Lightweight Detection of Node Presence in MANETs

Thi Minh Chau Tran [1], Björn Scheuermann, Martin Mauve

Heinrich Heine University Düsseldorf
Universitätsstr. 1, D-40225 Düsseldorf, Germany
Phone: +49 211 81-11634, Fax: +49 211 81-11638
{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. It uses a Bloom filter-based beaconing mechanism to aggregate and distribute information about the presence of network nodes. We describe the algorithm and discuss design alternatives. We assess the algorithm's properties both analytically and through simulation, and thereby underline the effectiveness and applicability of our approach.

Key words: 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

[1] Corresponding author.

impact of varying node presence has not yet been systematically studied, although it can affect the performance of a network significantly.

One example—by far not the only one, but a particularly good one—is reactive routing in MANETs with protocols like DSR [1] or AODV [2]. 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. 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.

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 [3]. 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.

The cost of aggregating presence information is a small number of so-called false positives, where nodes are wrongly considered to be present. Note that 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 three major contributions. First, it points out the problem of lightweight presence detection as an important open research problem in the con-

text of MANETs. Furthermore, it describes a solution to the problem. Finally, it assesses the performance of this solution in various regards, both analytically and by means of simulation.

Beyond these core contributions, a side effect of the mechanisms developed in this paper allows nodes to estimate their distance to other nodes. As shown by Gupta and Kumar [4], 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.

The remainder of this paper is structured as follows. In Section 2, we summarize related work. Section 3 presents the algorithm we propose for lightweight presence detection, as well as optional enhancements to the algorithm. Section 4 evaluates our approach both analytically and by means of ns-2 simulations in terms of false positive rates, the accuracy of the distance estimates, and the speed of information propagation. 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.

As already mentioned, routing protocols for mobile ad-hoc networks are able to determine the presence status of a node. However, reactive protocols like AODV [2] or DSR [1] 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 [5] can determine the presence of a node directly. 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 [6, 7], the grid location service (GLS) [8], and the hierarchical location service (HLS) [9]. 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 approaches for presence detection in wireless systems have been successful applications on their own, for example the Lovegety [10]. A number of research projects deal with the exchange of presence information via single-hop wireless communication, with or without infrastructure. Examples are [11–13]. But none of these systems considers presence detection over multiple wireless hops. The same can be said about protocols that are able to discover the one- or two-hop neighborhood. NHDP [14] is an example of such a protocol.

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 data distributed to each node would increase linearly with the number of nodes in the network, which increases the network load accordingly. 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.

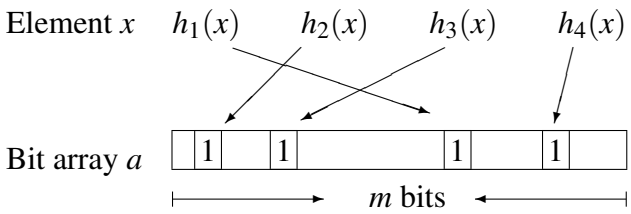


Figure 1. A Bloom filter.

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 [3], 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 $\{1, \ldots, m\}$. Any hash function with good random distribution and outputs long enough for the Bloom filter size can be selected.

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, as depicted in Figure 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

disseminate 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. Since beacons contain aggregated data and are sent only to neighboring nodes, presence information is disseminated without flooding the network. For such a scheme to work, the hash functions need to be agreed upon beforehand and then used by all nodes in the network.

However, in a presence detection system the removal of no longer present nodes is also necessary. The standard Bloom filter has the 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 [16]) 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 [17] and [18] 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 [19], 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 [20, 21] is used to continuously analyze a data stream like, e. g., web page hits. Like in counting Bloom filters [16], 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.

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

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

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. By means of analysis and simulation, we will later show that this "seen distance" is in fact very close to the real minimum hop distance.

Figure 2 shows the entries corresponding to a 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.

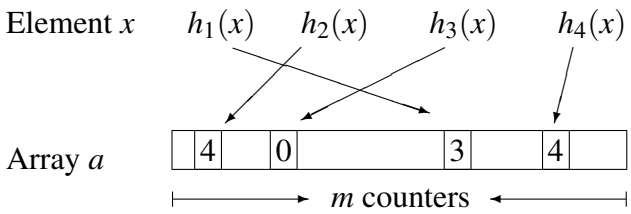


Figure 2. A soft state Bloom filter.

3.3 Decoupling Information Decay and Beaconing Intervals

In the presented algorithm, the "age" of presence information increases with each beacon interval. In some situations, it might prove beneficial to increase the flexibility by decoupling the beaconing rate and the speed of information decay. This is possible, but it requires a small modification to the proposed algorithm.

In order to understand why this is the case, consider a situation where the ageing of information is slowed down with respect to the beaconing in a naive way, by incrementing the counter values only every $b > 1$ beaconing intervals. Let A and B be two neighboring nodes; for simplicity, we neglect all other nodes and their beacons for the moment. Assume that some aggregate position a_i is currently at value 5 in both these nodes, but is no longer refreshed and should thus be subject to decay. A is first to age its counter values, and increments a_i to 6. But B's value of a_i is still 5, and hence A will very soon receive another beacon from B with $a_i = 5$, thus resetting A's aggregate. B will be next to age its own aggregate, but will in the very same way immediately be reset by A's next beacon. The reason for this misbehavior of the naive modification is that it may happen that a node repeats a counter value received from another node in its own broadcasted aggregate, without ageing it at least once.

This problem can be overcome by introducing the following small extension to the soft state Bloom filter based presence detection protocol: whenever a beacon is sent *without* previously ageing the counter values in the local aggregate, the sent-out *copy* of the aggregate must be decayed (and refreshed) prior to sending it to the neighbors. That is, if A's value of a_i is currently 5, and it sends a beacon without decaying its local copy of the aggregate, in its beacon it will broadcast the value $a_i = 6$. This ensures that a received counter value is never repeated un-decayed in own beacons, and hence avoids the described problem.

3.4 Aggregate Compression

Mitzenmacher showed that (standard) Bloom filters can be compressed to reduce their size [22]. In theory, these compressed Bloom filters are a space-optimal rep-

resentation of a set of items with a given false positive rate. On the one hand, this supports our choice of Bloom filters as the underlying data structure for our protocol. On the other hand, it points to an interesting additional option for further reducing the size of the presence detection beacons—or for achieving a lower false positive rate without increasing the size.

The counter values in soft state Bloom filters will typically not be equidistributed: not every counter value is equally likely. Therefore it is possible to save bandwidth by applying lossless compression to the beacons before they are transmitted. It turns out that arithmetic coding with an adaptive model as described by Witten et al. [23] is very well-suited for this particular task.

The basic idea behind arithmetic coding is to divide the interval $[0,1]$ first into sub-intervals according to the probability distribution of the possible values of the first character of the to-be-compressed input. For compressing soft state Bloom filters, the first character corresponds to the first Bloom filter position, i. e., to the first counter, which may take values in the range $0 \ldots 2^l - 1$. The sub-interval corresponding to the counter's value is then chosen. It is again subdivided, matching the probability distribution for the second entry, and so on. The sender generates a binary representation of a value within the finally resulting (tiny) interval. The problem that arises when this is to be applied in practice is that the probabilities of the input values need to be known by both the encoder and the decoder. It has been shown that, given a suitable probability distribution model, arithmetic coding is able to compress down virtually to the entropy limit.

Witten et al. use arithmetic coding with a dynamic model. In their scheme, the input probability distribution is "learned" while processing the input. Basically, the algorithm keeps track of the input distribution in the data processed so far, continuously adjusting the model. This is very well-suited for the problem at hand, because all entries of a soft state Bloom filter do indeed exhibit the same input probability distribution. So, what has been learned during the compression of the first part of the filter is indeed a very good model for the remaining part. Another big benefit in particular on resource constained hardware is that both encoder and decoder can be implemented very efficiently with integer arithmetic. Both require only a constant, very small amount of memory, and encoding and decoding complexity is linear in the input length.

We propose to apply arithmetic coding as a kind of filter in our presence detection protocol: it comes into action when beacons are assembled (then, the transmitted soft state Bloom filter is passed through the arithmetic coding encoder and is thereby significantly reduced in size), and when beacons are received from other nodes (in this case, the decoder restores the originally transmitted soft state Bloom filter).

Since the compression with arithmetic coding is lossless, this technique does not

have any direct impact on other aspects of the protocol itself or on which nodes are considered present or absent. Applying it comes at the cost of a slightly increased workload for the network nodes, for the compression and decompression of sent and received beacons; whether the achievable bandwidth savings outweigh this cost or not heavily depends on the particular devices and on the application scenario. However, as mentioned above, arithmetic coding can be implemented very efficiently even on resource-constrained devices, and it will thus often constitute an interesing extension.

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 five aspects: the reliability of the scheme in terms of the false positive rate, the accuracy of the seen distance, the effects of node movement, the speed of information propagation, and the effectiveness of compressing soft state Bloom filters with arithmetic coding.

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

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

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 turns out to be 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

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

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

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

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.

Note the implications of these effects: 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 means 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.

The results presented so far allow to estimate the false positive rate depending on the distance of some node and the number of t-hop neighbors. In practice, however, the latter will typically not be known, and it is hard to estimate. If, at run-time, a node intends to estimate the current false positive rate, however, this can also be done based on the node's local knowledge. Back to the initial point of our considerations, the probability of a false positive is the probability that all k filter entries corresponding to x are set by other nodes. Thus, given the status of the local aggregate, it is actually sufficient to know the probability that a randomly selected position is set.

Let $s(t)$ be the number of filter positions where the age stored in the counter value is less than t, i. e.,

$$s(t) := \left|\{i \mid 1 \leq i \leq m \text{ and } a_i < t\}\right|. \tag{6}$$

Then, the probability of a randomly selected position being set is $s(t)/m$, and consequently the probability of a false positive at distance t, which equals the probability of k randomly chosen positions being set, is simply given by

$$(s(t)/m)^k. \tag{7}$$

While estimating the same quantity, Formula 5 and Formula 7 are applicable to two different situations. The former calculates the expected false positive rates given a set of configuration parameters and thus can be used in tuning the approach's parameters to suit environment characteristics. On the other hand, using the latter, a node can rate the quality of specific positive answers from the presence detection service given the current status of its local aggregate. With such information, nodes can adapt the behavior of an application accordingly, depending on how likely a wrong answer is, and how big the potentially incurred overhead would be.

4.2 Accuracy of the Seen Distance

Let us now have a look at the relation between the seen distance and the true minimum hop distance between two nodes. That these two are related is intuitively clear: the soft state Bloom filter entries age by one over each hop, and hence the seen distance also increases by one every time the aggregate is handed over to the next node via a beacon. Due to the merging rule of the position-wise minimum, if information arrives over multiple paths, the lowest age (i. e., the lowest counter values, and hence the shortest path) will have precedence. However, there are effects that cause deviations between the seen distance at some point in time, and the true minimum hop distance.

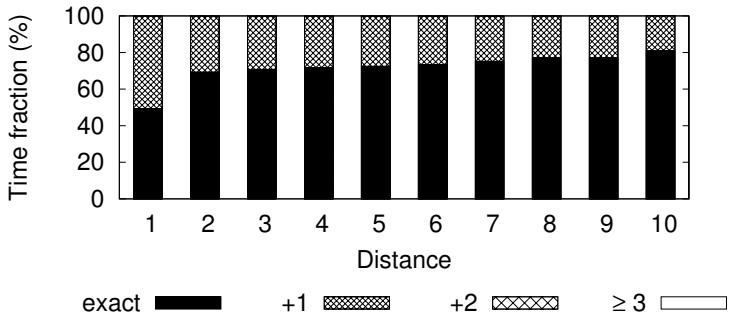
In the proposed presence detection algorithm, each node periodically increments the age of the Bloom filter positions in its aggregate before sending a beacon. While waiting for the next beaconing interval to expire, updates are received from the neighboring nodes, potentially resetting the incremented 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 occur always, even in the case of an ideal medium without losses. Higher oscillation amplitudes are caused by packet losses.

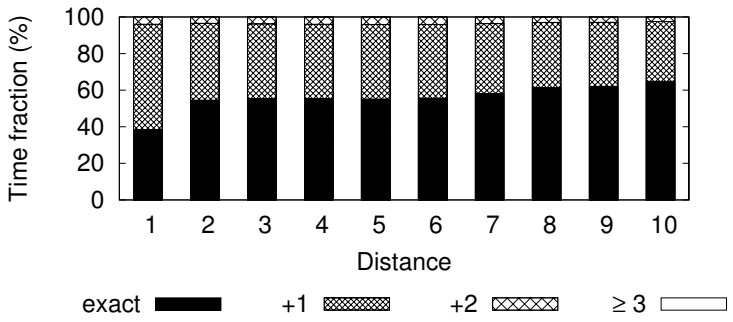
Simulations using the ns-2 network simulator [24] were carried out to see to which extent the error rates effect on the fluctuation of reported pair distances. We placed 200 nodes randomly on a square area of 1500 meters side length. These parameters make sufficiently sure that the network is fully connected; they are similar to the settings used in other work on wireless multihop networks such as, for instance, [25] and [26]. We used our presence detection algorithm with $m = 1024$ filter positions, $l = 4$ bits counter length, and a beaconing interval of three seconds. Ns-2's default transmission range of 250 meters was used. 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 3 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 3 (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.

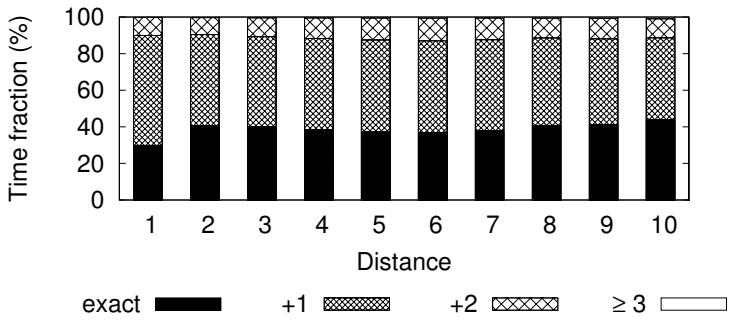
Whether the remaining oscillations are a problem essentially depends on the application's requirements. If an application requires higher stability of the seen distances, a simple way to achieve this is the following. In addition to the current local aggregate A, a node may store the aggregates from n previous broadcasting intervals, as they looked like just before incrementing the counters. When checking for the presence of node x, instead of using just the distance reported by the current aggregate, the minimum value from the current and all n previous aggregates is used. This technique yields very good results even and especially for small n, like $n = 2$ or $n = 3$. It comes at the cost of a slightly increased time until a no longer present node is considered absent, and until an actually increasing distance of a node is recognized. Both events will only be recognized after an additional delay of n beaconing intervals. This, however, will often be tolerable. The time until some newly arriving node is recognized as present does not change.



(a) Beacon error rate 0.



(b) Beacon error rate 0.25.



(c) Beacon error rate 0.5.

Figure 3. 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 "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

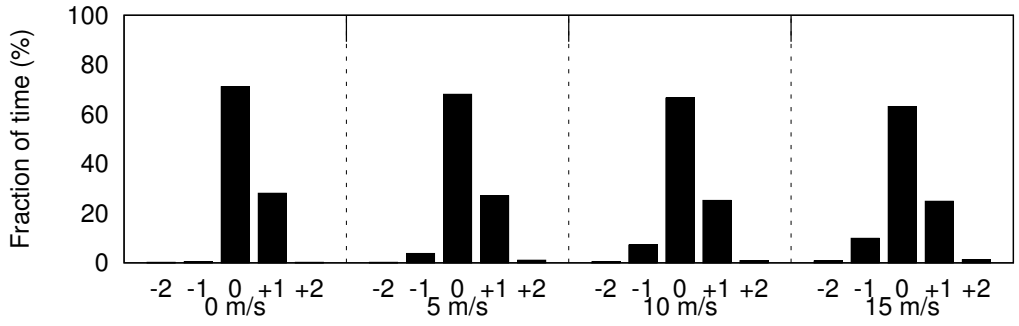


Figure 4. Deviations between seen distances and true shortest hop distances for different maximum node speeds.

to the modified Random Waypoint mobility model without pause times and with different maximum speeds. In order to overcome the well-known limitations [27], we have used the modified version of the Random Waypoint model, initialized with the steady-state distribution. The results given in Figure 4 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 node mobility is high, overestimated distances due to lost beacons 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.

4.4 Speed of Information Propagation

Our scheme uses periodic beaconing to distribute presence information. Thus, a naive assumption on the propagation speed could be that the information travels one hop per beaconing interval, i. e., a node at a distance of d hops from a newly arrived node will notice its presence after d broadcasting intervals. Further investigation, however, quickly shows that this approximation is far too pessimistic.

The key reason why the dissemination of presence information is much faster in practice is that the beaconing cycles of the nodes are not synchronized. Assuming independent offsets of the beaconing times, the time between the reception of information by some node and the next beaconing cycle of that node is only *half* a beaconing interval on average. Therefore, on average, the propagation of the information along a chain of nodes with length d hops takes only $d/2$ broadcasting

intervals. A dissemination delay of d broadcasting intervals is only the unlikely worst case.

In a topology that is more complex than a simple chain the results improve further: typically, there are many paths along which the information might propagate, and it is sufficient that it arrives along one of them. So, the time until a node x is recognized as present by some other node u is the minimum information propagation delay over all paths from x to u. Since the number of these paths quickly increases with increasing node density, it is reasonable to expect a propagation delay that is far below the worst case of d broadcasting intervals, and also less than the expected propagation delay along a single path of $d/2$ broadcasting intervals.

In order to illustrate this, we consider the situation of two nodes u and x that are two hops apart. We estimate the number of potential forwarders of the presence information between two such nodes. We assume a given node density $\rho > 0$ in the network area around the two nodes, and a circular one-hop neighborhood area of radius $r > 0$. Here, we sketch how an estimate of the delay of information propagation over two hops can be obtained. More details on the calculations can be found in the appendix.

Note that the distance d between two nodes that are two hops apart must be within $]r, 2r]$. It can be shown that the probability of having n nodes in the intersection of the radio ranges of two such nodes is

$$P(n \text{ nodes}) = \int_r^{2r} \frac{2\delta(\rho A(\delta))^n e^{-\rho A(\delta)}}{3r^2 n!}\, d\delta. \tag{8}$$

This can be used to calculate the probability of having n potential forwarders between two nodes that actually are two-hop neighbors (i. e., for which there exists at least one node that can directly communicate with both).

The expected time until a broadcasting interval expires at the first out of n potential forwarders is $B/(n+1)$, where B is the broadcasting interval length. Combining the results mentioned above (calculation details are given in Appendix A) yields an expected delay for the second hop's delay for a random pair of nodes u, x at two-hop distance of

$$\sum_{n=1}^{\infty} \frac{B \cdot \rho^n}{(n+1)!} \cdot \frac{\int_r^{2r} \delta(A(\delta))^n e^{-\rho A(\delta)}\, d\delta}{\frac{3}{2}r^2 - \int_r^{2r} \delta e^{-\rho A(\delta)}\, d\delta}. \tag{9}$$

In order to verify the predicted information dissemination speed, a simulation was carried out using the ns-2 network simulator [24]. 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

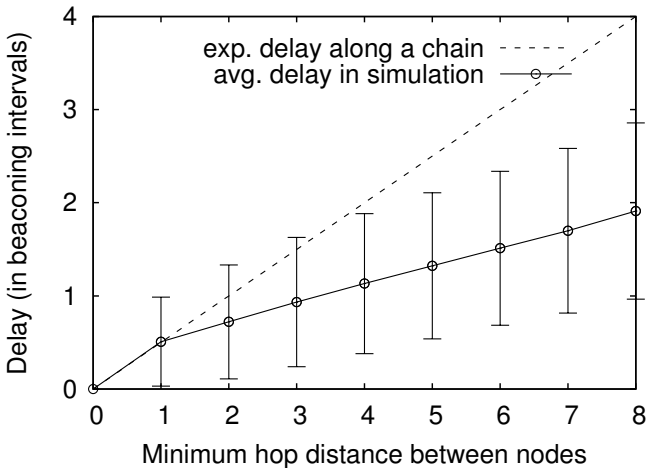


Figure 5. Average delay before a node is recognized as present.

interval of three seconds. The simulation uses IEEE 802.11 at 1 MBit/s. All local aggregates were initialized empty. Then, for each pair of nodes, the delay was measured until one node recognized the other one as present. Figure 5 shows the results of these simulations. The x-axis denotes the distance between the nodes, i. e., the minimum hop count. The y-axis then shows the average time until the presence information arrives, with 95-percentile error bars. The dashed line shows the expected dissemination speed along a single path.

It is evident that, as predicted by the theoretical arguments above, the dissemination speed is surprisingly high. For example, presence information travels over a distance of eight hops in less than two broadcasting intervals on average. This is because a high number of alternative paths are available, which increases the probability that for one of these paths the offsets of the nodes' periodic broadcasting are beneficially aligned, allowing for a quick information forwarding. For the first hop, there is only one single node which is able to forward the information, the source itself. At this point, the simulation results for the one-hop delay therefore match the delay predicted for a chain, $B/2$, exactly.

Evaluating (9) for the parameters of our simulation, i. e., for $\rho = 200/1500^2$ and $r = 250$, yields an expected delay for the second hop of 0.29 broadcasting intervals. In the simulation, the delay is even lower, around 0.23 broadcasting intervals. This is because it frequently happens that for a two-hop neighbor, an alternative path which is longer than two hops is even faster than all available two-hop paths. Such paths are not covered by (9).

So far, we have discussed the time until a node is considered present by other nodes after its arrival. The other interesting parameter is the time until it is no longer considered present after it has left. This, however, is straightforward: if the counters at the respective Bloom filter positions are no longer refreshed, they decay. Thus, after node x has left, it will be considered present by some other node u until the

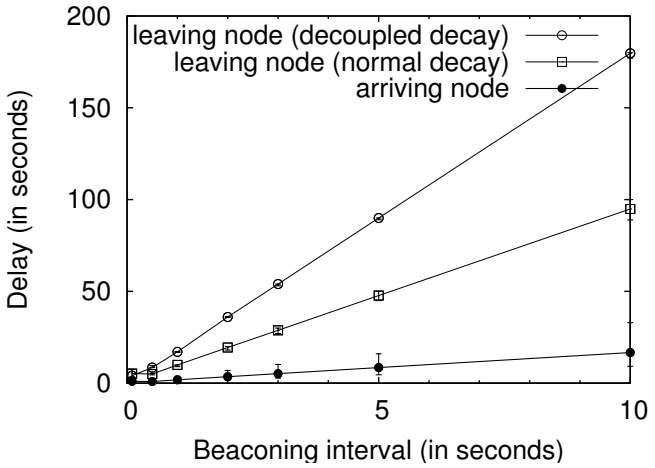


Figure 6. Time until information is received by all nodes.

counters exceed u's threshold T_u, which can be expected to happen after $T_u - t + 1$ more beaconing cycles, if x is currently seen at distance t. During that time, x will be seen at an increasing distance.

We conducted another set of simulations, with similar parameters as above, in order to underline our results. We chose a distance threshold of 10. In these simulations, nodes enter and leave the network. Figure 6 shows the time until a node is considered present by all other nodes within 10-hop distance after it has arrived, and the time until it is recognized by all as no longer present when it leaves. It also shows the time until all other nodes detect that a node has left if the beconing interval and the decaying speed are decoupled, as discussed in Section 3.3. Specifically, the decaying is slowed down by a factor of two in these simulations, in relation to the beaconing interval. The times are shown for different beaconing intervals with 99-percentile error bars. It can be seen that, as expected, both delays increase linearly with the beaconing interval length, and the delay until a no longer present node vanishes from the aggregate is generally longer than the delay until a newly arriving node is seen by the other nodes. As expected, the version with decoupled decaying exhibits a correspondingly longer delay until leaving nodes vanish from the aggregates: when aggregates are decayed only every other interval, the delay doubles.

The main lesson which one can learn from these results is that the propagation speed is significantly higher than what one would intuitively expect given a certain beaconing frequency. It is worth noting that these results hold for any information dissemination scheme that uses periodic broadcasting.

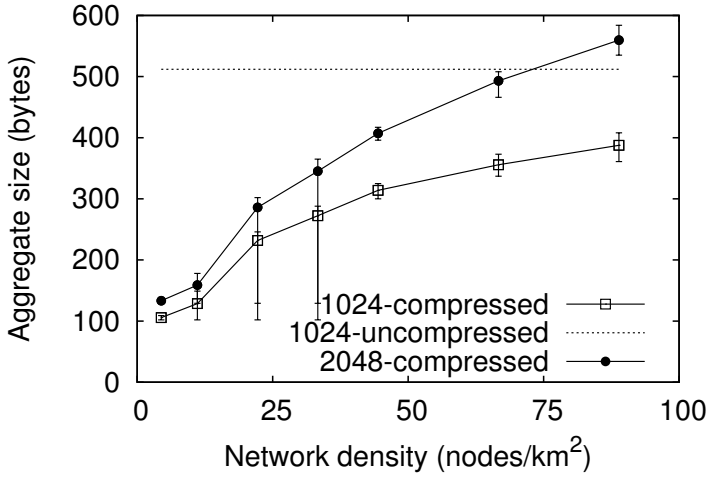


Figure 7. Average beacon size when compressed.

4.5 Effectiveness of Arithmetic Coding Compression

In Section 3.4 we have introduced a mechanism to reduce the size of the beacons by piping the transmitted and received soft state Bloom filters through an arithmetic coding encoder or decoder, respectively. Now, we assess how effective this mechanism turns out to be in practice. For this purpose, we have implemented arithmetic coding based compression and applied it to the aggregates occurring in the presence detection service in different simulated scenarios.

We found that the size of the beacons can—depending on the density of the network—typically be reduced by 30–80 %, as shown in Figure 7. The figure shows—for a varying node density—the size of compressed 2048-entry filters, compressed 1024-entry filters, and uncompressed 1024-entry filters (represented by the dotted line). It can be seen in Figure 7 that compressed soft state Bloom filters with 2048 entries could under most circumstances easily replace uncompressed 1024-entry soft state Bloom filters without requiring more bandwidth. For practical applications, this implies that, with compression, we could reduce false positive rate by using Bloom filters of larger size at no additional bandwidth cost.

4.6 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 m and k, which are directly related to the Bloom filter, 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 a 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

$$k = \frac{m \cdot \ln 2}{n(t)}. \tag{10}$$

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—considering today's wireless hardware—actually quite limited and therefore particularly challenging for a proactive, beacon-based protocol). 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, (10) 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 %.

As we have pointed out in Section 4.1, the false positive rate also quickly decreases with a smaller search radius: 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.

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: reactive routing. 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 [2]. 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 is delayed until the presence detection service indicates that the destination node is present.

We want to stress that this "model application" provides a hint on the possible effects of presence detection for a real application. It therefore complements the application-independent evaluation results from the previous section. The results shown here may not be arbitrarily generalized: the degree to which other applications benefit from presence detection depends on their cost associated with not having information on the presence of nodes and whether this outweighs the cost of the presence detection service.

The simulation study was conducted using ns-2 [24], with setups similar to those used before: 200 nodes 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. Like above, the nodes move according to the modified Random Waypoint mobility model [27]. The random speeds are in the range from 1 to 10 meters per second and the pause time is 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. All plots in this section show 95 % confidence intervals.

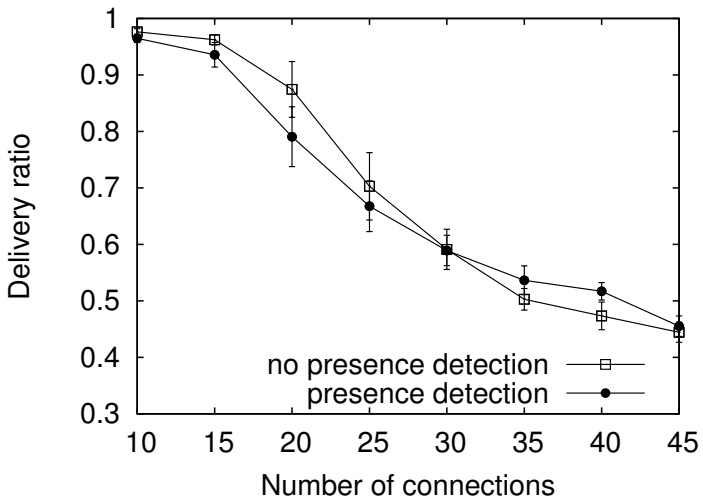


Figure 8. Worst case network performance.

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 8 shows the network performance, in terms of the packet delivery ratio, for an increasing number of connections. The negative impact of the presence detection is quite low, and sometimes there is actually a slightly better performance with presence detection.

The reason for these unexpected performance benefit 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 are 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 9 shows the effects on the network performance, again with 95% confidence intervals, when the number of additional connection attempts to non-present nodes increases. It can be seen that, without presence detection, more

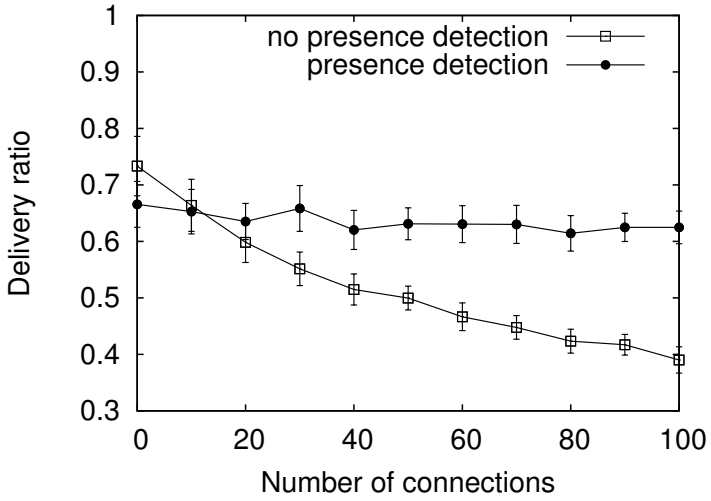


Figure 9. Packet delivery ratio as the number of connection attempts to non-present nodes increases.

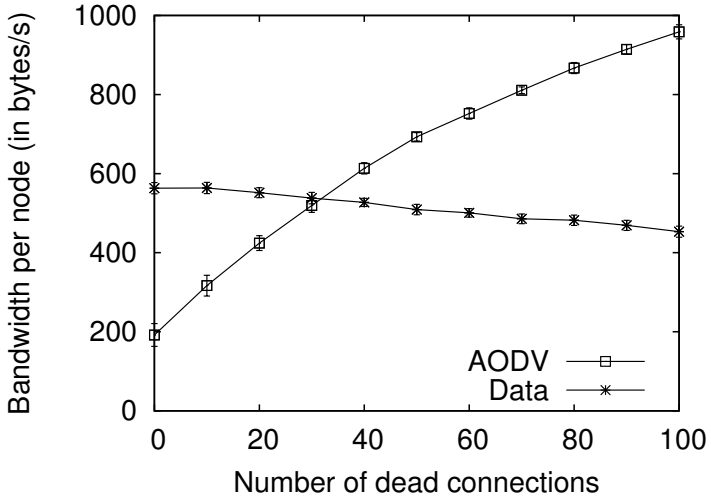
connection attempts to non-present nodes severely deteriorate the network performance. The performance with presence detection, on the other hand, does not show significant negative effects, no matter how many connections to non-present nodes are attempted.

Figure 10 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. The beaconing load in the case with presence detection 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.

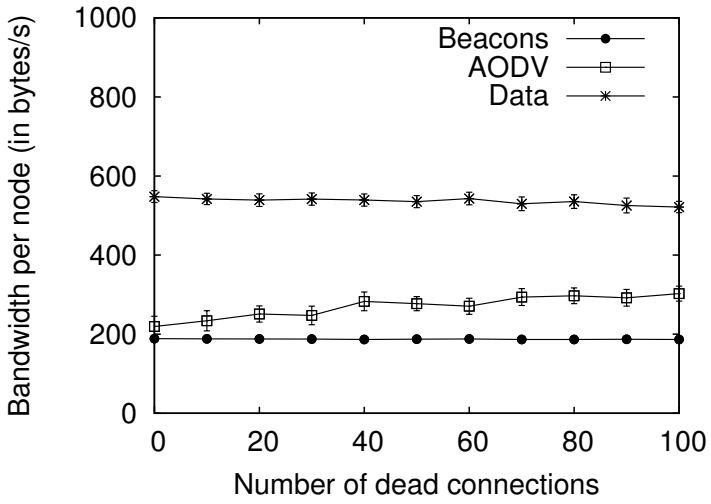
The situation with presence detection services running is much better. Since the presence detection service blocks the vast majority of connection attempts to non-present nodes, the bandwidth usage does not change significantly. There is only a minor increase in bandwidth used by AODV, which is due to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

We also carried out the same simulation but using compressed aggregates for presence detection service. Figure 11 shows a small gain of network performance in terms of delivery ratio compared with the case when aggregates are not compressed. The gain can be explained by smaller bandwidth usage in the case of compressed Bloom filter as depicted in Figure 12.



(a) Without presence detection.



(b) With presence detection.

Figure 10. Bandwidth use per node as the number of connection attempts to non-present nodes increases.

5.3 Arriving and Leaving Nodes

In this 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 13 it can once again be seen that the presence detection service helps to achieve a substantially better network performance.

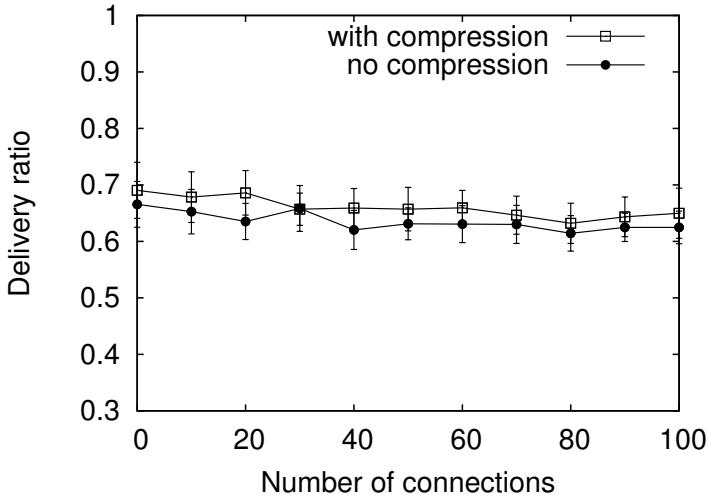


Figure 11. Delivery ratio in the case with compressed and uncompressed Bloom filters.

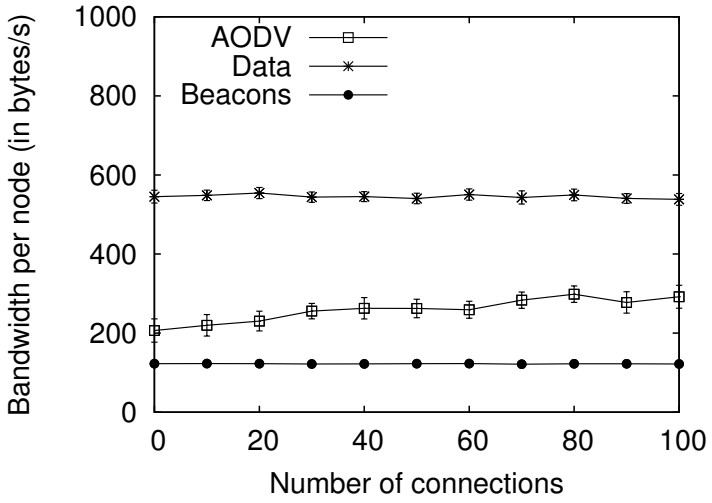


Figure 12. Bandwidth in the case with compressed Bloom filters.

6 Conclusion

In this paper we investigated the problem of detecting whether specific nodes are present in a mobile ad-hoc network. We believe that presence detection is an important 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.

We developed a scalable solution to the presence detection problem, which is also able to estimate the number of hops required to reach a given node. A soft state variant of the well-known Bloom filter allows for an efficient aggregation of presence information. In particular, we extended Bloom filters to support soft state decay and refresh operations. Our algorithm aggregates presence information using these soft

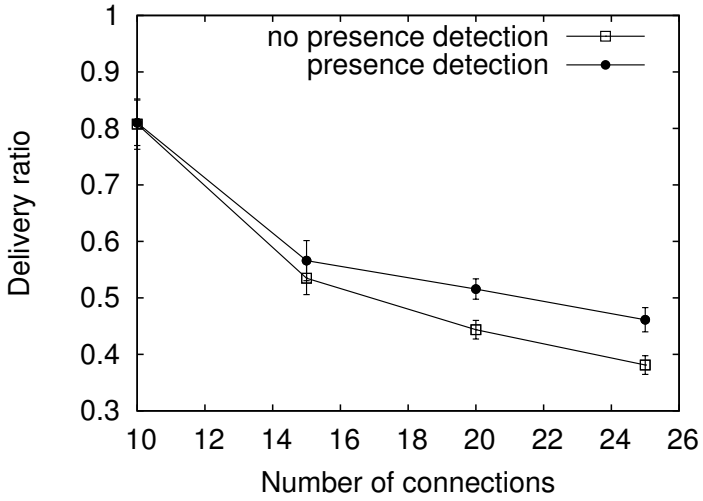


Figure 13. Delivery ratio with arriving and leaving nodes.

state Bloom filters. The aggregation comes at the cost of an adjustable amount of false positives, while it guarantees the absence of false negatives.

We investigated several key aspects of this approach, such as the speed of information propagation, the probability of false positives, and the bandwidth consumption by means of analysis and simulation. To underline the practical benefits that can be obtained by using presence detection, we also presented simulation results where presence detection was applied to the route discovery process of AODV. We were able to avoid issueing route requests to non-present destination nodes. We showed that the additional network load for the presence detection beacons is very limited and that substantial performance gains are possible if spurious flooding of the network with unsuccessful route discovery attempts can be avoided.

Acknowledgments

The authors would like to thank the German Research Foundation (DFG) for funding this research.

References

[1] D. B. Johnson, D. A. Maltz, Dynamic Source Routing in Ad Hoc Wireless Networks, in: T. Imielinski, H. F. Korth (Eds.), Mobile Computing, Kluwer Academic Publishers, Norwell, MA, USA, 1996, pp. 153–181.

[2] C. E. Perkins, 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, 1999, pp. 90–100.

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

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

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

[6] I. Stojmenovic, Home Agent Based Location Update and Destination Search Schemes in Ad Hoc Wireless Networks, Tech. Rep. TR-99-10, University of Ottawa (Sep. 1999).

[7] S. Giordano, M. Hamdi, Mobility Management: The Virtual Home Region, Tech. Rep. SSC/1999/037, EPFL-ICA, Lausanne, Switzerland (Oct. 1999).

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

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

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

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

[12] M. Terry, E. D. Mynatt, K. Ryall, 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, 2002, pp. 816–817.

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

[14] T. Clausen, C. Dearlove, J. Dean, Manet neighborhood discovery protocol (NHDP), draft-ietf-manet-nhdp-07 (2008).

[15] C. Lindemann, 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, 2006.

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

[17] S. C. Rhea, J. Kubiatowicz, Probabilistic Location and Routing, in: INFOCOM '02: Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies, Vol. 3, 2002, pp. 1248–1257.

[18] R. Gilbert, K. Johnson, S. Wu, B. Y. Zhao, 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, 2006, pp. 57–59.

[19] C. Lee, S. Yoon, E. Kim, 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, 2006, pp. 9–12.

[20] K. Cheng, L. Xiang, M. Iwaihara, H. Xu, 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, 2005, pp. 63–69.

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

[22] M. Mitzenmacher, Compressed bloom filters, IEEE ACM Transaction on Networking 10 (2002) 604–612.

[23] I. H. Witten, R. M. Neal, J. G. Cleary, Arithmetic coding for data compression, Commun. ACM 30 (6) (1987) 520–540.

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

[25] B. Karp, H. T. Kung, GPSR: Greedy perimeter stateless routing for wireless networks, in: MobiCom '00: Proceedings of the 6th Annual ACM International Conference on Mobile Computing and Networking, 2000, pp. 243–254.

[26] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, M. Gerla, The impact of multihop wireless channel on TCP throughput and loss, in: INFOCOM '03: Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies, Vol. 3, 2003, pp. 1744–1753.

[27] J. Yoon, M. Liu, B. Noble, Random waypoint considered harmful, in: INFOCOM '03: Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies, 2003.

A Propagation Delay Calculations

In this appendix, we detail the calculations on the propagation delay of the presence information over two hops. We assume a constant network density $\rho > 0$ with randomly placed, equidistributed nodes, and a fixed communication radius $r > 0$. Let u and x be two randomly placed nodes. Under the condition that their distance δ is in $]r, 2r]$—a necessary condition for u and x being two hops apart—, it follows from basic geometric probability calculations that the probability density function of their distance is

$$f(\delta \mid r < \delta \leq 2r) = \frac{2}{3r^2}\delta. \tag{A.1}$$

The area in which potential forwarders are located is the lens-shaped intersection of the one-hop neighborhoods of u and x. The size of this area is

$$A(\delta) = 2r^2 \arccos\frac{\delta}{2r} - \delta\sqrt{r^2 - \frac{\delta^2}{4}}. \tag{A.2}$$

The number of nodes in this area is Poisson distributed. Thus, we obtain the probability of n nodes being in the intersection of the one-hop neighborhoods of u and x, if their distance is in $]r,2r]$:

$$P(n \text{ forwarders}) = \int_r^{2r} \frac{2\delta(\rho A(\delta))^n e^{-\rho A(\delta)}}{3r^2 n!}\, d\delta. \tag{A.3}$$

Two nodes with distance in $]r,2r]$ are two-hop neighbors if and only if there is at least one node in the intersection area of their one-hop neighborhoods. Then the probability of n potential forwarders, $n > 0$, is

$$P(n \text{ forwarders} \mid 2 \text{ hops}) = \frac{P(n \text{ forwarders})}{1 - P(0 \text{ forwarders})}. \tag{A.4}$$

We now turn towards the expected time until the first forwarder's broadcasting intervals expires, given that the offsets are independent. The time until the first forwarder broadcasts can be modelled as the minimum of n pairwise independent random variables which are all equidistributed in $[0,B[$, for beaconing interval B. The probability density of the minimum (in $[0,B[$) is

$$f_n(t) = \frac{n}{B}\left(1 - \frac{t}{B}\right)^{n-1}. \tag{A.5}$$

This can be shown by induction over n, exploiting the fact that $\min\{X_1,\ldots,X_{n+1}\} = \min\{\min\{X_1,\ldots,X_n\},X_{n+1}\}$. Then the expected time until the first node sends a beacon is

$$\int_0^B \tau\frac{n}{B}\left(1 - \frac{\tau}{B}\right)^{n-1} d\tau = \frac{B}{n+1}. \tag{A.6}$$

By combining (A.4) and (A.6) it results that the expected time for the second hop delay for any node pair u, x with two hops distance is

$$\begin{aligned}
&\sum_{n=1}^{\infty} \frac{B}{n+1} \cdot \frac{P(n \text{ nodes})}{1-P(0 \text{ nodes})} \\
&= \sum_{n=1}^{\infty} \frac{B}{n+1} \cdot \frac{\int_r^{2r} \frac{2\delta(\rho A(\delta))^n e^{-\rho A(\delta)}}{3r^2 n!}\, d\delta}{1-\int_r^{2r} \frac{2\delta e^{-\rho A(\delta)}}{3r^2}\, d\delta} \\
&= \sum_{n=1}^{\infty} \frac{B \cdot \rho^n}{(n+1)!} \cdot \frac{\int_r^{2r} \delta (A(\delta))^n e^{-\rho A(\delta)}\, d\delta}{\frac{3}{2}r^2 - \int_r^{2r} \delta e^{-\rho A(\delta)}\, d\delta}.
\end{aligned} \tag{A.7}$$