# An Experimental Evaluation of a QoS Signaling API for Network-aware Multimedia Applications in NGN

Ognjen Dobrijevic
University of Zagreb, FER
Unska 3, HR-10000 Zagreb, Croatia
+385 1 6129 755
ognjen.dobrijevic@fer.hr

Maja Matijasevic
University of Zagreb, FER
Unska 3, HR-10000 Zagreb, Croatia
+385 1 6129 757
maja.matijasevic@fer.hr

## ABSTRACT
Heterogeneous environment of the Next Generation Network, involving diverse user devices and access options, as well as versatile network infrastructure, calls for closer cooperation between network entities and multimedia applications in achieving end-to-end Quality of Service (QoS). This typically involves a signaling mechanism, as a means for an application to request a certain QoS from the network, and in return to receive notifications related to various network conditions of interest, hence possibly adapting to them. This so-called "network-aware" approach served as the basis for designing the Dynamic Service Adaptation (DSA) Model for signaling, negotiation, and adaptation of QoS parameters for multimedia services. Based on DSA Model, the DSA API was developed. It encompasses generic functionality related to session-level QoS signaling. The paper presents designed signaling scenarios and an experimental evaluation of DSA API in a laboratory environment, using a prototype network-aware multimedia application that features a virtual 3D Web-based environment.

## Categories and Subject Descriptors
C.2.2 [**Computer-Communication Networks**]: Network Protocols - *Applications*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems - *Client/server*; C.4 [**Performance of Systems**]: *Performance attributes*.

## General Terms
Measurement, Performance, Languages.

## Keywords
Multimedia, QoS, signaling, negotiation, Application Programming Interface (API), IP Multimedia Subsystem (IMS).

## 1. INTRODUCTION
The concept of the Next Generation Network (NGN) assumes an environment consisting of diverse access options and user devices, as well as versatile network infrastructure [10]. End-user and service requirements on network Quality of Service (QoS) increase in such an environment, calling for closer cooperation between network entities and applications to achieve desirable QoS level. Targeted cooperation comprises a signaling mechanism. It enables applications to request a special treatment from the network entities to be applied to the associated traffic flows, as well as to receive information on particular network conditions, possibly adapting to them. Such applications are referred to as "network-aware" applications [1].

The "network-aware" approach served as the basis for designing the Dynamic Service Adaptation Model (DSAM) [7]. DSAM is a generic model that provides signaling, negotiation, and adaptation of QoS parameters for multimedia services. It is built upon the concept of end-to-end QoS signaling for adaptive IP multimedia applications, which are currently being revisited from the NGN perspective. The QoS signaling and negotiation functionality of DSAM is "wrapped" into an Application Programming Interface (API), described in our previous work [2]. This API, named Dynamic Service Adaptation API (DSA API), is used to extend applications with the QoS signaling functionality at the session layer. Such an API relieves application developers of the need to know signaling protocol details and simplifies the reuse of the implemented functionality, possibly leading to shorter development time.

This work presents an experimental evaluation of DSA API in a laboratory environment, using a prototype multimedia application. The prototype application is implemented using the API, thus incorporating the ability to receive information that can be used to adapt the service to variable conditions of the environment. Performed evaluation is based on carrying out measurements in a laboratory testbed and analyzing results with proposed performance parameters.

The paper is organized into six sections, as follows. Sections 2 and 3 provide a brief overview of DSAM and DSA API. Section 4 describes the prototype network-aware application and its behavior. DSAM laboratory implementation serves as the environment for evaluating the API. Section 5 presents designed signaling scenarios, measurements procedures, and analysis of evaluation results. Section 6 summarizes and concludes the paper.

## 2. DSA MODEL
DSAM is a model for negotiation and dynamic adaptation of QoS parameters for advanced multimedia services. We briefly describe the model in this section, and the interested reader is referred to [7] for more details.


MoMM2008, November 24–26, 2008, Linz, Austria.

The model deals with end-to-end (E2E) signaling of QoS requirements at the session level. It includes the overall process of QoS negotiation during session establishment when a user accesses a multimedia service, as well as of renegotiation of QoS parameters and service adaptation in the course of an active session. DSAM considers the heterogeneity of the NGN by addressing following (groups of) parameters, which are used during QoS (re)negotiation:

- availability of network resources and their costs (relating to the communication network),
- service requirements (relating to the multimedia service and server platform hosting it), and,
- client characteristics, which include end-user terminal features and access network options, as well as user preferences (relating to the user's personal preferences and client platform).

A set of parameters that relate to personal preferences and client platform are referred to as "client profile", while the "service profile" includes service-related parameters and requirements.

Renegotiation and adaptation procedures are initiated throughout the session duration, in response to dynamic changes in addressed model parameters. QoS renegotiation can be invoked by each of the sides engaged: the client, the server, and the communication network (i.e. network QoS control entities). The model particularly focuses on the following scenarios:

- *Session establishment* involves negotiating initial session parameters and setting up the session;
- *Change in service requirements* relates to addition or subtraction of service components (e.g., starting media streaming) which result in signaling and negotiation process, for instance, to modify current resource reservation;
- *Change in client profile* refers to a substantial variation in any client profile parameter (e.g., replacement of the user terminal) and results in sending a new client profile from the client side; and
- *Change in resource availability* includes an alteration of authorized network resources and results in signaling the new conditions by the network QoS control entities.

As DSAM is independent of the specific network architecture or scenario, its applicability has been demonstrated by mapping it onto a standardized NGN architecture, the 3rd Generation Partnership Project (3GPP) IP Multimedia Subsystem (IMS) [11]. IMS specifies standard interfaces and service capabilities, along with a common IP based infrastructure, which provide easier introduction and utilization of new (multimedia) services. The exchange of signaling messages between involved DSAM entities has been designed in accordance with the 3GPP specifications [11][12][13], namely, for each of the scenarios addressed by the model. Developed signaling flows have served as the basis for signaling functionality of DSA API. It should be noted that a similar idea is currently being explored in Java Community Process for IMS in general, under JSR 281 IMS Services API [14]. This IMS API is intended to provide a high-level API for accessing IMS services, by hiding IMS technology details and exposing service-level support for easier development of applications.

# 3. DSA API

The main purpose of DSA API is to extend applications of the client (representing an end-user) and the server (hosting a multimedia service) with QoS signaling functionality at the session layer. DSA API involves signaling the initial service requirements (service profile), the initial client characteristics (client profile), and the final service configuration required to establish the session. Figure 1 shows the communication environment corresponding to the NGN, which influenced the design of this API. It also illustrates the scope of DSA API as related to DSAM. Applications incorporating the API are capable of exchanging messages with the network control entities, thus complying with the network-aware approach.

Using designed message exchange between the signaling components of the API, the communication end-points are able to establish or terminate session, signal changes that dynamically arise in the environment after the session has been established, and (re)negotiate service parameters (along with QoS configurations). The signaling protocol messages are used to convey parameters for authorization and reservation of network resources, essential for an adequate network response to client and service requirements.

DSA API has been organized in two parts: the client-part, known as DSA Client API, and the server-part, known as DSA Server API. DSA Client API is to be used by the client application/process when communicating with DSAM signaling entities in the network and the server side, while DSA Server API is to be utilized by the server application/process when communicating with DSAM signaling entities and the client side.

DSA API has been implemented in Java programming language. Client and service profiles have been specified using the Extensible Markup Language (XML). Parser for these profiles was implemented using the Simple API for XML (SAX) parser [17]. As mentioned earlier, the signaling flows related to scenarios addressed by DSAM are based on 3GPP IMS specifications and use the IETF's Session Initiation Protocol (SIP) [6] for performing E2E signaling. DSA API Reference Implementation has been realized with the NIST-SIP API [16]. The Reference Implementation includes approximately 5000 lines of Java source code. (This also gives a rough idea of the benefit for the application developer from using the API.) Some details of DSA API implementation have been presented in [2]. The API is made available under GPL-like license at http://ve.tel.fer.hr.

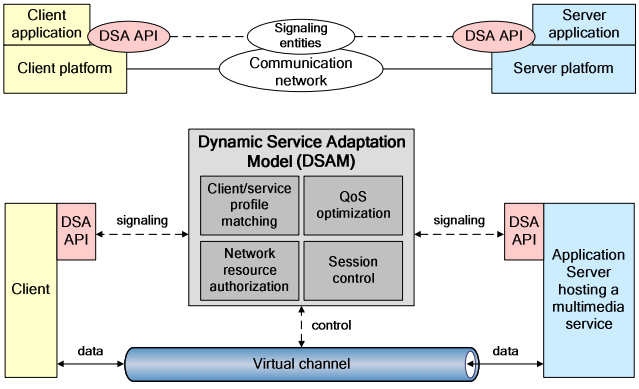


**Figure 1. The scope for the application of DSA API**

Signaling functionality for DSAM scenarios can be invoked using the following methods of the API:

- *establishSession*,
- *signalNewServiceRequirements*,
- *signalNewClientCapabilities*,
- *terminateSession*, and,
- *signalNewResourceAvailability*.

The first four methods may be invoked by the application (the client and/or the server), and the last one may be invoked by the network, i.e. the network signaling entity that has received resources-related information from the lower level QoS enforcement and monitoring network entities.

# 4. DSAM LABORATORY PROTOTYPE

DSAM laboratory implementation (Figure 2) logically consists of several functional components. Client application runs within the User Equipment (UE), i.e. a device by means of which the user accesses the multimedia service. Server application runs on an Application Server (AS) hosting the prototype application (NVR AS). Client and Server applications incorporate DSA API functionality. Negotiation which takes place between UE and NVR AS uses signaling messages to convey client and service profiles. The profiles are being transferred to another DSAM-specific AS, which hosts functions for matching the profiles and optimization of QoS parameters (QMOP AS) [8]. These functions serve to determine feasible service configuration, which is to be delivered to the user, with optimized amount of required network resources. In the context of IMS, QMOP AS thus acts as a QoS service enabler. Feasible service configuration is being forwarded through network entities (in IMS, P-CSCF and S-CSCF) to elements responsible for authorization and reservation of network resources (in IMS, PDP and PEP). They control an emulated data channel (Virtual channel) used for delivering multimedia content with applied QoS. Monitoring data flows provides feedback information to be used for QoS adaptation (e.g., for invoking the signaling denoting a significant change in resource availability).

## 4.1 Prototype multimedia application

The prototype network-aware multimedia application, named *Inheritance Chase* (Figure 3), is placed on NVR AS. *Inheritance Chase* is a multi-user virtual 3D environment on the Web. In this interactive game, players follow a set of streaming audio/video "clues", which are to lead the winner to a hidden last will needed to receive the vast inheritance.

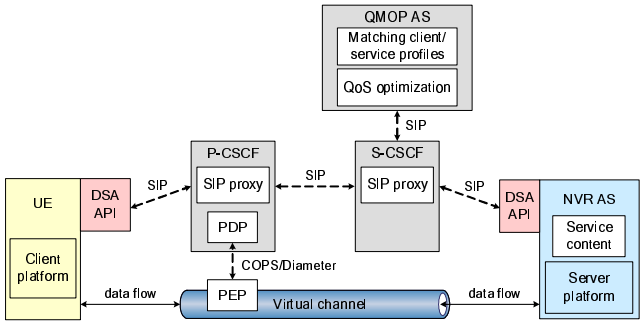


**Figure 2. DSAM laboratory implementation**

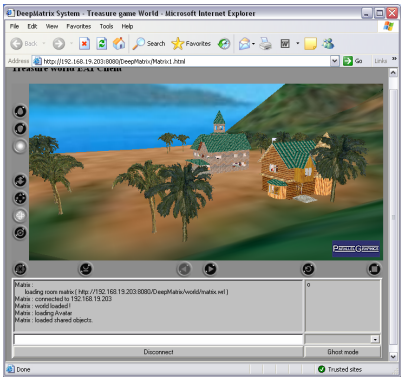

**Figure 3. Prototype multimedia application**

This multimedia application has been developed in three different service configurations, each differing in number of media components and their quality:

(1) a configuration that supports "high-quality" audio-video streaming,

(2) a configuration that supports "low-quality" audio-video streaming, and

(3) a configuration that only supports audio streaming (and no video).

The particular service configuration to be used is determined by the negotiation procedure specified by DSAM.

# 5. EVALUATION OF DSA API

We propose six parameters for evaluation of DSA API. To the best of our knowledge, there is no standardized evaluation approach to assess the performance of a SIP signaling API and its impact on the effectiveness of application execution. We used some similar research efforts [3][4][5][9] as guidelines. The following parameters have been defined, for the three signaling scenarios addressed by DSAM:

- Response time – the time interval required for all the messages to be exchanged in a particular signaling scenario,
- Signaling throughput – the overall amount of signaling information (including sizes of: IP protocol header; header introduced by the TCP transport protocol; SIP protocol header; and the SIP message body) that is sent and received by DSA API components within a given signaling scenario,
- Signaling header size – the average size of SIP protocol header in the signaling messages (Figure 4) for a particular signaling scenario,
- CPU load – the average amount of CPU load over time that is induced by the execution of DSA API components,
- Memory consumption – the average amount of memory consumption that is induced by the execution of DSA API components, and
- Initialization time – the amount of time required to initiate DSA API components.

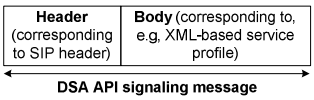


**Figure 4. Illustration of signaling message header**

The signaling scenarios are described next.

## 5.1 Signaling scenarios for measurements

All measurements are performed while an end-user is accessing the prototype multimedia application and navigating through the virtual environment. After the application is accessed and service content is retrieved, changes in addressed DSAM parameters are emulated. This triggers adequate signaling and renegotiation procedures, showing the capability of the application to participate in the negotiation process and possibly adapt to the changes.

Each measurement includes the following "evaluation scenario". It starts with the end-user sending the request for establishing session, with this request specifying the user's ability to support "high-quality" audio-video streaming (in that particular evaluation scenario). During the negotiation process, the parameters for reservation of network resources are conveyed. Session establishment results in the initial retrieval of the service content (loading the 3D scene shown in Figure 3). The Unified Modeling Language (UML) sequence diagram, shown in Figure 5, depicts the signaling flow that results in session establishment.

Using the method *establishSession* provided by DSA API, UE is capable of initiating session establishment. This method must be provided with a certain client profile. In this evaluation scenario, a client profile describing an access option consisting of a personal desktop computer (PC) and the UMTS (Universal Mobile Telecommunications System) network is used. Moreover, the profile specifies user's desire to retrieve service components of "high-quality". Each session establishment, when using SIP, begins with sending a SIP INVITE request. Configuration of the laboratory testbed is set up in such a way that all the signaling messages, which are exchanged, traverse P-CSCF, S-CSCF, and QMOP AS. NVR AS signals service requirements in a SIP response to the INVITE. Service parameters that are being negotiated include, among others, service components in terms of media types, their format and desired media codec (if applicable), level of QoS, etc. Successful session establishment ends with the UE sending a SIP ACK request. Session establishment results in final service configuration produced by QMOP AS, which is used for reservation of network resources and initial virtual 3D scene (Figure 3) retrieval.

While navigating through and exploring the virtual world, the user triggers the audio/video streaming clues. This addition of new multimedia components to the session initiates signaling of new service requirements and negotiation of corresponding QoS parameters (DSAM scenario *Change in service requirements*). Successful negotiation results in audio/video streaming being started. The UML sequence diagram, shown in Figure 6, depicts the signaling flow that results in signaling new service requirements. In this evaluation scenario, the change refers to addition of audio/video streaming upon user's request. After receiving such a request, server application invokes the *signalNewServiceRequirements* method of the API to initiate renegotiation of QoS parameters. This leads to sending a SIP UPDATE request incorporating new service requirements. With the assistance of QMOP AS, session end-points agree on a new service configuration. This configuration is used for altering resource reservation and enriching the established session with media streaming (the "talking" face in Figure 7).

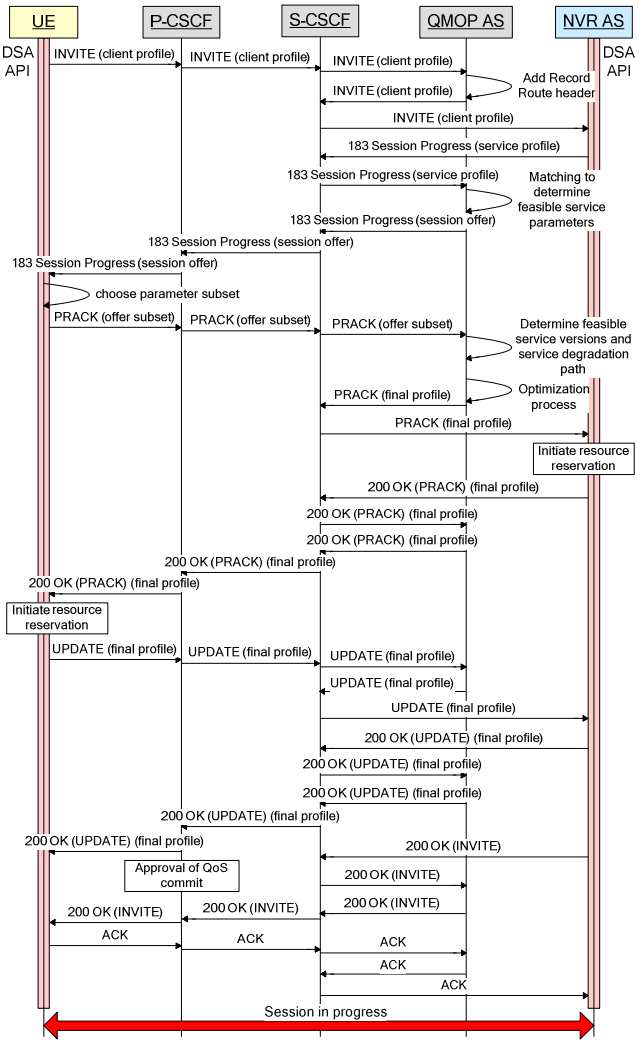


**Figure 5. Session establishment (SE)**

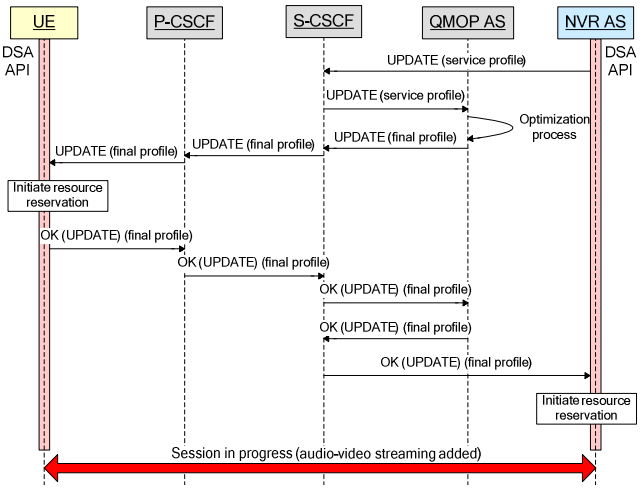


**Figure 6. Signaling new service requirements (SNSR)**

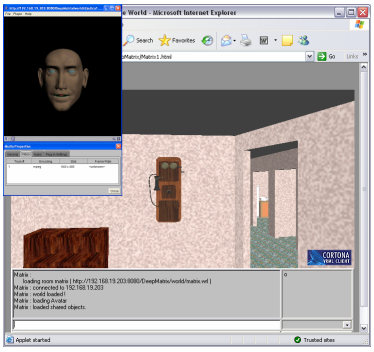

**Figure 7. Media streaming added to the service**

While streaming takes place, a decrease in authorized network resources is emulated, leading to this change being signaled to the end-points (DSAM scenario *Change in resource availability*). Outcome of this operation is the adaptation of media content delivery in the terms of a media codec change. The UML sequence diagram, shown in Figure 8, depicts the signaling flow that results in signaling new network resource availability. It is assumed that QoS related network entities (PEP and PDP) "know" which previously authorized network resources have increased or decreased. This change at the bearer is further signaled to the network control element (i.e. P-CSCF), which in return initiates sending a SIP UPDATE request that informs session end-points of a change in resource availability. Negotiation between UE and NVR AS results in a new service configuration. Server application uses the new configuration to adapt delivery of service content. This evaluation scenario involves emulating a decrease in available resources while audio/video streaming is taking place. It results in applying a "lower-quality" media codec to audio streaming (Figure 9).

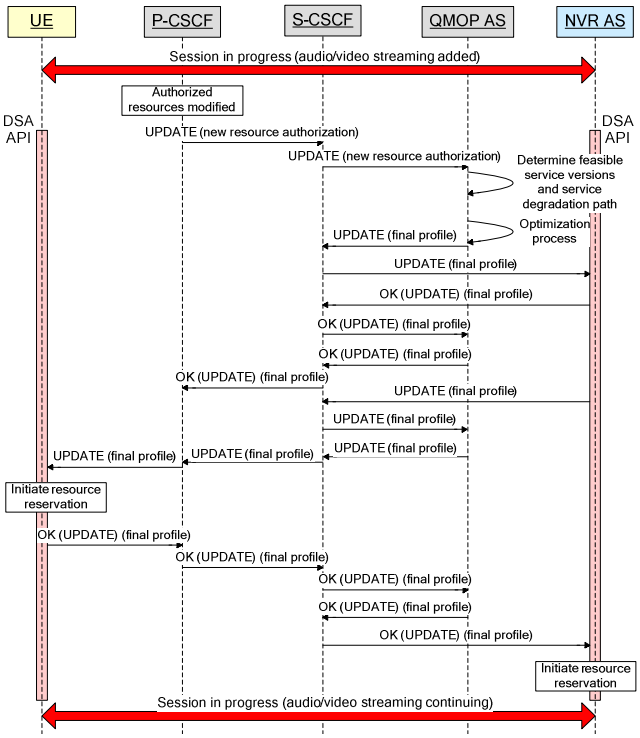


**Figure 8. Signaling new resource availability (SNRA)**

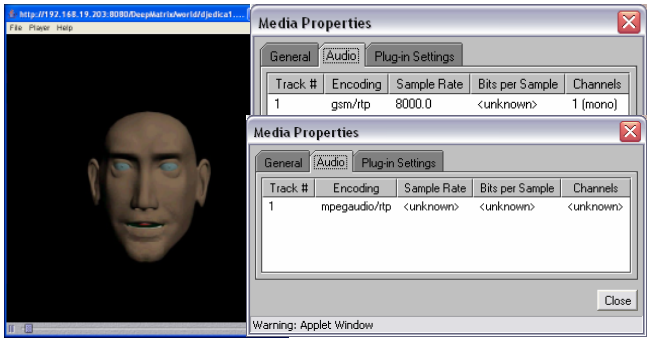


**Figure 9. Adapting media content due to a decrease in resource availability**

Afterwards, a new device configuration supporting "low-quality" audio/video streaming is signaled by the client application. This way, a change in client capabilities is simulated (DSAM scenario *Change in client profile*). The UML sequence diagram that models associated signaling flow is identical to the one for session establishment (Figure 5). In the end, the session is terminated upon user's request and network resources are released (DSAM scenario *Session termination*).

## 5.2 Experimental testbed

Evaluation measurements were conducted in a "live" experimental network shown in Figure 10. Configuration of the associated PCs hosting DSAM components is depicted in Table I.

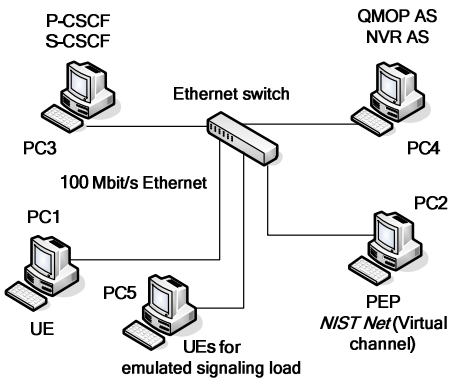


**Figure 10. Topology of the experimental testbed**

**Table I. Configuration of the testbed nodes**

| Node | Operating system | Configuration |
|---|---|---|
| PC1 | Windows XP | Pentium IV, CPU 1.6 GHz, RAM 512 MB |
| PC2 | Fedora Core | Pentium IV, CPU 1.6 GHz, RAM 512 MB |
| PC3 | Windows XP | Pentium IV, CPU 1.7 GHz, RAM 1 GB |
| PC4 | Windows XP | Pentium IV, CPU 2.4 GHz, RAM 512 MB |
| PC5 | Linux Kbuntu | Pentium IV, CPU 3.0 GHz, RAM 1 GB |

Measurements were performed for different emulated access networks (parameters described in Table II, with loss ratio not being essential and, therefore, being omitted) and for different values of signaling load imposed at the AS that uses DSA API (i.e. NVR AS). Access networks were emulated using the *NIST Net* network emulator (http://www-x.antd.nist.gov/nistnet/).

In order to produce more realistic network conditions, signaling load imposed by several users (UEs) simultaneously negotiating QoS with the server (NVR AS) was introduced. The load was generated by the UEs (PC5 in Figure 10) that were based on the *SIPp* traffic generator (http://sipp.sourceforge.net/). *SIPp* is able to establish and terminate SIP sessions, and was adjusted to make calls, i.e. session attempts, based on the *Session establishment* signaling flow. Imposed signaling load is expressed in *call/s*, and, e.g., 5 *call/s* equals to 5 new session attempts each second. This process of making calls with *SIPp* is repeated while the user represented with the UE on PC1 (Figure 10) executes evaluation scenario described in previous subsection.

For each combination of access network type and imposed signaling load, we repeated the evaluation scenario for 15 times and measured values of defined evaluation parameters. We will focus on the *Session establishment* (SE), the *Signaling new service requirements* (SNSR), and the *Signaling new resource availability* (SNRA) signaling scenarios, because the *Signaling new client capabilities* signaling flow is almost identical to *Session establishment*. As the limited transmission characteristics of the access network often constitute a performance bottleneck, the goal was also to analyze their impact onto DSA API performance, as well as the impact of increasing number of simultaneous users.

**Table II. Emulated access networks**

| Type | Bandwidth | Delay [ms] |
|---|---|---|
| GPRS (General Packet Radio Service) | 170 kbit/s | 45 |
| UMTS (Universal Mobile Telecom. System) | 384 kbit/s | 45 |
| WLAN (Wireless Local Area Network) | 1500 kbit/s | 10 |
| LAN (Local Area Network) | 100 Mbit/s | 10 |

## 5.3 Evaluation results

Average response time for each of the signaling scenarios is depicted in relation to emulated access networks and imposed signaling load by Figure 11, Figure 12, and Figure 13. Table III shows the interval start and end time, as well as an overall size of the profiles and service configurations ("session description data") being conveyed in a particular signaling scenario. It should be noted that the amount of session data transferred in various signaling scenarios differs significantly, with the maximum of about 37 kilobytes being transferred during *Session establishment*.

Response time was measured at the "side" that is responsible for invoking message exchange in a specific signaling scenario. For SNSR, response time was measured at NVR AS, for SNRA at P-CSCF, and for SE at UE.

**Table III. Session description data and time interval per signaling scenario**

| Signaling scenario [Session data] | Interval start time | Interval end time |
|---|---|---|
| Session establishment [37,257 byte] | Sending the INVITE by UE | Sending the ACK by UE |
| Signaling new service requirements [3,472 byte] | Sending the UPDATE by NVR AS | Receipt of the 200 OK by NVR AS |
| Signaling new resource availability [3,250 byte] | Sending the first UPDATE by P-CSCF | Receipt of the last 200 OK by P-CSCF |

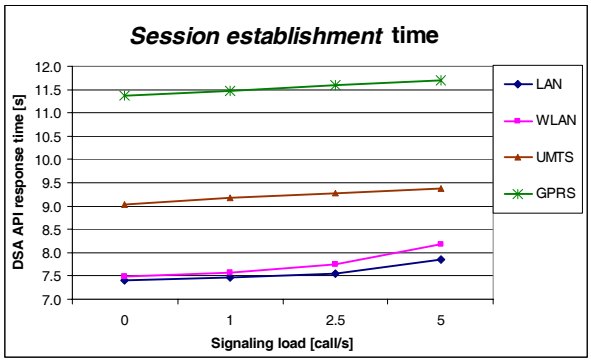


**Figure 11. Average SE response time**


**Figure 12. Average SNSR response time**

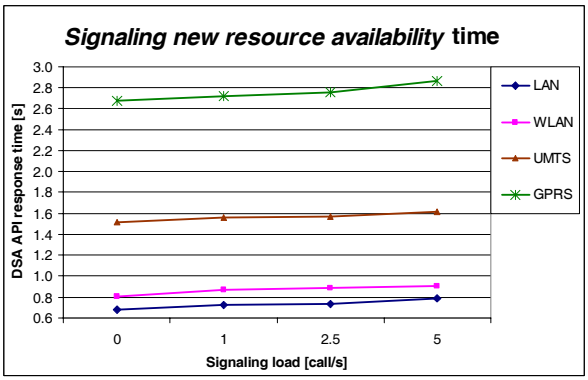


**Figure 13. Average SNRA response time**

Results in Figures 11 - 13 show that response times of *Session establishment* are about 4 to 11 times higher than those of *Signaling new service requirements* and *Signaling new resource availability*. This was expected, as they include exchange of a higher number of signaling messages and a larger amount of signaling information in general. Furthermore, when comparing each of the scenarios across various access network types, it is clear that response time is shorter for LAN and WLAN comparing to UMTS and GPRS. This was expected as well, due to a lower delay and a higher bandwidth. When taking into consideration increase of imposed signaling load, it can be noticed that, in general, response time grows slower for UMTS and GPRS than for LAN and WLAN.

If comparing these results to evaluation criterion proposed in [4], stating that session establishment should last no longer than 2-5 s and renegotiation during the session no longer than 1 s, we can see that DSA API is close to complying with the criterion. Further analysis related to response time could be based upon the evaluation metrics defined in [5].

Average signaling throughput is given separately for DSA client-side API (Figure 14, only *Session establishment* is shown) and DSA server-side API (Figure 15, Figure 16, and Figure 17).

If compared one to another in relation to *Session establishment*, average signaling throughput of DSA server-side API is about 1.2 times higher than that of DSA client-side API. This is the result of about 37 kilobytes of session description data and a similar overall amount of signaling information being traversed through the server-side in a proportionally shorter period of time. Similarly to response time, signaling throughput rises as the bandwidth and the delay of access network improve (from GPRS to LAN). It is important to note that response times and signaling throughputs directly depend on our particular implementation performance, thus leaving room for improvement. Signaling throughput generally decreases when imposed signaling load increases (e.g., Figure 14). Taking into consideration this increase, it can be noticed that, in general, throughput decreases slower for UMTS and GPRS than for LAN and WLAN.

Average signaling header size for each of the signaling scenarios is depicted by Table IV. Values in the table represent only the SIP header size. They are related to the fields that form SIP header. Utilization of most fields in DSAM laboratory implementation is

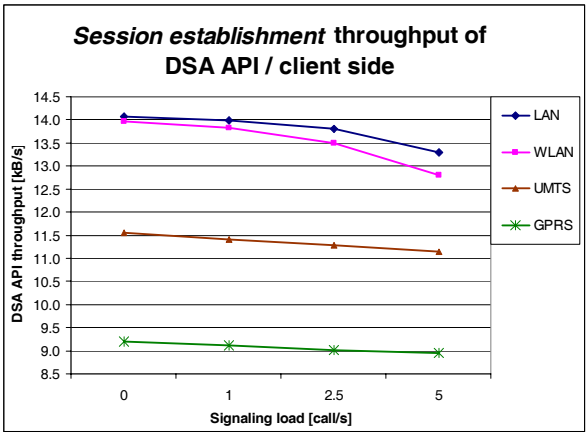


**Figure 14. Average SE signaling throughput (DSA Client API)**

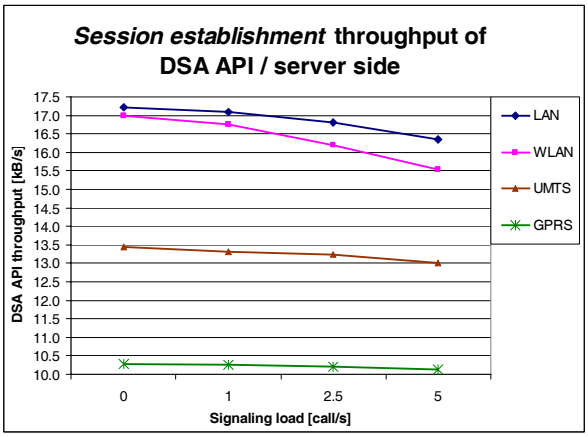


**Figure 15. Average SE signaling throughput (DSA Server API)**

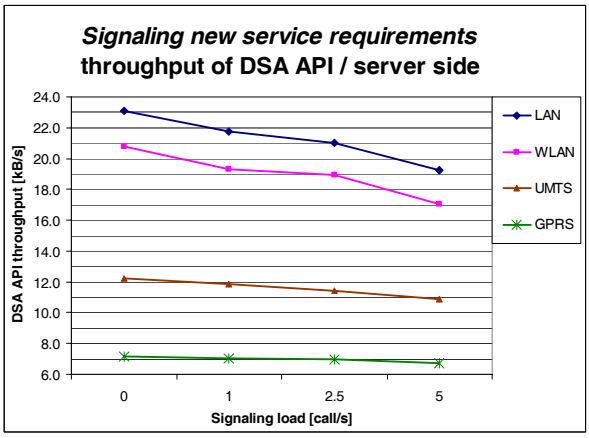


**Figure 16. Average SNSR signaling throughput (DSA Server API)**

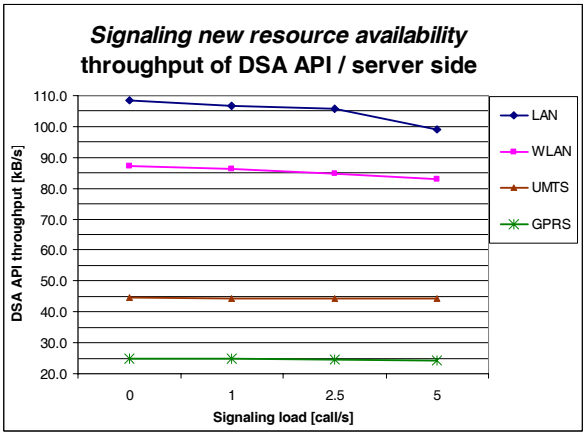


**Figure 17. Average SNRA signaling throughput (DSA Server API)**

a result of how the SIP protocol is used in IMS (and not a result of DSAM functionality).

**Table IV. Average signaling header size**

| Signaling scenario | Average header size [byte] |
|---|---|
| Session establishment | 750.76 |
| Signaling new service requirements | 769.76 |
| Signaling new resource availability | 789.76 |

Results related to CPU load are highlighted with average CPU load for *Session establishment*, given separately for DSA client-side API (Figure 18) and DSA server-side API (Figure 19).

Average CPU load of DSA client-side API increases as the access network characteristics improve in terms of bandwidth increase and delay decrease. This can be explained by the same amount of signaling information traversing the client-side in a shorter period of time (when changing from GPRS to LAN, Figure 11), thus causing a higher "instantaneous" CPU load. On the other hand, CPU load that is induced by DSA Client API decreases as imposed signaling load increases. This is due to the equal amount of signaling information traversing the client-side in a longer period of time, as a result of an increased processing time of the server side. Average CPU load of DSA server-side API (Figure 19), on the contrary, increases almost exponentially as imposed signaling load increases, showing poor scalability. It is, however, interesting to notice that the change of access network provides no effect on CPU load of DSA Server API. CPU load of DSA Server API for *Signaling new service requirements* and *Signaling new resource availability* provided similar results.

Average memory consumption of DSA API is depicted by Table V and Table VI (the latter showing memory consumption of DSA Server API for *Session establishment*, in relation to imposed signaling load). Memory consumption of DSA Client API (Table V) is independent of the access network and imposed signaling load, and totals about 27 to 30 megabytes. Average memory consumption of DSA Server API increases with the increase of imposed signaling load, but also shows independence of the access network type.

Initialization time of DSA API is given in Table VII. It is the amount of time required to initiate DSA API components at the server-side and the client-side.

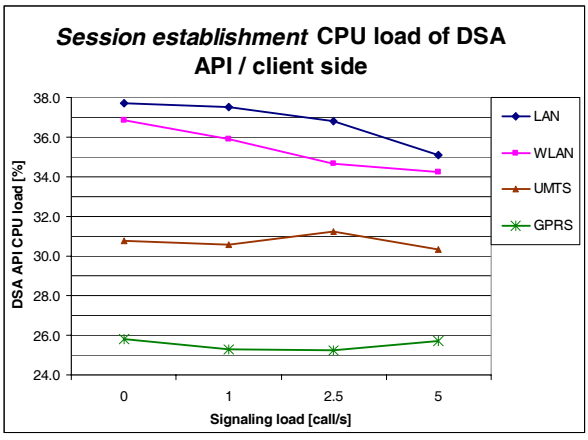


**Figure 18. Average SE CPU load (DSA Client API)**

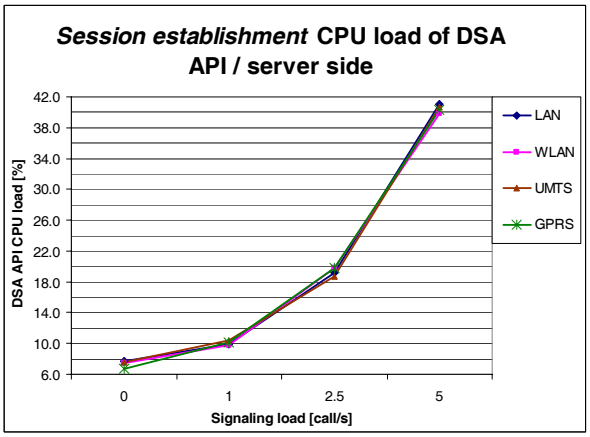


**Figure 19. Average SE CPU load (DSA Server API)**

**Table V. Average memory consumption (DSA Client API)**

| Signaling scenario | Memory consumption [Mbyte] |
|---|---|
| Session establishment | 27.43 |
| Signaling new service requirements | 30.02 |
| Signaling new resource availability | 29.79 |

**Table VI. Average SE memory consumption (DSA Server API)**

| Imposed signaling load [call/s] | Memory consumption [Mbyte] |
|---|---|
| 0 | 23.2 |
| 1 | 33.12 |
| 2.5 | 49.94 |
| 5 | 83.50 |

**Table VII. Initialization time**

| DSA Client API [ms] | DSA Server API [ms] |
|---|---|
| 399.31 | 720.26 |

# 6. SUMMARY AND CONCLUSIONS

This paper presents an experimental evaluation of DSA API, which provides an advanced QoS signaling functionality to network-aware IP multimedia applications in NGN. The evaluation is based on the six proposed parameters, namely, the Response time, the Signaling throughput, the Signaling header size, the CPU load, the Memory consumption, and the Initialization time, which could help estimate the impact of the API on the application execution. An experimental testbed using a virtual 3D environment with audio/video streaming components, called Inheritance Chase, was used to perform measurements. The proposed evaluation parameters have been related to various emulated access networks, from 2G/3G to WLAN and LAN, as

well as to different values of imposed signaling load that created multi-user environment.

Comparing the response time results to evaluation criterion proposed in [4] shows that DSA API introduces response delay that is close to comply with the criterion. Signaling load imposed by the API could prove to be significant, especially for access networks with lower transmission capacity (such as 2G, or even 3G), if we compare the throughput with bitrates of media standards such as MPEG-4 [15]. Every signaling protocol introduces overhead in the form of control header required to convey "useful" information. Although signaling header size of DSA API is far from being negligible, it is still considerably smaller than the size of message bodies (session description data) in the signaling scenarios. CPU load and memory consumption imposed by the API appear to be "significant", especially when number of simultaneous *signaling and negotiation* sessions increases. However, it should be emphasized that all evaluation results are application-specific and also depend on our particular DSAM implementation performance, leaving some room for its enhancement.

Future work could focus on performing measurements for various prototype applications and comparing results among them, as well as addressing scalability issues in more complex situations (i.e. with larger number of simultaneous users).

## 7. ACKNOWLEDGMENTS

The authors acknowledge the support of research projects "Content Delivery and Mobility of Users and Services in New Generation Networks" (036-0362027-1639), funded by the Ministry of Science, Education and Sports of the Republic of Croatia, and "Future Advanced Multimedia Service Enablers" of Ericsson Nikola Tesla, Croatia.

## 8. REFERENCES

[1] Cao, J., Zhang, D., McNeill, K. M., and Nunamaker, Jr., J. F. 2004. An Overview of Network-Aware Applications for Mobile Multimedia Delivery. In Proceedings of the 37th Annual Hawaii International Conference on System Sciences (Big Island, Hawaii, USA, January, 2004). 292-301.

[2] Dobrijevic, O., Mosmondor, M., and Matijasevic, M. 2007. Design of a QoS Signaling API for Advanced Multimedia Applications in NGN. In Proceedings of the 4th International Conference on Information Technology: New Generations (Las Vegas, Nevada, USA, April, 2007). 56-64.

[3] Fu, X., Schulzrinne, H., Tschofenig, H., Dickmann, C., and Hogrefe, D. 2006. Overhead and Performance Study of the General Internet Signaling Transport (GIST) Protocol. In Proceedings of the 25th IEEE International Conference on Computer Communications (Barcelona, Spain, April, 2006). 01-12.

[4] Guenkova-Luy, T., Kassler, A. J., and Mandato, D. 2004. End-to-End Quality-of-Service Coordination for Mobile Multimedia Applications. IEEE J. Sel. Area. Comm. 22, 5 (June 2004), 889-903.

[5] Malas, D. 2008. SIP End-to-End Performance Metrics (draft-ietf-pmol-sip-perf-metrics-01.txt). IETF Work in progress (June, 2008).

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

[7] Skorin-Kapov, L. and Matijasevic, M. 2005. Dynamic QoS Negotiation and Adaptation for Networked Virtual Reality Services. In Proceedings of the 6th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (Taormina, Italy, June, 2005). 344-351.

[8] Skorin-Kapov, L., Mosmondor, M., Dobrijevic, O., and Matijasevic, M. 2007. Application-Level QoS Negotiation and Signaling for Advanced Multimedia Services in the IMS. IEEE Commun. Mag. 45, 7 (July 2007), 108-116.

[9] Zeadally, S., Zhang, L., Zhu, Z., and Lu, J. 2004. Network application programming interfaces (APIs) performance on commodity operating systems. Inform. Software Tech. 46, 6 (May 2004), 397-402.

[10] –, 2004. General Overview of NGN. ITU-T Recommendation Y.2001.

[11] –, 2006. IP Multimedia Subsystem (IMS); Stage 2 (Release 5). 3GPP TS 23.228.

[12] –, 2006. IP Multimedia (IM) session handling; IM call model; Stage 2 (Release 6). 3GPP TS 23.218.

[13] –, 2006. Signalling flows for the IP multimedia call control based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3 (Release 5). 3GPP TS 24.228.

[14] JSR 281: IMS Services API [Online: http://www.jcp.org/en/jsr/detail?id=281]

[15] MPEG Home Page [Online: http://www.chiariglione.org/mpeg/]

[16] NIST, Project IP telephony / VoIP [Online: http://snad.ncsl.nist.gov/proj/iptel/]

[17] SAX [Online: http://www.saxproject.org/]