Enabling Network Diagnostics in Time-Sensitive Networking: Protocol, Algorithm, and Hardware

Zeyu Wang[*§], Xiaowu He[*§], Xiangwen Zhuge[*], Shen Xu[*], Fan Dang[†], Jingao Xu[*] and Zheng Yang[*✉]
[*] School of Software and BNRist, Tsinghua University [†] Global Innovation Exchange, Tsinghua University
{ycdfwzy, horacehxw, zgxw18}@gmail.com, xushen20@mails.tsinghua.edu.cn,
dangfan@tsinghua.edu.cn, {xujingao13, hmilyyz}@gmail.com

***Abstract*—Time-Sensitive Networking (TSN) is foreseen as a foundational technology that enables Industry 4.0. It offers deterministic data transmission over Ethernet for critical applications such as industrial control and automotive systems. However, TSN is susceptible to hardware and software errors, necessitating an effective diagnostic system. Traditional network diagnostic tools are inadequate for TSN fault localization due to the unique characteristics of TSN. In response, this paper presents TSNCard, a cross-cycle postcard-based diagnostic system tailored for TSN. TSNCard introduces a novel telemetry protocol that leverages the cyclical nature of TSN networks for data collection at each node. This protocol, coupled with dedicated analytic algorithms and hardware innovations within switches, forms a comprehensive system for TSN monitoring and fault localization. Extensive experiments on both simulation and physical testbeds show that TSNCard can 100% localize the root cause of the TSN misbehavior while adhering to industrial bandwidth restrictions. TSNCard not only bridges the gap in the TSN protocol stack, but also serves as a versatile toolkit for time-synchronized network analysis, paving the way for future research.**

***Index Terms*—Time-Sensitive Networking, Network Diagnostics, Industrial Networking**

I. Introduction

Time-Sensitive Networking (TSN) is recognized as a foundational technology towards Industry 4.0 [1]. It enables deterministic data transmission of time-sensitive traffic alongside best-effort traffic in a time-synchronized Ethernet network, catering to time-critical applications like industrial control and automotive. To achieve this, TSN operates on the principle of time-division multiplexing. A centralized scheduler calculates the precise forwarding time of all critical flows at each hop, preventing any potential conflicts [2].

Despite its precision, TSN is susceptible to hardware and software errors similar to traditional network devices, resulting in time synchronization errors, incorrect forwarding time, or packet disorder. In industrial scenarios where TSN is essential, swiftly localizing and rectifying such failures is imperative to maintain the stable operation of production environments. However, current TSN standards typically rely on a static configuration issued by a central controller, lacking a closed-loop control system for traffic monitoring and analysis. Existing preliminary efforts can only detect the misbehavior without fault localization, and they remain largely theoretical without practical implementation [3], [4].

[✉] Zheng Yang is the corresponding author.
[§] Zeyu Wang and Xiaowu He are co-primary authors.

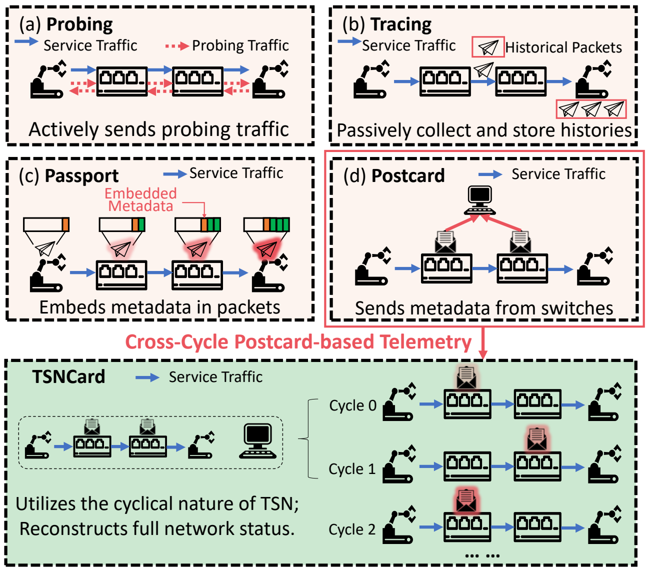

(a) Probing
Service Traffic
Probing Traffic
Actively sends probing traffic
(b) Tracing
Service Traffic
Historical Packets
Passively collect and store histories
(c) Passport
Service Traffic
Embedded Metadata
Embeds metadata in packets
(d) Postcard
Service Traffic
Sends metadata from switches
Cross-Cycle Postcard-based Telemetry
TSNCard
Service Traffic
Cycle 0
Cycle 1
Cycle 2
Utilizes the cyclical nature of TSN;
Reconstructs full network status.

Figure 1. A comparison of network diagnostic tools.

In traditional Ethernet, especially data center networks, there are an abundance of methods for traffic analysis and fault diagnostics. As shown in Fig. 1, they can be broadly categorized into *probing* [5]–[8], *tracing* [9]–[11], *passport-based telemetry* [12]–[14], and *postcard-based telemetry* [15]–[17]. These methods typically encompass three major stages to perform network diagnostics: (i) traffic measurement stage that monitors and records real-time statistics for target data streams; (ii) data collection stage that stores or collects the statistics necessary for diagnostics; and (iii) subsequent issue analytics stage that diagnoses the network issues based on collected information.

Nevertheless, different from traditional Ethernet's best-effort approach, TSN necessitates predetermined schedule for the critical traffic transmission. It reserves dedicated bandwidth and nanosecond-level precise time slots for critical data frames. This requirement significantly challenges conventional diagnostic methods across their traffic measurement, data collection, and issue analytics stages, as further explored below:

- **Chain of Errors in Issue Analytics Stage.** TSN's time-division multiplexing principle results in tight interdependence among data stream schedules. This tight coupling means that any variance in one stream's transmission can adversely affect others, leading to a chain of errors. The analytics algorithms in traditional methods are not designed for such fine-grained

Table I
PERFORMANCE OF EXISTING METHODS AND TSNCARD ADDRESSING THE CHALLENGES IN TSN DIAGNOSTICS.

	Tracing	*Probing*	*Passport*	*Traditional Postcard*	***TSNCard***
Error Localization	N/A	✗	✗	✗	✓
Bandwidth Cost	N/A	14Mbps	15Mbps	59Mbps	**1Mbps**
Measurement Acc.	N/A	$92\mu s$	$178\mu s$	$163\mu s$	$\mathbf{<0.1}\mu s$

forwarding time discrepancies, thus fall short in identifying the root cause of these faults (§II-B-C1).

• **Bandwidth Constraint for Data Collection Stage.** In industrial networks, the majority of the bandwidth is reserved for high-priority business traffic, limiting room available for network monitoring and diagnostics. This constraints traditional tools, designed for environments with abundant bandwidth, to collect sufficient data for comprehensive analysis (§II-B-C2).

• **Time Inaccuracy during Traffic Measurement Stage.** TSN requires data to be transmitted synchronously with nanosecond-level precision. This requires diagnostic tools to precisely measure critical traffic to notice the discrepancy between actual transmission time and the predetermined schedule. However, existing tools generally lack the capability to measure transmission time from hardware logic or access the synchronized clock information, thus are inadequate for TSN misbehavior identification (§II-B-C3).

Table I demonstrates some preliminary results in our case study (details in §II-B). We find that none of existing methods can perform well in these three stages of fault diagnostics. Therefore, a brand new traffic analysis paradigm is required to tackle TSN's unique requirements. This poses many challenges that are addressed in this work: (i) how to design a cooperation mechanism between switches and the analytic server to gather adequate diagnostic information under various constraints; (ii) how to ensure swift and precise root cause localization in complicated error scenarios; and (iii) how to achieve transmission time measurement with high precision and low overhead.

In this paper, we present **TSNCard**, a cross-cycle post**Card**-based diagnostic system for **TSN**. TSNCard is the first system for comprehensive TSN traffic monitoring and fault localization. Overall, TSNCard's design and implementation excel across three layers. To be specific:

• **On the protocol front:** we introduce a cross-cycle postcard-based telemetry protocol. This protocol utilizes the cyclical nature of TSN, collecting data frame transmission information across cycles at each hop. It enables the analytic server to reconstruct the necessary state of critical flows and comprehensively understand the network performance (§IV-A).

• **On the algorithm front:** we design two algorithms on the analytic server. First, a flow-centric fault refinement algorithm is proposed to localize the root cause of network problems based on the deviation between data frames' scheduled and actual forwarding time. Second, to save analysis overhead, we introduce a bandwidth-adaptive postcard prioritization algorithm to adjust the amount of postcards to collect according to the application requirements (§IV-B).

• **On the hardware front:** we propose a suite of hardware technologies within TSN switches. They enable the measurement of high-precision synchronized transmission time without disrupting the flow of critical data frames, supporting the execution of upper-layer algorithms and protocols integral to TSNCard's functionality (§IV-C).

We implement TSNCard based on the Xilinx Zynq platform [18] with software and hardware co-design. Comprehensive experiments are carried out on both simulation and physical testbeds with more than 24,000 test cases. The experiment results demonstrate that TSNCard can 100% localize the root cause of the time-related TSN misbehavior in any circumstance, operates smoothly under 1Mbps bandwidth limitation, and takes only about $10ms$ to complete diagnostic data collection.

In summary, this paper makes three contributions.
(1) We design and implement TSNCard, as far as we are aware of, the first comprehensive TSN traffic diagnostic system that can pinpoint the root cause of forwarding misbehavior. It effectively resolves the gap in existing TSN standards, enhancing its capabilities for closed-loop traffic analysis.
(2) We propose the cross-cycle postcard-based telemetry protocol, along with an array of software and hardware innovations, to systematically realize comprehensive, efficient, and low-overhead fault localization in TSN.
(3) We extensively evaluate the performance of TSNCard and comparative systems on simulation and physical testbeds. The results demonstrate TSNCard's superior performance.

Contribution to the community. We make TSNCard's prototype implementation publicly available[1]. TSNCard can serve as a toolkit to effectively analyze any network traffic precisely in an IEEE 802.1AS [19] compatible time-synchronized network, benefiting not only future research, but also the industry for network monitoring, traffic analysis, etc.

II. BACKGROUND AND MOTIVATION

A. TSN Scheduling and Forwarding Misbehavior

TSN uses a well-designed global schedule to reserve dedicated bandwidth for critical traffic. Fig. 2(a) and (b) demonstrate a simple network topology and the predetermined TSN schedule. As seen, the schedule dictates the exact transmission times of data packets on each network link. For example, the packet in flow f_1 is supposed to leave the output port of the device DE1, switch SW1, and switch SW2 at timestamps 0, 1, and 5, respectively. Following the schedule, the TSN switches configure their egress ports' gate control lists to regulate when each packet can depart from their egress ports, thus ensuring the smooth operation of a TSN network. Nevertheless, this well-designed schedule also results in strong interdependence among critical flows, introducing unique challenges.

We define TSN forwarding misbehavior as instances in which the actual transmission times of the data packets deviate from the planned schedule. These deviations may manifest as packets that leave a switch earlier or later than scheduled or, in more severe cases, packets being lost entirely. For instance, as

[1]https://github.com/MobiSense/TSNCard

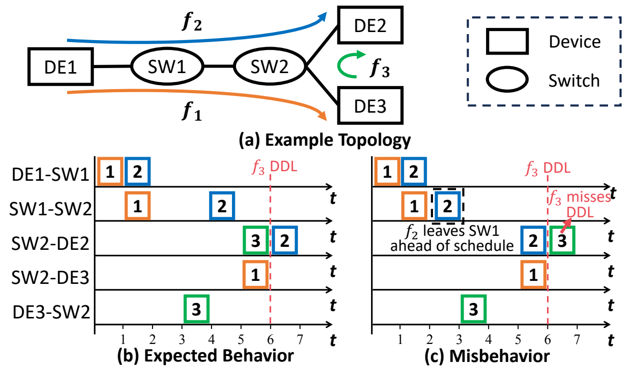


Figure 2. Forwarding misbehavior and chain of errors.

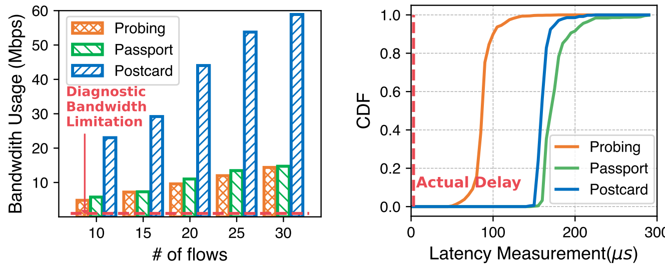
depicted in Fig. 2(c), the packet of flow f_2 leaves switch SW1 ahead of schedule while the packet of flow f_3 leaves switch SW2 with a delay. Such deviations can lead to end-to-end transmission delays that exceed the application requirements, causing errors in critical industrial processes.

B. Limitation of Current Practice

To study the effectiveness of existing Ethernet diagnostic systems, we deploy a testbed with TSN switches and evaluation toolkits (setup detailed in §V). As mentioned in §I, *tracing* methods are not applicable in our scenario. Therefore, we implement *probing*, *passport*, and *traditional postcard* methods to tackle the TSN fault localization problem. The overall results are demonstrated in Table I. As seen, neither of the existing methods could successfully localize the root cause of TSN misbehavior. We dig deep into the underlying reasons and find that the challenges are three-fold:

C1: Chain of Errors. In TSN, packets are carefully scheduled for transmission over shared network links, creating complex interdependence among data streams. This time-division multiplexing setup ensures deterministic transmission for critical industrial traffic. However, it also implies that a delay or misalignment for one packet can disrupt the transmission of subsequent streams, leading to a chain of errors from the original malfunctioning device. As a result, when the end-to-end delay of a data stream exceeds scheduled deadline, numerous switches may already be experiencing forwarding misbehaviors, making it difficult to pinpoint the root cause.

Fig. 2 demonstrates a typical chain-of-errors problem we encountered. As the root cause of misbehavior, the switch SW1 in Fig. 2(a) encounters an internal hardware logic malfunction. This affects the packet departure time of flow f_2. As a result, the packet from flow f_2 arrives at switch SW2 earlier than the schedule, subsequently delaying the transmission of flow f_3. From an operational point of view, only flow f_3 appears to miss its deadline, while flow f_1 and f_2's end-to-end latency is acceptable. However, if the network operators or diagnostic tools only inspect the switches and devices along the path of flow f_3, they will never find the root cause of the issue, i.e., switch SW1. According to our investigation, all existing tools lack the granularity necessary to identify the forwarding misbehavior at each switch. Therefore, they are unable to effectively trace the error back to its origin.



(a) Limited available bandwidth. (b) Inaccurate time measurement.

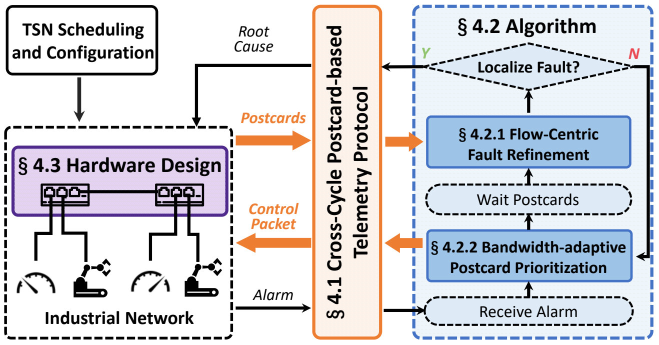
Figure 3. Limitation of current practice.

C2: Limited Available Bandwidth. In a typical industrial production network, most of the bandwidths are allocated to high-priority traffic, e.g., periodic control and raw sensor data, leaving limited bandwidth for traffic monitoring and diagnostics. According to Li et al. [20] and our field study in typical manufacturing factories, in a typical 100Mbps industrial network, there are only about 1Mbps free bandwidths left for network analysis. However, traditional Ethernet diagnostic methods, designed for data centers with abundant bandwidth, rely on continuous generation, collection, and storage of historical telemetry data. Therefore, these methods may not be effective under the constrained bandwidth available in TSN.

We evaluate the existing methods in our testbed for traffic analysis. Specifically, we remove background traffic (video, sensor data, etc.), leaving only periodic control traffic in the network. Then, we apply each of these methods to continuously collect analytic information of these critical signals, and record their bandwidth utilization. As shown in Fig. 3a, the bandwidth consumption of these methods increases linearly with the number of critical flows. When the number of flows exceeds 10, all of their bandwidth consumption exceeds the 1Mbps limit, leading to potential traffic congestion and loss of analytic information.

C3: Inaccurate Time Measurement. TSN critical data frames are required to be transmitted with nanosecond-level precision, adhering to the predetermined schedule. Therefore, to effectively identify potential forwarding misbehavior in TSN, it is essential to acquire high-precision packet-level transmission time at each hop. Existing methods either only have end-to-end delays or Round-Trip Time (RTT), lacking the necessary hop-level fine granularity (e.g., *tracing*, *probing*), or rely on software-based time synchronization that provides insufficient measurement accuracy (e.g., *passport*, *traditional postcard*). Furthermore, since each critical data stream conforms to the strict timetable, embedding analytic information into packets may affect their transmission time, leading to discrepancies between observed results and actual performance.

We set up the testbed with a switch connecting to two evaluation devices and measure the one-hop delay from a device to the switch. To be specific, *probing* collects the RTT along the path of the source device, switch, sink device and then divide it by 4; others rely on the Network Time Protocol (NTP) for clock synchronization and send the receiving time back from the switch to the source device with embedded metadata (*passport*) or customized packet (*traditional postcard*).



TSN Scheduling and Configuration
Root Cause
Postcards
§ 4.3 Hardware Design
Control Packet
Alarm
Industrial Network
§ 4.1 Cross-Cycle Postcard-based Telemetry Protocol
§ 4.2 Algorithm
Y
N
Localize Fault?
§ 4.2.1 Flow-Centric Fault Refinement
Wait Postcards
§ 4.2.2 Bandwidth-adaptive Postcard Prioritization
Receive Alarm

Figure 4. System architecture of TSNCard.

Theoretically, the one-hop delay primarily consists of packet propagation delay on the network cable and the PHY (physical layer) processing delay, which typically does not exceed $1\mu s$. As illustrated in Fig. 3b, *probing* exhibits an average deviation of about $92\mu s$, while *passport* and *traditional postcard* show deviations over $160\mu s$. Such a level of inaccuracy in time measurement is problematic for TSN traffic analysis.

Lessons Learned. To advance effective TSN fault localization, our investigation reveals three aspects could be enhanced:
(i) According to C1, we can design a new network analysis paradigm that is able to pinpoint the root cause of the complex chain of errors, thus facilitating effective and responsive repair of critical system failures.
(ii) Motivated by C2, the fault localization algorithm should be adaptive to the bandwidth limitation of particular environments and ensure no vital information is lost in the process.
(iii) Following C3, we should enable the diagnostic system to access globally synchronized high-precision time, allowing them to record each packet's entering and leaving time.

C. TSNCard: System Goals

Goal 1: Full Coverage on Time-related Misbehavior. TSNCard should ensure comprehensive data collection and reliably identify the root cause of time-related periodic network issues in all scenarios. This is essential for the effectiveness of any TSN fault localization system (§V-B).
Goal 2: Light Weight. TSNCard should maintain low bandwidth consumption, low data collection time, and avoid disrupting existing critical traffic flows. This allows the diagnostic system to be seamlessly deployed under the constrained industrial environments (§V-C).

III. System Overview

TSNCard systematically tackles the aforementioned challenges to achieve reliable fault localization in TSN. As depicted in Fig. 4, it comprises the protocol layer for network telemetry and data collection, the algorithm layer for intelligent data analysis and fault localization, and the hardware layer for high-precision and seamless postcard generation.

Key Functional Modules.
• *Cross-Cycle Postcard-Based Telemetry* focuses on efficiently collecting network diagnostic information. It leverages TSN's periodic nature to gather and analyze data over multiple cycles, reducing bandwidth consumption while maintaining diagnostic information coverage.
• *Flow-Centric Fault Refinement* identifies and traces the misbehavior within network flows. It utilizes the collected postcards to reconstruct error chains and pinpoint the root cause of network faults.
• *Bandwidth-Adaptive Postcard Prioritization* decides whether each switch should generate postcards based on the available network bandwidth. It optimizes the balance between localization efficiency and network load, ensuring efficient and timely fault localization.
• *In-switch Postcard Generation* creates and sends postcards with high-precision timestamps without interfering with the transmission of critical traffic. It is a foundational module to support the smooth operation of TSNCard's protocol and algorithm design.

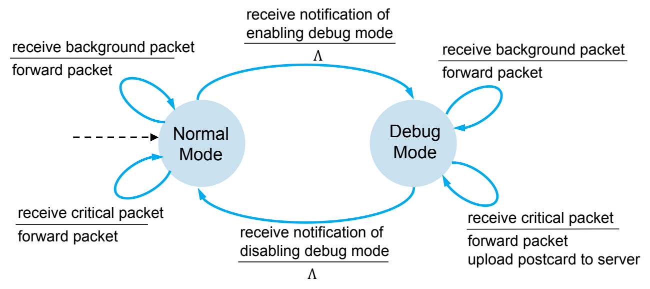
Fig. 4 exhibits the overall workflow of TSNCard. As seen, there are sensors and industrial devices operating over TSN. When a critical data stream is timed out, the end devices will send an alarm to notify TSNCard's analytic server. The analytic server then relies on the *Bandwidth-Adaptive Postcard Prioritization* algorithm to select switches for postcard collection. These switches, equipped with TSNCard's customized hardware logic, send postcards with transmission timestamps and critical packet identities to the server. Afterwards, the server employs the *Flow-Centric Fault Refinement* algorithm to analyze this information and try to locate the root cause of the network issue. Once the root cause is identified, an operator will be notified to repair malfunctioning devices. Otherwise, the server instructs the *Bandwidth-Adaptive Postcard Prioritization* module with the next flows or nodes to analyze and iterate the process until the root cause is localized.

IV. Design and Implementation

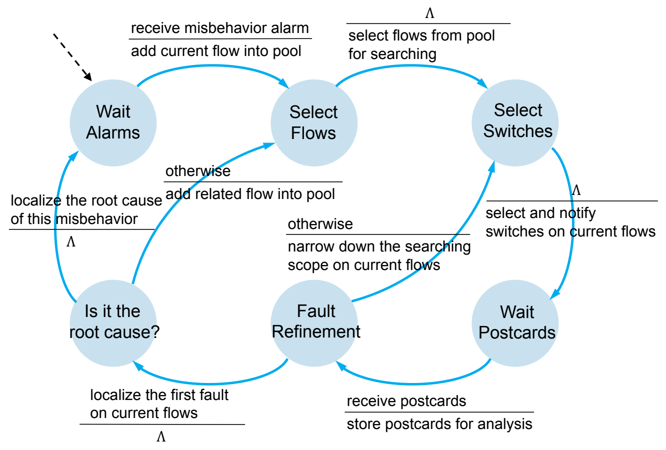
A. Protocol: Postcard-based Telemetry

As explained in §II-B, traditional Ethernet diagnostic tools lack the ability to localize malfunctioning switches or end devices in the TSN network. In TSNCard, we propose a Postcard-based Cross-Cycle Telemetry Protocol dedicated to localizing broken root switches or end devices in the TSN network. The automatons in Fig.5 depict the design of this protocol. The first, as shown in Fig.5a, governs how the switches collect and upload postcards. If the switch is notified of enabling debug mode, it timestamps all critical packets' arrival and departure, i.e., rx and tx. But if not, the switch only conducts pure packet forwarding. The second one, as shown in Fig.5b, shows how to localize the root cause on the analysis server. Once receiving a misbehavior alarm from a sink end-device, the analysis server enters the diagnosis mode. The server will localize the first misbehavior on a single flow and then search all related flows recursively until the root cause is found in the whole network.

In Cross-Cycle Postcard-based telemetry Protocol, we introduce two modules, i.e., *Cross-cycle data fusion* and *Active postcard-based telemetry*, to overcome the weakness of traditional Ethernet diagnostic protocols.


receive notification of enabling debug mode
Λ
receive background packet
forward packet
receive background packet
forward packet
Normal Mode
Debug Mode
receive critical packet
forward packet
receive notification of disabling debug mode
Λ
receive critical packet
forward packet
upload postcard to server

(a) Automaton on the switch.


receive misbehavior alarm
add current flow into pool
Λ
select flows from pool for searching
Wait Alarms
Select Flows
Select Switches
otherwise
add related flow into pool
localize the root cause of this misbehavior
Λ
otherwise
narrow down the searching scope on current flows
Λ
select and notify switches on current flows
Is it the root cause?
Fault Refinement
Wait Postcards
localize the first fault on current flows
Λ
receive postcards
store postcards for analysis

(b) Automaton on the analytic server.

Figure 5. Cross-Cycle Postcard-based Telemetry Protocol.

*1) **Cross-cycle data fusion**:* Distinguished from traditional Ethernet, TSN usually serves applications whose traffic has periodic characteristics and requires strict low latency. As a result, if some device is misconfigured or broken down, the misbehavior in this traffic also appears periodically. This inspires us to propose cross-cycle data fusion. Specifically, it is unnecessary to snapshot the whole TSN network. The key characteristics of the time-sensitive flow, i.e., the arrival and departure times at each hop, are consistent in all cycles after misbehavior occurs. So in the current cycle, we can inspect the timestamps of only the traffic that we are currently most interested in, and check other flows in subsequent cycles. With the help of cross-cycle data fusion, we can dramatically decrease the bandwidth consumption during network telemetry. In addition, we can also make TSNCard efficient and flexible under different bandwidth limitations (refer to §IV-B2).

*2) **Active postcard-based telemetry**:* We design the TSN diagnostic protocol based on active postcard telemetry [15]. Benefiting from postcard telemetry, there is no need to modify the packet structure of critical flows, which means TSNCard is compatible with existing end-devices in TSN networks in factories. If a TSNCard switch enables debug mode, it will actively record the highly precise arrival (rx) and departure(tx) time on the hardware side (refer to §IV-C). On the software side, packets in the critical flows of interest are filtered. Then, rx, tx, and each packet's identity are packed into a postcard and sent to the server for further analysis (refer to §IV-B). The packet identity is a series of values that are invariant during packet propagation. The analysis server utilizes the identities to cluster packets together in the same flow. In our implementation, the packet identity contains the source and destination MAC addresses and the raw packet's MD5 value.

Table II
SEVEN CATEGORIES OF MISBEHAVIOR.

	Criterion	Misbehavior Category
Ingress-Related Misbehavior	$rx + T < rx^s$	Early Ingress
	$rx > rx^s + T$	Late Ingress
	$rx - rx^s > P$	Periods Late Ingress
Egress-Related Misbehavior	$tx + T < tx^s$	Early Egress
	$tx > tx^s + T$	Late Egress
	$tx - tx^s > P$	Periods Late Egress
Loss Misbehavior	No Postcard	Packet Loss

B. Algorithm: Fault Refinement and Bandwidth Adaptation

Due to the existence of *chain of error*, there may not be direct connections between the breakdown switch and misbehaved flows. Nevertheless, according to C1 in §II-B, the misbehavior is propagated along the flow path and leaves traces. This inspires us to propose *Flow-Centric Fault Refinement* module. It exploits misbehavior traces, which are collected by postcards, to recover the complete error chain and localize the breakdown switch, i.e., the root cause. Furthermore, in order to make TSNCard lightweight enough for the actual product line in factories, we introduce *Bayesian Guided Postcard Prioritization*. It automatically adjusts the postcard collection strategy according to the bandwidth limitation, balancing the bandwidth consumption and the fault localization delay.

*1) **Flow-centric fault refinement.**:* The misbehavior trace refers to the inconsistency of the actual packet arrival/departure time with the schedule. As depicted in Table II, we classify misbehavior into seven categories. The second column presents the criteria for the identification of the misbehavior. The superscript s denotes the scheduled timestamp. P is the period of this flow while T is the tolerable deviation between the actual timestamp and the scheduled one. If the server fails to collect some postcard, it is a Packet Loss misbehavior. To avoid postcard loss from the switch to the analytic server, TSNCard conducts collection in three consecutive cycles. The Packet Loss misbehavior is recognized when, and only when none of these three are collected.

Flow-centric fault refinement adopts two complementary phases to tracking the misbehavior comprehensively.

Intra-flow refinement phase. In a single flow, if one switch misbehaves, there are two scenarios: (i) The misbehavior is caused by the previous node; (ii) the misbehavior just occurs on the current switch, that's to say, the current switch is the first misbehaved node and all previous nodes behave normally. In a word, the misbehaved switch could not be the root cause when there is another misbehaved switch on its upstream. In the intra-flow refinement phase, we check each switch in flow f using the criteria in Table II until finding the first misbehaved switch SW*. We can assert that switch SW* is the cause of flow f missing deadline.

Inter-flow refinement phase. However, from a global perspective, switch SW* may not be the root cause. This is because the misbehavior, as shown in Fig. 2, may pass between flows. In the inter-flow refinement phase, we check all connected flows of flow f to judge whether SW* is the root cause. If all connected flows meet the deadline and behave normally, then SW* must be the root cause and needs repair. Otherwise, go back to the intra-flow refinement phase and check misbehaved flows and switches recursively.

*2) **Bandwidth-adaptive postcard prioritization**:* In the intra-flow refinement phase, if all postcards of current flows are collected in one cycle, the bandwidth constraint is surpassed. On the other hand, if taking advantage of periodic characteristics and collecting one postcard each cycle, the available bandwidth is underutilized and the fault localization delay will be high. As mentioned before, misbehavior is transitive, so the misbehaved switches on a specific flow are continuous. This implies that it is feasible to select several switches to check and narrow down the scope of the first misbehaved switch iteratively. A naive greedy strategy is to prioritize switches with more flows passing through them. This strategy is somehow reasonable because more pass-through flows mean a greater likelihood of being impacted, but it fails to take the topology and flow characteristics into account.

TSNCard introduces *bandwidth-adaptive postcard prioritization* to guide the switch selection. We refer to the Naive Bayes algorithm and design a heuristic selection strategy. As shown in Alg. 1, the K switches with the largest $P(y|x,f)$ values are selected (line 4-5), where $P(y|x,f)$ denotes the probabilities of there is misbehavior on switch y in flow f under the condition switch x in flow f misbehaves. The Alg. 2 shows how to calculate $P(y|x,f)$. According to Bayes' theorem, there is

$$P(y|x,f) = \frac{P(x|y,f) \cdot P(y,f)}{P(x,f)}, \quad (1)$$

where $P(*,f)$ denotes the switch $*$ in flow f misbehaves. Since $P(x,f)$ is constant, we have (line 6 in Alg. 2)

$$P(y|x,f) \propto P(x|y,f) \cdot P(y,f). \quad (2)$$

In Eq.(2), $P(y,f)$ is the prior probability while $P(x|y,f)$ is the likelihood. In our implementation, we exploit a network simulator to generate a mass of network operation data and obtain $P(y,f)$ according to statistics (line 1 in Alg. 2). $P(x|y,f)$ denotes the probability that the misbehavior on switch y is transmitted to switch x on flow f. As x and y are both in the flow f, we can decompose $P(x|y,f)$ as follows (line 5 in Alg. 2)

$$P(x|y) = P(x_1|y)P(x_2|x_1)\cdots P(x_m|x_{m-1})P(x|x_m), \quad (3)$$

where all link (x_t, x_{t+1}) belong to flow f. For each $P(x_{t+1}|x_t)$, we notice that it has two properties: (1) $P(x_{t+1}|x_t)$ is **inversely proportional** to D, where D is the number of flows that pass through link (x_t, x_{t+1}). (2) $P(x_{t+1}|x_t) \in [0,1]$. If $D = 1$, $P(x_{t+1}|x_t) = 1$. If $D \to +\infty$, $P(x_{t+1}|x_t)$ should be relatively small because it is very likely that the misbehavior could be transmitted to another flow.

In our implementation, we set $P(x_{t+1}|x_t) = 1 - 0.5(1 - e^{-0.5D}))$ empirically (line 4 in Alg. 2). As a result, the entire calculation progress of $P(y|x,f)$ is shown in Alg. 2.

ALGORITHM 1: Bandwidth-adaptive postcard prioritization

```
input : Current flow $f$, misbehaved switch $x$, $K$
output: $K$ switches in $f$ for postcard collection
1 $path$ ← Get all switches from the source of $f$ to switch $x$;
2 foreach switch $y$ in $path$ do
3     Calculate $P(y|x,f)$
4 $sorted\_path$ ← sort $path$ in descending order according to
      $P(y|x,f)$;
5 $K\_Switches$ ← the first $K$ switches in $sorted\_path$;
6 return $K\_Switches$
```

ALGORITHM 2: Calaculate $P(y|x,f)$

```
input : Current flow $f$, switch $y, x$
output: $P(y|x,f)$
1 Obtain $P(y,f)$ according to statistics;
2 $P(x|y,f) \leftarrow 1$;
3 foreach link $(s,t)$ in path from $y$ to $x$ in flow $f$ do
4     $P(s|t) \leftarrow 1 - 0.5(1 - e^{-0.5D_{s,t}})$;
5     $P(x|y,f) \leftarrow P(x|y,f) \cdot P(s|t)$;
6 $P(y|x,f) \leftarrow P(x|y,f) \cdot P(y|f)$;
7 return $P(y|x,f)$
```

C. Hardware: High-precision In-switch Postcard Generation

As demonstrated in Fig. 6(a), an intuitive approach for timestamping is to implement external timestamper devices and insert them into network cables. They forward the data directly between input/output ports and utilize the 802.1AS SYNC messages passing for time synchronization. When critical flows pass through a timestamper, they can record the transmission time of the packet and generate corresponding postcards. However, this method has two downsides: (1) The PHY (Physical Layer) processing on the timestamper introduces additional packet transmission latency. (2) It necessitates extra devices attached to each network cable, leading to higher deployment costs. Accordingly, in TSNCard, we choose the in-switch postcard design, integrating the hardware implementation into existing TSN switch's logic as a plugin. In this way, TSNCard can access the local high-precision clock of each switch, and will not introduce additional PHY processing delay. Specifically, TSNCard leverages a NetFPGA-based TSN switch architecture [21], [22] and the Xilinx Zynq platform. As shown in Fig. 6(b), it consists of a hardware PL (Programmable Logic) part and a software PS (Processing System) part. The PL records the transmission and packet identity and uploads them to PS, while the PS packages and sends postcards to the analytic server. The core design of TSNCard's hardware implementation is as follows:

Timestamp Modules. The timestamp modules are supposed to measure the precise timing a critical data frame arrives at (rx) and departure from (tx) the TSN switch. The rx timestamp module is positioned right after the MAC (Media Access Control) and before its buffering FIFO. It records the 64bit rx nanosecond-level timestamp when the first byte of each packet is decoded from MAC. Similarly, the tx timestamp module continuously monitors the mirrored output data stream and records the timestamp when the first byte of a packet is successfully sent.

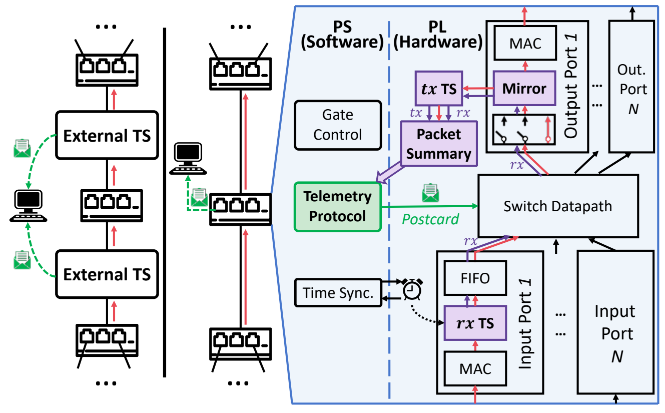


Figure 6. Illustration of external timestamper and TSNCard's in-switch postcard hardware design.

Frame Mirroring Mechanism. The frame mirroring mechanism is designed to capture the critical traffic information without hindering its transmission. It is placed after the transmission selection module (containing the TSN cyclical control gates) of each output port. The module mirrors the AXI-stream signal into two paths: the main path directly connects to the port's PHY, while the other path copies the data byte-by-byte into a packet-mode buffer FIFO (First-In-First-Out queue), strictly following the state machine transition process of the main signal. In addition, the module incorporates a FIFO time stamp rx and a tx, each with 64 bit width. When the first byte of a mirrored data frame is stored in the buffer, the accompanying rx signal is pushed into the rx FIFO, and the current timestamp is pushed into the tx FIFO. After the whole packet is stored in the buffer FIFO, the packet content, rx and tx timestamps, will pop out from the FIFO and be ready for transmission to PS.

Following the trend of in-network computing, major network device vendors nowadays are actively embedding their switches with additional programmable logic. For example, Kontron's PCIE-0400-TSN NIC contains an Altera Cyclone v5 FPGA [23], Intel Tofino switch contains a programmable ASIC chip supporting P4 language [24]. These designs enable the end users to get more control over network switch's operation logic. Given the modular design and validated performance of TSNCard, integrating it into these commercial TSN switches would require minimal redevelopment effort.

V. Evaluation

A. Experimental Methodology

Testbed setup. We integrate TSNCard's hardware design into the a FPGA-based TSN switch on top of the Xilinx Zynq platform [18], and run the analytic algorithm on a server equipped with a 36-core Intel Core i9-10980XE@3.00GHz and 128GB RAM. As depicted in Fig. 7a and Fig. 8a, we set up two testbeds with different topologies, i.e., Ring6 and A380. A380 is a simplified topology of the control network used on Airbus A380 [25], while Ring6 is popular in industrial networking settings [26]. On each testbed, we stochastically generate several critical flows with a period of $1ms$. Each flow's deadline is between 50 and 200μs. We model the TSN scheduling problem as a Satisfiability Modulo Theory (SMT) and solve it with a public Z3-solver [27].

Simulation setup. In addition to physical testbeds, we also conduct comprehensive simulation evaluations with large-scale network topologies based on OMNeT++ and INET Framework [28], [29]. We use the Barabási–Albert model [30] to generate network topologies with 30 TSN devices and 20 TSN switches. Specifically, each time a new switch is added to the generated network. Each new switch is connected to m existing switches with a probability that is proportional to the number of links that existing switches already have. For the devices, each of them is randomly connected to a switch.

Errors in TSN switches. In our evaluations, we focus on the following three kinds of errors in TSN switches:

- **Incomplete Protocol Support.** Commercial TSN products may not fully implement or comply with the complex and numerous TSN standards, leading to operational discrepancies. To mimic the incomplete support of the guard band mechanism in IEEE 802.1Qbv [2], we intentionally delay the frame transmission time in our experiments.
- **Network Misconfiguration.** It is very error-prone to configure the network switches, especially by hand. We deliberately modify the Gate Control List (GCL) to simulate such network configuration errors.
- **Hardware Logic flaws.** Hardware designs are not always trustworthy. Sometimes commercial devices may overlook certain corner cases, resulting in imperceptible hardware logic flows. We modify the logic of the egress queue on the specific switch's output port to emulate these hardware bugs.

We refer to the above errors as Packet, Gate, and Queue, respectively.

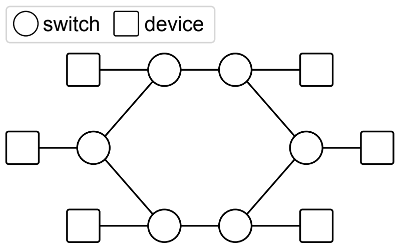
Metrics. We first use success rate, the rate TSNCard localizes the breakdown switches successfully, to evaluate the overall performance. We further measure the bandwidth consumption and localization latency. Since the computation latency of the fault refinement algorithm is relatively small and negligible, we only take the postcard collection latency into account. l_t, the collection latency at cycle t, is measured at 1Mbps bandwidth. The final total postcard collection latency L is calculated as follows

$$L = \sum \max\left(l_t, P\right). \tag{4}$$

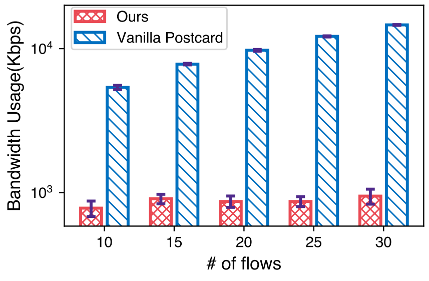
Dataset for Bayes algorithm. In order to calculate the prior probability in bandwidth-adaptive postcard prioritization, we generate a large amount of traffic data using the network simulator OMNeT++. Specifically, for every flow schedule in each topology, we generate thousands of sets of network traffic data and use them to statistically compute $P(y, f)$.

B. Overall Performance

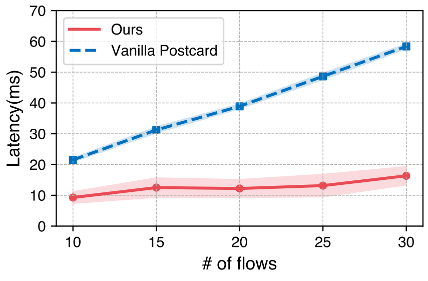
Table III displays TSNCard's success rate of localizing the breakdown switch. We conduct testbed experiments on Ring6 and A380 topologies respectively. On each topology, we generate 10~30 flows and randomly add errors in one of the switches. We also conduct simulation experiments on Random



(a) Topology

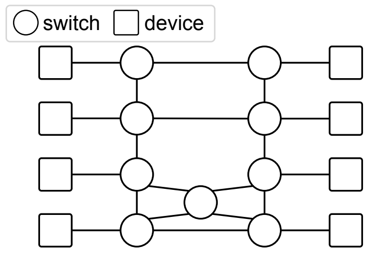


(b) Bandwidth Usage

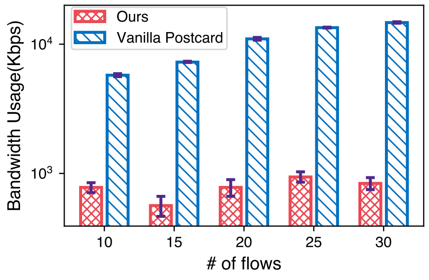


(c) Postcard Collection latency

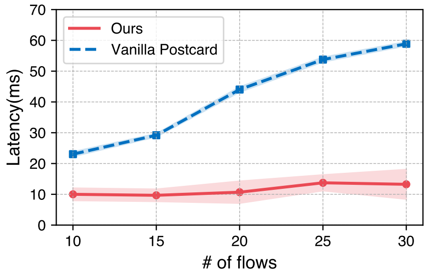
Figure 7. Performance on Ring6



(a) Topology



(b) Bandwidth Usage



(c) Postcard Collection latency

Figure 8. Performance on A380

Table III
SUCCESS RATE OF ROOT CAUSE LOCALIZATION.

Topology	#Flows	#Cases				Success Rate
		Total	Queue	Gate	Packet	
Ring6	10	1185	147	326	712	100%
	15	1337	197	267	873	100%
	20	1296	131	262	903	100%
	25	1209	146	328	735	100%
	30	1186	164	200	822	100%
A380	10	1248	198	182	868	100%
	15	1285	158	379	748	100%
	20	1373	227	345	801	100%
	25	1249	212	318	719	100%
	30	1206	192	330	684	100%
Random	50	1242	184	255	803	100%
	100	1346	154	376	816	100%
	150	1285	208	421	656	100%
	200	1377	171	359	847	100%

topology and generate 50∼200 flows. In every configuration, over 1000 tests are conducted. As shown in Table III, TSNCard successfully localizes the switch with errors in all test cases, demonstrating the effectiveness of TSNCard.

Fig. 7b and Fig. 8b illustrate the bandwidth usage of TSNCard and *vanilla postcard*, which collects all postcards in the TSN network at one time. In all cases, TSNCard consumes less than 1Mbps bandwidth, while the *vanilla postcard* consumes about 10 times more bandwidth and the consumption increases significantly with the number of flows. Moreover, in Fig. 8b, TSNCard's bandwidth consumption of 10 flows is larger than that of 15 flows. The rationale is that the generated flows in the 15-flow case are relatively short and have less interdependence. TSNCard can localize the breakdown switch with fewer postcards.

Fig. 7c and Fig. 8c depict the postcard collection latency of TSNCard and *vanilla postcard*. TSNCard exhibits a stable latency of around 10ms, largely unaffected by traffic scale. On the contrary, the *vanilla postcard* already experiences a latency of around 20ms with 10 flows, reaching approximately 60ms with 30 flows. TSNCard excels a 50∼83% improvement compared to the *vanilla postcard*.

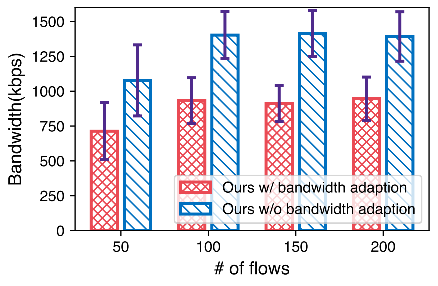
C. Component Study

Bandwidth Adaptive postcard prioritization. First, we conduct simulation experiments to validate the importance of the bandwidth adaption module. In this experiment, we remove the bandwidth adaptation module from TSNCard and collect all postcards on current flows in one cycle. As depicted in Fig. 9a, TSNCard's bandwidth usage is always no more than 1Mbps, while the baseline, i.e., Ours w/o bandwidth adaptation, consumes more bandwidth because it collects more postcards. Thanks to TSNCard's cross-cycle mechanism, the bandwidth consumption of the baseline is bounded even when the number of flows continues to increase. As for Fig. 9b, TSNCard completes the postcard collection within $15ms$, while the baseline takes slightly more, about 17 ms. Furthermore, when the number of flows≥100, we find that the latency hardly changes, which is due to the fact that the length of the error chain does not grow with the number of flows. In fact, since flows are randomly generated, the vast majority of chains of errors have a length of 2 to 4.

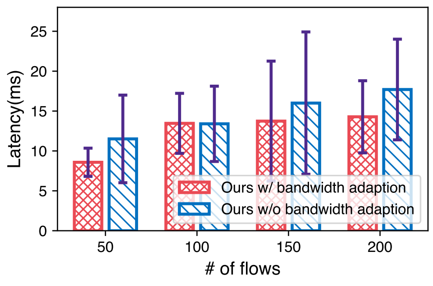
Second, we dive into Bayes-guided postcard prioritization module. We compare it with three postcard selection strategies

- **Random**: Select K switches on the current flows randomly.
- **Uniform**: Select K switches on the current flows uniformly.
- **Greedy**: Select K switches where most flows pass on the current flows.

As shown in Fig. 10, we measured the data collection latency of these methods at different bandwidths. The latency of Bayes-guided postcard prioritization is always the lowest compared to the other three strategies. Moreover, it is notable that the latency of the four methods becomes almost the same



(a) Bandwidth usage



(b) Postcard Collection Latency

Figure 9. Performance on Random Topology

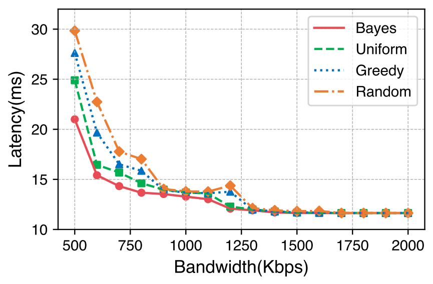


Figure 10. Performance on Different Searching Methods

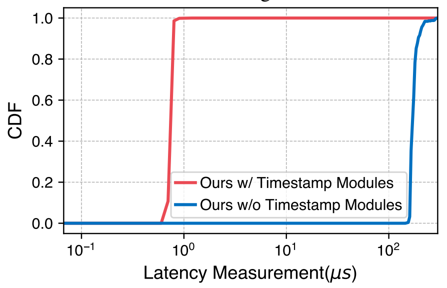


Figure 11. Tracing Precision

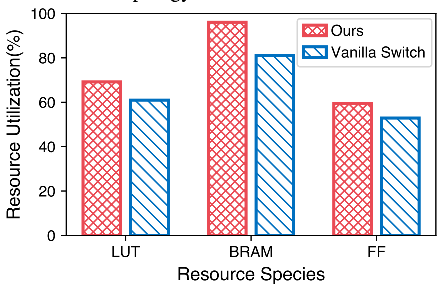


Figure 12. Resource Utilization

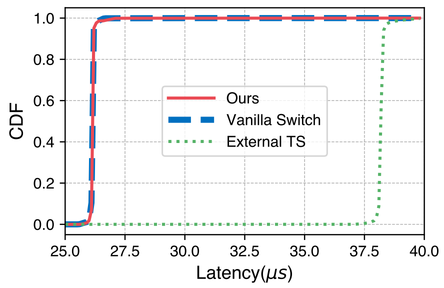


Figure 13. Packet Forwarding

when the bandwidth is larger than 1.5Mbps, This is because the bandwidth is large enough to collect all postcards of the current flows in one cycle.

Timestamp Modules. We remove Timestamp Modules and obtain rx and tx of packets from the software side. Fig. 11 illustrates the packet propagation delay measured by both methods. It is noticed that, compared to obtaining timestamps from the software, the accuracy of TSNCard is improved by two orders of magnitude, and the jitter is also significantly reduced. This suggests that Timestamp modules are essential for TSNCard to monitor and diagnose the TSN network.

D. Overhead Study

Hardware resource utilization. Fig. 12 illustrates TSNCard's utilization of Look Up Table (LUT), Block RAM (BRAM) and Flip Flops (FF) on FPGA. Compared to the original TSN switch, TSNCard uses around 8% more LUT, 15% more BRAM and 7% more FF. The result demonstrates that TSNCard demands few hardware resources and could be integrated into traditional TSN switches with little additional hardware cost.

Packet Forwarding Delay. We compare the packet forwarding delay of TSNCard switch to that of the vanilla TSN switch and External TS (refer to Fig. 6(a)). Since the propagation delay is constant, we send packets between two TSN devices connected by a switch and measure the end-to-end delay. The results are shown in Fig. 13. The packet forward delay receives little impact from the hardware modules introduced by TSNCard. This implies that TSNCard provides an almost imperceptible approach to monitoring critical TSN traffic. On the other hand, the delay of External TS increases significantly, so this class of measurement tools is not well suited for TSN monitoring.

VI. Related Work

Probing implies sending test packets through the network actively to gather information on network conditions such as latency, packet loss, and throughput. Classic tools, including Ping [5] and Traceroute [6], are essential for diagnosing connectivity issues and understanding the network topology. Pingmesh [7] exploits probing methods to measure network latency in large-scale data centers, identifying whether issues are network-related. Time Series Latency Probes (TSLP) [8] represents an advanced application of probing, focusing specifically on detecting congestion on inter-domain links.

Tracing in network diagnostics involves tracking the path of packets through the network to identify bottlenecks, performance issues, and routing paths. Many trace-based tools, such as Tcpdump [9] and Wireshark [10], are crucial for network troubleshooting and analysis. Everflow [31] uses a packet filtering mechanism to trace specific packets across network components, aiding in fault identification and resolution in data center networks. dShark [11], another tracing tool, focuses on distributed packet capture in the network to diagnose network problems, offering deep insight into packet behavior between different network hops.

In-band network telemetry (INT) [12] is an emerging technique for collection and analysis of network statistics. Distinguished from traditional diagnostic techniques, it inserts metadata into packets at each hop and collects them for analysis. It is also named *passport* because its process is similar to the stamping of passports. In order to reduce the telemetry data added to each packet, PINT [13] efficiently encodes the requested data across multiple packets. Moreover, TCP-INT [14] enhances INT by making telemetry data directly available to end-hosts, offering a more integrated perspective on network and application performance.

Postcard telemetry [15] generates and sends packet metadata to the analytic server for packets traversing them, resembling a post office sending postcards for each traveler. Benefiting from the postcard mechanism, network telemetry protocols are able to collect metadata without interference with the packet forwarding time. HyperSight [16] incorporates a declarative query language and a Bloom Filter Queue (BFQ) algorithm for behavior-level network monitoring. NetSight [17] is an extensible platform based on the postcard mechanism that captures packet histories, providing an in-depth view of network traffic and helping to efficiently troubleshoot the network.

Besides, there is some preliminary work focusing on the diagnostics of Time-Sensitive Networking. TSN-Insight [3] enables the computation of the clock offset between each node and the master node, marking a critical step towards monitoring time-synchronization issues. Albeit inspiring, it ignores the issue of inter-node transmission delay. TSNPeeper [4] proposes a theoretical model that stores the TSN forwarding misbehavior on switches and utilizes active probing packets to collect them for the first time. However, it cannot localize the root cause in the potential chain of errors.

VII. Conclusion

In this work, we design and implement TSNCard, a pioneering network diagnostic system for Time-Sensitive Networking. TSNCard represents a significant leap forward in TSN diagnostic capabilities, integrating advanced telemetry protocols, fault refinement algorithms, and cutting-edge hardware design. On this basis, TSNCard enables localizing network faults in bandwidth-constrained TSN applications quickly and accurately. Extensive evaluations on simulation and testbeds demonstrate TSNCard's superior performance. We envision TSNCard as a critical step in advancing TSN development.

VIII. Acknowledgments

This work is supported in part by the National Key Research Plan under grant No. 2021YFB2900100, the NSFC under grant No. 62302259 and No. 62302254.

References

[1] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business & information systems engineering*, vol. 6, no. 4, pp. 239–242, 2014.

[2] *Enhancements for Scheduled Traffic*, IEEE Std. 802.1Qbv, 2015.

[3] T. Bu, Y. Yang, X. Yang, W. Quan, and Z. Sun, "TSN-Insight: An Efficient Network Monitor for TSN Networks," in *Sigcomm Poster*, 2019.

[4] C. Zhang, B. Zhou, Z. Tian, L. Cheng, Y. Liu, H. Zhang, S. Chen, Y. Wan, W. Xu, T. Pan, Y. Xu, Y. Wang, H. Zhu, and B. Liu, "TSN-Peeper: An Efficient Traffic Monitor in Time-Sensitive Networking," in *2022 IEEE 30th International Conference on Network Protocols (ICNP)*, Oct. 2022, pp. 1–11.

[5] J. Postel, "Internet Control Message Protocol," Internet Engineering Task Force, Request for Comments RFC 792, Sep. 1981.

[6] G. S. Malkin, "Traceroute using an ip option," Internet Engineering Task Force, Request for Comments RFC 1393, Jan. 1993.

[7] C. Guo, L. Yuan, D. Xiang, Y. Dang, R. Huang, D. Maltz, Z. Liu, V. Wang, B. Pang, H. Chen, Z.-W. Lin, and V. Kurien, "Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 139–152, Aug. 2015.

[8] A. Dhamdhere, D. D. Clark, A. Gamero-Garrido, M. Luckie, R. K. P. Mok, G. Akiwate, K. Gogia, V. Bajpai, A. C. Snoeren, and K. Claffy, "Inferring persistent interdomain congestion," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '18. New York, NY, USA: ACM, Aug. 2018, pp. 1–15.

[9] "Tcpdump and libpcap," Nov. 2023. [Online]. Available: https://www.tcpdump.org/index.html

[10] Wireshark, "The world's most popular network protocol analyzer," Nov. 2023. [Online]. Available: https://www.wireshark.org/

[11] D. Yu, Y. Zhu, B. Arzani, R. Fonseca, T. Zhang, K. Deng, and L. Yuan, "{dShark}: A General, Easy to Program and Scalable Framework for Analyzing In-network Packet Traces," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, 2019, pp. 207–220.

[12] C. Kim, A. Sivaraman, N. Katta, A. Bas, A. Dixit, L. J. Wobker *et al.*, "In-band network telemetry via programmable dataplanes," in *ACM SIGCOMM*, vol. 15, 2015, pp. 1–2.

[13] R. Ben Basat, S. Ramanathan, Y. Li, G. Antichi, M. Yu, and M. Mitzenmacher, "PINT: Probabilistic In-band Network Telemetry," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '20. New York, NY, USA: ACM, Jul. 2020, pp. 662–680.

[14] G. Jereczek, T. Jepsen, S. Wass, B. Pujari, J. Zhen, and J. Lee, "Tcp-int: lightweight network telemetry with tcp transport," in *Proceedings of the SIGCOMM'22 Poster and Demo Sessions*, 2022, pp. 58–60.

[15] H. Song, G. Mirsky, T. Zhou, Z. Li, T. Graf, G. Mishra, J. Shin, and K. Lee, "On-Path Telemetry using Packet Marking to Trigger Dedicated OAM Packets," Internet Engineering Task Force, Internet Draft draft-song-ippm-postcard-based-telemetry-16, Jun. 2023.

[16] Y. Zhou, J. Bi, T. Yang, K. Gao, J. Cao, D. Zhang, Y. Wang, and C. Zhang, "HyperSight: Towards Scalable, High-Coverage, and Dynamic Network Monitoring Queries," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1147–1160, Jun. 2020.

[17] N. Handigol, B. Heller, V. Jeyakumar, D. Mazieres, and N. McKeown, "I Know What Your Packet Did Last Hop: Using Packet Histories to Troubleshoot Networks," in *Proceedings of 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI '14)*. Seattle, WA, USA: USENIX Association, Apr. 2014.

[18] Xilinx. (2021, Jul.) Socs with hardware and software programmability. [Online]. Available: https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

[19] *Timing and Synchronization for Time-Sensitive Applications*, IEEE Std. 802.1AS, 2020.

[20] E. Li, F. He, Q. Li, and H. Xiong, "Bandwidth Allocation of Stream-Reservation Traffic in TSN," *IEEE Transactions on Network and Service Management*, vol. 19, no. 1, pp. 741–755, Mar. 2022.

[21] N. Zilberman, Y. Audzevich, G. A. Covington, and A. W. Moore, "NetFPGA SUME: Toward 100 Gbps as Research Commodity," *IEEE Micro*, vol. 34, no. 5, pp. 32–41, Sep. 2014.

[22] Z. Yang, Y. Zhao, F. Dang, X. He, J. Wu, H. Cao, Z. Wang, and Y. Liu, "CaaS: Enabling Control-as-a-Service for Time-Sensitive Networking," in *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*. New York City, NY, USA: IEEE, May 2023, pp. 1–10.

[23] K. Group. (2023) Pcie-0400-tsn network interface card. [Online]. Available: https://www.kontron.com/en/products/pcie-0400-tsn-network-interface-card/p151637

[24] I. Corporation. (2023) Intel® tofino™ 2. [Online]. Available: https://www.intel.com/content/www/us/en/products/details/network-io/intelligent-fabric-processors/tofino-2.html

[25] F. Boulanger, D. Marcadet, M. Rayrole, S. Taha, and B. Valiron, "A time synchronization protocol for a664-p7," in *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*. IEEE, 2018, pp. 1–9.

[26] P. N. America. (2019) Industrial topology options and profinet. [Online]. Available: https://us.profinet.com/wp-content/uploads/2019/08/Topology.pdf

[27] Z3Prover. (2021) Github repository of z3prover. [Online]. Available: https://github.com/Z3Prover/z3

[28] A. Varga, "OMNeT++," in *Modeling and Tools for Network Simulation*, K. Wehrle, M. Güneş, and J. Gross, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 35–59.

[29] "INET Framework for OMNEST/OMNeT++," Nov. 2023. [Online]. Available: https://github.com/inet-framework/inet

[30] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[31] Y. Zhu, N. Kang, J. Cao, A. Greenberg, G. Lu, R. Mahajan, D. Maltz, L. Yuan, M. Zhang, B. Y. Zhao, and H. Zheng, "Packet-Level Telemetry in Large Datacenter Networks," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. London United Kingdom: ACM, Aug. 2015, pp. 479–491.