# Traffic Class Assignment for Mixed-Criticality Frames in TTEthernet

Voica Gavriluţ and Paul Pop
Technical University of Denmark
Kongens Lyngby, Denmark
voga@dtu.dk

## ABSTRACT

In this paper we are interested in mixed-criticality applications, which have functions with different timing requirements, i.e., hard real-time (HRT), soft real-time (SRT) and functions that are not time-critical (NC). The applications are implemented on distributed architectures that use the TTEthernet protocol for communication. TTEthernet supports three traffic classes: Time-Triggered (TT), where frames are transmitted based on static schedule tables; Rate Constrained (RC), for dynamic frames with a guaranteed bandwidth and bounded delays; and Best Effort (BE), for which no timing guarantees are provided. HRT messages have deadlines, whereas for SRT messages we capture the quality-of-service using "utility functions". Given the network topology, the set of application messages and their routing, we are interested to determine the traffic class of each message, such that all HRT messages are schedulable and the total utility for SRT messages is maximized. For the TT frames we decide their schedule tables, and for the RC frames we decide their bandwidth allocation. We propose a Tabu Search-based metaheuristic to solve this optimization problem. The proposed approach has been evaluated using several benchmarks, including two realistic test cases.

## 1. INTRODUCTION

*Mixed-criticality* systems have functions with different safety-criticality requirements, e.g., highly critical, mission critical, non-critical. For example, a network backbone in a modern vehicle has to integrate Advanced Driver Assistance Systems (ADAS) functions, which rely on high-bandwidth data from sensors, e.g., video cameras and Light Detection And Ranging (LIDAR), with power-train functions that have tight timing constraints but use small frame sizes, and infotainment services, which are not critical.

Due to the increase in complexity, and the need to reduce costs, such mixed-criticality applications are today implemented in *integrated architectures*, where functions of different criticality share the same distributed platform. Although there have been many safety-critical protocols proposed, only few of them can support the separation required by mixed-criticality messages [11].

There is an increasing interest in Ethernet-based solutions, since Ethernet has high-bandwidth and reduced costs. However, Ethernet is known to be unsuitable for real-time and safety-critical applications [4]. In this paper we are interested in the TTEthernet protocol [12], which extends the IEEE 802.3 Ethernet standard to provide deterministic time-critical services for mixed-criticality real-time applications. TTEthernet supports three traffic classes: Time-Triggered (TT), Rate Constrained (RC) and Best Effort (BE).



TT frames have the highest priority and are transmitted at predetermined points in time, based on synchronized distributed schedule tables. By synthesizing carefully the schedule tables, TT frames can have low end-to-end latency and low jitter. RC frames are compliant with ARINC 664p7 (Avionics Ethernet) [1] and have a guaranteed bandwidth. RC frames may be delayed by other RC frames or by TT frames. However, analysis methods exist that bound their worst-case end-to-end delays [15], providing timing guarantees. BE frames are compliant with IEEE 802.3 Ethernet and have the lowest priority, without any timing guarantees.

In this paper we are interested in mixed-criticality applications, which have functions with different timing requirements, i.e., hard real-time (HRT), soft real-time (SRT) and functions that are not time-critical (NC). In our model, HRT messages have hard deadlines, whereas for SRT messages we capture the quality-of-service (QoS) using soft deadlines and "utility functions", which model the relative importance of SRT messages and how the performance of the system degrades if the SRT soft deadlines are missed. Similar to the debate in real-time systems between time-triggered and event-triggered implementations [8, 10], there is no agreement on the appropriate traffic class for the mixed-criticality messages, which depends on the particularities of the applications. Therefore, in this paper, we are interested in the problem of *Traffic Class Assignment* for mixed-criticality messages in TTEthernet.

Given the network topology, the set of application messages and their routing, we are interested to determine the traffic class for each message, such that all HRT messages are schedulable and the total utility for SRT messages is maximized. For the TT frames we decide their schedule tables, and for the RC frames we decide their bandwidth allocation. We consider that the NC messages are implemented using the BE traffic class, and we do not consider the BE traffic class for HRT or SRT messages. However, the HRT and SRT messages can be implemented with the TT or the RC traffic class, as both traffic classes provide real-time guarantees. In case the NC messages require QoS guarantees, they can be treated as SRT messages. We propose a Tabu Search-based metaheuristic to solve this optimization problem.

The related work is presented in the next section. To the best of our knowledge, this is the first time such a problem has been addressed in the context of Real-Time Ethernet protocols.

### 1.1 Related Work

There have been several comparisons between time-triggered (TT) and event-triggered (ET) approaches, both at the task-level [8], and at message-level [10]. In [8], the authors decide which tasks should be TT and which ET, showing that the right choice depends on the particularities of the applications. Researchers [10] have also compared two networking approaches, i.e., Time-Division Multiplexing (TDM) with an ET approach in the context of Networks-on-

Chip. Their conclusion is that ET improves the bandwidth usage, whereas TDM is suited when the latencies have to be reduced.

In the context of ARINC 664p7, researchers have shown how to optimize the priorities of the RC traffic flows [5], and have proposed an extension to the Optimal Priority Assignment algorithm used for real-time tasks. The algorithm assigns higher priorities to traffic flows that have more stringent timing requirements. For TTEthernet, researchers have proposed approaches to synthesize the communication schedules [9, 3]. Their methods are able to handle up to 100,000 TT frames, but ignore RC frames.

A lot of work has been done for the analysis and optimization of communication protocols, including TTEthernet [15, 14]. Researchers, for example, have proposed approaches to decide the routes for the frames, their packing and fragmenting and the schedule tables for the TT frames [14]. However, all the related work has assumed that the traffic classes are decided, and have not addressed the problem of Traffic Class Assignment.

Finally, an interesting approach is the FTT-Ethernet protocol [7], which supports arbitrary traffic scheduling policies, and has mechanisms for dynamic QoS management.

## 2. ARCHITECTURE MODEL

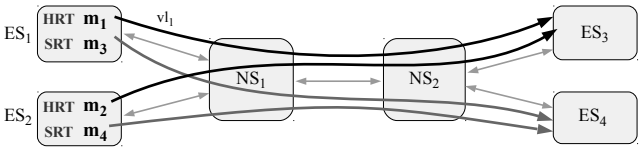
A TTEthernet network is composed of several clusters. We model a cluster as an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where the vertices (or nodes) $\mathcal{V} = \mathcal{ES} \cup \mathcal{NS}$ denote the set of all End Systems (ESes) and Network Switches (NSes), respectively, and the edges $\mathcal{E}$ are the physical links interconnecting the ESes and NSes. An ES is composed of a CPU, memory, I/Os, and a network interface card. Our traffic class assignment problem is applied at the cluster level. A cluster provides the clock synchronization needed for the schedule tables of the TT class. TT frames leaving a cluster have to be transformed into RC frames, which do not require clock synchronization for their transmission. Fig. 1a shows an example cluster with 4 ESes and 2 NSes.

A dataflow link (DL) $dl_i$ represents a directed connection between two nodes in $\mathcal{V}$. A sender task in a source ES is connected to a receiver task in a destination ES through a dataflow path (DP) $dp_i$, which is a sequence of interconnected DLs. The set of all DLs is denoted with $\mathcal{DL}$ an the set of all DPs is $\mathcal{DP}$. A frame in TTEthernet has one source, but it may have multiple receivers. The separation required by mixed-criticality frames is enforced through the concept of a *Virtual Link* (VL), which is a "logical unidirectional connection from one source end system to one or more destination end systems" as defined by ARINC 664p7 [1]. We consider that the routing of the VLs is given for all frames. We model a VL $vl_i$ as a set of DPs, one for each receiver, and all DPs sharing the same sender. $\mathcal{VL}$ is the set of all VLs. Fig. 1a shows 4 VLs. For example, $vl_1$ for the frame of message $m_1$ from $ES_1$ to $ES_3$, has the routing $ES_1$, $NS_1$, $NS_2$, $ES_3$ as depicted with a thick black arrow.

## 3. APPLICATION MODEL

In this paper, we consider mixed-criticality applications. Our application model captures the messages in the applications, and their timing requirements. The spatial separation required for safety-criticality is achieved via the VLs. We consider three types of timing criticality: Hard-Real Time (HRT) messages, which have strict deadlines, Soft Real-Time (SRT) messages, for which we are interested to maximize their "utility" and Non-Critical (NC) messages, which have no timing requirements. We denote the set of all messages in a cluster with $\mathcal{M} = \mathcal{M}^{HRT} \cup \mathcal{M}^{SRT} \cup \mathcal{M}^{NC}$, where the three sets correspond to the set of all HRT, SRT and NC messages.

Each message $m_i \in \mathcal{M}$ has a source $ES_i^{src}$ and one or more destinations $\mathcal{ES}_i^{dest}$, and a given size $m_i.size$. HRT and SRT messages are periodic, with a period $m_i.period$. Both HRT and SRT messages have a deadline, $m_i.deadline$. The HRT deadline is *hard*, i.e., if the deadline is missed, it may result in catastrophic consequences. The SRT deadline is *soft*, i.e., the performance of the system degrades if the deadline is missed.




(a) Example architecture model

| Msg. | Size | Period | Deadline / (Utility) |
|---|---|---|---|
| $m_1 \in \mathcal{M}^{HRT}$ | 50 B | 2 ms | 1 ms |
| $m_2 \in \mathcal{M}^{HRT}$ | 62.5 B | 3 ms | 2 ms |
| $m_3 \in \mathcal{M}^{SRT}$ | 500 B | 4 ms | 1.5 ms / (max. 6; 0 at 2.6 ms) |
| $m_4 \in \mathcal{M}^{SRT}$ | 750 B | 4 ms | 2.5 ms / (max. 6; 0 at 4.1 ms) |

(b) Example application model

Figure 1: Example system model

For SRT messages we use a positive non-increasing utility function, denoted with $m_i.utility(t)$, where $t$ is a time instant relative to start of the message transmission from its source $ES_i^{src}$. The utility starts from a positive value and sometimes after the soft deadline reaches a zero value. We consider that the system engineer specifies the utility functions of SRT messages to capture their relative importance and how the performance of the system degrades if their soft deadlines are missed, see [2] for a discussion on utility functions. If a SRT message $m_i$ arrives within its soft deadline $m_i.deadline$, then its utility value is maximal. However, if the deadline is missed, the utility value will decrease with time, as specified by the definition of the utility function $m_i.utility(t)$.

Fig. 1b shows an example[1] application model, with two HRT messages $m_1$ and $m_2$ and two SRT, $m_3$ and $m_4$. The VLs and their routing are presented in Fig. 1a. In this example, we use a simple linear utility function for both SRT messages $m_3$ and $m_4$, starting at a maximum utility of 6, linearly decreasing after the soft deadline, reaching a utility of zero at 2.6 ms and 4.1 ms, respectively. Our model does not explicitly capture NC messages, which we assume that will always be assigned to the BE traffic class.

## 4. TTETHERNET TRAFFIC CLASSES

The TT traffic class is defined in [12]. TT frames have the highest priority and are periodically sent and received at a-priori known points in time, which are stored in static schedule tables; let $\mathcal{S}$ denote the set of all such tables in the cluster. ESes and NSes rely on their sending schedule tables $\mathcal{S}_S$ to forward a TT frame on an outgoing port. At a receiving port, the arrival time of the TT frame is compared to the time specified in a receiving schedule table $\mathcal{S}_R$. If the TT frame arrives outside of a "receiving window" relative to $\mathcal{S}_R$, it will be discarded, as it is considered faulty.

The RC traffic class is defined in ARINC 664p7 [1], and provides guaranteed bandwidth at link level [1]. Let us remember from Sect. 2 that each frame $f_i$ is assigned a virtual link $vl_i$, which captures its routing and ensures its separation from the other frames. The sending ESes shape the RC frames using a "traffic regulator" function at the VL-level, ensuring that the periodic frames are separated with a minimum time interval, called Bandwidth Allocation

[1]Ethernet frames sizes are constrained between 64 B and 1518 B, but in this toy example we use smaller values for simplicity.

Gap, or BAG, specified for the VL, i.e., $vl_i.BAG$. VLs also have a parameter $vl_i.L_{max}$, which is the maximum frame size allowed to be transmitted by the VL $vl_i$.

A RC frame is queued in outgoing ports using FIFO queues, and will be sent only if there are no other RC frames ahead of itself, and if there are no TT frames transmitting at that time. TTEthernet uses three "traffic integration policies": *timely block*, which postpones the transmission of any lower priority frame (RC or BE) which can interfere with the sending of a TT frame; *shuffling*, which delays the high priority TT frames until lower priority frames complete their transmission; and *preemption*, which interrupts (preempts) the transmission of the lower priority frames to transmit a TT frame. In this paper we use, without loss of generality, the *timely block* integration policy.

We assume, as mentioned in Sect. 3 that the NC messages are assigned to the BE traffic class, therefore we do not discuss the BE traffic class in this paper.

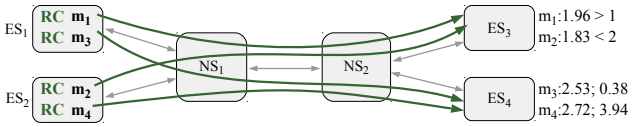
We are interested to decide the traffic class of each message $m_i \in \mathcal{M}$, which we capture with the function $\mathcal{TC}(m_i) : \mathcal{M} \rightarrow \{TT, RC\}$. Based on legacy constraints, or on the experience of the system engineer, some messages may already be assigned a traffic class. However, most of the messages will not have a pre-assigned traffic class. An interesting feature of TTEthernet is that a frame passing through a NS can change its traffic class [12]. For example, a frame can arrive as TT and leave as RC. Thus, the traffic class is assigned for each dataflow link $dl_i \in vl_i$ of $m_i$. However, in this paper we assume that a frame will not change its traffic class during its transmission, and we leave this aspect for future research.

Our optimization also decides the schedule tables $\mathcal{S}$ for TT frames and the *BAG*s and $L_{max}$es for the VLs of RC frames. Note that messages are packed into frames before they are transmitted. We do not consider the packing of multiple messages into a frame, since this has already been discussed in [14]. However, depending on how the $L_{max}$ and *BAG* of a VL $vl_i$ for a RC message $m_i$ are set, we may need to fragment a RC message into multiple frames. For example, if we set $vl_i.L_{max} = m_i.size/2$, i.e., half of the message size, then we need to split $m_i$ into two frames. We do such fragmenting for RC frames, depending on the VL parameters, but we do not consider the fragmenting of TT frames, leaving this for future work.

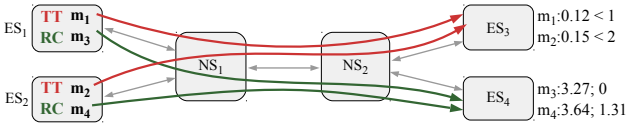
## 5. PROBLEM FORMULATION

As an input to our problem we have (i) the topology of the cluster $\mathcal{G}(\mathcal{V}, \mathcal{E})$ and (ii) the set of messages $\mathcal{M} = \mathcal{M}^{HRT} \cup \mathcal{M}^{SRT} \cup \mathcal{M}^{NC}$; for each message we know its parameters, as described in Sect. 3, including the VLs and their routing. We are interested to determine a network configuration $\Psi$ such that all HRT messages are schedulable and the total utility for SRT messages is maximized. Deciding on a network configuration $\Psi$ means determining, for each message $m_i \in \mathcal{M}$ with its VL $vl_i$, (1) the traffic class $\mathcal{TC}(m_i)$. In case $\mathcal{TC}(m_i)$ is TT, we also decide (2) the sending and receiving schedule tables $\mathcal{S}^{m_i}$ for $m_i$. If $\mathcal{TC}(m_i)$ is RC, we decide (3) the $BAG_i$ and $L_{max_i}$ for the VL $vl_i$.

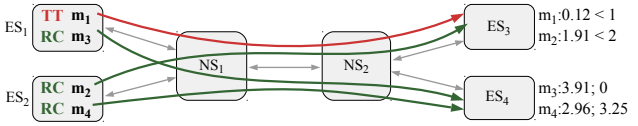
Let us consider the architecture and application from Fig. 1; as mentioned, the maximum utility of both SRT $m_3$ and $m_4$ is 6, so the total maximum utility achievable is 12. We are interested to determine the traffic class for each message. For a given traffic class assignment $\mathcal{TC}$, we determine for this small example, the optimal schedule tables $\mathcal{S}$, optimal *BAG* and $L_{max}$ values, i.e., such that the utility of SRT messages is maximized and the HRT messages are schedulable. For the RC frames, we use the worst-case end-to-end delay (WCD) analysis from Sect. 6.2 to determine their WCD $R_i$. For the TT frames, the WCD is derived directly from the schedule tables, as the time when the frame is received at its destination, relative to its sending time. A HRT frame is schedulable if its WCD is lower or equal to the deadline, and the utility of a SRT frame $m_i$ is given by $m_i.utility(WCD)$, as specified in Fig. 1b.



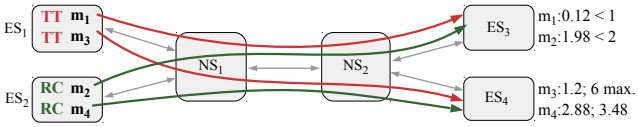

(a) All messages are RC; $m_1$ is **not** schedulable; total utility is only 4.32 out of 12.




(b) HRT messages are TT and SRT are RC. $m_1$ and $m_2$ are schedulable, but the total utility is only 1.31 out of 12.




(c) HRT $m_2$ is RC, but still schedulable; the total utility is 3.25, but $m_3$'s utility is zero.




(d) HRT $m_2$ is RC, SRT $m_3$ is TT. HRT are schedulable, and the total utility is increased to 9.48. $m_3$ has a maximum utility.

Figure 2: Motivational example

Multiple traffic classes are necessary to support mixed-criticality applications. For example, if we do not have the TT traffic class, and make all messages RC, we obtain the solution depicted in Fig. 2a. In all the examples in Fig. 2 we indicate next to the message source the traffic class used; we depict the respective VL with green for RC and red for TT. We write next to the destination of each HRT message its WCD and compare it to its deadline; next to each SRT message destination we have its WCD, followed by its utility. As we can see from Fig. 2a, if all frames are RC, then $m_1$ misses its deadline, i.e., $1.96 > 1$, and the total utility for SRT messages is only 4.32 out of the maximum of 12.

A possible solution would be to use TT for HRT messages and RC for SRT, as depicted in Fig. 2b. Such an approach is used implicitly, for example, in [14], which does not attempt to optimize the traffic classes. As expected, by using the TT traffic class for HRT messages $m_1$ and $m_2$, we can make them schedulable, since we can synthesize the TT schedules such that the HRT messages have a very low latency. However, as discussed in Sect. 4, TT frames have the highest priority, and when doing the traffic integration (timely block is considered in the paper), the RC frames may be delayed by the TT frames. Due to these delays, the utility of SRT message $m_3$ is zero and $m_4$ has a very small utility of 1.31 out of maximum 6. Recall that schedules are optimal with respect to HRT schedulability and SRT utility; in this case, delaying the TT frames will not help the RC frames because of the *timely block* integration policy, which does not allow a RC frame to start its transmission if it may delay a TT frame.

By using the RC traffic class for the HRT message $m_2$ instead of TT, we will get in Fig. 2c a larger WCD for $m_2$, of 1.87 ms instead

of 0.15 ms in Fig. 2b. However, since $m_2.deadline = 2$, $m_2$ is still schedulable. In addition, since $m_2$ is now not sent based on a schedule table, it will have less negative impact on the RC messages. Thus, $m_4$'s utility will be 3.25 out of 6. The utility of $m_3$ is still zero. If we further optimize the traffic class assignment, and we modify the solution in Fig. 2c to change the traffic class of SRT $m_3$ from RC to TT (as depicted in Fig. 2d), we are able, by carefully deciding on the schedule table for the TT $m_3$ frames, to reduce its WCD and thus increase $m_3$'s utility to the maximum of 6.

As this motivational example shows, only by optimizing the assignment of traffic classes to the mixed-criticality frames, we are able to obtain good quality solutions, which guarantee the schedulability of HRT messages while maximizing the utility for the SRT messages. Note that making all the frames TT regardless of their timing criticality could also be a solution. However, in practice, legacy RC messages have to be integrated in the system, and the system engineer may prefer that some frames are RC for flexibility. Updating schedules to accommodate new messages may trigger revalidation activities, which are costly. In addition, as the number of frames increases (systems may have tens of thousands of frames, even millions of frames [9]), the ESes and NSes will run out of memory for the required schedule tables. Although methods such as [9] can handle a large number of TT frames, they are not able to integrate the schedulability analysis of RC frames.

# 6. OPTIMIZATION STRATEGY

The problem presented in the previous section is NP-complete [16]. We propose a Tabu Search (TS)-based metaheuristic solution, called Traffic Class Assignment (TCA) to solve this optimization problem. TS-based approaches have been shown to produce good quality results for the problem of optimizing the configuration of TTEthernet systems [14]. TCA takes as input the network topology $\mathcal{G}(\mathcal{V}, \mathcal{E})$, the set of mixed-criticality messages $\mathcal{M} = \mathcal{M}^{HRT} \cup \mathcal{M}^{SRT}$, including the properties of each message, and the legacy traffic class assignment $\mathcal{TC}^0$, i.e., some messages may already be assigned a fixed traffic class. TCA produces as output an implementation $\Psi$, which contains the traffic class assignment $\mathcal{TC}$, the schedule tables $\mathcal{S}$ for TT frames and the *BAG*s and $L_{max}$es for the RC frames.

TS metaheuristics [6] search for that solution which maximizes a *cost function*. The cost function used to evaluate a visited solution is presented in Sect. 6.1. TS is based on a neighborhood search technique, where the current solution is modified using design transformations (also called *moves*) to generate neighboring solutions. The moves we propose, including an example of how TS works for our problem, are presented in Sect. 6.3. To avoid revisiting recently explored solutions, TS keeps a *tabu list*, which is a selective history of solutions that have already been visited. If the currently explored solution is better than the best known solution, it is saved as the "best-so-far" solution. A solution on the tabu list may be selected based on an "aspiration criteria", which allows the exploration of tabu solutions if they are better than the "best-so-far". To avoid getting trapped in a local optima, TS uses "diversification", i.e., forcing the algorithm to look in unexplored areas. The diversification method we use is similar to the "restart diversification" in [6], applied when no improvements are observed after a given number of $K$ iterations. Our TCA stops when a given time-limit has been reached.

TS can start from any initial solution, including a random solution. In our TCA implementation, we have selected the initial solution as follows. For the traffic class assignment $\mathcal{TC}$ we consider that HRT messages are TT and SRT messages are RC, under the constraints imposed by the given $\mathcal{TC}^0$. For each RC message $m_i$, the initial *BAG* and $L_{max}$ are chosen such that $vl_i.BAG$ will be the greatest allowed power of 2 less than $m_i.period$ (the "power of 2" constraints for BAGs come from the standard) and $vl_i.L_{max}$ will be $m_i.size$. Regarding the TT schedules, we are interested to derive an initial schedule such that there is space for RC frames, which are lower priority than TT, to transmit. This means that we avoid scheduling TT frames back-to-back in large blocks, which would introduce very large delays for RC frames. Thus, for each message $m_i \in \mathcal{M}^{TT}$, on each link $dl_j$ where the message is routed, a random value is picked within a certain *Scheduling Interval* $SI_{m_i,dl_j}$. The scheduling interval for each pair $(m_i, dl_j)$ is defined such that the resulted schedule is valid, i.e., a message should be sent only after it has arrived.

## 6.1 Cost Function

We evaluate each solution visited by the Tabu Search $\Psi$ using the following cost function:

$$Cost(\Psi) = wp_{HRT} \cdot \delta_{HRT} + \sum_{m_i \in \mathcal{M}^{SRT}} m_i.utility(WCD(m_i)) \quad (1)$$

where the first term represents a constraint which checks for the schedulability of HRT messages, and the second term is the total utility of SRT messages. $\delta_{HRT}$ captures the "degree of schedulability" of a solution and is defined as

$$\delta_{HRT} = \sum_{m_i \in \mathcal{M}^{HRT}} \min(0, m_i.deadline - WCD(m_i)) \quad (2)$$

where $WCD(m_i)$ is the worst-case end-to-end delay of the HRT message, calculated as presented in Sect. 6.2. Note that $\delta_{HRT}$ will be zero in case all HRT messages are schedulable, i.e., WCD is smaller than the deadline, otherwise it is a negative value. We multiply $\delta_{HRT}$ with a penalty value $wp_{HRT}$, which has been set as two times the value of the maximum total utility. If HRT messages are schedulable and thus $\delta_{HRT}$ is zero, the first term in Eq. 1 will not contribute to the cost function, and the search will attempt to maximize the total utility (the second term). However, if HRT messages are not schedulable, the penalty value will push TCA to search for schedulable HRT solutions.

## 6.2 WCD Analysis

The worst-case end-to-end delay (WCD) of a message is calculated differently depending on its traffic class. Recall that TT frames are sent based on schedule tables. For a TT message $m_i$ packed into a frame $f_i$ that is sent from a source ES to multiple destination ESes, the WCD is the maximum time in the receiving schedules of the destination ESes, i.e., the time the last frame is received at its destination, relative to its sending at the source ES.

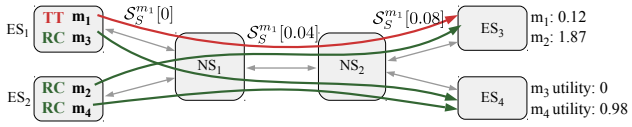
There have been several WCD analyses proposed for RC frames using ARINC 664p7 [5]; however, they do not take into account the impact of TT frames on the RC latencies. Recently, researchers have proposed a WCD analysis [15] for RC frames in TTEthernet, taking into account TT frames. In this paper, we have extended this analysis to determine the WCD of a RC frame. Compared to the work in [15], we have to account for the possible fragmenting of a RC frame, decided by the VL parameters. Thus, a RC message $m_i$ will be split into $k = \left\lceil \frac{m_i.size}{vl_i.L_{max}} \right\rceil$ frames $f_{i,1}, f_{i,2} \ldots f_{i,k}$, to fit into $vl_i.L_{max}$. Let $R_{m_i}$ be the WCD determined by the analysis in [15] for the last frame $f_{i,k}$ of $m_i$. Then, the WCD of $m_i$ is $WCD(m_i) = vl_i.BAG \cdot (k-1) + R_{m_i}$, where $vl_i.BAG$ is the period of the frames of message $m_i$. Note that the analysis of $R_{m_i}$ accounts for the multiple destinations of $m_i$, taking the largest WCD over the destinations.

## 6.3 Tabu Search Moves and Example

The neighborhood of the current solution is generated using three types of moves (1) Switch Traffic Class, *STC*, (2) Modify Schedule, *MS*, and (3) Modify RC VL Parameters, *MVL*. These moves are applied randomly according to probabilities chosen experimentally.

(1) The STC move is applied on any message, except for those which are covered by $\mathcal{TC}^0$. As the name implies, if the message has the traffic class TT, it will be switched to RC. The corresponding RC parameters *BAG* and $L_{max}$ are set as described in the initial TCA solution. If the message is RC, then it is made TT, and the schedules for this new TT frame are determined as in the initial TCA solution.

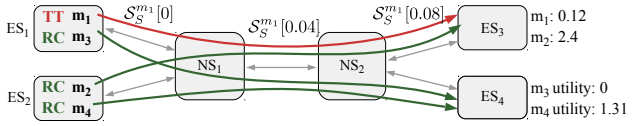
(2) The MS move is applied to TT messages. TT messages are transmitted as frames over several dataflow links. We first select the dataflow link $dl_j$ of the $vl_i$ of TT message $m_i$. The link $dl_j$ defines a subtree on which the move is applied, i.e., the subtree



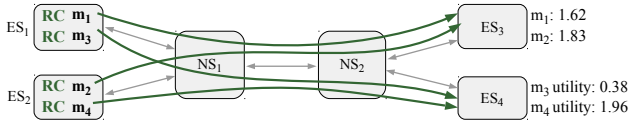

(a) The *current* solution is the example from Fig. 2c; Cost=0.98

| Message | $\mathcal{TC}$ | link | $\mathcal{S}_S/(BAG, L_{max})$ | iterations |
|---|---|---|---|---|
| $m_1$ | TT | $NS_1 - NS_2$ | [0.09] | 14 |
| $m_2$ | RC | — | (4, 125) | 5 |
| $m_3$ | TT | $ES_1 - NS_1$ | [1] | 0 |
| $m_3$ | TT | $NS_1 - NS_2$ | [1.3] | 7 |

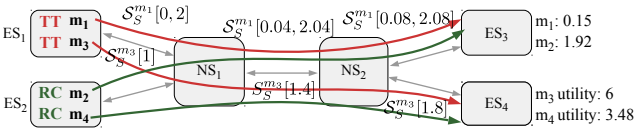
(b) Tabu list



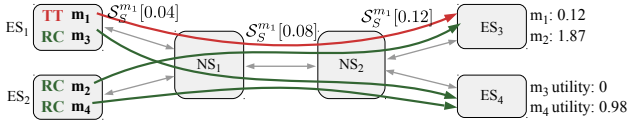

(c) Modify RC VL: *BAG* and $L_{max}$ are doubled; $Cost = -1.89$; tabu




(d) Switch Traffic Class of $m_1$ from TT to RC; $Cost = -2.62$; non-tabu




(e) Switch Traffic Class of $m_3$ from RC to TT; $Cost = 9.48$; non-tabu




(f) Modify Schedule of $m_1$ on $ES_1 - NS_1$ by postponing it with 0.04 ms; $Cost = 0.98$; non-tabu

Figure 3: Example TS neighborhood search

starting on $dl_j$ and connecting $dl_j$ with all the destination ESes $\mathcal{ES}_i^{dest}$. Then, we decide randomly if the frame on that subtree should be "postponed", i.e., delayed, or "advanced", i.e., scheduled at an earlier point in time. The resulted schedule is checked to be valid, as discussed, the time in the schedule when a frame is sent on a $dl_j$ should not be before it arrives.

(3) The MVL move is applied to the VLs of RC messages. Thus, for a $vl_i$, we randomly decide if we should double or halve the $vl_i.BAG$ value, taking care that the resulted value is valid, i.e., it is in the set of allowed "power of 2" values specified in the standard. Each change in $vl_i.BAG$ implies also a corresponding change in $vl_i.L_{max}$ such that the bandwidth of $vl_i$ is preserved, e.g., when BAG is doubled, then $L_{max}$ doubles as well.

Let us illustrate how TCA works. Let us consider the example from Fig. 1, and let us assume that the *current* solution is the solution depicted in Fig. 3a, which is also the *best-so-far* solution. Similar to the examples in Fig. 2, we denote the traffic class next to the message source, and the WCD and utility values next to the destination of the message. In addition, we also show, for the TT frames, the sending schedules $\mathcal{S}_S$ for each of the dataflow links where these are transmitted. For example, in Fig. 3a $m_1$ is sent on the last link, from $NS_2$ to $ES_3$ at time 0.08 ms. The $(BAG, L_{max})$ parameters for the RC VLs are $(vl_2.BAG, vl_2.L_{max}) = (2, 62.5)$, $(vl_3.BAG, vl_3.L_{max}) = (4, 500)$ and $(vl_4.BAG, vl_4.L_{max}) = (4, 750)$.

Recall that TCA uses a *tabu list* to avoid revisiting recently visited solutions. The tabu list for Fig. 3a is presented in the table in Fig. 3b, and stores on each row information about a particular solution visited in the past. Instead of storing the complete solution, we only store information related to the move that has generated the solution, i.e., the transformations performed. Thus, we store the message involved, the dataflow link (for TT frames) and the schedules $\mathcal{S}_S$ for the TT frames and *BAG* and $L_{max}$ for the RC frames. In the last column, we store the number of iterations this solution has been considered tabu. This value starts at the "tabu tenure", which we set to 25, and is decremented every iteration.

We first remove from the tabu list the entries whose "iterations" became 0, e.g., the line of $m_3$ in Fig. 3b. Next, we generate from the *current* solution the neighborhood solutions using the moves presented earlier. Since the neighborhood can be quite large, we restrict the neighborhood to a *Candidate List* of $n$ solutions. We use $n = 7$ in our experiments, but for the purpose of this example let us assume that $n = 4$. Thus, the *Candidate List* is obtained by randomly applying the moves on the *current* solution, obtaining the candidate solutions in Fig. 3c–f. For each candidate solution we write the following in its caption: the move that has generated it, the value of the *Cost* function, for which we considered in this example a penalty $wp_{HRT} = 8$, and if the move is tabu or not.

TCA will select that neighbor which improves the cost function and it is not tabu. For the neighbors in Fig. 3c–f, the neighbor in Fig. 3c is not replacing *current* because it is both, contained in the tabu list and its cost value is less than the value of *current*. The other candidates Fig. 3d–f are not in the tabu list (note that we have removed the third row in the tabu list in Fig. 3b). We will select that solution, which maximizes the cost (in our case, Fig. 3e), to replace *current* and the *best-so-far* (since that one is improved as well). The search is continued from the new *current* solution in a similar way.

# 7. EXPERIMENTAL EVALUATION

For the evaluation of our *Traffic Class Assignment* (TCA) optimization strategy we used three synthetic test cases, *tc1* to *tc3* and two real-life case studies, *SAE* and *orion*. Orion is the "Orion Crew Exploration Vehicle" case study from [14], and SAE is the "SAE

| Name | ESes | NSes | No. HRT msgs. | No. SRT msgs. | SFS | | TCA | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | %HRT schedulable | %SRT utility | Running time (h:min) | %HRT schedulable | %SRT utility |
| tc1 | 8 | 3 | 9 | 11 | 44.44% | 90.27% | 00:50 | 100% | 100% |
| tc2 | 8 | 3 | 11 | 23 | 54.54% | 85.07% | 2:30 | 100% | 99.63% |
| tc3 | 8 | 3 | 17 | 28 | 47.06% | 64.10% | 3:45 | 100% | 95.77% |
| SAE | 15 | 7 | 40 | 39 | 70.00% | 81.72% | 5:00 | 100% | 94.61% |
| orion | 31 | 15 | 99 | 87 | 45.45% | 78.80% | 12:30 | 94.94% | 98.68% |

Table 1: Experimental results

automotive communication benchmark" [13]. The number of ESes and NSes in the architecture, as well as number of HRT and SRT messages in the applications, are presented in columns 2–5, respectively, in Table 1. TCA was implemented in Java (JDK1.8) and all experiments were run on Intel Xeon E5-2665 machines at 2.4 GHz.

In the first set of experiments we were interested to determine the quality of our TCA algorithm. Thus, we have compared the results obtained with TCA to a straightforward solution, named SFS, which considers that both HRT and SRT messages are RC. SFS is also implemented as a Tabu Search, but we do not have the moves related to the traffic class assignment and TT schedule modifications, and instead only use moves that change the RC VL parameters. We have run both TCA and SFS on the test cases in Table 1, and we show for both of them two values, for each test case: the percentage of HRT messages found schedulable (%HRT), and the percentage of total utility of SRT messages, compared to the maximum utility achievable (%SRT utility). The time limit used for both TCA and SFS for each test case is given in column 8 in hours and minutes. This time limit has been set such that TCA has a good chance to find the near-optimal result, considering the size of the design space for the particular test case.

As we can see from the table, SFS, which uses only the RC traffic class, is unable to obtain schedulable solutions (for 4 out of 5 test cases only about half of the HRT messages are schedulable), and the utility of the SRT messages is lower compared to our TCA approach. By optimizing the assignment of traffic classes to mixed-criticality messages, we were able to obtain with our TCA schedulable solutions in most cases (100% schedulable HRT messages), or very close to full 100% schedulability. TCA is also able to significantly improve the utility compared to SFS, for example, from 64.10% to 95.77% utility in the case of *tc3*. TCA scales well with the size of the system (network and applications), being able to obtain good quality results also for the larger case studies.

We were also interested to compare TCA with the optimal solution. Due to the complexity of the problem, we were able to run an exhaustive search to get the optimal solution only for the smaller test case *tc1*. TCA has also been able to find the optimal result for this case, after a runtime of 50 minutes.

# 8. CONCLUSIONS

In this paper we have considered mixed-criticality applications, using hard real-time and soft real-time messages, implemented on a TTEthernet-based distributed system. We have used a hard deadline for the HRT messages and a utility function for SRT messages. We have proposed a Tabu Search-based metaheuristic, which we called Traffic Class Assignment (TCA), to determine the assignment of traffic classes (Rate Constrained, RC and Time-Triggered, TT) to the mixed-criticality HRT and SRT messages. TCA also optimizes the schedules for the TT frames and the RC virtual link parameters, Bandwidth Allocation Gap (*BAG*) and maximum frame size ($L_{max}$). As the experimental results show, our proposed TCA approach is able to determine, in a reasonable time, schedulable solutions (HRT messages meet their deadlines) which also improve the overall utility of the SRT messages. As future work, we are interested to extend our model to consider that the traffic class is assigned at the dataflow link-level, and not per message. We have considered the fragmenting of frames only for the RC class, since this was necessary when changing the VL parameters. We are also interest to extend our work to take into account the fragmenting of TT frames.

# 9. REFERENCES


[1] Aeronautical Radio, Inc. *ARINC 664P7: Aircraft Data Network, Part 7, Avionics Full-Duplex Switched Ethernet Network*. 2009.

[2] G. Buttazzo, G. Lipari, L. Abeni, and M. Caccamo. *Soft Real-Time Systems*. Springer, 2005.

[3] S. S. Craciunas and R. Serna Oliver. Combined task- and network-level scheduling for distributed time-triggered systems. *Real-Time Systems*, 52(2):161–200, 2016.

[4] J. D. Decotignie. Ethernet-based real-time and industrial communications. *Proceedings of the IEEE*, 93(6):1102–1117, 2005.

[5] T. Hamza, J.-L. Scharbarg, and C. Fraboul. Priority assignment on an avionics switched ethernet network (QoS AFDX). In *IEEE Workshop on Factory Communication Systems*, pages 1–8, 2014.

[6] G. Kendall and E. K. Burke. *Search methodologies: introductory tutorials in optimization and decision support techniques*. Springer, 2005.

[7] P. Pedreiras, P. Gai, L. Almeida, and G. C. Buttazzo. FTT-Ethernet: A flexible real-time communication protocol that supports dynamic QoS management on ethernet-based systems. *IEEE Transactions on Industrial Informatics*, 1(3):162–172, 2005.

[8] T. Pop, P. Pop, P. Eles, and Z. Peng. Analysis and optimisation of hierarchically scheduled multiprocessor embedded systems. *International Journal of Parallel Programming*, 36(1):37–67, 2008.

[9] F. Pozo, W. Steiner, G. Rodríguez-Navas, and H. Hansson. A decomposition approach for SMT-based schedule synthesis for time-triggered networks. In *IEEE Conference on Emerging Technologies & Factory Automation*, pages 1–8, 2015.

[10] W. Puffitsch, R. B. Sorensen, and M. Schoeberl. Time-division multiplexing vs network calculus. In *International Conference on Real Time and Networks Systems*, pages 289–296, 2015.

[11] J. Rushby. Bus architectures for safety-critical embedded systems. In *Embedded Software*, pages 306–323. Springer, 2001.

[12] SAE. *AS6802: Time-Triggered Ethernet*. SAE International, 2011.

[13] SAE International. *SAE Technical Report J2056/1*. 1993.

[14] D. Tamas-Selicean, P. Pop, and W. Steiner. Design optimization of TTEthernet-based distributed real-time systems. *Real-Time Systems*, 51(1):1–35, 2015.

[15] D. Tamas-Selicean, P. Pop, and W. Steiner. Timing Analysis of Rate Constrained Traffic for the TTEthernet Communication Protocol. In *IEEE International Symposium on Real-Time Distributed Computing*, pages 119–126, 2015.

[16] J. D. Ullman. NP-complete scheduling problems. *J. Comput. Syst. Sci.*, 10(3):384–393, 1975.