Message Scheduling in Time-Triggered Protocols

Zdeněk Hanzálek
Thanks to: P. Šůcha, P. Jurčík, P. Burget
Czech Technical University in Prague

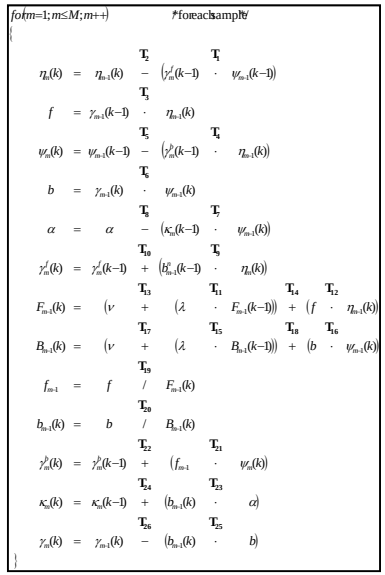
Contents

1. Project Scheduling with Temporal Constraints
2. Profinet IO IRT Message Scheduling
   - Formulation
   - Use of time constraints
   - Experiments
3. Energy efficient scheduling for cluster-tree Wireless Sensor Networks
   - Formulation of the cyclic scheduling problem
   - Experiments
4. Conclusions

Project Scheduling with Temporal Constraints

Original Motivation

- High-level **synthesis**
  - algorithm description by an oriented graph
  - off-line scheduling
  - automatic code generation
- Specific HW architecture – **FPGAs**
  - high degree of parallelism
  - dedicated units (e.g. floating point)
  - pipelining
  - shared memory
  - reconfiguration
- **Optimality -** objective is to find a feasible schedule with minimal C_{max}
  - Branch&Bound algorithm
  - ILP formulation



Project Scheduling with Temporal Constraints

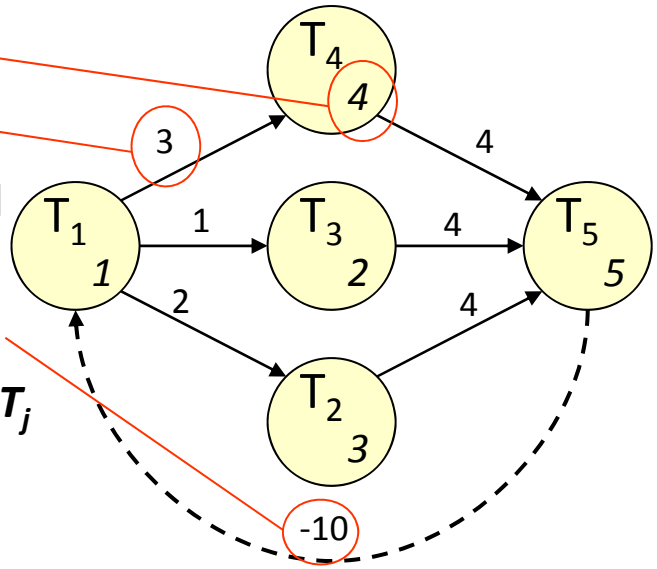
Temporal constrains are sometimes called **generalized precedence constraints** or **positive and negative time lags** or **relative time windows** .

- start times of two tasks are mutually constrained by arc length:

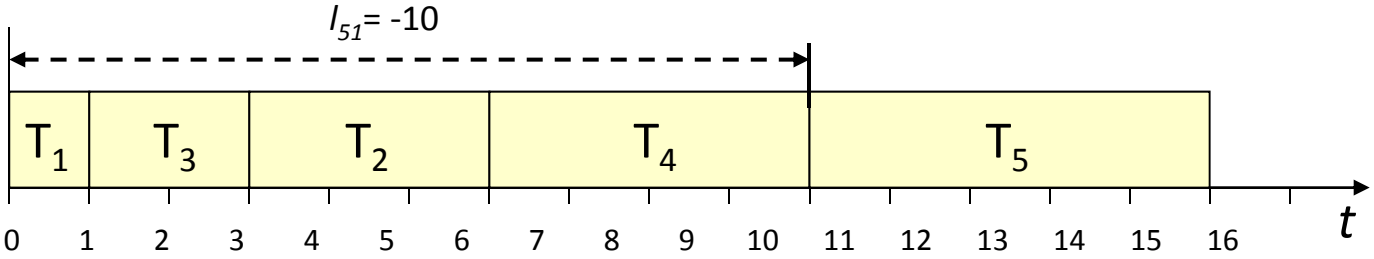
$$s_i + l_{ij} \leq s_j$$

Problem representation by **oriented graph *G***

- **node** - instruction - task T_i with **processing time** p_i
- **forward arc** /positive sign/ - express **precedence delay**, including pipelining or processing time on unconstrained resources
- **backward arc** /negative sign/ - express relative **deadline**, the latest start time s_j of T_j relative to the start time s_j of T_j
- **non-preemptive off-line** scheduling



Optimal feasible schedule on one processor:



Problem Complexity

Decision version of our problem (decide existence of feasible schedule) is NP-complete, since it is P-reducible from Bratley's problem (which is P-reducible from 3-PARTITION problem).

Instance of Bratley's problem ⇒ instance of our problem

Independent tasks

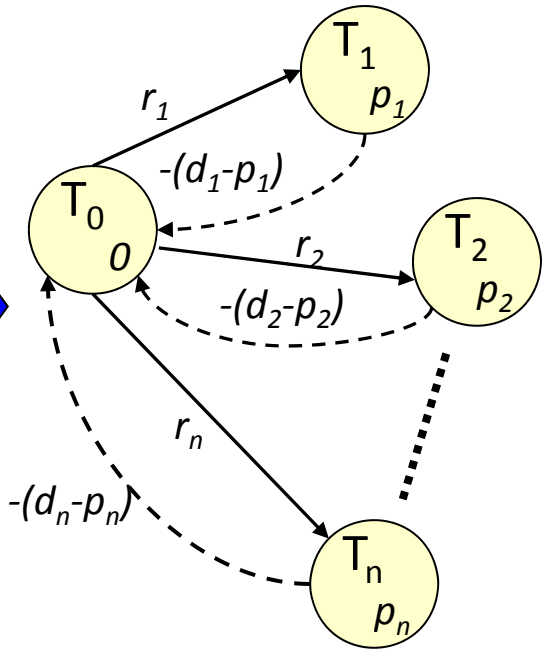
$1 | r_j, d_j | C_{max}$

$r = [r_1, r_2, \ldots, r_n]$

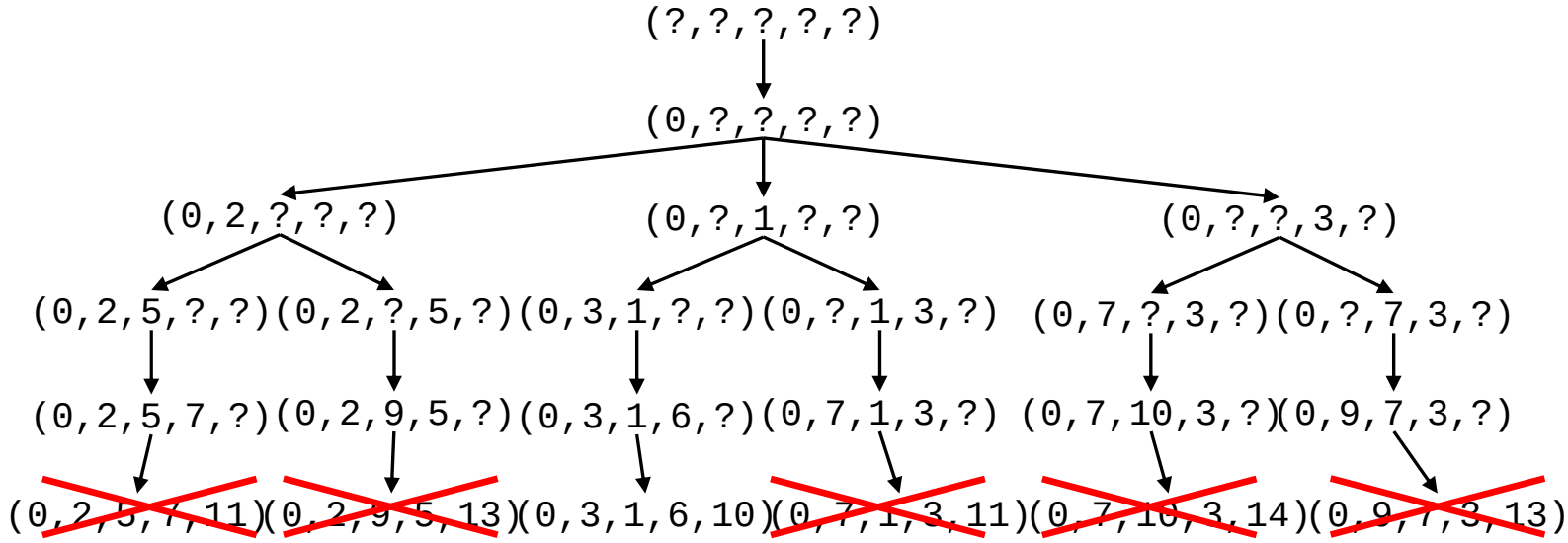
$p = [p_1, p_2, \ldots, p_n]$

$d = [d_1, d_2, \ldots, d_n]$

P-reducible



Search space of systems with time constraints



Complexity of the system **optimization** is the same as the one of the system **verification.**

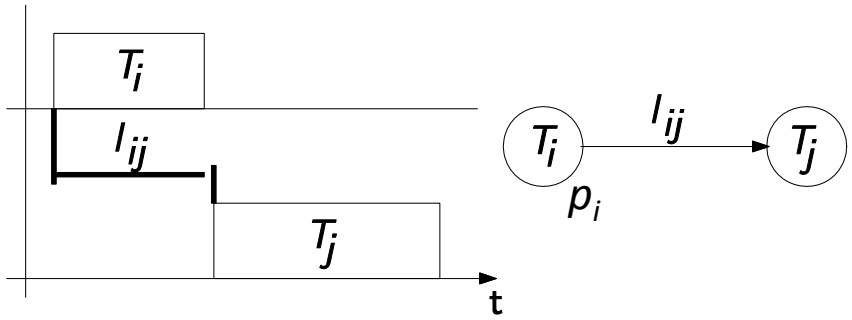
In practice: sub-optimal solution, found in a part of the search tree (e.g. by heuristic algorithm) **has practical value**, but **partial verification** (i.e the one which does not consider all possible behaviors) has none.

It is **easier to synthesize the time-constrained system** than to leave the freedom to the designer and consequently verify its time properties.

Modeling by temporal constraints
Temporal constraints with positive l_{ij}

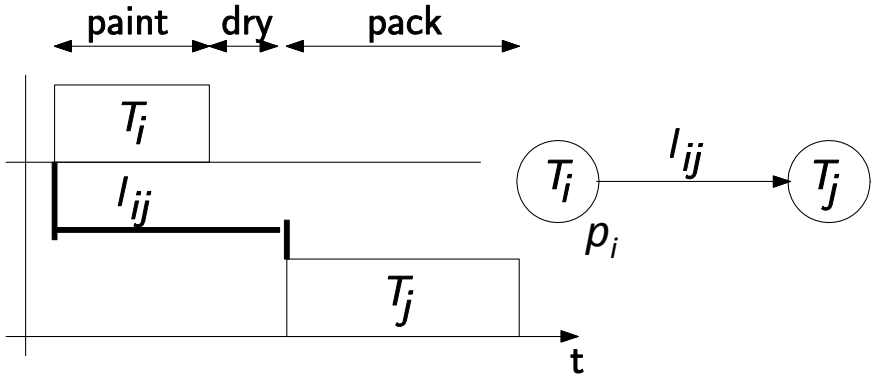
a) $l_{ij} = p_i$

- **"normal" precedence relations**
- execution of the next task may start after execution of the previous task



b1) $l_{ij} > p_i$

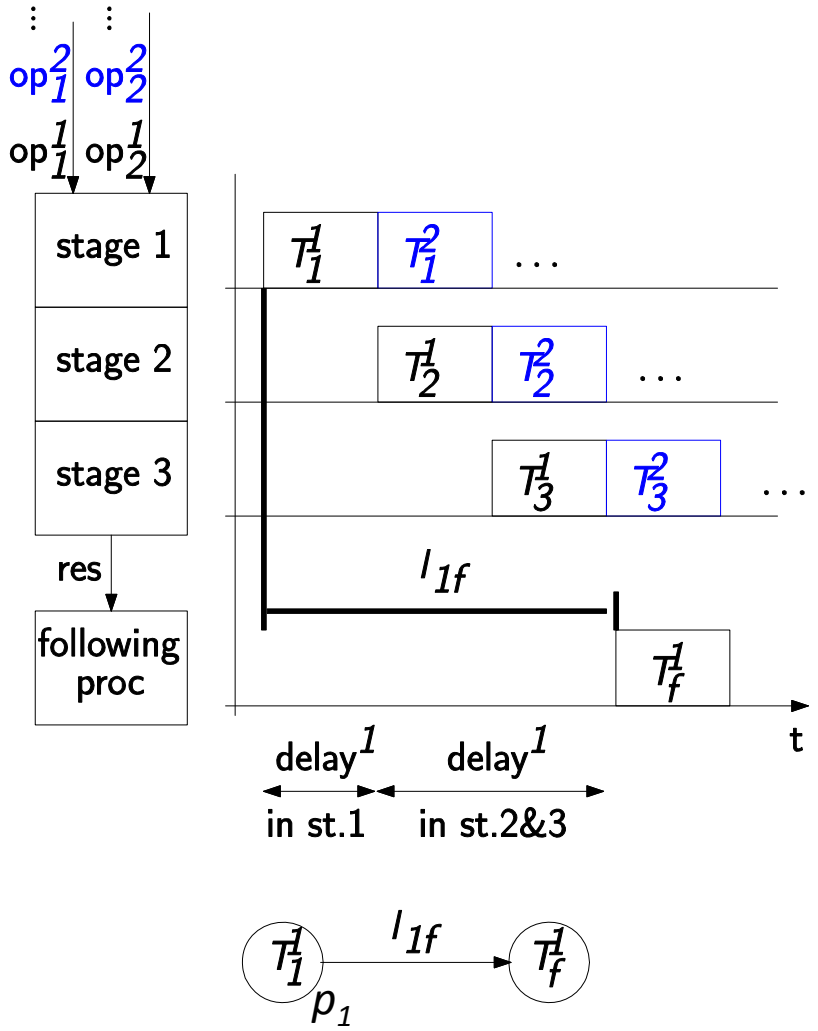
- execution of the next **task may start some time after completion of the previous task**
- example of a dry operation performed in sufficiently large space



Temporal constraints with positive l_{ii}

b2) another example with $l_{ij} > p_i$ - **pipelined ALU**

- We assume the processing time to be equal in all stages
- **Stage 1 reads new operands each p_1**
- **Result is available l_{1f} tics later**
- **Stages 2 and 3 are not modeled** since we have enough of these resources and they are synchronized with stage 1

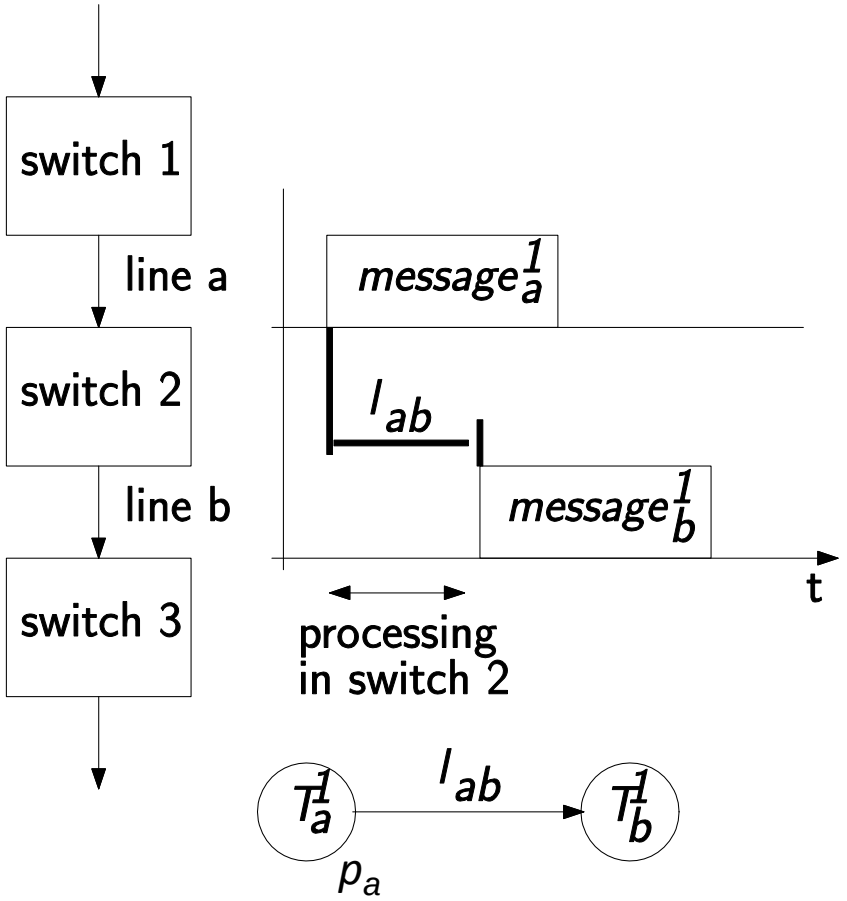


Temporal constraints with positive l_{ii}

c) $0 < l_{ij} < p_i$

Partial results of the previous task may be used to start execution of the following task. E.g. **cut-through mechanism**, where the switch starts transmission on the output port earlier than it receives complete message on the input port

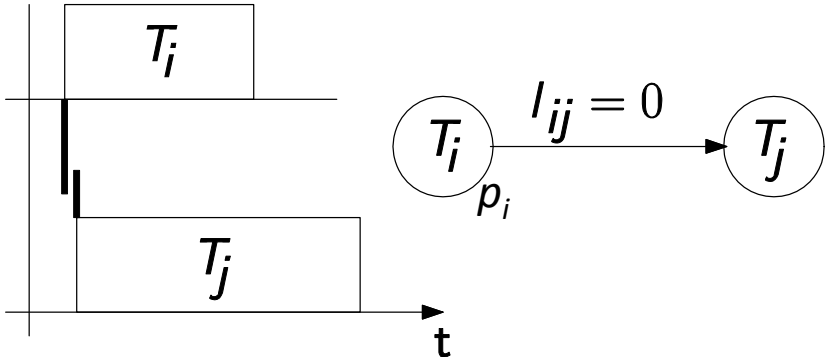
- Resources are communication links
- l_{ab} presents **transmission** (of one bit) **through the switch**
- Different parts of the same message are transmitted by several communication links at the same time



Temporal constraints with zero or negative l_{ij}

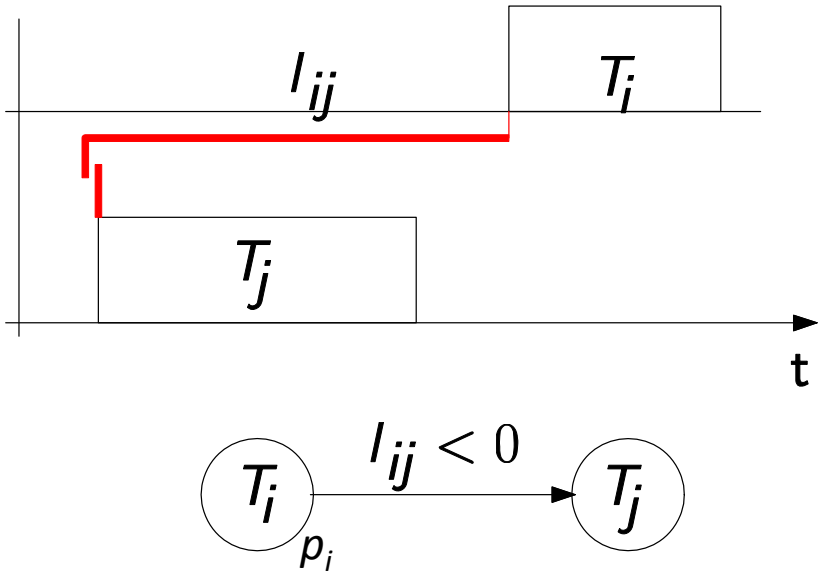
d) $l_{ij} = 0$

- Task T_i has to start earlier or at the same time as T_j



e) $l_{ij} < 0$

- Task T_i has to start earlier or **at the most $|l_{ij}|$ later** than T_j
- It looses the sense of “normal ” precedence relation, since T_j can precede T_i
- It presents **relative deadline** of T_i with respect to start time of T_j



Problem Formulation by ILP

processor constraints ($(n^2-n)/2$ decision variables x_{ij}) … "big M"

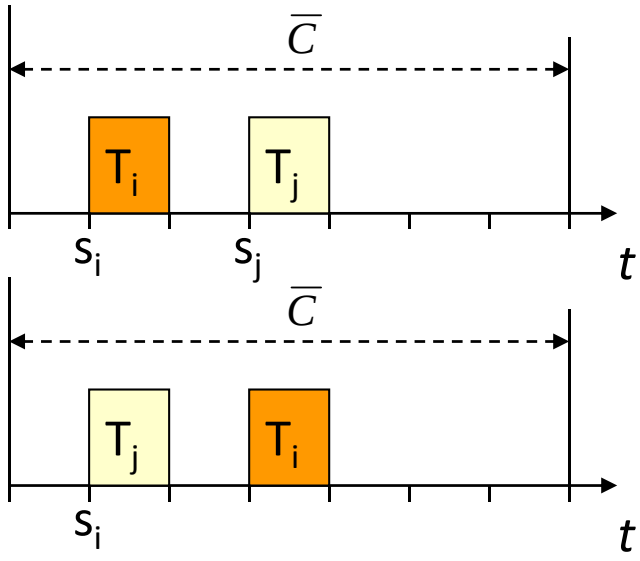
$$\forall i, j \in \langle 1, n \rangle, i < j, \qquad p_j \le s_i - s_j + x_{ij} \cdot \overline{C} \le \overline{C} - p_i$$

a) when T_i precedes T_j (x_{ij}= 1)

$$s_j \ge s_i + p_i$$

b) when T_i succeeds T_j (x_{ij}= 0)

$$s_i \ge s_j + p_j$$

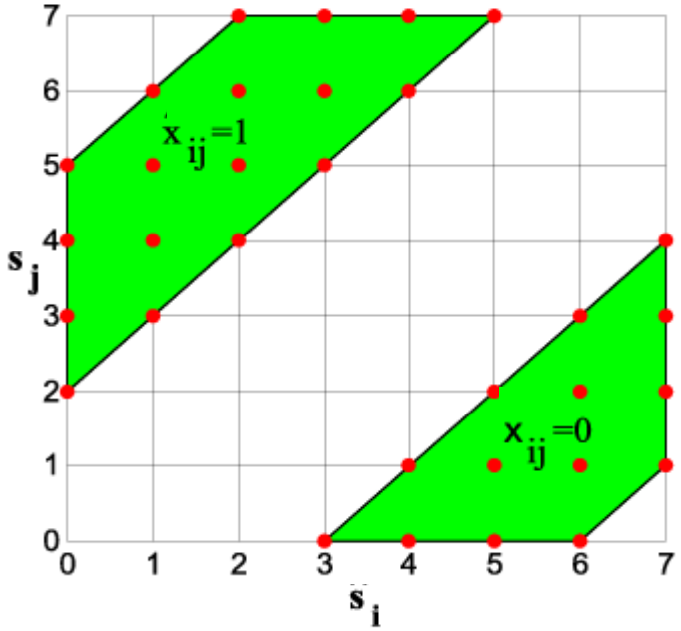


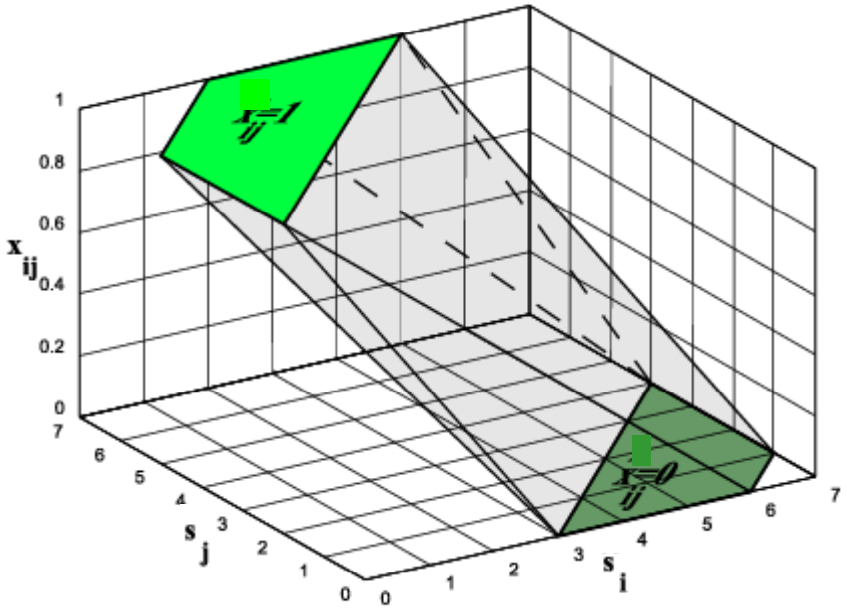
s_j

Processor constraints (cont.) – XOR relation

Example:

T_i and T_j without precedence constraints





ILP program – for m dedicated processors

$$\min C_{\max}$$

subject to:

$$s_j - s_i \geq l_{ij}$$

$$p_j \leq s_i - s_j + \overline{C} \cdot x_{ij} \leq \overline{C} - p_i$$

$$s_i + p_i \leq C_{\max}$$

where:

$$s_i \in \langle 0, \overline{C} - 1 \rangle, \; x_{ij} \in \langle 0,1 \rangle, \; C_{\max} \in \langle 0, \overline{C} \rangle$$

$$s_i, x_{ij} \text{ are integers.}$$

objective function - minimizes makespan

precedence constraint - restriction given by graph G

processor constraints -

- at maximum one task is executed at a given time
- this constraint exists for each couple of tasks allocated to the same processor

PS|temp|C_{max} - Related Work

Resource Constraint Project Scheduling with Time Constraints

- [B.Roy 59] – Project planning – Metra Potencial Method – introduction of **positive and negative time lags**
- [Brucker, P., Drexl, A., Mohring, R., Neumann, K., Pesch, E., 99] Resource-constrained project scheduling: Notation, **classification**, models, and methods
- [K.Neumann, Ch.Schwindt, J. Zimmermann] – Project Scheduling with **Time Windows** and Scarce Resources
- [W.Herroelen, B.D. Reyck, E.Demeulemeester] - Resource-constrained project scheduling : **A survey** of recent developments
- [B.D. deDinechin] - Simplex scheduling: More than lifetime-sensitive **instruction scheduling**
- [R.Alur, D.Dill] - A theory of **timed automata**.
- [K.G. Larsen KG, P. Petterson P, W. Yi] - **UPPAAL** in a nutshell.
  - **guards** and **time invariants** represent positive and negative time lags
  - Difference Bound Matrices (DBM) - symbolic representation of states by clock zones

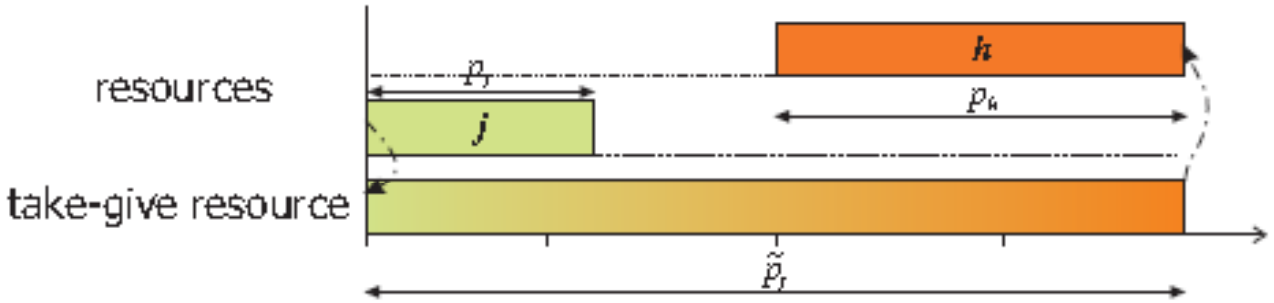
PS|temp|C_{max}

- 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
- **multiprocessor activity** - multiple resources required by one activity
- an event - activity i with $p_i = 0$ is executed on resource k with capacity $R_k = \infty$
- **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
- $\sum_{v=1}^{R_k} z_{ivk} = r_{ik}$ holds for each activity $i \in \mathcal{V}$ and each resource $k \in \mathcal{R}$

Changeover time (sequence dependent setup-up time)

- $o_{ij} \in \mathbb{R}_0^+$ satisfies **triangular inequality**: $o_{ij} \leq o_{il} + o_{lj}$ $\forall (i, l, j) \in \mathcal{V}^3$
- $s_j \geq s_i + p_i + o_{ij}$ holds when **activity** i **precedes activity** j and they are **assigned to the same unit** of give resource: $z_{ivk} = z_{jvk} = 1$
- the inequality holds for both immediate and non-immediate precedence, due to the satisfaction of triangular inequality

Take-give Resources



- occupation i **requires** $a_{ilk} \in \{0, 1\}$ units of take-give resource $k \in \mathcal{Q}$
- occupation i **starts** its execution at s_i and **completes** at $C_i = s_i + p_i$
- take-give resource $k \in \mathcal{Q}$ has **capacity** of $Q_k \in \mathbb{Z}^+$ and $Q_k < \infty$
- **there is a path** from i to l with $d_{il} > 0$
- **take-give assignment** $\tilde{z}_{ivk} \in \{0, 1\}$ is equal to 1 when occupation i is assigned to unit v of take-give resource k, and 0 otherwise
- **changeover time on take-give resources** $\tilde{o}_{ij} \in \mathbb{R}_0^+$

IRS heuristic algorithm – main loop

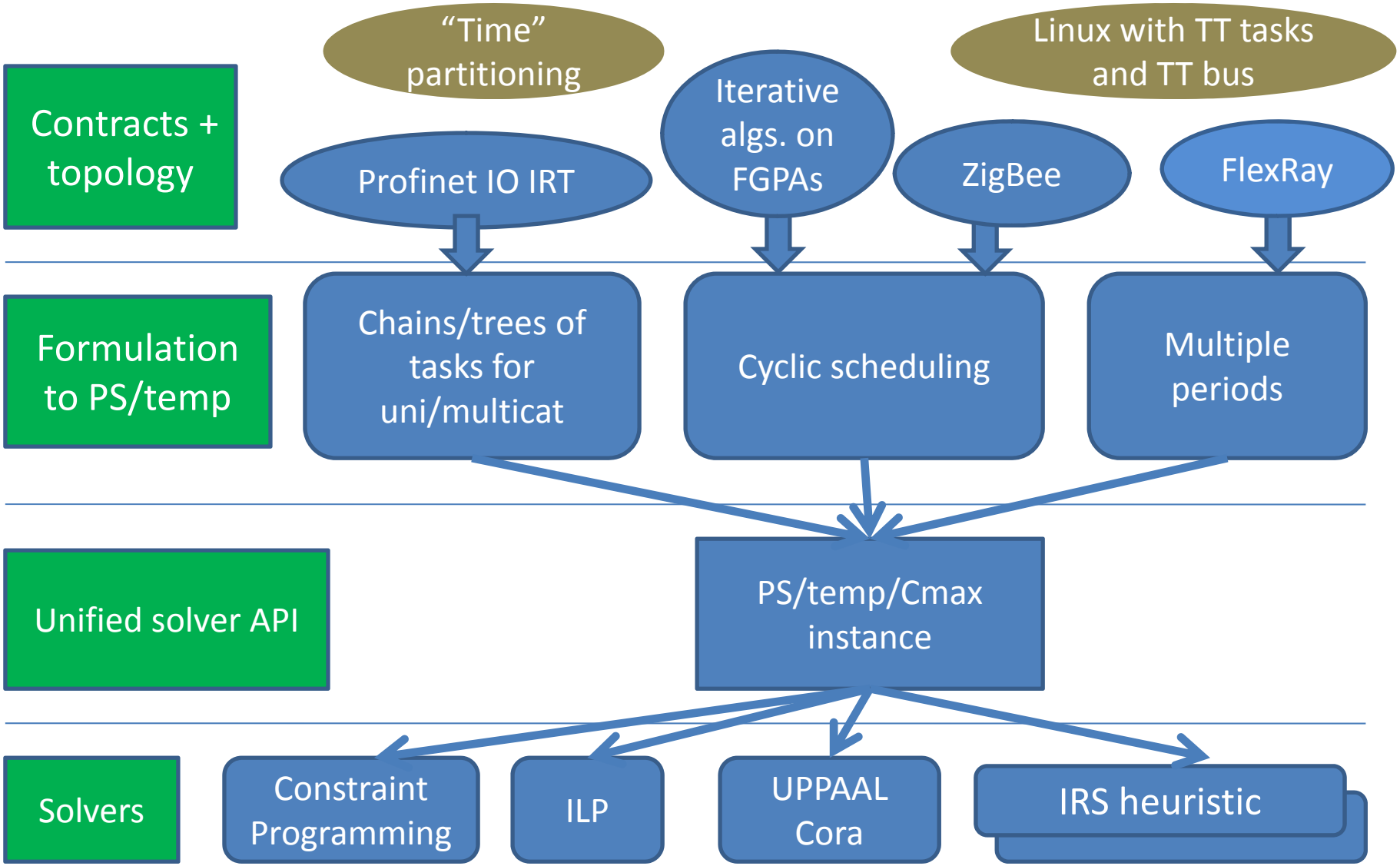
input : an instance $PS|temp, o_{ij}, tg|C_{max}$, $budgetRatio$
output: schedule S^{best}.
Calculate $d_{i,j}$ $\forall (i,j) \in \mathcal{V}^2$;
Calculate LB and UB;
$C = LB$; $S^{best} = \{\}$;
$budget = budgetRatio \cdot n$;
$priority(i) = d_{i,n+1}$ $\forall i \in \mathcal{V}$;
while $LB \leq UB$ **do**
    $S =$ `findSchedule`$(C, priority, budget)$;
    if S *is feasible* **then**
        $S_{left} =$ `shiftLeft`(S); % constraints and order are kept
        $UB = C_{max}(S_{left}) - 1$; $S^{best} = S_{left}$;
    else
        $LB = C + 1$;
    end
    $C = \lceil (LB + UB)/2 \rceil$; % interval bisection method
end

Find feasible schedule for given C

`findSchedule`$(C, priority, budget)$
$s_i = -\infty \ \forall i \in \{0, \ldots, n+1\}$;
$scheduled = \{\}$;
while $budget > 0$ *and* $|scheduled| < n + 2$ **do**
    % choose unscheduled activity with highest priority
    $i = \arg\max_{\forall i \in \mathcal{V}: i \notin scheduled} (priority_i)$;
    $ES_i = max_{\forall j \in \mathcal{V}: i \in scheduled} (s_j + d_{ji})$;
    $LS_i = C - p_i$;
    $s_i =$ `findTimeSlot`(i, ES_i, LS_i);
    % schedule activity i by force and unschedule conflicting activities
    $scheduled =$ `scheduleActivity`$(i, s_i, scheduled)$;
    $budget = budget - 1$;
end

findTimeSlot(i, ES_i, LS_i) finds the earliest s_i with no conflicts on resources. If there is no such room, then $s_i = ES_i$ when being scheduled for the first time otherwise $s_i = s_i^{prev} + 1$.

General approach to TT applications



Profinet IO IRT Message Scheduling

Problem statement

Profinet IO IRT is an Ethernet based hard-real time communication protocol

- uses static schedules for time-critical data
- the schedule is computed during the engineering phase
- the schedules are downloaded into the nodes, each of which contains a special hardware switch
- the schedule breaks the standard forwarding rules intentionally to ensure that no queuing delays occur in the switches

Our objective

- provide documented solution
- formulate the problem in terms of Project Scheduling with temporal constraints
- use temporal constraints, to solve more complex problems than the algorithm being part of Simatic Manager by Siemens

Related work

Ethernet Powerlink, DDS, Ethercat, TT-Ethenet,…

Max Felser: Real-Time Ethernet - industry prospective, Proceedings of the IEEE, 2005

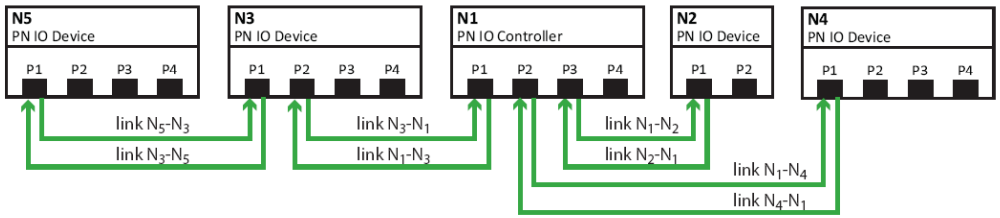
Application Layer protocol for decentralized periphery and distributed automation, Specification for PROFINET, IEC 61158, 2007

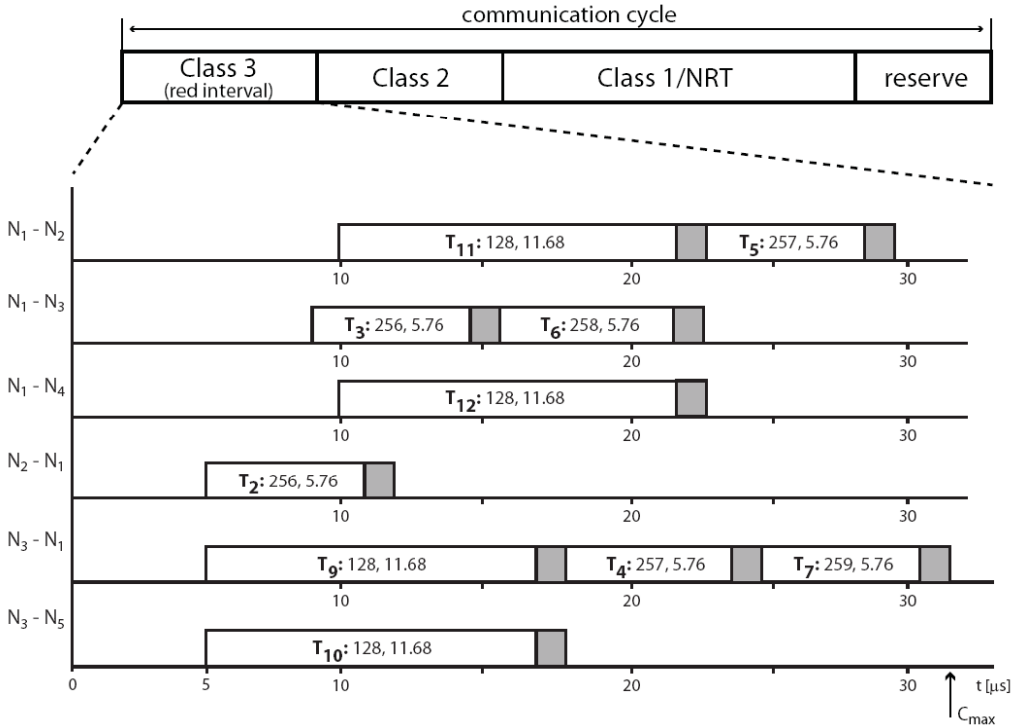
Hermann Kopetz , Gunther Bauer, The time-triggered architecture, Proceedings of the IEEE, 91/1, 2003

Paulo Pedreiras , Luís Almeida: The FTT-Ethernet Protocol: Merging Flexibility, Timeliness and Efficiency, ECRTS'02

Traian Pop, Paul Pop, Petru Eles, Zebo Peng, Alexandru Andrei: Timing analysis of the FlexRay communication protocol, Real-Time Systems,39/1, 2008

Profinet IO Industrial Protocol





Tree topology

- switch integrated in each node (special HW for IRT)
- full duplex

RT Class 3 - red interval - IRT

- highest-priority
- strictly isochronous - Precision Transparent Clock Protocol (PTCP in IEC 61158)
- data are forwarded according to a static communication schedule

Input Parameters of the Scheduling Problem

List of links

- link delay

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

$$T_{LD} = T_{TxD} + T_{CD} + T_{RxD} + T_{ad} + T_{BD}$$

List of messages

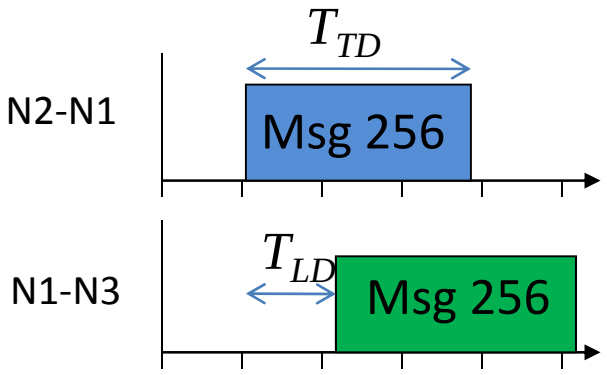
- source
- destination(s)
- transmission delay
- required
  - release date
  - deadline
  - end-to-end delay
- multicast message – used e.g. for synchronization

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

Problem Refinement

Objective is to find a shortest schedule for the red interval based on a network topology/parameters, message parameters and required position in the schedule

- messages on the same link are separated with a minimum inter-message gap (added to transmission delay T_{TD})
- as soon as the first bit of a message is received, it may be forwarded to another link, i.e. if $T_{LD} < T_{TD}$, two nodes may process a different part of the same message at the same time



overlapping precedence relation

Solution of Profinet IO IRT scheduling

Formulation in terms of the Resource Constrained Project Scheduling with Temporal Constraints minimizing the schedule makespan (denoted PS|temp|C_{max})

Tree topology of nodes

- determines the rooting of messages

Unicast message

- chain of tasks executed on dedicated communication links
- chain starts at the source node and ends at the destination node

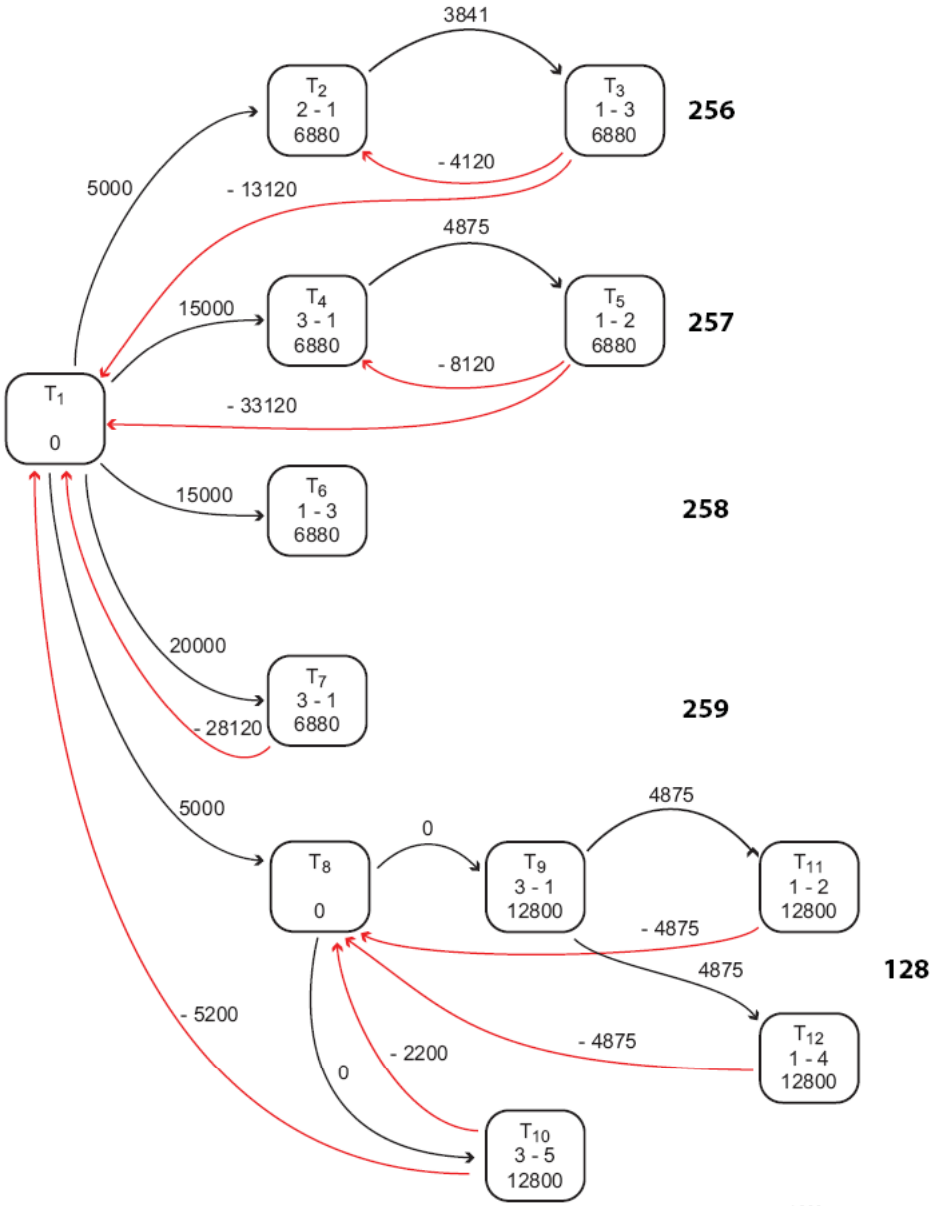
Multicast message

- tree of tasks

Modeling of Profinet IO IRT by PS|temp|C_{max}

Task execution corresponds to a transmission of a message on the respective link

- Transmission delay - processing time equal to T_{TD}
- Line delay - edge with positive weight T_{LD}
- Release date – edge with positive weight from dummy task to source task
- Deadline - edge with negative weight from sink task to dummy task
- Required end-to-end delay - edge with negative weight from sink task to source task



Computational results

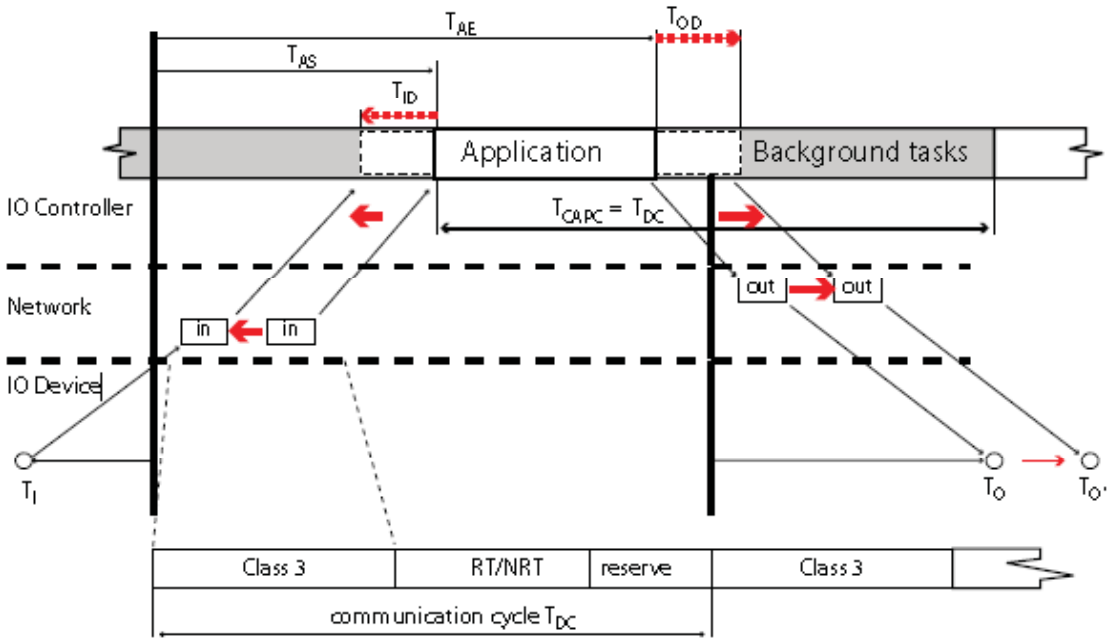
Optimal algorithms (ILP) were used to evaluate performance of heuristics

- complexity is related to the number of conflicting tasks
- heuristics have very small deviation from the optimum

Topology	n	T_C^{ILP}	C_{max}^{ILP}	T_C^{FBS}	C_{max}^{FBS}	T_C^{IRS}	C_{max}^{IRS}	$C_{max}^{Simatic}$
1	142	17.41 s	47.20 μs	60 s	47.20 μs	0.016 s	47.20 μs	47.20 μs
2	292	7200 s	116.00 μs (13.44%)	60 s	154.47 μs	0.156 s	116.00 μs	116.00 μs
3	442	7200 s	150.40 μs (20.70%)	-	-	0.422 s	150.40 μs	150.40 μs
4	962	7200 s	243.18 μs (30.49%)	-	-	4.040 s	219.20 μs	219.20 μs
5	512	7200 s	116.00 μs (16.41%)	-	-	0.733 s	116.00 μs	116.00 μs
6	882	7200 s	155.55 μs (24.44%)	-	-	9.345 s	154.66 μs	150.40 μs

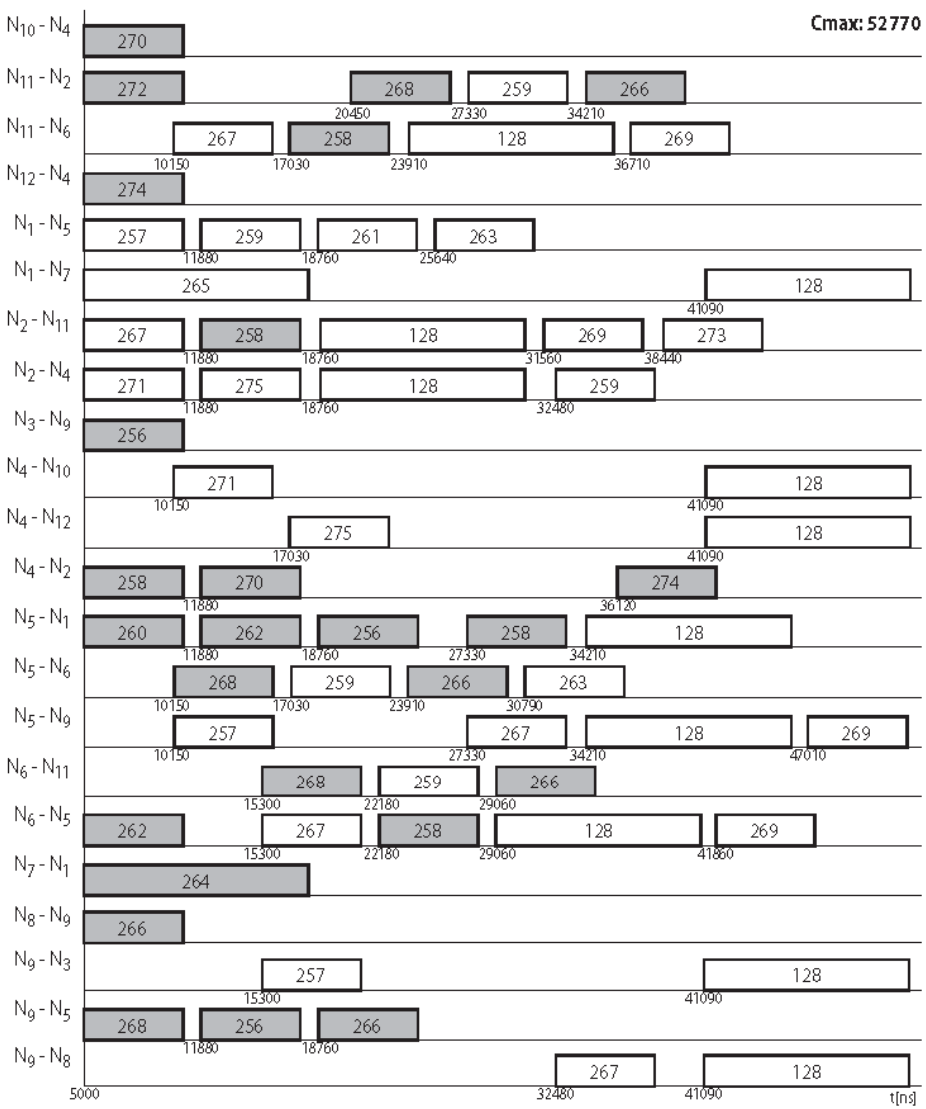
Separation of input and output messages by timing constraints

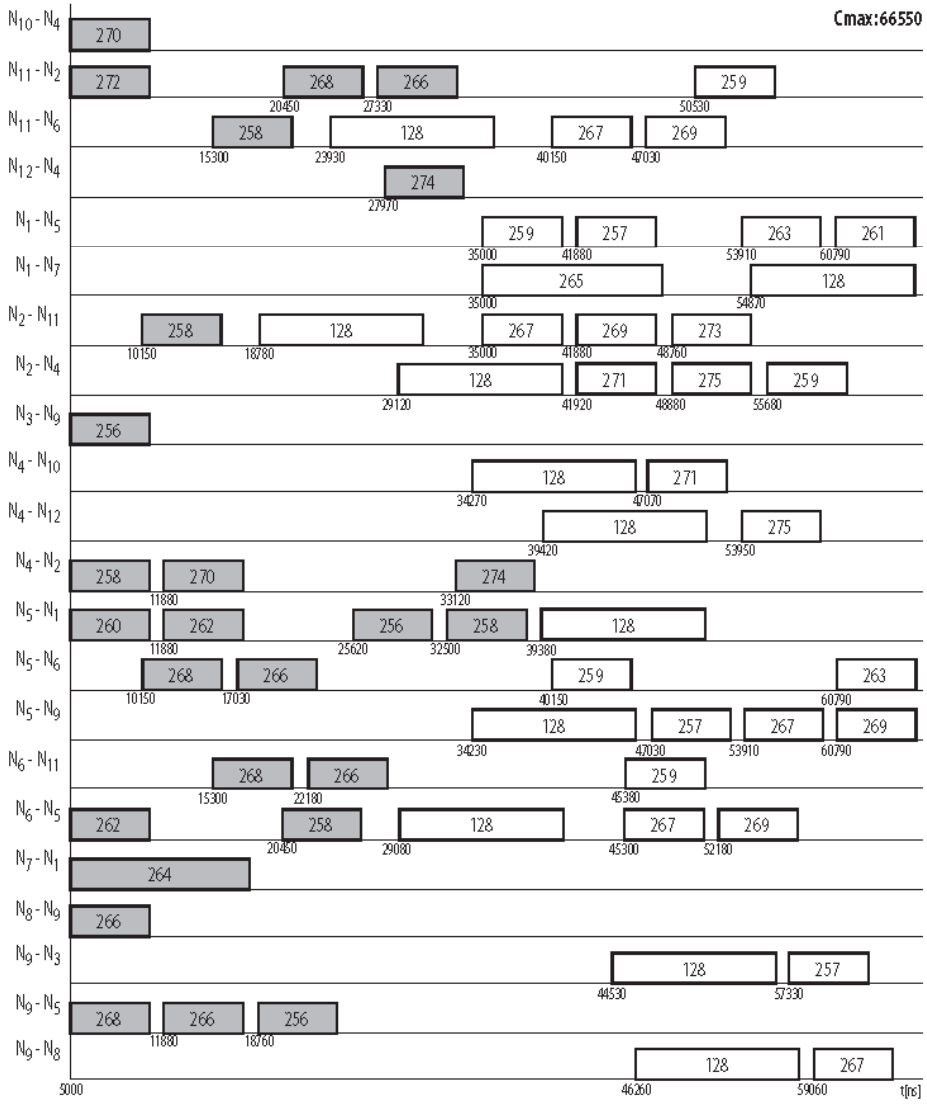
Prolongation of the time available for the controller application



Another example - enforcing immediate retransmission of synchronization messages - using the cut-through mechanism

Schedule of input and output messages without/with time constraints





Rescheduling with time constraints

Users require adaptations which leads to the extension of the system

Running application may be affected by the change

We keep original schedule and we add

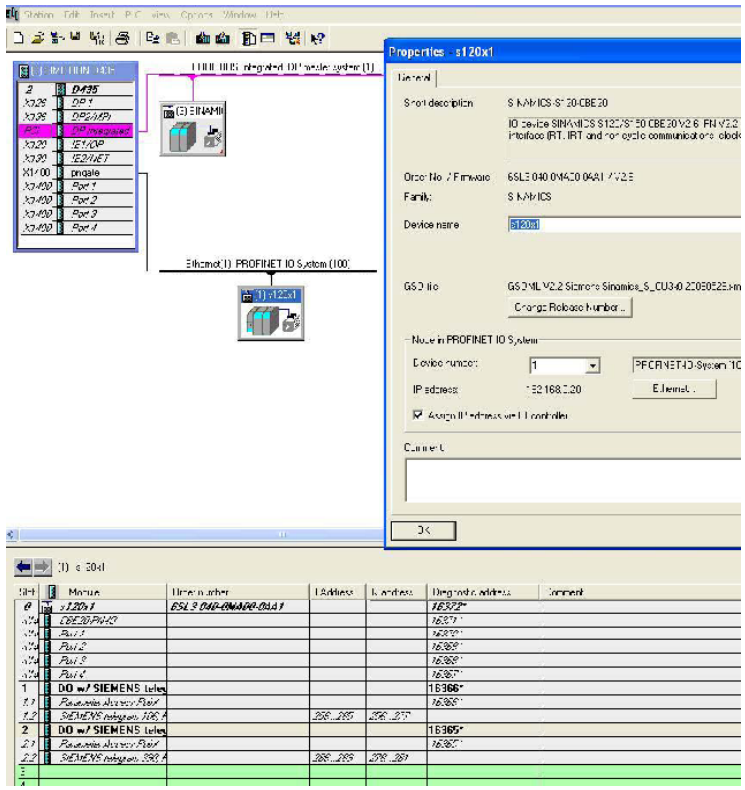
- new messages
- new nodes

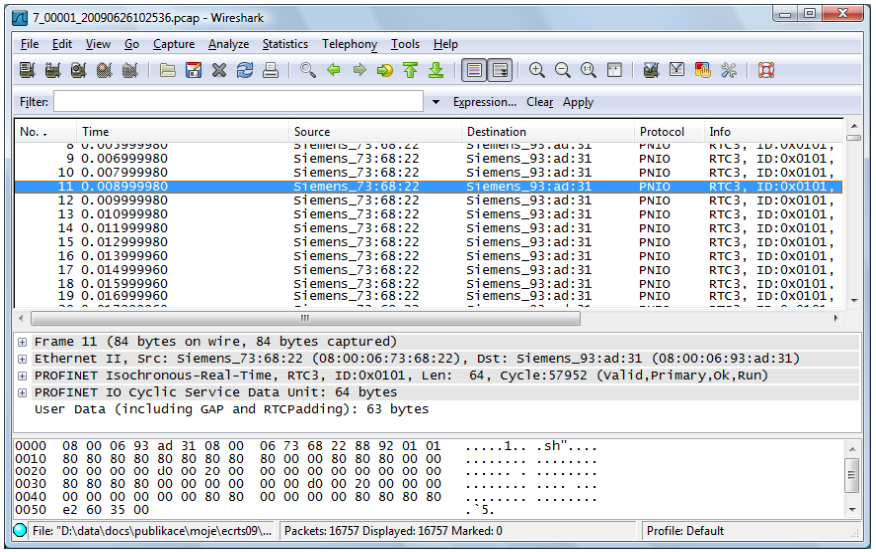
Simple solution – fix position of original task and use the same scheduling algorithms

Topology	$\bar{n}$	T_C^{ILP}	C_{max}^{ILP}	T_C^{IRS}	C_{max}^{IRS}	$\bar{n}$	T_C^{ILP}	C_{max}^{ILP}	T_C^{IRS}	C_{max}^{IRS}
1	152	21 s	47.20 μs	0.064 s	47.20 μs	159	23 s	69.53 μs	0.546 s	-
2	311	101 s	134.06 μs	1.218 s	134.06 μs	329	107 s	148.63 μs	1.731 s	-
3	476	263 s	173.57 μs	3.318 s	173.57 μs	499	277 s	208.78 μs	4.228 s	-
4	1043	1824 s	256.13 μs	28.394 s	256.13 μs	1095	2063 s	334.23 μs	37.986 s	-
5	560	478 s	134.02 μs	5.252 s	134.02 μs	587	519 s	179.53 μs	6.645 s	-
6	966	1746 s	182.18 μs	23.147 s	182.18 μs	1011	1907 s	239.68 μs	30.685 s	-

Experiments on real HW

Simple topology demonstrating the functionality of our scheduling taken by HW configuration tool





The correct communication can be seen in the listing from Wireshark (hardware solution with time stamps) - displays the communication record of messages repeating every cycle

Energy efficient scheduling for cluster-tree Wireless Sensor Networks with time-bounded data flows: application to IEEE 802.15.4/ZigBee

Problem statement

To find a static schedule for static WSN which specifies
when the clusters are active while:

- communicating all data flows
- avoiding possible cluster collisions
- meeting all data flows' end-to-end deadlines
- minimize the energy consumption of the nodes

Related work

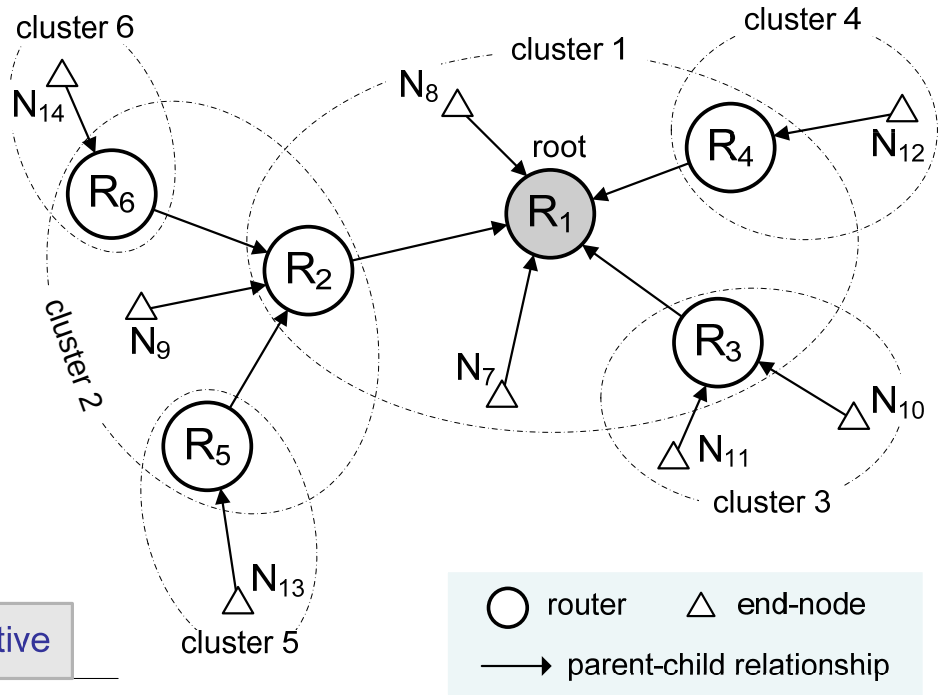
Cluster-Tree Sensor Networks

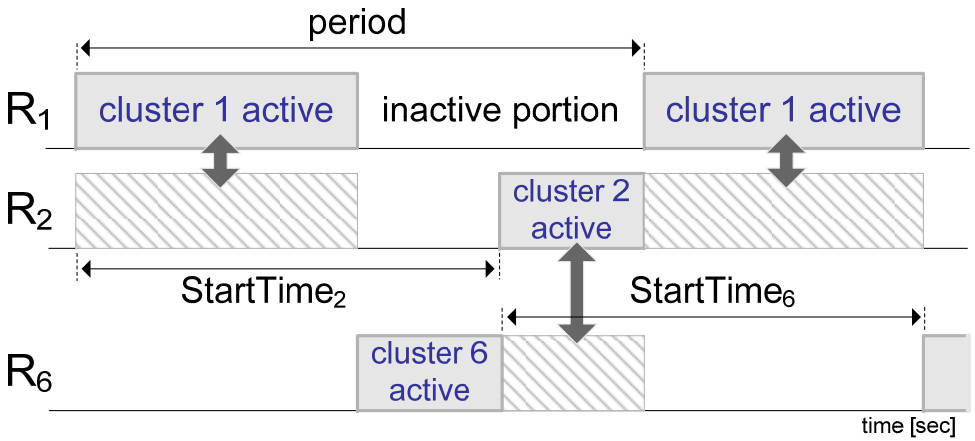
- Koubaa, A. Cunha, M. Alves, and E. Tovar, "TDBS: a time division beacon scheduling mechanism for ZigBee cluster-tree wireless sensor networks," Real-Time Systems Journal, 2008.
- P. Jurcik, R. Severino, A. Koubaa, M. Alves, and E. Tovar, "Real-Time Communications over Cluster-Tree Sensor Networks with Mobile Sink Behaviour," RTCSA 2008.

Our work addresses the problem of multiple data flows with end-to-end deadlines while minimizing the energy consumption of nodes.

Cluster-tree topology model

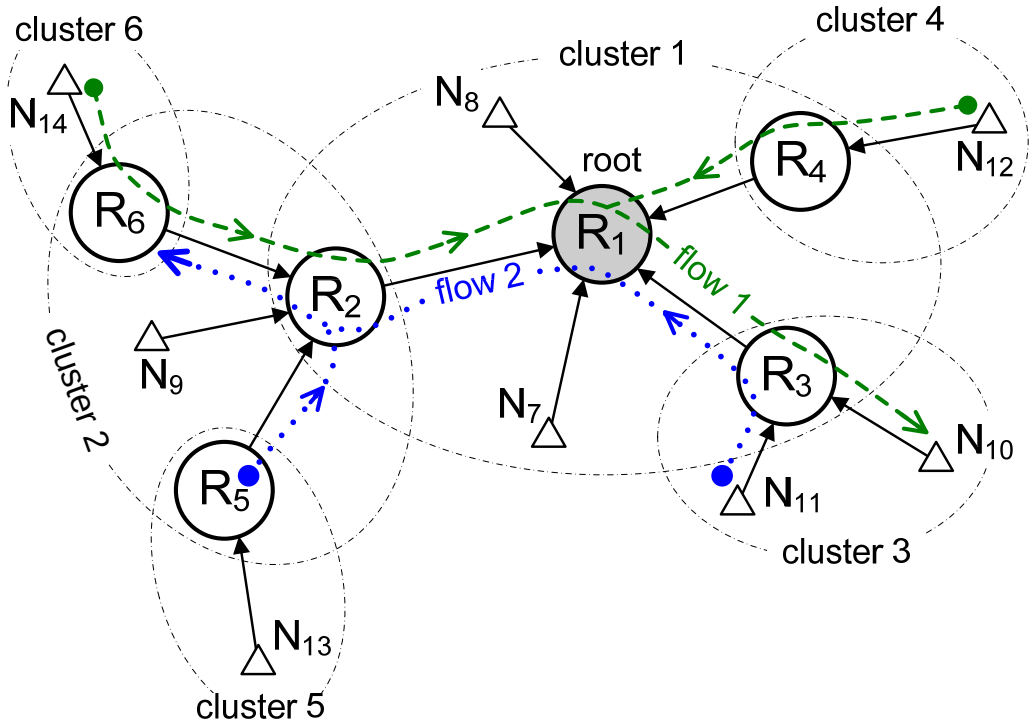
- static wireless sensor networks (WSNs)
  - cluster-tree topology
    - in-tree
    - deterministic routing
- cluster
  - star sub-network
  - cluster-head
  - active & inactive portions
- collision domains





Data flow model

- data flows
  - predefined
  - time-bounded
  - multi-source mono-sink
  - periodic data
  - parameters: [flow 1]
    - sources [N14, N12]
    - sink [N10]
    - required period [0.4]
    - sample size [64]
    - end-to-end deadline [0.2, 0.1]



- communication errors
  - bounded number of retransmissions

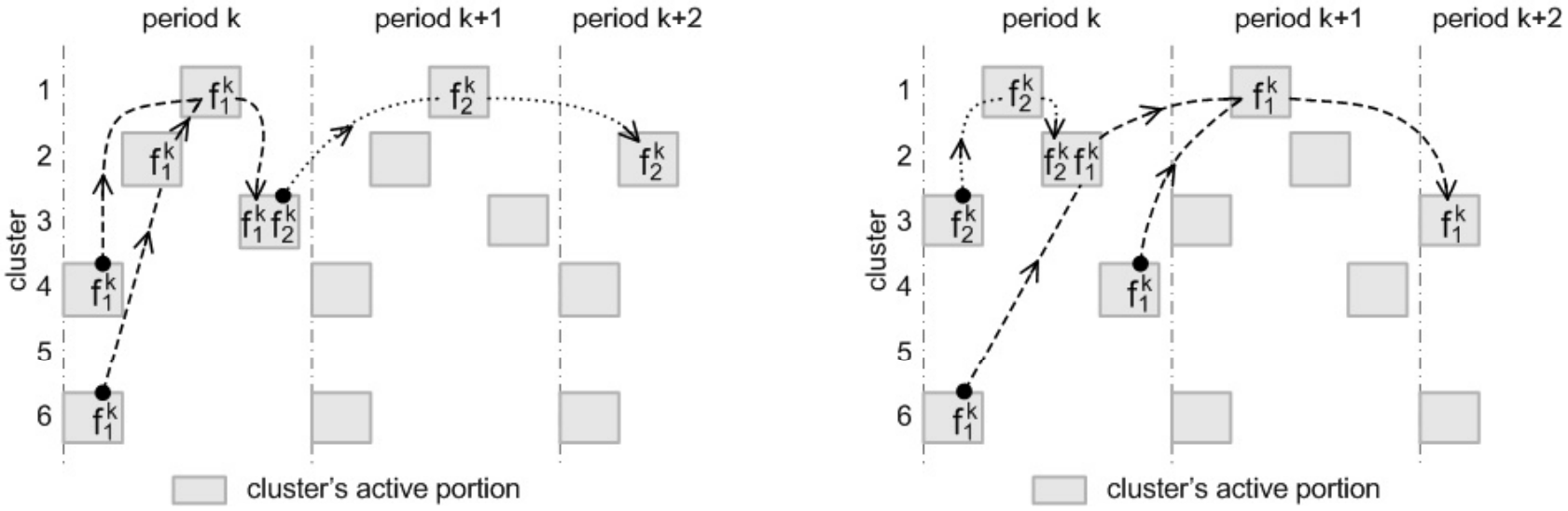
Cyclic nature of the scheduling problem

One wave of the flow goes over several periods

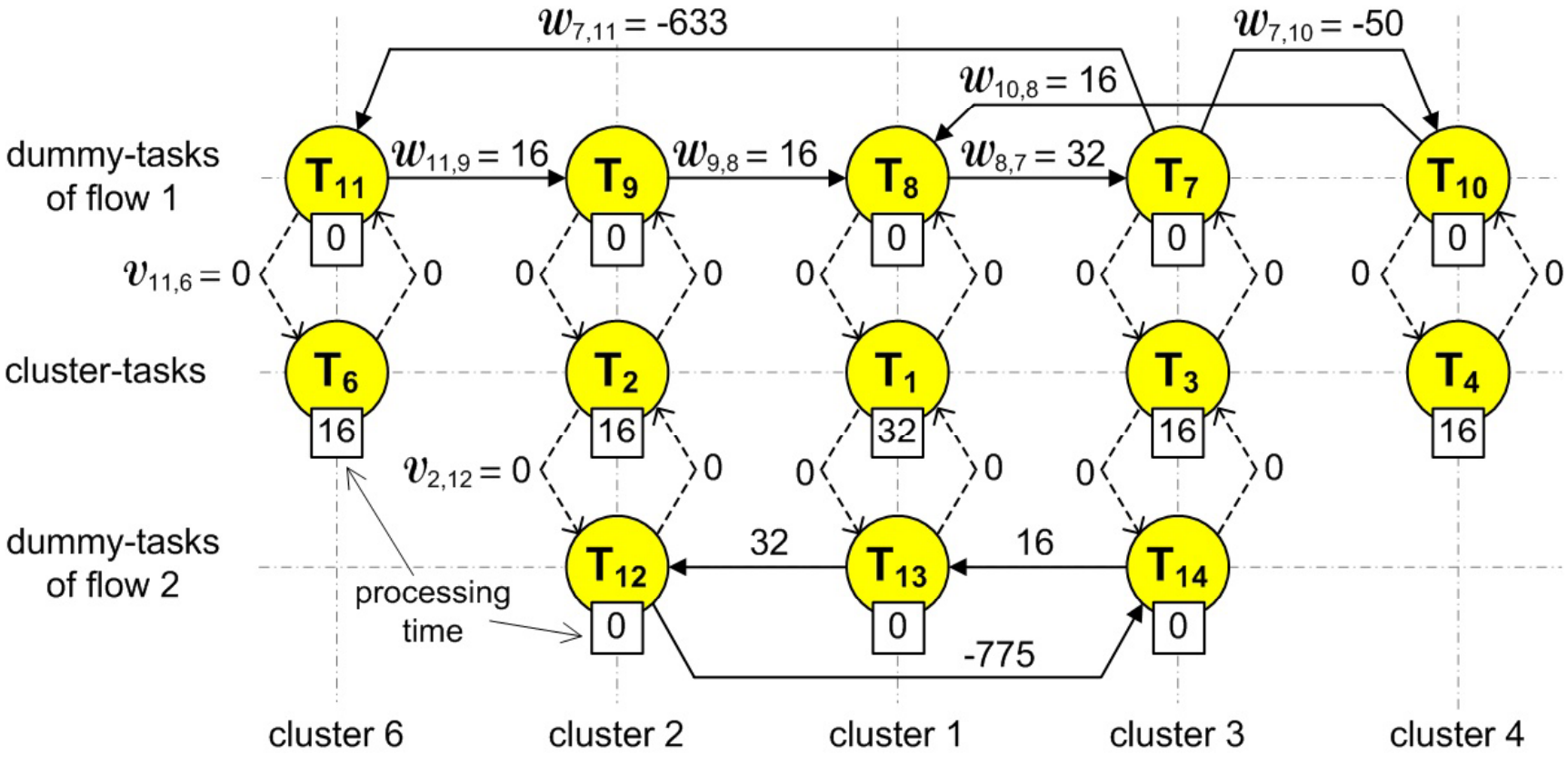
$$s_i = \hat{s}_i + \hat{q}_i \cdot \mathrm{BI}$$

It is cyclic problem due to the three aspects:

- the cluster is active only once during the period, i.e. all the flows in a given cluster are bound together
- the flows are deadline constrained
- the flows have opposite directions



Graph of the tasks



precedence constraints - positive edges

negative edges – end-to-end deadlines

offset precedence constraints - dashed edges, e.g. $\hat{s}_6 = \hat{s}_{11}$

ILP formulation for cyclic extension of PS|temp|C_{max} scheduling problem.

$$\min \sum_{i=1}^{n} \hat{s}_i + \hat{q}_i \cdot \mathrm{BI} \tag{8}$$

subject to:

$$\hat{s}_j + \mathrm{BI} \cdot \hat{q}_j - \hat{s}_i - \mathrm{BI} \cdot \hat{q}_i \geq w_{ij} \qquad \forall (i,j); i \neq j, w_{ij} \neq -\infty \tag{9}$$

$$\hat{s}_j - \hat{s}_i \geq v_{ij} \qquad \forall (i,j); i \neq j, v_{ij} \neq -\infty \tag{10}$$

$$\hat{s}_i - \hat{s}_j + \mathrm{BI} \cdot x_{ij} \geq p_j \qquad \forall \{i,j\} \in \mathcal{M}; i < j \tag{11}$$

$$\hat{s}_i - \hat{s}_j + \mathrm{BI} \cdot x_{ij} \leq \mathrm{BI} - p_i \qquad \forall \{i,j\} \in \mathcal{M}; i < j \tag{12}$$

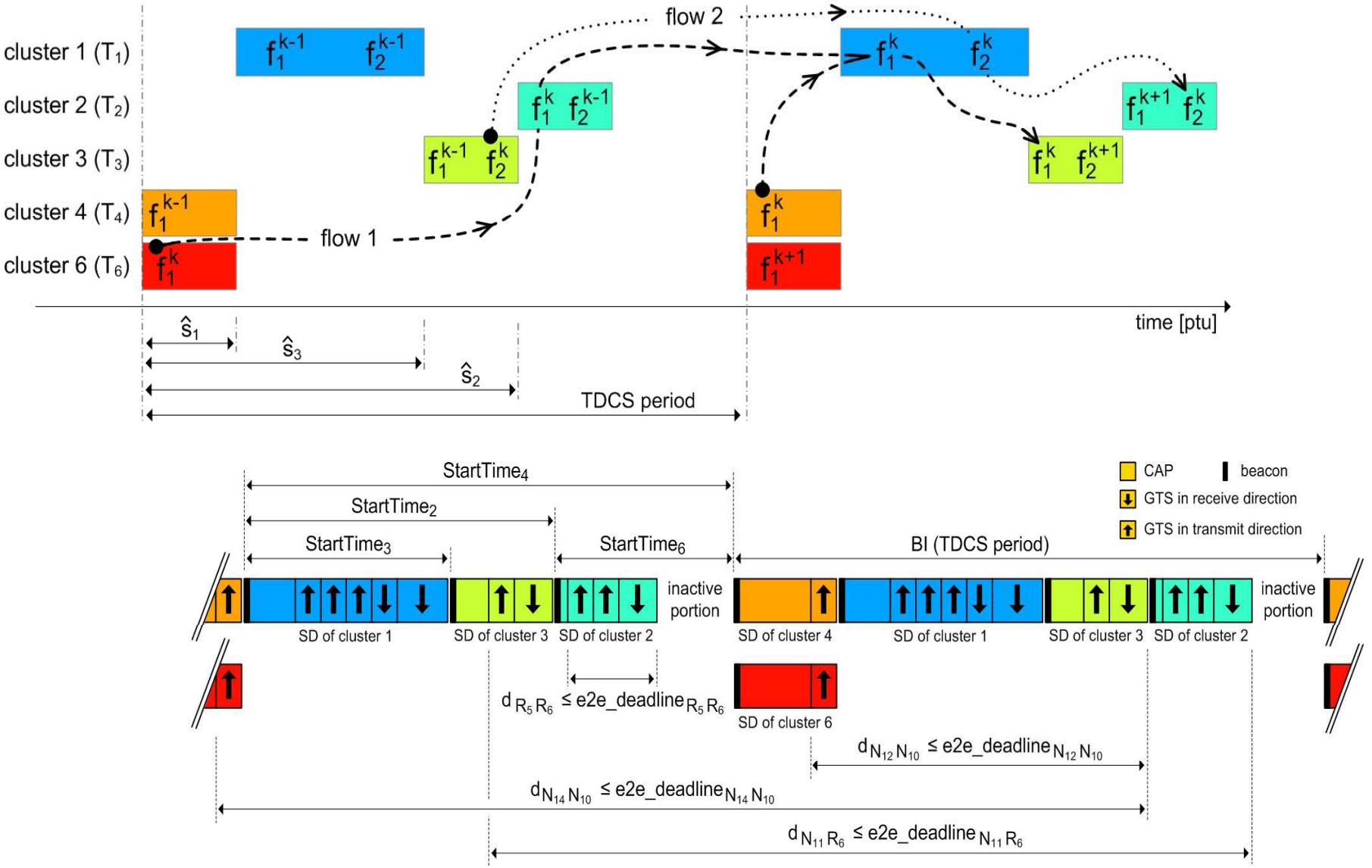
where: $\hat{s}_i \in \langle 0, \mathrm{BI} - p_i \rangle$; $\hat{q}_i \geq 0$; $\hat{s}_i, \hat{q}_i \in \mathbb{Z}$; $x_i \in \{0,1\}$

Pseudo code of the TDCS algorithm

```
(BO, SO, StartTime, GTS_params) = TDCS(C, A, flows)
01 begin
02   (BO_min, BO_max, p, SO, V, W, GTS_params) =
                                   init(C, A, flows)
03   BO = BO_min
04   while BO ≤ BO_max
05     (ŝ, q̂, feasible) = ilp_solve(V, W, BO, p)
06     if feasible
07       BO = BO + 1
08     else
09       break
10     end
11   end
12   /* calculate the StartTime parameter
                            of each cluster */
13   StartTime = config_params(ŝ, BO)
14 end
```

A – adjacency matrix of cluster-tree topology;
C – matrix of collision domains

Gantt chart of the cyclic schedule



TIME COMPLEXITY OF TDCS

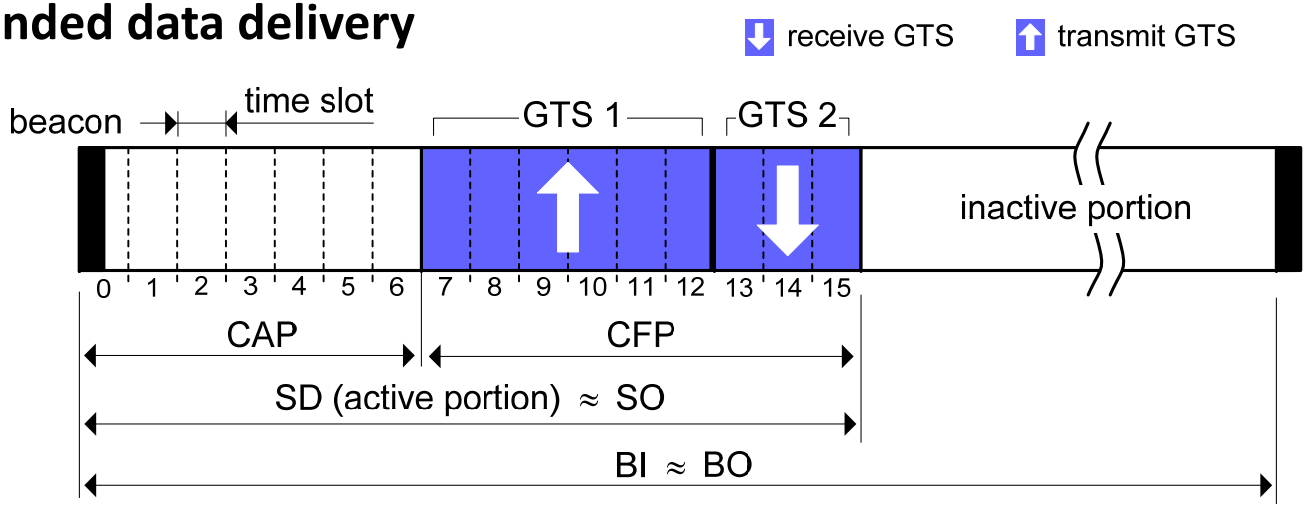
$total\_routers$ ($total\_nodes$)	N_{flow}	N_{source}	N_{task}	$time_{compact}$ [sec]	$time_{feasible}$ [sec]
11 (44)	2	3	19	0.126	0.105
		6	19	0.137	0.1
	4	3	31.5	0.408	0.19
		6	33.5	0.605	0.184
16 (64)	2	3	26.5	0.428	0.2 (1)
		6	27	2.75 (2)	0.22
	4	3	40	8.94 (2)	0.63 (1)
		6	40.5	19.38 (2)	0.407 (2)
20 (80)	2	3	25	3.74 (2)	0.21
		6	24	9.54 (2)	0.21
	4	3	44.43	–	0.63
		6	43.75	–	0.54
	8	3	97	–	6.14 (6)
		6	88.5	–	18.11 (8)
40 (160)	2	3	36.65	–	0.55 (1)
		6	34.90	–	0.51 (2)
	4	3	66.30	–	29.75 (5)
		6	68.95	–	19.72 (3)
	8	3	111.5	–	56.38 (8)
		6	114	–	61.26 (10)
60 (240)	2	3	40.9	–	1.61
		6	40.15	–	1.33 (2)
	4	3	73.75	–	7.68 (2)
		6	75.4	–	25.62 (4)
	8	3	154.5	–	61.1 (8)
		6	153	–	67.7 (11)

IEEE 802.15.4/ZigBee

- low data rates (20-250 kbps), long BI period(several seconds)
- adaptive duty cycle (active portion/BI period)
- energy/latency trade-off

- Contention Access Period (CAP)
  - CSMA/CA medium access
  - best-effort data delivery
- **Contention Free Period (CFP)**
  - **guaranteed time slots (GTS)**
  - **time-bounded data delivery**

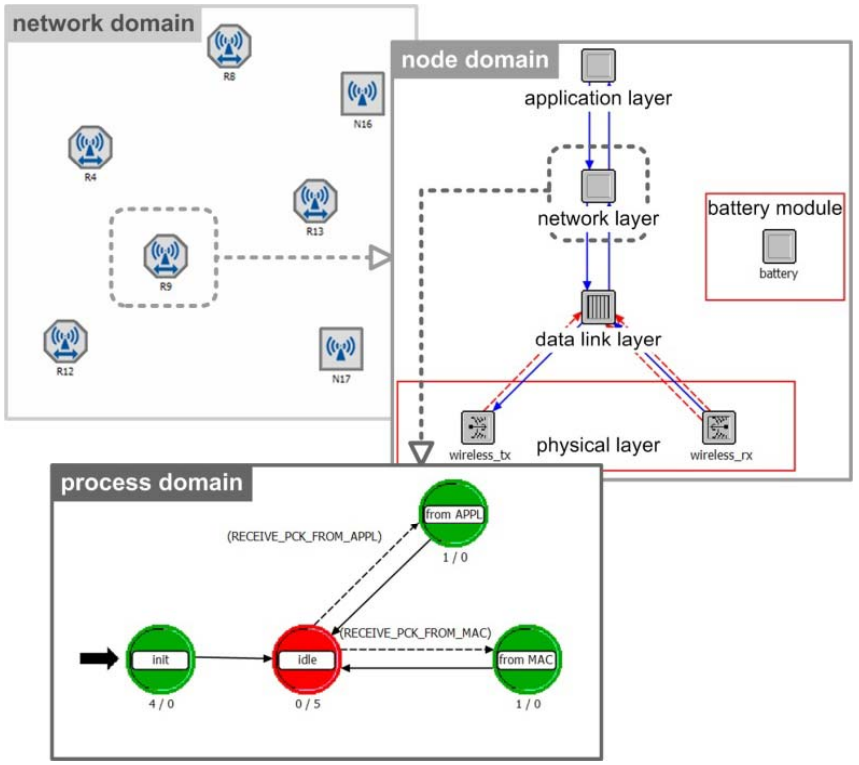
- IEEE 802.15.4 standard
  - physical layer
  - data link layer
- ZigBee specification
  - network layer
  - application layer



IEEE 802.15.4/ZigBee simulation model

- Opnet Modeler simulator

- physical layer (IEEE 802.15.4)
- data link layer (IEEE 802.15.4)
  - slotted CSMA/CA
  - GTS mechanism
- network layer (ZigBee)
  - star and cluster-tree topologies
- application layer
  - real-time data traffic
  - best-effort data traffic
- battery module

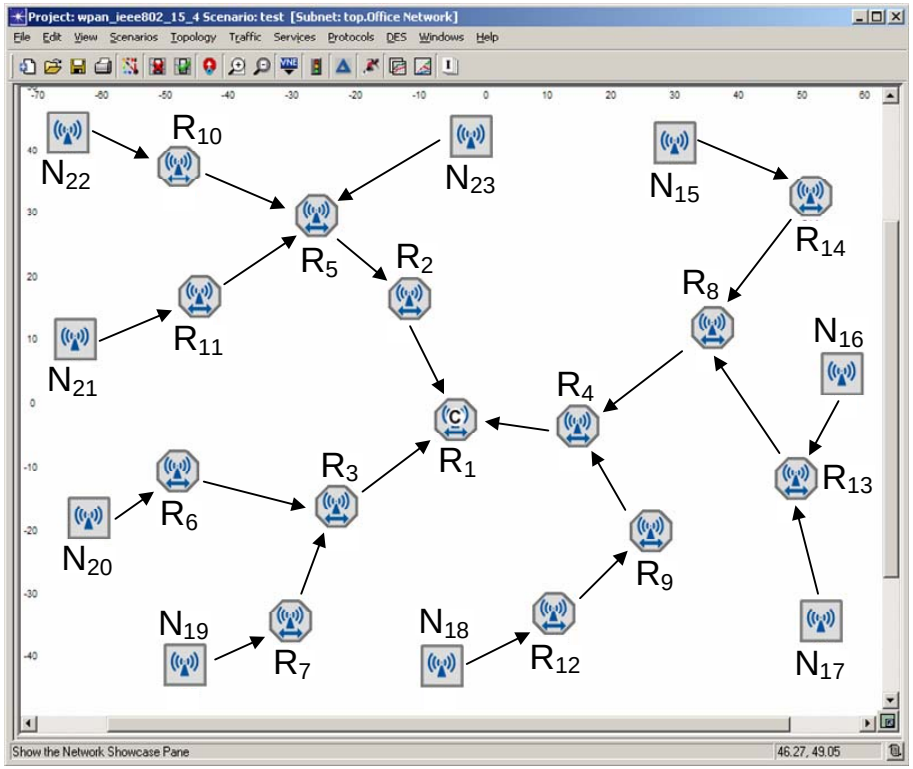


- http://www.open-zb.net
  - from 2007: >5000 downloads, >78000 visitors

Simulation study

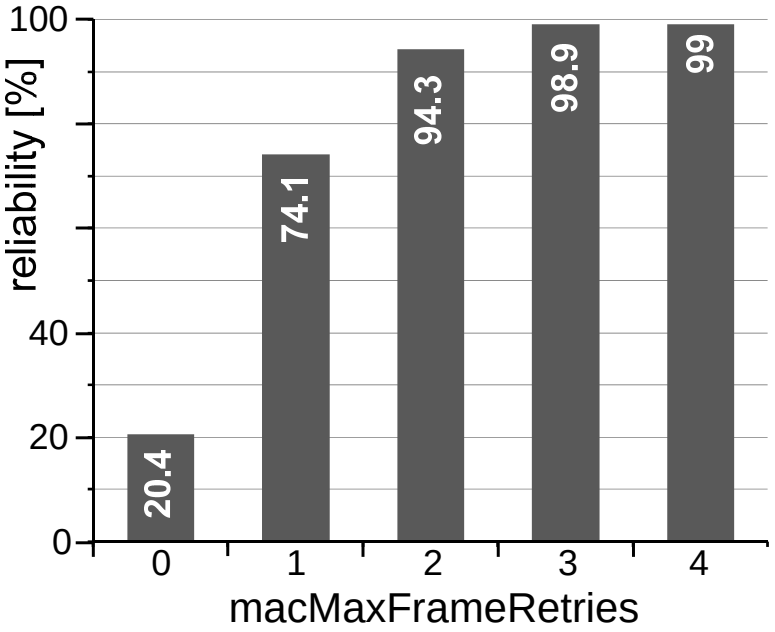
- How the maximum **number of retransmissions** impacts the **reliability** of data transmission, **energy** consumption of the nodes, **end-to-end** communication delay in IEEE 802.15.5 cluster-tree WSN?

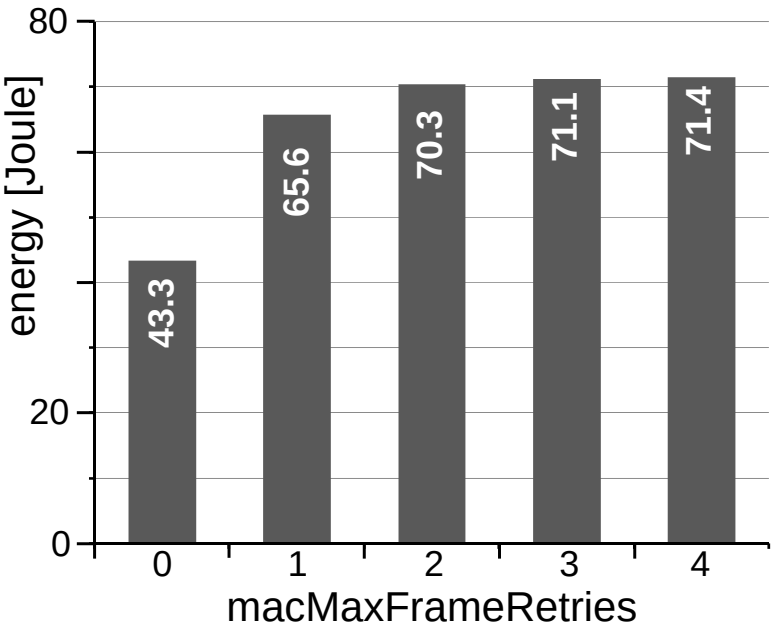
- setup
  - 14 clusters
  - 23 TelosB motes
  - 3 data flows
  - homogenous channel
  - error rate = 20%
  - one run = 20 minutes



Impact of number of retransmissions on reliability and energy

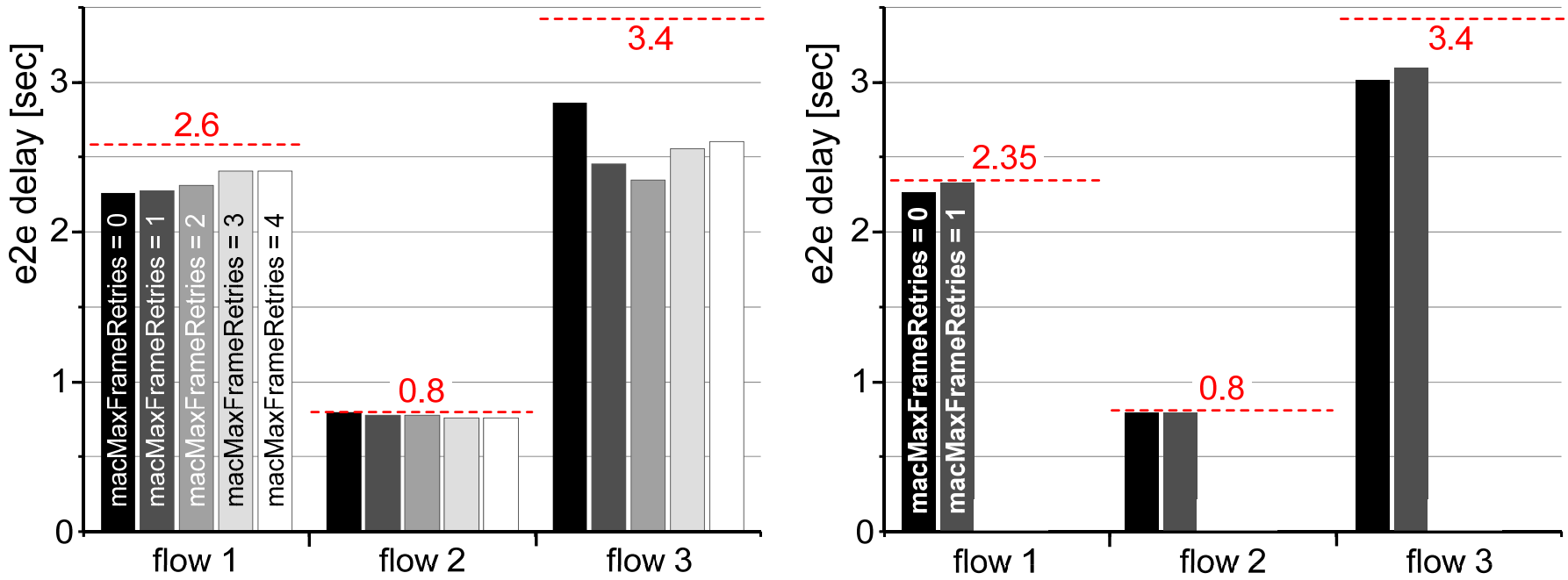
- reliability = dispatched frames / received frames
- sum of the energy consumption of all nodes





Impact of number of retransmissions on end-to-end delay

Maximum e2e delay as a function of the maximum number of retransmissions



- flow 1
  - e2e deadline = 2.6 sec leads to *macMaxFrameRetries* = 4
  - e2e deadline = 2.35 sec leads to *macMaxFrameRetries* = 1

Conclusions

Summary

- PS is useful to divide problem formulation and algorithms
- Time Constraints are useful to represent various applications
- optimal solutions for up to 100 tasks by ILP, CP, B&B
- efficient heuristics for up to 1000 tasks (even for larger problems depending on number of tasks conflicting on one resource)

Future work

- Linux with TT tasks and TT bus, problems with WCET
- Adaptive behavior (incremental Floyd's algorithm) …mode changes
- Evaluate composability of this TT approach for component based design
- Redundancy mechanisms
- Adoption of our approach by some industrial tool

More details

Recent papers

- Z. Hanzálek, P. Šůcha: Time Symmetry of Project Schedulig with Time Windows and **Take-give Resources**, MISTA 2009
- Z. Hanzálek, P. Jurčík: Energy Efficient Scheduling for Cluster-TreeWireless Sensor Networks With Time-Bounded Data Flows: Application to **IEEE 802.15.4/ZigBee**, IEEE Transactions on Industrial Informatics, Vol. 6, No. 3, August 2010
- Z. Hanzálek, P. Burget, P. Šůcha: **Profinet IO IRT** Message Scheduling With Temporal Constraints , IEEE Transactions on Industrial Informatics, Vol. 6, No. 3, August 2010