Discrete Mathematics and Theoretical Computer Science DMTCS vol. **13:1**, 2011, 95–112

Deterministic Recurrent Communication and Synchronization in Restricted Sensor Networks

Antonio Fernández Anta[1†] Miguel A. Mosteiro[2,3‡] Christopher Thraves[3,4§]

[1] *Institute IMDEA Networks, Spain*
[2] *Department of Computer Science, Rutgers University, USA*
[3] *LADyR, GSyC, Universidad Rey Juan Carlos, Spain*
[4] *ASAP Project team, IRISA/INRIA Rennes, France*

received 24th September 2010, revised 23rd February 2011, accepted 15th April 2011.

Monitoring physical phenomena in Sensor Networks requires guaranteeing permanent communication between nodes. Moreover, in an effective implementation of such infrastructure, the delay between any two consecutive communications should be minimized. The problem is challenging because, in a restricted Sensor Network, the communication is carried out through a single and shared radio channel without collision detection. Dealing with collisions is crucial to ensure effective communication between nodes. Additionally, minimizing them yields energy consumption minimization, given that sensing and computational costs in terms of energy are negligible with respect to radio communication. In this work, we present a deterministic recurrent-communication protocol for Sensor Networks. After an initial negotiation phase of the access pattern to the channel, each node running this protocol reaches a steady state, which is asymptotically optimal in terms of time efficiency, and optimal (0) or constant (for a worst-case adversary) in terms of transmissions overhead, which we use as energy efficiency metric. As a by-product, a protocol for the synchronization of a Sensor Network is also proposed. Furthermore, the protocols are resilient to an arbitrary node power-up schedule and a general node failure model.

Keywords: Sensor networks, Recurrent communication, Synchronization, Radio networks, Distributed computing.

1 Introduction

A Sensor Network is an infrastructure deployed in a hostile or remote area for monitoring purposes. The basic entities of a Sensor Network are called *sensor nodes*, small devices provided with radio-communication, processing, and sensing capabilities. Upon being distributed at random in the area of interest, sensor nodes have to build a communication system from scratch. A strong shortcoming in Sensor Networks is the energy supply of sensor nodes. Consequently, one of the main challenges is the efficient administration of such resource, extending the usability of the network. In sensor nodes, sensing and

†Email: `antonio.fernandez@imdea.org`.
‡Email: `mosteiro@cs.rutgers.edu`.
§Email: `cbthraves@gmail.com`.

A preliminary version of this work is included in Fernández Anta et al. (2010a).

This research was supported in part by NSF grant no. 0937829, Comunidad de Madrid grant S2009TIC-1692 and Spanish MICINN grant TIN2008–06735-C02-01, and French ANR project Shaman. The authors would like to thank the anonymous referees for their useful comments and suggestions.

computational costs in terms of energy consumption are negligible with respect to radio communication. Thus, it is crucial to optimize the communication schedule. In a harshly restricted Sensor Network, the communication is carried out by means of a single and shared radio channel where nodes may broadcast messages to all neighboring nodes but no collision detection mechanism is available. Therefore, special mechanisms to effectively transmit or receive a message are required. Indeed, a node b receives a message transmitted from a neighboring node a only if neither b nor the other neighbors of b transmit at the same time. Otherwise, a collision occurs and the messages are garbled. Furthermore, b is not able to recognize the difference between this garbled message received and the background noise present in the channel if no transmission is produced.

The mechanism used by a node to decide to transmit or receive at any time is called the *transmission schedule*. Some transmission schedules use randomness to avoid collisions, but frequently involve a large number of redundant transmissions, consequently incurring in excessive energy consumption. On the other hand, deterministic transmission schedules, although efficient in terms of energy consumption, usually provide only large time guarantees for successful communication. Therefore, the problem addressed in this work, i.e., to find a deterministic transmission schedule with optimal time and energy guarantees of successful communication, is a fundamental question in Sensor Networks.

The rest of the document is organized as follows. In Sections 2, the model and problem definition are presented. In Section 3, our results are presented and contextualized with the previous results. Section 4 contains synchronization algorithms of independent interest. Deterministic recurrent communication algorithms are introduced in Section 5. Finally, conclusions are presented in section 6.

2 Model and Problems Definition

Regarding network topology and connectivity, and node constraints, we use the restrictive model in Fernández Anta et al. (2007), Fernández Anta et al. (2009), and Farach-Colton et al. (2009) summarized here as follows.

NETWORK AND NODES: Let us denote with V the set of sensors. Each sensor node is assumed to have a unique identification number (ID) in $\{0, \ldots, n-1\}$. Sensors are expected to be deployed at random in the area of interest, which is represented by the 2-dimensional Euclidean space $\mathcal{R}^2$ together with their Euclidean metric. Each sensor is provided with a radio system to communicate with the rest of the network, but each radio system has only a limited range for transmissions and receptions. It is assumed that the transmission range and the reception range are the same, and it is referred as the communication range. Consequently, each node is able to communicate with a restricted number of other sensors, the ones deployed within its communication range. In this work, we use an undirected graph $G = (V, E)$ to model the topology of the network. Each node in V represents a sensor node, and the link $(u, v) \in E$ represents that nodes u and v are in communication range[(i)]. Let us denote with $N(v)$ the set of neighbors of node v. Let $n = |V|$ denote the number of nodes in the network, and let $\Delta = \max_{v \in V} |N(v)|$ be the maximum degree of a node in G (i.e., the network). Finally, we use D to denote the diameter of the network. Unless otherwise stated, we assume that n, Δ and D are known by all the nodes in the system. (We assume the precise values are known for clarity, but limiting that knowledge to asymptotically tight upper bounds yield the same results asymptotically.) Regarding computational resources of sensor nodes, node-memory size is restricted only to $O(\Delta + \log n)$ bits. Were the deployment of nodes uniform (random geometric

(i) This model corresponds to a *Geometric Graph*. Generalizing the results to arbitrary graphs is left for future work.

graph) as it is popularly assumed in the Sensor Networks literature (cf. Doherty et al. (2001) and Song et al. (2005)), our protocols would work even if the node-memory size is restricted to just $O(\log n)$ bits.

LOCAL SYNCHRONY: Time is assumed to be slotted in equal-length *time slots* or *steps*. It is assumed that the length of a slot is sufficient to transmit one message, i.e., each transmission occurs in a given slot. As in other Radio Networks work, we assume that the slots of all nodes are in phase, i.e., they all start and finish at the same time instants. For convenience, we assume a *global time* that takes non-negative integer values and advances one unit per step. Note that this is a virtual device and that the nodes do not have access to its value. For convenience we assume that the global time is the number of time steps since the first nodes in the system have been powered up or *awakened*. We assume the availability of a hardware clock mechanism at each node, denoted local-clock, such that, starting from 0 when the node is awakened, the clock is incremented by one automatically at the end of each time slot[(ii)]. Then, for all $i \in V$ and $t \in \mathbb{Z}^+$, $\mathsf{local\text{-}clock}_i(t)$ denotes the value of local-clock of node i at time step t before being incremented. In the first step t executed by a node i, $\mathsf{local\text{-}clock}_i(t) = 0$.

NODE AWAKENING AND TYPES OF ADVERSARIES: Nodes are in two possible states, *sleeping* and *awake*. It is assumed that initially all nodes are sleeping. The nodes are assumed to be awakened by an adversary.[(iii)] Without loss of generality, it is assumed that every node of the network is eventually awakened. In the rest of the paper x will be used to denote the first node awakened by the adversary, breaking ties arbitrarily. As x is always awake (see below), $\forall t \geq 0 : \mathsf{local\text{-}clock}_x(t) = t$. Regarding node reliability, as customary in the Sensor Networks literature, we assume that nodes may fail. I.e., a node may crash and stop working. The adversary decides when to crash and recover (awake again) nodes. However, if crashes and recoveries occur arbitrarily, due to determinism, there exist topologies for which the adversary may stop a node from receiving any message, even if connectivity is required.[(iv)] Thus, limitations to the crash/recovery schedule are in order. In this work, we consider node failures as long as: (i) node x is always awake (in fact it would be enough if there is always some node that has the global time up and running), (ii) a node that is awakened at time t is awake and connected (possibly indirectly) to node x at least the whole period $[t, t + S]$, where S is the length of the stabilization time (as defined below). Failure models where multi-hop communication is provided along time as links become available (as in *opportunistic networking* Pelusi et al. (2006) or *delay-tolerant networks* Fall (2003)) are also feasible for the stabilization phase of the present problem. The extension of the present analysis to those models is left for future work. In this work, we consider two types of adversaries.

Definition 1 *A* τ-adversary *is an adversary that awakens all the nodes of the network within a window time of size τ, i.e., no node is awakened at a time $t \geq \tau$. Additionally, a τ-adversary does not recover crashed nodes. The parameter τ is assumed known by the nodes.*

Definition 2 *An* ∞-adversary *is an adversary that has no restriction on when nodes are awakened.*

COMMUNICATION: Each radio system transmits and receives in a single and shared radio channel. Therefore, at each step, each node decides between transmission mode or reception mode. Moreover, node v *receives* from node u in a slot if and only if node u is the only neighbor of v transmitting in that slot, and v is in reception mode at that slot. In the case that two or more neighbors of node v transmit

(ii) Observe that, if not readily available, the described mechanism can be implemented as a software counter.

(iii) In contrast with the wake-up problem studied in the literature, we do not assume that sleeping nodes may additionally be awaken by the transmission of a neighboring node.

(iv) For any time slot t, if none or more than one neighbor transmit, do nothing. Otherwise, put the transmitter to sleep during t.

in the same slot a collision occurs at node v. A node v is not able to distinguish between silence (none of the nodes in $N(v)$ transmits) and collision. We denote the communication range as r. A customary assumption in Sensor Networks (cf. Doherty et al. (2001) and Song et al. (2005)) is that nodes can adjust the power of transmission to a smaller level. Such an assumption introduces only a constant factor in the number of nodes that have to be deployed to maintain connectivity, as long as the reduction factor is constant. The following geometric argument shows why. A circle of radius r can be inscribed in a square of side $2r$. For any $c > 0$, a square of side $2r$ can be completely covered by c^2 squares of side $2r/c$ laid down as a grid. A square of side $2r/c$ can be inscribed in a circle of radius $\sqrt{2}r/c$. Hence, to guarantee connectivity, a constant reduction in the radius can be compensated with a constant increase in the node density. For the sake of clarity, in this work we assume instead that nodes can duplicate their transmission range to $2r$. Such an assumption does not yield an extra asymptotic cost due to the same argument. Notice that, independently of this assumption, the maximum degree Δ and diameter D defined before correspond to the underlying graph G defined for range r.

Deterministic Recurrent Communication Problem The problem solved in this paper is called *deterministic recurrent communication*. The goal in solving this problem is to provide a communication service that can be used by the components of a distributed application residing in different nodes to exchange *application messages*. Thus, the service must allow a component in a node to recurrently communicate with the components in neighboring nodes. For the sake of clarity, we assume that all nodes run application components that have an infinite supply of application messages to transmit.

Definition 3 *A distributed protocol solves the* deterministic recurrent communication *(DRC) problem if it guarantees that, for every step t and every pair $(u, v) \in E$, there is some step $t' \geq t$ such that, in step t', v receives an application message from u.*

The protocols proposed in this paper are adaptive, in the sense that when nodes are awakened, they run a *start-up phase*. During this phase, nodes use *control messages* to agree on a periodic transmission schedule. After the start-up phase, a *stable phase* starts in which they use the agreed transmission schedule to exchange application messages. For some of the protocols, control messages still have to be used in the stable phase. We use three goodness parameters to evaluate these protocols. The first one is the maximum number of steps of the start-up phase for any node, called the *stabilization time*. Then, we define the following metrics to evaluate energy and time efficiency *in the stable phase*. For any $(u, v) \in E$ and any $i > 1$, let $M_u^i(v)$ be the number of transmissions of u between the $(i-1)^{th}$ and the i^{th} receptions of application messages from u at v, and $M_u(v) = \sup_i M_u^i(v)$. In order to measure time we denote $R_u^i(v)$ the time (number of time slots) that are between the $(i-1)^{th}$ and the i^{th} receptions of application messages from u at v, and $R_u(v) = \sup_i R_u^i(v)$. We define the *delay* of a protocol for DRC as $\max_{(u,v)\in E} R_u(v)$. We define the *transmissions-overhead rate* of a protocol for DRC as $\max_{(u,v)\in E} M_u(v)$. In words, with respect to any pair of nodes $\{u, v\}$, the transmissions-overhead rate is the maximum (throughout the network) of the number of transmissions from u that v either does not receive due to collisions or does not use for the application (control messages), between any consecutive application messages that v does receive from u.

Oblivious Protocol: We say that a deterministic protocol is *oblivious* if whether a node u is in transmission or reception mode at step t is a function only of u's ID and $\mathsf{local\text{-}clock}_u(t)$ (the number of steps that u has been awake). I.e., we consider that a protocol is oblivious if the computation to decide whether a node u is in transmission or reception mode is oblivious with respect to external information to u. Such an oblivious deterministic protocol that solves DRC exists, an example is the communication protocol proposed in Fernández Anta et al. (2007). In the rest of the paper, the oblivious deterministic recurrent communication protocol that solves DRC will be modeled as a binary function $\texttt{ORC}$ on $V \times \mathbb{Z}^+$. Then, for all $u \in V$ and all $j \in \mathbb{Z}^+$ we have $\texttt{ORC}(u, j) \in \{transmit, receive\}$. Since $\texttt{ORC}$ solves DRC, then $\texttt{ORC}$ provides a time delay such that every pair of nodes connected by a link in the network communicate without collision with each other within this delay. The delay of this protocol will be denoted by T. Since oblivious protocols have no start-up phase, this means that if nodes u and v, such that $(u, v) \in E$, are awake and run $\texttt{ORC}$, in every interval of T steps they will receive from each other.

In this paper, the goal is to derive protocols that solve DRC with asymptotically optimal delay and optimal or constant transmissions-overhead rate, even if they incur in significant stabilization times. We design our protocols assuming the existence of an *oblivious deterministic recurrent communication protocol* that solves DRC with bounded delay and no start-up phase.

The Synchronization Problem As a by-product of the protocols proposed in this paper for DRC, we propose also deterministic protocols that solve the *synchronization problem* under both classes of adversaries defined. In the synchronization problem it is assumed that each node has a slot counter $\textsf{global-clock}$ (incremented in every step) and a Boolean variable $\textsf{synced}$ indicating whether it is synchronized or not. The slot counters of all synchronized nodes must have the same value. For each node $i \in V$ and time slot t, let $\textsf{global-clock}_i(t)$ and $\textsf{synced}_i(t)$ be, respectively, the slot counter and the Boolean variable of node i at the beginning of time slot t. We say that a network is *synchronized* at a time step $t \in \mathbb{Z}^+$ if, for all $i, j \in V$, such that $\textsf{synced}_i(t) = \textsf{synced}_j(t) = true$, it holds that $\textsf{global-clock}_i(t) = \textsf{global-clock}_j(t)$. We say that a node $i \in V$ is *synchronized* at time step $t \in \mathbb{Z}^+$ if the network is synchronized at time step t and $\textsf{synced}_i(t) = true$.

Definition 4 *We say that a protocol solves the* synchronization problem *if there exists a time t from which the protocol guarantees that the network is synchronized at all times after* t*, and for any node that awakes at time* t'*, there exists a time* $t'' \geq t'$ *when the node is synchronized. The maximum time between a node awaking and getting synchronized is the* synchronization time *of the protocol.*

In the synchronization protocols proposed here each node initializes its counter $\textsf{global-clock}$ to 0 and increments it by 1 every step. A node can also adopt a larger $\textsf{global-clock}$ value from another node. Then, since x is the first node awake and it never fails, it will always have the largest $\textsf{global-clock}$ counter, i.e., for each node $u \in V$ and each $t \geq 0$, if u is awake at time t then $\textsf{global-clock}_u(t) \leq \textsf{global-clock}_x(t)$. Moreover, $\forall t \geq 0 : \textsf{local-clock}_x(t) = \textsf{global-clock}_x(t) = t$.

3 Framing Our Results with Related Work

To the best of our knowledge, deterministic recurrent communication under a restricted Sensor Network model was only studied in Fernández Anta et al. (2007) and later improved in Fernández Anta et al. (2009). It was shown in the latter an oblivious protocol with optimal transmissions-overhead rate and delay at most $\Delta(n + \Delta)(\ln(n + \Delta) + \ln \ln(n + \Delta))$, which was shown to be optimal delay-wise for a subclass of non-adaptive protocols for most values of Δ. For adaptive protocols, it was shown in that work a delay of $O(\Delta^2 \log \Delta)$ relaxing memory size constraints and an asymptotically optimal delay of $O(\Delta)$ additionally limiting the adversarial node awakening schedule. In the present paper, a worst-case asymptotically optimal $O(\Delta)$ delay bound is proven, even removing those restrictions.

The question of how to disseminate information in Radio Networks has led to different well-studied important problems such as *Broadcast* (cf. Bar-Yehuda et al. (1992), Kushilevitz and Mansour (1998), and Dessmark and Pelc (2007)), *Selection* (cf. Kowalski (2005)), and *Gossiping* (cf. Liu and Prabhakaran (2002) and Chlebus et al. (2001)). These problems differ in the number of nodes that hold a possibly different message to disseminate to all nodes in the network. Although these are one-shot communication primitives, some of the results obtained could be used repeatedly to achieve recurrent communication.

Deterministic solutions for Broadcast and Gossiping (cf. Chrobak et al. (2000) and Czumaj and Rytter (2003)) include assumptions such as simultaneous startup or the availability of a global clock, which are not feasible in Sensor Networks. The selection problem, on the other hand, was studied in Kowalski (2005) under a model where the node awakening schedule is adversarial, proving the existence of a $O(\Delta^2 \log n)$ algorithm and showing constructively how to obtain an algorithm that achieves $O(\Delta^2 \operatorname{polylog} n)$. These results are obtained for a model where nodes turn off upon successful transmission. Thus, they do not apply to our setting.

In Alon et al. (1992), the authors show a deterministic distributed protocol to simulate the message passing model in radio networks. Using this technique, each node receives a transmission of all its neighbors after $O(\Delta^2 \log^2 n / \log(\Delta \log n))$ steps. Again, simultaneous awakening of nodes is required, a feature that can not be assumed in restricted models of Sensor Networks. In the same paper, lower bounds for this problem are also proved by showing bipartite graphs that require $\Omega(\Delta \log \Delta)$ rounds. Bipartite graphs with maximum degree $\omega(1)$ are not embeddable in geometric graphs therefore these bounds do not apply to our setting.

Related lines of work from combinatorics include *selectors*, *selective-* and *strongly-selective families* (cf. Indyk (2002), Clementi et al. (2001), Dyachkov and Rykov (1983), Bonis et al. (2003), and Chlebus and Kowalski (2005)). The application of any of these combinatorial objects to recurrent communication in Radio Networks would require simultaneous awakening of the participating nodes. Within the scope of the *wake-up* problem, the existence of a combinatorial structure called *radio-synchronizer* was shown in Chrobak et al. (2004), later explicited in Chlebus and Kowalski (2004). The existence of an extension of radio-synchronizers, called *universal-synchronizers*, was also shown in the latter, and a constructive proof of universal-synchronizers was given in Chlebus et al. (2005). In Radio Networks terminology, a radio-synchronizer is an n-set of schedules of transmissions (one for each node) such that, for any node awakening schedule and for any subset of Δ nodes, there is a time step when exactly one of the Δ nodes transmits. Synchronizers (radio- or universal-) are of the utmost importance in Radio Networks because they tolerate arbitrary rotations of each schedule of transmissions. In other words, they can be used obliviously without assuming any specific node awakening schedule, as opposed to the present work where we assume an initial "stable" phase each time a node is activated. Furthermore, due to the same reason, synchronizers could be used repeatedly to implement a recurrent communication primitive. An extension of synchronizers that guarantees a successful transmission within certain interval forever was studied in Chlebus et al. (2006) with the name of *transmitters*. Transmitters and synchronizers can be used as a communication primitive as long as it is enough for each node to receive messages from *some* neighboring node infinitely many times. In the present paper, we study a recurrent communication primitive that requires each node to receive from *each* neighboring node infinitely many times. (See Definition 3.)

In order to compute a transmission schedule that solves DRC with asymptotically optimal delay bound, we include in the algorithms presented in this paper a synchronizing phase. Within the scope of Radio Networks, the problem of globally synchronizing the network has been recently studied in Dolev et al.

(2009), but their model includes a single-hop network and many channels of communication.

The application of Radio Network wake-up protocols to global synchronization was studied in Chrobak et al. (2004), Chlebus and Kowalski (2004), and Chlebus et al. (2005). In their model, nodes may be awaken adversarially, but additionally they may be also awaken by the transmission of another node. The synchronization technique proposed takes advantage of the latter and works only after all nodes have been awaken. Thus, it can only be applied to our setting under a τ-adversary, adding an initial waiting phase to ensure that all nodes are awake before running that protocol. Extending the best running time obtained in Chlebus et al. (2005) by the additional τ waiting steps gives $O(\tau+\min\{n, D\Delta\}\Delta \text{ polylog } n)$. Whereas the synchronization algorithm of Gouda and Herman (1990), suited here for a τ-adversary and using the `ORC` protocol of Fernández Anta et al. (2007), yields a running time of $\tau + Dn\Delta \log n$. Although which of these protocols is more efficient depends on the parameters instance, we propose the latter for clarity of the presentation towards the more general adversary.

A study of the impact of using collision detection in wireless networks is presented in Schneider and Wattenhofer (2010). The authors compare lower and upper bounds for algorithms with and without collision detection. They use as comparison criterion three different problems in the context of wireless networks, Maximal Independent Set, Broadcasting, and Coloring. Schneider and Wattenhofer (2010) prove that the benefit of using collision detection in the time complexity depends on the task that is performed. Particularly, on one extreme the authors show that in the case of coloring problem there is no asymptotic gain at all with using collision detection. In the opposite extreme, the authors present a deterministic algorithm that solves broadcasting problem in time $O(D \log n)$, which is an exponential improvement over prior work for some particular cases of D.

A different line of research, but related with synchronization in distributed systems, is the line that studies Gradient Clock Synchronization (GCS) problem. GCS problem search for the minimum worst-case clock skew between neighboring nodes. Locher and Wattenhofer (2006) studied oblivious clock synchronization algorithms. They prove that, when oblivious algorithms are used, the clock skew between neighboring nodes can be significantly larger than the proven lower bound. On the other hand, they present an oblivious clock synchronization algorithm with worst-case skew of $O(d+\sqrt{D})$ between any two nodes at distance d, where D denotes the diameter of the network. The same problem was studied in a dynamic setting by Kuhn et al. (2010). In their work, dynamics comes from the fact that communication links can appear and disappear at any time. The authors show that, as far as two nodes remain at the same distance for $O(D)$ time, then bounds for the static case apply to the dynamic case. For instance, if two nodes remain at distance d from each other for $O(D)$ time, then it is possible to synchronize their clocks to within $O(d \log(D/d))$. Since, that value is known to be the optimal for the static case, then it is also the optimal for the dynamic case. The authors also prove that the $O(D)$ time required for their algorithm to achieve the best possible clock synchronization is the smallest possible. Nevertheless, GCS problem and their bounds do not apply in our framework because in our setting better conditions are assumed. Therefore, whereas in GCS problem it is possible to synchronize clocks of two nodes at distance d to within $O(d \log(D/d))$, in our setting the whole network is synchronized.

Czyzowicz et al. (2009) and Bienkowski et al. (2010) study the consensus and the mutual exclusion problems under several multiple access channel (MAC) models. These MAC models differ on whether or not the nodes can detect collisions, have a global clock, or know the total number of nodes. They show that, if none of these features is available, the mutual exclusion problem can not be solved even using randomization (Bienkowski et al. (2010)), and that consensus cannot be solved deterministically (Czyzowicz et al. (2009)). Then, they show that if any of these features is available, both problems can be solved, and explore the complexity of the solutions. The main difference between the models used in these papers and the model assumed here is that they assume a single hop network. A review of more related literature can be found in Fernández Anta et al. (2010b).

Our Results In this work, we present an adaptive protocol that solves DRC with asymptotically optimal delay, and optimal or constant transmissions-overhead rate depending on the wake-up adversary. We model the arbitrary node awakening schedule, and node failures, with the two types of adversaries previously defined, τ-adversary and ∞-adversary. As a building block of our deterministic computation of an optimal transmission schedule, we include a synchronization algorithm for each of type of adversary. Once nodes are synchronized, we provide a $27(\Delta+1)$ coloring of the network, where Δ is the maximum degree with communication range r. Thus, the transmission schedule guarantees, in the case of the first adversary, that for every time interval of length $27(\Delta+1)$ slots, each node has at least one successful transmission to all its neighbors. In the case of the less restricted ∞-adversary, the transmission schedule has to be resilient to the awakening of new nodes. Thus, after synchronization, each time step is doubled extending the length of that interval to $54(\Delta+1)$ slots. Due to the pigeonhole principle, these delays between reception of each neighboring node are asymptotically optimal.

Given that the efficient use of energy is crucial to extend the life-cycle of a sensor node, and that the radio-communication cost in terms of energy dominates other consumption factors, it is extremely important to minimize the number of transmissions produced that do not achieve effective application communication. The protocols presented in this paper are shown to have optimal transmissions-overhead rate of 0 for the restricted adversary. (Recall that we measure the energy efficiency in terms of transmissions produced between consecutive successful receptions of application messages. Hence, when no transmission is lost or use for control messages, the transmissions-overhead rate is 0.) For the unrestricted adversary, the protocol presented achieves a transmissions-overhead rate of $27(\Delta+1)/n \in O(1)$.

4 Synchronization Protocols

In this section, we present the protocol that solves the synchronization problem under an ∞-adversary. For a τ-adversary, the synchronization protocol used is a re-creation of the algorithm presented in Gouda and Herman (1990). The interested reader may find the details in Fernández Anta et al. (2010b).

We present now the protocol CONTMAXSPREAD, designed to solve the synchronization problem against an ∞-adversary. Observe that, due to the nature of an ∞-adversary, any synchronization protocol has to keep sending synchronization messages during all its execution, even after the network has been synchronized. In this way, any new node awakened after the network is synchronized recognizes this fact, and joins the network adopting the common value of the global clock.

Hence, the synchronization protocol CONTMAXSPREAD has two phases, a *synchronization phase,* and an *application phase.* In the synchronization phase, the largest global-clock is spread through the network. However, as mentioned above, CONTMAXSPREAD keeps sending synchronization messages in the application phase. The protocol CONTMAXSPREAD sets up the synchronization flag synced to communicate the current synchronization state of the network (from a node's point of view). Roughly speaking, during the first $T_1 = 3n^2 + 2nT$ [v] steps of the synchronization phase, a node listens for messages from the network. That listening part is devoted to provide the node with the current synchronization state of the network. If the network is synchronized, and some node has synced $= true$, when the node wakes up, it

[v] Recall that T is the delay of an oblivious protocol used as a subroutine.

will know about it before the listening period is over and, without having to send any message will get synchronized. If that does not happen, during the next $T_2 = 2nT$ steps of the synchronization phase, the node transmits to its neighbors its value global-clock and its synchronization flag synced following ORC. As will be shown, at the end of this subphase the network (and hence the node) has to be synchronized. During the application phase, a node transmits its value global-clock and its synchronization flag synced (perhaps, piggybacked in an application message), but this time, the transmission is done in a round robin fashion, i.e., if the identifier of the node is equal to the value global-clock modulo n, then the node transmits. More details of the CONTMAXSPREAD protocol are presented in Algorithm 1.

Algorithm 1: CONTMAXSPREAD pseudocode for node v. local-clock is v's hardware clock, $T_1 = 3n^2 + T_2$, $T_2 = 2nT$, and T is the delay of the ORC protocol.

initialization
    set global-clock to 0
    set synced to false
    start tasks 1 and 2 concurrently

task *1*
    `// synch phase`
    while global-clock $< T_1 + T_2$ *and* synced $=$ *false once for each time slot* **do**
        if ORC$(v,$ local-clock$) = transmit$ *and* global-clock $\geq T_1$ **then**
            transmit (global-clock,synced) with radius r
        increment global-clock
    set synced to true
    stop task 2
    `// application phase`
    foreach *time slot* **do**
        if global-clock $= v$ **then**
            transmit (global-clock,synced) with radius r
        increment global-clock $\mod n$

task *2*
    upon *reception of* (global-clock$'$, synced$'$) *from other node* **do**
        if *not* synced **then**
            set synced to synced'
            set global-clock to $\max\{$global-clock, global-clock$'\}$

Lemma 1 *The global clock of a node u awakened before $T_1 + T_2$ satisfies that either* global-clock$_u(T_1 + T_2) \leq T_2$*, or* global-clock$_u(T_1 + T_2) =$ global-clock$_x(T_1 + T_2) = T_1 + T_2$.

Proof: Recall that the adversary is restricted so that once a node is awakened, it is awake and connected to x (recall that x is the first node awake) for at least the stabilization time. This means that, up to time $T_1 + T_2$, if node $u \in V$ is awake, there exists some *time ordered path* $x = v_0, v_1, \ldots, v_l = u$ in the network connecting x to u such that $l < n$ and, for all $0 < i \leq l$, local-clock$_{v_{i-1}}(t) \geq$ local-clock$_{v_i}(t)$. We call the *distance* from u to x as the smallest number of edges of any of these time ordered paths. Since all the time steps, and hence global clocks, considered in this proof are smaller than $T_1 + T_2$, then no node fails, and all awake nodes are in the synchronization phase of the algorithm, have a time ordered path to x, and have synced $= false$.

We show that a node u that at time T_1+T_2 has a global clock different from x's must have global-clock$_u(T_1+T_2) \leq T_2$. Let us consider a node u awakened before $T_1 + T_2$ and whose global clock at that time is global-clock$_u(T_1+T_2) < T_1+T_2$. A node that is awakened before T_1 and whose distance to x is d has the same global clock as x by time $T_1 + d \cdot T$, as formalized in the following claim.

Claim 1 *A node u that is awakened before T_1 and whose distance to x is d has the same global clock as x by time $T_1 + d \cdot T$, i.e., global-clock$_u(T_1 + d \cdot T)$ = global-clock$_x(T_1 + d \cdot T)$.*

Proof: From the condition of Line 7, at time T_1 node x starts transmitting with the schedule defined by ORC, and will do so for at least $T_2 = 2nT$ steps (from the condition of Line 6). Consider a time ordered path $x = v_0, v_1, \ldots, v_d = u$ from x to u. Since the delay of ORC is T, node v_1 will receive from x before step $T_1 + T$. When that happens, v_1 updates its global-clock variable as shown in Lines 18 to 20. Hence, global-clock$_{v_1}(T_1 + T)$ = global-clock$_x(T_1 + T)$. Then, from the condition of Line 7, at time T_1+T node v_1 starts transmitting with the schedule defined by ORC, and will do so for at least $(2n-1)T$ steps (from the condition of Line 6). A simple induction shows that node v_i, $1 < i \leq d$, will satisfy global-clock$_{v_i}(T_1 + iT)$ = global-clock$_x(T_1 + iT)$. The claim follows since $v_d = u$. □

Thus, given that the distance is always less than n, a node awakened before T_1 has the the same global clock as x by time $T_1 + nT \leq T_1 + T_2$. Therefore, to complete the proof, it remains to consider the case where u was awakened at some time within the global-time interval $[T_1, T_1 + T_2)$. To prove the claim in that case, it is enough to prove that global-clock$_u(T_1 + T_2)$ = local-clock$_u(T_1 + T_2)$ because, given that u did not wake up before T_1, it holds that local-clock$_u(T_1 + T_2) \leq T_2$.

Let us assume, by way of contradiction, that u has global-clock$_u(T_1 + T_2) \neq$ local-clock$_u(T_1 + T_2)$. This means that u has received a message before time T_1+T_2 with a field global-clock$'$ larger than its own global clock, and has adopted it. The value global-clock$'$ is received because some node v had global-clock$_v(t)$ = local-clock$_v(t) \geq T_1$ at some time $t < T_1 + T_2$, and transmitted this value (using schedule ORC). Let us denote the propagation path (not necessarily time ordered) of this value before reaching u as $v = q_0, q_1, ..., q_s = u$.

From $t < T_1 + T_2$, local-clock$_v(t) \geq T_1$, and $T_1 > T_2$, it is derived that v was awakened before T_1. Let d be the distance from v to x. Then, node v has the same global clock as x by time $T_1 + d \cdot T$, as shown in the claim above. Then, by time $T_1 + d \cdot T$ v transmits, using the ORC schedule, the same global clock as x. Furthermore, it does this for at least $(2n - d)T$ steps.

Returning to the path $v = q_0, q_1, ..., q_s = u$, we have that q_1 must have received (and hence was awake) some message from $v = q_0$ (in particular, the global clock that was later propagated as global-clock$'$) before $T_1 + d \cdot T$. Furthermore, q_1 has received from v a message with the global clock of x by $T_1 + (d+1)T$. Applying the same argument, we conclude that q_2 received (and hence was awake) from q_1 before $T_1 + (d+1)T$ and has received the global clock of x by time $T_1 + (d+2)T$. Inductively, $q_s = u$ has received the global clock of x by time $T_1 + (d+s)T$. Since $d + s < 2n$, this mean that u has the same global clock as x by time $T_1 + T_2$, which is a contradiction. □

Theorem 1 CONTMAXSPREAD *solves the synchronization problem under any ∞-adversary with synchronization time $T_1 + T_2$, where $T_1 = 3n^2 + 2nT$ and $T_2 = 2nT$.*

Proof (Sketch of the proof): To prove the theorem, it is enough to prove that, at any global time step $t \geq T_1 + T_2$, any node in the network is either synchronized with x, or it is still in the listening part of the synchronization phase.

Claim 2 *For any node $v \in V$ and any time step $t \geq T_1 + T_2$, if v is continuously connected to x, it takes at most $3n^2$ time steps for v to have the global time (be synchronized), even under failures (as defined in Section 2), unless v goes back to sleep before.*

Proof: Consider a node v that is awake in $[t, t+3n^2)$ but does not have the global time at time $t \geq T_1+T_2$. From the condition that v is connected to x, until v becomes synchronized, within each consecutive $3n$ steps a new node must be synchronized: at most n steps to synchronize some non-synchronized node, at most n steps when those newly synchronized nodes may maintain connectivity without synchronizing other nodes, and, if all the synchronized nodes that are neighbors of non-synchronized nodes fail before synchronizing any of them, at most n further steps to synchronize at least one new node that has to awake to maintain connectivity, which is possible because at least one synchronized (x) node must stay awake. Due to failures, the same node may be re-synchronized, but not more than twice during the period $[t, t + T_1 + T_2)$ due to the stable-time requirement, and the above cost already accounts for both synchronizations. Thus, the claim follows. □

Lemma 1 shows that at global time $T_1 + T_2$ a global-clock in the network is either synchronized with x's global clock, or its value is smaller than T_2, which is smaller than T_1 by $3n^2$ time steps. Since the stabilization time is $T_1 + T_2$, from the failure model (as defined in Section 2), a node whose global clock is smaller than T_2 has been awake no more than T_2 time steps, and will be awake and connected to x at least $3n^2$ additional time steps. Consequently, from Claim 2, at global time $T_1 + T_2$, every node who is transmitting messages does it with x's global clock. Then, any node with global clock smaller than T_2 receive x's global clock before its own global clock reaches the value T_1. Finally, due to the same reason, if a node is awakened at global time $t \geq T_1 + T_2$, before its local clock reaches the value $3n^2$, it receives a message with x's global clock. Then, that node is synchronized without transmitting itself in the synchronization phase. □

5 Communication Scheme

In this section, we show how to solve DRC for τ- and ∞- adversaries. Both protocols are algorithmically similar and can be broadly described for each node $v \in V$ as follows. Upon waking up, v runs three phases: synchronization, coloring, and application. During the first phase, v synchronizes itself (as defined in Section 2) with the node x that woke up first in the network. During the second phase, v chooses a color that has not been chosen by any neighboring node. Finally, by mapping colors to time slots (thanks to the global synchronization achieved), the application phase of i corresponds to its stable phase. Given that the color chosen by v is unique within radius $2r$ of v, but the application messages are transmitted with radius r, all nodes within distance r of v receive v's application messages.

Moving to how do we implement each phase, synchronization is implemented using the protocols of Section 4 for the τ- and ∞- adversaries respectively. The coloring phase, on the other hand, is implemented by each node announcing the color chosen so that, by appropriate bookkeeping of the available colors at each node, nodes within distance $2r$ do not choose the same color (avoiding the hidden-terminal

Algorithm 2: DRC pseudocode for node v under τ-adversary. local-clock is the hardware node clock. T is the delay of the ORC protocol.

```
initialization
    set global-clock to 0
    set color to null
    set synced to false
    set available-colors to {0, 1, ..., 27(Δ + 1) − 1}
    start tasks 1 and 2 concurrently
task 1
    // synchronization phase
    while global-clock < D · T + τ and synced = false once for each time slot do
        if ORC(v, local-clock) = transmit and global-clock ≥ τ then
            transmit (global-clock,color,synced) with radius 2r
        increment global-clock
    set synced to true
    // coloring phase
    while global-clock < D · T + τ + n once for each time slot do
        if global-clock ≡ v(mod n) then
            set color to one of the available-colors
            transmit (global-clock,color,synced) with radius 2r
        increment global-clock
    stop task 2
    // application phase
    foreach time slot do
        if global-clock = color then
            transmit (application-message) with radius r
        increment global-clock mod 27(Δ + 1)
task 2
    upon reception of (global-clock′, color′, synced′) from other node do
        set available-colors to available-colors − {color′}
        if not synced then
            set synced to synced’
            set global-clock to max{global-clock, global-clock′}
```

problem). To avoid collision of transmissions and simultaneous choice of the same color, taking advantage of the global synchronization achieved in the previous phase, each colored node chooses an available color and announces its choice in a time slot selected in a round-robin fashion according to ID. For the application phase, again thanks to the global synchronization achieved, each node transmits its application messages in a round-robin fashion, but now according to its color. (We refer curious readers to Clementi et al. (2004), where the authors prove that round-robin is optimal for fault tolerant broadcasting in wireless networks.)

For the τ-adversary (see Algorithm 2), thanks to the inclusion of a τ-long waiting period at the beginning of the synchronization phase, the above described phases are executed synchronously by all nodes in the network. In other words, all nodes in the network finish the synchronization (resp. coloring) phase and begin the coloring (resp. application) phase at the same time. For the ∞-adversary (see Algorithm 3) on the other hand, new nodes may be woken up while others are already in the coloring or application phases. Thus, control messages have to be sent always to handle these late arrivals. The transmissions corresponding to those control messages are produced in a round-robin fashion according to ID. The co-existence of both types of messages during the application phase is handled by devoting even slots (w.r.t. global time) to control messages and odd slots (w.r.t. global time) to application messages, at the cost of duplicating the time delay. The slots for control messages, let us call them *control slots*, are reserved one for each of the n nodes in sequence. Likewise, the slots for application messages, let us call them *application slots*, are reserved for each of the $27(\Delta + 1)$ colors in sequence. Given that those two classes of slots are interleaved, a complete sequence of control slots, one for each node, takes $2n$ time slots, let us call this quantity t_c. Likewise, a complete sequence of application slots, one for each color, takes $54(\Delta + 1)$ time slots, let us call this quantity t_a. A global-time counter increased in each round of communication is used to synchronize these slots, but it can not be increased forever due to physical limitations. So, it has to be periodically reset. However, in order to guarantee the delay claimed, it can be reset only synchronically with the above described sequences of control and application slots, which is guaranteed resetting the counter modulo $t_c t_a = 108(\Delta + 1)n$.

Regarding the space complexity of these protocols, a node needs to store its own ID ($O(\log n)$ bits) and after the coloring phase one of $O(\Delta)$ colors ($O(\log \Delta)$ bits), where Δ is the maximum degree of the network with communication range r. Additionally, each node has to keep track of the colors still available ($O(\Delta)$ bits), and maintains a counter that reaches a maximum count in $O(\Delta n)$ ($O(\log n)$ bits). Thus, the overall space complexity for each node is $O(\Delta + \log n)$ bits. The stabilization time, delay, and transmissions-overhead rate of the protocols described can be proved applying the results of Section 4 and standard analysis of the round-robin algorithms used. We establish those bounds in the following theorems.

Theorem 2 *Given a Sensor Network of n nodes, the protocol presented in Section 5 solves the DRC problem under a τ-adversary with stabilization time at most $D \cdot T + \tau + n$, where T is the delay of the* `ORC` *protocol. The delay of this DRC protocol is $27(\Delta + 1)$ which is asymptotically optimal, and the transmissions-overhead rate is 0 which is optimal.*

Proof: First, we show correctness and time efficiency of the stabilization phase. (Refer to Algorithm 2.) According to the failure model assumed (Section 2), nodes do not fail during the stabilization phase. Hence, correctness and time efficiency of the stabilization phase are studied in a scenario without failures. For any node, the synchronization phase lasts at most $D \cdot T + \tau$ steps. Furthermore, it was proved in Fernández Anta et al. (2010b) that within $D \cdot T + \tau$ steps after the first node is woken up, all nodes have been synchronized. I.e., all nodes have the same global-clock and synced values. Thus, all nodes finish the synchronization phase and start the coloring phase at the same time (when the condition in Line 8 of Algorithm 2 is false). After that, the coloring phase takes exactly n further steps (see Algorithm 2 Line 13) yielding a total stabilization time of $D \cdot T + \tau + n$ as claimed.

In order to prove that the coloring obtained is correct for the underlying graph G (where only edges of length at most r are included), consider a disk D_v of radius $2r$ centered in any node $v \in V$, and notice that 27 disks of radius $r/2$ are sufficient to completely cover D_v. (See Figure 1.)

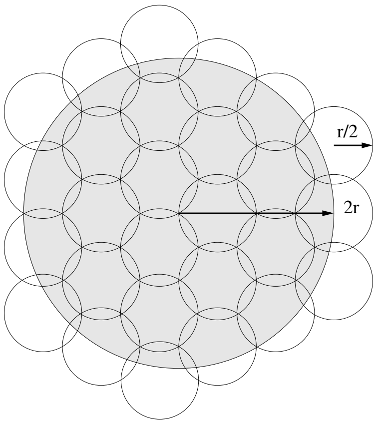

Fig. 1: Illustration of Theorem 2.

Given that Δ is the maximum degree of G, a disk of radius $r/2$ contains at most $\Delta + 1$ nodes. (Notice that for a disk of radius r this is not necessarily true.) Then, there are at most $27(\Delta+1)$ nodes in D_v. So, $27(\Delta+1)$ colors are enough to ensure that all nodes in D_v can choose a different color. Given that nodes choose a color in a round-robin fashion according to ID (thanks to the global synchronization achieved), after the n steps of the coloring phase all nodes have chosen a color.

Defining some ordering among the $27(\Delta + 1)$ colors, the coloring provided defines a transmission schedule such that every node receives a message from each of its neighboring nodes every $27(\Delta + 1)$ slots. Thus, the DRC problem is solved with delay $27(\Delta+1)-1$, which is asymptotically optimal given that, in the worst case, all nodes have Δ neighbors and must receive a message from each of them. Given that all transmissions of every node are received by all of its neighbors within radius r, the transmissions-overhead rate is 0, which is optimal. Finally, given that a τ-adversary cannot recover crashed nodes, then failures do not affect the recurrence on the communication protocol. □

Theorem 3 *Given a Sensor Network of n nodes, upon being woken up by a ∞-adversary, the protocol presented in Section 5 solves the DRC problem under an ∞-adversary with stabilization time at most $6n^2+4nT+4n$, where T is the delay of the ORC protocol. The delay of this DRC protocol is $54(\Delta+1) \in O(\Delta)$ and the transmissions-overhead rate is $27(\Delta+1)/n \in O(1)$.*

Proof: Again, due to the failure model assumed (Section 2), nodes do not fail during the stabilization phase, which in this case lasts until the coloring is finished. Hence in the first part of the proof, we consider a scenario without failures. For the sake of clarity, assume first that all nodes are woken up simultaneously. Then, the claim can be proved along the same lines of Theorem 2. (Refer to Algorithm 3.) For any node, the synchronization phase lasts at most $6n^2 + 4nT$ steps. Furthermore, it was proved in Theorem 1 that within $3n^2 + 4nT$ steps after a node is woken up (in this case $6n^2 + 4nT$ due to the duplication of slots after the synchronization phase), it becomes synchronized (w.r.t. global time). I.e. all nodes have the same global-clock and synced values. Thus, all nodes finish the synchronization phase and start the coloring phase at the same time (when the condition in Line 8 of Algorithm 3 is false). After that, the coloring phase takes at most $4n$ further steps (see Algorithm 3 Line 14), a waiting period of $2n$ steps (the reason

Algorithm 3: `DRC` pseudocode for node v under ∞-adversary. local-clock is the hardware node clock. T is the delay of the `ORC` protocol.

```
initialization
    set global-clock to 0
    set synced to false
    set color to null
    set available-colors to {0, 1, ..., 27(Δ + 1) − 1}
    start tasks 1 and 2 concurrently
task 1
    // synchronization phase
    while global-clock < 6n^2 + 4nT and synced = false once for each time slot do
        if ORC(v, local-clock) = transmit and global-clock ≥ 6n^2 + 2nT then
            transmit (global-clock,color,synced) with radius 2r
        increment global-clock
    set synced to true
    // coloring phase
    while color = null once for each time slot do
        if global-clock ≡ 2v(mod 2n) and global-clock ≥ 6n^2 + 4nT + 2n then
            set color to one of the available-colors
            transmit (global-clock,color,synced) with radius 2r
        increment global-clock
    stop task 2
    // application phase
    foreach time slot do
        if global-clock ≡ 2v(mod 2n) then
            transmit (global-clock,color,synced) with radius 2r
        if global-clock ≡ (2color + 1)(mod 54(Δ + 1)) then
            transmit (application-message) with radius r
        increment global-clock mod 108(Δ + 1)n
task 2
    upon reception of (global-clock′, color′, synced′) from other node do
        set available-colors to available-colors − {color′}
        if not synced then
            set synced to synced’
            set global-clock to max{global-clock, global-clock′}
```

for this will be clear soon) followed by a period of $2n$ steps when nodes choose colors in a round-robin fashion using only even steps (w.r.t. global time). Nodes leave the coloring phase as soon as they get colored, but the whole $2n$ steps are needed for the node with ID n. Thus, the total stabilization time is at most $6n^2 + 4nT + 4n$ as claimed.

The correctness of the coloring obtained can be proved as in Theorem 2. Defining some ordering among the $27(\Delta+1)$ colors, the coloring provided defines a transmission schedule. Given that only odd steps are used for application messages, every node receives a message from each of its neighboring nodes every $54(\Delta+1)$ slots. Thus, the delay of this protocol is $54(\Delta+1)-1$ as claimed. Regarding the transmissions-overhead rate, all transmissions of every node are received by all of its neighbors, i.e. there are no collisions. However, only the transmissions produced in the odd steps correspond to application messages. The transmission in even steps correspond to control messages. Then, for any node $v \in V$ running the application phase, the transmissions-overhead rate is the number of control messages transmitted by v between any consecutive application messages transmitted by v, which are the same for any node, hence, the maximum. The rate of transmissions of v due to control messages is $1/2n$ (see Algorithm 3). Whereas the rate of transmissions of v due to application messages is $1/(54(\Delta + 1))$. Then, the transmissions-overhead rate is $27(\Delta + 1)/n$, which is in $O(1)$ because $\Delta < n$.

To complete the proof, we consider now the impact of a node $v \in V$ that is woken up "late" with respect to the first node woken up in the network (global time). As shown in the proof of Theorem 1, v is globally-synchronized seamlessly (before v produces any transmission) thanks to the waiting period of $6n^2 + 2nT$ steps, as long as the already synchronized nodes keep transmitting the global time forever as done in Lines 10, 16, and 21 in Algorithm 3. Furthermore, it was proved in the same theorem that within at most $6n^2 + 4nT$ steps after v is woken up, it becomes synchronized (w.r.t. global time). After becoming synchronized, v will produce transmissions in a round-robin fashion according to the protocol avoiding any collisions. After entering the coloring phase, thanks to the initial waiting period of $2n$ steps, v updates properly its set of available colors after receiving transmissions of all synchronized neighbors within distance $2r$. Thus, after at most $4n$ steps v has been colored properly and has announced its color choice. Thus, the claimed stabilization time, delay, and transmissions-overhead rate also holds for v. Finally, notice that a node that fails and then rejoins the network can be considered as a node that is woken up "late". Consequently, failures do not affect the behavior of the communication protocol. □

6 Conclusions

In this work, the addressed problem is how to guarantee permanent communication between nodes in Sensor Networks. The proposed solution is a distributed deterministic protocol that works in two phases. The first phase provides the network with synchrony, while the second phase provides each node with a color, that determines its transmission schedule for its subsequent steady state. The protocol, in its strongest version, is designed such that it works no matter when nodes are awakened. Nodes that awake after the protocol started do not affect the progress already achieved by the protocol. Actually, those nodes, due to the design of the protocol, are able to recognize the present state of the computation and join the network seamlessly. It is proved that the introduced protocol achieves asymptotically optimal delay and constant transmission-overhead rate. Furthermore, since the proposed solution is deterministic, each node knows (or may know) its transmission and reception pattern. Therefore, every node can turn off their radio system when there is not a transmission or reception scheduled, and then achieve an efficient usage of energy.

Some interesting questions remain open for future work. On one hand, in this work, failures of nodes can be present only in the steady state of a node. Therefore, an open challenge is to develop a protocol that works under a stronger failure model. For instance, two open problems are, how to merge two networks already in their steady state, and whether it is possible to deal with a failure model that allows a node to fail at any time. On the other hand, although the introduced protocol guarantees asymptotically optimal permanent communication between nodes when the network is in the steady state, the time required to reach such state may be too long for some applications. Therefore, a second open challenge is how to improve the initial phase in order to reduce the stabilization time. In fact, it is important to know what is the minimum time required for a network to reach an steady state with (asymptotically) optimal permanent communication in terms of delay minimizing transmission-overhead rate.

References

N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. Single round simulation in radio networks. *J. Algorithms*, 13:188–210, 1992.

R. Bar-Yehuda, O. Goldreich, and A. Itai. On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization. *J. Comput. Syst. Sci.*, 45:104–126, 1992.

M. Bienkowski, M. Klonowski, M. Korzeniowski, and D. R. Kowalski. Dynamic sharing of a multiple access channel. In J.-Y. Marion and T. Schwentick, editors, *STACS*, volume 5 of *LIPIcs*, pages 83–94. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010. ISBN 978-3-939897-16-3.

A. D. Bonis, L. Gąsieniec, and U. Vaccaro. Generalized framework for selectors with applications in optimal group testing. In *Proc. of 30th Intl. Colloquium on Automata Languages and Programming*, pages 81–96, 2003.

B. Chlebus and D. Kowalski. A better wake-up in radio networks. In *Proc. 23rd Ann. ACM Symp. on Principles of Distributed Computing*, pages 266–274, 2004.

B. Chlebus and D. Kowalski. Almost optimal explicit selectors. In *Fundamentals of Computation Theory, 15th international symposium*, volume 3623 of *Lecture Notes in Computer Science*, pages 270–280. 2005.

B. Chlebus, L. Gąsieniec, A. Lingas, and A. Pagourtzis. Oblivious gossiping in ad-hoc radio networks. In *Proc. of the 5th Intl. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 44–51, 2001.

B. Chlebus, L. Gąsieniec, D. Kowalski, and T. Radzik. On the wake-up problem in radio networks. In *Proc. of 32nd Intl. Colloquium on Automata Languages and Programming*, pages 347–359, 2005.

B. S. Chlebus, D. R. Kowalski, and M. A. Rokicki. Adversarial queuing on the multiple-access channel. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing*, pages 92–101, 2006.

M. Chrobak, L. Gąsieniec, and W. Rytter. Fast broadcasting and gossiping in radio networks. In *Proc. of the 41st IEEE Ann. Symp. on Foundations of Computer Science*, pages 575–581, 2000.

M. Chrobak, L. Gąsieniec, and D. Kowalski. The wake-up problem in multi-hop radio networks. In *Proc. of the 15th Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 992–1000, 2004.

A. Clementi, A. Monti, and R. Silvestri. Selective families, superimposed codes, and broadcasting on unknown radio networks. In *Proc. of the 12th Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 709–718, 2001.

A. E. F. Clementi, A. Monti, and R. Silvestri. Round robin is optimal for fault-tolerant broadcasting on wireless networks. *J. Parallel Distrib. Comput.*, 64(1):89–96, 2004.

A. Czumaj and W. Rytter. Broadcasting algorithms in radio networks with unknown topology. In *Proc. of the 44th IEEE Ann. Symp. on Foundations of Computer Science*, pages 492–501, 2003.

J. Czyzowicz, L. Gasieniec, D. R. Kowalski, and A. Pelc. Consensus and mutual exclusion in a multiple access channel. In I. Keidar, editor, *DISC*, volume 5805 of *Lecture Notes in Computer Science*, pages 512–526. Springer, 2009. ISBN 978-3-642-04354-3.

A. Dessmark and A. Pelc. Broadcasting in geometric radio networks. *J. Discrete Algorithms*, 5(1):187–201, 2007.

L. Doherty, K. S. J. Pister, and L. E. Ghaoui. Convex optimization methods for sensor node position estimation. In *Proc. of the 20th IEEE International Conference on Computer Communications (INFOCOM)*, pages 1655–1663, 2001.

S. Dolev, S. Gilbert, R. Guerraoui, F. Kuhn, and C. Newport. The wireless synchronization problem. In *Proc. of the 28th Ann. ACM Symp. on Principles of Distributed Computing*, pages 190–199, 2009.

A. Dyachkov and V. Rykov. A survey of superimposed code theory. *Probl. Contr. Inform. Theor.*, 12(4), 1983.

K. Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, pages 27–34, New York, NY, USA, 2003. ACM. ISBN 1-58113-735-4. doi: http://doi.acm.org/10.1145/863955.863960. URL `http://doi.acm.org/10.1145/863955.863960`.

M. Farach-Colton, R. J. Fernandes, and M. A. Mosteiro. Bootstrapping a hop-optimal network in the weak sensor model. *ACM Trans. Alg.*, 5(4):1–30, 2009.

A. Fernández Anta, M. A. Mosteiro, and C. Thraves. Deterministic communication in the weak sensor model. In *Proc. 11th Intl. Conf. Principles of Distributed Systems*, pages 119–131, 2007.

A. Fernández Anta, M. A. Mosteiro, and C. Thraves. Deterministic recurrent communication in restricted sensor networks. INRIA RR 00486270, 2009.

A. Fernández Anta, M. A. Mosteiro, and C. Thraves. Deterministic recurrent communication and synchronization in restricted sensor networks. In *Proceedings of the 6th International Workshop on Algorithms for Sensor Systems, Wireless Ad Hoc Networks and Autonomous Mobile Entities (ALGOSENSORS)*, volume 6451 of *Lecture Notes in Computer Science*, pages 62–73. Springer-Verlag, 2010a.

A. Fernández Anta, M. A. Mosteiro, and C. Thraves. Deterministic recurrent communication and synchronization in restricted sensor networks. INRIA RR 00486277, 2010b.

M. G. Gouda and T. Herman. Stabilizing unison. *Information Processing Letters*, 35(4):171–175, 1990.

P. Indyk. Explicit constructions of selectors and related combinatorial structures, with applications. In *Proc. of the 13th Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 697–704, 2002.

D. R. Kowalski. On selection problem in radio networks. In *Proc. 24th Ann. ACM Symp. on Principles of Distributed Computing*, pages 158–166, 2005.

F. Kuhn, C. Lenzen, T. Locher, and R. Oshman. Optimal gradient clock synchronization in dynamic networks. In A. W. Richa and R. Guerraoui, editors, *PODC*, pages 430–439. ACM, 2010. ISBN 978-1-60558-888-9.

E. Kushilevitz and Y. Mansour. An $\Omega(D \log(N/D))$ lower bound for broadcast in radio networks. *SIAM J. Comput.*, 27(3):702–712, 1998.

D. Liu and M. Prabhakaran. On randomized broadcasting and gossiping in radio networks. In *Ann. Conference on Computing and Combinatorics*, pages 340–349, 2002.

T. Locher and R. Wattenhofer. Oblivious gradient clock synchronization. In S. Dolev, editor, *DISC*, volume 4167 of *Lecture Notes in Computer Science*, pages 520–533. Springer, 2006. ISBN 3-540-44624-9.

L. Pelusi, A. Passarella, and M. Conti. Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. *Communications Magazine, IEEE*, 44(11):134 –141, Nov. 2006. ISSN 0163-6804. doi: 10.1109/MCOM.2006.248176.

J. Schneider and R. Wattenhofer. What is the use of collision detection (in wireless networks)? In N. A. Lynch and A. A. Shvartsman, editors, *DISC*, volume 6343 of *Lecture Notes in Computer Science*, pages 133–147. Springer, 2010. ISBN 978-3-642-15762-2.

W.-Z. Song, Y. Wang, X.-Y. Li, and O. Frieder. Localized algorithms for energy efficient topology in wireless ad hoc networks. *MONET*, 10(6):911–923, 2005.