Demo - dSM: distributed Service Manager for Seamless Mobility

Vincent Verdot
Alcatel-Lucent Bell Labs France &
Institut TELECOM SudParis
Email: vincent.verdot@alcatel-lucent.com

Noël Crespi
Institut TELECOM SudParis
Multimedia Mobile Networks and Services dept.
Email: noel.crespi@it-sudparis.eu

***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?**

The answer is obviously "no", so in 2007 we introduced the *distributed Service Manager*, a system which purpose is to assure the mobility and the continuity of user's services. The main challenges was to conceive a model totally independent from the device, the application, or the type of service. We defined a new architecture addressing these issues and proposed an innovative vision of the "service" concept. Finally we implemented a prototype, now operational, for demonstrating the high potential of our approach.

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 fully profit from the environment characteristics. The challenge is to maintain the service when the user is moving, changing access technology or switching device, all the while adapting it to its new environment: different terminal capabilities, peripherals or network properties.

If the mobility issue is yet a real challenge by itself, when applied to services a new dimension appears: *continuity*. Indeed, service delivery MUST be seamlessly transferred to guarantee a satisfactory experience to the user. But continuity is a broad concept, hard to define and so hard to measure. To simplify, we identified two types of continuity: temporal and contextual, corresponding to quantitative and qualitative metrics.

Most of current works focus on the former continuity aspect as they usually correspond to low-layer mechanisms which are just unaware of service's context. Therefore, with the distributed Service Manager we intend to fully address the *service continuity* by providing a system both efficient and flexible (*i.e.* context-aware).

The remaining of this demonstration proposal will be organized as follows. In section II we will introduce notable research approaches on service continuity topic and discuss about their strengths and weaknesses. In section III we will briefly present our system, the distributed Service Manager and the prototype we implemented. Then in section IV we will describe the corresponding demonstration and its requirements. Finally we will conclude with demo expectations and future works.

II. Existing Solutions

Service mobility management may be handled at any level, however most research works on this topic focus on network and application layers. We can also note that service mobility situations may occur in various scenarios: network configuration update, switch between access points or technologies, device handover, application switch, etc. We chose to specifically address the terminal handover use case (*i.e.* a service is transferred between two different devices) as it covers most of mobility-related issues, excepted maybe the inter-technology transfers, also known as "vertical handovers".

We do not address this latter use case as it is a pure network-level issue, widely studied and typically handled with interworking mechanisms for specific pairs of technologies (*e.g.* [1] or [2]). Moreover, terminal handover is really interesting as it brings strong context-based constraints, requiring a full and smart adaptation of the service to its real new environment: network, hardware and software capabilities. The need to seriously consider the service's context makes this approach very interesting as service mobility efficiency is too often only evaluated on a lowest-delay basis, regardless to the end-user experience.

Many service mobility solutions exist: as proprietary mechanisms or standardized protocol extensions (SIP, MobileIP, SCTP, RTP, RTSP, etc), we will not detail them here. Based on a specific protocol (applications) for high-level solutions, or based on data (streams) for lower-level ones, they all share a common serious lack: a poor service context management. Indeed, if the transfer delay is efficiently minimized, offering a great temporal continuity, contextual continuity is dramatically poor. However, transferring a service requires an efficient transfer (quantitative constraint) of the service logic but also a smart adaptation of the service context (qualitative constraint). This is the objective the distributed Service Manager intend to reach.

III. The distributed Service Manager

A. Principle

The main principle of our approach is to control the applications and their resources of a user's devices to eventually transfer the corresponding services to another terminal. This system assures the continuity of the services over a set of devices by transferring the application resources on demand. These resources are then adapted to the new application and thus the service is resumed with its context.

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

We will not detail further the distributed Service Manager as it introduces many new concepts that would confuse the reader. So, to keep it simple we could say that the system relies on three major mechanisms:

- an interface defined between the dSM and the applications to transparently control the services,
- a formal description of the service context to conveniently transfer and adapt it to the new environment,
- an efficient management of resources based on references to seamlessly transfer any kind of data.

For more information about the distributed Service Manager, we invite you to read the original article (cf. [3]).

B. Prototype

A dSM prototype was recently implemented. It is a Sun's Java application, deployed on every user's devices. A peer-to-peer overlay is established with all managed hosts (we used JXTA for P2P network management). Then via a console or a simple GUI (based on Eclipse's Rich Client Platform), it is possible to see other terminals and the hosted services. The interface also allows to select a remote service and on a single mouse click to "get" it transferred to the current device.

Only a minimal set of features were implemented, enough to allow us to realize measures and prove the feasibility of our system. We also had to modify several applications to make them "compliant" and then be able to experience a service transfer between very different environments, considering the device, the operating system and the application. So to sum up, we successfully tested the transfer of a text-edition service between a Microsoft Windows device and a GNU Linux, each one using a different text-editor; the context consisted of two resources: the cursor position and the text body.

The dSM was conceived as a *mobility enabler*, *i.e.* it brings a new capability to the device via a simple API (few lines of code to make an application dSM-compliant). dSM is today an open-source project hosted on *Launchpad* (cf. [4]).

IV. Demonstration

A. Description

A simple demonstration of the distributed Service Manager prototype could proceed as follows.

1) A first device is powered on (*e.g.* laptop with MS Windows), the dSM application and its User Interface (graphical or console) is presented to the audience.
2) A text-editor is then launched on this terminal, we explain how the application communicates with the dSM and how it is automatically registered.
3) We briefly use the text-editor application, typing some text and positioning the cursor at a precise place within the document's body (we could also use undo history or text formatting to show adaptation mechanisms).
4) Then, according to the scenario, typically *"Bob using his laptop in the bus finally arrives at home and wants to continue writing his document on his Personal Computer, an obviously handier solution"*, another device (*e.g.* Linux-based) is powered on (or simply get connected to the same network).
5) Both devices automatically discover each other, the dSM UI displays the newly found peer and lists the services remotely hosted, here the text-edition one.
6) Then we select the service, click (or enter) the "GET" command, and the service is automatically switched (usually less than 1 second). The first application is closed, a new text-editor is started on the destination terminal and the service is resumed with the appropriate text body, cursor position (and possibly undo history or any other if implemented).
7) A little explanation of the involved mechanisms is provided to the audience (possibly some support slides).
8) *An additional service transfer to a third device or back to the initial one could be achieved, it will show that the context is adapted by adding and removing resources.*

B. Requirements

Requirements are minimal as the whole demonstration could take place on a single laptop (using virtualization) which we will provide (so all the system will be already set up).

Nevertheless the demo could also be achieved on several physically distinct terminals, then an additional computer and a wireless (or even basic Ethernet) network would be required. Finally, a projector to display device's screen or presentation support slides would make the demonstration even more clear to the audience.

V. Conclusion

The main objective of this demonstration is to show the high potential of a new service mobility approach. The distributed Service Manager prototype demonstrates that the contextual continuity is essential and not only a "nice to have" feature regarding the user experience. Finally it presents several efficient and innovative mobility mechanisms which could be reused to address other future mobility-related issues.

REFERENCES

[1] J. Mcnair and Fang Zhu. Vertical handoffs in fourth-generation multinetwork environments. *Wireless Communications, IEEE*, 11(3):8–15, 2004.

[2] M. Li, Y. Fei, V.C.M. Leung, and T. Randhawa. A new method to support umts/wlan vertical handover using sctp. *Wireless Communications, IEEE*, 2004.

[3] Vincent Verdot, Noel Crespi, and Yann Gaste. A distributed service manager for seamless service continuity. In *ICNS '07: Proceedings of the Third International Conference on Networking and Services*, page 15, Washington, DC, USA, 2007. IEEE Computer Society.

[4] Vincent Verdot. distributed service manager open-source project. https://launchpad.net/dsm, 2008.