Implementation of Third Party Media Server Controller for IMS Networks

M. Femminella, R. Francescangeli, F. Giacinti, E. Maccherani, A. Parisi, G. Reali
University of Perugia
Dipartimento di Ingegngeria Elettronica e dell'Informazione
Via G. Duranti, 93 – 06125 Perugia

email address: name.surname@diei.unipg.it

ABSTRACT

In this paper, we describe the implementation of a media server controller that is used for implementing multimedia services by third parties of IMS networks. This implementation is done by using the well established Mobicents framework. The functionality of the server has been tested by using the Open IMS technology developed at the Fraunhofer Fokus institute. Results show the effectiveness of this implementation.

Keywords

IMS, Application Server, Media Server, JSLEE, Mobicents.

1. INTRODUCTION

The evolution of Next Generation Networks (NGN) has produced some basic principles for network and service design. The availability of high level object oriented tools has helped to define protocols and Service Oriented Architecture (SOA) based services including features such as service discovery and easy integration of 3rd party components [1][2][4]. On the whole, networks are characterized by an increased distributed intelligence able to provide self-healing, self-provisioning and self-monitoring functions.

The design principle behind these features is the transparent integration of service components through a set of rules used to control the network elements (NEs), service availability, traffic distribution, QoS levels and whatever is involved in operation [2][3].

In this work we refer to the ETSI Telecom & Internet converged Services and Protocols for Advanced Networking (TISPAN) model for Internet Multimedia Subsystem (IMS) architecture [5] [6]. In more detail, we present an implementation of a service for IMS networks, provided by a third party. This application consists of a Media Server (MS) which host a number of different media made available to NGN/IMS providers through standard interfaces.

The implementation is carried out by the widespread open source components, such as the Mobicents, which was proven to perform properly in operation [7][8], and the well known Open IMS framework developed at the Fraunhofer Fokus Institute [9].

Mobicents is a Service Logic Execution Environment (SLEE) implementation, which is a hosting environment for telecom applications, referred to as "container" in the Java terminology. It is specifically designed to fulfill the requirements of telecom services, which are typically asynchronous and impose real-time constraints (high throughput, short latency). Being composed of several layers of abstraction, which are combined in a multi-tier architecture, a SLEE simplifies the development of new value-added services by providing the non functional features needed for their execution. The resulting benefit is that the developer can focus only on aspects related to the implementation of the service logic, since the complexity of the underlying layers is masked by high-level APIs.

The Java Community, within the Java APIs for Integrated Networks (JAIN) activities, offers a set of standard frameworks and open protocol APIs for the creation of telecommunications services. Telecom providers may greatly benefit from those technologies, as the utilization of standard common interfaces simplifies new services integration and improves interoperability with multiple network, protocols, and different end-user platforms [1]. Among those activities, the JAIN SLEE (JSLEE) provides a set of specifications [10] for the implementation of a SLEE container. This kind of server is one of the most promising candidates for the deployment of application services in the new convergent telecommunications paradigm based on the IMS [11].

Service functionality has been proved in a well established test bed which can emulate typical real operation conditions. The choice of a test service based on SIP signaling is due to the widespread use of this signaling protocol. In fact, SIP is not only the de-facto standard for the signaling in the VoIP and IP multimedia communications, but it has also been included in the IMS specifications as the signaling protocol to manage data sessions (it is also used to transport other signalling protocol messages [12]), and nowadays it is increasingly used even in other scenarios (e.g. online gaming).

In more detail, we will present critical aspects regarding some key platform mechanisms, especially by focusing on the JBoss

Transaction Manager; in addition we will show relevant results about the impact of some configuration alternatives; finally we will prove the service functionality through the Core IMS platform developed at the Fraunhofer Fokus institute.

We have chosen the Mobicents JSLEE since, at the time of writing, it was the only open source implementation compliant with the JSLEE v1.0 specifications. We have already verified that this platform is suitable for production deployment, being a viable alternative to high-cost commercial SLEE solutions [7].

The manuscript is organized as follows. The next section gives an overview of the implemented framework. Section III describes the reference scenario. Section IV presents the test bed. Section V presents the relevant results. Finally, Section VI provides some final remarks.

2. BACKGROUND

2.1 JAIN SLEE

The JSLEE activity specifies a Java-based, event-oriented container for the execution of carrier grade telecom services [10]. Various Java Enterprise Edition (JEE) technologies are employed in that framework, by adapting them to the specific needs of telecom applications.

The service application logic is implemented in system components called Service Building Blocks (SBB). In operation, the JSLEE server creates a pool of SBB objects and manages them according to a well defined lifecycle. SBBs operate asynchronously by receiving, processing, and triggering events. They can be attached to data streams called Activity Contexts, by which they receive events from other system entities. Also, SBBs may be linked together by parent-child relations, in order to implement the service logic in a modular fashion.

Events are internally managed by a functional element called Event Router, which delivers each event to the appropriate SBB. For what concerns the overall performance, this component plays a critical role, since it processes all system events.

External network occurrences, such as SIP messages, are translated into internal Java events by so-called Resource Adaptors (RA). More generally, the set of implemented RAs constitutes an abstract interface layer that allows the SLEE to access external resources. In practice, a developer who wants to enable the service interoperability with a particular protocol stack can simply utilize the respective RA interface methods in his service code (e.g. methods to access a particular SIP header field). Later, he only needs to deploy the proper RA .jar file into the SLEE.

2.2 Mobicents JAIN SLEE Server

The Mobicents Communication Platform is an open source project, currently owned by Red Hat [13][14]. It includes a SLEE, a Media Server, a Presence Server, and a SIP Servlet Server. The counterparties are represented by Rhino, a commercial JSLEE by Open Cloud [15]; Convergent Service Platform, a commercial JSLEE by jNetX [16].

At the time of writing, the Mobicents JSLEE (MSLEE) is still under development. We have used the v.1.2.2 GA in our performance tests. It comes with a SIP RA which is already compliant with the new JSLEE v.1.1 specifications.

The MSLEE employs several JEE components, such as: Container Managed Persistence (CMP) fields, which enable data persistence for SBB objects; Java Database Connector (JDBC) drivers; Java Management Extensions for the environment management and monitoring; Java Naming and Directory Interface (JNDI), which offers lookup functions for service registration.

2.3 JBOSS Application Server

The MSLEE, as well as the other core servers of the Mobicents Platform, is installed on the JBoss AS [17]. JBoss acts as a hosting environment, that is to say, a container for containers. It offers capabilities such as service and SLEE configuration management, deployment, and thread pooling.

2.4 System Configuration: Critical Aspects

The MSLEE framework structure is characterized by a relevant complexity, as it integrates the JBoss AS, and the Java Virtual Machine (JVM). In order to achieve an efficient setup of the overall platform, some critical aspects of the architecture must be taken into account.

2.4.1 Java Memory Management

Being based on the Java technology, the MSLEE relies of the automatic Java Garbage Collector (GC) mechanism for memory cleaning and defragmentation [18]. The drawback is that the developer has not the full control over the GC behaviour, which may even pause the running application [19]. Such pauses are clearly critical for telecom applications. As a matter of fact, the SLEE may freeze when dealing with post-pause avalanche restarts (during pause, it accumulates unprocessed messages).

The JVM v.6 includes different GCs, in order to meet the requirements of different classes of applications [19]. In this paper, we present performance evaluated by the Parallel GC scheme only, which is the default one. Since other works suggest to use the Concurrent GC scheme, we will test it in future works, also because it requires a lot of tuning of GC parameters (see [20]).

2.4.2 Transactions

In data transmission critical environments, transactions are typically used to maintain a computer framework (i.e. a database, a file system or an application) in a known, consistent state, after system or database failures.

As regards transactions, the MSLEE relies on the JBoss AS which natively supports Java Transaction API (JTA), thus allowing any transaction manager implementation to be used. JBoss is by default configured to use the so-called “JTA compatible in-VM” transaction manager. It is very fast, but does have two limitations:

- it is unable to do automated recovery after a server crash;
- whereas it does support propagating transaction contexts with remote calls, it does not support propagating transaction contexts to other virtual machines, so all transactional work must be done in the same virtual machine as the JBoss server.

In some advanced telecom services, such as banking transactions, the provider may need a transactional behavior to ensure that all the operations made on the remote database (which may involve the transfer of subscriber personal information) remain consistent during all system operation, as well as after a system failure. Such a guarantee comes with a great performance overhead, in the form of disk writing operations, which are directly related to the number of transactions processed by the service. Therefore, it is of great importance for the provider to carefully evaluate whether to use transactions or not, in order to ensure maximum performance. In practice, the service developer can:

– choose to use a JTA compatible Transaction Manager capable of automatic detection of those operations that do not need a transaction (i.e. read only SELECT query), in order to avoid unnecessary disk writing when the shared resource is accessed;

– manually configure the shared resource to be accessed either in a transactional or non-transactional way, by using two different connection pools. In that manner, the developer is free to choose the transaction management when needed by his service. This is accomplished by simply inserting the proper JNDI reference in the code.

Also, there can be VoIP services in which the transactions are not needed at all, typically when performance requirements are most stringent. This is the case of forwarding, redirect and control services, for which there is no need to change the state of shared resources.

3. REFERENCE ARCHITECTURE

Our reference scenario is a trusted IP network of a telecom provider which offers multimedia SIP-based services by means of an AS and an MS.

The network is based on the IMS Core reference architecture [5], as depicted in Figure 1. The TISPAN specifications define the following functional core entities that implement the Call/Session Control Functions (CSCFs):

– the Serving CSCF (S-CSCF) provides management functions such as subscriber authentication and service control;

– the Proxy CSCF (P-CSCF) is an inbound/outbound SIP proxy that acts as the first IMS contact element for SIP requests;

– The Interrogating CSCF (I-CSCF) represents the edge proxy for the particular provider IMS domain. It is in charge of the subscriber registration.

The interface among those three elements is based on the SIP protocol.

The CSCF elements rely on an authentication server called Home Subscriber Server (HSS). The HSS stores the subscriber policy information. Its interface is based upon the Diameter protocol.

Also, our reference scenario includes an AS and an MS. The AS is directly involved in the call signaling flow by the S-CSCF through a SIP protocol interface. The AS and the MS communicate through Media Gateway Control Protocol (MGCP) messages.

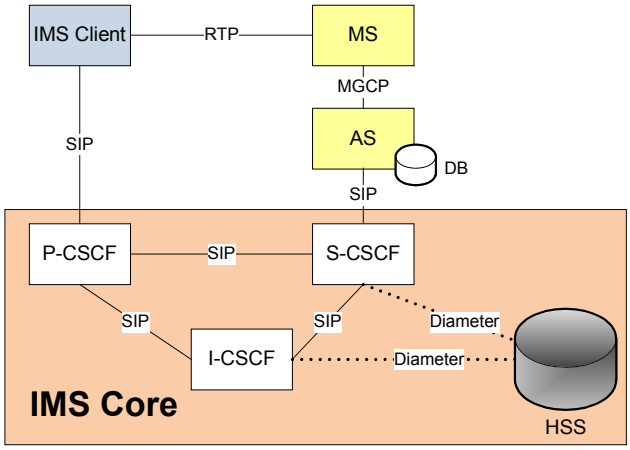

Figure 1 – Reference scenario

We consider the case of a VoIP call initiated by an IMS Client subscriber. The IMS client is located in the provider's IP home network, so that no roaming procedure takes place. The initial SIP INVITE triggers an Initial Filter Criteria (iFC) that identifies the particular caller by parsing the FROM header. The iFC instructs the S-CSCF to give to the AS the control of the call. In that way, the AS receives all the signaling messages, including the initial INVITE which contains the Session Description Protocol (SDP) parameters of the IMS Client to set up media session. In the AS we have implemented a SIP-MGCP gateway service for the media session activation The AS obtains the service profile by querying its database; then it contacts the MS which sends back to the caller a prerecorded audio announcement in an RTP session (Figure 3). As the audio message ends, the MS notifies the AS about the end of the media session. The AS, in turn, terminates the call by sending a SIP BYE message.

4. TEST BED IMPLEMENTATION

The service has been tested by using the network topology illustrated in Figure 2.

We have used the FOKUS Open IMS Client software (OpenIC), in its free Lite version, developed in the framework of the FOKUS Open IMS Playground [22]. The software was installed on a Pentium 4 class PC at 2.4GHz with 512MB RAM and the Ubuntu Linux 32bit OS.

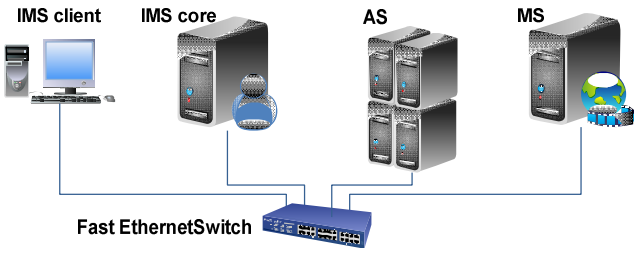

Figure 2 – Test bed

The IMS core network has been implemented by using the Open IMS Core by FOKUS [9]. It has been installed on an ArchLinux

based server, 2.6.28 64bit kernel, with an Intel Core2 Duo E6700 CPU @ 2.66GHz and 4GB RAM. Thus all the IMS Core components were deployed in the same machine. The Open IMS Core is an open source implementation of the IMS CSCFs which were developed as extensions to the SIP Express Router (SER). It also includes a lightweight HSS, called FOKUS Home Subscriber Server (FHoSS). FHoSS is Java-based and relies on a MySQL database. As stated by the developers, the Open IMS Core is not intended to become or act as a commercial product but it has the sole purpose to provide a reference implementation useful for IMS technology testing and, as in our case, services prototyping.

We populated the FHoSS MySQL database by using the bundled SQL scripts included in the OpenIMS Core package. Subsequently, we have introduced a set of iFCs to support our own Application Server through the FHoSS web configuration interface.

The AS was implemented by using a MSLEE server version 1.2.3.GA installed on top of JBoss v.4.2.3.GA and deployed on a Fujitsu-Siemens server class machine PRIMERGY TX300 S4 with dual Intel Xeon E5410 @ 2.33 GHz (i.e. 8 CPU cores) and 8GB RAM. The server OS is the Arch Linux x64 (kernel 2.6.28). The JVM used was the Sun 1.6.0_12 64-bit Server.

The Media Server was implemented by using the Mobicents Media Server (MMS). MMS was deployed on a MSLEE server installed on a server class computer equipped with an Intel Core 2 Duo E6600 @ 2.4 GHz and 4GB RAM. The server OS is the Arch Linux x64 (kernel 2.6.28). The JVM was the v.1.6.0_12 64-bit Server.

In order to configure both the MSLEE servers, we set a JVM heap of 3GB (by setting the equal *max* and *min* values as recommended in [21]). We have also chosen to use the Parallel GC by adding the following command line flags to the configuration script file of the JBoss AS: *-Xms3000m –Xmx3000m -XX:+UseTLAB*. Regarding the transaction related overheads we have disabled all the database transactions in order to improve the overall performance of the service.

All the test bed components described above have been physically connected together using a dedicated Fast Ethernet switch and Cat. 5E Ethernet cables. All the computers were configured with static IP addresses and a custom DNS server capable of resolving the IMS core network domain names was used during our tests.

We have used the Wireshark traffic analyzer tool in order to verify the system functionality. We have captured the overall network traffic by sniffing on a switch monitor port, which has been set to replicate the traffic received by all the test bed components.

5. RESULTS

In this section we describe in detail the implemented service and the relevant exchanged messages.

The signaling flow is depicted in Figure 3, whereas the IP addresses of testbed devices and the relevant messages are reported into Table 1 and Table 2, respectively.

1. The initial INVITE is sent towards the IMS network. As specified in the configured iFC, the S-CSCF checks the FROM Header to identify the caller, and forwards the request to the MSLEE AS.
2. The MSLEE receives the INVITE which includes the client SDP. Then it queries its database (Figure 1) to retrieve the service profile and sends an MGCP Create Connection command (CRCX) to the MS. In the CRCX are specified the parameters by which the AS instructs the MS how to start and manage the media session. In particular, it has to use the first available Announcement Endpoint to fulfil the request (media/trunk/Announcement/$).
3. The MS process the CRCX command. Once the Announcement Endpoint is created, the Endpoint SDP is included into the CRCX response (200 Transaction executed normally). The CRCX response also contains the Connection Identifier and the used Specific MS Endpoint. That response is sent back to the AS which, in turn, sends a 200 OK SIP Response towards the IMS network. The 200 OK contains the Endpoint SDP in order to properly setup the media session.
4. As the ACK message is received by the AS, thus completing the SIP three-way handshake, the AS sends a Notification Request command (RQNT) to the MS. In that message, various parameters are specified, such as the media flow to be streamed by the Endpoint, that in this case is the URL of the audio file to be played. Also the set of required events to be notified back (e.g. “announcement completed”), and the particular entity to which that events had to be notified (the AS), are both included.
5. The MS sends back to the AS an RQNT command after which it starts to play the announcement message, by sending RTP packets directly to the IMS Client.
6. When the announcement is completed, the MS sends a Notify (NTFY) command to the AS that includes the occured event (“announcement completed”).
7. The AS replies to the NTFY message and sends back to the MS the 200 OK response. It also sends a Delete Connection command (DLCX) including the Specific Endpoint and the connection identifier as parameters. This has been done in order to release the resources used by the endpoint-connection link. At the same time the AS sends a SIP BYE request towards the IMS Client in order to hang up the call.

With the presented media server controller we have tested the integration feasibility of the MSLEE as an AS within the IMS framework. Although being simple, our service represents a proof of concept for the implementation of more complex multimedia services that rely on the capabilities offered by the MSLEE server. As a matter of fact we have already implemented a Back-To-Back User Agent SIP signaling architecture as a single MSLEE SBB. Such an architecture grants the total control over the entire signaling flow [7] thus enabling the creation of advanced multimedia services. For instance, it may be possible to build a WEB-based click-to-call application with a buddy list that includes both IMS subscribers and non-IMS users and resources. Those users can then leverage from all the features of the SLEE based AS-MS Mobicents couple; such as conference call, Interactive Voice Response (IVR), voice mailbox and other advanced multimedia services. The resulting benefits would embrace the flexibility and convergence offered by the IMS architectural framework and the ease of development and

multimedia capabilities offered by the MSLEE and MMS open source platform.

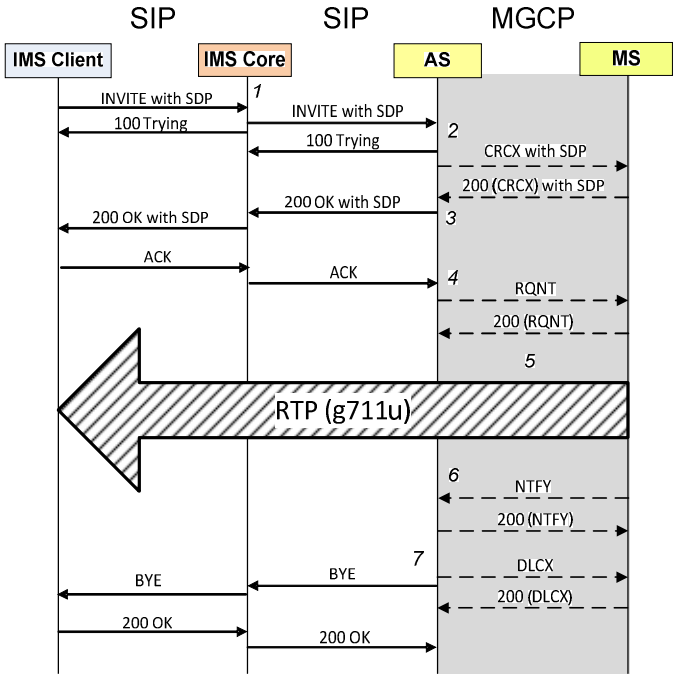

Figure 3 – Signaling and Media Flow

Testbed device	IP Address
IMS Client	192.168.1.173
IMS Core	192.168.1.170
AS	192.168.1.197
MS	192.168.1.2

Table 1 – IP Addresses

6. CONCLUSION

In this paper we present the results of the implementation of a third party Media Server Controller for IMS Networks for multimedia services. Our system is completely based on the open source Mobicents Platform. We have developed a multimedia service and we have integrated it with the FOKUS Open IMS Core framework. We have verified the functionality of the overall system and the suitability of the utilization of the Mobicents JAIN SLEE as an AS in the IMS context.

A strong aspect of JAIN SLEE specification is the abstract layer made of Resource Adaptors which offers the interoperability with external resources and network protocols. This modular architecture has proved to ease the integration of the IMS Core with an external AS, by means of a SIP RA. In the same way, the development of complex multimedia services has been carried out by involving an external Media Server through a MGCP RA. Also, the utilization of an open source modular architecture reduces the required effort for the service development, as the programmer benefits from the RA high level APIs, and consequently its time-to-market.

7. REFERENCES

[1] Rothenberg C. E., Roos A, 2008. A Review of Policy-Based Resource and Admission Control Functions in Evolving Access and Next Generation Networks, Journal of Network and System Management 16: 14–45.

[2] Blum N., Jacak P., Schreiner F., Vingarzan D., Weik P. 2008. Towards Standardized and Automated Fault Management and Service Provisioning for NGNs, Journal of Network and System Management, 16:63–91.

[3] D. Di Sorte, M. Femminella, G. Reali, 2004. A QoS Index for IP Services to Effectively Support Usage-based Charging, IEEE Communications Letters, vol. 8(11), November 2004, pp. 686-688.

[4] Cheng Y., Leon-Garcia A., Foster. 2009. Toward an Autonomic Service Management Framework: A Holistic Vision of SOA, AON, and Autonomic Computing, IEEE Communications Magazine.

[5] ETSI ES 282 007. 2008. Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); IP Multimedia Subsystem (IMS); Functional architecture", V2.0.0, 2008-05.

[6] Kovacikova T., Segec P. 2007. NGN Standards Activities in ETSI. Proceedings of the Sixth International Conference on Networking (ICN'07).

[7] Femminella M., Francescangeli R., Giacinti F., Maccherani E., Parisi A., Reali G. 2009. Design, Implementation, and Performance Evaluation of an Advanced SIP-based Call Control for VoIP Services. IEEE ICC 2009, June 2009, Dresden Germany.

[8] Femminella M., Francescangeli R., Giacinti F., Maccherani E., Parisi A., Reali G. 2009. Scalability and Performance Evaluation of a JAIN SLEE-Based Platform for VoIP Services. ITC 21, 15-17 September 2009, Paris, France.

[9] Open Source IMS Core. http://www.openimscore.org

[10] JSR 240, JAIN SLEE v1.1 Web Site http://jcp.org/en/jsr/detail?id=240.

[11] Khlifi H., Gregoire J.-C. 2008. IMS Application Servers: Roles, Requirements, and Implementation Technologies, Internet Computing, IEEE, May-June 2008, Volume: 12, Issue: 3, 40-51.

[12] Gourraud C. 2007. Using IMS as a Service Framework, IEEE Vehicular Technology Magazine, March 2007, 4-11.

[13] Mobicents Web Site, available at: http://www.mobicents.org.

[14] Mobicents Google Pages Web Site, available at: http://groups.google.com/group/mobicents-public/web?pli=1.

[15] Open Cloud Web Site, available at: www.opencloud.com.

[16] jNetX Web Site, available at: www.jnetx.com.

[17] JBoss Web Site, available at: www.jboss.com.

[18] Java SE 6 HotSpot Virtual Machine GC Tuning, Web Site: http://java.sun.com/javase/technologies/hotspot/gc/gc_tuning_6.html#available_collectors.

[19] Van Den Bossche B. et al. 2006. J2EE-based Middleware for Low Latency Service Enabling Platforms. Proceedings of Global Telecommunications Conference.

[20] Van Den Bossche B. et al. 2006. Enabling Java-based VoIP backend platforms through JVM performance tuning. IEEE VoIP MaSe.

[21] J. Jamae, P. Johnson, “JBoss in Action”, Manning, 2009.

[22] Fraunhofer FOKUS Open IMS Playground, Web Site: http://www.fokus.fraunhofer.de/en/fokus_testbeds/open_ims_playground/index.html

Index	Protocol	Message	Relevant fields
1	SIP/SDP	Invite	Request-Line: INVITE sip:alice@open-ims.test SIP/2.0 Message Header … From: "bob" <sip:bob@open-ims.test>;tag=1 To: <sip:alice@open-ims.test> … Message Body Session Description Protocol Owner/Creator, Session Id (o): user1 53655765 2353 IN IP4 192.168.1.173 Session Name (s): - Connection Information (c): IN IP4 192.168.1.173 Time Description, active time (t): 0 0 Media Description, name and address (m): audio 30000 RTP/AVP 0 8 Media Attribute (a): rtpmap:0 PCMU/8000 Media Attribute (a): sendrecv
2	MGCP	Create Connection	CRCX (CreateConnection) Transaction ID: 1 Endpoint: media/trunk/Announcement/$@192.168.1.2:2729 Version: MGCP 1.0 [The response to this request is in frame 1965] Parameters CallId (C): 1 ConnectionMode (M): sendrecv Session Description Protocol Owner/Creator, Session Id (o): user1 53655765 2353 IN IP4 192.168.1.173 Session Name (s): - Connection Information (c): IN IP4 192.168.1.173 Time Description, active time (t): 0 0 Media Description, name and address (m): audio 30000 RTP/AVP 0 8 Media Attribute (a): rtpmap:0 PCMU/8000 Media Attribute (a): sendrecv
3	MGCP	Response to Create Connection	Response Code: The requested transaction was executed normally. (200) Transaction ID: 1 Response String: The requested transaction was executed normally. [This is a response to a request in frame 1963] [Time from request: 0.304328000 seconds] Parameters ConnectionIdentifier (I): 1 SpecificEndpointID (Z): media/trunk/Announcement/19@192.168.1.2:2729 Session Description Protocol Owner/Creator, Session Id (o): MediaServer 723013 723013 IN IP4 192.168.1.2 Session Name (s): session Connection Information (c): IN IP4 192.168.1.2 Time Description, active time (t): 0 0 Media Description, name and address (m): audio 1043 RTP/AVP 0 8 Media Attribute (a): rtpmap:0 pcmu/8000 Media Attribute (a): rtpmap:8 pcma/8000
4	MGCP	Notification Request	RQNT (NotificationRequest) Transaction ID: 2 Endpoint: media/trunk/Announcement/19@192.168.1.2:2729 Version: MGCP 1.0 [The response to this request is in frame 1998] Parameters NotifiedEntity (N): 192.168.1.197@192.168.1.197:2728 RequestIdentifier (X): 1 SignalRequests(S): A/ann@1(http://192.168.1.2:8080/mgcpdemo/RQNT-ULAW.wav) RequestedEvents (R): A/oc@1 (N)
5	RTP	Data	Payload type: ITU-T G.711 PCMU (0)
6	MGCP	Notify	NTFY (Notify) Transaction ID: 3 Endpoint: media/trunk/Announcement/19@192.168.1.2:2729 Version: MGCP 1.0 [The response to this request is in frame 2405] Parameters NotifiedEntity (N): 192.168.1.197@192.168.1.197:2728 RequestIdentifier (X): 1 ObservedEvents (O): A/oc@1
7	MGCP	Delete Connection	DLCX (DeleteConnection) Transaction ID: 4 Endpoint: media/trunk/Announcement/19@192.168.1.2:2729 Version: MGCP 1.0 [The response to this request is in frame 2421] Parameters ConnectionIdentifier (I): 1

Table 2 – Relevant Messages