Redundancy Management for Safety-Critical Applications with Time Sensitive Networking

Maryam Pahlevan[1] and Roman Obermaisser[3]
University of Siegen, Germany
maryam.pahlevan@uni-siegen.de

Abstract— **In modern cyber physical systems like industrial automation systems and Advance Driver Assistance Systems (ADAS), safety is considered as the main concern. Failures in safety-time critical systems may lead to high economic losses as well as dangers for humans and the environment. Therefore, the Time Sensitive Networking (TSN) task group not only introduced real-time properties to Standard Ethernet, but also developed a novel fault-tolerance mechanism called Frame Replication and Elimination for Reliability (FRER). FRER offers highly reliable communication for Time-Triggered (TT) traffic.**

Simulation tools are seen as a cost and time efficient approach to evaluate and verify network protocols during the development phase and before the actual implementation. As no simulation model in the state-of-the-art implements the time-based features (e.g. time aware shaping and policing) and non-time-based properties (e.g. FRER) of TSN at the same time. Hence, in this paper we present a simulation model for temporal properties and redundancy management in TSN networks using the Riverbed simulation framework. In addition, we introduce the fault injection mechanisms to evaluate the reliability (e.g. violation of end-to-end deadline and packet loss) of TT communication based on FRER and different faults. We demonstrate the applicability of the framework using the realistic network use cases and traffic profiles.

I. INTRODUCTION

To take advantage of various Ethernet technologies that offer high bandwidth and seamless connectivity, TSN introduces temporal isolation for mixed-critically systems in forms of IEEE 802.1 protocol extensions. TSN [1] using Time Aware Shaper (TAS) [2] and a new synchronization mechanism (i.e. IEEE 802.ASRev) [3] ensures the deterministic delivery of TT messages in the presence of other traffic types (e.g. Best Effort traffic).

In addition to bounded latency and low jitter requirements, fault tolerance is seen as a key feature of mission critical systems. Standard Ethernet is unable to recover seamlessly from either transient or permanent faults. Therefore, to address fault tolerant communication over Ethernet-based networks, a wide range of protocols such as Rapid Spanning Tree Protocol (RSTP) [4], Parallel Redundancy Protocol (PRP) and High-availability Seamless Protocol (HSP) [5] were developed. All aforementioned protocols enhance reliability and availability of systems using redundant paths.

RSTP, in case of any failure, finds an alternative path for the existing active route in up to 2 seconds. Thus, this approach is not applicable for safety-critical systems with stringent temporal constraints. To resolve this issue, PRP duplicates messages and sends them over two separate networks. Consequently, if one copy of the frame does not reach the destination node due to a failure (e.g. link failure), the second copy will be delivered to the receiver from another network without any interruption. HSP uses the same approach as PRP, but instead of using two separate networks, it transmits duplicated frames over 2 redundant paths within one network. It is good to note that HSP is mainly designed for ring topologies [6].

TSN is tailored to real-time systems with strict timing and safety requirements. Consequently, it is required to support zero packet loss, guaranteed end-to-end delay and low jitter. To achieve this goal, the TSN task group proposes IEEE 802.1CB (i.e. FRER) [7] as a new Ethernet sub-protocol. FRER provides a robust and deterministic behavior for time-sensitive systems using the same fault tolerance concept that is specified in PRP and HSP [7]. FRER is a more generic solution and mitigates some of the common PRP and HSP challenges. For instance, FRER is deployable over any network structure and is limited neither to ring topology nor to two parallel networks. Furthermore, FRER permits seamless communication between Standard Ethernet devices and FRER-capable nodes and despite of PRP and HSP it does not require any proxy for this purpose.

Networking experts and technology manufactures use simulation frameworks extensively to emulate and validate new networking solutions. The implementation and deployment of novel protocols on hardware is a very time consuming and expensive process. Beside this protocol usually require plenty of modifications due to constant changes in the solution designs. As many TSN protocols are not finalized yet, in [8] we developed the simulation models for the time-based features of TSN. Using the TSN simulator, we allow to validate the real-time capability of TSN solutions. In this paper, we extend the TSN models which are presented in [8] to support different FRER functionalities. The fault tolerance capability of TSN is essential for message exchanges between mission-critical applications. We develop the FRER functions in a modular manner, so that they can seamlessly be integrated to the existing TSN models at appropriate stages. In addition, we present a fault injection model to verify the correctness and applicability of the FRER module. We simulate different faulty behaviors (including transient and permanent errors) in our emulated TSN network and evaluate the impact of the FRER module on the reliability of every stream for different scenarios. To best of our knowledge, this is the first work which develops simulation models with both time-based and non-time-based services of TSN. The prior TSN simulation frameworks focus on either non-time-based services (e.g. [9]) or temporal features (e.g. [8]) of TSN. Therefore, this work provides a more comprehensive simulation platform for modeling, performance and reliability evaluation of TSN networks.

The rest of the paper is structured as follows: Section II provides a brief overview of the FRER protocol. In section III, the simulation models of FRER capable devices are described in more details. Section IV presents the evaluation of experimental results which are derived from different network dynamics. The last section concludes the paper.

II. Redundancy Management in TSN

The TSN task group introduces the IEEE 802.1CB standard to improve robustness and reliability of stream transmissions especially for safety-critical traffic. A sender or a relay system (e.g. switch) with FRER capability, first generates and encodes a sequence number for each outgoing frame. Then it forwards the multiple copies of the packets towards the destination over multiple routes. Hence, in case of any failure in one of the routes, the packet is delivered to the destination via the redundant path. Therefore, the FRER mechanism decreases the probability of traffic loss considerably. In TSN, the IEEE 802.1Qca [10] protocol is deployed to configure the alternative routes for each stream. In addition, to avoid network overloading, the duplicated frames are eliminated either at intermediate relay systems or at a receiver.

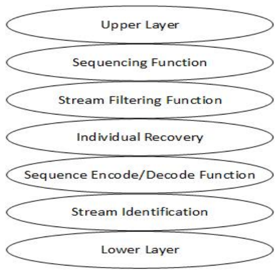


Fig. 1: FRER functions

As shown in Figure 1, FRER consists of five different functions. Depending on the packet processing direction in a FRER capable device, each function acts differently.

A. Sequencing

This module generates a sequence number for every frame passed down from the upper layer to the physical layer. In contrast, for each frame passed up in the protocol stack, the sequencing function examines the sequence number of a frame and discards the duplicated frames whose copies have been received before.

B. Stream Splitting Function

This function makes zero, one or more copies of every packet passed down to the physical layer according to the stream split table. Each copy of a packet will be transmitted to the destination via separate paths.

C. Individual Recovery Function

It checks the sequence number of every frame passed up in the protocol stack and eliminates the frames whose duplicates have been received previously. This function and sequencing provide similar services, but individual recovery functions can apply to multiple ports while the sequencing module is port specific. It is good to note that the recovery module will process a packet only if the integrity of the frame is verified by the lower layer (e.g. physical layer) validity checks.

D. Sequence Encode/Decode

This function encodes a sequence number generated by the sequencing function into the frame passed down for transmission. In the reception direction, the function derives the sequence number from the frame passed up in the protocol stack. FRER introduces the Redundancy tag (R-TAG) as an example of sequence number formatting. R-TAG comprises of three parts: 1) EtherType which has the value F1C1. 2) Reserved field which occupies the second two octets of R-TAG and is reserved for future revisions of IEEE 802.1CB. 3) Sequence number field which is encoded in the last two octets of R-TAG. Figure 2 illustrates the format of the R-TAG.

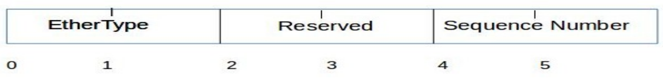


Fig. 2: R-TAG header structure

The encoding mechanism must be known to all relay systems and end-systems. Otherwise they cannot decode the sequence number from incoming frames.

E. Stream Identification Function

This function specifies a stream identifier for every frame received either from the physical layer or the upper layer. The stream identifier determines to which stream the frame belongs and how it should be processed by other FRER functions. The four different stream identification approaches are listed in Table I. The stream identification functions can also overwrite some of the parameters of the frame's header to reflect the stream identifier.

Stream Identification functions	Stream Identifier
Null Stream identification	Dst MAC address, VLAN ID
Source MAC and VLAN	Src MAC address, VLAN ID
Destination MAC and VLAN	Dst MAC address, VLAN ID
IP octuple	Dst MAC address, VLAN ID IP src address, IP dst address IP next protocol, src port, dst port

TABLE I: Stream Identification Functions

Depending on the network design, the FRER functions can be placed in the protocol stack of device's port in different orders. To be more specific, each port of a device selects different FRER functionalities. Due to this flexible design, the FRER capable devices are able to inter-operate with network elements that are unaware of FRER. Furthermore, FRER protects TSN networks from the faulty behaviors like a stuck transmitter. To detect such errors, a network element with FRER capability saves the history of a stream's sequence numbers. Using this information it discards packets with a sequence number different from the expected value.

III. System Models for Fault Tolerant TT Communication

We developed and integrated the FRER module to the TSN time-aware models which we implemented in [8] using the Riverbed simulation framework. Riverbed [11] (former known as OPNET) is a powerful commercial tool which is widely used in the industry and academia to simulate and evaluate different communication layers, network elements and protocols.

A. TSN Configuration Parameters

In TSN networks, the redundant routes and schedules of TT streams are computed offline using the knowledge of the network topology and the TT traffic profiles. In addition, in our TSN models we use the source MAC address and VLAN ID as a stream identifier. For this reason, we provide TT traffic profiles and port-specific GCLs for all TSN-capable devices within our simulator statically. A TT traffic profile defines the stream characteristics while a port-specific GCL determines the state of egress queues at a certain time slot. For more details on configuration parameters, refer to [8].

B. TSN Switch Model

We build our switch model which provides real-time and fault tolerance capability of TSN on top the standard Ethernet switch. In the TSN switch model, first all incoming frames are passed to the stream identification function. This module identifies frames using the source MAC address and VLAN ID fields. This stream identification function is passive and does not modify the frame passed either up or down to protocol stack. The stream identification function using the stream identity table categorizes packets according to the following traffic types: TT frames, asynchronous traffic and BE messages.

After identifying the incoming stream, the ingress time-based filtering function applies to TT frames. This function filters a TT frame that arrives outside the expected time window. As a next step, the sequence decoding function is invoked. This function determines whether the packet has an R-TAG or not. If the frame carries the R-TAG, it retrieves the sequence number from the last two octets of the R-TAG and then calls the sequence recovery function. In the counter case, if the frame does not contain an R-TAG, the sequence generation function is invoked. The frame without the R-TAG is sent from an FRER-unaware device. This assumption provides an opportunity to simulate a TSN network with

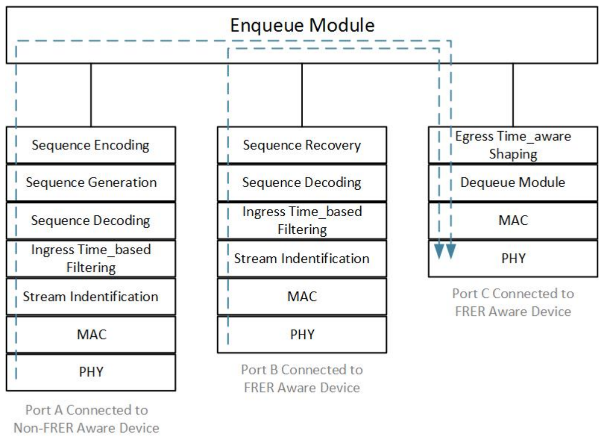

Enqueue Module
Sequence Encoding
Sequence Generation
Sequence Decoding
Ingress Time_based Filtering
Stream Indentification
MAC
PHY
Port A Connected to Non-FRER Aware Device
Sequence Recovery
Sequence Decoding
Ingress Time_based Filtering
Stream Indentification
MAC
PHY
Port B Connected to FRER Aware Device
Egress Time_aware Shaping
Dequeue Module
MAC
PHY
Port C Connected to FRER Aware Device

Fig. 3: Packet processing phases in the TSN switch model.

the combination of FRER-unaware and FRER-aware network elements.

In the stream identity table, for every stream in addition to stream parameters (e.g. VLAN ID), the last received sequence number is stored in a parameter called LastRecSeqNum. The sequence recovery function examines the sequence number of the frame against the LastRecSeqNum of the stream to which the message belongs. This function uses the VectorRecoveryAlgorithm. Therefore, if the packet's sequence number is equal to LastRecSeqNum + 1, the enqueue module is invoked. In the counter case, if the packet's sequence number does not match the expected sequence number (i.e. LastRecSeqNum + 1), the packet will be dropped.

For every stream in the stream identity table, the last generated sequence number is stored in a parameter called LastGenSeqNum. For the first frame of a specific stream, the sequence number generation function sets the LastGenSeqNum to zero. For subsequent frames, this function increments the LastGenSeqNum by one. To diminish the impact of transient faults, this function resets the LastGenSeqNum when it reachs the value 50. The sequence encoding function encodes the LastGenSeqNum of the stream into the R-TAG format. To be more specific, the function creates the R-TAG for the frame and sets the last two octets of the R-TAG to the value of the LastGenSeqNum parameter.

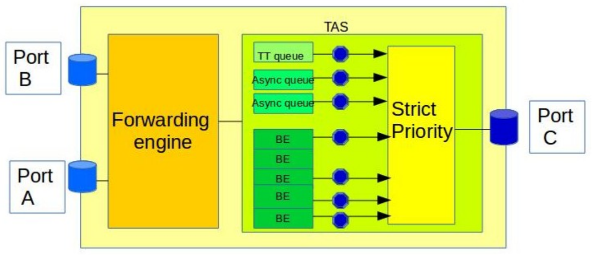


Fig. 4: The queuing scheme of TSN switch model

After encoding the sequence number, the control is passed to the enqueue module. The enqueue module enqueues the frame to the correct egress queue. The queuing scheme of a TSN switch model is presented in Figure 4. This module enqueues TT messages to the egress port's TT queue. Asynchronous messages are put into one of the asynchronous queues depending on the frame's priority. The BE messages are also put into the outgoing port's BE queues depending on the frame's priority. It is good to note that the strict priority scheme is deployed between the TT, asynchronous and BE queues. After enqueuing, the egress time-based shaping function specifies which queue is eligible to transmit the next packet according to the port-specific GCL. Finally the dequeue module transmits the packet from the chosen queue to the attached link. Figure 3 depicts the block diagram of packet processing in the TSN switch model.

C. TSN End System Model

An end-system model can be either talker or listener. Depending on the role, different flow control mechanisms are specified for the end-system model. The FRER logic of end-system and switch models is identical. On the transmission side in the end-system model, first the source module generates frames according to the outgoing traffic profiles. After that, the frame is passed to the sequence generation module to compute appropriate sequence numbers. Then the sequence encoding function adds the desired R-TAG to the frame. As a next step, the enqueue module puts the frame into a certain queue based on the traffic type. The end-system model uses the same queuing scheme as the switch model. At the final stage, the dequeue module sends out the message from the queue which is selected by TAS to the attached link.

For a frame that is received from the physical link, first the stream identification function identifies the stream that the frame belongs to using the source MAC address and the VLAN ID fields. For TT streams, the ingress time-based filtering module discards frames that are received at unscheduled time slots and forwards the rest of the frames to the sequence decoding function. The sequence decoding function retrieves the sequence number of the frame from the R-TAG. The sequence recovery function compares the sequence number of the frame against the stream's LastRecSeqNum and then discards the frame that has the wrong sequence number (i.e. not LastRecSeqNum + 1). This function uses the VectorRecoveryAlgorithm. It is good to mention that in our simulator in order to offer temporal properties, the simulation time is considered as the global clock. Therefore, the local clocks of all devices are synchronized to the global clock on a regular basis (i.e. 100 milliseconds). Figure 5 presents the ingress and egress flow control mechanisms within a TSN end-system model.

D. Fault Injector

FRER is introduced to protect the TSN systems against unwanted and faulty behaviors. To validate safety and fault tolerance capability that is offered by FRER, we developed a fault injector model. This model simulates different faults (e.g. link failure). Therefore, the fault injector provides an opportunity to investigate the behaviors of TSN capable devices in such fault states. Our fault injector emulates the following faults:

- Link Failure: This failure can be easily emulated by enabling the link failure parameter for a specific link.
- Crash Failure: The crash failure can be emulated by setting the device-specific failure/recovery attribute.
- Stuck Transmitter: This failure occurs when a sender transmits messages with the same sequence numbers instead of incremental values. To simulate this failure, the end-system model sends frames with the repetitive sequence numbers.
- Omission: The omission failure happens when a sender fails to transmit a certain frame or a specific packet is not delivered to the receiver. The sending end-system emulates this failure by transmitting frames with non consecutive sequence numbers.
- Resequencing: When the frames which belongs to a certain stream arrive at the receiver in a wrong order, the resequencing failure occurs. To simulate this failure, the sending end-system delays transmitting frames with the lower sequence numbers. Therefore, it first sends frames with the higher sequence numbers.

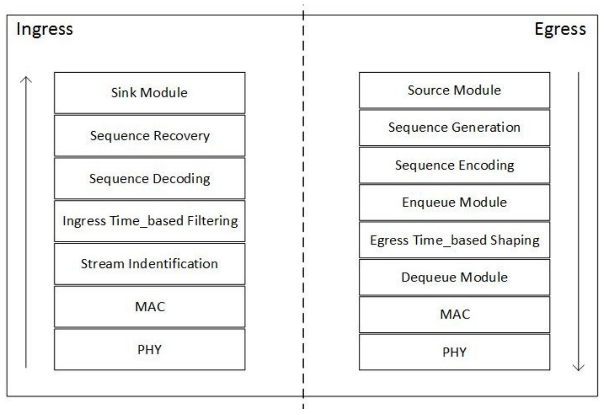


Fig. 5: Ingress and egress flow control of a TSN end-system

IV. Experiments and Evaluation

Our TSN simulator comprises devices which support time-based (e.g. IEEE 802.1Qbv and 802.1Qci) and non-time-based features (e.g. IEEE 802.1CB) of TSN and communicate with each other via full-duplex 100 Gbps physical links. This simulation executes on a PC with 32GB memory and a dual-core 2.4 GHz CPU.

A. Experimental Setup

To evaluate the impact of FRER on the reliability of mission-critical applications, we use an example layout of an Ethernet-based Train Communication Network (TCN). This network like other Ethernet-based train topologies comprises an Ethernet Train Backbone (ETB) and an Ethernet Consist Network (ECN). As shown in Figure 6, all devices in ECNs are connected to the ETB via two redundant ETB lines and also two separate ETB switches. In addition, within every consist network, ECN switches are connected to each other using a ring topology. Consequently, this layout offers redundant paths at both ECN and ETB level and meets the high safety demands of mission-critical train applications (e.g. braking system).

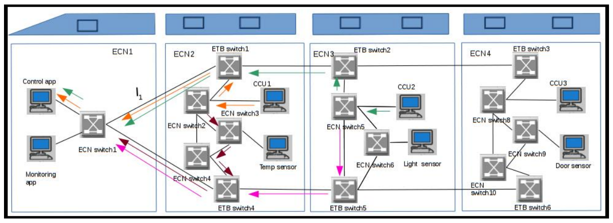


Fig. 6: Experimental network structure. The redundant paths of $s1$ are shown by orange and purple arrows while green and pink arrows illustrate the $s2$ redundant routes.

In this set up, within every consist network (except ECN1), first the sensor collects data at rate of 100 milliseconds and then sends samples to the corresponding Central Computing Unit (CCU). After that, every CCU sends sensor data to the control application and the monitoring application which resides in ECN1. Finally, the control application processes the sensor samples and sends back process messages towards each CCU. It is good to note that we assume all aforementioned streams are TT. Hence, their redundant routes and transmission schedules are allocated statically. To achieve a more realistic use case, we consider a background traffic from the monitoring application towards all CCUs. Table II details the parameters of all streams which are sent over the experimental network.

Stream	$Src \rightarrow Dst$	Period (ms)
$s1$	$CCU1 \rightarrow control\ app$	200
$s2$	$CCU2 \rightarrow control\ app$	200
$s3$	$CCU3 \rightarrow control\ app$	200
$s4$	$temp\ sensor \rightarrow CCU1$	100
$s5$	$light\ sensor \rightarrow CCU2$	100
$s6$	$door\ sensor \rightarrow CCU3$	100
$s7$	$CCU1 \rightarrow monitoring\ app$	200
$s8$	$CCU2 \rightarrow monitoring\ app$	200
$s9$	$CCU3 \rightarrow monitoring\ app$	200

TABLE II: TT Stream Specifications

B. Experiments and Results

To investigate the behavior of TSN capable models in the presence of different transient and permanent faults, we simulate the following scenarios:

1) Messages with identical R-TAG header: In the first scenario, the fault injector forces the control application to send process messages with repetitive sequence numbers towards the CCUs for specific time intervals (i.e. from 1.2 to 3 seconds). As the graph in Figure 7 depicts, CCU1 is not receiving any process message from the control application during this period of time. The FRER logic of ECN switch 1 discards all process messages with the same R-TAG header and does not permit the faulty packets consume the network resources (e.g. bandwidth and memory). Therefore, FRER protects the TSN network against this transient fault. CCU1 resumes receiving the process messages just after the control application recovers from this failure (at 3 seconds). As the behaviors of all CCUs in the described condition is identical, we just present the result for CCU1.

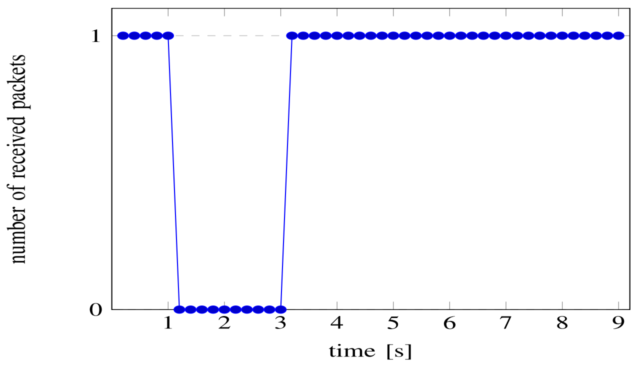


Fig. 7: The number of process packets are received by CCU1 in presence of repetition failure in the control application

2) Injecting frames with wrong sequence numbers: To emulate the omission failure, the fault injector modifies the sequence number generation function of CCU1 so that it increments the LastGenSeqNum by 3 instead of 1. Consequently, ECN switch 2 notices that some CCU1 messages are missing. Therefore, it discards all frames that originated from CCU1 in order to mitigate this faulty behavior. As graph in Figure 8 shows, the control application and the monitoring application are not receiving any message from CCU1 just after the fault injected to the network (i.e. at 1.2 seconds). CCU1 recovers from this failure at 3 seconds and starts to send frames with the consecutive sequence number. ECN switch 2 continues discarding messages that are sent from CCU1, because in ECN switch 2, the LastRecSeqNum of CCU1 stream is not aligned with the R-TAG header of messages sent from CCU1. ECN switch3 resumes accepting CCU1 frames at 7.2 seconds. The reason is that the CCU1 messages start carrying the expected sequence numbers due to reseting the LastGenSeqNum parameter in CCU1.

The fault injector also modifies the sequence number generation function of CCU2 so that for every two consecutive frames first it increments the LastGenSeqNum by 2 and then decrements this parameter by 1. Due to this modification which simulates a resequencing failure, ECN switch 5 first receives a CCU2 frame with higher sequence number and then a message with lower sequence number. Hence, as shown in Figure 9, ECN switch 5 does not forward any CCU2 message to the control application and the monitoring application just after noticing this faulty behavior (i.e. at 1.2 seconds). In both aforementioned scenarios, the first switch on the path to the control application (i.e. ECN switch 2 and ECN switch 5 respectively), discards faulty frames.

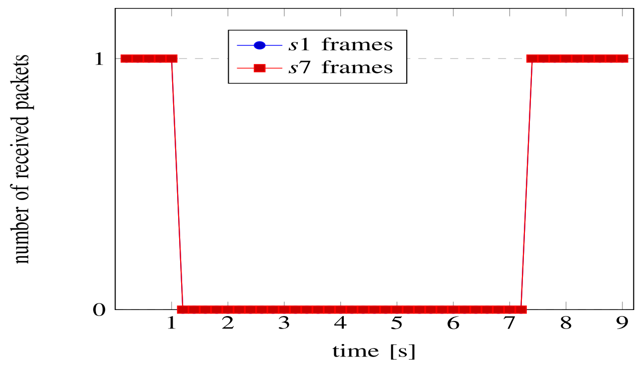


Fig. 8: The number of $s1$ and $s7$ packets received by the control and monitoring applications in case of Omission failure

Therefore, FRER in addition to enhancing the network fault tolerance, improves the overall network resource utilization. To achieve this, FRER eliminates the probability of forwarding the faulty packets over the TSN network.

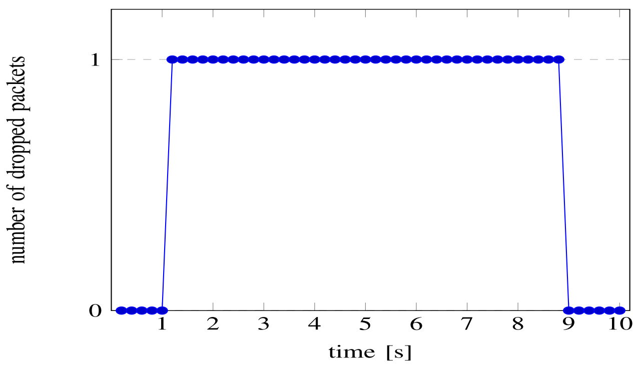


Fig. 9: The number of $s2$ and $s8$ packets are discarded by ECN switch 5 in case of Resequencing failure in CCU2

3) Testing FRER against link failures: As described before, FRER offers safety and fault-tolerance capability using redundant paths. Therefore, we set up redundant paths for $s1$ and $s2$ before running simulations. We also disable RSTP over our simulated network to allow forwarding frames from two redundant routes simultaneously.

The fault injector at 1.2 second makes $l1$ fail. Before the occurrence of the $l1$ failure, ECN switch 1 receives $s1$ and $s2$ frames from two separate paths (which are illustrated in Figure 6). Then it forwards the frames which arrive first and eliminates the duplicated messages. As the graph in Figure 10.a shows, the control and the monitoring applications do not experience any traffic loss from $s1$ and $s2$. After the $l1$ failure, ECN switch 1 still receives $s1$ and $s2$ frames from the redundant paths and delivers them to the control and monitoring applications. These applications are not receiving any $s3$ messages after the $l1$ breakdown. The reason is that we do not set up redundant paths for the $s3$ stream. Hence, after the failure in the primary route, $s3$ frames are not delivered to the control and monitoring applications anymore.

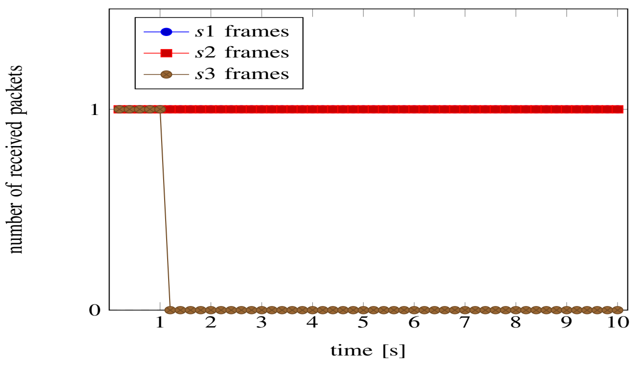
It is good to note that the redundant routes of $s1$ comprises different numbers of links. Consequently, as graph in Figure 10.b presents, the end-to-end delay of $s1$ has different values before and after the $l1$ failure. However, the $s2$ end-to-end latency remains unchanged during the simulation. The redundant paths of $s2$ unlike the $s1$ routes have the same number of physical links. In our simulator, every frame remains in the TSN switch for 2 μs (i.e. $D_{\text{processing}}$). In addition, the size (s_m) of all frames which are sent over our network is set to 64 bytes and the bandwidth (b_l) of all links is set to 100 *Gbps*. Hence, the frame's transmission delay (D_t) within the TSN switch is calculated as follows:

$$D_t = s_m/b_l = (64bytes)/(100Gbps) = 5.12ns$$

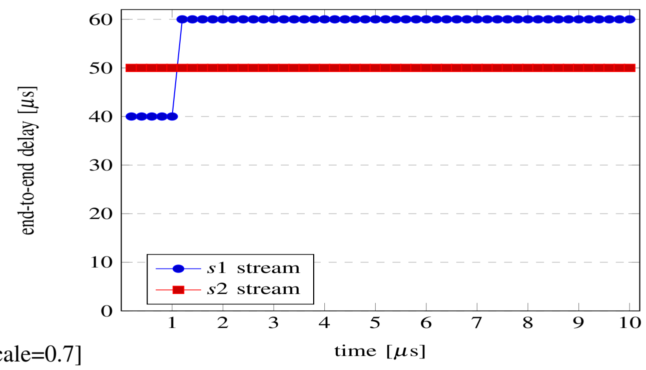
As the propagation delay ($D_{\text{propagation}}$) of each link is set to 8 μs, the frame's end-to-end delay considering chosen route is calculated as follows:

$$D_{\text{e2e}} = num.hops * (D_{\text{processing}} + D_t + D_{\text{propagation}})$$

For instance, D_{e2e} of $s1$ before and after the $l1$ failure is computed as follows:



[yscale=0.7]



(a) The number of $s1$, $s2$ and $s3$ packets are received by the control application in case of l_1 failure

(b) The end-to-end delay of $s1$ and $s2$ streams in presence of $l1$ failure

Fig. 10: The reliability of $s1$, $s2$ and $s3$ streams in case of $l1$ failure

$$Before failure: D_{\mathrm{e2e}} = 4 * (2\mu s + 5.12ns + 8\mu s) = 40.02\mu s$$

$$After failure: D_{\mathrm{e2e}} = 6 * (2\mu s + 5.12ns + 8\mu s) = 60.03\mu s$$

4) Testing FRER against crash failures: To evaluate the impact of crash failures, the fault injector sets the failure attribute of ETB switch 2. Consequently, ETB switch 2 stops forwarding frames to neighbor switches. After ETB switch 2 crash (i.e. at 1.2 seconds), the control and monitoring applications do not receive $s3$ anymore, because the only route of $s3$ passes through ETB switch 2. However, these applications continue receiving $s2$ from the redundant route which does not pass through the ETB switch 2. The crash failure results are presented in Figure 11.

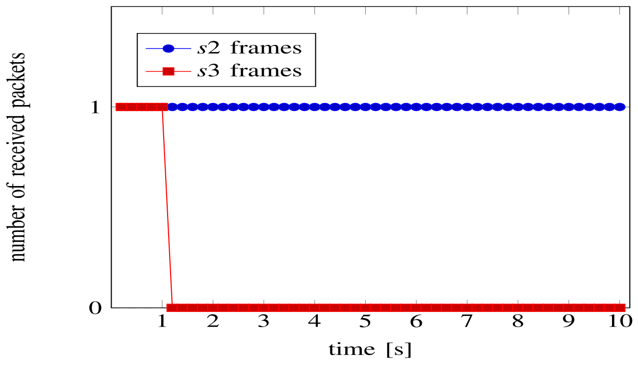


Fig. 11: The number of $s2$ and $s3$ packets are received by the control application in case of ETB switch 2 crash

It is good to mention that FRER cannot protect TSN networks from crash failures of all devices. For instance, when the fault injector sets the ECN switch 1 to a crash failure, no TT stream can flow between ECN1 and the other ECNs. ECN switch 1 is a single point of communication between the control application and the other devices in our simulator.

V. CONCLUSION

In this paper, we extended the simulation models of TSN time-based features to support fault-tolerant communication. To be more specific, we introduced the FRER procedure in our simulator to evaluate reliable TSN networks. The implementation of the FRER logic is done in a modular way and is integrated with the existing TSN models without significant modifications.

These TSN capable models are used to evaluate and validate the fault-tolerance capability of TSN devices in the absence of real TSN hardware with FRER logic. To achieve a comprehensive evaluation, we enforce different faults on a highly redundant train network structure and evaluate the reliability (e.g. packet loss) in different fault conditions. The experimental results verify that the IEEE 802.1CB standard protects TSN systems against transient errors (e.g. stuck transmitter, resequencing) and offers bounded end-to-end delay and zero packet loss in case of permanent errors (e.g. link failure, node crash).

ACKNOWLEDGMENT

This work was done as part of Safe4RAIL Project, Grant Agreement No. 730830.

REFERENCES

[1] “Institute of Electrical and Electronics Engineers, Time-Sensitive Networking,” in *Time-Sensitive Networking Task Group. [Online]. Available: http://www.ieee802.org/1/pages/tsn.html*, IEEE, 2017.

[2] “Institute of Electrical and Electronics Engineers, Inc. 802.1Qbv - Enhancements for Scheduled Traffic,” in *Time-Sensitive Networking Task Group. [Online]. Available: http://www.ieee802.org/1/pages/802.1bv.html*, IEEE, 2016.

[3] “Institute of Electrical and Electronics Engineers, Inc. 802.1AS-Rev - Timing and Synchronization for Time-Sensitive Applications,” in *Time-Sensitive Networking Task Group. [Online]. Available: http://www.ieee802.org/1/pages/802.1AS-rev.html*, IEEE, 2017.

[4] “IEEE Std 802.1D, IEEE Standard for Local and metropolitan area networks Media Access Control (MAC) Bridges,” IEEE, 2004.

[5] “IEC 62439-3 Ed.03, Industrial communication networks - High availability automation networks - Part 3: Parallel Redundancy Protocol (PRP) and High-availability Seamless Redundancy (HSR),” IEC, 2016.

[6] S. A. Nsaif and J. M. Rhee, “Seamless ethernet approach,” in *Consumer Electronics (ICCE), 2016 IEEE International Conference on*, pp. 385–388, IEEE, 2016.

[7] “Institute of Electrical and Electronics Engineers, Inc. 802.1CB - Frame Replication and Elimination for Reliability,” in *Time-Sensitive Networking Task Group. [Online]. Available: http://www.ieee802.org/1/files/private/cb-drafts/d2/802-1CB-D2-9.pdf*, IEEE, 2017.

[8] M. Pahlevan and R. Obermaisser, “Evaluation of time-triggered traffic in time-sensitive networks using the opnet simulation framework,” in *Parallel, Distributed and Network-based Processing (PDP), 2018 26th Euromicro International Conference on*, pp. 283–287, IEEE, 2018.

[9] P. Heise, F. Geyer, and R. Obermaisser, “Tsimnet: An industrial time sensitive networking simulation framework based on omnet++,” in *New Technologies, Mobility and Security (NTMS), 2016 8th IFIP International Conference on*, pp. 1–5, IEEE, 2016.

[10] “Institute of Electrical and Electronics Engineers, Inc. 802.1Qca - Path Control and Reservation,” in *Time-Sensitive Networking Task Group. http://http://www.ieee802.org/1/files/private/ca-drafts/d2/802-1Qca-d2-1.pdf*, IEEE, 2015.

[11] “Introduction to Riverbed Modeler Academic Edition,” in *https:splash.riverbed.com/docs/DOC-4833*, 2018.