

**VŠB – TECHNICAL UNIVERSITY OF OSTRAVA**  
**FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE**  
***DEPARTMENT OF TELECOMMUNICATIONS***

# **M A S T E R   T H E S I S**

***MODELLING AND SIMULATION OF SIP AND IAX SESSIONS***  
***(MODELOVÁNÍ A SIMULACE SIP A IAX RELACÍ)***

2016

Bc. PAMBO Arão Minamau

## Diploma Thesis Assignment

Student: **Bc. Arao Minamau Pambo**  
Study Programme: N2647 Information and Communication Technology  
Study Branch: 2601T013 Telecommunication Technology  
Title: **Modelování a simulace SIP a IAX relací**  
**Modelling and Simulation of SIP and IAX Sessions**

The thesis language: English

### Description:

IAX protocol is a native communications protocol of Asterisk platform, data from multiple sessions are merged into a single stream of packets between two endpoints, reducing the IP overhead. Aim of this thesis is to investigate behaviour of IAX protocol, provide comparison to SIP and implement a simulation model for SIP and IAX traffic.

1. Introduction to Voice over IP and SIP protocol.
2. Asterisk and its native protocol IAX.
3. Building trunks between Asterisks based on SIP and IAX.
4. Measurements of traffics on SIP and IAX sessions.
5. Design and implementation of the simulation model for SIP and IAX traffic.
6. Conclusion.

### References:

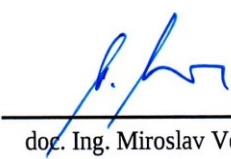
- [1] M. Voznak, J. Rozhon. Approach to stress tests in SIP environment based on marginal analysis. In *Telecommunication Systems*. Springer, March 2013, Volume 52, Issue 3, pp 1583-1593.  
[2] M. Boucadair. *Inter-Asterisk Exchange (IAX): Deployment Scenarios in SIP-Enabled Networks*. Wiley, 2009, ISBN 978-0-470-77072-6.

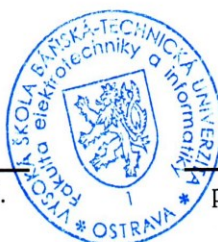
Extent and terms of a thesis are specified in directions for its elaboration that are opened to the public on the web sites of the faculty.


Supervisor: **doc. Ing. Miroslav Vozňák, Ph.D.**

Date of issue: 01.09.2014

Date of submission: 29.04.2016

  
doc. Ing. Miroslav Vozňák, Ph.D.  
Head of Department



  
prof. RNDr. Václav Snášel, CSc.  
Dean of Faculty

# **DECLARATION**

*I hereby declare that I have worked this thesis on my own. I have referenced all the literary sources and publications that I have used.*

*In Ostrava, on June 29<sup>th</sup> 2016.*

A handwritten signature in blue ink, appearing to read 'Petr Čech', written over a dotted line.

*Student's signature*

## **ACKNOWLEDGENTS**

I would like to thank my supervisor Assoc. Prof. Doc. Ing. Miroslav Vozňák, PhD. for his guidance, comments, hours of fruitful discussions and also for drawing my attention to the VoIP technology, mainly to the SIP and IAX protocols.

## **ABSTRACT**

My thesis is focused on simulating a functioning model of SIP and IAX and compare these two VoIP protocols. This is done by implementing an Asterisk server onto two virtual machines with Ubuntu operating system where I build a trunk system for both protocol, tested it by calling the peers in both directions, captured the traffic passing through and analyzed it with Wireshark. The acquired data is then implemented and presented on a chart form for a better view and comparison of the two parallel protocols.

***KEY WORDS: VOIP, SIP, IAX, TRUNKING, ASTERISK, RTP, SRTP, UDP, BANDWIDTH***

## **ABSTRAKT**

Moje práce je zaměřena na simulaci funkčnosti modelu SIP a IAX a porovnání těchto dvou VoIP protokolů. To je provedeno zavedením Asteriskem serveru na dva virtuální počítače s operačním systémem Ubuntu, kde je vybudován trunk systém pro oba protokoly a to tak, že spojuje volající v obou směrech, zachycuje průchod a analyzuje pomocí Wireshark. Získaná data jsou pak použita a prezentována ve formě grafů pro lepší přehlednost a srovnání obou paralelních protokolů.

***KLÍČOVÁ SLOVA: VOIP, SIP, IAX, TRUNKING, ASTERISK, RTP, SRTP, UDP, BANDWIDTH***

## CONTENTS

<b>1 INTRODUCTION TO VOICE OVER IP AND SIP PROTOCOL.....</b>	<b>9</b>
1.1 Advantages of VoIP.....	11
1.2 Drawbacks of VoIP.....	13
1.3 Understanding the Session Initiation Protocol – SIP.....	16
1.3.1 SIP Security Abilities.....	17
1.3.2 SIP definitions.....	18
<b>2. ASTERISK AND ITS NATIVE PROTOCOL IAX.....</b>	<b>25</b>
2.1 Asterisk Definition.....	25
2.1.1 Asterisk architecture.....	25
2.1.2 Modules.....	26
2.1.3 Types of Modules in Asterisk.....	27
2.1.3.1 Applications Module.....	27
2.1.3.2 Bridging Modules.....	27
2.1.3.3 Call Detail Recording – CDR Modules.....	27
2.1.3.4 Channel Event Logging Modules – CEL.....	28
2.1.3.5 Channel Drivers.....	28
2.1.3.6 Codec Translators.....	28
2.1.3.7 Format Interpreters.....	28
2.1.3.8 Dialplan Functions.....	29
2.1.3.9 PBX Modules.....	29
2.1.3.10 Resource Modules.....	29
2.2.1 Configuration of backends.....	29
2.2.2 Timing interfaces.....	29
2.2.3 Calendar integration.....	29
2.2.4 RTP Implementations.....	30
2.2.5 Configuration Files.....	30
2.2.6 Logging Files.....	30
2.2.7 Asterisk Dial Plan.....	31
2.3 The Hardware for Asterisk.....	32
2.3.1 Asterisk Installation.....	32

2.4	Inter Asterisk eXchange – IAX.....	34
2.4.1	IAX: Towards Lightweight Telephony Architectures.....	36
2.4.2	Solving VoIP Problems with IAX.....	38
2.4.3	IAX Ensures Reliability.....	39
2.4.4	Registering IAX.....	39
2.4.5	Transportation of Media Streams in IAX.....	40
2.4.6	IAX Codec Negotiation.....	40
2.4.7	IAX and Security Related Issues.....	40
2.4.8	Advantages of IAX.....	40
2.4.9	IAX Full Frames.....	44
2.4.10	IAX Communication Call Flow.....	47
2.5	Comparison of Main Differences of SIP vs IAX Protocols.....	48
<b>3.</b>	<b>BUILDING TRUNKS BETWEEN ASTERISK BASED ON SIP AND IAX.....</b>	<b>50</b>
<b>4.</b>	<b>MEASUREMENT OF TRAFFICS ON SIP AND IAX SESSIONS.....</b>	<b>57</b>
4.1	SIP Installation .....	57
<b>5.</b>	<b>DESIGN AND IMPLEMENTATION OF THE SIMULATION MODEL FOR SIP AND IAX TRAFFIC.....</b>	<b>63</b>
5.1	SIP Model.....	63
5.2	IAX Model.....	63
<b>6.</b>	<b>CONCLUSION.....</b>	<b>65</b>
	<b>LITERATURE.....</b>	<b>66</b>
	<b>ACRONYMS.....</b>	<b>68</b>
	<b>LIST OF TABLES.....</b>	<b>70</b>
	<b>LIST OF FIGURES.....</b>	<b>71</b>
	<b>LIST OF GRAPHS.....</b>	<b>72</b>
	<b>APPENDIX.....</b>	<b>73</b>

## **INTRODUCTION**

Voice over Internet Protocol (VoIP) has been prevailing in the telecommunication world since its emergence in the late 90s as a new technology transporting multimedia over the IP network. It is very common today for people to make phone calls with IP phones or client software (e.g. Skype, iChat, and Google Talk) on their computer. Many telecommunications companies and other organizations have been moving their telephony infrastructure to their data networks, because it provides a cheaper and clearer alternative to traditional public service telephone network (PSTN) phone lines.

Even though the VoIP service is getting popular, its technology is still developing. Its deployment throughout the world is much faster than at the time of the traditional telecommunication system, though it often lacks compatibility and scalability with existing systems. Nevertheless, VoIP has already conquered a significant pie of the telephony market, given the fiscal savings and flexibility that it can provide. In this thesis, I will focus on the SIP and IAX as two major protocols of this technology.



## **1. INTRODUCTION TO VOICE OVER IP AND SIP PROTOCOL**

Voice over IP (VoIP) is a technology for the delivery of voice communications and multimedia sessions over an IP (Internet Protocol) network, such as the Internet [5]. The VoIP technology allows many benefits for customers and communication services providers. In fact, the VoIP approach allows the reduction of calls and communication infrastructure costs, helps the provision of new communication services (instant messages, video calls, images transfer, etc.), ensures users and services mobility, allows the integration and collaboration with other applications (email, web browser, instant messenger, social networking applications), and provides an online tracking and managing system.

Nowadays, Voice over Internet Protocol (VoIP) constitutes a privileged field of service innovation. One benefit of the VoIP technology is that it may be deployed using a centralized or a distributed architecture. The majority of today's VoIP systems are deployed using the client-server centralized architecture. One of the most efficient approaches used in the deployment of centralized VoIP systems is based on the use of IAX (Inter-Asterisk Exchange), which is an open-source signaling and/or data exchange protocol.

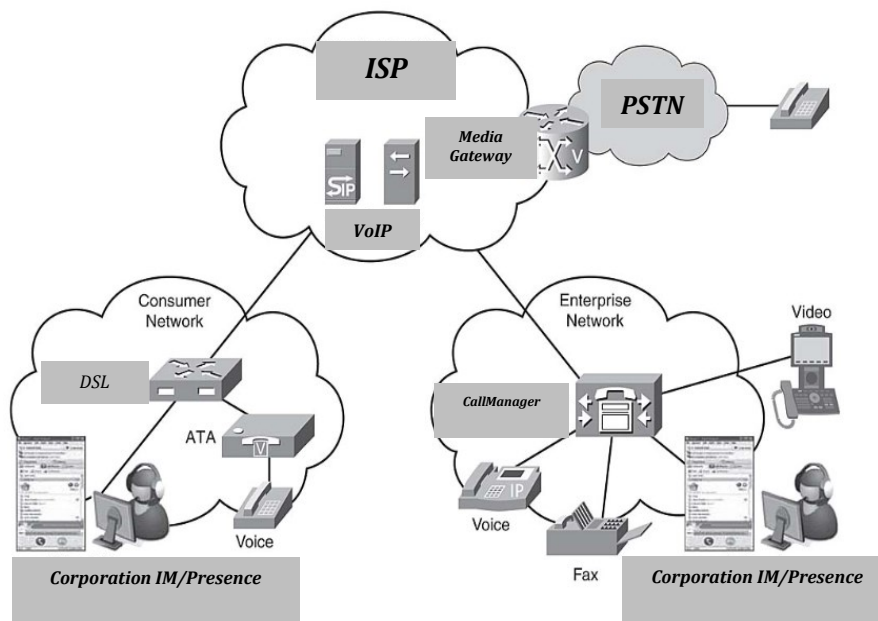
Even though they are currently and widely used, client-server VoIP systems suffer from many weaknesses such as the presence of single points of failure, an inefficient resources management, and system non-scalability. In order to cope with the development of scalable and reliable VoIP systems, the development community tends towards the deployment of the VoIP service using a peer-to-peer distributed architecture. The goal of this project is to investigate the behavior of IAX protocol, provide comparison to SIP and implement a simulation model for SIP and IAX traffics.

Since corporation started using digital voice coding, such as Integrated Service Digital Network – ISDN, they have thought about convergence between telephony and IT environment in order to transmit data, voice and video applications using one and the same medium. Unfortunately, each of these applications has different needs.

Data transmission requires different line bandwidths and does not care for reliability of connection, and, on the opposite, voice and video transmissions need a constant bandwidth and guaranteed time of delivery.

The structures of available networks are different and only meet the needs of the application they have been created for. In data networks, everyone can use the available bandwidth to the maximum extent, which means that the line capacity is exploited efficiently. On the contrary, telephone network reserves a channel per call regardless of data transmission (e.g. one party is speaking only during a standard call, there is no data transmission from the non-speaking party but the channel remains busy).

A lot of voice and video transmission technologies using real-time IP networks (Internet), generally called VoIP (Voice over IP), have been developed as an alternative to the standard circuit-switching telephone network. As a result of natural selection, only two of them are now implemented in telecommunications, which improves interaction and compatibility of products from different companies. These two technologies are H.323 and SIP (RFC 3261) [9].



*Fig. 1 – VoIP service architecture with many different types of services integrated [15]*

## 1.1 – ADVANTAGES OF VoIP

The reason for the prevalence of VOIP is that it gives significant benefits compared to legacy phone systems. The key benefits are as follows [6]:

- ❖ **Cost savings** – The most attractive feature of VoIP is its cost effective potential. When we move away from public switched telephone networks, long distance phone calls become inexpensive. Instead of being processed across conventional commercial telecommunications line configurations, voice traffic travels on the Internet or over private data network lines.

For the companies, VoIP reduces cost for equipment, lines renting, manpower, and maintenance. All of an organization's voice and data traffic is integrated into one physical network, bypassing the need for separate Private Branch eXchange – PBX tie lines. Although there is a significant initial setup cost, significant net savings can result from managing only one network and not needing to sustain a legacy telephony system in an increasingly digital and data-centered world. Also, the network administrator's burden may be lessened as they can now focus on a single network. There is no longer a need for several teams to manage a data network and another to manage a voice network. For consumers, VoIP reduces the charge of subscription or usage, especially for long distance and international calls.

- ❖ **Rich media service** – The legacy phone system mainly provides voice and fax services even though limited video service is also possible. However, users' demands are much higher than that, as shown in today's rich media communications through the Internet. People check out friends' presence (such as online, offline, busy status), send instant messages, make voice or video calls, transfer images, etc. VoIP technology makes rich media service possible, integrating with other protocols and applications.

Rich media service does not only provide multiple options of media to users, but also creates new markets in the communications industry, such as VoIP service in mobile phones.

- ❖ **Phone portability** – The legacy phone system assigns a phone number with a dedicated line, this means that technically a user cannot move the home phone to another place if they still want to use the very phone number. It is a common

issue to call the phone company and ask for a phone number update when moving to a new house. However, VoIP provides number mobility: The phone device can use the same number virtually everywhere as long as it has proper IP connectivity. Many businesspeople today care along their IP phones or softphones when traveling and use the same numbers everywhere.

- ❖ ***Service mobility*** – The context of mobility here comprehends service availability for the customer at any place they move. Wherever the phone goes, the same services could be available, such as call features, voicemail access, call logs, security features, service policy, etc.
- ❖ ***Integration and collaboration with other applications*** – VoIP protocols (such as Session Initiation Protocol – SIP, H.323) (RFC 4123) run on the application layer and are able to integrate or collaborate with other applications such as email, web browser, instant messenger, social networking applications, and more. The integration and collaboration create synergy and provide valuable services to the users. Typical examples are voicemail delivery via email, click-to-call – C2C service on a website, voice call button on an email, presence information on a contact list, etc.
- ❖ ***User control interface*** – Most VoIP service providers provide a user control interface, typically a web GUI, to their customers so that they can change features, options, and services dynamically. For instance, the users log in to the web GUI and change call forwarding number, speed dial, presence information (online/offline status), music-on-hold option, anonymous call block and more.
- ❖ ***No geographical boundary*** – The VoIP service area becomes virtualized without geographical limits. That is, the area code or country code is no longer bound to a specific location. For instance, a given customer could live in Prague and subscribe to a USA phone number, which makes it possible that all calls to the USA become domestic calls (cheaper) even though the customer lives in the Czech Republic.
- ❖ ***Rich features*** – VoIP provides rich features like click-to-call on a web page, Find-Me-Follow-Me (FMFM), selective call forwarding, personalized ring tones (or ring back tone), simultaneous rings on multiple phones, selective area or country code, etc.

These significant benefits are behind the prevalence of VoIP compared to legacy phone systems. In fact, most service providers have already started or at least have planned to migrate their PSTN (Public Switched Telephone Network) infrastructure to an IP-based one, thus showing a signal of future telephony systems to be totally IP based and the Circuit Switched (core voice system) would be no longer needed.

One benefit of the VoIP technology is that it may be deployed using either a centralized or a distributed architecture. The majority of today's VoIP systems are deployed using a client-server centralized architecture, which is a system that relies on the use of a set of interconnected central servers responsible for the registration of users, and the management of VoIP sessions between these [5].

Different signaling protocols have been proposed for the deployment of client-server VoIP systems such as the ITU-T H.323 Session Initiation Protocol – SIP (RFC 3261), and the Inter-Asterisk eXchange – IAX (RFC 5456).

The current VoIP systems do mainly rely on the use of SIP and IAX signaling protocols. Even though it was proposed for security and flexibility purposes, SIP (RFC 3261) suffers from many weaknesses. In fact, nowadays SIP becomes more and more complex due to the incremental modification of SIP specifications in order to improve the protocol adaptability. Moreover, SIP suffers from the difficulties of crossing NAT (Network Address Translation) and firewall boxes. IAX (RFC 5456) protocol is considered as a possible candidate to solve SIP problems as it is a robust protocol which supports NAT and firewalls traversal since no IP addresses are enclosed in IAX (RFC 5456) signaling messages.

Moreover, IAX (RFC 5456) allows signaling and data traffic exchange in contrast with SIP which is limited to the signaling task.

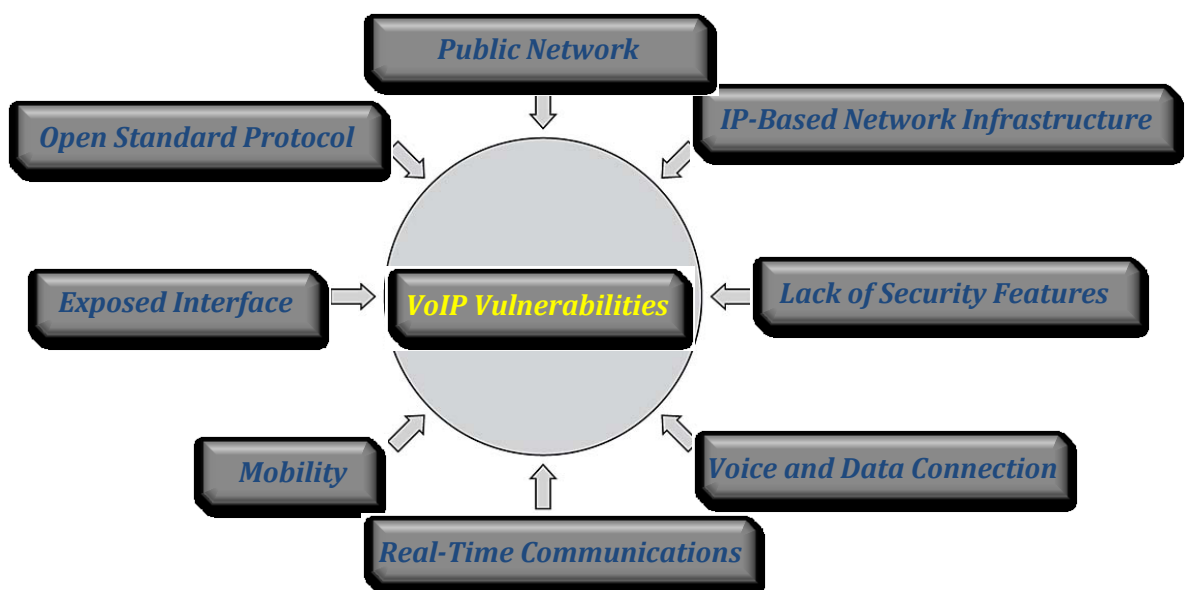
## **1.2 – DRAWBACKS OF VoIP**

The benefits of VoIP do not come free of charge. There are significant disadvantages for using VoIP, and I will name some:

- ❖ ***Complicated service and network architecture*** – Integrated rich media services (such as voice, video, IM, presence, and fax) make it difficult to design the service and network architecture because many different types of devices for each service are involved, as well as different protocols and characteristics of each media. Rich features such as click-to-call and find me-follow me (C2C, FMFM) also make the architecture more complicated because many different applications (such as web and email) and platforms are involved. This complication requires extra time and resources when designing, testing, and deploying. It also causes several errors and makes it harder for troubleshooting and isolate them.
- ❖ ***Interoperability issues between different protocols, applications, or products*** – There are multiple VoIP protocols (such as SIP, H.323, Media Gateway Control Protocol – MGCP), and product companies which choose whatever they like when developing products, which means there are always interoperability issues between the products that use different protocols. Even between the products using the same protocol, interoperability issues still come up because of different ways of implementation, different versions (extensions), or different feature sets. Therefore, it is common for VoIP service providers to spend a significant amount of time and resources for testing interoperability and resolving related issues.
- ❖ ***Quality of service (QoS)*** – Voice and video streams flow over an IP network as real-time packets, passing through multiple networks and devices (such as switches, routers, firewalls, and media gateways). Therefore, ensuring QoS is very difficult and costs lots of time and resources to meet the user's expectations. The main factors in QoS are packet loss, delay, and jitter (*packet delay variation*).
- ❖ ***Power outages*** – Legacy home phones continue to work even during a power outage because the phone line supplies 48 volts constantly. However, VoIP phones use regular data network lines that do not provide power in most cases, which means we cannot use VoIP phones during power outages. Yet, there are inline power solutions (such as Power over Ethernet), but these are mainly for enterprise environments.

- ❖ **Security issues** – In a legacy phone system, the security issue is mainly intercepting conversations that require physical access to phone lines or compromise of the office PBX. In VoIP based on open or public networks, security issues are much more than that. Between a caller and callee, many elements (such as IP phones, access devices, media gateways, proxy servers, and protocols) are involved in setting up the call and transferring the media. Each element has vulnerable factors that are targets for attackers.
- ❖ **Legal issues (lawful interception)** – Legal wiretapping in VoIP, also called lawful interception (LI), is much more complicated than that in legacy phone systems, because of the complexity of VoIP service architecture.

Basically, these vulnerabilities are derived from the characteristics of VoIP that are shown in the figure below:



*Fig. 2 - Possible sources of security breaches in VoIP systems*

Among these disadvantages, the security issues are becoming more serious because traditional security devices such as firewalls and Intrusion Detection Systems – IDS and protocols encryption cannot protect VoIP services or networks from recent intelligent threats.

### 1.3 - UNDERSTANDING THE SESSION INITIATION PROTOCOL – SIP

Session Initiation Protocol (SIP) (RFC 3261) is a *signaling protocol used to create, manage and terminate sessions in an IP based network*. A session could be a simple two-way telephone call or it could be a collaborative multi-media conference session. This makes possible to implement services like voice-enriched *e-commerce*, web page click-to-dial or Instant Messaging with buddy lists in an IP based environment [8].

SIP is an application layer control protocol that can establish, modify, and terminate multimedia sessions (conferences) such as Internet telephony calls.

SIP can also invite participants to already existing sessions, such as multicast conferences. Media can be added to and/or removed from an existing session [13]. SIP transparently supports name mapping and redirection services, which supports personal mobility allowing users to maintain a single externally visible identifier regardless of their network location.

SIP supports five facets of establishing and terminating multimedia communications:

- ❖ ***User location***: determination of the end system to be used for communication;
- ❖ ***User availability***: determination of the willingness of the callee party to engage in communications;
- ❖ ***User capabilities***: determination of the media and media parameters to be used;
- ❖ ***Session setup***: "*ringing*", establishment of session parameters at both callee and calling party;
- ❖ ***Session management***: including transfer and termination of sessions, modifying session parameters, and invoking services.

SIP is not a vertically integrated communications system per se, rather it is a component that can be used with other Internet Engineering Task Force – IETF protocols to build a complete multimedia architecture. Typically, these architectures will include protocols such as the Real-time Transport Protocol – RTP (RFC 1889) for transporting real-time data and providing quality of services – QoS feedback, the Real-Time streaming protocol – RTSP (RFC 2326) for controlling delivery of streaming media, the Media Gateway Control Protocol – MEGACO (RFC 3015) for controlling gateways to the Public Switched Telephone Network (PSTN), and the Session Description Protocol – SDP (RFC 2327) for describing multimedia



sessions [10].

Therefore, SIP should be used in conjunction with other protocols in order to provide complete services to the users. However, the basic functionality and operation of SIP does not depend on any of these protocols.

### 1.3.1 – SIP SECURITY ABILITIES

The SIP protocol describes several security features, the main security features of the SIP protocol are: *message authentication*, *message encryption*, *media encryption*, *transport layer security* and *network layer security*. Only message authentication is ensured by SIP protocol, the other abilities are allowed by other security protocols such as S/MIME, SRTP/SRTCP (RFC 3711), TLS (RFC 5246), and IPSec (RFC 6071) [5]. In the following, a brief presentation of the main security features of the SIP signaling protocol.

**Message Authentication:** SIP (RFC 3261) ensures the authentication of signaling messages (REGISTER, INVITE, and BYE) to avoid registration hijacking attacks and prevent unauthorized calls and denial of services – DoS or annoyance attacks.

**Message Encryption:** SIP relies on the S/MIME (Secure/Multipurpose Internet Mail Extensions) (RFC 5751) protocol to encrypt the headers of the signaling messages (except the “Via”, and “Route” headers) which helps end-to-end confidentiality, integrity, and authentication between participants. S/MIME (RFC 5751) provides the flexibility for more granular protection of header information in SIP messages as it allows a selectively protection of SIP message fields.

**Media encryption:** Secure RTP – SRTP (RFC 3711) protocol ensures the encryption of media packets encryption which helps the guarantee of the confidentiality and integrity of exchanged media.

**Transport Layer Security – TLS:** TLS (RFC 5246) protocol is used to provide a transport-layer security of SIP messages (requests, responses). Actually TLS ensures the encryption of entire SIP requests and responses which ensures the confidentiality and integrity of messages.

**Network Layer Security:** SIP (RFC 3261) relies on the use of IPSec (RFC 6071) at the network layer which enhances the security of IP network communications by encrypting and authenticating data. IPSec (RFC 6071) is very useful to provide security between SIP entities, especially between a user agent (UA) and a proxy server.

### 1.3.2 – SIP DEFINITIONS

The following terms have special significance for SIP [4].

- ❖ **Address-of-Record:** An address-of-record – AOR is a SIP (RFC 3261) or SIPs Uniform Resource Identifier– URI (RFC 5630) that points to a domain with a location service that can map the URI to another URI where the user might be available.

Typically, the location service is populated through registrations. An AOR is frequently thought of as the "public address" of the user.

- ❖ **Back-to-Back User Agent:** A back-to-back user agent – B2BUA is a logical entity that receives a request and processes it as a user agent server – UAS. In order to determine how the request should be answered, it acts as a user agent client – UAC and generates requests. Unlike a proxy server, it maintains dialog state and must participate in all requests sent on the dialogs it has established. Since it is a concatenation of a UAC and UAS, no explicit definitions are needed for its behavior.
- ❖ **Call:** A call is an informal term that refers to some communication between peers, generally set up for the purposes of a multimedia conversation.
- ❖ **Call Stateful:** A proxy is call stateful if it retains state for a dialog from the initiating *INVITE* to the terminating *BYE* request. A call stateful proxy is always transaction stateful, but the converse is not necessarily true.
- ❖ **Client:** A client is any network element that sends SIP requests and receives SIP responses. Clients may or may not interact directly with a human user. User agent clients and proxies are clients.

- ❖ **Core:** Core designates the functions specific to a particular type of SIP entity, i.e., specific to either a stateful or stateless proxy, a user agent or registrar. All cores, except those for the stateless proxy, are transaction users.
- ❖ **Dialog:** A dialog is a peer-to-peer SIP (RFC 3261) relationship between two UAs that persists for some time. A dialog is established by SIP messages, such as a \_2XXX response to an INVITE request. A dialog is identified by a call identifier, local tag, and a remote tag. A dialog was formerly known as a call leg in RFC 2543.
- ❖ **Downstream:** A direction of message forwarding within a transaction that refers to the direction that requests flow from the user agent client to user agent server.
- ❖ **Final Response:** A response that terminates a SIP transaction, as opposed to a provisional response that does not. All \_2XX, \_3XX, \_4XX, \_5XX and \_6XX responses are final.
- ❖ **Header:** A header is a component of a SIP message that conveys information about the message. It is structured as a sequence of header fields.
- ❖ **Header Field:** A header field is a component of the SIP message header. A header field can appear as one or more header field rows. Header field rows consist of a header field name and zero or more header field values. Multiple header field values on a given header field row are separated by commas. Some header fields can only have a single header field value, and as a result, always appear as a single header field row.
- ❖ **Header Field Value:** A header field value is a single value; a header field consists of zero or more header field values.
- ❖ **Home Domain:** The domain providing service to a SIP (RFC 3261) user. Typically, this is the domain present in the URI in the address-of-record of a registration.
- ❖ **Informational Response:** Same as a provisional response.
- ❖ **Initiator, Calling Party, Caller:** The party initiating a session (and dialog) with an INVITE request. A caller retains this role from the time it sends the initial INVITE that established a dialog until the termination of that dialog.
- ❖ **Invitation:** An INVITE request.

- ❖ ***Invitee, Invited User, Called Party, Callee:*** The party that receives an INVITE request for the purpose of establishing a new session. A callee retains this role from the time it receives the INVITE until the termination of the dialog established by that INVITE.
- ❖ ***Location Service:*** A location service is used by a SIP (RFC 3261) redirect or proxy server to obtain information about a caller's possible locations. It contains a list of bindings of address-of-record keys to zero or more contact addresses. The bindings can be created and removed in many ways; this specification defines a REGISTER method that updates the bindings.
- ❖ ***Loop:*** A request that arrives at a proxy, is forwarded, and later arrives back at the same proxy. When it arrives the second time, its Request-URI is identical to the first time, and other header fields that affect proxy operation are unchanged, so that the proxy would make the same processing decision on the request it made the first time. Looped requests are errors, and the procedures for detecting them and handling them are described by the protocol.
- ❖ ***Loose Routing:*** A proxy is said to be loose routing if it follows the procedures defined in this specification for processing of the Route header field. These procedures separate the destination of the request which is present in the Request-URI from the set of proxies that need to be visited along the way also present in the Route header field. A proxy compliant to these mechanisms is also known as a loose router.
- ❖ ***Message:*** Data sent between SIP (RFC 3261) elements as part of the protocol, as these messages are either requests or responses.
- ❖ ***Method:*** The method is the primary function that a request is meant to invoke on a server. The method is carried in the request message itself. Example methods are *INVITE* and *BYE*.
- ❖ ***Outbound Proxy:*** A proxy that receives requests from a client, even though it may not be the server resolved by the Request-URI. Typically, a UA is manually configured with an outbound proxy, or can learn about one through auto-configuration protocols.
- ❖ ***Parallel Search:*** In a parallel search, a proxy issues several requests to possible user locations upon receiving an incoming request. Rather than

issuing one request and then waiting for the final response before issuing the next request as in a sequential search, a parallel search issues requests without waiting for the result of previous requests.

- ❖ **Provisional Response:** A response used by the server to indicate progress, but that does not terminate a SIP transaction. \_1XX responses are provisional, other responses are considered final.
- ❖ **Proxy, Proxy Server:** An intermediary entity that acts as both a server and a client for the purpose of making requests on behalf of other clients. A proxy server primarily plays the role of routing, which means its job is to ensure that a request is sent to another entity "closer" to the targeted user. Proxies are also useful for enforcing policy, for example making sure a user is allowed to make a given call. A proxy interprets, if necessary, rewrites specific parts of a request message before forwarding it.
- ❖ **Recursion:** Client recurses on a \_3XX response when it generates a new request to one or more of the URIs in the Contact header field in the response.  
**Redirect Server:** A redirect server is a user agent server that generates \_3XX responses to requests it receives, directing the client to contact an alternate set of URIs.
- ❖ **Registrar:** A registrar is a server that accepts REGISTER requests and places the information it receives in those requests into the location service for the domain it handles.
- ❖ **Regular Transaction:** A regular transaction is any transaction with a method other than INVITE, ACK, or CANCEL.
- ❖ **Request:** A SIP message sent from a client to a server, for the purpose of invoking a particular operation.
- ❖ **Response:** A SIP message sent from a server to a client, for indicating the status of a request sent from the client to the server.
- ❖ **Ringback:** Ringback is the signaling tone produced by the calling party's application indicating that a called party is being alerted (ringing).
- ❖ **Route Set:** A route set is a collection of ordered SIP (RFC 3261) or SIPS URI (RFC 5630) which represents a list of proxies that must be traversed when sending a particular request. A route set can be learned, through headers like

Record-Route, or it can be configured.

- ❖ **Server:** A server is a network element that receives requests in order to service them and sends back responses to those requests. Examples of servers are proxies, user agent servers, redirect servers, and registrars.
- ❖ **Sequential Search:** In a sequential search, a proxy server attempts each contact address in sequence, proceeding to the next one only after the previous has generated a final response. A \_2XX or \_6XX class final response always terminates a sequential search.
- ❖ **Session:** From the SDP (RFC 4566) specification: "A multimedia session is a set of multimedia senders and receivers and the data streams flowing from senders to receivers. A multimedia conference is an example of a multimedia session." (RFC 2327) (A session as defined for SDP (RFC 4566) can comprise one or more RTP (RFC 3550) sessions.) As defined, a callee can be invited several times, by different calls, to the same session. If SDP (RFC 4566) is used, a session is defined by the concatenation of the SDP (RFC 4566) user name, session ID, network type, address type, and address elements in the origin field.
- ❖ **SIP Transaction:** A SIP (RFC 3261) transaction occurs between a client and a server and comprises all messages from the first request sent from the client to the server up to a final (non \_1XX) response sent from the server to the client. If the request is INVITE and the final response is a non \_2xx, the transaction also includes an ACK to the response. The ACK for a \_2xx response to an INVITE request is a separate transaction.
- ❖ **Spiral:** A spiral is a SIP request that is routed to a proxy, forwarded onwards, and arrives once again at that proxy, but this time differs in a way that will result in a different processing decision than the original request. Typically, this means that the request's Request-URI differs from its previous arrival. A spiral is not an error condition, unlike a loop; a typical cause for this is call forwarding. For instance, a user calls Samuel@vsb.cz. The vsb.cz proxy forwards it to Samuel's PC, which in turn, forwards it to Beth@vsb.cz. This request is proxied back to the vsb.cz proxy. However, this is not a loop. Since the request is targeted at a different user, it is considered a spiral, and is a valid

condition.

- ❖ **Stateful Proxy:** A logical entity that maintains the client and server transaction state machines defined by this specification during the processing of a request, also known as a transaction stateful proxy. A (transaction) stateful proxy is not the same as a call stateful proxy.
- ❖ **Stateless Proxy:** A logical entity that does not maintain the client or server transaction state machines defined in this specification when it processes requests. A stateless proxy forwards every request it receives downstream and every response it receives upstream.
- ❖ **Strict Routing:** A proxy is said to be strict routing if it follows the Route processing rules of RFC 2543 and many prior work in progress versions of this RFC. That rule caused proxies to destroy the contents of the Request-URI when a Route header field was present. Strict routing behavior is not used in this specification, in favor of a loose routing behavior. Proxies that perform strict routing are also known as strict routers.
- ❖ **Transaction User – TU:** The layer of protocol processing that resides above the transaction layer. Transaction users include the UAC core, UAS core, and proxy core.
- ❖ **Upstream:** A direction of message forwarding within a transaction that refers to the direction that responses flow from the user agent server back to the user agent client.
- ❖ **URL-encoded:** A character string encoded according to RFC 2396,
- ❖ **User Agent Client – UAC:** A user agent client is a logical entity that creates a new request, and then uses the client transaction state machinery to send it. The role of UAC lasts only for the duration of that transaction. In other words, if a piece of software initiates a request, it acts as a UAC for the duration of that transaction. If it receives a request later, it assumes the role of a user agent server for the processing of that transaction.
- ❖ **User Agent Server – UAS:** A user agent server is a logical entity that generates a response to a SIP request. The response accepts, rejects, or redirects the request. This role lasts only for the duration of that transaction. In other words, if a piece of software responds to a request, it acts as a UAS for the

duration of that transaction. If it generates a request later, it assumes the role of a user agent client for the processing of that transaction.

- ❖ **User Agent – UA:** A logical entity that can act as both a user agent client and user agent server.

SIP is limited to only the setup, modification and termination of sessions. It serves four major purposes:

- ❖ SIP allows for the establishment of user location, i.e. translating from a user's name to their current network address.
- ❖ SIP provides for feature negotiation so that all of the participants in a session can agree on the features to be supported among them.
- ❖ SIP is a mechanism for call management, e.g. adding, dropping, or transferring participants during an already established call.
- ❖ SIP allows for changing features of a session while it is in progress.



## **2. ASTERISK AND ITS NATIVE PROTOCOL IAX**

### **2.1 ASTERISK DEFINITION**

Asterisk is an open source framework for building communications applications. Asterisk turns an ordinary computer into a communications server. Asterisk powers IP PBX systems, VoIP gateways, conference servers and other custom solutions [12]. It is used by small businesses, large businesses, call centers, carriers and government agencies, worldwide. Asterisk is a multichannel PBX (Private Branch Exchange) which supports several VoIP (voice over IP) signaling protocols, such as SIP (RFC 3261), H.323 and IAX (RFC 5456).

#### **2.1.1 ASTERISK ARCHITECTURE**

Traditional Private Branch eXchange – PBX has a logical difference between stations (telephone sets) and trunks (resources that connect to the outside world). This means, for instance, that we cannot install an external gateway on a station port and route external calls to it without requiring the users to dial the extension number first. Also, the concept of an off-site resource (such as a reception desk) is much more difficult to implement on a traditional PBX, because the system will not allow external resources any access to internal features.

To be fair, many traditional PBX's do offer this type of functionality. However, most of the times it is poorly designed, limited in features, and requires complex proprietary software to be installed in the PBX (such as vendor's and/or specific protocol extensions).

Asterisk, on the other hand, does not have an internal concept of trunks or stations. In Asterisk, everything that comes into or goes out of the system passes through some sort of a channel [2].

Asterisk is very different from other, more traditional PBXs, in that the dialplan in Asterisk treats all incoming channels in essentially the same manner.

In a traditional PBX, there is a logical difference between stations (telephone sets) and trunks (resources that connect to the outside world). This means that an external gateway cannot be installed on a station port and route external calls to it without requiring the users to dial the

extension number first [2].

There are many different kinds of channels; however, the Asterisk dialplan handles all channels in a similar manner, which means that, for instance, an internal user can exist on the end of an external trunk (e.g. a cell phone) and be treated by the dialplan in exactly the same manner as that user would be if they were on an internal extension.

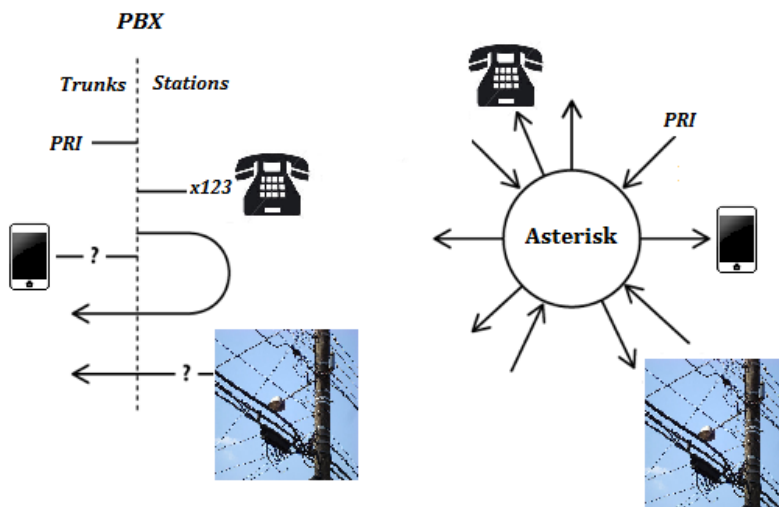


Fig. 3 – Asterisk vs PBX architecture

### 2.1.2 MODULES

Asterisk is built on *modules*. A module is a loadable component that provides a specific functionality, such as a channel driver (e.g. *chan\_sip.so*), or a resource that allows connection to an external technology (such as *func\_odbc.so*). Asterisk modules are loaded based on the */etc/asterisk/modules.conf* file. It is actually possible to start Asterisk without any modules at all, although in this state it will not be capable of doing anything, therefore, it is necessary to understand the modular nature of Asterisk in order to appreciate the architecture [2].

### **2.1.3 TYPES OF MODULES IN ASTERISK:**

- ❖ Applications module
- ❖ Bridging modules
- ❖ Call detail recording (CDR) modules
- ❖ Channel event logging (CEL) modules
- ❖ Channel drivers
- ❖ Codec translators
- ❖ Format interpreters
- ❖ Dialplan functions
- ❖ PBX modules
- ❖ Resource modules
- ❖ Add-on modules
- ❖ Test modules

#### **2.1.3.1– APPLICATIONS MODULE**

In this module, I will focus on the Dialplan applications which are used in *extensions.conf* to define the various actions that can be applied to a call. *The Dial()* application, for example, is responsible for making outgoing connections to external resources and is arguably the most important dialplan application.

#### **2.1.3.2 – BRIDGING MODULES**

Bridging modules perform the actual bridging of channels in the new bridging application programming interface – API. Each provides different features, which get used in different situations depending on what a bridge needs.

#### **2.1.3.3 – CALL DETAIL RECORDING – CDR MODULES**

CDR modules are meant to facilitate as many methods of storing call detail records as possible. We can store CDRs to a file (the default), a database, Remote Authentication Dial In User Service – RADIUS, or *syslog*.

### **2.1.3.4 – CHANNEL EVENT LOGGING MODULES**

Channel event logging – CEL, provides much more powerful control over reporting of call activity. By the same token, it requires more attention planning of the dialplan, and by no means will it work automatically.

### **2.1.3.5 – CHANNEL DRIVERS**

Without channel drivers, Asterisk would have no way to make calls. Each channel driver is specific to the protocol or channel type it supports (SIP, ISDN, etc.). The channel module acts as a gateway to the Asterisk core [2].

### **2.1.3.6 – CODEC TRANSLATORS**

The codec translators allow Asterisk to convert audio stream formats between calls. Let's say a call comes in on a PRI circuit (using G.711) and needs to be passed out a compressed SIP (RFC 3261) channel (e.g., using G.729, one of many codecs that SIP (RFC 3261) can handles), the relevant codec translator would perform the conversion.

Digium distributes some additional useful codec modules: `codec_g729`, `codec_silk`, `codec_siren7`, and `codec_siren14`. These codec modules are not open source for various reasons, a license must be purchased to use `codec_g729` and the others are free [2].

### **2.1.3.7 – FORMAT INTERPRETERS**

Format interpreters perform the function of codec translators, but they do their work on files rather than channels. If we have a recording on a menu that has been stored as GSM, a format interpreter would need to be used to play that recording to any channels not using the GSM codec. If we store a recording in several formats (such as WAV, GSM, etc.), Asterisk will determine the least costly format to use when a channel requires that given recording [2].

### **2.1.3.8 – DIALPLAN FUNCTIONS**

Dialplan functions are applications that provide many useful enhancements to things like string handling, time and date wrangling, and open database – ODBC connectivity.

### **2.1.3.9 – PBX MODULES**

The private branch exchange – PBX modules are peripheral modules that provide enhanced control and configuration mechanisms. For example, *pbx\_config* is the module that loads the traditional Asterisk dialplan.

### **2.1.3.10 – RESOURCE MODULES**

Resource modules integrate Asterisk with external resources. This group of modules has effectively turned into a catchall for things that do not fit in other categories, which are broken into some subgroups of modules that are related.

## **2.2.1 – CONFIGURATION OF BACKENDS**

Asterisk is configured using text files in */etc/asterisk* by default. These modules do offer an alternative configuration method.

### **2.2.2 – TIMING INTERFACES**

Some operations in Asterisk require a timing source. These modules provide timing to Asterisk from a variety of sources. Some cases where Asterisk needs a timing source include file playback and conferencing using the *ConfBridge()* application.

### **2.2.3 – CALENDAR INTEGRATION**

Asterisk includes some integration with calendar systems. You can read and write calendar information from the dialplan. You can also have calls originated based on calendar entries. The core calendar integration is provided by the *res\_calendar* module. The rest of the

modules provide the ability to connect to specific types of calendar servers.

#### 2.2.4 – REAL–TIME TRANSPORT PROTOCOL IMPLEMENTATIONS

The core of Asterisk does not include an RTP implementation. If we are using one of the VoIP channel drivers that uses RTP, we must also load the *res\_rtp\_asterisk* module. This RTP implementation could be replaced with a custom one if Asterisk was used on a system that included custom hardware such as a digital signal processing – DSP with more efficient RTP (RFC 3550) processing.

The multicast RTP implementation is only used by the *chan\_multicast\_rtp* channel driver, which is useful for paging a large number of phones [2].

NAME	PURPOSE	STATUS
<i>res_rtp_asterisk</i>	Provides RTP	Essential
<i>res_rtp_multicast</i>	Provides multicast RTP	New

*Table 1– RTP implementations*

#### 2.2.5 – CONFIGURATION FILES

The Asterisk configuration files include *extensions.conf*, *sip.conf*, *modules.conf*, and dozens of other files that define parameters for the various channels, resources, modules, and functions that may be in use.

#### 2.2.6 – LOGGING FILES

Asterisk is capable of generating several different kinds of log files. The */var/log/asterisk* folder is where things such as call detail records (CDRs), channel events from CEL, debug logs, queue logs, messages, errors, and other output are written.

This folder will be extremely important for any troubleshooting efforts you undertake.

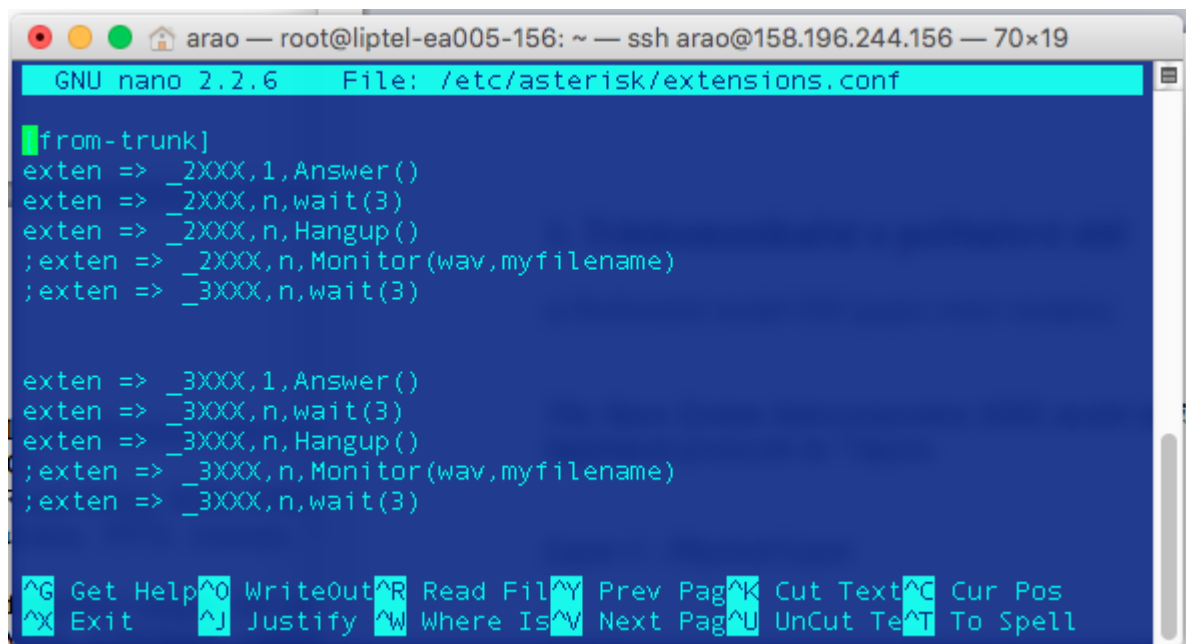
## 2.2.7 – ASTERISK DIAL PLAN

The dialplan is the heart of Asterisk. All channels that arrive in the system will be passed through the dialplan, which contains the call-flow script that determines how the inbound calls are handled.

A dialplan can be written in one of three ways:

- Using traditional Asterisk dialplan syntax in */etc/asterisk/extensions.conf*
- Using Asterisk Extension Logic (AEL) in */etc/asterisk/extensions.ael*
- Using Lua in */etc/asterisk/extensions.lua*

For the realization of my simulation, I used the traditional Asterisk dialplan syntax in */etc/asterisk/extensions.conf*



The screenshot shows a terminal window with a title bar indicating the user 'arao' is connected via SSH to 'liptel-ea005-156'. The terminal is running 'GNU nano 2.2.6' and editing the file '/etc/asterisk/extensions.conf'. The content of the file is as follows:

```
[from-trunk]
exten => _2XXX,1,Answer()
exten => _2XXX,n,wait(3)
exten => _2XXX,n,Hangup()
;exten => _2XXX,n,Monitor(wav,myfilename)
;exten => _3XXX,n,wait(3)

exten => _3XXX,1,Answer()
exten => _3XXX,n,wait(3)
exten => _3XXX,n,Hangup()
;exten => _3XXX,n,Monitor(wav,myfilename)
;exten => _3XXX,n,wait(3)
```

At the bottom of the terminal, there is a status bar with various keyboard shortcuts: ^G Get Help, ^O WriteOut, ^R Read File, ^Y Prev Page, ^K Cut Text, ^C Cur Pos, ^X Exit, ^J Justify, ^W Where Is, ^V Next Page, ^U UnCut Text, and ^T To Spell.

Fig. 4 – Asterisk dialplan syntax – *extensions.conf*

## 2.3 – THE HARDWARE FOR ASTERISK

Asterisk is capable of communicating with a vast number of different technologies. In general, these connections are made across a network connection; however, connections to more traditional telecom technologies, such as the PSTN, require specific hardware. Many companies produce this hardware, such as Digium (the sponsor, owner, and primary developer of Asterisk), Sangoma, Rhino, OpenVox, Pika, Voicetronix, Junghanns, Dialogic, Xorcom, beroNet, and many others [2].

### 2.3.1 – ASTERISK INSTALLATION

Asterisk installation depends very much on the operating system of the physical device as Asterisk relies so heavily on having priority access to the CPU, it is essential that we install Asterisk onto a server without any graphical interface, such as the X Windowing system (Gnome, KDE, etc.). Both RHEL and Ubuntu ship a GUI-free distribution are designed for server usage. In my case, I will stick with Ubuntu, as the Ubuntu Server.

I have created two virtual machines onto one of the Technical University of Ostrava's servers, with the following features, where each has been allocated an IP address to and an name, i.e. **VMIA** (IP: 158.196.244.146) and **VMIB** (IP: 158.196.244.156), where I have carried out all my practical work presented in this thesis.

Above are shown my Ubuntu version, system load capacity as well as the hardware description where my virtual machines have been running. There are many more commands (*lscpu*, *lshw -short*, *hwinfo*, *lspci*, *lsscsi*, *lsusb*, *inxi -Fx*, *lsblk*, etc) to check the hardware information of a system running on a Linux operating system.

Due to security reasons and a need of remote connection, I had been connecting on a secured shell (*ssh*) basis through the University's Virtual Private Network *EDUROAM* with the help of Cisco's AnnyConnect Secure Mobility Client to access the Server where my virtual machines are running.



As we already know, it is a must to be logged in as a super user to have more privileges of performing any activity on a core level such as installing/uninstalling anything in Linux. The Asterisk installation I had to get logged in as a root to be able to install it, set up my SIP trunk and IAX configurations and manage them remotely.

After logging in as a root, I issued the commands: *apt-get update*, then *apt-get upgrade* and at last *reboot*, with these commands I meant to update the system, then get the latest version of it and reboot the whole system to allow those downloaded applications to take effect.

To get Asterisk basic dependencies, logged in as a root I issued the command *apt-get install build-essential wget libssl-dev libncurses5-dev libnewt-dev libxml2-dev linux-headers-\$(uname -r) libsdl2-dev uuid-dev*

Downloading the source tarballs, these commands will get the current version of DAHDI, libpri and Asterisk:

```
cd /usr/src/  
wget http://downloads.asterisk.org/pub/telephony/dahdi-linux-complete/dahdi-linux-complete-current.tar.gz  
wget http://downloads.asterisk.org/pub/telephony/libpri/libpri-1.4-current.tar.gz  
wget http://downloads.asterisk.org/pub/telephony/asterisk/asterisk-11-current.tar.gz
```

Extracting the files from the tarballs:

```
tar zxvf dahdi-linux-complete  
tar zxvf libpri  
tar zxvf asterisk
```

Installing Digium Asterisk Hardware Device Interface – DAHDI:

```
cd /usr/src/dahdi-linux-complete  
make, make install, and make config
```

Installing Primary Rate Interface Library – libpri:

```
cd /usr/src/libpri
make, make install
```

Installing Asterisk:

I proceeded on by selecting options when the menu select command runs. Then selected “Save, Exit” and the installation will continue

```
cd /usr/src/asterisk
./configure
make menuselect
make
make install
make config
make samples
```

Starting Digium Asterisk Hardware Device Interface – DAHDI:

```
/etc/init.d/dahdi start
```

Starting Asterisk and connect to the Command Line Interface – CLI:

```
/etc/init.d/asterisk start
asterisk -vvvvvr
```

Verifying the installation by checking for the *DAHDI* and *libpri* versions on the Asterisk CLI:

```
*CLI> dahdi show version
DAHDI Version: 2.6.1 Echo Cancellor: HWEC
*CLI> pri show version
libpri version: 1.4.13
```

## **2.4 INTER ASTERISK EXCHANGE – IAX**

IAX is the Inter-Asterisk eXchange protocol, which facilitates VoIP connections between servers, and between servers and clients that also use the IAX protocol. IAX was created

through an open source methodology rather than through a traditional, standards-based methodology. It is an open protocol originally used by Asterisk, a dual-licensed open source and commercial Private Branch Exchange – PBX server from Digium. Independent IAX (RFC 5456) implementations may be open, proprietary, or licensed in anyway the author seems fit without royalty to the protocol creators [11].

IAX (RFC 5456) is a robust and full-featured, yet, simple protocol. In general, it is enough that to handle most common types of media streams. However, the protocol is highly optimized for VoIP calls where low-overhead and low-bandwidth consumption are priorities.

This pragmatic aspect makes IAX (RFC 5456) more efficient for VoIP than protocols that consider possibilities far beyond current needs and specify many more details than are strictly necessary to describe or transport a point-to-point call. Furthermore, because IAX (RFC 5456) is designed to be lightweight and VoIP-friendly, it consumes less bandwidth than more general approaches. IAX (RFC 5456) is a binary protocol, designed to reduce overhead, especially in regards to voice streams. Bandwidth efficiency, in some places sacrificed in exchange for bandwidth efficiency for individual voice calls. For instance, when transmitting a voice stream compressed to 8 kbps with a 20 ms packetization, each data packet consists of 20 bytes. IAX (RFC 5456) adds 20% overhead, 4 bytes, on the majority of voice packets while RTP (RFC 3550) adds 60% overhead with 12 additional bytes per voice packet [11].

In addition to efficiency, IAX's single static UDP port approach makes IAX traffic easy for network managers to shape, prioritize, and pass through firewalls. IAX's basic structure is that it multiplexes signaling and multiple media streams over a single UDP stream between two computers.

IAX (RFC 5456) also uses the same UDP port for both its signaling and media messages, and because all communications regarding a call are done over the same point-to-point path, NAT traversal is much simpler for IAX than for other commonly deployed protocols [11].

#### **2.4.1 IAX: TOWARDS LIGHTWEIGHT TELEPHONY ARCHITECTURES**

IAX stands as an interesting alternative besides classical protocols, deployed nowadays by service providers for their conversational service offerings (e.g. H.323 and SIP). IAX (RFC 5456) is a path-coupled protocol that is used for both signaling and media-control operations. Moreover, it provides interesting features such as management of signaling and media transfer, support for native provisioning functions and firmware maintenance. IAX (RFC 5456) is a simple protocol, which has the advantage of being IP version agnostic, leading to avoidance of network address translator – NAT traversal complications [3].

This issue represents a real asset, as NAT boxes are nowadays a tremendous challenge in conversational architectures and services and require additional patches, especially in home gateway equipment and the first service equipment (notably “Hosted NAT Traversal” facility). Moreover, this combined simplicity and completeness makes it germane to avoid resorting to a SIP Protocol Suite (SIP, SDP, RTP, RTCP, STUN, ICE, TURN,...) [3] .

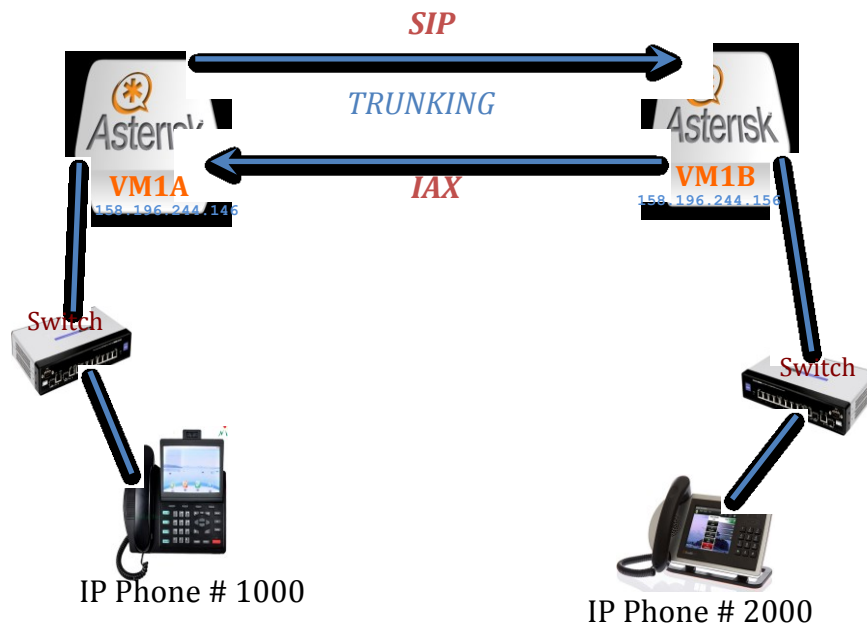
The IAX protocol offers significant features unavailable in other existent VoIP signaling protocols. Apart from its simplicity, the main characteristics of the IAX protocol are as below:

- ❖ IAX (RFC 5456) is transported over UDP (User Datagram Protocol) using a single port number. The default IAX port is 4569 (RFC 5456).
- ❖ The IAX (RFC 5456) registration philosophy is the same as of SIP (RFC 3261). An IAX (RFC 5456) registrant should contact a registrar server with specific messages. Contact information is then retrieved by the registrar server and stored in its system within a time period.
- ❖ IAX couples signaling and media paths. The decoupling is possible once the connection has been successfully established. This characteristic is denoted ‘path-coupled’ protocol, in contrast with the ‘path-decoupled’ approach assumed by SIP (RFC 3261).
- ❖ IAX (RFC 5456) does not require a new protocol for the exchange of media streams. It handles media streams itself. Different media types may be sent by IAX: voice, video, image, text, HTML and so on.
- ❖ IAX defines reliable and unreliable messages. IAX-unreliable messages are media

flows which are not acknowledged nor retransmitted if lost in the network. IAX reliability is ensured for control messages thanks to several IAX application identifiers maintained by the involved parties. Reliable messages should be acknowledged; if not, these messages are retransmitted.

- ❖ With IAX, NAT traversal is not an issue anymore as IP addresses do not need to be enclosed in IAX signaling messages.
- ❖ IAX defines a set of messages used to monitor the status of the network; these messages can be exchanged during or outside an active call.
- ❖ IAX offers the means to check whether a remote call participant is active or inactive.
- ❖ Native IP security methods can be deployed jointly with IAX as IAX allows exchange of shared keys. It may be used either with plain text or in conjunction with encryption mechanisms like AES (Advanced Encryption Standard). Unlike SIP, no confusion is raised by identity related information used to enforce authentication.
- ❖ IAX authentication is implemented thanks to the exchange of authentication requests, which enclose a security challenge. This authentication challenge should be answered by the remote peer and encrypted according to the adopted encryption method. If encryption negotiation fails, the call is aborted immediately.
- ❖ IAX provides a dedicated scheme to supervise its devices and messages through a specific procedure.
- ❖ IAX allows a procedure to check the availability of a new firmware version for a given device type. The encoding of firmware binary blocks is specific to IAX devices and is out of the scope of the IAX communication protocol itself.
- ❖ IAX can be easily deployed to provide heterogeneous calls between IPv4 and IPv6 realms.

Below is a scheme of IAX trunk architecture:



*Fig. 5 – IAX trunking architecture*

The activation of IAX (RFC 5456) in an operational network will simplify current architectures and therefore there will be no need to introduce expensive and SIP-unfriendly nodes. The proposed IAX (RFC 5456) introduction scenario is accompanied by an extension to SDP (RFC 4566) to allow smooth migration and media optimization.

#### **2.4.2 SOLVING VoIP PROBLEMS WITH IAX**

Unlike other VoIP signaling protocols, IAX (RFC 5456) uses a single port number to send and/or receive both media and control data, i.e. port 4569 precisely. IAX (RFC 5456) does not require the inclusion of IP-related information in its call setup requests, it uses one single session for both signaling and media traffic exchange. A multiplexing capability is supported by IAX to distinguish outbound from inbound sessions. For the inbound flows, a given IAX peer uses an identifier called *Destination Call Number – DCN* at the seventh layer (application layer) of the OSI model.

The Internet Assigned Numbers Authority – IANA has allocated a single port (4569) for all outbound and inbound traffic flow for IAX sessions and uses UDP protocol for its traffic flow. IAX reliability features, such as control messages are acknowledgement and supported at the application level.

### **2.4.3 IAX ENSURES RELIABILITY**

IAX defines and differentiates reliable and unreliable messages. Unreliable messages are mainly media flows which are not acknowledged nor retransmitted if lost when sent over the network. IAX reliability is ensured for control messages, owing to several IAX application identifiers maintained by the peers. Reliable messages should be acknowledged; if not, these messages are retransmitted. The order of received messages is achieved by exploiting a timestamp enclosed in them [3].

### **2.4.4 REGISTERING IAX**

IAX supports the registration procedure, even if it is considered an optional feature. The IAX registration philosophy is the same as for SIP (RFC 3261). An IAX (RFC 5456) registrant contacts a registrar server with specific messages. The contact information is then retrieved by the registrar server and stored in its system within a time period [3].

IAX (RFC 5456) registration differs from SIP registration as the latest uses a dedicated message called *REGISTER* to notify a register server about its Address of Contact – AoC. An AoC may be a Fully Qualified Domain Name – FQDN or an IP address and a port number. If a registered user agent – UA is behind a network address translator – NAT, this AoC must be modified by an Application-Level Gateway – ALG so as to replace its information with some which is more pertinent [3].

Unlike with SIP, IAX registration messages do not enclose any IP-related information (i.e. IP address and port number). The registrar server extracts the source IP-related information and stores it. This information will be used to contact the registrant user agent as no ALG is required to be modified on IAX messages.

### **2.4.5 TRANSPORTATION OF MEDIA STREAMS IN IAX**

Different from SIP, IAX does not require a new protocol to exchange media streams as various media types may be sent by IAX: voice, video, image, text, HTML (Hypertext Markup Language) and so on. IAX uses an optimized protocol header of 4 bytes to send audio messages. Nevertheless, it is important to keep in mind that IAX media streams use the same port number as for control/signaling messages.

### **2.4.6 IAX CODEC NEGOTIATION**

IAX does negotiate a codec when issuing the first call-establishment request. A given IAX peer can indicate one or several Compression/Decompression – CODECs it supports. The remote IAX peer must select one of these CODECs and indicates it in its response message.

### **2.4.7 IAX AND SECURITY RELATED ISSUES**

IAX (RFC 5456) is a point-to-point communication protocol, as a native IP security protocol and architectures can be deployed jointly with IAX (RFC 5456) as it allows exchange of shared keys and may be used either with plain text or in conjunction with encryption mechanisms such as Advanced Encryption Standard – AES, which are based on a shared secret. The IAX (RFC 5456) authentication procedure is implemented by an exchange of authentication requests which enclose a security challenge.

This authentication challenge should be answered by the remote peer and encrypted according to the adopted encryption method. If encryption negotiation fails, the call should be terminated.

### **2.4.8 ADVANTAGES OF IAX**

- ❖ IAX is a binary protocol specially created to reduce crosstalks voice applications. In some cases, the power transmission line deteriorates over in favor of transfer capacity for individual calls. For instance, if a call is to be transfer to a compressed 8 kbps and 20ms packetization, each data packet has 20 bytes. IAX increases overall value, 4



bytes crosstalk and RTP added another 12 apartment into each packet. Binary protocols are also resistant against buffer over-run attacks.

- ❖ IAX is a very robust and simple protocol that can transfer almost all kinds of transmission. However, it is optimized for the needs of the VoIP call, where the priority of the low demands on the transmission lines and small circuits. These priorities are of a great deal for a VoIP protocol than the other protocols that are more detailed properties, but extend into areas that are necessary in today's data transfer or peer to peer communications. It is not wrong to say that IAX is designed as a light and friendly VoIP protocol that uses less network bandwidth than other protocols.
- ❖ In addition to performance, IAX (RFC 5456) uses SIP (RFC 3261) as well as a static UDP port which facilitate the network administrator's work, to easily manage, prioritize and permit communication firewalls. The basic idea of IAX is multiplexing signals, messages and communications into a single UDP channel between two nodes. It also uses the same UDP (4569) port for signal transmission of new calls and communication and thanks to the fact that all communication calls were made at the same point-to-point route the NAT (RFC 2663) settings are much lighter than the other protocols.

IAX protocol was designed to provide control and transmission of VoIP data between Asterisk servers. Nowadays it is also used for connections between clients and servers which support the protocol. The present version of IAX is IAX2 since the first version of IAX is obsolete. IAX is a protocol designed and thought for VoIP connections (audio streaming) although it can support other types of data (for example video streaming) [16].

**The main goals of IAX are:**

- ✓ Minimize bandwidth usage for both control and media transmissions with specific emphasis on individual voice calls.
- ✓ Avoid NAT problems (Network Address Translation).
- ✓ Support the ability to transmit dialplan information.

To reduce the bandwidth IAX2 uses a binary protocol instead of a text protocol like SIP and that is why the size of IAX2 messages is less than SIP messages. In order to avoid NAT problems, IAX2 uses UDP transport protocol, normally on port 4569, (IAX1 used port 5036),

and both, signaling information and data go together using the same protocol (unlike SIP) Therefore, IAX has less NAT problems and it can pass through routers and firewalls in a better way [16].

- ❖ IAX has its own frames (full and mini frames) to carry control, audio, video messages and signals. IAX full frames are responsible to carry the signals and the control messages, while Mini frame are the only type of frames which hold the data during the media transmission session between two endpoints A and B. Each audio/video flow is of IAX Mini Frames –  $M\_frames$  contains 4 bytes header (source call number 2 bytes and timestamp also 2 bytes. The IAX audio packet format consists of four parts, which are the *IP header*, the *UDP header*, the *IAX header* and the *IAX payload* (which varies depending on the codec used) [1]. Audio packet format of the IAX protocol is shown below.

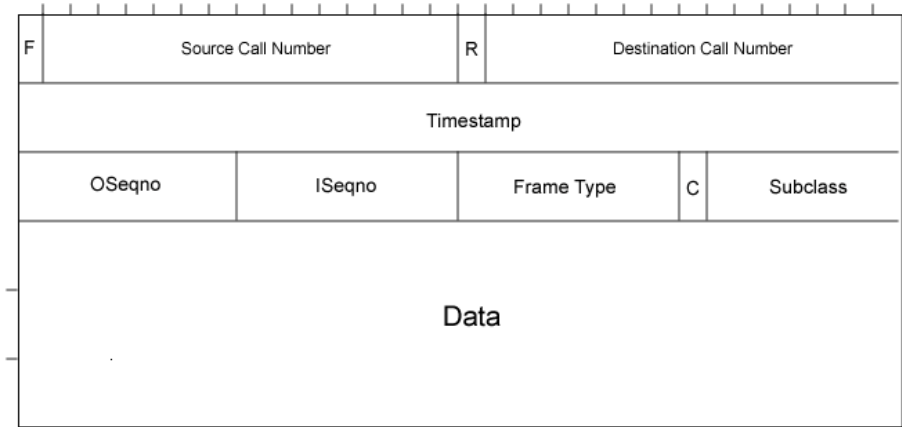
<b>IP [20 Bytes]</b>	<b>UDP [8 Bytes]</b>	<b>IAX header [4 Bytes]</b>	<b>IAX Payload</b>
----------------------	----------------------	-----------------------------	--------------------

*Table 2 – IAX Audio Packet Format*

The frames or messages that are sent in IAX2 are binary and therefore each bit or each set of bits has a meaning. There are basically two types of frames; Full and Mini Frames.

### **Full Frames**

Full Frames are the only frame types are transmitted reliably. This means that the recipient host must return some type of message back to the sending host immediately upon reception.



*Fig.6 Illustration of a full frame [14]*

The meaning of each field is the following one:

**F:** is used in indicate whether a frame is a Full Frame or not. A value of 1 in this field indicates the frame is a Full Frame and a value 0 indicates the frame is something other than a Full Frame.

**Source Call Number:** a 15-bit unsigned integer that is used to track a media stream endpoint on the source host.

**R:** Set to the value 1 if this frame is being retransmitted and the value 0 for the initial transmission

**Destination Call Number:** the same as source call number but with destination instead of source.

**Timestamp:** To stamp the time of each packet

**OSeqno:** Outbound stream sequence number. OSeqno always begins with 0 and increases monotonically.

**ISeqno:** is similar to OSeqno, except that it is used to track the ordering of inbound media frames.

**Frame Type:** Frame type

**C:** If C is set to 1, the Subclass value is interpreted as a power of two. If C is set to 0, Subclass is interpreted as a simple 7-bit unsigned integer value.

**Subclass:** Subclass type of message.

**Data:** Data sent in binary format [14].

### Mini Frames:

A Mini Frame is used to send media with a minimal protocol overhead. These frames are not transmitted reliably. If one of these frames is lost it is not retransmitted. The binary format for mini frames is the following one:

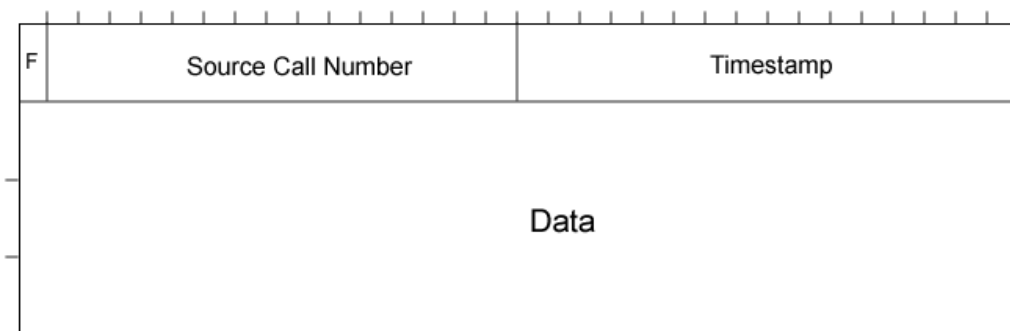


Fig. 7 Illustration of a mini frame [14]

The meaning is similar to the Full frames but the **F**: bit is set to 0. The Timestamp in the Mini Frame is truncated. The client generally maintains a 32-bit full timestamp.

When sending Mini Frames, only the low 16 bits of the timestamp are sent in the Timestamp field. When the 16-bit timestamp wraps around, a Full Frame is sent to allow the other end to synchronize its full 32-bit timestamp counter [14].

#### 2.4.9 IAX FULL FRAMES VALUES

The Type Frame field of **F** frames along with the subclass field determines the function of the package that has been sent or received. These are the control signaling messages. The Type Frame field is a set of 8 bits (1 byte) and the main values that can take are shown in the following table [17]:

<i>Type frame Value</i>	<i>Description</i>	<i>Details</i>
00000001	DTMF	To send DTMF digits
<b>00000002</b>	<b>Voice Data</b>	<b>Subclass field show the audio codec used. See next table</b>
00000003	Video	Subclass field show the video codec used.
<b>00000004</b>	<b>Control</b>	<b>Provide session control. They refer to control of the devices connected to the IAX endpoint. Subclass field show the specific control message. See table 3.</b>
00000005	NOT USED	
<b>00000006</b>	<b>IAX control</b>	<b>Provide IAX protocol specific endpoint management. They are used to manage IAX protocol interactions that are generally independent of the type of endpoints. Subclass field show the specific IAX control message.</b>
00000007	Text	
00000008	Image	
00000009	HTML	

*Table 3 – Main values of the "Type Frame" field of Full Frame [17]*

<i>Subclass Value (Type Frame =0x02)</i>	<i>Description (codec used in the conversation)</i>
0x0001	G.723.1
0x0002	GSM
0x0004	G.711 u (u-law)
0x0008	G.711 a (a-law)
0x0080	LPC10
0x0100	G.729
0x0200	Speex
0x0400	iLBC

*Table 4 – Meaning of the subclass values for Frame Type ="0x02" (voice data) [17]*

<i>Subclass Value (Type Frame =0x04)</i>	<i>Description</i>
0x01	Hangup
0x02	Ring
0x03	Ringing (ringback)
0x04	Answer
0x05	Busy Condition
0x08	Congestion Condition
0x0e	Call Progress

*Table 5 – Meaning of the subclass values for Frame Type ="0x04" (Control) [17]*

<i>Subclass Value (Type Frame =0x05)</i>	<i>Description</i>	<i>Details</i>				
0x01	New	Initiate a new call		0x10	RegRej	Registration reject
0x02	Ping	Ping request		0x11	RegRel	Registration release
0x03	Ping	Ping reply		0x12	VNAK	Video/Voice retransmit request
0x04	Ack	Acknowledgement		0x13	DPRReq	Dialplan request
0x05	Hangup	Initiate call teardown		0x14	DPRRep	Dialplan response
0x06	Reject	Reject		0x15	DIAL	Dial
0x07	Accept	Accepted		0x16	TXReq	Transfer request
0x08	AuthReq	Authentication request		0x17	TXCNT	Transfer connect
0x09	AuthRep	Authentication reply		0x18	TXACC	Transfer accept
0x0a	Inval	Invalid call		0x19	TXReady	Transfer ready
0x0b	LagReq	Lag request		0x1a	TXRel	Transfer release
0x0c	LagRep	Lag reply		0x1b	TXRej	Transfer reject
0x0d	RegReq	Registration request		0x1c	QUELCH	Halt audio/video transmission
0x0e	RegAuth	Registration authenticate		0x1d	UNQUELCH	Resume audio/video transmission
0x0f	RegAck	Registration acknowledgement		0x20	MWI	Message waiting indication
				0x21	Unsupport	Unsupported message

*Table 6 – Meaning of the subclass values for Frame Type = "0x06" (IAX control) [17]*

#### 2.4.10 IAX COMMUNICATION CALL FLOW

In order to understand IAX protocol, below is a real example of communication using the main IAX messages:

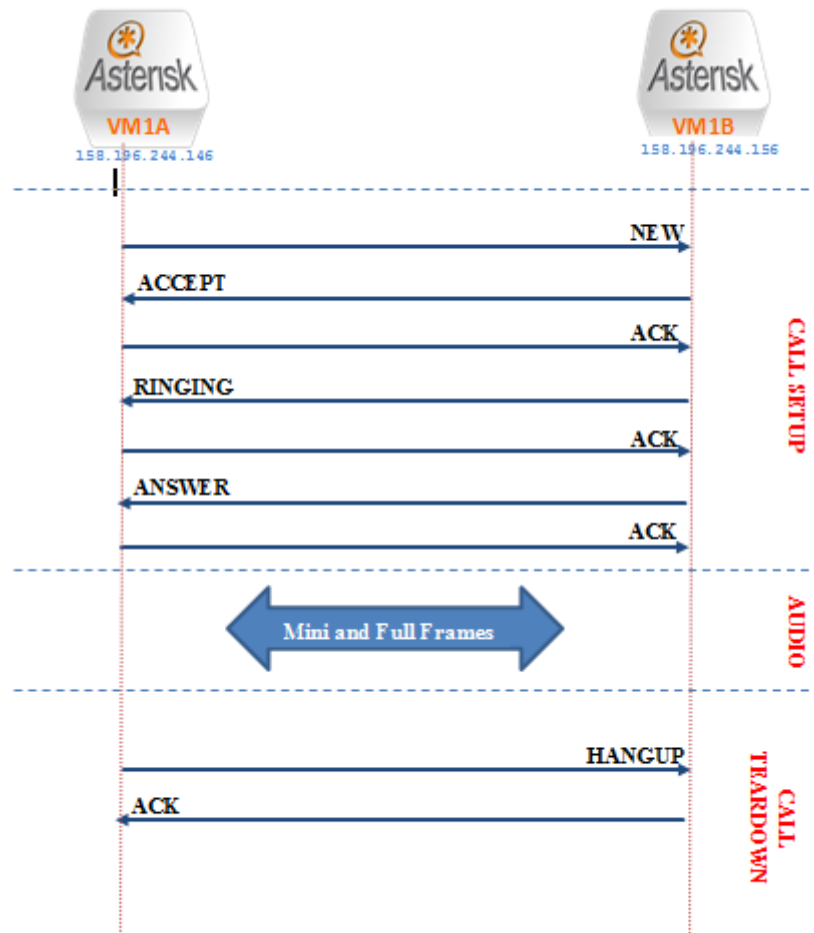


Fig. 8 – IAX call flow example [18]

An IAX call has three common steps or situations:

##### **Call setup:**

A terminal starts the connection and sends a "new" message. The called terminal replies with an "accept" message and the caller replies also with an "Ack" message. Next, the called terminal gives the "ringing" signal and the caller sends an "Ack" to confirm the reception of the message. Finally, the called accepts the call with an "answer" message and the caller acknowledges that message. The call setup is then established [18].

### **Media or Audio Flow:**

Mini and Full frames are sent in either direction with the audio data. Each flow is comprised mostly of IAX Mini Frames (M frames) which contain a simple 4-byte header that targets bandwidth efficiency. The flow is supplemented by periodic Full Frames (F frames) that include synchronization information. It is important to notice that these audio messages are using the same UDP protocol that are using the signaling messages (*call setup* and *call teardown*) avoiding NAT issues [18].

### **Call Teardown**

The *call teardown* is done just by sending a "*hangup*" and *acknowledge* messages from one direction to the other [18].

## **2.5 COMPARISON OF MAIN DIFFERENCES OF SIP vs IAX PROTOCOLS**

IAX was developed with the goal to improve some of the problems related with SIP in VoIP system once the following issues/differences were defined:

### **Bandwidth**

The bandwidth used by IAX is less than the one used by SIP since the messages are binary instead of text messages (SIP). IAX also tries to reduce the headers of the messages reducing therefore the bandwidth used [19].

### **NAT**

Signaling and data travel together in IAX avoiding the problems of NAT that usually appear in SIP. Signaling and data in SIP travel using different protocols and that is why NAT problems appear. Audio stream have to pass through routers and firewalls. SIP usually needs a STUN server to avoid these problems [19].

### **Standardization and utility**

SIP is a protocol standardized by the IETF long time ago and it is widely used by the equipment and software manufacturers. IAX is still being standardized and for this specific reason not many devices can use it [19].



### **Ports used**

IAX uses only one port (4569) to send signaling and data of all the calls (inbounds and outbound calls). To make this possible, IAX uses a trunking system which enables it multiplexing both signaling and multiple media streams over a single User Datagram Protocol (UDP). SIP, otherwise, uses one port (5060) for signaling and two RTP ports for each audio connection (at least 3 ports). For instance, if we have 100 simultaneous calls we should use 200 RTP ports and one port for signaling (5060) [19].

### **Audio flow when using a server**

If SIP is using a server, all its signaling messages always do pass through the server but audio messages (RTP flow) can travel end to end without passing through that server. In IAX, signaling and data must always pass through IAX server, thus increases the bandwidth needed by the IAX servers when there are many simultaneous calls [19].

### **Other functionalities**

IAX is a protocol developed for VoIP and video transmission and it has many interesting functionalities, for example, the possibility to send or receive dialplans. These functionalities are very useful when using an Asterisk PBX. SIP is a general purpose protocol and can transmit any information, not only audio and video [19].

### 3 BUILDING TRUNKS BETWEEN ASTERISK BASED ON SIP AND IAX

Asterisk is organized in *Contexts*, therefore, we can define a context for each thing, and for instance, we can say that all my phones will start in *context phones*. We can say that every calls coming from outside should go to the *context inbounds*, if I want to make an outgoing call it goes to the *context outbound* calls. A context is a name space where things can be defined or set to not to touch things on the nearby context.

For instance, talking about extensions, there could be context A with extension 100 and context B also with extension 100; this is completely a different name space in the SIP configuration.

To set up my SIP context, I proceeded as per below:

Logged in as a super user *sudo -s* to get *root* access, I issued the following command:

#### VM1A

```
root@liptel-ea005-146~#nano /etc/asterisk/sip.conf
```

```
[general]
context=local ;Default context for inbound calls. Defaults to 'default'
transport=udp ;Sets the default transports protocol. Primary default transport to be determined by
order.
bindaddr=158.196.244.146 ;IP address to bind (listen on), in my case for all network interfaces on
the VM1A.

[VM1B]
type=peer ; Match inbound requests to a configuration entry using the source IP address and port
number.
host=158.196.244.156 ;finds the client or host name. It'd be dynamic if the phone were to register
itself.
defaultuser=VM1A
context=from-trunk
insecure=invite
allow=all ; All codecs are allowed here, normally alow and ulow only, as by default (disallow=all)
secret=70030
qualify=yes ;This is allowed to check the client's availability every 60 seconds.
trunk=yes
callevts=yes
notifyhold=yes
notifyringing=yes
```

[1000]

type=friend ; *This enables matching rules for both peer and user. This is the setting most commonly used for SIP phones.*

host=dynamic ; *The device will register with asterisk*

context=local ; *This is where calls from the device will enter the dialplan*

allow=all

secret=5000

qualify=yes

directmedia=no

trunk=yes

callevts=yes

notifyhold=yes

notifyringing=yes

[1001]

type=friend

host=dynamic

context=local

allow=all

secret=5000

qualify=yes

directmedia=no

trunk=yes

callevts=yes

notifyhold=yes

notifyringing=yes

## **VM1B**

*root@liptel-ea005-156~#nano /etc/asterisk/sip.conf*

[general]

context=local ; *Default context for inbound calls. Defaults to 'default'*

transport=udp ; *Sets the default transports protocol. Primary default transport to be determined by order.*

bindaddr=158.196.244.156 ; *IP address to bind (listens on), in my case for all network interfaces on the VM1A.*

[VM1A]

type=peer ; *Match inbound requests to a configuration entry using the source IP address and port number.*

host=158.196.244.146 ; *finds the client or host name. It'd be dynamic if the phone were to register itself.*

defaultuser=VM1B

context=from-trunk

insecure=invite

allow=all ; *All codecs are allowed here, normally **alow** and **ulow** only, as by default (disallow=all)*

secret=70030

qualify=yes

trunk=yes

callevts=yes

```
notifyhold=yes
notifyringing=yes
```

```
[2000]
```

```
type=friend ; This enables matching rules for both peer and user. This is the setting most commonly used for SIP phones.
```

```
host=dynamic ; The device will register with asterisk
```

```
context=local ; This is where calls from the device will enter the dialplan
```

```
allow=all
```

```
secret=5000
```

```
qualify=yes
```

```
directmedia=no
```

```
trunk=yes
```

```
allevents=yes
```

```
notifyhold=yes
```

```
notifyringing=yes
```

```
[2001]
```

```
type=friend
```

```
host=dynamic
```

```
context=local
```

```
allow=all
```

```
secret=5000
```

```
qualify=yes
```

```
directmedia=no
```

```
trunk=yes
```

```
callevts=yes
```

```
notifyhold=yes
```

```
notifyringing=yes
```

Settings for the *extensions.conf* of the trunk (SIP and IAX trunking) can be viewed in the appendix attached to this project.

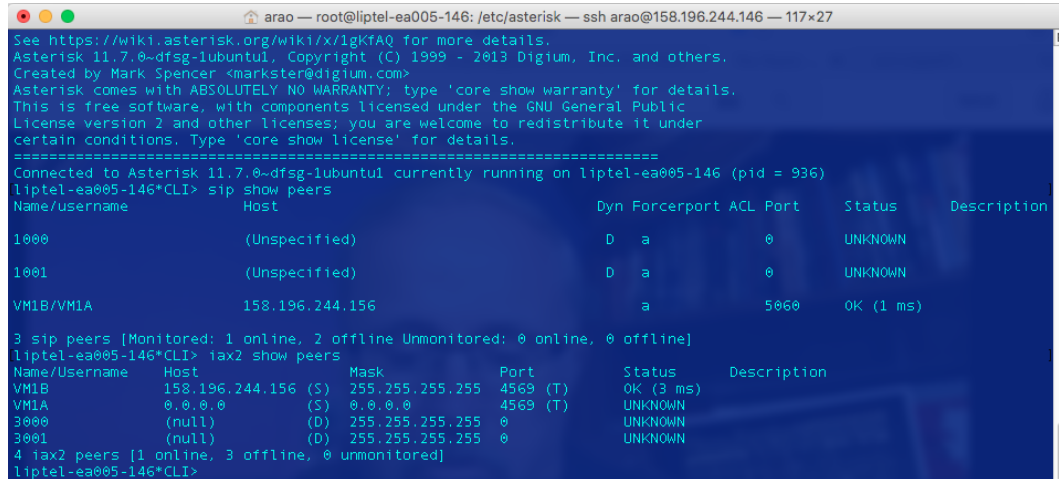
I used the same *dialplan* for both SIP and IAX as well as the same contexts with a change on the IAX extension where I have added three more digits (*\_011\_2XXX*) and highlighted with the colon three, meaning that the first three digits are to identify that the call will go on IAX through the very trunk that it shares with SIP which is using the extension *\_2XXX*.

With the command *asterisk -vvvvvv* I reconnected to the Asterisk then reloaded the SIP and IAX2 and I could see the peers on each node.

## VM1A

`root@liptel-ea005-146:/etc/asterisk# asterisk -vvvvvr`

With the commands `sip show peers` and `iax2 show peers` I can see all the peers defined in my `sip.conf` as seen in the picture:



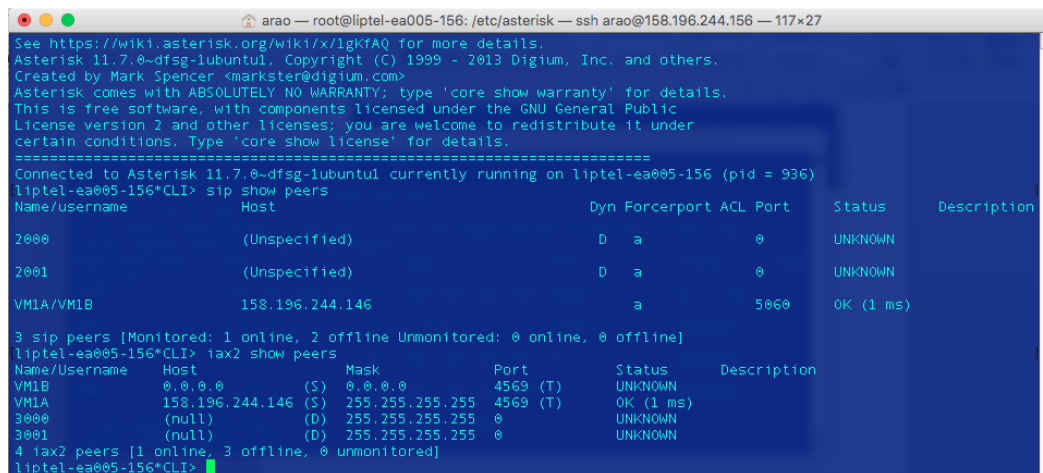
```
See https://wiki.asterisk.org/wiki/x/1gKfAQ for more details.
Asterisk 11.7.0~dfsg-1ubuntu1, Copyright (C) 1999 - 2013 Digium, Inc. and others.
Created by Mark Spencer <markster@digium.com>
Asterisk comes with ABSOLUTELY NO WARRANTY; type 'core show warranty' for details.
This is free software, with components licensed under the GNU General Public
License version 2 and other licenses; you are welcome to redistribute it under
certain conditions. Type 'core show license' for details.
=====
Connected to Asterisk 11.7.0~dfsg-1ubuntu1 currently running on liptel-ea005-146 (pid = 936)
liptel-ea005-146*CLI> sip show peers
Name/username      Host                Dyn Forcerport ACL Port    Status    Description
1000                (Unspecified)      D   a              0         UNKNOWN
1001                (Unspecified)      D   a              0         UNKNOWN
VM1B/VM1A          158.196.244.156    a              5060      OK (1 ms)

3 sip peers [Monitored: 1 online, 2 offline Unmonitored: 0 online, 0 offline]
liptel-ea005-146*CLI> iax2 show peers
Name/Username      Host                Mask          Port    Status    Description
VM1B               158.196.244.156    (S) 255.255.255.255 4569 (T) OK (3 ms)
VM1A               0.0.0.0            (S) 0.0.0.0           4569 (T) UNKNOWN
3000               (null)             (D) 255.255.255.255 0         UNKNOWN
3001               (null)             (D) 255.255.255.255 0         UNKNOWN
4 iax2 peers [1 online, 3 offline, 0 unmonitored]
liptel-ea005-146*CLI>
```

Fig. 9 – Configured SIP and IAX peers for VM1A

## VM1B

`root@liptel-ea005-156:/etc/asterisk# asterisk -vvvvvr`



```
See https://wiki.asterisk.org/wiki/x/1gKfAQ for more details.
Asterisk 11.7.0~dfsg-1ubuntu1, Copyright (C) 1999 - 2013 Digium, Inc. and others.
Created by Mark Spencer <markster@digium.com>
Asterisk comes with ABSOLUTELY NO WARRANTY; type 'core show warranty' for details.
This is free software, with components licensed under the GNU General Public
License version 2 and other licenses; you are welcome to redistribute it under
certain conditions. Type 'core show license' for details.
=====
Connected to Asterisk 11.7.0~dfsg-1ubuntu1 currently running on liptel-ea005-156 (pid = 936)
liptel-ea005-156*CLI> sip show peers
Name/username      Host                Dyn Forcerport ACL Port    Status    Description
2000                (Unspecified)      D   a              0         UNKNOWN
2001                (Unspecified)      D   a              0         UNKNOWN
VM1A/VM1B          158.196.244.146    a              5060      OK (1 ms)

3 sip peers [Monitored: 1 online, 2 offline Unmonitored: 0 online, 0 offline]
liptel-ea005-156*CLI> iax2 show peers
Name/Username      Host                Mask          Port    Status    Description
VM1B               0.0.0.0            (S) 0.0.0.0           4569 (T) UNKNOWN
VM1A               158.196.244.146    (S) 255.255.255.255 4569 (T) OK (1 ms)
3000               (null)             (D) 255.255.255.255 0         UNKNOWN
3001               (null)             (D) 255.255.255.255 0         UNKNOWN
4 iax2 peers [1 online, 3 offline, 0 unmonitored]
liptel-ea005-156*CLI>
```

Fig. 10 – Configured SIP and IAX peers for VM1B

IAX configurations:

### **VM1A**

root@liptel-ea005-146:/home/arao# nano /etc/asterisk/iax.conf

GNU nano 2.2.6

File: /etc/asterisk/iax.conf

```
[general]
language=English_US
transport=udp
bindaddr=158.196.244.146
```

```
[3000]
type=friend
host=dynamic
context=local
allow=all
secret=5000
qualify=yes
trunktimestamp=yes
callerid=CEO <3000>
```

```
[3001]
type=friend
host=dynamic
context=local
allow=all
secret=5000
qualify=yes
trunk=yes
trunktimestamp=yes
callerid=HR <3000>
```

;Outbound calls

```
[VM1B]
type=peer
defaultuser=VM1A
host=158.196.244.156
context=from-trunk
insecure=invite
allow=all
secret=5000
qualify=yes
```

```
auth=md5
trunk=yes
trunktimestamp=yes
```

```
; Inbound calls
```

```
[VM1A]
context=incoming
invite=insecure
secret=5000
qualify=yes
type=friend
auth=md5
allow=all
trunk=yes
trunktimestamp=yes
```

### **VM1B**

```
root@liptel-ea005-156:/home/arao# nano /etc/asterisk/iax.conf
```

```
GNU nano 2.2.6          File: /etc/asterisk/iax.conf
```

```
general]
language=English_US
transport=udp
bindaddr=158.196.244.156
```

```
[3000]
type=friend
host=dynamic
context=local
allow=all
secret=5000
qualify=yes
trunktimestamp=yes
callerid=CEO <3000>
```

```
[3001]
type=friend
host=dynamic
context=local
allow=all
secret=5000
qualify=yes
trunktimestamp=yes
callerid=HR <3000>
```

```

[VM1A]
type=peer
defaultuser=VM1B
host=158.196.244.146
context=from-trunk
insecure=invite
allow=all
secret=5000
qualify=yes
auth=md5
trunk=yes
trunktimestamp=yes
; Inbound calls

```

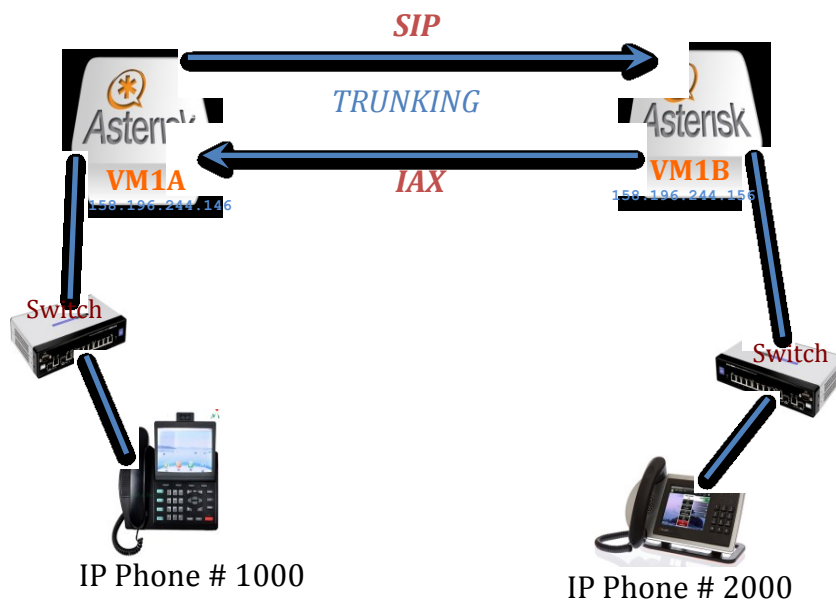
```

[VM1B]
context=from-trunk
invite=insecure
secret=5000
qualify=yes
type=friend
auth=md5
allow=all
trunk=yes
trunktimestamp=yes

```

SIP and IAX are sharing the same *dialplan* as presented in the appendix. More configured details of these settings can be seen when issued the command *sip show peer VM1A* and *sip show peer VM1B* as well as *iax2 show peer VM1A* and *iax2 show peer VM1B* respectively.

The whole architecture will look as per below:



*Fig. 11 – SIP and IAX trunking architecture*



## 4 MEASUREMENTS OF TRAFFICS ON SIP AND IAX SESSIONS

For traffic measurement on the SIP and IAX sessions, I measured it from the trunk itself. I proceeded so by installing *sipp* which allowed me to call a given number from the extensions, set up a short scrip to determine the number of calls at a time, with the help of *tcpdump* I could catch the traffic on one side and analyzed it with the help of Wireshark.

### 4.1 SIPP INSTALLATION

As my Asterisk is running on Ubuntu, I proceeded installing the *SIPP* as follows:

```
sudo apt-get install build-essential libncurses5-dev
```

```
wget
```

```
http://downloads.sourceforge.net/project/sipp/sipp/3.2/sipp.svn.tar.gz?r=&ts=1314783436&use_mirror=puzzle -O sipp.svn.tar.gz
```

```
tar -xzf sipp.svn.tar.gz
```

```
cd sipp.sh
```

as well as *iax.sh* for IAX traffic testing.

The following script is used as a loop for the calls to be multiplexed:

```
#!/bin/bash
```

```
for i in {1..n}
```

```
do
```

```
cp sipp.call /var/spool/asterisk/outgoing/$i.call
```

```
done
```

Where  $n$  denotes the range and or number of calls to be multiplexed in one second. As per default, one single SIP call has a capacity or bandwidth of 90 kbps in a full-duplex mode, to reach this capacity it is necessary to have a media as the signaling itself is not enough to measure all the necessary data of a given call.

Below is a figure of a 3 SIP calls simulated in this project:

```

GNU nano 2.2.6 File: sipp.sh Modified
#!/bin/bash
for i in {1..3}
do
  cp sipp.call /var/spool/asterisk/outgoing/${i}.call
done
  
```

Fig. 12 – Loop calls script

Once the loop is set, the command `./sipp.sh` or `./iax.sh` is issued to call the number from the VM1A (158.196.244.146) to VM1B (158.196.244.156) and on another terminal I issue the command `tcpdump -i eth0 -w /home/arao/SIP3.pcap` to capture the traffic passing through, save it with the file name `SIPn.pcap` (*n=3 in this loop*) or `IAXn` and issue another command `scp a rao@158.196.244.146:/home/arao/SIPn.pcap .` to copy the captured file into my local computer to analyze it with Wireshark by looking at the *endpoints* under the *statistics menu* as seen below:

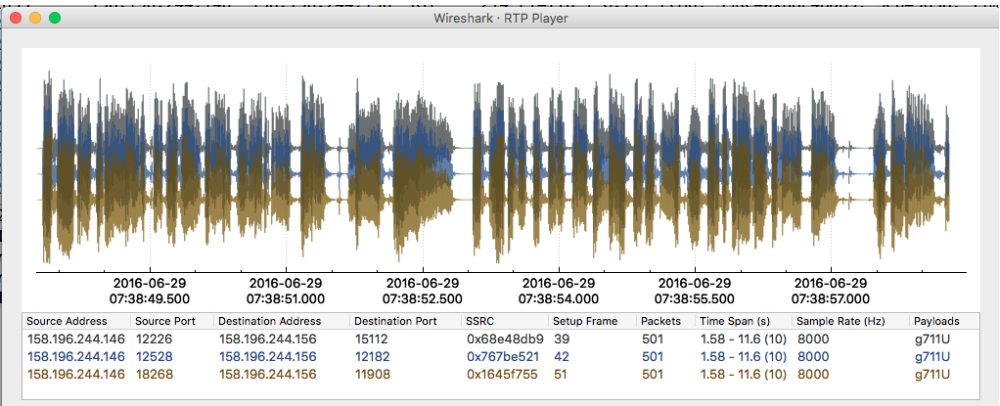
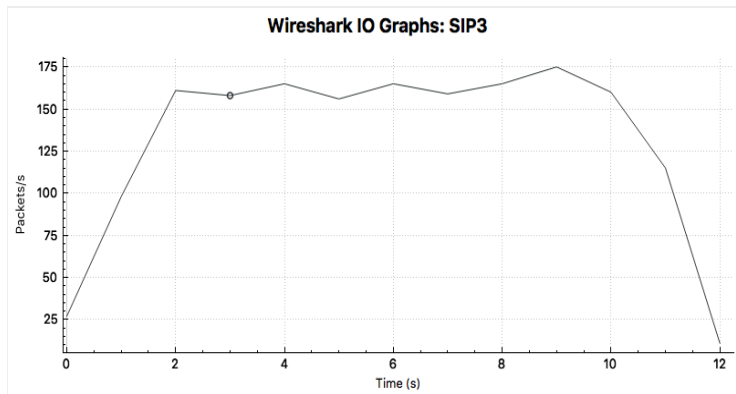
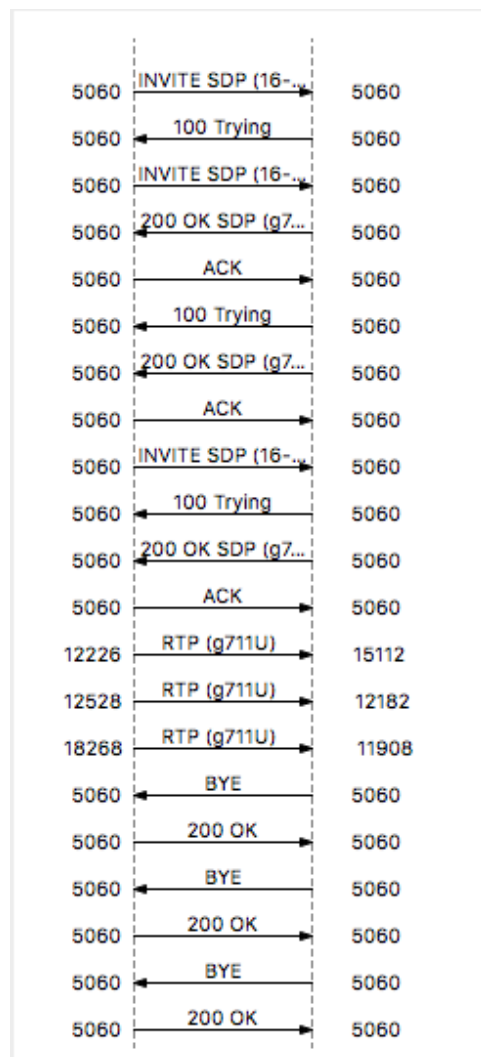


Fig. 13 – Three SIP calls with media



*Graph. 1 – Three SIP calls Input/Output with media*

Below is the 3 SIP calls flow:

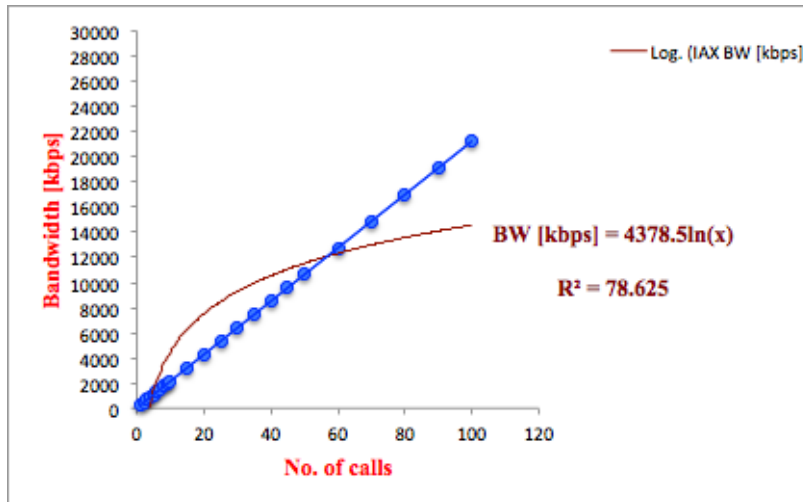


*Fig. 14 – Three SIP calls flow*

I repeated this process with twenty three loops for each of the protocols and the acquired data was analyzed separately with the goal of finding out how much bandwidth each (SIP and IAX) protocol uses during one call with a duration of three seconds as predefined in the dialplan. In the next table is seen the data measured individually, though its presentation will be separated and followed by the charts resulted of it:

Order	No. Calls	SIP BW [kbps]	IAX BW [kbps]
1	1	288	264
2	2	576	472
3	3	848	672
4	4	1136	872
5	5	1424	1080
6	6	1696	1280
7	7	1984	1488
8	8	2264	1752
9	9	2552	1952
10	10	2832	2152
11	15	4248	3232
12	20	5656	4256
13	25	7072	5336
14	30	8488	6408
15	35	9904	7424
16	40	12736	8504
17	45	12552	9576
18	50	14144	10656
19	60	16976	12752
20	70	19584	14856
21	80	22632	17000
22	90	25472	19104
23	100	28296	21248

*Tab. 7 – Measured data on SIP and IAX traffic sessions.*

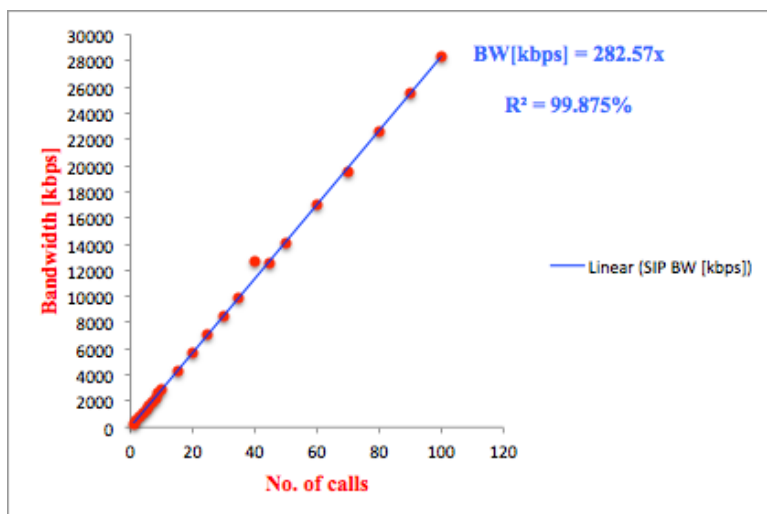


*Graph 2 – Measured IAX traffic*

Above is seen the IAX behavior as it grows to a logarithm model thus, I assume that above 100<sup>th</sup> call IAX reaches a saturation.

I used the regression model to analyze my data and found the equation presented in the graph where it can be seen the IAX's bandwidth utilization of 78.625% meaning that it can manage and operates more calls at this very bandwidth as it also grows linearly.

SIP calls do grow linearly in terms of bandwidth utilization, however at higher numbers on range of ten thousand SIP calls in ten seconds the system collapses and reboots itself. It shown in the graph that SIP calls grow linearly and has a higher saturation time.



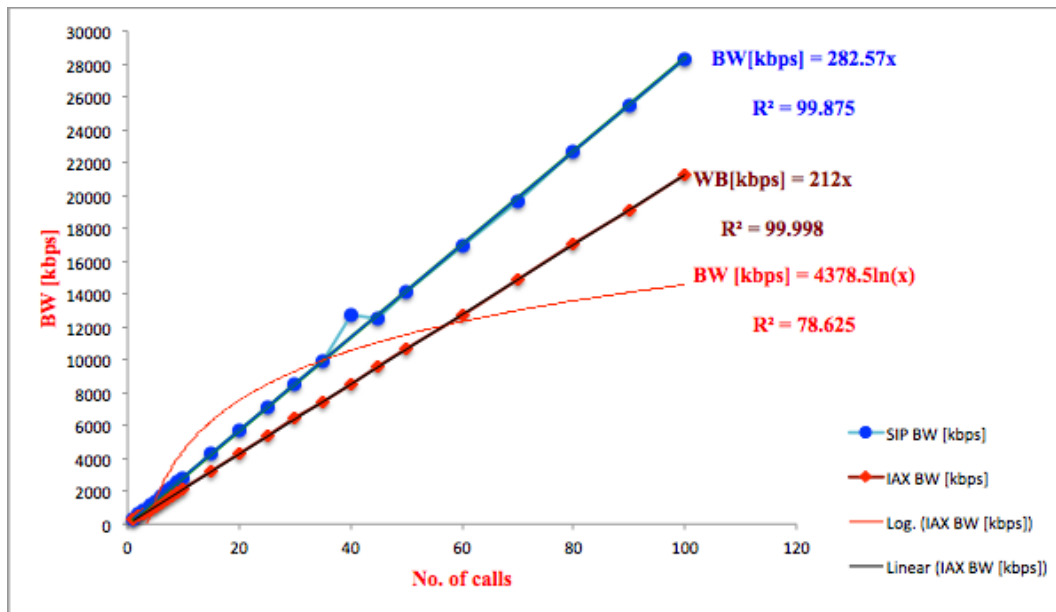
*Graph 3 – Measured SIP traffic*

SIP calls do grow linearly in terms of bandwidth use, however at huge numbers on a range of ten thousand SIP calls in ten seconds the system collapse and reboots itself. It shown in the graph that SIP calls grow linearly and has a shorter saturation time.

The new metric is an issue when the SIP traffic generator cannot be ordered to create certain number of simultaneous calls. In this case it is necessary to calculate the number of calls generated per second [7]. This can be done by this relation:

$$C_S = C_R * T \quad (1)$$

Where  $C_R$  is the Call Rate,  $C_S$  is the number of simultaneous calls planned to be generated and  $T$  is the time interval defining the call duration (media length) should be [7].



*Graph 4 – SIP vs IAX measured traffics*

I also noticed some consistency of the step index resulted in an expected shape of the chart needed to show the differences of these two protocols with a regression line and a linear representation. Both protocols do show to rise linearly and the equations resulted from each can be calculated to find the exact bandwidth of each measured number of call carried on this project.

## 5 DESIGN AND IMPLEMENTATION OF THE SIMULATION MODEL FOR SIP AND IAX TRAFFIC.

The SIP and IAX simulation resulted in two mathematical equations, one linear for SIP and a logarithm equation for IAX as we can see below:

### 5.1. SIP MODEL:

The model resulted in this simulation is a regression line, which has the formula:

$$Y = bX + A \quad (2)$$

Where  $Y$  is the predicted score,  $b$  is the slope of the line, and  $A$  is the  $Y$  intercept. In my model for SIP calls, I got the following:

$$[BW] = 282.57x, \quad ==> \quad x = \left\| \frac{[BW]}{282.6} \right\| \quad (3)$$

Where  $x$  denotes the number of calls that can be processed at a given time. For this model I got a line as geometric figure as it can be seen in the graph number three with a coefficient of determination  $R^2 = 99.986\%$  where we see how close the data are to the fitted regression line, thus I can say that SIP calls process in a linear way. In real scenarios this model is only valid and applicable when strictly considered the following:  $\{x \in \mathbb{R}: \geq 1\}$ . As per the formulae, I have ignored the value of  $A$  as it is negative.

### 5.2 IAX MODEL:

For IAX measurements also resulted the regression model with the formula denoted in equation (2) showing that it also grow linearly and when drawn the logarithm line I get the equation where the negative value representing  $A$  as per the regression formulae is also ignored.

$$[BW] = 212x \implies x = \left\lceil \frac{[BW]}{212} \right\rceil \quad (4)$$

Where  $x$  denotes the number of calls that can be processed. For this model I got a logarithm equation as it can be seen in the graph number two and its coefficient of determination  $R^2 = 78.625\%$  where again it shown how close the data are to the fitted regression line.

The equation resulted in the other line drawn from the graph is logarithm and again the  $A$  value is here ignored because it is negative:  $BW \text{ [kbps]} = 4375 \ln(x)$ , where  $\ln(x)$  denotes the number IAX calls per three seconds. Solving this equation for  $x$ :

$$4375 \ln(x) = BW \text{ [kbps]}$$

As it is represented here, the  $x$  is a number that corresponds to the bandwidth available for IAX in this model as it grows logarithm and reaches saturation of call numbers above 100.



## 6 CONCLUSION

During this final project I paid special attention to and made reference to theoretical part where I described VoIP protocol functions, mainly SIP and IAX as well as the functionality of these in details as it gave me a good understanding on what I would do in the practical part. I realized that for the practical part it is necessary to choose the correct Asterisk Version as there are many limitations and security related issues which make some functions limited if not impossible on any of these versions. In my case I have chosen the Asterisk TLS 11.7.0 (*liptel-ea005-146\*CLI> core show version*

*Asterisk 11.7.0~dfsg-1ubuntu1 built by bulld @ lamiak on a x86\_64 running Linux on 2013-12-24 06:02:10 UTC*) as it is a Long Term Supported version, I was able to build my SIP and IAX trunk, set the extensions, dialplan and manage them.

The practical part was quite challenging, however, I fulfilled the whole task with my own endurance and learned about the real differences, advantages as well as the drawbacks of each protocol and I can say with confidence which one I would prefer to use in a real scenario.

Bearing in mind that the main objective of VoIP technology is cost effectiveness, scalability and easy management of a whole telecommunication system, in my case IAX proved to be more effective as it does not increase the bandwidth utility when it reaches its saturation level. SIP calls increase the bandwidth utility in a linear way as it showed in the model applied in this simulation and Asterisk proved also not to be able to process more than five thousand SIP calls/minute at once, therefore I had to limit the number of calls up to one hundred as it can be seen in the graphs where I represent and compare traffics of both SIP and IAX.

I also concluded that in every one hundred SIP calls, IAX has 25% of cost effectiveness for this very amount of calls which is a benefit when applied in real scenario and/or business.

## LITERATURE

- [1] ALIWI, Hadeel and Putra SUMARI. Real-Time Translation Module Between IAX<sup>65</sup> and RSW. *International Journal Computer Networks & Communications (IJCNC)*. Vol. 6, No. 3, May 2014. ISSN: 0974 – 9322.
- [2] BRYANT, R., L. MADSEN., J. MEGGELEN. *Asterisk: the definitive guide*. Fourth edition. Sebastopol: O'Reilly, 2013. ISBN 9781449332426.
- [3] BOUCADAIR, Mohamed. *Inter-asterisk exchange (IAX) deployment scenarios in SIP-enabled networks*. Chichester, U.K.: Wiley, 2009. ISBN 0470770724.
- [4] JOHNSTON, Alan, B. *SIP: understanding the Session Initiation Protocol*. 3rd ed. Boston: Artech House, c2009. ISBN 978-1607839958.
- [5] LAZZEZ, Amor and Thabet SLIMANI. Deployment of VOIP Technology: QoS Concerns. *International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE)*. Vol. 2, Issue 9, September 2013. ISSN 2278-1021.
- [6] PARK, Patrick. *Voice over IP security*. Indianapolis, Ind.: Cisco Press, c2009. ISBN 1587054698.
- [7] VOZNAK, Miroslav and Jan ROZHON. Approach to stress tests in SIP environment based on marginal analysis. In *Telecommunications Systems*. Springer, March 2013, Volume 52, Issue 3, pp. 1583-1593.

## Electronic resources

- [8] BANERJEE, Kousik. SIP (*Session Initiation Protocol*). [online]. [cit. 2015-12-08]. Available from: <http://www.siptutorial.net/SIP/>
- [9] MASTALIR, Jan. *Understanding SIP-Based VoIP* [online]. [cit. 2015-12-07]. Available from: [https://www.packetizer.com/ipmc/sip/papers/understanding\\_sip\\_voip/](https://www.packetizer.com/ipmc/sip/papers/understanding_sip_voip/)
- [10] ROSENBERG et al. SIP: Session Initiation Protocol. [online]. [cit. 2015-02-16]. Available from: <https://www.ietf.org/rfc/rfc3261.txt>

- [11] SPENCER et al. IAX: Inter-Asterisk eXchange Version 2. [online]. [cit. 2015-10-10], ISSN: 2070-1721. Available from: <https://tools.ietf.org/html/rfc5456>
- [12] <http://www.asterisk.org/get-started> [online]. [cit. 2016-11-04]
- [13] <http://www.cs.columbia.edu/sip/draft/sip-bis/functionality.tex>. [online]. [cit. 2016-01-04]
- [14] <http://www.en.voipforo.com/IAX/IAX-frames.php>. [online]. [cit. 2016-07-06]
- [15] <http://core0.staticworld.net/images/idge/imported/article/nww/2009/01/01fig01-100278400-orig.jpg>. [online]. [cit.2015-11-06]
- [16] <http://www.en.voipforo.com/IAX/IAX-architecture.php>. [online]. [cit. 2016-07-06]
- [17] <http://www.en.voipforo.com/IAX/IAX-F-frames.php>. [online]. [cit. 2016-07-06]
- [18] <http://www.en.voipforo.com/IAX/IAX-example-messages.php>. [online]. [cit. 2016-07-06]
- [19] <http://www.en.voipforo.com/IAX/IAXvsSIP.php>. [online]. [cit. 2016-07-06]

## ACRONYMS

VoIP	<i>Voice Over Internet Protocol</i>
IP	<i>Internet Protocol</i>
IAX	<i>Inter-Asterisk Exchange</i>
SIP	<i>Session Initiation Protocol</i>
ISDN	<i>Integrated Service Digital Network</i>
QoS	<i>Quality of Services</i>
PSTN	<i>Public Switched Telephone Network</i>
DSL	<i>Digital Subscriber Line</i>
ATA	<i>Analog Telephone Adapter</i>
IM	<i>Instant Messaging</i>
PBX	<i>Private Branch eXchange</i>
C2C	<i>Call-to-call</i>
GUI	<i>Graphical User Interface</i>
USA	<i>United States of America</i>
FMFM	<i>Find-Me-Follow-Me</i>
ITU-T	<i>International Telecommunication Union</i>
NAT	<i>Network Address Translation</i>
MGCP	<i>Media Gateway Control Protocol</i>
LI	<i>Lawful Interception</i>
IDS	<i>Intrusion Detection System</i>
IETF	<i>Internet Engineering Task Force</i>
RTP	<i>Real-Time Protocol</i>
RFC	<i>Request For Comment</i>
RTSP	<i>Real-Time Streaming Protocol</i>
MEGACO	<i>Media Gateway Control Protocol</i>
SDP	<i>Session Description Protocol</i>
S/MIME	<i>Secure/Multipurpose Internet Mail Extensions</i>
SRTP/SRTCP	<i>Secure Real-Time Protocol/Secure Real-Time Protocol Control Protocol</i>
TLS	<i>Transport Layer Security</i>
IPSec	<i>Internet Protocol Security</i>
DoS	<i>Denial of Services</i>
UA	<i>User Agent</i>
AOR	<i>Address of Record</i>
URI	<i>Uniform Resource Identifier</i>
B2BUA	<i>Back-to-Back User Agent</i>
UAS	<i>User Agent Server</i>
UAC	<i>User Agent Client</i>
SIPS	<i>Secure Session Initiation Protocol</i>
ID	<i>Identifier</i>
CDR	<i>Call Detail Recording</i>
CEL	<i>Channel Event Logging</i>
API	<i>Application Programming Interface</i>
RADIUS	<i>Remote Authentication Dial In User Service</i>
PRI	<i>Primary Rate Interface</i>
WAV	<i>Waveform Audio File Format</i>
GSM	<i>Global System for Mobile Communications</i>

ODBC	<i>Open Database Connectivity</i>
CLI	<i>Command Line Interface</i>
AEL	<i>Asterisk Extension Logic</i>
CPU	<i>Central Processing Unit</i>
KDE	<i>Kubuntu Desktop Environment</i>
RHEL	<i>Red Hat Enterprise Linux</i>
VM1A	<i>Virtual Machine number One-A</i>
VM1B	<i>Virtual Machine number One-B</i>
BIOS	<i>Basic Input/Output System</i>
SSH	<i>Secure shell</i>
EDUROAM	<i>Education Roaming</i>
DAHDI	<i>Digium Asterisk Hardware Device Interface</i>
UDP	<i>User Datagram Protocol</i>
STUN	<i>Simple Traversal for UDP through NAT</i>
ICE	<i>ISDN Capacity Enhancement</i>
TURN	<i>Traversal Using Relay NAT</i>
AES	<i>Advanced Encryption Standard</i>
DCN	<i>Destination Call Number</i>
IANA	<i>Internet Assigned Numbers Authority</i>
AoC	<i>Address of Contact</i>
FQDN	<i>Fully Qualified Domain Name</i>
ALG	<i>Application Layer Gateway</i>
HTML	<i>Hypertext Markup Language</i>
CODEC	<i>Compression/Decompression</i>
kbps	<i>Kilobit per second</i>
RSW	<i>Real Time Switching Control Criteria</i>
LTS	<i>Long Term Support</i>
DTMF	<i>Dual-Tone Multi-Frequency</i>
iLBC	<i>Internet Low Bitrate Codec</i>

## LIST OF TABLES

*Table 1 – RTP implementations*

*Table 2 – IAX Audio Packet Format*

*Table 3 – Measured data on SIP and IAX traffic sessions.*

## LIST OF FIGURES

*Fig. 1 – VoIP service architecture with many different types of services integrated*

*Fig. 2 – Possible sources of security breaches in VoIP systems*

*Fig. 3 – Asterisk vs PBX architecture*

*Fig. 4 – Asterisk dial plan syntax - extensions.conf*

*Fig. 5 – IAX trunking architecture*

*Fig. 6 – Illustration of a full frame [14]*

*Fig. 7 Illustration of a mini frame [14]*

*Fig. 8 – IAX call flow example [18]*

*Fig. 9 - Configured SIP and IAX peers for VM1A*

*Fig. 10 – Configured SIP and IAX peers for VM1B*

*Fig. 11 – SIP and IAX trunking architecture*

*Fig. 12 – Loop calls script*

*Fig. 13 – Three SIP calls with media*

*Fig. 14 – Three SIP calls flow*

## **LIST OF GRAPHS**

*Graph 1 – Three SIP calls Input/Output with media*

*Graph 2 – Measured IAX traffic*

*Graph 3 – Measured SIP traffic*

*Graph 4 – SIP vs IAX measured traffics*



## APPENDIX

### *SIP INSTALLATION*

Below are presented the technical features of each of my virtual machines:

#### **VM1A**

Welcome to Ubuntu *14.04.4 LTS (GNU/Linux 3.13.0-79-generic x86\_64)*

\* Documentation: <https://help.ubuntu.com/>

System information as of Sun Feb 07 23:34:17 CET 2016

System load: 0.0                      Processes:      82  
Usage of /: 15.4% of 14.64GB   Users logged in: 0  
Memory usage: 8%                      address for eth0:  
158.196.244.146  
Swap usage: 0%

#### **VM1B**

Welcome to *Ubuntu 14.04.4 LTS (GNU/Linux 3.13.0-79-generic x86\_64)*

\* Documentation: <https://help.ubuntu.com/>

System information as of Sun Feb 07 23:38:36 CET 2016

System load: 0.27                      Processes:      81  
Usage of /: 15.4% of 14.64GB   Users logged in: 0  
Memory usage: 8%                      IP address for eth0: 158.196.244.156  
Swap usage: 0%

By issuing the command list show hardware (*lshw*), I can present a description of my physical equipment used in this simulation, as shown below:

## VM1A

root@liptel-ea005-146:~# lshw

liptel-ea005-146

description: Computer

product: VMware Virtual Platform ()

vendor: VMware, Inc.

version: None

serial: VMware-56 4d 5e 46 e7 66 f6 6d-13 15 76 2d 9b 3f b6 2d

width: **64 bits**

capabilities: smbios-2.4 dmi-2.4 vsyscall32

configuration: *administrator\_password=enabled boot=normal*

*frontpanel\_password=unknown keyboard\_password=unknown power-on\_password=disabled uuid=564D5E46-E766-F66D-1315-762D9B3FB62D*

\*-core

description: Motherboard

product: 440BX Desktop Reference Platform

vendor: Intel Corporation

physical id: 0

version: None

serial: None

\*-firmware

description: BIOS

vendor: Phoenix Technologies LTD

physical id: 0

version: 6.00

date: 06/22/2012

size: 87KiB

capabilities: isa pci pcmcia pnp apm upgrade shadowing escd cdboot bootselect edd  
int5printscreens int9keyboard int14serial int17printer int10video acpi smartbattery  
biosbootspecification netboot

\*-cpu:0

*description: CPU*

*product: Intel(R) Xeon(R) CPU E5-2450 0 @ 2.10GHz*

*vendor: Intel Corp.*

*physical id: 4*

*bus info: cpu@0*

*slot: CPU socket #0*

*size: 2100MHz*

*capacity: 4230MHz*

*width: 64 bits*

## VM1B

root@liptel-ea005-156:~# lshw

liptel-ea005-156

description: Computer

product: VMware Virtual Platform ()

```

vendor: VMware, Inc.
version: None
serial: VMware-56 4d 1e 1d 7b 68 e0 12-1d e0 2e 34 27 8a cd 50
width: 64 bits
capabilities: smbios-2.4 dmi-2.4 vsyscall32
configuration: administrator_password=enabled boot=normal
frontpanel_password=unknown keyboard_password=unknown power-on_password=disabled
uuid=564D1E1D-7B68-E012-1DE0-2E34278ACD50
*-core
  description: Motherboard
  product: 440BX Desktop Reference Platform
  vendor: Intel Corporation
  physical id: 0
  version: None
  serial: None
*-firmware
  description: BIOS
  vendor: Phoenix Technologies LTD
  physical id: 0
  version: 6.00
  date: 06/22/2012
  size: 87KiB
  capabilities: isa pci pcmcia pnp apm upgrade shadowing escd cdboot bootselect edd
int5printscreens int9keyboard int14serial int17printer int10video acpi smartbattery
biosbootspecification netboot
*-cpu:0
  description: CPU
  product: Intel(R) Xeon(R) CPU E5-2450 0 @ 2.10GHz
  vendor: Intel Corp.
  physical id: 4
  bus info: cpu@0
  slot: CPU socket #0
  size: 2100MHz
  capacity: 4230MHz
  width: 64 bits

```

## VM1A

```
root@liptel-ea005-146~#nano /etc/asterisk/extensions.conf
```

```

[local]
; dial out through trunk
exten => _2XXX,1,Dial(SIP/VM1B/${EXTEN})
exten => _2XXX,n,Hangup()

exten => _0112XXX,1,Dial(IAX2/VM1B/${EXTEN:3})

```

```
[incoming]
include => local
```

```
;dial local number
exten => _1XXX,1,Answer()
exten => _1XXX,n,Playback(beep)
exten => _1XXX,n,Hangup()
```

```
[from-trunk]
;dial local number
exten => _1XXX,1,Answer() ;Dial(SIP/${EXTEN})
exten => _1XXX,n,Playback(beep)
exten => _1XXX,n,wait(3)
exten => _1XXX,n,Hangup()
```

## **CONFIGURATIONS SETTINGS:**

### **VMIB**

```
root@liptel-ea005-156~#nano /etc/asterisk/extensions.conf
```

```
[from-trunk]
exten => _2XXX,1,Answer()
exten => _2XXX,n,wait(3)
exten => _2XXX,n,Hangup()
;exten => _2XXX,n,Monitor(wav,myfilename)
;exten => _3XXX,n,wait(3)
```

```
exten => _3XXX,1,Answer()
exten => _3XXX,n,wait(3)
exten => _3XXX,n,Hangup()
;exten => _3XXX,n,Monitor(wav,myfilename)
;exten => _3XXX,n,wait(3)
```

SIPP files (XML scenarios) were not added to this appendix as I was unsuccessful to extract them from the servers.