Delay Optimized Time Slot Assignment for Data Gathering Applications in Wireless Sensor Networks

Ichrak Amdouni, Ridha Soua, Erwan Livolant, Pascale Minet
INRIA Rocquencourt, 78153 Le Chesnay cedex, France
Email: firstname.name@inria.fr

***Abstract*—Wireless sensor networks, WSNs, are an efficient way to deal with low-rate communications in confined environments such as mines or nuclear power plants because of their simplicity of deployment and low cost. In these application domains, WSNs are used to gather data from sensor nodes towards a sink in a multi-hop convergecast structure. In this paper, we focus on a traffic-aware time slot assignment minimizing the schedule length for tree topologies and for two special deployments (i.e. linear and multi-linear) representative of unusual environments. We formalize the problem as a linear program and provide results on the optimal number of slots. We then propose a delay optimized algorithm with two heuristics that minimize on the one hand the energy consumption and on the other hand the storage capacity as secondary criteria.**

I. Context and motivations

The spectacular progress in miniaturization of radio and sensor technologies contributes to the development of wireless sensor networks, in short WSNs. WSNs are made of wireless sensor nodes, each of them being equipped with sensing devices, a processing unit and a radio module. Sensor nodes are generally deployed to monitor the physical conditions of their environment. Therefore, WSNs have been frequently put into use for data gathering applications, where sensed data are routed over a tree. Such a communication scheme is also called convergecast. We can cite many examples of automated data convergecast applications like environment monitoring, building automation, military field monitoring, industrial process control, precision agriculture, ehealth, to name a few. In these applications we speak about *raw-data convergecast* if every packet is forwarded individually. In such a case, intermediate nodes in a data gathering tree simply apply the store and forward strategy, without processing the received packets.

Most of these applications share the requirement of deterministic delay bounds and a guarantee on packet delivery. Medium access protocols that are contention-based protocols are clearly inadequate as they suffer from collision and non deterministic delays especially under heavy traffic conditions. Time Division Multiple Access, TDMA, is a contention-free protocol where nodes share the same channel and time is divided into time slots. Each node transmits data in its allocated slots. Slots are usually organized in a frame which is repeated periodically. Hence, it is obvious that the TDMA protocol is well adapted for collision-free packet transmission with QoS support. Furthermore, the TDMA deterministic scheduling is energy efficient. It avoids collisions that waste energy, does not need idle listening and allows low power devices to turn off their radio in time slots not allocated to them. In this work, we are interested in TDMA based scheduling for raw-data convergecast, assuming that each time slot contains only one message. We observe that nodes close to the sink have a higher traffic demand. Hence granting an equal number of slots to each node is not adequate. That is why we investigate the problem of convergecast scheduling that ensures to any node a channel access that is proportional to its traffic demand. Our objective is to find a time optimal traffic-aware slot assignment. Indeed, minimizing the number of slots in the TDMA cycle contributes to improve the network performances. First, the maximum packet delay being equal to one TDMA cycle, dividing the TDMA cycle duration helps to meet the time constraints of packets. This property is crucial for applications with strong time constraints. Second, the throughput measured at the sink is the number of slots granted to the children of the sink to send their packets divided by the TDMA cycle duration in slots and multiplied by the network capacity. Decreasing the TDMA cycle by a factor multiplies the throughput by the same factor. Third, given a data gathering period consisting of activity and inactivity periods, minimizing the schedule length reduces the activity period of nodes and allows them to save more energy.

This paper encompasses the following contributions:
○ After a brief state of the art in Section II, we define in Section III the Time Slot Assignment problem denoted *TSA* in WSNs. We then provide a formulation of this problem as an Integer Linear Programming (ILP) optimization problem.
○ We determine in Section IV theoretical lower bounds for *TSA* in linear, multi-linear and tree topologies.
○ We then present $TRASA$, our TRaffic Aware Slot Assignment algorithm. We describe in Section V the properties of the algorithm. We prove that it is time optimal for linear networks. Then, we evaluate its performance in Section VI.

II. Related work

We detail in this section only scheduling protocols that are traffic aware: each node is assigned a number of slots equal to its traffic demand.

Incel et al. present Local-TimeSlotAssignment algorithm for raw data convergecast in [1]. Their key idea is to keep the sink busy in receiving packets. Authors have proven that the schedule length obtained when interfering links are eliminated is lower bounded by $max(2n_k - 1, N - 1)$ where n_k is the

978-1-4673-1291-2/12/$31.00 (c) 2012 IEEE

maximum number of descendants of sink children and N is the number of network nodes. Their algorithm provides the minimal schedule length in this case. Their algorithm does not support immediate acknowledgement (i.e. the receiver acknowledges the received packet in the slot of the sender). TreeMAC, [3], aims at achieving high throughput. The sink assigns to each child a number of contiguous frames equal to the number of its descendants such that any two children have different frames. Each child repeats this process for its children. Furthermore, each frame comprises three colored slots. Each node computes its pseudo-level, equal to its distance to the sink minus one modulo 3. Nodes with the same pseudo-level have the same colored slot. Hence, the number of slots is equal to $3(N-1)$, where N is the number of nodes. RoMAC, proposed by Huang et al. [2] is an enhancement of TreeMac using nodes identifiers to assign slots to nodes. Each source node i sends its own data in the i^{th} slot of the i^{th} frame in the TDMA cycle, whereas each intermediate node j forwards the received data from node i in the j^{th} slot of the i^{th} frame. Virtual identifiers can be allocated to nodes, the number of virtual identifiers assigned to a node is proportional to the traffic demand of this node.

SPR in [4] requires a small number of buffers which is a key challenge in WSNs. In its basic form, SPR considers the routing tree as an overlay of the paths from each leaf to the sink. Each path is scheduled separately and each packet generated by a source is pipelined to the sink.

In [5], authors introduced a novel level based scheduling which minimizes the schedule length. First, a linear network is built from the initial network. Each node in the latter corresponds to a level in the original network. Second, the schedule of the original network can be deduced by considering the coloring of the linear network. Authors have proved that for linear or tree networks when (1) each node has one packet to transmit (2) and only nodes that hear each other interfere, the maximum number of slots obtained by their algorithm is $3N-3$ where N is the number of nodes. For a general tree graph $G=(V,E)$ and interference graph $C=(V,I)$, the maximum number of slots obtained by the level-based scheduling is $\alpha(N-1)$, where α is the number of colors used in the linear network corresponding to G and C.

In [6], Gandham et al. have computed theoretical lower bounds on the number of slots required in linear, multi-line and tree network topologies. The algorithm they propose is close to these bounds in case of linear and multi-line topologies.

Zhang et al. have proposed in [7] an algorithm that achieves one slot more than the optimal in case of linear topologies.

III. The Time Slot Assignment Problem

In this section, we define the Time Slot Assignment problem, denoted *TSA*.

A. Assumptions

◦ **A1. Data gathering applications and sink tree:** In data gathering applications, a node, called sink or gateway, is in charge of collecting data sent by all other nodes. Hence, the typical traffic pattern is many-to-one routing and leads to a spanning tree T rooted at the sink node.

◦ **A2. Application data:** In each data gathering cycle, each node except the sink has its own data to transmit to its parent in addition to the data received from its children. Some nodes (for example, the children of the sink), need more than one slot to transmit their data.

◦ **A3. Time slot:** The time slot duration must allow the transmission of at least one packet and its acknowledgement. During the slot, the sender transmits its packet to its receiver. This receiver, a 1-hop neighbor of the sender, acknowledges the received packet in the same time slot.

◦ **A4. Conflicting nodes:** Two nodes are said *conflicting* if and only if they cannot transmit in the same time slot. For any given node u, the set of nodes conflicting with u is an input for the time slot assignment problem.

◦ **A5. No message loss and no node failure.**

B. Problem statement

The time slot assignment problem, *TSA*, under the assumptions introduced in Section III-A, consists in assigning slots to network nodes, such that no two conflicting nodes are scheduled in the same slot while minimizing the schedule length. Besides, this scheduling must ensure that each node transmits towards the sink, both its own packets and the packets generated in its subtree. To summarize, our aim is to build a minimal valid scheduling.

*Definition 1 (**Valid scheduling**):* A scheduling is said valid if and only if:

- any node is assigned a number of slots sufficient to transmit all its traffic.
- any node is assigned a slot if and only if it has data to transmit during this slot.
- any two conflicting nodes do not transmit in the same time slot.

*Definition 2 (**Minimal scheduling**):* A valid scheduling is said minimal if and only if no other valid scheduling has a smaller number of time slots.

C. Formalization of the Time Slot Assignment problem

The network is modeled as a graph $G=(V,E)$, V is the set of vertices representing network nodes, and E is the set of edges representing the communication links. Let $V = V_s \bigcup V_g$, where V_g and V_s representing respectively the set of sinks (gateways) and the set of source nodes in the network, with $V_s \bigcap V_g = \emptyset$. For any source s, let p_s denote its demand which corresponds to the number of packets that it has to transmit in the TDMA cycle. For each node $v \in V$, we define $I(v)$ the set of nodes that interfere with v. Let $E^+(v)$ denotes the set of links through which v can transmit. $E^-(v)$ is the set of links through which v can receive data.

For any link e, we define a_e^t the activity of e in the slot t, in other words $a_e^t = 1$ iff there is at least one transmission of a packet on the link e in the time slot t. Furthermore, let u_t be the use of a slot t, ie $u_t = 1$ iff there is an activity of at least one link on the slot t. Finally, f_e^s denotes the set of packets generated by the source s using the link e in a TDMA cycle.

The objective is to obtain the minimum total number of slots: $min \sum_{t \leq T_{max}} u_t$, t being in $[1, T_{max}]$ interval.

T_{max} denotes the maximum length of TDMA cycle. This objective is subject to:

$$a_e^t \leq u_t \quad ,\forall e \in E, t \leq T_{max} \tag{1}$$

$$\begin{aligned} a_e^t + a_{e'}^t \quad &\leq 1 \quad ,\forall v \in V, \forall e \in E^+(v), \\ &\forall w \in I(v), \forall e' \in E^+(w), \\ &t \leq T_{max} \end{aligned} \tag{2}$$

$$\sum_{s \in V_s} f_e^s \leq \sum_{t \leq T_{max}} a_e^t \quad ,\forall e \in E \tag{3}$$

$$\sum_{e \in E^+(s)} f_e^s = p_s \quad ,\forall s \in V_s \tag{4}$$

$$\sum_{g \in V_g} \sum_{e \in E^-(g)} f_e^s = p_s \quad ,\forall s \in V_s \tag{5}$$

$$\sum_{s \in V_s} \sum_{e \in E^+(i)} f_e^s = p_i + \sum_{s \in V_s} \sum_{e \in E^-(i)} f_e^s \quad ,\forall i \in V_s \tag{6}$$

$$\begin{aligned} \textstyle\sum_{e \in E^+(i)} a_e^t \quad \leq \quad & \textstyle\sum_{e \in E^-(i)} \sum_{tp \in \{1..t-1\}} a_e^{tp} \\ & + p_i - \textstyle\sum_{e \in E^+(i)} \sum_{tp \in \{1..t-1\}} a_e^{tp} \\ & ,\forall i \in V_s, t \leq T_{max} \end{aligned} \tag{7}$$

Constraint 1 binds the use of a time slot to at least the activity of one link in this slot. Constraint 2 ensures that two conflicting nodes do not transmit in the same time slot. Constraint 3 ensures the correspondence between the activities and the flows on links. Constraints 4, 5 and 6 express the conservation of messages. The last constraint guarantees that a packet is received or generated by a node before this node transmits it.

Based on this model, an optimal time slot assignment can be obtained by linear programming tools such as GLPK (GNU Linear programming kit) [11]. The optimal results obtained by the GLPK solver are presented in Section VI-A.

IV. Theoretical bounds on the number of slots

In this section, we focus on theoretical lower bounds of cycle length for three different topologies: linear, multi-linear and tree. The two first ones are special topologies of the third one and are representative of WSNs deployed in confined areas such as the airplane fuselage (linear) or mines with several galleries (multi-linear).

A. Additional assumptions

For simplicity reasons, we assume that in each time slot the transmitter sends only a single packet. Nodes are randomly deployed in the 2-dimensional plane. Two nodes u and v are 1-hop neighbors if and only if their distance is lower than or equal to the transmission range R. For any integer $h > 1$, any two nodes u and v are h-hop neighbors if and only if u is $(h-1)$-hop away from a 1-hop node of v.

In this paper, we consider that interferences are limited to 2 hops. Consequently, we assume that any two nodes u and v within 2-hop neighborhood from each other do not transmit in the same time slot. Indeed, let u and v two-hop neighbors and w their 1-hop neighbor. If u and v transmit at the same time, a collision will occur on w.

Based on this assumption, we can define the label of a node.

*Definition 3 (**Node label**):* For any network node u different from the sink, we define $label(u)$ = (distance(u,$sink$)-1) modulo 3.

In line networks, the assigned labels (starting by the sink child) are respectively 0,1,2,0,1,2,etc. It follows that only nodes having the same label can be assigned the same slot.

B. Number of slots in linear networks

Theorem 1: In linear networks, a lower bound on the number of slots is Max($N-1$, $3N-6$), where N is the number of nodes including the sink.

Proof: Consider a linear network with $N \geq 1$ nodes, where u_0 is the sink node and any node u_i is at a distance $i < N$ from the sink. It is clear that for $N = 1$, 2 or 3 the theorem is true. Now let us assume $N \geq 4$. Assuming each node needs one slot to transmit its own data, the sink needs $N-1$ slots to receive data from all nodes. Besides, let u_1, u_2 and u_3 the three closest nodes to the sink. No two nodes among these three nodes can transmit data simultaneously. Consequently, the number of slots occupied by these nodes is the sum of the number of packets they have to transmit, that is $(N-1) + (N-2) + (N-3)$. Hence the theorem. ■

C. Number of slots in multi-line networks

Theorem 2: In multi-line networks, a lower bound on the number of slots is Max($N-1$, $3n_k-3$), where N is the number of nodes including the sink, and n_k is the maximum number of nodes in a line starting with a child of the sink.

Proof: The sink requires $N-1$ time slots to receive all the packets generated in the network. Moreover, consider the longest line and let n_k be the number of nodes in this line starting with a child of the sink. From theorem 1, at least $3(n_k+1)-6 = 3n_k-3$ slots are required to transmit data to the sink. Hence the theorem. ■

D. Number of slots in tree networks

Theorem 3: In tree networks, a lower bound on the number of slots is Max($N-1$, $2n_k-1$, $3n_j-3$), where N is the number of nodes including the sink, n_k is the maximum number of nodes in a subtree rooted at a child of the sink and n_j is the maximum depth of nodes.

Proof: The sink requires $N-1$ time slots to receive all the packets generated in the network. Moreover, let us consider u_k the child of the sink with the highest number of descendants. Let n_k-1 be this number. At least n_k slots are needed by u_k to transmit its packets and at least n_k-1 slots are needed by u_k to receive the packets from its children. Since all these transmissions are sequential, at least $2n_k-1$ slots are needed. If now we consider the longest line in the network. Let n_j be the depth of the deepest node. According to theorem 1, at least $3n_j-3$ slots are needed. Hence, the number of slots is at least max($N-1$, $2n_k-1$, $3n_j-3$). ■

V. TRASA: TRaffic-Aware time Slot Assignment

The main objective of TRASA algorithm is to achieve a time minimal scheduling while ensuring a fair medium access where any node is granted a number of slots proportional to its packet demand.

A. Principles

TRASA is based on the following rules:

1) Any node has a *priority* and a set of conflicting nodes.
2) Nodes compete for the current time slot if and only if they have data to transmit.
3) For any slot, the first scheduled node is the node having the highest priority among all the nodes having data to transmit.
4) Any node can be scheduled in any time slot if it does not interfere with nodes already scheduled in this slot.

B. Algorithm presentation

In this section, we present a centralized version of TRASA given by the Algorithm 1. The algorithm iterates over N the set of nodes having data to transmit and sorted according to their priority. In each iteration, the algorithm determines the set of nodes scheduled in the current time slot starting at t, and the number of slots allocated to each of them. The node u with the highest priority is scheduled first (line 6). Further, any other node in the sorted set N is given the same time slot if and only if it does not conflict with nodes already scheduled in this slot (see the while loop of line 13). TRASA ends when all packets generated in the network are transmitted to the sink.
Two versions of TRASA are simulated. Indeed, at any iteration, when a node is scheduled, it is allocated either:
◦ only one time slot: this version is denoted *oneSlot*;
◦ as many time slots as required by the node with the highest priority: this version is denoted *manySlots*.
Intuitively, the *manySlots* version allows a node to transmit its packets successively, avoiding switching delays between the sleep and the awake states.

Concerning the definition of the priority, we evaluate TRASA for two heuristics:
◦ *prio=descNb*: The priority of any node is given by its number of descendants. Intuitively, a node with a high number of descendants will have a high number of packets to transmit.
◦ *prio=remPckt*parentDem*: $remPckt$ means the number of packets the node has in its buffer at the current iteration. $parentDem$ is the total number of packets the parent of the node has to forward in a cycle. The idea behind this heuristic is to reduce the number of buffered packets by favoring nodes having packets to transmit to a parent with a high number of descendants.

C. Properties: bounds on the number of slots

In this section, we present the theoretical properties of TRASA for the heuristic with $prio = descNb$, assuming that:
A6. Each source node generates exactly one packet per TDMA cycle.
A7. For any node u, the only nodes conflicting with u are its parent, its children, its grandparent, its brothers and its grandchildren.

1) TRASA for linear networks:

Property 1: Applied to a linear network of N nodes, TRASA schedules these nodes according to the following sequence of labels: (010)-(210)*. That is: (1) All nodes with label 0 are scheduled simultaneously, followed by all nodes with label 1, etc. (2) the sequence (210) is repeated a number of times equal to $N \ div \ 3$, where div is the integer division operator.

Proof: For space limitation we do not provide the proof. This property results from the fact that nodes are scheduled in the order of their number of descendants. ■

Theorem 4: TRASA ensures a **time optimal** scheduling using $Max(N-1, 3N-6)$ slots for any linear network of N nodes.

Proof: Consider a linear network of N nodes, where u_1, u_2, u_3 are the three closest nodes to the sink u_0. From property 1, each time slot is occupied by one of these nodes. Consequently, any other node u_i can be scheduled with one of these nodes depending on its label. It means that the number of slots in the TDMA cycle is equal to the number of slots required by u_1, u_2 and u_3 which is $(N-1)+(N-2)+(N-3) = 3N-6$. We have proved in Theorem 1, that any valid scheduling requires at least $Max(N-1, 3N-6)$ slots. As a consequence, TRASA that reaches this lower bound is optimal for linear networks. ■

2) TRASA for multi-line networks:

Property 2: For any multiline network, let u_i be the child of the sink with the highest number of descendants denoted n_i, TRASA requires at least $Max(N-1, 3n_i-3)$ slots if there is no other child of the sink with the same number of descendants n_i, and $Max(N-1, 3n_i-2)$ otherwise.

Proof: Let u_i be the child of the sink with the highest number of descendants denoted n_i. Let u_j be the child of the sink with the second highest number of descendants denoted n_j. Since nodes u_i and u_j are 'brothers' and u_i has a priority higher than u_j, u_i will occupy the first slot in parallel with any descendant of u_i having label 0, and any descendant of u_j having label 1. The second slot will be assigned to any descendant of u_i having label 1 and any descendant of u_j

Algorithm 1 TRASA algorithm.

```
1: Input: a spanning tree T, where each node u has d_u packets to
   transmit and a set of conflicting nodes Conflict(u).
2: Output: The scheduling of nodes in the TDMA cycle
3: t = 1 /* current time slot */
4: while sum_u d_u do
5:    N = List of nodes having data to transmit sorted according to
   their priority
6:    u = node with the highest priority in N
7:    if "oneSlot" then
8:       nbSlot = 1
9:    end if
10:   if "manySlots" then
11:      nbSlot = d_u
12:   end if
13:   while N ≠ ∅ do
14:      u=node with the highest priority in N
15:      nbAssignSlot = min(d_u, nbSlot)
16:      assign slots t to t + nbAssignSlot − 1 to node u
17:      d_u -= nbAssignSlot
18:      d_parent(u) += nbAssignSlot
19:      N = N \ ({u} ∪ Conflict(u))
20:   end while
21:   t+ = nbSlot
22: end while
```

with label 0, etc. It results that the TDMA cycle has the label sequence *(010)-(210)** for the descendants of u_i and the label sequence *(102)-(012)** for the descendants of u_j. Similarly to property 1, the sequence (012) relative to u_j is repeated $n_j \ div \ 3$ times. Consequently:

∘ If the two branches contain the same number of nodes, the node u_j will send its last packet after the node u_i and hence, one additional slot to $3n_j - 3$ is required.

∘ If $n_i > n_j$, all descendants of u_j will be able to share slots with those used by the descendants of u_i. ■

Theorem 5: Applied to a multi-line network, the number of slots n used by TRASA verifies: $n \geq Max(3n_k - 3, N - 1)$, where N is the total number of nodes including the sink, and n_k is the highest number of descendants of the sink children.

Proof: Assuming each node has only one packet to transmit, and hence requires one time slot, at least $N - 1$ slots are needed by the sink to receive data from these nodes. From theorem 4, if we consider the child of the sink with the highest number of descendants n_k, $3(n_k + 1) - 6$ slots are needed to schedule nodes on this branch. ■

Notice here that since the children of the sink cannot share the same slot, the cycle length is strictly higher than this bound in some scenarios as explained in property 2.

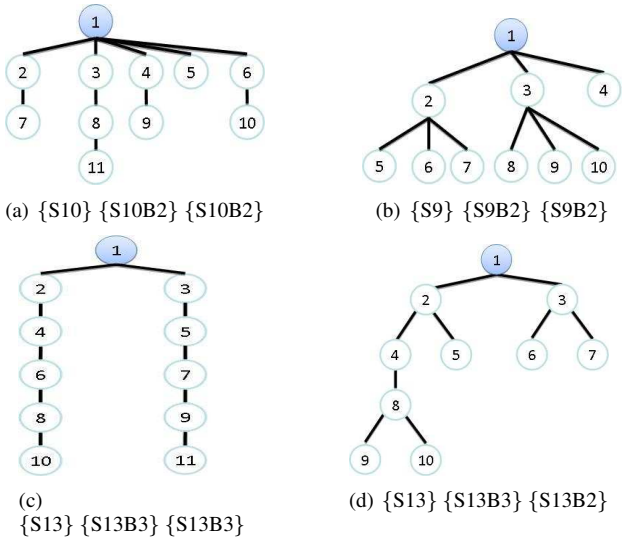
VI. Performance evaluation

A. Comparison with the optimal results

We used the GLPK solver to find the optimal time slot assignment, taking as inputs: (1) the model file corresponding to the problem formalization expressed in GLPK language, and (2) the data file describing the network topology, the packet demand of each node and the conflicting nodes of any network node. We obtained optimal results for various multi-line and tree topologies. For small problem sizes (few nodes, each source generates a single packet) results are obtained within an acceptable duration of time. Nevertheless, when the WSN becomes large and even for moderate network sizes (e.g. 30 nodes) and simple topologies (e.g multiline networks), the time required to compute the optimal solution is higher than one day.

Figure 1 illustrates the number of slots and the maximum size of buffers obtained by the model and TRASA with its two heuristics *prio=descNb* and *prio=remPckt*parentDem*. The caption of each subfigure follows the pattern $\{Sa\}\{SbBb\}\{ScBc\}$, where Sa stands for the optimal number of slots, Sb and Bb (Sc and Bc respectively) are the number of slots and the maximum number of buffers obtained by TRASA with $prio = descNb$ ($prio = remPckt * parentDem$ respectively). Notice that the TRASA algorithm provides the optimal number of slots in all these topologies.

Besides, for tree topologies, we have proved in theorem 3 that the lower bound on the number of slots is the maximum of three terms. For each of them, we can find a topology such that TRASA reaches this term. Hence, TRASA is optimal on these topologies. For example, with a branch factor of 3 and 20 nodes, we get $2n_k - 1$= 2*11-1=21 slots, which is optimal according to theorem 3. For 50 nodes and a branch factor of 3, we get 49=$N - 1$ slots, which is optimal according to theorem 3. For 10 nodes where a sink child is the head of a line with 4 nodes and the other sink child has 3 children, TRASA reaches the lower bound of $3n_j - 3$= 3*5-3=12 slots, that is the optimal.



(a) {S10} {S10B2} {S10B2}

(b) {S9} {S9B2} {S9B2}

(c) {S13} {S13B3} {S13B3}

(d) {S13} {S13B3} {S13B2}

Fig. 1. Examples of tree and multi-line topologies

B. Simulation results

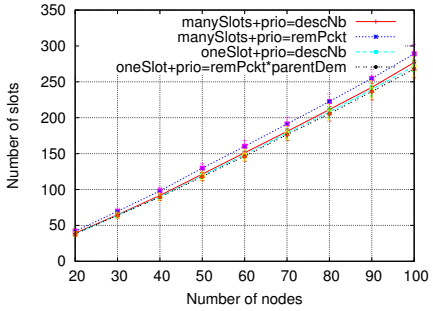
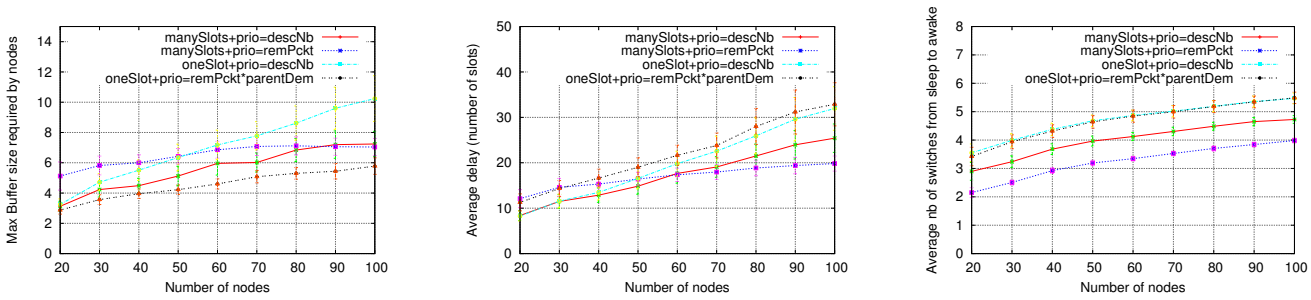


Fig. 2. The number of slots.

We developed a Java based simulation tool and performed simulations with the two versions and two heuristics of TRASA. We compare the TRASA performance with a slot assignment where the priority is given by the number of remaining packets, denoted $remPckt$ (i.e the number of packets present in the buffer of the node considered). We generate random graphs deployed in a given area ($100m$x$100m$), where the number of nodes ranges from 20 to 100. We build trees where the maximum number of children is 3. Unlike the previous section where the only existing links are those in the tree, we assume that a link exists between two nodes if and only if their distance is less than or equal to the radio range ($30m$). Consequently, additional links to the tree links are considered. In the following, each result is an average of 20 runs for small topologies, and 50 runs for large topologies.

We first evaluate the total number of slots for the two heuristics of TRASA (see Figure 2). Both heuristics of TRASA give the same number of slots, and outperform the heuristic $remPckt$. This result justifies the heuristic $remPckt *$



(a) (b) (c)

Fig. 3. TRASA performance regarding (a) Maximum buffer size (b) Average delay (c) Number of radio switches.

$parentDem$ that takes into account not only the number of remaining packets but also the parent demand of any node. Further, the slot number is not impacted by the number of slots assigned to any node in each iteration: $OneSlot$ and $manySlots$ versions of TRASA.

Simulation results show that $OneSlot + remPckt * parentDem$ ensures the smallest buffer size as illustrated in Figure 3(a) which is explained by the pipeline effect favored by this heuristic. The opposite of one might think, assigning the highest priority to a node having the highest number of remaining packets does not accelerate the buffer release, as illustrated by the result of $manySlots + remPckt$. We evaluate the average delay as the average number of slots that one packet takes to reach the sink once it is transmitted by its source. Figure 3(b) shows that TRASA achieves the smallest delays for the heuristic $manySlots + descNb$. This is explained by the fact that if the priority is given by the number of descendants, nodes close to the sink have the highest priority, and hence the probability that the sink receives data in a time slot is high, which reduces the data gathering delays. Based on the slot allocation, any node should be awake in its slots and the slots of its children assuming tree unicast communications, and can turn to the sleep state in the remaining time. Maximizing the sleep duration of a node is the key to allow it to save energy. Moreover, reducing the radio state switches contributes also in energy saving as these switches are energy consuming. Figure 3(c) shows that as expected the $manySlots$ version allows to reduce the number of radio state switches since nodes are allowed to send their packets in consecutive time slots.

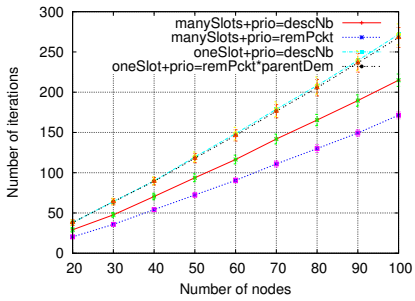


Fig. 4. The number of iterations.

Similarly, the number of iterations is reduced with the $manySlots$ version as illustrated in Figure 4.

VII. Conclusion

In this paper, we focus on data gathering applications which are the most frequent applications supported by WSNs. Assuming a slotted medium access, we investigate the raw convergecast problem looking for a minimal schedule length. A smaller schedule length improves the end-to-end delays and reduces the energy consumption. We focus more particularly on specific topologies such as linear or multi-line which are well adapted to confined environments and compute lower bounds for the time slot assignment problem. These bounds are compared with the optimal solution given by the GLPK solver. We present the TRASA algorithm and prove that it is optimal for linear topologies. Furthermore, applying TRASA to particular different tree topologies, we show that the optimal schedule length is reached. TRASA heuristics outperform solutions only based on the number of remaining packets.

References

[1] D.O. Incel, A. Ghosh, B. Krishnamachari, K. Chintalapudi, *Fast data collection in treebased wireless sensor networks.* IEEE Transactions on Mobile Computing, vol. 1, pp. 86-99, Jan 2012.

[2] R. Huang, W.Z. Song, M. Xu, B. Shiraz, *Localized QoS-Aware Media Access Control in High-Fidelity Data Center Sensing Networks*, The 1st International Green Computing Conference, IGCC'10, Chicago, USA, August 2010.

[3] W. Z. Song, R. Huang, B. Shirazi, and R. LaHusen, *TreeMAC: Localized TDMA MAC Protocol for Real-time High-data-rate Sensor Networks*, Journal of Pervasive and Mobile Computing, Percom 09, vol. 5, pp. 750-765, Dec 2009.

[4] V. Turau, C. Weyer, C. Renner, *Efficient Slot Assignment for the Many-to-One Routing Pattern in Sensor Networks*, First International Workshop on Sensor Network Engineering (IWSNE'08), Santorini Island, Greece, June 2008.

[5] S.C. Ergen, P. Varaiya, *TDMA scheduling algorithms for wireless sensor networks*, Wireless Networks, vol. 16, pp. 985-997, May 2010.

[6] S. Gandham, Y. Zhang, Q. Huang, *Distributed time-optimal scheduling for convergecast in wireless sensor networks*, Computer Networks, vol. 52, pp 610-629, 2008.

[7] H. Zhang, F. Osterlind, P. Soldati, T. Voigt, M. Johansson, *Time-optimal convergecast with separated packet copying: scheduling policies and performance*, Technical Report, KTH Electrical Engineering Royal Institute of Technology, Sweden, May 2009.

[8] G.-S. Ahn, E. Miluzzo, A. T. Campbell, S. G. Hong, and F. Cuomo, *Funneling-mac: A localized, sink-oriented mac for boosting fidelity in sensor networks,* 4th ACM Conference on Embedded Networked Sensor Systems (SenSys 2006), Boulder, CO, USA, November 2006.

[9] J. Mao, Z. Wu, X. Wu, *A TDMA scheduling scheme for many-to-one communications in wireless sensor networks.* Computer Communications. vol. 30, pp.863-872, Feb 2007.

[10] H. Choi, J. Wang, E. Hughes, *Scheduling for information gathering on sensor network*, Wireless Networks, vol. 15, pp. 127-170, Jan 2009.

[11] http://www.gnu.org/software/glpk/