A New Service Discovery Scheme Adapting to User Behavior for Ubiquitous Computing[1]

Yeo Bong Yoon and Hee Yong Youn

School of Information and Communications Engineering,
Sungkyunkwan University, 440-746, Suwon, Korea
{yuny,youn}@ece.skku.ac.kr

Abstract. Discovering primary services requested by users is a very difficult but crucial task in networked system, especially in ubiquitous computing system. The service discovery also has to be an accurate, fast, and stable operation. The typical lookup services such as SLP, Jini, and UPnP focus on either wired networks or particular networks. In this paper, thus, we propose a new service discovery algorithm using service agents adapted to user behavior. We compare the proposed algorithm with the DEAPspace algorithm using real experiment. It reveals that the proposed scheme offers services to the users seamlessly and accurately as much as DEAPspace, while the number of packets used is about half of it.

Keywords: Distributed service, service discovery, ubiquitous computing, user behavior, and wireless network.

1 Introduction

Various computing and networking paradigms have been evolving continuously in the past decades. Nowadays, the users no more need to adjust to the new computing systems because they were designed to learn the pattern of user behavior and adapt themselves to it. Ubiquitous computing system sets up quite efficient environment based on the habits of users. There also exist numerous kinds of small devices such as cellular phone, notebook, mobile devices, and PDA which are being deployed radically in the field nowadays. Advent of such mobile devices allows people to get any service in any place. However, having mobile devices is not enough to construct efficient ubiquitous computing environment. Here it is essential to provide the services not only efficiently but also promptly.

The ubiquitous computing system requires systematic registration and management of services for allowing seamless service in huge network environment. Furthermore, small devices have to rely on battery to operate in all kinds of mobile networks. Therefore, the key issue is how to offer stable and prompt services to the users using small battery power. For this, the ubiquitous computing system has to employ energy-

[1] This research was supported by the Ubiquitous Autonomic Computing and Network Project, 21st Century Frontier R&D Program in Korea and the Brain Korea 21 Project in 2004. Corresponding author: Hee Yong Youn.

O. Gervasi et al. (Eds.): ICCSA 2005, LNCS 3482, pp. 19 – 28, 2005.

efficient, seamless, and user-adaptable service discovery technology. A variety of services have been developed for users in mobile internet environment. Because of this, service discovery technology becomes very important in mobile network.

There exist several solutions for the issue mentioned above, which have been proposed and adopted in the real world. The Service Location Protocol (SLP) [1] is a standard protocol which offers scalable framework automatically discovering resources in the IP network. Lookup service was defined by Sun Microsystems to discover and manage services in Jini [2] middleware system based on CORBA [3]. Jini Lookup Service provides mainly naming and trading service in the network. The Service Discovery Protocol (SDP) [4] was addressed by Bluetooth Forum to compose a Bluetooth piconet and user services. UPnP [5] of Microsoft defines an architecture for pervasive peer-to-peer network connectivity in home network.

Service discoveries with SLP and Jini are very stable and powerful in terms of offering services to the users. However, they do not focus on the mobile network and their operations are restricted to low power mobile devices. In this paper, thus, we propose a new service discovery algorithm using service agents adapted to user behavior. In the proposed algorithm each node has a number of requested services. Network traffic is substantially reduced in the proposed scheme by keeping a list of services maintained according to the frequency requested. If a service infrequently used is requested, it is served by the on-demand model. The proposed algorithm keeps the latest services which are frequently requested, and sends infrequently requested service to reduce network traffic. We compare the proposed algorithm with the DEAPspace algorithm [6-8] using real experiment. It reveals that the proposed scheme offers services to the users seamlessly and accurately as much as DEAPspace, while the number of packets used is about half of it. Note that reducing packet transmission is a very important issue for small mobile devices of limited energy.

The rest of the paper is organized as follows. Section 2 briefly reviews the existing discovery services. Section 3 proposes a new service discovery algorithm adapted to user behavior. Section 4 presents the experiment results. Finally, we conclude the paper in Section 5.

2 Related Works

Numerous approaches have been proposed to discover various services in a network. Here we briefly review two systems among them, DEAPspace and Konark [9]. DEAPspace furnishes an algorithm that discovers services in a single hop adhoc network, while Konark is a middleware offering service discovery in multi-hop adhoc network. DEAPspace and Konark are the representative system in terms of performance and effectiveness for service discovery in single hop and multi-hop adhoc network, respectively, and thus they are mostly adopted as references.

2.1 DEAPspace

IBM has developed DEAPspace that solves the problem of service discovery in wireless single-hop adhoc networks. It restricts itself to a small network by assuming a single-hop adhoc network and broadcasting fit in a message. Its final proposal is for

fast concentration of available service information in the network. Maintenance of a centralized node storing all the service information in the network is difficult and complicated. DEAPspace thus selects a distributed approach with the push model in which servers send unsolicited service advertisements to the clients.

In the DEAPSpace, each node has a world view which stores a list of all services offered in the network. Furthermore, each node periodically broadcasts its own world view to other nodes using the adaptive backoff mechanism. If a node receives a service advertisement, it checks the service id and expiration time of its own world view, and decides whether to advertise a service message or update the cache data. Thereafter, the node increases the rate of broadcasting. If new service advertisements offered in the network have longer lifetime values than those in the internal cache of a node, it adds them into its own internal cache [10,11].

2.2 Kornark

Konark is a middleware designed specifically for discovery and delivery of services in multi-hop adhoc networks. It thus aims at wireless adhoc network that is usually larger than the network DEAPspace supports. The Konark architecture mimics a typical operating system and it is programming language independent. It provides a framework in which services are described and delivered using open standard XML technology over IP network connectivity.

Konark supports both the push and pull model. When a node receives a service message of the multicast address of the Konark, it multicasts different world view of relevant services and other services contained in the received message. When a client sends service requests, the servers replies unsolicited service advertisements. Each node adopts Konark SDP Manager which is responsible for discovery of the requested services, and registration and advertisement of its local services. To discover services in the network, clients use a discovery process known as active pull mechanism. Each node joins a locally-scoped multicast group.

The SDP Manager of each node maintains a cache, called a service registry. The service registry is a structure that enables devices to store their local services. It also allows them to maintain information on the services that they might have discovered or received via advertisements.

3 The Proposed Scheme

In general, if a server frequently advertises the service list, the clients can quickly discover the requested service. Here discovery service must not take too much traffic while offering services in a timely fashion. The two conflicting goals, less traffic and fast response, should be well balanced when the services are available to use. In addition to satisfying these goals, a service discovery which is also user-adaptable is proposed.

3.1 The Overview

For ubiquitous computing it will be efficient to use service discovery for both the server and client such as Jini lookup service. The lookup service is located in the

middle of the clients to manage the requested services. This kind of service discovery approach based on agent system can provide fast, efficient, and seamless services to the one needing them. However, the server and client system can be effective only in the networked environment. Mobile devices are not fixed at specific locations inside the mobile network. This is one of the reasons that a server system is not set in mobile network. Mobile devices need to frequently send a broadcast message to get service information with respect to service location, service time, and so on. They have their own repository to keep the latest service information.

Note that most people want to repeatedly use only some specific services from the network. They are just interested in the services which are related to their job or hobby. It is thus better to offer an efficient service mechanism only for a confined set of services instead of inefficient wide spectrum of diverse services. If a user is not interested in a specific service the user may demand it infrequently or never does that.

The basic idea of the proposed scheme is to substantially reduce the traffic due to infrequently requested services as identified above. Providing service list adapted to user behavior and preference will significantly reduce network traffic and offer fast services. The proposed scheme employs an approach in which each device broadcasts such adapted service list for the distributed system. Moreover, when a user wants to use services, the user sends a request message like an on-demand service model for reducing network traffic.

3.2 The Flow of Operation

Figure 1 shows how the proposed service discovery operation works, which consists of the following three steps for example.

Step 1. If Node-2 needs a service, it checks the service cache in its own repository.
Step 2. If the service is not in the service cache, it sends a request to other devices.
Step 3. If other devices have the service requested by Node-2, the device broadcasts the service list to other devices.

In the proposed scheme each node has a service cache (SC) storing the adaptable service list. Figure 2 shows the SC consisting of service elements (SE). Each SE has three fields; Service ID, Expiry, and Hit. Service ID is used to distinguish the service and Expiry indicates how long the service can be used by the users. Hit indicates how many times the user requested the service. When a user requests a service, its Hit value is increased by 3 points.

Each Hit value is decreased by 1 point if not requested in every 20 minutes. As a result, the Hit value of frequently requested service can maintain relatively higher Hit value than that of uninterested services decreasing continuously. All the SEs in the SC are sorted by the Hit value. We assume that ten most frequently requested service elements in the SC belong to a set of special interests, while others are plain service elements probably unrelated to the users. The ten SEs are called ‘hot elements’, and the number of hot elements can be varied for each specific implementation.

The hot elements are kept in the latest service list of the SC, and the message containing them is broadcast more frequently than other messages to reduce network traffic.

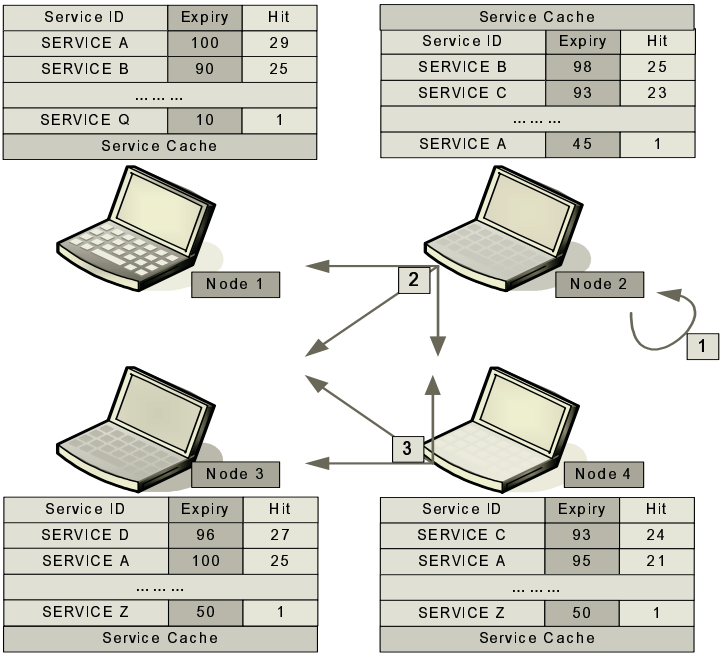

Fig. 1. The operation flow in the proposed scheme

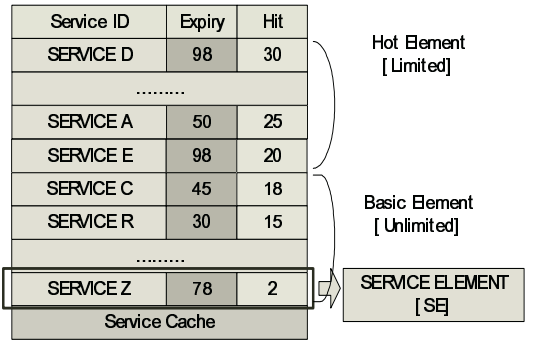

Fig. 2. The structure of service cache

3.3 Algorithm

The proposed algorithm consists of three main functions; Local Service Function, Receive Element Function, and Receive Service Function. Figure 3 shows a pseudo code of Local Service Function checking whether the requesting service is in the SC or not. Figure 4 shows how Receive Element Function works when a device receives a message requesting a specific service from it. Finally, Receive Service Function, defined in Figure 5, settles other's SC receiving a message from a remote node.

3.3.1 Local Service Function

Line 3: When a user requests a particular service, the hit count of the SE is increased.
Line 4: The SC is sorted by the hit counts of the SEs to maintain the latest data reflecting the usage.
Line 5: Local Service Function checks if the requested service exists in the SC. In Figures 3, MINE in line 3 is its own SC.
Line 7: Check_expiry function has two kinds of operations. If the requested service id belongs to hot elements, check_function checks the expiration time of all the hot elements. When the requested service element is just a basic element, check_function checks the expiration time of just the service element.
Line 9: If the hot element and requested service have a smaller value than minimum value of expiration time, local service function broadcasts a query message. One of the user's favorite services or just requested service will soon be shut down.
Line 14: If the own SC does not have the service element, Local Service Function also broadcasts the query message.

```
1   LOCAL SERVICE (SE d.ID)
2   {
3         Increase (MINE.d.HIT)
4         Sort (MINE)
5         IF (d.ID ∈ MINE.ID)
6         {
7           IF (Check_expiry())
8              {
9                  BROADCAST ELEMENT(d.ID)
10             }
11        }
12        ELSE
13        {
14          BROADCAST ELEMENT (d.ID)
15        }
16  }
```

Fig. 3. The pseudo-code of the Local Service Function

3.3.2 Receive Element Function

When a node receives a broadcast message, the Receive Element Function checks whether the SE is in its own SC or not. If it exists in its own SC and the expiration time is greater than the minimum value, the Receive Element Function broadcasts all the SEs in the SC. It is unnecessary for a node to request the SE even though the node does not have the SE. This is because the node will receive the SE from other nodes sooner or later through the Receive Service Function message. Note that the service lists are synchronized through the distributed service in the network.

```
1   RECEIVE ELEMENT (c.ID)
2   {
3         IF (c.ID ∈ MINE.ID)
4         {
5             IF(MINE.ID.EXPIRY > MIN)
6             {
7                 BROADCAST SERVICE (MINE)
8             }
9         }
10   }
```

Fig. 4. The pseudo-code of the Receive Element Function

3.3.3 Receive Service Function

Line 4-7: When a node receives a new SE from a remote node, the node inserts that in its own SC. The service is stored at bottom of the SC.

Line 10-11: If the SE is already in the SC, it checks the expiry time. If the received SE has the latest date, the Receive Service Function updates the expiry time of it.

```
1   RECEIVE SERVICE (REMOTE)
2   {
3         For each r □ REMOTE
4         IF (r □ MINE)
5         {
6           Insert(r, MINE)
7         }
8         ELSE
9         {
10          IF(r.expiry > MINE.expiry)
11          {  Update(r, MINE) }
12        }
13   }
```

Fig. 5. The pseudo-code of the Receive Service Function

4 Performance Evaluation

This section describes the experiment environment and the results of experiment.

4.1 Experiment Environment

In our experiment we use four 2.4GHz Pentium IV machines with 1GB of main memory each. We assumed the number of services in a network is 50, and each ser-

vice has a random expiration time from 1 to 100. Expiration time is decremented in every two minutes. Also, every service updates its own expiry time to a random number between 50 and 100 in every twenty minutes. Each node has 30 service elements at the starting point. When we set the value of time-out as 3'00'', the average time-out with the DEAPspace algorithm turned out to be 4'25''. Thus, we set a random time value between 2'00'' and 4'00'' as the time a user requests a service in the experiment with the proposed algorithm.

4.2 Experiment Result

Figure 6 shows the average expiry time value of the proposed scheme and DEAPspace. We can see that the difference between the two is very small, which means that the proposed scheme offers seamless and accurate service to the users as much as DEAPspace does. Notice that each peak point in the plots represents service update time in every twenty minutes as we preset.

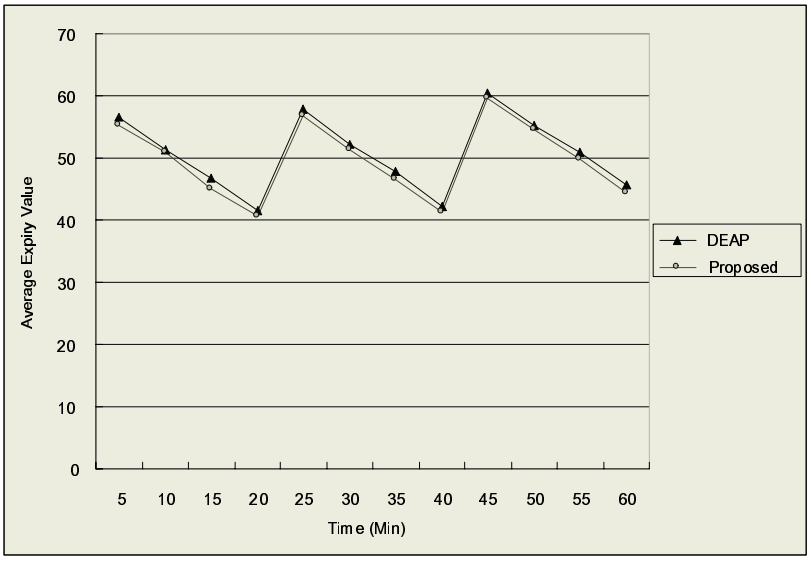

Fig. 6. Comparison of average expiration time.

Figure 7 shows the amount of packets transmitted in every ten minutes. Recall that when a user requests a service belonging to uninterested service elements (non-hot elements), the proposed scheme broadcasts a query message to get the information of others. Proposed-1 is the case that services in the SC are requested equally likely. Meanwhile, Proposed-2 is the case that user requests some service elements more frequently than the others. When this happens, the Local Service Function checks the expiration time of the hot elements. If one of the hot elements has a minimum expiration time, the node broadcasts a query message until it gets the latest data of the service at every request time. As a result, Proposed-2 uses more packets than

Proposed-1. Notice from Figure 6 that Proposed-1 and Proposed-2 use fewer packets than DEAPspace. Also, notice from the figure that the amount of packets of the proposed scheme and DEAPspace linearly increases as time passes. In other words, the average number of packets transmitted per minute is about 300 and 600 for the proposed scheme and DEAPspace, respectively. The proposed scheme uses about half of the packets used by DEAPspace, which is a quite substantial reduction.

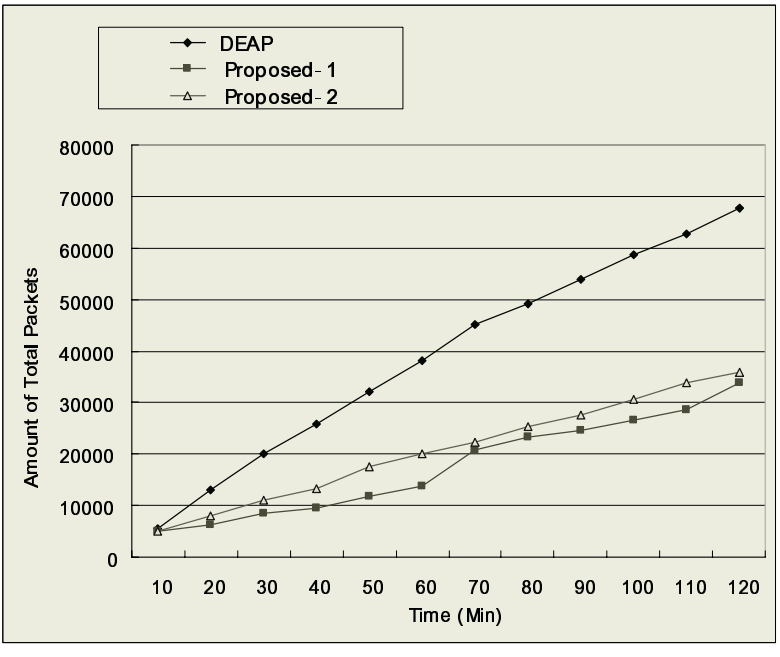

Fig. 7. Comparison of total number of packets transmitted

5 Conclusion

We have addressed a new computing paradigm in which the users demand individual service. The ubiquitous system requires registration and management of systematic control of services for each user in mobile network. For this reason, service discovery technology becomes very important in mobile network. We have also reviewed some present solutions such as Jini, SLP, SDP, and UPnP with more attention to DEAPspace and Konark. Most of the present service discoveries are focusing on networked system or particular network such as Bluetooth or Piconet.

In this paper we have proposed an energy-efficient service discovery algorithm using a list of services adapted to the frequency of requests. Comparing with DEAPspace, we have shown that the proposed scheme allows the same service quality using much less network traffic. Experiment with four machines displayed that the proposed scheme uses about half of the amount of packets used by DEAPspace. We will ex-

pand the experiment considering various factors such as the type of the services requested and duration of usage.

When a hot service element cannot be offered to a user, several packets need to be transmitted to get the service information. We will carry out research to get over the problem using effective service registration.

References

[1] SUN Microsystems: Jini Architecture Specification, Version 1.2. http://wwws.sun.com/software/jini/specs/jini1.2html/jini-title.html (2001)

[2] Guttman, E.: Service location protocol. Automatic discovery of IP network services. in Internet Computing, IEEE (1999) Volume 3, Issue 4, 71-80

[3] Object Management Group: The Common Object Request Broker Architecture. Core Specification, Version 3.0.3, Editorial changes formal/04-03-12 (2004)

[4] Bluetooth, Specification of the Bluetooth System, Specification Volume 1, 2001, Specification Volume 2, 2001. http://www.bluetooth.com

[5] UPnP FORUM, Universal Plug and Play Device Architecture Version 1.0.1, http://www.upnp.org, 2003

[6] M. Nidd : Service Discovery in DEAPspace, IEEE Personal Communications, August 2001.

[7] R. Hermann, D. Husemann, M. Moser, M. Nidd, C. Rohner, A. Schade : DEAPspace - Transient Ad-Hoc Networking of Pervasive Devices, Computer Networks, vol. 35, pp. 411-428, 2001.

[8] M. Nidd, "Timeliness of Service Discovery in DEAPspace," 29th Int'l. Conf. Parallel-Processing," 29th Int'l. Conf. Parallel Proc. Workshop. Pervasive Comp., Aug. 2000, pp. 73–80.

[9] S. Helal, N. Desai, V. Verma, C. Lee : Konark - A Service Discovery and Delivery Protocol for Ad-hoc Networks, Proceedings of the Third IEEE Conference on Wireless Communication Networks (WCNC), New Orleans, March 2003.

[10] Honghui Luo, Michel Barbeau : Performance Evaluation of Service Discovery Strategies in Ad Hoc Networks, 2nd Annual Conference on Communication Networks and Services Research (CNSR 2004) Fredericton, N.B., Canada, May 19-21, 2004

[11] M. Barbeau, E. Kranakis, Modeling and Performance Analysis of Service Discovery Strategies in Ad Hoc Networks, Proceedings of International Conference on Wireless Networks (ICWN), Las Vegas, Nevada, 2003.