Fault-Resilient Topology Planning and Traffic Configuration for IEEE 802.1Qbv TSN Networks

Ayman A. Atallah, Ghaith Bany Hamad, Otmane Ait Mohamed
Department of Electrical and Computer Engineering
Concordia University
Montreal, QC, Canada
Email: {a_atal, g_banyha, ait}@encs.concordia.ca

***Abstract*—Time-Sensitive Networking (TSN) is a set of IEEE standards that are being developed to enable a reliable and real-time communication based on Ethernet technology. It supports Time-Triggered (TT) traffic to allow a low latency as well as deterministic timing behavior. TSN adapts the concept of seamless redundancy to ensure interruption-free fault-resilience. In this paper, our goal is to synthesize a network topology that supports seamless redundant transmission for TT messages. Therefore, we propose a greedy heuristic algorithm for joint topology, routing, and schedule synthesis. The proposed algorithm is capable to generate fault-resilient topology that guarantee feasible routing and scheduling for TT traffic. In particular, the topology is constructed iteratively such that all messages are routed through disjoint paths with a feasible schedule and the network cost is minimized. To achieve this goal, we formulate the topology synthesis problem as iterative path selection problem. Starting from a weighted undirected graph which represents an initial fully-connected network, the cost implied of using each link is mapped as arcs weights in the graph. Then, we adapt Yen's algorithm to iteratively find the minimum-cost paths for the considered messages. The scalability and the efficiency of the proposed approach are demonstrated using 380 synthetic test cases. The results show that the proposed approach is capable of finding fault-resilient topology with up to 50% less cost compared to the typical approach. Moreover, the approach scalability is validated e.g., it handles 24 ECUs with 600 messages problems within an average time of 8 sec.**

***Index Terms*—Network Reliability, Fault-Resilience, Topology Planning, TSN Network, IEEE 802.1Qbv, Greedy Heuristic, Routing, Scheduling.**

I. INTRODUCTION

Recent years have shown an increase in the demand for the distributed embedded systems for safety-critical applications such as automotive and avionics [1]. In this type of systems, many sensors and actuators are simultaneously controlled by several Electronic Control Units (ECUs). It is essential to develop networks that guarantee a reliable and real-time communication to realize this category of systems. Time Sensitive Networking (TSN) is a set of IEEE standards that define the specifications for an Ethernet-based switched networks to match the reliability and timing requirements for safety-critical applications [2]. TSN-based networks address the system's real-time requirements by supporting a class of Time-Triggered (TT) traffic or so-called scheduled traffic.

On the other hand, TSN networks fulfill the reliability requirements by adapting the concept of seamless redundancy. In particular, every critical message is transmitted through multiple disjoint paths which guarantees an interruption-free communication under links failure. Typically, seamless redundancy is realized using multiple copies of the network. Although, such redundancy can provide the desired level of reliability, with the current constraints on the system area and power such approach is not always applicable. Recently, researchers and system designers started investigating the possibility of integrating the fault-resilience constraint at the network planning stage as presented in [3]. In other words, synthesizing a fault-resilient network topology to provide multiple disjoint paths for critical messages. As expected, initial results show that such integration ensures the required level of reliability in a more efficient manners than the full network redundancy. Such synthesis minimizes the overall number of components in the networks which in turn reduces the overall cost.

An inherent complexity related to the fault-resilient synthesis is the dependency between the network topology and the traffic delivered by that network. For example, the number of disjoint paths that the topology can provide between node v_k and v_j depends on the messages size and the required redundancy level. So that, joint topology and routing synthesis is essential to guarantee feasible routing as introduced in [3]. However, the proposed TSN network synthesis approach in [3] considers only non-scheduled traffic (i.e., not TT).

When TT traffic is considered, additional constraints are related scheduling to be stratified. These constraints add further dependencies between the network topology and traffic in the case of TT traffic. However, the problem is that scheduling feasibility cannot be guaranteed unless the schedule is determined with the topology synthesis. Recalling that solving each one of topology or schedule synthesis is NP-hard problem [4], [5]; thus, the joint synthesis will obviously suffer from serious scalability limitations. Hence, ILP- and SMT-based solutions such as the one proposed in [6] and [7], respectively, are not capable of tackling realistic large-scale networks.

***Contribution*:** In this paper, we are interested in TSN topology synthesis, as well as critical traffic routing and scheduling. The contribution of this paper is three-fold:

1) A new approach to investigate the applicability and efficiency of satisfying the fault-resilience constraints

978-1-5386-5992-2/18/$31.00

at the topology synthesis phase for TT network is proposed.
2) A new greedy heuristic algorithm for incremental network topology and configuration synthesis is proposed. The scalability, as well as, capabilities of the proposed algorithm are evaluated. 380 synthetic test cases with various topology sizes as well as, number of messages are used for an extensive evaluation.
3) An adaptive solution for cost-aware topology planning is introduced. In particular, a mapping of the cost of the network topology into a weighted undirected graph is proposed. This approach tries to minimize the cost through finding the shortest path in the graph. In fact, this technique allows adapting the efficient routing algorithms such as Yen's algorithm [8] to synthesize network topology iteratively.

The remainder of this paper is structured as follows. We discuss related works in Section 2. The considered system model is presented in Section 3. In Section 4, we introduce a motivational example to define the addressed problem. In Section 5, we describe the joint topology, routing, and scheduling algorithm for TT multi-hop networks. We present the experimental evaluation in Section 6. Finally, the conclusion is introduced in Section 7.

II. Related Work

Highly reliable communication is vital for safety-critical applications. Many recent literatures address the topic of fault-resilience for TSN networks. Authors in [9] provide an excellent overview of the different fault-resilience concepts for IEEE 802.1 TSN networks.

In [10], a mathematical analysis of the gain in the transmission reliability due to temporal redundancy is introduced. In our previous work [11], we exploit the temporal redundancy approach to develop a reliability-aware routing algorithm for TSN network. This approach utilizes an ILP-based formulation to determine the path and temporal redundancy level for each message such that the required Mean-Time-to-Detect-Error (MTTDE) is satisfied. Despite the resilience efficiency of the temporal redundancy against transient transmission error, it is incapable to tolerate permanent link failure scenario since all replicas are sent through the same route.

According to fault-resilient topology planning for TSN network, a heuristic, as well as constrained programming-based solutions for fault-tolerant topology planning and traffic routing are introduced in [3]. The proposed techniques tackle a special class of rate-constraint traffic namely, Urgency-Based Scheduler (UBS), which does not require scheduling tables. The network cost and area are minimized, whereas, the traffic is routed through the predefined number of disjoint paths. However, these techniques are not suitable for TT traffic. In particular, the strategy of joint topology and routing synthesis for time-triggered networks does not guarantee feasible scheduling.

Regarding traffic configuration in TSN networks, Laursen et al [12] introduce a schedulability-aware routing for TT traffic in TSN network to generate routes that increase the chance of finding a feasible schedule. Authors formulate the routing problem of multi-hop TT network by a set of ILP constraints that takes into account an additional parameter, namely, maximum scheduled traffic load. This parameter reduces the chance of getting a conflict between TT messages. Furthermore, several works have addressed the joint routing and scheduling synthesis. In [13] and [6] ILP formulations to jointly solve the routing and scheduling problems of TT traffic, which offer higher optimization capabilities, are introduced. However, these ILP-based techniques suffer from scalability limitation, i.e., it is applicable for small size problems.

TABLE I: Example Library for Bridges Modules

Module ID	Number of Ports	Cost
b_1	2	3
b_2	3	6
b_3	4	8

III. System Model

In this paper, we consider an Ethernet-based multi-hop switched architecture compliant with IEEE TSN standard. We consider a set of ECUs $\mathcal{E}$ that exchange a set of TT messages $\mathcal{M}$ through a set of bridges $\mathcal{B}$. The messages in $\mathcal{M}$ should be sent through the network according to an overlap-free schedule such that the output ports of the bridges along the message's path are dedicated for that message before it arrives. The set of ECUs $\mathcal{E}$ as well as the set of TT messages $\mathcal{M}$ are predefined as input to the problem. On the other hand, the number of bridges in the network, as well as the physical connection between these bridges are decision variables obtained by the proposed algorithm. All physical links in the network are full-duplex and identical in terms of speed.

The network topology is modeled as a weighted undirected graph $G(\mathcal{V}, \mathcal{W})$, where $\mathcal{V} \in \mathcal{E} \cup \mathcal{B}$ denotes the set of vertices, whereas, $\mathcal{W}$ denotes the set of weighted arcs. $\mathcal{A}$ denotes the library of the bridges that can be used to construct the topology. Each element in $\mathcal{A}$ is defined by its monetary cost as well as its number of ports. An example of the bridges library is shown in Table I. Each message, $m \in \mathcal{M}$, is defined by the following tuple $\langle m.src, m.dest, m.size, m.period, m.tl \rangle$ where *m.src* and *m.dest* denote the message source and destination, respectively. *m.size* and *m.period* denote the frame size and period between frames, respectively. Finally, *m.tl* denotes the Tolerance Level (TL) of the message which specifies the required number of disjoint paths through which the message should be routed.

We assume that TL of the messages are specified by the design engineers according to the criticality of the application that exchanges these messages. Multiple replicas of each message, m^i where $i \in m.rl$, are transmitted through specific set of disjoint paths, $r_m^i \in R_m$. The path r_m^i is denoted by an ordered sequence of connected vertices starting from *m.src* and ending at *m.dest*. The set of selected paths in the network for all messages composes the traffic routing, $\mathcal{R}$. In fact, the

TABLE II: TT Messages $\mathcal{M}$ for The Motivational Example.

Index	(M.src, M.dest)	M.size (KB)	M.period (ms)	M.tl
M_1	(ECU_3, ECU_2)	1.5	3	1
M_2	(ECU_3, ECU_2)	0.5	3	2
M_3	(ECU_1, ECU_2)	0.5	2	2
M_4	(ECU_4, ECU_3)	1	2	1

transmission instance of message m_i on each link along r_m^i is specified according to a global schedule, $\mathcal{S}$.

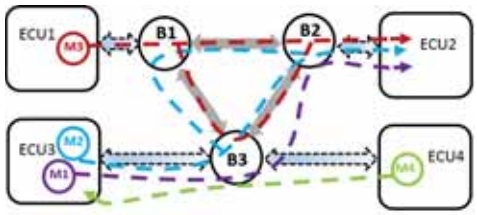
IV. Motivation

Inter-dependency between the network topology and the traffic routing implies handling both tasks jointly. In fact, joint topology and routing synthesis strategy is adapted to generate fault-resilient topology in [3] for UBS traffic. In contrast to this work, we consider TT traffic which requires an overlap-free schedule. Hence, ignoring the schedule feasibility during topology synthesis may result in an infeasible solution which requires re-routing or even re-synthesis of the topology. We will denote the strategy of synthesizing the schedule in different step as *separate-synthesis*.

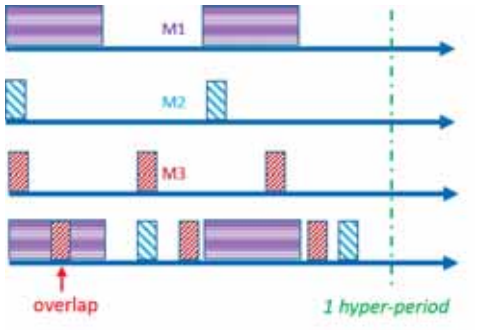
To further explain this problem, let us consider the following set of ECUs $\mathcal{E} = \{ECU_1, ECU_2, ECU_3, ECU_4\}$ to be connected. The available bridges library $\mathcal{A}$ is depicted in Table I where the physical links are identical with a transmission rate of 1 Mbit/s. The set of TT messages $\mathcal{M}$ to be delivered by the network is defined in Table II. It is required to synthesize a network topology that satisfies the following constraints: i) feasible routing, i.e., each message is routed through disjoint paths according to the predefined TL, and ii) feasible schedule, i.e., all messages replicas should be scheduled as described in Section III.

The solution obtained by *separate-synthesis* strategy is depicted in Fig. 1(a). The physical link $\langle B_3, B_2 \rangle$ carries the heaviest load which occupies 75% of the bandwidth. Apparently, the synthesized topology and routing fulfill the fault-resilience as well as bandwidth constraints. However, finding a schedule for this configuration is infeasible as illustrated in Fig. 1b since message M_1 does not fit with M_2 and M_3 in link $\langle B_3, B_2 \rangle$. In fact, re-routing is not sufficient for this case. Hence, the topology has to be re-synthesized in order to get a feasible schedule.

Handling the topology, routing and schedule synthesis jointly can get ride of the feasibility problem. In fact, solving these three problems jointly using either ILP or SMT formulation is possible since the formulation for each problem already exists as in [3], [11], and [7]. However, this approach is not capable of handling real problems with large $\mathcal{E}$ and $\mathcal{M}$. Thus, we propose a greedy heuristic algorithm for joint topology, routing and schedule synthesis (JTRSS). The synthesized solution by JTRSS algorithm for our example is shown in Fig. 1(b). A new bridge B_4 is added with two links to satisfy the scheduling constraint.



(a) The topology generated by *separate-synthesis*.



(b) Messages timeline for infeasible schedule.

Fig. 1: Example on separate-synthesis leads to infeasible schedule.

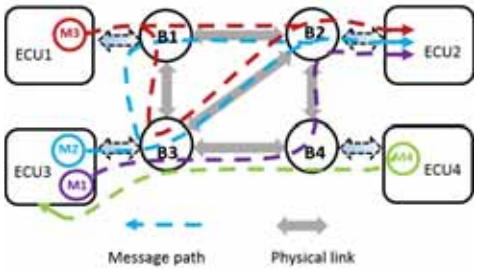


Fig. 2: The topology generated by JTRSS algorithm

V. Proposed Joint Topology, Routing, and Schedule Synthesis (JTRSS)

In this section, we present the proposed fault-resilient JTRSS technique for IEEE 801.2Qbv based networks.

A. Topology Synthesis

We formulate the synthesis of network topology as a labeling problem based on the graph model. In particular, consider the set $\mathcal{B} = \{b_1, b_2, b_3, ..., b_K\}$ and the set $\mathcal{L} = \{l_1, l_2, l_3, ..., l_N\}$ that represent the bridges in the network topology and the links which connect these bridges, respectively. K is an upper bound for the number of bridges specified by the user and N is the number of links in a fully-connected network which equal to K (K- 1)/ 2. Hence, the set $\mathcal{D} \in \mathcal{B} \cup \mathcal{L}$ represents the decision elements that can be parts of the network topology. Then, let the vector $\mathcal{T} = \{t_{d_1}, t_{d_2}, t_{d_3}, ...t_{d_{(K+N)}}\}$ be a labeling vector whose

element t_{d_i} represents the assignment of a label to the element $d_i \in \mathcal{D}$. Each element $t_{d_i} \in \mathcal{T}$ specifies the type of d_i from the library $\mathcal{A}$, whereas $t_{d_i} = \emptyset$ means that t_{d_i} is discarded element. Hence, the vector $\mathcal{T}$ describes the synthesized network topology.

The network topology, as well as the traffic configuration, are optimized according to the cost function $\mathcal{C}(\mathcal{T})$ which is formulated to incorporate both monetary cost of the topology, the delay of the messages, and the traffic schedulability, as expressed in Eq. (1).

$$\mathcal{C}(\mathcal{T}, \mathcal{R}, \mathcal{S}) = \mathcal{C}_{cost}(\mathcal{T}) + \alpha \cdot \mathcal{C}_{hops}(\mathcal{R}) + \mathcal{C}_{overlap}(\mathcal{S}) \quad (1)$$

where the coefficient α is a non-negative weighting parameter that controls the relative importance of $\mathcal{C}_{cost}(\mathcal{T})$ compared to $\mathcal{C}_{hops}(\mathcal{R})$. The monetary cost term $\mathcal{C}_{cost}(\mathcal{T})$ quantifies total topology cost as in Eq. (2).

$$\mathcal{C}_{cost}(\mathcal{T}) = \sum_{d_i \in \mathcal{D}} \mathcal{A}\{t_{d_i}\} \quad (2)$$

where $\mathcal{A}\{td_i\}$ is the cost of element d_i according to the library $\mathcal{A}$, while the discarded element does not count, i.e., $\mathcal{A}\{\emptyset\} = 0$. Second term $\mathcal{C}_{hops}(\mathcal{R})$ aims to minimize the messages delay by encouraging routing the traffic through shorter paths. In particular, $\mathcal{C}_{hops}(\mathcal{R})$ is equal to the summation of total length of all paths in $\mathcal{R}$. Given that, we assume the links are identical in terms of speed, then the number of hops also represents messages delay. Last term $\mathcal{C}_{overlap}(\mathcal{S})$ ensures the schedulability of $\mathcal{M}$, i.e., $\mathcal{C}_{overlap} = 0$ if a feasible overlap-free schedule is exist and $\mathcal{C}_{overlap} = \infty$ otherwise.

Synthesizing a network that achieves the global minimum of $\mathcal{C}(\mathcal{T}, \mathcal{M})$ is NP-hard and impractical for large-scale problems. Instead, the proposed algorithm constructs the solution iteratively by handling one message by iteration such that the incremental cost $\Delta_i\mathcal{C}$ expressed as Eq. (3) is minimized.

$$\Delta_i\mathcal{C} = \mathcal{C}_i - \mathcal{C}_{i-1} \quad (3)$$

where $\mathcal{C}_i$ is the value of the cost function after iteration *i*. To achieve this goal, we map the cost function $\Delta_i\mathcal{C}$ as path costs in a weighted undirected graph in which the minimum-cost path between *m.src* and *m.dest* implies $\min\{\Delta_i\mathcal{C}\}$. Hence, the incremental synthesis problem in each iteration is considered as a problem of finding the shortest path which can be solved efficiently using Yen's algorithm.

The main steps of the proposed JTRSS technique are illustrated in Algorithm 1. The inputs are the set of ECUs $\mathcal{E}$, the set of TT messages $\mathcal{M}$, the component library $\mathcal{A}$ and the initial number of bridges $N_{\mathcal{B}}$. Every iteration, JTRSS algorithm modifies the topology by changing a set of labels in $\mathcal{T}$. It is worth noting that the iterative labeling during the algorithm evolution can either activate links and bridges as new parts of the network topology or modify the bridges types to other types that support more ports. Hence, the topology is iteratively constructed while determining the routing and scheduling of messages in $\mathcal{M}$ one by one. As the algorithm finishes, the labeling vector $\mathcal{T}$, $\mathcal{R}$, and $\mathcal{S}$ will represent the synthesized topology, routing, and schedule, respectively.

Algorithm 1: Greedy Heuristic for Joint Topology, Routing, and Schedule Synthesis

```
   Input  : {E, M, A, N_B}
   Output: {T, R, S}
1  W ← W_init
2  R ← ∅, S ← ∅
3  M_ψ ← orderMessages(M)
4  while M_ψ ≠ ∅ do
5      W_m ← removeJointPaths(W, R_m)
6      flag ← 0, k ← 0
7      while flag == 0 do
8          k ← k + 1
9          r_m^k ← findNextPath(G(V, W_m), m, k)
10         [S, flag] ← assignMessage(S, m, r_m^k)
11     end
12     R_m ← R_m ∪ r_m
13     L_tolerance^m ← L_tolerance^m + 1
14     if L_tolerance^m == m.tl then
15         remove m from M_ψ
16     end
17     {W, T} ← updateWeights(W, T, r_m)
18 end
```

The first step in the algorithm is to determine an ordered list of messages $\mathcal{M}_\psi$ using the function *orderMessages* in line 3. Two criteria are followed to order $\mathcal{M}$: i) frame frequency where the messages with more frequent frames are assigned first since the messages with lower frequency are easier to schedule; and ii) frame size where the frames of the same frequency, larger frames are assigned first since the smaller ones are easier to schedule.

For every message in $\mathcal{M}_\psi$, JTRSS in each iteration finds one path which is disjoint with the selected paths in previous iterations. Given that R_m is the set of the selected paths for message *m*, the function *removeJointPaths* generates a message-specific arcs set $\mathcal{W}_m$. In this set, the links belong to R_m are disabled by having *infinite* weights. This ensures that the new generated path is disjoint with all links in R_m. Then, the function *findNextPath*, which based on Yen's algorithm [8], finds r_m^k which is the k^{th} shortest path for message *m* through $G(\mathcal{V}, \mathcal{W}_m)$. The function *assignMessage* searches for available time slots along r_m^k to schedule *m*. In case *m* is scheduled, $\mathcal{S}$ is updated, the flag is activated to stop the searching and the path r_m^k is added to the routing set R_m for message *m*. Otherwise, the term $\mathcal{C}_{overlap}(\mathcal{S}) = \infty$, thus, the message is tried to be scheduled in the next shortest path. Then, $\mathcal{M}_\psi$ is updated, i.e., if *m* reached the desired tolerance level, it is removed from $\mathcal{M}_\psi$ as depicted in lines 14 to 16.

B. Adaptive Weighting and Labeling

As new route r_m is added to the routing set, $\mathcal{R}$, which in turn changes the network topology, the arcs weights $\mathcal{W}$ are updated to represent the next $\Delta_{i+1}\mathcal{C}$. In particular, the function *updateWeights* is responsible for $\mathcal{W}$ updating as shown in line 17 in Algorithm 1.

Algorithm 2: The function $updateWeights()$

Input : $\{\mathcal{W}, \mathcal{T}, r_m\}$
Output: $\{\mathcal{W}^*, \mathcal{T}^*\}$
1 $\bar{\mathcal{T}} \leftarrow activateElements(\mathcal{T}, r_m)$
2 $\mathcal{T}^* \leftarrow upgradeBridges(\bar{\mathcal{T}}, r_m)$
3 $\bar{\mathcal{L}} \leftarrow findOpenLinks(\mathcal{T}^*, r_m)$
4 $\forall l \in \bar{\mathcal{L}} : w_l^* \leftarrow \infty$
5 $\mathcal{W}^* \leftarrow updateSelectionCost(\mathcal{W}, \mathcal{T}^*, \mathcal{A})$

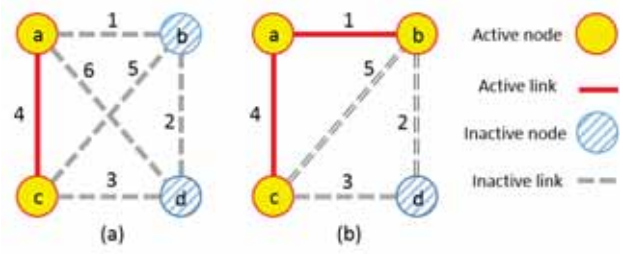


Fig. 3: Example on weights updating.

The main steps in the implementation of the *updateWeights* function are shown in Algorithm 2. The function *activateElements* finds and activates the bridges and links that are used for the first time. Then the function *upgradeBridges* finds and changes the type of the already active bridges that get an additional port in the current iteration. After that, the function *findOpenLinks* is responsible of finding the discarded arcs $\bar{\mathcal{L}}$ that cannot carry massages due to the limited number of ports supported by the bridges in $\mathcal{A}$. The weight of those arcs w_l^* are assigned to be ∞ which prevents using them in the next iteration. Finally, the function *updateSelectionCost* computes $\Delta_{i+1}\mathcal{C}_w$ for each arc $w \in \mathcal{W}$.

In order to illustrate the mechanism of the function *updateWeights*, consider the graph example depicted in Fig. 3. In particular, Fig. 3(a) shows the network status before updating the weights given that the added path in this iteration is $r_m = \langle a, b \rangle$ and the bridge type available in $\mathcal{A}$ supports only two ports. The function *activateElements*, in line 1, finds that constructing the path r_m implies activating bridge *b* and link l_1. Then, the function *findOpenLinks* adds the link l_6 to $\bar{\mathcal{L}}$ since bridge *a* cannot upgrade to support more than two ports according to $\mathcal{A}$. Hence, the arc that connects between node *a* and *d* is removed in line 3 i.e., $w_6^* = \infty$. The function *updateSelectionCost* changes the cost of selecting links $\{l_1, l_2, l_5\}$ to become as the following: $w(l_1) = \alpha$ since it is already part of the network. $w(l_2) = b_{cost} + l_{cost} + \alpha$, where b_{cost} and l_{cost} are the monetary cost of the link and the bridge since selecting l_2 in the next iteration implies activating the bridge *d* as well as the link l_2. Finally, $w(l_5) = l_{cost} + \alpha$ since both bridges *b* and *c* are already activated. Hence, selecting l_5 in the next iteration implies activating a single link.

VI. Experimental Results

In this section, we evaluate the scalability of the proposed JTRSS algorithm in terms of runtime with different numbers of ECUs and messages. Furthermore, we show the efficiency of JTRSS with respect to the topology cost and the scheduling feasibility comparing to two different approaches. In this analysis, 380 synthetic test cases are generated to perform the evaluation. MATLAB 2014a is employed to implement the algorithm. The reported results have been carried out on a workstation with an Intel Core i7 6820HQ processor running at 3.0 GHz and 16 GB RAM.

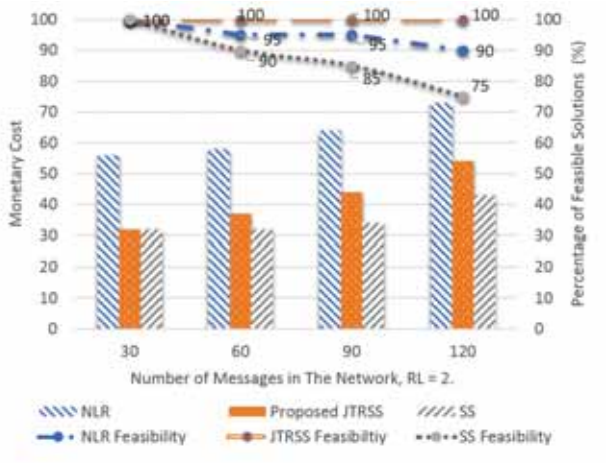
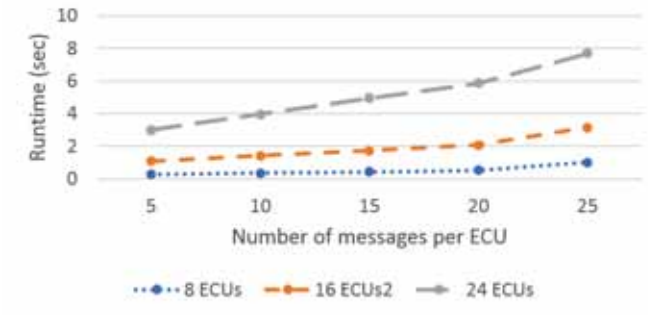


Fig. 4: Average cost and the percentage of feasible scheduling for 80 synthetic test cases with different number of messages randomly distributed among 6 ECUs.

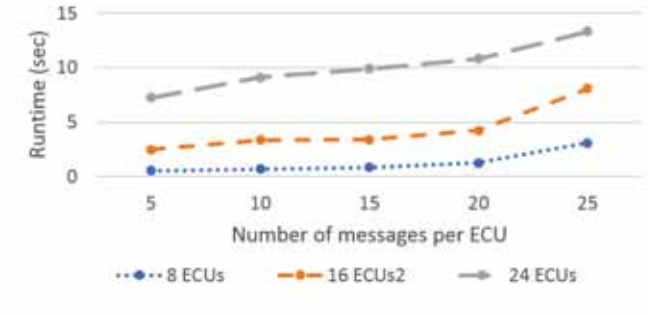
A. Efficiency Evaluation

First, we evaluate the efficiency of the proposed algorithm by comparing it with two different approaches. The first approach is based on Network-Level Redundancy (NLR) strategy as described in [14]. This approach is based on realizing the multi-path transmission by using multiple copies of the network topology. The second approach is based on the Separate-Synthesis (SS) strategy described in Section 4. In fact, this approach is influenced by the work introduced in [3]. To compare these approaches, we generate 80 synthetic test cases with a various number of messages (30, 60, 90, and 120) randomly distributed among 6 ECUs. All messages have a payload of 1,500 Bytes, a period of (1, 2, 3, and 10ms) and a required $TL = 2$. For all experiments, the links speed is 100 Mbit/s and the bridges library $\mathcal{A}$ in Table I is used.

The average cost and the percentage of feasible scheduling are depicted in Fig. 4. It can be noticed that the main drawback of NLR approach is the higher cost. Nevertheless, some cases did not provide feasible schedule since the routing and scheduling are not solved jointly. Whereas, SS approach provides the minimum topology cost since it does not consider the feasibility of messages scheduling. However, it suffers from a high percentage of the cases that lead to infeasible schedule. In this case rerouting of the traffic or even re-synthesize the topology by adding some new bridges and links may be required. On the other hand, the proposed JTRSS provides less cost than NLR approach while ensuring feasible scheduling.



(a) Tolerance level = 2



(b) Tolerance level = 3

Fig. 5: Average runtime for 300 synthetic test cases with varied numbers of ECUs and messages.

B. Scalability Evaluation

To evaluate the scalability of the proposed approach, we used 300 synthetic test cases to cover the following settings. The problems are composed of 8, 16, and 24 ECUs that exchange numbers of messages varied from 5 to 25 message per ECU. All messages have a payload of 1,500 Bytes, which is the Ethernet Maximum Transmission Unit (MTU), and a period of 1, 2, 3, and 10ms. Whereas, the required tolerance level is chosen to be 2 or 3. The source and destination of the messages are assigned randomly. The average runtime for the cases of each size is depicted in Fig. 5. The results show high scalability of the proposed JTRSS algorithm such that, it is capable of handling large-scale problem, e.g., 24 ECUs with 600 messages within very short runtime about 8 seconds in average. Furthermore, the linear increase of runtime with respect to the number of ECUs as well as the number of messages is pointing to a potential of handling real large-scale synthesis problem which will be investigated in future work.

VII. CONCLUSIONS

In this paper, we propose a fault resilient Joint Topology, Routing and Scheduling Synthesis JTRSS algorithm for IEEE 802.1Qbv TSN networks. The proposed algorithm is capable to generate fault-resilient topology that guarantee feasible routing and scheduling for TT traffic. JTRSS algorithm is based on constructing the topology iteratively by adding one path every iteration. The problem is modeled as a weighted undirected graph in which weights are updated every iteration, i.e., the selected path implies the minimum additional cost on the whole topology. Yen's algorithm is adapted to find the minimum-cost topology increment in each iteration as shortest path problem. JTRSS approach is evaluated using 380 synthetic test cases and compared with two fault-resilience approaches. The results show good scalability and better performance compared to NLR and SS approaches. In particular, proposed JTRSS outperform NLR approach in terms of cost minimization. Furthermore, JTRSS guarantees feasible scheduling in contrast to SS approach.

REFERENCES

[1] P. Pop, M. L. Raagaard, S. S. Craciunas, and W. Steiner, "Design optimisation of cyber-physical distributed systems using ieee time-sensitive networks," *IET Cyber-Physical Systems: Theory & Applications*, vol. 1, no. 1, pp. 86–94, 2016.

[2] "IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic," *IEEE Std*, March 2015.

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

[4] B. Wang and J. C. Hou, "Multicast routing and its qos extension: problems, algorithms, and protocols," *IEEE network*, vol. 14, no. 1, pp. 22–36, 2000.

[5] M. R. Garey and D. S. Johnson, "Computers and intractability: A guide to the theory of npcompleteness (series of books in the mathematical sciences), ed," *Computers and Intractability*, p. 340, 1979.

[6] F. Smirnov, M. Glaß, F. Reimann, and J. Teich, "Optimizing message routing and scheduling in automotive mixed-criticality time-triggered networks," in *Design Automation Conference (DAC)*. IEEE, 2017, pp. 1–6.

[7] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling real-time communication in ieee 802.1 qbv time sensitive networks," in *Proceedings of International Conference on Real-Time Networks and Systems*. ACM, 2016, pp. 183–192.

[8] J. Y. Yen, "An algorithm for finding shortest routes from all source nodes to a given destination in general networks," *Quarterly of Applied Mathematics*, vol. 27, no. 4, pp. 526–530, 1970.

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

[10] F. Smirnov, M. Glaß, F. Reimann, and J. Teich, "Formal reliability analysis of switched ethernet automotive networks under transient transmission errors," in *Design Automation Conference (DAC)*. IEEE, 2016, pp. 1–6.

[11] A. A. Atallah, G. Bany Hamad, and O. Ait Mohamed, "Reliability-aware routing of avb streams in tsn networks," in *Accepted International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE)*, June 25-28, 2018.

[12] S. M. Laursen, P. Pop, and W. Steiner, "Routing optimization of avb streams in tsn networks," *ACM SIGBED Review*, vol. 13, no. 4, pp. 43–48, 2016.

[13] "ILP-based joint routing and scheduling for time-triggered networks, author=Schweissguth, Eike and Danielis, Peter and Timmermann, Dirk and Parzyjegla, Helge and Mühl, Gero, booktitle=Proceedings of International Conference on Real-Time Networks and Systems, pages=8–17, year=2017, organization=ACM."

[14] B. Annighoefer, C. Reif, and F. Thieleck, "Network topology optimization for distributed integrated modular avionics," in *Digital Avionics Systems Conference (DASC)*, 2014, pp. 1–12.