Vol. 45 No. 12 IPSJ Journal Dec. 2004

Regular Paper

Service Initiation and Migration for Real-time Communication Services in the Ubiquitous Networking Environment

Naoki Imai,[†] Manabu Isomura,[†] Hiroki Horiuchi[†] and Sadao Obana[††]

We present a service initiation and migration system for real-time communication services in a ubiquitous networking environment. The key concepts are (1) the isolation of a few functions as common components with SIP protocol from real-time applications, (2) the incorporation of service migration into service initiation so that users can change resources at any time during communication, and (3) service negotiation which enables all user requests and situations to be reflected in the service. Therefore, users can enjoy real-time communication even when they face resource heterogeneity and changeable situations in the ubiquitous networking environment. The implementation of this system and the experimental results are also described, which demonstrate that users can switch devices and applications seamlessly without terminating the session.

1. Introduction

The rapid spread of always-on connections such as ADSL (Asymmetric Digital Subscriber Line) and FTTH (Fiber to the Home) has increased the expectations and demands for real-time applications such as voice phone, TV phone, and instant messengers. Demand will be further accelerated by the development and spread of mobile communication environments including cheap, high-speed wireless LAN or 3G cellular phones.

At the same time, following up on the ubiquity and heterogeneity of networked resources such as devices and applications, the next-generation networking environment, often referred to as a ubiquitous networking environment, has just started to emerge. In this environment, ubiquity enables a user to enjoy various services on the network anytime and anywhere. On the other hand, owing to heterogeneity, the user must select from among many suitable resources according to the situation. These resources that were selected by the user might no longer remain optimal as the surrounding situation changes. It is therefore preferable for both service initiation and migration to be provided in an integrated approach for a future ubiquitous networking environment.

As for service initiation, SIP (Session Initiation Protocol)[1] is a major protocol standardized by IETF. With the protocols in the sequential drafts[2),3)], SIP provides a service on which users' preferences are reflected having the benefit of SIP proxy. However, it is impractical to store and manage all information about users' devices in an intensive fashion because they may belong to different management domains. Additionally, the policy or preference information in the server is rather static because we have no precise, low cost way of determining a user's changeable context.

To achieve service migration, call transfer functions such as the SIP Refer method[4] enable users to switch devices. However, since handoffs must be executed one by one, it takes time and effort if, for example, both users want to change their devices simultaneously. In the ubiquitous networking environment where simultaneous handoffs often occur, this wasteful procedure (SIP Refer method) causes, in the worst case, handoff failures. Consider a scenario where two users talking on their PDAs by voice phone would like to change to a TV phone on their PCs. In that case, the SIP Refer method will not provide a simultaneous handoff. These two users therefore have no other recourse but to restart their conversation on the TV phones by PC after terminating communication on the PDAs. This is necessary because communication between a voice phone (PDA) and a TV phone (PC) is impossible.

In this paper, we propose a service initiation and migration system that achieves a service configuration reflecting both the user situations and preferences, and simultaneous handoff be-

† KDDI R&D Laboratories Inc.
†† Advanced Telecommunications Research Institute International (ATR)

tween devices for real-time communication services. The rest of this paper is organized as follows. Section 2 describes a sample scenario for a real-time communication service followed by the requirements and the design policy of our proposal. Section 3 describes the service initiation and migration system, and Section 4 describes its implementation and performance evaluation. Section 5 describes related studies. Finally, in Section 6 we offer our conclusions.

2. Requirements and Design Policy

This paper aims at achieving real-time communication services. A practical example of a service scenario is shown as follows (see **Fig. 1**).

Alice is watching television in the living room in her house and wants to talk with Bob. There are two devices at her disposal. One is a mobile device such as a cellular phone or handheld PDA for making voice phone calls. The other is a desktop PC that is equipped with a TV phone as well as a voice phone. Alice wants to talk to Bob on the TV phone so she makes a call with a service list stating that here primary choice is the TV phone and her secondary choice is voice phone on her mobile device. When Bob receives the incoming call from Alice, the service list appears on his mobile device. He chooses voice phone since he is not at home. This starts the mobile device's voice phone application. Bob walks home while talking to Alice. Since he also has a desktop PC with a TV phone, as soon as he arrives home, he transmits a service list over his mobile device stating that his first choice device is now the TV phone. Alice receives this list and presses the "migration" button on her PDA while moving to her PC. Their telephone call immediately shifts to the TV phone using PCs, and they can continue their conversation...

Requirements for the service initiation and migration system based on the above scenario are shown in the following subsection.

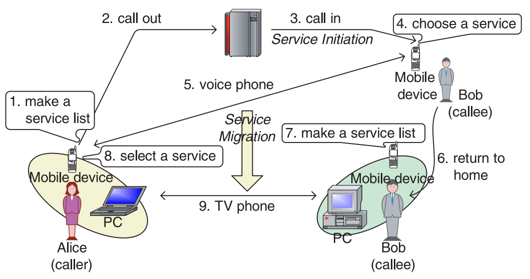


Fig. 1 An example of a service scenario.

2.1 Requirements

In the service scenario, Alice and Bob continue real-time communication while switching devices and applications according to their situation. The requirements for achieving this service are described below in terms of resource heterogeneity and ubiquity.

[Req.1] Making a call using a preferable device
At the beginning of the real-time communication service, the callee's present situation must be predicted accurately in order to select appropriate resources for connection to the callee. If the wrong resources are selected, it will be impossible to even setup a connection. However, this complicated selection operation is too large a burden to impose on the user, especially in the ubiquitous networking environment. Therefore, the outgoing call must be made simply, using the most appropriate device from the caller's point of view.

[Req.2] Consideration of both users' situations
In the ubiquitous networking environment, a user usually has two or more means for real-time communication. However not all of these resources will meet the user's needs. These resources will instead be limited according to the situation in which the user wants to make use of the service. Therefore, all user requests need to be reflected as much as possible when determining the method for performing a service, which is referred to as the service configuration in this paper.

[Req.3] Dynamic and flexible resource handoff
Even though the service begins by using the resources that meet Req.2, those resources do not necessarily remain optimal until the end of the service. In the ubiquitous networking environment, dynamic changes in the user situation occur frequently during a service. Restarting the service with new resources after having terminated the ongoing service is inefficient. The user must therefore migrate to new resources that are more suited to the current situation without breaking off the real-time communication service. To provide flexible communication service, simultaneous handoff of both users must also be considered.

2.2 Design policy

2.2.1 Overview

This subsection discusses our design policy to meet the above requirements.

Connectivity to a callee

In the ubiquitous networking environment, various applications on heterogeneous devices are available to the user. To meet Req.1, it must be possible to establish communication with a callee from an arbitrary device selected by the caller. Either of the following two approaches may be used.

[**Approach 1-a**] Users register their policy and/or preference with the policy server in the network. The registered information is referred to when an incoming call occurs. The call is directed to the appropriate device according to the callee's information after being appropriately translated, if necessary.

[**Approach 1-b**] The same real-time application is installed into all user devices so that two arbitrary devices can be connected with each other.

Approach 1-a has two problems. Firstly, since all the user resources might not belong to the same domain, complex coordination is required between multiple servers. Secondly, although the user context may change dynamically, it is difficult to precisely define the users' present situation at a low cost. Therefore, incoming calls are routed only according to fixed rules configured in advance in the server.

Approach 1-b also has a problem. If these all devices have a common real-time application using a platform like Java[5)], then Req.1 is technically satisfied because all devices can communicate with each other. However, it is difficult to develop and spread a uniform application for all devices that might range from low function mobile phones or PDAs to high performance PCs.

In this paper, we adopted approach 1-b', a modification of approach 1-b which aims to consider resource ubiquity and heterogeneity. Approach 1-b' separates connection setup and session management functions from real-time applications. This is a different approach from current applications where the connection setup and session management functions are not separated. This approach enables the suitable development of applications for devices with different capabilities, and dynamic switching using session information. On the other hand, the **connection setup function** must be installed as a common function in all the devices.

Servers are still required even in approach 1-b' for locating the callee in the connection setup. In this paper, we assume that all users have a mobile device such as a PDA or a cellular phone, and that only that device is registered with the server to receive calls, rather than all other devices in the user environment. This restriction can reduce the load on both the network and the server. Note that the device by which a callee actually starts a real-time communication service is not necessarily the registered one since the user can switch to another device even during service initiation as described later.

Service negotiation

To meet Req.2, the service configuration must be determined between the users. There are two approaches to deciding the configuration.

[**Approach 2-a**] Service configuration is determined by the network server executing a matching process for the end users' available resources.

[**Approach 2-b**] The service configuration is determined by the exchange of resource candidate information between the end users.

Approach 2-a suffers from the same problems as approach 1-a. Therefore, we adopted approach 2-b, and service negotiation consists of exchanging requests containing information about the service configuration candidates. The **service negotiation function** must be installed in all the devices because service negotiation occurs whenever users initiate or migrate service. To execute service negotiation, the **resource management function** is required. This function allows the user to identify the available resources and make a service list of service configuration candidates in order of priority.

Resource Handoff

To meet Req.3, a function must be provided that executes the resource handoff in accordance with the service negotiation results. It must also be possible to execute a resource handoff during service initiation because the callee might want to change his device in response to the service negotiation. In our system, the devices involved in resource handoff directly communicate with each other instead of communicating through a server. This is because the sequence using a server becomes complicated especially in the case of simultaneous handoffs where many resources interact with each other. The end-to-end signaling approach can therefore prevent the handoff delay from increasing by reducing the required interaction.

Before starting resource handoff, the user devices must exchange information regarding the new connection points (i.e., IP address), and an application in the new devices must be activated. Therefore, the **resource handoff function** is also installed into all the devices.

2.2.2 Incorporating Service Migration into Service Initiation

Figure 2 shows service initiation and migration throughout the entire service flow. Service initiation is divided into three phases: a connection setup phase, a service negotiation phase, and a resource handoff phase. This implies that service initiation includes the service migration operation, which is sub-divided into service negotiation phase and resource handoff phase. This process is what allows a user to switch devices even during service initiation. For example, a user making a call by mobile phone can immediately select and start a TV phone on his PC. The service initiation subdivision makes the service migration mechanism reusable and at the same time provides flexible service migration for the users.

It is also essential that the protocol used by the common component does not impose a heavy load on a device, and that it can be easily ported for various platforms. This is essential because there are many types of devices ranging from low function mobile phones or PDAs to high performance PCs, and there are also many operating systems for them. We therefore adopted SIP [1] as the common component. Here, SIP is a text-based, lightweight protocol like HTTP (Hyper Text Transfer Protocol) [6]. In addition to session initiation, SIP is suitable for service negotiation in which text-based messages are exchanged since it can be easily implemented by using the INFO method [7] after establishing a session. SIP also provides great versatility since it is a protocol standardized in IETF, it is implemented for various operating systems, and it is broadly used in current applications such as VoIP.

2.3 Function Components

Figure 3 shows the function components on a device and the interaction between them based on the descriptions in Sections 2.2.1 and 2.2.2. Each device has a common component including a Service Manager with a SIP stack, a Resource Manager, and a Telephony Service providing GUI (Graphical User Interface) for resource handoffs. These also have various real-time applications.

The Service Manager is the main function of the common component. It is composed of the following functions: a connection setup function, a service negotiation function, and a resource handoff function. These functions use SIP protocol. Therefore, they are integrated into one common module, referred to as the Service Manager. Each component in a device can communicate with each other through the Service Manager.

The Resource Manager has a resource management function. It stores resource information within the device, and collects the information from those devices available to the user. It also informs the Service Manager of resource information when receiving a request.

In addition to previous common components, it is necessary to include a function for interaction with the user. The Telephony Service provides a GUI that allows the user to make or terminate calls, and which also creates a service list.

Separation of real-time applications and the session management function requires applications to be modified for the system as described below.

- The Service Manager must be able to find a way of starting and terminating the application and necessary parameters.

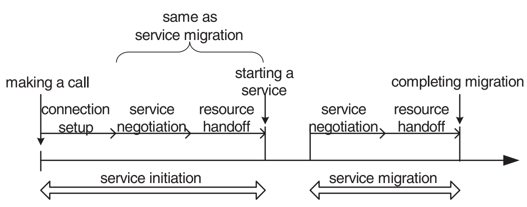


Fig. 2 Service initiation and migration.

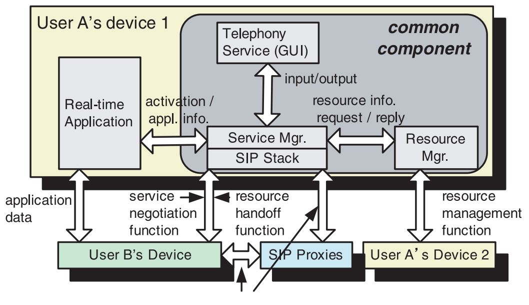


Fig. 3 Function components on each device.

- The Service Manager must be able to start and terminate the application.
- The Service Manager must be able to export and import state information from and to the application to resume it, if necessary.

To meet the first requirement listed above, the application must prepare an information file which describes the execution path for the application with arguments or terminate commands. Although our system requires some modification to the application, the Service Manager is common to all applications, which enables them to be switched.

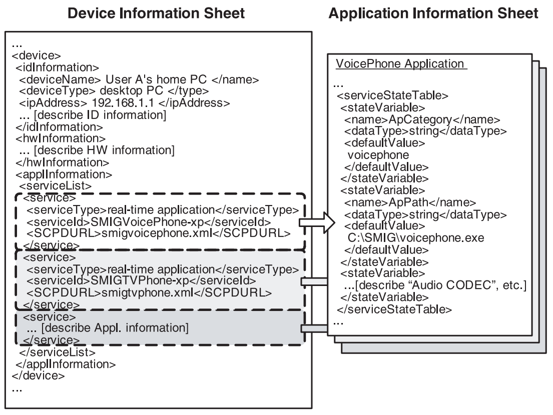
3. Service Initiation and Migration for Real-time Communication Services

3.1 Resource Management

The user must identify available resources before which equipment is most suitable for performing the services. Service discovery and management schemes are therefore necessary. In our system, each device manages its own resources, so there are no central servers. This is important because intensive management in a central server creates high loads both on the network and on the server. Another reason is that service migration occurs between resources that are close to and can be found directly from each other. In the device discovery scheme, we impose no restrictions as long as the required information can be gathered, for example, in peer-to-peer communication via Bluetooth[8] or ZigBee[9]. We therefore focus especially on the resource management below.

The Resource Manager stores resource information in a format compatible with UPnP (Universal Plug and Play). The information description is divided into two types: device information and application information. These are described in XML (eXtended Markup Language) and have a hierarchical relation as shown in **Fig. 4**.

The device information sheet describes identification (ID) information, hardware (HW) information and application information. ID information includes items such as the given name of the device, a device type for easy identification (such as "PDA", "desktop PC", "cellular phone", etc.), and a current IP address. HW information includes the CPU, memory size, screen resolution, I/O devices, and other data. A new category of information can be easily added to both the ID information and the HW information by defining a new XML tag. The application list includes an application category, application ID at the device, and the file name of the application information sheets. In this paper, the application category is set to "real-time application". Ideally, those applications belonging to the same category can be changed continuously.



```
...
<device>
 <idInformation>
  <deviceName> User A's home PC </name>
  <deviceType> desktop PC </type>
  <ipAddress> 192.168.1.1 </ipAddress>
  ... [describe ID information]
 </idInformation>
 <hwInformation>
  ... [describe HW information]
 </hwInformation>
 <applInformation>
  <serviceList>
  <service>
   <serviceType>real-time application</serviceType>
   <serviceId>SMIGVoicePhone-xp</serviceId>
   <SCPDURL>smigvoicephone.xml</SCPDURL>
  </service>
  <service>
   <serviceType>real-time application</serviceType>
   <serviceId>SMIGTVPhone-xp</serviceId>
   <SCPDURL>smigtvphone.xml</SCPDURL>
  </service>
  <service>
   ... [describe Appl. information]
  </service>
  </serviceList>
 </applInformation>
</device>
...
```

```
...
<serviceStateTable>
 <stateVariable>
  <name>ApCategory</name>
  <dataType>string</dataType>
  <defaultValue>
   voicephone
  </defaultValue>
 </stateVariable>
 <stateVariable>
  <name>ApPath</name>
  <dataType>string</dataType>
  <defaultValue>
   C:\SMIG\voicephone.exe
  </defaultValue>
 </stateVariable>
 <stateVariable>
  ...[describe "Audio CODEC", etc.]
 </stateVariable>
</serviceStateTable>
...
```

Fig. 4 Resource description in XML.

Application information sheets are created for each application. They describe an application type (such as "TV phone", "voice phone", "messaging", etc.), a program path and arguments for activation, media codec, application and other parameters. Application production information including the manufacturer and name is also necessary to uniquely identify an application in service negotiation between the users. In addition to the above, new types of information can also be added easily to the application information sheet.

As described in Section 2.3, the Service Manager must refer to the application information all the time. For example, when it starts an application, the application execution path and some arguments are required. Therefore, the device information sheet must be updated so that the Service Manager can find the correct file name corresponding to the application information sheets. However, it is too large a burden for a user to revise the device information sheet every time a new application is installed. In the system, it is automatically imported to or deleted from the device information sheet whenever the application is installed or uninstalled. This automation alleviates the user's configuration load.

3.2 Connection Setup

In the proposed system, both end users' devices are connected with each other during the

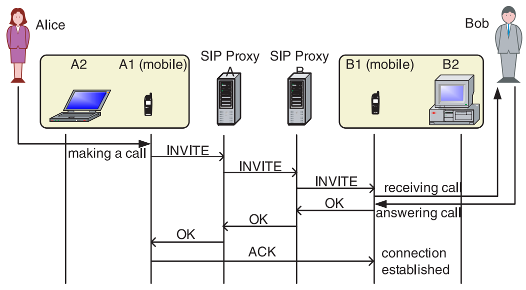


Fig. 5 Connection setup.

connection setup phase. However, no application media packets are exchanged at that time. Exchange starts after the service negotiation has determined the service configuration. This implies that the connection established by connection setup function will be used as a signaling channel for service negotiation.

Figure 5 shows the connection setup sequence using SIP. In addition, each device registers its current location with a registrar (not shown in Fig. 5) the same as typical applications with SIP. However, it is important to note once again that the system only makes a portable device issue a registration packet in order to reduce the load on the network and registrar.

Firstly, Alice sends an INVITE request to a local SIP Proxy A via an arbitrary device (A1 in the figure). This request is forwarded to B1 through the SIP proxy B. Here, B1 (Bob's portable device) alerts him to the incoming call from Alice so that he can decide whether to answer the call. If he answers the phone, an OK response is transferred to A1 through Proxy B and Proxy A. After receiving the response, A1 sends an acknowledgement message, ACK, to B1 to confirm its receipt.

In our system, the SDP (Session Description Protocol)[10] media description which, in typical SIP applications, determines the media type is not attached to INVITE requests or OK responses because it still has not been determined which application to use. The Context-Length field in the INVITE request and OK response are set to 0 in the connection setup.

Standard SIP applications determine several parameters for communication within the connection setup. Therefore, users can start to communicate instantly after exchanging parameters. Our method, on the other hand, has several extra procedures including service negotiation which must be completed before the exchange of application data can start. However, the service negotiation system enables both users to select and change resources even during service initiation. In addition, simultaneous resource handoff can be achieved by service negotiation while it is difficult to achieve with SIP REFER method as mentioned in Section 1.

3.3 Service Negotiation

The service configuration is determined by service negotiation. Service negotiation is made up of three phases: a service list creation phase, a configuration negotiation phase, and a service information exchange phase.

A service list created by a user using the Telephony Service includes candidates for service configuration in order of priority. (Telephony Service is illustrated in more detail in Section 4.1). For each candidate, application production information and the application's name are listed, enabling the application to be identified by both the users. Parameter candidates for execution are also included. To make a service list, resource information is collected by the Resource Manager on the current device. When making a call, a service list is needed to determine the service configuration. Alice makes an actual list before making her call in Fig. 5. As soon as the connection is setup, the configuration negotiation is executed based on this service list.

Once the communication service begins, both users can initiate service negotiation at any time. Note that there is no difference between these negotiation sequences. In the following, we make Alice the initiator of the service negotiation for the sake of simplicity.

The configuration negotiation is based on an offer/answer or offer/re-offer/answer model similar to the exchange of SDP messages. **Figure 6** shows the sequence of configuration negotiation and exchange of information for resource handoff. In our system, configuration negotiation is limited to a three-phase exchange to prevent endless continuation.

Arriving at Bob's device, the service list is displayed to him. He chooses one item from the list (Case 1), terminates the negotiation (Case 2), or makes his own service list (Case 3, Case 4). For example, Case 1 occurs when Bob prefers a configuration item described in the list provided by Alice. The other cases occur when there are no configurations meeting Bob's needs, or when Bob has no resources

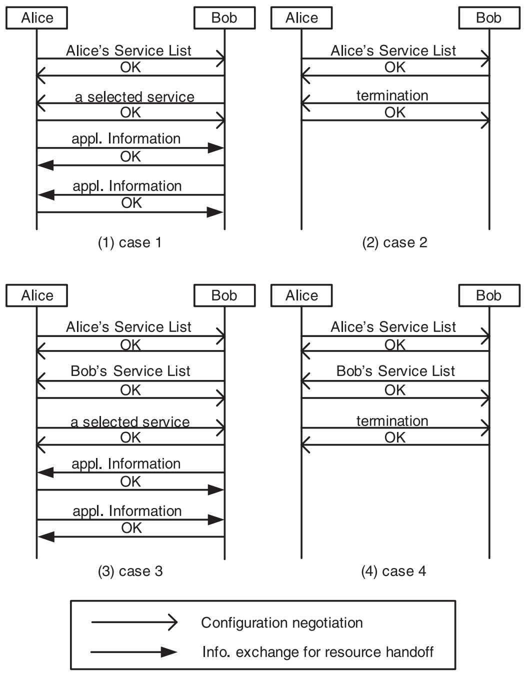


Fig. 6 Service negotiation.

available to meet the provided configurations. In Case 3 and Case 4, Bob creates and sends back a new service list that does not include the service configuration previously presented by Alice. Alice then has two alternatives. One alternative is to choose one service from the list as Bob did in Case 1 (Case 3). The other is to terminate the negotiation as Bob did in Case 2 (Case 4). As described above, Alice does not re-create a new service list.

The service list includes the application production information and application information such as available video and voice codec lists. When the correspondent user decides on a service, that user must designate the selectable items (e.g., video and voice codec), as is the case with an exchange of SDP messages.

When the service configuration is fixed, the information required for the resource handoff is exchanged. This information includes the IP address of the new device and port numbers if necessary. Exchanging service lists and information for the resource handoff is conducted by the INFO method in SIP.

Before the Service Manager offers a service list to a user through the Telephony Service, information about the availability of surrounding devices is obtained from Resource Manager. Therefore, the service list configured and sent by a user reflects her situation and needs. As

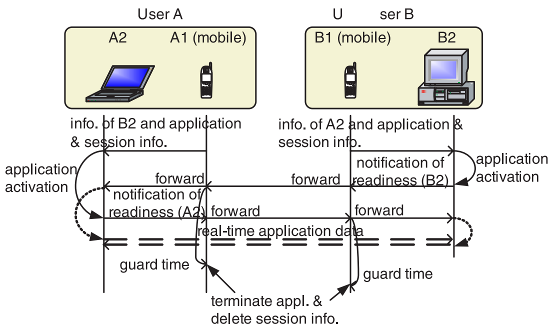

User A
U ser B
A2
A1 (mobile)
B1 (mobile)
B2
info. of B2 and application & session info.
info. of A2 and application & session info.
application activation
notification of readiness (B2)
application activation
forward
forward
notification of readiness (A2)
forward
forward
real-time application data
guard time
guard time
terminate appl. & delete session info.

Fig. 7 Resource handoff.

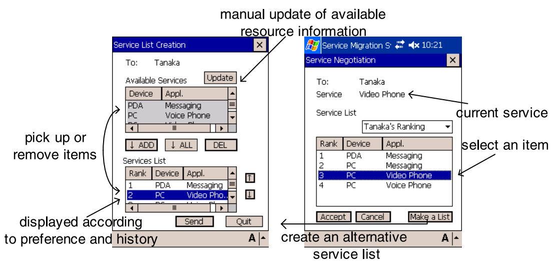
the same process is executed at the received user side, both users' situation and needs can be reflected in the actual service configuration.

In the case of conventional SIP applications, negotiation with the SDP determines several parameters used in the given application. In our system with service negotiation, users can determine which application will be executed and which device will be used. This is enabled by the interaction between Service Manager and Resource Manager.

3.4 Resource Handoff

Figure 7 illustrates the sequence in resource handoff between devices and applications that follows the service negotiation presented in Fig. 6. Session information, including the session status established by connection setup, and the correspondent's information including the new IP address, port number, and application parameters, are transferred from the old device to the new device of each user. New devices activate the application according to the information received. When the device is ready to start communication, it notifies the correspondents' old devices. This message is exchanged between each device in order to determine when to start sending data packets from the new application.

In the system, guard time is configured in advance in the Service Manager. The Service Manager terminates the current application and cleans up session information after the guard time has passed. The Service Manager starts counting Guard Time when it receives the packets denoting notification of readiness. This enables users to communicate with each other through the current applications until the new application is prepared so that data packets between the users continue to flow during resource handoff. This is referred to as a seamless resource handoff.


manual update of available resource information
pick up or remove items
displayed according to preference and history
current service
select an item
create an alternative service list
Service List Creation
Update
↓ ADD
↓ ALL
DEL
Send
Quit
Service Migration S 10:21
Service Negotiation
Accept
Cancel
Make a List

Service List Creation

To: Tanaka

Available Services

Device	Appl.
PDA	Messaging
PC	Voice Phone

Services List

Rank	Device	Appl.
1	PDA	Messaging
2	PC	Video Pho..

Service Negotiation

To: Tanaka
Service Video Phone

Service List Tanaka's Ranking

Rank	Device	Appl.
1	PDA	Messaging
2	PC	Messaging
3	PC	Video Phone
4	PC	Voice Phone

(a) service list creation (b) configuration negotiation

Fig. 8 Screenshots of the Telephony Service on a PDA.

4. Prototype Implementation

In this section, we present a prototype implementation for using and evaluating service migration performance. We implemented the common component for the service initiation and service migration system for PC and PDA. Some applications were also developed as proof of concept demonstrators.

4.1 Telephony Service

The Telephony Service is used when a user makes, receives, and terminates a call, and creates and answers a service list. **Figure 8** shows two example screenshots of the Telephony Service on a PDA. Figure 8 (a) illustrates the GUI for creating a service list. Available resources are displayed in the upper part of the screen, and a user picks the preferable configurations. Although resource information is collected by the Resource Manager at regular intervals, which can be configured by the user, the user can manually renew this information by pressing the "update" button. Since the Telephony Service can store and process user preferences and communication history, the service list can be created automatically based on the recorded data. The service list firstly displayed in the lower part of the screen is made based on the user personal communication history.

Figure 8 is a screenshot presented to a user receiving a service list. Configuration items can be ranked according to both the correspondent and user preferences. If the receiving user wants to create an alternative service list by himself (Case 3 or Case 4 in Fig. 6), he can change to the service list creation screen (Fig. 8 (a)) by pressing the "Make a List" button.

4.2 Testbed Network and Scenarios for Experimentation

As shown in **Fig. 9**, the experimental network consists of two subnets separated by an L3 switch, and each subnet has a SIP Proxy, a PC, and a PDA connected to each other. The devices have different operating systems: Linux for the SIP Proxies, Windows XP Professional for the PCs, and Windows Mobile 2003 for the Pocket PC for PDAs. A user can enjoy voice phone and instant messaging by PDA, as well as TV phone using a PC. Users can even communicate between a PC and a PDA if they are using voice phone or instant messaging. SIP Proxies and PCs have an Ethernet interface while the PDA is connected by an IEEE 802.11b wireless LAN to the network. Location registrars are combined with SIP Proxies in the environment. Users can initiate the real-time communication service via an arbitrary device and the call always arrives at the PDA.

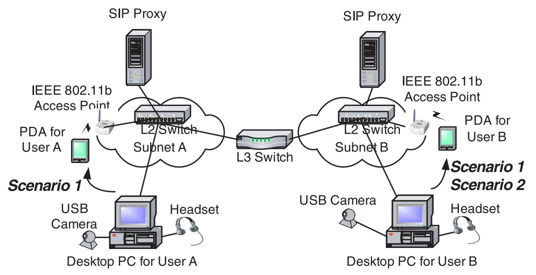


Fig. 9 Testbed network.

We adopted the following scenarios to measure the service migration period.

[Scenario 1] User A and User B are operating TV phones on PCs. They then change to voice communication on PDAs.

[Scenario 2] User A and User B are operating TV phones on PCs. They change to voice communication, User A continues to use a PC while User B switches to a PDA.

4.3 Experimental results

We measured the time for service negotiation and resource handoff for each scenario. We adopted Case 1 in Fig. 6 as the service negotiation in which the configuration negotiation finished in two-phase exchange. Furthermore, to eliminate the time needed for manual operation, creation of the service list was excluded from the measurement and it was answered automatically.

Figure 10 illustrates the sequence in Scenario 1. The service list contains application candidates, each of which is composed of parameter candidates including application production information and codec (1). A VoIP application with PDA_B is selected by User B (automatically in the experiment), and the selected application production information and

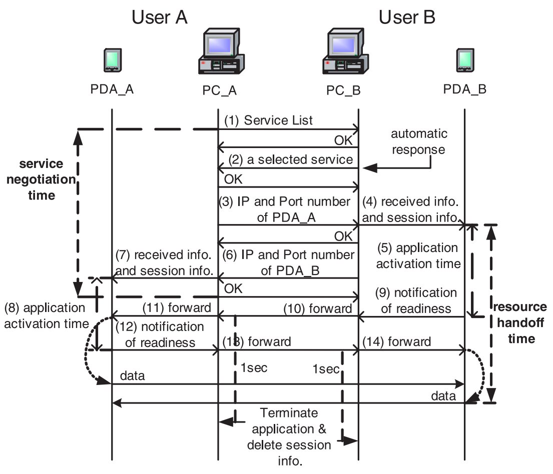

User A
User B
PDA_A
PC_A
PC_B
PDA_B
service negotiation time
(1) Service List
OK
(2) a selected service
OK
automatic response
(3) IP and Port number of PDA_A
(4) received info. and session info.
OK
(5) application activation time
(6) IP and Port number of PDA_B
(7) received info. and session info.
OK
(8) application activation time
(9) notification of readiness
(10) forward
(11) forward
(12) notification of readiness
(13) forward
(14) forward
resource handoff time
1sec
1sec
data
data
Terminate application & delete session info.

Fig. 10 Sequence in Scenario 1.

codec are sent back to PC_A (2). At this point, as application and codec are finally determined, PC_A sends the final application name, IP address and Port number of PDA_A (3). The received information and session information are forwarded to PDA_B (4), and PDA_B activates the application (5). Session information includes state information established through the SIP connection setup in order to distinguish the session from other sessions. PC_B sends the IP address and Port number of PDA_B (6), which is forwarded to PDA_A (7). Then, PDA_A activates the application (8).

When the VoIP application in PDA_B is ready, it notifies PC_B, which is relayed to PDA_A through PC_A (9-11). PDA_A also sends a notification packet to PC_A, which is similarly transferred to PDA_B through PC_B (12-14). Each application starts to send data packets when the notification packet is received (11, 14). Data packets start to be sent when the ready packet is received. In the experiment, the guard time for terminating the old application and deleting session information is set to 1s after receiving the notification packet.

Table 1 shows the average time calculated from ten measurements. Service negotiation starts when User A sends a service list and ends when User A receives service information for the resource handoff. Resource handoff starts when a message is issued from any old device to a new device and ends when voice communication packets are received by the new devices. The total time shows the time between starting service negotiation and finishing resource handoff. It is important to note that the total time is different from the sum of the service negotiation time and resource handoff time. This is because the resource handoff operation actually begins before service negotiation is completed as illustrated in Fig. 10.

Table 1 Measurement results.

	Service negotiation	Resource handoff	Total
Scenario 1	245 ms	313 ms	518 ms
Scenario 2	244 ms	274 ms	480 ms

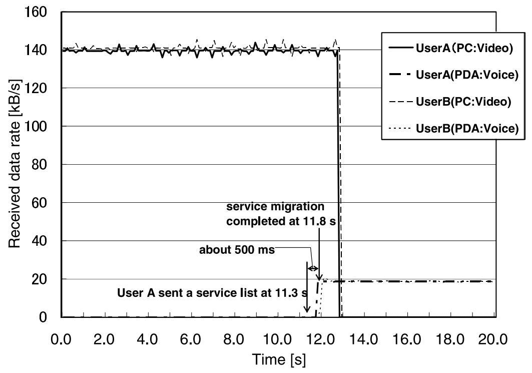


Fig. 11 Received data rate during service migration.

Both of the service negotiation times are almost equal because the negotiation operation is exactly the same in the two scenarios. In contrast, the resource handoff time in Scenario 2 is shorter than that in Scenario 1 by 39 ms. In the experiment, User B's old device starts resource handoff before sending information for resource handoff to User A's old device (see Fig. 10). User A's old device can begin resource handoff only after receiving that information. As a result, the length of the handoff period depends on when User A's new device is set up. In Scenario 2, User A continues to use the PC and there is no need to exchange messages between devices. Consequently, the resource handoff in Scenario 2 can finish within a shorter time, to totally eliminate the handoff time.

Figure 11 shows data reception rates during service migration for each user's application in Scenario 1. User A sends a service list at 11.3 s, and service migration is finished at 11.8 s. Scenario 2 has trends similar to Fig. 11 and achieves a shorter service migration as described above.

In the figure, it takes about 1 second from starting the voice phone to terminating the TV phone since we configured the guard time before quitting the TV phone in the experiment. The measurement shows it takes about 500 ms for service migration. However, from the perspective of communication, Fig. 11 shows that data packets continue to be exchanged between User A and User B. Therefore, seamless resource handoff can be achieved during service migration. As about 200 ms is necessary for activating the voice phone application, the migration time can be further reduced by shortening the start-up time.

5. Related Works

Our work is related to several research studies and products. In this section, we discuss these studies and products in terms of connectivity and resource handoff.

5.1 Connectivity

Mobile IP[11] and Mobile IPv6[12] enable a user to locate a correspondent node and to continue communicating even when their IP addresses are changed. However, they only support device mobility and it is difficult to switch between devices or applications.

Instant messaging services such as ICQ[13] and MSN Messenger[14] can keep a user connected to a network. Though they achieve contactability, they do not attempt to address service migration without breaking the ongoing session.

Universal Inbox[15], TOPS[16], MPA[17], and EAPEC[18] also achieve contactability. Incoming communication packets to a user are stored or redirected to specific devices according to user preferences and the situation. Our system concept is similar to these in that the user is regarded as a communication end-point. However, in these systems, only the callee's preferences are reflected in the service despite the many resources surrounding the caller. Also, resource handoffs are not provided during a service since they assume a relatively stable environment during the period of a service. Our system differs on these points.

In reference[2], the caller's preference can be reflected in the service. However, the caller cannot change his device during service initiation even if he would like to use another one to match the callee's preference.

5.2 Resource Handoffs

Nahrstedt[19] and Kikuta[20] describe resource handoff schemes, but mainly for streaming applications. In their system, resource handoffs occur only on the user side because the other side is a contents server. On the other hand, our system aims to provide a real-time communication service, enabling all the users involved to choose suitable resources and to change them seamlessly,

AMID[21] deals with resource handoffs, focusing especially on the network interface and device heterogeneity. IPMoA[22] and Application-layer mobility[23] also provides a way of changing devices in the middle of a session. Although their end-to-end approach is similar to our system, our system achieves simultaneous resource handoff reflecting the requests of both users through service negotiation. We believe that simultaneous resource handoff should be supported in the ubiquitous networking environment.

Wang[24] addresses the problem occurring in SIP simultaneous handoff by exchanging member lists. However, since it assumes resource uniformity as in typical SIP applications, it does not deal with selection of suitable resources.

Roam system[25] presents a seamless application framework for heterogeneous devices on a Java platform[5]. It achieves migration of Java applications even between a PC and a PDA by using dynamic instantiation, offloading computation, and transformation functions. However, since it requires considerable time to migrate, it is difficult to apply to real-time applications.

Our system has the following features compared to the works discussed above. Service migration is incorporated into service initiation, which enables a user to change resources flexibly at anytime once a connection is setup. An end-to-end service negotiation is also introduced that allows all users to choose suitable resources in service migration.

6. Conclusion

This paper showed a service initiation and service migration system which aims to provide a real-time communication service in the ubiquitous networking environment. The key concepts of this system are as follows:

(1) We isolate a few functions for service initiation and migration from applications as the common component, and introduce SIP, a light-weight, wide-spread protocol so that our system can be developed on various platforms.

(2) We incorporate service migration into service initiation so that the service migration function becomes reusable, and also so that the user can change the re-

source at any time once a connection is setup.

(3) Service negotiation enables all user requests and situations to be reflected in service configuration.

Experimental results show that it takes approximately 500 ms for service migration. This is partly because processing the INFO method during service negotiation and initiating an application in new devices is not very fast. However, our system does provide users with seamless communication in terms of real-time communication service.

The common component must be small in terms of size to allow it to be mounted even in small devices. In this implementation, the size of the common component for the PDA is about 227 kB including DLL files for the SIP stack. Just for reference, the KDDI Corporation's upper size limit for applications for cellular phone in commercial service is about 300 kB–600 kB. Compared to this, the size of our implementation is relatively small but it is essential for future projects to tackle.

Studies which will be necessary in the future also include device discovery technology which is outside the scope of this paper. In the ubiquitous environment where there are plenty of networked resources, this technology is increasingly important from the perspective of energy efficiency and security.

Acknowledgments The authors would like to thank Mr. Tohru Asami, President, Chief Executive Officer of KDDI R&D Laboratories Inc., and Dr. Tohru Hasegawa, Executive Director, for their continuous encouragement to this research.

References

1) Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and Schooler, E.: SIP: Session Initiation Protocol, IETF RFC 3261 (2002).
2) Rosenberg, J., Schulzrinne, H. and Kyzivat, P.: Caller Preferences for the Session Initiation Protocol (SIP), IETF Internet-Draft, draft-ietf-sip-callerprefs-10 (2003).
3) Rosenberg, J., Schulzrinne, H. and Kyzivat, P.: Indicating User Agent Capabilities in the Session Initiation Protocol (SIP), IETF Internet-Draft, draft-ietf-sip-callee-caps-03 (2003).
4) Sparks, R.: The Session Initiation Protocol (SIP) Refer Method, IETF RFC 3515 (2003).
5) Java Technology. http://java.sun.com/
6) Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and Berners-Lee, T.: Hypertext Transfer Protocol — HTTP/1.1, IETF RFC 2616 (1999).
7) Donovan, S.: The SIP INFO Method, IETF RFC 2976 (2999).
8) The Official Bluetooth Website. http://www.bluetooth.com/
9) ZigBee Alliance. http://www.zigbee.org/
10) Handley, M. and Jacobson, V.: SDP: Session Description Protocol, IETF RFC 2327 (1998).
11) Perkins, C.: IP Mobility Support for IPv4, IETF RFC 3344 (2002).
12) Johnson, D., Perkins, C. and Arkko, J.: Mobility Support in IPv6, IETF Internet-Draft, draft-ietf-mobileip-ipv6-24.txt (2003).
13) ICQ instant messenger. http://web.icq.com/
14) MSN Messenger. http://messenger.msn.com/
15) Raman, B., Katz, R. and Joseph, A.: Universal Inbox: Providing Extensible Personal Mobility and Service Mobility in an Integrated Communication Network, *Proc. 2nd WMSCA* (2000).
16) Anerousis, N., Gopalakrishnan, R., Kalmanek, C., Kaplan, A., Marshall, W., Mishra, P., Onufryk, P., Ramakrishnan, K. and Sreenan, C.: TOPS: An Architecture for Telephony over Packet Networks, *IEEE J. Selected Areas in Communications* (*JSAC*), Vol.17, No.1, pp.91–108 (1999).
17) Maniatis, P., Roussopoulos, M., Swierk, E., Lai, K., Appenzeller, G., Zhao, X. and Baker, M.: The Mobile People Architecture, *ACM Mobile Computing and Communications Review* (*MC2R*), Vol.3, No.3, pp.36–42 (1999).
18) Kamioka, E. and Yamada, S.: Environment-adaptive Personal Communications Realizing Ubiquitous Computing Networks, *IEICE Trans. Comm.* (*Japanese Edition*), Vol.J85-B, No.5, pp.755–767 (2002).
19) Nahrstedt, K., Xu, D., Wichadakul, D. and Li, B.: QoS-aware Middleware for Ubiquitous and Heterogeneous Environments, *IEEE Communications Magazine*, Vol.39, No.11, pp.140–148 (2001).
20) Kikuta, Y., Kasai, H., Kawasaki, N. and Yamazaki, K.: Design of Seamless Service Environment for Adaptive Service Transfer among Terminals, *Proc. 8th MoMuC* (2003).
21) Ohta, K., Yoshikawa, T., Nakagawa, T., Isoda, Y. and Kurakake, S.: Adaptive Terminal Middleware for Session Mobility, *Proc. 23rd ICDCSW* (2003).

22) Thai, B., Wan, R., Seneviratne, A. and Rakotoarivelo, T.: Integrated Personal Mobility Architecture: A Complete Personal Mobility Solution, *Mobile Networks and Applications* (*MONET*), Vol.8, No.1, pp.27–36 (2003).
23) Schulzrinne, H. and Wedlund, E.: Application-layer Mobility using SIP, *ACM Mobile Computing and Communications Review* (*MC2R*), Vol.4, No.3, pp.47–57 (2000).
24) Wang, H. and Katz, R.: Mobility Support in Unified Communication Networks, *Proc. 4th WoWMoM* (2001).
25) Chu, H., Song, H., Wong, C., Kurakake, S. and Katagiri, M.: Roam, A Seamless Application Framework, *Journal of Systems and Software*, Vol.69, No.3, pp.209–226 (2004).

(Received April 5, 2004)
(Accepted October 4, 2004)

Naoki Imai was born in 1974. He received his M.E. and Ph.D. degrees from The University of Tokyo in 2003 and 2003 respectively. He had worked in KDDI R&D Laboratories Inc. since 2003 and has been engaging in the research areas of ubiquitous networking and ITS (Intelligent Transport Systems). He is a member of IPSJ, IEICE, and IEEE-CS.

Manabu Isomura was born in 1972. He received his M.E. degree from Nagoya Institute of Technology in 1997. He had worked in KDDI R&D Laboratories Inc. since 2000. His current research interests are IP mobility, ITS (Intelligent Transport Systems) and ubiquitous networking. He is a member of IEICE.

Hiroki Horiuchi was born in 1960. He received the B.E., M.E. and Ph.D. degrees from Nagoya University in 1983, 1985 and 2000 respectively. He is a senior manager of Ubiquitous Networking Laboratory, KDDI R&D Laboratories Inc. Since joining the Labs. in 1985, he has been engaged in research on computer communication, distributed processing, network management, intelligent transport systems and ubiquitous networking. He received the Excellence Award in 54th and 58th Annual Convention from IPSJ in 1996 and 2000, and the Young Engineers Award from IEICE in 1994. He is a member of IPSJ and IEICE.

Sadao Obana was born in 1953. He was received BS, MS and Ph.D. degrees form Keio University in 1976, 1978 and 1993 respectively. After joining KDDI (former KDD) in 1978, he engaged in R&D in the field of packet exchange systems, network architecture, OSI (Open Systems Interconnection) protocols, database, distributed processing, network management and ITS (Intelligent Transport Systems). In 2004, he joined Advanced Telecommunications Research Institute International (ATR) and is now a director of Adaptive Communication Research Labs. in ATR. His current research areas are wireless mobile ah-hoc network, ubiquitous wireless sensor network and ITS. He received an Award of Minister of Education, Culture, Sports, Science and Technology in 2001. He was a member of the board of IPSJ in 1999–2001 and is a IPSJ Fellow.