A Flexible P4-Based Pin-Point In-Band Network Monitoring

Toshihiro Sato
Graduate School of Computer and Information Sciences
Hosei University
Tokyo, Japan
toshihiro.sato@dsl.k.hosei.ac.jp

Toshio Hirotsu
Fuculty of Computer and Information Sciences
Hosei University
Tokyo, Japan
hirotsu@hosei.ac.jp

***Abstract*—Telemetry is a new generation network monitoring technology, which could reduce the loads of probes by setting the frequency of probes and target of monitoring to switches. In-band Network Telemetry (INT) is a technique of telemetry which enables precise monitoring without reducing the traffic rate. It embeds the telemetry headers including information and instructions of INT processes, then INT enabled switches respond to the instructions in hardware-level. However, if all INT compatible switches simply respond the instruction, it will cause explosive growth of amount of collecting information and makes the load of analysis much heavier. In this paper, we discuss about design and implementation of selective telemetry mechanism which is fully controllable from edge-side. Using the proposed mechanism, administrators can select the switches which is required to be analyzed, then reduce the cost of monitoring. In our proposed mechanism, types of information and list of target switches are marked in the INT header, and only the target INT-enabled switches process the INT operation according to the instruction in the INT header of each packet. Our proposed approach will increase the controllability of telemetry and enhance the flexibility of management of large-scale network.**

***Index Terms*—Software Defined Networking, In-band Network Telemetry**

I. Introduction

In-band Network Telemetry (INT) is attracting attention in the context of service operation on cloud infrastructure and management of wide-area communication infrastructure. INT-enabled switches insert INT header into forwarding packets, to notate commands and collect data for network analysis. This mechanism enables precise network measurement without the performance degradation of network switches.

In the hyper-scale network environment, it is difficult to provide sufficient monitoring performance using SNMP which realizes monitoring by sending request to every switch on each moment. Generated SNMP requests will be explosively increased in the hyper-scale network, and switches consume the processing time to handle these requests. Finally, these overloads will affect the performance of real network traffic. INT is a promising technology that enables lightweight monitoring of hyper scale network. However, if all switches in the hyper-scale network respond to the INT instruction embedded into the packet, increase of the INT information will exhaust the INT header space, and will cause huge of gathered information to analysis.

Another problem will cause on monitoring the service level on the cloud platform. In the recent service architecture, many various microservices are working and cooperate each other to provide a single task. On monitoring these services, supports for the distributed tracing is required in the cloud platform. INT will be useful technology to realize such tracing services in the platform level, and the control interface to specify the monitoring targets, such as virtual NICs of the microservices, will need be opened for cloud users.

In this paper, we propose a pin-point selective network telemetry technique, that enables to specify the monitoring network entity flexibly from the edge-side. We also implement a prototype system using BMv2 P4 model switch and show the feasibility of our proposed approach.

II. Related Works

A. *Sel-INT*

Sel-INT [1] reduces collecting data by controlling sampling rate of each switches with controller-based SDN. In Sel-INT, extended Openvswitch controls packets with its bit string to enable execution of INT. The extended switch supports three extended matching rules for judging existence of INT header, type of adding telemetry data, and source of packet is the controller of not. The controller decides the sampling rate of each switch according to the topology of switches, then sets the rates to each switch. When a switch receives the message from controller, it extracts the sampling rate and marks it to its own flow table, then processes INT process for following packets at the sampling rate. Sel-INT enables telemetry only for the specific switches, but stable SDN control core is required, and hard to choose monitoring network entity arbitrary from edge-side.

B. *PINT*

PINT [2] reduces the amount of collecting data by selecting INT operations probabilistically, and provide similar accuracy with normal (non-PINT) INT. Each switch choose processing INT operations from hash of the ID embedded in the packet and switch ID. The telemetry information is separated into several blocks, and choose a block to embed into a packet INT header. PINT achieves both of reduction of the telemetry data size and accuracy of the telemetry by choosing INT operation and telemetry blocks using the hash function following a uniform distribution. However, it is difficult to choose the telemetry targets pin-point using hash function.

III. Design and Implementation of Pin-point INT

Proposed system is composed of edge-side controller, switches, and data analyzer (Figure 1). Edge-side controller converts user monitoring request to the detailed INT instructions, then control edge-switch to embed it into packets. Each

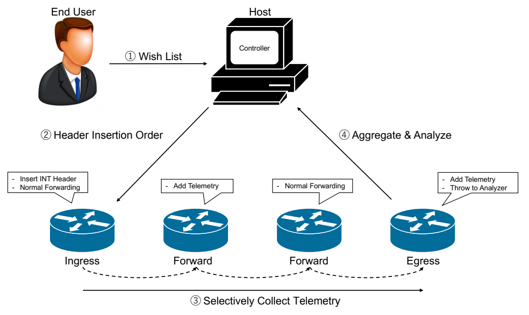

Fig. 1: Structure of Prototype System

switch reads INT header of each packet and perform the INT processing based of the INT instruction. INT instructions contain the information whether each switch performs INT operation or not. It also includes the type of INT request. When a switch is the target of INT action, it checks the type of requested data and insert it into INT header. It also sends the gathered telemetry information to the edge-side controller when all INT data for the packet has been collected. Data analyzer extracts the telemetry data from the message and accumulate them.

A. INT Header

Our proposed scheme encodes the instruction of each switch INT behavior into the INT header. Figure 2 shows the structure of proposed INT header. It contains six fields, *Mapinfo*, *Length*, *etherType*, *Ingress*, *Point* and *Metadata*.
MapInfo is 2-byte bitmap information to instruct a type of requested telmetry data. We defined 6 types of telemetry data, *Input Port Number*, *Output Port Number*, *Hop Latency*, *Bandwidth*, *Switch ID* and *Timestamp*. An instruction can include a set of arbitrary types, where data size is up to 8 bytes.
Length shows the number of telemetry data in that INT header. Total size of telemetry data can be calculated as the product of each telemetry data size derived from the *MapInfo* and this *Length* information.
etherType tells the type of header comes next to this INT header.
Ingress and *Point* is used to express the target switches.
Metadata contains the telemetry data from all target switches. In Figure 2, *Metadata* fields are depicted in the case when the size of each telemetry data is 1 byte.

Next, we describe how the target switches are decided using *Ingress* and *Point* fields. *Ingress* shows the number of hops between edge-side controller and the first starting switch that handles INT operation. The number of the field is decreased on each hop until the packet reaches starting switch. Switches run the INT instruction if the *Ingress* field denotes 0, which means that the packet has been passed the directed number of hops from edge-side controller. Switches after the starting point check the *Point* field to determine whether the switch process the INT operation or not. *Point* field is 24-bit width bitmap. Each bit of the field act as a boolean flag to direct the INT execution, so following 24 switches can handle the INT operation. On checking *Point* field, each switch processes the INT operation if the first bit of the field is 1, then shift the *Point* field bits before forwarding. The switch that is directed to process the INT operation checks the *MapInfo* field, insert the directed telemetry data into INT header, and then add 1 to *Length* field. The newest added telemetry data is located next to the *Point* field. When *Point* field become 0, no more switches add telemetry data to this packet, so the switch send a whole telemetry data.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    MapInfo    |    Length     |           etherType           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Ingress    |                     Point                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Metadata n   |  Metadata n-1 |      ...      |  Metadata 1   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fig. 2: Structure of INT Header

B. Switch

We describe the design and implementation of the proposed INT enabled switch using P4. In the *Parse* step, P4 switch separates a packet into four structures, Ethernet header, INT header, INT metadata and IPv4 header. Metadata fields in our proposed INT header are separately managed as the P4 array structure that handles multiple same structure data. In P4 switch, it first separates INT header and read the value of *MapInfo* field and *Length* field to determine the size and amount of INT metadata, then prepare a header structure called *meta* which has the same bit length with an INT metadata according to the *MapInfo* field. Each P3 array data has the same size with the determined *meta* structure, and the array has *Length* counts elements to store the parsed INT metadata embedded in the packet. All packet data following the INT header are stored to a structured data and are preserved until the *Deparse* step.

In the *Ingress* step, the switch edits each field of INT header structure parsed in the *Parse* step, to control the behavior of packet after the next hop. Other ingress actions, such as checking destination IP address, finding the route, editing MAC addresses, which are not related to the proposed INT operation are also handled in this step. And proposal system is assuming the use of INT, so checking and editing each field of INT header is also handled in here. This *Ingress* step edits the field but does not add the information, so P4 switch can process this step just checking and editing the prepared *Ingress* field of INT header.

After the processing of the *Ingress* step, P4 adds information to the packet in the *Egress* step. In the common P4 processing model, this step is used to insert new headers. In our proposed system, only the INT metadata is inserted, and it is already prepared as the P4 array structure. Only the task of this step is handling all INT action except editing *Ingress* field of INT header, which is already handled in *Ingress* step.

First, P4 checks the value of *Ingress* field. If it is 0, P4 checks the first bit of *Point* field. As written in Section III, all other INT actions are processed only if the first bit is 1. On processing the INT operations, P4 checks *MapInfo* field and

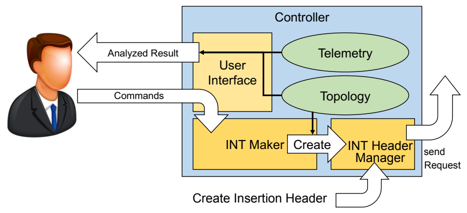

Fig. 3: Design of Controller

create a new structure with as same bit size as each element of INT metadata array. This structure is used to store all requested information inside. After all data are stored into the structure, P4 switch inserts this to the top of INT metadata array. This action could be easily achieved by using the build-in function called *push_top* in P4, which inserts blank element into the top of array, then overwrites the blank element with the new structure. The inserted elements are treated as invalid elements in initial, then P4 switch need to mark the element as valid using another build-in function called *setValid*. Finally, the switch increments the *Length* field.

In the last *Deparse* step, P4 restructures all data into packet format, which are parsed in *Parse* step and edited in *Ingress* and *Egress* steps. The packet structure separated in the *Parse* step, such as Ethernet header, INT header, INT metadata and IPv4 header, are ordered same as the original order. If some fields of the packet are removed at this switch, it is also stripped at this step. All packet data following INT header are automatically set to the next of last called header structure after the ordering of header structures has been finished.

C. Controller

Controller runs near the edge of INT-enabled area in the network. User of the INT, such as administrator, operator, or system developer, request the controller to collect some kinds of telemetry data on specified switches. On receiving the request, the controller creates an INT header which could process the request, then pass it to the edge switch to direct the insertion of the INT header.

Figure 3 shows the detail of action of the controller. The controller has two databases called “Telemetry” and “Topology”, and consists of three modules; “User Interface”, “INT Maker”, and “INT Header Manager”. Telemetry database contains the newest telemetry data collected from switches in the monitoring environment, and topology database holds the list of the switches and links in the monitoring network environment.

User Interface shows the status of the network environment in real-time and receives the INT actions’ requests from users. The visual image of the status monitor shown to the users is created from all switch IDs and links between switches in monitoring environment from topology database. In this image, switches and links are drawn with icons and lines with switch ID, and collected telemetry data are mapped on the figure with color-coding switch icons according to the anomaly of the telemetry data. The users’ requests are recorded in the controller, and users could change INT action whenever they want as the response from the monitoring map.

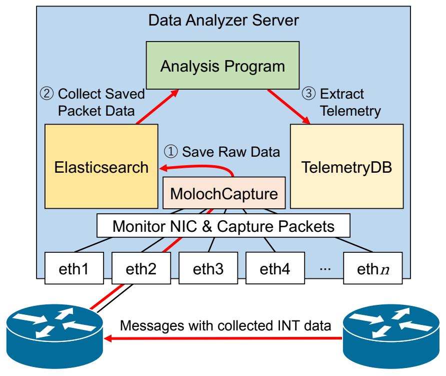

Fig. 4: Design of Data Analyzer

INT Maker is the core module of the controller which is called through User Interface. It receives a user’s commands, gets related topology data from Topology database, and compose the detail structure of INT header to complete the user’s request. The composed INT header is passed to the INT Header Manager.

INT Header Manager manages the INT Headers which is generated from users’ requests. It holds the newest set of INT headers, and updates the header which is modified by the INT Header Maker. INT Header Manager also send the request to insert the INT header to the edge switch with the target flow information.

D. Data Analyzer

Data Analyzer received the message carrying the telemetry data from INT-enabled switches. It extracts telemetry information from the messages and accumulate them for later analysis and status monitoring. Figure 4 shows the structure inside Data Analyzer. This figure depicts the Data Analyzer as a standalone server, but it could be deployed to any server, such as inside of edge node with controller. Data Analyzer consists of four modules Moloch, analysis program, and two databases that are Elasticsearch and TelemetryDB.

Moloch is used to capture the telemetry messages and analysis them. MolochCapture takes the monitoring the specific network interfaces always, and captures packets arrived at those network interfaces. It extracts and stores the raw telemetry data from the packets, and uses Elasticsearch to index them internally. Moloch also provides a GUI interface, MolochViewer, that browses the collected packets, and API to analysis captured data.

Analysis program is implemented using Python and handles the part of extracting Telemetry data from INT header, which is difficult to analysis using only the Moloch API. Analysis program gets the file path of the saved packets through Ealsticsearch using Moloch API, then parse and extract telemetry information. It also accesses Controller to find the user who request the telemetry information. The user information is difficult to carry on the INT header because of the size limitation and difficulty of the unique identity. Our program determines the user who requested the INT action from the information of both controller and telemetry data.

IV. Experiments and Evaluation

We evaluated the overhead and effectiveness of proposed selective INT mechanism. Prototype system of the proposed mechanism is implemented BMv2 P4 model switch, so the shown result is useful to estimate the tendency of the overhead which increase with the number of switches respond to the telemetry request. We also show the power of proposed pin-point telemetry through the experiments of investigating the unknown topology of the network and changing the telemetry rules in flexible and dynamic.

We prepared a Mininet environment with 24 switches. On this environment, we compared the hop latency measured in each switch by changing the number of target switches to 4, 6, 8, 12 and 24. Figure 5 shows the result of measurement. This result shows that required processing time decreases with selectively reducing the number of switches.

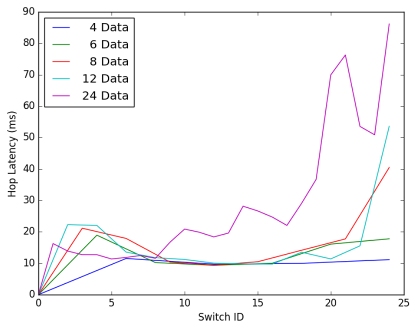

Fig. 5: Packet Transfer Latency

Next, we show an example of network administration task of trouble shooting. In this example, administrator first monitors the hop latency several surrounding switches. Latency between all switches on the route to several absolutely specified switches are monitored and displayed (Figure 6 left). Some of the Switches are depicted with the colored icons in green and red. These colors are showing the measured hop latency relatively small (green) or large (red). Switches colored in white means out of the target of the telemetry action. From this view, only one switch with ID *S8* is marked as switch with the heavy hop latency. Then user change the target of the telemetry only the switch *S8*. View of the monitoring is quickly update to the new image (Figure 6 rigth).

V. Discussion

The result of section IV shows that proposed pin-point INT can make the number of INT target switch to a minimum and reduce the overhead of the telemetry. Our proposed system can specify the target switches of INT action using both of relative distance from Controller and absolute switch ID. Using the relative distance, our examples shows that the connectivity and status of the network is easily determined only using the in-band telemetry monitoring. Our proposed system will enable users to specify the telemetry target switches flexibly with the combination of the relative distance and absolute IDs. The prototype monitoring system supports generation of the INT header arranged for the users' request, manages the telemetry

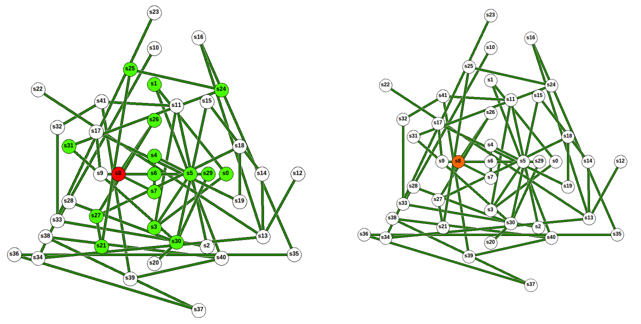

Fig. 6: Monitor View (original and updated)

requests, gather and analysis the telemetry data, and view the graphical image of the topology and status. This will help to reduce the cost on monitoring complicated networks.

The limitation of the proposed system is the range of the switches which are the targets in-band telemetry. Our proposed mechanism specifies the INT target switches using *Ingress* offset and *Point* bitmap. It enables to collect the telemetry up to a series of 24 switches starting from the 255 hops far from the edge switch. This means maximum degree of the network that our system can monitor is 279. We consider the limitation of the network size is practically enough for current network environment. The limitation of a series of 24 switches can be relaxed with some brief extension of the proposed mechanism. One approach is to use the multiple INT headers with the different *Ingress* offset specifying the multple starting points. Another approach is modifying the lengths of the bit patterns of *Ingress* and *Point*. In both way, total size of the telemetry header is basically limited. With considering this limitation, we consider a series of 24 switches are also practically enough.

VI. Conclusion

In this paper, we proposed a flexible in-band network telemetry (INT) system that enables pin-point designation of telemetry target switches. Each switch supporting our INT mechanism determines whether it executes the telemetry action or not with just processing the telemetry header in each packet. We implemented the prototype system using P4 BMv2 model switch on Mininet and evaluate the effectiveness of our system through some experiments. The prototype system manages users' INT requests, and the result of monitoring graphically. Our proposed system enables administrators and operators to monitor the network status easily and precisely.

References

[1] 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*, vol. 17, no. 2, pp. 708–721, 2020.

[2] R. Ben Basat, S. Ramanathan, Y. Li, G. Antichi, M. Yu, and M. Mitzenmacher, "PINT: Probabilistic In-Band Network Telemetry," in *Proceedings of SIGCOMM '20*, 2020, p. 662 − 680. [Online]. Available: https://doi.org/10.1145/3387514.3405894

[3] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming Protocol-Independent Packet Processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, p. 87 − 95, Jul. 2014. [Online]. Available: https://doi.org/10.1145/2656877.2656890