How to Use In-band Network Telemetry Wisely: Network-wise Orchestration of Sel-INT

Shaofei Tang, Sicheng Zhao, Xiaoqin Pan, and Zuqing Zhu, *Senior Member, IEEE*

***Abstract*—As a promising network monitoring technique, in-band network telemetry (INT) helps to visualize networks in a fine-grained and real-time manner. Meanwhile, to address the overheads of INT, people have proposed a few selective INT (Sel-INT) approaches that only select a portion of packets in each flow to insert INT fields and distribute different types of INT fields over the selected packets. In this paper, we study how to use Sel-INT wisely in a network such that the tradeoff between monitoring accuracy/coverage and INT overheads can be balanced well. Specifically, we try to orchestrate the Sel-INT schemes of flows in both network- and flow-levels. For the network-level optimization, we model it as an INT planning problem in which the Sel-INT schemes of flows should be determined to maximize the information gain of INT as well as minimize the bandwidth overheads of INT. We formulate an integer linear programming (ILP) model to tackle the problem, prove its $\mathcal{NP}$-hardness, and leverage Lagrangian relaxation to design a polynomial-time approximation algorithm for it. The flow-level optimization considers a dynamic network environment, and we propose to combine deep learning (DL) based traffic prediction with Sel-INT, such that the Sel-INT scheme of each individual flow can be updated timely and adaptively. We implement the proposal in a small but real network testbed and experimentally demonstrate self-adaptive orchestration of Sel-INT with it.**

***Index Terms*—In-band network telemetry (INT), Selective INT (Sel-INT), Software-defined networking (SDN), Programmable data plane (PDP), Lagrangian relaxation (LR).**

I. Introduction

OVER past decades, the Internet has evolved as an indispensable part of people's daily lives, with innumerable connected devices, complicated network infrastructures, fast-emerging new technologies, and, of course, many challenges [1]. Particularly, the long-term challenge since the inception of the Internet, *i.e.*, how to monitor networks accurately and efficiently, has become even harder to address. The difficulty comes from at least the following three aspects. Firstly, the advances on physical-layer technologies for wired and wireless networks [2–8] and heterogeneous network environments have made the correlations between network status and quality-of-service (QoS) metrics (*e.g.*, bandwidth, latency and jitter) more sophisticated. Secondly, the wide usage of virtualization techniques (*e.g.*, virtual network slicing [9–12] and network function virtualization (NFV) [13–16]) has decoupled network services from physical devices, and thus increased flexibility is achieved at the cost of complexity. Finally, the ever-growing network devices and applications can easily flood network monitors with unprocessable status data [17, 18]. Hence, traditional network monitoring approaches, which are usually based on out-of-band polling (such as SNMP [19] and sFlow [20]), can hardly catch up with the evolving requirements.

The introduction of software-defined networking (SDN) [21, 22] and programmable data plane (PDP) [23, 24] has fueled the development of network monitoring techniques. Specifically, the centralized network control and management (NC&M) achieved by SDN makes network monitoring much easier, while PDP enables network operators to customize their network monitoring schemes without being restricted by existing network protocols. Therefore, by leveraging the benefits of SDN and PDP, people proposed in-band network telemetry (INT) [25] to achieve fine-grained and real-time network monitoring. More specifically, by examining the INT instructions precoded in the header of each packet, PDP switches collect required network statistics (in switch/interface/buffer-level), encode them as INT fields, and insert the INT fields into the packet, for recording the exact network status experienced by the packet and how it gets processed hop-by-hop.

Note that, the initiative of the open networking foundation (ONF) on protocol independent forwarding (PIF) [26] suggests that PDP can be based on either the programming protocol-independent packet processors (P4) [23] or the protocol-oblivious forwarding (POF) [24]. P4 defines the way to write and compile packet processing programs with the P4 language, such that a PDP switch can be programmed in two phases, *i.e.*, configuration and runtime [23]. On the other hand, POF assembles packet processing pipelines in runtime by installing flow tables in PDP switches [24], which is similar to the approach used by OpenFlow except that the flow tables are protocol-oblivious [27]. Hence, INT can be realized with both P4 [28, 29] and POF [30] to visualize network operations.

However, despite its benefits, INT is associated with noticeable overheads from two perspectives. First of all, as inserting INT fields into packets is actually a heavy-weight operation for PDP switches, INT can affect packet processing performance, especially when it is realized on software-based PDP switches [30]. Secondly and more importantly, the insertion of INT fields increases packet lengths and causes extra bandwidth usage, which, if not controlled well, might severely degrade the overall performance of the network under monitoring. Previously, to address the overheads of INT, people have proposed various techniques, including inserting INT fields in packets selectively [30–32], distributing the INT fields for different types of telemetry data to different packets [33, 34], and compressing telemetry data to shorten the lengths

S. Tang, S. Zhao, X. Pan, 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).

X. Pan is also with the Engineering Technology Center, Southwest University of Science and Technology, Mianyang, Sichuan 621010, P. R. China.

Manuscript received on January 12, 2021.

of INT fields [32]. These techniques have been proven to be effective, but, to the best of our knowledge, the problem of how to use them wisely (*i.e.*, optimizing their operation schemes and parameters to balance the tradeoff between monitoring accuracy/coverage and INT overheads) has not been studied yet. Note that, we can hardly achieve efficient and effective network monitoring without considering this problem. For instance, if two flows go through a same switch, we might collect the same type of telemetry data on the switch with both flows, which induces redundant INT operations.

In this paper, we study how to wisely use the selective INT (Sel-INT) technique developed in our previous work [30] in a network. Specifically, we try to optimize the orchestration of Sel-INT in both network- and flow-levels. For the network-level optimization, we consider an INT planning problem, where the network carries a few elephant flows that can be used for Sel-INT, and the Sel-INT scheme of each flow should be determined to minimize the bandwidth overheads of INT as well as maximize the information gain of INT. Here, we assume that for a switch, each type of telemetry data has its own information, which can be considered as the gain of monitoring it. The INT planning problem essentially consists of two subproblems: 1) how to select the types of telemetry data to collect at each PDP switch such that the information gain of INT can be maximized (*INT data selection*), and 2) how to assign the selected telemetry data to flows so that the overheads of INT can be minimized (*INT data assignment*).

We first formulate an integer linear programming (ILP) model to optimize the INT planning problem, and prove its $\mathcal{NP}$-hardness. Then, we develop a technique to preprocess the constraints of the problem to get a flow-covered graph, for reducing the solution space. Based on the flow-covered graph, we propose a polynomial-time approximation algorithm by leveraging the Lagrangian relaxation. Numerical simulations confirm that our approximation algorithm can obtain near-optimal solutions quickly with only a few iterations.

For the flow-level optimization, we consider a dynamic network environment, and expand our POF-based Sel-INT system designed in [30] to achieve self-adaptive orchestration of Sel-INT. Specifically, we combine deep learning (DL) based traffic prediction with Sel-INT, and show that self-adaptive network monitoring can be realized by adjusting the parameters of Sel-INT according to traffic prediction. The proposed system is implemented and experimentally demonstrated in a small but real network testbed that consists of six stand-alone POF-based switches. The major contributions of this work are as follows:

- To the best of our knowledge, this is the first study that leverages network- and flow-level orchestration of Sel-INT to optimize the tradeoff between monitoring accuracy/coverage and INT overheads for a whole network.
- For the network-level optimization, we formulate a planning problem to optimize the overheads and information gain of INT jointly, prove its $\mathcal{NP}$-hardness, and design a polynomial-time approximation algorithm to solve it.
- For the flow-level optimization, we propose to combine DL-based traffic prediction with Sel-INT, and design and experimentally demonstrate a POF-based system that can achieve self-adaptive orchestration of Sel-INT.

The rest of paper is organized as follows. Section II briefly surveys the related work. We first describe the problem of Sel-INT orchestration and related challenges in Section III. Then, we formulate the ILP model for network-level optimization and design the approximation algorithm in Sections IV and V, respectively. Section VI discusses the simulations about network-level Sel-INT orchestration. The system design and demonstrations of flow-level Sel-INT orchestration are shown in Section VII. Finally, Section VIII summarizes the paper.

II. Related Work

The first technical specification about INT was released in 2016 [25], which defines the architecture of an INT-based network monitoring system, the operation principle of INT, and the related header fields and packet formats. Here, the INT works in a per-packet manner, which means that when a flow is selected for INT, each switch on its routing path will insert all the required INT fields in each of its packets. Since then, there have been a few hardware- or software-based implementations of per-packet INT, and people have explored the application scenarios for network monitoring and troubleshooting.

As for hardware implementations, DeepInsight [35] was shown to visualize networks with a temporal resolution in the order of nanoseconds, and Netcope [29] was demonstrated to realize per-packet INT at a line-rate of 100 Gbps. Most of the software implementations of per-packet INT [28] were based on P4 and leveraged BMv2 [36], which is a P4-based software switch. Their packet processing capability is far below that of their hardware counterparts, but the flexibility of software enables them to realize proof-of-concept demonstrations of new features conveniently. The application scenarios of per-packet INT have also been discussed in existing studies, where it could help a NFV platform to achieve latency-aware feedback [37], realize high precision congestion control [38], and assist the operation of a wireless sensor network [39].

However, sampling network status on the per-packet basis might not be necessary in most of the network monitoring cases, especially when the line-rate is relatively high [30, 32]. Therefore, to avoid the excessive over-sampling of per-packet INT, people designed several selective INT techniques that insert INT fields in packets selectively and distribute the INT fields for different types of telemetry data to different packets [30–32]. Although these techniques can effectively reduce the overheads of INT and our Sel-INT in [30] even realizes runtime programmability, how to use them in the way such that their operation schemes and parameters are optimized for each flow to balance the tradeoff between monitoring accuracy/coverage and INT overheads has not been discussed.

In order to use INT wisely, we need to orchestrate its operation schemes for different flows in a network-wise manner. Following this direction, previous studies have proposed approaches generally in two categories. In the first category, people introduced probe flows and tried to optimize the operation schemes of INT for probe flows [40]. Nevertheless, using probe flows for INT-based network monitoring has two drawbacks. Firstly, probe flows may not experience exactly the same network state as application flows, which affects the

accuracy of monitoring and makes the monitoring not real-time. Secondly, probe flows cause extra bandwidth usages and increase the packet processing load of switches, and thus they can impact the forwarding of application flows. In terms of problem formulation, the routing paths of probe flows are variables when optimizing their INT operation schemes, but if we apply INT on application flows, their routing paths usually should not be changed just for INT. This makes the optimization of probe flow-based INT orchestration different from the one considered in this work. In the second category, researchers studied how to optimize the operation schemes of INT for application flows [41].

However, no matter which category the existing studies on network-wise orchestration of INT belong to, none of them has explored the flexibility of selective INT. This motivates us to investigate how to optimize the orchestration of Sel-INT in both network- and flow-levels in this paper.

III. Sel-INT Orchestration

In this section, we first describe the challenges of using INT wisely in a network, and then explain the network model and problem definition of Sel-INT orchestration.

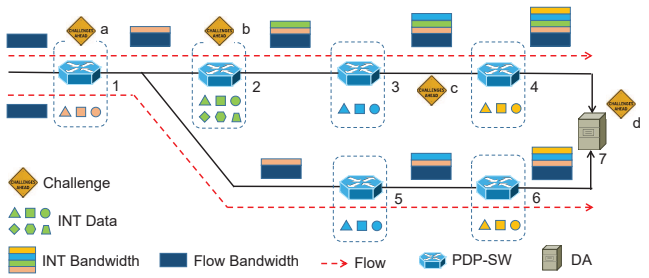


Fig. 1. Operation principle of Sel-INT.

A. Operation Principle of Sel-INT and its Challenges

Fig. 1 explains the operation principle of Sel-INT and the challenges that it faces. Here, the network carries a few application flows, which will last for a while (*e.g.*, elephant flows) and need the INT service. Different from the traditional per-packet INT, our Sel-INT [30] only selects a portion of packets in each flow to insert INT fields, and distributes different types of INT fields over the selected packets. Hence, Sel-INT can effectively reduce the overheads of INT on packet processing and bandwidth usage. However, this is not good enough, because the Sel-INT schemes of the flows should be optimized jointly to address the following challenges.

- *Problematic data collection*: As there are multiple flows in the network and they can share PDP switches (PDP-SWs) and links, the telemetry data collected by the Sel-INT on them will have spatial and temporal dependencies. Therefore, if their Sel-INT schemes are not orchestrated well, we may either over-sample or under-sample certain telemetry data on the PDP-SWs. For instance, the two flows in Fig. 1 share *PDP-SW* 1, and thus if their Sel-INT schemes are not optimized jointly, we not only lose the optimal monitoring coverage of *PDP-SW* 1, but also make the data parsing and assembling in the data analyzer (DA) more complex (*i.e.*, *Challenges* a and d in Fig. 1).
- *Potential performance bottleneck*: Sel-INT introduces overheads on both the packet processing in PDP-SWs and the bandwidth usage on links, and thus the Sel-INT schemes of flows should be optimized under the corresponding resource constraints. For example, the Sel-INT scheme of the top flow in Fig. 1 should not overload any of the PDP-SWs on its routing path or cause congestions on the links *i.e.*, *Challenges* b and c in Fig. 1).

Fig. 2 provides an illustrative example to explain the necessity of Sel-INT orchestration. We can see that using *Flow* 3 to collect the telemetry data on *PDP-SW* 2 marked with the green square and getting it processed in *DA* 6 are not proper, because the corresponding INT field does not get transferred over the shortest feasible path and thus causes unnecessary bandwidth overheads. Apparently, using *Flow* 1 to collect the telemetry data on *PDP-SW* 2 and getting it processed in *DA* 7 will be a better solution. Meanwhile, as *Flows* 2 and 3 both collect the telemetry data on *PDP-SW* 3 marked with the pink triangle and circle, the telemetry data is over-sampled and thus brings redundant packet processing load and bandwidth usage.

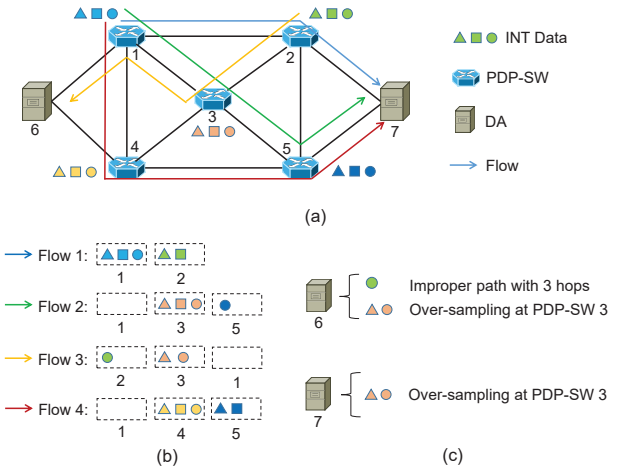


Fig. 2. Example on Sel-INT orchestration, a) network topology and flow routing, b) Sel-INT schemes of flows, and c) issues with the Sel-INT schemes.

To address the challenges mentioned above, we need to formulate and solve the problem of Sel-INT orchestration. Note that, this problem should be tackled in both the network- and flow-levels. Specifically, the network-level optimization considers a static INT planning problem to determine the Sel-INT schemes of all the flows jointly such that the tradeoff between monitoring accuracy/coverage and INT overheads can be balanced well, while the flow-level optimization is more like a dynamic INT provisioning problem, whose solution fine-tunes the Sel-INT scheme of each flow to make sure that it can adapt to the dynamic network environment.

B. Network Model and Problem Definition

We model the network as an undirected graph $G(V, E)$, where V and E are set of network nodes and links, respectively. Here, we have two types of network nodes, which are

the PDP-SWs that can apply Sel-INT to flows and are included in set V_s, and the DAs that are used to collect INT fields from packets and are in set V_d. Each PDP-SW $v_s \in V_s$ has a packet sampling capacity for Sel-INT as Z_{v_s}, and each link $e \in E$ has a bandwidth capacity for Sel-INT as B_e. Note that, in the network-level optimization, we assume that the values of $\{Z_{v_s}\}$ and $\{B_e\}$ are pre-known, because for the static network planning, they can be understood as estimated peak-/average-values. On the other hand, the values of $\{Z_{v_s}\}$ and $\{B_e\}$ in the flow-level optimization are time-varying, and we will study how to leverage DL-based traffic prediction to obtain them and adjust the Sel-INT scheme of each flow dynamically to achieve self-adaptive network monitoring.

Meanwhile, the set of telemetry data types that a PDP-SW v_s can collect is defined as M_{v_s}. For a telemetry data type $m \in M_{v_s}$, the length of its INT field is s_m bytes (*i.e.*, the bandwidth overhead), and its collection brings the information gain of $\eta_{v_s,m}$ on network monitoring. The set of flows considered for Sel-INT is defined as F, and the k-th flow uses the routing path $p_k \in P$ and has an average rate of f_k^r packets/sec. If the k-th flow passes through PDP-SW v_s, the adjacent collections of telemetry data $m \in M_{v_s}$ with it should be spaced by $t_{v_s}^m$ in the time domain, to avoid over- and under-sampling. Hence, the Sel-INT operation for the k-th flow on PDP-SW v_s consumes a packet sampling capacity of $\lceil \frac{1}{t_{v_s}^m} \rceil$.

With the network model above, the network-level optimization tries to maximize the total information gain of INT for network monitoring and minimize the overall bandwidth overheads of INT. Specifically, it needs to solve two subproblems: 1) how to select the telemetry data to collect with Sel-INT (*INT data selection*), and 2) how to assign the selected telemetry data to flows and collect it at PDP-SWs (*INT data assignment*).

IV. ILP Model for Network-level Optimization

In this section, we formulate an ILP model to solve the network-level optimization of Sel-INT orchestration.

Notations:

- $G(V, E)$: the network's topology, where V and E are the sets of network nodes and links, respectively, and $V = V_s \cup V_d$ includes both PDP-SWs (V_s) and DAs (V_d).
- B_e: the available bandwidth capacity on link $e \in E$, which can be used for Sel-INT.
- Z_{v_s}: the packet sampling capacity on PDP-SW $v_s \in V_s$, which can be used for Sel-INT.
- M_{v_s}: the set of telemetry data types that a PDP-SW $v_s \in V_s$ can collect.
- F: the set of the flows that are considered for Sel-INT.
- P: the set of flow paths, where $p_k \in P$ is the routing path of the k-th flow in F.
- $L_{(v_s,v)}^k$: the set of all the links before link $(v_s, v) \in E$ on the path of the k-th flow.
- $h_{v_s}^k$: the number of remaining hops on the path of the k-th flow after PDP-SW $v_s \in V_s$.
- f_k^r: the average packet rate of the k-th flow.
- ω_e^k: the boolean indicator that equals 1 if the k-th flow passes through link $e \in E$, and 0 otherwise.
- $\sigma_{v_s}^k$: the boolean indicator that equals 1 if the k-th flow passes through PDP-SW $v_s \in V_s$, and 0 otherwise.
- s_m: the length of the INT field for $m \in M_{v_s}$ in bytes.
- $t_{v_s}^m$: the monitoring period for collecting telemetry data $m \in M_{v_s}$ at PDP-SW v_s.
- $\eta_{v_s,m}$: the information gain brought by collecting telemetry data m at PDP-SW v_s, for network monitoring.

Variables:

- $\pi_{v_s,m}^k$: the boolean variable that equals 1 if telemetry data m is collected by the k-th flow at v_s, and 0 otherwise.

Objective:

The total information gain brought by the Sel-INT schemes on flows can be calculated and normalized as

$$\Phi_g = \frac{1}{\sum\limits_{v_s \in V_s} |M_{v_s}|} \cdot \left(\sum_{v_s \in V_s} \sum_{m \in M_{v_s}} \sum_{k=1}^{|F|} \sigma_{v_s}^k \cdot \pi_{v_s,m}^k \cdot \eta_{v_s,m} \right). \tag{1}$$

According to the operation principle of INT, the bandwidth overhead brought in at one PDP-SW will be inherited by the following hops of a flow's path, and thus the overall bandwidth overheads of INT can be calculated and normalized as

$$\Phi_c = \frac{1}{\sum\limits_{e \in E} B_e} \cdot \left(\sum_{v_s \in V_s} \sum_{k=1}^{|F|} \sum_{e=(u_s,v') \in L_{(v_s,v)}^k} \sum_{m \in M_{u_s}} \omega_e^k \cdot \pi_{u_s,m}^k \cdot s_m \cdot \lceil \frac{1}{t_{u_s}^m} \rceil \right). \tag{2}$$

Note that, for a type of telemetry data $m \in M_{v_s}$ on a specific PDP-SW v_s, its bandwidth usage per hop is

$$B_{v_s}^m = \pi_{v_s,m}^k \cdot s_m \cdot \lceil \frac{1}{t_{v_s}^m} \rceil, \tag{3}$$

and its total bandwidth usage since PDP-SW v_s is

$$\tilde{B}_{v_s}^m = \pi_{v_s,m}^k \cdot s_m \cdot \lceil \frac{1}{t_{v_s}^m} \rceil \cdot h_{v_s}^k. \tag{4}$$

Therefore, Eq. (2) can be further simplified as

$$\Phi_c = \frac{1}{\sum\limits_{e \in E} B_e} \cdot \left(\sum_{v_s \in V_s} \sum_{k=1}^{|F|} \sum_{m \in M_{v_s}} \sigma_{v_s}^k \cdot \pi_{v_s,m}^k \cdot s_m \cdot \lceil \frac{1}{t_{v_s}^m} \rceil \cdot h_{v_s}^k \right). \tag{5}$$

Then, the optimization objective of Sel-INT orchestration is

$$\text{Maximize} \quad \Phi = \alpha \cdot \Phi_g - \beta \cdot \Phi_c, \tag{6}$$

where α and β are the non-negative weights to balance the importance of Φ_g and Φ_c. We set $\alpha \gg \beta$ to ensure that maximizing Φ_g is the primary objective.

Constraints:

$$\sum_k \sum_{e=(u_s,v') \in L_{(v_s,v)}^k} \sum_{m \in M_{u_s}} \omega_e^k \cdot \pi_{u_s,m}^k \cdot s_m \cdot \lceil \frac{1}{t_{u_s}^m} \rceil \leq B_{(v_s,v)}, \quad \forall v_s \in V_s,\ (v_s, v) \in E. \tag{7}$$

Eq. (7) ensures that the bandwidth used for Sel-INT will not exceed available bandwidth capacity for it on each link.

$$\sum_k \pi_{v_s,m}^k \leq 1, \quad \forall v_s \in V_s,\ m \in M_{v_s}. \tag{8}$$

Eq. (8) ensures that each type of telemetry data m at PDP-SW v_s is collected by one flow at most, to avoid over-sampling.

$$\pi_{v_s,m}^k \cdot \lceil \frac{1}{t_{v_s}^m} \rceil \leq \sigma_{v_s}^k \cdot f_k^r,\ \forall k \in [1, |F|],\ v_s \in V_s,\ m \in M_{v_s}. \tag{9}$$

Eq. (9) ensures that a type of telemetry data $m \in M_{v_s}$ can only be assigned to the k-th flow, if the flow passes through PDP-SW v_s and its packet rate can satisfy the required monitoring period for collecting m (*i.e.*, $t^m_{v_s}$).

$$\sum_{k} \sum_{m \in M_{v_s}} \pi^k_{v_s,m} \cdot \lceil \frac{1}{t^m_{v_s}} \rceil \leq Z_{v_s}, \quad \forall v_s \in V_s. \tag{10}$$

Eq. (10) ensures that for each PDP-SW, the Sel-INT schemes of flows at it will not exceed its available packet sampling capacity for Sel-INT, *i.e.*, not overloading the PDP-SW.

Theorem 1: The network-level optimization of Sel-INT orchestration is an $\mathcal{NP}$-hard problem.

Proof: We prove the $\mathcal{NP}$-hardness of the network-level optimization of Sel-INT orchestration by reducing it into a general case of a well-known $\mathcal{NP}$-hard problem. We first have $B_{(v_s,v)} = f^r_k = +\infty$ to relax Eqs. (7) and (9), and then let $|V_s| = 1$ and $|F| = 1$. Hence, the optimization is described by Eqs. (6), (8) and (10), which satisfy the description of a general 0-1 knapsack problem [42]. Specifically, we can treat each type of telemetry data $m \in M_{v_s}$ at PDP-SW v_s as an item whose weight and value are $\lceil \frac{1}{t^m_{v_s}} \rceil$ and $\eta_{v_s,m}$, respectively, while the PDP-SW v_s becomes the knapsack with a capacity of Z_{v_s}. Then, the original problem becomes to find a way to select items to put into the knapsack such that their total value is maximized. This is the general case of the 0-1 knapsack problem, which is known to be $\mathcal{NP}$-hard [42]. Hence, we prove the $\mathcal{NP}$-hardness of the original problem. ■

V. Approximation Algorithm for Network-level Optimization

As the network-level optimization of Sel-INT orchestration is $\mathcal{NP}$-hard, we, in this section, leverage the Lagrangian relaxation (LR) to design a polynomial-time approximation algorithm to solve it. Specifically, the approximation algorithm works as follows. It first incorporates a two-step preprocessing to 1) process the constraints of the network-level optimization to get a flow-covered graph, for reducing the solution space, and 2) transform the hard constraint. Then, it constructs the LR problem, which relaxes the remaining hard constraint into the objective. The solution of the LR problem provides an upper-bound on the optimal solution of the original problem. Next, the algorithm builds the Lagrangian dual problem based on the LR one, until a best feasible solution of the original problem can be obtained under the current condition. As the original problem is for maximization, the feasible solution gives a lower-bound on its optimal solution. Finally, the algorithm optimizes the upper-/lower-bounds iteratively to reduce the gap between them, until a near-optimal solution can be obtained.

A. Preprocessing

Before leveraging the procedure of LR, we preprocess the constraints in Eqs. (7) and (9) with the following two steps.

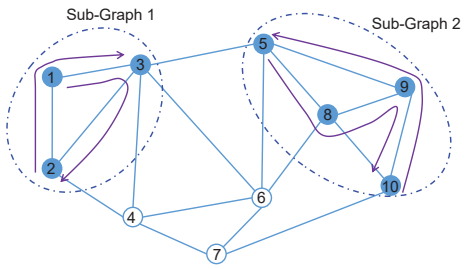
1) Extracting the Flow-covered Graph: Since our Sel-INT orchestration does not utilize any probe flows and all the Sel-INT schemes are enabled by the flows in F, we can preprocess the network topology $G(V, E)$ based on the flows' routing paths in P to simplify our algorithm design. Specifically, a feasible solution of the ILP in Section IV will not include the nodes and links that do not carry any flow in F, according to the principle of Sel-INT. Hence, we remove such nodes and links from $G(V, E)$ to extract a flow-covered graph $G'(V, E)$, to avoid checking the solutions that are apparently infeasible and improve the time-efficiency of our approximation algorithm. For instance, in Fig. 3(a), the flows only pass through the nodes marked as blue and the links circled by the dash-dotted lines. Therefore, we can obtain a flow-covered graph $G'(V, E)$ that includes two sub-graphs and a virtual link whose bandwidth capacity is 0, as shown in Fig. 3(b). As INT fields will only be transmitted on the links in the sub-graphs, the virtual link is irrelevant to our problem solving and thus we set its bandwidth capacity as 0.

Algorithm 1 explains the detailed procedure to extract the flow-covered graph. *Lines* 1-11 generate the flow-covered graph based on $G(V, E)$ and the flows' paths in P. Then, for those types of telemetry data at a PDP-SW, which requires too short sampling period that cannot be satisfied by any of the flows passing through the PDP-SW, we also remove them to further reduce the solution space (*Lines* 12-18). The time complexity of *Algorithm* 1 is $O(|V_s| \cdot \max(|M_{v_s}|))$. Note that, after the preprocessing with *Algorithm* 1, the sets V_s, E and $\{M_{v_s}\}$ are updated. Hence, for simplicity, in the following discussions throughout Section V, V_s, E and $\{M_{v_s}\}$ denote the sets preprocessed by *Algorithm* 1 if without specific statement.

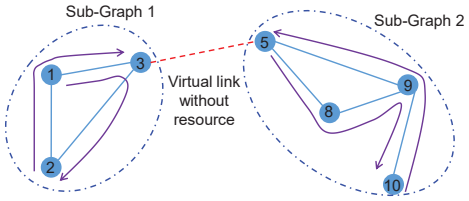
Algorithm 1: Extracting the Flow-covered Graph

Input: $G(V, E)$, P, $\{M_{v_s}\}$, F, $\{\lceil \frac{1}{t^m_{v_s}} \rceil\}$.
Output: $\{M_{v_s}\}$, $G'(V, E)$.
1 **for** *each* $v_s \in V_s$ **do**
2 **if** v_s *is not used in* P **then**
3 $V_s = V_s \setminus v_s$;
4 **end**
5 **end**
6 **for** *each* $e \in E$ **do**
7 **if** e *is not used in* P **then**
8 $E = E \setminus e$;
9 **end**
10 **end**
11 connect the remaining sub-graphs in $G'(V, E)$ with zero-resource virtual links;
12 **for** *each* $v_s \in V_s$ **do**
13 **for** *each* $m \in M_{v_s}$ **do**
14 **if** *none of the flows on* v_s *can satisfy* $t^m_{v_s}$ **then**
15 $M_{v_s} = M_{v_s} \setminus m$;
16 **end**
17 **end**
18 **end**
19 update the constraints with $\{M_{v_s}\}$ and $G'(V, E)$;

2) Transforming the Hard Constraint: By checking the network-level optimization defined by Eqs. (6)-(10), we can see that Eq. (7) is a hard constraint. This is because Eq. (7) is a link-based formula that induces complex dependencies among variables $\{\pi^k_{v_s,m}\}$, *i.e.*, the Sel-INT scheme on a PDP-SW depends on those on the PDP-SWs before it and it in turn affects those on the PDP-SWs after it. Therefore, to simplify the optimization, we sum up both sides of all the constraints



(a) Network topology



(b) Flow-covered graph

Fig. 3. Example on flow-covered graph extraction.

defined by Eq. (7) to get a sightly-relaxed one as

$$\sum_{(v_s,v)\in E}\sum_{k}\sum_{e=(u_s,v')\in L^k_{(v_s,v)}}\sum_{m\in M_{u_s}}\omega^k_e\cdot\pi^k_{u_s,m}\cdot s_m\cdot\lceil\frac{1}{t^m_{u_s}}\rceil \leq \sum_{(v_s,v)\in E}B_{(v_s,v)}. \tag{11}$$

Note that, *Algorithm* 1 has already removed the elements that are in V_s, E and $\{M_{v_s}\}$ and are irrelevant to our problem-solving. This ensures that Eq. (11) is tighter than the one obtained by simply aggregating the constraints defined by Eq. (7) without *Algorithm* 1. Meanwhile, by transforming the link-based formula in Eq. (2) to the node-based one in Eq. (5), we have already verified that for the network-level optimization, link- and node-based constraints/objective are equivalent while the node-based ones are simpler. Hence, with the same procedure to transform Eq. (2) into Eq. (5), we simplify Eq. (11) to its node-based formula as

$$\sum_{v_s\in V_s}\sum_{k}\sum_{m\in M_{v_s}}\sigma^k_{v_s}\cdot\pi^k_{v_s,m}\cdot s_m\cdot\lceil\frac{1}{t^m_{v_s}}\rceil\cdot h^k_{v_s} \leq \sum_{(v_s,v)\in E}B_{(v_s,v)}. \tag{12}$$

Algorithm 2: Assigning Telemetry Data to Flows

Input: $\{\bar{\eta}^k_{v_s,m}\}$, $\tilde{F}=\emptyset$.
Output: $\tilde{F}=\{f_{v_s,m}\}$.
for *each* $v_s\in V_s$ **do**
  for *each* $m\in M_{v_s}$ **do**
    find the flow k to maximize $\bar{\eta}^k_{v_s,m}$ in Eq. (16);
    $f_{v_s,m}=k$, insert $f_{v_s,m}$ in $\tilde{F}$;
  end
end

B. Lagrangian Relaxation Problem

After the two-step preprocessing mentioned above, we relax the constraint in Eq. (9) and will consider it in Section V-C when constructing a feasible solution in *Algorithm* 4. Then, the network-level optimization are defined with the constraints in Eqs. (8), (10) and (12) and the objective Eq. (6). However, Eq. (12) is still a hard constraint, and thus we leverage LR to relax it. Specifically, Eq. (12) can be expressed with $A\cdot x\leq b$ in the scalar format, which is its Lagrangian relaxed term, as

$$b-A\cdot x=\frac{1}{\sum_{(v_s,v)\in E}B_{(v_s,v)}}\cdot\left(\sum_{(v_s,v)\in E}B_{(v_s,v)} -\sum_{v_s\in V_s}\sum_{k}\sum_{m\in M_{v_s}}\sigma^k_{v_s}\cdot\pi^k_{v_s,m}\cdot s_m\cdot\lceil\frac{1}{t^m_{v_s}}\rceil\cdot h^k_{v_s}\right). \tag{13}$$

Then, the hard constraint in Eq. (12) can be removed to be put in the following relaxed objective for getting the LR problem.

$$\begin{aligned}Z_{\text{LR}}(\Lambda)=\underset{\{\pi^k_{v_s,m}\}}{\text{Maximize}}\quad&\Phi+\lambda\cdot(b-A\cdot x)=\\ &\sum_{v_s\in V_s}\sum_{m\in M_{v_s}}\sum_{k}\cdot\left(\frac{\alpha}{\sum_{v_s\in V_s}|M_{v_s}|}\cdot\sigma^k_{v_s}\cdot\pi^k_{v_s,m}\cdot\eta_{v_s,m}\right.\\ &\left.-\frac{\beta}{\sum_{(v_s,v)\in E}B_{(v_s,v)}}\cdot\sigma^k_{v_s}\cdot\pi^k_{v_s,m}\cdot s_m\cdot\lceil\frac{1}{t^m_{v_s}}\rceil\cdot h^k_{v_s}\right)\\ &+\frac{\lambda}{\sum_{(v_s,v)\in E}B_{(v_s,v)}}\cdot\left(\sum_{(v_s,v)\in E}B_{(v_s,v)}\right.\\ &\left.-\sum_{v_s\in V_s}\sum_{k}\sum_{m\in M_{v_s}}\sigma^k_{v_s}\cdot\pi^k_{v_s,m}\cdot s_m\cdot\lceil\frac{1}{t^m_{v_s}}\rceil\cdot h^k_{v_s}\right),\end{aligned} \tag{14}$$

where $\Lambda=\{\lambda\}$ is the vector of Lagrangian multipliers. As we have $\lambda\geq 0$, $Z_{\text{LR}}(\Lambda)$ provides an upper-bound on the original objective Φ for a specific λ. Then, Eq. (14) can be reduced to

$$Z_{\text{LR}}(\Lambda)=\underset{\{\pi^k_{v_s,m}\}}{\text{Maximize}}\quad\sum_{v_s\in V_s}\sum_{k}\sum_{m\in M_{v_s}}\bar{\eta}^k_{v_s,m}\cdot\pi^k_{v_s,m}+\lambda, \tag{15}$$

where $\bar{\eta}^k_{v_s,m}$ is the Lagrangian-modified information gain as

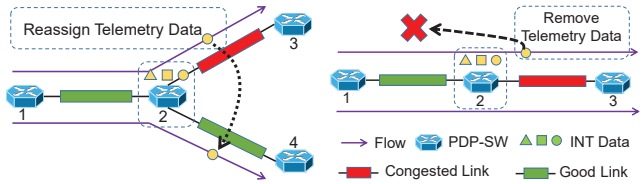
$$\bar{\eta}^k_{v_s,m}=\left(\frac{\alpha}{\sum_{v_s\in V_s}|M_{v_s}|}\cdot\eta_{v_s,m}- \frac{\beta+\lambda}{\sum_{(v_s,v)\in E}B_{(v_s,v)}}\cdot s_m\cdot\lceil\frac{1}{t^m_{v_s}}\rceil\cdot h^k_{v_s}\right)\cdot\sigma^k_{v_s}. \tag{16}$$

Since the second term on the right side of Eq. (15) is independent of $\{\pi^k_{v_s,m}\}$, we transform the LR problem as

$$Z_{\text{LR}}(\Lambda)=\underset{\{\pi^k_{v_s,m}\}}{\text{Maximize}}\quad\sum_{v_s\in V_s}\sum_{k}\sum_{m\in M_{v_s}}\bar{\eta}^k_{v_s,m}\cdot\pi^k_{v_s,m}, \quad\text{s.t. Eqs. (8) and (10).} \tag{17}$$

We use the following two steps to solve the LR problem in Eq. (17). Firstly, for each type of telemetry data $m\in M_{v_s}$ at a PDP-SW v_s, we try to assign it to a flow k that passes through v_s, such that the corresponding Lagrangian-modified information gain (*i.e.*, $\bar{\eta}^k_{v_s,m}$) can be maximized. Secondly, we finalize the types of telemetry data to collect at each PDP-SW, to achieve the objective in Eq. (17). The first step can

be easily accomplished with *Algorithm* 2. Note that, in *Line* 3, there could be multiple flows that all can maximize the Lagrangian-modified information gain $\bar{\eta}^k_{v_s,m}$. In this case, we just randomly select one of them to put in $f_{v_s,m}$ because the flows are equivalent. The time complexity of *Algorithm* 2 is $O(|V_s| \cdot \max(|M_{v_s}|))$. The problem in the second step can be solved with the dynamic programming in *Algorithm* 3, whose time complexity is $O(|V_s| \cdot \max(|M_{v_s}|) \cdot \max(|Z_{v_s}|))$.



(a) Reassignment of telemetry data (b) Removal of telemetry data

Fig. 4. Construct feasible solution based on the output of *Algorithm* 3.

C. Construction of Feasible Solution

In each iteration, *Algorithm* 3 gets an optimal solution of $Z_{\text{LR}}(\Lambda)$ for a specific λ. However, the solution can still violate the bandwidth capacity constraint in Eq. (12), due to the LR. Hence, we design *Algorithm* 4 to construct a feasible solution of the original problem, with the approaches shown in Fig. 4. Specifically, if the Sel-INT schemes from *Algorithm* 3 violate the bandwidth capacity constraint, we either reassign some of the telemetry data to other feasible flows (as in Fig. 4(a)) or remove certain telemetry data from the flows (as in Fig. 4(b)).

We can easily verify the correctness of *Algorithm* 4. In the worst case, it can simply remove all the telemetry data from the flows to ensure that the bandwidth capacity constraint is satisfied. Meanwhile, *Algorithm* 4 is necessary for our LR-based algorithm, because *Algorithm* 3 only solves a relaxed version of the original problem. Note that, there might be other ways of building a feasible solution based on the infeasible one from *Algorithm* 3. As long as an approach can accomplish this, we can replace current *Algorithm* 4 with it, which will not affect the performance of our LR-based algorithm.

In *Algorithm* 4, *Line* 1 is for the initialization, where Z^* records the objective of the feasible solution, and ξ_e is the indicator to show whether the bandwidth capacity constraint on link $e \in E$ is satisfied or not. Hence, *Algorithm* 4 should only process the links with $\xi_e = 0$ to convert the infeasible solution from *Algorithm* 3 to a feasible one. Then, the for-loop that covers *Lines* 2-34 checks each link in E to ensure the bandwidth capacity constraint. Here, *Lines* 2-7 determine whether the bandwidth capacity constraint of a specific link (v_s, v) is satisfied or not. If not, the for-loop covering *Lines* 8-34 checks each flow that uses the link, and tries to leverage the two approaches in Fig. 4 to correct the violation. After all the links having been checked, *Lines* 35-46 update $\{\pi^k_{v_s,m}\}$ to ensure the constraint in Eq. (9). Then, Z^* denotes the objective of the obtained feasible solution. As the original problem is for maximization, the feasible solution constructed by *Algorithm* 4 makes sure that Z^* provides a lower-bound. The time complexity of *Algorithm* 4 is $O(|E| \cdot |F|^2 \cdot \max(|M_{v_s}|))$.

Algorithm 3: Dynamic Programming to Get $Z_{\text{LR}}(\Lambda)$

Input: $\{\lceil \frac{1}{t^m_{v_s}} \rceil\}$, $\{\bar{\eta}^k_{v_s,m}\}$, $\{Z_{v_s}\}$, $\tilde{F}$.
Output: $\{\pi^k_{v_s,m}\}$, $Z_{\text{LR}}(\Lambda)$.

1 $Z_{\text{LR}}(\Lambda) = 0$;
2 **for** *each* $v_s \in V_s$ **do**
3 $\quad \mathbf{D}_{v_s} = \{d^{m,n}_{v_s} = 0, m \in [1, |M_{v_s}|], n \in [1, |Z_{v_s}|]\}$;
4 $\quad$ **for** *each* $m \in [1, |M_{v_s}|]$ **do**
5 $\quad\quad$ **for** *each* $n \in [1, |Z_{v_s}|]$ **do**
6 $\quad\quad\quad$ **if** $\lceil \frac{1}{t^m_{v_s}} \rceil > n$ **then**
7 $\quad\quad\quad\quad d^{m,n}_{v_s} = d^{m-1,n}_{v_s}$;
8 $\quad\quad\quad$ **else**
9 $\quad\quad\quad\quad d^{m,n}_{v_s} = \max\left(d^{m-1,n}_{v_s}, d^{m-1,n-\lceil \frac{1}{t^m_{v_s}} \rceil}_{v_s} + \bar{\eta}^k_{v_s,m}\right)$;
10 $\quad\quad\quad$ **end**
11 $\quad\quad$ **end**
12 $\quad$ **end**
13 $\quad Z_{\text{LR}}(\Lambda) = Z_{\text{LR}}(\Lambda) + d^{|M_{v_s}|,|Z_{v_s}|}_{v_s}$;
14 $\quad \{\pi^k_{v_s,m} = 0, \ \forall m \in M_{v_s}, \ k \in F\}$;
15 $\quad m = |M_{v_s}| + 1, \ n = |Z_{v_s}|$;
16 $\quad$ **while** $m > 1$ *AND* $n > 0$ **do**
17 $\quad\quad$ **if** $d^{m,n}_{v_s} > d^{m-1,n}_{v_s}$ **then**
18 $\quad\quad\quad k = f_{v_s,m-1}$, $\pi^k_{v_s,m-1} = 1$;
19 $\quad\quad\quad n = n - \lceil \frac{1}{t^{m-1}_{v_s}} \rceil$;
20 $\quad\quad$ **end**
21 $\quad\quad m = m - 1$;
22 $\quad$ **end**
23 **end**

D. Solving Lagrangian Dual Problem

After getting the feasible solution with *Algorithm* 4 for a specific λ, we still need to optimize the choice of λ to minimize the penalty due to the relaxed constraint (*i.e.*, $\lambda \cdot (b - A \cdot x)$ in Eq. (14)). This can be done by further processing $Z_{\text{LR}}(\Lambda)$ with the sub-gradient method [43]. Specifically, we initialize λ as 0, and update its value in the i-th iteration as

$$\lambda_{i+1} = \lambda_i - \mu_i \cdot f(\lambda_i), \tag{18}$$

where $f(\lambda_i)$ is the sub-gradient function that is defined as

$$f(\lambda) = \frac{\partial Z_{\text{LR}}(\Lambda)}{\partial \lambda} = b - A \cdot x, \tag{19}$$

and μ_i is the step length. As μ_i also affects the convergence of the sub-gradient method, we calculate it as follows [44]

$$\mu_i = \frac{\nu_i \cdot (Z_{\text{LR}}(\lambda_i) - Z^*)}{||f(\lambda_i)||^2}, \tag{20}$$

where $\nu_i \in (0, 2]$ is a scaler variable and Z^* is the best-known feasible solution of the original problem. Here, we set $\nu_i = 2$ initially and then divide it by 2 if $Z_{\text{LR}}(\lambda_i)$ does not get updated for a fixed number of iterations. Note that, only when we have $\lambda \geq 0$, the solution from *Algorithm* 4 is a feasible one to the original problem. Therefore, we modify Eq. (18) as

$$\lambda_{i+1} = \max\{0, \ \lambda_i - \mu_i \cdot f(\lambda_i)\}, \tag{21}$$

E. Overall Procedure

Finally, *Algorithm* 5 shows the overall procedure of our approximation algorithm. *Lines* 1-2 are for the initialization,

Algorithm 4: Construction of Feasible Solution

```
1  Z* = 0, {ξ_e = 0, ∀e ∈ E};
2  for each e = (v_s, v) ∈ E do
3      get current bandwidth usage B̃_(v_s,v) on link (v_s, v)
       with the left-side of Eq. (7);
4      if B̃_(v_s,v) ≤ B_(v_s,v) then
5          ξ_(v_s,v) = 1;
6      end
7  end
8  for each (v_s, v) ∈ E do
9      if ξ_(v_s,v) = 1 then
10         continue;
11     end
12     for each flow k ∈ [1, |F|) that uses link (v_s, v) do
13         for each m ∈ M_{v_s} do
14             get B^m_{v_s} with Eq. (3);
15             m* = argmax_{m∈M_{v_s}}(B^m_{v_s}), flag = 0;
16             for each k' ∈ (k, |F|] do
17                 if flows k and k' only share v_s AND
                   ξ_(v_s,v^{k'}_s) = 0 then
18                     π^{k'}_{v_s,m*} = 1, f_{v_s,m*} = k';
19                     π^k_{v_s,m*} = 0, flag = 1, break;
20                 end
21             end
22             if flag = 0 then
23                 π^k_{v_s,m*} = 0, flag = 1;
24             end
25             update B̃_(v_s,v) and other related ones;
26             if B̃_(v_s,v) ≤ B_(v_s,v) then
27                 ξ_(v_s,v) = 1, break;
28             end
29         end
30         if ξ_(v_s,v) = 1 then
31             break;
32         end
33     end
34 end
35 for each v_s ∈ V_s do
36     for each m ∈ M_{v_s} do
37         k = f_{v_s,m};
38         if π^k_{v_s,m} = 1 then
39             if ⌈1 / t^m_{v_s}⌉ ≤ σ^k_{v_s} · f^r_k then
40                 Z* = Z* + η̄^k_{v_s,m};
41             else
42                 π^k_{v_s,m} = 0;
43             end
44         end
45     end
46 end
```

and then the while-loop uses N iterations at most to reduce the relative dual gap (RDG) between the upper- and lower-bounds (ub and lb, respectively) below a preset ratio γ (*Lines* 3-24). Here, the RDG is defined as

$$RDG = \frac{ub - lb}{ub}. \tag{22}$$

The values of N and γ are selected empirically, and we will analyze their impacts in the numerical simulations in Section VI. *Lines* 4-5 obtain the LR problem and solve it to get $Z_{\text{LR}}(\lambda_i)$ and $\{\pi^k_{v_s,m}\}$. Then, we check whether the obtained $Z_{\text{LR}}(\lambda_i)$ is smaller than the best-known upper-bound stored in ub. If yes, we update ub with $Z_{\text{LR}}(\lambda_i)$ and reset the counter n (*Lines* 6-7). Otherwise, we increase n by 1 (*Line* 9). Then, we check whether n is larger than the preset threshold T_h. If yes, it means that ub has not been updated for a while, and thus we should divide the step length μ_i by 2 (*Lines* 11-13).

Next, we obtain a feasible solution to the original problem (Z^*) based on the output of *Algorithm* 3, by using *Algorithm* 4 (*Line* 14), and update the lower-bound lb accordingly (*Lines* 15-17). *Lines* 18-20 check whether the current RDG is below the preset threshold γ. If yes, we have got a near-optimal solution to the original problem, and thus can stop the iterations. Otherwise, *Lines* 21-23 calculate μ_i and λ_{i+1} to prepare for the next iteration. The time complexity of *Algorithm* 5 is $O(N \cdot |V_s| \cdot \max(|M_{v_s}|) \cdot \max(|Z_{v_s}|))$.

Algorithm 5: Overall Procedure

```
1  i = 1, λ_i = 0, ν_i = 2, ub = +∞, lb = 0, n = 0;
2  preprocess the original problem with Algorithm 1;
3  while i ≤ N do
4      apply Algorithms 2 and 3 to get Z_LR(λ_i) and {π^k_{v_s,m}};
5      calculate {η̄^k_{v_s,m}} with Eq. (16);
6      if Z_LR(λ_i) < ub then
7          ub = Z_LR(λ_i), n = 0;
8      else
9          n = n + 1;
10     end
11     if n > T_h then
12         ν_i = ν_i/2, n = 0;
13     end
14     get a feasible solution and Z* with Algorithm 4;
15     if Z* > lb then
16         lb = Z*;
17     end
18     if (ub − lb)/ub ≤ γ then
19         break;
20     end
21     calculate μ_i with Eq. (20);
22     calculate λ_{i+1} with Eqs. (19) and (21);
23     i = i + 1;
24 end
25 use the most recent {π^k_{v_s,m}} to calculate the objective with
   Eq. (6);
```

If we denote the exact solution of the original problem defined by the ILP in Section IV as Z_{ILP}, we have $Z_{\text{ILP}} \leq Z_{\text{LR}}(\lambda_i) = ub$. Meanwhile, *Algorithm* 5 gets a feasible solution as $Z^* = lb$. Therefore, the approximation ratio of *Algorithm* 5 can be obtained as

$$\zeta = \frac{Z^*}{Z_{\text{ILP}}} = \frac{lb}{Z_{\text{ILP}}} \geq \frac{lb}{Z_{\text{LR}}(\lambda_i)} = \frac{lb}{ub} \geq 1 - \gamma, \tag{23}$$

which verifies that it is an approximation algorithm.

F. Extensions to Address More Sophisticated Network Models

The aforementioned algorithm design only considers one node attribute (*i.e.*, Z_{v_s} is one-dimensional). However, with minor modifications, our algorithms can also tackle the network-level optimization of Sel-INT orchestration where each node has multi-dimensional attributes, as long as the node attributes are independent among different nodes. Specifically, for such

a more sophisticated network model, we can redefine the network as an attributed graph $G\left(V, E, A^V\right)$, where V and E are still the sets of network nodes and links, respectively, while $A^V : V \rightarrow \mathbb{Z}^\zeta$ is an attribute function that maps a node v_s to a ζ-dimensional integer vector $\mathbb{Z}^\zeta$, for representing the attributes of the node. Then, the attributes of all the nodes in the network can be denoted as a matrix $A^{VI} \in \mathbb{Z}^{v_s,\zeta}$.

Then, the general principle of our algorithm design is still valid. We can come up with ζ constraints to describe the ζ-dimensional attributes, each of which can be formulated similarly as Eq. (10). Then, we can obtain A^{VI} as a diagonal matrix, because the node attributes are independent among different nodes. Note that, the preprocessing is still suitable for the attributed graph, and the Λ in the LR problem will accordingly become a vector of Lagrangian multipliers whose length equals the rank of A^{VI} (*i.e.*, ζ). Next, we can still leverage LR to relax the ζ hard constraints and put them into the objective. Finally, with a similar procedure of *Algorithm* 5, an LR-based approximation algorithm can be designed.

VI. Numerical Simulations for Network-level Optimization

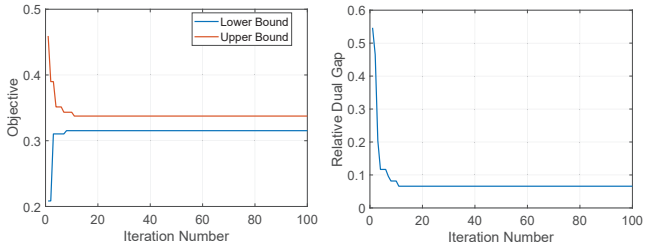
In this section, we discuss the numerical simulations that evaluate the performance of the algorithms for the network-level optimization of Sel-INT orchestration.

A. Simulation Setup

The simulations consider three network topologies with different sizes, *i.e.*, the NSFNET topology [45] that consists of 14 PDP-SWs and 6 DAs, the US-Backbone topology (USB) [45] that includes 24 PDP-SWs and 10 DAs, and a random topology (RT-50) that is generated with the GT-ITM tool [46] and have 50 PDP-SWs and 20 DAs. The average node degrees of NSFNET, USB and RT-50 are 2.80, 3.11, and 4.14, respectively. We assume that on each PDP v_s, there are $|M_{v_s}| = 10$ types of telemetry data for Sel-INT. For each type of telemetry data $m \in M_{v_s}$, the length of its INT field (s_m) is randomly selected from $[4, 20]$ bytes [25, 41], and the information gain ($\eta_{v_s,m}$) brought by collecting it distributes within $[10, 40]$ units. At each PDP-SW v_s, the packet sampling capacity (Z_{v_s}), which can be used for Sel-INT, is uniformly distributed within $[50, 100]$ kilo packets per second (Kpps). On each link e, the available bandwidth capacity that can be used for Sel-INT is randomly selected within $[300, 500]$ Mbps.

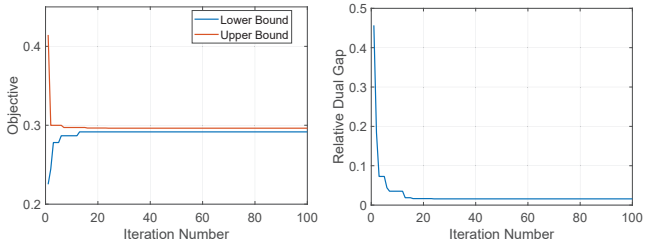
For each flow $k \in F$ that can be used for Sel-INT, its average packet rate (f_k^r) is randomly selected from $[5, 12]$ Kpps[1]. The source and destination of each flow are randomly selected, and its routing path is pre-calculated with the Dijkstra algorithm. For the ILP model, we set $\alpha = 1$ and $\beta = 0.1$ to make sure that maximizing Φ_g is the primary objective. The simulations are conducted on a computer with 3.20 GHz Intel i5-6500 CPU and 16 GB memory, and the simulation environment is MATLAB 2018a with GLPK v4.64. In the simulations, we average the results from 5 independent runs to get each data point, for ensuring sufficient statistical accuracy.

In addition to the ILP and the LR-based approximation algorithm, we also consider a heuristic. For the subproblem of *INT data selection*, the heuristic transforms it into $|V_s|$ independent 0-1 knapsack problems, and solves them with dynamic programming. For the subproblem of *INT data assignment*, it assigns each selected telemetry data to the first feasible flow that will introduce the smallest bandwidth overhead, until all the selected telemetry data has been assigned or the available bandwidth capacity for Sel-INT has been used up.



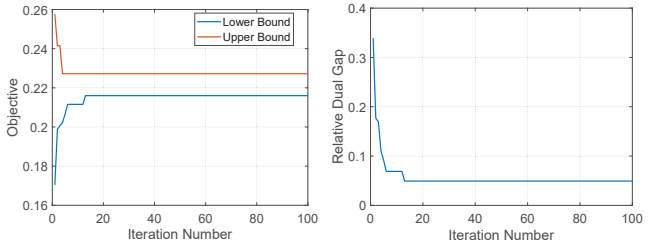
(a) Lower- and upper-bounds (b) Relative dual gap

Fig. 5. Convergence performance of *Algorithm* 5 on NSFNET topology.



(a) Lower- and upper-bounds (b) Relative dual gap

Fig. 6. Convergence performance of *Algorithm* 5 on USB topology.



(a) Lower- and upper-bounds (b) Relative dual gap

Fig. 7. Convergence performance of *Algorithm* 5 on RT-50 topology.

B. Convergence Performance of Approximation Algorithm

We first investigate the convergence performance of the LR-based approximation algorithm (*Algorithm* 5) with simulations on the three topologies (*i.e.*, NSFNET, USB and RT-50). Here, we empirically set $T_h = 7$, and assume that there are $|F| = 4$ flows in the network, and their routing paths have $[5, 7]$ hops. Figs. 5-7 show the simulation results. In Figs. 5(a), 6(a) and 7(a), we observe that for all the three topologies, the lower- and upper-bounds on the optimization objective in Eq. (6) converge

[1]Note that, the packets in each flow should have different sizes, which follow the distributions of Internet traffic [47]. Hence, as our Sel-INT selects packets to insert INT fields, it can always avoid long packets and prevent making packet sizes longer than the maximum transmission unit (MTU).

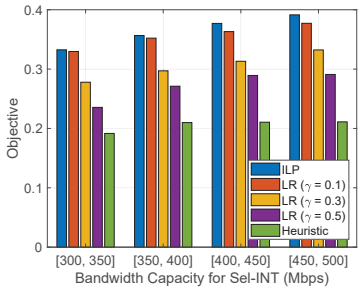
TABLE I
AVERAGE RUNNING TIME (SECONDS)

	ILP			Heuristic			*Algorithm* 5		
	NSFNET	USB	RT-50	NSFNET	USB	RT-50	NSFNET	USB	RT-50
$\|F\| = 4$	0.3203	0.3971	0.9118	0.0032	0.0036	0.0076	0.0315	0.0693	0.0967
$\|F\| = 6$	1.7048	2.5349	6.0418	0.0058	0.0065	0.0119	0.0456	0.1083	0.1219
$\|F\| = 8$	964.4011	–	–	0.0076	0.0082	0.0160	0.1258	0.1305	0.1349
$\|F\| = 10$	41379.9028	–	–	0.0092	0.0116	0.0191	0.1480	0.1772	0.1986
$\|F\| = 50$	–	–	–	0.0455	0.0648	0.1211	2.6533	3.2538	4.4431
$\|F\| = 100$	–	–	–	0.0889	0.1285	0.2968	5.5605	8.4921	9.0670

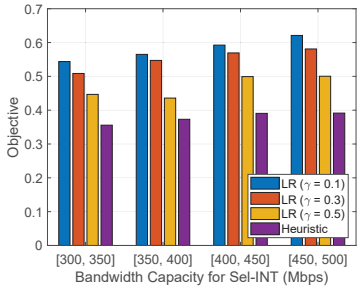
fast within 20 iterations. The convergence performance of *Algorithm* 5 can be seen even more clearly by checking the results on RDG in Figs. 5(b), 6(b) and 7(b), which show that the RDG can be reduced to below $\gamma = 0.06$ quickly. Hence, the simulation results verify that *Algorithm* 5 can converge fast to guarantee good time-efficiency.

C. Performance Benchmarking

Then, we run simulations to compare the performance of the ILP, approximation algorithm and heuristic on the network-level optimization for Sel-INT orchestration. The simulations consider both the small-scale problem with $|F| = 4$ and the large-scale one with $|F| = 50$. Meanwhile, to see the effect of available bandwidth capacity for Sel-INT (*i.e.*, $\{B_e\}$), we further divide the range of $\{B_e\}$ to four smaller ones, and perform simulations with each of them.



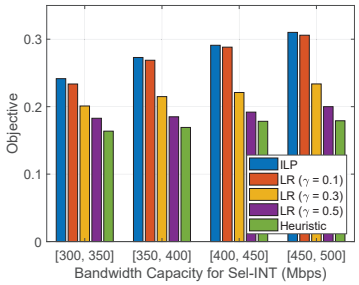
(a) $|F| = 4$



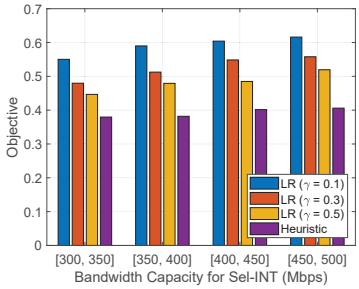
(b) $|F| = 50$

Fig. 8. Optimization objective with NSFNET topology.

1) Optimization Objective: Figs. 8-10 illustrate the algorithms' results on the objective. Here, for LR-based approximation algorithm (LR), we consider $\gamma = \{0.1, 0.3, 0.5\}$. In the figures, if we compare the objectives from a same algorithm for $|F| = 4$ and $|F| = 50$, we can see that the objective for $|F| = 50$ is always much larger. This is because when there are more flows to be considered for Sel-INT, the network can be better covered with the flows' paths and thus more telemetry data can be collected to have a larger information gain. Due to the time complexity of the ILP, it can only solve the problems with $|F| = 4$. In Figs. 8(a), 9(a) and 10(a), we can see that LR always provides near-optimal solutions whose gaps to the exact ones from the ILP satisfy γ (*i.e.*, the gaps decrease with γ). Among the three algorithms, the heuristic performs the worse in the scenarios with $|F| = 4$. The advantage of LR over the heuristic persists in the large-scale scenarios with $|F| = 50$, as shown in Figs. 8(b), 9(b) and 10(b).



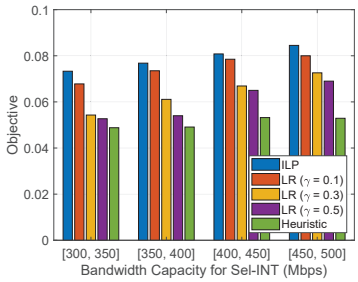
(a) $|F| = 4$



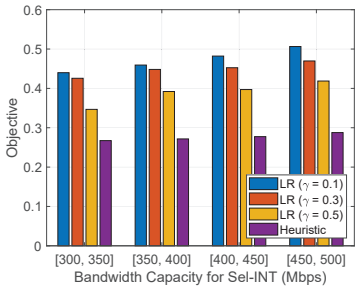
(b) $|F| = 50$

Fig. 9. Optimization objective with USB topology.

2) Coverage of Telemetry Data: Next, we compare the coverage of telemetry data from the algorithms. This time, we still consider the three topologies but only show the results of the LR and heuristic for $|F| = 50$. Here, we set $\gamma = 0.1$ for the LR. The simulations calculate the average ratio of collected types of telemetry data to all the possible types of telemetry data to collect, and plot the results in Fig. 11. It can be seen that the LR collects much more telemetry data than the heuristic, which further justifies its effectiveness on the network-level optimization of Sel-INT orchestration. Meanwhile, for both algorithms, the coverage of telemetry data increases with the available bandwidth capacity for Sel-INT.



(a) $|F| = 4$



(b) $|F| = 50$

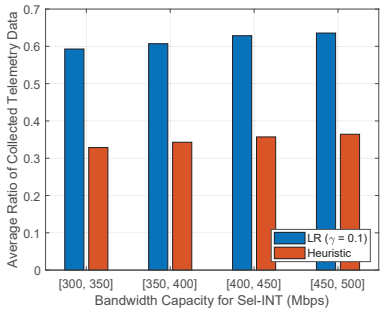
Fig. 10. Optimization objective with RT-50 topology.

3) Time Complexity: Finally, the simulations analyze the time complexity of the algorithms. We list their average running time in Table I. We observe that the running time of the ILP increases fast with the scale of the problem, which makes it become intractable when $|F|$ is larger than 8 for NSFNET and is larger than 6 for USB and RT-50. On the other hand, the LR (*Algorithm* 5) is much more time-efficient than the ILP and only takes $\sim$9 seconds to tackle the problem in RT-50 with $|F| = 100$. The heuristic runs the fastest among the three algorithms, and its running time can be one magnitude shorter than that of the LR. This is because the LR takes several iterations to obtain the near-optimal solutions.

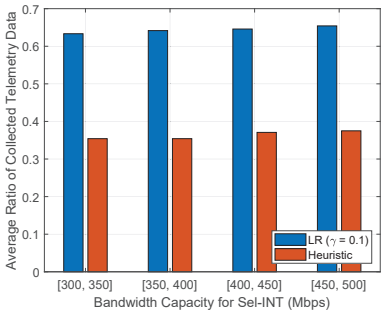
VII. Flow-level Optimization and Experimental Demonstrations

The network-level optimization orchestrates the Sel-INT schemes of flows in a network based on a few key parameters, which are assumed to be static (*e.g.*, $\{Z_{vs}\}$, $\{f_k^r\}$, and $\{B_e\}$). Nevertheless, these parameters are actually time-varying in a practical network. Therefore, the network-level optimization actually orchestrates the Sel-INT schemes of flows in a coarse but global manner, based on estimated values of the key parameters. It balances the tradeoff between monitoring accuracy/coverage and INT overheads for the whole network, and should be invoked from time to time when the key parameters vary a lot. However, considering the fact that the network-level optimization will update the Sel-INT schemes of all the flows and thus can bring in excessive operational complexity, we should not invoke it frequently.

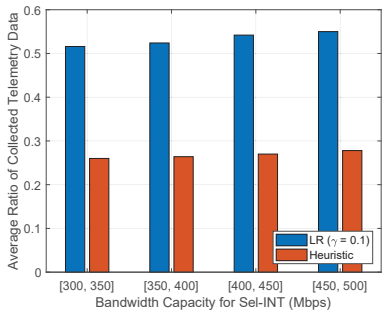
This motivates us to study the flow-level optimization that can change the Sel-INT scheme of each flow adaptively according to the status of a dynamic network environment. Hence, in between two adjacent network-level optimizations, we can leverage the flow-level optimization to orchestrate the Sel-INT schemes of flows in a self-adaptive but local manner. To this end, we can achieve network-wise orchestration of Sel-INT all the time to use INT wisely.



(a) NSFNET



(b) USB



(c) RT-50

Fig. 11. Average ratio of collected telemetry data.

A. System Design

Note that, the flow-level optimization needs to quickly adjust the Sel-INT scheme of each flow to adapt to network state changes, which cannot be done without the assistance from the control plane. Therefore, we expand our Sel-INT system developed in [30], and add in the support of Sel-INT orchestration in the control plane. Fig. 12 shows the system design to achieve the flow-level optimization. Here, the data plane consists of hosts, POF-based PDP-SWs that support Sel-INT, and home-made DAs that can extract telemetry data from the INT fields in packets and analyze the data for network monitoring and troubleshooting. The centralized controller in the control plane receives reports from the DAs, and further analyzes them with a DL-assisted module to ensure that the QoS demands of the applications (APPs) running in the data plane are handled well. For instance, for an application whose QoS is sensitive to the bandwidth available to its flow, the

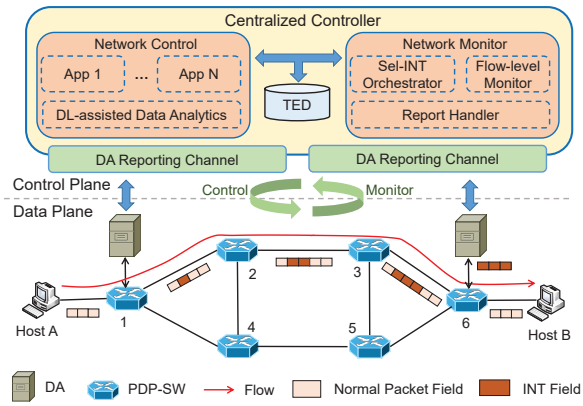


Fig. 12. Design of the network system to achieve flow-level optimization of Sel-INT orchestration.

controller predicts future bandwidth usage and invokes a path change if a congestion can be foreseen on its current path.

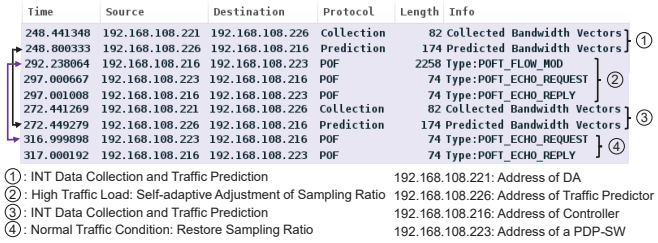
The traffic engineering database (TED) stores the provisioning schemes of active applications (*e.g.*, the routing paths of their flows, bandwidth usages, and other key parameters). In addition to the network control and TED, the network monitor is introduced in this work to achieve the flow-level optimization of Sel-INT orchestration. Specifically, the report handler processes the reports on network status from the DAs, the flow-level monitor not only observes but also forecasts the statistics of application flows, and based on the outputs from the flow-level monitor, the Sel-INT orchestrator calculates and updates the Sel-INT scheme of each flow to realize self-adaptive Sel-INT in a dynamic network environment.

In the network system, we implement the POF-based PDP-SWs based on the OpenvSwitch platform (OVS). Specifically, in [30], we expanded OVS v2.6.90 to make it support POF and Sel-INT, and realized a software-based PDP-SW, namely, OVS-POF, which can achieve a data-rate of 10 Gbps for 256-byte packets. The DA is home-made and can process packets that include INT fields at a speed up to 2 million packets per second (Mpps) [30, 34]. The hosts are commercial traffic analyzers that can generate/receive packets with various sizes at a data-rate up to 10 Gbps. The centralized controller is developed based on the famous ONOS platform. Note that, to show control plane operations clearly, we take the DL modules out of the controller and implement them based on TensorFlow on an independent server.

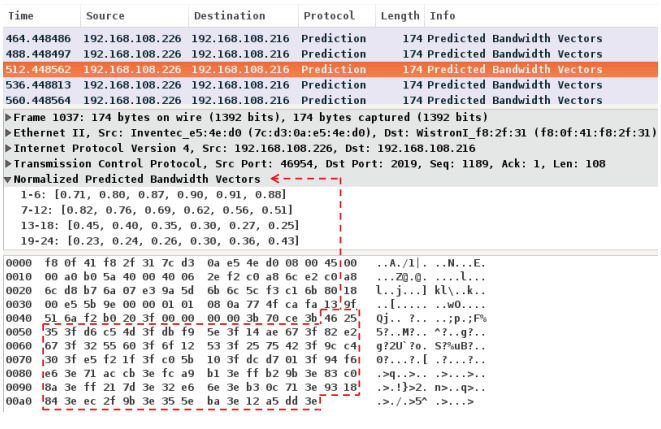
B. Experimental Demonstrations

To demonstrate the flow-level optimization of Sel-INT orchestration experimentally, we build a small but real network testbed whose data plane uses the configuration in Fig. 12 to have two hosts, six PDP-SWs, and two DAs. Each PDP-SW or DA is realized on a stand-alone Linux server, and the hosts, PDP-SWs and DAs are interconnected with 10GbE ports.

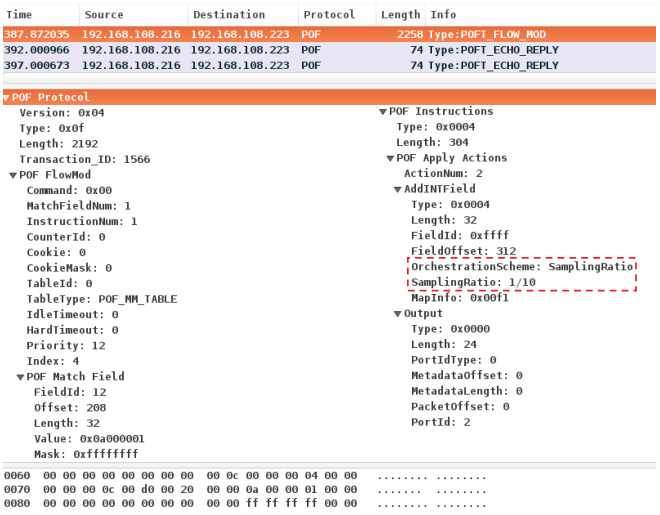
As a proof-of-concept demonstration, our experiment focuses on one of the most intimidating challenges of using INT in a dynamic network environment, *i.e.*, the bandwidth overheads of INT can cause congestions during rush hour. Therefore, the flow-level optimization should either forecast



(a) Flow of control messages



(b) Traffic prediction report to controller



(c) *FlowMod* message to update Sel-INT scheme on a PDP-SW

Fig. 13. Wireshark captures of control messages for flow-level optimization.

or quickly detect the congestions, and then adjust the Sel-INT schemes of flows accordingly to reduce the bandwidth overheads of INT. In the experiment, we let the DL modules predict future traffic for the controller based on the reports from the DAs. When the controller foresees a peak period of traffic, it will reduce the packet sampling ratio of Sel-INT to decrease the resulting traffic load, and it will restore the packet sampling ratio to ensure monitoring accuracy/coverage, when the peak period has been over. Therefore, self-adaptive network monitoring can be realized based on the closed-loop control driven by DL-based data analytics and Sel-INT (*i.e.*, the intelligent orchestration of "INT for INT").

In the experiment, we make the hosts generate flows whose traffic fluctuations follow practical traces taken from real-

world wide-area networks [48], and each traffic predictor is built based on the long/short-term memory based deep neural network (LSTM-DNN), which is known to be effective on forecasting time series [49]. Specifically, after proper training, the traffic predictor achieves an accuracy of 92.33% in the experiment. Fig. 13 shows the Wireshark captures of control messages used in the flow-level optimization of Sel-INT orchestration. The flow of control messages in Fig. 13(a) explains how the controller adjusts the Sel-INT scheme of a flow according to its traffic prediction, for self-adaptive network monitoring. During operation, a DA constantly collects the telemetry data about bandwidth usage of the flow by extracting and analyzing the related INT fields, and with the telemetry data, it assembles reports on the flow's bandwidth usage and sends them to the traffic predictor periodically.

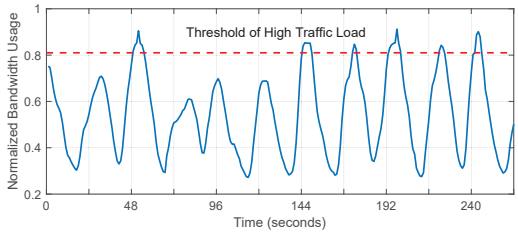
After receiving a report from the DA, the traffic predictor forecasts the traffic fluctuation in the next $\tau = 24$ seconds, and forwards the traffic prediction to the controller with the message format shown in Fig. 13(b). Next, the controller determines whether and how the current Sel-INT scheme of the flow should be updated, and instructs the related PDP-SWs to do so by installing *FlowMod* messages in them. Specifically, the decision-making procedure in the controller works as follows, if we denote the flow being monitored as f_k (*i.e.*, assuming it is the k-th flow in F). The controller first gets the flow's peak bandwidth usage $\hat{b}_{k,\tau}$ for the next period of τ. Meanwhile, two parameters were pre-defined in the controller: 1) the threshold of high bandwidth usage on a link (B_{th}), and 2) the maximum bandwidth that can be used for the Sel-INT on each flow ($\hat{B}_{\text{INT}}$). We also introduce a ratio $\eta \in (0, 1]$ to adjust the accuracy of the Sel-INT-based flow monitoring, *i.e.*, more telemetry data can be collected with the Sel-INT on the flow if η is larger, and *vice versa*.

Next, we consider the accuracy of the DL-based traffic prediction, and define the average accuracy of forecasting the traffic of flow f_k over a future period of τ as $p_{k,\tau}$. Note that, we can compensate the negative effect of non-ideal prediction accuracy by reserving bandwidth margin according to $p_{k,\tau}$ in the design of Sel-INT, to minimize the cases of unexpected congestions. Hence, we let the controller to determine the sampling ratio $r_{k,\tau}$ of flow f_k over the next period of τ as

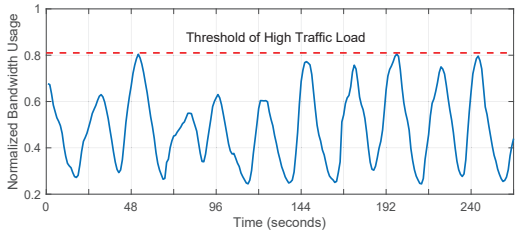
$$r_{k,\tau} = f\left(\left\lfloor \frac{\min\left[\eta \cdot \max(B_{th} - \hat{b}_{k,\tau}, 0), \hat{B}_{\text{INT}}\right]}{B_e} \right\rfloor, p_{k,\tau}\right), \quad (24)$$

where $f(\cdot)$ is a piecewise function obtained empirically based on numerous experiments conducted in our testbed.

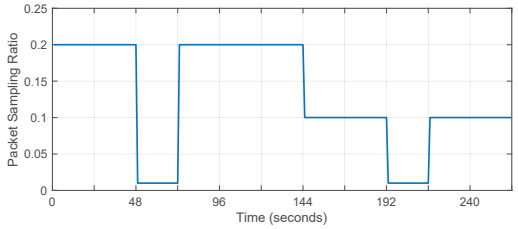
Fig. 13(c) illustrates the details of such a *FlowMod* message, which tells the PDP-SW to update the packet sampling ratio of Sel-INT to 0.1. Fig. 14(a) shows the bandwidth usage of the non-adaptive Sel-INT scheme. We can see that as the packet sampling ratio of Sel-INT does not get changed according to the traffic fluctuation of the flow, the bandwidth overheads of INT can cause high traffic load conditions frequently. On the other hand, with the flow-level optimization, the bandwidth usage in Fig. 14(b) avoids congestions successfully, and Fig. 14(c) shows that the packet sampling ratio of Sel-INT gets adjusted adaptively according to the traffic fluctuation.



(a) Bandwidth usage of non-adaptive Sel-INT



(b) Bandwidth usage of self-adaptive Sel-INT



(c) Packet sampling ratio of Sel-INT

Fig. 14. Self-adaptive adjustment of packet sampling ratio of Sel-INT.

VIII. Conclusion

This paper addressed the problem of using Sel-INT wisely in a network by formulating and solving the network- and flow-level optimizations of Sel-INT orchestration. Specifically, we studied how to adjust the Sel-INT schemes of flows to balance the tradeoff between monitoring accuracy/coverage and INT overheads in network- and flow-levels. We modeled the network-level optimization as an INT planning problem, which chooses the Sel-INT schemes of flows such that the information gain of INT can be maximized while the bandwidth overheads of INT can be minimized. We formulated an ILP model for the problem, proved its $\mathcal{NP}$-hardness, and leveraged LR to design a polynomial-time approximation algorithm for it. Numerical simulations verified that our proposed algorithm can provide near-optimal solutions time-efficiently.

Next, for the flow-level optimization, we considered a dynamic network environment, and investigated how to adjust the Sel-INT scheme of each individual flow timely and adaptively in it. Specifically, we combined DL-based traffic prediction with Sel-INT, and designed and experimentally demonstrated a POF-based system to realize self-adaptive orchestration of Sel-INT. Experimental results confirmed that our system can adjust the packet sampling ratio of Sel-INT timely and adaptively to avoid the congestion caused by INT overheads.

Acknowledgments

This work was supported in part by the NSFC projects 61871357, SPR Program of CAS (XDC02070300), and Fundamental Funds for Central Universities (WK3500000006).

REFERENCES

[1] Cisco Annual Internet Report 2020 (Accessed on Nov. 20, 2020). [Online]. Available: https://www.cisco.com/c/en/us/solutions/executive-perspectives/annual-internet-report/index.html.

[2] 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.

[3] J. Mukherjee and B. Ramamurthy, "Communication technologies and architectures for space network and interplanetary Internet," *IEEE Commun. Surveys Tuts.*, vol. 15, pp. 881–897, Second Quarter 2013.

[4] M. Zhang *et al.*, "Bandwidth defragmentation in dynamic elastic optical networks with minimum traffic disruptions," in *Proc. of ICC 2013*, pp. 3894–3898, Jun. 2013.

[5] W. Shi, Z. Zhu, M. Zhang, and N. Ansari, "On the effect of bandwidth fragmentation on blocking probability in elastic optical networks," *IEEE Trans. Commun.*, vol. 61, pp. 2970–2978, Jul. 2013.

[6] 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.

[7] A. Caiti *et al.*, "Linking acoustic communications and network performance: Integration and experimentation of an underwater acoustic network," *IEEE J. Ocean. Eng.*, vol. 38, pp. 758–771, Oct. 2013.

[8] 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.

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

[10] L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. of INFOCOM 2014*, pp. 1–9, Apr. 2014.

[11] 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.

[12] 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.

[13] 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.

[14] W. Fang *et al.*, "Joint spectrum and IT resource allocation for efficient vNF service chaining in inter-datacenter elastic optical networks," *IEEE Commun. Lett.*, vol. 20, pp. 1539–1542, Aug. 2016.

[15] Q. Sun, P. Lu, W. Lu, and Z. Zhu, "Forecast-assisted NFV service chain deployment based on affiliation-aware vNF placement," in *Proc. of GLOBECOM 2016*, pp. 1–6, Dec. 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] 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.

[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] 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.

[20] 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.

[21] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69–74, Mar. 2008.

[22] 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.

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

[24] S. Li *et al.*, "Protocol oblivious forwarding (POF): Software-defined networking with enhanced programmability," *IEEE Netw.*, vol. 31, pp. 58–66, Mar./Apr. 2017.

[25] C. Kim *et al.*, "In-band network telemetry (INT)," *Tech. Spec.*, Jun. 2016. [Online]. Available: https://p4.org/assets/INT-current-spec.pdf.

[26] Protocol Indepedent Forwarding (Accessed on Nov. 20, 2020). [Online]. Available: https://opennetworking.org/news-and-events/protocol-independent-forwarding/.

[27] H. Song, "Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane," in *Proc. of ACM HotSDN 2013*, pp. 127–132, Aug. 2013.

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

[29] 100G in-band network telemetry with Netcope P4 (Accessed on Nov. 20, 2020). [Online]. Available: https://www.netcope.com/Netcope/media/content/100G-In-band-Network-Telemetry-With-Netcope-P4.pdf.

[30] 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.

[31] 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.

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

[33] 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.

[34] 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.

[35] Deep insight (Accessed on Nov. 20, 2020). [Online]. Available: https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/network-analytics/deep-insight.html.

[36] P4 software switch - Behavioral Model version 2 (Bmv2) (Accessed on Nov. 20, 2020). [Online]. Available: https://github.com/p4lang/behavioral-model.

[37] F. Cugini *et al.*, "P4 in-band telemetry (INT) for latency-aware VNF in metro networks," in *Proc. of OFC 2019*, pp. 1–3, Mar. 2019.

[38] Y. Li *et al.*, "HPCC: High precision congestion control," in *Proc. of ACM SIGCOMM 2019*, pp. 44–58, Aug. 2019.

[39] A. Karaagac, E. De Poorter, and J. Hoebeke, "In-band network telemetry in industrial wireless sensor networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, pp. 517–531, Mar. 2020.

[40] Z. Liu *et al.*, "NetVision: Towards network telemetry as a service," in *Proc. of ICNP 2018*, pp. 1–2, Sept. 2018.

[41] R. Hohemberger *et al.*, "Orchestrating in-band data plane telemetry with machine learning," *IEEE Commun. Lett.*, vol. 23, pp. 2247–2251, Dec. 2019.

[42] G. Michail, "The 0-1 knapsack problem - an introductory survey," *Tech. Spec.*, 2016. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.47.4378.

[43] M. Held, P. Wolfe, and H. Crowder, "Validation of subgradient optimization," *Math. Program.*, vol. 6, pp. 62–88, Feb. 1974.

[44] D. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1999.

[45] Q. Lv, F. Zhou, and Z. Zhu, "On the bilevel optimization to design control plane for SDONs in consideration of planned physical-layer attacks," *IEEE Trans. Netw. Serv. Manag.*, in Press 2020.

[46] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. of INFOCOM 1996*, pp. 594–602, Mar. 1996.

[47] E. Garsva, N. Paulauskas, and G. Grazulevicius, "Packet size distribution tendencies in computer network flows," in *Proc. of eStream 2015*, pp. 1–6, Apr. 2015.

[48] 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

[49] 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, Sep. 2019.