ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften und Informatik
Institut für Verteilte Systeme

Instant-X: SOA for Multimedia Communication in NGNs

Jan-Patrick Elsholz, Holger Schmidt, Sven Schober, and Franz J. Hauck

Technical Report Nr. VS-R11-2009 | 2009-11-10

This work was presented at the Kick-off Meeting GI/ITG KuVS Fachgespräch “NGN Service Delivery Platforms & Service Overlay Networks” November 13, 2009 at the premises of Fraunhofer Institute FOKUS, Room 1008, Kaiserin-Augusta-Allee 31, 10589, Berlin, Germany
http://www.fokus.fraunhofer.de/en/ngni/news_events/events/kuvs_sdp_fg/kick_off_meeting/index.html

Instant-X: SOA for Multimedia Communication in NGNs

Jan-Patrick Elsholz, Holger Schmidt, Sven Schober, Franz J. Hauck
Institute of Distributed Systems
Ulm University, Germany
Email: {jan-patrick.elsholz,holger.schmidt,sven.schober,franz.hauck}@uni-ulm.de

I. Introduction

Our society requires the ability to communicate everywhere. Social networks and globalisation even strengthen this need. Upcoming next generation networks (NGNs) [1] provide the required base infrastructure. Yet, current communication software is built on top of basic protocols and available encoders/decoders. Further mechanisms, such as signalling, have to be coordinated and configured by the application developer. Thus, the development of such software is a non-trivial task. Due to the fact that proprietary implementations do not share a common application programming interface (API), developers have to change the application for later integration of novel protocols. This leads to inflexible and incompatible software.

In this work, we present the novel multimedia middleware *Instant-X*. In contrast to related work, *Instant-X* introduces a programming model with an API that abstracts from specific multimedia communication protocols and technical details. This eases development and allows reusing and replacing protocols without changing the application code. *Instant-X* uses OSGi [2] as a component platform providing multimedia functionality in terms of services according to the service oriented architecture (SOA) [3]. This allows dynamic deployment of components, such as provider- and network-specific ones. These are dynamically discovered, selected and loaded with our OSGi for the Cloud (OSGi4C) platform [4]. We advocate that only an integrated middleware, such as *Instant-X*, is able to implicitly manage the dynamic coordination and configuration of multiple multimedia application components, which highly eases application development.

II. Instant-X

In the following, we sketch our *Instant-X* approach with respect to our programming model and the generic API. The former is illustrated by a small example.

A. Programming Model

There are three fundamental elements of our programming model: *binding*, *session* and *context*.

A *binding* is the local endpoint of a single application represented by a uniform resource identifier (URI). The binding is responsible for activating this URI (e. g., register a session initiation protocol (SIP) URI and keep this registration alive). A finite state machine reflects this expiring activation (e. g., keep-alive interval for SIP registration).

A *session* represents a peer-to-peer (P2P) relationship between two or more actors (i.e., the participants). The actors are uniquely identified by URIs, such as a *SIP URI* [5]. These URIs are encapsulated in *bindings*. Thus, the session refers to its local binding and contains the URI of a remote binding. Analogue to the session term of the ITU-T X.200 standard [6], a session is mandatory for connection- as well as connectionless-mode communication (e. g., streaming vs. instant messaging). Again, a finite state machine represents the state of the P2P relationship.

A *context* contains optional parameters for session and binding. For example, the credentials needed for registering a SIP URI.

Figure 1 shows an example of our programming model. Here, two participants establish a *session*, in which multimedia data is transferred. A SIP session is used to exchange the necessary parameters for multimedia data, which is transferred from the data source to the data sink in terms of real-time transport protocol (RTP) sessions (e. g., audio and video streams). These depend on the major SIP session. The source and the sink are represented by URIs and therefore encapsulated into bindings.

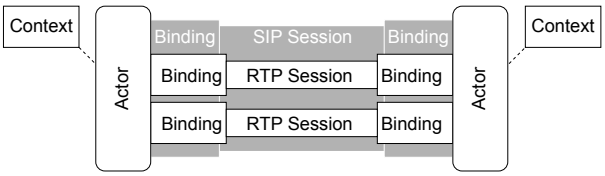

Fig. 1. Example of our *Instant-X* programming model

B. Generic API

The basic idea of our *Instant-X* platform is a generic API with different protocol implementations.

Each task of the API (e. g., signalling) is encapsulated into a separate OSGi bundle exporting the functionality as an OSGi service (see Figure 2). Whereas the programming model remains the same, various protocols may be used without changing the application itself. Through this loose coupling of protocols (i.e., protocol services) according to the SOA paradigm, we gain a high reusability of code. The underlying OSGi infrastructure with our OSGi4C platform takes care of dependencies and resolves them automatically. Figure 2 illustrates our approach with a Voice over IP (VoIP) example. The VoIP application itself is an ordinary OSGi bundle using our generic *Instant-X* API. Behind this API there are different protocol services for signalling and data transmission, such as the InterAsterisk eXchange version 2 protocol (IAX2) [7], SIP [5], H.323 [8], RTP and the user datagram protocol (UDP). According to the supplied URI from the application, our generic API selects the appropriate service for the specific protocols. For this purpose, the OSGi service registry is used. In case of a missing service, our OSGi4C platform automatically tries to discover and download the needed OSGi bundles [4]. This process is transparent for the VoIP application. If no appropriate protocol bundle is found, an exception will be returned.

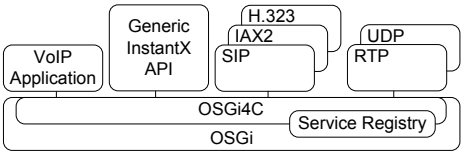

Fig. 2. Example of a VoIP application on the *Instant-X* platform

This generic API allows developing applications, which remain unchanged, even in case of protocol changes. Thus, integrating upcoming protocols or enhancing applications to existing standards afterwards is not a big issue. This highly eases application development.

C. Programming Model Example

Figure 3 outlines a SIP-based VoIP application implemented with *Instant-X*. The example shows a SIP session between alice@a.com and bob@b.com on Alice's side. For the purpose of clarity, the data transmission is indicated by a greyed-out RTP session. However, the RTP session is handled analogue to the SIP session described below.

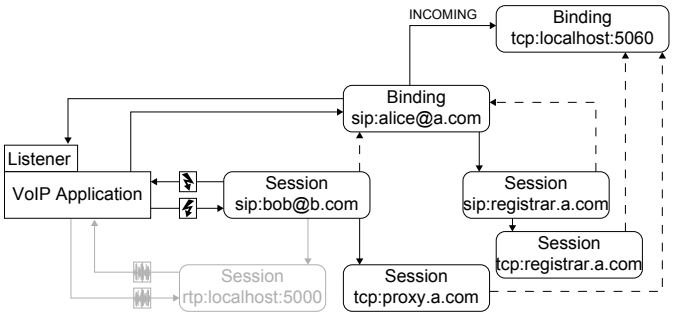

Fig. 3. Example of a SIP-based VoIP application with *Instant-X*

Initially, the VoIP application creates a binding (i.e., a SIP binding). For this purpose, our generic *Instant-X* API transparently calls the underlying protocol implementation. Then, the VoIP application registers itself as listening application for incoming sessions. In this example, the SIP binding uses a TCP binding.

In addition to the two participating actors, the SIP infrastructure requires communication with other network entities, such as registrar and proxy. Thus, the SIP binding uses a registrar session according to our session definition. In order to transmit signalling messages to the remote actor, e. g., bob@b.com, the application creates a SIP session with the initially created SIP binding. Depending on the context, the SIP session sends the messages directly to the destination or uses a proxy session for sending the messages via the SIP proxy. Incoming signalling messages arrive at the SIP binding, which informs the registered application. The messages are passed to the application in terms of a SIP session. If there is none it will be created. Additionally, the SIP binding takes care of registering the local URI at the SIP registrar. For this purpose, it uses a registrar session. In our example, this registrar session internally uses a TCP session to transmit the registration messages.

III. Related Work

There is a lot of software supporting the development of multimedia applications by providing basic components, such as multimedia codecs. For instance, *DirectX* [9] provides COM components, and Java media framework (JMF) [10] as well as *Apple Quicktime* [11] provide Java libraries. JMF even supports RTP for multimedia streaming. Still, these mechanisms have to be coordinated and configured.

NGN defines a communication architecture where service-related functions are independent from underlying transport-related technologies with a separation of control functions [1]. One example is the *IP multimedia subsystem* (IMS). Unlike IMS, we do not distinguish between *signalling sessions* and contained *media objects*. Thus, our approach actually reflects real-world protocol implementations, such as IAX2, and we provide an even more generic API. The Java specification request (JSR) 281 [12] defines a high-level IMS Services API for easier development of applications. Yet, it strongly focuses on SIP and RTP as basic protocols.

The *Network-integrated Multimedia Middleware* (NMM) is a middleware for distributed multimedia services [13]. It focuses on home entertainment and allows discovering available devices. These can be integrated into a multimedia flow graph, which describes the media flow from the source to the target. Due to the fact that signalling is part of the media stream, signalling protocols such as SIP cannot be used with all features (e. g., call referral). Unlike *Instant-X*, NMM does not provide a generic API and thus does not allow dynamic replacement of underlying protocols at runtime.

There are approaches for generic APIs for dedicated purposes. *Quartz* [14] provides a generic interface for QoS management and *Noja* [15] one for data transmission. Yet, both systems are restricted to specific parts or certain protocols of multimedia applications. The idea of *Instant-X* is to provide an integrated fully-fledged solution.

IV. Conclusion

We presented *Instant-X*, a novel service-based middleware for distributed multimedia applications in NGNs. *Instant-X* provides a novel programming model with a generic API for essential multimedia application tasks with respect to signalling and data transmission. This API abstracts from concrete implementations, which allows replacing protocol implementations without changing the application code. This highly eases developing multimedia applications. For our prototype, OSGi bundles provide multimedia functionality in terms of services. This allows sharing common multimedia tasks in multiple applications according to the SOA paradigm. *Instant-X* offers dynamic deployment of unavailable protocol implementations. Altogether, it builds a powerful platform for spontaneous communication applications.

References

[1] ITU, *General overview of NGN (ITU-T Recommendation Y.2001)*, 2004.

[2] OSGi Alliance, "OSGi Service Platform: Core Specification, Release 4," 2007.

[3] W. Schulte and Y. V. Natis, "'Service oriented' architectures, part 1," Gartner, SSA Research Note SPA-401-068, April 1996.

[4] H. Schmidt, J. Elsholz, V. Nikolov, F. Hauck, and R. Kapitza, "OSGi4C: Enabling OSGi for the Cloud," in *COMSWARE '09*. ACM, 2009.

[5] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol (IETF RFC 3261)," The Internet Society, 2002.

[6] ITU, *Information technology – Open Systems Interconnection – Basic reference model: The basic model (ITU-T Recommendation X.200)*, 1994.

[7] M. Spencer, B. Capouch, F. Miller, and K. Shumard, "IAX: Inter-Asterisk eXchange Version 2 (IETF Internet-Draft <draft-guy-iax-04.txt>)," 2008.

[8] ITU, "Packet-based multimedia communications systems (ITU-T Recommendation H.323)," 2006.

[9] Microsoft, "DirectX," http://www.microsoft.com/windows/directx/, 2008.

[10] Sun Microsystems, "Java Media Framework API Guide," http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/guide/, 1999.

[11] Apple Computer, "QuickTime," http://www.apple.com/quicktime/, 2008.

[12] Sun Microsystems, "JSR 281 IMS Services API," http://jcp.org/en/jsr/detail?id=281, 2009.

[13] M. Lohse, M. Repplinger, and P. Slusallek, "An Open Middleware Architecture for Network-Integrated Multimedia," in *IDMS/PROMS '02*. Springer, 2002.

[14] F. Siqueira and V. Cahill, "Quartz: A QoS Architecture for Open Systems," *ICDCS '00*, vol. 0, p. 197, 2000.

[15] A. Eichhorn, "Middleware Abstractions for Cross-Layer controlled Media Streaming," in *MAI '08*. ACM, 2008.