# Incremental scheduling of the Time-Triggered traffic on TTEthernet network Draft extended version

Zdeněk Hanzálek and Jan Dvořák
Czech Institute of Informatics, Robotics and Cybernetics
Czech Technical University in Prague
Prague, Czech Republic
Email: Zdenek.Hanzalek@cvut.cz

***Abstract*—Complex systems are often developed incrementally when subsequent models must be backward compatible with the original ones. This requirement is relevant not only to particular components of the system, but also to the technology that interconnects them. The need to exchange high-volume data, for example, multimedia streams for infotainment in the avionic systems, together with safety-critical data, puts demands on both the high bandwidth and the deterministic behavior of the communication. TTEthernet is a protocol that has been developed to face these requirements while providing the generous bandwidth of Ethernet with up to 1 Gbit/s and enhancing its determinism by enabling the transmission of the time-triggered messages. However, the efficiency of the time-triggered communication depends on the schedule it follows. Thus, synthesizing a good schedule that meets all the real-time requirements and preserves the backward compatibility with the schedules of preceding models is essential for the performance of the whole system.**

**In this paper[1], we study the problem of designing periodic communication schedules for time-triggered traffic. The aim is to maximize the uninterrupted gap for the remaining non-real-time traffic. The provided scheduling algorithm, based on MILP and CP formulation, can obtain good schedules in a reasonable time while preserving the backward compatibility. The experimental results show that the time demands of the algorithm grows exponentially with the number of messages to be transmitted, but, even for industrial-sized instances with more than 2000 messages, the algorithm is able to return the close optimal schedules in the order of hundreds of seconds.**

## I. Introduction

The development process in many industrial fields, e.g., automotive or avionics, is based on incremental steps where new models are an evolution of the previous ones. This incremental process enables the cost-efficient development for companies and reduces the test effort. Moreover, the customer is guaranteed that the new model is an upgraded version of the model that he or she is comfortable with. These benefits are a side effect of the backward compatibility that should be assured among incremental development steps. Backward compatibility affects external systems, e.g., human-machine interface, as well as internal ones such as the communication subsystem. The backward compatibility in communication subsystems significantly reduces the costs spent on debugging, testing, and maintenance as newly developed Electronic Control Units (ECUs) and diagnostic tools can follow the agreement on the sharing of communication resources achieved in previous development steps.

The incremental development process is already ingrained in the industrial practice. However, there is an ongoing effort to develop and produce new models even more cost-efficiently. One possibility on how to reduce production costs in complex interconnected systems is to combine safety-related communication together with non-critical communication into one common medium [1]. Safety-related communication requires determinism, while non-critical communication demands a huge bandwidth without hard timing constraints. In the past, these two communication flows were conducted separately, as there were no communication protocols that could handle the requirements of both. However, modern protocols, like TTEthernet, were developed to bear such a difficult task.

In TTEthernet, safety-related communication is exchanged based on a periodic time-triggered communication schedule, while non-critical communication fills the empty gaps in the schedule. Such a communication schedule has to be designed in a way that all the real-time requirements are met to enable the reliable and deterministic operation of the application. The creation of the schedule involves additional complexity compared to the bus or passive star topologies of networks like FlexRay [2] or CAN because TTEthernet supports complex switched topologies [3]. These aspects, together with the real case problem proposed by our avionics industry partner, have motivated us to face the problem of scheduling time-triggered communication on the TTEthernet network while keeping the incremental development process in mind.

The paper presents the algorithm for creating schedules for time-triggered traffic on the TTEthernet network while maximizing the minimal guaranteed continuous gap for the traffic with lower criticality. The study aims to develop the periodic scheduling algorithm, which preserves the backward

[1]cite as: Z. Hanzalek; J. Dvorak: Incremental Scheduling of the Time-triggered Traffic on TTEthernet Network, In: 11th International Conference on Operations Research and Enterprise Systems - ICORES, 302-313, 2022.

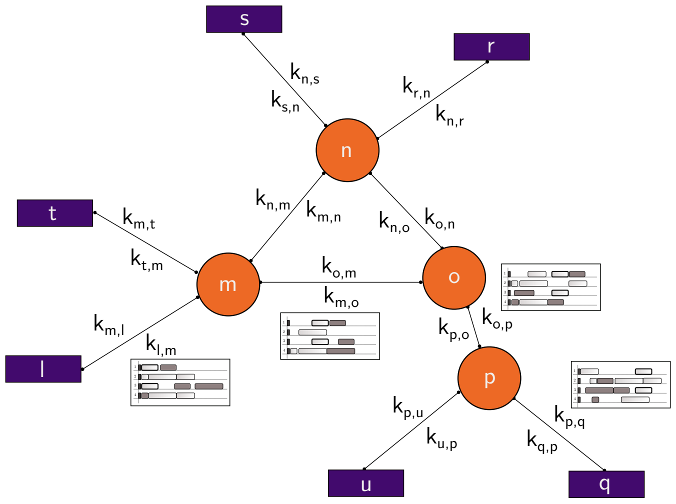

Fig. 1. An example of the TTEthernet network topology with the routing and scheduling of message $m_1$ from node l to node q

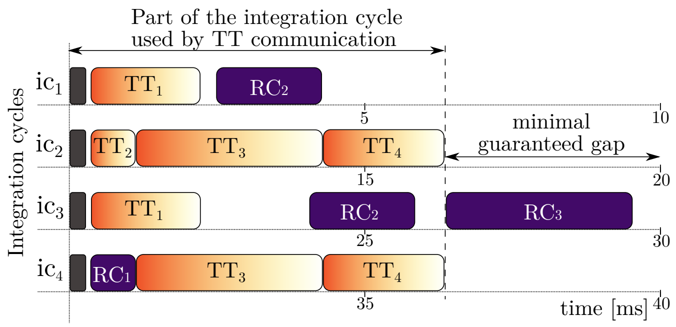

Fig. 2. The example of the communication on a link in one cluster cycle

compatibility with the original schedule. Finally, the influence of the backward compatibility on the communication schedule is analyzed, and the scalability of the proposed algorithm is presented.

### A. *TTEthernet Overview*

TTEthernet (TT stands for Time-Triggered) is an extension of Ethernet for deterministic communication developed as a joint project among the Vienna University of Technology [4], TTTech, and Honeywell, and standardized as SAE AS 6802 [5] in 2011. It operates at Level 2 of the ISO/OSI model, above the physical layer of Ethernet. It requires a switched network with full-duplex physical links, such as Automotive Ethernet standard 1000BASE-T1. An example of the TTEthernet topology is depicted in Fig. 1.

The global time in the system is assured by the clock synchronization protocol, where the clocks of all the interconnected ECUs are being synchronized periodically. Every synchronization period is called an *integration cycle*.

The traffic with various time-criticality is integrated into one physical network. There are three traffic classes in TTEthernet. These classes, ordered by decreasing priority, are Time-Triggered (TT), Rate-Constrained (RC) and Best-Effort (BE) traffic.

The TT traffic class has the highest priority. A jitter shorter than µs can be achieved on a physical layer (the physical layer jitter also depends on the connected network devices). The TT messages are periodic. We assume that they are strictly periodic (i.e., no jitter in application level is allowed) in agreement with [6]. The least common multiple of their period is called the *cluster cycle*.

For traffic with less strict timing requirements, the RC traffic class can be used. This traffic class conforms to the ARINC 664p7 specification [7] (also called AFDX). The RC traffic represents event-triggered communication, which does not follow any schedule known in advance.

A simple example of the TT traffic, together with the RC traffic on one direction of a physical link, is presented in Fig. 2. In the figure, the particular integration cycles are situated in rows, and the horizontal axis represents the time instants in the particular integration cycle. The length of the cluster cycle (40 ms) is equal to four times the length of the integration cycle (10 ms) here. The figure shows that messages $TT_1$, $TT_3$ and $TT_4$ have the same period (twice the duration of the integration cycle - i.e., 20 ms) and $TT_2$ has a period equal to four times the integration cycle length. The dark message at the beginning of each integration cycle is the synchronization message.

Standard Ethernet traffic can be transmitted through the network too. Such traffic is called the Best-Effort (BE) traffic and has the lowest priority.

When the TT traffic is used together with other traffic classes, a TT message could be delayed by another RC or BE message. The delay happens when a TT message arrives while an RC or BE message is in transmission. The Timely block integration policy, which causes no extra delay of the TT traffic, is used in this paper. In this case, an RC or BE message can only be transmitted if there is enough time for the transmission of the entire message before the next TT message is scheduled. If there is insufficient time, the transmission of the RC or BE message is postponed until after the TT message is transmitted. It additionally means that the TT traffic follows the schedule without any delays.

### B. *Related works*

The area of time-triggered communication scheduling on Ethernet-based networks has already been examined in many publications. Steiner [8] was among the first to study the problem. They described the basic constraints for scheduling the communication in the TTEthernet network and provided the Satisfiability Modulo Theories (SMT) formulation that was able to find a feasible schedule for small instances with up to 100 messages. The concept of schedule porosity was introduced in [9]. The porosity (the allocated blank slots for RC messages spread over the integration cycle) is introduced to the schedule to decrease the delay posed on RC traffic by T traffic. Thus, the porosity allows improving the communication delays while the complexity of the direct delay optimization [10] is tackled. To evaluate the impact of the porosity on the

RC traffic, Steiner et al. provided a pessimistic worst-case delay calculation, which was consequently tightened by a new method published by Tamas-Selicean et al. in [6]. A more detailed study of the impact of the time-triggered schedule on the RC communication has been presented in [11]. In [12], Tamas-Selicean et al. employed the TabuSearch algorithm to overcome the scalability problem of previous SMT formulations. As noted by [13], porosity scheduling has a disadvantage that gaps introduced at the beginning of the scheduling process do not consider the profile of the RC traffic. The concept of porosity is also weak in the case of scheduling TT messages with short periods. Wang et al. [14] used back-to-back schedule optimization, which aims to minimize the standard deviation of the messages offset in the integration cycle (hence, create as compact schedule as possible), to overcome the weakness of the porosity approach. The concept of minimization of the TT communication block length, called makespan minimization, was presented by Dvorak et al. in [15]. The paper formulated the scheduling problem as an RCPSP model to solve the problem efficiently. However, their method did not allow one to preserve the backward compatibility, and the quality of the resulting schedule was limited by the use of naive shortest-path-tree routing algorithm. Pozo et al. in [16] used a divide-and-conquer method to overcome the scheduling scalability limitations in large-scale hybrid networks considered to be used in, for example, smart cities in the future.

Based on the given TTEthernet communication schedule, Craciunas et al. scheduled the tasks on the communication endpoints in [17]. Furthermore, they presented a holistic scheduling algorithm that makes network-level schedules together with task-level schedules in [18]. Zhao et al. studied the problem of holistic security-aware scheduling in [19]. They used a modified TESLA authentication mechanism to protect the authenticity of the messages and provided MILP-formulation based scheduling algorithm.

The closely related problem to TTEthernet scheduling is the scheduling of TT communication for IEEE 802.1Qbv, which is the standard of the IEEE Time-Sensitive Networking group. Craciunas et al. derived the scheduling constraints for the TT communication on IEEE 802.1Qbv in [20] and provided an SMT model that aims to minimize the number of queues needed to schedule a given set of messages. Consequently, Zhao, together with Pop and Craciunas, provided the calculus for the Worst-case delays in [21]. Rottenstreich et al. [22] are using a greedy algorithm to find the shortest schedule for strictly periodic data streams and show that the greedy algorithm is able to find the optimal solution in special cases that often occur in practice.

All the published papers aim to create schedules from scratch, and none of them considers backward compatibility with the preceding systems, which limits the use of the proposed method in industries with an incremental development process.

### C. Contribution and paper outline

The main contributions of this paper are:

1) The formal description of the incremental TTEthernet scheduling problem with real-time constraints.
2) The three-stage heuristic algorithm, which includes
    - the routing algorithm that balances the communication load among the links
    - the message-to-integration cycle assignment algorithm that balances the communication load among the integration cycles
    - the message scheduling method based on the constraint programming model of the problem
3) An examination and discussion of the impact of the incremental aspect on TTEthernet scheduling.
4) An evaluation of the proposed algorithm from quality and performance point of view.

The paper is organized as follows: Section II describes the studied problem of the incremental TT message scheduling in the TTEthernet network comprehensively. In Section III, the proposed method of the schedule creation is described consisting of a message routing method, a load-balancing heuristic, and a CP based formulation of the scheduling problem. The method and the impact of the backward compatibility on the scheduling are evaluated and discussed in Section IV. Section V concludes the paper.

## II. Problem statement

This paper aims to design a method for finding feasible strictly periodic schedules for time-triggered communication on the TTEthernet network so that the maximal part of the remaining bandwidth can be preserved for the RC and BE messages, the timing constraints are satisfied, and the backward compatibility with the original schedule is preserved. All aspects of the tackled problem are described in this section.

### A. Messages

Each message $m_i$ from a set of the TT messages $M$ that is to be scheduled has the following parameters:

- $t_i$ - period
- $c_i$ - message length in the number of bits consisting of a payload, headers and interframe gap
- $d_i$ - deadline
- $r_i$ - release date
- $q_i$ - identifier of the transmitting node
- $Q_i$ - set of the receiving node identifiers (the set contains only one receiving node in the case of a unicast message)

The message period $t_i$ is assumed to be an integer multiple of the length of the integration cycle $ic$. The length of the resulting schedule is determined by the length of the cluster cycle $cc$. The cluster cycle consists of set of the integration cycles $I$. The transmission time of message $m_i$ has to be smaller than or equal to the duration of the integration cycle (it would not be possible to send a synchronization message otherwise), and its length $c_i$ does not exceed the maximal Ethernet frame length of 1530 bytes. Deadline $d_i$ and release date $r_i$ are assumed to have the value in the range $0 \leq r_i \leq d_i \leq t_i$.

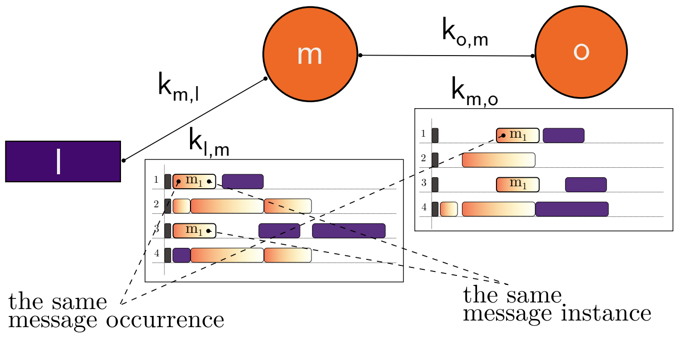

Fig. 3. Visualization of the difference between message occurrence and message instance

## B. Network topology

The TTEthernet topology consists of nodes and links which interconnect them. The nodes $e_i \in E$ are divided into two classes: *redistribution nodes* $E^R$ and *communication endpoints* $E^C$. The communication endpoints are nodes that generate or process the data (e.g., sensors, actuators, control units, and other ECUs). Thus, only the identifier of a communication endpoint can be assigned to message $m_i$ as transmitter $q_i$ or one of the receivers from set $Q_i$. The redistribution nodes, on the other side, are switches without any of their own data to transmit and serve as intermediary nodes for the communication. In Fig. 1, the communication endpoints are titled by "ECU", and the redistribution nodes have arrows drawn on the top side. The front side of each node is labeled by its name.

Each hop in the network introduces a technical delay caused by queuing in the ingress and egress port. Such a delay in a switch is represented by parameter $\tau$ for the TT messages. The value of $\tau$ can be in the range from 1 µs to 2.4 µs according to the network configuration [23].

Each link $k_{i,j}$ from a set of links $K$ connects two nodes $e_i$ and $e_j$. This connection covers just one direction of the full-duplex communication. Therefore, two links $k_{i,j}$ and $k_{j,i}$ model one full-duplex physical link between nodes $e_i$ and $e_j$. These two links are two independent resources from the scheduling point of view. The instance of message $m_i$ in link $k_{l,m}$ is called a *message instance* $m_i^{l,m}$. The set of all the message instances is denoted by $MI$. All the transmissions of some message $m_i$ in one particular link represent the same message instance. The *message occurrence*, on the other hand, represents all the transmissions of some message $m_i$ in one particular integration cycle. The difference between the message instance and the message occurrence is graphically explained in Fig. 3. The figure shows the detailed view on the sub-segment of the network topology from Fig. 1 with node $e_l$, $e_m$ and $e_o$ only. Both links of any physical link are labeled here already.

## C. Message routing

A sequence of the links $S_i^q = (k_{l,m}, k_{m,o}, ..., k_{p,q})$ represents the routing path of message $m_i$ from transmitter $q_i = e_l$ to receiver $e_q \in Q_i$ through the redistribution nodes $e_m, ..., e_p$. The union of all the routing paths $\cup S_i^q \;|\forall q \in Q_i$ for a given message $m_i$ determines the routing tree $S_i$. For example, the transmission of message $m_1$ through routing path $S_1^q$ is presented in Fig. 1. Only one direction of each physical link is labeled in the figure for the sake of simplicity. The routing paths $S_i^q$ are not known in advance. Therefore, finding the appropriate routing trees is part of the optimization process.

## D. Original schedule

Additionally, the original schedule is given for the incremental scheduling. The original schedule defines the start time of transmissions for the message instances $\tilde{m}_i^{k,l}$ from the subset of the message instances $\widetilde{MI} \subset MI$ which are already present in the original schedule. The original schedule can be determined for a subset of messages as well as for a subset of its message instances. This representation of the original schedule even allows the extension of the topology as well as the extension of the set of receivers for the already present messages. More precisely, all the modifications of the topology and message set that do not force the backward compatibility to be broken are allowed in the incremental development process.

## E. Schedule

The schedule is so-called strictly periodic, which means that the next message occurrence of message $m_i$ in a particular link appears in the schedule exactly $t_i$ time units after the current one. Therefore, the positions of all the message occurrences of message $m_i$ in the strictly periodic schedule can be deduced from the position of the first message occurrence and its periodicity.

A feasible schedule has to fulfill the following hard constraints:

**Completeness constraint:** Each message $m_i \in M$ has to be scheduled.

**Contention-free constraint:** Any link is capable of transferring at most one message at a time.

**Timing constraint:** Each message has to be transmitted after its release date and received by all the receivers before its deadline.

**Transmission compactness constraint:** The message transition from transmitting node $q_i$ to all the receivers from $Q_i$ has to be accomplished in one integration cycle.

**Backward compatibility constraint:** The start of transmission time must be preserved for all the message instances participating in the original schedule.

**Precedence constraint:** Message $m_i$ has to be scheduled in link $k_{m,o}$ at least $\tau$ time units after it is scheduled in $k_{l,m}$ if $k_{l,m}$ precedes $k_{m,o}$ in $S_i^q$.

## F. Objective

The coherent TT traffic segment should be compressed as much as possible to preserve the maximum part of the remaining bandwidth for the RC and BE traffic. This idea follows the practice from the FlexRay bus or Profinet, where the dedicated communication segment is allocated for the TT traffic. The TT traffic can be scheduled at the beginning of the integration cycle, and the remaining coherent gap in the integration cycle without the TT traffic is preserved for the RC and BE traffic. The gap, which is the shortest among all the links, is denoted as a *minimal guaranteed gap* (see Fig. 2). Considering the constraints and aspects above, the goal of the scheduling is to find a feasible schedule for the TT messages, which maximizes the minimal guaranteed gap.

## III. Algorithm

The described problem is extensively complex as it involves scheduling together with routing. The algorithm that would solve the whole problem at once would put extreme demands on the computational resources or time needed to find the solution. Thus, the incremental scheduling problem proposed in the paper is decomposed into three subproblems to tackle the computational effort. The solution of each subproblem fixes some decisions for the subsequent problem. Hence, the incremental scheduling algorithm can be considered as being divided into three stages. In the first stage (Sec. III-A), the routing of the messages is established. In the second stage (Sec. III-B), the algorithm finds the assignment of the messages to the particular integration cycles. The transmission times for each message in each link are decided in the last stage (Sec. III-C).

### A. Messages routing problem

The network topology is often a tree in industrial networks. It means that there are no cycles and, therefore, only one possible path from a communication endpoint to any other endpoint exists. Thus, the determination of the routing is trivial in such a case. However, the TTEthernet does not restrict the network topology to the tree. The cycles introduce new redundant paths for messages that can serve as a backup during a partial network malfunction. Moreover, the appropriate selection of the routing path of the message can balance the load among the network. Thus, redundant paths can remove the bottleneck of the tree topology. However, the TT messages have to know which path they are routed through in advance.

$$
\begin{array}{lll}
\min & F & \\
\text{s.t.} & f_{l,m} \leq F & \forall k_{l,m} \in K \\
& \sum_i c_i^{l,m} \cdot \frac{cc}{t_i} \cdot x_{i,l,m} = f_{l,m} & \forall k_{l,m} \in K \\
& \sum_l x_{i,l,m} = 1 & \forall m_i \in M;\ \forall m \in Q_i \\
& \sum_m x_{i,l,m} \geq 1 & \forall m_i \in M;\ l = q_i \\
& \sum_l x_{i,l,m} \leq \sum_k x_{i,m,k} & \forall m_i \in M;\ \forall m \in E^R \\
& \sum_l x_{i,l,m} \leq 1 & \forall m_i \in M;\ \forall m \in E^R \\
& \sum_l x_{i,l,m} \geq \frac{\sum_k x_{i,m,k}}{deg^-(m)} & \forall m_i \in M;\ \forall m \in E^R \\
& s_{i,m} \geq 1 + s_{i,l} - B + Bx_{i,l,m} & \forall m_i \in M;\ \forall k_{l,m} \in K \\
& s_{i,m} \leq 1 + s_{i,l} + B - Bx_{i,l,m} & \forall m_i \in M;\ \forall k_{l,m} \in K \\
& s_{i,l} = 0 & \forall m_i \in M;\ l = q_i \\
& x_{i,l,m} = 1 & \forall m_i^{l,m} \in \widetilde{MI} \\
& x_{i,l,m} \in \{0,1\} & \forall m_i \in M;\ \forall l,m \in E \\
& f_{l,m} \in \mathbb{Z}_0^+ & \forall l,m \in E \\
& F \in \mathbb{R} & \\
& s_{i,l} \in \mathbb{Z}_0^+ & \forall m_i \in M;\ \forall l \in E
\end{array} \tag{1}
$$

Therefore, the first stage of the algorithm finds the routing. In accordance with the claim above, the algorithm aims to find such a routing that the network load is as balanced among the links as possible. The balanced network gives a good premise that the resulting communication schedules will be shorter than in the case of an unbalanced network. Thus, this routing objective corresponds to the aims of the scheduling algorithm.

An MILP model is used to solve the routing subproblem and decide routing tree $S_i$ for each message.

The binary variable $x_{i,l,m}$ decides whether the message $m_i$ is routed through the link $k_{l,m}$, and variable $f_{i,m}$ represents the load of the link $k_{l,m}$. The real variable $F$ is, consequently, the load of the busiest link. The auxiliary variable $s_{i,l}$ assigns a numerical label to each node $e_l$ for each message $m_i$. The label determines the depth of the node $e_l$ in the routing tree $S_i$. The artificial constant $B$ represents any number that is bigger than the maximal depth ($\max s_{i,l}$). Parameter $c_i^{l,m}$ represents the transmission time of message $m_i$ in link $l_{l,m}$. Note, that if the links are configured to have a different bandwidth, then the transmission time of the same message varies among the links.

The objective of the MILP model minimizes the load of the busiest link. The first constraint, together with the objective, ensures that the value of $F$ equals the load of the busiest link. The second constraint calculats the load $f_{l,m}$ for each link. The third and fourth constraints force the routing path for each message to also contain the receiving nodes and the transmitting node. The fifth, sixth, and seventh constraint assure that the redistribution nodes serve as the inner nodes of the routing tree. The eighth, ninth and tenth constraints guarantee the routing tree of any message not to contain the cycle. $B$ represents any constant that is larger than the number of nodes in the topology here. Its aim is to make the model ignore the constraints if the value of $x_{i,l,m}$ is equal to zero. The eleventh constraint forces the resulting routing to satisfy the backward compatibility.

The routing tree $S_i$ defines the set of links in which the message is to be scheduled and specifies the precedence relations among the message instances.

### B. Integration cycle assignment problem

To distribute the messages among the integration cycles, we used an idea from the multiprocessor scheduling area. In the area, if all the workload of the tasks is distributed among the processors evenly, then the part of the integration cycle used by the TT communication has a good chance to be minimal. Following that, the algorithm tries to distribute the messages among the integration cycles evenly. All the precedence constraints, the time lags imposed by the switch delay $\tau$, and the real-time constraints are relaxed here. The integration cycle assignment problem is formulated as the following MILP model 2.

The binary variable $a_{i,j} = 1$ iff message $m_i$ is assigned to the integration cycle $j \in \{0 \ldots t_i\}$. Similarly, the binary parameter $\tilde{a}_{i,j} = 1$ iff message $m_i$ was scheduled to the integration cycle $j \in \{0 \ldots t_i\}$ in the original schedule. The first constraint assures that the first message occurrence appears in exactly one of the possible integration cycles. Thus, it satisfies the completeness constraint. The second constraint makes the variable $z$ have the value equal to or greater than the time needed to exchange all the messages in any integration cycle of any link in the network. The constraint is evaluated for each link and each integration cycle in the cluster cycle so that the transmission times of all the message occurrences assigned to the particular integration cycle in the given link are summed up. The resulting total time must be less than or equal to variable $z$. The aim of the MILP model is to find such an assignment that minimizes $z$. Thus, the maximal time needed for the message exchange among all the resources is minimized. The third and fourth constraint force the messages to be assigned to the integration cycle, which can satisfy the release date and deadline constraints. The last constraint forces the messages from the original schedule to be assigned to the corresponding integration cycle in the new schedule.

$$
\begin{array}{lll}
\min\limits_{a_{i,j}} & z & \\
\text{s.t.} & \sum\limits_{j} a_{i,j} = 1 & \forall i \in M \\
& \sum\limits_{m_i \in k_{l,m}} c_i^{l,m} \cdot a_{i,j \bmod t_i} \leq z & \forall j,l,m \mid j \in \{0 \ldots \frac{cc}{ic}\} \\
& a_{i,j} = 0 & \forall i,j \mid d_i < j \cdot ic \\
& a_{i,j} = 0 & \forall i,j \mid r_i > (j+1) \cdot ic \\
& a_{i,j} = 1 & \forall i,j \mid \tilde{a}_{i,j} = 1 \\
& a_{i,j} \in \{0,1\};\ z \in R & \forall i,j
\end{array} \qquad (2)
$$

The resulting assignment balances the load among the integration cycles, follows the routing of the messages, and preserves the timing and backward compatibility constraints.

### C. Link schedules creation problem

The constraint programming model is employed to create the resulting schedule. For the description of the model, the IBM CP Optimizer formalism [24] will be used. The CP model is based on so-called interval variables which, in our case, represent each message instance $m_i^{l,k} \subset MI$ in the schedule. The set of message instances to be scheduled on a particular link is known since the routing of the messages has been already decided. For each interval variable, the solver decides its start time. In the model, the time is considered as a relative offset to the start time of the integration cycle. Thus, two message instances that are scheduled with the same offset in the integration cycle, but with a different integration cycle are considered as being scheduled at the same time.

The objective of the scheduling is to minimize the part of the integration cycle used by the TT communication:

$$\min \max_{i,l,m} \text{endOf}(m_i^{l,m})$$

The length of the message is preserved by:

$$\text{lengthOf}(m_i^{l,m}) = c_i^{l,m} \quad | \forall i \in M; l,m \in E$$

Further constraints have to be introduced to satisfy the timing constraints. Due to the known message instance to the integration cycle assignment, the release date $\hat{r}_i$ and deadline $\hat{d}_i$ relative to the integration cycle in which message $m_i$ is transmitted are also known. Thus, the timing constraints can be defined as the start time limitation of the related interval variable:

$$\text{startMin}(m_i^{l,m}) = \hat{r}_i \quad | \forall m_i^{l,m} \in MI$$
$$\text{endMax}(m_i^{l,m}) = \hat{d}_i \quad | \forall m_i^{l,m} \in MI$$

Similarly, the backward compatibility is assured to be satisfied by:

$$\text{startOf}(m_i^{l,m}) = \text{startOf}(\tilde{m}_i^{l,m}) \quad | \forall m_i^{l,m} \in \widetilde{MI}$$

where $\text{startOf}(\tilde{m}_i^{l,m})$ denotes the offset of the message instance in the original schedule.

The contention-free constraint is necessary to be satisfied next. From the model point of view it means that no two message instances, which appear in the same integration cycle and on the common link, can overlap. As the assignment of the message instances to the links (routing) and integration cycles is already decided, it can be trivially deduced in which link and integration cycle message $m_i$ appears. Let us denote $MI_i^{l,m}$ the set of message instances which appear in the same integration cycle $ic_i$ and link $k_{l,m}$. Now, the contention-free constraint is stated as:

$$\text{noOverlap}(MI_i^{l,m}) \quad | \forall i \in I; l, m \in K$$

where noOverlap is a CP operator that keeps all the interval variables in the given set to be scheduled in distinct time intervals.

Finally, it is necessary to keep the precedences among message instances. In this case, the precedence constraints are given by the routing of the message $S_i$ and by the technical delay caused by switching the logic in the redistribution nodes. Let $P_i^{l,m}$ be the set of predecessors of the message instance $m_i^{l,m}$ in $S_i$. The message instance $m_i^{k,l}$ is part of the $P_i^{l,m}$ if and only if $x_{i,k,l} = 1$ (see the MILP model for the routing). Consequently, the precedence constraint is formulated as:

$$\text{endBeforeStart}(p, m_i^{l,m}, \tau) \quad | \forall p \in P_i^{l,m}; \forall i \in M; l, m \in K$$

With these constraints, the CP model for the message scheduling is defined completely.

## IV. Experimental results

The proposed scheduling method was tested on a PC with Intel ® Core$^{\text{TM}}$ i7-4610M CPU (two cores with 3 GHz and hyper-threading) and 32 GB RAM. The algorithm uses the Gurobi ILP Solver for determining the messages routing and for solving the Integration cycle assignment problem. The Link schedules creation problem was solved by the IBM CP Optimizer. The time to solve a benchmark instance was limited to $5\,\text{min}$.

The benchmark instances used in this study were synthetically generated. The benchmark generator defines the topology first. In this study, a random graph topology is generally used. Consequently, the message parameters are generated randomly, considering the imposed limitations. These imposed limitations are described individually in detail for each test in the following sections. Each message is assigned to either the broadcast, multicast, or unicast group. Finally, the set of receivers is generated for each message according to the group.

The generation of the incremental scheduling benchmark instances was performed in reversed order, i.e., the benchmark instance for the last incremental iteration was generated first. Then the original instances for the incremental scheduling are made. For example, to generate an instance for the penultimate incremental iteration, the last instance is taken, and some of the messages, nodes, and links are removed from the instance according to the given pruning ratio.

To provide statistically significant results, thirty instances of each benchmark set were generated, and the presented values represent the mean value from all these instances.

### A. *Evaluation of the routing algorithm*

The proposed algorithm uses the message routing that aims to support the scheduling objective by the uniform scattering of the messages among the links. Thus, it unloads the communication on the most utilized links. To evaluate the algorithm's performance, the proposed routing method is compared to the Shortest path tree (SPT) routing method. The SPT routing method minimizes the number of hops the message needs to take to get from the transmitter to the receivers. It optimizes the overall bandwidth utilized by the communication on the network, but it does not prevent the bottlenecks caused by the individual overutilized links.

The comparison of both methods is presented in Fig. 4. For the evaluation, the complete scheduling algorithm was executed while the MILP or SPT method was used for the routing.

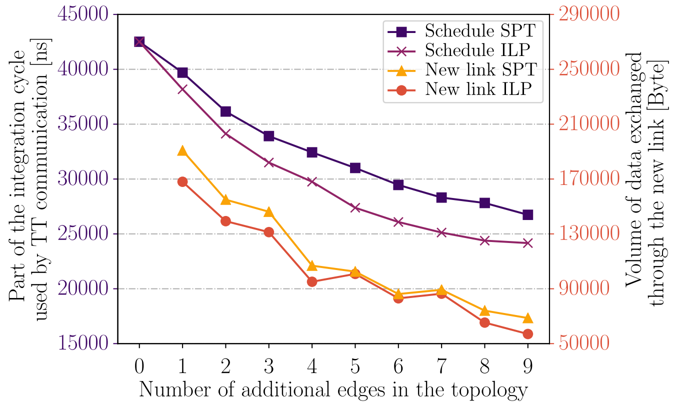

Fig. 4. The evaluation of the routing quality

Benchmark sets with 200 TT messages generated with periods in a range from one to three integration cycles and with Ethernet frame lengths in full range (i.e., up to 1500 bytes) were used to test the routing algorithm.

Fig. 4 presents how the method is able to utilize the redundant links in the graph. The x-axis denotes the number of redundant links added to the tree topology. The left y-axis, related to the Schedule SPT and Schedule ILP lines, represents the duration of the part of the integration cycle used for the TT communication and the right y-axis, related to the New link ILP and New link SPT line, represents the volume of the data exchanged through the newly introduced links.

As can be observed from "Schedule SPT" and "Schedule ILP" lines in Fig. 4, adding nine additional edges to the tree topology can shorten the duration of the part of the incremental cycle used by the TT communication to almost 50%. However, the benefits of the redundant links is that it can utilize the proposed method based on the MILP model (labeled as "Schedule ILP" in the figure) better than the method based on the SPT algorithm. On the tree topology, both methods behave equally as the routing tree is already decided by the topology. However, as the number of additional links is increasing, the proposed method acts significantly better than the SPT method.

The "New link" lines, on the other hand, shows how the significance of the newly introduced links evolves with the number of additional edges in the topology. The routing algorithms are able to forward through the last added message only about one-third of the data volume compared to the data volume it was able to forward through the first added link.

## *B. Impact of the incremental scheduling on the schedule*

The backward compatibility constraint introduced to the scheduling problem causes the overhead in the resulting schedule. To measure the overhead of the incremental scheduling over the non-incremental scheduling, another experiment was performed. The new set of benchmark instances was generated with ten incremental iterations. In the case of the incremental scheduling, the resulting schedule from the previous iteration was used as the original schedule. The first incremental scheduling iteration has an empty original schedule. All the messages were generated with a period in a range from one to three integration cycles and with Ethernet frame lengths in full range. The last incremental iteration instances contained 500 TT messages.

The results from the experiment are presented in Fig. 5. The incremental scheduling iteration is situated on the x-

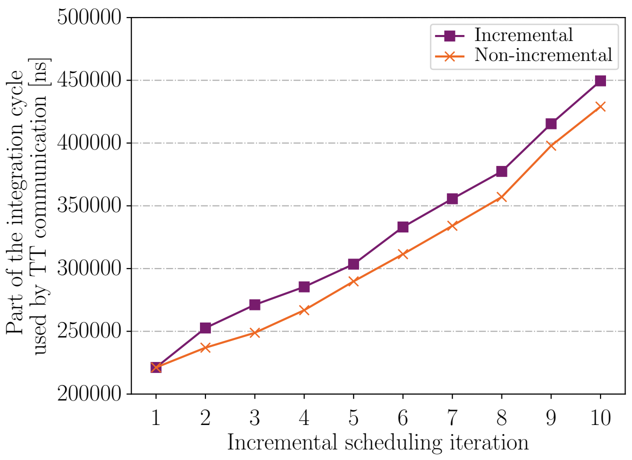

Fig. 5. The difference between the incremental and non-incremental scheduling

axis of the graph, and the schedule duration of the part of the integration cycle used by TT communication is on the y-axis. The dark purple row represents the result from the algorithm where the original schedule is considered, while the orange row represents the results of the algorithm, assuming no original schedule is given.

The result for the first incremental iteration is the same for incremental and non-incremental scheduling as no original schedule is used and, consequently, no backward compatibility constraint is applied in both cases. The most notable change in the difference between the incremental and non-incremental scheduling is present in the second incremental iteration. The prolongation caused by new messages in the case of the incremental scheduling is almost twice as much compared to the prolongation caused by the new messages in the case of the non-incremental scheduling. This causes the fact that the messages that were already scheduled in the first incremental iteration cannot be moved in the incremental scheduling. Thus, the new messages cannot be incorporated in such an efficient way as in the case of the non-incremental scheduling. However, this overhead also introduces a new porosity to the schedule. The further incremental scheduling iterations are able to use this porosity to place the new signals into those gaps efficiently. That is the reason why the overhead stays almost constant in the future scheduling iterations, and the difference between the incremental and non-incremental schedule is almost the same as can be observed in the graph.

## *C. Evolution of the schedule utilization over incremental iterations*

To support the statement from the previous section, the way how the utilization of the schedule evolves with the incremental iterations has been studied more deeply. For the purpose of this section, the utilization of the schedule (or just *utilization*) is defined as the portion of the part of the integration cycle used for the TT communication that is utilized by the message transmission averaged over all the integration cycles. In other words, considering only the part of the integration cycle used by the TT communication, the utilization represents the portion of the time when the link is busy. In order to test the utilization, similar benchmark instances were generated to the testing of the impact of the incremental scheduling. The only exception is the topology of the network. To avoid the impact of the routing on the porosity test, the generated instances use a tree topology. Two different measurements were performed.

Firstly, the average of the schedule utilization over the whole network (all the links) has been measured. The resulting graph can be seen in Fig. 6. The figure presents the evolution for two

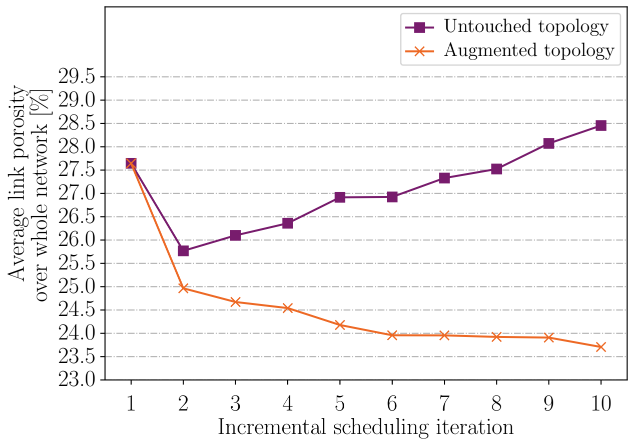

Fig. 6. The average utilization of the communication over the whole topology

cases: the dark purple line shows how the utilization evolves when the topology is not extended during the incremental iterations; the orange line shows the utilization evolution when the topology grows together with the number of messages. The x-axis represents the incremental scheduling iteration, and the y-axis represents the average utilization of the schedule in percent. This unutilized/free space can be used by the new messages in the future incremental scheduling iteration that are local (their transmitter and receivers are close to each other according to the network topology). Note that less than 30 % of the schedule is used according to the graph. This small amount is caused by the tree topology, where the links close to the leaves of the topology are rarely used. The figure also shows that the overall schedule utilization increases if there is no topology extension which is caused by adding new messages to the schedule and the significant part of them are local messages (thus, the part of the integration cycle used by the TT communication is not prolonged too significantly). If the topology is extended, the new sparse links introduce a lot of unutilized bandwidth to the schedule, which causes a decrease in the overall utilization.

Opposed to the new local messages that, as can be observed from Fig. 6, can be easily incorporated into the schedule without prolongation of the part of the integration cycle used by TT communication because of the low utilization of the links close to leaves, the new messages that need to transit the root node could cause the prolongation of the part easily. To study how the schedule utilization can impact the scheduling of these messages, the second measurement was performed where the utilization was calculated only on the most utilized link.

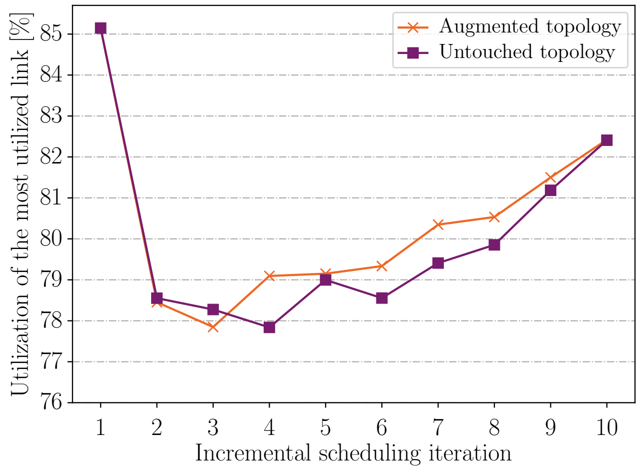

Fig. 7. The porosity of the most utilized link

The utilization of the most utilized link is presented in Fig. 7. Here, the utilization is about 77 to 86 %, depending on the particular incremental scheduling iteration. This makes it harder to incorporate the long-distance messages efficiently into the schedule without prolongation of the part of the integration cycle used by the TT communication. The figure also shows that there is no significant difference for the long-distance messages, whether the topology is extended (but no cycles introduced) or not as the topology does not influence the flow of the messages over the most utilized link.

These graphs also support the statement from the last section as both the utilization metrics are the worst in the first incremental scheduling iteration, making the second scheduling iteration the most difficult. After that, the utilization metric changes slowly and, thus, the difference between the efficiency of the incremental and non-incremental scheduling is similar.

## *D. Scalability of the scheduling algorithm*

The previous experiments aimed to study the quality and the impact of the incremental scheduling on the resulting schedule. However, the scalability of the algorithm and its computational complexity needs to be examined next. The scalability of the algorithm is strongly dependent on three factors - the lengths of the messages, the distinct message periods, and, finally, the number of messages in the instance. The impact of all of these factors on the algorithm's runtime will be studied in the following subsections.

### D.1 The scalability of the algorithm based on the message length

Firstly, the influence of the maximal allowed message length is presented. For this experiment, the instances with the maximal allowed message lengths from 1 byte to 1500 bytes of the payload were generated. To ensure that the finding of the optimal schedule and proving that the found solution is optimal will be accomplished in twenty minutes (the time limit for the computation of one benchmark instance is extended for all the scalability tests), the instances contained just 50 messages. The period of the generated messages is randomly chosen from one, two, four, or eight times the duration of the integration cycle. The results of the test are presented in Fig. 8.

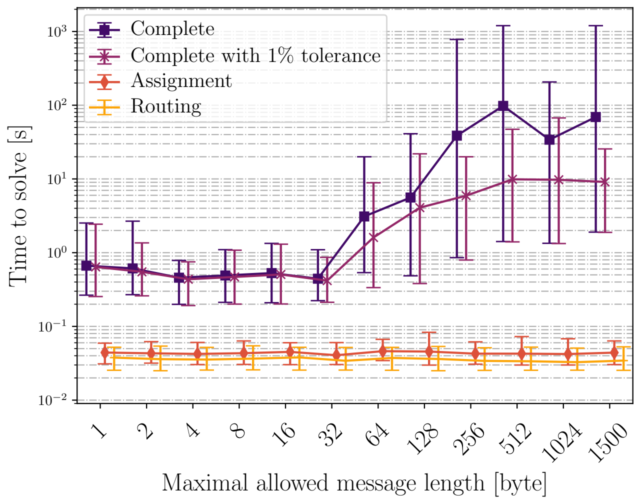

Fig. 8. The scalability of the scheduling method according to used message lengths

The x-axis of the graph represents the maximal allowed message length while the y-axis represents the time needed to solve the given instance set in the logarithmic scale. The orange line presents the time needed for the routing, the red line presents the time needed for finding the assignment of the messages to the incremental cycles, the magenta line represents the time needed to find a solution and to prove that its value is at maximum 1 % off the optimum, and the dark purple line is the total time needed for the creation of the optimal schedule. Note that the error bars represent the range in which the results for the particular benchmark set were acquired (the lower cap is minimal obtained value, and the upper cap is maximal obtained value) rather than the standard deviation because the standard deviation was often larger than the mean (and it is not possible to depict them in a logarithmic scale). The position of the markers was slightly adjusted for each line of the graph, even if they represent the same value on the x-axis, to make the reading of the error bars easier.

The graph shows that the major part of the time is consumed by the link scheduling algorithm. The time complexity of the routing algorithm and also the messages to the integration cycle assignment algorithm does not depend on the message lengths at all. On the other hand, the time demands of the link scheduling algorithm are almost constant for messages with lengths up to 32 bytes, and then it grows exponentially. Also, the time range for solving the instances was much wider in the case of instances with a payload larger than 64 bytes. The variance in time demands for those instances is caused by the uncertain difficulty of proving that the current scheduling solution is optimal. It can also be read, from the figure, that the tolerance of 1 % can reduce the scheduling time by one order in the case of messages with a full variety of their lengths.

### D.2 The scalability of the algorithm based on the message periods

The second experiment evaluates the influence of the time demands on the allowed set of message periods. Two different scenarios are commonly used in practice - messages with arbitrary periods and messages with harmonic periods. The messages with arbitrary periods can have periods equal to any integer multiple of the integration cycle, while, in the case of harmonic periods, the messages can have periods only equal to any duration of the integration cycle multiplied by a power of two. The harmonic periods ensure the shortest possible cluster cycle, which can grow fast in the case of arbitrary periods because the cluster cycle is equal to the least common multiple of all the possible periods. To examine the behavior of the algorithm in both scenarios, two separate tests were performed.

Each benchmark instance contains 50 messages with up to eight bytes of payload. The general graph topology has been used. The results for the messages with the arbitrary periods are presented in Fig. 9, and the results for the messages with the harmonic periods are shown in Fig. 10. The x-axis of the graphs represents the maximal allowed message period used.

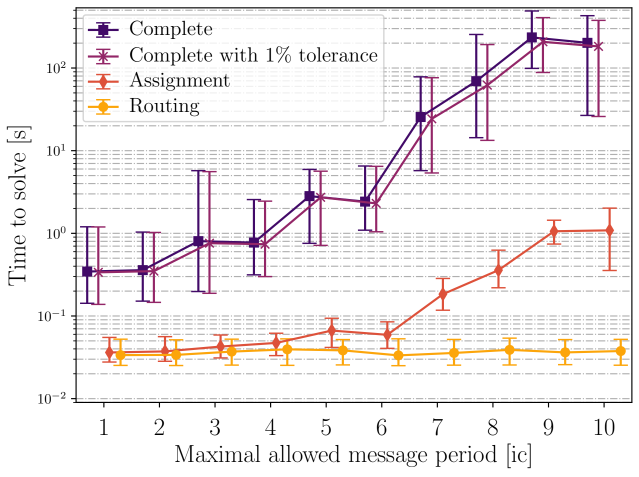

Fig. 9. The scalability of the scheduling method according to the used message periods

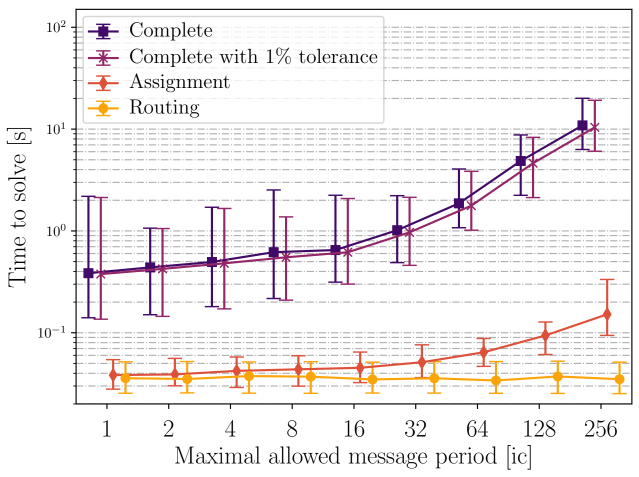

Fig. 10. The scalability of the scheduling method according to the used harmonic periods

The routing algorithm is not dependent on the set of used periods, and its computing demands stay low during the whole evaluation. The assignment algorithm is affected by the number of the used message periods because the messages with longer periods have more possible integration cycles to be assigned to. The computational complexity of the link scheduling algorithm grows exponentially with an increasing set of possible periods. However, the computational demands grow much steeper in the case of arbitrary periods. It is caused by the difficulty in the scheduling of the messages whose periods are co-prime. This statement is also supported by the observation that the computational complexity grows most significantly if a new prime period is introduced to the instance (specifically, if the messages with periods $5 \cdot ic$ and $7 \cdot ic$ are introduced in our case). The second reason is that the long cluster cycle (e.g., $2520 \cdot ic$ for the case with a maximal allowed period equal to $10 \cdot ic$) means a significant increase in the scheduling model variable domains for the constraint programming optimization. The computation complexity of the algorithm is considerably much lower if the periods are harmonic. In this case, the complexity growth is mainly cause by the prolongation of the cluster cycle, which is not significant as in the case of the instances with the arbitrary periods because the cluster cycle is only as long as the longest period here. The graphs also show that the benevolence of the loss of 1 % in the optimality does not help to reduce the time demands much here.

**D.3 The scalability of the algorithm based on the message counts**

The most important thing is to know how the complexity grows with the increasing number of messages, which is the subject of the third scalability experiment. The benchmark instances with messages containing up to 8 bytes of payload and with a maximal allowed period equal to $8 \cdot ic$ were generated. For these benchmark instances, harmonic periods were used, and the topology of the network corresponds to a general graph. The results of the experiment are depicted in Fig. 11. The x-axis represents the number of messages used in the instance.

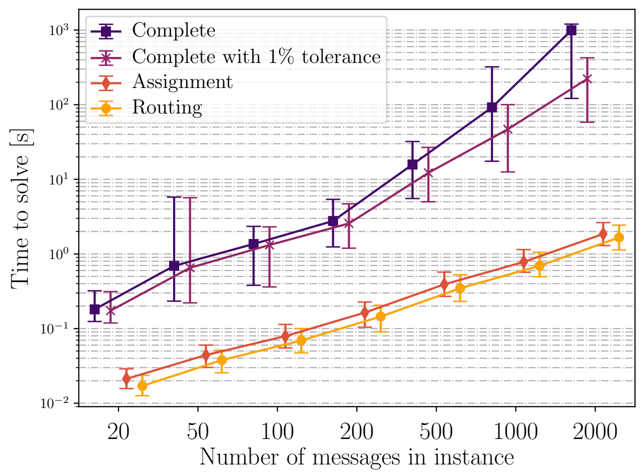

Fig. 11. The scalability of the scheduling method according to the number of messages

The results show that the routing algorithm and assignment algorithm are affected by the number of messages in a similar way as the link scheduling algorithm. The narrow range of the results (with the exception of instance set with 50 messages which is affected by outliers) shows that the increase in the number of messages stably influences the computational complexity. However, the complexity grows exponentially in the number of messages. While scheduling links with instances containing 100 messages to optimality took 1 s, the instances with 2000 messages needed almost 20 min. For bigger instances, the result with a proven distance of the link scheduling objective function value not further than 1 % from the optimal value can be obtained faster by an order even if this tolerance needs to be proven by the solver.

## *E. Evolution of the scheduling objective function in time*

It is necessary to note that the optimal solution for the routing algorithm, together with the optimal solution for the link scheduling algorithm, does not ensure the optimal solution from the problem statement point of view. Thus, in practical cases, it is not needed to wait until the link scheduling algorithm proves that the current solution is optimal (or close to optimum in the case of the 1 % tolerance), and the search can be stopped sooner. This allows for creating schedules for bigger industrial size instances in a reasonable time. To show how the duration of the part of the incremental cycle used by the TT communication evolves in time, the same benchmark instance set with 2000 messages as in Section IV-D.3 was used. The evolution of the duration of the part of the incremental cycle used by the TT communication during the link schedule creation is presented in Fig. 12.

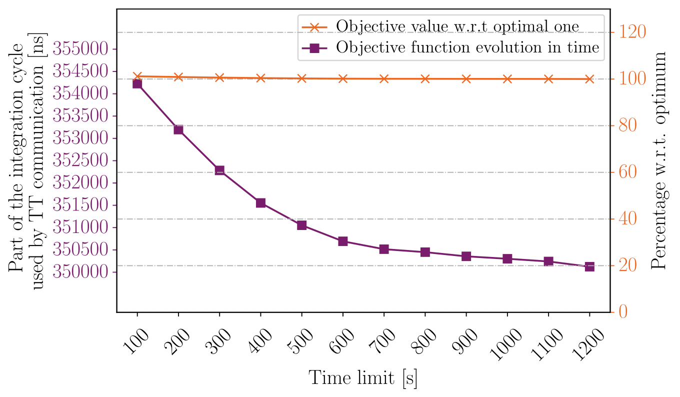

Fig. 12. The evolution of the duration of the part of the incremental cycle used by the TT communication in the time domain

The x-axis of the figure represents the time limit for the scheduling. At the time of 1200 s, all the instances in the set were solved to optimality. The left y-axis represents the average duration of the part of the integration cycle used by the TT communication over the whole benchmark set. The left y-axis represents the ratio of the obtained objective value at a particular time compared to the optimal one in percentage. Until time 100 s, the scheduling algorithm was not able to find a feasible solution for two instances out of thirty. At time 200 s, there was only one instance without finding a feasible solution. In all further times, all the instances have found a solution. The major change in the duration occurs in the beginning, and the slope is decreasing in time. Even if there is some improvement in the average duration of the part of the integration cycle used by the TT communication during the whole scheduling period, the improvement between the average duration obtained at time 100 s and 1200 s is less than 1.3 %. Thus, it is sufficient to use a solution, which is not necessarily optimal from the link schedules creation problem point of view but obtained in a shorter time, in many practical cases.

## *F. Scheduling of the real industrial instances*

The work has been motivated by our industrial partner, who develops electronic systems for the avionic industry. This section describes the behavior of the proposed algorithm on the real instance obtained from the partner. The instance contains 1922 messages (407 unicast messages and 1515 multicast messages) with a payload of up to 1036 bytes and periods from the set $\{12.5\,\mathrm{ms}, 25\,\mathrm{ms}, 50\,\mathrm{ms}, 100\,\mathrm{ms}, 200\,\mathrm{ms}, 1000\,\mathrm{ms}\}$. The system consists of 38 nodes. The topology is based on three switches SW1, SW2 and SW3 that are mutually interconnected. Each such switch is in the topology twice called SW1_A and SW1_B, and these two instances are interconnected too. Thus, the backbone of the network is based on such a double triangle topology. Each endpoint is connected to one of those switches. The schedule for this instance, which is optimal from all the subproblem's point of view, was obtained after less than $250\,\mathrm{s}$. The schedule utilization of the most utilized link reached 96.3% while the schedule utilization averaged over the whole network was only 19.6%. This shows that the integration cycle assignment was able to distribute the messages in the most utilized link very efficiently. Figure 13 presents how the payload is distributed among the links in the network. The links are classified according to the volume of

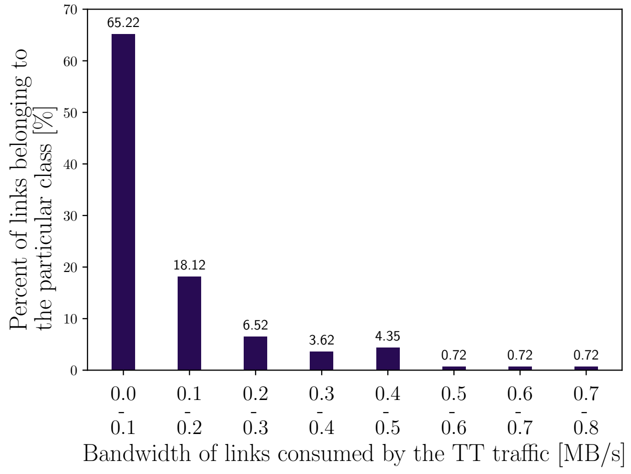

Fig. 13. The histogram of the links utilization

the TT data transmitted through them per second. The first class includes links with the volume from 0 to 0.1 MB/s, the second one includes the links with the volume from 0.1 to 0.2 MB/s, etc. The y-axis then represents the percentage of the links that belong to the particular class. The histogram shows that almost 70 % of the links have very low utilization of the links. About 50 % of these links are empty. That means that these links are connected to the endpoints that serve as a transmitter or a receiver (note that each transmission direction of the Full-duplex physical link is represented by two separate links here). Moreover, there is just one link in the class with 0.7 - 0.8 MB/s. This link (and also the links from classes with 0.5-0.6 MB/s and 0.6-0.7 MB/s) connects the redistribution node with the communication endpoint. Thus, the routing algorithm was not able to reroute some messages from this link through another path to lighten the link. The backbone links belong to the 0.2-0.3 MB/s and 0.3-0.4 MB/s classes. The interesting observation is that, considering the 100 MB/s TTEthernet network, the TT communication utilizes less than 1 % of the bandwidth in this instance. The rest of the bandwidth is used for the RC and BE communication.

## V. Conclusion

The paper focuses on the problem of the incremental time-triggered communication scheduling on the TTEthernet network, and studies the influence of introduced backward compatibility on the resulting schedules.

The incorporation of the event-triggered Rate-Constrained traffic with the Time-Triggered traffic is very important for mixed-criticality applications [25]. Event-triggered communication is usually used in industrial applications nowadays. However, the pressure placed on, for example, the automotive or avionics industries to verify and certify its systems on a component and system integration level pushes system developers to use time-triggered traffic for safety-related communication as its behavior is deterministic. However, the creation of schedules for time-triggered communication is a challenging task.

We have followed the idea of separating the time-triggered traffic and event-triggered traffic already used in [15], which was inspired by the scheme of the FlexRay bus communication cycle. The objective has been to maximize the minimal guaranteed coherent gap left in each integration cycle on each link that can be continuously used by the Rate-Constrained and Best-Effort traffic while keeping the backward compatibility with the original schedules created in the previous development iterations. The problem has been decomposed into three subproblems - (i) message routing, (ii) deciding in which integration cycles each message will be exchanged and, finally, (iii) creating schedules for each link on the network. We have designed the algorithm based on the MILP formulation of the routing and integration cycle assignment problem and the CP formulation of the link scheduling problem.

The experiments show that the incremental scheduling on one side prolongs the part of the integration cycle used by the TT communication in the order of a percent (in the experiments it was about 1 %), but it brings the advantage of backward compatibility. The performance of the algorithm is dependent on the number of messages in the instance, the length, and the periodicity of the messages. However, the experiments show that the method can return good results even for industry sized instances in a few minutes.

## List of Symbols and Abbreviations

**Abbreviations**

| | |
|---|---|
| BE | Best-Effort |
| CP | Constraint Programming |
| ECU | Electronic control unit |
| ILP | Integer Linear Programming |

| | |
|---|---|
| RC | Rate-Constrained |
| SMT | Satisfiability modulo theories |
| SPT | Shortest Path Tree |
| TT | Time-Triggered |

**Symbols**

| | |
|---|---|
| $\tau$ | Hop delay |
| $\tilde{a}_{i,j}$ | Predicate denoting if message $m_i$ was scheduled to integration cycle $ic_j$ in original schedule |
| $\tilde{MI}$ | Message instances present in the original schedule |
| $\tilde{o}_i^{l,m}$ | Start of the transmision time for message instance $m_i^{l,m}$ in the original schedule |
| $a_{i,j}$ | Decision variable for integration cycle assignemnt problem |
| $c_i$ | Length of message $m_i$ |
| $c_i^{l,m}$ | Transmission time of message $m_i$ in link $k_{l,m}$ |
| $cc$ | Cluster cycle |
| $d_i$ | Deadline of message $m_i$ |
| $E$ | Set of nodes |
| $E^E$ | Communication endpoints |
| $E^R$ | Redistribution nodes |
| $e_i$ | Node $i$ |
| $F$ | Load of the most bussy link |
| $f_{i,m}$ | Load of the link $k_{l,m}$ |
| $I$ | Set of integration cycles |
| $ic$ | Integration cycle |
| $K$ | Set of links |
| $k_{i,j}$ | Link between nodes $e_i$ and $e_j$ |
| $m_i$ | Message $i$ |
| $m_i^{l,m}$ | Instance of message $m_i$ on link $k_{l,m}$ |
| $MI$ | Set of message instances |
| $MI_i^{l,m}$ | Set of message instances which appear in integration cycle $ic_i$ and link $k_{l,m}$ |
| $Q_i$ | Set of receivers of message $m_i$ |
| $q_i$ | Transmitter of message $m_i$ |
| $r_i$ | Release date of message $m_i$ |
| $S_i$ | Routing tree of message $m_i$ |
| $S_i^q$ | Routing path of message $m_i$ to receiver $e_q$ |
| $s_{i,l}$ | Numerical label of node $e_l$ for message $m_i$ |
| $t_i$ | Period of message $m_i$ |
| $x_{i,l,m}$ | Decision variable for routing |
| $z$ | Maximal bandwidth utilization among integration cycles |

## References

[1] Z. Li, H. Wan, Z. Pang, Q. Chen, Y. Deng, X. Zhao, Y. Gao, X. Song, and M. Gu, "An enhanced reconfiguration for deterministic transmission in time-triggered networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 1124–1137, 2019.

[2] International Organization for Standardization. (2015) ISO 17458 - FlexRay communications system. [Online]. Available: http://www.iso.org

[3] S. Tuohy, M. Glavin, C. Hughes, E. Jones, M. Trivedi, and L. Kilmartin, "Intra-vehicle networks: A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 534–545, 2015.

[4] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, "The time-triggered ethernet (TTE) design," in *8h IEEE Int. Symp. Object-Oriented Real-Time Distributed Computing (ISORC)*. IEEE, 2005, pp. 22–33.

[5] SAE International, "AS6802: Time-Triggered Ethernet," SAE International, Tech. Rep., 2011.

[6] D. Tamas-Selicean, P. Pop, and W. Steiner, "Timing Analysis of Rate Constrained Traffic for the TTEthernet Communication Protocol," in *18th IEEE Int. Symp. on Real-Time Distributed Computing*, Auckland, New Zealand, 2015, pp. 119–126.

[7] ARINC (Aeronautical Radio, Inc.), "ARINC 664P7: Aircraft Data Network, Part 7, Avionics Full-Duplex Switched Ethernet Network," ARINC (Aeronautical Radio, Inc.), Tech. Rep., 2009.

[8] W. Steiner, "An Evaluation of SMT-Based Schedule Synthesis for Time-Triggered Multi-hop Networks," in *31st IEEE Real-Time Syst. Symp. (RTSS)*, San Diego, CA, USA, 2010, pp. 375–384.

[9] ——, "Synthesis of static communication schedules for mixed-criticality systems," in *14th IEEE Int. Symp. Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW)*, Newport Beach, CA, USA, 2011, pp. 11–18.

[10] A. Sharifnassab and S. J. Golestani, "On the possibility of network scheduling with polynomial complexity and delay," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3850–3862, 2017.

[11] M. Boyer, H. Daigmorte, N. Navet, and J. Migge, "Performance impact of the interactions between time-triggered and rate-constrained transmissions in ttethernet," in *8th European Congress on Embedded Real Time Software and Syst.*, Toulouse, France, 2016, pp. 159–168.

[12] D. Tamas-Selicean, P. Pop, and W. Steiner, "Design optimization of TTEthernet-based distributed real-time systems," *Real-Time Syst.*, vol. 51, no. 1, pp. 1–35, Jan 2015.

[13] W. Steiner, M. Gutirrez, Z. Matyas, F. Pozo, and G. Rodriguez-Navas, "Current techniques, trends and new horizons in avionics networks configuration," in *34th IEEE/AIAA Digital Avionics Syst. Conf. (DASC)*, Prague, Czech Republic, 2015, pp. 1–26.

[14] J. Wang, P. Ding, Y. Wang, and G. Yan, "Back-to-back optimization of schedules for time-triggered ethernet," in *37th Chinese Control Conference (CCC)*, Wuhan, China, July 2018, pp. 6398–6403.

[15] J. Dvořák, M. Heller, and Z. Hanzálek, "Makespan minimization of time-triggered traffic on a ttethernet network," in *Proc. 13th IEEE Int. Workshop on Factory Commun. Sys. (WFCS)*, Trondheim, Norway, 2017, pp. 1–10.

[16] F. Pozo, G. Rodriguez-Navas, and H. Hansson, "Methods for large-scale time-triggered network scheduling," *Electronics*, vol. 8, no. 7, 2019.

[17] S. S. Craciunas, R. S. Oliver, and V. Ecker, "Optimal static scheduling of real-time tasks on distributed time-triggered networked systems," in *Proc. IEEE Emerging Technology and Factory Automation (ETFA)*, Barcelona, Spain, 2014, pp. 1–8.

[18] S. S. Craciunas and R. S. Oliver, "Combined task- and network-level scheduling for distributed time-triggered systems," *Real-Time Syst.*, vol. 52, no. 2, pp. 161–200, Mar 2016.

[19] R. Zhao, G. Qin, Y. Lyu, and J. Yan, "Security-aware scheduling for ttethernet-based real-time automotive systems," *IEEE Access*, vol. 7, pp. 85 971–85 984, 2019.

[20] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks," in *Proc. 24th Int. Conf. on Real-Time Networks and Syst. (RTNS)*, Brest, France, 2016, pp. 183–192.

[21] L. Zhao, P. Pop, and S. S. Craciunas, "Worst-case latency analysis for ieee 802.1qbv time sensitive networks using network calculus," *IEEE Access*, vol. 6, pp. 41 803–41 815, 2018.

[22] O. Rottenstreich, M. Di Francesco, and Y. Revah, "Perfectly periodic scheduling of collective data streams," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1332–1346, 2017.

[23] T. Steinbach, F. Korf, and T. C. Schmidt, "Comparing time-triggered Ethernet with FlexRay: An evaluation of competing approaches to real-time for in-vehicle networks," in *Proc. 8th IEEE Int. Workshop on Factory Commun. Syst.*, Nancy, France, 2010, pp. 199–202.

[24] P. Laborie, "A (Not So Short) Introduction to CP Optimizer for Scheduling," in *27th Int. Conf. Automated Planning and Scheduling (ICAPS)*, 2017.

[25] A. Novak, P. Sucha, and Z. Hanzalek, "Efficient algorithm for jitter minimization in time-triggered periodic mixed-criticality message scheduling problem," in *Proc. 24th Int. Conf. on Real-Time Networks and Syst. (RTNS)*, Brest, France, 2016, pp. 23–31.