The Effect of Mobility on Local Service Discovery in the Ahoy Ad-Hoc Network System

Patrick Goering, Geert Heijenk, Boudewijn Haverkort, and Robbert Haarman

Faculty of EEMCS / DACS
University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands
patrick.goering@utwente.nl*, geert.heijenk@utwente.nl,
brh@ewi.utwente.nl and rhaarman@inglorion.net

Abstract. Ahoy, a protocol to perform local service discovery in ad-hoc networks is described in this paper. The protocol has been implemented in a discrete-event simulator to study its performance in case of a multihop mobile ad-hoc network. Especially the effect of mobility on the network load and the probability of finding services is investigated. Experiments show that the load caused by advertisement messages is very low, even when the mobility is increasing. For low speeds the percentage of found services is close to the maximum possible, while even at high speeds the probability of finding a service is still reasonable.

1 Introduction

Ad-hoc networks are used to enable wireless communication between mobile nodes without making use of any infrastructure. Users of these networks want to interconnect their devices, make use of services provided by other devices, and have the possibility to offer their own services to other devices. In many cases it is useful to be able to find the nearest available service, also called service discovery, such as a printer or scanner. In such an ad-hoc network environment we want to be able to find services that are located nearby, while keeping in mind the limited power and network capacity.

In [1] we described a simple local service discovery protocol and presented experiments in a static situation, without node mobility. In this paper, we introduce a keep_alive mechanism to the protocol to save bandwidth. We use keep_alive messages instead of larger advertisement messages where possible. We implemented this new protocol, now named Ahoy, in a discrete-event simulator. Furthermore, in this paper we study the effect of mobility, the amount of bandwidth it takes to keep information about local services up-to-date, and the effect on the success probability of queries; this has not been done before.

The organization of this paper is as follows. Section 2 describes other research related to service discovery in ad-hoc networks. Section 3 discusses our service discovery solution using attenuated Bloom filters; the hash functions we use, the effect of mobility on the protocol, and the impact of false positives. Section 4 describes the simulation setup and Sect. 5 gives simulation results of the protocol when there is mobility. Finally, Sect. 6 presents the conclusions and future work.

* This work is supported by the Dutch Ministry of Economic Affairs under the Innovation Oriented Research Program (IOP GenCom, QoS for Personal networks at Home).

2 Related Work

Several service discovery protocols have been developed for computer networks. We look at the suitability of some of them for local service discovery in a mobile ad-hoc network (MANET). One distinction we can make is between centralized and distributed solutions. An example of a centralized service discovery system, with a directory server that stores all available services is SLP [2]. All devices in the network have to communicate with this directory server, which is a disadvantage for mobile ad-hoc networks, as there is limited network capacity and this server might not always be reachable.

A distributed solution has some advantages in a mobile ad-hoc network [3]; it can be proactive or reactive. A proactive solution forwards advertisements of available services to all nodes when there are any changes, whereas in a reactive solution query messages are forwarded through the network at the time a service is needed. A proactive solution has the advantage of services being available quicker at the cost of some bandwidth. An example of such an approach is Zeroconf [4], e.g., implemented as Apple Bonjour, an IETF protocol that enables the discovery of services on a local area network. A usable IP network is automatically created without the need for configuration or special servers, but it is limited to a single subnet. Bonjour does allow service discovery outside a single subnet, but this requires special DNS configuration and a connection to an infrastructure network. Thus in a multihop ad-hoc network Apple Bonjour cannot be used for local service discovery. In [5] the newscast epidemic protocol is used to provide a robust overlay network that adapts to (large) changes in a dynamic network. It uses quite some bandwidth to accomplish this, which makes it not suitable for wireless networks. Another approach was taken with HESED [6], where query messages are multicasted to all nodes. Selective edge nodes are used to reduce the number of multicast packets. Matching servers multicast their information to all nodes as well. Clients cache this information and use it to reduce the number of query messages.

For service discovery in ad-hoc networks, where we want to discover services located nearby, we need a fully distributed solution, suitable for multihop networks. Many nodes in ad-hoc networks will be mobile with a wireless network interface. Furthermore, such an approach should work as soon as a new node arrives, without the need to pre-establish a cluster or group. The Group-based Service Discovery Protocol (GSD) [7] is a distributed protocol for MANETs. In GSD, advertisements are limited to nodes within a maximum number of hops and services are grouped to allow selective forwarding of queries. The grouping of services is based on the semantics of the service descriptions and is predefined. Queries are forwarded to a node that has advertised services in the same group as the service might be available near this node. GSD does allow users to find a service within a maximum number of hops, but not necessarily the nearest service. Furthermore, groups need to be predefined and services classified in these groups to make use of selective forwarding of queries. Proximity Discovery Service (PDS) [8] also provides proximity based service discovery. This solution relies on the availability of the real geographic location by using a GPS satellite receiver, which we don't consider as a requirement for our protocol.

Attenuated Bloom filters have been used in [9] for context discovery. There, an analysis is done on the false positive probability and the size and depth of the used Bloom filters. Optimum values are found for several parameters to make efficient use of bandwidth. Here, we extend the service discovery protocol with a keep-alive mecha-

nism to minimize the bandwidth usage further. Also, the impact of mobility on service discovery with respect to bandwidth usage and reachability of services is evaluated.

The Ahoy protocol is a combination of a proactive and a reactive approach; a summary of available services is forwarded between neighbors upto a certain number of hops and queries are selectively forwarded using the information in the summary. The work presented in this paper further evaluates the idea of using attenuated Bloom filters for service discovery, especially in the presence of mobility.

3 The Ahoy Service Discovery Protocol

The Ahoy service discovery protocol is described in this section. Firstly, an overview is given of the protocol and the usage of Bloom filters. Secondly, the advertising and querying algorithms used in the protocol are explained. Then we discuss the hash functions used, the effect of mobility on the protocol, and the impact of false positives.

3.1 Overview

A Bloom filter [10] can be used to describe the membership of objects in a set, with a small chance of false positives. It consists of an array of w bits, initially all set to 0. A number of b independent hash functions over the range [1,w] is used to map a text string to the Bloom filter. A total number of b bits are set in the Bloom filter, one for each hash result. Some of the bits might be overlapping, that is, two different text strings might partially map to the same bits. A service, represented by a text string, is considered to be represented in a Bloom filter when all bits corresponding to the b hashes of this text string are set. For local service discovery in ad-hoc networks we propose to use attenuated Bloom filters [11]. They were introduced as a method to optimize the performance of location mechanisms especially when objects to be found are located nearby. We use Bloom filters in our service discovery protocol, because they can highly compress service availability information and thus reduce the bandwidth usage of the protocol.

An attenuated Bloom filter is a stack of standard Bloom filters of depth d. Every row in the filter represents objects at a different distance, indicated by the number of hops. Nodes maintain a separate attenuated Bloom filter for every link to a direct neighbor. This enables to select a link where an object most likely can be found, a so-called matching link. Periodically, an advertisement packet is broadcasted to all direct neighbors. The packet contains an attenuated Bloom filter, which represents the services reachable through the sending node. The attenuated Bloom filter is created by combining, a simple OR operation, all attenuated Bloom filters from all available links. This result is shifted down one layer and then combined with the node's own Bloom filter. See below for more details of the advertisement procedure.

When a client wants to find a specific service it will check whether the service is available locally. If this is not the case the client will check its attenuated Bloom filters and send a query packet to any link with a matching Bloom filter. A node receiving such a query again will check for local availability of the service, check its attenuated Bloom filters, and forward the query to any link with a matching Bloom filter. A node that does provide the service will send a response packet back along the path the queries

followed (in reverse order), as will be explained later. When the client receives the response packet it can call upon the service, although there is a possibility it does not exist.

3.2 Advertising

Algorithm 1 shows the actions taken by each node independently when packets related to advertisements arrive in a node.

```
1  switch received packet do
2      case keep_alive
3          if BC_ID != previous_BC_ID then
4              send (request_update packet to originating link);
5          update cleanup timer for neighbor;
6      case advertisement
7          if BC_ID != previous_BC_ID then
8              store received attenuated Bloom filter;
9              store BC_ID;
10             foreach layer do
11                 combine attenuated Bloom filters from links;
12             if advertisement packet != previous advertisement packet then
13                 send (advertisement packet to all links);
14         update cleanup timer for neighbor
15     case request_update packet
16         send (advertisement packet to originating link);
```

Algorithm 1: Advertisement (Run by Each Node Independently)

Advertisement messages include the attenuated Bloom filter as well as a broadcast identification field (BC_ID) that uniquely identifies the advertisement packets per neighbor. Keep_alive messages are broadcasted to all direct neighbors every *period* seconds. This *period* consists of a fixed part and a random part. The random part is added to desynchronize the keep_alive messages from all nodes. It also prevents peaks in bandwidth used by keep_alive messages. The keep_alive messages include the BC_ID value from the last sent advertisement packet. Because the size of the keep_alive messages is small compared to the size of the advertisements, the bandwidth usage is lower than in the case where we would send advertisement messages periodically. This also means we can send more keep_alive messages for the same network load and thus be able to detect changes in the network, like a new neighbor with new services, quicker. Sending one UDP packet in an ad-hoc network also involves MAC, IP and UDP header overhead, where part of the MAC header is transmitted at a lower speed. Thus the number of messages we can send more does not only depend on the time it takes to send advertisement and keep_alive messages of a specific size, but also on the time it takes to transmit the overhead. Receiving a keep_alive packet (line 2) with a specific BC_ID signifies all neighboring nodes that the services announced with the advertisement packet with the same BC_ID from the same neighbor are still valid. The only action a node takes when the BC_ID matches is postponing the clean up procedure as explained below. When

the BC_ID does not match (line 3), this signifies that the node has old information and should request an update of the available services reachable through the neighbor that sent the keep_alive packet. This situation could occur when an advertisement packet is lost, e.g., caused by interference in the radio link or by a new node moving into range. A request update packet is then sent directly to the neighbor, there is no need to broadcast this packet.

When an advertisement message is received (line 6) and the BC_ID differs from the BC_ID in the last advertisements from the same neighbor (line 7), the BC_ID and attenuated Bloom filter in the message are stored. All attenuated Bloom filters from all neighbors are combined (line 10). In case the newly constructed advertisement differs from the last advertisement sent (line 12), an advertisement message with the combined attenuated Bloom filter is broadcasted to all direct neighbors.

Upon receipt of a request update packet (line 15) a node will send an advertisement packet containing a full attenuated Bloom filter directly to the requesting neighbor.

A clean up procedure removes information of services reachable through a neighbor when this neighbor is out of reach of a node for a certain amount of time. A node is considered to be out of reach when the keep_alive messages are no longer received from this node. As packets can get lost in a wireless environment, a certain number of consecutive keep_alive messages should be allowed to be missed before the clean up procedure is started. After this number of keep_alive messages are missing, detected by not receiving a keep_alive packet for a number of keep_alive periods, the node will construct a new attenuated Bloom filter that represents the services reachable through this nodes at this specific time, i.e., without the services of the node that got out of reach. An advertisement packet containing this Bloom filter is transmitted to all neighbors when it contains information that is different from the previous advertisement.

Advertisements are not transmitted immediately after receiving new information from a neighbor for two reasons. Firstly, several nodes might receive an updated advertisement at the same time. Sending an advertisement immediately would result, with high probability, in collisions of the advertisement packets in the wireless network. Secondly, sending an update will likely trigger a neighbor to also send an updated advertisement. Randomization by adding a delay here allows to incorporate the information from this advertisement and will limit the number of messages per second nodes will send.

3.3 Querying

Algorithm 2 shows the query algorithm as it is run by each node independently.

Query messages contain a query identification (Q_ID), a maximum number of hops the message should be forwarded as well as a Bloom filter representing the queried service. The Q_ID together with the address of the originating node of the query is used to detect duplicates. When a node receives a query (line 2) with a Q_ID it has seen before, it can discard this query. In case the Q_ID is new (line 3), the maximum hops value is decremented by one. As the initial maximum hops value is d, up to a maximum of d hops away all services that match the query can be found. When the received query matches a Bloom filter previously received from any neighbor (line 7,8), the Q_ID value is stored together with the link the message was received from and the source address

```
1  switch received packet do
2      case query
3          if not originating node and Q_ID match previous query then
4              maximum_hops -= 1;
5              if service matches Bloom filter locally then
6                  send (response packet to originating link);
7              foreach link L do
8                  if service matches Bloom filter for link L upto maximum_hops then
9                      send (query packet to L);
10                     store Q_ID and link Q query was received from;
11     case response
12         if Q_ID matches previous query then
13             send (response packet to link Q);
```

Algorithm 2: Query (Run by Each Node Independently)

of the originating node. This information is used to send the response back along the path the query traveled. The query is then forwarded to all neighbors with a matching Bloom filter. The destination of the query, a node with a matching service, will send a response back to the node the query was received from first. Upon receipt of a response packet (line 11), a response packet is send to the neighbor the query was received from (line 12), as known from the previously stored Q_ID. All nodes repeat this process until the query arrives at the originating node.

3.4 Hash functions

For our protocol we need a number of hash functions, which distribute the bits set uniformly over the entire Bloom filter for the service being hashed. This is to make the probability of false positives as low as possible with a given Bloom filter width. In essence, we can tolerate some false positives, so this can be one of the criteria for the size of the Bloom filters being used. We use universal hashing [12]. These hash functions have the property that for any two distinct inputs the probability of a collision between those two inputs is the same as if we where using a uniform random function. In our service discovery protocol different nodes use hash functions to generate a number of bits to be set in a Bloom filter for announcing services as well as for querying for these services. Therefore, every node must use the same set of hash functions to be able to find services. The number of hash functions needed is determined by the number of bits that need to be set for every service that has to be represented in a Bloom filter. For the bandwidth usage of the protocol to be minimized, about half the bits need to be set in a Bloom filter [10]. In that case the false positive probability is low, while the width of the Bloom filters is not too high. The Bloom filters of d layers still have to fit in a single IP packet, to avoid the overhead of sending multiple IP packets for a single advertisement message.

3.5 False positives

False positives affect the service discovery as follows: a service can appear to be available through a specific neighbor due to a false positive. False positives can show up

as an effect of the combining of Bloom filters. When this happens in the lowest layer, only this node is affected, all other neighbors will not have the false positive. However, when it happens in layer *d-1*, all neighbors will have the same false positive in layer *d*, due to the way the advertisement procedure works. A query for such a service will be forwarded until it reaches a node where the false positive does not occur; the query along this path would then be silently dropped.

Generally, more nodes can be reached as the distance, and thus the maximum number of hops, increases. The lower layers of an attenuated Bloom filter will then contain more services, which means more bits are set. Thus, the probability of a false positive is higher in the lower layers than in the higher layers of an attenuated Bloom filter.

3.6 Mobility

In a situation, without node movement and without changes of services, results for queries are returned in the time it takes to forward and process the query messages for a maximum of *d* hops and sending back a response message. The bandwidth used by the protocol in this situation comes from sending keep_alive messages as well as the queries and responses themselves. A more challenging situation for a service discovery protocol appears when mobile nodes are moving around, so that there are changes in the services reachable. The frequency of the keep_alive messages determines how quick an update for a change in reachable neighbors and thus the update for the change in reachable services is propagated. When one node has a change in any of its services, potentially all nodes in a radius of *d* hops will exchange advertisement messages to notify all nodes in their range about the change.

4 Simulation Setup

The protocol described in the previous section has been implemented in the discrete-event simulator OPNET Modeler, version 11.5 [13]. The manet_station_adv model from the OPNET model library was used as a basis for our protocol implementation. In the following experiments mobile nodes are placed in a simulation area. The nodes have one wireless network interface that supports the IEEE 802.11b [14] standard for communicating with each other. The modified OPNET model was set to IEEE802.11b mode with a bitrate of 11 Mbps. Packets are always sent at 11 Mbps, there is no rate adaptation. For the moment the transmission range is limited to 300 meters. When a node is within a radius of 300 meters the free space path-loss propagation model, receiver sensitivity and transmission power are used to determine whether the transmission is successful or there is packet loss, i.e., packet transmissions may fail due to collisions or radio conditions. When a node moves out of this 300 meter radius of a sending node there is no interference with this sending node. We do not use a routing protocol in these experiments; from client to service the attenuated Bloom filters determine where packets are sent. The response messages from service back to the client follow the reverse path. Every simulation run is done with different seeds for the random number generator. The random number generator is used for all randomness in the standard OPNET models, as well as the random times in our model. Also both the advertised strings and the query strings are picked randomly using this random number generator,

unless stated otherwise. More specifically, every node advertises one service. We select 10 random ASCII characters as input for the hash function for every service and for every simulation run. The clean up period is set to 40 seconds and the keep_alive *period* is 15 seconds plus a random time, drawn every time a message is sent, uniformly distributed between 0 and 2 seconds. All protocol messages are sent over IP version 4 and UDP, thus for every message there is an overhead of an IP and a UDP header. The total message size further depends on the width and depth of the Bloom filter. Table 1 shows the values of the parameters used in the experiments.

5 Experiments

Four different experiments have been selected to study the behaviour of our protocol in varying degrees of mobility. The first two experiments examine a mostly static situation, where all nodes are fixed in a grid structure. One node moves through this grid at different speeds and we examine the effect of the advertisement delay parameter, as introduced in Sect. 3.2. Experiment 2 extends this experiment by investigating the effect the delay has on reachability of services. The last two experiments introduce more mobility; all nodes are moving randomly in the simulation area. Experiment 3 focuses on the advertisement load depending on the average speed of the nodes and the maximum Bloom filter depth d. Experiment 4 investigates the percentage of successful queries achieved by our protocol compared to the maximum possible as determined by the transmission range and the position of the nodes.

5.1 Limited mobility experiments

We consider the situation where our service discovery protocol is used in an ad-hoc network in which the nodes are relatively static. This scenario shows the effect of a single node traveling through an area, where many stationary ad-hoc nodes provide services. We position 61 nodes in a grid structure, which will stay stationary during the experiments. The grid structure was chosen for basic understanding of the interaction between nodes. Here one extra node (node_62) with a starting position outside the reach of any other node starts moving after 20 minutes simulation time. In these 20 minutes all other nodes learn each others services and then send keep_alive messages periodically. As the keep_alive messages for all nodes will be transmitted at random times, they will not be synchronized to each other. The mobile node moves with a constant speed from the bottom left to the top right in a straight line, as illustrated in Fig. 1. The spacing between all nodes is 300 meter, so that the mobile node will be in reachable distance of at least one node as long as it is in the center area (for about 2500 m).

The speed of this node is varied from 0 to 100 km/hr (27.8 m/s) in steps of 10 km/hr and for each speed 20 simulation runs are done. We let the simulation run long enough to allow the mobile node move through the entire network, until it is out of reach of any node again. The travel time depends on the actual speed of the node, the higher the speed the shorter the required simulation time. Node_62 starts learning about available services and the Bloom filters are updated as it moves through the simulation area. The depth of the attenuated Bloom filters d is 5. Note that since the radius of the network is also 5, services of the center node are propagated to the edge of the network.

Table 1. Parameters

parameter	exp1,2	exp3,4
advertisement delay	0-2 + unif[0,0.5] s	2 + unif[0,0.5] s
d	5	1-5
w	512 bits	
b	8	
period	15 s + unif[0,2] s	
range	300 m	

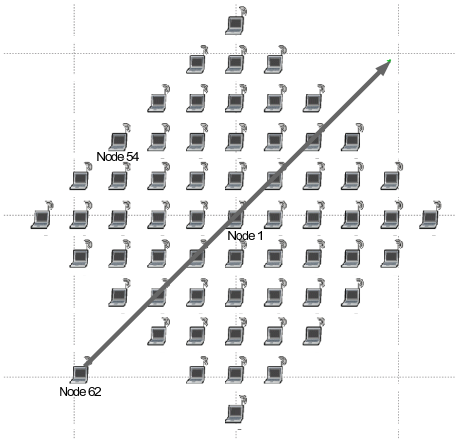

Fig. 1. Node movement

They would not propagate any further for a larger radius of the network. We selected the number of bits $b = 8$ and the Bloom filter width $w = 512$ bits. The size of one advertisement message, including MAC, IP, and UDP headers is then 384 bytes.

Experiment 1 In this experiment we measure the number of advertisement messages sent by the moving node (node_62), the center node (node_1), and the average for all nodes in the network. We do this while the moving node is within range of at least one of the static nodes of the network, i.e., we count all messages from the time the moving node comes within reach of the network until this node leaves the area where the other nodes are. The node is considered to be in range of one of the other nodes as long as it has neighbors for which it keeps an attenuated Bloom filter. After the last neighbor information has been removed through a clean up operation, we consider the node to be out of reach. We use the advertisement delay as a parameter for these experiments. The advertisement delay consists of a fixed and a random part. We always choose the random delay uniformly between 0 and 0.5 seconds, but vary the fixed delay. For different speeds the time we do the measurements will be different.

Figure 2 shows the number of advertisement messages per second per node averaged over all nodes during the simulation runs. We can observe that the advertisement load increases linearly with the speed of the mobile node. As the speed of the mobile node approaches zero, we see no advertisement messages at all. Nodes only send keep_alive messages to inform each other that there are no changes. As the speed of the mobile node increases, the number of advertisement messages per second increases as well, to keep all nodes up-to-date of the changed services. As the speed is increased to 100 km/hr (27.8 m/s) advertisement load increases to approximately 0.07 message per second. On average, nodes then use 215 bps ($\approx 0.002\%$ of 11 Mbps) for sending advertisement messages. As a reference: the AODV [15] routing protocol sends 608 bps of HELLO messages when a node is part of an active route. Note that in this graph the influence of the advertisement delay parameter is hardly visible.

When we look at the number of sent advertisements per second for two specific nodes some interesting phenomena can be observed. Both the mobile node (see Fig. 3) and the center node (see Fig. 4) exhibit the same linear increase of the advertisement load as the speed of the mobile node increases. In both figures the average number of advertisement messages from Fig. 2 is included for reference. However, the load of the center node, and especially of the mobile node increases much steeper than the load of an average node. At 100 km/hr (27.8 m/s) the load of the center node is almost twice the load of an average node, the load of the mobile node is approximately four times that value. This can be explained by the fact that the center node is on the trajectory of the mobile node, so most of the time, changes in reachability of services (from the mobile node) occur within d=5 hops. Of course the reachability of services from the mobile node changes continuously, so that the mobile node has an even higher advertisement load. Note however that the absolute value of the advertisement load is still very low.

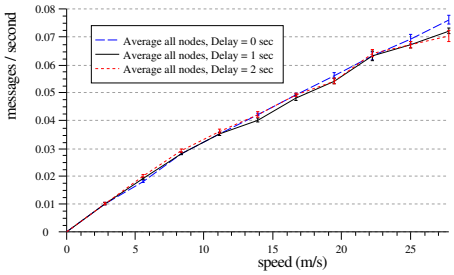

Fig. 2. Average number of advertisement messages from all nodes

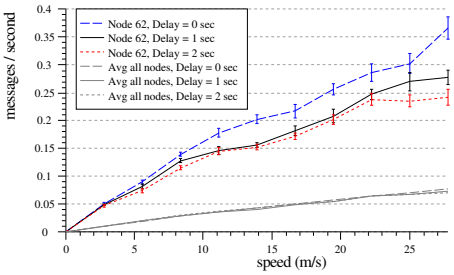

Fig. 3. Advertisement messages from node_62

Let us now consider the influence of the advertisement delay parameter on the advertisement load. From Fig. 2, it can be observed that the parameter hardly has any effect for an average node, although at higher speeds of the mobile node, some influence starts to be visible. For the mobile node itself, the impact of the advertisement delay is more pronounced (see Fig. 3). Even at moderate speed, significant savings in the advertisement load can be achieved by delaying advertisements for 1 or 2 seconds. For the center node (Fig. 4) the effect of the parameter is less straightforward. There appear to be nonlinear increases of the load for certain delay values, especially for the center node, but also for the mobile node. Below, we will give an explanation for these deviations.

When we look at the mobile node moving from the bottom left to the top right through the area, as shown in Fig. 5, we see that the mobile node will get in range of the nodes A and C as soon as it reaches the position of node D. From this moment on, nodes A and C can receive the keep_alive messages sent by the mobile node, and the mobile node can receive the keep_alive messages sent by nodes A and C. Keep_alive messages are sent periodically, so some time will elapse before either node A or C receives one from the mobile node, or the mobile node receives one from node A or C. Let us assume that the mobile node is the first node to receive a keep_alive message, and it receives it

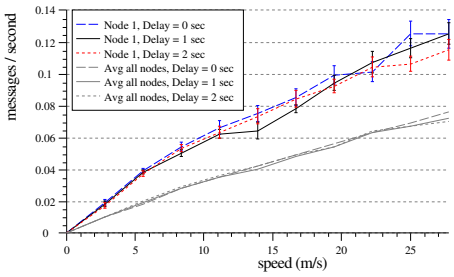

Fig. 4. Advertisement messages from node_1

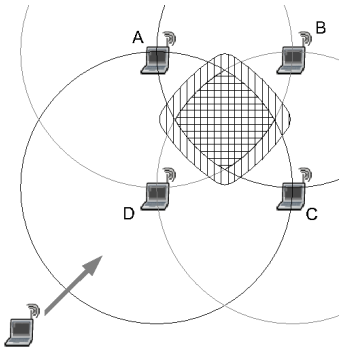

Fig. 5. Effect of delay

when it is exactly on the border of the outer shaded area. This will trigger the exchange of a request_update message and a subsequent unicast advertisement message, because the keep_alive message will refer to an unknown advertisement. Taking into account that this advertisement will contain as yet unknown services, its receipt will trigger the mobile node to broadcasting a new advertisement message. However, the mobile node will only broadcast the advertisement after waiting for the advertisement delay, during which the mobile node can travel to the border of the inner shaded area.

If the mobile node is within the inner shaded area when broadcasting its new advertisement to nodes A and C, the broadcast will also be received by node B, which will save the transmission of an extra broadcast message. It can be seen that this saving only occurs when the time of waiting for a keep_alive message from node A or C, plus the advertisement delay (plus the time for exchanging some messages) is sufficient for the mobile node to travel from the point where it gets in reach of nodes A and C to the point where it also gets in reach of node D. Note that a similar, yet slightly different effect occurs if either node A or C is the first node to receive a keep_alive message from the mobile node.

So, the speed of the mobile node influences the advertisement load in two different ways. If the speed increases, more changes in reachable services are detected, and hence more advertisements are sent in the same time period. On the other hand, if the speed increases, savings in the number of broadcasts to send can be made. The latter effect depends on the advertisement delay, among others, and is not continuous. Note that the same effect can also be observed for random topologies, although in a less deterministic way.

Experiment 2 For the same network as in the above experiment, we now want to know when a service can be found as long as the mobile node (node_62) is moving through the grid. We let node_54 try to find one of node_62's services. At the same time we also let the mobile node try to find a service advertised by the center node, node_1. Both node_1 and node_62 advertise a static, not randomly determined, service. When there is no match in any of the layers of the attenuated Bloom filter, nothing is done. When a match is found in one of the layers of the attenuated Bloom filter, a query is sent in that direction and forwarded as long as a match can be found and until it

reaches node_62. At some points the location of the target node is not where it should be according to the apparently outdated information in the Bloom filters. We want to find how often the service is still found and a reply is sent back to and received by the originating node. After 20 minutes, the time the mobile node starts moving, we start querying for the services. The time between query tries on both nodes is exponentially distributed with a mean value of one second. In the optimal case a service can be found when a network path of at most d hops exists between the client and the service. In our regular grid network we know when such a network path exists and thus we can normalise the number of successful queries to the maximum number of queries that could be successful in this scenario.

We see in Fig. 6 that at low speeds the service is almost always found, it approaches 100% when the speed is low. When there is no mobility the service will always be found as long as a path of at most d hops exists. For increasing speeds the percentage of found services decreases and is also more dependent on the delay parameter. We can also see nonlinear behaviour in the percentage of successful queries for different speeds, caused by the deviations in the number of advertisement messages found in the previous experiment. When for a certain speed there are less advertisement messages, this means there is less information of available services. On average the probability of success to find a service will be lower as well for this specific speed. When we look at the effect of the delay parameter, we see that for smaller delays the service is more often found, because all nodes in the path between client and service will know about changes quicker. This comes at the price of more bandwidth usage when changes are detected.

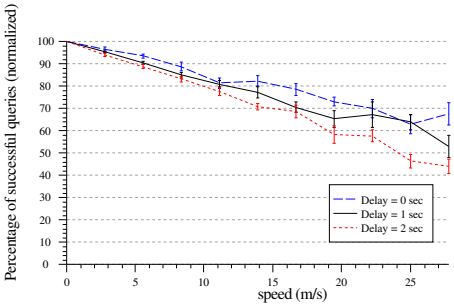

Fig. 6. Reachability of mobile node's services from node_54

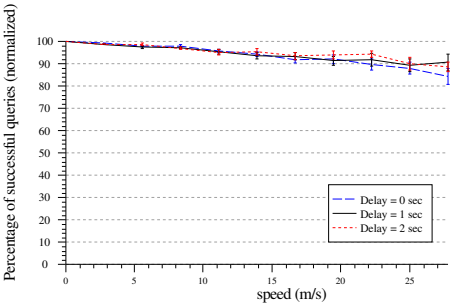

Fig. 7. Reachability of center node's services from mobile node

When we also compare the percentage of successful queries from the mobile node to the center node, see Fig. 7, we see a difference. At higher speeds the percentage of found services decreases more for node_54 than for node_62. This can be explained by the distance or number of hops between client and service. The number of hops is always large from node_54 to node_62, namely the maximum depth $d = 5$ at which the service still can be found. Especially in this case a higher delay causes the number of found services to be lower, as there are more nodes in the path between client and service that need time, proportional to the advertisement delay value, to update their

neighbors with new information. The percentage of successful queries from node_62 to node_1 is less dependent on the speed and delay, because here client and service are always moving towards each other or away from each other. As the position of the mobile node changes there is always a node within one hop distance, that knows a correct path less than a total of *d = 5* hops to the service. Thus only a small part of the path between client and service needs to be updated to be able to keep finding the service.

5.2 Full mobility experiments

The following experiments investigate a more mobile situation, where a number of users move through an area. There are 25 nodes in a simulation area of 1500×1500 meters, which gives the same node density as in the limited mobility experiments. The nodes are all moving according to a random waypoint pattern. The starting positions of the nodes are uniformly spread over the simulation area and the nodes start moving as soon as the simulation starts. For all nodes a destination is chosen distributed uniformly over the simulation area. Nodes move towards their destination with a random speed and pause at their destination for a random amount of time. After the pause a new destination is chosen for the node with a new random speed. To prevent nodes getting trapped at low speeds, as shown in [16], in our simulations we use a minimum speed of 0.1 m/s. The maximum speed varies from 1 to 20 m/s and the wait time is uniformly distributed between 0 and 30 seconds. A higher maximum speed means a higher average level of mobility for the nodes in our simulation. All simulation runs use the same start positions for the nodes, but we use 10 different random waypoint patterns per speed. The total simulation time is 60 minutes. We discard 20 minutes of simulation time to allow the random waypoint model to reach steady state. Then we start collecting results, thus for different seeds the positions will be different from the moment we start looking as well. We then vary the maximum depth *d* of the attenuated Bloom filter from 1 to 5 and do 10 simulation runs for each value of the depth parameter. Node_1 is the only node that advertises a fixed service. For these experiments the advertisement delay was uniformly distributed between 2 and 2.5 seconds. All nodes send queries for a service node_1 advertises with an exponentially distributed query rate with a mean value of 5 seconds. We start querying after 20 minutes of simulation time.

From [9] we know that the false positive probability can be calculated as $P_{fp} \approx (1 - (1 - 1/w)^{bx_j})^b$, where x_j represents the number of services in layer *j*. When using the simulation parameters from experiments 3 and 4, the worst case false positive probability occurs when all nodes are reachable in the lowest layer: $P_{fp} \approx 0.012\%$. Then for every query the probability of a false positive in layer *d* of the attenuated Bloom filter is approximately 0.012%. However, at the next hop the number of services represented in layer *d*-1 and thus the false positive probability is significantly lower. In most cases a query message due to a false positive is forwarded only one hop and then it will be discarded.

Experiment 3 In this experiment we study the effect of mobility on the number of advertisement and query messages in the network respectively.

First, Fig. 8 shows for different maximum depths *d* and increasing mobility the increase in the average number of advertisement messages per node. For low speeds

doubling the maximum depth of the attenuated Bloom filter from one to two causes an almost twice as high number of advertisement messages, but going from a maximum depth d of four to five has a smaller effect. When we first look at increasing the maximum depth d from one to two, we see the difference in the absolute number of advertisement messages increase as the maximum speed increases. However, proportionally seen there is a decrease. When we increase the maximum depth d more the influence on the number of advertisement messages gets smaller. Also increasing the speed any further does not result in an increase in the number of advertisement messages anymore. Thus at some point adding more mobility to the network does not give a higher advertisement load anymore. Nevertheless, this does have an effect on the probability of finding a service as we describe below. The average number of advertisement messages in both cases is limited by the keep_alive *period*, changes in the network cannot be detected any quicker anymore. This means a higher maximum depth d does not result in more advertisement messages. Off course the size of the advertisement packets is larger as information from an extra layer in the attenuated Bloom filter has to be transmitted, so there is still a cost involved when choosing the maximum depth d for a specific situation. Without mobility the number of advertisement messages is 0 for all depths. For a depth $d = 5$, the number of advertisement messages increases to 0.16 messages per second as the speed is increased to 20 m/s, which is still a low absolute value.

Second, we examine the average query load experienced by the mobile nodes. Figure 9 shows that for increasing depth d of the attenuated Bloom filter the average number of queries increases more. For nodes located further away, there are more possible paths from client to service over which queries are sent. When increasing the speed of the mobile nodes, there is first a decrease in the query load at 3 m/s. From speeds from 5 to 10 m/s the number of messages increases again. Speeds above 15 m/s result in a faster increase which is more pronounced for larger depths d of the Bloom filters. There are two effects that play a role, first, in a low mobility situation all Bloom filters are almost always up-to-date. When, somewhere on the path from client to service, a connection is broken due to mobility, this results in fewer query messages. Queries are no longer forwarded from that point on. Second, for higher speeds, the clean up period causes more nodes to be listed as direct neighbor. During this clean up period the information from the old neighbors is still kept, while new neighbors are being discovered. This explains the overall increase as more direct neighbors results in more queries. The maximum number of query message occurs at a speed of 20 m/s for depth d=5, where we observe 6.3 query messages per second. Note that, the effect of false positives as calculated in Sect. 5.2 is negligible in these results.

Experiment 4 We finally study the effect of mobility and the maximum depth d of the attenuated Bloom filter on the probability to find a service. As the nodes are sending queries when they find a match in their Bloom filters, we count the number of successfully found services, that is, a reply came back to the originating node. This is a measure for the reachability of the service with our protocol for different mobility patterns and varying maximum depth of the attenuated Bloom filters.

Figure 10 shows the percentage of successful queries for different levels of mobility and different values of the maximum depth d. As d increases, more layers are added

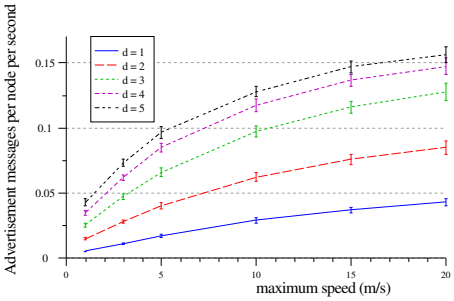

Fig. 8. Advertisement load

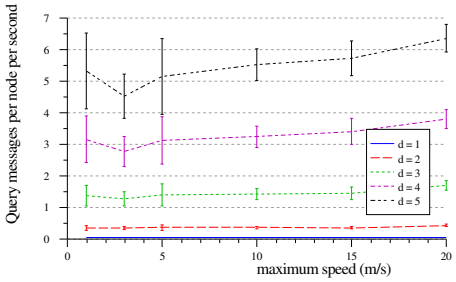

Fig. 9. Query load

tot the attenuated Bloom filters, the number of successfully found services increases as well. As the number of layers present gets higher, adding another layer does not give a large increase in the number of successfully found services anymore. Adding layers 4 and 5 gives us a slightly higher percentage of successful queries while increasing the advertisement cost. For d = 5, the percentage of successful queries is 73%. As the average speed of the nodes increases, the number of successful queries decreases, resulting in 55% success for d = 5. Propagating the changes in available services takes time, for a depth of d = 5 we have a maximum total delay of five times the advertisement delay before all nodes know about the changes in the network. Thus for a higher maximum depth d this effect is bigger, because there is a longer path of nodes between client and service that needs to know about changes in the topology. More mobility makes it more difficult to find services a larger number of hops away. There is a slight increase above 15 m/s, caused by the limited size of the area. A node forgets about neighbors after a fixed interval determined by the cleanup period. When there is more mobility nodes learn faster about more new neighbors, in effect, increasing the amount of information in the Bloom filters. More often a query is sent based on outdated information, but nodes along the path towards the service might already have new information, resulting in a successful query.

We can distinguish two main reasons for queries being unsuccessful; due to the properties of the protocol or due to the radio range combined with the location of the nodes during the simulation runs. To determine the efficiency of our protocol, we need to look at the maximum number of successful queries possible only. For a maximum depth of one, a range of r meters, and an area of aXa meters the maximum percentage of successful queries S_{max} can be approximated by dividing the area in range of node_1 with the total surface area: $S_{max} = \pi r^2/a^2$. To get such a maximum percentage of successful queries for all depths d we calculated the number of nodes within d hops from the random waypoint patterns used in the simulation runs. Figure 11 shows the percentage of successful queries normalized to the maximum possible due to network limitations. Generally the percentage is lower when the maximum number of hops increases. For low speeds the percentage of successful queries achieved by the protocol is around 99%. As speed increases towards 20 m/s the number of successful queries decreases to 91% in case d = 5.

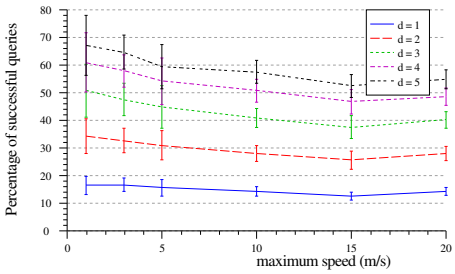

Fig. 10. Successfull queries depending on the speed

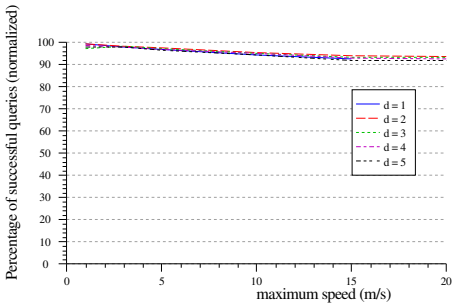

Fig. 11. Maximum reachability depending on the speed

6 Conclusions and Further Work

In this paper we have studied the lookup capabilities of a new service discovery protocol using Bloom filters in several scenarios. We thereby focused on the impact of node mobility on the performance of the protocol.

Attenuated Bloom filters can be used for local service discovery in ad-hoc networks where nodes may be mobile. To keep the information about available services in the vicinity up-to-date, advertisement messages need to be transmitted to neighbors. A delay between the time new information is received and an advertisement is sent helps to keep the bandwidth usage low. However, with a high delay the probability of finding services is lower, especially when there is more mobility in the ad-hoc network. The load of advertisement messages in a situation where nodes are moving randomly increases when there is more mobility, but is still quite low. For situations with higher mobility, the number of advertisement messages stabilizes to a maximum. For low mobility situations our protocol can find services close to 100% of the maximum possible as determined by the location and the transmission range of the nodes. With increased mobility still in a large number of cases (91% in experiment 4) services can be successfully found.

The Ahoy service discovery protocol has been implemented in a prototype [17]. In this MSc thesis, the feasibility of using Bloom filters for service discovery in ad-hoc networks is shown, and some alternative choices where investigated.

Further work includes adding more optimizations to the protocol to support node mobility even better. For instance, the number of advertisement messages can be further reduced by limiting the maximum number of hops a query can be propagated depending on the distance to the nearest service as found from the information in the attenuated Bloom filter.

References

1. Goering, P., Heijenk, G.: Service Discovery Using Bloom Filters In Ad-Hoc Networks. In: Participants Proceedings of the Dutch PhD Network on Computing and Imaging. (June 2006) 219–227

2. Veizades, J., Guttman, E., Perkins, C., Kaplan, S.: Service Location protocol. RFC 2165 (June 1997)
3. Hoebeke, J., Moerman, I., Dhoedt, B.: Analysis of Decentralized Resource and Service Discovery Mechanisms in Wireless Multi-hop Networks. In: Proc. WWIC 2005. LNCS 3510, Xanthi, Greece (May 2005) 181–191
4. Cheshire, S., Aboba, B., Guttman, E.: Dynamic Configuration of IPv4 Link-Local Addresses. RFC 3927 (May 2005)
5. Voulgaris, S., van Steen, M.: An epidemic protocol for managing routing tables in very large peer-to-peer networks. In: Proc. 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management. LNCS 2867 (2003) 41–54
6. Yang, Y., Hassanein, H., Mawji, A.: Efficient service discovery for wireless mobile ad hoc networks. In: 4th ACS/IEEE International Conference on Computer Systems and Applications. (2006) 571–578
7. Chakraborty, D., Joshi, A., Finin, T., Yesha, Y.: GSD: a Novel Group-based Service Discovery Protocol for MANETs. In: 4th IEEE Conference on Mobile and Wireless Communication Networks (MWCN). (September 2002) 140–144
8. Meier, R., Cahill, V., Nedos, A., Clarke, S.: Proximity-based service discovery in mobile ad hoc networks. In: 5th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS'05). LNCS 3543, Athens, Greece (2005) 115–129
9. Liu, F., Heijenk, G.: Context discovery using attenuated bloom filters in ad-hoc networks. Journal of Internet Engineering (2007)
10. Bloom, B.: Space/Time Trade-offs in Hash Coding with Allowable Errors. Communications of the ACM **13**(7) (1970) 422–426
11. Rhea, S., Kubiatowicz, J.: Probabilistic location and routing. In: Proc. of INFOCOM 2002. Volume 3. (2002) 1248–1257
12. Carter, J., Wegman, M.: Universal classes of hash functions. In: Proc. 9th annual ACM symposium on Theory of computing. (1977) 106–112
13. OPNET modeler software, available: http://www.opnet.com/products/modeler
14. LAN MAN Standards Committee of the IEEE Computer Society: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE std. 802.11b (1999)
15. Perkins, C., Belding-Royer, E., Das, S.: Ad hoc on-demand distance vector (aodv) routing. RFC 3561 (July 2003)
16. Yoon, J., Liu, M., Noble, B.: Random Waypoint Considered Harmful. In: Proceedings of IEEE INFOCOM. Volume 2. (2003) 1312–1321
17. Haarman, R.: Ahoy: A Proximity-Based Discovery Protocol. Master's thesis, Computer Science, University of Twente (January 2007)