Visualize Your IP-over-Optical Network in Realtime: a P4-based Flexible Multilayer In-band Network Telemetry (ML-INT) System

Bin Niu, Jiawei Kong, Shaofei Tang, Yingcong Li, Zuqing Zhu, *Senior Member, IEEE*

***Abstract*—Nowadays, with the fast development of backbone networks, the performance monitoring and troubleshooting on IP-over-optical networks have become increasingly important. However, a real-time monitoring scheme, which is programmable to reveal the end-to-end multilayer information of an arbitrary flow in an IP-over-optical network, is still absent. Also, for such a monitoring scheme, how to balance the tradeoff between accuracy and overhead has not been studied yet. To address these challenges, we design a P4-based flexible multilayer in-band network telemetry (ML-INT) system to visualize an IP-over-optical network in realtime. Specifically, the flexible ML-INT scheme only selects a small portion of packets in an IP flow to encode in-band network telemetry (INT) headers, while each INT header only contains a part of the statistics of all the electrical/optical network elements (NEs) on the flow's routing path. We design the packet processing pipelines for the scheme, program P4-based hardware programmable data-plane (PDP) switches to implement the pipelines, develop optical performance monitors that can cooperate with the PDP switches to facilitate ML-INT, and implement a high-performance data analyzer that can extract, parse and analyze the INT data carried by high-speed IP flows (*i.e.*, with an arrival rate up to 2 million packets per second (Mpps)). The whole flexible ML-INT system is experimentally demonstrated in a small-scale but real IP-over-optical network testbed. The experimental results verify that our proposal only introduces very small overhead and can make the IP-over-optical network more visible in realtime for performance monitoring and troubleshooting.**

***Index Terms*—IP-over-optical network, Multilayer in-band network telemetry (ML-INT), Programmable data-plane (PDP).**

I. Introduction

RECENTLY, with the development of emerging network services, both the traffic and infrastructure of backbone networks are undergoing dramatic changes [1]. Specifically, the traffic has been not only growing exponentially in volume but also becoming increasingly bursty and dynamic [2, 3], and this has applied heavy pressure on the rigid IP-over-optical infrastructure of backbone networks [4–6]. Therefore, network operators first need a much more flexible network architecture that can build dynamic lightpaths adaptively in the optical layer for time-variant traffic flows in the IP layer, and then require an effective network control and management (NC&M) scheme that can make intelligent and timely decisions to groom and route the traffic flows over the lightpaths for realizing efficient resource utilization and high quality-of-service (QoS) simultaneously.

The first demand can be fulfilled by incorporating the technical innovations on flexible-grid elastic optical networks (EONs) [7–10] and leveraging the symbiosis of IP and EON technologies to compose IP-over-EONs [11, 12]. However, it is never an easy job to satisfy the second demand, since an efficient NC&M scheme needs systematic supports from a few areas [13], some of which are still in development. Here, an important but tricky one is the visualization of IP-over-optical networks for fine-grained performance monitoring and troubleshooting [14]. The major challenge is that IP-over-optical networks are becoming more and more agile and programmable at the cost of increased complexity [15, 16].

For both IP and optical networks, the investigations on performance monitoring and troubleshooting have a long history. For instance, the simple network management protocol (SNMP) [17] has been standardized a few decades ago for pulling statistics from the network elements in an IP network, while how to monitor metrics such as optical signal-to-noise ratio (OSNR) and dispersion in an optical network has been reviewed in [18, 19] more than a decade ago.

Although these techniques have been proven to be effective, the recent advances on IP-over-optical networks have brought in new challenges in three aspects. Firstly, the monitoring and troubleshooting on an IP-over-optical network should not treat its IP and optical layers separately. For example, the shrink on an IP flow's receiving bandwidth can be caused by either the congestion in an intermediate IP router or the OSNR degradation on a used lightpath. Hence, an effective multilayer monitoring scheme is necessary to find the real root-causes [20]. Secondly, considering the burstiness on traffic in today's backbone networks, fine-grained and realtime monitoring is desired [21–23]. Nevertheless, fine-grained and realtime monitoring would lead to unbearable complexity and flood the NC&M system with tremendous status data [24–26], if it is based on a centralized scheme. Lastly but not the least, the monitoring scheme should be programmable to reveal the end-to-end information of an arbitrary flow with high accuracy and minimum overhead.

The last two challenges can be addressed by leveraging the in-band network telemetry (INT) [27], which was designed based on a programming protocol-independent packet processor (P4) based programmable data-plane (PDP) [28] and enables network operators to achieve realtime, distributed and end-to-end flow monitoring in packet networks. However, to

B. Niu, J. Kong, S. Tang, Y. Li and Z. Zhu are with the School of Information Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, P. R. China (email: zqzhu@ieee.org).

Manuscript received on January 24, 2019.

the best of our knowledge, there is no reference design to resolve the first challenge, since multilayer monitoring needs to collect comprehensive and realtime statistics regarding both electrical/optical network elements (NEs) on an arbitrary flow's routing path. Although optical parameters can be easily collected with a few techniques [29], how to merge the results with IP network statistics and encode them to realize multilayer INT (ML-INT) is a challenging problem. Note that, only extending the INT header fields defined in [27] to include optical parameters is far from enough. This is because encoding the statistics of all the electrical/optical NEs on a path as INT fields and inserting them in packet headers would lead to excessive bandwidth overheads.

In this paper, we design and experimentally demonstrate a P4-based flexible ML-INT system to resolve the aforementioned issues and to visualize an IP-over-optical network in realtime. Specifically, we design a flexible ML-INT scheme that uses a fixed sampling scheme to select a small portion of packets in an IP flow according to a preset sampling rate and encodes INT headers on them. Each INT header only contains a part of the statistics of all the electrical/optical NEs on the flow's routing path. This method can not only minimize the total bandwidth overhead of INT but also greatly reduce the length of the INT header on each INT packet (*i.e.*, a packet that carries an INT header). The rationale behind this selective insertion is that the line-rate of a backbone network is usually very high (*i.e.*, 10 Gbps and beyond) and thus sampling network status per-packet would not be necessary.

The major contributions of this work are as follows.

- We design the packet processing pipelines for the flexible ML-INT scheme, and program P4-based hardware PDP switches to implement the pipelines.
- We develop optical performance monitors that can cooperate with the PDP switches to facilitate ML-INT.
- We implement a high-performance data analyzer that can extract, parse and analyze the INT data carried by high-speed IP flows (*i.e.*, with data-rates up to 10 Gbps).
- We build a small-scale but real IP-over-optical network testbed, which includes P4-based PDP switches as the IP layer and establishes its optical layer with bandwidth-variable wavelength selective switches (BV-WSS'), erbium-doped fiber amplifiers (EDFAs) and fiber links, and use it to demonstrate the effectiveness of our flexible ML-INT scheme experimentally.
- Experimental results verify that our proposal only introduces very small overhead to visualize the IP-over-optical network in realtime for monitoring and troubleshooting.

The rest of the paper is organized as follows. Section II briefly reviews the related work. We describe the architecture of the flexible ML-INT system and its operation principle in Section III, while the system implementation is presented in Section IV. The experimental demonstrations are discussed in Section V. Finally, Section VI summarizes the paper.

II. Related Work

In earlier days of the Internet, people have designed and standardized a few out-of-band monitoring approaches for IP networks, *e.g.*, SNMP [17] and NetFlow [30]. As these monitoring approaches are centralized and use periodic polling to collect network status from NEs, they have a few limitations. Firstly, the polling-based data collection can hardly reveal the network status in realtime. Secondly, the centralized mechanism has scalability issues, since flooding of status data to the centralized NC&M system will happen if the polling interval is too small and/or the NEs in the network are many. Lastly but most importantly, they can hardly operate at the flow-level granularity to provide the end-to-end information regarding an arbitrary flow with high accuracy.

Therefore, people considered in-band monitoring schemes as alternatives. For instance, NetSight [31] was designed to capture every packet processing procedure for easing troubleshooting. The invention of P4-based PDP [28] has boosted up the research and development on in-band monitoring [32] and eventually led to the publication of INT specification [27]. It is promising to see that P4-based INT can be implemented on hardware PDP switches to make every packet processing visible at a line-rate of 100 Gbps [33], and this motivates us to realize ML-INT with the help of hardware PDP switches.

The optical performance monitoring schemes have been surveyed in [18, 29], and they provide us with the enabling techniques to design our flexible ML-INT. However, none of these studies has considered how to integrate optical performance monitoring and the INT for IP networks for realizing ML-INT. Recently, the authors of [34] have designed and demonstrated an interesting optical layer telemetry service, which utilized the gPRC protocol to enable on-demand streaming of realtime monitoring results that were dynamically retrieved from a configurable set of optical NEs. Nevertheless, they only focused on how to deliver the realtime monitoring results but did not consider the actual data collection for the optical layer telemetry service. Moreover, as their telemetry service is not multilayer-capable, it can hardly reveal the end-to-end information of an arbitrary IP flow in realtime and with high accuracy. POINT [35] is the only known system that has considered ML-INT, but as it did not try to insert INT headers in packets in a flexible and selective manner, its INT packets could have excessively long packet headers. In other words, the overhead due to ML-INT was not optimized in POINT.

III. System Design

A. System Architecture

Fig. 1(a) shows the overall architecture of our flexible ML-INT system, which resides in an IP-over-optical network. The IP layer consists of P4-based PDP switches and client hosts, where the PDP switches are interconnected by the lightpaths established in the optical layer. We design the packet processing pipelines for the flexible ML-INT and implement them in the PDP switches. Then, if an IP flow between two client hosts will be involved in ML-INT, the ingress and subsequent PDP switches on its routing path will select a portion of the flow's packets according to a preset sampling rate and insert INT fields in them (as illustrated in Fig. 1(a)).

Each INT field contains a required statistic of an electrical or optical NE on the flow's path, *e.g.*, the packet forwarding

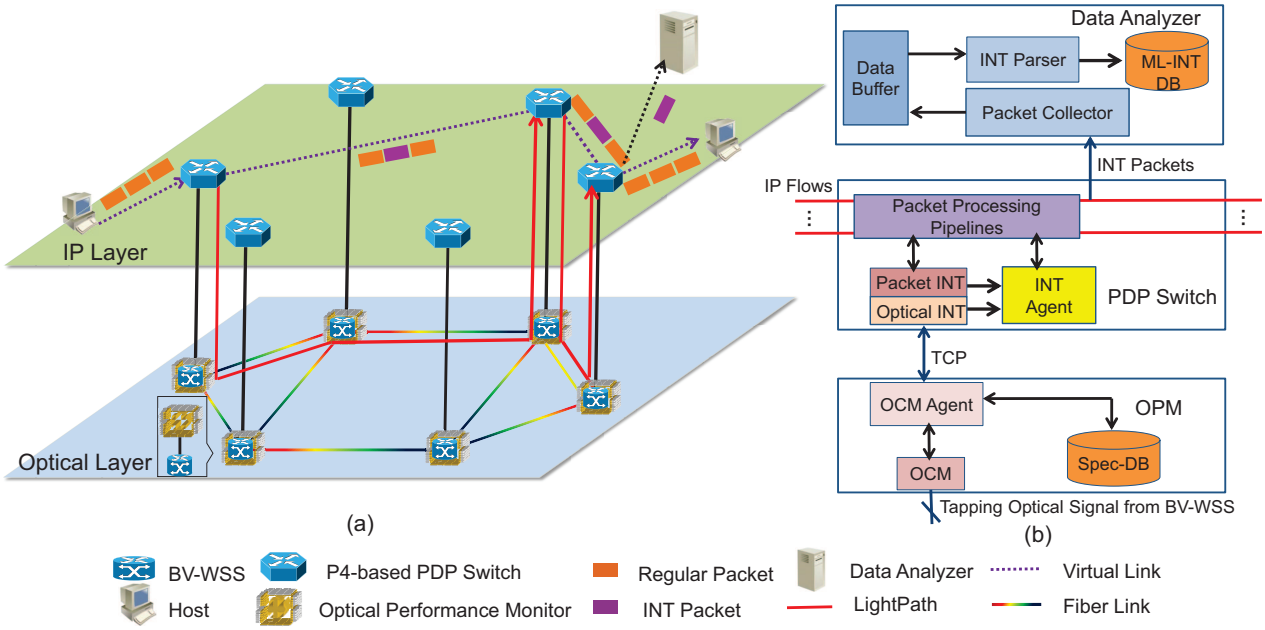


Fig. 1. System architecture of flexible ML-INT system, BV-WSS: Bandwidth-variable wavelength selective switch, OPM: Optical performance monitor, OCM: Flex-grid high resolution optical channel monitor, ML-INT DB: Multilayer INT database, Spec-DB: Optical spectrum database.

latency in a PDP switch or the OSNR of a lightpath at the input port of a BV-WSS. Hence, even though not all the packets contain INT fields (*i.e.*, not all the packets are INT packets) and an INT packet only includes a fragment of the required statistics regarding all the electrical/optical NEs on the flow's path, a complete and realtime view about all the NEs on the path can be got after aggregating the INT fields in different INT packets. Here, to make the ML-INT transparent to client hosts, the egress PDP switch converts the INT packets back to regular ones by removing all the INT fields in them, while the INT packets are simultaneously sent to a homemade data analyzer for data aggregation, analysis and storage.

For the optical layer, we consider an EON and build it with BV-WSS' (*i.e.*, for routing and spectrum assignment (RSA) [36–38]) and fiber links with inline EDFAs. To facilitate ML-INT, we implement an optical performance monitor (OPM) on each BV-WSS to collect parameters of the lightpaths that go through it. We also design the interface between an OPM and the PDP switch that directly connects to it, such that the PDP switch can poll the OPM in realtime and request for the optical parameters of an arbitrary lightpath going through its BV-WSS. For ML-INT, each PDP switch can collect a flow's statistics in realtime, such as its input/output ports, length of the packet queue for it, and its latency through the switch, while each OPM is based on optical spectrum analysis to obtain the optical information of a lightpath, including the OSNR, input/output power, and central wavelength.

The detailed architectures inside the data analyzer, PDP switch, and OPM and the interactions among them are plotted in Fig. 1(b). The OPM taps optical signal from the input ports of its BV-WSS for performance monitoring. Here, to achieve high resolution optical spectrum analysis, we build the OPM based on a flex-grid high resolution optical channel monitor (OCM) [39]. The OCM Agent gets optical spectrum data from the OCM, analyzes it to identify the lightpaths, extracts the lightpaths' optical parameters, and stores them in the spectrum database (Spec-DB). Meanwhile, the OCM Agent also communicates with the Optical INT module in a PDP switch using TCP and responds to the requests for lightpaths' optical parameters in realtime.

Each PDP switch can be programmed with the P4 language [28] to define packet processing pipelines in its forwarding hardware, and it is the key enabling NE of our flexible ML-INT system. In addition to the Optical INT module, we also implement a Packet INT module in each PDP switch to collect the statistics of an arbitrary flow going through the switch in realtime. The statistics regarding the IP and optical layers are aggregated by the INT Agent, while the packet processing pipelines are customized to define: 1) the sampling rate for INT insertion, 2) the maximum number of INT fields that can be inserted in an INT packet, 3) the electrical/optical NEs that are selected for statistic collection, and 4) the required statistics of each selected NE.

When an INT packet reaches the egress switch on its routing path, it will be mirrored to the data analyzer and converted back to a regular one before being sent to the client host, also by the switch's packet processing pipelines. In our flexible ML-INT systems, we consider two types of PDP switches, *i.e.*, Linux servers equipped with P4-enabled SmartNICs [40] and programmable ASIC switches [33], and design proper packet processing pipelines according to their operation principles.

The data analyzer is homemade and runs on a Linux server. Specifically, we optimize its design and implementation to ensure that it can receive high-throughput INT packets (*i.e.*, up to 2 million packets per second) and parse the INT fields in them correctly, for extracting the statistics of the IP and optical layers to store in the ML-INT database (ML-INT DB). Note that, there could be multiple data analyzers in the flexible

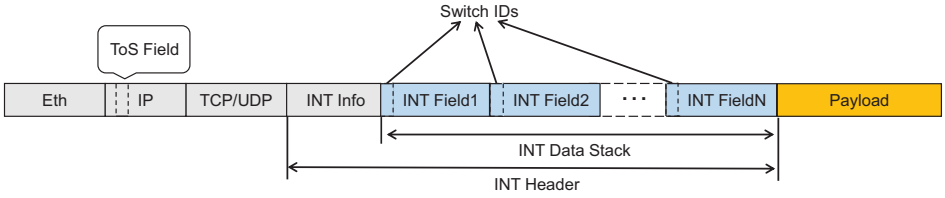


Fig. 2. Packet format for realizing flexible ML-INT.

ML-INT system, each of which is placed close to an edge switch in the IP layer. Hence, the data analyzers operate in a distributed manner to cover the ML-INT for different flows, *i.e.*, improving the scalability of performance monitoring and troubleshooting. Specifically, as they can visualize the IP-over-optical network in realtime to see how the flows are handled end-to-end, they only need to feed very digested network status to the centralized NC&M and let it invoke network reconfiguration when necessary.

B. Operation Principle

Our flexible ML-INT system leverages the INT packet format defined in [27] with some extensions. Specifically, we insert an INT header after the TCP/UDP header of an IP packet and convert it to an INT packet. The INT header consists of an *INT Info* field and an INT data stack that includes a series of *INT Fields*, as shown in Fig. 2. Here, the *INT Info* contains the information about the INT header, such as its version number, the number of *INT Fields* that have been inserted, and the remaining space for more *INT Fields*[1]. In an INT packet, each *INT Field* in its INT data stack corresponds to a hop on its routing path, which consists of a lightpath in the optical layer and the lightpath's destination PDP switch in the IP layer.

Since ML-INT can collect multiple statistics to monitor the electrical/optical NEs in the hop, we empirically limit the number of statistics that can be included in an *INT Field* to its smallest possible value (*i.e.*, two), to avoid generating an excessively long INT header. For the two statistics in an *INT Field*, one is mandatorily set as the *Switch ID* of the PDP switch that inserts it, and the other can be flexibly selected from all the supported statistics regarding electrical/optical NEs. If the number of the required statistics of a hop is two or more, our system distributes them in different INT packets and leaves the data aggregation to the data analyzer.

Therefore, the INT header of each INT packet in our flexible ML-INT system would not be longer than that of an INT packet in the traditional single-layer INT system [27], and as long as the sampling schemes are properly designed, our system can collect network statistics timely and realize performance monitoring in realtime. Meanwhile, since our system does not insert the INT header to each packet in a flow, we need an indicator in the IP header to distinguish INT packets from normal ones. To achieve this, we leverage the *ToS* field in IP header, which is set as $0x5c$ in an INT packet.

To realize the flexible ML-INT, we design the packet processing pipelines as shown in Fig. 3, program them with the P4 language, and implement the programs in the PDP switches. When a packet from a client host first enters the IP-over-optical network, it encounters the ingress PDP switch whose packet processing pipeline is shown in Fig. 3(a). The pipeline first uses a parser to identify the concerned fields in the packet, and then the match-action table checks the fields to determine whether the packet is in a flow that has been selected for ML-INT. Note that, which flows need the ML-INT service is decided by the network operator according to the flows' QoS requirements and network status. If not selected for ML-INT, the packet is directly sent to the forwarder, which will send it to the next hop. Otherwise, the packet is passed to the INT selection module through the packet buffer.

The INT selection module obtains tokens from the INT arbiter to realize selective INT header insertion, which means that the frequency of token generation in the INT arbiter determines the sampling rate of the selective insertion. Then, the packet will be sent to the forwarder if no token has been generated, and will be inserted with an INT header, otherwise. The INT arbiter also determines the type of the statistics (*i.e.*, INT type) to be encoded in the *INT Field*, while the actual values of the statistics are given by the INT Agent. Finally, the forwarder sends the INT packet to the next hop.

The pipeline for an intermediate PDP switch on the flow's routing path is simpler, as illustrated in Fig. 3(b). Here, after the parser, the pipeline first determines whether the packet is an INT one. If yes, the INT insertion module will insert an *INT Field* in the packet according to the INT types from the INT arbiter. For the egress PDP switch, its pipeline in Fig. 3(c) is similar as that of an intermediate switch, except for the duplicator and INT removal module placed after the INT insertion module. The duplicator mirrors each INT packet, and the INT removal module then deletes the INT header on each INT packet before forwarding it to a client host. Meanwhile, the mirrored INT packet is sent to the data analyzer.

Note that, the INT arbiter is important in our packet processing pipelines for ML-INT. Specifically, in a PDP switch, each ML-INT flow is identified by and associated with an INT arbiter. In an ingress PDP switch, every INT arbiter consists of three registers. The first register is the counter to record the number of packets that an ML-INT flow has transmitted, the second one is in charge of generating tokens for the ML-INT flow according to a selection ratio, while the last one determines which INT type should be inserted into an ML-INT

[1]Before inserting an INT header in a packet, each PDP switch will hypothetically check whether the length of the packet after the insertion would be longer than the maximum transmission unit (MTU) of its network. If yes, the switch will not insert the INT header in the packet and save the operation for the next packet that is short enough. This helps us avoid the complexity from packet fragmentation and concatenation.

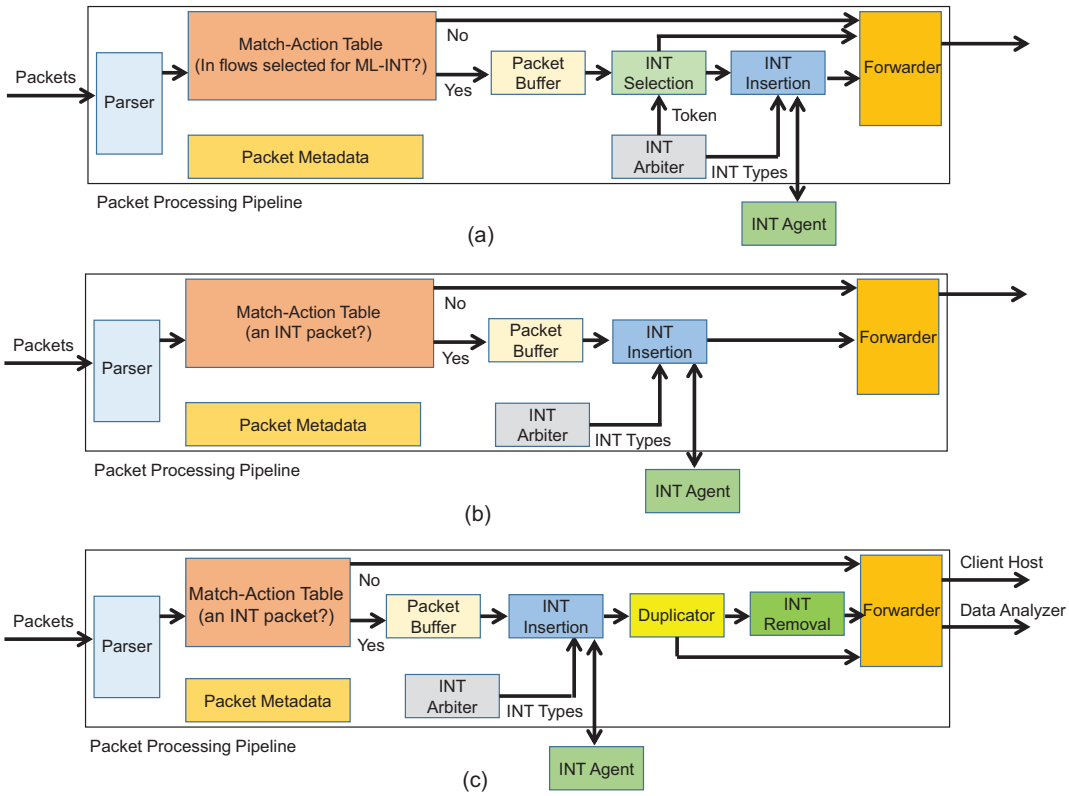


Fig. 3. Packet processing pipelines designed for realizing flexible ML-INT in (a) the ingress switch, (b) an intermediate switch, and (c) the egress switch.

packet. On the other hand, in intermediate and egress switches, each INT arbiter only includes two registers, where the register for generating tokens is omitted. Here, the counters will be queried periodically, and if the system finds that a counter has not been updated since the last query, the corresponding INT arbiter will be cleared and recycled. Meanwhile, our design also supports the readjustment of the flexible ML-INT scheme of each flow (*i.e.*, the sampling rate of selective INT header insertion and the required statistics in each hop) at runtime. This is because we design the INT arbiter such that it can be updated during the runtime of each PDP switch, without being taken offline for recompilation.

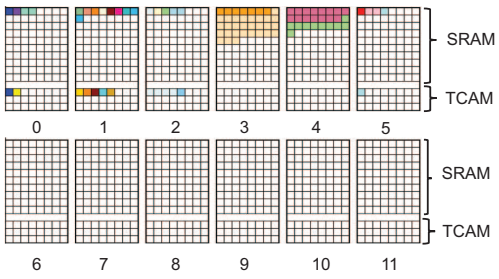
IV. Implementation

As we have explained before, each PDP switch in our flexible ML-INT system can be either a Linux server equipped with P4-enabled SmartNICs or a programmable ASIC switch. The SmartNICs are based on NFP-4000 network processor units (NPUs) that can be programmed by both P4 and Micro C, while the programmable ASIC switch is actually the 3.2 Tbps Barefoot switch with a Tofino ASIC that can be programmed by P4. The table in Fig. 4(a) summarizes the resource usage of our implementation on a SmartNIC, *i.e.*, how much space in percentage our implementation uses on different types of memory/storage in the SmartNIC. It can be seen that our implementation still leaves plenty of memory/storage space on the SmartNIC for other applications.

Fig. 4(b) shows the resource utilization of our implementation in a Tofino ASIC, which indicates that we only use

Resource Type	Usage (%)
Local Memory	54.65
Cluster Local Scratch Storage	28.74
Cluster Target Memory	45.01
Internal Memory	18.03
External Memory	3.74

(a)



(b)

Fig. 4. Memory resource utilizations of our ML-INT implementation on (a) SmartNIC, and (b) programmable ASIC switch.

small amount of static random-access memory (SRAM) and ternary content-addressable memory (TCAM) in the first six stages on the Tofino ASIC. The hardware of the OPM is based on an OCM, while we program its software part (*i.e.*, the OCM Agent and Spec-DB in Fig. 1(b)) with Python. The Optical INT module and INT Agent in each PDP switch are

also programmed with Python, and they run in the operating system of the switch. The INT Agent provides the latest optical parameters of a concerned lightpath to the switch's packet processing pipeline through internal APIs. We implement the data analyzer with the C language based on the Libpcap [41].

V. Experimental Demonstrations

The setup used for experimental demonstrations is shown in Fig. 5. Here, the optical layer is an EON, where each optical node is built with the commercial 1×9 BV-WSS' that operates within $[1528.43, 1566.88]$ nm and has a bandwidth allocation granularity of 12.5 GHz, while each optical link is a fiber link with an inline EDFA. The PDP switches in the IP layer are equiped with 10GbE optical ports, and thus each lightpath in the optical layer has a capacity of 10 Gbps. We emulate the client hosts with a commercial traffic generator/analyzer. Our experimental demonstrations include two parts. We first verify the proposed functionalities of our flexible ML-INT system have been implemented correctly. Then, we utilize two use-cases of the flexible ML-INT system to explain the benefits of visualizing an IP-over-optical network in realtime.

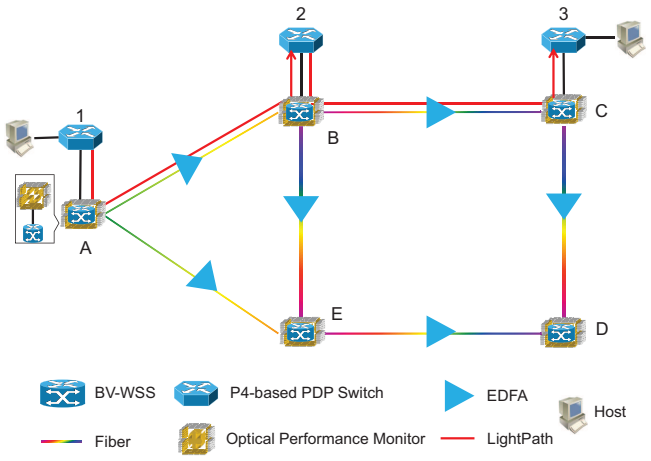


Fig. 5. IP-over-optical network testbed used for experimental demonstrations.

A. Verification of Functionalities

To verify the functionalities of our proposed system, we connect the source and destination client hosts to *Switches* 1 and 3 in the IP layer, respectively, and set up two lightpaths in the optical layer (*i.e.*, A-B and B-C) to connect *Switches* 1 and 2 and *Switches* 2 and 3, respectively. Hence, the IP flow between the client hosts uses *Switches* 1-3 as the ingress, intermediate, and egress switches, respectively. Then, we set the sampling rate of selective INT header insertion as 0.5, and make the ML-INT scheme to collect the input port and packet queuing latency of the flow on *Switch* 1, the packet queuing latency of the flow and the OSNR and input power of *Lightpath* A-B on *Switch* 2, and the packet queuing latency of the flow and the OSNR and input power of *Lightpath* B-C on *Switch* 3. Fig. 6(a) shows six consecutive packets captured at the output of *Switch* 2 with Wireshark. Here, the Wireshark has been extended for being able to parse the new header fields introduced for ML-INT.

We can see that three of the six captured packets are INT packets, which confirms the correct implementation of selective INT header insertion. Also, if we look into an INT packet, we observe that its *ToS* field in IP header has been set as $0x5c$. Then, to see the detailed INT headers, we expand the Wireshark captures of the three INT packets and list the results in Figs. 6(b)-6(d), respectively. It can be seen that in addition to the mandatory INT data on *Switch IDs*, the second INT data has different types in the three INT packets. Moreover, if we aggregate the data in the INT fields in the INT packets, we can get the input port and packet queuing latency of the flow on *Switch* 1, and the packet queuing latency of the flow on *Switch* 2 and the OSNR and input power of *Lightpath* A-B, which is exactly as we programmed.

Next, we keep the ML-INT scheme but pump packets with different sizes at a rate of one million packets per second (*i.e.*, 1 Mpps) through the routing path. We set the sampling rate of the selective INT header insertion as $\{0, 0.1, 0.5, 1\}$, and measure the receiving bandwidth at the output of *Switch* 2 to investigate the bandwidth overhead of the flexible ML-INT. Figs. 7(a) and 7(b) show the results about the bandwidth and ML-INT overhead at the output of *Switch* 2, respectively, when different initial packet sizes (*i.e.*, the size of a packet without the INT header) are used. Here, we define the ML-INT overhead as the ratio between the bandwidth used by INT headers and the overall bandwidth. The results indicate that our flexible ML-INT system can reduce the bandwidth overhead due to INT greatly. Specially, the worst ML-INT overhead is 30.4%, which happens when the initial packet size is 64 bytes and we use a sampling rate of 1 to insert an INT header in each packet, while for the packets whose initial sizes are 256 bytes or larger, the ML-INT overhead is very small and can be ignored even when the sampling rate is 1.

Finally, we verify the performance of the data analyzer. We fix the packet rate at 2 Mpps, select the sampling rate as 1, and run the experiments for 20 seconds. The realtime statistics regarding the packet queuing latency of the flow on *Switch* 3 and the OSNR and power of *Lightpath* B-C are plotted in Figs. 8(a)-8(c), respectively. The data analyzer works correctly to deliver continuous ML-INT data regarding the latency through *Switch* 3, and the power and OSNR of *Lightpath* B-C, even when the arrival rate of INT packets is 2 Mpps.

B. Visualizing IP-over-Optical Network for Troubleshooting

To further demonstrate the benefits and effectiveness of our flexible ML-INT system, we perform experiments on three use-cases to visualize the IP-over-optical network in realtime for troubleshooting. The routing path of the flow stays unchanged, but we add background flows and lightpaths in the testbed. Meanwhile, to minimize the bandwidth overhead, we set the sampling rate of selective INT header insertion as 0.01.

1) First Use-case: We inject background flows at *Switch* 2 with a total capacity of 8 Gbps, and make them share *Switch* 2 and *Lightpath* B-C with our concerned flow from *Switch* 1 to *Switch* 3. Both the background and concerned flows are

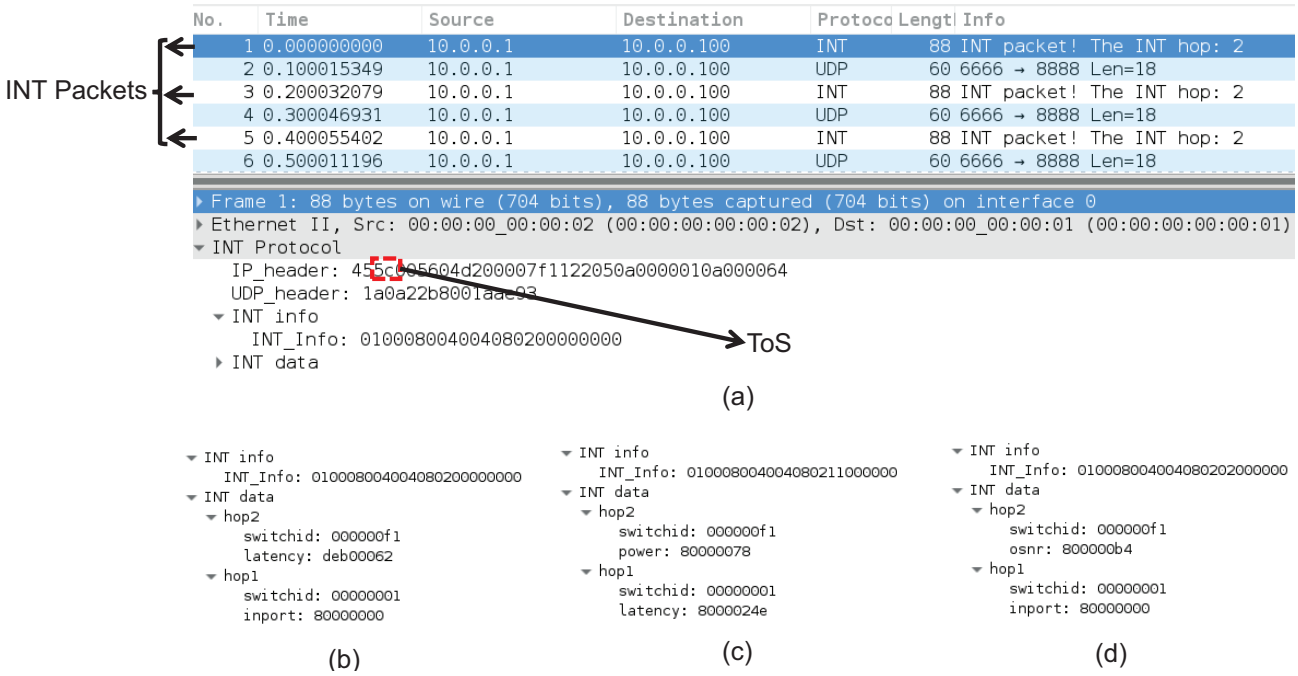


Fig. 6. Wireshark captures to verify functionalities of our flexible ML-INT system, (a) packets captured at the *Switch* 2, (b) INT header of the first INT packet, (c) INT header of the second INT packet, and (d) INT header of the third INT packet.

based on TCP. The capacity of the concerned flow is 1.1 Gbps at $t = 0$, and changes to 2.2, 4.4, 2.2, and 1.1 Gbps at $t = 18$, 34, 45, and 60 seconds, respectively. Hence, we expect *Switch* 2 to be slightly congested within $[18, 34]$ and $[45, 60]$ seconds and to be heavily congested within $[34, 45]$ seconds. We program the flexible ML-INT system to collect the packet queuing latency at *Switches* 1-3, and plot the results from the data analyzer in Figs. 9(a)-9(c).

Meanwhile, to verify the correctness of the measurements from our flexible ML-INT system, we use the commercial traffic analyzer to measure the end-to-end latency of the flow and show the results in Fig. 9(d). We observe that among the three switches, only *Switch* 2 has significantly-increased queuing latency during $[18, 60]$ seconds (in Fig. 9(b)), which follows the same trend as that of the end-to-end latency in Fig. 9(d). Hence, the results indicate that there is congestion on *Switch* 2 during $[18, 60]$ seconds.

Moreover, the results in Fig. 9(b) also indicate that the latencies during $[18, 34]$ and $[45, 60]$ seconds are shorter than that within $[34, 45]$ seconds. This is because the congestion is severer during $[34, 45]$ seconds. Specifically, the PDP switches provide different latencies when they are experiencing different levels of congestion, according to our experiments. Also, the latency from a PDP switch includes both the queuing and processing latencies, and the latter also increases when congestion occurs. This is the reason why the latency difference between the high and low congestion levels in Fig. 9(b) is not the same as that in Fig. 9(d). To this end, we can see that our system can detect not only the occurrences of congestion in the IP layer but also their locations and severities.

2) Second Use-case: We set the capacity of the concerned flow as 8 Gbps, and make sure that there is no exception in the IP layer. The IP-over-optical network will experience

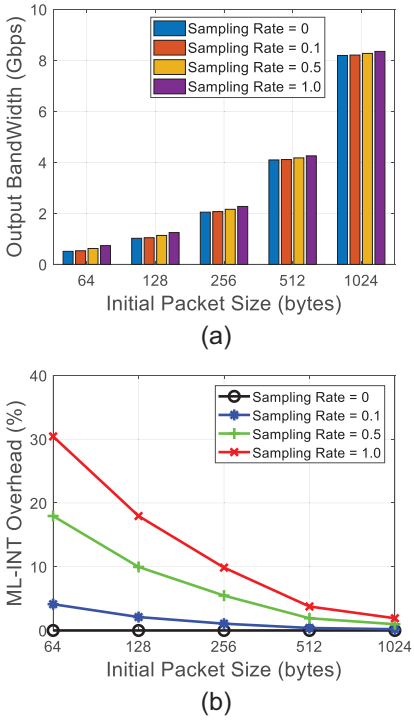


Fig. 7. Measurements at the output of *Switch* 2 regarding (a) bandwidth, and (b) ML-INT overhead, when different initial packet sizes and sampling rates are considered.

unexpected power loss on *Lightpath* B-C after $t = 32$ seconds. We still collect the packet queuing latency at *Switches* 1-3 (in Figs. 10(a)-10(c)), and moreover, the power levels of the two lightpaths terminated at *Switches* 2 and 3 are also collected (in Figs. 10(d) and 10(e)) for troubleshooting. Meanwhile,

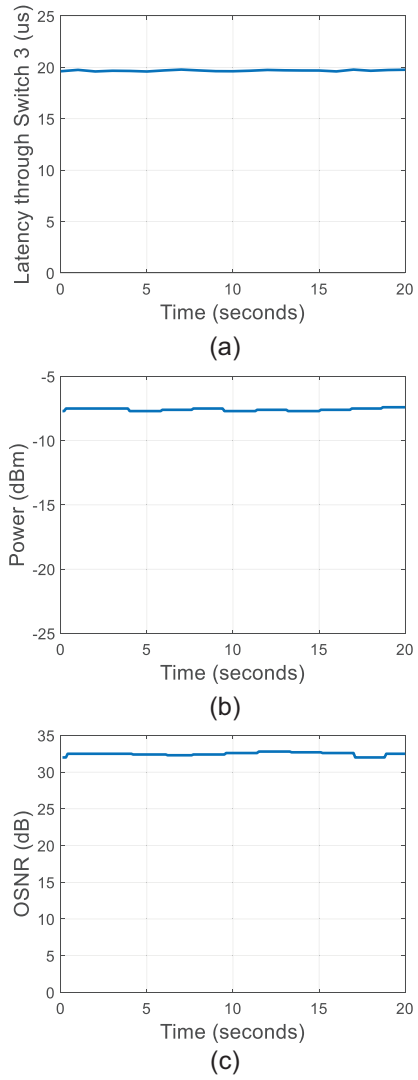


Fig. 8. Realtime statistics provided by the data analyzer regarding (a) packet queuing latency through *Switch* 3, and (b) power and (c) OSNR of *Lightpath* B-C, when the arrival rate of INT packets is 2 Mpps.

we measure the end-to-end bandwidth of the flow with the commercial traffic analyzer and show the results in Fig. 10(f). The latency results in Figs. 10(a)-10(c) confirm that there is no traffic congestion in the IP layer, but the end-to-end bandwidth in Fig. 10(f) shows dramatic drops after $t = 32$ seconds.

Switch 2 uses the programmable ASIC switch, and due to its superior packet processing performance, the packet queuing latency in it (Fig. 10(b)) is much shorter than those in *Switches* 1 and 3 that use SmartNICs (Figs. 10(a) and 10(c)). As there is no congestion in the IP layer, we need to debug the optical layer by checking the power results in Figs. 10(d) and 10(e). Although the results in Fig. 10(d) indicate that the power level of *Lightpath* A-B is normal, Fig. 10(d) shows that there is a sudden power drop at $t = 32$ seconds on *Lightpath* B-C, which coincides with the bandwidth drop in Fig. 10(f). To this end, we can infer that the sudden drop on the flow's end-to-end bandwidth is due to the power drop on *Lightpath* B-C.

3) Third Use-case: We still keep the capacity of the concerned flow as 8 Gbps. There will be no exception in the IP-layer, but the IP-over-optical network will experience unexpected power and OSNR changes on the *Lightpath* B-C after $t = 66$ seconds. The results on the packet queuing latency at *Switches* 1-3 are plotted in Figs. 11(a)-11(c), respectively, which indicate that the PDP switches in the IP layer do not encounter any congestion. Meanwhile, the optical layer information regarding the lightpath terminated at *Switch* 2 also does not show any exception in Fig. 11(d) either. However, the end-to-end bandwidth measurement in Fig. 11(f) suggests that the IP flow experiences significant bandwidth decreases after $t = 66$ seconds. These exceptions should be attributed to the unexpected power and OSNR changes on the lightpath terminated at *Switch* 3 (*i.e.*, *Lightpath* B-C), as shown in Fig. 11(e). Nevertheless, since the power and OSNR of *Lightpath* B-C change constantly throughout the experiment, we cannot use a simple threshold-based mechanism to identify the normal and abnormal zones of the lightpath's operation.

Because our ML-INT system can reveal the time and spatial correlations among the statistics collected from both the IP and optical layers, we can make it consider more combinations of OSNR and power and investigate their combined effect on the receiving bandwidth in the IP layer. This can be done in advance, and we store the measurements as historical INT data (as shown in Fig. 11(h)). Then, based on the data analytics in Fig. 11(h), we can easily identify the abnormal zones in Fig. 11(g) for the operation of *Lightpath* B-C, *i.e.*, *Zones* 1 and 2. Hence, we can see that since our ML-INT system can reveal the time and spatial correlations among the statistics collected from an IP-over-optical network, it can be easily integrated with data analytics schemes to detect relatively complicated exceptions and identify and locate their root causes.

Note that, even though these three use-cases are relatively simple, they do verify the effectiveness of our flexible ML-INT system on visualizing the IP-over-optical network in realtime for troubleshooting. More importantly, the analysis of the INT data and the troubleshooting based on it are conducted by the data analyzer without any involvement of the centralized NC&M, which will be informed only after a failure's root-cause having been located. This suggests that our proposal can achieve on-demand, fine-grained, and distributed performance monitoring to provide the end-to-end information regarding an arbitrary flow in the IP-over-optical network, without causing any additional complexity on the centralized NC&M.

VI. Conclusion

In this work, we designed a P4-based flexible ML-INT system to visualize IP-over-optical networks in realtime. More specifically, we selectively inserted INT data in the packets of a flow to not only minimize the total bandwidth used for INT but also greatly reduce the length of the INT header on each INT packet. The flexible ML-INT system was experimentally demonstrated in a small-scale but real IP-over-optical network testbed, which included P4-based PDP switches as the IP layer and built its optical layer with BV-WSS', EDFAs and fiber links. The experimental results verified that the INT overhead of our proposal is very small while it does make an IP-over-optical network more visible in realtime for performance monitoring and troubleshooting.

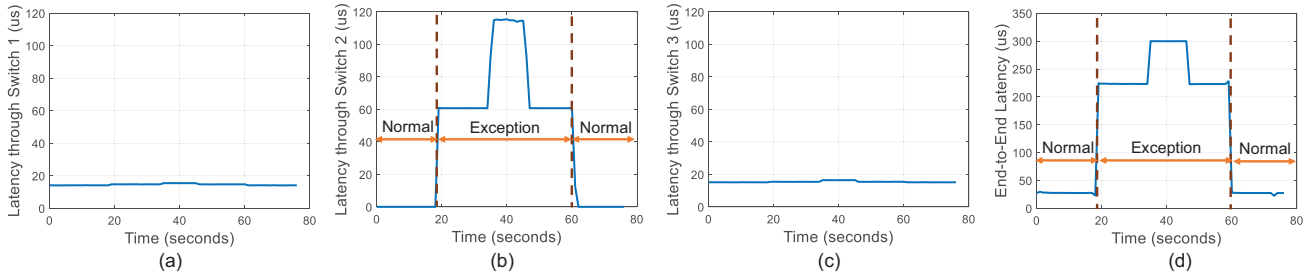


Fig. 9. Results for the first use-case, (a) packet queuing latency at *Switch* 1, (b) packet queuing latency at *Switch* 2, (c) packet queuing latency at *Switch* 3, and (d) end-to-end latency measured by commercial traffic analyzer.

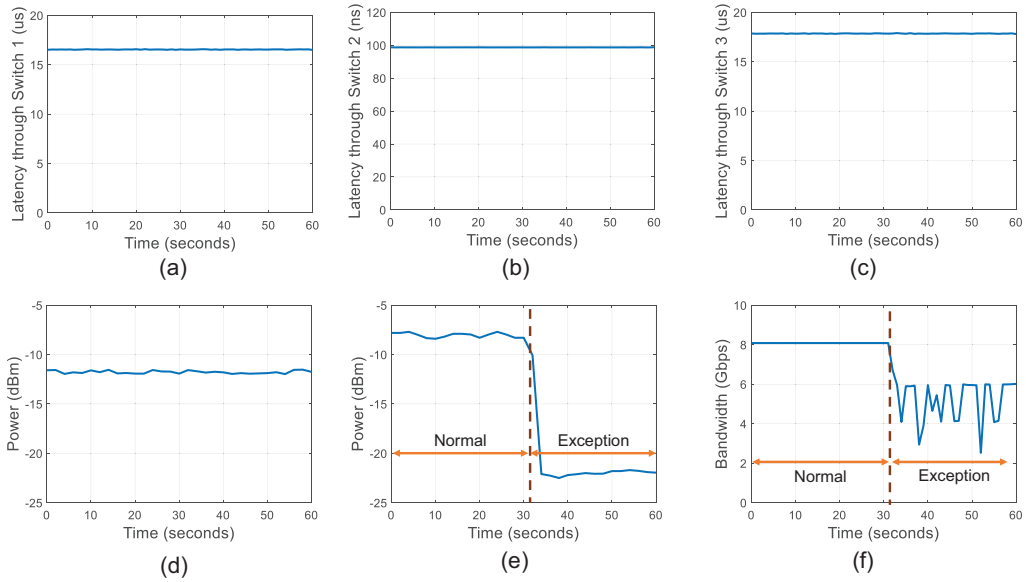


Fig. 10. Results for the second use-case, (a) packet queuing latency at *Switch* 1, (b) packet queuing latency at *Switch* 2, (c) packet queuing latency at *Switch* 3, (d) power of lightpath terminated at *Switch* 2, (e) power of lightpath terminated at *Switch* 3, and (f) end-to-end bandwidth measured by commercial traffic analyzer.

Acknowledgments

This work was supported by the NSFC projects 61871357 and 61701472, CAS key project (QYZDY-SSW-JSC003), and NGBWMCN key project (2017ZX03001019-004).

References

[1] Cisco Visual Networking Index: Forecast and Methodology, 2017-2022. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html

[2] A. Jain *et al.*, "B4: experience with a globally-deployed software defined WAN," in *Proc. of ACM SIGCOMM 2013*, pp. 3–14, Aug. 2013.

[3] B. Kong *et al.*, "Demonstration of application-driven network slicing and orchestration in optical/packet domains: On-demand vDC expansion for Hadoop MapReduce optimization," *Opt. Express*, vol. 26, pp. 14 066–14 085, 2018.

[4] P. Lu *et al.*, "Highly efficient data migration and backup for Big Data applications in elastic optical inter-data-center networks," *IEEE Netw.*, vol. 29, pp. 36–42, Sept./Oct. 2015.

[5] J. Yao, P. Lu, L. Gong, and Z. Zhu, "On fast and coordinated data backup in geo-distributed optical inter-datacenter networks," *J. Lightw. Technol.*, vol. 33, pp. 3005–3015, Jul. 2015.

[6] P. Lu and Z. Zhu, "Data-oriented task scheduling in fixed- and flexible-grid multilayer inter-DC optical networks: A comparison study," *J. Lightw. Technol.*, vol. 35, pp. 5335–5346, Dec. 2017.

[7] O. Gerstel, M. Jinno, A. Lord, and S. Yoo, "Elastic optical networking: A new dawn for the optical layer?" *IEEE Commu. Mag.*, vol. 50, pp. s12–s20, Feb. 2012.

[8] Z. Zhu, W. Lu, L. Zhang, and N. Ansari, "Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing," *J. Lightw. Technol.*, vol. 31, pp. 15–22, Jan. 2013.

[9] Y. Yin *et al.*, "Spectral and spatial 2D fragmentation-aware routing and spectrum assignment algorithms in elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. A100–A106, Oct. 2013.

[10] L. Gong *et al.*, "Efficient resource allocation for all-optical multicasting over spectrum-sliced elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. 836–847, Aug. 2013.

[11] O. Gerstel *et al.*, "Multi-layer capacity planning for IP-optical networks," *IEEE Commun. Mag.*, vol. 52, pp. 44–51, Jan. 2014.

[12] S. Liu, W. Lu, and Z. Zhu, "On the cross-layer orchestration to address IP router outages with cost-efficient multilayer restoration in IP-over-EONs," *J. Opt. Commun. Netw.*, vol. 10, pp. A122–A132, Jan. 2018.

[13] R. Casellas, R. Martinez, R. Vilalta, and R. Munoz, "Control, management, and orchestration of optical networks: Evolution, trends, and challenges," *J. Lightw. Technol.*, vol. 36, pp. 1390–1402, Apr. 2018.

[14] R. Govindan *et al.*, "Evolve or die: High-availability design principles drawn from Google's network infrastructure," in *Proc. of ACM SIGCOMM 2016*, pp. 58–72, Aug. 2016.

[15] W. Lu, X. Yin, X. Cheng, and Z. Zhu, "On cost-efficient integrated multilayer protection planning in IP-over-EONs," *J. Lightw. Technol.*, vol. 35, pp. 5335–5346, Dec. 2017.

[16] J. Guo and Z. Zhu, "When deep learning meets inter-datacenter optical network management: Advantages and vulnerabilities," *J. Lightw. Technol.*, vol. 36, pp. 4761–4773, Oct. 2018.

[17] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A simple network management protocol (SNMP)," *RFC 1098*, May 1990. [Online]. Available: https://tools.ietf.org/html/rfc1157

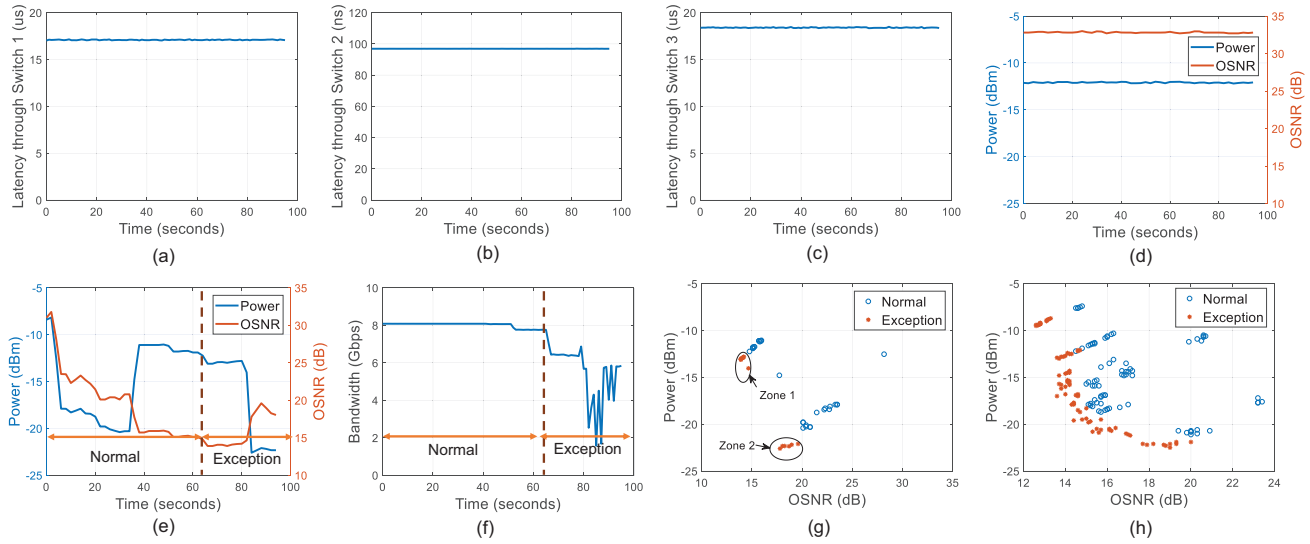


Fig. 11. Results for the third use-case, (a) packet queuing latency at *Switch* 1, (b) packet queuing latency at *Switch* 2, (c) packet queuing latency at *Switch* 3, (d) power and OSNR of lightpath terminated at *Switch* 2, (e) power and OSNR of lightpath terminated at *Switch* 3, (f) end-to-end bandwidth measured by commercial traffic analyzer, (g) INT data regarding the correlation between power and OSNR, and (h) historical INT data.

[18] D. Kilper *et al.*, "Optical performance monitoring," *J. Lightw. Technol.*, vol. 22, pp. 294–304, Jan. 2004.

[19] Z. Zhu *et al.*, "Jitter and amplitude noise accumulations in cascaded all-optical regenerators," *J. Lightw. Technol.*, vol. 26, pp. 1640–1652, Jun. 2008.

[20] H. Fang *et al.*, "Building network nervous system with multilayer telemetry to realize AI-assisted reflexes in software-defined IP-over-EONs for application-aware service provisioning," in *Proc. of OFC 2019*, pp. 1–3, Mar. 2019.

[21] X. Chen *et al.*, "Flexible availability-aware differentiated protection in software-defined elastic optical networks," *J. Lightw. Technol.*, vol. 33, pp. 3872–3882, Sept. 2015.

[22] J. Yin *et al.*, "Experimental demonstration of building and operating QoS-aware survivable vSD-EONs with transparent resiliency," *Opt. Express*, vol. 25, pp. 15 468–15 480, 2017.

[23] Z. Zhu *et al.*, "Build to tenants' requirements: On-demand application-driven vSD-EON slicing," *J. Opt. Commun. Netw.*, vol. 10, pp. A206–A215, Feb. 2018.

[24] X. Chen *et al.*, "Leveraging master-slave openflow controller arrangement to improve control plane resiliency in SD-EONs," *Opt. Express*, vol. 23, pp. 7550–7558, Mar. 2015.

[25] B. Zhao, X. Chen, J. Zhu, and Z. Zhu, "Survivable control plane establishment with live control service backup and migration in SD-EONs," *J. Opt. Commun. Netw.*, vol. 8, pp. 371–381, Jun. 2016.

[26] H. Huang *et al.*, "Realizing highly-available, scalable and protocol-independent vSDN slicing with a distributed network hypervisor system," *IEEE Access*, vol. 6, pp. 13 513–13 522, 2018.

[27] C. Kim *et al.* In-band network telemetry (INT). [Online]. Available: https://p4.org/assets/INT-current-spec.pdf

[28] P. Bosshart *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 87–95, Jul. 2014.

[29] Z. Dong *et al.*, "Optical performance monitoring: A review of current and future technologies," *J. Lightw. Technol.*, vol. 34, pp. 525–543, Jan. 2016.

[30] B. Claise, "Cisco systems NetFlow services export version 9," *RFC 3954*, Oct. 2004. [Online]. Available: https://tools.ietf.org/html/rfc3954

[31] N. Handigol *et al.*, "I know what your packet did last hop: Using packet histories to troubleshoot networks," in *Proc. of NSDI 2014*, pp. 71–85, Apr. 2014.

[32] C. Kim *et al.*, "In-band network telemetry via programmable dataplanes," in *Proc. of ACM SIGCOMM 2015*, pp. 1–2, Aug. 2015.

[33] Barefoot Deep Insight. [Online]. Available: https://www.barefootnetworks.com/products/brief-deep-insight/

[34] F. Paolucci, A. Sgambelluri, F. Cugini, and P. Castoldi, "Network telemetry streaming services in SDN-based disaggregated optical networks," *J. Lightw. Technol.*, vol. 36, pp. 3142–3149, Aug. 2018.

[35] M. Anand, R. Subrahmaniam, and R. Valiveti, "POINT: An intent-driven framework for integrated packet-optical in-band network telemetry," in *Proc. of ICC 2018*, pp. 1–6, Jun. 2018.

[36] L. Gong, X. Zhou, W. Lu, and Z. Zhu, "A two-population based evolutionary approach for optimizing routing, modulation and spectrum assignments (RMSA) in O-OFDM networks," *IEEE Commun. Lett.*, vol. 16, pp. 1520–1523, Sept. 2012.

[37] W. Lu and Z. Zhu, "Dynamic service provisioning of advance reservation requests in elastic optical networks," *J. Lightw. Technol.*, vol. 31, pp. 1621–1627, May 2013.

[38] M. Zhang, C. You, H. Jiang, and Z. Zhu, "Dynamic and adaptive bandwidth defragmentation in spectrum-sliced elastic optical networks with time-varying traffic," *J. Lightw. Technol.*, vol. 32, pp. 1014–1023, Mar. 2014.

[39] Flexgrid High Resolution Optical Channel Monitor (OCM). [Online]. Available: https://www.finisar.com/roadms-wavelength-management/focm01fxc1mn

[40] SmartNIC. [Online]. Available: https://www.sigarch.org/the-new-life-of-smartnics/

[41] Libpcap. [Online]. Available: https://www.tcpdump.org/