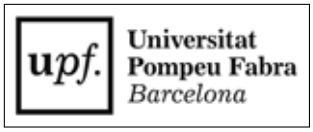
Real-Time Communication Network Architecture Design for Organizations with WebRTC

Pedro Vílchez

TFG UPF / YEAR 2015

DIRECTOR/S OF THE TFG:
Miquel Oliver, Victor Pascual
DEPARTMENT:
Departament de Tecnologies de la Informació i les Comunicacions (DTIC)



Dedicated to my family

Acknowledgments

Special thanks to Victor Pascual and Miquel Oliver for his mentorship. Thanks to Victor Oncins and Angel Elena (craem) for his feedback and help.

Thanks to Daniel Pocock for its work on rtcquickstart.org. Thanks to webrtchacks.com and all its team for the useful articles.

Thanks to Rachel Thalmann for revising English on certain parts of this memory.

Thanks to all the people that works for the democratization of communications.

Thanks for reading. Thanks for your time.

Abstract

The present project introduces disrupting technology WebRTC (Web Real-Time Communication) which supports browser-to-browser applications without the need of third party plug-ins. It details how, since its release by Google in 2011, WebRTC is evolving and changing the way communications are understood. This project also discusses how to set up real-time communications in organizations with WebRTC, specifically the use cases of video and audio calls. The organization under discussion is Guifi.net and the elements analyzed are: requirements, component selection, network architecture design, implementation and demo.

Resum

El projecte introdueix la tecnologia disruptiva WebRTC (comunicació web en temps real), que suporta aplicacions de navegador a navegador sense la necessitat de complements adicionals. Es detalla com, des de que va ser alliberat per Google al 2011, està evolucionant i canviant la forma en que les comunicacions són enteses. Aquest projecte també desenvolupa com posar en marxa comunicacions en temps real per organitzacions amb WebRTC, específicament els casos d'ús de trucades de veu i vídeo. La organització per desenvolupar-ho és Guifi.net i els elements que ho analitzen: requeriments, disseny d'arquitectura de xarxa, selecció de components, implementació i demostració.

Resumen

El proyecto introduce la tecnología disruptiva WebRTC (comunicación web en tiempo real), que soporta aplicaciones de navegador a navegador sin necesidad de complementos adicionales. Se detalla cómo, desde que fue liberado por Google en el 2011, está evolucionando y cambiando la forma en que son entendidas las comunicaciones. Este proyecto también desarrolla cómo poner en marcha las comunicaciones en tiempo real en organizaciones con WebRTC, específicamente los casos de uso de llamadas de voz y vídeo. La organización para desarrollarlo es Guifi.net y los elementos que lo analizan son: requerimientos, diseño de la arquitectura de red, selección de componentes, implementación y demostración.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

A project presentation, motivation, objectives and outline of the chapters.

1.1 Motivation

In my previous Bachelor's thesis [Vílchez, 2014] I developed two ideas: *together we can build a community network* and *it is simple to start*. This project explores how to provide Real-Time Communications (RTC) to a community network such as Guifi.net. Very few large cities have a community network. Barcelona's community network is called Guifi.net. However, the majority of Barcelona's population has no knowledge nor understanding of Guifi.net. Since the communications market is already highly commercial and competitive, users have a very wide choice of providers and therefore no need to think any further. In areas such as smaller cities and towns where large telecommunication companies have not invested in infrastructure, there is a far greater incentive to participate in a community network.

Inside a community network, the implementation of other services besides Internet inherently adds value to the network and could help people understand its advantages. There are services based on contents which are difficult to maintain. But RTC as a service can be easily applied on a large scale. Once RTC has been set up, it is merely a question of time before people start using it. For various reasons a community network generally functions within a narrow radius, and an RTC system could enhance social cohesion within the community.

To apply an RTC system to Guifi.net's community network is a challenge. Attempts to implement an RTC system have failed. WebRTC is a new opportunity to try again.

1.2 Objectives

Study if the following objectives are reachable for Guifi.net:

- Free and secure communication between users via an RTC system and community network infrastructure.
- Backward compatibility with VoIP[1] network. Hence, users can communicate to other VoIP operators from inside and/or outside Guifi.net.
- Designing RTC network architecture to fit the community network scenario.
- Ease of installation and usability of RTC.

1.3 Outline

This document is organized in 5 chapters.

Chapter 1 Introduction introduces the project. The motivation to start this project and the general objectives to achieve.

Chapter 2 Fundamentals of RTC presents the basics of the concepts communications and real-time quality measurements.

Chapter 3 State of the art discusses the state of the art of WebRTC and its associated technologies.

Chapter 4 Methodology contains a general analysis of the topic and a description of the processes involved in order to achieve this project.

Chapter 5 Contributions and Results develops different elements to accomplish RTC in Guifi.net organization: requirements, component selection, network architecture design, implementation and demo.

Chapter 6 Conclusions and Future Work provides a general assessment of the project

[1] Voice over IP. The use of telephone adapted to the Internet network.

Chapter 2

FUNDAMENTALS OF RTC

The generic concepts and infrastructures that make possible real-time communications and important quality parameters to consider.

2.1 Communications

In order to design communications in a system, the different elements must know what to do with the messages. That is why it is needed protocols. A protocol defines the format and the order of messages exchanged between two or more communicating entities, as well as the actions taken on the transmission and/or receipt of a message or other event.

Real-time communications are even more complex, the information has to arrive fast and it is convenient to prepare the receiver of the communication. Signaling is the process of sending control information over networks to monitor, control, route, and set up sessions between devices. These sessions include video and audio conference calls, data sessions, video calls, and mobile and landline telephone calls. Signaling is also used to set up instant messaging and chat sessions. A signaling protocol, is a protocol that needs signaling features.

There are different approaches to achieve real-time systems, that is why it is needed compatibility between the systems. A gateway allows equipment with different protocols to communicate with one another. For example, gateways are used when incompatible video systems are used for a video conference.

It is also important how to transport the data, there are two fundamental approaches: circuit switching and packet switching.

Packet switching and the Internet. The messages are divided in appropriate sized chunks of data known as packets. Packets travel from the source through a network, when they arrive at its destination, the messages are reconstructed. The network makes its best effort to deliver packets in a timely manner, but it does not

make any guarantees. Because the resources of the network are used on demand, not reserved, as a consequence, a packet may have to wait for access a communication link. Each packet could be routed in a different network path depending on the network state. Some of them could be lost if there is a congestion (a lot of traffic, a large number of packets). The network equipment needed to build a packet switching infrastructure is affordable, flexible and can manage easily a large amount of data (throughput). The internet is an example of a network of networks that consists of lots of private, public, academic, business, and government networks of local to global scope. The technical operation and standardization of the Internet is done with the RFCs[1] (Request for Comments) and is an activity of the IETF (Internet Engineering Task Force). The protocol stack of the Internet is the TCP/IP stack, the name was given due to the importance of this two protocols on the system.

Circuit switching and traditional telephony. The resource (a network path composed of communication channels between source and destination) is reserved for the duration of the communication session between the endpoints. In one hand, is a guaranteed constant transmission rate, good link quality, in the other hand, is an expensive resource that is being wasted in silent periods. One of the techniques used to reduce the silent periods is the multiplexing of the circuits in time (Time-Division Multiplexing, TDM) or frequency (Frequency-Division Multiplexing, FDM). The Public Switched Telephone Network (PSTN) is an example of an aggregate network operated by national, regional and local telephony operators that used this type of network paradigm in the past. Now it is moving towards a packet switching network. The technical operation of the PSTN uses the standards created by the ITU (International Telecommunication Union[2]). The protocol stack of PSTN network is the SS7 stack (Signaling System No. 7). Later, it was added the possibility to include generic data to its networks with ISDN (Integrated Services Digital Network).

IETF and ITU did efforts to adapt its networks to the requested uses: transport of generic data and real-time data. Internet is becoming the standard way to transport any kind of data. IETF did operations to include traditional telecommunication operators inside Internet, for example SIGTRAN[3] family of protocols (compatibility with SS7 and ISDN stacks). There is also the Quality of Service (QoS) concept, that gives priority to chosen packets to arrive faster. That is why Internet can manage real-time data with reasonable delay.

SS7, ISDN and TCP/IP stacks are based on the Open Systems Interconnection model (OSI model[4]). Each protocol belongs to one layer. Each layer provides its

[1] In this document each IETF protocol is accompanied by its RFC.

[2] Formerly the International Telegraph Union.

[3] Derived from signaling transport, working group of IETF.

[4] ISO/IEC 7498-1:1994.

service by performing certain actions within that layer and by using the services of the layer directly below it, this is called the service model. Table 2.1 shows the difference between the and OSI model and the TCP/IP stack, each row is a layer. In TCP/IP, physical layer is implemented with hardware, for example with NIC (Network Interface Controller) or WNIC (Wireless Network Interface Controller), generally uses the IEEE 802 family, and its standardization is managed by IEEE (Institute of Electrical and Electronics Engineers.). Link layer is typically implemented by an element called switch, they provide close connectivity. Network layer is typically implemented by an element called router, they provide far connectivity. Session and presentation layers has to be implemented by the developer of the application (↗).

Table 2.1: OSI, TCP layers and its description.

OSI layer	Description	TCP/IP layer
Application	Network and application services	Application
Presentation	Data format	↗
Session	Signaling of data	↗
Transport	Connection establishment side to side	Transport
Network	Logic addressing (far)	Network
Link	Physical addressing (close)	Link
Physical	Binary signal and transmission	Physical

Table 2.2 shows the name of each Protocol Data Unit (PDU), the most important protocols and the organization that standardizes it. At link layer the most common PDU is the Ethernet frame, with its associated MAC address (Media Access Control). At the network layer the datagram, with its associated IP address (Internet Protocol). The version 4 of IP, IPv4 [Postel, 1981a], contains 2^{32} (4.3 billion) addresses, as this is not enough, that is why Internet network is upgrading to IPv6 [Deering and Hinden, 1998] with 2^{128} (more than $7.9 \cdot 10^{28}$ times as many as IPv4). Meanwhile, there are basically two types of IPv4 addresses: private IP [Rekhter et al., 1996], not reachable via Internet, and public IP, reachable via Internet. It is difficult to remember IP addresses to access different locations, that is why there is DNS (Domain Name Server, [Mockapetris, 1987]), it associates human readable addresses (domain) to IPs. An IP address includes a port in the range of 0 to 65535, its main purpose is to share a single physical connection to another place (host), for example an IP destination can has more than one different services available; the different services is usually associated with a specific port. At the transport layer, one unreliable and fast protocol, UDP (User Datagram Protocol, [Postel, 1980]) and another reliable, TCP (Transmission Control Protocol, [Postel, 1981b]). For the real-time topic, UDP is massively used in the real-time

communication, because TCP favors reliability over timeliness. TCP is usually used for the signaling. At the application layer there are many possibilities, an example is the DNS, briefly described above.

Table 2.2: PDU associated with each layer in TCP/IP model

TCP/IP layer	PDU	Standardization
Application	Message	IETF
Transport	Datagram (UDP), Segment (TCP)	IETF
Network	Datagram or Packet (IP)	IETF
Link	Frame (MAC)	IEEE

Every PDU has two types of fields: header and payload. A PDU is encapsulated with the layer below (see Figure 2.1). If the header part is greater than payload part is called overhead. If it occurs, the communication is not very efficient, specially if they are sent lots of packets. Signaling data is overhead in the sense that it is not user data, but is useful to establish a call.

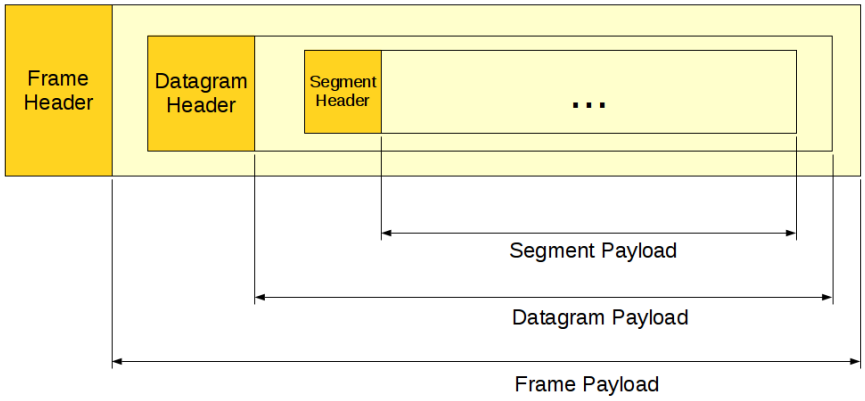


Figure 2.1: PDU encapsulation with IP and TCP protocols, application is not specified.

Before initializing a call is a requirement to convey media details to the participants, SDP (Session Description Protocol, [Handley et al., 2006]) provides a standard representation for such information.

During a call, there is the RTP (Real-time Transport Protocol, [Schulzrinne et al., 2003]). The majority of the RTP implementations are build on top of UDP. RTP is a protocol used to encapsulate multimedia content (audio, video). Its header contains a sequence number and timestamp. The sequence number increments by one for each RTP data packet sent, and may be used by the receiver to detect packet loss and to restore packet sequence. The timestamp registers the time when the RTP

packet has been generated, it helps with the synchronization and jitter[5] calculations. The RTP Control Protocol (RTCP) periodically sends packets with statistics information to participants in a streaming multimedia RTP session. RTP and RTCP provide information on the quality of communication but do nothing to fix or improve it. An application may use this information to control quality of service parameters. RTCP itself does not provide encryption[6] or authentication methods. If they are needed, it is available the SRTP (Secure Real-time Transport Protocol, [Baugher et al., 2004]).

More information about packet switching can be found at [Kurose and Ross, 2013] and about general telecommunications and signaling at [Dodd, 2012]. Some parts of that sources were included in this section.

2.2 Real-time quality parameters

Network is defined by router, switch and link components that interconnect all the path between receiver and its destination (them included). The real-time communication depends on how the network is affected by the following parameters.

The bandwidth is the number of bits[7] that can arrive from the source to the destination through the network in one unit of time (for example, MB/s). A greater bandwidth delivers higher definition (better quality) of the media content. A packet can be lost if it is never received at the destination; it is represented by packet loss percentage (%). A communication suffers jitter when different packets arrive with different delays at its destination. It is defined delay as the time between transmitting the packet and arriving at its destination; it is represented by milliseconds (ms). Due to the nature of packet switching, packets can be delivered out of order. The arrived packet can present a corruption of data. The prior parameters presented affect negatively to the communication. Its undesirable effects might be silent lapses and interruptions of the communication.

Real-time data transport is managed with UDP. This protocol does not perform retransmission, it means that the communication should be fixed for the next upcoming packets. Real-time communications require low bandwidth and delay. This two requirements can be achieved easily with the technologies nowadays.

Some parts of [Velázquez, 2010] and [Frank, 2004] were included in this section. This sources are about voice and its quality, but it can be extended to certain forms of real-time video because of the improved technology nowadays.

[5] Jitter is briefly defined in the next section.

[6] An encryption system guarantees privacy, only authorized participants can read the content.

[7] A bit can be 0 or 1. A stream of this symbols represents information. In telecommunications the information unit is the bit, expressed with the unit b. In computer science the Byte, expressed with the unit B. 1 Byte is 8 bits.

Chapter 3

STATE OF THE ART

Important mechanisms and different technologies that use WebRTC and other associated technologies such as SIP and XMPP

3.1 Internet's evolution

The Internet is in constant evolution, and different mechanisms are necessary for solving problems affecting RTC.

Apart from IPv4 addresses to destination hosts, people can sign up with devices onto host services and be reachable via Internet. A clear example of this is the email system; bob@example.com means that the user, Bob, is registered in the domain example.com that is reachable via Internet. DNS needs an additional parameter to reach Bob; in the case of email the parameter is MX record (Mail eXchanger record, detailed in the SMTP[1] standard [Klensin, 2008]). This is a resource record that specifies a mail server responsible for accepting email messages on behalf of a recipient's domain. To advertise other network services such as SIP and XMPP, the SRV record (SeRVice record, [Gulbrandsen et al., 2000]) is necessary. The NAPTR record (Name Authority PoinTeR, [Mealling, 2002]) allows a simple contact address such as bob@example.com to contain the available forms of contact and its priority to different services.

In the current upgrade from IPv4 to IPv6, NAT (Network Address Translation, [Senie, 2002]) has been standardized in order to alleviate the scarcity and depletion of IPv4 addresses. NAT allows a public IP address to be reused among many different private networks (and its private IP addresses). It functions by connecting private local IP addresses and port tuples to one or more public IP addresses and port tuples. Unfortunately this introduces connectivity problems,

[1] Simple Mail Transfer Protocol, defines the electronic mail (e-mail) transmission.

especially in peer-to-peer connections[2] (p2p). There are two server utilities which can be configured into a device to solve these problems. Both are required to be on the public network. A STUN server (Session Traversal Utilities for NAT, [Rosenberg et al., 2008]) allows a device to find out its public IP address if the device is located behind a NAT. STUN service keeps the NAT binding alive. A TURN server (Traversal Using Relays around NAT, [Mahy et al., 2010]) acts as a transport address intermediary between the two people wishing to communicate. Unlike STUN, TURN is resource-intensive for the provider. The problem with these two techniques is that they are optimal in some network topologies but a poor choice in others. That is why the most interesting solution for NAT traversal is ICE (Interactive Connectivity Establishment, [Rosenberg, 2010]). ICE exchanges, using the offer/answer SDP model, a multiplicity of IP addresses and ports, that includes STUN and TURN, which are then tested for connectivity by peer-to-peer connectivity checks. The best candidate transport address is then selected.

The World Wide Web[3], the Web, is the most common way to use the Internet. It has a client-server distributed architecture model[4]. Web servers act as providers of resources or services and web browsers as client requesting. It means that it follows a request-response communication model: client requests and server responses. The browsers navigates between different web pages provided by the web servers. The web navigation could be unencrypted with HTTP[5] (HyperText Transfer Protocol, [Fielding et al., 1999]) requests or encrypted with HTTPS (HTTP Secure, [Rescorla, 2000]). The success of the Web is promoting the use of web applications for all possible services. Two important features of web applications are that they are cross-platform, the browser is a default application in most of the operating systems, and easy to update or upgrade, the browsers only has to refresh the page to obtain a new version. An important problem of the request-response communication model was its complexity associated to the use of bidirectional communication or streaming [Loreto et al., 2011], for example to call through a browser. The solution is provided below, the Websocket protocol.

As Internet is such a vast, open and widely distributed network, it is therefore also open to undesirable consequences. For example, devices reachable via Internet are constantly attacked by other devices trying to exploit its weaknesses in computer systems and computer networks and thus can affect the security of an organization. For this and other reasons, its administrators deny by default any service which is not explicitly important to the organization. One of the compo-

[2] A connection that goes directly from a host to another host, without an intermediate server.

[3] An information system for interlinked hypertext documents and other digital resources that are accessed via the Internet.

[4] `http://www.w3.org/Proposal.html`.

[5] HTTP is a coordinated effort by the IETF and the World Wide Web Consortium (W3C).

nents used to control the incoming and outgoing network traffic on an applied rule set is the firewall. But the firewall itself can also have undesirable consequences; in large organizations it can be less flexible. Take as an example a university with students interested in computers who are experimenting with services related to courses or projects; they may be having connectivity problems due to the restrictive nature of the firewall.

Another common mechanism to control the networks is the proxy. A proxy is a network entity that acts as an intermediary between two communication entities. A web proxy satisfies HTTP requests on the behalf of an origin web server. A proxy has the ability to inspect traffic flowing through the service, and can ensure no traffic flows through to the Internet without inspection or control.

Firewalls and web proxies commonly admit traffic via ports 80 (HTTP) and 443 (HTTPS). The Websocket protocol (WS, and WSS for Websocket Secured, [Fette and Melnikov, 2011]) introduces bidirectional communication between a browser and a web server via ports 80, 443 or any other specified port. This provides websocket-based services an ability to traverse firewalls and proxies, and also prepares the way for a new generation of bidirectional applications, such as RTC.

For an organization, many different applications and services imply complexity in fields such as usability, account management, security. One account per user is enough; LDAP (Lightweight Directory Access Protocol, [Sermersheim, 2006]) and OAuth (OAuth 2.0 Authorization Framework, [Hardt, 2012]) are two appropriate authentication services which facilitates access and management of applications in an organization. They can provide the service in centralized or hierarchically distributed manner.

3.2 Voice, video calls and instant messaging

There are two strong open standard alternatives to supply the use cases of voice, video calls and instant messaging: SIP and XMPP.

SIP (Session Initiation Protocol, [Rosenberg et al., 2002]) is a protocol used to establish, modify and terminate multimedia sessions in the Internet. It is one of the VoIP[6] available services, an alternative to H.323 from the ITU. It is text-based, on a request/response transaction model from HTTP and SMTP; for example encrypted SIP is SIPS, and it has similarities to the relation between HTTP and HTTPS. SIP itself provides a TCP or UDP signaling channel, SDP to agree a media session and a RTP media channel. SIMPLE[7] is a concluded Working Group

[6] Voice over IP, allows telephone calls via Internet.

[7] SIP for Instant Messaging and Presence Leveraging Extensions `https://tools.ietf.org/wg/simple/`.

at IETF that extended SIP with messaging and presence, this source offers an overview of the work done [Rosenberg, 2013]. Important telecommunication enterprises provides SIP trunking[8] such as Telefónica[9] and AT&T[10]. SIP have hardware client implementations called SIP-based VoIP phones, and software client implementations called softphones. Old telephones can have a SIP conversion with ATA (Analog Telephone Adapter).

XMPP[11] (Extensible Messaging and Presence Protocol), is a protocol to manage instant messaging and presence of your contact list, called roster in its terminology. It is defined in the IETF [Saint-Andre, 2011] and the XEPs (XMPP Extension Protocols) are worked by the XMPP Standards Foundation[12] (XSF). It is one of the IM (Instant Messaging) applications available. It encrypts its traffic with TLS such as SIP and HTTP. It is text-based streaming XML (Extensible Markup Language) data in close to real-time. XMPP itself provides a signaling channel, typically via the TCP. Jingle [Ludwig et al., 2009] is a signaling protocol that extends XMPP with audio and video calls and is designed to interwork with SIP through gateways. Massively used services provides XMPP-based solutions such as WhatsApp Messenger with 800 million monthly active users[13]. The XMPP clients usually are implemented as a software device.

Because of firewalls and proxies, clients usually have connectivity problems connecting to SIP (port 5060) and XMPP services (port 5222). XMPP started using BOSH (Bidirectional-streams Over Synchronous HTTP, [Paterson et al., 2014]), this had the problems of using HTTP for bidirectional communication. Now, with the standardization of Websocket protocol, there are SIP over websocket [Castillo et al., 2014] and XMPP over websocket [Stout et al., 2014].

3.3 WebRTC

WebRTC is the technology able to make calls through the browser and real-time communications in general. In this section are presented the two organizations working on WebRTC standardization, IETF and W3C. The different concepts laid out are inspired by the IETF Internet-Draft (IETF-ID) *Overview: Real Time Protocols for Browser-based Applications*[14].

[8] SIP trunking means interconnection to an operator through SIP.

[9] `https://www.globalsolutions.telefonica.com/en/wholesale/products-services/global-voice/sip-trunking/`.

[10] `http://www.business.att.com/enterprise/Service/voice-services/null/sip-trunking`.

[11] Formerly the protocol was originally named Jabber.

[12] Formerly the Jabber Software Foundation.

[13] `https://www.facebook.com/jan.koum/posts/10153230480220011`.

[14] `https://tools.ietf.org/html/draft-ietf-rtcweb-overview-13`.

WebRTC is a browser-embedded media engine released by Google in 2011. A media engine packages multimedia processing components into an optimized software solution for smoother integration and enhanced performance. WebRTC is also an effort by IETF and W3C to add standardized RTC capabilities into browsers through APIs. IETF RTCWEB[15] Working Group is producing architecture and requirements for selection and profiling of the on-the-wire protocols. W3C WEBRTC[16] Working Group is defining browser APIs to enable real-time communications in web browsers. IETF and W3Cs' work on WebRTC is still in progress. Figure 3.1 shows the parts played by the IETF and W3C in the browser model. The source[17] is Victor Pascual's presentation of WebRTC.

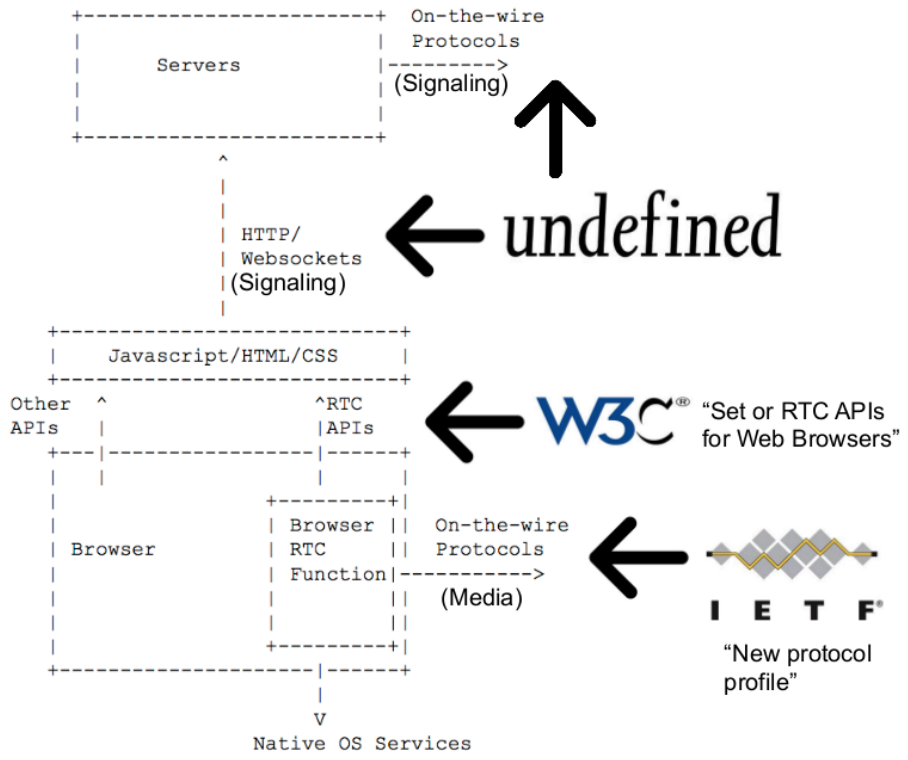


Figure 3.1: WebRTC browser model

The WebRTC trapezoid architecture model (see figure 3.2) shows the elements that make possible the communication between browsers. JS/HTML/CSS are the

[15] https://tools.ietf.org/wg/rtcweb/.

[16] http://www.w3.org/2011/04/webrtc/.

[17] http://www.slideshare.net/victorpascual/webrtc-standards-update-october-2014/4.

components to do a web application, it includes the WebRTC W3C APIs. It is required bidirectional communication channel between the web browser and the web server, it can be used an application-defined over Websocket Protocol such as: XMPP or SIP. The signaling path is composed by the communication channels used between entities participating in signaling. Once signaling is transferred, it starts a media path, a communication channel where multimedia content follows from one WebRTC device to another.

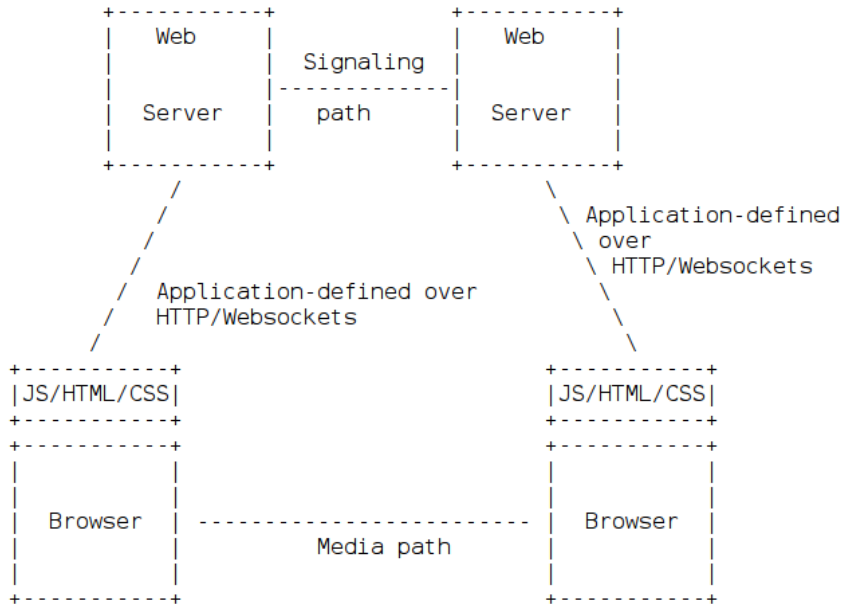


Figure 3.2: WebRTC trapezoid architecture model. Source: IETF WebRTC Overview

Data transport is defined in IETF-ID *Transports for WebRTC*[18]. Generally, it is TCP for signaling channel and UDP for media channel. The WebRTC implementation could support accessing the Internet through an HTTP proxy with a *connect* header as specified in IETF-ID *The ALPN HTTP Header Field*[19]. It refers to IETF-ID: *DSCP and other packet markings for RTCWeb QoS*[20], about the usage of QoS with DSCP (Differentiated Services Code Points, [Nichols et al., 1998]).

Data framing is defined in *Web Real-Time Communication (WebRTC): Media Transport and Use of RTP*[21]. Data framing is RTP traffic. RTP and RTCP must be

[18] https://tools.ietf.org/html/draft-ietf-rtcweb-transports-08.

[19] http://tools.ietf.org/html/draft-ietf-httpbis-tunnel-protocol-04.

[20] https://tools.ietf.org/html/draft-ietf-tsvwg-rtcweb-qos-03.

[21] https://tools.ietf.org/html/draft-ietf-rtcweb-rtp-usage-23.

multiplexed on a single transport-layer flow. It has to support the use of multiple media streams (simultaneous SSRC values) in a single RTP session, more information at IETF-ID *Sending Multiple Media Streams in a Single RTP Session*[22]. WebRTC uses as RTP Profile the Extended Secure RTP Profile for RTCP-Based Feedback (RTP/SAVPF, [Ott and Carrara, 2008])

The complete specification of RTP for a particular application domain requires the choice of an RTP Profile. For WebRTC use, the For non-RTP data, IETF-ID *WebRTC Data Channels*[23] defines generic transport service that allows browsers to exchange generic data from peer to peer. Additionally, the document discusses associated use cases and requirements. Data channels use SCTP (Stream Control Transmission Protocol, [Stewart, 2007]). It is a protocol that provides reliable or partially reliable message transport. A SCTP data channel has two streams with the same Stream Identifier, one in each direction, which are managed together. Stream Identifier uniquely identifies a stream. IETF-ID *WebRTC Data Channel Establishment Protocol*[24] (DCEP) is a protocol that establishes bidirectional data channels over an SCTP association with a consistent set of properties about how reliable it is wanted the transmission.

Securing. IETF-ID *Security Considerations for WebRTC*[25] analyzes the security threats of WebRTC. IP Location Privacy (section 5.4) is a threat that concerns the privacy-oriented usage of VPN tunnels to be anonymous in Internet. VPN (Virtual Private Network, [Andersson and Madsen, 2005]), extends a private network (with private IPs) across a public network (with public IPs) through its. This security threat is caused by the ICE procedure because it reveals a lot of information about the callee's location; that is why it should be prevented in undesirable cases. Daniel Roesler provides a demo of this security threat[26]. Cullen Jennings in a presentation[27] said that it is only affected by split VPN tunnels, because it reveals both external interfaces.
IETF-ID *WebRTC Security Architecture*[28] defines a security architecture for WebRTC that takes in account the security threats of the other document. IP location privacy is out of scope. It defines the DTLS (Datagram Transport Layer

[22] https://tools.ietf.org/html/draft-ietf-avtcore-rtp-multi-stream-07.
[23] https://tools.ietf.org/html/draft-ietf-rtcweb-data-channel-13.
[24] https://tools.ietf.org/html/draft-ietf-rtcweb-data-protocol-09.
[25] https://tools.ietf.org/html/draft-ietf-rtcweb-security-09.
[26] https://diafygi.github.io/webrtc-ips/.
[27] https://www.terena.org/activities/tf-webrtc/meeting2/slides/20150509-Cisco-WebRTC.pdf.
[28] https://tools.ietf.org/html/draft-ietf-rtcweb-security-arch-11.

Security, [Rescorla and Modadugu, 2006]) handshake[29]. It is a SCTP over DTLS (*DTLS Encapsulation of SCTP Packets*[30]) for the data channel and a DTLS-SRTP [Fischl et al., 2010] for the media channel. The media channel or the data channel cannot be unencrypted. Once the DTLS handshake has completed, the keys are exported [Rescorla, 2010] and used to key SRTP for the media channels. DTLS fingerprints could be self-signed but it avoids a man-in-the-middle (MitM) attack if participants trust in a third party identity service or certificate authority.

Data formats. IETF-ID *WebRTC Audio Codec and Processing Requirements*[31] proposes as required to implement the audio codecs G.711[32], Opus [Valin et al., 2012]. IETF-ID *WebRTC Video Processing and Codec Requirements*[33] proposes as required to implement the video codecs: H.264[34], VP8 [Bankoski et al., 2011].

Connection management is defined in *Javascript Session Establishment Protocol*[35] (JSEP). It describes the mechanisms to control the signaling of a multimedia session and its relation with the W3C RTCPeerConnection API. To reduce its complexity it is recommended to adapt the JSEP API into an application-defined over Websocket Protocol library[36], for example with XMPP or SIP. JSEP controls the signaling, particularly: (1) local and remote session descriptions with SDP and (2) participant's network addresses and ports available for the exchanging of real-time data (ICE). A simple example audio/video call is available at the Section 7 of this JSEP draft.

Presentation and control. W3C APIs define the user interaction with the browser. W3C Working Draft *WebRTC 1.0: Real-time Communication Between Browsers*[37] defines important API interfaces for WebRTC, the most relevant: (1) RTCPeerConnection in section 4 *Peer-to-peer connections* allows browsers to communicate directly, (2) RTCSessionDescription (JSEP) in section 4.7 *Session Description Model*, (3) RTCDatachannel (data channel) in section 5 *Peer-to-peer Data API* and (4) RTCStats in section 7 *Statistics Model* is an evaluation tool of the communication.
W3C Working Draft *Media Capture and Streams*[38] defines the Media Capture API. It facilitates to the browser the management of local media devices such as audio and video.

[29] A handshake determines how the communication of a protocol starts..
[30] `https://tools.ietf.org/html/draft-ietf-tsvwg-sctp-dtls-encaps-09`.
[31] `https://tools.ietf.org/html/draft-ietf-rtcweb-audio-07`.
[32] `https://www.itu.int/rec/T-REC-G.711-198811-I/en`.
[33] `https://tools.ietf.org/html/draft-ietf-rtcweb-video-05`.
[34] `https://www.itu.int/rec/T-REC-H.264-201402-I/en`.
[35] `https://tools.ietf.org/html/draft-ietf-rtcweb-jsep-09`.
[36] A list with examples at `https://webrtchacks.com/vendor-directory/`.
[37] `http://www.w3.org/TR/2015/WD-webrtc-20150210/`.
[38] `http://www.w3.org/TR/2015/WD-mediacapture-streams-20150414/`.

Local system support functions improve the user experience. Examples of local functions are echo cancellation, volume control, camera focus, zoom, etc. Some of these functions are defined in Audio and Video Codec documents of IETF-ID and MediaStream API of W3C presented before.

The book [Loreto and Romano, 2014] was consulted as an additional source of information about WebRTC.

Chapter 4

METHODOLOGY

Analysis of the selected topic for this project, its scope, resources associated with the project, its planning, tasks and work style.

4.1 SWOT analysis of WebRTC

An analysis of Strengths, Weaknesses, Opportunities and Threats (SWOT) will help the decision-making and tasks for the project.

- Strengths
  - Ease of use: real-time communication is supported without the need for additional applications or plug-ins.
  - It helps to solve connectivity problems caused by NAT, Firewall, etc.
  - Solved the problem of selection of video and audio codecs.
  - It is based on open standards, open source software implementations and has a royalty free patent[1].
  - It has well general acceptance in both worlds: enterprise and community.
  - The communication between peers is bidirectional and can be P2P
  - WebRTC standard does not specify signaling: it can be used in very different scenarios.
  - The communication channel between peers is encrypted
- Weaknesses

[1] `http://www.webrtc.org/faq`.

  - It is not implemented in all browsers.
  - The different browsers that implement WebRTC could have incompatibilities.
  - WebRTC has incompatibility at transport level with SIP, a gateway is needed.
  - Security compromised when using split VPN-tunnels, the IP address before the VPN-tunnel is exposed.

- Opportunities
  - WebRTC can be used in web based softphone for VoIP. Easy to install, easy to update.
  - A WebRTC audio call could be routed to traditional telephony.
  - It uses javascript as programming language, this language has the widest developer's community.
  - It encourages a new generation of web applications using its strengths. For example, WebRTC can easily implement through a web page a *click to call button* or a phone box[2] (call without account).
  - Google bought a relevant enterprise focused on media engines called GIPS (Global IP Solutions) and released it as WebRTC. Not only is becoming a standard in terms of Internet (IETF) and Web (W3C) technologies; but also a *de facto* standard in the industry because of the media engine's quality. This source[3] analyzed media engine market consequences after WebRTC release, the result is that some of the companies are starting to use WebRTC. WebRTC reduces RTC systems complexity, that is why new companies entered market of media engine and are starting to deploy its own solutions.

- Threats
  - WebRTC standard do not specify signaling: this can produce a positive or negative fragmentation of projects. Positive fragmentation: different projects for different applications. Negative fragmentation: divided effort.
  - Is a work in progress technology, it is being changed.

[2] `https://freephonebox.net/`
[3] `https://bloggeek.me/webrtc-media-engine`.

4.2 Scope

There are lots of RTC systems for different purposes. This project focuses in the work of IETF organization and Internet. Guifi.net is part of Internet, and has additional constraints to take in account.

The following topics are worked only in its fundamentals: RTC standard systems from ITU[3], referred to in the 2, and XMPP, referred to in the 3.

In general it has chosen technologies based on open standards, open source implementations and royalty free patent.

4.3 Resources

There are costs related to the activity of this project in terms of equipment and human effort.

Table 4.1 shows the equipment resources and its economic estimation. Observations:

- Guifi.net connectivity to Barcelona, a reachable IPv4 10.0.0.0/8[4] has not direct cost.
- Nearly all software involved is open source and has no direct cost.
- Usually the cost of installation it's greater or equal than the cost of equipment.

Table 4.1: Equipment resources

Material	Estimated cost (euro)
Guifi.net equipments in my home	200
PC with virtualization capabilities (home)	1000
Guifi.net equipments in university	1000
PC with Internet public IPv4 (university)	300
Total	2500

The human effort part was financed by the university in the form of a grant to the author, representing a cost of 2800 euro. A bachelor's thesis corresponds in Europe to 500 hours of work.

This implies a total cost of approximately 6000 euro

[4]Range of IP's used by Guifi.net and private networks.

4.4 Planning

The project can be separated in two phases. The first phase is a long preamble of studying VoIP and WebRTC. The second phase is an agile plan. Figure 4.1 shows the two phases in a gantt chart.

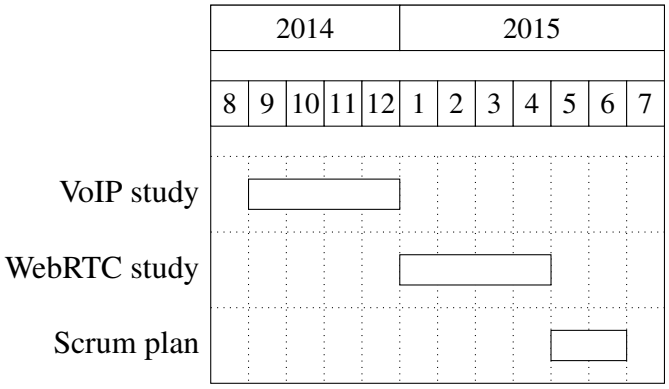


Figure 4.1: General gantt chart

In the first phase, while studying VoIP the intention was to work about VoIP and Guifi.net. But Miquel Oliver encouraged me to do it about WebRTC. He presented me Victor Pascual, a SIP and WebRTC expert. It was hard to realize a convenient project, because this technology involves lots of protocols, other technologies, and it's being modified now. In this phase It were settled the necessary concepts to start the project.

The second phase is an agile plan, inspired by the Scrum methodology. Scrum is one of the Agile methods[5] used for software development. The important fact is that promotes adaptive planning and flexible response to change. Scrum, particularly, is a general method that should be adapted to a concrete scenario.

The Scrum Team consists of a Product Owner, the Development Team, and a Scrum Master. The work of the scrum team according to the Scrum Guide[6] is *deliver products iteratively and incrementally, maximizing opportunities for feedback. Incremental deliveries of "Done" product ensure a potentially useful version of working product is always available*. The roles are

- Product Owner: *is responsible for maximizing the value of the product and the work of the Development Team*

[5]There are different metodologies grouped into agile. The process started with the write of the Agile Manifesto (12 principles) `http://agilemanifesto.org/iso/en/principles.html`. Since February 2001, this manifesto remains unchanged.

[6]`http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf`.

- Development Team: *consists of professionals who do the work of delivering a potentially releasable Increment of "Done" product at the end of each Sprint*

- Scrum Master: *is responsible for ensuring Scrum is understood and enacted*

*The heart of Scrum is a **Sprint**, a time-box of one month or less during which a "Done", usable, and potentially releasable product Increment is created*

4.4.1 Scrum plan

It is necessary to adapt the different concepts that comprise the scrum methodology for this particular project.

Roles:

- Product Owner (in some way, stakeholders): Mentors, University, people interested in the project. The author is interested in the output of the project because is volunteer in Guifi.net.

- Development Team: assumed by the author

- Scrum Master: assumed by the author, optionally could be assumed by mentors.

This means that the author has to see the project with different points of view.

The Sprint time is approximately one week, because it is assumed that the minimum time-box possible to do a release of the product is one week. The product comprise two major tasks: the theory (documentation, memory) and practice (how this theory is fitted to the real world experiments). The tasks are explained with more detail in the next section.

Figure 4.2 shows the Scrum plan with the different sprint phases (s1, s2, s3, s4) and important milestones:

- d1: project charter and tasks, delivery to mentors

- d2: first consistent draft memory, delivery to mentors

- d3: set title and abstract to the thesis, delivery to university

- d4: thesis, delivery to assigned tribunal

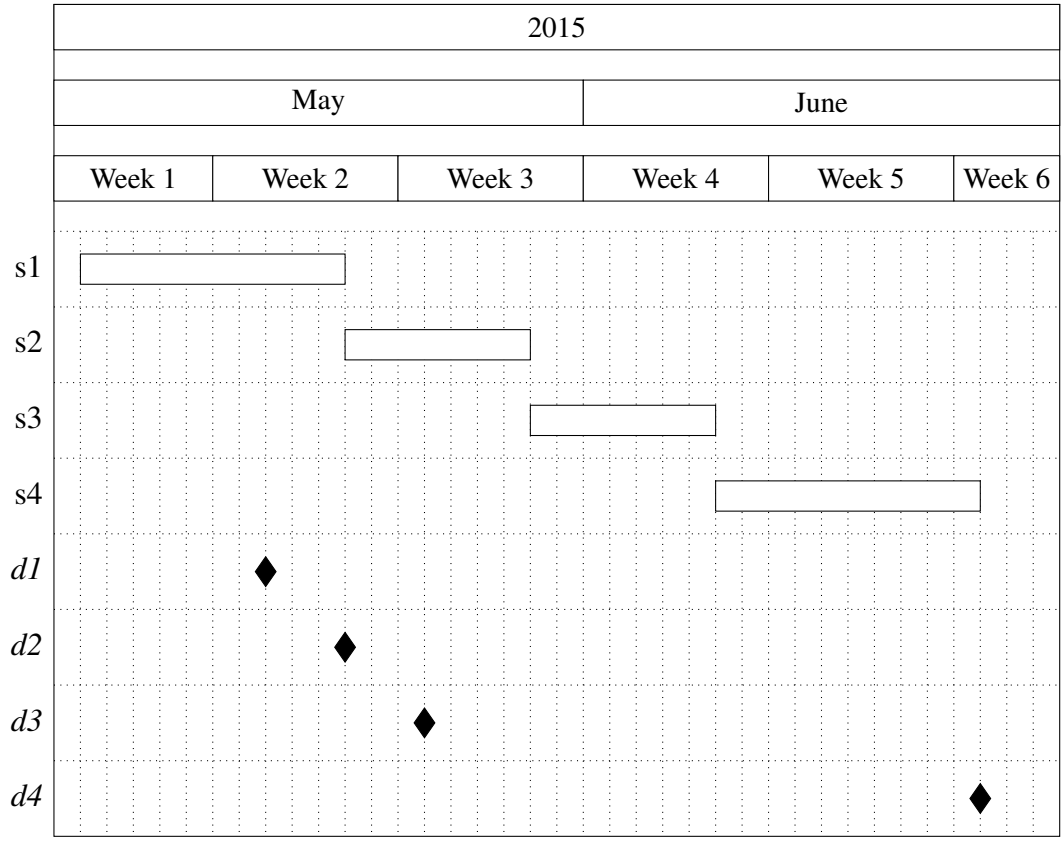


Figure 4.2: Scrum plan gantt chart

4.4.2 Metatools

To ensure the scrum plan and the project, different tools were used:

- Emacs orgmode: is a plain text syntax and software that facilitates different operations
  - nested concepts: it is possible to fold and unfold nested concepts different parts. This brings facilities to take different points of view of the project.
  - write the memory of the project and export to UPF publication constraints.
  - diary: used as self evaluation tool. Time spent in some operations. Place to record when was discovered something.
  - tasks: to write things to do and mark them as TODO and DONE. To see overall progress of the project.

- Git: is a distributed version control system that helps to ensure the work is not lost. It can has a local and remote copy of all different states (commits) of the project. It is very flexible to do changes and apply.

- Github[7] repository: is a social network that uses git and has the largest community. A place to host and share open source projects. This project is hosted as a repository in `https://github.com/pedro-nonfree/guifi-webrtc`. Featured files:

  - diary.org: record of activity in time
  - tasks.org: parts to do for the project
  - doc directory: independent parts written before starting the memory, or that needs isolation
    - doc/index.org: organize the different files of this directory
  - latexbuild directory: place where emacs orgmode thesis file is exported to latex and compiled to PDF
    - thesis.org: source code of memory
    - thesis.pdf: memory

4.5 Tasks and work style

The tasks for this project are divided in two components: theory and practice.

Inside **theory**, there is:

- Documentation

  - Things to say/explain: what should be said that at the moment is missing (checklist)
  - Parts to fill: developed parts that are missing few details (checklist)
  - Parts to fix: developed parts that are incorrect and should be fixed (checklist)
  - Questions: related to the writing or the theory part, that it is needed an answer (checklist, done when answered)
  - Review: concepts that should be reviewed again, after a scheduled date (checklist)

[7] The web implementation is proprietary software, but it can be easily migrated to other open source tools such as `http://gitlab.com` or `http://gogs.io/`.

  - Memory document has tools to track the state of different sections. For theory, it will be specially important:
    * Fundamentals
    * State of the Art
    * Result and Contributions

- Search of information
  - What things should be read (checklist).

Inside **practice**, there is components' testing. The task is to configure and execute the proposed network architecture design.

Figure 4.3 shows the **work style**, how the objectives will be accomplished and its quality. Stable means that it should be clear and complete its content; best effort that it will work in the best way possible but with less priority:

- Requirements: use cases, constraints needed for the chosen organization. (quality: stable)
- Design: architecture design that fits the requirements. (quality: stable)
- Implementation: component selection, protocols (quality: best effort)
- PoC (Proof Of Concept): applications that shows some of the results (quality: best effort)

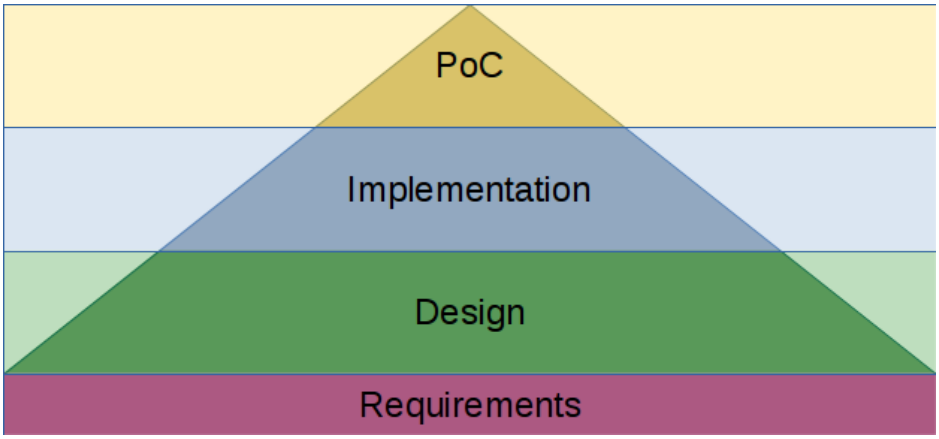


Figure 4.3: Work style diagram

Chapter 5

CONTRIBUTIONS AND RESULTS

Network architecture design and analysis of how an RTC system is adapted to Guifi.net

5.1 Guifi.net architecture

Guifi.net is a commons telecommunication network started in 2004 that is developed with an agreement that guarantees that the network is freely accessible, open to participation, open to knowledge and neutral in data transport. The network interconnects the people that takes part in it (the community) and facilitates the access to Internet. The Guifi.net foundation gives legal support to the agreement and the network.

The community is all the people that have access to the Guifi.net network; it is segmented in zones. Each zone is a geographic place where there are different organizations. As individuals there are volunteers, which maintain the network without SLA (Service-Level Agreement) and freelance professionals, with SLA. As groups there are non-SLA groups, usually non-profit associations of volunteers and professionals, and SLA groups, companies.

Guifi.net has statistics and control of all its devices with an open source web application[1] based on drupal[2] v6. Figure 5.1 shows statistics of a device in terms of availability, delay and incoming and outgoing traffic. The web application connects to the specific zone server which has a software called snpservices[3] (Spontaneous Networking Platform) whose traffic is SNMP (Simple Network Management Protocol, [Harrington et al., 2002]). The web application also controls the network: each node could has a unique IPv4. Therefore, it is easy to determine a

[1] `https://guifi.net`.
[2] `https://drupal.org`.
[3] `https://github.com/guifi/snpservices`.

potential attack; moreover aggressor and victim share the same legal framework. Hence, Guifi.net has better security than Internet.

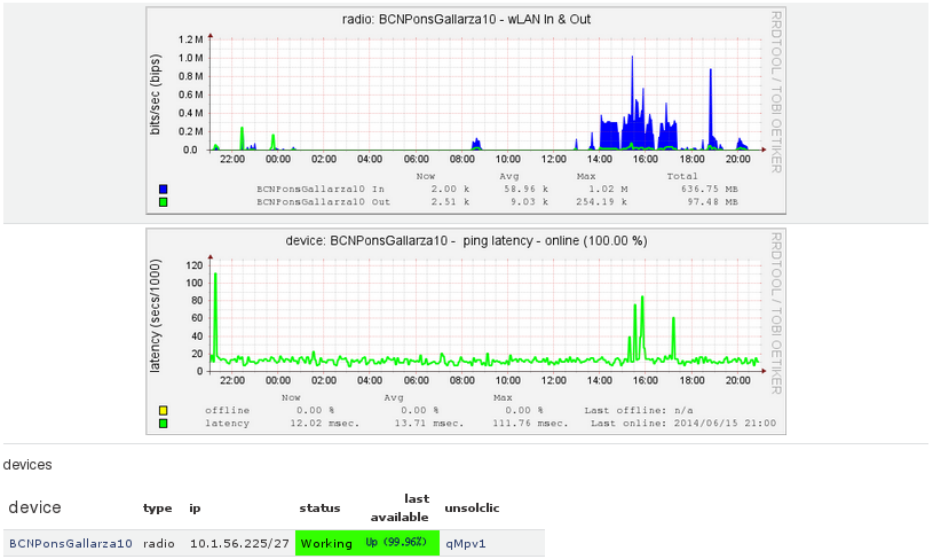


Figure 5.1: Statistics and control of a Guifi.net's device

Guifi.net is an heterogeneous network which has different technologies. There are mainly two routing protocol models: traditional and mesh. Traditional model is used since Guifi.net's start, it is similar to models of organizations such as Internet, which means that is ready for production environments. Figure 5.2 shows where is Guifi.net, the majority of zones uses this model. Its design has two components: supernode, which extends the network and together conform the backbone network, and a node or client node, that connects to a supernode. Its routing devices usually have factory's firmware such as AirOS from Ubiquiti[4]) and RouterOS from Mikrotik[5], both proprietary. Routing protocol implementation in traditional model has two variations: BGP (Border Gateway Protocol, [Lougheed and Rekhter, 1991]), where each supernode is an AS (Autonomous System[6]), or BGP for messages that goes inside or outside the AS, OSPF (Open Shortest Path First, [Moy, 1998]) inside the AS. Mesh model is a more experimental and modern approach based on autodiscovering neighbors. Some Barcelona's zones are using this model. Its design has one component, a node, that extends the network. Its routing devices have open source firmwares based on OpenWrt[7] such

[4] `https://www.ubnt.com/`.

[5] `http://www.mikrotik.com/`.

[6] single administrative entity with a specific routing policy which anounces at least one network prefix (a set of IPs).

[7] `https://openwrt.org/`.

as qMp[8] and libremesh[9]. There are two routing protocol implementations available in mesh model. Bmx6[10] works on network layer; it is available in qMp or libremesh firmware. Batman-adv[11] works on link layer with linux kernel tools; it is available in libremesh firmware. Table 5.1 summarizes the differences between the two models presented, and provides extra information.

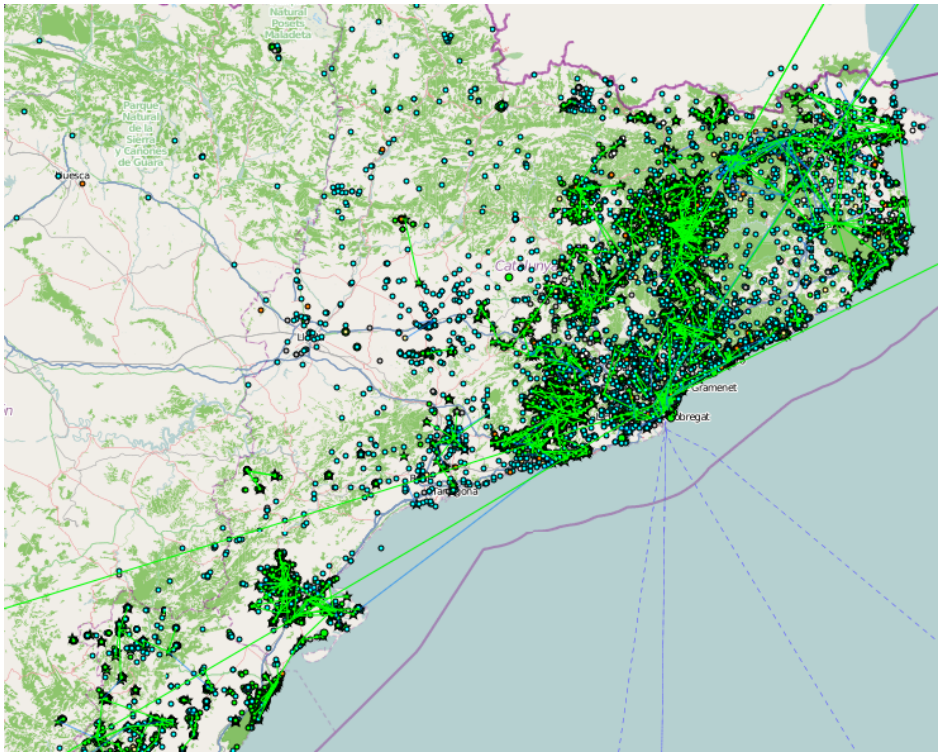


Figure 5.2: Map of Guifi.net's coverage

The Guifi.net's network components are switchs, routers, servers and links. In traditional model supernodes have as much redundancy links as possible and this generates a mesh topology called backbone network, client nodes connect to existing supernodes, this generates a star topology called access network. That is why the relationships between components in traditional model are similar to Internet; but Internet has a hierarchical backbone network (see figure 5.3, source wikipedia[12]). The relationships between components in mesh model tend to a

[8] https://qmp.cat.

[9] http://libre-mesh.org/.

[10] http://bmx6.net/projects/bmx6.

[11] http://www.open-mesh.org/projects/batman-adv/wiki.

[12] https://en.wikipedia.org/wiki/File:Internet_Connectivity_Distribution_%26_Core.svg.

Table 5.1: Traditional Model vs Mesh model

	Traditional model	Mesh model
Deployed IP version	IPv4	IPv4, IPv6
Firmware license	Mostly private	Free open source
Routing protocol	BGP, OSPF	bmx6, batman-adv
Average deployment time	Longer	Shorter
Average cost per link	More	Less
Common operations	Technical level	User level
Uncommon operations	Technical level	Technical and Programming level

mesh network topology. Figure 5.4 shows real network topologies deployments, left image is from wikipedia[13].

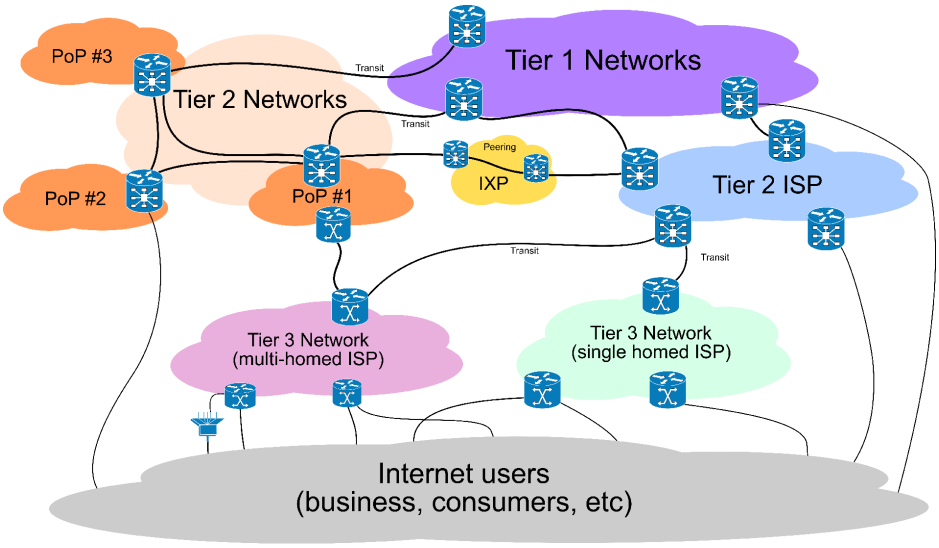


Figure 5.3: Internet relationships. Backbone network: Tiers 1, 2, 3. Access network: Internet users.

The vast majority of communication links are wireless, but there are some places where Guifi.net has deployed optical fiber. Also, in large cities such as Barcelona there are GRE-based (Generic Routing Encapsulation, [Farinacci et al., 2000]) tunnels to improve connectivity using optical fiber services of ISP (Internet Service Provider). Table 5.2 shows features of 802.11 Wi-Fi technologies. There are remaining a, b, g 802.11 protocols from old links that still work, but most are 802.11n. Mesh model only uses 802.11n. Recently, 802.11ac is being tested

[13] `https://en.wikipedia.org/wiki/File:Internet_map_1024.jpg`.

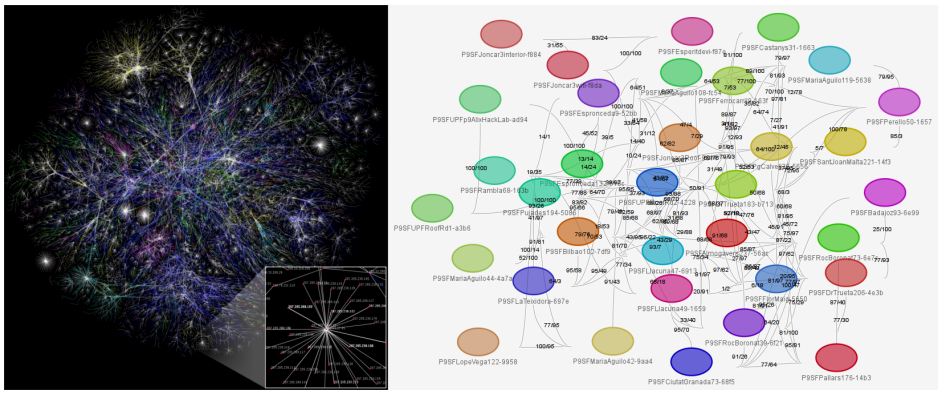


Figure 5.4: Left: Traditional model topology is similar to Internet's topology. Right: Mesh model topology, graph of a Guifi.net neighbor's network of Barcelona.

in some point to point links of traditional model; but at the moment is not possible in mesh model because of the lack of support for 802.11ac in OpenWRT. The table's source is from section 4.2.3.1 of IRTF Internet-Draft *Alternative Network Deployments. Taxonomy and characterization*[14]; it describes community networks such as Guifi.net. This draft document is a work in progress of GAIA Research Group[15, 16] (Global Access to the Internet for All) from IRTF. IRTF (Internet Research Task Force) is a research organization parallel to IETF. It describes community networks.

The most common services in Guifi.net's network are: snpservices, dnsservices and internet sharing. Snpservices, as seen above, provides mechanisms of control and statistics about Guifi.net's nodes. Dnsservices is a service executed in zones' servers, regularly it queries Guifi.net's web to get its domain configuration, because these DNSs are configured via Guifi.net's web. There are multiple ways to share Internet; the most common, from traditional model, is a federated HTTP proxy. It shares internet with Guifi.net users thanks to an authenticated system based on LDAP. In the mesh model, users have direct access to a shared internet via layer 3 (network) tunnels. An equivalent solution to get direct access to Internet in traditional model is layer 4 (application) tunnels, but is less efficient.

An improvement to dnsservices is to include SRV record for XMPP and SIP near MX record's place in the Guifi.net's web configuration (see figure 5.5)

Cloudy[17] is service that proposes integration in a node of snpservices, dnsser-

[14] `https://tools.ietf.org/html/draft-manyfolks-gaia-community-networks-02`.

[15] `https://trac.tools.ietf.org/group/irtf/trac/wiki/gaia`.

[16] `https://irtf.org/gaia`.

[17] `http://cloudy.community/`.

Table 5.2: 802.11 (Wi-Fi) Technologies

802.11 prot	Release date	Freq (GHz)	BWdth (MHz)	Data Rate per stream (Mbps)	indoor (m)	outdoor (m)
a	Sep 1999	5	20	6, 9, 12, 18, 24 36, 48, 54	35	120
b	Sep 1999	2.4	20	1, 2, 5.5, 11	35	140
g	Jun 2003	2.4	20	6, 9, 12, 18, 24 36, 48, 54	38	140
n	Oct 2009	2.4/5	20	7.2, 14.4, 21.7, 28.9, 43.3, 57.8, 65, 72.2	70	250
n	Oct 2009	2.4/5	40	15, 30, 45, 60, 90, 120, 135, 150		
ac	Nov 2011	5	20	Up to 87.6		
ac	Nov 2011	5	40	Up to 200		
ac	Nov 2011	5	80	Up to 433.3		
ac	Nov 2011	5	160	Up to 866.7		

vices, federated proxy and many more. A node can manage its services, add a new one, and announce or discover services from other cloudy nodes through Guifi.net.

Networks and services of Guifi.net are scalable and extendable. For the network's case, that is because Guifi.net is a small Internet's network architecture. For the service's case, that is because Guifi.net-related services are simple; Cloudy is complex, but is based on a software that manages the Community-Lab testbed of an European project called CONFINE[18]. The least scalable and extendable part of Guifi.net is its web, but is not very problematic: the web does not provide multimedia content, forks can be done because is an open source project, and the web architecture has solutions to prevent scalability and extendability problems such as load balancing.

5.2 Requirements

5.2.1 Network requirements

Analysis of network requirements for an hypothetical RTC system in Guifi.net network. Guifi.net is not a network designed for RTC, but has promising features to transport it.

[18] `https://wiki.confine-project.eu/`.

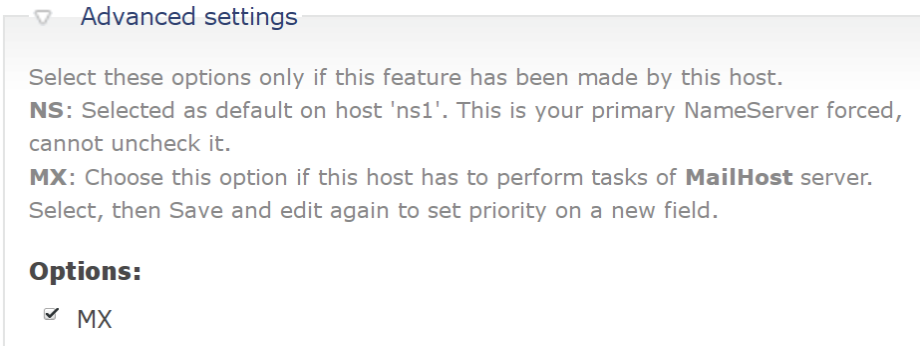


Figure 5.5: MX record in Guifi.net web

Guifi.net is a dumb network, i.e., a network that only routes packets and does not differentiate between applications. It has in-band signaling, i.e., signaling information uses the same channel as the user data. QoS could prioritize real-time traffic over non-real-time traffic, but currently is complex to apply QoS in Guifi.net because of the difficulty to set up globally, firmware incompatibilities and lack of interest. Nearly all Guifi.net operators offering VoIP have deployed QoS with RouterOS in the parts of the network where they manage; this affects negatively to the Guifi.net's heterogeneity.

Delay negatively affects interaction in multimedia applications. For audio, G.114[19] determines that if delay is below 150 ms, then most applications would not be significantly affected, while delays above 400 ms are unacceptable for general network planning purposes. For interactive video, [Simpson, 2008] determines that delay in each direction should be below 150 msec and normally must be below 400 msec. For one-way applications, delay is usually not a significant factor.

Figure 5.6 shows a delay test for two different long network paths. The results show that the delay could be adequate to use RTC applications in Guifi.net.

A delay test for two different long network paths (see figure 5.6) shows that the delay could be adequate to use RTC applications in Guifi.net.

The bandwidth of Guifi.net's links should be enough to pass multimedia information. The mandatory audio and video WebRTC's codecs are discussed. G.711 audio codec needs a throughput of 64 Kbps bitrate (8 kHz sampling frequency x 8 bits per sample). Opus audio codec, according to [Valin et al., 2012], scales from low bitrate narrowband speech at 6 Kbps to very high quality stereo music at 510 Kbps. Table 5.3 shows throughput of H.264 video codec, source from Table 4-6

[19]International Telecommunication Union. ITU-T Recommendation G.114 `http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-G.114-200305-I!!PDF-E`.

```
traceroute to 10.1.24.1 (10.1.24.1), 30 hops max, 60 byte packets
 1  BCNLlenguadoc17p0--76090.ip.guifi.net (10.1.57.193)  2.957 ms  5.789 ms  5.786 ms
 2  BCNLaFabraRd2--57543.ip.guifi.net (10.1.56.1)  5.778 ms  9.585 ms  9.588 ms
 3  BCNLaFabraRd1--49174.ip.guifi.net (10.1.56.2)  9.584 ms  9.579 ms  9.573 ms
 4  BCNOnzedeSetembreNO--6345.ip.guifi.net (172.25.55.33)  11.841 ms  11.838 ms  14.569 ms
 5  BCNmoli--13529.ip.guifi.net (172.25.33.14)  14.567 ms  14.559 ms  17.008 ms
 6  BDNCtraStaColomaST--5395.ip.guifi.net (10.139.16.129)  17.007 ms  9.547 ms  8.412 ms
 7  BDNGuasch69ST--5533.ip.guifi.net (10.139.16.33)  11.370 ms  13.638 ms  18.578 ms
 8  BDNGuaschTrentauST--5566.ip.guifi.net (10.139.16.161)  18.575 ms  18.565 ms  18.556 ms
 9  BCNpontTreballST--6381.ip.guifi.net (172.25.6.209)  18.544 ms  21.504 ms  21.502 ms
10  BCNAragoOlostRd1--12992.ip.guifi.net (10.138.27.97)  24.556 ms  24.558 ms  24.551 ms
11  BCNAragoOlostRB750--40490.ip.guifi.net (172.25.34.109)  31.480 ms  31.466 ms  31.426 ms
12  BCNArago621-OMNITIK--38930.ip.guifi.net (172.25.52.169)  23.254 ms  18.340 ms  21.485 ms
13  BCNgranVia940Rd1--15066.ip.guifi.net (10.139.92.33)  24.092 ms  24.083 ms  24.083 ms
14  BCNhangarST2--15624.ip.guifi.net (10.139.94.97)  29.397 ms  29.384 ms  34.196 ms
15  HW-AcerRouter--27612.ip.guifi.net (172.25.38.161)  34.180 ms  34.158 ms  34.139 ms
16  BCNdsgRouter--40605.ip.guifi.net (172.25.35.125)  34.118 ms  34.097 ms  36.821 ms
17  10.228.207.3 (10.228.207.3)  36.818 ms  39.008 ms  38.953 ms
18  * UPC-CN-C6-E208-9d04--57728.ip.guifi.net (10.1.24.1)  31.530 ms  35.558 ms
traceroute to 10.228.193.1 (10.228.193.1), 30 hops max, 60 byte packets
 1  BCNLlenguadoc17p0--76090.ip.guifi.net (10.1.57.193)  2.936 ms  2.907 ms  2.905 ms
 2  BCNLaFabraRd2--57543.ip.guifi.net (10.1.56.1)  6.327 ms  6.321 ms  6.309 ms
 3  BCNLaFabraRd1--49174.ip.guifi.net (10.1.56.2)  8.588 ms  8.582 ms  8.613 ms
 4  BCNOnzedeSetembreNO--6345.ip.guifi.net (172.25.55.33)  11.461 ms  11.461 ms  11.454 ms
 5  BCNmoli--13529.ip.guifi.net (172.25.33.14)  16.575 ms  18.985 ms  18.987 ms
 6  BDNCtraStaColomaST--5395.ip.guifi.net (10.139.16.129)  22.690 ms  7.927 ms  10.395 ms
 7  BDNGuasch69ST--5533.ip.guifi.net (10.139.16.33)  13.405 ms  13.393 ms  16.365 ms
 8  BDNGuaschTrentauST--5566.ip.guifi.net (10.139.16.161)  16.357 ms  19.868 ms  19.856 ms
 9  BCNpontTreballST--6381.ip.guifi.net (172.25.6.209)  19.839 ms  19.821 ms  22.467 ms
10  BCNstPauParlamentST--6380.ip.guifi.net (10.138.27.225)  22.462 ms  24.964 ms  24.918 ms
11  BCNmargaritRB600--11404.ip.guifi.net (10.138.27.33)  24.894 ms  24.885 ms  30.105 ms
12  BCNVistaRicaRB2011--57060.ip.guifi.net (10.228.201.193)  20.117 ms  18.148 ms  20.901 ms
13  BCNLorda17-RBx86--38506.ip.guifi.net (10.228.199.193)  20.972 ms  20.952 ms  20.934 ms
14  BCNRembrandtRd1--57824.ip.guifi.net (10.139.95.97)  23.901 ms  23.894 ms  26.263 ms
15  BCNcstlljs352-RB--45277.ip.guifi.net (172.25.37.41)  29.886 ms  33.500 ms  33.487 ms
16  BCNMargarit92ST1--31936.ip.guifi.net (10.228.194.225)  33.471 ms  33.454 ms  33.437 ms
17  BCNriereta24ST1--29423.ip.guifi.net (10.228.193.1)  40.293 ms  40.288 ms  43.127 ms
```

Figure 5.6: Measure of delay with traceroute tool, takes 1 and 2

of [Simpson, 2008]. VP8 video codec is an alternative of H.264 and has a similar throughput.

Table 5.3: MPEG Video Compression Comparison

	MPEG-1	MPEG-2	MPEG-3	MPEG-4 AVC/H.264
Common stream rates	500-1500 Kbps	2.5-50 Mbps	100 Kbps-10 Mbps	1-4 Mbps SD[20] 4-10 Mbps HD[21]

Figure 5.7 shows a bandwidth test of a Youtube's video using *Stats for nerds* option (video source[22]). It uses different qualities: top's image has quality 144p[23], and average throughput 247 Kbps, middle's image has quality 360p and average throughput 740 Kbps, bot's image has quality 1080p and average throughput 4324 Kbps.

[73]SD: video image with a resolution of 1950's standards.

[74]HD: video image with resolution greater than SD.

[22]`https://www.youtube.com/watch?v=BdYKR1RsYFY`.

[23]for example 144 is width's image and p means progressive video.

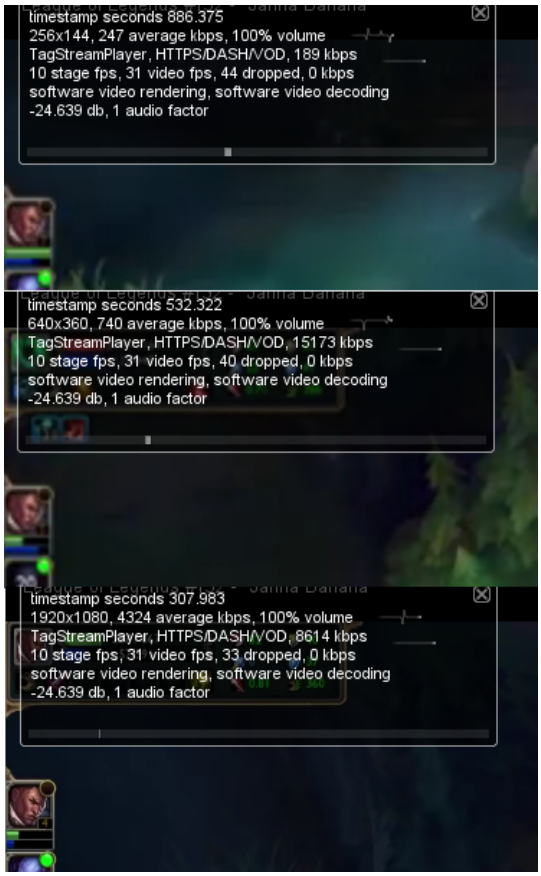


Figure 5.7: Youtube's videos with different qualities

For a common 802.11n link, from 15 to 150 Mbps, it can manage some RTC sessions concurrently. In a videoconference's situation, the most important part is audio, and the video could have low quality; because from the communicative point of view, audio is more important than video.

5.2.2 Generic use cases

The use cases developed are those that are bold.

Definitions:

- Actors or Roles have a user and/or admin account. The user has minimum permissions in the application, the admin has all the permissions in the application. It can be defined one or more middle actors that have more permissions than user and less permissions than admin.

- A call refers to an audio or video call, bidirectional communication with

video and/or audio channel.

- A user is available if is connected to the service and busy.

A general RTC service could be defined as it follows:

1. **Send calls: a user calls another user with an audio channel. Optional channels of communication if available: video and chat.**
    - Access to the service through an application.
    - Authentication.
    - Decision of which user it is wanted to call.
    - The call is accepted by the other user.
    - Bidirectional communication.
    - One of the two users stop the communication.
2. **Receive calls: a user receives a call only if is connected to the service with at least one device and is available.**
    - Access to the service through an application.
    - Authentication.
    - A lapse of random time until a call is received. If there are more devices of the same user, all of them receive the call, but only one can accept it.
    - The call is accepted.
    - Bidirectional communication.
    - One of the two users stop the communication.
3. A user can subscribe or unsubscribe to the RTC service.
4. Call history: a user can see the calls sent, received, missed. One can delete it.
5. Missed call notification
    - When a user calls to another user that is not available. A missed call notification is generated to be delivered to the other user.
    - A user has been notified by a missed call if: (1) the device is compatible with this service and (2) one is available.
6. Contact list

- A user can see the status (online, offline, busy, etc.) of another user in its contact list if one is allowed by the other user.
- A user can add or remove another user from the contact list.

7. Chat rooms
   - A user can be in a public place where there are rooms and people talk openly.
   - A user can speak privately to the users connected to this place.
   - The identity in the chat room is the same as in the contact list.
8. User preferences
   - User can set its own photo, nickname and description.
   - Users can set if a room is able to record a history conversation (and files) such that users that connect and disconnect can follow the conversation.
   - User can change password of its account.
9. Share advanced media
   - User can share its screen with another user.
   - User can share files of limited size in a room or privately to another user. The data is temporarily stored.
   - Share N streams to N users (Multi-user bidirectional videoconference).
   - Share one stream to N users (Streaming).
10. Administration
    - User can only change its settings. Admin can change configuration of all users.
    - Users can report other users because of a social conflict, admin is notified.
11. **Integration: all the services are integrated and is the same account.**

The Guifi.net service is defined as it follows:

1. A user can connect to a server if he could reach it with good quality, if not, one can easily install it in its zone.

2. If a server reach another, the users of a server can communicate to the users of another server.

3. **The service is compatible with VoIP Guifi.net project.**

5.2.3 Required components by specific use cases

This section presents the components required for the network architecture design. Concretely, the following requirements are described: send, receive calls, subscribe, unsubscribe and have integration. For a WebRTC scenario, the components are distributed.

- Signaling protocol (SIG) manages the side to side connections and logic to establish the call. The two peers must use a compatible signaling protocol.

- Gateway adapts or converts the communication to work with different communication systems. The most important difference between communication systems is the signaling protocol. There are two types of gateways: Signaling Gateway (SGW) and Media Gateway (MGW).

- Application interface gives appropriate interaction to the actors in order to perform the different operations. It is distributed, a web server (WSRV) offering a page, and a client executing the web application (WAPP) through the web browser.

- Authentication service (AUTH) restricts access of service only to permitted users. It differentiates available operations depending on if is user or admin. It provides single sign on, after the web page is accessed, the user can operate.

- Connectivity solver (CS) a set of tools to avoid common communication problems that appear in networking scenarios. The most common problems are NAT and firewall traversal.

- Database (DB) stores and encrypts personal information or preferences for a particular user. It is accessible through web application if succeed in authentication.

5.3 Component selection

Selection of technologies for each component.

- SIG: SIP, because is a signaling protocol that since its origins is designed to perform calls.

- SGW: SIP over Websockets. It is required to transmit SIP signaling via Web Application.

- MGW: Conversion between SIP and WebRTC media channels is required due to incompatibilities at transport layer. According to an author[24] the majority of RTP traffic in VoIP networks is not secured today; when it is secured, most of the deployments he has seen use SDES (Session Description Protocol (SDP) Security Descriptions for Media Streams [Andreasen et al., 2006]). Guifi.net's VoIP project use RTP tunneled with VPN.

- Web Application. WebRTC is being implemented in web browsers. That is why is required to use web technologies. Given web application's constraint, other selections have to be done. Another possibility is to do an standalone WebRTC application using the native code[25], but it requires more development and is less flexible (see SWOT analysis).

  - WAPP: HTML/CSS/JS. HTML (HyperText Markup Language[26]) for web content, CSS (Cascading Style Sheets[27]) for web design, JS (Javascript[28]) for language programming features. JS has the commands to access WebRTC API's features.

  - WSRV: HTTP server. It serves HTML/CSS/JS pages to web browsers.

- AUTH: OAuth. It is used to authorize users to HTTP services.

- CS: STUN, TURN. They are servers which WebRTC needs to solve some of the possible connectivity problems.

- DB: RDBMS (Relational Database Management System[29]). Stored information is few and has a clear structure. The number of users is low.

[24] `https://webrtchacks.com/webrtc-must-implement-dtls-srtp-but-must-not-implement-sdes/`
[25] `http://www.webrtc.org/native-code`
[26] `http://www.w3.org/html/`
[27] `http://www.w3.org/Style/CSS/Overview.en.html`
[28] `http://www.w3.org/standards/webdesign/script`
[29] Database based on relational database model. Usually employs SQL as their query language.

5.4 Network architecture design

Component diagram (see 5.8) is the key point to explain the different procedures and gateways.

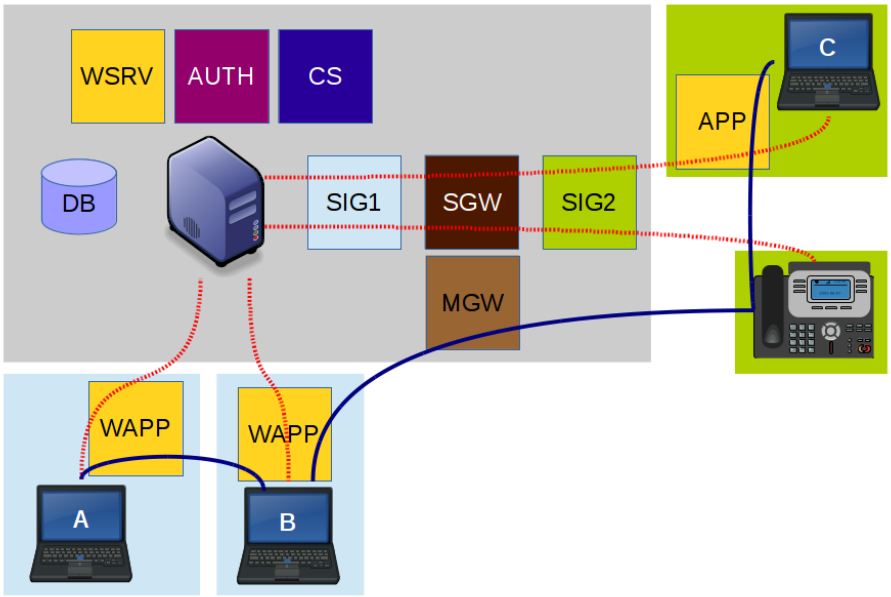


Figure 5.8: Component Diagram

Signaling between server and clients. Component diagram's figure shows four clients (A, B, C, phone) which have signaling connections to a server. The server has integrated multiple components; but they could be distributed in different server nodes. Laptops A and B have web application (WAPP) provided by the web server (WSRV) which connect to signaling server (SG1) through SIP over websocket (red line). Laptop C has a standalone softphone application which connects to signaling server (SG2) through SIP. Phone connects to signaling server (SG2) through SIP. Figure 5.9 shows a signaling gateway (SGW) which translates SIP over websocket to SIP.

Media transport between clients. Blue lines between clients shows how is handled the media. Between WebRTC or between SIP, media could be p2p. Figure 5.10 shows a media gateway (MGW) which translates media that goes between WebRTC and SIP clients. Additionally, if needed, connectivity solver component provides STUN and TURN services to clients. TURN service is expensive; it could require authentication for its use.

Authentication. In WebRTC, clients use OAuth (AUTH) to authenticate their users. They obtain an identity to sign their DTLS key, and they gain access to the

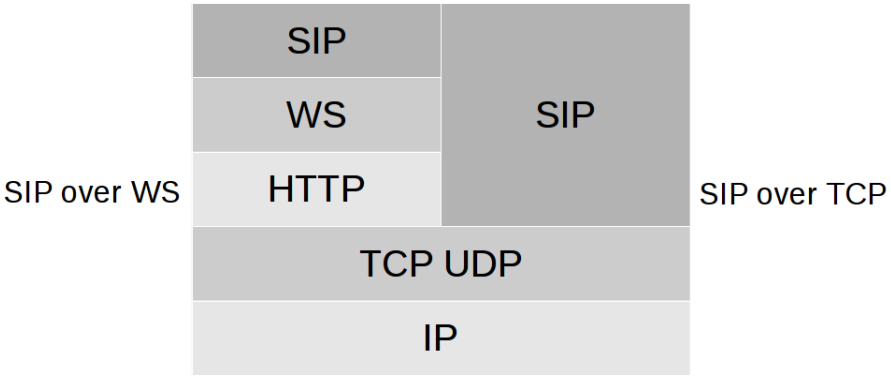


Figure 5.9: Signaling Gateway

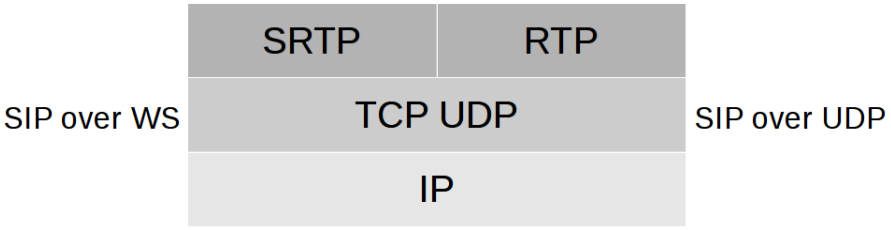


Figure 5.10: Media Gateway

RTC application. In a context call, during exchange of DTLS keys, other users can verify foreign identities. With SIP, clients can authenticate to signaling service; but verification is not possible. This source explains Identity and DTLS procedure with greater detail[30].

Database (DB) manages users accounts; therefore it assist authentication module and other components that needs authentication.

5.5 Implementation

Software application's selection for each component.

- SIG: kamailio[31]. It is a long career project that comes from the origins of SIP Proxy. Kamailio can also manage websocket connections and link them to legacy SIP. Websocket secure (WSS) was not tested.
- SGW: jssip. It is written by the authors of SIP over websocket RFC.

[30] https://www.terena.org/activities/tf-webrtc/meeting2/slides/20150509-Cisco-WebRTC.pdf

[31] http://kamailio.org/

- MGW: rtpengine[32] (formerly rtpproxy-ng kamailio's module for mediaproxy-ng). It is prepared to interconnect WebRTC to RTP[33]; because it is compatible with SDES, DTLS-SRTP, etc. Rtpengine was not tested.
- WAPP: tryit.jssip.net[34]. It is a functional demo application proposed by jssip.
  - Browsers: Firefox and Chrome. They are the most used browsers[35].
- WSRV: apache HTTP server. It is a common, easy to use and well-known HTTP server. HTTPS was not tested.
- AUTH: kamailio. This is the fast way to start implementing an RTC system. Kamailio can manage accounts. OAuth was not tested.
- CS: coturn. It is an advanced (updated) version of rfc5766-turn-server[36] and is recommended by this source[37]. It was not tested.
- DB: MySQL. It is a common, easy to use and well-known relational database.

Not tested (summary): WSS (SIG), rtpengine (MGW), HTTPS (WSRV), coturn (CS), OAuth (AUTH).

5.6 Demo

This section shows a POC (Proof Of Concept) of WebRTC that runs on Guifi.net. It is about a video call between two users. Refer to appendices to obtain documentation about how to install sip proxy, and web application with WebRTC support.

WebRTC is still under development. That is why is recommended to use browsers' latest versions. For Google Chrome, version 42.0.2311.152. For Firefox, version 38.0.5

Figure 5.11 shows tryit web client. It is generic, and can be used to login other proxy servers. The login parameters needed are the SIP URI, for example `bob@10.1.56.201`, its password and web socket URI, for example `ws://10.1.56.201:8080`. This means that a kamailio sip proxy has IP `10.1.56.201`, it is listening websocket port `8080` and it has a user registered called Bob.

[32] `https://github.com/sipwise/rtpengine`
[33] `http://www.kamailio.org/w/tag/rtpproxy-ng/`
[34] `http://tryit.jssip.net/`
[35] `http://gs.statcounter.com/#all-browser-ww-monthly-201502-201502-map`
[36] `https://code.google.com/p/rfc5766-turn-server/`
[37] `https://webrtchacks.com/coturn/`

Figure 5.11: Web SIP Client

Figure 5.12 shows the next screen. Bob is successfully registered. To perform a call, fill green box with SIP URI and click call.

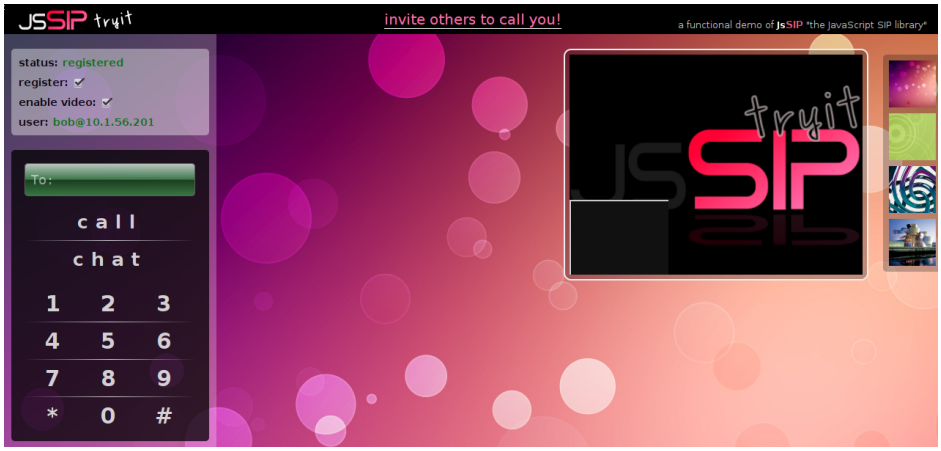


Figure 5.12: Waiting a call

Figure 5.13 shows a call in progress. There are three services available: audio, video and chat. Video is on the right side of the interface, smaller frame is Bob's screen and larger frame is from Alice; the videos' frame are similar because this POC used two laptops that are placed in the same location. Additionally, this call opened a chat (datachannel).

A wireshark[38] capture of call procedure from Bob's perspective (figure 5.14) shows how it is a typical call's SIP procedure encapsulated with websocket. The websocket's seven messages are:

[38]network traffic sniffer, `https://wireshark.org`

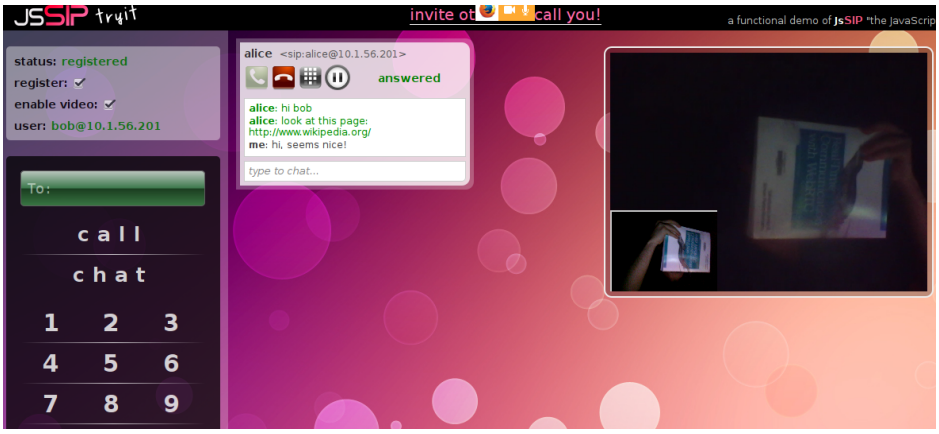


Figure 5.13: A call in progress

1. REGISTER Bob. After register, one can send or receive calls.
2. SIP 200 OK. Authentication success for Bob's user from SIP Proxy.
3. INVITE Alice (contains SDP offer). Bob request a call to Alice.
4. SIP 100 TRYING. SIP Proxy informs to Bob that its call is being processed.
5. SIP 180 RINGING. SIP Proxy informs to Bob that Alice SIP's device is ringing.
6. SIP 200 OK (contains SDP answer). Alice accepts the call
7. ACK. Bob confirms to SIP Proxy that he received Alice's acceptation of Bob's call.

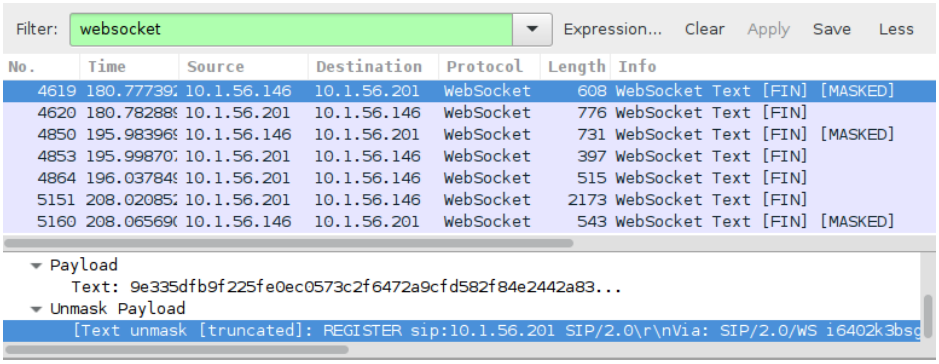


Figure 5.14: Wireshark capture of websocket's SIP

Chapter 6

CONCLUSIONS AND FUTURE WORK

What has been done and what remains to be done

6.1 Conclusions

Coming back to the objectives of this project, all are attainable. WebRTC is open source and encryption is mandatory for user data. Since WebRTC is a natural evolution of previous technologies, it is backward compatible with previous RTC solutions, to some degree; but this project could not evaluate it exhaustively. The network architecture design proposed is a good starting point for continuing to implement RTC in Guifi.net, because WebRTC design is as scalable and extendable as any web project. Up until now, many people have steadily been improving the ease of installation and usability of WebRTC and this work has attempted to summarize some of WebRTC's work lines. Also, it has proposed a direction for Guifi.net's developer community. For other organizations such as open source communities, companies, etc. a similar approach could be applied.

WebRTC greatly facilitates use of RTC in organizations of all sizes. Smaller organizations can solve the use cases proposed. Larger organizations can benefit by solving even more complex use cases. Pre-WebRTC attempts were more challenging and they tended to be implemented in larger organizations.

As WebRTC architecture design is similar to Web architecture design, it cannot be completely decentralized. This is one of the disadvantages of WebRTC; its architecture design needs at least one centralized server, because of that a minimum of organization is required in order to implement it. However the advantage of these servers is that they require low CPU usage and are therefore inexpensive.

There are three featured topics of WebRTC that are negative. Firstly, nowa-

days it is not possible to use WebRTC and be anonymous; the ICE procedure reveals a considerable amount of information about a web page visitor. Secondly, WebRTC media channel traffic is still not supported via an HTTP Proxy and this issue affects Guifi.net's traditional model. And thirdly, WebRTC and SIP interoperability need a gateway for the media channel. However, the gateway mainly requires network resources, which are available for free inside Guifi.net.

6.2 Future Work

The suggested direction is firstly to improve currently implemented use cases and secondly to continue to implement those remaining.

To improve currently implemented use cases, integration of the proposed network architecture of WebRTC with the Cloudy and the Guifi.net's web. With relatively little effort and thanks to the existence of these two projects, end-users could definitely profit from the ease of installation and usability of the WebRTC. Also, improve analysis and testing of backward compatibility of WebRTC and SIP.

To continue to implement remaining use cases, is needed for example contact list and presence service. The suggested direction is to develop a web-based CUSAX Client (Combined Use of the SIP and the XMPP, [Ivov et al., 2013]). CUSAX Client takes SIP for telephony-like services, and XMPP for instant messaging and presence services. The following libraries compatible with websocket are recommended for working on the CUSAX Client: jssip for SIP and strophe.js[1] for XMPP. This proposal pretends to take the advantages of each system.

A complementary direction for research is improving integration of the signaling services XMPP[2] and/or SIP[3] inside OpenWRT-based firmwares such as qMp and/or libremesh. Such an approach could further decentralize the RTC system but requires additional intelligence or automatic ways to federate a large number of devices.

[1] `http://strophe.im/strophejs/`, example `https://prosody.im/chat/`.

[2] `https://downloads.openwrt.org/barrier_breaker/14.07/x86/generic/packages/packages/prosody_0.9.4-1_x86.ipk`.

[3] `https://downloads.openwrt.org/barrier_breaker/14.07/x86/generic/packages/telephony/`.

Bibliography

[Andersson and Madsen, 2005] Andersson, L. and Madsen, T. (2005). Provider Provisioned Virtual Private Network (VPN) Terminology. RFC 4026 (Informational).

[Andreasen et al., 2006] Andreasen, F., Baugher, M., and Wing, D. (2006). Session Description Protocol (SDP) Security Descriptions for Media Streams. RFC 4568 (Proposed Standard).

[Bankoski et al., 2011] Bankoski, J., Koleszar, J., Quillio, L., Salonen, J., Wilkins, P., and Xu, Y. (2011). VP8 Data Format and Decoding Guide. RFC 6386 (Informational).

[Baugher et al., 2004] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and Norrman, K. (2004). The Secure Real-time Transport Protocol (SRTP). RFC 3711 (Proposed Standard). Updated by RFCs 5506, 6904.

[Castillo et al., 2014] Castillo, I. B., Villegas, J. M., and Pascual, V. (2014). The WebSocket Protocol as a Transport for the Session Initiation Protocol (SIP). RFC 7118 (Proposed Standard).

[Deering and Hinden, 1998] Deering, S. and Hinden, R. (1998). Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard). Updated by RFCs 5095, 5722, 5871, 6437, 6564, 6935, 6946, 7045, 7112.

[Dodd, 2012] Dodd, A. Z. (2012). *The essential guide to telecommunications*. Prentice Hall, fifth edition.

[Farinacci et al., 2000] Farinacci, D., Li, T., Hanks, S., Meyer, D., and Traina, P. (2000). Generic Routing Encapsulation (GRE). RFC 2784 (Proposed Standard). Updated by RFC 2890.

[Fette and Melnikov, 2011] Fette, I. and Melnikov, A. (2011). The WebSocket Protocol. RFC 6455 (Proposed Standard).

[Fielding et al., 1999] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. (1999). Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard). Obsoleted by RFCs 7230, 7231, 7232, 7233, 7234, 7235, updated by RFCs 2817, 5785, 6266, 6585.

[Fischl et al., 2010] Fischl, J., Tschofenig, H., and Rescorla, E. (2010). Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS). RFC 5763 (Proposed Standard).

[Frank, 2004] Frank, O. (2004). *Voice over 802.11*. Artech House, first edition.

[Gulbrandsen et al., 2000] Gulbrandsen, A., Vixie, P., and Esibov, L. (2000). A DNS RR for specifying the location of services (DNS SRV). RFC 2782 (Proposed Standard). Updated by RFC 6335.

[Handley et al., 2006] Handley, M., Jacobson, V., and Perkins, C. (2006). SDP: Session Description Protocol. RFC 4566 (Proposed Standard).

[Hardt, 2012] Hardt, D. (2012). The OAuth 2.0 Authorization Framework. RFC 6749 (Proposed Standard).

[Harrington et al., 2002] Harrington, D., Presuhn, R., and Wijnen, B. (2002). An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. RFC 3411 (INTERNET STANDARD). Updated by RFCs 5343, 5590.

[Ivov et al., 2013] Ivov, E., Saint-Andre, P., and Marocco, E. (2013). CUSAX: Combined Use of the Session Initiation Protocol (SIP) and the Extensible Messaging and Presence Protocol (XMPP). RFC 7081 (Informational).

[Klensin, 2008] Klensin, J. (2008). Simple Mail Transfer Protocol. RFC 5321 (Draft Standard).

[Kurose and Ross, 2013] Kurose, J. F. and Ross, K. W. (2013). *Computer Networking. A Top-Down Approach*. Pearson, sixth edition.

[Loreto and Romano, 2014] Loreto, S. and Romano, S. P. (2014). *Real-Time Communication with WebRTC*. O'Reilly, first edition.

[Loreto et al., 2011] Loreto, S., Saint-Andre, P., Salsano, S., and Wilkins, G. (2011). Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP. RFC 6202 (Informational).

[Lougheed and Rekhter, 1991] Lougheed, K. and Rekhter, Y. (1991). Border Gateway Protocol 3 (BGP-3). RFC 1267 (Historic).

[Ludwig et al., 2009] Ludwig, S., Beda, J., Saint-Andre, P., McQueen, R., Egan, S., and Hildebrand, J. (2009). Jingle. XEP-0166 (Standards Track).

[Mahy et al., 2010] Mahy, R., Matthews, P., and Rosenberg, J. (2010). Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). RFC 5766 (Proposed Standard).

[Mealling, 2002] Mealling, M. (2002). Dynamic Delegation Discovery System (DDDS) Part Three: The Domain Name System (DNS) Database. RFC 3403 (Proposed Standard).

[Mockapetris, 1987] Mockapetris, P. (1987). Domain names - concepts and facilities. RFC 1034 (INTERNET STANDARD). Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936.

[Moy, 1998] Moy, J. (1998). OSPF Version 2. RFC 2328 (INTERNET STANDARD). Updated by RFCs 5709, 6549, 6845, 6860, 7474.

[Nichols et al., 1998] Nichols, K., Blake, S., Baker, F., and Black, D. (1998). Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474 (Proposed Standard). Updated by RFCs 3168, 3260.

[Ott and Carrara, 2008] Ott, J. and Carrara, E. (2008). Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF). RFC 5124 (Proposed Standard).

[Paterson et al., 2014] Paterson, I., Smith, D., Saint-Andre, P., Moffitt, J., Stout, L., and Tilanus, W. (2014). Bidirectional-streams Over Synchronous HTTP (BOSH). XEP-0124 (Standards Track).

[Postel, 1980] Postel, J. (1980). User Datagram Protocol. RFC 768 (INTERNET STANDARD).

[Postel, 1981a] Postel, J. (1981a). Internet Protocol. RFC 791 (INTERNET STANDARD). Updated by RFCs 1349, 2474, 6864.

[Postel, 1981b] Postel, J. (1981b). Transmission Control Protocol. RFC 793 (INTERNET STANDARD). Updated by RFCs 1122, 3168, 6093, 6528.

[Rekhter et al., 1996] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., and Lear, E. (1996). Address Allocation for Private Internets. RFC 1918 (Best Current Practice). Updated by RFC 6761.

[Rescorla, 2000] Rescorla, E. (2000). HTTP Over TLS. RFC 2818 (Informational). Updated by RFCs 5785, 7230.

[Rescorla, 2010] Rescorla, E. (2010). Keying Material Exporters for Transport Layer Security (TLS). RFC 5705 (Proposed Standard).

[Rescorla and Modadugu, 2006] Rescorla, E. and Modadugu, N. (2006). Datagram Transport Layer Security. RFC 4347 (Proposed Standard). Obsoleted by RFC 6347, updated by RFCs 5746, 7507.

[Rosenberg, 2010] Rosenberg, J. (2010). Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245 (Proposed Standard). Updated by RFC 6336.

[Rosenberg, 2013] Rosenberg, J. (2013). SIMPLE Made Simple: An Overview of the IETF Specifications for Instant Messaging and Presence Using the Session Initiation Protocol (SIP). RFC 6914 (Informational).

[Rosenberg et al., 2008] Rosenberg, J., Mahy, R., Matthews, P., and Wing, D. (2008). Session Traversal Utilities for NAT (STUN). RFC 5389 (Proposed Standard). Updated by RFC 7350.

[Rosenberg et al., 2002] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and Schooler, E. (2002). SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard). Updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922, 5954, 6026, 6141, 6665, 6878, 7462, 7463.

[Saint-Andre, 2011] Saint-Andre, P. (2011). Extensible Messaging and Presence Protocol (XMPP): Core. RFC 6120 (Proposed Standard).

[Schulzrinne et al., 2003] Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V. (2003). RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (INTERNET STANDARD). Updated by RFCs 5506, 5761, 6051, 6222, 7022, 7160, 7164.

[Senie, 2002] Senie, D. (2002). Network Address Translator (NAT)-Friendly Application Design Guidelines. RFC 3235 (Informational).

[Sermersheim, 2006] Sermersheim, J. (2006). Lightweight Directory Access Protocol (LDAP): The Protocol. RFC 4511 (Proposed Standard).

[Simpson, 2008] Simpson, W. (2008). *Video Over IP*. Focal Press, second edition.

[Stewart, 2007] Stewart, R. (2007). Stream Control Transmission Protocol. RFC 4960 (Proposed Standard). Updated by RFCs 6096, 6335, 7053.

[Stout et al., 2014] Stout, L., Moffitt, J., and Cestari, E. (2014). An Extensible Messaging and Presence Protocol (XMPP) Subprotocol for WebSocket. RFC 7395 (Proposed Standard).

[Valin et al., 2012] Valin, J., Vos, K., and Terriberry, T. (2012). Definition of the Opus Audio Codec. RFC 6716 (Proposed Standard).

[Velázquez, 2010] Velázquez, T. (2010). Sistema de servidores voip del proyecto guifi.net. Bachelor's thesis. Universitat Autònoma de Barcelona (UAB).

[Vílchez, 2014] Vílchez, P. (2014). Starting, contributing and empowering community networks in cities. Bachelor's thesis. Universitat Pompeu Fabra (UPF).

Appendix A

DEMO INSTALLATION

Guide to install the demo. It assumes a Debian 7 Wheezy installation with Internet access and administration rights.

A.1 Kamailio with websocket support

Based on this source[1].

1. Add the following lines to `/etc/apt/sources.list`

   ```
   # kamailio
   deb http://deb.kamailio.org/kamailio wheezy main
   deb-src http://deb.kamailio.org/kamailio wheezy main
   ```

2. Validate the repository: download key `wget http://deb.kamailio.org/kamailiodebkey.gpg` and apply it `apt-key add kamailiodebkey.gpg`

3. Install kamailio with websocket support:
   `sudo apt-get install kamailio kamailio-websocket-modules kamailio-mysql-modules kamailio-tls-modules`

4. Install database: `sudo apt-get install mysql-server`. Remember the password, it will required later.

5. In file `/etc/default/kamailio` uncomment `RUN_KAMAILIO=yes`

6. In `/etc/kamailio/kamctlrc` put `SIP_DOMAIN=fill`, where `fill` is a domain or IP address

[1] `http://kb.asipto.com/kamailio:skype-like-service-in-less-than-one-hour`

7. Put `https://gist.github.com/jesusprubio/4066845` as configuration file for `/etc/kamailio/kamailio.conf` and modify with your IP address in lines:

```
#!substdef "!MY_IP_ADDR!172.16.190.128!g"
#!substdef "!MY_DOMAIN!172.16.190.128!g"
```

8. In /etc/kamailio/kamctlrc, put `SIP_DOMAIN=fill`, where fill is a domain or IP address, and uncomment `DBENGINE=MYSQL`, and uncomment the following lines:

```
## database host
DBHOST=localhost

## database name (for ORACLE this is TNS name)
DBNAME=kamailio

# database path used by dbtext, db_berkeley or sqlite
# DB_PATH="/usr/local/etc/kamailio/dbtext"

## database read/write user
DBRWUSER="kamailio"

## password for database read/write user
DBRWPW="kamailiorw"

## database read only user
DBROUSER="kamailioro"

## password for database read only user
DBROPW="kamailioro"
```

9. Restart kamailio service `sudo service kamailio restart`

10. Execute kamdbctl create, put MySQL password. Say yes to all questions.

11. Add users. Example; command to add Bob's user: `kamctl add bob bobsecret`, where bobsecret is his password.

12. Restart kamailio service `sudo service kamailio restart`

A.2 Tryit jssip Web SIP Client

1. Install HTTP server apache and git `sudo apt-get install apache2 git`.
2. Go to the following path with the command `cd /var/www` and obtain tryit application from `git clone https://github.com/opentelecoms-org/jssip-tryit-web-mirror`.
3. rename file to jssip: `mv oldname jssip`
4. Access the application in `http://server/jssip`, where server part could be a domain or an IP.