

On Fault-tolerant Scheduling of Time Sensitive Networks

Radu Dobrin

Mälardalen University, Västerås, Sweden

radu.dobrin@mdh.se

Nitin Desai 

Mälardalen University, Västerås, Sweden

nitin.desai@mdh.se

Sasikumar Punnekkat 

Mälardalen University, Västerås, Sweden

sasikumar.punnekkat@mdh.se

Abstract

Time sensitive networking (TSN) is gaining attention in industrial automation networks since it brings essential real-time capabilities at the data link layer. Though it can provide deterministic latency under error free conditions, TSN still largely depends on space redundancy for improved reliability. In many scenarios, time redundancy could be an adequate as well as cost efficient alternative. Time redundancy in turn will have implications due to the need for over-provisions needed for timeliness guarantees. In this paper, we discuss how to embed fault-tolerance capability into TSN schedules and describe our approach using a simple example.

2012 ACM Subject Classification Computer systems organization → Dependable and fault-tolerant systems and networks

Keywords and phrases Time sensitive networks(TSN), Fault-tolerant schedule, Time redundancy

Digital Object Identifier 10.4230/OASICS.CERTS.2019.5

Acknowledgements The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 764785, FORA—Fog Computing for Robotics and Industrial Automation.

1 Introduction

Time- and safety-critical control applications in the context of factories need real-time guarantees [26]. Commonly, such requirements are specified at design time and the system is expected to fulfil them during its entire operational life. This necessitates *guaranteed* and *bounded* latencies and low jitter for tasks/functions that are critical to safety (for the end-user) [24]. The design and development of distributed embedded systems driven by the Time-Triggered paradigm [17] has proven effective in a diversity of domains with stringent demands of determinism [7].

There has been a steady evolution from centralized control with the control logic embedded within a single controller to decentralized/distributed control where control is shared between multiple controllers. A key benefit of this is to provide greater robustness to failures. For instance, a distributed architecture is more conducive to safety, by ensuring critical functions have the possibility of being executed at multiple physical nodes and transported across multiple communication links (the basic notion of redundancy). However, from a network latency perspective, this may cause additional latencies due to multiple hops (when an alternate link or node is needed). When timeliness is of the essence, such an arrangement may not therefore be optimal in providing determinism.



© Radu Dobrin, Nitin Desai, Sasikumar Punnekkat;
licensed under Creative Commons License CC-BY

4th International Workshop on Security and Dependability of Critical Embedded Real-Time Systems (CERTS 2019).

Editors: Mikael Asplund and Michael Paulitsch; Article No. 5; pp. 5:1–5:12

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

44 In applications such as factory automation or automobiles, the systems could be subjected
 45 to high degrees of Electro Magnetic Interference (EMI) from the operational environment
 46 which can cause transmission errors. The common causes for such interference include
 47 cellular phones and other radio equipment inside the premises/vehicle, electrical devices
 48 like switches and relays, radio transmissions from external sources and lightning in the
 49 environment. Complete elimination of the effects of EMI is hard since exact characterization
 50 of all such interference defy comprehension. Though usage of an all-optical network could
 51 greatly eliminate EMI problems, it may not be favoured by many cost-conscious industries.

52 These interferences cause errors in the transmitted data, which could indirectly lead
 53 to catastrophic failures. To reduce the risks due to erroneous transmissions, designers
 54 usually provide elaborate error checking and error confinement features in the protocol (as
 55 in Controller Area Networks). Basic philosophy of these features is to identify an error as
 56 fast as possible and then re-transmit the affected message. This implies that in systems
 57 without spatial redundancy of communication medium/controllers, the fault-tolerance (FT)
 58 mechanism employed is time redundancy. On the other hand, time redundancy increases the
 59 latency of message sets; potentially leading to violation of timing requirements. Hence any
 60 reliability management approaches in critical systems needs to be a holistic one incorporating
 61 both space and time redundancy at the right levels based on the system characteristics,
 62 resource constraints, fault models and trade-offs from cost-performance perspectives.

63 The time sensitive networking (TSN) [11] is a set of evolving standards under the IEEE
 64 working group IEEE802.1, defining protocols that extend standard Ethernet to achieve
 65 real-time networking capabilities for industrial/factory automation application scenarios.
 66 The TSN standardization efforts consists of a number of (sub)standards that aim to achieve
 67 four key technological paradigms - clock synchronization (802.1ASrev), frame preemption
 68 (802.1Qbu), scheduled traffic (802.1Qbv), and redundancy management (802.1CB). These
 69 must work together at the Ethernet layer (L2) to ensure that safety functions are executed
 70 while meeting their respective deadlines and constraints.

71 The 802.1Qbv TSN standard provides scheduled traffic for time-triggered safety-critical
 72 data frames in a predetermined manner. However, in the presence of faults, a static schedule
 73 cannot satisfy system requirements particularly since the schedule has to be reconfigured.

74 Redundancy management in TSN (802.1CB) has been mainly focussing on space (link
 75 redundancy). It is typical to start with a simplified error model assumption that only
 76 singleton errors can occur in the systems and that they are separated at least by a known
 77 minimum interarrival time.

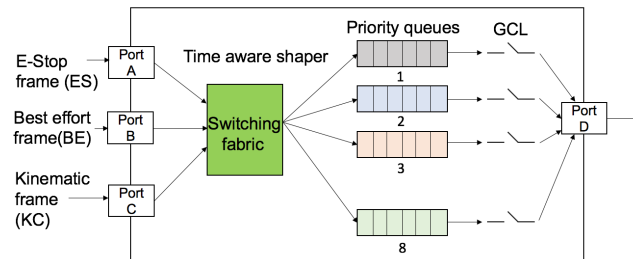
78 In this paper, our focus is on time redundancy and how to improve fault tolerance
 79 capability of the TSN schedule. The underlying assumptions and models in our work are
 80 in line with our previous work and also with that followed in [2] [1]. We extend our earlier
 81 works presented in [10] and adapt it to the context of TSN.

82 The rest of the paper is organised as follows. Section 2 introduces the reader to the
 83 concept of time sensitive networking. Section 3 describes our system model and outlines its
 84 basic components. In section 4 we detail the proposed approach. In section 5, we present an
 85 illustrative example for our approach. Section 6 discusses the research relevant and related
 86 to our work and finally we conclude with section 7 and provide some ideas for future work.

87 **2 Time Sensitive Networking**

88 The TSN standard is composed of a number of sub-standards. The most relevant sub-
 89 standard for our use-case is 802.1Qbv - *scheduled traffic*. In this use-case, we assume that the

90 802.1Qbv protocol is implemented both in the TSN switch as well as the control nodes and robots. A fundamental principle behind TSN [11] is the time-triggered protocol (TTP) [18].



91 ■ **Figure 1** TSN 802.1Qbv-enabled switch.

92 In Fig. 1, we have three ingress ports A, B, and C and a single egress port D. The safety
 93 frames (ES, KC) from control nodes are sent from ports A and B while port C sends
 94 best effort (BE) frames as shown. The Gate Control List (GCL) decides the exact times
 95 when the frames belonging to a specific priority queues will be allowed to pass through
 96 the egress port D. From a system safety perspective, ES and KC frames must be given
 97 higher priority than BE frames.

98 Such systems depend on redundant communication schedules that contain global time-
 99 based information of message transmissions with conflict-free paths through the switches.
 100 The static schedule of a time-triggered system maximizes predictability, while the schedule
 101 in an event-triggered network unfolds dynamically at run-time depending on the occurrence
 102 of events [18]. A time-triggered network ensures the partitioning of the system into a set
 103 of independent fault containment regions (FCR), which operate correctly regardless of an
 104 arbitrary fault outside the region.

105 **3 System model**

106 Having provided the background required for this use-case, the problem we tackle can be
 107 stated as follows:

108 *“How do you guarantee delivery of safety-critical data frames across a TSN enabled
 109 network in the presence of faults specified by a fault model?”*

110 In order to quantify relevant system parameters, we present a system model that is
 111 composed of sub-models that tackles each aspect of the system function.

112 **3.1 System and error model**

113 We assume a distributed real-time architecture consisting of sensors, actuators and processing
 114 nodes communicating over a time sensitive network. The communication is performed via
 115 a set of strict periodic messages, $\Gamma = \{M_1, M_2, \dots\}$, with mixed criticality levels. The
 116 criticality of a message indicates the severity of the consequences caused by its failure and
 117 corresponds to the amount of resources allocated for error recovery in terms of guaranteed
 118 re-transmissions. The basic assumption here is that the effects of a large variety of transient
 119 and intermittent (hardware) faults can effectively be tolerated by a simple re-transmission of
 120 the affected frames. We assume that a fault can adversely affect only one message frame at a
 121 time and is detected by all nodes in the network. Γ_c represents the subset of critical messages
 122 out of the original message set and Γ_{nc} represents the subset of non-critical messages, so
 123 that $\Gamma = \Gamma_c \cup \Gamma_{nc}$.

124 A message consists of N frames, $N \geq 1$, and the network communication is assumed
 125 to be non-preemptive during the frame transmissions. Though sub-standard 802.1Qbu is
 126 introducing preemption of frames in TSN, for simplicity's sake we have not considered it in
 127 current work. Of course, messages composed of more than 2 frames can preempt each other
 128 at frame boundaries. Additionally, the non-preemptiveness of message frames may cause a
 129 higher priority message to be blocked by a lower priority message on the same link for at
 130 most one frame length. This priority inversion phenomenon can affect all messages except
 131 the lowest priority one, and only once per message period, before the transmission of the
 132 first message frame [9].

133 Each message M_i is characterized by a 4-tuple $\langle T_i, D_i, N_i, R_i \rangle$, where T_i is the period,
 134 D_i is the relative deadline, N_i is the number of frames that form this message and R_i is the
 135 fault tolerant requirement in terms of the number of re-transmissions the message needs to
 136 be able to execute upon faults. Hence, the total number of frames that need to be guaranteed
 137 for re-transmission r_i is calculated by

$$138 \quad r_i = \lceil N_i * R_i \rceil \quad (1)$$

139 Note that for non-critical messages $R_i = 0$. Additionally, rate constrained and best effort
 140 messages have a priority P_i .

141 In an error-free scenario, the worst case transmission time C_i of message M_i is

$$142 \quad C_i = N_i * f * \tau_{bit} \quad (2)$$

143 where f is the maximum frame size and τ_{bit} is the transmission time for a bit.

144 Each *message instance* M_i^j is characterized by a *feasibility window* delimited by its earliest
 145 start time $est(M_i^j)$ and its deadline D_i^j .

146 Obviously, in order to be able to guarantee the specified fault tolerance requirements,
 147 the maximum network utilization of the critical messages together with their required re-
 148 transmissions can never exceed 100% of the bandwidth capacity. This will imply that, during
 149 the error recovery, non-critical message transmissions may need to be shed in order to avoid
 150 overload conditions.

151 3.2 Traffic model

152 Real-time traffic in control systems is highly regular and periodic. The schedules for such
 153 traffic can be statically synthesized during design phase. This plan may not only define the
 154 communication paths and bandwidth reservations, but also particular points in a network-
 155 wide reference time at which messages are to be transmitted. Such a plan that incorporates
 156 the time aspects is called a “communication schedule” and the execution of the schedule by
 157 the network obeys the time triggered approach.

158 Safety critical messages are usually transmitted through TT class since bounded delivery
 159 latency is guaranteed [24].

160 A basic fault tolerance mechanism in the presence of faults is to re-transmit an alternate of
 161 the original message at a later time instant. This is suitable for single errors during message
 162 transmissions. It is also assumed that no errors affect the alternate message transmission.
 163 For simple cases one could consider the re-transmission of the original message itself, but the
 164 approach could as well cater to initiation of another alternate task leading to an alternate
 165 message (for example in critical scenarios warranting an ‘emergency stop’).

4 Proposed approach

Our research objective is to provide efficient and fault-tolerant scheduling algorithms and mechanisms for TSN that ensure:

1. All safety critical messages (time triggered traffic) have guaranteed correct delivery within their deadlines under given fault assumptions
2. All non-critical traffic is given best-effort schedulability guarantees
3. The generated schedules also possess flexibility to incorporate evolving changes traffic patterns particularly in the absence of faults

We make the following assumptions to start with:

1. A fault can affect only one message at a time
2. A specified number of re-transmissions of the message is sufficient to overcome the effect of a transient or intermittent fault.
3. There exist sufficient fault detection capabilities (such as watchdog timers, CRC etc.) in the system so that a fault can be detected reliably within a specified short time interval.
4. There exist ARQ mechanisms so that the sender node is able to know within a specified time whether the message sent has reached the destination (or intermediate node) correctly.

Our ongoing research efforts aims at providing specific contributions in the following directions:

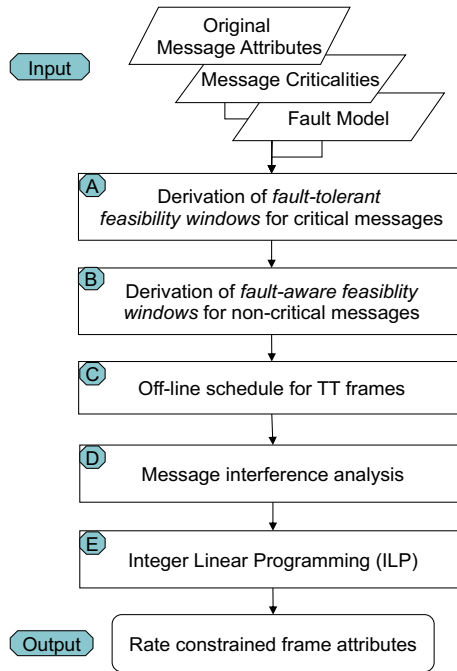
1. The use of phased re-transmissions that can achieve better bandwidth utilization than possible with the approach of Alvarez et. al. [1].
2. Combined scheduling of critical and non-critical messages using the concept of fault-tolerant windows and fault aware windows
3. Making more realistic fault model assumptions
4. Making our schemes more flexible to support evolution of systems
5. Suggesting mechanisms for implementation for induction into standards

The focus of current paper is only first two items above. Here we propose an approach to jointly schedule critical and non-critical messages as time triggered and rate constrained traffic in TSN. We propose to schedule critical messages with completely known attributes as time triggered traffic. Some critical messages, however, may have requirements that cannot be accommodated in an off-line schedule a-priori. These are, instead, scheduled as rate constrained (RC). It is essential, however, to schedule them at priority levels that guarantee their re-transmissions in case of faults. At the same time, we aim to provide the non-critical best effort traffic the best possible service in case the system is not overloaded due to faults.

The key concept in our proposed approach is the derivation of the feasibility windows for the message transmissions. Traditionally the feasibility window for a message is the time interval between its earliest start time (or release time) and its deadline. These parameters, however, do not typically express the fault tolerant requirements on the critical messages, e.g., a message transmission finishes just before its deadline, will not leave enough time for a feasible (before its deadline) re-transmission in case the message is hit by a fault. We propose the derivation of new feasibility windows for each message instance $M_i^j \in \Gamma$ that reflect the FT requirements.

While transmitting non-critical messages using a background priority band can be a safe and straightforward solution, our aim is to provide non-critical messages a better service than what can be achieved through background scheduling. Hence, depending on the criticality of the original set of messages, the new feasibility windows we are looking for differ as:

1. *Fault-Tolerant* (FT) feasibility windows for critical messages
2. *Fault-Aware* (FA) feasibility windows for non-critical messages



■ **Figure 2** Methodology overview

212 While critical messages need to be entirely transmitted within their FT feasibility windows
 213 to be able to be feasibly re-transmitted upon an error, according to the reliability requirements,
 214 the derivation of FA feasibility windows has two purposes: 1) to prevent non-critical messages
 215 from interfering with critical ones thus causing a critical message to miss its deadline, while
 216 2) enabling the transmission of the non-critical messages at high priority levels in error free
 217 situations.

218 The major steps of the proposed methodology are shown in Figure 2. The inputs to the
 219 method are message attributes, criticalities and fault model in terms of frequency of faults
 220 and fault-tolerance requirements.

221 Since the size of the FA feasibility windows depends on the size of the FT feasibility
 222 windows, in our approach we first derive FT-feasibility windows and then FA feasibility
 223 windows (as steps A and B Figure 2). Then, we assign time slots for TT traffic and priorities
 224 to rate constrained traffic to ensure the message transmissions within their newly derived
 225 feasibility windows.

226 Subsequently we generate an off-line schedule for the TT traffic (in step C) followed
 227 by assigning message identifiers (priorities) for the rate constrained traffic (in step D) that
 228 ensure the message transmissions within their new feasibility windows, thus, fulfilling the FT
 229 requirements. We generate an offline schedule for the TT messages, by using the Earliest
 230 Deadline First (EDF) heuristics and provisioning for the specified number of re-transmission
 231 upon faults. Then we identify the optimal priorities for the critical rate based traffic in order
 232 to ensure its coexistence with the TT traffic, as well as its FT requirements. At the same
 233 time, we derive the priorities for the non-critical messages that ensure their timeliness in the
 234 absence of faults. As the network utilisation will heavily increase due to the re-transmissions
 235 of the critical messages under faults, we assume that in these situations the non-critical
 236 messages are shed by their sending nodes.

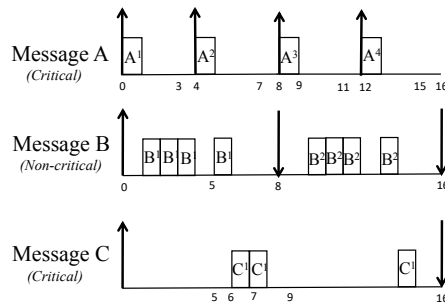
237 In some cases, however, a fixed priority scheme cannot express all our assumed FT
 238 requirements and error assumptions on the rate constrained traffic. General FT requirements
 239 may require that instances of a given set of periodic messages needs to be transmitted in
 240 different order on different occasions. Obviously, there exists no valid fixed priority assignment
 241 that can achieve these different orders. Our approach proposes a priority allocation scheme
 242 based on EDF at message instance level that efficiently utilizes the resources while minimizing
 243 the priority levels. We use Integer Linear Programming (ILP) (Step E) to off-line analyze
 244 the interference between the message frames and to derive the minimum number of fixed
 245 priorities that guarantees the message transmissions within their FT/FA Feasibility Windows.

246 **5 Discussions**

247 We discuss our approach by resorting to a simple but instructive example detailed below.

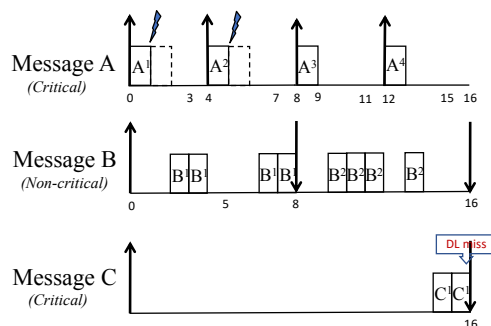
248 A set of three messages A, B , and C are considered, wherein A and C are critical and B
 249 is non-critical with periods $T(A) = 4$, $T(B) = 8$, $T(C) = 16$ and transmission times, $C(A)$
 250 $= 1$, $C(B) = 4$, $C(C) = 3$. We assume the deadlines for the messages equal their periods.
 251 We re-transmit only critical messages when subject to a single fault per message instance.
 252 Fig. 3 shows a feasible message transmission under the assumption of 'no faults'.

253 Our proposed approach is illustrated in a set of figures depicting various scenarios and
 254 schedules. As part of our motivation for the proposed fault tolerant windows based approach,
 255 we first show the Rate monotonic(RM) schedule for the message transmissions in Fig. 3.



■ **Figure 3** RM-based schedule with no faults - but not an FT schedule

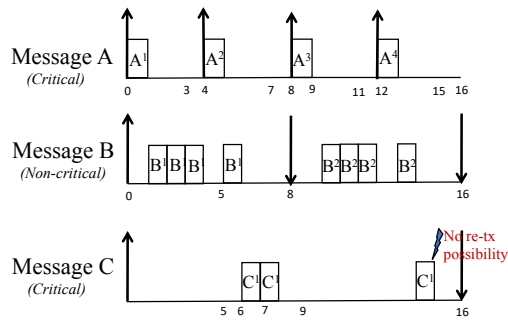
256 Fig. 4 shows the infeasibility of the critical message C in case 2 instances of A a hit by
 257 faults and need to be re-transmitted.



■ **Figure 4** Two faults on message A causing even primary of critical message C to miss deadline.

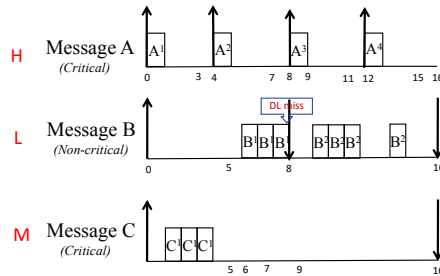
5:8 On Fault-tolerant Scheduling of Time Sensitive Networks

258 Fig. 5 shows that that the critical message C cannot even tolerate a single fault.



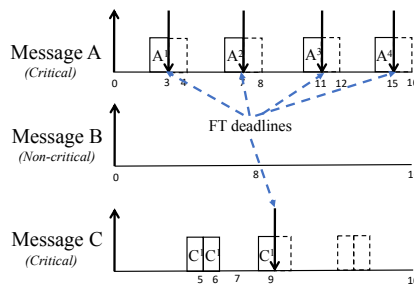
■ **Figure 5** A single fault in message C prevents re-transmission possibilities.

259 A solution could be, however, to increase the priority of the critical message C above
 260 the priority of the non-critical message B. In this case, however, the first instance of B will
 261 always miss its deadline, even in a fault free scenario (Fig. 6).



■ **Figure 6** Priority modification (non-RM) still causing B to miss deadline.

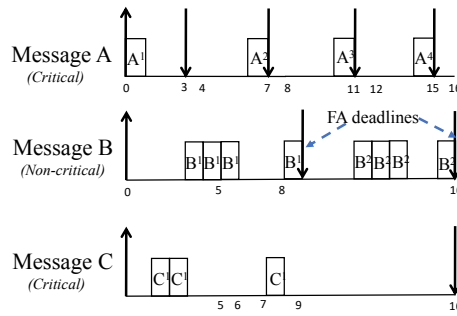
262 Fig. 7 illustrates the derivation of the fault tolerant (FT) windows for critical messages A
 263 and C. The dashed boxes represent the re-transmissions that would be needed if the critical
 264 messages were to experience a single fault per instance.



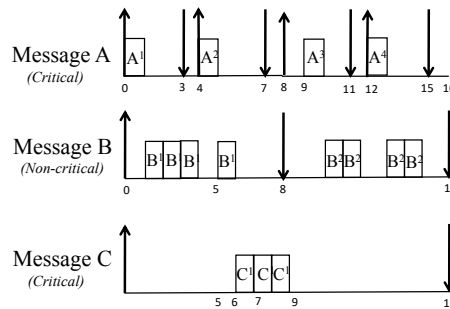
■ **Figure 7** Derivation of FT windows for critical messages.

265 Fig. 8 shows the derivation of fault aware (FA) windows for non-critical message (B).
 266 This is done after the fault tolerant windows for the critical messages have been calculated.
 267 Fig. 9 shows a resulting schedule which would meet all deadlines under a fault free scenario.

268 Fig. 10 illustrates the benefits provided by our approach. The critical message A is
 269 transmitted in a time-triggered (TT) manner while the other messages are assigned to the

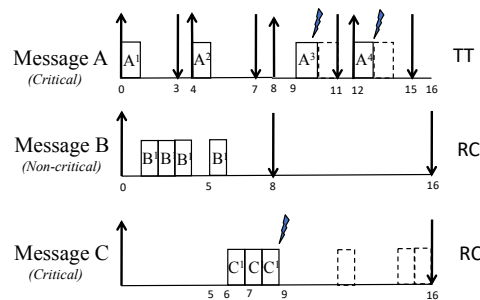


■ **Figure 8** Derivation of fault-aware windows for non-critical messages.



■ **Figure 9** Fault free messages with no deadline misses.

270 rate-constrained traffic class (RC). We have three faults occurring on the critical messages.
 271 Our scheduling approach ensures that all critical messages are scheduled in a fault tolerant
 272 manner while only one instance of the non-critical message fails to meet the deadline.

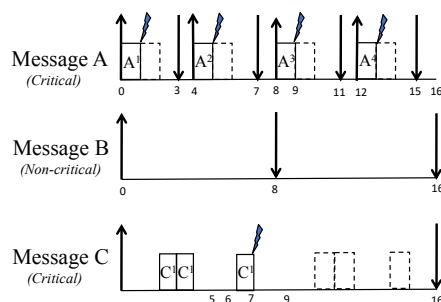


■ **Figure 10** Three faults with all critical messages scheduled and only one instance of non-critical message un-schedulable.

273 Finally, Fig. 11 shows a scenario with 5 faults. Critical messages A and C still remain
 274 fault tolerant while the non-critical message B is prevented from execution. However, in the
 275 event that critical messages do not experience faults, the non-critical message B can still
 276 meet its deadline, thereby providing a better service than background scheduling.

277 In summary, the above example scenarios shows that:

- 278 ■ For a given message set A,B,C where A and C are critical, a RM priority assignment
 279 ($A > B > C$) will not guarantee the 100% FT (i.e. one re-transmission upon a potential



■ **Figure 11** Each critical message instance with a fault can be re-transmitted while meeting deadlines.

280 fault per message instance). C will miss its deadline.
 281 ■ A new priority ordering where the non-critical message has a lower priority than the
 282 critical message ($A > C > B$) will guarantee the 100% FT of the critical messages but B
 283 will miss its first deadline even in a fault free scenario

284 What we propose is a new stream/message allocation and priority assignment that:

- 285 ■ Maximises the FT capability of critical messages in the presence of fault
- 286 ■ Maximises the service of the non-critical messages in the absence of faults

287 We derive FT feasibility windows and FA feasibility windows based on Chetto and Chetto [6]
 288 and further Aysan *et al* [4] that ensure the above. In the example, we put A in the TT traffic
 289 and B and C in the RC traffic with priorities derived by an ILP solver given the feasibility
 290 windows constraint.

291 6 Related work

292 For scheduling on time-triggered networks, Steiner [22] introduced a method to synthesize
 293 the time-triggered traffic using an SMT YICES solver. A common approach is to have a
 294 “recovery slack” in the schedule in order to accommodate time needed for re-executions in
 295 case of faults [10]. It has been shown that the time-triggered paradigm which forms a core
 296 part of the time sensitive network standard (as 802.1Qbv time-aware shaper [23]) ensures
 297 the *fail-silent semantics* whereby a packet is received only if correct or not received at all.
 298 Fault recovery studies such as [25] depend on a fast spanning tree reconfiguration algorithm
 299 to reduce the total fault recovery time, and a delayed link inactivation scheme that allows
 300 real-time connections which are not affected by the failed links/switches to continue to exist.

301 Recent approaches by Steiner *et al* [21][13] based on reconfiguration of GCL schedules
 302 at runtime for 802.1Qbv TSN discuss a configuration agent that is aware of the traffic
 303 conditions at each node in the network. The objective is to ensure that new traffic flows can
 304 be accommodated with use of as few queues in the switch ports as possible while maintaining
 305 a feasible schedule.

306 Proenza *et al* [5] [19] have proposed a *flexible* time triggered paradigm for distributed
 307 real time systems. Flexibility refers to the adaptation of the nodes to new and evolving hard
 308 real-time requirements such as periodic and sporadic messages and updating the parameters
 309 of such messages at run-time. Some recent studies on reconfiguration by means of spatial
 310 and temporal techniques are discussed in [2] [3]. Desai *et al* [8] discuss safety of industrial
 311 automation systems, although focused towards fog/edge paradigms. Pozo *et al* [20] have
 312 shown that schedules can be “repaired” to combat the presence of faults.

313 Recovery from a transient or permanent fault in a time triggered network implies a
 314 certain amount of flexibility in the mechanisms to ask for changes in real-time requirements
 315 at runtime to reconfigure nodes and switches according to a new schedule [12]. Gutiérrez *et*
 316 *al* [14] and Raagard *et al* [21] discuss a configuration agent that can synthesize new schedules
 317 for TSN at runtime.

318 Time synchronization aspects are also extremely crucial and addressed in works such as
 319 [16] and [15].

320 **7 Conclusion and Ongoing Work**

321 Scheduling of safety-critical data frames (and tasks) constitutes a fundamental design
 322 requirement. The principal limitation of the time-triggered approach is the inability to
 323 adapt to unanticipated changes in the system parameters such as traffic patterns or faults.
 324 This causes the schedule not to guarantee the transmission of all frames within their timing
 325 requirements. If the network does not contain a backup schedule predicting that specific
 326 change, the schedule needs to be synthesized again from scratch, which is computationally
 327 and time intensive.

328 With respect to 802.1Qbv, our goal is to ensure that the schedule offsets representing
 329 the opening and closing of the gates (the GCL table) for the TSN switches are recalculated
 330 first for the TT traffic while simultaneously meeting the timing requirements (deadlines) for
 331 message transmissions.

332 In this paper we saw how our FT/FA aware scheduling approach provides critical messages
 333 to meet their deadlines even when all instances of the critical messages experience single
 334 faults. Additionally, in case faults do not occur, we have the possibility for non-critical
 335 messages to be served in a better way (compared to background scheduling). We are currently
 336 in the process of performing detailed evaluation of the approach thorough simulations.

337 As part of ongoing work, we are focusing on transmitting critical messages as time-
 338 triggered traffic which will enforce a much stricter time assignment.

339 **References**

- 340 **1** I. Alvarez, C. Drago, J. Proenza, and M. Barranco. First study of the proactive transmission of
 341 replicated frames mechanism over TSN. *16th International Workshop on Real Time Networks*
 342 *(RTN), ECRTS*, 2018.
- 343 **2** I. Álvarez, J. Proenza, and M. Barranco. Mixing Time and Spatial Redundancy Over Time
 344 Sensitive Networking. In *IEEE/IFIP International Conference on Dependable Systems and*
 345 *Networks Workshops (DSN-W)*, June 2018. doi:10.1109/DSN-W.2018.00031.
- 346 **3** M. Ashjaei, P. Pedreiras, M. Behnam, L. Almeida, and T. Nolte. Evaluation of dynamic
 347 reconfiguration architecture in multi-hop switched ethernet networks. In *IEEE Emerging*
 348 *Technology and Factory Automation*, pages 1–4, Sep. 2014. doi:10.1109/ETFA.2014.7005322.
- 349 **4** H. Aysan, R. Dobrin, and S. Punnekkat. Fault tolerant scheduling on control area net-
 350 work (CAN). *IEEE International Workshop on Object/component/service-oriented Real-time*
 351 *Networked Ultra-dependable Systems*, 2010.
- 352 **5** A. Ballesteros, J. Proenza, and P. Palmer. Towards a dynamic task allocation scheme for
 353 highly-reliable adaptive distributed embedded systems. In *2017 22nd IEEE International*
 354 *Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4, Sep. 2017.
 355 doi:10.1109/ETFA.2017.8247773.
- 356 **6** H. Chetto and M. Chetto. Some results of the earliest deadline scheduling algorithm. *IEEE*
 357 *Transactions on Software Engineering*, 15(10), Oct 1989. doi:10.1109/TSE.1989.559777.

- 358 7 S.S. Craciunas and R.S. Oliver. Combined task- and network-level scheduling for distributed
359 time-triggered systems. *Real-Time Systems Journal*, 52(2), March 2016.
- 360 8 N. Desai and S. Punnekkat. Safety of fog-based industrial automation systems. In *Proceedings*
361 *of the Workshop on Fog Computing and the IoT*. ACM, 2019. doi:10.1145/3313150.3313218.
- 362 9 M. Di Natale. Scheduling the CAN Bus with Earliest Deadline Techniques. *IEEE Real-Time*
363 *Systems Symposium*, pages 259–268, November 2000.
- 364 10 R. Dobrin, H. Aysan, and S. Punnekkat. Maximizing the fault tolerance capability of fixed
365 priority schedules. In *RTCSA*, pages 337–346, 09 2008. doi:10.1109/RTCSA.2008.6.
- 366 11 N. Finn. Introduction to Time-Sensitive Networking. *IEEE Communications Standards*
367 *Magazine*, 2(2):22–28, JUNE 2018. doi:10.1109/MCOMSTD.2018.1700076.
- 368 12 D. Gessner, J. Proenza, M. Barranco, and A. Ballesteros. A Fault-Tolerant Ethernet for Hard
369 Real-Time Adaptive Systems. *IEEE Transactions on Industrial Informatics*, pages 1–1, 2019.
370 doi:10.1109/TII.2019.2895046.
- 371 13 M. Gutiérrez, A. Ademaj, W. Steiner, R. Dobrin, and S. Punnekkat. Self-configuration of IEEE
372 802.1 TSN networks. In *2017 22nd IEEE International Conference on Emerging Technologies*
373 *and Factory Automation (ETFA)*, pages 1–8, Sep. 2017. doi:10.1109/ETFA.2017.8247597.
- 374 14 M. Gutiérrez, W. Steiner, R. Dobrin, and S. Punnekkat. A configuration agent based on the
375 time-triggered paradigm for real-time networks. In *2015 IEEE World Conference on Factory*
376 *Communication Systems (WFCS)*, pages 1–4, May 2015. doi:10.1109/WFCS.2015.7160584.
- 377 15 Marina Gutiérrez, Wilfried Steiner, Radu Dobrin, and Sasikumar Punnekkat. Synchronization
378 quality of ieee 802.1as in large-scale industrial automation networks. In *23rd IEEE Real-Time*
379 *and Embedded Technology and Applications Symposium*, April 2017.
- 380 16 E. Heidinger, F. Geyer, S. Schneele, and M. Paulitsch. A performance study of audio video
381 bridging in aeronautic ethernet networks. In *IEEE International Symposium on Industrial*
382 *Embedded Systems (SIES'12)*, pages 67–75, June 2012. doi:10.1109/SIES.2012.6356571.
- 383 17 H. Kopetz and G. Bauer. The time-triggered architecture. *Proceedings of the IEEE*, 91(1):112–
384 126, Jan 2003. doi:10.1109/JPROC.2002.805821.
- 385 18 H. Kopetz and G. Grunsteidl. TTP - a time-triggered protocol for fault-tolerant real-time sys-
386 tems. In *FTCS-23 The Twenty-Third International Symposium on Fault-Tolerant Computing*,
387 pages 524–533, June 1993. doi:10.1109/FTCS.1993.627355.
- 388 19 P. Pedreiras and A. Luis. The flexible time-triggered (FTT) paradigm: an approach to
389 QoS management in distributed real-time systems. In *Proceedings International Parallel and*
390 *Distributed Processing Symposium*, April 2003. doi:10.1109/IPDPS.2003.1213243.
- 391 20 F. Pozo, G. Rodriguez-Navas, and H. Hansson. Schedule reparability: Enhancing time-triggered
392 network recovery upon link failures. In *RTCSA*, 2018. doi:10.1109/RTCSA.2018.00026.
- 393 21 M. L. Raagaard, P. Pop, M. Gutiérrez, and W. Steiner. Runtime reconfiguration of time-
394 sensitive networking (TSN) schedules for Fog Computing. In *2017 IEEE Fog World Congress*
395 *(FWC)*, pages 1–6, Oct 2017. doi:10.1109/FWC.2017.8368523.
- 396 22 W. Steiner. An Evaluation of SMT-Based Schedule Synthesis for Time-Triggered Multi-hop
397 Networks. In *IEEE Real-Time Systems Symposium*, Nov 2010. doi:10.1109/RTSS.2010.25.
- 398 23 W. Steiner, S. S. Craciunas, and R. S. Oliver. Traffic Planning for Time-Sensitive Com-
399 munication. *IEEE Communications Standards Magazine*, 2(2):42–47, JUNE 2018. doi:
400 10.1109/MCOMSTD.2018.1700055.
- 401 24 D. Tamas-Selicean, P. Pop, and W. Steiner. Synthesis of Communication Schedules for
402 TTEthernet-based Mixed-criticality Systems. In *IEEE/ACM/IFIP International Conference*
403 *on Hardware/Software Codesign and System Synthesis*, 2012. doi:10.1145/2380445.2380518.
- 404 25 S. Varadarajan and T. Chiueh. Automatic fault detection and recovery in real time switched
405 Ethernet networks. In *IEEE INFOCOM Conference on Computer Communications*, March
406 1999. doi:10.1109/INFCOM.1999.749264.
- 407 26 M. Wollschlaeger, T. Sauter, and J. Jasperneite. The future of industrial communication:
408 Automation networks in the era of the internet of things and industry 4.0. *IEEE Industrial*
409 *Electronics Magazine*, 11(1):17–27, March 2017. doi:10.1109/MIE.2017.2649104.