Design Optimization of Frame Preemption in Real-Time Switched Ethernet

Taeju Park
Real-Time Computing Laboratory
University of Michigan, Ann Arbor
Email: taeju@umich.edu

Soheil Samii
General Motors R&D, Michigan, USA
Linköping University, Sweden
Email: soheil.samii@gm.com

Kang G. Shin
Real-Time Computing Laboratory
University of Michigan, Ann Arbor
Email: kgshin@umich.edu

***Abstract*—Switched Ethernet has been, and will also be increasingly common in current and future real-time and embedded systems. The IEEE 802.1 working group has recently developed standards and technologies, commonly referred to as *Time-Sensitive Networking* (TSN), to enhance switched Ethernet with timeliness and dependability. We address, for the first time, the synthesis problem for the TSN frame preemption standards IEEE 802.3br-2016 and 802.1Qbu-2016 by introducing two new configuration parameters: *flow to queue* and *queue to Express/Preemptable MAC interface* assignments. We present an optimization framework to determine these configuration parameters with reliability as the optimization goal. Our proposed framework is shown to outperform commonly used priority-assignment as well as intuitive approaches.**

I. Introduction

Switched Ethernet is increasingly used in various real-time embedded and cyber-physical systems to meet the increasing bandwidth requirements in industrial automation, avionics, and automotive electronics. The IEEE 802.1 Time-Sensitive Networking (TSN) Task Group developed a set of standards to enhance the timeliness and dependability of switched Ethernet. Examples of these standards include credit-based shaping (IEEE Std 802.1Qav-2009), time synchronization (802.1AS-2011), time-triggered communication (802.1Qbv-2015), and frame preemption (802.1Qbu-2016 and 802.3br-2016).

The successful deployment of real-time applications on Ethernet with these new technologies relies on design-time synthesis and optimization to determine network configurations. Several researchers have recently developed methods and techniques for the optimization of real-time Ethernet app design. A large body of related work deals with the synthesis of time-triggered Ethernet schedules for hard real-time applications [4, 15, 7]. This has recently been extended to synthesize gate control lists (IEEE Std 802.1Qbv for time-driven scheduling) [12], as well as towards robustness to link failures [2] and mixed-criticality applications [16, 17]. Researchers have also addressed routing synthesis [8, 10] and traffic class assignment [6].

While most past research focused on synthesis of communication schedules and priorities, little attention has been paid to frame preemption on Ethernet. Thiele and Ernst [18] proposed a worst-case timing analysis for networks employing frame preemption. To the best of our knowledge, there has not been any proposal for the design of real-time, preemptable Ethernet.

Our contributions: We introduce and address the synthesis problem for frame preemption on Ethernet according to the recently developed 802.1Qbu-2016 and 802.3br-2016 standards. Specifically, we study the problem of assigning priorities, and hence queue allocation, of each real-time data flow, as well as, for each queue, assigning whether the flow is transmitted in the preemptable MAC (pMAC) or express MAC (eMAC) interface of the corresponding egress port. We present a genetic algorithm-based optimization framework that exploits a worst-case preemption-aware timing analysis, and use it to improve reliability of networks as a case study. For the effective and efficient use of the framework, we propose an initialization algorithm for each goal. We evaluate networks of different sizes and complexities, including a real-life automotive system.

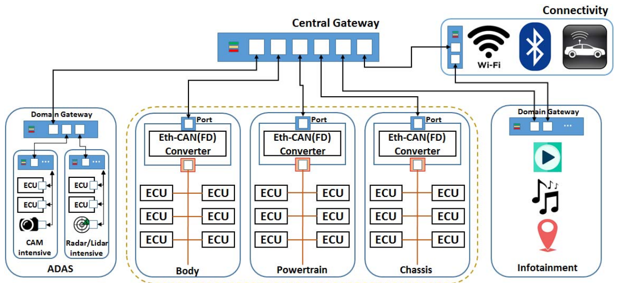


Fig. 1: Example of an in-vehicle system architecture

II. Networked System Model

An example in-vehicle architecture is provided in Fig. 1. Electronic Control Units (ECUs) are grouped by their functions, forming *functional domains*. Each domain may use a different network protocol to optimize the implementation cost while satisfying its requirements. In the example, the powertrain domain uses Controller Area Network (CAN) or CAN with flexible data-rate (CAN-FD), whereas the ADAS domain uses switched Ethernet as its internal network. Moreover, the switched Ethernet may be used as a backbone network in future vehicle architectures.

The architecture can be abstracted as a tree-topology switched-Ethernet system as shown in Fig. 2. End-stations represent either domain controllers or ECUs equipped with an Ethernet port, and switches represent gateways. An end-station has only one physical Ethernet port, and the port is physically linked to a corresponding port of a switch. A switch has multiple ports, and each port is physically linked to a corresponding port of either an end-station or another switch.

A. Preemption Supportive Port Model

We assume that every Ethernet port supports frame preemption based on IEEE standards 802.1Qbu-2016 and IEEE

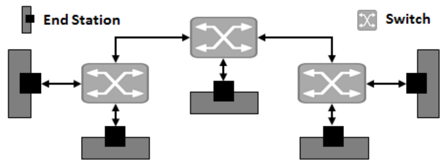


Fig. 2: Abstracted system architecture

802.3br-2016. Each egress port has at most 8 queues to serve different classes of frames, and each queue is mapped to either the express MAC (eMAC) interface or the preemptable MAC (pMAC) interface of the port, as shown in Fig. 3. A queue mapped to the eMAC interface is called an *express queue*, and a queue mapped to the pMAC interface is a *preemptable queue*. Consistent with practical implementations, we assume that the queues are strictly prioritized, and thus a frame in a higher priority queue always wins the transmission arbitration against a frame in a lower priority queue.

If a frame is forwarded to an express queue, then it passes through the eMAC interface for transmission, and the frame is called an *express frame*. Otherwise, the frame is called a *preemptable frame*. An express frame cannot be preempted by any other frames, whereas a preemptable frame can be preempted by any express frames. Note that preemptable frames cannot preempt each other. Hence, only one level of frame preemption is allowed. The formats of both express and preemptable frames are shown in Fig. 4.

Even though, technically, a higher-priority queue can be mapped to a pMAC and a lower-priority queue can be mapped to eMAC, it is unlikely scenario in practical implementations. Thus, we assume that every express frame has higher priority than any preemptable frame. Based on this assumption, an egress port i can be defined as $ep_i = \{Q_i, V_i, r_{TX,i}, \zeta_i\}$ where

- Q_i: the set of queues in the egress port i. $Q_i = \{q_{i,0}, q_{i,1}, ..., q_{i,n}\}, n \leq 7$. $q_{i,0}$ is the highest-priority queue, $q_{i,1}$ is the second highest-priority queue and $q_{i,n}$ is the lowest- priority queue in order;
- V_i: the corresponding end-station or switch with egress port i;
- $r_{TX,i}$: physical link speed; and
- ζ_i: boundary between express and preemptable queues.

For example, if ζ_i is 3, then $q_{i,0}, \ldots, q_{i,3}$ are express queues and $q_{i,4}, \ldots, q_{i,n}$ are preemptable queues.

B. *Traffic Flow Model*

An in-vehicle traffic flow is initiated periodically and has fixed, given route. We group in-vehicle traffic flows into four classes; control signal, sensor data (vehicle dynamics), raw

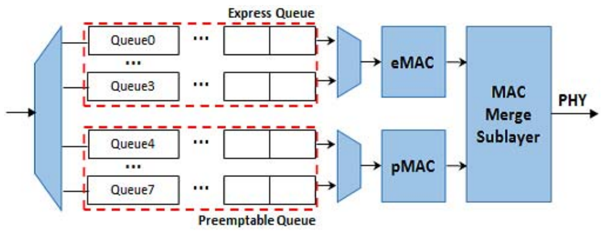


Fig. 3: Frame preemption supportive port

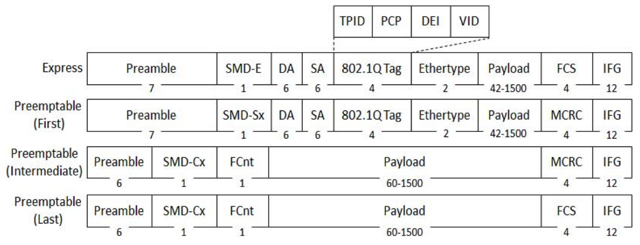


Fig. 4: MAC frame format

data from camera or radar (lidar), and *others* as shown in Table I. The control signal, sensor data, and raw data from camera or radar (lidar) are usually used by time-critical applications, and thus these data flows have deadlines. On the other hand, non-time-critical traffic flows do not have deadlines. A traffic flow is defined as $f_i = \{T_i, D_i, L_i, P_i, J_i, \chi_i, \rho_i\}$ where T_i is period of f_i, D_i is the relative deadline of f_i, L_i is the maximum payload size of f_i, P_i is the value of priority code point (PCP) of f_i, J_i is the release jitter of f_i, χ_i is the class of f_i and ρ_i is the route of f_i.

As shown in Fig. 4, the PCP value is specified in the Ethernet header of frames of a traffic flow. The frames are placed in one of the eight queues based on their PCP value. For example, suppose $PCP = k$ is mapped to $q_{i,k}$. If the PCP value of a frame is 0, then the frame is queued into $q_{i,0}$. Because queues in egress ports have priority, the priority of frames are intrinsically determined by their PCP values. We assume that, for any egress port i, $PCP = k$ is mapped to $q_{i,k}$. Thus, a traffic flow has the same priority in all egress ports in its route from source to destination.

III. SYNTHESIS PROBLEM

To utilize the standardized frame preemption, we must synthesize not only the assignment of frames of traffic flows to queues but also the assignment of queues to one of the two MAC interfaces (eMAC and pMAC). That is, we need to determine not only a set of P_i for given traffic flows but also a set of ζ_i for given egress ports; the latter parameter due to our assumption that express queues have higher priority than preemptable queues. To our knowledge, there is no solution in literature that deals with the two assignments at once. This paper addresses these problems at once for the recently standardized frame preemption technology.

The example shown in Fig. 5 stresses the importance of deciding ζ_i. As explained before, the placement of type boundary at an egress port determines the type of frames of traffic flows. For example, the frames of f_1 are express and the frames of f_2 and f_3 are preemptable at the egress port ep_1 when $\zeta_1 = 2$. Hence, placement of the type boundary affects the maximum queuing delay of the frames at the egress port.

Suppose the maximum transmission time of frames f_1, f_2 and f_3 is 5. That is, the maximum waiting time for the entire transmission of a frame is 5 at an egress port. Also, suppose that the maximum waiting time is 1 when an express frame waits for the transmission of a fragment of a preemptable frame. Then, with the configuration of $\zeta_1 = 2$ and $\zeta_2 = 4$, the maximum queuing delay of frames of f_1 at ep_1 is 1 because the frames can preempt any frames of f_2 and f_3. Also,

	Critical Frame			Non-Critical Frame
	Class 1	Class 2	Class 3	Class 4
Type	Control	Sensor (Vehicle Dynamics)	CAM, Radar	Others
Period	10ms-40ms	40ms-100ms	20ms-100ms	50ms-1s
Deadline	same as period			No deadline
Maximum Payload Size	10-100byte	100-200byte	400-1500byte	1500byte

TABLE I: Ethernet traffic flow classes

ζ_1	ζ_2	R_1^+	R_2^+	R_3^+
2	2	12	30	30
2	4	16	26	30
2	6	16	30	30
4	4	20	22	30
4	6	20	26	30
6	6	20	30	30

TABLE II: Worst-case E2E latency with different type boundary configurations

that at ep_2 is 5 because a frame of f_1 needs to wait for the entire transmission of a frame of f_2 in the worst case. Thus, the worst-case end-to-end (E2E) latency of frames of f_1 is $1+5+5+5 = 16$. The worst-case E2E latency of given traffic flows (R_i^+) with different boundary configurations are shown in Table II. The given traffic flow is schedulable ($R_i^+ \leq D_i$) only with the configuration of $\zeta_1 = 4$ and $\zeta_2 = 4$.

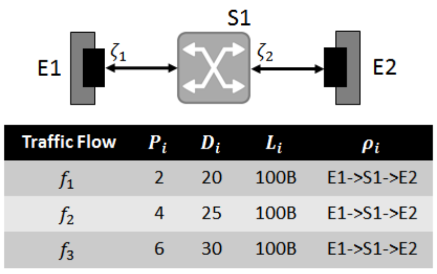
As shown in the above example, synthesis of PCP and placement of type boundaries directly affect the worst-case E2E latency of the frames. That is, we can configure the worst-case E2E latency of the frames by controlling the assignments to achieve a given optimization goal (G).

Finding the optimal assignments for a given goal is non-trivial because of the huge combinatorial design space, so it is computationally infeasible to explore all possible combinations even for a small number of traffic flows. For example, if there are n traffic flows and m egress ports, there are at least 8^{n+m} different combinations, because there are 8 possible priority levels for each traffic flow and 8 queues in an egress port.

IV. Generic Solution Approach

As mentioned in the previous section, it is computationally infeasible to find the optimal assignments for a given goal G. Thus, we propose a heuristic framework based on the well-known genetic algorithm (GA).

Definition 1 (Individual)**.** Each individual has two genes PS and TBS, where PS is a set of P_i for given traffic flows and TBS is a set of ζ_i for given egress ports. An individual is thus defined as $s_i = (PS, TBS)$.



Traffic Flow	P_i	D_i	L_i	ρ_i
f_1	2	20	100B	E1->S1->E2
f_2	4	25	100B	E1->S1->E2
f_3	6	30	100B	E1->S1->E2

Fig. 5: Example of showing importance of ζ_i decision.

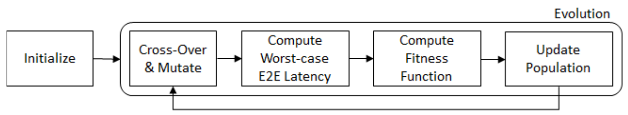


Fig. 6: The overview of GA-based framework

Definition 2 (Population)**.** The k^{th} population ϕ_k is a set of individuals. The individuals in a population are always sorted by the fitness function. We denote by $\phi_{k,i}$ the individual that has the i^{th} best result of fitness function in the k^{th} population.

A. *Overview of GA-Based Framework*

Fig. 6 provides an overview of our GA-based framework. The initialization step sets up an initial population (ϕ_0) with which the algorithm starts and continues evolution to approach a convergence point until the number of evolutions exceeds the pre-defined maximum. Since we want to optimize the set of P_i's and the set of ζ_i's, PS and TBS are the evolution parameters. During the evolution step, the algorithm generates a new individual repeatedly by inheriting the evolution parameters from two randomly chosen individuals in the current population (cross-over) and by mutating the inherited parameters.

After the mutation, the framework computes the worst-case E2E latency for the given traffic flows using the state-of-the-art worst-case E2E analysis technique [18]. With the computed worst-case E2E latency, the framework obtains the result of the fitness function for the new individual. The given optimization goal G is often used as the fitness function. The result for the new individual is compared to that for individuals in the current population. If the new individual has a better result than any individuals in the current population (ϕ_c), the current population is updated.

B. *Evolution Procedure*

The evolution procedure consists of selection, cross-over, mutation, and population update as illustrated in Fig. 7. The selection step selects two individuals randomly from the current population (ϕ_c). In the cross-over step, a new individual is generated, and the individual inherits the evolution parameters from the selected individuals. The new individual inherits PS from the first-selected parent and TBS from the second-selected parent. In the mutate step, either P_i of a traffic flow or ζ_i of an egress port is changed to a random value. Through the cross-over and mutate steps, the evolution parameters are shaken, and thus the new individual can have a better or worse result than its parents. To evaluate the new individual, the worst-case E2E latency and the fitness function are re-computed with the changed PS and TBS. If the new individual has a better result than the lowest-ranked individual

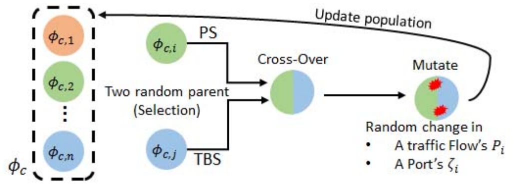


Fig. 7: Evolution procedure

($\phi_{c,n}$) in the current population (ϕ_c), the current population is updated by placing the new individual in the population and removing the lowest-ranked individual from the population.

V. Case Study: Reliability

Reliable communication is one of the most important requirements for vehicle applications. Unexpected functional failures could lead to disastrous results, such as damage/loss of properties or lives. Recent studies [11, 3] claim that the well-known automatic repeat request (ARQ) protocol [9] is a promising way to improve reliability for real-time switched Ethernet.

We assume that every end-station supports the following ARQ protocol. When a frame of traffic flow i is initiated by a sender, the frame is first forwarded into an ARQ buffer. Then, ARQ sets an expiration time for the frame. The worst-case roundtrip time of the traffic flow is used as the expiration time. ARQ then copies the frame and forwards the frame to the egress port without waiting for the completion of transmission of any prior frames. When the receiver receives the frame correctly, it sends an acknowledgement (ACK) frame to the sender. The ACK frame has same priority as its corresponding data frame. If the ACK frame arrives at the sender within the expiration time, the frame is removed from the ARQ buffer. If the ACK frame does not arrive at the sender within the expiration time or the negative acknowledgement (NAK) frame arrives at the sender, ARQ copies and re-transmits the frame again.

ARQ improves reliability via this re-transmission mechanism. We will explain below how the proposed GA-based framework can be applied to maximize the number of "allowable" re-transmissions.

A. Applying the GA-Based Framework

1) Optimization Goal (Fitness Function): Critical Scaling Factor (CSF) [14, 13] is a well-known metric that represents the number of times each task can be (re-)executed to recover a system from unexpected failures without missing its deadline. Thus, a higher CSF value means higher reliability. In the context of networking, the metric represents the maximum number of re-transmissions allowed without missing deadlines. Thus, CSF for a traffic flow is defined as

$$CSF_i = \frac{D_i - R_i^+ - R_{ACK,i}^+}{R_i^+ + R_{ACK,i}^+}, \tag{1}$$

where R_i^+ is the worst-case E2E latency of frames of traffic flow i, and $R_{ACK,i}^+$ is the worst-case E2E latency of ACK frames of traffic flow i.

Maximizing the number of allowable re-transmissions for given traffic flows translates to maximizing the minimum CSF among all traffic flows. Thus, it can be formulated as a max-min optimization problem as:

$$\mathbf{G} : Maximize \left\{ \underset{i}{Min} \; \{CSF_i\} \right\} \tag{2}$$

2) Generating the initial population: Competing frames have a seesaw relationship with respect to the worst-case E2E latency. Since CSF depends on the worst-case round trip time ($RTT_i^+ = R_i^+ + R_{ACK,i}^+$), the competing frames also have a seesaw relationship with respect to CSF. Thus, the above optimization problem can be cast into the problem of balancing CSF for given traffic flows.

We want to generate an initial individual which has a well-balanced CSF among given traffic flows. We make three hypotheses about a set of P_i's which leads to well-balanced CSF:

H1: The lower the class, the higher the PCP;
H2: Traffic flows are well distributed to eight queues;
H3: The smaller the payload size, the higher the PCP.

(**H**1) In general, frames of lower-class traffic flows have relatively short deadlines (less room for re-transmission) than those of higher class traffic flows. Thus, assigning higher priorities to lower-class traffic might result in a well-balanced CSF.

(**H**2) If queues are not used fairly, this would increase the queuing delay of traffic flows. Suppose, for example, there are two frames that need to be queued. If the frames are forwarded into the same queue, they interfere with each other. However, if the frames are forwarded to different queues, only lower-priority frames would be interfered by higher-priority frames. So, the latter case might result in a well-balanced CSF.

(**H**3) Transmission time of a frame with a smaller payload is smaller than that with a larger payload size. Since the interference by a frame with large transmission time is less likely to keep the safety margin larger than that by a frame with small transmission time, a frame with smaller payload should have a higher PCP than that with larger payload to enable a well-balanced CSF.

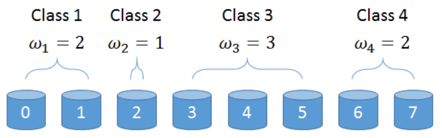


Fig. 8: Assign the isolated queue resource to each class

Based on the above hypotheses, we generate an initial population. For **H**1, the algorithm assigns separate queue resources to each class. The first ω_1 queues are assigned to class 1 and the next ω_2, ω_3 and ω_4 queues are assigned to class 2, 3 and 4 as illustrated in Fig. 8. $\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4\}$ denotes the queue resource isolation. For example, the frames of class 1 traffic flows should be buffered in the first ω_1 queues, and thus the PCP of the frame should be $\{0, \ldots, \omega_1 - 1\}$. There are a total of 60 different Ω's. Because of the manageable number of ways to consider, the algorithm explores the entire search space to find the best Ω.

For **H**2, the maximum payload of each queue is limited in the initialization process. The payload limit is computed for

Number of End Stations	5 - 20
Number of Switches	3 - 7
Number of Frames	100 - 500
Portion of Class {1,2,3,4}	{5%, 5%, 50%, 40%}

TABLE III: Simulation Configuration

each class, and thus the queues assigned to the same class have the same payload limit. The payload limit for each class is computed as

$$PL_k = \frac{\sum_{f_i \in F_k} L_i}{\omega_k}, \tag{3}$$

where F_k is the set of traffic flows whose class is k and ω_k is the number of queues assigned to class k. For example, suppose that queue 0 and 1 are assigned to class 1 and $PL_1 = 100$. Also, suppose that the sum of payloads of traffic flows assigned to queue 0 already reached 100. Then, the remainder of class 1 traffic flows should be assigned to queue 1.

For **H**3, the algorithm sorts the given frames in ascending order by the payload size, and fills up from the highest priority queue to the lowest priority queue with the sorted frames. Thus, a frame with the smaller payload is buffered into the higher priority queue.

VI. Evaluation

A. Evaluation Setup

We have evaluated the performance of our proposed genetic-based framework for the above optimization goal by developing an evaluation tool, *Priority Assignment Evaluation for TSN (PAET)*. The roles of PAET is (1) to generate parameterized random networks and traffic, (2) to apply baseline algorithms or our proposed GA-based framework to the generated network and traffic, (3) to compute the fitness function, and (4) to execute a series of test. When we compute the fitness function, we use the state-of-art timing analysis [18] to compute the worst-case E2E latency. In the following, we describe how to generate the network and traffics, and also explain the baselines used in this paper.

1) Baselines: To our knowledge, there is no proposed solution in the literature to address our design problem. Hence, we use the two assignment algorithms AOPA and RPA, which are widely used for other in-vehicle network protocols, and an intuitive assignment approach (FPCP) as the baselines:

- FPCP: Fixed PCP for each class. (Class1: 0, Class2: 2, Class3: 4, Class4: 7).
- AOPA: Audsley's Optimal Priority Assignment [1].
- RPA: Robust Priority Assignment [5]

Because these baselines only determine a set of P_i for given traffic flows, we have to decide the ζ_i for given egress ports. We assign a fixed value to all egress ports. For example, $\forall i\ \zeta_i = 0$. Because the performance of the baselines can vary according to the value of ζ_i, we test all possible values (0, 1, ..., 7) and use the best performance with the fixed borderline assignment as the results of baselines. For example, if the performance of FPCP with $\zeta_i = 2$ is better than the performance of FPCP with $\zeta_i = 4$, we use results with $\zeta_i = 2$ as the results of FPCP.

2) Network Generation: PAET receives the number of frames, the number of end stations and the number of switches as input parameters. The value of the input parameters are in the range shown in Table. III. With the given parameters, the network generator in PAET generates a random tree-topology network according to the following sequences.

- Step 1. Assign a random corresponding switch to each end station.
- Step 2. Select two random switches to make connection between them.
- Step 3. If there already exists a valid route between the randomly chosen switch, go to step 2 to avoid making any cycle. Otherwise, make a connection between them.
- Step 4. Check whether all the switches have at least two links or not. If all the switches have at least two links, then terminate. Otherwise, go to step 2.

3) Traffic Generation: PAET receives the total number of traffic flows as an input parameter. With the given total number of traffics, the traffic generator in PAET generates random traffic flows according to the following sequences:

- Step 1: Generate a traffic flow and decide its class with fixed portion of each class shown in Table III.
- Step 2: Decide period, deadline, maximum payload size, and release jitter for the traffic flow. The range of these values are shown in Table I.
- Step 3: Choose two random different end stations. The first one is source of the traffic flow and the other is destination of it. Because the network has a tree topology, the route of the traffic between the source and the sink is implicit and unambiguous.

B. Evaluation Results and Analysis

We compare our algorithm with the thre baselines explained above by generating 400 different test cases. The 'Init' is the results after applying the designed initialization algorithm, and the 'Genetic' is the results after 100,000 evolutions.

1) Schedulability: We first evaluate the baselines and our algorithms in terms of schedulability as shown in Fig. 9. With ARQ, $\forall i\ RTT_i^+ \leq D_i$ means schedulable. Without ARQ, $\forall i\ R_i^+ \leq D_i$ means schedulable because there is no acknowledgement. Because $RTT_i^+ > R_i^+$, the schedulability without ARQ is higher than that with ARQ.

According to the results, while all generated test cases are schedulable with our algorithm, some of the test cases are unschedulable with the baselines. These schedulability difference mainly comes from the queue usage. We know that there are eight queues in an egress port. However, FPCP assigns priorities to traffic flows based on their class, and thus only four queues in an egress port are used. In other words, the other four queues are not used. Also, AOPA and RPA assign the lowest possible priority to each traffic flow, and thus lower priority queues are more likely used than higher priority queues. As a result, the given resources are not used efficiently, and these algorithms cannot benefit from the load balancing effect in the worst-case E2E latency.

2) Reliability (CSF): The average maximum CSF for the generated 400 test cases are shown in Fig. 10 (Left). Because CSF represents the number of possible re-transmission, we

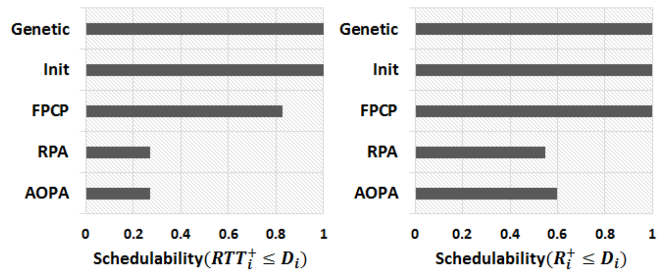


Fig. 9: (Left) Schedulability with ARQ (Right) Schedulability without ARQ

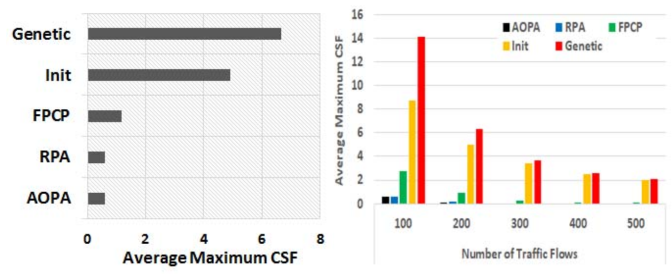


Fig. 10: (Left) The average of maximum CSF for the generated test cases with different assignment algorithms (Right) The average of maximum CSF for different number of traffic flows

assume that CSF is 0 if a given test case is unschedulable. In terms of CSF, the baselines (FPCP, AOPA, and RPA) have the low performance because these algorithms do not efficiently utilize the given queue resources in egress ports as we explained in the previous subsection. Therefore, results with Init and Genetic are better than those with FPCP, AOPA, and RPA.

We believe that the Init algorithm is an advanced version of FPCP for CSF optimization. It assigns priorities to given traffic flows based on their class like FPCP. However, it evenly distributes the given traffic flows to the eight queues. Thanks to the efficient use of given resources, Init shows better results than the baselines. The results of Genetic shows not only the effect of efficient use of given queue resources but also the effect of optimization of P_i and ζ_i. Even though the results of Genetic is not the optimal value, the average maximum CSF with Genetic is about 8.5 times higher than that with AOPA and RPA, about 3.9 times higher than that with FPCP, and 1.32 times higher than that with Init.

As shown in Fig. 10 (Right), Genetic shows much better performance with the small number of traffic flows. This is because, with the small number of traffic flows, frames with small deadline can have small worst-case E2E latency, making large number of re-transmissions feasible.

C. Scalability

We have measured the average execution time of 1 evolution for different number of traffic flows to evaluate the scalability of our approach. The measured execution time on our machine (Intel Xeon E5-2683 @ 2.10GHz, 128GB Memory) are shown in Table. IV. The results show that the execution time of 1 evolution is exponentially increased along the number of traffic flows. For example, the execution time of 1 evolution for 400 traffic flows is about 16 times larger than the execution time of 1 evolution for 100 traffic flows. Because of the exponential increase, this approach is not suitable for large number of traffic flows cases. However, we believe that the performance is acceptable for practical automotive use cases, in particular due to that our method is applied at design time.

Traffic Flows	100	200	300	400	500
Execution Time	0.3s	1.15s	2.86s	5.02s	8.02s

TABLE IV: Execution time of 1 evolution

VII. Conclusions

We addressed the synthesis problem for frame preemption on Ethernet (802.1Qbu-2016 and 802.3br-2016). We presented a GA-based framework to determine priorities and queue assignment to the two new MAC interfaces eMAC and pMAC. Our experimental results confirmed that our framework can improve transmission reliability in a real-time Ethernet network and outperforms existing assignment algorithms when adapted to the frame-preemption synthesis problem.

References

[1] N.C. Audsley. On priority assignment in fixed priority scheduling. *Information Processing Letters*, 79(1):39 – 44, 2001.
[2] G. Avni, S. Guha, and G. Rodriguez-Navas. Synthesizing time-triggered schedules for switched networks with faulty links. In *2016 International Conference on Embedded Software (EMSOFT)*, pages 1–10, Oct 2016.
[3] P. Axer, D. Thiele, and R. Ernst. Formal timing analysis of automatic repeat request for switched real-time networks. In *Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES 2014)*, pages 78–87, June 2014.
[4] Sofiene Beji, Sardaouna Hamadou, Abdelouahed Gherbi, and John Mullins. Smt-based cost optimization approach for the integration of avionic functions in ima and ttethernet architectures. In *Proceedings of the 2014 IEEE/ACM 18th International Symposium on Distributed Simulation and Real Time Applications*, DS-RT '14, pages 165–174, Washington, DC, USA, 2014. IEEE Computer Society.
[5] R. I. Davis and A. Burns. Robust priority assignment for fixed priority real-time systems. In *28th IEEE International Real-Time Systems Symposium (RTSS 2007)*, pages 3–14, Dec 2007.
[6] Voica Gavriluţ and Paul Pop. Traffic class assignment for mixed-criticality frames in ttethernet. *SIGBED Rev.*, 13(4):31–36, November 2016.
[7] Z. Hanzalek, P. Burget, and P. Sucha. Profinet io irt message scheduling with temporal constraints. *IEEE Transactions on Industrial Informatics*, 6(3):369–380, Aug 2010.
[8] Sune Mølgaard Laursen, Paul Pop, and Wilfried Steiner. Routing optimization of avb streams in tsn networks. *SIGBED Rev.*, 13(4):43–48, November 2016.
[9] Shu Lin, D. J. Costello, and M. J. Miller. Automatic-repeat-request error-control schemes. *IEEE Communications Magazine*, 22(12):5–17, December 1984.
[10] Rouhollah Mahfuzi, Amir Aminifar, Soheil Samii, Ahmed Rezine, Petru Eles, and Zebo Peng. Stability-aware integrated routing and scheduling for control applications in ethernet networks. 2018.
[11] Mischa Möstl, Daniel Thiele, and Rolf Ernst. Invited - towards fail-operational ethernet based in-vehicle networks. In *Proceedings of the 53rd Annual Design Automation Conference*, DAC '16, pages 53:1–53:6, New York, NY, USA, 2016. ACM.
[12] R. Serna Oliver, S. S. Craciunas, and W. Steiner. Ieee 802.1qbv gate control list synthesis using array theory encoding. In *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 13–24, April 2018.
[13] Sasikumar Punnekkat, Rob Davis, and Alan Burns. Sensitivity analysis of real-time task sets. In R. K. Shyamasundar and K. Ueda, editors, *Advances in Computing Science — ASIAN'97*, pages 72–82, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.
[14] J. Regehr. Scheduling tasks with mixed preemption relations for robustness to timing faults. In *23rd IEEE Real-Time Systems Symposium, 2002. RTSS 2002.*, pages 315–326, 2002.
[15] W. Steiner. An evaluation of smt-based schedule synthesis for time-triggered multi-hop networks. In *2010 31st IEEE Real-Time Systems Symposium*, pages 375–384, Nov 2010.
[16] W. Steiner. Synthesis of static communication schedules for mixed-criticality systems. In *2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, pages 11–18, March 2011.
[17] Domitian Tamas-Selicean, Paul Pop, and Wilfried 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*, CODES+ISSS '12, pages 473–482, New York, NY, USA, 2012. ACM.
[18] D. Thiele and R. Ernst. Formal worst-case performance analysis of time-sensitive ethernet with frame preemption. In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–9, Sept 2016.