Detecting the Presence of Nodes in MANETs

Thi Minh Chau Tran Björn Scheuermann Martin Mauve

Computer Networks Research Group
Heinrich Heine University
Düsseldorf, Germany

{tmct,scheuermann,mauve}@cs.uni-duesseldorf.de

ABSTRACT

While mobility in the sense of node movement has been an intensively studied aspect of mobile ad-hoc networks (MANETs), another aspect of mobility has not yet been subjected to systematic research: nodes may not only move around but also enter and leave the network. In fact, many proposed protocols for MANETs exhibit worst-case behavior when an intended communication partner is currently not present. Therefore, knowing whether a given node is currently present in the network can often help to avoid unnecessary overhead. In this paper, we present a solution to the presence detection problem. Our method uses a Bloom filter-based beaconing mechanism to aggregate and distribute information about the presence of network nodes. Analytical and simulation results show interesting properties of presence detection in wireless multihop environments and underline the effectiveness and practical applicability of our approach.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms

Algorithms, Design, Performance

Keywords

Presence Detection, Mobile Ad-Hoc Networks, MANETs, Soft State Bloom Filter

1. INTRODUCTION

Node mobility is a key challenge in mobile ad-hoc networks. As a consequence, the impact of a dynamic network topology on medium access, network and transport functionality has been studied extensively. However, there is another aspect of mobility besides having to deal with a dynamic network topology. Mobility also implies that nodes may enter and leave the network at any time. This can happen either physically by entering or leaving the network area, or logically by switching the networked device on or off. In contrast to supporting dynamic topologies, the impact of varying node presence has not yet been systematically studied, although it can affect the performance of a network significantly.

For example, let us consider reactive routing with protocols like DSR [12] or AODV [18]. These have been designed and evaluated under the premise that all communication partners to which a route is to be established are actually present in the network. Routes are found by flooding route requests. The flooding is repeated if no answer arrives within some time interval. Thus, if the intended communication partner is not present, route discovery causes a maximum amount of unnecessary network traffic. The problem could be avoided if there was a way to tell whether some potential communication partner is currently present or not. While proactive routing protocols would certainly be able to provide this service, they additionally spend resources to track all other mobility induced topology changes. This overhead was the key reason to develop reactive routing protocols in the first place. We therefore argue that a presence detection scheme should be lightweight: it should track only the presence of nodes and not the state of all links.

Many other protocols and applications for mobile ad-hoc networks could likewise profit from presence detection. Further examples are mechanisms for service discovery, where one wants to find a provider of a certain service, or location services used for geographic routing. Both could benefit from a way to check whether the subject of the query is present at all, before actually attempting to locate it.

Furthermore, as shown by Gupta and Kumar [8], the per-node capacity of an ad-hoc network decreases dramatically with the average distance between communication partners. Therefore, it is desirable that applications and algorithms for mobile ad-hoc networks are able to determine the distance between communication partners before attempting to exchange data.

In this paper we propose a lightweight presence detection service for mobile ad-hoc networks. It enables nodes to check whether other nodes are present within a given hop-count radius. Our approach is based on a space-efficient approximate set membership data structure called Bloom filter [1]. Essentially, this provides a lossy compression of presence information in order to minimize communication overhead. In our approach, the nodes periodically announce the compressed presence information that they have gathered so far in beacon messages. They integrate the information received from neighbors into their own knowledge base and include it in their next announcement. Stale presence information about nodes that have left will automatically vanish.

CHANTS'07, September 14, 2007, Montréal, Québec, Canada.

The cost of aggregating presence information is a small number of so-called false positives, where nodes are wrongly considered to be present. This is not critical for typical applications of a presence detection service. When such a service is used, for instance, to avoid unnecessary route discoveries with reactive routing, the cost of a false positive is an unnecessary discovery attempt. This is exactly what happens without presence detection. Thus, in the rare error case, the system behaves like one without presence detection.

This paper has two major contributions. First, it points out the problem of lightweight presence detection as an important open research problem in the context of MANETs. Furthermore, it describes a solution to the problem and analyzes its performance.

The remainder of the paper is structured as follows. In Section 2, we summarize related work. Section 3 presents the algorithm we propose for lightweight presence detection. Section 4 evaluates our approach both analytically and by means of ns-2 simulations. In Section 5, we discuss an application of our approach for avoiding unnecessary overhead in reactive MANET routing and present corresponding simulation results. Finally, the paper is concluded by a summary and an outlook on future work in Section 6.

2. RELATED WORK

Although detecting the presence of nodes is a largely unexplored field, not only in the context of MANETs, there are a number of research directions that deal more or less directly with the presence of nodes in wireless networks.

Routing protocols for mobile ad-hoc networks are able to determine the presence status of a node. However, reactive protocols like AODV [18] or DSR [12] induce very high, unnecessary network traffic in the case of a route discovery to a non-present destination. Thus, they might actually benefit from an additional presence detection service. In contrast, proactive routing protocols like OLSR [11] can determine the presence of a node. However, as stated in the introduction this comes at the cost of tracking the complete network topology, too. This is ony viable if a proactive routing protocol can be efficiently employed in a given environment.

Close relatives to presence detection are location services for geographic routing. Examples are homezone-based systems [7,20], the grid location service (GLS) [16], and the hierarchical location service (HLS) [13]. However, a location service provides significantly more information about a node than just telling whether it is present or not, namely its current position. Therefore, the cost to keep it up-to-date and to lookup in such a service is much higher. Like reactive routing, most location services exhibit their worst-case behavior in terms of effort in the case of a request for information about a non-present node.

Some systems for presence detection in wireless systems have been successful applications on their own, for example the Lovegety [10]. A number research projects deal with the exchange of presence information via single-hop wireless communication, with or without infrastructure. Examples are [4,9,21]. But none of these systems considers presence detection over multiple wireless hops.

Finally, [15] presents a design for an instant messaging system for sparse mobile ad-hoc networks called SPEED. Instead of the presence of nodes, the authors consider the dissemination of presence states of users, such as “available”, “busy”, or “do not disturb”, with the assumption that users’ devices stay in the network even if a user is “non-present”. SPEED distributes user presence information via periodical announcements and requests, both of which are flooded in the network. This design is well-suited for an instant messaging service. However, for our purposes a much more lightweight solution is necessary. Interestingly, like many of the previously discussed protocols, SPEED generates significant overhead in the case of users whose devices are not present in the network. Therefore, it might in fact profit from additional node presence detection in such cases.

3. SCALABLE PRESENCE DETECTION

We assume that each node has some static, unique ID. This could, for example, be a MAC- or statically assigned IP address.

The most naive approach for a presence detection mechanism would be simply distributing a list of all available node IDs throughout the network. This could be done, e. g., by transmitting the IDs of the nodes via beacon messages. The obvious problem of this approach is that the amount of information distributed to each node would increase linearly with the number of nodes in the network. This is not appropriate for a lightweight service.

In order to avoid this problem, we propose to aggregate presence information. Generally, aggregation can be lossy or lossless. Lossless aggregation is efficient if there is some structure in the entries to be aggregated. For instance, IP routing table entries can be efficiently aggregated without losses, because the addresses are organized hierarchically. For nodes present in a MANET, such a structure is typically not given. Thus, we propose to use a lossy aggregation scheme.

When performing lossy aggregation of presence information, two types of errors may occur. Either a node may be reported as being present while it is not, or it may be reported as being absent while it is in fact present. The former situation is called a false positive, the latter a false negative. Given the application of presence detection in mobile ad-hoc networks, a false negative would “hide” an actually present target node, which is generally not acceptable. A low rate of false positives, on the other hand, is often quite tolerable: in the rare case of a false positive, the applications would simply behave as if there was no presence detection service active, e. g., a routing protocol would attempt to set up a route to a node not present in the network. The actual absence of the node, and thus the occurence of a false positive will become clear if communicating with the node fails. Therefore, for presence detection a scalable data structure that supports lossy aggregation of presence information without introducing false negatives is needed. Bloom filters are such a data structure.

3.1 The Bloom Filter

A Bloom filter [1] is a data structure that represents a set $S = \{s_1, s_2, \ldots, s_n\}$ of n elements to support membership queries. It is described by an array a of m bits, which are initially set to 0. k independent hash functions $h_1, \ldots, h_k$ are used, each maps every possible item in the set to a uniformly distributed value in the range $\{1, \ldots, m\}$. There are two basic operations that can be performed on a Bloom filter. New elements can be added, and the presence of an element can be queried.

To add a new element s, which is a node ID in our specific case, the bits at positions $h_1(s), h_2(s), \ldots, h_k(s)$ in a are set to 1. In order to determine the presence of some node x, the bits at these positions are checked. If any of these is 0, then it is certain that x is not in S. Otherwise, it can be assumed that x is in S with some remaining probability of a false positive that occurs when an element is actually not in the set, but all respective bit positions have been set to 1 by adding other elements.

The union of two Bloom filters is calculated by a bit-wise OR operation. Hence, to dissemminate presence information, nodes may periodically send beacons containing a Bloom filter of the node IDs they know are present. Upon receiving such a beacon, they can merge the received information with what they already know.

However, in a presence detection system the removal of no longer present nodes is also necessary. The standard Bloom filter has the central drawback that there is no method to delete a value once it has been added. Since it is entirely possible that a given bit position has been set to 1 during the addition of more than one element, simply deleting all bit positions that refer to the element that should be removed is not an option.

3.2 Presence Detection

If we ignore the specifics of Bloom filters for a moment, there are two alternative approaches to remove the presence of a node from an aggregate. The first is to remove it explicitly when the node leaves the network. This is called a hard state approach. Alternatively, the presence information for each node can decay over time and has to be refreshed periodically in order to remain in the aggregate. Using this technique yields a so-called soft state approach.

There are extensions of Bloom filters (e. g., the counting Bloom filter in [5]) that enable explicit removal of items. Those could be used as a basis of a hard state approach. However, a hard state approach faces two major challenges when used for presence detection in MANETs. First, it must be guaranteed that the event of a node leaving the network actually triggers the removal. Second, the information on this event needs to be distributed in the network. Both problems are very hard to solve in the given decentralized environment. We therefore use a soft state approach. This requires an extension of Bloom filters that enables decay and refresh operations.

Soft state Bloom filters

In order to support these, we modify the Bloom filter used in the aggregates. Each entry now consists of l bits instead of one, where l is typically small, e. g., $l = 3$ or $l = 4$. Each of these l-bit words stores a counter. These counter values can be interpreted as the "age" of the respective Bloom filter entry. A node initializes all counters with the maximum value $2^l - 1$. This maximum value indicates that the position of the Bloom filter is not set, it is equivalent to setting a bit position to 0 in the standard Bloom filter. Periodically, just before a beacon is sent, a node applies the hash functions to its own ID. It sets all counters at the resulting positions to 0. All other counters that are not already at the maximum value are incremented by one. Thus, each node continuously announces its own presence with the age of 0 and ages all other presence information by 1 each interval.

The aging of the information means that entries of leaving nodes will eventually die out. Hence, it provides the removal of nodes from the Bloom filter if they are no longer present. As already discussed, it is also desirable to be able to put a limit on the distance to a target node for it to be considered present. Applications can thereby take the expected communication effort into account, respecting the inherent capacity limits of wireless multihop networking. More formally, this can be stated as: node x is regarded as present by some other node u if and only if the length of the shortest hop-count path from u to x does not exceed some threshold T_u. Thus, x should be regarded as non-present by u if either x is not at all in the network or it is too far away. Since the entries in our scheme age with every hop over which the information is propagated, their value—as a side effect—also provides an indication of the distance to sought-after nodes.

Our Bloom filter modification bears certain similarities to some of the many Bloom filter variants that have been discussed in the literature. For example in the routing context, in [19] and [6] schemes are discussed which use a set of Bloom filters: there is one Bloom filter for each hop distance, containing the information on all nodes at that distance. In [14], a scheme for service discovery is proposed that stores the minimum distance to a service using Bloom filters. The Time-Decaying Bloom Filter (TDBF) as introduced in [2,3] is used to continuously analyze a data stream like, e. g., web page hits. Like in counting Bloom filters [5], but unlike in our approach, the *number of occurrences* of an item is stored in counters at each bit position. TDBF then reduces the contribution of less recent occurrences by periodically decaying all counters. However, to the very best of our knowledge no previously existing Bloom filter variant provides a solution for the the central problem tackled by our soft state Bloom filters: the removal of elements in a Bloom filter if they are no longer refreshed.

Aggregate structure

We define the structure of an aggregate as follows. It consists of m l-bit entries $a_1, a_2, \ldots, a_m$, each of which is interpreted as an integer in the range $0, \ldots, 2^l - 1$. a_i represents the age of the Bloom filter's "bit" i. Initially, all the a_i are set to $2^l - 1$.

l needs to be chosen large enough to account for the maximum of all nodes' distance thresholds. Thus, if N is the set of nodes,

$$l > \max_{u \in N} \lceil \log_2 T_u \rceil. \tag{1}$$

Each node u can select and change its T_u at will, but l is fixed and constant for the whole network.

Timeout and refresh

To accommodate the new aggregate structure, the operations on the Bloom filter are modified accordingly. The most central one is the timeout and refresh operation. Each node performs it periodically. It consists of three steps:

1. Increment each a_i by one, if it is not already at the limit of $2^l - 1$.

2. Refresh the information about the node's own presence, by setting $a_{h_j(ID)} = 0$ for all $j = 1, \ldots, k$, where ID is the ID of the local node.

3. Broadcast the updated aggregate to the neighbors.

This algorithm results in each position eventually reaching the maximal value when it is no longer refreshed by some node. Therefore, a no longer present node will vanish from the aggregate.

Merge operation

When a node receives a beacon message, the information is merged into the local aggregate. Instead of the bit-wise OR for standard Bloom filters, we use a position-wise minimum operation, i. e., we set each Bloom filter position to the minimum of local and received ages.

Query operation

In order to determine whether some other node x is present, a node u checks its local aggregate at positions $h_1(x), h_2(x), \ldots, h_k(x)$. Let

$$t := 1 + \max_{1 \le i \le k} a_{h_i(x)}. \tag{2}$$

If $t = 2^l$, we conclude that x is not present. Otherwise, we say that it is seen at a distance of t hops from u, with some probability of a false positive. Depending on the choice of the threshold T_u, u considers x as present or not accordingly. We will soon see that this "seen distance" is in fact very close to the real minimum hop distance.

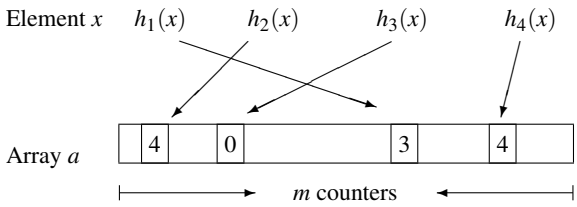


Figure 1: A soft-state Bloom filter.

Figure 1 shows the entries corresponding to some sought-after node x. The age of the information about x is four, which means that x is seen at a distance of five hops. For instance, if $T_u = 10$ then x is said to be present. If $T_u = 3$ the node is said to be not present, as the distance at which x is seen exceeds the value of T_u.

4. EVALUATION

In the previous section, we introduced an algorithm to disseminate presence information and to reliably remove no longer present nodes with a soft state approach. In this section, we assess the performance and suitability of the proposed scheme. In particular, we concentrate on three aspects: the reliability of the scheme in terms of the false positive rate, oscillations of the seen distance, and the effect of node mobility.

4.1 False Positive Rate

For a practical application of presence detection, it is desirable to know the false positive rate: what is the probability of considering a non-present node to be present?

A false positive occurs when the bit positions corresponding to the sought-after node are all set by other added elements. In standard Bloom filters, the probability that a false positive occurs depends on three factors: the number of bit positions in the filter m, the number of hash functions k, and the number of elements n that are present in the set. The probability that a bit position is still zero after n elements with k bit positions each have been added is $(1-1/m)^{kn}$. Thus, the probability that all k bit positions of the sought-after node are one can be calculated as

$$\left(1-\left(1-\frac{1}{m}\right)^{kn}\right)^k \approx \left(1-e^{-kn/m}\right)^k. \qquad (3)$$

So, if standard Bloom filters were used, the false positive rate would depend on the total number of nodes in the network. For our modified Bloom filter with decaying of information and counters at each filter position, however, the situation is slightly more complicated—and significantly better.

Consider a node x that is regarded as present by u based on the local aggregate. As stated earlier, the maximum age at x's entries in the Bloom filter plus one is the distance in hops at which x is seen. Let this number like before be denoted by t. If x is actually not present *at this distance*, all of x's entries must have been set by other nodes. All these nodes must be at distance t hops or less. Let $n(t)$ denote the number of nodes within this maximum distance. Then, the probability that a non-present node is considered present at a distance of less than or equal to t is

$$P_{\text{fp}}(m,k,t,\rho) = \left(1-\left(1-\frac{1}{m}\right)^{kn(t)}\right)^k \approx \left(1-e^{-\frac{kn(t)}{m}}\right)^k. \qquad (4)$$

Consequently, the probability of a false positive at distance $t \geq 1$ is

$$P_{\text{fp}}(m,k,t,\rho) - P_{\text{fp}}(m,k,t-1,\rho). \qquad (5)$$

Being seen wrongly at distance t can either mean not being in the network at all, or being in the network but at a larger distance. These cases are hard to distinguish, since the Bloom filter positions that should contain information on x are all overridden by information on other nodes.

An important point is that the probability of a false positive at distance t does not depend on the total number of nodes in the network, but only on the number of nodes within the t-hop neighborhood, and therefore on the local network density. This also implies that the confidence in the presence of a node increases rapidly when it comes closer: the closer a node is seen, the higher is the chance that it is in fact there. This can be exploited by an application, by being more aggressive or spending more resources on contacting a node which is detected nearby and is therefore very likely to be actually present.

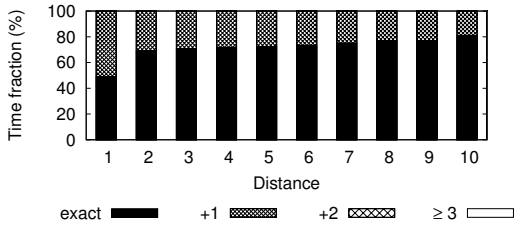
4.2 Oscillations

In the proposed presence detection algorithm, each node periodically increments the age of the Bloom filter positions in its aggregate before sending a beacon. While waiting for the next beaconing interval to expire, updates are received from the neighboring nodes, potentially resetting the incremented positions to their previous values. Therefore, the Bloom filter entries go up and down periodically. Furthermore, if beacons are lost—either due to transmission errors or because of congestion or packet collisions—, this will lead to temporarily further increased counters: information gets through only from time to time. This influences the distance at which other nodes are seen; the seen distance is not always exactly the minimum hop count distance, but it will oscillate between the exact distance and slightly higher values.

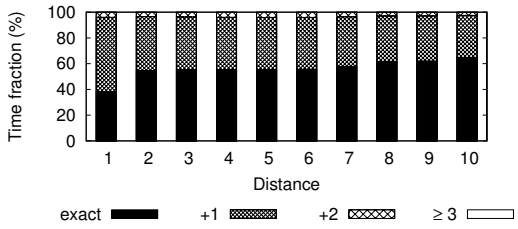
The oscillations due to the alternating decay and refresh operations always occur, even in the case of an ideal medium without losses. Higher oscillation amplitudes are caused by packet losses.

Simulations using the ns-2 network simulator [17] were carried out to see to which extent the error rates effect on the fluctuation of reported pair distances. We placed 200 nodes randomly on a square area of 1500 meters side length, and used our presence detection algorithm with $m = 1024$ filter positions, $l = 4$ bits counter length, and a beaconing interval of three seconds. A randomized error model was used to make a varying fraction of beacon receptions, 0 %, 25 %, and 50 %, fail, in addition to medium-related losses like those caused by packet collisions. Figure 2 shows the influence of the true distance between two nodes and the beacon packet error rates on the oscillations. Each vertical bar stands for a shortest hop distance; it is an average over all node pairs with the respective distance. It shows for which fraction of time the distance has been reported correctly, as well as how long and how far it has been overestimated.

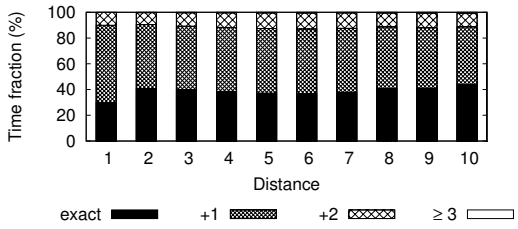
The results show that even high packet loss rates have surprisingly little effect on the oscillation of reported distances. For most of the time, either the reported distances between node pairs are correct, or they are overestimated by one, i. e., the decay operation has occurred and the Bloom filter positions have not yet been refreshed by a newly received aggregate. Distance estimates that are wrong by more than two hops are extremely rare: overestimates by three or more are barely visible in Figure 2 (c). The reason for this trait is that there often exist many alternative paths between a pair of nodes. They increase the chances for the information to get through and provide significant redundancy for the best path to be a fast one.



(a) Beacon error rate 0.



(b) Beacon error rate 0.25.



(c) Beacon error rate 0.5.

Figure 2: Oscillations of distance estimates for varying beacon packet error rates.

4.3 Node Movement

The movement of the nodes in a mobile ad-hoc network can influence the presence detection in two ways. First, the distances between nodes change over time, and there is some delay until these changes are correctly reflected in the presence detection aggregates. Furthermore, a node that moves to a different network area might essentially "carry" presence information with it, making nodes from the area where it is coming from appear closer to the nodes in the new vicinity than they actually are.

In order to assess the impact of these effects, we carried out another set of experiments with key parameters chosen as above, except that the nodes move according to the Random Waypoint mobility model without pause times and with different maximum speeds. The results given in Figure 3 show that at all considered movement speeds, the shortest hop-distances are reported quite accurately. For this figure, we took 100 snapshots at random points in simulation time, calculated the true hop-distances for each node pair based on the current node positions, and compared them to the distances reported in the respective presence detection aggregates. With increasing mobility, there is an increase in underestimated distances, due to the reasons discussed above. However, even when the node mobility is high, overestimated distances due to beacon losses and oscillations are much more common than underestimated ones. In the case of zero maximum speed, i. e., no mobility at all, there is a tiny number of cases in which the distance is underestimated; these are false positives that make a node appear closer than it is.

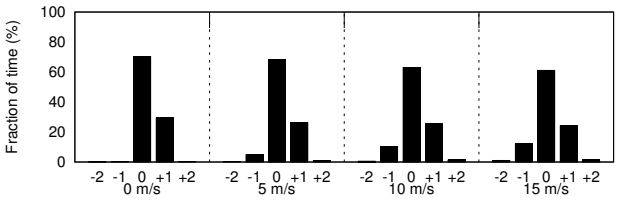


Figure 3: Deviations between seen distances and true shortest hop distances for different maximum node speeds.

4.4 Choosing the Parameters

In the discussed algorithm, there are quite a number of parameters: the Bloom filter length m, the number of hash functions k, the distance thresholds T_u to be used, the number of bits per Bloom filter position l, and the beaconing interval B. The best-suited values for these parameters depend on the application's requirements. Typically, one will find a tradeoff between bandwidth usage for beaconing, the delay of presence and disappearance detection, the false positive rate, and the achievable hop distance thresholds T_u. We consider it a very valuable property of our approach that it is possible to adjust this tradeoff in a very wide range, and thereby to tailor it for many specific application scenarios.

While it is relatively straightforward how one can find appropriate values for T_u, l, and B, the parameters that are directly related to the Bloom filter, m and k, deserve a little more attention. Actually, the optimal combination of m and k depends on the distance that is of main interest to the application: for given m, the optimal k for a minimum false positive rate is not the same for each distance. The analytical results regarding the false positives provide some hints. For given Bloom filter length m and node count $n(t)$ within the considered t-hop radius, the false positive rate according to (4) is minimized for

$$k = \frac{m \cdot \ln 2}{n(t)}. \tag{6}$$

In practice, k must of course be chosen to be an integer.

Before we focus on a specific application for presence detection in the next section, let us now consider a concrete example network to illustrate what our results presented above actually mean in practice. Consider a mobile ad-hoc network using a 1 MBit/s channel (which is actually quite limited, considering today's wireless hardware). We allow each node to spend 0.2 % of this bandwidth for presence detection beacons. Say that there is a total of 200 nodes in the network, which has a diameter of approximately ten hops. Let us furthermore assume that, for the considered application and network, a beaconing frequency of one beacon every three seconds suffices. So, the presence detection beacons may have a size of up to 3 s· 2 KBit/s = 768 bytes. We want to cover the whole network with the presence detection, so we may set $l = 4$. Therefore, we can use $m = 1400$ and still have plenty of space left for headers. If we decide to optimize the false positive rate for longer distances, i. e., for the whole network, (6) suggests we use $k = 5$ hash functions. In this configuration, the expected false positive rate for a node at maximum search radius is 3.47 %. This will be fine for many applications. By spending slightly more bandwidth, 0.25 % instead of 0.2 %, and using $m = 1800$ and $k = 6$, the expected false positive rate is reduced to 1.33 %. Within a smaller search radius, the false positive rate also quickly decreases: if, for example, within some smaller radius there are only 100 nodes, there will only be 0.24 % false positives in the 0.2 % bandwith usage case, and 0.05 % when allowing to use 0.25 % of the bandwidth.

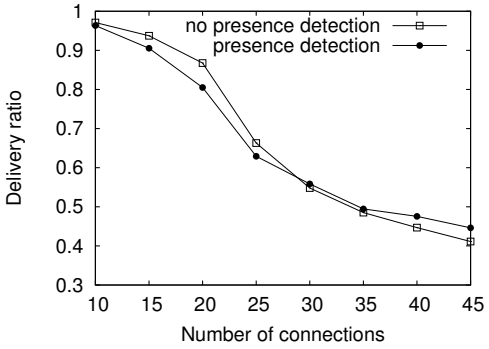


Figure 4: Worst case network performance.

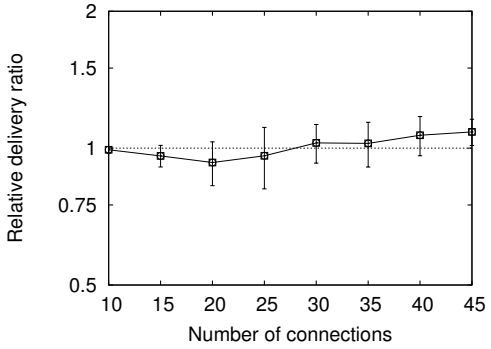


Figure 5: Relative performance in the worst case.

5. AN EXAMPLE APPLICATION

After having discussed the general features and applicability of presence detection in mobile ad-hoc networks, we now focus on one possible application. Finding a route to some destination node when a reactive routing protocol is used can be an expensive process, as it typically requires flooding a route request in the network. If the destination device is not present or too far away from the source device to be reached within the TTL limitation of the route request packet, much bandwidth is wasted. Furthermore, a route request is typically repeated if no answer arrives before some timeout expires, thus reactive routing exhibits worst-case behavior in the case of a non-present node.

In this section, we will analyze the presence detection service when it is used in conjunction with AODV routing [18]. We query the presence detection service at the source node before starting a route request for a new connection. If the destination node is considered present, the connection will be initiated as usual. Otherwise, the route request will be delayed until the presence detection service indicates that the destination node is present.

The simulation study was conducted using ns-2 [17], with setups similar to those used before. There are 200 nodes at random positions in an area of 1500 by 1500 meters. IEEE 802.11 at 1 MBit/s network bandwidth is used; note that a low communication bandwidth is the worst case for a protocol that causes a constant beaconing load. The communication radius is 250 meters, with a 550 meter carrier sense range. The nodes move according to the Random Waypoint mobility model at a maximum speed of 10 meters per second and with a pause time of 20 seconds. All connections last 100 seconds and start at some random time between 10 and 190 simulation seconds. During a connection, the source node sends CBR traffic with four data packets per second, each with a payload size of 512 bytes. The results are averages over 25 scenarios, each with different traffic and movement patterns. Here, compared to the example given in the previous section, we can tolerate even some more false positives, because the negative impact is at most an unnecessary route discovery—ten percent seems tolerable. We trade the additional false positives off for a reduced beacon size and use $m = 1024$, $k = 4$, and a beaconing interval of three seconds.

5.1 Worst Case Performance

The worst case for the presence detection service is a situation where all network nodes are permanently present. In that case, the network will not profit from presence detection services, while they still consume bandwidth. Figure 4 shows the network performance, in terms of the packet delivery ratio, for an increasing number of connections. For a lower network load, the negative impact of the presence detection is quite low. More interestingly, when the network load increases, there is actually a slightly better performance with presence detection.

Not only for the worst case evaluations, but also in our subsequent simulations, the results come from many different, randomly generated topologies. Their characteristics and thus the measured values vary in a very broad range. However, not only their variance, but also the covariance is high. For the same number of connections, in "good" topologies, the delivery ratio both with and without presence detection can be in the order of 90 %, while in a "bad" topology both are only around 50 %. To provide a more robust measure, we also look at the *relative performance:* for each single topology, we divide the delivery ratio with presence detection by the delivery ration without presence detection. The value will be one if both perform equally well, above one if there is a performance benefit with presence detection, and below one in case of a performance degradation. In Figure 5, we give the (geometric) mean of those values. The error bars show the standard deviation. A logarithmic y-axis avoids a distorted representation: a value of 0.5 is the loss of performance corresponding to the performance gain indicated by a value of 2. The results confirm that there is a small degradation in case of a lower network load, and a small improvement in higher load situations.

The reason for these unexpected performance benefits with presence detection is a little subtle. When the network is so congested around some node that it cannot send its data packets, the node will not send any presence announcement until the situation improves. This is an acceptable behavior since there is no use for a node in a congested area to announce its presence: the node is not able to receive any more data packets anyway. When congested nodes are temporarily considered non-present, connections attempts to those nodes will be delayed until they are again able to send beacons, i. e., they are effectively back in the network. Thus, employing presence detection has accidentally introduced some form of congestion control. We do not consider this to be a true advantage of presence detection, but it is certainly an interesting observation which may be exploited in future work.

5.2 Introducing Absent Nodes

Now we keep the number of "working" connections to present, available nodes fixed at 25. Figure 6 shows the effects on the network performance when the number of additional connection attempts to non-present nodes increases. The corresponding relative evaluation is shown in Figure 7. It can be seen that, without presence detection, more connection attempts to non-present nodes severely deteriorate the network performance. The performance with presence detection, on the other hand, does not show signif-

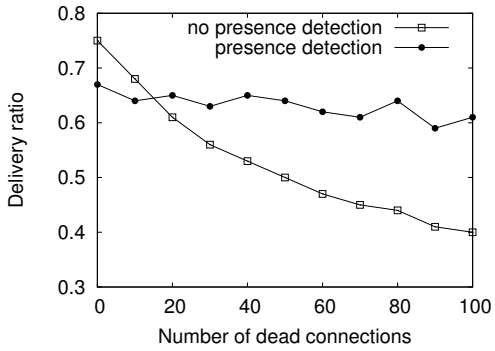


Figure 6: Packet delivery ratio as the number of connection attempts to non-present nodes increases.

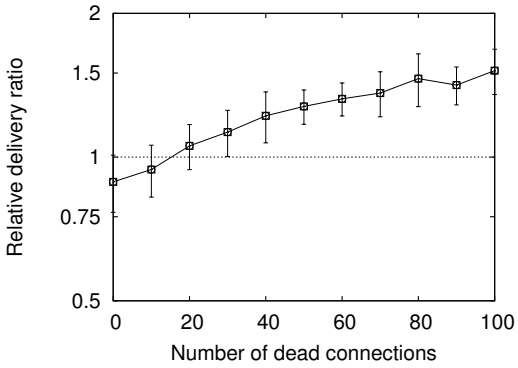
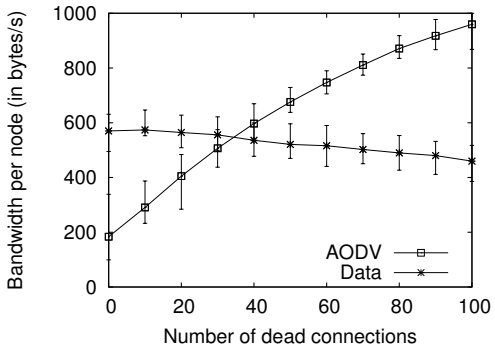
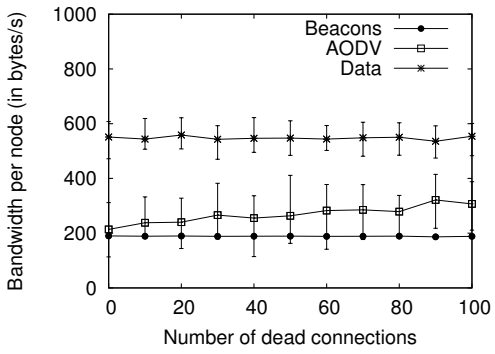


Figure 7: Relative performance with connection attempts to absent nodes.



(a) Without presence detection.



(b) With presence detection.

Figure 8: Bandwidth use per node as the number of connection attempts to non-present nodes increases.

icant negative effects, no matter how many connections to non-present nodes are attempted.

Figure 8 compares the average bandwidth spent by a node in the network with and without presence detection, broken down to bandwidth used for routing packets, data packets, and presence detection beacons. Here, for AODV and data packets, the error bars show the minimum and maximum over all our simulations. The beaconing load is constant. It is obvious that, without presence detection, the bandwidth used for routing packets quickly increases, while the available bandwidth for application data decreases. Only a small portion of the bandwidth is actually effectively used.

The situation with presence detection services running is much better. Since the presence detection service blocks the vast majority of connection attempts to non-present nodes, the bandwidth usage does not significantly change. There is only a minor increase in bandwidth used by AODV, which is due to false positives, causing some non-present nodes to appear as present.

Note that in a real network, the amount of connection attempts to non-present nodes heavily depends on the nature of the network and the application. Our results here show that it is possible to gain a benefit from exploiting presence information.

5.3 Arriving and Leaving Nodes

In our last set of simulations, nodes actually enter and leave the network. One third of the nodes to which connections are attempted is permanently present, another third are switched on at some random time during the simulation, and the remaining nodes are initially present, but are switched off at some random time. This means that some connections run smoothly, some others can possibly be delayed until the destination is up, or they might be abruptly terminated because the destination is switched off while the connection is running. In Figure 9 and the corresponding relative values in Figure 10, it can once again be seen that the presence detection service helps to achieve a substantially better network performance.

6. CONCLUSIONS AND FUTURE WORK

In this paper we investigated the problem of detecting whether specific nodes are present in a mobile ad-hoc network. We developed a scalable solution to this problem, which is also able to estimate the number of hops required to reach a given node. It aggregates presence information using Bloom filters, which we extended to a soft state variant. The aggregation comes at the cost of an adjustable amount of false positives, while it guarantees the absence of false negatives.

We believe that presence detection is a vital building block for wireless multihop networks. Service discovery, routing, and location services are just three examples where presence detection is vital to ensure an acceptable system performance if the presence of nodes changes over time. There are significant opportunities for further research in this area. On the one hand, many algorithms and protocols in the area of mobile ad-hoc networks have completely ignored the problem of dynamic node presence. It would be interesting to investigate the impact that a truly dynamic network would have on those approaches and how a presence detection system can improve the situation. On the other hand, for some applications it may be possible to compress presence information even further using a more aggressive aggregation strategy.

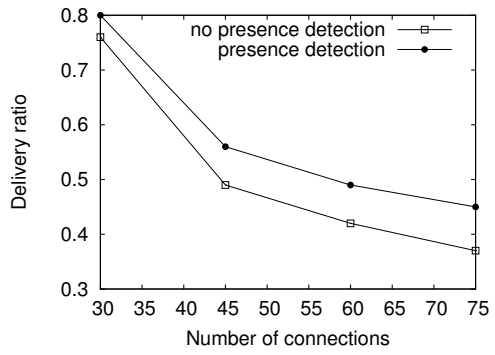


Figure 9: Delivery ratio with arriving and leaving nodes.

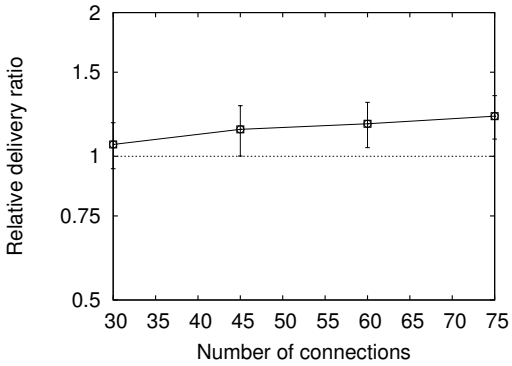


Figure 10: Relative performance with arriving and leaving nodes.

7. REFERENCES

[1] B. H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, 3(7):422–426, July 1970.

[2] K. Cheng, M. Iwaihara, L. Xiang, and K. Ushijima. Efficient Web Profiling by Time-Decaying Bloom Filters. *Database Society of Japan Letters*, 4(1):137–140, June 2005.

[3] K. Cheng, L. Xiang, M. Iwaihara, H. Xu, and M. M. Mohania. Time-Decaying Bloom Filters for Data Streams with Skewed Distributions. In *RIDE-SDMA '05: Proceedings of the 15th International Workshop on Research Issues in Data Engineering: Stream Data Mining and Applications*, pages 63–69, Apr. 2005.

[4] N. Eagle and A. Pentland. Social Serendipity: Mobilizing Social Software. *IEEE Pervasive Computing*, 4(2):28–34, Apr. 2005.

[5] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. *IEEE/ACM Transactions on Networking*, 8(3):281–293, June 2000.

[6] R. Gilbert, K. Johnson, S. Wu, B. Y. Zhao, and H. Zheng. Location Independent Compact Routing for Wireless Networks. In *MobiShare '06: Proceedings of the 1st International Workshop on Decentralized Resource Sharing in Mobile Computing and Networking*, pages 57–59, July 2006.

[7] S. Giordano and M. Hamdi. Mobility Management: The Virtual Home Region. Technical Report SSC/1999/037, EPFL-ICA, Lausanne, Switzerland, Oct. 1999.

[8] P. Gupta and P. R. Kumar. The Capacity of Wireless Networks. *IEEE Transactions on Information Theory*, 46(2):388–404, Mar. 2000.

[9] L. E. Holmquist, J. Falk, and J. Wigström. Supporting Group Collaboration with Interpersonal Awareness Devices. *Springer Personal and Ubiquitous Computing*, 3(1/2):13–21, Mar. 1999.

[10] Y. Iwatani. Love: Japanese Style. Wired News, online, http://www.wired.com/news/culture/0,1284,12899,00.html, June 1998.

[11] P. Jacquet, P. Mühlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized Link State Routing Protocol. In *INMIC '01: Proceedings of the 5th IEEE International Multi Topic Conference*, pages 62–68, Dec. 2001.

[12] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In T. Imielinski and H. F. Korth, editors, *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, Norwell, MA, USA, Jan. 1996.

[13] W. Kiess, H. Füßler, J. Widmer, and M. Mauve. Hierarchical Location Service for Mobile Ad-Hoc Networks. *ACM Mobile Computing and Communications Review*, 8(4):47–58, Oct. 2004.

[14] C. Lee, S. Yoon, E. Kim, and A. Helal. An Efficient Service Propagation Scheme for Large-Scale MANETs. In *MPAC '06: Proceedings of the 4th International Workshop on Middleware for Pervasive and Ad-Hoc Computing*, pages 9–12, Nov. 2006.

[15] C. Lindemann and O. P. Waldhorst. Effective Dissemination of Presence Information in Highly Partitioned Mobile Ad Hoc Networks. In *SECON '06: Proceedings of the 3rd IEEE ComSoc Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, Sept. 2006.

[16] R. Morris, J. Jannotti, F. Kaashoek, J. Li, and D. S. J. DeCouto. CarNet: A Scalable Ad Hoc Wireless Network System. In *Proceedings of the 9th ACM SIGOPS European Workshop*, pages 61–65, Sept. 2000.

[17] The Network Simulator ns-2, version 2.30. Online, http://www.isi.edu/nsnam/ns/.

[18] C. E. Perkins and E. M. Royer. Ad-hoc On-Demand Distance Vector Routing. In *WMCSA '99: Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, Feb. 1999.

[19] S. C. Rhea and J. Kubiatowicz. Probabilistic Location and Routing. In *INFOCOM '02: Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1248–1257, June 2002.

[20] I. Stojmenovic. Home Agent Based Location Update and Destination Search Schemes in Ad Hoc Wireless Networks. Technical Report TR-99-10, University of Ottawa, Sept. 1999.

[21] M. Terry, E. D. Mynatt, K. Ryall, and D. Leigh. Social Net: Using Patterns of Physical Proximity Over Time to Infer Shared Interests. In *CHI '02: Extended Abstracts on Human Factors in Computing Systems*, pages 816–817, Apr. 2002.