An Integrated Solution for High-efficiency In-band Network Telemetry

Xinxin Xiong
School of Informatics, Xiamen University
China

Yi Xie
School of Informatics, Xiamen University
China

Xiaochou Chen*
Information and Network Services Center, Xiamen University
China

Shaojie Zheng
Information and Network Services Center, Xiamen University
China

Wenju Huang
School of Informatics, Xiamen University
China

Jiahao Feng
School of Informatics, Xiamen University
China

ABSTRACT

The advent of in-band network telemetry (INT) facilitates the dynamic and fine-grained monitoring of network conditions. Nevertheless, the current INT specification incurs a substantial measurement overhead, diminishing bandwidth utilization, and potential data fragmentation. Moreover, the rapid accumulation of telemetry reports may give rise to data lakes in the collector, resulting in computational burden and telemetry degradation. To address these two problems, we propose an integrated solution for high-efficiency in-band network telemetry, including an improved INT scheme, SF-INT, to reduce measurement overhead and the usage of SmartNIC to accelerate report processing. SF-INT employs network node registers to store the telemetry information collected at previous nodes by previous packets. When a packet arrives at one node, the node performs store-and-forward actions to determine the telemetry information stored in the current register and the telemetry information carried by the packet to the next hop. The packet carries only one node's telemetry information during a telemetry process, thus keeping a constant packet size. In contrast, the standard INT increases the packet size due to the hop-by-hop insertion of metadata. SmartNIC offloads report processing from the CPU and obtains comprehensive telemetry data by analyzing a set of telemetry reports. The testbed experiment results have demonstrated that our solution reduces measurement overhead and efficiently processes reports at nearly the line rate.

CCS CONCEPTS

• **Networks** → **Network measurement**; *Network monitoring*.

KEYWORDS

In-band Network Telemetry, SmartNIC, Programmable Network

*Corresponding author: Xiaochou Chen, cxc@xmu.edu.cn

APNet 2024, August 3–4, 2024, Sydney, Australia

ACM ISBN 979-8-4007-1758-1/24/08
https://doi.org/10.1145/3663408.3663425

ACM Reference Format:
Xinxin Xiong, Yi Xie, Xiaochou Chen, Shaojie Zheng, Wenju Huang, and Jiahao Feng. 2024. An Integrated Solution for High-efficiency In-band Network Telemetry. In *The 8th Asia-Pacific Workshop on Networking (APNet 2024), August 3–4, 2024, Sydney, Australia.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3663408.3663425

1 INTRODUCTION

In recent years, there has been a notable expansion in the scale of data center networks. Network congestions, traffic bursts, and elephant flows may happen in fast-changing networks [2]. Network measurement technology assists in understanding network conditions, facilitating the timely detection of these issues. Traditional network measurement focuses on end-to-end performance, but monitoring modern networks requires more granular information within a network. With the rise of programmable data plane (PDP), in-band network telemetry (INT) has attracted widespread interest. INT enables the source node to insert an INT header containing telemetry instructions into ordinary packets and collect metadata from each network node, achieving fine-grained monitoring of network conditions.

However, the current INT specification [6] has two limitations. First is the problem of high overhead. When one node on the telemetry path processes and forwards a data packet, the node's metadata, such as "Node ID" and "Hop Latency," is inserted after the INT header in this packet. Consequently, the packet size will consistently grow with the path length, leading to excessive overhead and reducing link utilization. In the context of IPv4, packet fragmentation occurs when the packet size exceeds the Maximum Transmission Unit (MTU). In IPv6, however, the prohibition of on-path fragmentation causes packet drops, prompting packet retransmissions. Both scenarios result in performance degradation. Reserving appropriate space for INT in the packet header is also challenging because telemetry paths change dynamically. Insufficient space leads to packet loss or fragmentation, the superfluous space waste bandwidth.

Second is the processing issue of high-speed telemetry reports. The sink node strips the inserted INT header and metadata from each ordinary packet and generates a telemetry report. Subsequently, the INT collector obtains and stores telemetry data from several reports for further analysis. With the increased network bandwidth, continuously generated reports at high speeds may accumulate as a data lake [21] in the collector if they are not processed

promptly. This phenomenon not only diminishes the measurement accuracy of INT but also increases computational pressure. CPU multi-threading may alleviate this to a certain extent, at the cost of heavy resource burden.

To reduce the INT overhead, researchers employed different sampling-based schemes [10–13] to reduce the number of data packets or the amount of INT metadata involved in telemetry, thereby reducing bandwidth usage. For example, INT-label [12] utilizes a label sampling-based scheme to reduce overhead. These sampling schemes provide a tradeoff between measurement accuracy and bandwidth usage. However, they still need to address the problem of packet size increasing with the number of nodes, and the INT collector also needs optimization. Recently, the eXpress Data Path (XDP) [15, 16] is used to speed up the collection and analysis of telemetry data, thereby reducing CPU usage and storage consumption. Moreover, SmartNIC applies its programmable features to implement customized functions, such as accelerating the processing of telemetry reports [3]. Since SmartNIC offloads many functions from the CPU, much CPU resource is liberated from the hardware level.

We propose an integrated solution for high-efficiency in-band network telemetry. It implements high efficiency by designing an improved method named Stateful-INT (SF-INT) to address the overhead problem and leveraging a SmartNIC-equipped collector to process telemetry reports quickly. SF-INT's feature exists in each network node's register, which stores the telemetry information containing some fields in the SF-INT header and metadata. When a packet arrives, the node will perform store-and-forward actions based on the telemetry information in its register. With the newly defined SF-INT header, the packet size remains constant for the same telemetry instructions. Then, the sink node separates the metadata from an ordinary packet and generates one telemetry report. Finally, SmartNIC accelerates telemetry report processing, prevents a data lake from appearing in the collector, and reduces the CPU's burden. After receiving a set of telemetry reports with the same ?02:4C in the SF-INT header, SmartNIC's high-concurrency design enables efficient report parsing, facilitating metadata extraction and obtaining comprehensive telemetry data for all nodes.

The main contributions include:

(1) We propose an integrated solution for improving INT's efficiency with low overhead and powerful report processing capacity. SF-INT utilizes node's register to store telemetry information and deals with arriving packets according to the store-forward action rules. The metadata length will remain the same during transmission because the newly defined SF-INT will not insert nodes' metadata hop-by-hop, then saving bandwidth. The SmartNIC-equipped collector accelerates telemetry report processing while reducing the CPU consumption. It can obtain complete telemetry data for all nodes on the path from a set of reports with the same ?02:4C .

(2) We have implemented the integrated solution, where SF-INT is prototyped on Intel Tofino programmable switches, and telemetry reports are processed on Netronome SmartNIC. Experiment results have shown that SF-INT effectively reduces telemetry overhead, where the packet size is only affected by telemetry instructions and does not increase with the number of network nodes. The additional latency due to the collection of multiple reports is slight, such as only 9.3`B in an 8-node telemetry path. Therefore, SF-INT bargains away slight delay in exchange for the low-overhead attribute. Moreover, our solution can achieve a processing throughput of 7.13 "??B, and the processing rate of telemetry reports keeps up with the arrival rate.

(3) We have implemented and evaluated the integrated solution in the IPv6 environment. Our work is a helpful attempt to promote the application of INT in the next-generation network.

2 BACKGROUND

2.1 In-band Network Telemetry

In-band network telemetry (INT) is a framework for collecting and reporting network states through a data plane, which can help network operators gain a more granular understanding of network states, facilitate network troubleshooting, and monitor performance. The current INT specification [6] has three types of INT headers: MD-type, Destination-type, and MX-type. It is common practice to employ the MD type to obtain detailed information about all nodes along the path, whose process is shown in Fig.1. When an ordinary packet arrives, a source node inserts an INT header containing telemetry instructions, followed by its INT metadata. Telemetry instructions determine the information (e.g., Node ID, Hop latency) required to collect. Accordingly, the transit node inserts its INT metadata following the INT header and then forwards the packet to the next node. The sink node strips the INT header and aggregates INT metadata inserted in the packet to generate a report. Finally, the original packet continues the journey toward the destination, while the telemetry report is transmitted to the collector to obtain telemetry data.

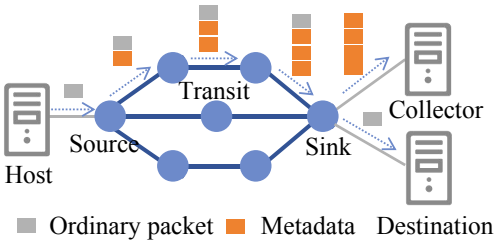

Figure 1: Overview of INT

2.2 SmartNIC

SmartNIC has many advantages [19] and is widely used in many areas, such as offloading functions, accelerating packet processing [4, 8, 17], and storage optimization [18, 20] . This paper adopts Netronome SmartNIC, which is equipped with the NFP-4000[5] multi-threaded and multi-core stream processor, where each stream processing core, micro-engine (ME), is regarded as a small CPU core. Therefore, Netronome SmartNIC can execute programs concurrently. It has two 10 1?B Ethernet interfaces and 2 of DDR3 DRAM. We employ this memory to design a ring buffer for caching INT metadata extracted from telemetry reports. Moreover, Netronome SmartNIC supports P4 Match-Action operations and a subset language of C and MicroC. Therefore, we use P4 to perform preliminary

parsing of telemetry reports and MicroC to perform the fine-grained extraction and cache operations.

3 METHODOLOGY

3.1 SF-INT Design

In the standard INT scheme, the hop-by-hop process of inserting metadata causes packet size to grow with the number of nodes, leading to high telemetry overhead. To address this issue, we designed a low-overhead scheme, SF-INT. Its main idea is to utilize one register on each programmable switch to store telemetry information containing the values of *?02:4C* , *()* , and the telemetry metadata of a particular node, defined in the SF-INT header (refer to Subsection 3.1.1). When a node receives a packet with the SF-INT header, it performs a store-and-forward action based on the telemetry information of the arrival packet and the telemetry information stored in its registers (refer to Subsection 3.1.2). Each forwarded packet only contains one node's telemetry information, while the packet size is fixed along the entire path, which reduces telemetry overheard compared with the standard INT. The IPv6 encapsulation of SF-INT is presented in Subsection 3.1.3.

3.1.1 SF-INT header. The SF-INT header is shown in Fig.2, which is modified based on the standard INT-MD header [6]. We defined some new fields in the reserved bits of the INT-MD header. The SF-INT flag field, denoted as , occupies 1 bit, whose values of "1" and "0" indicate that this packet is an SF-INT packet or an standard INT header. The urgent field, denoted as * , occupies 1 bit, where "1" represents the emergency mode and will switch to the standard INT. The stateful field *()* occupies 2 bits, which indicates the current state of the SF-INT header in the packet. Three states represent the order of priority from high to low: High (H), Medium (M), and Low (L). It is an essential factor when one network node determines store-and-forward actions.

We reuse the original "Domain Specific ID" field as the new "packetID " field, which occupies 16 bits. SF-INT obtains comprehensive metadata of all nodes along the transmission path by dealing with a set of SF-INT reports with the same *?02:4C* . Moreover, the SF-INT metadata stack, followed by the SF-INT header, inserts or updates one node's telemetry metadata according to the store-and-forward action rules.

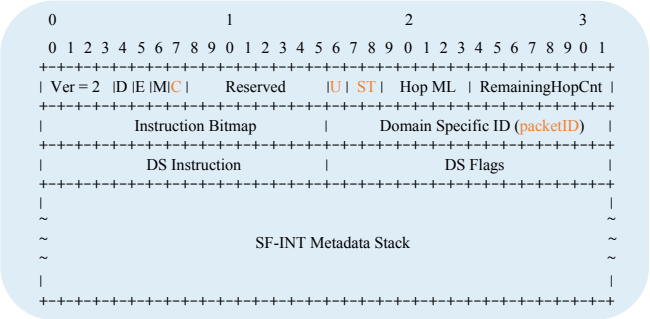

Figure 2: SF-INT header and metadata stack

3.1.2 Store-and-forward Actions. We design the store-and-forward rules based on the particular fields (i.e., *()* priority state and *?02:C*) in the SF-INT header. Each network node utilizes a register to temporarily store the telemetry information (containing the *()* , *?02:4C* fields in the SF-INT header and metadata). When one packet arrives, the network node updates its local register and forwards this packet to the next hop based on the *?02:4C* and *()* using the forwarding rules in Table1. Here, the packet only carries the current node's telemetry information or the current register's telemetry information. Finally, the SmartNIC-equipped collector acquires the telemetry metadata of the full-path nodes after receiving a set of SF-INT reports with the same *?02:4C* . Therefore, SF-INT achieves in-band network telemetry with a constant packet size.

Table 1: SF-INT's Rules of store-and-forward actions

Rule	State		Action	
	Arrival packet	Register store	Forward packet	Register update
1	M(A)	H(S)	H(S)	L(A)
2	M(A)	L(S)	M(C)	H(A)
3	M(A)	E()	M(C)	H(A)
4	H(A)	H(S)	H(S)	H(A)
5	H(A)	L(S)	L(S)	L(A)
6	H(A)	E()	H(A)	E()
7	L(A)	H(S)	H(S)	L(A)
8	L(A)	L(S)	L(S)	L(A)
9	L(A)	E()	L(A)	E()

The complete forwarding rules in Table 1 ensure that the INT collector is able to obtain the full path of telemetry data by aggregating a set of SF-INT reports with the same *?02:4C* . Given a telemetry instruction, each packet carries the same amount of telemetry information with the fixed packet size. The design principle of rules is letting the packet carry the metadata with high priority and saving the metadata with low priority in the node's register. For simplicity, we adopt some abbreviations: (1) Empty register, () denotes the node register is empty. (2) denotes the telemetry metadata carried by the arrival packet; *(* denotes the metadata stored in the node register; and denotes the metadata collected at the current node. (3) List the priorities in descending order: high (H), medium (M), and low (L).

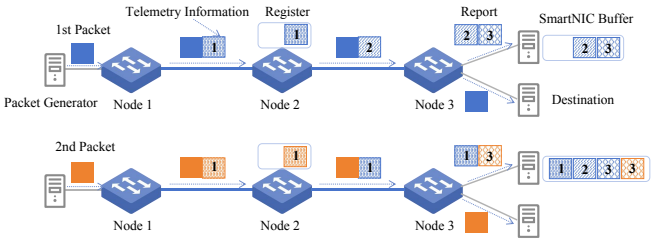

Figure 3: SF-INT Prototype: store-and-forward process

As an example, Fig.3 shows a store-and-forward process in a simple network. Here, a packet generator generates ordinary IPv6

packets and sends them to a destination after traversing three network nodes. When the first packet (with blue color) arrives at Node1, Node1 inserts its telemetry metadata (with blue grids labeled 1) and the priority state of ( ) = " into the SF-INT header of the first packet. When the first packet arrives at Node2, its register is empty. According to Rule 3, Node2 stores the first packet's telemetry information (?02:4C = 1, ( ) = , and Node1's metadata) in the local register, while the forwarded packet carries Node2's metadata and sets ( ) = " . As a sink node, Node3 directly sends the SmartNIC an SF-INT report, which includes its telemetry metadata (with blue grids labeled 3) and the carried telemetry metadata of Node2 (with blue grids labeled 2).

After receiving the 2nd packet, Node1 inserts its telemetry metadata (with orange grids labeled 1) and the priority state of ( ) = " into the SF-INT header of the 2nd packet. When the 2nd packet arrives at Node 2, its register is not empty, so Node 2 selects rule 1 according to the information matching. Node2 stores the carried telemetry information (?02:4C = 2, ( ) = !, and Node1's metadata) in the local register (with orange grids labeled 1). The forwarded packet carries Node1's metadata stored in its register (with blue grids labeled 1) and sets ( ) = . Again, Node3 directly sends the SmartNIC an SF-INT report, which includes its telemetry metadata (with orange grids labeled 3) and the carried telemetry metadata of Node1 (with blue grids labeled 1). Finally, the SmartNIC collector gets the telemetry metadata at all nodes collected by the first packet (?02:4C = 1) in the full path.

3.1.3 IPv6 Encapsulation of SF-INT. We utilize the hop-by-hop extension header in an IPv6 packet to encapsulate an SF-INT packet, where the "next header" field in the basic IPv6 header equals 0. As shown in Fig. 4(a), the SF-INT header and other related headers are embedded between the IPv6 basic header and the packet payload, where a newly defined "INT option" represents the type of the subsequent header and its length. The default type is the SF-INT header defined in Subsection 3.1.1; one optional type is the standard INT header in INT 2.1 [6]. The utilization of hop-by-hop extensions ensures that all network nodes process SF-INT packets. The newly designed INT option and the SF-INT header facilitate the completion of telemetry tasks without modifying the fields employed by existing protocols.

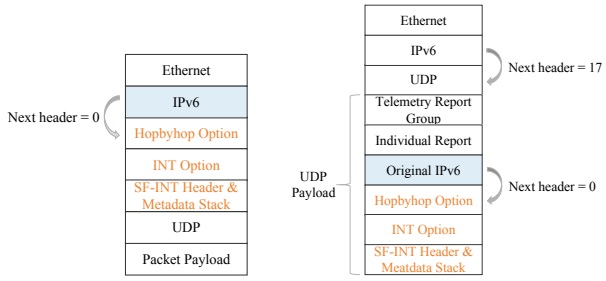

Figure 4: Packet Formats of SF-INT

Similarly, we modify the format of telemetry reports based on Telemetry Report Format 2.0 [7]. As shown in Fig.4(b), we use the UDP payload to carry the header and metadata stripped from the sink node, where the "next header" field in the basic IPv6 header equals 17.

3.2 SmartNIC Processing

Currently, network bandwidth is on the rise, and packet transmission rates are increasing. Telemetry reports are generated at high speed when conducting the tasks of in-band network telemetry. Using a CPU will consume many resources, but the processing speed of reports may not meet requirements. By leveraging SmartNIC, we offload the report processing from the CPU, effectively reducing its workload and achieving high-performance report processing.

We have a SmartNIC-equipped collector to process reports. The SF-INT reports generated by the sink node enter the SmartNIC through the physical interface. As shown in Fig.5, the report processing process mainly consists of three parts: (1) The P4 module performs parsing and Match-Action operations and then calls the C function. (2) The Micro-C module defines the storage structure and writes the metadata into that structure. (3) The host module enables threads to read data located in the storage structure and store it in the Redis database according to requirements.

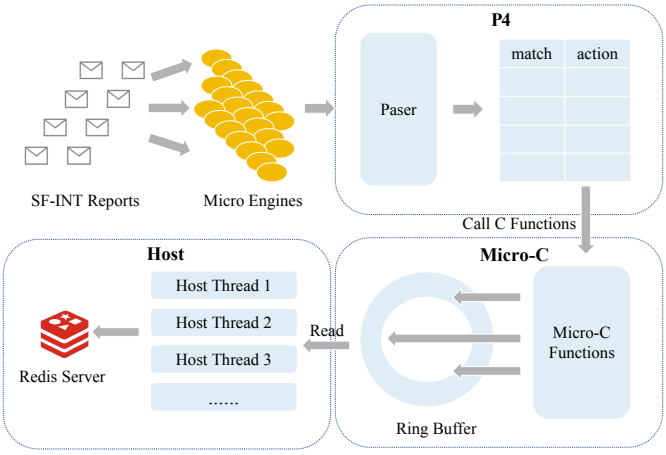

Figure 5: SmartNIC's Processing of SF-INT Reports

P4 module: We utilize 54 micro engines (MEs) on the Netronome SmartNIC for programming, each with four threads running the same program. After SF-INT reports enter through the physical interface, the built-in load-balancing algorithm allocates them to different MEs for processing. The P4 module will perform field parsing on these reports according to the report specification shown in Fig. 4(b), and the successfully parsed reports will be sent to the Match-Action section. Then, this part will match the field in the SF-INT header to filter out the fully parsed SF-INT reports. Finally, the fully parsed reports are handed over to the Micro-C module for processing.

Micro-C module: We define a ring buffer of 32K in the Micro-C module to store the metadata in the reports. We utilize the ?02:4C field of each SF-INT report as the hash index. When the ?02:4C exceeds the buffer length, the index will be reset to start a new storage round. In this way, the metadata in the reports with the same

?02:4C will be stored in the same location to realize the collection of complete telemetry data. The telemetry instructions determine the storage structure, saving SmartNIC memory consumption. For example, if five telemetry instructions need to be measured, the storage structure contains five entries.

We use semaphore mechanism to enable MEs in SmartNIC to compete for the position of the buffer. When a thread of a ME occupies the location, other threads need to wait. Micro-C supports embedded assembly language, and we utilize this property to implement the operations of occupying and releasing semaphores. As a result, this approach reduces the overhead associated with reading and writing to the SmartNIC's memory. When a set of reports with the same ?02:4C is processed, a complete set of metadata (i.e., telemetry data for that all nodes) will be stored at this location.

Host module: The host module enables threads to read telemetry data on the ring buffer. Based on the specific telemetry needs, these telemetry data can be employed for analysis or periodically refreshed into the Redis database.

4 EVALUATION

4.1 Testbed Setup

As shown in Figure 6, the testbed includes eight network nodes operating at Intel Tofino P4 programmable multi-pipeline switches (2 pipelines) [1], two Ubuntu servers (kernel version 4.15.0-142-generic) with 24 Intel Xeon 2.30GHz CPU cores and 128G memory, and one destination host. Each link between two switches is 100 1?B (with a thick solid line), while each link between a switch and a server/host is 10 1?B (with a thin solid line).

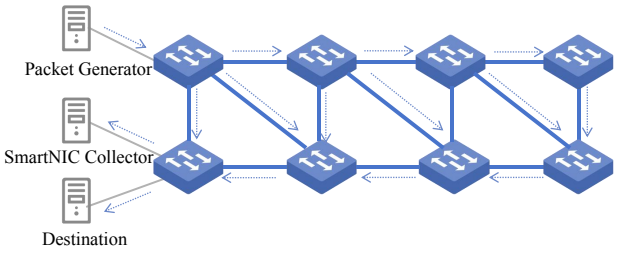

Figure 6: Testbed Topology

One server is the INT collector equipped with Netronome Agilio CX 2x10GbE SmartNIC, responsible for processing telemetry reports and analyzing telemetry data as required. The other server is the packet generator with the DPDK[9] test tool TRex [14], which sends 64-byte packets to the host with a data rate of 10 1?B. The node connected to the packet generator is the source node, the node connected to the INT collector is the sink node, and the others are transit nodes. We conduct experiments in the IPv6 environment to promote the INT deployment in the next-generation network.

Next, we evaluate the performance of our solution when the transmission path consists of different numbers of network nodes (from 2 to 8) and the telemetry instructions increase from 2 to 5. These telemetry instructions include "Node ID," "Hop latency," "Ingress timestamp," "Egress timestamp," and "Level 2 Ingress/Egress interface ID", selected from the INT specification [6]. In particular, "Node ID" is the mandatory instruction identifying the node inserting the telemetry metadata.

[1]One pipeline can act as a network node since it contains the blocks of ingress, packet replication engine, and egress, having their own parser, deparser, and match-action units. And each pipeline is mapped to a set of physical switch ports.

4.2 Telemetry Overhead

To evaluate the telemetry overhead in an IPv6 environment, we measured the size of a telemetry packet when launching 5 telemetry instructions on the transmission path with 8 network nodes. As shown in Fig. 7, the standard INT lengthens the packet during hop-by-hop forwarding because each node inserts its INT metadata. It will lead to high overhead, especially when the path is long. Many sampling-based INT schemes [10–13] also suffer from the same problem because of embedding metadata continuously. In contrast, SF-INT exhibits the overhead benefit of constant-size packets during the whole path for identical telemetry instructions. Each node executes store-and-forward actions according to the register's telemetry information and predefined rules. The length of telemetry packets is affected by the number of instructions instead of the number of nodes.

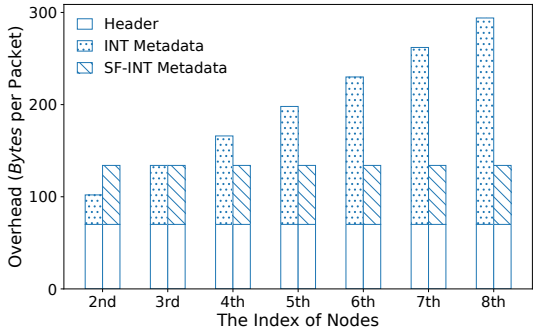

Figure 7: Overhead comparison of SF-INT and INT

4.3 Throughput

We use TRex to send IPv6 packets up to 14.7 " ??B, 9.89 1?B (nearly the full rate, 10 1?B) when changing the number of telemetry instructions from 2 to 5. Fig. 8(a) compares the packet sending rate and the total throughput, which is the output rate of the processed report in SmartNIC. The throughput of our integrated solution increases with the sending rate, and its maximum reaches 9.13 Gbps. It demonstrates that our solution, which consists of SF-INT and a SmartNIC-equipped collector, can efficiently perform telemetry measurements and process the generated telemetry reports over a 10 link (nearly the line rate).

We further investigate the report processing capacity of SmartNIC. As shown in Fig 9, the processing rate of SmartNIC reaches up to 7.13 " ??B in the case of 2 telemetry instructions. Moreover, the curve with the same number of instructions is stable and will not decrease with the number of nodes. Compared to the scheme that employed SmartNICs to process standard INT reports [3], our SF-INT holds distinct advantages. Comparing Fig.9(a) and Fig.9(b), for the same number of telemetry instructions, the report departure

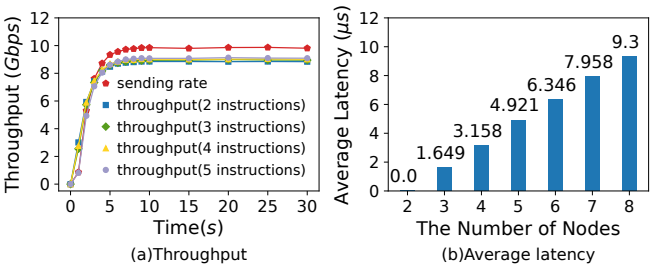

Figure 8: Perfomrance of our integrated solution

rate (Output) is near the report arrival rate (Input) in the SmartNIC, indicating its excellent processing capacity. The extraction and analysis of SF-INT metadata from telemetry reports will hardly impact the total throughput. It is also noteworthy that the SmartNIC offloads report processing from the host CPU, thus alleviating the CPU burden at the hardware level.

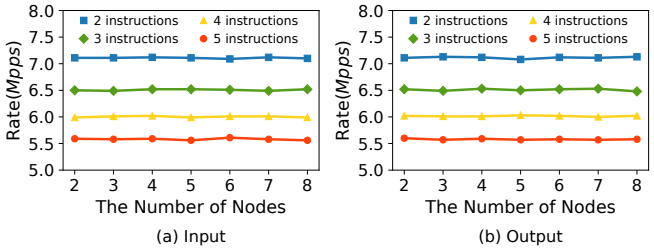

Figure 9: Processing rate of SmartNIC

4.4 Latency

SF-INT obtains all nodes' metadata from a set of telemetry reports with the same *?02:4C* . According to the store-and-forward rules, the collector requires # − 1 reports to obtain comprehensive metadata on the #-node path. It introduces an additional latency for receiving these telemetry reports accompanied by multiple packets, which arrive at different times.

Using the *tshark* tool, we captured the nanosecond-level timestamps of telemetry reports when arriving at the SmartNIC's physical port. Then, we calculated the average interval delay of # − 1 adjacent reports for each round of 65536 reports [2]. As shown in Fig. 8(b), the number of nodes mainly affects the latency. The more nodes there are, the more telemetry reports with the same *?02:4C* are required to obtain comprehensive metadata. However, packet arrival rates are speedy in today's data center networks. Experimental results show that the latency is small, with an average of 9.3 `B in the 8-node path, which is acceptable to many applications. Furthermore, 99% of latency values are below the average. That is, SF-INT bargains away slight delay in exchange for the low-overhead attribute.

[2]The field of *packetID* occupies 16 bits, 65536=2^{16}.

5 CONCLUSION

This paper has designed and implemented the integrated solution for high-efficiency in-band network telemetry, including the low-overhead SF-INT and the high-performance SmartNIC-equipped collector. We have built a real testbed to verify our solution using Intel Tofino switches and Netronome SmartNIC. Experimental results have shown that our solution can achieve low overhead by keeping a constant packet size during the whole path at the cost of slight delay. Moreover, the SmartNIC can process SF-INT telemetry reports at nearly the line rate. In our future work, we will make improvements in the following areas:

Comparisons under different application scenarios. SF-INT has demonstrated significant potential in addressing the issue of INT packet inflation. We will conduct a comprehensive evaluation of SF-INT on various application scenarios, comparing it with the popular solutions in the research community, such as PINT [1], INT-Label [12], and Delta-INT [11]. This evaluation will consider multiple performance metrics, including switch memory utilization, bandwidth occupancy, and other relevant factors.

Couple with the existing sampling INT schemes. Given that real applications may not require every packet to embed the INT metadata for all nodes. We plan to further reduce overhead by skipping some inessential nodes, sampling partial packets, and removing redundant metadata, which increases the adaption to various environments.

Introduce 40G SmartNICs. Considering the increasing bandwidth of modern networks, we will experiment with higher specification SmartNICs to achieve higher throughput. 40G SmartNICs exhibit superior capabilities compared with their 10G counterparts and stand an excellent chance to accelerate the processing of telemetry reports. We will also compare our solution with the mature ones, like DPDK[9], by evaluating the speed in processing telemetry reports and CPU consumption.

Enhance the ability to handling exceptions. When facing special network events such as packet loss and route updates, enhance INT to quickly obtain the complete path's telemetry reports and flexibly respond to complicated network conditions such as multipath data transmission.

ACKNOWLEDGMENTS

This work was partially supported by the National Key R&D Program of China (No. 2023YFB4502703). Zhuobin Xu and Xiaoli Lu from the Information and Network Services Center of Xiamen University are acknowledged for their help with the experiments.

REFERENCES

[1] Ran Ben Basat, Sivaramakrishnan Ramanathan, Yuliang Li, Gianni Antichi, Minian Yu, and Michael Mitzenmacher. 2020. 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*. 662–680.

[2] Henrique B. Brum, Carlos R. P. Dos Santos, and Tiago C. Ferreto. 2023. Providing Fine-grained Network Metrics for Monitoring Applications using In-band Telemetry. In *IEEE International Conference on Network Softwarization (NetSoft)*. 116–124.

[3] Yixiao Feng, Sourav Panda, Sameer G Kulkarni, KK Ramakrishnan, and Nick Duffield. 2020. A Smartnic-Accelerated Monitoring Platform for In-band Network Telemetry. In *IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. 1–6.

[4] Jiaxin Lin, Tao Ji, Xiangpeng Hao, Hokeun Cha, Yanfang Le, Xiangyao Yu, and Aditya Akella. 2023. Towards Accelerating Data Intensive Application's Shuffle Process Using SmartNICs. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 7, 2 (2023), 1–23.

[5] Netronome Systems, Inc. 2020. *Netronome NFP-4000 Flow Processor.* https://d3ncevyc0dfnh8.cloudfront.net/media/documents/PB_NFP-4000-7-20.pdf

[6] P4.org. 2020. *In-band Network Telemetry (INT) Dataplane Specification Version 2.1.* https://p4.org/p4-spec/docs/INT_v2_1.pdf

[7] P4.org. 2020. *Telemetry Report Format Specification Version 2.0.* https://p4.org/p4-spec/docs/telemetry_report_v2_0.pdf

[8] Sourav Panda, KK Ramakrishnan, and Laxmi N Bhuyan. 2022. Synergy: A Smartnic Accelerated 5G Dataplane and Monitor for Mobility Prediction. In *IEEE International Conference on Network Protocols (ICNP)*. 1–12.

[9] DPDK Project. 2024. *Data Plane Development Kit.* https://www.dpdk.org

[10] Toshihiro Sato and Toshio Hirotsu. 2021. A Flexible P4-Based Pin-Point In-Band Network Monitoring. In *Asia-Pacific Network Operations and Management Symposium (APNOMS)*. 258–261.

[11] Siyuan Sheng, Qun Huang, and Patrick PC Lee. 2021. DeltaINT: Toward general in-band network telemetry with extremely low bandwidth overhead. In *IEEE International Conference on Network Protocols (ICNP)*. 1–11.

[12] Enge Song, Tian Pan, Chenhao Jia, Wendi Cao, Jiao Zhang, Tao Huang, and Yunjie Liu. 2021. INT-label: Lightweight In-band Network-Wide Telemetry via Interval-based Distributed Labelling. In *IEEE INFOCOM Conference on Computer Communications*. 1–10.

[13] Dongeun Suh, Seokwon Jang, Sol Han, Sangheon Pack, and Xiaofei Wang. 2020. Flexible Sampling-based In-band Network Telemetry in Programmable Data Plane. *ICT Express* 6, 1 (2020), 62–65.

[14] TRex Team. 2020. *TRex Traffic Generator.* https://trex-tgn.cisco.com

[15] Nguyen Van Tu, Jonghwan Hyun, Ga Yeon Kim, Jae-Hyoung Yoo, and James Won-Ki Hong. 2018. Intcollector: A High-Performance Collector for In-band Network Telemetry. In *International Conference on Network and Service Management (CNSM)*. 10–18.

[16] Jonathan Vestin, Andreas Kassler, Deval Bhamare, Karl-Johan Grinnemo, Jan-Olof Andersson, and Gergely Pongracz. 2019. Programmable Event Detection for In-band Network Telemetry. In *IEEE international conference on cloud networking (CloudNet)*. 1–6.

[17] Feng Wang, Gongming Zhao, Qianyu Zhang, Hongli Xu, Wei Yue, and Liguang Xie. 2023. OXDP: Offloading XDP to SmartNIC for Accelerating Packet Processing. In *IEEE International Conference on Parallel and Distributed Systems (ICPADS)*. 754–761.

[18] Xingda Wei, Rongxin Cheng, Yuhan Yang, Rong Chen, and Haibo Chen. 2023. Characterizing Off-path SmartNIC for Accelerating Distributed Systems. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*. 987–1004.

[19] Jiarong Xing, Yiming Qiu, Kuo-Feng Hsu, Songyuan Sui, Khalid Manaa, Omer Shabtai, Yonatan Piasetzky, Matty Kadosh, Arvind Krishnamurthy, TS Eugene Ng, et al. 2023. Unleashing SmartNIC Packet Processing Performance in P4. In *Proceedings of the ACM SIGCOMM Conference*. 1028–1042.

[20] Yitian Yang and Youyou Lu. 2023. NICFS: a file system based on persistent memory and SmartNIC. *Frontiers of Information Technology & Electronic Engineering* 24, 5 (2023), 675–687.

[21] Elisabeta Zagan and Mirela Danubianu. 2023. Data Lake Architecture for Storing and Transforming Web Server Access Log Files. *IEEE Access* 11 (2023), 40916–40929.