Modeling Service Discovery in Ad-hoc Networks

Fei Liu
fei.liu@utwente.nl

Patrick Goering
patrick.goering@utwente.nl

Geert Heijenk
geert.heijenk@utwente.nl

Department of Computer Science
University of Twente, P.O. Box 217,
7500 AE Enschede, the Netherlands

ABSTRACT

A protocol for service discovery using attenuated Bloom filters has been proposed for ad-hoc networks. Based on our study, it can well save network bandwidth compared to conventional approaches. We have built both an analytical model and a simulation model to evaluate the performance of our novel discovery mechanism. A comparison of these two models shows that the analytical model yields very accurate results. Further, an extension to the discovery protocol is introduced, and shown to yield significant performance gains. Finally, the impact of node mobility on the performance of our discovery mechanism is evaluated, and shown to be moderate.

Categories and Subject Descriptors

C.2.2 [Network protocols]: Network protocols – *General*; C.4 [Computer Systems Organization]: Performance of Systems – *Design studies*

General Terms

Performance, Design, Verification.

Keywords

service discovery, attenuated Bloom filters, ad-hoc networks.

1. INTRODUCTION

Multi-hop ad-hoc networks (MANETs) as non-infrastructure wireless networks are widely used in emergency and temporary scenarios. The nodes participating in ad-hoc networks are mostly lightweight battery-supported devices. How to minimize the network traffic and processing power for nodes is one of the major challenges to ad-hoc networks. Considering those questions, we have proposed a novel service discovery mechanism for ad-hoc networks by using attenuated Bloom filters in [14][11]. A Bloom filter [2] is a bit vector for representing a set to support membership queries. It can represent sets of service or context information in a simple and efficient way. Attenuated Bloom filters are layers of Bloom filters, which summarize service information based on the distance (number of hops). By using attenuated Bloom filters, our mechanism supports probabilistic querying with a small probability of false positives.

PE-WASUN'07, October 22, 2007, Chania, Crete Island, Greece.

In this paper, we use the term service discovery, although the discovery mechanism described here can also be used for context discovery.

We have built an analytical model [14] and a simulation model [11] to evaluate the performance of our mechanism. The research in [14] shows that, compared to conventional solutions, using attenuated Bloom filters to discover service information can well save traffic load in practical situations in a fully distributed static ad-hoc network. Further, research in [11] shows that our approach gives nodes a good view of the services in their neighborhood, even if nodes are moving.

The contributions of this paper are the following. (1) We investigate the accuracy of an approximation that has been used in literature to express the false positive probability of Bloom filters. (2) We evaluate the accuracy of the analytical model we have proposed by comparing it with simulation results. (3) We examine the impact of an extension to the discovery protocol, replacing periodic broadcast messages with keep-alive messages. Finally (4) we investigate the impact that mobility of nodes has on the performance of our service discovery mechanism.

The paper is structured as follows. Chapter 2 will introduce related work on service discovery mechanisms. Chapter 3 will give a brief explanation of our algorithm, its extension, and both analytical and simulation models. Chapter 4 will show the experimental results for the comparison of both models and for the impact of node mobility on the performance. In Chapter 5 we will conclude the validation and propose future work.

2. RELATED WORK

Service discovery is one of the major challenges in ad-hoc networks, due to their distributed infrastructure, mobility, and limited resources. The conventional approaches can be categorized into two types: proactive discovery and reactive discovery. In a proactive approach, nodes keep a table or list that describes the location of service information in a part of or the entire network. In the rest of this paper, we call this approach complete advertisement protocol. In contrast, in a reactive approach, nodes do not have any idea about the service information in the network. A query is sent to all the nodes in the network whenever information is required. We name this approach non advertisement protocol. The choice of reactive versus proactive approach depends significantly on the network and service context and on the interaction with the underlying routing protocol [13].

Research has been done to improve the conventional approaches. Client-server approaches are used in some protocols, such as Bluetooth Service Discovery Protocol (SDP) [3]. An SDP server is normally used to register or store service information. Clients

can obtain the service information via SDP servers. This kind of approach is not suitable for a fully distributed ad-hoc network. Cluster-based approaches are also often used in ad-hoc networks, such as the Intentional Naming System (INS) [1]. Nodes in one cluster are aware of all information from all nodes in the cluster and usually a service directory is used to support inter-cluster queries. Further, a hierarchical approach is also one of the popular ways to improve the discovery in ad-hoc network, like Group-based Service Discovery protocol (GSD) [6]. Services are described in a class/subclass hierarchy to support selectively forwarding queries. Periodical advertisement and peer-to-peer caching are used in GSD.

However, none of the protocols mentioned above really contributes to saving bandwidth which is a requirement for usage in an ad-hoc network. A Bloom filter, as a space saving data structure, is often used to enhance the membership query for a set of information, such as spell check or web cache. It can also be used for service discovery. In [9], Bloom filters have been proposed to be used for a secure Service Discovery Service (SDS). Services were stored with Bloom filters locally to speed-up queries. In contrast, attenuated Bloom filter have been introduced in [16] to enhance the location mechanism for peer-to-peer networks especially while the required location is nearby. It consists of layers of basic Bloom filters used to provide probabilistic location and routing to enhance querying within a small range. Nodes exchange and update the attenuated Bloom filters with neighbors directly. By updating the filters, nodes have the knowledge of the information further away with decreased accuracy. Based on this idea, we have proposed a fully distributed and lightweight service discovery protocol using attenuated Bloom filters for ad-hoc networks in [14][11].

3. SERVICE DISCOVERY USING ATTENUATED BLOOM FILTERS

A Bloom filter is a bit array of w bits. All bits are set to 0 by default. Each piece of information, i.e. a string describing a service, is coded with b independent hash functions over the range $\{1…w\}$ agreed upon globally. The hash results are used as bit positions. All the bits in those positions are set to 1. As a result, this bit vector summarizes the collection of service information.

The string describing a service can be defined according to DNS SRV records [12]. This is also used in Apple's Bonjour service discovery protocols, Multicast DNS [7] and DNS-SD [8]. A hierarchy of domain name, transport protocol, service type and instance are used to define a service name. In [7], the domain name is ".local" and the transport protocol can be either tcp or udp. The service type registry is maintained and published by [10]. The instance is a user-friendly name, intended to be used while browsing for services. The transport protocol and service type are prefixed by an underscore to distinguish them from the domain name part of the service name. An example of a service name would be: "Color Printer._printer._tcp.local.".

Querying can be easily performed by hashing the query string and comparing the results with the Bloom filter. If all the bit positions of the query indicated to one are also set to one in the Bloom filter, the queried information is most likely (but not certainly) contained by the filter, with small false positive probability. A false positive occurs if a service is not stored in the filter, but each hash result for the queried services equals a hash result of any of the stored services. In this case the corresponding bits are set in the Bloom filter, although the service is not present. Otherwise, the filter definitely does not contain the queried information; there is no false negative in Bloom filters.

We use independent Bloom filters to represent the reachable service information for each number of hops away. These layers of Bloom filters, all with the same width w, are called attenuated Bloom filters, of which the i^{th} layer filter contains the service information i hops away. We use d to indicate the depth or number of layers an attenuated Bloom filter consist of. d is also the maximum number of hops a query will be forwarded. Context aggregation can be easily performed by using attenuated Bloom filters. To aggregate two filters A and B, each layer of filter A needs to be bitwise OR'd with the corresponding layer of filter B. The result is the aggregated filter of A and B. Attenuation of the Bloom filter is done by moving all layers one layer down (i.e. layer i becomes layer $i + 1$) and discarding the last layer (layer d). In our proposal, nodes spread information on available services by aggregating attenuated Bloom filters received from their neighbors, attenuating them, and applying the hash results for their own offered services to the first layer of the attenuated filter. The resulting attenuated Bloom filter is broadcasted to all direct neighbors.

3.1 Introduction to the Protocol

The service discovery protocol can be defined in three critical phases: service exchange, service query, and update and maintenance. Note that the first two phases have been proposed in earlier work [14], whereas the method proposed for the third phase is new here.

3.1.1 Service Exchange

Each node stores several attenuated Bloom filters: a Bloom filter containing the services from the node itself, a separate attenuated Bloom filter for each direct neighbor, and one outgoing filter which attenuates and merges all filters from neighbors and its own filter. Nodes periodically broadcast their Bloom filters and a *generation-id* to uniquely identify the information in the Bloom filters. Whenever a node receives a filter from an unknown node, it will establish a link for this node, update its filter by aggregating the current filter with the one received from this new neighbor, and broadcast the new filter to all its neighbors. Any node receiving updates from neighbors will also aggregate the changes and generate one new outgoing filter. By exchanging the filters, nodes have an overview of what kind of information is available in what range.

3.1.2 Service Query

Whenever there is a query generated by a client, the node will check its local cache, which stores the local services. If there is a match, a response will be sent back to the client. Otherwise, the query will be hashed into a basic Bloom filter, and compared with the attenuated Bloom filters from all the neighbors. If there is a match at any layer of any of those filters, this query will be propagated to the matching neighbors. Queries are propagated using unicast, for every neighbor with a match in the Bloom filter a query message will be sent. An alternative is to use broadcast for query messages. In that case, a node needs to send a query message only once, whenever there is a match in the Bloom filter

for at least one neighbor. The disadvantage is that all neighbors have to receive and process these query messages. Which method is more efficient in terms of energy usage, depends on the number of neighbors with a match in the Bloom filter and the total number of direct neighbors. The nodes that receive a query message will check both the local cache and stored filters from their neighbors. If there is any match in their local cache, a response will be sent back to the originating node. If there is any match of the stored filters within the query range, this query message will be forwarded with decremented *hop_counter*. Note that a *hop_counter* is used to restrict the query range. Queries will only be sent to a limited number of hops away. The original value of *hop_counter* equals the depth of the Bloom filters, *d*. When a node receives the same query again, as detected by a unique query identification (*Q_ID*), it will drop the query. If no match is found, e.g., because of a false positive match in a node earlier along the path between the destination node and the client, the query will be discarded. Nodes keep track of the incoming link for each forwarded query, so that the response messages can be routed back along the same path.

3.1.3 Update and Maintenance

Nodes periodically broadcast their attenuated Bloom filters to keep the other nodes updated. Further, whenever there is any change in the filter, the node will also broadcast its filter to inform the neighbors. If a node does not receive a message from one neighbor for some period, it will consider this neighbor is not available anymore (e.g. left the network or out of reach) and delete this neighbor's information from its filter.

In the situation when nodes are mobile, experiments have shown us that periodically broadcasting complete attenuated Bloom filters wastes quite a lot of network bandwidth, especially when the network does not have frequent changes. We have improved the protocol by sending keep-alive messages periodically when there is no change of service information, instead[11]. The purpose of keep-alive messages is to notify the existence of the link to neighboring nodes and the continued availability of the services represented by nodes. A keep-alive message is a short message, which contains the *generation-id* of the last broadcasted attenuated Bloom filter announcement from this node. Nodes broadcast keep-alive messages periodically, when there is no change in the network. The smaller message size saves bandwidth; as otherwise, a complete attenuated Bloom filter containing the latest changes will be broadcasted. If a node receives a keep-alive message with different *generation-id*, it will know its information is out of date, and send an update request. A node that receives an update request will send the latest filter to the requesting node.

3.2 Introduction to the Analytical Model

In order to evaluate the performance of our service discovery protocol, we have established an analytical model [14] to calculate the network cost in Matlab 7.1. In this paper, we are discussing a network with grid structure, in which each node has 4 direct neighbors in range. In this model, we define two types of cost in the network: cost for successful querying ($C^a_{successfulquering}$) and overhead cost ($C^a_{overhead}$).[1] Successful querying cost is caused by positive query results while the overhead cost is induced by the advertisement and false positive queries. Note that in this model, we focus on a static network, in which mobility and changes are not considered. The total cost of a node can be defined as the sum of those two types of cost. Overhead cost is the sum of advertisement cost ($adcost^a$) and false positive query cost ($fpcost^a$):

$$cost^a = C^a_{successfulquerying} + C^a_{overhead}, \quad (1)$$

$$C^a_{overhead} = adcost^a + fpcost^a. \quad (2)$$

This model focuses on the overhead cost. Since advertisements are broadcasted periodically at a constant rate, the advertisement cost can be defined as:

$$adcost^a = \mu \cdot adpack, \quad (3)$$

where μ is the advertisement (update) rate[2], and $adpack$ is the advertisement packet size.

The false positive cost, $fpcost^a$, is caused by the false positive probability of Bloom filters. This cost will be incurred when query messages are sent due to a false positive match at some layer in the attenuated Bloom filter, whereas these messages will not yield any (positive) result. This cost can take place on all links up to *d* hops away from the originating node. This can be calculated as:

$$fpcost^a = \lambda \cdot \sum_{i=1}^{d} cost^a_{fp,i}, \quad (4)$$

where λ is the query rate[2], and $cost^a_{fp,i}$ denotes the total cost of all false positive queries transmitted to nodes *i* hops away from the node under consideration by nodes *i-1* hops away. Such a transmission, with a packet size *qpack*, is done if the attenuated Bloom filter received from the intended receiver (at i^{th} hop) of the query by the node at *i-1* hop gives a false positive in layer *d-i*. This cost can be given as:

$$cost^a_{fp,i} = P_{fp,d-i} \cdot numofTransmission^a_{fp,i} \cdot qpack. \quad (5)$$

Here, $numofTransmission^a_{fp,i}$ denotes the potential number of transmissions caused by false positives level *i*. These are the transmissions of the nodes *i-1* hops away from the node under consideration to their neighbors. Therefore, we have:

---

[1] The superscript [a] is used when a symbol refers to the analytical model. The superscript [s] is used to refer to the simulation model. When the superscript is omitted, the symbol applies to both.

[2] Note that only the rate is relevant here. Our result for the (average) cost is valid for any distribution of the actual time between consecutive advertisements or queries.

$$numofTransmission_{fp,i} = \begin{cases} 4 & i = 1 \\ 4 \cdot (i-1) \cdot (4-1) = 12 \cdot (i-1) & i > 1 \end{cases}. \quad (6)$$

Further, the advertisement and query packet size are counted as:

$$adpack = header_{MAC} + header_{IP} + header_{UDP} + header_{AD} + w \times d, \quad (7)$$

$$qpack = header_{MAC} + header_{IP} + header_{UDP} + header_{Q} + w. \quad (8)$$

Finally, $P_{fp,d-i}$ represents the false positive probability of the layer *d-i*. This can be defined as [14]:

$$P_{fp,j} = \sum_{k=1}^{\min(b \cdot x_j, w)} P\{false\ positive \mid m_j = k\} \cdot P\{m_j = k\}, \quad (9)$$

where w denotes the width of Bloom filter; b stands for the number of hash functions used; x_j indicates the number of services represented in layer j; m_j represents the number of bits actually set in the layer j. For a grid structured network, $x_j = s \cdot (1 + 2j(j+1))$, where s is the number of services offered per node. [14] has proven that formula (9) can be rewritten as:

$$P_{fp,j} = \sum_{k=1}^{\min(b \cdot x_j, w)} \left(\frac{k}{w}\right)^b \cdot \frac{S(b \cdot x_j, k) \cdot \frac{w!}{(w-k)!}}{w^{b \cdot x_j}} = \frac{1}{w^{b \cdot (x_j+1)}} \cdot \sum_{k=1}^{\min(b \cdot x_j, w)} k^b \cdot S(b \cdot x_j, k) \cdot \frac{w!}{(w-k)!}, \quad (10)$$

where

$$S(b \cdot x_j, k) = \frac{1}{k!} \cdot \sum_{l=1}^{k} (-1)^{k-l} \cdot \binom{k}{l} \cdot l^{b \cdot x_j} \quad (11)$$

is called the Stirling number of the 2nd kind.

Formula (10) is computationally complex, and can cause numerical problems for larger values of d. Therefore we are looking for a good approximation for it. In [14], we have posed that for $b \cdot x_j$ is small compared to w, we can estimate the false positive probability by:

$$P^*_{fp,j} \approx \left(1 - \left(1 - \frac{1}{w}\right)^{b \cdot x_j}\right)^b \approx \left(1 - e^{-b \cdot x_j / w}\right)^b. \quad (12)$$

Note that in literature (e.g. [4]), the first approximation step is sometimes presented as being exact. This is not the case, as we have shown in [14]. In order to evaluate the accuracy of the approximation, we do some numerical test for realistic parameter values. We use parameter value $j = 2$ and $s = 1$, so that $x_j = 13$, so 13 services are represented in the Bloom filter. For simplicity, we denote $P_{fp,2}$ and $P^*_{fp,2}$ as *P_fp* and *P_fp**, respectively. Figure 1 shows the exact *P_fp* using formula (10) and the approximate *P_fp** using formula (12) as a function of w, for several values of b. *P_fp* decreases for increasing w. The exact *P_fp* from formula (10) is slightly higher than the approximate one, especially for low w. However, the difference between the exact and approximate *P_fp* is getting smaller as w increases. This can also be observed from Figure 2, which depicts the relative inaccuracy of the approximation, (*P_fp* - *P_fp**) / *P_fp*, for increasing w. When $b \cdot x_j$ is small compared to w, we obtain a small *P_fp*, and formula (12) is a very good approximation of formula (10). In the optimal situation, we are seeking for a reasonable size of attenuated Bloom filters (w and d) with a certain capacity (b and x_j) that causes few false positives without generating large packets to be advertised. Therefore, formula (12) can be used to estimate the minimal overhead network cost in our model.

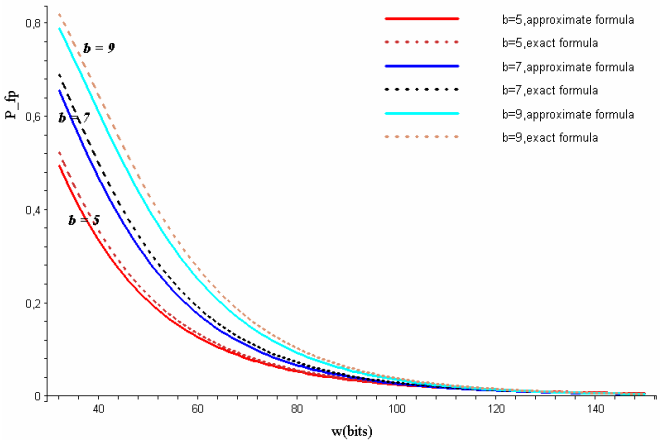

Figure 1. Comparison between the exact and approximate formulas for false positive probability

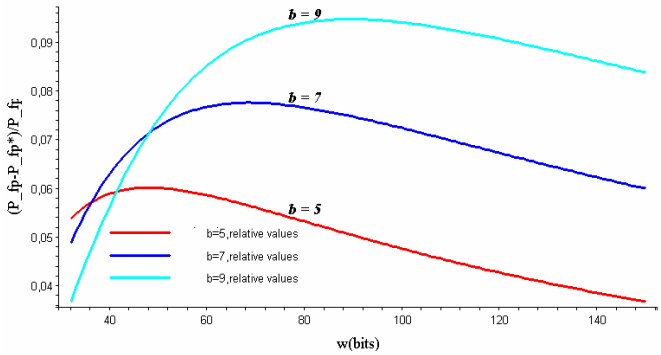

Figure 2. Relative inaccuracy of false positive probability

3.3 Introduction to the Simulation Model

To be able to compare the analytical model from Section 3.2 with the simulation model as introduced in [11] we have to set up the simulation with the same parameters as much as possible. Therefore we also use nodes in a network with a grid structure and every node has 4 direct neighbors in transmission range. We have 61 nodes in such a grid structure, which results in the center node being able to reach any node in the network in a maximum of 5 hops. The simulation model has been implemented in the discrete event simulator OPNET Modeler [15] version 11.5. A MANET node from the OPNET model library was modified to also support our protocol, consistent with Section 3.1. All nodes use the IEEE802.11b [17] standard for communication with each other. The maximum communication range is 300 meters. To allow for a good comparison with the analytical model, it is assumed that nodes beyond these 300 meters are not interfered by the transmission, nor will they carrier sense the transmission. Collisions can occur when multiple nodes are transmitting at the

same time and for broadcast packets there is no retransmission mechanism. Therefore we desynchronize messages being sent by introducing randomness in the timing. All protocol messages for the discovery protocol are encapsulated in a UDP packet and sent over IPv4.

In the simulator every node can offer one or more services. These services are randomly generated and as such for every simulation run with a different seed, the services offered will be different. All nodes in the network will advertise their services, if any, to their direct neighbors, together with any service information received from neighbors before.

To generate services and queries we first generate a random text string. This random text string is then converted into a Bloom filter by using b independent hash functions. Each hash function uses universal hashing [5] to distribute the bits set in the Bloom filter uniformly over the entire width of the Bloom filter.

Queries are generated randomly using the same random number generator. We can select what node will try to send queries and at which rate it should generate them. A generated query will only result in a query message being sent when there is a match in a Bloom filter representing the services reachable through one of the direct neighbors.

We will be looking at the cost of false positives as well as the cost of sending advertisement messages from the point of view of the center node.

4. EXPERIMENTAL RESULTS

In this section, we show our experimental results. Three basic experiments have been done to compare the results of the analytical model and the simulation model as described in Section 3. Experiment 1 demonstrates the evaluation of network cost for varying width (w) of Bloom filters of both models. Experiment 2 verifies the optimal value of w and b under certain depth d from both the analytical model and the simulation model. Experiment 3 compares the network cost of using Bloom filters and two other conventional discovery mechanisms. Further, experiment 4 shows the influence of node mobility on the performance of our discovery mechanism. In the first three experiments, we use the model without keep-alive messages, while experiment 4 will use the keep-alive mechanism.

Before describing the experiments in Section 4.2 – 4.5, we describe the setup of the comparison in Section 4.1.

4.1 Comparison Setup

In order to obtain the overhead network cost of one node, we count the number of packets sent out by this node in the simulation, including periodical broadcast packets and query packets. Note that, on average, counting all query packets sent out by a single node (which may be originated up to d-1 hops away) is equivalent to counting all query packets that are sent by nodes up to d-1 hops away as a result of a query generated in a single node. The latter approach is taken in the analytical model, whereas the former approach is taken in the simulation model, so that we can focus on the center node, disregarding boundary effects.

We assume there is no queried service information existing in the network. 10 hours of simulation will be done for each simulation run. We will observe the behavior of the center node. By dividing the total number of bits transmitted by the node by the total simulation time, we will obtain the overhead network cost. Since services and queries are randomly generated, 100 independent runs will be done to calculate a 90% confidence interval of the overhead network cost. The related analytical results will be compared with the average and confidence interval from the simulations.

In the basic simulation experiments without mobility, nodes artificially refresh the attenuated Bloom filters and advertise them periodically (i.e. without being triggered by the keep-alive mechanism). We assume the advertisement period is 600 seconds plus a random time uniformly chosen between 0 and 10 seconds to reduce collisions, as explained in Section 3.3. During one advertisement period, 600 queries are sent. Therefore, we have μ = 1/605sec and λ = 600/605sec on average for the analytical model. Note that the keep-alive message mechanism is only used in the mobile scenario.

We implemented the analytical model with approximate false positive probability (formula (12)) in Matlab 7.1. In order to evaluate the accuracy of the approximation, we use Maple 9.5 to calculate formula (10) and achieve the overhead cost with exact false positive probabilities.[3]

Some basic experiments parameters are set as follows: $header_{MAC}$ = 224 bits; $header_{IP}$ = 160 bits (assuming the use of IPv4); $header_{UDP}$ = 64 bits; $header_{AD}$ = 32 bits; $header_{Q}$ = 192 bits. We assume one service per node so that s = 1.

4.2 Experiment 1

In this experiment, we observe the network cost under different widths (w) of the Bloom filters for both analytical and simulation models. Two sets of experiments have been done: experiment (a) with d = 3 and b = 15 (see Figure 3) and experiment (b) with d = 5 and b = 13 (see Figure 4). The value of b is in both cases the optimum value for the respective depths d of the Bloom filter.

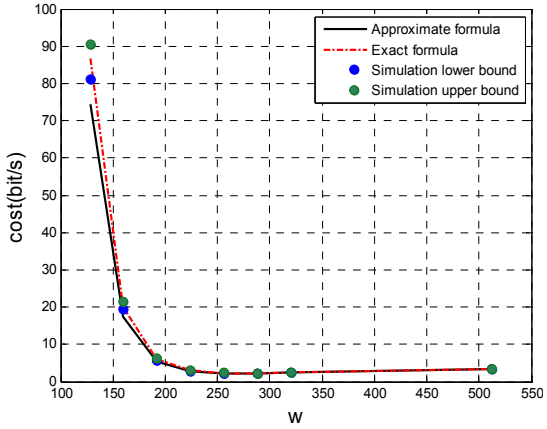

Figure 3. Network cost for d = 3, b=15

[3] The use of Maple 9.5 enabled us to avoid some of the numerical problems we had when evaluating formula (10) with Matlab 7.1 at the cost of a longer computation time.

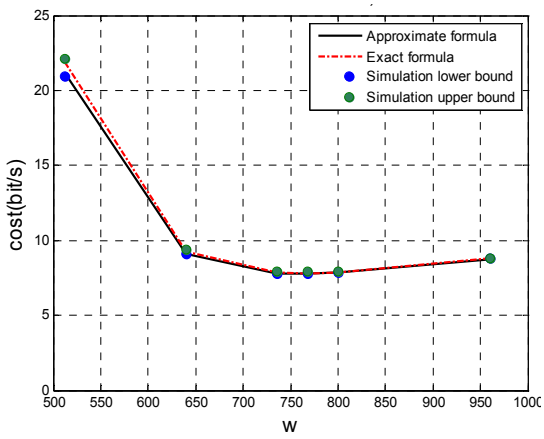

Figure 4. Network cost for d = 5, b=13

The figures show that the results from both models are very close. Simulation results also prove that there exist values of w and b for a certain value of d which leads to the minimum network load. The value of w is the same as the one from the analytical model, which is 288 bits for d = 3 and 768 bits for d = 5. The results obtained with the approximate formula (12) are very close to those from the exact formula (10), especially when $b \cdot x_d$ is smaller than w, i.e. when the false positive probability is small. The results from the exact analytical formula are always in or very close to the 90% confidence interval of the simulation results.

4.3 Experiment 2

One of the purposes of the analytical models is to help to determine the optimal attenuated Bloom filters size and the number of hash functions in use for achieving the minimum network load. We ran a number of simulations with different values for the parameters w and b when d equals 3, 4, and 5, respectively. The results are shown in the following table.

Table 1. Optimal situations when d equals 3, 4, and 5

d	w (bit)	b	BF approximate cost (bit/s)	BF exact cost (bit/s)	Simulation results 90% confidence interval (bit/s)
3	288	15	2.31	2.32	(2.33, 2. 35)
4	480	13	4.41	4.43	(4.48, 4.54)
5	768	13	7.85	7.87	(7.89, 7.97)

The values of w and b for optimal network load achieved from the three models are exactly the same. The network cost is slightly different, due to the false positive probabilities. Experiments reveal that in the optimal situation, the network cost for the approximate analytical model is always within 0.5% of the exact model. The costs from the analytical models are very close to the 90% confidence interval from the simulation results.

The results from Experiment 1 and 2 verify both the analytical and simulation models. The analytical model can well estimate the optimal parameter values for specific network situations. Both approximate and exact analytical model are very precise to achieve the optimal parameters. Further, the exact analytical model can achieve more accurate results with non-optimal parameters. However, the complicated calculation from the exact model is not convenient for larger networks. In such cases, the approximate model can be a very good estimation to calculate optimal parameter values. Furthermore, the simulation model can be used to test much more complicated situations that analytical models can not do, such as mobility of nodes.

4.4 Experiment 3

As discussed in [14], the performance of attenuated Bloom filters is highly dependent on the ratio of query and advertisement rate. Over a wide range of realistic values for this ratio using attenuated Bloom filters achieves a lower network cost than other alternative solutions, such as complete advertisement and non advertisement. In this experiment, we observe this property of the proposed mechanism by comparing service discovery using attenuated Bloom filters with two conventional solutions we mentioned in Section 2.

As a typical proactive discovery protocol, complete advertisement floods a complete description of all service information types to all network nodes within d hops. Advertisement cost is the main concern in this situation. We assume that each service information type can be presented in c bits. We assume each node up to d-1 hops away to broadcast the advertisement, so that we have:

$$cost_{compl\_ad} = \mu \cdot Totalnumofnodes_{d-1} \cdot (header_{MAC} + header_{IP} + header_{UDP} + header_{AD} + s \cdot c) \quad (13)$$

Non advertisement is a typical reactive protocol. In this case, nodes do not advertise service information types. When a query comes, nodes forward it to all the neighbors, up to d hops from the originator. The cost for querying is counted as the cost for sending queries uni-directionally to all nodes in the network. So, the number of transmissions also equals the total number of nodes within d hops minus 1:

$$cost_{no\_ad} = \lambda \cdot (Totalnumofnodes_d - 1) \cdot (header_{MAC} + header_{IP} + header_{UDP} + header_Q + c) \quad (14)$$

We assume in this experiment each service information type can be represented in 32 bits, i.e., c = 32bits. The approximate analytical model is used in here. The fixed value for μ_A = 1/605sec. We vary the value of λ_A from 10^{-3}/605 sec to 10^7/605 sec. The comparison has been done for values of d from 3 to 5. Experiments show that the simulation model gives very close results compared to the analytical models. The conclusion can be drawn that our protocol performs better than the non advertisement solution when λ_A/μ_A is larger than 0.1; and it performs better than the complete advertisement solution even if λ_A/μ_A is 10^8. We conclude that our service discovery mechanism consumes less network traffic than two conventional approaches in practical situations. Figure 5 shows the results for d = 3.

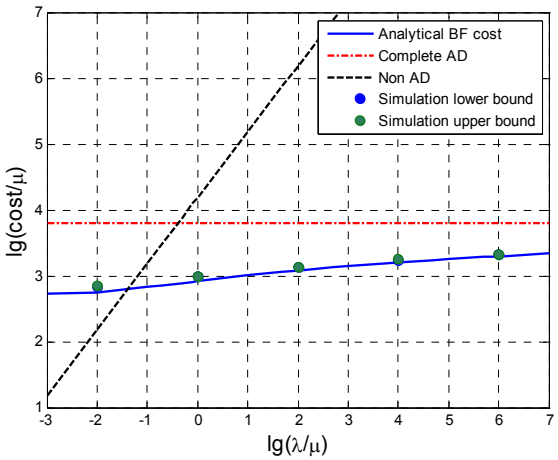

Figure 5. Performance results for λ/μ

4.5 Experiment 4

Mobility is an important aspect regarding the performance of our protocol. See [11] for more information about the impact of mobility. In order to investigate the impact of mobility, and to verify the performance of the proposed mechanism for topologies more general than the grid structure that was used in the previous experiments, we do an additional experiment with the simulation model. In this experiment we use the simulation model to show the network cost consisting of advertisement cost, keep-alive cost, and false positive query cost. We place 25 mobile nodes uniformly distributed in a simulation area of 1500x1500 meters. All nodes move according to a random waypoint pattern, thus for each node a destination is chosen uniformly distributed over the simulation area and nodes move towards that destination with a speed randomly chosen between 0.1 and *maxspeed* m/s. Upon arrival at this destination, after a random waiting period between 0 and 30 seconds, nodes pick a new destination. Nodes send advertisement messages to each other when changes in available services have been detected and otherwise keep-alive messages to save bandwidth. The period of the keep-alive messages is 15 seconds. Each node will advertise one service to its neighbors. The depth of the attenuated Bloom filter d equals 3, the width w equals 288, and b is set to 15. Note that these are the optimal values found in Experiment 2. For all nodes the query rate λ is set to 1 query per second. We do 100 runs with 10 hours of simulation time each. As before services we query for do not exist in the network, thus all queries sent are the result of false positives. We vary the amount of mobility through the value of *maxspeed* from 1 to 20 m/s. In Figure 6**,** we plot the network cost as a function of the speed of the mobile nodes. Note that the size of the 90% confidence interval of the displayed average value for the cost of broadcast and keep-alive messages was below 1%. In a static situation, i.e. *maxspeed* = 0 m/s, we have only keep-alive messages and no advertisements. This clearly shows the advantage of using small keep-alive messages to detect changes, rather than exchanging the lengthier advertisements periodically. As the speed increases more advertisement messages are needed to inform neighbors about new nodes and services. Also with increasing mobility, the number of false positives increases as on average the number of services in the Bloom filters increases; old information is not removed immediately, while new information is being added promptly. The decision to remove old information can only be made after at least one keep-alive message has been missed. Due to the unreliable nature of ad-hoc networks in practice it is better to wait for at least two keep-alive periods. In essence for higher mobility a different value of w is more optimal. Results from Experiment 1 show that there are two segments for the change of network cost separated by the optimal value for the width (w_o). If w is smaller than w_o, the network cost decreases exponentially while w is increasing. If w is larger than w_o, the network cost increases almost linearly with very gentle slope. The number of services and the amount of mobility is hard to predict. Therefore, we prefer to set w slightly higher than the w_o we calculated for the static situation, as the protocol will then operate in the linear region. This reduces the false positive cost significantly in case the node density or the number of services per node is higher than expected.

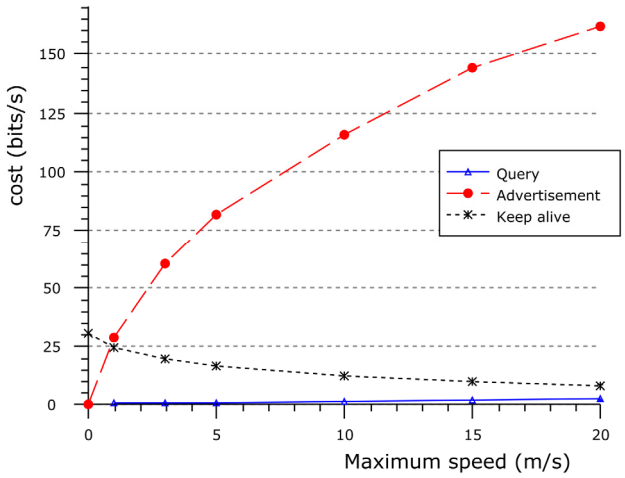

Figure 6. Network cost with mobility

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have described analytical and simulation models for our novel approach of service discovery using attenuated Bloom filters. We observed the overhead network cost for the two models in static ad-hoc networks. The results from the analytical models are always within or very close to the 90% confidence interval from the simulation model. We conclude that both models are validated. The analytical model proves to be a very good tool to compute the proper size of attenuated Bloom filters in order to achieve optimal network cost. A comparison between conventional approaches and our protocol has been done as well. Results from both the analytical model and the simulation model show that the performance of attenuated Bloom filters highly depends on the ratio of query and advertisement rate. Compared to conventional solutions, our approach causes a significantly lower traffic load over a wide and practical range of parameter settings. The usage of keep-alive messages, as introduced in this paper, saves bandwidth compared to using an advertisement only system, especially when node mobility is not too high.

In the case of more general network topologies and mobility of nodes the following conclusions can be drawn. For increasing node speeds, network cost is increasing, but overall the network cost and especially the cost of false positives remains relatively

low. When mobility is taken into account it appears that the optimal values of *w* and *b* are dependant of this.

Ongoing and future work includes further investigation of network behavior with mobility and further refinement of a protocol prototype that we have implemented. Meanwhile, security is also one of our interests. Further, the quality of our protocol, in terms of stability and scalability, will also be subject to further study.

6. ACKNOWLEDMENTS

This work is part of Freeband AWARENESS project (http://awareness.freeband.nl) and QoS for Personal Networks at Home project (PN@home, http://qos4pn.irctr.tudelft.nl/). Freeband is sponsored by the Dutch government under contract BSIK 03025. QoS for PN@Home is supported by the Dutch Ministry of Economic Affairs under the Innovation Oriented Research Program (IOP GenCom).

7. REFERENCES

[1] Adjie-Winoto, W., Schwartz, E., Balakrishnan, H., and Lilley, J. *The Design and Implementation of an International Naming System*. In Proceedings 17th ACM Symposium on Operating Systems Principles. SOSP'99, Dec 12-15 1999, Charleston, South Carolina, United States, ISBN: 1-58113-140-2, pp. 186-201.

[2] Bloom, B.H. *Space/Time Trade-offs in Hash Coding with Allowable Errors.* Communications of the ACM 13(7), 422-426.

[3] Bluetooth Consortium *Specification of Bluetooth System Core Version 2.0: Part C, Service Discovery Protocol (SDP).* November 2004.

[4] Broder, A., and Mitzenmacher, M. *Network Application of Bloom Filters: a Survay* In Internet Math, 2003, Vol. 1, No. 1, pp. 485-509.

[5] Carter, J.L., and Wegman, M.N. *Universal Classes of Hash Functions*. In Journal of Computer and System Sciences, 1979, vol.18, pp. 143-154.

[6] Chakraborty, D., Joshi, A., Finin, T., and Yesha, Y. *GSD: a Novel Group-based Service Discovery Protocol for MANETs*. In Proceedings 4th IEEE Conference on Mobile and Wireless Communication Networks, MWCN 2002, September 9 - 11 2002, Stockholm, Sweden, ISBN 0-7803-7605-6, pp. 140- 144.

[7] Cheshire, S., and Krochmal, M. *Multicast DNS*. Draft-cheshire-dnsext-multicastdns-06.txt (work in progress), August 2006.

[8] Cheshire, S., and Krochmal, M. *DNS-Based Service Discovery*. Draft-cheshire-dnsext-dns-sd-04.txt (work in progress), August 2006.

[9] Czerwinski, S., Zhao, B., Hodes, T., Joseph, A., and Katz, R. *An Architecture for a Secure Service Discovery Service*. In Proceedings of ACM/IEEE MobiCom Conference, MobiCom'99, Aug 15-20, 1999, Seattle, Washington, United States, ISBN:1-58113-142-9, pp. 24-35.

[10] DNS-SD, *DNS SRV (RFC 2782) Service Types*. http://www.dns-sd.org/ServiceTypes.html

[11] Goering, P.T.H., Heijenk, G.J., Haverkort B., and Haarman, R. *Effect of Mobility on Local Service Discovery in Ad-Hoc Networks*. To appear in Proceedings of the 4th European Performance Engineering Workshop, September 27-28, 2007, Berlin, Germany.

[12] Gulbrandsen, A., Vixie, P., and Esibov, L. *A DNS RR for specifying the location of services (DNS SRV)*. RFC 2782, February 2000.

[13] Hoebeke, J., Moerman, I., Dhoedt, B., and Demeester, P. *Analysis of Decentralised Resource and Service Discovery Mechanisms in Wireless Multi-hop Netowrks.* In Proceedings Third International Conference on Wired/Wireless Internet Communications. WWIC2005, May 11-13 2005, Xanthi, Greece, Springer Verlag 3510/2005, ISSN 0302-9743, ISBN 3-540-25899-X, pp. 117-127.

[14] Liu, F., and Heijenk, G. *Context Discovery Using Attenuated Bloom filters in Ad-hoc Networks*. In Journal of Internet Engineering, 2007, Vol. 1, No. 1, pp. 49-58.

[15] OPNET modeler software, available: http://www.opnet.com/products/modeler.

[16] Rhea, S., and Kubiatowicz, J. *Probabilistic Location and Routing*. In Proceedings 21st Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2002, June 23-27 2002, New York, United States, ISSN 0-7803-7476-2, vol.3, pp. 1248-1257.

[17] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. IEEE std. 802.11b, 1999.