Reducing Packet Transmissions in Ad Hoc Routing Protocols by Adaptive Neighbor Discovery

Sven Hanemann, Ralph Jansen and Bernd Freisleben

Department of Mathematics and Computer Science, University of Marburg
Hans-Meerwein-Straße, D-35032 Marburg, Germany
{hanemann,jansen,freisleb}@informatik.uni-marburg.de

Keywords: ad hoc network, routing protocols, bandwidth control, beaconing

Abstract

An ad hoc network is a collection of mobile wireless nodes forming a temporary network without any established infrastructure. Dynamic network topology changes due to node mobility, limited transmission range, limited channel bandwidth and limited power supply require the use of dedicated routing protocols. To operate well, such routing protocols must rely on a method for efficiently discovering neighbor connections in a highly dynamic environment. However, neighbor detection may consume as much bandwidth as routing itself. In this paper, we present an algorithm which dynamically adapts itself to the movements of nodes to achieve maximal neighbor detection accuracy with minimum bandwidth consumption. Particular emphasis is put on the detection quality among nodes with strongly different moving speeds.

1. Introduction

An ad hoc network is a collection of mobile nodes forming a temporary network without the aid of any established infrastructure or centralized administration. Ad hoc networks differ significantly from existing fixed networks. First, the topology of interconnection may be quite dynamic. Second, the conditions of a wireless environment with lossy channels, interference, limited bandwidth and limited power supply are quite different from the conditions in fixed networks. Many routing protocols exhibit their least desirable behavior when presented with these conditions.

Over the last years, many routing protocols dealing with the specific challenges of mobile ad hoc networks have been developed. These protocols can be broadly classified into two basic categories, reactive and proactive.

Reactive protocols build and maintain routes only when needed to reduce routing overheads. Examples of reactive protocols are the Dynamic Source Algorithm (DSR) [9] and Ad Hoc On-Demand Distance Vector routing (AODV) [11]. Several modifications of DSR and AODV try to reduce route discovery latency and routing overheads by extending these protocols to compute multiple paths at source and intermediate nodes in a single route discovery attempt [10, 12].

Proactive protocols [13] operate like existing routing algorithms in the Internet: they build up tables or trees with routing information according to the current network topology. A clear disadvantage is the continuous use of bandwidth for update messages.

Routing protocols of both categories require that each mobile node cooperates in a distributed way to forward packets or messages received from other nodes. The knowledge of reachable neighbors and their available services is the basic requirement for all further network operations, and particularly important for routing calculations. New neighbors are only detected if they broadcast an announcement. In the same fashion, the loss of a neighbor node will only be detected if communication attempts with this neighbor node fail. Since nodes may move while they are communicating, a constant need for new information about changes in their environment exists. Considering that network partitions and link breakdowns are normal events in an ad hoc network, routing announcements are necessary to achieve an acceptable accuracy of detecting new neighbors and to maintain the list of existing services. The minimally needed rate of announcements strongly depends on the acceptable error rate of working communication links to neighbors, the possible change rate of the network topology and the mobility of the cooperating nodes.

Tuning the interval duration for networks containing nodes with different speed must be based on a worst case...
calculation leading to a non-optimal fit. Simulations [1] have shown that announcement intervals should be carefully selected because they are critical for the performance of high level protocols. In this paper, we investigate the interdependency between the announcement interval for neighbor discovery, the average error rate and bandwidth consumption. It will be shown that neighbor detection can consume as much bandwidth as routing for a given error rate of usable communication links to neighbors. An improved adaptive announcement algorithm will be presented that dynamically adapts itself to the movement patterns of nodes with highly different speeds in order to optimize the announcement rate for a given error rate of working communication links. Simulations will be shown to demonstrate the performance of the algorithm with respect to its neighbor detection quality and bandwidth consumption in different scenarios. Furthermore, the performance of a routing algorithm which uses the information produced by the algorithm for adaptive neighbor discovery to determine routes to all possible nodes is evaluated. Again, the benefits of the proposed algorithm regarding the number of working routes and bandwidth will become evident. The paper is organized as follows. Section 2 presents the basic idea of the proposed algorithm for adaptive connection discovery. Section 3 describes the implementation of the adaptive algorithm as part of our wireless network simulator. In section 4, simulation results of the algorithm in different scenarios are presented, and in section 5 the interaction of routing and connection discovery is evaluated via simulation. Section 6 concludes the paper and outlines areas for future research.

2. Adaptive Neighbor Discovery
The motivation for an adaptive neighbor discovery algorithm is to get maximal service availability with minimum bandwidth consumption for a given error rate in different wireless network environments. Nodes may move with totally random speed and movement patterns. Most other approaches to achieve the same goal either concentrate on features of specific algorithms [10,12] or introduce additional parameters like measurements of signal strength [6]. However, nearly all proactive protocols use neighbor discovery or fixed beaconing for route maintenance, even some reactive protocols like AODV use beaconing. The implementation of an adaptive neighbor discovery algorithm can improve their performance for a given connection error rate. The most natural approach for neighbor discovery for ad hoc networks uses periodically transmitted announcements from every node to determine connectivity [3,4]. The announcement interval can be easily tuned to fit error rate or bandwidth requirements as long as nodes move with equal speed. To achieve a desired error rate, the highest node speed has to be taken into account. Hence, ad hoc networks with very different node speeds and constant announcement intervals will show either unsatisfying detection results or will exhibit high bandwidth consumption. An approach using constant time intervals is therefore not satisfying. A first approach for an adaptive announcement interval solution is the use of an “immediate reply” to newly detected nodes. When a node detects the announcement of a new neighbor, it sends an answering announcement regardless of its own announcement interval. This shortens the detection time for new neighbors.

The detection accuracy can be determined by the following calculation: A node with an announcement interval time \( I \) sends an announcement every \( I \) time units. If the node moves into the transmission range of a new neighbor, it will be discovered between 0 and \( I \) time units. Since the chance of discovery is equally distributed over time, the mathematical description of the expected mean value for Interval between 0 and \( I \) can be used to describe the average time until discovery:

\[
\int_{0}^{I} \frac{I}{x} \, dx = \frac{1}{2} I \quad (1)
\]

In case of an immediate reply announcement, the sending node will be discovered first. The receiving node will send the immediate reply and will be discovered. Assuming that both nodes transmit independently of each other, the average time until discovery can be calculated as the minimum between both discovery intervals:

\[
\frac{\int_{0}^{I} \min(x,y) \, dx}{I} = \frac{\int_{0}^{I} x \, dx + \int_{0}^{I} y \, dx}{I} = \int_{0}^{I} \frac{x}{3} \, dx \quad (2)
\]

It can be seen that the expected value for discovery time has been improved. Unfortunately, the loss of a neighbor will only be discovered with \( \frac{1}{2} I \). Therefore, the average time for the discovery of a topology change is improved to a value between \( \frac{1}{2} I \) and \( \frac{1}{3} I \).

For constant node speed, the detection time is proportional to the failure of working communication links because topology changes are only discovered after they have actually happened. The error of working communication links is reduced due to the improved neighbor detection time.

An immediate reply is a useful option to a constant announcement scheme, because it offers a first step towards an adaptive algorithm. An immediate reply also increases the efficiency of all routing algorithms which select announcement rates dynamically. Depending on the
node environment, these routing algorithms will select different announcement intervals. A fully adaptive announcement solution adjusts the announcements intervals of the whole network. Every node selects its own announcement interval by evaluating the available information about its movement. To do this, information about past announcements has to be available.

With information about the number of connection changes (new neighbor or lost neighbor), a node can calculate an announcement interval for a given error rate of working communication links. Fast moving nodes will detect many changes, which will lead to increased announcement rates and finally to a reduced neighbor detection delay.

Nodes can also measure the announcement intervals of their neighbors to measure their own movement. This approach can have unpredictable results because the nodes influence each other permanently. The use of such indirect information to calculate announcement intervals should be minimized.

A problem in ad hoc networks is the detection of connection losses. If a connection between nodes breaks due to mobility or transmission errors, there will be no loss message. Only the missing announcements will indicate the breakdown of a connection. To confirm a breakdown, short priority message can be used to request a neighbor announcement over a possibly broken link. If not answered within a given time, the connection can be considered broken. The use of dynamic announcements forbids a simple fixed time count to detect a missing announcement. Therefore, we propose to add a time field to every announcement which informs the receiving nodes when the next announcement should be expected.

3. Implementation

To evaluate the proposed improvements by adaptive neighbor discovery, we implemented them as part of our simulator for wireless networking protocols [5, 8]. Our dynamic adaptation algorithm is based on measurements of detected network topology changes. This change rate can be measured and used as the prime parameter for the calculation of appropriate announcement intervals.

In the simulator, the announcements and signals of nodes are sent to a central distributing unit. This distributing unit knows the transmission range of every node and distributes the announcements only to nodes in the transmission range of the sending node. Every node maintains a table to store the registered changes and the corresponding times. Thus, every node has its own view of its environment and can calculate its own announcement interval accordingly.

An announcement contains the following information:

a) the identity of the sender (ID)

b) a time value describing when to expect the next announcement

c) information about status of the sender

Each node has a table with an entry for every known neighbor. The table contains the recent announcements of all neighbors and will be updated when announcements arrive. When an announcement is received from a neighbor with no table entry, the table will be updated and an immediate reply announcement is sent. A connection loss to a neighbor is detected when the time limit for the expected next announcement is exceeded. The table entry of this node will be deleted if the second time value for the next announcement is also exceeded. Every change in the table is an event registered in a second table. When a node detects a topology change, it saves the time since the last event happened in a history table. This table has a limited size, so a new value may delete the oldest table entry. The table is used to calculate the average time between changes. An estimation of the time requires at least one table entry. The time is calculated as a weighted average over the history table. The weights are needed to increase the importance of recent entries over older entries. The basic principle of the adaptive neighbor discovery algorithm is shown in Figure 1.

![Figure 1: Adaptive neighbor discovery](image)

Only a few parameters are needed. All nodes have a table of size $N$ showing the time between changes. This table is empty at startup and can grow up to $N_{max}$. The entries of the table are $T_i$ to $T_N$. A new entry is always put as $T_i$ into the table, the index of the old values is increased by 1. If a table size of $N_{max}$ is reached, the oldest entry $T_N$ is deleted. The set of weights $W_i$ to $W_N$ are integers describing how recent events are more important than older events and control how fast the algorithm will adapt to measured changes.

The average time between changes $CT_{avg}$ is calculated as

$$CT_{avg} = \frac{\sum_{i=1}^{N} T_i \cdot W_i}{\sum_{i=1}^{N} W_i}$$

(3)

It is necessary for reliable node detection to set a minimum announcement period for the case of no
changes through a long time period to enable at least a
delayed detection of new changes. Also, a maximum
announcement period should be set to prevent a network
from overreacting to short periods of frequent changes.
The parameter $AR_{\text{min}}$ describes the minimum and $AR_{\text{max}}$
the maximum limit for the announcement rate,
respectively the announcement period of node changes
between these two rates, controlled by the estimated
mobility in the network. The adaptation is controlled by
the announcement factor $AF$. This factor describes how
fast announcements should be sent for a given $CT_{\text{avg}}$. The
adaptive time between announcements $AT_{\text{ad}}$ is then
calculated as

$$AT_{\text{ad}} = \min(AR_{\text{max}}, \max(AR_{\text{min}}, CT_{\text{avg}} \cdot AF))$$

and will be used for the next announcement.

4. Simulation Results

To compare static and dynamic announcement
algorithms, a number of simulations have been performed
using our discrete event simulator for mobile wireless
networking protocols [5, 7]. The simulation setup consists
of 100 nodes which are placed in a rectangular area (3000
x 3000 meters). All nodes are allowed to move and have
an equal fixed transmission range. The equal transmission
range guarantees that all connections work in both
directions. A practical example for this kind of setup is a
large construction area, where people with mobile devices
move around. The transmission range of a single device is
limited to 400 meters; it covers only a small part of the
area. The cooperation of nodes enables communication
across the entire area. The movement speed of the nodes
ranges from zero to 15 meters per second. The first
simulation runs are based on the assumption that the
network is loss free. Communication between nodes is
enabled or disabled only by the movement of nodes. The
nodes have unidirectional discrete transmission ranges;
below the transmission range, two nodes will be able to
receive packets from each other. All nodes move at an
equal speed to randomly chosen destinations in the area.
Then, a new destination is chosen and movement
continues. These first simulation series were conducted to
investigate the correct connection discovery rate and the
number of announcement packets.
The simulation system generates a complete map of all
existing links in the network; this map is compared to the
map calculated by the tested algorithm. All parameters
used for adaptation are listed in Table 1. The results are
shown in Figure 2, where the $x$-axis shows the node
movement speed (in meter/second), and the $y$-axis shows
how many errors have been recognized for the neighbor
detection. There are basically two kinds of detection
errors: new neighbors and lost neighborhood relations that
have not been discovered.

Figure 2 shows the results of 5 series of simulation runs.
Each simulation run analyzes one hour of networking
time with nodes at a given speed. We compared our
adaptive algorithm against an algorithm with constant
announcement intervals using four fixed announcement
periods ranging from 1 to 60 seconds. During the
simulation, some topology changes occurred that were not
immediately recognized by the test algorithms. These
recognition errors were counted by the simulator and used
as the performance criteria. The simulator counts how
long lost connections are still considered to be available
and how long new connections are not being discovered.
The performance of an algorithm is measured by the
average number of wrong or undiscovered links of a
single node in the network.
The adaptive algorithm in Figure 2 is configured to limit
the maximum error rate. It is evident that the adaptive
algorithm achieves this goal almost independent of node
movement speed.

Figure 3 shows the average bandwidth consumption
caused by the algorithm with respect to the node
movement speed measured by the number of packets sent.
The average bandwidth consumption of the algorithm
with constant announcements is independent of
movement speed. The slight increase with movement
speed is caused by the use of the immediate reply rule.
The average bandwidth consumption of the proposed
adaptive algorithm increases with movement speed but is
well below the consumption for the 1-second constant
announcement algorithm.

We also tested the adaptive algorithm with highly
different movement speeds for the individual nodes. A
network with equally distributed movement speeds
ranging from zero to two meters per second shows an
expected average error rate of 0.38 of working
communication links and a total average bandwidth of
200,000 packets.
A more realistic scenario where most of the nodes are not moving or have only low speed and only 10 percent of the nodes move with high speed shows the same error rate but a reduced consumption of 130,000 packets. These results demonstrate the capability of the adaptive algorithm to guarantee an average constant error rate of working communication links regardless of node movement speed. Another important aspect is the tolerance of the adaptive algorithm against movement pauses and speed changes. To test this scenario, the simulation series was repeated with a modified movement pattern. A node chooses a target point, moves to it and pauses there for a randomly chosen time between 1 and 1600 seconds. After pausing, the node starts moving to a new target point. Figure 4 shows the average number of connection errors each node experienced during simulation. The error level decreases, because larger pause times generally lead to lower movement and therefore to less topology changes. The results of the 60 second static algorithm were omitted because of error rates beyond scale (they start about 10 for zero pausing time). Figure 4 shows for zero pausing time exactly the same results as Figure 2 for a moving speed of two meters per second. The results clearly show that the average error level of the adaptive algorithm is still constant compared to a non-adaptive algorithm. Figure 5 displays the average bandwidth consumption. The consumption of the adaptive algorithm reaches the same level as the static 4-second interval algorithm. However, the corresponding error results of Figure 4 show a much better performance. This indicates that the adaptive algorithm is a good choice for situations with irregular moving patterns. Now, we analyze the distribution of detection errors within nodes. To evaluate the distribution we measure the time a node suffers from undetected new links or undiscovered link breaks. If a node has multiple errors, the time duration of each error is counted separately. These times are used for evaluating the average detection error each node has to deal with. Figure 6 shows the error distribution of 3 groups moving at highly different speeds.

Table 2 lists the movement parameters and results for all groups. The results show the equalization effect of the adaptive algorithm. Nodes using the algorithm have less variance in their detection accuracy compared to nodes using a constant announcement. This leads to the conclusion that nodes benefit uniformly from adaptive connection discovery.

5. Impact on Routing

All previous simulations evaluated the performance of continuous and adaptive link detection. In the next step, we expand the simulation to a routing algorithm that uses...
the data from connection discovery and calculates routes to all possible nodes. The used TERA [7] algorithm is based on asynchronous distributed distant vector routing [2], but uses several additional tables to allow path reconstruction. This avoids the count to infinity problem. Tree based routing allows to store a path from a destination to root A in the predecessor entries \( V_i^A \) as shown in Figure 7. Any path can be traced back to the root using these entries, enabling the detection and avoidance of any loop. Each node in the network has an identifier \( k \). These nodes are able to communicate with each other via links which are assumed to have a cost (called “distance”). The distance between source \( A \) and destination \( i \) is given by \( d_{ij} \) and must be positive. Nodes with \( d_{ij} < 8 \) are called neighbors. Without a direct way of communication between nodes \( i \), \( j \) the distance is assumed to \( d_{ij} = 8 \); the distance of a node to itself is zero \( (d_{ii} = 0) \). The status of a node \( k \) is defined by the following tables:

- The routing table \( D_i^k \); the best available distance to \( i \)
- The successor table \( N_i^k \); id of the neighbor which used to forward to \( i \)
- The predecessor table \( V_i^k \); id of the predecessor node of the chosen shortest path.

The next two tables store all collected information available due to neighbor announcements:
- (c) The neighbor predecessor table \( V_i^j \); id of the predecessor of \( i \) offered by neighbor \( j \).
- (d) The neighbor distance table \( ND_{ij}^k \) shows the best available distance to destination \( i \) offered by neighbor \( j \).
- (e) The neighbor predecessor table \( NV_{ij}^k \) shows the predecessor in the shortest path to \( i \) offered by neighbor \( j \).

\[
d_{ij} = \begin{cases} \text{distance of a node to itself} & \text{if } i = j \\ \text{distance from } i \text{ to } j & \text{otherwise} \end{cases}
\]

**Figure 7: Path recovery in tree exchange routing**

After initialization, the iteration process with the iteration counter \( t \) begins. Every node starts a neighbor update and predecessor correction for the received minimum spanning trees [7]. Then, node \( k \) recalculates a new spanning tree using the information stored in \( ND_{ij}^k(t) \) and \( NV_{ij}^k(t) \). The new minimum spanning tree is stored in tables \( D_i^k(t) \), \( N_i^k(t) \) and \( V_i^k(t) \). If a calculation results in different tables compared to the result from the previous iteration, the tables \( D_i^k(t) \), \( V_i^k(t) \) must be sent to all neighbors which are \( \{i \mid d_{ij} < \infty\} \). All kind of distance vector algorithms use this iteration process which causes information distribution in a step by step manner. Each iteration disseminates the information one hop forward through the network.

Due to the recalculation in each node, only important information will be sent to neighbors; this is the first step to bandwidth efficiency in routing. With unrestricted shortest path calculation, the tables \( D_i^k(t) \) and \( V_i^k(t) \) must be sent to all neighbors if at least one of the following criteria is true:

\[
D_i^k(t) \cap D_i^k(t-1) \neq \emptyset \vee N_i^k(t) \cap N_i^k(t-1) \neq \emptyset \vee V_i^k(t) \cap V_i^k(t-1) \neq \emptyset
\]

The simulations use nearly the same parameters as the previously described simulations. For computational reasons, the number of nodes was reduced. Thus, the evaluation of reachability is based on the following parameters: 50 nodes in an area of 1000x1000 meters; varying node movement speeds; pausing nodes with a mean of 300 seconds pause time and a transmission range of 200 meters. The result in Figure 8 presents the average number of correct routes of a node as a function of node movement speed. The lifetime of all correct routes is calculated during simulation and used to specify the average number of correct routes.

**Figure 8: Evaluation of working routes**

The simulation shows that at low mobility all algorithms will perform well and offer a mean of 45 working routes. The mean value does not reach the maximum of 49 possible routes due to network partitions. The adaptive algorithm shows almost constant detection accuracy and has almost constant performance in finding routes. The last two experiments analyze the bandwidth consumption of connection discovery and routing. The packets sent for routing were counted separately from the announcement packets to compare the bandwidth consumption between routing and announcements. Figure 9 shows the total number of packets.

The number of packets strongly depends on the movement. This is obvious because the number of topology changes is proportional to node movements and thus to routing calculation which is always performed when a change of topology is registered. We noticed an increased consumption of the routing algorithm when it suffers from a lot of false information about available connections.
There is a correlation between routing and connection detection packets exchanged for routing. Packets exchanged for routing can indicate new connections in the same way like announcement packets. If a node receives routing packets from undiscovered neighbors or fails to send packets it will inform the connection detection. This results in a better performance of static and adaptive connection detection in combination with routing. Figure 10 shows the bandwidth consumption caused directly by connection discovery. It shows that that the route discovery requires as much bandwidth as the routing process itself. In case of a 1-second announcement interval, more bandwidth is spent for announcements than for any routing calculation.

6. Conclusions
We have presented an adaptive algorithm for connection discovery in mobile wireless networks. Our main target was the dynamic adaptation of the algorithm to the movement of nodes to get maximal service availability with minimal bandwidth consumption. We have demonstrated the importance of connection discovery, which can require more bandwidth than routing calculation for a given error rate of working connections. The results show clearly that an adaptive algorithm for neighbor discovery is capable of keeping the error rate at an almost constant level regardless of node movement. In contrast, a constant error rate causes some unavoidable costs, since high movement speed requires more announcements within the same time. The simulations have indicated that adaptive connection discovery is well suited for proactive routing in wireless networks. Routing algorithms like TERA that use the knowledge about connection stability and route distance to minimize the communication overhead for route maintenance can clearly benefit from adaptive connection discovery by adapting path optimization to node movement. There are several issues for further research. Most importantly, we are currently examining routing algorithms and combinations of routing algorithms which use the knowledge about connection stability to minimize communication overhead for route maintenance by selecting routes that do not contain highly mobile nodes for services that require a certain amount of reliability.

References