Design optimisation of cyber-physical distributed systems using IEEE time-sensitive networks

ISSN 2398-3396
Received on 20th October 2016
Revised on 6th November 2016
Accepted on 8th November 2016
doi: 10.1049/iet-cps.2016.0021
www.ietdl.org

Paul Pop[1] ✉, *Michael Lander Raagaard*[1], *Silviu S. Craciunas*[2], *Wilfried Steiner*[2]

[1]*Technical University of Denmark, Kongens Lyngby, Denmark*
[2]*TTTech Computertechnik AG, Vienna, Austria*
✉ *E-mail: paupo@dtu.dk*

Abstract: In this study the authors are interested in safety-critical real-time applications implemented on distributed architectures supporting the time-sensitive networking (TSN) standard. The on-going standardisation 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) flows with bounded end-to-end latency as well as best-effort messages. The authors first present a survey of research related to the optimisation of distributed cyber-physical systems using real-time Ethernet for communication. Then, the authors formulate two novel optimisation problems related to the scheduling and routing of TT and AVB traffic in TSN. Thus, the authors consider that they know the topology of the network as well as the set of TT and AVB flows. The authors are interested to determine the routing of both TT and AVB flows as well as the scheduling of the TT flows such that all frames are schedulable and the AVB worst-case end-to-end delay is minimised. The authors have proposed an integer linear programming formulation for the scheduling problem and a greedy randomised adaptive search procedure-based heuristic for the routing problem. The proposed approaches have 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 [1]. In this paper we are interested in the collection of standards colloquially known as time-sensitive networking (TSN) which are supplementing the well-known *IEEE 802.1* Ethernet architecture with real-time capabilities. [We will not provide references to all standards, but these can be found based on their name.] 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 [2]. Therefore, several extensions to Ethernet have been proposed, such as, TTEthernet [3] and ARINC 664 specification part 7 [4]. The *IEEE 802.1 Higher Layer LAN Protocols Working Group* has also moved in this direction by standardising a set of enhancements making up TSN, introducing new traffic shapers enabling *IEEE 802.1* to support safety-critical and real-time applications. We will collectively refer to these real-time and safety-critical Ethernet extensions as deterministic Ethernet (DE).

First, *IEEE 802.1Q-2005* introduced support for prioritising the best-effort (BE) traffic such that prioritised traffic could have a higher quality-of-service. 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 on-going and the complete set of substandards making up TSN has not yet been finalised. 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.

We consider real-time applications implemented using TSN distributed cyber-physical systems. As an input to our problem we have (i) the network topology, (ii) the set of TT flows and (iii) the set of AVB flows. We are interested in determining the TT schedules called gate control lists (GCL), and the routing of the TT and AVB flows, such that all flows are schedulable and their WCDs of AVB flows are minimised. For the GCL synthesis, we have proposed an integer linear programming (ILP) formulation. For TT flow routing, we use the shortest path as the route. However, for routing AVB flows, we have proposed a greedy randomised adaptive search procedure (GRASP)-based heuristic [5] on a search space which has been reduced using a *K* shortest paths-based algorithm [6].

1.1 Brief review of related work

This section presents a brief review of related work on design optimisation for DE networks. We start from a broad perspective, looking at topology design, the introduction of new traffic types and the assignment of traffic types to messages. We then present work related to typical communication synthesis problems, such as routing, scheduling, frame packing and fragmenting. A common constraint that needs to be satisfied is the schedulability of messages, hence we also discuss work on simulation and timing analysis.

1.1.1 Network planning and design: The problem of determining the network topology, i.e. the number of network switches and their interconnection via physical links and to the end systems (ES) is called 'network planning and design'. This problem has been addressed for DE in the context of industrial Ethernet [7] and TTEthernet in aerospace [8].

For safety-critical applications, the focus is on network reliability and redundancy optimisation. An annotated overview of system reliability optimisation which covers also network reliability is presented in [9]. In [10], the authors present the latest research results in network reliability optimisation. Several network reliability measures have been proposed in the literature, such as connectivity, resilience and performability. Researchers have proposed several approaches to the optimisation problem, including heuristics, metaheuristics and exact solutions based, for example, on mathematical programming [10].

However, these results cannot be applied directly to DE. One of the basic assumptions of earlier works on network reliability optimisation is that once a fault is detected, the network will reconfigure itself to avoid the fault. That is, new routes will be found for messages. In the case of DE the routes for safety-critical applications are typically *static*: they are loaded into the ES and network switches at design time, and it is not possible to change the routing dynamically, at runtime. In this context, researchers have proposed a fault-tolerant topology selection for TTEthernet [11]. However, for non-critical applications, runtime reconfiguration, including routing, is a relevant problem.

1.1.2 Traffic types: To increase its determinism, Ethernet has been extended with new 'traffic classes', which we call in this paper traffic *types*, to distinguish from the AVB traffic classes. The basic Ethernet traffic type is called 'best effort', and there are no timing guarantees provided for this traffic type. ARINC 664p7 [4] has introduced the rate constrained (RC) traffic type, that has bounded end-to-end latencies. TTEthernet [3] has extended ARINC 664p7 with the TT traffic type, which is transmitted based on static schedule tables and has the highest priority. We have mentioned the TT and AVB traffic types in the context of TSN. Researchers have also proposed new traffic types, such as the urgency-based scheduler (UBS) [12]. UBS assures low and predictable latency with a reduced implementation complexity, and the timing guarantees are provided in the absence of a global notion of time, which is required in TSN for the TT traffic types.

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 flows, the other AVB flows 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. However, for those situations where several traffic classes may be appropriate, researchers have also proposed optimisation methods to determine the traffic types for each message [13].

1.1.3 Routing: Routing optimisation is a well-studied subject where Wang and Hou [14] and Grammatikakis *et al.* [15] provide excellent overviews of the different centralised and distributed routing algorithms. Researchers have also addressed routing in safety-critical systems [16, 17]. For ARINC 664p7, Al Sheikh *et al.* [18] proposed an approach to find the optimal routes in ARINC 664p7 networks using mixed ILP. Tămaş-Selicean *et al.* [19] have used a Tabu search-based metaheuristic to, among other things, optimise the routing of the RC traffic type to minimise the WCDs in TTEthernet systems.

Regarding routing in TSN, AVB flows are typically established at runtime using the stream reservation protocol (SRP) where either the rapid spanning tree protocol or *shortest path bridging* are used to determine the routing. The future enhancements around TSN will support more sophisticated runtime routing algorithms, and the possibility to also determine the routes offline. We have proposed an offline routing optimisation approach for AVB in [20].

1.1.4 Scheduling: In this paper we focus on design optimisation problems related configuration of TSN networks. However, messages are produced by tasks, and the interaction between the task scheduling and message transmission has to be considered in the overall system design. Solutions to the problem of joint task and message scheduling have been proposed in the literature [21, 22] considering TT networks.

Researchers have proposed several approaches for the scheduling of time-division multiple access (TDMA) networks, such as TT protocol, the static segment of FlexRay and TDMA networks-on-chip, see [8] for a discussion. Regarding DE, there is a fundamental difference between the scheduling of frames in TTEthernet and in TSN. In TSN, as we will discuss in Section 4.1, the times in the schedule tables do not refer directly to frames as in TTEthernet, but to queues, which are controlled by 'gates' that may be open or closed based on GCLs.

For the scheduling TTEthernet frames, Steiner [23] proposed an approach for the synthesis of static TT schedules, where he ignored the RC traffic and used a satisfiability modulo theories (SMT)-solver to find a solution which satisfies an imposed set of constraints. The same author has proposed an SMT-solver approach to introduce periodic evenly-spaced slots into the static schedules to help reduce RC delays in [24]. More recent work has shown how the SMT-based approach can be extended to handle very large systems [25]. Suethanuwong [26] proposed a scheduling approach of the TT traffic, ignoring RC traffic, that introduces equally distributed available time slots for BE traffic. Tămaş-Selicean *et al.* [19] have used a Tabu search-based metaheuristic to determine the schedules of TT frames such that the WCDs of RC frames is minimised.

Recent work has also addressed the scheduling of TSN, i.e. *IEEE 802.1Qbv*. Craciunas *et al.* [27] discuss the issues affecting the deterministic behaviour of time sensitive traffic in TSN, and use SMT and optimisation modulo theories (OMT) solvers to decide the assignment of frames to queues and the queue GCLs such that the schedulability constraints are satisfied. In [28], the authors are interested to guarantee minimum network delay for time sensitive flows and map the scheduling problem to the 'no-wait job-shop scheduling problem', which is then solved using Tabu search.

1.1.5 Frame packing and fragmenting: Researchers have also addressed the issue of frame packing [29, 30]. Frame packing is one of the fundamental features for some communication protocols. For example, EtherCAT [31] is a master/slave protocol, where the master packs several 'datagrams' (i.e. messages) into a single frame, regardless of the destination, and sends the frame to all the slaves. Recent work has also addressed the ARINC 664p7 protocol. Ayed *et al.* [32] proposed a packing strategy for multi-cluster networks, where the critical avionics subsystems are based on CAN buses, and are interconnected via ARINC 664p7. This strategy, meant to minimise the CAN bandwidth through the ARINC 664p7 network, performs packing at the CAN-ARINC 664p7 gateway based on a timer. Messages are not packed based on destinations, but on availability. As a consequence, all the messages packed in a frame are delivered to all the possible destinations. Moreover for ARINC 664p7, Al Sheikh *et al.* [18] proposed a packing strategy for messages with the same source and destinations, with the goal of minimising the reserved bandwidth.

Mikolasek *et al.* [33] proposed a segmentation algorithm for the standard Ethernet messages in TT Ethernet, an academic Ethernet-based protocol that supports standard Ethernet traffic and TT messages. This algorithm fragments the standard Ethernet messages into smaller frames that can be transmitted between two TT frames, reducing transmission preemption and increasing throughput. For TTEthernet, researchers [19] have shown how the issues of frame packing and fragmenting can be integrated into an overall optimisation problem together with scheduling and routing, aiming at guaranteeing timing properties.

1.1.6 Simulation and timing analysis: The schedulability of the TT traffic is determined by the TT schedules (or GCLs in TSN), which have to be synthesised such that all frames meet their deadlines. For other types of traffic, researches have proposed analyses to determine the worst-case end-to-end delay. A flow is then schedulable if the WCD is smaller than its deadline. Several groups have developed academic simulators, for example [34, 35], but these do not provide any timing guarantees.

Latency analysis methods have been successfully applied to RC traffic in ARINC 664p7 networks [36]. However, they cannot be directly applied for the performance analysis of RC traffic in TTEthernet due to the static TT schedules. Zhao *et al.* [37] proposed a network calculus (NC)-based analysis to compute the WCD of RC frames by considering variable size of TT frames and focusing on the shuffling integration policy. A recent analysis for RC flows in TTEthernet has been proposed by Tămaş-Selicean *et al.* [38]. The authors use a response-time analysis based on the concept of 'busy period' and show that they are able to significantly reduce the pessimism compared with previous approaches.

For AVB traffic in TSN, the *AVB Latency Math* equation from *IEEE 802.1BA* specifies a WCD equation to be used as a decentralised admission test by the bridges (BR) when AVB flows 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.* [39] used compositional performance analysis to derive the WCD of the AVB flows. Bordoloi *et al.* [40] proposed improvements to this approach and supplied proofs of the correctness. De Azua and Boyer [41] proposed a theoretical NC model to derive the WCD. The AVB Latency Math has been extended in [20] to consider the effect of TT traffic on the latency of the AVB flows.

2 Architecture model

The Ethernet standard defines a switched multi-hop network being composed of ES and 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 $G(\boldsymbol{V}, \boldsymbol{E})$ where the set of vertices $\boldsymbol{V}$ is the union of all the $\boldsymbol{ES}$ and the BR $\boldsymbol{B}$, $\boldsymbol{V} = \boldsymbol{ES} \cup \boldsymbol{B}$. The edges $\boldsymbol{E}$ represent the physical connections and we denote the data link dl from the vertex $v_a \in \boldsymbol{V}$ to $v_b \in \boldsymbol{V}$ as $dl_u = [v_a, v_b]$. The physical transmission rate of this link is denoted by $dl\_u.rate$ and is typically 100 Mbps or 1 Gbps. An example model is presented in Fig. 1, having seven ESs and four 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 $ES_j \in \boldsymbol{ES}$ to one receiver $ES_k \in \boldsymbol{ES}$. 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

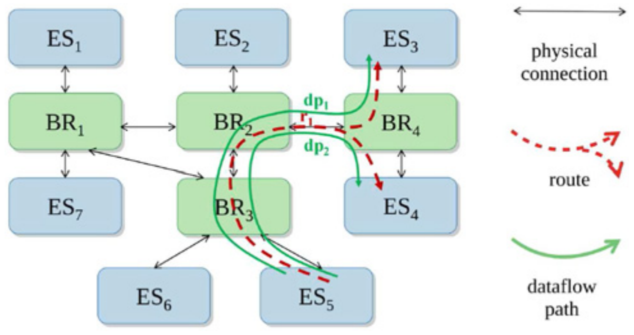

Fig. 1 *Example TSN topology model*

the term route r to denote the set of datapaths making up the route which may have multiple destinations. Hence, the r representing a multicast to n destinations is defined as the set of n datapaths, $r = \{dp_1, \ldots, dp_n\}$. 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*, denoted by f_m, 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 [19]. Both the AVB- and TT-traffic classes are used to carry periodic frame instances denoted as *flows*. The set of all AVB flows is defined as $\boldsymbol{S}^{\mathrm{AVB}}$ and the set of all TT flows $\boldsymbol{S}^{\mathrm{TT}}$. The BE traffic-class is not explicitly modelled as it is considered outside the scope of this paper.

Each TT flow $s_i \in \boldsymbol{S}^{\mathrm{TT}}$ is defined by the following attributes: $s_i.r$, which is the route, s_i.size, which is the maximum data size, $s_i.T$, which is the period of the flow, and $s_i.deadline$, which is the maximum end-to-end latency. The model used to capture the TT GCLs is presented in Section 4.1. If the message size exceeds the size of an Ethernet Maximum Transmission Unit (MTU), i.e. 1500 B, it is split into multiple frames. Thus, we define a flow instance, $F_k = \mathcal{F}_i^{[v_a, v_b]}$, as the set of frames of flow, s_i, that are transmitted on link, $[v_a, v_b]$. Without loss of generality we consider that TT flows are unicast, i.e. there is a single destination ES for each flow. For each TT flow $s_i \in \boldsymbol{S}^{\mathrm{TT}}$ we know its source $src(s_i) \in \boldsymbol{ES}$ and destination $dest(s_i) \subseteq \boldsymbol{ES}$.

AVB flows may be multicast. For each AVB flow $s_i \in \boldsymbol{S}^{\mathrm{AVB}}$ we know its source $src(s_i) \in \boldsymbol{ES}$ and destinations $\boldsymbol{dest}(s_i) \subseteq \boldsymbol{ES}$, as well as the maximum size of a frame in the flow $s_i.size$ and its class $s_i.class$, A or B. The TSN group is working towards supporting fully customisable 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 flow s_i, which may depend on the AVB class characteristics.

The routing of an AVB-flow s_i, to be determined by our routing optimisation, 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. The higher priority queues are used for the TT traffic, the following queues are used for different classes of AVB flows and finally the remaining queues are used for prioritised BE traffic. The AVB queues make use of a transmission selection algorithm in the form of the credit-based shaper (CBS), explained in Section 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. We assume that the gates are opened in

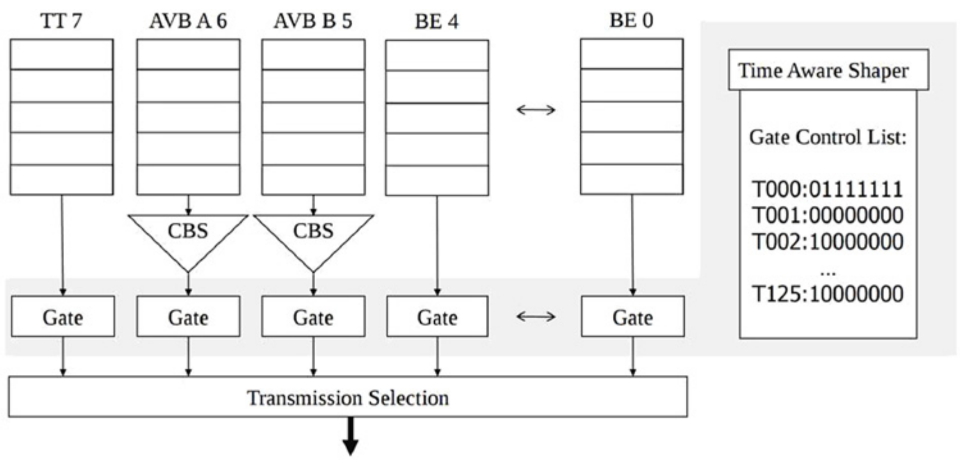

Fig. 2 *Typical TSN bridge configuration*

a mutually exclusive pattern, i.e. no two queue gates are opened at the same time.

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 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*, according to a port-specific 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, *T*000: 0111111 means that at time '*T*000' 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 is synchronised across the BR using the *time synchronisation* 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.

In our model, we capture the GCL of a TT flow s_i in terms of an offset, period and duration. For example, in Table 2 the GCL for s_4 is $\langle$offset, period, duration$\rangle = \langle 0\,\mu\text{s}, 62.5\,\mu\text{s}, 10.4\,\mu\text{s}\rangle$, 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.

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 standardised 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 1–4, 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 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

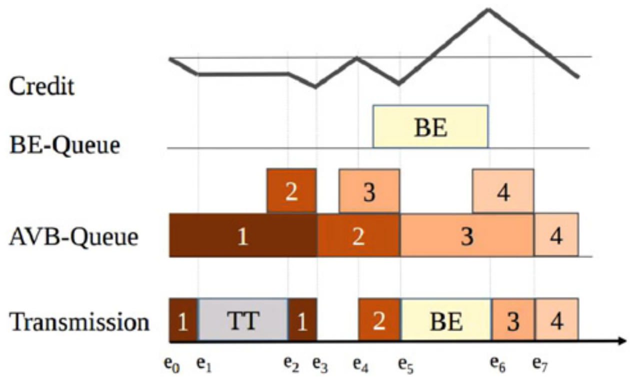

Fig. 3 *Example AVB transmission*

since the credit is negative, the lower prioritised 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 $G(\boldsymbol{E}, \boldsymbol{V})$, (ii) the set of AVB flows $\boldsymbol{S}^{\text{AVB}}$ with their properties and (iii) the set of TT flows $\boldsymbol{S}^{\text{TT}}$ with their properties. As an output of our problem, we determine (1) the routes r_i for each of the flows $s_i \in \boldsymbol{S}^{\text{AVB}} \cup \boldsymbol{S}^{\text{TT}}$ (2) the assignment of TT frames to egress port queues and (3) the GCLs for the queues, i.e. the offset ϕ_m for each frame f_m on all the TT flows $s_i \in \boldsymbol{S}^{\text{TT}}$.

We are interested to determine a solution such that (a) all the flows are schedulable ($s_i.deadline$ not exceeded), (b) the number of queues used by the TT flows is minimised (to maximise the queues available for AVB and BE flows) and (c) the WCD of the AVB flows $\boldsymbol{S}^{\text{AVB}}$ are minimised.

5.1 TT schedule synthesis example

Consider the topology from Fig. 4 with the TT flows specified in Table 1. The data size for the four flows is 2, 1, 1, and 3 times MTU, respectively, which means that the message is split into just as many frames. Furthermore, the deadline for each flow is assumed equal to its period. In this example we are interested to find a feasible schedule with the minimum number of TT queues. Fig. 5 shows an optimal schedule (in terms of number of queues) for the example. The figure shows the frame transmission on each link, corresponding to the points in time where the gate of the corresponding queue is open. The frame label denotes the flow ID, and the ID of the assigned egress port queue. For instance, on link $[ES_1, BR_1]$, the frame f_m of flow s_2 (blue) is scheduled at offset 0 and on queue 1. Flow s_2 has a period of 62.5 µs which means that f_m is repeated twice within the hyperperiod of all the flows, which is 125 µs in this example.

Note that two queues are used in switch BR_1 for the egress port associated with device ES_3. In particular, the two frames of flow s_1 are assigned to queue 2. This is inevitable in this example, because the same queue cannot be used simultaneously for frames from different links. This constraint, as well as others required to ensure feasible schedules, are discussed in more detail in Section 6.1.

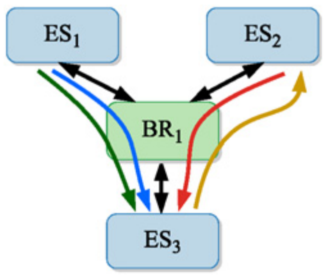

Fig. 4 *Network topology for TT schedule synthesis example*

Table 1 $\boldsymbol{S}^{\text{TT}}$ flows for TT schedule synthesis example

Stream	Route	Size, B	Period, µs
s_1	$[ES_2, BR_1], [BR_1, ES_3]$	3000	125
s_2	$[ES_1, BR_1], [BR_1, ES_3]$	1500	62.5
s_3	$[ES_1, BR_1], [BR_1, ES_3]$	1500	62.5
s_4	$[ES_3, BR_1], [BR_1, ES_1]$	4500	62.5

5.2 AVB routing optimisation example

Let us consider the topology from Fig. 1 with the flows summarised in Table 2. 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 flows in Table 2 denotes the corresponding AVB utilisation of the respective flow on a 100 Mbps network and is calculated as $U_{100} = \left(s_i.size/s_i.period \cdot 100\,\text{Mbps}\right)$. This measure is only used as a part of this example.

We have proposed in [20] an approach 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 utilisation measure, that an AVB flow s_i of class x is schedulable, if it is routed such that no data link in the route is utilised more than A_x by AVB and TT traffic. Let us assume that the TT flow s_4 corresponds to a bandwidth utilisation 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 optimise the routing of AVB flows 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 5.2 where the flows s_2 and s_3 are not schedulable, i.e. the sum of utilisation contributions from each flow exceeds 75% on the datalinks $[BR_1, BR_2]$ and $[BR_3, BR_2]$. An optimised solution is illustrated at 5.2 where, compared with the infeasible solution in 5.2, 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 optimising the routing of AVB flows we are able to find schedulable solutions.

6 Optimisation strategy

The scheduling problem presented in the previous section is similar to the flow-shop scheduling problem which is known to be NP-complete [42], with the packing and fragmenting of frames adding to the complexity of the problem. The routing optimisation problem is NP-hard. Exhaustively enumerating every path between two vertices has been proven NP-hard [43].

Our proposed optimisation strategy consists of the following steps: (i) In the first step, we decide the routes of the TT flows

Table 2 $\boldsymbol{S}$ used for the motivational example

AVB Class A flows $\boldsymbol{S}^{\text{AVB}}$					
Stream	Endpoints	Size, B	Period, µs	Deadline, ms	U_{100}, %
s_1	$ES_1 \rightarrow ES_4$	400	62.5	2	51
s_2	$ES_5 \rightarrow ES_3, ES_4$	350	125	2	22
s_3	$ES_7 \rightarrow ES_2$	480	125	2	31
	TT flows $\boldsymbol{S}^{\text{TT}}$				
Stream	Route		GCL, µs		
s_4	$[ES_6, BR_3], [BR_3, BR_2], [BR_2, ES_2]$		$\langle 0, 62.5, 10.4 \rangle$		

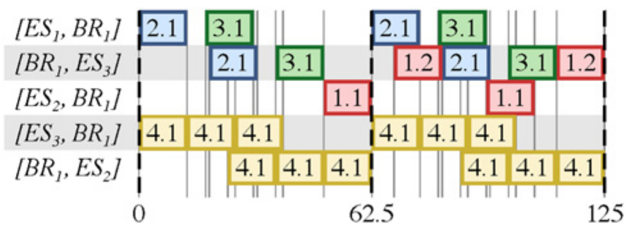

Fig. 5 *Feasible solution for TT schedule synthesis example*

using Dijkstra's algorithm that finds the shortest paths in a graph. (ii) In the second step, we use an ILP formulation called *GCL Synthesis* (GS), see Section 6.1, to solve the GCL synthesis problem for scheduling the TT flows. (iii) Finally, given the TT routes and schedules obtained in the first two steps, in the third step we decide the routes for the AVB flows using a GRASP-based approach presented in Section 6.2.

6.1 ILP for TT schedule synthesis

The problem of deciding the GCL for each egress port such that all TT flows are schedulable is NP-hard. For each egress port, we wish to find a schedule that minimises the number of queues used for TT traffic, such that we maximise the number of queues available for AVB and BE traffic. We use an ILP approach to solve this optimisation problem.

Similar to [27], we consider that both the ES and BR are 'scheduled', i.e. they are synchronised, hence we can rely on the schedules produced to isolate the TT flows. We have adapted the SMT constraints from [27] to be used in our ILP formulation.

In our ILP model we use two primary decision variables, $\phi_m \in \mathbb{N}$ and $\rho_k \in \mathbb{N}$. ϕ_m determines the offset in μs for frame, f_m, within its period. ρ_k determines the ID of the queue which flow instance, F_k, is assigned to.

An auxiliary variable, $\kappa_{a,b} \in \mathbb{N}$, is introduced for each link, $[v_a, v_b]$. It models the number of queues used for TT traffic in v_a on the egress port to v_b. The objective function is specified as in (1), with constraint (2) ensuring the desired semantics for κ.

$$\min \sum_{[v_a,v_b]\in E} \kappa_{a,b} \tag{1}$$

$$\text{s.t.} \quad \rho_k \le \kappa_{a,b}\left(\forall [v_a, v_b] \in E\right)\left(\forall F_k \in \mathcal{I}^{[v_a,v_b]}\right) \tag{2}$$

$\mathcal{I}^{[v_a,v_b]}$ denotes all flow instances on link, $[v_a, v_b]$.

Additional auxiliary variables and constraints are introduced to enforce the correct temporal behaviour for TSN communication. These are described in the following.

6.1.1 Frame constraints: Each frame should be scheduled such that it completes within its period. This is captured in constraint (3), where $\mathcal{F}$ denotes the set of all TT frames.

$$\phi_m \le f_m.T - f_m.L \quad \left(\forall f_m \in \mathcal{F}\right) \tag{3}$$

6.1.2 Link constraints: A physical link can only transmit a single frame at a time. This means that frames on the same link cannot overlap in the time domain. Because two frames, f_m and f_n, on a link can have different periods it is necessary to enforce the link constraint for the hyperperiod, $hp_{m,n}$, of the two periods, $f_m.T$ and $f_n.T$. This is calculated as the least common multiple, $lcm(f_m.T, f_n.T)$. After this point in time the schedule for the two considered flows repeats. Constraint (4) models the case where f_m finishes before f_n starts, constraint (5) models the opposite case. The binary variable, $\sigma_{m(\alpha),n(\beta)} \in \{0, 1\}$ is introduced to model disjunction between the two cases by adding a large constant, M, to one of the cases – but not both – thereby trivially satisfying the inequality.

$$\begin{aligned}&\left(\phi_m + \alpha \cdot f_m.T\right) + f_m.L \\ &\quad \le \left(\phi_n + \beta \cdot f_n.T\right) + M \cdot \left(1 - \sigma_{m(\alpha),n(\beta)}\right)\end{aligned} \tag{4}$$

$$\begin{aligned}&\left(\phi_n + \beta \cdot f_n.T\right) + f_m.L \\ &\quad \le \left(\phi_m + \alpha \cdot f_m.T\right) + M \cdot \sigma_{m(\alpha),n(\beta)} \\ &\quad\quad \left(\forall f_m, f_n \in \mathcal{F}^{[v_a,v_b]} | m \ne n\right)(\forall \alpha \in A)\left(\forall \beta \in B\right)\end{aligned} \tag{5}$$

where $A = \{0, 1, \dots, (hp_{m,n}/f_m.T) - 1\}$ and $B = \{0, 1, \dots, (hp_{m,n}/f_n.T) - 1\}$. Thus, each pair of $\alpha \in A$ and $\beta \in B$ represents a scenario where f_m and f_n potentially overlap in the hyperperiod of $f_m.T$ and $f_n.T$. Analogously to $\mathcal{I}^{[v_a,v_b]}$, $\mathcal{F}^{[v_a,v_b]}$ denotes all TT frames on $[v_a, v_b]$.

6.1.3 Flow transmission constraints: A frame of a flow must be fully transmitted on one link before transmission is initiated on the subsequent link of the route. This is modelled in constraint (6).

$$\begin{aligned}&\phi_m + f_m.L + \delta \le \phi_n \\ &\quad \left(\forall s_i \in S\right)\left(\forall f_m \in \mathcal{F}_i^{[v_x,v_a]}\right)\left(\forall f_n \in \mathcal{F}_i^{[v_a,v_b]}\right)\end{aligned} \tag{6}$$

$\mathcal{F}_i^{[v_a,v_b]}$ denotes the set of frames (flow instance) of flow, s_i on $[v_a, v_b]$. δ denotes the network precision, i.e. the maximum difference in the local clocks of any two devices in the network.

6.1.4 End-to-end constraints: All flows must arrive at their destination within their deadline, i.e. the end-to-end latency cannot exceed the specified deadline. This is captured in constraint (7)

$$\begin{aligned}&\left(\phi_n + f_n.L\right) - \phi_m \le s_i.e2e \\ &\quad \left(\forall s_i \in S | m = \text{first}\left(\mathcal{F}_i^{src(s_i)}\right) \wedge n = \text{last}\left(\mathcal{F}_i^{\text{dest}(s_i)}\right)\right)\end{aligned} \tag{7}$$

where $src(s_i)$ and $\text{dest}(s_i)$ denote the first and last link in the route of flow s_i, respectively. Furthermore, $\text{first}(F_k)$ and $\text{last}(F_k)$, denote the first and last frame of a flow instance, respectively.

6.1.5 Frame isolation constraints: To prevent frames of different TT flows to interleave in the egress queues, a frame isolation constraint is introduced. The constraint specifies that frame, f_m, of one flow cannot arrive at a device as long as a another frame, f_n, from another flow is in the queue. f_m can only arrive at the device after f_n has left the queue and its transmission on the associated link is initiated. This is modelled in constraints in the following equations.

$$\begin{aligned}&\phi_n + \beta \cdot f_n.T + \delta \\ &\quad \le \phi'_m + \alpha \cdot f_m.T + M \cdot \left(\omega_{m(\alpha),n(\beta)} + \varepsilon_{k,l} + \varepsilon_{l,k}\right)\end{aligned} \tag{8}$$

$$\begin{aligned}&\phi_m + \alpha \cdot f_m.T + \delta \\ &\quad \le \phi'_n + \beta \cdot f_n.T + M \cdot \left(\left(1 - \omega_{m(\alpha),n(\beta)}\right) + \epsilon_{k,l} + \epsilon_{l,k}\right) \\ &\quad\quad (\forall [v_a, v_b] \in E)(\forall F_k, F_l \in \mathcal{I}^{[v_a,v_b]} | k \ne l) \\ &\quad\quad (\forall f_m \in F_k)\left(\forall f_n \in F_l\right)(\forall \alpha \in A)\left(\forall \beta \in B\right)\end{aligned} \tag{9}$$

where ϕ'_m denotes the offset of frame f_m on its previous link, and similarly for ϕ'_n. The binary variable, $\omega_{m(\alpha),n(\beta)}$, is used to implement disjunction in the same way as $\sigma_{m(\alpha),n(\beta)}$ in Section 6.1.2. The binary variable, $\varepsilon_{k,l} \in \{0, 1\}$, is 1 if $\rho_k < \rho_l$ and 0 otherwise. Thus, the expression, $\varepsilon_{k,l} + \varepsilon_{l,k}$, evaluates to 1 if F_k and F_l are different queues, and 0 otherwise, i.e. the constraints are only valid for flow instances assigned the same queue. Constraints (10) and (11) are required to enforce the described semantics of $\varepsilon_{k,l}$ and $\varepsilon_{l,k}$

$$\rho_l - \rho_k - 1 - M \cdot \left(\epsilon_{k,l} - 1\right) \ge 0 \tag{10}$$

$$\begin{aligned}&\rho_l - \rho_k - M \cdot \epsilon_{k,l} \le 0 \\ &\quad \left(\forall [v_a, v_b] \in E\right)(\forall F_k, F_l \in \mathcal{I}^{[v_a,v_b]})\end{aligned} \tag{11}$$

6.2 GRASP-based AVB routing optimisation

We assume that the TT flows have been routed and scheduled, and that all the TT flows meet their deadlines. In this section we are concerned with the routing of AVB flows. Our proposed AVB *routing optimisation* (RO) approach, uses the following strategy. First, we reduce the search space using a K shortest paths heuristic as described in Section 6.2.2. Then, we employ a GRASP-based metaheuristic, presented in Section 6.2.3 to search the *reduced search space* where each candidate solution is evaluated using the cost function presented in Section 6.2.1.

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

$$\mathrm{cost}(\mathcal{R}) = O_1(\mathcal{R}) \cdot W_1 + O_2(\mathcal{R}) \cdot W_2 + O_3(\mathcal{R}) \cdot W_3 \qquad (12)$$

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

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

where $\mathcal{T}_{\mathrm{wc}}$ is the WCD of an AVB flow calculated as presented by us in [20]. The first term of (12) 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 penalised with the weight W_1.

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

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

It is envisioned that a practical implementation will use individual weights for every flow so that they can be prioritised, 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 utilisation 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 flows $s_i \in \boldsymbol{S}^{\mathrm{AVB}}$. This way, shorter-routes and multicasts with late branching points are preferred.

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

6.2.2 Search space reduction: We reduce the search space to only consider the K shortest path of every datapath dp_j of every flow $s_i \in \boldsymbol{S}^{\mathrm{AVB}}$ using the *K shortest paths* algorithm [6]. K shortest paths returns K unique routes of increasing length, starting from the shortest route. For example, for the applications in Table 2 and $K = 1$ only the shortest paths as depicted in 5.2 are considered, which leads to an infeasible solution. However with $K = 2$ longer paths are considered as well, and the feasible solution depicted in 5.2 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 utilisation. 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 Section 7, will lead to schedulable solutions.

The *K shortest path* algorithm has a time complexity of $O(|\boldsymbol{E}| + |\boldsymbol{V}| \cdot \log |\boldsymbol{V}| + K)$, where $|\boldsymbol{V}|$ 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.

6.2.3 GRASP: GRASP [5] is a meta-heuristic optimisation, which searches for that solution which minimises 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 flow $s_i \in \boldsymbol{S}^{\mathrm{AVB}}$) based on a randomised 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 illustrated in Algorithm 1 (see Fig. 6). We start from an empty solution **sol**, and we construct a complete solution, for all the flows in $\boldsymbol{S}^{\mathrm{AVB}}$, one route at a time. The routes are determined for each dataflow path dp_j of a flow $s_i \in \boldsymbol{S}^{\mathrm{AVB}}$. Thus, the **removeRandom** function removes and returns a not yet processed datapath dp randomly from a flow in $\boldsymbol{S}^{\mathrm{AVB}}$. For dp, we greedily try at random several possible routes. Thus, the **assignPath** function assigns a route r amongst the K possible candidates. We try θ routes for each dp, we keep the best routing solution found for dp, and we continue the while loop in Algorithm 1 (Fig. 6).

Phase (i) is implemented as an iterative algorithm, which loops until all flows in $\boldsymbol{S}^{\mathrm{AVB}}$ have been assigned a route. In each iteration, we select randomly and remove from $\boldsymbol{S}^{\mathrm{AVB}}$ a datapath dp_j of a flow $s_i \in \boldsymbol{S}^{\mathrm{AVB}}$. For dp_j, we greedily try at random $\theta = K/2$ 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 flows in $\boldsymbol{S}^{\mathrm{AVB}}$, 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 $\beta = |\boldsymbol{S}^{\mathrm{AVB}}|$ iterations, the local search is terminated. The values of the parameters θ and β are based on empirical tests.

Algorithm 1

```
function CONSTRUCTSOLUTION
    sol ← {}
    while $S^{AVB}$ not empty do
        $dp$ ← removeRandom($S^{AVB}$);
        best_r ← ∅
        for repeat $\theta$ times do
            curr_r ← assignPath($dp$)
            if cost(sol ∪ curr_r) < cost(sol ∪ best_r) then
                best_r ← curr_r
            end if
            sol ← sol ∪ best_r
        end for
    end while
    return sol
end function
```

Fig. 6 *Phase (i)*

7 Experimental evaluation

In our first set of experiments we have compared our GCL Synthesis ILP approach with the SMT approach from [27], i.e. the 'optimised frame isolation' results from Fig. 7. To facilitate the comparison, we have assumed as in [27] that the macrotick (which defines the granularity of time events for the links), propagation delay, and transmission rate, of all links are 1 µs, 0, and 1 Gbps, respectively. We have performed the comparison on the test cases listed in the first column of Table 3, which were taken from [27]. The number of ES and BR are presented in the two 'Architecture' columns and the number of TT flows are in column 4. We have used the *shortest path* to route the TT flows in the architectures. For the details of the test cases, the reader is referred to [27]. The AVB flows were ignored in this experiment. This first set of experiments

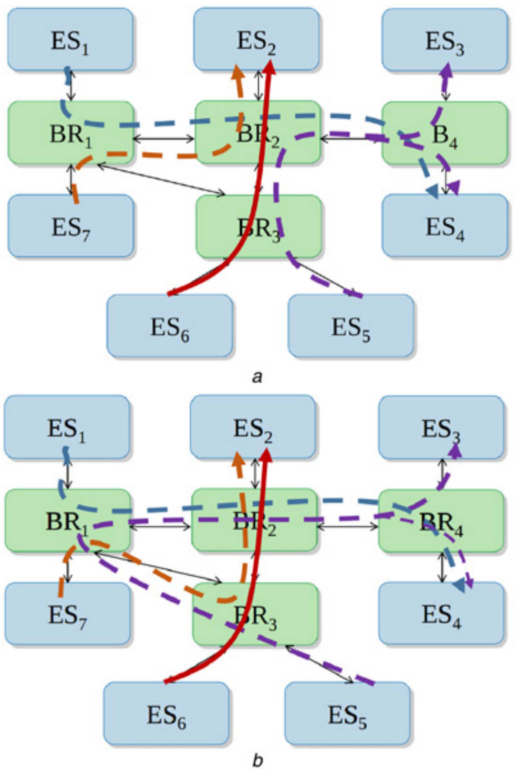

Fig. 7 *Two routing solutions of the flows in Table 2 on the network from Fig. 1. The colored dashed lines represent AVB flows and the full red line is the TT flow*

a Shortest routes
b Optimised routes

has been run on an 2.5 GHz Intel Core i5 processor, using the CPLEX ILP solver [44].

For all the test cases, our ILP formulation has obtained the optimal results, i.e. the same minimum number of TT queues such that all the TT frames are schedulable and all the scheduling constraints specified in Section 6.1 are satisfied, i.e. the schedules are valid. The last two columns in Table 3 show the time, in seconds, for ILP and SMT, respectively. As we can see from the table, our ILP can obtain the optimal results in shorter time, except for the last two test cases. In general, our ILP formulation does not scale well with the problem size, as we can see from the last two rows in Table 3. As future work, we are interested to implement a heuristic algorithm that can solve industrial-scale problem sizes, and which can take into account during the GCL synthesis the impact of the TT frames on the AVB frames' schedulability.

In our second set of experiments we were interested to evaluate our proposed AVB RO solution. We have used four different network topologies each with one or more application sets, see Table 4 **MOTIV**, the topology and application-set introduced in Section 5.2. **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 [19].

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 flows, respectively. 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 normalise the effects of O_2 and O_3 while heavily penalising unschedulable solutions (O_1). The GCLs used for the TT frames have been determined using the approach from [19], and adapted to consider TSN.

We are interested to determine the quality of our GRASP-based AVB RO approach. Thus, we have compared the results obtained with RO for each test case, with the results of a straightforward solution (SFS), which always uses the shortest paths for the routes. The results of the comparison can be found in Table 4. For RO, we have used a 15 min 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 labelled O_1, we have the number of unschedulable AVB flows, out of the total AVB flows 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 labelled O_3, we have the number of datalinks needed for each routing solution. A small number of links means less utilisation. As we can see, besides finding schedulable solutions, our RO is able to reduce the number of links needed, compared with SFS. Finally, we also show the cost function value for RO. Note that since SFS results are not schedulable, the cost function is heavily penalised so we do not show it in the table. We denote with – the penalised 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.

Table 3 Comparison of ILP vs. SMT

	Architecture		Appl.	ILP	SMT
ID	$\|ES\|$	$\|BR\|$	$\|\mathbf{S}^{TT}\|$	Time, s	Time
T01	3	1	5	0.58	0.33
T04	3	1	5	2.57	7.84
T05	3	1	3	4.61	321.76
T10	5	2	5	5.60	3.33
T11	5	2	4	7.38	64.76
T18	3	1	5	8.63	12.92
T12	5	2	5	80.19	1.13
T14	3	1	3	44.15	0.51

Table 4 Comparison of RO against SFS

	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	20	36	1 Gbps	18	1	7	175	0	167	206.99
ABB_T2				16	3	5	155	0	145	179.94
ABB_T3				16	6	7	155	1	151	–

8 Conclusions

In this paper we have addressed distributed cyber-physical systems that use TSN for the communication infrastructure. We have discussed the typical optimisation problems addressed in this context, and formulated a problem related to the TT flows scheduling and AVB flows routing.

We have proposed an ILP formulation to determine the assignment of TT flows to the queues in the BR and the synthesis of the gate control lists, which control the opening and closing of the queues. We have applied our ILP approach to several test cases and we have compared it with the related work that uses SMT solvers to derive a solution. Our ILP formulation is able to quickly find optimal solutions for smaller test cases, but does not scale well for larger test cases.

We have also proposed a GRASP-based optimisation strategy for the routing of AVB flows. We have seen that our GRASP-based metaheuristic on top of the *K shortest path* search space reduction technique can solve effectively the AVL routing optimisation problem. In order to evaluate the timing properties of a given routing candidate, we have used the worst-case delay analysis from [20], which has extended the *delay formula* from *IEEE 802.1BA* to take into account the effect of TT traffic and preemption. The evaluation of the AVB routing optimisation strategy on several test-cases indicates that it is possible to find good quality solutions within a reasonable time.

In our future work we are interested in heuristic algorithms that can solve the two problems simultaneously, taking into account the influence of TT flows on the AVB frame latencies, and which are capable of tackling large realistic test cases.

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. The GRASP algorithm has been implemented by Sune Mølgaard Laursen at the Technical University of Denmark, who has also contributed to earlier drafts of this manuscript.

10 References

1 Rushby, J.: ‘A comparison of bus architectures for safety-critical embedded systems’. Technical Report, Computer Science Laboratory, SRI International, 2001
2 Decotignie, J.D.: ‘Ethernet-based real-time and industrial communications’, *Proc. IEEE*, 2005, **93**, (6), pp. 1102–1117
3 SAE: ‘AS6802: time-triggered ethernet’.Technical Report, 2011, p. 108
4 ARINC: ‘Aircraft data network, part 7, avionics full-duplex switched ethernet network’. Technical Report, 2009
5 Feo, T.A., Resende, M.G.C.: ‘A probabilistic heuristic for a computationally difficult set covering problem’, *Oper. Res. Lett.*, 1989, **8**, (2), pp. 67–71
6 Eppstein, D.: ‘Finding the K shortest paths’, *SIAM J. Comput.*, 1999, **28**, (2), pp. 652–673
7 Krommenacker, N., Rondeau, E., Divoux, T.: ‘Genetic algorithms for industrial Ethernet network design’. Proc. of IEEE Int. Workshop on Factory Communication Systems, 2002, pp. 149–156
8 Tămaş-Selicean, D., Pop, P., Madsen, J.: ‘Design of mixed-criticality applications on distributed real-time systems’. PhD thesis, Technical University of Denmark, 2014
9 Kuo, W., Prasad, V.R.: ‘An annotated overview of system-reliability optimization’, *IEEE Trans. Reliab.*, 2000, **49**, (2), pp. 176–187
10 Konak, A., Smith, A.E.: ‘Network reliability optimization’, in Resende, Mauricio G.C., Pardalos, Panos M (Eds.): ‘Handbook of optimization in telecommunications’ (Springer, 2006, 1st edn.), pp. 735–760
11 Gavrilut, V., Tămaş-Selicean, D., Pop, P.: ‘Fault-tolerant topology selection for TTEthernet networks’. Proc. Safety and Reliability of Complex Engineered Systems Conf., 2015, pp. 4001–4009
12 Specht, J., Samii, S.: ‘Urgency-based scheduler for time-sensitive switched ethernet networks’. Proc. of Euromicro Conf. on Real-time Systems, 2016, pp. 75–85
13 Gavrilut, V., Pop, P.: ‘Traffic class assignment for mixed-criticality frames in TTEthernet’. SIGBED Review, 2016
14 Wang, B., Hou, J.C.: ‘Multicast routing and its QoS extension: problems, algorithms, and protocols’, *IEEE Netw.*, 2000, **14**, (1), pp. 22–36
15 Grammatikakis, M.D., Hsu, D., Kraetzl, M., *et al.*: ‘Packet routing in fixed-connection networks: a survey’, *J. Parallel Distrib. Comput.*, 1998, **54**, (2), pp. 77–132
16 Herpel, T., Kloiber, B., German, R., *et al.*: ‘Routing of safety-relevant messages in automotive ECU networks’. Proc. Vehicular Technology Conf. Fall, 2009, pp. 1–5
17 Pedreiras, P., Almeida, L.: ‘Message routing inmulti-segment FTT networks: the isochronous approach’. Proc. of Parallel and Distributed Processing Symp., 2004, pp. 122–129
18 Al Sheikh, A., Brun, O., Chéramy, M., *et al.*: ‘Optimal design of virtual links in AFDX networks’, *Real-Time Syst.*, 2013, **49**, (3), pp. 308–336
19 Tămaş-Selicean, D., Pop, P., Steiner, W.: ‘Design optimization of TTEthernet-based distributed real-time systems’, *Real-Time Syst.*, 2014, **51**, (1), pp. 1–35
20 Laursen, S.M., Pop, P., Steiner, W.: ‘Routing optimization of AVB streams in TSN networks’. SIGBED Review, 2016
21 Zhang, L., Goswami, D., Schneider, R., *et al.*: ‘Task-and network-level schedule co-synthesis of Ethernet-based time-triggered systems’. Proc. of Asia and South Pacific Design Automation Conf., 2014, pp. 119–124
22 Craciunas, S.S., Serna Oliver, R.: ‘Combined task-and network-level scheduling for distributed time-triggered systems’, *Real-Time Syst.*, 2016, **52**, (2), pp. 161–200
23 Steiner, W.: ‘An evaluation of SMT-based schedule synthesis for time-triggeredmulti-hop networks’. Proc. Real-Time Systems Symp., 2010, pp. 375–384
24 Steiner, W.: ‘Synthesis of static communication schedules for mixed-criticality systems’. Proc. Int. Symp. on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops, 2011, pp. 11–18
25 Pozo, F., Steiner, W., Rodríguez-Navas, G., *et al.*: ‘A decomposition approach for SMT-based schedule synthesis for time-triggered networks’. Proc. of Conf. on Emerging Technologies AND Factory Automation, 2015, pp. 1–8
26 Suethanuwong, E.: ‘Scheduling time-triggered traffic in TTEthernet systems’. Proc. of Conf. on Emerging Technologies Factory Automation, 2012, pp. 1–4
27 Craciunas, S.S., Serna Oliver, R., Chmelík, M.: ‘Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks’. Proc. of Int. Conf. on Real-Time Networks and Systems, 2016
28 Dürr, F., Nayak, N.G.: ‘No-wait packet scheduling for IEEE time-sensitive networks (TSN)’. Proc. Int. Conf. on Real-Time Networks and Systems, 2016
29 Pop, P., Eles, P., Peng, Z.: ‘Schedulability-driven frame packing for multicluster distributed embedded systems’, *ACM Trans. Embedded Comput. Syst.*, 2005, **4**, (1), pp. 112–140
30 Saket, R., Navet, N.: ‘Frame packing algorithms for automotive applications’, *J. Embedded Comput.*, 2006, **2**, (1), pp. 93–102
31 EtherCAT Technology Group, ‘ETG 1000 EtherCAT Specification’ (EtherCAT Technology Group, 2013, 1st edn.)
32 Ayed, H., Mifdaoui, A., Fraboul, C.: ‘Frame packing strategy within gateways for multicluster avionics embedded networks’. Proc. Emerging Technologies Factory Automation, 2012, pp. 1–8
33 Mikolasek, V., Ademaj, A., Racek, S.: ‘Segmentation of standard Ethernet messages in the Time-Triggered Ethernet’. Proc. Int. Conf. on Emerging Technologies and Factory Automation, 2008, pp. 392–399
34 Meyer, P., Steinbach, T., Korf, F., *et al.*: ‘Extending IEEE 802.1 AVB with timetriggered scheduling: a simulation study of the coexistence of synchronous and asynchronous traffic’. Proc. IEEE Vehicular Networking Conf., 2013, pp. 47–54
35 Alderisi, G., Patti, G., Bello, L.: ‘Introducing support for scheduled traffic over IEEE audio video bridging networks’. Proc. Emerging Technologies Factory Automation Conf., 2013, pp. 1–9
36 Scharbarg, J.-L., Ridouard, F., Fraboul, C.: ‘A probabilistic analysis of end-to-end delays on an AFDX avionic network’, *IEEE Trans. Ind. Inf.*, 2009, **5**, (1), pp. 38–49
37 Zhao, L., Xiong, H., Zheng, Z., *et al.*: ‘Improving worst-case latency analysis for rate-constrained traffic in the Time-Triggered Ethernet network’, *IEEE Commun. Lett.*, 2014, **18**, (11), pp. 1927–1930
38 Tămaş-Selicean, D., Pop, P., Steiner, W.: ‘Timing analysis of rate constrained traffic for the TTEthernet communication protocol’. Proc. Int. Symp. on Real-Time Distributed Computing, 2015, pp. 119–126
39 Diemer, J., Thiele, D., Ernst, R.: ‘Formal worst-case timing analysis of Ethernet topologies with strict-priority and AVB switching’. Proc. IEEE Intl. Symp. on Industrial Embedded Systems, 2012, pp. 1–10
40 Bordoloi, U., Aminifar, A., Eles, P., *et al.*: ‘Schedulability analysis of ethernet AVB switches’. Proc. Embedded and Real-Time Computing Systems and Applications Conf., 2014, pp. 1–10
41 De Azua, J.A.R., Boyer, M.: ‘Complete modelling of AVB in network calculus framework’. Proc. Int. Conf. on Real-Time Networks and Systems, Versaille, France, 2014, pp. 55–64
42 Garey, M.R., Johnson, D.S.: ‘Computers and intractability: a guide to the theory of NP-completeness’ (W. H. Freeman & Co., 1979, 1st edn.)
43 Valiant, L.G.: ‘The Complexity of enumeration and reliability problems’, *SIAM J. Comput.*, 1979, **8**, (3), pp. 410–421
44 CPLEX Optimizer. Available at https://www-01.ibm.com/software/commerce/optimization/cplex-optimize