Energy-Efficient Initialization Protocols for Radio Networks with no Collision Detection *

K. Nakano[†] S. Olariu[‡]

Abstract

A radio network (RN, for short) is a distributed system consisting of n radio stations. We assume that the stations are small, bulk-produced, hand-held devices running on batteries and cannot be distinguished by serial or manufacturing number. Since recharging batteries may not be possible while on mission, we are interested in designing protocols that are highly energy-efficient.

The initialization problem is to assign each of the n stations in the RN a unique ID. The initialization problem is non-trivial since the stations are assumed to be indistinguishable. The problem is fundamental, since practically all communication protocols for RNs proceed under the assumption that the RN has been initialized in advance.

The main contribution of this work is to propose energy-efficient randomized initialization protocols for RNs lacking collision detection capabilities. First, we show that if the number n of stations is known beforehand, the single-channel RN can be initialized by a protocol that terminates, with probability exceeding $1 - \frac{1}{n}$, in $O(n)$ time slots, with no station being awake for more than $O(\log \log n)$ time slots. We then go on to address the multi-channel case and show that if k, $(k \geq 1)$, channels are available, an n-station RN can be initialized, with probability exceeding $1 - \frac{1}{n}$, in $O(\frac{n}{k} + \log n)$ time slots, with no station being awake for more than $O(\log \log n)$ time slots.

Keywords: radio networks, wireless networks, mobile computing, initialization.

1 Introduction

In recent years, wireless and mobile communications have seen an explosive growth both in terms of the number of services provided and the types of technologies that have become available. Indeed, cellular telephony, radio paging, cellular data, and even rudimentary multimedia services have become commonplace and the demand for enhanced capabilities will continue to grow into the foreseeable future [2, 6, 25, 28]. It is anticipated that in the not-so-distant future, mobile users will be able to access their data and other services such as electronic mail, video telephony, stock market news, map services, electronic banking, while on the move [6, 11, 23].

1.1 Radio networks – generalities

A radio network (RN, for short) is a distributed system with no central arbiter, consisting of n radio transceivers, henceforth referred to as stations. We assume that the stations are bulk-produced, commodity hand-held devices running on batteries and that it is impossible or impractical to distinguish them by serial or manufacturing numbers.

An important task to perform is that of assigning the stations in an n-station RN Identification Numbers (IDs, for short), very much like IP addresses, in the range 1 to n. This task is known as initialization and is very important as many of the existing protocols tacitly assume that the stations already have IDs. The initialization problem is non-trivial since the stations are assumed to be indistinguishable. Further, since the stations are power-limited it is of importance to design energy-efficient initialization protocols for single-hop RNs both in the case where the system has a collision detection capability and in the case where this capability is not present.

As customary, time is assumed to be slotted and all the stations have a local clock that keeps synchronous time by interfacing with a Global Positioning System (GPS, for short) [8, 13, 19, 20]. We note here that under current technology, the commercially available GPS system[1] provides location information accurate within 22 meters as well as time information accurate to within 100 nanoseconds [8]. This is more than sufficient for the stations of the RN to synchronize .

The stations are assumed to have the computing

*Work supported in part by ONR grant N00014-97-1-0526.

†Dept. of Electrical and Computer Engineering, Nagoya Institute of Technology, Showa-ku, Nagoya 466-8555, JAPAN, nakano@elcom.nitech.ac.jp

‡Department of Computer Science, Old Dominion University, Norfolk, Virginia 23529, U.S.A., olariu@cs.odu.edu

[1] GPS systems using military codes achieve an accuracy that is one order of magnitude better than commercial codes.

power of a laptop computer; in particular, they all run the same protocol and can generate random bits that provide local data on which the stations may perform computations.

The stations communicate using k, $(k \geq 1)$, radio frequencies channels. We assume that in a time slot, a station can tune to one channel and/or transmit on at most one, possibly the same, channel. A transmission involves a data packet whose length is such that the transmission can be completed within one time slot. We employ the commonly-accepted assumption that when two or more stations are transmitting on a channel in the same time slot, the corresponding packets collide and are garbled beyond recognition. We distinguish between RN systems based on their capability to detect collisions. Specifically, in the RN with collision detection (CD, for short), at the end of a time slot the status of a radio channel is:

- NULL – no transmission on the channel
- SINGLE – exactly one transmission on the channel
- COLLISION – two or more transmissions on the channel

In the RN with no collision detection (no-CD, for short), at the end of a time slot the status of a radio channel is:

- NOISE – ambient noise: either no transmission or collision of several transmissions on the channel
- SINGLE – exactly one transmission on the channel.

In other words, the RN with no-CD cannot distinguish between ambient noise and two or more transmissions colliding on the channel. Several workers have pointed out that from a practical standpoint the no-CD assumption makes a lot of sense, especially in the presence of noisy channels which, we argue, tends to be the norm rather than the exception. Given the additional limitations, it is far more challenging to design efficient protocols for RNs with no-CD than for those with collision detection capabilities [4, 5].

It is very important to realize that since the stations of the RN are running on batteries, saving battery power is exceedingly important, as recharging batteries may not be an option while on mission. It is well known that a station expends power while its transceiver is active, that is, while transmitting or receiving a packet. It is perhaps surprising at first that a station expends power even if it receives a packet that is not destined for it [3, 9, 11, 21, 24, 26, 27]. Consequently, we are interested in developing protocols that allow stations to power their transceiver off (i.e. go to sleep) to the largest extent possible. Accordingly, we judge the goodness of a protocol by the following two yardsticks:

- the overall number of time slots required by the protocol to terminate
- for each individual station the total number of time slots when it has to be awake in order to transmit/receive packets.

The goals of optimizing these parameters are, of course, conflicting. It is relatively straightforward to minimize overall completion time at the expense of energy consumption. Similarly, one can minimize energy consumption at the expense of completion time [26, 27, 28]. The challenge is to strike a sensible balance between the two by designing protocols that take a small number of time slots to terminate while being, at the same time, as energy-efficient as possible.

1.2 State of the art

As stated previously, the initialization problem is to assign each of the n stations of an RN an integer ID in the range 1 to n such that no two stations receive the same ID. The initialization problem is fundamental in both network design and in multiprocessor systems [16].

Recently, Hayashi et al. [12] presented a protocol that initializes a single-channel, n-station RN with CD in $O(n)$ time slots with probability at least $1 - \frac{1}{2^{O(n)}}$. The protocol repeatedly partitions the stations of the RN into nonempty subsets until, eventually, every subset consists of a single station. Since the stations have to constantly monitor the channel, the protocol of [12] is not energy-efficient.

Later, Bordim et al. [7] showed that, if the number n of stations is known beforehand, a single-channel n-station RN with CD can be initialized with probability at least $1 - \frac{1}{n}$, in $O(n)$ time slots with no station being awake for more than $O(\log n)$ time slots. This protocol uses the initialization protocol of [12] as follows: Having partitioned the n-stations into $\frac{n}{\log n}$ subsets of roughly $\log n$ stations, each subset is initialized individually. The key observation is that a station needs to be awake only for the time necessary to initialize the subset to which it belongs. By using the protocol of [12], each subset can be initialized in $O(\log n)$ time slots with probability at least $1 - \frac{1}{2^{O(\log n)}}$ and, thus, no station has to be awake for more than $O(\log n)$ time slots. Once each subset has been initialized, a simple prefix sum computation allows the stations to update their local ID to the desired ID. Bordim et al. [7] went on to show how to use the single-channel initialization protocol to initialize, with probability exceeding $1 - \frac{1}{n}$, a k-channel, $(1 \leq k \leq \frac{n}{\log n})$, RN with CD in $O(\frac{n}{k})$ time slots, with no station being awake for more than $O(\log n)$ time slots.

1.3 Summary of our contributions

Our main contribution is to propose energy-efficient initialization protocols for RNs with no-CD requiring no station to be awake for more than $O(\log \log n)$ time slots.

We begin by showing that a single-channel RN with no-CD can be initialized by a protocol terminating, with probability at least $1 - \frac{1}{n}$, in $O(n)$ time slots with no station being awake for more than $O(\log n)$ time slots. Although the time performance of our protocol is the same as that in [7], our protocol works in the no-CD case and the technique used is totally different. Our protocol starts out by assigning each station a unique temporary ID in the range 1 to $16n$. After that, using the temporary ID, the protocol reassigns the stations unique IDs in the range 1 to n.

We then go on to improve this protocol such that no station needs to be awake for more than $O(\log \log n)$ time slots. The key idea is to assign the various stations in the RN temporary IDs in the range 1 to $16n$ in such a way that with high probability, every group of consecutive $64 \ln n$ integers contain the temporary ID of some station. In turn, this allows us to initialize each group locally with no station being awake for more than $O(\log \log n)$ time slots. Once this local initialization is performed, the local ID numbers can be updated to global ID numbers by performing the energy-efficient prefix sums protocol that we discuss in Section 3.

Finally, we extend the single-channel initialization protocol to a k, $(k \geq 1)$, channel RN with no-CD. Specifically, we show that with probability exceeding $1 - \frac{1}{n}$ this protocol terminates in $O(\frac{n}{k} + \log n)$ time slots, with no station being awake for more than $O(\log \log n)$ time slots.

The remainder of this work is organized as follows. Section 2 reviews basic probability theory that will provide the machinery for evaluating the performance of initialization protocols. Section 3 discusses energy-efficient prefix sums protocols and a simple initialization protocol that will be needed in the remainder of this work. Section 4 discusses an energy-efficient protocol that assigns the stations of an RN temporary ID numbers in a not-to-large range. This protocol turns out to be the workhorse of the whole paper. In Section 5 we show how to use the protocol developed in Section 4 to initialize a single-channel, n-station RN with no-CD in such a way that no station is awake for more than $O(\log n)$ time slots. In Section 6 we show that one can initialize a single-channel, n-station RN with no-CD with no station being awake for more than $O(\log \log n)$ time slots. In Section 7 we provide simulation results showing that our initialization protocol in Section 6 performs extremely well in practice. In Section 8 we show that if k, $(k \geq 1)$, channels are available, one can initialize an n-station RN with no-CD in $O(\frac{n}{k} + \log n)$ time slots with no station being awake for more than roughly $O(\log \log n)$ time slots. Finally, Section 9 offers concluding remarks and points out directions for further work.

2 A brief refresher of basic probability theory

The main goal of this section is to review elementary probability theory results that are useful for analyzing the performance of our protocols. For a more detailed discussion of background material we refer the reader to [15].

Throughout, $\Pr[A]$ will denote the probability of event A. Let $E_1, E_2, \ldots, E_m$ be arbitrary events over a sample space. The well known De Morgan law states that

$$\overline{\bigcap_{i=1}^{m} E_i} = \bigcup_{i=1}^{m} \overline{E_i} \tag{1}$$

where $\overline{E_i}$ is the event that occurs if and only if E_i does not. In addition, it is known that

$$\Pr[\bigcup_{i=1}^{m} E_i] \leq \sum_{i=1}^{m} \Pr[E_i], \tag{2}$$

with equality holding if the events are independent.

For example, assume that the stations of an RN have been partitioned into k groups $G_1, G_2, \ldots, G_k$ and that for a given i, $(1 \leq i \leq k)$, the probability of the event $\overline{E}_i$ that group G_i fails to satisfy a predicate $\mathcal{P}$ is p_i. Now, (1) and (2) combined guarantee that the probability of the event that all the groups satisfy predicate $\mathcal{P}$ is:

$$\Pr[E_1 \cap E_2 \cap \cdots \cap E_k] = 1 - \Pr[\overline{E_1 \cap E_2 \cap \cdots \cap E_k}] = 1 - \Pr[\overline{E_1} \cup \overline{E_2} \cup \cdots \cup \overline{E_k}] \geq 1 - \sum_{i=1}^{m} p_i. \tag{3}$$

Notice that (3) holds regardless of whether or not the events $\overline{E_i}$ are independent.

Let $X_1, X_2, \ldots, X_n$ be independent Poisson trials such that for $1 \leq i \leq n$, $\Pr[X_i = 1] = p_i$ and $\Pr[X_i = 0] = 1 - p_i$ and write $X = X_1 + X_2 + \cdots + X_n$. It is well known that the expected value of X is given by

$$E[X] = E[X_1]+E[X_2]+\cdots+E[X_n] = p_1+p_2+\cdots+p_n. \quad (4)$$

To analyze the tail of X, we shall make use of the following estimate, commonly referred to as Chernoff bound [15]:

$$\Pr[X > (1+\delta)E[X]] \quad < \quad \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{E[X]} \quad (0 \le \delta) \quad (5)$$

We also use the following inequality that follows from (5):

$$\Pr[X > (1+\epsilon)E[X]] \quad < \quad e^{-\frac{\epsilon^2}{3}E[X]} \qquad (0 \le \epsilon \le 1) \quad (6)$$

Finally, we take note of a well-known result that will be used frequently in the remainder of this work.

Lemma 2.1 For every positive constant c, the sequence $\left(1-\frac{c}{n}\right)^n$ is monotonically increasing and

$$\lim_{n\to\infty}\left(1-\frac{c}{n}\right)^n = \frac{1}{e^c}. \quad (7)$$

3 Energy-efficient prefix sums protocols

The purpose of this section is to present various prefix sums protocols which are key ingredients of our energy-efficient initialization protocols.

3.1 A generic prefix sums protocol

Consider a single-channel, n-station RN populated by stations $S(1), S(2), \ldots, S(n)$ and assume that for every i, $(1 \le i \le n)$, station $S(i)$ stores a number $N(i)$. Initially, only station $S(i)$ knows $N(i)$. The goal is to compute for every i, $(1 \le i \le n)$, the sum $N(1)+N(2)+\cdots+N(i)$ referred to as the i-th prefix sum of $N(1), N(2), \ldots, N(n)$.

The details of the prefix sums protocol follow. To begin, station S(1) transmits $N(1)$ on the channel. At this time all the stations have their transceiver powered off (i.e. are asleep) except for S(1) that is transmitting and S(2) that is picking up $N(1)$. Now, S(2) computes $N(1)+N(2)$ and transmits the result to $S(3)$. In this time slot all the stations are asleep except for S(2) and S(3). Upon receiving $N(1)+N(2)$, S(3) computes $N(1)+N(2)+N(3)$ and transmits the result to S(4). This is then continued until, at the end of $n-1$ time slots, S(n) receives $N(1)+N(2)+\cdots+N(n-1)$ and computes $N(1)+N(2)+\cdots+N(n)$.

Thus, all the prefix sums can be computed in $n-1$ time slots. Importantly, no station is awake for more than two time slots. To summarize, we state the following result.

Lemma 3.1 The prefix sums protocol involving the n stations of a single-channel RN can be completed in $n-1$ time slots, with no station being awake for more than two time slots.

Next, we show how to extend the simple prefix sums protocol we just discussed to the multi-channel case. Consider a k-channel, n-station RN with channels $\mathrm{C}(1), \mathrm{C}(2), \ldots, \mathrm{C}(k)$ and stations $\mathrm{S}(1), \mathrm{S}(2), \ldots, \mathrm{S}(n)$. We first assume that $k \le \frac{n}{\log n}$. For simplicity of exposition we also assume that both n and k are a power of 2.

The protocol consists of three phases that we discuss in detail below. In Phase 1, we begin by partitioning the n stations into k equal-sized groups $G(1), G(2), \ldots, G(k)$ of $\frac{n}{k}$ stations each. Let $S(i,j)$, $(1 \le i \le k; 1 \le j \le \frac{n}{k})$, denote the j-th station in group $G(i)$, that is, $S(i,j) = S((i-1)\frac{n}{k}+j)$. Since, by assumption, $k \le \frac{n}{\log n}$, each group contains at least $\frac{n}{k} \ge \log n \ge \log k$ stations. Having dedicated channel $\mathrm{C}(i)$ to group $G(i)$, $(1 \le i \le k)$, we proceed to compute the local prefix sums within each group $G(i)$ using Lemma 3.1. Thus, at the end of Phase 1, each station $S(i,j)$ in group $G(i)$ stores the local prefix sum $N((i-1)\frac{n}{k}+1)+N((i-1)\frac{n}{k}+2)+\cdots+N((i-1)\frac{n}{k}+j)$. Further, the last station $S(i,\frac{n}{k})$ stores the sum $y(i) = N((i-1)\frac{n}{k}+1)+N((i-1)\frac{n}{k}+2)+\cdots+N(i\frac{n}{k})$ of all the items stored by the stations in $G(i)$. By Lemma 3.1, Phase 1 terminates in $\frac{n}{k}-1$ time slots with stations $S(i,1)$ and $S(i,\frac{n}{k})$ being awake for one time slot and the other stations being awake for two time slots.

Let $y[a,b] = y(a)+y(a+1)+\cdots+y(b)$, $(1 \le a < b \le k)$, denote a partial sum of the sums. Recall that at the end of Phase 1 each station $S(i,j)$, $(1 \le i \le k; 1 \le j \le \frac{n}{k})$, knows the local prefix sum $N((i-1)\frac{n}{k}+1)+N((i-1)\frac{n}{k}+2)+\cdots+N((i-1)\frac{n}{k}+j)$. Thus, once $S(i,j)$ learns the global prefix sum $y[1,i-1]$, it can immediately compute the desired prefix sum $N(1)+N(2)+\cdots+N(j) = y[1,i-1]+N((i-1)\frac{n}{k}+1)+N((i-1)\frac{n}{k}+2)+\cdots+N((i-1)\frac{n}{k}+j)$.

In Phases 2 and 3, we compute the global prefix sums $y[1,1], y[1,2], \ldots, y[1,k-1]$. Note that, in fact, the last prefix sum $y[1,k]$ is not really necessary.

Consider the binary tree of partial sums illustrated in Figure 1. The sums $y(1), y(2), \ldots, y(k)$ appear in left-to-rigth order in the k leaves of the tree. Each internal node corresponds to the sum of its descendants. Further, we define the level of a node to be the distance from a leaf. Clearly, a node at level i, $(0 \le i \le \log k)$, has 2^i leaf-descendants and, thus, corresponds to a partial sum of 2^i elements.

A nice property of the binary tree of partial sums is that each prefix sum can be obtained by the sum of partial sums chosen at most one from each level. For ex-

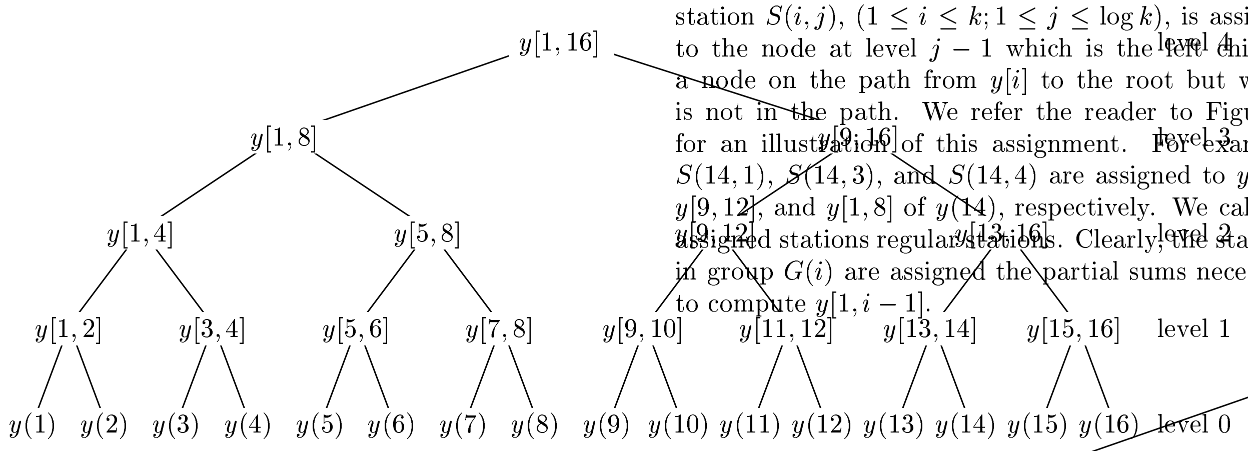

Figure 1. Illustrating the binary tree of partial sums

ample, the prefix sum $y[1,13]$ can be computed by the sums $y(13)$ (level 0), $y[9,12]$ (level 2), and $y[1,8]$ (level 3). In other words, $y[1,13] = y(13) + y[9,12] + y[1,8]$. These three partial sums can be found as follows: Consider the path from leaf $y(14)$ to the root. For every internal node on the path the left children not on the path are $y(13)$, $y[9,12]$, and $y[1,8]$, which are exactly the nodes to be chosen for prefix sum $y[1,13]$. In general, the prefix sum $y[1,i-1]$, $(2 \le i \le k)$, can be computed by identifying the path from node $y(i)$ to the root and by picking the left children of the nodes on this path that are themselves not on the path. Using this property, we will compute all necessary prefix sums $y[1,1], y[1,2], \ldots, y[1,k-1]$.

Recall that, at the end of Phase 1, each station $S(i,\frac{n}{k})$, $(1 \le i \le k)$, knows the value of $y(i)$. Using this fact, Phase 2 performs the following two tasks simultaneously:

Task 1 all the necessary partial sums are computed,

Task 2 stations in $G(i)$, $(2 \le i \le k)$, receive the partial sums necessary to compute $y[1,i-1]$.

Since, as noted before, $y[1,k]$ is not necessary, we do not have to compute the partial sums of the rightmost path of the binary tree. For Task 1, we assign the first station $S(i,1)$ in each group $G(i)$ to a necessary partial sum of the binary tree as illustrated in Figure 2. Since the number of necessary partial sums is $(\frac{k}{2}-1)+(\frac{k}{4}-1)+\cdots+1 < k$, this assignment is feasible. We call these stations primary stations. For Task 2, we assign each station $S(i,j)$, $(1 \le i \le k; 1 \le j \le \log k)$, to a node (i.e. partial sum) in the binary tree as follows: station $S(i,j)$, $(1 \le i \le k; 1 \le j \le \log k)$, is assigned to the node at level $j-1$ which is the left child of a node on the path from $y[i]$ to the root but which is not in the path. We refer the reader to Figure 3 for an illustration of this assignment. For example, $S(14,1)$, $S(14,3)$, and $S(14,4)$ are assigned to $y(13)$, $y[9,12]$, and $y[1,8]$ of $y(14)$, respectively. We call the assigned stations regular stations. Clearly, the stations in group $G(i)$ are assigned the partial sums necessary to compute $y[1,i-1]$.

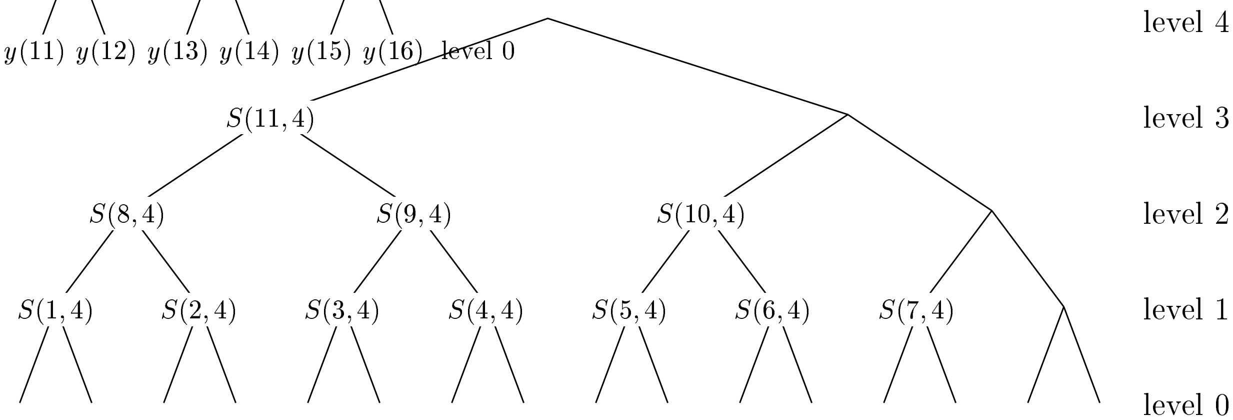

Figure 2. Illustrating the primary stations

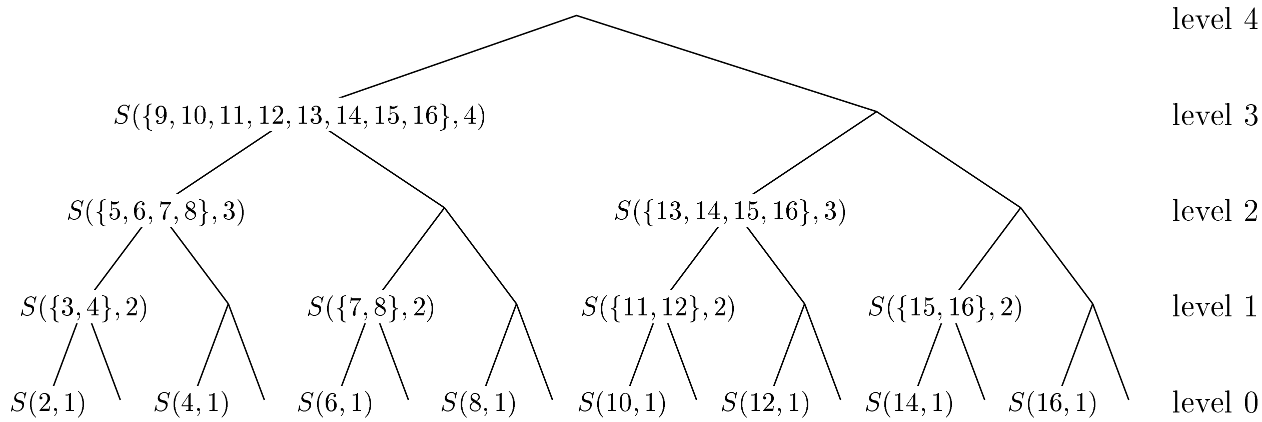

Figure 3. Illustrating the regular stations

We are now in position to show the details of Phase 2. For Task 1, taking turns, the stations $S(2i-1,\frac{n}{k})$ and $S(2i,\frac{n}{k})$, $(1 \le i \le \frac{n}{2}-1)$, transmit $y(2i-1)$ and $y(2i)$ on channel $C(i)$. The primary station $S(i,1)$ of each $y[2i-1,2i]$ $(1 \le i \le \frac{n}{2}-1)$, at level 1 monitors the channel $C(i)$ in these two time slots, picks up $y(2i-1)$ and $y(2i)$, and computes $y[2i-1, 2i] = y(2i-1) + y(2i)$. Next, the primary stations of $y[4i-3, 4i-2]$ and $y[4i-1, 4i]$ $(1 \leq i \leq \frac{n}{4}-1)$ transmit $y[4i-3, 4i-2]$ and $y[4i-1, 4i]$ on channel $C(i)$, respectively. The primary station of $y[4i-3, 4i]$ picks up these two values, and computes $y[4i-3, 4i] = y[4i-3, 4i-2] + y[4i-1, 4i]$. The same procedure is repeated until all the stations have received the necessary information. Clearly, Task 1 terminates in $2k$ time slots with stations $S(i, \frac{n}{k})$ being awake for one time slot and the primary stations $S(i, 1)$ being awake for three time slots.

It is sufficient for Task 2 that each regular station monitors an appropriate channel in an appropriate time slot because, in Task 1, every necessary partial sum is transmitted on a channel in a given time slot. Thus, in Task 2, every regular station is awake for one time slot. Therefore, Phase 2 is completed in $2k$ time slots with the primary stations $S(i, 1)$ being awake for at most four time slots and the other stations being awake for at most two time slots.

In Phase 3, we compute the sum of partial sums to obtain all necessary prefix sums $y[1, 1], y[1, 2], \ldots, y[1, k-1]$. For example, we compute $y(13) + y[9, 12] + y[1, 8]$ to obtain the prefix sum $y[1, 13]$. Since at the end of Phase 2 every station assigned to a node knows the corresponding partial sums, we can use the prefix sums protocol in Lemma 3.1. For example, at the end of Phase 2, stations $S(14, 1)$, $S(14, 3)$, and $S(14, 4)$ know $y(13)$, $y[9, 12]$, and $y[1, 8]$, respectively. Thus, by executing the prefix sums protocol of Lemma 3.1 the stations $S(14, 1)$, $S(14, 3)$, and $S(14, 4)$ can compute the global prefix sum $y[1, 13] = y(13) + y[9, 12] + y[1, 8]$. After the prefix sums protocol, $S(14, 4)$ broadcast global prefix sum $y[1, 13]$ to all the stations in $G(14)$. Then, each of them computes the sum of the global prefix sum $y[1, 13]$ and its local prefix sum, which is exactly the desired prefix sum. Since each of the k groups has at most $\log k$ primary stations, the prefix sums protocol of Lemma 3.1 runs in $\log k - 1$ time slots, with no station being awake for more than two time slots. Further, the final broadcast takes one time slots. Therefore, Phase 3 runs in $\log k$ time slots with the first stations $S(i, 1)$ in each group being awake for at most two time slots, and the other stations being awake for at most three time slots.

By Phases 1, 2, and 3 combined, the prefix sums can be computed in $\frac{n}{k} + 3 \log k - 1$ time slots with no station being awake for more than eight time slots.

Consequently, we have proved the following result.

Lemma 3.2 The prefix sums of a sequence of n numbers can be computed on a k-channel, n-station RN with no-CD in $\frac{n}{k} + 3 \log k - 1$ time slots, with no station being awake for more than eight time slots, provided that $k \leq \frac{n}{\log n}$.

If $k > \frac{n}{\log n}$, then group $G(i)$, $(1 \leq i \leq k)$, has fewer than $\log k$ stations, and the prefix sums protocol of Lemma 3.2 does not work. In this case, we use only $\frac{n}{\log n}$ channels even if more than $\frac{n}{\log n}$ channels are available. By Lemma 3.2, the prefix sums can be computed in $\log n + 3 \log k - 1 = \frac{n}{k} + O(\log n)$ time slots. Further, $\frac{n}{k} + 3 \log k - 1 = \frac{n}{k} + O(\log n)$ whenever $k \leq \frac{n}{\log n}$. Thus, we have the following result.

Corollary 3.3 The prefix sums of a sequence of n numbers can be computed on a k-channel, n-station RN with no-CD in $\frac{n}{k} + O(\log n)$ time slots, with no station being awake for more than eight time slots regardless of the number k of the channels.

Suppose that the RN has n stations each of which has a unique ID in the range $[1, m]$, $(m \geq n)$. Let $S(i)$ denote the station with ID i, $(1 \leq i \leq m)$. Note that for some i, $S(i)$ may not exist. Suppose that each station $S(i)$ stores a number $N(i)$. The prefix sum of $N(i)$ (or $S(i)$) is the sum of numbers with indices no more than i, that is,

$$\sum \{N(j) \mid 1 \leq j \leq i \text{ and } S(j) \text{ exists}\}.$$

The sparse prefix sums problem asks to compute the all prefix sums.

We design a protocol for solving the sparse prefix sums problem in recursive manner as follows. When the protocol terminates the following three conditions are satisfied:

(ps1) Every station $S(i)$, $(1 \leq i \leq n)$, stores its prefix sum.

(ps2) The last station, that is, station $S(l)$ such that no station $S(i)$ with $i > l$ exists, has been identified

(ps3) The protocol takes $2m - 2$ time slots and no station is awake for more than $2 \log m$ time slots.

If $m = 1$, then $S(1)$ knows $N(1)$ and the above conditions are verified. Now, assume that $m \geq 2$. Partition the n stations into two groups $\mathcal{P}_1 = \{S(i) \mid 1 \leq i \leq \frac{m}{2}\}$ and $\mathcal{P}_2 = \{S(i) \mid \frac{m}{2} + 1 \leq i \leq m\}$. Recursively compute the sparse prefix sums in $\mathcal{P}_1$ and $\mathcal{P}_2$. By the induction hypothesis, conditions (ps1)–(ps3) above are satisfied and, therefore, each of the two subproblems can be solved in $m - 2$ time slots, with no station being awake for more than $2 \log m - 2$ time slots. Let $S(j)$ and $S(l)$ be the last active stations stations in $\mathcal{P}_1$ and in $\mathcal{P}_2$, respectively. In the next time slot, station $S(j)$ transmits the sum $\sum \{N(i) \mid 1 \leq i \leq j \text{ and } S(i) \text{ exists}\}$ on the channel. Every station in $\mathcal{P}_2$ monitors the channel and updates accordingly the value of its prefix sum. In one additional time slot station $S(l)$ reveals its identity. The reader should have no difficulty to confirm that the protocol satisfies (ps1)–(ps3) above are satisfied. Thus, we have the following result.

Lemma 3.4 The prefix sums problem on the single-channel RN of n stations with unique ID $[1, m]$ $(m \geq n)$ can be solved in $2m-2$ time slots with no station being awake for more than $2\log m$ time slots.

To conclude this section we point out the relevance of Lemma 3.4 to the initialization problem. Assume that the n stations of a single-channel RN possess integer IDs in some range 1 to m with $n \leq m$, with no two stations sharing the same ID. The protocol `Reduce(`m`)` presented below assigns the stations integers in the range 1 to n. We assume that m is known to all the stations. However, n need not be known at the beginning of the protocol.

Protocol `Reduce(`m`)`
    every station $S(i)$ generates a number $N(i) = 1$;
    assign to station $S(i)$, $(1 \leq i \leq m)$, the ID number $\sum\{N(j) | 1 \leq j \leq i \text{ and } S(j) \text{ exists}\}$.

By Lemma 3.4, the sparse prefix sums of the $N(i)$s yield distinct ID numbers in the range 1 to n. Thus, we have proved the following important result.

Lemma 3.5 A single-channel, n-station RN each of whose stations has a unique ID in the range 1 to m, $(n \leq m)$, can be initialized in $2m-2$ time slots with no station being awake for more than $2\log m$ time slots.

4 Assigning temporary IDs

The key idea of our initialization protocol is to assign a (large) subset of the n stations of the RN temporary IDs in a (not too large) range 1 to m. Once this is done, we use protocol `Reduce(`m`)` discussed in Section 3 to assign the stations IDs in the range 1 to n.

The main goal of this section is to discuss and analyze the performance of a protocol that assigns temporary IDs in the range $s+1$ to $s+m$ to a subset of the stations of a single-channel, n-station RN. For now, assume that all the stations know s and m and that $n \leq m$. The details are spelled out as follows:

Protocol `Assign-Temporary-ID(`m, s`)`
each station picks an integer uniformly at random in the range 1 to m and then goes to sleep;
for $i \leftarrow 1$ to m do
    the stations that have picked integer i wake up and transmit on the channel;
    if the status of the channel is SINGLE then
        the unique station that has transmitted records the number $s+i$ as its temporary ID.
endfor

Clearly, protocol `Assign-Temporary-ID(`m, s`)` terminates in m time slots and every station is awake for exactly one time slot. When the protocol terminates, a subset of the n stations have received temporary IDs in the range $s+1$ to $s+m$.

Imagine that, taking turns, the n stations pick integers in the range 1 to m. It is clear that the j-th, $(1 \leq j \leq n)$, station to select may end up picking an integer that was already chosen by one of the $j-1$ preceding stations. We refer to this as a collision event and observe that its probability is bounded above by $p_j = \frac{j-1}{m}$. (To see this, observe that $\frac{j-1}{m}$ is the probability of a collision occurring in case no collision was experienced by any of the previous stations.) Clearly, a collision event may prevent at most two stations from receiving their temporary IDs. Thus, in the presence of x collision events, up to $2x$ stations may fail to receive temporary IDs.

We are interested in obtaining an upper bound on the number of stations that fail to receive temporary IDs at the end of the protocol. For this purpose, let X_i, $(1 \leq i \leq n)$, be independent random variables with $\Pr[X_i = 1] = p_i = \frac{i-1}{m}$ and $\Pr[X_i = 0] = 1 - p_i$. Further, let $X = X_1 + X_2 + \cdots X_n$. It is very important to note that X provides an upper bound on the number of collision events occurring in the process of assigning n stations integer IDs in an arbitrary interval of size m. The expected value of X is

$$E[X] = \sum_{i=1}^{n} p_i = \sum_{i=1}^{n} \frac{i-1}{m} = \frac{n(n-1)}{2m}. \tag{8}$$

and the tail of X can be bounded using (6):

$$\Pr[X > \frac{n(n-1)}{m}] = \Pr[X > 2E[X]] < e^{-\frac{1}{3}E[X]} = e^{-\frac{n(n-1)}{6m}}. \tag{9}$$

Equation (9) guarantees that with probability exceeding $1-e^{-\frac{n(n-1)}{6m}}$ the number of collision events occurring in the protocol is no more than $\frac{n(n-1)}{m}$. Since, as we saw, each collision event may prevent up to two stations from receiving their temporary IDs, we have the following result.

Lemma 4.1 With probability exceeding $1 - e^{-\frac{n(n-1)}{6m}}$, when protocol `Assign-Temporary-ID(`$m, s$`)` terminates, fewer than $\frac{2n(n-1)}{m}$ stations have not received their temporary IDs.

Let f be a positive real-valued function over the naturals. By choosing m to satisfy the condition

$$\frac{n(n-1)}{6m} \geq \ln(nf(n)), \tag{10}$$

we obtain the following corollary.

Corollary 4.2 Assume that condition (10) holds. With probability exceeding $1 - \frac{1}{nf(n)}$, when protocol `Assign-Temporary-ID`(m, s) terminates, fewer than $\frac{2n(n-1)}{m}$ stations have failed to receive temporary IDs.

Next, suppose that when protocol `Assign-Temporary-ID`(m, s) is performed condition (10) does not hold. In other words, we have

$$\frac{n(n-1)}{6m} < \ln(nf(n)).$$

Clearly, for the random variable X defined above, we have

$$E[X] \;=\; \frac{n(n-1)}{2m} \;<\; 3\ln(nf(n)).$$

Let δ be the number satisfying $(1+\delta)E[X] = 10\ln(nf(n))$. Then, we have

$$1+\delta = \frac{10\ln(nf(n))}{E[X]} > \frac{10\ln(nf(n))}{3\ln(nf(n))} = \frac{10}{3}. \tag{11}$$

By using the Chernoff bound in (5) we can write,

$$\begin{aligned}
\Pr[X > 10\ln(nf(n))] &= \Pr[X > (1+\delta)E[X]] \\
&< \left(\frac{e}{1+\delta}\right)^{(1+\delta)E[X]} \\
&< \left(\frac{3e}{10}\right)^{10\ln(nf(n))} \\
&< \left(\frac{1}{e}\right)^{\ln(nf(n))} = \frac{1}{nf(n)}.
\end{aligned}$$

We just proved that with probability exceeding $1 - \frac{1}{nf(n)}$ the number X of collision events is bounded by $10\ln(nf(n))$. Since each collision event may prevent up to two stations from receiving their temporary IDs, we have proved the following result.

Lemma 4.3 Assume that the condition (10) does not hold. With probability exceeding $1 - \frac{1}{nf(n)}$, when protocol `Assign-Temporary-ID`(m, s) terminates, fewer than $20\ln(nf(n))$ stations have failed to receive temporary IDs.

5 Initializing the RN within logarithmic awake time

We are now in a position to present our first initialization protocol for single-channel, n-station RNs with no-CD that terminates with probability exceeding $1-\frac{1}{n}$ in $O(n)$ time slots with no station being awake for more than $O(\log n)$ time slots

At this time, we are in a position to make the function $f(n)$ introduced in Section 4 precise. For reasons that will become clear later we write

$$f(n) = \log\log n + 3. \tag{12}$$

In outline, our initialization protocol performs repeatedly `Assign-Temporary-ID`(m, s) with an appropriate choice of parameters m and s to guarantee that, with probability at least $1 - \frac{1}{n}$, all the n stations receive temporary IDs. The details of our initialization protocol is as follows.

Protocol `Initialize`(n)
$s \leftarrow 0$;
for $i \leftarrow 1$ to $\log\log n + 3$ do
  all stations with no temporary ID participate in protocol `Assign-Temporary-ID`$(\frac{n}{2^{i-4}}, s)$;
  $s \leftarrow s + \frac{n}{2^{i-4}}$;
endfor;
initialize all stations by using protocol `Reduce`$(16n)$

Since protocol `Assign-Temporary-ID`$(\frac{n}{2^{i-4}}, s)$ terminates in $\frac{n}{2^{i-4}}$ time slots with no station being awake for more than one time slot, the for loop in protocol `Initialize`(n) takes altogether

$$\sum_{i=1}^{\log\log n+3} \frac{n}{2^{i-4}} \quad < \quad 16n$$

time slots and no station is awake for more than $f(n) = \log\log n + 3$ time slots.

As we are about to show, with probability exceeding $1 - \frac{1}{n}$ when the for loop in protocol `Initialize`(n) terminates, all the n stations have received temporary IDs in the range 1 to $16n$. Therefore, by using the protocol `Reduce`$(16n)$ the stations will receive unique IDs in the range 1 to n.

Our plan is to use Corollary 4.2 and Lemma 4.3, combined, to show that with probability exceeding $1-\frac{1}{n}$, at the end of $\log\log n$ iterations of the for loop, at most $20\ln(nf(n))$ stations have failed to receive temporary IDs. For this purpose, we need the following technical result.

Lemma 5.1
Assume that i satisfies $\frac{n}{2^{i-1}2^{2^i}} \geq 20\ln(nf(n))$. With probability exceeding $1-\frac{i}{nf(n)}$, at the end of the i-th iteration of the for loop in protocol `Initialize(n)`, at most $\frac{n}{2^{i-1}2^{2^i}}$ stations have not received temporary IDs.

Proof. We prove Lemma 5.1 by the induction on i.
Basis: In the first iteration of the for loop protocol `Assign-Temporary-ID(8n,0)` is executed. Since, at this time, $m = 8n$, condition (10) is satisfied and Corollary 4.2 guarantees that with probability exceeding $1-\frac{1}{nf(n)}$, at the end of the first iteration no more than $\frac{2n(n-1)}{8n} < \frac{n}{4} = \frac{n}{2^0 2^{2^1}}$ stations have not received temporary IDs.
Inductive step: Assume that at the end of the $(i-1)$-th iteration of the for loop, with probability exceeding $1-\frac{i-1}{nf(n)}$, a number $n' \leq \frac{n}{2^{i-2}2^{2^{i-1}}}$ of stations have not received temporary IDs.

In the i-th iteration `Assign-Temporary-ID(`$\frac{n}{2^{i-4}}, s$`)` is executed. If at the beginning of the i-th iteration condition (10) is not satisfied, Lemma 4.3 and the induction hypothesis, combined, guarantee that with probability exceeding $1-\frac{i-1}{nf(n)}-\frac{1}{nf(n)} = 1-\frac{i}{nf(n)}$ the number stations that have not received temporary IDs at the end of the i-th iteration is at most

$$20\ln(nf(n)) \leq \frac{n}{2^{i-1}2^{2^i}}$$

as claimed.

We may assume, therefore, that condition (10) holds at the beginning of the i-th iteration. Now, Corollary 4.2 and the induction hypothesis combined, guarantee that with probability exceeding $1-\frac{i-1}{nf(n)}-\frac{1}{nf(n)} = 1-\frac{i}{nf(n)}$ at the end of the i-th iteration at most

$$\frac{2n'(n'-1)}{\frac{n}{2^{i-4}}} \leq \frac{\frac{2n}{2^{i-2}2^{2^{i-1}}}\frac{n}{2^{i-2}2^{2^{i-1}}}}{\frac{n}{2^{i-4}}} = \frac{2n^2}{2^{2i-4}2^{2^i}\frac{n}{2^{i-4}}} = \frac{n}{2^{i-1}2^{2^i}}$$

stations have not received temporary IDs.

Consequently, with probability exceeding $1-\frac{i}{nf(n)}$, at the end of the i-th iteration fewer than $\frac{n}{2^{i-1}2^{2^i}}$ stations have not received temporary IDs and the proof of Lemma 5.1 is complete. □

Let I be the largest integer satisfying

$$\frac{n}{2^{I-1}2^{2^I}} \geq 20\ln(nf(n)). \quad (13)$$

By Lemma 5.1, at the end of the I-th iteration of the for loop, with probability exceeding $1-\frac{I}{nf(n)}$ fewer than $\frac{n}{2^{I-1}2^{2^I}}$ stations have failed to receive temporary IDs.

Let us follow the $(I+1)$-th iteration. If at the beginning of this iteration condition (10) is satisfied then by (3), Corollary 4.2 and Lemma 5.1 combined, with probability exceeding $1-\frac{I+1}{nf(n)}$ at most $\frac{n}{2^I 2^{2^{I+1}}}$ stations have not received their temporary IDs. Since I was chosen to be the largest integer that satisfies equation (13), we must have

$$\frac{n}{2^I 2^{2^{I+1}}} < 20\ln(nf(n)).$$

On the other hand, if at the beginning of the $(I+1)$-th iteration condition (10) is not satisfied, then (3) and Lemma 4.3 combined guarantee that with probability exceeding $1-\frac{I+1}{nf(n)}$ at the end of the $(I+1)$-th iteration, fewer than $20\ln(nf(n))$ stations have failed to receive temporary IDs.

Observe that for sufficiently large n, the integer I specified in (13) must satisfy the double inequality:

$$\frac{n}{2^I 2^{2^{I+1}}} < 20\ln(nf(n)) \leq \frac{n}{2^{I-1}2^{2^I}}.$$

The reader should be in a position to confirm that this implies $I = \lfloor \log\log n \rfloor$ for sufficiently large n. Thus, we have proved the following result.

Lemma 5.2 With probability exceeding $1-\frac{1}{n}$, at the end of $\log\log n$ iterations of the for loop in protocol `Initialize(n)`, fewer than $20\ln(nf(n))$ stations have failed to receive temporary IDs.

By Lemma 5.2 at the end of the first $\log\log n$ iterations of the for loop at most $20\ln(nf(n))$ stations have yet to receive their temporary IDs. Let us trace the execution of the last two iteration of the for loop where `Assign-Temporary-ID(`$\frac{n}{2^{I-2}}, s$`)` and `Assign-Temporary-ID(`$\frac{n}{2^{I-1}}, s$`)` are executed.

More specifically, we are interested in evaluation the probability that no collision event occurs in the $(I+2)$-th or the $(I+3)$-th iteration of the for loop. For simplicity, we assume the exactly $20\ln(nf(n))$ stations still participate in the protocol at the beginning of the $(I+2)$-th iteration. Proceeding as before, the probability that the j-th, $(1 \leq j \leq 20\ln(nf(n)))$, station experiences a collision event in either of the $(I+2)$-th iteration is at most

$$p_j = \frac{j-1}{\frac{n}{2^{I-2}}} = \frac{(j-1)2^{I-2}}{n} \leq \frac{(j-1)\log n}{4n}.$$

Thus, for either $i = I+2$ a collision event occurs with probability at most

$$\begin{aligned} \sum_{j=1}^{20\ln(nf(n))} p_j &< \frac{400\log n(\ln(nf(n))^2}{8n} \\ &< \frac{50\log n(\ln(nf(n))^2}{n} \qquad \text{(since } I \leq \log\log n\text{)} \\ &< \frac{1}{n^{\frac{1}{2}}} \qquad \text{(for sufficiently large } n\text{)} \end{aligned}$$

Thus, with probability exceeding $1-\left(\frac{1}{n^{\frac{1}{2}}}\right)^2 = 1-\frac{1}{n}$ there are no collisions events in at least one of the $(I+2)$-th or the $(I+3)$-th iterations. In other words, with probability exceeding $1-\frac{1}{n}$ the first $f(n) = \log\log n+3$ iterations of the for loop succeed in assigning temporary IDs in the range 1 to $16n$ to all the stations. Finally, by applying protocol `Reduce(`$16n$`)` every station receives a distinct ID number in the range 1 to n. Moreover, by Lemma 3.5, this latter task is completed in $32n-2$ time slots with no station being awake for more than $2\log(16n) = 2\log n+8$ time slots. Consequently, protocol `Initialize(`n`)` terminates in at most $16n+32n-2 = 48n-2$ time slots with no station being awake for more than $2\log n+8+\log\log n+3 = 2\log n+\log\log n+11$ time slots. The following theorem summarizes our findings.

Theorem 5.3 With probability exceeding $1-\frac{1}{n}$, protocol `Initialize(`n`)` initializes a single-channel, n-station RN with no-CD in at most $O(n)$ time slots with no station being awake for more than $O(\log n)$ time slots.

6 Initializing the RN within doubly logarithmic awake time

The main goal of this section is to show that one can modify protocol `Initialize(`n`)` discussed in Section 5 to reduce the number of awake time slots per station to at most $3\log\log n+15$.

Recall that the for loop in protocol `Initialize(`n`)` takes $16n$ time slots altogether and no station is awake for more than $f(n) = \log\log n+3$ time slots. Moreover, with probability exceeding $1-\frac{1}{n}$, when the for loop terminates, all the stations have received temporary IDs. Our analysis in the previous section revealed that the bulk of the $2\log n+\log\log n+11$ time slot awake time experienced by the stations stems from performing protocol `Reduce(`$16n$`)`. Our plan is to take advantage of the particular distribution of a (dense) subset of the temporary IDs to reduce the number of awake time slots.

6.1 Taking advantage of a uniform distribution

To give the reader an idea of our approach, we now show how one could initialize the stations with temporary IDs in the range 1 to $8n$. Thereafter, we will proceed to shuffle temporary IDs and will apply the same technique to all the stations.

Recall that, as showed in Section 5, with probability exceeding $1-\frac{1}{n\log\log n}$, as a result of running protocol `Assign-Temporary-ID(`$8n,0$`)`, at least $\frac{3n}{4}$ stations have been assigned temporary IDs in the range 1 to $8n$. Moreover, it is easy to see that the resulting temporary IDs come from an underlying uniform distribution over the range 1 to $8n$. Consequently, for an arbitrary fixed i, $(1\le i\le 8n)$, the probability that some station has received temporary ID i is at least $\frac{\frac{3n}{4}}{8n} = \frac{3}{32}$.

Now, partition stations with IDs in the range 1 to $8n$ into $\frac{n}{4\ln n}$ groups $G(1), G(2), \ldots, G(\frac{n}{4\ln n})$ such that group $G(i)$, $(1\le i\le \frac{n}{4\ln n})$, involves the stations with IDs in the range $32(j-1)\ln n+1$ to $32j\ln n$. Since each group $G(i)$ involves stations with IDs in the range of $32\ln n$ integers, the previous discussion guarantees that $G(i)$ is empty with probability at most $\left(1-\frac{3}{32}\right)^{32\ln n}$. Now Lemma 2.1 allows us to write

$$\left(1-\frac{3}{32}\right)^{32\ln n} \quad < \quad e^{-3\ln n} = \frac{1}{n^3}.$$

By (3), it follows that with probability exceeding $1-\frac{1}{4n^2\ln n}$, each of the $\frac{n}{4\ln n}$ groups $G(1), G(2), \ldots,$ $G(\frac{n}{4\ln n})$ contains at least one station.

It is natural, at this point, to initialize the stations in each group $G(i)$, $(1\le i\le\frac{n}{4\ln n})$, individually, using protocol `Reduce(`$32\ln n$`)`. By Lemma 3.5, for each group $G(j)$, this task terminates in at most $64\ln n-2$ time slots with no station being awake for more than

$$2\log(32\ln n) = 2\log\ln n+10 < 2\log\log n+10$$

time slots. The overall number of time slots is $\frac{n}{4\ln n}\cdot(64\ln n-2) = 16n-\frac{n}{2\ln n}$. At this point, each station in group $G(i)$, $(1\le i\le\frac{n}{4\ln n})$, has a local ID within $G(i)$. Finally, the prefix sums of $|G(1)|, |G(2)|, \ldots, |G(\frac{n}{4\ln n})|$ are computed. Clearly, we can use the first station in each group $G(i)$ for this computation. Since, with probability exceeding $1-\frac{1}{n}$, none of the groups is empty, we can use the prefix-sums protocol of Lemma 3.1. This takes $\frac{n}{4\ln n}-1$ time slots with the first station in each group being awake at most two time slots. After that, each station updates its local ID in the obvious way to obtain its final ID. Thus, we proved that by taking advantage of their uniform distribution, the stations with temporary IDs in the range 1 through $8n$ can be initialized, with probability exceeding $1-\frac{1}{n}$, in $16n-\frac{n}{2\ln n}+\frac{n}{4\ln n}-1 = 16n-\frac{n}{4\ln n}-1$ time slots, with no station being awake for more than $2\log\log n+13$ time slots.

6.2 Handling the general case

Unfortunately, we cannot use the technique of Subsection 6.1 to initialize the stations whose temporary IDs are in the range $8n+1$ through $16n$, because the temporary IDs in this range are not uniformly distributed. In particular, we cannot guarantee that in every group of consecutive $c \ln n$ integers, at least one has been assigned as the temporary ID of some station. For convenience we let

Group A: denote all the stations whose temporary IDs are between 1 and $8n$, and

Group B: denote all the stations with temporary IDs between $8n + 1$ and $16n$.

To overcome the aforementioned difficulty, we use the following trick. We "shuffle" the temporary IDs in the whole range 1 to $16n$ such that

- for $1 \leq i \leq 8n$, $i \to 2i - 1$, and
- for $8n + 1 \leq i < 16n$, $i \to 2i - 16n$.

In other words, after the shuffling, the original IDs of stations in group A become odd numbers spread over the whole range 1 to $16n$, while the original IDs of stations in group B become even numbers spread over the whole range 1 to $16n$. It is clear that every station can compute its new ID and that no two stations will have the same ID at the end of the shuffling.

The key insight is that since the IDs of stations in group A were uniformly distributed over 1 to $8n$, the new IDs of stations in group A are still uniformly distributed over the range 1 to $16n$.

Now, partition the stations in the range 1 to $16n$ into $\frac{n}{4 \ln n}$ groups $G'(1), G'(2), \ldots, G'(\frac{n}{4 \ln n})$ of $64 \ln n$ integers each, such that group $G'(i)$, $(1 \leq i \leq \frac{n}{4 \ln n})$, involves stations with IDs in the range $64(j-1) \ln n + 1$ to $64 j \ln n$. Since all the stations in $G(i)$ are included in $G'(i)$, each of the $\frac{n}{4 \ln n}$ groups $G'(1), G'(2), \ldots, G'(\frac{n}{4 \ln n})$ contains at least one station with probability exceeding $1 - \frac{1}{4n^2 \ln n}$.

As before, we initialize the stations in each group $G'(j)$, $(1 \leq j \leq \frac{n}{4 \ln n})$, individually, using protocol `Reduce`$(64 \ln n)$. By Lemma 3.5, for each group $G'(j)$, this task terminates in at most $128 \ln n - 2$ time slots with no station being awake for more than $2 \log(64 \ln n) = 2 \log \ln n + 12 < 2 \log \log n + 12$ time slots. The overall number of time slots is $\frac{n}{4 \ln n} \cdot (128 \ln n - 2) = 32n - \frac{n}{2 \ln n}$. At this point, each station in group $G'(j)$, $(1 \leq j \leq \frac{n}{4 \ln n})$, has a local ID within $G'(j)$. Finally, the prefix sums of $|G'(1)|, |G'(2)|, \ldots, |G'(\frac{n}{4 \ln n})|$ are computed. Clearly, we can use the first station in each group $G'(j)$ for this computation. Since, with probability exceeding $1 - \frac{1}{n}$, none of the groups is empty we can use the prefix-sums protocol of Lemma 3.1. This takes $\frac{n}{4 \ln n} - 1$ time slots with the first station in each group being awake for at most two time slots. After that, each station updates its local ID in the obvious way to obtain the desired final ID. This implies that we can initialize all the stations in $32n - \frac{n}{2 \ln n} + \frac{n}{4 \ln n} - 1 = 32n - \frac{n}{4 \ln n} - 1$ time slots with no station being awake for $2 \log \log n + 15$ time slots.

Recalling that the for loop in protocol `Initialize`(n) takes $16n$ time slots altogether and that no station is awake for more than $\log \log n + 3$ time slots, we have proved the following important result.

Theorem 6.1 With probability exceeding $1 - \frac{1}{n}$, a single-channel, n-station RN with no-CD can be initialized in $O(n)$ time slots, with no station being awake for more than $O(\log \log n)$ time slots.

7 Simulation results

We have simulated the initialization protocol presented in Section 6 in order to get a feel for how well it performs in practice. Since protocol `Reduce`$(64 \ln n)$ is deterministic, we have focused on the for loop. The fundamental motivation for this is that since the running time of protocol `Reduce`$(64 \ln n)$ can be perceived as a fixed overhead, the more stations get their temporary IDs in each iteration of the for loop, the faster will the protocol terminate.

With this in mind, we have simulated the for loop in protocol `Initialize`(n) for $n = 10$, $n = 100$, $n = 1000$, $n = 10000$, $n = 100000$, and $n = 1000000$.

For each value of n as above, we have run the simulation 100000 times. We were particularly interested in the number of stations that failed to receive temporary IDs at the end of particular runs of `Assign-Temporary-ID`. We refer the reader to Table 1 where the results of the simulation are summarized.

For example, taking $n = 100$, the for loop involves $\lceil \log \log n \rceil + 3 = \lceil 5.73 \rceil = 6$ iterations corresponding, respectively, to

- `Assign-Temporary-ID`$(800, 0)$,
- `Assign-Temporary-ID`$(400, 800)$,
- `Assign-Temporary-ID`$(200, 1200)$,
- `Assign-Temporary-ID`$(100, 1400)$,
- `Assign-Temporary-ID`$(50, 1500)$, and
- `Assign-Temporary-ID`$(25, 1550)$.

As mentioned, in order to get the results in the table, this was then repeated 100000 times. The simulation showed that after having run

n	10	100	1000	10000	100000	1000000
$\log\log n + 3$	4.73	5.73	6.31	6.73	7.05	7.31
`Assign-Temporary-ID`$(8n, 0)$	1.07	11.65	117.54	1174.80	11750.93	117516.66
`Assign-Temporary-ID`$(4n, 8n)$	0.0048	0.35	3.43	34.19	340.79	3402.84
`Assign-Temporary-ID`$(2n, 12n)$	0.0024	0.0026	0.0072	0.059	0.58	5.74
`Assign-Temporary-ID`$(n, 14n)$	0.0004	0	0	0.0002	0	0
`Assign-Temporary-ID`$(\frac{n}{2}, 15n)$	0	0	0	0	0	0
`Assign-Temporary-ID`$(\frac{n}{4}, 15.5n)$	0	0	0	0	0	0

Table 1. The number of stations that have not received temporary ID after each iteration: the average of 100000 executions

`Assign-Temporary-ID`$(800, 0)$ an average of 11.65 stations failed to receive temporary IDs.

Likewise, after the subsequent execution of `Assign-Temporary-ID`$(400, 800)$, only $0, 35$ stations have not received temporary IDs. After executing `Assign-Temporary-ID`$(200, 1200)$ only 0.0026 stations were yet to receive temporary IDs. Finally, all the stations received their temporary IDs at the end of `Assign-Temporary-ID`$(100, 1400)$.

The results in Table 1 show that our protocol performs in practice even better than what the analytical model predicted.

8 Initializing a k-channel RN with no-CD

The main goal of this section is to extend the initialization protocol presented in Section 6 to the multichannel case. Consider a k-channel, n-station RN with channels $\mathrm{C}(1), \mathrm{C}(2), \ldots, \mathrm{C}(k)$.

It is straightforward to see that protocol `Assign-Temporary-ID`$(\frac{n}{2^{i-4}}, s)$ when ported to k channels, in the obvious way, terminates in $\lceil \frac{n}{2^{i-4}k} \rceil$ time slots, with no station being awake for more than one time slot. The for loop in protocol `Initialize`(n) takes altogether

$$\sum_{i=1}^{\log\log n+3} \lceil \frac{n}{2^{i-4}k} \rceil \quad < \quad \sum_{i=1}^{\log\log n+3} (\frac{n}{2^{i-4}k} + 1) < \frac{16n}{k} + \log\log n + 3$$

time slots, and no station needs to be awake for more than $\log\log n + 3$ time slots.

At this point, all the stations have received temporary IDs in the range 1 through $16n$. The idea now is to use k channels to: perform the following tasks:

Task 1 initialize
each of the groups $G'(1), G'(2), \ldots, G'(\frac{n}{4\ln n})$ individually, and

Task 2 compute the prefix sums of $|G'(1)|, |G'(2)|, \ldots, |G'(\frac{n}{4\ln n})|$.

As discussed in Subsection 6.2, using one channel, the stations in each group $G'(i)$, $(1 \leq i \leq \frac{n}{4\ln n})$, can be initialized in $128\ln n - 2$ time slots with no station being awake for more than $2\log\log n + 12$ time slots. If $k \leq \frac{n}{4\ln n}$ by assigning the $\frac{n}{4\ln n}$ groups to the k channels as evenly as possible, Task 1 can be performed in $(128\ln n - 2) \cdot \frac{n}{4k\ln n} = \frac{32n}{k} - \frac{n}{2k\ln n}$ time slots with no station being awake for more than $2\log\ln n + 12$ time slots. If $k > \frac{n}{4\ln n}$, then we use $\frac{n}{4\ln n}$ channels, assign one channel to each group, and initialize every group in $128\ln n - 2$ time slots simultaneously. Thus, Task 1 can be done in $O(\frac{n}{k} + \log n)$ time slots with no station being awake for more than $2\log\log n + 12$ time slots.

By Corollary 3.3, Task 2 can be completed in $2\frac{n}{4k\ln n} + O(\log\frac{n}{4\ln n}) = \frac{n}{2k\ln n} + O(\log n)$ time slots with no station being awake for more than eight time slots, regardless the number k of channels. Thus, we have proved the following important result.

Theorem 8.1 With probability at least $1 - \frac{1}{n}$, the k-channel n-station RN can be initialized in $O(\frac{n}{k} + \log n)$ time slots with no station being awake for more than $O(\log\log n)$ regardless of the number k of channels.

9 Concluding remarks and open problems

Our main contribution was to propose energy efficient initialization protocols for individual clusters of a radio network. First, we showed that if the number n of stations is known beforehand, the single-channel RN can be initialized by a protocol that terminates, with probability exceeding $1 - \frac{1}{n}$, in $O(n)$ time slots with no station being awake for more than $O(\log n)$ time slots.

Next, we showed that this initialization protocol can be implemented in such a way that with probability exceeding $1 - \frac{1}{n}$, the protocol terminates in at most $O(n)$ time slots, and no station has to be awake for more than $O(\log\log n)$ time slots.

Finally, we addressed the multi-channel case and showed that if k (≥ 1) channels are available, an n-station RN can be initialized, with probability exceeding $1 - \frac{1}{n}$, in $O(\frac{n}{k} + \log n)$ time slots, with no station being awake for more than $O(\log\log n)$ time slots.

A number of problems remain open. First, the simulation results presented in Section 7 seem to indicate that our protocols terminate much faster than what our proofs show. It would be interesting to see if the constants exhibited in this paper can be improved upon, perhaps by a tighter analysis.

More importantly, however, at this time we do not know how to design energy efficient initialization protocols for the case where the number n of stations is neither known nor a good approximation thereof is available. This promises to be an exciting area for further research.

References

[1] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg, Single-round simulation on radio networks, Journal of Algorithms, 13, (1992), 188–210.

[2] D. J. Baker, Data/voice communication over a multihop, mobile, high frequency network, Proc. MILCOM'97, Monterey, CA, 1997, 339–343.

[3] N. Bambos and J. M. Rulnick, Mobile power management for wireless communication networks, Wireless Networks, 3, (1997), 3–14.

[4] R. Bar-Yehuda, O. Goldreich, and A. Itai, Efficient emulation of single-hop radio network with collision detection on multi-hop radio network with no collision detection, Distributed Computing, 5, (1991), 67–71.

[5] 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, Journal of Computer and Systems Sciences, 45, (1992), 104–126.

[6] U. Black, Mobile and Wireless Networks, Prentice-Hall, Upper Saddle River, NJ, 1996.

[7] J. L. Bordim, J. Cui, T. Hayashi, K. Nakano, and S. Olariu, Energy-efficient initialization protocols for ad-hoc radio networks, Proc. of International Symposium on Algorithms and Computation, Chennai, India, December 1999, LNCS 1741, Springer-Verlag, 1999, 215–224.

[8] P. H. Dana, The geographer's craft project, Dept. of Geography, UT Austin, Sept. 1999, http://www.utexas.edu/depts/grg/gcraf/notes/gps/gps.html.

[9] W. C. Fifer and F. J. Bruno, Low cost packet radio, Proceedings of the IEEE, 75, (1987), 33–42.

[10] M. Gerla and T.-C. Tsai, Multicluster, mobile, multimedia radio network, Wireless Networks, 1, (1995), 255–265.

[11] E. P. Harris and K. W. Warren, Low power technologies: a system perspective, Proc. 3-rd International Workshop on Multimedia Communications, Princeton, 1996.

[12] T. Hayashi, K. Nakano, and S. Olariu, Randomized initialization protocols for Packet Radio Networks, Proc. 13th International Parallel Processing Symposium, San Juan, April 1999, 544–548.

[13] E. D. Kaplan, Understanding GPS: principles and applications, Artech House, Boston, 1996.

[14] C. R. Lin and M. Gerla, Adaptive clustering in mobile wireless networks, IEEE Journal on Selected Areas in Communications, 16, (1997), 1265–1275.

[15] R. Motwani and P. Raghavan, Randomized Algorithms, Cambridge University Press, 1995.

[16] K. Nakano, Optimal initializing algorithms for a reconfigurable mesh, Journal of Parallel and Distributed Computing, 24, (1995), 218-223.

[17] K. Nakano and S. Olariu, Randomized $O(\log\log n)$-round leader election protocols in Packet Radio Networks, Proc. of International Symposium on Algorithms and Computation, (LNCS 1533), 209–218, 1998.

[18] K. Nakano, S. Olariu, and J. L. Schwing, Broadcast-efficient protocols for ad-hoc radio networks with few channels, IEEE Transactions on Parallel and Distributed Systems, 12, (1999), 1276–1289.

[19] M. Joa-Ng and I.-T. Lu, A peer-to-peer zone-based two-level link state routing for mobile ad-hoc networks, IEEE Journal of Selected Areas in Communications, 17, (1999), 1415–1425.

[20] B. Parkinson and S. Gilbert, NAVSTAR: global positioning system – ten years later, Proceedings of the IEEE, 1983, 1177–1186.

[21] R. A. Powers, Batteries for low-power electronics, Proceedings of the IEEE, 83, (1995), 687–693.

[22] R. Ramanathan and M. Steenstrup, Hierarchically-organized, multihop wireless networks for quality-of-service support, Mobile networks and Applications, 3, (1998), 101–119.

[23] D. Raychaudhuri and N. D. Wilson, ATM-based transport architecture for multiservice wireless PCN, IEEE Journal of Selected Areas in Communications, 12, (1994), 1401–1414.

[24] A. K. Salkintzis and C. Chamzas, An in-band power-saving protocol for mobile data networks, IEEE Transactions on Communications, COM-46, (1998), 1194–1205.

[25] R. Sanchez, J. Evans, and G. Minden, Networking on the battlefield: challenges in highly dynamic multihop wireless networks, Proc. of IEEE MILCOM'99, Atlantic City, NJ, October 1999.

[26] K. Sivalingam, M. B. Srivastava, and P. Agrawal, Low power link and access protocols for wireless multimedia networks, Proc. IEEE Vehicular Technology Conference VTC'97, Phoenix, AZ, May, 1997.

[27] M. Stemm, P. Gauthier, and D. Harada, Reducing power consumption on network interfaces in hand-held devices, Proc. 3-rd International Workshop on Multimedia Communications, Princeton, 1996.

[28] J. E. Wieselthier, G. D. Nguyen, and A. Ephemerides, Multicasting in energy-limited ad-hoc wireless networks, Proc. MILCOM'98, 1998.