INT-label: Lightweight In-band Network-Wide Telemetry via Interval-based Distributed Labelling

Enge Song*, Tian Pan*†, Chenhao Jia*, Wendi Cao‡, Jiao Zhang*, Tao Huang*†, Yunjie Liu*
*State Key Laboratory of Networking and Switching Technology, BUPT, Beijing 100876, China
†Purple Mountain Laboratories, Nanjing 211111, China ‡Peking University, Beijing 100871, China
*{songenge, pan, 564822241, jiaozhang, htao, liuyj}@bupt.edu.cn ‡caowendi@pku.edu.cn

***Abstract*—The In-band Network Telemetry (INT) enables hop-by-hop device-internal state exposure for reliably maintaining and troubleshooting data center networks. For achieving *network-wide telemetry*, orchestration on top of the INT primitive is further required. One straightforward solution is to flood the INT probe packets into the network topology for maximum measurement coverage, which, however, leads to huge bandwidth overhead. A refined solution is to leverage the SDN controller to collect the topology and carry out centralized probing path planning, which, however, cannot seamlessly adapt to occasional topology changes. To tackle the above problems, in this work, we propose *INT-label*, a lightweight In-band Network-Wide Telemetry architecture via interval-based distributed labelling. INT-label periodically labels device-internal states onto sampled packets, which is cost-effective with minor bandwidth overhead and able to seamlessly adapt to topology changes. Furthermore, to avoid telemetry resolution degradation due to loss of labelled packets, we also design a feedback mechanism to adaptively change the instant label frequency. Evaluation on software P4 switches suggests that INT-label can achieve 99.72% measurement coverage under a label frequency of 20 times per second. With adaptive labelling enabled, the coverage can still reach 92% even if 60% of the packets are lost in the data plane.**

I. Introduction

Fine-grained, network-wide visibility is vital to reliably maintaining and troubleshooting high-density, mega-scale modern data center networks to accommodate heterogeneous mission-critical applications [1, 2]. However, traditional network management protocols, such as SNMP [3], fall short of high-resolution monitoring of highly dynamic data center network traffic, due to the inefficient controller-driven, per-device polling mechanism. With end host-launched full-mesh pings, Pingmesh [4] is capable of providing the maximum end-to-end latency measurement coverage. However, it cannot extract hop-by-hop latency or look into the queue depth inside switches for in-depth analysis, but, for network applications such as load balancing [5], failure localization [6, 7] and management automation [8], these underlying information is increasingly insightful. In-band Network Telemetry (INT) [9], one of the killer applications of P4 [10], allows probe or user packets to query device-internal states, such as queue depth and queuing latency, when they pass through the data plane pipeline, which is considered promising for high-precision monitoring and has been embedded into vendors' latest merchant silicon [11–13]. However, as a chip-level primitive, INT simply defines the contract between the incoming packets and the device-internal states for monitoring. For network-wide telemetry, further orchestration on top of INT is needed.

This work is supported by the National Key Research and Development Program of China (2019YFB1802600), the National Natural Science Foundation of China (61702049, 61872401). Corresponding author: Tian Pan.

There are generally two design patterns to realize network-wide measurement coverage as shown in Fig. 1, that is, sending probe packets [4, 5, 14, 15] and in-situ measurement [16–19]. HULA [5] follows the probe packet paradigm and adopts the ToR switches to flood the probes into data center network's multi-rooted topology for measurement coverage. Since each probe sender does not have the global view of the network to make any coordination, one link may be repetitively monitored by many probes with large bandwidth overhead. For high-resolution monitoring, the bandwidth waste will get even worse. To overcome this limitation, centralized probing relies on the SDN controller [20] to make optimized probing path planning. For example, INT-path [15] collects the network topology and generates non-overlapped probing paths that cover the entire network with the minimum path number using an Euler trail-based algorithm. INT-path is theoretically perfect but still has deployment flaws. First, the path planning result is tightly coupled with the topology. Any topology change in failure-prone data center networks will cause topology recollection and path recalculation at the controller, and probe generators/collectors (*i.e.*, path endpoints) reassignment at the data plane, interrupting the 24/7 telemetry service. Besides, it embeds source routing [21] into the probe packet to specify the route the probe takes. This even bloats the probe packet header especially for longer probing paths and needs additional switch support of source routing forwarding. Finally, the probe packet may have potential different forwarding treatment with user traffic, which affects the measurement accuracy.

As in-situ measurement approaches, [16] and [17] adopt a "probeless" architecture, that is, they insert the INT header and INT metadata stack directly into the user packet without using any probe packet. Specifically, the INT header is inserted into sampled packets of specific flows from network ingress, and each forwarding node writes INT metadata into those packets labelled with the INT header. The INT sample rate is adjusted according to the traffic fluctuation of specific flows. Labelling user packets at the flow level will inevitably cause redundant probing since different flows may share the same path and many paths aggregate at core devices. Besides, they cannot

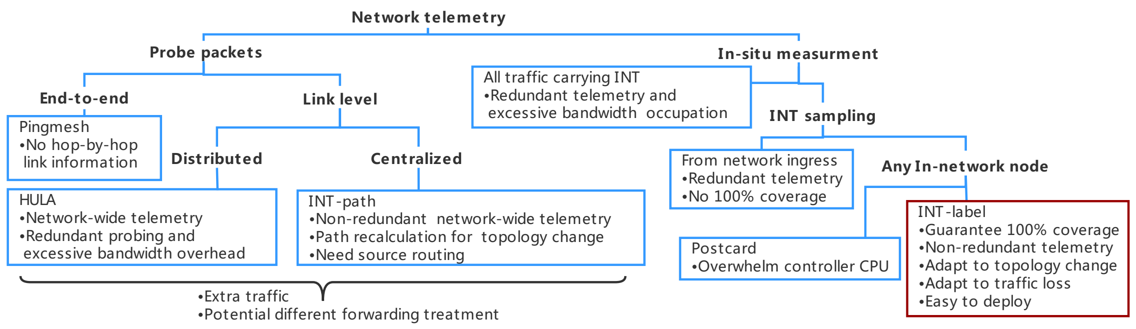

Fig. 1. Landscape of network telemetry approaches.

guarantee network-wide measurement coverage. In postcard-based approaches [18, 19], each device reports its internal states to the controller every time a triggering packet visits the device. However, each device separately reporting the device-internal states in a high frequency tends to overwhelm the southbound channel as well as the controller's CPU.

To tackle the above problems, in this work, we propose *INT-label*, a lightweight In-band Network-Wide Telemetry architecture. INT-label follows the "probeless" architecture. The INT-label-capable device periodically labels device-internal states onto user packets rather than explicitly introducing probe packets. Specifically, on each outgoing port of the device, the packets will be sampled according to a predefined label interval T and labelled with the instant device-internal states. As a result, INT-label can still achieve network-wide coverage with fine-grained telemetry resolution while introducing minor bandwidth overhead. Along the forwarding path consisting of different devices, the same packet will be labelled independently simply according to the local sample decision, that is to say, INT-label is completely *stateless* without involving any probing path-related dependency. Therefore, there is *no* need to leverage the SDN controller for conducting centralized path planning. In other words, INT-label is decoupled from the topology, allowing seamless adaptation to link failures. Like INT, INT-label also relies on the programmability of the data plane, which is being well supported by a growing number of merchant silicon architectures such as Barefoot's Tofino and Broadcom's Trident 3. In INT-label, the INT information is extracted and then sent to the controller at the last-hop device for analysis, so the in-network labelling is *transparent* to the end hosts. To tackle the issue of excessive uploads of INT packets, we design a *probabilistic labelling* algorithm which makes label decision for an incoming packet based on the number of already collected INT metadata in that packet, in order to aggregate INT metadata into fewer packets to reduce the number of uploaded INT packets. Besides, to avoid telemetry resolution degradation due to loss of labelled packets on some congested links, we design a *feedback* mechanism to adaptively change the label frequency when the controller gets aware of the packet loss by analyzing the telemetry result.

The main contributions are summarized as follows:

- We propose INT-label, an INT-based "probeless" telemetry architecture, featuring lightweight and stateless. We elaborate the packet encapsulation format and the corresponding packet processing pipeline (§II).
- We design an interval-based distributed labelling strategy, allowing non-redundant labelling to maximally cover the entire network. We also propose a probabilistic labelling algorithm for excessive INT uploads suppression. Furthermore, a feedback mechanism is added for label frequency adaptation to avoid telemetry resolution degradation due to loss of labelled packets (§III).
- We demonstrate that INT-label can theoretically achieve non-redundant network-wide telemetry. Besides, we establish a mathematical model to analyze the distribution of the number of collected INT metadata in a packet under different scales of the FatTree topology (§IV).
- We implement an INT-label prototype using software P4 switches [22] and the project is available at the git repository [23]. Extensive evaluation suggests that INT-label can achieve 99.72% measurement coverage under a label frequency of 20 times per second. The probabilistic labelling algorithm reduces the number of uploaded INT packets by 35.2%. With adaptive labelling enabled, the measurement coverage can still reach 92% even if 60% of the packets are lost. Besides, INT-label consumes only 3% extra bandwidth compared with HULA [5] (§V).

II. Telemetry System Architecture

Design overview. The basic idea of INT-label is that every network device conducts distributed *in-band packet labelling* at each port to write the real-time device-internal states onto the packet header of traffic passing through the port. The packet label operation is triggered periodically at each port and the label or not decision is made locally without requiring any global synchronization. The distributed labelling guarantees network-wide telemetry coverage except that, without relying on additional network-wide probes, INT-label cannot monitor device ports that have no traffic. However, the link states of the ports without traffic are relatively insignificant for network management and easy to speculate. For example, the latency is very low and there is no queue buildup. Generally, the network-wide telemetry resolution is expected to be as high as possible, that is, the INT information of each port should be updated as frequently as possible at the controller. However, this will lead to tremendous overhead [24]. If the device writes INT metadata to each packet passing through its port, the controller will receive excessive redundant information,

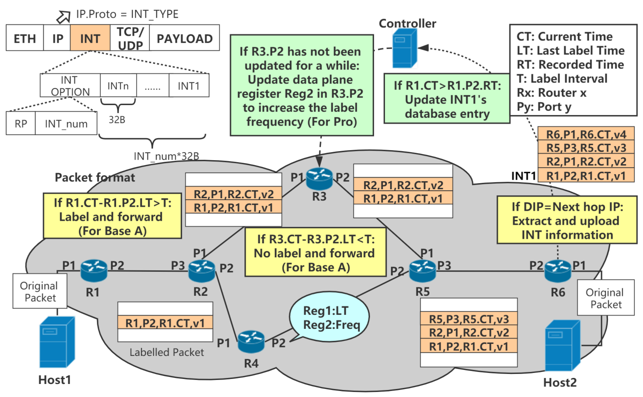

Fig. 2. In-band network-wide labelling (INT-label) architecture.

which will dramatically increase the bandwidth consumption of the switch-controller channel and the processing load of the controller. In real deployment, the appropriate label frequency should be decided according to the given telemetry resolution requirement in the constraint of affordable system cost.

Packet encapsulation. Since the INT metadata is tagged to user traffic, the in-network label operation should (1) not interfere with the original IP packet forwarding, (2) be transparent to the end hosts. As shown in Fig. 2, we place the INT field after the IP protocol header, because the packet label operation is conducted at each port and the IP header needs to be parsed first to get the forwarding port by looking up the routing table. We fill the *PROTOCOL* field in the IP header with *INT_TYPE* when writing INT metadata to the packet for the first time. The first field of the INT header is *INT OPTION*, which contains the original value of the *PROTOCOL* field (*RP*) and the number of INT metadata (*INT_num*). *RP* is used to restore the original packet at the last-hop network device. A variable-length INT label stack is allocated after the *INT OPTION* field. Each INT label occupies 32B, containing the information such as device ID, ingress/egress port ID, latency and queue depth. Both encapsulation and decapsulation of the INT field are conducted inside the network, completely transparent to the end hosts. In the FatTree topology, the packet can reach the destination after at most 5 hops, so the variable part of the INT header is 160B (32B*5) at the longest.

Forwarding behaviors. The INT label operation is done at the outgoing port after route lookup. Each port of the switch/router has a special register (*i.e.*, *Reg1* in Fig. 2) that records the last INT label time. As shown in Fig. 2, when the packet is ready to be forwarded, the network device determines whether the port has written INT metadata to any packet within a *label interval T*. If the port has written INT metadata to a previous packet not long ago, no action will be taken to the current packet. Otherwise, the port will label the current packet and update *Reg1*. If the *PROTOCOL* field is not *INT_TYPE*, we will also need to change it to *INT_TYPE* for INT header encapsulation and write the original *PROTOCOL* field to the *INT OPTION* for later restoration. At the last hop, we will extract the INT header and upload the INT information to the SDN controller. The original IP packet is restored with the *RP* field in the *INT OPTION* and forwarded transparently to the destination. Different from INT-path [15], INT-label is stateless without involving any probing path-related dependency. Therefore, global view of the topology and centralized path planning are unnecessary. In INT-label, each device only needs to make sure that each of its ports is probed within a certain period of time.

Telemetry data collection. The controller has a database in which the records of each port are saved, including the port states such as latency, queue depth and the *timestamp of labelling*. Due to the potential end-to-end latency variation of different paths, even for the same port, the earlier labelled INT information may not reach the controller first. Therefore, the controller needs to determine whether the database needs to be updated according to the newly arrived INT information. If the timestamp of the newly reached record is greater than the one stored in the database, the database should be updated according to the record; otherwise, the newly reached record ought to be discarded. In this way, the latest global network view is always maintained inside the database.

Excessive uploads suppression. The INT information we collect from the data plane needs to be uploaded to the controller for further analysis. However, excessive uploads of INT packets will contend the southbound interface and overload the controller's CPU. To tackle this problem, we further change the original interval-based labelling to the *probabilistic labelling* according to the number of already collected INT metadata in the packet. In our design, the packet which has already been labelled for several times will have a higher probability to be labelled again in its forwarding path. As a result, the INT metadata will be more concentrated. By aggregating once completely randomly distributed INT metadata, the number of packets carrying INT information can be reduced without compromising the telemetry granularity.

Labelled packet loss handling. However, there may be packet loss on the unreliable links. If the labelled packet is lost in transmission, the port will not know this and only believe that it has provided the INT metadata during this period of time. This will result in the failure of the controller to update the database in a timely manner. To resolve this issue, we design *adaptive labelling*. If the controller finds that a port has not been updated for a long time, it suggests that there may be packet loss after the packet is forwarded through this port. We need to increase the frequency of packet labelling at this port, so that the number of packets carrying the INT information will increase, thus weakening the impact of packet loss. As shown in Fig. 2, the controller updates a special data plane register (*i.e.*, *Reg2*) to modify the instant INT label frequency. Increasing the label frequency will not further deteriorate the network situation, because there is no additional increase in the number of packets in the network.

III. Labelling Algorithms

A. Algorithm Description

We first propose a simple interval-based labelling algorithm (Base A) based on the interval between the current time and last label time in the egress port, followed by a more sophisticated one with label times-based probabilistic labelling (Base B). At last, we introduce the adaptive labelling (Pro).

Algorithm 1: Interval-based labelling (Base A)

```
Input: Label interval T, each arriving packet
Output: (Labelled or unlabelled) outgoing packet
1  Function LabelPacket(packet):
2      if IP.PROTOCOL $\neq$ INT_TYPE then
3          Write original IP.PROTOCOL to RP and modify the IP.PROTOCOL to INT_TYPE
4          Set INT_num in the INT OPTION to 0
5      Write INT metadata and increase INT_num by 1
6      Write $R_x.CT$ to the register Reg1 of the port $R_x.P_y$
7  Function BaseA(T, packet):
8      Look up the routing table to get the forwarding port $R_x.P_y$
9      Get last label time $R_x.P_y.LT$ from the register Reg1 and current time $R_x.CT$ of router $R_x$
10     if $R_x.CT - R_x.P_y.LT > T$ then
11         LabelPacket(packet)
12     else
13         NoAction()
14     if DIP = next hop IP and IP.PROTOCOL = INT_TYPE then
15         Extract and upload INT information to the controller
16         Modify the IP.PROTOCOL with the RP
17         Recalculate the checksum
18     Forward packet
```

Interval-based labelling (Base A). Algo. 1 details the packet labelling and label decision making algorithm, which is invoked for each arriving packet. For packet labelling, if the packet has never been labelled, we need to allocate the INT header and change the *IP.PROTOCOL*. For packet restoring purpose, the original *PROTOCOL* field needs to be recorded (line 2-4 in Algo. 1). The INT metadata will then be written to the packet header with *INT_num* incremented by 1. In addition, the last label time (*LT*) in *Reg1* needs to be updated with *CT*, which is the current time read from the timer of the device. (line 5-6). The Base A algorithm leverages the difference of *CT* and *LT* to determine if labelling is required. The label operation will be done only when (1) there is an arriving packet, (2) the label interval *T* has elapsed (line 10-13). It should be noted that line 12-13 is a placeholder function reserved for the Base B algorithm, and no action will be taken in Base A. We compare DIP with the next-hop IP to determine whether the device is at the last hop (line 14). If it is true and the packet has been labelled, the INT information will be extracted and sent to the controller while the original packet is forwarded transparently to the destination (line 15-18).

Label times-based probabilistic labelling (Base B). Algo. 2 describes the probabilistic labelling algorithm. The basic idea of Base B is to label each packet with a probability p, which is determined by the number of already collected INT metadata in the packet (*INT_num*). The larger the *INT_num*, the higher the probability p. The difference between Base B and Base A is that when $R_x.CT - R_x.P_y.LT \leq T$, the device may also label the packet instead of doing nothing (line 13 in Algo. 1). Due to space limitation, Algo. 2 only describes the different part from Base A. Replacing line 13 of Algo. 1 with line 4-7 of Algo. 2 is the complete pseudocode

Algorithm 2: Label times-based probabilistic labelling (Base B)

```
Input: Label interval T, each arriving packet
Output: (Labelled or unlabelled) outgoing packet
1  Function BaseB(T,packet):
2      ......
3      else
4          Read INT_num from the INT OPTION
5          $p = f(INT\_num)$
6          if $random(0,1) < p$ then
7              LabelPacket(packet)
8      ......
```

of Base B. Specifically, if $CT - LT > T$, the packet is still labelled (line 10-11 in Algo. 1). Otherwise, the device first reads *INT_num* from the *INT OPTION* (line 4 in Algo. 2). Then, it calculates the labelling probability p according to *INT_num* (*i.e.*, $p = f(INT\_num)$), and randomly labels the packet with the probability p (line 5-7).

$f()$ can be any increasing function, depending on specific design requirement. The simplest is $f(INT\_num) = \frac{INT\_num}{MAX\_hop-1}$. When $INT\_num = 0$, $p = 0$; when $INT\_num = MAX\_hop - 1$, $p = 100\%$. In this way, the packets carrying fewer INT metadata are still labelled according to the interval, and the packets carrying more INT metadata have a higher probability to be labelled. Compared with Base A, Base B increases the total label times in the data plane, but to a certain extent, makes INT metadata more aggregated. At the same time, as the increase of the data plane label times, the device states collected by the controller is updated slightly faster.

Algorithm 3: Telemetry result collection

```
Input: Each arriving packet with INT information
Output: Updated INT database at the controller
1  Function ResultCollection(packet):
2      Extract INT_info from packet
3      for $R_x$, $P_y$, $R_x.CT$, Latency in INT_info do
4          if the key $(R_x, P_y)$ not in the INT database then
5              Add database entry $(R_x, P_y)$ : $R_x.CT$, Latency into the INT database
6          else
7              Get recorded time $R_x.P_y.RT$ (i.e., previously inserted $R_x.CT$) according to the key.
8              if $R_x.CT > R_x.P_y.RT$ then
9                  Update the INT database with $R_x.CT$ and Latency (i.e., the latest INT metadata)
```

Telemetry result collection. Algo. 3 demonstrates the telemetry result collection algorithm. These steps are shared by both Base A, B and Pro algorithms. At the controller, each packet with the INT information may contain multiple INT metadata collected along the route, each of which will be regarded as a separate entry and installed into the INT database, that is, one telemetry packet uploaded from the data plane may trigger a string of database insertions. The INT database is indexed by the unique key (device ID, port ID) with value filled with the telemetry data as well as the timestamp that the telemetry data is labelled at the data plane (we call this timestamp "recorded time" (RT)). Through adding entries with the latest timestamps, it is ensured that the INT database contains the latest telemetry results (line 8 in Algo. 3).

Algorithm 4: Adaptive labelling (Pro)

```
Input: INT database, label interval T, parameter k
Output: Updated value $S$ in Reg2
1  Initialize: Last time Reg2 is updated (i.e., LRT) = current time
2  Function Pro(INT database, T, k):
3      while True do
4          if current time $-$ LRT $> k * T$ then
5              Get recorded time $R_x.P_y.RT$ according to the key
               $(R_x, P_y)$ from the INT database.
6              if current time $- R_x.P_y.RT > k * T$ then
7                  Reg2 $=$ Reg2 $+ 1$
8                  LRT $=$ current time
9              else
10                 if Reg2 $> 0$ then
11                     Reg2 $=$ Reg2 $- 1$
12                     LRT $=$ current time
```

Controller-driven adaptive labelling (Pro). Adaptive labelling (Algo. 4) is leveraged to handle telemetry resolution degradation due to loss of labelled packets. Based on the Base A/B algorithm, the Pro algorithm adds additional modules to both controller and data plane for dynamically adjusting the label interval T, which is originally fixed in the Base A/B algorithm. In order to dynamically adjust the label interval, we have added a register *Reg2* to store a variable S at each port of the device. The initial value of S is set to 0, which will be updated by the controller. The value of S suggests that the label interval T needs to be reduced by 2^S times. That is, the label interval of Base A/B changes from T to $T/2^S$, which can be easily implemented with the right shift operation. By analyzing the network-wide device states, the controller can determine if the label frequency of some ports need to be changed. For example, if the recorded time of a port has not been updated for a very long time (*i.e.*, $CT-RT > k*T$), the controller will increase S in *Reg2* to raise the label frequency (line 6-7 in Algo. 4). Otherwise, the controller will decrease S except that $S = 0$ (line 10-12). In order to ensure that S will not be changed too often in a short time, we record the last timestamp as *LRT* when *Reg2* is updated and modify S after a period of time from the *LRT* (line 4).

B. Case Study

Fig. 2 shows the label-based INT collection and upload process, as well as adaptive labelling. Firstly, Host1 sends a packet with the destination of Host2. When forwarding the packet, the routers decide whether to label INT metadata to the packet according to the line 10 in Algo. 1. Also, the routers determine if the next hop is the destination address. If so, they will extract the INT information from the packet and restore the original packet (line 14-16). Thus, the transparency of the INT collection process to the host would be guaranteed. When receiving the INT information uploaded by R6, the controller will parse them one by one (as described by the line 2-3 in Algo. 3). Then, the newer records are stored in the database through comparison of the timestamps (line 7-9). If a port has not been updated for a long time, the controller will modify *Reg2* of the corresponding port for label frequency increase.

IV. Theoretical Analysis

A. Proof of Correctness

Firstly, we demonstrate that the Base A algorithm can achieve both network-wide probing and non-redundant labelling. Here, network-wide probing means that all links with traffic will be monitored during a given time window while non-redundant labelling means that none of these links will be monitored more than once during a pre-defined interval.

1) Proposition 1: Base A can probe the device states of all the ports with traffic within the interval $T + T_s$, where T is the label interval, T_s is the maximum packet arrival interval.

Proof. We construct a simple proof by contradiction. Assuming that Base A cannot achieve network-wide probing within $T + T_s$, that is to say, the time interval between the two label operations can be greater than $T + T_s$, indicating that there is at least a port $R_x.P_y$ that does not write INT metadata to the packets during the time period $[t, t + T + T_s]$. That also means $t > R_x.P_y.LT$. On the other hand, there must be a packet forwarded to other network devices through this port within the time period $[t, t + T + T_s]$, since the time interval between packets arriving at the port is less than or equal to T_s. Now consider a packet with arrival time of $t + T + T_p$, where $0 < T_p \leq T_s$. $\because\ t > R_x.P_y.LT\ and\ T_p > 0$. $\therefore t + T + T_p - R_x.P_y.LT > t + T + T_p - t = T + T_p > T$. According to the line 10 in Algo. 1, the port needs to write INT metadata to this packet at its arrival time $t + T + T_p$. Since $0 < T_p \leq T_s$, the port writes INT metadata to the packet during the time period $[t, t + T + T_s]$, which contradicts the assumption. Hence, the assumption is incorrect and Base A can achieve network-wide probing within the interval $T + T_s$.

In other words, we can complete the probing of all the ports with traffic at time $t+T+T_s$ from any given time t. The value of T_s is roughly inversely proportional to the rate of traffic passing through the port. So, when the traffic rate becomes higher, the time required to probe all the ports with traffic will approach T. Conversely, if the traffic rate is low, T_s will inevitably increase, which suggests that the port states cannot be monitored as frequently as expected. However, since the traffic rate is generally extremely high in a data center network, T_s is far less than T thus $T + T_s$ is very close to T.

2) Proposition 2: Base A conducts non-redundant labelling within the interval T, where T is the label interval.

Proof. Similarly, the contradiction method is served to prove the non-redundant probing property of Base A. Assuming that INT-label can cause redundant probing, that is, a port $R_x.P_y$ writes INT metadata to at least two packets within time T. The moments that two packets carrying device-internal states leave the port are denoted as $t1$ and $t2$, where $t2 > t1$. According to our assumption, $t2 - t1 < T$. When the packet is to be forwarded at time $t2$, the label or not decision in Algo. 1 will be made. At this time, $R_x.CT = t2$, $R_x.P_y.LT \geq t1$. $\therefore R_x.CT - R_x.P_y.LT \leq t2 - t1 < T$. According to the line 10 in Algo. 1, the port should not write INT metadata to the packet before forwarding. That is, the packet being forwarded at time $t2$ will not carry INT information, which contradicts the assumption. So the assumption is incorrect and Base A achieves non-redundant probing within the label interval T.

B. Analysis of INT label times distribution

In this subsection, we analyze the distribution of the number of INT metadata carried by the packets of Base A and Base B under different scale FatTree topologies. The analysis is divided into three parts: (1) the relationship between the number of hops required for a packet to reach the destination and the size of the FatTree, (2) the label times distribution of Base A, (3) the label times distribution of Base B.

End-to-end hop number vs. FatTree topology size. We analyze the number of hops required for a packet to reach the destination under a k-ary FatTree as introduced in [25]. The k indicates its pod number, and each pod contains $k/2$ aggregation and edge switches separately. Each edge switch is directly connected with $k/2$ hosts. The total number of hosts is $k^3/4$. We assume that the host1 located at <*pod1, edge1*> sends packets to all other hosts. Since there are $k^3/4$ hosts in total, host1 sends $k^3/4-1$ packets. In the FatTree topology, the end-to-end hop number of a packet is only related to the positional relationship between the sender and the receiver. For ease of description, we assume that the location of the receiver is <*podR, edgeR*>. Let H be the number of hops required for the packet to reach the destination. Next, we analyze the end-to-end hop number case by case.

1. If *edgeR* = *edge1*, $H=1$. Each edge connects $k/2$ hosts, so there are $k/2-1$ packets with an H value of 1.

2. If *edgeR* $\neq$ *edge1* and *podR* = *pod1*, $H=3$. A pod contains $k/2$ edge switches, and each edge switch connects $k/2$ hosts, so there are $(k/2-1)*k/2$ packets with an H value of 3.

3. If *podR* $\neq$ *pod1*, $H=5$. There are k pods in the k-ary FatTree network, and each pod contains $(k/2)^2$ hosts, so there are $(k-1)*(k/2)^2$ packets with an H value of 5.

So H is a discrete random variable whose value can be 1, 3, and 5. The probability distribution of H can be calculated as: $P(H=1)=\left(\frac{k}{2}-1\right)/\left(\frac{k^3}{4}-1\right)$, $P(H=3)=\left(\frac{k}{2}-1\right)*\frac{k}{2}/\left(\frac{k^3}{4}-1\right)$, $P(H=5)=(k-1)*\left(\frac{k}{2}\right)^2/\left(\frac{k^3}{4}-1\right)$.

Label times distribution of Base A. In Base A solution, each switch independently determines if a packet needs to be labelled, whether the packet has already carried INT metadata or not. Therefore, the packet labelling by different switches are independent and unrelated events in probability theory. We assume that X_i represents whether the packet is labelled after passing through the i-th switch. X_i is a discrete random variable whose value is 0 or 1 (representing unlabelled or labelled), so X_i is a typical *Bernoulli trial*. We assumed that the average interval of packets passing through each port is T_s and the label interval is T. There will be a packet being labelled among T/T_s packets, so the probability of a packet being labelled is $p=1/(\frac{T}{T_s})=\frac{T_s}{T}$. Therefore, $X_1,X_2,...,X_n$ all obey the *Bernoulli trial* with p being $\frac{T_s}{T}$.

There is a theorem in probability theory as follows: In a Bernoulli experiment with n trials and the probability of success on each trial is p. Let A count the number of successes in n trials, then the probability of event $A=s$ is: $P(A=s)=C_n^s p^s(1-p)^{n-s}, s=0,1,2,\dots,n$.

In Base A solution, let Y_n count the number of label times after n hops. $\because\ Y_n \sim B(n,\frac{T_s}{T})$. $\therefore\ P(Y_n=s)=C_n^s\left(\frac{T_s}{T}\right)^s\left(1-\frac{T_s}{T}\right)^{n-s}, s=0,1,2,\dots,n$.

Let Z_A represent the label times of a packet sent by host1 in Base A solution, so the probability distribution of Z_A is: $P(Z_A=s)=\sum_{l=1,3,5}P(H=l)*P(Y_l=s), s=0,1,\dots,5$. By plugging the above relations into the expression of $P(Z_A=s)$, we get the probability distribution of label times in the Base A solution. Limited by space, we will not expand it here but put the results on our git repository [23].

Label times distribution of Base B. In Base B solution, each switch labels packets further according to the number of INT metadata carried in the packets. The larger the number of metadata, the greater the probability of being labelled. It is obviously different from Base A that whether the packet of Base B will be labelled on the subsequent switch will be affected by the labelling of previous switches. Therefore, each labelling cannot be regarded as an independent unrelated event, we need to analyze the correlation between the two labelling.

In Base B solution, even if $CT-LT\leq T$, the switch may label the packet and refresh the last label time (*LT*), which reduces the probability of labelling based on the interval. We denote the probability based on the interval and the probability based on the INT metadata number as p_B and $f(\textit{INT\_num})$, respectively. The larger $f(\textit{INT\_num})$ is, the smaller p_B is. For example, when $f(\textit{INT\_num})=1$, then $p_B=0$. The switch will only label packets based on the INT number. Conversely, the smaller $f(\textit{INT\_num})$ is, the larger p_B is. When $f(\textit{INT\_num})=0$, $p_B=\frac{T_s}{T}$. It can be seen that p_B is a function of $f(\textit{INT\_num})$, denoted as $p_B=g(f(\textit{INT\_num}))$. According to the above analysis, $g(x)$ is a monotonically decreasing function as $g(x)\in\left[0,\frac{T_s}{T}\right], x\in[0,1]$.

For the convenience of explanation, we denote N and T as INT number-based and interval-based labelling, respectively. $P(N)=f(\textit{INT\_num})$, $P(T)=g(f(\textit{INT\_num}))$. Next, we analyze the probability of joint events case by case.

1. The packet is labelled based on the INT number, denoted as $N\bar{T}$. According to the *total probability rule*, $P(N)=P(T)P(N\mid T)+P(\bar{T})P(N\mid\bar{T})$. Because N and T cannot happen at the same time, $P(N\mid T)=0$. Therefore $P(N)=P(\bar{T})P(N\mid\bar{T})$. According to the *formula for conditional probability* $P(B\mid A)=P(AB)/P(A)$, $P(N\mid\bar{T})=P(N\bar{T})/P(\bar{T})$. Therefore, $P(N\bar{T})=P(\bar{T})P(N\mid\bar{T})=P(N)=f(\textit{INT\_num})$.

2. The packet is labelled based on the interval, denoted as $\bar{N}T$. As with 1, we can get the conclusion: $P(\bar{N}T)=P(T)=g(f(\textit{INT\_num}))$.

3. The packet is not labelled, denoted as $\bar{N}\bar{T}$. Because N and T cannot happen at the same time, $P(NT)=0$. Therefore, $P(\bar{N}\bar{T})=1-P(NT)-P(\bar{N}T)-P(N\bar{T})=1-f(\textit{INT\_num})-g(f(\textit{INT\_num}))$.

In Base B solution, let W_n count the number of label times after n hops. Obviously, W_n is a discrete random variable with values of $0, 1, 2, \ldots, n$. Easy to get, $P(W_{n+1} = i) = P(W_n = i) * (1 - g(f(i)) - f(i)) + P(W_n = i - 1) * (f(i-1) + g(f(i-1)))$, $i = 0, 1, \ldots, n+1$. The first term represents that when $W_n = i$, the packet is still not labelled by the $n+1$th hop switch, so $W_{n+1} = i$. The second term represents that when $W_n = i - 1$, the packet is labelled by the $n+1$th hop switch, so $W_{n+1} = i$. The boundary terms involved in this formula are all 0, such as $P(W_n = -1) = 0, P(W_n = n+1) = 0$. Then we analyze the probability distribution of W_1. Since the packet before the first-hop switch labelling will not carry any INT information, whether the packet is labelled at the first-hop switch is affected only by the interval, so the labelling probability is the same as that of Base A, which is $\frac{T_s}{T}$. Therefore, $P(W_1 = 0) = 1 - \frac{T_s}{T}$ and $P(W_1 = 1) = \frac{T_s}{T}$. According to the expression of $P(W_{n+1})$, we derive the probability distribution of W_2, and then the probability distribution of $W_3, W_4, \ldots$.

Similar to the analysis in Base A, we only need the distribution of W_1, W_3, and W_5. Z_B represents the label times of a packet sent by host1 in Base B solution, so the probability distribution of Z_B is: $P(Z_B = s) = \sum_{l=1,3,5} P(H = l) * P(W_l = s), s = 0, 1, \ldots, 5$. By plugging the above relations into the expression of $P(Z_B = s)$, we get the probability distribution of label times of Base B. Also, we put the results on our git repository [23].

V. Evaluation

A. Experiment Setup

We build an emulation-based network prototype to demonstrate INT-label's performance. The configuration is Intel Xeon Silver 4116 CPU with 12 cores and 24 threads, and 32GB memory with Ubuntu 18.04 OS. The prototype is based on Mininet [26] and consists of 1 controller, 4 Spine switches, 4 Leaf switches, 4 ToR switches and 8 servers, organized into the same FatTree topology with HULA [5]. Among the 8 servers, server1 and server8 randomly send traffic (packet size = 1kB) to others. The default traffic rate is around 5Mbps. With pre-installed flow tables, traffic passes through 24 ports (some ports are deliberately bypassed without traffic). The default label interval is 50ms. The maximum INT header length is 160B (32B*5). The default network-wide telemetry resolution is 10 times/s (here, we set a fairly coarse-grained default telemetry resolution limited mainly by BMv2's poor performance [22]). The default $f()$ is $f(0) = 0, f(i) = 1, i = 1, \ldots, 4$. The prototype consists of 829 lines of Python and 317 lines of P4, available at the git repository [23].

Network testbed. Mininet [26] is used to establish the virtual network environment. We use the simple switch model of BMv2 (a software-based P4 switch) [22] to realize the virtual switch so that arbitrary packet header customization of INT can be allowed. The simple switch model of BMv2 is good enough for our design verification despite the implementation difference it may have with the actual hardware P4 switch. Besides, we implement a remote controller through a separate OS process that receives notifications from the data plane and communicates with the database through socket.

Traffic generation. Python's Scapy library is employed to write scripts for traffic generation and collection. The route of the background traffic is specified by the source routing [21], and the background traffic rate is dominated by the sleep function at the sender (by pausing the sending for a while).

B. Results

Impact of the background traffic rate. Fig. 3 shows the impact of the background traffic rate on the network-wide coverage rate and the bandwidth occupation of INT encapsulation overhead. The *coverage rate* is defined as the labelled port number divided by the total number of ports with traffic passing by during every 100ms (the default telemetry resolution). $bandwidth\ occupation = \frac{\textit{bandwidth of INT}}{\textit{bandwidth of traffic}}$. In §IV, we have demonstrated that our architecture can achieve network-wide probing if the background traffic rate is much larger than the label frequency. In other words, the coverage rate can be affected by the packet arriving rate of each port. As shown in Fig. 3, the coverage rate increases with the background traffic rate, which confirms our analysis. Furthermore, as the background traffic rate increases, the bandwidth occupation rate of the INT encapsulation with a constant label frequency decreases. The reason lies in that, as the traffic rate of the port increases, the average INT information carried by each packet gradually decreases. When the background traffic rate reaches a certain scale, the coverage rate no longer increases significantly, which is a turning point. Besides, the coverage rate and the bandwidth occupation of Base B are always higher than those of Base A due to the possibility of conducting redundant labelling. We set the default background traffic rate as the rate at the turning point in Fig. 3.

The number of packets carrying INT metadata: Base A vs. Base B. Fig. 4 shows the number of packets with INT information under different label intervals, with a total of 10000 packets generated. It can be seen from the figure that as the label interval increases, the number of labelled packets of Base A decreases and the number of labelled packets of Base B is always lower than that of Base A. Specifically, the number of labelled packets of Base B decreases much faster than that of Base A. This is because the increase in T leads to more packets performing probabilistic labelling based on the INT metadata number, resulting in a more concentrated distribution of INT metadata, which in turn leads to a more obvious reduction in the number of labelled packets. When T = 100ms, compared with Base A, the number of labelled packets of Base B is relatively reduced by 35.2%.

The number of label times: Base A vs. Base B. Fig. 5 shows the total label times in the data plane under different label intervals, with a total of 10000 packets generated. As can be seen from the figure, Base B only causes a small number of redundant labels, which is insignificant compared with the reduction in the number of labelled/uploaded packets.

Right label interval for a given telemetry resolution. For network management, a network operator may need to determine which label frequency is appropriate for cost-effective network-wide telemetry under a certain telemetry resolution.

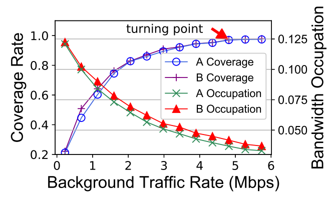

Fig. 3. The impact of label interval on coverage rate and bandwidth occupation

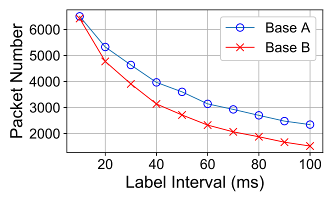

Fig. 4. The number of packets labelled with INT metadata under different label intervals.

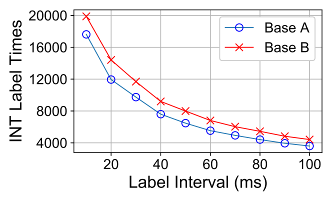

Fig. 5. The data plane label times under different label intervals.

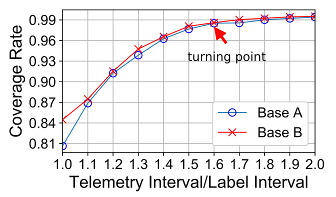

Fig. 6. How the relation between label interval and telemetry resolution affects the coverage rate.

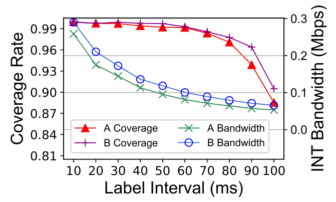

Fig. 7. The impact of label interval on coverage rate and INT bandwidth occupation.

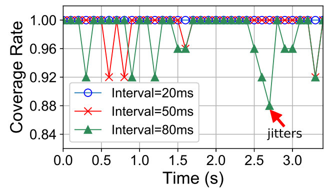

Fig. 8. The impact of data plane label interval on network-wide coverage rate changes over time.

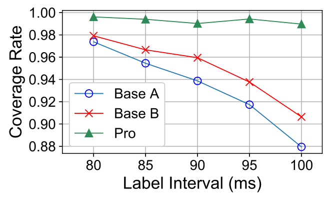

Fig. 9. Different coverage rates under Base A/B and Pro strategies.

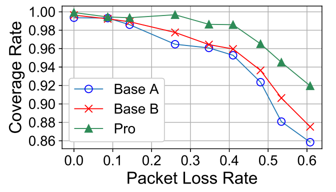

Fig. 10. Network-wide coverage degradation due to loss of packets under Base A/B and Pro strategies.

Over-low label frequency can cause incomplete network-wide view, while over-high label frequency is unnecessary and inefficient. Fig. 6 shows how to choose the appropriate label frequency according to the telemetry resolution requirement. For example, regardless of using Base A or Base B, the network-wide coverage rate in Fig. 6 has exceeded 0.98 when the ratio of telemetry interval/label interval is 1.6, which helps determine the right label interval if the required telemetry resolution (*i.e.*, 1/telemetry interval) is provided. Given the default telemetry resolution of 10 times/s, our system can achieve 99.72% measurement coverage under a label frequency of 20 times/s (telemetry interval/label interval = 2 at this time).

Impact of label interval on coverage rate and bandwidth overhead. Fig. 7 demonstrates the impact of label interval on coverage rate and bandwidth overhead. We set the telemetry interval to 100ms and make the background traffic rate fast enough. As shown in Fig. 7, the coverage rate increases as the label interval decreases. While a smaller label interval corresponds to a higher coverage rate, the bandwidth occupation also becomes greater. Label interval of 60ms is the turning point of the coverage curve. And the same conclusion can be drawn from 100ms/60ms = 1.6 that telemetry interval/label interval = 1.6 can achieve a good tradeoff between coverage rate and bandwidth overhead. In addition, compared to Base A, Base B always has a higher coverage rate and bandwidth occupation due to its redundant labelling property.

Impact of the data plane label interval on coverage rate changes over time. Fig. 8 shows the impact of label interval on network-wide coverage rate changes over time for Base A algorithm. As shown in Fig. 8, when T = 20ms, the coverage always remains at 100%, which means that a smaller label interval has a more stable and higher coverage. We find that a larger label interval causes more jitters of the network-wide coverage rate, which is triggered by the labelled packets queuing in congested links. Although the label intervals in Fig. 8 are all below the default telemetry interval (*i.e.*, 100ms), the labelled packets with larger label interval are more likely to miss the packet collection of the latest telemetry round.

Performance of Base A/B and Pro under different label intervals. Fig. 9 shows the relationship between the coverage of the Base A/B and Pro algorithms and the label interval when label interval and telemetry interval are closer. As shown in Fig. 9, the coverage of the Pro algorithm remains around 99%, while the coverage of Base A/B algorithm gradually decreases with the increase of label interval, because the Pro algorithm can automatically increase the label frequency when coverage decreases. This indicates that the Pro algorithm can always achieve higher coverage, while the coverage of the Base A/B algorithm is affected by the relationship between the label interval and the telemetry interval.

Network coverage rate degradation due to traffic loss. Fig. 10 shows coverage rate degradation due to traffic loss. We deliberately change the background traffic rate, limit the forwarding rate and max queue depth of BMv2 to generate packet loss. Fig. 10 suggests that network coverage rate will degrade due to packet loss since the labelled packets will get lost as well. Specifically, the Pro algorithm outperforms the Base A/B algorithm thanks to the built-in adaptive labelling mechanism. With adaptive labelling enabled, the coverage rate can still reach 92% even if 60% of the packets are lost.

Packet loss due to rate limit. Fig. 11 compares the packet loss rate of Base A/B and HULA due to deliberate rate limit. We limit the bandwidth and max queue depth of BMv2 so that there is no packet loss while sending background traffic alone. It can be seen that there is no packet loss of Base A/B because of its ultra-low bandwidth occupation. HULA induces packet loss, which decreases from 12% to 5% as the probe interval increases from 10ms to 100ms.

Distribution of INT label times. Fig. 12 shows the number of packets under different label times, with a total of 10000 packets. It can be seen from the figure that the number of packets with label times of 1 in Base A is the largest. And the number of packets gradually decreases with the increase of label times. The number of packets whose label times is 5 is extremely small. Although the number of packets with label times of 1 in Base B is also the largest, the number of

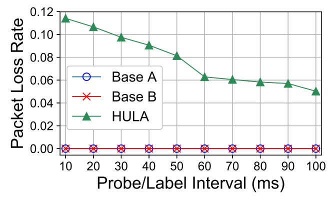

Fig. 11. Packet loss rate (due to rate limit) under different label/probe intervals (Base A/B vs. HULA).

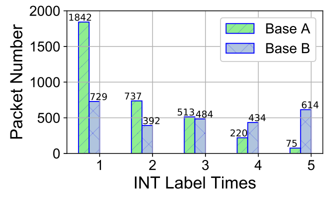

Fig. 12. Distribution of INT label times in Base A/B.

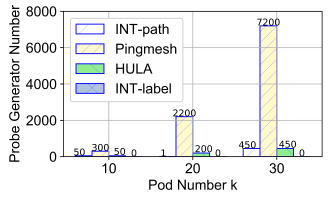

Fig. 13. The number of probe generators required under different scale FatTree topologies.

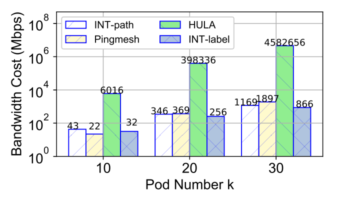

Fig. 14. The bandwidth overhead for network-wide telemetry under different scale FatTree topologies.

packets with label times of 4 and 5 is quite large as well. Even the number of packets whose label times is 5 is second only to those whose label times is 1. In addition, the packets with label times 1 and 2 in Base B are much less than those in Base A, and the packets with label times 4 and 5 are much more. In summary, the Base B algorithm does make INT information more concentrated in some packets, which coincides with the conclusion we've reached in §IV.

The number of probe generators required. Fig. 13 shows the number of probe generators required by different methods under different scale FatTree topologies. INT-label does not need to allocate any probe generator due to its "probeless" mechanism, so its number is 0 in any topology. The number of probe generators of INT-path is positively correlated with the number of odd vertices in the topology [15]. And the odd vertices in the FatTree depends on its scale. When there is no odd vertex, only one probe generator is needed to send and receive probes. When there are $2k$ odd vertices, $2k$ probe generators are needed. The number of Pingmesh is positively correlated with the sum of the number of hosts and ToR switches. The number of HULA is positively correlated with the number of ToR switches. So the number of Pingmesh is the most, and the number of INT-path and HULA are lower. Therefore, INT-label has enormous advantages in deployment as it does not require even a single probe generator.

Network-wide telemetry bandwidth overhead. Fig. 14 shows the bandwidth cost comparison under different scale FatTree topologies. Because INT-label only needs to adds a piece of INT header to packets, it occupies little bandwidth. HULA has a large number of redundant probes due to its broadcast mechanism, so its bandwidth cost is always much higher than that of other methods. Pingmesh's performance is better when the topology size is smaller, but gradually deteriorates as the topology scale increases. It should be noted that Pingmesh is incapable of measuring hop-by-hop link latency. Although INT-path realizes non-redundant telemetry, it needs to generate extra probe packets, resulting in more bandwidth cost than INT-label. It can be concluded that INT-label can achieve network-wide device-internal state coverage with minimal bandwidth overhead.

VI. Related Work

Network monitoring/telemetry provides better visibility into the intricate networks, making it easier to maintain and troubleshoot the widespread distributed systems [27, 28]. However, traditional management protocols, such as SNMP [3], fail to achieve fine-grained monitoring due to its inefficient controller-driven device state polling, which cannot promptly adapt to the drastic traffic dynamics in data center networks. Other approaches such as NetFlow [29], sFlow [30], IPFIX [31] maintain per-flow counters instead of monitoring the low-level packet queuing behaviors. The protocol-independent forwarding architecture [10, 32] unleashes the power of data plane programmability, enabling network devices to arbitrarily edit packet headers. The In-band Network Telemetry (INT) [9, 33] takes full advantage of this capability, allowing probe/user packets to query device-internal states as they pass through the data plane pipeline.

In some use cases, INT is activated to monitor specific flows by labelling each packet or some of the packets of these user-specified flows [34, 35]. While in other use cases, network-wide telemetry is favored for network-wide traffic load balancing [5] or failure detection/localization [4, 6]. As INT is essentially an underlying primitive for device-internal state exposure, network-wide telemetry further requires high-level orchestration on top of INT [15].

Telemetry systems [4, 5, 15] get network telemetry results by sending probe packets, but extra probe packets also lead to extra traffic and potentially different forwarding treatment.

For "probeless" architectures, if all traffic carries INT telemetry data, it will lead to great telemetry redundancy. To reduce redundancy, the idea of sampling gets popular. However, sampling directly from the network ingress [16, 17] will cause considerable telemetry redundancy as well. Besides, it is also difficult to realize the network-wide telemetry within a given interval. Postcard [18] can realize the non-redundant network-wide telemetry, but a large number of INT packets will be uploaded and overwhelm the controller's CPU.

VII. Acknowledgement

We would like to thank Dr. Haoyu Song from Futurewei Technologies for his valuable input on the landscape of network telemetry approaches as well as the label times-based probabilistic labelling. We would also like to thank the anonymous reviewers for their constructive comments.

VIII. Conclusion

In this work, we propose *INT-label*, a lightweight In-band Network-Wide Telemetry architecture without explicitly using probe packets. INT-label periodically labels device-internal states onto sampled user packets, which is cost-effective with minor bandwidth overhead. In INT-label, the network-wide monitoring is completely decoupled from the topology, allowing seamless adaptation to topology changes. INT-label is ready to be deployed in mega-scale modern data centers.

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," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 3–14.

[2] Y. Zhu, N. Kang, J. Cao, A. Greenberg, G. Lu, R. Mahajan, D. Maltz, L. Yuan, M. Zhang, B. Y. Zhao *et al.*, "Packet-level telemetry in large datacenter networks," in *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4. ACM, 2015, pp. 479–491.

[3] C. Hare, "Simple network management protocol (snmp)." 2011.

[4] C. Guo, L. Yuan, D. Xiang, Y. Dang, R. Huang, D. Maltz, Z. Liu, V. Wang, B. Pang, H. Chen *et al.*, "Pingmesh: A large-scale system for data center network latency measurement and analysis," in *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4. ACM, 2015, pp. 139–152.

[5] N. Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford, "Hula: Scalable load balancing using programmable data planes," in *Proceedings of the Symposium on SDN Research*. ACM, 2016, p. 10.

[6] C. Jia, T. Pan, Z. Bian, X. Lin, E. Song, C. Xu, T. Huang, and Y. Liu, "Rapid detection and localization of gray failures in data centers via in-band network telemetry," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–9.

[7] Y. Zhou, J. Bi, T. Yang, K. Gao, C. Zhang, J. Cao, and Y. Wang, "Keysight: Troubleshooting programmable switches via scalable high-coverage behavior tracking," in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*. IEEE, 2018, pp. 291–301.

[8] A. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcón, M. Solé, V. Muntés-Mulero, D. Meyer, S. Barkai, M. J. Hibbett *et al.*, "Knowledge-defined networking," *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 3, pp. 2–10, 2017.

[9] C. Kim, A. Sivaraman, N. Katta, A. Bas, A. Dixit, and L. J. Wobker, "In-band network telemetry via programmable dataplanes," in *ACM SIGCOMM*, 2015.

[10] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[11] "Barefoot deep insight," https://barefootnetworks.com/products/brief-deep-insight/, 2017.

[12] "1000-fold increase in detection precision of packet loss," https://www.huawei.com/ke/press-events/news/2019/6/first-ifit-pilot-5g-transport-network-beijing-unicom-huawei, 2019.

[13] "Broadcom trident 3," https://www.broadcom.com/products/ethernet-connectivity/switching/strataxgs/bcm56870-series, 2019.

[14] N. Katta, A. Ghag, M. Hira, I. Keslassy, A. Bergman, C. Kim, and J. Rexford, "Clove: Congestion-aware load balancing at the virtual edge," in *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*. ACM, 2017, pp. 323–335.

[15] 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 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 487–495.

[16] Y. Kim, D. Suh, and S. Pack, "Selective in-band network telemetry for overhead reduction," in *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*. IEEE, 2018, pp. 1–3.

[17] S. Tang, D. Li, B. Niu, J. Peng, and Z. Zhu, "Sel-int: A runtime-programmable selective in-band network telemetry system," *IEEE Transactions on Network and Service Management*, 2019.

[18] N. Handigol, B. Heller, V. Jeyakumar, D. Maziéres, and N. McKeown, "Where is the debugger for my software-defined network?" in *Proceedings of the first workshop on Hot topics in software defined networks*, 2012, pp. 55–60.

[19] H. Song, T. Zhou, and Z. Li, "Export user flow telemetry data by postcard packets," IETF Secretariat, Internet-Draft draft-song-ippm-postcard-based-telemetry-00, October 2018. [Online]. Available: http://www.ietf.org/internet-drafts/draft-song-ippm-postcard-based-telemetry-00.txt

[20] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[21] C. A. Sunshine, "Source routing in computer networks," *ACM SIGCOMM Computer Communication Review*, vol. 7, no. 1, pp. 29–33, 1977.

[22] "P4 switch behavioral model," https://github.com/p4lang/behavioral-model, 2018.

[23] "Int-label repository," https://github.com/graytower/INT_LABEL, 2020.

[24] E. Song, T. Pan, C. Jia, W. Cao, T. Huang, and Y. Liu, "Int-filter: Mitigating data collection overhead for high-resolution in-band network telemetry," in *2020 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2020, p. To appear.

[25] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM computer communication review*, vol. 38, no. 4, pp. 63–74, 2008.

[26] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010, p. 19.

[27] A. Wundsam, D. Levin, S. Seetharaman, and A. Feldmann, "Ofrewind: enabling record and replay troubleshooting for networks," in *USENIX Annual Technical Conference*. USENIX Association, 2011, pp. 327–340.

[28] D. Yu, Y. Zhu, B. Arzani, R. Fonseca, T. Zhang, K. Deng, and L. Yuan, "dshark: A general, easy to program and scalable framework for analyzing in-network packet traces." in *NSDI*, 2019, pp. 207–220.

[29] B. Claise, G. Sadasivan, V. Valluri, and M. Djernaes, "Cisco systems netflow services export version 9," 2004.

[30] P. Phaal, S. Panchen, and N. McKee, "Inmon corporations sflow: A method for monitoring traffic in switched and routed networks," 2001.

[31] J. Quittek, T. Zseby, B. Claise, and S. Zander, "Requirements for ip flow information export (ipfix)," RFC 3917 (informational), Tech. Rep., 2004.

[32] H. Song, "Protocol-oblivious forwarding: Unleash the power of sdn through a future-proof forwarding plane," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 127–132.

[33] "In-band network telemetry (int) - p4.org," https://p4.org/assets/INT-current-spec.pdf, 2018.

[34] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh *et al.*, "Hpcc: High precision congestion control," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019, pp. 44–58.

[35] R. Ben Basat, S. Ramanathan, Y. Li, G. Antichi, M. Yu, and M. Mitzenmacher, "Pint: probabilistic in-band network telemetry," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 662–680.