Near-Optimal Probing Planning for In-Band Network Telemetry

Ariel G. Castro, Arthur F. Lorenzon, Fábio D. Rossi, Roberto I. T. da Costa Filho,
Fernando M. V. Ramos, Christian E. Rothenberg, Marcelo C. Luizelli

***Abstract*—In-band Network Telemetry (INT) is gaining traction as an advanced network monitoring approach. Despite a few recent initiatives to orchestrate the collection of in-band network statistics, state-of-the-art approaches fall short when it comes to efficiently collect telemetry items while subjected to real-world constraints. In this letter, we propose Probe Planning for In-Band Network Telemetry (P^{2}INT) to coordinate how probing packets are generated and routed to ensure that all links are covered so that the required in-band network telemetry data is collected. We theoretically formalize the problem as a Integer Linear Programming model and propose an efficient mathematical programming-based heuristic to solve it. Our results show that P^{2}INT outperforms the closest contender by a factor of up to 6x concerning the number of probing cycles generated.**

***Index Terms*—In-band Network Telemetry; INT; Math-heuristic; MILP; data plane programability; P4.**

I. Introduction

In-band Network Telemetry (INT) has recently emerged as a promising near real-time network monitoring to improve network visibility [1], [2], [3]. Due to the rich spectrum of benefits behind INT, there is increasing attention from the networking ecosystem fostered by the rapid adoption of programmable data planes and domain-specific networking description languages (e.g., P4 [4]). In short, INT consists of instrumenting the collection of low-level network statistics directly from the data plane. In the classic hop-by-hop INT (a.k.a INT-MD (eMbed Data)[1]), an INT source node embeds instructions into production network packets. Then, INT transit nodes embed metadata while an INT sink node strips the instruction out of the packet and sends the accumulated telemetry data to an INT collector.

In this work, we consider using probe packets to instruct network devices to collect telemetry data. Figure 1 illustrates the entire INT process. In the first step, probing packets are generated aiming at instrumenting the collection of telemetry data along a given path. For example, the red flow (i.e. f_1) – that is routed through the devices A, E, F, G, H, and I – carries instructions to collect telemetry data from devices A to H. In the second step, the collected telemetry data is extracted and reported to an INT collector.

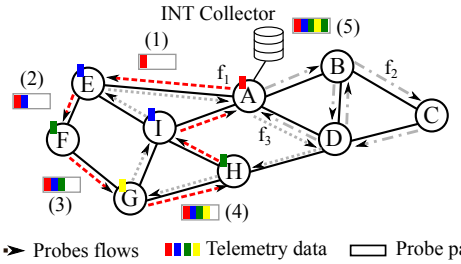


Fig. 1. Example of a solution for the probing planning problem, illustrating a snapshot where probing packets (f_1, f_2, f_3) collect telemetry data from selected network devices.

Recently, investigations have made the first efforts towards the orchestration of INT data collection to improve network-wide visibility. Liu et al. [2], Pan et al. [3], Bhamare et al.[5], and Geng et al. [6] have focused on performing network telemetry through active INT-based probing packets. These strategies have relied either on *Euler Circuits* [2], [3] or on actual routing paths [5], [6] to instrument the forwarding of probes. In turn, Marques et al. [7] and Hohemberger et al. [8] have focused on the embedding of INT data into production network packets. Marques et al. [7] designed heuristic approaches to orchestrate how network flow packets collect network telemetry data, while Hohemberger et al. [8] designed a machine learning-based model to wisely choose and collect INT data based on its importance. Further, others studies [9] have focused on the design of operational INT mechanisms and on reducing the amount of INT messages being reported.

Despite current efforts towards near real-time in-band network telemetry, the coordination of INT probing packets to collect network information efficiently is still full of gaps and challenges. The first attempts [2], [3], [5] to tackle this problem contributed with initial steps but suffer from (i) uncoordinated probing packet generation, and (ii) neglected capacity constraints. By using uncoordinated probes, there is an increasing transmission overhead from active probes to the INT collectors (as discussed later on). Furthermore, relaxing capacity constraints of probing packets simplifies the problem – but unrealistic from an operational point of view[2]. In this letter, we introduce P^{2}INT – Probe Planning for In-Band Network Telemetry – to coordinate how probing packets are generated and routed in order to ensure that all links are visited and all required INT data is collected. Although existing solutions have either focused on INT probes to monitor link

This research was partially supported by National Council for Scientific and Technological Development (CNPq) (grant 427814/2018-9), São Paulo Research Foundation (FAPESP) (grant 2018/23092-1), Rio Grande do Sul Research Foundation (FAPERGS) (grants 19/2551-0001266-7,20/2551-000483-0, 19/2551-0001224-1) and by the Portuguese national funds through FCT via UIDB/50021/2020 and PTDC/CCI-INF/30340/2017 (uPVN) projects.

Ariel G. Castro, Arthur F. Lorenzon, and Marcelo C. Luizelli are with the Federal University of Pampa, Brazil.

Fabio D. Rossi is with the Federal Institute Farroupilha, Brazil.

Roberto I. T. da Costa Filho is with Instituto Federal de Educação, Ciência e Tecnologia Sul-Rio-Grandense, Brazil.

Fernando M. V. Ramos is with the University of Lisbon, Portugal.

Christian E. Rothenberg is with the University of Campinas, Brazil.

[1]INT specification: https://github.com/p4lang/p4-applications/blob/master/docs/INT_v2_1.pdf

[2]For instance, in the Netronome SmartNIC architecture, the ingress timestamp has 64 bits, while the ingress port 16 bits.

connectivity (e.g. [2], [3]) or focused on the collection itself of INT data (e.g. [7], [8]), they still miss how to jointly optimize the way to collect telemetry data and cover network links. To tackle this problem, we theoretically formalize P^{2}INT as a Integer Linear Programming model. The model consists of a generalization of two well-known optimization problems – namely, Capacitated Arc Routing problem and Bin Packing problem [10] and, therefore, it is an NP-hard problem. We introduce a novel mathematical programming-based heuristic that wisely guides the MILP model to find a high-quality solution. Results show that P^{2}INT outperforms the state-of-the-art solution [3] by a factor of 6x related to the number of probes, at the same time that makes better usage of available resources (up to 3x) and decreases the transmission overhead to INT collectors (up to 2x).

II. P^{2}INT: Probing Planning for In-band Network Telemetry

A. Problem Overview

The P^{2}INT problem consists of defining optimized probing cycles to cover a given network infrastructure, i.e., in terms of telemetry data and network connectivity. It is noteworthy that the complete network coverage, both in terms of links and nodes, enables the assessment of end-to-end metrics based on different composition rules (e.g., multiplicative, additive, and concave) [11]. This approach is crucial for operating in large-scale networks such as the 5G device-to-device ecosystem, where path-based measurements are prohibitive due to the massive number of available paths. The P^{2}INT problem is not trivially solved. First, probing packets are space-bounded (i.e., w.r.t. bytes), and therefore it is infeasible (in most cases) to collect all network telemetry data with a single packet. Second, routing a probing packet is challenging. Probes need to be routed in such a manner that telemetry data requirements are met, while avoiding extra overheads on production network traffic (e.g., excessive generation of probing cycles). Figure 1 illustrates a network infrastructure with nine programmable forwarding devices (ranging from A to I), each having exactly one equal-sized telemetry data (represented by colored rectangles). These telemetry data represent data planes' internal states (e.g., queue occupancy or processing time), which are used by specialized monitoring applications [8] (e.g., DDoS detection). In the example, probing packets are limited to collect at most five telemetry data. There exists a set of active probing cycles (i.e., f_1, f_2, and f_3) which are responsible for continuously (i) collecting telemetry data and (ii) checking network connectivity. Probing cycles f_1, f_2, and f_3 are routed and instrumented to collect a given subset of telemetry data. For instance, probing cycle f_1 collect telemetry data from forwarding devices A to H, while probing cycle f_3 from devices D and I. Observe that all network links are covered by at least one probing cycle.

B. Model description and notation

The proposed optimization model considers a physical network infrastructure $G = (D, L)$ and a set of telemetry items V. Set D in network G represents programmable forwarding devices $D = \{1, ..., |D|\}$, while set L consists of unidirectional links interconnecting pair of devices $(i, j) \in (D \times D)$. Similarly to the recent literature [7], [8], we consider that there exists a set of telemetry items V available. Each forwarding device $i \in D$ is able to embed a subset of items $V_i \subseteq V$ into a probing packet. Each telemetry item $v \in V$ has its size defined by function $S : V \rightarrow \mathbb{N}^+$.

We consider there is at most $|P|$ probing cycles (i.e., $P = \{1, 2, ..., |P|\}$) to collect telemetry items from forwarding devices D. Packets in a probing cycle are encapsulated in a forwarding protocol, and therefore the amount of available space to embed telemetry items in packets is bounded by a constant, defined by function $U : P \rightarrow \mathbb{N}^+$ (e.g., $U(p)$ lower or equal to the MTU data link). The larger is this set P, the higher is the amount of decision variables in the model – and so the search space. A worst-case upper-bound for $|P|$ can be estimated as $|P| = |V| \cdot |D| \cdot |L|$. Probing cycles P are routed within the network infrastructure G – i.e., the packet is generated in a given source device, is routed through a subset of devices, and returns to its origin. We denote the cycle taken by the probing $p \in P$ as function $\mathcal{C} : P \rightarrow \{D_1 \times ... \times D_{|D|}\}$. Probing cycles $p \in P$ can collect telemetry items from forwarding devices $i \in \mathcal{C}(p)$. The set of telemetry items collected by probing cycle $p \in P$ is represented by pairs $(i, v) : i \in D, v \in V_i$ and is given by the function $\mathcal{T} : P \rightarrow \{D \times V\}$. A feasible cycle satisfy the upper-bound $U(p)$, that is $\sum_{i \in \mathcal{C}(p)} \sum_{v \in V_i : (i,v) \in \mathcal{T}(p)} S(v) \leq U(p)$. Observe that a given cycle $p \in P$ can visit a forwarding device $i \in \mathcal{C}(p)$ and not necessarily collect the set of telemetry items associated. Yet, our model does not restrict a cycle $p \in P$ to collect any telemetry item[3]. We denote the origin (starting/ending device) of each cycle $p \in P$ as function $O : P \rightarrow D$. Therefore, our model is generic to consider single- and multi-source probing cycle scenarios (i.e., cycles might start at different INT sources).

Given the problem input, the optimization problem seeks a feasible solution that minimizes the number of probing cycles, while visiting all network links and collecting the required telemetry data. The model output is denoted by a 3-tuple $\chi = \{Z, X, Y\}$. Variables from $Z = \{ z_{p,v,i}, \forall p \in P, v \in V, i \in D\}$ indicate that a forwarding device i embed telemetry item v into a probing packet from cycle p. Variables from $X = \{ x_{p,i,j}, \forall p \in P, (i, j) \in L\}$ indicates that network link $(i, j) \in L$ is used to route probing cycle $p \in P$. Last, variable $Y = \{ y_p, \forall p \in P\}$ is used to keep track of probing cycles used by the solution. Next, we describe the ILP formulation.

$$\textbf{Minimize} \quad \sum_{p=1}^{P} y_p \tag{1}$$

Subject to:

$$\sum_{p \in P} z_{p,v,i} = 1 \qquad \forall i \in D, v \in V_i \tag{2}$$

$$z_{p,v,i} \leq \sum_{j \in D} x_{p,j,i} \qquad \forall p \in P, i \in D, v \in V_i \tag{3}$$

$$z_{p,v,i} + x_{p,i,j} \leq 2 \cdot y_p \qquad \forall p \in P, (i, j) \in L, v \in V_i \tag{4}$$

[3]Telemetry items might be only available to specific queues inside the data plane.

$$\sum_{j \in D} x_{p,i,j} - \sum_{j \in D} x_{p,j,i} = 0 \qquad \forall p \in P, i \in D \quad (5)$$

$$\sum_{p \in P} x_{p,i,j} + x_{p,j,i} \geq 1 \qquad \forall (i,j) \in L \quad (6)$$

$$\sum_{i \in D} \sum_{v \in V_i} z_{p,v,i} \cdot S(v) + \sum_{i \in D} \sum_{j \in D} x_{p,i,j} \leq U(p) \qquad \forall p \in P \quad (7)$$

$$\sum_{i \in S} \sum_{j \in S} x_{p,i,j} \leq |S| - 1 \qquad \forall p \in P, S \subseteq \{D - O_p\}, |S| \geq 2 \quad (8)$$

$$z_{p,v,i} \in \{0,1\} \qquad \forall p \in P, v \in V_i, i \in D \quad (9)$$

$$y_p \in \{0,1\} \qquad \forall p \in P \quad (10)$$

$$x_{p,i,j} \geq 0 \qquad \forall p \in P, v \in V_i, i \in D \quad (11)$$

Constraint set (2) ensures that generated probing cycles collect the required network telemetry data. Constraint set (3) ensures that if telemetry item v is collected from forwarding device i, then there should have a probe being routed through i. Constraint set (4) accounts for the number of probing cycles in use. In short, it sets a cycle p as active whenever variables $z_{p,v,i} = 1$ or $x_{p,i,v} = 1$. Constraint set (5) ensures flow conservation on probing cycles. In other words, they generate probing cycles without ramification or self-loops. In turn, constraint set (6) guarantees a probing cycle covers at least one link direction. Constraint set (7) ensures that the available capacity is not violated either by the telemetry items collected or by the network links being covered. Observe that $\sum_{i \in D} \sum_{j \in D} x_{p,i,j}$ limits the probing cycle length. Constraint set (8) is the well-known sub-tour elimination constraints, ensuring that generated cycles are strongly connected [12]. Last, constraint sets (9)–(11) define the domains of output variables. It is worth mentioning that the complexity of the proposed model comes from (i) capacitated probe packets, (ii) non-uniform size of telemetry data, and (iii) cycle definition.

III. A Math-Heuristic Approach to P^{2}INT

To tackle the P^{2}INT complexity and come up with near-optimum solutions, we introduce a mathematical programming-based heuristic. To minimize the number of probing cycles in the solution χ, our strategy optimizes a few probing cycles at once, to merge them by reallocating telemetry data to other cycles. The idea consists of iteratively choosing a subset of probing cycles to be optimized (i.e., their variables Z, X, Y are freely changed), while the others remain fixed. Algorithm 1 presents the proposed approach. We first compute a feasible solution χ to the P^{2}INT problem (line 1). Then, we iteratively select a subset of k probing cycles (lines 5-10), and enumerate the list of variables $x_{p,i,j} \in \mathcal{D} \subseteq X$ related to them (line 11); variables listed in $\mathcal{D}$ will be subject to optimization, while others will remain unchanged (line 12).

We take advantage of meta-heuristic VNS (Variable Neighborhood Search) to systematically iterate over subsets of probing cycles. Further, we prioritize subsets with higher potential for improvement – i.e., probing cycles that might be merged (discussed in Subsection III-C). For each subset of probing cycles, we submit its set of variables $\mathcal{D}$ along with χ to a mathematical programming solver. The goal is to obtain a set of values to those variables listed in $\mathcal{D}$, so that a better solution is found. In case there is no improvement, we rollback

Algorithm 1 Overview of the fix-and-optimize heuristic.

Input: T_{global}: global time limit, T_{local}: time limit for each solver run, $\mathcal{K}_{init}, \mathcal{K}_{end}$: initial/final neighborhood size, $\mathcal{K}_{inc}$: increment for neighborhood size, $NoImprov_{max}$: max. rounds without improvement
Output: χ: best solution found to the optimization model

1: $\chi \leftarrow$ initial feasible solution
2: **if** a feasible solution does not exist **then** fail **else**
3: $k \leftarrow \mathcal{K}_{init}$
4: **while** T_{global} is not exceeded **and** $k \leq \mathcal{K}_{end}$ **do**
5: $\mathcal{N}_k \leftarrow$ current neighborhood, i.e. tuples of k probing cycles
6: $\mathcal{N}_{k,shr} \leftarrow$ tuples from $\mathcal{N}_k$, whose cycles share devices
7: $\mathcal{N}_{k,any} \leftarrow \mathcal{N}_k \setminus \mathcal{N}_{k,shr}$
8: $NoImprov \leftarrow 0$
9: **while** $\{\mathcal{N}_{k,shr}, \mathcal{N}_{k,any}\} \neq \emptyset$ **and** $NoImprov \leq NoImprov_{max}$ **do**
10: $\mathcal{T} \leftarrow$ next unvisited neighbor (w.r.t Equation 16)
11: $\mathcal{D} \leftarrow$ list of variables $x_{p,i,j}$ from cycles in neighbor $\mathcal{T}$
12: $\chi' \leftarrow$ solution χ optimized by the solver, under time limit T_{local}, and making variables not in $\mathcal{D}$ as fixed
13: **if** χ' is a better solution than χ **then**
14: update χ to reflect solution χ';
15: $k \leftarrow \mathcal{K}_{init}$;
16: **break**
17: **else**
18: $NoImprov \leftarrow NoImprov + 1$
19: **end if**
20: **end while**
21: **if** no improvement was made **then** $k \leftarrow k + \mathcal{N}_{inc}$ **end if**
22: **end while**
23: **return** χ
24: **end if**

and pick the probing cycle subset that follows. We run this process iteratively until a better solution is found. Once it happens, we replace the incumbent solution with χ' (line 14), and restart the process (i.e., $k \leftarrow \mathcal{K}_{init}$). This loop continues until we have explored the most promising combinations of available cycles, or T_{global} execution time is exceeded.

A. Obtaining an initial solution

The first step of our algorithm (line 1) is generating a feasible solution χ. The solution is one that satisfies all constraints, though not necessarily a high-quality one, in terms of used probing cycles. There are several ways to generate feasible solutions to P^{2}INT. We propose two approaches. The first consists of adapting the `Edge Randomization` (`ER`) heuristic – an approach widely applied in Arc Routing Problems. `ER` starts a probing cycle from a random unvisited network link. Then, the algorithm randomly chooses an adjacent forwarding device and collect as many network telemetry items as possible. While the probing capacity is not depleted, the algorithm keeps repeating this procedure. Once it happens, the probing cycle returns to its origin using the shortest-path approach. The procedure is repeated until all network telemetry items are collected and all network links are visited.

The second approach is based on recent state-of-the-art work `PathPlanning` proposed by Pan et al. [3].Their proposal does not consider probe capacity, nor data plane telemetry items. We adapt their DFS-like algorithm to consider both requirements in the best effort approach. Our adaptation of the proposed DFS-like strategy only moves on to a next network link *iff* there is enough capacity on the current probe to collect telemetry data and to return to its origin.

B. Neighborhood selection and prioritization

We carefully choose the subset of probing cycles $\mathcal{D} \in X$ that will be optimized. We explore the search space using

VNS. In a nutshell, VNS organizes the search space in k-neighborhoods. Each neighborhood is determined as a function of the incumbent solution (χ), and a neighborhood size k. We build a neighborhood as a combination of any k probing cycles. Formally, we define a k-neighborhood as a set composed of k-tuples $\mathcal{N}_k = \{\, p \,|\, p \subseteq P \wedge y_p = 1 \,\}$. The number of neighbors in a k-neighborhood is given by the binomial coefficient $\binom{\sum_{p=1}^{P} y_p}{k}$. We focus only on active probing cycles to build our neighborhood, since other variables in the model are easily inferred once the probing cycles are defined.

The time required by the solver to optimize a solution χ and a subset $\mathcal{D} \subseteq X$ is often small. However, processing every candidate subset $\mathcal{D}$ from the entire k-neighborhood is impractical. Therefore, we prioritize those neighbors that might lead to a better solution. We prioritize tuples in the k-neighborhood set $\mathcal{N}_k$ according to two observations: *(i)* it is more probable to merge probing cycles if they are not over-committed (i.e., the higher the residual capacity, the better); and *(ii)* it is easier to merge cycles that are close to each other. We define a tuple priority, as a function of its residual capacity. The residual capacity of a tuple $\mathcal{T} \in \mathcal{N}_k$ is given by $r : \mathcal{T} \rightarrow \mathbb{R}^+$, according to Equation 16.

$$r(\mathcal{T}) = \sum_{p \in \mathcal{T}} \left(U(p) - \Big(\sum_{i \in D} \sum_{v \in V_i} z_{p,v,i} \cdot S(v) + \sum_{i \in D} \sum_{j \in D} x_{p,i,j} \Big) \right) \quad (16)$$

We break down a k-neighborhood set into two distinct sets. The first one is formed by tuples whose probing cycles sharing forwarding devices ($\mathcal{N}_{k,shr}$) – i.e., $\cap_{p \in \mathcal{T}} \neq \emptyset$. The second set is formed by remaining tuples in $\mathcal{N}_k$ ($\mathcal{N}_{k,any} = \mathcal{N}_k \setminus \mathcal{N}_{k,shr}$), *i.e.* those tuples whose cycles do not share any forwarding device. We first process the tuples of $\mathcal{N}_{k,shr}$. Then, we process the remainder ones ($\mathcal{N}_{any}$). Last, our approach takes as input $\textit{NoImprov}_{max}$. It indicates the maximum number of iterations without improvement that is allowed over a given neighborhood. We stop processing the current neighborhood once *NoImprov* exceeds $\textit{NoImprov}_{max}$ (line 9).

IV. Evaluation

A. Setup

The proposed model was ran using IBM *CPLEX Optimization Studio* 12.9 to obtain optimum solutions, while the proposed heuristic approach was implemented using Java language. Experiments were performed on a machine with AMD Threadripper 2920X processor and 80 GB of RAM, using the Ubuntu 16.04 operating system. We considered different physical network instances that were generated with Brite [13], following the Barabasi-Albert model [14]. We used physical network infrastructures varying from 10 to 200 forwarding devices. We vary the amount of available space to embed telemetry items in probing packets (i.e. $U(p)$) from 100 to 1500 Bytes. Further, we assume that forwarding devices have from 2 to 8 possible telemetry items to export, varying its size $S(v)$ uniformly from 2 to 20 Bytes [3]. P^{2}INT considers the following parameters $T_{global} = 6h$, $T_{local} = 600s$, $\mathcal{K}_{init} = 2$, $\mathcal{K}_{end} = 4$, $\mathcal{K}_{inc} = 1$, and $\textit{NoImprov}_{max} = 15$. The fine-tuning and sensitive analysis of these parameters is out of the scope of this work. For example, P^{2}INT can trade solution quality for resolution time by adjusting T_{global} and T_{local}. Using the t-test method, we found that 30 runs of each experiment is enough to achieve a confidence level 95% or higher.

Baseline. We compare P^{2}INT against (i) the optimal solution (`OPT`), (ii) the `Edge Randomization` (`ER`), and (iii) the recent state-of-the-art work `PathPlanning` (`PP`) [3].

Reproducibility. Our implementation is publicly available in order to encourage full reproducibility of our experiments[4] and foster the design of new solutions.

B. Results

We analyze the quality of the proposed approach by evaluating: (i) the number of probing cycles generated; (ii) the resource usage of probing cycles; (iii) the data transmission overhead to INT collectors; (iv) the INT collector usage; and (v) the network link coverage.

Probing cycles. Figure 2(a) illustrates the amount of probing cycles generated for an increasing size of network infrastructures (from 10 to 200). P^{2}INT comes up with quality-wise solutions compared to the optimal and the state-of-the-art approach `PP`. Our solution is able to approach the optimal value for small network infrastructures (up to 20 nodes)[5] At the same time, the `PP` produces solutions with up to 2x the number of probes considering small networks. For medium- to large-scale networks, P^{2}INT produces (on average) solutions with 2.2x and 3.70x fewer cycles compared to `PP` and `ER`, respectively. This behavior is explained by the ability of P^{2}INT to jointly route probing packets and collect items.

Probe scalability. Figure 2(b) depicts the impact of probing packet capacity with respect to the number of generated cycles. For the purpose of this evaluation, we show the results of a 50 node network infrastructure[6]. As the probe capacity increases, we observe a sharp reduction in the number of probing cycles – as there is more room to accommodate network telemetry data. When comparing P^{2}INT to its contenders, we observe that it is able to generate solutions with up to 5.5x and 4.6x fewer cycles than `PP` and `ER`, respectively – e.g., to $U(p) = 1500$.

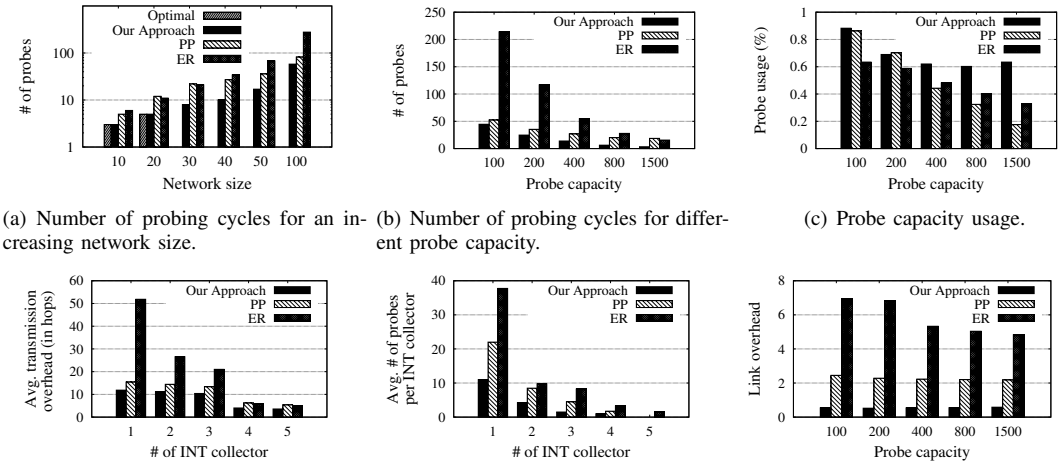
Probe capacity. Figure 2(c) illustrates the average probing capacity usage by generated cycles. Observe that P^{2}INT can utilize up to 3x more available capacity than `PP` (e.g., $U(p) = 1500$). On average, P^{2}INT utilizes 70% of available resources, while the other strategies (`PP` and `ER`) 48% and 50%, respectively. By using available resources efficiently, P^{2}INT produces solutions with minimum overheads.

Network efficiency. Probing cycles might be computed in a way that they are not routed through an INT collector. In this case, at some point in the generated cycle, the collected INT data needs to be sent to a given monitoring sink. For the purpose of this evaluation, we consider that there are up to five INT collectors placed optimally according to the requirements of each solution. In other words, the INT collectors were placed connected to a forwarding device in a way that the

[4]Available implementation of our simulation: https://anonymous.4open.science/r/de4b2622-2b86-457b-82a0-fe2ad10004d8/

[5]Optimal solutions for large-scale ($|D| > 20$) networks are unfeasible due to NP-hardness. For small instances, the computing time surpasses 24h.

[6]The results for other network infrastructures follow the same behavior.



(a) Number of probing cycles for an increasing network size.

(b) Number of probing cycles for different probe capacity.

(c) Probe capacity usage.

(d) Transmission overhead.

(e) Collector load.

(f) Link overhead.

Fig. 2. In-band probing cycles performance metrics.

distance (in hops) to probing cycles is minimized. Figure 2(d) illustrates the transmission cost (in hops) to the closest INT collector. We opt to illustrate this cost in hops as the volume of transmitted data depends on the probe capacity/usage and the frequency that probe packets are generated. Observe that the more INT sinks are available in the infrastructure, the lower is the transmission overhead. Also, note that P^{2}INT keeps this transmission cost as low as possible even when there exists just one INT collector. On average, `PP` and `ER` generate solutions with 1.38x and 2.26x higher transmissions overheads than P^{2}INT, respectively.

Collector load. Figure 2(e) shows the collector load as the number of probing cycles assigned to each INT collector. Considering two INT collectors, solutions can cover 83% (our approach), 82% (`PP`) and 78% (`ER`) of all probing cycles. Note that the observed controller load can be substantially reduced if a data plane filtering mechanism is considered (e.g., [9]).

Network coverage. Figure 2(f) depicts the network link coverage as the average probing cycles per network link. Higher values indicate that network links are being over-covered (i.e., multiple times), representing a waste of resources. P^{2}INT, on average, keeps this value close to one, while the other strategies produce solutions with network links begin covered by up to seven probing cycles (i.e., 7x more than the necessary).

V. Final Remarks

In this letter, we formalized the Probing Planning for In-band Network Telemetry (P^{2}INT) employing a MILP model and introduced a scalable mathematical-based heuristic to solve it. While our approach outperforms state-of-the-art heuristics (e.g., factor 6 w.r.t probing cycles), it is still limited to (i) static solutions over time (i.e., probing cycles do not change); (ii) fixed-throughput of probing packets (i.e., all probing cycles operates at the same packet rate); and (iii) agnostic to network services (i.e., probing cycles are not aware of running functions or services). Addressing these limitations from the theoretical and operational point of view (e.g., control and data plane integration) is part of our future work.

References

[1] V. Jeyakumar, M. Alizadeh, Y. Geng, C. Kim, and D. Mazières, "Millions of little minions: Using packets for low latency network programming and visibility," *ACM SIGCOMM CCR*, vol. 44, no. 4, pp. 3–14, 2014.

[2] Z. Liu, J. Bi, Y. Zhou, Y. Wang, and Y. Lin, "Netvision: Towards network telemetry as a service," in *IEEE ICNP*, Sep. 2018, pp. 247–248.

[3] T. Pan, E. Song, Z. Bian, X. Lin, X. Peng, J. Zhang, T. Huang, B. Liu, and Y. Liu, "Int-path: Towards optimal path planning for in-band network-wide telemetry," in *IEEE INFOCOM*, Apr 2019, pp. 1–9.

[4] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM 14*, vol. 44, no. 3, pp. 87–95, Jul. 2014.

[5] D. Bhamare, A. Kassler, J. Vestin, M. A. Khoshkholghi, and J. Taheri, "Intopt: In-band network telemetry optimization for nfv service chain monitoring," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–7.

[6] Y. Geng, S. Liu, Z. Yin, A. Naik, B. Prabhakar, M. Rosenblum, and A. Vahdat, "SIMON: A simple and scalable method for sensing, inference and measurement in data center networks," in *USENIX NSDI 19*. Boston, MA: USENIX Association, Feb. 2019, pp. 549–564.

[7] J. Marques, M. Luizelli, R. Da Costa, and P. Gaspary, "An optimization-based approach for efficient network monitoring using in-band network telemetry," *Journal of Internet Services and Applications*, no. 1, p. 16, Jun 2019.

[8] R. Hohemberger, A. G. Castro, F. G. Vogt, R. B. Mansilha, A. F. Lorenzon, F. D. Rossi, and M. C. Luizelli, "Orchestrating in-band data plane telemetry with machine learning," *IEEE Communications Letters*, vol. 23, no. 12, pp. 2247–2251, 2019.

[9] J. Vestin, A. Kassler, D. Bhamare, K. Grinnemo, J. Andersson, and G. Pongracz, "Programmable event detection for in-band network telemetry," in *2019 IEEE 8th International Conference on Cloud Networking (CloudNet)*, 2019, pp. 1–6.

[10] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. NY, USA: W. H. Freeman & Co., 1979.

[11] R. C. Filho, W. Lautenschläger, N. Kagami, M. Luizelli, V. Roesler, and L. Gaspary, "Scalable qoe-aware path selection in sdn-based mobile networks," in *IEEE INFOCOM*, April 2018, pp. 989–997.

[12] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a large-scale traveling-salesman problem," *Journal of the Operations Research Society of America*, vol. 2, no. 4, pp. 393–410, 1954.

[13] A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: an approach to universal topology generation," in *IEEE MASCOTS 2001*, 2001.

[14] R. Albert and A.-L. Barabási, "Topology of evolving networks: Local events and universality," *Physical Review Letters*, vol. 85, pp. 5234 – 5237, Dec 2000.