TRASA: TRaffic Aware Slot Assignment Algorithm in Wireless Sensor Networks

Ichrak Amdouni, Pascale Minet

TRASA: TRaffic Aware Slot Assignment Algorithm in Wireless Sensor Networks

Ichrak Amdouni and Pascale Minet
INRIA Rocquencourt
78153 Le Chesnay cedex, France
Email: ichrak.amdouni@inria.fr, pascale.minet@inria.fr

***Abstract*—In data gathering applications which is a typical application paradigm in wireless sensor networks, sensor nodes may have different traffic demands. Assigning equal channel access to each node may lead to congestion, inefficient use of the bandwidth and decrease of the application performance. In this paper, we prove that the time slot assignment problem is NP-complete when $p-hop$ nodes are not assigned the same slot, with $1 \leq p \leq h$ for any strictly positive integer h. We propose *TRASA*, a TRaffic Aware time Slot Assignment algorithm able to allocate slots to sensors proportionally to their demand. We evaluate the performance of *TRASA* for different heuristics and prove that it provides an optimized spatial reuse and a minimized cycle length.**

I. Context and motivations

A typical application in Wireless Sensor Networks (WSNs) is data gathering. Sensor nodes are deployed in the region of interest to periodically collect and report sensed data to a sink node. To achieve this many-to-one communication, sensor nodes form a data gathering tree rooted at the sink. This communication paradigm is also known as "convergecast" [1].

Depending on the application requirement, different objectives have to be taken into account in the design of MAC protocols. For instance, reliability is required mainly in critical applications like fire detection. In this disaster scenario, reducing the end-to-end delays is also a major requirement. Energy-efficiency is an important concern specially when sensors are operating for a long period of time. It has been proved that, under heavy traffic conditions, contention-free medium access based on TDMA outperforms contention-based protocols.

The main task of TDMA-based MAC protocols is to assign time slots to sensor nodes and to schedule the medium access based on these slots. The transmission schedule allows nodes to send and receive data packets without collision. Further, any node can enter into sleep mode during inactive periods, thus achieving low duty cycle and saving energy. Although the use of TDMA requires synchronization between cooperating sensors, it is an efficient way of mitigating the limitations of CSMA based networks. However, the performance of the TDMA-based medium access protocols may dramatically decrease if the TDMA parameters are not aware of the application requirements [2]; for example, a very long frame may increase the latency. Moreover, traditional MAC protocols tend to give nodes equal channel access, while the sensor traffic demands may differ. Indeed, nodes close to the sink forward more data than leaf nodes in the data gathering tree. This is the "funneling effect" [3]. Allocating equal numbers of time slots to sensor nodes may lead to congestion, packet loss, and inefficient use of the bandwidth. Consequently, channel access should be proportional to the sensor demand. That is why we investigate in this paper the *Time Slot Assignment* problem, denoted *TSA*, regarding the application requirements. We propose *TRASA*, *TRaffic Aware Slot Assignment algorithm* for WSNs. Assuming a sensor network where each node has a specific number of packets to transmit to its parent in the data gathering tree, *TRASA* assigns each node a number of slots proportional to its traffic demand, and schedules its activity. Moreover, *TRASA* allows the allocation of slots to nodes with heterogeneous demands. Consequently, the algorithm addresses the funneling effect and ensures a fair medium access. *TRASA* builds a schedule and ensures that data reach the sink in one cycle. *TRASA* takes into account both tree communication links and other interfering links. Via simulations, we evaluate the impact of interfering links on *TRASA* algorithm, and show that as the spatial reuse decreases, the schedule length increases. Since the time slot assignment problem is NP-complete as we will prove hereafter, *TRASA* relies on an heuristic determining the priority of each node. We compare this heuristic with another heuristic and justify that *TRASA* ensures an optimized schedule length and a good tradeoff between the schedule length, the average end-to-end delay and the maximum buffer size required by each node.

The remainder of this paper is organized as follows. In section II, we provide a state of the art about the *TSA* problem. In section III we describe the assumptions adopted and define the problem statement. In section IV, we prove that the *TSA* problem is NP-complete, assuming that two nodes that are h-hop away, with h a positive integer > 1 interfere. Section V describes the *TRASA* algorithm and its properties. In section VI, we present the performance evaluation of *TRASA* and conclude in section VII.

II. State of the art

Despite the existence of a variety of scheduling schemes, few of them allocate a number of slots proportional to node demand. In this section, we present a state of the art of existing schemes classified according to their awareness of the traffic demand. This classification is illustrated in Figure 1.

◦ **Protocols that do not depend on the traffic demand, or assume that nodes aggregate all the data they have to**

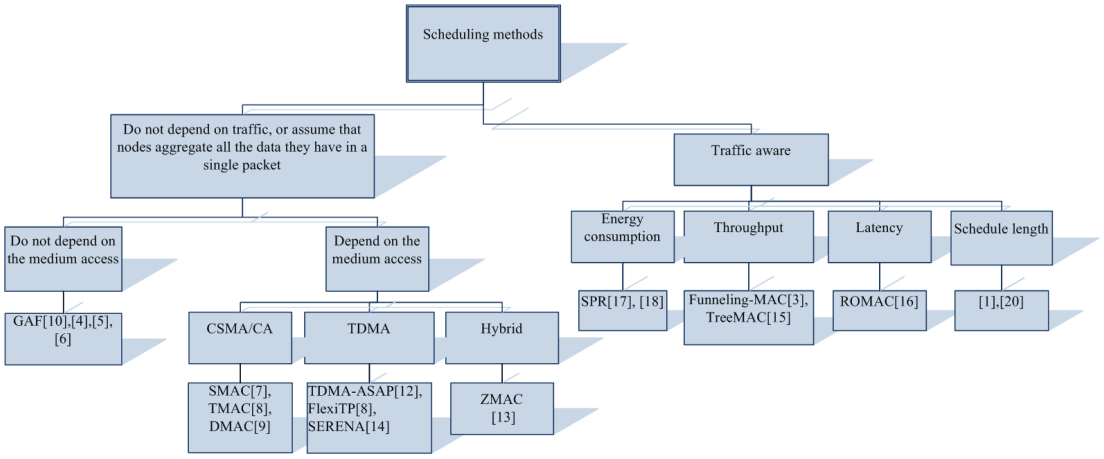


Fig. 1. Taxonomy of time slot assignment techniques.

transmit in a single time slot:
With these protocols, any node receives exactly one slot. For space reasons, we do not detail here this class of algorithms.

○ **Traffic aware protocols:**
In these protocols, the number of slots received by a node depends on its traffic demand. We present some examples of *T*SA algorithms classified according to their main objective. Notice that one algorithm may satisfy more than one objective.

1) Maximizing throughput of received data at the sink:
In [3], Gahng et al. presented Funneling-MAC to mitigate the funneling effect and boost application fidelity in WSNs. Funneling-MAC is hybrid, TDMA is used within small number of hops from the sink (called the intensity region), and CSMA/CA elsewhere. The sink is responsible for the scheduling of the nodes in this intensity region. Funneling-MAC allows the slot reuse between nodes distant more than 2 hops. The main goal of TreeMAC [15] is to achieve high throughput in real-time high-data-rate WSNs. Authors proved they achieve at least one-third of the optimum throughput assuming reliable links. In TreeMAC, slots are assigned from the root to the leaves, making it not suitable for large-scale networks. TreeMAC allows a slot reuse between nodes belonging to the same tree branch, but not between the subtrees. From energy point of view, authors show that TreeMAC outperforms CSMA and Funneling-MAC [3]. However, nodes in Tree-MAC are assigned disjoint transmitting time slots, which means that if the node can sleep between its transmitting slots, it should awake up many times to transmit and receive its data. The transitions between radio states are energy-consuming, so should be reduced.

2) Minimizing latency:
ROMAC [16] is a localized QoS-aware for high-fidelity data center sensing networks. Like TreeMAC, ROMAC divides the TDMA cycle into frames, each frame being composed of three slots and allocates slots to nodes proportionally to their demand. Compared to TreeMAC, ROMAC enhances the localization aspect, since each node can locally determine its frame and its time slots without relying on its parent like in TreeMAC. The node with identifier i transmits its own data in the frame i, and transmits the packets of any node j in its subtree in the j^{th} frame. Unlike TreeMAC and Funneling-MAC, ROMAC automatically adapts to routes changes, any node updates its set of frames every TDMA cycle. Consequently, ROMAC achieves lower delivery latency than these two protocols as the network size increases.

3) Minimizing schedule length:
Incel et al. [1] aimed at reducing the delays of data collection by minimizing the schedule length. They studied scheduling nodes where each node generates a packet at the beginning of the TDMA frame, in convergecast scenario keeping tree interference links only. They proved that the lower bound on the schedule length is given by: (1) the maximum node degree when packet aggregation at each intermediate node is considered, and (2) $max(2n_k - 1, N)$ where n_k is the maximum number of descendants of the sink children, otherwise. For this second case, authors proposed *Local-TimeSlotAssigment* algorithm in which the sink schedules an edge having the highest remaining number of packets, and any parent node with an empty buffer selects one child whose buffer is not empty at random respecting tree interfering links. This way, they ensure parallel transmissions among multiple branches of the tree, and keep the sink busy in receiving packets. In [20], authors proposed algorithms based on coloring. Two centralized solutions are described. First, in the node-based scheduling any slot is shared between nodes with the same color and any other node that does not conflict with them. Second, in the level-based scheduling, conflicts are defined between levels: the same color is assigned to two levels if they do not contain any couple of conflicting nodes. For each color, non conflicting nodes from all the levels having this color share the same slot. Besides other nodes that do not conflict with already scheduled nodes are scheduled simultaneously. This algorithm suffers from the energy waste because of the radio switches between active/sleep states.

4) Minimizing the energy consumption:
In [17], Turau et al. proposed SPR, to schedule each path in the routing tree separately. The slots assigned to any node are the union of its slots in each path. Hence, the spatial reuse of time slots is restricted to a common path, which makes SPR not efficient in terms of schedule length for networks where

the average number of children is high. Another example in this category is given in [18] where the energy efficiency is achieved by reducing the number of switches between the active and sleep states.

III. Time slot assignment problem

In this section we present the assumptions we adopt to study the time slot assignment problem *TSA*.

A. Assumptions and system model

∘ **Network model:** The network is modeled as a graph $G = (V, E)$, V is the set of vertices, and E is the set of edges representing the communication links. T is a spanning tree of G rooted at the sink node. We adopt the unit disk model, where nodes are modeled as a set of points in the 2-dimensional plane. Besides, there is no message losses. A node u receives a message sent by another node v, if and only if, their distance is lower than a given uniform transmission range R. Nodes u and v are then 1-hop neighbors. 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.
∘ **Interfering links:** Usually, in a data gathering application, the communications of a node are limited to its parent and its children. However, any node may have communication links with other nodes. If these links are not taken into account in node scheduling, collisions may occur. Hence, for a given node, we include all its interfering links, those in the tree and other ones. In the literature, solutions to the *TSA* problem for data gathering applications tend to limit the interferences to 2 hops. However, in real wireless networks, interferences may exceed this distance. Consequently, to generalize our study, we assume that two nodes that are p-hop away with $1 \leq p \leq h$ are interfering. h being a given positive integer > 1 which is a parameter of our algorithm. Hence, we denote the studied problem h_TSA.
∘ **Application data:** We consider a data gathering application. In each TDMA cycle, each node except the root has its own data to transmit 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.

B. Problem statement

We study the slot assignment problem under the assumptions introduced in Section III-A. The slot assignment problem consists in assigning slots to nodes in G, such that no two nodes that are p-hop away with $1 \leq p \leq h$ are scheduled in the same slot while minimizing the schedule length. Besides, this scheduling must ensure that each node transmits its own packets, and the packets generated in its subtree.

C. Lower and upper bounds

Theorem 1: The number of slots required by the TSA problem meets:

$$\text{Number of nodes} - 1 \leq \text{Number of slots} \leq \sum_{\text{node u}} \text{depth(u)}, \quad (1)$$

where depth(u) is the depth of node u in the tree.

Proof: In the best case, all the sink children transmit their packets in sequence. Indeed, there is no possible spatial reuse between sink children. Hence the lower bound. In the worst case, each slot corresponds to a single packet transmission. The number of slots needed by any packet generated by any node u at depth d in the data gathering tree is equal to d. Hence the upper bound. ■

IV. Complexity of the h_TSA problem

It has been proved in [19] and [20] that the *TSA* problem is NP-complete. In this paper, we generalize the study and prove that the h_TSA problem is NP-complete for any positive integer h where any two nodes that are less than or equal to h-hop away are not scheduled simultaneously.

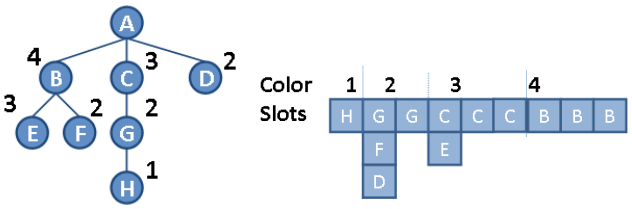
Theorem 2: The h_TSA problem, for any positive integer $h \geq 1$ is NP-complete.

Proof: The decision problem of *h_TSA* is given by: *Can nodes in a graph G be assigned S time slots (S is a positive integer) during which they can transmit their data to a sink node, such that any two nodes that are p-hop away, with* $1 \leq p \leq h$*, are not assigned the same time slot?*
To prove that the decision problem of *h_TSA* is NP-complete, we use the knowledge that the *h_Color* problem is NP-complete [21]. In [21], the *h_Color* problem is defined as *coloring a graph with the smallest number of colors (a color is represented by an integer) such that any two nodes that are p-hop away with* $1 \leq p \leq h$*, do not have the same color. In addition, the color assigned to any node is smaller than the color assigned to its parent in the data gathering tree.*
We need to prove that finding a solution to *h_TSA* is equivalent to finding a solution to *h_Color*.

Let G be a connected, undirected graph, and T its spanning tree. Notice that the construction of T can be done in polynomial time. Each node u in G has traffic demand d_u. Let C_G be a coloring of G solving the *h_Color* problem and using the colors $c_1, c_2, \ldots, c_{max}$. Let N_i be the set of nodes having the color c_i. We can build a slot assignment for nodes in G as follows. We sort the colors by increasing order. For the smallest color c_i not yet considered, we add to the cycle a number of slots equal to the maximum traffic demand of nodes in N_i, denoted max_{di}. Then, nodes in N_i are scheduled during these slots, each one has a number of slots equal to its traffic demand. Consequently:

1) This scheduling is conflict-free, nodes that are scheduled simultaneously have the same color, and hence do not interfere.
2) The scheduling allows each node to transmit all the data it has since it is assigned a contiguous number of slots equal to its demand. Besides, this scheduling ensures that each parent node accesses the medium after all its children, because it has a higher color than them, and slots are assigned to nodes having the smallest colors first.
3) The number of slots used is equal to $S = \sum_{i=1}^{c_{max}} max_{di}$.

Figure 2 illustrates the proof.



(a) (b)

Fig. 2. An example illustrating the proof of the NP-completeness of *T*SA problem.

Inversely, given a scheduling of G, composed of S slots, we need to color the graph according to the *h_Color* problem. We denote $S_s = s_1, s_2, \ldots, s_s$ the sequence of slots such that during each slot s_i at least one node is scheduled for the last time in the TDMA cycle. Then, for each slot s_i in S_s we assign the color i to all uncolored nodes occupying s_i such that s_i is their last time slot. Consequently, we get a number of colors bounded by the cardinal of S_s. The same color is assigned to nodes scheduled in the same slot, so the coloring is collision-free. Further, since the last slot of each parent can not be scheduled before the last slot of any of its children, each parent has a color greater than the color of its children. Hence the theorem. ■

V. TRASA Algorithm: Design and Properties

A. Overview and algorithm

The requirements of data gathering applications are various: a short TDMA cycle, a high throughput at the sink, small delays and low energy consumption. *TRASA* addresses these issues by maximizing the slot reuse in a cycle. Moreover, by favoring nodes with a high number of descendants, *TRASA* tends to minimize the TDMA cycle length and then saves energy. Figure 3 shows the slots computed by *TRASA* applied to the graph in Figure 2(a).

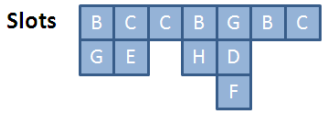


Fig. 3. An example of slot assignment by TRASA.

As the h_TSA problem is NP-complete, heuristics must be used. In *TRASA*, sensor nodes are sorted according to their priority. The priority of a node is given by the number of its descendants in the data gathering tree. Which means that the nodes close to the sink have the highest priority. The reasons for this choice are threefold: (1) Nodes with many descendants need to forward more packets than others, so scheduling them as soon as they have data avoids the congestion, and allows them to use less buffers. (2) The children of the sink are interfering, only one child can be active in a slot. They are the most determining factors of the schedule length (their number represents the lower bound of the number of slots). We believe that scheduling them first achieves a higher flexibility. (3) The probability of the sink to be in receive state each time slot is high, and similarly the throughput measured at the sink increases.

Algorithm 1 TRASA algorithm.

```
1: Input: a connected graph G, and its spanning tree T, where each node u has d_u packets to transmit
2: Output: Cycle: the TDMA frame with the assigned slots per node
3: Cycle.endingSlot = 0 /*the last slot assigned during the current iteration*/
4: Cycle.lastEndingSlot = 0 /*the first slot assigned during the current iteration*/
5: while (There is at least one node having data to transmit) do
6:    N_s = List of nodes having data to transmit sorted according to their priority
7:    u = node with the highest priority in N_s
8:    Cycle.addSlot(d_u) /*d_u slots are added to Cycle*/
9:    Cycle.lastEndingSlot = Cycle.endingSlot
10:   Cycle.endingSlot += d_u
11:   u.schedule.add((Cycle.lastEndingSlot,d_u))
12:   u.demand -= d_u
13:   u.parent.demand += d_u
14:   for node v in N_s do
15:      if (v ≠ u) && (v has d_v ≠ 0 packets) && (v does not interfere with nodes occupying the slots between lastEndingSlot and endingSlot) then
16:         d=Cycle.endingSlot-Cycle.lastEndingSlot
17:         if (d_v > d) then
18:            Cycle.add(d_v − d)
19:            Cycle.endingSlot+=d_v − d
20:         end if
21:         v.demand -= d_v
22:         v.parent.demand += d_v
23:      end if
24:      v.schedule.add((Cycle.lastEndingSlot, d_v))
25:   end for
26: end while
```

The pseudo-code of *TRASA* is given by algorithm 1.

Once the nodes are sorted, the node u with the highest priority is given a number of slots equal to its packet demand. This is because we assume for simplicity reasons that a time slot contains only one packet. Notice that *TRASA* is able to take into account sensor nodes with heteregeneous data rates. *TRASA* assigns node u d_u slots between indices $lastEndingSlot$ and $endingSlot$. Aiming at reducing the schedule length, *TRASA* achieves an optimized slot reuse. That is, a new time slot is added to the TDMA frame, if and only if, nodes having remaining data, could not be scheduled in the last time slot. The only limiting factor of the parallelism is the interferences which have to be considered to achieve a collision-free schedule. So to schedule other nodes simultaneously with the node u, *TRASA* iterates on the nodes having data to transmit according to the descending order of their priority. If a node v has packet(s) to send and is not interfering

with nodes already scheduled between $lastEndingSlot$ and $endingSlot$, v is scheduled in parallel with these nodes. The number of slots assigned to v is equal to its demand d_v. Notice that d_v may be higher than $endingSlot - lastEndingSlot$, in this case, the TDMA cycle is extended. *TRASA* stops when all nodes have transmitted all data they generated, and all data they received from their children.

B. TRASA properties

In this section, we discuss the properties of *TRASA*, and explain how it takes into account various performance criteria.

P1. Fair Access: The first key design of *TRASA* is to ensure a fair medium access to all nodes. A time slot being associated to a unique packet, a node has as many slots as packets to transmit. In a data gathering tree, any node traffic demand is the sum of the packets it generates and the packets generated in its subtree. *TRASA* assigns each node as many slots as its traffic demand. Hence, *TRASA* addresses the funneling effect and avoids congestions.

P2. Optimized spatial reuse: *TRASA* assigns slots to nodes in the order of their priority. When any node is scheduled, all non interfering nodes are scheduled simultaneously. Hence, the slot reuse is optimized, it is not restricted to a common branch, like [15], or inter-branches, like [17]. This spatial reuse provides an efficient use of the bandwidth.

P3. Optimized schedule length: In *TRASA*, the TDMA frame is extended by a time slot if and only if the nodes with a packet demand cannot be scheduled in the last added slot, because of the interferences. This behavior reduces considerably the schedule length. For instance, applied to the graph of Figure 2(a) for $h = 2$, *TRASA* gives a cycle length of 7 slots as illustrated by Figure 3. The cycle length would be 12 slots with TreeMAC.

P4. Optimized energy consumption: Minimizing the number of slots in a cycle reduces the activity periods and allows nodes to save energy. Besides, when any node is scheduled, it is allowed to send all data it has, which reduces the switches between the active and sleep states which are energy-consuming. With *TRASA*, any node is awake during its slots to transmit data and the slots of its children to receive data they transmit, it sleeps the remaining time to save energy.

P5. Optimized buffer usage: In WSNs, nodes have low capacity of storage. Accumulating many packets in a node may lead to buffer overflow and packet losses. In *TRASA*, the buffer usage is balanced during the TDMA frame. Indeed, only nodes having packets to transmit compete for a given time slot. Since the priority used is the number of descendants, the priority of any node is higher than the priority of all its children. Consequently a parent having data to transmit is in general scheduled before its children, so does not accumulate much data.

P6. Optimized delays and throughput: The optimized slot reuse of *TRASA* can be seen as a parallel progress of packets toward the sink, and in a balanced way between tree levels and subtrees, as much as allowed by the interference constraints. Consequently, the end-to-end delays are reduced. Furthermore, nodes close to the sink have the highest priority. This increases the probability that the sink receives data in a time slot, and hence enhances the data throughput. By construction of the cycle, *TRASA* guarantees that all data generated by nodes reach the sink in one cycle.

VI. Performance evaluation

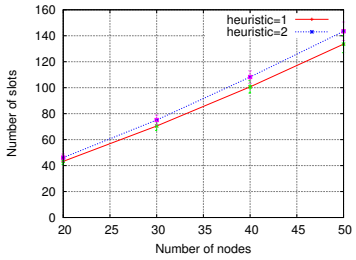
A. Simulation setup

Using a graph generator, we generate random graphs for a given number of nodes deployed in a 2D area ($1m$x$1m$), the transmission range is set to $0.4m$. The number of nodes ranges from 20 to 100. For each graph, we build a spanning tree defined by a maximum number of children per node. Notice that if the graph is disconnected, the construction of the tree is not possible. The corresponding graph is then not considered in the simulation process. The tree is built by adding children nodes consecutively. The first added node is the root. Any neighbor u of the root selects it as a parent if the maximum number of children (which is set to 3) is not reached by the sink. Otherwise, such a node u tries to associate to another node. This process continues until all nodes have a parent. Notice that implicitly, each node selects the parent with the minimum number of hops toward the sink, provided that this parent can accept further children. We assume that interferences are limited to 2-hops ($h = 2$), as usually assumed in *T*SA problems. Each node generates only one packet, and forwards packets it receives from its children. For simplicity reasons, we also assume that a slot contains only one data packet. Each result is the average of 40 simulation runs.

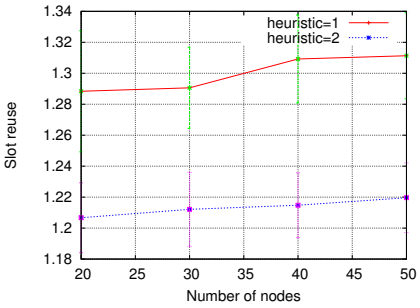
B. Simulation results

1) Choice of the heuristic: In *TRASA*, nodes are sorted according to their priority which is given by their number of descendants. This heuristic is denoted $heuristic = 1$. To validate this design choice, we compare this heuristic with another one, denoted $heuristic = 2$, where nodes with the highest number of descendants have the smallest priority. Figure 4(a) shows that $heuristic = 1$ outperforms $heuristic = 2$ in terms of slot number. This is because the slot reuse is increased, as illustrated in Figure 4(b). Figure 4(d) shows that the maximum size of buffers required by a node in $heuristic = 1$ is lower than in $heuristic = 2$. Indeed, when leaf nodes are scheduled first, a node is likely to accumulate packets before being able to transmit them. Whereas $heuristic = 1$ gives opportunity to nodes to send data as soon as possible, which considerably reduces average end-to-end delays as illustrated in Figure 4(c). However, nodes may access the medium more frequently during one TDMA cycle, which adds an energy cost when the number of switches between active and sleep state increases. For these reasons, $heuristic = 1$ is preferable and is selected in the following.

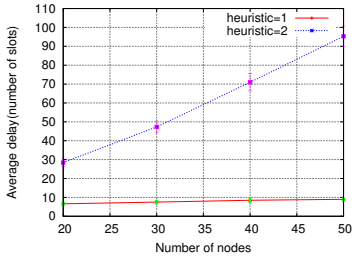
2) Impact of interference links: To evaluate the impact of the interference links on *TRASA* performance, we simulate two cases: (1) when all interfering links are considered; (2) when only tree links are considered. As expected, the schedule length in the first case is higher. For instance the



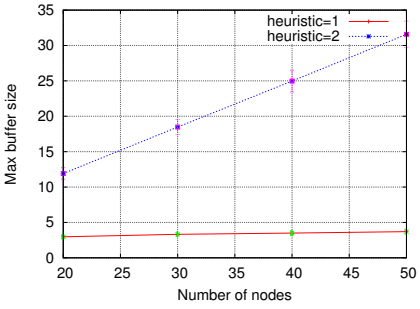
(a)



(b)



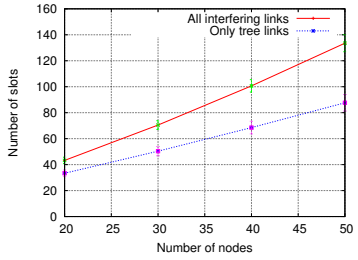
(c)



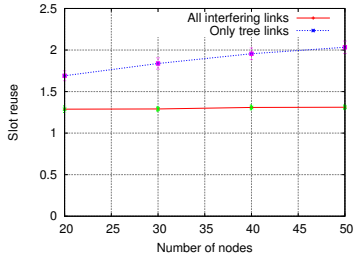
(d)

Fig. 4. Impact of the heuristic choice on: (a) Slot number; (b) Slot reuse; (c) Average delay, (d) Maximum buffer size.

number of slots raises from 88 to 135 for 50 nodes. The difference increases with the number of nodes as illustrated in Figure 5(a). For the same density, any node has more interfering links, so less nodes can be scheduled in the same slot. This justifies the fact that the slot reuse is higher in the second case, as illustrated in Figure 5(b). So, to avoid interferences and achieve collision-free schedule in any real wireless environment, the cost is the schedule length.

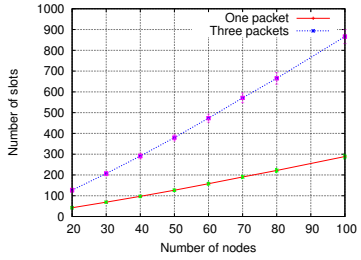


(a)

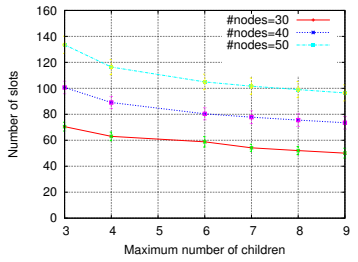


(b)

Fig. 5. Impact of the interference links on: (a) Number of slots; (b) Slot reuse.



(a)



(b)

Fig. 6. Impact of (a) Data rate (b) Maximum number of children on the number of slots.

3) Impact of the number of nodes, the node data rate and the maximum number of children: For the results illustrated in Figure 6, the density of nodes (average number of neighbors per node) varies between 4 and 20. The Figure 6(a) shows that the number of slots increases with the number of nodes.

This is because the density increases with the number of nodes, which reduces the spatial reuse of the slots. However, as the maximum number of children per node increases, the number of slots decreases (see Figure 6(b)). Indeed, when the maximum number of children increases, the tree depth decreases. Consequently, the number of slots required by each node which is proportional to its subtree size decreases. *TRASA* is able to schedule nodes with heterogeneous data rates. Figure 6(a) shows that the schedule length increases linearly with the data rate.

VII. Conclusion

In this paper we proved that the time slot assignment problem is NP-complete for any integer $h > 1$. We presented *TRASA* an algorithm that takes into account the application requirements to schedule nodes using an optimized number of slots. We have shown its excellent performance on representative scenarios by simulation. We plan to extend this work by defining a distributed version of *TRASA*.

References

[1] Incel, D.O., Ghosh, A., Krishnamachari, B., Chintalapudi, K.: *Fast data collection in treebased wireless sensor networks.* IEEE Transactions on Mobile Computing, 2009

[2] Gollan, N.; Schmitt, J., *Optimizing TDMA Design for RealTime Applications in Wireless Sensor Networks*, In Proceedings of 6. GI/ITG KuVS Fachgesprch "Sensornetze", Aachen, Germany, p87p90. July 2007.

[3] 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,* in Proc. 4th ACM Conference on Embedded Networked Sensor Systems (SenSys 2006), Boulder, CO, USA, Nov. 2006.

[4] M. Cardei, D. Du, *Improving wireless sensor network lifetime through power aware organization*, ACM Journal of Wireless Networks, May 2005.

[5] M. Cardei, M. Thai, Y. Li, W. Wu, *Energy-efficient target coverage in wireless sensor networks*, IEEE INFOCOM 2005, Miami, Florida, March 2005.

[6] J. Carle, D. Simplot-Ryl, *Energy-Efficient Area Monitoring for Sensor Networks*, Computer, vol. 37, no. 2, pp. 40-46, Febuary, 2004.

[7] W. Ye, J. Heidmann, D. Estrin, *An Energy-Efficient MAC Protocol for Wireless Sensor Networks*, IEEE INFOCOM, New York, USA, June 2002.

[8] T. V. Dam, K. Langendoen, *An adaptive energy-efficient MAC protocol for wireless sensor networks*, ACM SenSys'03, November 2003.

[9] G. Lu, B. Krishnamachari, C. Raghavendra, *An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks*, Parallel and Distributed Processing Symposium, April 2004.

[10] Y. Xu, J. Heidemann,D. Estrin, *Geography-informed Energy Conservation for Ad Hoc Routing*, In MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking Rome, Italy, July 2001

[11] L. L. Winnie, D. Amitava, C.O Rachel, *FlexiTP: A Flexible Schedule-Based TDMA Protocol for Fault-Tolerant and Energy-Efficient Wireless Sensor Networks* IEEE Transactions on Parallel and Distributed Systems, 19(6), 851-864, June 2008

[12] S. Gobriel, D. Mosse, R. Cleric, *TDMA-ASAP: senssor network TDMA scheduling with adaptive slot stealing and parallelism*, ICDCS 2009, Montreal, Canada, June 2009.

[13] I. Rhee, A. Warrier, M. Aia, J. Min, *Z-MAC: a hybrid MAC for wireless sensor networks*, SenSys'05, San Diego, California, November 2005.

[14] P. Minet, S. Mahfoudh, *SERENA: SchEduling RoutEr Nodes Activity in wireless ad hoc and sensor networks* IWCMC 2008, IEEE International Wireless Communications and Mobile Computing Conference, Crete Island, Greece, August 2008.

[15] 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, No. 6. (2009), pp. 750-765.

[16] Huang R.; Song W.Z.; Xu M.; Shirazi B., *Localized QoS-Aware Media A ccess Control in High-Fidelity Data Center Sensing Networks*, The 1st International Green Computing Conference, 2010

[17] Turau .; Weyer C.; Renner C. *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.

[18] Mao J., Wu Z., Wu. X, 2007. *A TDMA scheduling scheme for many-to-one communications in wireless sensor networks.* Computer Communications. 30, 4, February 2007.

[19] Choi H.; Wang J.; Hughes E. *Scheduling for information gathering on sensor network*, Wireless Networks, 2009

[20] Ergen S.C.; Varaiya P. *TDMA scheduling algorithms for wireless sensor networks*, Wireless Networks, May 2010

[21] Amdouni, I.; Minet, P.; Adjih, C., *Node coloring in wireless networks: complexity results and grid coloring*, WMNC 2011, Toulouse, France, October 26-28, 2011