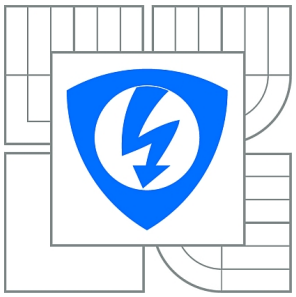
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

MOŽNOSTI PŘENOSU SIGNALIZACE SS7 PŘES IP SÍŤ S VYUŽITÍM ÚSTŘEDNY YATE

THE POSSIBILITIES OF SS7 SIGNALLING TRANSPORT OVER IP NETWORK USING YATE SWITCH

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

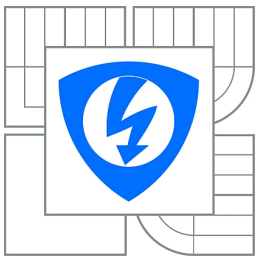
AUTOR PRÁCE — Bc. MHANNAD AL-ANQARI

AUTHOR

VEDOUCÍ PRÁCE — Ing. PAVEL ŠILHAVÝ, Ph.D.

SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Mhannad Al-Anqari ***ID:*** 83956
Ročník: 2 ***Akademický rok:*** 2012/2013

NÁZEV TÉMATU:

Možnosti přenosu signalizace SS7 přes IP síť s využitím ústředny YATE

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte možnosti přenosu signalizačního systému číslo 7 (SS7) přes síť na bázi IP protokolu a protokoly specifikujícím jeho konverzi. Zaměřte se zejména na protokoly SIGTRAN, BICC a SIP-T. Vytvořte experimentální konvergovanou síť s pomocí Open Source PBX YATE (Yet Another Telephony Engine), kde jsou tyto protokoly implementovány. Na základě realizovaných experimentů porovnejte dle Vámi specifikovaných kritérií jednotlivé přístupy. Vytvořte laboratorní úlohu, která umožní seznámení s jednotlivými přístupy. Text vlastní práce bude obsahovat podrobné návody instalace a konfigurace dílčích částí.

DOPORUČENÁ LITERATURA:

[1] Russell, T.. Signaling System #7. McGraw-Hill, New York 2002 , ISBN 0-07-146879-X
[2] Bosse, J.G.. Signaling in telecommunication networks. John Wiley & Sons, Ltd. En-gland 2002 , ISBN 0-471-66288-7.
[3] Freeman, R.L.. Telecommunication system engineering. John Wiley & Sons, 2004, ISBN 0-471-45133-9.
[4] YATE - Main – Documentation. [online], [cit. 2010-10-11]. Dostupné z http://www.yate.null.ro.

Termín zadání: 11.2.2013 ***Termín odevzdání:*** 29.5.2013

Vedoucí práce: Ing. Pavel Šilhavý, Ph.D.
Konzultanti diplomové práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Anotace:

Diplomová Práce se věnuje problematikou signalizačního systému číslo 7 (SS7), a to zejména přenosem signalizace SS7 přes sítě na bázi IP protokolu. K ověření možností přenosu byla vytvořena konvergovaná síť s využitím Open Source PBX YATE, kde jsou potřebné protokoly implementovány. V úvodu diplomové práce je uveden popis signalizačního systému SS7, který je následován vysvětlením funkce každé z vrstev (MTP2 až aplikační) a přenášených zpráv síti SS7. Dále byla věnována pozornost protokolům, které umožňují přenos SS7 přes IP síť. V diplomové práci byla také popsána architektura PBX YATE, konfigurační soubory a způsoby instalace v operačním systému Linux. Rovněž byly stručně popsány důležité soubory pro realizaci této práce. Experimentální práce byly zahájeny s využitím dvojice virtuálních počítačů, které měly naistalovány dvě různé PBX, a to YATE a Asterisk. Dalším krokem byla realizace konvergované sítě pro ověření teoretických předpokladů. Pro tyto účely již byly využity servery s instalovanými TDM kartami. S pomocí těchto serverů byly ověřeny protokoly SS7, SIGTRAN, implementována MGCP brána a protokol SIP-T. Experimenty byly úspěšné, nicméně lze jistě pokračovat dalšími a ověřit další možnosti.

Abstract:

This study examines the use of SS7 signaling system over IP networks by using the open source PBX YATE. At first it starts with describing the SS7 followed by an explanation of the function of each of its levels and the messages that are used within the SS7 network. The study then sheds some light on the ways of using SS7 inside IP network with the use of some protocols. It also discusses the architecture of YATE and its files, and how it is installed in Linux operating system. Finally, it describes the important files for delivering this task. The study was commenced by using two virtual machines that have two different open source PBX's which are YATE and Asterisk, and after acquiring some results by establishing communication between them via the means of SIP trunk, furthermore the study was extended to the laboratory in order to test it over real servers that have TDM cards, in order to apply the study by the means of SS7 protocols, SIGTRAN, MGCP gateway and SIP-T. The experiments have almost delivered successful communications after conducting a configuration for the files on multiple sides.

Prohlášení:

Prohlašuji, že svou diplomovou práci na téma „Možnosti prenosu signalizace SS7 pres IP sít s využitím ústredny YATE“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 29.5.2013 .....

podpis autora

Poděkování

Děkuji vedoucímu práce Ing. Pavlovi Šilhavému, PhD., za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce a především za projevenou ochotu a trpělivost.

V Brně dne 29.5.2013 .....

podpis autora

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Purkynova 118, CZ-61200 Brno, Czechia

http://www.six.feec.vutbr.cz

Poděkování

Výzkum popsaný v této habilitační práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

V Brně dne 29.5.2013 .....

podpis autora

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007-13
OP Výzkum a vývoj pro inovace

Table of Contents

1. SIGNALING SYSTEM 7 (SS7)

1.1 Introduction

Signaling systems in telecommunication networks can be divided according to various aspects, one of these aspects is the communication that is based on the interfaces. In this case, it is important to distinguish between subscriber and network signaling.

Subscriber signaling occurs, for instance, on the local loop, between terminal equipment and the PBX. Network signaling is used between the exchanges. In addition, it can be divided into CAS (Channel Associated Signaling) and CCS (Common Channel Signaling).

SS7 signaling network belongs to the CCS group of communication. In the CCS signaling transmission network, the signaling and conversational signals are independent on each other.

There are 31 channel intervals to transmit the signalization that are available in one PCM system. These channels incomparably increase the amount of information that can be transferred. Each channel interval (64 kbit/s) is able to serve about 1,000 voice channels.

The SS7 signaling network can be viewed as a packet network carrying signaling messages. It allows signal transmission from circuit switching to a packet-switching. Transmission of signaling messages with the circuit switching is mainly intended for control establishing and terminating voice connections, while the packet-switching is used for communication, such as user databases in mobile networks.

1.2. Network structure

SS7 signaling network consists mostly of signaling points (SP) and Signaling Transfer Point (STP). SP represents either a source or a destination for signaling messages while STP can be seen as a kind of router for signaling messages.

In practice, this means that STP handles only lower-layer protocol concerning routing and does not deal with the application layers. There are several types of signalization modes for the transmission in relation to the voice path. For Instance: Associated mode means that the signal path is coinciding with colloquial way between tow sp's. The quasi-associated mode is used for non SP neighbors of the signaling link.

Each signaling point (SP) has its own address, called Signaling Point Code (SPC), and has a length of 14 bits, (see figure 1). It consists of the following: Zone that corresponds to the area of domain,

the network ID which specifies the ground within the zone, and the SD ID which identifies the No. of the SP.

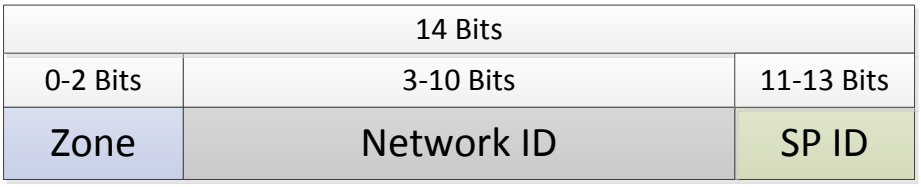


figure1 – SPC address (Signaling point code)

1.3. Protocol stack model

The signaling protocol is a set of rules that are used to transfer information from one point to another. Among its main features is the transmission of controlled information that is necessary to provide the required telecommunications services. Error detection and error correction are occurred during transmission or segmentation of larger messages to the signaling units.

SS7 protocols can be divided by their functions into four levels, (see figure 2). The figure shows the relationship between the Open System Interconnection (OSI) and the protocol model of SS7. This approach of allocating each layer is to enable easier changes in the implementation, which occurred in the modification process of one or more protocols. Seven layers of OSI model corresponds to four layers of SS7 model. Some features of the models are different.

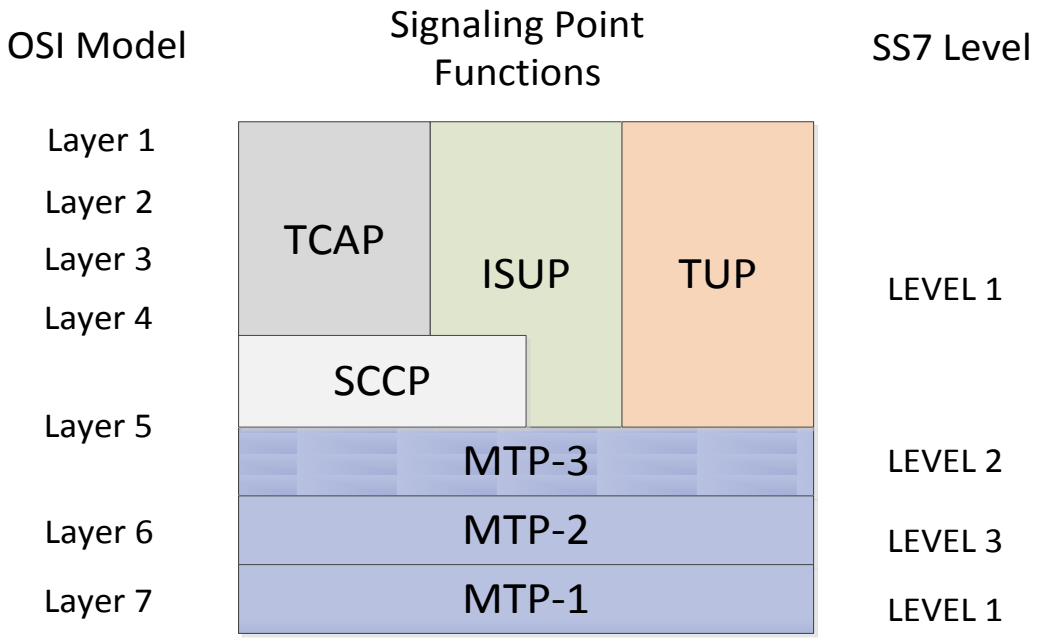


Figure 2 - SS7 protocol stack.

1.4 MTP layers

MTP is divided into three levels that correspond to the first three layers of the OSI model.

1.4.1. MTP1

The first level of MTP1 represents the physical layer, which is responsible of converting the bit stream into a form suitable for transmission over a medium, such as electrical or optical signal. The physical layer is characterized by mechanical and electrical properties of the connector and functional properties (baud rate, line code).

1.4.2 MTP2

The second level MTP2 corresponds to the link layer of the OSI model. The main task is to ensure reliable transmission between two neighboring signaling points SP / STP. For this purpose, the MTP layer is formed by signal unit SU (Signaling Units), which represent containers that transfer information to higher protocol layers. The basic structure of the signal unit is shown in figure 4.

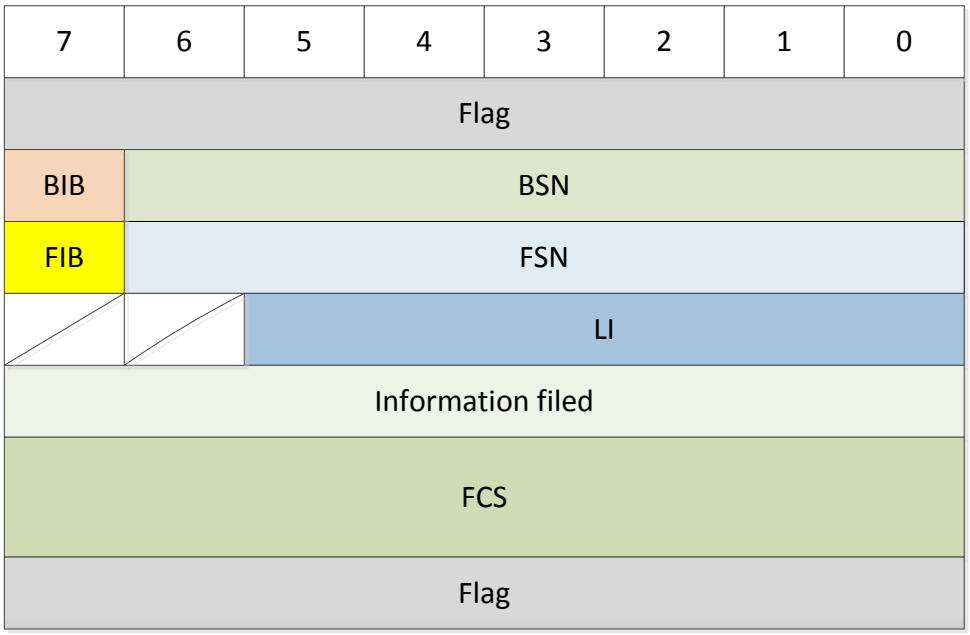


Figure 4 - Signaling unit SU

Each SU Start is marked by the occurrence of the same sequence of bits, called Flag. Followed by a counter of sent messages FSN (Forward Sequence Number), and successfully received messages BSN (Backward Sequence Number). The purpose is to indicate erroneously received messages or to notify repeated transfers serves bits BIB (Backward Indicator Bit) and FIB (Forward Indicator Bit). LI Length indicator carries information on the number of user information bytes contained in the information field. User information, in this context, is the higher protocol layer information. Transferring of information is secured using a cyclic code CRC-16. In case of error detection, MTP2 ensures retransmission erroneously received units. On MTP level, signal units are divided into three types, namely FISU (Fill-in SU), LSSU (Link Status SU) and MSU (Message SU). Fill unit FISU is sent when there is no another transmitted message, and it is used to monitor the signaling circuit and for the confirmation of correctly received messages LSSU and MSU, and does not carry any information. Status LSSU unit is used to monitor and control signal circuit between two neighboring SP/STP. MSU signaling unit transmits the information about the address, the destination, and the source of the SPC, about the type of protocol that is carried in the information field (eg, ISUP or SCCP), and data from a higher layer protocol.

1.4.3 MTP3

The main function of protocol-level MTP3 is routing, discrimination and distribution. After the arrival of signaling messages to the SP, MTP3 discriminate the function by the address, and then decides whether the message is intended for a certain SP. If the message is addressed to the local SP, it takes the distribution function and is delivered to a higher protocol layer. In case that the message is addressed to another SP, it is processed and sent according to the routing table, to the neighbor SP that is lying in the path to the target SP.

1.5. ISDN User Part Layer

ISUP protocol is used for establishment, management and termination of voice connections. It also provides additional services. Since signaling and colloquial network are mutually separated, signaling messages must have contained information on voice circuit. This identification is done by using the twelve-bit identifier CIC (Circuit Identification Code). All signaling information that belong to a given call conversation are transmitted at all times over the same signaling circuit. (Figure 5 illustrates the distribution of ISUP messages).

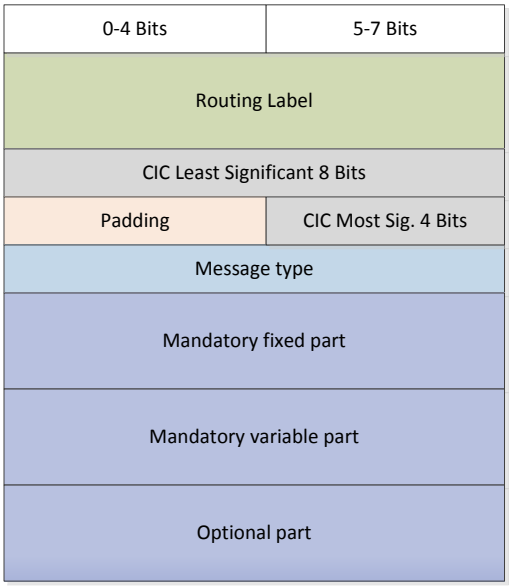


Figure 5 - ISUP message format

Following are selected signaling messages that are used to control the connection:

- IAM (Initial Address Message) - This message is used to initialize the call connection. In addition to the called party number information, it contains additional parameters such as a connection type, and whether it is used for echo cancellation, etc.
- ACM (Address Complete Message) - The message is sent in the opposite direction of the IAM, and to indicate that the call is processed, such as party is ringing.
- ANM (Answer Message) – It is a message that indicates whether the call is accepted.
- CPG (Call Progress Message) - This message serves informing the remote party (exchange) of an event relating to the call.
- REL (Release Message) – Reports informing the end of the call.
- RLC (Release complete of Message) - This message is sent to the other side as confirmation of the end of the call.

Bellow is an example that shows the process of the preparation and completion of a call connection between two parties. (See figure 6)

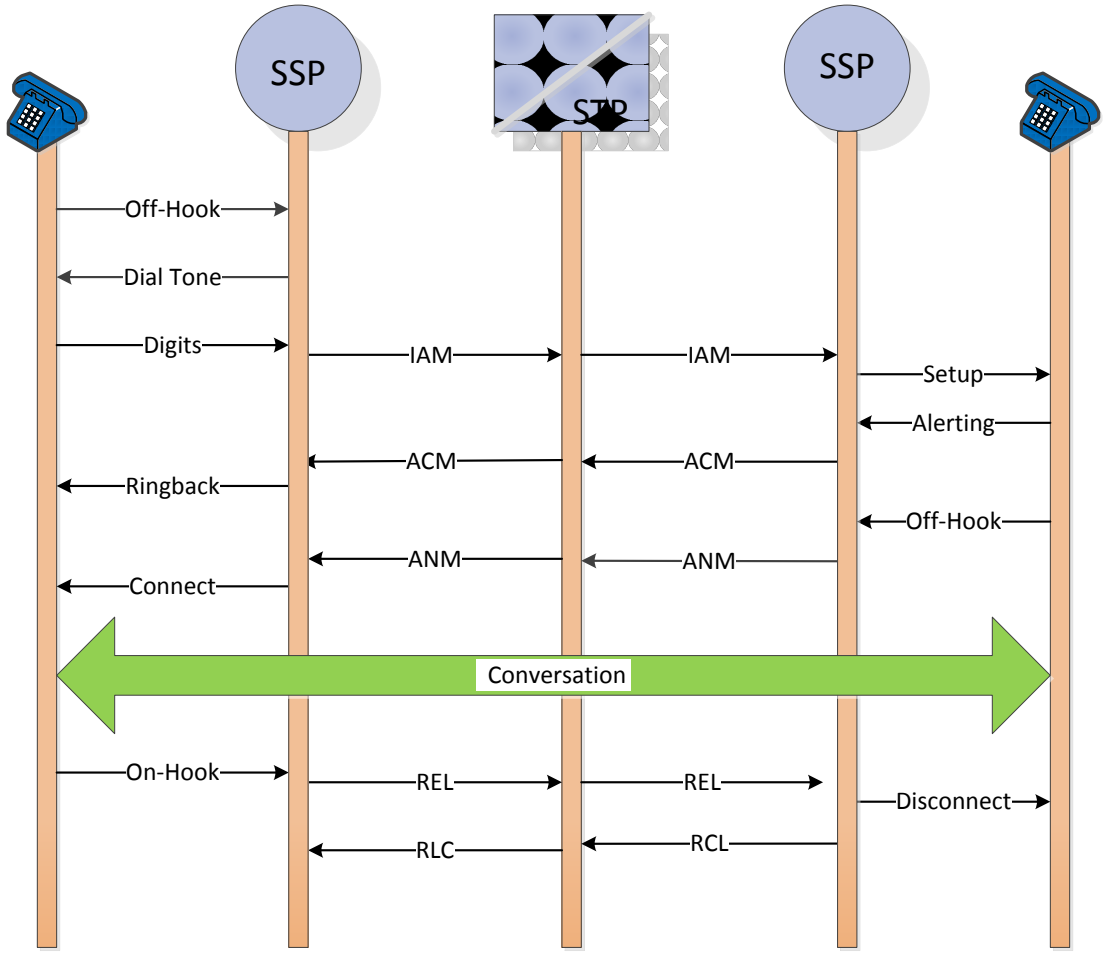


Figure 6 - Start and end of a call.

After dialing the called number, there will be initiation steps for the entire transmission. The default exchange sends IAM message containing basic information regarding the required connections, such as dialed number, connection type, subscriber type (ISDN, non-ISDN) and other optional parameters, depending on the configuration of the operator.

When the exchange gets the necessary information for further routing, the previous exchange will be indicated by ACM message. We can say that this message contains similar information as the IAM message. In addition, it can contain, for example, the information about the called party. The maximum time between messages IAM and ACM is monitored by the T7 timer (20s - 30s), and in the case of time expiration, it will terminate the connection (sending a message REL). In case that the call is routed to the mobile network, it may happen that the end time will be before that the party have the ringtones, this is due to long time ringing that might be more than the timer T7. The reason is to avoid aborting the connection. Exchange sends, just before the timer expires T7 ACM, message with "No Indication". Information about the ringtone of the called party is then passed to

the signaling message CPG. If the participant answers, the exchange will show it by ANM messages. The exchange, which terminates the connection, will send the message REL. An important parameter in this case is the reason for the termination. Remote exchange confirms RLC message.

2. Transferring signaling system SS7 over IP networks

IP telephony network uses the best available bandwidth compared to the public switched telephone network. In switched circuit telephone network, 64 kilobits per second (kbps) end-to-end circuit is reserved for each call. While in VoIP network, the voice will not be carried in one reserved single channel, it will rather be converted to packets. Those packets will be travelling across the IP networks in deferent paths, to reach its destination. And from these, we can note that the VoIP network can carry many times the number of voice calls as a switched circuit network with better voice quality.

Another major reason for applying this service is the broad use of IP networkers compared to the switched circuit telephone network, by the flexibilities, bandwidths, price, etc.

2.1 SIGTRAN protocol suite

SIGTRAN (Signaling Transport) is a new set of protocols that are created by the IETF (International Engineering Task Force). The aim is to define the mode of transmission SS7 signaling over IP-based networks. The architecture identifies two components: a transport protocol for the SS7 protocol layer, and a module to emulate lower layers of the protocol (see figure 8). For example, if the native protocol is MTP3, the SIGTRAN protocols will provide the equivalent functionality of MTP2. If the native protocol is ISUP or SCCP, the SIGTRAN protocols will provide the same functionality as MTP2 and MTP3. If the native protocol is TCAP, the SIGTRAN protocols will provide the functionality of SCCP (connectionless classes) and MTP2 and MTP3.

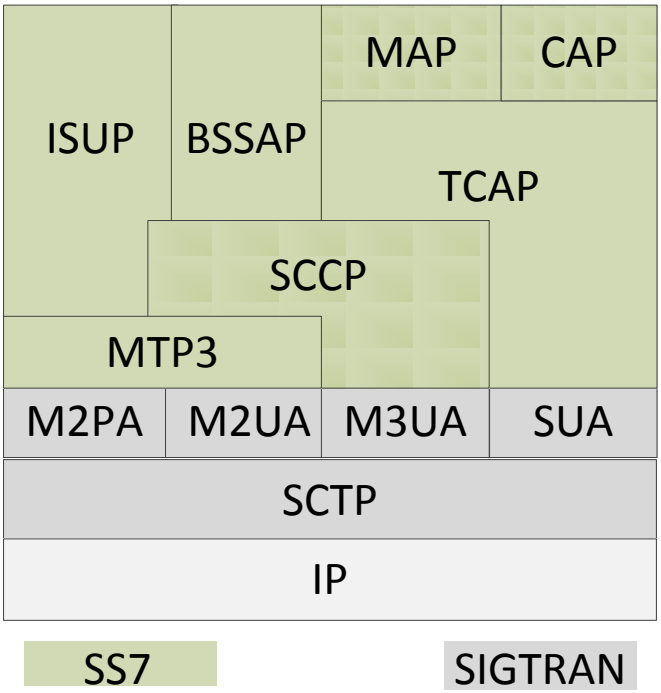


Figure 8 - SIGTRAN protocol model

Following are four protocols that are defined for the adaptation:

- MTP2 UA (M2UA) - The protocol uses the services of MTP3. This protocol is mainly used for client-server communication, such as SG - SP.
- MTP2 Peer-to-Peer (M2PA) - used mainly for communication SG-SG, and it is applicable, for example, to connect two SS7 over IP networks.
- MTP3 UA (M3UA) - Supports Peer-to-peer and client-server. The user is ISUP or SCCP protocol.
- SCCP UA (SUA) - The service uses, for example, TCAP protocol.

2.2 SCTP transport protocol

It uses the transport layer and deals with the transmission of telephone signaling over IP. Unlike the existing transport protocols TCP and UDP, SCTP uses several mutually independent channels operating simultaneously. It provides some of the same service features of both: it is message-oriented like UDP, and ensures reliable, in-sequence transport of messages with congestion control like TCP. (See figure 9).

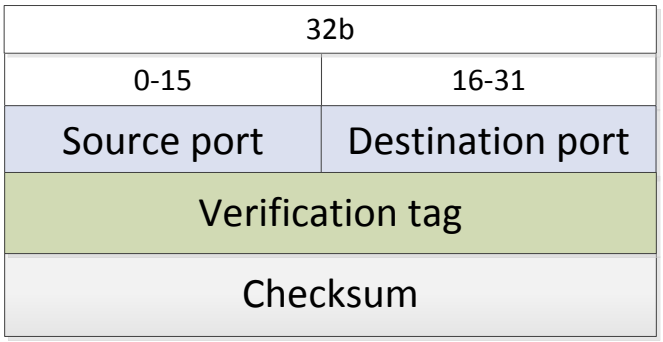


Figure 9 - SCTP Header Format

SCTP protocol header contains source and destination port field for 16 bits. Verification Tag Field with size of 32b, serving the recipient to verify the packet from the sender. The last filed is the checksum of the packet which uses the Adler-32 algorithm.

SCTP supports multi-homing when communicating node has several IP addresses. These addresses can be changed during the call setup, and it can be used for both IPv4 and IPv6 addresses. During the communication, one of these addresses is selected as the primary for which the data are sent.

2.3 M2PA adaptation protocol

The protocol supports the transmission of SS7 MTP3 signaling over IP networks. It uses SCTP transport protocol. M2PA supports asynchronous notification of state change operation.

Throw the MTP protocol, each node uses the MTP3 layer SS7 point code, and it is also applied to each signaling point in the IP network. M2PA also serves as an interface between MTP3 and MTP2.

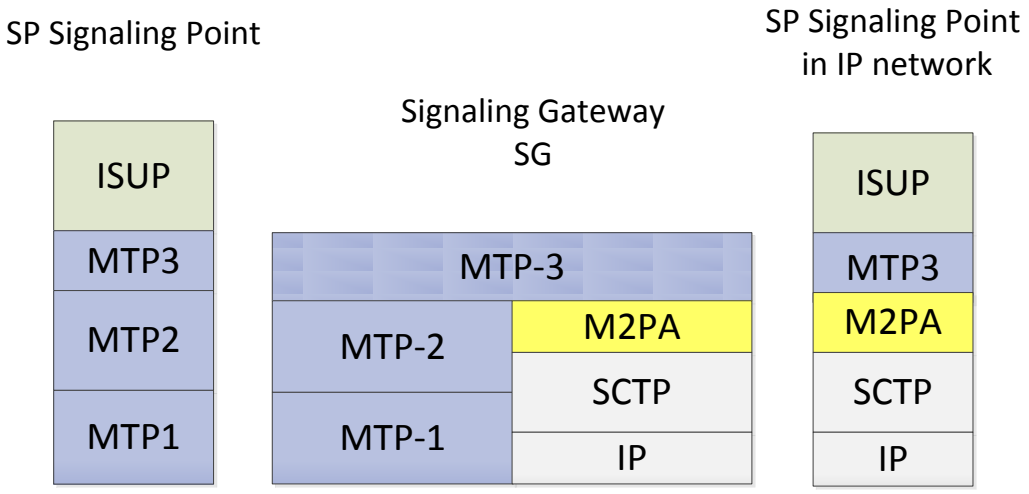


Figure 10 - Communication using M2PA protocol

Figure (10) illustrates the signaling point in SS7 network connected via a signaling gateway SG to signaling point in the IP network. The signaling gateway is essentially behaves like STP point.

2.4 M2UA: User Adaptation Layer

M2UA is a defined protocol for transmitting a SS7 MTP2 user signaling messages over IP using SCTP. This protocol is used for communication between a Signaling Gateway (SG) and Media Gateway Controller (MGC). (See figure 11). It is assumed that the SG receives SS7 signaling over a standard SS7 interface, using the SS7 Message Transfer Part (MTP) to provide transport. The SG acts as a Signaling Link Terminal.

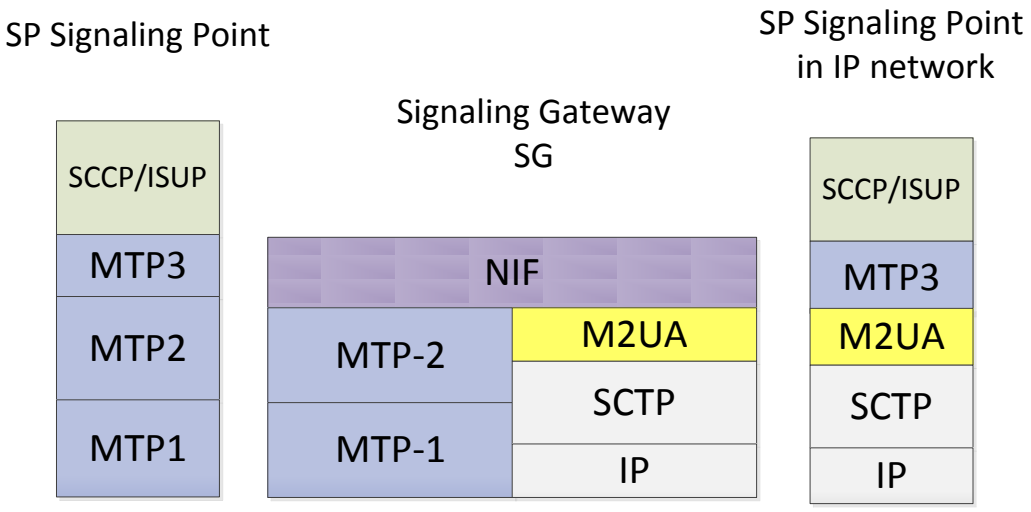


Figure 11 - Communication Protocol using M2UA

Signaling Gateway on one side expects SS7 signaling using MTP protocol. The gateway then sends via SIGTRAN protocol a received MTP3 message signaling point in the IP.

2.5 M3UA: MTP Level 3 User Adaptation Layer

M3UA supports the transport of any SS7 MTP3-User signaling (such as ISUP and SCCP messages) over IP, using the services of the Stream Control Transmission Protocol (SCTP). The protocol is used for communication between a Signaling Gateway (SG) and a Media Gateway Controller (MGC), or IP-resident database (as shown in figure 12). It is assumed that the SG receives SS7 signaling over a standard SS7 interface using the SS7 Message Transfer Part (MTP) to provide transporting information. [14]

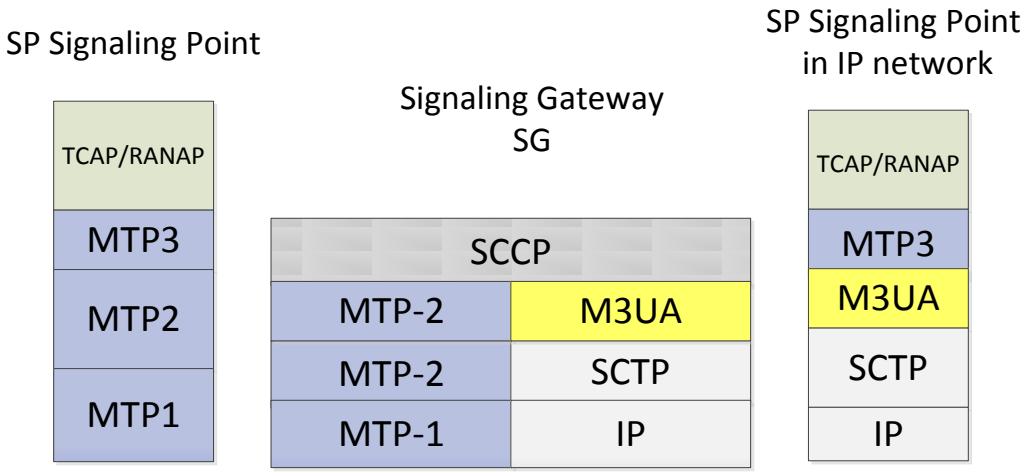


Figure 12 - Communication using M3UA protocol

2.6 SUA Protocol Adaptation

The Signaling Connection Control Part User Adaptation Layer (SUA) protocol details the delivery of SCCP-user messages (MAP & CAP over TCAP, RANAP, etc.) and new third generation network protocol messages over IP, between two signaling endpoints. Consideration is given to the transport from an SS7 Signaling Gateway (SG) to an IP signaling node (such as an IP-resident Database). This protocol can also support transport of SCCP-user messages between two endpoints wholly contained within an IP network. [14]

It is modular in design so that it can work on the architecture of signaling gateway - IP signaling point, and peer-to-peer IP signaling point. SUA supports the SCCP in the form of connected and connectionless services. Model of communication can be seen in figure 13.

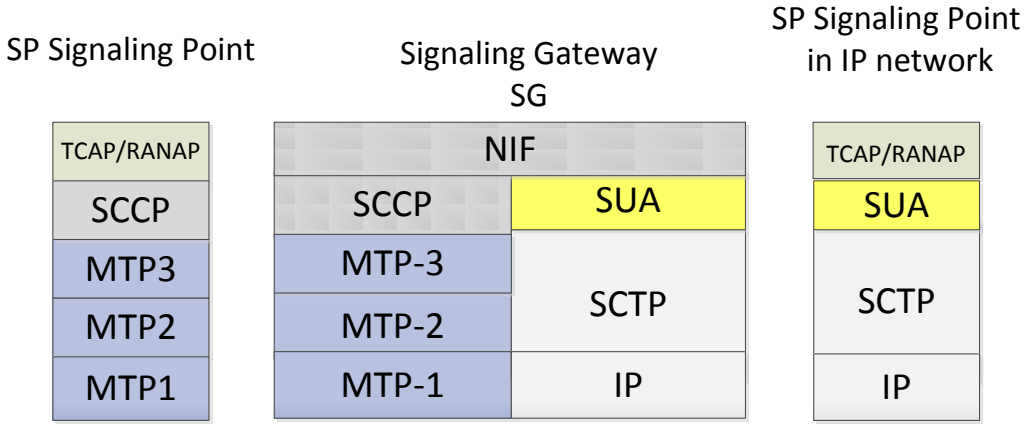


Figure 13 - SUA - structure for connectionless transmission

2.7 SIP-T interfacing

SIP protocol is an application layer that can setup, modify and end sessions, such as Internet telephony, multimedia conferences and similar applications. SIP is one of the most protocols used to perform Voice over IP. Performing telephony call signaling and transporting the associated audio media over IP gives significant advantages over traditional telephony, a VoIP network can give a big extension and vice versa for the traditional telephone networks.

SIP-T is a set of mechanisms for interfacing SIP with traditional telephone signaling. The main function of SIP-T is to provide protocol adaptation and feature transparency across interconnected points in PSTN-SIP.

There are three basic models use SIP-T for interacting calls with gateways. Calls that originate from the IP network can pass through a gateway to terminate at a PSTN endpoint, such as an IP phone. Conversely, a call generated from an analog phone that traverses a gateway to terminate in the IP network. Finally, a call might originate and terminate in the PSTN, but through a SIP-based network.

2.8 Transport Protocol BICC

It is another transport protocol BICC (Bearer Independent Call Control) that serves as a support narrowband ISDN service, in broadband backbone networks. It is compatible with today's designs networks and systems for the transmission of voice information. BICC protocol is able to work with wide range technologies, such as Asynchronous Transfer Mode ATM, IP and TDM.

BICC signaling is independent on the signal carrier media. The call control and the connection control are separated. Figure14 shows how the network works.

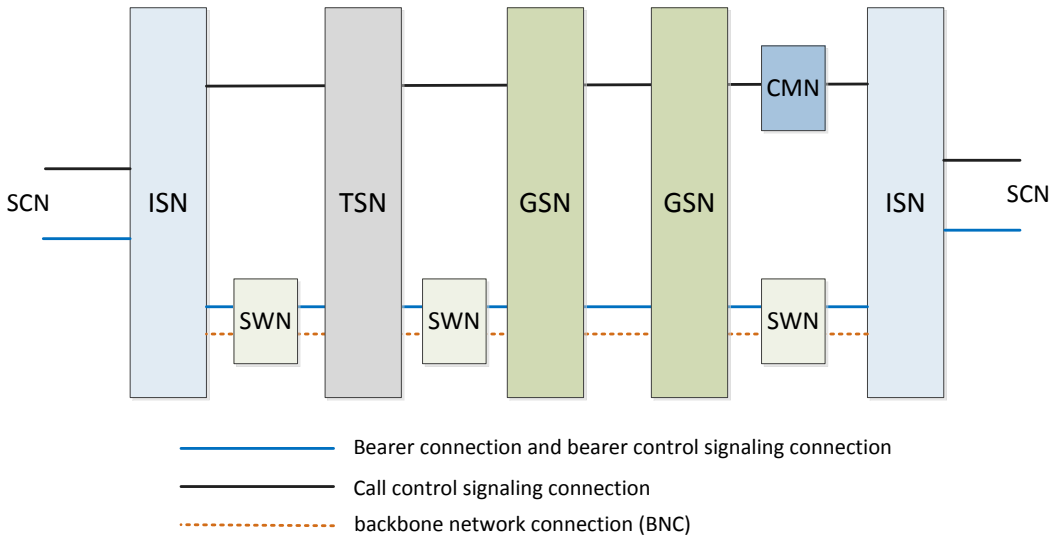


Figure 14 - BICC network model

Signaling and media transmission are managed with (serving nodes SN), which control signaling and bearer channels. There can be three types of service nodes:

1. Interface serving node (ISN)
2. Transit serving node (TSN)
3. gateway serving node (GSN)

The first-mentioned utility node provides an interface directly with the switched circuit network SCN. Transit nodes are the intermediate nodes and gateways connect separate networks BICC. TSN and GSN network topology give a greater flexibility and making it possible to increase the flow of data. Another thing that BICC network supports is the "call mediation node" CMN which only manages the signaling channels. Component "switching node" SWN is applied only where the carrier transmits over the ATM. In ATM networks, switches act as the transmission medium and supporting process control signaling messages. In BICC networks the function of ATM switches is achieved through SWN. The IP networks SWN is not needed because routers do not play here any signaling role because the carrier link is built and is known only to the initial and end point. Carrier connection between two adjacent nodes is called (backbone network connection BNC) and can pass through SWN.

BICC call control signaling is up to two exceptions based on the ISUP protocol. Call control and media control are included in the ISUP in one protocol. The BICC is handled by separate protocols and can even be carried on separate networks, as shown in figure14. This allows the BICC deal with any kind of media. The second difference is that while the ISUP signaling is transmitted via SS7 signaling network, BICC call control signaling is independent on network type - can be transmitted by an IP, ATM or even SS7 signaling networks with circuit switching. On the other carriers, signaling control BICC always uses the same network through which the carrier is connected.

2.8.1. BICC protocol suite

The development of BICC signaling is based on three rules:

- Operational separation between call control and media control
- The independence of the transmission technology
- Signaling does not affect existing interfaces on the signaling networks with circuit switching (SCN)

The BICC suite of protocols is specified in ITU-T recommendations Q.19xx. The protocols generally belong to two groups:

1. Signaling protocols for call control (BICC call control signaling protocols-) - Used for peer-to-peer communication between service nodes. It is closely associated with the ISUP protocol, however is not compatible with it.
2. Signaling protocols for control carriers (BICC Bearer-control signaling protocols) - Is used for exchanging information between the initial and end-user interface. Port numbers, IP addresses, and type of media are needed to establish a connection.

2.8.2Setting up a call

Figure 15 shows the procedure for establishing a connection between two adjacent nodes (A, B). Carrying channels are built from the previous operator node. All components communicate via messaging protocol until the communication between the MCF and MMSF runs internally. These modules are interconnected.

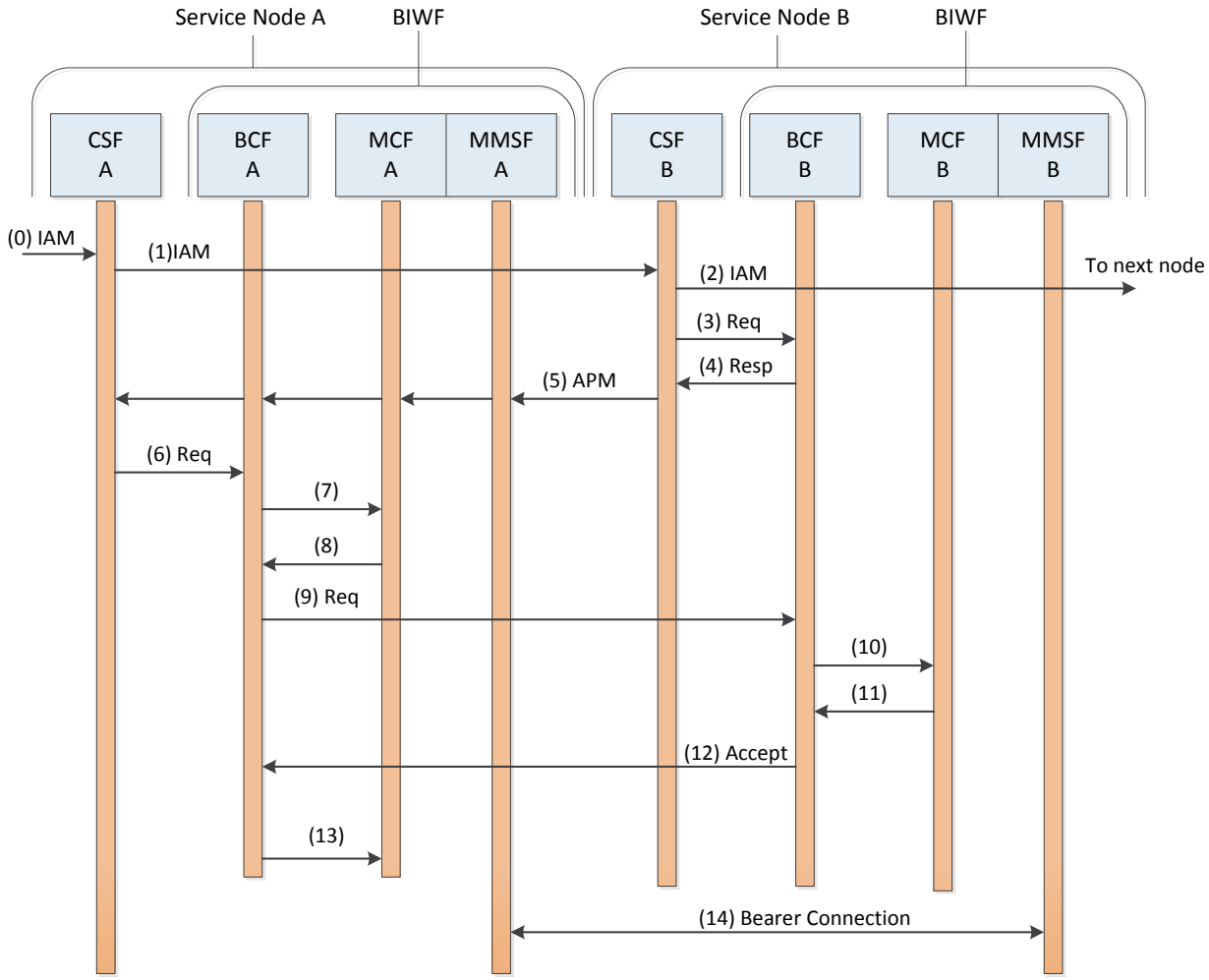


Figure 15 - BICC – Setting up a call

(A) Node in this case is considered to be the initial node, (B) is an end. The process begins (0) when the CSF (A) receives a IAM message. CSF (A) processes the message, selects media and determines the routing of calls. Once the path is selected CSF (A) sends report (1) IAM to CSF (B) with the media characteristics. The CSF (B) receives that message, and then determine, if it's needed to be routed, it sends (2) an IAM message to the next node. If CSF (B) is the selected node, the medium sends the request (3), and subsequently receives (4) from BCF (B) with the BCN-identifier ID and media characteristics. These information is sent (5) to CSF (A) using the APM message. CSF (A) forwards (6) information to the BCF (A).

BCF (A) then requests (7) and receives (8) from MMSF (A) for the bearer interface address. BCF (A) sends (9) the address BCF (B), BCF (B) forwards (10) message to MMSF (B). MMSF sends (11) its bearer address to BCF (B), it is then sent away (12) into BCF (A). BCF (A) forwards (13) the received address of node B carriers of MMSF (A). At this point the MMSF (A) and MMSF (B)

exchange addresses and they are now able to provide a voice communications. More detailed description of the parts BICC protocol can be found in the literature [3].

3. YATE

There are many kinds of PBX's (Private Branch Exchange) software. They might deliver the same services, except that some of them do not support some technology services in full scale. For example, YATE supports SS7 signaling in the same way that PBX's software does. We all know that ss7 has four levels, in the 4th layer we see that YATE supports two protocols; the ISUP used in SP and the SCCP used in the STP. Those protocols can work together and perform a batter network topology. It means that YATE is not only a server, but rather it can also act as a router.

YATE is an open source PBX for a new generation. It focuses on VoIP and PSTN (Public Switched Telephone Network). Some of its advantages include easy scalability. It supports voice, video, data services and IM (instant messaging) as well. It is written in C + + and supports scripting in various programming languages (PHP, Python and Perl libraries). YATE is distributed under the GPL. (4)

YATE can be used as:

- VoIP server
- VoIP client
- Conference server - with up to 200 channels in a single conference
- VoIP to PSTN gateway
- PC2Phone and Phone2PC gateway
- IP Telephony server and/or client
    - H.323 gatekeeper
    - H.323 multiple endpoint server
    - H.323<->SIP Proxy
    - SIP session border controller
    - SIP router
    - SIP registration server
    - IAX server and/or client
    - Jingle client or server
    - MGCP server (Call Agent)
- ISDN passive and active recorder
- ISDN, RBS, analog passive recorder
- SS7 switch
- Call center server
- IVR engine
- Prepaid and/or postpaid cards system

3.1. Installation package YATE

Installing PBX YATE and testing take place on the Linux distribution Ubuntu. Adding the YATE package can be either from the graphical interface through System \ Administration \ Synaptic Package Manager, or by using commands on the terminal. Before installation and using the software from a Personal Package Archive, first, we need to tell Ubuntu where to find the PPA, by typing the line:

```
add-apt-repository  ppa:vpol/yate
apt-get update
```

The system will now fetch the PPA's key. This enables Ubuntu system to verify that the packages in the PPA have not been interfered with since they were built.

Now we should tell the system to pull down the latest list of software from each archive it knows about, including the PPA, therefore, we just need to add:

```
sudo apt-get update
```

And now we are able to start installation of the software from the PPA. Another way to download and install the current version of the package library can be explained in the following steps:

```
wget http://yate.null.ro/tarballs/yate3/yate-3.3.2-
1.tar.gz
```

Unpacking the archive,

```
tar –zxvf  yate-3.3.2-1.tar.gz
```

Compiling and installing,

```
make && make install
```

There are two possible ways to run the PBX. First one is to run it from the actual folder by typing the following:

```
./run –vvvv –Dof
```

Second one is by typing yate from any directory within the terminal mode. And this must be followed by copying the new shared libraries from the installed folder to the usr/lib by typing:

```
cp libyate.so libyate.so.3.3.2 libyatejabber.so
libyatejabber.so.3.3.2 libyatemgcp.so libyatemgcp.so.3.3.2
libyatesig.so libyatesig.so.3.3.2  usr/lib
```

Then we have to create a Symbolic link from our configuration folder to the main etc folder:

```
ln –s /usr/local/etc/yate /etc/
```

And now we can run YATE without referring to the source folder by typing:

```
yate –vvvv –Dof
```

If the installation is successful, then we should have the exact results as shown in the following figure with no red lines:

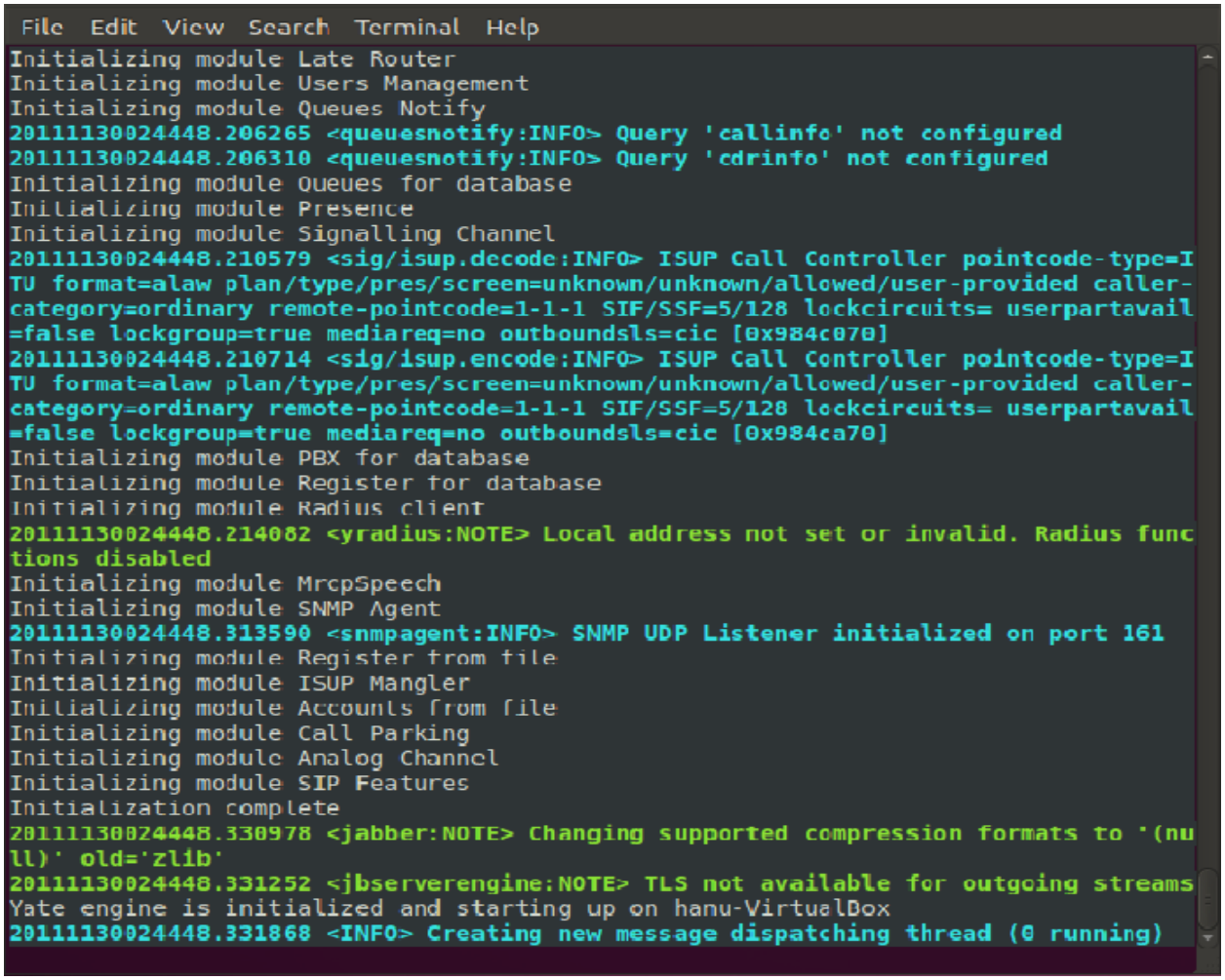


Figure16 the processes that appear by the server start up.

Suspending or aborting YATE is done by pressing Ctrl + C, while rebooting is by using the Ctrl + \.

3.2. YATE Architecture

The most important aspect of YATE is its message-passing system. It allows us to have a bigger flexibility than with plain functions. Mainly, because messages in YATE can have an arbitrary number of parameters, and can be sent to more than one module, by changing the priority. [4]

YATE's four main components are:

- Core - generic classes like String, Thread, Socket, Mutex
- Message Engine - message related classes like Message, Engine, Plugin
- Telephony Engine - telephony related classes like Driver, Channel
- Yate Modules - modules of YATE are equal, no matter whether they are telephony, routing or anything else, because of the message passing system.

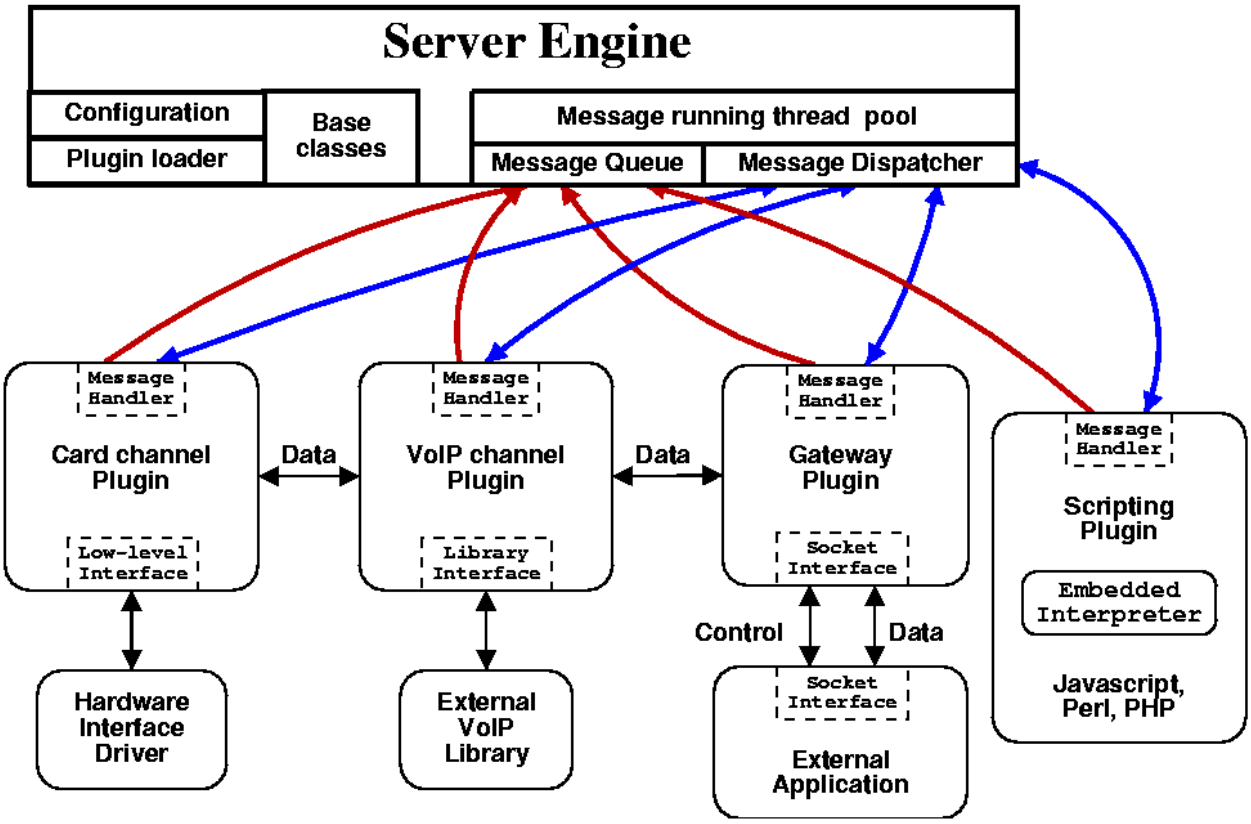


Figure 17.YATE architecture (Source: http://yate.ro/pmwiki/index.php?n=Main.Architecture (10.12.2011))

After starting the PBX, modules are loaded, including the routing plan. Modules can be all loaded or it can be defined in order to determine which concrete module needs to be loaded. The configuration can be set in the file yate.conf. In the [general] section, in order to allow all the modules to be loaded, we need to type enable after modload parameter. If we would like to load certain modules, we should define the required modules in the [modules] section, by adding the shape module YATE to enable. For example, ysipchan.conf = enable. Other possible yate.conf file parameters are as the following:

In the section [General]

- modpath - setting the path to the modules, if they have a location that is different from the standard one.
- extrapath - path to use other external modules.
- restarts - the time in seconds, after which the PBX restarts

The [localsym] is used for "violent" load modules without defining the global variables. Again in the shape yate = enable, section [nounload] for the modules that do not have to be unloaded from memory. The [preload] and [postload] are used to load the libraries.

In the [debug] section, the debug level can be set in the form name = level number,

eg sip = level 10, and in section [telephony] timeout for the default expiration time for channels, maxroute to limit the number of call, maxchans to limit the number of channels, and dtmfdups to enable or disable duplicate DTMF.

Principle in terms of routing and used protocols is shown in the following figure

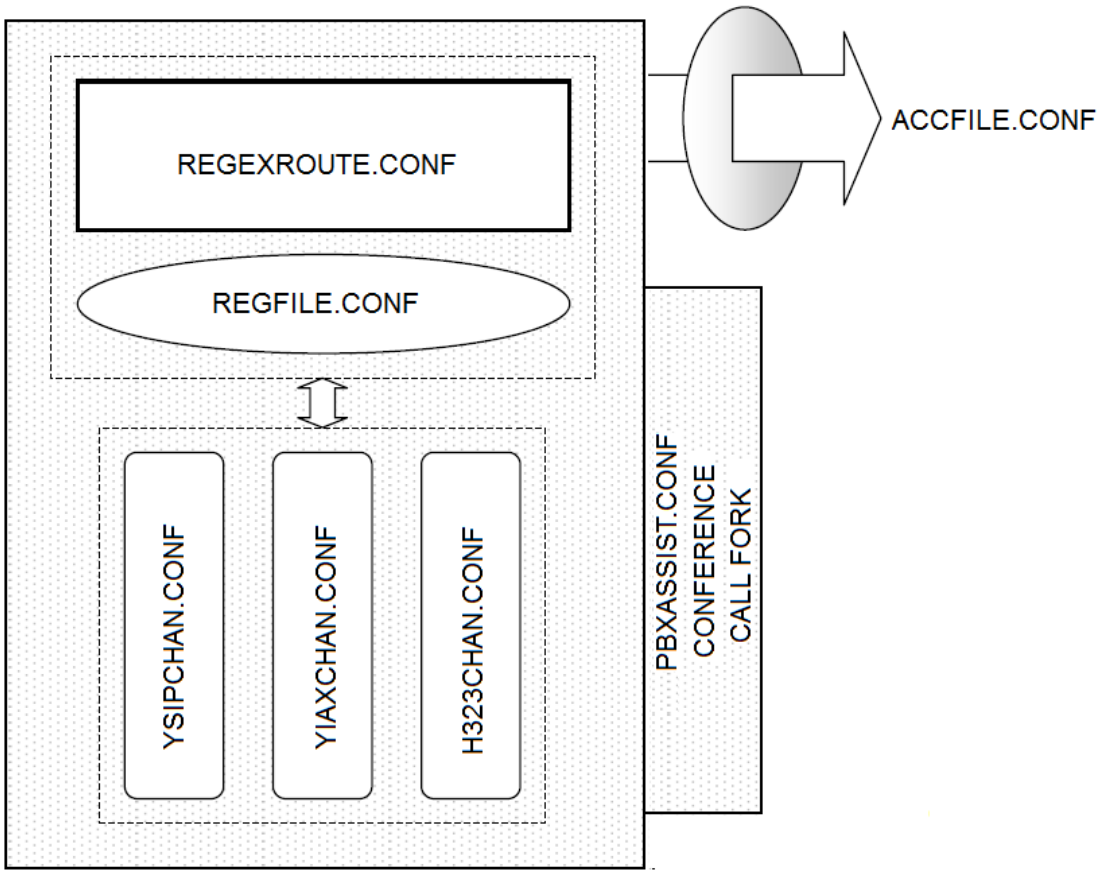


Figure 17 - Block diagram of the interconnection of individual modules.

3.3. Configuration files

There are many configuration files that support deferent services. The services are presented in instructions files, written in C++ language. Each configuration file is mapped to c code, which contain classes and functions that are used to process a parameter inside a configuration file. Those files can work together to adapt several technologies. Such as SS7, ISDN, etc.

3.3.1 Configuration SS7

Ysigchan.conf

This module is based on the signaling library and uses modules such as zapcard, wpcard, and sigtran, to create interfaces and circuits that are used to transfer signaling packets and audio data.

One of this module purposes is to serve the SS7, to handle the SS7 protocol (management, maintenance, routing), and to make and receive calls using the ISUP protocol.

To fulfill its purposes, the module creates the following objects:

- A signaling engine, used to keep track of all signaling components (circuit groups, call controllers, interfaces, etc).
- A SS7 router. Identify a network (route) to send outgoing SS7 packets and transmit them. Find a destination service for incoming SS7 packets.
- A SS7 management and a SS7 maintenance object used for the management and maintenance of the SS7 network.
- Any call controller (trunk) requested in the configuration file.

To set the proper functioning of SS7 signaling is necessary to define the four links which represents the SS7 layers in the ysigchan.conf file and they are: ISUP trunk, SS7 linksets, SS7 links. Each one of those four links will be considered as a protocol stack.

ISUP trunk contains the following settings:

[isup]	The name of ISUP trunk
Type	type of trunk, ie SS7-ISUP
pointcodetype	the type of SS7 point code, the possible parameters ITU ITU-T Q.704 ANSI ANSI T1.111.4 ANSI8 8-bit SLS China GF 001-9001 Japan T-Q704, NTT-Q704 Japan 5 5-bit SLS
Pointcode	local point code in the format network-cluster-member
Defaultpointcode	Point code for outgoing calls
Remotepointcode	pointcode for outgoing messages
Service	the service information field (protocol number, default 5)
Priority	priority in the service information octet, value regular, special, cicuit or facility
netindicator	network indicator in the SIO, values of international, spareinternational, national, reservednational.
lockgroup	allow sending requests to block / unblock
earlyacm:	boolean: Convert received early ACM user state into progress or ringing
voice:	string: Specify the span(s) containing the voice channels (L1)Multiple cardsmay be specified by simply separating them with a comma (',') character
offset:	int: Offset of voice circuit codes relative to low level circuit number
range:	string: Arbitrary ranges of circuits for outbound calls (can be repeated). Possible formats: range=name:list of circuits (like 1,3,5-9) range=name:strategy:list of circuits (like 1,3,5-9)

strategy:	string: The strategy used to allocate voice channels for outgoing calls Allowed values: increment Incremental search an idle channel starting with the last allocated channel decrement Decremental search an idle channel starting with the last allocated channel lowest Incremental search an idle channel starting with the first channel in group. Highest Decremental search an idle channel starting with the last channel in group. random Randomly choose an idle channel.
Strategy-restrict:	string: Defines channel allocation restrictions and behavior. This option is ignored when strategy is random. The allowed values: odd Allocate only odd channels even-fallback Allocate even channels, fall back to odd channels odd-fallback Allocate odd channels, fall back to even channels even Allocate only even channels
channelsync:	The interval (in seconds) at which the call controller will try to re-sync idle channels.
sls:	integer or keyword: Default Signaling Link Selection in outbound calls. Allowed values: auto Let SS7 Layer3 add a proper value last Last SLS uses (default for non-ITU) cic Uses the circuit number as SLS (ITU style) 0..255 Explicits numeric value, gets truncated to the SLS bit range
maxcalleddigits:	integer: Maximum number of digits in an IAM Called Party Number. If the called number is longer, the extra digits will be sent in a SAM message.
numplan:	string: Default numbering plan for outgoing calls Values: unknown, isdn, data, telex, national, private.
numtype:	string: Default number type for outgoing calls.
Values:	unknown, international, national, net-specific, subscriber, abbreviated ,reserved. Defaults to unknown if missing or incorrect.
presentation:	string: Default number presentation for outgoing calls.
Values:	allowed, restricted, unavailable. Defaults to allowed if missing or incorrect.
screening:	string: Default number screening for outgoing calls.

Values:	user-provided, user-provided-passed, user-provided-failed, network-provided. Defaults to user-provided if missing or incorrect.
inn:	boolean: Routing to Internal Network Number allowed Defaults to no.
format:	string: Default data format for outgoing calls. Values: alaw, mulaw, g721.
continuity:	string: Type of continuity check to perform on the circuits. If not set Calls that request continuity check will be rejected.
ringback:	boolean: Offers a ringback tone even if not provided by peer channel. If it fails the correct indication, no inband available is signaled.
location:	string: Exchange location to be set when sending Q.850 causes. Available values are: U User LPN Private network serving the local user LN Public network serving the local user TN Transit network RLN Public network serving the remote user RPN Private network serving the remote user INTL International network BI Network beyond the interworking point A default 'BI' will be used if not set or invalid.
confirm_ccr:	boolean: Sends a Loopback Ack (national use) in response to Continuity Check Request message. Defaults to yes.
drop_unknown:	boolean: Drops call or attempts to change circuit if an unknown or unsupported message is received in an early state.
needmedia:	keyword: When media absolutely is required to continue the call. In case of circuit failure the call may continue and retry circuit setup later. Allowed values: no Media not required, always continues the call(default) ringing Media required after call is ringing answered Media required after call is answered yes Media required from the beginning
ignore-grs-single:	boolean: Ignores (drops) circuit group, resets messages with range 0 (1 circuit affected). Defaults to no.
duplicate-cgb:	boolean: Duplicates all sent circuit group blocking requests
print-messages:	boolean: Prints decoded protocol data units to output

extended-debug:	boolean: Prints extended debug data (such as raw hex data) to output.

linkset is used to represent the SS7 MTP3. And it contains the following settings:

[linkset]	The name of linkset.
type:	keyword: Specify the linkset type, of Layer 3 (MTP3).
netind2pctype:	point code types used to map the incoming network indicators, one value, or 4 values separated by a comma, for example ITU or ANSI,ITU,ANSI,ITU.
netind2pctype:	string: Comma separated list of point code types used to map an incoming network indicator to a specific point code type. This option is required and is used to configure a SS7 MTP3 network. The listmust contain either a single type or 4 values, indicating the point code type for International, Spare International, National and Reserved National network in dicators. If only a single type is specified, it will be used to configure all types of network indicators. See the ISUP pointcodetype option for allowed values.
netindicator:	keyword: Default value of Network Indicator bits Allowed values: international, spareinternational, national, reservednational.
route:	string: Builds a destination route for the SS7 network. The format of this option is pointcodetype,label,priority. This parameter can be repeated to build multiple destination routes. The network will notify the router about its destination(s) and priority. If not specified, the priority is 100. A zero priority creates an adjacent route.
adjacent:	string: Builds an adjacent route for the SS7 network (A, E and F links).The format of this option is pointcodetype,label. This parameter can be repeated to declare multiple adjacent routers. The network will notify the router about its destination(s) and priority. The priority is always zero so an adjacent route will always match first.
local:	string: Declares a local pointcode for the SS7 network. The format of this option is pointcodetype,label. This parameter can be repeated to declare a local point codes per type. To be standards compliant, at least one local pointcode must be declared.
allowed:	string: List of point codes explicitely allowed from this SS7 network. An empty or missing list will allow access to all known routes
router:	string: Name of the SS7 Router to attach to A boolean false value disables attaching a router (unlikely).If no router is attached only a single User Part can be connected router=ss7router.
link:	string: Name of a SS7 Layer 2 link to create in the linkset. This parameter can be repeated to add more links to the linkset An explicit SLS can be provided after a comma.

autostart:	bool: Automatically enables the linkset at startup.
checklinks:	bool: Checks the links answer to maintenance messages (SLTM/SLTA) before placing them into service (normal behavior).
forcealign:	bool: Realign links that are no longer responding to SLTM.
checkfails:	integer: Interval in msec for resending SLTM messages (Q.707 T1). A value of zero disables link fail else the value is clamped between 4s - 12s.
maintenance:	integer: Interval in msec for sending SLTM messages (Q.707 T2) A value of zero disables periodic SLTM else it is clamped to range 30s - 5m.
layer3dump:	string: Filename to dump MTP3 packets.

This part is linked to the second level of SS7 (MTP2), and It has parameters that can be described below:

[link1]	the name of SS7 links
type:	keyword: Specify the link type. Allowed values: ss7-mtp2: SS7 Message Transfer Part - Layer 2. ss7-m2pa: SIGTRAN
autostart:	bool: Automatically try to align the MTP2 at startup or on failure. This should be enabled (default) for normal operation.
emergency:	boolean: Emergency align SS7 MTP2 layer at startup.
autoemergency:	boolean: Emergency align link if owner linkset is down.
filllink:	boolean: Configureure MTP2 to request link fill (packet repeat) when sending FISU or LSSU packets.
rxunderrun:	int: Maximum interval in ms between two packets before we report an underrun condition, zero to disable or 25+ rxunderrun=0
layer2dump:	string: Filename to dump MTP2 packets to

sigtransport.conf

sigtransport.conf is a support module provides a standard factor for the common lower layer of SIGTRAN protocols. Here is where the IP addresses, ports and protocols are configured. Each connection defined by this module is referenced by ysigchan or by some other modules.

Each section in this file describes a SIGTRAN connection. Connections are referenced from other configurations describing the upper layer.

`[name-of-connection]`	The name of the section identifies the connection.
`type:`	`keyword:` Socket type - sctp, tcp, udp, unix.
`stream:`	`bool:` Socket connection type. Designed for SCTP sockets to create a stream socket or a sequenced packet socket.
`local:string:`	Primary local address. Format is IPv4:port .
`localN:`	`string:` Additional local addresses, SCTP only Multiple addresses can be specified by incrementing the 1-based index at the end of 'local'.
`remote:`	`string:` Primary remote addressFormat is ipv4:port .
`remoteN:`	string: Additional remote addresses, SCTP only Multiple addresses can be specified by incrementing the 1-based index at the end of 'remote'.
`endpoint:`	`bool:` Set to true if this is an endpoint that actively tries to connect, false to listen for remote connections.

ysipchan.conf

The flowing setting is used to identify, and to set parameters for a sip user. Some of the parameters can be set as default or can be ignored.

`[general]`	This section sets global variables of the implementation.
`port:`	`int:` SIP UDP port, on which protocol is listening.
`addr:`	`ipaddress:` IP address to which phones can then be registered.
`maxpkt:`	`int:` Maximum received packet size, 524 to 65528, default 1500.
`buffer:`	`int:` Requested size of socket's receive buffer, 0 to use default.

thread:	keyword: Default priority of the SIP handling threads. Can be one of: lowest, low, normal, high, highest. High priorities need superuser privileges on POSIX operating systems. Low priorities are not recommended except for debugging.
floodevents:	int: How many SIP events retrieved in a row trigger flood warning.
maxforwards:	int: Default Max-Forwards header, used to avoid looping calls.
useragent:	string: String to set in User-Agent or Server headers.
realm:	string: Authentication realm to offer in authentication requests.
transfer:	bool: Allow handling the REFER message to perform transfers, enable in server mode, disable in client mode
registrar:	bool: Allow the SIP module to receive registration requests, enable in server mode, disable in client mode.
options:	bool: Build and send a default 200 answer to OPTIONS requests.
prack:	bool: Enable acknowledging provisional 1xx answers (RFC 3262).
info:	bool: Accept incoming INFO messages.
fork:	bool: Follow first forked 2xx answer on early dialogs.
progress:	bool: Send an "183 Session Progress" just after successful routing.
generate:	bool: Allows YATE messages to send arbitrary SIP client transactions.
nat:	bool: Enable automatic NAT support.
ignorevia:	bool: Ignore Via headers and send answer back to the source. This violates RFC 3261 but is required to support NAT over UDP transport.
lazy100:	bool: Do not generate an initial "100 Trying" for non-INVITE transactions unless a retransmission arrives before having a final answer.
dtmfinband:	bool: Generate DTMF inband by default.
dtmfinfo:	bool: Generate INFO messages to send keypad tones.
rfc2833:	bool: Offer RFC2833 telephone-event by default.

privacy:	bool: Processes and generates privacy related SIP headers.
secure:	bool: Generates and accepts RFC 4568 security descriptors for SRTP.
forward_sdp:	bool: Includes the raw SDP body to be used as-is for forwarding RTP.
rtp_localip:	ipaddress: IP address to bind local RTP to, empty to guess best.
rtp_start:	bool: Starts RTP when sending 200 on incoming instead of receiving ACK.
multi_ringing:	bool: Accepts provisional (1xx) messages even after 180 Ringing.
refresh_nosdp:	bool: Accepts session refresh reINVITEs that lack a SDP offer.
flags:	int: Miscellaneous SIP engine flags for broken implementations.
[registrar]:	Controls the behavior when it acts as registrar.
expires_min:	int: Minimum allowed expiration time in seconds.
expires_def:	int: Default expiration time if not present in REGISTER request.
expires_max:	int: Value used to limit the expiration time to something sane.
auth_required:	bool: Automatically challenges all clients for authentication.
nat_refresh:	int: Proposed client NAT refreshes interval in seconds.
async_process:	bool: Processes registrations asynchronously.
[sip-t]	Controls the SIP-T parameter handling.
isup:	bool: Build outgoing or decode incoming application/isup bodies. If enabled an incoming application/isup body will be decoded and added to the engine message issued by the receiving channel. If the channel needs to add more then one body to an outgoing message, a multipart/mixed body will be attached to the message Defaults to disable
[codecs]	This section allows to individually enable or disable the codecs by typing the codecs name, and to set it equal to true. The codecs are mulaw, alaw , gsm , lpc10, ilbc, amr, slin , g723, g726 , g729 , amr_octet.
default:	bool: Enable all unlisted codecs by default if a transcoder exists
[methods]	Use this section to allow server processing of various SIP methods by handling YATE messages.

regexroute.conf

This module describes the routes using a configuration file in which each number is matched using regular expressions.

[priorities]	Set the priorities for the insertion of the regular expression module.
handlerchain:	a priority of 0 disables the handler entirely
preroute:	int: Priority of the prerouting message handler
route:	int: Priority of the routing message handler
preroute:	Priority of the prerouting message handler
route:	int: Priority of the routing message handler
extended:	bool: Make the regular expressions case insensitive
prerouteall:	bool: Preroute even calls having a context or with empty caller
[$once]	First-time only global variables initialization. It is executed during first initialization before the [$init] section
[$init]	Reload time global variables initialization
[extra]	This section is used by the prerouting handler to classify calls by the caller name
[default]	This section is used by the routing handler to find the target of calls by the called name.

A basic form of regular expressions is as follow:

^	matches start of string
$	matches end of string
.	matches any character
[list]	matches one character in the list
[^list]	matches one character not in list

*	matches proceeding expression any number of times (including zero)
\+	matches proceeding expression at least one time
\?	matches proceeding expression zero or one time
\{N\}	matches proceeding expression exactly N times
\{N,\}	matches proceeding expression N or more times
\{N,M\}	matches proceeding expression between N and M times
\(\)	captures the contained subexpression

The following are some Examples for written expressions:

A number dialed to call a user linked to a trunk and with a specific codec:

```
^{number}&=sig/{user}:trunk={trunk name}:format={codec}
```

YATE	regexroute.conf
^0100&=sig/100:trunk=ISUP-LINK:format=alaw	

Routing a sip uri:

```
^{number}$=sip/sip:{user}@{ip_addres}
```

YATE	regexroute.conf
^200$=sip/sip:200@192.168.10.174	

Routing a number for any channel on an E1:

```
^{number}$={channel};format={codec}
```

YATE	regexroute.conf
^300$=zap/1-31;format=alaw	

4. Laboratory experiments

4.1. Testing using two virtual machines

Within the following processes, I am going to explain the application of two different servers that will deliver a connection between them. This method is called SIP trunk. As we have mentioned above, we have three or more protocols for establishing a connection between the users and servers. One of these methods is the SIP trunk. The application includes the usage of two stations that run Ubuntu 10.4. They all belong to the same subnet. One of the stations have YATE version 3.3.2 which has the following configuration:

YATE	regexroute.conf

```
[extra]
 call.ringing=80
[call.ringing]
[contexts]
.*=default
[default]
^2\(..\)$=sip/sip:\0;line=SIP_TRN
```

YATE	accfile.conf

```
[SIP_TRN]
enabled=yes
protocol=sip
username=SIP_TRN
authname=SIP_TRN
password=sip_trn
outbound=192.168.1.3
localaddress=192.168.1.2
```

YATE	regfile.conf

```
[general]
autocreate=yes
auth=100
register=100
route=100
preroute=100
```

```
[100]
enabled=yes
username=100
password=100
```

The second station runs Asterisk version 1.4.17 that is released particularly for Ubuntu, and has the following configuration:

Asterisk	extensions.conf

```
[general]
static=yes
writeprotect=yes

[globals]
[context1]
exten => _1XX,1,Dial(SIP/SIP_TRN/${EXTEN},20)
```

Asterisk	sip.conf

```
[general]
disallow=all
allow=alaw
allow=gsm
canreinvite=no
qualify=1000
dtmfmode=rfc2833

[SIP_TRN]
type=friend
username=SIP_TRN
secret=sip_trn
context=context1
host=192.168.10.174
transport=udp
canreinvite=no
```

```
[300]
type=friend
username=300
secret=300
nat=yes
host=dynamic
context=context1

[100]
type=friend
username=100
secret=100
nat=yes
host=dynamic
context=context1
```

PC1 and PC2 stations are connected via an IP network. The IP address for PC1 which has YATE is 192.168.1.2, and for the PC2 is 192.168.1.3 for Asterisk. Each station is recorded and both assigned by using SIP accounts. PC1 has the username "100", and PC2 has the username "200". The signaling between the two PBXs is fulfilled as shown in figure 18 & 19. All stations have set up accounts for making test calls required.

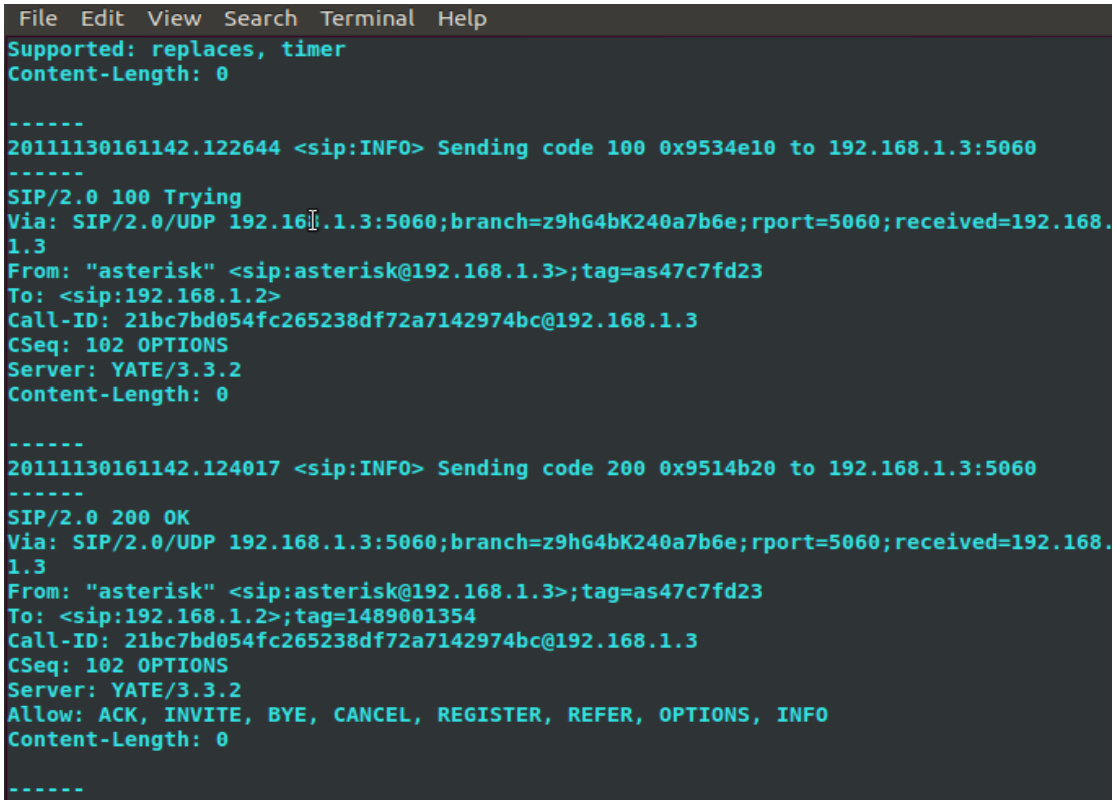


Figure18, the signaling that starts from PC1 to PC2

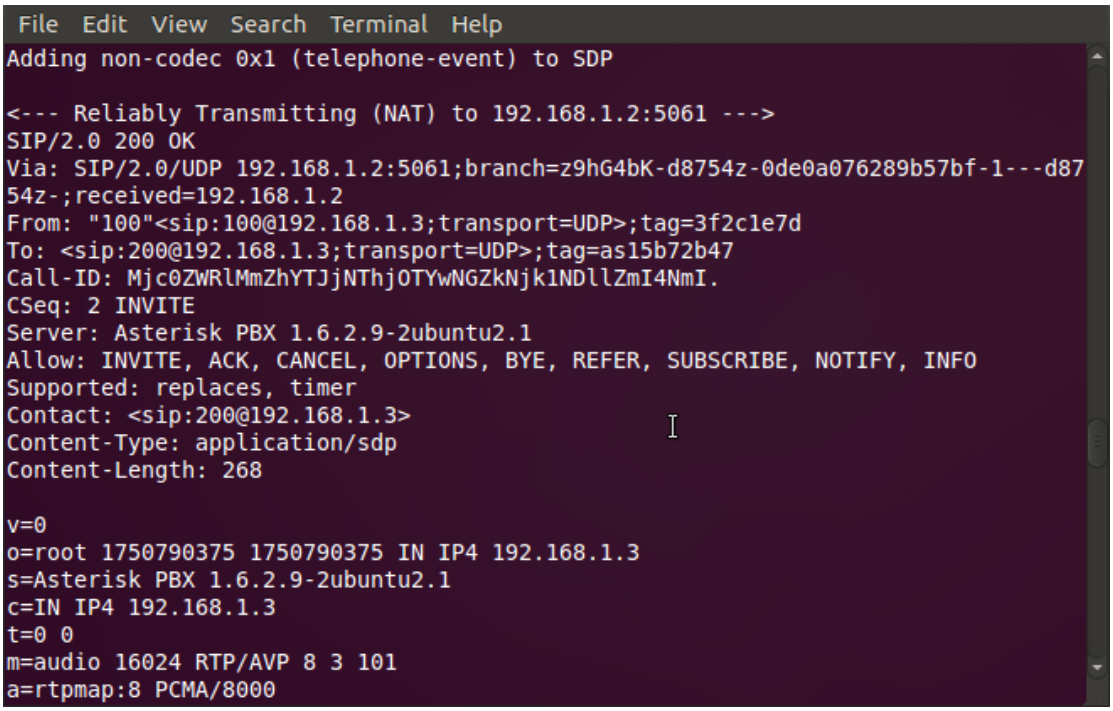


Figure 19, the signaling that starts from PC2 to PC1

Checking the connection can be done from both sides. From the Asterisk side, to verify the connection between PC1 and PC2, through which the status line will be shown. See Figure14. To verify the mutual registration status of PC1 and PC2, we can use the command sip show peers. (See figure 20). While from the YATE side by typing status sip accounts.

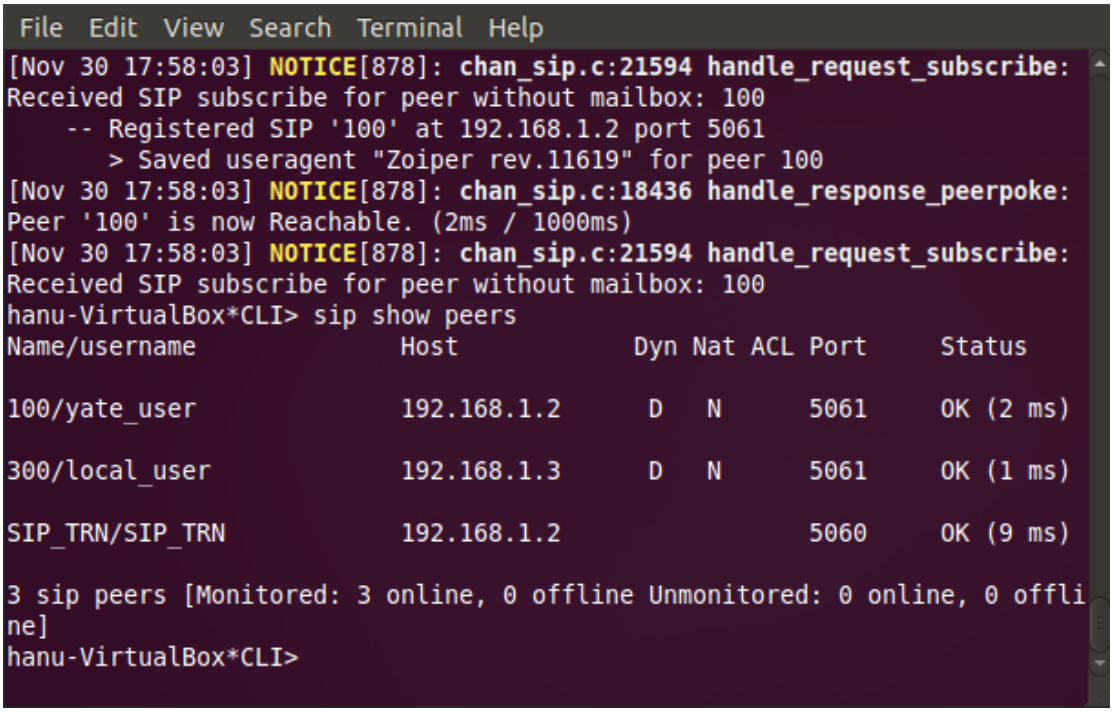


Figure 20 - SIP - verification of registered stations

Test calls are made in both directions PC1 and PC2. Both test calls were fine. The console listings of stations during test calls are shown on the drawings (figures 21, 22 & 23).

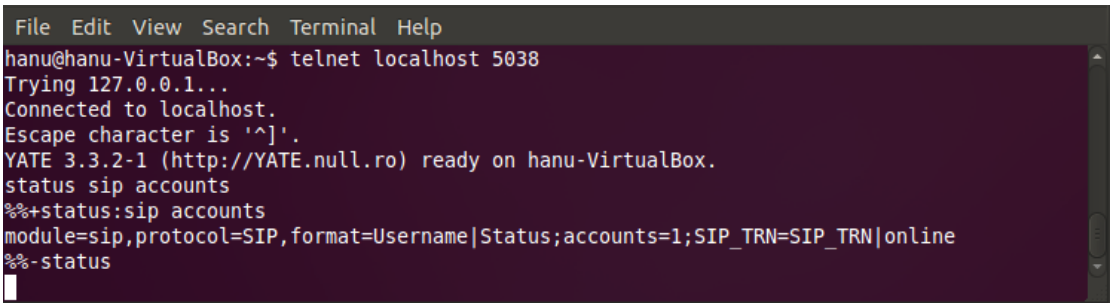


Figure 21 - Verification of the SIP registration station.

```
File  Edit  View  Search  Terminal  Help
hanu-VirtualBox*CLI> sip show users
Username                  Secret          Accountcode      Def.Context      ACL  NAT

100                       100                              context1         No   Always

SIP_TRN                   sip_trn                          context1         No   RFC3581

300                       300                              context1         No   Always

hanu-VirtualBox*CLI>
```

Figure 22 – The SIP users.

```
File  Edit  View  Search  Terminal  Help
    -- Got SIP response 486 "Busy Here" back from 192.168.1.2
    -- SIP/SIP_TRN-00000023 is busy
  == Everyone is busy/congested at this time (1:1/0/0)
    -- Auto fallthrough, channel 'SIP/100-00000022' status is 'BUSY'
  == Using SIP RTP CoS mark 5
    -- Executing [200@context1:1] Dial("SIP/100-00000024", "SIP/SIP_TRN/200,21")
 in new stack
  == Using SIP RTP CoS mark 5
    -- Called SIP_TRN/200
    -- SIP/SIP_TRN-00000025 is ringing
    -- Got SIP response 486 "Busy Here" back from 192.168.1.2
    -- SIP/SIP_TRN-00000025 is busy
  == Everyone is busy/congested at this time (1:1/0/0)
    -- Auto fallthrough, channel 'SIP/100-00000024' status is 'BUSY'
  == Using SIP RTP CoS mark 5
    -- Executing [200@context1:1] Dial("SIP/100-00000026", "SIP/SIP_TRN/200,21")
 in new stack
  == Using SIP RTP CoS mark 5
    -- Called SIP_TRN/200
    -- SIP/SIP_TRN-00000027 is ringing
    -- SIP/SIP_TRN-00000027 answered SIP/100-00000026
    -- Packet2Packet bridging SIP/100-00000026 and SIP/SIP_TRN-00000027
  == Spawn extension (context1, 200, 1) exited non-zero on 'SIP/100-00000026'
hanu-VirtualBox*CLI>
```

Figure 23 - call in progress.

The sip signaling flow between the PC1 & PC2:

```
  -- Called SIP_TRN/200

<--- SIP read from UDP:192.168.1.2:5060 --->
SIP/2.0 100 Trying
Via: SIP/2.0/UDP
192.168.1.3:5060;branch=z9hG4bK3a0c832b;rport=5060;received=192.168.1.3
From: "100" <sip:100@192.168.1.3>;tag=as6294b786
To: <sip:200@192.168.1.2>
Call-ID: 673a7824505503de3a84c3df0c300b4d@192.168.1.3
CSeq: 102 INVITE
Server: YATE/3.3.2
Content-Length: 0
```

```
<------------->
--- (8 headers 0 lines) ---

<--- SIP read from UDP:192.168.1.2:5060 --->
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP
192.168.1.3:5060;branch=z9hG4bK3a0c832b;rport=5060;received=192.168.1.3
From: "100" <sip:100@192.168.1.3>;tag=as6294b786
To: <sip:200@192.168.1.2>;tag=1724916170
Call-ID: 673a7824505503de3a84c3df0c300b4d@192.168.1.3
CSeq: 102 INVITE
Server: YATE/3.3.2
Contact: <sip:200@192.168.1.2:5060>
Allow: ACK, INVITE, BYE, CANCEL, REGISTER, REFER, OPTIONS, INFO
Content-Length: 0


<------------->
--- (10 headers 0 lines) ---
    -- SIP/SIP_TRN-00000033 is ringing

<--- SIP read from UDP:192.168.1.2:5060 --->
SIP/2.0 200 OK
Via: SIP/2.0/UDP
192.168.1.3:5060;branch=z9hG4bK3a0c832b;rport=5060;received=192.168.1.3
From: "100" <sip:100@192.168.1.3>;tag=as6294b786
To: <sip:200@192.168.1.2>;tag=1724916170
Call-ID: 673a7824505503de3a84c3df0c300b4d@192.168.1.3
CSeq: 102 INVITE
Server: YATE/3.3.2
Contact: <sip:200@192.168.1.2:5060>
Allow: ACK, INVITE, BYE, CANCEL, REGISTER, REFER, OPTIONS, INFO
Content-Type: application/sdp
Content-Length: 181

v=0
o=yate 1322961069 1322961069 IN IP4 192.168.1.2
s=SIP Call
c=IN IP4 192.168.1.2
t=0 0
m=audio 26512 RTP/AVP 8 101
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000

<------------->
--- (11 headers 8 lines) ---
Found RTP audio format 8
Found RTP audio format 101
Found audio description format PCMA for ID 8
Found audio description format telephone-event for ID 101
Capabilities: us - 0xa (gsm|alaw), peer - audio=0x8 (alaw)/video=0x0
(nothing)/text=0x0 (nothing), combined - 0x8 (alaw)
Non-codec capabilities (dtmf): us - 0x1 (telephone-event), peer - 0x1
(telephone-event), combined - 0x1 (telephone-event)
Peer audio RTP is at port 192.168.1.2:26512
```

```
list_route: hop: <sip:200@192.168.1.2:5060>
set_destination: Parsing <sip:200@192.168.1.2:5060> for address/port to
send to
set_destination: set destination to 192.168.1.2, port 5060
Transmitting (no NAT) to 192.168.1.2:5060:
ACK sip:200@192.168.1.2:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.3:5060;branch=z9hG4bK0166870e;rport
Max-Forwards: 70
From: "100" <sip:100@192.168.1.3>;tag=as6294b786
To: <sip:200@192.168.1.2>;tag=1724916170
Contact: <sip:100@192.168.1.3>
Call-ID: 673a7824505503de3a84c3df0c300b4d@192.168.1.3
CSeq: 102 ACK
User-Agent: Asterisk PBX 1.6.2.9-2ubuntu2.1
Content-Length: 0


---
    -- SIP/SIP_TRN-00000033 answered SIP/100-00000032
    -- Packet2Packet bridging SIP/100-00000032 and SIP/SIP_TRN-00000033

<--- SIP read from UDP:192.168.1.2:5060 --->
BYE sip:100@192.168.1.3 SIP/2.0
Call-ID: 673a7824505503de3a84c3df0c300b4d@192.168.1.3
From: <sip:200@192.168.1.2>;tag=1724916170
To: <sip:100@192.168.1.3>;tag=as6294b786
P-RTP-Stat: PS=161,OS=25760,PR=164,OR=26240,PL=0
Via: SIP/2.0/UDP 192.168.1.2:5060;rport;branch=z9hG4bK411826969
CSeq: 7 BYE
User-Agent: YATE/3.3.2
Max-Forwards: 70
Allow: ACK, INVITE, BYE, CANCEL, REGISTER, REFER, OPTIONS, INFO
Content-Length: 0


<------------->
--- (11 headers 0 lines) ---
Sending to 192.168.1.2 : 5060 (no NAT)

<--- Transmitting (no NAT) to 192.168.1.2:5060 --->
SIP/2.0 200 OK
Via: SIP/2.0/UDP
192.168.1.2:5060;branch=z9hG4bK411826969;received=192.168.1.2;rport=5060
From: <sip:200@192.168.1.2>;tag=1724916170
To: <sip:100@192.168.1.3>;tag=as6294b786
Call-ID: 673a7824505503de3a84c3df0c300b4d@192.168.1.3
CSeq: 7 BYE
Server: Asterisk PBX 1.6.2.9-2ubuntu2.1
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY, INFO
Supported: replaces, timer
Content-Length: 0
```

4.2 Laboratory experiment for physical connections

Our aim in this experiment is to examine YATE and test its capabilities for adopting the SS7 over IP. The protocols involved to achieve the aim as we mentioned are, SIGTRAN, SIP-T. Unfortunately the BICC it's not done yet by the developers so it will not be a part of our implementation in the following sections. Our simple network that we are going to use is illustrated in (Figure 24).This simplified network is sufficient for our configurations. All the stations were named and addressed in such a way that node2 for example has a point code 2 and users' range from 200 to 299.

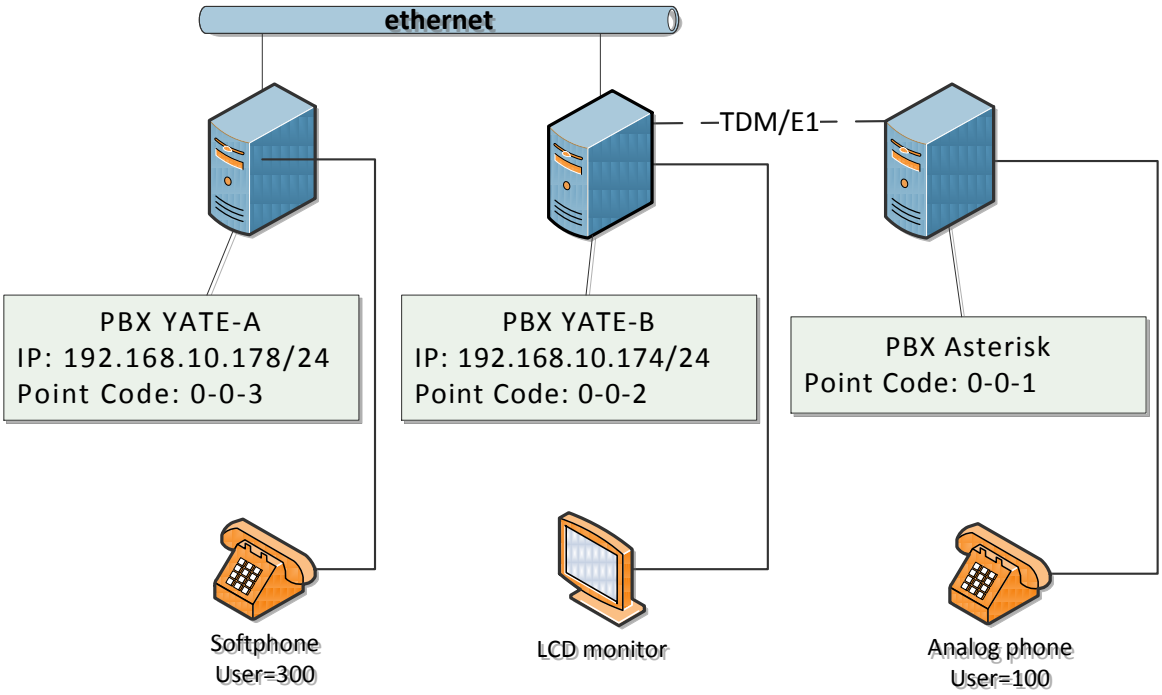


Figure 24 – Illustration of a simple diagram that was used to accomplish the configurations

4.2.1 SS7 over SIGTRAN

4.2.1.1. SS7 configuration

In this scenario, we shall create a connection between the two servers using a TDM connection. In order to establish this connection, both sides should agree for a specific configuration after installing all the drivers and necessary cards. On the YATE side, we should add and modify the configuration files; ysigchan.conf, zapcard.conf and regexroute.conf. From

ASTERISK side, we should also create and modify SS7.conf and extensions.conf files, in term of the linkset, by identifying the point code type, the MSU that contains the Originating Point Code (OPC) and the Destination Point Code (DPC), D and B channels, and some optional sets.
In Ysigchan.conf where we identified the top three layers - ISUP, mtp3, and mtp2 - we created the forth layer (shown below) by setting a point code type of ITU, the Originating Point Code of (0-0-2), Destination Point Code (0-0-1) that refers to the next SSP endpoint, signaling and voice channel and some sets such as format type, Signaling Link Selection and numbering plan, etc.

YATE-B	Ysigchan.conf

```
[link2-1]               ; ISUP that interconnects with node 0-0-1

enable=yes                    ; enable the ISUP link
type=ss7-isup                 ; the type of the link
pointcodetype=ITU
pointcode=2                   ; originating point code
defaultpointcode=2
remotepointcode=1             ;  destination point code
lockgroup=yes
earlyacm=yes                  ; convert ACM to stat ringing
sig=span1                     ; signaling channel
voice=span1                   ; voice channel
channelsync=0
numplan=isdn                  ; numbering plan
netindicator=national         ; network indicator in the SIO
numtype=national             ;the number type for outgoing calls
netind2pctype=ITU.ITU,ITU,ITU
presentation=allowed          ; default number presentation
format=alaw                   ;codec
inn=yes
sls=auto
```

For the third layer, we created a linkset named [linkset-0-0-2] which should also have a local and adjacent address at the network level. It contains also the point code type and a link that should be attached to represent the second layer.

YATE-B	Ysigchan.conf

```
[linkset-0-0-2]             ;creating the mpt3 linkset
type=ss7-mtp3
netind2pctype=ITU
local=ITU,2                 ;local point code
adjacent=ITU,1              ;adjacent point code
autostart=yes
link=link1                  ;mpt2 link
link1.sig=span1             ;mpt1 link
router=ss7router            ;ss7 router

[link1]                     ;creating the mtp2 link
type=ss7-mtp2
autostart=yes
emergency=no
fillink=yes
rxunderrun=0
```

Finally, we came to the identification of the lowest layer. That can be implemented in the file called zapcard.conf which keeps the configuration of cards by using the zaptel driver. This identification of the span is shown below:

YATE-B	zapcard.conf

```
[span1]
type=E1                     ;type of connection
offset=0                    ;the start of channel
sigchan=1                   ;signaling channel
voicechans=2-31             ;voice channels
dtmfdetect=disable
```

After making the entire stack, we came to routing, in the file called regexroute.conf where we just specify the called number and the stack we created to link it to.

YATE-B	regexroute.conf

```
[contexts]
.*=default

[default]
^1\(..\)$=sig/\0;trunk=link2-1 ;calling 1XX that are linked to a
                               ;trunk
```

On the opposite side, where the ASTERISK server is connected, we modified SS7.conf to serve as the ysigchan.conf in YATE for our configuration in terms of link sets, links, point code type, channels, etc.

Asterisk	SS7.conf

```
[linkset-TDM]
enabled => yes
enable_st => no
use_connect => yes          ;Reply incoming call with CON
hunting_policy => even_mru  ;The CIC hunting policy
context => context1
language => cz              ;The language for this context
t35 => 15000,timeout
subservice => auto
variant => ITU

[link-l1]
linkset => TDM
channels => 2-31           ;voice channels
schannel => 1              ;signaling channel
firstcic => 1              ;the same as offset in YATE
enabled => yes
echocancel => no
echocan_train => 350
echocan_taps => 128
rxgain => 0.0
txgain => 0.0
relaxdtmf => no
```

```
[host-PC-Alcatel]
enabled => yes
opc => 0x03            ;originating point code
dpc => TDM:0x04        ;destination point code
links => l1:1
ssn => 7
```

We did the same for extensions.conf which is the equivalent module for the regexroute.conf in Asterisk as shown below:

Asterisk	extensions.conf:

```
[general]
static=yes
writeprotect=yes

[globals]
TRUNK1 = SS7/TDM        ;TDM/E1 trunk

[context1]
exten => _[2-3]XX,1,Dial(${TRUNK1}/${EXTEN},20)
```

By applying all those configurations, the two servers now should verify the connection by three Signaling Network Management processes. The first process is the Traffic management which is responsible for dealing with signaling traffic. If the signaling link was activated, MTP3 sends an Signaling Link Test Message (SLTM) to the far end over the link with the node's point code. The link on which the message was sent can be identified from the routing label. The test is performed only if the SLTM matches the link on which the message was sent, and if the Destination Point Code in the routing label matches the Originating Point Code, the receiving node responds with a Signaling Link Test Acknowledgement (SLTA) containing the test pattern received in the SLTM message as shown in figure 25.

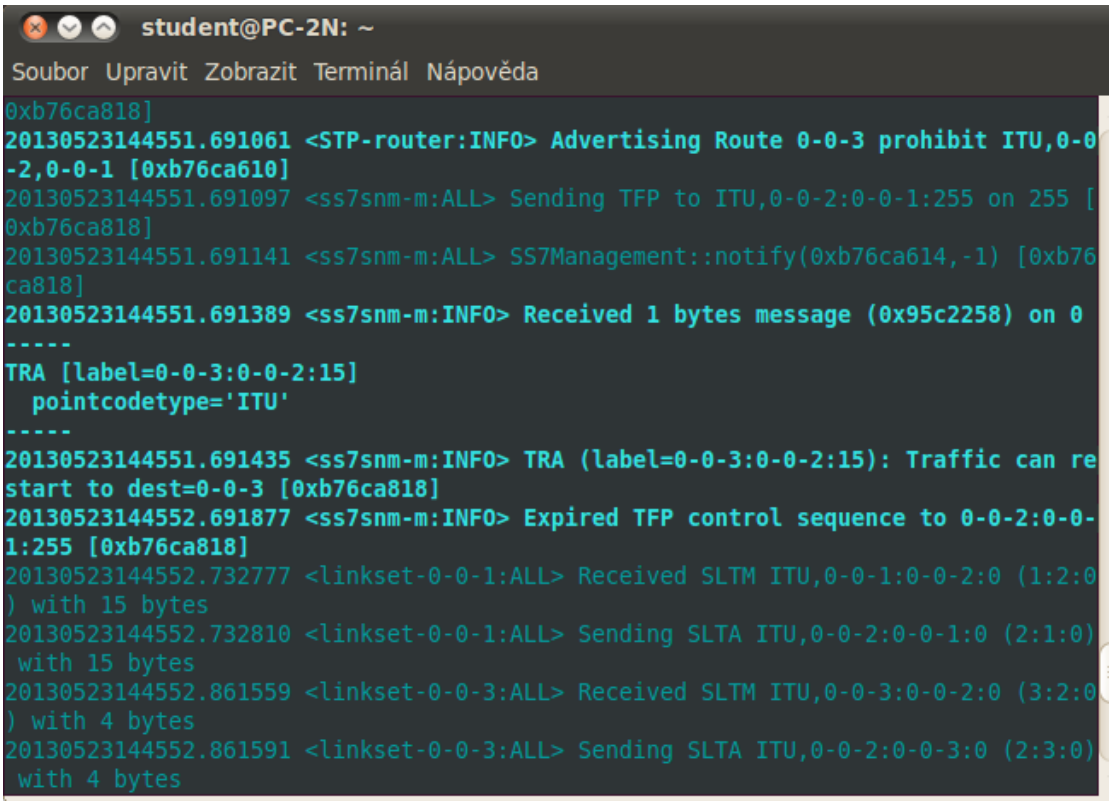


Figure 25 – SLTM and SLTA messages.

Routing management, which is the second process of Signaling Network Management that its main objective is to notify the nodes for events occurring that affect the route availability. Route Management sends notification messages to notify other nodes about the change in routing states, those notification messages are Transfer Prohibited (TFP), Transfer Restricted (TFR), Transfer Allowed (TFA) and Transfer Controlled (TFC). Routing management supplies information to traffic management and allows it to adjust traffic patterns and flow accordingly.

The last process is the Link management which activates, deactivates and restores signaling links. This process notifies the MTP users about the availability of signaling links, and invoking procedures to restore service when a disruption appears. This part of network management is very close to the physical hardware.

To verify the connection with the YATE console we may enter the command “control STP-router show” to identify whether the routing is allowed or prohibited. If allowed, then the traffic can reach the destination. As we can see in figure 26, the router is able to reach the destination of the two interconnected nodes.

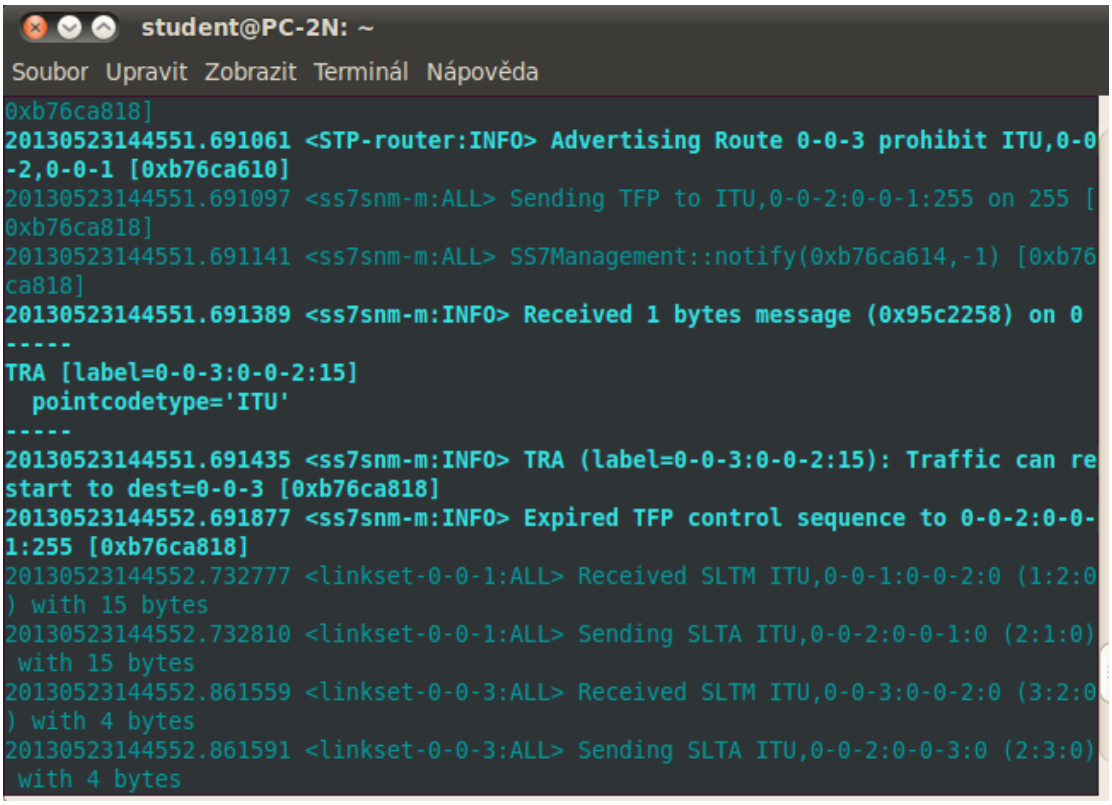


Figure 26 – STP routing table.

4.2.1.2. SIGTRAN configuration

SCTP is a significant protocol defined by the SIGTRAN group, which is used to carry PSTN signaling over IP. On this part a SCTP connection shall be implemented, through two YATE servers. To perform this connection, first we need to create an SCTP socket that can be implemented on the sigtransport.conf, and then we take that socket we named and link it to the ysigchan.conf exactly where we created the mtp2 and replace it with the m2pa. M2PA matches the MPT2 in the traditional SS7 for IP networks.

Creating a sctp connection can be done in the sigtransport.conf (viewed below for two YATE servers), where we should choose the socket type (sctp, udp, tcp, etc), local and destination IP addresses and ports and the endpoint type whether to try to connection or to a listening state for an incoming connection. However for implementing two completely independent signaling nodes, it is necessary to define two independent management and router parts in the ysigchan.conf.

YATE-A	ysigchan.conf:	YATE-B	ysigchan.conf:
[link3] type=sctp local=192.168.10.178:3566 remote=192.168.10.174:3566 stream=false endpoint=true		[link2] type=sctp local=192.168.10.174:3566 remote=192.168.10.178:3566 stream=false endpoint=true	

Then we take those links we created and link them to ysigchan.conf as it is viewed below:

YATE-A	ysigchan.conf:	YATE-B	ysigchan.conf:
[linkset-0-0-2] type=ss7-mtp3 netind2pctype=ITU local=ITU,0-0-3 adjacent=ITU,0-0-2 autostart=yes link=link3 emergency=yes router=ss7-router [link3] type=ss7-m2pa autostart=yes sig=link5		[linkset-0-0-3] type=ss7-mtp3 netind2pctype=ITU local=ITU,0-0-2 adjacent=ITU,0-0-3 autostart=yes link=link2 emergency=yes router=ss7-router [link2] type=ss7-m2pa autostart=yes sig=link5	

As mentioned, the two independent management and router parts should be included whether it is a Service Switching Point (SSP), or a Signal Transfer Point (STP). If we wish to make the YATE exchange work as an SSP, we set the parameter “transfer” that belongs to ss7-router section equals to “no”, otherwise to “yes” if the actual point serves as an STP.

YATE-A	ysigchan.conf:	YATE-B	ysigchan.conf:

```
[ss7-router]
type=ss7-router
local=ITU,0-0-3
transfer=no
management=ss7-mn

[ss7-mn]
type=ss7-snm
router=ss7router
local=ITU,0-0-3
autoallow=yes
changemsgs=yes
changesets=yes
neighbours=yes
```

```
[ss7-router]
type=ss7-router
local=ITU,0-0-2
transfer=no
management=ss7-mn

[ss7-mn]
type=ss7-snm
router=ss7router
local=ITU,0-0-2
autoallow=yes
changemsgs=yes
changesets=yes
neighbours=yes
```

After enabling all the configurations, the two nodes should exchange signaling messages to determine whether the end point can be reached by the Signaling Network Management processes or not, that were described in the previous section.

4.2.2.3. Implementing SS7 over SIGTRAN

After what was conducted so far, we came to a point where we integrated all the previous three nodes in such way that one of these nodes serves as an STP node. The STP node that was chosen is the one located between them (YATE-B) that supports the TDM and the Ethernet connection, for the purpose of performing the SS7 signaling over IP and vice versa.

To setup this connection we need to keep the two linksets made; the first linkset named [linkset-0-0-3] interconnects to YATE-A and the linkset [linkset-0-0-1] to Asterisk, the remaining task to do is to disable or remove the 4th layer that we created on YATE-B and set the parameter “transfer=yes” to enable YATE-B to serve as STP node.

YATE-B	ysigchan.conf:

```
[STP-router]
type=ss7-router          ;SS7 router
local=ITU,2              ;local point
transfer=yes             ;enable the STP router
management=ss7-mn

[ss7-mn]
type=ss7-snm               ;SS7 management
local=ITU,2                ;local point
neighbours=yes
```

On the left linkeset that is shown below, it interconnects with the next SSP node that has DPC of [0-0-3], Ethernet connection (SIGTRAN) while the right interconnects with SSP node that has DPC of [0-0-1], E1 TDM connection (the traditional SS7).

YATE-B	ysigchan.conf:

```
[linkset-0-0-3]
type=ss7-mtp3
netind2pctype=ITU
local=ITU,0-0-2
adjacent=ITU,0-0-3
autostart=yes
link=link2
emergency=yes
router=ss7router


[link2]
type=ss7-m2pa
autostart=yes
sig=link2
```

```
[linkset-0-0-1]
type=ss7-mtp3
netind2pctype=ITU
local=ITU,0-0-2
adjacent=ITU,0-0-1
autostart=yes
link=link1
link1.sig=span1
emergency=no
router=ss7router

[link1]
type=ss7-mtp2
autostart=yes
emergency=no
fillink=yes
```

Now we come to the final part, where we managed the ISUP for both ends with consideration that these ISUPs are not referring to the STP node in term of addressing while the linksets in the lowest

layers should contain the STP node addresses. The identification of the ISUP and the linksets are shown below where the configuration file of ysigchan.conf in YATE-A is on the right side while the file SS7.conf in Asterisk on the left side.

YATE-A	ysigchan.conf:	Asterisk	SS7.conf:

```
[link3-1]
enable=yes
type=ss7-isup
pointcodetype=ITU
pointcode=3
defaultpointcode=3
remotepointcode=1
lockgroup=yes
earlyacm=yes
sig=link3
voice=span1
channelsync=0
numplan=isdn
netindicator=national
numtype=national
netind2pctype=ITU
presentation=allowed
format=alaw
inn=yes
sls=auto

[linkset-0-0-2]
type=ss7-mtp3
netind2pctype=ITU
local=ITU,0-0-3
adjacent=ITU,0-0-2
route=ITU,1,100
autostart=yes
link=link3
emergency=yes
router=ss7-router
```

```
[linkset-TDM]
enabled => yes
enable_st => no
use_connect => yes
hunting_policy => even_mru
context => context1
language => cz
t35 => 15000,timeout
subservice => auto
variant => ITU

[link-l1]
linkset => TDM
channels => 2-31
schannel => 1
firstcic => 1
enabled => yes
echocancel => no
echocan_train => 350
echocan_taps => 128
rxgain => 0.0
txgain => 0.0
relaxdtmf => no

[host-PC-Alcatel]
enabled => yes
opc => 0x01
dpc => siuc:0x03,l1:0x02
links => l1:1
ssn => 7
```

```
[link3]
type=ss7-m2pa
autostart=yes
sig=link5
```

In the routing step, the configuration of the two nodes can be made as it is viewed below:

YATE-A	regexroute.conf	Asterisk	extensions.conf:

```
[contexts]
.*=default

[default]
^1\(..\)$=sig/\0;trunk=link3-1
```

```
[globals]
TRUNK1 = SS7/TDM

[context1]
exten =>
_3XX,1,Dial(${TRUNK1}/${EXTEN},20)
```

Forwarding the voice using the MGCP gateway was difficult since it doesn't supports the TDM/E1 therefore we added two MGCP gateways and one call agent in YATE-B; for YATE-A a call agent, and for Asterisk some additional configuration that can be made in YATE-B to forward the voice in E1.

Creating the MGCP gateway and the MGCP call agent in YATE can be implemented by the support of the files called mgcpgw.conf, and mgcpca.conf. They are configured and viewed below:

YATE-B	mgcpgw.conf:

```
[engine]
enabled=yes                    ;enable MGCP gateway
address=192.168.10.174
port=2427
request_ack=yes   ;Request an Acknowledge of the transactions
send_provisional=no


[ep yatea]
local_user=yatea               ;endpoint registration
local_host=192.168.10.174
remote_host=192.168.10.178
remote_port=2727
```

```
[ep assist]
local_user=assist
local_host=192.168.10.174
remote_host=192.168.10.174
remote_port=2727
```

And for the call agent that is configured in the same node which helps to carry the voice between the node (0-0-3) and the node (0-0-1) is shown below:

YATE-B	mgcpca.conf:

```
[engine]
enabled=yes            ;Enable the MGCP engine
address=192.168.10.174
port=2727
request_ack=yes    ;Request an Acknowledge of the transactions
send_provisional=no   ;don't Send 1xx provisional answers


[endpoint]             ;The user part of the local endpoint
user=assist
host=192.168.10.174
port=2727

[codecs]
default=no
mulaw=no
alaw=yes
gsm=no
lpc10=no
ilbc=no

[gw assist]
user=assist_22           ;Remote MGCP resource name
host=192.168.10.174      ;the hostname of the gateway
port=2427                ;Remote port to send packets
voicechans=1-10          ; The range of channels used for
                         ;voice (data) transfer
chans=10                 ;The number of E1 circuits
offset=21
bearer=alaw              ;Default bearer encoding
match_ntfy=yes
req_fax=no
req_t38=no
req_dtmf=none ;do not request a notification about DTMF events
forward_sdp=yes
```

The implementation for the two ISUPs that is managed to carry the voice, are shown bellow:

YATE-B	ysigchan.conf:

```
[ISUP-ASSIST]
   enable=yes
   type=ss7-isup
   pointcodetype=ITU
   pointcode=1-1-1
   defaultpointcode=1-1-1
   remotepointcode=2-2-2
   lockgroup=yes
   earlyacm=yes
   voice=span2_a
   strategy=increment
   channelsync=0
   numplan=isdn
   netindicator=national
   numtype=national
   netind2pctype=ITU.
   presentation=allowed
   screening=user-provided
   format=alaw
   print-messages=yes
   extended-debug=yes
   userparttest=0
   inn=no
   emergency=yes
   ringback=yes
   sls=cic
   conform_ccr=no
   ignore-grs-single=no
   ignore-cgb-single=no
   ignore-cgu-single=no
   continuity=no
   lockgroup=yes
   router=ASSIST-router
```

```
[ISUP-ASSIST2]
   enable=yes
   type=ss7-isup
   pointcodetype=ITU
   pointcode=2-2-2
   defaultpointcode=2-2-2
   remotepointcode=1-1-1
   lockgroup=yes
   earlyacm=yes
   voice=assist
   strategy=increment
   channelsync=0
   numplan=isdn
   netindicator=national
   numtype=national
   netind2pctype=ITU.
   presentation=allowed
   screening=user-provided
   format=alaw
   print-messages=yes
   extended-debug=yes
   userparttest=0
   inn=no
   emergency=yes
   ringback=yes
   sls=cic
   conform_ccr=no
   ignore-grs-single=no
   ignore-cgb-single=no
   ignore-cgu-single=no
   continuity=no
   lockgroup=yes
   router=ASSIST2-router
```

Followed are the linksets that represent the third and the second layer for the (ISUP-ASSIST) and (ISUP-ASSIST2) which are used to complete the two ISUPs that have been created:

YATE-B	ysigchan.conf:
[linkset10-MTP3-ASSIST] type=ss7-mtp3 netind2pctype=ITU local=ITU,1-1-1 adjacent=ITU,2-2-2 autostart=yes link=link10-MTP2-ASSIST emergency=yes checklinks=false checkfails=0 maintenance=0 router=ASSIST-router [link10-MTP2-ASSIST] type=ss7-m2pa sig=link10-MTP2-ASSIST autostart=yes	[linkset11-MTP3-ASSIST2] type=ss7-mtp3 netind2pctype=ITU local=ITU,2-2-2 adjacent=ITU,1-1-1 autostart=yes link=link11-MTP2-ASSIST2 emergency=yes checklinks=false checkfails=0 maintenance=0 router=ASSIST2-router [link11-MTP2-ASSIST2] type=ss7-m2pa sig=link11-MTP2-ASSIST2 autostart=yes

The two routers and managements that handle the connection between the two nodes to carry the voice:

YATE-B	ysigchan.conf:
[ASSIST-router] type=ss7-router local=ITU,1-1-1 transfer=no management=ss7snm-assist autostart=yes isolation=1000 testroutes=50000	[ASSIST2-router] type=ss7-router local=ITU,2-2-2 transfer=no management=ss7snm2-assist autostart=yes isolation=1000 testroutes=50000

```
[ss7snm-assist]
type=ss7-snm
router=ASSIST-router
local=ITU,1-1-1
changesets=no
neighbours=no
```

```
[ss7snm2-assist]
type=ss7-snm
router=ASSIST2-router
local=ITU,2-2-2
changesets=no
neighbours=no
```

The SCTP sockets that should be attached to the two ISUPs to complete the connection:

YATE-B	sigtransport.conf:

```
[link10-MTP2-ASSIST]
type=sctp
stream=false
local=127.0.0.1:3670   ;forward the packets to 3669 port
remote=127.0.0.1:3669  ;forward the packets to 3670 port
endpoint=true

[link11-MTP2-ASSIST2]
type=sctp
stream=false
local=127.0.0.1:3669
remote=127.0.0.1:3670
endpoint=true
```

The routing step that is used to carry the voice through MGCP gateway:

YATE-B	regexroute.conf:

```
[default]

^yatea_id\(..\)=sig/9999\1;trunk=ISUP-ASSIST;circuits=\1
^assist=tone/dial
^999922=tone/dial
^999922=tone/dial
^999923=tone/dial
^999924=tone/dial
^999925=tone/dial
^999926=tone/dial
^999927=tone/dial
^999928=tone/dial
^999929=tone/dial
^999931=tone/dial
```

If there is an incoming message from each call agent then both of them will be routed based on the ID number, for example if the gateway receives (yatea_id27) where number 27 indicates the circuit, then it will be added to number 9999 and routed through the trunk ISUP-ASSIST with the circuit 27.

Spans are used for signaling and voice channels. As shown in the below implementation example, one of the spans is called span2_a and is used for the voice channels, which is for the point code (1-1-1) called ISUP-ASSIST. Span2_sig is used for the signaling point code (0-0-1).

YATE-B	zapcard.conf

```
[span2_a]
format=alaw
type=E1
offset=0
voicechans=22-31    ;limiting the channels for the
                    ;actual span
dtmfdetect=disable

[span2_sig]
format=alaw
type=E1
offset=0
sigchan=1
```

```
dtmfdetect=disable
```

Adding a user (yatea) in the call agent that is located in the far end (YATE-A):

YATE-A	mgcpca.conf

```
[engine]
enabled=yes
address=192.168.10.178
port=0
request_ack=yes
send_provisional=no


[endpoint]
user=yatea
host=192.168.10.178
port=2727


[codecs]
default=no
alaw=yes

[gw yatea]
user=yatea_id22
host=192.168.10.174
port=2427
address=192.168.10.174
range=22-31
voicechans=1-10
chans=10
offset=22
bearer=alaw
match_ntfy=yes
req_fax=no
req_t38=no
req_dtmf=none
forward_rtp=yes
forward_sdp=yes
```

The routing expression for the call agent to send the dial tone:

YATE-A	regexroute.conf:

```
[default]
^yatea=tone/ring
```

The signaling and the voice channel that are attached to the fourth layer in the signaling point (0-0-3) will remain the same as the previous configurations except that the voice should be equal to (yatea):

YATE-A	zapcard.conf

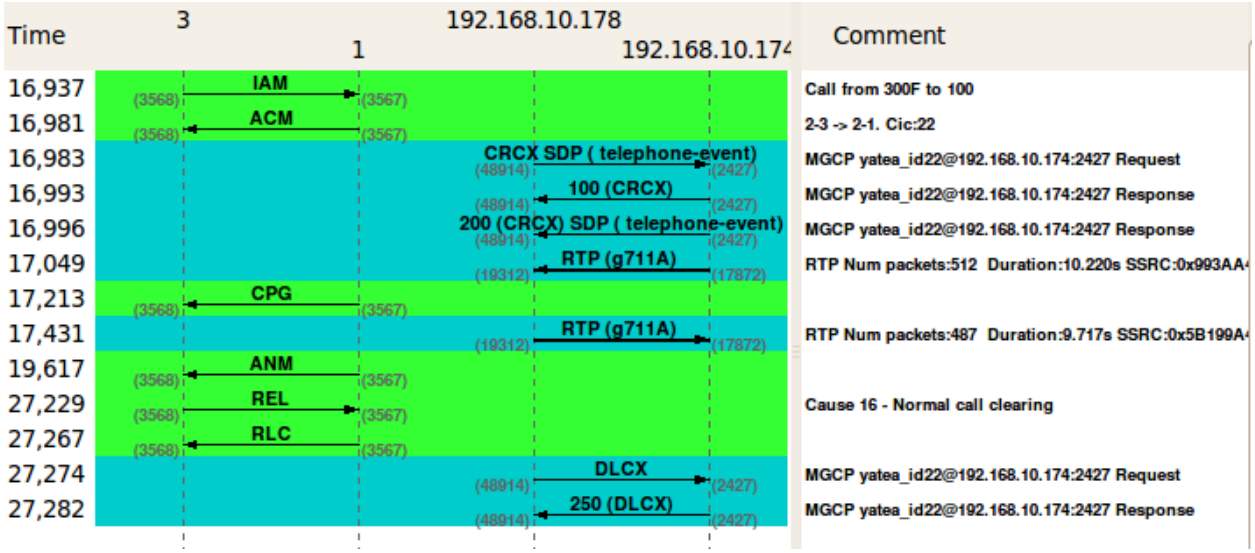
```
[span1]
type=E1
offset=0
voicechans=2-31
sigchan=1
dtmfdetect=disable
```

The sole task remaining to be conducted for the node (0-0-1) that is placed in the Asterisk, is to rearrange the channels to fit the channels that relate to the MGCP gateways (span2_a) and to the call agent as shown in the following:

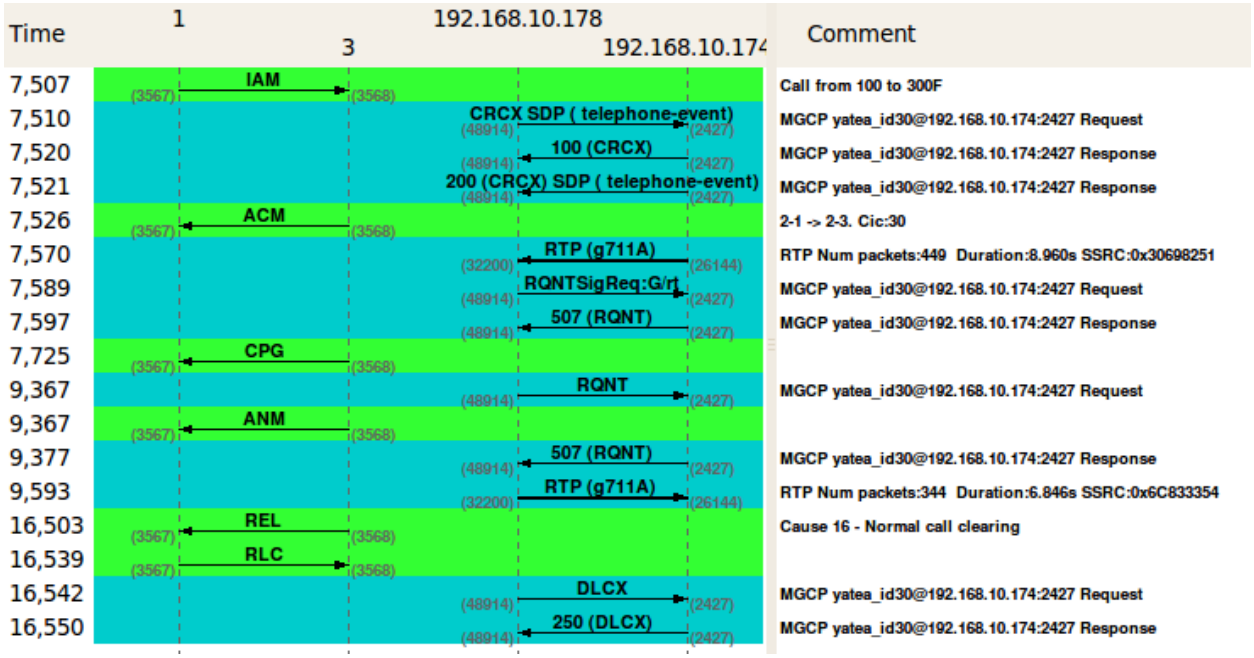
Asterisk	SS7.conf

```
[link-l1]
linkset => TDM
channels => 22-31
schannel => 1
firstcic => 1
enabled => yes
echocancel => no
echocan_train => 350
echocan_taps => 128
rxgain => 0.0
txgain => 0.0
relaxdtmf => no
```

Test calls are made in both directions for the node (0-0-1) and for the node (0-0-3). Both test calls were fine (see figure 28 & 29).



(Figure 28) the message flow of a call generated from user: 300.



(Figure 29) the message flow of a call generated from user: 100.

4.2.3 SIP-T implementation

SIP-T is one of other options to make the signaling between the PSTN and IP network. SIP-T stands for Session Initiation Protocol for Telephony. In this part, YATE-2 will be managed to work as a sip signaling proxy to adapt different links (E1 and Ethernet). To apply the sip-t in YATE, we need to refer to section [sip-t] that is located on the ysipchan.conf and set the parameter "isup" equaling to "enable" as described below:

YATE-B	ysipchan.conf

```
[general]
port=5060
addr=192.168.10.174
registrar=enable

[sip-t]

isup=enable
```

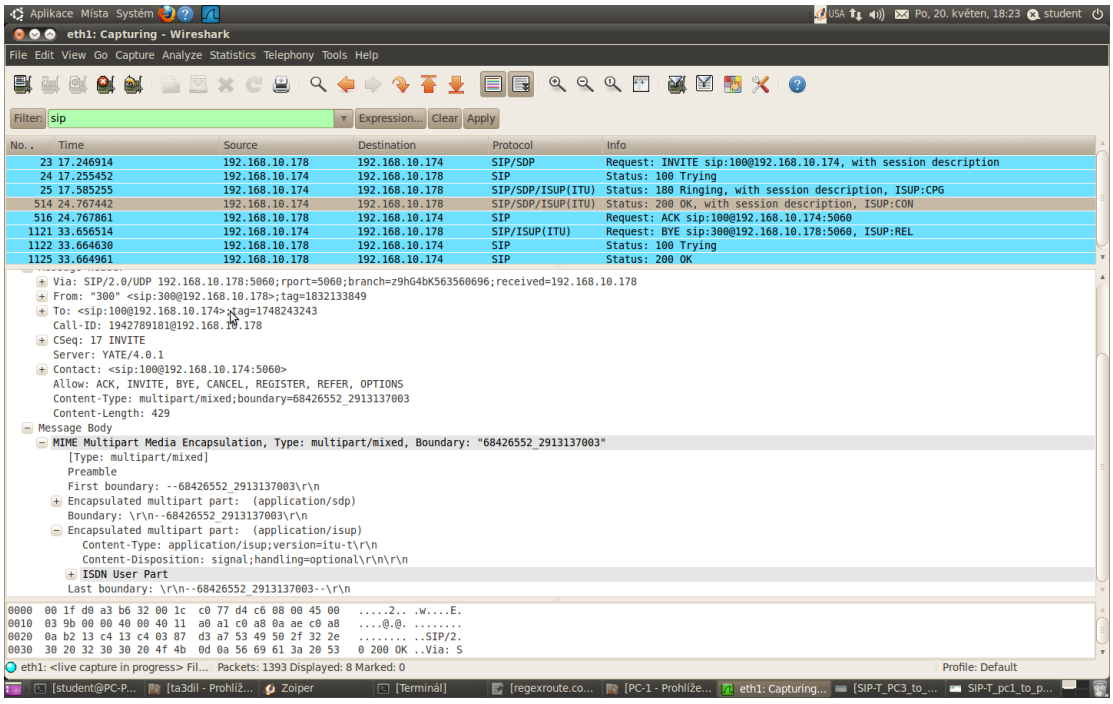
The encapsulation of the IAM message can be done in regexroute.conf (as shown below). Starting from the "message-prefix" line which tells YATE that there is an ISUP message that requires to be encoded inside the INVITE message by specific parameters related to the ISUP messages, or to be decoded for specific parameters as well, followed by including some routing procedures.

YATE-B	regexroute.conf

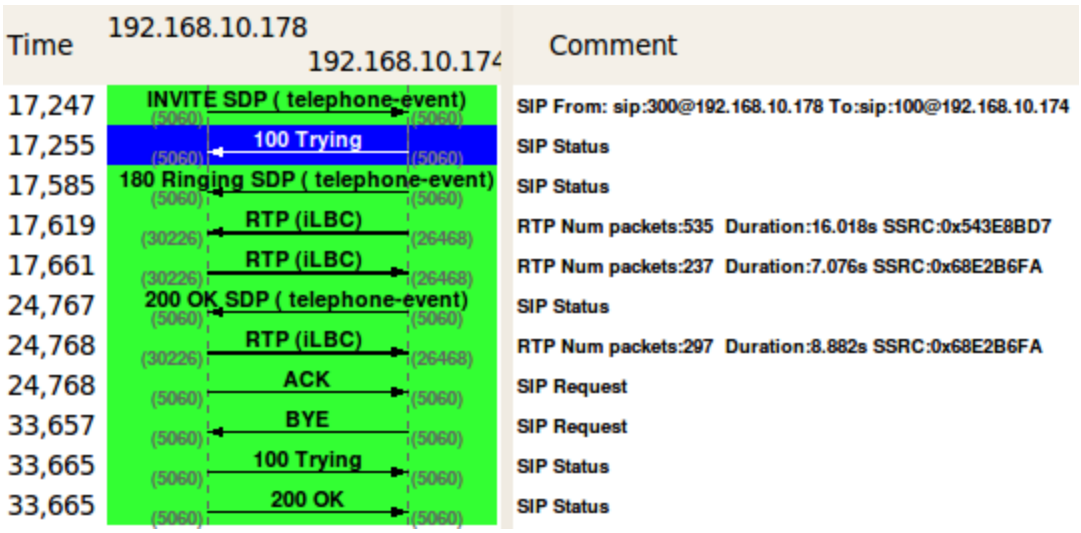
```
[contexts]
.*=default

[default]

.*=;message-prefix=isup.;    ;the start of encapsulation
  isup.message-type=IAM;
  isup.CallingPartyCategory=ordinary;
  isup.TransmissionMediumRequirement=speech;
  isup.CalledPartyNumber=${called};
  isup.CalledPartyNumber.nature=national;
  isup.CalledPartyNumber.plan=isdn;
  isup.CalledPartyNumber.inn=true;
  isup.CallingPartyNumber=${caller};
  isup.CallingPartyNumber.nature=national;
  isup.CallingPartyNumber.complete=true;
  isup.CallingPartyNumber.restrict=allowed;
```

```
    isup.CallingPartyNumber.screened=network-provided;
    isup.RedirectingNumber=${called};
    isup.RedirectingAddress=192.168.10.178;
    isup.Trunk=link2-1;


  ${isup.RedirectingNumber}=if
  ^1\(..\)*$=sig/\0;trunk=${isup.Trunk}
  ${isup.RedirectingNumber}=if
  ^3\(..\)*$=sip/\0@${isup.RedirectingAddress}
```

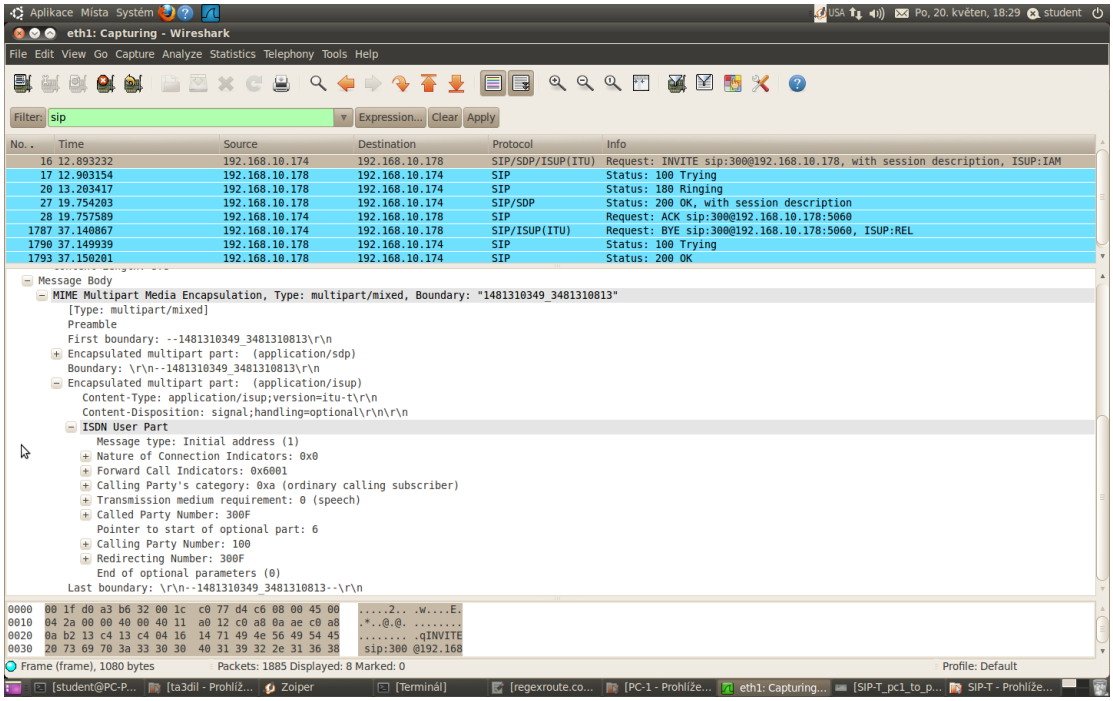
After applying the configurations, if user 100 in Asterisk wants to establish a connection with user 300 located on YATE-A then YATE-B will add the necessary information from IAM to create an INVITE message. On the contrary, if user 300 wants to establish the call to 100 then YATE-B receives an invitation message and creates an IAM message based on the INVITE message content. The result of this configuration for the first call can be seen in (figure 30 & 31) and the second call can be seen on the (figure 32 & 33).



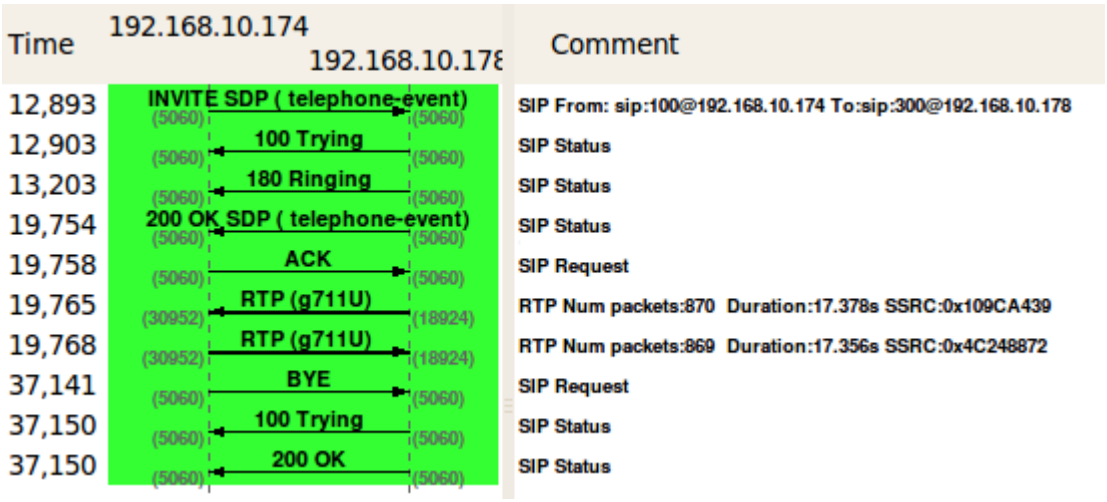
(Figure 30) illustrates the redirected call from user 300 to user 100 using the sip-t.



(Figure 31) the message flow of a call generated from user: 300.



(Figure 32) Illustrates the redirected call from user 100 to user 300 using the sip-t.



(Figure 33) the message flow of a call generated from user: 100.

4.2.3. BICC

BICC is not implemented yet in YATE by the programmer. Therefore in order to complete this part of the experiment, there are some libraries (which are listed in openSS7) which need to be applied with the requirement of excessive time consumption.

5. Conclusion

There are many kinds of open source PBX's (Private Branch Exchange) software. They might deliver the same services, except that some of them do not support some technology services in full scale. For example, YATE supports SS7 signaling in the same way that PBX's software does. We all know that SS7 has four levels, in the 4th level we see that YATE supports two protocols; the ISUP used in SP and the SCCP used in the STP. Those protocols can work together and perform a batter network topology. It means that YATE is not only a server, but rather it can also act as a router.

This study examines whether there is a way to use a SS7 network inside an IP networks with the use of open source PBX YATE, which supports those protocols used to create the network. The project was implemented and tested in TDM/E1 and Ethernet card, and traffic was monitored with monitoring software.

The study consists of using three servers that have two different open sources PBX's, two of them are YATE and the other is the Asterisk. The study includes implementing a communication between them by the mean of SIP trunk, SIGTRAN and SIP-T. The experiment has delivered a successful communication after conducting a configuration for the files on the multiple sides.

Difficulty was encountered for the transferring of the voice in the second experiment, since the MGCP module in YATE does not support the TDM/E1 connection to carry the voice. After achieving a signaling connection through the TDM/E1 link, there was a solution in the configuration files by adding additional nodes in STP server helped the MGCP module to carry the voice.

The study has showed that YATE is a developing system that may offer an opportunity that would make it possible to be used instead of the traditional PBX.

6. References

[1] Russell, T.. Signaling System #7. McGraw-Hill, New York 2002 , ISBN 0-07-146879-X

[2] Bosse, J.G.. Signaling in telecommunication networks. VAN BOSSE, John G., DEVETAK , Fabrizio U, Ltd. En-gland 2002 , ISBN 0-471-66288-7.

[3] Freeman, R.L.. Telecommunication system engineering. John Wiley & Sons, 2004, ISBN 0-471-45133-9.

[4] YATE - Main – Documentation. [online], [cit. 2010-10-11]. Available in http://www.yate.null.ro.

[5] Wikipedia. Stream Control Transmission Protocol [online]. 2009 [cit. 2009-05-06]. Dostupný z WWW: < http://en.wikipedia.org/wiki/SCTP >.

[6] IEC. SS7 over IP Signaling Transport & SCTP [online]. <http://www.iec.org>

[7] ITU-T Q.700: Specification of Siganlling System No.7, 03/2003

[8] ITU-T Q.701: Functional Description of MTP, 03/1993

[9] ITU-T Q.714: SCCP Procedures, 07/1994

[10] ITU-T Q.774: Transaction Capabalities Procedures, 06/1997

[11] TS 09.02: MAP Specification, 3GPP, 1997

[12] ITU-T Q.767: ISDN Procedures, 03/1993

[13] RFC 2960: Stream Control Transmission Protocol, 10/200

[14] http://www.protocols.com/pbook/sigtran.htm

[15] http://www.rfc-editor.org/bcp/bcp63.txt