O^2R: Performance-conscious FPGA Comparator Allocation for Online/Offline Reordering of In-Band Network Telemetry

Lizhuang Tan, Zongrui Sui
Shandong Computer Science Center (National Supercomputer Center in Jinan)
Jinan, China
tanlzh@sdas.org;zongruisui2002@gmail.com

Peiying Zhang
China University of Petroleum (East China)
Qingdao, China
zhangpeiying@upc.edu.cn

James Won-ki Hong
Pohang University of Science and Technology
Pohang, Korea
jwkhong@postech.ac.kr

Abstract

Out-of-order arrival of in-band network telemetry (INT) reports severely degrades the accuracy of network measurements and the efficiency of telemetry-driven applications. Existing FPGA-based reordering approaches either focus solely on online or offline reordering, failing to meet the concurrent requirements of real-time responsiveness and large-scale data processing. In this paper, we present O^2R, a performance-conscious comparator allocation framework that enables simultaneous online and offline reordering on a single FPGA. O^2R dynamically adjusts the weighting of out-of-order metrics, allocates comparator resources based on online/offline demand, and performs proportional fine-tuning to prevent long-term performance drift. Experimental results demonstrate that O^2R reduces online reordering latency by 11.1%–20.5% and improves the quality of partial ordering by 53.5%–73.6% compared with baseline, thereby significantly enhancing the responsiveness and accuracy of INT-based telemetry systems.

CCS Concepts

• **Networks → Network measurement**.

Keywords

Network Management, Network Measurement, In-band Network Telemetry, Programmable Data Plane

ACM Reference Format:
Lizhuang Tan, Zongrui Sui, Peiying Zhang, and James Won-ki Hong. 2025. O^2R: Performance-conscious FPGA Comparator Allocation for Online/Offline Reordering of In-Band Network Telemetry. In *Proceedings of ACM CoNEXT Workshop on In-Network Computing and AI for Distributed Systems (INCAS 2025).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3728424.XXXXXXX

1 Introduction

In-band Network Telemetry (INT)[8] has emerged as a promising network measurement paradigm that embeds telemetry instructions and metadata within live data-plane traffic. This design enables fine-grained, end-to-end, hop-by-hop visibility of the network state without the need for additional probe traffic[11].

In modern large-scale networks, telemetry tasks are often orchestrated through multipath[9] and multitask[2, 3] strategies, where telemetry packets traverse multiple paths concurrently. However, variations in path latency, congestion, and other network dynamics can cause telemetry packets to arrive out of order at the receiver, even when they originate from the same flow. Such packet reordering can be severe in practice[6].

The consequences of reordering are twofold. First, it undermines temporal accuracy in network state analysis, which is crucial for applications relying on precise timing information. Second, it can lead to data loss or statistical distortion when telemetry data are aggregated or correlated across flows. More specifically, INT-based network management applications can be broadly categorized into online and offline applications. Online applications place stringent requirements on the freshness of telemetry data, as exemplified by traffic engineering[5] and congestion control[4]. Offline applications are relatively less sensitive to data freshness, such as historical data analytics[10] and gray failure diagnosis[1]. Nevertheless, out-of-order telemetry data adversely affects both categories of applications, thereby degrading their functionality and performance.

Field-Programmable Gate Arrays (FPGAs), owing to their highly parallel and customizable hardware architecture, have been widely adopted in high-performance data processing scenarios. Leveraging FPGAs for reordering out-of-order telemetry data can fully exploit their parallel processing capability and low-latency advantages. However, existing approaches typically implement either online or offline reordering in isolation and lack designs specifically tailored to the multi-flow and multi-task characteristics of in-band network telemetry[7]. As a result, they struggle to simultaneously meet the stringent real-time requirements and large-scale data processing demands of practical deployments.

Within an FPGA, the comparator unit (CU) is the key hardware component that enables efficient reordering and reordering. This hardware primitive allows FPGAs to significantly outperform general-purpose CPUs and GPUs in INT packet reordering tasks. The number of available comparator units directly determines the degree of parallelism and resource utilization achievable on the FPGA. Consequently, a pressing challenge for current INT systems is how to efficiently reorganize out-of-order telemetry reports under

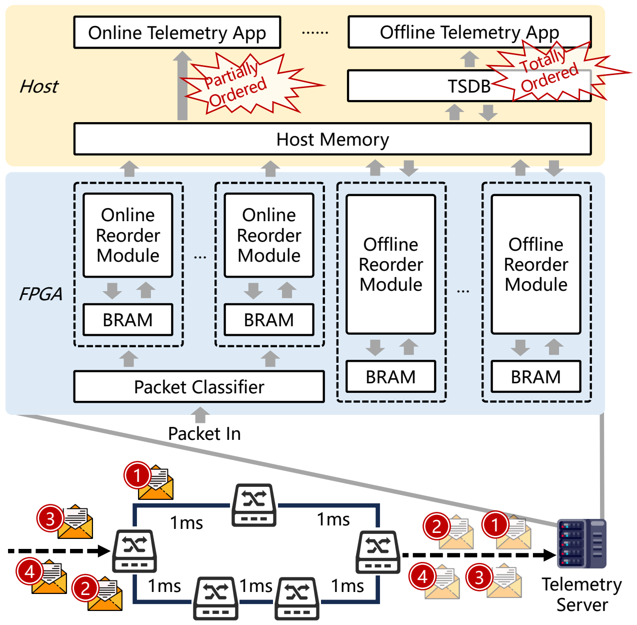

Figure 1: The framework overview of O^2R.

the constraints of limited FPGA comparator resources, in a way that supports both online and offline reordering simultaneously, thereby providing ordered data to upper-layer telemetry applications.

In this paper, we take the first step toward addressing this challenge by proposing a comparator resource allocation method, O^2R, that enables the simultaneous deployment of online and offline reordering on a single FPGA accelerator card. O^2R balances the requirements of high-speed online real-time telemetry data analysis and large-scale offline telemetry data processing, thereby improving FPGA resource utilization and telemetry processing performance.

2 O^2R

As shown in Fig. 1, due to path differences, the order of packets participating in in-band network telemetry upon arrival at the telemetry server does not match their transmission order, resulting in out-of-order measurement results for intermediate overlapping switches.

O^2R partitions the telemetry server into a host-side and an FPGA-side. The host-side handles telemetry applications and data management, whereas the FPGA-side is dedicated to reordering out-of-order telemetry results. The FPGA card is deployed with several online reorder (OnReorder) modules and offline reorder (OffReorder) modules. The telemetry results produced by the OnReorder module remain partially ordered and are consumed by online telemetry applications. A portion of the partially ordered telemetry results is stored in the time-series database (TSDB) and subsequently fully reordered by the OffReorder module, providing totally ordered data for offline telemetry applications.

2.1 Problem Formulation

The FPGA comparator allocation problem for in-band network telemetry out-of-order reordering can be formally described as follows.

Let the total number of available comparator units be denoted by m. We allocate n_1 comparators to the online reordering task and n_2 comparators to the offline reordering task. This can be expressed as a multi-objective optimization problem:

$$\max_{n_1,n_2} \quad \Big(-L_{\text{on}}(n_1),\ T_{\text{off}}(n_2) \Big), \qquad \text{s.t.} \quad n_1 + n_2 \le m, \tag{1}$$

where L_{on} represents the latency of the online reordering, and T_{off} denotes the throughput of the offline reordering.

2.2 Quantification of Out-of-order

To quantify the degree of out-of-order, O^2R adopts the following three metrics:

(1) Local Window Out-of-order Ratio (LWO). Within an online reordering observation window, the LWO is defined as the proportion of out-of-order packets to the total number of packets:

$$\text{LWO} = \frac{N_{\text{dis}}}{N}, \tag{2}$$

$$N_{\text{dis}} = \sum_{i=1}^{N} \mathbb{1}\left(p_i \ne p_i^{\text{ord}}\right), \tag{3}$$

where $\mathbb{1}(\cdot)$ is the indicator function, which takes the value 1 if the condition $p_i \ne p_i^{\text{ord}}$ holds, and 0 otherwise. Here, N denotes the total number of packets observed within the online reordering window; N_{dis} denotes the number of packets that are out of order within the window; p_i represents the position index of the i-th packet in the actual arrival sequence; and p_i^{ord} represents the expected position index of the i-th packet in the ideally ordered sequence.

(2) Cross-Window Out-of-order Ratio (CWO). The CWO is defined as the average proportion of cross-window out-of-order packets between two consecutive online reordering windows:

$$\text{CWO} = \frac{1}{K-1} \sum_{k=1}^{K-1} \frac{N_{k\to k+1} + N_{k+1\to k}}{N^{(k)} + N^{(k+1)}}, \tag{4}$$

where $N_{k\to k+1}$ denotes the number of packets whose ideal positions belong to window k but are actually observed in window $k+1$, and $N^{(k)}$ and $N^{(k+1)}$ denote the total number of packets within windows k and $k+1$, respectively. Here, K represents the total number of online reordering windows under observation.

(3) Weighted Out-of-order Distance (WOD). The WOD is defined as the average displacement of each out-of-order packet from its correct position in the ideally ordered sequence:

$$\text{WOD} = \frac{1}{N_{\text{dis}}} \sum_{i=1}^{N} \mathbb{1}\left(p_i \ne p_i^{\text{ord}}\right) \cdot \left| p_i - p_i^{\text{ord}} \right|. \tag{5}$$

In summary, LWO measures the short-term out-of-order degree within a single online window, CWO captures the global out-of-order degree across consecutive windows, and WOD quantifies the displacement distance of out-of-order packets. All these metrics are continuously updated in real time on the FPGA data plane using a hardware pipeline.

2.3 System Model

Based on Eq. 1, we define the overall system utility of O^2R as

$$U(L_{\text{on}}, T_{\text{off}}) = \lambda_1 \cdot f_{\text{on}}(L_{\text{on}}) + \lambda_2 \cdot f_{\text{off}}(T_{\text{off}}), \tag{6}$$

where $f_{\text{on}}(\cdot)$ and $f_{\text{off}}(\cdot)$ are utility functions that map the online reordering latency L_{on} and offline reordering throughput T_{off} to normalized utility values, respectively. The parameters λ_1 and λ_2 are weighting factors that balance the contributions of online and offline objectives.

The two utility component functions are defined as follows:

$$f_{\text{on}}(L_{\text{on}}) = \frac{L_{\text{target}}}{L_{\text{on}} + L_{\text{target}}}, \tag{7}$$

$$f_{\text{off}}(T_{\text{off}}) = \min\left(\frac{T_{\text{off}}}{T_{\text{target}}}, 1\right), \tag{8}$$

where L_{target} denotes the target online reordering latency and T_{target} denotes the target offline reordering throughput. The function $f_{\text{on}}(L_{\text{on}})$ monotonically decreases with latency, thus rewarding lower online latency, while $f_{\text{off}}(T_{\text{off}})$ increases with throughput and is capped at 1 once the throughput target is achieved.

Then, we define the resource demand evaluation functions for online and offline reordering, denoted by D_{on} and D_{off}, respectively, as follows:

$$D_{\text{on}} = w_1 \cdot \text{LWO} + w_2 \cdot \text{WOD}', \tag{9}$$

$$D_{\text{off}} = w_3 \cdot \text{CWO}, \tag{10}$$

where $\text{WOD}' = \min(\text{WOD}/\text{WOD}_{\max}, 1)$. Since the value range of the WOD depends on the scale of telemetry packets, we adopt the normalized weighted disorder distance WOD′ as the metric for quantifying the severity of reordering. The parameters w_1, w_2, and w_3 are weighting coefficients. $D_{\text{on}} \in [0, 1]$ and $D_{\text{off}} \in [0, 1]$ represent the normalized comparator resource demand intensities for online and offline reordering, respectively.

Furthermore, we define the resource allocation ratio for online reordering, denoted by R_{on}, as

$$R_{\text{on}} = \frac{D_{\text{on}} + \epsilon}{D_{\text{on}} + D_{\text{off}} + 2\epsilon}, \tag{11}$$

where $\epsilon > 0$ is a smoothing factor introduced to ensure the stability of the allocation.

Next, we define the expected performance of online and offline reordering under a given number of comparators. For online reordering, the predicted processing latency is

$$L_{\text{pre}}(n_1) = L_{\text{base}} + \frac{w}{n_1 \cdot f_{\text{clk}}}, \tag{12}$$

where L_{base} denotes the base latency including data reception and preprocessing time, w is the online reordering window size, n_1 is the number of comparators allocated to online reordering, and f_{clk} is the FPGA clock frequency. For each arriving packet, the parallel comparators quickly insert it into the correct position. The number of comparators determines the parallel processing capability of online reordering under multiple telemetry flows. Consequently, the processing latency is proportional to the window size and inversely proportional to the number of comparators.

Offline reordering can be regarded as a conventional parallel comparison process. Given the size of data to be reordered and the available offline comparators, the predicted reordering throughput can be expressed as

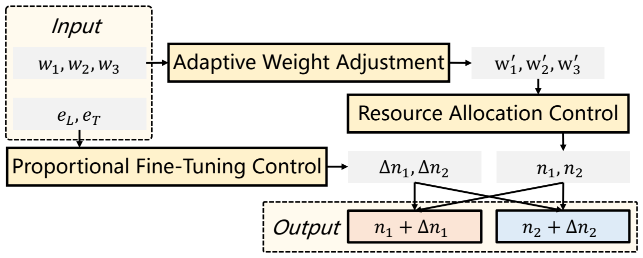

Figure 2: The algorithm procedure of O^2R.

$$T_{\text{pre}}(n_2) = \frac{n_2 \cdot f_{\text{clk}}}{\log_2(N \cdot K)}, \tag{13}$$

where $N \cdot K$ is the data size, $\log_2(N \cdot K)$ represents the number of merge levels in a merge-sort-based algorithm, and n_2 is the number of comparators allocated to offline reordering. The number of comparators determines the parallel processing capability at each merge level. Thus, the throughput is proportional to the number of comparators and inversely proportional to the logarithm of the data size.

2.4 Performance-conscious Allocation

We propose a performance-feedback-based FPGA comparator resource allocation algorithm, which consists of three key components, as shown in Fig. 2:

(1) **Adaptive Weight Adjustment:** This component updates the weights of the out-of-order metrics (LWO, CWO, and WOD) used in the resource allocation decision process by monitoring the deviation between actual system performance and target performance. In doing so, it enables the allocation strategy to adapt to dynamic network conditions while maintaining system stability, as shown in Alg. 1, where $\beta \in [0, 1]$ is a smoothing factor used to prevent rapid changes in the weights.

(2) **Resource Allocation Control:** Serving as the main control loop of the comparator resource allocation process, this component determines the resource distribution by leveraging the weights obtained from the adaptive weight adjustment component, as shown in Alg. 2.

(3) **Proportional Fine-Tuning Control:** Built upon the resource allocation control component, this component performs fine-grained adjustments to correct residual performance deviations, preventing long-term performance drift and avoiding oscillations in the comparator resource allocation process, as shown in Alg. 3. In this algorithm, u_L represents the suggested resource adjustment for online reordering in the latency dimension. If $u_L > 0$, the actual latency is lower than the predicted target, and thus the comparator resources allocated to online reordering can be slightly reduced. Conversely, if $u_L < 0$, the actual latency exceeds the predicted target, requiring an increase in comparator resources for online reordering. Similarly, u_T represents the suggested resource adjustment for offline reordering in the throughput dimension. If $u_T > 0$, the actual throughput is higher than the predicted target, and thus the comparator resources allocated to offline reordering can be slightly reduced. If $u_T < 0$, the actual throughput is lower

Algorithm 1: Adaptive Weight Adjustment

```
Input      : LWR, CWR, WDD; $L_{\text{actual}}$, $T_{\text{actual}}$; $L_{\text{target}}$, $T_{\text{target}}$;
             $w_1^{\text{prev}}$, $w_2^{\text{prev}}$, $w_3^{\text{prev}}$
Output     : $w_1'$, $w_2'$, $w_3'$
Parameters : $\theta_L, \theta_T, \alpha_1, \alpha_2, \alpha_3, \beta, w_{\min}, w_{\max}$
1  $E_L \leftarrow \frac{|L_{\text{actual}} - L_{\text{target}}|}{L_{\text{target}}}, \quad E_T \leftarrow \frac{|T_{\text{actual}} - T_{\text{target}}|}{T_{\text{target}}}$
2  if $E_L > \theta_L$ then
3      $\Delta w_1 \leftarrow \alpha_1 \cdot E_L, \quad \Delta w_2 \leftarrow \alpha_2 \cdot E_L$
4  else
5      $\Delta w_1, \Delta w_2 \leftarrow 0$
6  if $E_T > \theta_T$ then
7      $\Delta w_3 \leftarrow \alpha_3 \cdot E_T$
8  else
9      $\Delta w_3 \leftarrow 0$
10 for $i \leftarrow 1$ to 3 do
11     $w_i \leftarrow w_i^{\text{prev}} + \beta \cdot \Delta w_i$
12 $w_i' \leftarrow \frac{w_i}{\sum_{j=1}^{3} w_j}, \quad i = 1, 2, 3$
13 for $i \leftarrow 1$ to 3 do
14     $w_i' \leftarrow \min(\max(w_i', w_{\min}), w_{\max})$
15 return $w_1'$, $w_2'$, $w_3'$
```

than the predicted target, necessitating an increase in comparator resources for offline reordering. And K_p is the proportional gain, α is the maximum adjustment ratio, and floor$(\cdot)$ denotes the floor (round-down) function.

3 Evaluation Results

We complete the verification on Xilinx VU37P FPGA, which is equipped with a 10GE port. We generate between 10 and 50 concurrent INT flows. Each flow is randomly distributed across three sub-paths, with the path delay of the i-th sub-path modeled as $d_i \sim \mathcal{N}(\mu_i, (2\text{ ms})^2)$ where $\mu_i \in \{7, 10, 13\}$ ms. This setup induces packet reordering at the telemetry server. For offline reordering, we prepare a fixed batch of 512/1024/2048 MB of telemetry data as the workload to stress the offline reordering.

As shown in Fig. 3, The O^2R achieves a 11.1%–20.5% reduction in online reordering latency compared with baseline[7]. O^2R improves the quality of the online reordering result, achieving a 53.5%–73.6% reduction in the local disorder rate.

4 Conclusion

This paper introduces O^2R, the performance-conscious FPGA comparator allocation framework for INT reordering. Future work will explore adaptive window sizing, integration with multipath-aware schedulers, and extending O^2R to heterogeneous accelerators for improved scalability and energy efficiency.

Acknowledgments

This work was supported by National Key R&D Program of China under Grant No.2024YFB2907000, Shandong Provincial Natural Science Foundation under Grant No. ZR2022LZH015 and ZR2023QF025, National Natural Science Foundation of China under Grant 62402257, and Institute of Information & Communications Technology Planning & Evaluation (IITP) (RS-2024-00392332).

Algorithm 2: Resource Allocation Control

```
Input      : $w_1'$, $w_2'$, $w_3'$, $m$
Output     : $n_1$, $n_2$
Parameters : $\epsilon$
1  Update $D_{\text{on}}$ and $D_{\text{off}}$ according to Eqs. 9 and 10
2  Update $R_{\text{on}}$ according to Eq. 11
3  $n_1 \leftarrow \lfloor m \cdot R_{\text{on}} \rfloor$
4  $n_2 \leftarrow m - n_1$
5  return $n_1$, $n_2$
```

Algorithm 3: Proportional Fine-Tuning Control

```
Input      : $n_1$, $n_2$; $L_{\text{actual}}$, $T_{\text{actual}}$
Output     : Updated $n_1$, $n_2$
Parameters : $K_p$, $\alpha$
1  $e_L \leftarrow L_{\text{pre}}(n_1) - L_{\text{actual}}, \quad e_T \leftarrow T_{\text{pre}}(n_2) - T_{\text{actual}}$
2  $u_L \leftarrow K_p \cdot e_L, \quad u_T \leftarrow K_p \cdot e_T$
3  $\Delta n_1 \leftarrow \text{floor}(u_L \cdot n_1 \cdot \alpha), \quad \Delta n_2 \leftarrow \text{floor}(u_T \cdot n_2 \cdot \alpha)$
4  return $\Delta n_1$, $\Delta n_2$
```

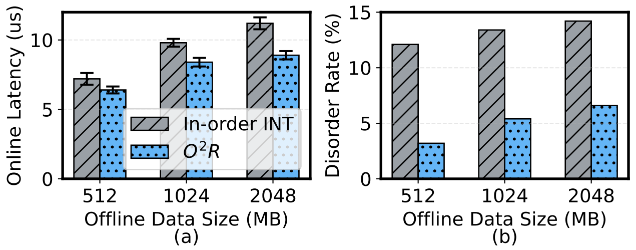

Figure 3: The improvement result of O^2R.

References

[1] Hongyang Chen, Benran Wang, Guangba Yu, Zilong He, Pengfei Chen, Chen Sun, and Zibin Zheng. 2025. NetScope: Fault Localization in Programmable Networking Systems With Low-Cost In-Band Network Telemetry and In-Network Detection. *IEEE Transactions on Networking* (2025), 1–17.

[2] Wei Chen, Ye Tian, Zhongxiang Wei, Jiangyu Pan, and Xinming Zhang. 2022. Task scheduling for probabilistic in-band network telemetry. *IEEE/ACM Transactions on Networking* 30, 6 (2022), 2858–2869.

[3] Fuliang Li, Qianchen Yuan, Yuhua Lai, Zhenbei Guo, Elliott Wen, Tian Pan, Xingwei Wang, and Jiannong Cao. 2025. INT-Source: Topology-Adaptive In-Band Network-Wide Telemetry. *IEEE Transactions on Networking* (2025), 1–15.

[4] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, et al. 2019. HPCC: High precision congestion control. In *SIGCOMM'19*. ACM, 44–58.

[5] Jonatas A Marques and Luciano Paschoal Gaspary. 2023. Advancing Network Monitoring and Operation with In-band Network Telemetry and Data Plane Programmability. In *NOMS'23*. IEEE, 1–6.

[6] Mimi Qian, Lin Cui, Fung Po Tso, Yuhui Deng, and Weijia Jia. 2023. OffsetINT: Achieving high accuracy and low bandwidth for in-band network telemetry. *IEEE Transactions on Services Computing* 17, 3 (2023), 1072–1083.

[7] Zongrui Sui, Lizhuang Tan, Huiling Shi, Wei Zhang, and Peiying Zhang. 2025. In-Order INT: Efficient Reordering of Out-of-Order In-Band Network Telemetry Packets via CPU-FPGA Co-Design. In *APNOMS'25*. IEEE, 1–4.

[8] Lizhuang Tan, Wei Su, Wei Zhang, Jianhui Lv, Zhenyi Zhang, Jingying Miao, Xiaoxi Liu, and Na Li. 2021. In-band network telemetry: A survey. *Computer Networks* 186 (2021), 107763.

[9] Shaofei Tang, Sicheng Zhao, Xiaoqin Pan, and Zuqing Zhu. 2022. How to use in-band network telemetry wisely: Network-wise orchestration of Sel-INT. *IEEE/ACM Transactions on Networking* 31, 1 (2022), 421–435.

[10] 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 *APNET'24*. ACM, 115–121.

[11] Qianchen Yuan, Fuliang Li, Tian Pan, Yuhua Lai, Yetao Gu, Xingwei Wang, and Jiannong Cao. 2025. INT-Partition: Hierarchical and Fault-Tolerant In-Band Network Telemetry. *IEEE Transactions on Networking* (2025), 1–15.