Wireless Netw (2015) 21:2395–2411
DOI 10.1007/s11276-015-0925-0

Interference-free scheduling with minimum latency in cluster-based wireless sensor networks

Alfredo Navarra · Cristina M. Pinotti · Mario Di Francesco · Sajal K. Das

Published online: 27 February 2015

Abstract This article addresses wireless sensor networks (WSN) whose nodes are organized in groups (i.e., clusters) and follow a duty-cycle. Each cluster is locally managed by a cluster head which employs a medium access control protocol to avoid interferences in all intra-cluster communications. Nevertheless, inter-cluster interferences can still occur. To this regard, we consider two clusters as interfering if their hop distance is at most t, with $t \geq 2$, in the cluster connectivity graph. Under such a model, we target convergecast data collection of aggregated traffic and show that finding a minimum-latency interference-free convergecast schedule up to distance t is NP-hard for cluster-based WSNs with arbitrary topologies. Due to the hardness result, we restrict our attention to cluster-tree WSNs which can model ad hoc WSN deployments. We optimally solve the problem on trees for $t = 2$ by minimizing both the latency and the schedule length. Then, for any $t \geq 2$, we propose a minimum-latency interference-free algorithm that obtains a slot assignment with guaranteed approximated latency in $O(nt)$ time, where n is the number of clusters in the WSN. We also discuss a distributed implementation of such a scheduling algorithm that results in an exchange of $O(nt)$ messages. Moreover, we consider a minimum-latency data collection in complete trees of arbitrary degree as a special case. We finally validate our findings by a simulation study on synthetic tree topologies.

Keywords Wireless sensor networks · Inter-cluster communication · Interference-free scheduling · Minimum-latency aggregated convergecast · NP-hardness · Trees

A. Navarra · C. M. Pinotti (✉)
Department of Mathematics and Computer Science, University of Perugia, Perugia, Italy
e-mail: cristina.pinotti@unipg.it

A. Navarra
e-mail: alfredo.navarra@unipg.it

M. Di Francesco
Department of Computer Science, Aalto University, Espoo, Finland
e-mail: mario.di.francesco@aalto.fi

S. K. Das
Department of Computer Science, Missouri University of Science and Technology, Rolla, MO, USA
e-mail: sdas@mst.edu

1 Introduction

In a typical wireless sensor network (WSN), sensor nodes are deployed over a sensing area to monitor a phenomenon of interest. Sensor nodes self-organize into a multi-hop network to form a wireless communication infrastructure. A special node, called *sink*, communicates with sensor nodes and makes the collected data available to end users through the Internet. In detail, the sink is the source and the destination of the two main representative communication patterns in WSNs, i.e., *broadcast* and *convergecast*, respectively. Several application scenarios, including environmental monitoring as well as industrial automation and control, have long network lifetime and low latency as their major requirements for data collection [1]. Hence, to efficiently support communications in the network, sensor nodes are usually organized into a hierarchical structure. For instance, nodes can be grouped into *clusters*. One node within each cluster is designated as *cluster head* to represent the rest of the sensors and manage their communications. A cluster-based architecture is suitable to reduce the network management overhead. Furthermore, cluster heads are suitable to perform in-network processing and data aggregation [2, 3]. Clusters are also useful to conserve energy, which is one of the major challenges in WSNs [4]. To this end, cluster heads can define a *duty-cycle*, i.e., a periodic pattern of intervals in which nodes are either active or inactive. During the active periods, all nodes in a cluster interact through some channel access mechanism specified by the cluster head. During the inactive periods, instead, nodes sleep to save energy. This approach is taken, for instance, in WSNs based on the ZigBee specifications [5–8]. Even though the duty-cycle can be locally defined by individual cluster heads, some coordination in the network is still required because communications occurring at the same time and involving different clusters may result in inter-cluster interference. Moreover, the actual choice of the active and inactive periods affects the latency of network communications, in terms of the time elapsed between a message transmission at the sender and the related reception at the receiver [9, 10]. Hence, finding a scheduling that minimizes the latency in the routing process also conserves energy.

This article addresses the problem of interference-free scheduling of communications in cluster-based WSNs. Our goal is to achieve low-latency communication despite the presence of a duty-cycle and of inter-cluster interference. In order to formulate the problem, we model the WSN as a graph whose vertices are the cluster heads and whose edges model symmetric communication links between such cluster heads. We also relate interferences to the number of hops between the nodes in such a graph. Specifically, we consider two cluster heads as interfering if their hop distance is at most t. In contrast with most of existing works, we address the general scenario where t can be greater than 2, as using $t = 2$ may not be enough for an actual interference-free slot assignment. For instance, two nodes belonging to two different clusters (i.e., not adjacent to each other) may be physically close enough that they actually interfere each other. Our approach guarantees that two cluster heads cannot receive at the same time if they are at a reciprocal distance less than or equal to t. Clearly, avoiding interferences only on the basis of the hop distance t may not solve all possible conflicts. In fact, distance-based interference models do not capture all physical layer effects nor the additive nature of interference. A transmitter may not interfere because it is far away; however a group of such distant transmitters may collectively cause substantial interference. A more accurate characterization is enabled by the so called *signal-to-interference and noise ratio* (SINR) model (see, e.g., [11, 12] and references therein). Given a set of stations that are simultaneously transmitting in the plane, the SINR model allows to identify a reception zone for each station, consisting of the points where its transmission is received correctly. Although the SINR model is considered more realistic, it makes very difficult to handle algorithmic issues. The interest in our proposed model based on the hop distance t comes from its simplicity and from the fact that increasing on t guarantees satisfactory solutions in terms of interference avoidance.

In the following, we focus on the problem of scheduling communications for convergecast data collection in cluster-based WSNs with aggregated traffic. Here aggregated convergecast means that all messages collected by the cluster head at each hop during a certain time frame are reduced to a single message. In such a scenario, we formally define the t-interference-free Minimum-latency Convergecast (in short, the $t-\mathrm{MC}$) problem. Given a cluster connectivity graph, the $t-\mathrm{MC}$ problem is to find an interference-free schedule between cluster heads (nodes) that are $t \geq 2$ hops away, such that the convergecast latency is minimized. We show that the $t-\mathrm{MC}$ problem is NP-hard for cluster-based WSNs with arbitrary topologies. We then focus on tree topologies which are representative of ad hoc WSN deployments. First, we propose a solution for the case $t = 2$ which computes a scheduling with minimum latency in $O(n \log \Delta)$ time, where Δ is the maximum number of children of any node in the tree. Then, for any interference distance $t \geq 2$, we propose the Scheduling-Level-by-Level algorithm that computes a scheduling with guaranteed approximated latency in $O(nt)$ time, where n is the number of clusters in the WSN. Such an algorithm provides a ρ-approximation factor for the minimum latency, with $\rho < \frac{1}{c}\left(\left\lfloor \frac{t}{2} \right\rfloor + 1\right)\frac{\overline{\gamma}}{\widehat{\gamma}}$, $c \in \left[\frac{1}{2}, 1 - \frac{\lfloor \frac{t}{2} \rfloor + 1}{h+1}\right]$, where: $\overline{\gamma}$ and $\widehat{\gamma}$ are, respectively, the largest and the smallest set of descendants at distance $\left\lfloor \frac{t}{2} \right\rfloor$ from any node $x \in \mathcal{T}$; h is the height of the tree. We also discuss a distributed implementation of such a scheduling algorithm that results in an exchange of $O(nt)$ messages. Moreover, we give a $\rho < 2(t+1)$ approximation for the minimum latency in complete trees of arbitrary degree as a special case. We finally validate the accuracy of the approximation factor given by our algorithm via a simulation study on synthetic trees. The obtained results show that our findings are accurate.

The rest of the article is organized as follows. Section 2 overviews the related work. Section 3 introduces the model, defines the $t-\mathrm{MC}$ problem in cluster-based WSNs with arbitrary topologies, and analyzes its computational complexity. Section 4 focuses on cluster-based WSNs with tree topology: it presents the optimal algorithm for $t = 2$ and the approximate interference-free scheduling algorithms for $t \geq 2$. Section 5 evaluates the performance of our approach by a simulation study. Finally, Sect. 6 concludes the article.

2 Related work

Approaches for scheduling communications in WSNs have been extensively studied in the existing literature [13–17], also with focus on energy conservation [4, 18].

Scheduling for data collection in tree-based WSNs was surveyed in [13]. Many solutions actually focused on minimizing the length of the schedule (i.e., the overall number of slots assigned). Among them, the work in [15] addressed scheduling communications in tree-based WSNs and proposed different heuristics for slot assignment. One of them specifically targets convergecast data collection and assigns slots to nodes according to their level in the tree. Even though this approach may appear similar to the one taken here, it is worth noting that the scheduling problem addressed in [15] is to find the schedule with minimum length (i.e., to find a graph coloring with the minimum number of colors). Instead, this article explicitly aims at minimizing the convergecast latency in addition to the length of the schedule. Although the length of the schedule is a lower bound for the convergecast latency, it is not sufficient to minimize the length of the schedule to achieve the minimum latency. Indeed, the problem we address is more challenging than plain coloring as it has to satisfy a network-wide latency constraint. As a consequence, most of the results obtained for coloring different kinds of graphs, as those in [19], cannot be applied to the scenario considered here.

The problem of minimum-latency scheduling for WSNs was also addressed in existing works. In [20], a solution for optimal scheduling in WSNs was first obtained by considering linear topologies, and then sub-optimally extended to multi-line, tree, and general networks. Such a solution obtains a closed formula for the latency, given the number of messages that each source node has to transmit to the sink. A similar approach was taken in [21, 22], again by extending solutions for multi-line trees to generic trees. Specifically, in [21] it was first shown that the problem of convergecast latency is NP-hard for arbitrary WSN topologies. In [22] a distributed solution was proposed for both tree-based and general WSNs, as opposed to the centralized solutions in [20]. The works mentioned above share two common assumptions: (1) raw-data convergecast is considered, where all data generated by sensor nodes are relayed to the sink; and (2) slots are assigned to individual sensor nodes. In this article, instead, we consider aggregated traffic that characterizes cluster-based networks in which data, reported from the individual sensors and aggregated at the cluster head, are then routed towards the sink. We remark that dealing with aggregated data rather than raw-data completely changes the problem. In fact, the scheduled activities do not require full paths for each data from each individual sensor toward the sink, but just full paths from each cluster head towards the sink. Indeed, in our model, depending on the external conditions, not all the sensors must always transfer data towards the sink. Clusters are then very useful for performing in-network processing and data aggregation when the individual nodes produce data not in a regular and continuous way [3]. The cluster head receives, in the assigned slot, the available data from the individual nodes in its cluster. Using a suitable medium access control (MAC) protocol, such as IEEE 802.15.4 standard [23], the cluster head solves the intra-cluster interferences faster and more efficiently than assigning a single time slot to each sensor in the cluster.

Moreover, in this article, we search for the minimum latency scheduling using as fewer slots as possible. That is, we allow that two clusters at hop distance greater than t reuse the same time slot to receive data from its own cluster members. For these reasons, the problem discussed here is radically different from that discussed in [20–22].

Minimum latency convergecast with data aggregation was considered in [24]. The problem was shown to be NP-hard even when the underlying graph is a grid. Furthermore, a $(\Delta - 1)$-approximation algorithm was proposed, where Δ is the maximum degree of the network. The minimum-latency aggregation schedule (MLAS) algorithm was proposed in [25] for synchronous multi-hop wireless networks. MLAS consists in finding the shortest schedule for data aggregation subject to interference constraints. MLAS was studied under the protocol interference model in which each node has a unit communication radius and an interference radius $\rho \geq 1$. Three algorithms were devised for $\rho = 1$ with an approximation ratio of $15R + \Delta - 4$, $2R + O(\log R) + \Delta$ and $1 + O(\frac{\log R}{\sqrt[3]{R}})R + \Delta$, respectively, where R is the transmission radius. Then, two of such algorithms were extended to the case of $\rho > 1$. A distributed algorithm with an approximation ratio of $16R + \Delta - 14$ was proposed in [26] under a similar radio model. For sensor networks deployed in a linear domain, that are represented as unit interval graphs, a 2-approximation algorithm has been recently proposed in [27]. That work also addresses the cases of k-regular unit interval graphs, grids and tori. All the solutions mentioned above assign slots to nodes instead of clusters, in contrast with the scenario considered here. Furthermore, the problem formulation relies for the connectivity graph on the unit disk graph model, which is not assumed in this work.

The minimum delay beacon scheduling (MDBS) problem was introduced in [28] to find an interference-free slot assignment which minimizes the convergecast latency in cluster-based ZigBee WSNs with aggregated traffic. The MDBS problem was shown to be NP-hard for networks in which interferences are described by an arbitrary graph. Optimal and sub-optimal solutions to the MDBS problem were then presented for linear, ring and tree topologies in [7]. In [8], the solutions proposed for the MDBS are extended to be effective for both periodic and event-based data reporting. Although our approach has some similarity with MDS problem, we provide a different formulation based on the interference distance t rather than relying on a separate conflict graph. Furthermore, our heuristic solution runs in $O(nt)$ time where n is the number of clusters in the WSN, as opposed to the $O(n^2)$ time required by [7, 28]. Finally, our approach is more general, since it is not specifically targeted to ZigBee WSNs.

In [29], an efficient link scheduling protocol has been proposed. It gives a constant factor approximation on the optimal throughput in delivering aggregated data from all the nodes of a tree to the root/sink. The main differences with respect to our approach reside in the usage of multiple frequency channels, and the assignment of time slots to the edges of the input tree.

Preliminary results concerning this work have been presented in [30]. This article extends our previous work by providing a more general definition of the problem, an optimal algorithm for the case $t = 2$, a distributed version of the scheduling algorithm for arbitrary t, as well as additional simulation results.

3 Minimum-latency convergecast

In this section, we first introduce the system model and the related assumptions, then we formally define the minimum-latency convergecast problem in cluster-based WSNs.

3.1 System model

We consider a multi-hop static WSN in which nodes report data to the sink, according to the convergecast communication pattern [13, 31]. We assume that: nodes are organized into clusters, each having a cluster head; messages are aggregated as they flow through the network [3]; and a duty-cycle mechanism is used for energy conservation [6]. Specifically, nodes within a cluster are active for a fixed period of time, referred to as *slot*, then they sleep to save energy. During a slot, nodes use a certain medium access control (MAC) protocol—for instance, the IEEE 802.15.4 standard [23]—to report collected data to their cluster head. As a consequence, we assume that intra-cluster communications do not result in interference, as channel access is regulated by the cluster head. As different clusters need to communicate to each other in order to relay messages towards the sink, one node in a cluster may have two different roles at different slots. On the one hand, a cluster head collects data from the (regular) nodes belonging to its own cluster. On the other hand, the same cluster head behaves as a regular (i.e., non cluster head) node when forwarding collected data to other clusters. The slots of the different clusters in the network have the same duration, and repeat periodically within a *frame*, as illustrated in Fig. 1a. Two cluster heads are said to *interfere* if they use the same (receiving) slot and they are at a hop distance not greater than t. Then, we assume that the frame can accommodate k slots, where k is the number of different slots required to serve the clusters without inter-cluster interferences. In other words, k is the length of the frame and corresponds to the length of a feasible plain coloring of the cluster graph.

We model a cluster-based WSN by the *connectivity graph* $G = (V, E)$, where V represents all cluster heads (simply referred to as *nodes* in the rest of the article) and E models the symmetric communication links between the cluster heads. Note that the sink is also a cluster head, hence, it belongs to V as well. We do not make any assumption on how to obtain the clusters and the connectivity graph, which represents the cluster-based WSN topology. For instance, the clusters can represent the sensors in the vicinity of some slightly more powerful nodes that are selected as cluster heads and the connectivity graph can represent adjacent clusters.

In this article, we explicitly address the scenario where interfering nodes (i.e, clusters) can be at most t hops away from each other in the connectivity graph, where t is a constant not smaller than two. In this case, interferences can be accounted for with the help of the *interference graph* G^t defined as the t-th power of the connectivity graph G. Note that $G^t = (V, E^t)$ has the same set of nodes V as G but the set of edges $E^t = \{(i,j) : d(i,j) \leq t\}$, where the distance $d(i,j)$ is the minimum number of edges on any path between the nodes i and j in G.

For instance, Fig. 1b shows a sample cluster-based WSN with a tree topology. In the figure, cluster heads are shown as bigger circles with a dark background. Each cluster consists of a cluster head and its one-hop neighbors. Note that such nodes are either cluster heads or regular nodes, indicated by smaller circles with a light background in the figure. Precisely, V consists of four cluster heads in the figure, i.e., those marked as R and C_i, with $i \in [1, 3]$. The solid lines between the nodes in V indicate the edges E of the tree topology, while the dashed lines indicate, along with the edges E, the interferences when t is set to three. The schedule depicted in Fig. 1c refers to the tree topology in Fig. 1b. Since the resulting conflict graph is complete, $k = 4$ slots are necessary to obtain an interference-free convergecast. Note that nodes C_1, C_2, and C_3 are active for two slots during one frame because they need to both send and receive messages, while the sink node R is active just for one slot because it does not transmit data to any other cluster. In each cluster, the regular nodes are only transmitting at the time slot dictated by their cluster head and no slots are assigned to them.

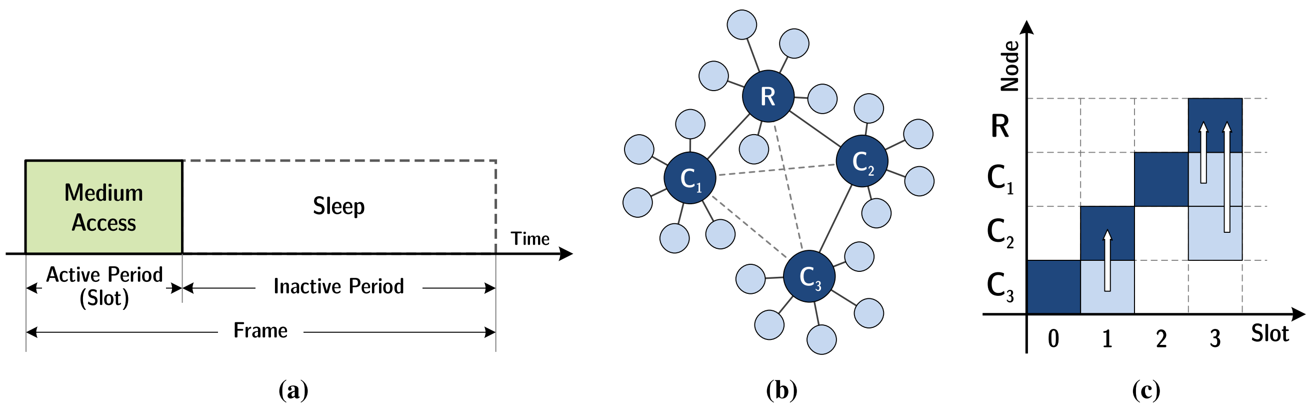

Fig. 1 **a** Duty-cycled channel access in cluster-based WSNs. From each cluster point of view, the frame consists of a single active period (i.e., the receiving slot assigned to its cluster head), and a long inactive period (i.e., all the remaining time in the frame). Overall, the frame contains as many slots as the number of distinct slots required to avoid inter-cluster interference. **b** Sample cluster-tree network topology: nodes with a *dark background* represent cluster heads, while nodes with a *light background* represent (regular) sensors; *solid lines* between the cluster heads indicate the connectivity graph, *dashed lines* (along with the connectivity graph) denote interferences. **c** Interference-free schedule for the WSN shown in (**b**). The *dark squares* represent the receiving slots assigned to the cluster heads while the *light squares* indicate the transmitting slots. For instance, cluster head C_3 receives at time slot 0 from its own cluster nodes and transmits at time slot 1 because so it reaches the cluster head C_2 in the connectivity graph. The frame length is $k = 4$ since any two cluster heads cannot receive during the same slot

3.2 Problem definition

The minimum-latency convergecast problem is to find a sequence of slots, to be periodically repeated in each frame, which guarantees an interference-free scheduling in G and minimizes the latency between the sensor nodes and the sink. Latency is measured as the number of slots between: the instant a message is sent by the source node, and the instant the corresponding (aggregated) message is received by the sink.

The first aspect to consider is that latency cannot be smaller than the number of slots in a frame. This motivates us to study the minimum frame length suitable for our problem. Let *t-Interference-Free Schedule* ($t-$IFS) be a slot assignment $s : V \longrightarrow \mathbb{N}$ which is guaranteed to be interference-free up to the interference distance t. Formally, given a connectivity graph $G = (V, E)$ and the associated interference graph $G^t = (V, E^t)$, a $t-$IFS assigns the slot $s(i)$ to cluster head $i \in V$ so that $s(i) \neq s(j)$ if $(i,j) \in E^t$. Let the *size* S_t of the $t-$IFS be the largest slot assigned to the nodes in V, i.e, $S_t = \max_{i \in V} s(i)$. Then, the *Minimum t-Interference-Free Schedule* ($t-$MIFS) problem is to find the $t-$IFS of minimum size, denoted by $S_t^*(G)$, over all possible $t-$IFSs. The optimal solution for the $t-$MIFS gives the minimum frame length for our problem. Hence, there is no feasible solution for the minimum-convergecast problem if the frame length k is smaller than the minimum number $S_t^*(G)$ of slots required to solve the $t-$MIFS problem.

However, the $t-$MIFS problem is NP-hard for WSN represented by an arbitrary graph. In fact, the $t-$MIFS problem can be reduced (by mapping slots to colors) to a classical node coloring problem on the interference graph G^t. The size of the largest clique in G^t is a lower bound on the minimum size $S_t^*(G)$ of any $t-$IFS, but finding such a size is a well-known hard problem. Due to the difficulty in computing $S_t^*(G)$, we formally define the communication latency using $k \geq S_t^*(G)$. The *report latency* of two adjacent nodes i and j, such that $(i,j) \in E$, is the number of slots, denoted by w_{ij}, that node i has to wait to relay its collected sensory data to node j. If $s(j) \geq s(i)$ then $w_{ij} = s(j) - s(i)$, otherwise $w_{ij} = s(j) - s(i) + k$. In other words, $w_{ij} = [s(j) - s(i)] \bmod k$. The *cumulative report latency*, for each node $i \in V$ connected to the sink, is the sum of the report latencies on the edges of the path from i to the sink in G yielding the minimum latency. Finally, the *convergecast latency*, denoted as $L(G)$, is the maximum of the cumulative report latencies of individual nodes. Obviously, if G is not connected, the cumulative report latency is set to $+\infty$.

In order to better characterize the role played by k, we say that an *inversion* occurs on the path directed from node i towards the sink if there is one edge (u, v) such that $s(u) > s(v)$. Note that message delivery from a source node i to the sink along a certain path p lasts as many complete frames as the number of inversions minus one on such a path. That is, the cumulative report latency L on a path with $c \geq 1$ inversions satisfies $L > (c - 1)k$. This makes the impact of the schedule length (i.e., frame length) on the communication latency apparent. Although a frame of minimum length reduces the latency caused by each inversion, it increases the chance that inversions occur.

We now define the *t-interference-free minimum-latency convergecast* (*t-MC*) problem as follows:

Definition 1 Given a graph $G = (V, E)$, a node $r \in V$, and a constant interference distance t, the $t-$MC problem is to find a $t-$IFS mapping $s : V \rightarrow [0, k-1]$ in $G^t = (V, E^t)$ such that the convergecast latency $L(G)$ towards r is minimized for some $k \geq S_t^*(G)$.

The next theorem shows that $t-$MC problem remains NP-hard even if the minimum frame length to optimally solve the $t-$MIFS problem on G is known.

Theorem 1 *The $t-$MC problem is NP-hard even when $k = S_t^*(G)$.*

Proof In the following, we prove that the decision version of $t-$MC problem with $t = 2$ is NP-complete. We denote such a problem as $2-\mathrm{MC}(d)$: it asks whether there exists a solution with latency at most d for the 2-MC or not. The proof proceeds by a reduction from the satisfiability (SAT) problem, which is known to be NP-complete [32]. The proof that the $t-\mathrm{MC}(d)$ is NP-complete clearly implies that the general $t-$MC problem is NP-hard for $t \geq 2$.

Given a set U of variables and a collection C of clauses over U, the SAT problem is to determine whether a satisfying truth assignment for C exists or not. Given an instance of SAT, we construct an instance of the decision version of 2-MC provided by a graph G as in Fig. 2.

Let $|U| = n$, $|C| = m$, and $k = n \times (m+1) + 3$. We construct G as follows. There is one node r connected to each node but one of a clique K_{k-3} of $k-3$ nodes and to two other nodes p and q. The excluded node of K_{k-3} is connected to p, q, and to one further node s. Among the nodes of K_{k-3} connected to r, n nodes are chosen, each one corresponding to a variable $v \in U$. Each of those n nodes is

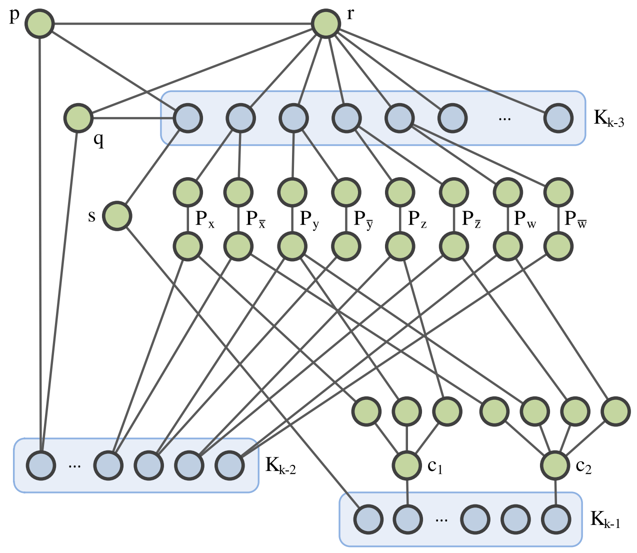

Fig. 2 The graph G resulting from the transformation of an instance of SAT with $n = 4$ and $m = 2$ and clauses $c_1 = xyz$ and $c_2 = \bar{x}y\bar{z}w$ to an instance of 2-MC

connected to a different pair of paths of length two referred as P_v and $P_{\bar{v}}$, $v \in U$. The other extremities of each pair of paths are connected to a different node of another clique K_{k-2} of $k-2$ nodes. A node of K_{k-2} not connected to any path is connected to both p and q.

For each clause $c_i \in C$, $1 \leq i \leq m$, G contains one node denoted as c_i, and a number of nodes equal to the number of literals appearing in c_i connected to the corresponding c_i. Each node representing a literal ℓ, is then connected to P_v ($P_{\bar{v}}$, resp.) on the same side of the connection to K_{k-2}, if $\ell = v$ ($\ell = \bar{v}$, resp.). Moreover, each c_i is connected to a different node of another clique K_{k-1} of $k-1$ nodes. Finally, one of the nodes of K_{k-1} not connected to any c_i is connected to s. By construction, it follows that k is the minimum value required to solve the 2-MC in G.

We are now ready to show that there exists a truth assignment for C if and only if there exists a positive answer for $2-\mathrm{MC}(d)$ in G with $d = 2k-1$ from r.

($\Rightarrow$) Let us assume that SAT has a positive answer, i.e., there exists a satisfying truth assignment for C. We show that $2-\mathrm{MC}(d)$ also has a positive answer considering r as the root, i.e., there exists a $t-$IFS $s : V \rightarrow [0, k-1]$ in G such that the convergecast latency towards r is $L(G) \leq d$. Let $s(r) = \gamma = n \times (m+1) + 2$, $s(p) = \alpha = n \times m$, and $s(q) = \beta = \alpha + 1$. It follows that the two nodes of each path connected to the n nodes of K_{k-3} must necessarily be assigned one with α, and the other with β, since these are the only two possible assignments left from the connections imposed by G. Moreover, node s must necessarily be assigned to γ, as well as all nodes $c_1, \ldots, c_m$. We assign $\alpha - 1$ to the node of K_{k-3} connected to p and q, while we assign values from $\beta + 1$ to $\beta + n$ to the n nodes connected to the paths of length two. We assign $\alpha - 2$ to the node of K_{k-2} connected to p and q, while we assign $\gamma - 1$ to the node of K_{k-1} connected to s. We assign different values in the range $[0, n \times m - 1]$ to the nodes connected to c_i, $1 \leq i \leq m$, representing the appearing literals. For each path P_v ($P_{\bar{v}}$, resp.) with $v \in U$, we assign β (α, resp.) to the node closer to r and α (β, resp.) to the other node, if v is set to true (false, resp.) in the solution of SAT. Finally, all the remaining nodes of the cliques are assigned the remaining values, arbitrarily.

The latency from any node in K_{k-3} or in K_{k-2} is clearly at most d as at most one inversion can be encountered on the shortest path towards r. Any node in K_{k-1}, on the path towards r passing through s, requires a latency of at most $2k-1$. In fact, from K_{k-1} to s, the latency met is at most $k-1$, and another contribution of k arises on the path from s to r towards p, for instance. From any c_i, there must exist a path towards r that passes through K_{k-3} but not through s that provides a latency of at most d. Such a path includes at least one path of two nodes corresponding to the literal set to true by the SAT assignment where values α and β are met in order without providing *inversions*. Hence the values met go from γ passing through one value in $[0,\ldots,n\times m-1]$, then α, β, one value in $[\beta+1,\ldots,\beta+n]$, and finally γ again. Any nodes representing a literal not set to true by the SAT assignment incurs one inversion along the path of two nodes that represents it since β and α are met in the inverse order, however its latency is at most $2k-1$ as its value is different from γ. Overall, $L(G)\leq d$.

($\Leftarrow$) Let us assume to have a positive answer to $2-\mathrm{MC}(d)$. We then show how to assign the appropriate Boolean values to the variables in U, hence satisfying all the clauses in C. Since the latency from any c_i, $1\leq i\leq m$, is at most d, there must be a path from each c_i towards r that passes through K_{k-3} via P_ℓ, where ℓ is a literal appearing in c_i. Any other option requires to pass through node s, and hence the value assigned to the root would be met three times, that is, the latency would be greater than d. Let us consider then the path from c_i to K_{k-3}, it must go through either a P_v or a $P_{\overline{v}}$ depending whether v or $\overline{v}$ appears in c_i. In the former case (and similarly in the latter), the two values met along P_v must be increasing as otherwise the constraint on d would be violated. In such a case, we set to true the literal v ($\overline{v}$ in the latter case). Since any c_i admits a similar argument, there will be at least one literal for each clause that is set to true, hence obtaining a solution for SAT. □

Due to the hardness of the $t-\mathrm{MC}$ problem on arbitrary graphs, we restrict ourselves to solve the $t-\mathrm{MC}$ problem on cluster-based WSNs with a tree topology in the rest of this article.

4 Scheduling for cluster-based WSNs with a tree topology

In this section, we first show how to optimally solve the 2-MC problem with the minimum frame length on arbitrary trees. Then, for $t\geq 3$, we derive lower-bounds for the convergecast latency for arbitrary trees as well as sub-optimal latency by applying the Scheduling-Level-By-Level algorithm. Finally, we design a distributed version of such a scheduling algorithm.

In a WSN with a cluster-tree topology, cluster heads (or simply nodes) form a tree. Let us start with some preliminary notation on trees. A graph $\mathcal{T}=(V,E)$ is a *free tree* when it is connected and it has exactly $|V|-1$ edges. The *distance* $d(u,v)$ between two nodes u and v is the length of the unique path from u to v in $\mathcal{T}$. A *rooted tree* is a free tree in which a node r is identified as the *root* and all other nodes are ordered by levels, where the level $\ell(v)$ of a node v is equal to $d(r,v)$. Thus, all the nodes adjacent to v are partitioned into its *parent* at level $\ell(v)-1$, and its children at level $\ell(v)+1$. The *height* h of a tree $\mathcal{T}$ is the maximum level of its nodes. An *ordered tree* is a rooted tree in which an ordering is imposed among all the children of every node v. Moreover, we denote by $\delta(v)$ the number of children of node v and by $\Delta=\max_{v\in\mathcal{T}}\delta(v)$ the maximum number of children for any node in the ordered tree $\mathcal{T}$.

In the following, for the sake of conciseness, an ordered tree will be called a *tree*. Given a tree $\mathcal{T}$ of height h, an integer $\ell\leq h$, and a node v, the *induced subtree* $\mathcal{T}_\ell(v)$ consists of the subtree rooted at v and height ℓ. For each node v of $\mathcal{T}$, $A_i(v)$ denotes the *ancestor* of v at distance i from v, with $0\leq i\leq h-\ell(v)$. Clearly, $A_i(v)$ is at level $\ell(v)-i$. Besides, $D_i(v)$ indicates the set of the *descendants* of v at distance i from v; thus the descendants belong to level $\ell(v)+i$. Finally, $|D_i(v)|$ denotes the cardinality of $D_i(v)$, for $0\leq i\leq h-\ell(v)$. Note that $|D_1(v)|=\delta(v)$.

The notation used in the article is summarized in Table 1 for convenience.

4.1 Optimal 2-MC on trees with minimum frame length

We now discuss the Min-Latency algorithm that computes the minimum latency for the 2-MC problem in tree networks (Algorithm 1). The minimum latency L^* is computed as $L^*=\ell 1(r)$ by invoking Min-Latency on $\mathcal{T}(r)$. The algorithm visits the tree in post order. For each node v, it computes the minimum latency $\ell 1(v)$, i.e., the minimum

Table 1 Summary of the used notation

Symbol	Description
$G=(V,E)$	Graph of nodes (i.e, cluster heads) V and edges E
$\mathcal{T}$	Tree
$\mathcal{T}(u)$	Subtree of $\mathcal{T}$ rooted at node u
$d(u,v)$	Distance between nodes u and v
t	Interference distance
k	Number of slots in a frame
$s(u)$	Slot assigned to node u
$L(G)$	Convergecast latency of G
$\ell(v)$	Level of node v
h	Tree height
$A_i(u)$	Ancestor of u at distance i from it
$D_i(u)$	Set of the descendants of u at distance i from it
$\delta(u)$	Number of children of node u, i.e., $\delta(u)=\lvert D_1(u)\rvert$
Δ	Maximum number of children for any node in $\mathcal{T}$
$\mathcal{T}_h(u)$	Subtree of height h rooted at u
$\mathcal{T}_\ell$	Set of all nodes $u\in\mathcal{T}$ such that $l(u)\leq\ell$
$N_t(u)$	Set of the neighbors of u at reciprocal distance $\leq t$

time slot at which node v can receive all the data from its subtree. The algorithm starts by assigning the label $\ell 1(v) = 0$ to each leaf v (line 2) because each leaf does not need to receive any data from other nodes. Then, once the labels of the children $c_1, \ldots, c_{\delta(v)}$ of node v have been computed, the label $\ell 1(v)$ is assigned according to line 9. Precisely, the label assigned to node v is the minimum slot when node v can receive after that all its children have received in different time slots the data collected in their subtrees. For example, if v is the parent node of only leaves, which are labeled as zero, then the label of $\ell 1(v) = \delta(v)$ since all children of v must transmit towards v at different time slots because they are at a reciprocal distance of two. If v has two children w_1 and w_2 with labels greater than zero, then $\ell 1(v) \geq \max\{\ell 1(w_1), \ell 1(w_2)\} + 1$. Moreover, if v has two children with the same label $\overline{\ell 1}$ greater than zero, then $\ell(v) \geq \overline{\ell 1} + 2$.

Theorem 2 *The* Min-Latency *algorithm computes in $O(n)$ time the minimum latency necessary to solve the 2-MC problem.*

Proof The algorithm derives the minimum slot at which a generic node v can receive the data collected by all its children. For each node, such a value must be greater than the number of its children. In fact, each child receives data at a different slot as the interference parameter is $t = 2$. Moreover, such a value must be greater than all the labels computed for the children of v. Namely, the slot at which v can receive follows those during which all its children have received, in turn, the data directed to them. To compute the label $\ell 1(v)$ in such a way that the two conditions above are verified, the children of v are considered according to the non-decreasing order of their current labels (line 6). Then, the cycle at line 9 of Algorithm 1 counts $\ell 1(v)$ as follows. Initially, we assign $\ell 1(v) = \ell 1(c_1) + 1$ to v, as c_1 were the unique child of v. Then, we examine all its children. By considering a child w, we update the value $\ell 1(v) = \max\{\ell 1(v), \ell 1(w)\} + 1$. If $\ell 1(w) > \ell 1(v)$ then $\ell 1(v)$ becomes $\ell 1(w) + 1$ because v must wait for all the data of $T(w)$ to be aggregated by w before receiving from it. Otherwise, if $\ell 1(w) \leq \ell 1(v)$, at slot $\ell 1(v)$, the aggregated data are already available at w and $\ell 1(v)$ is just increased of one unit to allow w to transmit to v. □

Once the minimum latency has been computed, the optimal slot assignment for an arbitrary tree when $t = 2$ is obtained by the Opt-Scheduling algorithm (Algorithm 2). It first assigns the slot $L^* = \ell 1(r)$, returned by the Min-Latency algorithm, to the root r of $\mathcal{T}$. It then visits $\mathcal{T}$ in pre-order and uses a frame of length $k = L^* + 1$. The children of r are visited in non-increasing order with respect to the labels assigned by Algorithm 1. For the $i-$th child c_i of r, time slot $(\ell 1(r) - i) \bmod k$ is assigned to node c_i and Opt-Scheduling is recursively invoked on the subtree $\mathcal{T}(c_i)$.

In the following we prove that the convergecast latency obtained by Algorithm 2 is equal to L^* and thus it is optimal. The proof is based on the fact that Algorithm 2 considers $\mathcal{T}$ ordered in the opposite way with respect to Algorithm 1, that is, for each node v, its children are considered in a non-increasing order with respect to the labels obtained after applying Algorithm 1.

Theorem 3 *The* Opt-Scheduling *algorithm optimally solves the 2-MC problem using a frame of length $k = L^* + 1$.*

Proof Let us first prove that, for each node $w \in \mathcal{T}$, the label $\ell(w)$ assigned by Algorithm 2 is never smaller than label $\ell 1(w)$ assigned by Algorithm 1. First, the root gets assigned exactly to the same value $\ell(r) = \ell 1(r) = L^*$.

Next, we prove that the property holds for the children of r. To this end, recall that the children of r are considered in Algorithm 2 in the reverse order with respect to Algorithm 1, i.e., node w which is the $i-$th child c_i in Algorithm 1 corresponds to the $(\delta(r) - i + 1)$-th child $c_{\delta(r)-i+1}$ in Algorithm 2. Moreover, observe that, according to Algorithm 1, $\ell 1(w) + (\delta(r) - i + 1) = \ell 1(c_i) + (\delta(r) - i + 1) \leq \ell 1(r)$, for $1 \leq i \leq \delta(r)$. Whereas, according to Algorithm 2, node w is assigned to slot $\ell(w) = \ell(r) - (\delta(r) - i + 1)$. If, by contradiction $\ell(w) < \ell 1(w)$, one has $\ell(r) = \ell(w) + (\delta(r) - i + 1) < \ell 1(w) + \delta(r) - i + 1) < \ell 1(r) = \ell(r)$, thus raising a contradiction. The same proof can be repeated for the children of any generic internal node w assuming, by induction, that $\ell(w) \geq \ell 1(w)$. We can then conclude that $\ell(w) \geq \ell 1(w)$ holds for each $w \in \mathcal{T}$.

We are now able to the prove that Algorithm 2 does not generate inversions. Let us assume by contradiction there is an inversion along some path from a leaf to r. Let z and w be the first two nodes along such a path starting from the leaf that incur in an inversion. If z is the parent of w, then $\ell(z) < \ell(w)$. Let w be the $i-$th child of z according to the provided order of T. Slot $\ell(w)$ has been computed as $(\ell(z) - i) \bmod k$. Since we assume that an inversion occurs, it must be $\ell(z) \leq i$. In Algorithm 1, the computed $\ell 1(z)$ is always greater than $\delta(z)$ because for each child at least one unit is added and thus $\ell 1(z) > i$. Thus, by the previous property, $\ell(z) \geq \ell 1(z) > i$ contradicting that an inversion can occur.

Finally, since the algorithm does not generate any inversions, the latency is L^* because L^* is the slot assigned to the root and, by construction, there is at least one path starting from a leaf labeled as zero. □

Algorithm 1: MIN-LATENCY($\mathcal{T}(v), v$)

```
output: Minimum latency $\ell1(w)$ for each node
        $w \in \mathcal{T}(v)$
1 if $v$ is leaf then
2     $\ell1(v) = 0$;
3 else
4     foreach $w \in$ children($v$) do
5         MIN-LATENCY($\mathcal{T}(w), w$);
6     Let $c_1, \ldots, c_{\delta(v)}$ be the children of $v$ sorted in such
      a way that $\ell1(c_i) \leq \ell1(c_j)$ for $1 \leq i < j \leq \delta(v)$ ;
7     $\ell1(v) = \ell1(c_1) + 1$;
8     for $2 \leq i \leq \delta(v)$ do
9         $\ell1(v) = \max\{\ell1(v), \ell1(c_i)\} + 1$;
```

Algorithm 2: OPT-SCHEDULING($\mathcal{T}(v)$,v,$\ell(v)$,k)

```
output: Slot assignment for nodes of $\mathcal{T}(v)$
1 Let $c_{j_1}, \ldots, c_{j_{\delta(v)}}$ be the children of $v$ sorted in
  non-increasing order w.r.t. labels $\ell1$ computed by
  Algorithm 1;
2 for $1 \leq i \leq \delta(v)$ do
3     $\ell(c_i) = (\ell(v) - i) \bmod k$;
4     OPT-SCHEDULING($\mathcal{T}(c_i), c_i, \ell(c_i), k$);
```

By observing that $S_2^*(T) = \max\{\delta(r) + 1, \Delta + 2\}$, it is worth to point out that the minimum latency cannot be guaranteed by using a frame of length $k = S_2^*(T)$, even though a feasible solution for 2-MC can be obtained.

Theorem 4 *There exists a tree T of height h such that the minimum latency for solving 2-MC using a frame of length* $k = S_2^*(\mathcal{T})$ *is equal to* $(S_2^*(\mathcal{T}) - 1)h$, *while the minimum latency is* $L^* = (S_2^*(\mathcal{T}) - 1)h - (h - 1)$.

Proof It is enough to consider a tree $\mathcal{T}$ of height h where the root has $\delta(r) = S_2^*(\mathcal{T}) - 1$ and each internal node w has $\delta(w) = S_2^*(\mathcal{T}) - 2$. All the leaves are at the same level h. By applying Algorithm 1, one has $L^* = (k-2)(h-1) + (k-1) = (k-1)h - (h-1)$.

Let us now consider the slot $\ell(r)$ assigned to the root r by an optimal solution that uses a frame of length $k = S_2^*(\mathcal{T})$. Since $\delta(r) = S_2^*(\mathcal{T}) - 1 = k - 1$, there must be a child with label $(\ell(r) + 1) \bmod k$. Let v be such a child, the argument above can be repeated, that is, since $\delta(v) = k - 2$ and the slot $\ell(r)$ cannot be reused for a child of v, there must be a child of v with label $(\ell(v) + 1) \bmod k$. By proceeding similarly until a leaf is reached, the cumulative latency is equal to $(k-1)h$, as the same argument can be applied h times. □

Theorem 3 has established that the minimum latency L^* can be obtained by using a frame of length $L^* + 1$. Now, we want to search for the minimum length k of the frame that guarantees such a minimum latency.

First of all let us observe that, using Algorithm 2 with $\ell(r) = L^*$, the convergecast latency is L^* independent of the used value of k. Indeed, the report latency between node v and its child c_i is equal to i independent of the value of k, since $\ell(c_i) = (\ell(v) - i) \bmod k$ and $(\ell(v) - (\ell(v) - i) \bmod k) \bmod k = i$. However, for $k = S_2^*(\mathcal{T})$ it may happen that Algorithm 2 assigns the same slot to two nodes at a distance equal to two, and hence it would not return a feasible scheduling. For instance, let us consider the tree $\mathcal{T}$ used in the proof of Theorem 4. If we had used Algorithm 2 for such a tree, at least one grandchild of the root would have been assigned the same slot as the root, thus producing an unfeasible scheduling.

Nevertheless, setting $k = \min\{S_2^* + \Delta, L^* + 1\}$ is sufficient to always obtain a feasible solution with minimum latency with Algorithm 2. Indeed, we have proven such in Theorem 3 for the case of $k = L^* + 1$. For the other case, we prove that the slot $\ell(v)$ assigned to a generic node v cannot be repeated at any of its grandchildren. Indeed, the node v, its children $c_1, \ldots, c_{\delta(v)}$ and the children of any child c_i will be assigned to at most 2Δ consecutive slots, starting from $\ell(v)$ and decreasing at most up to $\ell(v) - 2\Delta + 1$, possibly going through 0 and $k - 1$, which are all different slots since $k = S_2^* + \Delta > 2\Delta$.

To find the minimum frame length k which guarantees the minimum latency L^*, a binary search can be computed within the range $[S_2^*(\mathcal{T}), \ldots, \min\{S_2^* + \Delta, L^* + 1\}]$. After selecting a value of k in such an interval, Algorithm 2 finds the actual slot assignment. If the constraint on $t = 2$ is preserved, i.e., the returned slot assignment is feasible, then a new test with a smaller k can be performed, otherwise a larger value of k must be tested. The binary search provides the minimum value for k that guarantees the minimum latency in time $O(n \log \Delta)$ recalling that $S_2^* = \Theta(\Delta)$. Hence, we can conclude that:

Corollary 1 The minimum frame length to optimally solve the 2-MC problem by Algorithm 2 can be found in $O(n \log \Delta)$ time.

In the following, we extend our results to arbitrary values of t. When $t \geq 2h$, all the nodes of the tree must be assigned to different slots since $S_t^*(\mathcal{T}) = n$. Hence, the $t-$MC problem can be optimally solved in $O(n)$ time by traversing the tree in post-order and assigning consecutive slots to the nodes. It remains to be studied the case $t < 2h$. We first provide a lower bound for the convergecast latency and we then propose an algorithm to find a bounded sub-optimal solution to the $t-$MC problem in $O(nt)$ time. Such a solution uses more than $S_t^*(\mathcal{T})$ slots.

4.2 Lower-bound for convergecast latency on trees

Let us first recall that the number of slots $S_t^*(\mathcal{T})$ to optimally solve the $t-$MIFS problem on trees can be

Fig. 3 The t-simplicial neighborhood $N_t(x)$ of a given node x in $\mathcal{T}$

$$N_t(x) = \begin{cases} \bigcup_{i=0}^{\frac{t}{2}-1} \left[D_i(A_{t-i}(x)) \cup D_i(A_{t-1-i}(x))\right] \cup D_{\frac{t}{2}}(A_{\frac{t}{2}}(x)) & \text{if } t \text{ is even} \\ \bigcup_{i=0}^{\lfloor \frac{t}{2} \rfloor} \left[D_i(A_{t-i}(x)) \cup D_i(A_{t-1-i}(x))\right] & \text{if } t \text{ is odd} \end{cases} \quad (1)$$

calculated in $O(nt)$ time, as proved in [33]. This result exploits the following fact: all the nodes belonging to the neighborhood $N_t(x)$ of node x, defined in Fig. 3, are at reciprocal distance at most t and must be assigned to different slots. Therefore, $S_t^*(\mathcal{T}) = \max_{x \in \mathcal{T}} N_t(x)$ slots are required to optimally solve the $t-$MIFS problem.

Let $L_t(v)$ be the latency at node v. For each internal node $v \in \mathcal{T}$, the messages that have been delivered to the nodes in $D_{\lfloor \frac{t}{2} \rfloor}(v)$ must traverse a path of length $\lfloor \frac{t}{2} \rfloor$ to reach v. Therefore, the latency at v is at least $\lfloor \frac{t}{2} \rfloor$ slots larger than the maximum latency in $D_{\lfloor \frac{t}{2} \rfloor}(v)$. Moreover, since at least $\mathcal{T}_{\lfloor \frac{t}{2} \rfloor}(v)$ distinct slots are required to traverse $\mathcal{T}_{\lfloor \frac{t}{2} \rfloor}(v)$ (as argued by the $t-$MIFS problem), the latency at v must be at least $\mathcal{T}_{\lfloor \frac{t}{2} \rfloor}(v)$ greater than the minimum latency at the nodes in $D_{\lfloor \frac{t}{2} \rfloor}(v)$. Thus, $L_t(v)$ can be defined recursively as:

$$L_t(v) = \begin{cases} 0 & \text{if } v \text{ is a leaf} \\ \max\left\{ \max_{u \in D_{\lfloor \frac{t}{2} \rfloor}(v)} L_t(u) + \left\lfloor \frac{t}{2} \right\rfloor, \right. & \\ \left. \min_{u \in D_{\lfloor \frac{t}{2} \rfloor}(v)} L_t(u) + \left|\mathcal{T}_{\lfloor \frac{t}{2} \rfloor}(v)\right| - 1 \right\} & \text{otherwise} \end{cases}$$

Easily it follows:

Lemma 1 Given the interference distance t, a tree $\mathcal{T}$ of height h with $h \geq t$ and rooted at node r, the convergecast latency yields $L(\mathcal{T}) \geq \max\{h, S_t^*(\mathcal{T}) - 1, L_t(r)\}$.

Algorithm 3: DESCENDANTS($x, \mathcal{T}, t$)

```
output: the sets $D_0(x), \ldots, D_t(x)$ and their sizes
1 $D_0(x) \leftarrow \{x\}$; $|D_0(x)| \leftarrow 1$;
2 for $1 \leq i \leq t$ do  $D_i(x) \leftarrow \varnothing$; $|D_i(x)| \leftarrow 0$;
3 if $x$ is not leaf then
4     foreach $v \in$ children($x$) do
5         DESCENDANTS($v, T, t$);
6         for $i \leftarrow 1$ to $t$ do
7             $D_i(x) \leftarrow D_i(x) \cup D_{i-1}(v)$;
8             $|D_i(x)| \leftarrow |D_i(x)| + |D_{i-1}(v)|$;
```

Algorithm 4: COUNT-SLOTS($\mathcal{T}, t$)

```
output: $|\Gamma_i|$ for level $\ell \in \mathcal{F}_i$, with $0 \leq i \leq t$
1 $\ell(r) \leftarrow 0$; $|\Gamma_0| \leftarrow 1$; for $1 \leq i \leq t$ do $|\Gamma_i| \leftarrow 0$;
2 $Q \leftarrow \varnothing$; $r \leftarrow$ root($T$); enqueue ($r, Q$);
3 while $Q \neq \varnothing$ do
4     $u \leftarrow$ dequeue($Q$);
5     foreach $v \in$ children($u$) do
6         $\ell(v) \leftarrow \ell(u) + 1$;
7         if $\ell(v) \leq \lfloor \frac{t}{2} \rfloor$ then $|\Gamma_{\ell(v)}| \leftarrow |\Gamma_{\ell(v)}| + 1$;
8         $i \leftarrow (\ell(v) + \lfloor \frac{t}{2} \rfloor) \bmod (t+1)$;
9         $|\Gamma_i| \leftarrow \max(|\Gamma_i|, |D_{\lfloor \frac{t}{2} \rfloor}(v)|)$;
10        if $\ell(v) \leq h - \lfloor \frac{t}{2} \rfloor$ or $\ell(v) \leq \lfloor \frac{t}{2} \rfloor$ then
11            enqueue ($v, Q$);
```

4.3 The SCHEDULING-LEVEL-BY-LEVEL algorithm

Our algorithm applies as much as possible the following two operations:

1) assigning the same slot to any two nodes in the same level of the tree when their distance is exactly $t+1$;
2) assigning the same set of slots to the nodes belonging to two levels at distance equal to $t+1$.

With reference to Operation (1), the nodes at level $\ell \geq \lfloor \frac{t}{2} \rfloor$ of $\mathcal{T}$ can be partitioned, visited from left to right, into disjoint subsets, each consisting of consecutive nodes at a mutual distance not greater than t. The nodes in these subsets share a common ancestor at distance $\lfloor \frac{t}{2} \rfloor$. Indeed, each of such subsets at level ℓ corresponds to the descendants $D_{\lfloor \frac{t}{2} \rfloor}(x)$ at distance $\lfloor \frac{t}{2} \rfloor$ of some node x which belongs to level $\ell - \lfloor \frac{t}{2} \rfloor$. Consecutive subsets at level ℓ are then descendants of consecutive nodes at level $\ell - \lfloor \frac{t}{2} \rfloor$. Hence, the distance between nodes in such consecutive subsets is greater than t because their lowest common ancestor belongs to a level smaller than $\ell - \lfloor \frac{t}{2} \rfloor$. Instead, all the nodes at level $\ell < \lfloor \frac{t}{2} \rfloor$ are at pairwise distance smaller or equal to t and hence belong to the same subset.

Algorithm 5: SCHEDULING-LEVEL-BY-LEVEL $(\mathcal{T}, t, \{|\Gamma_i|\})$

```
output: the slot $s(v)$ for each node $v \in \mathcal{T}$
1  $k \leftarrow \sum_{h=0}^{t} |\Gamma_h|$; $\mu_{-1} \leftarrow k$;
2  for $0 \le i \le t$ do  $\mu_i \leftarrow \mu_{i-1} - |\Gamma_i|$;
3  $r \leftarrow$ root$(T)$; $s(r) \leftarrow \mu_0$; $\ell(r) \leftarrow 0$; $prec \leftarrow r$;
4  $Q \leftarrow \varnothing$; if $\lfloor \frac{t}{2} \rfloor > 1$ then enqueue $(r, Q)$;
   insert-queue $(D_{\lfloor \frac{t}{2} \rfloor}(r), l(r) + \lfloor \frac{t}{2} \rfloor, Q_D)$;
5  while $Q \neq \varnothing$ do
6      $u \leftarrow$ dequeue$(Q)$;
7      foreach $v \in$ children$(u)$ do
8          $\ell(v) \leftarrow \ell(u) + 1$; $i \leftarrow \ell(v)$;
9          if $\ell(prec) \neq \ell(v)$ then  $temp \leftarrow \mu_i$;
10         $s(v) \leftarrow temp$; $temp \leftarrow temp + 1$; $prec \leftarrow v$;
11         if $\ell(v) < \lfloor \frac{t}{2} \rfloor - 1$ then enqueue $(v, Q)$;
12         $\ell_D \leftarrow \ell(v) + \lfloor \frac{t}{2} \rfloor$;
13         insert-queue $(D_{\lfloor \frac{t}{2} \rfloor}(v)$, $\ell_D$, $Q_D)$;
14 while $Q_D \neq \varnothing$ do
15     extract-queue (S, level, $Q_D$);
16     $u \leftarrow$ first$(S)$; $\ell(u) \leftarrow level$; $i \leftarrow level$
       mod$(t+1)$;
17     $temp \leftarrow \mu_i$; $\ell_D \leftarrow \ell(u) + \lfloor \frac{t}{2} \rfloor$;
18     foreach $u \in S$ do
19         $s(u) \leftarrow temp$; $temp \leftarrow temp + 1$;
20         insert-queue $(D_{\lfloor \frac{t}{2} \rfloor}(u)$, $\ell_D$, $Q_D)$;
```

Therefore, in order to guarantee an interference-free assignment for each level ℓ, it is sufficient to assign a number of slots

$$\gamma_\ell = \begin{cases} |D_\ell(r)| & \text{if } \ell < \left\lfloor \frac{t}{2} \right\rfloor \\ \max_{\{x \in \mathcal{T} | l(x) = \ell - \lfloor \frac{t}{2} \rfloor\}} \{|D_{\lfloor \frac{t}{2} \rfloor}(x)|\} & \text{if } \ell \ge \left\lfloor \frac{t}{2} \right\rfloor \end{cases}$$

where $D_0(r) = \{r\}$ by definition.

By Operation (2), the set of slots assigned to level i can be reused for a family of levels $\mathcal{F}_i = \{\ell : \ell \equiv i \bmod (t+1)\}$ by still guaranteeing a $t -$ IFS for $\mathcal{T}$. Specifically, the size of the same group Γ_i of slots assigned to the levels $\mathcal{F}_i$, $0 \le i \le t$, is $|\Gamma_i| = \max\{\gamma_\ell : \ell \in \mathcal{F}_i\}$.

In the following, based on the observations above, we propose an algorithm, called SCHEDULING-LEVEL-BY-LEVEL, which obtains a $t -$ IFS with a convergecast latency of $L(\mathcal{T}) < \lceil \frac{h+1}{(t+1)} \rceil k$ by using $k = \sum_{i=0}^{t} |\Gamma_i|$ slots.

The proposed solution assumes that some preprocessing is performed before running the actual scheduling algorithm. The first preprocessing step obtains, for each node x of T, the set $D_{\lfloor \frac{t}{2} \rfloor}(x)$ and the size $|D_{\lfloor \frac{t}{2} \rfloor}(x)|$ of the descendants at distance $\lfloor \frac{t}{2} \rfloor$ from x. The recursive algorithm DESCENDANTS obtains such parameters starting from the root r of $\mathcal{T}$, as illustrated in Algorithm 3. The related computation can be performed in $O(nt)$ time by visiting the tree in postorder and by storing the descendants in linked lists such that the t unions of lists required for each node run in $O(t)$ time. The second preprocessing step derives the number $|\Gamma_i|$ of slots to be assigned at the levels in the different families $\mathcal{F}_i$. The COUNT-SLOTS algorithm obtains such parameters by performing a modified breadth-first search of $\mathcal{T}$, as illustrated in Algorithm 4. While traversing the tree with the help of the queue Q (line 2), COUNT-SLOTS computes the level $\ell(v)$ of each node v in $\mathcal{T}$ (line 6). Once such level is known, the algorithm calculates the number of slots required to avoid interferences. To this end, there are two different cases. For the uppermost $\lfloor \frac{t}{2} \rfloor$ levels, $|\Gamma_i|$ is set to the number of nodes in those levels, since all the related nodes are at pairwise distance not greater than t (line 7). For the rest of the levels, COUNT-SLOTS finds the size of the largest set of descendants at distance $\lfloor \frac{t}{2} \rfloor$ from the nodes at level $\ell(v)$ as $|\Gamma_i|$, where $i = (\ell(v) + \lfloor \frac{t}{2} \rfloor) \bmod (t+1)$ (lines 8–9).

After the preprocessing steps, SCHEDULING-LEVEL-BY-LEVEL finds the actual node-slot assignment, as illustrated in Algorithm 5. As a preliminary phase, the algorithm obtains the slots allocated to each level $\ell \in \mathcal{F}_i$ of $\mathcal{T}$. In detail, it calculates first the starting slot μ_i dedicated to the levels in the family $\mathcal{F}_i$ as $\mu_i = \mu_{i-1} - |\Gamma_i|$, where $0 \le i \le t$ and $\mu_{-1} = k = \sum_{i=0}^{t} |\Gamma_i|$ (lines 1–3). Moreover, the algorithm visits the tree level-by-level by using two queues for different groups of nodes. The descendants at distance i with $0 \le i \le \lfloor \frac{t}{2} \rfloor - 1$ of the root (i.e., the nodes at level i) are stored in the queue Q (lines 4 and 11). Consecutive slots of Γ_i, starting from μ_i, are then assigned to the nodes in Q which belong to level i (line 10). Nodes belonging to the levels greater than or equal to $\lfloor \frac{t}{2} \rfloor$ are handled differently. Namely, such nodes are stored in the queue Q_D of the descendants as groups of nodes which share the same ancestor at distance $\lfloor \frac{t}{2} \rfloor$. Specifically, for each node v at level ℓ, $0 \le \ell \le h - \lfloor \frac{t}{2} \rfloor$, the nodes of the subset $D_{\lfloor \frac{t}{2} \rfloor}(v)$, which are obviously at a level greater than or equal to $\lfloor \frac{t}{2} \rfloor$, are inserted altogether into Q_D when v is visited (lines 13 and 20). Each of such subset of nodes is extracted all at once from Q_D while visiting level $\ell + \lfloor \frac{t}{2} \rfloor$. Such extracted nodes are assigned, from left to right, to consecutive slots in Γ_i starting from μ_i, where $i = (\ell + \lfloor \frac{t}{2} \rfloor) \bmod (t+1)$ (line 19).

Example 1 **(Slot assignment)** Figure 4 shows a sample tree with the corresponding slot assignment obtained by SCHEDULING-LEVEL-BY-LEVEL when the interference distance is $t = 3$. In the figure, the white number within a circle denotes the node identifiers, while the black number next to a circle represents the slot assigned to the corresponding node.

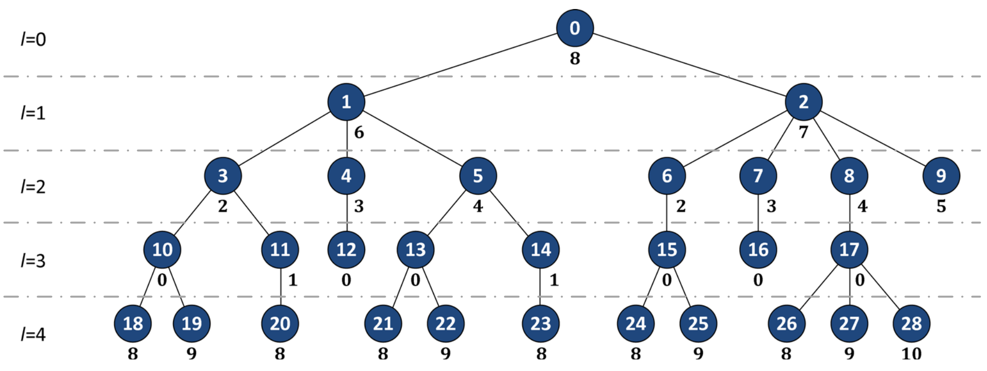

Fig. 4 Sample cluster-based WSN with tree topology and the corresponding slot assignment obtained by Scheduling-level-by-level; the number within a *circle* denotes the node number, while the number next to a *circle* represents the slot assigned to the corresponding node

In order to obtain the actual slot assignment, some preliminary operations are first performed on the tree. Specifically, Descendants counts the descendants at distance $\lfloor \frac{t}{2} \rfloor$ for each node in the tree. As $t = 3$, it is $\lfloor \frac{t}{2} \rfloor = 1$ thus $D_1(x)$, the descendants at distance 1 of a given node x, are simply the children of that node x. Then, Count-Slots partitions the levels of the tree in the families $\mathcal{F}_i$ that are $t+1$ levels apart, with $0 \leq i \leq t$. Thus, it is $\mathcal{F}_0 = \{0, 4\}$, $\mathcal{F}_1 = \{1\}$, $\mathcal{F}_2 = \{2\}$, and $\mathcal{F}_3 = \{3\}$. The next phase derives, for each family of levels, the number of slots which guarantees an interference-free scheduling. The $|\Gamma_i|$ associated to the $i-$th family of levels is equal to the maximum number of children which share a common parent in all levels belonging to that family. Hence, the slots assigned to the first family of levels is $|\Gamma_0| = \max\{\gamma_0, \gamma_4\}$, where $\gamma_0 = |D_0(0)| = 1$ and $\gamma_4 = \max\{|D_1(x)| : l(x) = 3\} = 3$. Similarly, the slots assigned to the rest of the families are: $|\Gamma_1| = \max\{|D_1(x)| : l(x) = 0\} = 2$; $|\Gamma_2| = \max\{|D_1(x)| : l(x) = 1\} = 4$; and $|\Gamma_3| = \max\{|D_1(x)| : l(x) = 2\} = 2$. Once the number of slots $|\Gamma_i|$ is available for each family of levels, the total number of slots k is obtained trivially as the sum of all $|\Gamma_i|$, that is, $k = \sum_{i=0}^{3} |\Gamma_i| = 11$.

At this stage, Scheduling-level-by-level has all the necessary information to assign the slots to all nodes in the tree. In order to assign slots level by level in increasing order, the algorithm derives the starting slot μ_i associated to the levels belonging to the $i-$th family $\mathcal{F}_i$. The starting slot of the first family is set to $\mu_0 = k - |\Gamma_0| = 8$. The starting slots for the rest of the families are: $\mu_1 = \mu_0 - |\Gamma_1| = 6$; $\mu_2 = \mu_1 - |\Gamma_2| = 2$; and $\mu_3 = \mu_2 - |\Gamma_3| = 0$.

The slot assignment for the first two levels of the tree is handled in a slightly different way than for rest of the tree. This is because all nodes at levels zero and one (i.e., those in the subtree of size $\lfloor \frac{t}{2} \rfloor$ rooted at the sink) interfere with each other. As a consequence, no slots can be reused by different nodes belonging to the same level in this case. To this end, Scheduling-level-by-level just assigns μ_i as the slot associated to the first node of the $i-$th family, and then assigns increasing slot numbers to the rest of the nodes in that level, for $0 \leq i \leq 1$. That is, $s(0) = \mu_0 = 8$ at level $l = 0$. Moreover, $s(1) = \mu_1 = 6$ and $s(2) = \mu_1 + 1 = 7$ at level $l = 1$.

For each of the other levels with $l \geq 2$, nodes are grouped together so that the nodes in a group are those which share the same parent, i.e., their ancestor at distance one. Nodes belonging to different groups can then reuse the same slots. With reference to Fig. 4, we can see that nodes at level 2 can be divided in two groups: the children of node 1, namely, $\mathcal{C}_1 = \{3, 4, 5\}$; and the children of node 2, namely, $\mathcal{C}_2 = \{6, 7, 8, 9\}$. Within each group, slots are assigned sequentially starting from μ_2, since level 2 belongs to the second family $\mathcal{F}_2$. As a consequence, it is: $s(3) = \mu_2 = 2$, $s(4) = \mu_2 + 1 = 3$, and $s(5) = \mu_2 + 2 = 4$; while $s(6) = \mu_2 = 2$, $s(7) = \mu_2 + 1 = 3$, $s(8) = \mu_2 + 3 = 4$ and $s(9) = \mu_2 + 4 = 5$. The process is similar for levels 3 and 4. Note that level 4 belongs to the first family $\mathcal{F}_0$, as $4 \bmod (t+1) = 0$, thus it can reuse the slots assigned to level 0, i.e., those starting from μ_0. $\triangle$

Theorem 5 *The* Scheduling-Level-By-Level *algorithm is correct and can be computed in* $O(n)$ *time.*

Proof Let us consider a node v that gets assigned slot $s(v)$. By construction, any other node u that gets assigned to the same slot, i.e., $s(u) = s(v)$ is $t+1$ levels up or down the tree or u is at the same level as v but their lowest common ancestor is at distance $\lceil \frac{t}{2} \rceil$ from them. As a consequence, the algorithm is correct, i.e., it obtains an interference-free schedule.

With respect to the time complexity, Scheduling-Level-by-Level mainly performs a breadth-first-search of $\mathcal{T}$; therefore it requires $O(n)$ time. $\square$

Finally, by accounting for the time complexity of the Descendants and Count-Slots, the $t-$MC problem is solved in $O(nt)$ time.

4.4 Approximation factor

It remains to analyze how far is the latency achieved by SCHEDULING-LEVEL-BY-LEVEL from the optimal latency.

Theorem 6 When $h \geq 2\lfloor \frac{t}{2} \rfloor + 1$, SCHEDULING-LEVEL-BY-LEVEL provides an approximation factor of ρ, where $\rho < \frac{1}{c}(\lfloor \frac{t}{2} \rfloor + 1)\frac{\overline{\gamma}}{\widehat{\gamma}}$ with $\frac{1}{2} \leq c \leq 1 - \frac{\lfloor \frac{t}{2} \rfloor + 1}{h+1}$, $\widehat{\gamma} = \min_{\{x \in \mathcal{T}: |D_{\lfloor \frac{t}{2} \rfloor}(x)| \neq 0\}} |D_{\lfloor \frac{t}{2} \rfloor}(x)|$, and $\overline{\gamma} = \max_{x \in \mathcal{T}} |D_{\lfloor \frac{t}{2} \rfloor}(x)|$.

Proof Our goal is to evaluate the approximation factor ρ that bounds the ratio between the actual latency $L(\mathcal{T})$ obtained by the algorithm and the optimal latency $L^*(\mathcal{T})$. Since the optimal latency $L^*(\mathcal{T})$ is not actually known and since the lower bound in Lemma 1 cannot be easily defined as a closed-form expression, we derive a weaker lower bound for $L^*(\mathcal{T})$ as follows.

For traversing a subtree of height $\lfloor \frac{t}{2} \rfloor$, at least a latency equal to the maximum between $\lfloor \frac{t}{2} \rfloor$ and the size of the leaves of such a subtree is experienced. Let $\widehat{\gamma} = \min_{\substack{x \in \mathcal{T}: \\ |D_{\lfloor \frac{t}{2} \rfloor}(x)| \neq 0}} |D_{\lfloor \frac{t}{2} \rfloor}(x)|$ be the minimum number of leaves among all the subtrees of height $\lfloor \frac{t}{2} \rfloor$ rooted at any node $x \in \mathcal{T}$. To reach the root r starting from the leaves, at least $\left\lfloor \frac{h+1}{\lfloor \frac{t}{2} \rfloor + 1} \right\rfloor$ subtrees of height $\lfloor \frac{t}{2} \rfloor$ must be traversed. Independent from the number of slots in each frame, it is

$$L^*(\mathcal{T}) \geq \left(\left\lfloor \frac{h+1}{\lfloor \frac{t}{2} \rfloor + 1} \right\rfloor \right) \widehat{\gamma} \geq c \frac{h+1}{\lfloor \frac{t}{2} \rfloor + 1} \widehat{\gamma}$$

where

$$\frac{1}{2} \leq c \leq 1 - \frac{\lfloor \frac{t}{2} \rfloor + 1}{h+1} \text{ if } h \geq 2\left\lfloor \frac{t}{2} \right\rfloor + 1.$$

On the other hand, with respect to the actual latency, let us consider an arbitrary subpath of length $t+1$ starting at level $\ell \equiv t \bmod (t+1)$ and decreasing down to level $\ell - (t+1)$. The slots assigned to such a subpath form a sequence sorted in increasing order because the nodes at level $\ell \equiv i \bmod (t+1)$ are assigned to the slots in group Γ_i, where $t \geq i \geq 0$. The cumulative report latency along such a subpath is at most k, and hence, the cumulative report latency in $\mathcal{T}$ is upper bounded by $\sum_{i=1}^{\lceil \frac{h+1}{t+1} \rceil} k$. Since $k \leq (t+1)\max\{|\Gamma_i| : 1 \leq i \leq t\} = (t+1)\overline{\gamma}$, SCHEDULING-LEVEL-BY-LEVEL leads to an actual latency $L(\mathcal{T}) < (h+1)\overline{\gamma}$.

In conclusion, the approximation factor ρ yields:

$$\rho = \frac{L(\mathcal{T})}{L^*(\mathcal{T})} < \frac{(h+1)\overline{\gamma}}{c\frac{h+1}{\lfloor \frac{t}{2} \rfloor + 1}\widehat{\gamma}} < \frac{1}{c}\left(\left\lfloor \frac{t}{2} \right\rfloor + 1\right)\frac{\overline{\gamma}}{\widehat{\gamma}} \qquad (1)$$

□

Note that the above approximation factor becomes loose when the ratio $\frac{\overline{\gamma}}{\widehat{\gamma}}$ is high. In other words, the ratio is surely strict for almost balanced trees. However, the expression for ρ has two advantages: it explicitly shows the dependence on the interference distance t; it only relies on the size of the descendants at distance $\lfloor \frac{t}{2} \rfloor$ from the nodes in the tree.

4.5 Distributed version

In this section, we provide a distributed version of SCHEDULING-LEVEL-BY-LEVEL based on the structural properties of the network. Specifically, the distributed solution exploits the fact that the parameters of interest can be efficiently computed in a distributed way for a rooted tree [34].

Before proceeding further, recall that SCHEDULING-LEVEL-BY-LEVEL assigns the slot $s(u) = \mu_i + \delta(u)$ to node u belonging to level $\ell(u)$, where $i = |\ell(u)| \bmod (t+1)$. Also recall that: μ_i is the slot associated to the first node of the $i-$th family; $\delta(u)$ is a bijective function between the set of nodes in $D_{\lfloor \frac{t}{2} \rfloor}(A_{\lfloor \frac{t}{2} \rfloor}(u))$ and the integers in the interval $[0, \ldots, |D_{\lfloor \frac{t}{2} \rfloor}(A_{\lfloor \frac{t}{2} \rfloor}(u))| - 1]$. SCHEDULING-LEVEL-BY-LEVEL assigns the slot $\delta(u)$ to node u based on its order in the set $D_{\lfloor \frac{t}{2} \rfloor}(A_{\lfloor \frac{t}{2} \rfloor}(u))$, by counting the nodes in $D_{\lfloor \frac{t}{2} \rfloor}(A_{\lfloor \frac{t}{2} \rfloor}(u))$ at level $\ell(u)$ from left to right. The distributed version emulates the behavior of the centralized algorithm but it computes a different bijective function δ', irrespective of the left-to-right order, between the set of nodes in $D_{\lfloor \frac{t}{2} \rfloor}(A_{\lfloor \frac{t}{2} \rfloor}(u))$ and the integers in the interval $[0, \ldots, |D_{\lfloor \frac{t}{2} \rfloor}(A_{\lfloor \frac{t}{2} \rfloor}(u))| - 1]$. Nevertheless, the obtained schedule is still correct since the new mapping δ' remains bijective.

Let us now show how to compute the slot $s'(u)$ assigned to each node $u \in \mathcal{T}$ in a distributed way. In the following, we assume that the level of each node has been already computed by performing a distributed breadth-first-search of $\mathcal{T}$.

First, the set of descendants $D_{\lfloor \frac{t}{2} \rfloor}(u)$ of each node $u \in \mathcal{T}$ is computed as follows. Every node sends a message with its identifier and a hop counter h_c, initialized to 0, to its parent. A node that receives such a message increases the counter h and forwards it to its parent as long as $h_c < \lfloor \frac{t}{2} \rfloor$. After each message has traversed $\lfloor \frac{t}{2} \rfloor$ hops, the node $x = A_{\lfloor \frac{t}{2} \rfloor}(u)$ has received the identifiers of all the nodes in $D_{\lfloor \frac{t}{2} \rfloor}(x)$. Thus, node x calculates $|D_{\lfloor \frac{t}{2} \rfloor}(x)|$ locally and assigns a different integer $\delta'(u)$ between $[0,\ldots|D_{\lfloor\frac{t}{2}\rfloor}(x)|-1]$ to each node $u \in D_{\lfloor\frac{t}{2}\rfloor}(x)$. Subsequently, node x sends its assigned integer $\delta'(u)$ to the node $u \in D_{\lfloor\frac{t}{2}\rfloor}(x)$. Overall, $O(nt)$ messages are exchanged to compute $D_{\lfloor\frac{t}{2}\rfloor}(u)$ and $\delta'(u)$ for each node $u \in \mathcal{T}$.

Second, Γ_is are obtained in order to derive the starting slots μ_i for each $i \in 0,\ldots,t$. To this end, the maximum among $|D_{\lfloor\frac{t}{2}\rfloor}(v)|$ for each node v such that $\ell(v) \bmod (t+1) = i$ is computed. This operation can also be obtained in a distributed way, similar to the previous discussion. Given the Γ_i, the values k and μ_i with $0 \le i \le t$ are locally computed at the root according to lines 1–2 of Algorithm 5. Finally, each value μ_i is broadcast to all nodes at level $\ell(u)$ such that $\ell(u) \bmod (t+1) = i$, for $0 \le i \le t$. Also in this case, the computation requires exchanging $O(nt)$ messages overall. To complete the slot assignment, each node u locally computes $s'(u) = \mu_i + \delta'(u)$ where $i = \ell(u) \bmod (t+1)$.

In the special case of complete Δ-ary trees, the overhead and complexity of the distributed version further reduce. A *complete* Δ*-ary tree* $\mathcal{T}_\Delta$ of size n is a rooted tree in which all the leaves are at the same level and each internal node has exactly Δ children. By recalling that the levels are numbered from 0, the height of $\mathcal{T}_\Delta$ is $h = \lfloor \log_\Delta(n) \rfloor$. Slot assignment for a complete Δ-ary tree of height $h \ge t$ can be accomplished through Complete-Tree-Scheduling illustrated in Algorithm 6. It assumes that each node u knows the pair $u = (\ell(u), p(u))$, where $\ell(u)$ is the level of u and the position of u in level $\ell(u)$ from left to right, where $0 \le p(U) \le \Delta^{\ell(u)} - 1$. Such a pair can be computed for each node by computing a breadth-first-search of the tree in $O(n)$ time. Since it is well known how to perform a breadth-first-search in a distributed way by exchanging $O(n)$ messages [34, 35], slots can be assigned to each node in $O(1)$ time by exploiting only local information.

```
Algorithm 6: COMPLETE-TREE-SCHEDULING($v$)
output: the slot $s(v)$ for node $v = (\ell(v), p(v))$;
$s(v) = [(t - \ell(v)) \bmod (t+1)]\Delta^{\lfloor\frac{t}{2}\rfloor} + p(v) \bmod \Delta^{\lfloor\frac{t}{2}\rfloor}$;
```

Corollary 2 The Complete-Tree-Scheduling algorithm applied to a complete tree $\mathcal{T}_\Delta$ of height $h \ge t$ requires $U_t = (t+1)\Delta^{\lfloor\frac{t}{2}\rfloor}$ slots and provides a $t-$IFS with convergecast latency $L(\mathcal{T}_\Delta) < \left\lceil\frac{h+1}{t+1}\right\rceil U_t$. This solution guarantees a ρ-approximation for the $t-$MC problem, where

$$\rho < \begin{cases} (1+\epsilon)(t+1) & \textit{if t is even} \\ (1+\epsilon)\dfrac{t+1}{2} & \textit{if t is odd} \end{cases}$$

where $0 \le \epsilon < 1$.

Proof Due to the tree regularity, it is easy to see that $\overline{\gamma} = \widehat{\gamma} = \Delta^{\lfloor\frac{t}{2}\rfloor}$, and hence the approximation factor becomes independent of the neighborhood size. □

5 Evaluation

In order to evaluate our Scheduling-level-by-level algorithm for arbitrary t, we performed several simulation experiments. We used a custom simulator written in python and considered synthetic tree topologies consisting of both random and deterministic trees. Random trees were obtained by adding a number of children (i.e., cluster heads) uniformly distributed between one and three to the sink, then recursively to all child nodes up to a given height. Deterministic trees were represented by Δ-ary complete trees of a given height. To prevent the MAC protocol to affect the scheduling process, we generate only the cluster heads in each cluster and no regular nodes in our simulations.

We consider the following metrics to evaluate the performance of the proposed algorithms.

- *Convergecast latency* in terms of slots, as defined in Sect. 3.2.
- *Approximation factor* with two different granularities. The *coarse*-grained approximation factor is the value of ρ in Theorem 6, e.g., the one in Eq. (1). Moreover, the *fine*-grained approximation factor is the ratio between the actual latency and the lower bound given in Lemma 1.

We performed 200 independent replications of the experiment. In the plots, we reported the average values as well as the corresponding standard deviations, shown as error bars.

Figure 5 shows the convergecast latency in synthetic trees as a function of their height. The values in the plots, which also show the upper and the lower bounds, were derived for different values of the interference distance t. The figure shows that the actual convergecast latency (i.e., the one obtained by Scheduling-level-by-level) increases with the height h of the tree almost linearly, while the upper bound $\lceil\frac{h+1}{t+1}\rceil k$ of the latency has a sharper behavior as it increases by k every $t+1$ levels. Clearly, the convergecast latency increases with t as well. Furthermore, its value is close to the lower bound over the whole range of considered heights for $t = 2$ (Fig. 5), different from what happens for $t = 4$ (Fig. 5b) and $t = 6$ (Fig. 5c), especially for $h > 4$. Overall, the actual convergecast latency almost corresponds to the average value of the upper and the lower bounds, except for the region where $h \le \lfloor\frac{t}{2}\rfloor$. This happens because slots cannot be reassigned in the first $\lfloor\frac{t}{2}\rfloor$ levels, and the convergecast latency is affected by the size of the entire tree rather than how the nodes are distributed at a

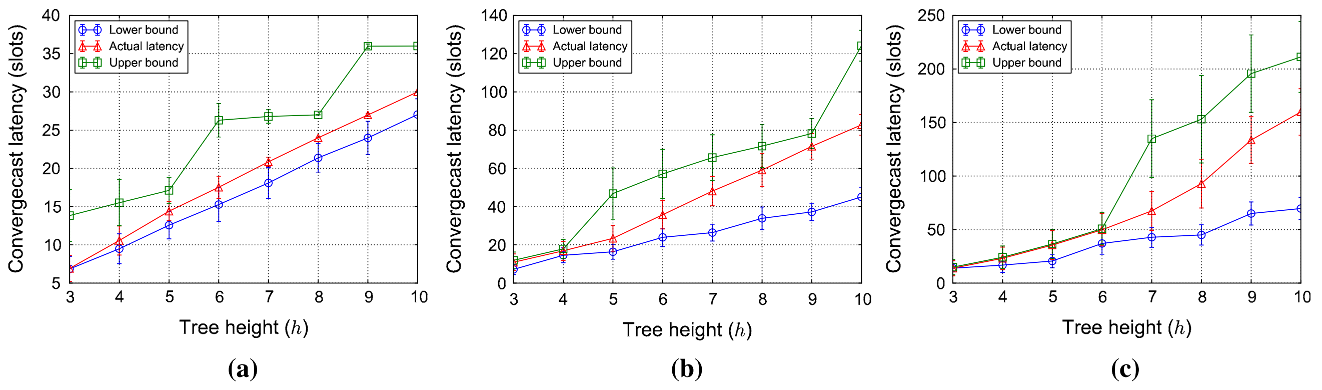

Fig. 5 Convergecast latency (along with the corresponding *upper* and *lower bounds*) in synthetic trees as a function of their height for different values of the interference distance t: **a** $t = 2$, **b** $t = 4$, and **c** $t = 6$

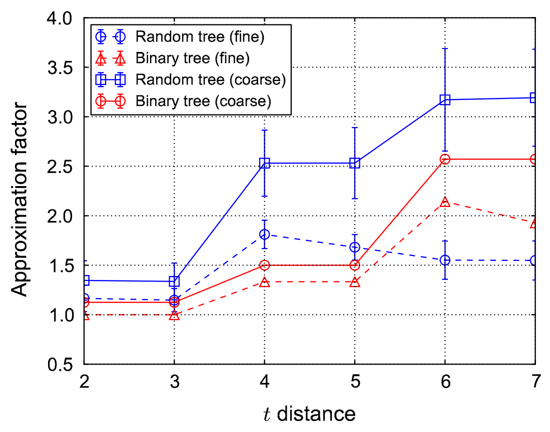

Fig. 6 Approximation factor in synthetic (random and complete binary) trees as a function of the interference distance t

given distance. This also explains why the upper bound and the actual latency are close.

Figure 6 shows the coarse- and fine-grained approximation factors in synthetic trees as a function of the interference distance t. In the figure, we considered random trees of height equal to seven and a complete binary tree of height equal to eight, in order to have the same number of clusters in the two cases. We observe that the coarse-grained approximation factor always exceeds the fine-grained one, as it should be. Moreover, the two approximation factors are close to each other (and also to one) for the balanced binary tree when $t < 6$. This happens because of the regularity of the tree, which makes the estimate of the coarser bound more precise. The difference between the fine- and the coarse-grained approximation factors is higher for random trees and increases with the interference distance t. This is due to the higher variability in the neighborhoods of the clusters, especially at the lowest levels of the tree.

In conclusion, our solution obtains a latency which is always bound to the optimal scheduling by a small factor, as it can be seen in Fig. 6. Thus, the results confirm the validity and the accuracy of the bounds presented in Sect. 4.3.

6 Conclusion

This article addressed latency-bounded data collection in cluster-based wireless sensor networks (WSNs) with reference to aggregated convergecast traffic. We characterized the problem of minimum-latency data collection in cluster-based WSNs with arbitrary topologies where nodes use a duty-cycled mechanism for energy conservation. We considered interferences between clusters up to a distance $t \geq 2$ and derived bounds for latency-sensitive applications. Due to the hardness of the problem, we focused our attention on tree topologies, as they represent a fundamental structure for convergecast data collection. We optimally solved the problem for $t = 2$ by finding a scheduling with minimum latency and minimum frame length in $O(n \log \Delta)$ time. For $t \geq 3$ we designed an approximate solution that runs in $O(nt)$ time, where n is the number of the clusters in the WSN. We further designed a distributed version of the proposed scheduling algorithm. Simulation results on cluster-based WSNs demonstrated that the obtained approximation factors are accurate.

Acknowledgments This work has been partially supported by the US National Science Foundation (NSF) under grant CNS-1049652, by the Academy of Finland under Grant No. 284806, and by the Italian Ministry of University and Research under Grant 2010N5K7EB—PRIN 2010 ARS TechnoMedia (Algoritmica per le Reti Sociali Tecno-mediate). Any opinions, findings and conclusions or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the NSF.

References

1. Willig, A. (2008). Recent and emerging topics in wireless industrial communications: A selection. *Industrial Informatics, IEEE Transactions on*, *4*(2), 102–124.

2. Augustine, J., Han, Q., Loden, P., Lodha, S., & Roy, S. (2013). Tight analysis of shortest path convergecat in wireless sensor networks. *International Journal of Foundations of Computer Science*, *24*(1), 31–50.
3. Madden, S. R., Franklin, M. J., Hellerstein, J. M., & Hong, W. (2005). TinyDB: An acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems*, *30*, 122–173.
4. Anastasi, G., Conti, M., Di Francesco, M., & Passarella, A. (2009). Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks*, *7*(3), 537–568.
5. The ZigBee Specification version 2.0. (2006).
6. Di Francesco, M., Anastasi, G., Conti, M., Das, S. K., & Neri, V. (2011). Reliability and energy-efficiency in IEEE 802.15.4/ZigBee sensor networks: A cross-layer approach. *Selected Areas in Communications, IEEE Journal on*, *29*(8), 1508–1524.
7. Pan, M.-S., & Liu, P.-L. (2014). Low latency scheduling for convergecast in ZigBee tree-based wireless sensor networks. *Journal of Network and Computer Applications*, *46*, 252–263.
8. Pan, M.-S., Liu, P.-L., & Lin, Yen Pei. (2014). Event data collection in ZigBee tree-based wireless sensor networks. *Computer Networks*, *73*, 142–153.
9. Keshavarzian, A., Lee, H., & Venkatraman, L. (2006). Wakeup scheduling in wireless sensor networks. In *Proceedings of MobiHoc 2006* (pp. 322–333).
10. Anastasi, G., Conti, M., Di Francesco, M., & Neri, V. (2010). Reliability and energy efficiency in multi-hop IEEE 802.15.4/ZigBee wireless sensor networks. In *Proceeding of the 15th IEEE Symposium on Computers and Communications (ISCC 2010)* (pp. 336–341).
11. Avin, C., Emek, Y., Kantor, E., Lotker, Z., Peleg, D., & Roditty, L. (2012). SINR diagrams: Convexity and its applications in wireless networks. *Journal of the ACM*, *59*(4), 18.
12. Keeler, H., & Blaszczyszyn, B. (2014). SINR in wireless networks and the two-parameter Poisson-Dirichlet process. *Wireless Communications Letters, IEEE*, *3*(5), 525–528.
13. Incel, O. D., Ghosh, A., & Krishnamachari, B. (2011). Scheduling algorithms for tree-based data collection in wireless sensor networks. In *Theoretical aspects of distributed computing in sensor networks* (pp. 407–445). Springer.
14. Bermond, J.-C., & Yu, M.-L. (2010). Optimal gathering algorithms in multi-hop radio tree-networks with interferences. *Ad Hoc and Sensor Wireless Networks*, *9*(1–2), 109–127.
15. Ergen, S. C., & Varaiya, P. (2010). TDMA scheduling algorithms for wireless sensor networks. *Wireless Networks*, *16*(4), 985–997.
16. Gandhi, R., Kim, Y.-A., Lee, S., Ryu, J., & Wan, P.-J. (2012). Approximation algorithms for data broadcast in wireless networks. *IEEE Transactions on Mobile Computing*, *11*(7), 1237–1248.
17. Jiao, X., Lou, W., Ma, J., Cao, J., Wang, X., & Zhou, X. (2012). Minimum latency broadcast scheduling in duty-cycled multihop wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, *23*(1), 110–117.
18. Shah, K., Di Francesco, M., Anastasi, G., & Kumar, M. (2011). A framework for resource-aware data accumulation in sparse wireless sensor networks. *Computer Communications*, *34*(17), 2094–2103.
19. Bertossi, A., Pinotti, C. M., & Tan, R. B. (2003). Channel assignment with separation for interference avoidance in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, *14*(3), 222–235.
20. Florens, C., & McEliece, R. (2003). Packets distribution algorithms for sensor networks. In *Proceeding of INFOCOM 2003*, (pp. 1063–1072).
21. Choi, H., Wang, J., & Hughes, E. A. (2009). Scheduling for information gathering on sensor network. *Wireless Networks*, *15*(1), 127–140.
22. Gandham, S., Zhang, Y., & Huang, Q. (2008). Distributed time-optimal scheduling for convergecast in wireless sensor networks. *Computer Networks*, *52*(3), 610–629.
23. IEEE 802.15.4, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). (2006).
24. Chen, X., Hu, X., & Zhu, J. (2009). Data gathering schedule for minimal aggregation time in wireless sensor networks. *International Journal of Distribution Sensor Networks*, *5*(4), 321–337.
25. Wan, P.-J., Huang, S. C.-H., Wang, L., Wan, Z., & Jia, X. (2009). Minimum-latency aggregation scheduling in multihop wireless networks. In *Proceedings of the 10th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)* (pp. 185–194). New York, USA: ACM.
26. Xu, X., Li, X. Y., Mao, X., Tang, S., & Wang, S. (2011). A delay-efficient algorithm for data aggregation in multihop wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, *22*(1), 163–175.
27. Gagnon, J., & Narayanan, L. (2015). Minimum latency aggregation scheduling in wireless sensor networks. In *Proceedings of the 10th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (Algosensors 2014)* (pp. 152–168). Berlin, Heidelberg: Springer.
28. Pan, M.-S., & Tseng, Y.-C. (2008). Quick convergecast in ZigBee beacon-enabled tree-based wireless sensor networks. *Computer Communications*, *31*(5), 999–1011.
29. Ghosh, A., Incel, O. D., Kumar, V. S. A., & Krishnamachari, B. (2011). Multi-channel scheduling and spanning trees: Throughput-delay trade-off for fast data collection in sensor networks. *IEEE/ACM Transactions on Networking*, *19*(6), 1731–1744.
30. Di Francesco, M., Pinotti, C. M., & Das, S. K. (2012). Interference-free scheduling with bounded delay in cluster-tree wireless sensor networks. In *The 15th International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM 2012)* (pp. 99–106).
31. Navarra, A., Pinotti, M. C., & Formisano, A. (2012). Distributed colorings for collision-free routing in sink-centric sensor networks. *Journal of Discrete Algorithms*, *14*, 232–247.
32. Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness* (1st ed.). San Francisco: W. H Freeman.
33. Bertossi, A., Pinotti, C. M., & Rizzi, R. (2003). Channel assignment on strongly-simplicial graphs. In *Proceedings of IPDPS 2003*.
34. Santoro, N. (2007). *Design and analysis of distributed algorithms*. Hoboken, NJ: Wiley & Sons Inc.
35. Lynch, N. A. (1996). *Distributed algorithms*. San Francisco, CA: Morgan Kaufmann.

Alfredo Navarra is Associate Professor since 2015 at the Mathematics and Computer Science Dept., University of Perugia, Italy. He received his Ph.D. in Computer Science in 2004 from "Sapienza" University of Rome. Before joining the University of Perugia in 2007 as Assistant Professor, he has been with various international research institutes like the INRIA of Sophia Antipolis, France; the Dept. of Computer Science at the Univ. of L'Aquila, Italy; the LaBRI, Univ. of Bordeaux, France. He has co-authored more than 100 publications in high quality international journals, book chapters and conference proceedings. His research interests include algorithms, computational complexity, distributed computing and networking.

Cristina M. Pinotti received the Dr. degree cum laude in Computer Science from the University of Pisa, Italy, in 1986. From 1987–1999, she was a researcher with the National Council of Research in Pisa. From 2000–2003, she was an associate professor at the University of Trento. Since 2004, she is a full professor at the University of Perugia. Her current research interests include sensor networks, wireless networks, design and analysis of combinatorial algorithms. She has published more than 50 refereed papers on international journals, and more than 50 papers in international conferences, workshops, and book chapters.

Mario Di Francesco received his Ph.D. degree in Information Engineering from the University of Pisa, Italy, in 2009. He is an Assistant Professor with the Department of Computer Science of Aalto University, Finland. His research interests include wireless sensor networks, pervasive computing, mobile networking, and performance evaluation. Dr. Di Francesco has been in the technical program committee of several international conferences related to wireless communications and pervasive computing, including the International Conference on Distributed Computing Systems (ICDCS), the IEEE International Conference on Pervasive Computing and Communications (PerCom), and the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM). He has been Co-Chair for the third workshop on the IoT: Smart Objects and Services (IoT-SoS 2014) and Guest Editor for the Special Issue on the Internet of Things of Pervasive and Mobile Computing.

Sajal K. Das an IEEE Fellow, is currently the Chair of Computer Science Department and Daniel St. Clair Endowed Chair in Computer Science at the Missouri University of Science and Technology, Rolla. Prior to 2013 he was a Distinguished Scholar Professor of Computer Science and Engineering and Founding Director of the Center for Research in Wireless Mobility and Networking (CReWMaN) at the University of Texas at Arlington. During 2008–2011, he served the US National Science Foundation as a Program Director in the division of Computer Networks and Systems. His current research interests include wireless and sensor networks, mobile and pervasive computing, cyber-physical systems including smart grid and smart healthcare, security and privacy, distributed and cloud computing, biological and social networks, applied graph theory and game theory. Dr. Das has published more than 600 technical papers and 51 invited book chapters in these areas, and coauthored three books. His h-index is 65 with more than 17,500 citations according to Google Scholar. He holds five US patents and received ten Best Paper Awards in prestigious conferences like ACM MobiCom, IEEE PerCom and IEEE SmrtGridComm. He is also a recipient of numerous awards for teaching, mentoring and research including the IEEE Computer Society Technical Achievement Award for pioneering contributions to sensor networks and mobile computing. Dr. Das serves as the Editor in Chief of Pervasive and Mobile Computing journal, and as Associate Editor of IEEE Transactions on Mobile Computing, ACM Transactions on Sensor Networks, Journal of Parallel and Distributed Computing, and Journal of Peer to Peer Networking and Applications. He is a co-founder of IEEE WoWMoM, IEEE PerCom, and ICDCN conferences.