Framework for IMS Service Scenario Implementation

Andrey Krendzel[1], Jawad Hussain[2,*],
Josep Mangues-Bafalluy[1], and Marc Portoles-Comeras[1]

[1] Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), IP Technologies Area
PMT, Av. Carl Friedrich Gauss 7, B4, 08860 Castelldefels – Barcelona – Spain
{andrey.krendzel,josep.mangues,marc.portoles}@cttc.cat
[2] Royal Institute of Technology (KTH),
Kungl Tekniska Högskolan, SE-100 44 Stockholm-Sweden
jawad.kth@gmail.com

Abstract. This paper presents an experimental framework for implementation of an IMS/NGN reference service scenario by means of open source software. Multiple service enablers are deployed to build this service scenario. Interoperability tests between the deployed IMS entities and user equipment are carried out, as well as performance measurements of signaling overhead and delay of different IMS procedures involved to support the service scenario. Then, some preliminary results are obtained. After that, the IMS prototype is integrated with in-lab UMTS/HSDPA and WLAN networks to test IMS procedures in more close-to-real environment. Additionally, practical experiences with the IMS testbed deployment are discussed.

Keywords: IMS, testbed, open source, service scenario, service enablers, interoperability, performance evaluation, decomposition.

1 Introduction

The IP multimedia subsystem (IMS) represents a uniform open architecture platform for a managed IP-based infrastructure that will enable the deployment of both basic calling services and an unlimited number of wireless-enhanced rich multimedia services that mix telecom and data services. In this context, rich means bundling multiple service enablers (e.g., voice/video connectivity, presence, instant messaging, conferencing, gaming, TV broadcasting) [1].

By itself, IMS does not specify any new service. Instead, it provides flexible tools and a unified platform for network operators and service providers to build and create their access-agnostic service scenarios by means of reusable service enablers [2].

To identify the problems related with the deployment of an IMS platform as well as to evaluate the performance of IMS-related components and different IMS/NGN services, there is the need for an IMS testing infrastructure [3]. One of ways to build an IMS testbed is to use open and vendor-independent source code that is available for free.

* This work was developed during an internship of Jawad Hussain at the CTTC.

There have been some publications concerning IMS testbed deployment based on open source tools and technologies. For instance, references [4] and [5] describe the experience of the authors when implementing IMS core components consisting of control functions and a subscriber database. On the other hand, references [6] and [7] deployed prototypes for interoperability testing and critical performance evaluation of a presence service, respectively. Additionally, reference [8] focuses on the implementation of a location service enabler on top of an open source IMS core. Thus, most of the above papers are focused on studying a certain IMS enabler (e.g., presence, XDMS, or location) or on studying the IMS core.

In this paper, we present an open IMS testbed deployment in the context of the implementation of a reference service scenario that integrates multiple service enablers. The main contributions of the paper may be summarized as follows:

- We develop a framework for reference service scenario implementation by previously decomposing the whole implementation process into four steps (planes). In the first plane, we describe the service scenario itself. In the second plane, we define the service enablers that the implementation of such a scenario requires. In the third plane, we consider the functional entities that these service enablers involve. In the fourth plane, we map all these functionalities into some IMS physical entities.
- We deploy these IMS entities using different open source packages and we discuss about our practical experience when development the IMS testbed.
- We carry out interoperability tests between the IMS entities and a client in our testbed. We also evaluate the performance of the IMS procedures of the tested service scenario by characterizing their delay and signaling overhead.
- We integrate the IMS prototype with the EXTREME Testbed® [32] to validate it in close-to-real environment. In particular, we test IMS procedures through UMTS/HSDPA and WLAN access technologies.

The rest of the paper is organized as follows. In Section II, a brief description of the IMS concept from the viewpoint of service provisioning principles is given. In Section III, an approach to an IMS service scenario implementation based ondecomposition is considered. In Section IV, the testbed architecture to support this service scenario is presented and practical issues concerning the testbed deployment are discussed. Section V focuses on test cases with regard to interoperability issues. Section VI presents the preliminary results of the measured performance metrics. In Section VII, aspects concerning integration of IMS testbed with in-lab UMTS/HSDPA and WLAN networks are considered and delay that these technologies bring in IMS procedures is discussed. Section VIII concludes the paper.

2 Main Principles of the IMS Concept

The IP multimedia subsystem (IMS) concept has been developed for the provision of IP multimedia services by means of IP multimedia sessions to users. While the IETF has standardized Internet protocols (e.g., SIP, Diameter, etc.), the 3GPP has defined the IP Multimedia Overlay platform (the IMS specification began in 3GPP Rel'5 in 2002) over different underlying technologies (e.g., GPRS/UMTS,

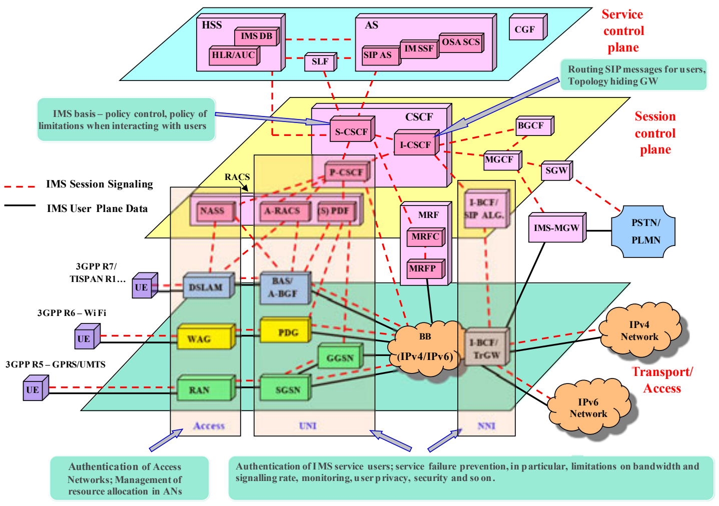

Fig. 1. IMS functional architecture overview

WLAN, DSL) for session control based on Internet protocols as well as the procedures to support generalized mobility across these technologies. IMS also inherits the traditional telecommunications experience concerning guaranteed QoS, flexible charging mechanisms, etc. [9]. An overview of the IMS functional architecture based on 3GPP, ETSI TISPAN, and ITU specifications is presented in Figure 1.

By analyzing Figure 1, one may observe that IMS integrates and develops some ideas from the Intelligent Network (IN) concept. For instance, in the IN concept, the logic of the service (e.g., a toll free or 800 number) was separated from the core switching system (TDM switches) by means of an external node called the service control point (SCP). There is also a triggering point, called the service switching point (SSP), that was added to TDM switches to forward a call related to an IN service towards the SCP by means of the Intelligent Network Application Part (INAP) protocol. The INAP protocol allows the SCP to control and monitor the switch [1]. Since services are no longer developed in the TDM switch, service providers enable developing different value-added services (VAS) for their networks without submitting a request to the core switch manufacturers and waiting for the long development process [10].

In the same manner, in IMS, service-related functions are independent of the underlying transport-related technologies [11]. Roughly speaking, the Call State Control Functions (CSCF) of IMS correspond to the functionality of the TDM switch/SSP in the IN concept, and a SIP Application Server (AS) in IMS corresponds to SCP in IN. Both concepts have triggering criteria to invoke a service, but the respective triggering mechanisms are completely different.

Besides, the IN offered an idea of service independent building blocks (SIBs) for reusable service functions. In accordance with this idea, a service is built as a composition of various SIBs.

In the same manner, in accordance with the IMS concept, a service itself is not standardized, but service building blocks that are reusable by various services. These building blocks are called service capabilities by 3GPP, service support capabilities by ITU-T, and service enablers by OMA [1].

The objectives of IN were not fully reached because of the lack of independence from the INAP protocol, the lack of software reusability, and the lack of openness by manufacturers and operators [1]. The approaches developed within the IN concept cannot be directly applied to IMS because of the service, signaling, and architectural differences between IN and IMS.

However, some principles of the IN concept can be useful in the context of IMS service provisioning. In particular, in IN there is a conceptual model defined in ITU-T recommendations (Q.1211, Q1213-Q.1215, Q.1218, Q.1219). In accordance with these recommendations, a service implementation process includes several planes. In the first plane, the service and its features are described; the second plane is the global functional plane, where SIBs and the global service logic are defined; the distributed functional plane defines functional entities that are involved in the implementation of a service and the relationship between them; the last plane deals with physical entities and protocols.

It is worthwhile to consider in a similar way the issues concerning the implementation of IMS services. It will help to provide a detailed view of service modeling in IMS. One such approach in the context of the deployment of an IMS reference service scenario is introduced in the next section. In this case, and based on decomposition, the whole implementation process is divided into some steps (planes).

3 Decomposition of the IMS Service Scenario Implementation Process

Similar to the IN conceptual model, our approach to the deployment of an IMS reference service scenario includes four planes, namely, service description plane, service enablers plane, functional plane, and implementation plane. The proposed decomposition into planes is illustrated in Figure 2. Each of these planes in the context of a service scenario is described below.

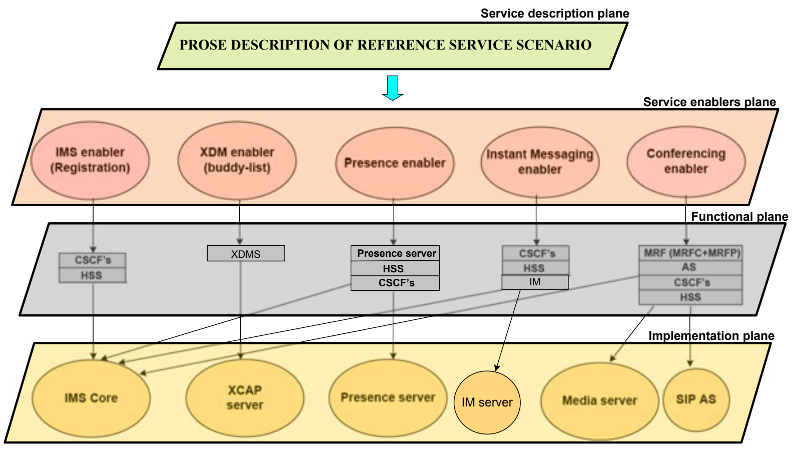

Fig. 2. Decomposition of the IMS service scenario implementation process

3.1 Service Description Plane

The service description plane (see Figure 2) deals with the description of a service scenario to be realized by means of the IMS platform. Any network issues needed for the service implementation are not considered in this plane.

As a case study we consider an abstract simple service scenario that describes a real life situation. Several similar scenarios can be developed and implemented by means of IMS to satisfy the different needs of subscribers.

The prose description of the reference service scenario used throughout this paper follows. There are three close friends from childhood time: Mark, Bob, and Nicholas living in one neighborhood. Now they are businessmen working at different places during the day. They are quite busy, but when all of them are free, they like to occasionally go to a night party together by previously discussing where it is better to go. Thus, they are interested in being subscribed to a service that from a user (a “friend”) perspective may be described as follows. Mark creates his contact list by adding the contacts of Bob and Nicholas in it. He is able to see in the contact list whether his friends are available and willing to communicate or not. Mark also allows spreading information about his availability to Bob and Nicholas. When all his friends have as status “available for a party”, Mark sends them in parallel a short message like “we have a conference call in 10 minutes to discuss where to go”. In 10 minutes, Mark initiates a conference (e.g., audio) with his friends.

This service scenario includes some procedures that must be supported to eventually deploy this service scenario. The procedures and service enablers that are needed to implement it are considered in the next plane, called service enablers plane.

3.2 Service Enablers Plane

Since it is assumed that the scenario is deployed by means of IMS, there is the need for a registration procedure in the IMS platform. Additionally, the following procedures must also be supported: the procedure to maintain a contact list, the procedure to handle information to be aware of contact availability and willingness to communicate (presence information), the procedure to send short messages, and the procedure to arrange a conference call between multiple participants. Note that some other procedures (e.g., those concerning policy and charging issues) should also be involved to support this service scenario, but they are out of the scope of the paper.

Thus, the service scenario requires some service enablers that perform the above procedures. In particular, the user registration procedure deals with mutual authentication between the user equipment (UE) and the IMS platform by using the 3GPP Authentication and Key Agreement (3GPP-AKA) mechanism. This process is carried out by means of the IMS core infrastructure (IMS enabler) including control functions and the subscriber database.

The procedure to maintain a contact list (buddy-list) including friends, family members, colleagues, or other groups with whom an individual may want to communicate is carried out by the XDM (XML document management) enabler. This enabler is in charge of making user-specific service-related data available to other services and service enablers within the IMS network [12]. The buddy-list is the type of data that is mainly associated with the XDM enabler.

The procedure to handle presence information is supported by the Presence enabler, which spreads user-related presence updates (i.e., current status of a user) throughout the network to other users who register to receive notification of changes of a user's presence and availability state [13].

The procedure to send short messages is fulfilled by means of the Instant Messaging (IM) enabler, which allows engaging two or more users in a real-time text messaging.

Finally, the procedure to arrange a conference call between two or more users is carried out by the Conferencing enabler, which supports communication between multiple participants. It allows a user to initiate, modify, and terminate media sessions. Conferencing applies to any kind of media stream by which users may want to communicate [14].

The above enablers belong to the plane of service enablers shown in Figure 2.

3.3 Functional Plane

Each service enabler is characterized by a set of functional entities (FEs) and relationships between them, as specified in the IMS standards. These functional relationships between FEs are usually illustrated by information flow diagrams [15].

The IMS enabler involved in carrying out the IMS registration procedure includes Call State Control Functions (CSCFs), namely, Proxy-CSCF (P-CSCF), Interrogating-CSCF (I-CSCF), Serving-CSCF (S-CSCF), and Home Subscriber Server (HSS). The CSCFs are responsible for routing signaling and managing sessions [33]. The HSS is a database including user identities (both public and private) and service-related information. It is responsible for Authentication, Authorization and Accounting (AAA) [33].

The Presence enabler deals with the widespread publication and subscription of presence information [15]. A user can subscribe to presence information for his/her contacts. If the contact accepts his request, the subscriber (watcher) will be registered for presence notification. Whenever a friend (presentity) publishes presence information, the IMS presence framework will notify the subscribed users. This enabler involves the IMS core functional entities (CSCFs) and the Presence server, which manages presence information uploaded by a presentity UE and handles presence subscription requests.

The XDM enabler deals with documents that are stored in (logical) repositories in the network, generically referred to as XML Document Management Servers (XDMS). Each repository is associated with a functional entity that uses the data in its associated repository to fulfill its functions [17]. Thus, the XDMS has a service- independent functionality that can be used in a variety of person-to-person or group communications [17].

The Conferencing enabler involves the CSCFs, the HSS, the Multimedia Resource Function (MRF) (consisting of Media Resource Function Controller (MRFC) and Media Resource Function Processor (MRFP)), and the Application Server (AS). The MRF provides media related functions, such as media manipulation (e.g., voice stream mixing) and playing of tones and announcements [16]. A user initiates a conference by means of the MRFC/AS entity of the user's home network that assigns a conference URI (Uniform Resource Indicator) to the conference and configures the

MRFP. The conference call is established and the RTP data begin flowing between the UE initiating the conference and the MRFP [15]. The conference initiator then uses the refer procedure to add more users to the conference. The new users establish a call to the conference URI included in the refer message. When the conference is in progress, RTP media streams are mixed and propagated to all participants [15].

The Instant Messaging (IM) enabler is the well-known instant messaging paradigm adopted in the IMS framework [18]. It involves the CSCFs, the HSS, and the IM server. An instant message (IM) is transferred by using a SIP MESSAGE, which is a SIP extension defined in IETF RFC 3428 [14]. Messages are directly sent out to the destinations through the IM server. Upon receipt of a MESSAGE request, a SIP UA (user agent) will reply with a 200 OK or a 202 Accepted response, which indicates that the SIP UA has received the SIP request message [14].

By focusing on the functional plane represented in Figure 2, one may notice that the CSCFs and the HSS functional entities are common to several enablers. Note that the "servers" (i.e., Presence, XDM, IM, and AS) in context of the plane are functional entities.

3.4 Implementation Plane

In this plane, all functional entities defined in the previous plane are mapped into the corresponding physical entities that have to be deployed in the IMS prototype. In fact, one or more functional entities may be mapped into the same physical entity. In particular, P-CSCF and I-CSCF are SIP proxy servers [7]. S-CSCF plays the role of SIP proxy and SIP registrar [7]. All these components together with the common database (HSS) are mapped into one physical block called the IMS core.

The presence functionality defined by the Presence enabler requires a SIP Presence server. The instant messaging functionality is supported by means of the SIP IM server. The conferencing enabler functionality requires a SIP media server to support the media resource function and a SIP AS to support the end-user service logic. An XCAP server is needed to provide the ability to query, modify, or delete data stored in XML.

Thus, all these physical entities (the IMS core, the Presence server, the XCAP server, the IM server, the Media Server, the SIP AS) must be deployed to support the above service scenario (see the implementation plane in Figure 2). Practical issues concerning the deployment of the testbed, including the above entities, are considered in next session.

4 Testbed Architecture and Deployment

The architecture of our IMS prototype to implement the reference service scenario is presented in Figure 3. In accordance with the service scenario, all friends live in one neighborhood and it is assumed that they belong to the same IMS network. Thus, we initially develop the testbed for intra-domain test cases. For other service scenarios, the testbed can be extended to inter-domain test cases.

We have analyzed available open source packages to build this testbed since they are free, they are not identified with a particular vendor, and they supply codes. We have installed and configured the above-defined IMS entities slightly modifying codes to support interoperability between them.

For the testbed deployment, we use a Linux desktop machine, Genuine Intel Pentium (R) Dual–Core CPU 2.50 GHZ with 4GB RAM, and Ubuntu 9.04 OS.

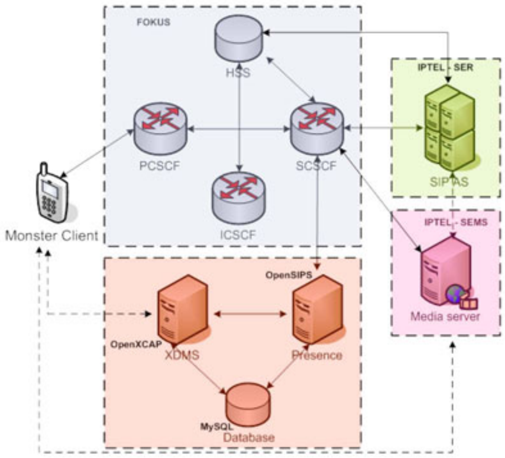

Fig. 3. The IMS testbed architecture

We have started with the development of the IMS Core. The four components (P/S/I-CSCFs and a lightweight HSS) are deployed by using open source code developed by FOKUS [19] as extensions to the SIP Express Router (SER) [26]. It is available through svn repositories at [19]. The lightweight HSS supports the diameter protocol and uses the Cx interface for diameter signaling with the S/I-CSCF [19]. User data is kept inside a MySQL database. We installed this open source (revision 778) and we found that it presents a very efficient and reliable implementation from the viewpoint of operations, administration, and maintenance (OA&M). All IMS core components run on one IP (on loopback), but using different ports.

The simplified flow sequence diagram captured in the testbed for the UE registration procedure is presented in Figure 4.

Our Presence server implementation is based on Open SIP Server (OpenSIPS) [20], which is a mature open source implementation of a SIP server. OpenSIPS (previously called OpenSER) is more than a SIP proxy/router, as it includes many application functionalities. It unifies voice, video, IM, and presence services. We have tested OpenSIPS as a Presence server and we have integrated it in our IMS core. For this purpose, we used OpenSIPS v.1.5, which is the last available version in svn repositories. The presence service based on the open source of this version demonstrates good interaction with the implemented IMS core entity in our testbed. It handles SIP SUBSCRIBE, PUBLISH, and NOTIFY methods. Presence data from subscribers and publishers are saved in a MySQL OpenSIPS database (see Figure 3) in the corresponding presentity and watcher tables. The simplified sequence diagrams for IMS presence subscription/notification and publication procedures based on the collected traces are shown in Figures 5 and 6, respectively.

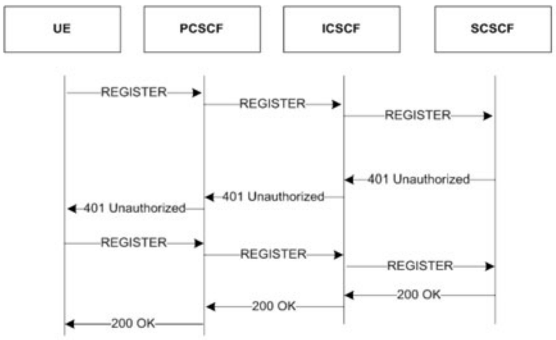

Fig. 4. Testbed registration procedure

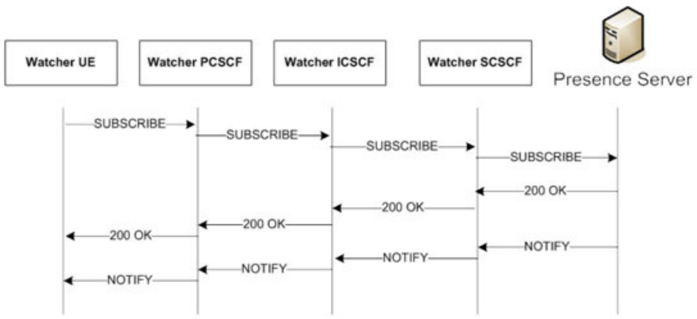

Fig. 5. Testbed presence subscription/notification sequence diagram

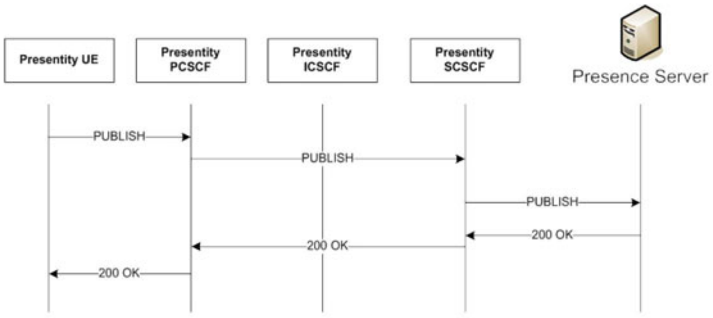

Fig. 6. Testbed presence publication sequence diagram

For the XCAP server implementation, the OpenXCAP source (v.1.1.2) has been used [21]. It is an open source package supporting a full-featured XCAP server. An XCAP server is used by SIP SIMPLE clients and servers to manage buddy-lists and policy for subscriptions to the presence service and it can support other types of published events by using SIP.

We have tested the interaction between the UE and the OpenXCAP server to manage buddy-list by means of the XCAP protocol (see Figure 7). Moreover, we made some configuration changes in OpenSIPS to support intercommunication between the Presence server and the OpenXCAP server. The OpenXCAP server uses the XML-RPC interface to interwork with the presence server, as it is illustrated in Figure 7. For this purpose, the xmlrpc_url port must match the OpenSIPS mi_xmlrpc port. The OpenXCAP backend is used for storage and authentication purposes, and data may be saved either in its own database or in the OpenSIPS database. We configured OpenXCAP to use the OpenSIPS database for service subscribers and XCAP resources.

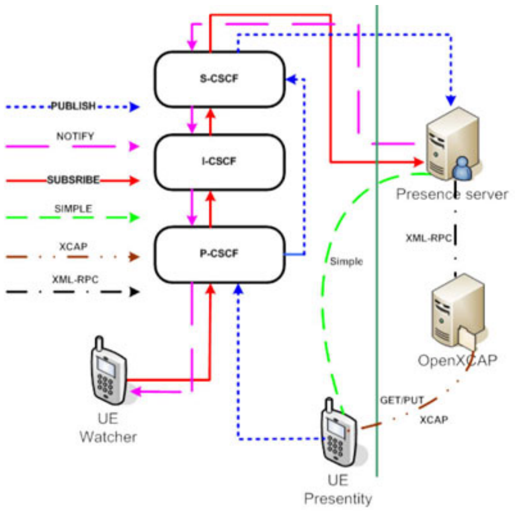

Fig. 7. Interaction diagram between the UE, Presence server, and XCAP server

As media server, we used SIP express media server (SEMS) v.1.1.1 developed by IPTEL [22]. It provides audio centric MRF core functionalities. We have tested its basic applications, like announcement and conferencing. SEMS runs on loopback, having the open-ims.test domain name. We used the SEMS implementation with unixctrl interface and configured it to handle the announcement and conferencing module support.

To support end user service logic, we deployed a SIP AS. There are many SIP application servers available, like Sailfin [23] or Mobicents [24] based on SIP servlet technology. However, we preferred to deploy a SIP AS based on the SIP express router (SER) [25], since the IMS core is also based on SER. In our IMS prototype, the server is located between the IMS core and SEMS entities. SER is a high-performance and configurable SIP server that can act as SIP registrar, proxy, or redirect server [25]. We used the SER 0.96 version [26] available at svn repositories that is able to interact with SEMS. We used SER in re-direct server mode to forward SIP INVITE, ACK, BYE, and CANCEL methods to SEMS. We configured SER on loopback and it has the same domain name (open-ims.test). SER writes the received SIP messages and forwards them to SEMS by using the unix socket server. The testbed conference call initiation diagram is shown in Figure 8.

Our IMS user agent is based on an open environment developed by FOKUS and called MONSTER (The Multimedia Open InterNet Services and Telecommunication EnviRonment) [27]. We used the most recent release, i.e., version 0.9.8. It consists of a suite of integrated applications, such as voice, video, photo sharing, contacts, etc. [27]. However, the MONSTER public version has limited functional features that can be extended by the use of the MONSTER API. In particular, we observed that it does not support video conference.

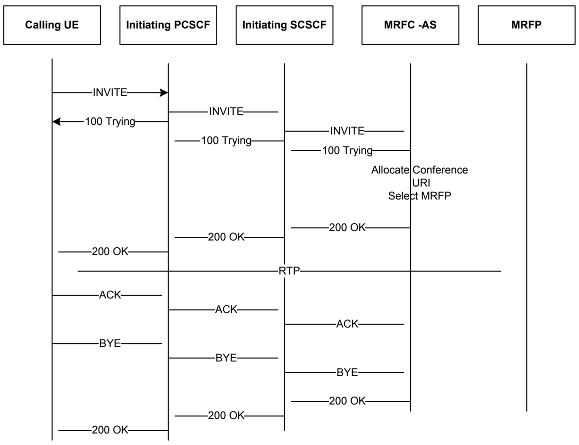

Fig. 8. Testbed conference call initiation sequence diagram

Besides, we also tried to deploy an Instant Messaging (IM) server. OpenSIPS used before to deploy the presence server can be also configured to support end-to-end IM. We installed and integrated an external e-jabbered IM server v.2.1.0 [28] by using the XMPP OpenSIPS module. However, later, we found that interoperability between the UE and the IMS server is not supported, since the current public MONSTER version does not provide IM functionality, though it can be developed and added by means of the MONSTER API. At the same time, the MONSTER client allows supporting the SIP MESSAGE method, and in the service scenario this feature can be used by friends instead of instant messaging. For this reason, currently, we do not use the IM server in our testbed.

Thus, we have deployed the following IMS components in our IMS testbed: the IMS core, the Presence server, the XCAP server, the Media server, the SIP AS, and the UE. The results of some relevant interoperability tests between the components for our reference scenario are considered in the next section.

5 Interoperability Tests

To check interworking between the UE and the Presence, XCAP, and Media servers, interoperability tests have been carried out. Table 1 presents all test cases run. Test cases 1-5 are related to the Presence server, test cases 6-8 are related to the XCAP server, and test cases 9-10 are related to the Media server, respectively.

Table 1. List of interoperability tests carried out over the testbed

TEST CASE #	*TEST TITLE*	*TEST CASE DESCRIPTION*
1	Publication of presence information	Presence information initially published by IMS UE must be correctly received by IMS subscribers that are eligible for receiving it
2	Publish modification	Published information modified by the UE must be correctly received by the presence subscribers
3	Subscription removal	If a watcher removes its presence subscription, a presentity must update presence information
4	Subscription refresh	Presence server must continue to send presence information to a watcher in case presence subscription is refreshed
5	Notification of presence information from multiple presentities	This case is similar to test case 1, but it is extended to multiple presentities
6	Group-list XDMS document creation	UE must be able to successfully create its buddy-list in the XCAP server by using http PUT/DELETE methods
7	Group-list XDMS document retrieval	By means of the http GET method, an IMS user can successfully download its buddy-list from the XCAP
8	Group-list XDMS document validation and deletion	The UE must be able to manipulate its buddy-list
9	Playing media announcements	The Media server must be able to play an audio announcement
10	Supporting conferencing application	The media server must support conferencing with two or more UEs

We have repeated each of the above test cases 15 times. Test results are presented in Table 2.

Table 2. Results of the interoperability tests

# OF TEST CASES	*# OF TRIALS*	*# OF PASSES*	*# OF FAILS*
10	*15*	*134*	*16*

As one can observe in Table 2, 134 out of 150 trials have been successful. All 16 failures have arisen in test cases related to the Presence server, when the UE of a presentity is being switched off. In this situation, the MONSTER UE is not always able to send a publish message to the Presence server. As a result, watchers can see the status of the presentity as "on-line" instead of "not available". The same observation has been obtained in [6], when testing the SIMPLE presence protocol implemented by the UCT client [29] and the IMS Communicator [30], which were developed before MONSTER. Both clients display the last presence information published by the UE [6]. Thus, this problem still exists in the public available version of the MONSTER client as well.

We also verified that all testbed sequence diagrams (see Figures 4-6 and Figure 8) are in conformance with the message sequence charts provided by the IMS standards.

6 Preliminary Results

Some preliminary results were obtained in our IMS testbed. In particular, we evaluated two performance metrics that characterize the delay and overhead of the IMS signaling procedures deployed to support the reference service scenario. In particular, the first metric accounts for the time elapsed since the UE sends a signaling message to the IMS server until it receives a response message back from the server according

Table 3. Test cases for evaluating the round trip signaling time of various procedures

TEST CASE #	*IMS PROCEDURE*	*TEST CASE DESCRIPTION*
1	IMS REGISTER	RTST between sending the initial REGISTER request to the IMS core and reception of the 200 OK message (total delay for registration and de-registration procedures)
2	IMS INVITE (IMS to IMS call)	RTST between sending the initial INVITE message to the IMS core and reception of the 200 OK
3	IMS watcher REGISTER/SUBSCRIBE	RTST that a watcher spends sending REGISTER request to the IMS core, SUBSCRIBE-ing to presence information about presentity and receiving NOTIFY message from presence server (PS)
4	IMS presentity REGISTER/PUBLISH	RTST that a presentity spends sending the REGISTER request to the IMS core, PUBLISH-ing to PS and reception of the 200 OK
5	IMS presentity PUBLISH	RTST that a presentity spends when Presence state changes (PUBLISH) for the Watched User and reception of the 200 OK message from the PS
6	IMS INVITE for Media server	RTST between sending the initial INVITE message to Media server and reception of the 200 OK message from the server
7	IMS HTTP PUT for XCAP server	RTST between the UE PUT-ting its buddy-list in the XCAP server and reception of the HTTP 200 OK message from the server
8	IMS HTTP DELETE for XCAP server	RTST between the UE DELETE-ing its buddy-list from the XCAP server and reception of the HTTP 200 OK message from the server

to a certain IMS procedure. We call it the round trip signaling time (RTST) below. The second performance metric is the signaling overhead (which accounts for the application layer payload) generated by SIP messages for a certain IMS procedure. We performed each test case for these metrics ten times and captured the associated traces by using Wireshark [31].

The description of test cases to evaluate the RTST for different IMS procedures is presented in Table 3.

The box-plot presented in Figure 9 shows the results of the signaling time evaluation of the different IMS procedures.

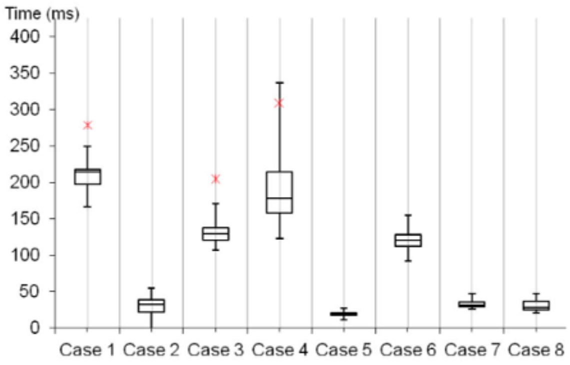

Fig. 9. RTST evaluation of different IMS procedures

The results of the signaling overhead generated by SIP messages in the above IMS procedures are presented in Table 4.

Table 4. Overhead (application layer payload) associated to each of the deployed procedures

TEST CASE #	*IMS PROCEDURE*	*AV. NUMBER OF BYTES*	*ST. DEVIATION*
1	IMS REGISTER	21.320	89.77
2	IMS INVITE (IMS to IMS call)	19487	12.98
3	IMS watcher REGISTER/SUBSCRIBE	23141	79.96
4	IMS presentity REGISTER/PUBLISH	15990	82.41
5	IMS presentity PUBLISH	4724	20.69
6	IMS INVITE for Media server	13130	5.61

The analysis of the results presented in Figure 9 (RTST) and Table 4 (overhead) brings the following observations. The total time that both registration and de-registration procedures take is quite high (the first test case). SIP messages between CSCFs are transmitted and processed in a small amount of time. On the other hand, the main component of the delay comes from the communication with the HSS when it is needed to retrieve information (e.g., when the S-CSCF downloads the authentication vector) from the MySQL database that stores user-related data. This observation is further confirmed by the second test case, in which just CSCFs are involved in routing and processing all signaling messages to establish an IMS to IMS call. As it may be observed, this procedure takes much less time, although it generates almost the same signaling overhead.

Test cases 3 and 4 do not contain the de-registration procedure, but the subscription and publication procedures initiated after the registration are considered, respectively. The procedures are also time-consuming. Especially, average RTST and dispersion are significant for the publication procedure. This is because of the authorization process of the watcher (subscriber) or publisher (presence entity) with the Presence server, which requests user presence data stored in the MySQL openSIPS database. Test case 5 further confirms this observation, as it considers a case for which presence state changes, hence sending PUBLISH messages to the Presence server, but without requiring the publisher to run an authorization process. As a result, the delay in this case is very low compared to the previous one. Note that in test case 3, more signaling overhead is generated than in test case 4 due to the additional payload bytes devoted to the NOTIFY message (see Figure 5 and 6).

The delay in test case 6 is caused by the interaction with the media server to assign a conference URI to the conference, to determine media capabilities, etc. The INVITE procedure for media server (test case 6) contains less message exchanges between IMS entities than the INVITE procedure for IMS to IMS call establishment (test case 2). As a result, the first one generates less overhead, as seen in Table 4.

Additionally, the RTST values obtained for test cases 7 and 8 (PUT/DELETE procedures of the HTTP protocol) show that the average time required to put/delete a buddy-list in/from the XCAP server is approximately of 25 ms, hence much lower than other more complex procedures.

Finally, if we consider the obtained results for our reference service scenario then from a user (a “friend”) perspective it generates an average of 40 Kbytes of signaling overhead and lasts for 280 ms, in our IMS prototype. The sequence of actions included in this calculation are: Mark’s *registration* to IMS core, *subscription* to the presence service to get *notifications* on the presence status of his friends, *publication* of his presence information so that it is available to his friends, and *initiation* of the conference call with them.

7 Integration of the IMS Prototype with the EXTREME Testbed®

To better validate IMS operation in close-to-real environment, we have started the integration of the IMS prototype with the EXTREME Testbed® [32]. In particular, we connected the UE to the deployed IMS platform through two transport technologies, namely UMTS/HSDPA and WLAN IEEE 802.11, as illustrated in Figure 10.

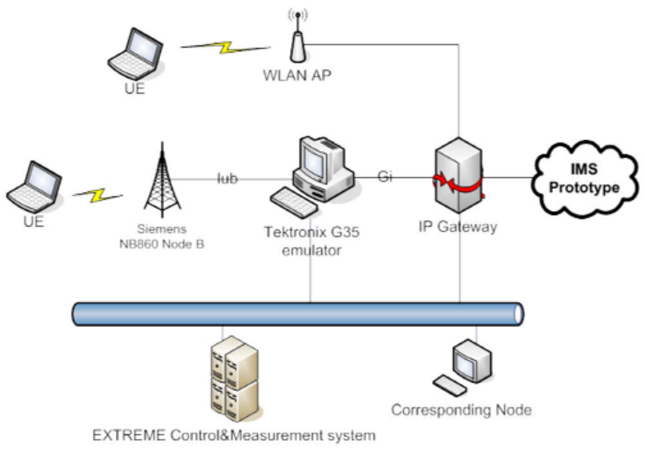

Fig. 10. Integration of the IMS prototype with EXTREME testbed ®

The in-lab real-time UMTS network consists of a real Node B (Siemens NB8860) with WCDMA and HSDPA functionalities, a protocol emulator (Tektronix K1297-G35) of the RNC and the PS core network (including SGSN, GGSN, and a subset of the functionalities of the HLR), and a PC acting as IP gateway to external networks [33]. An ATM optical interface connects the Node B to the G35 emulator (featuring a real Iub interface). The G35 communicates to the EXTREME platform via a Gigabit Ethernet interface (through the Gi interface) allowing interworking between UEs and nodes in the Internet.

The WLAN access point (AP) is based on IEEE 802.11g, configured in "Infrastructure BSS" mode, and connected to the IP Gateway. UMTS/HSDPA and WLAN wireless cards are used to connect to the UMTS and WLAN networks, respectively.

We integrated our IMS prototype through the IP gateway with the UMTS and WLAN networks. To check conformance, the IMS registration/de-registration procedures have been run (10 times) through both transport technologies. The results of the registration/de-registration delay for these procedures obtained in the IMS EXTREME testbed are presented in Table 5. It is interesting to compare them with the values of delay for the same procedures with the same transport technologies measured in the "IMS experience centre" [3] for a real-life test network using real UMTS/HSDPA equipment. The results got in the IMS experience centre testbed are also presented in Table 5.

Table 5. Comparison of IMS registration/de-registration delay in different testbeds

Testbed name	Network	Registration delay, ms (min/max/avg)	De-registration delay, ms (min/max/avg)
IMS EXTREME tes	WLAN (802.11g)	100/137/120	80/114/99
	UMTS/HSDPA	288/332/309	242/290/263
IMS experience centre testbed	WLAN (802.11g)	9/89/22	12/28/16
	UMTS/HSDPA	460/840/520	470/750/520

By analyzing the results obtained in the IMS EXTREME testbed we have observed that registration and de-registration delay over the WLAN network is composed mainly of the above-mentioned delay when retrieving data from the MySQL database (twice for each procedure, each time taking around 40-60 ms). The total registration and de-registration delay over WLAN almost coincides (difference is a few ms) with values obtained for both procedures in the IMS testbed (see test case 1 in Figure 9). Thus, the delay that the WLAN network itself brings is negligible.

On the other hand, the delay that the UMTS/HSDPA network brings is higher. It is almost three times more than in the IMS testbed (~570 ms vs. 210 ms). The results got in [33] in the UMTS/HSDPA EXTREME infrastructure observed an average downlink delay of 40 ms and uplink delay of 60 ms. For two message exchanges in both directions (see Figure 4) the delay takes around 200 ms plus the delay (ranging from 80 to 120 ms) caused by twice as much requests to the MySQL database. Thus, the total delay is around 300 ms for the registration and around 260 ms for de- registration, which is coherent with the results that we obtained (see Table 5).

By analyzing the results obtained in the IMS experience centre testbed one may observe that the delay measured in that case through a WLAN network is very small compared with our results. A potential hypothesis is that they use caching to minimize the delay caused by the MySQL database, since maximal registration delay is 89 ms (probably when retrieving data from the MySQL) which is close to our values, but the average registration/de-registration delay is 22 ms and 16 ms respectively (caching might be used).

On the other hand, the delay experienced through their UMTS/HSDPA network substantially exceeds that obtained in our testbed. A potential explanation may come from the differences in delay (around 90/100 ms downlink/uplink) observed in [33] between a commercial UMTS/HSDPA network and the UMTS/HSDPA EXTREME infrastructure, in which the UMTS core network (SGSN/GGSN) is emulated by means of the G35 test equipment. Given that some IMS procedures require messages to traverse this UMTS/HSDPA infrastructure multiple times, the difference in terms of total signaling delay is substantial increased.

Conclusions

A framework for deployment of an IMS prototype to implement a reference service scenario that involves multiple service enablers has been considered in the paper. The whole service implementation process has been decomposed into four planes, namely, service description, service enablers, functional, and implementation. As a result of the decomposition, we found that for the service scenario, one must deploy the IMS core, the presence server, the XCAP server, the Media Server, and the SIP application server. We installed various available open source packages featuring these IMS entities and we modified and configured them so that they can interact.

To evaluate the interworking between the IMS client [27] and the deployed IMS components, interoperability tests have been conducted. The UE was not always able to send a publish message to the Presence server when the equipment was being switched off. All the rest tests were successful. Besides, we found that the current public version of the client does not support instant messaging and video conference.

We verified that the IMS procedures (registration, subscription, etc.) in our testbed are in conformance with the message sequence charts provided by the IMS standards.

Then, preliminary results concerning the signaling overhead and delay of the different IMS procedures of the service scenario have been obtained. The reference service scenario generates an average of 40 Kbytes of signaling overhead and lasts for 280 ms. We observed that the main delay is caused by retrieving data from the MySQL database. We suppose that caching may reduce essentially the delay. To get access to the deployed IMS platform through in-lab UMTS/HSDPA and WLAN access technologies we integrated our prototype with the EXTREME Testbed® [32]. We observed that the additional delay caused by the WLAN network in IMS registration/de-registration procedures is very low. On the other hand, the delay that UMTS/HSDPA network infrastructure brings in the IMS procedures is essential. It may be several times more then the delay in the IMS platform itself for the procedures.

In future work, we are going to extend our IMS prototype by adding more functionality in the deployed framework as well as in the IMS client to support more complex service scenarios over heterogeneous access technologies. Besides, we are planning to conduct a more exhaustive performance evaluation of different IMS components by using a signaling traffic load generator.

Acknowledgement

This work was supported in part by the Spanish Ministry of Science and Innovation under grant number TEC2008-06826 (ARTICO), by the Catalan Regional Government under grant 2009SGR-940, and by the Spanish Ministry of Industry, Tourism and Commerce under grant number TSI-020301-2008-13 (WIMSAT).

References

[1] Bertin, E., Crespi, N.: IMS Service, Models, and Concepts. In: IP Multimedia Subsystem (IMS) Handbook, ch. 1. CRC Press, Boca Raton (2009)

[2] Sardella, A.: Building IMS-Capable Core Network. White paper, Juniper network (March 2006)

[3] Balakrishna, C.: IMS Experience centre. A real-life Test Network for IMS services. In: Proc. 5th Int. Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (2009)

[4] Anwar, B.P., Singh, K.: IMS SIP core server test bed. In: Proc. Int. Conference on IP Multimedia Subsystem Architecture and Applications (2007)

[5] Tang, J., Davids, C., Cheng, Y.: A study of an open source IP Multimedia Subsystem test bed. In: Proc. Qshine 2008, Hong Kong, China (July 2008)

[6] Maarabani, M., et al.: Interoperability testing of presence service on IMS platform. In: Proc. 5th Int. Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (2009)

[7] Lin, L., Liotta, A.: A critical evaluation of the IMS presence service. In: Proc. MoMM 2006 (2006)

[8] Reichl, P., et al.: Practical experiences with an IMS-aware location service enabler on top of an experimental open source IMS core implementation. Journal of Mobile Multimedia 2(3), 189–224 (2006)

[9] Magedanz, T.: IP Multimedia System (IMS)-Principles, Architecture and Applications. In: 2nd IEEE Workshop on Mobility Aware Technologies and Application, Montreal, Canada (October 2006)

[10] http://en.wikipedia.org/wiki/Intelligent_network

[11] ITU-T. Y.2001, Next Generation Networks – Frameworks and functional architecture models (December 2004)

[12] Al-Begain, K., et al.: IMS: A Development and Deployment Perspective. Wiley, Chichester (September 2009)

[13] 3GPP TS 23.141, Presence service; Architecture and functional description; Stage 2

[14] Ahson, S.A., Ilyas, M.: IP Multimedia Subsystem (IMS) Handbook. CRC Press, Boca Raton (2009)

[15] http://www.eventhelix.com/ims/

[16] http://en.wikipedia.org/wiki/IP_Multimedia_Subsystem

[17] http://www.openmobilealliance.org/Technical/release_program/xdm_v1_0_1.aspx

[18] Mikka, P., Georg, M., Hisham, K.: The IMS IP Multimedia Concepts and Services. John Wiley & Sons, Chichester (2006)

[19] Open IMS Core's Homepage, http://www.openimscore.org/

[20] OpenSIPS project, http://www.opensips.org/

[21] OpenXCAP, http://openxcap.org/

[22] SIP Express Media Server, http://www.iptel.org/sems

[23] Project Sailfin, https://sailfin.dev.java.net/

[24] Mobicents, http://www.mobicents.org/

[25] SIP Express Router, http://www.iptel.org/ser/

[26] SER and SEMS work together, http://ftp.iptel.org/pub/sems/doc/current/Configure-Sems-Ser-HOWTO.html

[27] MONSTER – the client, http://www.monster-the-client.org/

[28] Ejabberd, http://www.ejabberd.im/

[29] http://uctimsclient.berlios.de/

[30] http://developer.berlios.de/projects/imscommunicator/

[31] http://www.wireshark.org/

[32] http://www.cttc.es/en/project/EXTREME.jsp

[33] Dini, P., et al.: A real-time cellular system architecture to experiment with UMTS/HSDPA in a laboratory. In: Proc. of TRIDENTCOM 2009, Washington, USA (April 2009)