Adaptive Topology Discovery in Hybrid Wireless Networks[*]

RANVEER CHANDRA
Cornell University, USA

CHRISTOF FETZER
AT&T Labs - Research, USA

KARIN HÖGSTEDT
AT&T Labs - Research, USA

Abstract

Wireless networks in home, office and sensor applications consist of nodes with low mobility. Most of these networks have at least a few powerful machines additionally connected by a wireline network. Topology information of the wireless network at these powerful nodes can be used to control transmission power, avoid congestion, compute routing tables, discover resources, and to gather data. In this paper we propose an algorithm for topology discovery in wireless networks with slow moving nodes and present various performance characteristics of this algorithm. The proposed algorithm discovers all links and nodes in a stable wireless network and has an excellent message complexity: the algorithm has an optimal message complexity in a stable network and the overhead degrades slowly with increasing mobility of the nodes.

Keywords

Topology Discovery, Wireless Networks, Distributed Systems, Home Networks, SOHO Networks, Sensor Networks.

1 Introduction

An ad hoc network is a wireless network using peer to peer communication for network connectivity. Many applications of such networks, such as home, office and sensor networks, have one or more nodes that are relatively static. We call

[*]This work has been supported in part by DARPA/AFRL-IFGA grant F30602-99-1-0532 and in part by NSF-CISE grant 9703470, with additional support from the AFRL-IFGA Information Assurance Institute, from Microsoft Research and from the Intel Corporation. Work done while all authors worked at AT&T Labs – Research.

such networks *hybrid wireless networks*, and describe it in detail in Section 2 . The static nodes require the topology information of the networks for varied purposes. For example, the coordinator of a sensor network will need this information to learn of areas that are not monitored by his sensors. The manager of a relief operation could use the topology information to efficiently deploy his crew in a disaster site. A topology discovery algorithm will also help him to periodically view the location of his personnel and send reinforcements. Further, topology information will be crucial for network management of an ad hoc network. It will enable the manager to monitor topology control decisions within the network, and help in maintaining connectivity and conserving the scarce resources, such as power and bandwidth, of the network.

The required accuracy of the topology discovery algorithm varies with the application. For example, a coordinator of a sensor network could do with an approximate information, while the manager of a relief operation needs to know the precise topology of the network. In this paper we propose an efficient topology discovery algorithm for hybrid wireless networks, which has the degree of robustness as a parameter of operation. The overhead of the algorithm varies with the required accuracy of the application. Further, our algorithm gives optimal performance in a network with very low mobility. The complexity degrades slowly with an increase in mobility in the network.

In what follows, we first describe our system and failure model in Section 2. In Section 3, we then formally define the topology discovery problem that we solve in this paper. Section 4 goes on to describing the proposed topology discovery protocol, and in Section 5 we discuss some properties of the protocol. Section 6 describes our performance measurements, and in Section 7 we describe the related work. We conclude in Section 8.

2 System Model

We define a *hybrid wireless network* to consist of a set of computers connected by wireline and wireless network links. We classify computer nodes as either a wireline node (only connected to the wireline network), a mobile node (only connected via wireless links), and a gateway node (has wireline and wireless links). We denote the wireless nodes (consisting of mobile and gateway nodes) by W_i. A wireless link is either in base station mode or in ad hoc mode. Note that also the wireless link(s) of a gateway node can be in ad hoc mode. A precise definition of our system model is given in [8].

Communication System: We abstract away the details of the MAC and network layer and assume the following communication model. Each wireless node can send local unicast and broadcast messages. By *local* we mean that the message is not routed via intermediate nodes. Current MAC layers do not support robust local broadcast messages. We show in Section 4.2.2 how to implement such a mechanism. We assume that links are likely to be bidirectional, i.e., if W_i can receive messages from W_j then W_j can receive messages from W_i. This is a valid assumption because underlying wireless protocols, e.g., the 802.11 MAC layer, require links to be bidirectional too.

Failure Model: The network can drop messages (omission failure) or delay the delivery of messages (performance failure). A node has crash failure semantics, i.e., a node can only fail by prematurely stopping the execution of its program.

Stability Properties: To be able to talk more precisely about the properties of our topology discovery protocol, we classify network links as either *stable*, *disconnected*, or *unstable*. We say that a link is *stable* in a given time interval if all messages sent via the link are delivered and acknowledged in a timely manner. We call a link *disconnected* if the link drops all messages. A link that is neither stable nor disconnected is called *unstable*.

We call a network *stable* if each link is either stable or disconnected and all nodes can communicate with each other via a path of stable links. We call a network *semi-stable* if all nodes can communicate with each other via a path of stable links. Note that a semi-stable network that contains unstable links is not stable.

3 Problem Description

We define the topology discovery problem as follows. The discovery is initiated by some node C (*coordinator*). The protocol has to return the discovered topology to C within a bounded time. We assume that the topology is returned in form of a predicate T. Predicate $T(i, j)$ holds iff the protocol discovered a link $L_{i,j}$, i.e., that node W_j can receive messages sent by node W_i.

To specify that the topology returned by a topology discovery protocol reflects the topology of the wireless network is non trivial. The topology might change due to mobility during the run-time of the protocol and hence, we cannot require that the topology T is identical to the topology of the wireless network. Instead we restrict T by two constraints: **R1** and **R2**. Constraint **R1** states that if the protocol says that a link $L_{i,j}$ exists then, at least at some point during the execution of the protocol, a message must have been successfully sent via $L_{i,j}$. Constraint **R2** states that if T says that there is no link $L_{i,j}$ between two nodes W_i and W_j, then either the link is not stable or neither W_i nor W_j are reachable by C via stable links. A precise definition of the problem specification is given in [8].

4 Protocol Description

In this section we illustrate the difficulties associated with the design of a topology discovery algorithm for hybrid networks. We first briefly explain an algorithm which was proposed in [13] for switched LANs called Autonet, and then highlight the reasons why such algorithms for wired networks do not perform well in hybrid networks. We then explain how we have solved these problems.

4.1 Topology Discovery in Autonet

The basic topology discovery algorithm in Autonet comprises two phases. The coordinator, which could be any node in Autonet, initiates the first phase by broadcasting[1] a topology request message. A node marks the sender of the first received broadcast as a parent, notifies the parent of this relationship, and broadcasts the request message further. At the end of the first phase, i.e., when each node has broadcast exactly once, a network-wide spanning tree with the coordinator as root has been imposed across the physical network.

The second phase is initiated by the leaves of the spanning tree, and rises toward the coordinator. A node sends a response message to its parent after it has received one from all its children. The response message carries the topology information of all the nodes in its subtree. This is created from the topology information in the responses from all its children and its own neighborhood information. As a result, when the coordinator receives the response messages from its children, it can construct the topology of the entire network.

In the rest of this paper we refer to the first phase of the algorithm as the *Diffusion phase* and the second phase as the *Gathering phase*. Due to space constraints we refer the reader to [13] for a more detailed description of the topology algorithm used in Autonet.

4.2 Topology Discovery in Hybrid Wireless Networks

In wired networks, such as the switched LANs that the Autonet algorithm was designed for, the links are reliable and the immediate neighbors of each node are static. The tree structure that was built during the Diffusion phase of the algorithm can therefore be assumed to still be intact during the Gathering phase when the topology information of all the nodes is propagated back toward the coordinator. This is however not the case in hybrid networks where links are prone to failure both due to link interference and mobility of the nodes. Interference and mobility are the two main issues that complicates the design of a topology discovery algorithm for hybrid networks compared to the ones for wired networks.

In this section we describe our proposed algorithm for topology discovery in hybrid networks. Our algorithm builds upon the algorithms in [16, 13], and we will describe our algorithm by showing how we address the issues that arise due to link interference and node mobility. These issues can be summarized by the following four questions: 1) How does a node discover its neighbors? (4.2.1)

[1]A broadcast in Autonet is a set of unicasts to all its neighbors. It is just one message in Hybrid Wireless Networks.

2) How is the problem of broadcast collisions handled? (4.2.2) 3) How does the algorithm adapt to mobility? (4.2.3) 4) What if the mobility is very high? (4.2.4).

4.2.1 Neighbor Discovery

Autonet has point-to-point links, and assumes the knowledge of all immediate neighbors at each of the nodes. However, the broadcast medium in hybrid wireless networks causes the links to change dynamically with the mobility of nodes in the network. A topology discovery algorithm for such networks has to discover the most recent network topology to be effective, and this will require up to date neighborhood information at all the nodes in the network.

One way to build the neighborhood information is by periodic heartbeat messages as used in some protocols such as [15]. However, this scheme consumes more power and send messages since all the nodes broadcast at least once during the Diffusion phase of our protocol. Assuming reliable broadcasts, every node therefore receives a message from all its neighbors during the Diffusion phase and uses it to build an accurate picture of its immediate neighborhood. The assumption of reliable broadcasts is however not realistic in existing network protocols such as 802.11, and we describe in Section 4.2.2 how we implement a robust broadcast mechanism on top of existing MAC protocols.

It should be noted that a node will only add links to its neighbor list across which it has received at least one message during the execution of the protocol. This means that only links that satisfy requirement **R1** are added. Note that the request message will be received by each node W_i that is connected to the coordinator via a path of stable links. In particular, W_i will rebroadcast the request message and add all nodes connected to W_i via a stable link to its neighbor list. We show in the rest of this section how the neighbor lists of W_i will be forwarded to the coordinator – which is necessary and sufficient to satisfy requirement **R2**.

4.2.2 Robust Broadcast

The success of the topology discovery algorithm depends on the reliability of broadcasts. Broadcasts in most MACs for ad hoc networks are not reliable. In 802.11 for example, broadcasts are sent whenever the carrier is sensed free and can therefore result in a number of collisions, as is experienced in [12]. As described in Section 4.2.1, broadcast collisions can cause fewer links to be discovered. Furthermore, unreliable broadcasts might not necessarily result in the construction of a spanning tree during the Diffusion phase, even when the network is connected. As an example, we have simulated the Diffusion phase of the Autonet algorithm described in Section 4.1 using GloMoSim [19] with three different transmission powers, -10dBm being the weakest. The percentage of discovered links is shown in Figure 1. We see that as we increase the transmission power and therefore the neighbor density, more broadcast messages collide and a smaller fraction of the links are discovered.

We increase the robustness of broadcasts by using a variation of the RTS/CTS scheme proposed by [18]. The idea is to ensure that the broadcast reaches at least one node in the neighborhood. In our scenario, a node definitely knows of at least one neighbor, that is the node from which it received the broadcast. To use the scheme proposed in [18], we could let each node do a RTS/CTS with the node from which it received a request whenever it wishes to rebroadcast it. An RTS/CTS mechanism is however mostly helpful for large messages. The broadcast messages in the Diffusion phase are small in size, and the lengthy message exchange sequence of an RTS, a CTS, the actual request message, and finally an acknowledgement is unnecessary. Instead, a node simply broadcasts the message and expects an acknowledgement from the node from which it had received the request that it is now retransmitting. If a node does not receive the acknowledgement within a certain (randomized) amount of time, it rebroadcasts the message. This is repeated a small number of times, or until it receives an acknowledgement, which ever occurs first. Our simulations show that this scheme adds a lot of robustness to the Diffusion phase. For example, in the scenarios simulated in Figure 1, this scheme discovers 100% of the links for all three transmission powers. Furthermore, this scheme of robust broadcasts removes the need for an explicit unicast of a node before marking its parent as used in Autonet [13].

4.2.3 Adapting to Mobility

Autonet uses a spanning tree to discover the topology of the network. This scheme yields an accurate result when the links do not break during the execution of the protocol. However, mobility can result in link breakages along the spanning tree, which can seriously affect the performance of the protocol (Section 6). We therefore propose to use a *k-resilient mesh* spanning the network to withstand mobility of the nodes. A *k-resilient mesh* is a connected directed acyclic graph (DAG), rooted at the coordinator, in which any node W_i has at most k parents, where $1 \leq k \leq K$, and K is small, constant integer. A parent of W_i is any node W_j such that there exists a link in the DAG from W_i to W_j. We call k the *resiliency factor* of the mesh. Note that a k-resilient mesh is also a $(k+x)$-resilient mesh, $\forall 1 \leq x \leq K-k$. Figure 2 gives an example of a 3-resilient mesh.

The algorithm described in this section uses a mesh structure to handle mobility. The mesh provides alternate routes to the coordinator. A greater value of k implies more paths from the nodes, and therefore increases the resilience to mobility in the network. The value of k is specified by the coordinator, which usually has a good estimate of the amount of mobility in the network.

The algorithm used to build a spanning tree, described in Section 4.1, is a specific instance ($k = 1$) of the algorithm we describe here that we use to construct a k-resilient mesh.[2] In our algorithm, each node broadcasts at most k different

[2]A spanning tree is a 1-resilient mesh.

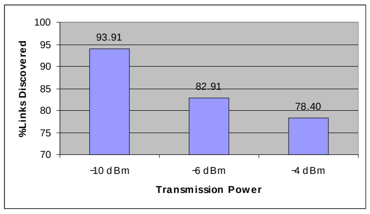

Figure 1: GloMoSim simulation results for three different transmission powers, using 50 nodes in a 200x200m area.

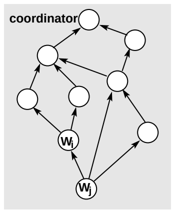

Figure 2: In a 3-resilient mesh each node has at most 3 parents.

messages. Further, each request message also maintains a `hopCount` field that is incremented by a node before rebroadcasting it. This field is required to keep the mesh loop free. The parameter *k* is specified by the coordinator in the first request message that initiates the Diffusion phase. On receiving a request message, a node adds the sender as its parent if 1) it has less than *k* parents, 2) the sender is not a parent already, and 3) the `hopCount` of the message is smaller than the `hop-Count` of the first request message that it had received during this protocol run. If the above three conditions are met, the node 1) increments the `hopCount` field in the message, 2) sets the parent field to be the sender of the broadcast message, and 3) rebroadcasts the new message as described in Section 4.2.2.

The above modifications to the Autonet algorithm require changes in the Gathering phase as well. To propagate the response messages back to the coordinator, the nodes use the mesh instead of the spanning tree as described in Section 4.1. This means that a node unicasts the response messages to all its parents in the mesh, and the number of parents per node is guaranteed to be less than or equal to *k*. Another problem with link breakages and mobility in the Autonet algorithm is that nodes wait for a response message from all their children before they send their responses to their parents. Since a child might move away and never return, nodes might wait forever and not let the algorithm run to completion. To avoid this problem, we let the nodes timeout after a carefully chosen wait period, described in [8], and unicast the response to their parents even if they haven't yet received a response from all their children.

4.2.4 Handling Extreme Mobility

High mobility in the network can cause a number of links to be unstable and for some nodes none of their parents might therefore be reachable. Such a node is unable to send its message to any of its parents during the Gathering phase and this could result in a very inaccurate topology information. The algorithm we propose has a provision to handle this situation by allowing nodes to enter in *panic* mode whenever they cannot communicate with any of their parents. The decision to allow the nodes to enter panic mode is taken by the coordinator when it initiates the Diffusion phase. Several parameters, such as the required accuracy of the algorithm, the allowed delay, and the amount of power left at the nodes are used to take this decision.

When a node enters panic mode, it sends its response message to all its neighbors determined as described in Section 4.2.1. If a node receives a response message from a parent in panic mode, it removes the parent from its list of parents. For example, suppose node W_i is the parent of node W_j. If W_i enters panic mode, then node W_j should not rely on successful communication of its response to node W_i, since W_i might still not be able to send the message any further up the mesh toward the coordinator. W_j thus removes W_i from its list of parents and enters panic mode itself if this list becomes empty.

A worse situation could arise when a node is unable to send its response message to any of its neighbors. Although a node in this case does not have any of its original neighbors discovered during the Diffusion phase, it might still get its message across if it broadcasts it. We use a robust broadcast slightly different from the one described in Section 4.2.2; since the messages are big, explicit RTS and CTS messages are used. The RTS/CTS is done with the last neighbor from whom any message was heard or received. To reduce the size of this broadcast we do not send the complete neighborhood information, but rather only the node information. The argument is that if all the neighbors of a node have failed, then its link information is not of any use to the coordinator anyway.

5 Protocol Properties

In this section we discuss some qualitative properties of our protocol. The quantitative performance is presented in Section 6.

Correctness: The protocol collects link information locally in a neighbor list and then collates and forwards the neighbor lists to the coordinator during the Gathering phase. Only senders of messages that were sent after the protocol was started are included in the neighbor list. Hence, our protocol satisfies property **R1**.

Property **R2** is only guaranteed if the panic mode is switched on. However, if the network is stable, the protocol satisfies **R2** even if the panic mode is switched off. Note that in a semi-stable network, the topology returned by the protocol includes all nodes and it contains all stable links but it might also contain unstable links. However, it will never contain disconnected links.

Message Complexity: The message complexity of the Diffusion phase is $O(N)$. We say that a mesh constructed by the Diffusion phase is semi-stable, iff each node is reachable from the coordinator via a path of stable links along the mesh. The message complexity of the Gathering phase is also $O(N)$ as long as the mesh is *semi-stable*.

When the mesh is not semi-stable, some nodes might enter panic mode. If a node in panic mode receives a neighbor list it has not received previously, it first unicasts a response message to all its neighbors. Assuming that a node has on average D neighbors, the average number of unicast messages in panic mode is at most DN per node. If all the unicasts messages the node sent failed, it broadcasts a condensed message.The number of robust broadcast messages in panic mode is at most N per node. The total number of unicast and broadcast messages in panic mode is thus at most DN^2 and N^2, respectively. The worst case message complexity for panic mode is thus $O(DN^2 + N^2) = O(DN^2)$.

6 Performance

Setup: The topology discovery algorithm was simulated in GloMoSim [19], which uses a parallel, event-driven simulation language called Parsec [5]. 50 nodes were randomly placed in a 200m$\times$200m area. IEEE 802.11 was used as the MAC protocol and the bandwidth was assumed to be 2 Mbps. The Random Waypoint model was used to model the mobility of the nodes in the network. In this model each node moves towards a random destination at a speed chosen randomly between a predefined minimum and maximum value. It then pauses for some duration and continues with this mobility pattern. In our simulations the minimum speed was set to 0 m/s and the pause time to 30 seconds. We expect the speed of nodes in our target application to be walking speed, i.e., approximately 1 m/s.

All nodes were set to have the same transmission power, although simulations were carried out for three different transmission powers: -10 dBm, -6 dBm and -4 dBm. At a transmission power of -10 dBm the average neighborhood size was about 4, at -6 dBm it was close to 11, and about 17 at -4 dBm. When the transmission power was -4 and -6 dBm, the network was semi-stable for all simulated speeds. At -10 dBm however, the network was only semi-stable up to 0.8 m/s. One node was designated as the coordinator and the percentage of links and nodes discovered at this node is presented in this section.

Evaluation: We evaluate our protocol using three different metrics. First, we measure the percentage of stable links discovered (See Section 2 for a definition of a stable link). It is unreasonable to expect that an algorithm would discover links that only existed for a short amount of time during the execution of the algorithm. Furthermore, information about links that do no longer exist is irrelevant, and could also lead to inaccurate information at the coordinator. We therefore compare the number of links that our protocol discovers, to the actual number of stable links, and use this fraction to evaluate our protocol.

Second, we measure the percentage of nodes discovered. In some applications the link information might not be of interest; it is simply enough to learn of the different nodes in the network. We therefore also present the percentage of nodes discovered as a way of evaluating our protocol.

Third, we measure the message overhead of the protocol. The protocol was executed with and without the panic mode and for different values of the resiliency factor of the mesh that is formed during the Diffusion phase. A higher resiliency factor increases the robustness of the protocol but at the expense of an increase in the message overhead. The panic mode further improves the robustness of our protocol but also results in an added increase in the number of messages sent during the Gathering phase. During the Diffusion phase, every node sends a constant number of messages, resulting in $O(KN) = O(N)$ messages, where N is the number of nodes, and K is the constant upper bound of the resiliency factor of the mesh constructed during the Diffusion phase. We therefore only present the message overhead incurred during the Gathering phase.

Results: In the simulations for transmission powers of -6 dBm and -4 dBm our protocol discovers 100% of the links and nodes for the speeds relevant to our target applications, i.e. 0 to 1.2 m/s. Panic mode kicks in after these speeds, and helps the protocol discover all the nodes and links in the network at the cost of extra messages. In the more common cases, when the 802.11 protocol is used, the transmission powers are expected to be much higher, typically around 15 dBm for the Lucent Wavelan [1]. Furthermore, the Cisco Aironet 350 Series supports six transmission power levels from 0 dBm to 20 dBm [2]. In these commonly occurring scenarios when the network is densely populated, our topology discovery protocol performs extremely well. However, some networks have a sparse density because of fewer nodes in the network or due to devices with low transmission power, e.g., sensors. The -10 dBm case illustrates one such extreme situation. In the rest of this paper we discuss this particular scenario in detail. Due to lack of space, we have eliminated a more detailed discussion of the results of the -6 and -4 dBm cases. In these cases the algorithm discovers the complete topology even at very high speeds with the help of panic mode. Interested readers are referred to [8] for these results.

Transmission Power: -10 dBm: The percentage of stable links discovered using

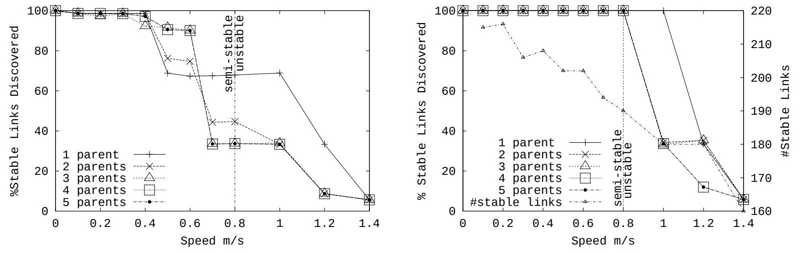

Figure 3: Percentage of stable links discovered for -10 dBm transmission power, and different values of the resiliency factor *k*. `#stable links` denotes the total number of stable links in the network. The network is semi-stable up to `0.8 m/s`.

a transmission power of -10 dBm is shown in Figure 3. Without the panic mode the algorithm discovers *nearly all* the stable links up to 0.4 m/s. An increase in speed reduces the robustness of the mesh and results in a lower percentage of stable links discovered. The reason for the relatively large decrease in the number of stable links discovered is that when the coordinator learns of a link from a node W_i to another node W_j, it cannot infer that there is a link from W_j to W_i since we do not assume all links to be bidirectional. Had we done that, the number of stable links discovered would have increased, both with and without the panic mode.

It is interesting to note that increasing the resiliency factor, i.e., the maximum number of parents allowed, is useful only up to a certain speed: in this case up to 0.6 m/s. When increasing the average number of parents, the average number of children per node grows, and thus also the probability that at least one link between a parent and a child breaks. Especially for higher speeds where link breakages are even more common, an increase in the number of parents thus results in a greater number of nodes that timeout before sending their response messages. Due to these timeouts, the propagation of neighborhood information to the coordinator is delayed proportionally to the number of parents. But in a network with sparsely connected nodes moving at a high speed, it is important to forward the information as fast as possible, since a link might disappear before it is used. In a sparsely connected network with high speed, the coordinator might therefore discover a higher percentage of stable links if the nodes are permitted fewer parents, i.e., if the resiliency factor is smaller.

We call this tradeoff between minimizing response time and maximizing mesh resiliency the *inversion property*. The inversion property suggests that the resiliency factor should be a tunable parameter, chosen according to the mobility and density of the network. In some applications it might not be necessary to discover 100% of the links, whereas it might be crucial to minimize the number of messages sent. To handle different kinds of applications, system configurations, and dynamically changing systems, it is therefore important to be able to tune the algorithm to best fit the current needs.

With panic mode turned off, the protocol discovers *all* the stable links as long as the network is stable. With panic mode turned on, the protocol discovers *all* the stable links as long as the network is semi-stable. At higher speeds the network is no longer connected using stable links. The coordinator is therefore unable to receive all the response messages and this results in a decrease in the percentage of stable links that are discovered. We can also see the effect of the inversion property in this case. When the nodes are allowed fewer parents, the response messages in the Gathering phase are sent sooner and the coordinator is therefore able to gather information over more links before they break, resulting in a higher percentage of stable links discovered.

The message overhead of the Gathering phase is shown in Figure 4. When the panic mode is turned off, the response messages are only sent along the mesh. No effort is made to repair the mesh and so the number of response messages stays

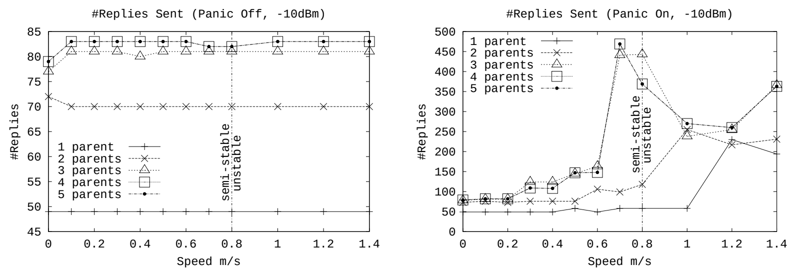

Figure 4: Number of response messages sent for -10 dBm transmission power, and different values of the resiliency factor *k*.

nearly constant across different speeds. The slight variations in each of the curves are due to the differing structure of the mesh at different speeds. As the resiliency factor increases, the average number of parents increases and more links are part of the mesh which results in a higher message overhead.

When the panic mode is turned on, the number of replies sent during the Gathering phase increases more with speed if a larger resiliency factor is used. The reason for this is the inversion property; when a node has only a few parents its timeout is shorter and it therefore has a higher probability that the links to its parents still exist. This has the effect that a network using a larger resiliency factor enters the panic mode for lower speeds than networks with a smaller value of *k*: the increase in the number of messages for $k = 3$, 4, and 5, occurs already at 0.7 m/s, whereas the increase for $k = 1$ and $k = 2$ does not occur until 1.2 and 1 m/s, respectively. The number of replies drop for higher speeds since the network is no longer semi-stable. More links are thus broken and fewer nodes enter the panic mode by receiving response messages sent by other nodes in panic mode.

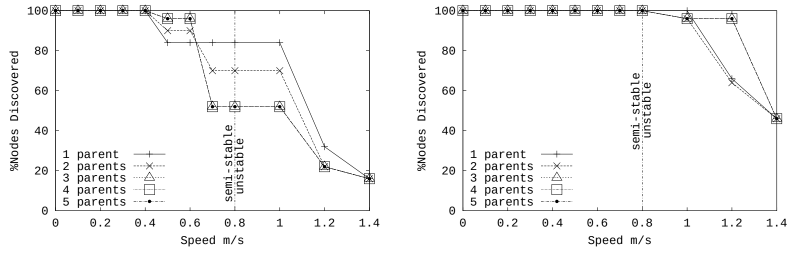

Figure 5: Percentage of nodes discovered for -10 dBm transmission power, and different values of the resiliency factor *k*.

Figure 5 shows the percentage of nodes discovered by the protocol. Without the panic mode, the percentage of nodes discovered decreases with an increase in speed. This is because the nodes are no longer connected using stable links along the mesh. We can also see that the inversion property causes better performance for a smaller value of the resiliency factor at speeds greater than 0.6 m/s. Again, the shorter timeout results in more information reaching the coordinator.

When the panic mode is turned on, the percentage of nodes discovered is significantly higher than the percentage of stable links discovered. This is due to the fact that only one link to or from the node has to be discovered for the coordinator to learn about the existence of the node.

Note that in Figure 3, 4, and 5 the protocol has a similar performance for $k = 4$ and $k = 5$. This is because the nodes only have around four neighbors, and they can therefore not have more than four parents even though five parents are permitted by the protocol.

7 Related Work

The problem of topology discovery in ad hoc networks is significantly different than in wired networks. There is no IP subnet hierarchy and nodes might have stale neighborhood information. Additionally, there is no popular network management protocol, such as SNMP, for ad hoc networks. Therefore, previously proposed topology discovery protocols for the internet, such as [3, 4, 7, 17] perform poorly in hybrid wireless networks.

There has however been some work related to topology discovery for ad hoc networks. [9] uses a clustering scheme to discover the network topology. The cluster heads are dynamically chosen based on geographic location or network connectivity, and the MIBs at cluster heads are used to gather topology information. However, this scheme has the overhead of constantly maintaining cluster heads in the network. Additionally, the information in the MIBs might be stale due to mobility and could fail to provide a complete link information of the network. Another topology discovery algorithm is presented in [14]. Mobile agents in the nodes periodically gather topology information and disseminate it to all the other nodes in the network. However, this scheme does not provide an instantaneous topology of the network. This algorithm is also extremely intensive in time and messages to discover a complete topology of the network. Further, link state routing protocols, such as TBRPF[6] and OLSR[11], require each node to constantly maintain a partial topology of the network. This is an overhead when the link information is required temporarily at a few nodes. Link state routing protocols also provide only network layer topology information and might not be able to discover the complete physical connectivity of the network. It should however be mentioned that none of the above protocols were designed to solve the particular problem of topology discovery we address in this paper.

In our setting, the entire topology information has to be learned at a few prespecified nodes. Previous approaches described above do not provide an efficient infrastructure to solve this problem. [10] attempts to solve our problem of topology discovery in sensor networks. A hierarchical tree-based clustering scheme gathers neighborhood information from all sensor nodes. However, this protocol provides only partial link information. It also assumes a reliable broadcast mechanism, which has not yet been developed for ad hoc networks, and the tree structure used is unstable and susceptible to link breakages in mobile networks.

8 Conclusion

In this paper we have defined a precise system model and problem statement for the topology discovery problem. We have also presented a reliable protocol for topology discovery in wireless networks. The protocol consists of two phases; the first phase diffuses the initial topology request message across the whole network and the second propagates the neighborhood information back to the initiating node. The second phase propagates the information using a mesh structure built during the first phase. This mesh can be used for any kind of data gathering application, and is not limited to topology discovery. The message complexity of the protocol is $O(N)$ in a stable network with N nodes, and it slowly degrades to a worst case of $O(DN^2)$ when the nodes are more mobile. We show that the protocol discovers close to 100% of the stable links and 100% of the nodes in the targeted applications.

References

[1] Lucent ORiNOCO, http://www.digit-life.com/articles/wirelesslanslucent.

[2] Aironet 350 Series, http://www.naswireless.com/products_cisco350.html.

[3] HP Openview, http://www.openview.hp.com/.

[4] IBM Tivoli, http://www.tivoli.com/.

[5] R. Bagrodia, R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, B. Park, and H. Song. Parsec: A parallel simulation environment for complex systems. *IEEE Computer*, 31(10):77 – 85, October 1998.

[6] B. Bellur and R. G. Ogier. A reliable, efficient topology broadcast protocol for dynamic networks. In *Proceedings IEEE INFOCOM 1999*, March 1999.

[7] Y. Breithart, M. Garofalakis, C. Martin, R. Rastogi, S. Seshadri, and A. Silberschatz. Topology discovery in heterogeneous IP networks. In *Proceedings IEEE Infocom*, volume 1, pages 265–274, 2000.

[8] R. Chandra, C. Fetzer, and K. Högstedt. A mesh-based robust topology discovery algorithm for hybrid wireless networks. http://www.research.att.com/∼christof/papers/topology.pdf.

[9] W. Chen, N. Jain, and S. Singh. ANMP: Ad hoc network management protocol. *IEEE Journal on Selected Areas in Communications*, 17(8):1506 – 1531, August 1999.

[10] B. Deb, S. Bhatnagar, and B. Nath. A topology discovery algorithm for sensor networks with applications to network management. Technical Report DCS-TR-441, Dept. of Computer Science, Rutgers University, May 2001.

[11] P. Jacquet, P. Muhlethaler, A. Qayyum, A. Laouiti, L. Viennot, and T. Clausen. Optimized link state routing protocol, March 2001. http://www.ietf.org/internet-drafts/draft-ietf-manet-olsr-04.txt.

[12] S. Y. Ni, Y. C. Tseng, Y. S. Chen, and J. P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings ACM/IEEE MobiCom*, pages 151–162, August 1999.

[13] T. L. Rodeheffer and M. D. Schroeder. Automatic reconfiguration in Autonet. In *Proceedings 13th ACM SOSP*, pages 183–197, October 1991.

[14] R. RoyChoudhury, S. Bandyopadhyay, and K. Paul. A distributed mechanism for topology discovery in ad hoc wireless networks using mobile agents. In *IEEE First annual workshop on Mobile and Ad Hoc Networking and Computing (MobiHoc)*, pages 145–146, 2000.

[15] E. M. Royer and C. E. Perkins. Multicast operation of the ad hoc on-demand distance vector routing protocol. In *Proceedings of ACM/IEEE MobiCom*, pages 207–218, August 1999.

[16] A. Segall. Distributed network protocols. *IEEE Trans. on Infor. Theory*, 29(1), Jan 1983.

[17] R. Siamwalla, R. Sharma, and S. Keshav. Discovering Internet topology. Cornell University, 1999.

[18] J. Tourrilhes. Robust broadcast: improving the reliability of broadcast transmissions on CSMA/CA. In *Proceedings of the 9th IEEE International Symposium on Personal Indoor and Mobile Radio Communications*, volume 3, pages 1111–1115, September 1998.

[19] X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: A library for parallel simulation of large scale wireless networks. In *Proceedings of the 12th Workshop on Parallel and Distributed Simulation*, pages 154–161, May 1998.