Design and Implementation of an IP Multimedia Subsystem (IMS) Emulator Using Virtualization Techniques

F. Galán[1,2], E. García[2], C. Chávarri[2], M. Gómez[3], D. Fernández[2]

1: Centre Tecnològic de Telecomunicacions de Catalunya (CTTC)
Av. Canal Olimpic, s/n - Castelldefels, Spain
fermin.galan@cttc.es

2: Departamento de Ingeniería de Sistemas Telemáticos (DIT)
Universidad Politécnica de Madrid (UPM)
Av. Complutense, s/n - Madrid, Spain
{galan,egarcia,chavarri,david}@dit.upm.es

3: Agora Systems, S. A.
Velázquez, 46 - Madrid, Spain
miguel.gomez@agora-2000.com

Abstract

Multimedia service provisioning in Packet-Switched networks is one of the most active research and standardization efforts nowadays, and constitutes the most likely evolution of current telecommunication networks. In this environment, the 3GPP IP Multimedia Subsystem (IMS) represents the service provisioning platform of choice for SIP-based content delivery in mobile and fixed environments. However, in spite of the advanced status of research and specification in this field, few IMS-based services have been defined or deployed, mainly due to the fact that production or experimental IMS platforms are scarcely available for service validation and testing. This paper addresses the design and implementation through virtualization techniques of an IMS testbed intended for the functional validation of services, presenting the involved technologies, the undertaken implementation strategy, and the experiences and tests carried out in the first stages of its development.

Keywords

IP Multimedia Subsystem, virtualization, VNUML, IMS, 3GPP

1. Introduction

Given the current trend in telecommunications industry towards data packet switching

(the so called *all-IP* approach), the IP Multimedia Subsystem (IMS) plays a key role as service provisioning platform in Next Generation Networks. Within the UMTS (*Universal Mobile Telecommunication System*) core network specification, the 3GPP (*Third Generation Partnership Project*) [1] defines IMS as a platform overlaid to the Packet-Switched domain for the provision of multimedia services (voice, data, messaging, streaming, etc.) with quality of service (QoS) and authentication, authorization, accounting and charging (AAAC) support. Moreover, IMS must provide a neutral framework for advanced services deployment (multimedia communications, intelligent network services, location-based services, etc.), allowing thus the inclusion of third-party service providers.

IMS was originally defined in UMTS Release 5 in early 2002, but since then it has been adopted by several other organizations for many different network environments. In particular, the adoptions by 3GPP2 [2] for CDMA2000 networks and ETSI TISPAN for NGN (Next Generation Networks) prove the success of the architecture and the interest that IMS is generating, expanding also its influence area to a vast set of wired and wireless access technologies, and introducing new operation and interworking environments.

However, the current deployment of UMTS production networks in Europe rarely includes IMS; they are usually based on UMTS releases previous to Release 5. This situation is mainly caused by the lack of an open IMS reference architecture that allows third-party providers to develop and test IMS-oriented services, even though standards are quite mature and stable currently. Although some manufacturers provide IMS demonstrators to its clients (limiting thus the access to such technology practically only to operators), and even some open-source efforts seem to be arising (like Open IMS), there is still a long way ahead in order to build a "reference IMS architecture" that would impulse the deployment of this technology in production networks.

In this context, the contribution presented in this paper tries to address that problem, describing an IMS testbed implemented using virtualization techniques. As virtualization backend, the VNUML (*Virtual Network User Mode Linux*) [3][4] general purpose tool is used.

The remainder of this article is organized as follows. We briefly describe the IMS architecture, as defined by the 3GPP, in section 2. Next, the design and implementation of the IMS testbed are addressed in sections 3 and 4 respectively. Finally, section 5 presents the main conclusions and future research lines.

2. IMS architecture

Figure 1 presents a general overview of the IMS architecture [5]. As shown in the diagram, one of the main characteristics of IMS is the separation between the transport or data layer and the signalling or control layer, allowing thus for easier service creation, deployment and management. The transport or data layer in IMS is in charge of providing IP connectivity to terminals, and allowing signalling and media exchange. Depending on the environment where IMS is being deployed, there are several access network alternatives like, for instance, GPRS (*General Packet Radio*

Service) in 3GPP networks. The signalling layer constitutes the core of the IMS, and is in charge of providing session control through call routing and policy enforcement.

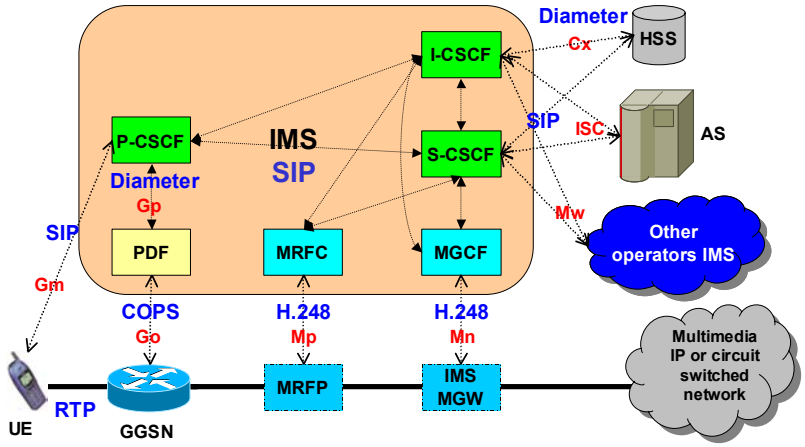

Figure 1: Simplified architecture and signalling protocols in IMS

IMS presents elements on both layers, depending on their role in the multimedia service provision process. IMS elements may be classified into the following functional groups depending on their role and features:

- **Control Elements** constitute the core of the IMS signalling layer. This element group is composed of three different types of SIP proxy/server entities named CSCFs (*Call Signalling Control Functions*). CSCFs are in charge of call routing and session control, each of them focusing on a different aspect:
    - The P-CSCF (*Proxy CSCF*) is the entry point to IMS for subscribers, as it is in charge of handling all the requests to/from users and forwarding them as required. It also performs security and authentication functions, signalling validation, signalling compression, and resource authorization and QoS management in the access network by interacting with the PDF (*Policy Decision Function*).
    - The S-CSCF (*Serving CSCF*) is the central node of the IMS session control framework. It behaves as a stateful SIP proxy/server providing registrar, authentication and session control features. There can be several S-CSCFs in the network, each of them focusing in a different set of capabilities or in order to meet certain capacity requirements. The S-CSCF is always located in the operator's home network, and it constitutes the central control point for operator-provided services by applying the filter criteria for the execution of IMS services.
    - The I-CSCF (*Interrogating CSCF*) constitutes the entry point to an IMS administrative domain for foreign IMS networks. It provides call routing features, and it may also optionally implement topology-hiding capabilities (the so-called THIG -*Topology-Hiding Inter-network*

Gateway-) by encrypting those parts of the SIP messages which contain sensitive information about the inbound domain.

- **Database Elements** are tightly coupled with the control layer. This element group is mainly based on the HSS (*Home Subscriber Server*) and its different database views. This server holds user-related information such as profiles, subscription data, location and security information, etc.
- **Service Elements** are intended for service hosting and execution within IMS. Since they are actually external components, and can be developed, operated and managed by third-parties, some consider this element group a separate layer within the IMS architecture. Depending on the nature of the element providing the service, we may distinguish between native and legacy elements. Native service elements are SIP AS (*Application Servers*), taking the roles of SIP proxies, UAs, or B2BUAs. The S-CSCF, upon a correct match of a service filter criterion, will forward the SIP request to appropriate SIP AS, which will in turn perform the associated service logic. Legacy service elements are inherited from prior architectures (e.g. OSA, IN, etc.), and require signalling translators to allow their introduction in IMS.
- **Resource Elements**, known as MRFs (*Media Resource Functions*), are media source and processing nodes applicable for the provisioning of native advanced multimedia services (e.g. announcement playback, transcoding, filtering, mixing, etc.). They are composed of a control part (MRFC -*Media Resource Function Controller*-), located in the signalling layer, and a media processing engine (MRFP -*Media Resource Function Processor*-) which is part of the media plane.
- Finally, **Interworking Elements** are translators intended for interconnection with foreign-service domains. Here again we may distinguish between a control part (MGCF -*Media Gateway Control Function*-), intended to tackle with signalling-translation issues, and a media-plane part (MGW -*Media Gateway*-), in charge of providing media processing features.

As shown in Figure 1, the IMS session control protocol is SIP (*Session Initiation Protocol*) [6], a modern and extensible protocol for session negotiation and initiation. In fact, although it is an IETF (*Internet Engineering Task Force*) protocol, several SIP extensions have been specifically developed for IMS. In addition to SIP, some other protocols are widely used in IMS, such as Diameter [7], to query user databases and the charging subsystem; COPS (*Common Object Policy Service*) [8], to perform local policy-based admission control; and H.248 [9], for media gateway control.

3. Emulator design

3.1 Goals and strategy

The design goal of the emulator is to provide a standard-compliant IMS testbed (as shown in Figure 1) suitable for the development and functional testing of services (in fact, implemented by Application Servers) by third-parties.

Although UEs (*User Equipments*) are not part of the network and, therefore, they are not considered part of the targeted emulator, their functionality has also been

developed, for operational reasons (UEs are required to perform service tests in the emulator).

Note that the aim is to provide functional reference architecture, not necessarily focused on performance (established sessions per second, hardware resources consumption, etc.), which would be a key factor in production IMS deployments.

Given that building an entire fully standard-complaint IMS from scratch is a hugely complex task, the first required step to undertake its implementation is to split its whole functionality in several *functional elements* and adopt a bottom-up implementation strategy. That is, the IMS emulator will be developed in several steps (*releases*). Starting from the minimal release, successive ones will include one more functional element each, until the definitive complete IMS emulator is achieved.

Three functionality axes have been considered (Figure 2), each one corresponding to one of the IMS main protocols: SIP (13 elements), COPS (1 element) and Diameter (4 elements). As seen in the figure, SIP is the main functional axis (the "longest" one).

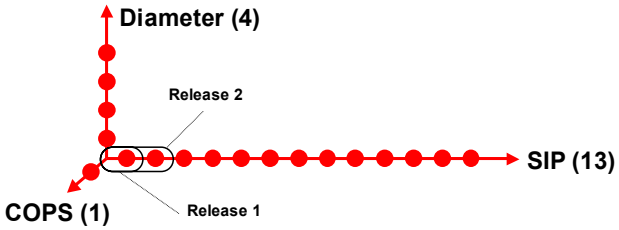

Figure 2: Functionality splitting

In SIP axis the functional elements are:

1. Basic SIP functionality, including the most of the SIP core (RFC 3261), the offer/answer SDP model (RFC 3264) and the "Path" (RFC 3227) and "Service-Route" (RFC 3608) extensions.
2. Preconditions framework (RFC 3112), including the PRACK (RFC 3262) and UPDATE (RFC 3311) methods.
3. Authentication (RFC 3329)
4. Network Event Package, including the SIP event notification framework (RFC 3265) and the specific event package for registrations (RFC 3680)
5. Signalling compression, including the SigCom framework (RFC 3320) and the specific dictionary (RFC 3485) and procedure (RFC 3486) for SIP.
6. Privacy, both mechanism (RFC 3323) and extensions for asserted identities in trusted networks (RFC 3325)
7. Messaging, implementing the MESSAGE method (RFC 3428)
8. 3GPP private extensions covered by RFC 3313 and RFC 3455.
9. Session re-negotiation, part of the SIP core specification (RFC 3261)
10. Session holding and restoration, part of RFC 3264
11. Session redirection, part of the SIP core specification (RFC 3261)
12. Session transfer, implementing the REFER method (RFC 3515)
13. Topology Hiding (THIG), implemented in the I-CSCF

In Diameter axis the functional elements are: Service-Based Local Policy (SBLP) in Gq interface (between P-CSCF and PDF, see Figure 1), HSS access, Offline charging and online charging. Finally, in COPS axis, there is only one functionality: the SBLP Go interface (between the access router and the PDF, see Figure 1).

The first emulator release (release 1) is the simplest, and includes SIP basic functionality only. The second release (release 2) includes the SIP preconditions extension (including the PRACK method [10]).

3.2 Network topology

All the different emulator releases are based on an IPv6 backbone that interconnects all the IMS functional entities (Figure 3). Given this general topology, several configurations can be set: one IMS domain, several IMS domains interconnected through routers, UEs connected directly to the backbone or connected though a QoS-enabled router, etc., allowing thus several testing scenarios.

The proposed emulator does not consider access network aspects. IMS is defined as part of the core network of 3G networks, and therefore the access network is out its scope (although some access requirements are assumed like, for instance, QoS capabilities). In fact, several network accesses have been proposed for IMS usage, like GPRS (in 3GPP networks), CDMA2000 (in 3GPP2 networks) or xDSL (in NGN networks).

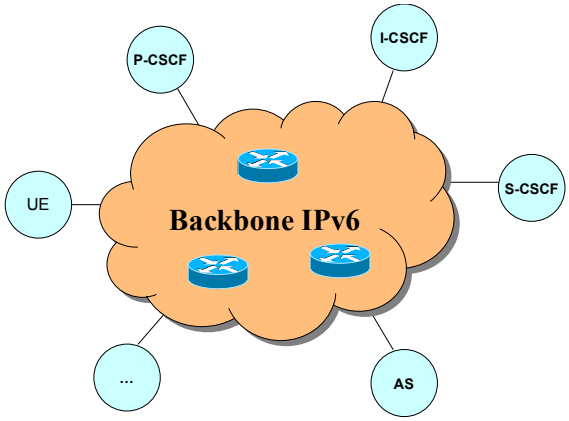

Figure 3: IMS emulator and IPv6 backbone

3.3 Virtualization

This network described in previous section is implemented using virtualization techniques. In particular, we are using VNUML *(Virtual Network User Mode Linux)* technology.

VNUML [3][4] is a general purpose virtualization tool* that brings the

* The tool is freely available (along with compressive tutorials, examples and reference documentation) at

possibility of easily building complex network scenarios using virtual machines, interconnected through virtual networks of precisely specified topologies. These virtual machines are the functional equivalent to real ones, and can run inside just one physical host. VNUML is not a virtualization technique itself, but a front-end to UML (*User Mode Linux*) [11], the actual software that allows running a Linux kernel (using its own resources like memory, disk, process space, etc.) as a conventional process in the physical host (Figure 4).

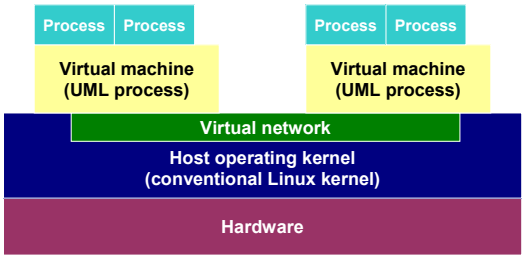

Figure 4: Process space using UML

The value added by VNUML (regarding the plain use of UML) is that the virtual scenario can be specified using a XML (*eXtended Markup Language*) file, that provides an abstract description of the network scenario to be created. Then, the VNUML parser reads the XML specification and sets up automatically the desired scenario, avoiding the need of knowing complex low-level details about UML and setting up the network manually (more time consuming and error prone). Section 4 shows an example of the XML used to build the IMS emulator.

UML (in combination with uml_switch, the related tool to implement virtual networks) allows to set up the capabilities of the virtual machines and virtual networks at booting time (for example, the memory and disk filesystem of each virtual machine, along with the number of network interfaces and how they are interconnected between them). No tool-focused GUIs are required. In addition, UML virtual machines run stand-alone (no management daemons are required) and can be accessed out-of-band and managed from command line. Given these characteristics, UML integration in VNUML parser is quite direct and easy and make it the preferable back-end among other virtualization alternatives (like Xen, Microsoft VServer or VMware).

From the point of view of the IMS emulator application considered in this paper, all the network elements can be implemented using just a physical machine (actually, a conventional PC), with the equipment and management effort saving (it is clear that managing just a PC is easier than managing a network of devices) that this implies. Nevertheless, UEs (that are not actually part of the emulator, as stated in section 3.1) are not implemented with virtual machines, since they usually require user intervention through GUIs and media capture and presentation capabilities. In contrast, they are realized by means of additional physical machines connected to the

---

http://www.dit.upm.es/vnuml

IMS emulator's network.

Using virtualization always implies a performance penalty comparing with the implementation using real equipment (the overhead imposed by the virtual machine and virtual network layers in Figure 4). However, given that the IMS emulator focuses on functionality and not in performance (as stated in section 3.1), this penalty is not a problem from the point of view of the application described in this paper.

3.4 Technologies

This section provides a brief description of the technologies used to implement the emulator and UEs. The whole platform is deployed using three standard PC, in top of an existing network infrastructure with IPv6 support. One of them supports the emulation of the IMS core, with VNUML, while the UEs have been kept in separate machines. The reason is that VNUML launches virtual machines based on Linux, which are suitable for our implementation of the IMS nodes, but would also put some restrictions over the end-user terminals, as well as limiting the performance.

CSCFs are, roughly speaking, SIP proxies with special extensions developed for IMS. However, most SIP proxy implementations existing today are based on IETF SIP core specification, and do not include the particular SIP extensions for IMS. Among several alternatives, we have chosen the well-known SIP Express Router (SER) software [12], which is available as an open-source solution which runs perfectly under Linux and, therefore, with VNUML. Its characteristics make it particularly suitable for our purpose:

- Flexibility: the router can be easily configured at launch time. Using a highly scriptable file, the behaviour of each IMS node (either P-CSCF, I-CSCF or S-CSCF) can be modelled in depth, specifying packet routes or special headers needed in any of them. This allows also for quick differentiation on the nodes, just by specifying a different configuration file
- Extensibility: some IMS-specific functionality is not covered by the default implementation of SER, but it is needed for CSCFs. Taking advantage of this facility, the Path module [13], will be implemented as an extension
- Robustness: SER is consolidated as a quite stable platform in VoIP scenarios, which is of special relevance in scenarios where several of these proxies are to be used, minimizing the chance of error attributable to them

In order to implement the UE, SIP Communicator (based on NIST SIP, the reference implementation for the Java JAIN SIP API recommendation) [14] is used. In addition to SIP functionalities, this client covers functions that, even if not related to IMS itself, are quite necessary from the service usage and testing point of view of using and testing (like the GUI –*Graphic User Interface*– or integration with JMF –*Java Media Framework*– as media stack).

SIP Communicator has the advantage of being open-source, so it can be modified in order to support IMS. As it is programmed in Java, it can run on different operative systems (Windows or Linux), posing no limits for the UE platform.

Regarding Diameter [7] and COPS [8], they imply transversal functionality

common to several network elements: not only CSCF and UE nodes, but also including and HSS and a PDF. Although they are not key functionalities in the first emulator releases (which focus on SIP), several technologies have been considered: OpenDiameter [15], the auth_diameter (which is included as a SER module), and jCOPS (a COPS implementation developed by University of Murcia) [16].

4. Emulator implementation

Release 1.0 and 1.1 of the emulator implement a two-domain IMS environment (Figure 5), that could represent the networks from two different operators. One of them is configured as "domain1.com" and the other as "domain2.com". Only one UE subscriber per domain has been considered, allowing for simplification and eliminating the need of a user registry.

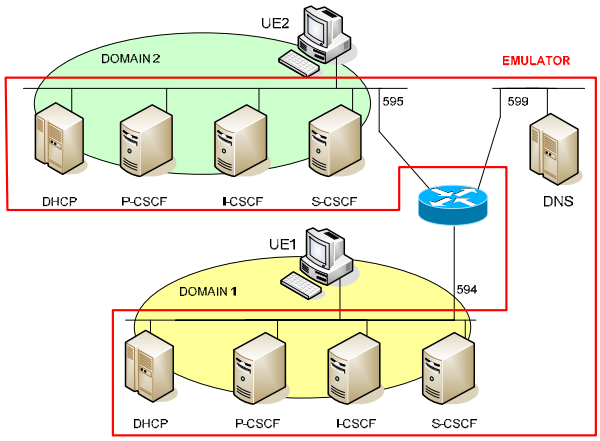

Figure 5: Release 1.1 network topology

Each domain has three different servers, implemented using different SER configurations to obtain the required IMS node functionality (P-CSCF, I-CSCF and S-CSCF), and also a fourth one that acts as a DHCP, enabling autoconfiguration procedures in the network. This independent node could as well implement new functions in future releases of the emulator, such as HSS, SLR or AS. Release 1.1 also implements a DNS for the network, so that queries between nodes are resolved and do not need to be hard-coded. DNS is implemented in a separate domain, so that it is "operator-independent". As noted before, only IPv6 is used for any communication.

While the UEs (and also the router) are physical machines, it is worth noting that all IMS servers (including both DHCP and DNS) are modelled as virtual machines in the VNUML emulation (Figure 6). Network separation is achieved using VLANs in the router, so that domains are actually separated at IP level. The benefits of VNUML usage can be easily understood: simplicity and, moreover, scalability, as more network domains, different operators, or new nodes, can be added just by configuring new VMs in the emulation, and VLANs if needed.

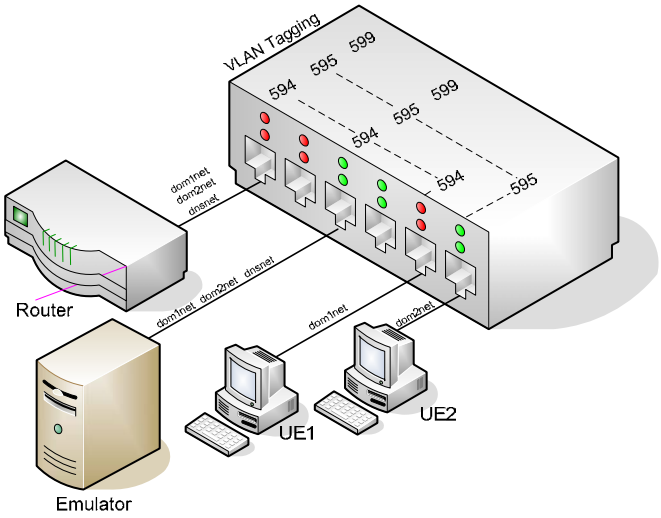

Figure 6: Physical topology of the network

Tests performed so far prove that 17 virtual machines can be run in a typical modern PC (Pentium 4 - 3GHz, 1GB RAM) without noticeable performance overhead. However, even in the case too many virtual machines could stress the same physical devices, scenario can be splitted into to (or more) hosts.

Apart from the configuration of the SER nodes, some implementation work has been completed so far to achieve a closer-to-standard IMS environment:

- Modifications in the UE (SIP Communicator) in order to support source routing through the "Route" SIP header (release 1), and acknowledgement of SIP provisional responses (PRACK method) [10] and session parameter update within early dialogs (UPDATE method) [17] (release 2).
- Development of an extension module for SER implementing the SIP registration of non-adjacent contacts (Path extension header field [13]).
- For validation purposes, several test cases have been considered (UE registration and call establishment between UEs) in the first release. In these tests, message exchange among the different network elements has been recorded (using network monitoring tools, like Ethereal) and compared with the expected messages (specified in the IMS standard [5]). Of course, full videoconferencing sessions can be established between UEs across the emulated IMS network.

A stripped down version of the VNUML XML specification used to build this scenario is shown in Figure 7 (please note that the actual VNUML XML specification includes many parameters -memory assigned to virtual machines, filesystem, etc.- that will be not shown here for the sake of simplicity). Virtual networks are specified in <*net*> tags (the **external** attribute connects the virtual network to a physical interface of the host, the **vlan** attribute specifies which VLAN ID to use) and virtual

machines in <*vm*> tags. The <*if*> tag specifies interface configuration; note that DCHP server configures both IP and MAC addresses while other nodes will receive their IP from the DHCP server automatically. As can be seen, the syntax of the language is quite clear and does not need a complex explanation. Configuration for nodes in domain2 is analogous to that for domain1, using *dom2net*.

```
<vnuml>
  <net name="dom1net" external="eth0" vlan="594"/>
  <net name="dom2net" external="eth0" vlan="595"/>
  <net name="dnsnet" external="eth0" vlan="599"/>
  <vm name="dns">
    <if id="1" net="dnsnet">
      <mac>fe:fd:00:00:00:01</mac>
    </if>
  </vm>
  <vm name="pcscf1">
    <if id="1" net="dom1net ">
      <mac>fe:fd:00:01:01:01</mac>
    </if>
  </vm>
  <vm name="icscf1">
    <if id="1" net="dom1net">
      <mac>fe:fd:00:01:01:02</mac>
    </if>
  </vm>
  <vm name="scscf1">
    <if id="1" net="dom1net">
      <mac>fe:fd:00:01:01:03</mac>
    </if>
  </vm>
  <vm name="dhcp1">
    <if id="1" net="dom1net">
      <mac>fe:fd:00:01:00:01</mac>
      <ipv4>10.0.1.1</ipv4>
    </if>
  </vm>
  […]
</vnuml>
```

Figure 7: IMS emulator VNUML XML

5. Current status and future work

It is quite clear that IMS will become one of the key technologies in telecommunications industry in the next years. In this context, the *Emulator* presented could be a powerful tool in order to consolidate this by providing a reference architecture. In addition, it can be very interesting as development platform for third-party service providers (lowering their development effort and time-to-market and, therefore, increasing the global services offer, which benefits IMS markets).

However, IMS as defined in the standard is a very complex network system. There are two key strategies in order to make its development affordable. First, splitting its functionality in a set of "atomic" functional elements, which will be introduced progressively as building blocks, in order to implement a more comprehensive emulator on each release. Second, using virtualization techniques (VNUML in particular) to easily set the network environment for the emulator. This way, the development effort can be shifted from the management and configuration of the network backend to the implementation of functionalities. It has to be taken into account that the main interest of the emulator relies on its functionality: the network backend supporting this functionality is of course needed, but not the target itself.

Currently, the release 1.1 is finished with a SIP basic functionality and the work is focused on release 2. Next steps will consist on the design and implementation of new releases, including new functionality, towards the consecution of a fully standard-complaint IMS emulator. Each of these releases, fully functional within its target feature set, is intended for the validation and testing of several IMS-based services and solutions, such as advanced centralised multipoint conferencing services

or seamless mobility across administrative domains.

ACKNOWLEDGMENT

The work presented in this paper is partially based on the results of the SAMURAI project (*Servicios y Aplicaciones Móviles de video sobre UMTS y Redes Avanzadas IMS*, Mobile-based video applications and services in UMTS and advanced IMS networks), funded by the PROFIT initiative of the Spanish National R&D programme.

References

[1] Third Generation Partnership Project, http://www.3gpp.org, last checked April 11th , 2006

[2] Third Generation Partnership Project 2, http://www.3gpp2.org, last checked April 11th , 2006

[3] D. Fernández, F. Galán, T. de Miguel, "Study and emulation of IPv6 Internet exchange (IX) based addressing models", *IEEE Communications Magazine*, vol. 42(1), pp. 105-112, Jan 2004

[4] F. Galán, D. Fernández, J. Ruiz, O. Walid, T. de Miguel, "A virtualization tool in computer network laboratories" in *5th International Conference on Information Technology Based Higher Education and Training*, Istanbul, Turkey, 2004

[5] 3GPP TS 23.228 "IP Multimedia Subsystem (IMS) (Stage 2) v7.2.0", Dec 2005

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

[7] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, J. Arkko, "Diameter base protocol", RFC 3588, IETF, Sep 2003.

[8] D. Durham et al, "The COPS (Common Open Policy Service) protocol", RFC 2748, IETF, Jan 2000

[9] "Gateway control protocol version 1", Recommendation H.248.1, ITU-T, Mar 2002

[10] J. Rosenberg, H. Schulzrinne, "Reliability of provisional responses in the Session Initiation Protocol (SIP)", RFC 3262, IETF, Jun 2002.

[11] J. Dike, "User Mode Linux" in *5th Annual Linux Showcase & Conf.*, Oakland CA, 2001

[12] SIP Express Router, http://www.iptel.org/ser/, last checked April 11th , 2006

[13] D. Willis, B. Hoeneisen, "Session Initiation Protocol (SIP) extension header field for registering non-adjacent contacts", RFC 3327, IETF, Dec 2002

[14] Sip Communicator, https://sip-communicator.dev.java.net/, last checked April 11th , 2006

[15] OpenDiameter Web Site, http://www.opendiameter.org/, last checked April 11th , 2006

[16] F. J. García, G. Martínez, A. F. Gómez, "An XML-Seamless Policy Based Management Framework" in *Third International Workshop on Mathematical Methods, Models, and Architectures for Computer Network Security*, St. Petersburg, Russia, 2005.

[17] J. Rosenberg, "The Session Initiation Protocol (SIP) UPDATE method", RFC 3311, IETF, Sep 2002