BAB 3
METODOLOGI PENELITIAN

Penelitian ini menggunakan *open source* dari Franhoufer Institute FOKUS yaitu *OpenIMSCore* yang terdiri dari *Call session Control Function* (CSCF) berperan sebagai elemen pusat dari *routing* untuk pensinyalan IMS dan *Home Subscriber Server* (HSS) yang disebut dengan FHoSS untuk memanejemen *user profiles* dan pengaturan routing [10] dan dari Communication Research Group University of Cape Town yaitu uctimslient, dan uctiptv yang berperan sebagai *client* dan server.

3.1 *Install OpenIMSCore*

OpenIMSCore merupakan *open source* yang beroperasi pada sistem operasi yang *open source* yaitu linux dan distronya. Pada penelitian ini dipakai sistem operasi Ubuntu. Agar dapat beroperasi, ada beberapa paket pendukung yang harus ada pada Ubuntu yaitu : *sun-java6-jdk, subversion, mysql-server, libmysqlclient15-dev, libxml2-dev, bind, ant, flex,* dan *bison. Install* paket ini dari *synaptic package manager* atau dari terminal :

sudo apt-get install subversion, sun-java6-jdk, mysql-server, libmysqlclient15-dev, libxml2, libxml2-dev, bind9, ant, flex, bison.

Pastikan pada terminal : # java –version (JDK>= 1.5)
java version "1.6.0_0"
IcedTea6 1.3.1 (6b12-0ubuntu6.6) Runtime Environment (build 1.6.0_0-b12)
OpenJDK Client VM (build 1.6.0_0-b12, mixed mode, sharing)

3.1.1 Mendapatkan Sumber Kode

Langkah pertama adalah membuat tempat dimana sumber kode akan diletakkan, seperti langkah berikut :

a. Buat direktori /opt/OpenIMSCore/
b. Pada direktori /opt/OpenIMSCore/ buat direktori ser_ims

Sumber kode dapat diunduh dari : *svn checkout http://svn.berlios.de/svnroot/repos/openimscore/ser_ims/trunk ser_ims*

c. Pada direktori /opt/OpenIMSCore/ buat direktori FHoSS

Sumber kode dapat diunduh dari *svn checkout http://svn.berlios.de/svnroot/repos/openimscore/FHoSS/trunk FHoSS*

3.1.2 Kompilasi Sumber Kode

Setelah langkah tersebut diatas selesai maka dilakukan penyusunan (*compile)* pada kedua direktori tersebut :

Pada direktori ser_ims : *sudo make install-libs all*

Pada direktori FHoSS : *sudo ant compile deploy*

3.1.3 Konfigurasi DHCP dan DNS

Konfigurasi awal dapat dilakukan pada *localhost*, edit file dari /etc/dhcp3/dhclient.conf dan aktifkan dari baris *prepend domain-name-servers 127.0.0.1*. File open-ims DNS digandakan dan diletakkan pada bind folder: *sudo cp /opt/OpenIMSCore/ser_ims/cfg/open-ims.dnszone /etc/bind/*. Pada file /etc/bind/named.conf.local, tambahkan perintah

```
zone "open-ims.test" {
        type master;
        file "/etc/bind/open-ims.dnszone";
        };
```

, dan kemudian tambahkan perintah pada file /etc/resolv.conf :

```
nameserver 127.0.0.1
search open-ims.test
domain open-ims.test
```

Kemudian *restart bind* tersebut agar aktif : *sudo /etc/init.d/bind9 restart.*

pastikan dengan melakukan *ping pcscf.open-ims.test* akan ada respon.

3.1.4 Membentuk *Database*

Proses ini bekerja pada mysql, dengan membuat suatu database :

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

mysql -u root -p -h localhost < FHoSS/scripts/hss_db.sql

mysql -u root -p -h localhost < FHoSS/scripts/userdata.sql

3.1.5 Konfigurasi dari IMS *Core*

Konfigurasi server open-ims dengan file *configurator.sh* sesuai dengan nama domain dan IP *address* server.

Agar lebih memudahkan dalam menjalankan *OpenIMSCore* , gandakan file ekstension cfg, sh, dan xml dari ser_ims ke *OpenIMSCore* :

cp ser_ims/cfg/.cfg .*

cp ser_ims/cfg/.xml .*

cp ser_ims/cfg/.sh .*

Pada direktori …/FHoSS/deploy/ terdapat file startup.sh dimana dilakukan penggantian file *Java home* sesuai letaknya pada komputer.

3.1.6 Menjalankan *OpenIMSCore*

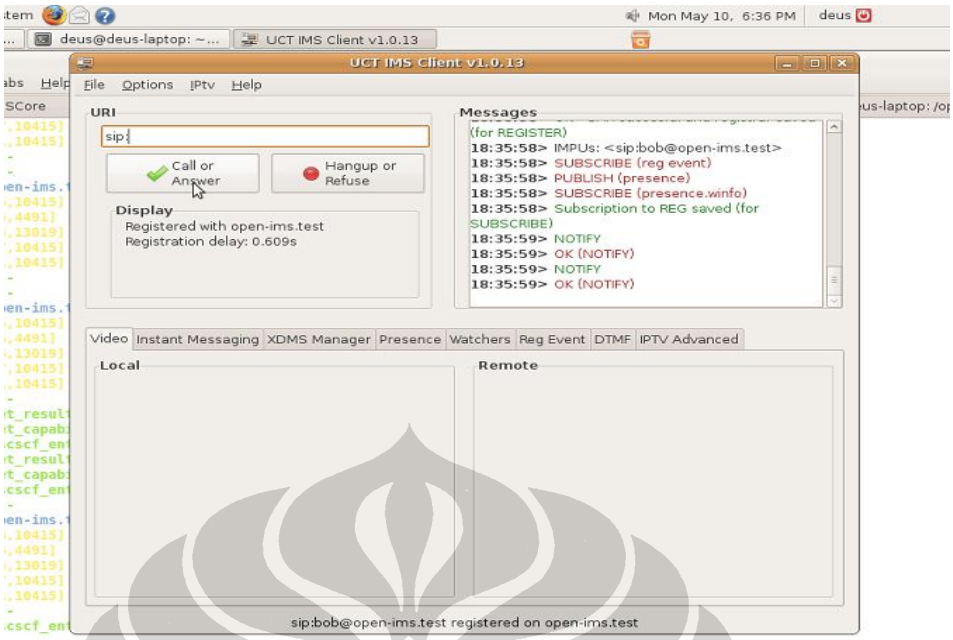
Untuk menjalankan *opensource* ini, buka direktori *OpenIMSCore* dan jalankan file pcscf.sh, icscf.sh, scscf.sh, dan fhoss.sh secara pararel. Lakukan pengecekan dengan membuka web *interface* pada http://localhost:8080/ dengan *login* hssAdmin dan *password* hss.

3.1.7 Mengubah Nama Domain dan Alamat IP

Agar dapat dilakukan pengujian didalam jaringan maka alamat IP *local host* harus diubah, perubahan alamat IP dan nama domain berada pada file *configurator.sh* yang tersimpan pada direktori /opt/OpenIMSCore/ser_ims/cfg/. Demikian juga pada file /etc/named.conf serta /etc/bind/open-ims.dnszone.

3.2 *Install IMS Client* dan IPTV *Streaming* Server

Untuk IMS *client* dan server IPTV pada *openimscore* dapat diunduh dari web http://uctimsclient.berlios.de/ [11] dengan memperhatikan paket pendukungnya. Untuk menguji dari *OpenIMSCore* yang dijalankan, kita menjalankan *uctimsclient*, default dari aplikasi ini adalah alice dan bob. Jika berhasil akan tampil sebagai berikut seperti Gambar 3.1.



Gambar 3.1 *Register* dari *uctimsclient* sukses

3.3 *Wireshark*

Wireshark adalah sebuah program *network packet analyzer* yang melakukan *capture* paket-paket pada *network* dan menampilkannya secara terperinci. Wireshark digunakan untuk melakukan analisa jaringan komputer dengan melakukan pengukuran beberapa parameter QoS seperti *jitter*, *delay*, *packet loss*, dan *throughput* serta mampu melakukan *capture protocol* yang sedang berjalan dalam jaringan tersebut. Wireshark dapat didownload secara gratis pada *website* www.wireshark.org.

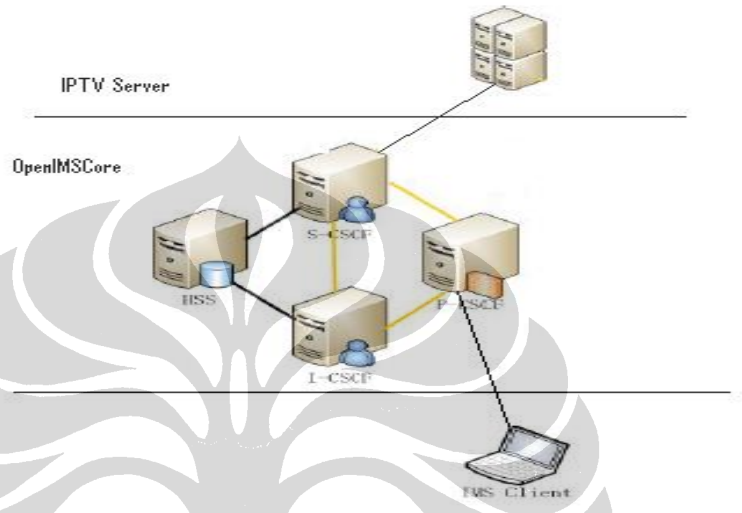
3.4 Kebutuhan Perangkat Keras

Adapun kebutuhan perangkat keras ini adalah kabel UTP, konektor RJ.45, hub, laptop, dan *access point*. Dengan spesifikasi sebagai berikut :

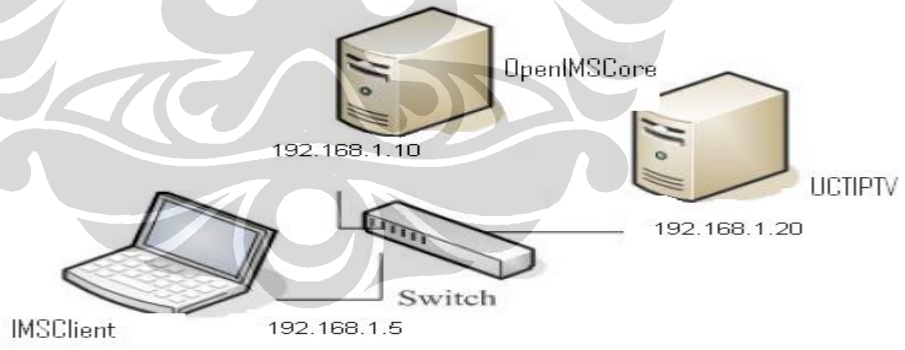
a. Laptop intel dual core 1.83 Ghz, memory 1 Gb sebagai open-ims.test
b. Laptop intel dual core 1.83 Ghz, memory 3Gb sebagai server IPTV
c. Laptop intel dual core 1.83 Ghz, memory 512 Mb sebagai open-ims.test1
d. Laptop intel dual core 1.73 Ghz, memory 1 Gb sebagai *client*
e. *Access point* 2.4 Ghz.

3.5 *Testbed* IMS pada *Platform* IPTV

Untuk melakukan *roaming* IMS berdasarkan layanan pada IPTV, maka perlu diuji IPTV dapat bekerja pada IMS*Core*. Gambar 3.2 adalah sistem fungsi untuk layanan IPTV pada IMS*Core*. Dan untuk infrastruktur *testbed* seperti pada Gambar 3.3.



Gambar 3.2 Sistem fungsi untuk IMS pada layanan IPTV



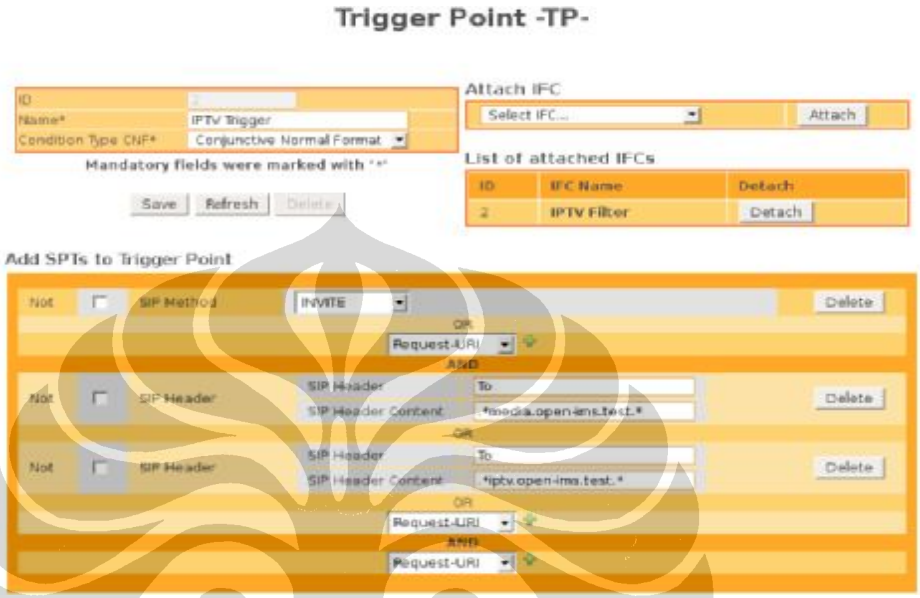
Gambar 3.3 Arsitektur IMS pada layanan IPTV

Adapun langkah-langkahnya adalah sebagai berikut :

1. Jalankan semua komponen *OpenIMSCore*

Pada saat menjalankan semua komponen *IMSCore*, konfigurasi FHoSS seperti pada Gambar 3.4 untuk meneruskan permintaan IPTV ke aplikasi server yaitu :

a. Tambahkan aplikasi server dengan memakai *port* 7070
b. Tambahkan *trigger point* seperti sip: channel1@iptv.open-ims.test atau sip: channel1@media.open-ims.test.
c. Hubungkan aplikasi server dengan *trigger point* dengan *initial filter criteria*
d. Tambahkan iFC ke *default service profile.*



Gambar 3.4 Konfigurasi FhoSS [11]

2. Jalankan IPTV Server, dengan perintah sebagai berikut :

Uctiptv [quality] [channel] [file1] [file2]

Sebagai contoh : uctiptv 500000 2 media/movie.avi media/movie2.avi

Quality adalah kualitas dari *streaming*, pilihan antara 100.000 dan 1.000.000

Channel adalah sebuah nilai antara 1 dan 3 tergantung dari banyak *channel* yang digunakan.

3. Jalankan UCTIMSClient

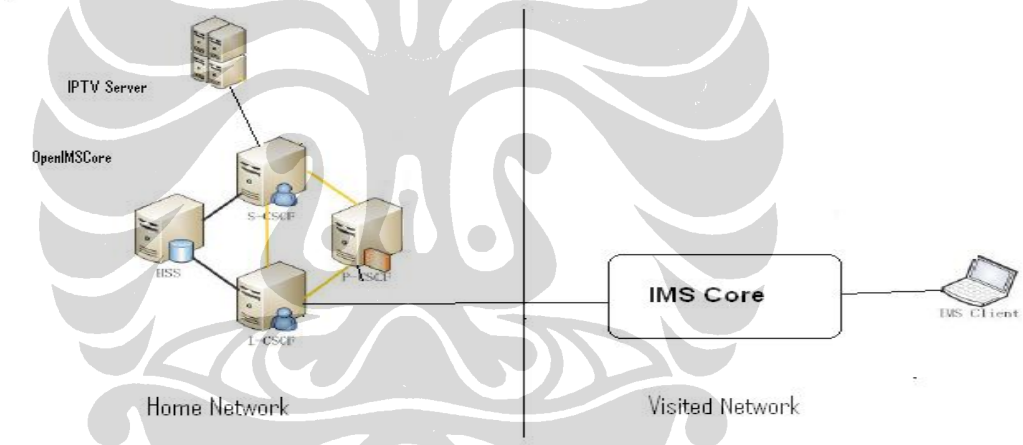
Daftarkan dengan IMS*Core*, atur media server pada media *preference* untuk hubungan dengan *trigger point* yang di set pada FHoSS. Pada *uctimsclient* terdapat *bit rate codec* yang harus diperhatikan yaitu satu untuk video dan satu untuk audio. *Streaming* video di *enkode* menggunakan variabel *bit rate codec* dan audio di *enkode* menggunakan konstan *bit rate codec* [6].

3.6 Skenario *Roaming* IMS pada Layanan IPTV

Suatu pelanggan (*user*) yang kita sebut dengan Alice atau Bob adalah pelanggan dari *home network* dimana pelanggan tersebut akan mengakses layanan diluar dari jaringannya yaitu *home network* atau kita sebut pelanggan berada pada *visited network* . Jaringan dari *visited* adalah jaringan wireless,

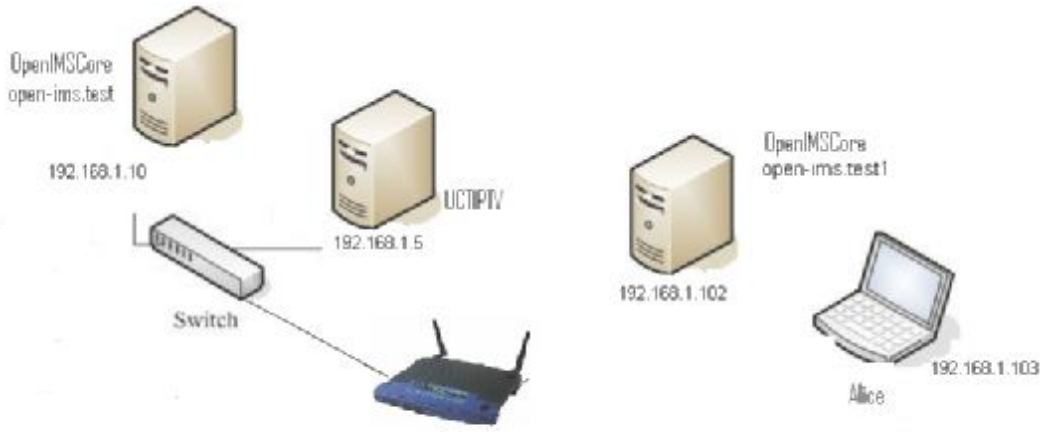
Pelanggan Alice (alice@open-ims.test) terdaftar sebagai *user* pada operator open-ims.test dan data teregister pada hss.open-ims.test. Alice melakukan *roaming*, dimana ia melakukan akses layanan IPTV melalui operator 2 yaitu open-ims.test2. Pada saat uji coba pelanggan akan melakukan perpindahan tempat (bergerak) pada titik dimana wireless tidak terdeteksi dan kemudian kembali dimana pelanggan memasuki jaringan wireless.

Sistem fungsi *testbed* untuk skenario ini dapat dilihat pada Gambar 3.5 dan arsitekturnya seperti pada Gambar 3.6.

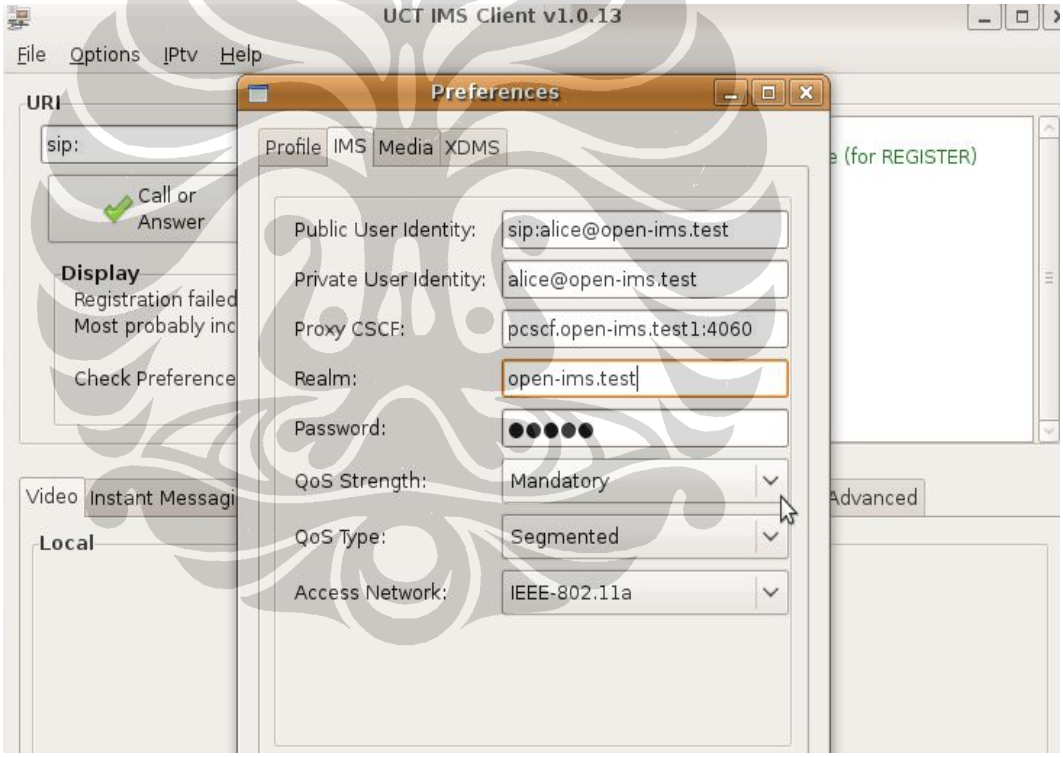


Gambar 3.5 Sistem fungsi *roaming* IMS pada *platform* IPTV

Gambar 3.7 menunjukkan ketika suatu pelanggan akan mendaftar ke jaringannnya yaitu *home network*, setelah ada respon balik, pelanggan dapat melakukan akses layanan IPTV melalui jaringan IMS.



Gambar 3.6 Arsitektur *roaming* IMS pada *platform* IPTV



Gambar 3.7 Pelanggan melakukan *register* ke *home network*

3.7 Quality of Service

Untuk menjamin kesuksesan suatu layanan IPTV harus menawarkan kualitas yang bagus. Suatu jaringan yang kuat menjamin suatu level *Quality of Service* (QoS) yang merupakan langkah utama dari layanan IPTV. QoS untuk

jaringan pengiriman IPTV harus melakukan optimasi untuk meminamalisasi *jitter, delay,* dan *packet loss* untuk streaming video melalui IP.

Pengukuran yang dilakukan pada saat akses di *home network* dan *roaming* adalah *jitter*, *delay*, *packet loss* dan *throughput*.

a. *Jitter* atau variasi kedatangan paket, hal ini diakibatkan oleh variasi-variasi dalam panjang antrian, dalam waktu pengolahan data, dan juga dalam waktu penghimpunan ulang paket-paket di akhir perjalanan *jitter*. *Jitter* lazimnya disebut variasi *delay* ,berhubungan erat dengan *latency*, yang menunjukkan banyaknya variasi *delay* pada transmisi data di jaringan. *Delay* antrian pada *router* dan *switch* dapat menyebabkan *jitter*. Variasi dari jitter paket dengan rumusan sebagi berikut :

$$Jitter = \frac{|\text{jitter } 1| + |\text{jitter } 2| + \ldots + |\text{jitterN}|}{N} s \quad (3.1)$$

b. *Delay* (*latency*), adalah waktu yang dibutuhkan data untuk menempuh jarak dari asal ke tujuan. *Delay* dapat dipengaruhi oleh jarak, media fisik, kongesti atau juga waktu proses yang lama. Delay dengan rumusan sebagai berikut :

$$Delay = \frac{|\text{delay } 1| + |\text{delay } 2| + \ldots + |\text{delay N}|}{N} s \quad (3.2)$$

c. *Packet Loss,* merupakan suatu parameter yang menggambarkan suatu kondisi yang menunjukkan jumlah total paket yang hilang, dapat terjadi karena *collision* dan *congestion* pada jaringan dan hal ini berpengaruh pada semua aplikasi karena *retransmisi* akan mengurangi efisiensi jaringan secara keseluruhan meskipun jumlah *bandwidth* cukup tersedia untuk aplikasi-aplikasi tersebut. Umumnya perangkat jaringan memiliki *buffer* untuk menampung data yang diterima. Jika terjadi kongesti yang cukup lama, *buffer* akan penuh, dan data baru tidak akan diterima. *Packet loss* dihitung dengan rumusan sebagai berikut :

$$\textit{Packet loss} = \frac{\text{Jumlah paket yang hilang}}{\text{Jumlah paket yang diterima}} \times 100\% \qquad (3.3)$$

d. *Throughput*, yaitu kecepatan (*rate*) transfer data efektif, yang diukur dalam bps. *Throughput* merupakan jumlah total kedatangan paket yang sukses yang diamati pada *destination* selama interval waktu tertentu dibagi oleh durasi interval waktu tersebut.

Skenario QoS untuk di *home network* dan *roaming* di *visited network* didasarkan dari kombinasi paramater dari server IPTV yaitu kualitas dari *streaming* (video bit rate) dan *bit rate code* dari uctimsclient pada menu *Preferences*, media (PCMA 64 kbps dan GSM 13,2 kbps) sebagai berikut :

1. Range kualitas dari *streaming* server : 400.000 dan 500.000 dan variasi dari ims *client* yaitu *bit rate codec* dan video *bandwith* (*medium* 60 kbps dan *high* 80 kbps)
2. Range kualitas dari *streaming* server : 700.000 dan 800.000 dan variasi dari ims *client* yaitu *bit rate codec* dan video *bandwith* (*medium* 60 kbps dan *high* 80 kbps)