Design of a QoS Signaling API for Advanced Multimedia Applications in NGN

Ognjen Dobrijevic[1], Miran Mosmondor[2], Maja Matijasevic[1]
[1]*University of Zagreb, Faculty of Electrical Engineering and Computing*
Unska 3, HR-10000 Zagreb, Croatia
[2]*Ericsson Nikola Tesla, R&D Center, Krapinska 45, HR-1000 Zagreb, Croatia*
{ognjen.dobrijevic, maja.matijasevic}@fer.hr, miran.mosmondor@ericsson.com

Abstract

As a standardized NGN architecture, the 3rd Generation Partnership Project IP Multimedia Subsystem (IMS) defines standard service capabilities and interfaces, as well as a common IP-based infrastructure, which enable operators to introduce new services in an efficient and flexible way. In order to exploit the capabilities offered by an IMS operator, multimedia application developers face a business challenge to create applications which can run on a variety of user devices, and possibly adapt to various network conditions and user preferences. Our goal was to identify common generic functionality related to session-level QoS signaling for advanced multimedia applications and to design a high-level application programming interface (API), which invokes this functionality. The paper describes the Dynamic Service Adaptation model on which the proposed API is based, the API specification, and its implementation in Java. A prototype multimedia application featuring a 3D networked game provides an example of using the API.

1. Introduction

One of the driving forces behind the concept of the next generation network (NGN) is the ability to provide advanced multimedia services across a wide range of user devices and over a heterogeneous network infrastructure. As a standardized NGN architecture, the 3rd Generation Partnership Project (3GPP) IP Multimedia Subsystem (IMS) [1] defines standard service capabilities and interfaces, as well as a common IP based infrastructure, which enable operators to introduce new services in an efficient and flexible way. Examples of advanced multimedia services to be offered include multimedia content streaming, interactive online games, and multi-user networked virtual worlds, similar to those available in the Internet today. By merging the telecom and the Internet worlds, IMS brings quality of service (QoS) support, advanced charging options, and the ability to seamlessly support 3rd party application providers. In order to make use of the capabilities offered by an IMS operator, multimedia application developers face a business challenge to create applications which can run on a variety of user devices, and possibly adapt to various network conditions and user preferences. Due to increased user/service requirements on network QoS, introducing "network-awareness" into media-rich networked applications could provide a critical advantage.

An idea behind network-awareness is not new: the need for end-to-end QoS signaling in networked multimedia systems has been recognized for Internet applications in the 1990s [2], and a number of mechanisms have been proposed at the time to reserve resources and dynamically adapt multimedia delivery according to network conditions [3]. The interdependence of these requirements may be addressed through the "application aspect" and the "communication aspect" [4]. Clearly, an exchange of control information, or signaling, is needed to request special treatment for traffic in the network, and to receive indications of particular conditions from the network that are of interest to the application.

Our work builds on two well-known concepts, that of end-to-end QoS signaling, and that of adaptive multimedia applications, which are now being revisited from the NGN, and more specifically, IMS perspective. While existing 3GPP specifications [5][6][7] describe procedures for QoS negotiation and signaling for multimedia applications such as

audio/video communication and multimedia messaging, the support for more advanced services, involving interactive applications with diverse and interdependent media components, is not addressed specifically and presents an open area of research. In addition to IETF Session Initiation Protocol (SIP) [8], now adopted by 3GPP, other protocols, such as End-to-End Negotiation Protocol for matching and coordinating QoS parameters [9], are being proposed. An evaluation of scenarios, involving relationships between application-level and network-level QoS signaling during session (re)negotiation and handover, can be found in [10].

This work is motivated by the challenge to identify common generic functionality related to session-level QoS signaling for advanced multimedia applications and to design a high-level application programming interface (API), which invokes this functionality. The benefits of using a signaling API are well-known from other areas in telecommunications, such as Parlay/OSA open APIs (http://www.parlay.org/en/specifications/) for call control, user interaction, mobility, etc. They include software reuse, shorter development time, and shielding the developer from the complexity and the specifics of the underlying signaling protocol(s).

The paper is organized into five sections, as follows. Section 2 gives a brief overview of the Dynamic Service Adaptation (DSA) model, on which the proposed API is based. Section 3 details the specification of DSA API. Section 4 describes the implementation of DSA API and its use in a prototype multimedia application featuring a 3D networked game, as a case study. Section 5 concludes the paper.

2. DSA model

In our previous work, we have proposed a generic Dynamic Service Adaptation (DSA) model, and its mapping to IMS [11]. In this paper, we go a step further by wrapping this functionality into an API to be (re)used for various multimedia applications. For the purposes of this paper, we will only refer to the most prominent features of the model, while the interested reader is referred to [11].

The model takes into account the heterogeneity of access options and advanced multimedia services in NGN by specifying the (sets of) parameters related to:

- end-user access network options and terminal capabilities (referring to client platform),
- user preferences (referring to client application, personal preferences, and/or (dis)ability),
- available resources and costs (related to network),
- service requirements (related to server application and/or server platform).

In this paper, we refer to "client profile" as the set of parameters encompassing the client platform and user preferences, and the "service profile" as the set of parameters describing the service requirements. DSA model focuses on the provisioning of end-to-end support for signaling QoS requirements at the session layer and not on the underlying mechanisms. It includes the entire process of session initiation, negotiation and renegotiation of QoS parameters, and service adaptation, from the point in time when an end-user accesses a service until (s)he terminates it. Negotiation/adaptation and optimization procedures are invoked throughout the service lifetime in response to significant network conditions. Each of the parties involved–the client, the server, and the network (i.e. network QoS control entities residing in the signaling path)–respond to dynamic changes in the system. The following scenarios are specifically addressed:

- *Session establishment* refers to starting the session with initial session parameters;
- *Change in service requirements* refers to addition or detraction of application components (e.g., starting or stopping video and audio streaming, adding a 3D object) which result in signaling the initiation or modification of current resource reservation, or release of network resources;
- *Change in client profile* refers to significant variations in any client profile parameter (user terminal hardware or software characteristics, access network conditions, user preferences) and are simulated by sending new client profile versions from the client side;
- *Change in resource availability* refers to variations of authorized network resources and results in signaling the new, or updated conditions to the session end-points.

While DSA model is independent of the particular network scenario, its applicability has been studied in the context of IMS. For each of the scenarios covered by the model, the exchange of signaling messages between involved parties has been specified according to the 3GPP specifications. This end-to-end signaling is preformed using SIP. Implementation of the signaling functionality has been used as the basis for DSA API.

3. DSA API

The functionality of the signaling API covers the signaling of initial service requirements (here specified as an XML-based service profile), of the initial client characteristics (here, an XML-based client profile), and of the final service configuration. The signaling occurs during the session establishment (i.e., service invocation) and the session update (i.e., service run-

time). It also includes the capability to receive notifications of events related to the session/signaling status. The effects of signaling may include service adaptation in response to varying conditions, as well as an adequate network response to client and service requirements in terms of network resources, with the overall goal of providing a better service to the user.

DSA API, shown in Figure 1, was designed with client/server architecture in mind.

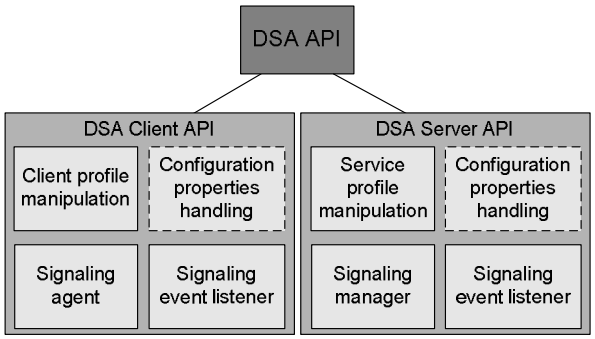

Figure 1. DSA API architecture

The client functionality is grouped together into the DSA Client API to be used by client (user) application(s), and the server functionality is grouped into the DSA Server API to be used by server application(s) hosting the services. The API is now described in more detail.

3.1. DSA Client API

The requirements for DSA API Client functionality include the ability to send client profiles and session descriptions, as well as to receive notifications of events occurring either in the network, or at the server side. Additionally, it includes the functionality for manipulating the client profile parameters. With these requirements in mind, the following three modules were identified (Figure 2): *Signaling agent*, *Signaling event listener*, and *Client profile manipulation*. The additional module, namely *Configuration properties handling*, contains the configuration file properties and methods for handling them.

Client profile manipulation module is responsible for the client profile creation and processing. *Signaling agent* module is the one responsible for handling all signaling messages in the Client API. Several methods were identified as mandatory for this signaling capability. The *establishSession()* method initiates the signaling exchange with other parties (namely, server applications) involved. It takes the XML-based client profile description as an input argument. An established session may be terminated at any time by invoking the *terminateSession()* method. The *changeInClientProfile()* method may be invoked in scenarios where a change in client profile parameters occurs, e.g., due to vertical handoff. The *changeInServiceRequirements()* method covers the scenario, in which the client side "perceives" the change in service requirements that is related to releasing network resources previously reserved for (a) particular service component(s). If a user is required to register to use the network services, then registration process may be invoked by the *register()* method.

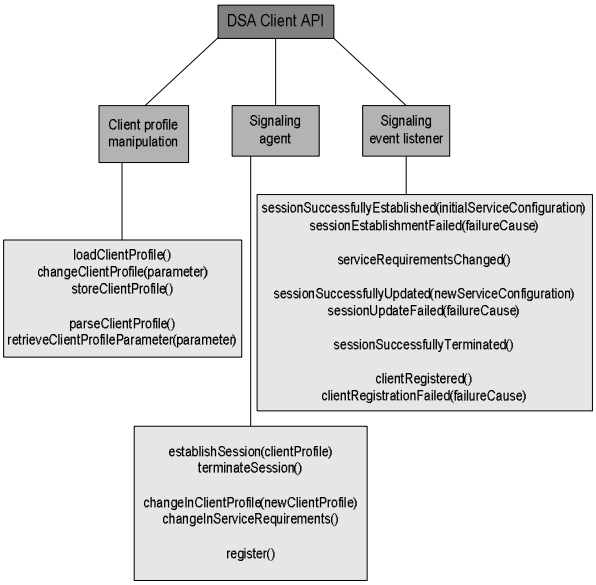

Figure 2. DSA Client API specification

Signaling event listener module is responsible for receiving events related to signaling status. These include notifications on status of session establishment (session successfully established, or, session establishment failed); session update (session successfully updated, or, session update failed); change in service requirements/signaling release of network resources (service requirements changed); and session termination (session successfully terminated), regardless of which entity/side initiated the signaling process. In addition, the client side can be notified of the registration process (client successfully registered, or, client registration failed).

3.2. DSA Server API

The purpose of DSA Server API is to provide an application developer with a means to specify initial service requirements for various versions of the service (e.g., for different terminals), as well as to specify the service behavior in terms of changes in service requirements triggered by user demands and network

conditions. The service requirements are carried by signaling messages in the form of the service profile.

The requirements for the Server API include the ability to send, receive, and process signaling messages related to service profiles and final service configurations, as well as to receive and properly interpret indications from other parties. Such indications serve to notify the server application of variable (network) conditions that are of interest to it. Specification of DSA Server API is shown in Figure 3.

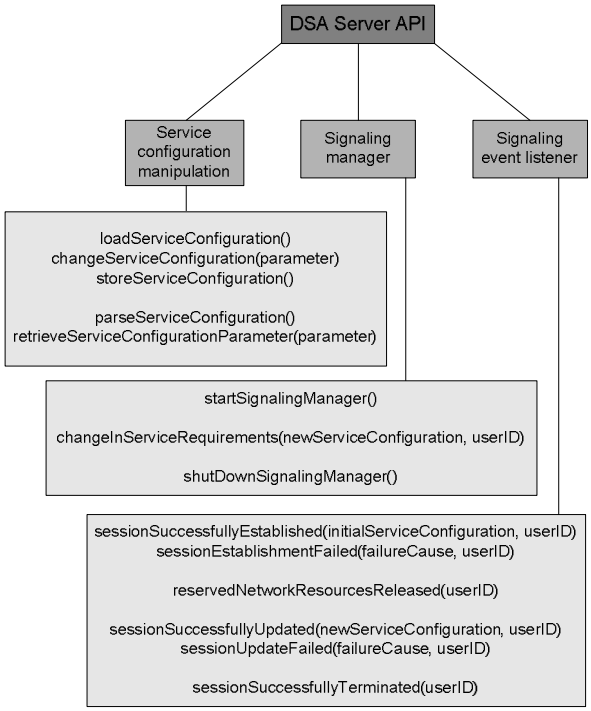

Figure 3. DSA Server API specification

Signaling manager module handles multiple clients (i.e., *Signaling agents*). The module receives, processes, and sends appropriate signaling messages, depending on the particular scenario. It comprises methods for starting and shutdown, as well as the method for handling a change in service requirements "perceived" by the server side. The change may result in invoking a different service configuration. *Service configuration manipulation* module is responsible for processing service configurations.

Signaling event listener module is, analogously to one on the client side, responsible for receiving events associated to the session status. *Session successfully established* and *Session establishment failed* events refer to session establishment between the client and the server, before the initial service retrieval. *Session successfully terminated* event relates to session termination. *Session successfully updated* and S*ession update failed* events arise in response to, e.g., variable network conditions. In addition, there is an event indicating release of reserved network resources.

Configuration properties handling module handles the configuration file properties.

3.3. The use of DSA API

The scope of the API, as related to DSA model, is illustrated by Figure 4. The client runs on a User Equipment (UE), and the server application runs on an End point Application Server of the application provider. In case of IMS, the application provider may be either the IMS operator, or the 3rd party application provider. Both the client and the server application are extended with signaling capability by using the DSA Client API and the DSA Server API, respectively. The client and service profiles matching and QoS optimization processes take place in the IMS operator domain (e.g., on a dedicated Application Server). They take client and service profiles as inputs, as well as network resources availability and cost, and have a feasible service configuration as output. The negotiated session parameters are passed over to the network QoS control entities, which control the logical data channel for delivering multimedia data. By using the same vertical interface, data from the network is monitored and may thus be used by the decision process.

4. Implementation and results

DSA API has been implemented in Java programming language and tested by using a prototype multimedia application in a laboratory testbed. The virtual channel was emulated by the NISTNet network emulator.

4.1. Software implementation

The DSA API Reference Implementation is based on the NIST-SIP API [12] and the 3GPP specifications [5] [6] [7]. Parser for the client and server profiles is based on the Simple API for XML (SAX) parser [13].

The reference implementation of DSA Client API includes two Java packages:

- *hr.fer.tel.nims.dsa.client*, and
- *hr.fer.tel.nims.dsa.client.clientprofilehandler*,

while the implementation of DSA Server API includes:

- *hr.fer.tel.nims.dsa.server*,
- *hr.fer.tel.nims.dsa.server.eventlistener*, and
- *hr.fer.tel.nims.dsa.profilemanipulation.*

The API is available for download from: *http://ve.tel.fer.hr*

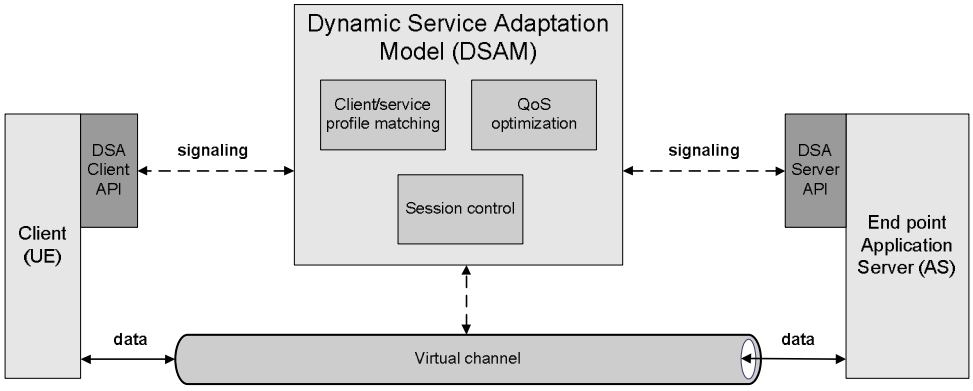

Figure 4. Scope of DSA API as related to DSA model

4.2. Prototype application using DSA API

In order to show the applicability of the API, a prototype Web-based multimedia application has been designed and developed. The application is a 3D virtual world featuring a Treasure Hunt-like game, which has been extended with signaling capability using DSA API. Several views of the virtual world are shown in Figure 5. The plot of the game is as follows: the players are potential heirs of an excentric dead millionaire, who hid a key to his treasure somewhere on his tropical island. Each player has to find it first in order to earn the inheritance. To achieve that, the player follows a series of audio and/or video clues. The 3D world objects are developed using the Virtual Reality Modeling Language (VRML). Audio and video clues are streamed through the network and displayed using Java Media Framework (JMF) API based players. The service content is placed onto an Apache Tomcat Web server.

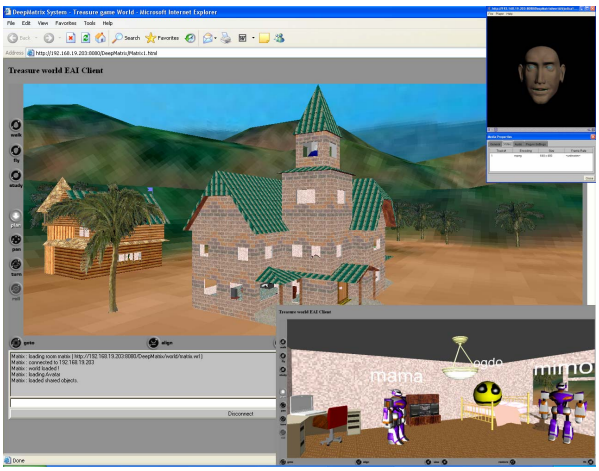

Figure 5. View of the 3D virtual world

When a player joins the game, the main 3D scene is retrieved from the server. Each player is represented by a virtual 3D character, or avatar, visible to other players. While exploring the world, players come across the clues, which they activate by selecting particular objects. All this media-rich content (virtual 3D scenes, avatars, real-time media) contributes to the service complexity in terms of transport layer QoS requirements. The game has been developed in three service versions, each described by a service profile:

(v1) high quality audio and video streaming,
(v2) low quality audio and video streaming,
(v3) low quality audio streaming only (no video).

The version specifies which service components, and in what format, may be delivered to a user. For each version, the content has been prepared in advance using different high- and low-quality codecs. Several types of clients were foreseen, assuming diverse terminal equipment/access networks/user preferences. Client profile format is based on the Session Description Protocol next generation [14].

The prototype application with DSA API was tested in four scenarios of DSA model. *Session establishment* is invoked by a player. It includes registering a user-terminal with the network, negotiating initial service configuration, and service (the main scene) retrieval based on the configuration. Network QoS control entity (represented in DSA model by the *Session control*, Fig. 4) is being signaled to allocate and release network resources used for scene download, as needed. *Change in service requirements* is caused by a user initiating, e.g., an audio/video streaming. The server updates the service requirements, and a new service configuration is negotiated based on information related to requested streams. The matching and QoS optimization calculates optimal audio and video codec combination. Reservation of network resources is set up on the virtual channel. *Change in client profile* is caused by, e.g., an increase in the player's access network bandwidth, which results in renegotiation, followed by the matching and the optimization. For instance, if

media streaming is in progress, and if a variation of the bandwidth increase is significant, an automatic change of the streaming quality will occur based on the updated service configuration. *Change in network resource availability* is detected by the *Session control*, which monitors the *Virtual channel* (Fig. 4). This again invokes renegotiation and the matching/optimization, which result in a new service configuration. Automatic changes of service parameters (e.g. audio codec due to a decrease of authorized resources) occur at both the client and server sides based on the negotiated profile.

In each of the scenarios, it has been demonstrated that, based on signaling, an adapted version of the service is delivered to the user. Thus, the required functionality of the API was achieved. Our preliminary measurements (where a 100 Mbit/s LAN emulated an access network) of the time needed for session establishment obtained a result of up to 8 seconds, while the renegotiation time was from less than 1 second to up to 4 seconds, depending on a scenario. These being only initial results, our current work focuses on developing more elaborate scenarios and studying the critical performance issues.

5. Conclusions

With an appropriate support within the network related to the matching and QoS optimization, DSA API offers various benefits to application developers. It may simplify the development of advanced multimedia applications with network-aware adaptation and thus shorten the development time. The proposed approach differs from current approaches, where applications either (1) do not use signaling at all (e.g., most Internet applications), or, (2) use a standard network and/or service-specific signaling protocol (e.g., SIP) but have the signaling capability built into, and thus inseparable from the application and/or the platform. While the second approach enables the exchange of control information, it is practically impossible to reuse this functionality due to tight coupling with the application. Moreover, this approach requires that the application developer knows the signaling protocol specifics very well, and is capable of building a "signaling agent" into each new application from scratch. Finally, once built into the application, signaling support can not be upgraded to, e.g., a newer release of the signaling protocol without significant effort in rebuilding the whole application. The proposed approach solves these problems, however, further research is needed to properly address performance and scalability issues.

6. Acknowledgments

This work was carried out within the research projects "Content Delivery and Mobility of Users and Services in New Generation Networks," supported by the Ministry of Science, Education and Sports of the Republic of Croatia, and "Networked Virtual Reality in IMS," supported by Ericsson Nikola Tesla, Croatia. The authors would also like to acknowledge the students Ivan Piskovic and Mirko Suznjevic for their contribution in developing the prototype application during eNTERFACE'06, the SIMILAR NoE Summer Workshop on Multimodal Interfaces.

7. References

[1] Camarillo, G., and M.A. Garcia-Martin, *The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds*, John Wiley & Sons, West Sussex, 2004.

[2] K. Nahrstedt, "Challenges of Providing End-to-End QoS Guarantees in Networked Multimedia Systems," *ACM Comp. Surv. Jour.*, Vol. 27, No. 4, December 1995, pp. 613-616.

[3] X. Wang, and H. Schulzrinne, "Comparison of Adaptive Internet Multimedia Applications," *IEICE Trans. on Commun.*, Vol. E82-B, No. 6, June 1999, pp. 806-818.

[4] M. Matijasevic, D. Gracanin, K.P. Valavanis, and I. Lovrek, "A Framework for Multi-user Distributed Virtual Environments," *IEEE Trans. SMC*, Part B, Vol. 32, No. 4, August 2002, pp. 416–429.

[5] —, 3GPP TS 23.228: IP Multimedia Subsystem (IMS); Stage 2, Release 7, June 2005.

[6] —, 3GPP TS 23.218: IP Multimedia (IM) session handling; IM call model; Stage 2, Release 7, June 2006.

[7] —, 3GPP TS 24.228: IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3, Release 7, December 2005.

[8] J. Rosenberg et al., "SIP: Session Initiation Protocol," IETF RFC 3261, June 2002.

[9] T. Guenkova-Luy, A.J. Kassler, and D. Mandato, "End-to-End Quality-of-Service Coordination for Mobile Multimedia Applications," *IEEE JSAC*, Vol. 22, No. 5, June 2004, pp. 889-903.

[10] R. Prior, S. Sargento, D. Gomes, and R.L. Aguiar, "Heterogeneous Signaling Framework for End-to-end QoS Support in Next Generation Networks," Proc. of 38th HICSS, CD-ROM Proceedings, Big Island, HI, USA, January 2005.

[11] L. Skorin-Kapov, and M. Matijasevic, "End-to-end QoS Signaling for Future Multimedia Services in the NGN," Proc. of New2AN 2006, *LNCS* vol. 4003, 2006, pp. 408-419.

[12] NIST-SIP [Online: http://snad.ncsl.nist.gov/proj/iptel/]

[13] SAX [Online: http://www.saxproject.org/sax1-roadmap.html]

[14] D. Kutscher, J. Ott, and C. Bormann, "Session Description and Capability Negotiation," IETF Internet Draft, draft-ietf-mmusic-sdpng-08.txt, February 2005.