Probabilistic Protocols for Node Discovery in Ad-hoc, Single Broadcast Channel Networks (Extended Abstract)

G. Alonso[1] **E. Kranakis**[2] **R. Wattenhofer**[1] **P. Widmayer**[1]

[1]Department of Computer Science
Swiss Federal Institute of Technology, ETH Zürich
Switzerland

[2]School of Computer Science
Carleton University
Canada

Abstract

The initial state in an ad-hoc network is a collection of nodes that are unaware of each other's presence. The very first step in joining and building the ad-hoc network is, thus, to discover other nodes. This node discovery procedure is a key step in configuring and optimizing the topology of the network. In spite of its relevance, node discovery has not yet been studied in detail and existing protocols are strongly tied to concrete network implementations. In this paper, we propose a model for node discovery that facilitates the analytical treatment of the problem. We concentrate on networks with a single shared broadcast channel. For these networks, we propose a number of protocols that shed light on the problem of node discovery.

1 Introduction

Ad-hoc networks are self-organized communication systems where the infrastructure (the participants, the routers, or the network configuration) is dynamically created and maintained. The building components are the very same nodes interested in exchanging messages. In general, self-organization requires complex protocols that are only now starting to be studied in detail. Of these protocols, the most basic one is *node discovery*. During node discovery, a node tries to find out which other nodes are within reach. The node broadcasts a message and waits to receive a response. When it receives such response from another node, it knows (a) that its presence is known to at least one other node and (b) the identity of that other node.

Behind its apparent simplicity, node discovery is a complex problem. For instance, the Bluetooth standard defines the problem away by using so called *asymmetric* protocols [5]. In an asymmetric protocol, nodes have pre-assigned roles that establish beforehand which node will be sending (discovering other nodes) and which nodes will be receiving (waiting to be discovered). Unfortunately, role pre-assignment is unrealistic as there are no centralized entities in an ad-hoc network that could assign the roles [12]. The alternative is, hence, to use *symmetric* protocols where a node can be both discovering nodes or waiting to be discovered. The problem of symmetric protocols is that the nodes must compete for access to the communication channel without being aware of the presence of other nodes and, therefore, without any chance to coordinate their transmissions to minimize collisions.

The problem of collisions when accessing a shared communication medium is well known. Collisions in a packet radio network are caused by either *direct* or *secondary* interference [3]. Direct interference occurs when two nodes transmit to each other at the same time. Secondary interference occurs when nodes unaware of each other's presence attempt to transmit at the same time.

Currently, there are two solutions to this problem in ad-hoc networks. One is frequency hopping as it is used in Bluetooth [5]. Frequency hopping minimizes the probability of collisions by having nodes transmitting in multiple channels. The other solution, used when a single broadcast channel is shared among all nodes, are randomized backoff protocols (see [13], among others), such as implemented in wireline networks. In this paper, we concentrate on systems with a shared broadcast channel such as those found in the Ricochet[1] or the Near-Term Digital Radio[2] networks [9].

Collision avoidance in a shared broadcast channel has been widely studied [3]. Existing collision avoidance protocols are based on the exchange of control messages among the nodes in order to dynamically establish a transmission schedule with the highest possible throughput. However, all these protocols assume nodes are already aware of the presence of other nodes. In other words, they assume that node

[1]http://www.metricom.com
[2]http://www.gordon.army.mil/tsmtr/ntdr.htm

discovery has already taken place.

Node discovery started to attract attention as the interest in ad-hoc networks increased. To our knowledge, only two node discovery protocols have been proposed in a model that is comparable with ours, both in the context of Bluetooth. Salonidis *et al.* [11, 12] propose and analyze a symmetric protocol for *2-node link formation* (i.e., node discovery with 2 nodes). The protocol is based on random schedules where each node switches between sending and listening at random and stays in each state for a random period of time. A random schedule is needed because the mean connection time (the time needed for the two nodes to discover each other) is infinite if the nodes switch state deterministically [11]. This *alternating states* protocol has been used as the basic building block for a scatternet formation protocol [12]. Law *et al.* [7] propose a probabilistic protocol for node discovery. In this protocol, a node decides with probability p to start discovering other nodes or, with probability $1-p$, to listen until it is discovered by other node. If a node does not manage to discover any other node in a period of time, it gives up. Similarly, if a node does not hear from any other node in a period of time, it also gives up. The protocols aim at establishing only one-to-one connections. The number of connections established in each round of the protocol is the smaller of the number of nodes in discovering mode and the number of nodes waiting to be discovered. (Note that Gasieniec *et al.* [4] recently studied the "wakeup problem", which is related to the node discovery problem; however, their results are not directly applicable in our context.)

These two protocols take advantage of the frequency hopping characteristics of Bluetooth and cannot be applied in single broadcast channel systems. In both cases, collisions are minimized by randomly assigning roles to each node. With a shared broadcast channel, however, once there are more than two nodes sending, collisions will occur. The goal here is to provide a formal characterization of node discovery protocols and suggest a simplified model for analysis and comparison in shared channel networks. Our optimization criteria is the time it takes for two nodes to discover each other in the presence of other nodes also engaged in node discovery. The protocols we propose have the same probabilistic flavor as in [7] but we consider additional techniques to minimize collisions. In addition, we also consider the *exponential backoff* approach where, to avoid collisions, a node backs off for a period of time before answering a message. In frequency hopping systems, exponential back off is also used but only as an internal step of the protocol (e.g., in Bluetooth, before a node sends an *inquiry response* message [5]). This comparison is important since backoff protocols are a standard way to deal with collisions in shared channel systems. In the paper, we show that probabilistic approaches might be an efficient alternative to backoff protocols. This result is relevant not only for shared channel systems but also for frequency hopping systems that could use a similar approach to the one we describe in all situations where exponential back off is used.

With this, the paper makes several contributions. First, we propose a model that can be used to analytically compute lower bounds for node discovery protocols (Section 2). Second, we describe several protocols that shed light on the problem of node discovery and perform better than existing solutions (Sections 3 and 4). We also indicate how these protocols can be used to explore other problems in ad-hoc networks (Section 5). With the exception of Theorem 2 no proofs are given in this paper. Additional proofs and details can be found in the full paper.

2 System Model and Problem Statement

2.1 Model

We assume a system with k nodes. The nodes do not have an ID, and the nodes do not know k. The nodes communicate by broadcasting messages. A node x_i, $1 \leq i \leq k$, can be in one of two states: *talking* (T) or *listening* (L). The state of each node is determined by the node discovery protocol run at that node. We assume that the nodes are synchronized, that is, they change states at the same point in time and remain in a given state for a period of time that is identical for all nodes. We also disregard the initial phase when not all nodes are operative and assume all nodes start working at the same time. With these assumptions, the model produces lower bounds for the cost of node discovery since phase differences and starting delays can only slow the protocols down.

This model is shown in Figure 1.. Each node is depicted

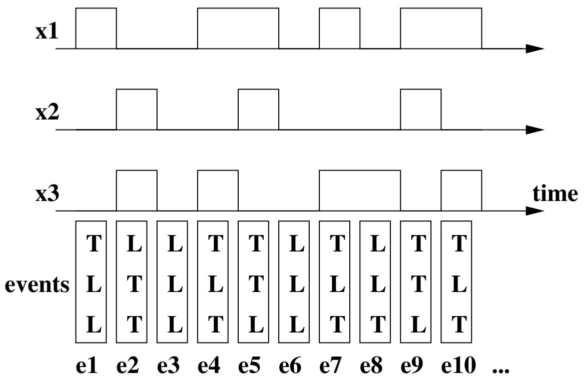

Figure 1. Schematic representation of the model ($k = 3$).

as a series of transitions between states (high being T and low being L). A *run*, R, of the protocol is a totally ordered sequence of *events*, $e_j,\ (1 \leq j \leq m)$, where each event contains the state of the k nodes in the system at that point in time (Figure 1). For simplicity, we will assume that e_i precedes e_j in a run if i, j. In addition, two events e_i, e_j are consecutive in a run with e_i preceding e_j if $j = i+1$. Thus, for $k = 2$, the possible events are $e_i \in \{\genfrac{}{}{0pt}{}{L}{L}, \genfrac{}{}{0pt}{}{L}{T}, \genfrac{}{}{0pt}{}{T}{L}, \genfrac{}{}{0pt}{}{T}{T}\}$. We will represent the state of a node x_i during event e_t as $x_i^{e_t}$.

2.2 Problem Statement

To formalize the problem of node discovery, we first need to define when messages can be received:

Definition 1 (Receiving a message) *A node x_i receives a message from a node x_j, $i \neq j$, during event e_t iff $x_i^{e_t} = L$, $x_j^{e_t} = T$, and $\forall x_k,\ k \neq j,\ x_k^{e_t} = L$.*

Note that this definition implies several things. First, if more than one node talks during an event, no message will be received during that event. Second, if a node receives a message during an event, all other nodes that are listening during that event will also receive the message. That is, if a node x_i receives a message from x_j in an event, all other nodes in the system will also receive the same message. This is a simplified model that, nevertheless, is a faithful representation of shared broadcast channels.

The node discovery protocol *terminates* when two nodes have found each other. This happens when a node successfully sends a message to another node and the second node manages to successfully send a reply message to the first node immediately afterwards. More formally:

Definition 2 (Protocol termination) *A run R terminates when the following two consecutive events occur:*

- $e_o \mid (\exists x_i^{e_o} = T) \wedge (\forall j \neq i,\ x_j^{e_o} = L)$*, and*
- $e_p \mid (\exists x_k^{e_p} = T,\ k \neq i) \wedge (\forall l \neq k,\ x_l^{e_p} = L)$

The termination of the protocol represents the point when two nodes have discovered each other. Figure 2 shows the two possible terminating sequences for $k = 2$: $\genfrac{}{}{0pt}{}{L\ T}{T\ L}$ or $\genfrac{}{}{0pt}{}{T\ L}{L\ T}$. This definition also has important implications for the model. The most relevant one is that the two events must be consecutive. This is because the second phase of node discovery usually involves a message that is explicitly addressed to one node (to the sender in the previous event). If this message is lost due to interference or because the intended recipient is not listening, then node discovery will not occur. The reason is that there are no acknowledgements in the protocol and a node has no way of knowing whether a message has been received or not. Although here we do not differentiate between types of messages, this behavior is captured by the constraint that the two events must be consecutive.

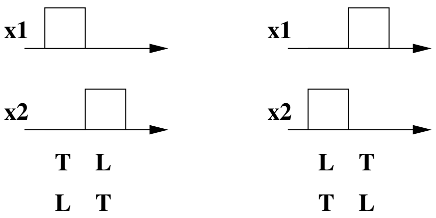

Figure 2. The two possible terminating event sequences ($k = 2$)

We define the *length* of a run to be the number of events from its start until its termination. Since the termination of the protocol requires two events, it follows that the minimum length of a run is 2. In the rest of this paper, we will focus on the design and analysis of node discovery protocols that minimize the expected length of a run.

3 Protocols for node discovery ($k = 2$)

We now propose several node discovery protocols for $k = 2$. These protocols will be later generalized to an arbitrary number of nodes. However, the discussion in the simpler case of $k = 2$ is very helpful to illustrate the problem of node discovery.

3.1 Random protocol ($\mathcal{RP}$)

In the *Random Protocol* ($\mathcal{RP}$) each node decides at random whether to talk or to listen. This protocol is similar in spirit to that proposed by Law *at al.* [7] except that in our protocol a node can change states from event to event. In the device discovery protocol of Law *at al.* nodes are permanently assigned to a state and remain in that state for the duration of the protocol. Obviously, this does not work in shared single broadcast channels. The link establishment protocol of Salonidis et al. [11] is also similar to $\mathcal{RP}$ in that the nodes change their state at random. However, unlike in Salonidis et al., in our protocol nodes remain in a given state for the entire duration of an event before considering switching state. Except for these differences, $\mathcal{RP}$ is equivalent to the protocols of Law *at al.* and Salonidis et al. but on a different network model.

In $\mathcal{RP}$, the probability of talking is p and the probability of listening is $q = 1 - p$. A simple analysis of the events required for termination gives the probability of node discovery in $\mathcal{RP}$ as $2p^2q^2$ (these are the two sequences shown in Figure 2). From here:

Theorem 1 (Expected length $E(R_{\mathcal{RP}})$) *The expected length of a run of the $\mathcal{RP}$ protocol, $R_{\mathcal{RP}}$ is given by the following expression:*

$$E(R_{\mathcal{RP}}) = \frac{2 - q - p^2}{2p^2q^2}$$

Proof (Theorem 1): See Theorem 7 with $k = 2$ □

3.2 Answering protocol ($\mathcal{AP}$)

$\mathcal{RP}$ tries to minimize collisions by forcing nodes to switch from talking to listening with certain probability. However, this ignores the pattern of events needed for the protocol to terminate. Since this pattern is known, it is possible to make the nodes behave in such a way so as to favor the occurrence of such patterns. A first example of this adaptation is the *Answering Protocol*, $\mathcal{AP}$. In $\mathcal{AP}$, a node will immediately answer if it receives a message from another node. In this way, at least one of the nodes is doing the correct thing after a message has been successfully sent from one node to the other. For comparison, in Bluetooth, when a node receives an *inquiry* message, it backs off for a random period of time, listens again on the same frequency, and when it receives a second *inquiry* message from the same node then it responds immediately [12]. In $\mathcal{AP}$ we do not introduce a delay and we do not wait for a second message. As soon as it receives a message, a node answers immediately. Such an approach will clearly not work for $k > 2$. However, understanding $\mathcal{AP}$ helps to illustrate the behavior of the backoff mechanism that will be used in the $\mathcal{SP}$ protocol.

In $\mathcal{AP}$, the following two event sequences are not possible: $\begin{smallmatrix} L & L \\ T & X \end{smallmatrix}$ or $\begin{smallmatrix} T & X \\ L & L \end{smallmatrix}$, where X represents either L or T. From here, the probability of node discovery in $\mathcal{AP}$ is $2pq^2$.

Theorem 2 (Expected length $E(R_{\mathcal{AP}})$) *The expected length of a run of the $\mathcal{AP}$ protocol, $R_{\mathcal{AP}}$ is given by the following expression:*

$$E(R_{\mathcal{AP}}) = \frac{2 - p^2 - q^2}{2pq^2}$$

Proof (Theorem 2): We will consider different sub-runs, $S_{R_{\mathcal{AP}}}$, of the protocol $R_{\mathcal{AP}}$. A sub-run has two phases. In phase 1, the protocol acts as a random protocol until either $\begin{smallmatrix} L \\ T \end{smallmatrix}$ or $\begin{smallmatrix} T \\ L \end{smallmatrix}$ is produced. The expected length of this phase is $\frac{1}{2pq}$ since the probability of the events of interest is $2pq$ and it is a sequence of independent events. Phase 2 consists of one event. The possible events are restricted in phase 2. If phase 1 finished with $\begin{smallmatrix} T \\ L \end{smallmatrix}$, phase 2 can be either $\begin{smallmatrix} L \\ T \end{smallmatrix}$ or $\begin{smallmatrix} T \\ T \end{smallmatrix}$. If phase 1 finished with $\begin{smallmatrix} L \\ T \end{smallmatrix}$, phase 2 can be either $\begin{smallmatrix} T \\ L \end{smallmatrix}$ or $\begin{smallmatrix} T \\ T \end{smallmatrix}$. If phase 2 is the event $\begin{smallmatrix} T \\ T \end{smallmatrix}$, the sub-run fails and a new sub-run starts. Otherwise the protocol terminates.

The expected length of a run of the protocol can be calculated in terms of how many sub-runs are needed. The expected length of a sub-run is $\frac{1}{2pq} + 1$. The probability that it fails is p. The probability that it succeeds is q. The probability of needing i sub-runs is, therefore, $p^{i-1}q$. From here:

$$\begin{aligned}
E(R_{\mathcal{AP}}) &= \sum_{i=1}^{\infty} i \cdot \left(\frac{1}{2pq} + 1\right) \cdot p^{i-1}q \\
&= \left(\frac{1}{2pq} + 1\right) \cdot q \cdot \sum_{i=1}^{\infty} i \cdot p^{i-1} \\
&= \left(\frac{1}{2pq} + 1\right) \cdot q \cdot \frac{1}{(1-p)^2} \\
&= \frac{1}{2pq^2} + \frac{1}{q} \\
&= \frac{2 - p^2 - q^2}{2pq^2}
\end{aligned}$$

□

3.3 Listen after talking protocol ($\mathcal{LP}$)

An alternative way to modify the $\mathcal{RP}$ protocol is based on the fact that after sending a message (i.e., talking), a node needs to listen in order to discover the other node. This idea leads to the *listening after talking* protocol, LP. For comparison, in frequency hopping systems like Bluetooth, nodes actively trying to discover other nodes alternative send messages and listen for answers according to a pre-established pattern of frequencies. This is the same mechanism exploited by the $\mathcal{LP}$ protocol. However, in single channel systems, there is no pattern of frequencies to follow, all messages are sent on the same channel. Thus, a node needs to listen more frequently because, when it talks, it might cause both direct interferences (with nodes that are trying to respond) and hidden terminal problems (for nodes that are either sending or responding to other nodes). The $\mathcal{LP}$ protocol takes this into account by forcing nodes to listen after they have sent. With this, in LP, the following two event sequences are not possible: $\begin{smallmatrix} L & X \\ T & T \end{smallmatrix}$ and $\begin{smallmatrix} T & T \\ L & X \end{smallmatrix}$, where X represents either L or T. From here, the probability of establishing a link in $\mathcal{LP}$ is $2p^2q$.

Theorem 3 (Expected length $E(R_{\mathcal{LP}})$) *The expected length of a run of the $\mathcal{LP}$ protocol, $R_{\mathcal{LP}}$ is given by the following expression:*

$$E(R_{\mathcal{LP}}) = \frac{2 - q^2}{2p^2q}$$

Proof (Theorem 3): See theorem 8 with $k = 2$. □

3.4 Conditional protocol ($\mathcal{CP}$= $\mathcal{AP}$ + $\mathcal{LP}$)

The protocols $\mathcal{LP}$ and $\mathcal{AP}$ are complementary in the sense that $\mathcal{AP}$ describes what to do after listening while $\mathcal{LP}$ describes what to do after talking. Both can be combined into a *conditional protocol* where a node always listens after having talked and answers if it receives a message. The interesting aspect of this protocol is that, for $k = 2$, it is enough to reach the events $\frac{L}{T}$ or $\frac{T}{L}$ to establish a link. After such event occurs, the system is deterministic and the next event will complete the node discovery procedure. Thus, the probability of node discovery is given by $2pq$.

Theorem 4 (Expected length $E(R_{\mathcal{CP}})$) *The expected length of a run of the $\mathcal{CP}$ protocol, $R_{\mathcal{CP}}$ is given by the following expression:*

$$\frac{2-q^2}{2pq}$$

3.5 Sleep protocol ($\mathcal{SP}$)

The protocols above introduce a probabilistic component that is common in frequency hopping systems but that, to our knowledge, is not used in single shared channel systems. In these systems, the most widespread approach to avoid collisions is to introduce delays to avoid simultaneous transmissions from several nodes. In conventional networks, collisions can be detected using CSMA-like systems [6]. When a collision is detected, the node waits for a random period of time before re-transmitting in the hope of avoiding a new collision. In our model, we are not concerned with standard collision detection (which takes place at the lower layers of the communication stack) but with the simultaneous response to a message from a node trying to discover other nodes. Of course, for $k = 2$ this does not make much sense since there cannot be simultaneous responses. However it is worthwhile to analyze this type of protocol in the simpler case to get a precise idea of its behavior.

To consider the effects of exponential backoff on our model, we introduce the *sleep protocol*, $\mathcal{SP}$. In $\mathcal{SP}$, after reaching either $\frac{L}{T}$ or $\frac{T}{L}$, the node that received the message suspends its activity for a period of time. This period can be modeled as a uniform random variable with values between 1 and n[3]. After this period, the node wakes up and sends a message back to the original sender.

[3]For an estimate of the value of n, we can look at the case of Bluetooth protocols. In Bluetooth, a node talking can scan all frequency ranges in $10\mu s$ or $20\mu s$ depending on where (which country) the system operates. Upon receiving an *inquiry* message, a node suspends its activity for of $639.375\mu s$. Assuming that the duration of an event in our model is the time it takes for a node to scan all frequencies, then the inactivity period varies between roughly 30 and 60 time intervals.

Before calculating the expected length of a run for $\mathcal{SP}$, it is necessary to show that the sleep time is optimal when a uniform random variable is used. This is an important step since the performance of $\mathcal{SP}$ depends on the probability distribution of the time the node remains silent. In the sequel we show that protocol $\mathcal{SP}$ is optimal for the uniform distribution.

Theorem 5 *Let $J_1, J_2, \ldots, J_k$ be i.i.d. random variables assuming values in the range $1..n$. The probability that the random variables $J_1, J_2, \ldots, J_k$ do not collide is maximized by the uniform distribution.*

With this, the expected length of a run of the $\mathcal{SP}$ protocol is as follows.

Theorem 6 (Expected length $E(R_{\mathcal{SP}})$) *The expected length of a run of the $\mathcal{SP}$ protocol, $R_{\mathcal{SP}}$, is given by the following expression:*

$$E(R_{\mathcal{SP}}) = \frac{1+(n+1)pq}{2pq^2}$$

3.6 Protocol comparison

The objective of these protocols is to minimize the delay involved in discovering other nodes. Figure 3 shows this

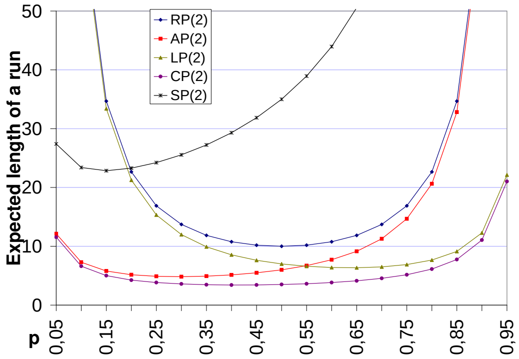

Figure 3. Expected length of a run for the different protocols as a function of p.

delay for the five protocols proposed as a function of the probability of talking p. The symmetric characteristics of the protocols $\mathcal{RP}$, $\mathcal{AP}$, $\mathcal{LP}$, and $\mathcal{CP}$ is clearly recognizable. $\mathcal{RP}$ has the narrowest range of values of p where it is feasible, with a minimum for $p = 0.5$. $\mathcal{AP}$ benefits from lower values of p (minimum at $p = 0.3$) since this diminishes the probability that the first sender talks at the send time the response is sent. In contrast, $\mathcal{LP}$ benefits from higher values of p (minimum at $p = 0.65$) since it compensates these higher values by forcing nodes to listen after they talk. Finally, $\mathcal{CP}$ benefits from these two properties and offers an acceptable response for the widest range of values of p (minimum at $p = 0.4$).

For $\mathcal{SP}$ we have used $n = 30$. $\mathcal{SP}$ follows the same pattern as $\mathcal{AP}$. This is not surprising as $\mathcal{SP}$ is, indeed, an $\mathcal{AP}$ protocol with a delay between the two terminating events. The delay caused by the inactive period before sending the response is what causes the curve to be shifted upwards and to the left. The faster deterioration in behavior as p increases that can be observed in $\mathcal{SP}$ is due to the hidden penalty in case the sub-run does not lead to node discovery: the inactive period is wasted time that increases the overall delay. This wasted time is proportional to p since the probability that the first sender is listening when the respond is sent decreases with higher p.

This comparison helps to understand the nature of the protocols but it is not entirely practical. The probability of sending, p, is a parameter of the protocol but does not need to be the same for all of them. Thus, to decide which protocol is the best, we need to compare the best values for each one of them. This is done in Figure 4, where it is clear

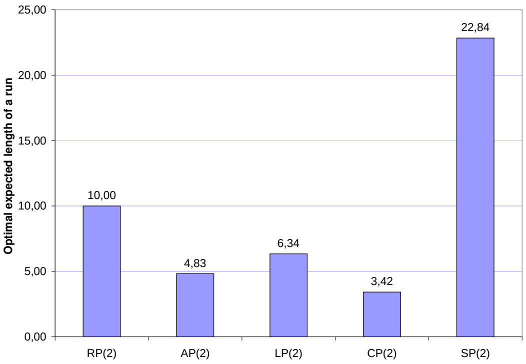

Figure 4. Optimal values for each protocol

that any of the proposed protocols performs better than $\mathcal{SP}$ unless $n = 1$ in which case $\mathcal{SP}$ reverts to $\mathcal{AP}$.

4 Competing for air time (node discovery with $k > 2$)

4.1 The random protocol in a k-node environment $\mathcal{RP}(k)$

With $k > 2$, an event is now a vector of k elements, s_i, $1 \leq i \leq k$, with values in the set $\{L, T\}$. For instance, for $k = 4$, a run of the protocol terminates with a sequence of the form: $\begin{smallmatrix} T & L \\ L & T \\ L & L \\ L & L \end{smallmatrix}$ (by Definition 2). For $k = 4$, there are 12 such sequences. Based on these termination sequences, the probability of node discovery is given by the expression $k(k-1)p^2q^{2k-2}$. The expected length of a run is, in this case, the expected number of events until any pair of nodes discover each other.

Theorem 7 (Expected length $E(R_{\mathcal{RP}(k)})$ *The expected length of a run of the $\mathcal{RP}(k)$ protocol, $R_{\mathcal{RP}(k)}$ is given by the following expression:*

$$E(R_{\mathcal{RP}(k)}) = \frac{1}{k} \cdot \left(\frac{1}{pq^{k-1}} + \frac{1}{(k-1)p^2q^{2k-2}} \right)$$

4.2 The $\mathcal{AP}$, and $\mathcal{CP}$ protocols in a k-node environment

We will refer to the $\mathcal{AP}$, $\mathcal{LP}$, and $\mathcal{CP}$ protocols in a k-node environment as $\mathcal{AP}(k)$, $\mathcal{LP}(k)$, and $\mathcal{CP}(k)$, respectively. However, of these three protocols, only $\mathcal{LP}(k)$ makes sense in practice. Both $\mathcal{AP}(k)$ and $\mathcal{CP}(k)$ force a node to answer immediately when it receives a message. When there are k nodes, the fact that a given node receives a message implies that there are $k-1$ nodes that have also received the same message. If all of them immediately reply, no message will get across due to the resulting interference. Thus, these nodes must either wait for a period of time (as in the $\mathcal{SP}$ protocol) or respond only with certain probability. If this probability is p, as it would be reasonable to assume, then $\mathcal{AP}(k)$ reverts to $\mathcal{RP}(k)$ and $\mathcal{CP}(k)$ becomes $\mathcal{LP}(k)$.

4.3 Listen after talking protocol in a k-node environment $\mathcal{LP}(k)$

In protocol $\mathcal{LP}(k)$ nodes are forced to listen after they have sent. The behavior of this protocol can be represented with a finite automaton (Figure 5). The automaton has four states. S is the initial state representing the state of the system when the protocol is initiated, and all events where there is no node in state L or there are j nodes in state T with $1 < j < k$. T_1 represents all events where exactly one node is in state T. T_k represents the event when all nodes are in state T. E is the final state reached once node discovery has been completed. Figure 5 also reflects the probability of the transitions among the states. The analysis of the protocol can be done based on this automaton.

For simplicity we use the following abbreviations:

$$\begin{aligned} A &= kpq^{k-1} \\ B &= \sum_{i=2}^{k-1} (k-i)pq^{k-i-1} \\ C &= (k-1)pq^{k-2} \\ D &= p^k \end{aligned}$$

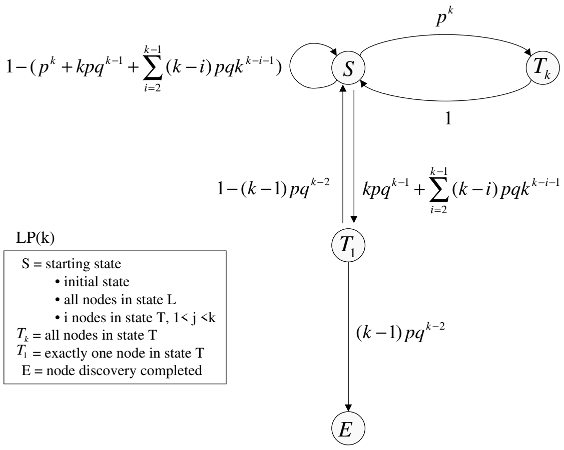

Figure 5. The $\mathcal{LP}(k)$ protocol as finite automaton

Theorem 8 (Expected length $E(R_{\mathcal{LP}(k)})$ *The expected length of a run of the $\mathcal{LP}(k)$ protocol, $R_{\mathcal{LP}(k)}$ is given by the following expression:*

$$E(R_{\mathcal{LP}(k)}) = \frac{1+A+B+D}{C(A+B)}$$

4.4 Protocol comparison

The relevance and practical impact of $\mathcal{LP}(k)$ and $\mathcal{RP}(k)$ can be best understood when compared with $\mathcal{SP}(k)$. In $\mathcal{SP}(k)$, nodes that initially receive a message go into an inactive period of length between 1 and n. Following Theorem 5, we will assume the length of this inactive period is determined based on a uniform random variable. For comparison purposes, the behavior of $\mathcal{LP}(k)$ and $\mathcal{RP}(k)$ is given by Theorems 7 and 8. The behavior of $\mathcal{SP}(k)$ is obtained through simulation. To validate the simulation, we have compared the results of the simulation for k=2 with the expression given in Theorem 6 (Figure 6.c).

In the small and medium ranges, the three protocols exhibit the same type of behavior. The curves have a $\bigcup$ shape with the arms tilting to the right or left depending on the probability p. In all cases, lower values of p lead to worse behavior for small values of k; higher values of p lead to worse behavior for large values of k (and unacceptable values for very large systems). Averaging over the small and medium range (Figure 6.d), $\mathcal{SP}(k)$ has the best behavior (performance does not degrade as sharply when not on the optimal point) although $\mathcal{LP}(k)$ does better in the central range. In fact, if the range for k is known in advance, it is always possible to come up with a better protocol than

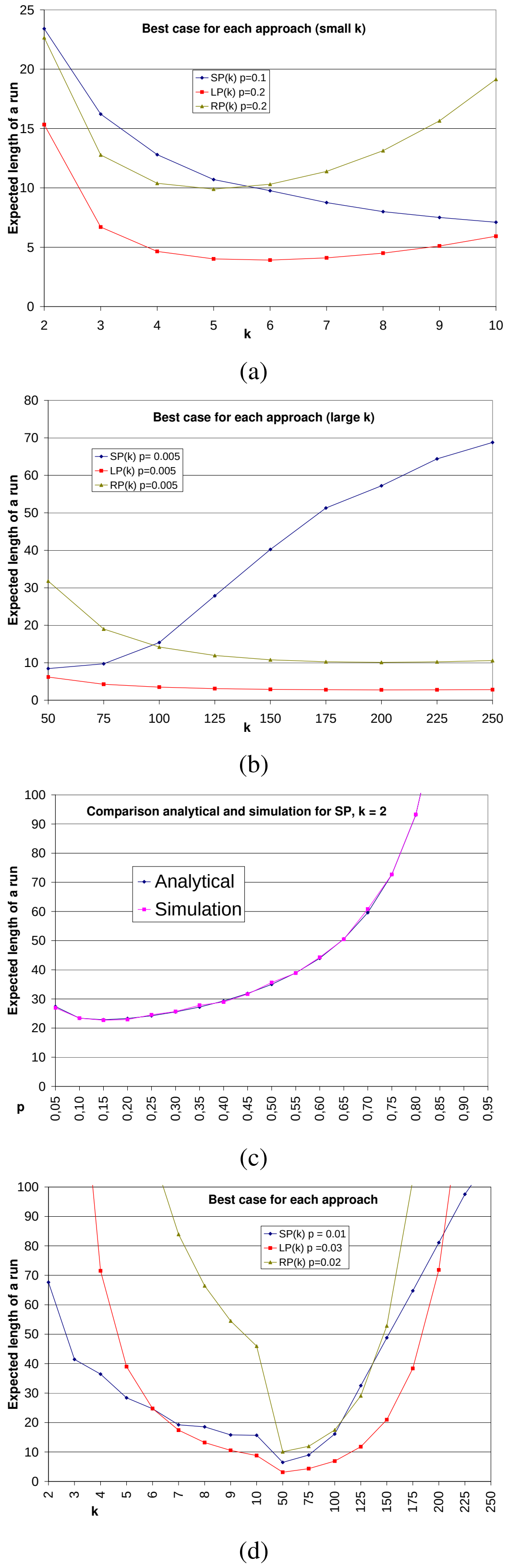

Figure 6. Comparison of the best cases for each protocol

$\mathcal{SP}(k)$. Figure 6.a shows the best cases for each protocol (p selected so as to minimize the expected length of a run) for small systems. Figure 6.b does the same but for large systems. In both cases, $\mathcal{LP}(k)$ is clearly a better option than $\mathcal{SP}(k)$.

5 Conclusions

We have analyzed the problem of node discovery in ad-hoc, single broadcast channel networks. We have proposed a formal model that captures the most relevant aspects of the problem and is suitable for analysis. Using this model, we described several protocols for node discovery. The analysis has shown that if the number of nodes involved in node discovery is unknown, $\mathcal{SP}(k)$ has the best performance on average. When the size of the system is known, $\mathcal{LP}(k)$ is clearly a better option.

In terms of future work, we are currently exploring the problem of node discovery under different network models and with more complex protocols. We are also in the process of studying adaptive protocols that change the probability of transmission or the protocol used as a function of the perceived system size. Such protocols will then be the basic building bock for scatternet formation protocols. With this, we expect to be able to define protocols that behave optimally in a wider range of system's sizes an that will shorten considerably the time to set up an ad-hoc network.

Acknowledgements

Part of this work was done while Evangelos Kranakis was visiting ETH, Zürich. Research supported in part by NSERC (Natural Sciences and Engineering Research Council of Canada) and MITACS (Mathematics of Information Technology and Complex Systems) grants. Gustavo Alonso and Roger Wattenhofer are supported in part by the NCCR MICS of the Swiss National Science Foundation.

References

[1] G. Blom and D. Thoburn, How Many Random Digits Are Required Until Given Sequences Are Obtained, J. Applied Probability, 19, 518-531, 1982.

[2] W. Feller, An Introduction to Probability Theory and its Applications, Vol. II, Wiley, 1966.

[3] R. Garcés, J.J. Garcia-Luna-Aceves, Collision Avoidance and Resolution Multiple Access for Multichannel Wireless Networks, IEEE Infocom 2000, March 26 - 30, 2000, Tel-Aviv, Israel.

[4] L. Gasieniec, A. Pelc, D. Peleg, The Wakeup Problem in Synchronous Broadcast Systems, SIAM Journal on Discrete Mathematics, May, 2001.

[5] J. Haartsen, Bluetooth Baseband Specification v. 1.0, www.Bluetooth.com

[6] L. Kleinrock and F.A. Tobagi, Packet Switching in Radio Channels: Part I - carrier sense multiple-access modes and their throughput-delay characteristics, IEEE Trans. Commun., volCOM-23, no. 12, pp. 1400-1419, 1975.

[7] C. Law, A.K. Mehta, K.-Y. Siu, Performance of a Bluetooth Scatternet Formation Protocol, The Second ACM Annual Workshop on Mobile Ad Hoc Networking and Computing (mobiHoc 2001), October 4-5, 2001, Long Beach, California, USA

[8] S. R. Li, A Martingale Approach to the Study of Occurrence of Sequence Patterns in Repeated Experiments, Annals of Probability, 8, 1171- 1176, 1980.

[9] R. Ramanathan, R. Rosales-Hain, Topology Control of Multihop Wireless Networks using Transmit Power Adjustment, IEEE Infocom 2000, March 26 - 30, 2000, Tel-Aviv, Israel.

[10] S. Ross, Stochastic Processes, John Wiley and Sons, 2nd edition, 1996.

[11] T. Salonidis, P. Bhagwar, L. Tassiulas, Proximity Awareness and Fast Connection Establishment in Bluetooth, The First ACM Annual Workshop on Mobile Ad Hoc Networking and Computing (mobiHoc 2000), August 11, 2000, Boston, Massachusetts, USA (also TR, The Institute for Systems Research, University of Maryland, 2000; http://www.isr.umd.edu/TechReports)

[12] T. Salonidis, P. Bhagwar, L. Tassiulas, R. LaMaire, Distributed Topology Construction of Bluetooth Personal Area Networks, IEEE 20^{th} Annual Joint Conference of the IEEE Computer and Communications Societies, (also TR 2001-1, The Institute for Systems Research, University of Maryland, 2001; http://www.isr.umd.edu/TechReports).

[13] Andrew S. Tanenbaum, Computer Networks, Prentice-Hall International, Inc., 1996.