Scheduling of PROFINET IRT Communication in Redundant Network Topologies

Lukasz Wisniewski¹, Verena Wendt¹, Jürgen Jasperneite¹,², and Christian Diedrich³

¹InIT – Institute Industrial IT, OWL University of Applied Sciences, D-32657 Lemgo, Germany
lukas.wisniewski@hs-owl.de, verena.wendt@hs-owl.de
²Fraunhofer IOSB-INA, D-32657 Lemgo, Germany
juergen.jasperneite@iosb-infraunhofer.de
³Otto-von-Guericke Universität, Magdeburg, Germany
christian.diedrich@ovgu.de

Abstract—Scheduling of time triggered communication systems belongs to the well known hard mathematical optimization problems. This paper tackles the communication scheduling problem in redundant network topologies, where the real-time process data has to be duplicated and sent on two different paths to its destination. This paper proposes an algorithm that is able to find a feasible schedule in a short time, while keeping the makespan of the schedule in average 10% shorter than the makespan of the schedule produced by a commercial tool, which is the current benchmark.

I. INTRODUCTION

Current Networked Control Systems (NCSs) build upon Time Triggered Ethernet based (TTEB) technologies achieve sufficient, high real-time performance and determinism, allowing communication with cycle times in the range of microseconds with jitter below 1µs. This high performance is achieved by a definition of strict communication rules, according to which the real-time communication is organized. These rules can be mapped to medium access approaches, such as the summation frame, where all nodes have to include or extract their real-time data to/from a frame that circulates in a logical ring. Another way to achieve this high performance is the utilisation of a strict TDMA communication schedule, where either one master node holds a schedule and polls all slave nodes for real-time data, or the schedule is distributed to all network members, which have to send data according to the assigned time slots. Such strict rules in form of a schedule are calculated for a specific setup that is defined by the network topology, involved devices with their individual characteristics, and communication requirements. While existing TDMA approaches are able to achieve high real-time performance, the level of flexibility of these systems is rather limited. In fact, whenever a setup changes, the communication rules need to be redefined and the whole NCS has to be (typically) restarted [1]. In future manufacturing systems that require much more flexibility and adaptability, scheduling as a part of the engineering process has to be performed much more frequently as well. By decreasing the scheduling time the overall reconfiguration time can therefore be optimised [1]. Currently, TTEB systems are typically scheduled using static schedulers that perform the scheduling in advance based on a priori knowledge. In the current literature, scheduling of time triggered communication systems is typically defined as a resource constrained project scheduling problem (RCPS), or particularly RCPS with temporal constraints (RCPS/TC) [2] or job shop scheduling (JSP) [3], which are known NP hard problems. Another method of scheduling is by using graph colouring techniques, as described in [4]. However, this also becomes an NP hard problem, where only a sub-optimal schedule can be found in a reasonable amount of time. Another method defines the scheduling problem as JSP, and solves it using a mixed integer linear program (MILP) formulation. MILP formulation had been also used by the authors in [5] and [6]. Both approaches were developed to schedule communication in FlexRay networks, the latter one also in TTP/C based systems. In [6], authors demonstrated that in small network setups composed of 4 devices, where only 20 jobs have to be scheduled, it is possible to calculate a feasible schedule in a time range between 10 seconds and 5 minutes. However, MILP is an NP-hard mathematical problem. Therefore, in almost all cases, heuristic solvers are used to solve such problems. As shown in [2], by using solvers such as CPLEX [7] and formulating scheduling problems as an integer linear program, it is possible to calculate an optimal solution for topologies, such as star or line, build upon 30 IO Devices (IODs). However, this process may take several hours. Sub-optimal schedules for the same scenarios can be found in a time range between few seconds up to several minutes. Authors in [8] showed that with SMT solvers, it is possible to calculate a communication schedule for a TTEB setup with 1000 frames in about half an hour. A completely different scheduling approach was presented in [9] and [10]. In both papers, authors proposed different approaches, where a packing mechanism was used to avoid communication scheduling. Authors in [11] proposed a greedy scheduling approach, able to calculate an optimal schedule for network topologies, such as line or tree. It has been shown that the complexity of the scheduling approach mainly depends on the time complexity of the sorting algorithm utilised in the scheduling process. However, the proposed algorithm does not support redundant network topologies, where in order to increase communication
reliability the process data is duplicated and sent using two (possibly disjoint) paths [12]. The aim of this paper is to extend this simple approach without influencing its time complexity, thus allowing to use it in future flexible manufacturing systems.

II. TIME TRIGGERED COMMUNICATION SCHEDULING

The job of the scheduler is to arrange a set of tasks in an order that fulfills a defined optimisation criteria, such as waiting time, throughput or makespan ($C_{\text{max}}$). In this work a task $T_i$ is defined as delivery of a frame from the source to its destination. So the scheduler has to calculate the start time ($s_i,j$) of each task, in such way that all defined constraints are fulfilled and deadlines met. Since in the transmission of a single frame, several devices may be involved, one task $T_i$ may consist of several activities, such that $T_i = \{t_{i,1}, ..., t_{i,n}\} : 1 \leq i \leq m, m \in \mathbb{N}$. An activity is defined as a transmission of a frame on a particular port. The superscript of $t$, defines the membership of an activity to a task and the subscript defines the consecutive number of a particular activity within a task.

The number of activities that belong to one task depends on the number of devices involved in a transmission of one frame. Furthermore, each device in a network requires a defined amount of time $p_i$ to process (transmit) each frame. This mean that during this time no other frames can be transmitted. This time can be calculated according to the following formula:

$$p_i = (\text{frameLength} + \delta) \cdot \text{PropagationDelay}$$  

where $\delta$ includes Preamble, SFD, and mandatory in Ethernet, the inter frame gap (IFG). $\text{PropagationDelay}$ in a Fast Ethernet is equal to 80ns for each sent byte of data. In a time triggered communication system based on the TDMA approach, the communication schedule is calculated for a specific network setup. According to the Engineering Guideline of the PROFINET IRT protocol [13] the setup can be described using few characteristic parameters, used later on by the scheduling algorithm:

- **Network topology** – describing how, and which devices are connected with each other, which ports had been used for these connections and what are the cable lengths.
- **Device characteristics** – including communication parameters of network components, such as switch forwarding delays (internal switch delay), ports delays (transmission (tx) and receiving (rx) port) and cable delays.
- **Messages** – holding an information about the source and destination of a particular message, its length, communication cycle or defining if a message should be send every cycle, every second, third, and so on.

Such scheduling problem with all mentioned above characteristic input parameters can be expressed using a graph as shown in figure 1. The graph has been based on the simplest ring topology as shown in figure 2a, consisting of one IO Controller (IOC) and two IO Devices (IODs). In the graph presented in figure 1, there are two types of edges, directed and undirected. Directed edges are automatically set by the precedence constraints of the consecutive activities that belong to the same task, e.g. $s_{i,1} > s_{i,2} > s_{i,3}$. It is due to the fact that TTEBs use a deterministic cut-through forwarding method, where queuing delays are excluded. As illustrated in figure 2b, the weight of each edge can be calculated using the following parameters: $\text{TxPortDelay}$, $\text{LineDelay}$, $\text{RxPortDelay}$ and the internal $\text{BridgeDelay}$ of the device, towards which a frame is sent, for more details see [11]. All these parameters, excluding $\text{LineDelay}$ are included in the device description file (DDF) of each TTEB device. The undirected edges (disjunctions) are used to describe dependencies between activities that belong to different tasks and have to be performed on the same port. So the edge direction has to be defined by the scheduling algorithm. The weight of this edge is described as the time necessary to completely transmit a frame, thus it is equal to $p_i$. In the problem presented in figure 1, all frames have 64 bytes and $\delta$ includes the IFG of 14 bytes specific for PROFINET IRT protocol. All disjunctions can be formalised in the following way. Having two activities $t_i$ and $t_j$, where the transmission time of each of them is defined by $p_i$ and $p_j$ and both of them have to be processed on the same port, the are two possibilities how they can be scheduled:

$$s_i + p_i + t_i \leq s_j \lor s_j + p_j \leq s_i$$  

where additionally $s_i, s_j \geq 0$. Disjunctions do not allow to formulate this optimization problem as a linear program (LP) that would be solvable in a polynomial time. Instead it becomes a MILP, where its time complexity grows while increasing the number of constraints. In the presented example, we have simple constraints defined by a performance of consecutive activities, belonging to the same task and disjunctive constraints that are particularly difficult to solve. The number of simple constraints $N'_{\text{seq}}$ for a ring topology can be calculated according to the following formula:

$$N'_{\text{seq}} = 4 \cdot \sum_{i=1}^{N} i$$  

Fig. 1. Representation of the scheduling problem - as a disjunctive graph

Fig. 2. Smallest possible ring topology
where $N$ is the total number of nodes in the setup and $4$ is a factor due to the fact that frames are send in two directions from IOC to IODs and conversely each frame is duplicated. The number of disjunctive constraints can be calculated using the binomial coefficient $\binom{n}{k}$. Here, $n_i$ defines the number of activities that have to be scheduled on a selected port and $k$ is equal to two, since always $2$ activities are considered out of $n_i$. Additionally, it has to be multiplied by a factor $2$, since each disjunction consists of $2$ inequalities, as shown with eq. $2$. The total number of inequalities describing the disjunctions $N_{ieq}''$ can be calculated according to the following formula:

$$N_{ieq}'' = \sum_{i=1}^{N} \frac{2^{n_i} n_i!}{2!(n_i - 2)!} = \sum_{i=1}^{N} n_i (n_i - 1)$$

(A) Slip-stream Effect - Solving Disjunctions

The first step will be explained based on the ring topology shown in figure 2a. Here, the communication is organised in two directions Uplink and Downlink, where each device sends two frames towards its destination (original and duplicate). Figure 3 illustrates a sorted list of activities, according to which the scheduling process will be executed. Sorting the proposed approach had been organised in two steps. In the first step, the slip-stream effect [11] is used to define only the order, in which a particular task will be scheduled at each device. In the second step, using linear programming (LP) formulation, the exact start time $s_i$ of each task is calculated. Sorting

$\begin{array}{c|c|c}
\text{Task ID} & \text{Direction} \\
\hline
1 & \text{IOC\rightarrowIOD1} & \text{IOD1\rightarrowIOD2} & \text{A Downlink} \\
2 & \text{IOD1\rightarrowIOC} & \text{B Uplink} \\
3 & \text{IOC\rightarrowIOD2} & \text{IOD2\rightarrowIOD1} & \text{C Downlink} \\
4 & \text{IOD2\rightarrowIOC} & \text{D Uplink} \\
5 & \text{IOC\rightarrowIOD1} & \text{E Downlink} \\
6 & \text{IOD2\rightarrowIOD1} & \text{IOD1\rightarrowIOC} & \text{F Uplink} \\
7 & \text{IOD2\rightarrowIOC} & \text{G Downlink} \\
8 & \text{IOD1\rightarrowIOD2} & \text{IOD2\rightarrowIOC} & \text{H Uplink} \\
\end{array}$

Fig. 3. Working principle of the slip-stream based scheduling - ring example of tasks was based on the metric $e_i$. In case of Downlink direction, tasks with the greatest $e_i$ are scheduled first. In case of Uplink direction, it is the reversed order. Afterwards, both lists are merged together. In order to better illustrate the working principle, highlight conflicts, and present necessary shifts of $s_i$, an additional sequence diagram is illustrated in figure 4. The algorithm takes as an input the sorted activity lists illustrated in figure 3 and calculates the $s_i$ of the first activity, while going through the column $j = 1$. After solving all conflicts and setting order at each device it increments $j$. The whole process of setting the order of performance of tasks at each device is illustrated in figure 4. Figure 4a illustrates the first conflict at the IOC on its both ports. This follows to a definition of the first two constraints, $A \succ E$ and $C \succ G$ at the IOC. In the next step, shown in figure 4b, $s_i$ of the second activity of task $A$ is calculated. Here a conflict situation with the task $H$ occurs, following to a definition of another constraint, $H \succ A$ at the IOD1. The next activity to be scheduled as shown in figure 4c, is the second activity of task $C$. Here, a conflict situation with the task $F$ occurs, following to a constraint $F \succ C$ at the IOD2. The next activity to be scheduled belongs to the task $F$, as shown in figure 4d. The shift is needed due to the conflict with already scheduled task $B$, what follows to the constraint $B \succ F$ at the IOD1. The last step is illustrated in figure 4e and represents the conflict of the task $H$. The conflict is caused by the task $D$, thus creating the last constraint $D \succ H$ at the IOD2. Using the slip-stream effect as described above, it is possible to set an order, in which processing of activities at any port of any device has to be performed. This relaxes all disjunctions and allows to define
simple inequality constraints to describe the whole scheduling problem. This set of constraints is used in the second step, in order to calculate the $s_i$ of each task.

### B. Linear Program - Calculating Start Times ($s_i$)

Based on the results from the previous step, the scheduling problem can be represented by two directed graphs shown in figure 5. Such graph form allows to formulate the problem as LP. The optimization criteria can be described again as a minimize($t_{n+1} - t_0$). This problem is solved using DualSimplex, the results are presented in the next section.

### C. Results

Evaluation results are presented in table I. The proposed approach was evaluated using ring topology. The size of the ring was set from 10 up to 100 nodes ($N_0$), increasing the number of nodes by 10 in each iteration. Similarly like authors in [2], it has been compared with the MILP formulation and with results obtained from the commercial engineering tool from Siemens (Simatic S7) [15] that includes a framework to schedule PROFINET IRT communication. Two metrics were compared, the $C_{max}$ and the scheduling time $S$, see table I. As mentioned before, MILP turned to be a very hard problem, where the calculation time exploded after increasing number of nodes to 20. However, in case of 10 nodes, an optimal solution was found. The result from the proposed approach in the first scenario was around 19% worse than in case of MILP. The proposed approach was in each scenario around 10% better, in terms of $C_{max}$ and several magnitudes better in terms of calculation time $S$ than the commercial tool S7.

### IV. CONCLUSIONS

The approach presented in this paper, extends the simple greedy algorithm proposed in [11] by support of redundant topologies, such as rings. It has been shown that the proposed method outperforms the existing commercially available tool in terms of calculation time and resource efficiency. Further work will focus on testing this approach in other redundant network topologies.

### REFERENCES


