A Distributed Service Manager for Seamless Service Continuity

Vincent Verdot[*] [**], Noël Crespi[**] and Yann Gasté[*]
[*] Alcatel-Lucent Research and Innovation, Nozay, France
[**] Institut National des Télécommunications (GET-INT), Wireless Networks and Multimedia Services, Evry, France
{vincent.verdot, yann.gaste}@alcatel-lucent.fr; noel.crespi@int-evry.fr

***Abstract*—The IT domain has considerably evolved, users are now surrounded with cheaper and more powerful terminals than ever. Mobile and fixed devices connected to high-speed network technologies can bring out for the owners almost any services. But if the technology advances allow the user to enjoy his services more freely and conveniently, are the mobility mechanisms ready to offer a real seamless experience? In this article, we answer this question by analyzing the service management issues and the strength and defaults of existing solutions. From these analysis, we propose a new distributed system that assures the continuity of services by controlling the applications and their resources over the user's devices. This solution brings out new user experiences by considering his devices as a single and coherent system providing services regardless of the device actually hosting them.**

Keywords: service continuity; session mobility; terminal handover; resource management.

I. INTRODUCTION

When a user is consuming a service such as writing a document, communicating with a friend or watching a movie, he would like the service to be available regardless of the mobility constraints and profit from the environment characteristics.

The challenge is to keep a service alive when the user is moving, changing access technology or switching device while adapting to its environment: different terminal capabilities, peripherals or network properties.

A. Information Technology Sphere has Changed

Small computer devices (such as PDA, smart phones or cellulars) are now highly-capable in terms of computing and memory power. On the other hand, wireless data access networks are also more efficient, providing greater bandwidth and a wider coverage.

Thus, mobile devices capabilities tend to be equivalent to fixed ones, able to offer almost the same services, possibly at a lower quality. Actually, the user now owns a set of fixed and mobile devices, providing equivalent features with different properties (access technology, display characteristics, specific peripherals, etc), so he uses a device according to the properties he wants to be applied to his service.

Moreover, nowadays the user terminal devices, ranging from mobile phones to desktop computers, are cheaper and now more powerful at a same price as their previous models. Thus the customers tend to own multiple devices, each with different characteristics like mobile, wide-screen or high-capacity storage.

B. A New Mobility Challenge: Service Continuity

This new context in the Information Technology sphere offers more freedom to the user who is no more limited to a set of services an a certain device. But this user-centric schema would reveal its full potential if the mobility aspects would be efficiently handled. For the best user experience, the mobility mechanisms must be transparent, and so guarantee seamless service continuity.

The concept of "service continuity" is an improvement of "service mobility". The service mobility is assured when a user can access his services from any device, while in service continuity he can also access to the service context (*i.e.* the set of information used by the application providing the service). In other words, in the former the user can use the same service from any device while in the latter he continues using his service, without interruption, from any terminal (*i.e.* he does not need to call back his correspondent, reopen the file he was editing… or realize any service initialization action).

Many works are focused on the mobility management at the lowest layers (*e.g.* horizontal and vertical handover), and the existing solutions are interesting but can not be applied to higher layers concerns. Because it is always necessary to keep the service context while moving in order to achieve the service continuity and only high-layer mechanisms are able to guarantee such mobility, especially when a service is transferred from a device to another (terminal handover).

The goal of this paper is to propose a novel and efficient model of service mobility management within a user's device set. The remainder of this article is organized as follows: Section II defines important keywords, presents service continuity issues and finally exposes the ongoing research works and the requirements for an efficient solution. In Section III we introduce our proposed model, describe its principle and the main mechanisms, then we present a use case and discuss the advantages and drawbacks of our solution and the potential improvements. Finally, in Section IV we summarize the lesson learned and conclude by referring to our work perspectives.

II. STATE OF THE ART

A. Services and Applications

In this paper, we often use the terms of "service" (a functionality) and "application" (its implementation). This is how they must be understood in this article.

A service is a functionality provided to a user. It can be for communication purpose (Voice over IP, Instant Messaging), edition (text editor, document reader), multimedia (video streaming, music)… any function provided by a device. We could classify the services into two categories: the connected services that imply several peers (client and/or servers) and require a network connection; and the local services that only require the local device. These two service categories impose different constraints, the former is usually more time-sensitive, requires synchronization and has a light context while the latter generally uses important resources in terms of computing and memory capacity.

An application is a program that realizes a service on a device. It can be considered as one instance of the service because different applications can provide the same service, as long as they provide the same basic functionality (*e.g.* sending and receiving messages with an IM service). As an application is a service instance, it consumes resources on the device executing it; these resources can be files, volatile memory information, input/output streams… and represent the raw material of the service.

B. Overview of Service Continuity Issues

Mobility management is a wide research area, which impacts every network layers. We are focused on the highest one because it is where we can most precisely control the service behavior and appreciate the user experience.

Nevertheless, the service mobility can not be only managed by high layer mechanisms, thus it must also be supported by lower layers better adapted for some mobility issues such as for horizontal or vertical handovers (change of access network or technology), many research efforts deal with these concerns, cf. [1][2][3]. In this article, we suppose that mobility mechanisms regarding the lower layers are correctly handled and transparent for the application. Then we have chosen to stay focused on the service continuity concept which is the main issue during a terminal handover.

The principle of service continuity is to provide a service without interruption in a mobile environment. One of the main challenges is being able to support a handover between two different terminals. Existing solutions are mainly based on synchronization between two terminals using the same application. With a standardized protocol that owns its specific transfer messages, it is rather easy to carry out a handover between two applications. But what about services not based on standardized protocols, which do not have their own integrated transfer mechanisms? A text-editor service for example can be provided by many applications, so transferring such a service from a device to another means transferring its context from an application to another by probably adapting it (c.f. Fig. 1).

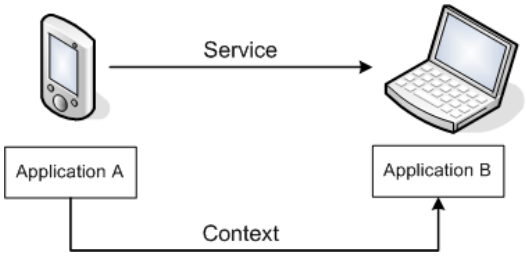

Figure 1. Service transfer mechanism between two devices.

When a transfer occurs between two terminals, the delay between the service switch request and the availability of this service on the new device is critical as it must be short enough to be acceptable for the user. This delay will be called "service transfer delay" in the remaining of the article.

The notion of "acceptable" is subjective as it depends on the user himself and the type of service. A real-time application for example, typically the communication services, requires the shortest delays to be satisfactory while a user would accept a couple of seconds for a text editor transfer. The notion of "seamless" service transfer is also subjective and similar to the definition of "acceptable", therefore we will consider a service transfer to be seamless if the delay is acceptable and no action is required from the user.

C. Toward a Service Continuity Solution

Research works on service continuity are rather scarce but some studies and existing solutions are really interesting as they bring useful concepts and mechanisms.

1) Display forwarding

A really basic solution is the principle of display forwarding (*e.g.* X-forwarding, Virtual Network Computing). The application is hosted by a device that realizes all the computing and plays the role of server, then any authorized terminal can remotely access to the application by connecting to this server. The server sends application outputs to the client terminal which sends the user's inputs to the hosting device. Thus, the user can switch of device; he will still be able to use the same application without interruption as long as he can stay connected to the server.

This service continuity mechanism has interesting properties:

- The handover delay between two devices is very short as only the display need to be transferred from the server.
- The handover is synchronized as the application does not stop.
- The mechanisms are independent of the service type and the device capabilities.

Nevertheless, this solution is not satisfactory for the following reasons. It requires the user to be always connected to the hosting device, so the service is no more operational if a network failure occurs or if the server is shutdown. Moreover, as the application is not transferred on the new device, it can not benefit of its new capabilities. Finally, there is no management of the user devices, so he needs to know where each application is hosted before to be able to switch the right service.

2) Session mobility

Another solution which consists of a subset of the service continuity issue is the session mobility concept. Session mobility has been widely studied in multimedia and communication services and several mechanisms are standardized.

The Session Initiation Protocol [4] standardized by the Internet Engineering Task Force is a reference signaling protocol that creates and manages multimedia sessions. This protocol has been adopted by the 3GPP[1] in the UMTS[2] with the

[1] 3rd Generation Partnership Project.

[2] Universal Mobile Telecommunications System.

IP Multimedia Subsystem [5], next generation core network and service architecture for telecommunication operators.

Several SIP extensions have been added to improve its functionalities. The new SIP method "Refer" [6] and the header "Replaces" [7] were introduced to manage the mobility of multimedia communications between terminals. Identified as a "session", a communication initiated with SIP is switched from a device to another on user request. As this service continuity solution is integrated into the signaling protocol, the corresponding standardized mechanisms are obviously more efficient; synchronization, interoperability and authentication issues are naturally handled. Certain research works have proposed advanced session mobility solutions based on SIP cf. [8][9]. Other protocols such as Real Time Streaming Protocol [10] or RTP [11] can also be used to assure a session mobility.

Nevertheless, the session mobility mechanisms can not be applied to any services because it is necessary for the signaling protocol to identify the "sessions" corresponding to the service. However, some services can not be identified as "sessions" or are independent of the signaling protocol providing the mobility mechanism, such as a text editor application. Moreover, as in the previous solution, the user needs to find which terminal is hosting the communication session he wants to switch. Finally, with this solution the user does not control the services, he just can move the sessions but the applications need to be ready to accept an incoming session to achieve a switch.

D. Requirements for a Service Continuity Management

Currently, there is no solution for service continuity management over a user's terminal set. Such a model should control the user's devices to provide seamless and transparent service management architecture. According to the analysis of the mechanisms and concepts studied in the previous section, we can emphasize the properties required for such a model.

- Transparent: the service management should be transparent to other applications, so the mechanisms need to consume low computing and memory resources.
- Efficient: the transfer delays should be optimized to appear seamless regarding to the service type.
- Service independent: any service type should be supported.
- Terminal independent: the services should be adapted to the device's capabilities.
- Self-configured: the devices and the services should be automatically managed (identification, location).
- Secured: user's services should be protected with authentication mechanisms, moreover the system should support transfer or device failures.

III. A NEW MOBILITY MANAGEMENT MODEL

As the existing mechanisms do not bring a satisfying solution, we try to define a new mobility management model that guarantees the service continuity with the previously defined constraints.

A. Principle

The main principle of our service management model is to control the applications and their resources provided by the user's devices to eventually transfer the corresponding services from a device to another.

This new model assures the continuity of the user's services over a set of his devices. This set is dynamic as it can consist in mobile terminals, devices turned-on/off… however the mobility mechanisms must remain operational. Therefore, the most appropriate architecture for this purpose is a distributed system as it supports any device addition or suppression while preserving its functionalities.

Thus, our solution consists in a distributed application integrated in each device operating system to be managed. This application, the Distributed Service Manager (DSM), establishes a secured network overlay with all user's devices to be managed. This network overlay is called Personal Service Environment (or PSE) and it is the base of our model.

A user's Personal Service Environment is a concept that represents the group of devices controlled by a Distributed Service Manager (c.f. Fig. 2). All devices belonging to a PSE are necessarily networked, DSM-enabled (with integrated DSM functions), and authorized by the user. A PSE consists of one or more devices, but a device always belongs to only one user's PSE.

Therefore, a PSE is a network overlay managed by the DSM which controls the services hosted by the corresponding devices. Thus, the DSM provides the user the necessary mechanisms to control his services from any device. He can transparently transfer the service from a terminal to another, regardless of the applications or the device delivering the service.

B. Detailled View

The Distributed Service Manager is integrated into the operating system of each device to be managed. The DSM needs to be tightly linked to the operating system functions as it

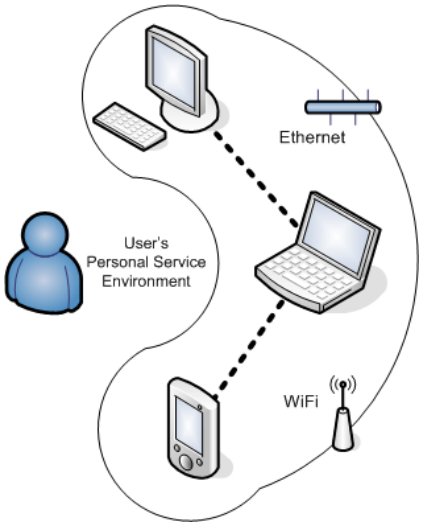

Figure 2. A Personal Service Environment composed of three DSM-enabled devices.

must control the applications and all the corresponding resources (files, network socket, data streams, etc). Then each device cooperates to manage the Personal Service Environment by performing the following tasks:

- manage the PSE nodes,
- control the applications,
- manage the user's services.

1) The nodes

The first role of a DSM is to manage the PSE by merging other PSE (from the same user) composed of one or more devices. When new devices are encountered, they are identified and authenticated to be merged to a single PSE (each DSM-enabled device is previously configured by the user). Fig. 3 illustrates an example of PSE management. In the schema a) three distinct PSE are present, the fourth device is not DSM-enabled so it can not belong to a PSE. In the schema b) the laptop has activated his Ethernet and WiFi network interfaces, thus it can connect to the two other PSE and then the DSM merges them into a single one.

PSE terminals can be located in different networks, especially when a device owns several network interfaces (*e.g.* Ethernet and WiFi), in such a case, the DSM must route received requests from an interface to another if necessary. An unresponsive device (turned-off, disconnected from the network, etc) is automatically detected by the DSM and removed from the PSE. Each PSE device has a list of other peers, updated on any change. The routing mechanisms are not detailed in this paper, but could consist in any peer-to-peer routing algorithm.

2) The applications

The main role of the DSM is to manage the user's services, for this purpose it needs to control the corresponding applications and their resources.

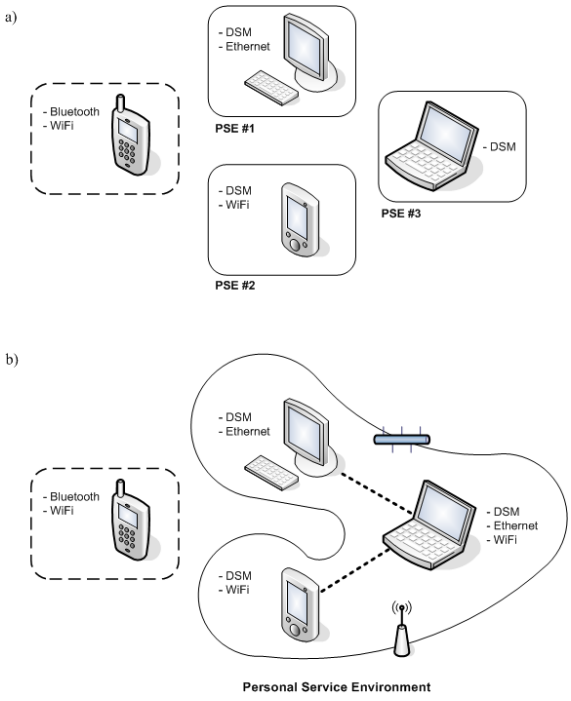

Figure 3. Management of Personal Service Environment devices.

The applications can be started, stopped, paused or resumed by the DSM on any device, this is achieved by the DSM functions integrated in each device operating-system. Start and stop actions are the basic launch and kill but the pause and the resume ones are specific functions that must be supported by the application. If an application does not support the resume and pause functions, the provided services will be ignored by the DSM and all the corresponding mobility mechanisms, these applications are non-DSM compliant.

The pause action must be handled by the application on DSM request, it consists in making the application to stop its process and to provide a snapshot of its resources.

When a service is provided to a user, an application (or several) is processing data to offer the required functionality. For example with a text-editor service, a text editor application such as *vi* processes many resources: the current edited files, the cursor position, the command history, etc. Each resource is constantly evolving during the application process; a snapshot is a frozen value of these resources, at the last stable state. The notion of "stable state" means that the resource values are coherent, *e.g.* in our case, the "command history" resource can not have the last command if it is still not applied to other resources. Finally, the application associates each resource listed in the snapshot with an identifier, preferably human-readable which should be standardized for well-known service types to assure an efficient interoperability between applications. The resource snapshot can only be provided by the application as a stable state is dependent on the service semantic. Thus, on pause request, the application stops its process and sends back a snapshot to the DSM.

The resume function must be handled by the application on DSM request with a snapshot as argument. It consists in starting the application with a predefined state of its resources, those provided in the snapshot. As the same service may be processed by different applications, a resuming application is free to discard unhandled resources listed in the snapshot. This way, the service can be adapted to different applications.

3) The services

All DSM-compliant services launched within a PSE are logged, and uniquely identified. Thus, the user is able to get, from any device belonging to the PSE, the list of services currently running and their properties (identifier, type, description, requirements, etc). He can also issue orders to manage these services over the PSE: start, stop or transfer. If the first two actions are basically realized by launching or killing the corresponding application, the third one is the most interesting as it allows realizing the service continuity. When a transfer order is issued by the user, the DSM requests the corresponding application to pause. Then a snapshot is generated and provided to the appropriate application of the new device, in resume mode. The application to be launched in the destination terminal is defined by the service type (previously logged).

4) Resources management

As described previously, during a service transfer the application resources are transferred from a device to another within a snapshot.

These resources can be really simple such as a font size, a cursor position (*e.g.* with a text-editor) but they could also be

more important in terms of memory usage: a text document, a video clip... or even un-transferable like a socket or a stream.

The size of all application resources can be huge and the transfer time of the corresponding snapshot will grow significantly. Thus the service transfer delay will certainly become intolerable for the user, non-appropriate to the service type and then the seamless mobility management can not be achieved.

We introduced in our model a specific mechanism to manage the application resources. Of course, some resources are mandatory for a given service; *e.g.* in text-edition the edited file is mandatory, so the user must wait as long as necessary to continue editing his document. But what about the last color used, the font face or the clipboard content? They may not be necessary, in particular if the new application does not support those options. So, it seems logical that it is up to the application (the destination one) to decide which resource is needed and must be transferred. Thus the mandatory resources take precedence over the optional and unsupported ones, and are transferred faster to minimize the service transfer delay.

As described before, the snapshot consists of a list of resources, each associated to an identifier. The resource management mechanism is based on this identifier, also called "resource anchor" which uniquely identify the resource in the device. When the application generates the snapshot, the DSM stores the resource values in the local device and keeps only the anchors in the snapshot. Then, when the new application receives the snapshot, it requests the needed resources to the DSM thanks to their identifier. Fig. 4 presents an example of resource management mechanism during a service transfer. In the schema a) the application A is paused, then it provides a list of its resources in the latest stable state associated to anchors and stops. Note that the application shutdown can be delayed to continue providing a service during the transfer delay (especially for time sensitive services). The Distributed Service Manager resumes the new application B with a simplified

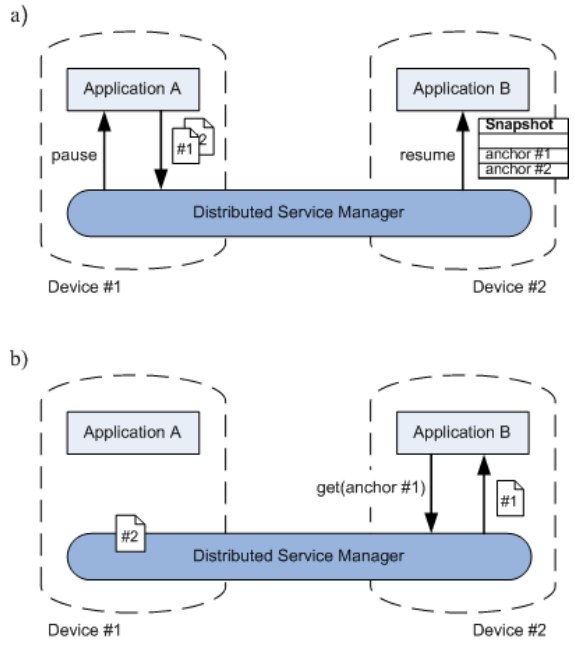

Figure 4. Resource management mechanism.

snapshot that only carries the list of anchors and stores the corresponding resources locally. Then, in the schema b) the application B requests from the DSM the resources it needs according to its functionalities and priorities. Finally, the data corresponding to the resource anchor is provided to the application.

The resource anchors are an efficient way to minimize the transfer delays and the snapshot size but it is also a solution to provide un-transferable resources for the destination device. Those resources are typically data stream or network sockets (input or output data stream). They share the property to continuously send or receive data; transferring these resources from a device to another is obviously impossible because depending on the operating system, but not the corresponding anchors.

For example, with a communication service, a device is continuously receiving media packets on a network socket. To transfer the service and the corresponding resources, the VoIP application adds to the snapshot the socket identifier (which is dependent on the operating system) and associates it to the anchor "incoming voice". When the new application will be resumed with the snapshot, it will request to the Distributed Service Manager the resource associated to this anchor, then the DSM will provide the data by connecting to the first device that will realize a network tunneling between the correspondent node and the new terminal. Of course, this mechanism imposes a strong constraint as the intermediate nodes needs to remain connected to assure the tunneling, nevertheless it is applicable to any service and resource.

An interesting feature made available through the resource anchors is the resource transitivity. When an application receives a snapshot with unsupported anchors, it simply ignores them, and the user will only use his service with fewer options. Then, if he switches again his service to a third device, he will send all the resource anchors even those he previously received, and if the new application supports them the service will be continued with more options. Thus, the service can be adapted to the device capabilities in lower and higher quality.

C. Example

Bob owns two devices (all DSM-enabled): a WiFi PDA and a personal computer (Ethernet connected to a WiFi access point) at home. He is in the bus and wants to write a letter to print it once arrived at home. So he launches a text-editor on his PDA, actually he starts a text-editor service on a PSE consisting in only one device: the PDA (his PC is turned-on but not connected to the PDA, so it belongs to another Bob's PSE). He edits his document and finally arrives at home. Bob's PDA connects to his WiFi access points and then both DSM (PDA and PC ones), detect each other, and merge into one DSM managing one PSE compound of two devices.

Bob enters his house and chooses to finish writing the letter on his PC that offers a wider screen and keyboard. So he switches the text-editor service to the PC by selecting it in a specific GUI listing the services currently running in the PSE. His preferred PC text-editor application (*e.g.* Open Office) assigned to the text-edition service type is automatically launched (*resume*) and the PDA application is stopped (*pause*). The service context, *i.e.* the application resources, is transferred from the PDA to the PC, and Bob can finish writing his letter, with all supported parameters configured as in the previous

application: file opened, cursor position, command history, etc. So he cancels the last operation (a misspelled word), inserts an image, changes the fonts and prints the document.

Thus, Bob launched his service only once in a PSE, then the devices become interfaces to the service (through an application), providing a different experience according to the terminals properties. His service was adapted to fit the new device capabilities (advanced software and hardware) because Bob chooses to benefit from the convenient characteristics of his PC but if he needs to use the mobility aspect of his PDA, he can switch his service to the PDA and the service will be adapted with fewer options but with the Bob's required mobile attribute.

The same example could be presented with any services: multimedia, communication, gaming, as long as the service context can be defined, *i.e.* the corresponding snapshot and its properties.

D. Discussion and improvements

The main strength of our model is to consider the user's set of devices as a single system providing his services, the Personal Service Environment, regardless of the device actually hosting it.

Our model successfully fits to the constraints listed in Section II.D. The Distributed Service Manager brings a low computing mechanism by distributing the functions over the PSE devices, it also assures a secure architecture that does not rely on a single device, finally it manages the PSE, the routing and the services automatically. The resource management mechanism is also a real strength of our solution, it makes possible the realization of optimized transfers that support any service types (any resources), and any devices (service adaptability). However, several points of our model could be more deeply studied to be certainly improved.

The standardization aspect of the snapshots and their resource anchors (c.f. Section 4) is really important as it would enable the interoperability between all applications providing a same service. Moreover, several properties could be associated to the snapshots to make the service adaptation mechanisms even more efficient: mandatory resources (for service delivery), resource groups (resources that can not be used if not all supported), service types, resource type and size, etc.

The resource management mechanism offers a way for remote devices to access to un-transferable data; nevertheless it is necessary to remain connected to the intermediate devices which assure a tunneling of the incoming data. This issue should be studied in future research works.

Other important mechanisms out of the scope of this article should be detailed such as the overlay network routing, the authentication features or the integration of the DSM functions into operating systems and applications.

IV. CONCLUSION

In this article we focused on the service management issues and more precisely the service continuity aspects for which existing solutions are not suitable if we consider heterogeneous devices and services.

We proposed a new model offering the user a seamless and convenient service continuity management to improve his experience by enabling a more natural usage of his services. We presented a distributed system that manages the user's services over all his devices, with continuity mechanisms adapted to the terminal capabilities. We also introduced an optimized resource management mechanism that makes the service transfers more efficient and suitable for any type of service.

In the next step of our research work, we will try to implement the Distributed Service Manager to evaluate its performances in managing the Personal Service Environment and the application resources, but also to measure the transfer delays, and the service adaptation capabilities of the system. This implementation will prove the feasibility of our model and the impact on the service and the user's experience.

We will also focus more precisely on the communication services as they imply challenging constraints: real-time data processing, multiple input and output streams, network dependant... A DSM prototype would allow precise measurement of the performances of such time-sensitive services and the potential network issues (distributed routing, bandwidth bottleneck, etc).

Finally, we will study how to represent such a distributed environment composed of multiple devices and services, how to conceive an intuitive and convenient interface between the user and his Personal Service Environments.

REFERENCES

[1] H. H. Choi and D. H. Cho, "Takeover: a new vertical handover concept for next-generation heterogeneous networks," *Vehicular Technology Conference, IEEE*, 2005.

[2] J. McNair, Z. Fang, "Vertical handoffs in fourth-generation multinetwork environments," *Wireless Communications, IEEE*, 2004.

[3] M. Li, Y. Fei, V.C.M. Leung, T. Randhawa, "A new method to support UMTS/WLAN vertical handover using SCTP," *Wireless Communications, IEEE*, 2004.

[4] J. Rosenberg et al, "SIP: Session Initiation Protocol," RFC 3261, *Internet Engineering Task Force*, June 2002.

[5] 3rd Generation Partnership Project, "IP Multimedia Subsystem (IMS)", TS 23.228.

[6] R. Sparks, "The Session Initiation Protocol (SIP) Refer Method," RFC 3515, *Internet Engineering Task Force*, April 2003.

[7] R. Mahy, B. Biggs and R. Dean, "The Session Initiation Protocol (SIP) "Replaces" Header," RFC 3891, *Internet Engineering Task Force*, September 2004.

[8] H. Schulzrinne, E. Wedlund, "Application-layer mobility using SIP," *Service Portability and Virtual Customer Environments, IEEE*, 2000.

[9] R. Shacham, H. Schulzrinne, S. Thakolsri, W. Kellerer, "The virtual device: expanding wireless communication services through service discovery and session mobility," *Wireless And Mobile Computing, Networking And Communications, IEEE*, 2005.

[10] H. Schulzrinne, A. Rao and R. Lanphier, "Real Time Streaming Protocol (RTSP)," RFC 2326, *Internet Engineering Task Force*, April 1998.

[11] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 3550, *Internet Engineering Task Force*, July 2003.