Handling Scheduling Uncertainties through Traffic Shaping in Time-Triggered Train Networks

Qinghan Yu[1,2,3], Xibin Zhao[1,2,3,*], Hai Wan[1,2,3], Yue Gao[1,2,3], Chenyang Lu[4], and Ming Gu[1,2,3]

[1]Key Laboratory for Information System Security, Ministry of Education
[2]Tsinghua National Laboratory for Information Science and Technology(TNList)
[3]School of Software, Tsinghua University
[4]Department of Computer Science and Engineering, Washington University in St. Louis

***Abstract*—While trains traditionally relied on field bus to support real-time control applications, next-generation trains are moving toward Ethernet as an integrated, high-bandwidth communication infrastructure for real-time control and best-effort consumer traffic. Time-Triggered Ethernet (TT-Ethernet) is a promising technology for train networks because of its capability to achieve deterministic latencies for real-time applications based on pre-computed transmission schedules. However, the deterministic scheduling approach of TT-Ethernet faces significant challenges in handling scheduling uncertainties caused by switch failures and legacy end devices in train networks. Due to the physical constraints on trains, train networks deal with switch failures by bypassing failed switches using a short circuiting mechanism. Unfortunately, this mechanism incurs scheduling errors as frames bypassing the failed switch may arrive ahead of the pre-computed schedule, resulting in early, unexpected, and out of order arrivals. Furthermore, as trains evolve from traditional communication technologies to TT-Ethernet, the network must support legacy end devices that may generate frames at times unknown to the TT-Ethernet. We propose a novel traffic shaping approach to deal with scheduling uncertainties in TT-Ethernet. The traffic shaper of a TT-Ethernet switch buffers early frames and then releases them at their pre-scheduled arrive time. Furthermore, we devise an efficient buffer management method for the traffic shaper in face of fault scenarios. Finally, we use the traffic shaper to integrate legacy devices into TT-Ethernet. We have implemented the traffic shaping approach in a 24-port TT-Ethernet switch specifically designed for train networks. Experiments show the traffic shaping strategy can effectively deal with scheduling uncertainties incurred by switch failures and legacy devices.**

I. Introduction

Traditional Train Communication Network (TCN) [1] has been widely used in current train control systems. TCN is a hierarchical combination of two field bus [2] systems: multifunction vehicle bus (MVB) and wire train bus (WTB). MVB inside each coach transmits at 1.5Mbps and WTB connecting the MVB parts transmits at 1Mbps. TCN has been standardized in IEC 61375. As the demand for multimedia and wireless network access increases, another data transmission network is built in parallel with the existing train control network. However, the two parallel networks increase the weight of cables, wiring complexity, and equipment costs compared to one single network. Thus, it is desirable to integrate the two networks into a unified communication infrastructure to reduce cost and simplify network management.

[*] Xibin Zhao is the corresponding author, Email: zxb@tsinghua.edu.cn

An integrated train network must support two types of frames: time-triggered (TT) frames for train control and best-effort (BE) frames for consumer traffic (e.g., WiFi access). TT frames of a real-time (RT) flow have fixed periods, offsets, deadlines and sizes. TT frames must satisfy their deadline and jitter constraints, while best-effort frames do not require timing guarantees. Compared to traditional field bus technologies, Ethernet brings significant advantages through high bandwidth, better interoperability and compatibility with existing network devices. Although standard Ethernet cannot provide deterministic real-time communication [3], various real-time Ethernet technologies [4]–[9] have been developed to provide latency guarantees for industrial applications. In particular, time-triggered Ethernet (TT-Ethernet) [10] has emerged as a promising technology for integrated train networks due to its capability to provide deterministic latencies to TT frames while supporting best-effort frames.

TT-Ethernet achieves deterministic communication for TT frames by scheduling their transmissions based on a scheduling table. As the timing characteristics of real-time flows in a train are usually known a priori, the communication schedule of TT frames can be computed offline. However, a train network must deal with scheduling uncertainties incurred by switch failures and legacy end devices.

Switch failures: As train control is safety critical, a train network must prevent switch failures from disrupting real-time communication of TT frames. While real-time Ethernet usually achieves fault tolerance using redundant switches and paths [11], [12], this approach is not applicable to train networks. Due to space and wiring limitations, there is no redundant switches on a train except for locomotives. Instead, according to the IEC 61375-2-5 standard [13], a switch designed for train networks automatically short circuit a failed switch to allow data flows to bypass it. Unfortunately, this mechanism incurs scheduling errors as frames bypassing the failed switch may arrive ahead of the pre-computed schedule, resulting in early, unexpected, and out of order arrivals.

Legacy end devices: As trains evolve from traditional communication technologies to TT-Ethernet, the network must support legacy end devices not designed for TT-Ethernet. As a legacy device may generate frames at times unknown to the TT-Ethernet, the network faces the challenge to support real-time communication involving the legacy devices without disrupting the time-triggered communication in the network.

In this paper, we explore traffic shaping as a mechanism to deal with scheduling uncertainties in train networks. While traffic shaping was traditionally used to manage traffic for network Quality of Service (QoS), we develop a novel traffic shaping approach to handle scheduling uncertainties incurred by switch failures and legacy devices in TT-Ethernet. Specifically, we design a traffic shaper to buffer the early frames until their expected arrival time, so that the frames are consistent with the scheduling table. Furthermore, to efficiently utilize buffer resource in traffic shaping, we propose a buffer management strategy. Finally, we use the traffic shaper to integrate legacy devices into TT-Ethernet. We have implemented the traffic shaping approach in a 24-port TT-Ethernet switch specifically designed for train networks. Experiments show the traffic shaping strategy can effectively deal with scheduling uncertainties incurred by switch failures and legacy devices.

In the remainder of this paper, we provide relevant background for train networks (Sec.II), discuss related works on fault tolerance and schedule synthesis (Sec.III), describe the traffic shaping mechanism (Sec.IV), present our experiments based on an FPGA implementation (Sec.V) and conclude (Sec.VI) this paper.

II. Background

In this section, we provide the background of train networks, including the composition of a train network, the passive bypass setting of switches and an overview of our time-triggered switch.

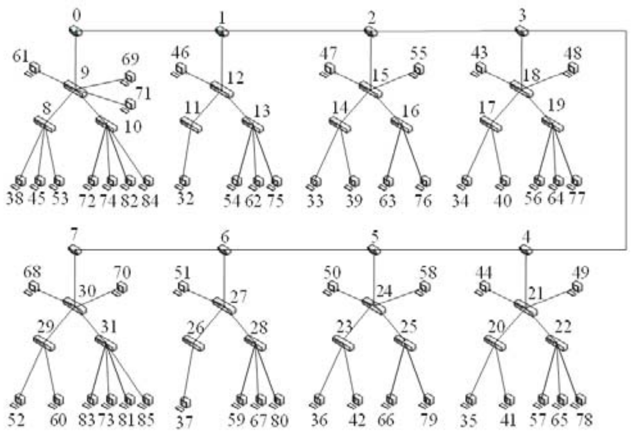

Fig. 1. An example of train network with 8 coaches

A. Train Networks

A train network comprises a subnet in each coach of the train and a *Ethernet Train Backbone (ETB)* connecting the subnets in different coaches. For example, Figure 1 illustrates a train network in a train with 8 coaches. Nodes labeled 0-7 are switches each located in a different coach. These switches form the ETB. Nodes labeled 8-31 are switches connected to end devices (labeled from 32-85). The switches and end devices within each coach form a subnet of ETB.

B. Passive Bypass Setting

According to the IEC 61375-2-5 standard ETB switches support a *passive bypass setting* to handle switch failures. When an ETB switch fails, the ETB automatically short-circuit the failed switch so that the ETB lines bypass the switch. This mechanism is implemented in the external enclosure of the ETB switch. For example, as shown in Figure 2, the failed ETB switch 2 is shorted out through a switch S. Consequently, frames from switch 1 directly arrive at switch 3 bypassing switch 2.

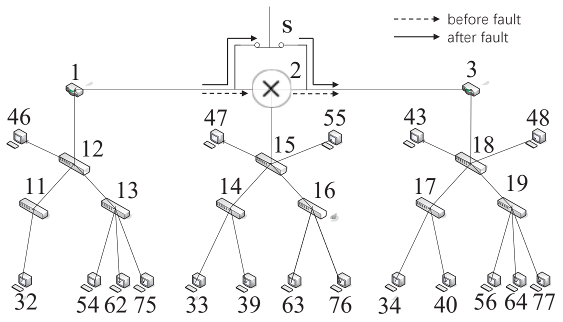

Fig. 2. An example of the passive bypass setting

C. Overview of the Real-Time Switch

We have implemented a real-time switch for train networks based on TT-Ethernet. This switch guarantees real-time frames satisfy their latency and jitter constraints while providing traffic shaping to handle scheduling errors. The architecture our switch is shown in Figure 3.

The modules in white provide the same function as Commercial-Off-The-Shelf (COTS) switches for best-effort frames, while the modules in gray implement a real-time path for real-time frames. The classifier module dispatches real-time frames, clock synchronization frames and best-effort frames to the corresponding module according to the Ether-Type domain of Ethernet II. The RT-engine forwards real-time frames according to pre-downloaded scheduling table. The sync time module processes clock synchronization frames and provides global clock for time-triggered scheduling. The multiplexer module judges which frame to transmit when conflict happens. The traffic shaper provides traffic shaping, which is the focus of this paper.

III. Related Work

Traditionally time-triggered networks rely on redundancy [14] to handle faults. Two common approaches to provide redundancy are spatial and temporal redundancy. Spatial redundancy employs replicated switches and links to handle faults. For example, Time-Triggered Protocol (TTP) [15] fail over to backup switches or links when failure is detected. When temporal redundancy is employed, a frame is scheduled multiple times. Under the hot standby approach, multiple copies of the same frame are transmitted containing serial numbers [16] in multiple channels. In contrast, with a cold

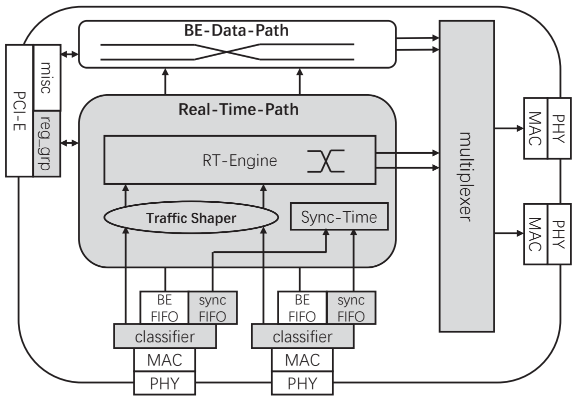

Fig. 3. Time-Triggered switch design for real-time train network. (For clarity only two ports are shown in this figure. Our actual switch implements 24 ports.)

standby approach, an alternative path computed by an offline scheduler is selected when failure is detected [17].

All the aforementioned are based on a classical fail-stop [18] model, which assumes no frame can go through a failed switch or link. Furthermore, all the techniques require redundant switches and links, or a network topology where alternative routes are available. In contrast, due to the space and wiring constraints in trains a ETB does not have replicated devices or alternative paths. Hence, traditional approaches based on redundancy are not compatible with train networks. The novelty of our work stems from the unique fault model and passive bypass settings in ETB switches.

IV. Traffic Shaping

In this section, we first illustrate three abnormal scenarios arising from the passive bypass setting of switches in train network, and define the switch failure problem which this paper first concentrates on. We then present the design of traffic shaping module for this problem and analyze the performance of our design. Finally, we discuss the mechanism to integrate legacy devices with the traffic shaping module.

A. Problem Definition

The RT engine showed in Figure 3 is implemented with two crossbars and shared buffers in the middle, scheduling TT frames according to pre-downloaded time table. Each entry of the table contains a receive point and a send point corresponding to a flow. Due to time precision, the receive point is extended to a receive window. The first crossbar forwards TT frames into the designated buffer according to the receive window, while the second crossbar retrieves frames from shared buffers according to the send point.

As shown in Figure 4, the solid arrows represent the normal scenarios. That is, TT frames is supposed to arrive within the corresponding receive window, and leave at send point.

When a switch fails, the passive bypass setting enables frames to be transmitted from the predecessor switch to the successor switch directly, leading to three abnormal scenarios:

1) All frames arrive at the successor switch in advance

2) Some frames arrive at the successor switch unexpectedly

3) A few frames arrive at the successor switch out of order

As shown in Figure 4, both F_a and F_c arrive at switch 3 in advance when switch 2 fails. The destination of F_b is switch 2, however, switch 3 unexpectedly receives F_b when switch 2 fails. For switch 2, the buffering time of F_a is longer than F_c, and F_c is supposed to leave the switch first. When switch 2 fails, F_a arrives at switch 3 earlier than F_c, instead of the normal scenario. That is, F_a arrives at switch 3 out of order.

All the abnormal scenarios lead to frame missing their corresponding receive window or mismatching the entry of the table. As a result, the frames are dropped by the scheduler, which violates the hard real-time constraints. This paper handles these three abnormal scenarios for TT-Ethernet, hence provides fault tolerance for train networks. However, frame loss of the devices connected to the failed switch is beyond this paper.

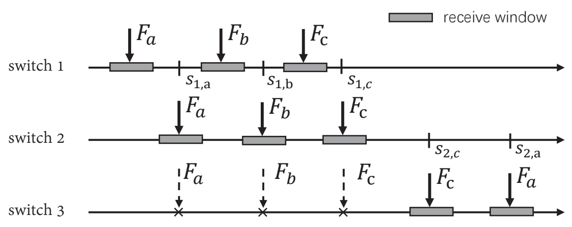

Fig. 4. Abnormal scenarios when switch fails. The axis strands for time. switch 1, switch 2, and switch 3 are connected in sequence. F_a, F_b, F_c are real-time frames transmitted between those switches. $s_{i,j}$ is the send point of F_j in switch i. The solid arrows represent the normal scenario, while the dashed arrows represent the abnormal scenario when switch 2 fails.

B. Traffic Shaping Module

When switch fails, frames arrive at the successor switch in advance. As shown in Figure 4, F_1 arrives at switch 3 in advance. To handle this scenario, we adopt traffic shaping approach by buffering the early arrival frames until legal arrival time. Therefore, these frames match the receiving window, and are accepted by the scheduler again.

Although a standard TT-Ethernet switch has 24 ports in train network, we illustrate the traffic shaping mechanism using a switch with 4 ports for clarity. As shown in Figure 5, traffic shaping is implemented by the traffic shaping module containing four sub-modules *Arbiter*, *Buffer Pool*, *Buffer Allocation* and *Ctrl Logic*, which will be descried in detail in this subsection.

1) Arbiter: The arbiter module specifies the forwarding direction of frames transmitted into the traffic shaping module. RT frames are forwarded directly into RT engine in normal case, and forwarded into buffer allocation module instead when switch failure is detected. A common mechanism of failure detection is Link Layer Discovery Protocol (LLDP).

The arbiter module works at port level, which means only frames from ports connected to the failed switch are redirected, and frames from other ports are not affected. Furthermore, RT engine retrieves data either from the arbiter module in normal case or the ctrl logic module when switch failure is detected. Thus, data race issue does not exist.

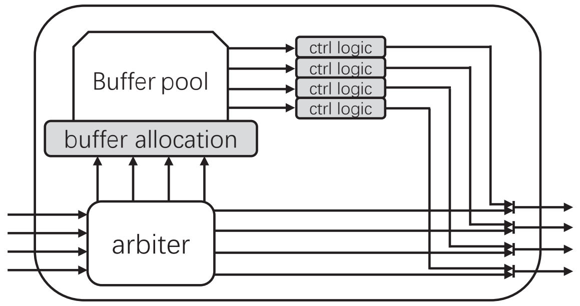

Fig. 5. Buffer Structure Design

2) Buffer Pool: Due to scenario 2 and 3, traditional input queue structure is no longer suited for out-of-order and unexpected frames. Alternatively, we introduce buffer pool for the traffic shaping module design. Buffer pool is a collection of storage resource in hardware, such as SRAM, DRAM, *etc*, in which early arrival frames are held.

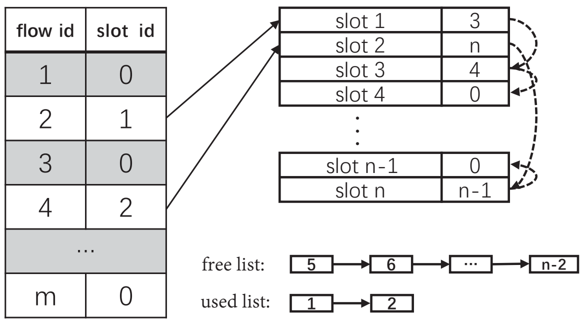

Fig. 6. Buffer Pool Organization

As Figure 6 shows in the right, the storage resource is divided into slots, which is the basic unit of the pool. Each slot contains two fields : data field and pointer field. The size of data field varies according to applications, such as 1518 bytes for a maximum Ethernet frame or 8 bytes for slices. Furthermore, data field size is pre-configured before train running and is equal for each slot. The frame content is stored in the data field, while the ID of the successor slot is stored in the pointer field. It is important to note that the pointer field is 0 if a slot is the last slice of a frame. Therefore, the whole pool is organized by linked list structure, which saves register resource and brings convenience for buffer management.

In Figure 6, the pointer field of slot 1 is 3, which means the successor slice is stored in slot 3. Alternatively, the pointer field of slot 4 is 0, which means slot 4 is the last slice of the frame. Therefore, slot 1,3,4 compose a complete frame.

3) Buffer Allocation: The buffer allocation module maintains two linked lists for buffer management: free list and used list. The free list contains buffers available to be allocated, while the used list contains header slots which are already allocated. Also, this module maintains a map of flow id and slot id, shown in left of Figure 6. Each flow id uniquely corresponds to a slot id. For example, flow 2 corresponds to slot 1. If the corresponding slot id equals 0, no slot is allocated to the flow. Otherwise, this slot id is the first slot allocated to the flow. For example, slot 1 is the first slot allocated to flow 2, which means its successors, slot 3, 4, are also allocated to flow 2. In contrast, slot 0 is allocated to flow 1, which means no slot is allocated to flow 1 currently.

During initialization, all the slots are added to the free list, and all the slots id in the map are set to 0. When a frame arrives, buffer allocation module first removes the header slot in the free list, allocates it to the frame and insert this slot into the tail of used list. If the slot size is less than the frame size, this module then repeatedly removes the header slot of the free list and allocates it to the frame until the entire frame is stored. All the slots allocated to the frame is maintained in allocating sequence by linked list in the buffer pool. If no slot is available for allocation, header slot in the used list and its successor slots in the linked list in the buffer pool are released, and added to the tail of free list. Finally, this module sets the first slot allocated as the slot id corresponding to the frame. In contrast, when a frame leaves, this module first look up the map for the slot id corresponding to the flow id of the frame. If the slot id equals 0, no furtherer action is executed. Otherwise, the slot and all its successor slots are then transmitted to the ctrl logic module. Finally, all the slots transmitted are released, and added to the tail of free list, while the first slot is removed from the used list.

4) Ctrl Logic: Each port corresponds to one ctrl logic module, and the module retrieves frames from the buffer pool for the corresponding port. This module looks up the scheduling table of the port and obtains the flow id and receive window of the frame which is supposed to be transmitted next time. After that, this module retrieves the frame from the buffer allocation module, and forwards the frame to RT engine according to the receive window, so that the early arrival frames can match the window.

Considering all three abnormal scenarios, buffer pool module caches frames arriving in advance, and therefore handle the first scenario. Also, the map structure instead of pure input queue records the storage address for each flow, and thus handles the third scenario. Finally, the slots reside in the used list for a long time are allocated to the unexpected frames. Hence, the release of the header of the used list handles the second scenario. In conclusion, the traffic shaping module handles all three abnormal scenarios.

C. Implementation Discussion

Bus is a common choice of buffer pool implementation, which communicates with external storage through fixed width bus. However, bus introduces read and write serialization, and brings data race, which results in extra access delay. In this subsection, we discuss methods for delay mitigation in hardware implementation.

1) Round Robin Fetch: As frames are sliced into slots, buffer allocation module can fetch frames in a round robin manner. That is, each time buffer allocation module responses to a slot fetch request for a ctrl logic module, it then responses to the request of the next ctrl logic module. With this strategy, short frames are no longer blocked by large frames, and therefore worst case delay is mitigated.

2) Prefetch Mechanism: A prefetch mechanism is implemented in ctrl logic module to utilize idle time. The ctrl logic prefetches 2 packets for each port in advance, and stores the prefetched packets in local buffer. At the left edge of the receive window , ctrl logic sends the packet to RT engine and starts to prefetch the next packet. This mechanism mitigates burst read and write operations, and therefore decreases access delay.

D. Analysis

For real time systems, the performance under worst case is the most significant concern. In this subsection, we discuss the maximum parallel ports supported by the traffic shaping module under worst case.

Each switch S_i has N_{port} ports and the link bandwidth is B_l. The clock cycle of the buffer pool is t_{clk}, and m is the cycle number of read operation delay of the pool. The length of frame is L_i for flow i, and the slot size is $Size_{slot}$ which should be compatible with the data width of buffer pool. We assume that the round robin manner is enabled when fetching slots. The worst case time delay $Delay_i$ for flow i is:

$$Delay_i = 2 \cdot \left\lceil \frac{L_i}{Size_{slot}} \right\rceil \cdot t_{clk} \cdot N_{port} + m \cdot t_{clk} \quad (1)$$

The worst case is that there are N_{port} frames queued for response during read and clkite operations. Since the round robin fetch is applied, the latency invoked by data racing is N_{port} times of the transmission time. Also, latency of read operation needs to be counted. To ensure a frame is able to be forwarded into the RT engine during its receive window, the following inequality is required:

$$Delay_i \leq Q_i + \frac{L_i}{B_l} \quad (2)$$

The right part of the Eq. 2 is the minimal advanced time when switch failure occurs. Q_i is the minimal queuing delay of flow i in a switch, and the latter part is transmission delay of flow i. With Eq. 2, we can get the restriction of N_{port}:

$$N_{port} \leq \frac{Q_i + \frac{L_i}{B_l} - m \cdot t_{clk}}{2 \cdot \left\lceil \frac{L_i}{Size_{slot}} \right\rceil \cdot t_{clk}} \quad (3)$$

For a 100Mbps network, a buffer pool with 200Mhz, $Slot_{size}$ of 64 bytes and read latency of 17 cycles, we can get the largest value of N_{port} supported by the traffic shaping module is 1015, which is far beyond demand. Despite this, the N_{port} still needs to be pre-computed with actual parameters, and checked whether Eq. 3 is satisfied before train runs.

E. Legacy Device Integration

Although TT-Ethernet is compatible with existing network devices, legacy devices of some vendors do not support clock synchronization in train network. When connected with legacy device, the arbiter forwards the arrival frames from the end device to the buffer allocation module. That is, those frames are regarded from failed routers. Then, the ctrl logic modules retrieve these frames according to the legal receive windows, which integrates these frames into synchronized networks. Unfortunately, a worst case delay of a whole period is invoked with this pure mechanism provided by the traffic shaping module, which can easily violates the latency constraint. Therefore, schedule synthesis should cooperate with this mechanism to satisfy latency constraint.

V. Hardware Experiment

We now discuss the experiments we conducted to measure the traffic shaping mechanism. The experiments in this section is based on FPGA hardware platform.

A. Experimental Settings

We use three time-triggered switches with linear topology, and two IXIA 400T traffic generators to setup time-triggered train network testbed, as shown in figure 7.

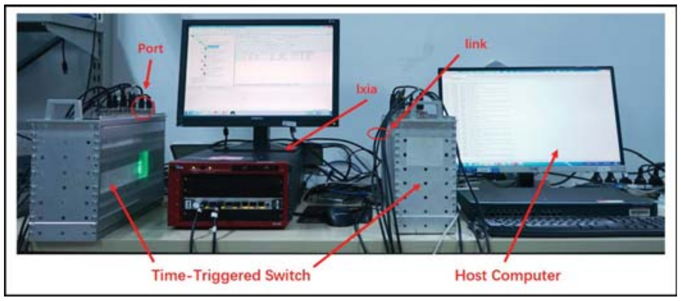

Fig. 7. Time-Triggered Train Network Testbed

Time-Triggered switch is a fully featured real-time switch providing fault tolerance with 24 ports, implemented by FPGA. IXIA 400T generates specific real-time frames with fixed periods, providing clock precision within 10 nanoseconds. The bandwidth of the whole network is 100Mbps and the clock synchronization precision over devices is 200ns using IEEE 1518.

B. Device Utilization

The FPGA used in the real-time switch is Xilinx Vertex 7, containing 607,200 registers, 303,600 LUTs and 1,030 Block RAM. Table I summarizes the device utilization for regular RT switch and RT switch with traffic shaping.

TABLE I
Device Utilization on Vertex 7

resource	switch	switch with traffic shaper	traffic shaper
LUTs	63.19%	71.13%	7.94%
FFs	27.26%	31.34%	4.08%
BRAMs	66.99%	87.96%	20.97%

Since the Vivado wire optimizer will consider the whole design when synthesis, obtaining device utilization of traffic shaping module individually from synthesis reports is not precise. Thus, we simultaneously obtain the device utilization of RT switch with and without traffic shaping module. We regard the difference of the utilization as the implementation cost. Table I shows traffic shaping module consumes little logic resource, as the LUTs and FFs usage is relatively small. Due to conversion between clock domain and requirement for caching frames, traffic shaping consumes a certain storage resource. In general, the implementation cost is acceptable.

C. Handling Switch Failures

In this subsection, we evaluate functionality of traffic shaping mechanism. We generate frames with size of 256 bytes, period of 32ms and deadline of 32ms using Ixia, which is regarded as unsynchronized device. These frames are forwarded by real-time switches with pre-downloaded schedules and are finally captured by Ixia with timestamps. To emulate the fault scenario in train network, we choose a fixed failure time point a (about 2500 ms) as shown in Figure 8 and cut off power of the second switch at that point.

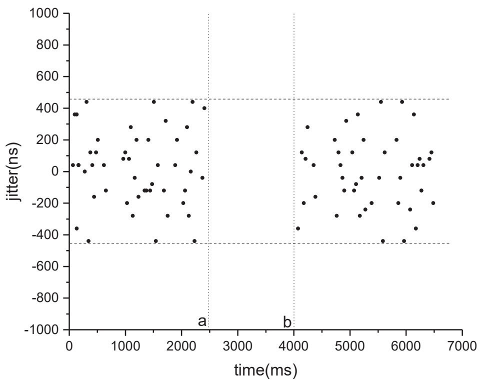

Fig. 8. Jitter during fault tolerance

Before time point **a**, jitter of the frames is within 450ns and satisfies train network constraint of 4ms. At time point **a**, the second real-time switch is cut off, therefore, no frames between time point **a** and **b** are captured. At time point **b**, the bypass setting is activated and the second switch is shorted. That is, the frames from the first switch are directly forwarded to the third switch bypass the second switch and thus arrives in advance. Due to the traffic shaping mechanism, the early arrival frames are cached in the buffer pool and retrieved according to the receive window. Therefore, the frames are captured again after time point **b**. Also, jitter of frames after time point **b** still satisfies train constraint.

VI. Conclusion

Although traditional Ethernet is nondeterministic, Time-Triggered Ethernet provides latency and jitter guarantees with fixed schedules. However, the deterministic scheduling approach of TT-Ethernet faces significant challenges in handling scheduling uncertainties caused by switch failures and legacy end devices in train networks. We propose a traffic shaping approach to deal with scheduling uncertainties in TT-Ethernet. To the best of our knowledge, this is the first paper to introduce traffic shaping for train network. Experiments show the traffic shaping strategy can effectively deal with scheduling uncertainties incurred by switch failures and legacy devices.

Acknowledgment

This research is sponsored in part by NSFC Program (No. 61527812), National Science and Technology Major Project No. 2016ZX01038101,MIIT IT funds (Research and application of TCN key technologies) of China, and The National Key Technology R&D Program (No. 2015BAG14B01-02).

References

[1] H. D. Kirrmann and P. A. Zuber, "The IEC/IEEE train communication network," *IEEE Micro*, vol. 21, no. 2, pp. 81–92, 2001.

[2] R. Patzke, "Fieldbus basics," *Computer Standards & Interfaces*, vol. 19, no. 5-6, pp. 275–293, 1998.

[3] H. Kopetz, "The rationale for time-triggered ethernet," in *Proceedings of the 29th IEEE Real-Time Systems Symposium, RTSS 2008, Barcelona, Spain, 30 November - 3 December 2008*, 2008, pp. 3–11.

[4] "Profinet," http://www.profibus.com/.

[5] "Ethercat technology group," http://www.ethercat.org/.

[6] "Time-triggered ethernet," http://www.tttech.com/.

[7] J. Jasperneite and J. Feld, "PROFINET: an integration platform for heterogeneous industrial communication systems," in *Proceedings of 10th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2005, September 19-22, 2005, Catania, Italy*, 2005.

[8] S. G. Robertz, K. Nilsson, R. Henriksson, and A. Blomdell, "Industrial robot motion control with real-time java and ethercat," in *Proceedings of 12th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2007, September 25-28, 2007, Patras, Greece*, 2007, pp. 1453–1456.

[9] P. Pedreiras, P. Gai, L. Almeida, and G. C. Buttazzo, "Ftt-ethernet: a flexible real-time communication protocol that supports dynamic qos management on ethernet-based systems," *IEEE Trans. Industrial Informatics*, vol. 1, no. 3, pp. 162–172, 2005.

[10] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, "The time-triggered ethernet (TTE) design," in *Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2005), 18-20 May 2005, Seattle, WA, USA*, 2005, pp. 22–33.

[11] Y. Chang, C. Chiu, S. Lin, and C. Liu, "On the design and analysis of fault tolerant noc architecture using spare routers," in *Proceedings of the 16th Asia South Pacific Design Automation Conference, ASP-DAC 2011, Yokohama, Japan, January 25-27, 2011*, 2011, pp. 431–436.

[12] M. Koibuchi, H. Matsutani, H. Amano, and T. M. Pinkston, "A lightweight fault-tolerant mechanism for network-on-chip," in *Second International Symposium on Networks-on-Chips, NOCS 2008, 5-6 April 2008, Newcastle University, UK. Proceedings*, 2008, pp. 13–22.

[13] IEC, "IEC 61375-2-5 Ed.1: Electronic railway equipment - Train Communication Network - Part 2-5: ETB - Ethernet Train Backbone."

[14] G. Bauer and H. Kopetz, "Transparent redundancy in the time-triggered architecture," in *2000 International Conference on Dependable Systems and Networks (DSN 2000) (formerly FTCS-30 and DCCA-8), 25-28 June 2000, New York, NY, USA*, 2000, pp. 5–13.

[15] H. Kopetz and G. Grünsteidl, "TTP - A time-triggered protocol for fault-tolerant real-time systems," in *Digest of Papers: FTCS-23, The Twenty-Third Annual International Symposium on Fault-Tolerant Computing, Toulouse, France, June 22-24, 1993*, 1993, pp. 524–533.

[16] M. Short and M. J. Pont, "Fault-tolerant time-triggered communication using CAN," *IEEE Trans. Industrial Informatics*, vol. 3, no. 2, pp. 131–142, 2007.

[17] G. Avni, S. Guha, and G. Rodríguez-Navas, "Synthesizing time-triggered schedules for switched networks with faulty links," in *2016 International Conference on Embedded Software, EMSOFT 2016, Pittsburgh, Pennsylvania, USA, October 1-7, 2016*, 2016, pp. 26:1–26:10.

[18] J. D. Lin and A. M. K. Cheng, "Real-time task assignment with replication on multiprocessor platforms," in *15th IEEE International Conference on Parallel and Distributed Systems, ICPADS 2009, Shenzhen, China, December 8-11, 2009*, 2009, pp. 399–406.