Gossiping in Jail

Avery Miller
University of Toronto, Toronto ON, Canada
a4miller@cs.toronto.edu

Abstract

Consider a set of prisoners that want to gossip with one another, and suppose that these prisoners are located at fixed locations (e.g., in jail cells) along a corridor. Each prisoner has a way to broadcast messages (e.g. by voice or contraband radio) with transmission radius R and interference radius $R' \geq R$. We study synchronous algorithms for this problem (that is, prisoners are allowed to speak at regulated intervals) including two restricted subclasses. We prove exact upper and lower bounds on the gossiping completion time for all three classes. We demonstrate that each restriction placed on the algorithm results in decreasing performance.

1 Introduction

Gossiping is a fundamental task for any radio sensor network. Each processor p_i in the network has a message m_i that it wishes to share with all processors in the network. When each process terminates, it has all of the messages. Gossiping can be used for performing any aggregate computation that requires information from each processor, such as discovering the network topology or reaching consensus.

We define several natural classes of deterministic algorithms by placing restrictions on which processors transmit simultaneously. The most restrictive of these are the *singleton* algorithms, in which only one processor may transmit during each transmission slot. Such algorithms are of interest when there is a collection of processors whose physical distance from one another is at most the interference radius R'. In this case, allowing two or more processors to transmit simultaneously would result in a wasted transmission slot: all processors found within the transmission radius of one transmitting processor are also found within the interference radius of another transmitting processor, so no message is received by any processor. In *collision-free* algorithms, more than one processor is allowed to transmit during a single transmission slot, unless the transmissions result in a collision. One benefit of designing a collision-free algorithm is that each transmitted message is received by all neighbours of the processor that transmitted it. This can make the correctness proof and running-time analysis of the algorithm much easier. *Unrestricted* algorithms have no restrictions on which processors may transmit during a single transmission slot.

We restrict our attention to static networks located in physical environments that can be modelled in one dimension, for example: roadways, sidewalks, or corridors. We proceed by defining our formal model of such networks, and then derive exact upper and lower bounds on the time complexity of gossiping for each of the three classes defined above. Using these bounds, we will show that each restriction placed on the algorithm negatively affects performance for some networks. Full details and proofs can be found in [7].

2 Related Work

Gossiping in static ad hoc radio networks has been thoroughly studied in networks where the transmission radius of each processor's radio is equal to its interference radius. Most of the results in the literature assume that the transmission radius is not necessarily the same for all processors and that processors do not initially know the network topology.

The simplest gossiping algorithm consists of repeating the Round Robin protocol for D rounds, where D is the network diameter. During each round, each processor transmits during the time slot corresponding to its unique ID number. The time complexity of this singleton algorithm is quadratic (or worse if the ID numbers are from a large domain). The first subquadratic algorithm was proposed by Chrobak, Gąsieniec and Rytter [3]. They presented a technique to extend any broadcasting algorithm with any time complexity $f(n)$ to an algorithm for gossiping which has time complexity $O(n\sqrt{f(n)}\log n)$. Applying the technique to a constructive broadcasting algorithm with time complexity $O(n^{3/2})$ (from Chlebus et al. [1]), the authors proved an upper bound of $O(n^{7/4}\log n)$ for constructive gossiping algorithms. Gąsieniec, Radzik and Xin [6] presented a non-constructive algorithm that solves gossiping in time $O(n^{4/3}\log^4 n)$. Chlebus, Gąsieniec, Lingas and Pagourtzis [2] showed several upper and lower bounds on the time complexity of gossiping by non-adaptive algorithms. An algorithm is *non-adaptive* if processors only use their ID and the total number of processors for deciding when to transmit. In contrast, an *adaptive* algorithm can use other information, such as detected collisions or the contents of received messages, to decide when processors will transmit. The authors gave a singleton algorithm that completes the gossiping task in ad hoc radio networks in time $(n-1)(n-2)+4$. They showed that there exists a family of networks for which any singleton algorithm takes at least $n^2 - O(n^{7/4+\epsilon})$ steps, for any $\epsilon > 0$. Further, they gave a non-constructive proof that there exists a non-adaptive deterministic algorithm for gossiping that completes in time $n^2 - \omega(n)$. They also proved a lower bound of $n^2/2 - O(n)$ time slots for any deterministic non-adaptive algorithm. Their lower bounds were proven for networks that are modelled by arbitrary neighbourhood graphs, that is, they do not restrict to networks that exist in a physical space. Real-world solutions to the gossiping task may not be subject to these lower bounds.

For networks in which the transmission radius is the same for all processors, Chlebus, Gąsieniec, Lingas and Pagourtzis [2] demonstrated a deterministic non-adaptive algorithm that completes the gossiping task in time $O(n^{3/2})$. When all processors are aware of the network topology, Gąsieniec, Potapov, and Xin [5] have provided a deterministic algorithm for the gossiping task that uses at most n time slots. The best known algorithm, especially useful for networks with small diameter, is presented by Cicalese, Manne, and Xin [4]. Their algorithm completes in $O(D + \frac{\Delta \log n}{\log \Delta - \log\log n})$ time if no processor has more than Δ neighbours.

The gossiping task can be solved using a randomized algorithm in which, at each step, a small subset of processors is randomly chosen to exchange information with its neighbours. This technique, which can be used to solve various tasks in mobile networks, is, confusingly, also called gossiping. We restrict our attention to deterministic solutions to the gossiping task.

3 Model

A static ad hoc radio network consists of a set P of n processors $p_1, \ldots, p_n$, in order from left to right, arranged at fixed arbitrary locations in one-dimensional space. Each processor runs a locally-stored deterministic algorithm. We assume that time is slotted and that each processor knows when each slot begins. Each time slot is long enough so that each message is completely transmitted, even in the case of a large message that contains all of the m_i's. So we assume that each processor always transmits all of the gossiping messages that it knows. The topology of the network is known by all processors. This is not an unrealistic assumption for static networks, since one can first execute a gossiping algorithm that is designed to discover the network topology, and then subsequently use a gossiping algorithm that assumes that the topology is known. We assume that the network is *connected*. The *diameter* D of the network is the length of the shortest sequence of processor transmissions such that a message originating from p_1 reaches p_n.

During the course of an algorithm, processors may perform wireless transmissions by broadcasting messages in both directions. A transmitted message is assumed to reach only those points within *transmission radius* R of the transmitting processor's location. Processors found within the transmission radius of p are p's *neighbours*. The signal sent by a transmitting processor reaches all points within distance $R' \geq R$ of the transmitting processor's location. The parameter R' is called the *interference radius*, since a processor q that is found within distance R' of a transmitting processor p cannot, at the same time, receive a message from a transmitting processor p' that is within distance R due to the signals interfering with one another. The occurrence of such signal interference is known as a *collision*. For our upper bounds, we assume that a processor cannot detect collisions, whereas for our lower bounds, we allow processors to detect collisions. Hence, our bounds apply both in the presence or absence of collision detectors.

4 Singleton Algorithms

In any gossiping algorithm, each processor must transmit at least once, since a processor that never transmits can never share its message. This implies that any singleton algorithm requires at least n transmission slots. When $D = 1$, the processors transmitting in order of increasing index is a straightforward algorithm that uses exactly n slots. When each processor transmits, all other processors in the network receive the transmitted message. In the remainder of this section, we assume that $D \geq 2$, and prove that $n + D - 2$ transmission slots are necessary and sufficient for singleton algorithms solving the gossiping task.

We present a singleton algorithm, `SingletonGossip`, that completes the gossiping task in a connected network using at most $n+D-2$ transmission slots. When a processor transmits, it transmits all of the messages that it knows. First, all processors that are not neighbours of p_1 transmit in order from right to left. Note that, when p_{n-i} transmits, the set of messages $\{m_{n-i}, \ldots, m_n\}$ arrives at p_{n-i-1}. Let p_r be the rightmost neighbour of p_1. Next, each of the processors p_1,p_2,...,p_r transmit in order from left to right. Just before it transmits, p_r knows all of the messages, so all neighbours of

p_r (which include $p_1, \ldots, p_{r-1}$) receive all of the messages. Finally, the complete set of messages is sent rightwards to p_n as fast as possible: the processor to transmit in a given transmission slot is the rightmost processor that received a transmission in the previous transmission slot. The pseudocode for this protocol is listed in Figure 1.

```
r = label of rightmost neighbour of $p_1$
for k = n downto (r + 1)
        $p_k$ transmits
for k = 1 to r
        $p_k$ transmits
next = label of rightmost neighbour of $p_r$
while (next < n)
        $p_{next}$ transmits
        next = label of rightmost neighbour of $p_{next}$
```

Figure 1: Pseudocode for `SingletonGossip`

The `while` loop executes exactly $D - 2$ times, which leads to the following result.

Theorem 1. *The total number of slots used by* ***SingletonGossip*** *is* $n + D - 2$.

Next, we prove a matching lower bound for any singleton algorithm that solves the gossiping task.

Lemma 2. *At least* $n+D-2$ *transmission slots are required by any singleton algorithm for gossiping among* n *processors.*

Proof. For each processor p_i, let t_i be the transmission slot during which p_i transmits for the first time. Suppose $t_{last} = \max\{t_1, \ldots, t_n\}$. Since at most one processor may transmit per transmission slot, and every processor must transmit at least once, we have $t_{last} \geq n$.

Let P' be the sequence of transmissions that occur after the first transmission by p_{last}. After p_{last} transmits, message m_{last} reaches p_1 by some (possibly empty) subsequence of transmissions B of P' by processors to the left of p_{last}. Similarly, m_{last} reaches p_n by some (possibly empty) subsequence of transmissions C of P' by processors to the right of p_{last}. Note that B and C are disjoint, so $|B| + |C| \leq |P'|$. Let B' be obtained from sequence B by reversing the order of transmission. Then, performing the sequence $A = (p_1, B', p_{last}, C)$ results in m_1 reaching p_n. Hence, $|A| \geq D$ and $|P'| \geq |B'| + |C| = |A| - 2 \geq D - 2$. Therefore, at least $n + |P'| \geq n + D - 2$ transmission slots are used. □ □

5 Collision-Free Algorithms

We describe a collision-free algorithm, `CFGossip`, that solves the gossiping task. It divides the network into segments of size $R + R' + 1$. The processors in each segment transmit in order from left to right (as much as possible in parallel), and then send messages from each end of the network to the opposite end. The algorithm ensures that no collisions occur.

A subset P' of $D-1$ processors is designated as the set of *forwarding* processors, and these processors are not scheduled to transmit during the first phase of the algorithm. Consecutive elements of P' are at most distance R apart and every non-forwarding processor is within distance R from some processor in P'.

During the first phase, the non-forwarding processors in each segment transmit in order from left to right. The length of each segment is sufficiently large to ensure that transmitting processors in non-neighbouring segments can never cause a transmission collision. To avoid collisions between transmitting processors in neighbouring segments, we schedule processors found in even-numbered segments during even-numbered slots, and processors in odd-numbered segments during odd-numbered slots. Additional processors may transmit if collisions do not result.

During the second phase of the algorithm, the forwarding processors share the messages that they received during the first phase. More specifically, the messages received during phase 1 by the leftmost processor in P' are transmitted by processors in P', in order from left to right. These transmissions are called *right-transmissions.* Simultaneously, the messages received during phase 1 by the rightmost processor in P' get forwarded by processors in P', in order from right to left. These transmissions are called *left-transmissions.* These can be done in parallel, except when a right-transmission and a left-transmission will cause a collision. In this case, two slots are used. Since the message of each non-forwarding processor is received by some processor in P' during the first phase, the messages of all processors get collected and forwarded during this process.

Let $k \leq 2\lceil R'/R \rceil + 2$ be the smallest integer such that the i^{th} and $(i+k)^{th}$ processors in P' can transmit simultaneously without causing a collision, for $1 \leq i \leq D-1-k$. The first phase uses at most $n-D$ slots. Two processors transmit during each slot of the first and last $\lfloor (D-k)/2 \rfloor$ slots of the second phase. When the processors performing right- and left-transmissions are close to one another, $D - 2\lfloor (D-k)/2 \rfloor - 2$ right-transmissions are performed, followed by a single transmission which acts as both a right- and left-transmission, followed by $D-2\lfloor (D-k)/2 \rfloor - 2$ left-transmissions. In total, `CFGossip` algorithm uses at most $n + 2\lceil R'/R \rceil$ transmission slots.

By Lemma 2, in any network with $D > 2\lceil R'/R \rceil + 2$, a singleton algorithm requires at least $n + D - 2 > n + 2\lceil R'/R \rceil$ transmission slots. It follows that `CFGossip` is better than any singleton algorithm for the gossiping task in such a network.

Next, we find a lower bound for the number of transmission slots required by any collision-free gossiping algorithm. Consider a network N where the processors are equally spaced distance $R/2+\epsilon$ apart, where $0 < \epsilon < R/(4R'/R - 2) \leq R/2$. This network is connected, since the distance between consecutive processors is $R/2 + \epsilon < R$. Since $\epsilon > 0$, the distance between p_i and p_{i+2} is greater than R, which implies the following.

Proposition 3. *For any $i \in \{1, \ldots, n-1\}$, a transmission by p_i is received by p_{i+1}, but not by any processor to the right of p_{i+1}. Similarly, for any $i \in \{2, \ldots, n\}$, a transmission by p_i is received by p_{i-1}, but not by any processor to the left of p_{i-1}.*

From Proposition 3, it follows that the number of hops between two processors p_a and p_b is equal to $|b-a|$, so the diameter of the network N is $n-1$. The distance $d(p_a, p_b)$ between two processors p_a and p_b is $|b-a|(R/2+\epsilon)$. Using the fact that $\epsilon < R/(4R'/R-2)$, we show that a collision occurs (at processor p_{j+1}) if there exists $1 < \ell \leq 2\lfloor R'/R \rfloor$ such that p_j and $p_{j+\ell}$ transmit during the same slot.

Proposition 4. *If* $1 \leq \ell \leq 2\lfloor R'/R \rfloor$ *and* $1 \leq j \leq n-\ell$, *then* $d(p_{j+1}, p_{j+\ell}) \leq R'$.

Let B be a sequence of sets of collision-free transmissions, and let B_t be the prefix of B of length t (where B_0 is the empty sequence). We denote by $RM(B,i)$ the rightmost processor that knows m_i after executing the transmission sequence B. We define $LM(B,i)$ analogously as the leftmost processor that knows m_i after executing the transmission sequence B. Proposition 3 implies the following two facts.

Proposition 5. *Suppose that* $RM(B_t, 1) = p_i$. *Then,* $RM(B_{t+1}, 1) = p_{i+1}$, *if* p_i *transmits during transmission slot* $t+1$, *and* $RM(B_{t+1}, 1) = p_i$ *otherwise.*

Proposition 6. *Suppose that* $LM(B_t, n) = p_i$. *Then,* $LM(B_{t+1}, n) = p_{i-1}$ *if* p_i *transmits during transmission slot* $t+1$, *and* $LM(B_{t+1}, n) = p_i$ *otherwise.*

To produce the desired lower bound, we consider the total number of transmission slots required for both m_1 to reach p_n and m_n to reach p_1. As a measure of progress, we consider the minimum number of transmissions needed to send a message between the leftmost processor that has received m_n and the rightmost processor that has received m_1. More formally, for any transmission slot t, let $p_a = RM(B_t, 1)$ and $p_b = LM(B_t, n)$, and define $h(B_t) = b-a$. Then, $h(B_0) = n-1$, and, for any t after which gossiping is complete, $h(B_t) = 1-n$. From Propositions 5 and 6, it follows that, for any transmission slot t before gossiping is complete, $h(B_t) = h(B_{t-1})$ if and only if neither $RM(B_{t-1}, 1)$ nor $LM(B_{t-1}, n)$ transmit. Further, it follows that $h(B_t) = h(B_{t-1}) - 1$ if exactly one of $RM(B_{t-1}, 1)$ or $LM(B_{t-1}, n)$ transmits, and $h(B_t) = h(B_{t-1}) - 2$ if both $RM(B_{t-1}, 1)$ and $LM(B_{t-1}, n)$ transmit. Along with Proposition 4, we get the following useful result.

Lemma 7. *For every transmission slot* t, $h(B_t) \geq h(B_{t-1}) - 2$. *If* $0 < |h(B_{t-1})| \leq 2\lfloor R'/R \rfloor$, *then* $h(B_t) \geq h(B_{t-1}) - 1$.

Theorem 8. *In any collision-free gossiping algorithm, for* $n > 2\lfloor R'/R \rfloor$ *processors, at least* $2(\lceil n/2 \rceil + \lfloor R'/R \rfloor - 1)$ *transmission slots are required before* p_1 *knows* m_n *and* p_n *knows* m_1.

Proof. Let B be the sequence of sets of collision-free transmissions performed by the algorithm and let B' be obtained from B by removing the sets in which no progress is made. Let t_4 be the length of B'. Then $h(B'_0) = n-1$, $h(B'_{t_4}) = 1-n$, and $h(B'_0), h(B'_1), \ldots, h(B'_{t_4})$ is a decreasing sequence.

By Lemma 7, $h(B'_t) = h(B'_{t-1}) - 1$ if $0 < |h(B'_{t-1})| \leq 2\lfloor R'/R \rfloor$. It follows that there exist transmission slots $0 < t_1 < t_2 < t_3 \leq t_4$ such that $h(B'_{t_1}) = 2\lfloor R'/R \rfloor - 1$, $h(B'_{t_2}) = 0$, and $h(B'_{t_3}) = -2\lfloor R'/R \rfloor - 1$. Hence, $t_2 - t_1 = h(B'_{t_1}) - h(B'_{t_2})$ and $t_3 - (t_2+1) = h(B'_{t_2+1}) - h(B'_{t_3})$.

By Lemma 7, for all slots t, $h(B'_{t-1}) - h(B'_t) \leq 2$. Hence, $h(B'_0) - h(B'_{t_1}) \leq 2t_1$, $h(B'_{t_2+1}) \geq h(B'_{t_2}) - 2$, and $h(B'_{t_3}) - h(B'_{t_4}) \leq 2(t_4 - t_3)$. By rearranging, we get $t_1 \geq \lceil (h(B'_0) - h(B'_{t_1}))/2 \rceil$ and $t_4 - t_3 \geq \lceil (h(B'_{t_3}) - h(B'_{t_4}))/2 \rceil$.

Therefore, the length of B' is $t_4 = (t_4 - t_3) + (t_3 - (t_2+1)) + (t_2 - t_1) + t_1 + 1 \geq 2(\lceil n/2 \rceil + \lfloor R'/R \rfloor - 1)$.
□ □

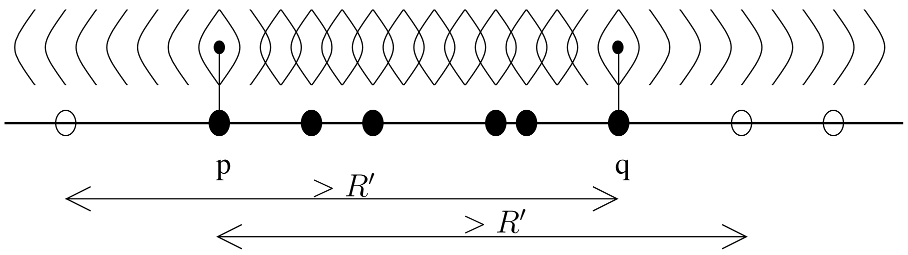

Figure 2: A situation where p and q are scheduled to transmit during the same slot despite causing transmission collisions

Our upper and lower bounds for gossiping by collision-free algorithms differ by at most 4.

6 Unrestricted Algorithms

We modify the `CFGossip` algorithm presented in Section 5 to create a more efficient gossiping algorithm called `Gossip`. The idea is to increase the number of parallel transmissions, while making sure that any processor affected by a transmission collision has already received the full set of messages to be gossiped. The first phase of `Gossip` is identical to the first phase of `CFGossip`. The second phase of `Gossip` is very similar to the second phase of `CFGossip`: the messages received during phase 1 by the leftmost processor in P' get forwarded by processors in P', in order from left to right, towards the rightmost processor in P' via right-transmissions. Simultaneously, the messages received during phase 1 by the rightmost processor in P' get forwarded by processors in P', in order from right to left, towards the leftmost processor in P' via left-transmissions. However, in the case of `Gossip`, after these messages 'cross', there are more slots in which exactly two processors transmit. More specifically, consider a processor p performing a right-transmission and a processor q performing a left-transmission such that any processor the right of q is more than distance R' from p and that any processor to the left of p is more than distance R' from q. We can show that all processors located between p and q have already received the full set of messages to be gossiped. Our `Gossip` algorithm schedules p and q to transmit during the same slot even if collisions occur, since these collisions do not hinder the completion of gossiping. This situation is depicted in Figure 2. The processors represented by filled circles have already received all of the gossiping messages.

Our `Gossip` algorithm uses at most $n+2\lceil R'/R\rceil-1$ transmission slots. The analysis of `Gossip` is similar to that of `CFGossip`, but with special attention paid to the slots after the single transmission which acts as both a right- and left-transmission. A more careful analysis shows an upper bound of $n+2\lceil R'/R\rceil-2$ for networks in which $D+k$ is even, for example, the network N in section 5, when the number of processors is odd and R' is a multiple of R. For this network, Theorem 8 implies that any collision-free gossiping algorithm requires at least $n+2\lceil R'/R\rceil-1$ transmission slots, and, thus, `Gossip` performs better than any collision-free algorithm.

The proof of the following result is similar to the proof of Theorem 8.

Theorem 9. *In any gossiping algorithm, for $n > 2\lfloor R'/R\rfloor$ processors, at least $2\lceil n/2\rceil+2\lfloor R'/R\rfloor-3$ transmission slots are required before p_1 knows m_n and p_n knows m_1.*

The upper bound for `Gossip` differs from the lower bound in Theorem 9 by at most 4.

7 Conclusion

All of the gossiping algorithms in the literature assume that the transmission radius R is equal to the interference radius R'. Our upper bounds consider the more general case, where $R' \geq R$. For one-dimensional static ad hoc radio networks, we have presented a gossiping algorithm that requires no more than $n + 2\lceil R'/R \rceil - 1$ transmission slots. We prove that this algorithm is close to optimal by creating a network for which any algorithm requires $2\lceil n/2 \rceil + 2\lfloor R'/R \rfloor - 3$ transmission slots. Restricting to the class of collision-free algorithms, we have presented an algorithm that requires at most $n + 2\lceil R'/R \rceil$ transmission slots, and have proven that this algorithm is nearly optimal by creating a network for which any collision-free algorithm requires $2\lceil n/2 \rceil + 2\lfloor R'/R \rfloor - 2$ transmission slots. Restricting further to the class of singleton algorithms, we have provided an algorithm that takes no more than $n - D + 2$ transmission slots for any network with diameter $D \geq 2$. We prove that this algorithm is optimal by providing a matching lower bound.

We have also demonstrated separations between the classes of singleton, collision-free, and unrestricted algorithms. We have proven that, for any one-dimensional radio network with diameter $D > 2\lceil R'/R \rceil + 2$, our collision-free algorithm uses less than $n + D - 2$ transmission slots while any singleton algorithm would require at least $n + D - 2$ transmission slots. We also demonstrate that there exist one-dimensional radio networks for which any collision-free algorithm requires at least $n + 2\lceil R'/R \rceil - 1$ transmission slots while our algorithm uses less than $n + 2\lceil R'/R \rceil - 1$ transmission slots.

All of the previous lower bounds for gossiping algorithms in static ad hoc networks assume that $R = R'$ and that the network topology is not known. To derive lower bounds when the network topology is not known, an adversary can be created that chooses the network topology based on the algorithm. As we assume that all of the processors know the network topology, we cannot use this kind of adversarial argument. We provide the first non-trivial lower bounds for gossiping in the more difficult setting in which the network topology is known in advance.

Our results also apply to the transmission complexity of the gossiping task, that is, the total number of transmissions performed by processors during an algorithm's execution. Characterizing the transmission complexity can be useful in situations where processors have a limited amount of battery power and must conserve energy. We note that two transmitting processors, p and p', contribute two transmissions to the transmission complexity, regardless of whether or not the transmissions by p and p' occur during the same transmission slot. Thus, any algorithm A can be converted into a singleton algorithm A' with the same transmission complexity as A. It follows that our matching upper and lower bounds for the time complexity of singleton gossiping algorithms completely describe the transmission complexity for gossiping in our network model.

For the classes of collision-free and unrestricted algorithms, our upper and lower bounds differ by a small additive constant. Further, for these two classes, we have only provided lower bounds on the time to complete gossiping in particular networks in which gossiping is difficult, rather than providing lower bounds that depend on various properties (such as the diameter) of an arbitrary network. Extending these bounds is left as an open problem. We may also consider the same problem in different networks models, such as: asynchronous networks, networks that exist in two or three dimensions, mobile networks, or, networks in which the network topology is not initially known by all processors. Some results for deterministic gossiping in one-dimensional networks with mobile processors can be found in [7].

References

[1] B. Chlebus, L. Gąsieniec, A. Östlin, and J.M. Robson. Deterministic radio broadcasting. In *Automata, Languages and Programming*, pages 717–728, 2000.

[2] B. Chlebus, L. Gąsieniec, A. Lingas, and A. Pagourtzis. Oblivious gossiping in ad-hoc radio networks. In *Proc. 5th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM'2001)*, pages 44–51, 2001.

[3] M. Chrobak, L. Gąsieniec, and W. Rytter. Fast broadcasting and gossiping in radio networks. In *IEEE Symposium on Foundations of Computer Science*, pages 575–581, 2000.

[4] F. Cicalese, F. Manne, and Q. Xin. Faster centralized communication in radio networks. In *International Symposium on Algorithms and Computation*, pages 339–348, 2006.

[5] L. Gąsieniec, I. Potapov, and Q. Xin. Time efficient gossiping in known radio networks. In *Proc. 11th Colloq. on Struct. Inform. and Comm. Complexity, SIROCCO'04.*, pages 173–184, 2004.

[6] L. Gąsieniec, T. Radzik, and Q. Xin. Faster deterministic gossiping in directed ad-hoc radio networks. In *9th Scandinavian Workshop on Algorithm Theory*, pages 397–407, 2004.

[7] A. Miller. Gossiping in one-dimensional synchronous ad hoc radio networks. Master's thesis, University of Toronto, 2009. `http://www.cs.toronto.edu/$\sim$a4miller`.