Push Driven Service Composition in Personal Communication Environments

Justinian Oprescu, Franck Rousseau, Laurentiu-Sorin Paun, and Andrzej Duda

LSR-IMAG Laboratory
BP. 72, 38402 Saint Martin d'Héres, France
{Justinian.Oprescu,Franck.Rousseau,Laurentiu-Sorin.Paun,
Andrzej.Duda}@imag.fr
http://www-lsr.imag.fr

Abstract. Our current mode of communication is mainly based on the *pull* model in which the user requests information or initiates a communication stream. We believe that a personal communication environment should behave according to the *push* model in which a source of data or the network infrastructure takes care of preparing communications and proposing them to the user. In this proactive way of operation, the only user intervention is to choose the right communication flow. In this paper we present the service discovery and composition in Omnisphere, a personal communication environment for wireless appliances. All elements in Omnisphere are considered as services that can be dynamically discovered and composed to form complex communication applications. Based on user preferences, device capabilities, and context Omnisphere makes use of existing discovery protocols such as SLP, Jini, UPnP, or DNS-SD to discover relevant services. Service descriptors provide further information on the data types generated on output or accepted on input. All this information allows Omnisphere to configure services and user applications so that the user can benefit from complex communication applications composed on demand.

1 Introduction

We present the service discovery and composition in Omnisphere [1], a personal communication environment for wireless appliances. The goal of Omnisphere is to mediate communications on behalf of the user in a proactive way. Many of our current applications are governed by the *pull* model in which the user requests information or initiates a communication stream (Web, e-mail etc). In this interactive mode of operation, the user is an active part and must initiate the process of information acquisition. In the *push* model, it is the source or the network infrastructure that proposes the information to the user in a similar way to the SIP protocol which invites the user to a communication session. Such a mode of operation is at the base of Omnisphere: it takes care of preparing communications and proposing them to the user. In this proactive way of operation, the only user intervention is to choose the right communication flow.

M. Conti et al. (Eds.): PWC 2003, LNCS 2775, pp. 505–510, 2003.

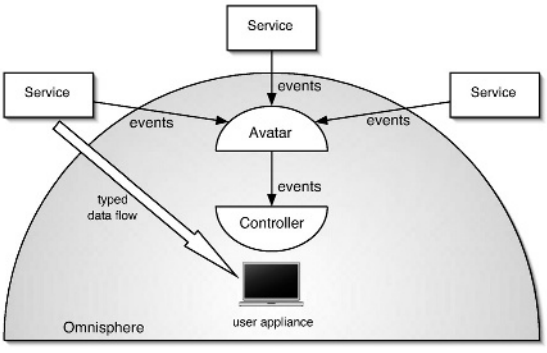

Fig. 1. Omnisphere architecture

For common applications we transform information sources that require user intervention into sources that notify the user about some events and then the user may decide to consume the proposed information or not (e.g. a NotifyMail service checks the user mailbox for new messages and notifies the user about important messages, the user may then read a message using a standard mailer and the POP protocol). The role of Omnisphere is to mediate notification events according to the user preferences, discover required services, configure them, and compose them into complex communication applications.

In the rest of the paper, we focus on the push driven service composition in Omnisphere. We consider that communication chains in a personal communication environment can be dynamically managed when all components follow the same paradigm – "everything-is-a-service": all basic services, sources, transcoders or filters, as well as user applications are viewed as services from the discovery point of view. Their descriptors can be found and used along with other information such as the user preferences to compose complex communication applications. Due to the space limitation we skip over the related work, obviously there are several projects related to our work.

The paper is organized as follows. Section 2 presents the architecture of Omnisphere. Section 3 describes our approach to service discovery and composition. Finally, we present conclusion and future work (Section 4).

2 Architecture of Omnisphere

Omnisphere is a communication and information universe surrounding wireless appliances (more detailed description can be found elsewhere [1]). It mediates various communication flows to relieve the user from any cumbersome details of finding the right service components, configuring, and composing them based on user preferences, device capabilities, and the current context. *Omnisphere* comprises the following elements:

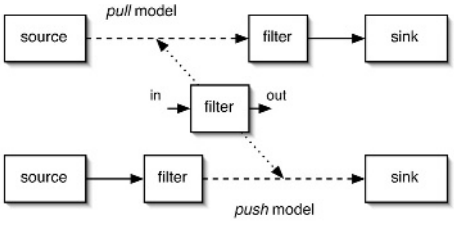

Fig. 2. Composition of basic services

- *Ambient services.* An ambient service is an abstract view of a communication component. It represents a high-level concept that allows us to construct complex services out of primitive ones by connecting them with *typed data flows*. An ambient service encapsulates some processing, provides a signaling interface for generating events and accepting a configuration commands, and publishes a service description.
- *Typed data flows.* A typed data flow is an abstract view of communication between ambient services. It allows to associate the type of data with a given flow.
- *User avatar.* An avatar represents the user in Omnisphere. Its role is to mediate event notifications concerning different data flows. It receives events from ambient services and acts according to the information on user preferences, the current context, and device capabilities to either notify the user or set up communications. It monitors the presence of user devices, manages services available at the given Omnisphere, and discovers services required for the user. An instance of the user avatar exists at each Omnisphere in which the user is present.
- *Omnisphere controller.* It runs on a user appliance and takes care of events sent to the user. When needed, it will configure and run applications used by the user to consume communication flows. When requested by the user, it will initiate the discovery process of available services in the present Omnisphere. Upon the arrival in a new place, it will register the device within a given Omnisphere and authenticate the user.

3 Service Discovery and Composition in Omnisphere

We can compose services in two ways. The first one is *pull driven*—a sink service looks for intermediate filter services and a source service to provide an end to end communication flow. An example of such a composition is RealPlayer that looks for a streaming source. The second way is *push driven*—it makes the network infrastructure the active part of the communication: a source service looks for a suitable sink to deliver a flow in a proactive manner, e.g. an e-mail notifier looks for a mailer on a PDA to inform the user about e-mails upon entering a personal communication environment. Figure 2 illustrates these approaches with

two chains of services: the composition of an application in the pull model starts with a sink that finds a source and places a filter (or transcoder) if needed. In the push model, the composition begins with the source that finds a suitable sink with an optional filter in between. The graph of service composition may be more complex than the simple chains shown in the figure. Although we provide both composition methods, we believe that the push model corresponds better to the requirements of Omnisphere.

The user avatar is the central element in the operation of Omnisphere. It takes care of all the operations that should be performed in the network infrastructure on behalf of the user. In this way most of the operations are delegated into the network infrastructure and are automated as much as possible. When an appliance enters Omnisphere, it authenticates itself so that based on the User ID and Appliance ID, Omnisphere can retrieve the `User Preferences` and `Device Capabilities`. `User Preferences` define what are the most common needs of the user in terms of services and different types of data and `Device Capabilities` provide information about the characteristics of appliances (screen resolution, processor speed, network interface). It also uses `Context` that adds some location specific information such as the geographic location, available devices, type of environment—public or private, and all other information related to the close neighborhood of an appliance. It then creates an avatar that discovers relevant services using a service discovery protocol. The discovered services are matched with the required characteristics provided by `User Preferences`, `Device Capabilities`, and `Context`. They are then configured and activated. If needed the avatar contacts the Omnisphere controller on the user appliance to run the user applications that receive data flows. To take into account state changes, for example when an appliance becomes inactive, or a new appliance enters Omnisphere, the service bindings are re-evaluated.

Such a discovery process relieves appliances, which may have limited resources, from the operation that may consume scarce resources and may require the availability of different discovery protocols on the appliance. It may also save critical radio channel resources, because only selected information is provided to the appliance.

3.1 Example

We present below an example of e-mail notification. Alice wants to use a mailer such as Netscape or Eudora at the office. Instead of running them all the time to periodically check her mailbox via POP, she wants the mailer to be run upon the arrival of a new mail and configured to use ssh tunnels with the company POP and SMTP servers. When she leaves the office, she wants to receive e-mail notifications as SMS messages on her cellular phone for some urgent and important messages. When attending a meeting, she wants to be notified on her PDA to decide whether it is worthwhile to read a mail. Upon arrival at home, she wants to read the mail on the home PC by using the SMTP server of her ISP.

Let us start with `User Preferences` for Alice.

Communication	Attributs	Preferred User Applications
`Mail`	`domain=imag.fr` `transport=secure`	`laptop-netscape, pda-popup, gsm-sms`

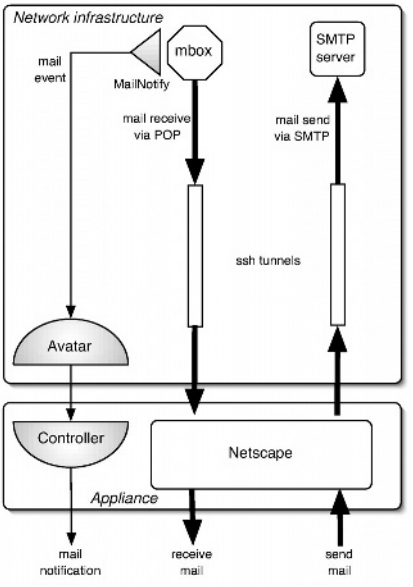

Fig. 3. Mail example

This information means that Alice wants to use her mailbox in the `imag.fr` domain and her preferred applications are the usual Natscape mailer on her laptop, a popup dialog window for deciding whether to read a mail, or a notifier that sends a SMS message to her cell phone. Each of these applications can be configured with some additional information such as the port number or the cell phone number.

The composition of ambient services is presented in Figure 3. A user mailbox exists on a server in `domain=imag.fr` and there is a service MailNotify able to check for a new mail of a given user. If Alice is in her office, Omnisphere looks for services that generate type `/mixed/e-mail:incoming` (we define a simple hierarchy of types similar to the DNS names). The query returns the addresses of incoming mail servers (POP, IMAP, HTTPS). Then, Omnisphere looks for a service that generates type `/mixed/e-mail:event`. In our example, the query returns the MailNotify service. A query for `/mixed/e-mail:outcoming` yields an outgoing local SMTP server.

User applications such as Netscape register with the service discovery protocol so that Omnisphere can find them. If several applications are discovered, in the case for example when Alice is in the office with her laptop and a PDA, Netscape is chosen as the preferred user application. At this point all the ele-

ments are discovered and Omnisphere can configure them and compose to form the complex Mail application.

When Alice goes to a meeting room, the avatar detects that she is not longer present in the office and stops the mailer. According to the preferences, it discovers a SMS notifier, configures it with the Alice's cell phone, and forwards notification events coming from the MailNotify service. In this way, Alice receives automatically an excerpt of the mail on her cell phone. Then, Omnishpere in the meeting room detects the Alice's PDA when she enters the room. It requests the migration of the avatar to the current location. The avatar stops forwarding mails to the cell phone and sends them to the Omnisphere controller on the PDA which pops up a window to ask Alice what to do with the incoming mail. She may decide to read it using a mailer on the PDA.

In general, there can be multiple instances of discovered services. In this case, we need to add sufficient attributes to `User Preferences` in order to help resolving such ambiguities.

4 Conclusion

We believe that future personal communication environments will be based on the *push driven* paradigm: sources of data push information towards the user and the network infrastructure takes care of mediating such communication flows. We experiment with this approach by designing and implementing Omnisphere. We have already prototyped the main elements of Omnisphere: the user avatar and the Omnisphere controller. The avatar uses the DNS-SD for service discovery—we have modified the Apple Rendezvous implementation of DNS-SD to take into account our definition of data types and wildcard service queries.

Our current work concerns seamless handover of the user between different places, the detection of the user movement, and implementing more complex scenarios such as *follow me video*—the user can continuously watch a video stream presented on different appliances when moving between different places.

References

1. F. Rousseau, J. Oprescu, L-S. Paun, and A. Duda. Omnisphere: a Personal Communication Environment. In *Proc. Thirty-Sixth Annual Hawaii International Conference on System Sciences (HICSS-36)*, Big Island, Hawaii, 2003.