Chapter – 4
OpenIMS Implementation

Chapter – 4
OpenIMS Implementation

4.1 Introduction to Open IMS

The IMS playground developed at FOKUS [83] is an open technology test field. This playground works to implement existing and new IMS standards; all major FOKUS implemented IMS components, i.e. CSCFs, HSS, MG (Media Gateway), MRF, Application Servers and so on, and integrate them into a single environment. It also provides different service platforms, such as "Open Service Access (OSA)/Parlay, JAIN Service Logic Execution Environment (SLEE), Web services/Parlay X, SIP Servlets, Call Processing Language (CPL)". [84]

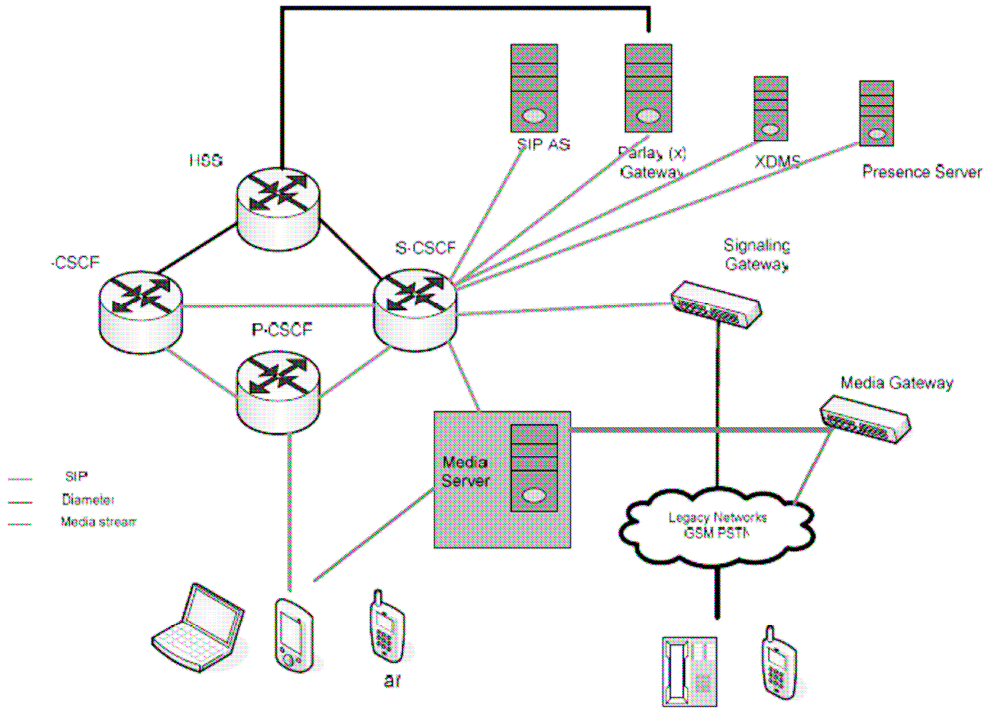

Figure 4.1 Overview of IMS playground@FOKUS [85]

Open Source IMS Core System [85] is an IP Multimedia System for test. The Fraunhofer Institute FOKUS created it. They point out that this Open Source IMS Core System is not intended to become or act as a product in a commercial context. Its sole purpose is to provide an IMS core reference implementation for IMS technology testing and IMS application prototyping for research purposes, typically performed in IMS test-beds. This target has also motivated the decision to use open source software (i.e. SER based on GPL).

Since IMS has already been used gradually and much efforts has been put forwards it, the main efforts now is for developing services. While there are already many Open Source projects established in the plain VoIP area for SIP clients, proxies, stacks and tools around the IETF sip standards, there are currently practically no Open Source projects with specific focus on the IMS.

The Fraunhofer Institute FOKUS focuses on Open Source software, which is a flexible and extendable solution.

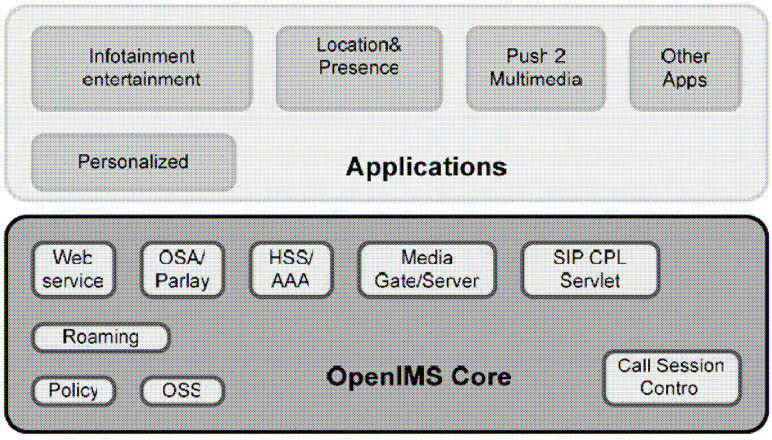

Figure 4.2 OpenIMS testbed at FOKUS [84]

The following figure 4.3 that some of the key components are the same in both IMS and OpenIMS, but some are missing in OpenIMS, and the SIP2IMS gateway only exist in OpenIMS.

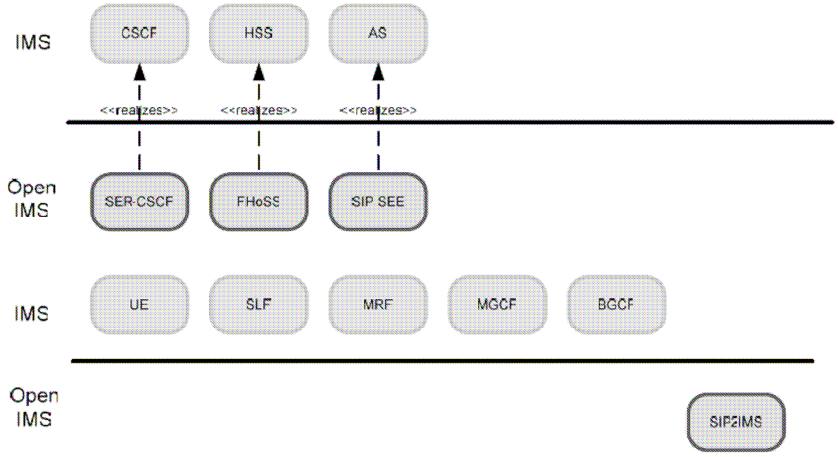

Figure 4.3 Comparisons of IMS and OpenIMS

4.1.1 Asterisk

Asterisk is an open source software implementation of a private branch exchange (PBX), which was created by Mark Spencer of Digium. [86], Asterisk

[http://www.asterisk.org] runs on OpenBSD, FreeBSD, Mac OS X and Sun Solaris. It supports Voice over IP protocols, such as SIP, H.323, IAX (inter-Asterisk exchange) and MGCP (Media Gateway Control Protocol). Thus, Asterisk can interoperate with many SIP telephones and Asterisk PBXes. Asterisk contains many features including voice mail, conference calling, interactive voice response and automatic call distribution. All these made Asterisk a very popular software implementation of PBX.

4.2 Applications over OpenIMS

In this section the discussion for most significant services that will be provided by IMS.

A. Presence Service in the OpenIMS

For presence service, the clients can send information that shows their status to the server and the server will inform availabilities or willingness of those clients to other users in the group.

Presentities choose what information they want to publish, and when watchers get the information, they decide how and when to communicate with the presentities. What's more, not only end-users but also other services can get the presence information. 'For example, an answering machine server is interested in knowing when the user is online to send them an instant message announcing that they have pending voicemails stored in the server'. [87] Therefore, the presence service is to be considered as the foundation of all the services.

As demonstrated in Figure 4.3 'depicts the presence IMS architecture that maps the already defined roles in presence to existing functional entities in the IMS.' [87] In IMS, terminal has two roles: watcher and Presence User Agent (PUA) refer the Presence Agent (PA) as a Presence Service (PS), and implemented Resource List Server (RLS) as Application Server (AS).

From the figure it is predict that most of the interfaces are still exist in IMS SIP or Diameter interfaces, thought they change their name to start with 'P'. Compared to 3GPP the IMS architecture, there are 2 new interfaces in presence IMS architecture, which are Pen interface and Ut interface. Pen interface allows an AS work as a PUA, and Ut interface can be used between any AS and IMS terminal. [87]

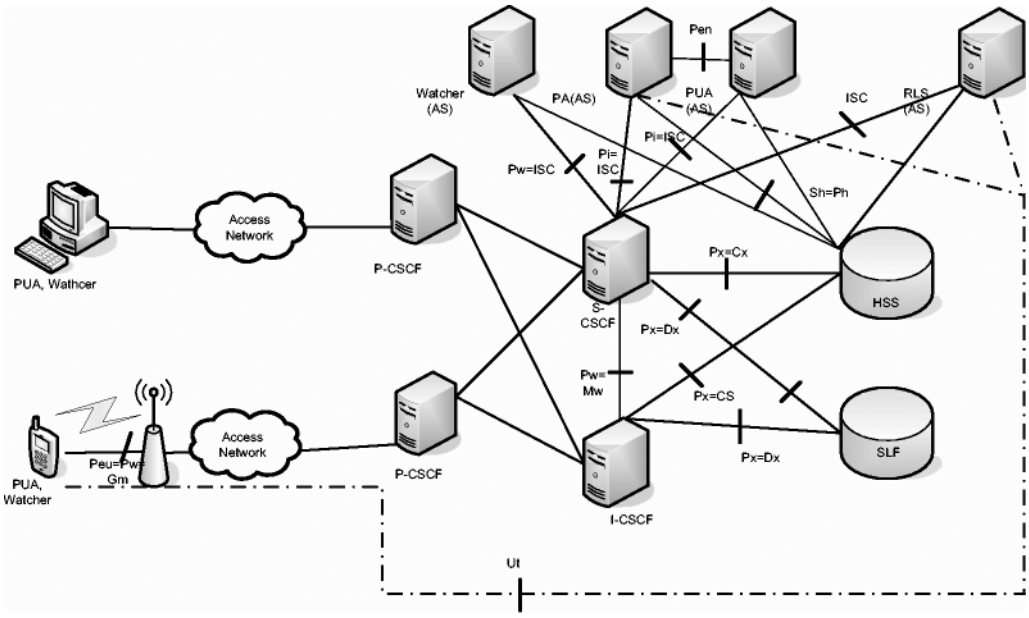

Figure 4.4: SIP-based presence architecture in the IMS [87]

B. Instant Messaging

A group of people can communicate with each other over a network, instant messaging is one of the ways to achieve. A user can choose targets to send messages to, depending on the presence status of other users.

If the instant messages are stand-alone messages then they will be sent in pager-mode, and if the messages belong to some existing session, they will be sent in Session-mode. [87]

C. SmartMessenger

The SmartMessenger 'is a multi-channel message delivery service.' [88] Users can communicate with each other by sending SMS and instant messages through the Parlay interface.

The SmartMessenger is also a Click2Dial application. A Text-to-Speech (TTS) on the phone can read messages, and the SmartMessenger uses the Call Control interface to make calls between VoIP and PSTN. [88]

D. IMS Push-to-Talk

Push to talk (PTT) provides one-to-one and one-to-many high-margin voice service for both business users and private consumers. IMS Push to talk is a good example of PIT, and includes key features of PTT, such as presence, one-to-many communication. It also has some particular services, for example, filtering of incoming calls or do-not-disturb status. Those services make IMS PTT a more intelligent and convenient service, and at the same time, it is easy to use, so IMS

PTT has a large market with a variety of users.

Ericsson IMS Push to talk solution is based on the Open Mobile Alliance, Push to Talk over Cellular (OMA PoC) standard, which defines a rich set of features such as presence and over-the-air provisioning. [89]

4.3 Related works: OpenIMS Implementation

During OpenIMS implementation and modeling task [90] is the most important reference for us. State-Of-the-Art Enterprise SIP Solutions were identified first. And then, they design a solution for SIP/VolP enterprise. The final problem involves research on migrating enterprise SIP/VOIP solutions towards IMS. In order to approve their concept and research based on the SIP technology, they established the "HiA-Teleca SIP/VOIP test-bed". It is now a common research platform for HiA and auSystems (from Teleca) [91] to build and develop competence in VOIP/SIP/IMS area. It will contribute development in future "All-IP" products and services. The test-bed supports SIP session handling, IPvoice, voicemail, conference calls, PC software and SIP hardware clients, PSTN connectivity via a SIP ISP, presence, TAPI for Microsoft Office integration. This part is very useful for us, because it is also need to implement solution basic on the test-bed with Open Source IMS core.

Except design a SIP/VolP enterprise solution, this paper also involves finding the way to migrate the existing SIP/VolP solution towards IMS. Based on their solution, they find four possible ways or enterprise SIP/VolP solutions to integrate legacy and future cellular phones using IMS.

Enterprise SIP/VolP to IMS migration alternatives is:

1. Forking is the simplest solution and has no requirement on clients or servers.
2. The client-based solution will give access to more features but requires an advanced SIP client and more configuration knowledge.
3. The presence solution is a network-based solution. It handles all necessary location updates without involving the client. It supports legacy phones.
4. Link registration solution is network based, too. It directly links the registration procedure by using "subscriber registration update service" provided by IMS-enabled operator. It may appear in the later phase of IMS migration.

4.3.1 Installation and Configuration of an OpenIMS

Since auSystems and HiA [91] have already established a research testbed for Enterprise VOIP solutions based on Open Source systems which contains all the necessary functions of a typical enterprise installation. As a starting point, plan to extend the existing testbed with a new IMS sub-domain based on the "Open Source IMS Core", so that the environment of OpenlMS to test and evaluate how

it works for modeling implementation and each service provision over this environment.

Considering the State-of-the-Art IMS Architecture, basic understanding about set up this testbed; need to do some installations and configurations of the key components of OpenIMS are FHoSS, CSCFs, SIP2IMS, etc. And with the leading of Installation-Guide, implement it step by step. The figure 4.5 shows the main steps of establishing OpenIMS Testbed:

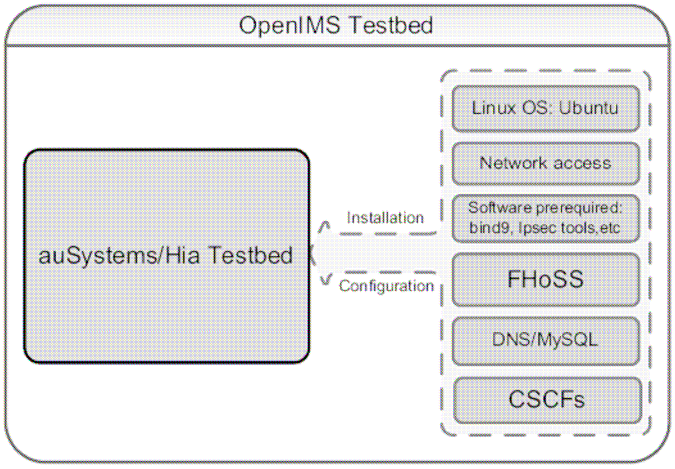

Figure: 4.5 Setting up of OpenIMS testbed

4.3.2 Installation of Prerequisites

As the figure 4.5 shows, before installing the key components of OpenIMS, the testbed requires prerequisites in hardware, network and software. For the environment of hardware, need a current Linux desktop class machine. Here, Ubuntu 6.10 [source http://www.ubuntu.com/] as Project Linux Operating System. Besides, in order to get ultimate performance, it is need to add several gigabytes of RAM and as many CPUs/Cores as needed.

The software should include the following requirements:

In order to ensure the system will work well, 100 MBytes of disk space are needed. Subversion is required to be installed so that could find fresh code. GCC3/4, JDK1.5, ant that uses for Java development is needed to be installed firstly. In this case, use MySQL as a database management system (DBMS), which is supported by FHoSS, I-CSCF and other functions that require a DBMS. Developed Iibxml2 and libmysql are required. Linux kernel 2.6 and ipsec-tools, which is used to setkey are needed to use IPSec SAs. Besides, the bind 9 used as name server of this project. And Firefox as project browser.

After make sure that all the requirements have already fulfilled, begin to install and configure the FHoSS, which is the core component of OpenIMS.

4.3.3 Installation and Configuration of FHoSS

The Open Source IMS Core would be incomplete without a Home Subscriber Server. FOKUS developed its own prototype HSS, the FOKUS HSS (FHoSS) [92] which is entirely written in Java and based upon Open Source software. As its purpose in the Open Source IMS Core is that of a database, the FHoSS is targeted mainly towards conformance rather than performance. It is mostly the glue between a Database Management System and the Diameter interfaces with the CSCFs and IMS application layer.

FHoSS stores the IMS user profiles and provides the location information of the user. Additionally, it is also a web-based management console. The architecture of FHoSS below illustrates the main components and the interfaces that are used in.

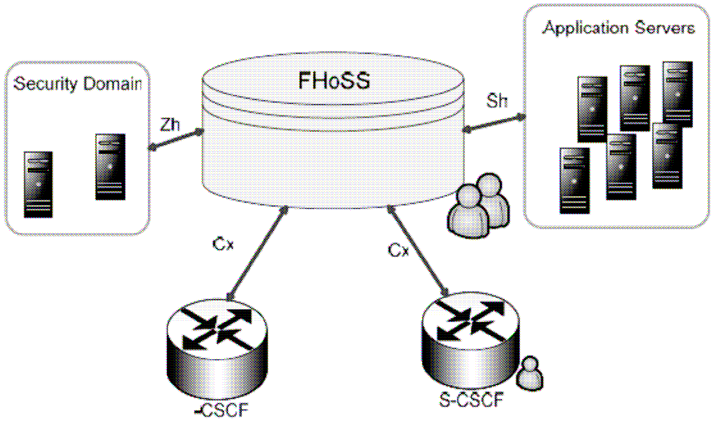

Figure 4.6: Architecture of FHoSS

From the figure 4.6, the entities that communicate with the FHoSS are the application server (AS) that hosts and services in the IMS environment, security domain and the Call State Control Function servers (CSCF). And the interface layer describes the external behavior of the HSS. FHoSS stores the user files via Sh interface point to Application Servers, and it communicates with CSCFs via Cx interface. Besides, it connects with Security Domain via Zh interface.

It extends the Open Source project 'Open Diameter' and implements the complex functionalities of the Cx and the Sh reference points based on Java. [88] The FHoSS stores the IMS user profiles along with authentication and authorization information and provides user status information along with a notification service via the Sh reference point to Application Servers. Additionally, it supplies S-CSCFs with the current filtering information on a user base over the Cx reference point.

4.3.3 1 Interface layer

The core of the FHoSS is the HssDiameterStack. It uses the DiameterPeer to send requests to other entities and retrieves the requests and responses via

CommandListener. There are three interfaces used in FHoSS, who are Sh, Cx, and Zh. They can be found in the de.fhg.fokus.cx, de.fhg.fokus.sh and the de.fhg.fokus.zh package. For each interface there is a direct implementation. This implementation can be found in the de.fhg.fokus.hss.server and subsequent packages. For every method of the interface there is a related operation class in the op packages.

These operations will be called by the interface implementations and will call in turn the diameter commands. Same as mention in the 3gpp specification [1], [2], every interface method is mapped to a number of Diameter requests. These requests are realized by implementing the CommandAction and CommandListener classes. Particular action for every command which is sent through the diameter peer, in the de.fhg.fokus.hss.diam.cx, de.fhg.fokus.hss.diam.sh and de.fhg.fokus.hss.diam.zh packages. For every command, which will be received by the Hss, one listener exists. The command listener will dispatch the requests to the interface methods.

4.3.3.2 Data Access Layer (DAL)

The operational data of the FHoSS is stored in a database. The Hibernate persistence framework was used to build a data access layer that is used to change the database system. The related data classes can be found in the de.fhg.fokus.hss.model package.

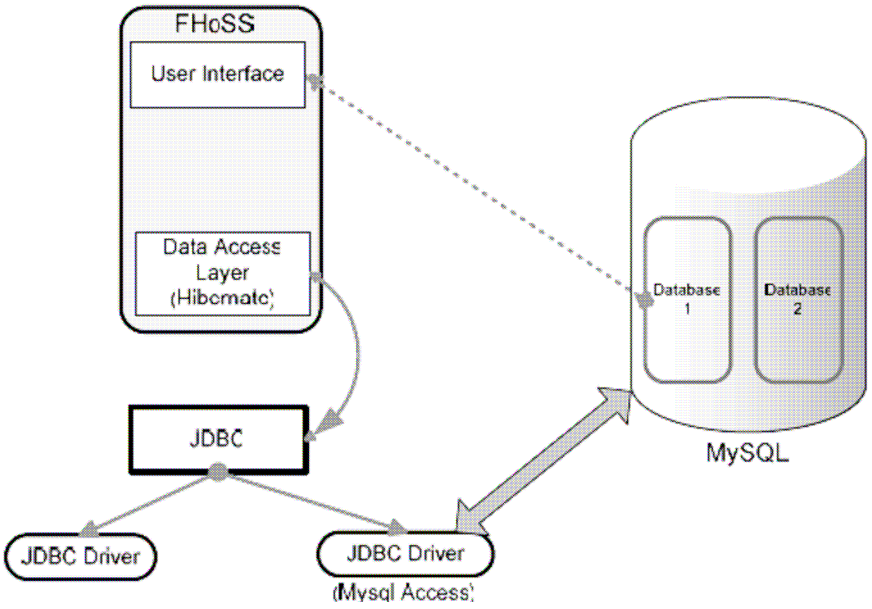

Figure 4.7: Structure of databases

The user data is kept inside a MySQL database. However, there is a Data Access Layer (DAL), which is based on JDBC, and then any database can be used as long as have a JDBC-driver for it. In this configuration, use of MySQL as the

back-end database engine. Since MySQL is far from optimal for this database, other database LDAP also consider for same evaluation, some checking and there actually exists a JDBC-driver for it, and Hibernate supports for LDAP. Therefore, in theory it should be possible to also use LDAP here www.openldap.0rg/idbcldap/1.

4.3.3.3 GUI

To manage and maintain the FHoSS, a web based management interface is provided. This provides a clear structure and separation of logic and GUI related tasks. The implementation of the GUI logic can be found at de.fhg.fokus.hss.form and package de.fhg.fokus.hss.action. The rendering is done by several Java Server Pages, which can be found in the src-web folder.

4. 4 Installation FHoSS

4.4.1 Get source code

As a start, fresh code http://svn.berlios.de/svnroot/repos/openimscore where source code of FHoSS/trunk available. The source code is pre-configured to work from a standard file path. Firstly, create the directory 'OpenIMSCore' in 'opt' and go there. After that create a new directory 'FHoSS' and check if the HSS is there:

```
mkdir /opt/OpenIMSCore
cd /opt/OpenIMSCore
mkdir FHoSS
svn checkout http://svn.berlios.de/svnroot/repos/openimscore/FHoSS/trunk FHoSS
```

4.4.2 Compile

Before compilation, must make sure have a JDK >=1.5. And then, use Ant to build and install the FHoSS. To build the FHoSS must first execute the gen target, to generate the data classes.

```
ant gen
```

After that use compile to build the binaries

```
ant compile
```

Installation is also done using ant. With the deploy target install the FHoSS

```
ant deploy
```

4.4.3 Configure the environment

Database in order to use the FhoSS, can find two sql scripts for the MySQL database in the root directory of this installation. Use these scripts to create a new MySQL database and to populate it with default values.

- To create the database and the tables:

```
mysql -u admin -p <hssdb.sql
```

- To create two demo users and initial values for service profiles:

```
mysql -u admin -p <userdata.sql
```

- After creating the database, check that the database isin there and accessible.

4.4.4 Configure the IMS core

By now MySQL is working and need to take a look at the configuration files in FHoSS/deploy/, these files and modify some parameters as appropriate for this installation. Configure the required configuration files located in the root of the deployment directory. [91]

o **"DiameterPeerHSS.xml":** modify the peer configuration here: like the FQDN, Realm, Acceptor Port or Authorized identifiers.

o **"hibernate.properties":** configure are the main properties for hibernate; implicitly is configured to connect to the mysql on the localhost (127.0.0.1:3306). The most relevant properties are:

o hibernate.connection.url=jdbc:mysql://127.0.0.1:3306/hssdb

o hibernate.connection.username=hss hibernate.connection.password=hss

- **"hss.properties":** Specify configuration like: on which address the tomcat is listening (e.g. host=localhost) and the relative path of the web interface of the FHoSS. (e.g. appPath=/hss.web.console). Other parameters like: operatorId, amfId or defaultPsiImpi can be specified here.

- "log4j.properties": Contains configuration for the logger. The most relevant things here are the output file path of the logger and the level of logging.

4.5 Installation and Configuration of CSCFs

As the extension of IMS, OpenIMS CSCFs use SER (SIP Express Router) to perform all CSCFs. SER is a worldwide-recognized SIP reference implementation. SER can be used as SIP registrar, proxy or redirect server and is capable of handling many thousands of calls per second [52]. OpenIMS CSCFs are required to maintain as much of SER's performance as possible. So that OpenIMS could

have a flexible exploitation. Open IMS CSCFs it consists of four main components: P-CSCF, I-CSCF, S-CSCF, and SIP2IMS Gateway. The figure 4.7 illustrates how the main components are located and the interfaces that are used in them.

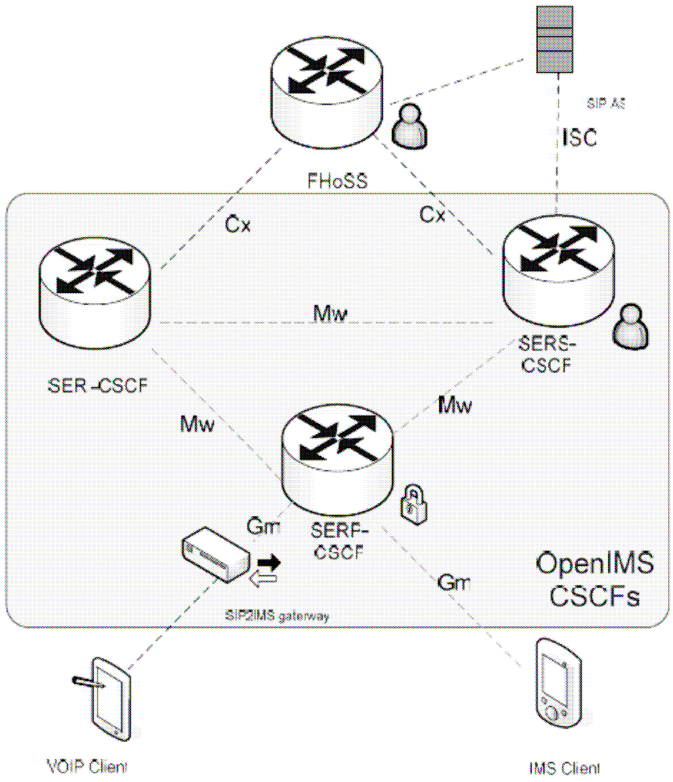

Figure 4.8: Architecture of OpenIMS CSCFs [93]

From the Figure 4.8, it is obvious that the P-CSCF is the initial entity from the mobile terminals to OpenIMS through a SIP2IMS gateway. P-CSCF uses Gm interface to exchange messages between user endpoint (UE) and CSCFs. SIP affords this process. And only the registered endpoints can insert the message to IMS. I-CSCF and S-CSCF are located in Home Domain. Both of them connect with HSS using Diameter over Cx interface. As to the communication of P-CSCF, I-CSCF and S-CSCF, SIP is used to exchange messages over Gm interface. The general flow is as follows: the User Agent, who wants to access OpenIMS, should be registered through P-CSCF; and then, P-CSCF finds the related Home Domain and also sends the message to I-CSCF; I-CSCF forwards to HSS to select the right S-CSCF.

4.5.1 Modules of OpenIMS CSCFs

The modules in CSCFs can parallel process and supplementary state information can be kept. Besides, there is a special focus towards scalability for both load distribution and data quantity.

The main modules of OpenIMS CSCFs are as following:

The CDiameterPeer Module (cdp)

This module is used for realm routing, specify each time the FQDN (Fully Qualified Domain Name) of the destination host when it is used. It is supposed to allow efficient Diameter communication to and from SER.

The IMS Service Control Module (isc)

OpenIMS and Interoperability with Asterisk/Sip Express VOIP Enterprise Solutions ADC & auSystems isc module is supposed to provide support for the ISC interface between the Serving-CSCF and the Application Servers. To use it, need the Serving-CSCF Module (scscf) loaded because it uses the registrar in there for Initial Filter Criteria storage.

The Proxy-CSCF Module (pcscf)

pcscf module is supposed to provide the functionality required for an Proxy-CSCF.

The Interrogating-CSCF Module (icscf)

The icscf module is supposed to provide the functionality required for an Interrogating-CSCF. Need the cpd module loaded to use it. This is because this module communicates using Diameter with the Home Subscriber Server over the IMS_Cx interface.

The Serving-CSCF Module (scscf)

This module is supposed to provide the functionality required for a Serving-CSCF. It is the same as icscf module. To use it, need the cpd module loaded. The reason is it communicates using Diameter with the Home Subscriber Server over the IMS_Cx interface.

The SIP-to-IMS Gateway Module (sip2ims)

It provides NAT Helper for SIP clients. This module is supposed to provide the translation capabilities to enable normal SIP User Endpoints to use the Open IMS Core through the Proxy-CSCF.

To accelerate testing and to integrate with SIP UEs and test tools, a gateway that helps SIP traffic to work in an IMS environment was required. The SIP-to-IMS gateway performs this adaptation tasks. At the moment it translates between MD5 and AKAv1-MD5 authentications and helps with special headers. The SIP2IMS Gateway can be been regarded as the helper module of the Open

source IMS Core project to allow the verification of services with current state-of-the art VoIP clients. Yet, due to its gateway nature, it filters much of the IMS advantages and should not be regarded as a full-blown IMS solution.

The Open Source IMS SIP2IMS Gateway allows transformation of IETF SIP messages to IMS conformant messages. It translates MD5 authentication to IMS AKA authentication. SIP2IMS can also enable developers to access core elements and to trial multimedia services by using a non-IMS VoIP client.

4.5.2 Installation steps for CSCF

A. Get the Source Code

On the same page http://svn.berlios.de/svnroot/repos/openimscore where the code of HSS, need to get the code of CSCFs - ser_ims/trunk. The source code is pre-configured to work from a standard file path. The directory /opt/OpenIMSCore, and now need to create new directory ser_ims under it, using 'mkdir' command. After that, need to check if the CSCFs are there.

```
mkdir ser_ims
svn checkout http://svn.berlios.de/svnroot/repos/openimscore/ser_ims/trunk ser_ims
```

B. Compile

For this step, make libs install all in ser_ims. This step takes some time to finish and promise that all the prerequisites had been installed.

```
cd ser_ims
make install-libs all
cd..
```

C. Configure the Environment

All the installation examples are configured to work only on the local loop back and the default domain configured as "open-ims.test". Therefore, replace 127.0.0.1 with this IP address and replace the home domain with this own one. Additionally, DNS needs to be configured in this step as well. Find a sample DNS zone file and copy it to this bind configuration directory. And then, edit named.conf and insert the file there. After that, restart the name server to test if the names are resolvable. It is needed to also install icscf.sql and sip2ims.sql into MySQL in this step. Checking out if the databases are in there and accessible cannot be forgotten.

```
mysql -u root -p -h localhost < ser_ims/cfg/icscf.sql
```

D. Configure the IMS Core

By now, MySQL and DNS can work well. To configure the IMS Core, just need to copy several files to /opt/OpenIMSCore. The files that need to be copied are: pcscf.cfg, pcscf.sh, icscf.cfg, icscf.xml, icscf.sh, scscf.cfg, scscf.xml, scscf.sh, sip2ims.cfg, sip2ims.sh. Need to edit these files to customized preferences.

```
cp ser_ims/cfg/*.cfg.
cp ser_ims/cfg/*.xml
cp ser_ims/cfg/*.sh .
```

4.5.3 Start the Components

The next step after installation is to start each module of CSCFs: pcscf.sh, icscf.sh, scscf.sh and sip2ims.sh at the same time. Besides, startup.sh is also used to start the FHoSS on a Linux/UNIX system. Note whether if the JAVAJHOME variable is set correctly.

```
./pcscf..sh
./icscf.sh
./scscf.sh
./sip2ims.sh
./startup.sh
```

Make sure that each component can connect to HSS. And in this process, observe periodical log messages with the content of the registrar and with the opened diameter links by default.

After that, need to check the web interface on http://localhost:8080/

4.5.4 Configure Subscribers

Access the management console using a web browser on http://localhost:8080/hss.web.console. The operator can enter in this website as 'hssAdmin' to read and manage the console. But the 'hss' can be used just as a reader. Both of the two users have the same code, which has already been given. [92]

After login in as 'hssAdmin', more information can be found. FHoSS comes provisioned with a couple of sample users by default. One user is 'Userdemo2', whose Public User Identity is: Userdemo2@open-ims.test. And the other is 'Userdemo1', whose Public User Identity is: Userdemo1@open-ims.test. In

addition to this obvious information, some concealed information can be read as well, such as each users' Private User Identities. With this information, create a new subscriber and its Private User Identity and Public User Identity. There are some buttons to click on. The new subscribers that created are 'user1' and 'user2'. It is easy to create new users by using FHoSS User Profiles. The figure 4.9 shows how the web-interface looks.

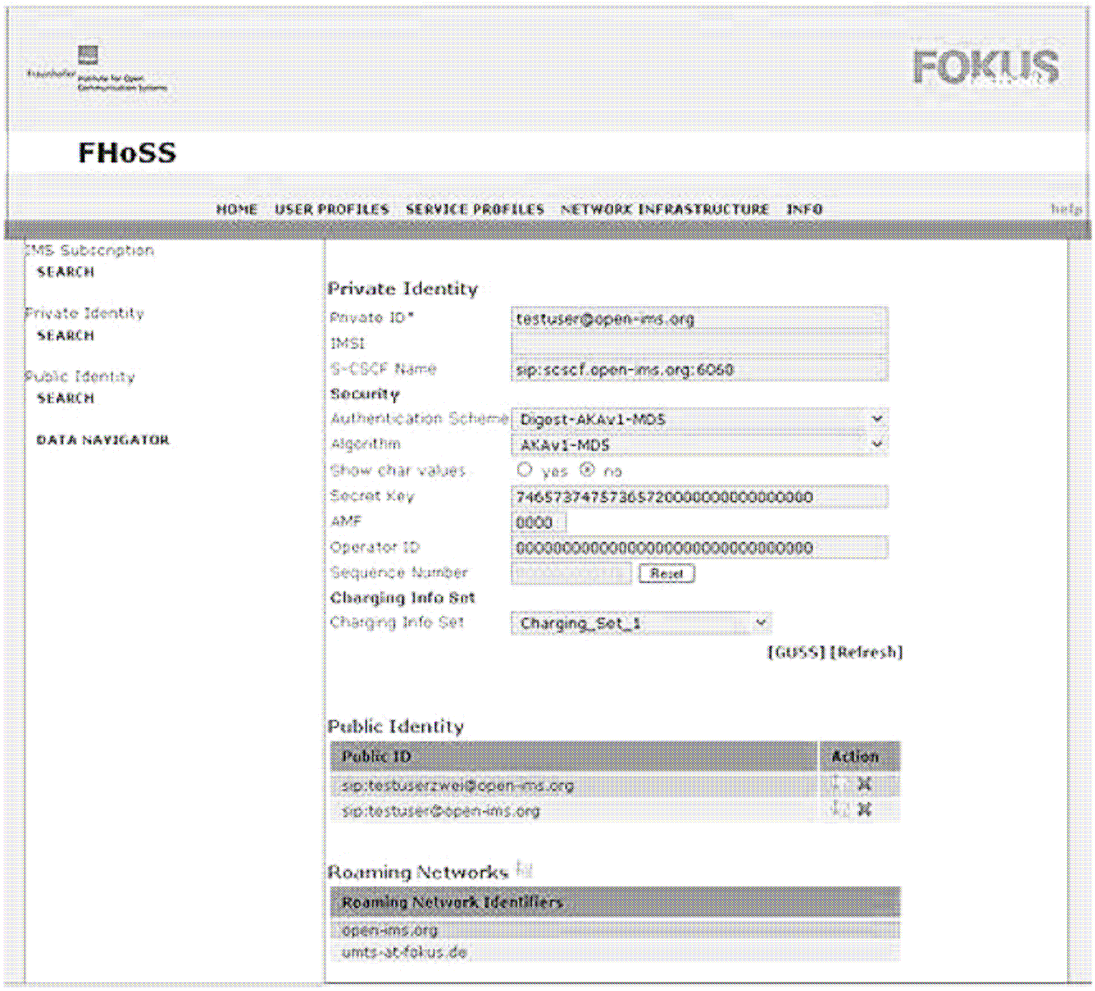

Figure 4.9 FHoSS User Profile

As it is mentioned in State-of-the-art, each IMS User is assigned at least one Private and one or more Public User Identities. IMS Private User Identity uses the format of NAI, which is:

user@operator.com

And the Public User Identity uses the format of SIP URI or TEL URI that can be presented as:

sip:useradmin@imstestbed.net or tel: +912812443300

Based on this, when create the Private User Identity for the new subscriber, followed NAI format which is: user2@open-ims.test and the Public User Identity is: sip: user2@open-ims.test.

In order to test, may add more Public User Identities for each subscriber after finish the whole installation. And may try the situation extended in 3GPP R6 that is a subscriber is assigned more than one Private User Identities and one Public User Identities can be used in for Private User Identities.

This step is continued with configuration Subscribers of FhoSS, have already created a new Subscriber and its Private User Identity and its Public User Identities. And now find that the SIP-to-IMS Gateway comes provisioned with the same couple of sample users. One is Userdemo2@open-ims.test. The other is Userdemo1@open-ims.test could use these in the sip2ims.credentials mysql table at first. And insert new ones.

The details of the new subscribers' configuration:

User2:

Private Identity: user1@open-ims.test

Secret Key: user1

OP: 0x00...0

AMF: 0x00...0

Use of Anonimity Key: enable

Public Identity 1: sip: user1@open-ims.test

Public Identity 2: tel: +912812443300

User1:

Private Identity: user2@open-ims.test

Secret Key: user2

OP: 0x00...0

AMF: 0x00...0

Use of Anonimity Key: enable

Public Identity 1: sip:user2@open-ims.test

4.6 Functionality evaluations of OpenIMS

Basic on the OpenIMS Testbed that already established in the previous section, in this section evaluate each component of OpenIMS conforms to the 3GPP specifications. For that some experiments to make sure all the necessary

functionality will be available to us. For that select the SIP Client or the IMS Client firstly, and then use these to make call session connected with OpenIMS to analyze the log message showing in Wireshark [94]. Use a table to show if the functionalities conform to 3GPP Release 6. Choose 3GPP Release 6 as the OpenIMS specification, and compare all functionalities of OpenIMS with it.

4.6.1 User Equipment

As the prerequisites, start testing with configuration of several SIP/IMS Clients. Select for install several SIP/IMS clients to find one that can support all functions and also are easy to operate.

4.6.1.1 SIPHardphone

A SIP Hardphone is an IP hardphone using SIP Protocol to register as a user. For that Grandstream GXP-2000 used in Test Bed, Grandstream GXP-2000, was the winner of Internet telephone in 2005, as one of the best SIP Clients.

4.6.1.2 SIP Softphone: X-Lite

X-Lite [95] is CounterPath's next-generation SIP based softphone. It works over IP-based system. Users can use it within an enterprise LAN or VoIP service provider network.

Figure 4.10 X-Lite softphone

As other softphones, X-Lite has all the standard telephone features, including two lines, Mute, Do not disturb, Three-way audio and video conferencing and so on. However, it also has some improved features and functions make X-Lite a

popular softphone on the market.

- Instant messaging and presence using the SIMPLE protocol
- "Zero touch" configuration
- "Zero touch" detection to detect the bandwidth that a user's computer can access Support the RFC 3261 SIP standard, support for H.263, H.263 1998

4.6.1.3 UCT IMS Client

The UCT IMS Client [96] is a soft-phone designed especially for to use together with the Fraunhofer FOKUS Open IMS Core. It is run on a Linux-based operating system, and is very easy to configure and use. It support the AKA algorithm, support instant messages, the current version also support voice calls. Because it is still under improvement, there are some known bugs and the user interface is also very simple.

4.6.2 Configuration of SIP/IMS Client

As there are two defaults IMS UE by FHoSS, use the existing UEs' information to configure SIP UE so that they can register by OpenlMS server successfully.

Configuration files of SIP UE

Userdemo2:
User part of the SIP URI: Userdemo2
Host part of the SIP URI/Domain/realm: open-ims.test
Password: Userdemo2
Strict Outbound Proxy: sip:pcscf.open-ims.test:4060

The main components that need to evaluate are at least: P-CSCF, I-CSCF, S-CSCF, FHoSS, SIP2IMS and the interfaces between them (Cx, Gm, Wm). Considering that the testing case could be quite complicated because of involving so many components, for the next phase, start experiment with 'Registration at the IMS-level'.

4.6.2.1 Configuration of X-Lite Softphone

Installed X-Lite 3.0 in one computer, and use "user2" which is into the FHoSS to register. First add a new account before using X-Lite to place or receive calls. Click "SIP Account Settings" in the menu and select the Enable checkbox for the desired account.

Figure 4.11 Setting up Account

4.6.2.2 Configuration of UCT IMS client

The UCT IMS Client 1.0.3 in the Linux OS/Ubuntu as the IMS client. After installing, used the "make" command from the src directory to compile, and then "./uctimsclient" is used to run the software. It's much easier for us to use it because two defaulted users "Userdemo2" and "Userdemo1" have already been configured in it.

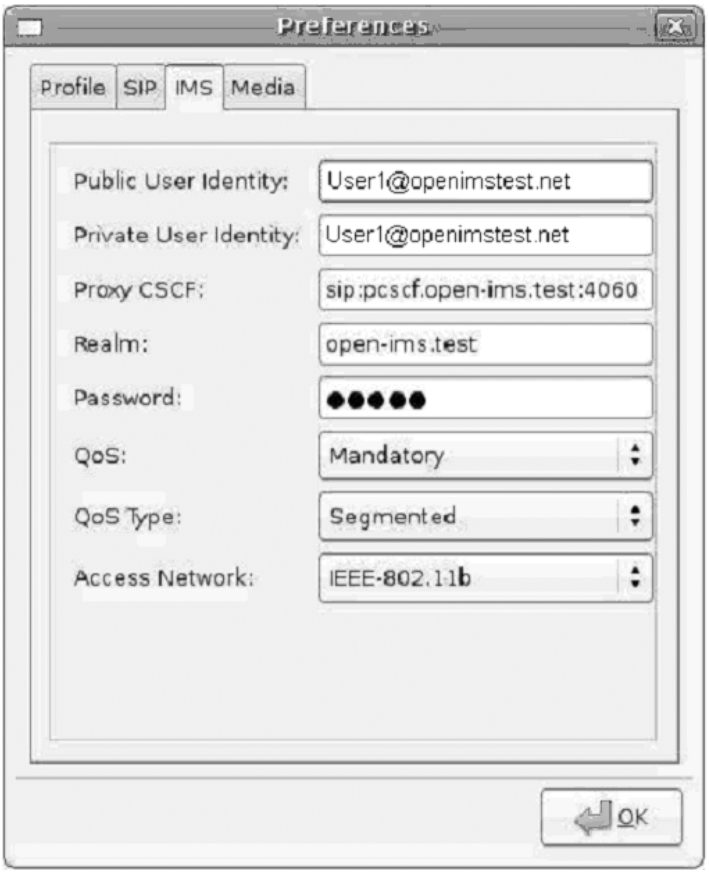

Figure 4-12 Preferences of UCT IMS client

Considering that don't have QoS requirements, set "QoS" to "None" for both Userdemo2 and Userdemo1 register.

4.6.3 Test tools

Wireshark [94] is the open source software that works as a network packet analyzer. It can capture network packets and try to display the packet data in detail. It can be used to examine what is going on inside a network cable, so that can use it to examine kinds of problems, to debug protocol implementations, or to learn network protocol internals. In addition to these, Wireshark is also available for both Linux and Windows OS, and it is easy to use for searching and filtering packets on many criteria packets display based on filters. It is used to analyze each packet, the SIP protocol and the Diameter protocol in this project.

SIPp [97] is the Open Source test tools used for performance testing of SIP protocol. SIPp is one of the best tools to do the performance testing because it can establish and release multiple calls with the INVITE and BYE methods. Besides, it can also read XML scenario files describing any performance-testing configuration. It features the dynamic display of statistics about running tests, periodic CSV statistics dumps, TCP and UDP over multiple sockets or multiplexed

with retransmission management, regular expressions and variables in scenario files, and dynamically adjustable call rates. It is also very useful to emulate thousands of user agents calling the SIP system.

Although SIPp is much powerful, it is not so easy to use. As the reason that it is not the software using window interface, the operator has to write commands to control the signaling procedure. The operator should also write the register-Userdemo2.xml and register.csv.

SIPp can be used to test many real SIP equipments like SIP proxies, B2BUAs, SIP media servers, SIP/x gateways, SIP PBX, etc. It is also very useful in emulating thousands of user agents calling the SIP system.

4.7 SIP signaling in OpenIMS

Session Initiation Protocol (SIP) is IP phone/Multimedia Session Protocol based on text coding which is defined by IETF. [98]

In SIP protocol, if “A” send request to B, then A is Client and B is server. Contrariwise, if B send request to A, then B is Client and A is server.

SIP used to control multimedia sessions between users. What it can do is set up, modify and stop the multimedia sessions. There are 6 ways to carry out those functions: [99]

- **INVITE:** Sending a call via invite user.
- **ACK:** Using together with INVITE information to confirm the UAC has received the finally response of INVITE request.
- **BYE:** User Agent uses this method to release call.
- **CANCEL:** Used to cancel an unfinished request, has nothing to do with completed request.
- **REGISTER:** User using this method to register the addresses in to server. User Agent sending REGISTER request to address when it starts working, in order to accomplish the registration to local server.
- **OPTIONS:** Used to inquire after available User Agent or Servers.

The figure 4.13 shows the different cases use in test bed setup.

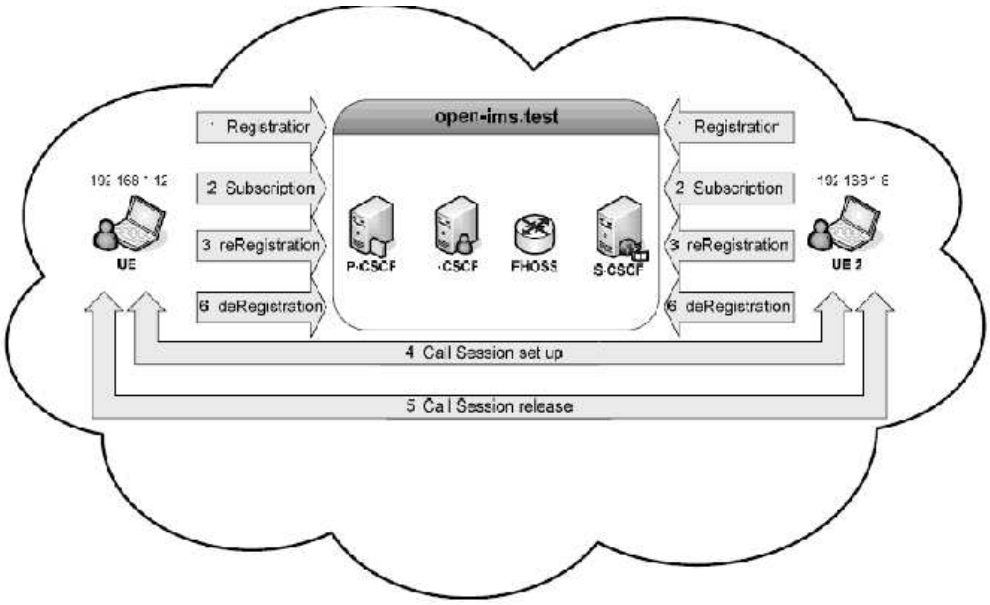

Figure 4.13 Test cases in evaluating OpenIMS

4.7.1 Registration Procedures

Before an IMS terminal begins any operation, there are several required prerequisites that have to follow. First thing is to establish an IMS service contract or subscription between the end-user and IMS service provider so that the end-user can be authorized when registering, making calls or doing other services.

After this establishment, IMS terminal needs to access to an IP-CAN (IP Connectivity Access Network) such as GPRS, ADSL or WLAN, because IP-CAN can provide the IMS terminal connection to the IMS home network or IMS visited network. But in this situation, just concentrate on the Open IMS core network. In order to make the whole structure much easier, put the IMS terminal in the same network as the IMS server. Therefore, the IP-CAN is not considered in the project.

In addition to the above aspects, IMS terminal also needs to discover the IP address of the P-CSCF that acts as an outbound/inbound SIP proxy server. This step is so important because IMS terminal can send and receive SIP signaling to or from the P-CSCF when P-CSCF discovery procedure is finished. When the prerequisites are fulfilled firstly, the IMS terminal can start to register to the IMS network. It's a regular SIP registration.

The figure 4-14 shows the general procedure of a IMS terminal registering to the OpenIMS Core network.

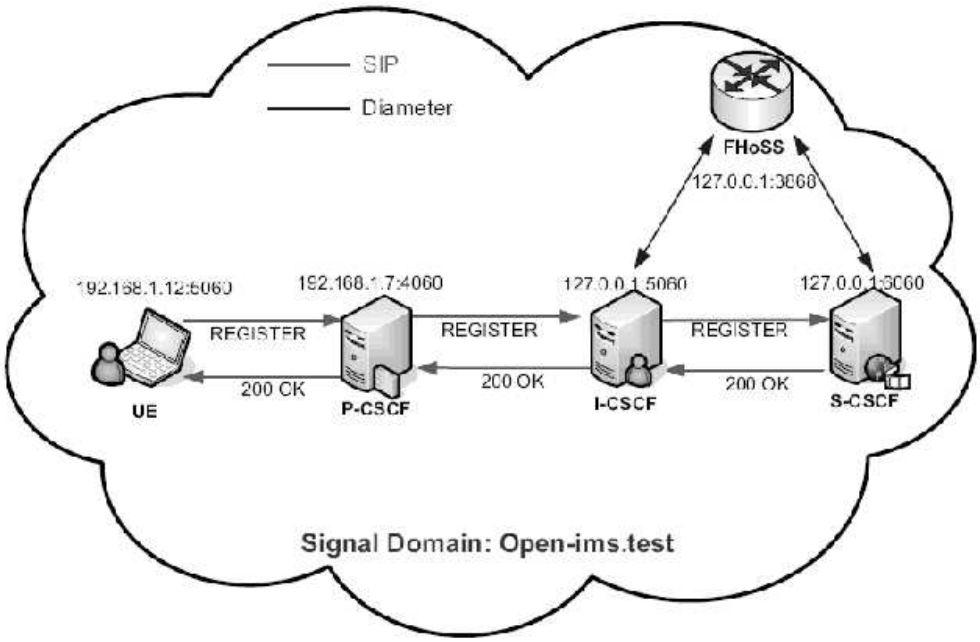

Figure 4-14 Registration for IMS client in OpenIMS

As shown in the figure 4-17, UE sends the SIP register request to P-CSCF, P-CSCF will send the register messages that contain P-CSCF name and user information to I-CSCF. I-CSCF obtains the HSS address and exchanges UMTS information with HSS through Cx interface, while at the same time HSS chooses the suitable S-CSCF for UE according to the request of I-CSCF and sends the address back to I-CSCF.

After that, I-CSCF will deliver the register information to the selected S-CSCF. HSS stores register information with user identity and the corresponding S-CSCF name. At the end, S-CSCF answers the 200 ok to UE, and then sends the connect information back to I-CSCF and releases all register information. At that time, UE is prepared to use needed multimedia services.

The purpose of registration for end-users is that they can be authorized to access the IMS network and use the IMS services. IMS Registration is accomplished by a SIP REGISTER request. First IMS terminal retrieves from ISIM the Private User Identities, Public User Identities and home network domain URI, and then it creates a SIP REGISTER request including the following four parameters.

- **The registration URI:** this is used to identify the home network domain, and it included in the *Request-URI* of the REGISTER request. In this situation, it is: *sip:open-ims. test.*
- **The Public User Identity:** it is a SIP URI that represents the user ID under registration. And it is included in the *to* header field value of the REGISTER request. In out case, it is: *sip:Userdemo2@open-ims.test.*

- **The Private User Identity:** this is only used for authentication, in this case, it is showed as: *Userdemo2@open-ims.test.* can find it in the in the *username* parameter of the *Authorization* header field value of SIP REGIDTER request.
- **The Contact address:** this is a SIP URI which includes the IP address of the IMS terminal or a host name where the user is reachable. It is contained in the SIP *contact* header. And it is showed as: *sip:Userdemo2@192.168.1.12:5060* in this case.

Internet Protocol, Src: 192.168.1.12 (192.168.1.12), Dst: 192.168.1.7(192.168.1.7) User Datagram Protocol, Src Port: 5060 (5060), Dst Port: 4060 (4060) Session Initiation Protocol

Request-Line: REGISTER sip:open-ims.test SIP/2.0 Message Header

Via: SIP/2.0/UDP 192.168.1.12:5060;rport;branch=z9hG4bK1480421391

Route: <sip:pcscf.open-ims.test:4060;lr>

From: <sip:Userdemo2@open-ims.test>;tag=1286929239

To: <sip:Userdemo2@open-ims.test>

Call-ID: 1298980893® 192.168.1.12

CSeq: 1 REGISTER

Contact: <sip:Userdemo2@ 192.168.1.12:5060>

Authorization: Digest username="Userdemo2@open-ims.test", realm="open-ims.test", nonce=" ", uri="sip:open-ims.test", response=" "

Max-Forwards: 70

User-Agent: eXosip/2.2.2

Expires: 600000

Supported: path

Content-Length: 0

4.7.2 Subscription procedure

By this stage, the registrar accepts the registration and creates a registration state. And the IMS terminal will subscribe to its registration-state information. The IMS terminal sends the SUBSCRIBER request for the event: reg to the P-CSCF and P-CSCF should proxy the request to the S-CSCF. After the S-CSCF receives the request, it acts as a SIP notifier and sends a 200 (OK) response, and the P-CSCF forwards the response to the terminal. Besides, the S-CSCF should also send a NOTIFY request and the P-CSCF shall forward it to the terminal. Contained in this NOTIFY request are all the Public User Identities allocated to the user during the user's registration state. At the end, the terminal answers with

a 200 (OK) response and the P-CSCF forwards the response to the S-CSCF.

4.7.3 Call Session procedure

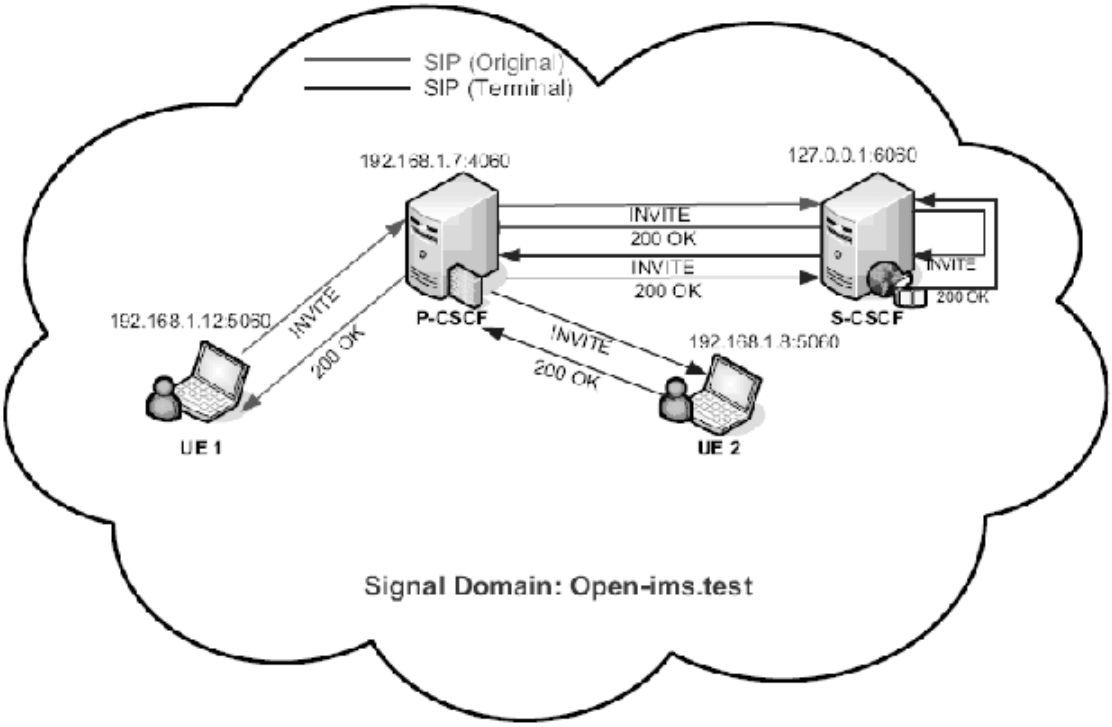

Figure 4.15: IMS client basic call setup

The IMS Terminal Sends an INVITE Request

The log message below shows this situation where the IMS client sends an INVITE request to the network. Taking a look at the message header field; This field is consisting of following header fields.

- **The Request-URI header field:** It contains the Public User Identity that shows Userdemo2's identity which is belongs to the operator "open-ims.test".
- **The Via header field:** In this header field, there are the IP address and the port number showing where the IMS terminal responses to the INVITE request. This header also shows what transport protocol will be used to transport SIP messages, in this case the protocol is "UDP".
- **The Contact header field:** The value of this header is set to the SIP URI that the IMS terminal is supported to receive subsequent request.
- **The Router header field:** The value of this header points to the P-CSCF in the visited network and the S-CSCF in the home network. As shown below, because this call is made within one network, the visit network and the home network are the same, which is known as "open-ims.test".

Route: <sip:pcscf.open-ims.test:4060;lr>

Route: <sip:orig@scscf.open-ims.test:6060;lr>

- **The P-Preferred-identity header field**: The value of this header set to "Userdemo2" and a SIP URI. When the P-CSCF forwarding the INVITE request to the home network, it will change this header filed into a P-Asserted-identity header field containing the same value.
- **The P-Asserted-identity header field:** Whenever the P-CSCF forwards the SIP request; the request always contains a P-Asserted-identity header field, which includes a Public User Identity value.
- The P-Assess-Network-info header field
- The From and To header fields
- The Privacy header field: This header field used to show that the user is willing to indicate some private information to called party.
- The Require, Proxy-Require, and Supported header fields The Supported header field declares the SIP extensions that will be used in the response. For instance, in this example, the IMS terminal shows that it supports the provisioned response extension "100rel".
- The Allow header fields: Though this header field is optional, it is important. It shows the SIP methods that the IMS terminal supports.
- The Content-Type and Content-Length header fields: The values of those two headers rely on the body included in the INVITE request.

Request-Line: INVITE sip:Userdemo1@open-ims.test SIP/2.0 Message Header

Via:SIP/2.0/UDP 192.168.1.12:5060;rport;branch=z9hG4bK2099595392

Route: <sip:pcscf.open-ims.test:4060;lr>

Route: <sip:orig@scscf.open-ims.test:6060;lr>

From: "Userdemo2" <sip:Userdemo2@open-ims.test>;tag=1990381184

To: <sip:Userdemo1@open-ims.test>

Call-ID: 1621887217® 192.168.1.12

CSeq: 20 INVITE

Contact: <sip:Userdemo2@192.168.1.12:5060>

Max-Forwards: 70

User-Agent: UCT IMS Client

Subject: IMS Call

Expires: 120

P-Preferred-Identity: "Userdemo2" <sip:Userdemo2@open-ims.test>

Privacy: none

P-Access-Network-Info: IEEE-802.1 la

Require: see-agree

Proxy-Require: see-agree

Supported: lOOrel

Allow: INVITE, ACK, UPDATE, INFO, CANCEL, BYE, OPTIONS, REFER, SUBSCRIBE, NOTIFY, MESSAGE

Content-Type: application/sdp

Content-Length: 322

The Originating P-CSCF Processes the INVITE response

The P-CSCF first checks if the Router header contains the value that the Service-Router header field the IMS terminal received. Since the Router header contains the value, P-CSCF knows that the Router header is correctly populated. Then, P-CSCF checks whether a P-Preferred-identity header is in the INVITE request. In this example, the P-Preferred-identity header exists, then P-CSCF deletes it in the INVITE and inserts a P-Asserted-identity header field and set its value to a SIP registered Public User Identity of the user.

The P-CSCF records the router and inserts a Record-Router header field with its own SIP URI when a SIP proxy wants to remain in the path for subsequent operation.

Request-Line: INVITE sip:Userdemo1@open-ims.test SIP/2.0 Message Header
Route: <sip:orig@scscf.open-ims.test:6060;lr>
Record-Route: <sip:mo@pcscf.open-ims.test:4060;lr>
Via: SIP/2.0/UDP 192.168.1.7:4060;branch=z9hG4bK17b.09719f96.0
Via: SIP/2.0/UDP
192.168.1.12:5060; rport=5060;branch=z9hG4bK2099595392
From: "Userdemo2" <sip:Userdemo2@open-ims.test>;tag=1990381184
To: <sip:Userdemo1@open-ims.test>
Call-ID: 1621887217@192.168.1.12
CSeq: 20 INVITE
Contact: <sip:Userdemo2@ 192.168.1.12:5060>
Max-Forwards: 16
User-Agent: UCT IMS Client
Subject: IMS Call
Expires: 120
Privacy: none

P-Access-Network-Info: IEEE-802.11 a
Require: see-agree
Proxy-Require: see-agree
Supported: 100 rel
Allow: INVITE, ACK, UPDATE, INFO, CANCEL, BYE, OPTIONS, REFER, SUBSCRIBE, NOTIFY, MESSAGE
Content-Type: application/sdp
Content-Length: 322
P-Asserted-Identity: "Userdemo2" <sip:Userdemo2@open-ims.test>
P-Charging-Vector: icid-value="P-CSCFabcd00000000460eadl600000075"; icid-generated-at=" 127.0.0.1"; orig-ioi="open-ims.test"

The Originating S-CSCF Processes the INVITE Request

The S-CSCF tries to route the SIP request based on the destination in the Request-URI. In this case the Request-URI is set to <sip: Userdemo1@open-ims.test>, so the S-CSCF tries to find a SIP server in the network named "open-ims.test". Since the request is forwarded within the homework, S-CSCF still keeps the P-Access-Network-lnfo header field in the request.

The Terminating S-CSCF Processes the INVITE Request

The S-CSCF in the terminating network used to take care of the callee receives the INVITE request. First, it verifies the callee by the Request-URI in the request, and it adds its own SIP URI to the Record-Router header field value to remains in the path. Then it continues with the processing of the INVITE request. It creates a new Request-URI with the contents of the Content header field value that were registered by the callee (<sip: Userdemo1@192.168.1.8:5060 SIP/2.0>). And it sets the Router header to that of the Path header that contains the P-CSCF in as showed below.

Router: <sip: term@pcscf.open-ims.test:4060;lr>

The S-CSCF is retargeted, so it inserts a P-Called-Party-ID header field which is set to the original Request-URI.

P-Called-Party-ID: <sip: Userdemo1@open-ims.test>

Then S-CSCF forwards the INVITE request including the P-Called-Party-ID header field and at the end the request will reach to the P-CSCF.

The Terminal P-CSCF Processes the INVITE request

In this case the caller set the Privacy header field to "none", therefore no privacy is required, so the P-CSCF keeps the P-Asserted-Identity header field in the INVITE request. Also, the P-CSCF extracts the Public User Identity from the P-Called-Party-ID header of the SIP INVITE request and identifies the Public User Identity of the callee.

The Callee's Terminal Processes the INVITE Request

The IMS terminal checks the P-Asserted-identity header field and gets the result that it presents, so the IMS terminal extracts the identity of the caller. Then it inspects the value of the P-Called-Party-ID to determine where the INVITE request is addressed. At the same time the IMS terminal inserts a Contact header whose value is a SIP URI that including the IP address and the port number.

The Callee's IMS Terminal Processes the PRACK request

When the PRACK is received at the callee the IMS terminal generates a 200OK response that is different from the 200OK to the INVITE request.

The situation of SIP client is very similar to the IMS client case. However, there are some differences described below.

As showed below, when the SIP terminal sends an INVITE request, there is no P-Preferred-identity header, no P-Access-Network-info header, no Privacy header and no Require, Proxy-Require, Supported header fields in the message header field.

Request-Line: INVITE sip:user1@open-ims.test SIP/2.0 Message Header
Via: SIP/2.0/UDP
192.168.1.15:34156;branch=z9hG4bK-d87543-d852681fdl5532 72-l~d87543-;iport
Max-Forwards: 70
Route: <sip:orig@scscf.open-ims.test:6060;lr>
Contact: <sip:user2@192.168.1.15:34156>
To: "user1 (Softphone)"<sip:user1@open-ims.test>
From: "user2"<sip:user2@open-ims.test>;tag=ef22a914
Call-ID: M2IwM2HN2QxZTJiZTcxNGFmNjEzZTFhNjNkZDE4MzI.
CSeq: 1 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE, SUBSCRIBE, INFO
Content-Type: application/sdp
User-Agent: X-Lite release 1006e stamp 34025
Content-Length: 325

From the basic session setup figure for SIP terminal, between the S-CSCF in the originating home network and the P-CSCF in terminating visited network, there is a terminating home network containing I-CSCF, HSS and S-CSCF. Therefore, discuss another situation below.

The Terminating I-CSCF Processes the INVITE request

The S-CSCF has found the I-CSCF at the SIP server in the home network. The I-CSCF is used to identify the callee in the Request-URI of the INVITE request and has to forward the SIP request to the S-CSCF allocated to the callee. To discover the address of the S-CSCF is allocated to the callee, the I-CSCF queries

the HSS where the address of the S-CSCF is stored with a Diameter Location-Information-Request (LIR) message. After the HSS received the message, it gets the stored S-CSCF address and inserts it in a Diameter Location-Information-Answer (LIA) message, sending it to the I-CSCF. Till then, I-CSCF can route the INVITE request to the S-CSCF that is allocated to the callee.

Another difference from the IMS case is that there is no PRACK response in the session for SI Pease.

4.8 Evaluation of OpenIMS components

4.8.1 Evaluation at the UE

Table 4-1: Evaluation at the UE

TS 24.229 v6			R6 Conform	R6 Non-conforma	comments
5.1 Procedures at the UE					
	5.1.1.2 Initial registration		#		
		On sending a REGISTER request		#	no security-client; no P-Access-Network-Info header
		On reseiving the 200 (OK)response to the REGISTER	#		without considering security
	5.1.1.3 Initial subscription to the registration-state event package			#	no P- Access-Network - Info header; no Event & Expires header
	5.1.1.5 Authentication				
		On receiving a 401 (Unauthorized)reponse to the REGISTER request		#	
		the 401 (Unauthorized) reponse to the REGISTER request is deemed to be valid		#	
		On receiving the 200 (OK) reponse for the security association protected REGISTER request		#	
	5.1.1.6 User-initiated deregistration				
		On sending a REGISTER request		#	no Security-Verity; no security-client; no P-Access-Network-Info
		On receiving the 200 (OK) reponseto the REGISTER request	#		
	5.1.2.1 Notification about multiple registered pubblic user identities		*(IMS)	*(sn>)	
	5.1.3.1 Initial INVITE request		*(IMS)		

As showed in the table, since testing is within one single domain and no security is required, so for both IMS and SIP clients, in most cases such as registration, deregistration and authentication, there are no security-client or security-server headers, no security-association, and no P-Access-Network-Info and P-Associated-URI headers. Besides, when UE initials subscription to the registration-state event package, the event header should set to "reg". But for SIP clients, the event header set to "message-summary". Otherwise, these testing situations are mostly consistent with the 3GPP TS 24.229 specification.

In Appendix B described the testing situations in detail according to the sub-clauses in the specification 3GPP TS 24.229.

4.8.2 Evaluation at the P-CSCF

Generally speaking, the functionality of the P-CSCF is mostly conformant to the specification of 3GPP R6 TS 24.229. The P-CSCF of OpenIMS support the Path and Service-Route headers, and the Path header is only used in the REGISTER request and its 200 (OK) response, while the Service-Route header is only applicable to the 200 (OK) response of REGISTER request.

The difference in OpenIMS is: for both IMS Client and SIP Client, there is not P-Charging-Function-Addresses header. Therefore, the functionality of P-CSCF with P-Charging-Function-Addresses header is not considered. The other difference is that there is no P-Media-Authorization header in OpenIMS, because what this project concentrates on is just OpenIMS Core in which the AS is not included.

As for the reason that the security is not considered, the REGISTER request is not protected. Therefore, the REGISTER request is not protected in this case, and the Security-Client header does not exist. The related information regarding security is different from the specification. For example, there are no security associations, Security-Server, or reg-await-auth timer. And the parameter "integrity-protected" is inserted with the value "no". And the architecture of the OpenIMS Core in this case is too simple to include the security, because all the components are fixed in a single domain.

Table 4-2 Evaluations at the P-CSCF

TS 24.229 v6			R6 Conformant	R6 Non-conformant	Comments
5.2 Procedure at the P-CSCF					
	5.2.1 General		#		
	5.2.2 Registration				
		When the P-CSCF receives a REGISTER request from the UE		#	not protected; no Security-Client header
		When the P-CSCF receives a 401(Unauthorized) response to a REGISTER request	#		Without considering security
		When the P-CSCF receives a 200(OK) response to a REGISTER request		#	no contact header; no P-Asserted-Identity header; no P-Charging-Function-Address; term-ioi is not received header
	5.2.3 Subscription to the user's registration-state event package				
		Upon receipt of a 200(OK) response to the initial REGISTER request		#	From header is not set to P-CSCF's SIP URI; the Expire header is
	5.2.5 Deregistration			* (SIP)	no functionality of deregistration for SIP
		5.2.5.1 User-initiated deregistration		#	does not remove the Public User Identity found in the To
	5.2.6 General treatment for all dialogs and standalone transactions excluding the REGISTER method				
		5.2.6.3 Requests initiated by the UE		#	HO P-Charging-Function-Address header
		5.2.6.4 Requests terminated by the UE	#		
	5.2.7 Initial INVITE				
		5.2.7.2 Mobile-originating case		#	doesn't insert the P-Media-Authorization header
	5.2.8.1 P-CSCF-initiated Call release				
		5.2.8.1.2 Release of an existing session	#		for IMS: P-CSCF serves the calling user, for SIP: P-CSCF serves the called user

The next difference is that P-CSCF cannot store the values received in the P-Charging-Function-Address header. The last difference is that a term-ioi parameter is not received in the P-Charging-Vector header.

For the situation where it receives a 200 (OK) response to the initial REGISTER request, The P-CSCF will generate a SUBSCRIBE request but the From header is not set to the P-CSCF's SIP URI. It set as: sip:Userdemo2@open-ims.test which is a Public User Identity's SIP URI. And the Expires header is still set to 600000 which are the same as the Expires header indicated in the 200 (OK) responses to the REGISTER request.

In deregistration case, for the SIP Client, it doesn't support the functionality of deregistration. For the IMS Client, there are some functionalities of deregistration are different from the specification.

When the P-CSCF receives an initial request for a dialog or a request for a standalone transaction, the request of IMS client contains a P-Preferred-Identity header, so the P-CSCF shall identify the initiator of the request by that public user identity. As to the SIP client, the situation is different. The request of the SIP client doesn't contain a P-Preferred-Identity header, so the P-CSCF shall identify the initiator of the request by a default public user identity.

There is no Service-Route header in this situation, and therefore not consider the related cases.

Both the IMS and SIP client add their own address via header, a situation that is conformant to the specifications. When the P-CSCF receives a 1xx or 2xx response to the before request, the P-CSCF shall not store the values received in the P-Charging-Function-Address header, because don't have this header in this cases.

When the P-CSCF receives an INVITE request from the UE, the P-CSCF shall respond to all INVITE requests with a 100 (Trying) provisional response that is conformant to the specification. But the P-CSCF doesn't insert the P-Media-Authorization header containing that media authorization token.

For the situation of call release, for the IMS clients, the P-CSCF serves the *calling* user of the session it shall generate a BYE request based on the information saved for the related dialog. And for SIP client, the P-CSCF serves the *called* user of the session it shall generate a BYE request based on the information saved for the related dialog. The situations of security association and access network are not considered.

In Appendix B described the testing situations in detail according to the sub-clauses in the specification 3GPP TS 24.229.

4.8.3 Evaluation at the I-CSCF

Table 4-3: Evaluation at the I-CSCF

TS 24.229 v6			R6 Conformant	R6 Non-conformant	Comments
5.3 Procedure at the I-CSCF					
	5.3.1 Registration procedure				
		5.3.1.2 Normal procedures		#	The SLF is not included
	5.3.2 Initial requests		#		don't consider the IP connective access network
		5.3.2.1 Normal procedures		#	for IMS: remove SIP URI, route the request; for SIP: contains more than one Route header.
	5.3.3 THIG functionality in the I-CSCF (THIG)			#	in single domain

Generally speaking, the I-CSCF behaves as a stable proxy during the registration procedure.

The I-CSCF decides which HSS to query, and possibly as a result of a query to the Subscription Locator Functional (SLF) entity. But in the OpenIMS Core, the SLF is not included.

All components in this situation are in a signal domain. Therefore, not considered the IP connective access network. That's the reason don't have *P-Access-Network-Info* headers. As the same reason, not see the procedures about I-CSCF shown in the Wireshark [94] log messages.

There is a situation, which is different on IMS Client and SIP Client:

When the I-CSCF receives an initial request for a dialog or standalone transaction, traced the log messages about IMS Client, and found that the I-CSCF removes its own SIP URI from the topmost *Route* header, and routes the request based on the *Request-URI* header field. With the trace on SIP Client, the situation is different. I-CSCF contains more than one *Route* header, and I-CSCF at first removes its own SIP URI from the topmost *Route* header, and then forwards the request based on the topmost *Route* header.

The situation of THIG is not considered, because the visited network and the home network are the same in this case.

In Appendix B described the testing situations in detail according to the sub-clauses in the specification 3GPP TS 24.229.

4.8.4 Evaluation at the S-CSCF

Table 4-4 Evaluation at the S-CSCF

TS24. 229 v6			R6 Conformant	R6 Non-conformant	Comments
5.4 Procedure at the S-CSCF					
	5.4.1.1 Introduction		#		
	5.4.1.2 Initial registration and user-initiated reregistration				
		5.4.1.2.1 Unprotected REGISTER	*(IMS)	*(SIP)	for SIP: no ik.ck field and no reg-await-auth header
	5.4.1.4 User-initiated deregistration			#	for IMS: integrity-protected =0; for SIP doesn't support function of deregistration
	5.4.2.1 Subscriptions to S-CSCF events				
		5.4.2.1.1 Subscription to the event providing registration state	#		without considering association security
	5.4.3.1 Determination of mobile-originated or mobile-terminated case		#		
	5.4.3.2 Requests initiated by the served user			#	no P-charging- Verity header; no P-charging-Function-Address header; no need of network-hiding; no P-Access-Network-Info header
	5.4.4.1 Initial INVITE		#		
	5.4.4.2 Subsequent requests				
		5.4.4.2.1 Mobile-originating case		#	
		5.4.4.2.2 Mobile-terminating case		#	
	5.4.5.1 S-CSCF-initiated session release				
		5.4.5.1.2 Release of an existing session	#		

As showed in the table 4-4, testing for registration procedures, both IMS clients and SIP clients support the Path header and Service-Route header. However, for IMS cases, those two headers also appear when the S-CSCF receives the "401 Unauthorized-Challenging the UE" which is not accordance to the specification 3GPP TS 24.229. According to the specification, the initiated register request in this testing case is unprotected. Under this condition, when receiving a REGISTER request with the "integrity-protected" parameter set to "no" or without the "integrity-

protected" parameter, the S-CSCF behaves almost as the specification says, but for both IMS and SIP clients, it doesn't start the timer reg-await-auth.

When an incoming SUBSRIBER request addressed to S-CSCF arrives containing the Event header, for both SIP client and IMS client, the S-CSCF can find the identity for authentication of the subscription in the P-Asserted-Identity header received in the SUBSRIBER request and also stores the value of the orig-ioi parameter received in the P-Charging-Vector header.

However, for SIP client, the SUBSCRIBER request is subscribed to the event "message-summary" instead of event "reg" described in the specification, so it cause all notify messages to give the "487 Package not Supported" information.

Looking at this testing cases for call session procedures, table 4.4 that when a request is initiated by served users, for both IMS and SIP clients, the S-CSCF doesn't insert P-Charging-Function-Addresses and there is no access-network-charging-info parameter in the P-Charging-Vector header field, and the S-CSCF doesn't remove the P-Access-Network-info header based on the destination URI. Besides, for both the SIP clients and the IMS clients, when the S-CSCF receives an INVITE request, the S-CSCF processes the initial INVITE request without examining the SDP.

In Appendix B described the testing situations in detail according to the sub-clauses in the specification 3GPP TS 24.229.

4.8.5 Evaluation at the SIP2IMS

To accelerate testing and to integrate with SIP UEs and test tools, a gateway that helps SIP traffic to work in an IMS environment was required. The SIP-to-IMS gateway performs this adaptation tasks. At the moment it translates between MD5 and AKAv1-MD5 authentications and helps with special headers. The SIP2IMS Gateway can been regarded as the helper module of the Open source IMS Core project to allow the verification of services with current state-of-the art VoIP clients. Yet, due to its gateway nature, it filters much of the IMS advantages and should not be regarded as a full-blown IMS solution.

The Open Source IMS SIP2IMS Gateway should allow transformation of IETF SIP messages to IMS conformant messages. It translates MD5 authentication to IMS AKA authentication. SIP2IMS can also enable developers to access core elements and to trial multimedia services by using a non-IMS VoIP client.

But in the real experiment, the gateway SIP2IMS is useless, use SIP client by changing the authentication-algorithm into MD5, while use IMS client by changing the authentication-algorithm into AKAv1-MD5. The SIP2IMS gateway doesn't afford the functionality to translate MD5 authentication to AKAv1-MD5 when use SIP client change into IMS client.

4.8.6 Evaluation at the Cx

In this situation, there are several commands that appear which are User-Authorization-Request (UAR), User-Authorization-Answer (UAA), Server-Assignment-Request (SAR), Server-Assignment-Answer (SAA), Location-Info-Request (LIR), Location-Info-Answer (LIA), Multimedia-Auth-Request (MAR), Multimedia-Auth-Answer (MAA). For both IMS clients and SIP clients, these examples are mostly in accord with the 3GPP TS 29.229. All the mandatory AVPs and most optional AVPS in those commands. However, there are no "Registration-Termination-Request (RTR)", "Registration-Termination-Answer (RTA)", "Push-Profile-Request (PPR)" and "Push-Profile-Answer" commands in these examples.

For both IMS client and SIP client in example, there are two values standing for success that are "DIAMETER_FIRST_REGISTRATION" & "DIAMETER_SUBSEQUENT_REGISTRATION".

The "DIAMETER_FIRST_REGISTRATION" appears in MAA, SAA and LIA commands while the "DIAMETER_SUBSEQUENT_REGISTRATION" appears in UAA command during the registration process.

There are also several AVPs that are showed in the [23] of 3GPP TS 29.229 that appeared in this examples, namely, Visited-Network-identifier AVP (600), Public-Identity AVP (601), Server-Name AVP (602), User-Data AVP (606), SIP-Number-Auth-Items AVP (607), SIP-Auth-Data-item AVP (612), Server-Assignment-Type AVP (614), Charging-information AVP (618), User-Authorization-Type AVP (623), User-Data-Already-Available AVP (624)

In Appendix B described the testing situations in detail according to the sub-clauses in the specification 3GPP TS 24.229.

4.8.7 Evaluation at the abnormal cases

At first, choose SJphone [100] as the SIP Client with which are always got the following error:

403 forbidden-not registered! You must register first with a s-cscf

Checked it and found that SJ Phone doesn't put Expires header to the REGISTER request, so although could get 200 OK, it comes with 0 binding, which means it is not registered. In addition, there is another problem with SJ Phone, which is that doesn't follow Service Route headers. Therefore, use X-Lite which can at least work instead of SJphone.

During the register process, several errors occurred, brief of each as following:

503 : Server Unavailable

To solve this problem, need to change the DNS address to: 192.168.1.7. This is

because the DNS service on that host knows about the open-ims.test DNS zone.

600: Busy everywhere

The following trace when got this error:

Status-Line: SIP/2.0 600 Busy everywhere-Forwarding to S-CSCF failed

status-code: 600

[Resent Packet: False] Message Header

via: SIP/2.0/UDP 192.168.1.12:65290;

To: "Userdemo1"<sip:Userdemo1 at Open-ims.test:tag=a6alc5f60faecf035alae5b6e96e979a-c29e

From:"Userdemo1" <sip:Userdemo1 at open-ima.testx tag=f66a8220

Call-ID: YjY4M7FjMGExM2FmZTNhZTI2ZGNKNDI2NmUxNzU5N2Y

Cseq: 1 REGISTER

Server: sip Express router(2.1.0-devl-OpenIMSCore (x86_64/Linux))

Content-Length: 0

Warning: 392 127.0.0.1:5060 "Noisy feebback tells: pid=13230 req_src_ip=192.168.1.7 req_src_port=4060 in_uri=sip:open-ims.test out_uri=sip:scscf.open-ims.test:4060 via_cnt==0"

During this occurrence, just restarted the system and this problem was resolved, however, got another error as shown below;

403 forbidden - HSS user unknown

Since it shows 'user unknown', focus on check logs that are connected to user and see what the response is. After checking, find that made some mistakes when set up the account.

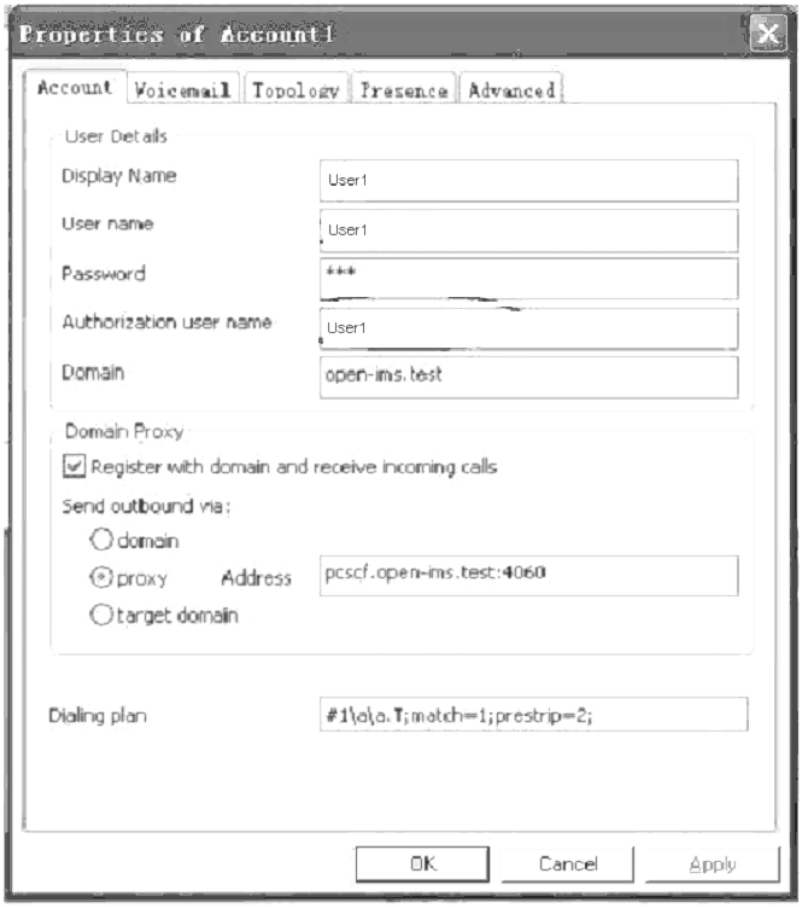

Figure 4-16 Mistakes in properties

As shown in figure 4-16, had some mistakes at 'user name' and 'Authorization user name'. Need to delete the space after 'Userdemo1' in 'user name', and have to change 'Authorization user name' into 'Userdemo1@open-ims.test'.

After made these two changes, tried to register again, and this time registration was successful!

During the register process, only met one problem said:

403 Forbidden—HSS returned no authentication vectors.

The S-CSCF sends the request for the HSS regarding the authentication vectors and also specifies the preferred scheme, and the HSS is looking on request, for an authentication scheme specified by the S-CSCF, which has to be used. Then, the HSS verifies if the user supports that scheme. If the user does not support the requested scheme, the HSS will send an error message to the S-CSCF with Authentication Scheme not supported. So to solve this problem need to configure the S-CSCF and the HSS for the same scheme.

For this case, first change the "Authentication algorithm" from "Digit-MD5" to"Digit-

AkAV1-MD5" from the HSS web interface. And then in the "scscf.cfg" file, change

```
modparam("scscf","registratiotn-default-algorithm","MD5")
#modparam("scscf","registration-default-algorithm","AKAV2-MD5")
#modparam("scscf","registrationi-default-algorithm","AKAVl-MD5")
```

TO:

```
modparam("scscf","registratiotn-default-algorithm","AKAVl-MD5")
#modparam("scscf","registration-default-algorithm","AKAV2-MD5")
#modparam("scscf","registrationi-default-algorithm","MD5")
```

After changing, restarted S-CSCF, and then tried to register again, and the problem was solved.

4.9 Solutions of interoperability between IMS and SIP

Through various kinds of experiments, found that most kinds of cases can be obtained, but they are just limited in using the same user equipment. For example, can set up the call session by using SIP Client - SIP Client, or IMS Client - IMS Client. In fact, that is too limited when used in enterprise solutions because some of the clients can only support SIP, and some can support IMS. No doubt companies want to find a way for the interoperability between SIP and IMS so that the establishment can be set up.

4.9.1 Use two S-CSCFs

The way to find out how to solve this problem is to know the reason why IMS service can only be used in the same user equipment. Make the experiment by using a SIP Client and an IMS Client at the same time. And then use the log message shown on Wireshark to analyze the reason.

During the register process, the following problem occured:

403 Forbidden—HSS returned no authentication vectors

After analyzing, found that the reason is something about authentication and the related components are may be S-CSCF and HSS.

In the evaluation of OpenIMS, One main feature of S-CSCF in OpenIMS is to retrieve authentication vectors. It supports several ways of authentication: AKAv1-MD5, AKAv2-MD5 and MD5. The authentication algorithm AKAv1-MD5 is supported for IMS Client, while the authentication algorithm MD5 is supported for SIP Client. The S-CSCF will choose one authentication algorithm when the clients register or call through OpenIMS. Therefore, if the clients belong to SIP clients or IMS clients, the S-CSCF could work correctly, but if the clients who want to connect to the OpenIMS belong to different kinds, the S-CSCF will be confused

about which authentication algorithm to choose, and then the error above will be shown.

Therefore, consider that if it is possible to use two S-CSCFs which support different authentication algorithms, one supporting 'AKAv1-MD5' for IMS Client, and the other supporting 'MD5' for SIP Client, so that the SIP Client and IMS Client can register at the same time, and implement some IMS services between them. This is one solution for the interoperability between IMS and SIP.

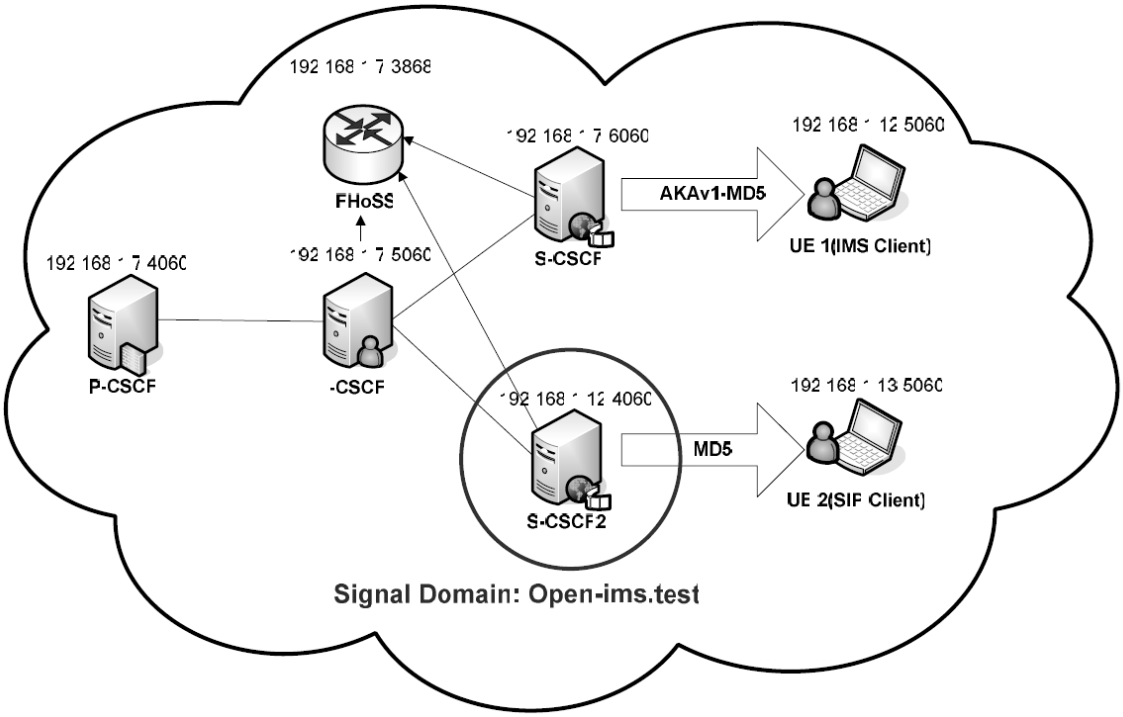

Figure 4.17: Interoperability between SIP and IMS

4.9.2 Installation and configuration of S-CSCF2

In order to reach this purpose, install the S-CSCF2 on another computer with the IP address and the port 168.192.1.12:4060, while the S-CSCF with the different IP address and the port number is: 168.192.1.7:5060. Some changes in the configuration files are required.

4.9.2.1 Add S-CSCF2 in DNS

cd var/named/etc/sites/open-ims.test
sudo vim open-ims.zone

This command is used to modify the zone.files. After open it, the scscf2 is added in the zone.file which is shown as follow:

```
$ORIGIN open-ims.test.
$TTL1W
@                       ID IN SOA       localhost. root.localhost. (
                                                2006101001      ; serial
                                                3H              ; refresh
                                                15M             ; retry
                                                1W              ; expiry
                                                ID )            ; minimum
                        ID IN NS        ns
ns                      ID IN A         192.168.1.7
pcscf                   ID IN A         192.168.1.7
open-ims.test.          ID IN A         192.168.1.7
icscf                   ID IN A         192.168.1.7
_sip                    ID SRV 0 0 5060 icscf
_sip._udp               ID SRV 0 0 5060 icscf
_sip._tcp               ID SRV 0 0 5060 icscf
open-ims.test.          ID              IN  NAPTR  10  50  "s"  "SIP+D2U
_sip._udp.open-
open-ims.test.          ID              IN  NAPTR  20  50  "s"  "SIP+D2T
_sip._tcp.open-ims.test.
scscf                   ID IN A         192.168.1.7
scscf2                  ID IN A         192.168.1.12
sip2ims                 ID IN A         192.168.1.7
hss                     ID IN A         192.168.1.7
ue                      ID IN A         192.168.1.7
presence                ID IN A         192.168.1.7
```

4.9.2.2 Installation of S-CSCF2

The installation steps are shown as follows

The first step is to create a new directory on the new computer. Then, to create the directory for serjms under the directory /opt/OpenIMSCore, get the Code of CSCFs - serjms/trunk from the page http://svn.berlios.de/svnroot/ repos/openimscore . After that, need to check if the CSCFs are there. The following step is to compile the CSCFs, which is to make libs install all in serjms. This step takes some time to finish and have to make sure all the prerequisites have been installed. Just need the new S-CSCF, so it is not necessary to install icscf .sql or other components of CSCFs into MySQL. The last step is to configure the IMS core; just need to copy the files of icscf: icscf.cfg, icscf.xml, icscf.sh to /opt/OpenIMSCore. Note that the necessary changes are needed in this situation.

```
mkdir /opt/OpenIMSCore
cd /opt/OpenIMSCore
```

```
mkdir ser_ims
svn checkout http://svn.berlios.de/svnroot/repos/openimscore/ser_ims/trunk ser_ims
cd ser_ims
make install-libs all
cd..
cp ser_ims/cfg/icscf.* .
```

4.9.2.3 Add S-CSCF2 in FHoSS

After the installation of s-cscf2, it should be edited in the icscf database and also inserted into the new one. The tables of icscf in database are shown as follows:

```
|Tables_in_icscf        +
| nds_trusted_domains
| s_cscf
I s_cscf_capabilities
```

Then select the data from s_cscf in this table, and edit the default S-CSCF as IMS S-CSCF with the uri as: sip:scscf.open-ims.test:6060 while inserting the new s-cscf2 as SIP S-CSCF with the uri as: sip:scscf2.open-ims.test:4060.

```
+----+-----------------+-------------------------------------+
id I name
+----+-----------------+-------------------------------------+
s cscf uri
| 1    | IMS S-CSCF | sip:scscf.open-ims.test:6060 |
2   | SIP S-CSCF | sip:scscf2.open-ims.test:4060 |
+----+-----------------+-------------------------------------+
```

Next, select the data from s_cscf_capabilities and edit it. The 0, 1 of capability represents the authentication algorithms of AKAv1-MD5 and MD5.

Therefore, 0 is assigned to IMS S-CSCF and 1 is assigned to the SIP S-CSCF.

```
+----+-----------------+------------------------------------+
id I id_s_cscf     | capability
+----+-----------------+------------------------------------+
| 1 | 1 | 0 |
| 2 | 1 | 0 |
| 3 | 2 | 1 |
| 4 | 2 | 1 |
+----+-------------+----------------+
```

Then, select the data from the diam_servers table. There is a default server and host for S-CSCF, and insert the new server and host for the S-CSCF2.

```
+-------------------------------------+-------------------------+
| server | host |
+-------------------------------------+------------------------- +
| sip:scscf.open-ims.test:6060 | scscf.open-ims.test |
| sip:scscf2.open-ims.test:4060 | scscf2.open-ims.test |
+-------------------------------------+-------------------------+
```

4.9.2.4 Modification in s-cscf/s-cscf2.cfg

As in the evaluation of abnormal cases in chapter, if the authentication algorithm in S-CSCF is not defined in right way, will receive the 403 error which is describe as HSS returning no authentication vectors. In case the error occurs, need to define the authentication algorithm in each s-cscf for each kind of client.

In the file of scscf.cfg, defined the default authentication algorithm as"Digit-AkAV1-MD5".

```
#modparam("scscf","registration_default_algorithm","MD5")
#modparam("scscf","registration_default_algorithm","AKAv2-MD5")
modparam("scscf","registration_default_algorithm","AKAv1-MD5")
```

And in the file of scscf2.cfg, define the default authentication algorithm as "Digit--MD5".

```
modparam("scscf","registration_default_algorithm","MD5")
#modparam("scscf","registration_default_algorithm","AKAv2-MD5")
#modparam("scscf","registration_default_algorithm","AKAv1-MD5")
```

After these steps, could register SIP Client and also IMS Client at the same time. Additionally, the call session can be established as well.

4.9.3 Registration and call session with S-CSCFs

As described earlier, still put the terminal and the server in the same network that is: open-ims.test. Following, will describe the situation where the SIP client registered and the call session to OpenlMS network use S-CSCF2.

4.9.3.1 Registration with S-CSCF 2

As showed in the figure 4.18, first the SIP terminal creates a SIP REGISTER request including four parameters that are the registration URI, the Public User Identity, the Private User Identity and the Contact address. In this situation:

- The registration URI is: *sip:open-ims.test*
- The Public User Identity is: *sip:user2@open-ims.testor sip:user1@open-ims.test*
- The Private User Identity is: user2@open-ims.test or user1@open-ims.test
- The Contact address is: user2(3)192.168.1.13:51066 or 11(3)192.168.1.5:5062

The SIP terminal sends the request to the P-CSCF. The P-CSCF inserts a P-Visited-Network-ID that shows where the P-CSCF is located. In this example this is shown: open-ims.test. And the P-CSCF also inserts a Path header with its own SIP URI to request the home network to forward all SIP requests. In this example, this SIP URI is: sip:term@pcscf.open-ims.test:4060. The P-CSCF discovers an I-CSCF in the home network and forwards the SIP REGISTER request to it.

The I-CSCF queries the HSS with a Diameter UAR message and the HSS answers with the Diameter UAA message. Eventually, the I-CSCF forwards the REGISTER request to the S-CSCF2.

The S-CSCF2 creates a Diameter MAR message and sends it to the HSS. The HSS then stores the S-CSCF2 URI in the user data for further query and answers with a Diameter MAA message. In this way, the S-CSCF2 has downloaded the authentication data from the HSS. After this, the S-CSCF2 creates a SIP 401 (Unauthorized) response and forwards it to the SIP terminal via I-CSCF and P-CSCF.

In response, the SIP terminal sends a new REGISTER request to the P-CSCF. Because the authentication was successful before, when the request is received by the S-CSCF2, it sends a Diameter SAR message to the HSS to inform it that the user is now registered and at the same time to download the user profile. Then, the S-CSCF2 sends a 200 (OK) response to the SIP terminal showing that the REGISTER request is successful.

By this stage, in theory the SIP terminal should send the SUBSCRIBER request for the event: reg to the P-CSCF and P-CSCF should proxy the request to the S-CSCF. The S-CSCF shall act as a SIP notifier and sends a 200 (OK) response, and the P-CSCF forwards the response to the terminal. In addition, the S-CSCF should also send a NOTIFY request and the P-CSCF shall forward it to the terminal. At the end, the terminal answers with a 200 (OK) response and the P-CSCF forwards the response to the S-CSCF.

However, in this example, the situation is quite different. First, after the SIP terminal receives a 200 (OK) response for the registration process from the S-CSCF2, it generates a new SUBSCRIBE request to event package: message-summary instead of event: reg. This request is forwarded to the S-CSCF2 via P-CSCF and I-CSCF, and the S-CSCF2 return also with the SUBSCRIBE request to the terminal. When the terminal receives the request, it sends a "489: Event Package not Supported" message to the P-CSCF and then the P-CSCF forwards the message to the S-CSCF2. The S-CSCF2 also forwards the message back to the terminal via I-CSCF and P-CSCF. This process is unexpected according to the theory, so think this SIP client is not supporting the event package: message-summary.

After this, instead of sending a SUBSCRIBE request to event: reg by the SIP terminal, the P-CSCF directly sends the SUBSCRIBE request towards the event package: reg to the I-CSCF, and the I-CSCF forwards the message to the S-CSCF2. Then the S-CSCF2 answers with a "200 Subscription to REG saved" message and the I-CSCF forwards the message to P-CSCF. Additionally, the S-CSCF sends a NOTIFY message to the P-CSCF and gets the answer 200 (OK).

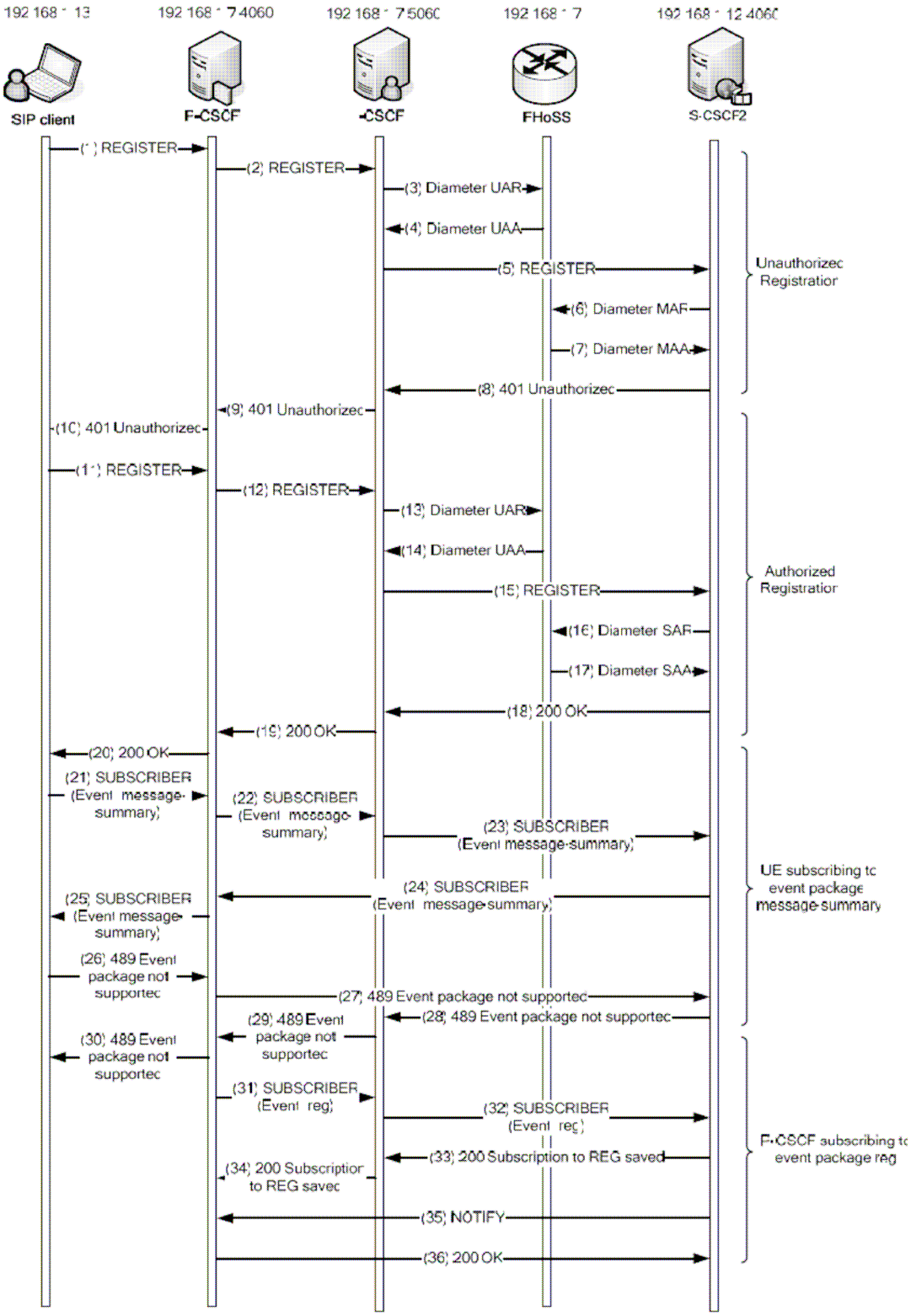

Figure 4.18 Registration SIP client to OpenIMS (with two S-CSCF)

4.9.3.2 Call session with S-CSCF 2

The figure 4.19 below shows the flow of call session setup between the SIP client and the IMS client. From this Wireshark logs cannot get detailed information about the S-CSCF2. From the figure, it can be seen that the call process between SIP and the IMS client is almost the same as the call process between SIP clients. For this, mention here that the "101 Dialog Establishment" message has not been defined in the 3GPP specification, so can consider it as an unexpected situation.

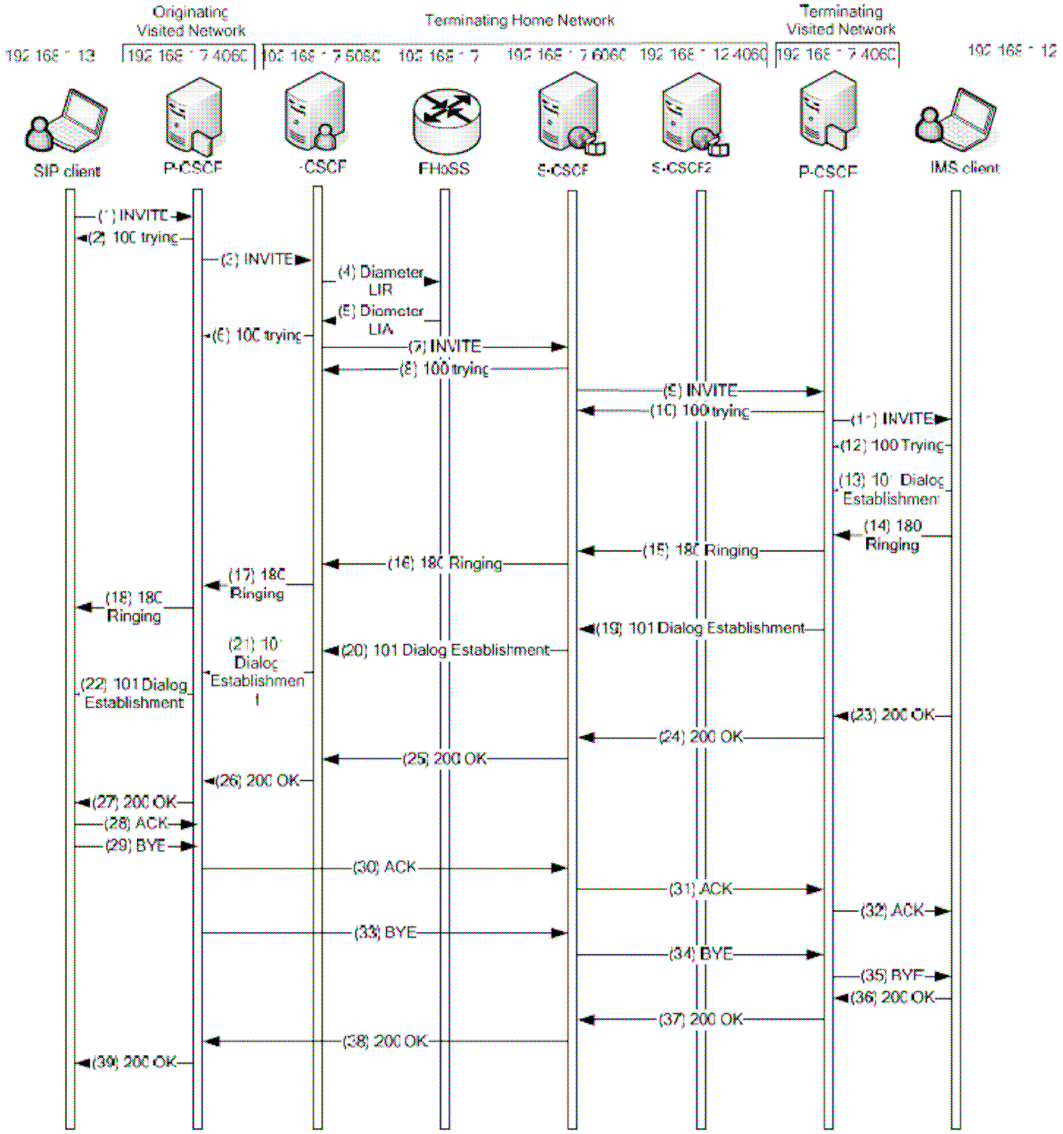

Figure 4.19: Call session setup between SIP client and IMS client

4.9.4 Evaluation about the solution with two S-CSCFs

The experiment shows it is possible to use two S-CSCFs supporting different authentication algorithms for SIP Client and IMS Client at the same time, so that the two clients can register to the OpenIMS at the same time and further establish

the call session. Although the solution can be implemented, there are still some latent problems.

4.9.4.1 Instability

After each registration or call session, not to continue to repeat the testing again, but to get **403 errors - HSS returned no authentication vectors.**

This error occurs when there is no matched authentication algorithm in S-CSCF. Therefore, check the scscf.cfg and the database in MySQL. There is no change in scscf.cfg or scscf2.cfg, but the data in MySQL/hssdb/impi is changed automatically, regarding this error occurance and solving process. IMPI is IP Multimedia Private Identity while IMPU is IP Multimedia Public Identity. They are contained in HSS user database.

The example below will illustrate how it changes. The data is too long to show, so it is shown with only the changeable parts.

The table is the data from MySQL/hssdb/impi, and is the state before testing. can derive from it that user2@open-ims.test and user1@open-ims.test are SIP Clients, and they both register to the S-CSCF 2 by using the "Digest-MD5" authentication algorithm, while Userdemo1@open-ims.test and Userdemo2@open-ime.test are IMS Clients who use "Digest-AKAv1-MD5" and register to the S-CSCF. Obviously, it is configured in the right way.

```
+--------------------+----+------------------------------+----+---------------------- +----+---------+
| impi_string |......| scscf_name |......| auth_scheme |......| algorithm |
+--------------------+----+------------------------------+---+-----------------------+---+---------+
|Userdemo1@open-ims.test|......|sip:scscf.open-ims.test:6060|......| Digest-AKAv1-MD5 |......| AKAv1|
|user2@open-ims.test |......|sip:scscf2.open-ims.test:4060|......| Digest-MD5 |......| NULL |
|user1@open-ims.test |......| sip:scscf2.open-ims.test:4060 |……| Digest-MD5 |…… | NULL |
|Userdemo2@open-ims.test|......|sip:scscf.open-ims.test:6060|......|Digest-AKAv1-MD5|......|AKAv1|
+----------------------+---+--------------------------------+---+-----------------------+---+--------+
```

Use of right configured data to test once, after that the data is changed by itself as following table.

```
+--------------------+----+------------------------------+----+---------------------- +----+---------+
| impi_string |......| scscf_name |......| auth_scheme |......| algorithm |
+--------------------+----+------------------------------+---+-----------------------+---+---------+
|Userdemo1@open-ims.test|......|sip:scscf.open-ims.test:6060|......| Digest-AKAv1-MD5 |......|
AKAv1|
|user2@open-ims.test |......|sip:scscf.open-ims.test:6060|......| |......| NULL |
|user1@open-ims.test|......|sip:scscf.open-ims.test:6060|……|Digest-AKAv1-MD5|……| NULL |
|Userdemo2@open-ims.test|......|sip:scscf.open-ims.test:6060|......|Digest-AKAv1-
MD5|......|AKAv1|
+----------------------+---+--------------------------------+---+-----------------------+---+--------+
```

As it shown in the table, the scscf_name may be changed with the other one using in IMS Client, or the auth_scheme may be changed into the other authentication algorithm or even into nothing. Especially, the changes is not fixed, sometimes, it changes as this; sometimes, it changes in the other similar way and sometimes it may not change. The situation is randomly.

The reason for the instability is because:

Assigned the S-CSCF 2 in the impi table manually, but this is not the normal way to do. As the trunk version of OpenIMS does not offer capability supporting, to make and modifications in the impi table. Therefore, changes in the databases do not change anything in the HSS functionality and that is also the reason that the assigned S-CSCF always remains the default S-CSCF after registration or other operations. Capabilities can be added to the branch version of FHoSS, but the capability functionality is not yet fully tested.

4.9.4.2 Call session released automatically

The other problem is the remote client always releases the call session automatically as soon as the callee receives the call. From the RTP package, the message can be sent from caller, while the callee cannot reply.

When change the X-Lite [95] to SIPp [97] to make a call initiated by user2 (SIP client) and invite Userdemo2 (IMS client), the call session can be established in the normal way.

When a call is initiated the request is transferred with SIP, while the actual audio is transferred over the Real-time Transport protocol (RTP) on another port. The end-to-end media transfer (RTP) contains only details of the private addresses and ports from the computer it was sent from. So when the client on the other side tries to return the media information it will fail, because the private address doesn't mean anything on the public network. [101]

4.10 Integration with SIP/VoIP solutions

Earlier in this chapter discussed implemented solution of the interoperability between IMS and SIP. However, this solution is only valid in one domain that is Test bed for Open IMS Test domain. Yet there are many users who still use non-IMS clients which directly connect to IMS, so a migration path to IMS is necessary.

In the migration stage, take for granted a user A has two identities; one for entry to the IMS domain while the other is used to enter the non-IMS domain. In this case, when someone tries to call user “A” in non-IMS domain, the user's terminals which are registered with the IMS domain needs to be reached.

4.10.1 Solution for migrate towards IMS

In [53], the authors came up with four possible solutions. In this work of research, attempt to implement their "client based" solution.

This solution demonstrates following:

"This solution requires an advanced client to inform enterprise SIP PBX that his location has moved to another domain, all incoming call to sip:userA@enterprise.com should be redirected to sip:userA@operate.com.

When user turns on the IMS terminal, it will register on IMS domain as sip:userA@operate.com. Then it will register on the enterprise SIP PBX as sip:userA@enterprise.com and inform SIP PBX about the location change.

When an incoming call from sip:userA@enterprise.com reaches SIP PBX, the SIP PBX will acts as a redirect server and sends a "302 Moved Temporarily" SIP message to caller userB and instruct userB to try userA's new location sip:userA@operate.com. Then userB will initiate a new call to sip:userA@operator.com which is directly sent to IMS domain, and userA's terminal."

In this situation, the "operator.com" which is IMS domain is this "open-ims.test" domain in HiA, and the "enterprise.com" is non-IMS domain in auSystems "agder-ikt104.hia.no".

Figure 4.20 & 4.21 shows the registration and basic call setup cases in theory.

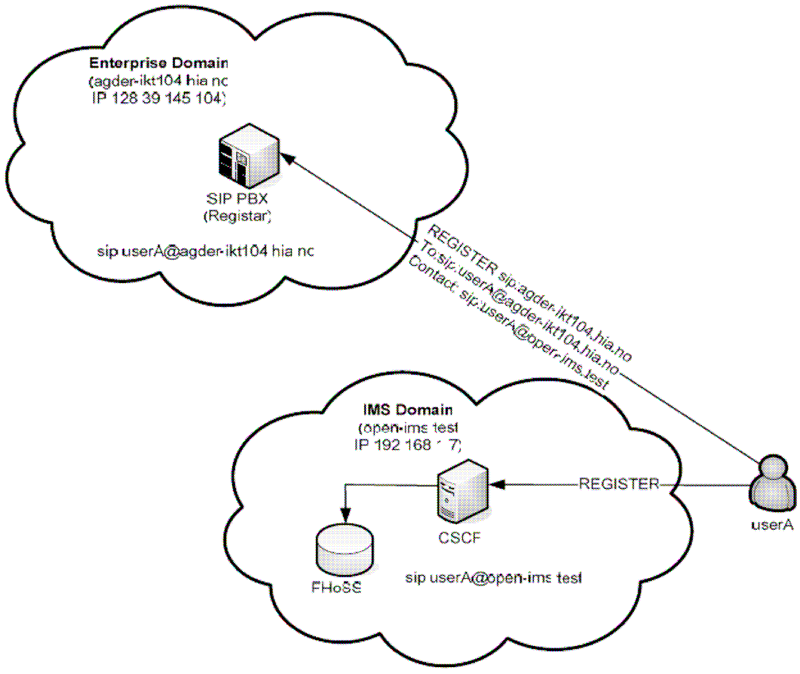

Figure 4.20: Client based solution—Registration case

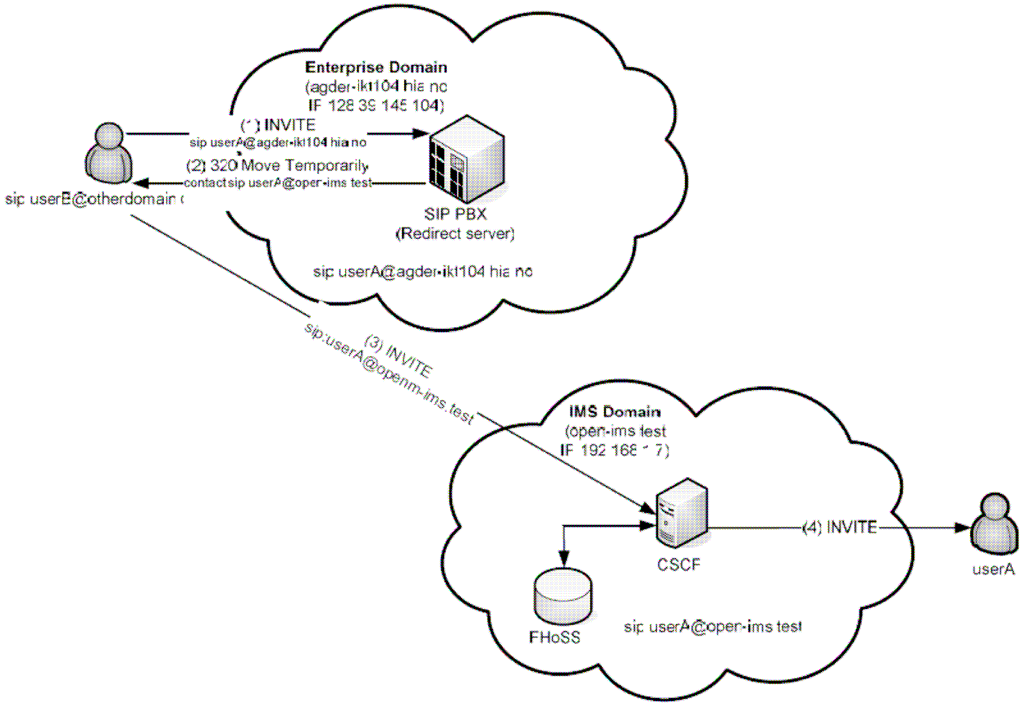

Figure 4.21: Client based solution—Call session setup case

4.10.2 Implementation of the "client based" solution

The SIP PBX in the HiA/auSystems testbed is already setup with two different SIP servers, one is SIP PBX: Asterisk 1.2.4, while the other is an alternative SIP server: openSER v1.0.1. Currently the Asterisk SIP server is working, while the openSER SIP server was a test installation, which is not working anymore. Hence, tried approaches with both Asterisk and openSER servers.

4.10.2.1 Using Asterisk server

Follow the introduction in [53], if use the Asterisk server, then this solution requires that the user terminal be SIP enabled and that domain settings are configurable, so that it can send SIP messages to the SIP PBX to inform the new location of the user.

Therefore, not to use normal SIP clients like X-Lite and GXP2000 that used before. Instead, use SIPp as this terminal where can set the "contact" address by ourselves.

4.10.2.2 Using openSER server

To make the situation easy, want to use normal SIP clients. Therefore, choose to add the openSER back to the testbed and let it work as a redirect server. After change the old Asterisk server to openSER redirect server, no longer need to use

a "contact" function since the redirect server will automatically redirect all invites.

4.10.2.3 Registration and Call session setup with redirect server

As demonstrate in the figure 4.22, User A registers to enterprise domain with the redirect server so that it gives the new location for further contact-information forwarding.

When User B tries to reach the User A in enterprise domain, the redirect server will tell User B the new location of User A, that is the redirect information. After User A receives this redirect information it will resend an INVITE request to the given address.

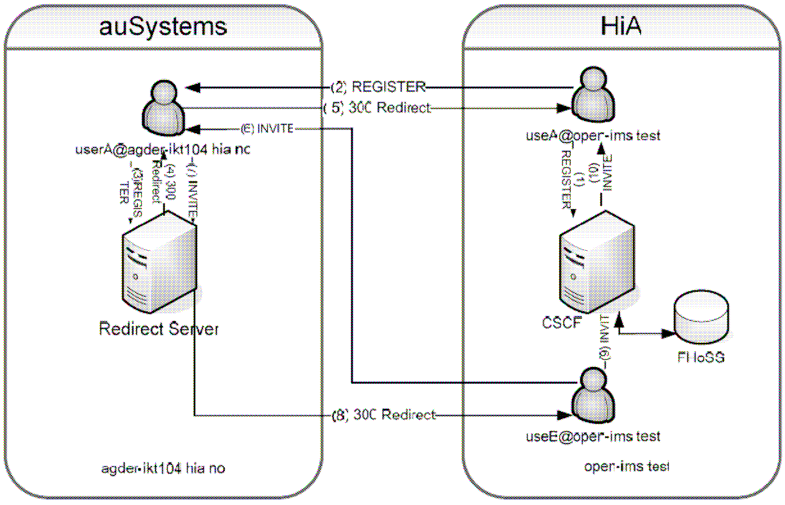

Figure 4.22 Registration and Call session setup with redirect server

In this case, use "user1" register to enterprise domain (agder-iktl 04.hia.no) with X-Lite, and let the redirect server tell the new location as "user1@open-ims.test". Below show the "300 redirect" message:

Session Initiation Protocol
Status-Line: SIP/2.0 300 Redirect
Message Header
Via:SIP/2.0/UDP
192.168.1.13:32824;branch=z9hG4bK-d87543-b523eela6248f76a-l-d87543-;
rport=57621 ;recei
ved=128.39.145.250

To:"user1"<sip:user1@agder-iktl04.hia.no>;tag=b27elald33761e85846fc98f5Oa7e58.cd9b

From: "Ii"<sip:user1@agder-iktl04.hia.no>;tag=67468573

Call-ID: OWU3YzIONjc2NTA2MTJkMmQ2ZDkOOWFmNGNiY2JkOTU.

CSeq: 1 SUBSCRIBE

Contact: sip:user1@open-ims.test

Server: Sip EXpress router (0.9.6 (i386/linux))

Content-Length: 0

Warning: 392 128.39.145.104:5060 "Noisy feedback tells: pid=12416

req_src_ip=128.39.145.250 req_src_port=57621 in_uri=sip:user1@agder-ikt!04.hia.no

out_uri=sip:user1 @open-ims.test via_cnt==1"

Then use registered IMS client "Userdemo2" and SIP client "user2" to call " user1@open-ims.test " and let the call redirect to open-ims domain "user1@open-ims.test".

In this experiment, the INVITE request didn't reach the destination "user1@open-ims.test".

When used IMS client to initiate the call, after the client had been told the new location of callee, there was no new INVITE request sent out to the new address. And when used SIP client to call "user1@open-ims.test". The new INVITE request only reached the P-CSCF in the OpenIMS and the P-CSCF sent another "300 Redirect" message instead of forwarding the request to the terminal client. In figure 4.23 below us show the process for this experiment when used SIP client "user2" initiates the call.

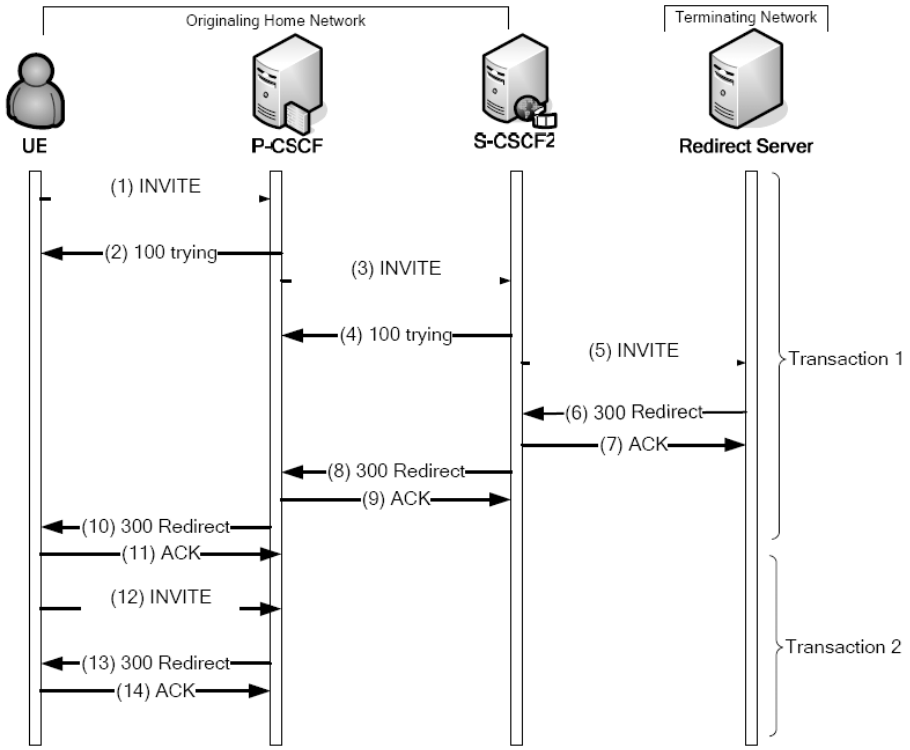

Figure 4.23 Call session setup between two domains

As see from the figure 4.23, there are two transactions in this dialog. For transaction 1 the CSeq equals 1, and for transaction 2 the CSeq value should equals 2 because it happened after transaction 1, but the Call-IDs for those two transactions should be the same since they are in the same dialog. The (1) INVITE, (10) 300 Redirect, (11) ACK, (12) INVITE and (13) 300 Redirect messages in Appendix C.

Wireshark trace for this call dialog and found out that the situation for INVITE requests in transaction 1 and in transaction 2 had different CSeq values and the same Call-ID values as they are suppose to, However, the "300 Redirect" response sent by P-CSCF in transaction 2 still had the same CSeq value as in the transaction 1. This shows that the P-CSCF still treats the new INVITE request to "user1@open-ims.test" as the resend INVITE request to "User1@open-ims.test". so it still responds with "300 Redirect" message instead of forwarding the request to the terminal user.

4.11 Evaluation of "client-based" solution

4.11.1 NAT issues

Since the "client-based" solution is related to two different domains, so it will refer to NAT (Network Address Translator) problem.

NAT is "a method by which IP addresses are mapped from one realm to another in

an attempt to provide transparent routing to hosts". [102] If a SIP server A is behind a NAT gateway, SIP server B which is on another side of NAT will not able to contact server A.

For this situation, open-ims.test domain is located in the Agder Mobility Laboratory network, which has only one external IP address. Therefore, when messages are transmitted within open-ims.test domain there are no NAT issues involved. However, when the messages are transmitted between the open-ims.test domain and open-ims.test domain, the NAT only helps for outgoing communication, that is, when the clients located in the open-ims.test domain want to send messages to clients in open-ims.test domain, the messages can go through the gateway and reach the clients in open-ims.test domain. But when the external clients user1@open-ims.test in domain want to communicate with internal clients in open-ims.test domain, the messages will only reach the gateway but not the internal clients because they only know open-ims.test domain's external IP address.

So need to reconfiguration and let the gateway redirect the traffic. Therefore the messages, which are sent from another domain, can reach the intended recipient.

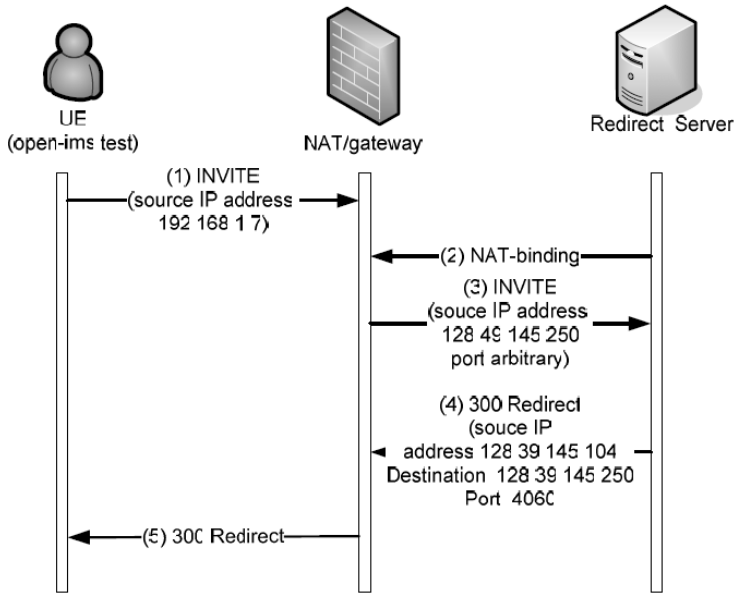

Figure 4.24: Call session setup related to firewall

4.11.2 Configuration of P-CSCF

400 Bad Request—Not following indicated Service-Routes

After reconfigured the firewall, tried to call from user2@open-ims.test to User1@open-ims.test again, but got a "400 Bad Request" response.

Re-evaluate the Wireshark trace and found out that the initiated INVITE

request to User1@open-ims.test and the new INVITE request to user1@open-ims.test after redirection have the same Call-ID. According to the default configuration for P-CSCF, the P-CSCF will reply with "400 Bad Request—-Not following indicated Service-Routes" message when the P-CSCF detects that the Call-IDs are the same for different INVITE request; therefore got the 400 response.

To solve this problem, changed the commands in the P-CSCF configuration file, letting it ignore the Call-ID problem and enforce the routes and let the dialog continue.

```
•       sl_send_reply("400","Bad Request - Not following indicated dialog routes");
•       break;
•       #Variant 2- enforce routes and let the dialog continue
>   # P_enforce_dialog_routes("term");
……
```

Figure 4.25 original commands in pcscf.cfg

```
•       #sl_send_reply("400","Bad Request - Not following indicated dialog routes");
•       #Variant 2- enforce routes and let the dialog continue
•       >   P_enforce_dialog_routes("term");
•       >   break;
```

Figure 4.26 changed commands in pcscf.cfg

As shown in figure 4.25 and figure 4.26, changed the commands, so the P-CSCF no longer sends "400 Bad Request" response. Instead, it will enforce routes and let the dialog continue.

4.12 Performance evaluation of OpenIMS

The OpenIMS testbed and have implemented the possible solution for interoperability between SIP and IMS. From the formal experiments, know that the OpenIMS can work as well as the solution for interoperability between SIP and IMS. The task now is to evaluate if the OpenIMS works well enough. This experiments on performance in following concerns:

- Number of users OpenIMS handle.
- System ability to handle user's requests.
- System support for provides normal operation as well for abnormal operation that fail to achieve expected result.

- Total latency for the system to respond users' requests, and the variations in this time.
- Time interval and check how many users' request the system can handle during that period.

In this part, choose SIPp as this test tool, for the reason that it is very powerful, flexible and free so that can send all kinds of RFC3261 message with it.

4.12.1 Testing steps

As refer to the command using in SIPp, there are several pre-requests before the test. The xml file and the csv file need to be written correctly first. Besides, in order to test how many users OpenIMS can handle, need many subscribers existing in FHoSS firstly.

4.12.1.1 Add 100 subscribers in databases

As described in the first test case, the key problem is showing how many users OpenIMS can handle lies in how to get more subscribers in FHoSS as there are just two default ones in it. Add several ones by manually to test if the OpenIMS works, but it is time consuming to add hundreds of thousands of subscribers in this way.

In this case, decide to generate 100 subscribers automatically by using simple Java programming. 'AddUser.java' is the file to generate 100 subscribers. It is attached in the Appendix D. In this file, generate 100 new subscribers with the names ranging from user0001 to user 0100. Take user 0010 as an example. Its accordingly username is userproc1@open-ims.test and the keyword is userproc1. The other subscribers are configured in the same way. This should be confirmed because they will be used in xml and csv files for registration and call session.

After writing the Java file, need insert it in the sql file, which can find under the directory */opt/OpenIMSCore/FHoSS/deploy/userdata.sql.* Then, the new 100 subscribers are established in the databases, and check them in the FHoSS user profile.

Note that the new 100 subscribers are created as SIP clients who register to the S-CSCF2 with authentication algorithm Digest-MD5. The reason is that initiate the testing in the simplest way for SIP client - SIP client. And if time permits, will extend it to SIP client - IMS client and IMS client - IMS client.

4.12.1.2 Write the xml files

The xml files are the most important parts are using SIPp. Write this own SIPp scenarios including *sip-reg.xml, sip-inv-uac.xml* and *sip-inv-uas.xml* basics on the templates gotten from [97]. The xml files are attached in the Appendix E and F. They are the cases for register and session set-up on the OpenIMS platform.

4.12..1.3 Use the commands to test

The following commands are using for registration and call sessions. The scenarios enable direct communication with the P-CSCF. And they can be called after saving the relevant files as xml-files.

sipp -sf reg-user2.xml 192.168.1.7:4060 -i 192.168.1.3 -p 5061 -m 1 sipp -sf reg-user1.xml 192.168.1.7:4060 -i 192.168.1.3 -p 5060 -m 1 sipp -sf uas-user2-user1.xml 192.168.1.7:4060 -i 192.168.1.3 -p 5060 -m 1 sipp -sf uac-user2-user1.xml 192.168.1.7:4060 -i 192.168.1.7 -p 5061 -m 1

From these commands, can see that the IP address of registrar is 192.168.1.7:4060 which towards to P-CSCF of OpenlMS. And the local IP address is 192.168.1.3, while user2 uses 5061 to register and user1 uses 5060 to register.

-m means how many times the experiment will run. And in the above command, it just runs once.

sipp -sf reg.xml -inf reg.csv 192.168.1.7:4060 -i 192.168.1.3 -p 5061 -m 10 The above command is another situation. It uses 10 different subscribers to register. The *reg.xml\\\e* is shown as follows:

REGISTER sip:[fieldO]@open-ims.test SIP/2.0

Via: SIP/2.0/[transport] [local_ip]:[local_port]

From: **[fieldO]<sip:[fieldO]** @open-ims.test>

To: [fieldO]<sip:[fieldO] @open-ims.test>

Call-ID: [call_id]

CSeq: 2 REGISTER

Contact: sip: **[fieldO]** @ [local_ip]: [local_port]

[fieldl]

Content-Length: [len]

And the *reg.csv\s* shown as follows:

user2; [authentication username=user2@open-ims.testpassword=user2] user1; [authentication username=user1@open-ims.test password=user1]

userOOOl; [authentication username=user0001 @open-ims.test password=user0001] user0002; [authentication username=user0002@open-ims.test password=user0002] user0003; [authentication username=user0003@open-ims.test password=user0003] userOCKM; [authentication username=user0004@open-ims.test password=user0004] userOOOS; [authentication username=user0005@open-ims.test password=user0005] user0006; [authentication username=user0006@open-ims.test password=user0006] userOOOV; [authentication username=user0007@open-ims.test password=user0007] userOOOS; [authentication username=user0008@open-ims.test password=user0008]

As shown from the above two figures, the blue color username is mapping to [field 0], while the red color authentication information is mapping to [field 1] in registration using for return 401 un-authentication message.

Note that the xml file and csv file are different from registration cases in regards to when to test call session.

```
INVITE sip:[fieldO]@open-ims.test SIP/2.0
Via: SIP/2.0/[transport] [local_ip]:[local_port]
From: [fieldO]<sip:[fieldO] @open-ims.test>
To: [fieldl]<sip:[fieldl] @open-ims.test>
Call-ID: [call_id] Content-Length: [len]
```

```
user2; user1
user1; userOOO1
userOOO1;user0002
user0002; userOOO3
userOOO3; user0004
user0004; userOOO5
userOOO5; user0006
user0006; userOOO7
userOOO7; userOOO8
userOOO8; user0009
```

These two figures show SIPp making 10 calls that are initiated by blue color users inviting red color users.

4.13 Hands on for OpenIMS performance assessment

When use all 100 subscribers to register, the SIP signaling process is the same as it registers using only one subscriber, which means the OpenIMS can support 100 subscribers to registering in the same period.

When the experiments for making a call session, design the cases as following:

First, make 20 calls from SIPp to UCT IMS client, which means use 'user2' to call 'Userdemo2' (SIP to IMS client). In each call, the caller just sends one invite message. The results can be shown in the table 4.5 below.

Table 4.5 SIPp-UCT IMS client (SIP to IMS)

20 calls with 1 call each time	Number	Reason
Successful calls	19	
Failed calls	1	603 Decline error

And then make the same situation from SIPp to X-Lite, which means use 'user2' to call 'user1' (SIP to SIP client). The table 4.6 shows the numbers.

Table 4.6 SIPp-X-Lite (SIP to SIP)

20 calls with 1 call each time	Number	Reason
Successful calls	18	
Failed calls	2	600 Busy Everywhere error

The next case that design is to make 20 calls, where the caller will send 30-invite message each time. Choose 30 calls each time because the SIPp is limited in sending 30 successful calls. The numbers are too small to see which way of using in OpenIMS network is better.The table 4.7 shows the situation from SIPp to UCT IMS client (SIP to IMS)

Table 4.7 SIPp-UCT IMS client (SIP to IMS)

20 testing cases with 30 calls each time	Original calls	Successful calls
1	30	14
2	30	7
3	30	15
4	30	10
5	30	8
6	30	5
7	30	7
8	30	7
9	30	8
10	30	9
11	30	13
12	30	10
13	30	10
14	30	8
15	30	9
16	30	5
17	30	14
18	30	15
19	30	9
20	30	11

The table 4.8 shows the situation from SIPp to X-Lite (SIP to SIP)

Table 4.8 SIPp – X-Lite (SIP to SIP)

20 testing cases with 30 calls each time	Original calls	Successful calls
1	30	28
2	30	17
3	30	29
4	30	19
5	30	19
6	30	24
7	30	25
8	30	28
9	30	18
10	30	21
11	30	17
12	30	20
13	30	18
14	30	15
15	30	16
16	30	18
17	30	7
18	30	20
19	30	15
20	30	21

From these two tables and the numbers, the graphs can be compared as follow, in the figure the lengthways axes shows the number of the successful calls while the horizontal axes shows the sequence number of this testing cases.

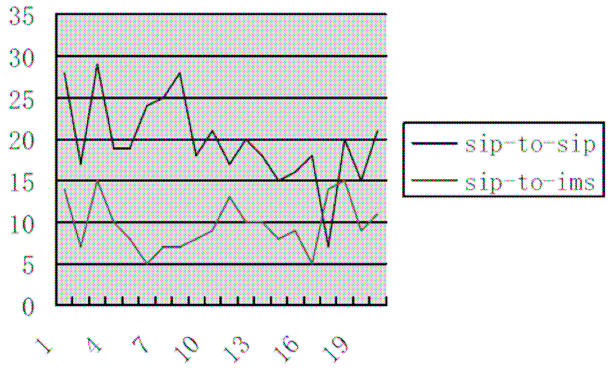

Figure 4.27 the difference cases access to OpenIMS

It is very clear from this graph that the performance of SIP-to-IP client is much better than the SIP-to-IMS client when accessing to the OpenIMS.

As for the reliability of OpenIMS, it is not so good for SIP client. The reason is that the SIP client registers to the S-CSCF2 that is added by us and the information of subscribers around it are always changing randomly. In that case, always have to change the databases. But for the IMS client, the situation is much better because the IMS client register to the S-CSCF.

The OpenIMS can work well in the normal situation, as well as in the abnormal situation.

- When make infinite calls through OpenIMS, it can work well for the first period, but it will become overloaded as the calls increase. The 600 busy will be presented to warn.
- The other situation is that a **408 time out** will be shown.
- When the OpenIMS finds that the user is not in the databases, it will show 403 **HSS forbidden** to inform operator to add it firstly.

And also design the other situations. Define the fixed seconds to see how the message package changes in each procedure. The fixed time is 150 seconds, and the caller will send 30 calls each time. The table 4.9 shows the clear changes from SIP to UCT IMS client.

Table 4.9: SIP to UCT IMS

	10s	15s	20d	30d	40s	50s
REGISTER	30	30	30	50	47	43
401	4	30	30	14	47	43
REGISTER	4	30	30	14	47	43
200 OK	4	21	20	14	37	34
INVITE	4	21	20	14	37	34
100 trying	4	21	20	14	37	34
101 Dialog Establishment	4	17	14	14	27	28
180 Ringing	0	2	0	1	0	3
200 OK	0	0	0	0	0	0
ACK	0	0	0	0	0	0

The table 4.10 is almost is the same as the above one, just for SIP to SIP.

Table 4.10 SIP to SIP

	10s	15s	20s	30s	40s	50s
REGISTER	46	50	57	50	60	70
401	45	25	53	50	60	70
REGISTER	45	25	53	50	60	70
200 OK	35	25	41	42	49	65
INVITE	35	25	41	42	49	65
100 trying	35	25	41	42	49	65
101 Dialog Establishment	0	25	0	0	0	0
180 Ringing	0	0	0	0	0	0
200 OK	0	0	0	0	0	0
ACK	0	0	0	0	0	0

From the data of the above two tables, can easily find that the performance of SIP-to-SIP is better than performance of SIP-to-IMS when they access the OpenlMS. But the SIP-to-IMS is much more stable than SIP-to-SIP.

4.14 Functionality evaluation of OpenlMS

4.14.1 Evaluate SIP and IMS clients

4.14.1.1 UCT IMS

The UCT IMS client who was created directly for OpenlMS is much more stable than the other kinds of clients. It has very simple interface so it's easy to operate. However, the functionalities of UCT IMS client are limited. For example, it doesn't support multiple accounts simultaneously.

4.14.1.2 X-Lite

X-Lite was used as SIP client and it was impressive to use. It has a very nice and easy user interface with all the common controls; therefore, it's easy to operate. The functionalities of X-Lite are considerable, for example, it supports multiple simultaneous connected accounts. However, sometimes it's unstable.

4.14.1.3 Grandstream GXP-2000

GXP-2000 was also used as SIP client for this project. It has powerful functionalities and it's more stale compare to X-Lite. GXP-2000 has a web interface that can be used to configure general or advanced settings and up to four accounts. But found out it's not so easy to operate. For example, by using the keys of the phone to dial username, to use phonebook and the processes are

very complex. And every time after change the information for accounts, need to reboot it for updating and this take some time.

Table 4.11 compare of three SIP/IMS clients

	Reliability in OpenIMS	**Functionality**	**Operation**
UCT IMS client	Most steady	Less function	Middling
X-Lite 3.0 SIP client	Least steady	More function	Easy to operate
GXP-2000 SIP client	Middling	More function	Hard to operate

4.14.2 Functionality evaluation

To evaluate the functionalities of OpenIMS is very time-consuming. Because 3GPP TS 24.229 Release 6 has many sub-clauses described for different situation, so need to check it very carefully and that took us some time.

The testing results showed that IMS clients' scenarios conform to 3GPP TS 24.229 better than SIP clients' scenarios. But all in all, for both IMS and SIP clients, the results are mostly conform to the specification. For detail information, this is portraying in Appendix B.

4.14.3 Solutions of interoperability between IMS and SIP

Two S-CSCFs support different authentication algorithm for SIP clients and IMS clients at the same time installed. Although the solution is feasible.

We to support SIP clients added this part that gave instability of S-CSCF2 that. After each registration of call, couldn't continue to repeat the testing, but to get **403 errors - HSS returned no authentication vectors.** That is because the users' data in MySQL always change automatically and randomly.

In this stage, also met another problem that is the remote client always released the call sessions automatically as soon as the UCT IMS client received calls from X-Lite. However, after use SIPp as SIP clients instead X-Lite, this problem no more appeared. This is due to the drawback of X-Lite.

4.14.3.1 Integration with SIP/VoIP solutions

In this task tried to implement "client based" solution for interoperability of non-IMS domain and IMS domain.

This solution was come up in [53] and was based on using Asterisk as SIP PBX. But to be supported by the idea, more functionality from the clients is required. The client has to be SIP enabled and domain setting configurable, so it can send SIP message to SIP PBX to inform location changing. Many clients are not supported by this solution. Although this problem can be worked out in some situations, for example, "if the enterprise SIP PBX has a web GUI to maintain

registrar, the current location can be updated through the Internet" [53], it is still not very flexible.

Under this situation, considered to use openSER as a redirect server instead Asterisk, so there are no extra requirement for clients.

Since this solution refer to two domains, so the NAT issues also considered in the project. However, this solution hasn't been implemented completely. Clients could register to non-IMS network with redirect server. After redirect server told the caller about the redirect information, caller resend a new INVITE request to callee who located in OpenIMS domain in order to establish the call session.