In-order INT: Efficient Reordering of Out-of-order In-band Network Telemetry Packets via CPU-FPGA Co-design

Zongrui Sui[1,2], Lizhuang Tan[1,2], Huiling Shi[1,2], Wei Zhang[1,2], Peiying Zhang[1,3]
[1]*Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Ji'nan, China*
[2]*Shandong Provincial Key Laboratory of Computing Power Internet and Service Computing, Shandong Fundamental Research Center for Computer Science, Jinan, China*
[3]*Qingdao Institute of Software, College of Computer Science and Technology, China University of Petroleum (East China), Qingdao, China*
Email: zongruisui2002@gmail.com, lzhtan@qlu.edu.cn, shihl@sdas.org, wzhang@sdas.org, zhangpeiying@upc.edu.cn

***Abstract*—With the emergence of Software-Defined Networking (SDN) and data plane programmability, In-band Network Telemetry (INT) has become a novel network measurement approach, enabling end-to-end, hop-by-hop network visibility. However, due to the complexity of network topologies and limitations in telemetry orchestration schemes, telemetry servers often encounter severe out-of-order issues when processing incoming telemetry reports. This results in stale and unreliable data source for upper-layer telemetry applications. This paper proposes a CPU/FPGA co-designed out-of-order reordering system for INT, which combines online and offline sorting techniques to enhance the processing efficiency of telemetry servers. To the best of our knowledge, this is the first study addressing out-of-order reordering in INT. Experimental results show that compared to a CPU-only solution, our approach achieves a 12.5X speedup on the 10GE NIC.**

***Index Terms*—Network Measurement, In-band Network Telemetry, Heterogeneous Computing, Online Reordering, Offline Reordering, Reordering Acceleration**

I. Introduction

As an emerging network measurement technology, In-band Network Telemetry (INT)[1] embeds telemetry instructions and information into business traffic. It no longer uses a separate control/management plane or out-of-band channel, and can achieve end-to-end network status measurement. In recent years, INT-based network routing, congestion control[2], and fault detection have significantly improved the efficiency of network control and management.

INT is a flow-level or path-level measurement method. However, when it is applied to network-wide measurement tasks, it faces a severe out-of-order problem. Based on the task orchestration[3], telemetry data is collected through multiple INT paths, which often partially overlap. Even telemetry packets originating from the same source flow may traverse different INT paths, resulting in inconsistencies between the order of generation and the order of arrival at the collector. If left unprocessed, such disorder can compromise the accuracy of temporal analysis for upper-layer applications. Therefore, reordering telemetry data at the collector side is essential to ensure data integrity and temporal correctness.

With the advancement of heterogeneous computing, CPU and Field Programmable Gate Array (FPGA) based heterogeneous architectures have been widely adopted for accelerating a variety of applications[4]. From the perspective of accelerating the reordering process in In-band Network Telemetry (INT), the introduction of FPGA significantly enhances the reordering efficiency of the telemetry server, and releases more CPU computational resources for INT applications.

In this paper, the disordering of In-band Network Telemetry data is discussed, and we propose an INT (In-band Network Telemetry) processing scheme to accelerate the process of reordering. The scheme employs the optimized online reordering architecture, which saves comparator resources compared with traditional architecture. On the basis of online reordering, we also propose an offline reordering architecture, which reduces the time required for offline reordering.

II. Background

A. Out-of-order Issue of INT

Fig. 1 illustrates the workflow of in-band network telemetry (INT). The SDN controller, according to the task orchestration policy, issues telemetry instructions to the network devices along the telemetry path, which embed their status information into service packets during forwarding, enabling real-time, end-to-end, per-packet information collection. The task orchestration typically generates multiple telemetry flows to cover the entire end-to-end path, which

This work was supported in part by the Shandong Provincial Natural Science Foundation under Grant No. ZR2022LZH015, ZR2023QF025, ZR2024LZH006 and ZR2023LZH017, the National Natural Science Foundation of China under Grant 62402257 and 62471493, the China University Research Innovation Fund under Grant No.2023IT207, and the Talent Project of Qilu University of Technology (Shandong Academy of Sciences) under Grant No.2023RCKY141.

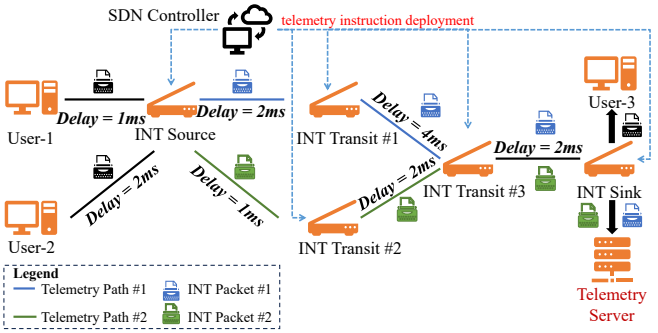

Fig. 1: In-band Network Telemetry

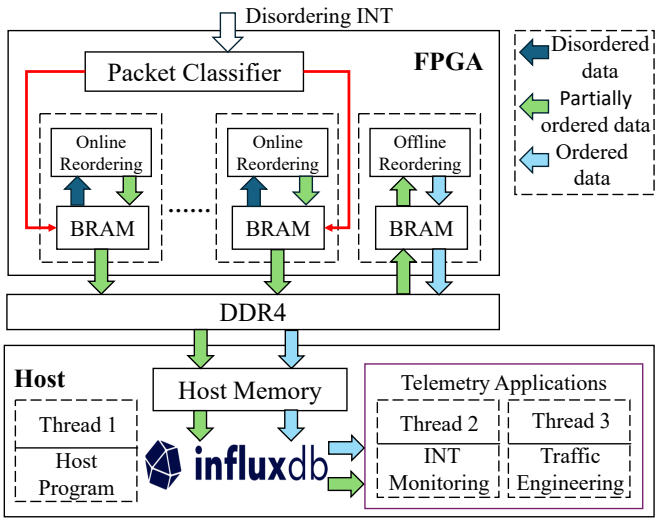

Fig. 2: The block diagram of collaboration

can easily lead to path overlaps. There is an overlap between Path#1 and Path#2 (INT Source, INT Transit #3, INT Sink). For telemetry data from the INT Source of the device, the order of arrival at the telemetry server may be $[(2\,\text{ms}, \text{INT metadata 1}), (1\,\text{ms}, \text{INT metadata 2})]$. Since the time-series database stores data in ascending order of timestamps by default, this results in the INT out-of-order problem.

TCP can ensure in-order delivery of service packets; however, it is not suitable for resolving out-of-order issues in in-band network telemetry (INT). This is because:

1) INT performs real-time, per-packet telemetry collection, where retransmitting lost packets is meaningless, as delayed packets no longer reflect the instantaneous network state.
2) TCP's ordering guarantee applies only within a single connection; in multi-source or multi-destination telemetry scenarios, cross-connection packet ordering cannot be ensured.

III. Architecture Design of In-order INT

Fig. 2 shows the CPU/FPGA collaboration architecture, consisting of a host and FPGA. The host runs the host program, which schedules reordering units and manages CPU–FPGA data transfers, and the telemetry applications, which process telemetry data for relevant network functions.

The INT packet carries telemetry data from multiple devices. In the FPGA, the packet classifier splits the packet and stores the data in BRAM by device ID. Multiple online reordering units run in parallel, each handling timestamp-based reordering for telemetry data from a single device.

The BRAM is divided into read and write sections. The read BRAM buffers telemetry data awaiting reordering, which is loaded by online and offline reordering units. After reordering, the data is stored in the write BRAM, temporarily buffered, then transferred to DDR4 and finally to the host memory.

IV. Reordering Method and Implementation Details

A. Online Reordering Architecture

To address packet reordering in networks, Hoang et al.[5] designed a sorting pipeline based on insertion reordering. However, the limitation of proposed multiple flows implement

TABLE I: Description of Parameters

Parameter	Description
T_{i}	Total number of telemetry packets in unit i
P_{i}	Insertion position in unit i
E_{ij}	Element in the j-th shift register of unit i
A_{i}	Element to be sorted in arriving packet i
R_{j}	Comparator result of comparing A_{i} with E_{ij}

is that the resource utilization of FPGA is very high. In order to reduce the resource utilization on FPGA, we propose a parallel insertion sequencer architecture based on time-multiplexed comparator, which reduces the comparator by half compared with the traditional parallel insertion reordering architecture.The traditional architecture is N registers corresponding to N comparators. Because INT collects status information from multiple devices in the network, reordering INT involves the multi-flow approach, which allocates the telemetry data from multiple sources to the different sorting unit, achieving higher throughput.It is important to note that when there are M flows in the INT, M reordering units are required, and the total number of comparators required C_{original} is

$$C_{\text{original}} = N \times M,$$

where N is the number of registers per unit.

Online reordering involves multi-flow reordering, and the method we propose allows two online reordering units to time-multiplex a shared set of comparators. In Fig. 3 the online reordering architecture for the reordering unit with a sorting window W = 16 is shown. The shown reordering units can run in parallel. And the parameters necessary for the explanation of the method are listed in Tab. I. When the telemetry packet in the network arrives at the reordering unit's arriving packet-1, the timestamp in the telemetry packet will be compared simultaneously with the timestamps of all packets in the reordering unit 1. The rules for determining the insertion

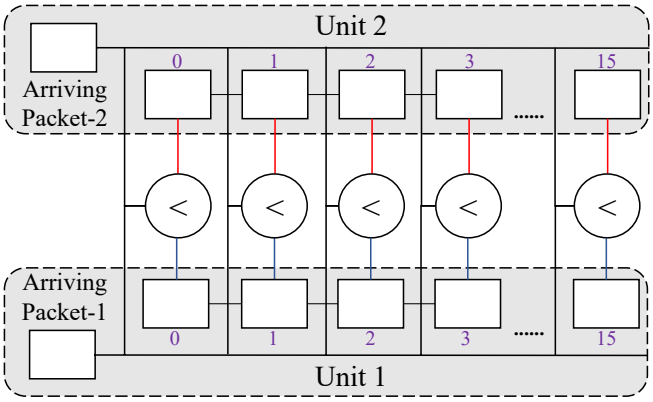

Fig. 3: The improved parallel insertion reordering framework

position P_1 are as follows.

1) If $A_1 > E_{1j}$, the output of the comparator is 0, else the output is 1.
2) If $[R_{11},\ldots,R_{1k}]$ = [0,...,0] and $[R_{1(k+1)},\ldots,R_{1\,T_1}]$ = [1,...,1], the insertion position P_1 = k.
3) When $[R_{11},R_{12},\ldots,R_{1T_1}]$ = [0,0,...,0], the insertion position $P_1 = T_1 + 1$.
4) When $[R_{11},R_{12},\ldots,R_{1T_1}]$ = [1,1,...,1], the insertion position P_1 = 1.

In the following, we discuss how the two reordering units time-multiplex a set of comparators. We mention that when the reordering unit 1 is performing right shifts followed, the comparators are in an idle cycle, which can be fully utilized by the reordering unit 2. The two reordering units in Fig. 3 respectively sort telemetry data from different network devices in the INT path.

B. Offline Reordering Architecture

After online reordering, the partially ordered telemetry data are stored in a time-series database. Merge sort is well-suited for such data, efficiently merging sub-sequences into a globally ordered sequence. Wei Song et al.[6] proposed the Parallel Merge Tree (PMT), improving performance over sequential sorters, while Papaphilippou et al.[7] introduced the Fast Lightweight Merge Sorter (FLiMS), a parallel architecture using only half the hardware resources to achieve higher FPGA throughput. Our method adopts FLiMS as the merge sorter in the parallel offline reordering architecture.

The architecture of offline reordering is shown in Fig. 4. To facilitate the explanation of the operational mechanism of the Input Manager, we treat every 32 telemetry data as a data block. After online sorting, the data within a block is ordered. However, the telemetry data across multiple blocks may be unordered. The Input Manager streams the telemetry data from a single data block into one FIFO, with the telemetry data from eight data blocks being streamed into eight separate FIFO. FLiMS, as the merge sorter in this parallel reordering architecture, merges the telemetry data from two sorted lists

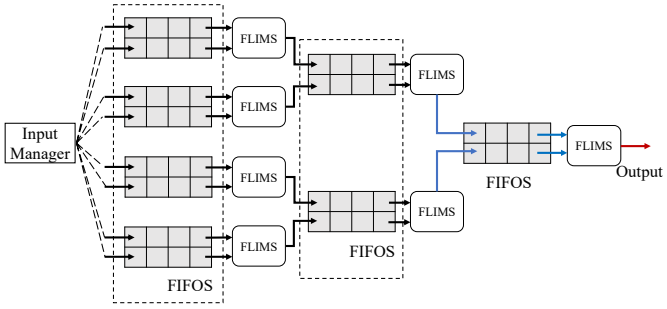

Fig. 4: The schematic diagram of offline reordering

into a single sorted queue. The implementation rules of FLiMS are as followed.

1) All FLiMS in Fig. 4 can run in parallel, provided they follow rules 2 and 3.
2) FLiMS stops functioning when any FIFO in two corresponding input FIFOs is empty.
3) FLiMS suspends if its output FIFO is full.

In the following, we describe the process of data transfer.Initially, the host transfers the data to be sorted into the FPGA's DDR4. Subsequently, the Input Manager transfers the data from DDR4 to the FIFOS in bursts[8].

V. EVALUATION

A. Experimental Setup

Our system uses the Xilinx VU37P FPGA, equipped with 32 GB of DDR4 memory (split into 2 channels) and 10 Gbps Ethernet ports. The CPU used in the experiment is Intel Xeon Gold 5318Y in host. The INT is encapsulated in UDP, and the telemetry packet size is 32 B.

B. Latency and improvement of disorder of a single online reordering unit

To quantify the disorder of a sequence, we define the disorder level D based on inversion count C and inversion distance sum S with weights α and β. For a sequence $a = [a_1, a_2, \ldots, a_N]$, an inversion pair (a_i, a_j) satisfies $i < j$ and $a_i > a_j$. The inversion distance sum is

$$S = \sum_{i=1}^{N} \sum_{j=i+1}^{N} 1(a_i > a_j) \cdot (j - i), \tag{1}$$

where $\mathbf{1}(a_i > a_j)$ is the indicator function. Let $C_{\max}$ and $S_{\max}$ be the maximum inversion count and distance sum for a sequence of length N. The disorder level is then defined as

$$D = \alpha \frac{C}{C_{\max}} + \beta \frac{S}{S_{\max}}, \quad 0 \le D \le 1, \quad \alpha + \beta = 1. \tag{2}$$

To evaluate the impact of different reordering window sizes W on disorder improvement and delivery latency during the online reordering phase, the described FPGA was used to receive 1024 INT packets containing telemetry information from a specific device. The experimental results are shown in the Fig. 5. In this experiment, the values of α and β are set to

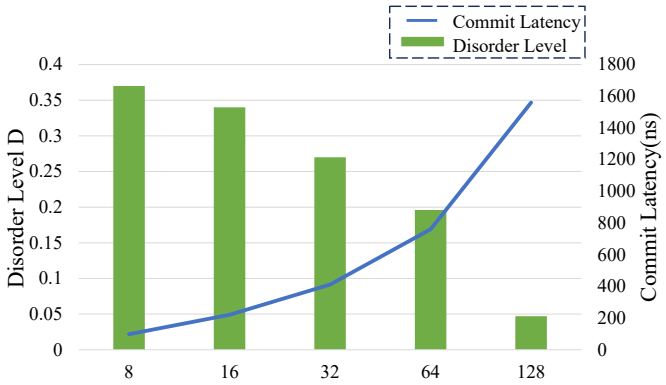

Fig. 5: The impact of different sorting window sizes on disorder improvement and delivery latency

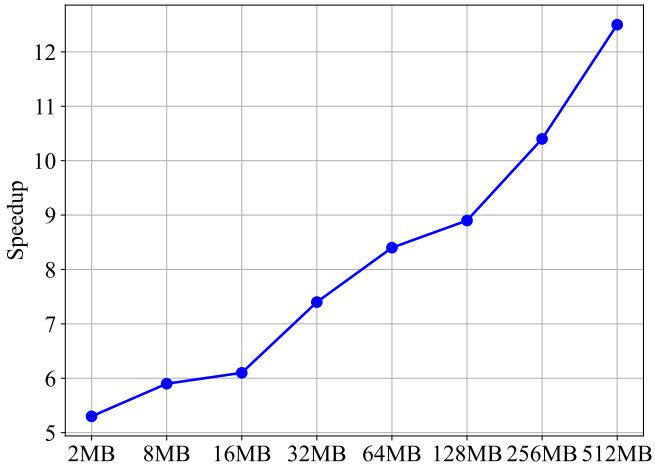

Fig. 6: The speedup of FPGA over CPU for different data sizes

0.5 and 0.5. The experimental results indicate that when the reordering window size W varies within a range of small values, increasing the window size does not significantly improve the disorder improvement, while delivery latency almost increases exponentially. When a larger sorting window is chosen, the improvement in disorder is very significant, especially when the window size is set to 128, where the disorder level is nearly eliminated, although the delivery latency is also very high, reaching 1590 ns. Therefore, the choice of reordering window size W should be selected based on the real-time application's requirements for disorder improvement and delivery latency.

C. Reordering performance

In this experiment, we evaluate overall reordering performance by processing telemetry data of 2 MB, 8 MB, 16 MB, 32 MB, 64 MB, 128 MB, 256 MB, and 512 MB from a network device in the telemetry path. The total reordering time includes both online and offline phases. For comparison, the same data are reordered on the CPU, using insertion sort for online reordering and the GCC C++ Standard Library's std::sort() for offline reordering, which serves as a standard high-performance baseline. Finally, we compute the speedup for different data sizes.

The experimental results are shown in Fig. 6.The results show that when the size of the data gradually increases, our acceleration performance gradually improves. When the data size is 32MB, the acceleration ratio is 7.4, whereas when the data size reaches 512MB, the acceleration ratio can reach 12.5. In conclusion, the data size is relatively large, the acceleration effect of our proposed CPU/FPGA-based reordering method becomes more efficient.

VI. Conclusion

In this paper, we propose a method based on a CPU-FPGA heterogeneous platform to accelerate reordering for INT. This method combines online reordering with offline reordering, meeting the real-time requirements of telemetry applications. Our experimental results prove that the proposed method can accelerate the entire reordering process of telemetry data. In the future, more efficient reordering algorithms and more adequate hardware resource utilization are important research directions for improving the performance of In-band Network Telemetry out-of-order reordering.

References

[1] L. Tan, W. Su, W. Zhang, *et al.*, "In-band Network Telemetry: A Survey," *Computer Networks*, vol. 186, p. 107 763, 2021.

[2] Y. Li, R. Miao, H. H. Liu, *et al.*, "HPCC: High precision congestion control," in *SIGCOMM'19*, ACM, 2019, pp. 44–58.

[3] T. Ouyang, H. Yao, W. He, T. Mai, F. Wang, and F. R. Yu, "Self-adaptive dynamic in-band network telemetry orchestration for balancing accuracy and stability," *IEEE Transactions on Network and Service Management*, 2025.

[4] Y.-K. Choi, J. Cong, Z. Fang, Y. Hao, G. Reinman, and P. Wei, "In-depth Analysis on Microarchitectures of Modern Heterogeneous CPU-FPGA Platforms," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 12, no. 1, pp. 1–20, 2019.

[5] V. Q. Hoang and Y. Chen, "Cost-effective Network Reordering using FPGA," *Sensors*, vol. 23, no. 2, p. 819, 2023.

[6] W. Song, D. Koch, M. Luján, and J. Garside, "Parallel Hardware Merge Sorter," in *FCCM'16*, 2016, pp. 95–102.

[7] P. Papaphilippou, C. Brooks, and W. Luk, "FLiMS: Fast Lightweight Merge Sorter," in *FPT'18*, IEEE, 2018, pp. 78–85.

[8] C. Ferry, T. Yuki, S. Derrien, and S. Rajopadhye, "Increasing FPGA Accelerators Memory Bandwidth With a Burst-Friendly Memory Layout," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 5, pp. 1546–1559, 2023.