Routing Optimization of AVB Streams in TSN Networks

Sune Mølgaard Laursen, Paul Pop
Technical University of Denmark
Kongens Lyngby, Denmark
{smla, paupo}@dtu.dk

Wilfried Steiner
TTTech Computertechnik AG
Vienna, Austria
wilfried.steiner@tttech.com

ABSTRACT

In this paper we are interested in safety-critical real-time applications implemented on distributed architectures using the Time-Sensitive Networking (TSN) standard. The ongoing standardization of TSN is an IEEE effort to bring deterministic real-time capabilities into the *IEEE 802.1* Ethernet standard supporting safety-critical systems and guaranteed Quality-of-Service. TSN will support Time-Triggered (TT) communication based on schedule tables, Audio-Video-Bridging (AVB) streams with bounded end-to-end latency as well as Best-Effort messages. We consider that we know the topology of the network as well as the routes and schedules of the TT streams. We are interested to determine the routing of the AVB streams such that all frames are schedulable and their worst-case end-to-end delay is minimized. We have proposed a search-space reduction technique and a Greedy Randomized Adaptive Search Procedure (GRASP)-based heuristic for this routing optimization problem. The proposed approaches has been evaluated using several test cases.

1. INTRODUCTION

Only a few communication protocols are suitable for supporting distributed safety-critical real-time applications which have strict timing and dependability requirements [13]. In this paper we are interested in the protocol colloquially known as *Time-Sensitive Networking* (TSN) which is compatible with the well-known *IEEE 802.1*[1] Ethernet architecture. There is a strong industrial interest in Ethernet because it (i) meets the increasing bandwidth demands, (ii) supports switched multi-hop network topologies and (iii) reduces the need for proprietary equipment leading to more cost effective and maintainable solutions. However, Ethernet is not suitable for real-time and safety critical applications due to the lack of real-time capabilities [6].

Therefore, several extensions to Ethernet have been proposed, such as, TTEthernet [14] and ARINC 664 Specification Part 7 [3]. The *IEEE 802.1 Higher Layer LAN Protocols Working Group* has also moved in this direction by standardizing a set of enhancements making up TSN, introducing new traffic shapers enabling *IEEE 802.1* to support safety-critical and real-time applications.

First, *IEEE 802.1Q-2005* introduced support for prioritizing the Best-Effort (BE) traffic such that prioritized traffic could have a higher Quality-of-Service (QoS). Following this, the IEEE *Audio-Video-Bridging* (AVB) *Task Group* was formed to develop another set of enhancements *IEEE 802.1BA* known as AVB. This standard introduces two new shaped AVB traffic-classes, with bounded *Worst-Case end-to-end Delays* (WCD). In 2012 the AVB Task Group was renamed to the *TSN Task Group* to reflect the shifted focus onto further extending the protocol towards safety-critical and time-sensitive transmissions by introducing the Time-Triggered (TT) traffic-type.

At the time of writing, the work on TSN is still ongoing and the complete set of substandards making up TSN has not yet been finalized. In this paper we consider TSN as supporting AVB with the currently finished sub-standards *IEEE 802.1Qbu Frame Preemption* and *IEEE 802.1Qbv Enhancements for Scheduled Traffic*. As their titles imply *IEEE 802.1Qbv* adds the TT traffic shaper and the closely connected *IEEE 802.1Qbu* the ability of having higher priority frames preempt lower priority frames.

The choice of traffic type, BE, AVB or TT, for each message, depends on the application, and we assume that the systems engineer has configured each message to a suitable traffic type. For example, TT can be used for periodic hard real-time applications, such as jitter-sensitive control applications in need of very tight bounds on their WCDs. AVB also provides guaranteed WCD bounds needed for hard real-time applications but is exposed to interference from TT streams, the other AVB streams as well as BE traffic resulting in substantially larger WCD bounds and jitter, depending on the scenario, making it more suitable for applications with less stringent timing requirements. BE is used for sporadic traffic not requiring timing guarantees.

We consider real-time applications implemented using TSN distributed architectures. As an input to our problem we have (i) the network topology, (ii) the set of TT streams and their routing and schedules called Gate Control Lists (GCL), and (iii) the set of AVB streams. We are interested to determine the routing of the AVB streams, such that all streams are schedulable and their WCDs and utilization minimized. We have proposed an extension to the timing analysis in AVB, which can take into account the presence of TT traffic and preemption. For routing, we have proposed a Greedy Randomized Adaptive Search Procedure (GRASP)-based heuristic [9] on a search space which has been reduced using a K shortest paths-based algorithm [8].

1.1 Related Work

Routing optimization is a well-studied subject where Wang et al. [17] and Grammatikakis et al. [10] provide excellent overviews of the different centralized and distributed routing algorithms.

AVB streams are typically established at runtime using the *Stream Reservation Protocol (SRP)* where either the *Rapid Spanning Tree Protocol* (RSTP) or *Shortest Path Bridging* (SPB) are used to determine the routing. But as will be shown in Sect. 5.1, using the shortest route is not necessarily the best solution to ensure schedulability. The future enhancements around TSN will support more sophisticated runtime routing algorithms, and the possibility to also determine the routes offline.

In the context of offline determined routing for messages in safety-critical systems, as considered in this paper, Tamas-Selicean et

[1]Due to the lack of space, we will not provide references to all standards, but these can be found based on their name.

al. [15] have used a Tabu Search-based metaheuristic to, among other things, optimize the routings of the rate-constrained traffic to minimize the WCDs in TTEthernet systems. Sheikh et al. [2] proposed an approach to find the optimal route in AFDX networks using Mixed Integer Linear Programming. However, no work has been done so far for optimizing the routes in TSN for the AVB streams.

For AVB systems, the *AVB Latency Math* equation from *IEEE 802.1BA* specifies a WCD equation to be used as a decentralized admission test by the bridges when AVB streams are established using the SRP. This equation is considered an approximation as it, depending on the scenario, can give both unsafe and overly pessimistic WCD bounds. Lately, global timing analysis, has gained some attention where Diemer et al. [7] used Compositional Performance Analysis (CPA) to derive the WCD of the AVB streams. Bordoloi et al. [4] proposed improvements to this approach and supplied proofs of the correctness. De Azua et al. [5] proposed a theoretical network calculus model to derive the WCD. None of these analyses considers the effect of TT traffic on the latency of the AVB streams. Several groups have measured the impact of TT-traffic on the AVB streams using simulation [12, 1], which does not provide any guarantees.

2. ARCHITECTURE MODEL

The Ethernet standard defines a switched multi-hop network being composed of *End Systems* (ES) and *Bridges* (BR) connected by physical links. Each ES contains memory, processing elements and network interface cards. Each BR contains multiple ports and is responsible to bridge ingress frames to egress ports depending on the destinations of the frame. We refer to the subset of reachable TSN ESs and BRs using only TSN aware devices as a *TSN Domain*.

The topology of a TSN Domain is modelled as a directed graph $G(\mathbf{V}, \mathbf{E})$ where the set of vertices $\mathbf{V}$ is the union of all the End Systems $\mathbf{ES}$ and the Bridges $\mathbf{B}$, $\mathbf{V} = \mathbf{ES} \cup \mathbf{B}$. The edges $\mathbf{E}$ represent the physical connections and we denote the data link dl from the vertex $V_j \in \mathbf{V}$ to $V_k \in \mathbf{V}$ as $dl_u = [V_j, V_k]$. The physical transmission rate of this link is denoted $dl_u.rate$ and is typically 100 Mbps or 1 Gbps. An example model is presented in Fig. 1, having 7 ESs and 4 BRs where the black double arrows represent the physical *full-duplex* links allowing traffic in both directions simultaneously.

A datapath dp_j is an ordered sequence of links connecting one sender $ES_j \in \mathbf{ES}$ to one receiver $ES_k \in \mathbf{ES}$. In Fig. 1 we have $dp_1 = [[ES_5, BR_3], [BR_3, BR_2], [BR_2, BR_4], [BR_4, ES_3]]$ and $dp_2 = [[ES_5, BR_3], [BR_3, BR_2], [BR_2, BR_4], [BR_4, ES_4]]$. We use the term route r to denote the set of datapaths making up the route which may have multiple destinations. So, the r representing a multicast to n destinations is defined as the set of n datapaths, $r = \{dp_1, ..., dp_n\}$. For example, in Fig. 1 we have that $r_1 = \{dp_1, dp_2\}$ connecting ES_5 to ES_3 and ES_4.

3. APPLICATION MODEL

Messages transmitted among ESs are wrapped in an Ethernet *frame* adding the necessary headers used by the network devices to route the frame to its destination. The issue of packing and fragmenting is orthogonal to our problem, and has been discussed in earlier works, for example in the context of TTEthernet [15]. Both the AVB- and TT-traffic classes are used to carry periodic frame instances denoted as *streams*. The set of all AVB streams is defined as $\mathbf{S}^{\mathrm{AVB}}$ and the set of all TT streams $\mathbf{S}^{\mathrm{TT}}$. The BE traffic-class is not explicitly modelled as it is considered outside the scope of this paper.

Each TT stream $s_i \in \mathbf{S}^{\mathrm{TT}}$ is defined by the following attributes: $s_i.r$, which is the route of s_i, $s_i.size$, which is the maximum size of a frame in s_i, $s_i.period$, which is the period of the stream. The model used to capture the TT GCLs is presented in Sect. 4.1.

For each AVB stream $s_i \in \mathbf{S}^{\mathrm{AVB}}$ we know its source $s_i.src \in \mathbf{ES}$ and destinations $s_i.\mathbf{dest} \subseteq \mathbf{ES}$, as well as the maximum size of a frame in the stream $s_i.size$ and its class $s_i.class$, A or B. The TSN group is working towards supporting fully customizable AVB classes, allowing the definition of more traffic classes. Hence, our model is general, and considers an arbitrary amount of traffic classes. To prevent starvation of lower priority traffic, each class x has an allocation ratio A_x denoting the fraction of bandwidth it may use. This value includes the TT traffic, so a value of $A_A = 0.75$ means that 75% of the bandwidth can be used for TT- and AVB Class A-traffic. We also know the period $s_i.period$ and deadline $s_i.deadline$ of each AVB stream s_i, which may depend on the AVB class characteristics.

The routing of an AVB-stream s_i, to be determined by our routing optimization, is captured by the function $\mathcal{R}(s_i)$ returning a route r.

4. TSN PROTOCOL

We present in this section how the traffic classes in TSN are being transmitted. The presentation is not intended to be exhaustive; instead, it focuses on the concepts needed in this paper. For details, the reader is directed to *IEEE 802.1Q-2012* for the BE and AVB classes, *IEEE 802.1Qbv* for TT and *IEEE 802.1Qbu* for details on preemption.

Each egress port in a BR has eight queues associated with it, and each queue has a priority, from seven (highest) to zero (lowest), see Fig. 2 for an illustration. Every frame contains a priority field determining which queue to be placed in. Typically, the highest prioritized queue is used for the TT traffic, the following two queues are used for different classes of AVB streams and finally the five remaining queues are used for prioritized BE traffic. The AVB queues make use of a Transmission Selection Algorithm (TSA) in the form of the *Credit-Based Shaper* (CBS), explained in Sect. 4.2. The transmission of TT frames is explained in the next section, but BE frames will not be further covered.

The *Transmission Selection* (see Fig. 2) initiates transmission from the highest priority queue that is *available* and has frames to transmit. The availability of each queue is controlled by (i) its transmission *Gate*, which can either be in an *open* or *closed* state and (ii) a CBS if present.

When a transmitting frame is preempted by a frame of higher priority, the transmitting frame finishes its current *fragment*, before initiating transmission of the higher priority frame. To compensate for this effect, such that the higher priority frame always can initiate transmission immediately, the lower priority queue is typically closed for the worst-case duration it takes to finish transmitting a

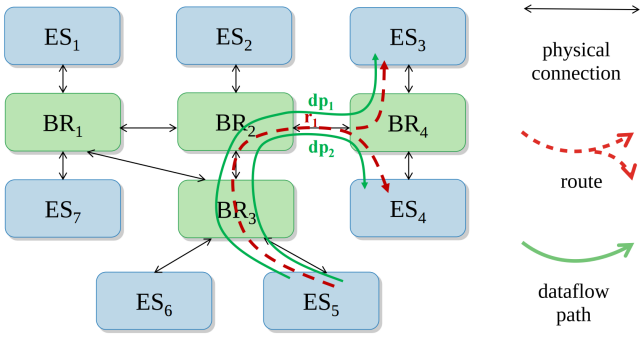


Figure 1: Example TSN topology model

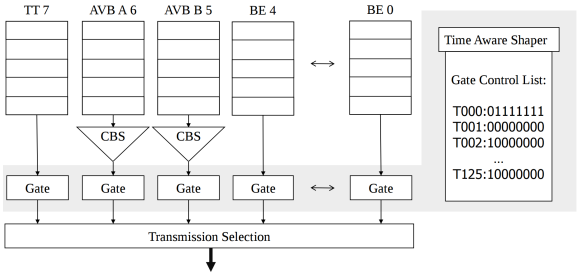


Figure 2: Typical TSN bridge configuration

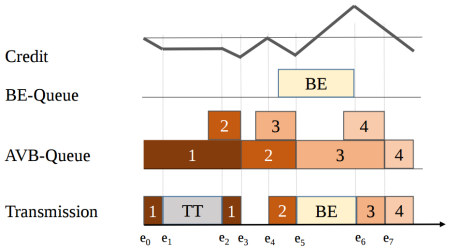


Figure 3: Example AVB transmission

fragment before opening the higher priority queue. The fragment size is typically 64 B. When a frame finishes its transmission, a special sequence needs to be transmitted separating it from the next frame. This sequence is denoted *Inter frame Gap* (IFG) and is typically 12 B and also needs to be accounted for when a preempted frame resumes its transmission. In this paper we assume that only TT traffic can preempt AVB traffic.

4.1 TT Traffic

The Gates are opened and closed by a *Time Aware Shaper* (TAS), according to a port-specific *Gate-Control List* (GCL) dictating the state of the gates at defined times relative to the start of the GCL. For example, in the GCL in Fig. 2, T000:0111111 means that at time "T000" relative to the start of the GCL, the TT queue, identified by using its queue priority as index, is closed (0) and all the rest are open (1). The start of these GCLs are synchronized across the bridges using the *time synchronization* defined in *IEEE 802.1AS*. Each GCL is repeated with a period typically set to be a multiple of the *Least Common Multiple* of all the periods used in the system.

As suggested in *IEEE 802.1Qbv*, to completely avoid interference from other traffic-classes, we assume the GCLs are constructed such that the TT queue is the sole available queue when open and all the remaining queues have been closed in advance. Thus every time the TT transmission gate opens, the TT frames can be transmitted immediately and by the synchronization of the GCLs on the entire path from sender to receiver, a TT frame can be transmitted without having to be subjected to any queueing delays. This makes the TT traffic class suitable for jitter- and latency-sensitive applications. In our model, we capture the GCL of a TT stream s_i in terms of an offset, period and duration. For example, in Table 1 the GCL for s_4 is $\langle offset,\ period,\ duration\rangle = \langle 0\ \mu s, 62.5\ \mu s, 10.4\ \mu s\rangle$, where the duration denotes the amount of time the TT queue has exclusive access to transmit a TT frame. Note that in the TSN implementation every bridge may have its own offset and duration value. We consider that, as suggested by *IEEE 802.1Qbv*, the GCLs are set such that the gate in BR_{k+1} opens immediately after the TT frame from the previous BR_k is received, such that there are no queuing delays for TT frames. Therefore, we can deduce the offsets for each BR in the TSN domain based on the $\langle offset,\ period,\ duration\rangle$ specified for the source ES of a TT stream s_i.

4.2 AVB Traffic

An AVB frame is transmitted when (i) the gate of its queue is open, (ii) there is no other higher priority frame being transmitted and (iii) if its CBS allows it. The CBS standardized in *IEEE 802.1Qat* in conjunction with the amendments in *IEEE 802.1Qbv* makes the queue available for transmission whenever the amount of *credits* is positive or zero. The purpose of the CBS is to shape the transmission of AVB frames in order to prevent bursts and starvation of the lower priority queues. Credits are initially zero, they are decreased with a *send slope* while transmitting and frozen while the gate is closed. Transmission is only initiated when credit is positive. The credit is increased with an *idle slope* when frames are waiting, but they are not being transmitted. If the queue is emptied while the credit is positive, the credit is set to zero. The *idle* and *send-slopes* are configuration parameters, which are set depending on the AVB class and experience of the systems engineer.

An example of how CBS works is illustrated in Fig. 3 where we have a TT frame, one AVB queue that has to transmit frames, 1 to 4, as well as a BE queue. The figure shows a timeline for the transmission on the bus, where a rectangle is a part of a frame, with the width representing the transmission time including the IFG. For TT, the rectangle also includes the effect of having to close the AVB queues before a scheduled transmission. The AVB and BE queues show on the y-axis the number of queued frames, and on the x-axis the waiting time in the queue. The value of the credit over time is presented on the top of the figure. Let us explain the transmission of the the AVB frames in Fig. 3 using the events (e_0) to (e_7) depicted on the bottom timeline:

(e_0) AVB Frame 1 starts to transmit and the credits are decreased according to the send slope. (e_1) Let us assume that a TT frame is scheduled as depicted in the bottom timeline of Fig. 3. The AVB queue is closed to make room for the TT transmission. AVB Frame 2 arrives and is enqueued while the credits are frozen. (e_2) The TT transmission finishes and the AVB-Queue opens and resumes the transmission of AVB Frame 1. During this transmission, the credits are decreased again. (e_3) Transmission of AVB Frame 1 finishes, but as the credit at this point is negative, AVB Frame 2 is not transmitted. Meanwhile, AVB Frame 3 is enqueued and the credits are accumulating according to the idle slope. (e_4) Credits have increased to zero, hence AVB Frame 2 is transmitting. During this transmission, the credits are decreasing according to the send slope. During this time, a BE frame is enqueued. (e_5) The transmission of AVB Frame 2 finishes, and since the credit is negative, the lower prioritized BE frame is selected for transmission. AVB Frame 4 is enqueued and credits are accumulating. (e_6) The transmission of the BE Frame finishes and AVB Frame 3 is selected for transmission. (e_7) The transmission of AVB Frame 3 finishes and the excess credits accumulated transmitting the BE frame are used to immediately initiate transmission of AVB Frame 4.

5. PROBLEM FORMULATION

The problem addressed in this paper is as follows: As an input to our problem we have (i) the Network Topology $G(\mathbf{E}, \mathbf{V})$, (ii) the set of AVB streams $\mathbf{S}^{AVB}$ with their requirements and (iii) the set TT streams $\mathbf{S}^{TT}$ with their routes and schedules. We are interested to determine a routing r_i for each of the AVB streams $s_i \in \mathbf{S}^{AVB}$ such that they are schedulable ($s_i.deadline$ not exceeded) and the worst-case end-to-end delays and the network utilization is minimized.

5.1 Motivational Example

Let us consider the topology from Fig. 1 with the streams sum-

AVB Class A streams $\mathbf{S}^{\text{AVB}}$

Stream	Endpoints	size	period	deadline	U_{100}
s_1	$ES_1 \rightarrow ES_4$	400 B	62.5 μs	2 ms	51%
s_2	$ES_5 \rightarrow ES_3, ES_4$	350 B	125 μs	2 ms	22%
s_3	$ES_7 \rightarrow ES_2$	480 B	125 μs	2 ms	31%

TT streams $\mathbf{S}^{\text{TT}}$

Stream	route	GCL (μs)
s_4	$[ES_6, BR_3], [BR_3, BR_2], [BR_2, ES_2]$	$\langle 0, 62.5, 10.4 \rangle$

Table 1: The $\mathbf{S}$ used for the motivational example.

marized in Table 1. Let us assume that the CBS slopes are set such that TT and AVB traffic use at most 75% of any link's bandwidth leaving 25% for the BE traffic so $A_A = 0.75$ (in this example we only consider AVB Class A traffic). The measure U_{100} in the last column for the AVB streams in Table 1 denotes the corresponding AVB utilization of the respective stream on a 100 Mbps network and is calculated as $U_{100} = \frac{s_i.size}{s_i.period \cdot 100\,\text{Mbps}}$. This measure is only used as a part of this example.

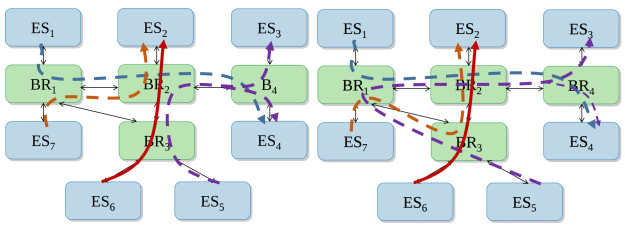
We show in Sect. 6.3 how to determine the WCDs. However, for the purpose of simplicity of this example, we will not use here the WCDs to define the schedulability but instead use the utilization measure, that an AVB stream s_i of class x is schedulable, if it is routed such that no data link in the route is utilized more than A_x by AVB and TT traffic. Let us assume that the TT stream s_4 corresponds to a bandwidth utilization on the path $[[ES_6, BR_3], [BR_3, BR_2], [BR_2, ES_2]]$ that leaves 50% bandwidth for the AVB Class A traffic.

In this paper we are interested to optimize the routing of AVB streams such that they are schedulable. This problem is non-trivial, since shortest-path routing may lead to non-schedulable solutions. For example: If we use the shortest path for each route, we get the solution in Fig. 4a where the streams s_2 and s_3 are not schedulable, i.e., the sum of utilization contributions from each stream exceeds 75% on the datalinks $[BR_1, BR_2]$ and $[BR_3, BR_2]$. An optimized solution is illustrated at Fig. 4b where, compared to the infeasible solution in Fig. 4a, neither s_2 nor s_3 use the shortest path. However, in this solution both s_2 and s_3 are schedulable. Note that full-duplex ensures no interference from s_3's routing through $[BR_1, BR_3]$ and s_2's routing through $[BR_3, BR_1]$.

As we can see from this example, only by optimizing the routing of AVB streams we are able to find schedulable solutions.

6. ROUTING OPTIMIZATION

The problem presented in Sect. 5 is NP-hard. Exhaustively enumerating every path between two vertices has been proven NP-hard [16] and as this would need to be done for every destination vertex of every stream $s \in \mathbf{S}^{\text{AVB}}$ to exhaustively evaluate every routing combination, this approach will lead to an intractable amount



(a) Shortest routes (b) Optimized routes

Figure 4: Two routing solutions of the streams in Table 1 on the network from Fig. 1. The colored dashed lines represent AVB streams and the full red line is the TT stream

of combinations in need of evaluation. Our proposed *Routing Optimization* (RO) approach, has two steps: (1) In the first step, we reduce the search space using a K shortest paths heuristic as described in Sect. 6.1. (2) In the second step, we employ a GRASP-based metaheuristic, presented in Sect. 6.4 to search the *reduced search space* where each candidate solution is evaluated using the cost function presented in Sect. 6.2. The cost function uses the analysis from Sect. 6.3 to determine the WCD of an AVB frame.

6.1 Search Space Reduction

We reduce the search space to only consider the K shortest path of every datapath dp_j of every stream $s_i \in \mathbf{S}^{\text{AVB}}$ using the K *shortest paths* algorithm [8]. K shortest paths returns K unique routes of increasing length, starting from the shortest route. For example, for the applications in Table 1 and $K = 1$ only the shortest paths as depicted in Fig. 4a are considered, which leads to an infeasible solution. But with $K = 2$ longer paths are considered as well, and the feasible solution depicted in Fig. 4b will be contained in the search-space. However, the idea of the heuristic in this paper is that good quality solutions can be found by combining routes which, although are not the shortest routes, they are not excessively long. Longer routes will generally increase the WCD of frames, and will lead to more overlap in general, which may increase the link utilization. Note that limiting each route $\mathcal{R}(s_i)$ to the K shortest is not guaranteed to find the optimal solution, but in practice, as the experimental results show in Sect. 7, will lead to schedulable solutions.

The K *shortest path* algorithm has a time complexity of $O(|\mathbf{E}| + |\mathbf{V}| \cdot log|\mathbf{V}| + K)$, where $|\mathbf{V}|$ is the number of nodes (ESs and BRs) in the network and $|E|$ is the number of physical links, therefore it scales well with the input.

6.2 Cost function

We define the cost of a solution as being the sum of the the objectives O_1, O_2 and O_3 multiplied with their respective weights W_1, W_2 and W_3:

$$cost(\mathcal{R}) = O_1(\mathcal{R}) \cdot W_1 + O_2(\mathcal{R}) \cdot W_2 + O_3(\mathcal{R}) \cdot W_3 \qquad (1)$$

The first objective O_1 counts the number of streams that exceed their deadlines, which is 0 if the solution is schedulable. Formally,

$$O_1 = \sum_{s_i \in \mathbf{S}^{\text{AVB}}} |\mathcal{T}_{wc}(\mathcal{R}(s_i)) > s_i.deadline|$$

where $\mathcal{T}_{wc}$ is the WCD of an AVB stream calculated as discussed in Sect. 6.3. The first term of Eq. 1 is a schedulability constraint, thus the associated weight W_1 is set to a very large value to direct the search away from unschedulable solutions. If O_1 is zero, the solution is schedulable hence the term is ignored. If O_1 is non-zero, the solution is not schedulable, and the cost function is heavily penalized with the weight W_1.

Once a solution is schedulable, the second objective O_2 attempts to minimize the WCDs by summing the fraction of each streams' WCD and its deadline. Formally,

$$O_2 = \sum_{s_i \in \mathbf{S}^{\text{AVB}}} \frac{\mathcal{T}_{wc}(\mathcal{R}(s_i))}{s_i.deadline}$$

It is envisioned that a practical implementation will use individual weights for every stream so that they can be prioritized, but for the sake of simplicity in this paper we use a single value weight W_2.

The third objective O_3 is used to improve the utilization characteristics of the network. O_3 counts the number of unique data-links dl_u used by the datapaths dp_j of the routing $\mathcal{R}(s_i)$ of all the AVB

streams $s_i \in \mathbf{S}^{\text{AVB}}$. This way, shorter-routes and multicasts with late branching points are preferred.

$$O_3 = \sum_{s_i \in \mathbf{S}^{\text{AVB}}} |\{\forall dl_u \in \{\forall dp_j \in \mathcal{R}(s_i)\}\}|$$

6.3 AVB frame WCD analysis

We have presented in Sect. 1.1 the related work on AVB WCD analysis, and concluded that none of the proposed methods take into account the impact of the TT traffic. In this section, we extend the *AVB Latency Math* from *IEEE 802.1BA* to consider TT streams. Although the AVB Latency Math has the drawbacks mentioned in Sect. 1.1, we have decided to use it in our cost function, since it is computationally efficient. However, our cost function can use any AVB analysis from related work, once they are extended to account for TT traffic.

We define the WCD $\mathcal{T}_{wc}$ of a stream s_i as the sum of all data-link latencies on the most delaying path from the source to any of its destinations given the route $\mathcal{R}(s_i)$. Formally,

$$\mathcal{T}_{wc}(s_i) = \max_{dp_j \in \mathcal{R}(s_i)} \sum_{dl_u \in dp_j} t_{wc}(dl_u, s_i)$$

Let us recall that TT streams are synchronized within a TSN domain. However, AVB streams are not synchronized, hence we have to find the worst-case scenario for a particular AVB stream under analysis, i.e., the situation that delays an AVB stream the most. We determine the worst-case scenario for an AVB stream s_i on each dataflow link dl_u where s_i is passing through. Such a link dl_u, may be shared by s_i with other TT and AVB streams. The delays due to other AVB streams are already accounted for in the definition of $t_{wc}(dl_u, s_i)$ in *IEEE 802.1BA* which also accounts for the case when the deadlines are larger than the stream periods. Here, we focus on the TT streams.

The TT streams are scheduled on dl_u based on their schedule tables (GCLs). In Fig. 5 we show with gray boxes an example set of the TT frames scheduled on a link dl_u. A grey rectangle shows the time period when the queue of the AVB frame under analysis is closed (we assume that the gate is closed because of the TT transmission). To determine the worst-case impact of this GCL on the frame f_i of our AVB stream s_i on dl_u, we consider that f_i has arrived just when a TT frame has started transmission. We try each start of a TT-frame to check which position is introducing the most delays, i.e., time when f_i cannot transmit because other TT frames are transmitting. Thus, we add the duration of each TT frame that preempts f_i, for the worst-case scenario. We assume the overheads due to preemption, mentioned in Sect. 4, are included in the duration of the TT transmission.

Algorithm 1 TT-interference

```
1:  function CALCULATETTINTERFERENCE(dl_u, t_wc)
2:      I_max ← 0
3:      for every Gate Close Event gce_j on dl_u do
4:          I_current ← 0, rem ← t_wc, gce_ptr ← gce_j
5:          while rem ≥ 0 do
6:              I_current ← I_current + gce_ptr.duration
7:              rem ← rem − (gce_ptr.next.start − gce_ptr.end)
8:              gce_ptr ← gce_ptr.next
9:          end while
10:         if I_current ≥ I_max then I_max ← I_current
11:         end if
12:     end for
13:     return I_max
14: end function
```

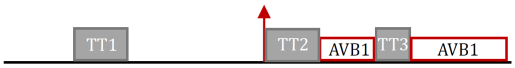


Figure 5: Worst-case interference exercised towards an AVB frame (red border) due to TT-traffic (grey)

We use Alg. 1 to determine the worst-case interference I_{max} on the t_{wc} of AVB frame f_i on a link dl_u. We call Alg. 1 immediately after the calculation of t_{wc}. `CalculateTTInterference` takes as input the link dl_u and the current t_{wc}, and outputs the worst-case interference I_{max} due to TT frames, which has to be added to the t_{wc} received as input. The algorithm loops through the *Gate Close Events* (GCEs) gce_j related to the link dl_u (lines 3–12). The gce is represented as a circular linked list, where gce_{ptr} is the current element, $gce.duration$ the time the gate is closed, and $gce.next$ is the next GCE. In the while loop (lines 5–9) we calculate the time $I_{current}$ when the gate is closed and the AVB stream is delayed, considering that the stream started to transmit at gce_j. We increase $I_{current}$ in each iteration, until there is no remaining fragment rem of the AVB frame to be transmitted. If gce_j is giving a larger worst-case $I_{current}$, we update I_{max}.

For the example in Fig. 5, the worst-case is given by the placement indicated in the figure, where the AVB frame is depicted as a rectangle with a red border.

6.4 GRASP

GRASP [9] is a meta-heuristic optimization, which searches for that solution which minimizes the *cost function*. GRASP is implemented as an iterative algorithm, where each iteration has two phases; (i) which constructs an initial solution (a routing assignment to each stream $s_i \in \mathbf{S}^{\text{AVB}}$) based on a randomized greedy algorithm and (ii) which performs a local search on the constructed solution to reach the local minimum. At the end of each iteration, if the cost of the local minimum found is lower than the cost of the best solution found, so far, the solution is stored as the "best-so-far". The termination condition is based on a given time limit.

Phase (i) is implemented as an iterative algorithm, which loops until all streams in $\mathbf{S}^{\text{AVB}}$ have een assigned a route. In each iteration, we select randomly and remove from $\mathbf{S}^{\text{AVB}}$ a datapath dp_j of a stream $s_i \in \mathbf{S}^{\text{AVB}}$. For dp_j, we greedily try at random $\alpha = K/2$ possible routes selected among the K possible candidates, and we keep the best routing solution $\mathcal{R}$ found. Once we have constructed a complete routing solution for all streams in $\mathbf{S}^{\text{AVB}}$, we use it as a starting point in a local search in Phase (ii).

Phase (ii) is based on a *Hill Climbing* algorithm. The routing candidate that leads to largest decrease in the overall cost is then selected as dp_j's new route. Whenever the cost has not improved for $\beta = |\mathbf{S}^{\text{AVB}}|$ iterations, the local search is terminated. The values of the parameters α and β are based on empirical tests.

7. EXPERIMENTAL EVALUATION

To evaluate our proposed solution, we have used four different network topologies each with one or more application sets: `MOTIV`, the topology and application-set introduced in Sect. 5.1. `SYNTH` a synthetic test-case created by us, `ABB`, an Industry 4.0 case study from ABB, with a mesh like topology that has a high connectivity and the `ORION` test-case which uses the architecture of the Orion Crew Exploration vehicle, adapted from [15] to consider TSN.

The number of ESs, BRs and the link rates are presented in columns 2–4, respectively. For each test case, we show in column 5 and 6 the number of AVB and TT streams, respectively. Due to the

	Architecture			Application		SFS		RO		
ID	$\|ES\|$	$\|BR\|$	Rate	$\|\mathbf{S}^{AVB}\|$	$\|\mathbf{S}^{TT}\|$	O_1	O_3	O_1	O_3	cost
MOTIV_T1	7	4	100 Mbps	3	1	1	12	0	14	20.60
SYNTH_T1	10	4	100 Mbps	4	1	4	14	0	18	24.04
ORION_T1	31	15	1 Gbps	20	3	3	139	0	136	170.49
ORION_T2				35	5	8	226	0	223	303.12
ABB_T1				18	1	7	175	0	167	206.99
ABB_T2	20	36	1 Gbps	16	3	5	155	0	145	179.94
ABB_T3				16	6	7	155	1	151	—

Table 2: Comparison of RO vs. SFS

lack of space, we are not able to give all the details of the stream parameters. However, instead, we make available the input files of the test cases (containing all the details) and the software used to obtain the results, see [11]. For our experiments we have used the weights $W_1 = 10,000, W_2 = 3$ and $W_3 = 1$ as well as $K = 50$ determined empirically to normalize the effects of O_2 and O_3 while heavily penalizing unschedulable solutions (O_1).

We are interested to determine the quality of our GRASP-based Routing Optimization (RO) approach. Thus, we have compared the results obtained with RO for each test case, with the results of the straightforward solution (SFS), which always uses the shortest paths for the routes. The results of the comparison can be found in Table 2. For RO, we have used a 15 minute time-limit in all experiments on an Intel i7-2600K processor.

For RO we have three columns of results whereas for SFS we have two columns. In the columns labeled O_1, we have the number of unschedulable AVB streams, out of the total AVB streams presented in column 5. A zero means that the solution is schedulable. As we can see SFS is not able to obtain schedulable results, whereas our proposed RO can find schedulable solutions in every case except for ABB_T3 (it is unknown if a feasible solution exists for this example). In the column labeled O_3, we have the number of datalinks needed for each routing solution. A small number of links means less utilization. As we can see, besides finding schedulable solutions, our RO is able to reduce the number of links needed, compared to SFS. Finally, we also show the cost function value for RO. Note that since SFS results are not schedulable, the cost function is heavily penalized so we do not show it in the table. We denote with — the penalized cost function for RO in the last column.

We were also interested to compare the results obtained by RO with the optimal results obtained by exhaustive search. We were able to obtain the optimal result on MOTIV, and RO was able to obtain the same optimal result on the test case. However, we were not able to complete an exhaustive search on the larger test cases. For example, an exhaustive evaluation using just $K = 2$ on the ORION and ABB test-cases will take at least half a year to process on the used system and the size of the search-space grows in the order of $O(|\mathbf{S}|^K)$. We believe that $K = 25$ or higher is needed to even find schedulable solutions on the ABB test cases.

8. CONCLUSIONS

We have proposed an optimization strategy for the routing of AVB streams in TSN-based systems. We have seen that our GRASP-based metaheuristic on top of the *K shortest path* search space reduction technique can solve effectively the optimization problem presented in Sect. 5. In order to evaluate the timing properties of a given routing candidate, we have extended the delay formula from *IEEE 802.1BA* to take into account the effect of TT traffic and preemption. The evaluation of the optimization strategy on several test-cases indicates that it is possible to find good quality solutions within a reasonable time.

9. ACKNOWLEDGMENTS

The research leading to these results has received funding from the Advanced Research & Technology for Embedded Intelligence and Systems (ARTEMIS) Joint Undertaking within the project EMC^2, under grant agreement no. 621429.

10. REFERENCES

[1] G. Alderisi, G. Patti, and L. Bello. Introducing support for scheduled traffic over IEEE audio video bridging networks. In *Proc. of the Emerging Technologies Factory Automation Conference*, pages 1–9, 2013.

[2] A. AlSheikh, O. Brun, M. Chéramy, and P.-E. Hladik. Optimal design of virtual links in AFDX networks. *Real-Time Systems*, 49(3):308–336, 2012.

[3] ARINC. Aircraft Data Network, Part 7, Avionics Full-Duplex Switched Ethernet Network. Technical report, 2009.

[4] U. Bordoloi, A. Aminifar, P. Eles, and Z. Peng. Schedulability analysis of Ethernet AVB switches. In *Proc. of the Embedded and Real-Time Computing Systems and Applications Conference*, pages 1–10, 2014.

[5] J. A. R. De Azua and M. Boyer. Complete Modelling of AVB in Network Calculus Framework. In *Proc. of the International Conference on Real-Time Networks and Systems*, pages 55–64, 2014.

[6] J. D. Decotignie. Ethernet-Based Real-Time and Industrial Communications. *Proc. of the IEEE*, 93(6):1102–1117, 2005.

[7] J. Diemer, D. Thiele, and R. Ernst. Formal worst-case timing analysis of Ethernet topologies with strict-priority and AVB switching. In *Proc. of the IEEE Intl. Symp. on Industrial Embedded Systems*, pages 1–10, 2012.

[8] D. Eppstein. Finding the K Shortest Paths. *SIAM J. Comput.*, 28(2):652–673, Feb. 1999.

[9] T. A. Feo and M. G. C. Resende. A Probabilistic Heuristic for a Computationally Difficult Set Covering Problem. *Oper. Res. Lett.*, 8(2):67–71, Apr. 1989.

[10] M. D. Grammatikakis, D. Hsu, M. Kraetzl, and J. F. Sibeyn. Packet Routing in Fixed-Connection Networks: A Survey. *Journal of Parallel and Distributed Computing*, 54(2):77 – 132, 1998.

[11] S. M. Laursen. Time Sensitive Network Configuration Framework. https://github.com/SMLaursen/tsncf, 2016.

[12] P. Meyer, T. Steinbach, F. Korf, and T. Schmidt. Extending IEEE 802.1 AVB with time-triggered scheduling: A simulation study of the coexistence of synchronous and asynchronous traffic. In *Proc. of the IEEE Vehicular Networking Conference*, pages 47–54, 2013.

[13] J. Rushby. A Comparison of Bus Architectures for Safety-Critical Embedded Systems. Technical report, Computer Science Laboratory, SRI International, 2001.

[14] SAE. AS6802 : Time-Triggered Ethernet. Technical report, November 2011.

[15] D. Tămaş-Selicean, P. Pop, and W. Steiner. Design optimization of TTEthernet-based distributed real-time systems. *Real-Time Systems*, 51(1):1–35, 2014.

[16] L. G. Valiant. The Complexity of Enumeration and Reliability Problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[17] B. Wang and J. C. Hou. Multicast routing and its QoS extension: problems, algorithms, and protocols. *IEEE Network*, 14(1):22–36, 2000.