Service Delivery Platforms

Developing and Deploying Converged Multimedia Services

Edited by

Syed A. Ahson • Mohammad Ilyas

CRC Press
Taylor & Francis Group
AN AUERBACH BOOK

Service Delivery Platforms

Developing and Deploying Converged Multimedia Services

Edited by
Syed A. Ahson • Mohammad Ilyas

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
AN AUERBACH BOOK

Auerbach Publications
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed in the United States of America on acid-free paper
10 9 8 7 6 5 4 3 2 1

International Standard Book Number: 978-1-4398-0089-8 (Hardback)

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the Auerbach Web site at
http://www.auerbach-publications.com

Contributors

Sofiane Abbar
PR*i*SM Laboratory
University of Versailles-Saint-Quentin-en-Yvelines
Versailles, France

Nilanjan Banerjee
IBM Research India
New Delhi, India

Mokrane Bouzeghoub
PR*i*SM Laboratory
CNRS and University of Versailles-Saint-Quentin-en-Yvelines
Versailles, France

Teck Yoong Chai
Institute for Infocomm Research
A*STAR (Agency for Science, Technology and Research)
Singapore

Dipanjan Chakraborty
IBM Research India
New Delhi, India

Byung-Chul Choi
Department of Engineering and Technology Management
Portland State University
Portland, Oregon

Rolan Christian
Centre for Telecommunications Access and Services (CeTAS)
School of Electrical and Information Engineering
University of the Witwatersrand
Johannesburg, South Africa

C. Costache
Siemens Program and System Engineering
Brasov, Romania

Tugrul U. Daim
Department of Engineering and Technology Management
Portland State University
Portland, Oregon

M. Demeter
Siemens Program and System Engineering
Brasov, Romania

Goran Đorđević
Institute for Manufacturing Banknotes and Coins NBS
Beograd, Serbia

Richard Good
Centre for Broadband Networks
University of Cape Town
Cape Town, South Africa

Athanasios Karantjias
Department of Computer Science
University of Piraeus
Piraeus, Greece

E. Kayafas
National Technical University of Athens
Athens, Greece

Helmut Krcmar
Information Systems
Technische Universität München
München, Germany

Teck Kiong Lee
Institute for Infocomm Research
A*STAR (Agency for Science, Technology and Research)
Singapore

Jan Marco Leimeister
Information Systems
Universität Kassel
Kassel, Germany

Stéphane Lopes
PR*i*SM Laboratory
University of Versailles-Saint-Quentin-en-Yvelines
Versailles, France

Han-hua Lu
Nanjing University of Posts and Telecommunications
Nanjing, People's Republic of China

Milan Marković
Security Department
Banca Intesa ad Beograd
Beograd, Serbia

Li-juan Min
Nanjing University of Posts and Telecommunications
Nanjing, People's Republic of China

Toshiyuki Misu
NEC Corporation
Tokyo, Japan

Sumit Mittal
IBM Research India
New Delhi, India

A. V. Nedelcu
Transilvania University of Brasov
Brasov, Romania

Lek Heng Ngoh
Institute for Infocomm Research
A*STAR (Agency for Science, Technology and Research)
Singapore

George Pentafronimos
Department of Computer Science
University of Piraeus
Piraeus, Greece

Nineta Polemi
Department of Computer Science
University of Piraeus
Piraeus, Greece

Christoph Riedl
Information Systems
Technische Universität München
München, Germany

D. N. Robu
Siemens Program and System Engineering
Brasov, Romania

Joaquín Salvachúa Rodríguez
Telematics Engineering Department
Polytechnic University of Madrid
Madrid, Spain

Judith E. Y. Rossebø
Department of Automation Networks
ABB Corporate Research
Billingstad, Norway

Ragnhild K. Runde
Department of Informatics
University of Oslo
Oslo, Norway

Luis Angel Galindo Sánchez
Innovation and Strategy
Telefonica Spain
Madrid, Spain

F. Sandu
Transilvania University of Brasov
Brasov, Romania

Anett Schülke
NEC Laboratories Europe
NEC Europe Ltd.
Heidelberg, Germany

Xu Shao
Institute for Infocomm Research
A*STAR (Agency for Science, Technology and Research)
Singapore

Richard Spiers
Centre for Broadband Networks
University of Cape Town
Cape Town, South Africa

Yan-fei Sun
Nanjing University of Posts and Telecommunications
Nanjing, People's Republic of China

Joseph Chee Ming Teo
Institute for Infocomm Research
A*STAR (Agency for Science, Technology and Research)
Singapore

Neco Ventura
Centre for Broadband Networks
University of Cape Town
Cape Town, South Africa

Ya-shi Wang
Nanjing University of Posts and Telecommunications
Nanjing, People's Republic of China

Jing Zhang
School of Economics and Management
Beijing University of Posts and Telecommunications
Beijing, People's Republic of China

Shun-yi Zhang
Nanjing University of Posts and Telecommunications
Nanjing, People's Republic of China

Yong Zheng
Nanjing University of Posts and Telecommunications
Nanjing, People's Republic of China

Luying Zhou
Institute for Infocomm Research
A*STAR (Agency for Science, Technology and Research)
Singapore

Contents

Chapter 1

Service Composition and Control for Next-Generation Converged Applications

Nilanjan Banerjee, Dipanjan Chakraborty, and Sumit Mittal

Contents

1.1 Introduction

Convergence between the two largest networks (Telecom and IP) is taking place very rapidly and at different levels: (1) network level: unification of IP networks with traditional Telecom networks through evolving standards (Session Initiation Protocol (SIP), Realtime Transfer Protocol (RTP), SS7, 3G) to support interoperability; (2) service level: traditional Telecom services like *voice calls* are being provisioned on the IP backbone (VoIP), while traditional IP services (most data-driven services such as multimedia, browsing, chatting, gaming, etc.) are accessible over the Telecom network.

Significant investment has been made at different layers of the converged stack, toward supporting data-driven enriched services such as multimedia, gaming, browsing, and so on. Such investments range from core network infrastructure changes (2G to 2.5G, leading to 3G, 4G) to adoption experiments with several standards (e.g., Wireless Application Protocol (WAP)) to support data-driven services (e.g., news feeds, mobile commerce, location-based service access), traditionally accessed over the IP network. Most content providers today have a *mobile* version of their information portals, which is accessible using mobile and cellular devices. Different game changing and competing technologies (e.g., Global Positioning System (GPS) versus cellular triangulation-driven location services, VoIP versus traditional telephone calls) in this converged market place are driving the big players to force an evolution in their business models.

With the market reaching saturation in several countries and revenues from voice calls decreasing rapidly, Telecom operators are aggressively looking at newer sources of revenue. So far, the typical model for providing data-driven services over the Telecom infrastructure has been through partnerships with *content providers*. In recent years, however, these partnership-driven services have been facing strong competition from *similar* technologies and applications provided by Internet Content providers. These applications can be accessed through a browser-enabled phone, while paying only for the connectivity charges, and thereby adversely affect revenues from the paid-for services hosted on the Telecom operator portal. Examples of such services range from VoIP and telephony conferencing services (e.g., skype, lycatalk etc.) to various content services (maps, ringtones, etc.). An increasing number of mobile users are now using browser-enabled phones to access these services, bypassing the Telecom portal. For example, it has been estimated that 60% of the mobile content traffic in the United States and 90% in Europe is off portal [1].

Telecom operators, however, have an edge over Internet service providers in terms of their still unmatched core functionalities of location, presence, call control, and so on, characterized further by carrier-grade Quality-of-Service (QoS) and high availability. As an example, imagine a next-generation instant conferencing application that dynamically connects friends who are *in the same location* in real time (e.g., mall) with a single click. Further, each user's device dynamically redirects/rejects calls depending on real-time preferences (e.g., allow access to my location only if I am in a public place).

Enablement of several such enriched converged services requires the core functionalities of the Telecom operator to be easily *accessible* and *composable* with third-party services providing the core application logic. Moreover, the underlying converged infrastructure (IP + telephony) needs to be smart enough to be able to provide and manage such enhanced converged services. Such services and applications require enhanced message routing and control in the core stack.

As we can see, a potential channel for the Telecom operators to increase their revenue is to offer these functionalities as services to developers for creating such new innovative applications. These developers can belong to not only the select partners of the Telecom operator, but also those involved in creating a variety of long-tail applications [2]. Additionally, due to the IP and telephony networks converging, developers can also compose their applications with third-party services available on the IP network. For example, Location and Presence information from Telecom can be clubbed with Google Maps to provide new workforce management solutions for mobile settings [3].

Recognizing the potential, Telecom operators have started investing heavily to redesign their back-end support systems (e.g., billing, provisioning, and network support systems) to address the challenges of providing the user with a unified communication, collaboration and service access experience. However, the core functionalities of location, presence, call control, and so on were so far used only internally by the carrier operator's core services (e.g., calls). As such, they were not accessible outside the network, and not easily integrable and interoperable with third-party services.

Toward alleviating these problems, there has been a concerted effort toward the creation of a blueprint for a common service delivery platform (SDP) over the past few years. Next-generation SDP* is an architectural solution that enables the reuse of service components trapped in "vertical service silos" by adopting a horizontal layered approach. There are capabilities beyond network enablers that need to be exposed through the operators SDP—these include mobility, operation support system–billing support system (OSS–BSS), functions such as billing and provisioning, subscriber profiles, QoS attributes of network elements, as well as profiles of devices supported by the operators' network.

* http://en.wikipedia.org/wiki/Service Delivery Platform

The salient components of an SDP are service creation environment, service orchestration environment, service execution environment, and service control and management environment. A number of standard bodies are working to come up with reference architecture for SDP. Examples are open mobile alliance service environment (OSE), Telemanagement Forum's Service Delivery Framework (SDF), Open Service Access (OSA) Parlay, and so on. Most of these reference designs expose the Telecom capabilities through flexible service-oriented architecture (SOA)-based programming interfaces such as Parlay-X [4]. Recently, Telecom operators are also exposing their capabilities through lightweight mashable interfaces or widgets to compete with the Web 2.0 Internet service providers. One such example is BT Web 21C [5]—a service aggregation environment that enables a user to create Telecom mashup applications very easily.

This chapter focuses on two integral aspects of next-generation SDPs—service *creation* and *control* infrastructure during service execution. In particular, it captures some of the inherent challenges in effective composition and control of Telecom capabilities for next-generation converged applications, in an open, collaborative environment—challenges that are inadequately addressed in existing standardized frameworks and protocol specifications.

1.2 Service Composition Framework

As mentioned earlier, core Telecom functionalities need to be accessible and composable with converged IP services to support the evolving business models. This leads to rapid integration and reuse of these services. Web 2.0-oriented mashup-style applications should also be able to effectively use Telecom functionalities (e.g., call services, SMS, location, etc.) in their application creation environments.

To support the basic operations such as voice and SMS, the building blocks of a Telecom infrastructure—location registries (Home Location Register/Visitor Location Register), accounting and billing services, SMS gateways, call servers, and so on—are already in place. However, these are not easy to utilize in new applications because they are not exposed using standardized frameworks and component models. Toward this, Telecom operators are steadily adopting SOA that would let developers access these services without knowledge of the underlying platform implementation. Web services, as an instantiation of SOA, have received much interest in the community due to their potential in facilitating seamless business-to-business or enterprise application integration. The Parlay consortium has defined a standard, called *Parlay-X* [4], that exposes Web service interface for core Telecom functionalities. On similar lines, IP Multimedia Subsystem (IMS) [6] provides a reference framework to expose these functionalities as services to Web-engineered systems using SIP [7].

Although efforts like Parlay-X and IMS are a step in the right direction, the rapid development of applications that utilize Telecom functionality still faces a number of challenges in a realistic setting. First, one needs to provide interfaces that shield the converged service developer from different Telecom protocols (Parlay-X, SIP, etc.), including the legacy ones. Second, one needs to package the Telecom functionalities so that they can be *readily* used in different programming styles (Java, HTML/JavaScript, etc.) other than pure Web service-based composition (e.g., BPEL). Finally, one also needs to encapsulate the invocation of Telecom functionality with various *coordination* rules, for example, those that correspond to managing the usage of a service, including monitoring, metering, and access control.

In this section, we focus on the above challenges and present SewNet—a framework that addresses these to enable rapid composition of Telecom services. In addition, we also provide a survey of related art in this space. SewNet is a service composition environment designed bottom-up to support rapid integration of Telecom functionalities with converged services. Its design is based on its own Telecom Service Reference, Encapsulation, and Coordination (T-Rec) "Proxy" model. This enables developers to seamlessly incorporate Telecom functionality and apply various coordination rules. Further, SewNet provides an eclipse-based* service composition environment based on the T-Rec model. SewNet can be used by different categories of developers, including Java, BPEL, and HTML/JavaScript programmers.

It is important to note that although we focus on Telecom here, the T-Rec proxy model is generic and applicable to third-party services available on the Web.

1.2.1 Problem Illustration and Motivation

We illustrate the problem with respect to the component-oriented diagram of a service that utilizes Telecom functionality, as shown in Figure 1.1. In general, such an application can be broken into two major blocks. First, there are *Telecom blocks* (represented as black rectangular boxes in the figure) that invoke a Telecom network functionality (e.g., invoking the location service or capabilities like SMS, Third-Party Call Control). The others are *non-Telecom blocks*, where the developers can

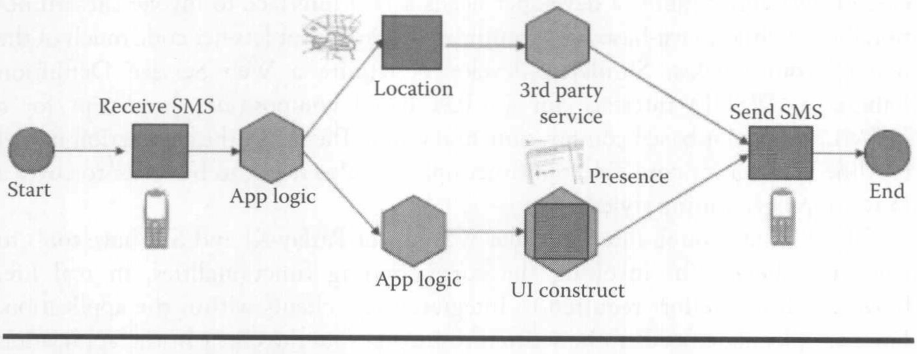

Figure 1.1 Model of a Telecom service.

* http://www.eclipse.org

embed various constructs (depicted by gray hexagonal boxes in the figure). For example, in a workforce management solution, these blocks can contain logic for scheduling agents on the basis of Location and Presence information provided by Telecom operator. Alternatively, these blocks can be user interface (UI) constructs, for instance, those enabled by various Ajax-based platforms. Finally, the non-Telecom blocks can also be invocation points for orchestration with third-party services available over the Web. Also, as the figure shows, to bind these Telecom and non-Telecom blocks, specification of complex control and data flows is also required during the service design.

We believe that for application development utilizing Telecom blocks composed with the non-Telecom ones, the following challenges need to be addressed:

First, functionality available within a Telecom operator is, in general, exposed using multiple protocols. For example, Presence-related information can be accessed via the SIP protocol, and Messaging capabilities using the Short Message Peer-to-peer Protocol (SMPP). As a step toward hiding protocol heterogeneity and complexity, the Parlay consortium has come up with the Parlay and the subsequent Parlay-X standards. Parlay-X exposes a Web Services interface for several Telecom functionalities. However, it does not cover the whole gamut of functionalities that can be offered by the Telecom operator, especially those requiring session control. Further, some of the Telecom functionalities can also be exposed through legacy protocols. Therefore, we need an *abstraction* model that provides interfaces shielding the application developer from the underlying protocols. This model should also allow seamless switching between different protocols, for example, when moving from legacy interfaces to the Parlay-X ones.

Second, developers who want to utilize Telecom functionality in their application can belong to different categories [2]. More specifically, composite applications modeled in Figure 1.1 can be written in Java, BPEL, HTML/JavaScript, and so on. Although editors corresponding to the various programming styles provide the developer with constructs for the non-Telecom blocks, they still require the Telecom functionalities packaged in a format suitable for incorporation. For example, in the case of Java applications, a developer needs a Java interface to invoke these functionalities (while a Java-based programming environment lets her code much of the non-Telecom blocks). Similarly, developers require a Web Service Definition Language (WSDL) interface for a BPEL-based composition, JavaScript for a HTML/JavaScript-based composition, and so on. Therefore, the abstraction model (outlined above) for core Telecom functionalities also needs to be *broad* to cover a range of programming styles.

Third, even though interfaces like WSDL (for Parlay-X) and SIP have tools to generate "clients" for invoking the corresponding functionalities, in real life, however, there is effort required to integrate these clients within the application. For example, code needs to be written to incorporate the client in the application code, while taking care of tertiary library dependencies for this client. It would help the developer immensely if the abstraction model pregenerates the clients corresponding to different programming styles, and packages them in a *structured* manner. Having a well-defined structured format would enable any application development environment (with some extensions to interpret this structure) to integrate these clients seamlessly.

Finally, when Telecom functionality gets used in an application, the Telecom operator, as well as the application developer, wants to coordinate its usage. For instance, one needs mechanisms for embedding logic for charging, specifying access control policies, and so on. Furthermore, with recent trends suggested by Web 2.0, application developers should be able to contribute, implicitly or explicitly, to the enrichment and refinement of the exposed Telecom functionality (and its usage). Therefore, the abstraction model needs to be *rich* enough to enable all of this.

A number of operators are already moving in the direction of making their core functionalities available for application development. For example, British Telecom has released a Software Development Kit [5] that enables its network services to be utilized in Web mashups. However, what is missing is an abstraction model which is broad, structured, and rich, as motivated above. We describe such a model next, and thereafter present a service creation framework on top of this model.

1.2.2 T-Rec Model

Figure 1.2 represents the basic concept of the T-Rec "Proxy" model. In essence, once an application has been broken down into Telecom and non-Telecom blocks, this model is used to realize the Telecom blocks, considering the programming style, while also enabling mechanisms for coordination and enrichment. In practice, these proxies would be created by a Telecom operator and be made available to application developers over the Web (or a converged IP network).

1.2.2.1 T-Rec Structure

The rich, structured T-Rec Proxies consist of the following elements:

- *Proxy representation:* Contains signatures of the methods (Application Programming Interface (API)) exposed by the proxy along with a textual

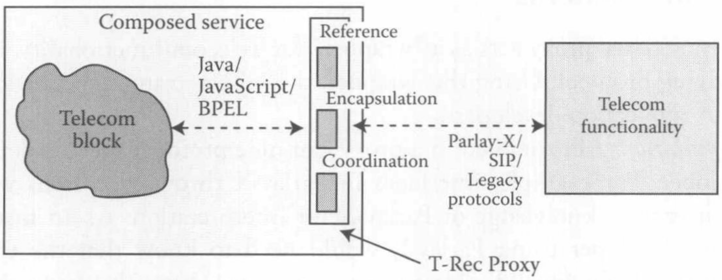

Figure 1.2 T-Rec "Proxy" model.

description of the service it represents. The APIs are designed to hide protocol-specific details and abstract the Telecom functionality to the programming language level. As discussed before, APIs corresponding to multiple styles (Java, BPEL, and JavaScript) should be created to support different environments.

- *Implementation:* This module connects to the Telecom service using the underlying protocol, and is available in different formats. For example, the implementation could be in the form of a *.jar* file for a Java proxy, a *.js* file for JavaScript, or could be encapsulated by visual constructs, such as widgets, and used inside HTML pages.
- *Configuration file:* Proxies come with a default setting but can be further configured by developers. This includes assigning default values for some of the parameters in an API, specifying the access control list, and so on. Such settings could also be functionality specific, for instance, restricting the size of SMS messages.
- *Metadata:* To enable easy look up, keywords and tags related to the proxy functionality are associated with it. New tags can be added to the proxies if required; for example, when a developer utilizes a proxy in a way that was not originally foreseen by its creator.
- *Utility snippets:* The proxies are populated with multiple code snippets on top of the basic functionality. For example, a "Presence" proxy may have a program fragment that parses the returned response (usually an XML document) for different attributes. These utilities can be suggested, in an appropriate manner, to developers who intend to use the proxy.
- *Unit test code:* Proxies contain codes that let different APIs supported in the proxy be tested in isolation. These are very helpful during testing and debugging.
- *Link to blogs:* Each proxy is linked to a blog entry where developers can log their experience of using the proxy. If multiple proxies are suggested during a look up, analyzing the blog entries can help the developers choose the most appropriate one for their task.

1.2.3 T-Rec Benefits

Intuitively, a T-Rec proxy acts as a "wrapper" for Telecom functionality, including its underlying protocol. Using this wrapper, the proxy creator can provide several benefits to application developers.

Encapsulation: APIs defined in a proxy can hide protocol-specific details from the developer. For example, interfaces in Parlay-X throw exceptions with error codes that require knowledge of Parlay-X for interpretation. As an instance, an application developer using Parlay-X would need to know that the error code *SVC0004* stands for invalid addresses in a message. Using the proxy model, we can encapsulate these error codes with higher-level exceptions, such as throwing *InvalidAddressException* whenever error code SVC0004 is returned. Moreover, using proxies, similar APIs can be exposed across different protocols. For example, various APIs in the Location proxy can have similar signatures for Parlay-X and SIP-Presence-based implementations.*

Coordination: When Telecom proxies get used in an application, the Telecom operator as well as the developer can manage and meter its usage. For instance, whenever the proxy corresponding to Location information gets invoked within an application, the Telecom operator can authenticate the developer and also charge some amount. In this case, proxies are configured to collect the relevant information, for example, developer Id, from the developer and send it to the operator. Similarly, the developer can configure the Location proxy to cache the location information locally within itself, and avoid connecting to the operator's infrastructure at each invocation.

The proxy model also provides an easy mechanism to incorporate various business contracts between the operator and the developer. For example, the implementation module in a proxy can be extended to make the proxy display advertisements on behalf of the operator, whenever it is invoked. In this case, logic can be such that the proxy picks what to advertise on a real-time basis.

Collaboration and reuse: Using the proxy model, developers can collaborate, share, and contribute toward enriching Telecom functionality. For instance, the user of the Location proxy in an application can publish a utility to parse the output of this service. This utility can be reused by other developers while incorporating this proxy in their applications. Similarly, the proxies can be configured to provide updates to a developer about new entries on the blog, utilities published recently, bug fixes, and so on. In the case of bug fixes, logic can be embedded in the proxy to automatically download the latest implementation modules.

1.2.4 SewNet Details

Next, we present the SewNet framework that utilizes the rich, structured T-Rec proxy model to enable seamless weaving of Telecom functionality with application logic and other constructs required to develop a service.

1.2.4.1 SewNet Architecture

As Figure 1.3 shows, SewNet has two main architectural components—SewNet Core and Composition Studio(s).

SewNet Core forms the backbone architecture that exposes Telecom functionality to developers through simple, intuitive interfaces for lookup and selection while allowing for developer participation and feedback through publishing and blogging.

* In SIP, location information is obtained by subscribing for the presence information, and parsing the returned document. We can wrap this under a *getLocation()* interface.