A Polynomial scheduling algorithm for IEEE 802.15.4/ ZigBee cluster tree WSN with one collision domain and period crossing constraint

Aasem Ahmad, Zdeněk Hanzálek
Department of Control Engineering
FEL, Czech Technical University in Prague
Prague, Czech Republic
ahmadaas@fel.cvut.cz, hanzalek@fel.cvut.cz

Claire Hanen
LIP6, Université P. et M. Curie, France
Paris, France
Claire.Hanen@lip6.fr

***Abstract*—Cluster scheduling is a crucial issue in cluster-tree Wireless Sensor Networks (WSNs). The paper presents a methodology that provides a Time Division Cluster Scheduling (TDCS) mechanism based on the shortest path problem. The objective is to meet all the flows' deadlines defined by the maximum number of crossed periods for each flow to reach its destination assuming one collision domain. Formulating the problem as the shortest path problem gives us a light exact algorithm suitable to the scarce properties of WSNs especially related to memory, power consumption and processors. Our polynomial algorithm leads to the minimization of the energy consumption and, consequently, the lifetime of the network is maximized by setting the TDCS period as long as possible. Since each cluster is active only once during the period, the given flow may span over several periods when there are flows with an opposite direction. The scheduling tool enables the system designers to efficiently configure all the required parameters of the IEEE 802.15.4/ZigBee beacon-enabled cluster-tree WSNs in the network configuration time.**

***Keywords*—cluster-tree, TDCS, IEEE 802.15.4, ZigBee, cyclic scheduling, energy efficiency, shortest path.**

I. Introduction

Wireless sensor network technology has been utilized by industrial monitoring and control systems in order to improve their functionality and efficiency [1]. In these systems, sensors are typically deployed to provide sensing and actuating actions even in hazardous and hard to reach areas [2], [3]. Other requirements, such as timeliness and energy efficiency, are important to be fulfilled. The use of WSNs in control applications has many advantages compared to wired solutions, where the installation and maintenance of cables and sensors are usually much more expensive than the cost of the sensors themselves.

One of the WSN topologies suited for the high predictability of performance guarantees [4] and energy efficient behavior is a *cluster-tree topology* (Fig. 1) where sensor nodes can switch off their transceivers and go into a sleep state to save energy. The cluster-tree network is supported by the IEEE 802.15.4/ZigBee [5], [6] standards which are leading technologies for low-cost, low-power and low-rate WSNs. The routing decisions are unique [6] and contention-free Medium Access Control (MAC) protocols are used to ensure collision-free access to the shared wireless medium.

In this paper, we follow the scenario described in [7]. We assume that the cluster-tree network has already been set up, i.e. each node knows its parent and child nodes (e.g. using the ZigBee's tree addressing scheme [6]). All nodes may have sensing and/or actuating capabilities, therefore they can be sources and/or sinks of data flows. We consider periodic time-constrained flows (each given by parameters such as a sink node, source nodes and deadlines) which must be known in the network configuration time. The flows traverse different clusters on their routing paths from the source nodes to the sink nodes. The fact that the cluster is active only once during the schedule period [6] leads to the so called cyclic behavior of the periodic schedule when there are flows with opposite directions in a WSN. Thus, the key problem to solve is to find a periodic schedule which specifies when the clusters are active while meeting all the flows' deadlines. Since wireless nodes are usually battery-powered, the objective is also to minimize the energy consumption of the nodes by maximizing the schedule period which leads to maximize the time when the nodes stay in low-power mode (consequently, to increase the lifetime of the network). Two major differences differentiate our work form the previous work done in [7]. The first one is measuring the flows' deadlines in terms of the maximum number of crossed periods instead of a precise *end_to_end deadline* and the second one is our assumption of one collision domain (i.e. at most, one cluster can be active at any given time) instead of multiple collision domains. These two simplifications reduced the complexity of the scheduling problem from an NP_hard problem [7] to a polynomial problem and allows us to formulate it as a shortest path problem.

A. Related work

Energy efficiency is an important requirement for WSNs in order to maximize the lifetime of the network. The major sources of energy waste in WSNs are collisions, overhearing and idle listening [8]. We eliminate those sources of energy waste by using a collision-free Time Division Cluster Schedule (TDCS) and a dedicated Guaranteed Time Slots (GTS) mechanism.

Koubaa et al. [9] have proposed an algorithm for collision-free beacon/superframe scheduling in IEEE 802.15.4/ZigBee cluster-tree networks, using the time division approach. The

focus of the work is on the feasibility of the periodic schedule, with the goal of a fair allocated bandwidth rather than low latency. In this case, it is possible to find the best schedule for router activation periods in order to avoid interference, but no timing requirements for the flows are assumed.

The authors in [10] suggest a GTS allocation algorithm for periodic real-time messages in a star topology. The algorithm determines the standard specific parameters for the network and a GTS descriptor to meet the timing constraints. However, the GTS information has to be broadcasted at the beginning of each beacon interval, which increases the network power consumption. In [11], the authors propose an extension to IEEE 802.14.4 to overcome its limitation related to the number of possible allocated GTSs in one superframe. The algorithm allows more than seven periodic nodes to be simultaneously configured to one coordinator and real time transmission can still be guaranteed for each periodic node. The bandwidth utilization and also the energy consumption are improved. The algorithm is based on a window scheduling algorithm (WSA) [12]. However, no clusters scheduling algorithm is addressed in [10] or [11] .

In [13], the authors present a solution to change the resource allocation of the Cluster-tree on the fly. This solution is directed at applications which need to deliver data to the root of the tree. The solution is not very effective in the case of simultaneous flows with opposite directions. In [14], the authors proposed a multicast mechanism in Zigbee cluster tree wireless sensor networks. However, no time-bounded flows are assumed.

This paper is built on the same scenario as [7] where the authors introduce a cluster and a GTS scheduling mechanism for a multi-hop cluster tree wireless sensor network minimizing energy consumption. The scheduling problem is NP_hard while assuming multiple collision domains and precise *end_to_end deadlines* for data flows. In order to find the schedule, the authors used integer linear programming which is not light and fast enough. In our work, we make two simplifications to the problem by defining the flows' deadlines in terms of the maximum number of crossed periods and assuming one collision domain. These simplifications enable us to solve the scheduling problem in polynomial time.

To the best of our knowledge, so far, no previous research has directly addressed an exact algorithm for solving cluster-tree scheduling problem in polynomial time taking time bounded data flows in opposite direction into consideration

B. Paper outline

The rest of the paper is organized as follows: First, we provide a generic system model, encompassing the cluster-tree topology, the data flow model and the cyclic behavior of the periodic schedule (Section II). Section III introduces the basic knowledge of IEEE 802.15.4/ZigBee needed to understand our algorithm. In section IV, a solution to the problem is presented and explained. In section IV-A, the method for determining the superframe duration and the beacon interval period for each cluster is presented. Then, in Section IV-B, we illustrate the constraint model for our problem and in Section IV-C, the transformation of the scheduling problem to the shortest path problem is described in details. Section IV-D demonstrates how to get the schedule from the shortest path tree. Section V shows our practical results. Finally, we draw the conclusion in section VI.

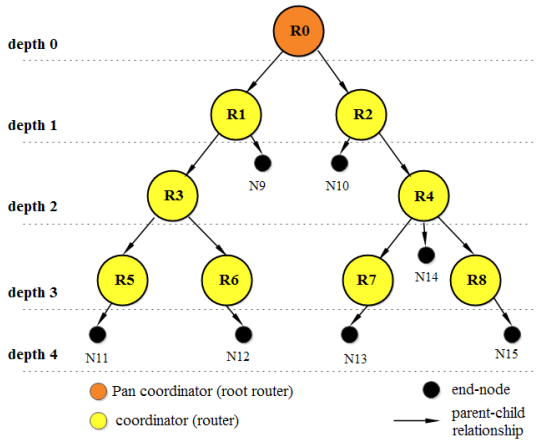

Fig. 1. Cluster-tree topology

II. System model

We consider a static deployment of wireless nodes which defines the physical topology of the cluster-tree where each pair of connected nodes can use the shared bidirectional wireless link for the data transmission between them. The logical topology defines the parent-child relationship between each pair of connected nodes.

A. Cluster-tree topology model

In cluster-tree WSNs, nodes are organized in logical groups, called clusters. There are two main types of nodes: routers and end-nodes. The nodes that can associate with previously associated nodes and can participate in multi-hop routing are referred to as routers. Each router forms a cluster and is referred to as its cluster-head. Each cluster-head is responsible for the synchronization in its cluster by sending periodic synchronization frames to its child nodes and also it handles all their data transmissions. The router that has no parent is called the root and it might hold special functions such as identification, formation and control of the entire network. The root is at depth zero. The nodes that do not allow association of other nodes and do not participate in routing are referred to as end-nodes. Both routers and end-nodes can have sensing capabilities, therefore, they are generally referred to as sensor nodes.

This *cluster-tree topology* (Fig. 1) can be represented by the adjacency matrix $A = (a_{ij})$, where $a_{ij} = 1$ if router j is the parent router of node i, otherwise $a_{ij} = 0$. Notice that A is a square matrix with dimension equal to the total number of nodes in a WSN ($total\_nodes$) [15].

In the *cluster-tree topology*, the multi-hop communication is deterministic because each node only interacts with its predefined parent router and child nodes. Messages are forwarded from cluster to cluster until reaching the sink. The time behavior of each cluster is periodic and the period of each cluster is divided into two portions. The *active* portion, during which the cluster-head enables the transmissions inside its cluster,

and the subsequent *inactive* portion. Each router (except the root) belongs to two clusters, once as a child node and once as a cluster-head.

In this paper, we assumed one collision domain, i.e. at most, one cluster can be active at any given time.

B. Data flow model

We consider periodic time constrained data flows. Each flow may have more than one cluster source (α_f) and exactly one cluster sink (β_f). In Fig. 2, we show an example for four flows where each flow has one cluster source and each cluster is represented as a node (i.e. node 0 represents the cluster formed by (R_0, R_1, R_2), node 1 represents the cluster formed by (R_1, R_3, N_9) and so on). A node periodically measures a sensed value with a given size defined by $sample\_size$ parameter and reports it to the sink. The req_period parameter defines the maximum value for node measurement period. In order to achieve some performance, we need to fix some limits for each flow. Thus, for each flow, we have chosen to limit the number of periods crossed by the flow till its delivery. So we defined h_{f_k} to be an integer value and associate the following constraint to each flow f_k: "In any periodic schedule, and any nonnegative integer y, if the y^{th} occurrence of flow f_k starts in interval $[x\lambda, (x+1)\lambda)$ where λ is the length of the schedule period, then this occurrence of f_k will end during or before the interval $[(x+h_{f_k})\lambda, (x+h_{f_k}+1)\lambda)$. We say that the periodic schedule meets the period crossing constraint for flow f_k, while data is delivered within, at most, $(h_{f_k}+1)$ periods".

Flow id	Source (α_f)	Sink (β_f)	Maximum number of crossed periods (h_f)
1	6	4	3
2	0	8	1
3	7	3	1
4	2	5	2

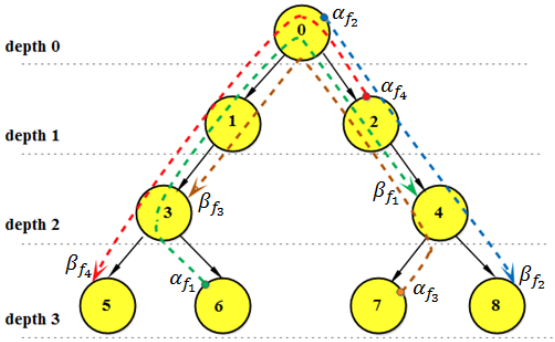

Fig. 2. Four time constrained data flows in the cluster tree topology.

In fact, the value of h_{f_k} is derived from the $e2e\_deadline_{f_k}$ as follows:

$$h_{f_k} = \begin{cases} \left\lfloor \dfrac{e2e\_deadline_{f_k}}{\lambda} \right\rfloor - 1 & \text{if } e2e\_deadline_{f_k} \geq \lambda, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

where $e2e\_deadline_{f_k}$ is the maximum time between the instant when the source sends the message and the instant when the sink receives the message for a given flow f_k. In the rest of the paper, we will use the notation *deadline* while meaning the maximum number of crossed periods.

C. Cyclic nature

The fact that the cluster is active only once during the schedule period leads to the so called cyclic behavior of the periodic schedule (i.e. time between the instant when a source sends the message and the instant when the sink receives this message spans over several periods) when there are flows with opposite direction in a WSN. For example, considering two opposite flows (see Fig. 2) f_1= (6→3→1→0→2 →4) and f_3 = (7→4 →2→0→1→3) and assuming a unit time length of each cluster activity, regardless of the amount of data they handle in any schedule of the node activities once during a period, the minimization of the number of crossed periods for the first flow is in contradiction with the minimization of the number of crossed periods for the second one. Fig 3 shows a schedule for the shared clusters between f_1 and f_3 which minimizes the number of crossed periods required for f_1. Notice that this schedule led to the worst case for flow f_3, where sending data from cluster 4 to cluster 1 required 3 crossed periods while it is completed in the same period (0 crossed period) for flow f_1. Hence, the proper order of the active portions of clusters is a subject of optimization even if the size of the data and collisions of clusters are not assumed.

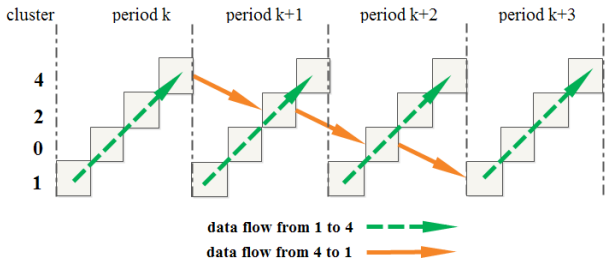

Fig. 3. Schedule example for four clusters in Fig 2.

III. Overview of IEEE 802.15.4/ZigBee

The IEEE 802.15.4 [5] standards are leading technologies for low-cost, low-power and low-rate WSNs. The physical and data link layers are specified by IEEE 802.15.4 while the network and application layers are defined by the ZigBee specification [6]. The MAC layer supports both the beacon and non beacon enabled mode. This paper only considers the beacon-enabled mode, since it supports cluster-tree topology. While IEEE 802.15.4 in the beacon-enabled mode supports only the star-based topology, the ZigBee specification has proposed its extension to the cluster-tree topology. In the particular case of ZigBee cluster-tree WSNs, a PAN coordinator is identified as the root of the tree and forms the initial cluster (Fig. 1). The other routers join the cluster-tree in turn by establishing themselves as cluster-heads (coordinator), starting to generate the beacon frames for their own clusters. Beacon frames are periodically sent at the *Beacon Interval* (BI) to synchronize the child nodes that are associated with a given cluster-head and to define a superframe structure (Fig. 4).

Each cluster's period, corresponding to the BI, is divided into active and inactive portions. During the inactive period, all

nodes associated to one coordinator, including the coordinator, can go into a sleep state to save energy. The inactive period may be void. The active period corresponding to *Superframe Duration* (SD) is subdivided into 16 equally sized time slots. The first time slot is occupied by the beacon frame and the remaining time slots are partitioned into a *Contention Access Period* (CAP) and optional *Contention Free Period* (CFP). During the (CAP), a slotted CSMA/CA protocol is used for the best-effort data delivery. Within the CFP cluster-head can allocate Guaranteed Time Slots (GTSs) to its child nodes. The CFP supports up to 7 GTSs and each GTS may contain multiple time slots which is determined by the length of the real time data. Each child node may request up to one GTS in the *transmit direction*, i.e. from the child node to the parent router, and/or one GTS in the *receive direction*, i.e. from the parent router to the child node. Note that a node to which a GTS has been allocated can still transmit the best-effort data within the CAP.

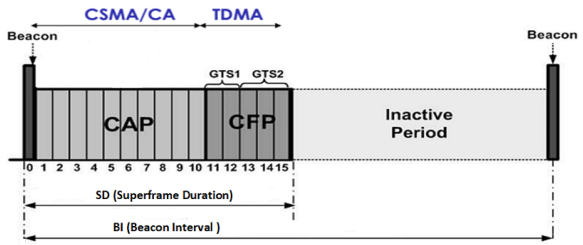

Fig. 4. Superframe structure.

The length of the active and inactive period as well as the length of a single time slot and the usage of GTS slots are configurable. Durations of the cluster's period (BI) and the cluster's active portion (SD) are defined by two parameters, the *Beacon Order* (BO) and the *Superframe Order* (SO) as follows:

$$\begin{aligned} BI &= aBaseSuperframeDuration \cdot 2^{BO} \\ SD &= aBaseSuperframeDuration \cdot 2^{SO} \end{aligned} \tag{2}$$

where $0 \leq SO \leq BO \leq 14$ and *aBaseSuperframeDuration* = 15.36 ms (assuming the 2.4 GHz frequency band and 250 kbps of bit rate) and denotes the minimum duration of active portion when $SO = 0$. Note that the ratio of the active portion (SD) to the whole period (BI) is called the *duty-cycle*.

IV. Time division cluster scheduling (TDCS) and its application to IEEE 402.15.4/ZigBee

The key idea of this paper is to find a feasible periodic Time division cluster schedule satisfying the flows' deadlines and minimizing the energy consumption. The objective is to formulate our algorithm in such a light and fast way in order to satisfy the WSN requirements. All clusters have an equal BI, defined by BO, but various SD, defined by SO, (i.e. various duty-cycle) to ensure efficient bandwidth utilization. The BI should be as long as possible to minimize the clusters' duty-cycle and, consequently, to minimize the energy consumption of the node. Our proposed solution consists of the following parts:

- Determine the duration of the active portion and the beacon interval of each cluster.
- Define the problem's constraint model which basically describes the flows' deadlines.
- Formulate the problem as the shortest path problem.
- Finally, find the schedule with respect to the shortest path tree.

The pseudo code of our algorithm is presented in Fig. 5.

A. Duration of the cluster's Active Portion and Beacon Interval in IEEE 802.15.4/ZigBee

we follow the methodology used in [7] to calculate the superframe duration (SD) for each cluster. The duration of (SD) is related to the number and length of each allocated GTS. Each GTS includes in addition to the effective data, the inter-frame spacing (IFS) eventual acknowledgment and retransmissions. IFS separates consecutive frames and it is equal to an SIFS (Short Inter-Frame Spacing) or an LIFS (Long Inter-Frame Spacing) according to the length of MAC frame. In the case of the acknowledged transmissions (i.e. $sample\_ack = 1$) the sender waits for the corresponding acknowledgment frame, at most, the *macAckWaitDuration* [5] (*macAWD*). If an acknowledgment frame is received within the *macAckWaitDuration*, the transmission is considered successful. Otherwise, the data transmission and waiting for the acknowledgment are repeated up to a maximum of *macMaxFrameRetries* [5] (*macMFR*) times. If an acknowledgment frame is not received after *macMaxFrameRetries* retransmissions, the transmission is considered failed. The duration of a GTS required for the whole data transmission (data frame, IFS, eventual acknowledgment and retransmissions) is expressed as:

$$\begin{aligned} \varphi &= \begin{pmatrix} frm\_size_i/rate & + \\ macAWD \cdot sample\_ack_i \end{pmatrix} + \Delta IFS \\ T_{GTS} &= \sum_{i=1}^{e} (macMFR \cdot sample\_ack_i + 1) \cdot \varphi \end{aligned} \tag{3}$$

where frm_size is the size of transmitted frame including the data payload, MAC and PHY headers; the $rate$ is the data rate equal to 250 kbps; ΔIFS is equal to SIFS or LIFS depending on the length of MAC frame; and e is the number of flows in the transmit or receive direction belonging to a given child node. The number of allocated time slots for a given GTS is then equal to:

$$N_{GTS} = \left\lceil \frac{T_{GTS}}{TS} \right\rceil \tag{4}$$

where TS is the duration of a time slot and is equal to SD/16. The number of time slots, N_{GTS}, is calculated for each allocated GTS in a given superframe. The remaining time slots of the SD are utilized for the best-effort traffic within the CAP. The allocated GTSs can not reduce the length of the CAP to less than *aMinCAPLength* [5].

The superframe duration (SD) is then computed iteratively starting from $SO = 0$. If the number of time slots required for all allocated GTSs in a given superframe is greater than $16 - \lceil aMinCAPLength/TS \rceil$, then SO is increased by 1 and the length of each GTS (Eq. (4)) is recalculated. This

```
01 BI_min = 0, BI_max = min_k(req_period_{f_k})
02 BO_max = ⌊log_2 (BI_max / aBaseSuperframeDuration)⌋
03 for each cluster i
04   SO_i = −1
05   repeat
06     for each child node j of cluster i
07       calculate N^T_{GTS,j} for all flows
                              in transmit direction
08       calculate N^R_{GTS,j} for all flows
                              in receive direction
09     end
10     SO_i = SO_i + 1
11   until Σ_j N^T_{GTS,j} + Σ_j N^R_{GTS,j} ≤
                              16 − ⌈aMinCAPLength/TS⌉
12  BI_min = BI_min + aBaseSuperframeDuration × 2^{SO_i}
13 end
14 BO_min = ⌈log_2 (BI_min / aBaseSuperframeDuration)⌉
15 BO = BO_min, feasible=0
16 while BO ≤ BO_max
17   (N_new, new_feasible) = solve(flows, BO, A)
18   if new_ feasible
19     feasible=new_feasible
20     N = N_new
21     BO = BO + 1
22  else
23     break
24   end
25 end
26 if feasible
27     BO = BO − 1
28     routers_order = topological_order(N)
29 end
```

Fig. 5. Pseudo code of time division cluster scheduling algorithm

procedure is repeated until all allocated GTSs fit into a given SD (see Fig. 5 (line 3 to line 11)).

The beacon interval BI is determined by the value of BO. To follow our assumption of one collision domain, The minimum value fo BO (BO_{min}) is set to the minimum integer value which is sufficient to fit all clusters' active portion. (Fig. 5 line 12, 14). The maximum value of BI, given by BO_{max} is rounded down to the nearest BI towards the shortest req_period among all of the flows (Fig. 5 lines 1 and 2). In fact, the larger the value of BI, the inactive period and the beacon transmission interval become longer. Thus the energy consumption in the network is reduced. So, we initialize BO to BO_{min} and we find the feasible TDCS of the problem using the $solve()$ function based on the shortest path formulation of the scheduling problem which is explained in section IV-C. If such a schedule exists, we increase BO by 1 as long as it does not reach its maximum value BO_{max}. We repeat this procedure till BO reaches BO_{max} or no feasible solution exists. Notice that the feasibility of the schedule is related to the maximum number of crossed periods for each flow which in turn is decreased when the value of BO is increased (Eq. 1). At the end, BO will be equal to the largest value satisfying the required flows' deadlines (Fig 5 lines 15 to 27).

B. Modeling Zigbee constraint

Before going into the details of the constraint model in our problem, let us introduce the graph representation of a given schedule. Consider the cluster-tree presented in Fig. 2, with 9 clusters and a TDCS schedule as presented in Fig. 6 assuming that all clusters are allocated equal bandwidth. Notice that this schedule is set to minimize the downstream traffic latency, (parents appear earlier in the schedule than the child nodes).

Fig. 6. TDCS for minimizing the downstream traffic latency.

Fig. 7 shows the graph representation of the schedule in Fig. 6 with respect to the cluster-tree topology. Each cluster is represented as a node and each pair of neighboring clusters are connected by an edge. The edge direction is determined by the schedule (i.e. if i and j are neighbor nodes and i proceeds j in the schedule then the edge is directed from node i to node j). In fact, for each edge (i, j) in Fig . 7 we derive that cluster j receives the data from cluster i in the same period while cluster i receives data from cluster j in the next period.

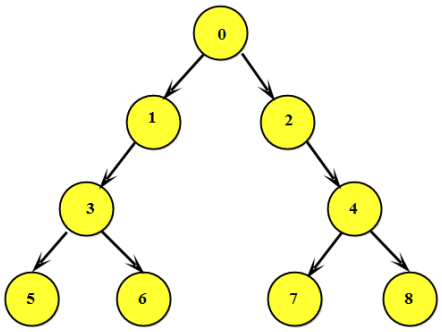

Fig. 7. Graph representation of schedule in Fig. 6

From the observation above, it follows that the number of crossed periods for each flow is determined by the number of edges from the source cluster of the flow to the sink cluster which have an opposite direction to the flow's direction. For example, if the flow from cluster 2 to cluster 5 started during period x, then the data will be delivered to cluster 5 during the period $x + 1$ because there is only one edge (0,2) which has the opposite direction to the flow, i.e. the number of crossed periods is 1.

Back to the constraint model and following the idea presented in [16], let us redraw Fig. 2 in a more detailed way as shown in (Fig. 8) where the clusters are represented as nodes and the set of edges represents the unique path for each flow in Fig. 2. One additional edge for each flow f_k is added to the graph, directed from the sink to the source and weighted by the $(h_{f_k} + 1)$ value. Each additional edge represents the maximum number of periods for flow f_k to reach its destination including the starting period. Then, as shown in (Fig. 8), there are nodes connected by edges in both directions (e.g. node 0 and node 1) and, of course, we may have cases where nodes are connected by edge(s) in one direction (e.g. node 3 and node 5). The latter case is called a bridge. In order to find the periodic schedule of our problem, we have to remove the additional edges and be sure that each pair of neighboring nodes is connected by edges having one direction (i.e. for each pair of neighboring nodes i and j we have to decide whether we will keep the (i, j) edges or (j, i) edges but not both). Of course our decision is

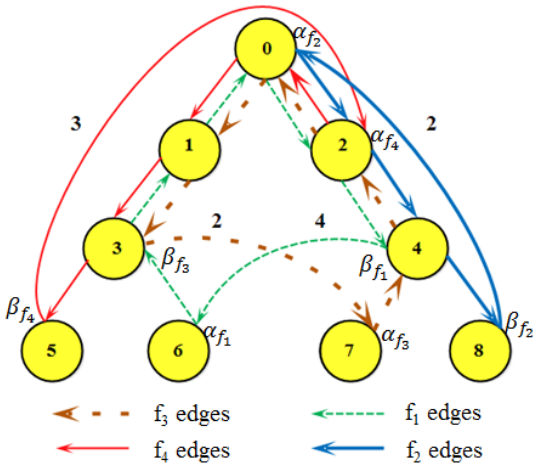

Fig. 8. The detailed graph version for the graph in Fig 2.

based on the maximum number of crossed periods for each flow (i.e. for each flow f_k the number of edges crossed by the flow which have the opposite direction to the flow direction does not exceed the value of h_{f_k}).

Let us denote in Fig. 8 by forward edges, the set of edges (i, j) where $depth(i) < depth(j)$, and backward edges, the set of edges (i, j) where $depth(i) > depth(j)$. Notice that i and j are a pair of neighboring nodes and each node has the same depth as in Fig. 2, i.e. node 0 has depth 0, while nodes 8,7,6,5 have depth 3. Then each data flow can have one of the following forms:

- f_k is an increasing flow if the unique path from α_{f_k} to β_{f_k} contains only forward edges.
- f_k is a decreasing flow if the unique path from α_{f_k} to β_{f_k} contains only backward edges.
- Otherwise f_k is a bidirectional flow. In this case, there exists a node z_{f_k} crossed by f_k where the path from α_{f_k} to z_{f_k} is decreasing and from z_{f_k} to β_{f_k} is increasing.

Notice that f_2 is an increasing flow, while f_1, f_3, f_4 are bidirectional flows. The constraint model is presented in [16] and illustrated in Fig. 9. Each variable N_j represents the number of forward edges from node 0 to node j in any feasible solution. The constraint (5a) is the topological constraint. Hence, for any pair of neighboring nodes i and j, the number of forward edges in any feasible solution from node 0 to node j is greater than or equal to the number of forward edges from node 0 to node i by 1 where $depth(j) > depth(i)$. The constraints (5b,5c,5d) are the flows constraints. The constraint (5b) restricts the number of crossed periods required for the increasing flow (i.e. the number of backward edges from the source of the flow f_k to its sink is less than or equal to h_{f_k}). The constraint (5c) is similar to (5b) but for the decreasing flow (i.e. it restricts the number of forward edges from the source of the flow f_k to its sink to be less than or equal to h_{f_k}). The constraint (5d) is for bidirectional flow which starts as a decreasing flow and then changes its direction to an increasing flow at node z_{f_k}. The constraint (5e) ensures a non-negative value for each variable N_j.

$$
\begin{cases}
\forall i, j \text{ are neighbor clusters and } depth(j) > depth(i) & \\
0 \leq N_j - N_i \leq 1 & \text{(5a)} \\
\text{For any increasing } f_k & \\
depth(\beta_{f_k}) - depth(\alpha_{f_k}) - (N_{\beta_{f_k}} - N_{\alpha_{f_k}}) \leq h_{f_k} & \text{(5b)} \\
\text{For any decreasing flow } f_k & \\
N_{\alpha_{f_k}} - N_{\beta_{f_k}} \leq h_{f_k} & \text{(5c)} \\
\text{For any bidirectional flow } f_k & \\
N_{\alpha_{f_k}} + depth(\beta_{f_k}) - depth(z_{f_k}) - N_{\beta_{f_k}} \leq h_{f_k} & \text{(5d)} \\
\text{For each node } j & \\
N_j \geq 0 & \text{(5e)}
\end{cases}
$$

Fig. 9. The constraint model.

Notice that the system matrix is totally unimodular with non negative constraint for the decision variables, so if we found a solution using linear programming LP, then it will be a non-negative integer solution. The flows' constraints of the example presented in Fig. 2 are illustrated in Fig .10.

$$
\begin{cases}
N_6 - N_4 \leq \ \ 1 & \text{for flow } f_1 \\
N_0 - N_8 \leq -2 & \text{for flow } f_2 \\
N_7 - N_3 \leq -1 & \text{for flow } f_3 \\
N_2 - N_5 \leq -1 & \text{for flow } f_4
\end{cases}
\tag{6}
$$

Fig. 10. Flows' constraints for the example presented in Fig. 2

C. *Clusters scheduling formulated as a shortest path problem*

This section presents the details of our contribution for solving the constraint model presented in Fig. 9 which enables us to find the right activation order of the clusters meeting flows' deadlines if such a solution exists. Obviously, the system could be solved using linear programming [16]. However, in order to find the schedule in a fast and reasonable amount of time, and to be sure that our algorithm is able to run in WSN platforms with scarce processing power, linear programming is not the best choice. Thus, we transformed our problem to the shortest path problem taking the advantage of the totally unimodular system matrix. The proposed algorithm is able to give us the exact solution in polynomial time if the system is feasible.

All the constraints in Fig. 9 have the form of $var_1 - var_2 \leq cons$ where var_1 represents N_i or N_j and the same for var_2, while $cons$ is a constant which may have a positive or negative integer value. As a result, all inequalities in our model will have the case of $var_1 - var_2 \leq cons$ when $cons \geq 0$ or the case of $var_2 - var_1 \geq -cons$ when $cons < 0$. If var_1 and var_2 are represented by two nodes connected by an edge describing the distance between them (note that the term distance is used to simplify the used methodology), then for the first case, the edge is weighted by $cons$ and directed from var_2 to var_1. In fact the edge's weight $(cons)$ in this case defines the maximum distance from var_2 to var_1. For the second case, the edge will be weighted by $-cons$ and directed from var_1 to var_2. The value $-cons$ defines the minimum distance from var_1 to var_2.

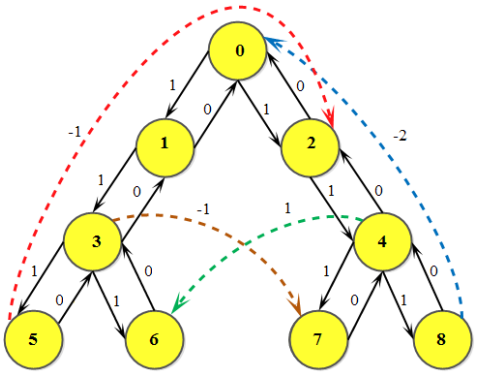

Fig. 11. Inequalities graph G.

From the observation above, it follows that our problem can be solved by the shortest path tree. To do so, we create a directed graph $G(V, E)$ as shown in Fig. 11, where each cluster is represented as a node. The set of edges is created with respect to the set of constraints. For each constraint in the form $N_i - N_j \leq cons$, we add an edge from node j to node i with weight equal to $cons$. In graph G, we kept all inequalities having the form of less or equal. In (Fig. 11), the set of solid edges represents the topological constraint (5a) while the set of dashed ones represents the constraints related to the type of the flow (increased, decreased, bidirectional). For example, the constraint for f_1 is represented by edge $(4, 6)$ weighted by 1 while the constraint for f_2 is represented by edge $(8, 0)$ weighted by -2. So the feasible solution of our problem is the shortest path from node 0 in the graph to all nodes in the graph G including node 0. (i.e. the value of N_j is the length of the shortest path from node 0 to node j). It is clear that $N_0 = 0$ in any feasible solution.

Since some weights may be negative, a negative cycle may exist in the inequalities graph. In this case, there is no feasible schedule solution satisfying the flows' deadlines. So the nonexistence of a negative cycle is a necessary condition to the feasibility. To prove it, suppose there exists a negative cycle in the inequalities graph, then there is an edge (i, j) with a negative weight $cons$ in this cycle. This edge (i, j) represents the constraint $N_j - N_i \leq cons$ where $cons < 0$. In fact, this constraint is equivalent to the constraint $N_i - N_j \geq -cons$. Hence, any feasible solution must satisfy this constraint (i.e. the shortest path form j to i is greater or equal to $-cons$). If the shortest path from j to i satisfies the new constraint then there is no negative cycle because all other paths (if any other exists) from j to i will have a greater or equal length to the shortest path from j to i. This is a contradiction.

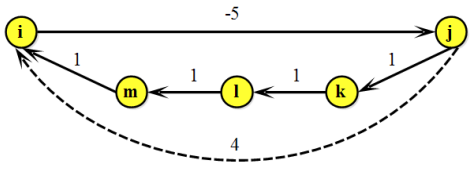

Fig. 12. Example of negative cycle length

Fig. 12 represents an example of an inequality graph with a negative cycle. Notice that the edge (i, j) represents the constraint $N_j - N_i \leq -5$ which is equivelant to $N_i - N_j \geq 5$ while the dashed edge (j, i) represents the path from node j to node i (i.e. the j, k, l, m, i path) and expresses the constraint $N_i - Nj \leq 4$. This is a contradiction.

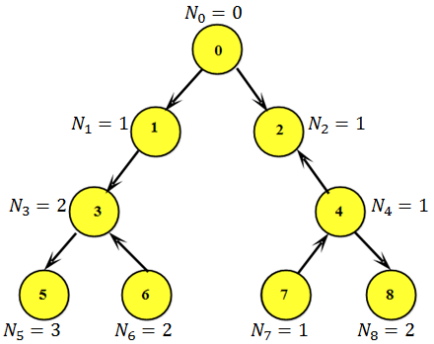

Fig. 13. Graph representation of the schedule found by solving the shortest path problem illustrated in Fig. 11.

In fact, if the problem has a feasible solution, then the shortest path formulation gives us only one feasible solution, while there could be other feasible solutions. But because we assume only one collision domain, then any solution has the same effect on the calculation of the beacon interval BI. For solving the shortest path tree problem, we use the Bellman_Ford algorithm which is able to detect the existence of cycles of negative length. Bellman_Ford runs in $O(|V||E|)$ time, where $|V|$ and $|E|$ are the number of nodes and edges respectively.

D. Finding the schedule

The last step in our algorithm is to get the schedule representing the order of the cluster activation from the graph in Fig. 13. Obviously, the graph is a directed acyclic graph (DAC) (i.e. directed graph with no cycle), hence, there exists at least one topological ordering of its nodes (i.e. ordering of its nodes as $1, 2, \ldots, n$ so that for every edge (i, j), we have $i < j$). Every topological ordering represents one possible schedule of the clusters. Fig. 14 illustrates one possible ordering where the allocated bandwidth for each cluster calculated in section IV-A are not taken into consideration (i.e. we show just the ordering of the clusters where the first active cluster is cluster 0 and the last one is cluster 5). This step is presented in Fig. 5 line 28.

0	1	7	4	2	8	6	3	5

Fig. 14. Possible routers activation order according to Fig .13.

V. EXPERIMENTAL RESULTS

In our experiment, we will focus on the time complexity of our algorithm. The effect of the length of the schedule period given by BO, to the energy consumption is demonstrated in details in [7]. The proposed algorithm was implemented in MATLAB and tested on a PC with an Intel Core i7 3520M Dual core (2.9 GH) CPU and 8 GB RAM.

The cluster-tree topology is constructed as follows: The routers are successively generated until the total number of the

routers in the network reaches $total\_routers$. Each router has 3 child end-nodes. The total number of nodes ($total\_nodes$) in the WSN is shown in parentheses in the first column of Tab. I.

TABLE I. TIME COMPLEXITY OF THE TDCS ALGORITHM.

$total\_routers$ ($total\_nodes$)	N_{flow}	N_{source}	$time_{feasible}$ [sec]
40	4	3	0.1028
		6	0.0993
(160)	8	3	0.1001
		6	0.0981
60	4	3	0.3211
		6	0.3100
(240)	8	3	0.3123
		6	0.3116
80	4	3	0.7946
		6	0.7716
(320)	8	3	0.7717
		6	0.8014
150	10	3	6.0020
		6	6.2147
(600)	15	3	6.1691
		6	5.9836

For each cluster-tree topology, we study the effect of the various number of data flows N_{flow} equals to 4 or 8 (10 and 15 for the last case) and the number of sources N_{source} of each data flow equals to 3 or 6. We fix the other parameters of data flows such as $req\_period = 2$ sec; $sample\_size = 120$ bits; $sample\_ack = 0$ and $e2e\_deadline = 150\ sec$ (for the last case where $total\_routers = 150$, we set $req\_period = 10\ sec$; $sample\_size = 120$; $sample\_ack = 0$ and $e2e\_deadline = 1000\ sec$). For each topology and for each combination of N_{flow} and N_{source}, we randomly generate a set of 20 instances and run the scheduling algorithm for each of them. Each instance is defined by number of routers, number of flows and number of sources for each flow. The average execution time $time_{feasible}$ for finding the feasible schedule with largest possible beacon interval (without taking the required time for generating the required topology into consideration) is shown in the third column in Tab. I. It is clear from the results that the computation is finished in a short time even for large scale networks which proves the strength of our proposed algorithm.

VI. CONCLUSION

The scarce resources properties for WSNs guided us to think about a fast and light algorithm which is able to find a feasible energy efficient schedule for one collision domain cluster-tree WSNs in a short time meeting the flows' timing constraints. The key step was definning the $end\_to\_end\ deadline$ in terms of the maximum number of the crossed periods. We demonstrated, through experiments, the power of our exact polynomial algorithm in terms of the required time to find the right activation order of the routers. Formulating the problem as a shortest path tree is a promising approach for our future work especially for a distributed version of our algorithm. In fact our proposed algorithm is not limited to WSNs; it may be applied to different applications where the limited number of periods is the metric to be fulfilled.

Acknowledgement

This work was supported by the Grant Agency of the Czech Republic under the Project GACR P103/12/1994.

REFERENCES

[1] M. Chitnis, Y. Liang, J. Y. Zheng, P. Pagano, and G. Lipari, "Wireless Line Sensor Network for Distributed Visual Surveillance," in *Proc. of the 6th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN)*, Oct. 2009, pp. 71–78.

[2] A. Willig, "Recent and Emerging Topics in Wireless Industrial Communications: A Selection," *IEEE Trans. on Industrial Informatics*, vol. 4, no. 2, pp. 102–124, May 2008.

[3] *IEEE P802.15 Wireless Personal Area Networks: Proposal for Factory Automation*, Working Draft Proposed Standard, Rev. 802.15.4-15/08/0571r0, 2009.

[4] P. Jurčík, R. Severino, A. Koubâa, M. Alves, and E. Tovar, "Real-Time Communications over Cluster-Tree Sensor Networks with Mobile Sink Behaviour," in *Proc. of the 14th IEEE International Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, Aug. 2008.

[5] *Part 15.4: wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs)*, IEEE SA Standards Board Std. 802.15.4, 2006.

[6] *ZigBee Specification*, ZigBee Standards Org. Std. 053 474r13, 2006.

[7] Z. Hanzálek and P. Jurčík, "Energy efficient scheduling for cluster-tree Wireless Sensor Networks with time-bounded data flows: Application to IEEE 802.15.4/ZigBee," *IEEE Transaction on Industrial Informatics*, vol. 6, no. 3, Aug. 2010.

[8] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC protocol for Wireless Sensor Networks," in *Proc. of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Jun. 2002, pp. 1567–1576.

[9] A. Koubâa, 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*, vol. 40, no. 3, pp. 321–354, Oct. 2008.

[10] S.-E. Yoo, P. K. Chong, D. Kim, Y. Doh, M.-L. Pham, E. Choi, and J. Huh, "Guaranteeing real-time services for industrial wireless sensor networks with ieee 802.15.4," *Industrial Electronics, IEEE Transactions on*, vol. 57, no. 11, pp. 3868–3876, Nov 2010.

[11] Y. Ding and S. H. Hong, "CFP scheduling for real-time service and energy efficiency in the industrial applications of IEEE 802.15.4," *Communications and Networks, Journal of*, vol. 15, no. 1, pp. 87–101, Feb 2013.

[12] S. H. Hong, "Scheduling algorithm of data sampling times in the integrated communication and control systems," *Control Systems Technology, IEEE Transactions on*, vol. 3, no. 2, pp. 225–230, Jun 1995.

[13] N. P. R. Severino and E. Tovar, "Dynamic cluster scheduling for cluster-tree WSNs," CISTER Reasearch Unit, Tech. Rep. CISTER-TR-130205, Feb. 2013. [Online]. Available: http://www.cister.isep.ipp.pt/

[14] H. Boujelben, O. Gaddour, and M. Abid, "Enhancement and performance evaluation of a multicast routing mechanism in zigbee cluster-tree wireless sensor networks," in *2013 10th International Multi-Conference on Systems, Signals and Devices (SSD)*, March 2013, pp. 1–8.

[15] R. Diestel, *Graph Theory*. Springer-Verlag, 2005.

[16] C.Hanen and Z. Hanzálek, "Grouping tasks to save energy in a cyclic wireless sensor network scheduling problem," in *Proceedings of the 6th Multidisciplinary International Conference on Scheduling: Theory and Applications*, Aug. 2013.