Fault-Tolerant Cyclic Queuing and Forwarding in Time-Sensitive Networking

Liwei Zhang[1], Tong Zhang[3], Wenxue Wu[1], Xiaoqin Feng[1], Guoxi Lin[1] and Fengyuan Ren✉ [1,2]

[1]Lanzhou University, China [2]Tsinghua University, China
[3]Nanjing University of Aeronautics and Astronautics, China

***Abstract*—Time-sensitive networking (TSN) provides deterministic time-sensitive transmission services for critical data at the link layer. Cyclic Queuing and Forwarding (CQF) defined by IEEE 802.1Qch is used for critical data transmission. However, unexpected data errors may occur due to transient faults like electromagnetic interference. At present, the solution to such faults defined in the IEEE TSN standards is to transmit multiple data copies on redundant paths, which introduces network resources wastage. Compared to redundant transmission, retransmission can reduce resource waste, but may violate the deterministic transmission guarantee in TSN. To tackle with this issue, we propose a time-redundant fault-tolerant mechanism for CQF, called fault-tolerant CQF (FT-CQF). On the basis of standard CQF, FT-CQF occupies an additional queue to cache copies of Time-Trigger (TT) flows and reserves time slots to forward them. According to the returned CRC-related messages, FT-CQF will decide whether to forward these copies. Non-TT flows can also be transmitted during this time when copies are not required to be forwarded. We implement FT-CQF in OMNeT++, and verify the performance of FT-CQF in typical network scenarios. The extensive simulation experiments show that FT-CQF is effective in terms of fault-tolerant effects, consumed resources, delay, and jitter.**

***Index Terms*—Time-sensitive networking, Cyclic Queuing and Forwarding, Fault-Tolerant**

I. Introduction

Hard real-time and safety-critical control applications in industrial automation networks require deterministic time-sensitive services [1]. However, both time-triggered (TT) and non-time-triggered (non-TT) flows are transmitted in the same network [2], which may violate the deterministic requirements of TT flows. Time-Sensitive Networking (TSN) developed by IEEE 802.1 Working Group is an effective solution to deterministic transmission of TT flows [3] [4]. TSN allows TT flows and non-TT flows to be transmitted in the same network, and manages different flows through a series of traffic management schemes to guarantee the deterministic communication of TT flows with bounded delay and extremely low jitter.

Time-Aware Shaper (TAS) [5] and Cyclic Queuing and Forwarding (CQF) [6] are two class-granularity traffic shaping mechanisms in TSN. TAS needs to plan the configuration of Gate Control List (GCL) on each queue in a switch to schedule TT flows. However, with changes in traffic characteristics and flow addition or deletion, the configuration cost of GCL would increase significantly. Alternatively, CQF is considered as an easy-to-use shaper. By cyclically switching the Ping-Pong queues, CQF guarantees hop-by-hop bounded delay.

When TSN messages are being transmitted, data errors may occur due to the influence of electromagnetic interference (EMI). The common sources of such interference include electrical equipment like radios, relays, and lightning in the environment. Moreover, it is difficult to completely eliminate the impact of EMI [7]. Therefore, the ability to tolerate such faults in TSN is very important. Recently, there has been a lot of research [8]–[10] on TSN fault-tolerant transmission. Most of them focus on spatial redundancy. Frame Replication and Elimination for Reliability (FRER) is a fault-tolerant solution defined by IEEE 802.1CB [11]. Although FRER is able to tolerate incorrect messages by sending copies of each frame on multiple disjoint paths, it can waste a lot of network resources. Additionally, configurations of multiple frame copies are complex. Unlike spatial redundancy, temporal redundancy is more cost-effective and can tolerate frame errors caused by transient faults through retransmission.

The receiver can detect frame errors by cyclic redundancy check (CRC) [12], and then notify the sender to retransmit the erroneous message. Based on this idea, Dobrin et al. proposed a fault-tolerant scheduling method for TAS to guarantee that high critical flows meet deadlines [7]. It sets priorities for each flow, but there are just eight priority queues on each port in TSN, far less than the number of flows. On the other hand, error occurance is uncertain. Therefore, the time to perform retransmission is also random, which will violate the deterministic transmission of TT flows. In addition, CRC-related messages may also have errors during transmission. Feng et al. considered the above issues in the offline fault-tolerant scheduling of TAS [13]. It formulated the constraints for TT flows, CRC-related messages, and retransmissions, and the reserved time slots for retransmissions can be used to transmit non-TT flows when no retransmissions are needed. However, the added constraints lead to higher configuration costs of GCL in TAS. Compared with TAS, CQF greatly simplifies the configuration complexity of GCL. CQF can guarantee hop-by-hop deterministic transmission and non-TT flows can be transmitted in the time slots that are not used up by TT flows. Based on the transmission characteristics of CQF, retransmission at each hop enables CQF to tolerate transient faults. To support retransmission at each hop in CQF-based TSN, TT flows can be retransmitted by occupying the

✉Corresponding author: Fengyuan Ren, rfy@lzu.edu.cn

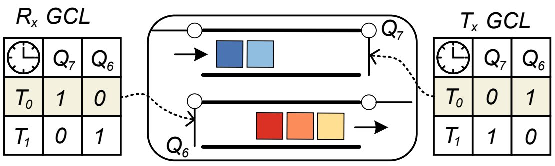


Fig. 1. Enqueue and dequeue operations in CQF.

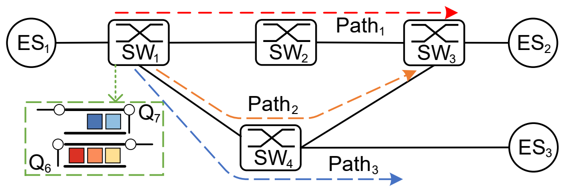


Fig. 2. An example of TSN reliability statement.

transmission resources of non-TT flows.

Based on the above investigations, we enhance CQF to tolerate fault caused by sporadic interference in communication and name enhanced CQF after FT-CQF. FT-CQF can enforce retransmission of incorrect frames notified by negative acknowledgement by reserving time slots at each hop. To support this fault-tolerant transmission, we propose the following three design concepts. Firstly, the double queues serving TT flows in the standard CQF are extended to three queues. The added queue is used to cache copies of TT flows. Secondly, the global time is divided into time slots of equal length, and the minimal time slot can satisfy the retransmission requirements. Lastly, the ingress port of the node configures the frame detection component *Checker*. *Checker* can notify the upstream node of the detection result of the TT frames and maintain that the original transmission sequence of TT flows will not be violated due to retransmission. The forwarding scheme of FT-CQF can enhance the reliability of the transmission between adjacent nodes and provide bounded delay and jitter for frames.

II. Background

A. CQF in TSN

CQF divides the global time into time slots of equal length. Two queues are used to perform enqueue and dequeue operations in a cyclic manner under the control of R_x GCL and T_x GCL, as shown in Fig. 1. where T_i (i = 0, 1) represents time slot i. The frame transmission follows two rules:

- The transmission of frames between two adjacent switches must be completed within the same time slot.
- If a switch receives a frame in a time slot, the frame must be sent in the next time slot to the next hop.

Thus, the end-to-end delay of a TT frame only depends on the time slot size and path length. Additionally, Non-TT flows can be transmitted in the time slots that are not used up by TT flows. As shown in Fig. 1, all TT frames in Q_6 are forwarded during T_0, during which time Q_7 is receiving TT frames. When T_1 starts, all TT frames in Q_7 are forwarded, and Q_6 receives TT frames. Non-TT flows are transmitted using the remaining time slots in T_0 and T_1.

B. Reliability in TSN

FRER has two main mechanisms: (i) Replication of flows via different paths at the source node and (ii) elimination of redundant frames per flows at the relay nodes or the destination node. At present, CQF-based TSN mainly adopts FRER to tolerate transient faults, which is called FRER-CQF. As shown in Fig. 2, FRER-CQF can guarantee the reliability of communication between ES_1 and ES_2 because there are two disjoint paths $Path_1$ and $Path_2$. Although they can tolerate the frame errors, they still have shortcomings. Specifically, we summarize the following two shortcomings of FRER-CQF in TSN.

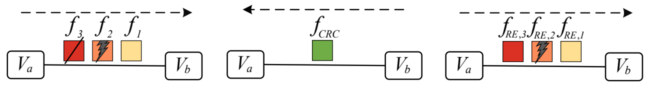
Waste of network resources: FRER-CQF needs to transmit each copy of the frame on disjoint paths in the network, which will occupy much more resources, resulting in wastage. As shown in Fig. 2, in order to ensure the reliability of communication between ES_1 and ES_2, FRER-CQF will inevitably occupy network resources on $Path_3$. If the frame does not fail on $Path_1$, the resources occupied by the copy of the frame on $Path_2$ will be wasted.

Complexity of configuration: FRER-CQF needs to configure multiple disjoint paths in the network, which will increase the complexity of configuration. In addition, there may be many network topologies that contain intersecting nodes in TSN, such as linear topology. Configuring FRER-CQF on this type of topology cannot achieve good fault-tolerance performance. As shown in Fig. 2, there is no other path between ES_1 and ES_3 that does not intersect with $Path_3$. Thus, it cannot to directly use FRER-CQF to guarantee transmission reliability.

III. Related Work and Motivation

Spatial redundancy mainly guarantees reliability through redundant space resources. IEEE 802.1 CB is a reliability scheme for spatial redundancy. An enhanced version of IEEE 802.1 CB was proposed in [14] and implemented in [15], which can prevent unintended frame eliminations independent from the deployment scenario. Gavrilut et al. proposed a fault-tolerant network topology, consisting of redundant physical links and bridges [16]. Similarly, a network topology that supports seamless redundant transmission for TT flows is proposed in [17]. However, these methods rely on multiple routes to transmit data to the destination even if no faults are identified.

Temporal redundancy improves reliability by repeatedly executing system processes where errors occur. To improve reliability through the temporal redundancy of TSN, Alvarez et al. proposed a mechanism to transmit multiple copies of frames to tolerate transient faults in data transmission [18]. Run-time recovery for TT flows is explored by Integer Linear Programming (ILP), and a heuristic algorithm is proposed in [19]. Although it provides better protection against multi-point failures than static redundancy, the online cost also jeopardizes time constraints. Thus, offline fault-tolerance schedulings are proposed. Zhou et al. performed scheduling and routing with a SMT-based and heuristic technique that considers both reliability constraints and end-to-end deadline constraints [20]. How-



(a) TT flows transmitted during t_1.

(b) CRC-related message transmitted during t_2.

(c) Copies of the TT flows transmitted during t_3.

Fig. 3. Schematic diagram of transmission for TT flows through the physical link [V_a, V_b].

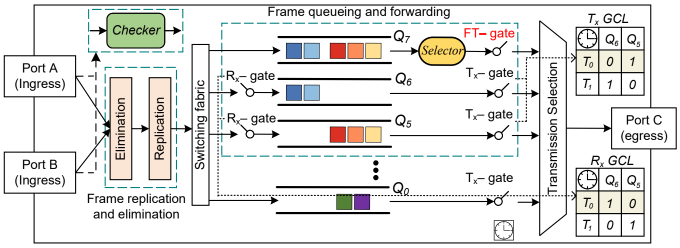


Fig. 4. Framework of switch.

ever, such pro-active mechanisms still waste network resources. Dobrin et al. proposed a fault-tolerant scheduling method CRC-based retransmission, which first assigns a virtual deadline to each flow, and then sets different priorities for each message according to the earliest deadline scheduling [7]. However, they did not consider the case of CRC-related messages errors, and the scheduling of the entire network. On the other hand, there are no formulas or experiments to demonstrate the efficiency of their method. Feng et al. proposed an offline fault-tolerant scheduling algorithm for TAS [13], which considered CRC-related messages and queue assignment of flows. The transmission window of the non-TT flow is computed after the TT flow is scheduled. The scheduled transmission windows for retransmissions can be used to transmit non-TT flow when no retransmissions are needed. However, their method increases the configuration cost of GCL.

The above related works show that spatial redundancy leads to high costs due to the need for additional hardware, and it is more suitable for tolerating permanent faults. However, transient faults in links are more likely to occur than permanent faults [21]. Therefore, using only spatial redundancy to tolerate transient faults may result in a significant increase in cost. Instead, temporal redundancy can tolerate transient faults, which is more cost-effective than spatial redundancy. At present, the TSN standard does not provide a temporal redundancy mechanism to tolerate transient faults, and CQF-based TSN mainly tolerates transient faults through spatial redundancy. For these reasons, we explore the application scheme of temporal redundancy mechanisms in CQF-based TSN.

IV. Fault-Tolerant CQF

A. Basic Idea

We use an example to explain how TT flows are transmitted over a physical link with transient faults between adjacent nodes. As shown in Fig. 3, the time slot is divided into three blocks: t_1, t_2 and t_3. t_1, t_2 and t_3 are used to transmit TT frames f_i $(i = 1, 2, 3)$, CRC related messages f_{CRC} and frame copies $f_{RE,i}$ of f_i respectively. In Fig. 3(a), the sender V_a transmits f_1, f_2, and f_3 to the receiver V_b. V_b drops the failed frame f_2 during t_1, and f_3 that arrives after f_2, which did not fail, is also dropped. After V_b receives all TT frames forwarded by V_a, V_b sends f_{CRC} to V_a, as shown in Fig. 3(b). If the value of f_{CRC} is *False*, or f_{CRC} suffers a transient fault, then V_a will perform retransmission. According to Fig. 3(a), f_{CRC} will carry *False* to V_a. As shown in Fig. 3(c), V_a immediately transmits $f_{RE,1}$, $f_{RE,2}$ and $f_{RE,3}$ to V_b after receiving f_{CRC}. V_b will receive $f_{RE,1}$, $f_{RE,2}$ and $f_{RE,3}$ in sequence during t_3. If $f_{RE,2}$ suffers a transient fault, then $f_{RE,2}$ will be dropped and $f_{RE,3}$, which did not fail, will be received. In addition, V_b also eliminates received redundant frame copies.

B. Fault Assumption

Based on duration, faults can be divided into permanent, transient, and intermittent (periodic) faults [22]. Permanent faults can persist for a considerable period of time. Transient faults appear in a short time and then disappear. Intermittent faults appear, disappear, and reappear continuously. If a frame suffers a transient fault during transmission, any bit of the frame may be changed unexpectedly [7] [23]. In this paper, we only focus on transient faults and retransmit frames only once per hop [13]. The reasons for single-retransmission are as follows: (1) Multiple retransmissions require more resources and can damage the schedulability of TT flows; (2) Multiple retransmissions will increase the design complexity and cost of the switch. Based on the above discussion, the sender does not perform the third forwarding regardless of whether the second forwarded frame is received correctly.

V. Designing FT-CQF

A. Framework of FT-CQF

As shown in Fig. 4, FT-CQF has three components: *Checker*, frame queueing and forwarding, and frame replication and elimination.

Checker**.** *Checker* verifies the correctness of the received frame and send f_{CRC} to the upstream node. Furthermore, *Checker* maintains that the original transmission sequence of TT flows is not destroyed.

Frame queueing and forwarding. FT-CQF will queue and forward TT flows. Additionally, FT-CQF will also cache copies of TT flows and decide to forward or remove these cached copies based on f_{CRC}.

Frame replication and elimination. To support retransmission at each hop, nodes in the network need to have the ability to replicate and eliminate frames. In view of this, We use frame replication and elimination defined in IEEE 802.1CB [11], which also introduces related technical principles.

Through the above component functions, the reliability will be enhanced and is not limited by network topologies. The technical details will be introduced in the following sections.

B. FT-CQF

1) Checker: *Checker* is implemented based on CRC. As shown in Fig. 3(a) and Fig. 3(c), the process of *Checker* verifying the received TT frames in each time slot is divided into two stages.

Checker will first verify the frames forwarded from Q_5 or Q_6 during t_1. During this period, if an erroneous frame is found, the erroneous frame and all subsequent frames forwarded from Q_5 or Q_6 will be dropped, and f_{CRC} with the value *False* will be sent to the upstream node at the end of t_1. t_1 is equal to $(\frac{Q_s \times MTU}{B} + D_h)$, where Q_s is the maximum amount of frames that each queue serving the original TT frames can hold, *MTU* is the maximum frame length, *B* is the link bandwidth, and D_h is the internal processing delay and propagation delay in a switch. If the frame forwarded during t_1 did not fail, then f_{CRC} with the value *True* will be sent to the upstream node. To return f_{CRC} in time, the priority of f_{CRC} needs to be planned to be second only to the TT flow, so that f_{CRC} can be scheduled from the highest priority queue except the queue serving TT flows. *Checker* will verify the frames forwarded from Q_7 during t_3. During this period, if an erroneous frame is found, only the erroneous frame will be dropped, and no more f_{CRC} will be sent to the upstream node at the end of t_3.

2) Frame Queueing and Forwarding: Considering that flows are divided into eight priorities in TSN, FT-CQF also follows the eight-queue design of standard CQF, in which two queues called CQF queues, Q_5 and Q_6, are used for TT flow queuing and forwarding. On this basis, an additional fault-tolerant queue Q_7 is introduced to cache and schedule copies of TT flows. As shown in Fig. 4, enqueue and dequeue operations are performed in Q_5 and Q_6 alternatively in a cyclic manner under the control of R_x GCL and T_x GCL and the frames in Q_5 and Q_6 are scheduled first. Q_7 performs dequeue operation under the control of *Selector* and the fault-tolerant gate (FT-gate). Q_7 does not receive frames from the input port, but only TT frames that are replicated inside the switch. If Q_5 or Q_6 drop frames because the queue is full, Q_7 will also drop copies of frames dropped by Q_5 or Q_6. To cache copies of all TT frames in the CQF queue within a time slot, the length of Q_7 is equal to the maximum length of Q_5 plus the maximum length of Q_6. The reason is that when one queue of CQF is performing a dequeue operation, the other queue is performing an enqueue operation. Additionally, Q_7 can also remove the frames that do not need to be retransmitted. Specifically, when f_{CRC} with the value *True* is received, Q_7 first records the first frame f_1 in the head of Q_7, and then removes the frames that have the same type as f_1 from the head of Q_7 in sequence. If a frame of different type than f_1 is encountered, stop the removal and wait for the next operation. The types of frames in Q_7 are divided according to whether the original TT frame enters Q_5 or Q_6. As shown in Fig. 4, when f_{CRC} with *True* is received, the three frames at the head of Q_7 are removed.

Selector is used to select frames that need to be retransmitted. The reason is that Q_7 caches copies of TT frames in Q_5 and Q_6 within a time slot. *Selector* will obtain the first frame f_1 in the head of Q_7 at the moment when f_{CRC} is sent in each time slot, and then selects the frames that have the same type as f_1 from the head of Q_7 to dequeue. If a frame of different type than f_1 is encountered, stop the selection and wait for the next operation. As shown in Fig. 4, the three frames at the head of Q_7 are copies of the TT frames in Q_5. When all the TT frames in Q_5 need to be retransmitted due to a fault, *Selector* selects the three frames at the head of Q_7 to dequeue.

FT-gate is closed at the start of each time slot and opened or closed according to the received f_{CRC}. The specific implementation of opening and closing FT-gate according to f_{CRC} includes the following three points: (1) FT-gate will be opened if f_{CRC} with the value *False* is received from the downstream node or an error occurs in f_{CRC} transmission. (2) If f_{CRC} is not received within the specific period, FT-gate will also be opened. The specific duration is t_{CRC} in Eq. (2). This value is slightly larger than the time it takes for the downstream node to transmit f_{CRC} to the upstream node, and it is calculated from the moment f_{CRC} is sent within each time slot. (3) FT-gate will remain closed if f_{CRC} with the value *True* is received. Moreover, if the FT-gate is opened, it will be closed at the end of the current time slot. The above operation will be carried out again after the start of the next time slot.

C. Time Slot Size

Base on the above design, the lower and upper bounds of the time slot size are given in this subsection. The maximum time slot is the greatest common divisor (GCD) of the periods of all flows, which is shown in Eq. (1). The minimum time slot needs to guarantee the scenario of three transmissions in Fig. 3, which forwards the most TT frames at one time. Therefore, the minimal time slot is expressed by Eq. (2).

$$T_c^{max} = GCD(F_{TT}.periods) \tag{1}$$

$$T_c^{min} = 2 \times (\frac{Q_s \times MTU}{B} + D_h) + t_{CRC} + sync_{prec} \tag{2}$$

where $t_{CRC} = \frac{MFL}{B} + C_{delay} + t_s$, F_{TT} is the set of TT flows, $sync_{prec}$ is the clock synchronization precision, and *MFL* is the minimum frame length, which is 64 bytes. C_{delay} is the internal processing delay and propagation delay of f_{CRC} in a switch, and t_s is the safety margin of the time spent by f_{CRC}. In practice, T_c^{min} is much smaller than T_c^{max}.

VI. Evaluation

A. Simulation Configuration

We implemented FT-CQF and FRER-CQF in OMNeT++ 6.0.1 [24] based on the INET 4.4.1 framework [25], and built a TSN transmission scenario carrying deterministic traffic queuing and forwarding on the OMNeT++ 6.0.1 simulation platform. We used different configurations to test the performance of FT-CQF, taking FRER-CQF as a reference.

1) Topology: The simulation topology is shown in Fig. 5. It consists of two end systems and eight switches. There are two disjoint paths between ES_1 and ES_2, so FRER-CQF can be configured for communication between ES_1 and ES_2. Additionally, the link between ES_1 and SW_1 and the link between SW_2 and ES_2 are not faulty.

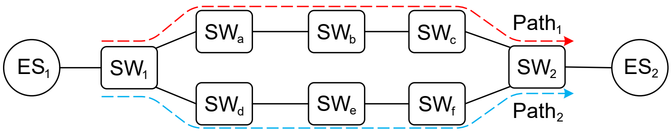


Fig. 5. Network topology.

2) Network Setting: The related network parameters that need to be set in our experiment are link bandwidth, queue length, and global time slot size. The queue length in a switch is equal to the number of frame buffers in the buffer pool. Here, we set the link bandwidth to 1000Mb/s, the length of each CQF queue in both FT-CQF and FRER-CQF to 5, the length of the fault-tolerant queue in FT-CQF to 10, and the global time slot size of FT-CQF and FRER-CQF to $125\mu s$. There are two reasons for the CQF queue length setting. First, a time-sensitive frame is not allowed to be cached in a switch for a long time to provide a deterministic forwarding service. Second, the cache resources of TSN switches are limited.

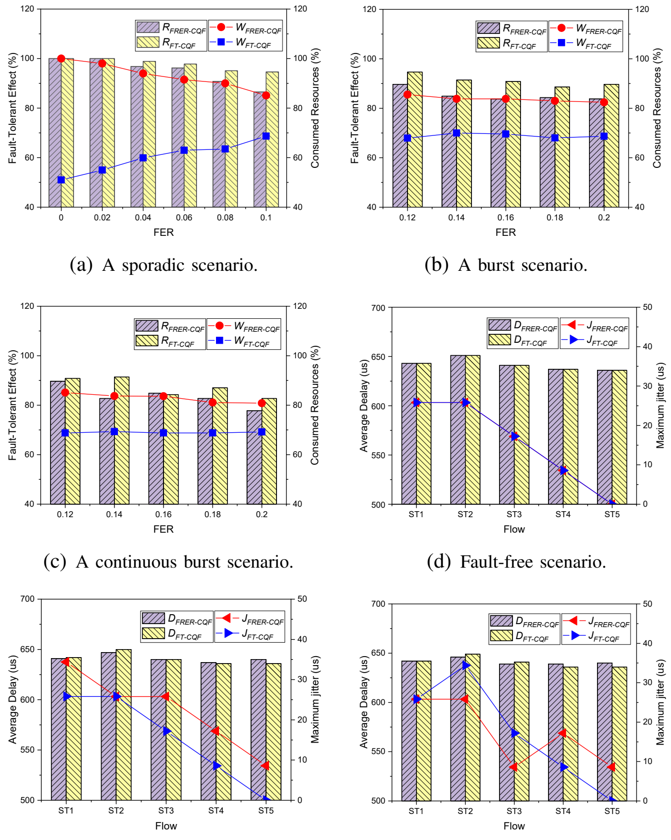
3) Traffic setting: We refer to the traffic characteristics described in IEC/IEEE 60802 standard for industrial automation networks [26]. In our experiment, flow are configured according to this standard. The payload of each flow is 1000 bytes, and the period is selected in $\{2, 4, 6, 8\}\, ms$.

In the simulation, the fault-tolerant effect, resource consumption, delay, and jitter of FT-CQF are evaluated and compared with those of FRER-CQF. The fault-tolerant effect is reflected in the ratio of the number of frames received by the destination to the number of frames sent by the source within a period of time in a specific fault scenario. The resource consumption rate is the total number of bytes of TT and their related flows in the network divided by the total number of bytes of TT and their related flows forwarded by FRER-CQF without faults scenario, where TT and their related flows represent TT flows, CRC related messages and copies of TT flows. The total number of bytes of TT and their related flows is the sum of the number of bytes of TT and their related flows transmitted by links between all adjacent nodes in the network. We counted the resource consumption rate produced by FT-CQF and FRER-CQF over a period of time in each scenario.

We describe fault scenarios in terms of Frame Error Rate (FER). Fault scenarios are divided into sporadic, burst, and continuous burst scenarios. The sporadic fault scenario is where the FER of the links of all adjacent nodes in the network is low. The burst fault scenario is where the FER of one link is high and the FER of other links is low. The continuous burst fault scenario is where the FER of two adjacent links is high and the FER of other links is low. Among them, links with low FER are called sporadic links. When FER exceeds 0.1, it means FER is high. In the experiment, the FER of the sporadic link in the burst fault scenario and the continuous burst fault scenario was fixed at 0.1.

B. Experimental Results

1) Fault-Tolerant Effect: Fig. 6(a), 6(b) and 6(c) show that the fault-tolerant effects $R_{FRER\text{-}CQF}$ and $R_{FT\text{-}CQF}$ in three different fault scenarios decrease with increasing FER. The reason is that an increase in FER causes more frames to be dropped. Therefore, the number of frames received by the destination will decrease. However, this experiment shows that FT-CQF has a stronger ability to resist transient faults, and the value of $R_{FT\text{-}CQF}$ minus $R_{FRER\text{-}CQF}$ reaches a maximum of 9%. The specific reasons are as follows. FT-CQF supports retransmission at each hop, that is, TT flows can be transmitted twice at each hop. Although FRER-CQF transmits TT flows on two disjoint paths, it only transmits the TT flow once at each hop. In the network topology shown in Fig. 5, FRER-CQF only tolerates one fault, while FT-CQF can tolerate multiple faults. Additionally, the burst scenario and the continuous burst scenario will have a greater impact on FT-CQF and FRER-CQF. However, FT-CQF exhibits stronger robustness.



(a) A sporadic scenario. (b) A burst scenario. (c) A continuous burst scenario. (d) Fault-free scenario. (e) A sporadic scenario. (f) A burst scenario.

Fig. 6. Evaluation results.

2) Consumed Resources: Fig. 6(a), 6(b) and 6(c) show the resource consumption rates $W_{FRER\text{-}CQF}$ and $W_{FT\text{-}CQF}$ in three different fault scenarios. Compared with FRER-CQF, the maximum resources saved by FT-CQF when transmitting TT and their related flows in sporadic, burst, and continuous burst scenarios are 49.02%, 17.6%, and 16.31%, respectively. In Fig. 6(a), $W_{FRER\text{-}CQF}$ decreases with the increasing FER, while $W_{FT\text{-}CQF}$ shows an opposite trend. The reason is that FRER-CQF will drop incorrect frames, whereas FT-CQF will perform retransmissions based on faults. As more frames are dropped, the resources consumed by FRER-CQF will decrease. The increase in FER will cause FT-CQF to increase the probability of retransmission at each hop, and the retransmitted data will increase resource consumption. In Fig. 6(b) and Fig. 6(c), $W_{FRER\text{-}CQF}$ and $W_{FT\text{-}CQF}$ remain in a constant range. The reason is that burst faults only affect a few links, and the number of lost frames and retransmissions is relatively fixed.

3) Average Delay: Fig. 6(d) shows that the average delay $D_{FRER\text{-}CQF}$ and $D_{FT\text{-}CQF}$ are the same. The reason is that the forwarding rules of FT-CQF and FRER-CQF are the same in the fault-free scenario, and the time slot size and path length of the flow are also the same. Fig. 6(e) and 6(f) show that the average delays $D_{FRER\text{-}CQF}$ and $D_{FT\text{-}CQF}$ are slightly different. The retransmission of FT-CQF is completed within the time slot where the erroneous frame is located, which does not increase the delay. However, the fault-tolerant performance of FRER-CQF and FT-CQF is different, which will lead to a different number of frames received by the destination. Therefore, there will be a slight difference in the average delay between FRER-CQF and FT-CQF.

4) Maximum Jitter: Fig. 6(d) show that the maximum jitter $J_{FRER\text{-}CQF}$ and $J_{FT\text{-}CQF}$ are the same. The reason is that the forwarding rules of FT-CQF and FRER-CQF are the same in the fault-free scenario, and the time slot size and path length of the flow are also the same. Fig. 6(e) and 6(f) show that the maximum jitter $J_{FRER\text{-}CQF}$ and $J_{FT\text{-}CQF}$ is slightly different under the corresponding fault scenarios. The reason is that the receiver will drop the frames that are erroneous due to transient faults, which may lead to variations in the delay of the frames arriving at the destination.

VII. Conclusion

This paper studied the problem of frame errors caused by transient faults in TSN and proposed a fault-tolerant cyclic queuing and forwarding (FT-CQF) mechanism to improve reliability. FT-CQF can retransmit the TT flows according to the CRC-related message returned by the downstream node. We constructed a TSN transmission scenario through OMNeT + + to verify the effectiveness of FT-CQF. The evaluation results show that FT-CQF significantly reduces resource overhead while improving reliability. Additionally, the delay and jitter of FT-CQF also meet the deterministic requirements.

Acknowledgment

We thank the anonymous reviewers for their valuable comments. This work is supported in part by the National Key Research and Development Program of China (No. 2022YFB2901404), and by National Natural Science Foundation of China (NSFC) under Grant No. 62132007, 62221003. Correspondence to: Fengyuan Ren (rfy@lzu.edu.cn), Tong Zhang (zhangt@nuaa.edu.cn)

References

[1] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0," *IEEE Industrial Electronics Magazine*, vol. 11, pp. 17–27, 2017.

[2] D. Tamas-Selicean, P. Pop, and W. Steiner, "Synthesis of communication schedules for ttethernet-based mixed-criticality systems," in *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pp. 473–482, 2012.

[3] N. Finn, "Introduction to time-sensitive networking," *IEEE Communications Standards Magazine*, vol. 2, pp. 22–28, 2018.

[4] M. Ashjaei, L. L. Bello, M. Daneshtalab, G. Patti, S. Saponara, and S. Mubeen, "Time-sensitive networking in automotive embedded systems: State of the art and research opportunities," *Journal of systems architecture*, vol. 117, p. 102137, 2021.

[5] "IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic," *IEEE Std 802.1Qbv-2015*, pp. 1–57, 2016.

[6] "IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks–Amendment 29: Cyclic Queuing and Forwarding," *IEEE 802.1Qch-2017*, pp. 1–30, 2017.

[7] R. Dobrin, N. Desai, and S. Punnekkat, "On Fault-Tolerant Scheduling of Time Sensitive Networks," in *Proc. 4th International Workshop on Security and Dependability of Critical Embedded Real-Time Systems (CERTS 2019)*, (Dagstuhl, Germany), pp. 5:1–5:13, 2019.

[8] S. Kehrer, O. Kleineberg, and D. Heffernan, "A comparison of fault-tolerance concepts for ieee 802.1 time sensitive networks (tsn)," in *Proc. Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pp. 1–8, 2014.

[9] T. Park, S. Samii, and K. G. Shin, "Design optimization of frame preemption in real-time switched ethernet," in *Proc. 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 420–425, 2019.

[10] A. Kostrzewa and R. Ernst, "Achieving safety and performance with reconfiguration protocol for ethernet tsn in automotive systems," *Journal of Systems Architecture*, vol. 118, p. 102208, 2021.

[11] "IEEE Standard for Local and metropolitan area networks–Frame Replication and Elimination for Reliability," *IEEE Std 802.1CB-2017*, pp. 1–102, 2017.

[12] WikiPedia, "Cyclic redundancy check." https://en.wikipedia.org/wiki/Cyclic_redundancy_check, 2023.

[13] Z. Feng, Q. Deng, M. Cai, and J. Li, "Efficient reservation-based fault-tolerant scheduling for ieee 802.1qbv time-sensitive networking," *Journal of Systems Architecture*, vol. 123, p. 102381, 2022.

[14] D. Ergenç and M. Fischer, "On the reliability of ieee 802.1cb frer," in *Proc. IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, pp. 1–10, 2021.

[15] M. Pahlevan, S. Amin, and R. Obermaisser, "Fault tolerant list scheduler for time-triggered communication in time-sensitive networks.," *J. Commun.*, vol. 16, pp. 250–258, 2021.

[16] V. Gavrilut, B. Zarrin, P. Pop, and S. Samii, "Fault-tolerant topology and routing synthesis for ieee time-sensitive networking," in *Proc. Proceedings of the 25th International Conference on Real-Time Networks and Systems*, p. 267–276, 2017.

[17] A. A. Atallah, G. B. Hamad, and O. A. Mohamed, "Fault-resilient topology planning and traffic configuration for ieee 802.1qbv tsn networks," in *Proc. 2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*, pp. 151–156, 2018.

[18] I. Álvarez, I. Furió, J. Proenza, and M. Barranco, "Design and experimental evaluation of the proactive transmission of replicated frames mechanism over time-sensitive networking," *Sensors*, vol. 21, 2021.

[19] W. Kong, M. Nabi, and K. Goossens, "Run-time recovery and failure analysis of time-triggered traffic in time sensitive networks," *IEEE Access*, vol. 9, pp. 91710–91722, 2021.

[20] Y. Zhou, S. Samii, P. Eles, and Z. Peng, "Reliability-aware scheduling and routing for messages in time-sensitive networking," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 20, pp. 1–24, 2021.

[21] I. Álvarez, D. Čavka, J. Proenza, and M. Barranco, "Simulation of the proactive transmission of replicated frames mechanism over tsn," in *Proc. 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1375–1378, 2019.

[22] I. Koren and C. M. Krishna, *Fault-tolerant systems*. Morgan Kaufmann, 2020.

[23] I. Álvarez, M. Barranco, and J. Proenza, "Towards a fault-tolerant architecture based on time sensitive networking," in *Proc. 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1113–1116, 2018.

[24] "OMNet++ Simulation. Version 6.0.1." https://omnetpp.org/.

[25] "INET Framework. Version 4.4.1." https://inet.omnetpp.org/.

[26] "IEC/IEEE 60802 TSN Profile for Industrial Automation." https://1.ieee802.org/tsn/iec-ieee-60802/.