A Network Service Framework for Mobile Pervasive Computing

Enyi CHEN, Yuanchun SHI and Guangyou XU
Department of Computer Science and Technology
The State Key Laboratory on Intelligence System and Technology
Tsinghua University, Beijing 100084
Email: {chenenyi00@mails, shiyc@mail, xgy-dcs@mail}.tsinghua.edu.cn

Abstract—In this paper, we propose a framework for wireless/mobile pervasive computing to augment capabilities and to save scanty resources of mobile host by making full use of available resources in the surrounding. Firstly, a mobile device finds and utilizes the cheap computer as a surrogate to forward the service requests. Secondly, after the surrogate discovers the services, it will filter, synthesize or handoff them according to the context information. We also take consideration of robustness, scalability and load balancing by utilizing multi surrogates. Service descriptions are encoded by the eXtensible Markup Language (XML) technology to provide flexibility. The analytical results on the performance of our framework show that it has significant advantages in terms of reducing wireless bandwidth consumption, and etc.

Keywords—Ubiquitous Computing; Nomadic Computing; Service Discovery; Service Synthesis; Service Migration; Context-Awareness

I. INTRODUCTION

Pervasive computing [1][2] and nomadic computing [3] are the computing paradigm that enables mobile devices, taken by user, to aware of their surrounding and peers, and to be capable of effectively providing services to, or using services from, the surrounding and peers. These devices include laptops, mobile phone, personal data assistants (PDA), and etc., which should be connected to the surrounding and peers by wireless technologies, such as IEEE 802.11 or Bluetooth, due to user’s high mobility and nomaditiy.

On the other hand, the decreasing cost of networking technology and hardware devices is enabling the large-scale deployment. In the foreseeable future, whenever you go, there is high speed and width network with lots of cheap PCs or Workstations on the fixed infrastructure, and access point of the wireless network to facilitate mobile user to access services around him, as the figure 1 shows. The emergence of Smart Spaces [4] and Smart Building [5] is the example of these trends.

This paper gives a framework about how a mobile device utilizes the surrogate (an available resource in the surrounding) for service advertisement, discovery, filtration, synthesis and migration. A surrogate is a cheap computer (Desktop Computer or Workstation) in the surrounding. When just using one surrogate to perform those tasks, any transient load spikes or network glitches affecting the surrogate can have a dramatic effect on the performance, especially for lots of mobile devices in the surrounding. By using multi surrogates, the side effect is minimized. Zero-configuration, adaptive service filtration, seamless service migration, push and pull service discovery based context information are also our design aims.

Section 2 gives three possible scenarios of network service in the mobile/pervasive computing. Section 3 describes the challenges of our platform. Section 4 discusses the architecture in details. Section 5 evaluates its performance. Section 6 gives a prototype implementation in our lab based on the framework. Section 7 situates the work with a discussion of related systems. Finally, we summarize and mention future work in section 8.

II. APPLICATION SCENARIOS

Here are three possible scenarios, which are just for purposes of illustration and not limitation. We have implemented the first one as a test-bed for our platform.

Using service in the environment: When you go to ICCT2003 conference with your laptop, you meet with some friends in the same research field. You decide to make a presentation to them about your current research achievement. So you open your laptop, and search for a nearby projector. You get a response from a projector in the next door. After you go to the room with your friends, you launch the service and press “start” button. Consequently, your laptop’s screen is displayed on the wall by the projector so that you can give your presentation.

Using service provided by peers: When you fly to Beijing Capital International Airport from Toulouse Blagnac Airport, you may transfer in Paris Charles de Gaulle International Airport. It is very boring since you have to wait for another two hours before your next flight in the CDG lounge. So you use your wireless enabled PDA to search for “games”. You find another traveler who launches a “Chinese Chess” game, which is just your favorite. So you connect to his device and play with him.
Using augmented service by seamless migration: When you work in front of your computer, your mobile phone rings. After you pick it up, what make you surprised is that the caller’s live video is displayed on your computer screen though the user’s mobile phone can display and capture the live video. Vice versa, your live video is also displayed on the caller’s computer screen.

III. CHALLENGES

Practical realization of the above application scenarios will require us to solve many difficult design and implementation problems. The following is our consideration when the framework is designed:

Zero-configuration: Manual configuration is time-consuming and difficult. If a user remains in a new computing environment for only 15 minutes, he does not want to spend the first 10 minutes configuring his mobile device manually. For example, a user want print a document in a new environment. Firstly he must configure his network connection, find a printer in the environment, install the driver of the printer and print the document. The process is boring and unacceptable. The most convenient method for user is that an interface is returned after he send “print” request. Next, what done by the user is only find the printed file in his mobile device and click “Print” button. Auto-configuration is important [6].

Service Discovery: The second challenge is how to cope with a large variety of services in the environment. Services are the applications with well-known interfaces to perform computation or actions on behalf of users, such as projector, printer, music server, web service and etc. You can get details information in [7].

Context Awareness: The third challenge is to obtain the information needed to function in a context-aware manner and to implement a context-aware system. A pervasive computing system must be cognizant of its user’s state and surrounding, and must modify its behavior based on this information. Dey defined context as “any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application [8]”. Context-awareness enables mobile user to use the nearest and idle projector in the first scenario.

Adaptive Service Synthesis: Service Synthesis is the post-processing and filtering operation. For example, in the second scenario, the PDA taken by user has only a small and gray display screen. So the system must rescale and gray the picture of “Chinese Chess” to him. Another example is that TTS technology should be adopted if a mobile user “browses” notice in a new environment by mobile phone (Mobile phone’s screen is small so that it is hard to read a long notice on it. On contrary, its “speaker” function easily enables the user listen to the notice).

Seamless Service Mobility: Service Mobility is the capability of service being transported from one device to another by finding equivalent functions. For example, in the third scenario, though the user’s mobile phone can display and capture the live video, the quality is very low due to limited bandwidth of cellular communication technology. The environment has equivalent function and the bandwidth is very high, so video communication is switched to the wired infrastructure. Consequently, the user is provided with better service, such as high quality video. This service handoff must be seamless, unobtrusive and free-distraction.

IV. ARCHITECTURE OF THE PLATFORM

A. Overall Architecture

As mentioned in section 1, with the emergence of smart spaces [4] and the dramatic decrease in the price of computing, environment where are over-provisioned are becoming a reality. So, we have reason to believe that, in the foreseeable future, public spaces such as airport lounges and coffee shops being equipped with computing resources, much as comfortable chairs and table lamps are provided today. The main goal of our framework is to make full use of these resources for service advertisement, discovery, filtration, synthesis and migration. It is composed of four main components: mobile clients, surrogates, context monitor and services in the environment, as the figure 2 shows. Services are the applications with well-known interfaces to perform computation or actions on behalf of users, such as projector, printer, music server, web service and etc. Mobile clients want to discover the services that are available in the environment and the mobile peers. Hardware (desktop computer or workstation) in the wired infrastructure plays important role in the framework. We call it a surrogate of mobile clients since it temporarily assists mobile clients in doing everything about service. Dispatching Surrogate is a special surrogate, which takes charge of configuring the network connection of mobile clients and selecting a suitable surrogate for mobile clients according to the context information. Context Monitor provides context information to surrogate to improve the performance or reduce search space. In the following section, we will discuss these modules in details.

1 This is another research project in our institute by Weikai XIE. You can read his paper “Developing a Formal Model for Context-Aware Computing Infrastructure” if you want to get more details (submitting to be published).
B. Using Surrogate

Dispatching Surrogate

We design a special surrogate named Dispatching Surrogate, as the figure 3 shows, which has two main tasks: one is to configure the Mobile Clients’ network connection; the other is to select a suitable service surrogate for Mobile Clients.

In the mobile/pervasive paradigm, the user is a nomad [3], who may move between different environments. Zero-configuration of network connection is important when a nomad enters into a new environment, especially for the one who only stays for a short time since manual configuration is not only boring but also time-consuming. Dispatching Surrogate takes this charge, in other words, it maintains zero configurations for the mobile host. Once the mobile host enters into an environment, it will automatically get a network connection with the environment. For example, if a DHCP service exists in the Dispatching Surrogate, it will automatically give IP to a mobile host when it enters into the surrounding.

As you know, in the foreseeable future, there are lots of cheap PCs or workstations whenever you go. Just as the previous discusses, one of the main design aims of our platform is to make full use of all computing resources in the environment. But how does the framework locate a suitable surrogate in the surrounding to assist a mobile host to discover and advertise services? Firstly, all the surrogates report the consumption to the Dispatching Surrogate, and the later records these data into the Log Repository. Then if a mobile host wants to locate a surrogate in the surrounding, it sends a request to the Inference Engine. After parsing the Log Repository and Context Monitor, the Inference Engine will return a suitable surrogate to the mobile host according to current load on the all surrogates. So if a surrogate is experience heavy load, other surrogate may be unloaded and can be utilized to achieve faster response time.

Service Surrogate

Figure 4 illustrates the components of a service surrogate. When a mobile client wants to send a service discovery or advertisement packets, it firstly find a surrogate in the surrounding (as the previous section discusses). Once a surrogate gets request packets from the mobile client, the Daemon Manager will parse the type of packet. If the packet is for advertisement, the Daemon Manager will firstly store the service advertisement into the Services Repository, and then the Advertising Manager will advertise the packet to the wired network. If the packet is for discovery, the Daemon Manager will firstly search it in the Services Repository, and the Forwarding Manager will forward the discovery request to the wired network if the Daemon Manager can not find any requested service. After discovery, the return may be a service list, including many available services. Getting the context information from Context Monitor, the service surrogate may filter out several most suitable services to the Synthesizing and Migrating Manager. The later has two tasks: one is called adaptive service synthesis, which means to change the transferred data according to the capability of mobile device 2. For example, if the PDA only displays gray image, the Synthesizing and Migrating Manager will convert a color image into a gray image when the PDA device browses an album service. If a mobile phone wants to read news service from Internet, it is inconvenient to read it on the limited display screen. So Synthesizing and Migrating Manager returns the voice to mobile phone by adopting TTS technology to convert text into speech. Another task is called seamless service migration, which means to utilize the input/output device in the environment to empower the mobile host. Just the scenario 3 shows, though the mobile phone can capture and display the live video, but the quality is much less than the ability of the camera and screen provided by environment. So our framework automatically migrate the live video service from his mobile phone to his desktop computer.

C. Service Advertisement, Discovery and Caching

User should be able to package their own resources or information as services and offers them to others. Such services require simple and rich description capabilities along with cross-platform usability. Each service is a bundle of two components – a service description file that describe a service, and a service object. While the service object can in the form of a class file, a DLL, a Corba object

---

2 This is another research project in our institute by Chunyuan LIAO. You can read his paper [9] “AMTM: an Adaptive Multimedia Transport Model” if you want to get more details.
or even hardware such as input/output devices according to the different system, the description file is a plain text file containing complete information about the characteristics and functions of the service. Our system uses XML to describe service description file (the identifying information submitted by services). XML allows the encoding of arbitrary structures of hierarchical named values; this flexibility allows service providers to create descriptions that are tailored to their type of service, while additionally enabling “subtyping” via nesting of tags.

**Service Description Language**

Our language for describing services is an XML-based language called SDL (Service Description Language). SDL is similar to WSDL (Web Services Description Language) [10] except that we tried to come up with a template that can describe any type of services, as the figure 5 shows. Our language may not be as deep and powerful as WSDL, but it has its own merits of simplicity and a wide range of services that can be described.

```
<Service>
  <Name> Name of the service ->
  <ServiceName>ServiceName</ServiceName>
  <Type of service ->
  <ServiceType>ServiceType</ServiceType>
  <Keywords> Keywords - For matching search requests ->
  <Keyword>
    <Word>Word</Word>
  </Keyword>
  <Keywords Characteristics of the service ->
  <Property>
    <Name>Name</Name>
    <Description>Description</Description>
    <Value>Value</Value>
  </Property>
  <Properties>
  </Properties>
  <Available functions ->
  <Functions>
    <Function>
      <Name> Name of the function in the service object ->
      <Description>Description</Description>
      <Parameters required to invoke the function ->
      <Parameter>
        <Name>Name</Name>
        <Type>Type</Type>
        <Description>Description</Description>
      </Parameter>
      <Return type for the function ->
      <ReturnType>
        <Name>Name</Name>
        <Type>Type</Type>
        <Description>Description</Description>
      </ReturnType>
    </Function>
  </Functions>
</Service>
```

**Fig.5. Service Description Language**

The root of the template is the Service tag. It represents the start of the service definition. The first child is `ServiceName`. It is used to define a user-friendly name for the service. This is followed by `ServiceType`, like a URL address, which defines the type of the service, e.g. `Service.Device.Output.Projector`. The third child is `Keywords`. It includes key word that could be possible search types used during discovery phase. Each keyword is encapsulated by `Word` tag.

While the first four children of the `Service` tag are primarily used in service discovery, `Properties` and Functions are mainly used in service advertisement. `Properties` describe the characteristics of the service. Each service can have any number of properties. Each property is a combination of a name, a description and a value.

**Functions** are related to the actual usage of the service. A service can have any number of functions. Each function is a combination of a name, a description, some parameters and some return parameters.

**Service Caching**

It is effective that adoption of cache mechanism to decrease the discovery response time and to alleviate network burden. In our platform, the service cache is called `Service Repository`, which is a structure that enables surrogates to store their own services. It also allows them to maintain information about services that they have discovered or received via advertisements.

Our `Service Repository` adopts a tree-based hierarchical structure, which would be stored as an XML file and would be easily scalable. An example service tree is shown in the figure 6. As we move down the tree from root to the leaves, services become more specific. Services is classified as all, generic or specific. It is very convenient to discover any type of service according to mobile user’s requirement. For example, `Service.Devices.Output` is a generic discovery request, which means to discover all output devices in the surrounding. And `Service.Entertainment.Games.ChineseChess` is a specific discovery request, which means only to discover Chinese Chess service.

![Fig.6. Hierarchical Tree of Services Repository](image)

**Service Advertisement and Discovery**

If clients use the discovery process to discover services in the surrounding, we call it pull mechanism. But if servers use the advertisement process to periodically announce their registered services, we call it push mechanism. In our platform, we adopt both pull and push approach for fast and efficient discovery. All service discovery and advertisement is performed not by mobile host but by the surrogate.

Like the lease method in JINI [11], our system also uses this method to update all the services in the Service Repository. All advertisements and registrations are for a specific and fairly short period of time. Long running

---

3 In our system, any surrogate not only can act as a proxy for mobile device but also can provide his own service.
clients and services must renew their leases periodically, entities that crash are automatically removed from all surrogates when the lease expires.

Discovery process has four steps: a mobile client sends out discovery request to a surrogate; the surrogate firstly searches in his Service Repository, and forwards the discovery request to the wired network when no service is found on the local repository; on receiving a discovery request from a surrogate, other surrogates perform the service matching and return service response message if some services are found on their repository; the first surrogate filters, synthesizes or migrates the received service list and returns some suitable service to the mobile client according to the context information.

V. EVALUATION

In this section, we will evaluate the performance of our platform and discuss the advantages in making use of computing resources in the environment. As the analysis presented below, the adoption of surrogate can apparently save the wireless bandwidth consumption. When just using one surrogate, any transient load spikes or network glitches affecting the surrogate can have a dramatic effect on the performance, especially for lots of mobile devices in the surrounding. It is obvious that the side effect is minimized by using multi surrogates. The merits of zero-configuration, service filtration, service synthesis, and service migration according to context information are also obvious. So we emphasize on the benefit of utilizing surrogate due to restriction on paper’s length.

Suppose that there are $M$ services in the wired network and $N$ services in the wireless network, as the figure 7 shows. Even though the data path from a mobile device to the surrogate may include several wired nodes, the path still can be abstracted as one-hop wireless link by only taking into consideration of the packet-loss characteristics. So, we can suppose that $a$ is the packet loss rate in the wired network and that $b$ is the packet loss rate in the wireless network.

Suppose that a mobile discovery node multicast a discovery request into the environment. To increase the probability of correctly receiving the multicast packet, the packet needs to be sent to the receivers multiple times, considering the packet loss rate $a$ and $b$.

A. Without a Surrogate

Given that $i$ services have received a query packet correctly at the end of the $n$th transmission time, the probability that $i+j$ services receive the query packet correctly at the end of the $(n+1)$th transmission time is given by:

$$p_1(S_{i+j,n+1} | S_{i,n}) = C_{M-1}^j (1 - r_1)^j \cdot r_1^{M-i-j}$$

$$p_2(S_{i+j,n+1} | S_{i,n}) = C_{N-1}^j (1 - r_2)^j \cdot r_2^{N-i-j}$$

where $p_1$ and $p_2$ is the probability of from the sender to wired services and wireless service respectively. $r_1$ is the packet loss rate from sender to wired services, and $r_2$ is the packet loss rate from sender to wireless services, which are

$$r_1 = 1 - (1 - a)(1 - b)$$

and

$$r_2 = 1 - (1 - b)(1 - b)$$

Let $P(S_{k,n})$ denotes the probability that $k$ services have received the packet by the end of $n$th transmission time. Thus,

$$P_1(S_{k,n}) = \sum_{i=0}^{k} p_i(S_{i,n-1}) P_i(S_{i,n-1})$$

$$= \sum_{i=0}^{k} P_i(S_{i,n-1}) \cdot C_{M-1}^{k-i} \cdot (1 - r_1)^{k-i} \cdot r_1^{M-k}$$

$$P_2(S_{k,n}) = \sum_{i=0}^{k} P_i(S_{i,n-1}) P_i(S_{i,n-1})$$

$$= \sum_{i=0}^{k} P_2(S_{i,n-1}) \cdot C_{N-1}^{k-i} \cdot (1 - r_2)^{k-i} \cdot r_2^{N-k}$$

where $P_1, P_2$ is shown in the figure 8.
B. With a Surrogate

Given that $i$ services have received a query packet correctly at the end of the $n$th transmission time, the probability that $i + j$ services receive the query packet correctly at the end of the $(n + 1)$th transmission time is given by:

$$p_M(S_{i} | S_{i,n}) = C_{M-i}^i (1-r_3)^i \cdot r_3^{M-i-j}$$  \hspace{1cm} (7)

$$p_N(S_{i} | S_{i,n}) = C_{N-i}^i (1-r_4)^i \cdot r_4^{N-i-j}$$  \hspace{1cm} (8)

where $p_3$ and $p_4$ is the probability of from the surrogate to wired services and wireless service respectively, $r_3$ is the packet loss rate from surrogate to wired services, and $r_4$ is the packet loss rate from surrogate to wireless services, which are

$$r_3 = 1-a$$  \hspace{1cm} (9)

$$r_4 = 1-b$$  \hspace{1cm} (10)

Let $P(k, n)$ denotes the probability that $k$ services have received the packet by the end of $n$th transmission time. Thus,

$$P_M(S_{k,n}) = \sum_{i=0}^{k} P_3(S_{i} | S_{i,n-1}) \cdot P_M(S_{i,n-1})$$

$$= \sum_{i=0}^{k} P_3(S_{i,n-1}) \cdot C_{M-i}^i (1-r_3)^i \cdot r_3^{M-k}$$  \hspace{1cm} (11)

$$P_N(S_{k,n}) = \sum_{i=0}^{k} P_4(S_{i} | S_{i,n-1}) \cdot P_N(S_{i,n-1})$$

$$= \sum_{i=0}^{k} P_4(S_{i,n-1}) \cdot C_{N-i}^i (1-r_4)^i \cdot r_4^{N-k}$$  \hspace{1cm} (12)

where $P_{3,4}(s_{i,0}) = \begin{cases} 1, i = 0 \\ 0, i > 0 \end{cases}$

The probability that all $M$ wired services have received query packet correctly by the end of $n$th transmission is $P_M(S_{M,n})$. The probability that all $N$ wireless services have received query packet correctly by the end of $n$th transmission is $P_N(S_{N,n})$. Let $Q_2(S_{M+N,n})$ denotes the probability that all $M$ wired services and all $N$ wireless services have received query packet correctly by the end of $n$th transmission. Thus

$$Q_2(S_{M+N,n}) = \text{Min}\{P_M(S_{M,n}), P_N(S_{N,n})\}$$

Let $a = 0.1$, $b = 0.2$. $Q_2(S_{M+N,n})$ is shown in the figure 9.

From the figure 8 and figure 9, when the number of query or advertisement packets is same, adoption of surrogate can increase greatly the probability that all $M$ wired services and all $N$ wireless services correctly receive packets.

VI. IMPLEMENTATION

We are still interested in exploring the application of pervasive computing or ubiquitous computing (UbiComp). We have implemented Smart Classroom Project, a test-bed for Smart Spaces, a research branch of UbiComp, which takes us last four years. Currently, we can launch and run our Smart Classroom, just as the demo videos on http://media.cs.tsinghua.edu.cn/~pervasive show. In the previous work, we don’t consider the adoption of wireless and handhold devices taken by mobile user. However, with the development of computing and network technology, the portable computing devices will be proliferated. It is very interesting and useful if mobile user can use mobile device to easily and unobtrusively get service from the environment (Smart Spaces). This lead to the project described on this paper.
Three are two meeting rooms in our institute. Almost a hundred graduate students, their supervisors and lots of invited experts often give research presentation or take seminar in these two rooms. In the past, when someone wants to show his slides, he must take his laptop to the front of the room and connect it to a VGA cable. And more unfortunately, the projector may not work for some laptops, so the shown files must be copy to another laptop. It is very boring and annoying. So we develop a prototype for universal access the projectors in our institute, as the figure 10 shows.

Currently, suppose there are 10 participants in on room. If someone wants to use the projector, he just open a web page and fill “Projector”, our system will feedback the above interface to him according to the context information, such as user’s location, projector’s state and etc. All that he should do in the next step is to open a file and then click the “start project” button.

![Flow and Interface of Prototype](image)

VII. RELATED WORKS

Conventional network service research is mainly based on wired network. Best known system includes Berkeley’s Ninja [12] for a secure wide-area Service Discovery Service, IETF Service Location Protocol (SLP) [13], Sun Microsystems’ Jini [11] and etc, but they can not support mobile device very well, especially the small one, such as mobile phone and PDA due to their earlier design and development.

It is a trend that the portable computing devices will be proliferated. So many researchers begin to develop mobile ad-hoc service system, such as IBM’s DEAPspace [14]. Without doubt, it is helpful for mobile user. However, with the emergence of Smart Spaces [4], the environment will be over-provisioned with computing resources and daily work and study hardware. If these resources can not be made full use of, it is extravagant for them. So we develop the initial network service platform for mobile pervasive computing.

Our platform is similar to CMU’s “Cyber Foraging” [15], which is also ongoing research. The difference between us is that we take consideration of zero-configuration, service filtration, service synthesis and service migration.

VIII. CONCLUSION

In this paper, we have described our initial network service platform for mobile pervasive computing. After giving three possible application scenarios, we discuss several challenges in designing the platform. Our main goal is to make full use of available resources in the surrounding so as to augment capabilities and to save scanty resources of mobile host. In other words, a mobile device utilizes the surrogate, an available computing resource, for service advertisement, discovery, filtration, synthesis and migration. By using multi surrogates, the side effect of transient load spikes or network glitches is minimized. Zero-configuration, adaptive service filtration, seamless service migration, push and pull service discovery based context information are also discussed. Performance evaluation proves our platform has advantages in terms of reducing wireless bandwidth consumption, ant etc. Our implementation of prototype gives an example of the usage of our platform.

It is not easy to implement a whole platform. There are lots things to do. Context-aware computing model, adaptive content filtration, application’s seamless mobility, design of killer-application and intelligent service matching mechanism and etc are our future works.

REFERENCES

[14] Hermann Reto, etc. DEAPspace – Transient ad hoc Networking of Pervasive Devices. PP. 411-428