Entropy-driven Adaptive INT and Its Applications in Network Automation of IP-over-EONs

Zichen Xu, Shaofei Tang and Zuqing Zhu, *Senior Member, IEEE*

(Invited Paper)

***Abstract*—Recently, IP over elastic optical network (IP-over-EON) has become a promising architecture for metro and core networks. This work studies how to visualize both layers of an IP-over-EON in real time, at different granularities (*e.g.*, at flow-level, lightpath-level, and link-level), and with self-adaptivity. Specifically, we consider the multilayer application of in-band telemetry (INT) and propose entropy-driven adaptive INT (namely, EntropyINT). We introduce stateful processing to programmable data plane (PDP) switches for EntropyINT, such that they can make local decisions to determine whether and what type of telemetry data about the IP and EON layers should be encoded in each packet. The local decisions are designed to be based on the amount of information that can be conveyed by telemetry data to the network automation system. Meanwhile, we make EntropyINT cooperate with out-of-band monitoring, to detect and locate exceptions in the EON layer. Our proposal is implemented in a real-world testbed of IP-over-EON, to evaluate its assistance to network automation. Experimental results verify the effectiveness of our proposal, and indicate that the telemetry data collected by EntropyINT and out-of-band monitoring can better assist the machine learning in network automation, for status prediction and anomaly detection.**

***Index Terms*—In-band telemetry (INT), In-network computing, Stateful processing, Optical performance monitoring, Machine learning, IP over elastic optical networks (IP-over-EONs).**

I. Introduction

OVER the past two decades, the booming of new network services has stimulated fast deployment of datacenters (DCs) globally [1]. This applied high pressure on the infrastructures of metro and core networks, and thus motivated many technical advances in this area [2, 3]. First of all, flexible-grid elastic optical networks (EONs) [4–7] have been architected to reshape the optical layer of metro and core networks, such that dynamic and fine-grained spectrum management can be enabled to adapt to the bandwidth demands of upper-layer applications seamlessly [8]. Then, the multilayer architecture of IP-over-EON [9, 10] has been widely considered to support the ever-growing IP-based network services well, *i.e.*, not only facilitating efficient resource utilization when carrying highly dynamic IP traffic, but also saving costs for operators [11].

Besides its advantages, IP-over-EON also brings in challenges, especially for network control and management (NC&M). This is because to support highly dynamic traffic with various quality-of-service (QoS) requirements, the IP and EON layers need to be managed jointly [10]. Moreover, the spreading of virtualization technologies (*i.e.*, virtual network slicing [12–14] and network function virtualization (NFV) [15, 16]) in metro and core networks has made them more adaptive and programmable at the cost of increased complexity. These challenges make it more difficult for the NC&M of IP-over-EONs to troubleshoot and restore from exceptions timely [17, 18]. The need of managing IP and EON layers jointly suggests that the NC&M should be centralized (*i.e.*, based on software-defined networking (SDN) [19]). Specifically, with SDN, one can use centralized controller(s) to manage the packet switches and optical elements in an IP-over-EON coordinately [20].

Nevertheless, a centralized NC&M scheme based on SDN is just the first step toward operating IP-over-EONs effectively and efficiently. More importantly, we need a network monitoring technique that can visualize the complex structure of an IP-over-EON in a fine-grained and real-time manner, such that exceptions can be detected, localized, and recovered quickly. Previously, people have developed many techniques to monitor packet/optical networks. For instance, SNMP [21], sFlow [22], Netflow [23], and RMON [24] can collect the statistics of network elements (NEs) in the IP layer, while techniques based on time or/and spectrum domain analysis have been developed to monitor the performance of the optical layer [25]. However, as these techniques let NEs report status data periodically or on-demand in the out-of-band way, they cannot precisely catch the real-time status of a dynamic network or reveal the end-to-end operation of an arbitrary flow.

Recently, with momentum gained from the fast development of programmable data plane (PDP) [26, 27], in-band network telemetry (INT) [28] was proposed to facilitate operators to visualize their networks in a programmable, real-time and per-flow manner. Specifically, according to the instructions from the control plane, each PDP switch on a flow's routing path records its own real-time status to encode and insert as specific INT fields in the packets of the flow. Hence, how the flow gets processed from source to destination can be reconstructed with the INT fields, to monitor its performance in a per-packet and per-hop way. Although INT was initially designed for packet networks, its unique benefits make it promising for IP-over-EONs too. Therefore, in our previous studies [29, 30], we have extended INT to design and experimentally demonstrate multilayer INT (ML-INT) systems that could visualize both layers of an IP-over-EON in real time.

However, there is still a technical gap to close, if the NC&M system of an IP-over-EON wants to leverage ML-INT to collect the telemetry data for network automation. This is because encoding the telemetry data of all the electrical/optical NEs on a flow's path as INT fields and inserting them in packets would lead to excessive bandwidth overheads in the IP-over-EON and stressful processing burden on the data analyzers at the

Z. Xu, S. Tang, 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).

network edge. Although the current implementations of ML-INT have already tried to address this issue by sampling the packets in a flow to insert INT fields (*i.e.*, selective INT field insertion [31]), there are still two unsolved problems. Firstly, in a dynamic network environment, it would be difficult for the operator to determine the INT sampling rate such that the tradeoff between INT overheads and monitoring accuracy can be balanced well. Secondly, the INT sampling rate needs to be adjusted on-the-fly to adapt to dynamic traffic condition, while relying on the SDN controller to do this task will not only induce noticeable latency but also increase the communication overheads between the control and data planes.

In this work, we propose entropy-driven adaptive INT (EntropyINT) to close the aforementioned technical gap. The idea is to introduce stateful processing to the PDP switches for ML-INT, such that they can make local decisions (*i.e.*, without involving the controller) to tell whether and what types of INT fields should be inserted in each packet. Specifically, we design the PDP switches to make local decisions based on the "information content" of the telemetry data (*i.e.*, the amount of information conveyed by the data to NC&M), and an INT field of telemetry data will only be inserted in a packet when the information content of the data is sufficiently large.

We first lay out the system design to realize the stateful processing for EntropyINT on PDP switches. Then, we implement our proposal, equip the EntropyINT-enabled PDP switches in a real-world network testbed of IP-over-EON, and conduct experiments to evaluate its assistance to network automation. Our experimental results indicate that with EntropyINT, the critical data points regarding the statistics of electrical/optical NEs can be caught more accurately and adaptively with reduced INT overheads. The results also suggest that the telemetry data collected by EntropyINT can better assist the machine learning in network automation, for status prediction and anomaly detection. Finally, we make EntropyINT cooperate with out-of-band monitoring, and verify that it can help the controller to only trigger out-of-band monitoring when necessary, *i.e.*, troubleshooting becomes more accurate with less overhead.

The rest of the paper is organized as follows. Section II briefly surveys the related work. We present the design of EntropyINT in Section III, while its implementation for the network automation of an IP-over-EON is discussed in Section IV. The experimental demonstrations are described in Section V. Finally, Section VI summarizes this paper.

II. Related Work

Due to its advantages, INT has spurred intense research activities since its inception [32], quickly followed by standardization [28]. As a PDP-enabled network monitoring technique, INT can be implemented based on programming protocol-independent packet processor (P4) [26] and protocol-oblivious forwarding (POF) [27]. However, the implementations of INT in early days (*e.g.*, in [33]) only targeted packet networks and inserted INT fields on per-packet basis. Then, based on this per-packet INT scheme, the studies in [34, 35] formulated optimizations to plan the paths of INT-enabled flows such that the coverage of monitoring can be maximized from the network perspective. Although they did route the INT-enabled flows to avoid collecting redundant telemetry data, they did not try to optimize INT itself to minimize the overheads.

In [36, 37], people proposed several schemes to filter out redundant telemetry data, for reducing the processing burden of data analyzers, but they did not try to reduce the bandwidth overheads of INT. To relieve the bandwidth overheads of INT, researchers have considered to sample packets or/and types of telemetry data for INT field insertions [31, 38, 39]. For instance, the PINT in [39] leveraged probability sampling to select the type of telemetry data to include in the INT header of each packet, and reduced the bandwidth overheads of INT effectively. Note that, for these selective INT schemes with sampling, how to determine the INT sampling rate and adjust it dynamically according to network status is the key to balance the tradeoff between network monitoring accuracy and INT overheads. Nevertheless, to the best of our knowledge, this problem has not been solved properly by existing studies.

Multilayer INT schemes for packet-over-optical networks have been designed and demonstrated in [29, 40, 41]. However, they still did not properly address the bandwidth overheads of INT. In [30, 42], we leveraged machine learning to analyze the telemetry data collected by ML-INT and utilized the results for network automation. More specifically, in [42], we showed that the INT sampling rate can be adjusted according to the predicted bandwidth usage of a lightpath, such that the bandwidth overhead of INT does not result in lightpath congestion. However, the INT sampling rate was adjusted by the SDN controller, and the information entropy of telemetry data was not considered.

III. System Design

In this section, we will describe the design of EntropyINT, including the packet format and operations on PDP switches.

A. Network Model and Design Considerations

In this work, we assume that the IP-over-EON is a metro or core network. Therefore, each flow in it is actually obtained by aggregating a number of IP flows between a pair of PDP switches in the IP layer, and with the multi-protocol label switching (MPLS), each PDP switch can identify the flow by checking its MPLS labels. In the EON layer, a flow is carried by a sequence of lightpaths for being forwarded from source to destination. As each lightpath can carry multiple flows, we assume that the operator can limit the maximum bandwidth usage of a flow, and if there is no restriction applied, a flow's maximum bandwidth usage is the smallest bandwidth capacity of the lightpaths that carry it (*i.e.*, the bottleneck lightpath).

As each flow is an aggregated one, its data-rate can be relatively high (*e.g.*, 1 Gbps and beyond), which means that the time interval between two adjacent packets will be very short. This implies that monitoring the flow's performance with INT in the per-packet and per-hop manner might not be necessary [31]. Therefore, sampling packets or/and types of telemetry data for INT field insertions is a promising way to reduce not only the bandwidth overheads of INT but also the data processing burden on data analyzers. However, the challenging problem here is how to determine the INT sampling rate and adjust it adaptively according to network status. This motivates us to design EntropyINT, which selects and inserts telemetry data in packets according to the data's importance of being collected. Specifically, to quantify the importance of telemetry data, we define it as the information content of the data based on the probability of the data's occurrence. In other words, we assume that telemetry data that occurs with lower probability should be reported with higher probability. With these considerations, EntropyINT can focus on important telemetry data automatically and balance the tradeoff between network monitoring accuracy and INT overheads adaptively.

B. Design of Packet Header

Similar to the ML-INT developed in [29], EntropyINT also selects the packets in a flow to insert INT fields that contain the telemetry data about real-time status of the electrical/optical NEs on the flow's routing path. Therefore, with EntropyINT, there will be two types of packets in an IP-over-EON, *i.e.*, the **normal packets** that are not selected for INT field insertion, and the **INT packets** that carry INT fields.

The packet format of an INT packet is shown in Fig. 1, which is adapted from the standard INT packet format defined in [28] with only minor modifications for ensuring the compatibility between EntropyINT and other INT versions. For each INT packet, EntropyINT only encodes and inserts one INT header in between its IP header and payload. The INT header consists of four fields, which are *Type*, *MapInfo*, *DeviceID*, and *Metadata*. If a packet is selected by a PDP switch in the IP layer for EntropyINT, the INT header will be inserted by the switch and it will not be removed before the packet exits the IP-over-EON. Note that, the packet format in Fig. 1 actually suggests that the INT header of an INT packet has a fixed length of 12 bytes, *i.e.*, the INT header only contains one type of telemetry data about a single hop.

The definitions of the fields in the INT header are explained as follows. *Type* has a length of 2 bytes and is filled with $0x0908$ to distinguish INT packets from normal ones. *MapInfo* is a 2-byte bitmap to identify the type of telemetry data that is encoded in the subsequent *Metadata* field. In this work, we support five types of telemetry data (as shown in Fig. 1), each of which is represented by a bit in *MapInfo*. Here, *Out Port* and *In Port* denote the input and output ports of the switch that the packet uses, respectively, *Hop Latency* is the processing latency of the packet on the switch, *Bandwidth* is the bandwidth usage on the output port of the packet, and *BER* is the bit-error-rate (BER) of the lightpath that carries the packet to the switch. Hence, we can see that among the five types of telemetry data, *Out Port*, *In Port*, *Hop Latency*, and *Bandwidth* are about the IP layer and they are directly collected by PDP switches, while *BER* tells the status of the EON layer, and it is first collected by a bandwidth-variable transponder (BV-T) and then forwarded to the PDP switch that is local to the BV-T. *DeviceID* contains the unique 4-byte ID of the PDP switch that inserts the INT header. Note that, we can assign any 4-byte value as the ID of a PDP switch, as long as it is unique in the IP-over-EON to identify the PDP

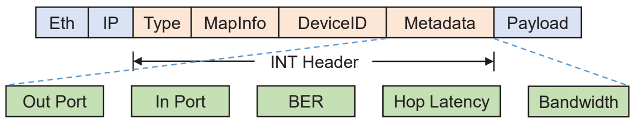


Fig. 1. Format of an INT packet in EntropyINT.

switch. Finally, the telemetry data with the type flagged in *MapInfo* is encoded in *Metadata*, which is also a 4-byte field.

The rationale behind the empirical setting that EntropyINT only allows an INT header to contain one type of telemetry data regarding a single hop is two-fold. Firstly, the setting minimizes the per-packet INT overheads, and thus it can minimize the probability that an INT packet is excessively long and exceeds the maximum transmission unit (MTU) of the IP-over-EON. Secondly, since an IP-over-EON is normally built for a metro or core network, the flows in it usually have relatively high data rates (*e.g.*, hundreds of Mbps or even higher) [43]. This suggests that their packet rates can easily reach one million packets per second (Mpps). However, the real-time statistics of electrical/optical NEs usually would not change dramatically at the scale of microseconds. Therefore, although the amount of telemetry data carried by each INT packet is minimized, EntropyINT can distribute different types of telemetry data regarding all the concerned NEs over a series of INT packets, while the time accuracy of network monitoring will not be degraded noticeably. We will further verify this argument in Section V with experimental results.

C. Operation of EntropyINT

Different from the existing selective INT schemes, EntropyINT does not select INT packets evenly according to a preset INT sampling rate, since such a scheme is not adaptive enough. Instead, it determines whether to insert an INT header in a packet and which type of telemetry data to encode in the *Metadata* of the INT header locally, based on the information content of the collected telemetry data in different types.

Note that, as the packet sampling of EntropyINT might not be uniform, we should first determine the upper-limit of the INT sampling or the smallest interval between two consecutive INT packets. Then, the bandwidth overheads introduced by EntropyINT will not make the flow being monitored use more bandwidth than the capacity that was allocated to it. We denote the flow being monitored as f_i, where i is its unique index. Then, the maximum number of INT packets that can be selected for f_i per second within the future period of τ is

$$N_{i,\tau} = \left\lfloor \frac{\min\left[(C_i - \hat{b}_{i,\tau}) \cdot \eta, \hat{B}_{\text{INT}}\right]}{L_{\text{INT}}} \right\rfloor, \tag{1}$$

where C_i denotes the maximum bandwidth capacity that f_i can use, $\eta \in (0, 1)$ is a preset parameter representing the ratio of the available bandwidth that can be used by INT, $\hat{B}_{\text{INT}}$ is also a preset parameter, which tells the maximum bandwidth usage of INT allowed in any case, $\hat{b}_{i,\tau}$ is the peak bandwidth usage of f_i within the future period of τ, and L_{INT} is the length of the INT header in Fig. 1, which is $12 \cdot 8 = 96$ bits. Then, the smallest interval between two consecutive INT packets that can be selected by a PDP switch for f_i within τ is

$$\Delta t_{i,\tau} = \frac{hop(f_i)}{N_{i,\tau}}, \tag{2}$$

where $hop(f_i)$ is the hop-count of the routing path of f_i (*i.e.*, the number of PDP switches that it passes through). Here, we introduce $hop(f_i)$ in Eq. (2) because an INT header in Fig. 1 can only carry one type of telemetry data regarding a single hop. In other words, the bandwidth allocated to INT should be evenly distributed to the hops of f_i. Note that, to get $\hat{b}_{i,\tau}$, the SDN controller of an IP-over-EON can leverage either rough estimation or sophisticated traffic prediction (*e.g.*, using a long short term memory (LSTM) module [42]).

With the the smallest sampling interval obtained with Eqs. (1) and (2), a PDP switch determines which type of telemetry data should be encoded in one INT packet according to the information content of the collected data. Specifically, the value of a type of telemetry data regarding an electrical or optical NE can be modeled as a random variable $X_{j,k}$, where j denotes the type of the telemetry data and k is the unique index of the NE. Then, if we denote the probability that the telemetry data takes a value x as $P(X_{j,k} = x)$, the information content of the event "$X_{j,k} = x$" can be quantified as

$$I = -\log_2[P(X_{j,k} = x)]. \tag{3}$$

With Eq. (3), we obtain a way to quantitatively compare the importance of telemetry data in different types, which enables each PDP switch to make local decisions on whether to insert an INT header and which type of telemetry data to encode in its *Metadata*. Hence, telemetry data can be collected more adaptively with less bandwidth overheads, while the accuracy and timeliness of network monitoring will not be impacted significantly. Meanwhile, we hope to point out that in order to maintain its throughput, a PDP switch usually should not perform complex computing when processing packets. Therefore, we introduce an information mapping table (IMT) to convert the calculation in Eq. (3) into a simple table lookup. Fig. 2 shows the layout of the IMT. Here, we assign the indices of the telemetry data for *Out Port*, *In Port*, *BER*, *Hop Latency*, and *Bandwidth* as $j = 1$ to 5, respectively.

First of all, the telemetry data for *Out Port* and *In Port* is semi-static for each flow, *i.e.*, the input and output ports used by a flow on one switch seldom change during its lifetime. Therefore, each type of the telemetry data (*i.e.*, *Out Port* or *In Port*) only has two entries in the IMT as

$$\begin{cases} I_j^{(1)} = -\log_2(1-\epsilon), & X_j \text{ is normal}, \\ I_j^{(2)} = -\log_2(\epsilon), & \text{otherwise}, \end{cases} \quad \forall j \in [1,2].$$

Here, X_j is the collected data for *Out Port* or *In Port*, and it is normal if the flow uses the expected output or input port, respectively. Meanwhile, we empirically assign a small probability of ϵ to denote the chance that the flow uses an output or input port other than the expected one. Ideally, the value of ϵ should be the average outage probability of a flow using unexpected output/input port at a switch. When the outage probability is not known, we can simply assign a small value (*e.g.*, $\epsilon = 0.001$) to make sure that the information content of such an outage event is reasonably large.

Out Port	*Normal:* $I_1 = -\log(1-\epsilon)$	*Otherwise:* $I_2 = -\log\epsilon$	——	————
IN Port	*Normal:* $I_1 = -\log(1-\epsilon)$	*Otherwise:* $I_2 = -\log\epsilon$	——	————
BER	$[x_1^{(0)}, x_1^{(1)})$: I_1	$[x_1^{(1)}, x_1^{(2)})$: I_2	$\cdots$	$[x_1^{(M-1)}, x_1^{(M)}]$: I_M
Hop Latency	$[x_2^{(0)}, x_2^{(1)})$: I_1	$[x_2^{(1)}, x_2^{(2)})$: I_2	$\cdots$	$[x_2^{(M-1)}, x_2^{(M)}]$: I_M
Bandwidth	$[x_3^{(0)}, x_3^{(1)})$: I_1	$[x_3^{(1)}, x_3^{(2)})$: I_2	$\cdots$	$[x_2^{(M-1)}, x_2^{(M)}]$: I_M

Fig. 2. Layout of information mapping table (IMT).

Secondly, the telemetry data for *BER* has been converted to its logarithmic form before being encoded in a *Metadata* field, *i.e.*, $X_3 = \lfloor -\log(\text{BER}) \rfloor$. Hence, the encoded telemetry data for *BER* is a positive integer, and it increases when the actual BER value decreases. Therefore, we allocate M entries in the IMT for *BER*, where the value of M is set empirically.

$$I_3^{(m)} = -\log_2(P_3^{(m)}), \ X_3 \in [x_3^{(m-1)}, x_3^{(m)}), \ \forall m \in [1, M],$$

where $x_3^{(m-1)}$ and $x_3^{(m)}$ are the lower-limit and upper-limit of the data in the m-th entry, we have $x_3^{(0)} = 0$, and $P_3^{(m)}$ is the probability that X_3 will fall in the range of $[x_3^{(m-1)}, x_3^{(m)})$. Note that, the probability distribution of $\{P_3^{(m)}, \ \forall m \in [1, M]\}$ can be obtained by analyzing the historical BER data of lightpaths in the IP-over-EON, and we assume that it is known.

Thirdly, the entries in the IMT for *Hop Latency* are obtained with a method similar to that of *BER*. The only exception is that the telemetry data for *Hop Latency* does not need to be encoded in its logarithmic form. Once again, the probability distribution of $\{P_4^{(m)}, \ \forall m \in [1, M]\}$ can be obtained by analyzing the historical packet processing latency of PDP switches in the IP-over-EON, and it is pre-known.

Fourthly, for the telemetry data for *Bandwidth*, we can allocate its entries in the IMT similar to those of *BER* and *Hop Latency*. However, the dilemma is that the bandwidth distribution of a flow (*i.e.*, $\{P_5^{(m)}, \ \forall m \in [1, M]\}$) might not always be known before we start monitoring the flow. Hence, we can first make each entry cover an equal range (*i.e.*, the value of $x_5^{(m)} - x_5^{(m-1)}$ is fixed) and initialize the probability distribution as $\{P_5^{(m)} = \frac{1}{M}, \ \forall m \in [1, M]\}$. Then, during the flow's lifetime, the controller updates the entries from time to time, to make the values of $\{x_5^{(m)}\}$ and $\{P_5^{(m)}\}$ adapt to the historical traffic data of the flow[1]. Fig. 3 shows the bandwidth distribution of a flow that we collected in an experiment with EntropyINT. Meanwhile, we hope to point out that in addition to $\{x_5^{(m)}\}$ and $\{P_5^{(m)}\}$, the controller can also adjust other settings of the IMT during the operation of a flow if necessary.

Finally, the overall procedure of EntropyINT can be summarized as follows. The controller first calculates the smallest interval $\Delta t_{i,\tau}$ for INT sampling with Eqs. (1) and (2), and provides it to the switches on the routing path of f_i. Then,

[1]Note that, as we consider the IP-over-EON as a metro or core network, each flow in it is actually an aggregated one by grooming many IP flows between a pair of switches in the IP layer. Therefore, the dynamic traffic on a flow can fluctuate with a predictable pattern [43] and last for a reasonably long period of time (*e.g.*, in hours or even days).

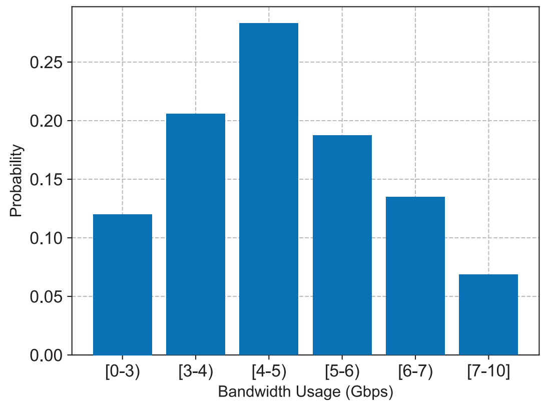


Fig. 3. Example on bandwidth distribution of a flow.

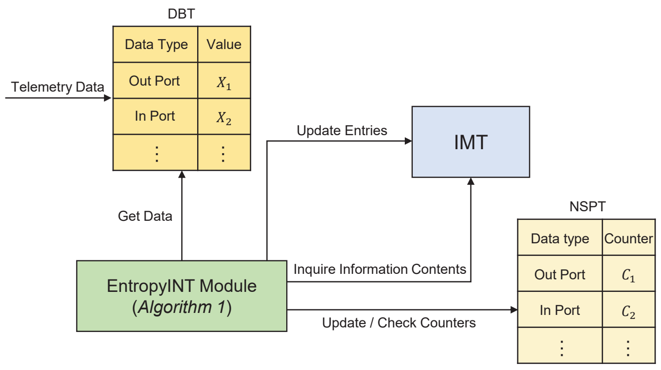


Fig. 4. Operation principle of EntropyINT.

according to $\Delta t_{i,\tau}$, each related switch performs **prioritized INT**, *i.e.*, it checks the IMT to find the telemetry data whose information content is the largest and encodes the data in the INT header of a packet. Meanwhile, to avoid the situation in which certain types of telemetry data will not be collected for a long time due to their relatively small information content, we also incorporate **polling-based INT**. Specifically, the polling-based INT defines a polling interval on each switch, and when a polling interval expires, the switch finds the type of telemetry data, which has been collected the least frequently during the polling interval, and encodes it in the INT header of a packet.

D. Procedure on PDP Switches

To realize the aforementioned principle of EntropyINT, we introduce an EntropyINT module and three tables in each PDP switch (as shown in Fig. 4). In addition to the IMT, the other two tables are the data buffering table (DBT) and the no-starving polling table (NSPT). As shown in Fig. 4, the DBT stores the most recent value of each type of telemetry data, and the value will be updated constantly by the PDP switch. Then, when a new packet arrives, the prioritized INT can leverage the DBT and IMT to find the telemetry data whose information content is the largest and encode it in the INT header of the packet. On the other hand, the NSPT is for the polling-based INT and it stores the times that each type of telemetry data has been encoded in a packet in the current polling cycle.

Our designed procedure on PDP switches to facilitate EntropyINT is illustrated in *Algorithm* 1. *Lines* 1-3 are for the system initialization. When each packet of a flow that is being monitored enters a PDP switch, the switch will collect the telemetry data for *Out Port*, *In Port*, *BER*, *Hop Latency*, and *Bandwidth* instantly, use the values of the telemetry data to check the IMT to get their information contents, compare the information contents with those of the existing telemetry data in the DBT, and update the corresponding entry in the DBT if the information content of the new data is larger (*Lines* 4-5). The reason for performing the data preparation with the DBT is that we would like to record and report critical telemetry data points as many times as possible. Note that, the introduction of the DBT can make an INT packet carry the telemetry data, which was actually collected before the packet entering a PDP switch, and thus induces certain time error for network monitoring. However, as the packet rates of the flows considered in this work are relatively high (*i.e.*, $\sim$1 Mpps or even higher), the time error will only be in the scale of microseconds and thus tolerable.

Algorithm 1: Procedure of EntropyINT on PDP Switches

```
1  get smallest INT sampling interval with Eqs. (1) and (2);
2  initialize IMT according to instructions from controller;
3  initialize DBT and NSPT;
4  while one packet of a flow being monitored comes in do
5      collect telemetry data, get information contents from
       IMT, and update DBT if necessary;
6      if the packet is not an INT packet then
7          if it is time for prioritized INT then
8              check IMT to find the telemetry data in DBT,
               which has the largest information content;
9              encode the telemetry data in an INT header
               to insert in the packet;
10             reset all the entries in DBT;
11             update corresponding counter in NSPT;
12         else
13             if it is time for polling-based INT then
14                 check NSPT to select the telemetry data
                   type with the least insertions;
15                 encode the telemetry data of selected type
                   in an INT header to insert in the packet;
16                 reset all the entries in DBT;
17                 update corresponding counter in NSPT;
18             end
19         end
20     end
21     send the packet out;
22     update IMT if asked by controller;
23 end
```

According to the design in Section III-B, we will only insert one type of telemetry data regarding a single hop in an INT packet. Therefore, *Line* 6 checks whether the packet already contains an INT header, and will only insert a new INT header in it if not. Then, in *Lines* 7-11, we explain how to perform prioritized INT insertion based on the information content of telemetry data. Specifically, with DBT and IMT, we can find the telemetry data that has the largest information content and

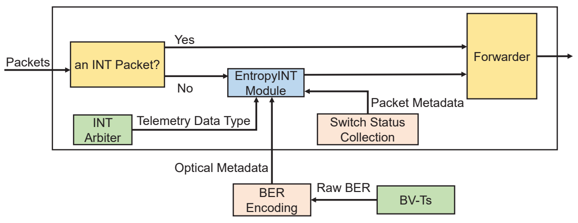


Fig. 5. System configuration of a PDP switch for EntropyINT.

encode it in an INT header to insert in the packet (*Lines* 8-9). Then, after the telemetry data of the selected type has already been encoded in an INT packet, all the entries in the DBT are reset (*Line* 10). Next, we update the counter of the data type in the NSPT (*Line* 11). As shown in Fig. 4, we maintain a counter for each type of telemetry data in the NSPT, to record the times that the type of telemetry data has been encoded in an INT packet. The rationale behind the NSPT is to avoid the situation in which certain type(s) of telemetry data are not encoded in INT packets for a relatively long time.

Lines 13-18 explain how to use the NSPT to conduct polling-based INT insertion, for avoiding the starving of one or more telemetry types in EntropyINT. Specifically, the polling-based INT is triggered periodically by the PDP switch, and instead of selecting the telemetry data whose information content is the largest, it chooses the telemetry data whose type has the least INT insertions until now (*Line* 14). Then, the remaining operations are the same as those in the prioritized INT insertion. Finally, *Lines* 21-22 finish the packet forwarding.

Fig. 5 shows the system configuration of a PDP switch that enables EntropyINT. The INT arbiter suggests the types of telemetry data to get to the EntropyINT module. As for the types of telemetry data, those about IP layer are provided by the PDP switch itself through collecting local statistics, and that about EON layer is obtained from the optical performance monitors on the BV-Ts that locally connect to the PDP switch.

IV. System Implementation

We implement our proposal of EntropyINT in an IP-over-EON to realize closed-loop network automation. The architecture of the system implementation is shown in Fig. 6, which is similar to the architecture that we designed in [42].

A. Data Plane

The data plane consists of an IP layer and an EON layer. There are three key NEs in the IP layer, *i.e.*, the end hosts, PDP switches, and data analyzers (DAs). The end hosts send/receive application traffic, which is groomed as aggregated flows by PDP switches with a label-based mechanism (*e.g.*, by assigning MPLS labels to them). Then, in the IP-over-EON, PDP switches can identify the flows by checking their labels. The PDP switches support the features of EntropyINT discussed in the previous section, and in this work, we realize the PDP switches by extending the POF-enabled OpenvSwitch (OVS-POF) developed in our previous work [31]. Hence, each PDP switch is actually a software-based switch running on a standalone Linux server, and it can achieve a packet processing rate of over 2 Mpps per port, regardless of the packet sizes.

The DA is designed to analyze the telemetry data collected with EntropyINT and reports digested network status information to the controller. Specifically, the flow-level monitor parses, extracts and decodes the telemetry data encoded in the INT headers of INT packets, and records the data in the data storage, and in the meantime, the data is analyzed by the deep learning (DL) based data analytics module, for future status prediction and anomaly detection. Then, when an anomaly is detected, it is sent to the exception handler, which will forward it to the controller through the EntropyINT monitoring channel for further processing. Note that, as INT packets are mirrored to DAs by the egress switches of flows, DAs are placed at the edge of the IP layer, together with end hosts. Therefore, the DAs actually conduct distributed data analytics to accomplish real-time and flow-level monitoring and troubleshooting, which offloads a significant part of NC&M tasks from the controller. Each DA is home-made and runs on a Linux server, and it can achieve a packet processing rate of over 2 Mpps.

The role of the EON layer is to set up and manage lightpaths to bridge the communications among PDP switches, and the key optical NEs there are the BV-T, optical line system (OLS), and bandwidth-variable wavelength-selective switch (BV-WSS). On each optical port of the BV-Ts and BV-WSS', we place an optical performance monitor (OPM) to collect its real-time statistics. The OPMs operate based on either spectrum or time-domain analysis. For instance, with spectrum analysis, the power, optical signal-to-noise ratio (OSNR), and optical spectrum on a port can be obtained for link-level monitoring, while the time-domain analysis on a BV-T can get the pre-forward-error-correction (pre-FEC) BER of a lightpath.

Since pre-FEC BER is the ultimate metric to judge the health of a lightpath, we let BV-Ts report it to their local PDP switches for being collected by EntropyINT. As for the statistics obtained with spectrum analysis, we design an out-of-band mechanism to report them to the controller, which will be elaborated in Section IV-C. Hence, our implementation can not only get the key metrics of lightpaths timely but also save the INT overheads used for reporting the status of optical NEs.

B. Control Plane

The control plane essentially consists of the centralized SDN controller, which is developed based on ONOS [44] and can manage the electrical/optical NEs in the IP and EON layers. In the controller, the traffic engineering database (TED) records the multilayer service provisioning schemes of traffic flows and works with the network control and network monitor modules to facilitate network automation. We design three modules in the network monitor to realize self-adaptive monitoring and troubleshooting. Specifically, the report handler receives and analyzes the reports from DAs to track the results of flow-level monitoring, the link-level monitor coordinates the OPMs in the EON layer and collects the statistics regarding optical NEs in the out-of-band manner (*i.e.*, through the out-of-band monitoring channel), and the telemetry orchestrator arranges the network monitoring schemes in the IP-over-EON.

Therefore, both the report handler and the telemetry orchestrator in the network monitor can interact with the network

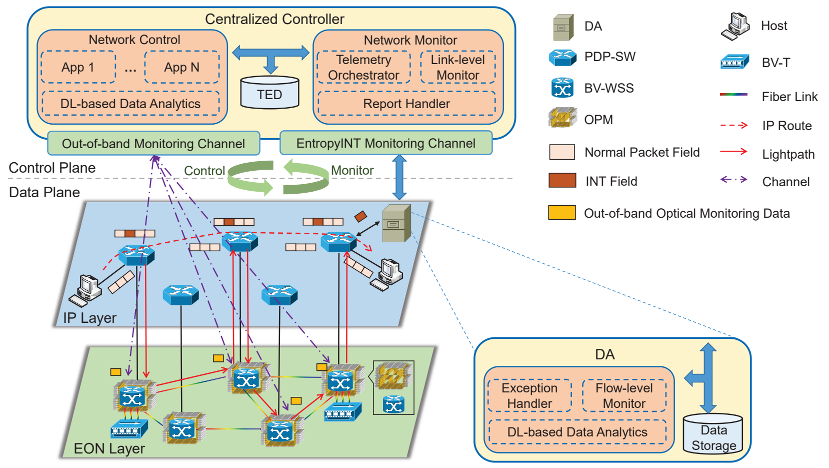


Fig. 6. Architecture of system implementation. TED: Traffic engineering database, DL: Deep learning, DA: Data analyzer, PDP-SW: PDP switch, BV-T: Bandwidth-variable transponder, BV-WSS: Bandwidth-variable wavelength-selective switch, OPM: Optical performance monitor.

control module. The report handler will forward the exceptions that it detects and locates in the IP-over-EON to the network control module, which will then adjust the multilayer provisioning schemes of the affected flows to restore their QoS parameters. On the other hand, the telemetry orchestrator consistently analyzes the multilayer provisioning schemes of flows, their QoS requirements, and the current settings and reports of their monitoring schemes (*i.e.*, including both EntropyINT and out-of-band monitoring), to determine whether the monitoring schemes need to be updated and how to update them accordingly. Then, if it is necessary, the telemetry orchestrator will suggest the network control module to update the monitoring schemes to adapt to the time-varying network status and the QoS demands of flows.

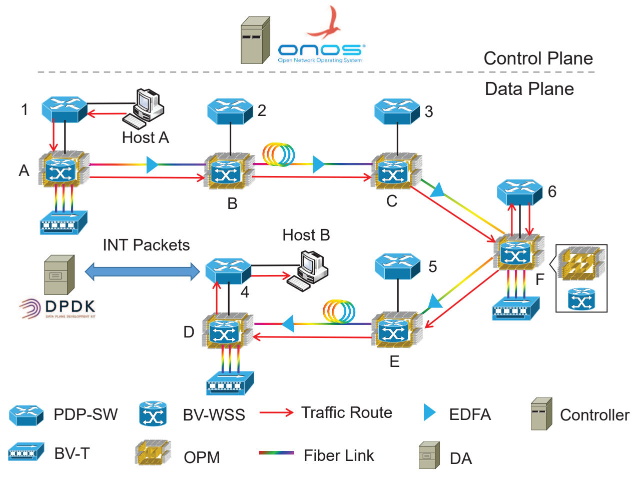


Fig. 7. Experimental setup.

C. Out-of-band Monitoring

Note that, with EntropyINT, the telemetry data regarding a lightpath (*i.e.*, pre-FEC BER) can only be collected at the BV-Ts generating/receiving it. Hence, EntropyINT can only monitor a lightpath in the end-to-end way, while the link-level monitoring that checks the lightpath's performance for each traversed fiber span might not be feasible. In other words, if a lightpath experiences multiple fiber spans, EntropyINT will overlook the intermediate optical switches because the local PDP switches of these optical switches are bypassed by the packets on the lightpath. However, in order to locate the exceptions in the EON layer, the controller needs to know the lightpath's performance on each traversed fiber span.

Therefore, we let the OPMs in the EON layer analyze the spectra of the signals on optical ports and report the results to the controller in the out-of-band manner. Then, the EON layer can be monitored comprehensively. Specifically, each OPM collects and analyzes the signal on an optical port at regular intervals (*i.e.*, one second), and stores the results (*i.e.*, the power, OSNR and optical spectrum) locally. When the BER results collected by EntropyINT suggest an exception in the EON layer, the controller will request the latest monitoring results regarding all the fiber spans that the concerned lightpath traverses from the corresponding OPMs. Hence, the out-of-band monitoring is triggered on an on-demand basis for locating exceptions in the EON layer. This not only saves the communication overheads between the data and control planes, but also reduces the data processing burden on the controller. In all, our system implementation in Fig. 6 combines in-band and out-of-band monitoring and leverages various techniques that can examine an IP-over-EON at different granularities (*e.g.*, at flow-level, lightpath-level, and link-level), to ensure the efficiency and effectiveness of network monitoring.

V. Experimental Demonstrations

In this section, we present the experiments that demonstrate the capabilities of our network automation system based on EntropyINT.

A. Experimental Setup

The proof-of-concept implementation of our network automation system is realized in the IP-over-EON testbed shown in Fig. 7. The EON layer consists of six Finisar 1×9 BV-WSS' that can switch optical spectrum at a granularity of 12.5 GHz, fiber spans with inline erbium-doped fiber amplifiers (EDFAs), and an OLS system based on Juniper BTI7800 platform. The OLS system deploys three BV-Ts on *Nodes A*, *D* and *F* in the testbed, respectively, each of which can generate/receive a lightpath with line-rate in the range from 100 to 400 Gbps and monitor the pre-FEC BER of the lightpath in real time. Meanwhile, we equip a commercial optical channel monitor (OCM), which can scan the whole C-band with a resolution of 312.5 MHz within hundreds of milliseconds, on each optical port in the EON layer to enable the out-of-band monitoring.

As for the IP layer, we have two end hosts, six PDP switches and a DA, all of which equip 10 GbE ports. Each end host is emulated with a software-based traffic generator/analyzer [45], which can generate/receive packets at a data-rate up to 10 Gbps. Each PDP switch is based on a software-based switch based on OVS-POF, and it connects its 10 GbE ports to the client side of OLS. The DA is a home-made software system.

In the experiments, we let *Host A* send dynamic traffic flows, whose data-rates range between 2 and 10 Gbps, consistently with the traces selected from [46], to *Host B*. As shown in Fig. 7, the traffic from *Host A* to *Host B* is carried by two lightpaths, which go through the BV-WSS' in sequence as *A-B-C-F* and *F-E-D*. Hence, the multilayer provisioning scheme of the traffic flow uses the two lightpaths in the EON layer, and gets the flow switched in the IP layer by *PDP-SW 6*.

B. Feature Validation

In Section III-C, we explained that the smallest sampling interval of EntropyINT ($\Delta t_{i,\tau}$) can be calculated with Eqs. (1) and (2), where the peak bandwidth usage of the monitored flow f_i within the future period of τ (*i.e.*, $\hat{b}_{i,\tau}$) should be known to the SDN controller of the IP-over-EON. In the experiments, we make the controller collect the data-rate of the dynamic flow from *Host A* at its ingress switch (*PDP-SW 1*) and leverage an LSTM module to predict its future traffic[2]. Fig. 8 shows the prediction results of the LSTM module that has been trained with the traces in [46]. It can be seen that the predicted traffic trace can approximate the real one well. Then, we run the experiments for $16,000$ traffic samples, and the experimental results indicate that the prediction accuracy is 92.33% and the average value of the relative errors is 4.5%. Therefore, the results confirm that it is feasible to use an LSTM model to get $\hat{b}_{i,\tau}$ and in turn estimate $\Delta t_{i,\tau}$ for EntropyINT accurately.

After obtaining the precise traffic prediction from the LSTM, the controller can periodically adjust the $\Delta t_{i,\tau}$ of EntropyINT adaptively. Based on the characteristics of the traces in [46], we make the controller update $\Delta t_{i,\tau}$ every 24 seconds in the experiments. The telemetry data on *Bandwidth* of the flow, which is collected by EntropyINT, is illustrated in Fig. 9(a). Here, we also perform an experiment with the adaptive INT scheme (AdaptiveINT) developed in [42], and make it use the packet format of EntropyINT in Fig. 1. However, as the PDP switches with AdaptiveINT cannot make local decisions to insert INT fields self-adaptively, they can only sample INT packets according to $\Delta t_{i,\tau}$. The results in Fig. 9(a) suggest that for this particular test case, both AdaptiveINT and EntropyINT can collect *Bandwidth* of the flow accurately, according to the $\Delta t_{i,\tau}$ provided by the controller.

[2]As the flow's data-rate varies according to a trace in [46], which was taken from a realistic metro/core network, it follows a predictable pattern [10, 43].

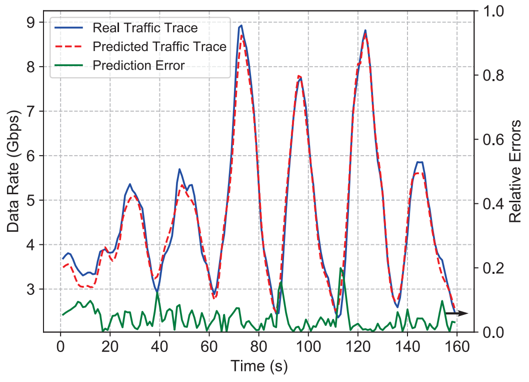
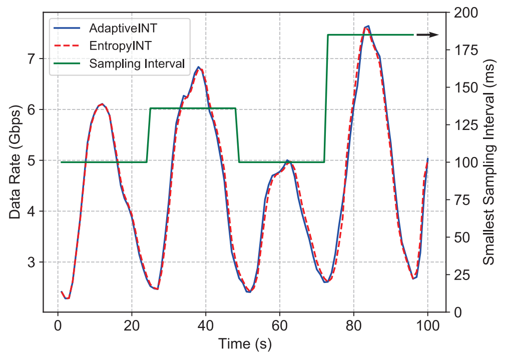
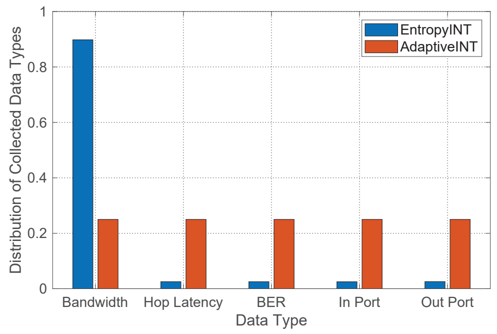


Fig. 8. Example on traffic prediction with LSTM module.



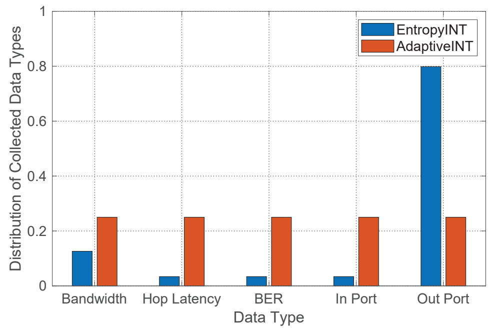
(a) Telemetry data on *Bandwidth* of flow



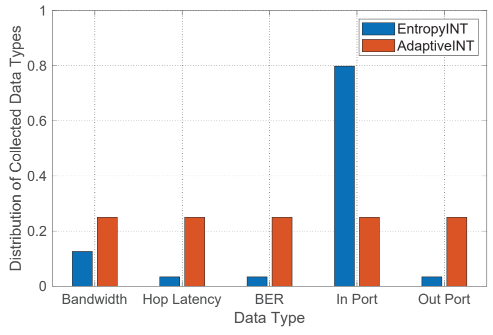
(b) Distribution of collected telemetry data types

Fig. 9. Network monitoring with AdaptiveINT and EntropyINT.

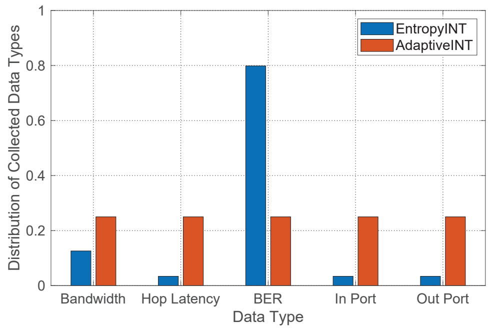
Note that, although the results on *Bandwidth* from AdaptiveINT and EntropyINT are similar in Fig. 9(a), they actually get the results differently, as indicated by the distribution of collected telemetry data types in Fig. 9(b). Specifically, AdaptiveINT just allocates INT packets to the five data types evenly because it does not enable stateful processing on each PDP switch for local decisions, while EntropyINT leverages intelligent local decisions to allocate most of the INT packets to collect *Bandwidth*, which is the telemetry data whose value changes the fastest and largest. In other words, as the experiments are conducted under normal network status, the telemetry data for *Out Port*, *In Port*, *BER*, and *Hop Latency*



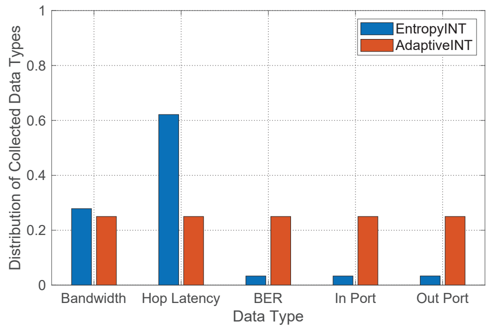
(a) *Out Port* is abnormal



(b) *In Port* is abnormal



(c) *BER* is abnormal



(d) *Hop Latency* is abnormal

Fig. 10. Distribution of collected data types in different test cases.

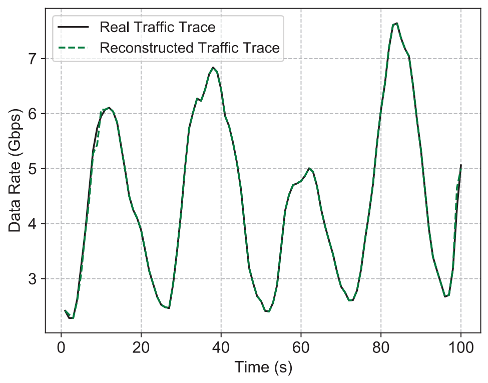


Fig. 11. Traffic trace reconstructed with telemetry data on *Bandwidth* when *Out Port* is abnormal.

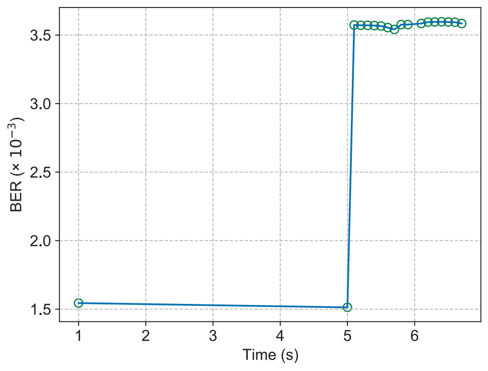


Fig. 12. BER collected by EntropyINT (at *PDP-SW 6*).

does not change as significantly as *Bandwidth*. Hence, according to our design of EntropyINT in Section III-C, the data samples of *Bandwidth* possess the most information contents, and thus they should be encoded in most of the INT packets.

Next, we introduce exceptions regarding *Out Port*, *In Port*, *BER*, and *Hop Latency* in the IP-over-EON, and redo the experiments, respectively. The new distributions of collected telemetry data types are shown in Fig. 10. We observe that EntropyINT indeed allocates most of the INT packets to the type of data whose values are abnormal. This is because compared with normal ones, abnormal values of telemetry data appear much less frequently in network monitoring and thus their information contents are much larger according to Eq. (3). Hence, by making local decisions based on information contents, the PDP switches with EntropyINT can focus their telemetry data collections on critical ones, to avoid redundant data that does not give much useful information to NC&M.

Meanwhile, it is interesting to notice that when *Hop Latency* is abnormal (in Fig. 10(d)), EntropyINT collects it less frequently than other abnormal data types (*i.e.*, *Out Port*, *In Port*, and *BER*). This is because *Hop Latency* normally changes faster and larger than *Out Port*, *In Port*, and *BER*, and thus when its value is abnormal, it carries smaller information content than the abnormal data for *Out Port*, *In Port*, and *BER*. The results in Figs. 10(a)-10(c) also suggest that in their test cases, EntropyINT only allocates $\sim$13% of the INT packets to collect *Bandwidth*. However, as the value of *Bandwidth* is changing fast, we plot the traffic trace reconstructed with such a small portion of INT packets in Fig. 11. It can be seen that the reconstructed trace still approximates the real one well. We also repeat the experiments for the test cases where *In Port* or *BER* is abnormal, and confirm that the results are similar.

Finally, we conduct an experiment to verify that EntropyINT can collaborate with out-of-band monitoring well for detecting exceptions in the EON layer. Specifically, when the flow from *Host A* to *Host B* is running, we deliberately misconfigure the

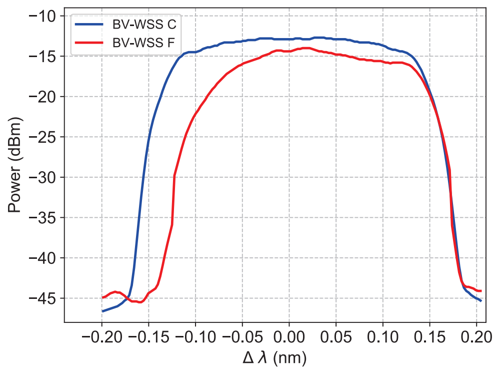


Fig. 13. Optical spectra collected with out-of-band monitoring.

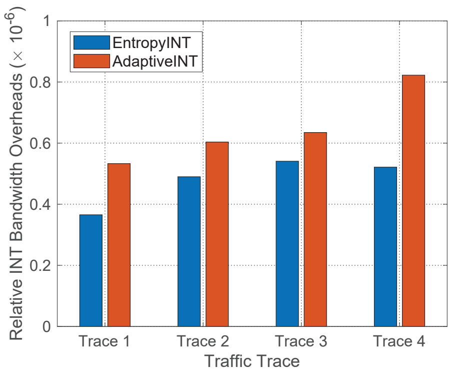


Fig. 14. Comparison of relative INT bandwidth overheads.

passband of *BV-WSS C* to cut off a small portion of the optical spectrum of the lightpath *A-B-C-F*. This makes the BER of the lightpath increase, which will be captured by EntropyINT immediately, as shown in Fig. 12. We can see that EntropyINT not only detects the abnormal BER timely but also makes the local decision to collect *BER* more frequently afterwards. Then, the DA will report the exception to the controller, which will in turn invoke the out-of-band monitoring to collect the optical spectra at the input ports of *BV-WSS' B*, *C* and *F*. This is because EntropyINT can only find that the lightpath's BER becomes abnormal, but cannot locate the root-cause in the EON layer. Hence, out-of-band monitoring should check the spectra of the lightpath at the BV-WSS' that it passes through in sequence, to locate the exception. Next, by analyzing the optical spectra (with a central wavelength of 1559.25 nm) and comparing those collected at *BV-WSS' C* and *F* (as illustrated in Fig. 13), the controller can easily determine that the exception actually happens at *BV-WSS C*.

C. Performance Benchmarking

In order to further compare the performance of AdaptiveINT and EntropyINT, we conduct more experiments. First of all, we would like to compare their INT bandwidth overheads. Specifically, we let *Host A* generate traffic according to four different traces in [46], turn on AdaptiveINT and EntropyINT, respectively, and measure the ratio of INT bandwidth overhead to total data transfers (*i.e.*, the **relative INT bandwidth overhead**). The results are shown in Fig. 14, which indicates that the relative INT bandwidth overheads of AdaptiveINT and EntropyINT are both smaller than 10^{-6} for all the traces, and EntropyINT always causes smaller INT bandwidth overhead.

Secondly, we compare the monitoring accuracy of AdaptiveINT and EntropyINT on fast-changing telemetry data. Here, we randomly introduce some traffic peaks, which last for very short periods of time, in the traffic from *Host A*. The traffic traces reconstructed with the telemetry data on *Bandwidth* from AdaptiveINT and EntropyINT are plotted in Fig. 15. We can see that EntropyINT precisely captures the short traffic peaks, while AdaptiveINT overlooks them. We also redo the experiments with a traffic trace that is much more dynamic, for a more general demonstration. Fig. 16 illustrates the results, which show the similar trend. Figs. 15 and 16 suggest that with intelligent local decisions, EntropyINT provides higher

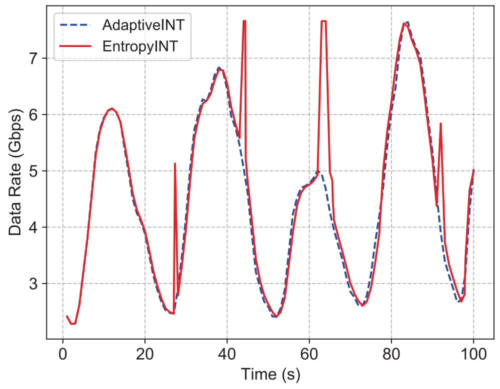


Fig. 15. Traffic traces reconstructed with telemetry data on *Bandwidth*.

monitoring accuracy than AdaptiveINT, especially for fast-changing telemetry data. Meanwhile, by jointly considering the results in Figs. 14-16, we can draw the conclusion that compared with AdaptiveINT, EntropyINT actually achieves higher monitoring accuracy with less INT overheads.

Finally, we evaluate the performance of the network automation enabled by AdaptiveINT and EntropyINT. Here, we consider two use-cases of the network automation, for the IP and EON layers, respectively. In the use-case for the IP layer, the network automation system leverages collected telemetry data to predict future bandwidth usage of the flow. Fig. 17 shows the prediction results from the LSTM module, based

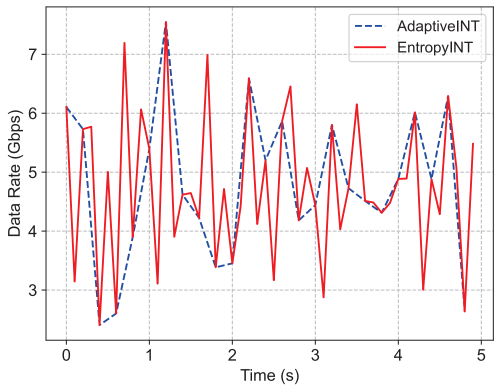
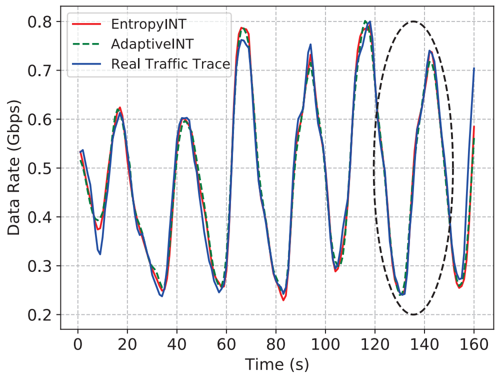
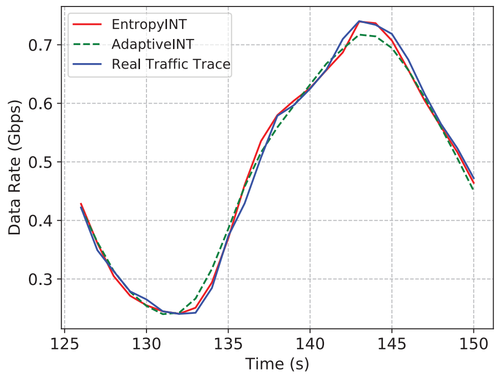


Fig. 16. Traffic traces reconstructed with highly-dynamic telemetry data on *Bandwidth*.



(a) Overall prediction results



(b) Zoomed-in view of the period of $[125, 150]$ seconds

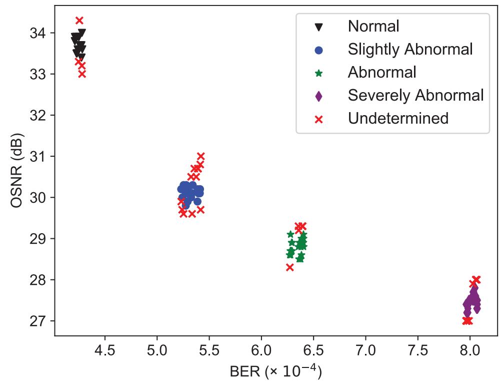
Fig. 17. Traffic traces predicted based on the telemetry data on *Bandwidth*, which was collected with AdaptiveINT and EntropyINT.

on the data on *Bandwidth* from AdaptiveINT and EntropyINT. We observe that the trace predicted based on the *Bandwidth* from EntropyINT can approximate the real one better. This further verifies the better monitoring accuracy of EntropyINT. We also run the experiments for $16,000$ traffic samples, and the results of the long-term predictions are listed in Table I, which indicate that both the prediction accuracy and the average value of relative errors from the experiments with EntropyINT outperform those with AdaptiveINT.

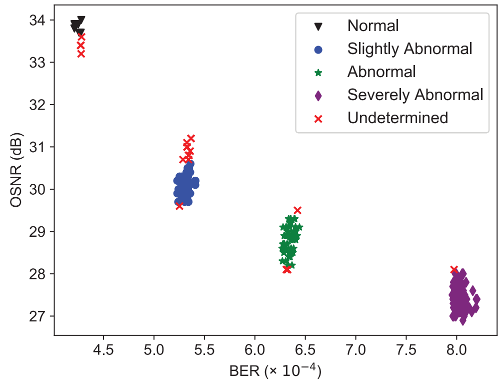
TABLE I
PERFORMANCE OF TRAFFIC PREDICTION

	Average Relative Error	Prediction Accuracy
EntropyINT	0.0435	92.67%
AdaptiveINT	0.0514	88.41%

In the use-case for the EON layer, the network automation system analyzes the telemetry data on *BER*, and combines it with the data on OSNR, which is collected with out-of-band monitoring, for lightpath status classification. Specifically, we try to classify the two-dimensional (2D) data of BER and OSNR with the density-based spatial clustering of application with noise (DBSCAN) [47]. The results on lightpath status classification are shown in Fig. 18. By comparing Figs. 18(a) and 18(b), we notice that the number of data samples collected by EntropyINT increases with the value of BER, which makes it easier to get the clusters for abnormal status and reduce the number of 2D data samples that cannot be classified. Specifically, for the data collected by EntropyINT and AdaptiveINT, the ratios of 2D samples that can be classified by DBSCAN are 92% and 74.8%, respectively. Note that, in real network operation, exceptions normally happen sparsely, which will make abnormal data samples only contribute to a very small portion of the collected telemetry data, and complicate the subsequent processing for data analytics [48, 49]. EntropyINT can relieve this issue by automatically collecting more abnormal data samples, and thus it can effectively improve the accuracy of status classification in network automation.



(a) Results of DBSCAN based on *BER* from AdaptiveINT



(b) Results of DBSCAN based on *BER* from EntropyINT

Fig. 18. Results on lightpath status classification.

VI. Conclusion

In this paper, we proposed a novel self-adaptive INT technique, namely, EntropyINT, to monitor IP-over-EONs. Specifically, we introduced stateful processing to PDP switches for ML-INT, such that they made local decisions to determine whether and what type of telemetry data about the IP and EON layers should be encoded in each packet. The local decisions were based on the information content of telemetry data. Meanwhile, we let EntropyINT cooperate with out-of-band monitoring, to detect and locate exceptions in the EON layer. Our proposal was implemented in a real-world network testbed of IP-over-EON, to evaluate its assistance to network automation. Experimental results confirmed that with EntropyINT, the critical data samples regarding the statistics of electrical/optical NEs could be caught more accurately and adaptively with reduced INT overheads. The results also suggested that EntropyINT cooperated with out-of-band monitoring well, and thus the telemetry data collected by them could better assist the machine learning in network automation, for status prediction and anomaly detection.

Acknowledgments

This work was supported in part by the National Key R&D Program of China (2020YFB1806400), NSFC project 61871357, SPR Program of CAS (XDC02070300), and Fundamental Funds for Central Universities (WK3500000006).

References

[1] Cisco Visual Networking Index, 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] 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.

[3] V. Dukic *et al.*, "Beyond the mega-data center: Networking multi-data center regions," in *Proc. of ACM SIGCOMM 2020*, pp. 765–781, Aug. 2020.

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

[5] 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.

[6] 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.

[7] 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.

[8] M. Filer *et al.*, "Elastic optical networking in the Microsoft cloud," *J. Opt. Commun. Netw.*, vol. 8, pp. A45–A54, Jul. 2016.

[9] 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.

[10] S. Liu *et al.*, "DL-assisted cross-layer orchestration in software-defined IP-over-EONs: From algorithm design to system prototype," *J. Lightw. Technol.*, vol. 37, pp. 4426–4438, Sept. 2019.

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

[12] L. Gong and Z. Zhu, "Virtual optical network embedding (VONE) over elastic optical networks," *J. Lightw. Technol.*, vol. 32, pp. 450–460, Feb. 2014.

[13] H. Jiang, Y. Wang, L. Gong, and Z. Zhu, "Availability-aware survivable virtual network embedding (A-SVNE) in optical datacenter networks," *J. Opt. Commun. Netw.*, vol. 7, pp. 1160–1171, Dec. 2015.

[14] L. Gong, H. Jiang, Y. Wang, and Z. Zhu, "Novel location-constrained virtual network embedding (LC-VNE) algorithms towards integrated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 24, pp. 3648–3661, Dec. 2016.

[15] M. Zeng, W. Fang, and Z. Zhu, "Orchestrating tree-type VNF forwarding graphs in inter-DC elastic optical networks," *J. Lightw. Technol.*, vol. 34, pp. 3330–3341, Jul. 2016.

[16] J. Liu *et al.*, "On dynamic service function chain deployment and readjustment," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, pp. 543–553, Sept. 2017.

[17] X. Chen, F. Ji, and Z. Zhu, "Service availability oriented p-cycle protection design in elastic optical networks," *J. Opt. Commun. Netw.*, vol. 6, pp. 901–910, Oct. 2014.

[18] 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.

[19] Z. Zhu *et al.*, "Demonstration of cooperative resource allocation in an OpenFlow-controlled multidomain and multinational SD-EON testbed," *J. Lightw. Technol.*, vol. 33, pp. 1508–1514, Apr. 2015.

[20] I. Maor *et al.*, "First demonstration of SDN-controlled multi-layer restoration and its advantage over optical restoration," in *Proc. of ECOC 2016*, pp. 1–3, Sept. 2016.

[21] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A simple network management protocol (SNMP)," *RFC 1157*, May 1990. [Online]. Available: https://tools.ietf.org/html/rfc1157.

[22] P. Phaal, S. Panchen, and N. McKee, "InMon corporation's sFlow: A method for monitoring traffic in switched and routed networks," *RFC 3176*, Sept. 2001. [Online]. Available: https://tools.ietf.org/html/rfc3176.

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

[24] R. Cole, D. Romascanu, C. Kalbfleisch, and S. Waldbusser, "Introduction to the remote monitoring (RMON) family of MIB modules," *RFC 3577*, Oct. 2015. [Online]. Available: https://tools.ietf.org/html/rfc3577.

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

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

[27] S. Li *et al.*, "Protocol oblivious forwarding (POF): Software-defined networking with enhanced programmability," *IEEE Netw.*, vol. 31, pp. 12–20, Mar./Apr. 2017.

[28] INT dataplane specification. [Online]. Available: https://github.com/p4lang/p4-applications/blob/master/docs/INT_v2_1.pdf.

[29] B. Niu *et al.*, "Visualize your IP-over-optical network in realtime: A P4-based flexible multilayer in-band network telemetry (ML-INT) system," *IEEE Access*, vol. 7, pp. 82 413–82 423, Jun. 2019.

[30] S. Tang, J. Kong, B. Niu, and Z. Zhu, "Programmable multilayer INT: An enabler for AI-assisted network automation," *IEEE Commun. Mag.*, vol. 58, pp. 26–32, Jan. 2020.

[31] S. Tang *et al.*, "Sel-INT: A runtime-programmable selective in-band network telemetry system," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, pp. 708–721, Jun. 2020.

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

[33] 100G in-band network telemetry with Netcope P4. [Online]. Available: https://www.netcope.com/Netcope/media/content/100G-In-band-Network-Telemetry-With-Netcope-P4.pdf

[34] G. Simsek, D. Ergenc, and E. Onur, "Efficient network monitoring via in-band telemetry," in *Proc. of DRCN 2021*, pp. 1–6, Apr. 2021.

[35] A. Castro *et al.*, "Near-optimal probing planning for in-band network telemetry," *IEEE Commun. Lett.*, vol. 25, pp. 1630–1634, May 2021.

[36] J. Vestin *et al.*, "Programmable event detection for in-band network telemetry," in *Proc. of CloudNet 2019*, pp. 1–6, Nov. 2019.

[37] E. Song *et al.*, "INT-filter: Mitigating data collection overhead for high-resolution in-band network telemetry," in *Proc. of GLOBECOM 2020*, pp. 1–6, Dec. 2020.

[38] Y. Kim, D. Suh, and S. Pack, "Selective in-band network telemetry for overhead reduction," in *Proc. of CloudNet 2018*, pp. 1–3, Oct. 2018.

[39] B. Basat *et al.*, "PINT: Probabilistic in-band network telemetry," in *Proc. of ACM SIGCOMM 2020*, pp. 662–680, Aug. 2020.

[40] 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, May 2018.

[41] A. Sgambelluri *et al.*, "Exploiting telemetry in multi-layer networks," in *Proc. of ICTON 2020*, pp. 1–4, Jul. 2020.

[42] S. Tang *et al.*, "Self-adaptive network monitoring in IP-over-EONs: When multilayer telemetry is flexible and driven by data analytics," in *Proc. of OFC 2021*, pp. 1–3, Jun. 2021.

[43] S. Bhattacharyya, C. Diot, and J. Jetcheva, "Pop-level and access-link-level traffic dynamics in a Tier-1 POP," in *Proc. of ACM SIGCOMM IMW 2001*, pp. 39–53, Nov. 2001.

[44] ONOS. [Online]. Available: https://onosproject.org/.

[45] pktgen-DPDK. [Online]. Available: https://git.dpdk.org/apps/pktgen-dpdk/.

[46] S. Liu and Z. Zhu, "Generating data sets to emulate dynamic traffic in a backbone IP over optical network," *Tech. Rep.*, 2019. [Online]. Available: https://github.com/lsq93325/Traffic-creation/blob/master/README.md?tdsourcetag=s_pctim_aiomsg

[47] DBSCAN. [Online]. Available: https://en.wikipedia.org/wiki/DBSCAN.

[48] X. Chen *et al.*, "Self-taught anomaly detection with hybrid unsupervised/supervised machine learning in optical networks," *J. Lightw. Technol.*, vol. 37, pp. 1742–1749, Apr. 2019.

[49] S. Liu *et al.*, "Highly-efficient and automatic spectrum inspection based on AutoEncoder and semi-supervised learning for anomaly detection in EONs," *J. Lightw. Technol.*, vol. 39, pp. 1243–1254, Mar. 2021.