Corrected Version

Context Discovery Using Attenuated Bloom Codes

Model Description and Validation

Fei Liu and Geert Heijenk

{fei.liu, geert.heijenk}@utwente.nl

CTIT Technical Report TR-CTIT-06-09

ISSN 1381-3625

Centre for Telematics and Information Technology

University of Twente, the Netherlands

March 20, 2006

Modified on November 12, 2006

Keywords: *Context discovery, Bloom code*

Abstract:

A novel approach to performing context discovery in ad-hoc networks based on the use of attenuated Bloom filters is proposed in this report. In order to investigate the performance of this approach, a model has been developed. This document describes the model and its validation. The model has been implemented in Matlab, and results are also shown in this document. Attenuated Bloom filters appear to be a very promising approach for context discovery in ad hoc networks compared to conventional solutions. The results show that using attenuated Bloom filters in context discovery can well save traffic load in a fully distributed ad hoc network in practical situations.

1 Introduction

Ad-hoc networks are non-infrastructure wireless networks in which most of the terminals are both mobile and power-consumption constrained. When one needs to obtain a service or context information from other devices, querying and fixing the location of the service or context information source might generate a lot of traffic. In a network with a high query rate, such traffic can be rather heavy. As a result, the terminals consume quite an amount of power and bandwidth for querying. An efficient context discovery mechanism needs to be developed for such situations.

This document is the extended version of [1]. It describes the development of a discovery mechanism for networks that are context aware. These networks utilize context information to improve their operation, or to enrich the services provided to users. We propose a novel approach to discover context information sources in an ad-hoc network based on the use of attenuated Bloom filters, which represents a decentralized space-efficient discovery method. Instead of broadcasting full information about the type and location of context information, nodes send attenuated Bloom filters which contain context type information for all the reachable nodes up to a certain number of hops away. Moreover, Bloom filters have a special feature of false positive probability, which leads to probabilistic querying. Queries are only forwarded in the directions which possibly contain the required information. Our analysis reveals that this type of probabilistic discovery method can substantially reduce the network load compared to discovery using traditional approaches.

This document is structured as follows. Section 2 will introduce our novel approach for performing context discovery using attenuated Bloom filters. In Section 3 we will describe an approximate model for the transmission costs of our method. Further we compare it with context discovery without attenuated Bloom filters. In Section 4 we will provide numerical results. Finally, in Section 5, we present our conclusions and propose future work.

2 Context discovery using attenuated Bloom filters

2.1 Related work on context discovery

Context discovery has a lot of resemblance to service discovery. In computer science, context refers to the circumstances under which a device is being used, while service can be any software or hardware entity that a user might be interested to utilize [2]. Both context and service can be described by a certain format or template. The discovery of both context and service can be understood as an action of looking for requested context/services in the network. Service Discovery Protocols (SDPs) can be classified into centralized and decentralized architectures. In ad-hoc network environment, nodes are both mobile and mostly battery-powered. Those characteristics fit well with some features of decentralized architectures. The choice for a proactive or reactive SDP in decentralized architectures depends substantially on the network and service context and on the interaction with the underlying routing protocol [3].

Service descriptions are different for various SDPs. The most popular format is the attribute-value structure [4]. For instance, the Service Location Protocol (SLP) [5] uses service templates which predefine the attributes in a template document readable by humans and machines. Service agents (SAs) advertise the location of one or more services; directory agents (DAs) store service location information centrally. Whenever necessary, user agents will look for the required services at SAs and DAs. The Bluetooth Service Discovery Protocol (SDP) [6] defines a service record consisting of the entire list of attributes, which is then stored in the SDP server. Clients will send requests to the SDP server to obtain the required services.

Further, hierarchical attribute-value pairs, which mostly rely on eXtensible Markup language (XML), are also used in some protocols, such as Global Service Discovery Architecture (GloSev)[7] and Group-based Service Discovery protocol (GSD)[8]. GloSev is proposed for worldwide and local area network usage. Services are described and categorized hierarchically by using the Resource Description Framework (RDF) which is based on Uniform Resource Identifiers (URI) and XML. The hierarchical service architecture is similar to the DNS domain name architecture. GSD is a distributed service discovery protocol for Mobile Ad hoc NETworks (MANETs). Services are described based on DARPA Agent Markup Language (DAML+OIL). Advertisements are sent periodically to nodes within a maximum number of hops. Each node has peer-to-peer caching to keep a list of local and remote services

that a node has received from advertisements. Services are also grouped to ease service discovery by selectively forwarding queries.

In some protocols, such as Jini [9], attributes are described as Java objects. Service objects are registered in service registries, which are also used to look up services. A client needs to download the service object and invoke it to access the service.

Among the protocols mentioned above, GloSev was developed for wide area networks. SLP and Jini were designed for local area networks. GSD and Bluetooth SDP are specifically for MANETs. Further, it is clear that which method is used to describe services, sending the complete service attributes causes heavy traffic. It is inefficient way in a high-density mobile ad hoc network with many services to be advertised and\or queried. Due to the limited battery power of terminals, a simple, efficient context description and discovery mechanism is required. Clearly, a mobile ad hoc network is less suitable for a centralized structure due to the mobility of the nodes. Nodes should not depend on other specific nodes to reach the required context information. Context discovery using attenuated Bloom filters can solve this problem in ad hoc networks.

2.2 Brief introduction of attenuated Bloom filters

A Bloom filter [10] is a data structure for representing a set in order to support membership queries. It can denote a set simply and efficiently, with a small probability of false positives. Bloom filters can be used in various network applications, such as distributed caching, P2P/overlay networks, resource routing, packet routing, and measurement infrastructure [11]. Bloom filters were also proposed to be used as an efficient approach for lossy aggregation and query routing for a Secure Service Discovery Service in [12]. Recently, researchers have explored the applications of Bloom filters to ad hoc networks, such as speeding-up cache lookups [13], group management [14], and hotspot-based trace back [15].

A Bloom code can represent a set of context information types. Each context type will be coded by using b independent hash functions over the range $\{1 \ldots w\}$, where w is the width of the filter. The default value for each bit in the Bloom code is 0. The bits of positions associated with the hashes will be set to 1. Our approach uses attenuated Bloom filters, each of which consists of a few layers of basic Bloom filters. The first layer of the filter contains the context type information for the current node, while the second layer contains the information about the nodes one hop away, and so on. In other words, a node can find the context type information i hops away in the i^{th} layer. When querying for a certain type of context information, the same hash functions are performed. If all positions in a Bloom filter indicated by one of the hashes contain a 1, the presence of the queried context type is likely (but not certain). Otherwise the context type is not present. The use of these attenuated Bloom filters introduces the possibility of having false positives, which will be resolved during a later stage of the context discovery process. By using attenuated Bloom filter consisting of multiple layers, context sources at more than one hop distance can be discovered, while avoiding saturation of the Bloom filter by attenuating (shifting out) bits set by sources further away.

For example, we assume a 6-bit Bloom filter with b equal to 2. If location information is hashed into {1, 3} and temperature information is mapped into {2, 5}, we obtain the filter shown in Figure 1.

0	1	2	3	4	5
0	1	1	1	0	1

Figure 1. A simple 6-bit Bloom filter

The filter will give a positive answer to queries for location or temperature information. It definitely does not contain presence information, which is hashed into {0, 3}. Nodes may also think humidity information {1, 5} is contained in this filter, but actually it is not. This situation is termed as false positive [11].

Context aggregation can be simply implemented by attenuated Bloom filters. When a node A receives incoming Bloom filters $filter_B$ and $filter_C$ from neighbors B and C respectively, it shifts all the contents of $filter_B$ and $filter_C$ one layer down and discards the last layer. The first layers will be filled with 0s. An OR operation will be done to those new filters, $filter_B'$ and $filter_C'$, and the first layer will filled by (first layer of) $filter_A$. Consequently, the Bloom filter of node A is updated such that the first layer represents the

local information from node A; the second layer contains the information from neighbor B and C; the third layer covers the information two hops away which can be reached via B or C. Figure 2 shows the process of context aggregation in a node.

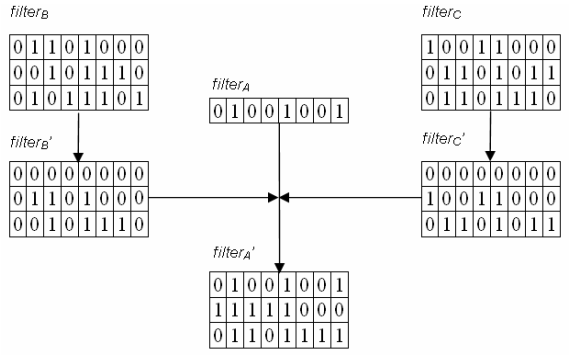


Figure 2. Context aggregation

2.3 Context discovery with attenuated Bloom filters

Context discovery by using attenuated Bloom filters is a method combining proactive and reactive discovery mechanisms. Nodes can obtain an overview of available context types and their distribution by exchanging Bloom filters. When a node enters a new environment, it first sends a Bloom filter with its own context information types to its neighbors. When neighbors receive this filter, they will merge the new incoming filter to their existing attenuated Bloom filters by shifting the incoming filter and doing an OR operation, as explained in Section 2.2, and rebroadcast the updated filter. The new node stores the incoming Bloom filters separately for each neighbor, and also generates a new filter aggregating all the reply filters with the local one. This new filter will be sent to the neighbors in the next advertisement. The Bloom filters will be exchanged periodically, or when they change.

When there is a query, the node will first check locally whether the required context information exists. If not, the query will be encoded into a Bloom code using the same hash functions and compared with the locally stored filters. If there is a match at any layer in any of the local filters, a query message containing the Bloom code will be sent to the neighbor from which the stored filter was received. The same action will be taken by a node receiving a query. When a node receives the same query again, it will drop the query. Further the path of the query message will be recorded by each node. If an exact match is finally found, a Context-Available (CAL) reply will be sent back along the same path. If no match is found, e.g., because of a false positive match in an earlier node, the query will be discarded. In this way, query messages will be filtered out as early as possible, depending on the stored attenuated Bloom filters. The originating node will set up a connection to the destination node based on the best CAL reply. Note that a hop counter is used to restrict the query range. Queries will only be sent a limited number of hops away, based on the depth of the Bloom filters. If no CAL is received by the originating node after a time out, it understands that the required information is unavailable in the current range. It will make a choice to enlarge the discovery area or send the query again after a certain time.

Note that recording the path of the query message requires maintaining (soft) state routing information in the nodes. This can put a burden on these nodes. Alternatively, the return path for CAL replies can be stored in the query messages. This will result in increased transmission costs.

3 Performance and Modeling

Queries due to false positives can potentially contribute significantly to the costs of context discovery. In order to reduce the number of unnecessary queries, we have to reduce the ratio of false positives. However, the minimum false positive ratio, which is by definition equal to 0, will result in large Bloom filters. We believe there is a balance to be struck between a reasonable false positive ratio and the size of Bloom filters to achieve the optimal network cost. The size of Bloom filters is decided by its width, depth, the number of hash functions, and the cardinality of the represented set. In this section, we will present a

model to find the optimum balance between those parameters to minimize the network cost. Further comparisons between context discovery with and without Bloom filters will be made.

3.1 Network structures and related vital parameters

To observe the performance of the model, we choose three typical network structures: grid structure, tree structure, and circle structure. A grid structure represents a simple and uniform network. A tree structure is used to calculate approximate upper bound of the network, due to the exponential increased number of nodes within range with the increase of distance (hops). Finally, a circle structure shows an ideal network structure for a fully distributed mobile ad hoc network.

Before discussing the network structures, we would first like to introduce some related vital parameters. In general, we assume that each node has the same number of context information types, s. Further all context types are supposed to be unique, and taken out of an infinitely large set of possible context types. The same width, w, and depth, d, of attenuated Bloom filters, and the same b hash functions are used for the entire network. Queries are forwarded by at most d hops, based on the depth of Bloom filters. General notation is listed in the Table 1.

Table 1. Notation

General		**Bloom Filter**	
Notation	Description	Notation	Description
s	Number of services (context information types) per node	w	the width of the filter
μ	advertisement (update) rate	d	the depth of the filter
λ	query rate	b	number of hash functions
n	network density (nodes/m^2)		
r	communication range		

3.1.1 Grid structure

We start with a simple situation with a grid network structure, where each node has 4 direct neighbors within communication range. We define *conn* as the number of neighbours for an arbitrary node. For this grid structure, of course, *conn* = 4. The structure is shown in Figure 3a.

Since we assume each node has the same number of unique context type information, the context information that a node could reach within a certain range depends substantially on the density of the nodes. In this model, the number of new nodes in i^{th} hop can be obtained as:

$$numofNewnodes_{i-g} = 4 \cdot i \tag{1}$$

This value increases linearly with the distance (the number of hops).

3.1.2 Tree structure

A tree structure with same number of branches for each node can be counted as an upper bound structure. To coordinate with the grid structure, we suppose that each node has 3 branches, so that *conn* = 4, as shown in Figure 3b.

The number of new nodes in i^{th} hop equals to:

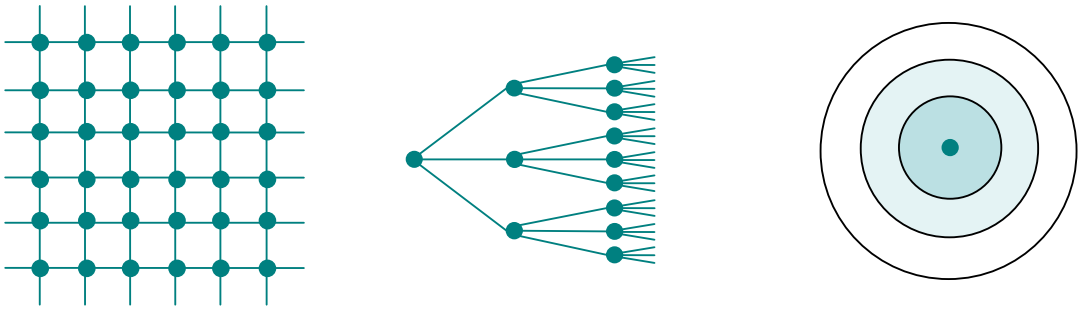
$$numofNewnodes_{i-t} = (conn-1)^i = 3^i \tag{2}$$

From Formula (2), the number of new reachable nodes increases exponentially. Apparently, this is a network with an increasingly high density of nodes. The direct result of this is much heavier traffic in the network.

3.1.3 Circle structure

Ideally, the two dimensional radio coverage of a mobile terminal is a circle. Therefore, a circle structured network is also assumed in our modeling. The communication range of nodes will be generalized as a set

of concentric circles, as shown in Figure 3c. Nodes located in the inner circle are the ones reachable within one hop from the center node. Nodes in one ring outside the inner circle are the ones reachable in two hops, and so on.



(a) Grid Structure (b) Tree Structure (c) Circle Structure

Figure 3. Network structures: (a) grid structure; (b) tree structure; (c) circle structure

We assume that the communication range, r, of a node is 30 meters. The density of the network is n nodes/m^2 on average. New nodes in i^{th} hop are ones located in the i^{th} ring from the center node. The number of these nodes is:

$$numofNewnodes_{i-c} = n\pi\left(i^2 - (i-1)^2\right)r^2 = (2i-1)n\pi r^2 \quad (3)$$

To coordinate with the grid structure, $numofNewnodes_{i-g}$ should equal to $numofNewnodes_{i-c}$. So we have

$$4i = (2i-1)n\pi r^2 \quad (4)$$

$$n = \frac{2}{\left(1-\frac{1}{2i}\right)\pi r^2} \quad (5)$$

If i is large, we can estimate the value of n by using the formula:

$$n \approx \frac{2}{\pi r^2} \quad (6)$$

From which we obtain n ≈ 0.0007.

3.2 Cost functions

There are two kinds of traffic in the network: advertisements and queries. Three types of advertisement messages can be identified: normal advertisement, updates, and maintenance. We assume that advertisements are broadcast periodically at a constant rate. Among queries, there are also two types based on the different answers: positive query and false positive query. Note that there is no false negative in Bloom filters. Therefore, the cost for a node is defined as the sum of the advertisement cost, positive cost and false positive cost. We determine the cost considering the transmission cost of a single node in the network. The cost is expressed in bits per second per node

$$cost = adcost + pcost + fpcost \quad (7)$$

Since an advertisement is broadcasted periodically, its cost $adcost$ can be derived by:

$$adcost = \mu \cdot adpack \quad (8)$$

where μ is the advertisement (update) rate; $adpack$ is the advertisement packet size.

The postive answer cost is denoted as *pcost*. To simplify the problem, we assume that queried services are not available in the network, which indicates all queries will result in a negative answer, or a false positive, i.e. *pcost*= 0.

Then *fpcost* represents the transmission cost for false positive queries incurred by a query initiated in the node under consideration. Transmission of such query messages can take place on all links up to *d* hops away from the node under consideration. Thus, we can denote the false positive cost as

$$fpcost = \lambda \cdot \sum_{i=1}^{d} cost_{fp,i} \tag{9}$$

We assume queries are performed at a certain rate λ, i.e., λ queries will be initiated per second per node. $cost_{fp,i}$ denotes the total cost of all false positive queries transmitted to nodes *i* hops away from the node under consideration.

Let us first determine the probability of generating a false positive as a result of a query. We define $P_{fp,j}$ as the probability of a false positive for layer *j;* x_j represents the number of services available in layer *j*. In this paper, we assume that the hash functions we choose are perfectly random. When an element is inserted, the probability that a certain bit is not set to one by a certain hash functions is (1-1/w). The probability that the bit is not set by *b* hash functions is $(1\text{-}1/w)^b$. After insertion of x_j elements, the probability that a specific bit is still 0 is equal to:

$$\left(1-\frac{1}{w}\right)^{bx_j} \approx e^{-bx_j/w} \tag{10}$$

So the probability that the bit is 1 is:

$$\left(1-\left(1-\frac{1}{w}\right)^{bx_j}\right) \tag{11}$$

The false positive probability, which is a specific set of *b* bits are 1, is approximately equal to:

$$P_{fp,j} \approx \left(1-\left(1-\frac{1}{w}\right)^{bx_j}\right)^b \approx \left(1-e^{-bx_j/w}\right)^b, \tag{12}$$

when $b \cdot x_j$ is small compared to *w*.

Formula (11) shows that the false positive probability depends on the width (*w*) and number hash functions (*b*) of the Bloom filters, and the number of services contained inside the filter. We define the number of services in j^{th} hop for the grid, tree, and circle structures as:

$$x_{j-g} = 4s \cdot \left(1+\sum_{k=1}^{j} k\right) \tag{13}$$

$$x_{j-t} = s \cdot \left(1+\sum_{k=1}^{j} (conn-1)^k\right) \tag{14}$$

$$x_{j-c} = s \cdot \left(1+n\pi(j \cdot r)^2\right) \tag{15}$$

The assumption is that all services that are reachable in *j* or fewer hops are represented at the j^{th} layer. A service that is represented at a certain layer is also assumed to be represented at all layers below, because of likely alternative paths with more hops to the node containing that service, which will impact the broadcasted attenuated Bloom filters.

In order to obtain this false positive query cost to the i^{th} hop, we have to count the possible number of query transmissions sent by nodes *i*-1 hops away to their neighbors, $numofTransmission_{fp,i}$. Such a transmission is indeed done, with a packet size qpack, if the attenuated Bloom filter received from the intended receiver of the query (the node at *i* hops) by the node at *i*-1 hops gives a false positive in layer

d-i. This false positive will occur with probability $P_{fp,d-i}$, where $P_{fp,j}$ is defined as the probability of a false positive occurring in layer *j*. Note that a false positive at any of the layers *j*<*d-i* automatically implies a false positive at layer *d-i*, since bits set in layer 0 are duplicated to all layers. So, false positives at layers *j*<*d-i* do not have to be accounted for. A false positive in a layer *j*>*d-i* does not result in a query to be transmitted, because the final destination of that query would be more than *d* hops away from the node originating the query. Finally, note that the query to be sent to the node at *i* hops will reach the node at *i*-1 hops for certain, since the false positive at layer *d-i* (as received by the node at *i*-1) will also appear at the query originating node at layer *i*-1. The resulting false positive query cost to the i^{th} hop can be given as:

$$cost_{fp,i} = P_{fp,d-i} \cdot numofTransmission_{fp,i} \cdot qpack \tag{16}$$

$numofTransmission_{fp,i}$ can be obtained by number of new nodes reached in $(i\text{-}1)^{th}$ hop times the number of neighbors each new node has, which is

$$numofTransmission_{fp,i} = numofNewnodes_{i-1} \cdot numofNeighbors \tag{17}$$

where *numofNeighbors* refers to the number of transmissions per node. The value of $numofTransmission_{fp,i}$ highly related to the network structures. We will discuss this in detail for the defined network structures.

3.2.1 Grid Structure

In the grid Structure, each node has *conn* = 4 direct neighbors. Except for the incoming interface, a node is going to forward queries to (*conn*-1) nodes. Based on Formula (1) and (17), we have $numofNeighbors_g$ = *conn-1*, so

$$numofTransmission_{fp,i-g} = 4 \times (i-1) \times (conn-1) \quad (i > 1) \tag{18}$$

Specially, for the first hop, the querying node is going to send queries to all 4 neighbors, so that

$$numofTransmission_{fp,1-g} = conn \quad (i = 1) \tag{19}$$

So we have the false positive query cost at hop *i* is

$$cost_{fp,i-g} = \begin{cases} conn \cdot qpack \cdot \left(1 - e^{-bx_{d-1}/w}\right)^b & i = 1 \\ qpack \cdot \left(1 - e^{-bx_{d-i}/w}\right)^b \cdot 4(i-1)(conn-1) & 2 \le i \le d \end{cases} \tag{20}$$

3.2.2 Tree Structure

For the tree structure in this report, we define that each node has 3 branches and *conn* = 4, so that each query will be forwarded to (*conn-1*) nodes. Based on the formula (2) and (17), we have $numofNeighbors_T$ = *conn* -1, so:

$$numofTransmission_{fp,i-t} = (conn-1)^{i-1} \cdot (conn-1) = (conn-1)^i \tag{21}$$

So the false positive query cost at hop *i* is:

$$cost_{fp,i-t} = \left(1 - e^{-bx_{d-i}/w}\right)^b \cdot (conn-1)^i \cdot qpack \tag{22}$$

3.2.3 Circle Structure

For the circle structure, let us suppose that each node can reach all the nodes within communication range, which are $(n\pi r^2 - 1)$ nodes excluding the querying node itself. When a node forwards a query to the next hop, there are potentially $(n\pi r^2 - 1)$ neighbors to transmit the query to. So we have $numofNeighbors = (n\pi r^2 - 1)$. These transmissions to the i^{th} hop will potentially be done by all nodes

that can be reached within i-1 hops from the node under consideration, excluding those that can also be reached within *i*-2 hops. Therefore, there will be $\left(n\pi(i-1)^2 r^2 - n\pi(i-2)^2 r^2\right) = (2i-3)n\pi r^2$ (i>1) nodes involved in transmission. This results in the following potential number of transmission sent in the i^{th} hop. Note that this does also includes transmissions to nodes that have already received the query before. These duplicates will be discarded upon reception.

$$numofTransmission_{fp,i-c} = (2i-3)n\pi r^2 \cdot \left(n\pi r^2 - 1\right) \quad 2 \le i \le d \tag{23}$$

For the first hop, the original querying node will send the query to all the nodes in range, so that the number of transmissions sent in 1st hop equals:

$$numofTransmission_{fp,1-c} = n\pi r^2 - 1 \tag{24}$$

So the false positive query cost at hop *i* is:

$$cost_{fp,i-c} = \begin{cases} (1-e^{-bx_{d-1}/w})^b \cdot \left(n\pi r^2 - 1\right) \cdot qpack & i = 1 \\ (1-e^{-bx_{d-1}/w})^b \cdot (2i-3)n\pi r^2 \cdot \left(n\pi r^2 - 1\right) \cdot qpack & 2 \le i \le d \end{cases} \tag{25}$$

For the sizes of the advertisement and query packets, we assume that the context discovery protocol using Bloom filters is running on top of UDP. For both advertisements and queries, besides the header of Bloom Filters protocol, the headers of UDP, IP, and MAC layer will be attached. The advertisements and queries packet size are defined as follows:

$$adpack = header_{MAC} + header_{IP} + header_{UDP} + header_{AD} + w \times d \tag{26}$$

$$qpack = header_{MAC} + header_{IP} + header_{UDP} + header_{Q} + w \tag{27}$$

Based on the Ad-hoc On-demand Distance Vector (AODV) routing protocol, we define the preliminary advertisement packet and query packet formats as follows:

- Advertisement packet format:

Type	Optional	Width of BFs	Depth of BFs	Hop Count
DATA				

Having taken a brief look at the advertising stage, the advertisement segment structure should consist of header fields and a data field. The header here is defined as 32 bits. It includes type field, optional field, width of Bloom filters, and hop counter. The data field is used for the Bloom filters of advertisement information.

  - The 4-bit type field is used to indicate the packet type.
  - The 4-bit optional field is used when you need to indicate the priority of the data. It can be specified more detail later.
  - The 8-bit width of Bloom filters field and the 8-bit depth of filter field are used to clarify the size of filter used in the current network.
  - The 8-bit hop count field is used to count the number of hops this packet traveled. (Actually this packet will only travel for one hop. We can use this field in another use, such as security)

- Query packet format :

Type	Reserved	Width of BFs	Freshness ID	Hop Count
Query ID				
Source ID				
Source Sequence Number				
Reserved				

Reserved
DATA

The querying stage requires the query packet contain more information than advertisement packet. The segment structure consists of header field and data field. The header here is defined as 192 bits, which includes type field, reserved field, width and depth of Bloom filters, hop count field, query ID, Source ID, Source sequence number, and reserved fields. The data field is used to contain the Bloom codes for querying.

- The 4-bit type field is used to indicate the packet type.
- The 4-bit reserved field is used to indicate the priority of the query. It can be specified more detail later.
- The 8-bit width of Bloom filters field is used to clarify the width of filter used in the current network. In our model, we assume the Bloom code of query have the same size of the width of Bloom filters.
- The 8-bit freshness field is used to indicate the freshness of the query.
- The 8-bit hop count field is used to count the number of hops this packet traveled.
- The 32-bit query ID field contains the ID of the query.
- The 32-bit source ID field includes the information of source node
- The 32-bit source sequence number field is used to record the sequence number of source node.
- The 64-bit reserved field is kept for further definition of the protocol, such as security, redundancy, etc.

So we have $header_{AD}$ =32 bits; $header_{Q}$ =192 bits.

3.3 Two Extreme Cases

To evaluate the performance of context discovery using attenuated Bloom filters, we have compared it with two alternative discovery solutions: complete advertisement and non advertisement.

Complete advertisement floods all network nodes within d hops with a complete description of all context information. Nodes have the complete map of the network, which indicates how nodes can send queries directly to the destination. It is a proactive protocol. The advertisement cost is the main concern in this situation. We assume that each context information type can be presented in c bits, so we have:

$$cost = adcost = \mu \times numofTrans \times (header_{MAC} + header_{IP} + header_{UDP} + header_{AD} + s \times c) \quad (28)$$

where $numofTrans$ denotes the number of transmissions for the entire advertisement within d hops. We suppose each node up to d-1 hops away to broadcast the advertisement, so that we have:

$$numofTrans = \left(1 + n\pi\left((d-1)\cdot r\right)^2\right) \quad (29)$$

In the non advertisement case, nodes do not advertise context information types. When a query comes, nodes forward it to all the neighbors. It is a reactive protocol. Nodes do not have any idea about the network. The queries are spreading around the whole network, up to d hops from the originator. There is no way to stop forwarding queries, even though the query node has already received an answer. The cost for querying is counted as the cost for sending queries unidirectionally to all nodes in the network, which is:

$$cost = qcost = \lambda(header_{MAC} + header_{IP} + header_{UDP} + header_{Q} + c)\left(1 + n\pi\left(d \cdot r\right)^2\right) \quad (30)$$

4 Experimental Results

The model described above has been implemented in Matlab 6.5. Using the model, four sets of experiments are done for all three network structures. The following tables and figures show the results of the experiments. Experiment 1 is used to achieve the optimal cost by choosing the proper width, w, of the Bloom filter and the number of hash functions, b, with given depth, d, of the filter, query rate, λ, and

number of services, s, per node. Experiment 2 shows the influence of the query rate, λ, on the network cost for given d and s. The influence of the query range, d, is evaluated in Experiment 3. In the final experiment, we show the impact of density of services, s, in the network.

In all the experiments below, we assume the update frequency $\mu = 0.1$ and each context information type can be represented in 32 bits, i.e., c = 32 bits. The sizes of headers are assumed as follows [16]: $header_{MAC}$ =160 bits; $header_{IP}$ =320 bits (assuming the use of IPv6); $header_{UDP}$ =64 bits; $header_{AD}$ =32 bits; $header_{Q}$ =192 bits.

4.1. Grid structure

In this section, the experiments for the grid structure are done.

4.1.1 Experiment 1

In this experiment, we assume query rate $\lambda = 0.1$, s =1. For each given value of filter depth, d, the experiment result shows that there exists a certain value of w and b which leads to the minimum network cost. The result is shown in Table 2. It is also compared with the complete and non advertisement under similar situations.

Table 2. Optimal BF cost for certain depth of filter compared with complete and non advertisment

d	w (bit)	b	BF cost (bit/s)	Complete Advertisement (bit/s)	Non Advertisement (bit/s)	Maximum number of services in BF
3	128	5	135	790	1843	13
5	256	5	218	2493	4608	41
7	448	5	477	5168	8602	85
10	896	5	1277	11005	16896	181

As we see from Table 2, for each depth of the filter, the proper width and number of hash functions leads to a minimum network cost which is much lower than for the cases of “complete advertisement” and “non advertisement”. The difference becomes larger as query range d increases. The final column shows the maximum number of services that are represented in one Bloom filter for the related depth of Bloom filter.

4.1.2 Experiment 2

Using Bloom filters, we can reduce the packet size by using simple and efficient coding. However, false positives also create redundant traffic. This trade-off heavily depends on the relation between the update frequency $\boldsymbol{\mu}$, and the query rate $\boldsymbol{\lambda}$. We expect that there exists a (high) value for $\boldsymbol{\lambda}$ at which the traffic generated due to false positives is more than the benefit of using Bloom filters. In contrast, if there are only few queries in the network, it does not pay to broadcast the context information to the entire network. A non advertisement protocol can perform better in this case. This experiment is going to discuss the suitable range of using Bloom filters for context discovery to achieve the minimum network cost.

We set μ as a reference, and change the value of λ. Here we talk about λ/μ. The experiments show that the suitable range of λ/μ decreases when the depth of the filter, d, increases. When each node has only one service (s = 1), the Bloom filter context discovery algorithm performs better than the non advertisement algorithm when λ is at least 0.1 times μ, i.e., if queries are generated at least once per 10 advertisement periods. Further, the Bloom filter algorithm performs better than the complete advertisement algorithm even if λ is 10^8 times μ, when d is between 3 and 10. However, the upper bound of λ/μ for better performance of Bloom filter discovery mechanism decreases, while d increases. Figure 4a shows the situation when $d = 5$.

We found that in practical situations the Bloom filter algorithm has a better performance. Therefore, it is a promising algorithm for the grid structure networks. Note that the axes in the Figure 4a are represented in log scale.

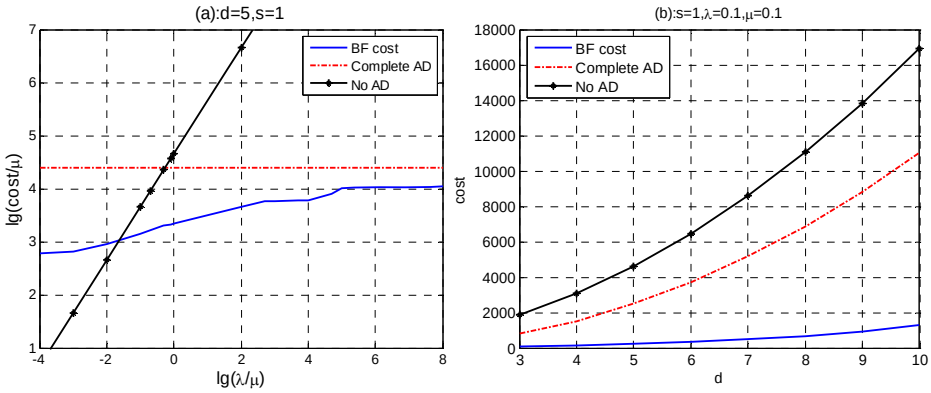


Figure 4. (a) Performance results for λ/μ when $s = 1$; (b) the impact from change of d ($s = 1$, $\lambda = 0.1$)

4.1.3 Experiment 3

With a larger search range (larger d), there are more context information types available within the range. On the other hand, a larger d also leads to larger size of the Bloom filter. In this set of experiments, we would like to see the impact of d.

We set the depth of the Bloom filter, d, from 3 to 10, and compare the performance with different values of s and λ (fixed $\mu = 0.1$). The results show that, in general, the Bloom filter algorithm has better performance than complete and non advertisement algorithms. There is a limit to the number of services and the query rate for which the algorithm has the best performance. When exceeding that limit, the performance of Bloom filter becomes worse. When the number of services within range (this depends on both d and s) and the query rate is quite high, the cost of using the Bloom filter algorithm increases significantly. For instance, this happens when $s = 4$, $\lambda = 50000$, and $d > 7$. Figure 4b shows the curve when $s = 1$, $\lambda = 0.1$.

4.1.4 Experiment 4

From the experiment above, we find that the number of services per node also has some influence on the network cost. In this set of experiments, we would like to investigate it in detail. We do this for fixed d and λ. The results show that there is some influence from s, but not much. When s is increasing from 1 to 6, i.e., the number of context sources increases from 0.0007 to 0.0042 per m^2, Bloom filter still has the best result among three alternative algorithms. The network cost of using a Bloom filter increases only a little bit faster than the complete advertisement algorithm. We can expect the Bloom filter algorithm to perform worse when s is really large, which will seldom happen in the reality (for given d and λ). Figure 5 shows the curve for three alternative algorithms when $d = 5$, $\lambda = 0.1$.

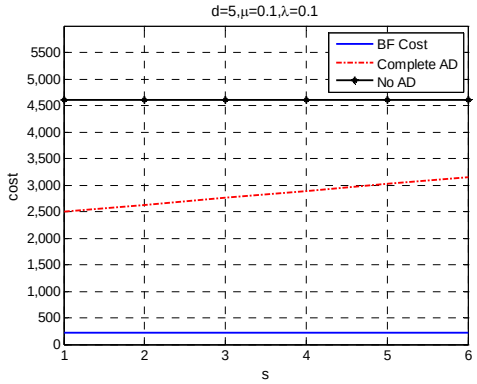


Figure 5. The impact from change of s ($d = 5$, $\lambda = 0.1$)

4.2 Tree Structure

In this section, the experiments for the tree structure are done.

4.2.1 Experiment 1

In this experiment, similar to the grid structure, we assume query rate $\lambda = 0.1$, s =1. For each given value of depth of filter, the similar experiment results, shown in Table 3, are obtained.

Table 3. Optimal BF cost for certain depth of filter compared with complete and non advertisment

d	w (bit)	b	BF cost (bit/s)	Complete Advertisement (bit/s)	Non Advertisement (bit/s)	Maximum number of services in BF
3	96	5	94	790	2995	13
5	352	3	348	7357	27878	131
7	1120	2	1926	66454	251827	1103
10	3808	1	82683	1795059	6802330	29534

As we see from Table 3, there exists a certain value of w and b which leads to the minimum network cost, even comparing with the cases of "complete advertisement" and "non advertisement". The difference is getting larger when query range d is increasing. The last column shows the maximum number of services that are represented in one Bloom filter for the related size of Bloom filter.

4.2.2 Experiment 2

This experiment is going to discuss the suitable range of using Bloom filters for context discovery to achieve the minimum network cost in the grid structure.

Similar with the Experiment 2 in the grid structure, we set μ as a reference, and change the value of λ. The experiments show that the suitable range of λ/μ decreases when the depth of the filter, d, increases. When each node has only one service ($s = 1$), to achieve better performance than the other two cases, the proper range for λ/μ of Bloom filters is decreasing as d is increasing. When $d = 3$, the Bloom filter algorithm performs better than the complete advertisement algorithm even if λ/μ is a value between 0.1 to above 10^8. When $d = 10$, this range decreases to (0.00001, 20). The Figure 6a shows the situation when $d = 5$. In here we have the range for λ/μ of 10^{-2} to 10^7.

The conclusion can be drawn that in practical situations the Bloom filter algorithm has a better performance.

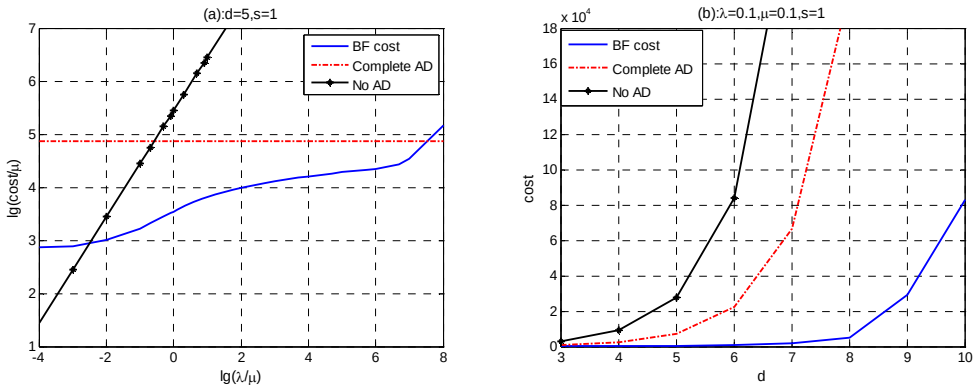


Figure 6. (a) Performance results for λ/μ when $s = 1$; (b) the impact from change of d ($s = 1$, $\lambda = 0.1$)

4.2.3 Experiment 3

In this set of experiments, we would like to see the impact of d in the tree structure network. We set the depth of the Bloom filter, d, from 3 to 10, and compare the performance with different value of s and λ (fixed $\mu = 0.1$). The results show that in general, the Bloom filter algorithm has better performance than complete and non advertisement algorithms. The limit to the number of services and the query rate for which the algorithm has the best performance is also found in the tree structure. When exceeding that limit, the performance of Bloom filter is getting worse. When the number of services within range (depends on both d and s) and the query rate is quite high, the cost of using the Bloom filter algorithm

increases significantly. For instance, this happens when $s = 4$, $\lambda = 10$, and $d > 5$. Figure 6b shows the curve when $s = 1$, $\lambda = 0.1$.

4.2.4 Experiment 4

In this set of experiments, we would like to investigate the impact from s for the tree structure in detail. We do this for fixed d and λ. The results show that there is some influence from s, but not much. When s is increasing from 1 to 6, Bloom filter still has the best result among three alternative algorithms. The network cost of using a Bloom filter increases only a little bit faster than the complete advertisement algorithm. We can expect the Bloom filter algorithm to perform worse when s is really large, which will seldom happen in the reality (for given d and λ). Figure 7 shows the curve for three alternative algorithms when $d = 5$, $\lambda = 0.1$.

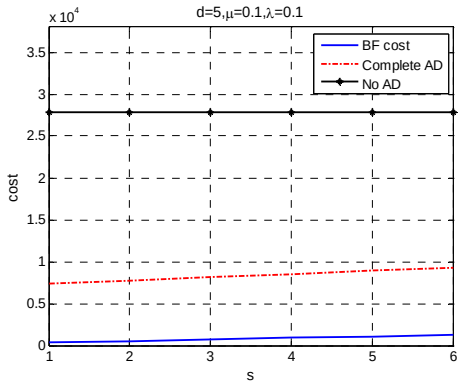


Figure 7 The impact from change of s ($d = 5$, $\lambda = 0.1$)

4.3 Circle Structure

The following sets of experiments are done in the circle structure.

4.3.1 Experiment 1

Similar to the two structures above, in this experiment, we aim to obtain the minimum network cost with a certain value of w and b for each given value of depth of filter. We set query rate $\lambda = 0.1$, s =1. The result is shown in Table 4.

Table 4. Optimal BF cost for certain depth of filter compared with complete and non advertisment

d	w (bit)	b	BF cost (bit/s)	Complete Advertisement (bit/s)	Non Advertisement (bit/s)	Maximum number of services in BF
3	64	5	79	547	1382	9
5	128	4	145	2006	3840	33
7	224	4	296	4438	7526	73
10	480	4	779	9910	15360	163

As we see from Table 4, the same conclusion as for the grid and tree structures can be drawn that for each depth of the filter, the proper width and number of hash functions leads to a minimum network cost which is much lower than for the cases of "complete advertisement" and "non advertisement". The difference is getting larger when query range d is increasing. Further, the last column shows the maximum number of services that are represented in one Bloom filter for the related size of Bloom filter.

4.3.2 Experiment 2

This experiment is going to discuss the suitable range of using Bloom filters for context discovery to achieve the minimum network cost. The comparisons are done for d is a value between 3 to 10.

As before, we observe the value of λ/μ with μ as a reference. The experiment results show that the suitable range of λ/μ decreases when the depth of the filter, d, increases. When each node has only one

service (s = 1), the Bloom filter context discovery algorithm performs better than the non advertisement algorithm when λ is at least 0.1 times μ. The Bloom filter algorithm performs better than the complete advertisement algorithm even if λ is 10^8 times μ. The Figure 8a shows the situation when d = 5.

When each node has 4 services (s = 4), the network requires larger Bloom filters to contain more information. The results show that for d = 5, the proper range of λ/μ is (0. 1, 500000); for d = 10, the proper range of λ/μ is (0.1, 500). The Figure 8b shows the result when d = 5.

We found that in practical situations the Bloom filter algorithm has a better performance. It is a promising algorithm for mobile ad hoc networks. Note that the axes in Figure 8 are represented in log scale.

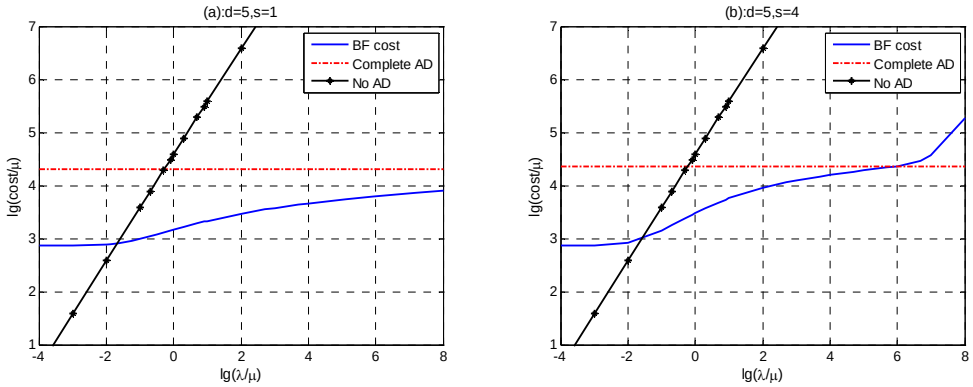


Figure 8. Performance results for λ/μ when s = 1 (a) and s = 4 (b)

4.3.3 Experiment 3

In this set of experiments, the impact of d for circle structure network is observed. We set the depth of the Bloom filter, d, from 3 to 10, and compare the performance with different value of s and λ (fixed μ = 0.1). The results show that in general, the Bloom filter algorithm has better performance than complete and non advertisement algorithms. There is an upper limit to the number of services and the query rate for which the algorithm has the best performance. When exceeding that limit, the performance of Bloom filter is getting worse. When the number of services within range (depends on both d and s) and the query rate is quite high, the cost of using the Bloom filter algorithm increases significantly. For instance, this happens when s = 4, λ = 20, and d > 9. Figure 9a shows the curve when s = 1, λ = 0.1.

4.3.4 Experiment 4

In this set of experiments, we investigate the influence on the network cost from changing the number of services per node. We fix the value of d and λ. The results show that there is some influence from s, but not much. When s is increasing from 1 to 6, i.e., the number of context sources increases from 0.0007 to 0.0042 per m^2, Bloom filter still has the best result among three alternative algorithms. The network cost of using a Bloom filter increases only a little bit faster than the complete advertisement algorithm. We can expect the Bloom filter algorithm to perform worse when s is really large, which will seldom happen in the reality (for given d and λ). Figure 9b shows the curve for three alternative algorithms when d = 5, λ = 0.1.

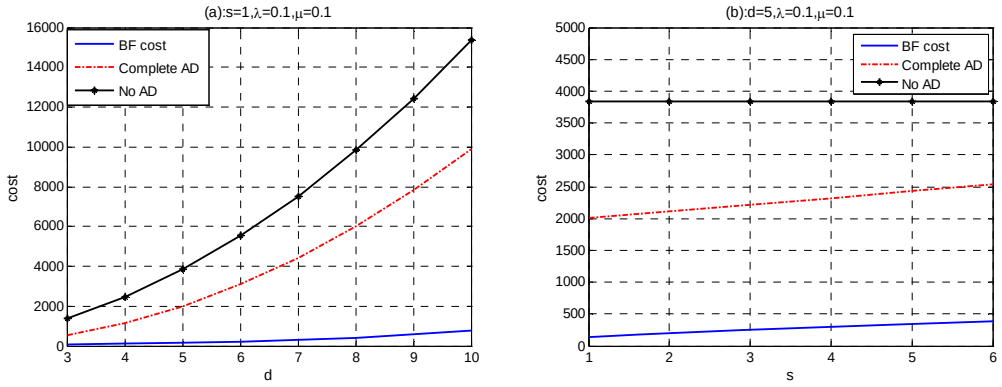


Figure 9. (a): the impact from change of d (s = 1, λ = 0.1); (b): the impact from change of s (d = 5, λ = 0.1)

4.4 Summary

We started the experiments from the grid structure. Based on it, we developed the tree structure and the circle structure. Compared with the grid and circle structures, we observe that the absolute network costs for the tree structure are higher than the other two under the same conditions, especially, when filter depth d increases. This can be explained by the exponentially growth of the number of nodes or context types within reach by source nodes (multi-hop considered) with the growth of d (compare with the Table 2, 3, and 4). In reality, the tree structure barely exists.

We obtained very close results from the grid and circle structures (compare with Figure 4a and 8a, Figure 4b and 9a). With the increase of d, the network cost of the circle structure increases slightly slower than the grid structure. This can be explained by the fact that a slightly lower number of services has to be represented in the Bloom filters. This can lead to the use of smaller Bloom filters (see Table 2 and 4) or to fewer false positives. The grid structure is an ideal case. The radio coverage of a mobile terminal is often considered as a circle. Therefore, the circle structure is a more realistic situation.

5 Conclusions and Further Work

The use of attenuated Bloom filters for advertising available context types in ad-hoc networks is very promising. We observed the performance of the model in three different network structures: a simple grid network, a tree network with high density of nodes, and a circle network which represents a fully distributed ad hoc network. Results show the combined cost of advertising and doing unsuccessful queries due to false positives. The same conclusion can be drawn from all three structures, namely that there exists an optimum size of Bloom filters to achieve optimal network cost. The performance of Bloom filters also highly depends on the ratio of query and advertisement rates, and query range of nodes. The density of network context information sources also has some influences. Especially, in a fully distributed ad hoc network in practical situations, this approach requires significantly less traffic load than advertising a full map of all available context types, or broadcasting queries when no advertisements are used. As such, it is a very promising compromise between these two extremes.

Future research should include further refinement of the model, e.g., finding a more thorough investigation of parameters and scenarios. A next step is to develop the idea further, by specifying a protocol, and testing this in a detailed, discrete event simulator and/or prototype. Security issues are also subject to future research. Finally, an interesting idea to explore is to use the broadcasting of attenuated Bloom filters to execute directed route requests (instead of undirected broadcasts) for ad hoc routing protocols such as AODV. Such an approach would allow ad-hoc nodes to establish routes only to other nodes with relevant context information, rather than establishing multiple routes first, and then finding out where the relevant context information is.

Acknowledgements

This work is part of the Freeband AWARENESS project (http://awareness.freeband.nl). AWARENESS stands for Context AWARE mobile NEtworks and ServiceS. The Freeband AWARENESS project focuses on service and network infrastructures that are needed to support context aware and pro-active applications on heterogeneous mobile networks. Particular attention is paid to mobile health applications for tele-monitoring and tele-treatment. The following organizations participate in Freeband AWARENESS: Lucent Technologies, Centre for Telematics and Information Technology, Telematica Instituut, Roessingh R&D, Ericsson, Twente Institute for Wireless and Mobile Communications, Yucat and TMS International. We would also like to thank Patrick Goering for his helpful comments.

References

[1]. Fei Liu, Geert Heijenk, "Context Discovery Using Attenuated Bloom filters", in *proceedings of Fourth International Conference, Wired/Wireless Internet Communications 2006, Bern, Switzerland*, May 2006.

[2]. Wikipedia. Context and service — wikipedia, the free encyclopedia, 2006.

[3]. Jeroen Hoebeke, Ingrid Moerman, Bart Dhoedt, Piet Demeester, "Anaylsis of Decentralized Resource and Service Discovery Mechanisms in Wireless Multi-hop Networks", *Third International Conference, Wired/Wireless Internet Communications 2005 Xanthi, Greece,* May 2005, Proceedings.

[4]. R. Marin-Perianu, P. H. Hartel, J. Scholten, "A Classification of Service Discovery Protocols", *Centre for Telematics and Information Technology, Univ. of Twente, The Netherlands*, Technical report nr. TR-CTIT-05-25, June 2005.

[5]. E.Guttman, C. Perkins, J. Veizades, M. Day, "Service Location Protocol version 2", *IETF, RFC 2608*, June 1999.

[6]. Bluetooth Consortium, "Specivication of Bluetooth System Core Version 1.0b: Part e, Service Discovery Protocol (SDP)", November 1999

[7]. Knarig Arabshian, Henning Schulzrinne, "GloServ: Global Service Discovery Architecture", *MobiQuitous*, pages 319-325, IEEE Computer Society, June 2004

[8]. Diapanjan Chakraborty, Anupam Joshi, Tim Finin, Yelena Yesha, "GSD: a Novel Group-based Service Discovery Protocol for MANETs", *4th IEEE Conference on Mobile and Wireless Communication Netowrks (MWCN)*, September 2002

[9]. Sun Microsystems, "Jini Architecture Specification Version 2.1", Nov 2005

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

[11]. Andrei Broder, Michael Mitzenmacher, "Network Applications of Bloom Filters: A Survey", *Internet Math 1(2003)*, no.4, 485-509

[12]. S. E. Czerwinski, B. Y. Zhao, T. D. Hodes, A. D. Joseph, R. H. Katz, "An Architecture for a Secure Service Discovery Service", In *Proc. Of MobiCom-99*, pages 24-35, N. Y., August 1999.

[13]. E. Papapetrou, E. Pitoura, and K. Lillis, "Speeding –up Cache Lookups in Wireless Ad-Hoc Routing Using Bloom Filters", *the 16th Annual International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC 2005)*, Sept 11-13, 2005, Berlin, Germany, 2005.

[14]. J. Liu, F. Sailhan, D. Sacchetti, V. Issarny. "Group Management for Mobile Ad hoc Networks: Design, Implementation and Experiment", in *Proceedings of the 6th IEEE International Conference on Mobile Data Management (MDM'2005)*, May 2005.

[15]. Yi-an Huang, Wenke Lee, "Hotspot-Based Traceback for Mobile Ad Hoc Networks"., in *Proceedings of the ACM Workshop on Wireless Security (WiSe'05)*, September 2005.

[16]. James F. Kurose, Keith W. Ross, "Computer Networking: A Top-Down Approach Featuring the Internet*", Addison Wesley Longman, Inc*. ISBN 0 2-1 47711 4, 2001