

Deterministic Recurrent Communication and Synchronization in Restricted Sensor Networks ^{*}

Antonio Fernández Anta^{1,2}, Miguel A. Mosteiro^{3,2}, and Christopher Thraves⁴

¹ Institute IMDEA Networks, Leganés, Spain

² LADyR, GSyC, Universidad Rey Juan Carlos, Móstoles, Madrid, Spain
`anto@gsyc.es`

³ Department of Computer Science, Rutgers University, Piscataway, NJ, USA
`mosteiro@cs.rutgers.edu`

⁴ ASAP Project team, IRISA/INRIA Rennes, Campus Universitaire de Beaulieu,
35043 Rennes Cedex, France.

Abstract. 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 energy and time efficiency. 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.

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

^{*} This research was partially supported by Spanish MICINN grant no. TIN2008-06735-C02-01, Comunidad de Madrid grant no. S2009TIC-1692, EU Marie Curie International Reintegration Grant IRG 210021, NSF grant no. 0937829, and French ANR project Shaman.

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 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 follow. 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. Finally, deterministic recurrent communication algorithms are introduced in Section 5.

2 Model and Problems Definition

Regarding network topology and connectivity, and node constraints, we use the restrictive model in [15, 16, 14] 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 $\{0, \dots, n - 1\}$. Sensors are expected to be deployed at random in the area of interest. 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

range⁵. 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 $k = \max_{v \in V} |N(v)|$ 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 , k 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 $O(k + \log n)$ bits. Were the deployment of nodes uniform (random geometric graph) as it is popularly assumed in the Sensor Networks literature [11, 24], 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. Without loss of generality [23], it is assumed 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 fictional 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 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 powered up, the clock is incremented by one automatically at the end of each time slot⁶. Then, for all $i \in V$ and $t \in \mathbb{Z}^+$, $\text{local-clock}_i(t)$ 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 , $\text{local-clock}_i(t) = 0$.

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.⁷ 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), $\forall t \geq 0 : \text{local-clock}_x(t) = t$. 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.⁸ Thus, limitations to the

⁵ This model corresponds to a *Geometric Graph*. Generalizing the results to arbitrary graphs is left for future work.

⁶ Observe that, if not readily available, the described mechanism can be implemented as a software counter.

⁷ In contrast with the wake-up problem studied in the literature, we do not assume that sleeping nodes may additionally be awakened by the transmission of a neighboring node.

⁸ For any time slot t , if none or more than one neighbor transmit, do nothing. Otherwise, put the transmitter to sleep during t .

crash/recovery schedule are in order. In this work, we consider node failures as long as: (i) the network stays connected (one connected component) at all times, (ii) 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), and (iii) each period when a node runs without failures lasts at least the length of the stabilization time (as defined in Section 2). 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 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 [11, 24] is that nodes can adjust the power of transmission to a smaller level, introducing only a constant factor in the number of nodes that has to be deployed to maintain connectivity. Instead, for the sake of clarity, in this work we assume that nodes can duplicate their transmission range to $2r$. Likewise, such an assumption does not yield an extra asymptotic cost. Notice that, independently of this assumption, the maximum degree k 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 $R_u^i(v)$ 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 $R_u(v) = \max_i R_u^i(v)$ ⁹. In order to measure time we denote $\Delta R_u^i(v)$ 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 $\Delta R_u(v) = \max_i \Delta R_u^i(v)$ ⁷. We define the *message complexity* of a protocol for DRC as $\max_{(u,v) \in E} R_u(v)$. We define the *delay* of a protocol for DRC as $\max_{(u,v) \in E} \Delta R_u(v)$.

In this paper, the goal is to derive protocols that solve DRC with asymptotically optimal message complexity and delay, 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. In this protocol, whether a node u is in transmission or reception mode at step t is a function only of u 's ID and $\text{local-clock}_u(t)$ (the number of steps u has been awake). Such oblivious deterministic protocols exist. An example is the Primed Selection communication protocol proposed in [15]. In the rest of the paper, the oblivious deterministic recurrent communication protocol will be modeled as a binary function ORC on $V \times \mathbb{Z}^+$. Then, for all $u \in V$ and all $j \in \mathbb{Z}^+$ we have $\text{ORC}(u, j) \in \{\text{transmit}, \text{receive}\}$. 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 ORC , in every interval of T steps they will receive from each other.

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 global-clock (incremented in every step) and a Boolean variable 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 $\text{global-clock}_i(t)$ and $\text{synced}_i(t)$ 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 $\text{synced}_i(t) = \text{synced}_j(t) = \text{true}$, it holds that $\text{global-clock}_i(t) = \text{global-clock}_j(t)$.

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 every node that awakes eventually gets synchronized. The maximum time between a node awaking and getting synchronized is the synchronization time of the protocol.*

⁹ If the maximum does not exist, this value is defined as ∞ .

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

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 [15] and later improved in [16]. It was shown in the latter an oblivious protocol with optimal message complexity and delay at most $k(n+k)(\ln(n+k) + \ln \ln(n+k))$, which was shown to be optimal delay-wise for a subclass of non-adaptive protocols for most values of k . For adaptive protocols, it was shown in that work a delay of $O(k^2 \log k)$ relaxing memory size constraints and an asymptotically optimal delay of $O(k)$ additionally limiting the adversarial node awakening schedule. In the present paper, a worst-case asymptotically optimal $O(k)$ 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* [2, 21], *Selection* [20], and *Gossiping* [22, 5]. 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 [8, 10] for Broadcast and Gossiping 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 by Kowalski [20] in a model where the node awakening schedule is adversarial, proving the existence of a $O(k^2 \log n)$ algorithm and showing constructively how to obtain an algorithm that achieves $O(k^2 \text{polylog } n)$. These results are obtained for a model where nodes turn off upon successful transmission. Thus, they do not apply to our setting.

In [1], Alon, Bar-Noy, Linial and Peleg gave 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 $O(k^2 \log^2 n / \log(k \log n))$ 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 $\Omega(k \log k)$ 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* [19, 9, 13, 3]. 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 [7], later explicited in [6]. 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 [4]. 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 k nodes, there is a time step when exactly one of the k 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. Furthermore, due to the same reason, synchronizers could be used repeatedly to implement a recurrent 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 [12], 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 [7, 6, 4]. In their model, nodes may be awoken adversarially, but additionally they may be also awoken by the transmission of another node. The synchronization technique proposed takes advantage of the latter and works only after all nodes have been awoken. 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 [4] by the additional τ waiting steps gives $O(\tau + \min\{n, Dk\}k \text{ polylog } n)$. Whereas the synchronization algorithm of [18], suited here for a τ -adversary and using the ORC protocol of [15], yields a running time of $\tau + Dnk \log n$. 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 more complete review of the related literature can be found in [17].

Our Results In this work, we present an adaptive protocol that solves DRC asymptotically optimally for message complexity and delay efficiency measures. 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 $19(k + 1)$ coloring of the network,

where k is the maximum degree¹⁰. Thus, the transmission schedule guarantees, in the case of the first adversary, that for every time interval of length $19(k+1)$ 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 $38(k+1)$ 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 message complexity of 0 for the restricted adversary, and a message complexity of $19(k+1)/n$ for the unrestricted adversary, which is asymptotically optimal if all nodes run application components that have an infinite supply of application messages to transmit.

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 [18]. The interested reader may find the details in [17].

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 $T_1 = 3n^2 + 2nT$ 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 will know about it before the listening period

¹⁰ Recall that k is the maximum degree with communication range r . The proposed protocols use also a communication range of $2r$. The value $19(k+1)$ is an upper bound on the maximum degree with this range.

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 can be found in [17].

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

Proof. Recall that the adversary is restricted so that the network is connected at all times and awake nodes are alive 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* $x = v_0, v_1, \dots, v_l = u$ (recall that x is the first node awake) in the network connecting x to u such that $l < n$ and, for all $0 < i \leq l$, $\text{local-clock}_{v_{i-1}}(t) \geq \text{local-clock}_{v_i}(t)$. 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 $\text{global-clock}_u(T_1 + T_2) \leq T_2$. Let us consider a node u awakened before $T_1 + T_2$ and whose global clock at that time is $\text{global-clock}_u(T_1 + T_2) < T_1 + T_2$. A node that is awakened before T_1 and whose distance to x is d has the same global clock as x by time $T_1 + d \cdot T$. (Broadcast time in a network of awakened nodes, see [17] for details.) 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 $\text{global-clock}_u(T_1 + T_2) = \text{local-clock}_u(T_1 + T_2)$ because, given that u did not wake up before T_1 , it holds that $\text{local-clock}_u(T_1 + T_2) \leq T_2$.

Let us assume, by way of contradiction, that u has $\text{global-clock}_u(T_1 + T_2) \neq \text{local-clock}_u(T_1 + T_2)$. 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 $\text{global-clock}_v(t) = \text{local-clock}_v(t) \geq T_1$ 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 $v = q_0, q_1, \dots, q_s = u$.

From $t < T_1 + T_2$, $\text{local-clock}_v(t) \geq T_1$, 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 $T_1 + d \cdot T$. 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 $v = q_0, q_1, \dots, q_s = u$, 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 means 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 (Proof sketch). 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. The proof of the following claim can be found in [17].

Claim. For any node $v \in V$ and any time step $t \geq T_1 + T_2$, 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.

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 $3n^2$ time steps smaller than T_1 . Consequently, 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 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 Round-robin fashion according to ID. For the application phase, again thanks to the global synchronization achieved, each node transmits its application messages in Round-robin fashion, but now according to its color.

For the τ -adversary, 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 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 Round-robin fashion according to ID. The coexistence 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 details of both algorithms can be found in [17].

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 $O(k)$ colors ($O(\log k)$ bits). Additionally, each node has to keep track of the colors still available ($O(k)$ bits), and maintains a counter that reach a maximum count in $O(kn)$ ($O(\log n)$ bits). Thus, the overall space complexity for each node is $O(k + \log n)$ bits. The stabilization time, delay, and message complexity 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. (The proofs can be found in [17].)

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 DRC protocol. The delay of this DRC protocol is $19(k+1)$ which is asymptotically optimal, and the message complexity is 0 which is optimal.*

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 $38(k + 1)$ and the message complexity is $19(k + 1)/n$, which are both asymptotically optimal.

References

1. N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. Single round simulation in radio networks. *J. Algorithms*, 13:188–210, 1992.
2. 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. *JCSS*, 45:104–126, 1992.
3. A. De Bonis, L. Gąsieniec, and U. Vaccaro. Generalized framework for selectors with applications in optimal group testing. In *ICALP*, pages 81–96, 2003.
4. B. Chlebus, L. Gąsieniec, D. Kowalski, and T. Radzik. On the wake-up problem in radio networks. In *ICALP*, 2005.
5. B. Chlebus, L. Gąsieniec, A. Lingas, and A. Pagourtzis. Oblivious gossiping in ad-hoc radio networks. In *DIAL-M*, pages 44–51, 2001.
6. B. Chlebus and D. Kowalski. A better wake-up in radio networks. In *PODC*, 2004.
7. M. Chrobak, L. Gąsieniec, and D. Kowalski. The wake-up problem in multi-hop radio networks. In *SODA*, 2004.
8. M. Chrobak, L. Gąsieniec, and W. Rytter. Fast broadcasting and gossiping in radio networks. In *FOCS*, 2000.
9. A. Clementi, A. Monti, and R. Silvestri. Selective families, superimposed codes, and broadcasting on unknown radio networks. In *SODA*, 2001.
10. A. Czumaj and W. Rytter. Broadcasting algorithms in radio networks with unknown topology. In *FOCS*, 2003.
11. L. Doherty, K. S. J. Pister, and L. El Ghaoui. Convex optimization methods for sensor node position estimation. In *INFOCOM*, pages 1655–1663, 2001.
12. S. Dolev, S. Gilbert, R. Guerraoui, F. Kuhn, and C. Newport. The wireless synchronization problem. In *PODC*, 2009.
13. A. Dyachkov and V. Rykov. A survey of superimposed code theory. *Probl. Contr. Inform. Theor.*, 12(4), 1983.
14. M. Farach-Colton, R. J. Fernandes, and M. A. Mosteiro. Bootstrapping a hop-optimal network in the weak sensor model. *ACM TALG*, 5(4):1–30, 2009.
15. A. Fernández Anta, M. A. Mosteiro, and C. Thraves. Deterministic communication in the weak sensor model. In *OPODIS*, pages 119–131, 2007.
16. A. Fernández Anta, M. A. Mosteiro, and C. Thraves. Deterministic recurrent communication in restricted sensor networks. INRIA RR 00486270, 2009.
17. A. Fernández Anta, M. A. Mosteiro, and C. Thraves. Deterministic recurrent communication and synchronization in restricted sensor networks. INRIA RR 00486277, 2010.
18. M. G. Gouda and T. Herman. Stabilizing unison. *IPL*, 35(4):171–175, 1990.
19. P. Indyk. Explicit constructions of selectors and related combinatorial structures, with applications. In *SODA*, 2002.
20. D. R. Kowalski. On selection problem in radio networks. In *PODC*, pages 158–166, 2005.
21. E. Kushilevitz and Y. Mansour. An $\Omega(D \log(N/D))$ lower bound for broadcast in radio networks. *SICOMP*, 27(3):702–712, 1998.
22. D. Liu and M. Prabhakaran. On randomized broadcasting and gossiping in radio networks. In *COCOON*, pages 340–349, 2002.

23. L. G. Roberts. Aloha packet system with and without slots and capture. *CCR*, 5(2):28–42, 1975.
24. 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.