INTView: Adaptive Planner for In-Band Network Telemetry without Detours

Mingtao Ji†, Chenwei Su†, Yuting Yan†, Zhuzhong Qian†*, Yu Chen†, Yibo Jin†, Sheng Zhang†, Baoliu Ye†
†State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
Email: qzz@nju.edu.cn

***Abstract*—Network visualization is essential for network operators to diagnose ongoing network failures and understand the quality of the network. In-Band Network Telemetry (INT) supports network visualization by inserting P4 switch state information (e.g., queue length, hop latency, and link utilization) into the specific INT packets. In order to achieve network-wide coverage, paths of INT packets need to be delicately designed to ensure non-overlapping, high performance, and low overhead. However, existing INT path planning solutions ignore capturing the dynamic network status and fail to obtain the optimum. In this paper, we model the INT path planning based on the directed Edge Cover problem upon the dynamic network status, with the objective of minimizing the INT path latency. Although it is actually a non-linear integer program problem, by adopting delicate transformations, we design approximation algorithms with performance guarantees. We implement our system prototype INTView based on INTCollector upon real devices, i.e., Barefoot Wedge100BF and Inspur Rack. Extensive evaluation upon realistic settings shows that our proposed algorithm achieves 2x performance improvement regarding the completion time, compared with state-of-the-art schemas.**

I. Introduction

Network visualization [1–3] is essential for network operators to diagnose and troubleshoot ongoing network problems since it helps operators understand the quality of the network and make better decisions. But traditional network monitoring technique, e.g., SNMP [4], fails to support network visualization, due to its coarse-grained period and high resource cost. In-Band Network Telemetry (INT) [5] is proposed to achieve network visualization by injecting specific INT packets to collect the hop-by-hop information (e.g., queue depth, queuing latency, etc.) stored in P4 switch, as is shown in the left of Fig. 1. Compared with SNMP, INT supports fine-grained management and low overhead monitoring. For example, INT powered by P4 switch provides a sub-second monitoring period and supplies nanosecond information, which boosts rapid development of network visualization [6].

In order to achieve network-wide visualization, the paths of INT packets need to cover the total network to avoid missing information [1]. However, planning paths for those INT packets suffers several severe challenges as follows:

Firstly, planned paths for INT should be guaranteed non-overlapping as much as possible, since such overlaps often lead to high redundancy and bandwidth waste. To be specific, the number of INT probes for each telemetry job is often massive because the frequency of INT is usually extremely high. Particularly for high-precision monitoring applications, overlapping paths often leads enormous bandwidth consumption. Secondly, INT for large scale network needs multiple paths, each of which corresponds to one INT packet, to cover different parts of network. On the one hand, the number of INT paths should be as small as possible since too many paths lead to a large amount of INT packets which will cause high overhead in the controller. On the other hand, the length of each INT path should also be as small as possible since a long path will lead to high latency. However, they are often contradictory, since shorter paths mean more paths are required for each INT. Thirdly, in order to satisfy the timeliness of the centralized controller, the length of different INT paths should be as balanced as possible. In general, the completion of INT job often depends on the last INT packet from the longest path to come back, i.e., the straggler. However, maintaining balanced INT paths is difficult [1, 2] due to the existing constraints such as non-overlaps, uncertainty of topology, etc.

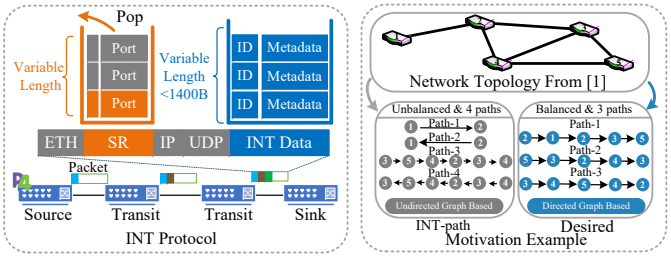


Fig. 1: In-band Network Telemetry and Motivation Example

However, the existing solutions often fail to face these challenges. For example, some works consider path planning as a vertex cover problem, assuming that one INT packet can obtain the total data stored in the P4 switch when it passed through [7, 8]. However, this assumption is invalid in current mainstream P4 switch, e.g., Barefoot Wedge100BF, since INT packet only fetches the data from its corresponding egress port instead of all ports. Based on this, some works consider path planning as an edge cover problem, but they often fail to obtain an excellent solution since they plan INT path based on undirected graph [1, 2]. For example, such works often plan unbalanced paths, as shown in the right of Fig. 1, which leads to straggler easily, postponing the completion.

Quite different from these works, we make INT path planning by treating network as a directed graph and obtain the

*The Corresponding Author is Zhuzhong Qian. This work is partially supported by National Science Foundation of China, under grant No. 61832005; China University Industry Research Innovation Foundation, under grant No. 2021FNA04005. And this work is also partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization. Thank Radin and Tina for their contributions to this paper.

optimum solutions on the same condition of INT-Path [1], as shown in the right of Fig. 1. We are the first to propose to make INT path planning based on the network status, i.e., link in network with latency. Since the latency of each link is often divergent, our path planning for INT is more reasonable in reality. Besides, the latency of each link often changes over time, which is considered in our problem. To be specific, we conclude our contributions as follows:

- To overcome the above challenges, we model the above mentioned directed graph based path planning problem and then design approximation algorithms, i.e., RMA and ABA, with theoretical performance guarantees, both of which achieve no overlap and low overhead.
- We reconstruct the INT protocol [9, 10] based on the source routing (SR) protocol, which can support our designed two algorithms. Specifically, the above algorithms output the paths as a series of port numbers, and INT protocol forwards packets according to them.
- We implement a system prototype, i.e., INTView, based on INTCollector [11], which achieves network visualization with low overhead and deploy it on real devices. We evaluate our algorithms with large scale simulations whose results show that they achieve 2x performance improvement compared with state-of-the-art schemas.

II. System Model And Problem Formulation

A. Preliminary Analysis and Challenges

In-Band Network Telemetry (INT): INT is wildly used to monitor networks by collecting the switch metadata (e.g., queuing latency, queuing length, etc.) hop by hop at the data plane in real-time. In general, INT Source Node is the first hop along each INT path, which is responsible for adding INT header. INT Sink Node is the last hop, which sends the INT packets to controllers who are responsible for collecting all the INT packets from the corresponding INT paths. Since the scale of network is massive, there are often several INT Source Nodes and several INT Sink Nodes.

Challenges to plan INT paths: In general, INT Source Node and Sink Node should be chosen carefully based on the complete network view obtained before, and there are three goals: In order to reduce unnecessary bandwidth occupation, INT paths should be non-overlapping. Then, the number of paths should be as small as possible, in order to lessen the processing overhead of the INT packets at controller. Finally, the length of paths should be as balanced as possible to meet the timeliness of the centralized controller.

Challenges to build math model: It is even hard only to formulate the Edge Cover problem. Although there are similar problems (e.g., CPP [12] or MPCPP [13]), they assume each node only plays one role, i.e., INT Source Node or INT Sink Node. However, in our problem, it is possible that one node plays different roles in different INT paths.

To overcome the above challenges, we make a formulation for our problem based on the directed graph with the Euler circuit. Actually, the network topology is a directed graph, since each link between any two switches has two directions. Thus, we see the network topology as a directed graph, which is greatly different from the previous works [1, 2]. Then, we have the following lemma for this directed graph.

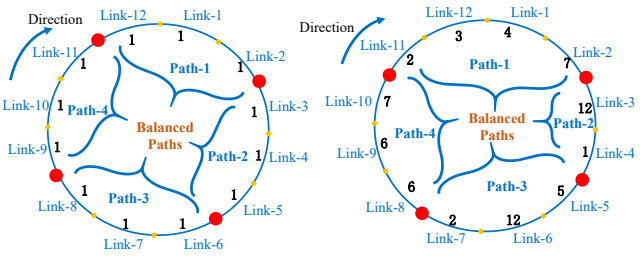


Fig. 2: Scenarios for Network-unaware and Network-aware

Lemma 1. *There exists Euler circuits [14] in the directed graph formed by network topology.*

Proof. According to the theory of Euler Circuits [15], if each node of a directed graph has equal in-degree and out-degree, the directed graph has an Euler circuit. Obviously, this is satisfied since each edge has two directions. □

Based on Lemma 1, we model our problem upon Euler circuit. As shown in the left of Fig. 2, we consider to cut apart this circuit by placing several red vertexes. The path formed by two consecutive red vertexes is the final INT path. The number of red vertexes represents the number of paths. INT-Path [1] is based on network-unaware where the weight of edges equals to 1. Obviously, we can always get the optimal solution for this case as shown in the left of Fig. 2. Furthermore, we consider a network-aware scenario where the weight of edges corresponds to the latency in each link.

B. Problem Formulation and Transformation

INT circuits: In order to formulate this, we first find the Euler circuits from the graph formed by our network topology. For each circuit, we denote it by E_c, $c \in \mathcal{C}$, where $\mathcal{C}$ is the set of Euler circuits upon our network topology. Furthermore, we use $E_{c,i}$ to represent the latency in link i, where $i \in E_c$.

Control Decisions: As shown in Fig. 2, any two consecutive red vertexes define a path. We use K to indicate the number of INT paths, i.e., the number of red vertexes in Fig. 2 and define $\mathcal{K} = \{1, 2, ..., K\}$. Each INT path-$j$, $j \in \mathcal{K}$, consists of several links with latency. We use $I_{i,j,c}$ to indicate whether link i is put into path-j, for circuit-c.

Control Problem $\mathscr{P}^O$: With the above models, we formulate the following optimization problem of minimizing the maximum INT path's latency among all the circuits:

$$
\begin{aligned}
[\mathscr{P}^O] \quad Min \quad & \min_{c}\{\max_{j}\{\sum\nolimits_i I_{i,j,c} * E_{c,i}\}\} \\
s.t. \quad & C_1 : \sum\nolimits_i (I_{i,j,c} - I_{(i+1)\%|E_c|,j,c})^2 = K, \quad \forall j, c, \\
& C_2 : \sum\nolimits_j I_{i,j,c} = 1, \quad \forall i, c, \\
& C_3 : K * T \leq \beta, \\
var. \quad & I_{i,j,c} \in \{1, 0\}, \forall i \in E_c, \forall j \in \mathcal{K}, \forall c \in \mathcal{C}, \\
& K \in N^+.
\end{aligned}
$$

Algorithm 1 Random Merge Algorithm (RMA)

Input: Euler Circuits E_c; The number of INT paths K;
1: Initial V according to E_c;
2: **while** $|V| > K$ **do**
3: Choose a uniform $v \in V$;
4: Contract(v);
5: **end while**
6: **return** The K INT paths;

C_1 ensures that all links are divided into K groups and all links of each group form a non-overlapped INT path. C_2 implies that each link should be assigned to only one path and C_3 ensures that the number of paths should be bounded because the controller often has limited processing capacity. And T represents the bandwidth overhead incurred by each INT packet while β is the controller's bandwidth.

The number of decision variables $I_{i,j,c}$ is non-deterministic, which is determined by K. Observing this problem, we find K needs to be decided before $I_{i,j,c}$ and it is affected by the processing capacity of the controller. In general, the processing capacity of each controller server has an upper bound [16]. Given a fixed controller server, K is a constant and we can test it in the real production environment. Then, we transform our original problem $\mathscr{P}^O$ as follows:

$$
\begin{aligned}
[\mathscr{P}^K] \quad Min \quad & \min_{c}\{\max_{j}\{\sum_{i} I_{i,j,c} * E_c\}\} \\
s.t. \quad & C_1, C_2, \\
var. \quad & I_{i,j,c} \in \{1,0\}, \forall i \in E_c, \forall j \in \mathcal{K}, \forall c \in \mathcal{C}.
\end{aligned}
$$

However, making decisions for such problem still has challenges since the constraints with square form. In the following, we design appropriate algorithms and give theoretical analysis.

III. Algorithm and Analysis

In order to solve our proposed problem effectively, we design two approximation algorithms, i.e., Random Merge Algorithm (RMA) and Average Based Algorithm (ABA), both of which have performance guarantees.

A. Random Merge Algorithm

We first design a baseline to solve problem $\mathscr{P}^K$, named, RMA, based on random strategy. RMA obtains excellent solutions after long-running. Specifically, we first introduce a function named $Contract()$ which merges two edges connecting v into one edge. After merging them, the weight of new edge equals to the sum of the two old edges. As shown in line 1 of Algorithm 1, the set V is initialized according to Euler Circuits E_c. Specifically, we set up a virtual vertex for any two adjacent edges. Subsequently, in line 4, this algorithm chooses a vertex randomly and merges the two edges connecting it.

Then, we give a theoretical analysis for such algorithm. Let set $\mathcal{S}^* = \{v_1^*, v_2^*, ..., v_{|E_c|-K}^*\}$ be the optimal solution and the probability of obtaining C is usually certain.

Lemma 2. *For optimum C, Pr[$\mathcal{S}^*$ is returned] =* $\frac{k!}{\prod_{i=1}^{K}(N-i+1)}$.

Algorithm 2 Average Based Algorithm (ABA)

Input: Euler Circuits E_c; The number of INT paths K;
1: SUM=$\sum_i E_{c,i}$, MAX=$\max_i E_{c,i}$, AVG=SUM/K;
2: Take consecutive edges starting from $i = argmax\{E_c\}$;
3: **for** $j = 1$ to K **do**
4: Initialize set S_j;
5: **if** j%2 == 0 **then**
6: Take consecutive edges from E_c, remove them, put them into S_j, ensure $AVG < sum(S_j)$;
7: **else**
8: Take consecutive edges from E_c, remove them, put them into S_j, ensure $AVG > sum(S_j)$;
9: **end if**
10: **end for**
11: **return** The K set S_1, S_2, ..., S_K;

Proof. The probability of returning $\mathcal{S}^*$ can be calculated:

$$
\begin{aligned}
&Pr[Return\ \mathcal{S}^*] = \prod_{i=1}^{N-K} Pr[v_i \notin \mathcal{S}^* | v_1, v_2, ..., v_{i-1} \notin \mathcal{S}^*] \\
&= \prod_{i=1}^{N-K} (1 - K/(N-i+1)) \\
&= ((N-K)/N) * ((N-K-1)/(N-1)) * ... * 1/(K+1) \\
&= K! / \prod_{i=1}^{K} (N-i+1).
\end{aligned}
$$

□

Based on this, we can calculate the probability of failing to return the optimal solution after running δ rounds. The time complexity of the algorithm RMA is related to $|V|-K$, which is a polynomial time algorithm.

Theorem 1. *Based on Lemma 2, running RMA for repeat independently δ times, the probability of failing to find the optimum solution set is less than $\frac{1}{e}$, where δ =* $\frac{\prod_{i=1}^{K}(N-i+1)}{K!}$.

Proof. We can calculate the probability as follows:

$$
\begin{aligned}
&Pr[Fail\ return\ \mathcal{S}^*] = (1 - K! / \prod_{i=1}^{K}(N-i+1))^{\frac{\prod_{i=1}^{K}(N-i+1)}{K!}} \\
&= (1 - 1/\delta)^{\delta} \leq 1/e.
\end{aligned}
$$

□

According to the analysis results, the performance of RMA is related with the number of iteration rounds. We further design an algorithm with high efficiency and low overhead.

B. Average Based Algorithm

Then we propose ABA, which obtains solutions efficiently and has tight theoretical guarantees. As shown in line 1 of Algorithm 2, we first calculate the sum, the maximum, and the average for the E_c, which guides us to make decisions in the following. Then, we traverse the set E_c starting from the maximum value. For the odd iteration, we make the sum of items in S_j exactly less than or equal to AVG, i.e., lines 5-6. For the even iteration, the sum of items is in S_j exactly more than or equals to AVG, i.e., lines 7-8. At last, ABA returns the K sets. From the above pseudocode in Algorithm 2, the time complexity of ABA is O(K). Note that if the number of sets is less than K, we will split some sets to fill up the total K

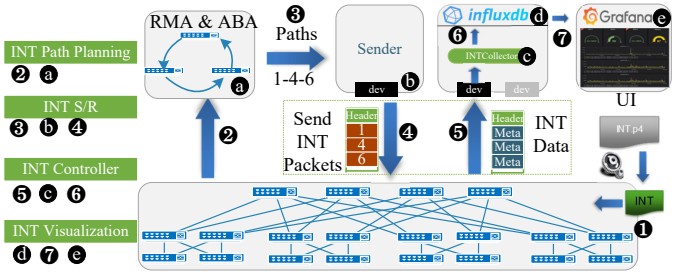


Fig. 3: Workflow of our Proposed INTView

sets. Then, after massive theoretical analysis, we prove that our Algorithm 2 has upper bounds upon the optimum solutions.

Theorem 2. *Our proposed ABA can be bounded as follows:*

- $MAX \geq SUM/2$*: ABA obtains the optimal solutions.*
- $SUM/|E_c| < MAX < SUM/2$*:* $SOL/OPT \leq K/\epsilon$*.*
- $MAX = SUM/N$*: ABA obtains the optimal solutions.*

where OPT *means the optimum solutions for our problem and* SOL *represents the solution of our proposed ABA.*

Proof. For the first case, since the maximum edge's weight among E_c is larger than $SUM/2$, our proposed Algorithm 2 will set the maximum edge into a single set. There is no better solution than this and this is the optimal solution.

For the second case, we have $MAX \leq OPT$ and $SUM/|E_c| \leq MAX$, which derives that

$$(SOL * 2)/|E_c| \leq SUM/|E_c| \leq MAX \leq OPT$$
$$\Rightarrow SOL/OPT \leq |E_c|/2.$$

Since $SUM/K \leq OPT$, we have $SOL/OPT \leq K$. Furthermore, in line 6 of Algorithm 2, if we make $sum(S_j) < SUM/\mathcal{E}$, our conclusion is improved to $SOL/OPT \leq K/\mathcal{E}$.

For the last case, since that maximum equals to the average in set E_c, all the elements in E_c are equal. Thus, after running ABA, all the K sets are equal, i.e., the optimal solutions. □

To conclude from above all, the solutions obtained by our proposed ABA are not too far away from the optimal ones.

IV. Implementation of INTView

We implement a network visualization platform where we incorporate our designed Algorithm 1 and Algorithm 2 into it. This platform is deployed on real P4 devices i.e, Barefoot Wedge100BF, and commercial servers. Based on this, we can monitor the network status graphically in real-time.

A. INTView Overview

Fig. 3 shows the overall design of our proposed INTView. This platform consists of four components, including INT path planning, INT Sender/Receiver (S/R), INT Controller [11], and Graphical display, which are illustrated as follows:

INT Path Planning: This component is responsible for planning INT paths according to the current network status, ensuring non-overlapping, low overhead, and high performance, i.e., step 2 in Fig. 3. Specifically, it runs our designed ABA, according to the different demands of network operators. The

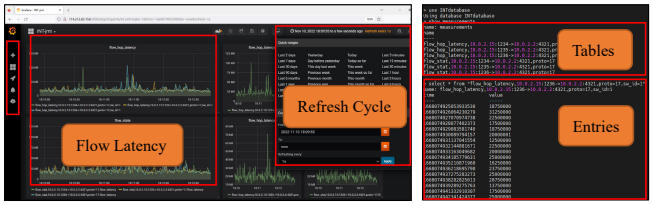


Fig. 4: INTView Dashboard and Database Table Entry

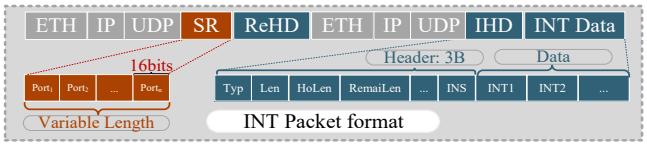


Fig. 5: Our Reconstructed INT Protocol

algorithms would output a series of INT paths in the form of ports which are used by the following INT S/R component.

INT S/R: On the one hand, this component encapsulates the calculated paths to INT packets based on our designed Source Routing protocol (see IV-B for details), i.e., step 3&4 in Fig. 3. On the other hand, it is also responsible for receiving and preliminary handling INT packets for the sake of adapting to INTCollector, i.e., step 5 in Fig. 3.

INT Controller: In general, INT for large-scale of network often leads to a large number of INT packets. In order to relieve the load of the controller, the researchers proposed INT Controller [11] which processes INT reports with a focus on high performance. Specifically, INT Controller processes and filters the received INT packets based on eXpress Data Path (XDP) [17]. We apply INT Controller into our INTView and use the time series database Influxdb, i.e., step 6.

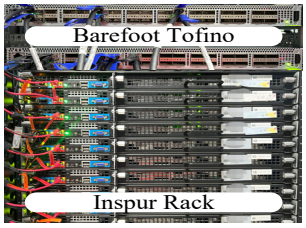
INT Visualization: This component provides INT data visualization for network operators in a real-time manner. Specifically, we use Grafana [18] as an interface for monitoring INT data on the top of the Influxdb database. Grafana obtains data from Influxdb according to users' different cycle configurations, i.e., step 7 in Fig. 3. Fig. 4 shows the Grafana interface with a custom dashboard visualizing INT data.

B. Data Plane Implementation

In this subsection, we introduce some technical details of data plane implementation regarding INT protocol. In order to support INTCollector, we implement INT by reference to INT Dataplane Specification V1 [9] and V2 [10]. Our INT packet format is shown in Fig. 5, and each field is as follows:

SR indicates the INT paths calculated from our algorithms RMA or ABA. Specifically, we use 2 bytes to store information required by each hop, of which 9 bits represent the port number, i.e., the length of the port number in the P4 metadata. Then, we use 7 bits (16 - 9) to represent the sign which is used to mark whether the current hop is last. We allocate a variable-length stack for this field, which means the top item of this field will pop up after each hop and this field will disappear after the last hop. P4 switches support these operations and emit the corresponding packet according to the sign.

ReHD indicates the overall information of INT report, including the length of the report, protocol version (INTv1



(a) Real Devices

Resources	Avg	Tol
TCAM	1.67%	2
SRAM	3.25%	13
VLIW	5.0%	1
Match xbar	1.72%	11
Stash	3.75%	3

(b) P4 Resources Usage

Fig. 6: Results of Our Preliminary Testbed Study

or INTv2), the switch ID, the timestamp of the report, and the serial number of the report. Specifically, the serial number of the report is implemented by using the register function and recorded locally on the switch. The corresponding value will automatically increase when a new report is generated.

IHD represents INT header, including overall information of INT within the current packet. For example, this field contains the number of hops, the amount of data per hop (in word, 1 word=32bit), INT instruction set, etc. Specifically, we have implemented more than 30 instruction sets including INT instruction 0003 and 0407, covering most of the INT services.

INT Data represents the actual INT data, which uses also variable-length stack. The length of this field will adaptively increase after each hop. As shown in Fig. 5, this field includes 8 kinds of INT metadata. For each telemetry, users could choose the required metadata by issuing corresponding instructions to the switches.

When a packet is received, P4 switch first checks whether it is an INT probe or not by extracting **ETH**. If so, **SR** indicates the egress ports, **ReHD** indicates the information of the INT report while **IHD** represents the INT header. After extracting the desired INT information, **IHD** filed update.

C. Telemetry Monitoring Implementation

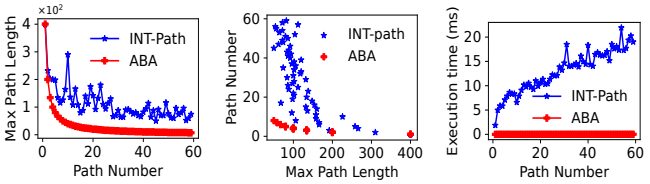
P4 switch brings us fine-grained network telemetry, however, at the cost of high overhead in the controller. INTCollector [11] is proposed to solve this with high performance.

INTCollector is based on the assumption that consecutive packets will contain similar information in general. Massive consecutive and redundant information will be removed. Besides, it also uses a user-defined threshold to filter packets. For example, the hop latency between consecutive packets differs by more than 100ns by default, and it will be stored in a database. Therefore, we change the default threshold according to our needs and visualize the INT data in the left of Fig. 4. We also observe the table and entry of the database and catch the three flows in our platform, as shown in the right of Fig. 4.

V. Experiment

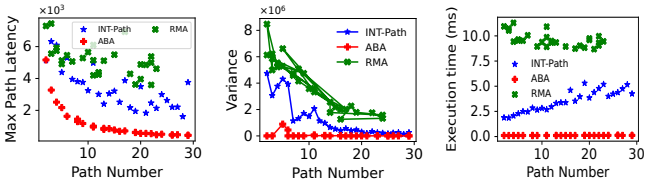
A. Data and Settings

Basic Setting: The experiments are conducted based on the system prototype INTView we mentioned before. According to the settings in works [1, 2], we use Watts-Strogatz model [19] to generate different network topologies with small-world



(a) Path Length (b) Path Number (c) Overhead

Fig. 7: Results of Network-unaware Path Planning



(a) Path Length (b) Variance (c) Overhead

Fig. 8: Results of Network-aware Path Planning

properties randomly, covering massive real network topologies. The network size we test is set ranging from 100 switches to 400 switches. Then, we implement our INT protocol.

Hardware/Software P4 switch: As shown in Fig. 6, We deploy our system prototype INTView upon two hardware-based P4 switches, i.e., Barefoot Wedge100BF-32X (with 3.2Tbps Tofino ASIC), and list the consumption of our INT protocol. For the large scale of the network, we use Mininet [20], and the switches are replaced by BEHAVIORAL MODEL [21], i.e, P4 switch. This emulation testbed is built upon PowerEdge R740 with Intel(R) Xeon(R) Silver 4110 CPU.. We emulate a topology with 400 switches in our experiment.

Algorithms and Metrics: We implement the algorithm of INT-Path for comparison, whose path planning algorithm is based on the undirected graph, and its performance relies heavily on the number of odd points. We make our comparisons under two situations. (1) $Network-aware$: The weight of each edge is set to 1, and INT-Path is based on this situation. (2) $Network-unaware$: The weight of each edge, i.e., the latency of each link, is taken from our INTView. The metrics include INT completion time and maximum path length.

B. Testbed Results

We show our experiment results from two aspects:

Network-unaware Path Planning: INT-Path designs its algorithm based on this scenario, and our algorithms achieve a 60% reduction in maximum path length over INT-Path.

As shown in Fig. 7 (a), we evaluate the maximum path length. The result shows that our proposed ABA has better performance than the baseline with the growth of path number. Fig. 7 (a) also shows us that our ABA often has stable results. The basic reason is that our algorithm makes path planning based on the directed graphs and often obtains the optimum solutions, in the situation of edge weight equivalent to 1.

On the other hand, we also evaluate the path number, as shown in Fig. 7 (b). The result shows that our ABA is able to obtain a small number of INT paths under the same situation as INT-Path. We vary the maximum-paths from 100 to 400, Fig. 7 (b) shows that our ABA has better performance while

the length is small. Since the total number of switches is 400, the maximum path length equal to 400 means that there only exists one path, which is the optimum and unique solution Fig. 7 (c) shows the comparison of overhead over INT-Path. We conclude that our ABA usually has excellent performance since the path planning time is often short.

Network-aware Path Planning: Furthermore, we evaluate the performance of our designed algorithm under the condition where the links of the network have latency. Results show that ABA achieves up to 2x performance improvement.

From Fig. 8 (a)-(c), RMA often has worse performance compared with ABA, due to that RMA is a random strategy and needs multiple iterations to achieve positive performance.

Fig. 8 (a) plots the total maximum path length. We conclude that our designed ABA often has the best performance among other algorithms. Especially, the performance of our ABA is often stable with the growth of path number.

Furthermore, we define a new metric named variance which describes the balance degree of different paths, which affects the total completion time for each INT. Fig. 8 (b) plots the results showing that INT paths of ABA are often balanced. Fig. 8 (c) shows the execution time for the three algorithms. In general, INT-Path consumes more time with the growth of path number, but our ABA consumes a little since the algorithm complexity is O(K) where K is the number of edges.

VI. Related Work

We summarize prior research in two categories, and highlight their drawbacks compared to our work, respectively.

A. Traditional Optimization for INT

In-band Network Telemetry (INT) was widely investigated in [22]. As an underlying device-level primitive, INT was first proposed in [5]. Then, EQuery transformed operators' queries into telemetry rules without exposing the underlying hardware details [23]. As a new framework of network telemetry, NetView [7] saw planning INT paths as Hamilton Cycle Problem [24] and proposed an optimal algorithm. Rumenigue, $et\ al$. [8] proposed a machine learning based INT orchestration model, which focused on passive INT and chose the appropriate set of flows to cover all the switches. However, most of these works failed to consider the influence of paths on INT, and vertex cover based path planning is not suitable for the current mainstream commercial P4 switch.

B. Edge Cover Based Path Planning

Different from these works, some works considered path planning as a $Edge\ Cover$ problem. INT-path [1] guides the INT packets by using source routing protocol and improved the INT performance by generating non-overlapped minimum paths. Furthermore, INT-probe [2] was proposed to reduce the number of paths by sacrificing the non-overlapped property slightly, which ensured the minimal number of overlapped links. P^{2}INT [25] provided an INT plan by generating probe packets to cover all links, which was proved near-optimal.

However, all of them consider network topology as an undirected graph and ignore the changing character of the network status, resulting in their failure to obtain the optimum solution for edge cover based path planning problems.

VII. Conclusion

In this paper, we propose a network visualization system prototype INTView, including an INT path planning algorithm, an INT collection platform, and visualization tools. Quite different from the previous work, we consider path planning as an Edge Cover problem. We overcome the challenges of modeling and design algorithms with performance guarantees. The testbed results show that our algorithm achieves 2x performance improvement, compared with the state-of-the-art schema. In the future, we consider INT data as a data stream [26] and solve this problem in an online manner.

References

[1] T. Pan and *et al*, "Int-path: Towards optimal path planning for in-band network-wide telemetry," in *IEEE INFOCOM*, 2019, pp. 487–495.

[2] T. Pan, X. Lin, and *et al*, "Int-probe: Lightweight in-band network-wide telemetry with stationary probes," in *IEEE ICDCS 2021*, pp. 898–909.

[3] M. Elbaham, K. K. Nguyen, and M. Cheriet, "A traffic visualization framework for monitoring large-scale inter-datacenter network," in *CNSM 2016*. IEEE, 2016, pp. 277–281.

[4] J. D. Case, M. Fedor, M. L. Schoffstall, and J. Davin, "Rfc1157: Simple network management protocol (snmp)," 1990.

[5] C. Kim and *et al*, "In-band network telemetry via programmable dataplanes," in *ACM SIGCOMM*, vol. 15, 2015.

[6] E. Song and *et al*, "Int-label: Lightweight in-band network-wide telemetry via interval-based distributed labelling," in *IEEE INFOCOM 2021*, pp. 1–10.

[7] "Netview: Towards on-demand network-wide telemetry in the data center," *Computer Networks*, vol. 180, p. 107386, 2020.

[8] R. Hohemberger and *et al*, "Orchestrating in-band data plane telemetry with machine learning," *IEEE CL*, vol. 23, no. 12, pp. 2247–2251, 2019.

[9] "P416 Language Specification," https://p4.org/p4-spec/docs/P4-16-v1.0.0-spec.html, 2017.

[10] "In-band Network Telemetry (INT) Dataplane Specification," https://p4.org/p4-spec/docs/INT_v2_1.pdf, 2020.

[11] N. V. Tu and *et al*, "Intcollector: A high-performance collector for in-band network telemetry," in *CNSM 2018*, pp. 10–18.

[12] M. Kwan, "Graphic programming using odd and even points," in *Chinese Math*, 1962.

[13] A. Osterhues and F. Mariak, "On variants of the k-chinese postman problem," 2005.

[14] L. B. H. Victor, "Eulerian path and circuit," 2010.

[15] Y. Nakamura and P. Rudnicki, "Euler circuits and paths," *Formalized Mathematics*, vol. 6, no. 3, pp. 417–425, 1997.

[16] H. Zhu and H. Chen, "Adaptive failure detection via heartbeat under hadoop," in *2011 IEEE Asia-Pacific Services Computing Conference*, 2011, pp. 231–238.

[17] T. Høiland-Jørgensen, J. D. Brouer, and et al, "The express data path: Fast programmable packet processing in the operating system kernel," in *CoNEXT 18*, New York, USA, 2018, p. 54–66.

[18] M. Chakraborty and A. P. Kundan, "Grafana," in *Monitoring Cloud-Native Applications*. Springer, 2021, pp. 187–240.

[19] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[20] "Mininet: Rapid Prototyping for Software Defined Networks," https://github.com/mininet/mininet, 2021.

[21] "Mininet: Rapid Prototyping for Software Defined Networks," https://github.com/p4lang/behavioral-model, 2020.

[22] L. Tan, W. Su, W. Zhang, J. Lv, and Z. Zhang, "In-band network telemetry: A survey," *Computer Networks*, vol. 186, p. 107763, 2021.

[23] Y. Ran and *et al*, "Equery: Enable event-driven declarative queries in programmable network measurement," in *NOMS 2018*, 2018, pp. 1–7.

[24] M. Kristiansen and *et al*, "Hamilton's modified principle applied to nonlinear circuit problems," *IEEE TCOME*, vol. 82, pp. 598–601, 1963.

[25] A. G. Castro and *et al*, "Near-optimal probing planning for in-band network telemetry," *IEEE CL*, vol. 25, no. 5, pp. 1630–1634, 2021.

[26] S. Liu and *et al*, "Online active learning for drifting data streams," *IEEE Transactions on NNLS*, vol. 34, no. 1, pp. 186–200, 2023.