Minimization of the Diffusion Delay of a Tree-Based Wireless Sensor Network

François Delobel, Alexandre Guitton, Michel Misson
Clermont-Ferrand University, LIMOS CNRS
Complexe scientifique des Cézeaux
63177 Aubière cédex, France
Email: {delobel,guitton,misson}@sancy.univ-bpclermont.fr

Waltenegus Dargie
Chair for Computer Networks, Faculty of Computer Science
Technical University of Dresden
01062 Dresden, Germany
Email: {waltenegus.dargie}@tu-dresden.de

Abstract—In wireless sensor networks, saving energy is crucial in order to increase the network lifetime. Energy is often saved by synchronizing the nodes activity, and having long periods of inactivity, or by having nodes exchange a global activity schedule. The synchronization and the exchange of a global schedule are two examples where an information is broadcasted from a specific node to the whole network. In this paper, we focus on the delay required to broadcast an information in the whole network using a tree topology. We first show that the diffusion delay can be significantly reduced by utilizing the parallelization of node processing. We provide an exact algorithm in order to find optimal solutions. Then, we propose a linear algorithm that is able to find good solutions. We compare the exact solution to the heuristic solution on a workstation and conclude that our heuristic is very competitive and can be used to reduce the diffusion delay of a broadcasted frame in a tree.

Keywords: wireless sensor network, synchronization delay, diffusion delay, tree topology, transmission scheduling.

I. INTRODUCTION

Wireless sensor networks (WSNs) are increasingly considered for a large variety of applications, including the monitoring of vast areas and military surveillance [1]. WSNs are composed of small devices, called motes, which are battery powered. A mote is a platform which has a set of sensors, limited computational capabilities, a short-range communication module and a small-sized memory.

The main objective of a WSN is to monitor an external phenomenon during a long time. The critical issue is often to reduce the energy used by the network protocols while being able to follow the evolution of the phenomenon. This is especially challenging as motes spend about as much energy when listening (or receiving) than when transmitting [2], [3], [4]. Motes have additional specificities compared to traditional computers. They often use a processor with limited computing capabilities, which yields to large processing times. Additionally, they are often based on a dual architecture (radio and processor) communicating through a SPI interface bus, which further increases the processing time.

In most WSNs, a global synchronization mechanism is often used to address the energy consumption issue. It consists in having motes share a common temporal basis. Once motes are synchronized, they can coordinate their activities, which helps reducing the energy used in several ways: (i) motes can schedule their inactivity period so that they are all sleeping as much

as possible, (ii) motes can minimize overhearing by listening only when a sender is transmitting, (iii) motes can refrain from transmitting simultaneously, which reduces collisions. Global synchronization is often achieved by broadcasting a beacon message on a tree structure. The beacon may contain global information about the schedules of the motes.

In this paper, we minimize the time required to broadcast a message to all the motes of a WSN. We show that the dual architecture of typical motes has an important impact on the diffusion delay, and we propose algorithms to reduce it. Our solution is generic and can be applied in order to broadcast a global information in a network, or to minimize the synchronization time for example. In the latter case, reducing the synchronization time is critical as it is an overhead for the nodes. The shorter the synchronization, the less energy is spent.

The plan of the paper is the following. In Sect. II, we characterize the diffusion delay on motes architectures and we formally describe the problem of minimizing the diffusion delay. In Sect. III, we present two exact solutions for the problem: with an integer linear program and with a branch-and-bound method. In Sect. IV, we propose an approximation of the problem which runs in a linear time with respect to the number of nodes in the network. This algorithm is implemented on a real mote platform. We conclude our work in Sect. V.

II. PROBLEM DESCRIPTION

Several protocols in WSNs are based on a tree topology, as it is easy to maintain when nodes mobility is low. Several synchronization protocols for WSN are described in [5].

The RBS protocol (Reference Broadcast Synchronization [6]) is a tree-based, receiver-to-receiver synchronization protocol. The authors of RBS show that the time between the transmission of a message and the reception of the message is subject to random delays, due to random access mechanisms or unpredictable processing time. The synchronization in RBS is only performed among receivers, and not with the sender of a message. This approach eliminates the time uncertainty at the sender compared with traditional sender-to-receiver synchronization protocols.

The TPSN protocol (Timing-sync Protocol for Sensor Networks [7]) is a tree-based, sender-to-receiver synchronization protocol. Nodes in TPSN are activated sequentially, depending on their depth on the tree. Nodes in TPSN estimate the clock drift using a handshake procedure with their parent. The main advantage of TPSN over RBS is that it consumes less energy, at the cost of a reduced accuracy.

The IEEE 802.15.4 standard [8] standard defines the lower layers of a low power wireless personal area network. The ZigBee standard [9] defines the upper layer of a low power wireless personal area network, and assumes that the lower layers operate IEEE 802.15.4. In ZigBee, motes are either coordinators or end-devices. The motes form a tree topology called the cluster-tree, where end-devices are leaves, coordinators are internal nodes, and the root is a special coordinator called the PAN (personal area network) coordinator. In the following, we use the ZigBee definitions of coordinators and end-devices.

A. Diffusion delay of a broadcasted frame

Synchronization protocols (or, more generally, diffusion protocols) are often based on a sequence of broadcasting. The interval between two broadcasts depends mostly on the processing time of the node. It also depends on the architecture of the mote. Indeed, motes are often built according to a dual architecture: a radio module (e.g. a CC2420 component [2]) and a processor (usually operating at 4 MHz or 8 MHz), interconnected through an SPI interface at 500 kbps. As the commonly used physical layer of IEEE 802.15.4 is operated at 250 kbps, the time required to transmit a frame through the SPI interface is half the time required to transmit the frame on the medium. The processing time, the possible SPI communication time, and the transmission time have significant impact on the overall diffusion delay of a broadcasted frame.

In order to compute the diffusion delay of a broadcasted frame, we assume that the frame is broadcasted in a multi-hop manner by all the coordinators of the network. As end-devices can only receive from their designated parent, all the coordinators of the network have to transmit. For simplicity reasons, we do not show in the rest of this paper the end-devices.

We assume that nodes transmit the frame sequentially. This is critical in order to guarantee that there is no collision between the frames, which would dramatically increase the delay. Allowing distant nodes to transmit simultaneously is an issue in WSNs [10], as the location of nodes is often unknown and propagation conditions may vary due to the mobility of nodes or to changes in the environment.

The PAN coordinator of the network computes a sequence containing all the coordinators of the network, and includes it in the diffusion frame. Then, it broadcasts the frame. When a coordinator n receives the diffusion frame from a node s, it determines the current relative time from the location of s in the sequence. Then, it computes its own transmission time based on the number of nodes between s and n in the

sequence[1].

Figure 1 shows an example topology of five nodes: node A is the PAN coordinator, and the other four nodes are coordinators. These coordinators have end-devices attached to them, but they are not depicted on the figure.
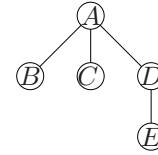

Figure 1. An example topology for the diffusion.

Figure 2 describes how time is spent by nodes during a diffusion of the frame, when the sequence is (A, B, C, D, E) and the topology is the one given on Fig. 1. First, A builds the frame in the main processing module and potentially sends it to its radio module through the SPI interface, which is depicted as a box with an arrow. Then, A transmits the frame. Simultaneously (the time of flight of frames being usually negligible in WSNs), the frame is received by B, C and D. The radio modules of each of these coordinators send the frame to the processing module (which is depicted as a box with an arrow) and they all process it (which is depicted as a box with PR). Coordinator B detects that it is its turn, and directly transmits the frame to its radio module. During this time, coordinators C and D are waiting. Coordinator C wakes up after B has completed the transmission[2]. The process goes on until all the coordinators have sent the frame. Note that in the figure, we set the transmission time on the SPI interface to half the time of the transmission on the channel. We also assumed a small processing time PR, which might not be the case in practice.
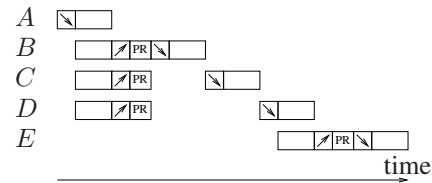

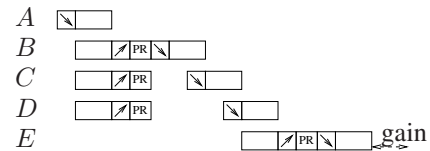Figure 2. Non-optimized diffusion delay for a given sequence.

Figure 3. Optimized diffusion delay for a given sequence.

Because the time required to send a frame from the processing module to the radio module is known (it only depends on

---

[1]The transmission duration is not the same for each node. At the moment, we assume that n can determine it from the sequence itself.

[2]Again, we assume here that C knows the time B is going to spend transmitting. We do not assume that C is listening to B.

the bandwidth of the SPI interface and on the length of the frame), coordinators can send the frame in advance. In the previous example, coordinator C does not need to wait for B to complete the transmission before sending its frame to the radio module. Indeed, coordinator C can wake up a short time before B completes the transmission, and C can start sending the frame to the radio module. Figure 3 shows an optimized version of the diffusion, with the same sequence and the same topology. The gain comes from the optimization at coordinators C and D.

Let us give a numerical estimation of the different times involved. The transmission of a medium-sized frame of 50 bytes on the wireless channel is performed at 250 kbps. Thus, it takes about 1.6 ms. The transmission of the frame on the SPI interface takes about 0.8 ms. The processing time of a frame on a mote takes about 1 ms when basic operations are performed.

B. Optimization of the diffusion delay

It can be seen on the previous example that a relatively long time is spent between coordinators D and E. When E receives the frame from D, it processes it, detects that it directly follows D in the order and send the frame as soon as possible. During this time, the medium is not used at all. In fact, all the coordinators are waiting for E to receive and process the frame. The delay would have been reduced if a coordinator had been between D and E in the order. In this way, E would have processed the frame while another coordinator was using the medium to send its frame.

Figure 4 shows the diffusion delay on the topology of Fig. 1, using the sequence (A, D, B, C, E). Notice that, in this sequence, E does not directly follow D. As can be seen, the reduction of the diffusion delay is very important compared to the initial sequence, due to the parallelization of the activities. While E processes the message, other coordinators are using the channel. The proposed sequence is optimal.
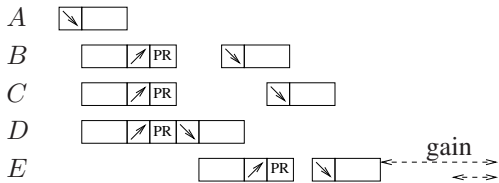

Figure 4.   Optimized diffusion delay for an optimal sequence.

C. Model and definitions

In this paper, we study how to build an optimal sequence for the diffusion of a frame, given a tree.

Let T be a tree representing a WSN, where N is a set of $|N| = n$ nodes. Let $O(T)$ be a sequence on T, representing the order of emission of the node. In the remaining part of the paper, we call $O(T)$ an *order*. As each node needs to wait for a beacon before being able to transmit, $O(T)$ has to be a *topological* order. We use the following notations. $father(n)$ is the node n' such that n' is the father of n in (T), $pred(x)$

(resp. $succ(d)$) is the node before (resp. after) x in an order $O(T)$, and $first(d)$ (resp. $last(d)$) is the first (resp. last) node of depth d that appears in $O(T)$.

In the previous subsection, we showed that the presence of a child directly after its father in an order has a negative impact on the delay. We call this situation a *conflict*. For a topological order $O(T)$ and a node x, we define the predicate $conflict(x)$ which is true if and only if $pred(x) = father(x)$. A conflict in an order $O(T)$ for a node x is represented by underlining x.

The diffusion delay depends on the number of nodes in the network, and on the number of conflicts. As the number of nodes in the network is fixed, we define the *duration* of an order $O(T)$ as the number of nodes x in T for which $conflict(x)$ is true. We use the notation $d(O(T))$ to denote the number of conflicts in an order $O(T)$.

Note that a depth-first order of T yields longest durations than a breadth-first order of T. In the following, we consider that a breadth-first order is a good solution.

III. EXACT SOLUTION

We propose in this section two algorithms in order to obtain exact solutions. The first is an integer linear program. The second is a branch and bound algorithm.

A. Integer linear program

A way to compute an optimal order $O(T)$ for a tree T is to define an integer linear program. Our program uses a set of m nodes denoted by N. A set P is used to represent the positions of the nodes in an order. Finally, father(n) denotes the father of a node n on the tree (we assume here that the father of the root node is the root node itself).

We also define the following relations. pos(n) represents the position of a node n in an order. permut(n, p) is a boolean matrix which describes to which position p a node n is mapped in an order. $\delta(n_1, n_2)$ is the difference of positions of nodes n_1 and n_2 in the order, plus m. The addition of m ensures that $\delta(n_1, n_2) \geq 0$, which is required later on (see Subsect. III-A2). conflict(n) indicates whether node n directly follows its father in the order, which induces a conflict, or not. Finally, the real variables λ_i and the binary variables x_i are used in order to model conflicts (see Subsect. III-A2). Note that our modelization is a mixed integer linear program due to the requirement of the variables λ_i to be real.

1) Modelization of the constraints: We use the following variables of the mixed integer linear program in our modelization. pos(n) is the position of n in the order (in $\{1, \ldots, P\}$). permut(n, p) indicates whether n is at position p or not (in $\{0, 1\}$). $\delta(n_1, n_2)$ is the difference of positions of nodes n_1 and n_2 in the order, plus m (in $\{0, \ldots, 2m\}$). conflict(n) indicates whether n directly follows its father in the order or not ($\{0, 1\}$). $\lambda_i(n)$ is in $\mathbb{R}^+$ and $x_i(n)$ is in $\{0, 1\}$.

Constraints 1 and 2 in Table I ensure that the order is a permutation of nodes (through the use of permut(n, p)). Constraint 3 attributes to each node n its position pos(n) in the order, while Constraint 4 ensures that the order is a topological

Minimize $\sum_{n \in N} \text{conflict}(n)$ such that:
(1) $\forall n \in N, \sum_{p \in P} \text{permut}(n, p) = 1$
(2) $\forall p \in P, \sum_{n \in N} \text{permut}(n, p) = 1$
(3) $\forall n \in N, \text{pos}(n) = \sum_{p \in P}(p \cdot \text{permut}(n, p))$
(4) $\forall n \in N, \text{pos}(n) \geq \text{pos}(\text{father}(n))$
(5) $\forall n_1, n_2 \in N^2, \delta(n_1, n_2) = \text{pos}(n_1) - \text{pos}(n_2) + m$
(6) $\forall n \in N, \text{conflict}(n) = \lambda_3(n)$
(7) $\forall n \in N, \delta(n, \text{father}(n)) = m\lambda_2(n) + (m+1)\lambda_3(n) + (m+2)\lambda_4(n) + 2m\lambda_5(n)$
(8) $\forall n \in N, \lambda_1(n) \leq x_1(n)$
(9) $\forall n \in N, \forall i \in \{2, 3, 4\}, \lambda_i(n) \leq x_{i-1}(n) + x_i(n)$
(10) $\forall n \in N, \lambda_5(n) \leq x_4(n)$
(11) $\forall n \in N, \sum_{i \in \{1,2,3,4\}} x_i(n) = 1$
(12) $\forall n \in N, \sum_{i \in \{1,2,3,4,5\}} \lambda_i(n) = 1$

Table I
CONSTRAINTS OF THE MIXED INTEGER LINEAR PROGRAM.

sort of the nodes. Finally, Constraint 5 attributes the correct value to $\delta(n_1, n_2)$.

2) Modelization of the conflict(n) *constraint:* Constraints 6 to 12 are used to model conflicts. Recall that a conflict occurs for a node n if and only if n directly follows its father in the order. This means that the difference between $\text{pos}(\text{father}(n))$ and $\text{pos}(n)$ is equal to 1. Thus, if we denote by $d(n)$ the difference of positions between n and $\text{father}(n)$ in the order, we can define conflict(n) in the following way: conflict$(n) = 1$ if $d(n) = 1$, and 0 otherwise. Unfortunately, conflict(n) is not a convex function of $d(n)$.

conflict(n) can be defined as a step function of a positive variable $\delta(n)$, with $\delta(n) = d(n) + m$. Formally, conflict(n) can be defined in the following way. Let $a_1 = 0$, $a_2 = m$, $a_3 = m + 1$, $a_4 = m + 2$ and $a_5 = 2m$, and conflict$(n)(a_1) = 0$, conflict$(n)(a_2) = 0$, conflict$(n)(a_3) = 1$, conflict$(n)(a_4) = 0$ and conflict$(n)(a_5) = 0$. Such a formulation can be used to model conflict(n) using a mixed integer linear program, as described in Ineq. 4.18 of [11] (notice that a_1 has to be greater than or equal to 0, and thus our addition of m in $\delta(n)$). Constraints 6 to 12 follow from this formulation.

The integer linear program described in the previous subsection is computationally expensive. Some instances of twenty nodes are solved in several hours by a workstation using GLPK (GNU Linear Programming Kit). Indeed, the modelization of the conflicts and the number of integer variables make the problem hard to solve. Thus, we decided to implement a branch and bound algorithm to find exact solutions quickly.

B. Branch and bound algorithm

A branch and bound algorithm [12] finds an optimal solution by exploring a tree $\mathcal{T}$ of all possible solutions. The quality of each solution is computed while exploring the tree. If the algorithm determines that all the solutions in a sub-tree have a lower quality than an existing solution, the algorithm stops the evaluation of the sub-tree, thus saving processing time. Therefore, the efficiency of a branch and bound algorithm comes from two criteria: (i) the quality of the initial solution, and (ii) the relationship between the quality of a solution $s \in \mathcal{T}$ and the quality of the solutions in the sub-tree of s.

The tree $\mathcal{T}$ explored by the branch and bound algorithm is an n-ary tree. $\mathcal{T}$ is built in the following way. Each node of the tree corresponds to an order. The root corresponds to the empty order. If a node of $\mathcal{T}$ corresponds to an order o of k nodes, it has $n - k$ children in $\mathcal{T}$. The i-th child corresponds to the order $o \cdot n_i$, where n_i is the i-th node of N not present in o_n. Tree $\mathcal{T}$ has $n!$ leaves, which are all the possible orders of the n nodes. Not all the possible orders are valid (as some of them are not topological), but the optimal order is one of the leaf of $\mathcal{T}$.

The *topological rule* ensures that orders that are not topological are not examined, as they cannot correspond to a valid solution. The *branching rule* assumes that the best current solution (possibly not optimal) is known and cuts subtrees of $\mathcal{T}$ whose duration is larger than the current best. The key here is that the duration of an order o_n is always smaller than or equal to the duration of any topological order $o_n \cdot n_i$.

The choice of the initial reference solution is crucial. We used our heuristic described in Sect. IV as a good initial solution.

Breadth-First algorithm (BF) is compared to the optimal solution (computed with Branch and Bound) using two types of generated trees: random trees and interconnection of random hot-spots.

1) Random trees: Random trees are designed to model interconnections between motes in a dense WSN. Nodes are added to the tree one by one. They are attached to any existing node of the tree, except to those having already more than R_m children. Note that this limitation is consistent with the cluster-tree topology formation of ZigBee [9]. Motes locations are not generated by this algorithm, because only tree structures have an impact on the order duration. Note that with this tree generation algorithm, trees tend to become complete rather than grow deeper.

In our simulations, we used $R_m = 5$. We varied the number of nodes from 50 to 2000. For each number of nodes, we generated 1000 trees. Generating all the trees and finding the optimal solution using the branch and bound algorithm takes about half an hour on an i7 930 workstation.

The percentage of trees where the breadth-first order was not optimal, varies between 15% and 21.1% (on average 17.9% with a standard deviation of 1.35). It does not depend on the number of nodes (from node numbers varying from 50 to 2000). The average duration of breadth-first orders, when they are not optimal, is on average about 2.17 (while the optimal duration is on average 1) with a standard deviation of 0.03. These results do not change as the number of nodes vary. Consequently, breadth-first orders are good approximations of the optimal order when random trees are generated.

2) Interconnection of random hot-spots: When the network is dense, it is possible to find optimal solutions having a very low duration. We consider here a network constituted of two hot-spots, interconnected by a chain of nodes. The root of the

tree is located in one of the hot-spot[3]. The generator first places x motes with sensors within the hot-spots, then adds as many motes as required in order to build a tree in each hot-spot, and interconnects the two hot-spots by deploying a chain of motes. The generated tree includes only the non sensor motes.

The topologies were generated using the following parameters. The communication range of motes is about 15 m. Each hot-spot has a diameter of $40 + \frac{x}{2}$ m, and consists of x sensors. The distance between the two hot-spots is about $8 * x$ m. The coordinator is added to the center of one hot-spot. We varied x from 5 to 20, and we generated 100 trees for each x. Note that the number of nodes n depends on x. Note that with the trees generated, the breadth-first order duration is usually far away from the optimal duration, which causes the branch and bound algorithm to take hours to solve instances of 60 nodes.

Figure 5 shows the average duration of Breadth-First and optimal orders, as x varies. It can be that the duration of the breadth-first order grows proportional to x, which is explained by the fact that the hot-spots are separated by a distance proportional to x. Variations in the breadth-first duration are due to different tree topologies. The optimal order yields durations of 1 on average, which means that the nodes of the first hot-spot are used by the optimal order to break the chain of nodes. If the two hot-spots were separated by a larger distance (for instance, if the distance between hot-spots exceeds $15x$), the optimal duration could not be equal to 1.
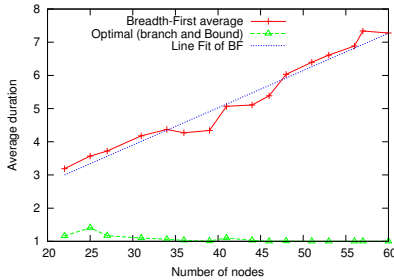

Figure 5. Average duration of breadth-first and optimal orders, for interconnections of hot-spots.

IV. HEURISTIC METHOD

In this section, we present an algorithm that approximates the optimal order in linear time. The general idea for this algorithm is to improve a breadth-first order by permuting and moving nodes in the order without changing the tree topology.

A. Permuted order

Let us consider a breadth-first order. Such an order can cause at most one conflict per depth d. If there is a conflict at depth d, we apply one of the three permutation rules. Permutation P_1 is applied iff there are at least three nodes on depth d. Permutation P_1 consists in swapping $succ(first(d))$ and

---

[3]Such a topology is a worst-case scenario for the duration of the order. Indeed, when a tree possesses a single, sufficiently long chain of nodes, all topological orders contain the nodes of the chain in sequence, and the optimal duration increases significantly.

$last(d)$ in the current order. Figure 6 shows Permutation P_1 applied on an example. Permutation P_2 is applied iff (i) there are exactly two nodes on depth d and (ii) $first(d)$ and $last(d)$ have the same father. Permutation P_2 consists in swapping $first(d)$ and $last(d)$ in the current order. Permutation P_3 is applied iff (i) there are exactly two nodes on depth d and (i) $first(d)$ and $last(d)$ have a different father. Permutation P_3 consists in swapping $first(i)$ and $last(i)$ for each depth i such that (i) $j < i \leq p$, (ii) there are exactly two nodes on depth i that have a different father and (iii) at depth j, there is either one node, two nodes with the same father or three or more nodes. If there are two nodes at depth j, they are swapped. If there are three or more nodes at depth j, a node n_j^x that is neither the father of $first(j+1)$ nor of $last(j+1)$ is swapped with node $last(j)$. Figure 7 shows Permutation P_3 applied on an example. A *permuted order* $P(T)$ can be computed by first considering permutations P_1 and P_2 from depth 1 to h, and then considering permutations P_3 from depths 1 to h.
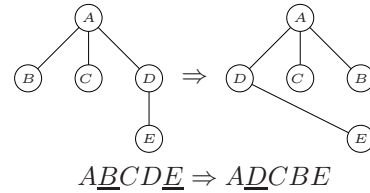

$$A\underline{B}CD\underline{E} \Rightarrow A\underline{D}CBE$$

Figure 6. Permutation P_1 reduces by one the duration of the order.

$$A\underline{B}CDEFGH\underline{I} \Rightarrow A\underline{D}CBFEHGI$$
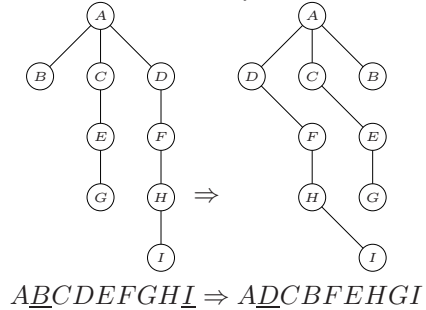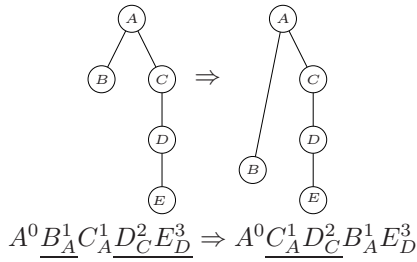
Figure 7. Permutation P_3 reduces by one the duration of the order.

B. Displaced order

The displaced order enhances the permuted order by removing the constraint of having nodes sorted by depths. Let $O(T)$ be a topological order. If $last(d)$ is the father of $first(d+1)$ in $O(T)$, and if depth d contains only one node $last(d)$, the displaced order moves an available node for $last(d)$ between $last(d)$ and $first(d+1)$. A node n_x is said to be *available* for a node $last(d)$ if it satisfies the following conditions: (i) n_x is directly preceded by node n_a and directly followed by node n_b, (ii) n_x has no child in the tree T (so that the order remains topological), (iii) n_x is before $last(d)$ in the order $O(T)$, and (iv) either n_a is the father of n_x and of n_b, or n_a is the father of neither n_x nor of n_b. Figure 8 shows an example of a displaced order, where the available node is B. A *displaced order* $D(T)$ is computed by considering displacements from depths 1 to h. Our heuristic consists in building $D(T)$.

Figure 8. Displacement rule D reduces by one the duration of the order.

$$A^0 \underline{B^1_A} C^1_A \underline{D^2_C E^3_D} \Rightarrow A^0 \underline{C^1_A D^2_C} B^1_A E^3_D$$

The computation of $D(T)$ is realistic for real motes, as it only requires a time which is a linear function of n, as follows. The breadth-first order $B(T)$ can be computed in $\mathcal{O}(n+m)$, where n is the number of nodes of T and m is the number of edges of T. As T is a tree, $m = n-1$. Thus, $B(T)$ can be computed in $\mathcal{O}(n)$. The (potential) application of permutations P_1 and P_2 at each depth requires $\mathcal{O}(h)$. The (potential) application of permutations P_3 at each depth also requires $\mathcal{O}(h)$: although P_3 considers swapping nodes at previous depths, it is not possible for the same node to be considered twice. The overall complexity for $P(T)$ is $\mathcal{O}(n+h) = \mathcal{O}(n)$.

The set $\mathcal{D}$ of all the available nodes, with respect to the last node of $P(T)$, can be computed in $\mathcal{O}(n)$. When the algorithm searches for an available node for $last(d)$, only the first node of $\mathcal{D}$, denoted n_x, needs to be considered. If n_x is not available for $last(d)$, this means that no further nodes in $\mathcal{D}$ are available for $last(d)$. If n_x is available, $\mathcal{D}$ has to be updated. As n_x has no children in T (due to the second constraint of the available nodes), the changes in $\mathcal{D}$ are limited to n_a and n_b, which are the neighbor nodes of n_x. In other words, once set $\mathcal{D}$ is computed (in linear time), it only decreases (in constant time). Thus, computing $D(T)$ requires $\mathcal{O}(n)$.

C. Simulation results

In this section, we study the performance of our heuristic with respect to breadth-first and and optimal approaches.

With the same random trees as previously, less than 0.8% of heuristic solutions are not optimal (on average 0.25% with standard deviation 0.16%). Note that heuristic solutions are at least as good as breadth-first solutions.

Figure 9 depicts the average duration of breadth-first, heuristic and optimal orders on small interconnected hot-spots. The figure shows that the heuristic improves greatly the duration of the breadth-first order, keeping the duration constant and close to the duration of the optimal solution. Figure 10 depicts the average duration of breadth-first and heuristic orders on large interconnected hot-spots, where the number of sensors varies from 0 to 2000, and 1000 instances are generated per size. Note that we were not able to run the branch and bound algorithm on such topologies. The duration of the heuristic order remains nearly constant (most of the conflicts of the breadth-first orders can be repaired), whereas the duration of

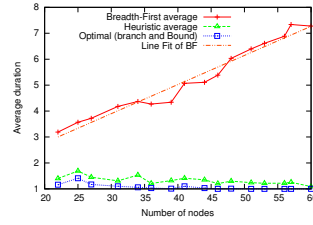the breadth-first order grows proportionally with the number of nodes.

Figure 9. Average duration of breadth-first, heuristic and optimal orders, on small interconnected hot-spot trees.
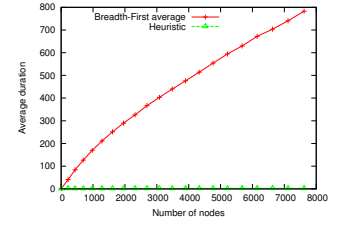
Figure 10. Average duration of breadth-first and heuristic orders, on large interconnected hot-spot trees.

V. CONCLUSION

Several protocols developed for WSNs require the diffusion of an information in the whole network in a guaranteed manner. In this paper, we study the delay required to propagate the information to all nodes. We show that the order in which nodes transmit the data has a critical impact on the overall delay. Consequently, we propose two exact algorithms and a heuristic one in order to reduce the diffusion delay. Simulations are performed on randomly generated trees and on randomly generated interconnection of hot-spots. We show that our heuristic is applicable in real scenarios and that it leads to near-optimal diffusion delays.

In a future work, we plan to adapt our study to graphs. We aim at finding the optimal diffusion order when the complete communication graph is given, rather than when a tree is given. We also plan to implement our heuristic on TelosB motes in order to check the computation cost and memory requirements for relatively large networks.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, 2001.
[2] "CC2420 - 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver," Chipcon, preliminary datasheet, 2004.
[3] J. Elson and K. Römer, "Wireless sensor networks: A new regime for time synchronization," in *Hot Topics in Networks*, 2002.
[4] J. Rahmé, N. Fourty, K. Al Agha, and A. Van Den Bossche, "A recursive battery model for nodes lifetime estimation in wireless sensor networks," in *IEEE Wireless Communications and Networking Conference*, 2010.
[5] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, "Clock synchronization in wireless sensor networks: A survey," *Ad-Hoc Networks*, vol. 3, no. 3, pp. 281–323, 2005.
[6] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *OSDI*, December 2002.
[7] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *ACM SenSys*, 2003, pp. 138–149.
[8] IEEE 802.15, "Part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs)," ANSI/IEEE, Standard 802.15.4 R2006, 2006.
[9] ZigBee, "ZigBee Specification," ZigBee Standards Organization, Standard ZigBee 053474r17, January 2008.
[10] X. Wang, "Spatial channel reuse in wireless sensor networks," *Wireless Networks*, vol. 14, no. 2, pp. 133–146, 2008.
[11] A. Billionnet, *Optimisation discrète*. Dunod, 2007.
[12] A. H. Land and A. G. D. Doig, "An automatic method of solving discrete programming problems," *Econometrica*, vol. 28, no. 3, pp. 497–520, 1960.