Randomized Leader Election Protocols for Ad-hoc Networks *

K. NAKANO
Nagoya Institute of Technology, Japan

S. OLARIU
Old Dominion University, U.S.A.

Abstract

Unlike the well-studied cellular networks that assume the existence of a robust infrastructure, Ad-hoc Networks (AHN, for short) are rapidly deployable, self-organizing, and do not rely on an existing infrastructure. These networks find applications to disaster-relief, search-and-rescue, law-enforcement, and collaborative computing, among others.

The main contribution of this work is to propose an extremely fast leader election protocol for AHNs. We begin by showing that for every $f \geq 1$, a single-hop single-channel n-station AHN can elect a leader among n identical stations in $\log\log n + 2.78\log f + o(\log\log n + \log f)$ time slots with probability at least $1 - \frac{1}{f}$.

Second, we show an important application of our protocol to collision resolution in single-hop, k-channel AHNs. More specifically, once we fix an arbitrary $f \geq 1$, the collision resolution problem in such a model can be solved in $\log\log n + 2.78\log k + 2.78\log f + o(\log\log n + \log k + \log f)$ time slots with probability at least $1 - \frac{1}{f}$.

Finally, we address the leader election problem on the single-channel AHN with diameter two. In this latter case we exhibit a leader election protocol terminating, with probability at least $1 - \frac{1}{f}$, in $2\log\log n + 11.10\log f + o(\log\log n + \log f)$ time slots, for every $f \geq 1$.

Keywords

Ad-hoc Networks, leader election, randomized algorithms.

1 Introduction

An Ad-hoc Network (AHN, for short) is a distributed system with no central arbiter, consisting of n radio transceivers, henceforth referred to as *stations*. The stations communicate using k communication channels denoted by $C(1), C(2), \ldots, C(k)$.

*Work supported, in part, by the NSF grant CCR-9522093 and by ONR grant N00014-97-1-0526.

At any one time, a station can tune to one such channel and/or transmit on at most one (possibly the same) channel. As customary, time is assumed slotted and all transmissions are edge-triggered, that is, take place at slot boundaries [3, 5]. We assume that the stations have a local clock that keeps synchronous time by interfacing with a Global Positioning System (GPS, for short) [6, 8, 15, 17]. It is worth noting that, under current technology, the commercially-available GPS systems provide location information accurate to within 22 meters as well as time information accurate [1] to within 100 nanoseconds [6]. In particular, this allows the stations to detect time slot boundaries and, thus, to synchronize.

We model an AHN by its underlying graph. Specifically, an AHN is thought of as a graph $G = (V, E)$, where V is the set of stations and two stations are connected by an edge in E whenever these stations can hear each other's transmission (i.e. can communicate in one hop). To avoid handling asymmetric situations, we assume that all the stations in the AHN have the same transmission range. Consequently, the graph A is undirected. In this paper we discuss AHNs with dense underlying graphs: in a *single-hop* AHN, every pair of stations are connected, and thus the diameter of the underlying graph is one; similarly, a *two-hop* AHN has a underlying graph of diameter two. We refer the reader to Figure 1 depicting a two-hop AHN.

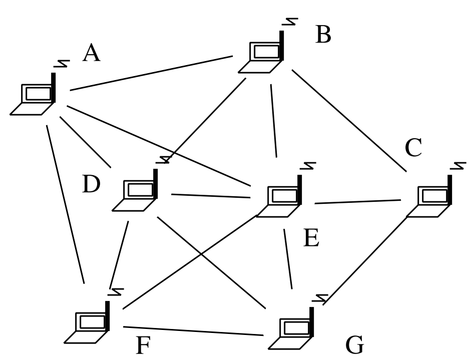

Figure 1: *Illustrating a two-hop AHN.*

Recall that radio transmission is omni-directional. As a consequence, a packet transmitted by a station will be received by all the stations within its transmission range. For example, in Figure 1, a packet transmitted by station A will be received by the stations B, D, E, and F as long as they are tuned to the same channel. Further, if a station receives packets transmitted by two or more stations in the same

[1]It is well documented that GPS systems using military codes achieve a level of accuracy that is orders of magnitude better than their commercial counterparts [6, 8]

time slot, then they collides and are garbled beyond recognition. Accordingly, for a station the *status* of a channel in a time slot is

- *NULL* if no station in its range transmits a packet on the channel,
- *SINGLE* if exactly one station in its range transmits a packet on the channel, and
- *COLLISION* if two or more stations in its range are transmitting on the channel.

Clearly, a packet is received successfully only if the status of the channel is SINGLE.

The problem that we address in this work is the well-known *leader election* problem which asks to designate one of the station as *leader*. In other words, after executing the leader election protocol, exactly one station learns that it was elected leader, while the remaining stations learn the identity of the leader elected. Historically, the leader election problem has been solved in wireline networks[1, 2, 9, 10, 18], where each station can specify a destination station.

It is customary to address the leader election problem on the AHN in three different scenarios:

Scenario 1: The number n of stations in the AHN is known in advance;

Scenario 2: The number n of stations in the AHN is unknown but an upper bound u on this number is known in advance;

Scenario 3: Neither the number of stations nor an upper bound on this number is known in advance.

It is intuitively clear that the task of leader election is the easiest in Scenario 1 and the hardest in Scenario 3, with Scenario 2 being in-between the two.

Several randomized protocols for single-hop AHNs have been presented in the literature. Metcalfe and Boggs [12] presented a leader election protocol for Scenario 1 that is guaranteed to terminate in $O(1)$ expected time slots. Their protocol is very simple: Every station repeats transmitting on the channel with probability $\frac{1}{n}$. When the status of channel becomes SINGLE, then a unique station that has broadcast is declared a leader. Later, Willard [19] showed that the leader election can be solved in $\log\log u + O(1)$ expected rounds if the conditions of Scenario 2 hold. Willard's protocol involves two stages: the first stage, using binary search, guesses in $\log\log u$ rounds a number i, $(0 \leq i \leq \log u)$, satisfying $2^i \leq n < 2^{i+1}$. The second stage finds the leader in $O(1)$ expected rounds using the protocol of [12]. Thus, the protocol elects a leader in $\log\log u + O(1)$ expected rounds. Willard [19] also improved this protocol to run under the conditions of Scenario 3 in $\log\log n + o(\log\log n)$ expected rounds. The first stage of the improved protocol uses the technique presented by Bentley and Yao [4], which finds an integer

i satisfying $2^i \leq n < 2^{i+1}$ for unknown n and u. However, it can be proved that, for every parameter $f \geq 1$, with probability at least $1 - \frac{1}{f}$, this latter protocol terminates in $\log\log n + \Omega(f)$ time slots.

Recently, Nakano and Olariu [14] presented two randomized leader election protocols running in $O(\log n)$ time slots with probability $1 - \frac{1}{n}$ and in $2\log\log n + o(\log\log n)$ time slots with probability $1 - \frac{1}{\log n}$ for Scenario 3. The disadvantages of these protocols is that the running time is not completely guaranteed. For example, the $O(\log\log n)$-time protocol may take a very large number of time slots to terminate. True, this only happens with probability $\frac{1}{\log n}$. However, if n is small, this probability is non-negligible.

The main contribution of this work is to present a very simple randomized protocol to solve the leader election problem under the conditions of Scenario 3 on a single-hop AHN in $\log\log n + 2.78\log f + o(\log\log n + \log f)$ time slots with probability at least $1 - \frac{1}{f}$ for every $f \geq 1$. We have also proved that any leader election protocol electing a leader with probability $1 - \frac{1}{f}$ $(f \geq 2)$ need to run in at least $\Omega(\log\log n + \log f)$ time slots. Due to the page limitation, the proof is omitted and can be found in the journal version of this paper.

Second, we discuss an important application of our leader election protocol to collision resolution in single-hop, k-channel AHNs. More specifically, we show that in such a model the collision resolution problem can be solved in $\log\log n + 2.78\log k + 2.78\log f + o(\log\log n + \log k + \log f)$ time slots with probability at least $1 - \frac{1}{f}$ for $f \geq 1$.

Finally, we show that a leader on a two-hop AHN can be elected in $2\log\log n + 11.10\log f + o(\log\log n + \log f)$ time slots with probability at least $1 - \frac{1}{f}$ for every $f \geq 1$.

2 A brief refresher of 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 [13].

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 parameter p. It is well known that X has a *binomial distribution* and that for every r, $(0 \leq r \leq n)$,

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

Further, the expected value of X is given by

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

To analyze the tail of the binomial distribution, we shall make use of the following estimates, commonly referred to as *Chernoff bounds* [13]:

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

$$\Pr[X < (1-\varepsilon)E[X]] \quad < \quad e^{-\frac{\varepsilon^2}{2}E[X]} \;\; (0 \le \varepsilon \le 1). \tag{2}$$

Let X be a random variable assuming only nonnegative values. The following inequality, known as the *Markov inequality*, will be also used:

$$\Pr[X \ge c \cdot E[X]] \quad \le \quad \frac{1}{c} \qquad \text{for all } c \ge 1. \tag{3}$$

3 Electing a leader in a single-hop AHN

The main goal of this section is to present the details of a randomized leader election protocol for an n-station, single-channel AHN. The workhorse of our leader election protocol is the protocols `Sieve0` and `Sieve1` whose details follow.

Protocol `Sieve0(`p`)`
    every active station transmits on the channel with probability $\frac{1}{2^{2^p}}$;
    if the status of the channel is COLLISION or SINGLE **then**
        the stations that have not transmitted become inactive
    endif

Protocol `Sieve1(`p`)`
    every active station transmits on the channel with probability $\frac{1}{2^{2^p}}$;
    if the status of the channel is SINGLE **then**
        the unique station that has transmitted becomes the leader and
        all stations exit the (main) protocol
    else if the status of the channel is COLLISION **then**
        the stations that have not transmitted become inactive
    endif

Note that `Sieve0(`p`)` does not elect a unique station as a leader even if the status of the channel is SINGLE.

In outline, our leader election protocol proceeds in three phases. In Phase 1 the calls `Sieve0(`0^2`)`, `Sieve0(`1^2`)`, `Sieve0(`2^2`)`, ..., `Sieve0(`t^2`)` are performed until, for the first time, the status of the channel is NULL in `Sieve0(`t^2`)`. At this point Phase 2 begins. In Phase 2 we perform the calls `Sieve0(`t^2-1`)`, `Sieve0(`t^2-2`)`, ..., `Sieve0(0)`. Clearly, if Phase 1 runs in $t+1$ time slots, then Phase 2 takes t^2 time slots. Finally, Phase 3 repeats the call `Sieve1(0)` until, eventually, the status of the channel is SINGLE, at which point a leader has been elected.

With this preamble out of the way, we are now in a position to spell out the details of our leader election protocol.

Protocol `Leader-election`
initially all the stations are active;
Phase 1:
  for $i \leftarrow 0$ **to** ∞
    `Sieve0`(i^2);
    exit for-loop if the status of the channel is NULL;
Phase 2:
  $t \leftarrow i^2 - 1$;
  for $i \leftarrow t$ **downto** 0
    `Sieve0`(i);
Phase 3:
  repeat
    `Sieve1`(0);
  forever

It is immediate that, upon termination, protocol `Leader-election` correctly returns a leader. It is clear that `Leader-election` elects a leader even if all the calls to `Sieve0` are replaced by `Sieve1`. We use `Sieve0` for Phases 1 and 2 to implement `Leader-election` run in a two-hop AHN efficiently.

We now turn to the task of evaluating the number of time slots it takes the protocol to terminate. In Phase 1, once the status of the channel is NULL the protocol exits the `for` loop. Thus, there must exist an integer t such that the status of the channel is:

- SINGLE or COLLISION in `Sieve0`(0^2), `Sieve0`(1^2), `Sieve0`(2^2), ..., `Sieve0`($(t-1)^2$), and
- NULL in `Sieve0`(t^2).

Determining a closed form for t appears to be rather difficult. For our purposes, however, computing a closed form for t is not necessary: all that we need is a suitable upper bound s on t.

Let $f \geq 1$ be arbitrary and write

$$s = \lceil \sqrt{\log\log(4fn)} \rceil. \tag{4}$$

Equation (4) guarantees that $s^2 \geq \log\log(4fn)$ and, thus, $2^{2^{(s^2)}} \geq 4nf$.

Assume that `Sieve0`(0^2), `Sieve0`(1^2), ..., `Sieve0`(s^2) are performed in Phase 1, and let X be the random variable denoting the number of stations that transmitted in `Sieve0`(s^2). Observe that in order for a given station to transmit in `Sieve0(`s^2`)` it must have transmitted in each `Sieve0(`j`)` with $0 \le j \le s-1$. In other words, the probability that a given station transmits in `Sieve0(`s^2`)` is

$$\frac{1}{2^{2^{(0^2)}}2^{2^{(1^2)}}\cdots 2^{2^{(s^2)}}} \quad \le \quad \frac{1}{2^{2^{(s^2)}}}.$$

Thus, the expected value $E[X]$ of X is

$$E[X] = \frac{n}{2^{2^{(s^2)}}} \le \frac{n}{4fn} = \frac{1}{4f}. \tag{5}$$

Using the Markov inequality (3) and (5), we can write

$$\Pr[X \ge 1] \le \Pr[X \ge 4fE[X]] \le \frac{1}{4f}. \tag{6}$$

Equation (6) guarantees that with probability at least $1-\frac{1}{4f}$, the status of the channel in `Sieve0(`s^2`)` is NULL. In particular, this means that

$$t \le s \text{ holds with probability at least } 1-\frac{1}{4f} \tag{7}$$

and, therefore, Phase 1 terminates in

$$t+1 \le s+1 = \lceil\sqrt{\log\log(4nf)}\rceil + 1 = O(\sqrt{\log\log n} + \sqrt{\log\log f})$$

time slots. In turn, this implies that Phase 2 terminates in at most

$$t^2 \le s^2 < (\sqrt{\log\log(4nf)}+1)^2 \le \log\log n + \log\log f + O(\sqrt{\log\log n} + \sqrt{\log\log f})$$

time slots. Thus, we have the following result.

Lemma 1 *With probability exceeding* $1-\frac{1}{4f}$, *Phase 1 and Phase 2 combined take at most* $\log\log n + \log\log f + O(\sqrt{\log\log n} + \sqrt{\log\log f})$ *time slots.*

Recall that Phase 2 involves t^2 calls, namely `Sieve0(`t^2-1`)`, `Sieve0(`t^2-2`)`, ..., `Sieve0(0)`. For convenience of the analysis, we regard the last call, `Sieve0(`t^2`)`, of Phase 1 as the first call of Phase 2. For every i, $(0 \le i \le t^2)$, let N_i denote the number of active stations just before the call `Sieve0(`i`)` is executed in Phase 2. We say that `Sieve0(`i`)` is *failure* if

> $N_i > 2^{2^i}\ln(4f(s^2+1))$ and the status of the channel is NULL in `Sieve0(`i`)`,

and, otherwise, *successful*. Let us evaluate the probability of the event F_i that `Sieve0(`i`)` is failure. From $(1-\frac{1}{n})^n \le \frac{1}{e}$, we have

$$\Pr[F_i] = \left(1-\frac{1}{2^{2^i}}\right)^{N_i} < e^{-\frac{N_i}{2^{2^i}}} < e^{-\ln(4f(s^2+1))} = \frac{1}{4f(s^2+1)}.$$

In other words, `Sieve(i)` is successful with probability exceeding $1 - \frac{1}{4f(s^2+1)}$. Let F be the event that all t^2 calls to `Sieve0` in Phase 2 are successful. Clearly,

$$F = \overline{F_0} \cap \overline{F_1} \cap \cdots \cap \overline{F_{t^2}} = \overline{F_0 \cup F_1 \cup \cdots \cup F_{t^2}}$$

and, therefore, we can write

$$\Pr[F] = \Pr[\overline{F_0 \cup F_1 \cup \cdots \cup F_{t^2}}] > 1 - \sum_{i=0}^{t^2} \frac{1}{4f(s^2+1)} \geq 1 - \frac{1}{4f}. \tag{8}$$

Thus, the probability that all the t^2 calls in Phase 2 are successful exceeds $\frac{1}{4f}$, provided that $t \leq s$. Recall, that by (7), $t \leq s$ holds with probability at least $1 - \frac{1}{4f}$. Thus, we conclude that with probability exceeding $1 - \frac{1}{2f}$ all the calls to `Sieve0` in Phase 2 are successful.

Assume that all the calls to `Sieve0` in Phase 2 are successful and let t', $(0 \leq t' \leq t^2)$, be the smallest integer for which the status of the channel is NULL in `Sieve0(`t'`)`. We note that since, by the definition of t, the status of the channel in NULL in `Sieve0(`t^2`)`, such an integer t' always exists. Our choice of t' guarantees that the status of the channel must be COLLISION in each of the calls `Sieve0(`j`)`, with $0 \leq j \leq t' - 1$.

Now, since we assumed that all the calls to `Sieve0` in Phase 2 are successful, it must be the case that

$$N_{t'} \leq 2^{2^{t'}} \ln(4f(s^2+1)). \tag{9}$$

Let Y be the random variable denoting the number of stations that are transmitting in `Sieve0(0)` of Phase 2. To get a handle on Y, observe that for a given station to transmit in `Sieve0(0)` it must have transmitted in each call `Sieve0(`j`)` with $0 \leq j \leq t' - 1$. Put differently, for a given station the probability that it is transmitting in `Sieve0(0)` is at most

$$\frac{1}{2^{2^{t'-1}} 2^{2^{t'-2}} \cdots 2^{2^0}} = \frac{1}{2^{2^{t'}-1}} = \frac{2}{2^{2^{t'}}}.$$

Therefore, we have

$$E[Y] \leq \frac{2N_{t'}}{2^{2^{t'}}} \leq \frac{2 \cdot 2^{2^{t'}} \ln(4f(s^2+1))}{2^{2^{t'}}} = 2\ln(4f(s^2+1)). \tag{10}$$

Select the value $\delta > 0$ such that

$$(1+\delta)E[Y] = 7\ln(4f(s^2+1)). \tag{11}$$

Notice that by (10) and (11) combined, we have

$$1 + \delta = \frac{7\ln(4f(s^2+1))}{E[Y]} \geq \frac{7\ln(4f(s^2+1))}{2\ln(4f(s^2+1))} = \frac{7}{2}.$$

In addition, by using the Chernoff bound (1) we bound the tail of Y, that is,

$$\Pr[Y > 7\ln(4f(s^2+1))] = \Pr[Y > (1+\delta)E[Y]]$$

as follows:

$$\Pr[Y > (1+\delta)E[Y]] < \left(\frac{e}{1+\delta}\right)^{(1+\delta)E[Y]} = \left(\frac{2e}{7}\right)^{7\ln(4f(s^2+1))} < e^{-\ln(4f(s^2+1))} < \frac{1}{4f}.$$

We just proved that, as long as all the calls to `Sieve0` are successful, with probability exceeding $1-\frac{1}{4f}$, at the end of Phase 2 no more than $7\ln(4f(s^2+1))$ stations remain active. Recalling that all the calls to `Sieve0` are successful with probability at least $1-\frac{1}{2f}$, we have the following result.

Lemma 2 *With probability exceeding* $1-\frac{3}{4f}$*, the number of remaining active stations at the end of Phase 2 does not exceed* $7\ln(4f(s^2+1))$.

Let N be the number of remaining active stations at the beginning of Phase 3 and assume that $N \leq 7\ln(4f(s^2+1))$. Recall that Phase 3 repeats `Sieve1(0)` until, eventually, the status of channel becomes SINGLE.

For a particular call `Sieve1(0)` in Phase 3, we let N', $(N' \geq 2)$, be the number of active stations just before the call. We say that `Sieve1(0)` is *successful* if

- either the status of the channels is SINGLE in `Sieve1(0)`, or
- at most $\lceil \frac{N'}{2} \rceil$ stations remain active after the call.

The reader should have no difficulty confirm that the following inequality holds for all $N' \geq 2$,

$$\frac{\binom{N'}{1} + \binom{N'}{2} + \cdots + \binom{N'}{\lceil \frac{N'}{2} \rceil}}{2^{N'}} \geq \frac{1}{2}.$$

It follows that a call is successful with probability at least $\frac{1}{2}$. Since N stations are active at the beginning of Phase 3, $\lceil \log N \rceil$ successful calls suffice to elect a leader.

Let Z be the random variable denoting the number of successes in a number α of independent Bernoulli trials, each succeeding with probability $\frac{1}{2}$. Clearly, $E[Z] = \frac{\alpha}{2}$. Our goal is to determine the values of ε and α in such a way that equation (2) yields:

$$\Pr[Z < \lceil \log N \rceil] = \Pr[Z < (1-\varepsilon)E[Z]] < e^{-\frac{\varepsilon^2}{2}E[Z]} = \frac{1}{4f}. \qquad (12)$$

It is easy to verify that (12) holds whenever

$$\begin{cases} (1-\varepsilon)E[Z] = \lceil \log N \rceil \\ \frac{\varepsilon^2}{2}E[Z] = \ln(4f) \end{cases} \tag{13}$$

hold true. Write

$$A = \frac{\lceil \log N \rceil}{2\ln(4f)}.$$

Solving for ε and $E[Z]$ in (13) we obtain:

$$0 < \varepsilon = \frac{2}{1+\sqrt{4A+1}} < 1$$

and

$$E[Z] = \ln(4f)[2A+1+\sqrt{4A+1}] < \ln(4f)(6A+2) = 3\lceil \log N \rceil + 2\ln(4f).$$

If we assume, as we did before, that $N \leq 7\ln(4f(s^2+1))$, it follows that

$$\lceil \log N \rceil \leq 3 + \log\ln(4f(s^2+1)) = O(\log\log\log\log n + \log\log f).$$

Thus, we can write

$$\alpha = 2E[Z] = 4\ln f + O(\log\log\log\log n + \log\log f).$$

Therefore, if $N \leq 7\ln(4f(s^2+1))$ then Phase 3 takes $4\ln f + O(\log\log\log\log n + \log\log f)$ time slots with probability at least $1-\frac{1}{4f}$. Noting that $N \leq 7\ln(4f(s^2+1))$ holds with probability at least $1-\frac{3}{4f}$, we have obtained the following result.

Lemma 3 *With probability at least $1-\frac{1}{f}$, Phase 3 terminates in at most $4\ln f + O(\log\log\log\log n + \log\log f)$ time slots.*

Now Lemmas 1 and 3 combined, imply that with probability exceeding $1-\frac{3}{4f}-\frac{1}{4f} = 1-\frac{1}{f}$ the protocol `Leader-election` terminates in

$$\begin{aligned} &\log\log n + \log\log f + O(\sqrt{\log\log n} + \sqrt{\log\log f}) \\ &+4\ln f + O(\log\log\log\log n + \log\log f) \\ &\quad = \quad \log\log n + 4\ln f + O(\sqrt{\log\log n} + \log\log f) \\ &\quad < \quad \log\log n + 2.78\log f + O(\sqrt{\log\log n} + \log\log f) \end{aligned}$$

time slots. We thus proved the main result of this section.

Theorem 1 *Let $f \geq 1$ be arbitrary. Protocol `Leader-election` terminates, with probability exceeding $1-\frac{1}{f}$, in $\log\log n + 2.78\log f + o(\log\log n + \log f)$ time slots.*

4 Application to collision resolution

The main goal of this section is to show how the leader election protocol developed in Section 3 can be used to solve efficiently the well-known *collision resolution* problem that arises in numerous contexts in distributed and parallel processing as well as in network protocol design.

As a rule, a collision is said to occur when two or more entities attempt to use simultaneously an unshareable resource. In multiprocessor systems, for example, a collision occurs when several processors attempt to write to common memory [7, 16]. In distributed computing a collision occurs when several processes attempt to enter a critical section at the same time [7, 11]. In AHNs, a collision occurs when two or more stations are transmitting on the same frequency at the same time. Given its fundamental and pervasive nature, it is not surprising that the collision resolution problem has received a lot of well-deserved attention in the literature [5, 3, 16].

We will be addressing the collision resolution problem in the context of a single-hop AHN endowed with k channels denoted by $\mathrm{C}(1), \mathrm{C}(2), \ldots, \mathrm{C}(k)$. Each station chooses independently a channel on which to transmit. The *collision resolution* problem asks to designate, for each of the k channels, a station that will be allowed to transmit on that channel.

It is clear that a leader election protocol for each channel can be used to solve the collision resolution problem. Specifically, for each channel $\mathrm{C}(i)$, $(1 \leq i \leq k)$, it suffices to elect a leader among the stations attempting to transmit on channel $\mathrm{C}(i)$. Of course, once the leader is elected, the collision resolution problem is solved and the elected station can proceed to transmit.

Suppose that protocol `Leader-election` is used to solve the collision resolution problem in each channel independently. By Theorem 1, each of the k instances of the leader election problem terminates, with probability exceeding $1 - \frac{1}{f'}$, in at most $\log\log n + 2.78 \log f' + o(\log\log n + \log f')$ time slots, for any choice of $f' \geq k$. Write $f' = kf$. Then, with probability exceeding $1 - \frac{1}{f}$ all the k instances of `Leader-election` terminate in

$$\begin{aligned} &\log\log n + 2.78 \log(fk) + o(\log\log n + \log(fk)) \\ &\quad = \quad \log\log n + 2.78 \log k + 2.78 \log f + o(\log\log n + \log k + \log f). \end{aligned}$$

As a consequence, we have the following result.

Theorem 2 *For every $f \geq 1$, the collision resolution problem on a k-channel, n-station AHN terminates, with probability exceeding $1 - \frac{1}{f}$ in $\log\log n + 2.78 \log k + 2.78 \log f + o(\log\log n + \log k + \log f)$ time slots.*

In real-life situations, the number k of channels satisfies $k \ll n$. In particular, for values of k that are polylogarithmic in n, we have the following corollary.

Corollary 1 *For every $f \geq 1$, the collision resolution problem on a k-channel, n-station AHN terminates, with probability exceeding $1 - \frac{1}{f}$ in $O(\log\log n) + 2.78\log f + o(\log f)$ time slots, provided that $k \leq (\log n)^{O(1)}$.*

5 Electing a leader in a two-hop AHN

The principle goal of this section is to extend the results of Section 3 to the case of AHNs of diameter two.

Our protocol for an AHN with diameter two essentially emulate the leader election protocol in Section 3. For this purpose, we are going to show `Sieve0(`p`)` and `Sieve1(`p`)` can be implemented to run in two and four time slots, respectively, on a AHN with diameter two. Thus, from Lemmas 1 and 3, a leader can be elected in

$$\begin{aligned} & 2(\log\log n + \log\log f + O(\sqrt{\log\log n} + \sqrt{\log\log f})) \\ + \quad & 4(4\ln f + O(\log\log\log\log n + \log\log f)) \\ = \quad & 2\log\log n + 16\ln f + o(\log\log n + \log f) \end{aligned}$$

time slots with probability at least $1 - \frac{1}{f}$ on a AHN with diameter two.

Recall that both in `Sieve0(`p`)` and in `Sieve1(`p`)` each active station transmits with probability 2^{2^p}. Let P denote a set of stations that has broadcast. Clearly, it is sufficient to determine whether $|P| = 0$ or $|P| \geq 1$ to implement `Sieve0(`p`)`. Similarly, to implement `Sieve1(`p`)` we need to find whether $|P| = 0$, $|P| = 1$, or $|P| \geq 2$. Thus, we will show how to do this.

Let $Status(s,t)$ denote the status of the channel that station s receives in time slot t.

Implementation of `Sieve0`

Step 1 Every station in P transmit on the channel.

Step 2 For every station s in S, if $Status(s,1) \neq NULL$ then s transmits on the channel. Each station s learns that $|P| = 0$ if $Status(s,2) = NULL$, and $|P| \geq 1$, otherwise.

We refer the reader to Figure 2 for an illustration. It is clear that this implementation correctly determines whether $|P| = 0$ or $|P| \geq 1$.

Subprotocol `Sieve1(`p`)` can be implemented as follows:

Implementation of `Sieve1`

Steps 1 and 2 Execute Steps 1 and 2 of the implementation of `Sieve0`. If $|P| = 0$ then, terminate.

Step 3 All stations s in P such that $Status(s,1) = COLLISION$ transmit on the channel.

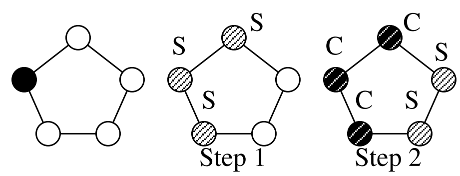

Figure 2: *Illustrating the implementation of* `Sieve0`

Step 4 All stations s in P such that $Status(s,3) \neq NULL$ transmit on the channel. Each station s leans that $|P| = 1$ if $Status(s,4) = NULL$, and $|P| \geq 2$ if $Status(s,4) \neq NULL$

The reader should refer to Figure 3 for the illustration. It is also clear that this implementation correctly determines whether $|P| = 0$, $|P| = 1$, or $|P| \geq 2$.

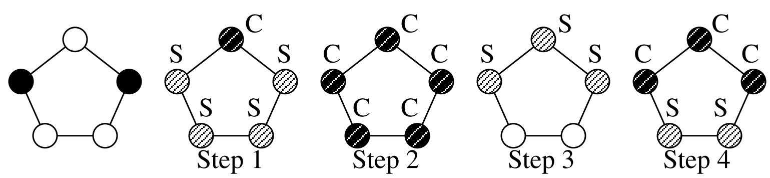

Figure 3: *Illustrating the implementation of* `Sieve1`

Thus, finally, we have,

Theorem 3 *Let $f \geq 1$ be arbitrary. With probability exceeding $1 - \frac{1}{f}$, a leader can be elected in a single-channel two-hop n-station AHN in $2\log\log n + 11.10\log f + o(\log\log n + \log f)$ time slots.*

References

[1] H. Abu-Amara, Fault-tolerant distributed algorithms for election in complete networks, *IEEE Transactions on Computers*, C-37, (1988), 449–453.

[2] Y. Afek and E. Gafni, Time and message bounds for election in synchronous and asynchronous complete networks, *SIAM Journal on Computing*, 20, (1991), 376–394.

[3] 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.

[4] J. Bentley and A. Yao, An almost optimal algorithm for unbounded search, *Information Processing Letters*, 5, (1976), 82–87.

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

[6] 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.

[7] H. El-Rewini and T. G. Lewis, *Distributed and Parallel Computing*, Manning, Greenwich, 1998.

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

[9] E. Korach, S. Moran,S. Zaks, Optimal lower bounds for some distributed algorithms for a complete network of processors, *Theoretical Computer Science*, 64, 1, (1989), 125–132.

[10] M. C. Loui, T. A. Matsushita, and D. B. West, Election in complete networks with a sense of direction, *Information Processing Letters*, 22, (1986), 185–187.

[11] N. Lynch, *Distributed Algorithms*, Morgan Kaufmann Publishers, 1996.

[12] R. M. Metcalfe and D. R. Boggs, Ethernet: distributed packet switching for local computer networks, *Communications of the ACM*, 19, (1976), 395–404.

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

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

[15] 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.

[16] B. Parhami, *Introduction to Parallel Processing*, Plenum Publishing, 1999.

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

[18] G. Singh, Leader election in complete networks, *Proc. ACM Symposium on Principles of Distributed Computing*, 1992, 179–190.

[19] D. E. Willard, Log-logarithmic selection resolution protocols in a multiple access channel, *SIAM Journal on Computing*, 15, (1986), 468–477.

Koji Nakano is with the Department of Electrical and Computer Engineering, Nagoya Institute of Technology, Showa-ku, Nagoya 466-8555, Japan. E-mail: nakano@elcom.nitech.ac.jp

Stephan Olariu is with the Department of Computer Science, Old Dominion University, Norfolk, VA 23529, U.S.A. E-mail: olariu@cs.odu.edu