A Gnutella Inspired Ubiquitous Service Discovery Framework for Pervasive Computing Environment *

Tianyin Xu†, Baoliu Ye†, Mitsunori Kubo‡, Arata Shinozaki‡, Sanglu Lu†
†State Key Lab. for Novel Software and Technology, Nanjing University, Nanjing 210093, China
{flydutchman, yebl, sanglu}@dislab.nju.edu.cn
‡Future Creation Lab, Olympus Corp., Shinjuku, Tokyo 163-0914, JAPAN
{mi_kubo, arata_shinozaki}@ot.olympus.co.jp

Abstract

Pervasive computing, the new distributed computing paradigm aiming at providing services anywhere anytime, poses unique challenges on service management and discovery. In this paper, we propose a new pervasive computing framework named USDM-PerComp by using a Web Service Server/Directory Server (WSS/DS) based two-level hierarchical topology to address the challenges involved. DSs within USDM-PerComp autonomously form a peer-to-peer (P2P) overlay. We present a Gnutella inspired distributed algorithm to support service discovery over the resulted P2P overlay, and further implement a service crawler using mobile agent techniques to perform the discovery and invocation. With the service crawler, service composition could be done automatically and spontaneously without any user concern. Besides, we develop a distributed JPEG encoding application to verify the practicability of our proposal. Our experience confirms the scalability and flexibility of USDM-PerComp. The pervasive applications could be easily as well as efficiently created under USDM-PerComp environment.

1. Introduction

Pervasive computing provides an attractive vision for accessing services anywhere, anytime. It has been regarded as the next generation of enabling Internet computing technology that would impact industry, government and daily life much as the personal computing revolution did. Its goal is to build a user-centric novel environment abounding with various kinds of information and services. The landscape of pervasive computing era is an environment where the focus of computing is turned away from personal computers towards collaborations among heterogeneous embedded devices with limited functionalities.

In recent years, fueled by advances in processing power, storage capacity, and battery life, the proliferation of mobile computing devices and wireless communication devices makes the hardware and network infrastructure for realizing this vision become reality. Some fundamental elements are even becoming viable commercial products. However, it is not a simple task to build a pervasive application with existing services provided by different providers. Developers have to deal with the interoperability among heterogeneous devices. In addition, the increased usage of mobile devices has also increased the amount of user effort required to dynamically operate devices dispersed over physical environments. The key challenge is how to make pervasive application integration easy and make it adapt to such highly dynamic heterogenous environments smoothly even if users and/or devices are roaming.

Web service technologies seem to be a promising candidate for this problem in nature. A web service is like an abstract middleware hiding the details about the application platform and programming language. Any software application or legacy system can be wrapped into web services conveniently [4]. Any individual or organization can implement any functionality as web services and publish them to the pervasive space easily. Nowadays, web service technologies have been widely accepted by both industry and academia and have been considered as the *de facto* standards for implementing pervasive services over the Internet. How to find the desired web services from the disordered massive ones dispersed over the decentralized pervasive space, and then gracefully integrate them to accomplish a specific user task, becomes one of the key issues to be addressed for pervasive application implementation.

*This work is partially supported by the National High-Tech Research and Development Program of China (863) under Grant No. 2006AA01Z199; the National Natural Science Foundation of China under Grant No. 90718031, 60573106, 60721002; the National Basic Research Program of China (973) under Grant No. 2002CB312002.

In this paper, we focus on the service management and discovery over large-scale dynamic heterogeneous pervasive computing environment and present ubiquitous web service discovery and management mechanisms. The main contributions of our work are as follows: 1) We propose a novel pervasive computing framework named USDM-PerComp environment (Ubiquitous Service Discovery and Management for Pervasive Computing) by using WSS/DS based two-level hierarchical topology to support scalable and flexible service management; 2) We present a Gnutella inspired distributed service discovery mechanism to improve the service searching efficiency across DSs; and 3) We implement a service crawler using mobile agent techniques to perform the Gnutella inspired algorithm for high personalized and agile service invocation/composition. Our target is to support service composition based spontaneous pervasive application construction so that service requests could be accomplished without user concern. In addition, we develop a distributed JPEG encoding application to verify the practicability of our proposal.

The remainder of this paper is organized as follows. Section 2 overviews related work. Section 3 introduces the framework of the USDM-PerComp environment. Section 4 describes the service management mechanism. Section 5 presents the service discovery algorithm. Section 6 demonstrates a distributed JPEG encoding application using our proposal. Finally, we conclude our work in Section 7.

2. Related Work

Service discovery essentially refers to the discovery of service description [7]. Although Web Service Description Language (WSDL) provides an XML-based general-purpose description scheme, it is weak in supporting semantic information. Previous researches show that *ontology* has the potential to enhance service description with semantic information and share contextual knowledge of services. Several ontology based service models such as OWL-S and SOUPA have been proposed. In this paper, we assume that all services in pervasive environments can be accurately described and matched via employing ontology based models.

There also have been many research literatures proposed to address the service discovery issues. However, most of them are designed just for local environments confined with fixed network conditions such as home and enterprise. Jin *et al.* [2] present a unified service discovery framework that allows users to designate QoS metrics by using Directory Server (DS) and DSA (DS Admin). It, however, is a centralized architecture that is unsuitable for pervasive environment. Campo *et al.* [1] present a light-weight discovery protocol called Pervasive Discovery Protocol (PDP). It is a fully distributed discovery protocol which aims at service interactions in wireless ad hoc networks. Kim *et al.* [3] present VSD (Service Discovery based on Volunteers), a service discovery architecture for multi-hop wireless ad hoc networks. In VSD, more capable and stable nodes carry out directory services as volunteers. Xu *et al.* [8] integrate PDP and VSD into a new discovery protocol named VPDP. Volunteers in VPDP architecture are middleware nodes with less limited resources.

Different from previous work, we propose to use Gnutella inspired service discovery mechanism with the USDM-PerComp environment to support high performance for service discovery, as well as high scalability and flexibility for service management. In addition, by using mobile agent techniques instead of collaborative logic control, our mechanism provides personalization and agility for service invocation and composition.

3. Framework Overview

To overcome the limitations of traditional distributed systems, we think the new pervasive service discovery algorithm, as well as management mechanism, should at least satisfy the following requirements. First, since people and computing resources are totally distributed over the physical space without any centralized control, the service discovery mechanism itself should be able to work over fully unstructured environments. Second, pervasive computing advocates a new generation of distributed computing paradigm where services are available anytime, anywhere. Consequently, the service management mechanism should be scalable and flexible so that web services could be registered/removed into/from pervasive environment freely. Besides, the service invocation and composition procedure should be automatic without user concern.

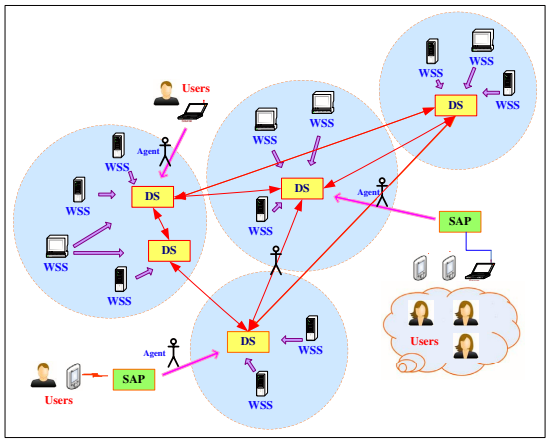

Figure 1. Topology of the USDM-PerComp.

In this paper, we present a USDM-PerComp environment to meet the above requirements. Fig. 1 shows

the overall system framework of USDM-PerComp environment. It mainly comprises three components:

(1) Directory Server (DS). A directory server rules a certain region in the pervasive computing environment. It maintains the meta information of all existing web services distributed within this region. Different from those in Reference [2], a DS contains a running platform to accommodate mobile agents from outer. A mobile agent residing on a specific DS can either search for desired services from the whole pervasive environment or directly invoke the services belonging to the region controlled by the DS. Note that, a DS can communicate with other DSs in the environment. All the DSs form a peer-to-peer overlay network.

(2) Web Service Server (WSS). Different from the conventional super web servers, a WSS could be acted by a small workstation, even a personal computer, so long as it is willing to contribute a limited resource for the pervasive environment. Most important of all, a WSS is not merely a hardware server which holds a lot of web services but contains a powerful software manager. Each WSS contains a *system agent* managing description information of all the web services on its local computing node.

(3) Service Access Point (SAP). An SAP is a user portal for accessing pervasive applications. It is usually implemented in the form of a website or acted by a well-known web server. Pervasive users could interact with it using their smart devices. By means of interacting with user devices, SAPs can get the user task specifications, and then build and launch mobile agents to achieve user tasks.

From Fig. 1, we can see that, the environment topology is totally distributed. Failures and crashes of some DSs or WSSs will not infect the whole system. Since the central server existed in traditional service environment is eliminated, we could gracefully avoid the bottleneck occurred in large scale application scenario such as a building or a university campus. Besides, we can enlarge the scale and enhance the functionality of the USDM-PerComp environment easily and conveniently.

We separate the above components both logically and implementationally. Our experience shows it is useful for building a practical pervasive computing environment where users can perform tasks that have not been designed into the system, thus multiplying the use of the sources of functionality.

4. Service Management

Service management is critical for effective utilization of services. Service discovery and composition rely on service management seriously. Fig. 2 illustrates the software hierarchy of the whole USDM-PerComp environment. The functionalities of service management are implemented by the two components at the lower level, *i.e.* DSs and WSSs.

In USDM-PerComp, the whole pervasive environment is divided into different disjoint regions. Each region contains numbers of WSSs that provide services to the whole environment. And each region is charged by one or more directory servers. A directory server holds the service distribution information of its region by maintaining a Regional Service Description List (RSDL), which includes the details of all the service descriptions. To maintain this list, a DS directly connects with every WSS within corresponding region. Each WSS maintains a Local Service Description List (LSDL), which records all the descriptions of web services residing on the local node. A WSS periodically sends its LSDL to the directory server. It will also dynamically send the LSDL to the DS upon receiving a service appending event or a service invalidation event. Once receiving a LSDL from one WSS, the DS will update corresponding items in the RSDL. (In fact, the RSDL is maintained by structuring the LSDLs from WSSs as a whole.)

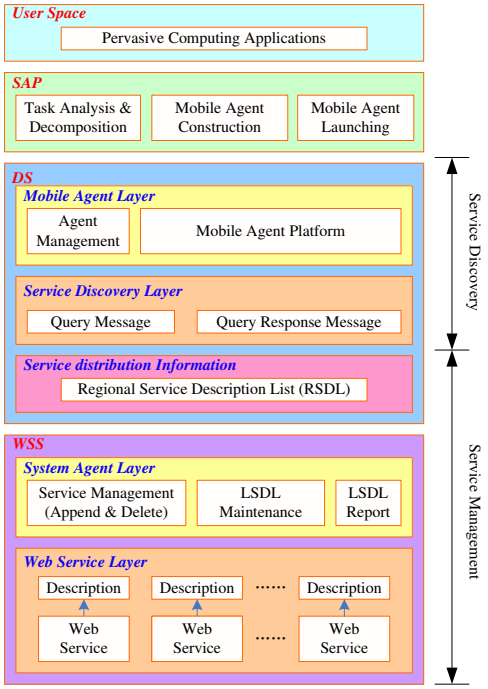

Figure 2. Hierarchy of the USDM-PerComp.

To publish a new web service, it is necessary for the service provider to register the service description information of the new one to the local WSS via a friendly register GUI. The GUI is usually in the form of XML based page or client software. Then the WSS adds a new entry in the LSDL corresponding to this service and sends the LSDL to the DS. Analogously, it is the responsibility of a WSS to detect service invalidation on the local node and report the exception to the corresponding DS. On each WSS, there exists a stationary *system agent* that plays the role of the web service

manager to do all these work centered at the LSDL.

We can see that it's easy to publish a new web service into the USDM-PerComp environment. What's more, it is also quite convenient to integrate a new WSS into the USDM-PerComp environment. A device can act as a WSS by merely installing the system agent software suite. This is just what pervasive computing advocates: anybody, any device can easily join in the environment to participate the computing. Apparently, a registration is necessary, *i.e.* the new coming WSS needs to register itself to the DSs. The registration also can be done with a friendly graphic interface, just choose a DS to connect, click your mouse, and that will be ok. After registration, the system agent starts to periodically send its LSDL to the corresponding DS. If a DS hasn't received the LSDL from a WSS for a certain time, the directory server will assume the WSS to be an invalid node and then delete all the information regarding the WSS from its RSDL.

5. Service Discovery

Service discovery emphasizes on offering available services to users and/or devices transparently [2]. In our pervasive environment, service discovery means the mobile agents should be able to discover and access the desired web services from ambient environment automatically.

In this section, we present a Gnutella inspired mechanism to support service discovery over unstructured P2P overlay formed by DSs. We first introduce the mobile agent and directory server before describing the details of the service discovery algorithm. Mobile agents reside on DSs and act as crawlers to perform the discovery activities.

5.1. Mobile Agent: Service Crawler

A mobile agent is the service crawler. It can roam through the whole pervasive environment freely, finding necessary web services, invoking them compositely to accomplish user tasks. Traditional solutions put all the work on clients. Unfortunately, this kind of solutions always introduce high overhead while resulting in low autonomy. The motivation behind mobile agent techniques is that we should migrate the computing to different locations rather than transferring bulk of data frequently. By using the mobile agent techniques, the data flow injected into the network could be significantly reduced. More important, the clients no longer have to concern with the whole computing process but just launch mobile agents to perform the work. This greatly improves the clients' computing performance and response time to users. Such clients are no other than the SAPs in the USDM-PerComp environment.

The main work of an SAP is to build mobile agents according to user task specifications. We assume that the task decomposition is specified by user task specifications. A mobile agent represents a specific user to do a specific task. The model is layered from services to tasks, and finally agents [4]. A task is in essence a user request, which could be accomplished by one service or the collaborative combination of multiple services. After the agent construction, the SAP launches the mobile agents into USDM-PerComp environment. Following that, a mobile agent starts to find services, migrate itself, invoke services. This procedure will be repeated until the agent eventually accomplishes its task.

In order to improve service discovery and invocation efficiency and avoid failure, mobile agents will never migrate to a WSS. WSSs usually have higher failure probability than DSs. The mobile agent will be unexpected invalidated if the WSS holding the mobile agent fails. Moreover, the cost of mobile agent migration could not be ignored. Migration cost will be unacceptable when the agent wants to invoke a lightweight web service. To balance these conflicts, mobile agents in USDM-PerComp environment perform service discovery and invocation only on the DS level.

5.2. Directory Server

As illustrated in Fig. 2, the functionalities of service discovery are implemented by the components in DSs. These components can be divided into two layers, *mobile agent layer* and *service discovery layer*. The latter provides interfaces to the former. The interfaces include sending *query messages* and receiving *query response messages*. The mobile agent layer includes a running platform, on which remote mobile agent can do searching and invocation. Thus, a DS has the ability of query and communication. A DS can communicate with other DSs to disseminate the group membership messages and search messages [5]. These behaviors are driven by mobile agents residing on it.

All the DSs form a peer-to-peer overlay network. Each DS can be seen as a functional node in the P2P overlay. The DSs are symmetric and could function as both clients and servers. The overlay composed by DSs in Fig. 1 can be depicted in Fig. 3. The real line represents the real TCP connection in the network, and the broken line represents the virtual logic connection in the peer-to-peer network.

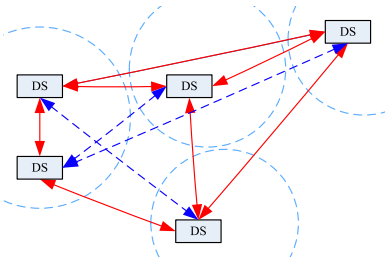

Figure 3. The P2P overlay formed by DSs.

There could be coexisting more than one DS in one region for the purpose of improving query efficiency and enlarging storage capability. Literature [2] proposed several methods to achieve a good performance and search efficiency. Such methods include creating service index, hotspot service replication and service distribution algorithms.

5.3. Service Discovery Mechanism

As described earlier, the DSs in our USDM-PerComp environment form an unstructured P2P overlay network. DS nodes are peers that performs tasks normally associated with both servers and clients. A DS provides client-side interfaces through which mobile agents can issue queries and receive search results. A DS is also responsible for the server-side interfaces that accept queries from other DSs, checking for services against their service distribution list, and responding with corresponding results. This network environment is very similar to the Gnutella network model.

We present a Gnutella inspired service discovery mechanism in our USDM-PerComp environment. The DSs and mobile agents use this mechanism to achieve high performance for service discovery in USDM-PerComp environment. To locate a web service, the mobile agent on a specific DS will let the DS initiate a constrained flood of the network by sending a query packet to all its neighbor DSs [6]. The scope of flooding is controlled with a time-to-live (TTL) field that is decremented on each hop [6]. Upon receiving a query packet, a DS checks its RSDL to see whether there have any services matching the query. If this is true, the DS sends a query response packet back towards the query originator through the overlay [6]. Whether or not a service match is found locally, the DS continues to flood the query through the overlay until it receives a message with a TTL of zero. Fig. 4 shows this service query and query response process.

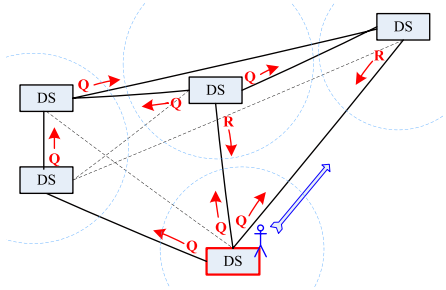

Figure 4. The service query and query response process using the Gnutella inspired service discovery mechanism.

When the query originator receives a query response message from another DS in the network, the mobile agent will migrate to that DS to invoke the required web service. If the query originator receives several query response messages, the mobile agent will record all these DSs' location and migrate to the best one. When an invocation is finished, the mobile agent will continue searching for other services for subsequent subtasks. This procedure repeats until the whole task is accomplished. Finally, the mobile agent goes back to the starting SAP and returns the result to the user.

We employ the similar *ping* and *pong* messages of Gnutella protocol to maintain the overlay. The two kinds of messages work as follows: any DS receiving a ping message replies a pong message back towards the originator, and forwards the ping onwards to the other neighbors [6]. In this way, the ping messages and the query messages could flood through the network if their TTL is large enough. The pong messages and the query response messages contain the IP addresses, ports and other location information. The query response messages additionally contain service matching information and unique client IDs associated with query messages.

In general, there're four types of messages working among DSs. The ping and pong messages are called *group membership messages* while the query messages and query response messages are called *search messages* [5].

6. A Case Study

To verify our proposal, we deployed a real testbed on a floor of a building to simulate USDM-PerComp environment. We also implemented a distributed JPEG encoding application on the testbed. The functionality of this application is to convert image files from bitmap format to JPEG format. Our target application scenario is a mobile environment where users may roam through the environment freely, issuing tasks anywhere, anytime.

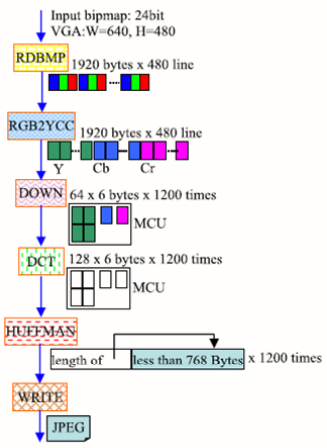

Figure 5. JPEG encoding workflow.

The JPEG encoding procedure is divided into six independent subtasks. The first subtask reads the raw bitmap file uploaded by the user. The last subtask writes the encoded data to a standard JPEG file. Fig. 5 demonstrates the entire workflow of the JPEG encoding application. Each intermediate subtask receives data set from its predecessor and sends the data result to its successor. As a result, these six subtasks form a sequential task chain. For each subtask, there already exist more than one web services that are capable of performing it in the pervasive environment. Using the proposed service discovery algorithm and management mechanism, agents can easily find and invoke each corresponding service to accomplish the encoding task.

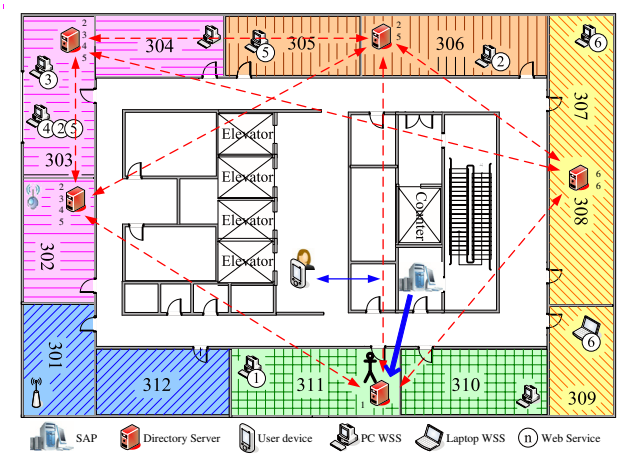

Figure 6. The deployment of the testbed.

Fig. 6 shows the deployment of the testbed. The rooms with the same pattern compose a special region. Each region is charged by one or two directory servers. And the circles centered with a number *n* represent the web services which can solve the *nth* subtask in the task chain. There is an SAP near the counter of the floor. In our simulation scenario, we assume a user wants to convert some pictures to JPEG format when waiting for the elevator. He or she interacts with the system with a PDA. The internal processing routine for this scenario is as follows. At first, the SAP builds a mobile agent for this task and launches the agent to the nearest DS in room 311 (see Fig. 6). When residing on the DS in room 311, the mobile agent searches the local RSDL and finds a RDBMP service in room 311 being capable of performing subtask 1. After invoking the RDBMP service in the local region, the mobile agent accomplishes subtask 1. Then the agent goes on searching and executes subtask 2. This time, the mobile agent can not find the relevant web services in the local RSDL. So the service discovery mechanism is used, *i.e.* the agent drives the DS in room 311 to send query messages to search for the RGB2YCC services. After a while, the agent receives two query response messages separately from the DSs in room 303 and 306. Both of them have the RGB2YCC services to do the second subtask. Since the workload of the WSS in room 303 is much heavier than the one in 306 (the information can be got in the service description entry), the mobile agent will choose the DS in room 306 to migrate to. The agent then repeats the similar routine until it finishes all the six subtasks. Accomplishing the task, the mobile agent should have the final JPEG files in itself. Finally, it migrates back to the SAP and submits the result picture.

7. Conclusions

In this paper, we propose a scalable and flexible pervasive computing framework named USDM-PerComp and present a Gnutella inspired distributed algorithm on it to support high performance for web service discovery. We describe the design and implementation of the algorithm as well as the USDM-PerComp environment. We also develop a distributed JPEG encoding application on a real testbed to verify the practicability of our proposal.

In the future, we will enhance the USDM-PerComp environment with much effort on making the pervasive environment be more context-aware and conduct further researches on service description and accurate service matching.

References

[1] C. Campo, C. García-Rubio, A. M. López, and F. Almenárez. PDP: A lightweight discovery protocol for local-scope interactions in wireless ad hoc networks. *Computer Networks*, 50(17):3264–3283, December 2006.

[2] B. Jin, L. Zhang, and Z. Zang. A unified service discovery framework. In *Proc. of the 6th International Conference on Grid and Cooperative Computing(GCC 2007)*, 2007.

[3] M. J. Kim, M. Kumar, and B. A. Shirazi. Service discovery using volunteer nodes for pervasive environments. In *Proc. of International Conference on Pervasive Services(ICPS'05)*.

[4] R. Liu, F. Chen, H. Yang, W. C. Chu, and Y. Lai. Agent-based web services evolution for pervasive computing. In *Proc. of the 11th Asia-Pacific Software Engineering Conference (APSEC '04)*, 2004.

[5] M. Ripeanu. Peer-to-peer architecture case study: Gnutella network. In *Proc. of the First International Conference on Peer-to-Peer*, 2001.

[6] S. Saroiu, P. K. Gummadi, and S. D. Gribble. Measuring and analyzing the characteristics of napster and gnutella hosts. *Multimedia Systems*, 9(2), 2003.

[7] Z. Song, Y. Labrou, and R. Masuoka. Dynamic service discovery and management in task computing. In *Proc. of the 1st Annual International Conf. on Mobile and Ubiquitous Systems:Network and Services(MobiQuitous '04)*.

[8] Z. Xu, M. Cai, and J. Dong. Service discovery in a dom-based middleware architecture. In *Proc. of the 2007 11th International Conference on Computer Supported Cooperative Work in Design*, 2007.