Green VoIP

A SIP Based Approach

GOCE TALAGANOV

KTH Information and Communication Technology

Degree project in
Communication Systems
Second level, 30.0 HEC
Stockholm, Sweden

Green VoIP: A SIP Based Approach

Goce Talaganov
gocet@kth.se

2012.07.02

Master's thesis

Examiner and Supervisor:
Professor Gerald Q. Maguire Jr.

Communication Systems (CoS)
School of Information and Communication Technology
KTH Royal Institute of Technology
Stockholm, Sweden

Abstract

This master thesis presents, examines, designs, implements, and evaluates with respect to energy efficiency a secure and robust VoIP system. This system utilizes a Session Initiation Protocol (SIP) infrastructure assisted by a cloud service, specifically focusing on small to medium sized enterprises (SME) and homes. This research focuses on using inexpensive, flexible, commodity embedded hardware (specifically a Linksys WRT54GL wireless router for the local site with a customized operating system, specifically DD-WRT). The idea is to reduce the local site's power consumption to very low levels by examining which functions can be done in a cloud service rather than at the local site.

The thesis presents the design of a low-power IP telephony system for the local site and the cloud site. A number of different usage scenarios and desirable features are described. The methodology for conducting a set of experiments is defined to perform stress-testing and to evaluate the low-power IP telephony system's design. The experiments concern the overall power consumption of the local site under various configurations, the VPN link's call capacity, the QoS metrics for the VoIP calls, the session request delay (SRD) and the registration request delay (RRD).

The results from these experiments show that there is a potential for significant power savings when using the proposed design for an IP telephony system.

Sammanfattning

Detta examensarbete presenterar, undersöker, utformar, implementerar, och försöker att utvärdera ett säkert och robust VoIP-system med energieffektivitet. Detta system använder en Session Initiation Protocol (SIP)-infrastruktur med hjälp av en molntjänst med särskild inriktning på, små, och medelstora företag (SME) och hemmanvändare. Denna forskning fokuserar att använda en prisvärt, billig, flexibel, med program inbyggd hårdvara (speciellt en Linksys WRT54GL trådl0s router för den lokala platsen med ett anpassat operativsystem DD-WRT). Tanken är att minska energiförbrukningen på, den lokala platsen till mycket låga nivåer genom att undersöka vilka funktioner, som kan köras på, ett molntjnst snarare än på, den lokala platsen.

Avhandlingen presenterar utformningen av ett IP-telefonisystem på, den lokala platsen med ett lågt strömbehov och på, molntjänsten. Ett antal olika användningsförhållanden och önskvärda egenskaper är beskrivna. Metodiken för att genomföra en rad experiment definieras för att utföra stresstester och för att utvärdera designen av IP-telefonisystem med ett lågt effektbehov. I försöken experimenteras den totala energiförbrukningen av den lokala platsen under olika konfigurationer, VPN-länkens samtalskapacitet, QoS-mätning för VoIP-samtal, Session Request Delay (SRD) och Registration Request Delay (RRD).

Resultaten från dessa experiment visar att det finns en potential för betydande energibesparing när du använder den föreslagna designen för en IP-telefoni system.

Acknowledgements

I offer my truthful gratitude to my supervisor Professor Gerald Q. Maguire Jr, who has supported me throughout my thesis with his patience and knowledge whilst allowing me the room to work in my own way. I attribute the level of my thesis to his encouragement and effort. Without him this thesis, too, would not have been completed or written. One simply could not wish for a better or friendlier supervisor.

Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

***NIX**	Unix based OS
1xx	SIP's Informational Responses
1xx	SIP's Informational Responses
2xx	SIP's Successful Responses
3GPP	3rd Generation Partnership Project
3xx	SIP's Redirection Responses
4xx	SIP's Client Failure Responses
5xx	SIP's Server Failure Responses
6xx	SIP's Global Failure Responses
AAA	Authentication, Authorization and Accounting
ACK	Acknowledgment
AES	Advanced Encryption Standard
ALG	Application Layer Gateways
API	Application Programming Interface
AP	Access Point
ARM	Acorn RISC Machine
ATA	Analogue telephone adapter
B2BUA	Back-to-Back User Agent
BSD	Berkeley Software Design OS
CD	Compact Disc
CLI	Command Line Interface
CO_2	Carbon dioxide
CODEC	coder-decoder
CPE	Customer premisses equipment
CPU	Central Processing Unit
DAC	Digital to Analogue Converter
DD-WRT	DresDen WRT
DHCP	Dynamic Host Configuration Protocol
DH	Diffie-Hellman
DNS	Domain Name System
DSLAM	Digital Subscriber Line Access Multiplexer
DSL	Digital Subscriber Line
DTMF	Dual-Tone Multi-Frequency
DVD	Optical disc storage

DiffServ/TOS Differentiated Services over Type of Service field in the IP

EEE Energy Efficient Ethernet

EPSU External Power Supply Unit

EU European Union

FIB Forwarding Information Base

GHz Gigahertz

GPIO General Purpose Input Output ports

GPL GNU General Public License

Gtalk Google Talk

H.323 ITU-T

HDMI High-Definition Multimedia Interface

HD High-Definition

HZ hertz

ICE Interactive Connectivity Establishment

IEEE Institute of Electrical and Electronics Engineers

IETF Internet Engineering Task Force

IGMP Internet Group Management Protocol

IPT IP Telephony

IPsec Internet Protocol Security

IP Internet Protocol

ITU-T ITUs Telecommunication Standardization Sector

ITU International Telecommunications Union

IVR Interactive Voice Response

IaaS Infrastructure as a Service

Intel VT-x Intel's Virtualization Technology

KVM Kernel-based Virtual Machine

L2 OSI Layer 2

L3 OSI Layer 3

LAN Local Area Network

MAC Media Access Control

MB Megabyte

MD5 Message-Digest Algorithm

MGCP Media Gateway Control Protocol

MIB Management Information Base

MIKEY Multimedia Internet KEYing

MIPS Million Instructions Per Second.

MIPS32 Microprocessor without Interlocked Pipeline Stages (32bit)

MOS Mean Opinion Score

MPEG4 AAC-LD MPEG-4 Low Delay Audio Coder

Mbps Mega bits per second

NAT Network Address Translator

NIST National Institute of Standards and Technology

NMS Network Monitoring System

OSI Open Systems Interconnection

OS Operating System

P2P Peer-to-Peer

PBX Private Branch Exchange

PC Personal Computer

PE Provider Equipment

PHY PHYsical layer of the OSI model

PSTN Public Switched Telephone Network

PTS Swedish Post and Telecom Agency

PT Payload Type

PaaS Platform as a Service

QoS Quality of Service

RADIUS Remote Authentication Dial In User Service

RAM Random Access Memory

RFC Request for Comments

RIB Routing Information Base

RTCP RTP Control Protocol

RTP Real-time Transport Protocol

Rx Receive

S/MIME Secure/Multipurpose Internet Mail Extensions

SAP Session Announcement Protocol

SA Security Association

SA Security Association

SDRAM Synchronous Dynamic Random-Access Memory

SD Secure Digital

SIP Session Initiation Protocol

SLA Service Level Agreement

SME	Small to Medium sized Enterprises
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SOHO	Small Office Home Office
SON	Self Optimizing Network
SRTP	Secure Real-time Transport Protocol
SSH	Secure Shell
STUN	Simple Traversal of UDP through NAT
SaaS	Software as a Service
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
TLS	Transport Layer Security
TURN	Traversal Using Relay NAT
TWH	Terra watt hour
Tx	Transmit
UAC	User Agent Client
UAS	User Agent Server
UA	User Agent
UDP	User Datagram Protocol
UMS	Unified Messaging System
UPnP	Universal Plug and Play
URI	Unified Resource Identifier
USB	Universal Serial Bus
VCR	Video Cassette Recording
VDSL2	Very high speed Digital Subscriber Line 2
VLAN	Virtual LAN
VM	Virtual Machine
VPN	Virtual Private Network
WAN	Wide Area Network
WLAN	Wireless Local Area Network
WPA	Wi-Fi Protected Access
WRT54GL	Linksys WRT54GL V1.1
Wh	Watt hour
WiFi	Wireless Fidelity
WoL	Wake-on-LAN
W	Watt
XML	Extensible Markup Language
ILBC	Internet Low Bit Rate Codec
NS-2	Network Simulator

1 Introduction

Increasingly, businesses and homes today receive their phone service through the Internet in the form of Voice over IP (VoIP), instead of local analogue or ISDN telephone lines. Companies are using their internal local and wide-area network infrastructure to replace legacy enterprise telephony networks. The benefits of this migration towards VoIP are numerous, starting from the fact that a single network is utilized in order to carry voice, video, and data, thus achieving convergence. An additional advantage is that using VoIP reduces costs (for both long distance and local calls) and empowers users[9]. All of this technology is possible because of the Voice over IP (VoIP) family of technologies. One of the main VoIP enablers is the session initiation protocol SIP.

SIP is the Internet Engineering Task Force (IETF) signalling protocol (see RFC 3261[10]) used for presence, messaging, VoIP, audio/video conferencing, and events notifications. This protocol is for IP-based real-time communications what HTTP is for the Web. SIP manages call set-up, routing, authentication, and sending messages between endpoints within an IP domain. Once a session is established, transport protocols such as the Real-time Transport Protocol (RTP) actually send the digitized and encoded voice data between endpoints[11]. (This is described further in section 2.3.2 on page 13.) Today most of the large telecommunications operators use VoIP to carry international traffic, as the transmission of traffic over IP-based networks yields tangible cost savings [11]. This demonstrates part of the success of VoIP.

The Swedish Post and Telecom Authority's latest report on the telecommunications market[12] states that subscriptions for VoIP in June 2011 had increased 19% since the corresponding time in 2010. The latest report from In-Stat forecasts that VoIP penetration among US businesses will increase rapidly over the next few years, reaching 79% by 2013, compared to 42% at the end of 2009[13].

While the VoIP market has grown along with all the other parts of the information and communication technology area, an issue that arises and relates to any technology that uses power, is how to minimise power consumption. One way to minimise power consumption is by making all of the systems smarter to enable them to offer their service(s), but with a much lower overall power consumption, thus reducing carbon emission and cost. Today saving power is a hot topic all over the Internet. For some users the focus is on longer battery life, for others it is reducing the problem of removing heat from their data centre. A major goal is to reduce the negative impact of computing on our planet.

Lanzisera, et al. [14] show that office building network switches and residential network equipment are two of the largest categories of energy use, consuming 40% and 30% of the total (amount of energy for networking equipment) respectively. Their report concludes that the network equipment consumes about one percent of a building's electricity and if left unchecked, this situation will lead to exponential increases in emissions and energy consumption, on the order of six percent annually. Their study points out that an office building or home can decrease its energy usage by more than ten percent. This shows

that enterprises and home users need to use network equipment that is energy efficient in order to contribute to the overall desired decrease in emission of carbon and at the same time building owners and home owners can save money by reducing the amount of energy consumed by their networking equipment.

In one of their technical reports concerning energy efficiency for SMEs and homes, Google [15] concludes based upon their own metrics and analysis, that the cloud can deliver a high-quality, reliable, and useful service at a much lower energy cost than other methods. Services such as file storage, calendaring, teleconferencing, voicemail, chat, and document management all can enjoy these energy economies. They compare the carbon emission impact of a single service, such as email, hosted locally in the businesss premises or in the cloud. Their findings show that if a small office of 50 people choose cloud email service over a locally hosted server there can be an annual per-user power savings of up to 170 kWh and a carbon footprint reduction of up to 100 kg of CO_2. Larger organizations show smaller, although still impressive efficiency gains.

There are not many specific research projects, public awareness, or proposed solutions on the topic of how green a VoIP based on SIP technology can be or what can be done in order to improve such a solution. In [16], Baset, et al. reflect on the problem of energy efficiency of existing VoIP systems, by measuring the power consumption of the existing solutions and by identifying the main factors that could be addressed in order to lower the power consumption of such systems. However, their research focuses on operators with the common hardware architectures that are used to provide a commercial VoIP infrastructure. (See the related work in section 3.1 starting on page 53.)

1.1 The Problem

The question that this thesis project addresses is how can we design and implement an energy efficient secure, and robust VoIP system that utilizes a SIP infrastructure assisted by a cloud service, specifically focusing on SMEs and homes by using commodity hardware, open source software, and existing networking techniques?

SMEs are divided by the European Union into three categories: medium-sized (<250 employees), small (<50 employees), and micro(<10 employees) [17]. In 1995, an average European household contained 2.5 people. This is expected to decrease further as the number of one-person households increases from 30% in 2000 to 36% by 2015 [18].

This master thesis project implements a VoIP infrastructure with part of the infrastructure running on a SIP server on a low power commodity processor assisted by a cloud service in order to provide a full featured IP telephony solution. This research investigates various techniques of how to reduce the local site's power consumption to very low levels by examining what functions can be done in a cloud service, rather than at the local site. Examples of such services include voicemail, conferencing, e-mail server, etc. This research focuses on using inexpensive, flexible, commodity embedded hardware (specifically a Linksys WRT54GL wireless router for the local site with a customized operating

system, specifically DD-WRT, see section 2.10 on page 41). This system is based on an MIPS32 chip-set architecture that requires low power to operate. Additionally, this platform supports open source development and debugging tools for implementation of a classic *NIX based operating systems which makes it ideal for embedded system experimentation. The main issues that will be discussed concern how to customize and tune the WRT54GLs hardware and software in order to turn it into a one box solution (i.e., treat it as an integrated networking device). This device will support both VoIP and the local site's network infrastructure by offering features including a WAN gateway, LAN switch, WLAN access point, DHCP server, NAT, DNS server, local firewall services, and other more advanced features such as VPN connectivity, SNMP, RADIUS authentication – while at the same time being energy efficient. Another important aspect is to define and examine VoIP usage scenarios and needs for features in the context of micro SMEs (< 10 employees) and homes, in order to customize the system according to the user's requirements. Figure 1 represents the overall architecture and context for the problem area of this thesis.

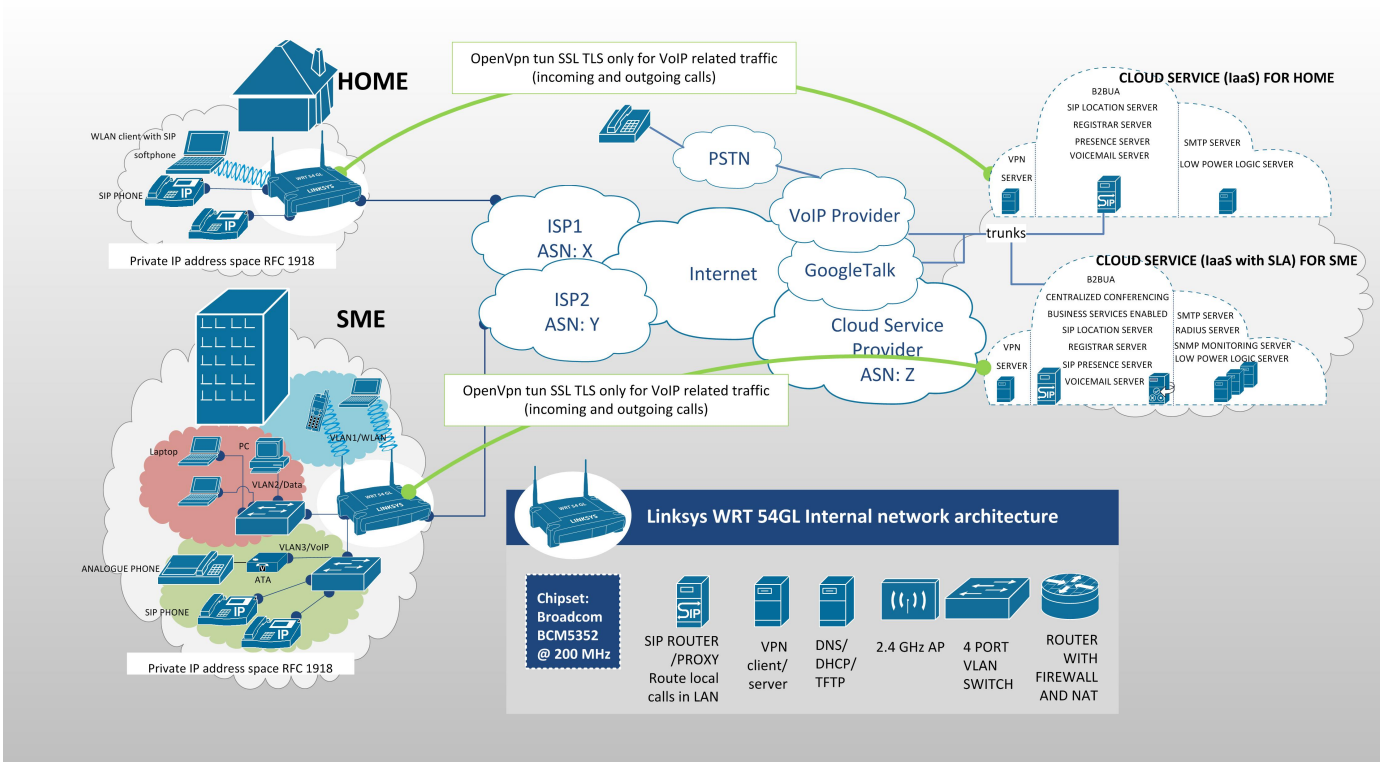


Figure 1: Overall architecture for a SIP based IP telephony solution. This architecture defines the problem scope of this thesis.

1.2 Thesis outline

This thesis is organized and structured linearly. The earlier chapters provide the necessary knowledge for understanding the rest of the thesis. It is strongly

recommended that the reader reads the earlier chapters (to acquire background knowledge) in order to understand the design and methodology of the resulting experiments.

In **Chapter 2** we thoroughly describe the background knowledge and relevant technology in the context of the problem area. This chapter describes VoIP technology using SIP, the cloud services for the cloud site, and the low-power hardware and software technology for the local site. Through reading this chapter the reader should acquire sufficient knowledge to understand the rest of this thesis. **Chapter 3** presents a sample of related work, in order for the reader to gather a sense of what has been done by others in the problem area of low-power IP telephony system for the SMEs and homes. **Chapter 4** describes the design of an IP telephony system for SMEs and homes (by defining the features and services of both the local and cloud site through example call scenarios). These scenarios serve as guide for configuration of the test-bed. In **Chapter 5** the methodology for conducting each of the experiments is explained along with the related performance metrics that are to be measured. **Chapter 7** presents our results, findings, and related data analysis. A conclusion is presented in **Chapter 8**. Finally, we suggest some future work in **Chapter 9**.

2 Background

In order for the reader of this thesis to have the necessary background knowledge, that is required to understand the problem area and the details of the implementation, this chapter describes the mechanisms of the technologies that are relevant to this thesis project. First we will give a general overview of a VoIP architecture, than we will describe SIP and its components, the media session (CODECs, RTP, RTCP, RTSP, SDP), security in a SIP based network, NAT traversal issue for SIP and the media streams, SIP services that could be used in the context of homes and SME users, the cloud technologies and services that are appropriate for homes and SME users, and the network services that are needed in the home and SME environments. At the end of this chapter we will look at low power and low cost hardware architectures, the open-source software for these devices, and the technical details of the Linksys WRT54GL. All of these topics will be related to the overall architecture that was shown in Figure 1.

2.1 Overview of VoIP Architecture

A typical VoIP ecosystem consists of: (a) signalling protocols for setting up a connection and (b) media flow protocols once the session has been established. The protocol stacks used for these two activities are shown in Figure 2.

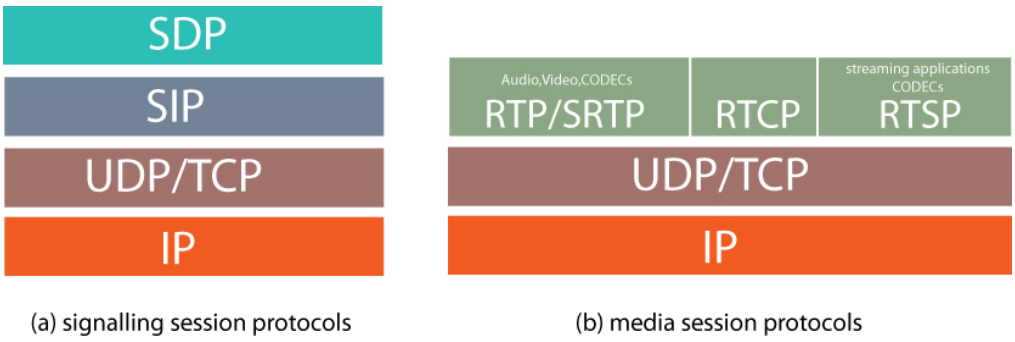


Figure 2: VoIP protocol stacks

Today the main standard for signalling is SIP (this is further described in section 2.2), which operates in the application layer independently of the transport protocol. After a session has been negotiated and established with SIP and the session description protocol (SDP) (this is described in section 2.3.5), another set of protocols is used to transfer the actual media packets (these protocols are described in section 2.3). The challenge for these media flow protocols is to obtain good voice quality across the network by minimizing the effects of packet loss and jitter. The SIP user agents can also potentially use echo cancellation techniques to minimise the effects of acoustic echo. The real time protocol (RTP) (described in section 2.3.2) is used to transport the actual voice and video packets on top of UDP. The real time control protocol (RTCP) (described in 2.3.3) is a reporting mechanism for the associated RTP stream. RTCP reports packet loss, jitter, etc. The real time streaming protocol (RTSP) (described in section 2.3.4) is used to control the real-time delivery of stored

data (for examples playing a voice or video message). The session announcement protocol (SAP) is a session description format used to advertise multicast session information, such as the multicast address to be used for a session and the start time of this session.

2.2 SIP

SIP is an application-level signalling protocol responsible for initiation, modification, and termination of communication sessions such as voice and video calls over IP. The sessions can involve two or more parties with multiple media streams. The structure of SIP incorporates elements from the Hypertext Transfer Protocol [19] and the Simple Mail Transfer Protocol [20]. Its text-based nature makes SIP a highly extendible, easily administered, and easily debuggable protocol which can be and is used in a plethora of IP services, most famous amongst which is VoIP. Unlike the H.323 ITU-T recommendation which wraps everything up in a tightly protocol-and-service defined suite, SIP is just a single component of a modular VoIP implementation. RFC 3261 defines six SIP request (known as methods, see Table 1): INVITE, ACK, BYE, CANCEL, REGISTER, and OPTIONS. Other requests as INFO, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, MESSAGE, and PUBLISH are defined in separate RFCs or Internet drafts[21]. In SIP the peers in a session are called user agents (UAs). A UA can function in one of the following roles:

User-agent client (UAC) A client application that initiates a SIP request.

User-agent server (UAS) A server application that contacts the user when a SIP request is received and that returns a response on behalf of the user.

Typically, a SIP endpoint is capable of functioning as both a UAC and a UAS, but functions only as one or the other in a given transaction. Whether the endpoint functions as a UAC or a UAS depends on whether that user agent initiated the request or is responding to a request.

Table 1: SIP generic message types

SIP Message type	Description
INVITE	The caller announces his IP address and ports along with the audio/video codecs he supports. This message initializes the call setup
100 Trying,180 Ringing	Provisioning status messages which inform the caller of the session setup progress
200 OK	The called party is ready to receive, including its IP address and ports and the codec chosen among the available ones announced in the INVITE message. Another status message
REGISTER	The user agent attempts to register his URI to the SIP server
ACK	The user agent acknowledges a status message

SIP enables UAs to establish and negotiate a session between them. Session description, resource reservation or prioritization, transport of the actual media content, and device control rely on protocols other than SIP. As a result, SIP is transport-protocol independent, which makes it future-proof and highly configurable. SIP uses a three-way handshake session setup procedure: INVITE/200/ACK (see Figure 3) if the setup is successful or INVITE/4xx-5xx-6xx/ACK for a failure. SIP is supplied with several security mechanisms to provide authentication, confidentiality, integrity, and identity checking. More about media security can be found in section 2.4.

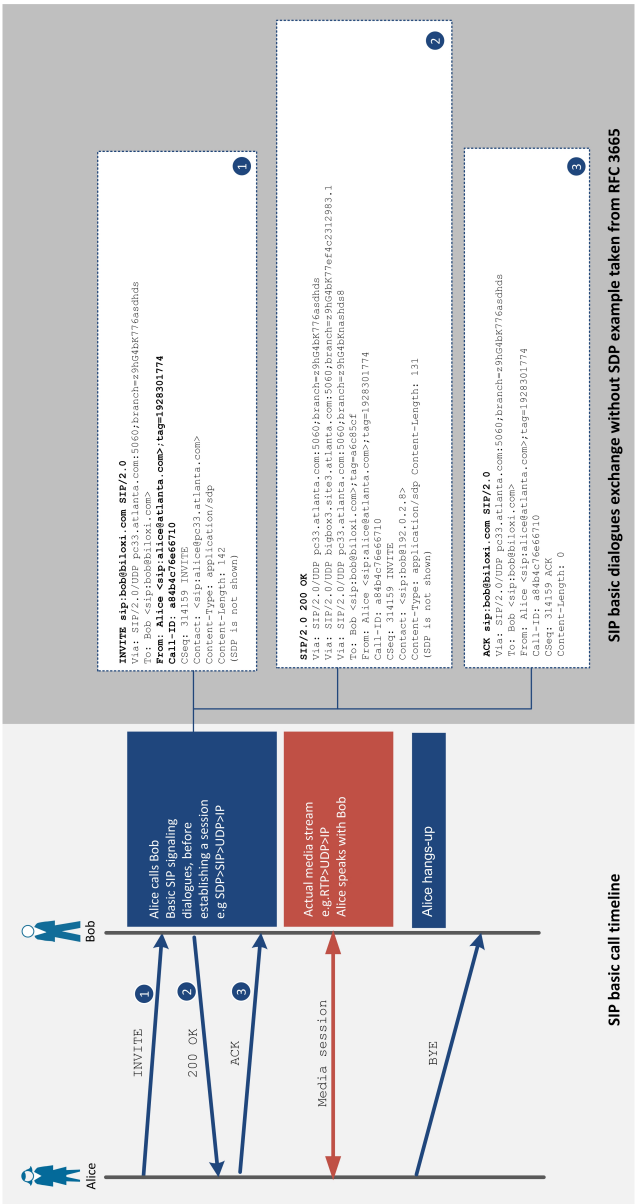
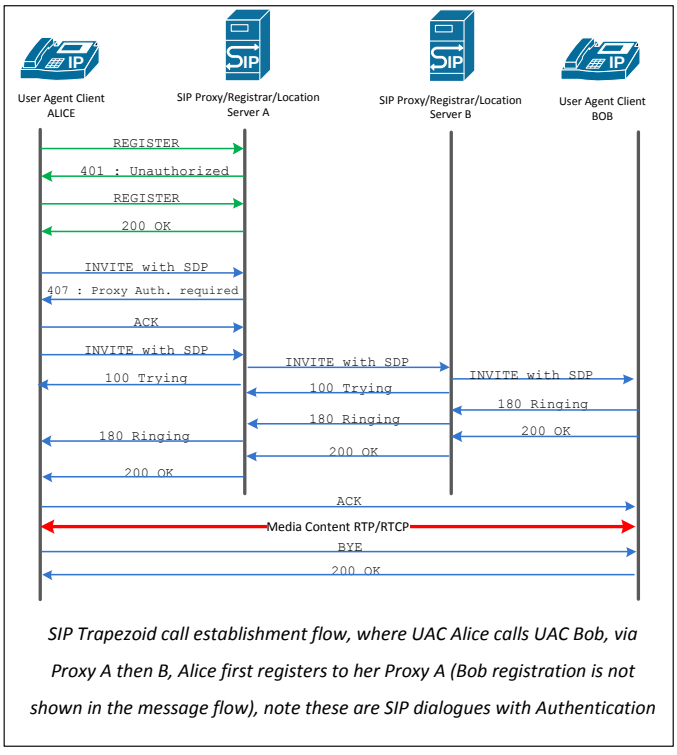


Figure 3: SIP basic call time-line and its signalling dialogues.

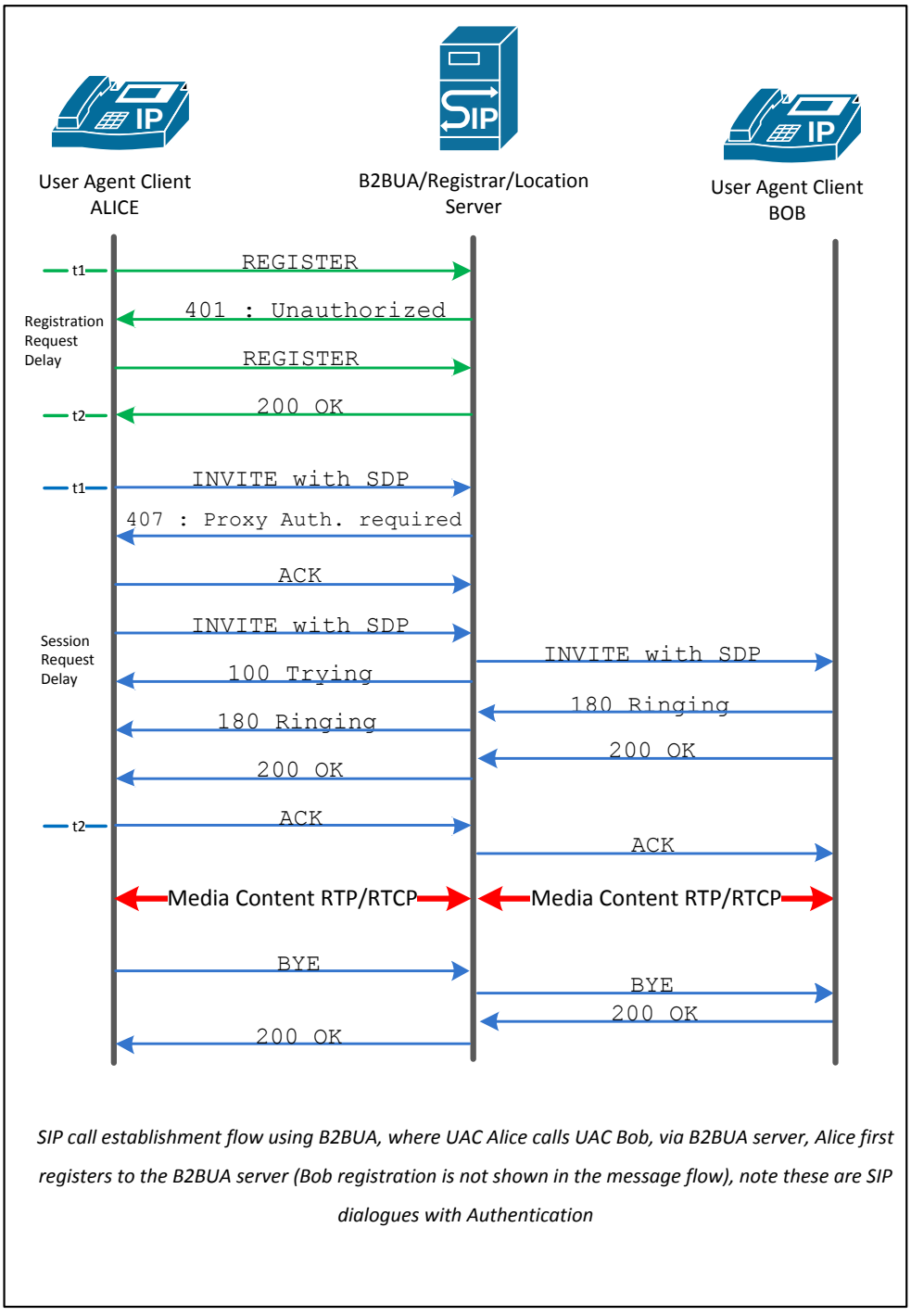
2.2.1 Authentication

SIP comes with an armory of internet authentication schemes. The most widespread among the mechanisms used for authentication between user agents or a user agent and a server is HTTP digest authentication. This scheme is stateless and challenge-based. After receiving a Register request, the user agent/server challenges the initiator of the request with a realm and a nonce to produce a shared secret, such as a username-password pair, to provide assurance of its identity. The realm and nonce are carried in the WWW-Authenticate field of a 401 Unauthorized SIP message (see Figure 4) or a 407 Proxy Authentication Required SIP message in the case of a proxy server. The server should then be able to judge if the authenticated user agent should be authorized or not by making independent use of a separate authorization scheme. The user agent sends credentials using the MD5 hash algorithm. This scheme provides message authentication and replay attack protection. SIP can also perform authentication through certificates, replicating the way web servers and web browsers use certificates for authentication. A user agent can utilize TLS to request a server's certificate and if it is issued by a trusted certificate authority and matches the server the agent wants to talk to, then the server is authenticated. There is also a provision for self-signed certificate mechanisms. Similar to the web case, the server can also use a certificate to authenticate the client, thus achieving mutual authentication.



SIP Trapezoid call establishment flow, where UAC Alice calls UAC Bob, via Proxy A then B, Alice first registers to her Proxy A (Bob registration is not shown in the message flow), note these are SIP dialogues with Authentication

Figure 4: SIP, call establishment flows: SIP Trapezoid



SIP call establishment flow using B2BUA, where UAC Alice calls UAC Bob, via B2BUA server, Alice first registers to the B2BUA server (Bob registration is not shown in the message flow), note these are SIP dialogues with Authentication

Figure 5: SIP, call establishment flows: SIP B2BUA

2.2.2 Confidentiality

SIP can protect its dialogues using encryption on a hop-by-hop or end-to-end basis. TLS encryption is deployed for transport layer encryption over TCP. As a transport mechanism, TLS is specified on a hop-by-hop basis, but a user agent can never be sure that TLS is used by every hop end-to-end. Consequently, TLS is most appropriate when non trusted hosts *a priori* try to establish a session in a hop-by-hop security scheme. IPSec on the other hand operates in the network layer and provides a shielded alternative to traditional IP. Unlike TLS, IPSec is mostly used between endpoints that share mutual trust between them, usually at the operating system/kernel level of hosts. For this reason, its presence or absence cannot easily be known by SIP. IPSec can be also used in a hop-by-hop basis, but this is a rather arduous to configure approach.

2.2.3 Integrity

The recipient of a SIP message needs to be certain that the message is unmodified. For that purpose SIP uses the integrity mechanisms of digital signatures or secured hashes. S/MIME signatures can also be used together with certificates to achieve integrity protection. In this case the recipient is required to know the sender's public key in order to verify the signature. Secure SIP, which implements TLS over every hop of the SIP path, can also provide integrity, while a digest can integrity-protect the SIP message.

From an architectural standpoint, the physical components of a SIP network can be grouped into two categories: clients (endpoints) and servers.

2.2.4 SIP clients

SIP phones in general, can act as either UAS or UAC. They can be softphones (for example running on a PC) IP hardphones which have phone capabilities installed and can initiate SIP requests and respond to requests.

Gateways, provide call control. Additionally, gateways provide many services, the most common being a translation function between SIP conferencing endpoints and other terminal types. This function includes translation between transmission formats and between communications procedures. In addition, the gateway can translate between audio and video CODECs and performs call setup and clearing on both the LAN side and the circuit-switched network side (in the case of a gateway between an internet and a public switched telephony network or mobile telephony network).

2.2.5 SIP servers

As Camariillo, et al. described in [10], depending upon how the SIP server handles incoming requests, originated from a user agent or proxy, the server can act as a:

SIP proxy The server forwards the received requests to another location (see Figure 4). Apart from its primary responsibility which is to perform call routing control, a proxy

server can also be engaged in authorization or policy enforcement and to some extent even network security. An example of an open-source implementation of a SIP proxy server and router is openSIPS [22].

Redirect server Rather than forwarding the received request, the server in this case responds with a 3xx redirection response which directs the source that initiated the request to contact a different URI. This type of server is usually incorporated in one of the many implementations of SIP servers, proxy, or a back-to-back user agent (B2BUA).

Registrar server This type of SIP server receives SIP registration requests. The user agent information is updated in a location database for the domain this SIP registrar handles, usually by a location service as described in [19]. This server is usually incorporated in one of the main implementations of SIP servers, proxy, or B2BUA.

B2BUA The Back-To-Back user agent resides between the endpoints of a call splitting the communication channel into two legs and interposing its own signalling (see Figure 5). In the caller leg it acts as an user agent server (UAS); while in the called party leg it acts as a UAC. Examining the INVITE message chain of events, the B2BUA on the caller side will receive the INVITE, process it, and answer with a 200 OK for a success or a 4xx / 5xx/ 6xx in case of failure while acting as a UAS. On the other side, it will send an INVITE message to the destination endpoint, receive the 200 OK or 4xx /5xx /6xx SIP response message, and acknowledge it, while playing the role of a UAC. Since all flows pass through the intermediary B2BUA server, this server is able to maintain complete call states and perform real-time accounting and call control (for example, in order to implement value-added features) A typical B2BUA server implementation is Asterisk. Asterisk can also act as full featured IP PBX and media gateway [23].

2.3 Media session protocols

In this section of our thesis project we will introduce the protocols that are used to carry and control the audio (and video) streams once a SIP session has been established between two or more endpoints. Understanding these protocols will give the reader an understanding of how media (such as actual voice packets) are carried over an IP network.

2.3.1 CODECs

As we previously stated, SIP is just a signalling protocol that negotiates, establishes, and tears down media sessions. The media session on its own utilizes

various protocols, each of which plays a specific role to provide a hopefully reliable and robust audio/video service experience to the user(s). The most important component in the media stream processing is the coder-decoder or CODEC. The coder part of the CODEC takes uncompressed digitized audio (or video) and applies an agreed algorithm in order to encode the audio (or video) signal in such a way that the decoder can recreate the previously encoded signal. The decoded audio (or video) signal is passed to a digital to analogue converter (DAC) and rendered as audio (or video). Quite often the CODEC supports compression in order to reduce the number of bits per unit time that have to be sent over the network. In this case the decoder will uncompress the data before passing it on to the DAC. It is important that the endpoints of a media session agree to use the same CODEC. Achieving this agreement is a process known as capabilities exchange. This capabilities exchange is provided by the SDP protocol (this is further explained in section 2.3.5) within the SIP signalling session. Most VoIP phones today support multiple CODECs in order to be able to negotiate the correct one while taking into account the available bandwidth and the desired audio or video quality. CODECs are divided into narrow band and wideband CODECs. Narrow band CODECSs come from the legacy of PSTN[1] and are generally designed to provide a 3kHz low-fidelity audio encoding while wideband CODECs take the advantage of the converged IP based networks by exploiting the greater available bandwidth in order to provide high definition audio that can range from 7kHz up to 20kHz in fidelity with one or more audio channels. When comparing CODECs (see Table 2) there are a lot of trade-offs to be considered. Some of these trade-off are described in [6] as:

- Audio bandwidth: higher is better
- Data rate or bit rate: fewer bits per second is better
- Audio quality loss: low is better
- Psychoacoustics: a higher mean opinion score (MOS) is better
- Processing power required for the CODEC: less is better
- Processor memory required: less is better
- Is the CODEC openly available? open is better
- Inserted delay (audio latency caused by the algorithm): less is better
- Resilience (how insensitive is the CODEC to lost or corrupted packets: more is better
- ITU standards-based? (standardized by the International Telecommunications Union - yes is better in some contexts)

[1]Some of the CODECs are also associated with digital media transmission, such as used in military radios.

Table 2: A comparison of several VoIP CODECs, parts recreated from [6]

CODEC	BW(kHz)	BitRate (kbps)	Processor (MIPS)	MOS
G.729	3.3	8	10.8	3.92
G.711	3.3	56	0.6	4.1
iLBC	3.3	15.2	15	4.14
G.722.2	7	10	38	-
G.722.1	7	24	5.5	-
G.722	7	64	14	-
G.719	20	32	18	-
MPEG4 AAC-LD	20	32	36	-

The bandwidth needed to send audio across an Ethernet can be calculated by combining the Bitrate x Sample size + layer two overhead from Ethernet + layer 3 overhead from IP + layer 4 overhead from (UDP+RTP). For audio the required bits rates are quite low - even for high quality stereo audio content.

2.3.2 RTP

After the analogue audio has been sampled, usually at 8000 discrete signal measurements per second, and quantified, then it is encoded and optionally compressed. This encoded audio is placed into an RTP payload [1]. RTP can carry a variety of real-time data (such as audio and video), thus providing a basic transport service for these types of data. RTP is usually transported on top of UDP, because we want a transport protocol that will support real-time applications. RTP on its own provides no mechanisms to ensure timely delivery, but RTCP (further explained in section 2.3.3) can be used by applications to know the quality of the RTP transmissions [1]. RTP provides sequence numbers in order to ensure orderly delivery where the initial value of this sequence number is chosen randomly. RTP supports mixing of streams, e.g. multiple audio streams can be mixed in the case of an audio conference, or an RTP stream can be translated by an intermediate device from one encoding to another while maintaining the synchronization source identifier. This translation process could be done by a SIP B2BUA server or by an RTP translator (further details about an RTP translator can be found in [1]). Note that RTP can synchronize multiple streams, e.g. audio with a video stream [24]. Figure 6 shows an RTP packet header with its fields [1]. The fields in the RTP header have the following meanings:

V	version number (current version is 2)
P	whether payload contains padding
P	whether header extension is present
CC	number of CSRC identifiers
M	profile specific marker for frames
PT	identifies the RTP payload type, e.g. G721 Audio

sequence number	incremental numbers used to determine a packet sequence and used by the receiver to detect packet loss
timestamp	sampling instant of the first octet in the RTP data packet
SSRC	uniquely defines the multimedia source, the initial value is determined randomly.
CSRC	used by mixers to identify sources in a mix. (0 to 15 sources)

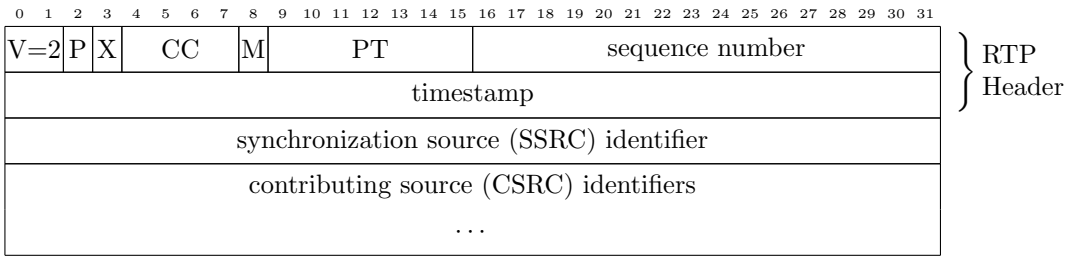


Figure 6: RTP packet header with its fields as in [1]

2.3.3 RTCP

RTCP is a companion protocol for controlling an RTP session. Together RTP and RTCP are used to create an RTP/RTCP session. RTCP provides end-to-end monitoring of quality of service (QoS). RTCP is responsible for three main functions:

- Feedback on performance of the application and the network
- Correlation and synchronization of different media streams generated by the same sender (e.g. combined audio and video)
- RTCP can be used to provide the identity of the sender for display via a user interface

Usually several RTCP packets are combined and encapsulated in the same UDP datagram (to reduce the overhead due to the UDP and lower layer headers). Figure 7 is an example of an RTCP header that is carried as a UDP payload.

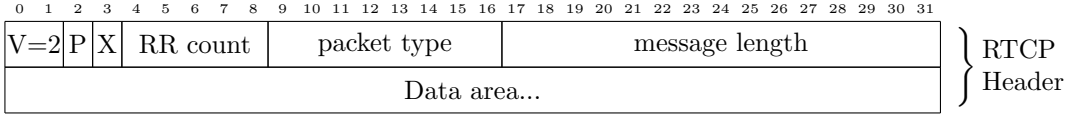


Figure 7: RTCP packet header that is carried as a UDP payload

The fields in the RTCP header have the following meanings:

V	the same version as RTP version
P	whether payload contains padding

RR count The number of reception report blocks contained in this packet
Packet type RTCP packet type (see Table 3)

Table 3: Some of the RTCP packet types as defined in [1]

Packet Type	Acronym	Description
200	SR	Sender report for transmission and reception statistics from active senders (periodically transmitted)
201	RR	Receiver report for reception statistics from participants that are not active senders (periodically transmitted), packet loss and jitter, timing and round-trip estimation
202	SDES	Description of who owns the source, including canonical name
203	BYE	Receiver, indicates end of participation
207	APP	Application specific functions

2.3.4 RTSP

Another protocol that can be used in a VoIP context is the Real Time Streaming Protocol (RTSP). RTSP is defined in RFC 2326 [7]. The protocol provides remote control of a network device. The commands which include pause/resume, fast forward, and rewind. These commands are similar to those of a VCR/CD/DVD remote controller. RTSP is a text-based protocol, similar to HTTP; however, there are new methods and unlike HTTP the RTSP protocol is not stateless, e.g. a streaming server maintains state. If there is an in-band media session, e.g. with RTP/RTCP, then the RTSP messages are sent out-of band over port 544. RTSP can be carried via TCP or UDP. A RTSP server and client can be implemented by using VideoLan[25] which incorporates a media player and a streaming server. Table 4 lists the types of methods in RTSP. An example of a DESCRIPTION message can be seen in Figure 8.

Table 4: Some of the RTSP methods (as described in [7])

Method	Description
SETUP	Causes the server to allocate resources for a stream and start
PLAY	Starts data transmission on a stream allocated via SETUP
PAUSE	Temporarily halts a stream without freeing server resources
RECORD	Initiates recording a range of media data according to the presentation description
DESCRIBE	Starts data transmission on a stream allocated via SETUP

```
============================================================
<title>Twister</title>
<session>
    <group language=en lipsync>
      <switch>
         <track type=audio
          e="PCMU/8000/1"
          src="rtsp://audio.example.com/twister/audio.en/lofi">
         <track type=audio
          e="DVI4/16000/2" pt="90 DVI4/8000/1"
          src="rtsp://audio.example.com/twister/audio.en/hifi">
       </switch>
         <track type="video/jpeg"
         src="rtspu://video.example.com/twister/video">
    </group>
</session>
============================================================
```

Figure 8: Example of RTSP DESCRIPTION message adapted from [2]

2.3.5 SDP

SDP is an important and vital protocol for VoIP. SDP is defined in RFC 4566 [26]. SDP is intended to be used for describing multimedia sessions for the purposes of session announcement, session invitation, and other forms of multimedia session initiation. In conjunction with SIP, SDP provides the backbone for VoIP media session control and establishment. Together SIP/SDP can be used to add and remove media, including media such as file transfers, screen sharing, and shared web browsing, all within a single session, without the user having to be aware of it. The mechanism of SIP and SDP utilize SIP messages to create sessions and carry session descriptions that allow participants to agree on a set of compatible media types and the destination IP address, transport protocol, and port number that will be used. The session descriptions are formatted using SDP. SDP is carried in SIP because on its own SDP is only a format for session description. SDP can also be transported via SAP, RTSP, and HTTP[26]. An SDP session description message includes the following media information:

- The type of media (video, audio, etc.)
- The transport protocol (RTP/UDP/IP, H.320, etc.)
- The format of the media (H.261 video, MPEG video, etc.)
- The remote address for media
- The remote transport port for media

Figure 9 shows an example of a SIP INVITE message, note the SDP format. In the message shown in Figure 9 the string *application/sdp* indicates the presence of SDP in this SIP message. The attribute formats of an SDP message are in the form $< type >=< value >$. Note that there should be no white space between the type (a single character) and the "=" sign. This is an SDP offer message destined to user *test*@130.237.209.244 as part of a SIP INVITE message. The meaning of the session description message in Figure 9 is:

v Protocol version.

o Owner/creator and session identifier.

s Session name, the name of the UA

c Connection information for the media, an IPv4 address 192.168.1.13 (in this example it is a private IPv4 address)

t Duration of the session is active, start and stop time values, here the time is 0 since there was no session yet and this is the initial offer message.

m Media name and transport address. Here the media type is "audio" and the receiving port number is 61634 for RTP audio-video protocol over UDP and 61635 (the next higher odd number) is used for RTCP port. The RTP payload supported types for this session are 97, 8, and 101, where 8 is the G.711 PCMA encoding, 97 and 101 are defined in the "a=" attributes since they are dynamic payload types [27].

a Session attribute. This optional attribute enables the use of new features by application writers. In this offer we see that the application uses a SPEEX CODEC with an 8000Hz sampling rate. This is a narrowband CODEC. The FMTP value of 101 indicates that dual tone multifrequency (DTMF) events are supported[2]. The attribute *sendrecv* means that the user is able to both send and receive media.

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
sip:test@130.237.209.244 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.13:34490;branch=z9hG4bK-d8754z-18051829140fb9dc-1---d8754z-;rport
Max-Forwards: 70
Contact: <sip:102@130.237.209.244:34490>
To: <sip:test@130.237.209.244>
From: "102"<sip:102@130.237.209.244>;tag=24b038ec
Call-ID: NDNhNTMwYjEzZGQ0NDE5ZTk2YTk2MmJhZDFiM2ZhZDY.
CSeq: 1 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE, SUBSCRIBE, INFO
Content-Type: application/sdp
Supported: replaces
User-Agent: X-Lite 4 release 4.1 stamp 63214
Content-Length: 402
                                                  <-----Here starts the SDP format
v=0
o=- 12974585049522295 1 IN IP4 130.237.209.244
s=CounterPath X-Lite 4.1
c=IN IP4 192.168.1.13
t=0 0
m=audio 61634 RTP/AVP 97 8 101
a=rtpmap:97 SPEEX/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
a=sendrecv
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Figure 9: Example of a SIP INVITE message with SDP format

2.4 Security in a SIP based network

As with any type of network application, VoIP is concerned with attacks on confidentiality, integrity, and availability. Confidentiality threats concern exposing the content of the session between the parties, but could also include exposure of call data (such as telephone numbers dialled or call durations). Integrity threats impact the ability to trust the identity of the caller, the message, the identity of the recipient, or the call record logs. Availability threats jeopardize the ability to make, receive, or maintain a call [29].

The motivation for an attack is based on a variety of factors including stealing

[2]Full details of the DTMF parameters are described in [28].

phone service, stealing long-distance service, impersonation of someone else to commit fraud, eavesdropping to commit fraud, and disruption of service to gain notoriety or for extortion[29].

The major question when supporting end-to-end security is what layer should provide this security, the network layer or some higher layer? The major alternatives are either IPSec (network layer) for all traffic or TLS (transport/application layer) for connection oriented traffic. For the real-time traffic the main alternative today is secure RTP (SRTP) operating at the transport/application layer [30]. Here we will present attacks that are specific to SIP related sessions. Note once again they can be made on the signalling and/or media session. Table 5 summarizes the scenarios for these types of attacks.

In section 2.2 we described the basic protection mechanisms against SIP's signalling session attacks. Media session attacks on RTP can be prevented by utilizing the SRTP. SRTP is described in detail in RFC 3711 [31]. SRTP provides both authentication (to counter integrity attacks) and confidentiality services for the payload being carried by the RTP protocol. The SRTP protocol has been designed to add minimal overhead to the packet size and to minimize the number of key pairs that must be shared between the communicating nodes [29].

Table 5: Potential attacks on a SIP based signalling and media sessions

Attacks	Attack Mechanism	Confidentiality	Integrity	Availability
SIP:				
	Registration Hijacking	X	X	X
	Message Modification	X	X	
	Cancel/Bye Attack			X
	Malformed Command			X
	Redirect	X		X
RTP:				
	RTP Payload	X	X	X
	RTP Tampering			X

When using SRTP, the RTP packets are encrypted by the sender and travel the entire network encrypted until they are decrypted by the receiver. The SRTP approach prevents eavesdropping, modification, and replay of packets. Example of modifying RTP packets could be manipulation of the sequence number and timestamps fields in the header of the RTP packet (explained in section 2.3.2), the packets could be resequenced or made unusable, which will result in an unsatisfactory session.

The keys for encrypting and authenticating of the RTP packets can be derived from a Multimedia Internet KEYing (MIKEY) phase defined in RFC 3830[32]. We can say that what IKE is to IPsec, MIKEY is to SRTP, a difference is they function on different layers. MIKEY supports three different authentication mechanisms: shared key, public key, and signed Diffie-Hellman authentication. Pre-shared secret keys provide one mechanism for MIKEY to generate session keys. The session keys are used to encrypt the messages sent via SRTP

(including both the RTP and RTCP datagrams). The second method of generating session keys relies upon public key cryptography. In this method nodes utilize asymmetric cryptography techniques to generate and exchange a session key. After this, the SRTP session proceeds just as with the preshared secret keys. In the signed Diffie-Hellman method both nodes contribute to the generation of the session key, providing perfect forward security. Secret key cryptographic algorithms are always utilized for the SRTP packets as they need to have a low compute cost in order to minimize the additional latency of the real-time delivery of the media stream.

2.5 NAT traversal

Network address translators (NATs) are devices that modify the IP address and possibly the transport protocol port numbers of IP packets as they are forwarded from one network to another. When an IP packet that originates from a NAT'd network needs to traverse the public Internet, the NAT replaces the local source IP addresses with a globally routable IP address. The benefit of a NAT is that it allows an internet connection with a single IP address to be shared. There are some people who believe that using a NAT also increases security since all the devices behind the NAT are hidden from the public Internet, thus hiding the internal network topology - however, this is largely security through obscurity. NATs and firewalls complicate the SIP signalling and RTP flow for both business and home users. NATs are currently only used in IPv4 networks. In IPv6 networks NATs are not necessary since there is an extremely large number of interface addresses available. Firewalls in an enterprise network usually drop UDP packets, however it is possible to create stateful firewalls for SIP calls, where RTP ports are dynamically opened in order to allow the media session for a limited amount of time (generally the duration of the call). The main problem with NAT is that there is no default mechanism for synchronization between the network and application layer regarding the use of IP addresses. A UA behind a NAT, when sending an `INVITE` message, will put its private address in the `Via:`, `Contact:`, and SDP's `c=` fields [3]. This is shown in Figure 10. The problem with the message shown in Figure 10 (this example is taken from [3]) is that the recipient of the message will not be able to route back a response since there is a private network address in the `Via` header. As per RFC 1918 such a private IP address is never routed via the global Internet. Also future requests will be misrouted because of an incorrect `Contact` header. Most importantly the RTP packets of User B can not be routed to the private IP address specified by User A in the `c=` field for the media in the SDP. The result will be that user B can hear user A, but not the other way around.

```
INVITE sip:UserB@there.com SIP/2.0
Via: SIP/2.0/UDP 10.1.1.221:5060;branch=z9hG4bKhjh
From: TheBigGuy <sip:UserA@customer.com>;tag=343kdw2
To: TheLittleGuy <sip:UserB@there.com>
Max-Forwards: 70
Call-ID: 123456349fijoewr
CSeq: 1 INVITE
Subject: Wow! It Works...
Contact: <sip:UserA@10.1.1.221>
Content-Type: application/sdp
Content-Length: ...
v=0
o=UserA 2890844526 2890844526 IN IP4 UserA.customer.com
c=IN IP4 10.1.1.221
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Figure 10: Example of the NAT problem in a SIP INVITE message

IETF has standardized three protocols to help assist in NAT traversal: Simple Traversal of UDP through NAT (STUN), Traversal Using Relay NAT (TURN), and Interactive Connectivity Establishment (ICE).

STUN is defined in RFC 5389[33]. STUN is a simple protocol that enables a UA to discover if it is behind a NAT and if it is, what is its public IP address is. A UA or a STUN client first contacts a STUN server located in the public Internet, the STUN response tells the STUN client its public IP address and source port base upon the IP packet that the STUN server received with the STUN request. If the sent and received addresses and ports are the same, there is no NAT. If they are different, then there is a NAT between the STUN client and STUN server. STUN enables UAs to correctly calculate all the parts of a SIP and SDP message, including providing a correct public IP address. However, STUN only works if one end-point is behind a NAT and the other end-point is directly addressable via the global Internet.

TURN solves the issue of two or more of the end-points being behind NATs. TURN is defined in RFC 5766[34]. TURN allows a client to obtain transport addresses from a TURN server on the public Internet. Since the TURN server is located in the public Internet, TURN addresses will always be routable. However, the cost is a less than optimal routing path and all traffic has to go both to and from the TURN server.

ICE is defined in RFC 5245[35]. ICE incorporates a mechanism that uses STUN and TURN in a peer-to-peer manner that guarantees that the most efficient routing through NATs will occur. STUN, TURN, and ICE all require support in the UA in order for the traffic to traverse NATs and firewalls.

Another concept that does not rely on a NAT traversal protocol is the use of *Application Layer Gateways (ALG)* or Session Border Controllers for firewall

traversal. An ALG is a SIP and RTP proxy that is trusted by the firewall. All packets that originate from the ALG device are implicitly trusted by the firewall. For NAT traversal the necessary fields are translated in each SIP or RTP packet from their internal value to the appropriate public. However, ALGs break the end-to-end nature of SIP and, as a consequence, break the SIP security mechanisms.

An implementation that does not directly break the end-to-end characteristics of SIP is the use of a B2BUA. However, in some sense the B2BUA does break the end-to-end semantics - but it does so by explicitly inserting a proxy in the middle of the path. Hence a B2BUA can be deployed as an anonymizer to ensure the caller's privacy - at the cost of the B2BUA being able to see **all** of the SIP and RTP traffic.

2.6 SIP Services

In order to create a basic VoIP system for SMEs and homes (as was shown in Figure 1 on page 3) simply providing call establishment is not enough. Legacy telephony features such as call forwarding, parking a call, click to dial, call on redial, conferencing, and voicemail may be desirable. Fortunately all of these services can be reproduced using SIP by various means. In addition to these services SIP can also provide messaging and presence. Additionally, SIP can easily be integrated with many applications. For example an SMTP email server can send a voicemail message as an email attachment, if the callee has an unanswered call or was busy when there was an incoming call to the callee. In this section we will present the technology and mechanisms for implementing these services by using SIP. These services are important, since some of them can be implemented at the local site, while some of them can be implemented in the cloud (as will be explained in section 2.8 starting on page 33).

2.6.1 SIP Business Call Services

SIP offers an extensive portfolio of call services without the need for extensions to the existing standard. In order to compile a set of services to be provided in a SME and home IP telephony system we will consider those services that are frequently used in a business. Based upon this list of services it is easy to derive the home scenario, since we consider it to be simpler.

Registration - is a method that provides localization for the SIP UAC. An UAC registers with a registrar server. The registration service can be with or without authentication. Performing authentication (see Figures: 4 and 5 on page 8) results in a longer SIP dialogue in which an MD5 Digest response is send from the UAC to the registrar[10] [36]. Note that registration only needs to occur when a new UA is to be used or is no longer to be used.

Call on Hold - in this call scenario Alice calls Bob, and then Alice places the call on hold to deal with something other than the session with Bob. Alice then takes the call off hold to speak further with Bob. Later Bob hangs up, terminating the call. The hold feature is unidirectional in nature. However, a UA that places the other party on hold will generally also stop sending media, resulting

in no media being exchanged between the UAs. This is explicitly achieved by using the $a = inactive$ SDP attribute if no media is sent, or the $a = sendonly$ attribute if media is still sent [26] [37] [36]. Note that there is no requirement to send another SDP message when placing a call on hold, although doing so can enable the user interface of the other parties to display that the call is explicitly on hold.

Call Forward - in this call scenario Alice calls Bob, but Bob wants this call to be forwarded to Carol. There can be various reasons for call forwarding, e.g. unconditional, busy, and no answer [38]. If when Alice called Bob he was busy and responded with 480 Temporary Unavailable or he did not respond to the call (resulting in a CANCEL message being issued by the server to Bob), then the call is forwarded to Carol with a new INVITE message [36].

Call Transfer - Alice calls Bob, Bob puts Alice on hold by sending the REFER method to the server or Carol. Now a new session is established between Alice and Carol which results in a BYE message being sent by Bob in order to terminate the session from Alice to Bob (and the reverse session)[36].

Call Pickup and Group Call Pickup - are a rather complex feature in terms of their SIP dialogues. This service allows a user to answer or pickup an incoming call that is ringing on another user's phone through his or her own phone. In this scenario we have Alice, Bob, Carol, and Bill. Alice calls Bob, Bob and Bill belong to the same call group (in this case a group named bbgroup [38]). Bob receives an incoming call, but he is not in the office to answer it, however Bill hears that Bob's phone is ringing so he picks up the call by dialling a command such as $*10* < Bob's extension >$, then a new INVITE will be created with the address $*10* < Bob's extension >$, where $*10*$ is a pick up dialling prefix and $< Bob's extension >$ is the target number to be picked up. If Bob and Bill do not belong to the same group, then the call could not be picked up by Bill [36].Please note that this example uses a numeric dial plan logic for the extensions and the prefixes (which is common to legacy PBX telephony services), however the same principle of operation can be obtained using alpha characters.

Find me or Follow me - enables Alice to be reached on multiple devices at once. For example, Alice prefers to be contacted either simultaneously or sequentially via her office, mobile, or soft phone. Now when Bob calls Alice, the INVITE received by the SIP proxy will be forwarded to all three devices, i.e., the INVITE will be forked to all three devices in the policy and manner that was specified in Alice's user profile at her incoming proxy. When Alice picks up the call from one device, then CANCEL messages will be send to all the other destinations [36].

Music on Hold - serves recorded media to Alice when Alice calls Bob and Bob places Alice on Hold. This is done by Bob sending an INVITE to Alice with the SDP media attribute $a = sendonly$, indicating that he will only send media, but not receive media. Following this Bob sets up an RTP media session between his music media server and Alice. This media server will stream media to Alice until Bob picks up the call from hold with a $a = sndrcv$ media attribute [37] [36]. At this point Bob has to terminate the media stream from the media server

to Alice and provide his own media stream.

Ring Group - is a concept that is similar to the mechanics of the follow me feature. It is a very useful business telephony feature as it provides improved availability. A common usage is for a technical support department. Here Alice, Bob, and Carol are all in the same call group and share an incoming extension. Thus a call to this extension will ring all the UAs associated with the three of them at the same time. When one of these people picks up the call, then a CANCEL is sent to the other destinations [36].

Call Park - is another office feature. Assume that Bill was calling Alice, but Bob picks up a call that was originally intended for Alice. However, Alice has now become available, thus Bob can park the incoming call, by first placing the call on Hold (e.g. by pressing the attended transfer soft button on his SIP hardphone) and sending a SDP message with the media attribute as sendonly. Bill will now hear music at this point (via a music on hold feature - as described above). Next, Bob parks the call by dialling 300 which is a special extension for parking calls (which is an application enabled and configured within the IP-telephony system). An announcement is played telling Bob where (on which extension from a dedicated range of extensions 301-320 for parking calls, configured on the IP-telephony system) the call has been parked (in this case it answers with 301). Next Bob calls Alice and lets her know that there is a call waiting for her on extension 301. Alice then retrieves the parked call by dialling the extension 301. This establishes a media session between Alice and Bill [36].

Message Waiting Indication - is a feature that utilizes the SUBSCRIBE and NOTIFY method [39] [36]. For example, Alice wants to have a visual message indication on her hard phone from the voicemail service if she has any voicemail waiting for her. Therefore, she SUBSCRIBEs to the notifier server provided by the voicemail server. Immediately the notifier will NOTIFY Alice if there is any new voice message waiting in her mailbox. Additionally, if Alice subscribed for a longer period of time, then if there is voicemail placed into her voicemail box during this time, then she will also receive a notification.

Automatic Redial - automatic redial is a nice feature for calling back a party who was busy when you first called them. The idea is that when Alice calls Bob and Bob is busy (indicated by Bob's UA by sending a "486 Busy here" message to Alice). Then Alice can SUBSCRIBE either to Bob's UA or to Bob's incoming proxy server to receive a notification when Bob is no longer busy. Assume that Alice's SUBSCRIBE was sent to Bob's UA. Then Bob's UA immediately sends a NOTIFY message with the "confirmed" attribute to Alice's UA. When Bob is free his UA will send a NOTIFY message with the "terminated" attribute to Alice's UA [36]. Alice's UA will then send a new INVITE to Bob's UA in order to establish a media session. For example, while Bob was speaking Alice was able to continue to work on her report and her phone was "on hook", but when Bob became free both phones will start to ring.

Instant messaging (IM) - is natively supported in SIP. In order to send an instant message a session is **not** required. Therefore Alice can send an IM to Bob through her UA by typing "How are you?" in the IM window. This

message can either be sent to a SIP proxy or directly to Bob's UA depending on the VoIP infrastructure and both Alice's outgoing proxy and Bob's incoming proxy. The sequence of dialogues is very simple, Alice sends a MESSAGE and Bob sends an ACK back with 200 OK [40]. An example of part of such a SIP MESSAGE dialogue is shown in Figure 11.

```
MESSAGE sip:alice@domain.com SIP/2.0
Via: SIP/2.0/TCP bob.domain.com;branch=z9hG4bK776sgdkse
Max-Forwards: 70
From: sip:alice@domain.com;tag=49583
To: sip:bob@domain.com
Call-ID: asd88asd77a@1.2.3.4
CSeq: 1 MESSAGE
Content-Type: text/plain
Content-Length: 18
How are you?
```

Figure 11: An example of part of a SIP MESSAGE dialogue, used for IM

2.6.2 SIP Voicemail and UMS

A Unified Messaging System (UMS) allows users to interact and access email, voice, and fax messages from a single interface.

The voicemail feature enables users to record messages for incoming calls that are not answered within a specified number of rings, or receive busy treatment, or are transferred directly to the voicemail service. The voice mail system also enables message retrieval. RFC 4458 [4] describes a convention for describing a mailbox and voicemail service information in the SIP URI so that vendors and operators can build interoperable systems.

According to Jennings, et al. in RFC 4458 [4] the UMS needs to know what mailbox should be used and the possible reasons for the type of service desired from the UMS. Based on why a call was not answered the UMS can deliver a different greeting to the caller, e.g. in the case of the callee being busy the message might be "Sorry I am busy, please leave a message..." or if the call is not answered or timed out the message might be "I am not home, leave a message...". All this information can be delivered via the existing SIP signalling by the incoming SIP proxy. This SIP proxy will redirect the call to the UMS.

Voicemail utilizes multiple components of a VoIP infrastructure. In the simplest scenario it can be composed of a UAC, the incoming SIP proxy, and UMS. This architecture is based entirely on the standard protocols SIP, SMTP, HTTP, and RTSP[3]. An example illustrating the operation of this open architecture for unified messaging is shown in Figure 12. The mechanism to support voicemail using the semantics of SIP works by encoding the information for the desired service into the SIP Request-URI that is sent to the UMS. Two pieces of information are encoded, the first is the target's mailbox that is to be used and the second is the SIP status code that caused this retargeting and that indicates the desired service. The userinfo and hostport parts of the Request-URI will identify the voicemail service. The target mailbox can be put in the

target parameter and the reason can be placed in the cause parameter. For example, if the proxy wishes to use Bob's mailbox because Bob's phone was busy, then URI sent to the UMS might be:

```
sip:voicemail@example.com;target=bob%40example.com;cause=486
```

Target is a URI parameter that indicates the address of the entity that is to be the target of this voicemail, e.g. Bob's mailbox and the cause is a URI parameter that is used to indicate the variant of the service that the voicemail server should perform when receiving this type of message.[4]. In order to retrieve recorded messages from the UMS the system can recognize and match the From header to the URI target header, if they match then the UMS can identify the specific voicemail and play those messages. Examples of the cause codes and the reason for the redirection are shown in Table 6. An example of redirecting calls when the callee is busy to a Voicemail system is shown in Figures 12 and 13. The process steps are:

- The SIP proxy server forwards the call when busy to voicemail as shown in Figure 13, for the case when Alice calls Bob. Bob's incoming proxy determines that Bob is busy, thus it forwards the call to Bob's voicemail mail box.
- Alternatively the endpoint, one of Bob's SIP phones, could forward the call when this UA is busy to Bob's voicemail mailbox if the endpoint is configured to forward calls when it is busy to the UMS.

Table 6: SIP redirecting reason values for voicemail feature

Redirecting reason	**Value**
Unknown/Not available	404
User busy	486
No reply	408
Unconditional	302
Deflection during alerting	487
Deflection immediate response	480
Mobile subscriber not reachable	503

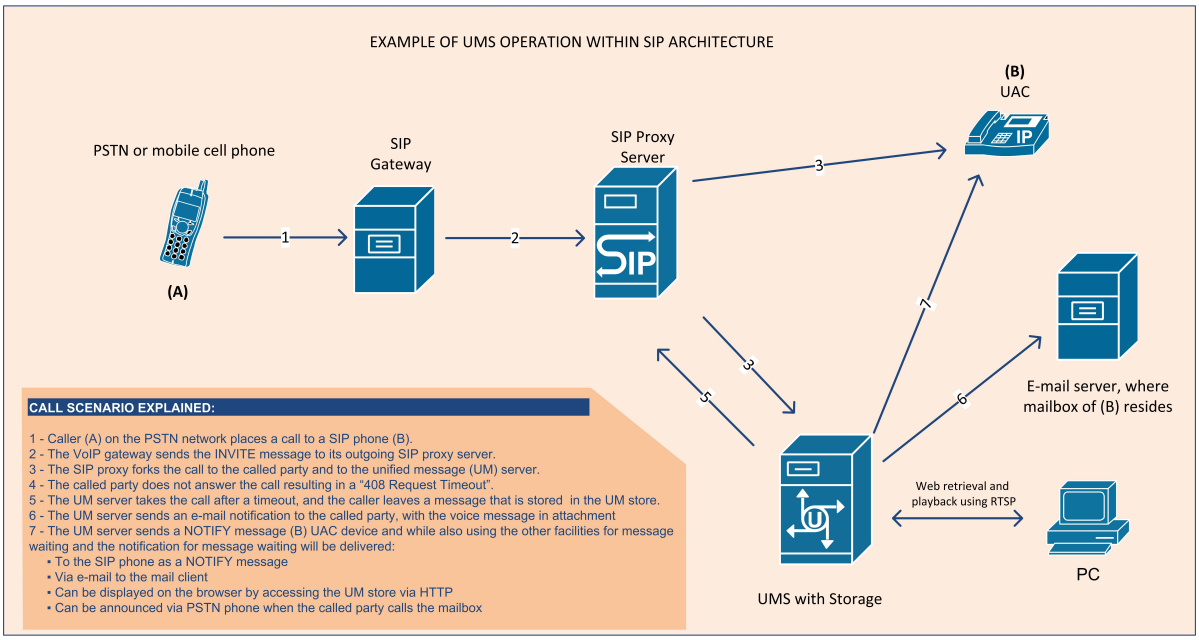


Figure 12: Example of UMS operation within SIP infrastructure, the proxy forwards the caller's INVITE to the callee's voicemail when the call is not answered, the example is based on the example shown in [3].

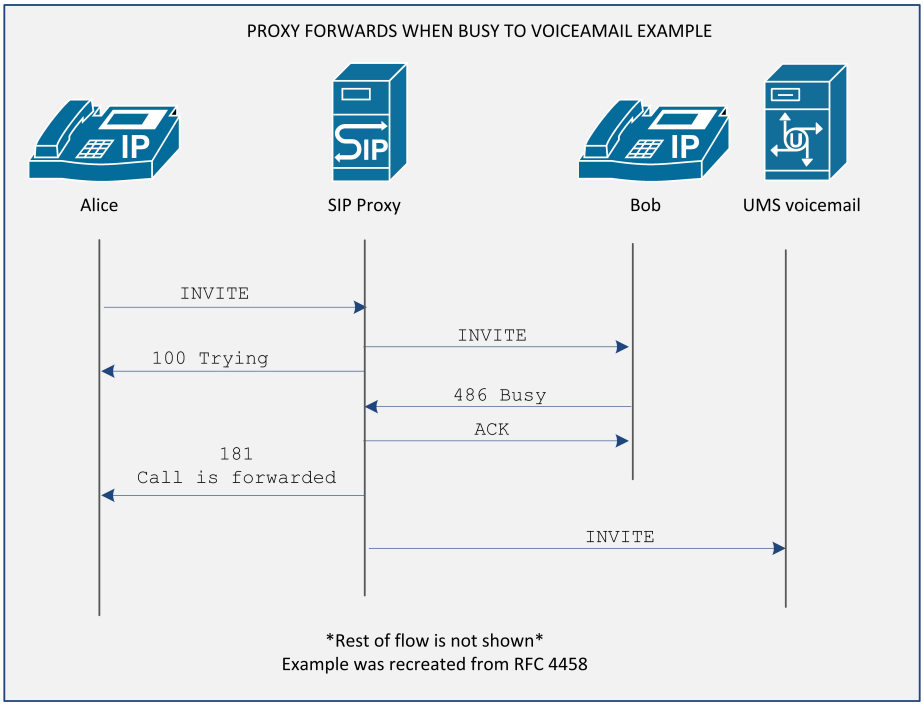


Figure 13: Message sequence diagram, when proxy forwards the call when busy, recreated from [4]

2.6.3 SIP Conferencing

Voice and video conferencing in the Internet is becoming a common application. This is a service that is increasingly useful at home and has long since been a regular telephony feature in the SME environment. The use of voice and video conferencing is growing with the penetration of high bandwidth data connections at reasonable prices, along with the advances in the CODECs. Voice and video communication sessions with multiple participants that consist of one or more SIP sessions, each of which combines SIP dialogues as required for the communication session between the participants to form a conference.

SIP conferencing is more complicated than a normal two party call. SIP can support many models of multi-party communications. Here we will present those that are necessary for deploying conferencing as a service in SMEs and homes. RFC 4353 [5] defines a framework for implementing conferencing with SIP.

According to RFC 4353[5], there are two generic models of conferencing: loosely and tightly coupled. Loosely coupled conferencing makes use of multicast media groups. In the loosely coupled model, there is no signaling relationship between the participants in the conference. There is no central point of control nor a conference server. The list of participants is gradually learned through control information that is passed as part of the conference (for example using the Real Time Control Protocol (RTCP)). Loosely coupled conferences are easily supported in SIP by placing multicast addresses within session descriptions. Using multicasting enables the conference to scale well. In the tightly coupled conference model, there is a central point of control. Each participant connects to this central point of control. This central controller provides a variety of conference functions, and may possibly perform media mixing functions as well.[41]. The disadvantage of this approach is that many ISPs do not support multicast traffic (other than their own IPTV traffic).

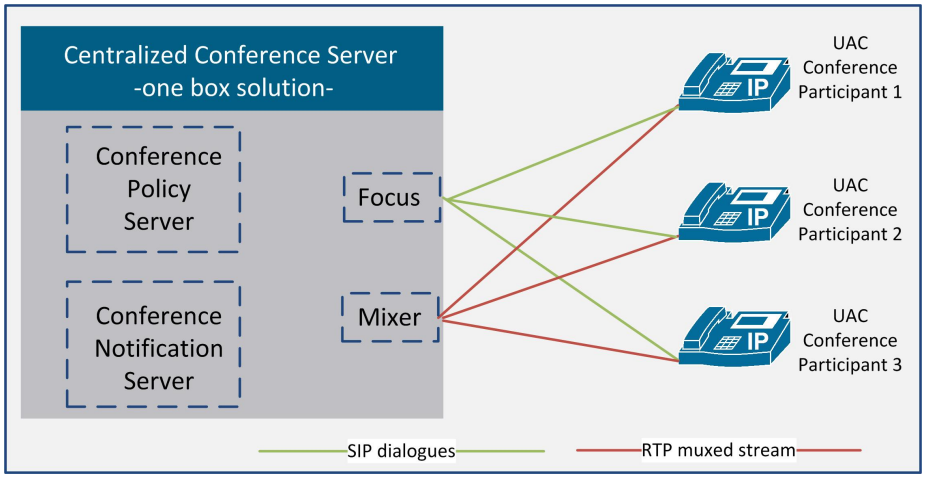


Figure 14: Tightly coupled, centralized, conference server, according to [5].

In the simplest deployment of conferencing as a service, there will be a single

physical server. This service will implement the conference focus, the conference policy server, and any media mixers. This is the classic "one box" solution, shown in Figure 14. The focus takes care of conference control, e.g. creating, modifying, and terminating conferences [42]. Conference policy is managed by the policy server, which configures the media server for each conference. The signaling model of a central configuration is always a star topology. More specific examples of this type of conference server can be found in RFC 5239 [42] which is a framework for Centralized Conferencing.

Another type of centralized conferencing is 3-way *ad hoc* or endpoint mixing which is initiated spontaneously from a 2-party session. For example, users A and B decide to add a third user C by calling user C. When one of the two participants, e.g. B calls C, B invites C to join the existing session, thereby turning the session into a conference. However B, is now responsible for the conference since he invited C. This means that B is responsible for multiplexing the media session between A and C. This approach does not provide a dependable conferencing infrastructure. However, this method can be provisioned by many SIP hard/soft phones. Another possible scenario is that C calls B and asks B to join the existing session that B has with A. [41].

SIP's syntax and semantics require extensions in order to provide tightly coupled conferencing. The extensions for supporting conferences are: addition of a Join header as defined in RFC 3911 [43], and the addition of the re-INVITE and REFER methods. RFC 5850 [44] defines a call control and multi-party usage framework for SIP. RFC 4579 [45] defines SIP call control conferencing for user agents. A variety of conferencing scenarios are described in RFC 4597 [46].

An important aspect of conferencing is policing. This ensures that only participants that know a password can enter a conference, thus achieving privacy based upon authentication and authorization to join the conference. Policing will be described further in Chapter 5.

2.7 SIP URI, DNS, and ENUM: Locating SIP servers

When creating an IP telephony solution using SIP, location, reachability, mobility, portability, proxy server redundancy, and load-balancing are important features. SIP uses SIP Uniform Resource Identifiers (SIP URIs) to represent who to contact or where to place a call. Initially when routing a SIP request a first step is address resolution, a process that computes the mapping between the URI and a specific user at a specific host/address. This process can utilize a DNS naming-authority pointer (NAPTR) and server (SRV) lookups, E.164 number to URI Mapping (ENUM), and location server lookup. Therefore in this section we will explain the mechanisms that SIP and other protocols use in order to provide these features.

2.7.1 SIP URI

A SIP URI is defined in RFC3261 [10], as providing a simple and extensible means for identifying a communication resource. SIP URIs can be placed in web-pages or sent in email messages in order to initiate a communication session

with the resource. Examples of communication resources include:

- a user of an online service,
- an appearance on a multi-line phone,
- a voicemailbox on a voicemessaging system,
- a PSTN number at a gateway service, and
- a group (such as "sales" or "helpdesk") in an organization.

A URI can be classified as a locator, a name, or both. A Uniform Resource Locator (URL) refers to the subset of URIs that, in addition to identifying a resource, provide a means of locating the resource by describing its primary access mechanism (e.g., its network location, type of transport and port to be used when contacting the resource). In the context of SIP there are three types of SIP URIs [10], [24], [47]:

- Address of record (AOR) that identifies a user
  - An example of a SIP URI AOR: `sip:goctala@lowpowersip.org` (It will need DNS NAPTR and SRV lookups for locating the SIP servers for `lowpowersip.org` domain as explained later in section 2.7.2)
- Fully Qualified Domain Name (FQDN) that can identify a specific device
  - An example of a SIP URI FQDN: `sip:goctala@77.28.100.1` or `sip:goctala@goce.lowpowersip.org` (for which an A record exists)
- Globally Routable UA URIs (GRUU) that identifies an instance of a user at a given UA, for the duration of the registration of the UA to which it is bound

There are two URI schemes for SIP that are similar to the "`mailto:`" URL (which allows specification of the SIP request-header field and the SIP message-body), these are "`sip:`" and "`sips:`". The later scheme is a secure SIP URI that requires TLS over TCP for security. With the "`sip:`" or "`sips:`" URI scheme we can define the subject, media type, or urgency of sessions that are initiated by using a URI on a web page or in an email message. The general form syntax in the case of a SIP URI is:

```
sip:user:password@host:port;uri-parameters?headers
```

An example of a SIP URI is: `sip:goctala@lowpowersip.org`. An example of a secure SIP URI is: `sips:goctala@lowpowersip.org`. The SIP URI: `sip:goctala@lowpowersip.org` is associated with the person "`goctala`". Note the SIP URI does not specify any specific computer. SIP URIs are very similar to e-mail addresses within SIP messages (e.g., in an INVITE message) to indicate the originator (From), current destination (Request-URI), intended recipient (To), and the redirection address (Contact).

2.7.2 SIP using DNS ENUM, NAPTR, and SRV records

In this section we will explain through examples the usage of DNS and the ENUM protocol (e.g., when a user dials a E.164 phone number) and how SIP utilizes DNS NAPTR and SRV records lookups. We will also explain through an

example the importance of DNS NAPTR and SRV during lookups for locating SIP servers.

The Domain Name System (DNS) is a scalable namespace system used in the Internet for translating names of network nodes into addresses. It also refers to resources on the Internet and in private networks. DNS names avoid specifics such as IP addresses and port numbers. DNS scales very well due to its distributed implementation and caching characteristics. The DNS services provided by a SIP based infrastructure should provide naming-authority pointer record (NAPTR) records [48] in conjunction with server (SRV) records [49] based on RFC 3263 [50] guidelines. This ensures high scalability, availability, security, and interoperability for the services that are to be deployed. A SIP UA or server should be able to discover what types of service are available for a name (such as SIP, email, or web service), what name to use for a SRV lookup, and (using the SRV record) what port and "A" records to use to find the IP address for the service. We will use an example in order to explain the mechanisms and services that are provided by SIP when using DNS lookups.

DNS and ENUM

Consider the SIP URI: `sip:+389-2-3114-835@lowpowersip.org;user=phone`. This URI signifies how to initiate a call from the Internet to the E.164 phone number `+389-2-555-1212` via a SIP proxy or directly to an IP telephony gateway at the domain `lowpowersip.org`. The `user=phone` tag is a hint to parsers that a telephone number is present in the username portion of the URI and is not just a numerical name. Note that the host portion of the URI (`lowpowersip.org`) is not necessarily a location for an IP telephony gateway or a SIP proxy since the creator of the URL may not know the location of the gateway or a SIP proxy. Therefore the UA/outbound SIP proxy at the creator of the SIP URI will see that this SIP URI includes a telephony URL (based on the format of the E.164 number written according to the ITU-T recommendation), and therefore it will use an E.164 number to URI Mapping (ENUM) protocol (defined in RFC6116[51] and RFC6117[52]) to locate the resource. ENUM bridges the gap between SIP and E.164 numbers by utilizing NAPTR and SERV lookups on a DNS ENUM server. The procedure of the ENUM protocol for locating a SIP resource for the `+389-2-555-1212` E.164 number can be seen in Figure 15 on page 31. Using ENUM, a translation occurs from the E.164 number to a SIP URI which in turns maps to other services. This provides portability. ENUM is supported by open-source SIP Proxies like SER, Kamailio, OpenSIPS, Asterisk, and some SIP phones (mainly those developed by SNOM).

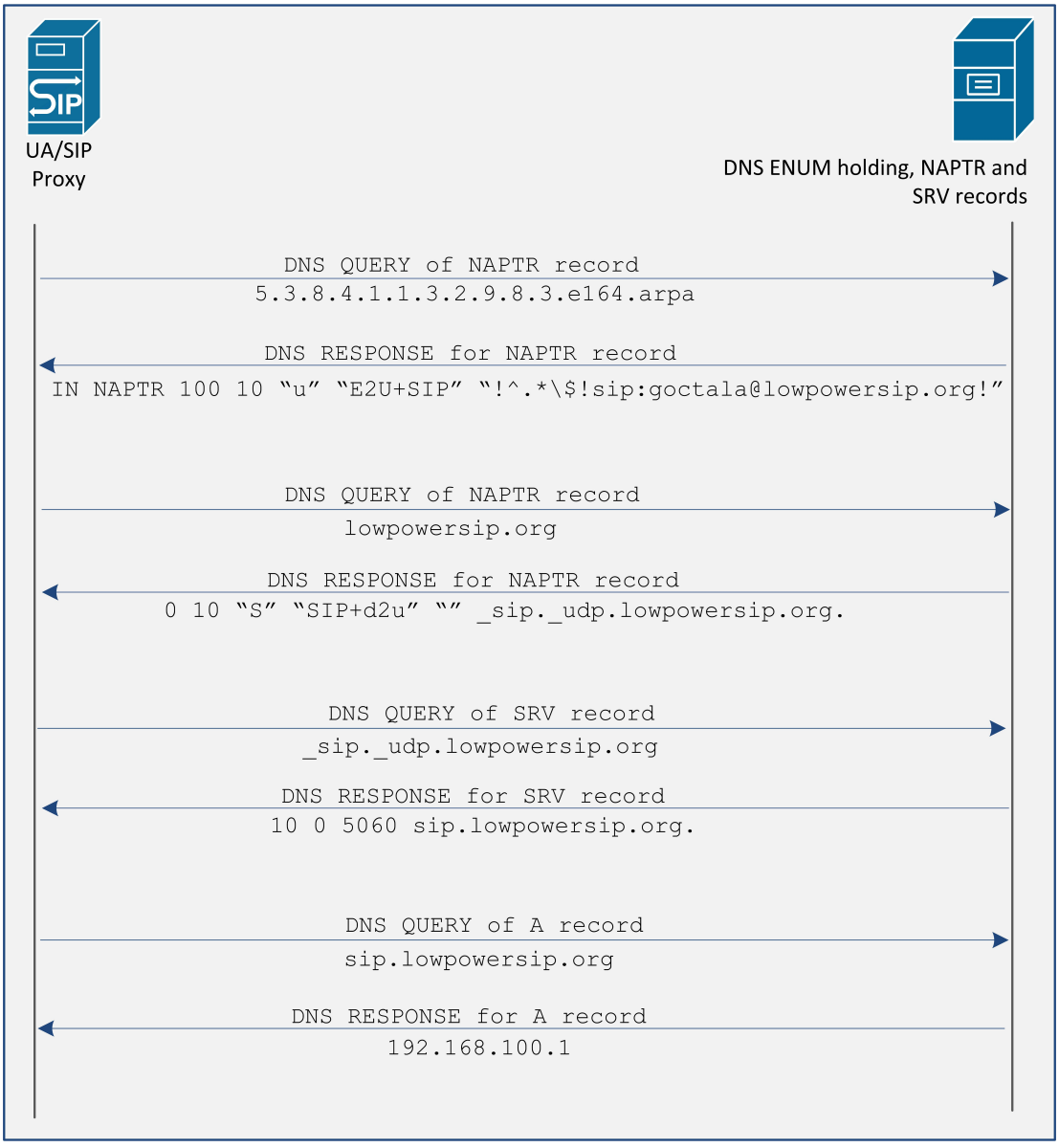


Figure 15: Example of DNS ENUM QUERY and RESPONSE message flows, when a SIP UA/Proxy server is contacting a DNS ENUM server (for NAPTR and SRV lookups) trying to resolve a new SIP URI for +389-2-3114-835 an E.164 phone number.

Locating SIP servers using DNS NAPTR and SRV lookups

Assume that a SIP UA/proxy server wants to find what SIP proxy server to use for sending an INVITE to `sip:goctala@lowpowersip.org`. Given this SIP URI the UA/proxy server does not know the IP address, transport protocol or port to send this INVITE (call) to. Since no transport protocol, or port is specified, and the target is not a numeric IP address, the client performs a NAPTR query for the domain in the URI (See Figure 16 on page 33) to its authoritative DNS. The DNS response in this case contains three ways (records) to contact lowpowersip.org, in the syntax of:

$$< order, preference, flags, service, regexp, replacement >$$

All of the answer records say that this type of record is an NAPTR for lowpowersip.org. The 10, 20, and 30 in the records correspond to the "order" (lower is preferred) to be used for this records. The "S" flag denotes that there is a SRV record lookup to be performed as a result of this record. There are four flags (together with the "S" flag) that can be used "A" (for "A", and "AAAA" record lookup), "U" (means that the NAPTR result is an absolute URI that the application should process), and "P" (implies a "non-terminal" rule where additional NAPTR lookups are necessary). The "service" denotes a transport service in the form of "SIP+D2X" where "X" can be either "U" (for UDP) or "T" (for TCP). In the answer provided in the first record uses UDP and the other two records use TCP. There is no "regular expression" specified for these NAPTR records which is usually related to ENUM (see Figure 15 for an example) for production of a substituted URI to the original request. The "replacement" and "regular expression" field are mutually exclusive. If one field exists, then the other one should not be used. The replacement is used as a result of the NAPTR lookup. There will be no substitution of the request since there is no "regular expression". Therefore because the lowest order is preferred (when the preference is the same for all records) the first to be contacted will make a SRV lookup for "`_sip._udp.lowpowersip.org`".

As a result of the SRV lookup we got two answers. The syntax of a SRV record is:

$$< Priority, Weight, Port, Target >$$

The first SRV record from the response (in Figure 16) has lower "Priority" which is preferred when the "Weights" are the same (as in this case). The "Port" number is 5060 for both records. Because the first SRV record has a higher preference (but equal weight) the "Target" of that record will be used as a result, and a DNS "A" record lookup will be performed for "sip.lowpowersip.org" which results in one or more IP addresses of the SIP server to be used to send the SIP INVITE.

In the example from Figure 16 the highest preference for the SIP UA/proxy server would be to send a SIP INVITE (for goctala@lowpowersip.org) via UDP to port 5060 on 192.168.100.1. If this attempt fails (time outs) the SIP UA/proxy server would perform an "A" lookup for the second SRV record from the response (sipcloud.lowpowersip.org). If that attempt fails as well, then the UA/proxy server will go back to the second NAPTR response and will perform

a SRV lookup on `_sip._tcp.lowpowersip.org` which will possibly result in a TCP connection to some other server and port combination.

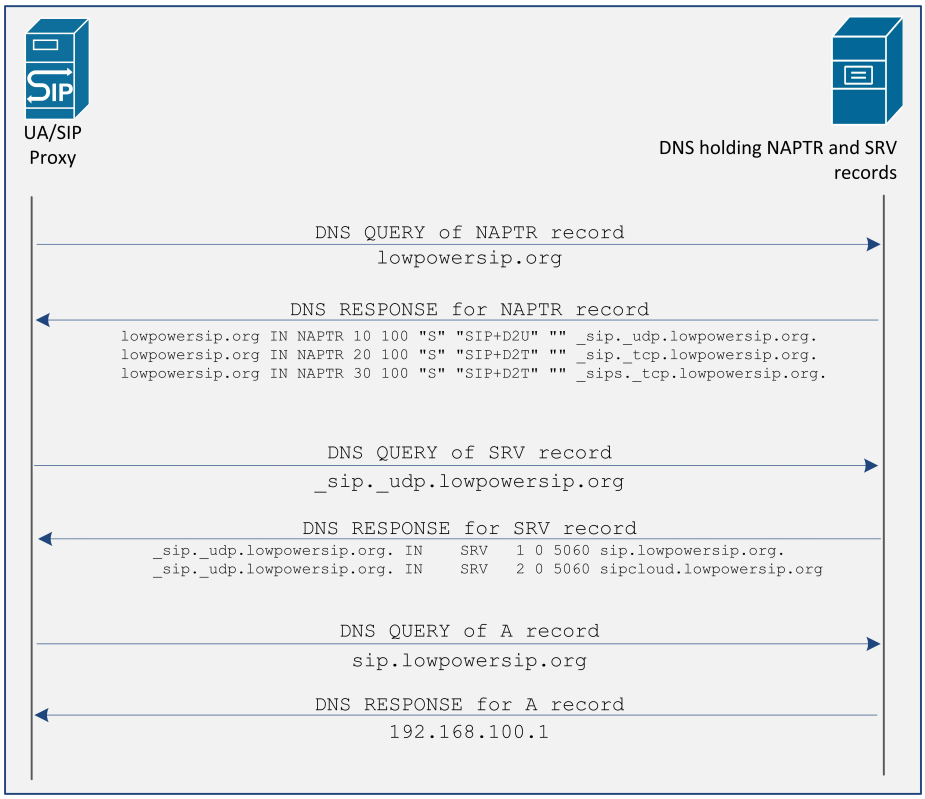


Figure 16: Example of DNS QUERY and RESPONSE message flows, in the case when a SIP UA/Proxy server contacts a DNS server trying to locate a SIP server to send a SIP INVITE to goctala@lowpowersip.org

2.8 Which cloud services for homes and SMEs?

In this section the basics of cloud computing will be described. In addition the types of services that are offered as industry standards by cloud service providers will be described. The local site (a home or SME) can make use of services offered by a cloud service provider to both provide greater scalability of services and to avoid the need to deploy these services locally. The idea is that the local site and the services provided in the cloud will form a VoIP enabled network that **collectively** offers the services that the user wishes. Because many of the services can be provisioned in the cloud, the local site's power consumption potentially will decrease. In order to understand what services we need to have provided by cloud service provider we will first describe the technology and different types of services that are offered by cloud providers in order to identify the best combination of local versus remote deployment.

2.8.1 What is cloud computing?

As described in Victor Delgado's Master's thesis [53] there is still no standardized definition of what cloud computing is, while experts and cloud providers each have their own interpretation and definitions of cloud computing. The U.S. National Institute of Standards and Technology (NIST), defines cloud computing as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction[54].

2.8.2 Cloud technologies

Today, cloud computing is possible on a large scale because of two technologies: virtualization and grid computing. Grid computing is a mechanism that utilizes the resources of many computers in a network in order to solve a single problem at the same time. In order to do this a software is needed that can distribute a program over several thousand computers. Grid computing can be thought of as a distributed large scale cluster computer. The difference between grid and cloud computing is that in a grid system the whole resource pool is allocated to a single user, while in cloud computing a user utilizes only a fraction of the resource pool thus enabling large numbers of users [53].

Virtualization allows multiple operating systems to be executed simultaneously on the same physical machine. Because of the abstraction of computer resources the physical resources can be shared by several virtual machines (VMs). However, to the user each virtual machine appears to be a separate physical machine, although they are actually sharing the resources of a single physical machine. Virtualization inserts an additional layer between the OS and the hardware. There are two flavours of virtualization: (A) hypervisor, an OS installed directly on the system, that directly monitors and manages the hardware resources (see Figure 17 (A)) or (B) a hosted architecture, where the virtualization layer is placed on top of a host operating system. Of these two flavours, a hypervisor is considered to be the fastest, most scalable, and robust option. Hypervisor OSs for Linux include: Kernel-based Virtual Machine (KVM) and Xen. KVM is an open source full virtualization solution for Linux. Using KVM, one can create and run multiple virtual machines that each appear as normal Linux processes and are integrated with the rest of the system. KVM runs on the x86 architecture and supports hardware virtualization technologies such as Intel VT-x and AMD-D. Each virtual machine has private virtualized hardware: a network interface, disk, graphics adapter, etc. [55][53]. KVM was first introduced in Linux kernel version 2.6.20 (released February 2007)[55].

XEN (first released in 2003) is a mature technology that is supported by many vendors and service providers. The Xen hypervisor is a powerful open source industry standard for virtualization. Xen offers a powerful, efficient, and secure feature set for virtualization of x86, x86-64, IA64, ARM, and other CPU architectures. It supports a wide range of guest operating systems including Windows, Linux, Solaris, and various versions of the BSD operating systems. Xen powers most public cloud services and many hosting services,

such as Amazon Web Services, Rackspace Hosting, and Linode. Commercial virtualization products such as Oracle's VM and XenServer are built on top of Xen, as well as desktop virtualization solutions such as Qubes OS and XenClient [56].

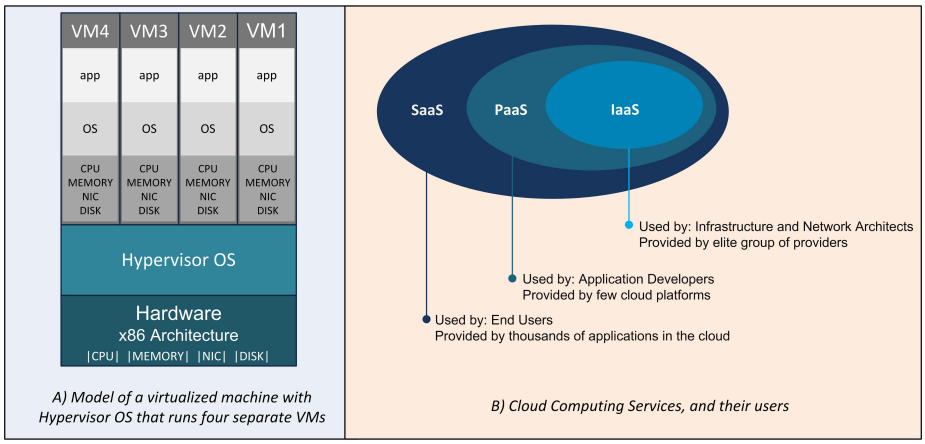


Figure 17: (A) A model of a virtualized machine using a hypervisor and (B) represents the service layers of cloud computing and their uses.

2.8.3 Deployment models

There are four different types of cloud infrastructure deployments: public, private, community, and hybrid [54]. Each of these is further described below.

Public clouds when the physical infrastructure and service/business models are owned by a cloud service provider. In this type of cloud, different customers run different applications while sharing the infrastructure and pay according to their resource utilization.

Private clouds are typically built and owned by a large enteprise and used only by this enterprise, here the customers can be branch offices or acquired businesses of the enterprise. However, this model can also be leased from a cloud service provider. The cloud provider custom builds, installs, and manages the cloud for its enterprise customer.

Community clouds If a group of customers share similar requirements, they can agree to share a common infrastructure. This potentially enables them to share the configuration and management of their cloud. This can be done either as a collaboration between the customers or leased from a third party or cloud service provider.

Hybrid clouds Any interaction between public and private cloud that form an infrastructure together can be seen as hybrid cloud. An enterprise can have a private cloud in their premises, but lease storage (for backup purposes only) as a service from a public cloud.

2.8.4 Cloud computing service models

The architecture of a cloud service provider is categorized upon their choice of cloud technologies into a stack of service types. This labelling was inspired by the everything as a service (XaaS) taxonomy. Today there are three main architectural categories (see Figure 17 (B)) that are offered as a service from a cloud service [54],[57],[53]. These categories can be described as:

Infrastructure as a Service (IaaS) The cloud service provider allows the customer to create a virtual machine (VM), e.g. a bootable file that can be executed by a virtual machine hypervisor emulating a physical computer, upload this file to the provider, and run the VM at the provider's data centre [57]. The customer is unaware of the underlying hardware and pays based upon the resources that they use. The resources that can be consumed can include any kind of resource, e.g. storage, runtime of a VM, or network bandwidth usage. The runtime costs often depend on the number of virtual CPUs presented to the VM by the hypervisor, the speed of the CPUs, and the amount of random access memory (RAM) available to the VM [58]. IaaS is aimed at network architects (see Figure 17 (B) and Figure 18 for examples). Some IaaS providers are: Amazon Web Services (AWS and EC2), RackSpace, GoGrid, and CloudSigma.

Platform as a Service (PaaS) The cloud service provider provides the customer with hardware and a toolkit environment according to the customer's needs. This toolkit environment include an OS, supported languages, data bases, etc. This toolkit enables the customer to build higher layer application services. PaaS is usually aimed at software developers, that want to host their applications in the cloud (see Figure 17 (B) and Figure 18 for examples). Some PaaS providers are Microsoft Azure, Bungee Connect, WorkXpress, Force.com for Google App Engine, and OrangeScape, among may others.

Software as a Service (SaaS) The end user of this service can only modify certain parameters of an application or software that is provided by the cloud service provider. Usually the cloud service provider provides a GUI for management and configuration of the SaaS by the end user via a web browser. Some of the SaaS providers and vendors are: SalesForce, Service-Now, Microsoft Office 365, Bime, Google Apps, Keynote, and Gomez.

2.8.5 Summary

There is no PaaS or SaaS solution on the market that addresses the problem of this thesis project, therefore we conclude that in order to enable all the services in the cloud as shown in Figure 1 requires an IaaS model from a public cloud, e.g. EC2 from Amazon, along with a service level agreement (SLA) that will guarantee the security and availability of the service.

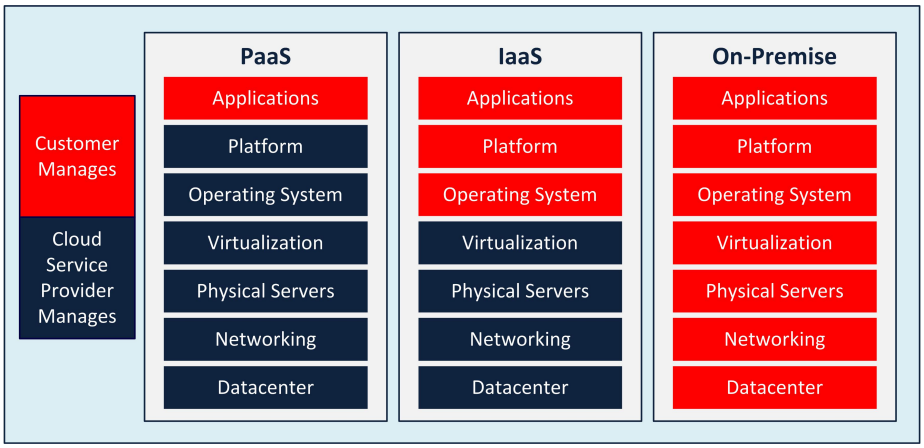


Figure 18: Management responsibilities for different types of cloud services.

2.9 Network Infrastructure Services for homes and SMEs

The network infrastructure for the home and SME environment (as ilustrated in Figure 1 on page 3) is an essential element for enabling the VoIP infrastructure for such an environment. The network infrastructure in the LAN must provide basic services in order to ensure connectivity, IP routing, DNS, DHCP, (optionally) NAT, and (optionally) WiFi connectivity. For the rest of this thesis we will assume that the home LAN consists of an Ethernet, unless otherwise explicitly mentioned. In order to connect the local site to a cloud service and to minimize the computational load of a VoIP system at the locate we will assume that there is always a broadband connection between the LAN and the internet. Additionally, some services, will need to be installed at the local site, while others will be supported by a cloud service. In this section we will review all the technologies that are involved in order to ensure a robust and reliable low power VoIP system for home and SME use, given the assistance of elements that are deployed in a cloud service.

2.9.1 DNS, DHCP, NAT, and TFTP

DNS is a globally distributed service which we rely on to resolve domain names into IPv4 addresses. So when alice@domain1.com calls bob@domain2.com, the first thing that the SIP proxy server does is resolve the SIP domain2.com into an IPv4 address by querying a known DNS server. After learning the corresponding IP address the SIP proxy can construct and send a SIP INVITE message to Bob's incoming proxy. It is important to note here that the domain name is resolved using a query for a SIP server for this domain name. This will cause a DNS query for a SIP service record associated with this domain name. The details of such service records are described in section 2.7.2 on page 29.

DHCP is a bootstrap protocol that is used to automatically assign an IPv4 address to a device, inform it of the IPv4 address of the gateway to the Internet, and inform it of the IPv4 address of a DNS server that is to be used. DHCP can also provide the address of a TFTP server from which a SIP hardphone

can download the software that is to execute along with its configuration files. These configuration files provide the addresses of the inbound and outbound proxies, dial plans, directories, etc. which provides the SIP hardphone with the information that it needs for initial registering and call establishment in a SIP based network. Details of the configuration of a DHCP server are discussed in section 6.8.

2.9.2 IP routing

Static or dynamic routing protocols can be used in order to determine where to route packets. Static routing in the home can be used in order to connect to the cloud service via a VPN tunnel, this also could be suitable for a SME; however, depending of the business needs, dynamic protocols might be used as well. While an SME might use a statically configured VPN, an SME could use dynamic routing protocols to dynamically connect to the cloud service via the best path, where this path can change as a function of the time of day, current loads, path failures, etc. Ideally a SME would have multiple access networks that they could use to connected to the Internet, thus providing greater reliability, while minimizing expenses.

2.9.3 Virtual LAN (VLAN)

A Virtual LAN (VLAN) enables a cluster of hosts to act as if they were connected by a LAN even though they are physically not connected to a common LAN. Using VLANs also enables the logical separation of networks independent of the physical topology of the network. In the home and SME context creating a separate VLAN for the VoIP traffic may be necessary in order to separate real-time media traffic from data traffic. A VLAN can be implemented by QoS and marking priorities on layer 2. One method of creating VLANs is to use a switch that implements the IEEE 802.1Q layer two protocols. However, it may be desirable (especially in the SME environment) to enable interVLAN routing when necessary. In interVLAN routing all the traffic between the VLANs will have to pass through the router and be routed based upon the router's routing information base (RIB). In the RIB, access list policies can limit or allow certain types of traffic according to the source and destination IP addresses. The router polices interVLAN communication on the IP level. This gives the administrator an inexpensive solution for controlling traffic using a single router.

2.9.4 Virtual Private Network (VPN)

The motivation for creating a VPN is primarily economic, but a secondary purpose may be to increase the network security of the traffic that is crossing an untrusted network (or networks). Companies and organizations frequently wish to connect their local offices or have personnel working from their home or while travelling. Providing this connectivity using provider leased lines is expensive. Today there are two main technologies for overlay VPNs: (A) site-to-site VPNs and (B) remote access VPNs. Site-to-site VPNs extend the enterprise's network resources. In the context of homes and SMEs connecting the LAN to the cloud service can be done via encrypted site-to-site VPN tunnels over a public and untrusted network such as the Internet, thus providing secure and robust

connectivity between the LAN and the cloud service network. The data can be encrypted on the transport layer by using the SSL protocol or on the IP layer by using IPsec. There are both advantages and disadvantages of both technologies. IPsec is more mature and robust. Additionally, IPsec can support any types of IP traffic. SSL based VPNs have also proved to be robust and reliable. When implementing SSL based VPNs an additional choice is whether to transport this traffic on top of UDP or TCP. The advantage of using UDP is that one avoids head of line blocking when a packet is lost (as would occur when using TCP as the transport protocol). The disadvantage of using UDP is that you have to open an explicit hole in the LAN's firewall and configure static forwarding of the port to the internal machine that is to receive the tunnelled traffic or place the machine that is to receive this traffic in the DMZ. However, in our case the tunnel is to and from the local site (WRT54GL), therefore we know the device's IP address and we can allocate the port, as well as open the hole in the firewall. Hence we can also avoid the head of line blocking that would degrade the RTP traffic's performance in terms of delay and jitter. Section 5.1 describes the measurements (methodology) for the performance difference between TCP and UDP.

2.9.5 Firewall

The basic idea behind a firewall is to protect the network from external threats. For example, a good default firewall policy is a Deny All rule, which simply prevents everything from crossing the firewall. In addition to this policy you can explicitly allow each service/protocol/port only for the services that you want (potentially constraining the hole through the firewall in terms of source and destination IP addresses, protocol, and port number(s)). In the home and SME context we can create a rule to allow all the VoIP traffic in/out to only come from the cloud network via the VPN network tunnel, while all other traffic might be dropped, except for HTTP port 80. It should be noted that it may be useful to allow SSH traffic to enter the LAN so that administrators can remotely connected to one machine inside the firewall, authentic themselves, and then carry out their tasks.

2.9.6 Wirless LAN (WLAN)

In order to enable local mobility in a home or SME network environment we could deploy a WLAN based on IEEE 802.11 a/b/g/n/... via one or more access points (APs). In order to increase the security (especially in the SME environment) we suggest employing WPA2 Enterprise mode. WPA2 is an enhanced version of WPA, based on the final, ratified version of IEEE 802.11i. IEEE 802.11i defines a mechanism for mutual authentication between an 802.11 client and an AP via an authentication server such as a RADIUS server. Remote Authentication Dial In User Service (RADIUS) is a standard protocol defined in RFC 2138 [59] and is a centralized protocol for Authentication, Authorization, and Accounting (AAA) of users in a network. A RADIUS server could be deployed in the cloud to provide WPA2 Enterprise mode for the SME environment, thus providing the enterprise with secure wireless local area network connectivity.

2.9.7 Quality of Service (QoS)

Quality of Service (QoS) in a VoIP enabled network is a complex issues and properly supporting QoS requires integration of various QoS mechanisms from multiple layers in the protocol stack. However, for a SME or home VoIP enabled network that is assisted by a cloud service, connected via a broadband connection with a data rate greater than 1.5Mbps we can employ VLAN type tagging. This means that the traffic from the VoIP VLAN will be tagged and prioritized over the connection to the cloud. This may require careful configuration and processing of the VPN traffic - for example, to make sure that the VPN's packets have the appropriate tags.

2.9.8 Backup

Having a reliable backup solution is an essential part of any network infrastructure. The primary purpose of these backups is to recover data after some type of local failure, be it by accidental deletion or corruption. The second main goal of creating backups is to enable roll-back of system configurations to previous working/trusted configurations. To backup configuration files from a running network device, we can use rsync[60]. Rsync is a tool (a software application and network protocol for Unix-like and Windows systems) for copying and backing up data from one location to another. It has an intelligent synchronization mechanism that only copies files which are different between the same directories at the source and destination. This mechanism makes the whole process run faster and consume less network bandwidth. Combining rsync together with the SSH protocol, the backup procedure can be done securely. In the context of homes and SMEs we can backup all of the local site's configuration files to a redundant cloud service backup solution. Similarly the configuration of the cloud service itself can be backed up locally to allow restoration of service, potentially even with another cloud service provider.

2.9.9 Network monitoring

A network monitoring system (NMS) can be deployed in a cloud service to monitor vital functions of the VoIP enabled network infrastructure in both the cloud and the local site. The NMS can provide a web based real-time representation of all the services. In order for this system to work we need SNMP management information bases on all the devices that are to be monitored. An example of an open source NMS is Nagios [61]. Nagios can be implemented in the cloud and can run custom made checks against services in the local site. This can provide the NMS with CPU and memory usage, the local SIP subscriber database, firewall status, VPN statistics, VoIP channels utilization and WAN connectivity statistics. Based upon the status of all (or some subset) of these device certain actions can be performed automatically, e.g. if there are no WLAN users after five o'clock in the SME's network then the network monitoring system can instruct the local site to turn off the WLAN access point and thus save power.

2.10 Low Power, Low Cost (embedded system) Architectures

The Linksys WRT54GL is a custom made embedded system that consists of an MIPS32 based processor. This system is designed to be controlled, by a custom made Linux based OS software (such as DD-WRT) or the vendor's own software. Details of the DD-WRT software are given in section 2.11. In order to understand the characteristics of this low power device, in this section we will explain the architecture and principles of these devices with a focus on WRT54GL.

2.10.1 Technology overview

In its simplest form an embedded hardware platform has an input interface where it receives information, this information is processed in order to provide output through an output interface [62][63]. An embedded system has the following generic characteristics: sophisticated functionality, real-time operation, low manufacturing cost, (optionally) uses an application specific processor(s), limited memory, operates for a long time without user intervention, and most important is to have low power consumption. There are various types of embedded systems according to their purpose: general computing (PDAs, set-top boxes, etc.), control systems (controlling a vehicle's engine), signal processing (RADAR, Sonar, DVD players), communication and networking (smart phones, internet appliances)[62][63]. The Linksys WRT54GL was designed to implement a low cost firewall, router, switch, and WLAN access point. When implementing an embedded system there are two main considerations for optimization: hardware and software. A typical embedded system consists of the following components [62][63]:

- **Hardware**
  - Processing element (microprocessor, micro-controllers)
  - Peripherals
  - Input/Output devices
  - Interfacing Sensors and Actuators
  - Interfacing Protocols
  - Memory
  - Bus
- **Software**
  - System Software (specialized operating system with a characteristic of dedication, real-time scheduling)
  - Application(s)

The processing element (the CPU) is typically a microprocessor or microcontroller. The main requirements for an embedded system typically are energy efficiency and high code density due to the limited amount of program and data memory. Professor Mark T. Smith characterizes the choice of microprocessors for such applications in terms of MIPS/Watt/$ - thus a processor that offers high performance at high power and high cost is less desirable than a processor that offer sufficient performance for low power and at low cost.

Microprocessors such as Intel's x86 Core2Duo use general purpose computer architectures. It is highly desirable that an embedded system is as highly integrated as possible, therefore the goal is to integrate onto a single chip the CPU core, the memory (both ROM and RAM), some I/O functions, network interfaces, etc. Additionally, there are other components such as, a timer module which allows the micro-controller to perform tasks for certain time periods, and serial I/O ports to allow data to flow between this chip and other chips.

2.10.2 Linksys WRT54GL

The Linksys WRT54GL is based on a Broadcom chipset to realize an embedded system. This device is a commodity hardware that can be bought on Amazon for US$50. It contains no fans or hard disks. The factory default, external power supply that comes with the WRT54GL, can provide a maximum output of 12V DC at 1A leading to a total system maximum power consumption of 12W. According to [8], we will need to limit this to values lower than 6W in on-state or full operation. This is a MIPS32 based architecture equipped with Broadcom's BCM5352 CPU or chipset [64]. In Figure 19 we can see the BCM5352 architecture.

The BCM5352 integrates a high-performance MIPS32 processor, IEEE 802.11 b/g MAC/PHY, SDRAM controller, and a configurable five-port Fast Ethernet switch. The BCM5352 provides WLAN connectivity supporting data rates of up to 125 Mbps and it is backward-compatible with IEEE 802.11 b/g. The BCM5352 supports a WAN connection via its configurable media interfaces. The per-port programmable four-level priority queues enable QoS (IEEE 802.1p) for guaranteed bandwidth applications, DiffServ/TOS, and L2/L3 IGMP snooping. The IEEE 802.1Q VLAN allows flexible implementation of VLAN grouping and WAN port segregation. The BCM5352 is designed to be used to implement low-cost, high performance systems for residential and SME markets [64].

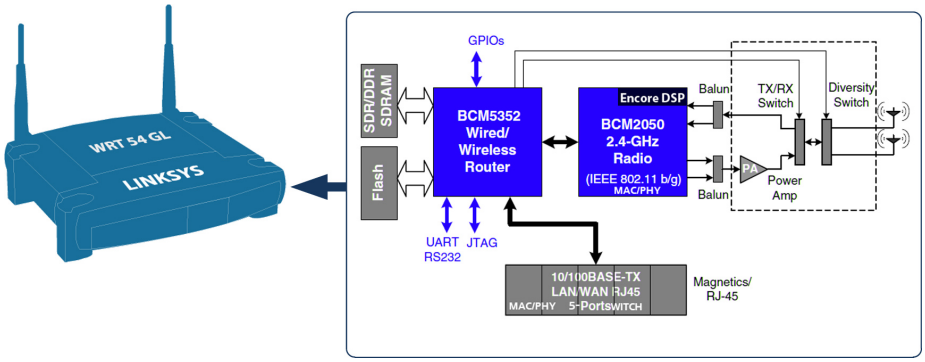


Figure 19: Linksys WRT54GL - SoC, hardware architecture overview of the BCM5352

The flash memory of the Linksys WRT54GL can contain an operating system (typically a Linux kernel), drivers, applications, and supporting tool chains. The size of the flash memory for WRT54GL v1.1 is 4MB and the RAM is 12MB. The

flash memory stores the firmware, in our case an OS image (a bootable file) with a flash memory file system containing applications and other files. WRT54GL is fully supported by the opensource firmware development communities such as Tomato, OpenWrt, and DD-WRT. This development software can be used to expand the functionalities of this device or even to implement a very different sort of device that uses these same interfaces.

2.10.3 DD-WRT

This thesis implements a customized version of the DD-WRT firmware on the WRT54GL and explores different software capabilities to enable this device to serve as an Internet gateway and the local part of an IP telephony system assisted by a cloud service (as shown in Figure 1 on page 3). The main reason for choosing DD-WRT is because it supports custom firmware modifications with the "Firmware Modification Kit" [65]. In addition, DD-WRT has a well structured and responsive forum community, bug trucker, and well written technical documentation. DD-WRT firmware is released under the terms of the GNU General Public License (GPL) for many IEEE 802.11a/b/g/h/n wireless routers based on a Broadcom or Atheros chip reference design. The firmware is maintained by BrainSlayer and is hosted at dd-wrt.com.

For the purpose of this thesis, the 4MB of flash memory, in WRT54GL v1.1, was a limiting factor for implementing of all the features that we need. This limited amount of memory meant that we could not install a generic firmware version build from DD-WRT that supports both VoIP and VPN features at the same time. To support this software we needed to install a flash memory of at least 8MB. In order to implement the necessary features that we initially need the following basic networking services (WLAN, NAT, DNS, DHCP, VLAN, SSH, telnet, etc.), SIP infrastructure (SIP router, registrar, and proxy), VPN client/server, QoS, SNMP, RADIUS authenticator, etc. Two additional actions are necessary, first to expand the flash memory by hardware modification of WRT54GL [66] and second to customize of a cross-compiled generic DD-WRT image using the "Firmware modification kit" [65]. Note that one of the goals is to eventually move many of these software functions to the cloud, thus reducing the software that needs to be executed and stored locally. One goal would be to reduce the amount of software to such a point that it would fit into 4MB, thus eliminating the need for any hardware modifications.

2.10.4 Raspberry Pi

An alternative hardware platform is the Raspberry Pi [67] a credit sized ARM based computer that costs US$35. The main market for the Raspberry Pi is to serve as an educational tool for learning programming. The Raspberry Pi can be compared with a 300MHz Pentium 2, but with HD capable graphics. The Raspberry Pi is equipped with a Broadcom BCM2835 chip. This chip contains an ARM1176JZFS processor, running at 700MHz, and a Videocore 4 GPU. The GPU is capable of BluRay quality playback, using H.264 at 40MBits/s. The GPU features a fast 3D core accessed using OpenGL ES2.0 and OpenVG libraries. Raspberry Pi, Model B, has 256MB RAM, 2 USB ports, a 10/100 Mbps Ethernet port for network connectivity, an HDMI port for video

output, audio output port, RCA composite video output, bootable SD card slot for storing OS and applications, and General Purpose Input Output ports (GPIO). Currently, Fedora, Debian, and ArchLinux are the supported ARM Linux distributions that can be installed on the SD card as the OS. Power is provided to the board via a 5V micro USB connector. The peak power consumption of the system is 3.5W. As of 9th of March, 2012 the Raspberry Pi was still not available for purchase through distribution channels, , hence we used only the Linksys WRT54GL runing the DD-WRT software as our hardware and OS platform.

2.11 DD-WRT Firmware/Software

This section explains the details and principles of the DD-WRT firmware/software when installed on a WRT54GL as third party firmware. This is important since one of the goals of this thesis project is to squeeze all the VoIP functionality of the local site (without hardware modification) into the 4MB of flash that is available on WRT54GL V1.1. In order to do this we will have to build a stripped custom version of the firmware that only incorporates the features and programs that are necessary for the local site operation when assisted by the cloud services (as explained in section 1.1 on page 2). We have to define which features or programs are not going to be used by the local site and remove them from the default firmware. In this way we leave more space (of the 4MB flash) for installation of additional packages or software (e.g. openvpn and OpenSER package) into the DD-WRT's firmware.

The DD-WRT firmware images for WRT54GL are a product of combining the SveaSoft's open source code, OpenWrt's kernel base, and the itsy package management system (ipkg) structure. The web interface of DD-WRT claims to be one of the most user friendly (and rich in features) of any WRT54GL firmware solution. This is the source of popularity today for DD-WRT and is where the developers focused most of their efforts. In essence the software functions are the same as in OpenWrt. The source code (and its versioning) for DD-WRT can be found at `http://svn.dd-wrt.com:8000/browser`, and most of it is under GPL license. The latest stable reported version (of the firmware image) for WRT54GL is 14929 and can be found at `ftp://dd-wrt.com/others/eko/BrainSlayer-V24-preSP2/2010/08-12-10-r14929/broadcom/`.

New firmware images can be installed on WRT54GL in three ways: via the Web interface (recommended for initial installation when flashing the factory default firmware from Linksys), via TFTP (recommended when DD-WRT firmware is already installed), and via the Joint Test Action Group (JTAG) ports (a last resort if we "brick" the router).

DD-WRT's feature availability depends on which firmware image is downloaded and installed on WRT54GL. Table 7 lists the available programs and associated functionality of different DD-WRT's firmware images for WRT54GL. The firmware version build names and size are: micro (1.7MB), mini (2.9MB), STD (3.6MB), VoIP (3.6MB), and VPN (3.6MB). Note that the flash memory of WRT54GL V1.1 is 4MB, which is partitioned in three main partitions: the first is for the common firmware environment (CFE) or the bootloader with

size of a 64KB, the second is the non-volatile RAM (NVRAM) partition which holds the values for the NVRAM variables with a size of 256KB, and the third partition is the firmware one (where DD-WRT resides) with a maximum size of 3776KB (approximately 3.6MB). In order to its use of flash memory, the DD-WRT firmware is loaded with sets of precompiled binaries (executables). The set of features that one firmware image offers corresponds to the binaries that are pre-compiled and loaded to that image.

Table 7: Features/programmes availability matrix for different types of DD-WRT's firmware builds for WRT54GL V1.1. Note that the Mega build is not supported on WRT54GL V1.1 because of the limited flash memory of 4MB, hence it is here just as a reference

Feature Description	**Type of DD-WRT's fimware builds for WRT54GL V1.1**					
	Micro (1.7MB)	**Mini (2.9MB)**	**STD (3.6MB)**	**VoIP (3.6MB)**	**VPN (3.6MB)**	**Mega (7.1MB)**
Access Restrictions	•	•	•	•	•	•
Anchor Free	•	•	•	•	•	•
Asterisk						•
Bandwidth Monitoring	•	•	•	•	•	•
Chillispot			•			•
Connection Warning Notifier		•	•	•	•	•
Dynamic DNS	•	•	•	•	•	•
EoIP	•	•	•	•	•	•
ext2 Support			•	•	•	•
ext3 Support						•
Hotspot System			•			•
HTTP Redirect	•	•	•	•	•	•
HTTPS Support for Web Mngt			•	•	•	•
IPv6				•		•
JFFS2		•		•		•
kaid			•			•
MMC/SD Support			•	•	•	•
NoCat			•		•	•
NTFS Support						
OpenVpn					•	•
Pound						•
PPTP Client/PPTP Server		•	•	•	•	•
ProFTPd						•
QoS	•	•	•	•	•	•
radvd			•	•		•
Repeater	•	•	•	•	•	•
RFlow			•	•	•	•
Samba/CIFS client			•	•		•
Security Log		•	•	•	•	•
SFTP						•
OpenSER				•		•
SMTP Redirect	•	•	•	•	•	•
SNMP			•	•	•	•
Iptables	•	•	•	•	•	•
Sputnik	•	•	•	•	•	•
SSH		•	•	•	•	•
Syslogd	•	•	•	•	•	•
tcpdump						•
Telnet	•	•	•	•	•	•
Tx power adjust	•	•	•	•	•	•
UPnP	•	•	•	•	•	•
USB						•
VPNC					•	•
Wake On LAN Server	•	•	•	•	•	•
WiFidog			•			•
WPA2 PSK/Ent	•	•	•	•	•	•
Wiviz		•	•	•	•	•

An example of a software executable specifically aimed for embedded Linux is BusyBox (which is incorporated into every DD-WRT firmware build). BusyBox is a multi-call binary that combines many common Unix utilities into a single executable. In DD-WRT most of the UNIX utilities (for use by the command line interface) are part of BusyBox and are symlinked to the BusyBox executable, for each function BusyBox acts like whatever it was invoked as. A list of compiled utilities for BusyBox version 1.10.3 on DD-WRT's standard (`dd-wrt.v24_std_generic.bin`) firmware version includes the commands:

```
arp, arping, ash, awk, basename, bunzip2, bzcat, bzip2,
cal, cat, chattr, chgrp, chmod, chown, chroot, chrt,
chvt, cksum, clear, cmp, comm, cp, cpio, cut, date, dd,
deallocvt, df, diff, dirname, dmesg, dos2unix, du, echo,
ed, egrep, eject, env, ether-wake, expr, false, fdisk,
fgrep, find, free, fsck, grep, gunzip, gzip, hdparm,
head, hexdump, hostname, httpd, hwclock, id, ifconfig,
insmod, install, ip, ipcrm, ipcs, kill, killall,
killall5, klogd, less, ln, logger, logname, losetup, ls,
lsattr, lsmod, md5sum, mkdir, mkfifo, mknod, mkswap,
modprobe, mount, mv, nc, netstat, nice, nmeter, nohup,
od, openvt, patch, pidof, ping, ping6, pivot_root,
printf, ps, pwd, rdate, readlink, realpath, renice,
reset, resize, rm, rmdir, rmmod, route, script, sed, seq,
sh, sha1sum, sleep, sort, start-stop-daemon, stat,
strings, stty, swapoff, swapon, sync, sysctl, syslogd,
tail, tar, tee, test, time, top, touch, tr, true, tty,
umount, uname, uncompress, uniq, unix2dos, unzip, uptime,
usleep, uudecode, uuencode, vi, watch, wc, which, who,
whoami, xargs, yes, zcat
```

2.11.1 The Boot Process

In order for the reader of this thesis project to understand the principles of operation of DD-WRT firmware, we will first explain (walk through) the booting process of a WRT54GL device up to the user-space, where programs are executed. We will than explain the default internal network configuration and setup of WRT54GL when DD-WRT is installed.

2.11.1.1 The Bootloader

WRT54GL (with DD-WRT firmware) boots, the common firmware environment (CFE) or bootloader takes control of the boot process. One of the first things that CFE does is check that the persistent non-volatile RAM (NVRAM) partition exists. If it does not exist, it is created using the NVRAM values stored in the CFE. Whether it is a new NVRAM partition or an existing one, the bootloader then reads the `boot_wait` parameter. If its value is set to on, it will

start a TFTP server[3] waiting for incoming connections for a specified period of time (approximately three seconds). Next, it will perform a cyclic redundancy check (CRC) on the firmware. If the firmware image is not corrupt, booting will proceed normally. If the firmware does not pass the CRC check (e.g., it is corrupt), the bootloader will go into the wait state until new firmware is received via TFTP. Refer to Figure 20 on page 48 for a graphical representation of the boot process.

It is important to note that the contents of the NVRAM partition are copied into the memory through the bootloader process. Later in user-space we can use the `nvram` command (available through the command line interface (CLI) in DD-WRT) to, manipulate these settings. However, these changes will exist in RAM only until they are committed to the NVRAM partition using the `nvram commit` command. A setting may not take effect until we execute scripts or programs that reread the NVRAM variables in the NVRAM partition. A final step in the bootloader process is the execution of the kernel image. The bootloader decompresses the kernel image from its known location in the flash memory into RAM and executes the kernel with the `init=/etc/preinit` option. An example of the parameters that are passed to the kernel (version 2.4.37 compiled for DD-WRT) at load time is:

```
root=/dev/mtdblock2 rootfstype=squashfs,jffs2 init=/etc/preinit
noinitrd console=ttyS0,115200
```

This set of parameters can be viewed via the DD-WRT's CLI (as in any other Linux based OS) by issuing the command `cat /proc/cmdline`, where `root=` is the partition where the rest of the DD-WRT file system is located, `rootfstype` is the format of the root partition (in this case it is squashfs which is a compressed read-only filesystem for Linux that corresponds to the physical 4MB of flash memory, and Journalling Flash File System version 2 (jffs2) for the mounted flash memory or SD card), `noinitrd` means that all drivers are built into the kernel (no need to load an initial RAM disk with extra drivers), `console` defines the device from which the kernel can accept login commands (in this example it is a serial port `ttyS0` with a speed of `115200` baud).

2.11.1.2 The Kernel

After the kernel bootstraps itself and issues the command `start_kernel`. The `/etc/preinit` script performs initial checks (read/write values from/into the NVRAM variables, creates directories, mounts a file system, etc.) for the pre-initialization process. An example of a preinit job is to make sure that the `ip_conntrack_max=` (which keeps track of the IPv4 connection buckets) variable has a value that corresponds to the RAM capacity of the device. For WRT54GL the maximum value is always set to 4096 connections to be tracked.

[3]Please note that WRT54GL V1.1 supports only TFTP server mode in the CFE bootloader process because of its vendor implementation. Other vendors might implement a TFTP client instead or both (and be able to choose between). When in the state of TFTP server/listener, the CFE bootloader (with its limited network connectivity support) answers ARP requests to 192.168.1.1, regardless of the IP address configured in the DD-WRT OS. Therefore in this stage a TFTP client on the network (with IP address other than 192.168.1.1 but within the /24 subnet), can `connect` to 192.168.1.1 and `put` a customized DD-WRT firmware image to the TFTP server.

As a primary job the `/etc/preinit` script mounts the `/proc` and `/sys` pseudo filesystems. It also prepares the `/dev` directory for access to console, tty, and media access devices. Right after this step the device drivers or kernel modules are loaded. The kernel mounts the rootfs (the root file system) and udev through the `/etc/preinit` script, a final step is to start the `init` daemon as a process by executing `/sbin/init`. This process becomes the first process ID (PID) and is responsible for starting all other processes.

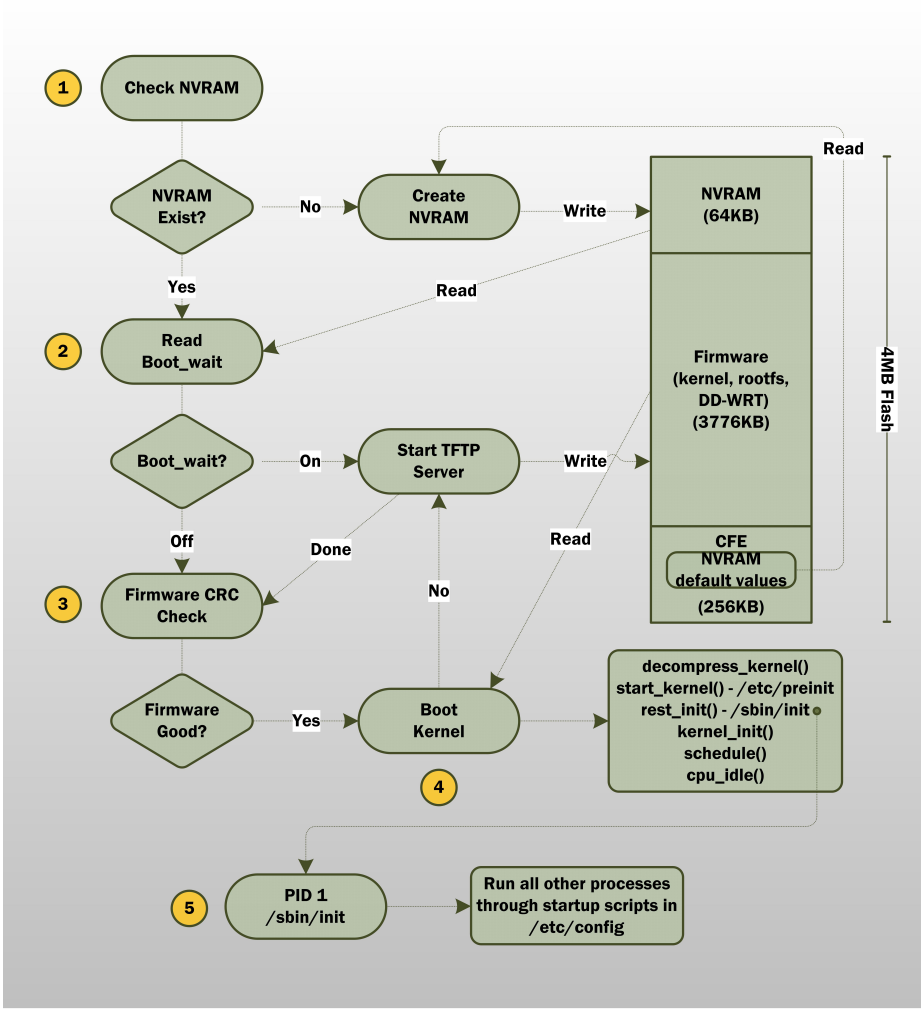


Figure 20: Flowchart of WRT54GL's (with DD-WRT firmware) boot process

2.11.1.3 The `init` daemon

User space execution starts after the kernel mounts the rootfs, right after that the first program run is `/sbin/init`. The `init` daemon calls `/sbin/rc S boot` (`S` for start and `boot` for booting, are parameters that are passed to the `/sbin/rc` program) that in turn calls the startup scripts that are located in:

```
/etc/config/
```

```
/opt/etc/config/
/jffs/etc/config/
/mmc/etc/config/
/tmp/etc/config/
```

The startup scripts are defined as `<program-name>.startup` files (Table 8 summarizes the types of script extensions that can be executed in DD-WRT and their meaning). When executed, these scripts, read the corresponding values (for a particular program) from the variables stored in the NVRAM in order to start the programs and pass the values (from the NVRAM) as parameters. Depending on the values of the variables (defined by `<program-name>.nvramconfig`) that are stored in the NVRAM a program can be started (or not).

Table 8: Script extensions types (for DD-WRT), that are present in /etc/config directory and are executed by /sbin/init and /sbin/rc daemons in order to start all the processes/programs and after that the uptime of a DD-WRT begins

Extensions in `/etc/config`	Description
`.startup`	will be executed on system startup, normally boot-time and before the firewall is configured. This script will also call `rc_startup=` NVRAM variable that stores custom startup scripts as value.
`.nvram.config`	these type of files define possible NVRAM variables for a particular program that are specific to the firmware version, all the possible variables for the NVRAM can be viewed with the `nvram show` command via the CLI.
`.webconfig`	this file is called when a change is made and confirmed via the Web UI in order to read or store a new value for a NVRAM variable and restart the program (via `<programname>.sh` script) that is related to the NVRAM variable.
`.firewall`	firewall configuration script that sets hooks to the netfilter hook API. This script also calls the `rc_firewall` NVRAM variable that stores custom firewall scripts as values. This script executes `iptables` rules for packet filtering (FORWARD, INPUT, and OUTPUT rules) and NAT (PREROUTING and POSTROUTING rules)
`.ip-down`	is run when a PPP connection has been shut down
`.ip-up`	is run when PPP connection is reestablished after a disconnect and after starting the firewall
`sesbutton`	is executed when the front button (also called Secure Easy Setup (SES)) of WRT54GL is pressed

The `init` daemon runs all the time, from boot until shutdown. On shutdown it will call `/sbin/rc K stop` in order to invoke all the shutdown scripts for proper termination of the programs. In a base or a standard image distribution of DD-WRT for WRT54GL (e.g., `dd-wrt.v24_std_generic.bin`) the following programs will run as a result of a successful boot process (or the `init` daemon):

init is the first process to run after the system boots, and in many ways it is the most important daemon. It always has a PID of 1 and is an ancestor of all user processes and all but a few system processes.

k∗ daemons that represent parts of the Linux Kernel that are managed as processes, they can be identified by their low PID. These

processes deal with various aspects of I/O, memory management, and synchronization of the disk cache.

cron is responsible for running commands at preset times. It accepts schedule files ("crontabs"). This process runs by default since processes such as "Watchdog Scheduler" depend on it.

dnsmasq DHCP and DNS server masquerading daemon

dropbear SSH remote login client/server used for management and configuration of DD-WRT.

telnetd remote login client/server used for management and configuration of DD-WRT.

httpd an HTTP server daemon used for management and configuration of DD-WRT.

watchdog connection watchdog is a mechanism that pings one (or more) other computers and if it can not reach them it will automatically reboot itself. This provides a crude way to correct situations where the router becomes wedged.

wland a user space daemon for access point and authentication servers. It implements IEEE 802.11 access point management, IEEE 802.1X/WPA/WPA2/EAP authenticators, and a RADIUS client.

2.11.2 Default Internal network

In order to configure the local site (WRT54GL) for all the services that it needs to support we have to understand the networking logic within the device when DD-WRT is installed on it. For example, if we want to configure dual WAN connections (1+1 redundancy) with a fail-over mechanism for the local site as form of a resiliency, we have to understand the internal design of the network logic in order to configure the device properly. Therefore in this section we will explain the default internal network configuration of WRT54GL with DD-WRT installed.

WRT54GL has built in router, a manageable switch, and a WLAN AP, which are interconnected. This can be seen in Figure 21. The built-in switch has six ports. Ports from zero to four are represented by physical RJ45 connectors, and these are the only visible ports on the WRT54GL. Port five is an internal port of the switch that interconnects with port eth0 from the router via an IEEE 802.1Q trunk. All the traffic via this port is tagged, according to its VLAN membership. There can be a maximum of five VLANs. By default two VLANs are configured on the switch (VLAN0 and VLAN1). The port-range 0-3 belongs to VLAN1 and represents part of the LAN. The LAN is formed by bridging the VLAN0 (ports 0-3) with eth1 (WLAN) interface. The router sees the LAN (VLAN0 + WLAN) as a br0 interface. The default VLAN is VLAN0, thus traffic that is not being tagged from the WLAN can reach VLAN0, since non assigned (untagged) traffic that comes on port 5 is bridged (not routed) to the default VLAN0. In this way reachability is achieved from WLAN to LAN and

vice versa via Layer 2. By default the LAN is configured as one network (e.g. 192.168.0.1/24).

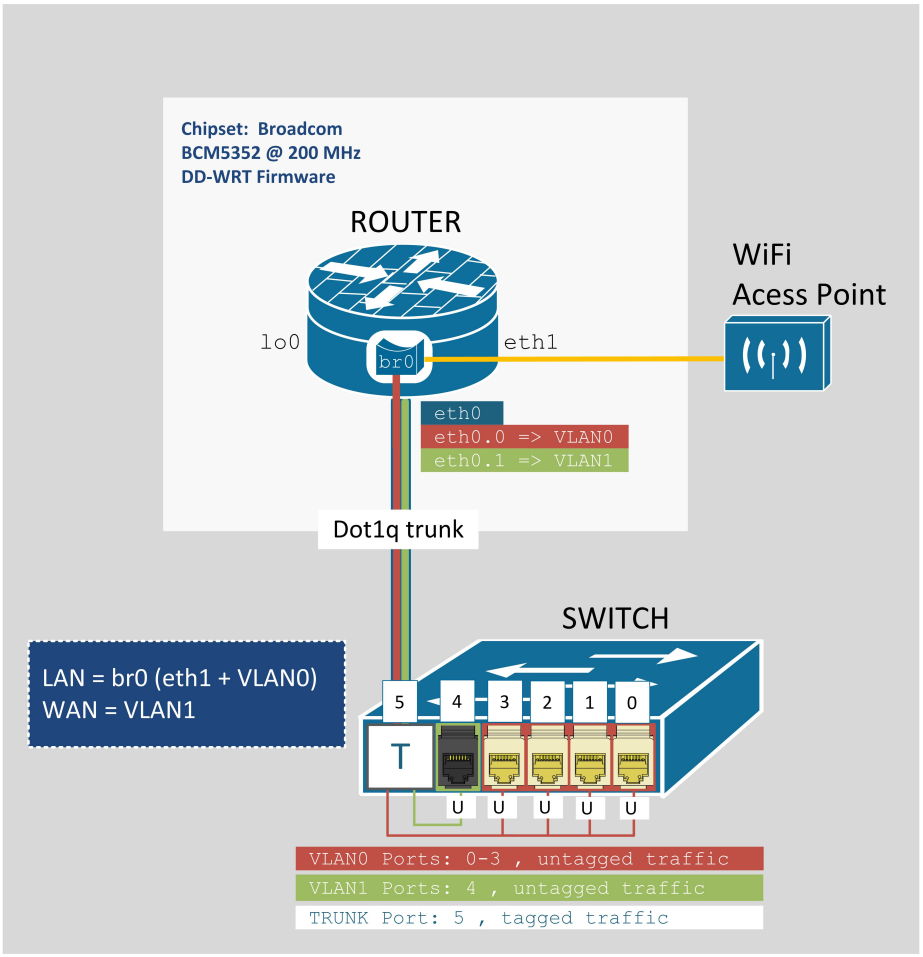


Figure 21: Default internal network operation logic and architecture diagram of WRT54GL (with DD-WRT firmware)

The WAN is associated with VLAN1 and has only port 4 assigned to it. The router routes traffic between WAN (VLAN1) and the LAN (br0). The firewall (iptables) rules are applied to traffic destined from VLAN1 to br0 and vice versa. QoS in DD-WRT can be configured via the intermediate queueing device (imq).

The default configuration of DD-WRT internal network can provide the following network services: Quality of Service (QoS), Network Address Translation (NAT), transparent web proxy, IP tunnelling, PPPoE, DHCP server/client, DNS server, telnet and SSH, WEB UI for configuration and management, bandwidth monitoring, etc.

DD-WRT's firmware provides command line utilities to manipulate all of the internal devices (of WRT54GL) and mechanisms. For example separating

the LAN from the WLAN with an independent DHCP server pool, assigning ports from VLAN0 to VLAN1, VLAN detached networks (LANs with different Internet Access), multiple SSIDs on the WLAN, etc.

3 Related Work

There are not many specific research projects or proposed solutions on the topic of how green a VoIP based on SIP technology can be or what can be done in order to minimize the energy consumption of such a solution. In this chapter we will present a sample of the work done by other researchers that is related to the problem area of this thesis project (as described in section 1.1 on page 2)

3.1 How green is IP-Telephony?

In [16], Baset, et al. examine the energy efficiency of existing VoIP systems. They divide and classify VoIP systems in to two categories: client-server (SIP UA and server infrastructure) and peer-to-peer (P2P) based (simillar to the approach used by Skype). They built energy models for these systems, and evaluated their power consumption and relative energy efficiency through analysis and a series of experiments. For the client-server model they built a SIP based VoIP infrastructure, with 2 server scenarios (SIP proxy and B2BUA) and use SIP hardphones as clients. For building and testing a P2P VoIP infrastructure they used Skype. Their findings show that even with efficient peers, a peer-to-peer architecture can be less energy efficient than a client-server architecture. Additionally, the presence of NATs in the network is a major obstacle to realizing energy efficient VoIP systems. Because most existing NAT devices maintain UDP bindings for only a short period of time, hardphones behind NATs need to periodically refresh the binding in order to reliably receive incoming calls. The hardphones achieve this by sending a SIP NOTIFY request every 15s to the SIP server, which replies with a 200 OK response. While wasteful, this method proved to be the only reliable way of maintaining NAT bindings. Because of this their analysis shows that in client-server VoIP systems with always-on hardphones, the total power consumed is dominated by the power consumption of the hardphones. They found that the SIP hardphones that they evaluated consumed between 3W to 6W. They also observed that the phone's power consumption does not change when placing a voice call. Moreover, because of these devices are always on in order to maintain reachability (using one of the NAT traversal techniques) they are the main component of the electrical power consumption of the entire VoIP system. In their experiments regarding a client-server infrastructure they found that a single SIP proxy server can handle approximately 500k subscribers, and consumes 210W under peak load. The RTP relay server or B2BUA server that they used consumed 240W and could relay 15k simultaneous calls, with each call having a bitrate of 64 kbit/s. Their study focused on IP telephony systems for a large number of customers, such as might be deployed for provider based solutions or large enterprises. In their study they also presented and suggested techniques for reducing energy waste in VoIP systems, specifically with regard to the client-server model, they suggested the following:

- Embed SIP user agents in the DSL/cable routers so that they can bypass the NAT inside the cable/DSL modem. The DSL/cable router usually has a public IP address. Such embedded UAs will likely require no media relays and do not need to send NAT keep-alive traffic.
- Use persistent TCP connections between the UAs and the SIP server.

Persistent TCP connections require significantly less keep-alive traffic for maintaining NAT bindings. However, such TCP connections will use up TCP control blocks in the operating system of the SIP server and if the SIP UAs are behind a common NAT there can be a problem with having sufficient numbers of ports (since all of the user's behind the NAT will appear to be coming from the same IP source address (or addresses).

- Use advanced NAT traversal techniques, such as ICE to allow user agents to detect network conditions and use RTP relays managed by the IP telephony service provider, only when absolutely required.

- Techniques such as Wake-on-LAN and Wireless Multimedia Extensions found in modern mobile computers may eventually lead to even more energy efficient always-on VoIP systems. Such devices could enter a power saving mode during periods of inactivity and be woken up remotely over the network upon arrival of an incoming call. Using power saving modes together with always-on VoIP is the focus of our ongoing work.

3.2 Data network equipment energy use and savings potential in buildings

In [14], Lanzisera, et al. present results of a study of network equipment energy use and include case studies of networks on a campus, a medium sized commercial building, and a typical home. To develop power use estimates, they measured the power consumption of network devices under varying conditions and combined this with actual power consumption values (rather than rated power) reported by manufacturers and third party test laboratories. They estimated that network equipment in the USA used 18 TWh, or about 1 percent of building electricity, in 2008 and that consumption was expected to grow at roughly 6 percent per year to 23 TWh in 2012, while the world usage in 2008 was 51 TWh. Their study shows that office building network switches and residential equipment are the two largest categories of energy use consuming 0.4% and 0.3% of the office or home's total electricity consumption. They estimate potential energy savings for different scenarios using forecasts of equipment stock and energy use, and savings estimates range from 20% to 50% of current usage based on full market penetration of efficient technologies. They observe that many residential and small business users receive equipment from their Internet service provider rather than via retail purchase. As a result the consumer has no choice or information about the energy use of this device, and the service provider has no economic incentive to provide efficient equipment since the customer is providing the electrical power for the customer premises equipment. Their study suggests that providers need to specify energy efficiency as a requirement in their specifications to manufacturers because the aggregate potential energy savings is very large. In their study three methods of saving energy are presented: Energy Efficient Ethernet (EEE)[68], improved power supply efficiency, and power consumption that scales with throughput.

- EEE [68] saves energy by putting the link to sleep when there is no data to communicate. Both ends of the link must support EEE to save energy, so broad market adoption should be a priority for policy efforts. EEE enables an Ethernet link that is idle to exit Active mode and enter a Low

Power Idle (LPI) mode. The power used in LPI mode is significantly less than in Active mode since some components of the physical layer (PHY) can be powered off. EEE brings the energy consumption of Ethernet links closer to their ideal consumption, which is directly proportional to the utilization of the link. Using EEE on all devices supporting gigabit Ethernet would result in a savings of 2.8 TWh in 2012 or 12% from 23 TWh (which is an estimate for the network equipment power consumption in the USA for 2012, as described in [14]). Energy will also be saved in the end devices connected to these products nearly doubling the overall savings. The savings assume all devices support EEE, because if either end of the link does not support EEE, no savings are achieved, so the full potential of EEE will take years to realize.

- Improved power supply efficiency is needed. Currently home and SME network equipment does not use an internal power supply, but rather these devices use an external power supply unit (EPSU). Because residential equipment typically uses an external power supply which is already regulated, significant savings are unlikely from power supply upgrades, as described in [14]. However, the fact is that most of the existing EPSUs are not very efficient - as compared to internal PC power supplies which are generally more than 80% efficient. To address this issue the EU parliament made a regulation (in directive 2005/32/EC of the European Parliament) on eco-design standards for external power supplies (to be sold or produced in the EU) that is aimed to contribute to the EU's target of reducing greenhouse gases by at least 20% by 2020 [69]. This regulation will lead to annual power savings (in the EU) of about 9 TWh by 2020 (for all devices that use EPSUs such as notebook computers, mobile phones, MP3 players, network equipment, etc.). As stated in the press release issued by the EU parliament regarding the regulation [69], the resulting savings of 9 TWh is enough to power Lithuania for a year and will reduce annual CO_2 emissions by more than three million tones. According to the regulation, as of 27th of April 2010, EPSUs that will be placed on the EU market must have no-load condition power consumption of no more than 0.50W. They must also have an average active efficiency of at least:
  - $0.500 \times P_0$ (nameplate output power of the EPSU): for EPSU with nameplate output power of less than 1.0W
  - $0.090 \times ln(P_0) + 0.500$: for EPSU with nameplate output power between 1.0W and 51.0W
  - 0.850: for EPSU with nameplate output power of more than 51.0W
- Dynamic power saving is possible by disabling unused ports in the switch fabric. The switch fabric (the hardware responsible for moving packets from port to port) is provisioned to move the maximum number of packets at all times. With 50% of the ports unused, this capacity can be reduced by 50%, while providing the same throughput per *active* port as the switch provides when all ports are connected and the fabric runs at full capacity. Disabling unused ports and scaling switch capacity to meet the needs of the utilized ports is the simplest and nearest term form of dynamic power savings. Within the network community, such dynamic energy saving approaches are seen as the best way to save energy in network equipment.

- Network equipment standby modes are commonly discussed as another energy savings opportunity a large fraction of the equipment experiences low utilization much of the time. People expect network connectivity to work on demand, rather than according to a schedule. Unfortunately, equipment in standby mode does not react appropriately to network traffic. Although it may be possible to have equipment wake-up very quickly (e.g., milliseconds) when traffic is presented, this really describes the dynamic power savings case. Based on the increasing demand for always-on networking in both residential and commercial buildings, network equipment standby modes are not seen as a viable, long-term savings strategy for the vast majority of networks.

3.3 Skilled in the Art of Being Idle: Reducing Energy Waste in Networked Systems

In [70], Nedevschi et al. focuses on reducing the 75% (of energy consumption for networked systems in the US) consumed in homes and enterprises. They state that this energy (112 TWh) is roughly equivalent to the yearly output of 6 nuclear plants. In order to reduce the energy consumption of the networked devices in home and enterprises they started by collecting data directly from 250 enterprise users on their end-host machines by capturing network traffic patterns and user presence indicators. With this data they seek to answer several questions: What is the potential value of proxying or using magic packets (the ones used in wake-on-LAN)? Which protocols and applications require proxying? How comprehensive does proxying need to be for energy benefits to be compelling? They found, that even though there is a great potential for energy saving, trivial approaches are not effective. They also found that achieving substantial savings requires a careful consideration of the trade-offs between the proxy's complexity and the idle-time functionality available to users, and that these trade-offs vary with the user's environment. Based on their findings, they proposed and evaluate a proxy architecture that exposes a minimal set of APIs to support different forms of idle-time behaviour.

3.4 Networked Power management for Home Multimedia

In [71], Virolainen and Saaranen, present a potential solution to enable more aggressive power management in networked home devices. This solution is based on the Universal Plug and Play (UPnP) protocol with Low Power (LP) specifications, which allows devices in power save modes to be discovered and awakened. It is also shown in a prototype that an efficient power management system can be built on top of UPnP LP, particularly if home router manufacturers support the UPnP LP proxy service in their devices. They conclude that, according to their study, UPnP LP has clearly proved its potential for improved home network power management as devices can use more aggressive power management methods and still be accessible when needed. However, UPnP LP is not a complete solution, and it requires efficient support from both hardware and software. As future work they plan to investigate wake up methods that are specifically needed for wireless mobile devices which cannot exploit Wake-on-LAN type methods.

3.5 Saving Energy in LAN Switches: New Methods of Packet Coalescing for Energy Efficient Ethernet

In [72], Mostowfi and Christensen, focused their research towards lowering the power consumption of small or home office (SOHO) Ethernet LAN switches (these LAN switches alone consume 8TWh per year in US alone), by proposing and evaluating, a new EEE policy (also proposed in [14]) of synchronous coalescing of packets in network hosts and edge routers. Their policy provides extended idle periods for all ports of a LAN switch, thus enabling greater energy savings than in the Ethernet PHY only. They evaluated their method using an ns-2 simulation model of a LAN switch. The results show that their method, can reduce the overall energy use of a LAN switch by about 40%, while introducing limited and controlled negative effects on typical Internet traffic and TCP. They estimate the potential energy savings that can be obtained by deploying adaptive coalescing on all future SOHO switches to be approximately 3.5 TWh/year in the U.S. alone.

3.6 Code of Conduct on Energy Consumption of Broadband Equipment for the European Union (EU)

The EU wrote a code of conduct document for the energy consumption of broadband equipment in the EU [8]. In this document the EU states that depending on the penetration level, the specifications of the equipment and the requirements of the service provider, a total European consumption of up to 50 TWh per year can be estimated for the year 2015. With the general principles and actions resulting from the implementation of the code of conduct in [8], the (maximum) electricity consumption could be limited to 25 TWh per year, this is equivalent to 5.5 Millions tons of oil equivalent (TOE) and to total saving of about €7.5 Billions per year. The Code of Conduct, sets out the basic principles to be followed by all parties involved in broadband equipment, operating within the European Community, with respect to the energy efficy of this equipment. The document defines values for both power consumption and response times (time to switch out of low power mode) for CPE, depending on their operating states and type of broadband access device. There are three operating states that are defined for the CPE (i.e., the home gateway): off-state, lower-power-state, and on-state. Each of these states is further described below.

Off-state In the off-state the device does not provide any functionality. This state is entered when the CPE is switched off or when it is disconnected from the mains. The only possible power consumption remaining is due to the power supply which should comply with the Code of Conduct for External Power Supplies [73]. The equipment can only leave this state by being switched on manually.

Low-power-state In the low-power-state the device is idle, with all the components (not processing user traffic on the WAN, WLAN, and LAN interfaces; with one phone connected, on hook, and off hook detection active) being in their

individual low-power states. In this state the device is not processing or transmitting a significant amount of traffic, but is ready to detect activity. The transition time from low-power-state to on-state for a component (such as AP or a switch port) should be very fast (much less than 1 second) in order to avoid adversely impacting the customer's experience. The establishment of an Ethernet link (connectivity) between two components may take more than 1 second, but must stay below 3 seconds. As stated in the [8] this longer transition time is tolerable since it requires user interaction to bring up the link (e.g. connect a network device to a switch port or to boot a PC).

On-state

The on-state of a home gateway is defined as all the components being in their on-state (all of the WAN, LAN, and WLAN interfaces are active and processing user traffic; with one active VoIP call). For the interfaces carrying user traffic a throughput of 25% of the available bandwidth in both directions (e.g. 25% Tx and 25% Rx) is to be considered.

In the code of conduct the home gateway power consumption targets are computed from the components according to the configuration (profile) of the home gateway. Table 9 shows an example profile of a fast Ethernet router with 1 WAN and 4 LAN Ethernet ports:

- in low-power-state: all LAN Ethernet ports disconnected
- in on-state: all LAN Ethernet ports active

Table 9: Power values for Ethernet home gateway and its components plus additional VoIP devices, recreated from [8]

Function	**low-power-state (W)**	**on-state (W)**
Central functions + Fast Ethernet WAN interface	2.5	3.3
4 port Fast Ethernet switch 10/100mbps	0.6	1.8
Wi-Fi interface single IEEE 802.11b/g	0.7	2.0
Total for Ethernet home gateway	3.8	7.1
Other Home Network Devices:		
ATA/VoIP gateway	1.5	2.2
VoIP telephone	3.0	3.7

3.7 A First Step Toward Green Wireline Broadband

In Eric Svensson's Master's thesis [74], the goal was to lower the power consumption of broadband networks by developing software algorithms that continuously and automatically configure the equipment to achieve this goal. An example of such an algorithm would be the Self Optimizing Network (SON) concept from wireless networks. The primary reasons for implementing such an algorithm are to decrease both the environmental impact of the networks and the operators' expenditures. The thesis focuses on lowering the power of vendor specific equipment from Ericsson, such as the IP DSLAM provider equipment (PE), connecting to customer premisses equipment (CPE), together forming a network, based on the VDSL2 standard. Even though the proposed algorithm, could not be applied to lower the power consumption of the Ericsson IP DSLAMs due to the inability to find suitable parameters to tune, this thesis gives a good example of a software based approach to lower the power consumption in a network and to lower the power consumption of network devices.

4 Low power IPT System design

In this chapter we will define the IP telephony (IPT) system that is to be used for SMEs and homes through examples of the main call scenarios and a low power logic scenario. From there we will derive the features and services of both the local and cloud site. This will serve as guidance for designing and configuration of the various parts of the test-bed. We assume that the reader is already familiar with the technology context of this thesis (described in Chapter 2, the problem area (described in section 1.1) and the related work (described in Chapter 3).

4.1 IPT system main call scenarios

In this section we will examine the main call scenarios for the IPT system (shown in Figure 1), through examples in order to illustrate (and enumerate) the patterns of operation in an SME (we consider the home scenario to be a subset of this behaviour). These scenarios will serve as implementation and design guides for a SIP based IP telephony system in accordance with Figure 1. We will first start by explaining the actors in the scenarios and their environment. The examples relate to Figures 22, 23, 24, 25, 26 starting on page 63.

Extension 100, belongs to **Alice** (email:alice@sme.com) who works in the SME's office. Her SIP extension/account is configured (username: 100, password:1234) in both the local site (in the WRT54GL's local SIP server) and the cloud site (in this scenario we consider this to be an Asterisk B2BUA server). Both configurations were made either by the company's IT personnel or by an authorized administrator via a web based GUI. When Alice's, SIP hardphone boots for the first time, it learns its relevant network settings and its TFTP configuration server through a DHCP response. Given this information the phone downloads and applies the corresponding configuration files for her profile from the TFTP server that is located in the WRT54GL[4]. Now Alice's phone has the WRT54GL's IP address configured as its outbound proxy and the Asterisk's server IP address as its inbound SIP proxy. Alice also registered at both the WRT54GL and Asterisk SIP registrars. Note that this dual registration is mainly necessary both to reduce the time required for searching for location information for a SIP peer and for redundancy[5]. For example, when calls are local (extension to extension within the local LAN) the SIP INVITEs need only be processed by the WRT54GL's outbound SIP proxy that can contact the local SIP registrar for location information, this we believe is a better design than if the SIP registrar in the cloud was to be contacted for local calls. Another advantage is that in this way if the cloud site is not available for some reason,

[4]Hence that the TFTP server can be easily deployed in the cloud site, however, if the cloud site is not reachable for some reason(e.g. connection to the cloud service is temporarily unavailable or there is an Internet access problem), this design ensures that the SIP hardphone would still be configured and local internal communication can occur. Note that the WRT54GL has to be powered on for there to be external connectivity, so the only additional power consumption of serving a file from this device is the actual energy required to serve the file out of memory. This is expected to be much less than the power required for the whole network path and the server to remotely provide the file.

[5]However, if Alice is within the local LAN and the WRT54GL fails, then she has no service and the data stored at the cloud registrar is of no use. Therefore the cloud registrar also has Alice's cellular phone registered.

then local calls can still be processed and established. Additionally, local clients can be created and exist only in the context of a local site. The SIP registrar at the Asterisk server is needed to process the incoming[6] SIP INVITEs faster, rather than contacting the registrar at the WR54GL. Note that if the local site is not available on the public Internet for some reason, incoming call requests could be processed and routed to the UMS in the cloud, thus still maintaining some availability of the IPT system and enabling messages to be taken for the local users or perhaps they can be notified via their cellular phones.

Extension 101 belongs to **Bob** (email:bob@sme.com) who also works in the same SME's office as Alice. His configuration procedure is the same as Alice's, meaning he is also registered with both the WRT54GL and Asterisk registrars, although he has a different password:4321 and username: 101 for authentication as well as different IP address for his SIP hardphone.

Paul is an external customer to Alice's and Bob's company. Paul's hardphone is connected at his company's LAN, and registered at his company's SIP registrar. Paul's company uses a commercial VoIP provider.

Emily owns a small shop, in another country from the country where Bob's and Alice's company resides. Emily places weekly orders with Bob's and Alice's company, either by e-mail or directly contacting them via the embedded Google talk application within her gmail account. Emily appears in Alice's and Bob's buddy list that is managed via the Asterisk server.

Scenario 1 (relates to Figure 23 and 22 on page 63)
Alice calls Bob. To do so Alice dials 101 on her hardphone which results in an INVITE message being sent to the WRT54GL outbound proxy. The WRT54GL checks if 101 is registered in the local database. Since it is, the INVITE is routed locally to Bob's phone, which starts to ring. After he answers the call the actual RTP/RTCP media stream flow is established directly between Alice's and Bob's phones.

Scenario 2 (relates to Figure 24 on page 64)
Bob calls Paul. In this scenario, Bob, first searches for Paul's contact information in the company's contact directory via the XML browser capability of his phone. This browser connects to a XML directory server running as a service in the cloud. Bob finds Paul's contact information and initiates an INVITE to paul@lowpower.com via the WRT54GL which is Bob's outbound SIP proxy. Paul's account is not known to the local outbound proxy and therefore the INVITE is forwarded via the VPN tunnel to the SIP proxy server that is located in the cloud. The Asterisk B2BUA Server also does not know Paul's location, therefore it routes Bob's INVITE through a pre-configured SIP trunk via Alice's and Bob's VoIP provider1. VoIP provider1 contacts VoIP provider2 which is Paul's VoIP provider and forwards the INVITE from Bob. Paul's phone now rings, after Paul answers the call, the actual media path will depend on both Paul's VoIP provider2's configuration and Bob's VoIP provider1's configuration.

[6]Here an *incoming* means a SIP INVITE message that comes from an external factor (e.g. a VoIP operator) on the SIP trunk at the cloud site, not within the IPT system (e.g. from the local site).

Whether the media paths are direct connections or whether the media path must pass through a path between the IP PBXs will be a function of their configurations.

Scenario3 (relates to Figure 26 on page 65 and Figure 14 on page 27)
Emily calls Alice, and later conferences with Bob and others. In this scenario Emily sees that Alice is online in her buddy list on her google talk account (an embedded application within her gmail account) and knowing that Alice is available Emily initiates a voice call by pressing the call button in her browser on Alice's contact information. This calling request is processed by the Google Talk VoIP network and forwarded to Alice's company's IP telephony system. This request will be passed to the company's Astersik B2BUA Server in the cloud, which receives the calling request on its configured Jabber/gtalk channel. Next the Astersik Server checks the dialplan logic for this request and sees that the request is destined to Alice. After checking the registrar the incoming proxy sees that Alice is registered with her SIP phone, at her office at the local site (via the WRT54GL), the Asterisk Server routes a SIP INVITE message to WRT54GL through the persistent VPN tunnel. The WRT54GL finds Alice's location in the registration database, and forwards the INVITE from emilly@gmail.com. This contact information is displayed (on the display of Alice's SIP hardphone) when Alice's phone starts to ring. The actual media and signaling path is encrypted from Emily's PC all the way to the Asterisk Server (this is achieved because the Asterisk server and Emily's browser established a TLS session via Google's gtalk Server), and then from the Asterisk Server to the WRT54GL the traffic travels inside the VPN tunnel. After Alice has answered Emilly's call, Alice realizes that they will need to talk to Bob and other two employees from their company in order to help Emily with her ordering. Therefore Alice transfers this call to a conference room by dialing 900 (by sending a `REFER` method to the Asterisk B2BUA), and then INVITEs Bob and the other two employees to connect to a specific conference room that is protected by a pre-shared password. The actual media path is destined to, and from, the centralized conference bridge that is located in the cloud, where it is mixed (as necessary) and distributed to all the parties.

Scenario 4 (relates to the scenario illustrated in Figure 12 on page 26)
Paul calls Alice but she is not there. When Paul's calling request is initially processed by the Asterisk B2BUA in the cloud, the Asterisk server sees that Alice is registered with the WRT54GL (with both her SIP hardphone and mobile smartphone, but she wants to be contacted only via her office phone), thus the Asterisk proxy routes the INVITE to the WRT54GL, which proxies the INVITE to Alice's phone. The phone rings but since Alice is not in the office, this request times out and results in a "408 Request Timeout" being sent to the Asterisk Server in the cloud via the WRT54GL. The Asterisk Server then forwards Paul's call to the UMS. This results in a message being played that says "Alice is not in her office at the moment, please leave a message". Paul leaves a message in Alice's voicemail box. This message is immediately sent as an attachment to an email that is sent to alice@sme.com via a SMTP server. A NOTIFY message is send to her office SIP hardphone, so that it can indicate that there is a new voicemail message waiting for her via illuminating an LED indicator on her hardphone. Alice receives a new email on her mobile phone,

which is a notification email that carries the voicemail message (as an audio file) as an attachment. Alice downloads the message and listens to it on her mobile smartphone that is connected to the Internet via a 3G/4G network. Now Alice can either choose to call Paul from her corporate mobile smartphone that has a SIP softphone connected to the corporate IP telephony system running in the cloud or it she can wait until she returns to the office and call Paul from there.

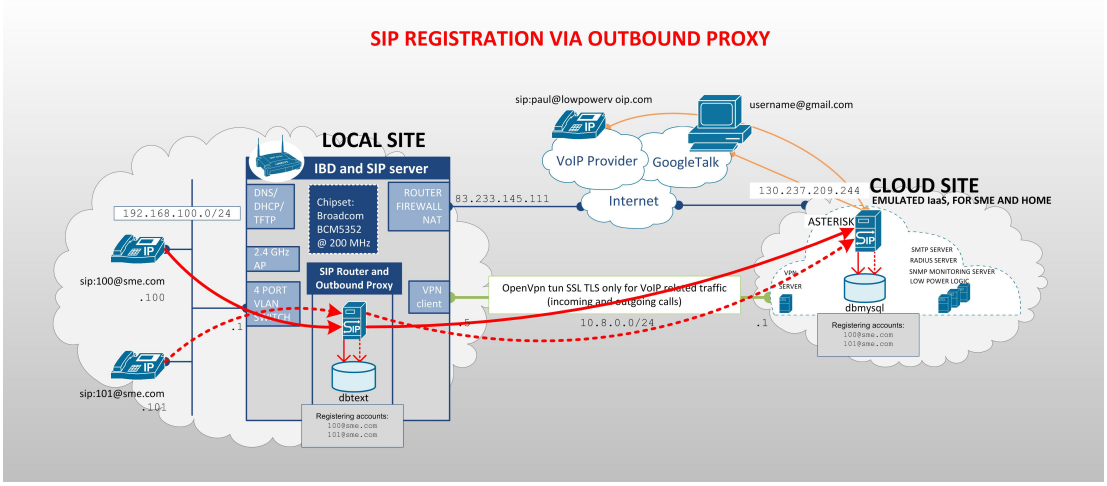


Figure 22: Example of SIP REGISTER message flows in the IPT system

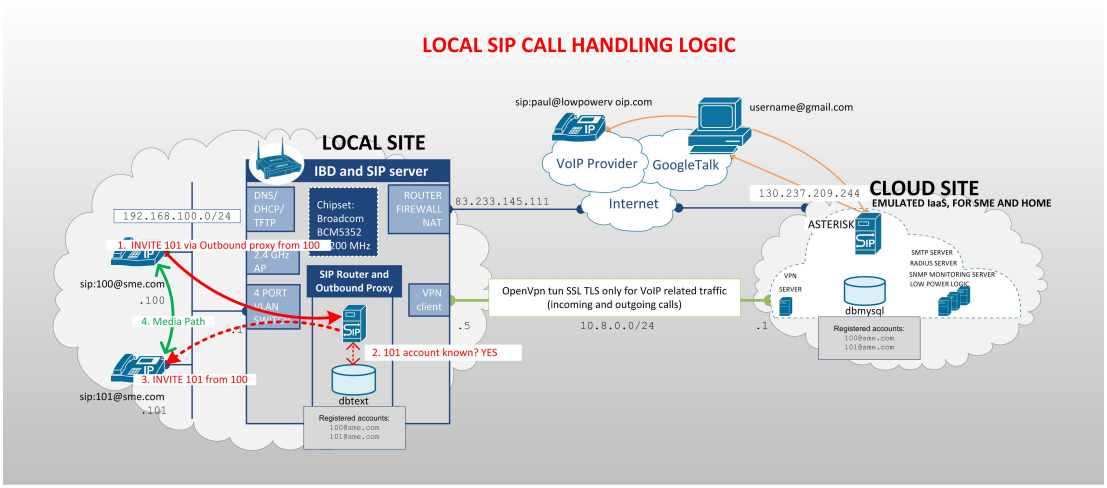


Figure 23: Example of how local call requests, stay within the local site, only the ones that are unknown to the local site SIP Server will be routed to the cloud SIP Server via the VPN tunnel

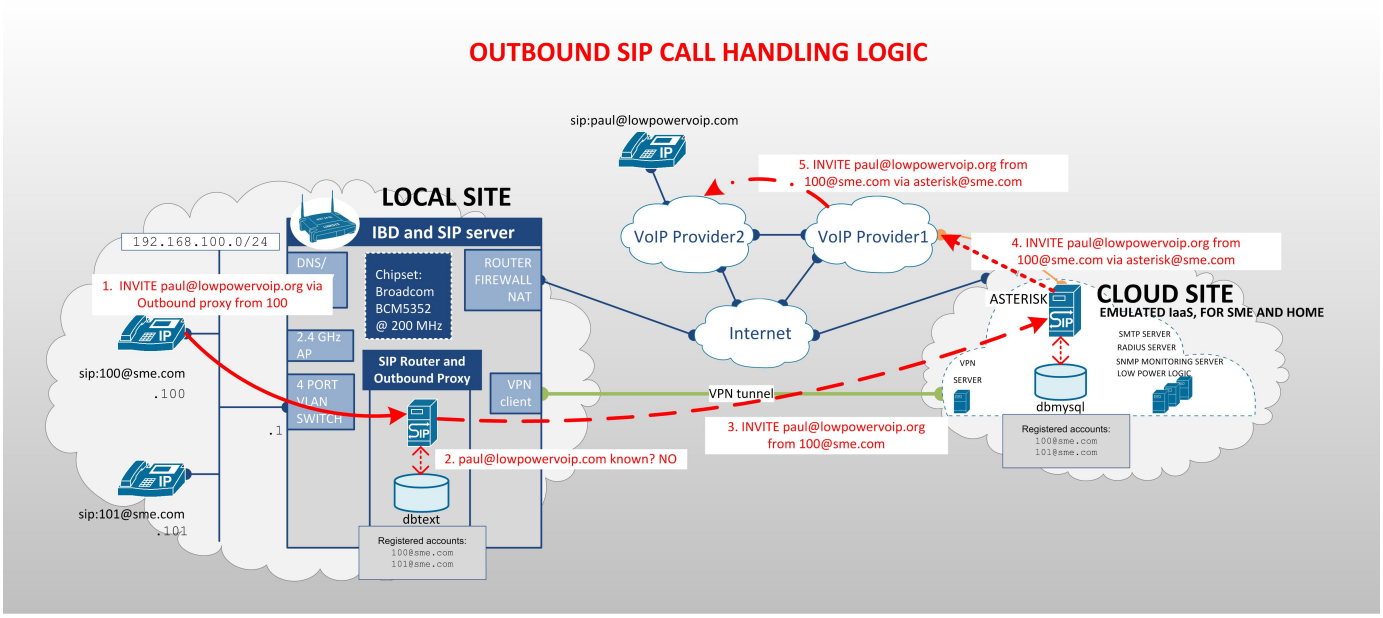


Figure 24: Example of how outgoing calls are processed by the IPT system.

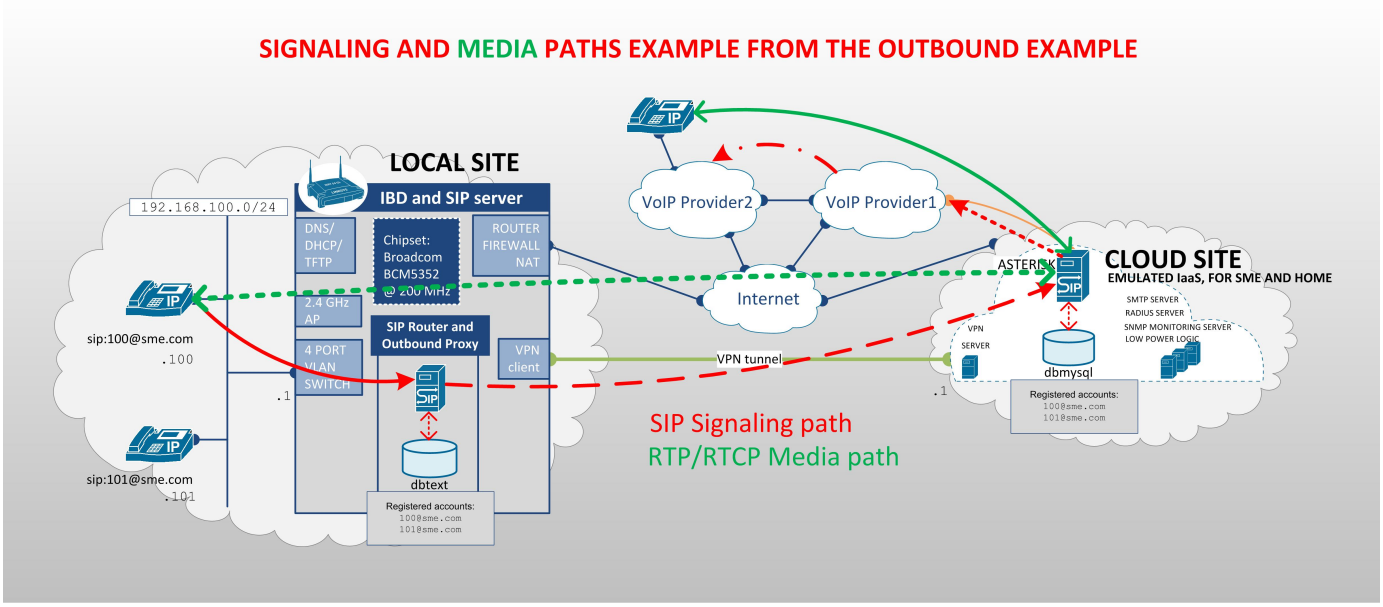


Figure 25: Example of the Media and Signaling flows for the scenario in Figure 24

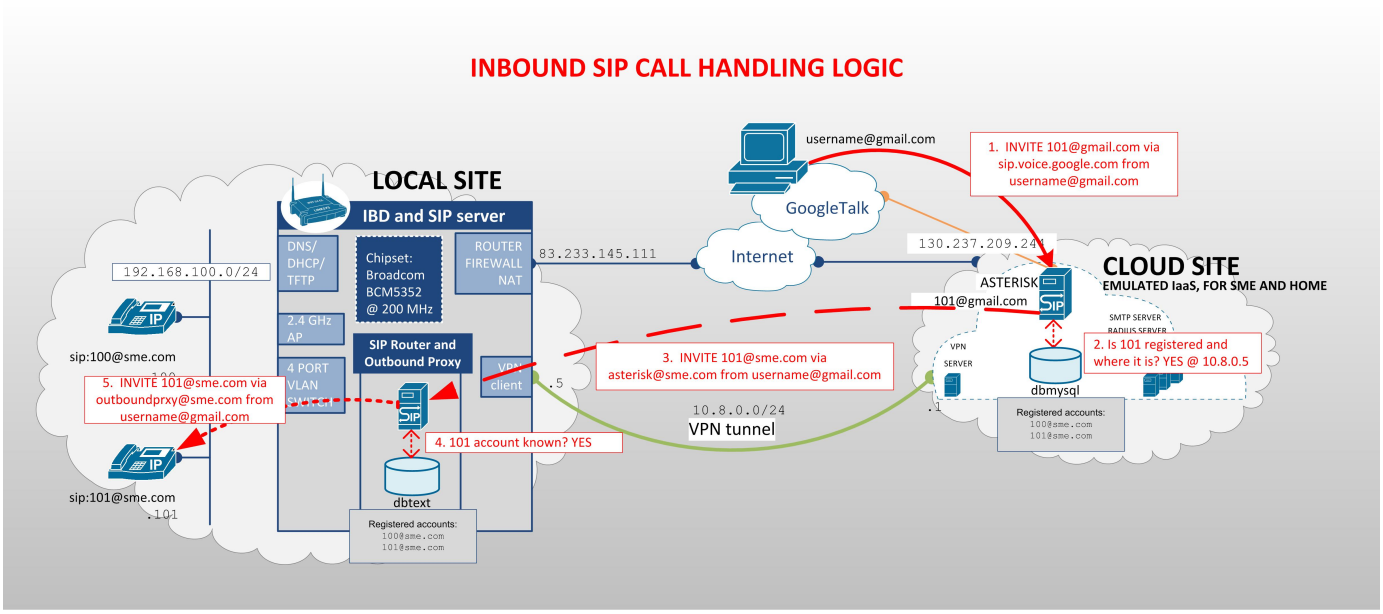


Figure 26: Example of Incoming call handling by the IPT system

4.2 Main Low-Power-Mechanism scenario

In an SME all IPT systems usually have day/night operating modes. These modes change the logic that processes incoming and outgoing calls. This logic could be automatically configured based upon the normal office hours, e.g. if the office opens at 8:00 o'clock in the morning then the day mode will be activated, and if the office closes at 17:00 then the night mode will be activated. Otherwise the mode might be configured manually by entering a sequence of numbers and codes on a SIP hardphone by a certain employee. When this employee enters the correct sequence it triggers the execution of the Low Power Logic scripts for the local site - thus reconfiguring the communication between the local site and the cloud.

In this scenario the IP telephony system is configured for an automatic day/night mode (via a timer module in the cloud) within the office hours 8:00-17:00. This means that every working day (Monday-Friday) the Asterisk Server automatically executes different policies for the processing of the incoming and outgoing calls (resulting in different Asterisk dial-plans to be used), depending on the hour and which day it is, according to predefined schedule logic in the cloud server. Thus at 8:00 o'clock the day mode operation is executed resulting in normal operation and processing of the incoming and outgoing calls, thus everybody can be reached (based upon where they are registered) and everyone can dial a number or a SIP URI, and this request will be processed according to the daytime-dialplan. At 17:00 o'clock, the IP telephony system enters in night mode, meaning all the inbound calls are automatically directed to the IVR system from where the calling party can leave a voicemail or the caller can learn the office hours (by listening to the playback of an audio message),

all outgoing calls are forbidden except for calls to emergency services. This timer logic of day/night operation is also used to trigger the Low Power logic mechanism/scripts in the cloud service. The cloud instructs the WRT54GL to enter Low Power mode by shutting down unnecessary functions. Since the WRT54GL assumes that no one is present at the office, the cloud Low Power script instructs the WRT54GL (at 17:00 o'clock) to power-off its 2.4GHz radio interface for the WLAN (which consumes approximated 1.3W of a total of 4W in normal operation). Additionally the script decreases the CPU clock frequency[7]. Another feature in this scenario is the flexibility to enter the day/night mode manually, e.g. Alice comes to the office on Saturday (the IPT system was automatically placed in night mode from Friday 17:00 to Monday 8:00 o'clock according to the predefined schedule at the cloud). In order to make some phone calls and read email, Alice enables the day mode by pressing a sequence of numbers on her phone $*75*password$, this enables the day service operation of the IPT system and executes the relevant Low Power scripts in the cloud to turn on the WLAN, LAN, and increase the CPU clock frequency, thus restoring normal network operations. To go back to night service mode, on exiting the premises she enters $*76*password$ on her hardphone before she leaves.

4.3 Design for the Local and Cloud site

In order to test the design of our low power IP telephony system we need to define the features and services of both the local and cloud site. This will enable us to implement and test a low power IP telephony system for SMEs and homes. This section presents the features and services of both the local and cloud site, and serves as a blue print for implementing a test-bed.

4.3.1 The Cloud site

This thesis project will implement and test a SIP based VoIP architecture as described in Figure 1. The testing will take place for all the main call scenarios and the main low power scenarios, as presented in sections 4.1 and 4.2.

The **cloud** service will be emulated as IaaS, using a Dell Optiplex 755 DT with one Intel®Core2™ DUO CPU (with both cores enabled) clocked at 2.33GHz, 2GB RAM, one 160GB SATA@7,200rpm hard disk, one power supply, and a single 10/100/1000baseT Network interface. The server is publicly available on the Internet via a provider aggregatable IP address (advertised from AS6821 as route 77.28.0.0/16). The physical location of the cloud site is in Skopje, Macedonia, connected to the Internet via a DSL subscription with a 10Mbps download and 768Kbps upload speed. The machine is connected to an uninterruptible power supply (UPS) system. Ubuntu Server 11.10 (GNU/Linux 3.0.0-15-server x86_64) is installed as the main OS, with the following configured roles and features:

[7]Considering the internal network architecture of WRT54GL with DD-WRT, as in Figure: 21, it is not possible to administratively and selectively disable (power-off) a physical LAN interface. We would have, additionally lowered the power consumption in night-mode by turning off LAN ports that are not related to IP telephony e.g. LAN printers or PCs. One active LAN port (with 100BASE-T Ethernet established link through an FTP CAT5e cable of 10 meters) consumes 0.25W

Asterisk Server Version 1.8.4.4 configured as a SIP B2BUA, Registrar and Location server; voicemail with e-mail execution via an SMTP server as UMS, meetme centralized conferencing, and support for the call services described in section 2.6.

XML Directory Server Version 1.2 of Open 79XX XML Directory Server [75] is an application designed to utilize the display features on Cisco 79XX IP Phones (since we are testing using a Cisco 7940G SIP hardphone). The Directory Server is configured to produce on-screen phone directories, search capabilities, text memos, and interactive display of user status. This server is based on PHP server language integrated with a MySQL server. This role is configured according to Scenario 2 from section 4.1 primarily as a corporate directory server, news and weather search engine, and displays user status (i.e.,availability).

OpenVPN Server Version 2.2.0 running in server mode for creating a TLS based VPN tunnel with the local site.

freeRADIUS Server Version 2.1.12 running as authentication server (RADIUS+EAP) and using the WRT54GL as an authenticator for IEEE 802.11 clients at the local site, thus providing WPA2 Enterprise mode according to the IEEE 802.11i standard.

Nagios for remote monitoring of WRT54GL and the cloud site via SNMP.

Sendmail SMTP server Version 8.14.4 as part of the UMS. This SMTP server is used to send voicemail messages (via execution of a customized script) as attachments to emails that correspond to the voicemail box configurations.

Low Power logic These scripts are executed (based on day/night mode). These scripts, shut down certain functions at the local site (WRT54GL) as instructed by the cloud.

OpenSSH Server Version 6.0 is used to allow remote management and configuration of the cloud site via the CLI.

4.3.2 The Local site

The **local site** consists of SIP hardphones (Cisco 7940G and Aastra 6730i) and the WRT54GL (with customized DD-WRT software images). The local site's physical location was tested in both Lund and Stockholm, Sweden. The Internet access speed will be 10Mbps for download and 10Mbps for upload.

The WRT54GL will be configured in two possible configurations: only a SIP outbound proxy server or as a SIP outbound proxy server with an integrated broadband device. Details of these configurations are given below.

1. WRT54GL configured as only a SIP Router/Outbound Proxy server that incorporates the following features:[8]

Fixing of NAT	Registrations are addressed to the SIP B2BUA in the cloud. The IP address in the contact header field is substituted with the VPN IP address of the WRT54GL. This is necessary for local call establishment logic and so that SIP requests from the outside will be routed correctly.
Registration	Registrations destined to the cloud server are also stored in the local database, which allows operation independent of the cloud service, e.g. to locally handle a direct call from the Internet (on the WAN interface) or a local call (on the LAN interface).
From header	With either the WAN or VPN tunnel IP address, depending of the configuration.
Local subscribers	Creation of local subscribers and accounts that are not provided by a VoIP provider
Local call support	Internal calls (signaling and media) stay within the LAN
Aliases	Support of aliases for the locally registered subscribers for the purpose of quick dialling
IM	SIP/SIMPLE messaging support via the WRT54GL for the local accounts and subscribers
TFTP	Server for the SIP hardphones, configurations provisioning when booting on the LAN.
Firewall	For protection from both external and internal threats, e.g. DoS attack
VPN	VPN client to connect to the VPN server in the cloud
ssh	For remote or local configuration of the system via CLI
LAN	4 port managed VLAN switch
SNMP	For remote monitoring from the cloud service (using Nagios)

[8]This version of the local site will be implemented on a WRT54GL without any hardware modification, for this purpose a modification of the DD-WRT firmware is needed, using the firmware modification kit.

NTP Client software support is needed in order to synchronize (with current time of day) the WRT54GL with a publicly available time server. This is because WRT54GL lacks a time of day clock with battery backup.

syslog logging of system events, useful when troubleshooting

2. WRT54GL configured as a SIP Router/Outbound server as described previously and an integrated broadband device (IBD) that incorporates the following features:[9]

All The features from above

MMC/SD Card to expand the storage capacity in order to support additional feature installation, this hardware modification is described in [66]

WoL Server (WRT54GL does not have hardware support to wake up on magic packets, e.g to be a WoL client, thus additional hardware will be needed)

WLAN With WPA2 Enterprise Mode (2.4 Ghz radio for the AP)

DNS server Configured with the following options:

```
ptr-record=1.100.168.192.in-addr.arpa,"siplocal.lowpowersip.org"
ptr-record=1.0.8.10.in-addr.arpa,"sipcloud.lowpowersip.org"
srv-host=_sip._udp.lowpowersip.org,sip.lowpowersip.org,5060,1
srv-host=_sip._udp.lowpowersip.org,sipcloud,5060,2
naptr-record=lowpowersip.org,10,100,s,SIP+D2U,,_sip._udp.lowpowersip.org
```

The reasons for this configuration can be found in section 2.7.2.

DHCP server Configured with the following option parameter: `dhcp-option=66,''TFTP server IP''`, that points a SIP hardphone on boot to the TFTP configuration server. Reasons for this configuration can be found in section 2.9.1

IP Routing Services (static,RIP and OSPF)

Tx Power adjustments

QoS Bandwidth Management, QoS L7 Packet Classifier (L7-filter)

IPv6

web server

[9] This version of the local site will be implemented on a WRT54GL with an added SD slot, for this purpose a specific modification of the DD-WRT firmware will be needed.

5 Methodology

In this section we will define the methodology to be used in testing our IPT system designed for homes and SMEs with regard to low power consumption. First we will describe what we will measure and how we will make these measurements. Following this we will state the performance metrics that we will be evaluating using results from the experiments. This methodology relates to the main problem definition as stated in section 1.1 and the resulting design of the IPT system described in Chapter 4.

5.1 Experiment1: VPN link UDP and TCP performance measurement

In this experiment we want to measure the VPN link performance between the local and the cloud site in terms of its actual latency (packet delivery delay), UDP jitter, UDP bandwidth, UDP packet loss, and TCP traffic throughput. The UDP parameters should provide a rough estimate of VoIP QoS parameters. The UDP metrics will give a sense of the expected link performance for the maximum number of simultaneous calls that can be achieved (e.g. when using RTP over UDP as a media session transport mechanism with G.711 u-law CODEC). The actual TCP throughput is important for the IPT system since most of the services that run in the cloud site use TCP as a transport protocol. Prior to implementing the IPT system the TCP performance results can serve as a guide when testing the VPN link when an SME or home is considering deploying an IPT system that is to be linked to a cloud site. UDP jitter is the inter-arrival packet delay variation caused by queuing, contention, and serialization effects on the path through the network and is usually measured in milliseconds. Jitter is an important metric to consider, especially for real time data delivery such as voice. The UDP jitter is usually measured as the difference between samples, then divided by the number of samples (minus 1). The actual latency between the sites can be measured as a one way delay (OWD)[10] in each direction using the network time protocol (NTP)[11] for global time synchronization. OWD will provides the actual delay for the media streams in each direction within the VPN. Packet loss is the number of packets or percentage of packets lost in a one-way media stream. Packet loss is easily computed when there is a reporting mechanism between the sites (such as RTCP reports). This value expresses the number of packets received relative to the number of packets that were sent from each site. There are a lot of variables that can affect the UDP parameters, such as the latency (due to asymmetric routes between the local and cloud sites), peak hours of traffic, route flapping due to congestion, utilized LAN, access link congestion, etc. The overall latency according to ITU-T G.114 recommendation should be a maximum of a 150 ms

[10]The OWD is the time difference between the occurrence of the first bit of a packet on the first observation point, e.g., the transmitting monitor interface, and the occurrence of the last bit of a packet on the second observation point, as defined in RFC 2679[76]. In contrast the round trip delay (RTD) is considered to be the time interval between the time instant a request packet is sent by a source node and the time instant a response packet is received from the destination time, as defined in RFC 2681[77].

[11]NTP is defined in RFC 1305[78]. It is a protocol used to synchronize the clock of a client to a reference time source, such as a radio or satellite receiver. It can provide accurate time typically within a millisecond on LANs and up to a few tens of millisends on WANs.

one-way. For the overall psychoacoustics of the voice quality, the jitter value depends upon the de-jitter buffer of the terminal equipment used. However, many VoIP service providers state that the jitter should not be larger than 1ms (for a mouth-to-ear stream). According to vendors, such as Cisco and Avaya, the average one-way Jitter should be targeted under 20 ms. Using the G.711 u-law CODEC the packet loss should not be larger than 1%. The bandwidth of the voice media stream depends upon the CODEC that is being used. For a G.711 u-law voice stream the overall bandwidth needed for a one-way media stream is 80kbps (CODEC payload (64 Kbps * 20 ms) + Ethernet overhead and 802.1Q (22B) + IP overhead (20B) + UDP and RTP overhead (20B) * packets per second(1000ms/20ms)). In order to simulate a VoIP call that uses G.711 u-law the UDP packet size should be 172B. In the VPN link performance experiment we will evaluate three different scenarios.

The first scenario concerns measurement of OWD and will consist of four sub-scenarios. The first two sub-scenarios are measurements of RTD in the VPN tunnel and RTD over the public Internet, to enable us to compute the cost of the VPN mechanism that is being used. The other two sub-scenarios are measurements of OWD within the VPN, where one sub-scenario is for an emulated media stream that originates from the cloud site and runs to the local site and the other is for an emulated media stream that originates from the local site and runs to the cloud site. RTD related sub-scenarios will be run from the local site. The RTD sub-scenarios will be run once for 2.30 hours when the cloud and local site are not utilized, a measurement will be made every second which means that we should have 9000 measurements (sample size) during that amount of time. The OWD sub-scenarios will run for 10 minutes (simulating a regular duration of a phone call) when the cloud and local site are not utilized, a measurement will be made of every arriving UDP packet, which yields a sample size of 30000 measurements. These sample size, should give us a correct estimate of the true value, if no variations are observed.

The second scenario of the VPN link performance experiment regards the jitter, bandwidth, and packet loss measurements of a UDP stream that will be run simultaneously (using a traffic generator tool) from/to the cloud and from/to the local site. The scenario has three sub-scenarios with the independent variable being the bandwidth of the traffic generator tool, in order to find the actual performance of the access links at both sites (Hence for this experiment the physical location of the cloud site is in Skopje, Macedonia, connected to the Internet via DSL subscription of 10Mbps download and 768Kbps upload speed, and the local site's physical location is in Lund, Sweden with Internet access speed of 10 Mbps for download and 10Mbps for upload, which means that the theoretical symmetrical bandwidth is 768kbps). Each of the sub-scenarios will be running with the duration of each run being 30 minutes, thus simulating a long phone call. The size of the UDP packet will be 172 bytes (160 bytes of payload + 12 bytes for RTP) and such a UDP packet will be sent every 20ms (approximately simulating a G.711 u-law VoIP call). We expect, approximately 562500 UDP packets to be generated in each direction per sub-scenario iteration (e.g. for bandwidth set to 500kbps), which represents the sample size to be

measured. We will run each of the sub-scenarios for four times[12].

For the third test scenario of the VPN link performance experiment we will find the actual TCP throughput with using a traffic generator tool. The independent variable will be the advertised TCP window size in range from 2,4,8,16,32,64,128, and 256KB, while sending a 6MB large file simultaneously in both directions. Note that the TCP window size should also be supported by the TCP window size parameters within the OS, therefore tweaking of this parameters will be needed (e.g. in Ubuntu Linux the default values for both the maximum advertising and receiving TCP window sizes is 128KB. This can be seen in `/proc/sys/net/core/wmem_max` for advertising and `/proc/sys/net/core/rmem_max` for receiving TCP window sizes). Using the formula for bandwidth delay product, we could estimate the best performance of TCP regarding the utilization of a certain link. The formula for bandwidth delay product is: (smallest link in path bits per second / 8) * RTD in seconds i.e. the ping time to the destination. Therefore we would expect that for a symmetrical link of 768kbps with RTD of 77ms the best performance could be expected when the sending TCP end is able to send at least 7KB of data at once (spread across a number of packets), and the receiving TCP end is able to receive up to 7KB of data at once (spread across a number of packets). This scenario yields eight sub-scenarios. Each of the sub-scenarios will be run for four times (See Footnote12) where the TCP window size factor consists of eight levels being the different values of TCP window sizes.

5.2 Experiment2: IPT system main call scenarios

In order for us to measure the IPT system's performance for each of the main call scenarios (1, 2, 3, and 4) from section 4.1 we will consider the steps explained in each of the scenarios to be strictly defined tasks that we will go through. For each task we will measure the task's success (i.e., this test concerns functional correctness - rather than details of performance). Therefore for each scenario we will need an actual implementation of the local and the cloud site (their design was presented in section 4.3) with all its features and configurations according to the scenario's needs. The functional correctness of the scenarios will be presented in form of a message call flow diagrams, that will be distilled from a various packet captures (merged from monitoring stations placed on key point in the IPT system e.g. the UAs at the local site, the WRT54GL, the cloud site, and the external UAs). We do not expect variations in the task success in

[12]This is in accordance to a customized unrandomized one (primary) factorial design of experiments (described in [79]). In this scenario we have one factor (with the independent variable being the bandwidth input to the tool) and the three levels or sub-scenarios being the three different input values of the bandwidth. The total number of runs for each scenario is calculated according to the following formula: $N = k * L * n$ where k = number of factors (= 1 for this design), L = number of levels (= 3 for this design), n = number of replications (=4 for this design). The number of replications is usually determined as the number of factors plus one for a very controlled environment. Even though in this scenario we will measure a large number of samples, still there is a big part of the environment that can not be controlled by us (such as the VPN link path with its actual network elements and their configuration that are part of various autonomous systems on the public Internet network). Therefore in order to increase the statistical precision and accuracy we will repeat the test in a controlled environment. Also, balance dictates that the number of test repetitions be the same at each level of the factor

either the sequence message call flow of the VoIP protocols (for the same setup and configuration of the IPT system). Since the result of these task is either pass or fail we can run them only once; however, in regards to reliability of the data for the VoIP message call flow dialogues we will run the tasks three times and if there is a variations between the dialogues within these three iterations we will present them in our results.

5.3 Experiment3: Main low-power mechanism scenario

In order for us to measure the main low-power scenario from section 2.10 we will consider two test scenarios. As a first test scenario we consider the steps explained in the main low-power mechanism scenario to be strictly defined tasks that we will go through and than measure each task's success. To actually perform the task measurement we will need an actual implementation and configuration of the local and cloud site (their design is presented in section 4.3) with all its features and configurations according to the scenario's needs. The second test scenario of the main low-power mechanism experiment concerns the overall power consumption and saving potential of the local site when running the day or night scripts from the cloud. Therefore we will measure the power consumption when the day script is executed and when the night script is executed in order to measure the overall power savings potential of the local site. The second test scenario can be designed as a single factor (low-power script) with two levels: day and night (See Footnote12). Since this environment is a very controlled one and the same code is going to be executed with the same hardware settings (of WRT54GL) we do not expect that there will be variations in the power consumption values between the measured samples that belong to the same factor level. Please note, that even though the environment will be controlled in this experiment, we will still run each of the sub-scenarios or levels (day or night) for four times, since the sample size is not that big. In each step of the sub-scenarios the power consumption will be measured (excluding the power consumed by the PSU) for five minutes before the execution of the script and for five minutes after the script execution which gives us 300 measurements (i.e., a sample size of 300).

5.4 Experiment4: WRT54GL's power consumption

For the purpose of this project we have developed two versions of the DD-WRT firmware (with the SD modification (sdmod) and no modification (nomod)) for the purpose of the local site (WRT54GL). This experiment gathers power consumption data for both the sdmod and nomod versions of WRT54GL in various configurations. The main experiment has two main test scenarios (sdmod and nomod), each of the main test scenarios has four sub-scenarios with the same configuration. These sub-scenarios are:

WRT54GL's CPU power consumption sub-scenario. In this test sub-scenario the WRT54GL is booted with no LAN (switch) ports active and no WLAN active (WiFi radio is turned off). This test sub-scenario has three different subsub-scenarios. In the first subsub-scenario the CPU is clocked at 183Mhz which is the lowest configurable value for the CPU.

In the second subsub-scenario the CPU is clocked at 200Mhz which is the default value for the CPU. In the third subsub-scenario the CPU is clocked at 250Mhz which is the highest configurable value for the CPU. Each subsub-scenario is executed for four consecutive times (please see Footnote12) each of these subsub-scenarios has a duration of 200 seconds (which leads to a sample size of 200) and at each step the overall power consumption is measured for the WRT54GL (excluding the PSU). Since the same code is going to be executed with the same hardware settings (of the WRT54GL) we do not expect that there will be variations in the power consumption values between the measured samples that belong to the same factor level. However, having multiple iterations of the experiments, increases the precision and the accuracy of our estimate of the true value.

WRT54GL's WiFI power consumption sub-scenario.

In this sub-scenario of the WRT54GL's power consumption experiment, the WRT54GL is booted with no LAN (switch) ports active and the WLAN active (i.e., the WiFi radio interface is turned on). The CPU is clocked at 200Mhz. The WRT54GL is configured as an AP with the default wireless settings and no client associations are established. This subsub-scenario is executed for four consecutive times (please see Footnote12) each of the subsub-scenarios has a duration of 200 seconds (which also defines the sample size as 200) and at each step the overall power consumption is measured for the WRT54GL (excluding the PSU). Since the same code is going to be executed with the same hardware settings (of the WRT54GL) we do not expect that there will be variations in the power consumption between the measured samples that belong to the same factor level. However, having multiple iterations of the experiments, increases the precision and the accuracy of our estimate of the true value.

WRT54GL's switch power consumption sub-scenario.

In this sub-scenario of the WRT54GL's power consumption experiment, the WRT54GL is booted with all five LAN (switch) ports active (Ethernet link established and moderate traffic is being exchanged between the devices on each port) and no WLAN interface is active (i.e., the WiFi radio interface is turned off). The CPU is clocked at 200Mhz. This subsub-scenario is executed for four consecutive times (please see Footnote12) each subsub-scenario has a duration of 200 seconds (i.e., leads to a sample size of 200) and at each step the overall power consumption is measured for the WRT54GL (excluding the PSU).

Since the same code is going to be executed on the same hardware settings (of the WRT54GL) we do not expect that there will be variations in the power consumption values between the measured samples that belong to the same factor level. However, having multiple iterations of the experiments, increases the precision and the accuracy of the true value.

WRT54GL's switch and WiFi power consumption subsub-scenario.

In this sub-scenario of the WRT54GL's power consumption experiment, the WRT54GL is booted with all five LAN (switch) ports active (Ethernet link established and moderate traffic is being exchanged between the devices on each port) and WLAN interface is active (i.e., the WiFi radio interface is turned on). The CPU is clocked at 200Mhz. This subsub-scenario is executed four consecutive times (please see Footnote12) each time with a duration of 200 seconds (which leads to a sample size of 200) and at each step the overall power consumption is measured for the WRT54GL (excluding the PSU). Since the same code is going to be executed with the same hardware settings (of WRT54GL) we do not expect that there will be variations in the power consumption values between the measured samples that belong to the same factor level. However, having multiple iterations of the experiments, increases the precision and the accuracy of our estimate of the true value.

5.5 Experiment5: Measuring the actual VPN link capacity for the maximum number of simultaneous or concurrent calls in regards to their QoS parameters

The idea of this experiment is to stress the actual IPT system solution over the path between the local and the cloud site with real VoIP calls from SIP UACs. For this experiment there are five test scenarios, namely when running one, two, three, four, and five simultaneous calls using G.711 u-law CODEC[13]. Each of the scenarios is executed for four times (see Footnote12) for a duration of 10 minutes (yielding a sample size of approximately 60000, 120000, 180000, 240000, and 300000 UDP packets for measurements with one, two, three, four, and five simultaneous calls each call with a duration of 10 minutes). Each of the five UACs are connected to the WRT54GL's switch ports. The calls are initiated towards the cloud site to the centralized conferencing server where they establish a conference call. Note that the WRT54GL is configured to serve as a SIP outbound proxy (incorporating a SIP registrar and local routing logic) from the perspective of the UAC. This experiment uses two SIP hard phones and three SIP softphones. There are QoS service parameters configured on the WRT54GL on Layer2 and Layer3 for the VoIP traffic. The VoIP traffic belongs to a separate VLAN3 that is being tagged and prioritized at the switch. In

[13]The number of the simultaneous calls was motivated by the analysis of the data after Experiment 1 was conducted, as we saw that there is high packet loss at the 0.5Mbps level.

the Linux router logic a hierarchical token bucket (HTB) packet scheduler is configured for the VPN interface. There are six priority levels or bandwidth reservations for the desired services. We have defined the highest priority level for the packet scheduling to be according to the netmask of the VoIP VLAN (e.g. in our case this is 192.168.100.64/26). During this experiment the CPU statistics will be logged from the WRT54GL (using vmstat) and its overall power consumption will be measured.

5.6 Experiment6: Auto bootstrapping and locating of SIP servers in the IPT system.

In this experiment we will perform tests according to strictly defined tasks and measure the task success. Here the tasks concern the bootstrap mechanism (that is assisted by the DHCP and TFTP servers) and the locating of SIP servers (for redundancy). This experiment concerns sections 2.9.1 and 2.7.2. First we will power-on and boot a SIP hardphone on the local site's network. We will enable the DHCP client on the hardphone. On the TFTP server we will place the necessary configuration files (SIP firmware, phone model configuration script, and phone specific configuration according to the phone's MAC address) for the phone. In the configuration files we will specify the SIP proxy servers, addresses as FQDNs. This will force the UAC depending on its capabilities to either first perform an DNS NAPTR record request or it will directly perform a DNS SRV record request towards the WRT54GL (which has a DNS server configured with NAPTR and SRV records) and than resolve (according to the priorities in the SRV response) the SIP proxy server's IPv4 address will be used to send the first REGISTER SIP message. We will use higher priority for the cloud site and lower priority for the local site SIP server. In order to test the redundancy we will disable the cloud site which should result in a time out of the first REGISTER request (note that we expect to receive a number of `ICMP destination unreachable (port unreachable)` messages from the cloud SIP registrar server, each caused by REGISTER attempts and their intervals will be due to the UAC as controlled by the timer F of RFC 3261[10] as described in section 4.3 from RFC 3263[50]). After this we expect that the UAC will contact the second priority server as specified by the SRV responses, which in this case is the WRT54GL. In this experiment we will monitor the traffic for TFTP, DNS, and SIP dialogues in order to measure the task's success. The related protocol dialogues and their timings will be shown as a filtered Wireshark output.

5.7 Experiment7: Measuring SIP's RRD and SRD for the WRT54GL according to RFC 6076

In order to perform SIP performance testing of the proxy and registration servers that is embedded in the WRT54GL an explanation is needed of what is going to be tested. The Registration Request Delay (RRD) and Session request delay (SRD) will be measured from the UAC's point of view. As defined in [19], RRD is a measurement of the delay in responding to a UAC's REGISTER request. The RRD is calculated using the following formula:

RRD = Time of Final Response(t2) - Time of REGISTER Request(t1)

In a successful registration attempt, the RRD is defined as the time interval from when the first bit (t1) of the initial REGISTER message containing the necessary information is passed by the originating UA to the intended registrar, until the last bit (t2) of the 200 OK is received indicating that the registration attempt has been completed successfully. This dialog includes an expected authentication challenge prior to receiving the 200 OK as shown in Figure 27.

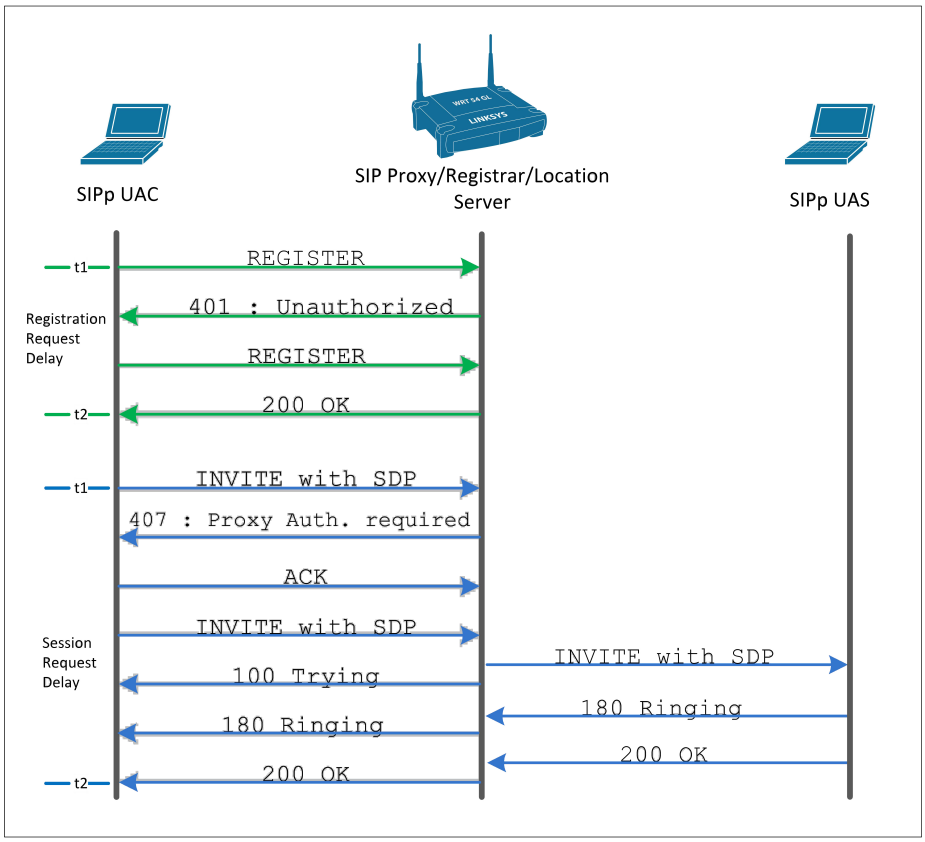


Figure 27: SIP dialogues with RRD and SRD defined

Measuring SRD is utilized to detect failures or impairments causing delays in responding to a UAC's session request. The result of an attempt to establish a session can be either a success or a failure. SRD will be measured only for a successfully established session [19]. SRD is calculated using the following formula:

SRD = Time of Status Indicative Response (t2) - Time of INVITE (t1)

According to [19], SRD is defined as the time interval from when the first bit(t1) of the initial INVITE message is sent by the originating user agent to the intended mediation or destination agent, until the last bit(t2) of the first non-100 Trying provisional response (180 Ringing or 200 OK in some cases) is received indicating an audible or visual status of the initial session setup request (see Figure 27). There will be two main scenarios in this experiment (SRD and RRD). Each scenario will have six sub-scenarios namely when 1, 5, 10, 25, 35,

and 50 calls are being simultaneously initiated. In each sub-scenario the call attempts are executed simultaneously and consecutively for 10 times every 3 seconds. We will run the SIP signalling over UDP with authentication since this is the most common scenario for SIP. The SIP dialogues for all of the sub-scenarios will be monitored from the UAC's perspective. At the server side (the WRT54GL) we will measure the CPU usage. The selected values for different numbers of registrations and call setup were motivated by our own experience and from the small (<50 employees) and micro(<10 employees) SME definitions as defined in [17]. If an average sized SME has fifty phones, then the peak of 50 registrations will happen after power failures, as the phones are usually not protected by an uninterruptable power supply, even though the servers might be.

6 Experimental testbed

In this chapter we describe the testbed for the experiments and their related scenarios as described in Chapter 5. The testbed, can be seen as a model for the actual configuration and implementation of an IPT system for homes and SME. The generic configuration of the cloud and the local site can be found in section 4.3. Figure 28 represents the actual testbed. During a period of five months, the IPT solution was extensively tested and used with all of its features. Here we will explain additional configuration information and details. Apendix A represents a methodological mindmap diagram that summarizes the experiments that were performed and their related scenarios and subscenarios. This appendix should help the reader of this thesis to understand the detailed methodology employed when conducting the experiments.

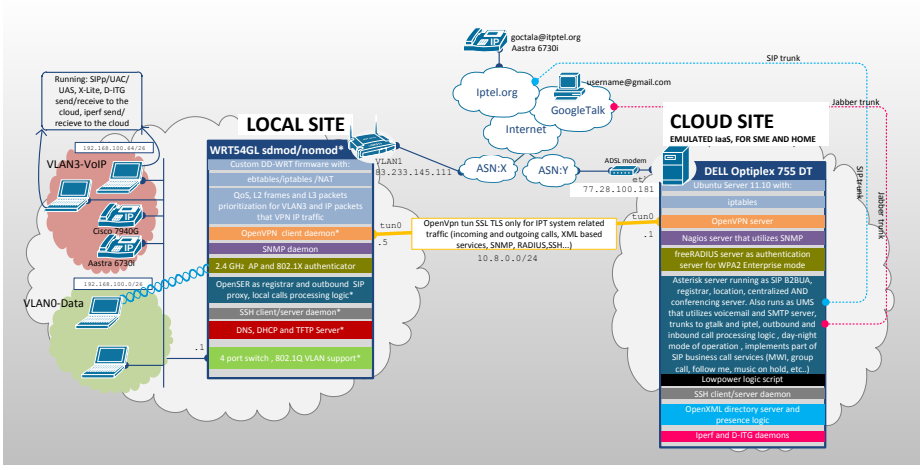


Figure 28: A network and features diagram, of the implemented testbed environment, where all of the experiments were conducted and measurements were made. The implementation of this diagram is also related to the IPT system design that was discussed and presented in Chapter 4.

6.1 Asterisk

Asterisk is a complete IP PBX that is software based, written in the C programming language, and runs on Linux. It is VoIP hybrid platform that can communicate using many protocols and can interoperate with a PSTN network by utilizing appropriate telephone interfaces (such as a E1 PCI card). Asterisk does PBX switching, acts as a media gateway, provides CODEC translation, and supports applications such as voicemail, interactive voice response, conferencing, and more. It is used by small businesses, large businesses, call centers, carriers, and governments worldwide. Asterisk is free and open source [80]. The channel API handles the type of connection a caller is arriving on, be it a VoIP connection (SIP, Skinny, IAX2, H.323, etc.), ISDN, PRI, etc. In our testbed Asterisk was implemented at the cloud site and it was configured as a SIP B2BUA server, registrar server, centralized conferencing server, voicemail server, and UMS, (by integrating the dialplan and contexts logic by utilizing

the necessary channels, voicemail application, and SMTP server). For the IPT system we have compiled Asterisk version 1.8.4.4 from source and resolved all of the dependencies. For our testbed we have configured three channels: SIP, Jabber (for the Gtalk trunk), and DAHDI dummy (emulating a reliable time source from the PC clock for the MeetMe() application which is an RTP mixer in a centralized conferencing mode). The SIP channel is configured by customizing the `extensions.ael`, `sip.conf`, and `extensions.conf` files. The `sip.conf` file is where the SIP users are defined. How the users are processed when arriving on a channel is based upon context and the dial plan defined in the `extensions.ael` and `extensions.conf` files. The Jabber channel is defined in the `jabber.conf` file and the gtalk trunk is defined in `gtalk.conf`. The centralized conferencing or `MeetMe()` application is defined in `meetme.conf`, where one conferencing room was created at extension 900 from the local site, along with a password for the participants in order to enter the conference. The voicemail application is defined in the `voicemail.conf` file, where five voicemail inboxes were created for five exttensions, the users from the local side could access their voicemail by dialing *72 and then an IVR system asks the user to authenticate with a password in order to access their voicemail. Note that applications are invoked based upon the dialplan and context logic that resides in `extensions.conf`. The UMS utilizes the dialplan and its context in order to determine (if busy with `486 Busy Here`, timed out with `408 Request Timeout`, or unavailable with `480 Temporarily Not Available` messages) when a caller should be redirected to the callee's voicemail. After the caller leaves a voice message (that is locally stored), the called user is notified via the MWI on his hardphone (with the NOTIFY method) and the called user receives the voicemail message as an attachment to an email sent from the configured sendmail SMTP server. The UMS utilizes the `system()` command in order to do this. For the testbed we used a gmail account in order to receive the notification emails from the UMS. Part of the SIP buisness call services were configured using the Astersik dialplan, but for others the dialplan logic of the UAC was used (this logic was downloaded via the configuration file from the TFTP server). In order to configure the day-night mode operation of the IPT system the `GotoIfTime()` application was used and configured to load two different sets of dialplans and it utilizes both the `system()` command (for manual execution through a SIP hardphone) and the `cron` (for automatic execution based on time of day) in order to execute the day or night low power script via SSH on the WRT 54GL. All of the configuration files regarding Asterisk can be found in Appendix B and the lowpower script can be found in Appendix C. The scripts execution mechanism is thoroughly described in section 6.6.

6.2 OpenSER

OpenSER is an open-source SIP proxy server (compliant to RFC3261 [10]) that is best described as a SIP express router (SER). It is able to manipulate the SIP headers and route packets at extremely high speeds, which makes this proxy extremely fast. OpenSER is being used by large VoIP providers and for embedded IP PBXes with very limited processing power. Third-party modules give OpenSER extreme flexibility to play roles for which it was not originally intended, such as NAT traversal, IMS, load balancing, and other functionalities.

SER was originally developed by the FhG Fokus research institute in Berlin, Germany, and released under the GPL license. It is written in ANSI C and can be easily ported to any platform. OpenSER is primarily used as a SIP proxy and registrar. The main configuration file for OpenSER is `openser.cfg`. This configuration file controls which modules are loaded and their respective parameters. The `openser.cfg` file has seven sections or blocks:

Global definitions	This portion of the file contains working parameters for OpenSER, including the listening ip:port pair for the SIP service and debug level.
Modules:	A list of external libraries required in addition to the core functionality. Modules are loaded with loadmodule.
Modules configuration	Modules can have parameters that needs to be set appropriately. These parameters are configured using modparam(modulename, parametername, parametervalue).
Main routing block	This is where the SIP message processing starts for each received message.
Secondary routing blocks	The administrator can define new routing blocks using the command route(). These routing blocks act as subroutines in the OpenSER script.
Reply routing blocks	Reply routing blocks are used to process reply messages, usually 200 OK.
Failure routing blocks	Failure routing blocks are used to process failure conditions such as busy or timeout.

In our testbed we use OpenSER to implement an outgoing SIP proxy and registrar server for the local site from the IP telephony system in both versions (sdmod and nomod) of WRT54GL running with a custom DD-WRT firmware image. The `openser.cfg` file, that was used during the experiments is included as Apendix D.

6.3 SIPp

To simulate the UAC in the RRD, and UAC and UAS for the SRD dialogs we used a tool called SIPp[81]. SIPp is an open source tool for evaluating and benchmarking the performance of SIP servers. SIPp can generate all the SIP messages involved in session establishment and termination, enabling it to emulate several predefined or customized user agent scenarios. Amongst its features are extensive statistics, periodic statistical dumps, a call rate distribution definition, IPv6 support, TLS support, SIP authentication, selection of transport protocol, multiple sockets, multiplexing with retransmission management, RTP media stream transport via RTP Echo, and PCAP based replay. In our experiment to measure SIP's RRD and SRD for the WRT54GL according to RFC 6076, SIPp

was emulating a UAC (for RRD) and UAS (only for SRD) to perform call setup and termination of multiple sessions with the WRT54GL's registrar and proxy server. For the RRD measurements, we prepared a separate custom scenario (written in XML) to emulate the registration request flows illustrated in Figure 27. The two scenarios emulate the client side for registration with authentication and session setup with authentication (when used as an UAS). The configuration of the XML scenario files is included in Appendix H. SIPp version 3.2 was running on both an Ubuntu 11.10 OS installed on an Intel®Core2™DUO CPU P8700@2.53GHz with 4GB of RAM for the UAC (for RRD only) and on Ubuntu 10.10 OS installed on an Intel®Dual™core CPU T4500@2.3GHz with 4GB of RAM.

6.4 Traffic generators

In our experiments we have used both Distributed Internet Traffic Generator (D-ITG)[82] version 2.8.0-rc1 to stress the performance of the VPN link between the local and cloud site for delay, an UDP (jitter, bandwith, and packet loss); and iperf[83] version 2.0.5 for TCP throughput. D-ITG is capable of producing traffic according to defined Inter Departure Time and Packet size stochastic processes. These stochastic processes are implemented as independent and identical distributions - such as uniform, constant, Cauchy, normal, Poisson, gamma, and Pareto random variables. D-ITG is able to emulate network (ICMP), transport (UDP, TCP), and application layer (DNS, Telnet, and VOIP) protocols. Both sending and receiving sides of the traffic flows can log information in order to calculate One-Way Delay (OWD), Round-Trip Time, packet loss, jitter, and throughput statistics of the generated traffic patterns. In our experiments we were using D-ITG for two scenarios: (A) to simulate G.711 u-law VoIP traffic using UDP as the underlying transport protocol (in order to assess the capacity of the VPN link for simultaneous calls) and (B) to simulate TCP traffic for other applications that run in the cloud such as the XML Directory web server, NMS administration via http, etc. (in order to evaluate the achievable throughput). In order to measure the OWD we have used D-ITG (sender) to send and timestamp equally sized UDP packets of 172 bytes at a constant rate of 50 pps (emulating a G.711 u-law media stream), and a D-ITG receiver where the packets were recieved and timestamped. For the OWD measurement we have used NTP clients on both machines (one in the cloud and the other at the local site) which were synchronizing with stratum2 NTP servers. In order to emulate the VoIP UDP streams we configured D-ITG to generate equally sized packets of 172 bytes at a constant rate of 200/250/350 pps (emulating 4/5/6 one-way simultaneous G.711 u-law media streams), which theoretically should result in 323.2/404/484 Kbps traffic rates (together with Layer2 and Layer3 overheads). This configuration simulates the media stream of VoIP traffic between the local and cloud sites. The traffic generating hosts were configured to both send and receive traffic so that the link was utilized with media streams in both directions, thus emulating 4/5/6 media streams of G.711 u-law VoIP calls. Iperf can create TCP and UDP data streams and measure the throughput of a network that is carrying them. The user can specify various parameters for the tool, among others (rather than using D-ITG) iperf can specify different TCP window sizes and can be used in both sender and receiver modes. In order to emulate and sense TCP throughput for the VPN link we

have configured iperf in both sender and receiver modes at each site where a 6MB MP3 file was sent between the two sides simultaneously (by running a script to initialize the process on both sides) with 2/4/8/16/32/64/128/256KB TCP window sizes. The commands that were executed during the experiments that involve D-ITG and iperf can be found in Appendix G.

6.5 Power measurements

For the purpose of power measurements we configured a testbed as shown in Figure 29. We used a MASTECH-345 digital multimeter as an ampmeter between the PSU and the WRT54GL. This multimeter has an RS-323 port that was connected to a PC via hyperterminal which logs the values from the measurements every second.

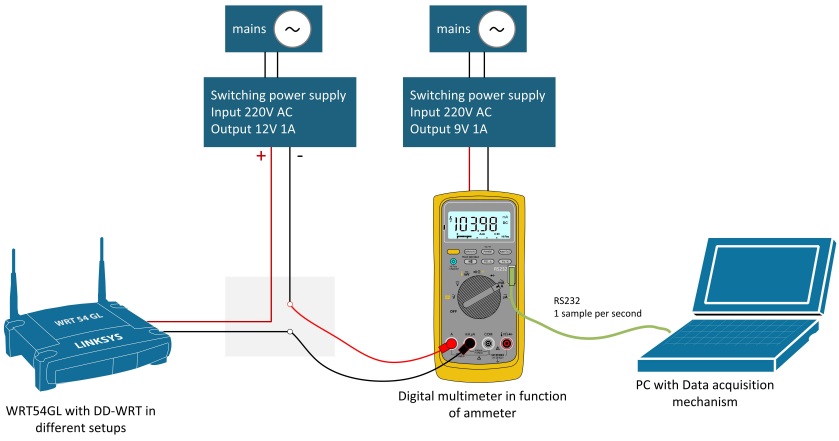


Figure 29: Power measurement, testbed setup, for all of the experiments that require power measurement of WRT54GL

6.6 Low-power mechanism script

In order to enter a lowpower state of the WRT54GL we execute a night-lowpower script (the trigger is based on time of day according to the closing hours of the SMEs) from the cloud site. In order to exit the lowpower state we execute a day-lowpowerscript (this is triggered based on time of day according to the opening hours of the SMEs). Our implementation uses a `cron` daemon (which executes, system commands based on the time-of-day) in order to execute the scripts, and it uses the Asterisk's `System()` function in the dialplan logic in order to enable manual execution of the scripts, e.g. from a SIP hardphone. We authorize the cloud site to administer the local site by utilizing RSA based public key authentication for the SSH login. This means, that we have created a key-pair (private and public key) using the `ssh-kegen` tool (by running the command `sudo -u asterisk ssh-keygen`). The public key is placed in the local site (DD-WRT) as a value in the NVRAM's parameter `sshd_authorized_keys=''public key''` and the private key is placed

in the file /var/lib/asterisk/.ssh/id_rsa at the cloud site. The user "asterisk" was created on both the local and the cloud site. This user (at the cloud site) has the privileges to execute the low-power scripts and to login via SSH and issue commands at the local site.

The WRT54GL will (if it is aware of a public key for the asterisk user that the cloud site is requesting to be logged in as, and if the cloud site quotes that public key) create an RSA challenge encrypted with that public key. If the cloud site knows the corresponding private key, it can decrypt the challenge and send it back to the WRT54GL, which now knows that the cloud site must have the secret and should therefore be allowed access. This form of authentication requires no trust of any host, network, or software - it relies purely on possession of the private user key by the client. The scripts can be found in Appendix C.

6.7 SIP hardphones and softphones

As SIP hardphones we have used two commercial models: a Cisco 7940G (flashed with P003-8-12-00 SIP firmware) and the other is a Aastra 6730i (flashed with V3.2.2.2044 SIP firmware). Both of these SIP hardphones have support for DHCP, TFTP, DNS A, PTR, and SRV record lookups, 3 SIP lines, XML browser capabilities allowing access to customized services and applications, shared call and bridged line appearances, call forward, call transfer, call waiting, intercom, 3-way conference, etc. Additional information about the phones can be found in [84] and [85], respectively. When each of the phones boots it access the TFTP server and checks if there is a firmware image file on the TFTP server. If there is, it checks if the version of the installed firmware (on the SIP hardphone) is older than the version stored on the TFTP server. If it is older, then the newer firmware is downloaded and installed. After this the configuration files are downloaded in order to configure the phone. For example, for the Aastra SIP hardphones there are four files that a SIP hardhone will search for, on a configured TFTP server:

1. The firmware file:

<phone model>.st This file (e.g. 6730i.st) contains information about the specific model of the phone and contains the language packs to load into the phone, this is the firmware image file.

2. The configuration files consist of three files called:

aastra.cfg This file contains general configuration information.

<model>.cfg (for example, 6730i.cfg) - This file contains model specific information, where "model" should be the same string that is used for the model name (e.g. 6730i.cfg)

<MAC>.cfg (for example, 00085D132EDF.cfg) - This file contains configuration information about the specific phone based upon its MAC address.

If there are different parameter values in each of the files the last configuration file to be loaded has precedence. As a result the <MAC>.cfg has higher priority

than <model>.cfg, which has in turn higher priority than aastra.cfg. The configuration files that were used in the experiments are included in Appendix E.

As softphones we have used X-Lite V4 from CounterPath corporation[86]. X-lite combines voice and video calls in a user-friendly interface. This softphone has built in standard telephone features such as: call display, MWI, speakerphone, mute, redial, hold, and call history. Among the standard features, the X-Lite softphone also incorporates enhanced features and functions, some of them are: video, IM, managed contact list, acoustic echo cancellation, voice activity detection, STUN and ICE NAT traversal, compliance to RFC3261 etc.

6.8 DNS, DHCP, and TFTP

In order to achieve a rapid response for DNS and DHCP queries/requests, we have implemented dnsmasq server version 2.61 at the local site (i.e., on the WRT54GL). Dnsmasq is a lightweight, easy to configure DNS forwarder and DHCP server. It is designed to provide DNS and, optionally DHCP, service to a small network. It can also respond to queries for the names of local machines which are not in the global DNS. The DHCP server is integrated with the DNS server and allows machines with DHCP-allocated addresses to appear in the local DNS with names configured either for each host or in a central configuration file. Dnsmasq supports static and dynamic DHCP leases and BOOTP/TFTP/PXE for network booting of diskless machines [87]. Dnsmasq was configured with PTR, SRV, and NAPTR records. The DHCP response provides the TFTP server's IP address to the SIP hardphones. The configuration files of dnsmasq can be seen in Figure **??** (within the testbed description). As the TFTP server we have installed atftp version 0.7-9 on the WRT54GL. Atftpd is a client/server implementation of the TFTP protocol that implements RFCs 1350, 2090, 2347, 2348, and 2349. The server is multi-threaded and the client presents a friendly interface using libreadline [88]. The configuration file for atftpd can be seen in Appendix F.

6.9 OpenVPN

OpenVPN is a proven, open source, robust, and secure VPN solution that is cross-platform capable [89]. The mechanism behind OpenVPN for creating virtual interfaces is provided by the operating system's TUN/TAP driver. Security in OpenVPN is handled by the OpenSSL cryptographic library which provides strong security using the Secure Socket Layer (SSL) protocol. Certificates are used for authentication, and symmetric and asymmetric ciphers for encryption. The cross platform support in OpenVPN makes it a flexible solution that can be installed on embedded routers and in the server in the cloud. In our IPT system we have installed OpenVPN version 2.2.0 on both the cloud (running as a server) and the local site (running as client). These roles were assigned in this way as the client can make a request to the server - thus opening an outgoing hole in the firewall to the server. OpenVPN is used to create the VPN tunnel between the cloud site and the local site, thus ensuring authentication, confidentiality, and integrity of the data exchanged between these two sites. The configuration file for both the local and the cloud site can be seen in Appendix H.4.

6.10 Nagios

Nagios is a free, open-source web-based network monitoring system (NMS) developed by Ethan Galstad [90] [61]. Nagios is designed to run on Linux, but can be also be used on UNIX variants. Nagios monitors the status of host systems and network services and notifies the user of problems. In common with many open source utilities, installation requires a degree of system administrator experience. Nagios is primarily a tool to diagnose, prevent, and deal with network problems. The increasing deployment of high bandwidth links and mission critical application over LANs connected to WANs demands more effective monitoring tools. A large number of plug-ins are available from the Nagios Library, thus Nagios can be customized according to the user's requirements. Nagios allows users to develop custom made checks of their network devices. In our IPT system we have installed and used Nagios version 3.3.1 to monitor both the cloud site and the local (DD-WRT on WRT54GL) site by utilizing SNMP. The monitoring checks the CPU load, network interface(s) status, PING reachability, SSH server status, RAM usage, uptime, HTTP server status, DNS (A, PTR, SRV, and NAPTR lookups) server status, number of WiFi associated clients, total number of processes, and the number of current users. The definition of the hosts and their related checks are located in `/usr/local/nagios/etc/objects/switch.cfg`. Appendix I presents the configuration files for the hosts.

6.11 Open 79XX XML Directory server

Open 79XX XML Directory is an application designed to fully utilize the display features on Cisco 79XX IP Phones. It includes on-screen phone directories, search capabilities, text memos, interactive user status, and more. The XML directory is accessed by pressing the Services button on the physical phone [75]. We have installed Open 79XX XML Directory server version 1.2 at the cloud site in order to provide the Cisco 7940G SIP hardphone (through its XML browser capabilities) with the following services (when a "services" button is pressed on the phone): a centralized directory search and listings, memo announcements, RSS news and weather feeds (e.g. from http://www.bbc.co.uk, http://weather.yahoo.com/, etc.), interactive user status (indicating whether the user is available or unavailable of a given contact found as a result of a search). For the purpose of this thesis we have also shown that is possible to use the Aastra's 6730i XML browser to use the centralized directory search and to display user status - in the same manner as with the Cisco hardphone. Adding contacts, memos, RSS weather, and news headlines is achieved through a web based interface. For additional configuration, such as customization, additional menu items and editing of the PHP files are required. An example for some of the services that Open 79XX XML Directory server provides can be found in Appendix J.

6.12 FreeRadius and OpenSSL

The FreeRADIUS server is a daemon for unix and unix like operating systems which allows one to set up a RADIUS server. This server can be used for authentication and accounting for various types of network access. To use the

server, one needs to correctly setup a client which will talk to the server. These clients include terminal servers, Ethernet switches, wireless access points, or a PC with appropriate software which emulates an access point (such as PortSlave, radiusclient, etc.). FreeRADIUS is being developed by a group of people who call themselves "the FreeRADIUS project" [91]. We have successfully installed freeRADIUS version 2.1.10 at the cloud site in order to provide WPA2-Enterprise certificate based, IEEE 802.1x, wireless authentication for our home and SME network at the local site by using Extensible Authentication Protocol-Transport Layer Security (EAP-TLS). To deploy a public key infrastructure and digital certificates we have used OpenSSL version 1.0.0e. Each of the client certificates was manually installed on a client machine. WRT54GL was configured as an authenticator (i.e., a client) to the freeRADIUS server in order to provide WPA2 Enterprise mode authentication and authorization, for WLAN clients in the home or SME network.

6.13 Data acquisition and processing of data

For data acquisition we were using Wireshark version 1.2.10 (for network interface captures) and minicom version 102-1 (for the serial port captures). We have also used Asterisk's debugging tool (a built-in console) for the SIP and the Jabber channels, in order to conform the SIP captures from the Wireshark and provide protocol message output for the encrypted signalling media stream, such as the media from a Gtalk trunk. The captured data was filtered and exported as either a comma separated value (CSV) or a PCAP. This data was further analysed and plotted using R version 2.11.1[92].

7 Results and Discussion

In this chapter we will present and analyse our results from the experiments that were conducted and present the measurements in relation to the methodology used when conducting the experiments.

7.1 Experiment1

Here we present the results from Experiment 1, that was conducted according to the methodology described in section 5.1. The testbed for this experiment is presented in Chapter 6. Figures 30 and 31 represent the OWD measurements for an emulated media stream. This stream was emulated using the D-ITG tool. The average OWD for the media stream sent from the cloud to the local site (shown in Figure 30) via the VPN link is 42.065 ms. There are 17 hops for this particular route according to the values within the TTL field from the IPv4 packet header. In this figure we can observe high peak intervals (every 120s) for the OWD, with duration of approximately 10s. These peaks are a result from the keep-alive mechanism settings within the OpenVPN server implementation. The VPN server pings the VPN client every 120s, with duration that is equal to getting 10 `ICMP echo reply messages`.

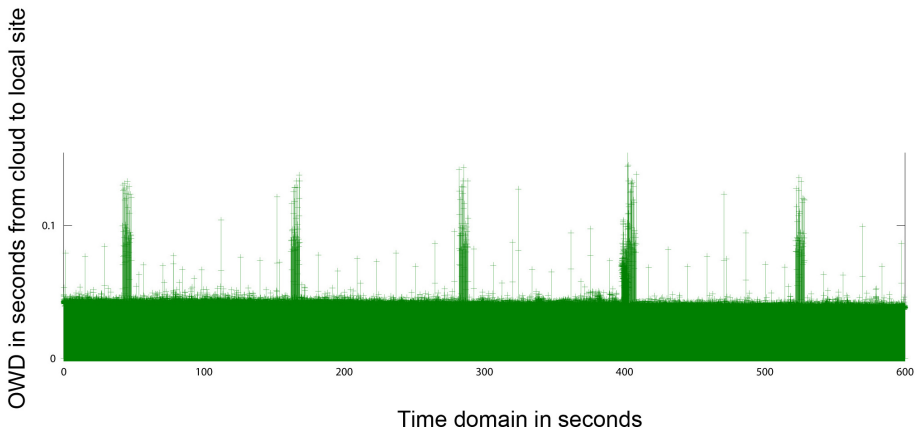


Figure 30: OWD for each UDP packet of an emulated media G.711 u-law media stream, originating from the cloud site to the local site.

The average OWD for the media stream sent from the local to the cloud site (Figure 31) via the VPN link is 59.579 ms. There are 20 hops for this particular route according to the values within the TTL field from the IPv4 packet header. This (the difference in number of hops and latencies) proves that the routing path between the two sites, which spans across multiple autonomous systems on the Internet is in fact asymmetric, most probably because of different BGP routing policies within the ISP networks for the destination routes. In this figure we can also observe high peak intervals (every 120s) for the OWD, with duration of approximately 10s. These peaks are a result from the keep-alive mechanism settings within the OpenVPN server implementation. The VPN server pings the VPN client every 120s, with duration that is equal to getting 10 `ICMP echo reply messages` from the client to the server. These measurements from

the OWD prove that the latency between the two sites is acceptable since they are both bellow the ITU-T G.114 recommendation, which specifies a maximum delay of a 150 ms (one-way) for a media stream.

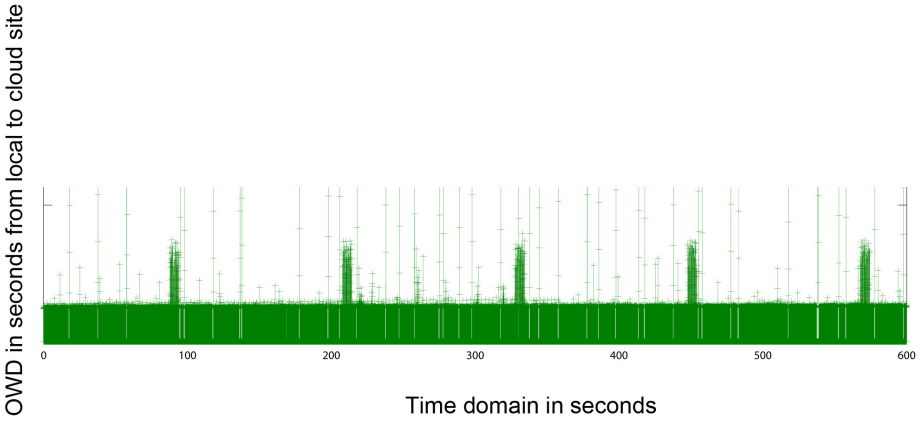


Figure 31: OWD for each UDP packet of an emulated media G.711 u-law media stream, originating from the local site to the cloud site.

Figure 32 is the result of the RTD measurements (measured at the local site). Here it is interesting to note that the cost of encrypting the traffic within the VPN is a 5.4% increase or 3.952ms over the delay for plaintext traffic on the same path.

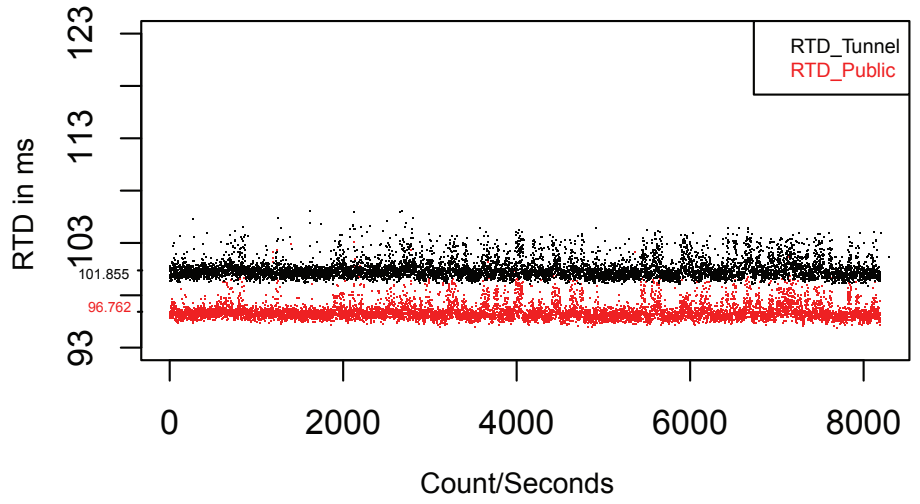


Figure 32: Round trip delay time measurements, between the local and the cloud site. The smaller ticks on the Y axis represent the RTD mean values.

Figures 33, 34, and 35 represent the measurements of UDP jitter, bandwidth, and packet loss on a simulated G.711 VoIP call over the VPN link between the local and cloud site using D-ITG as the traffic generator. Each set of the experiments in these figures is executed simultaneously in both direction (e.g.

from/to local and from/to cloud site) for four times, with each run having a duration of 30 minutes. The independent variable is the bandwidth input value for the D-ITG tool. The bars in each of the bar-plots represent the mean values, with their error bars respectively (representing the standard error). The size of the UDP packet was 172 bytes (160 bytes of payload + 12 bytes for RTP) and a packet was sent every 20ms for each call (approximately simulating G.711 VoIP call) using D-ITG tool. The results from these measurements give us a sense of the expected link performance for the maximum number of simultaneous calls that can be achieved. We can see from the figures that there is a difference in the QoS parameters values for the streams from local to cloud site and *vice versa*. The QoS parameter measurements for the media streams destined from the cloud to the local site outperforms the media streams from the local to the cloud site. This behaviour is a result of the asymmetric routing between the two sites, and the best effort model that is applied within the network elements on the Internet. However, even though the values of the overall achieved-bandwidth, jitter, and packet loss differ between the simultaneous one-way media streams, still most of the values are in the acceptable range. The jitter can be observed in Figure 34. As we can see all off the values are below 1ms (which we consider as a good metric, since anything below 20ms is considered to be good), even the configuration with the biggest variation (hence the error bars maximum values do not exceed 1ms). In Figure 35 the packet loss values can be observed. There is a greater percentage of packet loss of the media streams from the local to the cloud site than the ones from the cloud site to the local site. There is a 265% bigger packet loss of traffic when these media streams flow from the local site as compare with the case when these media streams from the cloud to the local site when the bandwidth is set to 0.35 Mbps. When the bandwidth is set to 0.4 Mbps, there is a 545% bigger packet loss in the uplink than the downlink direction, and when the bandwidth is set to 0.5 Mbps there is a 539% bigger packet loss in the media stream from the local to cloud site than for the media streams from the cloud site to the local site. We argued (in section 5.1) that the acceptable packet loss values should be below 1% for each of the media streams. However, this limit is only met by the media streams from cloud to local site (0.4Mbps and 0.35Mbps). The highest packet loss (17%) was measured on the media stream from local to cloud site when the traffic load was set to 0.5 Mbps.

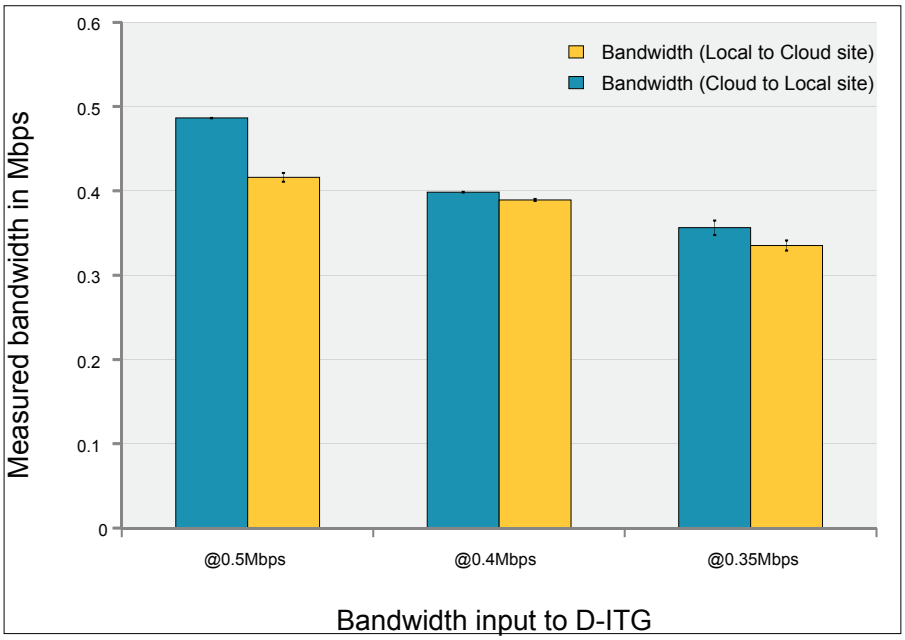


Figure 33: Achieved UDP bandwidth measurement for emulated VoIP media streams running simultaneously from/to local and from/to cloud site, both measurement and traffic generation are done using the D-ITG tool.

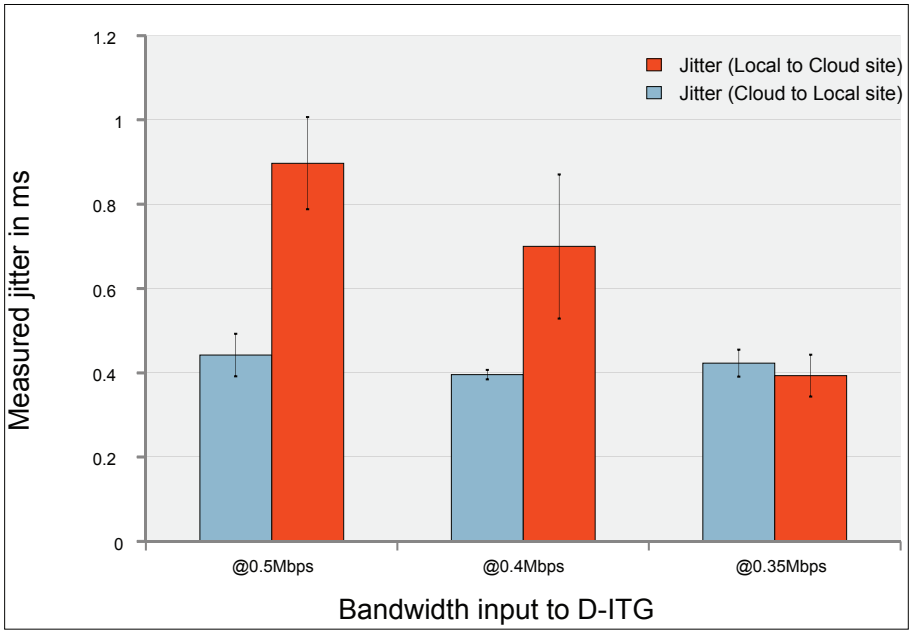


Figure 34: UDP jitter measurement for emulated VoIP media streams running simultaneously from/to local and from/to cloud site, both measurement and traffic generation are done using the D-ITG tool.

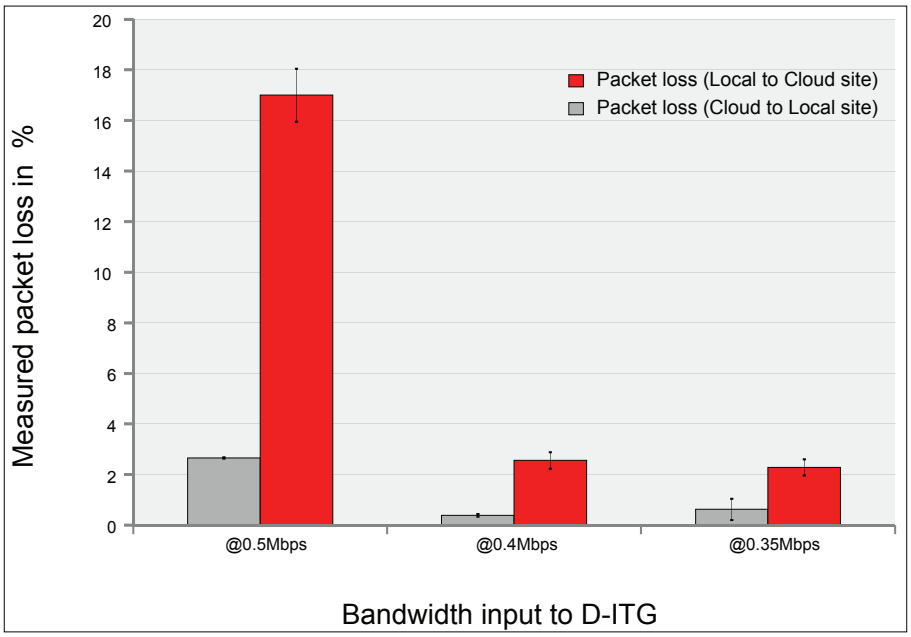


Figure 35: UDP packet loss measurement for emulated VoIP media streams running simultaneously from/to local and from/to cloud site, both measurement and traffic generation are done using the D-ITG tool.

In Figure 36, we can see the results of the measurements of the actual TCP throughput within the VPN link when sending a 6MB file simultaneously from/to both sides, using different TCP window sizes by utilizing the iperf tool. In the direction from cloud to local site we have the best measured performance with low variance when the window size was set to 8KB (0.562Mbps). This is in keeping with the calculated bandwidth delay product from section 5.1. In the direction from local to cloud site the best measured performance, with low variance can be seen also when the TCP windows (advertising and receiving) sizes were set to 32KB (giving 1.57Mbps). Since, there is more TCP related traffic destined from the cloud to the local site, one should tune the TCP windows (advertising and receiving) sizes at the local site and the cloud site to 8KB, for best TCP performance within this VPN link.

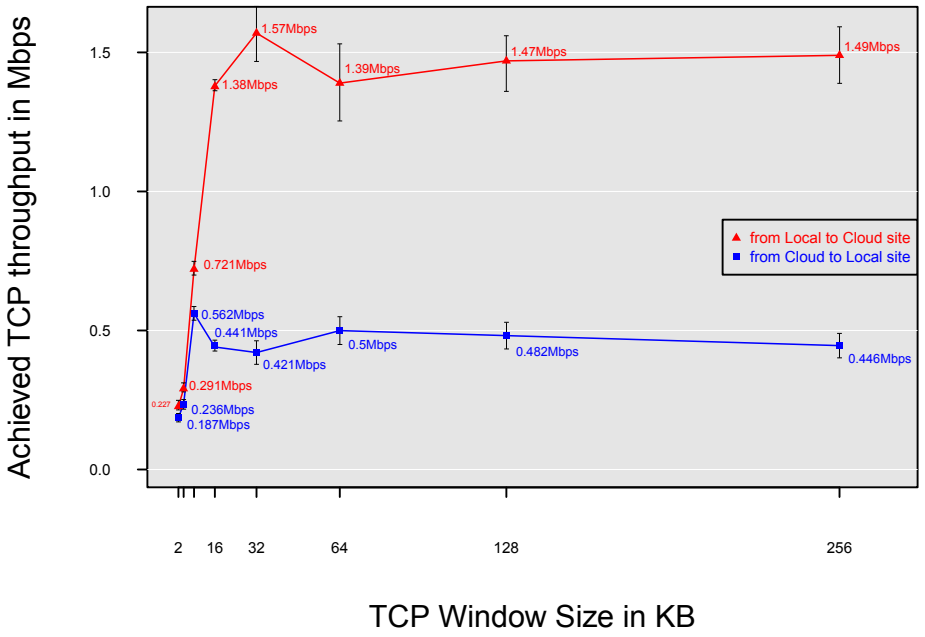
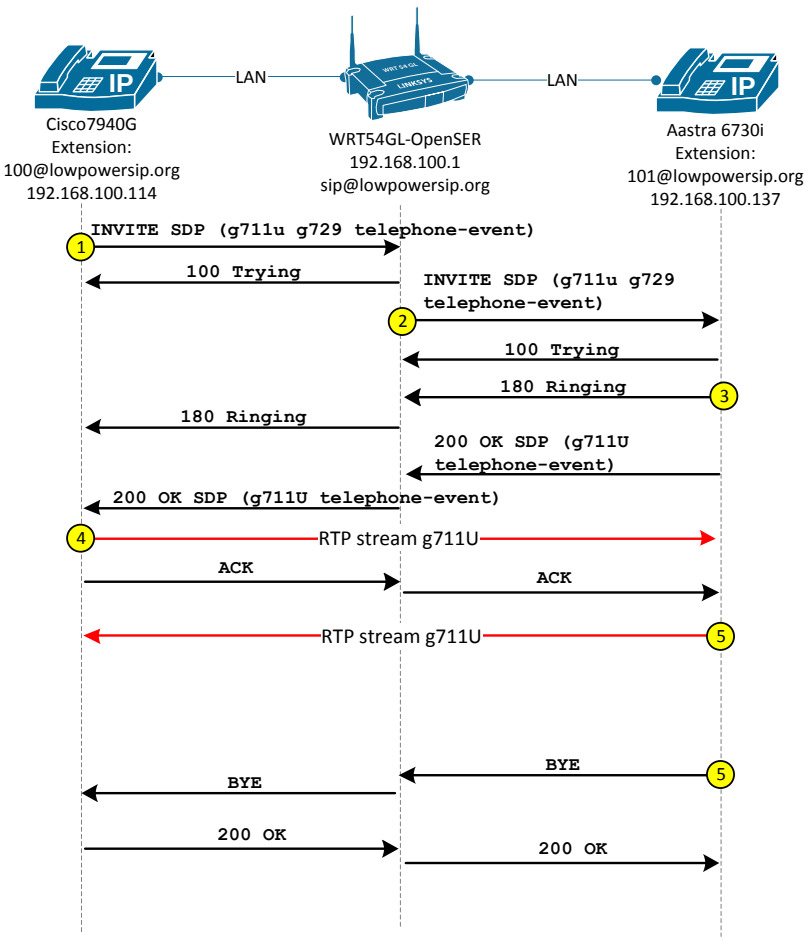


Figure 36: Achieved TCP throughput between the local and the cloud site, for different TCP window sizes. For this experiment the iperf tool was utilized when sending a 6MB MP3 file simultaneously in each direction.

7.2 Experiment2

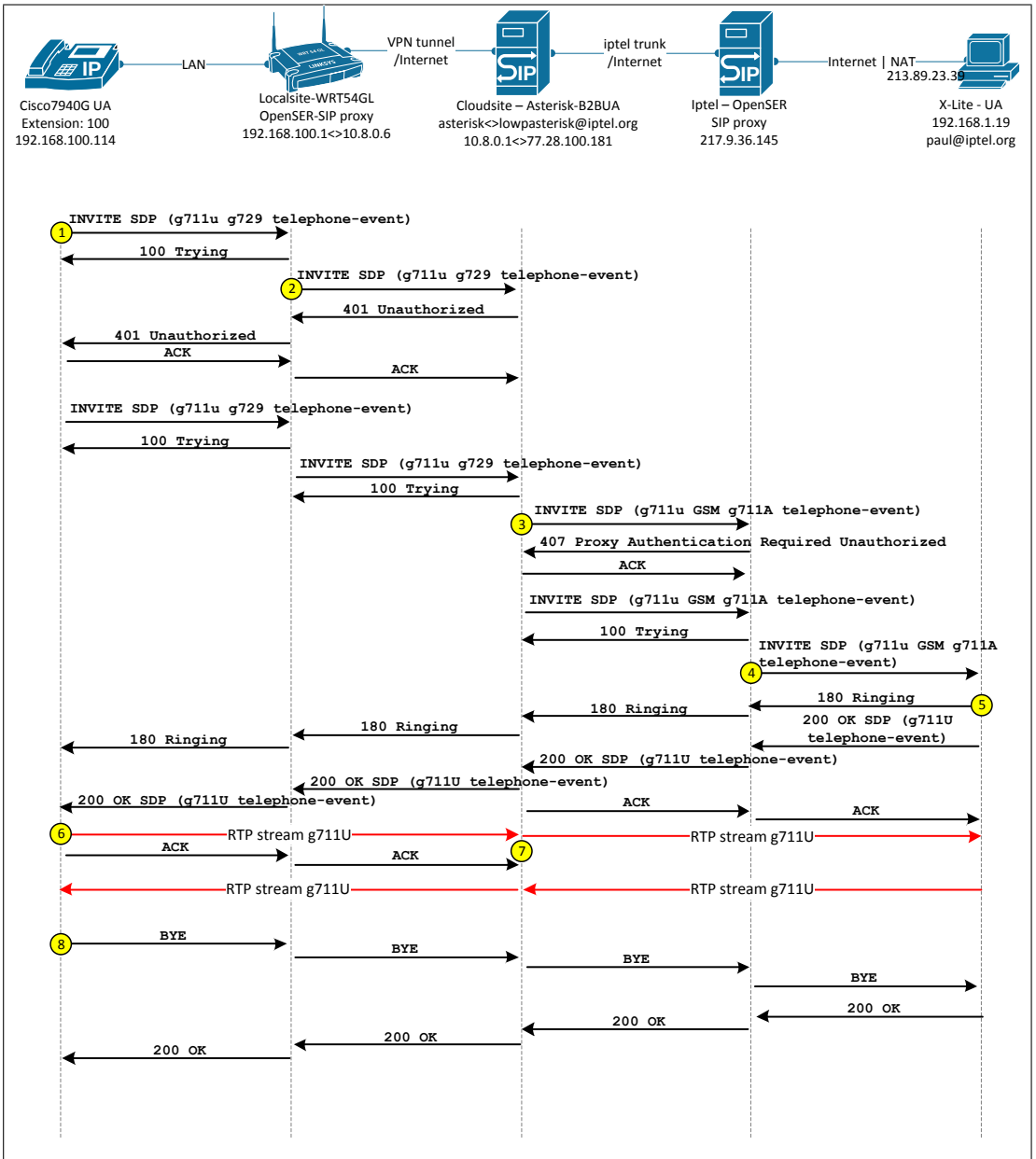
Figures 37, 38, 39, and 40, represent the results from the measurements that concern "Experiment2: IPT system main call scenarios" whose methodology was presented and discussed in section 5.2. For the actual testbed please see Chapter 6. The scenarios were introduced and presented in 4.1. The results are presented as the actual message callflows in the form of a ladder diagram. These messages were captured at various key points: the UA at the local site, the WRT54GL, the cloud site, and the external UA. For each call a .pcap file was created at each of the key points. Later the .pcap files were merged according to the time of day, since all of the machine were synchronized using NTP. From the merged .pcap file for each of the calls the actual message callflow was distilled. The numbered yellow circles on the ladder diagram represent an important event for which an explanation is given in the legend that corresponds to the appropriate number. The behaviour and operation of the IPT system (according to its configuration) is portrayed in these ladder diagrams. The results show that the task was successfully accomplished in each of the designed scenarios. Note that in Figure 39 according to scenario 3 from section 4.1 Alice should INVITE Bob and others to join the conference (somewhere after point 11 on the ladder diagram). This is not shown in the ladder diagram since the principle of operation is already portrayed within this ladder diagram (where Alice transfers the call to the conference server, the only thing that is left is that Bob and others should dial the extension 900 and enter their password for the conference room(after Alice has told them, that they should attend). Hence if there were more rooms on the conference server, Alice should have picked an

attended transfer instead (blind transfer) in order to listen to the room number in the prompt and let Bob and the others know which room number they should join). We have limited the size of the ladder diagram, in order to fit each on one page. All of the scenarios were executed three times, however no variance in the message call flow or a given task's success was noticed.



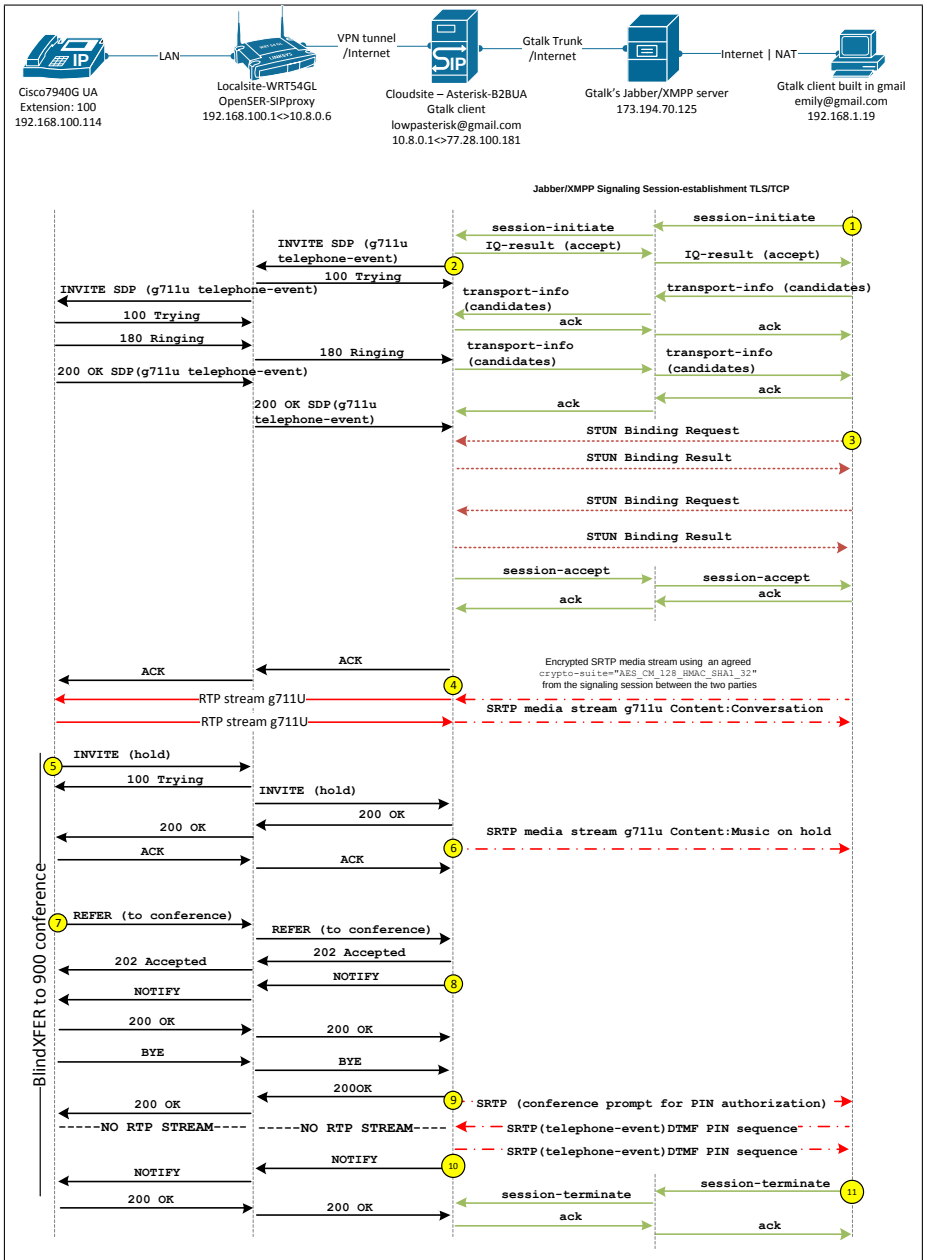
1. The INVITE is first send to the outbound proxy server (WRT54GL)
2. The outbound proxy server checks if the extension 101@lowpowersip.org is registered in its registry, since it is, the call is routed locally, and the INVITE is proxied to 192.168.100.137
3. The SIP hardphone rings (audible and visual signal)
4. After the 200 OK message with SDP is received from the outbound SIP proxy, an RTP stream is sent directly to the UA and an ACK for the 200 OK message is send via the outbound SIP proxy.
5. When the ACK is received, for the 200 OK with SDP, an RTP stream is sent directley to the UA.
6. Hang-up

Figure 37: Actual message callflow that relates to Scenario 1 from section 4.1.



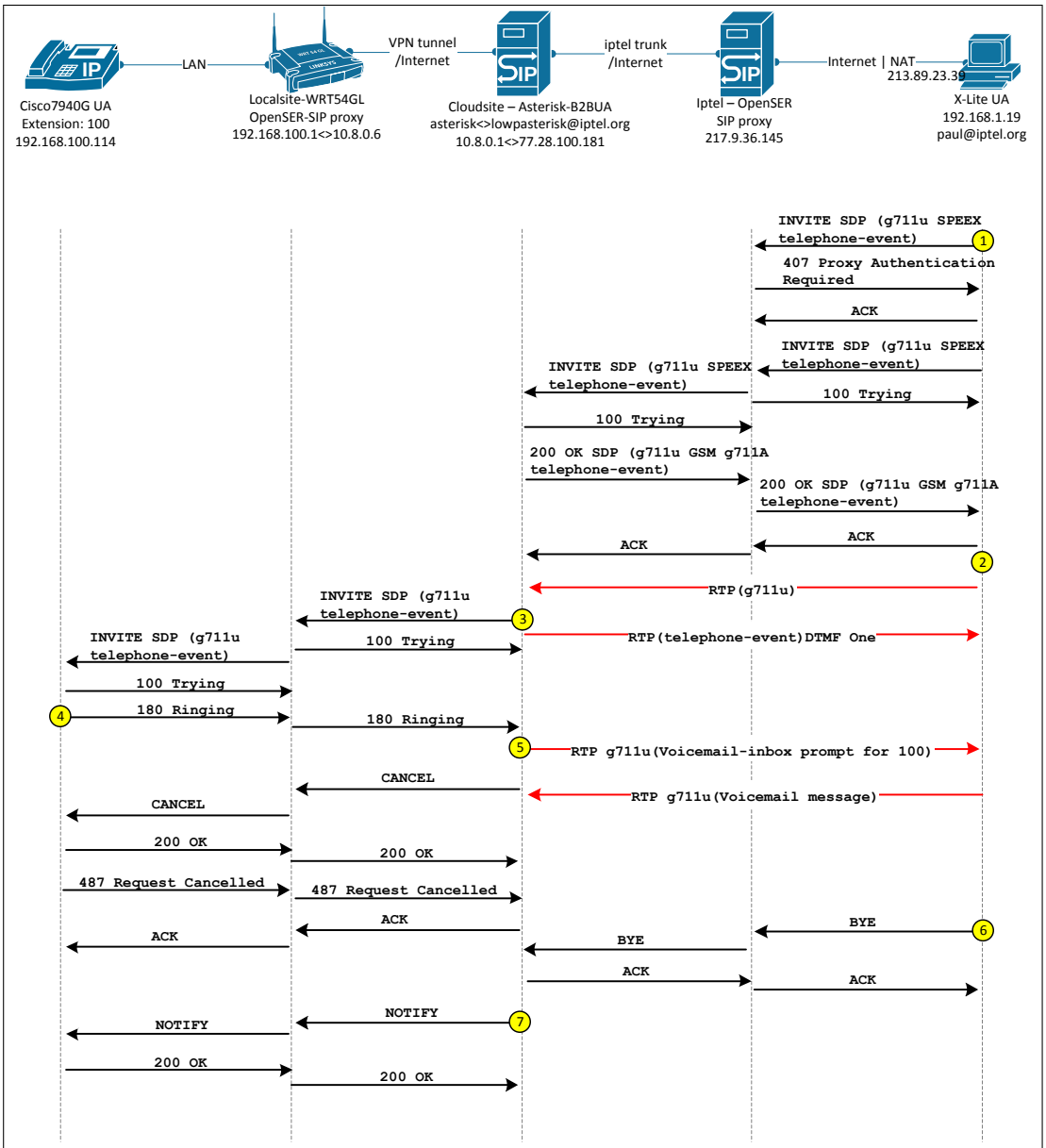
1. The INVITE is first send to the outbound proxy server (WRT54GL)
2. The outbound proxy server checks if the extension paul@iptel.org is registered in its registry, since it is not, the call is routed the configured SIP proxy server of the SIP hardphone, therefore the INVITE is proxied to 10.8.0.1
3. Here the B2BUA server is already registered to the iptel's registrar server (as a lowpasterisk@iptel.org client, not shown in this dialogue), the dial logic in the Asterisk server forces the `INVITE`s that contain `iptel.org` (in the `Host part:` field from the `To` header of the `INVITE` message) to be sent via the lowpasterisk@iptel.org UA. Therefore an `INVITE` is send from lowpasterisk@iptel.org (UA) to paul@iptel.org user via the iptel's SIP proxy, containing SDP parameters according to the B2BUA server. In this dialogue the UA authorizes with the iptel's SIP proxy (as a lowpasterisk@iptel.org client) and than sends the `INVITE` (but keeps the original `SIP Display info:` in the `From` header from extension 100)
4. After processing the INVITE and contacting the AOR (for paul@iptel.org) the iptel's SIP proxy server sends the INVITE to paul's location.
5. The SIP softphone rings (audible and visual signal)
6. After receiving the `200 OK` with `SDP` (sent from asterisk@10.8.0.1 UA) an RTP stream is sent to the asterisk UA.
7. Here the B2BUA server mixes/bridges the RTP streams between both of his internal UAs (asterisk and lowpasterisk@iptel.org).
8. Hang-up

Figure 38: Actual message callflow that relates to Scenario 2 from section 4.1



1. The user at the Gtalk client (emily@gmail.com), that is built in Gmail, initiates a session towards lowpasterisk@gmail.com
2. Here the Asterisk server functions as a gateway where on one side (to the right) he is connected as a Gtalk client (lowpasterisk@gmail.com) to the Gtalk network by utilizing the Jabber/XMPP protocol and on the other side (to the left) he is acting as a SIP UA. Therefore after receiving the session-initiate message (on the Jabber channel), the Asterisk server consults the dialplan logic and the AOR, as a result it sends an INVITE message towards the registered SIP extension: 100 by utilizing the SIP channel.
3. The establishment of and maintenance of RTP and RTCP channels is done by performing connectivity checks against the Asterisk server using STUN binding requests and responses. For a given candidate (lowpasterisk@gmail.com) to be considered usable, it must respond to a STUN binding request with an appropriate STUN binding response. In addition to the initial connectivity checks, STUN is used after the channel is established to ensure it is still alive.
4. After the Jabber/XMPP signaling session has successfully negotiated an SRTP media session (between the both gtlak clients), an SRTP stream from emily@gmail.com is sent to lowpasterisk@gmail.com and *vice verca*, the SRTP stream that is received by the Asterisk server is being processed (decrypted) and sent as RTP stream to the SIP UA at extension 100, hence the Asterisk server acts as a UA on the left site (SIP channel side). Here the content of the RTP streams is the actual conversation between Emily and the SIP UA at extension 100.
5. During a call with emily@gmail.com, the user at extension 100 presses the BlndXfr soft key (for unattended transfer), in order to transfer Emily to the conference server.
6. This places the call on hold (where music on hold is played to emily@gmail.com).
7. The user at extension 100 dials the number 900 where the conference server resides in order to transfer the call and hangs-up. This results in a SIP REFER method being sent to the Asterisk server UA for the conference server at extension 900. This method is accepted with `202 Accepted` message.
8. Notification that the transfer is active.
9. The call is being transferred to the conference server where Emily is prompted to enter a password for the conference room that was given in the conversation with the user at extension 100, Emily enters the password via the gtalk (text input) interface and it is placed successfully as the first participant in the conference, therefore music on hold is being played.
10. The Asterisk server notifies the UA at extension 100 that the media session has been successfully established.
11. Emily hangs-up the call by pressing the end key on the gtalk GUI interface.

Figure 39: Actual message callflow that relates to Scenario 3 from section 4.1



1. paul@iptel.org initiates a call to lowpasterisk@iptel.org, resulting in sending a SIP INVITE message with SDP
2. Media session is established between lowpasteris@iptel.org and paul@iptel.org
3. The Asterisk server is consulting the dial plan logic and AOR in order to contact the UA at extension 100, resulting in an INVITE message being sent to it. At the same time from a beep sound is played for one second (DTMF One) in order to signalize activity for paul@iptel.org while the INVITE is beeing processed further on by the WRT54GL.
4. The SIP hardphone is ringing (audible and visual signalization)
5. As a result of the dial plan logic (after six dial intervals), if there is no answer (e.g. `200 OK SDP`) the call is transferred to the voicemail inbox of extension 100, this result in a `CANCEL` message being sent to the UA at extension 100 (which stops the ringing and results in sending a `487 Request Cancelled` message) , a voicemail prompt is being streamed to paul@iptel.org. The user leaves a message which is recorded.
6. Hang-up
7. A NOTIFY message is sent to inform the UA at extension 100 (for simple-message-summary) which results in a message waiting indication (visual signalization). Als,o considering the configuration parameters in `voicemail.conf` file from the Asterisk server configuration, there is an email information for extension 100 (e.g. extension100@gmail.com). The asterisk server utilizes the sendmail SMTP server in order to send the newly recorded audio message (from the voicemail-inbox of extension 100) as an attachment to extension100@gmail.com.

Figure 40: Actual message callflow that relates to Scenario 4 from 4.1

7.3 Experiment3

In this experiment we have measured the main low-power scenario described in section 2.10, according to the methodology described in section 5.3 and the testbed in Chapter 6. The graph in Figure 41 represents the results from these measurements.

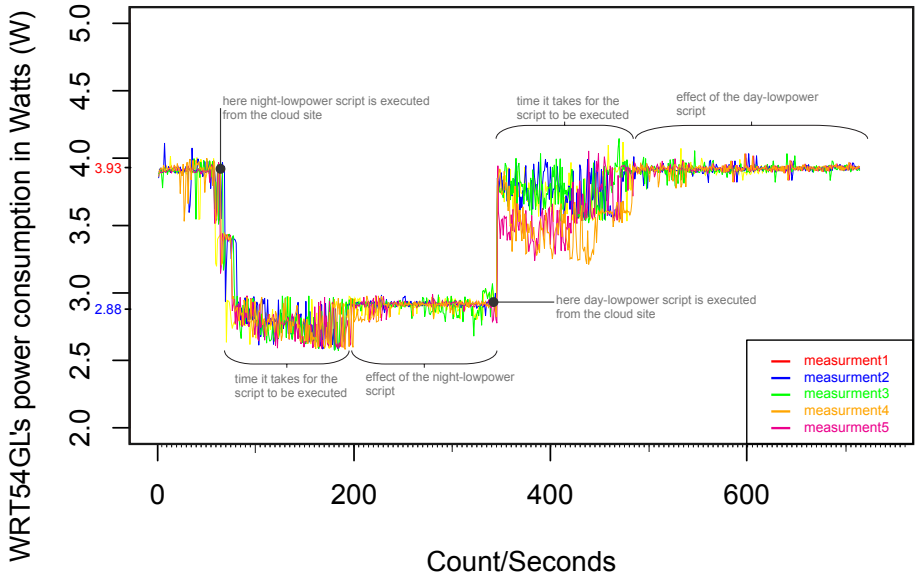


Figure 41: Results from the lowpower (day-night) scripts five consecutive executions, and their effect on the WRT54GL's power consumption. The smaller ticks on the Y axis represent the mean values from the effect of the night/day-lowpower script.

These measurements were made on an sdmod version of the WRT54GL with a CPU clocked at 200MHz. There were five runs of the scenario each with a duration of 700 seconds. For this experiment we created two fake extensions: 700 (that causes the execution of the night-lowpower script) and 800 (that causes the execution of the day-lowpower script). A SIP hardphone (Cisco 7940G) from the local site executes these scripts by dialling these extensions. The black points on the graph represent the time of the script's execution. There were three active switch ports on the WRT54GL: one for the Internet connection, one for the SIP hardphone, and one for a client PC. The WiFi was also active (with no clients associated) before the script execution. From these graphs we can see that a certain amount of time is needed for the script to be executed (due to the configuration of the script). The execution time of the night-lowpower script (up to the point where WRT54GL is configured according to the script) is 130 seconds. The execution time of the day-lowpower script (up to the point where WRT54GL is configured according to the script) is 140 seconds. From these results we can see that the power consumption of the WRT54GL decreases for 1.05W (mainly as a result of turning off the WiFi radio interface and clocking the CPU to 183MHz) The results of the measured tasks success from the low-power scenario from section 2.10 are successful for all of the performed tasks.

We can also see that there are no significant variations within the values of the performed measurements. This graph proves the potential power savings of the WRT54GL when serving as the local site's telephony gateway and SIP server in our IP telephony system. This savings of 1.05W occurs for each hour that the business is not operating (typically 15 or more hours per working day and 24 hours for each holiday) over the course of a year. This leads to a conclusion that for year 2013 (in the case of Sweden) there will be 6555[14] hours for power saving potential or 6.88KWh.

7.4 Experiment4

This experiment is related to the power consumption measurements for both the sdmod and nomod versions of WRT54GL in various configurations. The configurations are in accordance to the methodology that was explained for this experiment in section 5.4. The testbed for these measurements was presented in section 6.5. The loaded DD-WRT firmware in both versions (sdmod and nomod) is in accordance with the local site's design and definition as presented in section 4.3.2. The graphs in Figures: 42, 43, and 44 represent the measurements of the over-all power consumption of WRT54GL (nomod and sdmod) when the CPU is clocked at different frequencies(183, 200, and 250MHz). The mean values from the power consumption (after the boot process) measurements are indicated with smaller ticks on the Y axis of the graphs.

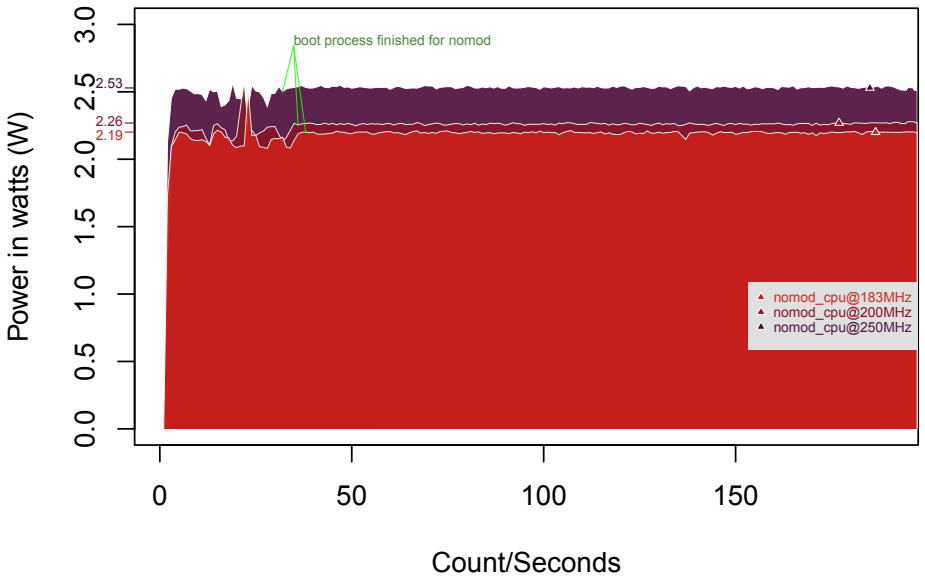


Figure 42: CPU power consumption for different clock sets on WRT54GL without hardware modification (nomod)

In table 10 we can see that the power consumption increase between the nomod and sdmod version of the WRT54GL is approximately 55 milliwatts.

[14](120 days of holidays and weekends * 24 hours) + (245 of working days * 15 hours) = 6555 hours

This represent the actual cost of the hardware modification in terms of power consumption. This difference is visually represented in the graph of Figure 44. From the graphs we can also see the actual cost (in terms of power consumption) when WRT54GL's CPU is clocked with the three different frequencies. The increase in power consumption from 183MHz to 200MHz is only 3% in comparison to the increase from 183MHz to 250MHz which is 15.5%.

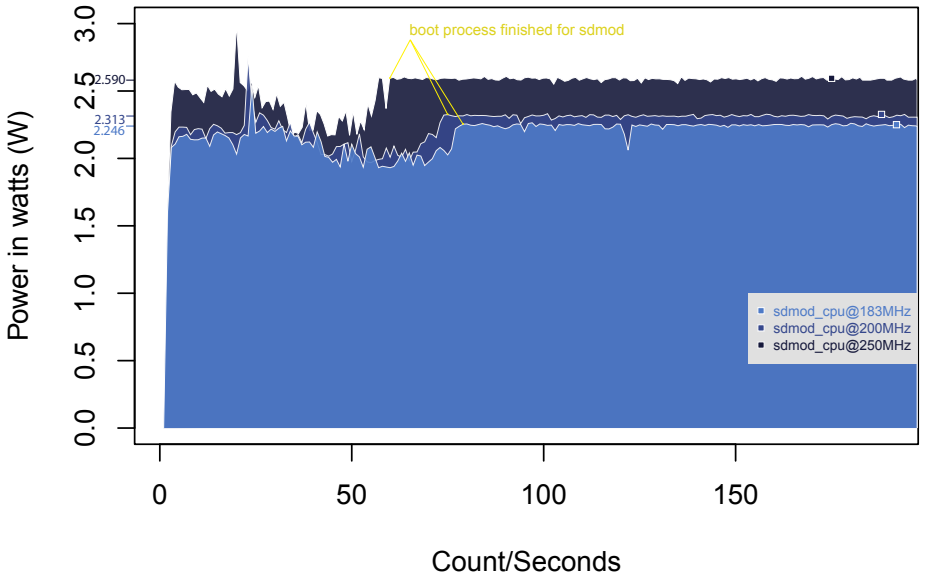


Figure 43: CPU power consumption for different clock sets on WRT54GL with SD hardware modification (sdmod)

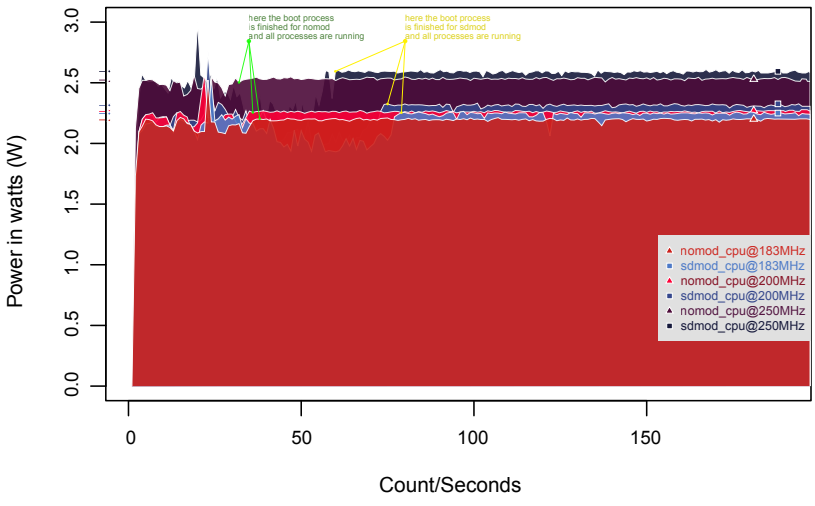


Figure 44: The difference in CPU power consumption for different clock rates on WRT54GL with SD hardware modification (sdmod) and without hardware modification (nomod).

Table 10: Power consumption values for the different CPU clock settings on sdmod and nomod version of WRT54GL

CPU clock (MHz)	nomod (W)	sdmod (W)	increase (nomod to sdmod) (%)
183	2.190	2.246	2.56
200	2.260	2.313	2.34
250	2.533	2.590	2.25

The graphs from Figures 45 and 46 represent mean comparisons of the measured power consumption values for the sdmod and nomod version of WRT54GL in regards to the Ethernet switch ports (all 5 active) only, WiFi radio interface only, Ethernet switch ports and WiFi radio running together, while the CPU was clocked at 200MHz. These configurations were presented in section 5.4. Table 11 presents the mean values (after the boot process has completed) from the measurements for both sdmod and nomod. Since the two versions do not differ significantly. In the third column from the table we summarize the costs (in percentage and Watts) from the different configurations in comparison to the nomod values from the experiment. The variance in the measured power consumption values between the different runs is insignificant.

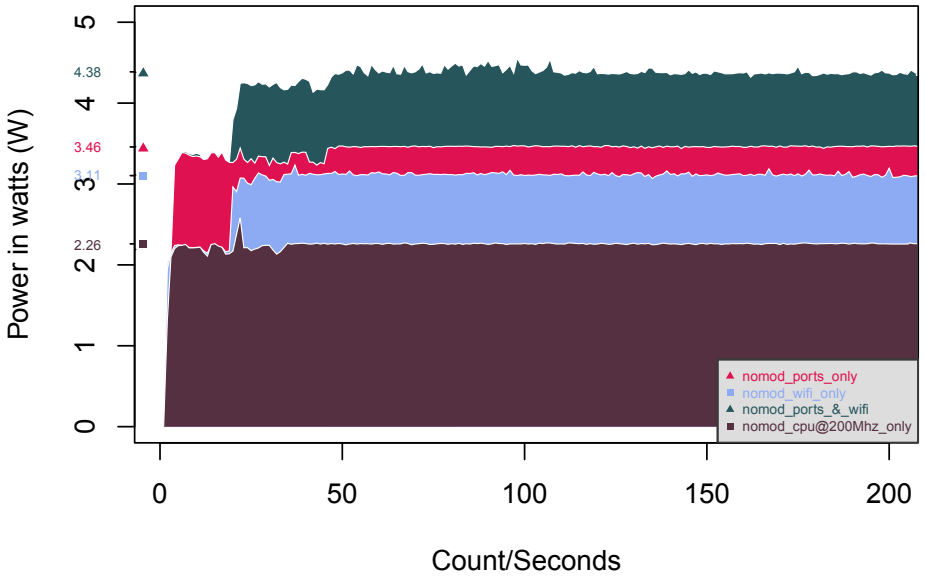


Figure 45: Measurements of the overall power consumption of the nomod version for WRT54GL in various configurations. Each of the graphs is plotted using the mean values from all the measurements in the corresponding subscenario testing.

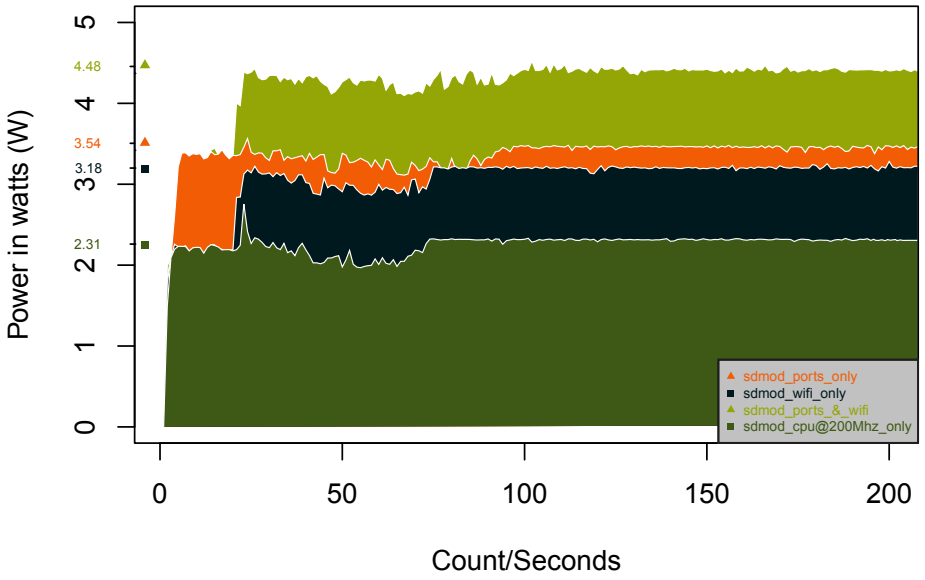


Figure 46: This graph plot is related to the measurements for the overall power consumption of the sdmod version for WRT54GL in various configurations. Each of the graphs is plotted using the mean values from all the measurements in the corresponding subscenario testing.

Table 11: Power consumption values for the different settings on sdmod and nomod version of WRT54GL when running at 200MHz clock speed. The values in the fourth and fifth column represent (nomod values) the cost or increase (in percentage and watts) and use the nomod CPU@200MHz value (2.26W) as a reference.

WRT54GL configuration	nomod (W)	sdmod (W)	Cost (%)	Cost (W)
CPU@200MHz only	2.26	2.31	-	-
WiFi only	3.11	3.18	38	0.85
Switch ports only	3.46	3.54	53	1.2
Switch ports and WiFI	4.38	4.48	93	2.12
Switch's single port	-	-	-	0.25

7.5 Experiment5

The idea of this experiment is to stress the actual IP telephony system solution over the tunnel between the local and the cloud site with real VoIP calls from SIP UACs. This experiment is in accordance with the methodology that was presented in section 5.5. The testbed for this experiment was presented in Chapter 6. The bar-plot in Figure 47 presents the mean values of the QoS parameters measurements.

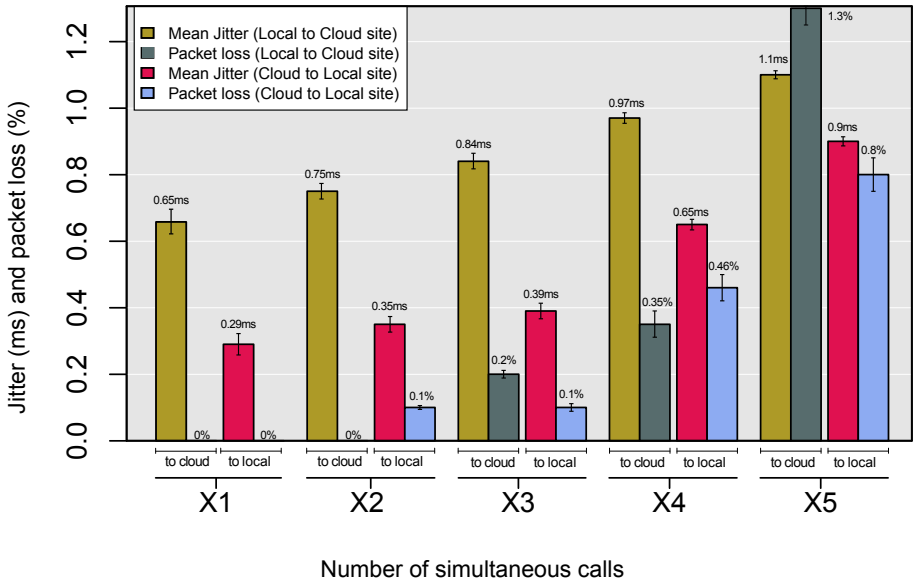


Figure 47: Mean QoS parameters values, for the different media call flows (between the local site and the cloud site), in the different number of simultaneous calls (note that two types of data are being shown one in units of ms and the other in percentage).

This experiment evaluates the capacity of the IPT system based upon the local-to-cloud site VPN link with regard to the number of simultaneous calls. The error bars in the bar-plot represent the variance (standard error) of the measured data. There are five different offer traffic loads in numbers of simultaneous calls and each given number of simultaneous calls was run ten times. For example, the third step (X3) is a mean value of ten iterations; where for each iteration, three simultaneous calls are initiated from the local to the cloud site. Each call or iteration lasts for 10 minutes. Once SIP establishes the media session there are two separate media streams for each call (from local to cloud site and vice versa). Here the important metrics that reflect the call quality are jitter, packet loss, packetization time of the CODEC, bandwidth used, and the power consumption were measured during each iteration. For media streams we used only the G711.U CODEC. It is important to note that the mean interpacket interval was 20ms for all the calls (as expected for the G711.U CODEC), the mean bandwidth for each direction of the call was 80kbps (hence for both direction that is 160kbps), finally there was no increase in power or CPU consumption in any of the iterations. From these results, we can see that the jitter values (local to cloud site) have a linear increase interval of approximately 14% as the number of simultaneous calls increases. However, this is not the case for the jitter values (cloud to local site), where the increase is still linear but with different interval values. For both of the flows it is evident that the jitter linearly grows with the number of simultaneous calls, From the results we can also observe that the jitter values for the flow local-to-cloud are higher (for more than 100% in X1, X2 and X3) than for the flow cloud-to-local, which is also in relation to the Experiment 1 results for the jitter, that were measured with the D-ITG tool. Since the packet loss and the jitter values are below 1 in almost all the cases (except X5: from

local to cloud) we consider that there is still room and support in the VPN link for six simultaneous calls(with acceptable QoS parameters). The packet loss values do not correspond with the Experiment 1 values which is a result of the applied QoS parameters configured on the WRT54GL on Layer2 and Layer3 for the VoIP traffic during this experiment (this is discussed in section 5.5).

7.6 Experiment6

The results from this experiment represent the bootstrap mechanism (that is assisted by DHCP and TFTP server) and locating of SIP servers (for redundancy) in the IP telephony system. The experiment was conducted according to the methodology presented in section 5.6. The testbed for this experiment is presented in Figure 48.

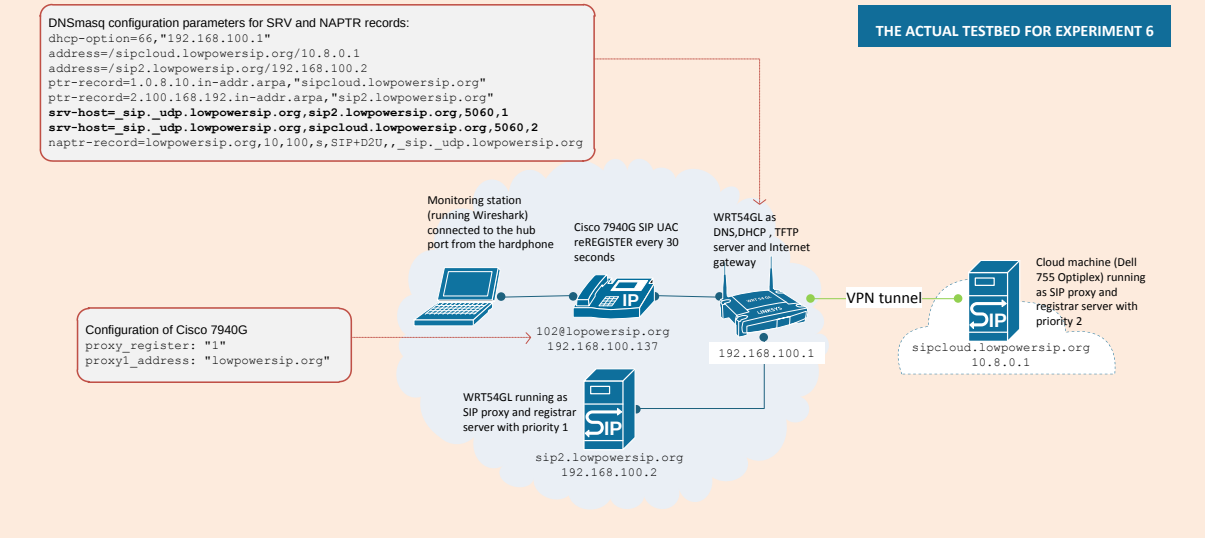


Figure 48: Experiment 6, actual testbed.

The results (in form of filtered wireshark output) from the experiment are presented in Figure 49. The text boxes to the right side of the wireshark output are discussions for the related steps. We strongly encourage the reader to go through these discussions carefully in order to understand the executed procedure and related outcome. The left side of the wireshark output presents the duration of time for the various steps in the experiment.

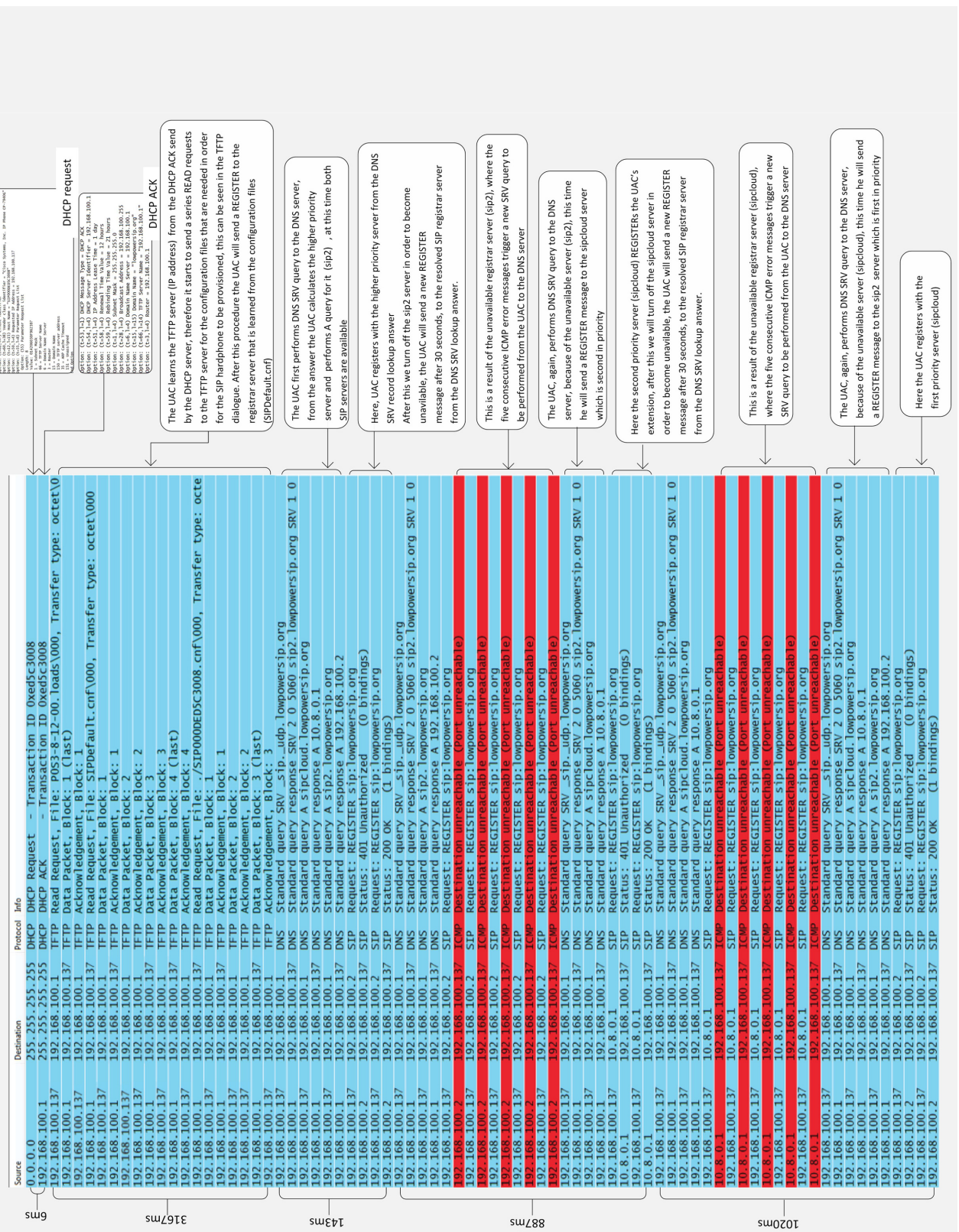


Figure 49: Experiment 6, results.

The first part of the wireshark output presnets the bootstraping process and the second part presents the process of locating the SIP servers within the IP telephony system, all from the perspective of the UA. From these results we conclude that the bootstrapping task is successfully accomplished within 3.2 seconds. The registering process (at the first yellow circle) of the UA (when using SRV lookups) was successful and had a duration of 143ms. It is interesting to note that the UA, did not perform an NAPTR lookup. The results show the redundancy with the fail-over mechanism is achieved for the SIP servers by utilizing the DNS SRV lookups from the UA. The first failover lasted for 887ms (at the second yellow circle), the second failover lasted 1020ms, which is longer than the first one (because here the timeout dialogues are between the local and the cloud site, which adds to the duration).

7.7 Experiment7

Here, we present the results of performance testing of the proxy and registration server that is embedded in the WRT54GL. The methodology for conducting this experiment was presented in section 5.7. The testbed was presented in sections 6.2 and 6.3. We used the nomod version of WRT54GL for this experiment.

The graph in Figure 50 represents the mean values (and their variance, calculated as standard error) for the RRD measurements (for 1, 5, 10, 25, 35, and 50 simultaneously executed and successfully established registrations) from a UAC (SIPp) perspective (i.e., the UAC registers with the WRT54GL SIP registrar server) against the mean values of the CPU load of the WRT54GL.

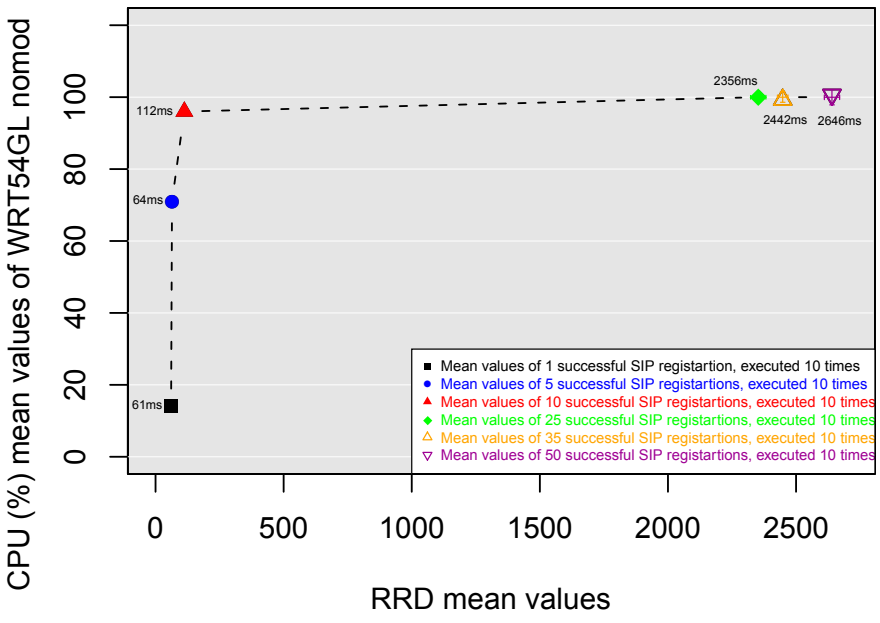


Figure 50: Mean values for the RRD measurements (SIP using UDP as transport mechanism with authentication) from a UAC (SIPp) perspective (that registers to WRT54GL SIP registrar server) against the mean values of the CPU load of the WRT54GL

From the graph we can see that as the number of simultaneous registrations increases the CPU load also increases. Based on these results, we can assume that the RRD values up to the values for 10 simultaneously executed registrations would be acceptable. In a situation when more than 10 simultaneously executed registrations can occur, a more reasonable approach would be to design the IPT system in such a way that the SIP registration service will be only provisioned from the cloud site. Interesting to note is that anything above 10 simultaneously executed and successfully established registrations causes a CPU load of 100%, which in return causes registration request timeouts that are also reflected in the RRD values.

The graph in Figure 51 represents the mean values (and their variance, calculated as standard error) for the SRD measurements (for 1, 5, 10, 25, 35 and 50 simultaneously executed and successfully established sessions) from a UAC (SIPp) perspective (that uses WRT54GL as a SIP proxy server) against the mean values of the CPU load of the WRT54GL. From the graph we can see that as the number of simultaneous session increases the CPU values also increase.

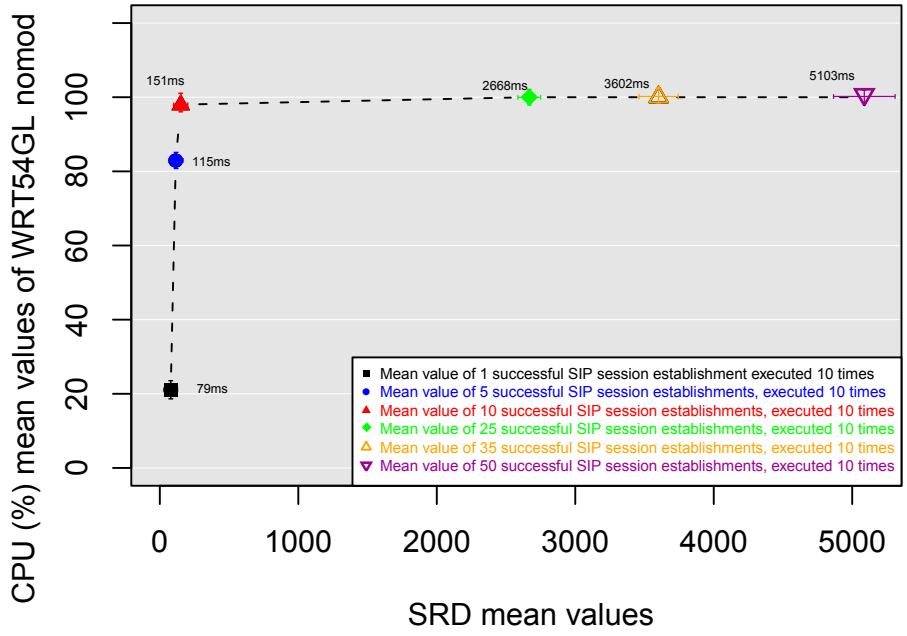


Figure 51: Mean values for the SRD measurements (SIP using UDP as transport mechanism with authentication) from a UAC (SIPp) perspective (this UAC sends INVITEs to SIPp UAS via the WRT54GL SIP proxy server) against the mean values of the CPU load of the WRT54GL.

Based on these results, we can assume that the SRD values up to the values for 10 simultaneously executed sessions would be acceptable. In a situation when more than 10 simultaneous new sessions are initiated, a more reasonable approach would be to design the IPT system in such a way that the SIP proxy will be based only in the cloud site. Interesting to note is that anything above

10 simultaneously executed sessions causes a CPU load of 100%, which in return causes session requests timeouts that are also reflected in the SRD values.

8 Conclusion

This master thesis project successfully designed, implemented, and tested a VoIP system with part of the infrastructure running on a SIP server on a low power commodity processor assisted by a cloud service in order to provide a full featured IP telephony solution.

The main goal of this thesis was to create a design for an energy efficient IP telephony system. Therefore in Chapter 4 we have defined the design of the IP telephony system by identifying the features that should be implemented in the local site and the cloud site. Along with the features we have identified we also defined the main call scenarios that this system should support and the main low-power (energy-efficient) based upon what the hardware could support.

A secondary goal of this thesis was to implement the design developed in first goal. This implementation was successful and tested extensively for a period of five months. During the implementation phase a main focus was the optimization (both hardware and software) of WRT54GL which was located at the local site and provided the local site's part of the IP telephony system. For that reason we have successfully created and tested two versions of the WRT54GL: with SD hardware modification and without. We have tested both versions of WRT54GL in terms of their difference in power consumption. To be able to utilize the nomod version of WRT54GL we reduced the amount of software to such a point that it fit into 4MB(flash memory), thus eliminating the need for any hardware modifications. However, our tests showed no significant difference in power consumption between the two modes of operation and configuration (both hardware and software). Hence the cost of running an sdmod version of WRT54GL is 55 milliwatts.

A third goal of this thesis was testing the performance of the IP telephony system with all of its features, through a series of experiments. These experiments were conducted according to a predefined methodology, presented in Chapter 5. The testbed for these experiments was described in Chapter 6. The results from these experiments are presented and discussed in Chapter 7.

In this thesis, we have demonstrated that energy efficiency can be achieved for a functional IP telephony system (in the context of homes and SME) with part of the infrastructure running on a SIP server on a low power commodity processor (such as the one that is found in WRT54GL) assisted by a cloud service (such as IaaS with an SLA) in order to provide a full featured IP telephony solution.

DD-WRT provided flexibility for the customization of the firmware releases using both cross compiling from source code or using the firmware modification kit. This platform offers a lot of possibilities to investigate a commodity embedded hardware platform.

Through the series of the experiments we have presented the overall power consumption of the WRT54GL in various configurations with a peak power consumption of 4.5W, and finally the effect of the low-power script, which decreases the overall power consumption of the unit by 1.05W. The savings

potential for the local site over the course of a year for an SME is 6.88KWh.

However, we have also seen that WRT54GL lacks the full control of its built in manageable switch (mainly because of its internal devices network architecture), specificity the possibility to administratively and selectively disable (power-off) individual physical LAN interfaces. We would have, additionally lowered the power consumption in night-mode by turning off LAN ports that are not related to IP telephony e.g., LAN printers or PCs. One active LAN port (with a 100BASE-T Ethernet established link through an FTP CAT5e cable of 10 meters) consumes 0.25W. Another disadvantage of WRT54GL in regards to being a part of an energy efficient IP telephony system is the lack of hardware support to wake up on magic packets, e.g to be a WoL client, thus additional hardware is needed. The experiments show that the WRT54GL's (nomod) booting time is 40 seconds which makes the WoL technology unusable in terms of a calling user experience, that is the waiting time for this device to boot and than a call to be established. However the WoL technology could be utilized when the low-power scripts from the cloud site will instruct the local site to fully hibernate, thus saving even more power.

We envision, that the methodology of our experiments can serve as a framework for conformance testing of any IP telephony system which includes performance testing of the signalling and the media layer. This yields some key performance indicators of the system and the services it should provide.

We have also seen that with our design, placing the SIP proxy and registrar at the local site for redundancy comes with a cost and results in a bottleneck. The resulting IP telephony system with a single WRT54GL can be used for a maximum number of 25 users. The WRT54GL can support maximum of 10 simultaneous SIP registrations and 10 simultaneous sessions. The system supports five concurrent calls on the link between the local and the cloud site (given the available bandwidth). Another approach to increase capacity would be to use another WRT54GL at the local site with load balancing logic by utilizing the DNS SRV record lookups, or move the registrar and proxy service from the local site to be only provisioned from the cloud site. However, the best testing scenario would be a real SME where this system can be utilized and extensively tested.

8.1 Reflections

During the course of this project we considered the economic and environmental aspects of this research by designing and implementing an energy efficient IP telephony system in the context of SMEs and homes using low cost commodity hardware such as the Linksys WRT54GL and emulated cloud IaaS. The results from this thesis show a potential for significant power savings when using the proposed design for an IP telephony system.

We have not encountered any significant ethical issues when carrying out this thesis project. The data that is being sniffed is deliberately produced by us, for the purpose of testing the IPT system.

9 Future work

In this chapter we suggest direction for future work and focus on things that were left undone.

One should take into consideration the possibility of modifying the WRT54GL hardware in order to support the WoL feature. This feature is not natively supported by the WRT54GL. This would allow the WRT54GL to sleep most of the time, and wake up when instructed by the cloud with a magic packet, e.g. based upon the starting time of the office hours. However this concept needs a lot of testing and could be a thesis project of its own. An example of a dedicated WoL hardware module can be found at http://rototron.info/WOL/WOL.php.

In our deployed IPT system the NMS checks basic services, such as CPU load, network interfaces status, PING, SSH, RAM usage, uptime, HTTP, DNS (A, PTR, SRV, and NAPTR lookups), number of WiFi associated clients, total processes, and number of current users. However, further monitoring is needed for a reliable and full featured NMS:

- bandwidth utilization of the interfaces with MRTG (incorporated into Nagios)
- SIP channel utilization status of both the local and the cloud site (could be useful for identifying peaks of traffic and therefore calibrate the telephony system)
- custom check for the VPN status (in terms of which client is connected to the VPN server, useful when there is more than one local site)
- weather map (which will present the QoS parameters for every VoIP call and its related message callflow)
- firewall availability

A further improvement of the IP telephony system's security, where SIPS and SRTP should be enabled and tested.

In order to enable conferencing within the local site (and keep the media streams local) for more than 3 participants (that is supported natively as a 3-way conference call service on most of the SIP hardphones), a possible solution would be to use a conference server in the local site running on a virtual machine. The virtual machine could run as a background process on one of the workstations in the SME (e.g. the machine of the secretary that runs a Windows OS). On every boot the virtual box process will be started in the background without any notice by the secretary. The conference server could be registered to the WRT54GL as a regular extension e.g. 910. An example of such a conference server is i2conf and its source code can be found at http://sourceforge.net/projects/i2conf/. An example of a tool that enables the VM to be run as a background process on Windows OS is VBoxVmService and can be found at http://vboxvmservice.sourceforge.net/. This conference server could be monitored by the NMS for availability.

As already discussed in section 2.10.4, an alternative hardware platform for the local site could be Raspberry Pi. Even though we have ordered (2 months ago) one Raspberry Pi, due to its high demand the first batch was sold instantly. Therefore we are waiting for the second batch.

References

[1] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," *Internet Request for Comments*, vol. RFC 3550 (Standard), jul 2003, updated by RFCs 5506, 5761, 6051, 6222. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3550.txt

[2] H. Schulzrinne, "A comprehensive multimedia control architecture for the internet," in *Network and Operating System Support for Digital Audio and Video, 1997., Proceedings of the IEEE 7th International Workshop on*, 1997, p. 6576.

[3] H. Sinnreich and A. B. Johnston, *Internet Communications Using SIP: Delivering VoIP and Multimedia Services with Session Initiation Protocol (Networking Council)*. New York, NY, USA: John Wiley & Sons, Inc., 2006.

[4] C. Jennings, F. Audet, and J. Elwell, "Session Initiation Protocol (SIP) URIs for Applications such as Voicemail and Interactive Voice Response (IVR)," *Internet Request for Comments*, vol. RFC 4458 (Informational), apr 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4458.txt

[5] J. Rosenberg, "A Framework for Conferencing with the Session Initiation Protocol (SIP)," *Internet Request for Comments*, vol. RFC 4353 (Informational), feb 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4353.txt

[6] J. Rodman, "VoIP to 20 kHz: Codec Choices for High Definition Voice Telephony," *Polycom White Papers*, jul 2008. [Online]. Available: http://www.polycom.com/global/documents/whitepapers/codecs_white_paper.pdf

[7] H. Schulzrinne, A. Rao, and R. Lanphier, "Real time streaming protocol (rtsp)," *Internet Request for Comments*, vol. RFC 2326 (Proposed Standard), apr 1998. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2326.txt

[8] P. Bertoldi and B. Atanasiu, "Code of Conduct on Energy Consumption of Broadband Equipment - Electricity consumption and efficiency trends in the enlarged European Union," *IESJRC. European Union*, Nov. 2008. [Online]. Available: http://ec.europa.eu/information_society/activities/sustainable_growth/docs/broadband_eq_code-conduct.pdf

[9] T. Wallingford, *Switching to VoIP "A solutions manual for netwok professionals". - Includes index.* Sebastopol, CA: O'Reilly, 2005.

[10] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," *Internet Request for Comments*, vol. RFC 3261 (Proposed Standard), jun 2002, updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922, 5954, 6026, 6141. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3261.txt

[11] M. Webb and M. Partner, "Breaking up is hard to do: The emergence of functional separation as a regulatory remedy," in *Proceedings of 8th International ITU Global Symposium of Regulators, Phuket*, 2008.

[12] K. Bianca, Gustafsson, D. Pamela, and F. Karin, "The swedish telecommunications market first half-year 2011," *Swedish Post and Telecom, Market Reports*, vol. PTS-ER-2011:21, Nov. 2011. [Online]. Available: http://www.pts.se/upload/Rapporter/Tele/2011/svtelem-halvar-2011-21-eng.pdf

[13] "In-Stat - VoIP penetration forecast to reach 79% of US businesses by 2013." [Online]. Available: http://www.instat.com/newmk.asp?ID=2721

[14] S. Lanzisera, B. Nordman, and R. Brown, "Data network equipment energy use and savings potential in buildings," *Energy Efficiency*, vol. 5, pp. 149–162, 2012, 10.1007/s12053-011-9136-4. [Online]. Available: http://dx.doi.org/10.1007/s12053-011-9136-4

[15] Google, "Googles Green Computing:Efficiency at Scale," *Google's Green Reports*, 2011. [Online]. Available: http://tinyurl.com/3dmnx4a

[16] S. Baset, J. Reich, J. Janak, P. Kasparek, V. Misra, D. Rubenstein, and H. Schulzrinnne, "How green is IP-telephony?" in *Proceedings of the first ACM SIGCOMM workshop on Green networking*, 2010, p. 7784.

[17] E. Commission, "COMMISSION RECOMMENDATION of 6 May 2003 concerning the definition of micro, small and medium-sized enterprises," *Official Journal of the European Union*, vol. L 124/37, may 2003. [Online]. Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2003:124:0036:0041:en:PDF

[18] EEA, "Household number and size in the EU," *Indicator Fact Sheet Signals 2001 Chapter Households*, vol. YIR01HH03, 2001. [Online]. Available: http://www.eea.europa.eu/data-and-maps/indicators/household-number-and-size/household-number-and-size/at_download/file

[19] D. Malas and A. Morton, "Basic Telephony SIP End-to-End Performance Metrics," *Internet Request for Comments*, vol. RFC 6076 (Proposed Standard), jan 2011. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6076.txt

[20] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," *Internet Request for Comments*, vol. RFC 2068 (Proposed Standard), jan 1997, obsoleted by RFC 2616. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2068.txt

[21] J. Prasad and B. Kumar, "Analysis of SIP and realization of advanced IP-PBX features," in *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*, vol. 6, 2011, p. 218222.

[22] OpenSIPS. (2012, mar) Open Source implementation of a SIP router/proxy server. [Online]. Available: http://opensips.org/

[23] Asterisk. (2012, mar) Asterisk is an open source framework for building communications applications - sponsored by Digium. [Online]. Available: http://www.asterisk.org/

[24] G. Q. M. Jr., "Practical voice over ip (VoIP): SIP and related protocols." [Online]. Available: http://www.ict.kth.se/courses/IK2554/VoIP-20110823a.pdf

[25] VLC. (2012, june) VideoLAN - VLC: Official site - Free multimedia solutions for all OS! [Online]. Available: http://www.videolan.org/index.html.

[26] M. Handley, V. Jacobson, and C. Perkins, "SDP: Session Description Protocol," *Internet Request for Comments*, vol. RFC 4566 (Proposed Standard), jul 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4566.txt

[27] H. Schulzrinne and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control," *Internet Request for Comments*, vol. RFC 3551 (Standard), jul 2003, updated by RFC 5761. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3551.txt

[28] H. Schulzrinne and T. Taylor, "RTP Payload for DTMF Digits, Telephony Tones, and Telephony Signals," *Internet Request for Comments*, vol. RFC 4733 (Proposed Standard), dec 2006, updated by RFCs 4734, 5244. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4733.txt

[29] D. Butcher, X. Li, and J. Guo, "Security challenge and defense in VoIP infrastructures," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1152–1162, Nov. 2007. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4346343

[30] J. Bilien, E. Eliasson, J. Orrblad, and J. olov Vatn, "Secure voip: call establishment and media protection," in *In 2nd Workshop on Securing Voice over IP*, 2005.

[31] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)," *Internet Request for Comments*, vol. RFC 3711 (Proposed Standard), mar 2004, updated by RFC 5506. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3711.txt

[32] J. Arkko, E. Carrara, F. Lindholm, M. Naslund, and K. Norrman, "MIKEY: Multimedia Internet KEYing," *Internet Request for Comments*, vol. RFC 3830 (Proposed Standard, aug 2004, updated by RFCs 4738, 6309. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3830.txt

[33] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing, "Session Traversal Utilities for NAT (STUN)," *Internet Request for Comments*, vol. RFC 5389 (Proposed Standard), oct 2008. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5389.txt

[34] R. Mahy, P. Matthews, and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)," *Internet Request for Comments*, vol. RFC 5766 (Proposed Standard), apr 2010. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5766.txt

[35] J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols," *Internet Request for Comments*, vol. RFC 5245 (Proposed Standard), apr 2010, updated by RFC 6336. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5245.txt

[36] A. Johnston, R. Sparks, C. Cunningham, S. Donovan, and K. Summers, "Session Initiation Protocol Service Examples," *Internet Request for Comments*, vol. RFC 5359 (Best Current Practice), oct 2008. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5359.txt

[37] J. Rosenberg and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)," *Internet Request for Comments*, vol. RFC 3264 (Proposed Standard), jun 2002, updated by RFC 6157. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3264.txt

[38] M. Badra, "NETCONF over Transport Layer Security (TLS)," *Internet Request for Comments*, vol. RFC 5539 (Proposed Standard), may 2009. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5539.txt

[39] R. Mahy, "A Message Summary and Message Waiting Indication Event Package for the Session Initiation Protocol (SIP)," *Internet Request for Comments*, vol. RFC 3842 (Proposed Standard), aug 2004. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3842.txt

[40] B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging," *Internet Request for Comments*, vol. RFC 3428 (Proposed Standard), dec 2002. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3428.txt

[41] T. Schmidt and M. Whlisch, "Group conference management with SIP," *SIP handbook: services, technologies, and security*, p. 123158, 2008.

[42] J. Li, W. Lei, and X. Zhang, "Design and implementation of a sip-based centralized multimedia conferencing system," in *Communication Software and Networks, 2009. ICCSN '09. International Conference on*, feb. 2009, pp. 40 –44.

[43] R. Mahy and D. Petrie, "The Session Initiation Protocol (SIP) "Join" Header," *Internet Request for Comments*, vol. RFC 3911 (Proposed Standard), oct 2004. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3911.txt

[44] R. Mahy, R. Sparks, J. Rosenberg, D. Petrie, and A. Johnston, "A Call Control and Multi-Party Usage Framework for the Session Initiation Protocol (SIP)," *Internet Request for Comments*, vol. RFC 5850 (Informational), may 2010. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5850.txt

[45] A. Johnston and O. Levin, "Session Initiation Protocol (SIP) Call Control - Conferencing for User Agents," *Internet Request for Comments*, vol. RFC 4579 (Best Current Practice), aug 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4579.txt

[46] R. Even and N. Ismail, "Conferencing Scenarios," *Internet Request for Comments*, vol. RFC 4597 (Informational), aug 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4597.txt

[47] J. Rosenberg, "Obtaining and Using Globally Routable User Agent URIs (GRUUs) in the Session Initiation Protocol (SIP)," *Internet Request for Comments*, vol. RFC 5627 (Proposed Standard), oct 2009. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5627.txt

[48] M. Mealling and R. Daniel, "The Naming Authority Pointer (NAPTR) DNS Resource Record," *Internet Request for Comments*, vol. RFC 2915 (Proposed Standard), sep 2000, obsoleted by RFCs 3401, 3402, 3403, 3404. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2915.txt

[49] A. Gulbrandsen, P. Vixie, and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)," *Internet Request for Comments*, vol. RFC 2782 (Proposed Standard), feb 2000, updated by RFC 6335. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2782.txt

[50] J. Rosenberg and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers," *Internet Request for Comments*, vol. RFC 3263 (Proposed Standard), jun 2002. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3263.txt

[51] S. Bradner, L. Conroy, and K. Fujiwara, "The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM)," *Internet Request for Comments*, vol. RFC 6116 (Proposed Standard), mar 2011. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6116.txt

[52] B. Hoeneisen, A. Mayrhofer, and J. Livingood, "IANA Registration of Enumservices: Guide, Template, and IANA Considerations," *Internet Request for Comments*, vol. RFC 6117 (Proposed Standard), mar 2011. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6117.txt

[53] Victor Delgado, "Exploring the limits of cloud computing," Masters Thesis, KTH Royal Institute of Technology, Stockholm, Sweden, nov 2010. [Online]. Available: http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/101118-Victor_Delgado-with-cover.pdf

[54] P. Mell and T. Grance, "The NIST definition of cloud computing," *NIST special publication*, vol. 800, p. 145, 2011. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[55] KVM. (2012, mar) KVM (for Kernel-based Virtual Machine) is a full open source virtualization solution for Linux on x86 hardware. [Online]. Available: http://www.linux-kvm.org/

[56] XEN. (2012, mar) Xen hypervisor is a powerful open source industry standard for virtualization. [Online]. Available: http://www.xen.org/

[57] A. Lenk, M. Klems, J. Nimis, S. Tai, and T. Sandholm, "What's inside the cloud? an architectural map of the cloud landscape," in *Software Engineering Challenges of Cloud Computing, 2009. CLOUD '09. ICSE Workshop on*, may 2009, pp. 23 –31.

[58] A. Lenk, M. Menzel, J. Lipsky, S. Tai, and P. Offermann, "What are you paying for? performance benchmarking for infrastructure-as-a-service offerings," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, july 2011, pp. 484 –491.

[59] C. Rigney, A. Rubens, W. Simpson, and S. Willens, "Remote Authentication Dial In User Service (RADIUS)," *Internet Request for Comments*, vol. RFC 2138 (Proposed Standard), apr 1997, obsoleted by RFC 2865. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2138.txt

[60] rsync. (2012, jun) backup tool - a software application and network protocol for Unix-like and Windows systems. [Online]. Available: http://rsync.samba.org/

[61] Nagios. (2012, may) The Industry Standard in IT Infrastructure Monitoring. [Online]. Available: http://nagios.org/

[62] P. Marwedel, *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems*. Springer Verlag, 2010.

[63] W. Wolf, *Computers as components: principles of embedded computing system design*. Morgan Kaufmann, 2008.

[64] "Broadcom.com - BCM5352EL - AirForce - 802.11g Router System-on-Chip with BroadRange Technology." [Online]. Available: http://www.broadcom.com/products/Wireless-LAN/802.11-Wireless-LAN-Solutions/BCM5352EL

[65] C. Heffner and J. Collake. (2012, mar) Firmware Modification Kit - documentation version 0.73 beta. [Online]. Available: http://code.google.com/p/firmware-mod-kit/

[66] "Linksys WRT54G-TM SD/MMC mod - DD-WRT wiki." [Online]. Available: http://www.dd-wrt.com/wiki/index.php/Linksys_WRT54G-TM_SD/MMC_mod

[67] (2012, mar) Raspberry pi | an ARM GNU/Linux box for $25. [Online]. Available: http://www.raspberrypi.org/

[68] "Ieee standard for information technology–telecommunications and information exchange between systems–local and metropolitan area networks–specific requirements part 3: Carrier sense multiple access with collision detection (csma/cd) access method and physical layer specifications amendment 5: Media access control parameters, physical layers, and management parameters for energy-efficient ethernet," *IEEE Std 802.3az-2010 (Amendment to IEEE Std 802.3-2008)*, pp. 1 –302, 27 2010.

[69] EU, "Commission Regulation (EC) No 278/2009 of 6 April 2009 implementing Directive 2005/32/EC of the European Parliament and of the Council with regard to ecodesign requirements for no-load condition electric power consumption and average active efficiency of external power supplies," *Official Journal of the European Union*, vol. L93/3, Apr. 2009. [Online]. Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:093:0003:0010:EN:PDF

[70] S. Nedevschi, J. Chandrashekar, J. Liu, B. Nordman, S. Ratnasamy, and N. Taft, "Skilled in the art of being idle: reducing energy waste in networked systems," in *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, ser. NSDI'09. Berkeley, CA, USA: USENIX Association, 2009, pp. 381–394. [Online]. Available: http://dl.acm.org/citation.cfm?id=1558977.1559003

[71] A. Virolainen and M. Saaranen, "Networked power management for home multimedia," in *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*, 2008, p. 331332.

[72] M. Mostowfi and K. Christensen, "Saving energy in lan switches: New methods of packet coalescing for energy efficient ethernet," in *Green Computing Conference and Workshops (IGCC), 2011 International*, july 2011, pp. 1 –8.

[73] P. Bertoldi, "Code of Conduct on Energy Efficiency of External Power Supplies," *IESJRC. European Union*, Apr. 2009. [Online]. Available: http://re.jrc.ec.europa.eu/energyefficiency/pdf/CoC_Power_Supplies_Version4-March2009.pdf

[74] E. Svensson, "A First Step Toward Green Wireline Broadband : A tool for systematic measurement of Digital Subscriber Line parameters as input to dynamic power optimization algorithms," Masters Thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2011. [Online]. Available: http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/111219-Eric_Svensson-with-cover.pdf

[75] M. Associates. (2012, apr) Open 79XX XML Directory. [Online]. Available: http://www.infradapt.com/csma_apps/xml_xmldir.php

[76] G. Almes, S. Kalidindi, and M. Zekauskas, "A One-way Delay Metric for IPPM," *Internet Request for Comments*, vol. RFC 2679 (Proposed Standard), sep 1999. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2679.txt

[77] G. Almes, "A Round-trip Delay Metric for IPPM," *Internet Request for Comments*, vol. RFC 2681 (Proposed Standard), sep 1999. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2681.txt

[78] D. Mills, "Network Time Protocol (Version 3) Specification, Implementation and Analysis," *Internet Request for Comments*, vol. RFC 1305 (Draft Standard), mar 1992, obsoleted by RFC 5905. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1305.txt

[79] J. Filliben *et al.*, "Nist/semtech engineering statistics handbook," *Gaithersburg: www.itl.nist.gov/div898/handbook, NIST*, 2002. [Online]. Available: http://www.itl.nist.gov/div898/handbook/

[80] M. Qadeer and A. Imran, "Asterisk voice exchange: An alternative to conventional epbx," in *Computer and Electrical Engineering, 2008. ICCEE 2008. International Conference on*, dec. 2008, pp. 652 –656.

[81] Richard Gayraud and Olivier Jacques, "Welcome to SIPp," http://sipp.sourceforge.net/, oct 2010. [Online]. Available: http://sipp.sourceforge.net/

[82] "D-ITG A tool for the generation of realistic network workload for emerging networking scenarios," oct 2012. [Online]. Available: http://www.grid.unina.it/software/ITG/

[83] "Iperf A tool for measuring maximum TCP and UDP bandwidth performance." oct 2012. [Online]. Available: http://sourceforge.net/projects/iperf/

[84] Aastra. (2012, may) Aastra 6730i SIP business deskphone. [Online]. Available: http://www.aastra.ca/document-library.htm?curr_nav=2&curr_fam=Aastra+6730i&prod_id=6167

[85] VoipInfo. (2012, may) Complete Guide for configuring various Cisco 79XX IP Phones with Asterisk. [Online]. Available: http://www.voip-info.org/wiki/view/Asterisk+phone+cisco+79xx

[86] "X-Lite A SIP softphone from CounterPath coorporation." oct 2012. [Online]. Available: http://www.counterpath.com/x-lite.html

[87] dnsmasq. (2012, may) Dnsmasq - a DNS forwarder for NAT firewalls. [Online]. Available: http://www.thekelleys.org.uk/dnsmasq/doc.html

[88] atftpd. (2012, may) Advanced TFTP server/client Freecode. [Online]. Available: http://freecode.com/projects/atftp

[89] OpenVPN. (2012, may) OpenVPN - Open Source VPN. [Online]. Available: http://openvpn.net/index.php/open-source/overview.html

[90] M. bin Mohd Shuhaimi, I. binti Roslan, Z. binti Zainal Abidin, and S. binti Anawar, "The new services in nagios: Network bandwidth utility, email notification and sms alert in improving the network performance," in *Information Assurance and Security (IAS), 2011 7th International Conference on*, dec. 2011, pp. 86 –91.

[91] FreeRadius. (2012, may) FreeRADIUS: The worlds most popular RADIUS Server. [Online]. Available: http://freeradius.org/

[92] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2008, ISBN 3-900051-07-0. [Online]. Available: http://www.R-project.org

A Methodology mind map for the experiments

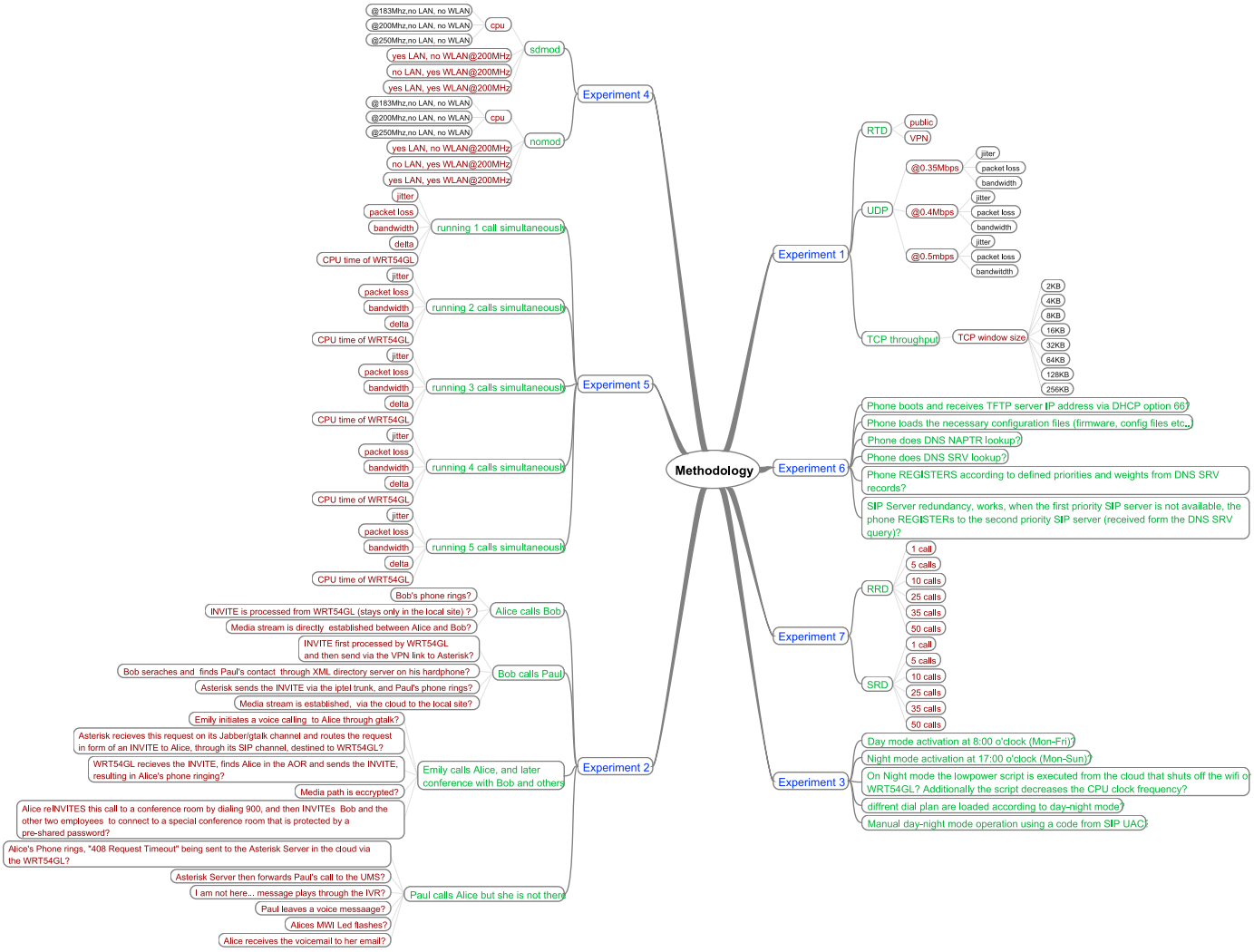


Figure 52: This figure represents the scenarios (for experiment 1,4,5, and 7) and tasks (for experiment 2,3, and 6) for testing, in each of the experiments from section 5.

B Asterisk configuration files

extensions.conf

```
[general]
static=yes
writeprotect=no
clearglobalvars=no
[globals]
CONSOLE=Console/dsp
[local]
include => voicemail
include => default
include => conference
include => internal-sip
include => trunklocal
include => parkedcalls
ignorepat => 9
exten => _1XX,2,Dial(SIP/${EXTEN},3)
exten => _1XX,3,voicemail(${EXTEN},u)
exten => s,1,Answer()
exten => s,n,Wait(0.1)
exten => s,n,SendDTMF(1)
exten => s,n,Set(name=${CALLERID(name)})
exten => s,n,Set(num=${CUT(name,@,1)})
exten => s,n,Set(CALLERID(all)=${num})
exten => s,n,Dial(SIP/lowpasterisk,8,tT)
exten => s,n,Voicemail(lowpasterisk,u)
exten => s,n,Playback(vm-goodbye)
exten => s,n,Hangup()
exten => 300,1,Dial(gtalk/asterisk/talaganov@gmail.com)
exten => _1XXXXXXXXXX,1,Dial(gtalk/asterisk/+${EXTEN}@voice.google.com)
exten => 700,1,System(/var/lib/asterisk/scripts/lowpower-night.sh)
exten => 800,1,System(/var/lib/asterisk/scripts/lowpower-day.sh)
exten => 888,1,Dial(SIP/goctala@iptel)
exten => 888,n,Playback(invalid)
exten => 888,n,Hangup
exten => 887,1,Dial(SIP/korobar@iptel)
exten => 887,n,Playback(invalid)
exten => 887,n,Hangup
[time]
exten => _X.,30000(time),NoOp(Time: ${EXTEN} ${timezone})
exten => _X.,n,Wait(0.25)
exten => _X.,n,Answer()
exten => _X.,n,Set(FUTURETIME=$[${EPOCH} + 12])
exten => _X.,n,SayUnixTime(${FUTURETIME},Zulu,HNS)
exten => _X.,n,SayPhonetic(z)
exten => _X.,n,SayUnixTime(${FUTURETIME},${timezone},HNS)
exten => _X.,n,Playback(spy-local)
exten => _X.,n,WaitUntil(${FUTURETIME})
exten => _X.,n,Playback(beep)
exten => _X.,n,Return()
[conference]
exten => 900,1,Meetme(1234,ocMspI)
exten => 900,2,hangup
[voicemail]
exten => *91,1,answer
exten => *91,2,voicemailmain(${CALLERID(num)})
exten => *91,3,hangup
[internal-sip]
include => voicemail
include => conference
include => local
exten => _1XX,1,answer
exten => _1XX,2,Dial(SIP/${EXTEN},15)
exten => _1XX,3,voicemail(${EXTEN},u)
exten => 888,1,Dial(SIP/goctala@iptel)
exten => 888,n,Playback(invalid)
exten => 888,n,Hangup
exten => 887,1,Dial(SIP/korobar@iptel)
exten => 887,n,Playback(invalid)
exten => 887,n,Hangup
[sipp]
exten => 1001,1,Answer
```

```
exten => 1001,n,SetMusicOnHold(default)
exten => 1001,n,WaitMusicOnHold(20)
exten => 1001,n,Hangup
exten => 1002,1,Answer
exten => 1002,n,Goto(demo,s,1)
exten => 1002,n,Hangup
```

sip.conf

```
[general]
context=default
allowoverlap=no
udpbindaddr=0.0.0.0
register => TCP://lowpasterisk@iptel.org:<******>@iptel.org
tcpbindaddr=0.0.0.0
srvlookup=yes
localnet=192.168.1.0/24; RFC 1918 addresses
localnet=192.168.100.0/24
externaddr = 77.28.100.181
externaddr = 10.8.0.1
nat = yes
tcpenable=yes
[authentication]
[basic-options](!)
dtmfmode=rfc2833
context=from-office
type=friend
[natted-phone](!,basic-options)
nat=yes
directmedia=no
host=dynamic
[public-phone](!,basic-options)
nat=no
directmedia=yes
[my-codecs](!)
disallow=all
allow=ulaw
allow=ilbc
allow=g729
allow=gsm
allow=g723
[ulaw-phone](!)
disallow=all
allow=ulaw
[test]
type=friend
allowguest=yes
context=internal-sip
host=dynamic
user=test
secret=1234
transport=udp
disallow=all
allow=ulaw
[gocet]
type=friend
context=local
host=dynamic
user=gocet
secret=1234
transport=udp
dtmfmode=rfc2833
disallow=all
allow=ulaw
mailbox=gocet
[100]
type=friend
context=internal-sip
host=dynamic
user=100
secret=1234
transport=udp
disallow=all
allow=ulaw
```

```
mailbox=100
[101]
type=friend
context=internal-sip
host=dynamic
user=101
secret=1234
transport=udp
disallow=all
allow=ulaw
[102]
type=friend
context=internal-sip
host=dynamic
user=102
secret=1234
transport=udp
disallow=all
allow=ulaw
[1000]
type=friend
secret=1000
host=dynamic
context=local
[sipp]
type=friend
context=sipp
host=dynamic
port=6000
user=sipp
canreinvite=no
disallow=all
allow=alaw
allow=ulaw
[iptel]
type=friend
username=lowpasterisk
secret=********
host=iptel.org
fromdomain=iptel.org
nat=yes
transport=tcp
qualify=yes
context=internal-sip
canreinvite=no
[lowpasterisk]
type=friend
username=lowpasterisk
secret=1234
host=dynamic
canreinvite=no
context=local
dtmfmode=rfc2833
disallow=all
allow=ulaw
nat=yes
mailbox=lowpasterisk
[goctala]
type=friend
username=goctala
secret=*******
host=iptel.org
fromdomain=iptel.org
nat=yes
transport=tcp
qualify=yes
context=internal-sip
canreinvite=no
```

gtalk.conf

```
[general]
context=local
allowguests=yes
bindaddr=0.0.0.0
```

```
externip=77.28.100.181
[guest]
disallow=all
allow=ulaw
context=local
connection=asterisk
```

jabber.conf

```
[general]
autoregister=yes
autoprune=no
debug=no
[asterisk]
type=client
serverhost=talk.google.com
username=goctala.korisnik@gmail.com/Talk
secret=********
port=5222
usetls=yes
usesasl=yes
statusmessage="@HisAsteriskServer"
timeout=100
```

meetme.conf

```
[rooms]
conf => 1234,12345,123456
```

voicemail.conf

```
[general]
format=wav49|gsm|wav
serveremail=asterisk
attach=yes
skipms=3000
maxsilence=10
silencethreshold=128
maxlogins=3
emaildateformat=%A, %B %d, %Y at %r
pagerdateformat=%A, %B %d, %Y at %r
sendvoicemail=yes ; Allow the user to compose and send a voicemail while inside
[zonemessages]
eastern=America/New_York|'vm-received' Q 'digits/at' IMp
central=America/Chicago|'vm-received' Q 'digits/at' IMp
central24=America/Chicago|'vm-received' q 'digits/at' H N 'hours'
military=Zulu|'vm-received' q 'digits/at' H N 'hours' 'phonetic/z_p'
european=Europe/Copenhagen|'vm-received' a d b 'digits/at' HM
[default]
100 => 1234,Goce Talaganov,talaganov@gmail.com,,|tz=european|attach=yes|saycid=yes
101 => 1234,Vedra Korobar,talaganov@gmail.com,,|tz=european|attach=yes|saycid=yes
1000 => 1000,Goce Talaganov,talaganov@gmail.com,,|tz=european|attach=yes|saycid=yes
gocet => 1234,Goce Talaganov,talaganov@gmail.com,,|tz=european|attach=yes|saycid=yes
lowpasterisk => 1234,Goce Talaganov,talaganov@gmail.com,,|tz=european|attach=yes|saycid=yes
[other]
1234 => 5678,Company2 User,root@localhost
```

C Lowpower Scripts

lowpower-day.sh

```
#!/bin/sh
#/usr/bin/ssh root@10.8.0.6 'wl radio on'
#/usr/bin/ssh root@10.8.0.6 'sleep 5; startservice httpd'
/usr/bin/ssh root@10.8.0.6 'nvram set vlan0ports="0 1 2 3 5*" '
/usr/bin/ssh root@10.8.0.6 'sleep 1; nvram set wl0_net_mode=enabled'
/usr/bin/ssh root@10.8.0.6 'sleep 1; nvram set overclocking=200'
/usr/bin/ssh root@10.8.0.6 'sleep 1; nvram set clkfreq=200'
/usr/bin/ssh root@10.8.0.6 'sleep 1; nvram set http_enable=1'
/usr/bin/ssh root@10.8.0.6 'sleep 1; nvram set httpd_enable=1'
/usr/bin/ssh root@10.8.0.6 'sleep 1; nvram commit'
/usr/bin/ssh root@10.8.0.6 'sleep 1; sync'
/usr/bin/ssh root@10.8.0.6 'sleep 1; reboot'
```

lowpower-night.sh

```
#!/bin/sh
#/usr/bin/ssh root@10.8.0.6 'wl radio off'
#/usr/bin/ssh root@10.8.0.6 'sleep 5; stopservice httpd'
/usr/bin/ssh root@10.8.0.6 'nvram set vlan0ports="0 1 5*" '
/usr/bin/ssh root@10.8.0.6 'sleep 1; nvram set wl0_net_mode=disabled'
/usr/bin/ssh root@10.8.0.6 'sleep 1; nvram set overclocking=183'
/usr/bin/ssh root@10.8.0.6 'sleep 1; nvram set clkfreq=183'
/usr/bin/ssh root@10.8.0.6 'sleep 1; nvram set http_enable=0'
/usr/bin/ssh root@10.8.0.6 'sleep 1; nvram set httpd_enable=0'
/usr/bin/ssh root@10.8.0.6 'sleep 1; nvram commit'
/usr/bin/ssh root@10.8.0.6 'sleep 1; sync'
/usr/bin/ssh root@10.8.0.6 'sleep 1; reboot'
```

D OpenSER

openser.cfg

```
# ----------- global configuration parameters ------------------------

debug=3            # debug level (cmd line: -dddddddddd)
fork=yes
log_stderror=no    # (cmd line: -E)
sip_warning=0
/*
Uncomment these lines to enter debugging mode
fork=no
log_stderror=yes
*/

check_via=no # (cmd. line: -v)
dns=no           # (cmd. line: -r)
rev_dns=no       # (cmd. line: -R)
port=5060
listen=udp:br0:5060
listen=udp:tun0:5060
disable_tcp=yes
fifo="/tmp/openser_fifo"

mhomed=1

#
# uncomment the following lines for TLS support
#disable_tls = 0
#listen = tls:your_IP:5061
#tls_verify = 1
#tls_require_certificate = 0
#tls_method = TLSv1
#tls_certificate = "/usr/etc/openser/tls/user/user-cert.pem"
#tls_private_key = "/usr/etc/openser/tls/user/user-privkey.pem"
#tls_ca_list = "/usr/etc/openser/tls/user/user-calist.pem"

# ------------------ module loading ----------------------------------

# Uncomment this if you want to use SQL database
loadmodule "/usr/lib/openser/modules/dbtext.so"
loadmodule "/usr/lib/openser/modules/sl.so"
loadmodule "/usr/lib/openser/modules/tm.so"
loadmodule "/usr/lib/openser/modules/rr.so"
loadmodule "/usr/lib/openser/modules/maxfwd.so"
loadmodule "/usr/lib/openser/modules/usrloc.so"
loadmodule "/usr/lib/openser/modules/registrar.so"
loadmodule "/usr/lib/openser/modules/textops.so"
loadmodule "/usr/lib/openser/modules/exec.so"
loadmodule "/usr/lib/openser/modules/milkfish_nathelper.so"

# Uncomment this if you want digest authentication
# mysql.so must be loaded !
loadmodule "/usr/lib/openser/modules/auth.so"
loadmodule "/usr/lib/openser/modules/auth_db.so"

# ----------------- setting module-specific parameters ---------------
modparam("usrloc", "db_mode", 2)
modparam("usrloc|auth_db", "db_url", "dbtext:///var/openser/dbtext")
modparam("auth_db", "calculate_ha1", 1)
modparam("auth_db", "password_column", "password")
modparam("auth_db", "user_column", "username")
modparam("auth_db", "domain_column", "domain")
modparam("rr", "enable_full_lr", 1)

# -------------------------  request routing logic -------------------

route{

        exec_msg("/usr/sbin/mf_sip_tracer.sh");

# initial sanity checks -- messages with
# max_forwards==0, or excessively long requests
if (!mf_process_maxfwd_header("10")) {
sl_send_reply("483","Too Many Hops");
exit;
};

if (msg:len >=  2048 ) {
sl_send_reply("513", "Message too big");
exit;
};

        # check for Registrations
        if (method=="REGISTER")
        {
                if (uri==myself)
                {
                        # wants to register only at WRT54GL, not at the cloud site
            if (!www_authorize("", "subscriber")) {
             www_challenge("", "0");
        exit;
        };
                        save("location");
                }
                else
```

```
                {
                        # wants to register at the cloud site
                        # log(1, "external REGISTER\n");
                        # check if user is already registered at internal registrar
                        if (!lookup("location"))
                        {

                                save_noreply("location");
                        };

remove_hf("Route");
        # address is wan address, port is default sip port 5060
                        fix_nated_contact("10.8.0.6");
                        route(1);
                };
                return;

        }
        else
{
record_route();
};

        # Check if package originates from LAN
        if (src_ip==192.168.100.1/255.255.255.0)
        {
                if (loose_route()) {
                         # mark routing logic in request
                         # log(1, "loose route\n");
                         append_hf("P-hint: rr-enforced\r\n");
                         route(1);
                };

                # log(1, "incoming package from lan\n");
                if (lookup("location") or lookup("aliases"))
                {       # if (uri=~"^sip:9911@.*") {
                        # rewriteuser("900");
                        # rewritehostport("10.8.0.1:5060");
                        # forward(uri:host, uri:port);
                        # exit;
                         # };
                        # log(1, "contact in local database\n");
                        if (uri=~"^sip:.*0*192.0*168.0*100")
                        {
                                # at least one command here
                                log(1, "internal call\n");
                        }
                        else
                        {

                                fix_nated_contact("10.8.0.6");
                                subst('/^From:(.*)sip:(.+)@0*192.0*168.0*100[0-9.]*(.*)$/From:\1sip:\2@10.8.0.6\3/ig');
                                if (method=="INVITE")
                                {
                                        setflag(5);
                                        t_on_reply("1");
                                };

                        };
                }

                        else
                        {

                                fix_nated_contact("10.8.0.6");
                                subst('/^From:(.*)sip:(.+)@0*192.0*168.0*100[0-9.]*(.*)$/From:\1sip:\2@10.8.0.6\3/ig');
                                if (method=="INVITE")
                                {
                                        setflag(5);
                                        t_on_reply("1");
                                };
                        };
                };
        }
        else
        {
                # log(1, "incoming package from wan\n");
                if (uri==myself)
                {
                        if (lookup("location") or lookup("aliases"))
                        {
                                # log(1, "contact in local database\n");
                                if (method=="INVITE" and uri=~"^sip:.*0*192.0*168.0*100")
                                {
                                        # log(1, "call from wan to lan\n");
                                        setflag(5);
                                        t_on_reply("1");
                                };
                        }
                        else
                        {
                                # log(1, "contact not found\n");
        sl_send_reply("404", "Not Found");
        return;
        };
                };

        };
        route(1);
```

```
}


onreply_route[1]
{
        # Check if package originates from LAN
        if (src_ip==192.168.100.1/255.255.255.0)
        {
                # log(1, "outgoing reply\n");
                # Fixing of private address in contact hf
                # address is wan address, port is default sip port 5060
                fix_nated_contact("10.8.0.6");
        }
else
        {
                # log(1, "incoming reply\n");
}

route[1] {
# send it out now; use stateful forwarding as it works reliably
# even for UDP2TCP
if (!t_relay()) {
sl_reply_error();
};
exit;
}
```

E SIP hardphones configuration files

aastra.cfg (6730i)

```
dhcp: 1 # DHCP enabled.
sip registration time: 100 # Eg. every 300 seconds, a re-register
# request is sent to the SIP server.
sip rtp port: 3000 # Eg. RTP packets are sent to port 3000.
sip silence suppression: 2 # "0" = off, "1" = on, "2" = default
sip proxy ip: lowpowersip.org # IP of proxy server.
sip proxy port: 5060 # 5060 is set by default.
sip registrar ip: 10.8.0.1 # IP of registrar.
sip registrar port: 5060 # 5060 is set by default.
sip digit timeout: 6
time server1: 10.8.0.1
time zone name: MK-Skopje
time zone code: CET
time format: 1 # 0=12 hr, 1=24 hr
date format: 4 # 4=dd/mm/yy
sip line1 auth name: 101
sip line1 password: 1234
sip line1 user name:100
sip line1 display name: Goce Talaganov
sip line1 screen name: Goce Talaganov
```

SIPDefault.cnf (Cisco 7940G)

```
image_version: "P0S3-8-12-00"
proxy_register: "1"
proxy1_address: "10.8.0.1"
proxy1_port:"5060"
outbound_proxy: "lowpowersip.org"
outbound_proxy_port: ""
nat_enable: ""
nat_address: ""
voip_control_port: "5060"
start_media_port: "16348"
end_media_port: "20134"
nat_received_processing: ""
dyn_dns_addr_1: ""
dyn_dns_addr_2: ""
tftp_cfg_dir: "./"
timer_register_expires: "30"
preferred_codec: g711ulaw
tos_media: "1"
enable_vad: "0"
network_media_type: "auto"
autocomplete: "1"
telnet_level: "2"
cnf_join_enable: "1"
semi_attended_transfer: "0"
call_waiting: "1"
anonymous_call_block: "0"
callerid_blocking: "0"
dnd_control: "0"
dtmf_inband: "1"
dtmf_outofband: "avt"
dtmf_db_level: "3"
dtmf_avt_payload: "101"
timer_t1: "500"
timer_t2: "4000"
sip_retx: "10"
sip_invite_retx: "6"
timer_invite_expires: "180"
sntp_mode: "unicast"
sntp_server: "10.8.0.1"
time_zone: "CEST"
dst_offset: "1"
dst_start_month: "March"
dst_start_day: ""
dst_start_day_of_week: "Sun"
dst_start_week_of_month: "4"
dst_start_time: "02"
dst_stop_month "Oct"
```

```
dst_stop_day: ""
dst_stop_day_of_week: "Sunday"
dst_stop_week_of_month: "8"
dst_stop_time: "2"
dst_auto_adjust: "1"
messages_uri: "*91"
services_url: "http://10.8.0.1/directory/PhoneUI/index.php"
directory_url: "http://10.8.0.1/directory/PhoneUI/searchdirectory.php"
logo_url: "http://10.8.0.1/asterisk-tux.bmp"
```

F atftpd

atftp (running through xinetd)

```
#
# atftp
#
service tftp
{
flags = REUSE
socket_type = dgram
protocol = udp
port = 69
instances = 30
wait = yes
user = nobody
server = /opt/sbin/atftpd
server_args = /opt/tftpboot
log_on_success = HOST PID
log_on_failure = HOST
disable = no
}
```

G D-ITG and iperf configuration

G.1 Iperf testing, TCP throughput commands

From each machine both client and server were run with the same advertising and receiving TCP windows sizes:

```
Client commands:
iperf -c 192.168.100.137 -F /home/gocet/6MB.mp3 -w 2k
iperf -c 192.168.100.137 -F /home/gocet/6MB.mp3 -w 4k
iperf -c 192.168.100.137 -F /home/gocet/6MB.mp3 -w 8k
iperf -c 192.168.100.137 -F /home/gocet/6MB.mp3 -w 16k
iperf -c 192.168.100.137 -F /home/gocet/6MB.mp3 -w 32k
iperf -c 192.168.100.137 -F /home/gocet/6MB.mp3 -w 64k
iperf -c 192.168.100.137 -F /home/gocet/6MB.mp3 -w 128k
iperf -c 192.168.100.137 -F /home/gocet/6MB.mp3 -w 256k

Server commands:
iperf -s -B 10.8.0.1 -w 2k
iperf -s -B 10.8.0.1 -w 4k
iperf -s -B 10.8.0.1 -w 8k
iperf -s -B 10.8.0.1 -w 16k
iperf -s -B 10.8.0.1 -w 32k
iperf -s -B 10.8.0.1 -w 64k
iperf -s -B 10.8.0.1 -w 128k
iperf -s -B 10.8.0.1 -w 256k
```

G.2 D-ITG testing OWD

An NTP server was configured on both sides (send and receive) that synchronize with stratum2 server, the commands are:

```
./ITGSend -m owdm -a 10.8.0.1 -c 172 -C 50 -T UDP -t 600000 /sender
./ITGRecv -l logfile //receiver
./ITGDec logfile -d 1 //distil a delay-only log file
octave ../src/ITGPlot/ITGplot delayfromlocal.txt //plot the delay log file
ntpq p //check the synchronization and offset of the NTP serve
```

G.3 Iperf and D-ITG for VoIP (G.711 u-law) call emulation

Emulating VoIP calls using D-ITG
Sender:

```
/ITGSend -a <ipaddress> -C 200 -rp 10001 VoIP -x G.711.2 -h RTP VAD
/ITGSend -a <ipaddress> -C 250 -rp 10001 VoIP -x G.711.2 -h RTP VAD
/ITGSend -a <ipaddress> -C 350 -rp 10001 VoIP -x G.711.2 -h RTP VAD
```

Reciever:

```
/ITGRecv -l logfile
```

VoIP call emulation according to bandwidth input:
Sender(s):

```
iperf -u -c <ipaddress> -b 0.35m -l 172
iperf -u -c <ipaddress> -b 0.4m -l 172
iperf -u -c <ipaddress> -b 0.5m -l 172
```

Reciever:

```
iperf -s -u -B <ipaddress>
```

H SIPp

H.1 Register scenario xml

```
<?xml version="1.0" encoding="ISO-8859-2" ?>

<!--  Use with CSV file struct like: 3000;192.168.1.106;[authentication username=3000 password=3000];
      (user part of uri, server address, auth tag in each line)
-->

<scenario name="register_client">
  <send retrans="500">
    <![CDATA[

      REGISTER sip:[remote_ip] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: <sip:[field0]@[field1]>;tag=[call_number]
      To: <sip:[field0]@[field1]>
      Call-ID: [call_id]
      CSeq: [cseq] REGISTER
      Contact: sip:[field0]@[local_ip]:[local_port]
      Max-Forwards: 10
      Expires: 120
      User-Agent: SIPp/Win32
      Content-Length: 0

    ]]>
  </send>

  <!-- asterisk -->
  <recv response="100" optional="true">
  </recv>

  <recv response="401" auth="true">
  </recv>

  <send retrans="500">
    <![CDATA[

      REGISTER sip:[remote_ip] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: <sip:[field0]@[field1]>;tag=[call_number]
      To: <sip:[field0]@[field1]>
      Call-ID: [call_id]
      CSeq: [cseq] REGISTER
      Contact: sip:[field0]@[local_ip]:[local_port]
      [field2]
      Max-Forwards: 10
      Expires: 120
      User-Agent: SIPp/Win32
      Content-Length: 0

    ]]>
  </send>

  <!-- asterisk -->
  <recv response="100" optional="true">
  </recv>

  <recv response="200">
  </recv>

  <!-- response time repartition table (ms)   -->
  <ResponseTimeRepartition value="10, 20, 30, 40, 50, 100, 150, 200"/>

  <!-- call length repartition table (ms)     -->
  <CallLengthRepartition value="10, 50, 100, 500, 1000, 5000, 10000"/>

</scenario>
```

H.2 The "fields" from the xml file are fed using the following csv info files

```
SEQUENTIAL
110;192.168.100.1;[authentication username=110 password=1234];
111;192.168.100.1;[authentication username=110 password=1234];
112;192.168.100.1;[authentication username=110 password=1234];
113;192.168.100.1;[authentication username=110 password=1234];
114;192.168.100.1;[authentication username=110 password=1234];
115;192.168.100.1;[authentication username=110 password=1234];
.
.
.
.
.
.
160;192.168.100.1;[authentication username=110 password=1234];
```

H.3 The commands used to trigger the scenarios with the desired csv info and parameters were:

Registrating 1, 5, 10, 25, 35, and 50 extensions at once at WRT54GL that are run 10 times each:

```
./sipp 192.168.100.1 -sf REGISTER_client.xml -inf REGISTER_client.csv -r 1 -l 1 -m 1 -nd -i 192.168.100.137
./sipp 192.168.100.1 -sf REGISTER_client.xml -inf REGISTER_client.csv -r 5 -l 5 -m 5 -nd -i 192.168.100.137
./sipp 192.168.100.1 -sf REGISTER_client.xml -inf REGISTER_client.csv -r 10 -l 10 -m 10 -nd -i 192.168.100.137
./sipp 192.168.100.1 -sf REGISTER_client.xml -inf REGISTER_client.csv -r 25 -l 25 -m 25 -nd -i 192.168.100.137
./sipp 192.168.100.1 -sf REGISTER_client.xml -inf REGISTER_client.csv -r 35 -l 35 -m 35 -nd -i 192.168.100.137
./sipp 192.168.100.1 -sf REGISTER_client.xml -inf REGISTER_client.csv -r 50 -l 50 -m 50 -nd -i 192.168.100.137
```

Session invites for 1, 5, 10, 25, 35 and 50 simultaneous calls, using WRT54GL as a SIP proxy, each runs 10 times:

```
./sipp -sn uac -s 1003 192.168.100.1 -r 1 -l 1 -m 1 -i 192.168.100.137 -p 5062 nd
./sipp -sn uac -s 1003 192.168.100.1 -r 5 -l 5 -m 5 -i 192.168.100.137 -p 5062 -nd
./sipp -sn uac -s 1003 192.168.100.1 -r 10 -l 10 -m 10 -i 192.168.100.137 -p 5062 nd
./sipp -sn uac -s 1003 192.168.100.1 -r 25 -l 25 -m 25 -i 192.168.100.137 -p 5062 -nd
./sipp -sn uac -s 1003 192.168.100.1 -r 35 -l 35 -m 35 -i 192.168.100.137 -p 5062 nd
./sipp -sn uac -s 1003 192.168.100.1 -r 50 -l 50 -m 50 -i 192.168.100.137 -p 5062 -nd
./sipp -sn uas
```

An example for using sipsak to register an account to e.g. iptel (using UDP as transport)

```
sipsak -U -C sip:goctala@iptel.org -a ******* -s sip:goctala@iptel.org
-vvvvvvv -E udp
```

H.4 OpenVPN configuration

localsite.cfg

```
client
dev tun
proto udp
remote 77.28.100.181 1194
resolv-retry infinite
nobind
persist-key
persist-tun
mute-replay-warnings
ca /opt/etc/openvpn/ca.crt
cert /opt/etc/openvpn/petko.crt
key /opt/etc/openvpn/petko.key
cipher AES-256-CBC
ns-cert-type server
comp-lzo
verb 4
mute 20
```

cloudsite.cfg

```
port 1194
proto udp
dev tun
ca /etc/openvpn/ca.crt
cert /etc/openvpn/goctala.crt
key /etc/openvpn/goctala.key # This file should be kept secret
dh /etc/openvpn/dh1024.pem
server 10.8.0.0 255.255.255.0
ifconfig-pool-persist /etc/openvpn/ipp.txt
client-config-dir ccd
route 192.168.100.0 255.255.255.0
push "dhcp-option DNS 10.8.0.1"
client-to-client
keepalive 10 120
cipher AES-256-CBC
comp-lzo
max-clients 10
daemon
persist-key
persist-tun
status /etc/openvpn/openvpn-status.log
log /var/log/openvpn/openvpn.log
verb 6
mute 20
```

I Nagios configuration file

```
# Define the switch that we'll be monitoring

define host{
        use             generic-switch                  ; Inherit default values from a template
        host_name       RTA                             ; The name we're giving to this switch
        alias           DD-WRT v24-sp2- build 14929     ; A longer name associated with the switch
        address         10.8.0.6                        ; IP address of the switch
        hostgroups      routers                         ; Host groups this switch is associated with
        }


define hostgroup{
        hostgroup_name          routers                 ; The name of the hostgroup
        alias                   SME and home routers    ; Long name of the group
        hostgroup_members       routers
        }

# Create a service to PING to switch

define service{
        use                     generic-service ; Inherit values from a template
        host_name               RTA             ; The name of the host the service is associated with
        service_description     PING            ; The service description
        check_command           check_ping!200.0,20%!600.0,60%  ; The command used to monitor the service
        normal_check_interval   5               ; Check the service every 5 minutes under normal conditions
        retry_check_interval    1               ; Re-check the service every minute until its final/hard state is
        }

# Monitor uptime via SNMP
define service{
        use                     generic-service
        host_name               RTA
        service_description     Uptime
        check_command           check_snmp!-C lowpowersip -o sysUpTime.0
        }

# Monitor SSH
define service{
        use                             generic-service         ; Name of service template to use
        host_name                       RTA
        service_description             SSH
        check_command                   check_ssh
        notifications_enabled           0
        }


# Monitor WAN vlan1 interface status on WRT54GL via SNMP
define service{
        use                     generic-service
        host_name               RTA
        service_description     WAN vlan1 interface status
        check_command           check_snmp!-C lowpowersip -o ifAdminStatus.6 -c 1 -m RFC1213-MIB
        }

define service{
        use                     generic-service
        host_name               RTA
        service_description     LAN br0 interface status
define service{
        use                     generic-service
        host_name               RTA
        service_description     tun0 interface status
        check_command           check_snmp!-C lowpowersip -o ifAdminStatus.9 -c 1 -m RFC1213-MIB
        }

# Monitor RAM and CPU
define service{
        use                     generic-service
        host_name               RTA
        service_description     Used RAM in kByte/sec
        check_command           check_snmp!-C lowpowersip -o 1.3.6.1.2.1.25.2.3.1.6.101 -u kb -w 12950 -c 12999
        }
```

```
define service{
        use                     generic-service
        host_name               RTA
        service_description     TOTAL AVL. RAM in kByte/sec
        check_command           check_snmp!-C lowpowersip -o 1.3.6.1.2.1.25.2.3.1.5.101 -u kb
        }

define service{
        use                     generic-service
        host_name               RTA
        service_description     CPU Load in percentage
        check_command           check_snmp!-C lowpowersip -o UCD-SNMP-MIB::laLoadInt.1
        }
# Monitor WiFi associated MAC clients
define service{
        use                     generic-service
        host_name               RTA
        service_description     WiFi associated MAC client
        check_command           check_snmp!-C lowpowersip -o .1.3.6.1.4.1.2021.255.3.54.1.3.32.1.4.1
        }
```

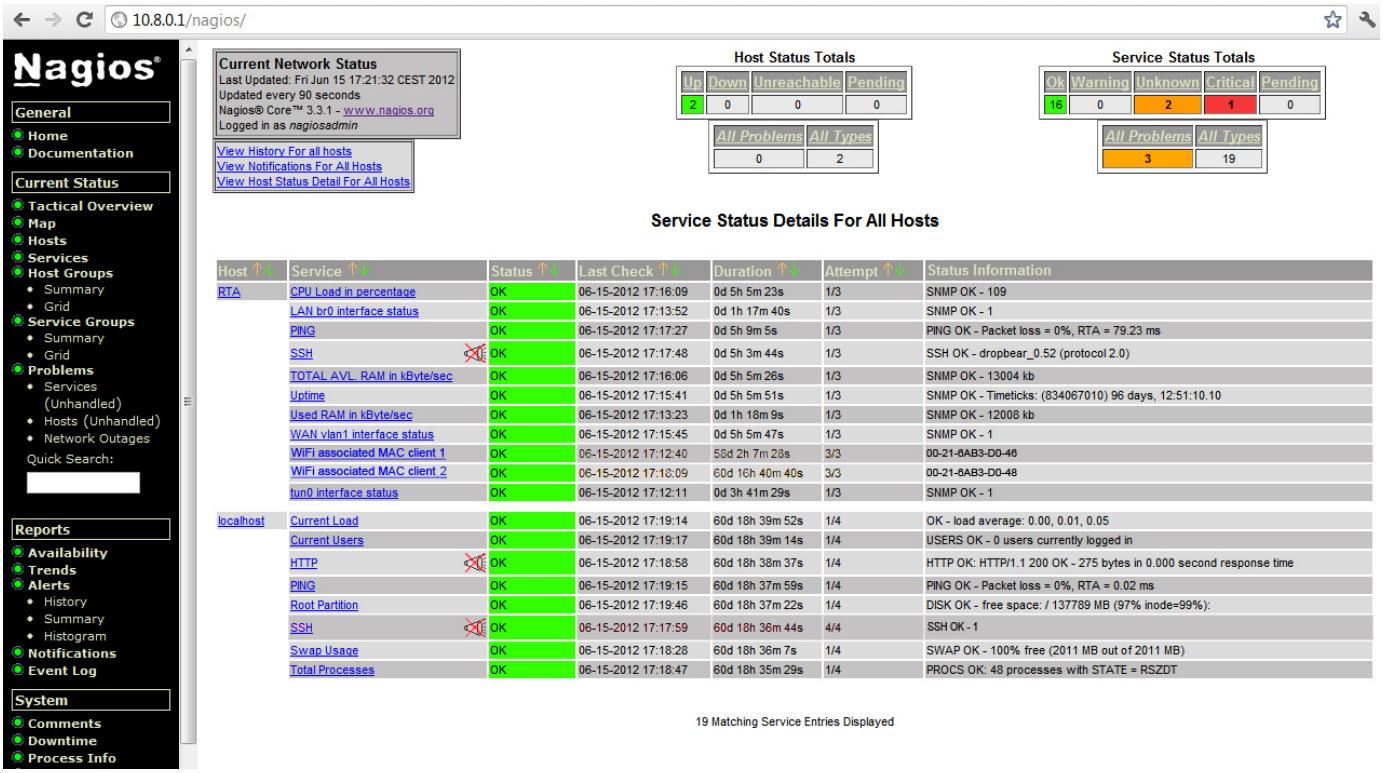


Figure 53: Snapshot of the Nagios GUI showing the operation of the local (RTA=WRT54GL) and the cloud site (localhost=Dell Optiplex 755)

J Open 79XX Directory usage example

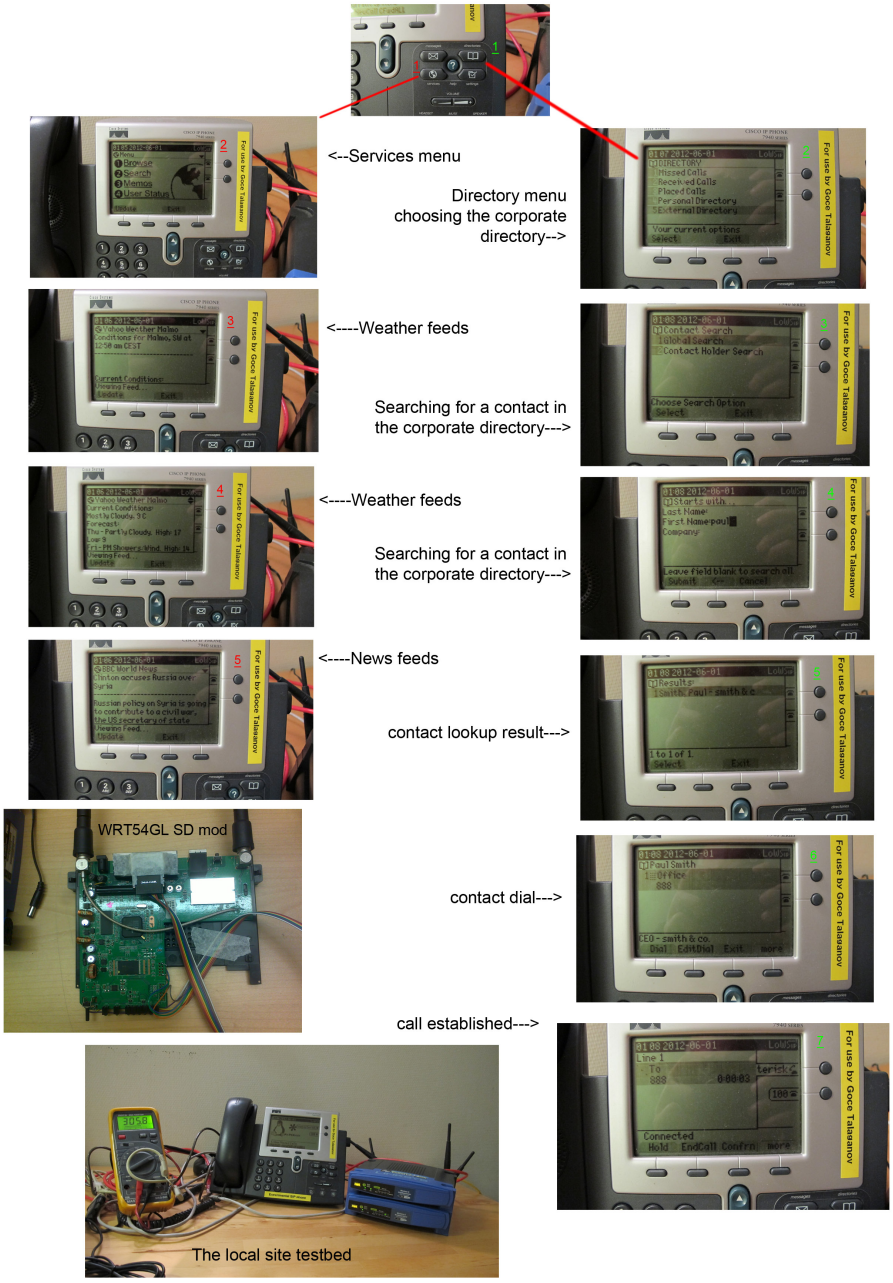


Figure 54: Example for some of the services that Open 79XX xml directory server provides.

TRITA-ICT-EX-2012:162

www.kth.se