J. Sens. Actuator Netw. **2013**, *2*, 449-485; doi:10.3390/jsan2030449

OPEN ACCESS

Journal of Sensor and Actuator Networks

ISSN 2224-2708

www.mdpi.com/journal/jsan

Article

An Adaptive Strategy for an Optimized Collision-Free Slot Assignment in Multichannel Wireless Sensor Networks

Ridha Soua *, Erwan Livolant and Pascale Minet

INRIA Rocquencourt, 78153 Le Chesnay cedex, France; E-Mails: erwan.livolant@inria.fr (E.L.); pascale.minet@inria.fr (P.M.)

* Author to whom correspondence should be addressed; E-Mail: ridha.soua@inria.fr; Tel.: +33-1-3963-5787; Fax: +33-1-3963-5566.

Received: 5 June 2013; in revised form: 21 June 2013 / Accepted: 27 June 2013 / Published: 16 July 2013

Abstract: Convergecast is the transmission paradigm used by data gathering applications in wireless sensor networks (WSNs). For efficiency reasons, a collision-free slotted medium access is typically used: time slots are assigned to non-conflicting transmitters. Furthermore, in any slot, only the transmitters and the corresponding receivers are awake, the other nodes sleeping in order to save energy. Since a multichannel network increases the throughput available to the application and reduces interference, multichannel slot assignment is an emerging research domain in WSNs. First, we focus on a multichannel time slot assignment that minimizes the data gathering delays. We compute the optimal time needed for a raw data convergecast in various multichannel topologies. Then, we focus on how to adapt such an assignment to dynamic demands of transmissions (e.g., alarms, temporary additional application needs and retransmissions). We formalize the problem using linear programming, and we propose an incremental technique that operates on an optimized primary schedule to provide bonus slots to meet new transmission needs. We propose AMSA, an Adaptive Multichannel Slot Assignment algorithm, which takes advantage of bandwidth spatial reuse, and we evaluate its performances in terms of the number of slots required, slot reuse, throughput and the number of radio state switches.

Keywords: multichannel wireless sensor networks; convergecast; time slot assignment; retransmissions; dynamic demands; optimized schedule; adaptivity

---

1. Introduction

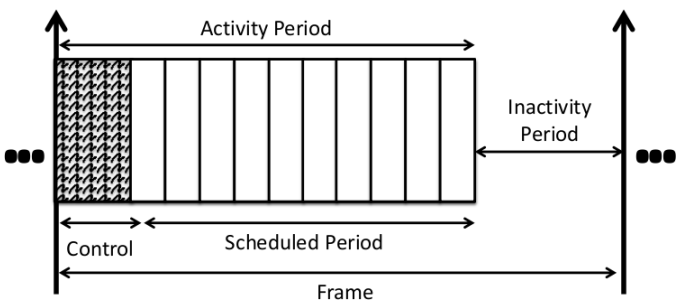
Data gathering or convergecast applications represent the majority of applications supported by wireless sensor networks (WSNs). Data from sensors are collected over a tree rooted at a special entity, called the sink, which is generally more powerful than the other nodes. This many-to-one communication paradigm is called convergecast. These applications generally require small delays for data gathering and time consistency of the data gathered. This time consistency is usually achieved by a small gathering period.

When the volume of data transmitted by any sensor is reduced, aggregation techniques are used to increase network efficiency and throughput. When several samples are transmitted in a single MAC (Medium Access Control) frame, the length of the frame is usually close to the maximum length allowed by the MAC protocol (e.g., 127 bytes for the IEEE 802.15.4 MAC protocol). As a consequence, no aggregation is possible in the intermediate nodes (*i.e.*, raw data convergecast). Other techniques must be investigated to achieve network throughput and efficiency. In this paper, we focus on raw data convergecast using multichannel techniques to ensure smaller gathering delays and a higher throughput.

In many real deployments of WSNs, the channel used by the WSN usually encounters perturbations, such as jamming, external interferences or noise caused by external sources (e.g., a polluting source, such as a radar) or other coexisting wireless networks (e.g., WiFi, Bluetooth). Commercial sensor nodes can communicate on multiple frequencies, as specified in the IEEE 802.15.4 standard. This reality has given birth to a **multichannel communication paradigm** in WSN. Multichannel WSNs significantly expand the capability of single-channel WSNs by allowing parallel transmissions and avoiding channels that are congested or whose performances are degraded by interfering devices.

Under heavy traffic conditions, contention-based protocols suffer from collisions and non-deterministic delays. In contrast, contention-free deterministic scheduling divides the time into frames, each frame comprising a certain number of time slots, and each node transmits data in its allocated slots. Conflicting nodes are not allowed to transmit in the same time slot on the same channel. Figure 1 depicts a data gathering frame. The frame can repeat over time, defining a cycle. The cycle length is equal to the duration between the beginnings of two successive frames, and it should meet the application requirements concerning data gathering delays. The frame consists of an activity period and an inactivity period. The activity period is subdivided into a control period, where nodes exchange control messages (e.g., hello messages to discover the neighborhood), and a scheduled period, where nodes transmit the user data towards the sink, according to pre-established scheduling. In the inactivity period, all nodes sleep to save energy, and no transfer is therefore possible in this period.

Figure 1. A data gathering frame.



On the one hand, since **the contention-free protocol** removes idle listening and overhearing, which are the main sources of energy drain, contention-free deterministic scheduling is appropriate for low power devices, since nodes turn off their radio in non-scheduled time slots ensuring energy efficiency and prolonging the network lifetime. More precisely, in a given time slot, only the sender and the receiver are awake, while all other nodes are sleeping. On the other hand, minimizing the number of slots in the contention-free period is crucial if we want to preserve the small data gathering delays. Our goal is to ensure that the data gathering is achieved in a single cycle and that all the packets generated in a cycle can be transmitted in a single cycle to the sink, assuming that all packets are periodically generated. With FIFO (First In First Out) queues, the worst case delay for delivering to the sink data generated by a sensor is equal to cycle-length + activity-period. This delay is obtained when the sensor whose transmission slot is the first in the activity period misses its opportunity to transmit and must wait until the next cycle to transmit its generated data. Furthermore, in the worst case, this packet is received by the sink in the last slot of the activity period. Notice that for any intermediate node, the message is always received in a slot preceding the slot where this node is scheduled to transmit. Hence, no additional delay is introduced by the forwarding. Furthermore, we notice that in a single channel context, the transmissions of interfering nodes are scheduled in different slots; whereas in a multichannel context, the transmissions of interfering nodes are scheduled in the same slot, but on different channels, provided that both the sender and the receiver have an available radio interface. As a result, the throughput is increased. In addition, the throughput can be further improved by equipping the sink with multiple radio interfaces.

The multichannel time slot assignment problem was proved to be NP (Non-deterministic Polynomial-time)-hard by Ghosh *et al.* in [1].

Moreover, in any raw data convergecast, nodes near the sink suffer from heavy traffic, and so, the time slot assignment should be traffic-aware: any node is granted a medium access time proportional to its traffic demand.

Therefore, in this paper, we tackle two problems: (1) the computation of lower bounds for a raw data convergecast in a multichannel WSN with heterogeneous demands of nodes and a sink with multiple radio interfaces and (2) **the design of an adaptive slot assignment algorithm** preserving the initial optimized assignment. This algorithm should adapt to:

- Retransmission of a message that has not been acknowledged at the MAC layer.
- Temporary change in the application needs due to alarms, for instance. Alarms are associated with strong delay requirements and must be reliably delivered to the sink. Several possibilities exist:
  - if a slot was assigned to the sender node, the alarm is sent first, taking the place of the regular data. This regular data will then be sent in an additional slot, granted by the adaptive solution.
  - if no slot was assigned to the sender node; in the worst case, the alarm is sent in the next cycle following the control message requesting the alarm transfer.

We aim to optimize the management of available resources, taking advantage of bandwidth spatial reuse to meet application requirements in terms of determinism, delay, throughput and reliability. Intuitively, we would like to make the slot assignment more flexible and able to adapt to application and environment variability. In other words, we would like to benefit from the intrinsic adaptivity of CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) in a deterministic slotted scheme.

We have to take into account several constraints, particularly in large-scale WSNs. First, the computation time is constrained by the computing power of the embedded node in charge of the slot assignment, usually the sink. Second, due to the small payload size in WSNs based on the IEEE 802.15.4 standard, the sink may need to transmit several messages to disseminate the amount of data defining the slot assignment. Third, to prolong the network lifetime, node residual energy must be saved by transmitting and receiving fewer messages. That is why **we propose a solution based on an incremental technique**. The slot assignment is built initially and totally rebuilt only when there are non transient changes in transmission needs. Indeed, it is preferable to transmit only the differential data corresponding to a temporary slot assignment. The incremental technique we propose, called AMSA, an Adaptive Multichannel Slot Assignment algorithm (see Section 4), consists of slight modifications of the initial slot assignment, computed by MODESA [2] (see Section 3) in order to adapt to new temporary demands of the nodes.

This paper is organized as follows. In Section 2, we give a brief state of the art related to multichannel slot assignment for raw data convergecast in WSNs. Particular attention is given to protocols that have tackled the adaptivity aspect. Section 3 extends our study published in [2] to heterogeneous traffic demands. By heterogeneous demands, we mean that nodes require different numbers of slots to transmit the data they generate in the convergecast tree. Notice that the demand of any node accounts only for the data generated by the node and not for the data it received from its children and has to forward to its parent. We also provide lower bounds on the number of slots required in various multichannel topologies. We then present MODESA, a time slot assignment algorithm that is close to the optimum. In Section 4, we define the incremental slot assignment problem in a multichannel environment and formalize it using integer linear programming. We then propose AMSA. We evaluate its performances in Section 5, comparing the number of additional slots assigned with an optimized allocation (the best case) and a naive one (the worst case). Finally, we conclude in Section 6.

2. Related Work

2.1. Multichannel Slot Assignment

To avoid a premature failure of the network, it is crucial to have a medium access control protocol that avoids wasting energy (idle listening, collisions and retransmissions). Typically, a contention-free deterministic medium access is efficient against this threat.

In [3], the authors address jointly the link scheduling and channel assignment for convergecast in networks operating according to the WirelessHART standard. The authors prove that for linear networks with N single buffer devices, the minimum schedule length obtained is $(2N-1)$ time slots with $\lceil N/2 \rceil$ channels. They also present an algorithm with time complexity, $O(N^2)$, to generate the time and channel optimal convergecast schedule. The solution does not provide spatial reuse of the bandwidth and is restricted to linear topologies that are not suitable for all real deployments. In addition, they focus only on single radio interface devices.

Tree-Based Multi-Channel Protocol (TMCP) [4] was designed to support data collection traffic. It begins by partitioning the network into multiple subtrees and then assigns a different channel to each

subtree rooted at a sink child. All nodes belonging to the same subtree use the same channel, thereby minimizing the interferences between subtrees. However, TMCP does not eliminate interferences within a subtree. After the channel assignments, time slots are assigned to the nodes. The drawback of TMCP is that the sink must be equipped with a number of radio interfaces and a number of channels equal to the number of its children.

Incel *et al.* [5], proved that if all interfering links are removed (with the required number of channels), the schedule length for raw data convergecast is lower bounded by $max(2n_k - 1, N)$ where n_k is the maximum number of nodes in any subtree rooted at a sink child of the routing tree and N is the number of source nodes. They also proposed a convergecast scheduling algorithm, Joint Frequency and Time Slot Scheduling (JFTSS), that achieves this lower bound only in topologies where the routing tree has an equal number of nodes on each branch.

All these studies have focused only on homogeneous traffic or a sink with a single radio interface. In this paper, we generalize these previous results, giving minimal bounds for raw data convergecast, taking into account both heterogeneous node demands and a sink with single or multiple radio interfaces.

2.2. Adaptive Slot Assignment

The above existing time slot assignment protocols for WSNs often operate independently of the dynamic application demands, e.g., retransmissions, alarms, additional injected traffic. This is, however, a particular challenge in systems with strong reliability or alarm-based systems. In this section, we present some studies that deal with the adaptivity of time slot assignments. While several studies have tackled this concern in mono-channel WSNs, a comprehensive technique is still lacking in multichannel WSNs.

In [6], the authors present EP-TDMA, an Evolutionary dynamic slots assignment algorithm which enhances P-TDMA, itself based on TDMA (Time Division Multiple Access) with CSMA/CA competition mechanism during the control phase. More precisely, the cycle consists of two steps: (1) a control phase, which is subdivided into a claim subphase and a response subphase, and (2) an information phase, where data transmission takes place. First, each phase consists of N slots, where N is the number of nodes in the network. Therefore, each node has its own slot. A node is said to be active if it has a packet to transmit in its assigned slot. During the claim phase, active nodes add information in claim packets to announce their need for more slots if they encounter a high workload. Then, each node collects its neighbors' claim packets and transmits a response packet in a response slot. Therefore, by exchanging response packets, any node, u, can compute the slots it can compete for (i.e. slots during the information phase) and the set of neighbors up to two hops away. Node u cannot use any slot already used by any other node in this set. To solve the slot competition between nodes, any node gets a random priority for each slot, except its own slot, for which it has the highest priority. After the exchange of claim and response packets, the nodes determine active nodes and active nodes' traffic within two hops. Then, each node knows all the slots it can obtain, *i.e.*, the slots for which it has the highest priority. This technique allows active nodes to obtain additional slots without the risk of collisions.

EP-TDMA enhances P-TDMA [7] by assigning slots to nodes based on their priority and workload. Nevertheless, it inherits the drawbacks of P-TDMA: under heavy traffic, the contention during the control phase increases, resulting in a low delivery ratio.

TDMA Scheduling with Adaptive Slot-Stealing and Parallelism (TDMA-ASAP) [8] enhances the TDMA protocol by allowing parallel transmissions, unused slot stealing and adaptive sleeping periods for sensors. The authors consider a tree-based routing scheme. A constraint graph is constructed taking into account not only parent-child constraints, but also the interference constraints. They apply graph coloring to the constraint graph to find a short schedule. When the traffic load is low, TDMA-ASAP allows slot stealing: a "slot stealer" node can take the vacant slot of its brothers and use it to transmit. A parent node is allowed to sleep if neither the owner nor the stealer sensor is transmitting. Nevertheless, if a node v steals the slot of u and w is a hidden node of v, then a collision may occur. That is why, the authors propose different variants of slot stealing (moderate, conservative and aggressive stealing) that handle collisions differently.

However, TDMA-ASAP does not support immediate acknowledgment (*i.e.*, any unicast message is acknowledged in the slot of the sender) and supposes in-network aggregation at each receiving node. Nevertheless, for many critical applications, aggregation may be incompatible with strong delay constraints or maximum message size. Thus, each packet has to be scheduled individually.

Compared to the previous study [8], the authors of [9] enhance TDMA-ASAP by minimizing the latency of the report of any event occurring during the cycle, assuming that events are uniformly distributed in a cycle:

- A slot spreading algorithm: Nodes in TDMA-ASAP steal only from adjacent or nearby slots. The proposed algorithm spreads out children slots evenly in the schedule.
- Stealing both in upstream and downstream communication: The authors assume that any cycle consists of upstream slots and optional downstream slots spread throughout the cycle. Since downstream slots are not necessary in every cycle, they can be stolen by upstream communications. If the children do not receive a control message from their parents, they treat the downstream slot reserved for their parent as a stealable slot.

In [10], Kanzaki *et al.* propose Adaptive Slot Assignment Protocol with Slot Migration (ASAP/SM), which is an extension of their previous work on Extended Adaptive Slot Assignment Protocol (E-ASAP). This latter is not traffic-aware, because nodes that have heavy traffic do not have enough assigned slots. ASAP/SM obeys E-ASAP rules for slot allocation. The decision of migrating to N_j a slot assigned to N_i depends on certain conditions: the amount of traffic at node N_i (channel requirements), channel utilization (number of slots assigned to N_i divided by the frame length) and the gap value (the difference between two nodes, N_i and N_j, in terms of channel utilization and traffic load). The cycle of slot migration is called an "update cycle". During this cycle, a node begins by calculating the gap value of its neighbors. Then, it sets priorities according to gap values: the larger the gap is, the higher the priority. For a node N_i, the target node (*i.e.*, the first node to be stolen) is its neighbor with the highest priority. Node N_i should select a slot, already allocated to its neighbor, to become its own. If node N_i does not find a slot that can be migrated, the next node with the highest priority is set as the next target node. In addition, to avoid slot migration oscillations, which lead to more control traffic, the authors propose a mechanism to cancel slot migration according to certain criteria.

Tselishchev *et al.* [11] target retransmission strategy for body area networks taking into account energy criteria and the temporal variation in the wireless links. The authors consider a single hop network, where a hub communicates directly with n sensors. The time is divided into rounds of m slots ($m > n$). In each round, the first n slots are dedicated to regular transmissions of packets. Hence, $(m - n)$ slots are left as sparse. When a sensor fails to transmit a packet, it is added to a list called the "retransmission eligibility list". This list contains nodes that have backlogged packets and an energy level greater than a specific energy threshold. Then, a node is randomly selected from the list and assigned a sparse slot. The process is repeated until the list becomes empty. They propose a technique called "Flip + Spread", which enhances the successful transmission ratio in a round that includes a retransmission. Assuming that the first transmission of a message m fails during the first round, this technique places the second transmission attempt of m at the end of the second round. The first transmission attempt of the next message after m is placed in the second round in the middle of the first transmission (previous round) and the second transmission attempt of m. Therefore, transmissions are spread over two rounds, provided that the number of nodes requesting a retransmission in any round is less than $m - n$. The shortcoming of this work is that the authors restrict their study to single hop networks. It is generally assumed that WSNs are deployed over a large area and, hence, multi-hop convergecast structure is usually adopted.

As a conclusion, we observe that none of the solutions studied previously guarantees the assignment of slots along the path of the raw data convergecast tree. Hence, there is no guarantee that the data, including those corresponding to the additional demands, will be gathered in a single frame. We propose AMSA to meet this requirement, if this is allowed by the inactivity period length. Furthermore, we notice that changes in transmission needs are handled either by a new slot assignment or by an incremental technique. For the reasons given in the previous section, we prefer the latter for transient changes.

3. Multichannel Slot Assignment with Heterogeneous Demands

To overcome the limitations of a single channel as mentioned in the previous section, we address jointly the optimal schedule and channel allocation problems.

3.1. Multichannel Slot Assignment Problem

We require a multichannel slot assignment technique that, under the assumptions described below, minimizes the number of slots assigned and ensures that no two conflicting nodes transmit simultaneously on the same channel.

As illustrated in Figure 1, we can now refine the structure of the activity period, which consists of a control period and a scheduled period. During the control period, nodes exchange messages to discover their neighbors and build a route to the sink. The scheduled period consists of the slots assigned by the time slot assignment algorithm. The cumulated duration of the scheduled and inactivity periods is expressed as a number of time slots, all the time slots being of constant size. By definition, this number is called $MaxSlot$. We also use the following definitions:

◦ **D1. Conflicting nodes:** Two nodes are said to be *conflicting* if and only if they cannot transmit to their respective parent in the routing tree, in the same time slot and on the same channel.

◦ **D2. Subtree:** $subtree(i)$ is the subtree rooted at the sink child, i.

◦ **D3. Subline:** $subline(i)$ is the subline starting at the sink child, i.

◦ **D4. Neighborhood** Two nodes u and v are one hop neighbors if and only if they can hear each other. In other words, the wireless link between them is symmetric. 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 one hop node of v.

Interested readers can refer to [2], where we presented a formalization of the multichannel time slot problem as a linear program.

3.2. Assumptions

We distinguish two types of assumptions:

- The primary assumptions that are required by the solutions proposed (scheduling algorithms), the computation of the theoretical bounds and the performance evaluation of the solution;
- The secondary assumptions that are not required by the solutions to work correctly. They are only introduced to simplify the computation of the theoretical bounds and the performance evaluation.

3.2.1. Primary Assumptions

◦ **A1. Heterogeneous demands of nodes:** The sink is in charge of gathering data from the other nodes. In each data gathering cycle, each ordinary node (*i.e.*, any node that is not a sink) transmits its own data to its parent in the data gathering tree rooted at the sink and forwards the data received from its children. Let $d_u > 0$ be the number of slots needed by any ordinary node, u, to transmit the data it generates in a cycle. For instance, a node, u, with $d_u = 3$ will be assigned three slots to transmit its own data. In addition, it will be assigned the number of slots required to forward the data received from its children.

◦ **A2. Node radio interface:** The sink is the only node having $n_{interf} \geq 1$ radio interfaces. Any ordinary node has a single radio interface.

◦ **A3. Available channels:** For the sake of simplicity, we assume that at each node, $n_{channel} > 1$ channels are available. These channels are numbered from one to $n_{channel}$. The routing convergecast tree is the same on any available channel. Since any ordinary node, u, has a single radio interface, one channel at most is active at any time on node, u. For the sink, there are at most $min(n_{interf}, n_{channel})$ active channels simultaneously.

◦ **A4. Slot size:** All time slots have an equal size. The slot size allows the transmission of p packets with integer, p, higher than or equal to one.

◦ **A5. Topology:** For any topology, the inputs of the scheduling algorithm include the links between nodes, the conflicting nodes and the routing tree. Notice that only symmetric links are used in the routing tree.

3.2.2. Secondary Assumptions

As mentioned above, these assumptions are not required by the scheduling algorithms, but serve to simplify the computation of theoretical bounds for raw data convergecast.

◦ **A6. Ideal environment:** We assume an ideal environment with no message loss and no node failure.

◦ **A7. Topology links:** We also assume that the only links present in the topology are those belonging to the convergecast tree.

◦ **A8. Conflicting nodes:** We assume a graph-based interference model, also called a protocol model: any two nodes, u and v, that are one-hop or two-hop neighbors are conflicting. Thus, they are not allowed to transmit to their respective parent in the tree in the same time slot and on the same channel. For any node, the set of its conflicting nodes is the same on any considered channel.

3.3. Theoretical Bounds on the Number of Slots for a Raw Data Convergecast

We can now give some lower bounds on the number of slots for a raw data convergecast with heterogeneous demands. These bounds are established under the primary and secondary assumptions given previously.

Lemma 1 *In any WSN with heterogeneous demands of nodes, a lower bound on the number of slots required by a raw data convergecast is* $\lceil \frac{\sum_{u \neq sink} d_u}{g} \rceil$*, where* $g = min(n_{interf}, n_{child}, n_{channel})$*, with* $n_{interf} \geq 1$ *is the number of interfaces of the sink,* n_{child} *the number of children of the sink,* $n_{channel} > 1$ *the number of available channels at each node and* d_u *is the number of slots needed by any ordinary node,* u*, to transmit its own data to its parent.*

Proof: In any network with heterogeneous node demands, the sink has to receive $\sum_{u \neq sink} d_u$ messages from its children. The number of simultaneous transmissions to the sink is limited by the number of children, the number of sink interfaces, as well as the number of available channels (each interface using a channel different from the channels used by the other active interfaces). Hence, the number of slots needed is higher than or equal to $\lceil \frac{\sum_{u \neq sink} d_u}{min(n_{interf}, n_{child}, n_{channel})} \rceil$; hence, the lemma. ■

3.3.1. Linear Networks

Lemma 2 *In any linear network with heterogeneous demands of nodes, where each node has* $n_{channel} > 1$*, the minimum number of slots for a raw data convergecast is* $d_{child(sink)} + 2 \sum_{u \neq sink, u \neq child(sink)} d_u$*, whatever the number of interfaces that the sink has, where* d_u *is the number of slots needed by any ordinary node,* u*, to transmit its own data to its parent.*

Proof: Consider any linear network with heterogeneous demands of nodes. The child of the sink, denoted node v, needs to transmit $\sum_{u \neq sink} d_u$ packets to the sink and needs to receive $\sum_{u \neq sink, u \neq child(sink)} d_u$ packets from its child. Since node v has a single interface, these transmissions cannot overlap. As a consequence, a lower bound on the number of slots is equal to $d_v + 2 \sum_{u \neq sink, u \neq child(sink)} d_u$; hence, the Lemma. ■

Lemma 3 *In any linear network with heterogeneous demands of nodes, where each node has* $n_{channel} > 1$*, the algorithm that schedules:*

- *In the current time slot,* $t \leq 2 * \sum_{u \neq sink, u \neq child(sink)} d_u$,
  - *if* t *is odd, any node that is* $(2h+1)$ *hops away from the sink, with* $h \in [0, \lfloor \frac{N-1}{2} \rfloor]$, *where* N *is the number of nodes;*
  - *if* t *is even, any node that is* $2h$ *hops away from the sink, with* $h \in [1, \lfloor \frac{N-1}{2} \rfloor]$.
- *the child of the sink in the next contiguous* $d_{child(sink)}$ *slots.*

provides the minimum number of slots for a raw data convergecast.

Proof:

- In the first slot, the sink child transmits one packet of its own data.
- In any odd slot, t, with $1 < t \leq 2 * \sum_{u \neq sink, u \neq child(sink)} d_u$, the sink child transmits any packet received from its children in the previous slot. Consequently this schedule ensures that any node, different from the sink and its child, transmits simultaneously with its grand-parent. Since these two nodes are conflicting on the same channel, they are only allowed to transmit on different channels. For instance, if we consider the odd slots, nodes at depth $4h+1$ will transmit on the first available channel, whereas nodes at depth $4h+3$ will transmit on the second available channel. For the even slots, nodes at depth $4h$ will transmit on the first available channels, and nodes at depth $4h+2$ will transmit on the second available channel.
- In the $(2 * \sum_{u \neq sink, u \neq child(sink)} d_u + 1)^{th}$ slot, the sink child transmits the last last packet received from its child.
- In the next $(2 * \sum_{u \neq sink, u \neq child(sink)} d_u - 1)$ slots, it transmits its $\sum_{u \neq sink, u \neq child(sink)} d_u - 1$ own packets.

With this schedule, any node alternates transmit and receive slots until it has transmitted its own data and the data generated by its downstream nodes. Hence, the child of the sink is kept busy in all slots, leading to a number of slots equal to this given by Lemma 2; hence, the lemma. ■

3.3.2. Multi-Line Networks

Theorem 1 *In any multi-line network with heterogeneous demands of nodes, a lower bound on the number of slots for a raw convergecast is Max(*$\lceil \frac{\sum_{u \neq sink} d_u}{g} \rceil$, $max_{i=child(sink)}(d_i + 2 \sum_{v \neq i, v \in subline(i)} d_v + \delta)$, *where* $g = min(n_{interf}, n_{child}, n_{channel})$, d_u *is the number of slots needed by any ordinary node,* u, *to transmit its own data to its parent,* n_{interf} *the number of interfaces of the sink and* $\delta = 1$ *if the* $(g+1)^{th}$ *child of the sink requires the same number of slots as the first one, assuming that sink's children are ordered according to the decreasing order of slots required, that is,* $d_i + 2 \sum_{v \neq i, v \in subline(i)} d_v$ *for the sink child,* i, *and zero otherwise.*

Proof: The sink requires at least $\lceil \frac{\sum_{u \neq sink} d_u}{min(n_{interf}, n_{child}, n_{channel})} \rceil$ time slots to receive all the packets generated in the network during one cycle (see Lemma 1). Moreover, let us consider the subline rooted at any sink child, i. From Lemma 2, at least $\sum_{u \neq i \in subline(i)} d_u$ slots are required to receive data and $\sum_{u \in subline(i)} d_u$ slots to transmit data to the sink. Furthermore, let us order the sink children according to the decreasing order of the number of slots they need, that is, $d_i + 2 \sum_{v \neq i, v \in subline(i)} d_v$ for the sink child, i. If the

$(g+1)^{th}$ sink child requires the same number of slots as the first one, then its schedule will require an additional slot. Indeed, the schedule of this line requires the same number of slots as the first one. However, the sink child starts to transmit one slot later, because there is no available channel or interface. Consequently, it will end one slot after. Hence, the value of δ. Of course, the number of slots should be dimensioned to meet the strongest requirements of the subline; hence, the theorem. ■

3.3.3. Tree Networks

Theorem 2 *In tree networks with heterogeneous demands of nodes, a lower bound on the number of slots for a raw data convergecast is Max(*$\lceil \frac{\sum_{u \neq sink} d_u}{g} \rceil$, $max_{i=child(sink)}(d_i + 2\sum_{v \neq i, v \in subtree(i)} d_v + \delta)$, *where* $g = min(n_{interf}, n_{child}, n_{channel})$, d_u *is the number of slots needed by any ordinary node,* u, *to transmit its own data to its parent,* n_{interf} *the number of interfaces of the sink,* n_{child} *the number of sink children and* $\delta = 1$ *if the number of nodes of the* $(g+1)^{th}$ *child of the sink requires the same number of slots as the first one, assuming that sink children are ordered according to the decreasing order of slots required, that is,* $d_i + 2\sum_{v \neq i, v \in subtree(i)} d_v$ *for the sink child,* i, *and zero otherwise.*

Proof: Let $g = min(n_{interf}, n_{child}, n_{channel})$. The sink requires $\lceil \frac{\sum_{u \neq sink} d_u}{g} \rceil$ time slots to receive all the packets generated in the network, as seen in Lemma 1. Furthermore, each child, i, of the sink has $\sum_{v \neq i \in subtree(i)} d_v$ packets to receive and $\sum_{v \in subtree(i)} d_v$ packets to transmit. Moreover, let us order the sink's children according to the decreasing order of the number of slots they need, that is, $d_i + 2\sum_{v \neq i, v \in subtree(i)} d_v$ for the sink child, i. If the $(g+1)^{th}$ sink child requires the same number of slots as the first one, then its schedule will require an additional slot. Indeed, the schedule of this subtree requires the same number of slots as the first one. However, the $(g+1)^{th}$ sink child starts to transmit one slot later than the first one, because at the first slot, all the available interfaces of the sink or all the available channels are used by the g first children of the sink. Consequently, the $(g+1)^{th}$ sink child will end one slot after; hence the value of δ. Of course, the minimum number of slots should work for the subtree requiring the highest number of slots; hence the theorem. ■

An example justifying the existence of δ in the lower bound for tree topologies is given in Figure 3.

3.3.4. T_s and T_n Configurations

By definition, a configuration is given by a topology (set of nodes and links) and the initial demands of nodes.

In the case of multiline and tree networks, we define two types of configurations, T_s and T_n:

- A configuration is denoted T_s when the optimal number of slots is imposed by the most demanding subtree rooted at a sink child, i. Its demand is equal to $d_i + 2\sum_{v \neq i, v \in subtree(i)} d_v$.
- A configuration is denoted T_n when the optimal number of slots depends only on the total number of demands and $g = min(n_{interf}, n_{child}, n_{channel})$. It is equal to $\lceil \frac{\sum_{u \neq sink} d_u}{g} \rceil$. Notice that a T_n configuration corresponds to a Capacitated Minimal Spanning Tree, where each branch has a total demand for slots, $\leq \lceil \frac{\sum_{u \neq sink} d_u}{g} \rceil$ [12].

See Section 3.5 for an illustrative example.

3.3.5. Specific Cases: Sink with a Single Radio Interface or Homogeneous Demands

In the specific case of a sink with a single radio interface, a lower bound on the number of slots for a raw data convergecast in any topology with heterogeneous demands of nodes is given by Max($\lceil \frac{\sum_{u \neq sink} d_u}{g} \rceil, \sum_{u \neq sink} d_u$), where d_u is the the number of slots needed by any ordinary node, u, to transmit its own data to its parent.

Indeed, when the sink has a single radio interface, it has to receive successively all the messages sent to it. Hence, the minimum number of slots required is given by Max($\lceil \frac{\sum_{u \neq sink} d_u}{g} \rceil, \sum_{u \neq sink} d_u$).

We also notice that if each node in the topology needs one slot to transfer its own data, we get the results published in [2]. Hence, Section 3 is a generalization to heterogeneous demands of nodes.

3.4. MODESA Algorithm

MODESA, a centralized raw data convergecast scheduling, was proposed in [2]. It takes into account the availability of multiple channels to reduce the activity period in the data gathering frame, while ensuring a fair medium access. MODESA builds the primary schedule after having collected the required information (e.g., topology links, node demands, routing tree, conflicting nodes) at the sink. The primary schedule is then broadcast to all nodes in a multihop fashion along the routing tree.

MODESA builds the multichannel schedule slot by slot. At each iteration, MODESA assigns the current slot to one or several tuples $(slotnumber, sender, receiver, channel)$, applying the following rules:

1. Any node has a *dynamic priority*. The priority is equal to $remPckt * parentRcv$, where $remPckt$ is the number of packets the node has in its buffer at the current iteration. $parentRcv$ is the total number of packets the parent of the node has to receive in a cycle. The idea behind this heuristic is to reduce the number of buffered packets by favoring nodes that have packets to transmit to a parent with a high number of packets to receive.
2. Nodes compete for the current time slot if and only if they have data to transmit.
3. In addition to being allowed to transmit in a slot, a node and its parent must have an available interface.
4. For any slot, the first scheduled node is the node with the highest priority among all the nodes that have data to transmit. If several nodes have the same priority, MODESA chooses the node with the smallest identifier. The selected node is scheduled on the first available channel, c.
5. Any competing node can be scheduled in the current time slot on channel c if and only if it does not interfere with nodes already scheduled on channel c in this slot.
6. Conflicting nodes that interfere with nodes already scheduled in this slot are scheduled on a different channel, if one is available. Otherwise, in a next slot.

3.5. Illustrative Example

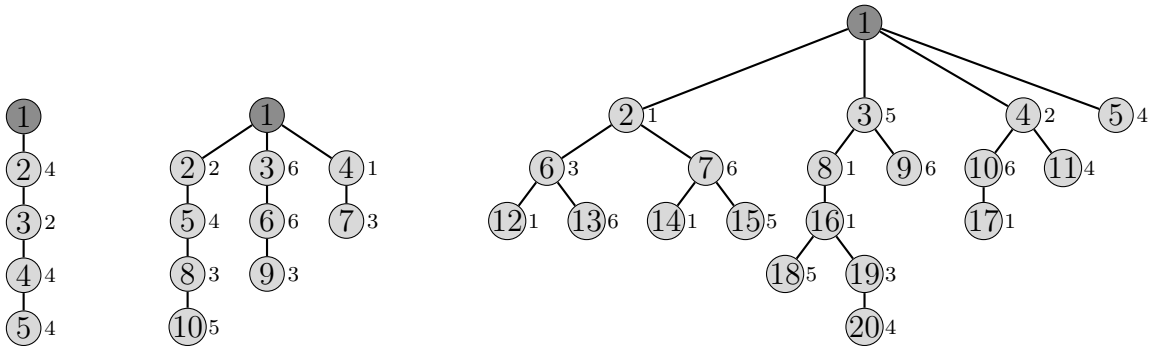
In this illustrative example, a slot allows the transmission of a single packet.

An optimal multichannel time slot assignment can be obtained by linear programming tools, such as GLPK (GNU Linear Programming Kit) [13] based on the model described in [2]. Figure 2 shows

the optimal number of slots, n_{slot}, for different single-sink topologies (linear 2(a), multiline 2(b) and tree 2(c)) with various numbers of sink interfaces, n_{interf}, and channels, $n_{channel}$. These optimal results are reached by the MODESA algorithm presented in Section 3.4.

T_s configurations are depicted in Figure 2b,c, both with two or three sink interfaces. In Figure 2c, there are two subtrees having the highest demand, rooted at nodes 2 and 3, respectively. Moreover, since the number of subtrees having the highest demand is equal to $g = min(n_{interf}, n_{child}, n_{channel}) = 2$, then $\delta = 0$, according to Theorem 2.

Figure 2. The optimal number of slots, n_{slot}, for various topologies with different numbers of sink interfaces, n_{interf}, and channels, $n_{channel}$, with the notation: (n_{interf};$n_{channel}$)=n_{slot}. (**a**) (1;2) = 24; (**b**) (1;2) = 33 (2;2) = 26 (3;3) = 26; (**c**) (1;2) = 65 (2;2) = 45 (3;3) = 45.



T_n configurations are depicted in Figure 2b,c with a single sink interface.

We can observe that a given topology may belong to one type or another depending on the number of sink interfaces. For instance, the topology depicted in Figure 2c belongs to the first type when the sink has a single radio interface and to the second type if the sink has two interfaces.

Figure 3a,b depict the same tree topology, but with different demands of node 5, assuming three sink interfaces and three channels. Figure 3a illustrates the case where the $(g+1)^{th} = 4^{th}$ sink child requires the same number as the $g = 3$ first ones. Its schedule requires one more slot than the case presented in Figure 3b. This is an illustration of Theorem 2.

Figure 3. Case where $\delta = 1$ when the sink is equipped with three radio interfaces and three channels. (**a**) (3;3) = 14; (**b**) (3;3) = 13.

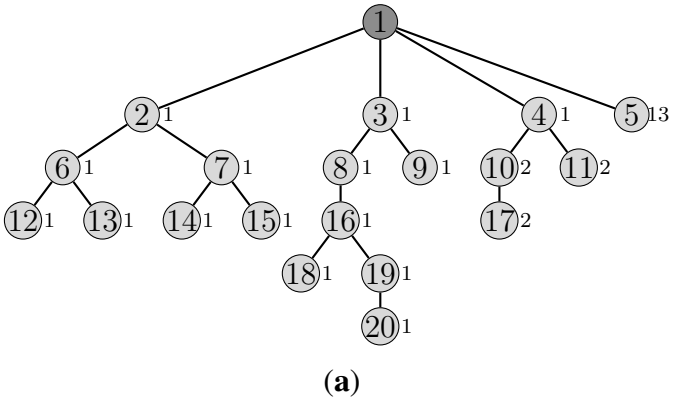
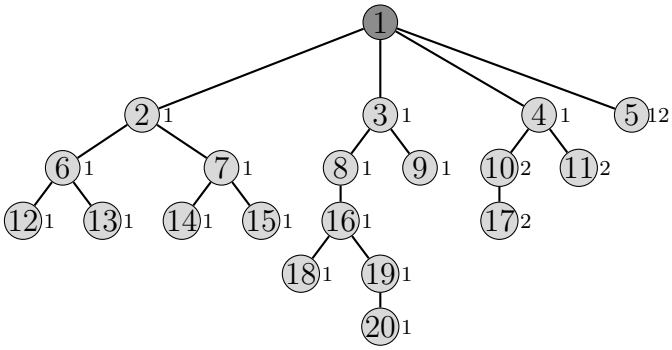


Figure 3. *Cont.*



(**b**)

4. Adaptive Multichannel Slot Assignment

As seen in Section 2.1, many existing time slot assignments lack flexibility. They are unable to adapt to additional demands (e.g., retransmission, new application needs, alarms). That is why in this section, we are looking for an incremental technique that is able to update a given optimized time slot assignment to meet new application needs. This technique should preserve the initial time slot assignment, provided, for instance, by MODESA, and minimize the number of extra slots added to the initial slot assignment.

4.1. Definitions

First, we introduce some definitions:

◦ **D5. Primary schedule:** is the initial optimized time slot assignment obtained with MODESA; it includes only slots needed by regular messages. These slots are said to be regular. Consequently, the number of regular slots granted to any ordinary node is equal to the sum of its initial demand and the initial demands of all its descendants.

◦ **D6. Bonus slot:** is a temporary slot assigned after a change notification (e.g., temporary change in the application need, retransmission, *etc*.).

◦ **D7. Secondary schedule:** is the optimized assignment of bonus slots obtained with AMSA.

◦ **D8. Slot path:** is a sequence of slots that allows the transmission of a message from any given ordinary node to the sink.

◦ **D9. Extra slot:** is a bonus slot that is appended at the end of the primary schedule. Any extra slot increases the length of the activity period.

◦ **D10. Complementary schedule:** is the optimized time slot assignment obtained with MODESA taking into account only the requests of bonus slots. It starts at the end of the primary schedule.

4.2. Assumptions

We also introduce some additional primary assumptions, which are required by the adaptive multichannel slot assignment.

◦ **A9. Service differentiation policy:** Each node transmits its messages according to their priority. Different priority levels can be distinguished:

- alarms have the highest priority,
- retransmissions of regular messages have medium priority,
- regular messages have the lowest priority.

◦ **A10. Computation time for bonus slot assignment:** We assume that the computation time for bonus slot assignment plus the time of the extra slots is less than the duration of inactivity.

4.3. Model

The network is formalized as a graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices representing the nodes of the network and $\mathcal{E}$ is the set of edges representing the communication links between the nodes.

Let $\mathcal{V} = \mathcal{V}_n \bigcup \mathcal{V}_s$, where $\mathcal{V}_n$ is the set of ordinary nodes and $\mathcal{V}_s$ represents the set of sinks, with $\mathcal{V}_n \bigcap \mathcal{V}_s = \emptyset$. For each node, $v \in \mathcal{V}$, we define $Conflict(v)$, the set of conflicting nodes that interfere with v when transmitting on the same channel. Moreover, let i_v denote the number of physical interfaces available at any node, v. For any ordinary node, $n \in \mathcal{V}_n$, let r_n correspond to the number of bonus slots requested by n to transmit the additional packets it generates.

Let $\mathcal{E}^+(v)$ denote the set of links through which a node, v, can transmit. Let $\mathcal{E}^-(v)$ be the set of links through which a node, v, can receive. For any link, $e \in \mathcal{E}$, let $f_{e,n}$ denote the number of slots needed to transmit over the link, e, the additional packets generated by node, n.

Let $\mathcal{C}$ be the set of channels usable for any transmission. The collision-free period is composed of slots, t, in the interval, $[1; T_{max}]$, where T_{max} denotes an upper bound of the frame length. This bound is reached when all packets are sent sequentially on the same channel. We then have: $T_{max} = \sum_n \sum_e f_{e,n} * depth_n$, where $depth_n$ is the depth of node, n, in the data gathering tree.

Let the parameter $A_{e,c,t}$ correspond to the activity of a link, e, on the channel, c, in the time slot, t, in the primary schedule, *i.e.*, $A_{e,c,t} = 1$, if and only if there is a transmission of a packet on the link, e, on the channel, c, in the time slot, t, in the primary schedule, and $A_{e,c,t} = 0$, otherwise. This binary parameter is given by the MODESA algorithm.

We define $b_{e,c,t}$ as bonus slot assignment for a link, e, on the channel, c, in the time slot, t, *i.e.*, $b_{e,c,t} = 1$, if and only if there is a transmission of a packet on the link, e, on the channel, c, in the time slot, t, in the bonus assignment, and $b_{e,c,t} = 0$, otherwise.

Furthermore, let u_t be the use of a slot, t, in other words, $u_t = 1$ means that there is at least one link activity (activity in the primary or secondary schedule) on at least one channel in the slot, t, and $u_t = 0$ denotes an empty slot.

Table 1 summarizes the inputs and variables of our model.

Table 1. Inputs and variables of the model.

Inputs	
$\mathcal{E}$	set of links in the topology
$\mathcal{V}$	set of nodes in the topology
$\mathcal{C}$	set of available channels
i_v	number of interfaces of the node, v
$Conflict(v)$	set of nodes conflicting with v
r_n	number of bonus slots requested by node, n
$A_{e,c,t}$	activity of the link e in time slot, t, on channel, c,
Variables	
u_t	utility of slot, t
$f_{e,n}$	number of slots needed to transmit over link, e, the additional packets generated by node, n
$b_{e,c,t}$	assignment of bonus slot, t, to link, e, on channel, c,

The objective is to minimize the number of slots, $t \leq T_{max}$:

$$min \sum_{t \leq T_{max}} u_t$$

with the following constraints:

$$\forall c \in \mathcal{C}, \forall e \in \mathcal{E}, t \leq T_{max}, \quad b_{e,c,t} + A_{e,c,t} \leq u_t \tag{1}$$

Constraint 1 binds the use of a time slot to at least the activity of one link on any channel in this slot in the primary schedule or the secondary schedule.

$$\forall v \in \mathcal{V}, \forall e \in \mathcal{E}^+(v), \forall w \in Conflict(v), \forall e' \in \mathcal{E}^+(w), \forall c \in \mathcal{C}, t \leq T_{max}, \quad b_{e,c,t} + A_{e',c,t} \leq 1 \tag{2}$$

Constraint 2 guarantees that a node cannot transmit while one of its conflicting nodes is already scheduled in the same slot on the same channel in the primary schedule.

$$\forall v \in \mathcal{V}, \forall e \in \mathcal{E}^+(v), \forall w \in Conflict(v), \forall e' \in \mathcal{E}^+(w), \forall c \in \mathcal{C}, t \leq T_{max}, \quad b_{e,c,t} + b_{e',c,t} \leq 1 \tag{3}$$

Constraint 3 ensures that for the bonus slot assignment, two conflicting nodes do not transmit on the same channel in the same time slot.

$$\forall v \in \mathcal{V}, t \leq T_{max}, \quad \sum_{c \in \mathcal{C}} \sum_{e \in \mathcal{E}^+(v)} b_{e,c,t} + \sum_{c \in \mathcal{C}} \sum_{e' \in \mathcal{E}^-(v)} b_{e',c,t} + \sum_{c \in \mathcal{C}} \sum_{e'' \in \mathcal{E}^+(v)} A_{e'',c,t} + \sum_{c \in \mathcal{C}} \sum_{e''' \in \mathcal{E}^-(v)} A_{e''',c,t} \leq i_v \tag{4}$$

Constraint 4 limits the number of simultaneous communications for a node, during the primary schedule and the bonus assignment, to its number of interfaces.

$$\forall e \in \mathcal{E}, \quad \sum_{n \in \mathcal{V}_n} f_{e,n} = \sum_{c \in \mathcal{C}} \sum_{t \leq T_{max}} b_{e,c,t} \tag{5}$$

Constraint 5 ensures the mapping between the activities on all channels and the packets sent on links.

$$\forall n \in \mathcal{V}_n, \quad \sum_{e \in \mathcal{E}^+(n)} f_{e,n} = r_n \tag{6}$$

$$\forall i \in \mathcal{V}_n, \quad \sum_{n \in \mathcal{V}_n} \sum_{e \in \mathcal{E}^+(i)} f_{e,n} = r_i + \sum_{n \in \mathcal{V}_n} \sum_{e \in \mathcal{E}^-(i)} f_{e,n} \tag{7}$$

$$\forall n \in \mathcal{V}_n, \quad \sum_{s \in \mathcal{V}_s} \sum_{e \in \mathcal{E}^-(s)} f_{e,n} = r_n \tag{8}$$

Constraints 6–8 express the conservation of messages, respectively, at the nodes requesting a bonus slot, at intermediate nodes and at the sinks.

$$\forall n \in \mathcal{V}_n, t \leq T_{max}, \quad \sum_{c \in \mathcal{C}} \sum_{e \in \mathcal{E}^+(n)} b_{e,c,t} \leq r_n + \sum_{c \in \mathcal{C}} \sum_{e \in \mathcal{E}^-(n)} \sum_{t' \in \{1..t-1\}} b_{e,c,t'} - \sum_{c \in \mathcal{C}} \sum_{e \in \mathcal{E}^+(n)} \sum_{t' \in \{1..t-1\}} b_{e,c,t'} \tag{9}$$

Constraint 9 makes certain that a packet is generated or received by a node before being transmitted or forwarded.

Notice that this model can be applied to compute the primary schedule by setting $A_{e,c,t} = 0 \quad \forall c \in \mathcal{C}, \forall e \in \mathcal{E}, t \leq T_{max}$

4.4. Theoretical Bounds on the Number of Extra Slots for a Raw Data Convergecast

We now provide lower and upper bounds for the number of extra slots added after the regular ones by any incremental solution. This should preserve the primary schedule and ensure that the data gathering, including the additional demands, is performed in a single frame.

Property 1 *The minimum number of extra slots is given by the difference between, on the one hand, the optimal slot assignment meeting for each node the sum of its initial and the new demands and, on the other hand, the number of regular slots.*

Proof: The number of slots required by the incremental algorithm is the sum of regular slots and extra slots. In the best case, this sum is equal to the number of slots obtained by the optimal algorithm to schedule both initial and new demands. Hence, the minimum number of extra slots is the difference between the number of slots given by the optimal algorithm and the number of regular slots. ■

Property 2 *The maximum number of extra slots is given by* $\sum_{u \ requesting \ node} depth_u \times r_u$*, where* $depth_u$ *is the depth of any ordinary node,* u*, in the convergecast tree and* r_u *is the number of bonus slots requested by* u*.*

Proof: The maximum number of extra slots added to the regular ones corresponds to the worst case, which occurs when there is no parallelism between the slot paths granted, corresponding to the additional demands. It is given by $\sum_{u \ requesting \ node} depth_u \times r_u$, where $depth_u$ is the depth of any ordinary node, u, in the convergecast tree; it is also the length of the path from this node to the sink, and r_u is the number of bonus slots requested by u. ■

We can improve this upper bound, by taking into account some parallelism in the transmissions. We then get:

Property 3 *The maximum number of extra slots is given by the* $MaxDepth + 2\sum_{u\ requesting\ node} r_u - 2F_{Maxdepth}$*, where* $MaxDepth$ *is the maximum depth in the convergecast tree of any ordinary node requesting bonus slots,* r_u *is the number of bonus slots requested by any ordinary node,* u*, and* $F_{Maxdepth}$ *is defined as follows:*

$$\begin{aligned} F_{Maxdepth} &= \sum_{k=1}^{maxDepth/2} 1_{2k} \quad \textit{if } Maxdepth \textit{ is even,} \\ &= \sum_{k=0}^{\lfloor maxDepth/2 \rfloor} 1_{2k+1} \quad \textit{otherwise,} \end{aligned}$$

with 1_k*= 1, if and only if there is at least one node,* u*, of depth,* k*, requesting bonus slots.*

Proof: We now build a schedule for all the requests for bonus slots. Like AMSA, this schedule assigns slots per path from the requesting node to the sink. Let u be the node with the greatest depth in the convergecast tree requesting bonus slots. Let $Maxdepth$ be its depth. If scheduled first, node u, will require $MaxDepth$ extra slots for the first bonus slot requested. For instance, in Figure 4, a node at $MaxDepth = 6$ will require six extra slots (see the grey slot path). For the next bonus slots required by u, if any, each of them will require two extra slots (see the black slot path in Figure 4 ending at slot 8). Indeed, the next transmission of u cannot be scheduled in the next slot, where the parent of u is transmitting. Similarly, all the bonus slots requested by the other nodes will need two additional extra slots. Hence, we get $MaxDepth + 2\sum_{w\ requesting\ node} r_w$ extra slots. We can also notice that in the slot, where u is scheduled, a node, v, of depth, $Maxdepth - 2$, can be scheduled simultaneously, but on another channel (assuming that the links in the convergecast tree are the only topology links). Applying this property recursively, we can then schedule in the $MaxDepth$ first slots, one bonus request by a node of depth, k, with k varying from $MaxDepth$ to two if $MaxDepth$ is even and from $MaxDepth$ to one, otherwise. Hence, we finally get the maximum number of extra slots equal to $MaxDepth + 2\sum_{u\ requesting\ node} r_u - 2F_{Maxdepth}$, with:

$$\begin{aligned} F_{Maxdepth} &= \sum_{k=1}^{maxDepth/2} 1_{2k} \quad \text{if } Maxdepth \text{ is even,} \\ &= \sum_{k=0}^{\lfloor maxDepth/2 \rfloor} 1_{2k+1} \quad \text{otherwise,} \end{aligned}$$

■

Notice that this bound does not take into account the possible parallelism between simultaneous transmissions of nodes belonging to the same depth. AMSA takes advantage of this parallelism.

Figure 4. Extra slots scheduling when $Maxdepth = 6$.

Depth \ Slot	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	-	1_2	-	1_4	-	1_6	-	2_6	-	3_6	-	1_5	-	2_5
2	1_2	-	1_4	-	1_6	-	2_6	-	3_6	-	1_5	-	2_5	-
3	-	1_4	-	1_6	-	2_6	-	3_6	-	1_5	-	2_5		
4	1_4	-	1_6	-	2_6	-	3_6	-	1_5	-	2_5	-	-	-
5	-	1_6	-	2_6	-	3_6	-	1_5	-	2_5	-	-	-	-
6	1_6	-	2_6	-	3_6	-	-	-	-	-	-	-	-	-

The performance evaluation reported in Section 5 will show how AMSA is close to the lower and upper bounds. We will also position AMSA with regard to an intermediate solution, where an optimized time slot assignment is computed by MODESA from the bonus slots requests and appended at the end of the primary schedule.

4.5. AMSA: Proposed Solution

4.5.1. Principles

The incremental solution we propose is based on the following principles. Each node that detects a change in its transmission needs (e.g., temporary change in the application need, retransmission, *etc*.), notifies the slot manager, usually the sink, by means of a control message. This control message is sent in the control part of the frame (see Figure 1) and forwarded to the parent in the convergecast tree, until it reaches the sink. AMSA builds the secondary schedule (see Algorithm 1). This is computed by the sink after having received the bonus requests sent, according to Algorithm 2. The sink tries to insert bonus slots into the primary schedule without increasing its length to meet the additional needs. If that is impossible, it takes new slots in the inactivity period. The sink transmits this new slot assignment downstream the tree, as the primary slot assignment was transmitted.

When a node receives the new slot assignment, it uses (indifferently) regular and bonus slots to transmit its messages according to the service differentiation policy expressed in Assumption A9. Notice that the bonus slots are allocated temporarily and are only valid for one data gathering frame. The associated use case corresponds to alarm transmissions or message retransmissions. If the application needs increase or decrease for a longer time, the new primary schedule is recomputed with MODESA.

4.5.2. AMSA Algorithm

The AMSA algorithm has two parts: one part is run by the sink in charge of slot assignment, whereas the other part is run by ordinary nodes. With AMSA, the sink processes the bonus requests from origin nodes one by one, assigning one bonus slot to the whole path to the sink, until all the bonus requests have been met. AMSA takes as input the primary schedule, the bonus requests and the conflicting nodes. It completes the primary assignment by adding one or several tuples $(slotnumber, sender, receiver, channel)$, applying the following rules:

1. Any node requesting a bonus slot has a static priority, which is equal to $depth_u * r_u$, where $depth_u$ depicts the depth of a node, u, in the convergecast tree and r_u is the number of requested bonus slots (see line 6 of Algorithm 1). This priority favors nodes requesting a longer slot path or a higher number of bonus slots. The goal is to minimize the number of extra slots. The length of a slot path is given by the depth of the requesting node. Since AMSA allocates slots per slot path, scheduling the longer path first helps AMSA to complete the schedule earlier.
2. Nodes having bonus requests are sorted according to their priorities: the node with the highest priority is selected first (see line 7 of Algorithm 1).
3. The sink serves the bonus requests from the selected node, assigning one bonus slot to the whole path to the sink (see lines 9 to 34 of Algorithm 1).

4. To be allowed to transmit in a slot, both the selected node node and its parent should have an available radio interface (see line 12 of Algorithm 1). Consequently, AMSA searches for the first slot where this condition is met.
5. AMSA searches for the first channel where this node does not conflict with the already transmitting nodes on the same channel. Hence, a node is scheduled in the earliest possible slot (see lines 13 to 22 of Algorithm 1).
6. Any ordinary node, u, maintains a counter, r_u, that corresponds to the number of bonus slots it will request: see Algorithm 2.

Algorithm 1 AMSA algorithm for the sink.

1: **Input:** $MaxChannel$ channels; $MaxSlot$ slots available in the scheduled and inactivity periods; a spanning tree, T, where each node, u, has a depth, $depth_u$, a set of conflicting nodes, $Conflict(u)$, and a request of bonus slots, r_u; a primary slot assignment with for each slot, t, the number of available radio interfaces for each node, u.
2: **Output:** The bonus slot schedule in the data gathering frame
3:
4: $\mathcal{N} \leftarrow$ the set of nodes having requested bonus slots
5: **while** $\mathcal{N} \neq \emptyset$ **do** // there are bonus slots to assign
6: Sort $\mathcal{N}$ according to the decreasing priority of nodes, with $prio_u \leftarrow depth_u \times r_u$
7: $u \leftarrow first(\mathcal{N})$ // assign slots to a path starting with node u
8: $t \leftarrow 1$ // starts from the first time slot
9: **while** u is not the sink **do** // assign a slot to node u
10: $ChannelFound \leftarrow False$
11: **while** $Not(ChannelFound)$ && $t \leq MaxSlot$ **do**
12: **if** the node, u, and its parent have an available interface in slot, t **then**
13: $c \leftarrow 1$ // selected channel
14: **while** $Not(ChannelFound)$ && $c \leq MaxChannel$ **do**
15: **if** node u does not conflict with nodes already scheduled on channel c in time slot t **then**
16: Assign time slot t to u on channel c
17: Update the request of node u and its parent
18: Update the priority of node u and its parent
19: $ChannelFound \leftarrow True$
20: **end if**
21: $c \leftarrow c + 1$
22: **end while** // channel
23: **end if** // available interface
24: $t \leftarrow t + 1$ // try another slot
25: **end while** // Slot
26: **if** the request of u has been totally served —— $Not(ChannelFound)$ **then**
27: **if** $u \in \mathcal{N}$ **then**
28: $\mathcal{N} \leftarrow \mathcal{N} \setminus \{u\}$
29: **end if**
30: **end if**
31: **if** $ChannelFound$ **then**
32: $u \leftarrow parent(u)$ // continue the assignment of the slot path considering the parent of u
33: **end if**
34: **end while** // slot path
35: **end while** // no pending demand

Algorithm 2 AMSA algorithm for any ordinary node, u.

Input: The additional application demand, Ad_u
Output: The number of bonus slots requested, r_u

$r_u \leftarrow Ad_u$
for all slots t where u transmits **do**
  if t is a regular slot && no acknowledgment received **then**
    $r_u = r_u + 1$
  else
    if t is a bonus slot && acknowledgment received **then**
      $r_u = r_u - 1$
    end if
  end if
  if t is the last slot where u transmits **then**
    if r_u **then**
      Send a BonusRequest message with r_u in the bonus field
    end if
  end if
end for

4.5.3. Discussions

AMSA brings the following advantages:

- *spatial reuse:* AMSA does not systematically require additional slots, due to its opportunistic behavior. Indeed, it takes advantage of spatial reuse to fill the slots with the new demands and, if that is impossible, adds a minimum number of slots. In both cases, AMSA ensures that (1) the number of available radio interfaces allows the transmission from the node considered to its parent and (2) no two conflicting nodes will transmit in the same slot on the same channel.
- *a unique algorithm that can be used both for retransmissions at the MAC level and new transmission needs at the application level.* The same algorithm is able to adapt to both application or MAC changes in their transmission needs. Furthermore, AMSA assigns a whole slot path. Indeed, each time a node requests an additional slot, the whole slot path corresponding to the slot sequence needed to reach the sink will, if possible, be granted.
- *optimized retransmissions and a simpler implementation:* on the one hand, we notice that with AMSA, only the sequence of slots starting with the transmitter that has not received the acknowledgment is allocated. On the other hand, the implementation is made simpler, because on any node, at any time, there is at most one pending message waiting for its acknowledgment.
- *an energy efficient convergecast:* Firstly, it minimizes the number of slots which is crucial from an energy point of view, as it allows nodes to sleep to save energy. On the other hand, conflict avoidance on the bonus and primary slots avoids collisions and, hence, contributes also to saving energy.

Furthermore, we compared the overhead induced by AMSA and MODESA regarding the number of messages and the complexity. Our proposed incremental technique requires fewer messages to provide the additional schedule: AMSA requires fewer messages than MODESA recalculated: only nodes that are involved in the assignment of bonus slots are destinations of the message giving the new schedule, unlike MODESA, where all nodes are involved. Regarding complexity, the following Table 2 summarizes the main differences between AMSA and MODESA.

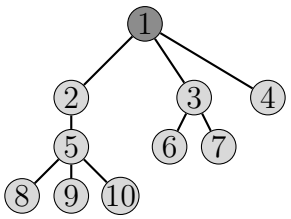
Table 2. Comparative table between AMSA and MODESA recalculated.

		AMSA	**MODESA recalculated**
Priority	nodes involved	only nodes requesting bonus slots	all nodes
	when it is computed	once at the beginning of the algo	at the beginning of any slot
Computation of the slot assigned to the highest priority node		any slot from the first one up to the current one	the current one
Number of transmissions to schedule		$\sum_{u \neq sink} depth_u * r_u$	$\sum_{u \neq sink} depth_u * (d_u + r_u)$

4.6. Illustrative Example

In this section, we provide two scenarios of bonus slot assignment. Figure 5 presents the topology of the network studied in our example. This network is composed of 10 nodes, including the sink (node 1). The sink has two radio interfaces. In both scenarios, there are two channels available at each node.

Figure 5. The topology of the network.



The optimal primary schedule given by MODESA is depicted in Figure 6. This assignment has a length of nine slots and corresponds to a primary demand of only one packet generated at each node except the sink. Slots are represented by a column. The notation, 2_1, means that node 2 transmits data to its parent on channel 1.

Figure 6. The primary schedule.

Slot / $tx \rightarrow rx$	1	2	3	4	5	6	7	8	9
$2 \rightarrow 1$	2_1		5_1		8_1		9_1		10_1
$3 \rightarrow 1$	3_2		6_2		7_2				
$4 \rightarrow 1$		4_1							
$5 \rightarrow 2$		5_1		8_1		9_1		10_1	
$6 \rightarrow 3$		6_1							
$7 \rightarrow 3$				7_1					
$8 \rightarrow 5$	8_2								
$9 \rightarrow 5$			9_2						
$10 \rightarrow 5$					10_2				

Figure 7 shows the primary schedule with the bonus slot assignment for the bonus request of node 6 asking for an additional slot. These results are given by the GLPK linear program (in black cells) and by

the AMSA algorithm (in grey cells), respectively. In this scenario, we observe that the schedule length is unchanged, despite the bonus path assignment from node 6. This is due to the available spatial reuse.

Figure 7. The bonus slot assignment for node 6.

Slot / $tx \rightarrow rx$	1	2	3	4	5	6	7	8	9
$2 \rightarrow 1$	2_1		5_1		8_1		9_1		10_1
$3 \rightarrow 1$	3_2		6_2		7_2		$6'_2$	$6'_1$	
$4 \rightarrow 1$		4_1							
$5 \rightarrow 2$		5_1		8_1		9_1		10_1	
$6 \rightarrow 3$		6_1				$6'_1$	$6'_1$		
$7 \rightarrow 3$				7_1					
$8 \rightarrow 5$	8_2								
$9 \rightarrow 5$			9_2						
$10 \rightarrow 5$					10_2				

The new optimal slot assignment given by MODESA that takes into account the additional demand of node 6 is presented in Figure 8. Changes from the primary schedule are represented in grey. Since this schedule is optimal and has the same length as the schedule modified by AMSA, the solution we propose, it follows that there exist scenarios where AMSA provides the optimal solution.

Figure 8. The new slot assignment taking into account the additional demand of node 6.

Slot / $tx \rightarrow rx$	1	2	3	4	5	6	7	8	9
$2 \rightarrow 1$	2_1		5_1		8_1		9_1		10_1
$3 \rightarrow 1$	3_2		6_2		$6'_2$		7_2		
$4 \rightarrow 1$		4_1							
$5 \rightarrow 2$		5_1		8_1		9_1		10_1	
$6 \rightarrow 3$		6_1		$6'_1$					
$7 \rightarrow 3$						7_1			
$8 \rightarrow 5$	8_2								
$9 \rightarrow 5$			9_2						
$10 \rightarrow 5$					10_2				

Figure 9 describes the bonus slot assignment for the bonus request of node 9 asking for an additional slot. In this case, the bonus slot assignment overflows from the primary schedule, taking slots from the inactivity period (slots 10 and 11).

Figure 9. The bonus slot assignment for node 9.

Slot / $tx \rightarrow rx$	1	2	3	4	5	6	7	8	9	10	11
$2 \rightarrow 1$	2_1		5_1		8_1		9_1		10_1		$9'_1$ $9'_1$
$3 \rightarrow 1$	3_2		6_2		7_2						
$4 \rightarrow 1$		4_1									
$5 \rightarrow 2$		5_1		8_1		9_1		10_1		$9'_1$ $9'_2$	
$6 \rightarrow 3$		6_1									
$7 \rightarrow 3$				7_1							
$8 \rightarrow 5$	8_2										
$9 \rightarrow 5$			9_2				$9'_2$		$9'_2$		
$10 \rightarrow 5$					10_2						

With the additional demand of node 9, MODESA obtains the assignment presented in Figure 10.

In both examples, AMSA obtains the same schedule length as the linear program; so its bonus slot assignment is optimal for these scenarios. Moreover, the bonus slot assignment has the same number of slots as with a complete recomputation of the optimal assignment obtained with the GLPK model presented in Section [2], taking into account the initial and additional transmission needs. We can also deduce from these examples that the length of the new schedule depends not only on the additional bonus slots requested, but also on the depth of requesting nodes in the convergecast tree.

Figure 10. The new schedule taking into account the additional demand of node 9.

Slot / $tx \rightarrow rx$	1	2	3	4	5	6	7	8	9	10	11
$2 \rightarrow 1$	2_1		5_1		9_1		8_1		$9'_1$		10_1
$3 \rightarrow 1$	3_2		6_2		7_2						
$4 \rightarrow 1$		4_1									
$5 \rightarrow 2$		5_1		9_1		8_1		$9'_1$		10_1	
$6 \rightarrow 3$		6_1									
$7 \rightarrow 3$				7_1							
$8 \rightarrow 5$			8_2								
$9 \rightarrow 5$	9_2				$9'_2$						
$10 \rightarrow 5$							10_2				

5. Performance Evaluation

We implemented MODESA and AMSA on a simulation tool based on GNU Octave [14] and evaluated the latency (schedule length), energy (radio state switches), bandwidth (slot reuse ratio) and throughput (sink interfaces occupation ratio) in various topologies of WSNs. The number of nodes varies from 10 to 100. We use the Galton-Watson process as a branching stochastic process to generate random trees: each node gives birth to a random number of children independently of the others and according to the same distribution. The number of children is at most equal to three. In addition, we assume that the only existing links are those in the tree. We suppose that all the nodes except the sink have a single radio interface, and we vary the number of sink radio interfaces from one to three. The maximum number of channels available at each node is equal to two.

We use a comparison with MODESA as a baseline of this study. Indeed, in [2], we prove that MODESA is optimal for line, multiline and balanced trees. MODESA is the algorithm used to find the primary schedule. We present two different cases of application of our adaptive AMSA algorithm: retransmissions and changing needs of the application. Furthermore, we distinguish two subcases: homogeneous and heterogeneous initial demands of nodes. It is worth noting that in a WSN, we can have different types of wireless sensors running at possibly different sampling rates. This results in non-homogeneous initial demand of nodes. The adaptive scheduling algorithm should be able to handle this heterogeneity.

In addition, in order to show how the convergecast structure can impact the schedule length, we differentiate the two types of configurations, T_s and T_n (see definitions in paragraph 3.3.4). This aims at deriving some intrinsic properties of each of them.

In the following, each result is the average of 20 simulation runs for small topologies ($\leq$ 30 nodes) and 100 runs for large topologies.

5.1. Retransmission Oriented Experiments

WSNs are typically deployed in industrial environments, where they are potentially exposed to external interference, making packet losses inevitable, even in a multichannel context. To enhance data reliability and response capability, retransmissions are crucial. In this section, we investigate the behavior of our proposed algorithm AMSA when nodes need to retransmit packets. We evaluate the number of extra slots allocated when each node requests 5%, 10%, 20% of its regular slots as bonus slots. These percentages reflect packet loss probability (low, moderate, high).

5.1.1. Homogeneous Initial Demands

First, we consider the two configurations of 20 nodes illustrated in Figure 11a,b. The first one corresponds to a T_s configuration, whereas the second belongs to the T_n configuration.

Figure 11. Two configurations of 20 nodes. (**a**) T_s configuration; (**b**) T_n configuration.

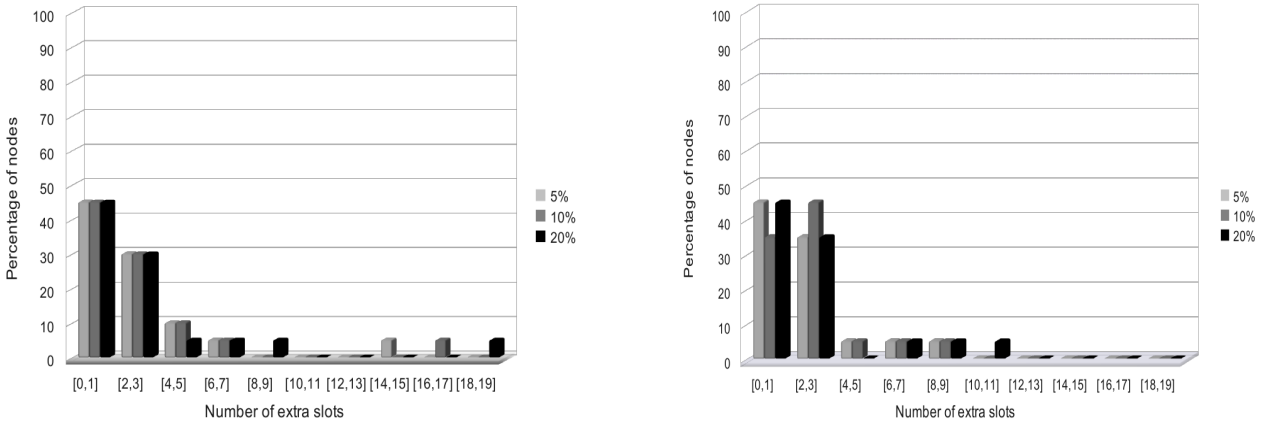
(**a**) (**b**)

As mentioned in previous sections, when all nodes initially have one packet to transmit, the number of regular slots for a node is the number of its descendants plus one slot for itself. The objective is to define which configuration is likely to request more extra slots when the retransmission rate of nodes increases.

As illustrated in Figure 12a, when the rate of retransmission increases, some nodes in T_s configuration need, respectively, $[14, 15]$, $[16, 17]$ and $[18, 19]$ for 5%, 10% and 20%. Whereas, we notice that for the T_n configuration, the maximum demand for extra slots is less than 12 slots, even when nodes require 20% of their regular slots as bonus slots. Furthermore, we can see that for the T_s configuration, node 4, which is the root of the most populated subtree, is the most greedy in terms of extra slots. For example, for a 5% demand of bonus slots, node 4 requires 15 extra slots in addition to the regular ones. Nevertheless, in the T_n configuration, the number of slots required in addition to the regular ones is balanced between all the nodes. As illustrated in Figure 12b, with a demand increasing up to 10% of the regular slots, 80% of nodes require only zero, one or two or three extra slots. Hence, the impact of bonus requests on the schedule length in the case of retransmission is more drastic in T_s configurations than in T_n configurations. This implies higher worst case data gathering delays.

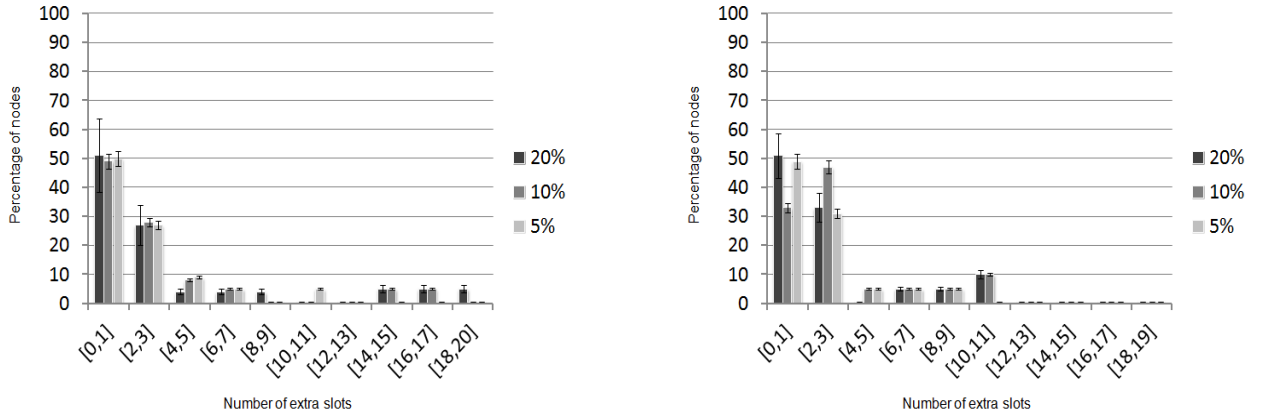
To further illustrate this point, we reproduce the same scenario on 25 T_s and 25 T_n configurations of 20 nodes. The previous results are confirmed, as depicted in Figure 13: T_s configurations are more greedy in terms of extra slots when the nodes need to retransmit.

Figure 12. Percentage of required extra slots. (**a**) in T_s configuration Figure 11a; (**b**) in T_n configuration Figure 11b.



(**a**) (**b**)

Figure 13. Percentage of required extra slots. (**a**) in T_s configuration; (**b**) in T_n configuration.

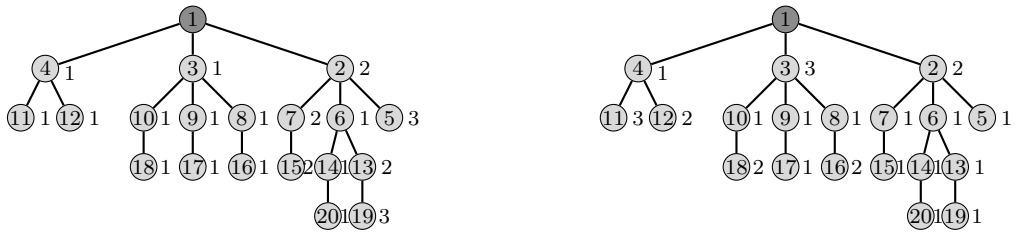


(**a**) (**b**)

5.1.2. Heterogeneous Initial Demands

We consider the topology represented in Figure 11b. Notice that this topology can switch from a T_s configuration to a T_n configuration and *vice-versa*, depending on the initial heterogeneous demands. We consider two scenarios of heterogeneous demands. These scenarios have the same global demand of 27, but the demands are distributed differently to get a T_s and a T_n configuration, as depicted, respectively, in Figure 14a,b. The optimal schedule length for the former is 32 slots and 27 for the latter. The two optimal schedules are reached by MODESA.

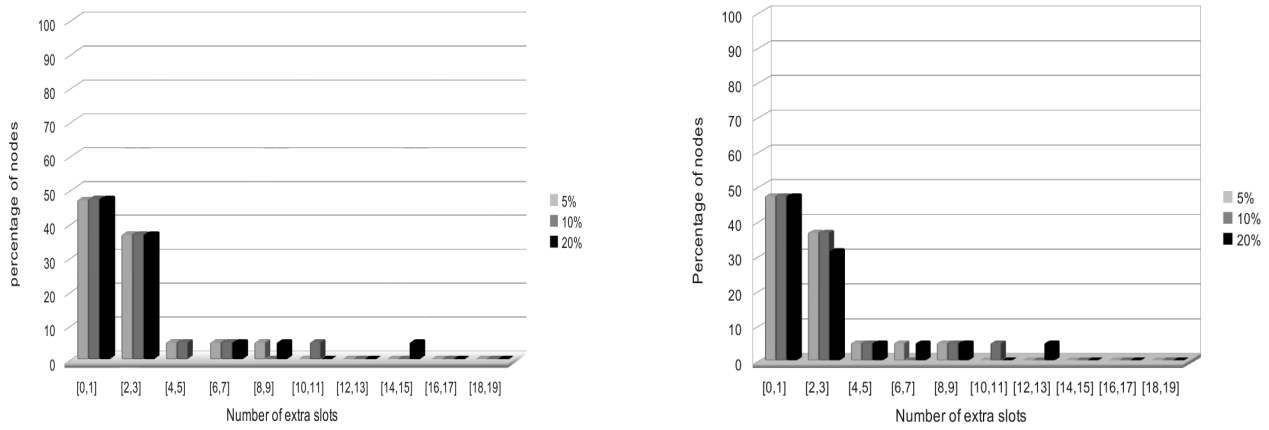
Figure 14. Two configurations for the same topology of 20 nodes. (**a**) T_s configuration; (**b**) T_n configuration.



(**a**) (**b**)

We conducted the same experiments as in Section 5.1.1, varying the retransmission rate of nodes: 5% (low), 10% (moderate) and 20% (high). The results are shown in Figure 15a,b.

Figure 15. Percentage of extra slots required. (**a**) in T_s configuration; (**b**) in T_n configuration.



(**a**) (**b**)

Overall, we notice the same conclusions as in the previous section: the T_s configuration needs more extra slots than the T_n configuration.

5.2. Temporary Change in the Application Needs-Oriented Experiments

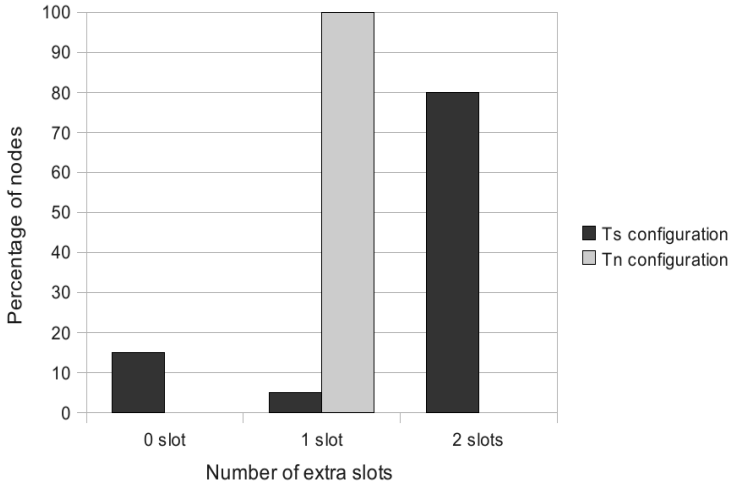
For energy constraints, a WSN operates on a low duty cycle. Nevertheless, the network can be subject to peaks in traffic. Indeed, an alarm or a physical phenomenon triggers sudden communication activity by nodes. The adaptive scheduling should enable low duty cycle WSNs to efficiently handle sudden traffic peaks. This section tackles the problem of changing application demands. In the following, we distinguish two cases: homogeneous and heterogeneous initial demands.

5.2.1. Homogeneous Initial Demands

In the first series of experiments, we consider two configurations of 20 nodes, a T_s configuration and a T_n configuration, as illustrated in Figure 11a,b. We first evaluate the impact on the schedule length of

a bonus request made by a node. The node that makes the bonus request is randomly chosen. Figure 16 shows the results obtained for each configuration. This experiment enables us to know the distribution of greedy nodes in terms of extra slots.

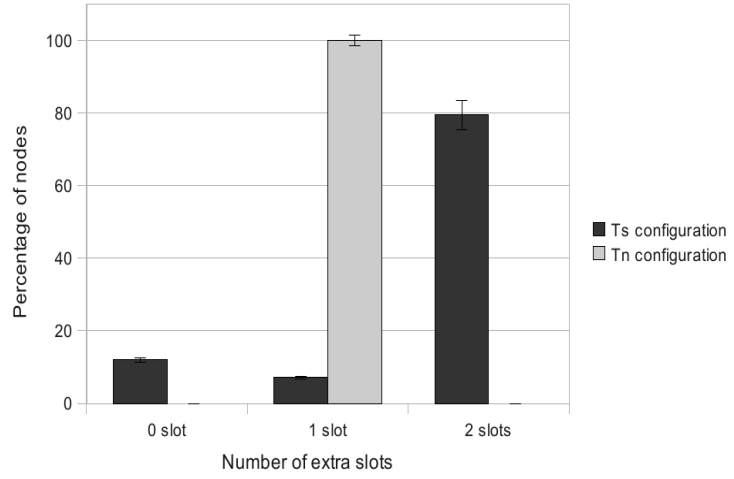
Figure 16. Distribution of nodes requiring extra slots.



The results show that for the T_s configuration, 80% of the nodes that retransmit need two extra slots, whereas 5% need only one extra slot and 15% require zero slots. For the T_s configuration, nodes that belong to the dominated subtree need two slots and, therefore, have a higher impact on the schedule length. For example, node 10 in the T_s configuration requires two extra slots, whereas node 9 does not require any extra slot. Moreover, in the T_n configuration, regardless of the node demanding the additional slot, the impact on the schedule length is the same: all nodes need only one extra slot to retransmit the failed packet. This can be explained by the fact that the optimal schedule length is imposed by the number of nodes and their additional demands for slots.

In order to generalize these results, we reproduced the previous experiment with 25 T_s and 25 T_n configurations of 20 nodes. As illustrated in Figure 17, we observe that all the nodes in T_n configurations have the same impact on the schedule length: they add one extra slot. For the T_s configurations, the nodes requiring the highest number of extra slots (two extra slots) belong to the most demanding subtree.

Figure 17. Impact on schedule length in case of nodes requiring extra slots.



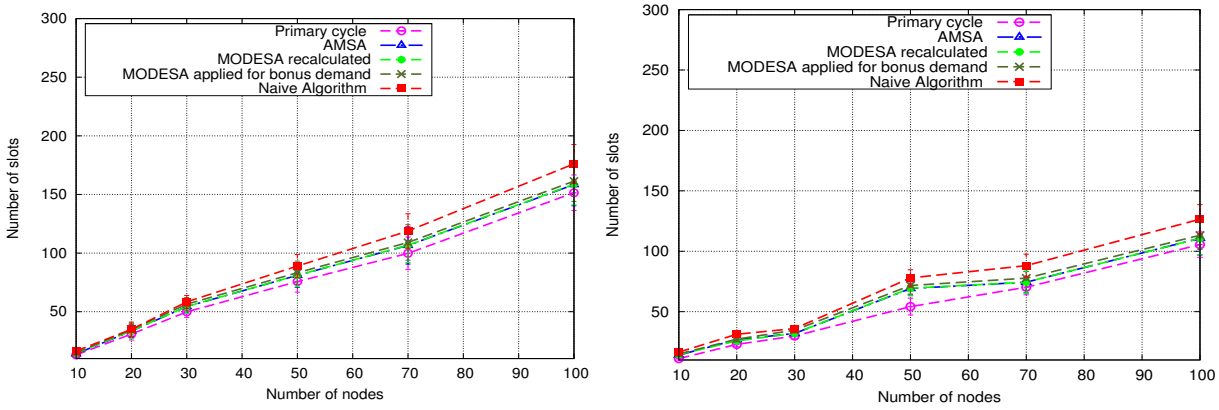
In the second series of experiments, we compare the performances of AMSA in larger networks with three other approaches:

- MODESA recalculated: MODESA is run taking into account both initial and bonus demands.
- MODESA applied for bonus demand: MODESA is run in the primary schedule and also in the complementary schedule to schedule bonus demands.
- Naive Algorithm: assigns each extra slot exclusively to one transmitter.

The primary schedule is given by MODESA, and we vary the percentage of nodes requiring an extra slot (5%, 10% and 20%).

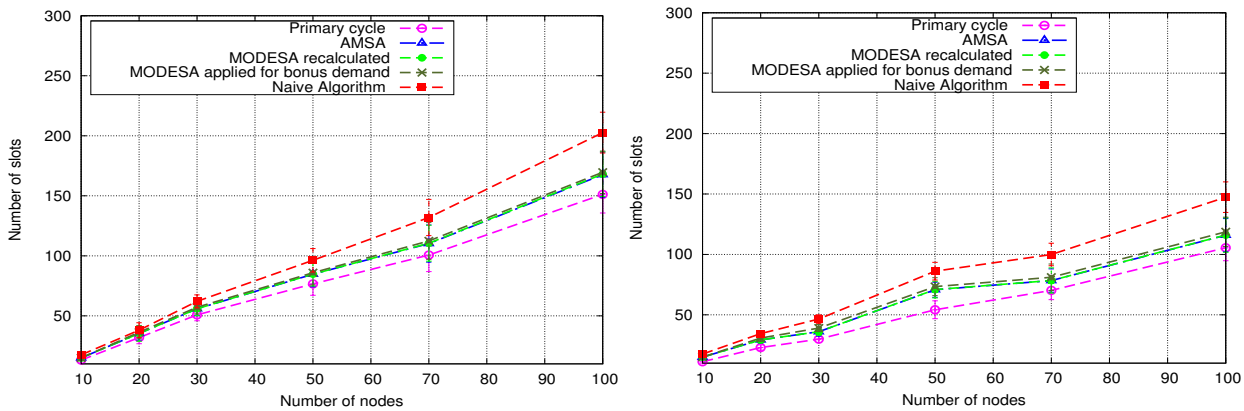
The results are depicted in Figures 18–20 respectively.

Figure 18. Average number of extra slots required with 5% of nodes having a bonus request. (**a**) in T_s configuration; (**b**) in T_n configurations.



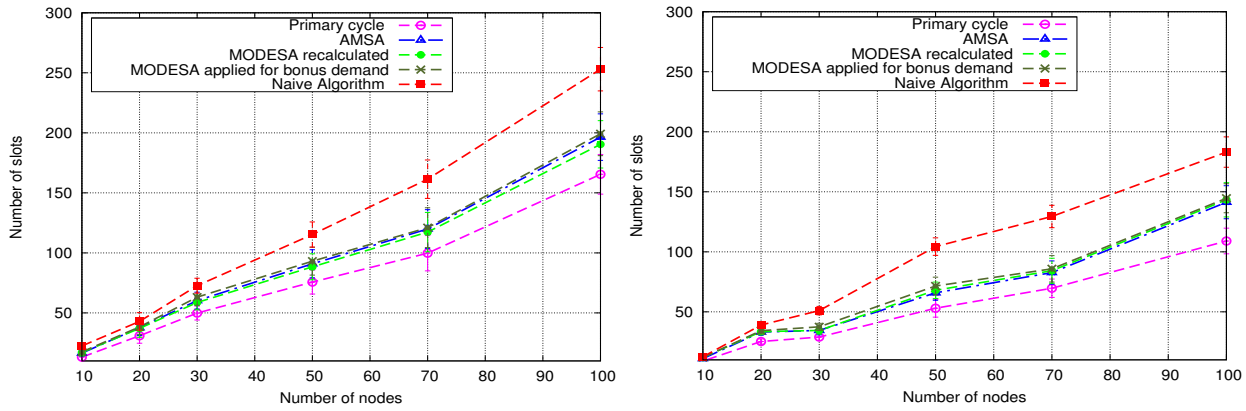
(**a**) (**b**)

Figure 19. Average number of extra slots required with 10% of nodes having a bonus request. (**a**) in T_s configuration; (**b**) in T_n configurations.



(**a**) (**b**)

Figure 20. Average number of extra slots required with 20% of nodes having a bonus request. (**a**) in T_s configuration; (**b**) in T_n configurations.



It appears that in both types of configurations (T_s and T_n), the number of extra slots increases with the number of nodes in the topology and the number of nodes requesting bonus slots. This is intuitive, because as the network gets larger, additional packets have to be forwarded to the sink. Note that T_s configurations are more greedy in terms of required extra slots than T_n configurations for the same percentage of nodes having bonus slots.

When 20% of nodes request a bonus slot, MODESA recalculated dominates AMSA more significantly (see T_s configurations in Figure 20a). AMSA fills the slots in the primary schedule with possible additional transmissions and, then, schedules the remaining ones in extra slots. Nevertheless, MODESA recalculated makes more spatial and frequency reuse by knowing all the demands of nodes at the outset. For example, in T_n configurations, as depicted in Figure 20a, AMSA needs 8 extra slots more than MODESA when 20% of 100 nodes require a bonus slot.

We also notice that MODESA recalculated and AMSA provide very close performances when 5% and 10% of nodes have a bonus request. This illustrates the merit of AMSA that provides the same performance level as MODESA recalculated, but with less complexity.

Figure 20 also shows that MODESA applied for bonus demands requires more slots to schedule additional demand than AMSA and MODESA recalculated. This can be explained by the fact that MODESA applied for bonus demands computes the complementary schedule separately from the primary one. Only bonus demands are scheduled in this complementary schedule. Thus, its schedule is not as optimized as MODESA recalculated.

This second series of experiments allows us to make some conclusions regarding the efficiency of AMSA that takes advantage of spatial reuse and multichannel to assign bonus slots.

In the third series of experiments, we varied the number of nodes from 10 to 100 and assumed that 20% of nodes request a bonus slot. We compared the AMSA assignment with the three other approaches (MODESA recalculated, MODESA applied for bonus demand, primary schedule) in terms of energy (evaluation of radio state switches), bandwidth (evaluation of slot reuse) and throughput (evaluation of sink interfaces occupation ratio). All the results are presented in Figures 21–23.

Figure 21. Slot reuse ratio with 20% of nodes having a bonus request. (**a**) in T_s configuration; (**b**) in T_n configurations.

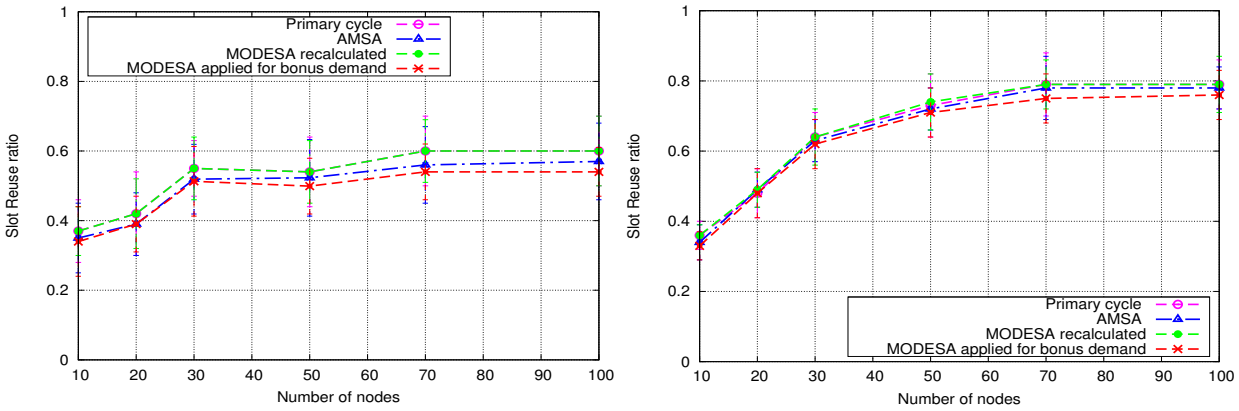
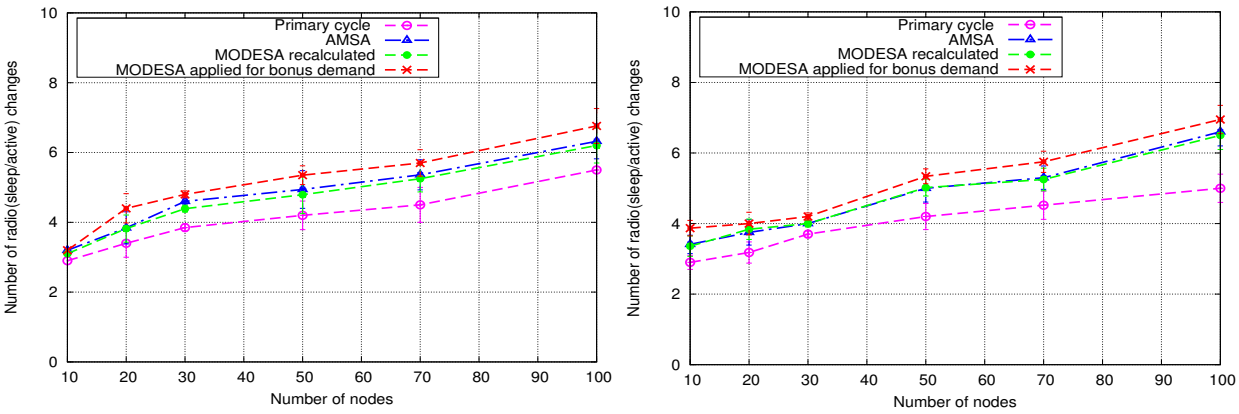


Figure 22. Average number of state switches per node with 20% of nodes having a bonus request. (**a**) in T_s configuration; (**b**) in T_n configuration.

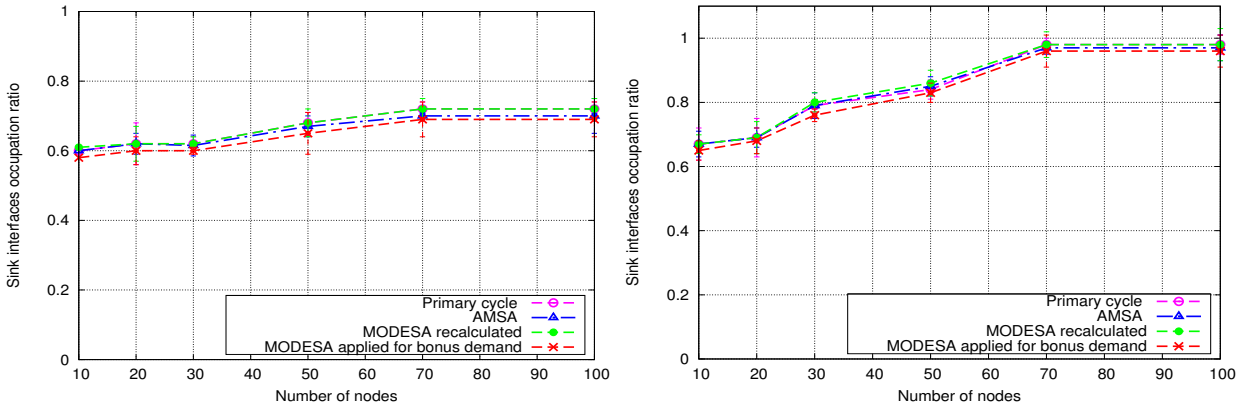


Overall, the performance of MODESA recalculated and AMSA are very close. More specifically, in T_n configurations, MODESA recalculated has slightly better performance than AMSA. For example, in Figure 21b, the two approaches achieve the same slot reuse ratio. However, in T_s configurations, MODESA recalculated slightly outperforms AMSA, as shown in Figure 21a, achieving higher slot reuse ratio. MODESA applied for bonus demands achieves a lower slot reuse than the other two approaches. Undoubtedly, the slot reuse ratio significantly reduces the end-to-end latency without any adverse effect on energy efficiency.

Similarly, as depicted in Figure 22a,b, the number of radio state switches of MODESA recalculated matches that of AMSA. This number of switches that takes into account both initial and bonus demands is not far from that achieved in the primary schedule. Nevertheless, MODESA applied for bonus demands makes more radio state switches. Thus, both AMSA and MODESA decrease energy consumption, due to state switches.

We also explored the throughput or the sink radio interface occupation ratio. This is defined as the number of times the sink receives packets divided by the total number of slots. Figure 23 illustrates that MODESA recalculated guarantees a slightly higher throughput than AMSA. Another remarkable phenomenon is that T_n configurations outperform T_s configurations in terms of throughput. Indeed, in T_s configurations, the sink interface is kept busy only one slot in two once the dominated subtrees have completed their transmissions.

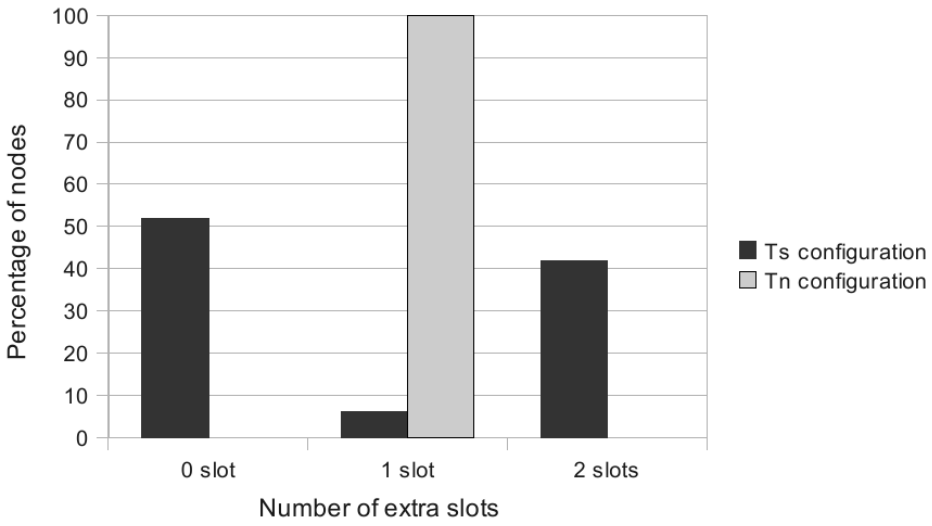
Figure 23. Sink interfaces occupation ratio with 20% of nodes having a bonus request. (**a**) in T_s configuration; (**b**) in T_n configuration.



5.2.2. Heterogeneous Initial Demands

We again consider the two types of convergecast configurations: T_s and T_n, as depicted, respectively, in Figure 14a,b. In these configurations, the initial demands of nodes are heterogeneous. The sink has a single interface, and nodes can transmit on three channels. As in Subsection 5.1.1, we first evaluate the impact on the schedule length of a bonus request made by a node. The node that makes the bonus request is randomly chosen. Figure 24 shows the results obtained in each configuration.

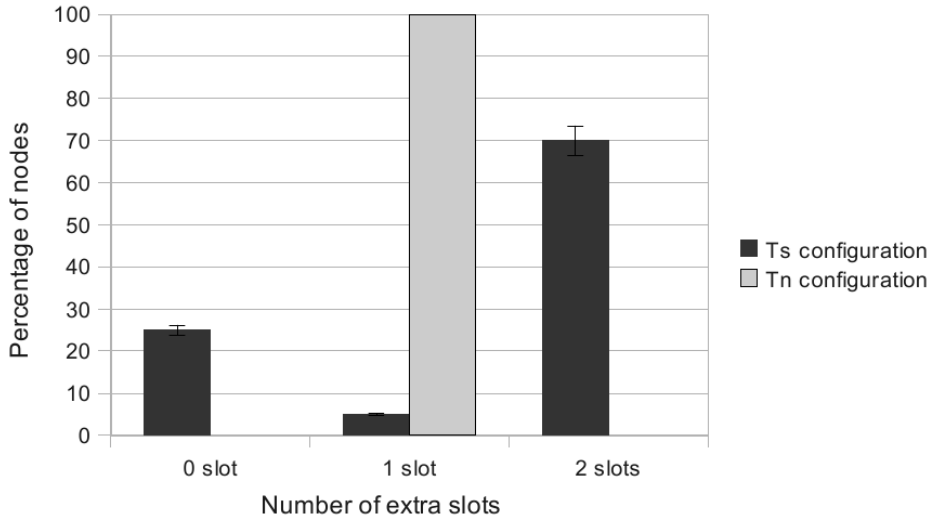
Figure 24. Distribution of nodes requiring extra slots.



We find the same conclusions as in the case of homogeneous initial demand: for the T_s configuration, 43% of the nodes that retransmit need two extra slots, while 5% need only one extra slot and 52% require zero slots. Nodes that need two extra slots belong to the dominated subtree. Moreover, for the T_n configuration, we come to the same conclusion: regardless of the node demanding the additional slot, the impact on frame length is the same: all nodes need one extra slot to retransmit the failed packet. All the nodes have the same impact on schedule length.

In order to generalize these results, we reproduced the previous experiment with 25 T_s and 25 T_n configurations of 20 nodes. Initially, nodes have heterogeneous demands between one and five. As shown in Figure 25, we observe that all the nodes in T_n configurations have the same impact on the schedule length: they add one extra slot. For the T_s configurations, the nodes requiring the highest number of extra slots (two extra slots in our simulation scenario) belong to the most demanding subtree.

Figure 25. Impact of demands for bonus slots on the schedule length.



After that, we considered larger topologies with 100 nodes. We varied the number of sink radio interfaces, as well as the number of channels. As mentioned previously, we compared the three approaches (AMSA, MODESA recalculated and MODESA applied for bonus slots) in terms of the number of slots. In this group of experiments, all the nodes initially have heterogeneous demands, uniformly distributed between one and five. In addition, 20% of the nodes request a bonus slot.

The simulation results show, as depicted in Figure 26, that T_s configurations are more greedy in terms of slots needed for raw data convergecast than T_n configurations. The T_n configurations are more balanced in terms of nodes demands per subtree rooted at a sink child. Hence, their schedule tends to be shorter. Consequently, we can make some recommendation regarding the routing tree built: it should balance the nodes demands on subtrees rooted at a sink child. These results generalize the work of Incel *et al*. [5] when nodes have heterogeneous demands.

We have also compared the number of slots obtained by AMSA and MODESA recalculated with the optimal number of slots. For a sink with one radio interface and three channels, MODESA recalculated (respectively AMSA) is optimal in 50% of the T_s configurations (respectively, 43.18%) and in 87% of the T_n configurations (respectively, 82%), as illustrated in Figure 27.

Figure 26. Number of slots required for convergecast in (**a**) T_S configurations (**b**) T_N configurations.

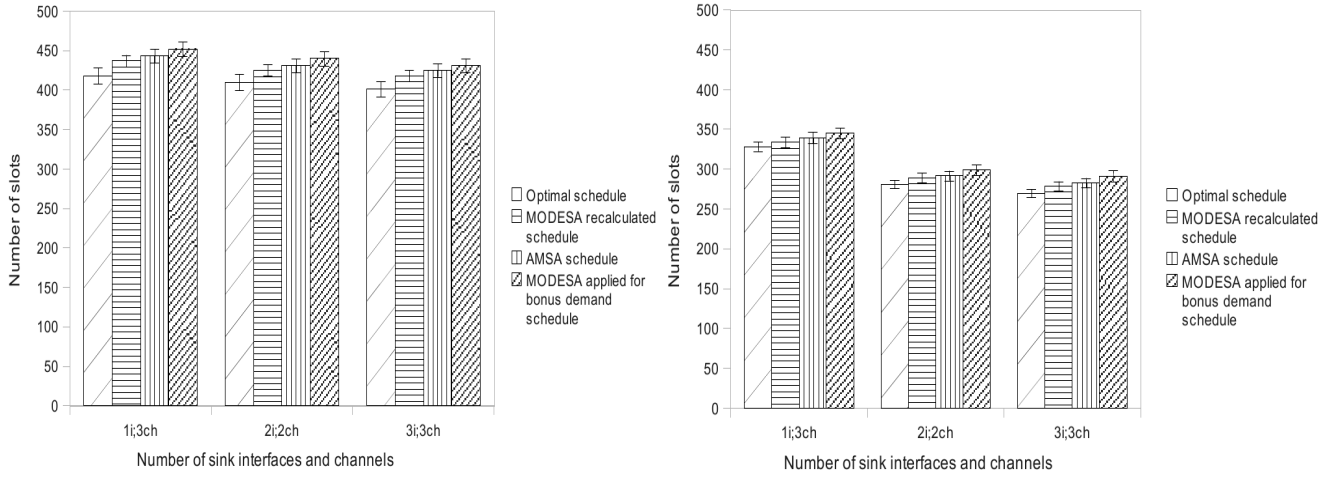
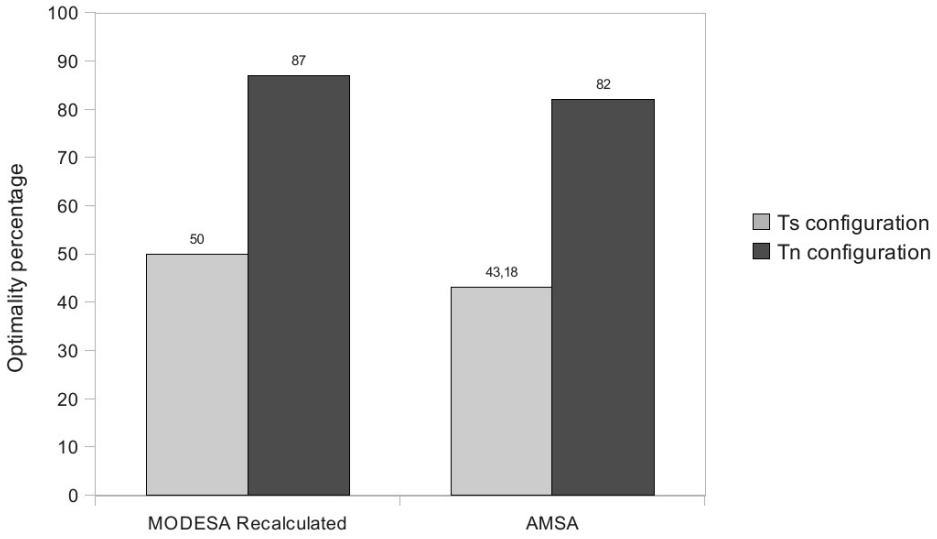


Figure 27. Optimality of MODESA recalculated and AMSA in T_s and T_n configurations.

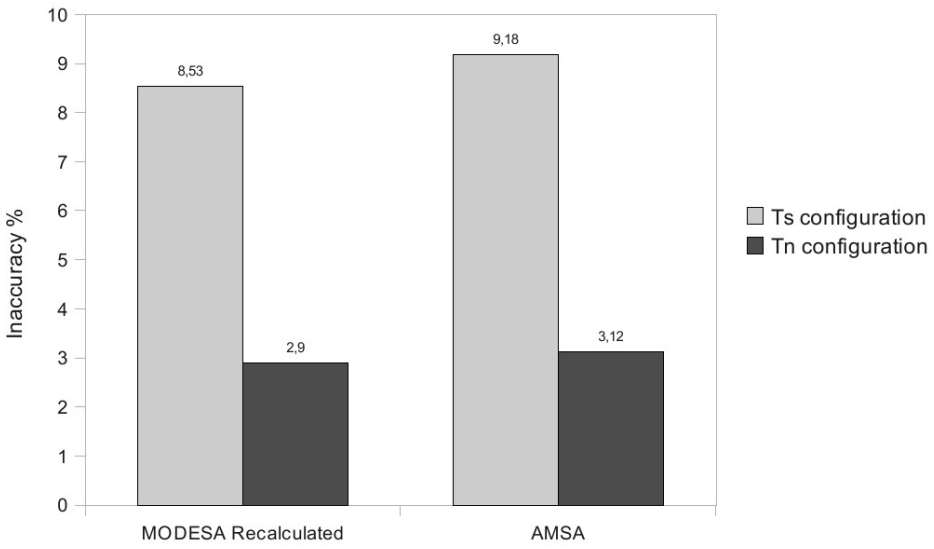


In addition, we evaluate the distance of AMSA and MODESA recalculated to the optimal schedule in configurations where they are not optimal. This distance is called inaccuracy and is computed as: $\frac{number\ of\ slots\ needed - optimal\ number\ of\ slots}{optimal\ number\ of\ slots}$. As depicted in Figure 28, the average inaccuracy of AMSA is 9.2% for the T_s configurations and 3.2% for the T_n configurations, demonstrating the very good behavior of AMSA. The average inaccuracy in both T_s and T_n configurations is below 10%.

These experiments demonstrate that AMSA and MODESA recalculated have comparable performances in terms of extra slots. This means that AMSA performs well in filling existing slots with possible transmissions without the need to recompute the schedule again unlike MODESA recalculated. The light weight of AMSA considerably enhances its adaptivity in networks with different traffic loads and additional demands. In addition, using multi-radio interfaces for the sink with multichannels

decreases the frame length. Hence, the activity period of nodes is decreased, allowing them to sleep more to save their limited energy.

Figure 28. Inaccuracy of MODESA recalculated and AMSA in T_s and T_n configurations.



6. Conclusion

In this paper, we studied the raw data convergecast in multichannel WSNs, where nodes have heterogeneous initial demands, and the sink is equipped with multiple radio interfaces. Assuming that the links in the convergecast tree are the only ones existing in the network topology, we computed the lower bound of the schedule length for data gathering. We have shown that T_n configurations require less slots for convergecast than T_s configurations. This is due to the fact that nodes demands are more balanced between the subtrees rooted at a sink child.

In addition, unexpected traffic loads or retransmissions can hamper the initial time slot allocation for data convergecast in a WSN. In order to address this issue, we tackled the problem of adaptive slot assignment in multichannel WSNs. We have formalized and solved this problem with linear programming. We have shown that in T_s configurations, where the most demanding subtree imposes the length of the primary schedule, the request for bonus slots made by nodes belonging to this subtree has a great impact on the number of extra slots required, unlike in T_n configurations, where the number of extra slots is much more balanced between all the nodes. We then proposed the AMSA algorithm, which unifies the management of additional slots, whatever their origin: changes in the application demands or in the medium access demands due to retransmissions. AMSA assigns bonus slots in an optimized primary schedule provided by MODESA, adding extra slots only when the slots allocated in the primary schedule are unable to meet the dynamic demands. It takes advantage of spatial reuse to reduce the number of bonus slots added to the regular ones. We evaluated the performances of AMSA in various multichannel topologies and how close it comes to an optimized schedule. The adaptivity of AMSA to both changes in the application or in the medium access demands ensures efficient convergecast in WSNs, especially in the case of low (5%) and moderate (10%) additional transmissions. When 20%

of nodes require additional slots, AMSA is optimal in 43% of the T_s configurations and 82% in the T_n configurations. In all cases, AMSA's schedule length is less than 10% higher than the optimal one.

Acknowledgments

We would like to acknowledge our reviewers and Richard James, who helped us to improve this manuscript.

Conflict of Interest

The authors declare no conflict of interest.

References

1. Ghosh, A.; Incel, O.D.; Kumar, A.; Krishnamachari, B. Multi-Channel Scheduling Algorithms for Fast Aggregated Convergecast in Sensor Networks. In Proceedings of the 6th IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS), Macau, China, 12–15 October 2009.
2. Soua, R.; Minet, P.; Livolant, E. MODESA: An Optimized Multichannel Slot Assignment for Raw Data Convergecast in Wireless Sensor Networks. In Proceedings of the 31 International Performance Computing and Communications Conference (IPCCC), Austin, TX, USA, 1–3 December 2012.
3. Zhang, H.; Soldati, P.; Johansson, M. Optimal Link Scheduling and Channel Assignment for Convergecast in Linear WirelessHART Networks. In Proceedings of the International Conference on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT'09), Seoul, Korea, 23–25 June 2009.
4. Wu, Y.; Stankovic, J.; He, T.; Lin, S. Realistic and Efficient Multi-channel Communications in Wireless Sensor Networks. In Proceedings of the International Conference on Computer Communications, INFOCOM'08, Phoenix, AZ, USA, 15–17 April 2008.
5. Incel, O.D.; Gosh, A.; Krishnamachari, B.; Chintalapudi, K. Fast data collection in tree-based wireless sensor networks. *IEEE Trans. Mob. Comput.* **2012**, *1*, 86–99.
6. Cui, M.; Yin, J.; Nie, J.; Zhang, J.; Cao, Y. An Evolutionary Dynamic Slot Assignment Based on P-TDMA for Mobile Adhoc Network. In Proceedings of the 11th IEEE Singapore International Conference on Communication Systems (ICCS), Guangzhou, China, 19–21 November 2008.
7. Gexin, P.; Shengli, X.; Caiyun, C. A collision-avoid dynamic slots assignment algorithm based on fixed TDMA. *China Inf. Secur.* **2005**, *11*, 115–120.
8. Gobriel, S.; Mosse, D.; Cleric, R. TDMA-ASAP: Sensor Network TDMA Scheduling with Adaptive Slot-Stealing and Parallelism. In Proceedings of the 29th IEEE International Conference on Distributed Computing Systems (ICDCS), Montreal, QC, Canada, 22–26 June 2009.
9. Yackovich, J.; Mosse, D.; Rowe, A.; Rajkumar, R. Making WSN TDMA Practical: Stealing Slots Up and Down the Tree. In Proceedings of the 17th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications(RTCSA), Toyama, Japan, 28–31 August 2011.

10. Kanzaki, A.; Hara, T.; Nishio, S. On a TDMA Slot Assignment Considering the Amount of Traffic in Wireless Sensor Networks. In Proceedings of the International Conference on Advanced Information Networking and Applications Workshops (WAINA), Bradford, UK, 26–29 May 2009.
11. Tselishchev, Y.; Libman, L.; Boulis, A. Energy-efficient Retransmission Strategies Under Variable TDMA Scheduling in Body Area Networks. In Proceedings of the 36th IEEE Conference on Local Computer Networks (LCN), Bonn, Germany, 4–7 October 2011.
12. Papadimitriou, C.H. The complexity of the capacitated tree problem. *Networks* **1978**, *8*, 217–230.
13. GLPK (GNU Linear Programming Kit). Available online: http://www.gnu.org/software/glpk/ (accessed on 10 May 2013).
14. GNU Octave. Available online: http://www.gnu.org/software/octave/ (accessed on 15 January 2013).