**An IMS-based IPTV Network**

**ECE 576: Professor Perros**

**Group 7**

**Yuri Kolesnikov**

**Nishant Karajgikar**

**Nikhil G. Khatu**

**Executive Summary**

**Problem Description**

The defined project goal was to implement IPTV services over IMS Application Server with the Open-IMS/Mobicents open source platform. Upon completion of the prototype, the proof of concept can be presented as the Next Generation Network IP streaming services.In creating the prototype one can gain familiarity with the principles behind IMS-based architecture and obtain hands-on experience of the subtleties in network service development.

**Results**

The functional IPTV prototype affords the user a basic level of control over the service. The functionality provided is fairly threadbare but allows the user to start, stop, rewind and switch channels at any time. Although, this developed functionality already exists, the team was able to obtain a comprehensive understanding of the Open-IMS/Mobicents platform. Insight of commercial IPTV network setup was gained along with following the development process.

**Problem definition**

**Problem under study**

Current IMS implementations are by and large non-NGN based. While they do allow for some interworking capabilities, most of the services are controlled by dedicated service control and application layer functions developed specifically for that particular service. Other implementations generally involve an NGN-based approach. The goal of our project is to implement the IPTV service over an IMS backbone. In doing so, we hope to obtain a clearer picture of the IPTV service and it’s associate components.

**Problem background**

**Similar Implementations:**

IPTV can be implemented in three different approaches:

1) Non-NGN based approach - in which limited interworking capabilities are supported. However, most of the services are controlled by dedicated service control and application layer functions.

2) NGN based approach - in which greater interworking capabilities and interactivity is supported over specific reference points between common and existing NGN components such as the Resource Admission and Control Subsystems.

3) IMS based IPTV architecture - It involves the usage of specific IPTV functions by way of IMS. It uses these functions in conjunction with pre-existing IMS functions. An important feature is that the setup of the session happens over the Session Initiation Protocol.

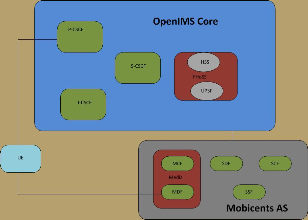
**SWOT Analysis**

|  |  |
| --- | --- |
| **Strengths** | Working prototype shows proof of concept. Can be scaled up for a real world deployment. |
| **Weaknesses** | RTSP/RTP dependent on bandwidth and susceptible to high latency. IPTV is generally implemented with proprietary industry solutions; the barriers to entry are costly. Prototype is based on Open source platforms with licensing conflicts. |
| **Opportunities** | Added functionality promotes IMS infrastructure: Video On-Demand, Live Streaming, Skip, Rewind. Opportunity to gain market share. |
| **Threats** | Non-IMS based infrastructure well developed. Has majority of current market share. |

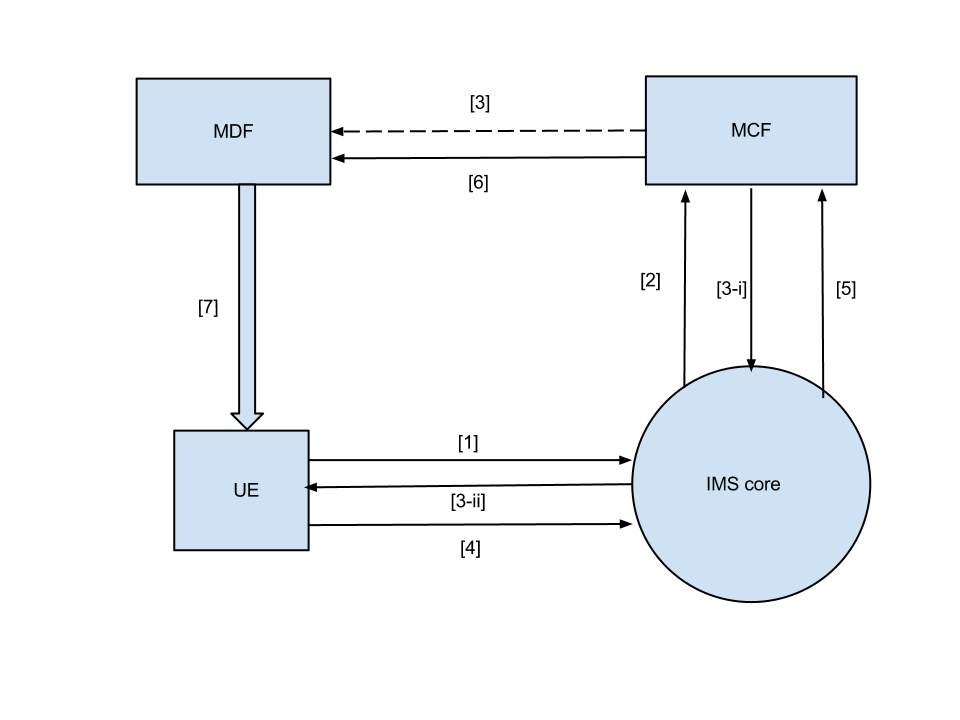
**Problem formulation**

**Details of design**

The major inspiration for the setup of the systems was the ETSI TS 182 027 document [1], which comprehensively covers the standard for established IPTV architectures. In accord with this documentation, the initial design is presented.



The original design included the use of the Open IMS core connected to SIP servlets encapsulated by the Mobicents MSS framework. The team initially developed the IPTV solution by modelling the following signal flow used in both the ETSI documents and the Networking Services book [2] :



[1] - UE contacts the IMS Core with the ‘REGISTER’ message.

[2] - The IMS core authenticates the user and forwards the message to the Media Control Function.

[3] - The third step has two important parts. The MCF verifies the user and looks up the user’s access privileges in the database. Then, it communicates with the user to let it know about the channels available to the user. Concurrently, the MCF informs the MDF to get the stream ready.

[4] - The UE makes its decision and relays that information to the IMS core.

[5] - The IMS Core forwards the user decision to the MCF.

[6] - The MCF informs the Media Delivery Function of the chosen stream.

[7] - The MDF streams the content to the UE.

**Technical problems**

During prototype development of the IPtv over IMS solution, we ran into a number of issues. Most of them arose due to the relatively obscure nature of the open-ims/mobicents platform. Lack of supporting documentation left the team trying to create and maintain a stable support system that included the Operating System, IMS Core, JAVA, Eclipse, VLC, Mobicents, and other supporting packages. The interdependencies made it time intensive to create a stable system. Some compatibility issues included are: the VLC-J library used for the UE client only operated on the Ubuntu 11.10 operating system due a requirement for VLC 2.0 or higher to be installed on the system. However, the configured IMS core/mobicents platform used for development only worked on the Ubuntu 11.04 operating system. Thus, integrating the systems did take a fair amount of tweaking. The problem was finally resolved by using an older version of the VLCJ library in the server running Ubuntu 11.04. The learning curve has been steep, yet there was much learnt through the development process.

**Steps Followed**

**Step 1:** Setup of development environment / Literature Study.

**Step 2:** Documentation of interfaces and functionality of customized service function.

**Step 3:** Configuration and setup of IMS core.

**Step 4:** Setup of JBoSS platform.

**Step 5:** MCF and MDF configuration with VLC libraries

**Step 6:** Development of added functionality

**Step 7:** UE Client Development

**Step 8:** Integration of Modules

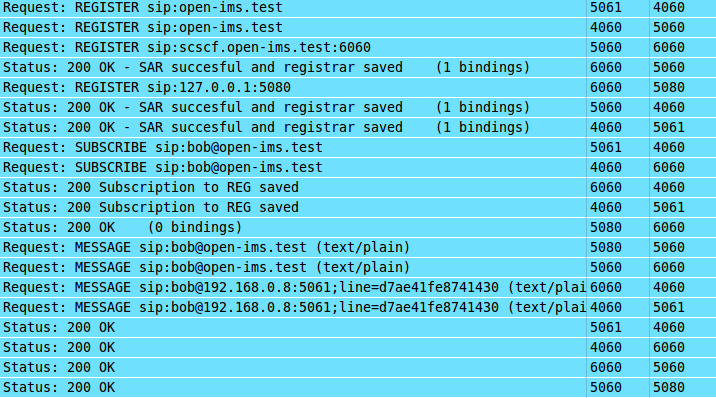
**Step 9:** Testing and verification

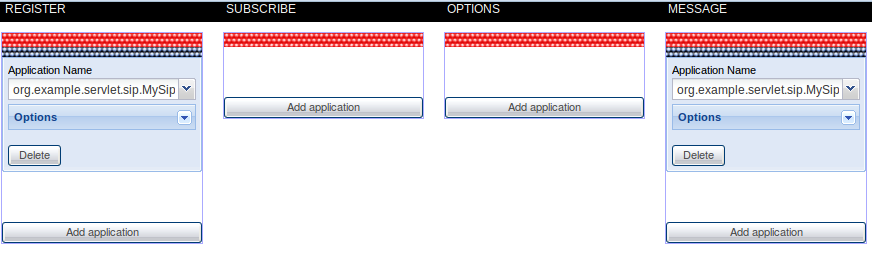
**Solution**

**Prototype**

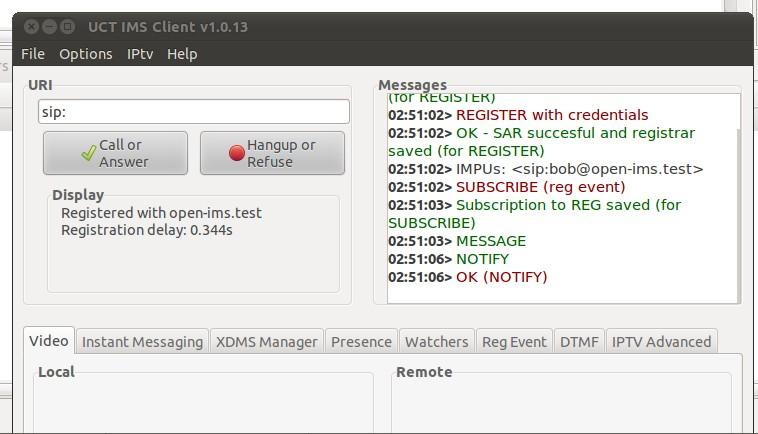
The team developed a prototype that applied open source components: Open-IMS Core, Mobicents on the JBOSS platform, UCT IMS Client, and VLC 2.0 Library. The implementation has the Core and Mobicents(with JBOSS) on the same platform using the loopback as the default ip address. Each function within the setup has an assigned port. (Listed in the appendix) While running wireshark one can view the SIP message exchanges between the Open-IMS Core and the Mobicents SIP Servlet.

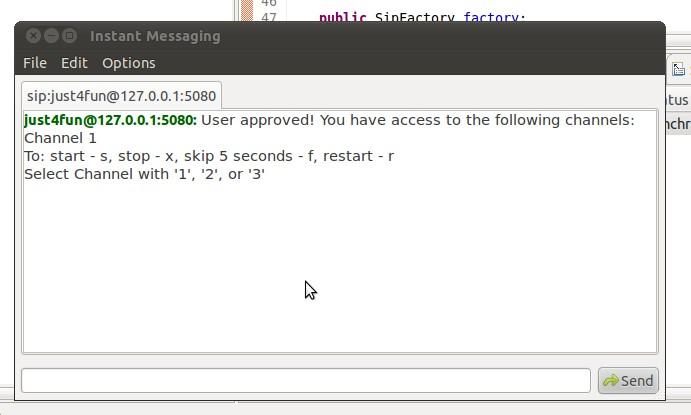
The image below shows a “REGISTER” message arriving at the Open-IMS scscf and being forwarded onto the SIP servlet at sip:127.0.0.1:5080 for processing.



The Open IMS core provides a web based management system where the user base, trigger points, filter criteria, and service profiles can be edited. The system is configured to forward SIP messages to the Mobicents Application Server. The Mobicents AS implements an Application Router that forwards SIP messages via a web managed servlet interface. The ‘MySipServlet’ class is fowarded the SIP Register and Message data.

After forwarding is enabled the ‘MySipServlet’ application can process incoming data by forwarding, replying, querying the database, and/or starting the video stream as necessary.





The images above show a client-side view of the application. The instant message window shows information that is generated by querying the mysql server database informing the user of available commands via SIP. The user can then send these commands back to the server through SIP messaging to enable stop/play/pause functionality.

Prototype Walkthrough

* Client registers with the core
* Core forwards the register message to MobicentsAS, in turn forwarded to the ‘MySipServlet’ servlet
* ‘MySipServlet’ queries the UPSF database, checks permission, and returns a string accordingly in the UCT IMS client chat window. (Illustrated above)
* Client is presented with a list of media control options
* IPtv media is streamed from the server over RTSP(Real Time Stream Protocol)[[1]](#footnote-0)

**Conclusions**

**Results Summary**

An IMS based IPTV service was developed using open-source IMS components. The service allows for a client to register to a stream, receive customized service availability data, and finally interact with the stream using SIP messaging. In doing so, the team has gained a comprehensive understanding of IMS as well as obtained a great deal of hands-on experience in setting up a network service.

**Future Plans**

Now that the basic framework is set, there are many possible future innovations that may be added. Using the VLC2.0 library in conjunction with the IMS setup, it should possible to offer the user more advanced options such as parental control, subtitle switching, dedicated advertisements, adaptive picture quality, etc. Additionally, this framework could be the first step to a mashup of network services, such as IPTV+Instant messaging/Social networking.

**References**

1) ETSI TS 182 027

2) Networking Services - QOS, Signalling, Processes - Dr. Harry Perros

3) Interactive, Personalized IPTV - Ignacio Más and Viktor Berggren, Ericsson Research.Rittwik Jana, John Murray, and Christopher W. Rice, AT&T Labs Research

4) IMS based IPTV services - Architecture and Implementation - Eugen Mikoczy, Dmitry Sivchenko, Bangnan Xu, Veselin Rakocevic

**Appendix**

Addressing details:

icscf - 127.0.0.1: 5060

pcscf - 127.0.0.1: 4060

scscf - 127.0.0.1: 6060

hss - 127.0.0.1: 3868

mysql - 127.0.0.1: 3306

Mobicents Application Server - 127.0.0.1: 5080

Default OpenCore Application Server - 127.0.0.1: 5065

Code:

package org.example.servlet.sip;

import java.io.IOException;

import java.util.ArrayList;

//import java.util.List;

import java.util.Properties;

import org.apache.log4j.Logger;

import org.example.servlet.sip.MediaServer;

import javax.naming.Context;

import javax.naming.InitialContext;

import javax.naming.NamingException;

import javax.servlet.ServletConfig;

import javax.servlet.ServletException;

import javax.servlet.sip.ServletParseException;

import javax.servlet.sip.SipApplicationSession;

import javax.servlet.sip.SipFactory;

import javax.servlet.sip.SipServlet;

//import javax.servlet.sip.SipServletMessage;

import javax.servlet.sip.SipServletRequest;

import javax.servlet.sip.SipServletResponse;

import java.sql.SQLException;

import java.util.Scanner;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

//import java.sql.SQLException;

import java.sql.Statement;

import java.util.Date;

//import com.mysql.jdbc.Driver;

public class MySipServlet extends SipServlet {

private MediaServer mediaServer;

private static final long serialVersionUID = 1L;

private static Logger logger = Logger.getLogger(MySipServlet.class);

public final static String USER\_LIST="userList";

private static final String CONTENT\_TYPE = "text/plain;charset=UTF-8";

/\*\* Init parameter key to retrieve the chatroom's address. \*/

public final static String CHATROOM\_SERVER\_NAME="chatroomservername";

public SipFactory factory;

public String serverAddress;

public void doRegister(SipServletRequest request){

System.out.println("REGISTER REQUEST RECEIVED\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

try {

SipServletResponse response = request.createResponse(SipServletResponse.SC\_OK);

response.send();

String from = request.getFrom().getURI().toString();

String toString = request.getTo().getURI().toString();

String returnedString = readDataBase(toString);

sendToUser(toString, returnedString) ;

//logger.info("READ DB STRING: "+returnedString);

//logger.info("FROM IS\*\*\*\*\*\*\*\*\*\* : "+from);

//logger.info("TO IS\*\*\*\*\*\*\*\*\*\*\*\* : "+toString);

//System.out.println("\*\*\*\*\*\* OK RESPONSE SENT FROM Sip Servlet\*\*\*\*\*\*\*\*");

} catch (IOException e) {

e.printStackTrace();

} catch(ServletParseException e){

System.out.println("CAUGHT SERVER PARSE EXPRESSION\*\*\*\*\*"+e.toString());

} catch (SQLException e) {

// TODO Auto-generated catch block

System.out.println("CAUGHT SQL EXCEPTION");

e.printStackTrace();

} catch (ClassNotFoundException e) {

System.out.println("CAUGHT ClassNotFoundException");

e.printStackTrace();

}

}

public void doMessage(SipServletRequest request){

//logger.info("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RECEIVED A MESSAGE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

try {

if(request.getContentType().contains("application/im-iscomposing+xml")){

SipServletResponse response = request.createResponse(SipServletResponse.SC\_OK);

response.send();

logger.info("USER IS TYPING");

return;

} else if(request.getContent().equals("s")) { //start

SipServletResponse response = request.createResponse(SipServletResponse.SC\_OK);

response.send();

String from = request.getFrom().getURI().toString();

String toString = request.getTo().getURI().toString();

mediaServer.startStream1();

sendToUser(from, "Received a start request : "+request.getContent().toString());

} else if(request.getContent().equals("x")) { //stop

SipServletResponse response = request.createResponse(SipServletResponse.SC\_OK);

response.send();

String from = request.getFrom().getURI().toString();

String toString = request.getTo().getURI().toString();

mediaServer.stop();

sendToUser(from, "Received a end request : "+request.getContent().toString());

}else if(request.getContent().equals("p")) { //stop

SipServletResponse response = request.createResponse(SipServletResponse.SC\_OK);

response.send();

String from = request.getFrom().getURI().toString();

String toString = request.getTo().getURI().toString();

mediaServer.pause();

sendToUser(from, "Received a end request : "+request.getContent().toString());

} else if(request.getContent().equals("f")) { //skip

SipServletResponse response = request.createResponse(SipServletResponse.SC\_OK);

response.send();

String from = request.getFrom().getURI().toString();

String toString = request.getTo().getURI().toString();

mediaServer.skip();

sendToUser(from, "Received a fast-forward request : "+request.getContent().toString());

} else if(request.getContent().equals("r")) { //restart

SipServletResponse response = request.createResponse(SipServletResponse.SC\_OK);

response.send();

String from = request.getFrom().getURI().toString();

String toString = request.getTo().getURI().toString();

sendToUser(from, "Received a restart request : "+request.getContent().toString());

} else if(request.getContent().equals("1")) { // Channel 1

SipServletResponse response = request.createResponse(SipServletResponse.SC\_OK);

response.send();

String from = request.getFrom().getURI().toString();

String toString = request.getTo().getURI().toString();

mediaServer.stop();

mediaServer.startStream1();

sendToUser(from, "Received a restart request : "+request.getContent().toString());

} else if(request.getContent().equals("2")) { //Channel 2

SipServletResponse response = request.createResponse(SipServletResponse.SC\_OK);

response.send();

String from = request.getFrom().getURI().toString();

String toString = request.getTo().getURI().toString();

mediaServer.stop();

mediaServer.startStream2();

sendToUser(from, "Received a restart request : "+request.getContent().toString());

} else if(request.getContent().equals("3")) { //Channel 3

SipServletResponse response = request.createResponse(SipServletResponse.SC\_OK);

response.send();

String from = request.getFrom().getURI().toString();

String toString = request.getTo().getURI().toString();

sendToUser(from, "Received a restart request : "+request.getContent().toString());

} else {

SipServletResponse response = request.createResponse(SipServletResponse.SC\_OK);

response.send();

logger.info("COULDN'T RECOGNIZE THE MESSAGE");

return;

}

} catch (IOException e) {

e.printStackTrace();

} catch(ServletParseException e){

System.out.println("CAUGHT PARSE EXCEPTION: ");

e.printStackTrace();

}

}

public String readDataBase(String temp) throws SQLException, ClassNotFoundException {

/\*

\* private Connection connect = null; private Statement statement =

\* null; private PreparedStatement preparedStatement = null; private

\* ResultSet resultSet = null;

\*/

Connection connect = null;

Statement statement = null;

ResultSet resultSet = null;

String blank\_string = "";

DriverManager.registerDriver(new com.mysql.jdbc.Driver());//just an example

// Setup the connection with the DB

// DriverManager.

connect = DriverManager.getConnection("jdbc:mysql://localhost/User\_database?"+ "user=root&password=csc576");

// Statements allow to issue SQL queries to the database

statement = connect.createStatement();

// Result set get the result of the SQL query

resultSet = statement.executeQuery("select \* from User\_database.registered\_users where Name = "+ "'" + temp + "'");

if (resultSet.next() == false) {

blank\_string += "Access Denied!";

} else

{

int channel1Specifier = resultSet.getInt("channel1");

int channel2Specifier = resultSet.getInt("channel2");

int channel3Specifier = resultSet.getInt("channel3");

blank\_string += "User approved! You have access to the following channels:\n";

if (channel1Specifier == 1) blank\_string += "Channel 1\n";

if (channel2Specifier == 1) blank\_string += "Channel 2\n";

if (channel3Specifier == 1) blank\_string += "Channel 3\n";

blank\_string += "To: start - s, stop - x, skip 5 seconds - f, restart - r\n";

blank\_string += "Select Channel with '1', '2', or '3'\n" ;

}

return blank\_string ;

}

private void sendToUser(String to, String message) throws ServletParseException, IOException {

SipApplicationSession session = factory.createApplicationSession();

SipServletRequest request = factory.createRequest(session, "MESSAGE", serverAddress, to);

request.setContent(message.getBytes(), CONTENT\_TYPE);

request.send();

}

public void doCancel(SipServletRequest request) {

System.out.println("\*\*\* Received CANCEL \*\*\*");

try {

request.createResponse(SipServletResponse.SC\_REQUEST\_TERMINATED).send();

} catch (IOException e) {

e.printStackTrace();

}

}

@Override

public void init(ServletConfig servletConfig) throws ServletException {

super.init(servletConfig);

logger.info("The chat room sip servlet has been started ! ");

try {

mediaServer = new MediaServer();

// Getting the Sip factory from the JNDI Context

Properties jndiProps = new Properties();

Context initCtx = new InitialContext(jndiProps);

Context envCtx = (Context) initCtx.lookup("java:comp/env");

factory = (SipFactory) envCtx.lookup("sip/org.example.servlet.sip.MySipServlet/SipFactory");

logger.info("Sip Factory ref from JNDI : " + factory);

} catch (NamingException e) {

throw new ServletException("Uh oh -- JNDI problem !", e);

}

getServletContext().setAttribute(USER\_LIST,new ArrayList<String>());

serverAddress = getServletConfig().getInitParameter(CHATROOM\_SERVER\_NAME);

logger.info("serverAddress is : " + serverAddress);

}

}

1. The RTSP protocol implements stream control of RTP/UDP. [↑](#footnote-ref-0)