SECURE SERVICE DISCOVERY PROTOCOL FOR AD HOC NETWORKS USING HASH FUNCTION

[1]Haitham Elwahsh, [2]Mohamed Hashem, [3]Mohamed Amin

[1]Computer Science, King Saud University
Haitham.elwahsh@gmail.com

[2]Professor of Information Systems, Faculty of Computer & Information Systems,
Ain shams University,
mhashem100@yahoo.com

[3]Assistant Professor of Mathematics, Faculty of Science,
Minoufiya University
mohamed_amin110@yahoo.com

Abstract. *Ad-hoc networks, mobile devices communicate via wireless links without the aid of any fixed networking infrastructure. These devices must be able to discover services dynamically and share them safely, taking into account ad-hoc networks requirements such as limited processing and communication power, decentralized management, and dynamic network topology, among others. Legacy solutions fail in addressing these requirements. In this paper, we propose a service discovery protocol with security features, the Secure Pervasive Discovery Protocol. $SPDP_h$ is a fully distributed protocol in which services offered by devices can be discovered by others, without a central server. It is based on One Way hash Chains, as well as protection of confidential information, secure communications, or access control and compared this with Pervasive Discovery Protocol PDP.*

Keywords: *ad-hoc networks, service discovery protocol, hash chains, security*

1 Introduction

Recent advances in microelectronic and wireless technologies have fostered the proliferation of small devices with limited communication and processing power. They are what are known as "pervasive systems". Personal Digital Assistants (PDAs) and mobile phones are the more "visible" of these kinds of devices, but there are many others that surround us, unobserved. For example, today most household appliances have embedded microprocessors. Each one of these small devices offers a specific service to the user, but thanks to their capacity for communication, in the near future they will be able to collaborate with each other to build up more complex services. In order to achieve this, devices in such "ad-hoc" networks should dynamically discover and share services between them when they are close enough. In ad-hoc networks composed of limited devices, it is very important to minimize the total number of transmissions, in order to reduce battery consumption of the devices. It is also important to implement mechanisms to detect, as soon as possible, both the availability and unavailability of services produced when a device joins or leaves the network. Security in these networks is also critical because there are many chances of misuse both from fraudulent servers and from misbehaving clients. In this paper, we propose a service discovery protocol with security features, the Secure Pervasive Discovery Protocol (SPDP). SPDP is a fully distributed protocol in which services offered by devices can be discovered by others, without a central server. It provides location of trusted services, as well as protection of confidential information, secure communications, identification between devices, or access control, by forming a reliable ad-hoc

DOI : 10.5121/ijcnc.2012.4410

network. The paper is organized as follows: section 2 enumerates the main service discovery protocols proposed so far in the literature, we will see that none of them adapts well to ad-hoc networks. Section 3 presents secure pervasive discovery protocol, SPDP, with its application scenario, and description of the algorithm. In section 4 we present the comparing of SPDP with service discovery Protocols. In section 5 we describe the One Way hash Chains as security support. In section 6 presents the simulation environment & results comparing $SPDP_h$ with other PDP Protocol. Finally, we conclude with some conclusions.

2 Related Works

Dynamic service discovery is not a new problem. There are several solutions proposed for fixed networks, with different levels of acceptance, like SLP [RFC2608, 1999] [1], Jini [Sun, 1999] [4] and Salutation [Miller and Pascoe, 2000] [5]. More recently, other service discovery protocols, specifically designed for ad-hoc networks, have been defined, some tied to a wireless technology (SDP for Bluetooth [SDP, 2001] [6], IAS for IrDA [IrDA, 1996]) [7], others that jointly deal with the problems of ad-hoc routing and service discovery (GSD [Chakraborty et al., 2002] [8], HSID [Oh et al., 2004]) [9], and others that work at the application layer of the protocol stack (DEAPspace [Nidd, 2001] [12], Konark [Helal et al., 2003] [13], and the post-query strategies [Barbeau and Kranakis, 2003]) [14]. Only a few protocols have built-in security, the most important are SSDS [Czerwinski et al., 1999] [16]and Splendor [Zhu et al., 2003]. However, these solutions cannot be directly applied to an ad-hoc network, because they were designed for and are more suitable for (fixed) wired networks. We see three main problems in the solutions enumerated: – First, many of them use a central server, such as SLP2, Jini and Salutation. It maintains the directory of services in the network and it is also a reliable entity upon which the security of the system is based. An ad-hoc network cannot rely upon having any single device permanently present in order to act as central server, and furthermore, maybe none of the devices present at any moment may be suitable to act as the server. – Secondly, the solutions that may work without a central server, like SSDP, are designed without considering the power constraints typical in wireless networks. They make an extensive use of multicast or broadcast transmissions which are almost costless in wired networks but are power hungry in wireless networks. – Thirdly, security issues are not well covered. SSDS provides security in enterprise environments but may not work in ad-hoc networks with mobile services. Splendor does not provide certificate revocation and trust models of PKIs. They both depend on trustworthy servers and they propose solutions which are provided at the IP level. Accepting that alternatives to the centralized approach are required, we consider two alternative approaches for distributing service announcements: – The “Push” solution, in which a device that offers a service sends unsolicited advertisements, and the other devices listen to these advertisements selecting those services they are interested in. – The “Pull” solution, in which a device requests a service when it needs it, and devices that offer that service answer the request, perhaps with third devices taking note of the reply for future use. In ad-hoc networks, it is very important to minimize the total number of transmissions, in order to reduce battery consumption. It is also important to implement mechanisms to detect as soon as possible both the availability and unavailability of services produced when a device joins or leaves the network. These factors must be taken into account when selecting between a push solution and a pull solution. The DEAPspace algorithm is the only service discovery protocol, listed above, that tries to minimize the total number of transmissions. It uses a pure “push” solution and each device periodically broadcast its “world view” although none of them has to request a service.

3 SPDP: Secure Pervasive Discovery Protocol

In this paper we propose a new service discovery protocol, the Secure Pervasive Discovery Protocol (SPDP), which merges characteristics of both pull and push solutions to improve the performance of the protocol. Also, SPDP provides security based on an anarchy trust management model. Such trust management model does not require neither a central trusted server nor a hierarchical architecture, so it is suitable to overcome the challenges imposed by ad-hoc networks such as no central management, no strict security policies and highly dynamic nature. The Secure Pervasive Discovery Protocol (SPDP) is intended to solve the problem of enumerating the services available in ad-hoc networks, composed of devices with limited transmission power, memory, processing power, etc. Legacy service discovery protocols use a centralized server that listens for broadcast or multicast announcements of available services at a known port address, and lists the relevant services in response to enquiries. The protocol we propose does away with the need for the central server. Ad-hoc networks cannot rely upon having any single device permanently present in order to act as central server, and further, none of the devices present at any moment may be suitable to act as the server. One of the key objectives of the SPDP is to minimize battery use in all devices. This means that the number of transmissions necessary to discover services should be reduced as much as possible. A device announces its services only when other devices request the service. Service announcements are broadcasted to all the devices in the network, all of which will get to know about the new service simultaneously at that moment, without having to actively query for it. In addition, SPDP allows sharing services safely, through an underlying trust management model between devices, which allows us to store service information from other "alleged" trusted service agents and later to use them if such information is really authentic and pright. Currently, the security support provided by service discovery protocols are focused on authentication, integrity, and confidentiality [RFC2608, 1999] [Czerwinski et al., 1999] [Zhu et al., 2003]. Even more, some of them include authorization services as part of the discovery [Zhu et al., 2003]. Such support is based on IPSec [Kent and Atkinson, 1998] or traditional PKI in the last case. However, these security services could be not necessary for the discovery, but they could cause energy and processing consumption. Protecting both energy and processing consumption is a very essential issue for devices with limited capabilities. So we have considered providing basic security services to prevent certain attacks (i.e. DoS, false announcements, and false services) and to avoid the sending of unnecessary messages. In the remainder of this section, we present the application scenario for SPDP and some considerations to be taken into account. Then, we will formally describe the algorithm used to implement it.

3.1 Application Scenario

Let's assume that there is an ad-hoc network, composed of D devices, each device offers S services, and expects to remain available in this network for T seconds. This time T is previously configured in the device, depending on its mobility characteristics. Each device has an SPDP User Agent (SPDP UA) and an SPDP Service Agent (SPDP SA). The SPDP UA is a process working on the user's behalf to search information about services offered in the network. The Service Agent SPDP (SPDP SA) is a process working to advertise services offered by the device. The SPDP SA always includes the availability time T of its device in its announcements. Each device has a cache associated which contains a list of the services that have been heard from the network. Each element e of the cache associated to the SPDP UA has three fields: the service description, the service lifetime and the service expiration time. The service expiration time is the time it is estimated the service will remain available. This time is calculated as the minimum of two values: the time the device has promised to remain available, and the time the server announced that the service would remain available. Entries remove themselves from the cache when their timeout elapses. With regard to security, each

device handles a list of reliable devices and the trust degree associated with them. Trust helps devices to limit their cache size; services from untrusted devices are not stored in the cache. Depending on the trust degree, a device decides to store the service offered by a device on its cache. When the devices access services, devices with biggest trust degree are selected in the first place.

3.2 Algorithm description

The SPDP has two mandatory messages: SPDP Service Request, which is used to send service announcements and SPDP Service Reply, which is used to answer a SPDP Service Request, announcing available services. SPDP has one optional message: SPDP Service Deregister, which is used to inform that a service is no longer available. Now, we will explain in detail how SPDP UA and SPDP SA use these primitives.

3.2.1 SPDP User Agent

When an application or the final user of the device needs a service of a certain type, it calls its SPDP UA. In order to support different application needs, in SPDP we have defined two kinds of queries:

– *one query–one response (1/1):* the application is interested in the service, not in which device offers it.

– *one query–multiple responses (1/n)*: the application wants to discover all devices in the network offering the service. In this kind of query, we introduce a special type of service, named ALL, in order to allow an application to discover all available services of all types in the network.

3.2.2 SPSP Service Agent

The SPDP SA advertises services offered by the device. It has to process SPDP Service Request messages and to generate the corresponding SPDP Service Reply, if necessary. In order to minimize the number of transmissions, the SPDP SA takes into account the type of query made by the remote SPDP UA.

4 Evaluating the SPDP protocol

In this section we present a performance evaluation study of SPDP in a ubiquitous computing environment. We compare our protocol with the theoretical distributed approaches, push and pull; because all the service discovery protocols defined in the literature are based on one of these approaches; and also we compare PDP with the service discovery protocol standard in Internet, SLP, and with UPnP's SSDP. This study was carried out through simulation using the well-known network simulator,NS-2. Our simulator is available in [Campo and Perea, 2004]. During the simulation, devices join the ubiquitous environment at random times, request and offer random services, and leave the network after a random time. The number of devices in the network varies over time, but its mean remains stationary. Random times follow exponential distributions, while random services follow uniform distributions. For simplicity we assume that each device offers just one service. The parameters of the simulation are**: the mean number of devices**, **the mean time they remain available in the network**, **the size of the caches**, **the mean time between service requests**, and **the total number of service types**. The

results of interests are: the number of messages (the number of messages transmitted in the network normalized to the number of service request), the service discovery ratio (the ratio of services discovered to the total number of services available in the network) and the error ratio (the ratio of services discovered that were not available in the network to the total number of services discovered). Figure 1 shows the number of messages transmitted, the service discovery ratio and the error ratio, in a scenario with 20 devices, an average device life time ranging from 600 to 19200 seconds, a cache size of 100 entries, 5 different types of services, and each device requesting a random service every 60 seconds. The SPDP number of messages is quite under those obtained for SLP and for pull solutions, while keeping the same service discovery ratio and error rate of them.

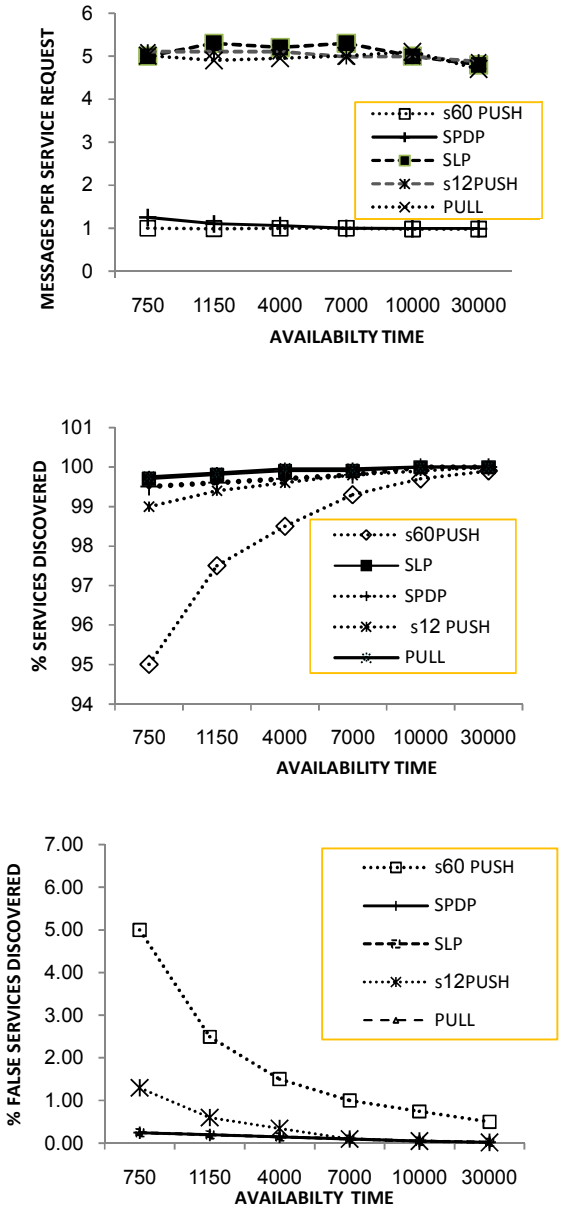


Figure 1. Comparison of SPDP with others protocols

(Figure. 2) shows the global power consumption in the same scenario as before. We see that, despite of using broadcast transmissions instead of unicast, and despite of sending bigger service requests (with the services already known from the cache), PDP achieves an important reduction in the power consumed, that is a reflection of the reduction of messages transmitted, and also of the lower energy cost associated with receiving than with transmitting. Only SSDP with an announcement period of "60 s" achieves less consumption, but at the cost of a higher error and lower service discovery percentages. Furthermore, as we will show later, PDP preserves the battery of the more limited devices, while the other protocols equally deplete the batteries of all devices. Regarding the delay in service discovery, it depends on the way the service discovery is done. In push mode protocols, the answer is obtained immediately from the cache. In directory-based protocols, the delay is the associated with transmitting a service request message to the directory, the processing time the directory needs to obtain the answer from its services database, and the transmission time associated with sending the reply. In pull mode protocols, as well as in Multicast DNS and PDP, the device broadcasts a service request (perhaps consulting first its local cache), and then it must wait for answers to come during a given period of time.

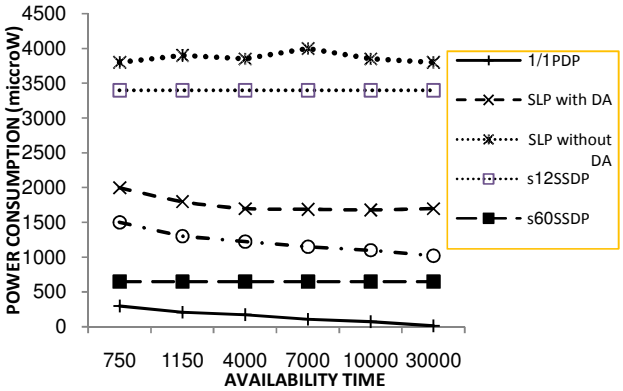


Figure 2 . Comparison of power consumption.

Now, we will study the impact of the number of devices in the network and the cache size in the performance of PDP. A PDP with cache 0 is equivalent to a pull mode.(Figure. 3) shows that if the cache size is big enough, the number of messages transmitted remains constant, since all the services are already known and stored in the cache. For small cache sizes, when the number of devices equals the cache size, the number of messages starts growing linearly. For cache 0 (pull mode) the increment is always linear. Now, we will demonstrate how PDP takes into account device heterogeneity, achieving a reduction of traffic transmission (and so power consumption) in the more limited devices.(Figure. 4) shows the percentage of replies sent by each kind of devices depending on its availability time. We have considered a scenario with 40 devices in mean, with five different availability times: 500, 2500, 4500, 6500 and 9500 s, with about 20% of devices (in mean) of each type. The rest of parameters of the simulation are the same as before, except that the cache size for devices with availability time 500 and 2500 is 10 services, while for devices with availability time 4500 and 6500 is 40 services and for devices with availability time 9500 is 100 services. This way we simulate that devices that move more frequently (PDAs, mobile phones) have less memory than devices that move less frequently (laptops or desktop computers). In (Figure. 4) we see that devices with greater availability time answer more requests, preserving power consumption of devices with smaller availability in the figure sum up 70%, because in PDP some requests generate no replies (all known services were already included in the request). Considering this, devices with availability of 9500 s answer almost 50% of the service requests. If other service discovery protocol were

used, all devices would answer with equal probability, 20%. This means that with PDP, fixed devices with greater availability time and less limitations answer most of the requests. This was one of the objectives of our protocol.(Figure. 4) also shows that devices with very small availability time (in our case, 500 s) answer more requests than devices with middle availability times. This is because these devices are highly mobile, continually change of networks, and in each new environment they arrive, they have to answer requests above their own services, to make them known to the rest of the devices. As we know, PDP is a fully distributed protocol, it does not rely in any central directory. However, with this simulation we show that PDP is designed time. It is worth mentioning that all percents shown

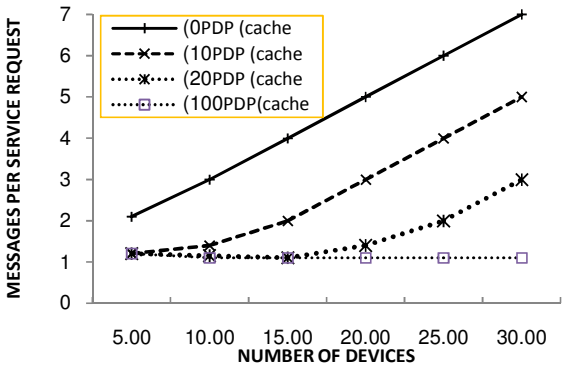


Figure 3 . Service replies per search for different number of devices.

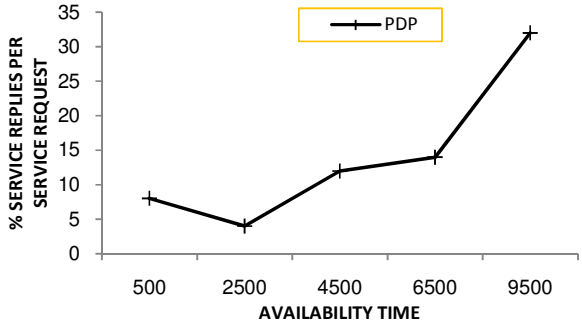


Figure 4. Service replies per search in an heterogeneous environment with PDP.

in such a way that, if there are devices that are less mobile (remain more time in the environment) and that have more memory, most of the queries will be answered by them, relieving the more mobile and limited ones of answering, and so preserving their battery. In this scenario we assume that devices with higher availability time also have greater cache's sizes. This is a realistic assumption, since fixed devices use to have more memory than mobile (small, battery powered) devices All the above figures considered PDP one query– multiple responses queries. If the application is interested in the service, not in which device offers it, PDP one query–one response (1/1) can be used instead, obtaining a further reduction in number

of messages and power consumption. (Figure. **5)** compares PDP one query–one response against the same service discovery protocols as before.

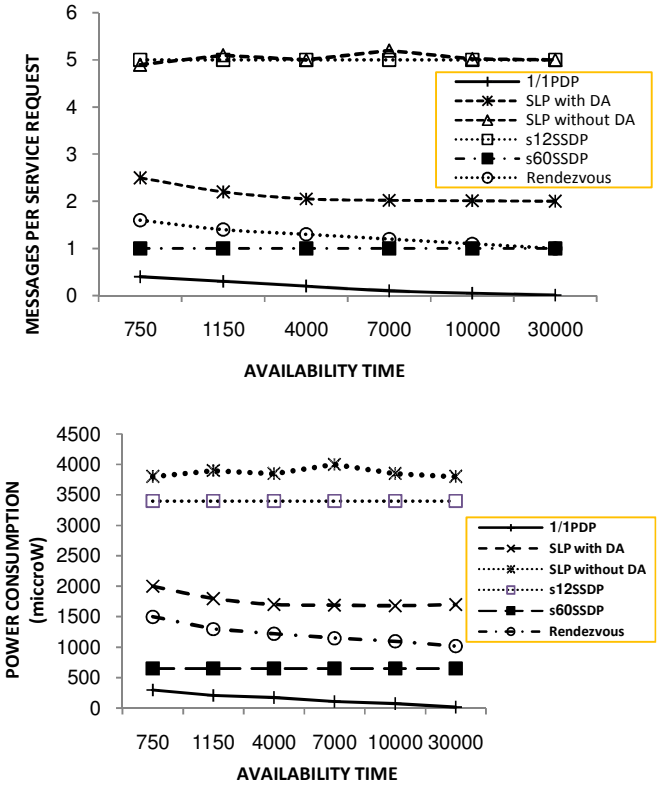


Figure 5. Comparison of PDP 1/1 with other protocols

5 One-way hash chains

Consider the example as shown in the figure 6 (a) and table 1 which illustrates the dynamic network topology and table entry updates. The table entries for node (1) as shown in tables 1 (a), 'Dest' is the destination node, with node (1) as the source node. Next node is the next neighbor node, to node (1) for that corresponding destination node. The table entry distance is measured as a hop count, (i.e. how many nodes are in between the source and destination node including the destination node). The distance metric shows the length for that destination node. The sequence number entry in the table corresponds to the number encapsulated in each table update message. At a given time, all nodes try to keep the table entries with the highest known sequence number. Figure 6 (a) shows ad-hoc network with 12 nodes participating in the network. Let us consider that node (1) is the source node and node 12 is the destination node. Table 1 (a) shows the table entries of the node (1) and the shortest routes to all other nodes in the network. The nodes communicate with each other if there is a change in the location of a particular node. The changed node sends table update message with an increased sequence number. The neighbor nodes propagate the update message to the whole network with an increased sequence number. At any given time, all the nodes try to keep the latest information with respect to the dynamic topology of the network. Figure 6 (b) shows the change in location of the node (9) in the example network shown in figure 6 (a). Table 1 (b) shows the table entries for the node (1) with respect to the change in network topology.

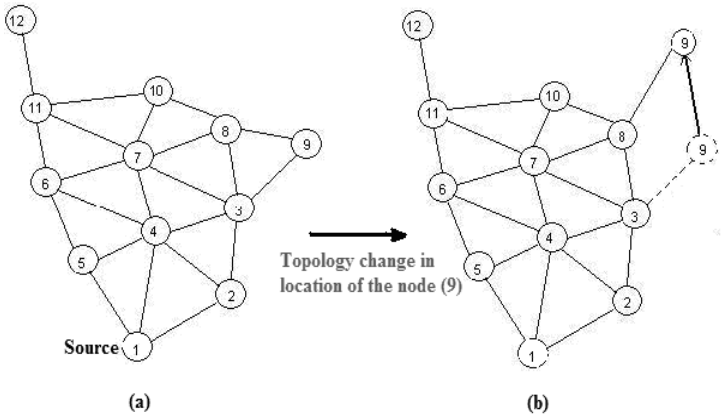


Figure 6 (a) Ad-hoc network with 12 nodes. (b) Ad-hoc network with change in location of the node 9.

Dest	Next node	Distance	Sequence no
2	2	1	16
3	2	2	32
4	4	1	38
5	5	1	45
6	5	2	89
7	4	2	98
8	2	3	111
9	2	3	118
10	4	3	186
11	5	3	189
12	5	4	192

(a)

Dest	Next node	Distance	Sequence no
2	2	1	16
3	2	2	32
4	4	1	38
5	5	1	45
6	5	2	89
7	4	2	98
8	3	3	111
9	***3***	***4***	***146***
10	7	3	186
11	6	3	189
12	11	4	192

(b)

Table 1: (a) table for node 1, (b) table for node 1 after topology change

$SPDP_h$ used destination sequence numbers, as in PDP; we also use these destination sequence numbers to provide replay protection of update messages in $SPDP_h$. $SPDP_h$ incorporates One-Way Hash function to authenticate in the update mechanism to enhance service Protocol the security. A one-way hash chain is built on a one-way hash function. Like a normal hash function, a one-way hash function, H, maps an input of any length to a fixed-length bit string. Thus,

$$H : \{0,1\}^* \rightarrow \{0,1\}^P$$

Where **P** is the length in bits of the output of the hash function, function **H** should be simple to compute yet must be computationally infeasible in general to invert. A more formal definition of one-way hash functions has many names: compression function, contraction function, message digest, fingerprint, cryptographic checksum, message integrity check (MIC), and manipulation detection code (MDC). Whatever one calls it, it is central to modern cryptography. One-way hash functions are another building block for many protocols [36]. To

create a one-way hash chain, a node chooses a random initial value **x**, where $\mathbf{x \epsilon \{0, 1\}^P}$ and computes the list of values

$$\mathbf{h_0, h_1, h_2, h_3, h_4, \ldots\ldots h_n}$$

Where $\mathbf{h_0 = x}$, and $\mathbf{h_i = H(h_{i-1})}$ for $\mathbf{0 < i \leq n}$, for some **n**.

The node at initialization generates the elements of its hash chain as shown above, from (left to right) (in order of increasing subscript **i**) and then over time uses certain elements of the chain to secure its routing updates. In using these values, the node progresses from (right to left) (in order of decreasing subscript i) within the generated chain.

Given an existing authenticated element of a one-way hash chain, it is possible to verify elements later in the sequence of use within the chain (further to the ''left'', or in order of decreasing subscript).

For example, given an authenticated $\mathbf{h_i}$ value, a node can authenticate $\mathbf{h_{i-3}}$ by computing $\mathbf{H(H(H(h_{i-3}))}$ verifying that the resulting value equals $\mathbf{h_i}$.

To use one-way hash chains for authentication, we assume some mechanism for a node to distribute an authentic element such as $\mathbf{h_n}$ from it hash chain is generated. A traditional approach for this key distribution is for a trusted entity to sign public-key certificates for each node; each node can then use its public-key to sign a new hash chain element for itself. Let us consider '**m**' is the number of nodes in the network, so the upper bound for the hop counts is **< m-1**. Let the hash chain values calculated using **H** be (h_1, h_2,..., h_n), where **n** is divisible by **m**, then for a table entry with sequence number **i**, let **k=((n/m)-i).** If the metric **j** (distance) is used to authenticate the update entry, then $\mathbf{h_{(km+j)}}$ is used to authenticate the table update entry for that sequence number **i** and distance **j**. Where $\mathbf{0 < i \leq (m-1)}$ and $\mathbf{n=(m-1)\times m}$. The group of elements used for routing update with sequence number **i** is:

$$\mathbf{h_{km}, h_{km+1}, h_{km+2}, \ldots, h_{km+m-1}}$$

For example: seq =1 $\mathbf{h_{n-m}, h_{n-m+1}, h_{n-m+2}, \ldots, h_{n-1}}$

seq =2 $\mathbf{h_{n-2m}, h_{n-2m+1}, h_{n-2m+2}, \ldots, h_{n-m-1}}$

So that one-way hash chains, for example we use **m=5**, were i=sequence number, j= distance, m=network diameter, n=length of hash chain as shown in table 4.2 one-way hash chain elements are used for authentication in reverse order. as $0 < i \leq (m-1)$ and $n = (m-1) \times m = 20$.

Note that from the above example a node generates its hash chain so that n is divisible by m. When a node first enters the network, or after a node has used most of its available hash chain elements, it can pick a new random **x**, generate a new hash chain from this **x**, and send the new generated $\mathbf{h_n = h_{19}}$ value to a trusted entity or an alternative authentication and distribution service. Each node is tagged with the destination nodes address, next hop node address, hash value, sequence number and metric as shown in figure 7

i \ j	0	1	2	3	4
1	h_{15}	h_{16}	h_{17}	h_{18}	h_{19}
2	h_{10}	h_{11}	h_{12}	h_{13}	h_{14}
3	h_5	h_6	h_7	h_8	h_9
4	h_0	h_1	h_2	h_3	h_4

Table 2: Example of One-way hash chains

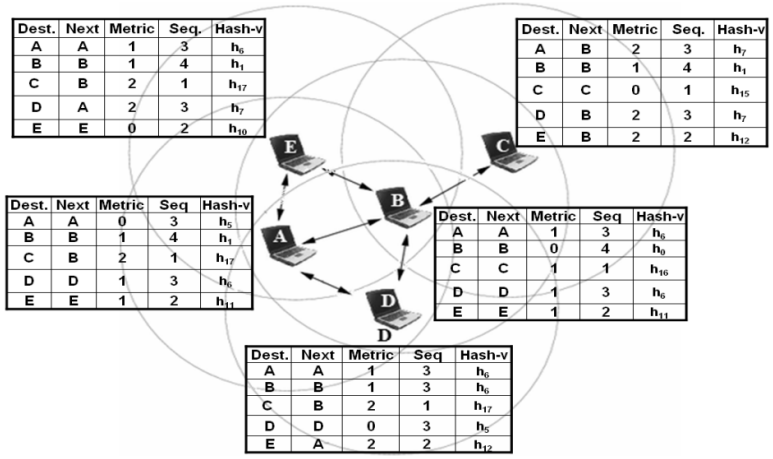


Figure 7 nodes is tagged with the destination nodes address, next hop node address, hash value, sequence number and metric

In the example given above for sequence number i, the node sets the hash value in that entry to its $\mathbf{h_{km}= h_{15}}$. If the node lists an entry for some other destination in that update, it sets the (*address in that entry to that destination nodes address, the metric and sequence number*) to the values for that destination in its table, and the its *hash value to the hash of the hash value received* in the routing update entry from which it learned that route to that destination. So that nodes receiving any update can easily authenticate each entry in the update, given any earlier authentic hash element from the same hash chain, the group of elements used for routing update with sequence number **i** is:

$$\mathbf{h_{km}, h_{km+1}, h_{km+2}, \dots , h_{km+m-1}}$$

A malicious node can modify $\mathbf{h_{(km+j)}}$ only if it knows the value of $\mathbf{h_{(km+j-1)}}$, which is impossible to calculate. So the hashing technique is used to authenticate the nodes participating in the ad-hoc network.

The attacker can never forge better distances or sequence numbers, attacker can only generate worse distances or sequence numbers, however, other information such as node name or next hop can be forged.

They differ from PDP in that they do not use an average weighted settling time in sending triggered updates. To reduce the number of redundant triggered updates, each node in PDP tracks, for each destination, the average time between when the node receives the first update for some new sequence number for that destination, and when it receives the best update for that sequence number for it (with the minimum metric among those received with that sequence number), when deciding to send a triggered update, each PDP node delays any triggered update for a destination for this average weighted settling time, in the hope of only needing to send one triggered update, with the best metric, for that sequence number. $SPDP_h$ does not use such a delay, in order to prevent attacks from nodes that might maliciously not use the delay. Since a node selects the first route it receives with highest sequence number and lowest metric as shown in figure 8, an attacker could attempt to cause more traffic to be routed through itself, by avoiding the delay in its own triggered updates. Such an attack could put the attacker in a position to eavesdrop on, modify, or discard other nodes' packets.

In addition, unlike PDP, when a node detects that its next-hop link to some destination is broken, the node does not increment the sequence number for that destination in its table when it sets the metric in that entry to infinity as shown in figure 9. Since ***higher sequence numbers take priority***, this node's routing update with this new sequence number must be authenticated, *but we did not include a mechanism for authenticating these larger sequence numbers*. Instead, the node flags its table entry for this destination to not accept any new updates for this same sequence number, effectively preventing the possible loop.

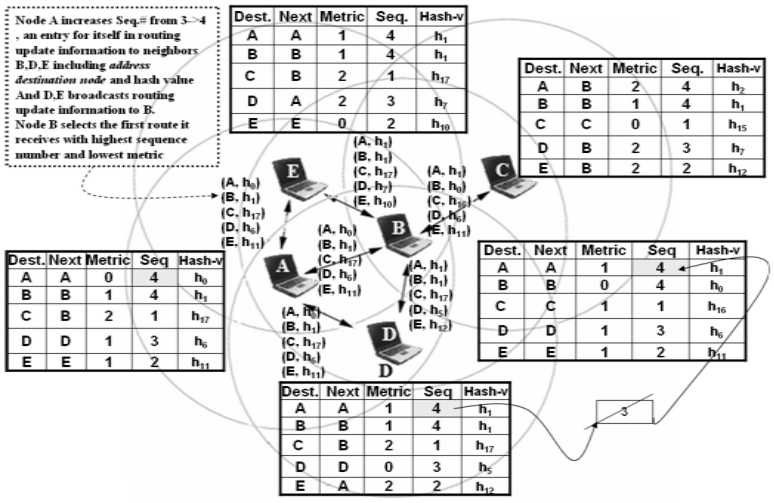


Figure 8: Ad-hoc networks with node (A) send update

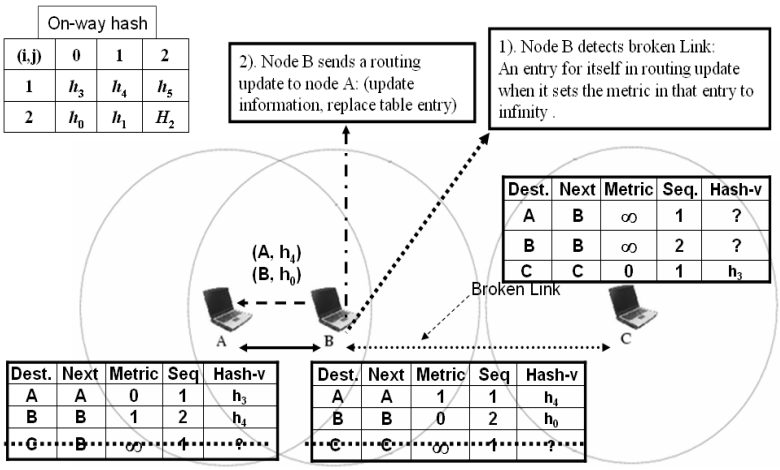


Figure 9: Ad-hoc networks with node B detect broken link

6. Simulation environment

During the simulation, devices join an ad hoc network at random times, request and offer uniformly distributed random services, and leave the network after a random time. Once in the ad hoc network, devices do not move until they leave the network. This simulates the behaviour of a user that arrives to a place, for example a conference room, uses some services from the environment or from other users' devices, offers some services while she is there, and after some time she leaves the room. In our simulation all devices can communicate with each other, either because they are all in transmission range, or because a multicast routing protocol is present.

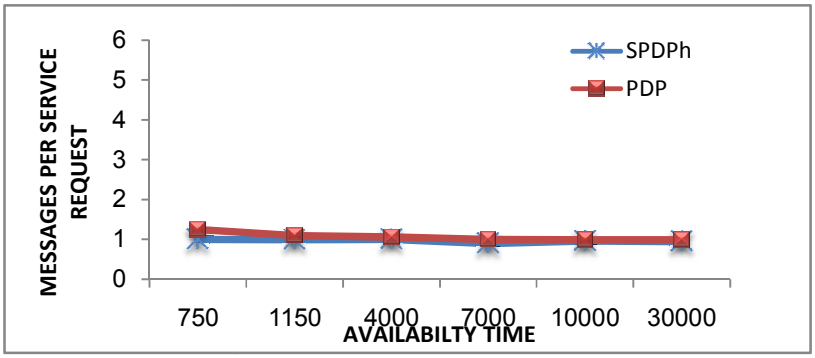
The number of devices in the network varies over time, but its mean remains stationary because new users join the network at the same rate they leave it. Random times follow exponential distributions, while random services follow uniform distributions. For simplicity we assume that each device offers just one service. The parameters of the simulation are: the mean number of devices, the total number of services types, the different kinds of devices, and for each of them, the mean time the device remain available in the network, the size of its cache, the mean time between service requests. The results of interest are: the number of messages (the number of messages transmitted in the network normalized to the number of service requested), the service discovery percentage (the percentage of services discovered to the total number of services available in the network) and the error percentage (the percentage of services discovered that were not available in the network to the total number of services discovered).

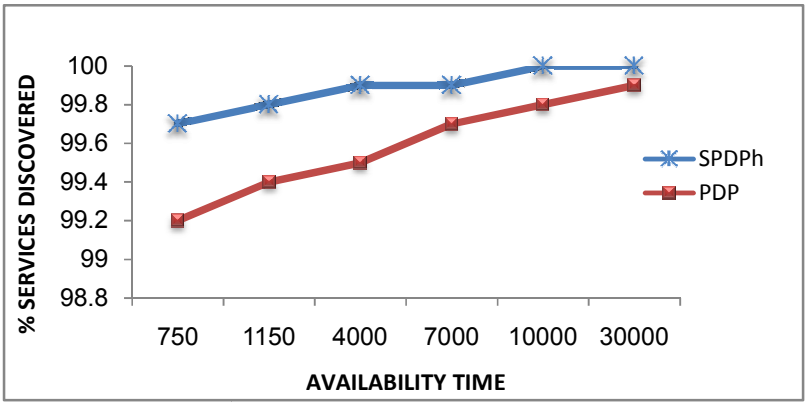
6.1. Simulation results

In this section we compare $SPDP_h$ with the service discovery protocol PDP. This study was carried out through simulation using the well-known network simulator, NS-2. Our simulator is available in [Campo and Perea, 2004]. During the simulation, devices join the ubiquitous environment at random times, request and offer random services, and leave the network after a random time. The number of devices in the network varies over time, but its mean remains stationary. Random times follow exponential distributions, while random services follow uniform distributions. For simplicity we assume that each device offers just one service. The parameters of the simulation are**: the mean number of devices**, **the mean time they remain available in the network**, **the size of the caches**, **the mean time between service requests**, and **the total number of service types**. The results of interests are: the number of messages (the number of messages transmitted in the network normalized to the number of service

request), the service discovery ratio (the ratio of services discovered to the total number of services available in the network) and the error ratio (the ratio of services discovered that were not available in the network to the total number of services discovered). Figure 10 shows the number of messages transmitted, the service discovery ratio and the error ratio, in a scenario with 20 devices, an average device life time ranging from 600 to 19200 seconds, a cache size of 100 entries, 5 different types of services, and each device requesting a random service every 60 seconds. The announcement period is a parameter that can be tuned to adapt to different needs. The $SPDP_h$ number of messages is quite under the obtained for the protocol PDP, while keeping the same service discovery percentage and error rate than PDP. The explanation for this is that in these protocols, devices broadcast service requests when they want to discover a service, so all the servers are discovered (i.e., service discovery percentage close to 100%). In Multicast DNS devices use broadcast replies, they store in a cache previously discovered services, and they send these already known services in the request, preventing these servers to answer, and so reducing the number of messages necessary quite PDP. $SPDP_h$ further reduces this number by allowing a device to answer not just with its own services but with all known services, not yet included in the request. While in $SPDP_h$, if there is no new service in the network, just a service request is needed. Thanks to the PDP_Service_Deregister messages, the $SPDP_h$ error percentage is kept close to 0%, with an almost imperceptible effect in the total number of messages.

In the figure we also show the performance of PDP discovery protocol, which equals the number of messages sent by $SPDP_h$ but with a penalty in the service discovery percentage and the error percentage. In $SPDP_h$, service announcements (the PDP_Service_Reply messages) are not periodical but triggered by the reception of a PDP_Service_Request, leading to a better performance.





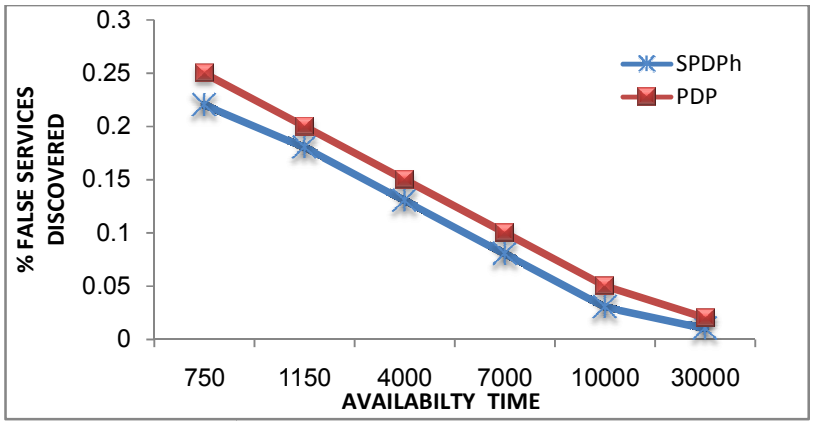


Figure 10: Comparison of $SPDP_h$ protocol with PDP protocol

(Figure 11) shows the global power consumption in the same scenario as before. We see that, despite of using broadcast transmissions instead of unicast, and despite of sending bigger service requests (with the services already known from the cache), $SPDP_h$ achieves an important reduction in the power consumed, that is a reflection of the reduction of messages transmitted, and also of the lower energy cost associated with receiving than with transmitting. $SPDP_h$ preserves the battery of the more limited devices, while the other protocols equally deplete the batteries of all devices. Regarding the delay in service discovery, it depends on the way the service discovery is done. The device broadcasts a service request (perhaps consulting first its local cache), and then it must wait for answers to come during a given period of time.

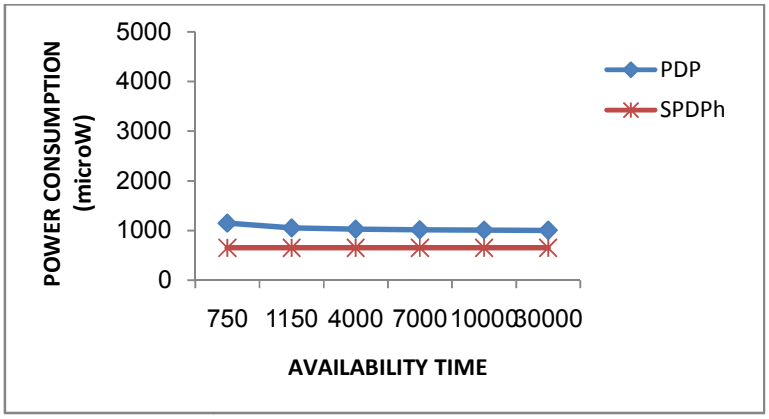


Figure. 11. Comparison of power consumption.

Now, we will demonstrate how $SPDP_h$ takes into account device heterogeneity, achieving a reduction of traffic transmission (and so power consumption) in the more limited devices.(Figure. 12) shows the percentage of replies sent by each kind of devices depending on its availability time. We have considered a scenario with 40 devices in mean, with five different availability times: 500, 2500, 4500, 6500 and 9500 s, with about 20% of devices (in mean) of each type. The rest of parameters of the simulation are the same as before, except that the cache size for devices with availability time 500 and 2500 is 10 services, while for devices with availability time 4500 and 6500 is 40 services and for devices with availability time 9500 is 100 services. This way we simulate that devices that move more frequently (PDAs, mobile phones) have less memory than devices that move less frequently (laptops or desktop computers). In (

Figure. 12) we see that devices with greater availability time answer more requests, preserving power consumption of devices with smaller availability in the figure sum up 70%, because in $SPDP_h$ some requests generate no replies (all known services were already included in the request). Considering this, devices with availability of 9500 s answer almost 50% of the service requests. If other service discovery protocol were used, all devices would answer with equal probability, 20%. This means that with $SPDP_h$, fixed devices with greater availability time and less limitations answer most of the requests. This was one of the objectives of $SPDP_h$ protocol. (Figure. 12) also shows that devices with very small availability time (in our case, 500 s) answer more requests than devices with middle availability times. This is because these devices are highly mobile, continually change of networks, and in each new environment they arrive, they have to answer requests above their own services, to make them known to the rest of the devices. As we know, $SPDP_h$ is a fully distributed protocol, it does not rely in any central directory. However, with this simulation we show that $SPDP_h$ is designed time. It is worth mentioning that all percents shown in such a way that, if there are devices that are less mobile (remain more time in the environment) and that have more memory, most of the queries will be answered by them, relieving the more mobile and limited ones of answering , and so preserving their battery.

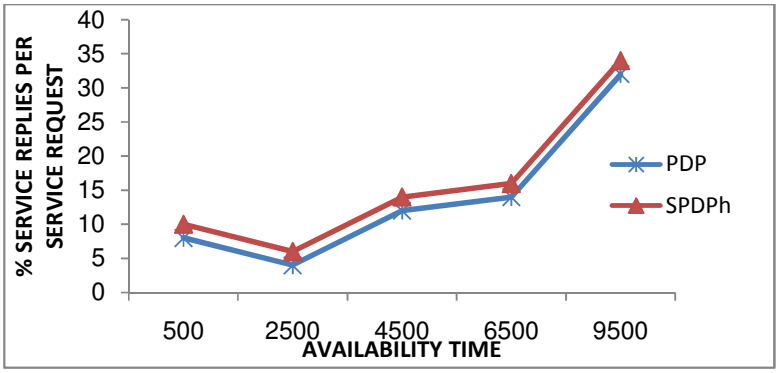
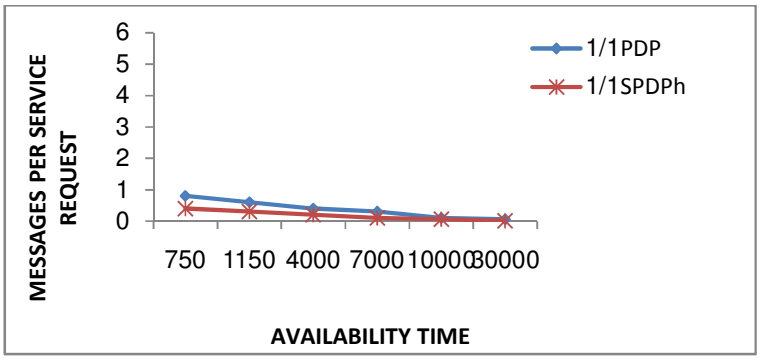


Figure 12. Service replies per search in an heterogeneous environment with PDP.

In this scenario we assume that devices with higher availability time also have greater cache's sizes. This is a realistic assumption, since fixed devices use to have more memory than mobile (small, battery powered) devices. All the above figures considered $SPDP_h$ one query– multiple responses queries. If the application is interested in the service, not in which device offers it, $SPDP_h$ one query–one response (1/1) can be used instead, obtaining a further reduction in number of messages and power consumption. Figure 13 compares $SPDP_h$ one query–one response against the same service discovery protocols as before.



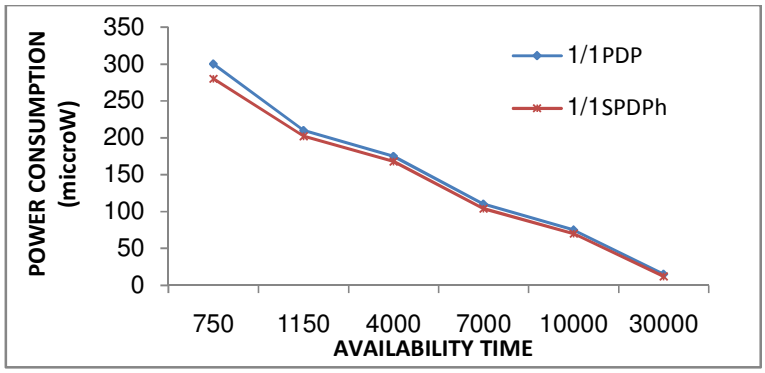


Figure 13: Comparison of PDP 1/1 with $SPDP_h$ 1/1

7 Conclusions

An ad-hoc networks possesses high fame in the development of mobile based technology. When a device connects to an ad-hoc network, it requires to know the services offered by the network. Ad hoc service discovery protocols have some features and characteristics, but also some lacks that $SPDP_h$ proposal will overcome. Between the characteristics, we could point out the following: (i) all of them do not rely on any fixed infrastructure, they work following a distributed approach; (ii) all of them use the broadcast or multicast support of the underlying network level; (iii) all of them include a random timer to avoid collisions when many devices answer a service request; (iv) all of them use a cache in the devices to minimize the number of transmissions; (v) all of them announce services with an associated TTL, that is the time the entry will remain in the cache of the devices that listen the announce; (vi) all of them, but Rendezvous, are based on a push mode approach, although they also support pull mode. This is because they prioritise low latency (in a push mode, the response is obtained immediately from the cache) against accuracy (some information in the cache may be staled, some services may not be known yet). Pull mode is supported when accuracy is more important that latency. Rendezvous is the only protocol exclusively based on a pull mode; (vii) finally, all of them, but Rendezvous, answer service requests with all the information known (i.e., stored in the cache), not just with the services offered by the device itself. The lacks observed in these protocols, and that $SPDP_h$ protocol should attack, are: (i) they do not take into account the different characteristics of the devices (battery, memory size), they handle all of them the same way. $SPDP_h$ protocol will care for the most limited devices, preserving their battery, and it will exploit the fixed, less limited ones; (ii) since, except Rendezvous, they are based on a push mode, announcements are transmitted even when no other device is in the network. This is a waste of battery in many situations, and will be avoided in $SPDP_h$ protocol; (iii) pull mode is supported, but it is not well integrated with the push mode. Applications must choose between consulting the cache or issuing a service request (pull). $SPDP_h$ protocol will be based on a better integration of push and pull modes, consisting on broadcasting service announcements, as in the push mode, not periodically but when a service request is received, as in the pull mode; (iv) they do not take into account different application needs. $SPDP_h$ protocol will take into account that some applications look for any device offering the service, while some others look for all the devices offering the service in the network; (v) since most devices are highly mobile, mechanisms for guaranteeing the consistency of the caches are needed. Rendezvous supports one of such mechanisms. When a device leaves a network, it may issue an announcement of all its services with TTL = 0 to delete previous announcements from the cache of all devices.

ACKNOWLEDGEMENTS

First, I thank ALLAH the most merciful and compassionate for giving me the power and the desire to finish this project. Second, I would like to present my deep special thanks to my supervisors Prof. Mohamed Hashem Abd El-Aziz and Dr. Mohammed Amine Abdullwahed for their valuable support during all the stages of this project including, but not limited to, teaching me the professional way of thinking, providing me with necessary references and helping me in the final appearance of this project. Their honest and precise review was really helpful to trace points of weakness in the project and strengthen them. Third, I am very grateful to my professors were always supporting me to learn and study and to all my family members who gave me spiritual support to complete this project successfully.

References

[1] E. Guttman, C. Perkins, J. Veizades, M. Day. RFC 2608: Service Location Protocol, Version 2, June1999.

[2] Yaron Y. Goland, Ting Cai, Paul Leach, Ye Gu, Simple service discovery protocol/1.0. Internet-draft (work in progress),April 1999. draft-cai-ssdp-v1-03.txt.

[3] Stuart Cheshire, DNS-Based Service Discovery. Internetdraft (work in progress), February 2004.

[4] Jini Architectural Overview, White Paper, 1999.

[5] Salutation Consortium, 1998. Available from: <http:// www.salutation.org>.

[6] Bluetooth Specification v1.1, Part E: Service Discovery Protocol (SDP).

[7] Infrared Data Association, Infrared data association link management 1.1, January 1996.

[8] Dipanjan Chakraborty, Anupam Joshi, Yelena Yesha, Tim Fini, GSD: A novel group-based service discovery protocol for MANETS, in: 4th IEEE Conference on Mobile and Wireless Communications Networks (MWCN 2002), Stockholm. Sweden, September 2002.

[9] Chang-Seok Oh, Young-Bae Ko, Jai-Hoon Kim, A hybrid service discovery for improving robustness in mobile ad hoc networks, in: The International Conference on Dependable Systems and Networks, DSN-2004, Florence, Italy, July 2004.

[10] Rajeev Koodli, Charles E. Perkins, Service discovery in ondemand ad hoc networks (draft-koodli-manetservicediscovery- 00.txt), Internet-draft (work in progress), October 2002.

[11] P.E. Engelstad, Y. Zheng, Evaluation of service discovery architectures for mobile ad hoc networks, in: 2nd Annual Conference on Wireless On-demand Network Systems and Services (WONS'05), January 2005, pp. 2–15.

[12] Michael Nidd, Service discovery in DEAPspace, IEEE Personal Communications, August 2001.

[13] Sumi Helal, Nitin Desai, Varum Verma, Konark—A service discovery and delivery protocol for ad-hoc networks, in:Third IEEE Conference on Wireless Communication Networks (WCNC), New Orleans, March 2003

[14] Michel Barbeau, Evangelos Kranakis, Modeling and performance analysis of service discovery strategies in ad hoc networks, in: International Conference on Wireless Networks, ICWN 2003, Nevada. Canada, June 2003.

[15] Apple, Rendezvous, 2004. Available from: <http://developer. apple.com/macosx/rendezvous/>.

[16] Steven E. Czerwinski, Ben Y. Zhao, Todd D. Hodes, Anthony D. Joseph, Randy H. Katz, An architecture for a secure service discovery service, in: Proc. Mobicom'99, 1999.

[17] Sumi Helal, Standards for service discovery and delivery, IEEE Pervasive Computing (July/September) (2002) 95– 100.

[18] C.-k. Toh, Ad Hoc Mobile Wireless Networks. Protocols and Systems, Prentice-Hall PTR, 2002.

[19] Laura Marie Feeney, Martin Nilsson, Investigating the energy consumption of a wireless network interface in an ad hoc networking environment, in: IEEE INFOCOM, 2001.

[20] C. Morais Cordeiro, H. Gossain, D.P. Agrawal, Multicast over wireless mobile ad hoc networks: present and future directions, IEEE Network 17 (1) (2003) 52–59.

[21] Bob Pascoe, Salutation-lite. Find-and-bind technologies for mobile devices, Technical Report, Salutation Consortium, June 1999.

[22] Alan Kaminsky, JiniME: Jini connection technology for mobile devices. Technical Report, Information Technology Laboratory, Rochester Institute of Technology, August 2000.

[23] Gong Li, JXTA: A network programming environment, IEEE Internet Computing (May–June) (2001) 88–95.

[24] Pavlin Dobrev, David Famolari, Christian Kurzke, Brent A. Miller, Device and service discovery in home networks with OSGi, IEEE Communications Magazine (August) (2002) 86–92.

[25] Robert Grimm, Janet Davis, Eric Lemar, Adam Macbeth, Steven Swanson, Tom Anderson, Brian Bershad, Gaetano Borriello, Steven Gribble, David Wetherall, Programming for pervasive computing environments, Technical Report, University of Washington, June 2001.

[26] Willian Adjie-Winoto, Elliot Schwartz, Balakrishnan, Jeremy Lilley, The design and implementation of an intentional naming system, in: 17th ACM Symposium on Operating Systems Principles (SOSP'99), December 1999, pp. 186– 201.

[27] Stuart Cheshire, Performing DNS queries via IP Multicast,Internet-draft (work in progress), February 2004.

[28] Levon Esibov, Bernard Adoba, Dave Thaler, Linklocal Multicast Name Resolution (LLMNR), Internet-draft (work in progress), July 2004.

[29] Sumi Helal, Nitin Desai, Varum Verma, Bekir Arslan,Konark: A system and protocols for device independent,peer-to-peer discovery and delivery of mobile services, IEEE Transactions on Systems, Man, and Cybernetics 33 (6)(2003) 682–696.

[30] C. Campo, Service discovery in pervasive multi-agent systems, in: Tim Finin, Zakaria Maamar (Eds.), AAMAS Workshop on Ubiquitous Agents on Embedded, Wearable,and Mobile Agents, Bologna, Italy, July 2002.

[31] C. Campo, M. Mun˜oz, J.C. Perea, A. Marı´n, C. Garcı´a- Rubio, GSDL and PDP: a new service discovery middlewareto support spontaneous interactions in pervasive systems, in: Middleware Support for Pervasive Computing (PerWare 2005) at the 3rd Conference on Pervasive Computing (PerCom 2005), March 2005.

[32] C. Siva Ram Murthy, B.S. Manoj, Ad Hoc Wireless Networks: Architectures and Protocols, Prentice-Hall PTR,2004.

[33] D. Meyer, RFC 2365: Administratively Scoped IP Multicast,July 1998 .

[34] C. Campo, J.C. Perea, Implementation of pervasive discovery protocol, 2004. Available from: <http://www.it.uc3m.es/celeste/pdp/>.

[35] E. Guttman, C. Perkins, J. Kempf, RFC 2609: Service Templates and Service: Schemes, June 1999.

[36] Applied Cryptography, Second Edition: Protocols Algorithms and Source Code in C. 01/01/1996, by Bruce Schneier

[37] Pushpita Chatterjee, Trust Based Clustering and Secure Routing Scheme for Mobile AD HOC Networks. International Journal of Computer Networks & Communications (IJCNC), Vol.1, No.2, July 2009.

[38] Ebtisam Amar, Selma Boumerdassi and Eric Renault, Hierarchical Location Service with Prediction in Mobile Ad-Hoc Networks. International Journal of Computer Networks & Communications (IJCNC), Vol.2, No.2, March 2010.

[39] Intisar Al-Mejibli and Martin Colley, Maximum Production of Transmission Messages Rate for Service Discovery Protocols. International Journal of Computer Networks & Communications (IJCNC) Vol.3, No.6, November 2011.

Authors

I am originally from El Menoufiya, Egypt. I had my education Master Degree in Computer Science at Menofia University, Egypt & Bachelor of Science in Computer Sciences in Qena, Egypt. I am currently Researcher at King Saud University as a Researcher. I obtained my certification for Microsoft Certified System Engineer on Microsoft Windows Server 2003&2008- April 2010. Teaching Assistant Computer Demonstrator, Al-Gazeera High Institute for Computer Sciences and Administrative Information System, Egypt, & Teaching and assisting in the delivery of academic education courses for undergraduates