Energy-Efficient Randomized Routing in Radio Networks *

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

Stephan Olariu
Department of Computer Science
Old Dominion University
Norfolk, Virginia 23529, USA
olariu@cs.odu.edu

ABSTRACT

The main contribution of this work is to present an energy-efficient routing protocol for single-channel, single-hop, radio networks (RN, for short). Such networks are usually deployed in support of special events including search-and-rescue, disaster relief, law enforcement, and collaborative computing, among many others. An RN is typically populated by hand-held, commodity devices, running on batteries. Since recharging batteries may not be possible while on mission, we are interested in designing protocols that are highly energy-efficient. One of the most effective energy-saving strategies is to mandate individual stations to go to sleep whenever they are not transmitting or receiving messages. Consequently, we are interested in protocols that allow stations to power off their transceiver to the largest extent possible.

Suppose that station $S(i)$, $(1 \leq i \leq p)$, of the RN stores s_i items. Each item has a unique *destination* which is the identity of the station to which the item must be routed. The *routing* problem asks to route all the items to their destinations, while expending as little energy as possible. Since, in the worst case, each item must be transmitted at least once, every routing protocol must take at least $n = s_1 + s_2 + \cdots + s_p$ time slots to terminate. Similarly, each station $S(i)$, $(1 \leq i \leq p)$, must be awake for at least $s_i + d_i$ time slots, where d_i denotes the number of items destined for $S(i)$.

It is well known that a station is expending 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. Since, in single-hop radio networks every station is within transmission range from every other station, the design of energy-efficient protocols is highly nontrivial! An additional complication stems from the inherent asymmetry of the routing problem: no destination knows the identity of the sender, precluding à priori arrangements between senders and receivers.

We show that for every $f \geq 1$, the task of routing n items can be completed, with probability exceeding $1 - \frac{1}{f}$, in $n + O(q + \ln f)$ time slots and that no station $\mathrm{S}(i)$, $(1 \leq i \leq p)$, has to be awake for more than $s_i + d_i + O(q_i + r_i \log p + \log f)$ time slots, where q_i is the number of stations that have items destined for $\mathrm{S}(i)$, $q = q_1 + q_2 + \cdots + q_p$, and r_i is the number of stations for which $\mathrm{S}(i)$ has items. Since $q_i \leq d_i$, $r_i \leq s_i$, and $q \leq n$ our protocol is close to optimal.

*Work supported by NASA grant NASI-19858, by NSF grant CCR-9522093, by ONR grant N00014-97-1-0526, and by Grant-in-Aid for Encouragement of Young Scientists (12780213) from Ministry of Education, Science, Sports, and Culture of Japan

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 cellular multimedia services have become commonplace and the demand for enhanced capabilities will continue to grow into the foreseeable future [1, 2, 3]. 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 [3, 5].

Unlike the well-studied cellular systems that assume the existence of a robust infrastructure, *radio networks* (RN, for short) must be rapidly deployable, self-organizing, and capable of multimedia service support. Radio networks suit well the needs specific to disaster relief, search-and-rescue, law enforcement, collaborative computing, and other special-purpose applications [7, 8]. An RN is a distributed system with no central arbiter, consisting of p radio transceivers $\mathrm{S}(1), \mathrm{S}(2), \ldots, \mathrm{S}(p)$, henceforth referred to as *stations*. We assume that each of the p stations has a unique ID in the range 1 to p. This is a reasonable assumption because the task of assigning unique ID numbers to the stations of an RN can be performed very efficiently [6].

The stations of the RN can only communicate through the radio frequency channel. As customary, time is assumed to be slotted and all the stations have a local clock that keeps synchronous time, perhaps by interfacing with a Global Positioning System (GPS, for short). We employ the commonly-accepted assumption that when two or more stations are transmitting in the same time slot, the corresponding packets *collide* and are garbled beyond recognition. Accordingly, in a given time slot the *status* of the channel is:

- *NULL:* if no station transmitted on the channel in the current time slot
- *SINGLE:* if exactly one station transmitted on the channel in the current time slot, and
- *COLLISION:* if two or more stations transmitted on the channel in the current time slot.

We assume that the stations of the RN are simple hand-held devices running on batteries and, therefore, saving power is exceedingly important, as recharging batteries may not be possible while on mission. As it turns out, significant energy is being expended by a station while its transceiver is active that is, while transmitting or receiving a packet. As an example, the DEC Roamabout portable radio consumes about 5.76 watts during transmission, 2.88 watts during reception and 0.35 watts while in idle mode [16]. It may come as a surprise that a station is expending power even if it is receiving a packet that is not destined for it [4, 5, 14, 15, 17]. It is well known that collisions of packets and the need for subsequent retransmission adversely impact the energy efficiency of protocols for RNs. Consequently, we are interested in developing collision-free energy-efficient 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, and
- for each individual station, the total number of awake time slots.

It is relatively straightforward to minimize the overall completion time at the expense of energy consumption. Similarly, one can minimize energy consumption at the expense of completion time [17, 18, 19]. 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.

Suppose that n items are stored by the p stations of an RN, such that station S(i), $(1 \leq i \leq p)$, stores s_i items. Each item has a unique *destination* which is the identity of the station to which the item must be routed. The *routing* problem asks to route all the items to their destinations, as illustrated in Figure 1. We assume that each packet consists of an item along with a few additional bits such as the index of a station, the number of items, etc. For all practical purposes, this prohibits large aggregates from being sent in one packet.

Since, in the worst case, every item must be transmitted at least once, every routing protocol must take at least $n = s_1 + s_2 + \cdots + s_p$ time slots to terminate. Further, each station $S(i)$, $(1 \leq i \leq p)$, must be awake for at least $s_i + d_i$ time slots, where d_i denotes the number of items destined for $S(i)$.

If energy efficiency is not an issue, routing on the RN is rather straightforward and has been discussed in the literature [10, 11, 12, 13]. Indeed, consider an instance of the routing problem involving n items stored, in some order, by the p stations of an RN. Taking turns, each station transmits its items, one by one, and every station tunes to the channel. The station whose identity matches the destination of the item being transmitted copies the item in its local memory. It should be clear that this simple protocol terminates in exactly n time slots and is, therefore, optimal. However, the protocol is not energy-efficient since every station has to monitor the channel for n time slots.

The main contribution of this work is to present an energy-efficient routing protocol for single-channel, single-hop radio networks. For every $f \geq 1$, our protocol terminates, with probability exceeding $1 - \frac{1}{f}$, in $n + 5q - 3p + 8 \ln f + 2\sqrt{2(q-p)\ln f}$ time slots; moreover, no station S(i), $(1 \leq i \leq p)$, has to be awake for more than $s_i + d_i + O(q_i + r_i \log p + \log f)$ time slots, where each q_i is the number of stations that has items destined for S(i), $q = q_1 + q_2 + \cdots q_p$, and r_i is the number of destination stations for which S(i) has items. Since $q_i \leq d_i$, $r_i \leq s_i$, and $q \leq n$ our protocol is close to optimal.

The remainder of this extended abstract is organized as follows: Section 2 discusses a basic idea of our energy-efficient routing protocol, that allows each station to determine the exact time slot when it should wake up to transmit its items. Using this idea, we also present a simple deterministic routing protocol that terminates in $n + p^2 - 1$ time slots, with station S(i) being awake for $s_i + d_i + 2p + 1$ time slots. This protocol is efficient when $p = O(\sqrt{n})$. However, for large p the protocol is not very efficient. Finally, in Section 3 we discuss the details of our energy-efficient routing protocol.

2. A RESERVATION-BASED ROUTING PROTOCOL

The main goal of this section is to present a simple reservation-based protocol that will serve as motivation for the design of an energy-efficient routing protocol. At the end of the *reservation protocol* every station knows the exact time slots when its turn has come to transmit on the channel.

Let $\alpha(i,j)$, $(1 \leq i,j \leq p)$, be the set of items stored by station S(i) whose final destination is station S(j). Write, further,

$$\begin{aligned}\alpha(*,j) &= \alpha(1,j) \cup \alpha(2,j) \cup \cdots \cup \alpha(p,j), \text{ and} \\ \alpha(i,*) &= \alpha(i,1) \cup \alpha(i,2) \cup \cdots \cup \alpha(i,p).\end{aligned}$$

In other words, $\alpha(*,j)$ is the set of items whose final destination is station S(j); and, $\alpha(i,*)$ is the set of items stored initially by station S(i). Clearly, our job is to route for every j, $(1 \leq j \leq p)$, the items in $\alpha(*,j)$ to station S(j). Observe that $s_i = |\alpha(i,*)|$ and $d_i = |\alpha(*,i)|$ denote, respectively, the number of items stored by station $S(i)$ at the beginning and at the end of the protocol.

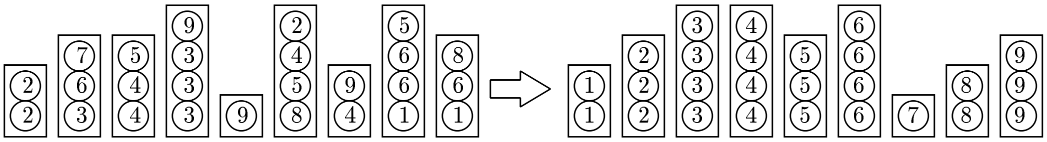

Figure 1: ***An instance of the routing problem on a 9-station RN***

The routing algorithm mentioned before can be described as follows:

Protocol `Simple_Routing`
for $j \leftarrow 1$ to p do
  for $i \leftarrow 1$ to p do
    station S(i) transmits, one after the other,
      the items in $\alpha(i, j)$;
    station S(j) monitors the channel and
      picks up the items in $\alpha(i, j)$;
  endfor
endfor

By tracing through the protocol `Naive_Routing`, it is easy to see that

- for every i and j, $(1 \leq i, j \leq p)$, station S(i) begins transmitting the items in $\alpha(i, j)$ in time slot $|\alpha(*, 1)| + |\alpha(*, 2)| + \cdots + |\alpha(*, j-1)| + |\alpha(1, j)| + |\alpha(2, j)| + \cdots |\alpha(i-1, j)| + 1$,
- for every j, $(1 \leq j \leq p)$, station S(j) starts picking up the items in $\alpha(*, j)$ in time slot $|\alpha(*, 1)| + |\alpha(*, 2)| + \cdots + |\alpha(*, j-1)| + 1$.

Thus, as soon as station S(i), $(1 \leq i \leq p)$, learns the values of

- the *global prefix sum* defined as $|\alpha(*, 1)| + |\alpha(*, 2)| + \cdots + |\alpha(*, i-1)|$, and
- for every j, $(1 \leq j \leq p)$, the *local prefix sum* defined as $|\alpha(1, j)| + |\alpha(2, j)| + \cdots |\alpha(i-1, j)|$

it can determine the exact time slot when it can start transmitting the items it holds and, by the same token, the time slot when it should start picking up the items destined for it. Altogether, station S(i) will be transmitting for $s_i = |\alpha(i, *)|$ time slots and will be receiving items for $d_i = |\alpha(*, i)|$ time slots. To summarize, we state the following simple result.

LEMMA 2.1. *If every station* S(i), $(1 \leq i \leq p)$, *knows the values of the global and local prefix sums above, the routing can be performed in n time slots with each* S(i) *being awake for exactly $s_i + d_i$ time slots.*

Suppose that station S(i), $(1 \leq i \leq p)$, has a number x_i and that we are interested in computing the prefix sums $x_1 + x_2 + \cdots + x_i$. To begin, station S(1) transmits x_1. 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 picks up x_1. Now, S(2) computes $x_1 + x_2$ and sends the result to S(3). In this time slot all the stations are asleep except for S(2) and S(3). Upon receiving $x_1 + x_2$, S(3) computes $x_1 + x_2 + x_3$ and transmits the result to S(4). This is then continued until, at the end of $p-1$ time slots, S(p) receives $x_1 + x_2 + x_3 + \cdots + x_{p-1}$. Thus, the prefix sums can be computed in $p-1$ time slots. Further, S(1) and S(p) are awake for one time slot, while all the other stations are awake for two time slots.

By executing the prefix sums protocol for $|\alpha(1, j)|$, $|\alpha(2, j)|$, $\ldots$, $|\alpha(p, j)|$, all the local prefix sums can be computed in $p(p-1)$ time slots. Further, S(1) and S(p) are awake for p time slots while all the other stations are awake for $2p$ time slots. At this moment, S(p) knows the values of $|\alpha(*, 1)|$, $|\alpha(*, 2)|$, $\ldots$, $|\alpha(p, 1)|$. Next, S(p) computes all the global prefix sums, and transmits them one by one on the channel. By tuning to the channel at the right moment, each station can easily determine the global prefix sum that is of interest to it. This latter task takes $p-1$ time slots with S(p) being awake for $p-1$ time slots and all the other stations are awake for one time slot. Therefore, we have the following result.

LEMMA 2.2. *Every station* S(i), $(1 \leq i \leq p)$, *can learn the values of the global and local prefix sums in $p^2 - 1$ time slots, with no station being awake for more than $2p+1$ time slots.*

Now, Lemma 2.1 and Lemma 2.2 combined, imply the following result.

LEMMA 2.3. *Consider a single-channel, single-hop radio network with p stations. The task of routing n items can be performed in $n + p^2 - 1$ time slots, with station* S(i), $(1 \leq i \leq p)$, *being awake for $s_i + d_i + 2p + 1$ time slots.*

3. RANDOMIZED ROUTING FOR RADIO NETWORKS

The main purpose of this section is to exhibit a randomized routing protocol for single-channel, single-hop radio networks. This first protocol is not energy-efficient since every station has to monitor the channel for the duration of the protocol. In the next section, we show how to modify this protocol to make it energy efficient.

Throughout, $\Pr[A]$ will denote the probability of event A. For a random variable X, $E[X]$ denotes the expected value of X. Let X be a random variable denoting the number of successes in n independent Bernoulli trials with parameters p and $1-p$. It is well known that X has a *binomial distribution* and that for every r, $(0 \le r \le n)$,

$$\Pr[X = r] \;=\; \binom{n}{r} p^r (1-p)^{n-r}. \tag{1}$$

Further, the expected value of X is given by

$$E[X] \;=\; \sum_{r=0}^{n} r \cdot \Pr[X = r] = np. \tag{2}$$

To analyze the tail of the binomial distribution, we shall make use of the following estimates, commonly referred to as *Chernoff bound* [9]:

$$\Pr[X \le (1-\epsilon)E[X]] \;\le\; e^{-\frac{\epsilon^2}{2}E[X]} \qquad (0 \le \epsilon \le 1). \tag{3}$$
$$\tag{4}$$

For convenience, we inherit the notation and terminology of Section 2. If p is large, for most values of i and j the set $\alpha(i,j)$ is empty. Thus, it makes sense to avoid computing prefix sums for the sets $\alpha(i,j)$ that are empty.

Suppose that m stations $\mathrm{S}(i_1), \mathrm{S}(i_2), \ldots, \mathrm{S}(i_m)$, $(1 \le i_1 < i_2 < \cdots < i_m \le p)$, have items to be routed to station $\mathrm{S}(1)$. In order for station $\mathrm{S}(i_j)$, $(1 \le j \le m)$, to know when to start transmitting its items, it must learn its *relative* position within $\mathrm{S}(i_1), \mathrm{S}(i_2), \ldots, \mathrm{S}(i_m)$. Once each $\mathrm{S}(i_j)$ knows its relative position, they can run a reservation protocol by computing in $m-1$ time slots the prefix sums of $|\alpha(i_1,1)|, |\alpha(i_2,1)|, \ldots, |\alpha(i_m,1)|$.

The problem of determining the relative position of station $\mathrm{S}(i_j)$ within $\mathrm{S}(i_1), \mathrm{S}(i_2), \ldots, \mathrm{S}(i_m)$ can be solved efficiently by assigning each station $\mathrm{S}(i_j)$, $(1 \le j \le m)$, a unique ID in the range 1 to m. We caution the reader that these IDs should not to be confused with the permanent IDs of the stations. The newly assigned IDs are *temporary* and serve a local purpose only.

More generally, at this point we develop a randomized initialization protocol that assigns unique IDs in the range 1 to $|P|$ to a subset P of stations. The initialization protocol first determines whether $|P| = 0$, $|P| = 1$, or $|P| \ge 2$. Clearly, this only involves mandating all the stations in P to transmit in a time slot and checking the status of the channel. If $|P| = 1$ then the unique station in P is assigned the ID of 1. If $|P| \ge 2$ then P is partitioned into two subsets P_1 and P_2. The above is then repeated for P_1 and P_2 until, eventually, every subset contains a unique station. The details are spelled out as follows:

Protocol `Initialize(`P`)`
$N \leftarrow 0$;
$P_1 \leftarrow P$;
$L \leftarrow 1$;
while $L \ge 1$ **do**
  $l \leftarrow 0$;
  for $i \leftarrow 1$ to L do
    if $|P_i| = 1$ **then**
      $N \leftarrow N + 1$;
      the unique station in P_i receives the ID of N
    else if $|P_i| \ge 2$
      every station in P_i flips a fair coin;
      $P'_{l+1} \leftarrow$ {stations flipping heads};
      $P'_{l+2} \leftarrow$ {stations flipping tails};
      $l \leftarrow l + 2$;
    endif
  endfor
  $L \leftarrow l$;
  $P_1 \leftarrow P'_1$, $P_2 \leftarrow P'_2$, … $P_L \leftarrow P'_L$
endwhile

It is easy to confirm that protocol `Initialize` correctly assigns unique IDs to all the stations in P. We refer the reader to Figure 2 for an illustration of `Initialize(`P`)`. We call the tree in this figure the *split* tree of P. Initially, P_1 contains five stations. Since $|P_1| \ge 2$, the first iteration of the **while** loop partitions P_1 into two subsets P_1 and P_2. In the second iteration, since both P_1 and P_2 have two or more stations, P_1 is partitioned into P_1 and P_2, while P_2 is partitioned into P_3 and P_4. In the third iteration, P_2 is ignored because it is empty, and the unique station in P_4 is assigned an ID of 1. P_1 and P_3 are partitioned again. In the fourth iteration, since P_1 and P_2 have one station each, they are assigned the IDs of 2 and 3, respectively. Further, P_4 is partitioned into P_1 and P_2. In the fifth iteration, the stations in P_1 and P_2 are assigned the IDs of 4 and 5, respectively.

Next, we discuss the implementation of `Initialize(`P`)`. We assume that each station in P stores global variables N, L, l, and i. Each iteration of the `for` loop is said to be a *trial* and corresponds to a node in the split tree. In each trial, we need to check whether $|P_i| = 0$, or $|P_i| = 1$, or $|P_i| \ge 2$. For this purpose, all stations in P_i transmit on the channel. If the status of the channel is NULL then $|P_i| = 0$, if the status is SINGLE then $|P_i| = 1$, and if the status is COLLISION then $|P_i| \ge 2$. Since the values of N and l are changed depending on the status of the channel, every station must constantly monitor the channel. Clearly, each trial is completed in one time slot. It follows that the number of time slots necessary to complete the initialization must equal the number of nodes in the split tree. The internal nodes of the split tree contain two or more stations, while the leaves are either empty or contain exactly one station. We say that an internal node is *good* if both its children are non-empty. It is clear that the split tree has exactly $|P| - 1$ good internal nodes. An internal node containing k stations $(k \ge 2)$ is good with probability at least

$$1 - 2\left(\frac{1}{2}\right)^k \ge \frac{1}{2}.$$

Let $P(j) = \{\mathrm{S}(i) \mid \alpha(i,j) \ne \emptyset\}$ denote the set of stations that have items destined for $\mathrm{S}(j)$. Further, let $q_j = |P(j)|$ and $q = q_1 + q_2 + \cdots + q_p$. Suppose that the calls `Initialize(`$P(1)$`)`, …, `Initialize(`$P(p)$`)` are performed

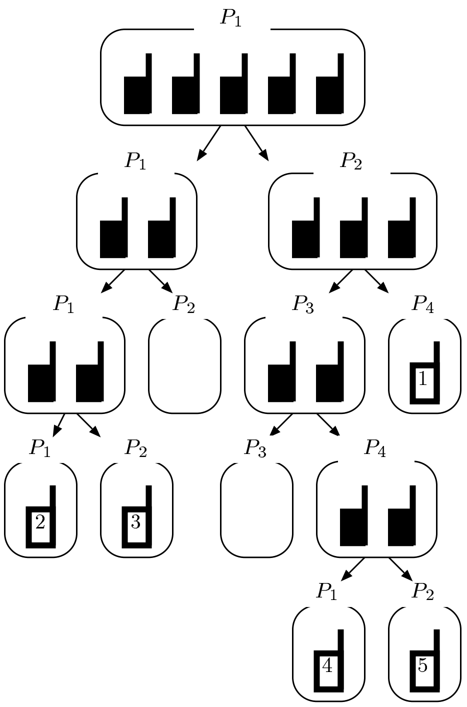

Figure 2: *Illustrating protocol* `Initialize`

sequentially. We are interested in evaluating the overall number of time slots necessary to complete these p initializations. For this purpose, we evaluate the total number of nodes in the p split trees. As discussed, the split tree corresponding to $P(i)$ has exactly $q_i - 1$ good internal nodes. Thus, the p split trees combined have exactly $\sum_{i=1}^{p}(p_i - 1) = q - p$ good internal nodes.

Let X be the number of good internal nodes among $2(q-p)+4\ln f+\sqrt{2(q-p)\ln f}$ internal nodes of these split trees. Since an internal node is good with probability at least $\frac{1}{2}$, $E[X] \geq (q-p) + 2\ln f + \frac{1}{2}\sqrt{2(q-p)\ln f}$. Hence, writing

$$0 < \epsilon = \frac{2}{1+\sqrt{1+\frac{2(q-p)}{\ln f}}} \leq 1$$

the probability that the p split trees have fewer than $q-p$ good nodes is at most

$$\begin{aligned} \Pr[X < q-p] &\leq \Pr[X < (1-\epsilon)E[X]] \\ &< e^{-\frac{\epsilon^2 E[X]}{2}} \qquad \text{(from (3))} \\ &\leq e^{-\ln f} = \frac{1}{f} \end{aligned}$$

Thus, with probability exceeding $1-\frac{1}{f}$, the p split trees have at most $2(q-p)+4\ln f+\sqrt{2(q-p)\ln f}$ internal nodes, and $2(q-p)+4\ln f - 1 + \sqrt{2(q-p)\ln f}$ leaves. It follows that with probability exceeding $1-\frac{1}{f}$, the calls `Initialize`$(P(1))$, `Initialize`$(P(2))$, ..., `Initialize`$(P(p))$ terminate in at most $4(q-p)+8\ln f+2\sqrt{2(q-p)\ln f}$ time slots.

Once the initialization is out of the way, the task of routing the items proceeds as in the deterministic protocol discussed in Section 2. It is easy to see that local prefix sums can be computed in q time slots, while the global prefix sums can be broadcast in $p-1$ time slots. Therefore, by Lemma 2.1, the routing protocol terminates in

$$\begin{aligned} & n + 4(q-p) + 8\ln f + 2\sqrt{2(q-p)\ln f} + q + p - 1 \\ < \quad & n + 5q - 3p + 8\ln f + 2\sqrt{2(q-p)\ln f} \end{aligned}$$

time slots. Consequently, we have proved the following theorem.

LEMMA 3.1. *Consider a single-channel, single-hop radio network with p stations. For every $f \geq 1$, the task of routing n items can be performed with probability exceeding $1-\frac{1}{f}$, in at most $n + O(q + \ln f)$ time slots.*

Since $q = q_1 + q_2 + \cdots q_p \leq n$, our protocol is efficient in terms of the overall completion time.

4. A RANDOMIZED ENERGY-EFFICIENT ROUTING PROTOCOL.

This section is devoted to making the routing protocol of Section 3 energy-efficient. Recall that the protocol in Section 3 involved the sequential execution of `Initialize`$(P(1))$, `Initialize`$(P(2))$, ..., `Initialize`$(P(p))$. Since the number of time slots necessary to complete `Initialize`$(P(i-1))$ is random, the stations involved in `Initialize`$(P(i))$ need to detect the termination of `Initialize`$(P(i-1))$. In other words, these stations need to constantly monitor the channel, thus consuming energy. To address this shortcoming, we plan to interleave the calls by essentially traversing the corresponding split forest in breadth-first order. The details are spelled out as follows.

Protocol `Interleaved_Initialize`$(P(1), P(2), \ldots, P(p))$
$P_1 \leftarrow P(1)$, $P_2 \leftarrow P(2)$, ..., $P_p \leftarrow P(p)$;
$l_0 \leftarrow 0$, $l_1 \leftarrow 1$, $l_2 \leftarrow 2$, ..., $l_p \leftarrow p$;
$N_1 \leftarrow N_2 \leftarrow \cdots \leftarrow N_p \leftarrow 0$;
while $L \geq 1$ **do**
  $l \leftarrow 0$;
  for $i \leftarrow 1$ to p do $(\star)$
    for $j \leftarrow l_{i-1}+1$ to l_i do
      if $|P_j| = 1$ **then**
        $N_i \leftarrow N_i + 1$;
        the unique station in P_j receives the ID of N_i;
      else if $|P_j| \geq 2$
        the stations in P_j flip a fair coin;
        $P'_{l+1} \leftarrow$ {stations flipping heads};
        $P'_{l+2} \leftarrow$ {stations flipping tails};
        $l \leftarrow l + 2$;
      endif
      $l'_i \leftarrow l$;
    endfor
  endfor
  $L \leftarrow l$;
  $P_1 \leftarrow P'_1$, $P_2 \leftarrow P'_2$, ... $P_L \leftarrow P'_L$;
  $l_1 \leftarrow l'_1$, $l_2 \leftarrow l'_2$, ... $l_p \leftarrow l'_p$;
endwhile

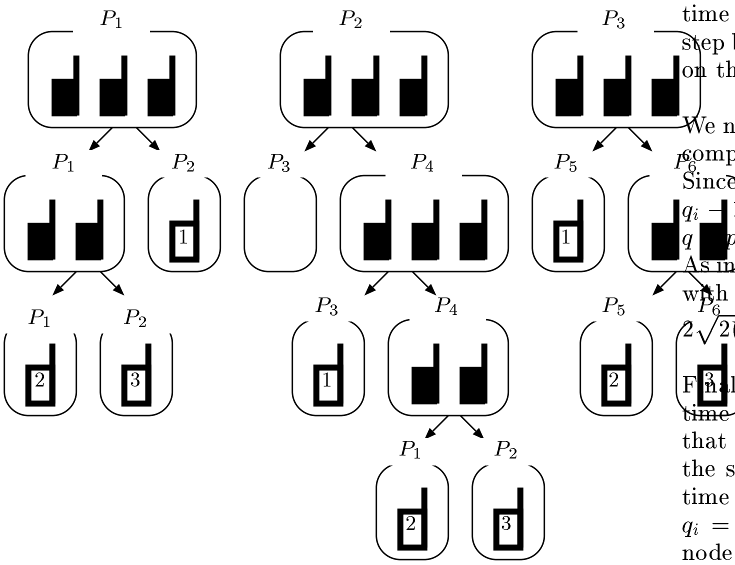

Figure 3: *Illustrating the split forest corresponding to* $P(1)$, $P(2)$, $P(3)$

We refer to an iteration of the outer **for** loop ($\star$) as a *round.* A single round suffices to determine for each P_j, $(1 \leq j \leq L)$, whether $|P_j| = 0$, $|P_j| = 1$, or $|P_j| \geq 2$. If $|P_j| = 1$, the unique station in P_j will receive its own ID. Otherwise, if P_j has two or more stations it is further partitioned. To understand the role played by l_i, note that at the beginning of a certain round the stations in $P(i)$ are partitioned into $P_{l_{i-1}+1}, P_{l_{i-1}+2}, \ldots, P_{l_i}$. At the end of the round, $P_{l_{i-1}+1}, P_{l_{i-1}+2}, \ldots, P_{l_i}$ are partitioned into $P'_{l'_{i-1}+1}, P'_{l'_{i-1}+2}, \ldots, P'_{l'_i}$. The variable N_i, $(1 \leq i \leq p)$, is used to denote the number of stations in P_i that have been already initialized. We refer the reader to Figure 3 for an illustration of the split forest corresponding to $P(1)$, $P(2)$, $P(3)$, each containing three stations.

We turn to the implementation of `Interleaved_Initialize`. Station $\mathrm{S}(i)$, $(1 \leq i \leq p)$, is responsible for maintaining the split tree rooted at P_i. For this purpose, each $\mathrm{S}(i)$ stores the value of N_i, l, l_{i-1}, l_i, l'_{i-1}, and l'_i. Each round is implemented in $2L$ time slots such that the time slots $2j-1$ and $2j$, $(1 \leq j \leq L)$, are used for P_j. Station $\mathrm{S}(i)$, $(1 \leq i \leq p)$, is also responsible for handling the subsets $P_{l_{i-1}+1}, P_{l_{i-1}+2}, \ldots, P_{l_i}$. For each j satisfying $l_{i-1}+1 \leq j \leq l_i$, the first time slot $2j-1$ is used to check whether $|P_j| = 0$, $|P_j| = 1$, or $|P_j| \geq 1$. Next, station $\mathrm{S}(i)$ updates the values of N_i, l, l'_{i-1}, and l'_i according to the status of channel. In time slot $2j$, station $\mathrm{S}(i)$ transmits on the channel the values of N_i and l. The stations in P_i monitor the channel and pick up N_i and l. If P_i is partitioned, then it will be partitioned into P_{l-1} and P_l. If P_i is a singleton, then the unique station receives ID N_i. Note that station $\mathrm{S}(i)$, $(2 \leq i \leq p,)$ has to monitor the channel in time slot $2l_{i-1}$ to pick up the value of l from $\mathrm{S}(i-1)$. Further, since every station has to know L, it must monitor the channel in the last time slot of every round. We refer the reader to Table 1 for a step by step execution of protocol `Interleaved_Initialize`, on the forest illustrated in Figure 3.

We now evaluate the total number of time slots necessary to complete `Interleaved_Initialize`$(P(1), P(2), \ldots, P(p))$. Since each $P(i)$, $(1 \leq i \leq p)$, has q_i stations, it has exactly $q_i - 1$ good internal nodes. Hence, overall, $\sum_{i=1}^{p}(q_i - 1) = q - p$ good internal nodes suffice to complete the protocol. As in Section , it can be shown that the protocol terminates, with probability exceeding $1 - \frac{1}{f}$, in $n + 5q - 3p + 8\ln f + 2\sqrt{2(q-p)\ln f}$ time slots.

Finally, let us turn to the task of evaluating the number of time slots during which each station must be awake. Recall that station $\mathrm{S}(i)$, $(1 \leq i \leq p)$, is responsible for managing the split tree of $P(i)$. Thus, $\mathrm{S}(i)$ has to be awake for two time slots for each node in the split tree of $P(i)$. Since $q_i = |P(i)|$, the split tree of $P(i)$ has $q_i - 1$ good internal node.

Let X be the random variable denoting the number of good nodes among $2(q_i-1) + 4\ln pf + \sqrt{2(q_i-1)\ln pf}$ internal nodes of the split tree of $P(i)$. As we discussed, since an internal node is good with probability at least $\frac{1}{2}$, $E[X] \geq (q_i - 1) + 2\ln pf + \frac{1}{2}\sqrt{2(q_i-1)\ln pf}$ Hence, letting

$$0 < \epsilon = \frac{2}{1 + \sqrt{1 + \frac{2(q_i-1)}{\ln pf}}} \leq 1$$

the probability that the split tree of $P(i)$ has fewer than $q_i - 1$ good internal nodes is at most

$$\begin{aligned} \Pr[X < q_i - 1] &\leq \Pr[X < (1-\epsilon)E[X]] \\ &< e^{-\frac{\epsilon^2 E[X]}{2}} \quad \text{(from (3))} \\ &\leq e^{-\ln pf} = \frac{1}{pf} \end{aligned}$$

Thus, with probability exceeding $1 - \frac{1}{f}$, the split tree of $P(i)$ has at most $2(q_i-1) + 4\ln pf + \sqrt{2(q_i-1)\ln pf}$ internal nodes, and $2(q_i-1) + 4\ln pf - 1 + \sqrt{2(q_i-1)\ln pf}$ leaves. Since for each node of the split tree station $S(i)$ must be awake for two time slots, the task of managing $P(i)$ forces station $\mathrm{S}(i)$ to be awake, with probability exceeding $1 - \frac{1}{pf}$, during $8(q_i-1) + 16\ln pf + 4\sqrt{2(q_i-1)\ln pf} = O(q_i + \log p + \log f)$ time slots.

Station $\mathrm{S}(i)$ must also be awake to participate in the protocols local to the split trees in which it is the destination of items. Specifically, let $r_i = |\{\mathrm{S}(j) \mid \alpha(i,j) \neq \emptyset\}|$ be the number of stations that have items destined for $\mathrm{S}(i)$. Clearly, $\mathrm{S}(i)$ belongs to exactly r_i of the split trees rooted at $P_1, P_2, \ldots, P_p$. We say that an internal node with k $(k \geq 2)$ stations containing $\mathrm{S}(i)$ is *successful* if $\mathrm{S}(i)$ also belongs to its child with at least $\lceil \frac{k}{2} \rceil$ stations. Clearly, the probability of an internal node being successful is at least $\frac{1}{2}$. Since each root of the split tree has at most p stations, $\log p$ successful internal nodes suffice for $\mathrm{S}(i)$ to reach a leaf in a particular split tree. Since $\mathrm{S}(i)$ belongs to r_i such trees, $r_i \log p$ successful internal nodes suffice.

Table 1: ***A synopsis of*** `Interleaved_Initialize`

time slots	stations transmitting	stations monitoring	channel status	data transferred	event occurs
1	P_1	P_1, S(1)	COLLISION	-	
2	S(1)	P_1, S(2)	SINGLE	$(N_1 = 0, l = 2)$	$P_1 \rightarrow P'_1, P'_2$
3	P_2	P_2, S(2)	COLLISION	-	
4	S(2)	P_2, S(3)	SINGLE	$(N_2 = 0, l = 4)$	$P_2 \rightarrow P'_3, P'_4$
5	P_3	P_3, S(3)	COLLISION	-	
6	S(3)	ALL	SINGLE	$(N_3 = 0, l = 6)$	$P_3 \rightarrow P'_5, P'_6$
$P_1 \leftarrow P'_1$, $P_2 \leftarrow P'_2$, $P_3 \leftarrow P'_3$, $P_4 \leftarrow P'_4$, $P_5 \leftarrow P'_5$, $P_6 \leftarrow P'_6$ $l_1 \leftarrow l'_1$, $l_2 \leftarrow l'_2$, $l_3 \leftarrow l'_3$, $L \leftarrow l$					
7	P_1	P_1, S(1)	COLLISION	-	
8	S(1)	P_1	SINGLE	$(N_1 = 0, l = 2)$	$P_1 \rightarrow P'_1, P'_2$
9	P_2	P_2, S(1)	SINGLE	-	
10	S(1)	P_2, S(2)	SINGLE	$(N_1 = 1, l = 2)$	ID 1 for $P(1)$ issued
11	P_3	P_3, S(2)	NULL	-	
12	S(2)	P_3	SINGLE	$(N_2 = 0, l = 2)$	
13	P_4	P_4, S(2)	COLLISION	-	
14	S(2)	P_4, S(3)	SINGLE	$(N_2 = 0, l = 4)$	$P_4 \rightarrow P'_3, P'_4$
15	P_5	P_5, S(3)	SINGLE	-	
16	S(3)	P_5	SINGLE	$(N_2 = 1, l = 4)$	ID 1 for $P(3)$ issued
17	P_6	P_6, S(3)	COLLISION	-	
18	S(3)	ALL	SINGLE	$(N_3 = 1, l = 6)$	$P_6 \rightarrow P'_5, P'_6$
$P_1 \leftarrow P'_1$, $P_2 \leftarrow P'_2$, $P_3 \leftarrow P'_3$, $P_4 \leftarrow P'_4$, $P_5 \leftarrow P'_5$, $P_6 \leftarrow P'_6$ $l_1 \leftarrow l'_1$, $l_2 \leftarrow l'_2$, $l_3 \leftarrow l'_3$, $L \leftarrow l$					
19	P_1	P_1, S(1)	SINGLE	-	
20	S(1)	P_1	SINGLE	$(N_1 = 2, l = 0)$	ID 2 for $P(1)$ issued
21	P_2	P_2, S(1)	SINGLE	-	
22	S(1)	P_2, S(2)	SINGLE	$(N_1 = 3, l = 0)$	ID 3 for $P(1)$ issued
23	P_3	P_3, S(2)	SINGLE	-	
24	S(2)	P_3	SINGLE	$(N_2 = 1, l = 0)$	ID 1 for $P(2)$ issued
25	P_4	P_4, S(2)	COLLISION	-	
26	S(2)	P_4, S(3)	SINGLE	$(N_2 = 0, l = 2)$	$P_3 \rightarrow P'_1, P'_2$
27	P_5	P_5, S(3)	SINGLE	-	
28	S(3)	P_5	SINGLE	$(N_3 = 2, l = 2)$	ID 2 for $P(3)$ issued
29	P_6	P_6, S(3)	SINGLE	-	
30	S(3)	ALL	SINGLE	$(N_3 = 3, l = 2)$	ID 3 for $P(3)$ issued
$P_1 \leftarrow P'_1$, $P_2 \leftarrow P'_2$, $l_1 \leftarrow l'_1$, $l_2 \leftarrow l'_2$, $l_3 \leftarrow l'_3$, $L \leftarrow l$					
31	P_1	P_1, S(2)	SINGLE	-	
32	S(2)	P_1	SINGLE	$(N_2 = 2, l = 0)$	ID 2 for $P(2)$ issued
33	P_2	P_2, S(2)	SINGLE	-	
34	S(2)	ALL	SINGLE	$(N_2 = 3, l = 0)$	ID 3 for $P(2)$ issued

Let Y be the number of successful internal nodes among $4r_i \log p + 16 \log p + 16 \log f$ internal nodes of these r_i trees. Clearly, $E[Y] \geq 2r_i \log p + 8 \log p + 8 \log f$ and we have,

$$\begin{aligned}\Pr[Y < r_i \log p] &< \Pr[Y < \frac{1}{2}E[X]] \\ &< e^{-\frac{1}{8}E[X]} \qquad \text{(from 3)} \\ &< e^{-(\log p + \log f)} < \frac{1}{pf}.\end{aligned}$$

Thus, with probability exceeding $1 - \frac{1}{pf}$, station $\mathrm{S}(i)$ is included at most $O(r_i \log p + \log p + \log f)$ internal nodes. It follows that, with probability exceeding $1 - \frac{1}{pf}$, station $\mathrm{S}(i)$ must be awake for at most $O(r_i \log q + \log f)$ time slots to be initialized.

Altogether, therefore, station $\mathrm{S}(i)$ is awake for at most $O(q_i + r_i \log p + \log f)$ time slots with probability at least $1 - \frac{1}{pf}$, and therefore, no station is awake for more than $O(q_i + r_i \log p + \log f)$ time slots with probability at least $1 - \frac{1}{f}$.

After the initialization, the global and local prefix sums are computed and then the items are routed as before. For the local prefix sums, each $\mathrm{S}(i)$ is awake for at most $2r_i$ time slots, and for global prefix sums, it is awake for at most two time slots. Therefore, by Lemma 2.1, $\mathrm{S}(i)$ is awake for

$$s_i + d_i + O(q_i + r_i \log p + \log f)$$

time slots. Consequently, we have, proved the following theorem.

THEOREM 4.1. *Consider a single-channel, single-hop radio network with p stations. For every $f \geq 1$, the routing of n items can be completed, with probability exceeding $1 - \frac{1}{f}$, in $n + O(q + \ln f)$ time slots; moreover, no station $\mathrm{S}(i)$, $(1 \leq i \leq p)$, has to be awake for more than $s_i + d_i + O(q_i + r_i \log p + \log f)$ time slots.*

5. REFERENCES

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

[2] D. Bertzekas and R. Gallager, *Data Networks*, Second Edition, Prentice-Hall, 1992.

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

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

[5] E. P. Harris and K. W. Warren, Low power technologies: a system perspective, *Proc. Third International Workshop on Mobile Multimedia Communications*, Princeton, NJ, September 1996.

[6] J. L. Bordim, J.Cui, T. Hayashi, K. Nakano, and S. Olariu, Energy-efficient initialization protocols for ad-hoc radio networks, *Proc. ISAAC'99*, 215–224.

[7] C. R. Lin and M. Gerla, Adaptive clustering for mobile wireless networks, *IEEE Transaction on Special Areas of Communications*, (1999), 1265–1275.

[8] W. Mangione-Smith and P. S. Ghang, A low power medium access control protocol for portable multimedia devices, *Proc. Third International Workshop on Mobile Multimedia Communications*, Princeton, NJ, September 1996.

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

[10] K. Nakano, Optimal sorting algorithms on bus-connected processor arrays, *IEICE Transactions Fundamentals*, E-76A, 11, (1994), 2008–2015.

[11] K. Nakano, S. Olariu, and J.L. Schwing, Broadcast-efficient sorting in the presence of few channels, *Proc. International Conference on Parallel Processing*, August 1997, 12–15.

[12] K. Nakano, S. Olariu, and J. Schwing, Broadcast-Efficient Algorithms on the Coarse-Grain Broadcast Communication Model with Few Channels, *Proc. International Parallel Processing Symposium*, (1998), pp. 31–35.

[13] K. Nakano, S. Olariu, and J. L. Schwing, Broadcast-efficient protocols for mobile radio networks, *IEEE Transactions on Parallel and Distributed Systems*, 10, (1999), 1276–1289.

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

[15] 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.

[16] S. Singh and C. S. Raghavendra, PAMAS – Power aware multi-access protocol with signalling for ad-hoc networks, *ACM Computer Communication Review*, 28, (1998), 5–26.

[17] 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.

[18] 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.

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