České vysoké učení technické v Praze
Fakulta elektrotechnická

Czech Technical University in Prague
Faculty of Electrical Engineering

Doc. Dr. Ing. Zdeněk Hanzálek

Rozvrhování pro časem řízené komunikační protokoly

Scheduling for Time-triggered Communication Protocols

Summary

This text summarizes our approach to a message scheduling in time-triggered protocols and its formulation as Resource Constrained Project Scheduling with Temporal Constraints. In addition, we extend the model by so called take-give resources, that are needed from the beginning of an activity to the completion of another activity. We formulate this problem by an integer linear programming.

Furthermore, we show, how to create a static schedule of the Profinet IO IRT communication protocol, which is an industrial Ethernet protocol standardized in IEC 61158. This approach offers an alternative to the available commercial tool, providing comparable results with respect to the resulting schedule makespan. Furthermore, we make use of temporal constraints providing a greater flexibility with respect to the individual messages. Due to this flexibility, it is possible to place the selected messages in various parts of the communication cycle (in order to increase the computational time available for the main-controller application, or to retransmit the synchronization message without holdup in the switch, or to add new messages into the original schedule). The solution is based on a formulation of the Profinet IO IRT scheduling problem in terms of the Resource Constrained Project Scheduling with Temporal Constraints.

Souhrn

Tato práce ilustruje náš přístup k rozvrhování zpráv v časem řízených komunikačních protokolech a formulaci tohoto problému pomocí Resource Constrained Project Scheduling s temporálními omezeními. Tento model dále rozšiřujeme o speciální typ zdroje nazvaný "take-give", který je využíván od začátku jedné aktivity po dokončení jiné aktivity. Problém je formulován pomocí celočíselného lineárního programování.

Dále ukazujeme, jak vytvořit statický rozvrh pro Profinet IO IRT komunikační protokol, což je průmyslový Ethernetový protokol standardizovaný v IEC 61158. Prezentovaný přístup umožňuje flexibilně formulovat časová omezení pro jednotlivé komunikační zprávy. Díky této flexibilitě můžeme umístit vybrané zprávy na začátek respektive na konec komunikačního cyklu (to umožní prodloužit čas potřebný pro hlavní řídicí smyčku nebo přeposlat synchronizační zprávu bez zdržení v zařízení switch nebo přidat nové zprávy do původního rozvrhu). Řešení je založeno na formulaci Profinet IO IRT rozvrhovacího problému pomocí Resource Constrained Project Scheduling s temporálními omezeními

Klíčová slova: rozvrhování, časem řízené komunikační protokoly, Profinet IO IRT, Resource Constrained Project Scheduling, celočíselné lineární programování.

Keywords: scheduling, time-triggered communication protocols, Profinet IO IRT, Resource Constrained Project Scheduling, integer linear programming.

Contents

1 Introduction **6**

2 Resource Constrained Project Scheduling with Temporal Constraints **8**

3 Integer Linear Programming Formulation **9**

4 Profinet IO Industrial Protocol **11**
4.1 Scheduling Problem Input Parameters 13
4.2 Problem Statement 15

5 Formalization of Profinet IO IRT scheduling as a RCPS/TC problem **16**
5.1 $PS\,|temp|\,C_{max}$ problem 16
5.2 Algorithm Description 18
5.2.1 Unicast Messages 18
5.2.2 Multicast Messages 18

6 Application Point of View **19**
6.1 Controller Application 20
6.2 Synchronization Messages using the Cut-Through Mechanism 20
6.3 Rescheduling - Adding New Messages and Nodes into the Original Schedule 21
6.3.1 Adding New Messages 21
6.3.2 Adding New Nodes and Messages 21

7 Conclusions **22**

8 Doc. Dr. Ing. Zdeněk Hanzálek **26**

1 Introduction

Contrary to the original intended use of the Ethernet, there has been a demand to use this technology even on the **field level** (real-time communication with sensors and actuators) of industrial automation. The intention is to use a single network technology, going through the whole factory up to the office level. The Ethernet has been increasingly adopted in industry, since it is wide-spread and well supported by chip manufacturers.

Profinet IO IRT is an Ethernet-based hard real-time communication protocol, which uses static schedules for time-critical data. Each node contains a special hardware switch, intentionally breaking the standard forwarding rules to ensure that no queuing delays occur for time-critical data. The objective of this paper is to find Profinet IO IRT schedules minimizing the makespan and respecting temporal constraints. Furthermore, the aim of this paper is to formulate the problem in terms of the Resource Constrained Project Scheduling with Temporal Constraints (RCPS/TC [1], [2], as described in Section 2), so that a possible user is not bound to a particular implementation but he/she can choose from a variety of algorithms solving this problem.

Here, we present some references that deal with time-triggered communication on the field level, and with the Ethernet and various industrial protocols built upon it. We also present some basic notions and related work referring to RCPS/TC.

However, the Ethernet was not designed to satisfy the requirements of real-time communication [3], [4], [5]. Therefore, many contributions from the research community suggested solutions to overcome the non-deterministic behavior of the Ethernet medium access and how to provide real-time communication guarantees. Some of the suggested methods propose traffic smoothing [6],[7] or a time-triggered approach [3], [8], [9]. The traffic smoothing relies on the fact that, up to a certain communication load, no collisions occur on the medium with a high probability. Thus, the generation of the messages by the application layer must be controlled in order to keep a low communication load. The centralized medium access may rely on the presence of a manager node (master) that determines when the individual controlled nodes (slaves) should transmit their messages, and ensures that the real-time data are separated in time from the non-real-time data. The decentralized access usually relies on the assumption that every node knows its schedule, defining when to start the communication.

The above mentioned approaches, which satisfy the real-time requirements with an Ethernet, have been realized in industrial standards, such as DDS [10], Ethernet Powerlink [11],[12], AFDX [13] or Profinet IO [14], [15],[16]. The latter one, Profinet IO (specifically, its RT Class 3, i.e. Isochronous Real Time – IRT) is dealt with in the rest of the paper.

Profinet IO IRT uses static schedules for time-critical data to fulfill the requirement on the timeliness of the data delivery. The individual-node schedules are downloaded into the nodes, each of which contains a switch. Thus, a special hardware (switch), unlike in the traditional Ethernet, is required to be able to accept the respective schedule. There is a commercial tool, part of the Siemens[1] Simatic Manager professional engineering software [17], that is able to generate the schedules. However, no papers describing the scheduling algorithm used in this commercial tool have been published. There is just some related work, such as [18], where graph algorithms used to solve some scheduling or planning problems are described, but they are not directly connected to the Profinet IO IRT scheduling. Paper [19] describes a scheduling algorithm that is based on an edge coloring of a graph that represents the network topology, but there are certain limitations, such as link

[1]Siemens, Simatic, Simotion and Sinamics are registered trademarks of Siemens.

delay and bridge delay parameters are not taken into consideration, and only half-duplex links are assumed.

Resource Constrained Project Scheduling problem with limited renewable resources and general temporal constraints is a well established model in the research community (see [20], [2], [21]).

Various types of solutions of Resource Constrained Project Scheduling problems with positive and negative time-lags have been proposed in literature. Most of exact algorithms are based on branch and bound technique [1] but this approach is suitable for problems with less than 100 activities [22]. A heuristic algorithm by Cesta et al [23] is based on constraint satisfaction problem solving. The algorithm is based on the intuition that the most critical conflicts to be resolved first are those involving activities with large resource capacity requirements. Another heuristic algorithm proposed in [24] combines the benefits of the "squeaky wheel" optimization with an effective conflict resolution mechanism, called "bulldozing". The possibility of improving on the squeaky wheel optimization by incorporating aspects of genetic algorithms is suggested in [25]. An overview of heuristic approaches is shown in [22] where the authors compare truncated branch and bound techniques, priority rule methods and schedule improvement procedures. A beam search heuristic is applied in [22] to scheduling of rolling ingots production. This problem cover renewable resources, changeover time and batching machines.

In addition, there are special resources (e.g. memory in the switch) that are needed from the beginning of an activity to the completion of another activity. This is why we extend the classical Resource Constrained Project Scheduling problem by so called **take-give resources**. Up to our knowledge, there is no work dealing with take-give resources in RCPS. Scheduling with **blocking operations** [26, 27] can be seen as a subproblem of scheduling with take-give resources. Operations are blocking if they must stay on a machine after finishing when the next machine is occupied by another job. During this stay the machine is blocked for other jobs, i.e. blocking operations models the absence of storage capacity between machines. On the other hand, there is a more general framework called **reservoirs** or **storage resources** [28] usually used to model limited storage capacity or inventory limits. In this framework each activity can replenish or deplete certain amount of a resource but the resource assignment is not considered. Therefore this framework can not deal with changeover times on given resource type, as it is required in the lacquer production problem to model mixing vessels cleaning.

This text is organized as follows: Section 2 describes the notation and the scheduling problem. The ILP problem formulation is shown in Section 3. Section 4 provides a brief review of the Profinet IO IRT standard and discusses the necessity to use special hardware (Ethernet switches). Furthermore, it shows how to derive the input timing parameters in order to obtain a message schedule, which can be executed on the Profinet IO IRT hardware. Section 4.2 refines the Profinet IO IRT scheduling problem including extension by temporal constraints (i.e. release dates, deadlines and end-to-end deadlines of the messages). The main contribution is shown in Section 5, presenting a solution of the problem. Section 6 illustrates how the problem extension by temporal constraints may be conveniently used in the application design.

2 Resource Constrained Project Scheduling with Temporal Constraints

We assume that the project deals with a set of $n+2$ non-preemptive activities $\mathcal{V} = \{0, 1, 2, ..., n+1\}$. Let $p_i \in \mathbb{R}_0^+$ be a processing time of activity i and $s_i \in \mathbb{R}_0^+$ be the start time of activity i. Activities 0 and $n+1$ with $p_0 = p_{n+1} = 0$ denote "dummy" activities which represent the project beginning and the project termination, respectively. The activities correspond to nodes of oriented graph $G = (\mathcal{V}, \mathcal{E})$ where $\mathcal{E}$ is a set of edges representing general temporal constraints between nodes. Each edge $e_{ij} \in \mathcal{E}$ from node i to node j is labeled by weight $\delta_{ij} \in \mathbb{R}$. The start times of activity i and activity j are subject to **temporal constraint** given by inequality

$$s_j - s_i \geq \delta_{ij} \qquad \forall (i,j) \in \mathcal{V}^2; e_{ij} \in \mathcal{E}. \tag{1}$$

Let d_{ij} is the length of the longest path from node i to node j, and $d_{ij} = -\infty$ when there is no such a path or when $i = j$. Inequality $s_j - s_i \geq d_{ij}$ holds for each tuple $(i,j) \in \mathcal{V}^2$ when there is no positive cycle. When there is a positive cycle, then $d_{ii} > 0$ for all nodes i taking part in such a cycle and consequently there is no feasible schedule.

The activities are executed on a set of m (renewable) resources $\mathcal{R} = \{1, 2, ..., m\}$. Resource $k \in \mathcal{R}$ has capacity of $R_k \in \mathbb{Z}^+$ units. Activity i requires $r_{ik} \in \mathbb{Z}_0^+$ units of resource k during its execution (i.e. from start time s_i to completion time $C_i = s_i + p_i$), such that $0 \leq r_{ik} \leq R_k$. Multiple resources may be required by one activity (sometimes called multiprocessor activity). Every activity i with $p_i = 0$ is executed on resource k with capacity $R_k = \infty$, such activity is assumed to be an event. Furthermore, we define assignment $z_{ivk} \in \{0, 1\}$, which is equal to 1 if activity i is assigned to unit v of resource k, and 0 otherwise. Consequently equation $\sum_{v=1}^{R_k} z_{ivk} = r_{ik}$ holds for each activity $i \in \mathcal{V}$ and each resource $k \in \mathcal{R}$.

In addition, we assume changeover time $o_{ij} \in \mathbb{R}_0^+$ (also called sequence dependent setup-up time) which satisfies triangular inequality. Occurrence of changeover in a given schedule is indicated by binary variable $change_{i,j,v,k}(s,z)$ which is equal to 1 if and only if $z_{ivk} = z_{jvk} = 1$ and activity i immediately precedes activity j on unit v of resource k. Consequently $s_j \geq s_i + p_i + o_{ij}$[2] holds when there exists unit v of resource k such that $change_{i,j,v,k}(s,z) = 1$.

For a given schedule (s,z), let the set of activities in progress at time t be denoted as $\mathcal{A}(s,z,t) := \{i \in \mathcal{V}; s_i \leq t < s_i + p_i\}$.

The **resource constraints** are defined as follows

$$\sum_{i \in \mathcal{A}(s,z,t)} r_{ik} \leq R_k \qquad \forall t \in [0, UB] \text{ and } \forall k \in \mathcal{R}; R_k < \infty, \tag{2}$$

where UB denotes an upper bound on the schedule length [1].

The NP-hard problem of minimizing a schedule length $C_{max} = \max_{i \in \mathcal{V}} \{C_i\}$, subject to the temporal and resource constraints, can be denoted by $PS\,|temp, o_{ij}|\,C_{max}$ in the tree-field notation of project scheduling problems (in order to avoid confusion with the start time, we use o_{ij} instead of s_{ij} proposed in [2]).

We extend this problem by a concept of **take-give resources** as follows. Let us assume a set of b take-give resources $\mathcal{Q} = \{1, 2, ..., b\}$. Take-give resource $k \in \mathcal{Q}$ has capacity of

[2]The inequality holds for both immediate and non-immediate precedence, due to the satisfaction of triangular inequality, i.e. $o_{ij} \leq o_{il} + o_{lj}\ \forall (i,l,j)\ \in \mathcal{V}^3$.

$Q_k \in \mathbb{Z}^+$ units such that $Q_k < \infty$. Analogously to activity executed on resources, we introduce **occupation** executed on take-give resources. Occupation i requires $a_{ilk} \in \{0,1\}$ units of take-give resource $k \in \mathcal{Q}$ during its execution. Occupation i starts its execution at s_i, start time of activity which takes a take-give resource, and finishes its execution at $C_l = s_l + p_l$, completion time of activity which gives back (i.e. releases) the take-give resource. Multiple take-give resources may be taken or given back by one activity, but there is at most one take-give resource taken by activity i and given back by activity l (i.e. $\sum_{k\in\mathcal{Q}} a_{ilk} \leq 1 \ \forall (i,l) \in \mathcal{V}^2$) and there is a path from i to l with $d_{il} > 0$. We define take-give assignment $\tilde{z}_{ivk} \in \{0,1\}$, which is equal to 1 if occupation i is assigned to unit v of take-give resource k, and 0 otherwise. Consequently equation $\sum_{v=1}^{Q_k} \tilde{z}_{ivk} = a_{ilk}$ holds for each $(i,l) \in \mathcal{V}^2$ and each resource $k \in \mathcal{Q}$.

In the same way, as for resources, we assume changeover time $\tilde{o}_{ij} \in \mathbb{R}_0^+$ on take-give resources which also satisfies triangular inequality. Occurrence of changeover in a given schedule is indicated by binary variable $\tilde{change}_{i,j,v,k}(s,\tilde{z})$ which is equal to 1 if and only if $\tilde{z}_{ivk} = \tilde{z}_{jvk} = 1$ and occupation i immediately precedes occupation j on unit v of take-give resource k. Consequently $s_j \geq s_l + p_l + o_{ij}$ holds when there exists unit v of resource k such that $\tilde{change}_{i,j,v,k}(s,\tilde{z}) = 1$.

For a given schedule (s,z), let the set of occupations in progress at time t is denoted as $\mathcal{O}(s,t,\tilde{z}) := \{(i,l) \in \mathcal{V}^2; \exists k \in \mathcal{Q}, a_{ilk} = 1, s_i \leq t < s_l + p_l\} \cup \{i \in \mathcal{V}; \exists j \in \mathcal{V}, \exists k \in \mathcal{Q}, \exists v \in \{1,\ldots,Q_k\}, \tilde{change}_{i,j,v,k}(s,\tilde{z}) = 1 \text{ and } s_i + p_i \leq t < s_l + p_l + o_{ij}\}$. The **take-give resource constraints** are as follows:

$$\sum_{(i,l)\in\mathcal{O}(s,t)} a_{ilk} \leq Q_k \qquad \forall t \in [0, UB] \text{ and } \forall k \in \mathcal{Q}. \tag{3}$$

A schedule $S = (s,z,\tilde{z})$ is feasible if it satisfies the temporal, resource and take-give resource constraints. A set of feasible schedules for given input data is denoted by $\mathcal{S}$. The problem to find a feasible schedule with minimal C_{max} can be denoted by $PS\,|temp, o_{ij}, tg|\,C_{max}$.

3 Integer Linear Programming Formulation

In this part, we define problem $PS|temp, o_{ij}, tg|C_{max}$ by Integer Linear Programming. Let x_{ij} be a binary decision variable such that $x_{ij} = 1$ if and only if i is followed by j and $\hat{x}_{ij} = 0$ if and only if j is followed by i. Furthermore, let $\hat{y}_{ij}$ be another binary decision variable such that if $\hat{y}_{ij} = 1$ the resource constraints are eliminated in effect, i.e. activities i and j can not share an unit.

We define $\mathcal{M}$ as a set of possible resource conflicts. The possible resource conflict is an unordered couple $\{i,j\}$, such that $\{i,j\} \in \mathcal{M}$ iff activity i and activity j share resource k of finite capacity (i.e. $\exists k \in \mathcal{R}; r_{ik} \cdot r_{jk} \geq 1$ and $R_k < \infty$) and the execution of one activity including changeover time can overlap with the execution of the other activity (i.e. $d_{ij} < (p_i + o_{ij})$ and $d_{ji} < (p_j + o_{ji})$).

We define $\mathcal{B}$ as a set of possible take-give resource conflicts. The possible take-give resource conflict is an unordered couple $\{i,j\}$, such that $\{i,j\} \in \mathcal{B}$ iff occupation i and occupation j share take-give resource k (i.e. $\exists l \in \mathcal{V}, \exists h \in \mathcal{V}, \exists k \in \mathcal{Q}; a_{ilk} \cdot a_{jhk} = 1$) and the execution of one occupation can overlap with the execution of the other occupation (i.e. $d_{ij} < 0$ and $d_{ji} < 0$).

Constraint (5) is a direct application of temporal constraint (1). Constraints (6), (7), (8), (9) and (10) correspond to resource constraints. The binary decision variables x_{ij} and y_{ij} define the mutual relation of activities i and j. Their relation is expressed with constraints (6) and (7). There are three feasible combinations:

$$\min s_{n+1} \tag{4}$$

subject to:

$$s_j - s_i \geq d_{ij}, \qquad \forall (i,j) \in \mathcal{V}^2;\ i \neq j,\ d_{ij} > -\infty \tag{5}$$

$$s_i - s_j + UB \cdot x_{ij} + UB \cdot y_{ij} \geq p_j + o_{ji}, \qquad \forall (i,j) \in \mathcal{V}^2;\ i \neq j,\ \{i,j\} \in \mathcal{M} \tag{6}$$

$$s_i - s_j + UB \cdot x_{ij} - UB \cdot y_{ij} \leq UB - p_i - o_{ij}, \qquad \forall (i,j) \in \mathcal{V}^2;\ i \neq j,\ \{i,j\} \in \mathcal{M} \tag{7}$$

$$-x_{ij} + y_{ij} \leq 0, \qquad \forall (i,j) \in \mathcal{V}^2;\ i \neq j,\ \{i,j\} \in \mathcal{M} \tag{8}$$

$$z_{ivk} + z_{jvk} - 1 \leq 1 - y_{ij}, \qquad \forall (i,j) \in \mathcal{V}^2, \forall k \in \mathcal{R}, \forall v \in \{1,\ldots,R_k\};\ i \neq j,\ \{i,j\} \in \mathcal{M},\ r_{ik} \cdot r_{jk} \geq 1 \tag{9}$$

$$\sum_{v=1}^{R_k} z_{ivk} = r_{ik}, \qquad \forall i \in \mathcal{V}, \forall k \in \mathcal{R};\ r_{ik} \geq 1, R_k < \infty \tag{10}$$

$$\tilde{p}_i = s_l + p_l - s_i, \qquad \forall (i,l) \in \mathcal{V}^2; \sum_{k \in \mathcal{Q}} a_{ilk} = 1 \tag{11}$$

$$s_i - s_j + UB \cdot \tilde{x}_{ij} + UB \cdot \tilde{y}_{ij} \geq \tilde{p}_j + \tilde{o}_{ji}, \qquad \forall (i,j) \in \mathcal{V}^2;\ i \neq j,\ \{i,j\} \in \mathcal{B} \tag{12}$$

$$s_i - s_j + UB \cdot \tilde{x}_{ij} - UB \cdot \tilde{y}_{ij} \leq UB - \tilde{p}_i - \tilde{o}_{ij}, \qquad \forall (i,j) \in \mathcal{V}^2;\ i \neq j,\ \{i,j\} \in \mathcal{B} \tag{13}$$

$$-\tilde{x}_{ij} + \tilde{y}_{ij} \leq 0, \qquad \forall (i,j) \in \mathcal{V}^2;\ i \neq j,\ \{i,j\} \in \mathcal{B} \tag{14}$$

$$\tilde{z}_{ivk} + \tilde{z}_{jvk} - 1 \leq 1 - \tilde{y}_{ij}, \qquad \forall (i,j,l,h) \in \mathcal{V}^4, \forall k \in \mathcal{Q}, \forall v \in \{1,\ldots,Q_k\};\ i \neq j,\ \{i,j\} \in \mathcal{B},\ a_{ilk} \cdot a_{jhk} = 1 \tag{15}$$

$$\sum_{v=1}^{Q_k} \tilde{z}_{ivk} = a_{ilk}, \qquad \forall (i,l) \in \mathcal{V}^2, \forall k \in \mathcal{Q};\ a_{ilk} = 1 \tag{16}$$

domains of input variables are: $d_{ij} \in \mathbb{R},\ p_i, o_{ij},\ \tilde{o}_{ij}, UB \in \mathbb{R}_0^+,\ r_{ik} \in \{0,1,\ldots,R_k\},\ a_{ilk} \in \{0,1\}$
domains of output variables are: $s_i \in [0, UB - p_i],\ z_{ivk}, \tilde{z}_{ivk} \in \{0,1\}$
domains of internal variables are: $\tilde{p}_i \in [0, UB],\ x_{ij}, \tilde{x}_{ij}, y_{ij}, \tilde{y}_{ij} \in \{0,1\}$

Figure 1: ILP model of $PS|temp, o_{ij}, tg|C_{max}$ problem

1. When $x_{ij} = 0$ and $y_{ij} = 0$, constraints (6) and (7) reduce to $s_j + p_j + o_{ji} \leq s_i$, i.e. j is followed by i.

2. When $x_{ij} = 1$ and $y_{ij} = 0$, constraints (6) and (7) reduce to $s_i + p_i + o_{ij} \leq s_j$, i.e. i is followed by j.

3. When $x_{ij} = 1$ and $y_{ij} = 1$, the constraints (6) and (7) are eliminated in effect and the activities i and j must be scheduled on different units.

4. Combination $x_{ij} = 0$ and $y_{ij} = 1$ is not feasible due to constraint (8).

The number of units is limited using variable z_{ivk} in constraints (9) and (10). The constraint (10) satisfies that each activity i is assigned to the appropriate number of units r_{ik} for each resource $k \in \mathcal{R}$. From constraint (9) follows that when two activities i and j can overlap, i.e. $y_{ij} = 1$ then the activities can not be processed on the same unit v on resource k since $z_{ivk} + z_{jvk} - 1 \leq 0$.

The inequalities (11), (12), (13), (14), (15) and (16) stand for take-give resource constraints. The variables $\tilde{x}_{ij}$, $\tilde{y}_{ij}$ and $\tilde{z}_{ivk}$ have the same meaning as x_{ij}, y_{ij} and z_{ivk} for resource constraints. The only difference is that $\tilde{p}_i$,processing time of occupation, is a variable while p_i ,processing time of activity, is a constant. Processing time $\tilde{p}_i$, expressed in equation (11) is given by s_i, start time of activity i which takes take-give resource $k \in \mathcal{Q}$ and completion time of activity l which gives back the take-give resource. Finally, the objective function of the ILP model (4) minimizes the start time of dummy activity $n + 1$, i.e. the last activity of the schedule.

4 Profinet IO Industrial Protocol

Profinet IO is defined in [14] and [15] as part of the international IEC 61158 standard and one of its aims has been to replace traditional fieldbus systems on the field and cell levels in the control system hierarchy. As there are many application areas with different requirements, there are also various classes of service defined in Profinet IO. In fact, there are 4 of them, called RT Class UDP, and RT Class 1 to RT Class 3, which differ in the real-time capabilities and the availability of the clock synchronization. For applications with soft real-time requirements, the basic communication principle relies on a switched full-duplex topology, where messages are forwarded according to the forwarding rules defined within IEEE 802 (see [29]).

The process data is exchanged between the **IO controller** and **IO device** using one of the specified real-time classes. There can also be **non-real-time − NRT** data (having lower priority than RT Class 1 and RT Class UDP) such as diagnostic or configuration information that are transmitted using the Profinet IO protocol, or other data using various protocols based on the TCP/IP suite. In-depth information about Profinet IO can be found in the IEC 61158 standard [14], [15], or in [30].

A typical topology, with four 4-port and one 2-port communication nodes, is depicted in Figure 2. There is a switch integrated in each node. Specifically, node N_1 is a Personal Computer (PC) equipped with a CP 1616 communication processor by Siemens [31], nodes N_3 and N_5 are PCs equipped with CP 1616 as well, node N_2 is an IM151-3 remote I/O device [32] by Siemens, and node N_4 is a Sinamics S120 drive [33] by Siemens as well. As shown in Figure 2, N_1 is a Profinet IO (PN IO) controller, the other nodes in the figure are PN IO devices.

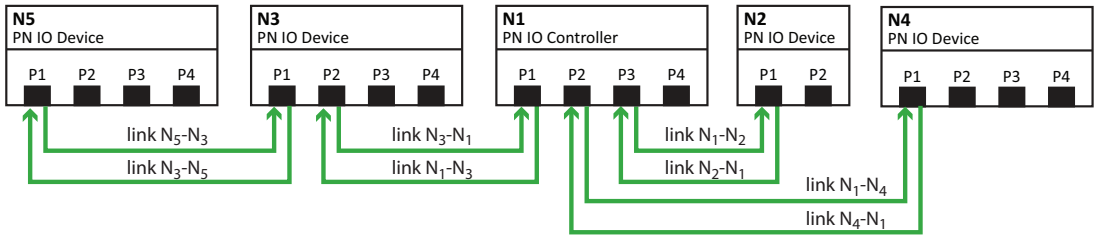

Figure 2: Topology with 5 nodes

Each communication link between two communication nodes is shown as two way links representing the full-duplex type of communication.

Figure 3 shows the **Profinet IO communication cycle** (also called communication period or send clock), divided into individual **communication intervals**.[3] For the purpose of this paper, the **red interval**[4] is important as it contains the **RT Class 3** data that are forwarded according to a static communication schedule. The RT Class 3 communication is the highest-priority one, and is required to be strictly isochronous. It means the individual data frames have to be transmitted at time instants, which are equidistant with respect to the consecutive communication cycles. This type of communication, also called **time-triggered**, relies on the ability of individual nodes to send the respective messages exactly at the pre-determined time instants. To be able to accomplish this, the nodes' clocks must be synchronized with such a precision that allows the jitter of the communication-cycle length to be as low as possible. Therefore, the **Precision Transparent Clock Protocol (PTCP)** is used, as defined in IEC 61158. In a typical scenario, the synchronization frames are sent in every communication cycle and are subject to the static schedule.

The schedule is computed during the engineering phase and loaded into the individual nodes. The lower part of Figure 3 shows an example schedule in the form of a Gantt chart, where each line corresponds to the link used and each task is labeled by its name, message ID and transmission delay T_{TD} in μs. Like in [30], the inter-message gaps (represented as grey boxes) are part of the transmission delay, which (e.g. for message 256) is $5.76+0.96 = 6.72\,\mu s$. However, in the example in this paper, we take the inter-message gap to be $1.12\,\mu s$ so that the resulting value of the T_{TD} would be $6.92\,\mu s$. Such a value has been taken from the data generated by the commercial tool in order to use the same input values for the computation as the commercial tool does. The schedule intentionally breaks the standard forwarding rules to ensure that no queuing delays occur in the switches (remember that a special switch is integrated in each node for RT Class 3 communication and no communication other than the scheduled one is allowed). If it occurred, it would be blocked by the respective switches. Please note that not all devices are required to exchange data, as N_5 has no data to send to N_3, although there is data transfer in the opposite direction (see Figure 3), for example.

[3] According to [14], it is also possible for the data to have various periods in the multiples of the communication cycle. The quality of having periods of various length is called the **reduction ratio**. However, we do not deal with this quality in the paper and take all messages (data) to have their reduction ratio equal to 1.

[4] The expression **red interval** is defined in [14] and means the communication interval dedicated for the RT Class 3 communication. As such, it is used in this paper.

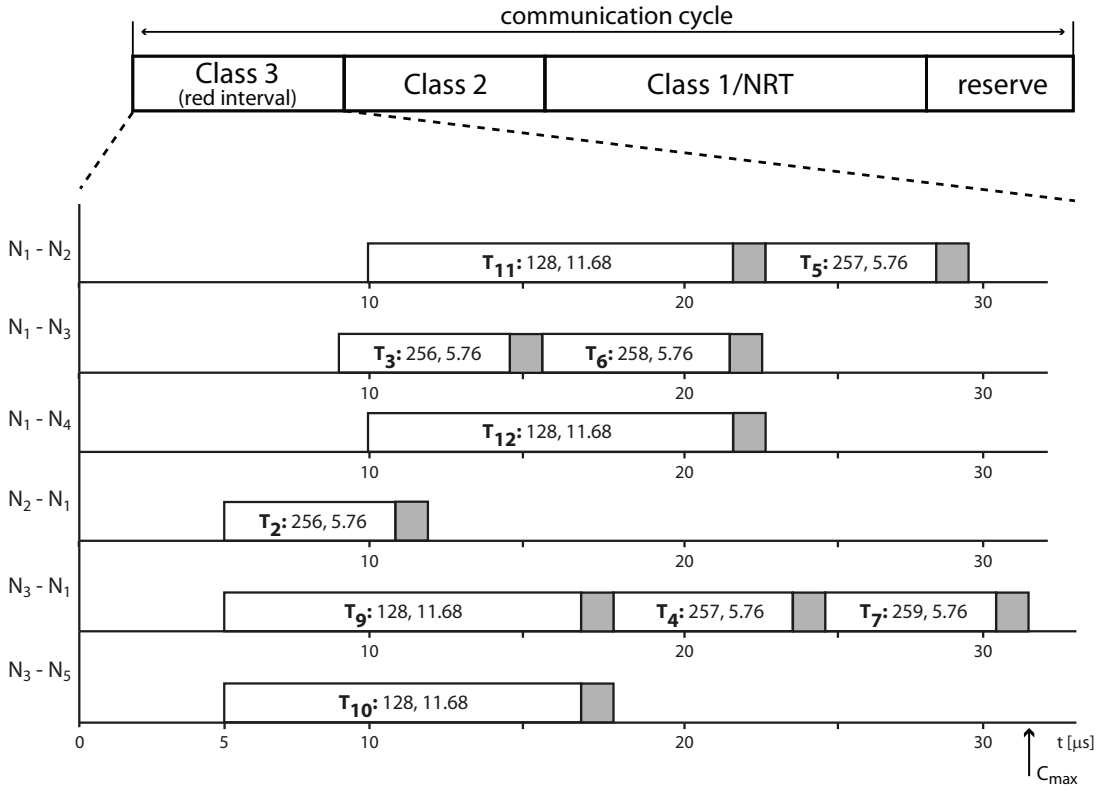

Figure 3: Profinet IO communication cycle

4.1 Scheduling Problem Input Parameters

Each switch introduces a considerable delay, composed of port-dependent and bridge-dependent parameters. An in-depth description of the delay parameters is provided in [14]. The main points, which are crucial for the specification of the scheduling problem, are presented here.

A link is a communication connection between two nodes and the link delay consists of the cable delay, the transmit-port delay and the receive-port delay. Another important parameter is the bridge delay because it represents the time, which is needed for a packet to be processed in the bridge. The packet is processed either locally (i.e., the local device is the recipient of the packet) or it is resent to another port. In the latter case the **cut-through** mechanism or the **store-and-forward** mechanism is used as it depends on how the schedule for individual messages behaves, as described later. Thus, the **communication link delay** T_{LD} between two ports is given as

$$T_{LD} = T_{TxD} + T_{CD} + T_{RxD} + T_{ad} + T_{BD}, \tag{17}$$

where $T_{CD} = T_{PD} \cdot L$. Here, T_{PD} is the single-bit propagation delay per one meter and L is the cable length given in meters. The value of T_{PD} usually ranges from 4 to 6 ns per one meter of cable, the typical value is 5 ns/m. The meaning of the other delay parameters is as follows: T_{TxD} is the transmit-port delay, T_{RxD} is the receive-port delay, T_{BD} is the bridge delay. The additional link delay parameter T_{ad} provides a safety margin for the clock synchronization precision.

The delay parameters for the topology in Figure 2 have been obtained from a real installation based on Profinet IO nodes, i.e. from the device data sheets called GSDML (Generic Station Description Meta Language) files. Namely, in the case of the CP-1616 device, the delay parameters are $T_{TxD} = 1192$ns, $T_{RxD} = 363$ns and $T_{BD} = 1720$ns. In

the case of the IM151-3 device, the delay parameters are $T_{TxD} = 158$ns, $T_{RxD} = 350$ns and $T_{BD} = 2720$ns. In the case of the Sinamics S120 device, the delay parameters are $T_{TxD} = 1212$ns, $T_{RxD} = 418$ns and $T_{BD} = 1920$ns.

link	$N_1 \to N_3$	$N_1 \to N_4$	$N_1 \to N_2$	$N_2 \to N_1$	$N_3 \to N_1$	$N_4 \to N_1$	$N_3 \to N_5$	$N_5 \to N_3$
T_{LD} [ns]	4875	5130	5862	3841	4875	4895	4875	4875

Table 1: Link delays relating to Figure 2

Based on these parameters and Equation (17), Table 1 can be computed to produce the resulting link delays for each of the links in the network. For the sake of simplicity, all links are considered to be 100 m long and the T_{PD} is considered to be 6 ns/m. As in the case of the inter-message gap, we also take the same value as generated with the commercial tool. The **additional link delay** is 1 μs as described in [14]. All these computations are based on the worst-case values. Real measured values could be read out from the devices' MIBs (Management Information Base), created using the SNMP protocol (Simple Network Management Protocol), but it is not a mandatory function and not all Profinet IO devices are required to support it.

ID	path	T_{TD} [ns]	$\widetilde{r}$ [ns]	$\widetilde{d}$ [ns]	$\widetilde{e}$ [ns]
256	$N_2 \to N_3$	5760	5000	20000	11000
257	$N_3 \to N_2$	5760	15000	40000	15000
258	$N_1 \to N_3$	5760	15000	–	–
259	$N_3 \to N_1$	5760	20000	35000	–
128	$N_3 \to \{N_1,N_2,N_4,N_5\}$	11680	5000	{–,–,–,18000}	{–,17675,17675,15000}

Table 2: Messages for the network in Figure 2

In order to be able to generate the message schedule, the messages to be communicated must be known **a priori**. For our example, the messages are given in Table 2.

The table contains messages with **process data** (message IDs from 256 to 259), and a **time-synchronization** message (having the message ID 128). Of course, only the RT Class 3 communication (i.e., messages being in the red interval) is taken into account.

The length of the messages is given by the application. The length of the synchronization information is 114 bytes, as defined by [14], and let us assume that the length of the process data is 40 bytes. [5]

In addition, 32 bytes are added to form the Ethernet-frame head and tail [34]. Therefore, the synchronization messages are expected to be 146 bytes long and the process data messages are expected to be 72 bytes long.

The **message transmission delay** T_{TD} is the time to transmit all the bits of a message from the transmit queue to the communication link. $T_{TD} = l \cdot b_{TD}$, where l is the message length in number of bits, and b_{TD} is the **single-bit transmission delay**, which is 10 ns in our case.

The schedule, resulting from the parameters in Tables 1 and 2, is shown in Figure 3. It can be seen that N_3 acts as a clock master as it is the source node of the synchronization message denoted as 128. This message is also an example of multicast communication as it is forwarded by every switch that receives it. Other messages are unicast, for example message 256 is forwarded just by N_1 to reach N_3. Columns $\widetilde{r}$, $\widetilde{d}$ and $\widetilde{e}$ represent the constraints required by the application engineer and are explained later in Section 4.2.

[5] The latter requirement is useful in cases when less than 40 bytes of data are required to be transmitted, but the minimum length of the Ethernet frames (72 bytes) must be kept.

4.2 Problem Statement

The objective of this paper is to find a time-triggered message schedule for the red interval (see Figure 3), which is based on a list of messages created in the engineering system, and on the network topology composed of individual switches and communication links among them as described in the previous section. To be able to propose the scheduling algorithm compliant with the IEC standard [14], the following rules are kept:

1. The maximum length of the red interval, given by the application requirements, defines the upper limit by which time the communication must be completed. At the beginning of the red interval there is a **safety margin** of $5\,\mu s$ (see [14]).

2. The messages on the same link are separated with a minimum inter-message gap, which is at least 960 ns (see [34]) but a commonly used value is 1120 ns. This inter-message gap is added to T_{TD} for the message-schedule computations.

3. As soon as the first bit of a message is received on a port (i.e., T_{LD} after it was sent from the previous node) and the message is to be forwarded to another link, it is forwarded. Such a mechanism is known as **a cut-through mechanism** and can be seen on message 256 in Figure 3. There may also be messages, which cannot be sent out immediately after their first bit is received because the outgoing port is busy at that time. Such behavior is common in ordinary Ethernet switches and causes unpredicted delays in the communication with such switches. However, with IRT switches, an outgoing port of a switch may be busy intentionally because the offline schedule says so. In such a case, a message may be delayed in the switch but the delay is always the same and is counted on for the rest of the communication cycle. We call such a situation **(partial) store-and-forward mechanism**, as is demonstrated on message 257 in Figure 3. Having a look at tasks T_4 and T_5 relating to message 257 in Figure 3 on links $N_3 - N_1$ and $N_1 - N_2$, respectively, we can see that T_5 is processed by node N_1 (i.e. message 257 is sent from node N_1) after a time, which is longer than T_{LD} because the outgoing port is busy processing task T_{11}. This situation is contrary to processing message 256 in tasks T_2 and T_3 where no delay occurs.

4. If the link delay T_{LD} is greater than the message transmission time T_{TD}, the respective message will be processed by one communication node at a time. However, if $T_{LD} < T_{TD}$, two nodes may process a different part of the same message at the same time.

5. The synchronization message does not have to be sent at the beginning of the interval. It can be planned anywhere within the interval if the synchronization message arrives at a device at equidistant time.

Furthermore, the scheduling of output messages in the red interval may be intentionally deferred (e.g. to accommodate future extensions of the computation time of the controller application). Therefore, the scheduling problem is extended by the message **release date** $\widetilde{r}$ (specifying the earliest moment when the first bit of the message is sent from the source node), the message **deadline** $\widetilde{d}$ (specifying the latest moment the last bit of the message is received by the destination node), and the message **end-to-end deadline** $\widetilde{e}$ (specifying the maximum time passing from the moment the first bit of the message was sent from the source node, to the moment the last bit of the message is received by the destination node).

Please note that this definition of the end-to-end deadline does not involve the processing time in the source and destination nodes. In this context, it is worth remembering that several tasks may constitute a message if the message is sent through several nodes to reach the destination node.

Parameters $\widetilde{r}$ and $\widetilde{d}$ form a fixed time-window for a message, whereas $\widetilde{e}$ constitutes a relative time-window to deliver the message at the destination, relative to the time it is sent by the source node. The time constraint given by $\widetilde{e}$ is redundant when $\widetilde{e} \geq \widetilde{d} - \widetilde{r}$, otherwise it cannot be replaced by $\widetilde{d}$, since the instant when the message is sent by the source node is not a-priori known.

5 Formalization of Profinet IO IRT scheduling as a RCPS/TC problem

This section describes an algorithm which finds a schedule of messages on communication links in a time-triggered network while minimizing the schedule length and respecting all temporal and resource constraints. Each unicast message may be seen as a chain of tasks executed on communication links starting at the source node and ending at the destination node. In a similar way, each multicast message is seen as a tree of tasks. As a result, each port of a node contains a list of tasks to be processed and for each task there is a start time, i.e., an instant, when the message is to be sent. This principle is used in Profinet IO IRT as described in Section 4 and there is a commercial design tool deriving such start times. Due to the tree topology (line topology, which is typical in industrial settings, is a special case of a tree) of the Profinet IO IRT communicating nodes, the routing of messages is determined, thus the routing is not subject to the discussed optimization and each task is assigned to a dedicated link. The objective of this work is to make a documented optimization algorithm, which is further able to incorporate various constraints, thus being able to solve more complex problems than the algorithm in the commercial tool. The key idea is to formulate the Profinet IO IRT scheduling problem in terms of the Resource Constrained Project Scheduling with Temporal Constraints minimizing the schedule makespan C_{max}, where each communication link is a resource, each transmission of a particular message on a particular link is a task, C_{max} is the length of the schedule and temporal constraints are used to describe $\widetilde{r}$, $\widetilde{d}$ and $\widetilde{e}$. Therefore, the scheduling algorithm first generates the instance of $PS\,|temp|\,C_{max}$ from the input data (such as the data in Tables 1 and 2), then the problem is solved and finally a result is interpreted in terms of the start times of the tasks executed on the communication links.

5.1 $PS\,|temp|\,C_{max}$ problem

The set of n tasks $\mathcal{T} = \{T_1, \ldots T_i, \ldots T_n\}$ with temporal constraints is given by a task-on-vertex graph G [35]. A message transmission over the communication link corresponding to task T_i is represented by vertex T_i on graph G and has a non-negative duration corresponding to p_i, the processing time of the task. The scheduling problem is to find a **feasible schedule** $(s_1, s_2, \ldots s_n)$, satisfying the **temporal constraints** and **resource constraints**, while minimizing the makespan C_{max}. The value of C_{max} corresponds to the minimum required length of the red interval in the case of Profinet IO.

Temporal constraints between the vertices are represented by a set of directed edges. Each edge from vertex T_i to vertex T_j is labeled by a weight w_{ij}, which constrains the

start times of the tasks T_i and T_j by the inequality

$$s_j - s_i \geq w_{ij}. \tag{18}$$

There are two kinds of edges: the edges with positive weights and the edges with negative weights, but Equation (18) holds for both of them. The edge with a positive weight w_{ij} (giving the minimum time lag), indicates that s_j, the start time of T_j, must be at least w_{ij} time units after s_i, the start time of T_i. The positive weights w_{ij} are used to represent:

- **precedence relation** ($w_{ij} = p_i$, i.e., T_j starts when T_i is completed at the earliest) is used for example, when the output product of T_i serves as the input of T_j
- **overlapping precedence relation** ($w_{ij} \leq p_i$, i.e., T_j starts $p_i - w_{ij}$ time units before the completion of T_i at the earliest) is used, for example, when the Profinet node uses a cut-through mechanism (e.g. communication of message 256 over the network in Figure 2 starts the transmission on link $N_1 \rightarrow N_3$ before the reception on link $N_2 \rightarrow N_1$ is completed)
- **release date** $\widetilde{r}_j$ of task T_j ($w_{ij} = \widetilde{r}_j$ and $s_i = 0$).

The edge, from vertex T_j to vertex T_i with a negative weight w_{ji} (giving the maximum time lag), indicates that s_j must be no more than $|w_{ji}|$ time units after s_i. Therefore, each negative weight w_{ji} represents $\widetilde{d}_j(s_i)$, a deadline of T_j depending on s_i, such that $\widetilde{d}_j(s_i) = s_i + |w_{ji}| + p_j$. Consequently:

- when T_j is the last task of the message and T_i is the first task of the same message, the edge with a negative weight may be used to represent the required end-to-end deadline $\widetilde{e}_{ji} = |w_{ji}| + p_j$
- when $s_i = 0$ (i.e., T_i is the task scheduled at time 0), the edge with a negative weight may be used to represent the absolute deadline $\widetilde{d}_j = |w_{ji}| + p_j$.

The non existence of a cycle with positive length is a necessary condition for the existence of a feasible schedule. Moreover, the solving algorithms have to consider the **resource constraints** preserving two tasks to be executed on the same resource (communication links in our case) at once. Two disjoint cases can occur for any unordered couple of tasks T_i and T_j executed on the same resource:
(i) task T_j is followed by task T_i and the corresponding constraint is $s_i - s_j \geq p_j$
(ii) task T_i is followed by task T_j and the corresponding constraint is $s_j - s_i \geq p_i$.
The exclusive OR relation between both cases leads to a non-convex representation of the set of feasible solutions (see ILP formulation in the Annex where enabling/disabling of the first/second case is handled by the decision variable x_{ij}).

A suboptimal solution of the RCPS/TC problem can be found by an Iterative Resource Scheduling (IRS) heuristic [36] which has proven good performance on several industrial size benchmarks. For a given upper bound of schedule length, the heuristic constructs a schedule according to the priority of tasks while the number of scheduling steps is limited by the budget. If a feasible schedule is found, the tasks are shifted to the left side, such that the time constraints and order (given by the values of decision variables x_{ij}) of the tasks is maintained. Furthermore, the improved schedule is used to update the upper bound. On the other hand, if a feasible schedule is not found within the given limit of

scheduling steps (i.e. budget) the schedule lower bound is increased. The best solution is searched iteratively using the bisection method over the interval given by the lower and upper bounds. As a task priority, we use the longest path from the task i to the end of the schedule.

5.2 Algorithm Description

The first step of the algorithm may be illustrated as a construction of the oriented graph G in Figure 4 from the data in Tables 1 and 2. Each communication link between two nodes is taken as one resource, which can execute the tasks to be scheduled. A task execution corresponds to a transmission of a message on the respective link. An assignment of the tasks to the resources is determined by the path traversed by the message. Since the routing in the tree topology is deterministic, each task is assigned to a specific resource (e.g. see Figure 4 where T_2 is assigned to the link 2-1). A unicast message corresponds to a chain of tasks and a multicast message corresponds to a tree of tasks.

5.2.1 Unicast Messages

Vertex T_2 in graph G in Figure 4, representing task T_2, corresponds to the transmission of message 256 over link 2-1 and T_3 corresponds to the transmission of the same message over link 1-3. In a similar way, T_4 and T_5 correspond to the transmission of message 257 over lines 3-1 and 1-2, T_6 corresponds to message 258 and finally T_7 corresponds to message 259. The dummy task T_1, preceding all other tasks, corresponds to the event which indicates the start of the schedule. Tasks $T_2 \ldots T_7$ are executed on the corresponding lines and task T_1 is executed on a dummy resource. The processing time of tasks are $p_2 = \ldots = p_7 = 6880$ and $p_1 = 0$. All unicast messages, in this example, have the same transmission delay of 5760 ns (see Table 2) and the inter–message time of 1120 ns.

The link delay (see Table 1) is represented as an edge with positive weight T_{LD} interconnecting the tasks of the given message in the chain corresponding to the path from the source to the destination node. The release date of a message is represented as an edge with weight $\widetilde{r}_{ID}$ from T_1 to the first task of the message. The deadline of a message $\widetilde{d}_j = |w_{ji}| + p_j$ is represented as an edge with weight $-(\widetilde{d}_{ID} - p_j)$ from T_j, the last task of the message, to T_1 scheduled at time 0. The required end-to-end deadline of a message is represented as an edge with weight $-(\widetilde{e}_{ID} - p_j)$ from T_j, the last task of the message, to T_i, the first task of the message.

5.2.2 Multicast Messages

The lower part of graph G in Figure 4 includes the synchronization message 128, which must be present in the final schedule. This is a **multicast message** and it is defined by the source node N_3 and a set of the destination nodes $\{N_1, N_2, N_4, N_5\}$.

The multicast tree of message 128 consists of the tasks $T_8, T_9, T_{10}, T_{11}, T_{12}$. Task T_8 is the dummy root of the tree (see the nodes connected by positive edges in Figure 4). Each link on the path from the source node to the destination node in Figure 2 corresponds to the task on the path from the root task to the leaf task in Figure 4.

The requirement to forward the synchronization message promptly by each switch may be expressed by the required end-to-end deadline $\widetilde{e}$ related to each destination vertex of the multicast message while using an edge with negative weight.

Graph G representing both unicast and multicast messages is given by W, the adjacency matrix of weights $w_{i,j}$, as shown in Figure 5. In W, $w_{i,j} = -\infty$ if there is no edge

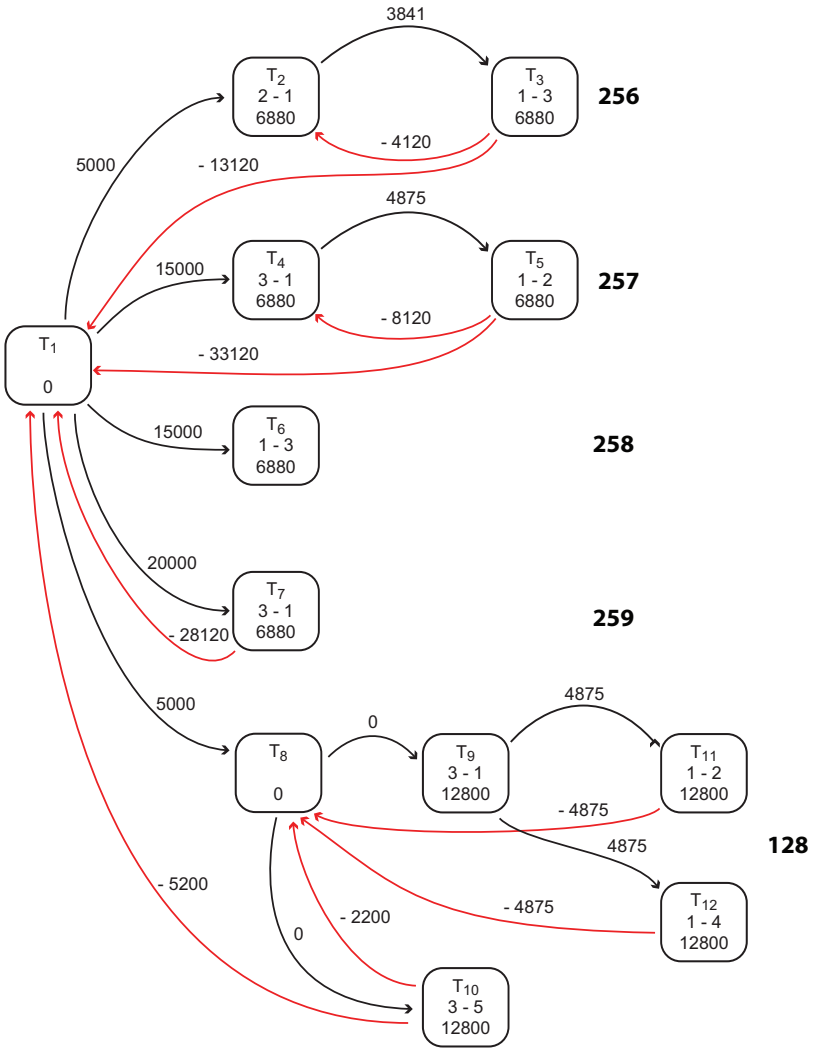

Figure 4: Graph of unicast and multicast messages for Fig. 2

in G from T_i to T_j. Each feasible schedule has to satisfy the resource constraints (at most one message is transmitted over the communication link at given time) and the temporal constraints (Equation 18 holds for all weights in the adjacency matrix W).

Therefore, the scheduling algorithm, which takes the parameters in Tables 1 and 2 as the input and generates the vector of start times s as the output, consists of two phases. The first phase takes the algorithm inputs and creates graph G, characterized by the processing time p of the tasks, the assignment a of tasks to links, and the adjacency matrix W. The second phase, computing the solution of the $PS\,|temp|\,C_{max}$ problem, takes p, a, W and finds s. The resulting schedule of the example is shown in Figure 3, where the dummy tasks are omitted.

6 Application Point of View

This section shows how the temporal constraints can be used by an application engineer to influence the timing of the messages with respect to the application demands. Three different application problems are defined by the temporal constraints: (1) prolongation of the computation time available for the controller application (see Subsection 6.1), (2) retransmission of messages without hold-up in the switch (see Subsection 6.2), and (3) ad-

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}
T_1	$-\infty$	5000	$-\infty$	15000	$-\infty$	15000	20000	5000	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_2	$-\infty$	$-\infty$	3841	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_3	-13120	-4120	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_4	$-\infty$	$-\infty$	$-\infty$	$-\infty$	4875	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_5	-33120	$-\infty$	$-\infty$	-8120	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_6	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_7	-28120	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_8	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	0	0	$-\infty$	$-\infty$
T_9	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	4875	4875
T_{10}	-5200	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-2200	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_{11}	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-4875	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_{12}	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-4875	$-\infty$	$-\infty$	$-\infty$	$-\infty$

Figure 5: Adjacency matrix of weights $w_{i,j}$

ding new messages and nodes into the original schedule (see Subsection 6.3). However, other application requirements can be fulfilled by formulating them in terms of the message scheduling as shown in Section 5.

6.1 Controller Application

In a typical control application, an input-data message is sent within the red interval from an input device to a controller, where the control algorithm is computed, and consequently the output-data message is sent within the next red interval from the controller to an output device. The control algorithm runs with a constant period equal to the length of the communication cycle T_{DC}, but the processing time of the control algorithm is shorter than T_{DC}.

For a given **input-output delay** (consisting of the transmission of inputs $\rightarrow$ computation $\rightarrow$ transmission of outputs) it is often required to have the computation part as long as possible. This requirement may be accomplished by moving the input-data messages into an earlier part of the red interval (using deadline $\widetilde{d}$), while the output-data messages must be moved to its later part (using release date $\widetilde{r}$).

Placement of the messages in the communication cycle may be useful in other situations as well, not only when we need to prolong the computation time of the main controller application. As an example, we have a situation when we use a PC, which does not contain the special Ethernet switch needed by Profinet IRT. Integration of such a device may be solved by dedicating a time window at the beginning of the communication cycle, which is used to send data from the PC to another node, which fully supports Profinet IRT. Therefore, the deadlines of the messages transmitted by the PC must be shorter than the release dates of the messages transmitted by the other node.

6.2 Synchronization Messages using the Cut-Through Mechanism

The scheduling of the synchronization messages can utilize the end-to-end deadline parameters for the messages to be forwarded without any additional delay. In the case of a hardware-based time stamping, which inserts the actual time value in the outgoing message, the delay is measured autonomously and is capable of adapting to any additional hold-up in the switch. However, the time-stamping mechanism is not standardized. Thus, for safety reasons, it is convenient to forward the synchronization message using the cut-through mechanism in every device. The instant delay necessary to recognize

the destination is hidden in the T_{BD} parameter, which T_{LD} is composed of. Thus, the end-to-end deadline values for each synchronization message ensure no hold-up occurs.

6.3 Rescheduling - Adding New Messages and Nodes into the Original Schedule

Several important parameters, like sampling periods, controller constants, priorities of the controller application tasks, are usually tuned at different levels of the application development cycle. Application engineers are usually quite reluctant to change them, once they have proven to be correct in the industrial environment. Nevertheless, the users, very often, require an extension of the system that includes additional messages and nodes. Such an addition usually leads to a change in the schedule when the scheduling algorithm is applied, as explained in Section 5. It means that the start time of a task in the new schedule may be different from the one in the original schedule. Therefore, some kind of temporal isolation is needed to keep the tasks in their original positions.

case	ID	path	T_{TD} [ns]	$\widetilde{r}$ [ns]	$\widetilde{d}$ [ns]	$\widetilde{e}$ [ns]
1	260	$N_4 \rightarrow N_3$	5760	5000	–	–
1	261	$N_3 \rightarrow N_5$	5760	5000	25000	–
1	262	$N_5 \rightarrow N_3$	5760	5000	15000	10000
1	263	$N_2 \rightarrow N_1$	5760	5000	–	–
1	264	$N_4 \rightarrow N_5$	5760	5000	–	35000
2	260	$N_6 \rightarrow N_5$	5760	5000	35000	–
2	261	$N_4 \rightarrow N_6$	5760	5000	35000	–
2	128	$N_3 \rightarrow N_6$	11680	5000	–	12800

Table 3: Additional messages

The $PS\,|temp|\,C_{max}$ formalism offers a straightforward solution to this problem. Task T_i, present in the original schedule at start time s_i, is fixed in its position by adding one edge with a positive weight of s_i from T_0 to T_i and by adding one edge with a negative weight of $-s_i$ from T_i to T_0. The additional messages are represented by additional tasks in the usual way as was shown in Subsection 5.2.

6.3.1 Adding New Messages

Let us illustrate this process on the original schedule shown in Figure 3 where the start times of the original tasks are:

$$s = [0, 5000, 8841, 17800, 22675, 15721, 24680, 5000, 5000, 5000, 9875, 9875]\, ns$$

In the first case (given in the upper part of Table 3), five additional messages are considered. The adjacency matrix, representing the graph with the original tasks fixed at their start time and additional tasks T_{13} to T_{20} corresponding to the additional messages, is shown in Figure 6.

The schedule of the first case is shown on the left side of Figure 7 (the original tasks are represented by white rectangles and additional tasks by grey rectangles).

6.3.2 Adding New Nodes and Messages

In the second case, the new node 6 (CP-1616 device) is joined to node 3, with a link delay of $T_{LD} = 4895$ ns in both directions. Additional messages are given in the lower part of

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	T_{16}	T_{17}	T_{18}	T_{19}	T_{20}
T_1	$-\infty$	5000	8841	17800	22675	15721	24680	5000	5000	5000	9875	9875	5000	$-\infty$	5000	5000	5000	5000	$-\infty$	$-\infty$
T_2	-5000	$-\infty$	3841	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_3	-8841	-4120	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_4	-17800	$-\infty$	$-\infty$	$-\infty$	4875	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_5	-22675	$-\infty$	$-\infty$	-8120	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_6	-15721	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_7	-24680	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_8	-5000	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	0	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_9	-5000	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	4875	4875	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_{10}	-5000	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-2200	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_{11}	-9875	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-4875	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_{12}	-9875	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-4875	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_{13}	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	4895	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_{14}	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_{15}	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_{16}	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_{17}	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
T_{18}	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	4895	$-\infty$
T_{19}	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	4875
T_{20}	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$

Figure 6: Adjacency matrix with original and additional tasks

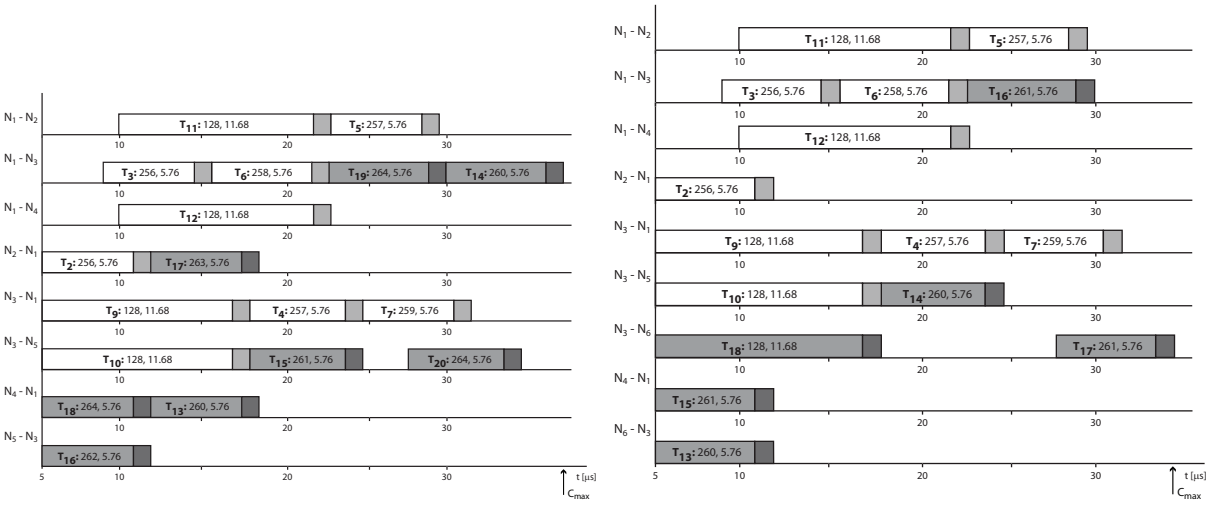

Figure 7: Schedules with additional messages and additional nodes (left - new messages, right - new nodes and new messages)

Table 3. The messages are added in the same way as for the first case. The schedule of the second case is shown on the right side of Figure 7.

7 Conclusions

Our work related to the time-triggered protocols has the following contributions and consequences:

1. formulation of a project scheduling problem with take-give resources,
2. solution of the problem by efficient heuristic,
3. a formal proof of the time symmetry mapping,
4. formulation of the problem as an instance of RCPS/TC, while creating a graph of tasks,

5. evaluation of the proposed solution on benchmarks, with up to 1000 tasks,

6. solution and evaluation of three different application problems (prolongation of the computation time available for the controller application, retransmission of the synchronization message without hold-up in the switch, and addition of new messages into the original schedule),

Further, we have used similar scheduling approach to IEEE 802.15.4/ZigBee beacon-enabled cluster-tree WSNs with the following contributions:

1. A formulation of the scheduling problem by a cyclic extension of RCPS/TC. Using this formulation, the users are not restricted to a particular implementation but they can make a similar extension to any of the algorithms solving this type of problem.

2. A solution of cyclic extension of RCPS/TC by an Integer Linear Programming (ILP), where a grouping of Guaranteed Time Slots (GTS) leads to very efficient ILP model having a few decision variables.

3. An application of this methodology to a specific case of IEEE 802.15.4/ZigBee cluster-tree WSNs.

References

[1] P. Brucker, A. Drex, R. Möhring, K. Neumann, and E. Pesch, "Resource-constrained project scheduling: Notation, classification, models, and methods," **European Journal of Operational Research**, vol. 112, no. 1, pp. 3–41, 1999.

[2] K. Neumann, C. Schwindt, and J. Zimmermann, **Project Scheduling with Time Windows and Scarce Resources**. Springer, 2003.

[3] P. Pedreiras and L. Almeida, "The FTT-Ethernet protocol: Merging flexibility, timeliness and efficiency," in **Proceedings of the** 14^{th} **Euromicro Conference on Real-Time Systems (ECRTS'02)**, 2002.

[4] J.-D. Decotignie, "A perspective on Ethernet as a fieldbus," in 4^{th} **Int. Conference on Fieldbus Systems and Their Applications (FeT'01)**, 2001.

[5] R. A. d. M. Valentim, A. H. F. Morais, G. B. Brandao, and A. M. G. Guerreiro, "A Performance Analysis of the Ethernet Nets for Applications in Real-Time: IEEE 802.3 and 802.3 1Q," in **2008 6TH IEEE INTERNATIONAL CONFERENCE ON INDUSTRIAL INFORMATICS, VOLS 1-3**, IEEE. IEEE, 2008, Proceedings Paper, pp. 919–924.

[6] S.-K. Kweon, K. G. Shin, and G. Workman, "Achieving real-time communication over Ethernet with adaptive traffic smoothing," 6^{th} **IEEE Real Time Technology and Applications Symposium (RTAS'00)**, vol. 00, p. 90, 2000.

[7] L. L. Bello, G. A. Kaczynski, and O. Mirabella, "Improving the real-time behavior of Ethernet networks using traffic smoothing," **IEEE Trans. Industrial Informatics**, vol. 1, no. 3, pp. 151–161, 2005.

[8] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, "The time-triggered Ethernet (TTE) design," 8^{th} **IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)**, vol. 00, pp. 22–33, 2005.

[9] J. Jasperneite, J. Imtiaz, M. Schumacher, and K. Weber, "A proposal for a generic real-time Ethernet system," **IEEE Trans. on Industrial Informatics**, vol. 5, no. 2, pp. 75–85, May 2009.

[10] O. D. D. S. (DDSIG), "Data-distribution service for real-time systems (DDS)," http://portals.omg.org/dds.

[11] **Ethernet Powerlink V2.0, Communication Profile Specification, Ethernet Powerlink Standardization Group**.

[12] L. Seno, S. Vitturi, and C. Zunino, "Analysis of Ethernet powerlink wireless extensions based on the IEEE 802.11 wlan," **IEEE Trans. on Industrial Informatics**, vol. 5, no. 2, pp. 86–98, May 2009.

[13] J.-L. Scharbarg, F. Ridouard, and C. Fraboul, "A probabilistic analysis of end-to-end delays on an AFDX avionic network," **IEEE Trans. on Industrial Informatics**, vol. 5, no. 1, pp. 38–49, Feb. 2009.

[14] IEC committee SC65C, **Application Layer protocol for decentralized periphery and distributed automation, version 2.2, Specification for PROFINET, IEC 61158-6-10/FDIS**. IEC, October 2007.

[15] ——, **Application Layer services for decentralized periphery and distributed automation, version 2.2, Specification for PROFINET, IEC 61158-6-10/FDIS**. IEC, October 2007.

[16] J. Jasperneite and J. Feld, "Profinet: an integration platform for heterogeneous industrial communication systems," in **10th IEEE Conference on Emerging Technologies and Factory Automation**, vol. 1, Sept. 2005.

[17] Siemens Simatic, "Isochrone Mode, Function Manual. 03/2006," document number A5E00223279-02.

[18] U. Lauther, "An extremely fast, exact algorithm for finding shortest paths in static networks with geographical background," **GI-Tage**, vol. 22, pp. 219–230, 2004.

[19] F. Dopatka and R. Wismueller, "A top-down approach for realtime industrial-Ethernet networks using edge-coloring of conflict-multigraphs," in **International Symposium on Power Electronics, El. Drives, Automation and Motion**, 2006.

[20] W. Herroelen, B. D. Reyck, and E. Demeulemeester, "Resource-constrained project scheduling : A survey of recent developments," **Computers and operations research**, vol. 25, no. 4, pp. 279–302, 1998, Elsevier.

[21] C. Schwindt, **Resource Allocation in Project Management**. Springer, 2005.

[22] B. Franck, K. Neumann, and C. Schwindt, "Truncated branch–and–bound, schedule-construction, and schedule-improvement procedures for resource-constrained project scheduling," **OR Spektrum**, vol. 23, no. 3, pp. 297–324, August 2001.

[23] A. Cesta, A. Oddi, and S. F. Smith, "A constraint-based method for project scheduling with time windows," **Journal of Heuristics**, vol. 8, no. 1, pp. 109–136, 2002.

[24] T. B. Smith, "An effective algorithm for project scheduling with arbitrary temporal constraints," in **Proceedings of the 19 th National Conference on Artificial Intelligence. (2004)**. San Jose, California, USA: AAAI Press, Menlo Park, California, 2004, pp. 544–549.

[25] J. Terada, H. Vo, and D. Joslin, "Combining genetic algorithms with squeaky-wheel optimization," in **GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation**. New York, NY, USA: ACM Press, 2006, pp. 1329–1336.

[26] A. Mascis and D. Pacciarelli, "Job-shop scheduling with blocking and no-wait constraints," **European Journal of Operational Research**, vol. 143, no. 3, pp. 498 – 517, 2002.

[27] P. Brucker and T. Kampmeyer, "Cyclic job shop scheduling problems with blocking," **Annals of Operations Research**, vol. 159, no. 1, pp. 161–181, 2008.

[28] P. Laborie, "Algorithms for propagating resource constraints in ai planning and scheduling: existing approaches and new results," **Artif. Intell.**, vol. 143, no. 2, pp. 151–188, 2003.

[29] R. Seifert, **The Switch Book**. Wiley Computer Publishing, 2000.

[30] M. Popp, **Industrial Communication with PROFINET**. PROFIBUS Nutzerorganisation, 2007.

[31] **Simatic Net CP1616/1604 Operating Instructions**, Siemens, 01 2009, document No. C79000-G8976-C218-03, available online.

[32] **ET 200S distributed I/O Interface module IM151-3 PN HIGH SPEED**, Siemens, 07 2009, document No. A5E01584179-02, available online.

[33] **Sinamics S120 Drive functions**, Siemens, 11 2009, document No. 6SL3097-4AB00-0BP0, available online.

[34] IEEE, **802.3 Standard: Part 3 – Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications**. The Institute of Electrical and Electronics Engineers, Inc., New York, 2005.

[35] B. Roy, "Graphes et ordonnancement," **Revue Francaise de Recherche Opérationnelle**, vol. 4, no. 25, pp. 323–333, 1962.

[36] Z. Hanzálek and P. Šůcha, "Time symmetry of project scheduling with time windows and take-give resources," 4th Multidisciplinary International Scheduling Conference: Theory and Applications. Dublin. August, 2009.

8 Doc. Dr. Ing. Zdeněk Hanzálek

Zdeněk Hanzálek graduated in Electrical Engineering at the Czech Technical University (CTU) in Prague in 1990. He obtained his PhD degree in Industrial Informatics from the Universite Paul Sabatier Toulouse and PhD degree in Control Engineering from the CTU. He worked on optimization of parallel algorithms at LAAS CNRS - Laboratoire d'Analyse et d'Architecture des Systemes in Toulouse (1992 to 1997) and on discrete event dynamic systems at LAG INPG - Institut National Polytechnique de Grenoble (1998 to 2000). From 2005, he holds a position of Associated Professor at the CTU, where he founded and coordinated the Industrial Informatics Group focusing on scheduling, combinatorial optimization algorithms, real-time control systems and industrial communication protocols. From 2011, he serves as Senior Manager of newly established Mechatronics group at Porsche Engineering Services s.r.o. in Prague. He has been involved in many international research projects (e.g. ARTIST2, FRESCOR, OCERA) and industrial contracts (e.g. Skoda, UniControls, UNIS, AZD, Volkswagen, Rockwell, Air Navigation Services). Zdenek is currently a deputy head at the Department of Control Engineering at CTU, a head of Industrial Informatics Group. He is teaching master course on Combinatorial Optimization with 120 students per year. In 2011 Zdeněk got "Award for an Excellent Research Achievement" from the rector of the Czech Technical University.

Recent journal papers:

- Čapek, R. - Šůcha, P. - Hanzálek, Z.: Production Scheduling with Alternative Process Plans, European Journal of Operational Research, article in press, Elsevier, doi:10.1016/j.ejor.2011.09.018.
- Špinka, O. - Holub, O. - Hanzálek, Z.: Low-Cost Reconfigurable Control System for Small UAVs. IEEE Transactions on Industrial Electronics, Volume 58, Number 3, Pages 880-889, March 2011 doi: 10.1109/TIE.2009.2030827.
- Sojka, M.- Pisa,P. - Faggioli, D. - Cucinotta, T. - Checconi, F. - Hanzalek, Z. - Lipari, G.: Modular Software Architecture for Flexible Reservation Mechanisms on Heterogeneous Resources, Journal of Systems Architecture, 2011, vol. 57, no. 4, p. 366-382, doi:10.1016/j.sysarc.2011.02.005, Elsevier.
- Kelbel, J. - Hanzálek, Z.: Solving production scheduling with earliness/tardiness penalties by constraint programming. Journal of Intelligent Manufacturing, 2011, vol. 22, no. 4, p. 553-562, doi 10.1007/s10845-009-0318-2, Springer.
- Šůcha, P., Hanzálek, Z.: A Cyclic Scheduling Problem with an Undetermined Number of Parallel Identical Processors. Computational Optimization and Applications, Volume 48, Number 1, p. 71-90, January 2011, doi: 10.1007/s10589-009-9239-4, Springer.
- Waszniowski, L. - Hanzálek, Z. - Doubrava, J.: Aircraft Control System Validation via Hardware-in-the Loop Simulation, Journal of Aircraft, vol. 48, issue: 4, Pages: 1466-1468 , July-August 2011, doi: 10.2514/1.C031229, AIAA.

- Trdlička, J. - Hanzálek, Z.: Distributed Algorithm for Real-Time Energy Optimal Routing based on Dual Decomposition of Linear Programming. International Journal of Distributed Sensor Networks, Vol. 2012, 13 pages, 2012.

- Hanzálek, Z. - Burget, P. - Šůcha, P.: Profinet IO IRT Message Scheduling with Temporal Constraints. IEEE Transactions on Industrial Informatics, Volume 6, Number 3, Pages 369 - 380, August 2010.

- Hanzálek, Z. - Jurčík, P.: Energy efficient scheduling for cluster-tree Wireless Sensor Networks with time-bounded data flows: application to IEEE 802.15.4/ZigBee. IEEE Transactions on Industrial Informatics. doi: 10.1109/TII.2010.2050144, Volume 6, Number 3, Pages 438 - 450, August 2010.

- Špinka, O. - Akesson, J. - Hanzálek, Z. - Arzen, K.: Open Physical Models in Control Engineering Education. International Journal of Electrical Engineering Education, 2010, vol. 47, no. 4, p. 448-459, Manchester University Press.

- Trdlička, J. - Hanzálek, Z.: Distributed Multi-Commodity Network Flow Algorithm for Energy Optimal Routing in Wireless Sensor Networks. Radioengineering. 2010, vol. 2010, no. 4, p. 579-588. ISSN 1210-2512.

- Waszniowski, L. - Krákora, J. - Hanzálek, Z.: Case Study on Distributed and Fault Tolerant System Modelling Based on Timed Automata. Journal of Systems and Software. Volume 82, Issue 10, October 2009, Pages 1678-1694, Elsevier.

- Krákora, J. - Hanzálek, Z.: FPGA Based Tester Tool for Hybrid Real-Time Systems. Microprocessors and Microsystems - Embedded Hardware Design. November 2008, vol. 32, no. 8, p. 447-459, doi:10.1016/j.micpro.2008.07.003, Elsevier Science.

- Šůcha, P., Hanzálek, Z.: Deadline Constrained Cyclic Scheduling on Pipelined Dedicated Processors Considering Multiprocessor Tasks and Changeover Times, Mathematical and Computer Modelling, Volume 47, Issues 9-10, May 2008, p. 925-942, doi:10.1016/j.mcm.2007.05.009, Pergamon-Elsevier Science.

- Waszniowski, L. - Hanzálek, Z.: Formal Verification of Multitasking Applications Based on Timed Automata Model, Real-Time Systems, Volume 38, Number 1, January, 2008, p. 39-65, doi: 10.1007/s11241-007-9036-z, Springer.

- Šůcha, P., Hanzálek, Z., Heřmánek, A., Schier, J.: Scheduling of Iterative Algorithms with Matrix Operations for Efficient FPGA Design - Implementation of Finite Interval Constant Modulus Algorithm, The Journal of VLSI Signal Processing, Volume 46, Number 1, January, 2007, p. 35-53, doi:10.1007/s11265-006-0004-y, Springer.

More than 50 conference papers and 3 chapters in monographs.

Selected projects:

- ADORES - ADaptive scheduling and Optimization algorithms for distributed Real-time Embedded Systems - GACR P103/12/1994, 2012-15
- NewTechno - Optimized Production of Electro Contact and Harness Material for Future Vehicles - Eureka OE09004 . Framework for Real-time Embedded Systems based on COntRacts - FP6 IST-034026 FRESCOR
- Modular FLY-BY-WIRE Control System for Light Aircraft - Tandem framework - Ministry of Industry and Trade of the Czech Republic .
- Embedded Systems Design - NoE in FP6 - IST-004527 ARTIST2
- Open Components for Embedded Real-time Applications IST 35102 OCERA
- Global Communication Architecture and Protocols for new QoS services over IPv6 networks - IST 32696 GCAP
- Intensive Fish Culture Optimisation - IST 31080 IFiBO
- Participation in the evolution of the standardization for embedded software for automotive industry - Czech Academy of Sciences

Industry Cooperation and Products:

- European Train Control System - commercialized via AZD to Czech Railways
- ORTE - open source implementation of RTPS communication protocol .
- Train Communication Network in VxWorks - via UniControls Ltd. to Alstom
- TORSCHE Scheduling Toolbox for Matlab - freely available
- Petri Net Matlab toolbox - freely available
- TDCS scheduling tool for IEEE 802.15.4/ZigBee beacon-enabled cluster-tree WSNs
- Personnel Scheduling for Air Navigation Services of the Czech Republic
- QoS management and performance evaluation of CAN-CAN gateway - Volkswagen
- Processor Expert Embedded Real-Time Target for Matlab/Simulink - UNIS

Chair in Conference Committees:

- EUROSYS 2013, European Professional Society on Computer Systems Conference, Prague, April 2013.
- Euromicro Conference on Real-Time Systems 2008, Prague, July 2-4.
- IEEE Workshop on Parallel and Distributed Real-Time Systems 2007, In conjunction with IPDPS 2007, Long Beach, California, USA, March 26-27.

- IEEE Workshop on Parallel and Distributed Real-Time Systems 2006, In conjunction with IPDPS 2006, Island of Rhodes, Greece, April 25-26.

Member in Program Committees:

- RTSS 11, IEEE Real-Time Systems Symposium
- RTAS 11, IEEE Real-Time and Embedded Technology and Applications Symposium
- ECRTS 10, ECRTS 09, ECRTS 07, ECRTS06, ECRTS05, Euromicro Conference on Real-Time Systems
- WPDRTS 06, WPDRTS 05, IEEE Workshop on Parallel and Distributed Real-Time Systems
- WFCS 10, WFCS08, WFCS 06, WFCS 04, WFCS 2000, IEEE International Workshop on Factory Communication Systems
- ETFA07, ETFA 06, ETFA 05, IEEE International Conference on Emerging Technologies and Factory Automation
- RTCSA 2010, IEEE International Conference on Embedded and Real-Time Computing Systems and Applications
- CACSD 06, CACSD 04, IEEE International Symposium on Computer Aided Control Systems Design
- IFAC 05, IFAC world congress

Summer Schools and National Events:

- Embedded systems colloquium Prague 2005, 2007, 2008, 2009, 2010, 2011
- Embedded RTLinux Intro 2007 Summer School, Prague, June 18-22
- ARTIST2 Graduate Course on Embedded Control Systems 2006, Prague, April 3-7
- IFAC Summer School on Control, Computing and Communication 2005, Prague, June 27 - July 1

Lectures:

- Combinatorial Optimization
- Distributed Control Systems
- Scheduling

Graduated PhD. students:

- Supervised by Zdeněk: Přemysl Šůcha, Jan Krákora, Petr Jurčík, Michal Kutil, Ondřej Špinka and Michal Sojka.
- Co-supervised by Zdeněk: Josef Čapek, Miroslav Dub, Libor Waszniowski, Ondřej Dolejš, Martina Svádová a Pavel Burget.