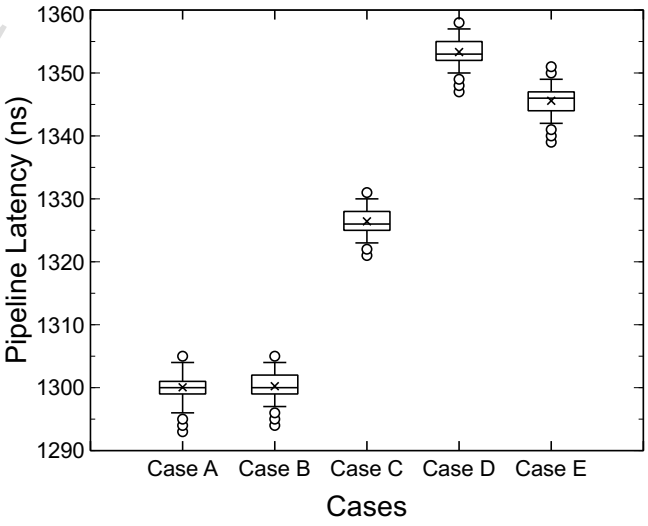


Figure 21: Pipeline latency for cases A to E

As shown in Figure 21, the switch performs best when the fewest tables are matched and actions

Cases	Role	Description
Case A	L3 router	This is the reference case in which a packet is fully parsed (Ethernet/IPv4/UDP or TCP) and L3 forwarding is done.
Case B	MPLS LER	Extending Case A, this data plane performs as an MPLS LER, adds the MPLS header and MPLS fields. Forwards the traffic based on the MPLS label.
Case C	MPLS LER + INT source	Same case as Case B but the device performs as an INT source (adds INT shim and metadata headers and also the INT metadata stack) to the packet.
Case D	MPLS LSR + INT transit	In this case, the switch parses as many headers as a packet could include (Case A packet + MPLS + all INT headers). Adds telemetry to the INT metadata stack and exchanges the MPLS label.
Case E	MPLS LER + INT sink	This final case represents an INT sink that removes all the INT headers and performs L3 forwarding.

Table 2: Description of each use case, including the major actions performed.

	Ethernet				MPLS					IPv4					UDP				INT shim				INT metadata				INT metadata stack			
Cases	*P*	*A*	*R*	D	*P*	*A*	*F*	*R*	D	*P*	*A*	*F*	*R*	D	*P*	*A*	*R*	D	*P*	*A*	*R*	D	*P*	*A*	*R*	D	*P*	*A*	*R*	D
Case A	In-Eg			In-Eg						In-Eg				In-Eg	In-Eg			In-Eg												
Case B	In-Eg			In-Eg	Eg				In-Eg	In-Eg				In-Eg	In-Eg			In-Eg												
Case C	In-Eg			In-Eg	Eg				In-Eg	In-Eg				In-Eg	In-Eg			In-Eg	Eg			In-Eg	Eg			In-Eg				In-Eg
Case D	In-Eg			In-Eg	In-Eg				In-Eg	In-Eg				In-Eg	In-Eg			In-Eg	In-Eg			In-Eg	In-Eg			In-Eg	In-Eg			In-Eg
Case E	In-Eg			In-Eg						In-Eg				In-Eg	In-Eg			In-Eg	Eg				In-Eg				In-Eg			

* A green colored cell indicates that a particular action (*P*, *D*, *A*, *R*, *F* was perfomed for a particular case. *P* indicates that the header is parsed, *D* indicates that the header is deparsed, *A* implies that a header has been added (not parsed but deparsed), *R* means that a header has been removed (parsed but not deparsed) and *F* indicates that the header has been used to determine forwarding. Having the PSA architecture as a reference (see Figure 1b), *In* (for Ingress) and/or *Eg* (for Egress) at *P* and *D* column boxes implies that parsing or deparsing has been done in one or both pipelines.
* MPLS header removal was perfomed at a PHP router not the at the INT sink.
* Case E does not generate the INT report and is not included in this table.

Table 3: Comparison of header actions performed in each case

executed and but primarily when the fewest headers have to be parsed. Case A and B perform similarly, and although Case B adds an MPLS header, the time needed by the switch to execute it is indistinguishable from L3 forwarding (Case A). Cases C to E show significantly different results. When INT-related actions are performed, the ingress parser and egress parser need to parse INT headers, and the time spent on those tasks is higher than cases A or B. For instance, the median time spent on Case C was 25 ns higher than Case A or B. While Case C results were expected, the number of data plane actions performed by Case D is lower than those of Case C, although a similar pipeline latency was expected. However, the results show that an MPLS LSR and INT transit pipeline latency, Case D, is higher than Case C. The reason for this is because more headers must be parsed and deparsed at both ingress and egress compared with Case C. Similarly, Case E would probably require more time to execute all tasks, as more tables have to be applied, and more actions have to be executed. However, the results show a 10ns lower median pipeline latency than Case D. Once again, the explanation points out to the parser because fewer headers have to be deparsed at egress, taking less time to output the packet.

7. Conclusion

This work has proved the feasibility of deploying a hybrid NGS monitoring system over a legacy MPLS network and that deploying P4 devices opens the door to a set of new monitoring features. The ability of P4 to program packet processing pipelines enables the implementation of MPLS label and path latency verification mechanisms, which are both useful to check the compliance of SLAs. Moreover, this paper also presents the applicability of promising TE mechanisms to improve traffic flows Quality of Service (QoS). The simulation tests demonstrated the possibility to use telemetry information (e.g., end-to-end latency) to feed-back traffic engineering algorithms and change the forwarding rules at runtime. The data plane code developed for tests shows that no significant delay is imposed on packets with INT metadata appended to the stack.

While this research work achieved promising results for integrating P4 programmable switches and legacy MPLS routers, the effort required for implementation was demonstrated to be significantly higher than expected. While open-source centralized control planes are available, an attempt to build a fully functional MPLS-compatible control plane requires a significant amount of development. Implementing a fully

functional LDP application and a full-featured IGP can indeed be a burdensome task that may prevent service providers from integrating an SDN network using programmable devices along with their networks. While the deployment effort is high, the advantages of incorporating a hybrid NGS network are numerous. Incorporating an in-band telemetry protocol like INT to monitor service providers' MPLS traffic can provide insight into the provider's network and feed-back applications that enhance traffic forwarding procedures. INT header integration and MPLS label verification were deployed and tested to work with P4 switches and legacy MPLS routers. Finally, the use cases that leverage end-to-end latency tests (e.g., SLAs, eMPLS-TE, etc.) should be tested using time synchronization mechanisms in future work to corroborate our tests in this study.

Acknowledgements

The content of this paper is an extension of the case proposed in the MSc thesis titled *Monitoring and Fault Localisation in Provider Networks* by Signe Erdmann Thomsen.

This work is supported by the GN4 - phase 2 project under GÉANT 2020 FPA and co-funded by the European Unions Horizon program under Grant Agreement No. 731122 (GN4-2). Additionally, this work was supported in part by the Spanish Ministry of Economy, Industry and Competitiveness through the State Secretariat for Research, Development and Innovation under the "Towards zero touch network and services for beyond 5G (TRUE5G)" PID2019-108713RB-C54 project and also in part by the Department of Economic Development and Competitiveness of the Basque Government through the "5G for Basque RIS3" (5G4BRIS3) KK-2020/00031 research project. Finally, this research was also supported by the Spanish Ministry of Science, Innovation and Universities within the project TEC2017-87061-C3-2-R (CIENCIA/AEI/FEDER, UE).

We would also like to thank the team of *MPLS Inband Network Telemetry* draft specification [15] authors and especially Jai Kumar for answering our emails and exposing the disadvantages related to telemetry header location.

References

[1] E. Koeze, N. Popper, The Virus Changed the Way We Internet, The New York Times (Apr 2020). URL https://www.nytimes.com/interactive/2020/04/07/technology/coronavirus-internet-use.html

[2] A. U. Rehman, R. L. Aguiar, J. P. Barraca, Network functions virtualization: The long road to commercial deployments, IEEE Access 7 (2019) 60439–60464. doi:10.1109/ACCESS.2019.2915195.

[3] E. van Eyk, J. Grohmann, S. Eismann, A. Bauer, L. Versluis, L. Toader, N. Schmitt, N. Herbst, C. L. Abad, A. Iosup, The spec-rg reference architecture for faas: From microservices and containers to serverless platforms, IEEE Internet Computing 23 (6) (2019) 7–18. doi:10.1109/MIC.2019.2952061.

[4] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, D. Walker, P4: Programming protocol-independent packet processors, SIGCOMM Comput. Commun. Rev. 44 (3) (2014) 87–95. doi:10.1145/2656877.2656890. URL https://doi.org/10.1145/2656877.2656890

[5] T. P. A. W. Group, Specification documents for the P4Runtime control-plane API, https://github.com/p4lang/p4runtime (2020).

[6] The P4.org Applications Working Group. Contributions from Alibaba, Arista, CableLabs, Cisco Systems, Dell, Intel, Marvell, Netronome, VMware., In-band Network Telemetry (INT) Dataplane Specification, version 2.1 (6 2020).

[7] A. Bahnasse, F. E. Louhab, H. A. Oulahyane], M. Talea, A. Bakali, Novel sdn architecture for smart mpls traffic engineering-diffserv aware management, Future Generation Computer Systems 87 (2018) 115 – 126. doi:https://doi.org/10.1016/j.future.2018.04.066. URL http://www.sciencedirect.com/science/article/pii/S0167739X17323725

[8] A. Bahnasse, M. Talea, A. Badri, F. E. Louhab, S. Laafar, Smart hybrid SDN approach for MPLS VPN management on digital environment, Telecommunication Systems 73 (2) (2019) 155–169. doi:10.1007/s11235-019-00603-6. URL https://doi.org/10.1007/s11235-019-00603-6

[9] M. M. Tajiki, B. Akbari, N. Mokari, L. Chiaraviglio, SDN-based resource allocation in MPLS networks: A hybrid approach, Concurrency and Computation: Practice and Experience 31 (8) (2019) e4728, e4728 cpe.4728. doi:10.1002/cpe.4728.

URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4728

[10] W. Feng, Z.-L. Zhang, C. Liu, J. Chen, Clé: Enhancing security with programmable dataplane enabled hybrid SDN, in: Proceedings of the 15th International Conference on Emerging Networking EXperiments and Technologies, CoNEXT '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 76–77. doi:10.1145/3360468.3368185.
URL https://doi.org/10.1145/3360468.3368185

[11] I. Martinez-Yelmo, J. Alvarez-Horcajo, M. Briso-Montiano, D. Lopez-Pajares, E. Rojas, ARP-P4: deep analysis of a hybrid SDN ARP-path/P4Runtime switch, Telecommunication Systems 72 (4) (2019) 555–565. doi:10.1007/s11235-019-00588-2.
URL https://doi.org/10.1007/s11235-019-00588-2

[12] A. A. Ghaith, S. C. Tan, R. Kaspin, N. Omar, A. T. Samsudin, Hybrid software-defined network monitoring, in: R. Montella, A. Ciaramella, G. Fortino, A. Guerrieri, A. Liotta (Eds.), Internet and Distributed Computing Systems - 12th International Conference, IDCS 2019, Naples, Italy, October 10-12, 2019, Proceedings, Vol. 11874 of Lecture Notes in Computer Science, Springer, 2019, pp. 234–247. doi:10.1007/978-3-030-34914-1_23.
URL https://doi.org/10.1007/978-3-030-34914-1_23

[13] Z. Su, T. Wang, Y. Xia, M. Hamdi, CeMon: A cost-effective flow monitoring system in software defined networks, Computer Networks 92 (2015) 101–115. doi:https://doi.org/10.1016/j.comnet.2015.09.018.
URL https://www.sciencedirect.com/science/article/pii/S1389128615003291

[14] J. Suárez-Varela, P. Barlet-Ros, Flow monitoring in software-defined networks: Finding the accuracy/performance tradeoffs, Computer Networks 135 (2018) 289–301. doi:https://doi.org/10.1016/j.comnet.2018.02.020.
URL https://www.sciencedirect.com/science/article/pii/S1389128618300872

[15] J. Kumar, J. Lemon, Y. Peleg, K. Kompella, MPLS Inband Network Telemetry, Internet-Draft draft-kumar-mpls-mint-00, Internet Engineering Task Force, work in Progress (Mar. 2019).
URL https://datatracker.ietf.org/doc/html/draft-kumar-mpls-mint-00

[16] A. Karaagac, J. Hoebeke, In-band Network Telemetry for 6TiSCH Networks, Internet-Draft draft-karaagac-6tisch-int-00, Internet Engineering Task Force, work in Progress (Jan. 2020).
URL https://datatracker.ietf.org/doc/html/draft-karaagac-6tisch-int-00

[17] F. Brockners, S. Bhandari, C. Pignataro, H. Gredler, J. Leddy, S. Youell, T. Mizrahi, D. Mozes, P. Lapukhov, remy@barefootnetworks.com, daniel.bernier@bell.ca, J. Lemon, Data Fields for In-situ OAM, Internet-Draft draft-ietf-ippm-ioam-data-09, Internet Engineering Task Force, work in Progress (Mar. 2020).
URL https://datatracker.ietf.org/doc/html/draft-ietf-ippm-ioam-data-09

[18] H. Song, B. Gafni, T. Zhou, Z. Li, F. Brockners, S. Bhandari, R. Sivakolundu, T. Mizrahi, In-situ OAM Direct Exporting, Internet-Draft draft-ietf-ippm-ioam-direct-export-00, Internet Engineering Task Force, work in Progress (Feb. 2020).
URL https://datatracker.ietf.org/doc/html/draft-ietf-ippm-ioam-direct-export-00

[19] H. Song, F. Qin, H. Chen, J. Jin, J. Shin, In-situ Flow Information Telemetry, Internet-Draft draft-song-opsawg-ifit-framework-12, Internet Engineering Task Force, work in Progress (Apr. 2020).
URL https://datatracker.ietf.org/doc/html/draft-song-opsawg-ifit-framework-12

[20] T. Pan, M. Gao, E. Song, Z. Bian, X. Lin, In-band Network-Wide Telemetry, Internet-Draft draft-tian-bupt-inwt-mechanism-policy-00, Internet Engineering Task Force, work in Progress (Apr. 2020).
URL https://datatracker.ietf.org/doc/html/draft-tian-bupt-inwt-mechanism-policy-00

[21] N. Van Tu, J. Hyun, J. W. Hong, Towards onos-based sdn monitoring using in-band network telemetry, in: 2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS), 2017, pp. 76–81.

[22] J. Hyun, N. Van Tu, J. W. Hong, Towards knowledge-defined networking using in-band network telemetry, in: NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, 2018, pp. 1–7.

[23] F. Cugini, P. Gunning, F. Paolucci, P. Castoldi, A. Lord, P4 in-band telemetry (int) for latency-aware vnf in metro networks, in: Optical Fiber Communication Conference (OFC) 2019, Optical Society of America, 2019, p. M3Z.6. doi:10.1364/OFC.2019.M3Z.6.
URL http://www.osapublishing.org/abstract.cfm?URI=OFC-2019-M3Z.6

[24] S. E. Thomsen, Monitoring and fault localisation in provider networks, Master's thesis, MSc Telecommunication (2019).

[25] E. Ollora Zaballa, D. Franco, ederollora/Hybrid_SDN_MPLS: v0.1 (Dec. 2020).
URL https://github.com/ederollora/Hybrid_SDN_MPLS

Eder Ollora Zaballa

Eder Ollora Zaballa is a third year PhD student in the Network Technologies and Service Platforms group at the Technical University of Denmark. He studied the Telecommunications Engineering BSc at the University of the Basque Country. He has previously being involved in the European project NGPaaS. His research interest focuses on control and data plane programming (ONOS, Openflow, P4 and P4Runtime), distributed control planes, slicing, telemetry and network security within SDN networks.

David Franco

David Franco received his MSc. degree in Telecommunication Engineering from the University of the Basque Country (UPV/EHU) in 2018. He joined the Communications Engineering Department of the UPV/EHU as a researcher in the I2T (Engineering and Research on Telematics) research lab. His research is focused on communication systems for railway signalling and Software Defined Networking (SDN). He is a current PhD student in Telecommunication Engineering at UPV/EHU.

Signe Erdman Thomsen

Signe Erdman Thomsen is an MSc Telecommunication engineer graduated from the Technical University of Denmark. She pursued her Master's thesis at the Network Technologies and Service Platforms group along with Henrik Wessing, Eder Ollora Zaballa and Telia Denmark. She finished her MSc degree with the highest mark, which relates to Monitoring and fault localization in provider networks. She currently is an employee of the Telia Denmark.

Marivi Higuero

Marivi Higuero obtained her BS and MS degrees in Electrical Engineering, in 1992, in the University of the Basque Country (UPV/EHU), and Ph.D. degree in the same University in 2005. She worked in Sarenet, an Internet Service Provider, as a member of the technical department in this company. Nowadays she works as assistant professor in the Communications Engineering Department in the University of the Basque Country. She is also member of the I2T research lab in the same University. Her research interests include computer networks and services, sensor environments, mobility and Security.

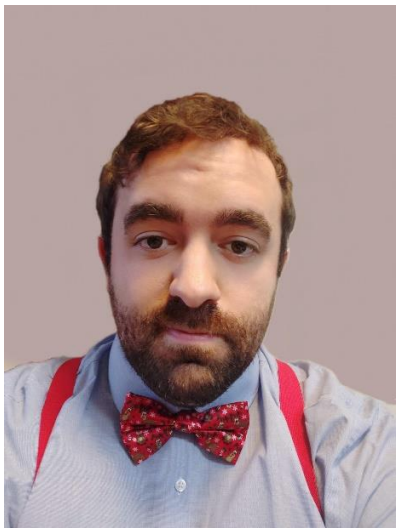
Henrik Wessing

Henrik Wessing completed his PhD study on electronic control of optical infrastructures at the Technical University of Denmark in 2006. While pursuing the Ph.D. study, Henrik Wessing participated in numerous European and national research projects. Since 2013 he has been involved in the European Infrastructure GÉANT projects (GN3, GN3plus, GN4-1, GN4-2) as taskmember or task leader focusing on network performance from network technologies to service monitoring. In addition he is currently Head of Studies for B.Sc. Network Technology and it at DTU, in which position the gap between the students and the network performance monitoring research can be minimised.

Michael S. Berger

Michael Stübert Berger was born in 1972 and received the M.Sc. EE and Ph.D. from the Technical University of Denmark in 1998 and 2004. He is currently Associate Professor at the university within the area of switching and network node design. Michael Berger has been participating in several projects with relation to TSN. He was project leader on the national project ERAN (Ethernet for RAN) where TSN was explored in the Front haul Network. Furthermore, he coordinated the participation from DTU in a Eurostars Project on TSN (Fronthaul for CRAN). He is currently responsible for the Department participation in a Nordic University HUB projet on Industrial IoT, Fog computing and TSN and he is currently Mentor of 1 Post Doc and 2 PhD students in the area of TSN and deterministic networks

Eder Ollora Zaballa



David Franco

Signe Erdman Thomsen

Marivi Higuero

Henrik Wessing

Michael S. Berger

Eder Ollora Zaballa: Conceptualization, Methodology, Writing- Original draft preparation, Writing - Review & Editing, Investigation Software. **David Franco**: Conceptualization, Methodology, Writing- Original draft preparation, Writing - Review & Editing, Investigation. **Signe Erdman Thomsen**: Conceptualization, Methodology. **Marivi Higuero**: Supervision. **Henrik Wessing**: Supervision, Validation. **Michael S. Berger**: Supervision.

Conflict of interest

There is no conflicting interest with the funding projects for this paper although we decide to show the supporting projects but we also do it in the paper:

- This work is supported by the GN4 - phase 2 project under GÉANT 2020 FPA and co-funded by the European Unions Horizon program under Grant Agreement No. 731122 (GN4-2).
- Additionally, this research has also been supported in part by the 5GCity Project funded by the Spanish Ministry of Economy and Competitiveness under Grant TEC2016-76795-C6-5R and in part by the Basque Government through the Elkartek Program (ref: KK-2020/00031).