Aalto University
School of Electrical Engineering
Department of Communications and Networking

Petri Juhani Leppäaho

Design of Application Layer Gateways for Collaborative Firewalls

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology.

Espoo 03.05.2012

Thesis supervisor: Prof. Raimo Kantola
Aalto University

Thesis instructor: D. Sc. (Tech.) Nicklas Beijar
Aalto University

Aalto University **Abstract of Master's Thesis**

Author: Petri Juhani Leppäaho

Name of the Thesis: Design of Application Layer Gateways for Collaborative Firewalls

Date: 03.05.2012 **Language:** English **Number of pages: X + 111**

School: School of Electrical Engineering

Department: Department of Communications and Networking

Professorship: Networking Technology **Code:** S.38

Supervisor: Prof. Raimo Kantola, Aalto University

Instructor: D. Sc. (Tech.) Nicklas Beijar, Aalto University

IPv4 address exhaustion has been a common concern for a couple of last decades. The increased number of users and services has consumed the remaining addresses rather rapidly.

To alleviate the problem of address exhaustion, Network Address Translation (NAT) Classless Inter-Domain Routing and a new version of IP, namely IPv6 have been proposed. NATs translate the source address and often also the port number of the client sending an IP-packet to a server in the public address space. This is a problem for application layer protocols that refer to the hosts in the private address space using IP addresses. Usually, hosts that reside in the private address space are not reachable from the public network. Thus, a NAT is the crudest kind of firewall: it blocks all incoming traffic while it lets hosts in the private network access the public Internet. This, for example, blocks all incoming VoIP calls to hosts in the private realm. The IETF has a solution to NAT traversal but this solution has drawbacks.

Customer Edge Switching (CES) introduces a new type of collaborative firewall that is meant to replace NATs and tries to remove the drawbacks in known NAT traversal solutions. For many protocols, CES as such, provides a generic traversal mechanism. For protocols that carry address information on application layer additional algorithms are needed at the edge node.

In this thesis a prototype is developed to include Application Layer Gateway Functions to support two application layer protocols. These protocols are Session Initiation Protocol (SIP) and File Transfer Protocol (FTP). The testing done with these Application Layer Gateways proves that the developed prototype works with the protocols that carry contact information inside the payload, letting the hosts in different private address realms to communicate.

Keywords: CES, SIP, FTP, TCP, ALG, NAT

Aalto-yliopisto **Diplomityön tiivistelmä**

Tekijä: Petri Juhani Leppäaho

Työn nimi: Sovellustason yhdeyskäytävävien suunnittelu vuorovaikutteisiin palomuureihin

Päivämäärä: 03.05.2012 **Kieli:** englanti **Sivumäärä: X + 111**

Korkeakoulu: Sähkötekniikan korkeakoulu

Laitos: Tietoliikenne- ja tietoverkkotekniikan laitos

Professuuri: Tietoverkkotekniikka **Koodi:** S.38

Valvoja: Prof. Raimo Kantola, Aalto-yliopisto

Ohjaaja: TkT Nicklas Beijar, Aalto-yliopisto

Huoli IPv4 osoitteiden loppumisesta on ollut esillä jo parin viimeisen vuosikymmenen ajan. Lisääntynyt käyttäjien ja palvelujen määrä on kuluttanut osoitteita melko ripeästi.

Tätä ongelmaa on pyritty ratkaisemaan, osoitteenmuutoksilla (NAT), luokattomalla reitityksellä (CIDR) ja uudella IP versiolla, tarkemmin IPv6. NAT muuttaa lähdeosoitteen ja usein myös portin numeron julkisen verkon osoitteeksi. Tämä aiheuttaa ongelmia sovellustason protokollissa, jotka viittaavat käyttäjiin yksityisen verkon osoitteden pohjalta. Usein yksityisen verkon käyttäjät eivät ole saavutettavissa julkisesta verkosta. Siten NAT toimiikin yksinkertaisimpana mahdollisena palomuurina: estäen kaiken sisäänpäin tulevan liikenteen sallien kuitenkin yksityisen verkon käyttäjien olla yhteydessä julkiseen Internettiin. Tämä estää esimerkiksi kaikki sisään tulevat VoIP puhelut. IETF on kehittänyt osoitteenmuutoksen, mutta tällä ratkaisulla on kuitenkin haittansa.

Customer Edge Switching (CES) esittelee uudenlaisen tilallisen palomuuri, jonka tarkoituksana on korvata NAT-laitteet, pyrkii poistamaan haittoja nykysissä ratkaisuissa. Useta protokolla, CES mukaanlukien, pyrkivät tarjoamaan yleisen ratkaisun NAT-laitteiden läpäisyyn. Protokolla jotka kuljettavat yhteystietoja sovellustasolla tarvitsevat ylimääräisiä toiminnallisuuksia reunalaitteisiin.

Tässä työssä protyyppiä on kehitetty tarjoamaan sovellustason yhdyskäytävä kahdelle sovellustason protokollalle. Nämä protokollat ovat Session Initiation Protocol (SIP) ja File Transfer Protocol (FTP). Näiden sovellustason yhdyskäytävien testauksen perusteella voidaan osoittaa, että kehitetty prototyyppi pystyy toimimaan myös sellaisten protokollien kanssa jotka kuljettavat yhteystietoa tietosisällössään, mahdollistaen eri verkkoihin sijoittuneiden käyttäjien keskinäisen kommunikaation.

Avainsanat: CES, SIP, FTP, TCP, ALG, NAT

Acknowledgements

This Master's Thesis has been done for Aalto University, School of Electrical Engineering, Department of Communications and Networking in July 2011 – April 2012. This thesis is a part of the larger project regarding Customer Edge Switching.

I want to thank Raimo Kantola, my supervisor, for the opportunity to work in this project. His knowledge about the field provided plenty of information.

My gratitude also goes to Nicklas Beijar, my instructor. During the work he helped to find better solutions and gave many useful ideas.

I would also like to thank Jesus Llorente. He was also working in the project, but with a different task. His knowledge and friendship during the work were irreplaceable.

I want to thank Mari for her support and patience during my studies and thesis.

I also want to thank my parents for their support during my studies and thesis. Their appreciation towards the education has pushed me forward in my studies.

I am grateful for my brothers just for being the way they are. They have provided much needed relaxation during my studies and the thesis work.

May 3, 2012

Petri Leppäaho

Table of Contents

List of Figures

List of Tables

Abbreviations

ALG	Application Layer Gateway
ARP	Address Resolution Protocol
CES	Customer Edge Switching
CETP	Customer Edge Traversal Protocol
DNS	Domain Name System
DoS	Denial-of-Service
FTP	File Transfer Protocol
FQDN	Fully Qualified Domain Name
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICE	Interactive Connectivity Establishment
ICMP	Internet Control Message Protocol
ID	Identity
IMAP	Internet Message Address Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISP	Internet Service Provider
LAN	Local Area Network
LPN	Local Private Network
NAT	Network Address Translation
NAPTR	Naming Authority Pointer
POP	Post Office Protocol
OSI	Open Systems Interconnection
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SSH	Secure Shell
STUN	Session Traversal Utilities for NAT
TCP	Transmission Control Protocol
TURN	Traversal Using Relays around NAT
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
UNSAF	Unilateral Self Address Fixing
URI	Universal Resource Identifier

1. Introduction

In the current Internet, the addressing has become an increasing problem. Addresses provided by IPv4 are not enough. IPv6 would provide a sufficient number of addresses to solve the global addressing problem. However, the number of the users who have moved to IPv6 remains low. Different Network Address Translations (NATs) have prolonged the use of IPv4. NATs can also cause some problems to the protocols operating on higher OSI-layers. Customer Edge Switching (CES) tries to provide an alternative to NATs. It operates between customer and provider network providing similar functionality as the NAT. The key issue here is that CES can also provide protocol specific processing and that a CES network can have both clients and server that are globally reachable.

Unwanted traffic in the Internet is another problem. Network administrators deploy Firewalls, Intrusion detection systems and virus scanning software to protect their networks from this malice. Upon an incoming message a firewall will make a decision either to admit the packet or discard it. There are two types of Firewalls: stateless and stateful. A stateless firewall limits its actions to filtering incoming packets one by one. For many protocols this is not enough. Stateful firewalls keep flow state and can change their behavior for subsequent packets based on the fact they first receive a certain packet.

A Firewall can reside either in a network device such as router or in a host. With mains powered devices both of these options are extensively used. When the host is battery powered, it is possible that unwanted traffic can deplete the battery if it reaches the host leading to Denial of Service. This motives developing new solutions for protecting user devices from malicious traffic.

To address the problem of unwanted and malicious traffic, address exhaustion in IPv4 and the limitations of NATs, we have developed the concept of Customer Edge Switching. The idea of Customer Edge Switching is that all customer devices are placed in different private address realms and they communicate smoothly through edge nodes we call Customer Edge Switches that switch between the realms.

1.1 Research Problem

The purpose of this thesis is to develop and test the protocol specific algorithms for the Customer Edge Switching for Session Initiation Protocol (SIP) and File Transfer Protocol (FTP).

These protocols were chosen because they are well known examples of the protocols that carry IP address information in the role of identifiers in the application layer. This feature causes problems with NATs and a similar problem with CES. We set the requirement that our solution must provide an interrupt driven access to mobile devices. This is an enhancement compared to the present state of the art where User Agent Server in the SIP application needs to keep alive a NAT binding in order to be reachable by polling an address in the public network. We believe that this is important for the energy consumption in the battery powered devices and it also helps to reduce signaling overhead over the air interface.

1.2 Objectives

A previous prototype is further developed during this thesis. At the beginning of this work, the understanding of the functionality of the existing prototype was really important, in order to be able to add more functionality to it. We considered the Session Initiation Protocol as more challenging task and the algorithms that enable the prototype to support SIP were implemented before the work on File Transfer Protocol.

We also decided that both of these protocols need to be tested thoroughly to verify that the development of the algorithms was properly carried out and the logic worked as expected. Because of this both of the protocols were decided to be tested first in a virtual environment as this was easier when developing these algorithms. The functionality of the algorithms was tested in both virtual and physical environments. Results from these tests are included in the thesis.

The focus of this work is on the logic not on performance. The key idea is that the algorithms are designed in such a way that they do not require changes in the hosts.

1.3 Scope

This thesis provides a solution for SIP using either IP addresses or domain names. Analysis of the behavior of the SIP with Session Traversal Utilities for NAT (STUN), Traversal Using Relay NAT (TURN) and Interactive Connectivity Establishment (ICE) are not provided in this thesis. The reason is that if Customer Edge Switching were adopted, STUN/TURN would become obsolete. FTP is studied with communication between two CES devices and also with client/server located behind the same CES. The software used for thorough SIP testing is limited to one server and one client. The thesis does not take any action towards other protocols.

We will not do any performance testing because our prototype is programmed using Python and not optimized for high performance at all. We will not consider mobility scenarios. We will not provide a systematic analysis of the scalability aspects of our solution. Security is briefly mentioned but it is not addresses systematically in this thesis.

1.4 Structure

In the second chapter the concept and the functionality of the CES is explained. The prototype of CES is updated as a part of this thesis. The third chapter explains how Network Address Translation works. NAT affects the protocols operating on the application layer. Application layer protocols, SIP and FTP, are analyzed in the fourth chapter. Understanding how these protocols perform is vital to follow through the rest of the thesis. In the fifth chapter requirements for the Application Layer Gateways, are set. These give an idea how the research problem is approached. The solution for SIP ALG is presented in the sixth chapter. This chapter describes how individual algorithms process packets. The seventh chapter provides information about FTP ALG; the algorithms related to FTP traffic are explained. In the eighth chapter the structures, related to physical and virtual test scenarios, are explained. This helps understanding how the actual tests were performed. The ninth chapter evaluates the results from the tests. In the tenth chapter the conclusions for the thesis are presented.

2. Basics about CES implementation

The purpose of this chapter is to offer knowledge about the existing CES concept and prototype. At first, the history about CES as a concept is discussed. After that CES is described on high level. Then the chapter provides information about the functionality of CES, regarding both concept and prototype.

2.1 Background of the CES

David Clark introduced an idea of redefining the term End-to-End (E2E). The suggested term was Trust-to-Trust (T2T) [2]. CES prototype, which is used in the thesis, is based on work by Lauri Virtanen 2009. He created a working prototype as a part of his master's thesis [23]. The work is, however, partly related to the previous work done by Jussi Ryynänen 2008 [20].

The actual concept of the CES functionality was presented by Raimo Kantola 2009 [7]. This is presented in the Figure 2.1. CES devices are located at the edges of the customer networks and they are connected to the public network. IP addresses from the private networks are not revealed to the public network.

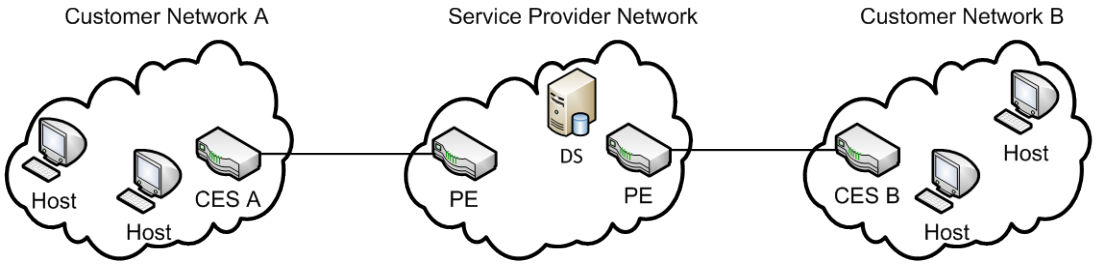

Figure 2.1: CES architecture

The code that was used to create the prototype is incremented during this thesis. Thus, providing solutions to the problems with two application layer protocols, Lauri Virtanen mentioned in his thesis to be a part of the future work [23].

2.2 CES briefly

The aim of the CES approach is to provide more functionality than traditional NAT devices. One key aspect of the CES is that it can remove the problem of the lack of free IPv4 addresses. It is possible to either reuse IPv4 addresses with CES devices or another option is to use routed Ethernet [11]. CES also strives towards improved security with Customer Edge Traversal Protocol (CETP) [9]. Some of the key terms related to the CES are presented next.

An *identity* (ID) is a label that points to an object such as a user, a host or a service on a host or the combination of user on a particular host. The scope of an *identity* can be public or local. A *public identity* is understood in the same way by hosts in many different domains (such as all ISP networks). A *local identity* is guaranteed to be unique only in one domain. We assume that a normal identity is not used for routing. We assume that an object may have many identities of different types.

An *address* is an identity of a location in a network (abnormal identity). An address can be either private or public. A public address is understood in the same way by many domains while a private address is guaranteed to point to the same location only in one domain. We do not talk about global addresses because the target of our design is to facilitate global communication using both public and private addresses.

A *name* (e.g. domain name) is an identity of an object in a textual format.

In a CES and identity is a key to admission policy database. A (domain) name can be mapped to an identity and an address for routing purposes using e.g. DNS NAPTR records. [12]

CES uses the Customer Edge Traversal Protocol (CETP). It supports many types of identities. In this thesis we do not discuss the details of CETP. For this work it is enough to know that CETP carries the source and target identities in the tunneling header and all traffic from the edge to edge is tunneled through the core network. In addition, CETP can carry many types of Type-Length-Value (TLV) elements for trust establishment between the edge nodes. Also these aspects of CETP are out of the scope in this work. [9] [13]

We always assume that one or several CES devices connect a stub network to the core network. This means that a CES does not carry transit traffic. It only either originates or terminates traffic. However, a CES can act as an anchor point in mobility scenarios. This limitation is needed in order to allow using local identities in global communication in some scenarios and still avoid routing loops.

2.3 CES-to-CES

As shown in the Figure 2.1 the structure behind the CES concept is that CES devices are located at the edges of the private networks. The idea is that customers in one network can still communicate with customers located in the other network by performing DNS queries before communication. One of the main ideas in the CES approach is that private addresses are not published to the public network. Because of this the users in the private networks can only be reached by IDs representing a specific host.

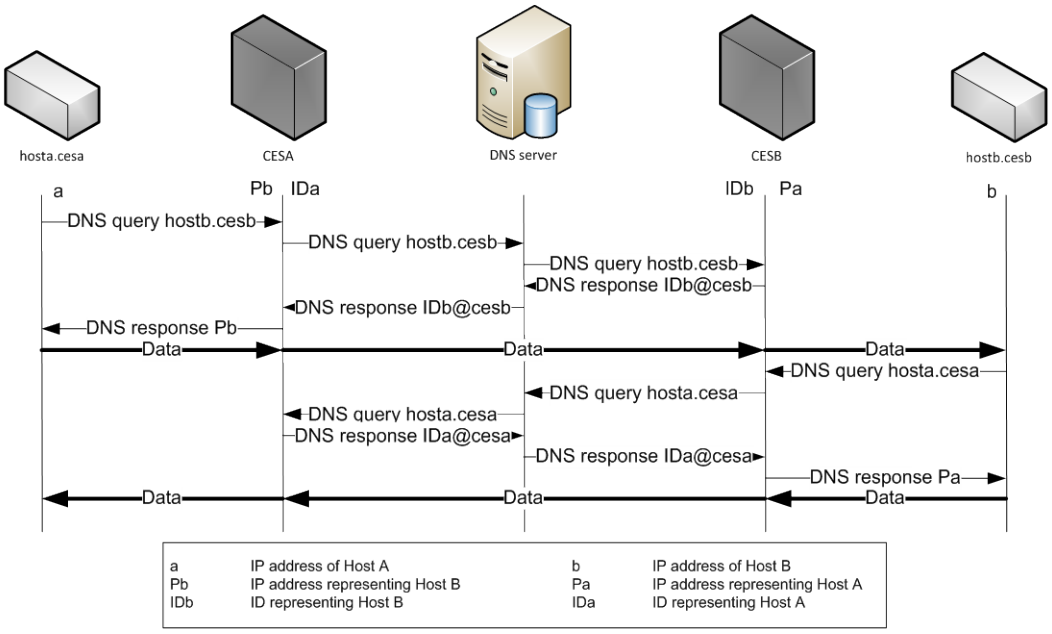

Figure 2.2: The basic idea of CES-to-CES communication

In the Figure 2.2 Host_A wants to contact Host_B located in another network. DNS query is first sent to CES_A, because CES acts as default DNS server for the hosts that it is serving. CES_A checks that the destination requested in the query is not located in the same private network. CES_A then forwards the query to the DNS server which further forwards the query to the CES_B. CES_B sees that Host_B is located in the same private network. CES_B allocates an address in the private side and creates a mapping with that address and with the ID of the Host_B. A DNS response is transferred to CES_A that also creates a mapping, but with ID of Host_A and with newly allocated private address. CES_A then informs Host_A that Host_B can be reached via this proxy address. After this, Host_A can start sending data towards Host_B.

Host_B can perform a DNS query to check if the other host is actually the one that he claims. The query is processed in the same way. The only difference is that there is no need for additional mappings as there has been earlier communication between these two hosts. Usually, these kinds of queries are not performed. However, they can be used to increase security.

2.4 Public

Most methods in the current prototype, regarding communication with public addresses, are based on the work by Jesus Llorente [10]. The SIP Application Layer Gateway, presented later in the Chapter 6, naturally uses these when the correspondent host is located in the public network. The main idea behind communication between private and public hosts is visualized in the Figure 2.3.

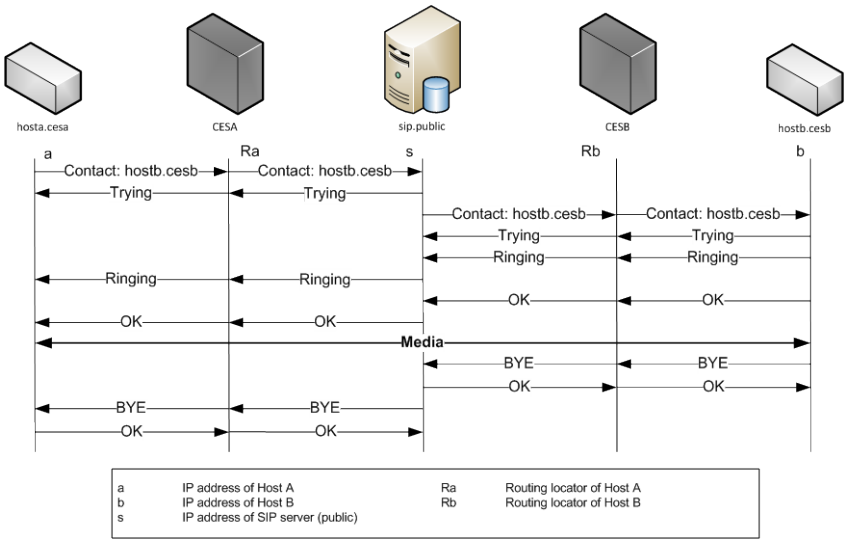

Figure 2.3: The basic idea of public communication

Figure 2.3 shows that a SIP server is located in the public network and therefore has a public IP address. Two clients that are located in different private networks have registered to the server. For clarity, the figure does not go too much into details. Examples of SIP test scenarios can be found in Chapter 6.

As the public address of the server is commonly known the client located in CES_A network can try to contact Host_B by sending a request straight towards the server's address. Because the addresses of the private hosts are still hidden, the routing locator information is used to reach the hosts behind CES devices [3]. In this case they are IP

addresses allocated from the public address pool. The CES also has mapping information, so it knows what host the specific dynamic public address represents.

The server sends the request towards the Host_B. The address representing this host is similar to the one of the originator. Thus, the CES_B is able to send the request to the Host_B. After additional signaling the media traffic flows between the two hosts.

2.5 Private

The original version of the prototype, which was created by Lauri Virtanen [23], did not provide support for private local connections. These algorithms were implemented as a part of this thesis to be able to test various different scenarios presented later in the thesis.

On a general level, the local traffic is first detected as outbound. Then the actual destination address of the packet dictates, if the packet is indeed destined to the private network. In this case the packet is sent to a Local Private Network (LPN). However, when the packet's destination does not belong to an LPN, it is processed as a normal outbound packet, similar to the CES-to-CES outbound. We have several Local Private Networks and they can be reused as private networks.

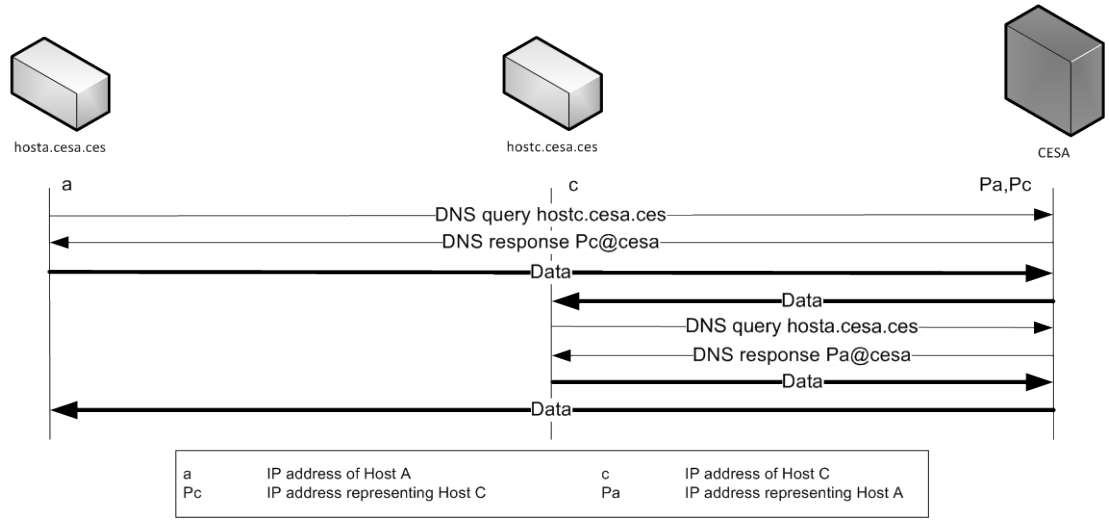

Figure 2.4: The basic idea of private communication

Figure 2.4 shows how traffic flows between two hosts located behind the same CES device. Initially the hosts have no information about each other. In the example case

Host_A begins the communication by sending a DNS query regarding Host_C. The prototype allocates addresses for the both sides, one for the originator and one for the destination. These addresses are used to create a mapping so the hosts can reach each other via these proxy addresses. After this the CES replies to the query with a DNS response telling the host that the destination can be reached through the previously allocated proxy address.

Usually the receiver does not need to perform an additional DNS query. The hosts can send data both ways after the first DNS response. However, this kind of process can happen if the Host_C wants to check that the sender really is who it claims to be.

3. Network Address Translation and Application Layer Gateways (NAT traversal)

This chapter describes how NAT works, why it is used and also describes the problems introduced by the Network Address Translation. These problems can prevent the usage of some higher level protocols. After presenting the problems, the chapter introduces certain ways to prevent or minimize these problems. The existing solutions and Application Layer Gateways are studied at the end of the chapter.

3.1 Network Address Translation

The basic idea behind Network Address Translation is to map IP addresses related to the hosts in one network to different IP addresses representing the hosts in another network. Processing should be transparent to the users. Two goals of using NATs are: alleviating the address shortage and protecting a private network from incoming traffic. [21] IANA has allocated three address spaces for private addressing: 10/8, 172.16/12 and 192.168/16. [15]

3.2 Problems with Network Address Translation

It is common knowledge that Network Address Translations create problems with the protocols transferring a packet that carries ports and addresses in the payload of the packet. As an example, if the translation involves Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) and the addresses or the ports are changed, the checksums located in the headers need to be updated with proper values matching the translated addresses and ports. [1]

Address translation is sometimes application specific. Thus, Application Layer Gateway (ALG) is required for individual applications. Naturally the payload must not be encoded to enable alterations in the payload. [21]

3.2 Existing Solutions

Interactive Connectivity Establishment (ICE) provides one way to deal with NATs. The good thing about ICE is the fact that it does not need any UDP relay. ICE is

designed for NAT traversal not for a specific protocol working on the application layer. ICE allows two hosts, called agents, to communicate by using some signaling protocol, like Session Initiation Protocol. ICE does not need to know the properties of the NAT. [16]

Session Traversal Utilities for Network Address Translators (STUN) offers some help, but it does not fix everything. First it requires that clients support STUN and a specific STUN server is located in the public network. STUN functions as a basic client-server protocol. ICE is a usage of STUN. [19]

Traversal Using Relays around Network Address Translators (TURN) allows clients to request another client to act as a server relaying packets from other hosts. These hosts can be also called peers. A client obtains an IP address and port located on the server. This relayed transport address is advertised to the peers so that they can contact the client via that address. [17]

Our new approach attempts to replace the NATs by a device called Customer Edge Switch (CES). In this solution the private addresses are not published. Instead users in the private address space are identified by using IDs. In the current prototype the length of the IDs is four octets. This is decided to be enough for the testing purposes. The Customer Edge Traversal Protocol (CETP) supports many types of IDs up-to 127 octets in length. [8]

3.4 Application Layer Gateways

As previously mentioned in this Chapter, a NAT does not offer support for the applications. Specific Application Layer Gateways (ALGs) are used to provide transparent operations to the application layer protocols. Protocols that carry IP addresses or TCP/UDP ports in the payload need more detailed processing. ALGs are in a sense like proxies. Both provide application specific communication between the client and the server. However, the proxies often use a specific protocol for relaying data between the clients and the servers. [22]

As an example, the Application Layer Gateway for FTP should manage both sequence and acknowledgement numbers. Also specific processing is needed when a packet containing PORT or PASV command is encountered. [21]

What comes to SIP and FTP, the behavior of our prototype resembles more an Application Layer Gateway than NAT. Each packet is processed individually before

setting up a connection and when the state exists the packets are handled by general CES mechanisms.

While the adoption of our solution would be ongoing we need to interoperate with the SIP UAs that use STUN/TURN for NAT traversal. These UAs would appear as public hosts to the hosts located behind the CES device.

4. Application Layer Protocols

This chapter presents two application layer protocols that are essential to this thesis: Session Initiation Protocol (SIP) and File Transfer Protocol (FTP). Basics about SIP are explained first. Then the chapter describes in a more detailed way how SIP works. After that the problems network address translation causes for SIP are analyzed. Examples where SIP can be used are also given. The second part of the chapter focuses on FTP, another important protocol for this thesis. In addition, this chapter specifies why understanding Transmission Control Protocol (TCP) is essential for FTP to work with CES.

4.1 Session Initiation Protocol (SIP)

This section is dedicated to description of the SIP, the functionality of the protocol itself and explanation of the used software. This helps understanding the test scenarios and the solutions later in the thesis.

4.1.1 SIP briefly

Session Initiation Protocol is an application layer protocol that usually runs over UDP. This means that information contained in the SIP message is transferred within the UDP payload. This is illustrated in the Figure 4.1. However, optionally SIP can run over TCP, but it is not as common as using UDP.

Figure 4.1: SIP packet

Hosts involved in the SIP communication are called User Agents (UAs). UAs can be split into User Agent Clients (UACs) issuing requests and to User Agent Servers (UASs) processing those requests and responding to them. The UA initiating the session and therefore acting as caller is UAC and the receiver of the call the callee is UAS. [6]

4.1.2 SIP functionality

Below a `REGISTER` message is presented. It is a SIP message without Session Description Protocol (SDP) content. Messages as this are the most basic type of the messages that can be sent by SIP. Processing information in these messages is quite straight forward. Actual algorithms are discussed later in the Chapter 6.

```
REGISTER sip:sip.cesa.ces SIP/2.0
Via: SIP/2.0/UDP 10.1.0.120:50600;rport;branch=z9hG4bKzyjjiaay
Max-Forwards: 70
To: "hosta" <sip:100@sip.cesa.ces>
From: "hosta" <sip:100@sip.cesa.ces>;tag=rvafj
Call-ID: pevwdhwscixzjvf@10.1.0.120
CSeq: 469 REGISTER
Contact: <sip:100@10.1.0.120:50600>;expires=3600
Allow:
INVITE,ACK,BYE,CANCEL,OPTIONS,PRACK,REFER,NOTIFY,SUBSCRIBE,INFO,MESSA
GE
User-Agent: Twinkle/1.1
Content-Length: 0
```

`INVITE` is a SIP message with SDP information. This The message contains the information how the parameters, related to setting up a session, are configured. Managing and changing information in the right way, in the message presented below, is one of the key aspects of this thesis.

```
INVITE sip:101@sip.cesa.cesSIP/2.0
Via: SIP/2.0/UDP 10.1.0.120:50600;rport;branch=z9hG4bKvivdxwuk
Max-Forwards: 70
To: <sip:101@sip.cesa.ces>
From: "hosta" <sip:100@sip.cesa.ces>;tag=tckbg
Call-ID: wjzwjyngyspdjwt@10.1.0.120
CSeq: 326 INVITE
Contact: <sip:100@10.1.0.120:50600>
Content-Type: application/sdp
Allow:
INVITE,ACK,BYE,CANCEL,OPTIONS,PRACK,NOTIFY,SUBSCRIBE,INFO,MESSAGE
Supported: replaces,norefersub,100rel
User-Agent: Twinkle/1.1
Content-Length: 299

v=0
o=100 806923301 155642630 IN IP4 10.1.0.120
s=-
c=IN IP4 10.1.0.120
t=0 0
m=audio 8000 RTP/AVP 97 98 8 0 3 101
a=rtpmap:97 speex/8000
a=rtpmap:98 speex/16000
```

```
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:3 GSM/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
a=ptime:20
```

The fields, represented in the previous `INVITE` message, are explained next [4] [18]:

- INVITE: contains Uniform Resource Identifier (URI) related to callee
- Via: gives information where the response should be sent
- To: states the receiver of the message
- From: states who sent the message
- Call-ID: often encrypted random ID, can contain information about host
- Contact: destination to direct following requests
- o=: the originator and identification of the session
- c=IN IP4: information about the connection
- m=audio: media type and address (port)

All of the previously analyzed information contained in the SIP message is important when processing a packet with the algorithms in SIP ALG. When a packet travels through a CES device, the local addresses need to be changed to allocated proxy addresses. These same addresses are also used in the outer layers of the packet.

Depending on the client application and on the server application, the settings related to the fields introduced earlier, might have some differences. Meaning that the possible options in which the user can affect are more limited in some softphones. The configurations of the clients/servers are explained later in this chapter.

4.1.3 Issues with Network Address Translation (NAT)

NAT was discussed in the Chapter 3. This part focuses how NAT works with Session Initiation Protocol.

As the Section 4.1.2 indicated, SIP messages have many internal addresses. Because of these addresses any changes done to outer layers of the packets can/will prevent the SIP from working. Either the packets are dropped by the UAs when the addresses

mismatch or the configuration of the session is set up in the wrong way and the RTP traffic does not flow end to end.

As an example, let us consider a case of two clients located behind different devices performing NAT operations. If the addresses are not changed by a device located at the edge of the network, the clients will try to send RTP traffic to the local (private) address of the other client. This happens if the signaling works for some reason. More specific explanation can be found for example in the thesis by Lauri Virtanen. [23]

4.1.4 SIP applications

This section gives information about the most used client and server software. It is important to know how they operate in order to understand the testing part later in the thesis. The pictures regarding the configurations in the applications can be found in Appendix C.

While doing the tests different clients were used to check that the prototype can support various client programs. However, most commonly used client software was Twinkle. Other clients that were tested are 3CXphone, Ekiga and Linphone. The purpose here is to provide enough information about how Twinkle operates. Other clients work in a similar way but they have more restricted configuration or redoing tests with them is troublesome.

The following figures show how Twinkle 1.1 can be configured. Settings that are vital for most tests are analyzed. Some aspects that are not that important are just briefly mentioned. The settings presented here, match settings that can be also configured to other client software programs. Although, the key issue is that at least some of the settings are named differently in each softphone.

Figure C.1 shows how Twinkle stores user information. Your name field can be used to show a name to other users when calling or messaging. User name is a number in this case, because 3CX phone system stores users as extension numbers. Three digit numbers are used to allow enough separation between the clients. The server actually does not allow shorter extensions.

Domain represents the local SIP domain that can be set in the server settings. This field was a bit confusing as configuring it wrong does not prevent registering to the

server. The SIP server just blocks any following messages, because they do not belong to the domain the server represents.

Authentication name and password are used when communicating with the server. 3CX phone blocks any unauthorized signaling. The SIP server is referenced here quite often, but its specifications are described more thoroughly in section 5.1.4.

Settings visualized in the Figure C.2 state where a client sends requests and register messages. The current configuration shown in the figure, with outbound proxy, means that a client sends all messages to the outbound proxy address. The outbound proxy in this case is stated as FQDN, sip.cesa.ces which specifies a dedicated SIP server. In this case the server is located in the private network behind CES_A.

Because outbound proxy is not an IP address, a client needs to query for the actual address. Query is directed to CES_A that allocates a proxy address for the client to communicate with the SIP server. DNS response contains this proxy address. The mapping including two proxy addresses, one for the client one for the server allow CES to forward traffic from the client to the server and vice versa. With this arrangement UA and the proxy can indeed belong to two different private address realms behind the same CES.

Expiry states how long the registration is valid. In the tests, described in this thesis, the expiry is not so important. One test takes usually less than five minutes to perform and capture. Thus, long registration is not needed, but it does not affect the tests either. Registration at startup is disabled, as a client registering automatically is often harmful. If the test environment is not ready, meaning that CES prototype is not running, when client tries to perform DNS query, the client will freeze as long as the DNS query times out.

Under RTP options a user can choose between various codecs that Twinkle tries to use. Figure C.3 illustrates this aspect of the client. Preference of the codec is defined by how high it is listed in the active codecs. How these codecs are visible in the SIP messages is explained earlier in Section 4.1.2.

For the test scenarios explained in this thesis, the differences between the codecs were not an important issue. The actual functionality of the prototype was more important than checking which of the codecs could provide better performance than other.

However, Twinkle remains as a viable client for possible future testing because it allows easy and fast switching from one configuration to another also in this aspect. Other clients did either not provide this option or if they did, it was far more restricted.

Twinkle also provides an option to have many different user profiles, as can be seen in the Figure C.4. As other configurations before this were all user specific, it is really positive to be able to configure user profiles to match different test scenarios and then just run proper profile for each test. While this may not sound as much of a feature it really speeds up redoing tests or switching between the tests. Again, the difference between Twinkle and the other softphones is significant.

A user can state the desired IP address or network interface for the Twinkle 1.1. This option can be useful when clients have more than one interface and there is a possibility that Twinkle can startup by using the wrong one. Default can be set by IP address or actual network interface.

If the client machine is running more than one interface, it is possible to run scripts in the client machine before initiating Twinkle. The scripts actually just make setting up a test scenario faster. Running a script usually takes down one or two interfaces. This forces Twinkle to use the desired interface. Machines running client software might be connected to the Internet for updates and software installation. These scripts can disable the interface that is not needed for the specific test.

Network settings for Twinkle are shown in the Figure C.5. It clarifies that Twinkle tries to run SIP and RTP in the ports that they are specified to be run in. SIP specifications were explained in Section 2.1. An option to force Twinkle to use a certain port proved to be a valuable possibility. The prototype allows softphones to use IP addresses or Fully Qualified Domain Names.

Currently processing of the messages is triggered by application type and port. This means that when a softphone sends `REGISTER` via port 5060, the prototype uses algorithms designed to work with IP addresses. However, if the client uses port 50600, the prototype detects it as message that requires different processing and it uses algorithms dedicated to the FQDN approach that maximizes user names for identification. In the future, one of these methods might become better suited for various scenarios, but the current version supports both.

The main window of the Twinkle softphone is presented in the Figure C.6. All the previous analyzed settings can be found under the Edit tab. Buddy list shows that in addition to several profiles it is possible to add several contacts. In the same way this can speed up the testing. The number of the callee, the receiver of the call, can be typed in manually in the call field. First number dictates the extension number which points directly to a certain user. The desired server is defined after this.

As mentioned earlier, automatic registration is disabled. Thus, clients need to register to the server via register action found under registration tab. Deregistration from the server is also important if similar tests are run subsequently. This removes the client from the connected user list in the server's extension status.

Twinkle itself understands FQDN addresses and performs a DNS query whenever it receives a packet with FQDN instead of IP address. However, the softphone also behaves in conservative way as it does not send SIP packets with FQDN at all. It always uses IP addresses in the SDP content.

One setting that proved to cause quite much problems, was to disallow an extension to be used outside of the Local Area Network (LAN). It can be found in 3CX settings by first selecting an extension. This brings up the window presented in the Figure C.7. Under the tab labeled as "Other", there is an option to enable or disable this property. When the extension is not allowed to be used outside LAN, it cannot register to the server. Settings listed under the general tab need to match the settings in the client.

TABLE 4.1 ADDRESSING OF THE CLIENTS IN THE SIP SERVER

User name	hosta	hostb	hostc	hostd
Extension number	100	101	102	103

Table 4.1 shows how the addressing of the extensions is done in the test scenarios. These settings are required to be the same in the client and in the server. In addition to normal calls it is possible to connect two servers via trunks. If there is a SIP server in the CES_A network and a SIP server in the CES_B network, the users can register to these servers locally and still be able to reach hosts in the other network by using a trunk that connects the servers together. When using a trunk, both data and signaling is transferred through a SIP server.

TABLE 4.2 TRUNK CONNECTIONS IN CESA

Trunk name	SIP CES_B(DNS)	SIP CES_B(static)	SIP public	SIP CES_B
Extension number	2xxx	6xxx	4xxx	8xxx

In the Table 4.2, the trunks in the server located behind CES-A, are listed. When the server uses a trunk connection, the trunk has to be configured in both ends. If the client calls with one of these prefixes, the server uses the trunk to forward the message straight to the other server specified by the prefix. Specific users are again defined by the extension numbers which were presented in the Table 4.1.

The server can be set to run in a specific port. As a default, it runs in the port 5060. The value of the port can be found under network settings. However, when testing FQDN algorithms, the server is running in the port 50600. The algorithms in the prototype are created in such a way that a specific port can trigger different processing. Because of this, it is relatively fast to switch from one scenario to another.

Under advanced settings the server must have the same domain as the clients. Otherwise the server does not allow the clients to make any calls. Registration will anyway succeed. The same setting tab has also an option to allow calls to external SIP URIs. This was enabled to prevent potential problems.

When processing SIP messages with FQDNs the 3CX server performs in the same way as Twinkle. It also performs a DNS query after receiving a packet with FQDN information. Still it does not sent SIP packets with FQDNs itself. Thus, maintaining conservative style.

3CX contains already some anti-hacking mechanisms that can prevent potential DoS attacks. These include failed authentication requests and unanswered challenge requests. Source of malicious traffic can be blacklisted, if behavior fulfills the values set in the server.

4.2 File Transfer Protocol (FTP)

This section is dedicated to detailed explanation of the FTP protocol. File Transfer Protocol is closely related to the Transmission Control Protocol (TCP) and therefore it is also analyzed.

4.2.1 FTP over TCP

File Transfer Protocol provides a reliable way to transfer data between hosts. Before going more into detail about FTP, some clarifications about Transmission Control Protocol are presented. Thus, understanding FTP, running on top of TCP is easier. Structure of the FTP packet is shown in the Figure 4.2.

Figure 4.2: FTP packet

Reliability of the transfers is ensured by Transmission Control Protocol. TCP is designed to fit multi-layered hierarchy of protocols and it provides reliable end-to-end communication. The receiver uses the sequence numbers to order the packets and to remove the duplicates. The sender retransmits the packet, if the ACK is not received within a predetermined time window. [5]

Acknowledgement Number dictates, the next Sequence number, the sender is expecting to receive. This is shown in Figure 4.3.

$$Seq_{sent} = Ack_{last\ received}$$

Figure 4.3: Calculation of the new Sequence number

Next Acknowledgement number is the sum of Sequence number and the length of the header with data included. This is visualized by Figure 4.4.

$$Ack_{sent} = Seq_{last\ received} + Length_{last\ segment}$$

Figure 4.4: Calculation of the new Acknowledgement number

As an ALG is changing the addresses it can cause changes in the length. Addresses are in the string format and therefore it is easy to cause differences whenever they are changed. These differences need to be considered every time the CES is changing the addresses. This is discussed later in the thesis when ALG for FTP is analysed.

File Transfer Protocol tries to achieve four goals that are: contribute to file sharing, encourage the use of remote machines, minimize the effect of the differences in the storage systems and transfer data in a reliable and efficient way. Commands used in the FTP connection specify the parameters. In a similar way, the aspects of the file system can be determined. [14]

FTP creates two TCP connections: one for the control and one for the data. A TCP session is identified by the source and destination addresses and ports. An FTP connection can be either active or passive. In active mode the client announces the address where it wants to receive the data. This is done by sending PORT command. In the passive mode the client requests an address from the server with PASV command. Server responds with the address that can be used to get the data. This means that the server is either sending the data to a certain address or allowing the host to access it via a certain address. [14]

4.2.2 FTP functionality

The following capture from the command prompt presents how the communication works between two hosts when using FTP. List of files or actual data is not included to maintain clarity. The client commands can be seen in the lines starting with “ftp>”.

```
tester@hardy2:~$ ftp hardy4.cesa.ces
Connected to hardy4.cesa.ces.
220 (vsFTPd 2.0.6)
Name (hardy4.cesa.ces:tester):
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing. ...
226 Directory send OK.
ftp> passive
Passive mode on.
ftp> ls
227 Entering Passive Mode (10,1,1,10,113,24).
150 Here comes the directory listing. ...
226 Directory send OK.
ftp> passive
Passive mode off.
```

```
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing. ...
226 Directory send OK.
ftp> exit
221 Goodbye.
```

The test scenarios analyzed later in the thesis follow the same commands structure as previously shown. These test scenarios include local and CES-to-CES communication. More information related to the tests can be found in the Chapters 7, 8 and 9.

4.2.3 Issues with Network Address Translation (NAT)

Network Address Translation breaks FTP because the IP addresses, contained in the messages “Request: Port” (active) and “Response: 227” (passive), are not changed. This causes a mismatch with the IP addresses in the outer header and IP addresses in the payload of the IP packet, preventing the communication between the hosts.

Following aspects also introduce challenges: the new sequence number is defined by the old acknowledgement number and the new acknowledgement number is the sum of the old length and the old sequence number. Because the addresses are handled in the string format rather than octets by the prototype, any changes in the addresses and therefore in the length of the packet may cause problems. These issues need to be considered when designing the Application Layer Gateway for the FTP.

4.2.4 FTP applications

The FTP client installed in both, physical and virtual machines, is just a basic user interface for File Transfer Protocol. This program enables the transmission of data to the remote hosts or from the remote hosts. The program is run via command line as seen earlier in this chapter.

5. Requirements for Application Layer Gateways

This chapter specifies the requirements that were set when prototyping ALGs with CES began. The chosen solutions are also implemented in the actual prototype. In addition, this chapter explains why certain applications are used with the prototype testing. The first part discusses SIP and the second part of the chapter explains requirements when implementing FTP for CES.

5.1 SIP Application Layer Gateway

The first requirement for SIP ALG was that a client must be able to register to a server. Initially, the server and the client were placed in the same network. This helped checking, how the configurations in the software should be reflected to the messages processed by the prototype. This also narrowed down the options between different software to be used as the client or as the server.

The second requirement was related to the communication with the public network. The target was to be able to register to a server located on the opposite side of a CES device. The client can be located in either the local or the public network, but then the server is placed in another network. After registration has succeeded, the testing continued with SIP MESSAGEs. This can ensure that changing the addresses described in the Chapter 4 works properly.

After this, the public network part continued with the processing of the SIP packets that contain SDP information. As explained in the Chapter 4, the packets with an SDP part require more specific processing, like allocating addresses/ports for RTP traffic.

After establishing a connection for RTP traffic in both directions, it was important to be able to terminate the call in the right manner. This means that all the messages required to end the call are transferred from one client to the server and then to the other client.

When one part of the solution was implemented, the algorithms related to different scenarios were started. These algorithms supported private network and CES-to-CES communication. Later, the algorithms using FQDNs instead of IP addresses were added to have two different options for message processing in every scenario. As a Fully Qualified Domain Name is one way to distinguish users, it can also be used as an identity. [8]

The last important issue in the SIP ALG design was related to allocating resources. Depending on the test scenario, the resources needed for the communication can be allocated in a different phase of the SIP message processing. It is also important to release these addresses that are not used anymore.

5.2 FTP Application Layer Gateway

As a thesis objective, the FTP connections were required in the CES-to-CES communication and in the local communication. More specific information about FTP was given in the Chapter 4.

First requirement was that a host acting as a client is able to log in to the server. The order in the development of the algorithms was from public network communication to the private network communication and finally to the CES-to-CES communication.

After the login was successful, the client needed to be able to download a file from the server. Performance was not the big issue in this thesis, so the transfer times are not recorded or analyzed.

Switching the transfer from active to passive and vice-versa, was an important feature. However, implementing it proved to be rather challenging as the aspects of the TCP, Seq/Ack, required specific processing to maintain a continuous signal flow.

At the end, the prototype was tested for simultaneous connections. This means that there can be more than one client communicating with the server at the same time. Transferring data, between the client and the server, needed to work also with parallel connections.

5.3 Design choices

When the Application Layer Gateways were designed it was decided that all of the traffic in the private network should flow through the CES device. This approach is similar to the solutions in the current mobile networks. It also improves the security of an individual user as the users are hidden from each other. However, the analysis of security is not an important aspect in this thesis.

The chosen approach naturally increases the amount of transferred packets. Another way to do this would be to transfer packets directly from a host to another host

whenever possible. While this could be potentially a better solution in the short run, we argue that our design is more suitable for the future research. Our approach allows placing many private address realms behind one CES.

Any addresses located in the public network are left unchanged. This makes it easier to change from current edge devices to the CES approach without any additional requirements. Related to this, both of the Application Layer Gateways presented in this thesis are transparent to the end user.

The differences between the implemented version and the potential approach are discussed after example runs in Chapter 6 and also in Chapter 7. This is done to provide detailed comparison between the choices.

6. SIP Application Layer Gateway

This chapter describes the design of the ALG that we implemented in the CES prototype to allow it to work with SIP. At this stage the prototype can support various methods of SIP communication including CES-to-CES, public, local and trunk.

6.1 SIP ALG Briefly

When the application layer in the CES prototype receives a packet, it checks what protocol it could match. In addition, the source and the destination of the packet give information whether the packet is inbound, outbound or local. If the protocol happens to be SIP, recognized by UDP being sent to ports 5060 or 50600, the correct algorithm within the SIP ALG is called. For testing purposes the division between IPs and FQDNs are done with port numbers. This means that IP addresses use port 5060 and FQDNs use port 50600. Next, the Application Layer Gateway that the prototype now provides for SIP is explained in more detail.

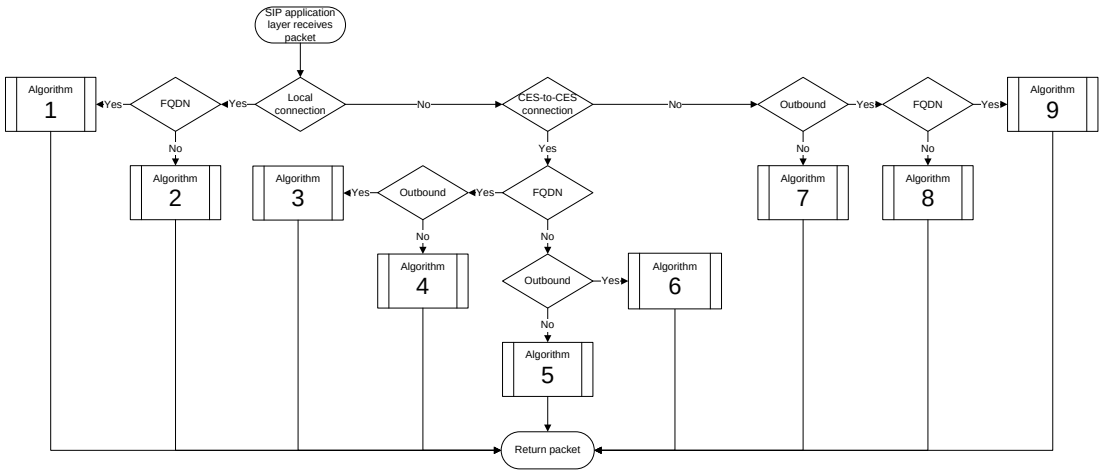

Figure 6.1: SIP application layer gateway functionality

Figure 6.1 visualizes how different algorithms in the prototype get called. Depending on the packet's destination, the packet can be classified as a local, meaning that a packet is transferred inside a Private Local Network, or a non-local packet. Local packets can be further divided into packets that use IP addresses and packets that use

FQDNs. This determines if the local IP addresses, source IP and RTP IP, within SIP messages, are replaced by the proxy IPs or with the FQDN of the host.

Other packets can be divided to packets belonging to public communication, meaning communication with a host or a server located in a public network, or CES-to-CES communication. Both of these categories are split in a similar way as local packets. By these divisions we get nine different algorithms that can be seen in the Table 6.1. The numbers shown in the table match the numbers in figure 6.1. Each of the algorithms gets analyzed more thoroughly later in the chapter.

TABLE 6.1 SIP ALG ALGORITHMS IN MORE DETAIL

Number	Algorithm name and type	Functionality
1	Local-FQDN	Packets inside private network
2	Local-IP	Packets inside private network
3	CES-to-CES-out-FQDN	Packets between CES devices
4	CES-to-CES-in-FQDN	Packets between CES devices
5	CES-to-CES-in-IP	Packets between CES devices
6	CES-to-CES-out-IP	Packets between CES devices
7	Public-in-IP	Packets from public network
8	Public-out-IP	Packets to public network
9	Public-out-FQDN	Packets to public network

Example runs, presented later in this chapter, are made by using both versions of the algorithms. The FQDN versions get more focus as they perform better than the algorithms using IP addresses. These algorithms are easier to understand than the IP versions. They also provide a more flexible approach to the problem dealing with SIP messages with CES. Algorithms that use IP addresses are still explained.

The prototype uses a DNS server to direct DNS queries. However, the DNS server is not visible in any of the example runs to maintain similar style among the figures. Queries, that would have been directed first to the server and then to either CES device, are visualized just with the queries to the right CES device. The actual structures in the test environment are explained in the Chapter 9.

TABLE 6.2 SIP TEST SCENARIOS ANALYZED IN MORE DETAIL

Test scenario	Caller	Callee	Server	Algorithm type
First public scenario	CES_A	CES_B	Public	IP
Second public scenario	Public	CES_A	CES_A	IP
Third public scenario	CES_A	Public	CES_A	FQDN/IP
Private network scenario	CES_A	CES_A	CES_A	FQDN
CES-to-CES scenario	CES_A	CES_B	CES_A	FQDN
Trunk scenario	CES_A	CES_B	CES_A & CES_B	FQDN

The test scenarios shown in the table 6.2 are presented next. These scenarios were chosen to give examples of different settings. The total of 48 scenarios, which were actually tested are presented later in the thesis.

It is important to notice that the CES device is configured with a setting file when it is started. The setting file has information about the hosts located behind the CES. The SIP server is also given a premium rate address, with static mapping, that can be used to contact the server from the public network. As some of the scenarios involve CES-to-CES communication, the file has preset ID for every host behind the CES device. Currently there is no specific policy for handling the IDs.

6.2 Communication with the public network

This section focuses on explaining how the three algorithms dedicated to communication with the public network actually work. The public user can use either IP or FQDN as an IP address. A different algorithm is required for outbound traffic depending on the situation. The same inbound algorithm can be used in both cases.

The algorithms supporting public communication with IP addresses were implemented very first when the prototype was modified. Implementing these algorithms gave quite a good idea how the addresses and ports, contained in the SIP messages, should be changed. The port numbers were quite problematic as they are only required in the public approach. The prototype follows the SIP specifications and RTP is run on an even port and RTCP is run on the next odd port. Example ports for RTP and RTCP are 8000 and 8001.

In addition to the algorithms explained in this chapter, the SIP class contains smaller functions that help with the processing of the messages and keep the main algorithms clearer. These functions perform tasks including replacing old values with new ones and creating the new RTP/RTCP map. The RTP map means addresses and ports that

are allocated and mapped for the real-time traffic. In the same way resources are allocated and mapped for real-time signalling using RTCP. Individual algorithms regarding the SIP message processing in public are represented next.

6.2.1 ALG algorithms for public communication

Figure 6.2 shows how the Public-out-IP algorithm in the SIP class of the prototype actually processes the packet. The algorithm first checks if the packet contains any SDP information. As explained in Chapter 4, this means information required to set up the RTP connections.

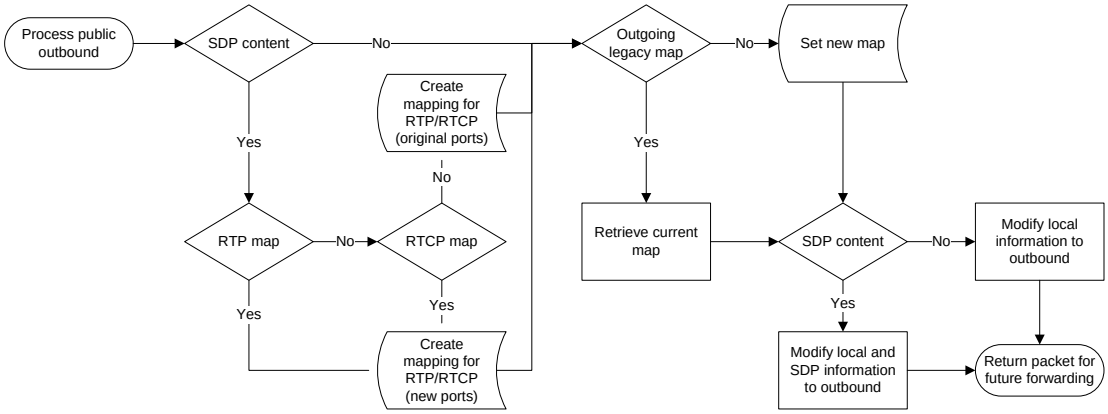

Figure 6.2: SIP ALG Public-out-IP algorithm

Assuming the packet contains an SDP part, the algorithm first checks if the original ports, usually these are 8000 for the RTP and 8001 for the RTCP, suggested in the SDP are available. If they are both free, the algorithm creates a mapping for these ports. However, if either of the ports is taken, the algorithm chooses new ports, even and odd, to be used for the mappings. RTP mapping is used to transfer media between the hosts and RTCP mapping is used to transfer control information.

After this, the algorithm checks for an existing outgoing map. If there has already been a communication between the source and the destination it reuses the mapping. Otherwise a new mapping is created between the source and the destination. Outgoing map dictates how the address from the private network is translated to the address in the public network. The host in the private network is reachable from the public network by using this outbound address.

The last part of the algorithm sets the new information into the packet. This means changing the local information to the outbound information according to the previously created mappings. Another function is used to change all desired values. As an example: changing local IP to outbound IP; local port to outbound port; RTP IP to new RTP IP and RTP port to new RTP port.

Recalculating the length of the packet is also important as the length of the new addresses and ports can be different from the original ones. If the actual length does not match with the value informed by the content length field, it is possible that the packet gets dropped by the receiver. The function changing the values stores the original length of the packet and calculates how long the new packet is. By doing this, it is possible to replace the old value with the new one.

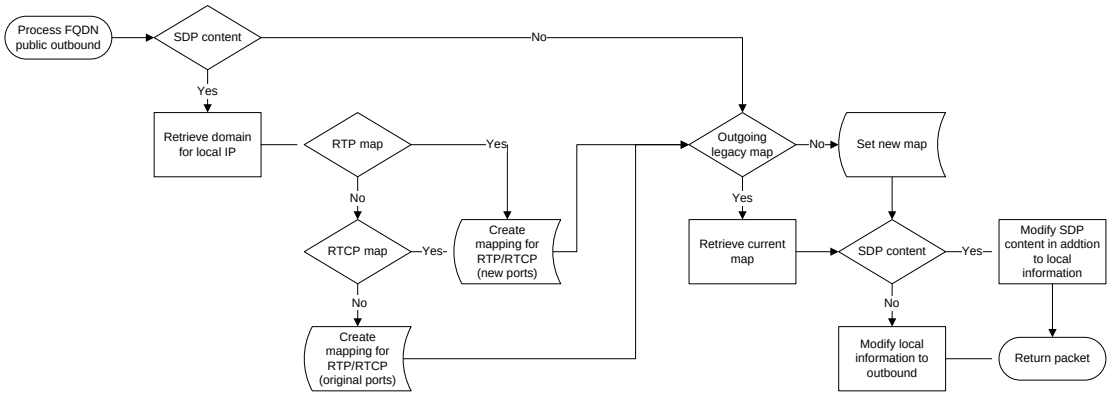

Figure 6.3: SIP ALG Public-out-FQDN algorithm

The Public-out-FQDN algorithm that is illustrated in the Figure 6.3 performs in quite a similar way as the previous public algorithm using IP addresses. Actually the only difference is that RTP IP is not replaced by another IP representing the proxy address. FQDN is used instead. The algorithm retrieves the FQDN matching the local IP. After this the local IPs are replaced by the FQDN. This allows the receiver of the call to perform a DNS query which allocates the proxy addresses and creates a mapping in the receiver side. Because the sender side already has a mapping, the receiver will get the correct proxy address as a result of the DNS query.

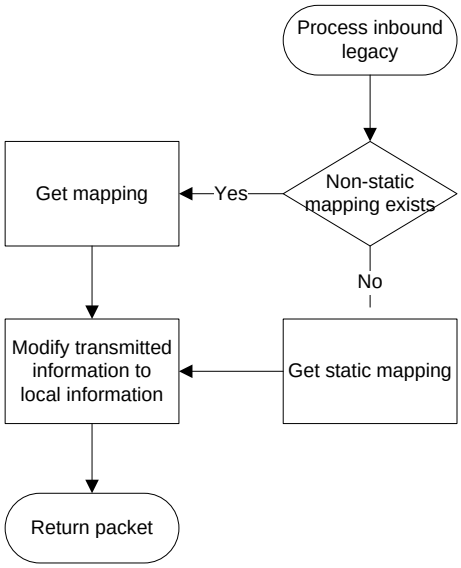

Figure 6.4: SIP ALG Public-in algorithm

The public-in algorithm is one of the simplest algorithms in the SIP ALG. It first checks if there is a static mapping. This means checking for preconfigured mappings. These are for example available when the CES has been configured with a public IP address that can be used to reach the SIP server. Static mappings can be used also for other services/protocols that might need specific processing.

Only the SIP servers are available to be directly connected from the public network. If there is inbound RTP traffic, the client has communicated with other party earlier. The first `INVITE` initiating the call is transferred via the server. Thus, a mapping for RTP traffic is created when the packet travels in the private network.

6.2.2 Example runs related to public communication

The first scenario represents the most likely setting for a SIP session when using CES devices. The main purpose of this test is to show that the prototype can support similar scenarios as the traditional NATs. This kind of situation is possible when using the services of a public service provider. The servers are located in the public network, while the clients are in a private network.

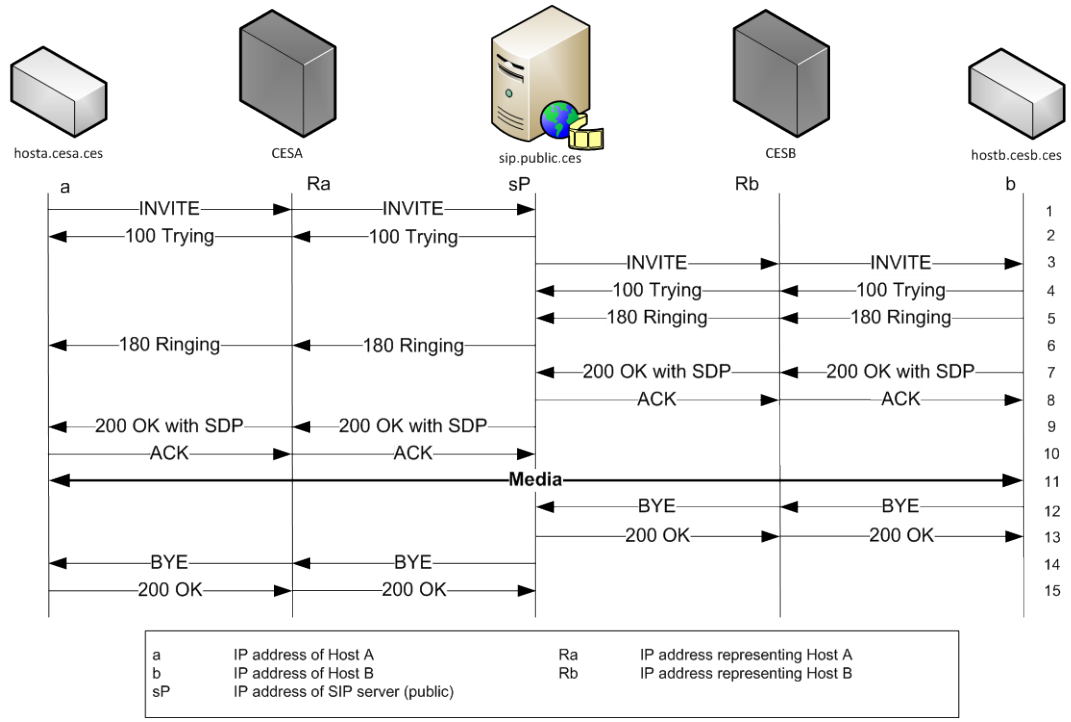

Figure 6.5: SIP ALG with public SIP server

The explanation of the messages shown in the Figure 6.5 is presented next. Host_A located in the CES_A network initiates the call and therefore acts as the caller. The callee of the call, Host_B in CES_B network, terminates the call. Both clients are already registered to the SIP server.

1. Host_A sends `INVITE` to initiate a call with Host_B towards the server (s). The message contains the information: ”`INVITE sip:101@sip.public.ces SIP/2.0, Via: SIP/2.0/UDP Ra:5060, c=IN IP4 Ra`”. Where Ra is routing locator of CES_A. The ALG inserts it in place of the IPv4 address of Host_A.
2. Server responds with `100 Trying` (Ra).
3. Server sends `INVITE` towards Host_B (Rb). CES_B forwards `INVITE` to Host_B. The message contains the information: “`INVITE sip:101@sip.public.ces SIP/2.0, Via: SIP/2.0/UDP sip.public.ces:5060, c=IN IP4 Ra`”. The algorithms that CES uses for dynamic allocation of routing locators for inbound CES traffic are explained in more detail in the work by Jesus. [10]
4. Host_B sends `100 Trying` towards the server.
5. Host_B sends `180 Ringing` towards the server.
6. Server sends `180 Ringing` towards Host_A.

7. Host_B sends `200 OK` with SDP towards the server (s). The message contains the information: "`SIP/2.0 200 OK, Via: SIP/2.0/UDP sip.public.ces:5060, c=IN IP4 Rb`".
8. Server responds with `ACK`.
9. Server sends `200 OK` with SDP towards Host_A (Ra). The message contains the information: "`SIP/2.0 200 OK, Via: SIP/2.0/UDP sip.public.ces:5060, c=IN IP4 Rb`".
10. Host_A responds with `ACK`.
11. RTP traffic flows between the hosts.
12. Host_B sends `BYE` towards the server (s). CES_B forwards it.
13. Server responds with `200 OK`.
14. Server sends `BYE` towards Host_A.
15. Host_A responds with `200 OK`.

Naturally both of the CES devices maintain their own NAT tables. Allocated addresses and mappings related to the previous flow are presented in the Tables 6.3 and 6.4. The first table shows the mappings in the CES device at the edge of CES_A network and the second one depicts the situation in CES_B.

TABLE 6.3 SIP PUBLIC-PRIVATE CONNECTION CESA

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
a	8000	Ra	8000	Rb	8000	UDP	60	A
a	8001	Ra	8001	Rb	8001	UDP	60	A
a	5060	Ra	5060	sP	5060	UDP	3600	A

TABLE 6.4 SIP PUBLIC-PRIVATE CONNECTION CESB

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
b	8000	Rb	8000	Ra	8000	UDP	60	A
b	8001	Rb	8001	Ra	8001	UDP	60	A
b	5060	Rb	5060	sP	5060	UDP	3600	A

As this test was done to show that the prototype and the ALG can provide the same functionality than current NATs, there are no specific tricks involved.

In the second test the server was located behind a CES device. This could be seen as a corporate network. The corporate has its own SIP server located in the private network. The employees can connect to the server from both, the public or the private network.

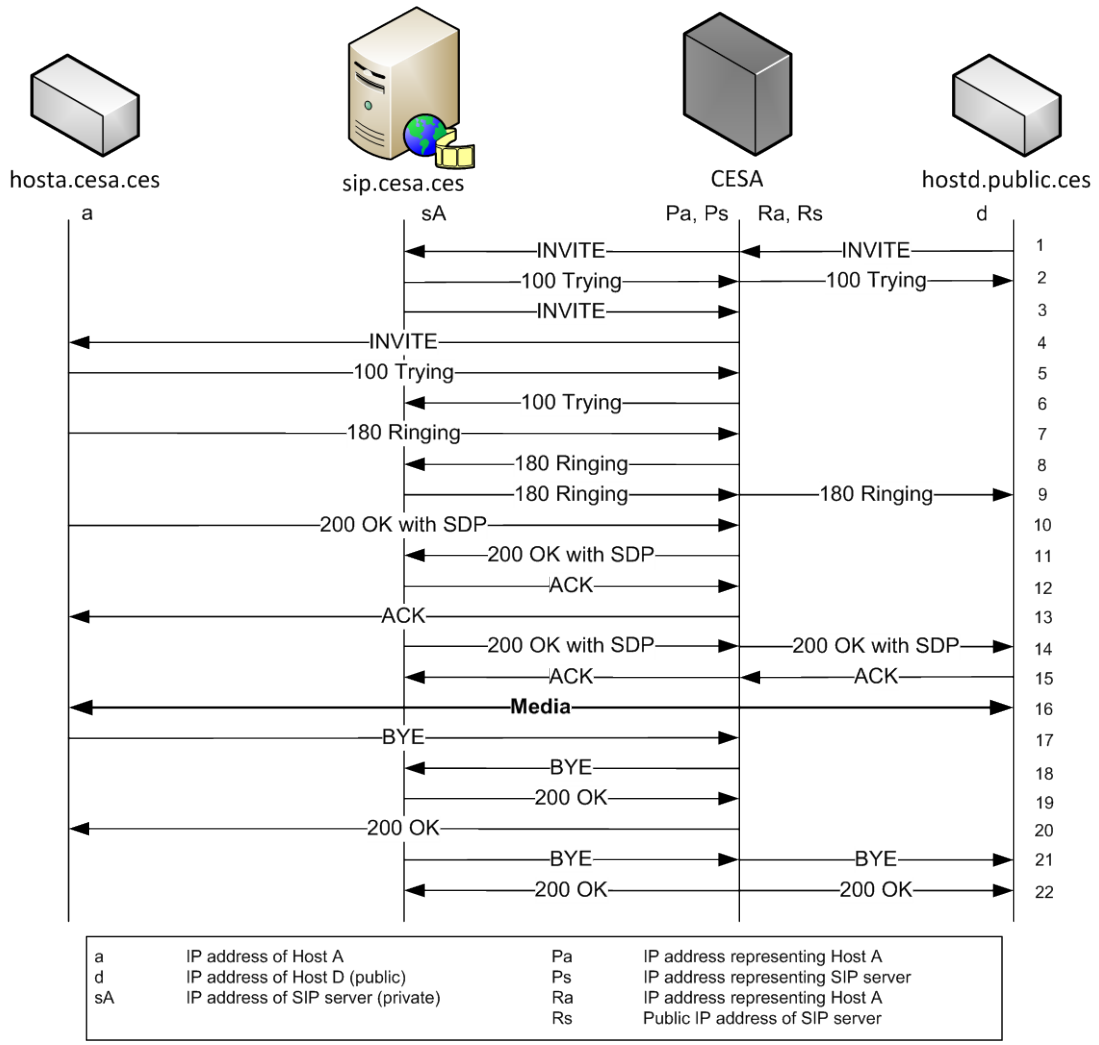

Figure 6.6: SIP ALG incoming public call

In this scenario both clients have registered with the server earlier. In this test the client in the public network wants to contact the client located in the private network. Host_D acts as a caller and Host_A terminates the call. The explanation of the individual messages, shown in the Figure 6.6, follows next.

1. Host_D sends `INVITE` to initiate call with Host_A towards the server (Rs). The message contains the information: "`INVITE sip:100@sip.cesa.ces SIP/2.0, Via: SIP/2.0/UDP d:5060, c=IN IP4 d`".
2. Server responds with `100 Trying` (d).
3. The server sends `INVITE` towards Host_A (Pa). The message contains the information: "`INVITE sip:100@sip.cesa.ces SIP/2.0, Via: SIP/2.0/UDP sip.cesa.ces:5060, c=IN IP4 d`".

4. CES_A forwards `INVITE` to Host_A. The message contains the information: "`INVITE sip:100@sip.cesa.ces SIP/2.0, Via: SIP/2.0/UDP sip.cesa.ces:5060, c=IN IP4 d`".
5. Host_A sends `100 Trying` towards the server.
6. CES_A forwards `100 Trying` towards the server.
7. Host_A sends `180 Ringing` towards the server.
8. CES_A forwards `180 Ringing` towards the server.
9. Server sends `180 Ringing` towards Host_D.
10. Host_A sends `200 OK` with SDP towards the server (Ps). The message contains the information: "`SIP/2.0 200 OK, Via: SIP/2.0/UDP sip.cesa.ces:5060, c=IN IP4 a`".
11. CES_A forwards `200 OK` with SDP towards the server (s). The message contains the information: "`SIP/2.0 200 OK, Via: SIP/2.0/UDP sip.cesa.ces:5060, c=IN IP4 temporal_IP`" (not used for mapping in this case).
12. Server sends `ACK` towards Host_A.
13. CES_A forwards `ACK` towards Host_A.
14. Server sends 200 OK with SDP towards Host_D (d). CES_A forwards the message. The message contains the information: "`SIP/2.0 200 OK, Via: SIP/2.0/UDP sip.cesa.ces:5060, c=IN IP4 Ra`".
15. Host_D responds with `ACK`.
16. RTP traffic flows between the clients.
17. Host_A sends `BYE` towards the server (Ps).
18. CES_A forwards `BYE`.
19. Server sends `200 OK` towards Host_A.
20. CES_A forwards `200 OK`.
21. Server sends `BYE` towards Host_D.
22. Host_D responds with `200 OK`.

The actual addressing and allocation related to this flow can be seen in the Tables 6.5 and 6.6. The first table represents how the addresses are allocated to communicate with the client in the public network. The second one shows how the client and the server communicate in the private network.

TABLE 6.5 SIP PUBLIC-PRIVATE CONNECTION CESA

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
a	8000	Ra	8000	d	8000	UDP	60	A
a	8001	Ra	8001	d	8001	UDP	60	A
sA	5060	Rs	5060	d	5060	UDP	300	A

TABLE 6.6 SIP PRIVATE CONNECTION CESA

NAT TABLE CES-to-CES				
Source		Destination		Type
IP	Proxy IP	IP	Proxy IP	
a	Ps	sA	Pa	local
sA	Pa	a	Ps	local

If the traffic would not flow through the CES device, it would remove six packets, which is not that much compared to all of the RTP traffic. In this case, the approach that is currently implemented is really close to the situation where the hosts could communicate directly with each other in the private network.

In the third scenario, depicted in Figure Host_A residing in the CES_A network initiates the call. The callee, Host_D that is located in the public network, terminates the call. DNS queries from Host_D are normally directed first to the DNS server in the public network. Communication between the server and Host_D uses public algorithms. Local algorithms, which are used between Host_A and the server, are explained later in this chapter.

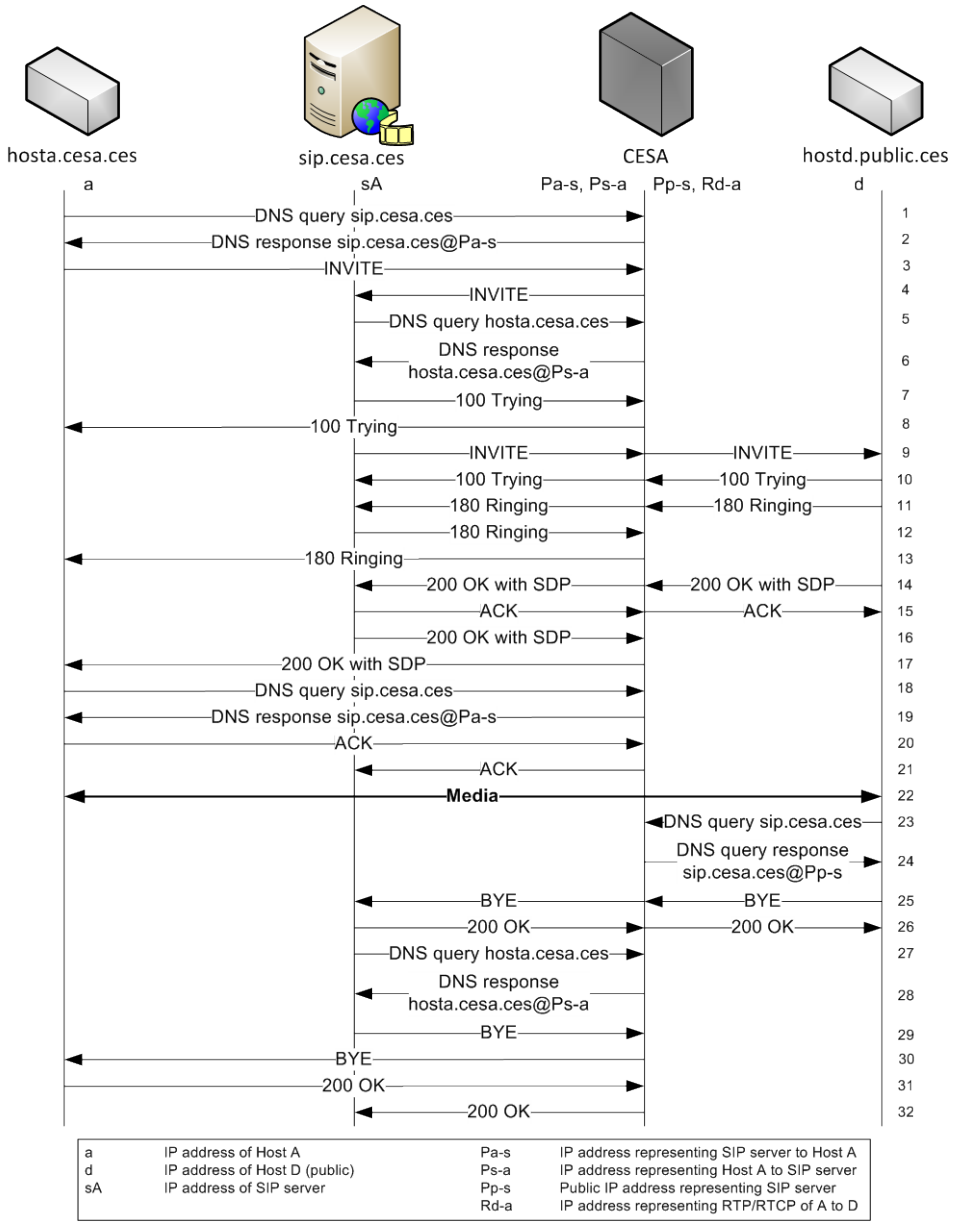

Figure 6.7: SIP ALG outgoing call with Private SIP server

The message flow needed to create a call between the local and the public client is visualized in Figure 6.7. This means that the UAC is located in the CESA network while the UAS is located in the public network. Users are registered in the SIP server before the analysis begins to minimize the number of the messages. Thus, both clients have knowledge which addresses to use to reach the SIP server. The algorithms use the domains instead of the IP addresses.

As mentioned, DNS queries from Host_D are shown as they would go straight to CES_A. Authentication performed by the SIP server is not shown. These actions are

done to ensure readability of the message flow. The extensions are the following: Host_A 100 and Host_D 103.

Clarification of the individual messages is provided next. Explanation and actual message are matched by numbers in the figure as well as in the description. DNS queries are explained only once. Identical queries later in the flow are skipped. Some additional information is provided when explaining `INVITE`s and `200 OK`s with SDP content.

1. Host_A queries for address dedicated to the SIP server (sip.cesa.ces).
2. CES_A responds with previously allocated proxy address (Pa-s).
3. Host_A sends `INVITE` to initiate a call with Host_D towards the server. The message contains the information: "`INVITE sip:103@sip.cesa.ces SIP/2.0, Via: SIP/2.0/UDP a:50600, c=IN IP4 a`".
4. CES_A forwards `INVITE` to the SIP server. The message contains the information: "`INVITE sip:103@sip.cesa.ces SIP/2.0, Via: SIP/2.0/UDP hosta.cesa.ces:50600, c=IN IP4 hosta.cesa.ces`".
5. Server queries for address dedicated to the Host_A (hosta.cesa.ces).
6. CES_A responds with previously allocated proxy address (Ps-a).
7. Server sends `100 Trying` towards Host_A.
8. CES_A forwards `100 Trying` to Host_A.
9. Server sends `INVITE` towards Host_D (d). CES_A forwards `INVITE` to Host_D. Proxy address is created for incoming RTP traffic (Rd-a). The message contains the information: "`INVITE sip:103@sip.cesa.ces SIP/2.0, Via: SIP/2.0/UDP sip.cesa.ces:50600, c=IN IP4 Rd_a`".
10. Host_D sends `100 Trying` towards the server (Pp-s). CES_A forwards it.
11. Host_D sends `180 Ringing` towards the server (Pp-s). CES_A forwards it.
12. Server sends `180 Ringing` towards Host_A.
13. CESA forwards `180 Ringing` to Host_A.
14. Host_D sends `200 OK` with SDP towards the server (Pp-s). Connection information is public IP of Host_D. CES_A forwards it. The message contains the information: "`SIP/2.0 200 OK, Via: SIP/2.0/UDP sip.cesa.ces:50600, c=IN IP4 d`".
15. Server sends `ACK` towards Host_D (d). CES_A forwards it to Host_B.

16. Server sends `200 OK` with SDP towards Host_A. The message contains the information: "`SIP/2.0 200 OK, Via: SIP/2.0/UDP sip.cesa.ces:50600, c=IN IP4 d`".
17. CES_A forwards `200 OK` with SDP to Host_A. The message contains the information: "`SIP/2.0 200 OK, Via: SIP/2.0/UDP sip.cesa.ces:50600, c=IN IP4 d`".
20. Host_A sends `ACK` towards the server (Pa-s).
21. CES_A forwards `ACK` to the SIP server.
22. RTP traffic flow between the hosts.
23. Host_D queries for address dedicated to the SIP server (sip.cesa.ces).
24. CES_A responds with dedicated public address (Pp-s).
25. Host_D sends `BYE` towards the server (Pp-s). CES_A forwards it.
26. Server sends `200 OK` towards Host_B (b). CES_A forwards it to Host_B.
29. Server sends `BYE` towards Host_A.
30. CES_A forwards `BYE` to Host_A.
31. Host_A sends `200 OK` towards the server (Pa-s).
32. CES_A forwards `200 OK` to the SIP server.

The following tables show how the addresses are allocated and also how the mappings are created in the previous flow. The communication between the hosts in private and the host in the public network is shown in the Table 6.7. Communication happening in the private network follows the approach show in the Table 6.8.

TABLE 6.7 SIP PUBLIC-PRIVATE CONNECTION CESA

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
a	8000	Rd-a	8000	d	8000	UDP	15	A
a	8001	Rd-a	8001	d	8001	UDP	15	A
sA	50600	Pp-s	50600	d	50600	UDP	300	A

TABLE 6.8 SIP PRIVATE CONNECTION CESA

NAT TABLE CES-to-CES				
Source		Destination		Type
IP	Proxy IP	IP	Proxy IP	
a	Pa-s	sA	Ps-a	local
sA	Ps-a	a	Pa-s	local

One aspect that will most likely be implemented in the future prototype is DNS caching. This can potentially reduce the number of the packets quite a lot, but in this case as the queries only involve one CES it would have no effect. If the Host_A and the server had communicated directly, it would have removed those DNS queries/responses and a couple of more messages, 15 messages to be exact. This starts to be quite a difference but still quite small when the RTP traffic is considered.

6.3 Communication in Local Private Network

The algorithm for a private network was probably the most complicated thing to solve when using IP addresses. This was because the local private algorithm is not only used for communication behind one specific CES device. As seen in the previous example it can be used as a part of public communication.

Local is also an important part of CES-to-CES and trunk communication, both are discussed later in this chapter. Local private has only two algorithms that use either IP or FQDN as addresses. Both of the algorithms are now explained in more detail.

6.3.1 ALG algorithms for Private Network

The first check in the algorithm, Figure 6.8, divides messages according to whether they have SDP content or not. SIP messages that do not have SDP content require checking if there are any unnecessary resources allocated. This happens if the caller cancels the call or the callee rejects it. In this case resources allocated by the other parts of the algorithm are released back to the pool. This prevents consuming resources needlessly.

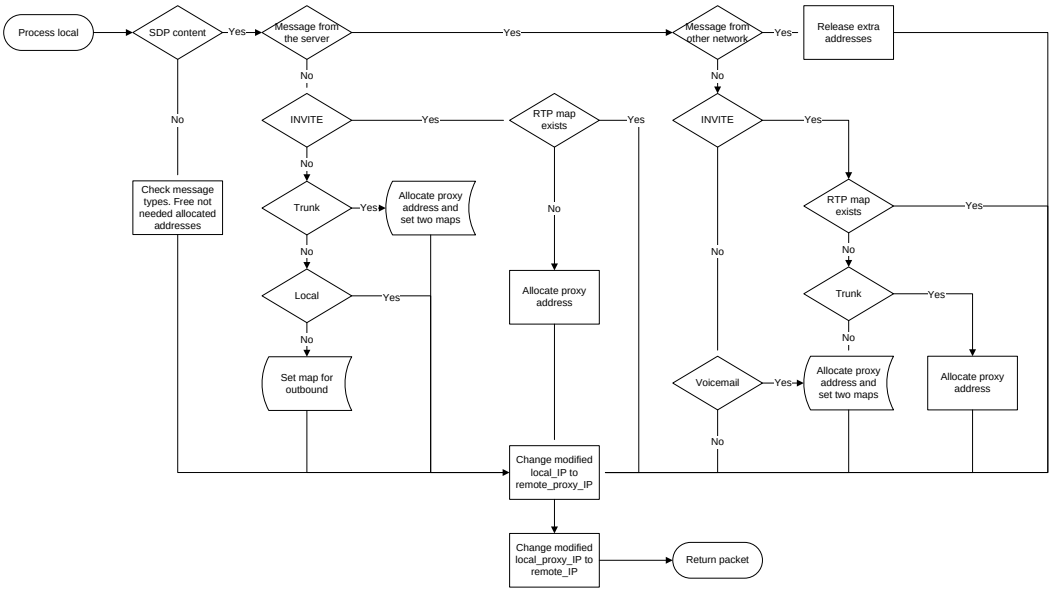

Figure 6.8: SIP ALG Local-IP algorithm

Because the local address of the SIP server is known by the CES device, at least in the current prototype, it can detect if the message comes from the client or from the server. If the message comes from the client and it is an `INVITE`, the CES allocates a new proxy address for the client to be used for the RTP traffic. In a case the client is resending the `INVITE` it just reuses the mapping.

If the message from the client is `200 OK` with SDP, the CES checks whether it is a response to an `INVITE` via a trunk. In such a situation CES allocates two proxy addresses for the RTP and clears the temporary list that stores information used to create the mapping without extra allocations. The temporary list is used to store information from previously encountered packets to help processing of the packets received later. By doing this the CES functions in a way like a stateful firewall.

When it is a response to a local `INVITE` it just reuses the mapping allocated with the `INVITE`. However, if it is an answer to an `INVITE` from a client behind another CES device, a new mapping is created for the RTP traffic.

Messages containing SDP, which originated from outside of the private network and are forwarded by the SIP server, require that any previously allocated proxy addresses are released. These addresses are the ones allocated by local `INVITE`s. The new proxy address is allocated as described in the previous paragraph.

In a case the message is not an `INVITE`, the CES checks if the call is going to the voicemail. The voicemail requires allocating one more proxy address and then setting

two maps, one from the server side and one from the client side. The `200 OK`s with SDP content are transferred using existing mappings.

If the server forwards an `INVITE`, the first check performed by the algorithm inspects the NAT table for a RTP mapping. An existing RTP mapping is normally reused. If there is no mapping, CES checks for the `INVITE` related to a trunk connection. If the communication is going to use a trunk, one proxy address is allocated. When the server is forwarding a local `INVITE` one proxy is allocated and two maps are set.

At the end of the algorithm the local IP is changed to the remote proxy IP representing the originator and local proxy IP is changed to IP related to the destination.

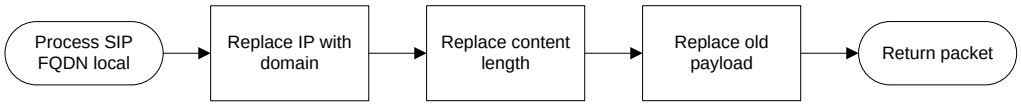

Figure 6.9: SIP ALG Local-FQDN algorithm

The Local-FQDN algorithm is shown in the Figure 6.9. Its functionality is a lot easier to understand than the Local-IP algorithm. The algorithm first retrieves the local and the remote domain. Then it changes the IP addresses related to those domains to FQDNs. This changes the content length which needs to be recalculated. At the end of the algorithm the old payload is replaced by a new payload.

6.3.2 Example run of Private Network with FQDN algorithm

Because of the local nature of the communication, there is no specific DNS server involved. Instead, in this scenario CES_A acts as a DNS server. All queries are directed to it.

As both of the clients are located in the same Local Private Network as the server, CES_A needs to allocate many proxy addresses to identify each communication. Two proxy addresses for each host to communicate with the server and also two proxy addresses to transfer RTP traffic from the caller to the callee.

When Host_A registers to the server, CES_A allocates one address that Host_A sees as a server and one address that represents the client to the server. In the following Figure 6.8 these addresses are identified as Pa-s and Ps-a. Proxy addresses related to

Host_C follow the same pattern. The addresses related to real time traffic between the clients can be seen as Ra-c and Rc-a.

This scenario tries to keep a similar test setting as public. This is done to help the comparing of the signalling flows between the different scenarios. Because of this, Host_A, extension 100, initiates the call while Host_C, extension 102, terminates it.

On a general level, this scenario is a bit similar to the one presented earlier in the section 6.2.2. In this test, both the clients and also the server are located behind the same CES device. This case reminds much the situation where a corporation has its own SIP server within the corporate network.

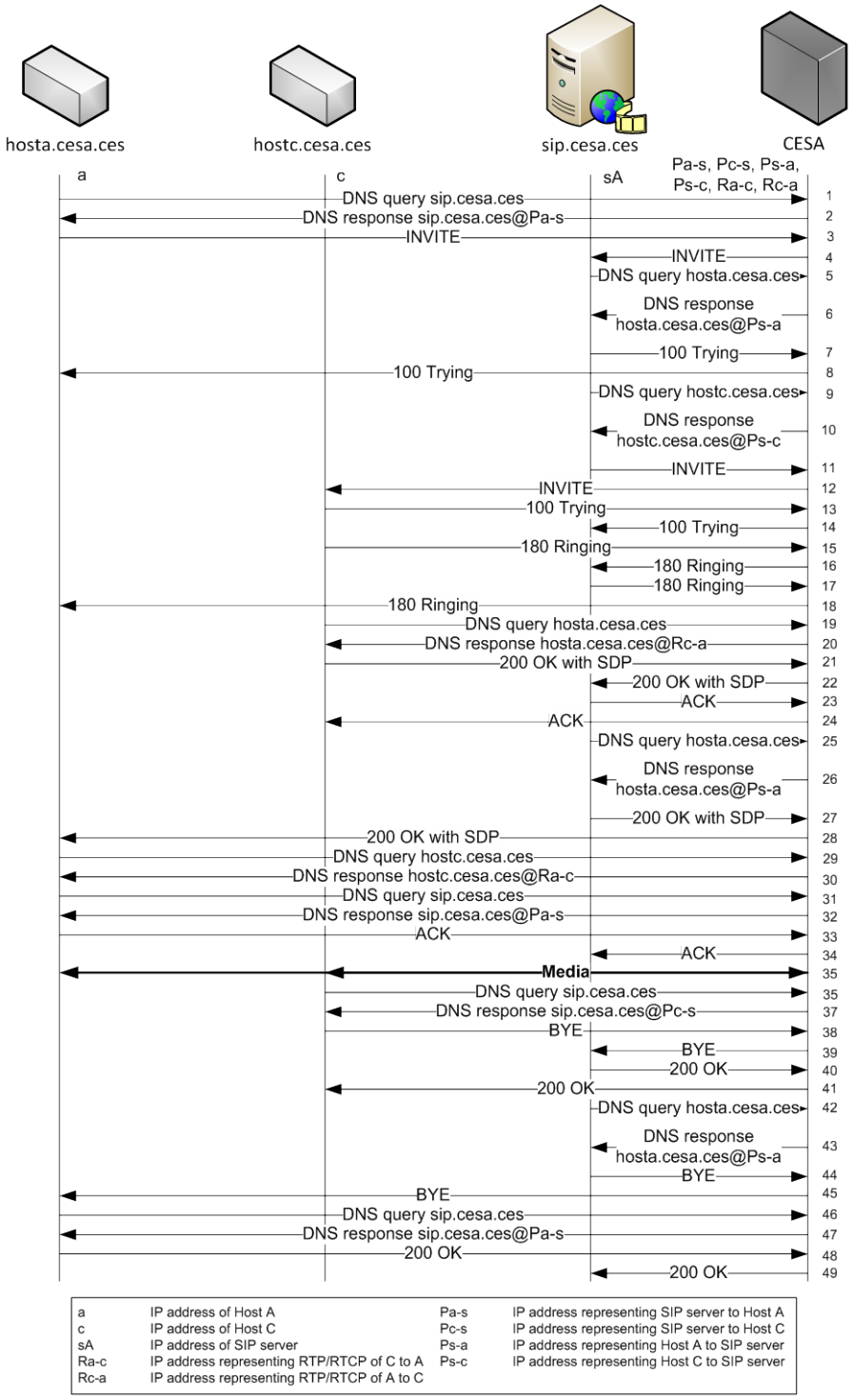

Figure 6.10: SIP ALG private network communication with Local-FQDN algorithm

Figure 6.10 shows an example how SIP signalling works in private communication. Private, in this case, means that both users are located behind the same CES device. The algorithm used to handle the messages performs actions according to FQDNs instead of IP addresses.

Authentication performed by the server is not visible in the message flow to maintain clarity. Both users are also already registered to the server. Individual messages are analysed next.

1. Host_A queries for address dedicated to the SIP server (sip.cesa.ces).
2. CES_A responds with previously allocated proxy address (Pa-s).
3. Host_A sends `INVITE` to initiate call with Host_C towards the server. The message contains the information: "`INVITE sip:102@sip.cesa.ces SIP/2.0, Via: SIP/2.0/UDP a:50600, c=IN IP4 a`".
4. CES_A forwards `INVITE` to the SIP server. The message contains the information: "`INVITE sip:102@sip.cesa.ces SIP/2.0, Via: SIP/2.0/UDP hosta.cesa.ces:50600, c=IN IP4 hosta.cesa.ces`".
5. Server queries for address dedicated to the Host_A (hosta.cesa.ces).
6. CES_A responds with previously allocated proxy address (Ps-a).
7. Server sends `100 Trying` towards Host_A.
8. CES_A forwards `100 Trying` to Host_A.
9. Server queries for address dedicated to the Host_C (hostc.cesa.ces).
10. CES_A responds with previously allocated proxy address (Ps-c).
11. Server sends `INVITE` towards Host_C. The message contains the information: "`INVITE sip:102@hostc.cesa.ces SIP/2.0, Via: SIP/2.0/UDP sA:50600, c=IN IP4 hosta.cesa.ces`".
12. CES_A forwards `INVITE` to Host_C. The message contains the information: "`INVITE sip:102@hostc.cesa.ces SIP/2.0, Via: SIP/2.0/UDP sip.cesa.ces:50600, c=IN IP4 hosta.cesa.ces`".
13. Host_C sends `100 Trying` towards the server.
14. CES_A forwards `100 Trying` to the SIP server.
15. Host_C sends `180 Ringing` towards the server.
16. CES_A forwards `180 Ringing` to the SIP server.
17. Server sends `180 Ringing` towards Host_A.
18. CESA forwards `180 Ringing` to Host_A.
19. Host_C queries for address dedicated to the Host_A (hosta.cesa.ces).
20. CES_A responds with newly allocated proxy address (Rc-a).
21. Host_C sends `200 OK` with SDP towards the server. The message contains the information: "`SIP/2.0 200 OK, Via: SIP/2.0/UDP sip.cesa.ces:50600, c=IN IP4 c`".

22. CES_A forwards `200 OK` with SDP to the SIP server. The message contains the information: "`SIP/2.0 200 OK, Via: SIP/2.0/UDP sip.cesa.ces:50600, c=IN IP4 hostc.cesa.ces`".
23. Server sends `ACK` towards Host_C.
24. CES_A forwards `ACK` to Host_C. This happens as the server initiates new session when it is forwarding the `INVITE` and thus acting as UAC.
27. Server sends `200 OK` with SDP towards Host_A. The message contains the information: "`SIP/2.0 200 OK, Via: SIP/2.0/UDP hosta.cesa.ces:50600, c=IN IP4 hostc.cesa.ces`".
28. CES_A forwards `200 OK` with SDP to Host_A. The message contains the information: "`SIP/2.0 200 OK, Via: SIP/2.0/UDP sip.cesa.ces:50600, c=IN IP4 hostc.cesa.ces`".
29. Host_A queries for address dedicated to the Host_C (hostc.cesa.ces).
30. CES_A responds with newly allocated proxy address (Ra-c).
33. Host_A sends `ACK` towards the server.
34. CES_A forwards `ACK` to the SIP server.
35. RTP traffic flow between the hosts.
36. Host_C queries for address dedicated to the SIP server (sip.cesa.ces).
37. CES_A responds with previously allocated proxy address (Pc-s).
38. Host_C sends `BYE` towards the server.
39. CES_A forwards `BYE` to the SIP server.
40. Server sends `200 OK` towards Host_C.
41. CESA forwards `200 OK` to Host_C.
44. Server sends `BYE` towards Host_A.
45. CESA forwards `BYE` to Host_A.
48. Host_A sends `200 OK` towards the server.
49. CES_A forwards `200 OK` to the SIP server.

Table 6.9 illustrates how the actual addresses and mappings are created in the previous flow. Each host, whether it is a client or a server, has two proxy addresses. The server needs to communicate with both clients and the clients need another address for RTP traffic.

TABLE 6.9 SIP PRIVATE CONNECTION CESA

NAT TABLE CES-to-CES				
Source		Destination		Type
IP	Proxy IP	IP	Proxy IP	
sA	Ps-a	a	Pa-s	local
sA	Ps-c	c	Pc-s	local
a	Pa-s	sA	Ps-a	local
a	Ra-c	c	Rc-a	local
c	Pc-s	sA	Ps-c	local
c	Rc-a	a	Ra-c	local

As both clients and the server are located in the same private network, the possibility to transfer packets directly starts to be a tempting option. The introduced option has 34 more packets and twice as much RTP packets as direct communication, because the RTP packets are also transferred via proxy addresses. However, it does not require that the users are aware of every other user or device. This can still make quite a big difference in the mobile environment or with other battery powered devices.

6.4 CES-to-CES communication

Communication between two CES devices is one of the key aspects for the prototype. CES-to-CES communication with SIP usually takes the form that one client is in the same network with the SIP server, CES_A as an example, and another client is located behind another CES device, CES_B network in this case.

As Section 6.3 indicated, CES-to-CES needs to use the local private algorithm when the traffic is processed within the Local Private Network hosting the client and the server. Communication between the server and the client in the other network uses CES-to-CES algorithms.

There are four different algorithms dedicated to the CES-to-CES communication. Two of those support traditional IP addresses and two functions use Fully Qualified Domain Names instead. These algorithms are now described in more detail to give better picture how they actually work.

6.4.1 ALG algorithms for CES-to-CES communication

The SIP Application Layer Gateway algorithm to process outbound traffic with IP addresses is visualized in Figure 6.11. The first check performed by the algorithm

spots `INVITE`s and 200 OKs with SDP forwarded by the server. Information in the temporary list is updated when processing `INVITE`s to help creating RTP mappings later. This means that the temporary list stores information from the previously encountered packets.

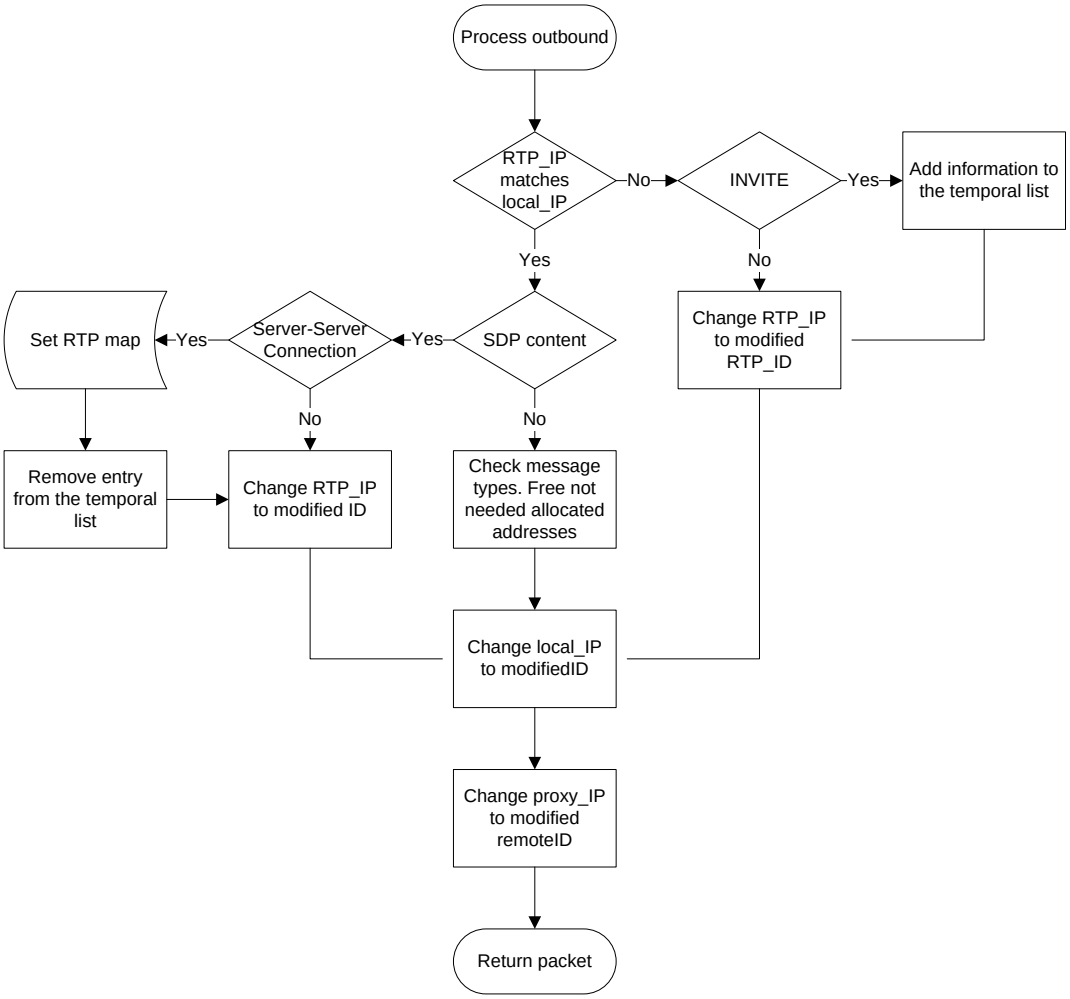

Figure 6.11: SIP ALG CES-to-CES-out-IP algorithm

In the current version the temporary list does not have timeout, but the items are cleared from it after processing specific packets. The timeout can be added to the list in the future as it is not more than one additional field. After this RTP IP is changed in both cases to the modified RTP ID. RTP ID is the ID representation of the source.

If the packet containing SDP has RTP IP matching a local IP, CES checks if the packet belongs to a trunk connection. In a trunk connection, RTP traffic goes via the servers. RTP mapping for a trunk connection is created straight away. Any unnecessary information in the temporary list is deleted. The temporary list is used to store information about the packets that have been encountered only once. The list

helps parsing the messages and also with the allocation of the addresses. Other option is that the packet comes from the client. Finally CES changes RTP IP to modified RTP ID as in previous paragraph.

When the packet does not have SDP content CES checks what kind of packet it actually is. This is done to check potential cancels and rejects. By doing this it is possible to avoid unnecessarily allocating resources.

Regarding every packet, at the end, the algorithm replaces local IP, representing the originator, with modified ID related to that client/server. After that proxy IP is replaced by modified remote ID, representing the destination behind another CES. Naturally the packet length is also recalculated by the algorithm.

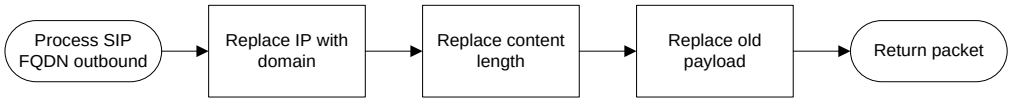

Figure 6.12: SIP CES-to-CES-out-FQDN algorithm

Figure 6.12 shows how the outbound algorithm using FQDNs works. As it became apparent with the local private algorithm, the algorithms that use Fully Qualified Domain Names are not as complicated as those using IP addresses.

At first, the algorithm gets the domain related to local IP. The local IP within this algorithm means the source of the packet. After this the local IPs contained in the SIP message are changed to this domain, as an example, changing 10.1.0.120 to hosta.cesa.ces.

Quite often this process changes the length of the packet. Thus, new content length needs to be calculated with the packets including SDP content. Finally, the payload with old information is replaced by the payload with the new payload.

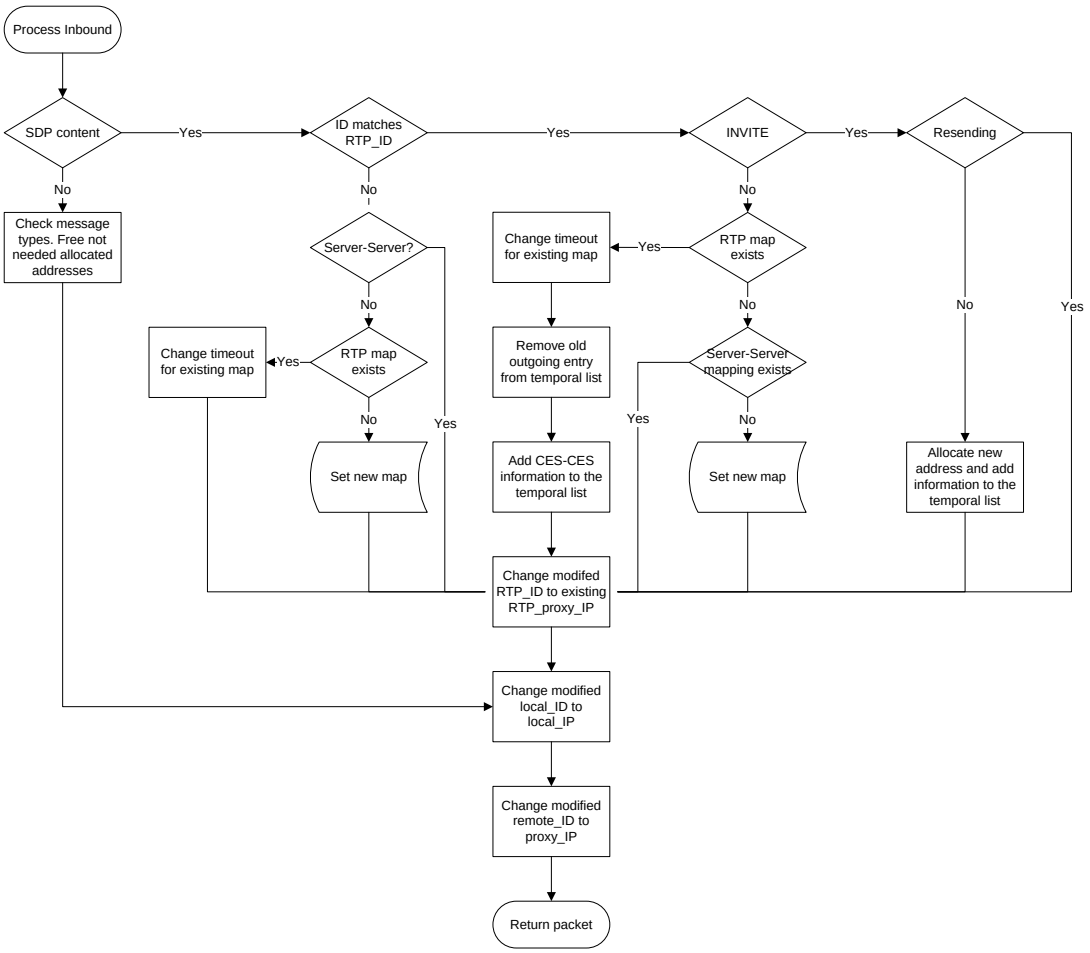

Figure 6.13: SIP ALG CES-to-CES-in-IP algorithm

Figure 6.13 demonstrates how the inbound algorithm performs. This algorithm does everything based on IP addresses. The algorithm divides packets depending on whether they have SDP content or not. If the packet does not have SDP content, the algorithm checks for cancellation or rejection of the call. By doing this SIP ALG does not waste resources as it frees previously allocated resources that are not anymore needed.

The packet that contains SDP information and is forwarded by the server is checked first for trunk connection. This can be verified by comparing the RTP ID, the ID representing the source of the RTP traffic, with the ID representing the source of the packet. If these IDs do not match, the communication is indeed happening via a trunk. The mapping already exists in a trunk connection as the servers have communicated earlier. Also the fact that RTP traffic flows via the servers helps changing RTP ID to existing RTP proxy IP.

Another case that can cause mismatch between ID and RTP ID is that the server forwards the `INVITE` or the `200 OK` with SDP content, when initiating a normal client-client session. The next check was done because there were differences between the clients. Twinkle as an example tries to send RTP straight after sending `200 OK` with SDP. If there is no mapping yet, the CES will allocate a new mapping automatically for RTP. The timeout for this mapping is changed to have a similar timeout with mappings related to SIP. If the `200 OK` packet with SDP content arrives before RTP traffic, a new mapping is created.

In the case when ID and RTP ID match, the algorithm checks if the message is an `INVITE`. `INVITE`s require that information is stored to the temporary list, as the source of the RTP in the private network is still unknown. Resending the `INVITE` does not cause any additional processing. In both cases RTP ID is changed to proxy IP.

Another situation that causes RTP ID and ID to be the same is when client/server sends `200 OK` with SDP. If the message is from the server, indicating a trunk connection, there should already be a mapping to support the RTP traffic. In a client-client connection a new map is created. The proxy IP replaces again RTP ID.

In the last case there is already a mapping when CES receives a `200 OK` with SDP content. Changing the timeout occurs in the same way as described earlier in this section. As explained, this is done to ensure integrity. Any information stored to the temporary list is cleared. Information about CES-to-CES connection is added to the temporary list. The temporary list is in a sense really close to the mechanism used in some stateful firewall as it stores information that it uses later on for decisions. This helps creating the mapping when the message is seen again. After these procedures RTP ID changes to proxy IP.

All of the packets processed by the algorithm are changed in the same way at the end of the algorithm. Local ID, representing the destination, changes to local IP and the remote ID, representing the source, changes to proxy IP.

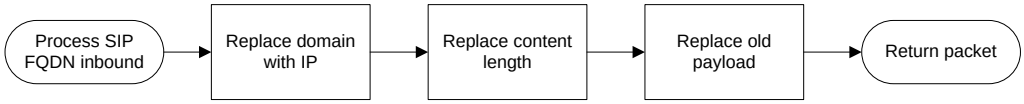

Figure 6.14: SIP ALG CES-to-CES-in-FQDN algorithm

SIP ALG inbound algorithm that uses FQDNs instead of IP addresses is presented in the Figure 6.14. As with the previous FQDN algorithms it differs a lot from the algorithm performing operations based on IP addresses. The algorithm searches for the domain related to the local IP address. After that domains inside the packet are replaced by the local IP.

Replacing information within the packet frequently affects the content length. This requires that old content length is changed to new one calculated after the changes. The old payload is swapped to a new payload at the end of the algorithm.

6.4.2 Example run of CES-to-CES using FQDN algorithms

CES-to-CES communication is probably the most important aspect for the prototype. One reason introduced at this point is that it uses also the local private algorithm. Another reason why this case is presented here is the fact that the prototype needs to support public aspects. Because of these requirements, the example runs regarding the public communication and the private communication were presented before this.

In this scenario one client is located in a CES_A network while another is in a CES_B network. The SIP server is placed in the CES_A network. Actually, for testing purposes, the location of the server does not matter as either possible location is just a mirror image of the other option. This kind of scenario can happen if for example one client is inside a corporate network and another is connected to the same network from home.

A DNS server which responds DNS queries is located in the public network between the CES devices. Queries from a private network are first directed to CES then, if necessary, to the DNS server in the public network. The server redirects queries to the proper CES, as an example, query regarding hostb.cesb.ces is directed to CES_B. More specific information about the placement of each client, server and device is presented in Chapter 9.

Host_A acts as a caller, sending the `INVITE`, and Host_B as a callee. Host_B terminates the call by sending the `BYE`. This follows the same approach as the previous example runs. A similar pattern in the example runs is used to help comparing the runs between each other.

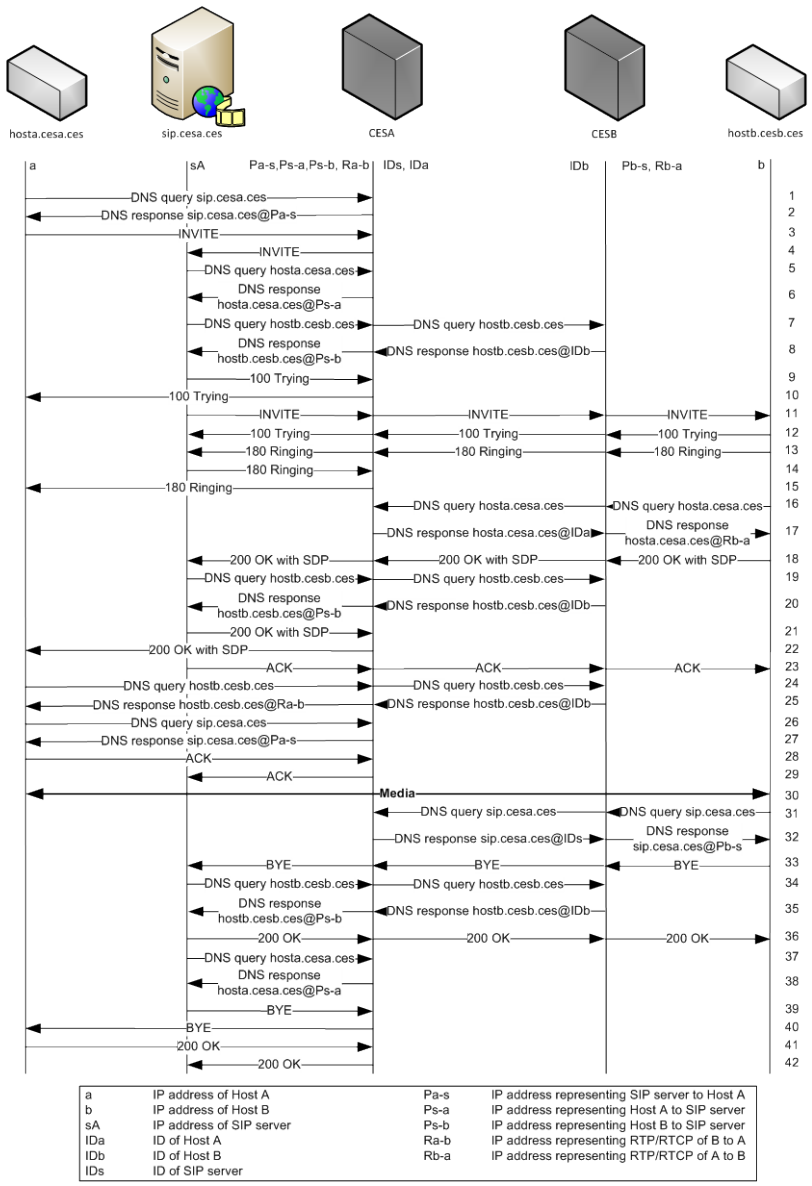

Figure 6.15: SIP ALG CES-to-CES communication with FQDN algorithms

Figure 6.15 represents how signalling flow happens when a client in the CES_A network calls to a client in the CES_B network. Both users are already registered in the SIP server located in the same network as the caller. This means that they already have allocated proxy addresses towards the server. Naturally, the server has one proxy address for representing each of the users. The algorithms used to enable the call use domains instead of IP addresses. The packet capture related to the figure is located in Appendix A.

There is no specific DNS server used in this scenario. Queries are directed to the other CES device. Authentication required by the server is removed from the message flow to keep it as simple as possible.

What happens when these messages are processed is described next.

1. Host_A queries for address dedicated to the SIP server (sip.cesa.ces).
2. CES_A responds with previously allocated proxy address (Pa-s).
3. Host_A sends an `INVITE` to initiate call to hostb towards the server. The message contains the information: "`INVITE sip:101@sip.cesa.ces SIP/2.0, Via: SIP/2.0/UDP a:50600, c=IN IP4 a`". The clients are registered to the same server residing in the CES_A network.
4. CES_A forwards `INVITE` to the SIP server. The message contains the information: "`INVITE sip:101@sip.cesa.ces SIP/2.0, Via: SIP/2.0/UDP hosta.cesa.ces:50600, c=IN IP4 hosta.cesa.ces`".
5. Server queries for address dedicated to the Host_A (hosta.cesa.ces).
6. CES_A responds with previously allocated proxy address (Ps-a).
7. Server queries for the allocated address of the Host_B (hostb.cesb.ces). Query is transferred to CES_B, because host is located behind that CES device.
8. CES_B responds with ID representing Host_B (IDb). CES_A changes this to proxy IP address (Ps-b).
9. Server sends `100 Trying` towards Host_A (Ps-a).
10. CESA forwards `100 Trying` to Host_A.
11. Server sends `INVITE` towards Host_B (Ps-b), CES_A changes this to IDb and sends it to CES_B, CES_B forwards INVITE to Host_B. The message contains the information: "`INVITE sip:101@hostb.cesb.ces SIP/2.0, Via: SIP/2.0/UDP sA/sip.cesa.ces:50600, c=IN IP4 hosta.cesa.ces`". The IP address of the SIP server sA is changed to corresponding FQDN by the CES_A.
12. Host_B responds with `100 Trying` that gets forwarded back to the server.
13. Host_B responds with `180 Ringing` that gets forwarded back to the server.
14. Server sends `180 ringing` to Host_A so that the caller knows the callee has been reached.
15. CES_A forwards `180 ringing` to the Host_A.
16. Host_B queries the address dedicated to the Host_A (hosta.cesa.ces). Query is transferred to CES_A, because host is located behind that CES device. The query is initiated when the Host_B answers to the call.
17. CES_A responds with ID representing Host_A (IDa). CES_B changes this to proxy IP address (Rb-a). This address is used for real-time transfer.
18. Host_B sends `200 OK` with Session Description Protocol to the server. The message contains the information: "`SIP/2.0 200 OK, Via:`

`SIP/2.0/UDP sip.cesa.ces:50600, c=IN IP4 b/hostb.cesb.ces`". Processing is similar as with message 11.

21. Server sends `200 OK` with SDP towards Host_A. The message contains the information: "`SIP/2.0 200 OK, Via: SIP/2.0/UDP hosta.cesa.ces:50600, c=IN IP4 hostb.cesb.ces`".
22. CESA forwards `200 OK` with SDP to Host_A. The message contains the information: "`SIP/2.0 200 OK, Via: SIP/2.0/UDP hosta.cesa.ces:50600, c=IN IP4 hostb.cesb.ces`".
23. Server acknowledges that it has received `200 OK` with SDP from Host_B. This is software specific behaviour, when the SIP server acts as client.
24. Host_A queries address dedicated to the Host_B (hostb.cesb.ces). Query is transferred to CES_B, because host is located behind that CES device.
25. CES_B responds with ID representing Host_B (IDb). CES_A changes this to proxy IP address (Ra-b). This address is used for real-time transfer.
28. Host_A sends acknowledgement to the server.
29. CES_A forwards `ACK` to the server.
30. RTP traffic flow between the hosts.
33. Host_B terminates call by sending `BYE` to the server.
36. Server responds with `200 OK`.
39. Server sends `BYE` to Host_A.
40. CES_A forwards `BYE`.
41. Host_A responds with `200 OK`.
42. CES_A forwards `200 OK`.

The allocation of the addresses and mapping those addresses, regarding the previous message flow is presented in the following tables. The table 6.10 shows how mappings are created in CES_A and the Table 6.11 how it is done in CES_B.

TABLE 6.10 SIP CES-TO-CES CONNECTION CESA

NAT TABLE CES-to-CES				
Source		Destination		Type
IP	Proxy IP	IP	Proxy IP/ID	
a	Pa-s	sA	Ps-a	local
sA	Ps-a	a	Pa-s	local
sA	Ps-b	IP CES_B	IDb	CES-to-CES
a	Ra-b	IP CES_B	IDb	CES-to-CES

TABLE 6.11 SIP CES-TO-CES CONNECTION CESB

NAT TABLE CES-to-CES				
Source		Destination		Type
IP	Proxy IP	IP	Proxy IP/ID	
b	Rb-s	IP CES_A	IDs	CES-to-CES
b	Rb-a	IP CES_A	IDa	CES-to-CES

When there are more than one CES device the possibility to use DNS caching starts to be more appealing as it can remove quite many packets from the traffic flow. In the example case it would remove six packets. As with the private network scenario Section 6.3.2, the possibility that users in a private network can address each other directly would have quite a big impact on the amount of the messages. In this scenario it would remove 17 messages, which is still not that impressive when compared to the RTP traffic.

6.5 Communication using a trunk between two servers

Trunk connection means a communication between clients that have registered to two different SIP servers and these servers can communicate via a trunk. The caller is registered to one server and the callee to another. Trunks can be used even if the users are not in the same network as the server. It is possible to use trunks even if the caller/callee register to a server located behind different CES devices.

Two servers are required to create a trunk. Both servers need to have settings configured to allow trunk connections. Clients need to use a specific prefix to call via the desired trunk. The prefix is used to define the destination server, in which the callee should be registered. In this test environment using single digit prefix is sufficient. The functionality of the used SIP server is described in detail in chapter 4.

6.5.1 Algorithms for trunk communication

There are no specific algorithms for trunk connections. When hosts communicate via trunks they use both local and CES-to-CES algorithms. If the client and the server are located behind the same CES device, registration and other signalling between the client and the server use the local private algorithm. If the caller and the callee are behind different CES devices, communication between the servers requires CES-to-CES algorithms. A similar approach is taken when the client or the server is in the

public network. The only difference is that algorithms used for communication are implemented for public communication not CES-to-CES. More specific details about these algorithms can be found in previous sections of this chapter.

As with other algorithms, trunks can also use either the algorithms using IP addresses or the algorithms that use FQDN. This allows creating even really complicated and unlikely test settings, which can be useful when testing the performance of the prototype. Chapter 9 represents the structure of both the virtual and the physical test environment.

6.5.2 Example run of trunk communication between two sites

Trunk connection was chosen to be the last example as it does not have its own algorithms. Packets sent by it are processed by local and CES-to-CES/public algorithms. Explanations of those algorithms can help understanding how packets involved in the trunk connection are actually handled. As previously mentioned, the trunk connection requires two servers.

In this test example one is located in the CES_A network and another in the CES_B network. In a similar way, two clients are situated into both networks. Host_A registers to the SIP server in CES_A network. In the same way Host_B registers to the server located in CES_B network. Registrations are not included in the message flow.

The trunk communication resembles CES-to-CES communication also in the aspect of the DNS server. The queries from the hosts are directed the CES devices. CES will reply with DNS response if it has information of the desired destination. Otherwise, query is transferred to a DNS server in public network.

The structure of the call follows the same suite as in the previous examples. This means that Host_A, acting as caller, sends the `INVITE`. Then the call is terminated by the Host_B by sending the `BYE`.

This kind of scenario using trunks can happen if there are two corporate networks in separate locations. Both of these private networks have their own SIP server. These SIP servers are the only devices communicating between the networks. Because of this, the signalling and the RTP traffic both flow through the SIP servers.

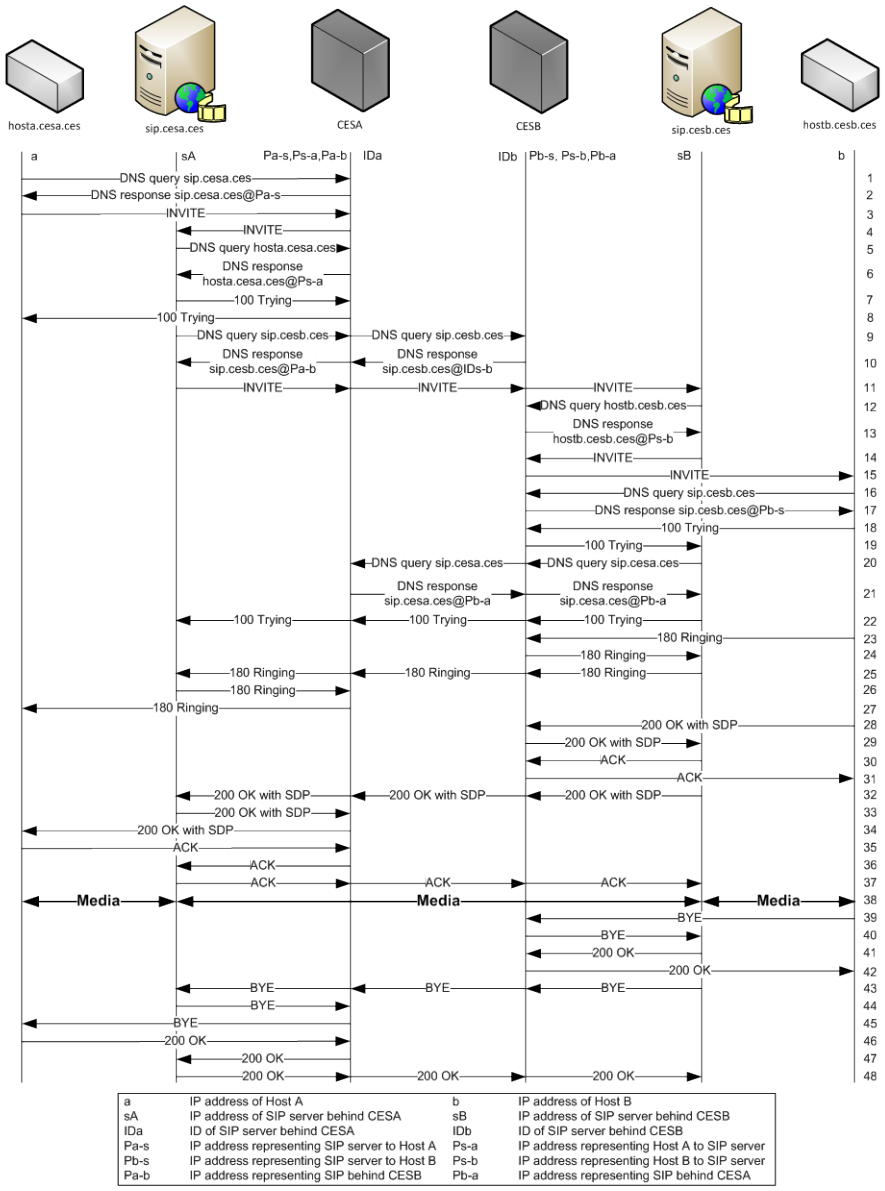

Figure 6.16: SIP ALG trunk communication with FQDN algorithms

Figure 6.16 visualizes how signalling works when clients located and registered in different networks initiate and receive a call. Host_A located in CES_A network and registered to sip.cesa.ces acts as a caller. Thus, Host_B located in CES_B network acts as a callee while registered to server sip.cesb.ces.

There is no specific DNS server presented in this figure. However, the actual test environment uses a DNS server, as presented in chapter 9. Queries are directed to the other CES device. Authentication required by the server is removed from the message flow to keep it as simple as possible. Because of the number of the messages, subsequent DNS queries are not visible in the figure.

Actual signalling flow is explained next.

1. Host_A queries for the address allocated to the SIP server (sip.cesa.ces).
2. CES_A responds with previously allocated proxy address (Pa-s).
3. Host_A sends `INVITE` with a prefix to initiate a call to Host_B towards the server. The message contains the information: "`INVITE sip:8101@sip.cesa.ces SIP/2.0, Via: SIP/2.0/UDP a:50600, c=IN IP4 a`". The `INVITE` needs to be sent to the server in the same network. Extension numbers and trunks were discussed in section 4.1.4.
4. CES_A forwards `INVITE` to the SIP server. The message contains the information: "`INVITE sip:8101@sip.cesa.ces SIP/2.0, Via: SIP/2.0/UDP hosta.cesa.ces:50600, c=IN IP4 hosta.cesa.ces`".
5. The server queries for the address allocated to the Host_A (hosta.cesa.ces).
6. CES_A responds with previously allocated proxy address (Ps-a).
7. The server sends `100 Trying` towards Host_A.
8. CES_A forwards `100 Trying` to Host_A.
9. The server queries the address allocated to the server behind CES_B (sip.cesb.ces). Query is transferred to CES_B, because the server is located behind that CES device.
10. CES_B responds with the identification of the server (IDb). The response is returned to SIP server by CES_A with proxy address (Pa-b).
11. The SIP server in the CES_A network sends an `INVITE` towards the server in the CES_B network. The `INVITE` is forwarded to the server. The message contains the information: "`INVITE sip:101@sip.cesb.ces SIP/2.0, Via: SIP/2.0/UDP s_a/sip.cesa.ces:50600, c=IN IP4 sA/sip.cesa.ces`". sA stands as private address of the SIP server located in the CES_A network. The private address is replaced by the domain name when CES_A processes the packet.
12. Server queries for the address allocated to the Host_B (hostb.cesb.ces).
13. CES_B responds with previously allocated proxy address (Ps-b).
14. The server sends `INVITE` towards Host_B. The The message contains the information: "`INVITE sip:101@hostb.cesb.ces SIP/2.0, Via: SIP/2.0/UDP s_b:50600, c=IN IP4 s_b`".
15. CES_B forwards `INVITE` to Host_B. The The message contains the information: "`INVITE sip:101@sip.cesb.ces SIP/2.0, Via: SIP/2.0/UDP sip.cesb.ces:50600, c=IN IP4 sip.cesb.ces`".

16. Host_B queries for address allocated to the SIP server (sip.cesb.ces).
17. CES_A responds with previously allocated proxy address (Pb-s).
18. Host_B sends `100 Trying` towards server.
19. CES_B forwards `100 Trying` to server.
20. The server queries address allocated to the server behind CES_A (sip.cesa.ces). Query is transferred to CES_A, because the server is located behind that CES device.
21. CES_A responds with identification of the server (IDa). Response is returned to the SIP server by CES_B with proxy address (Pb-a).
22. The SIP server in the CES_B network sends `100 Trying` towards server in CES_A network. `100 Trying` is forwarded to the server.
23. Host_B sends `180 Ringing` towards the server.
24. CES_B forwards `180 Ringing` to the server.
25. The SIP server in the CES_B network sends 180 Ringing towards server in CES_A network. `180 Ringing` is forwarded to the server.
26. The server sends `180 Ringing` towards Host_A.
27. CES_A forwards `180 Ringing` to Host_A.
28. Host_B sends `200 OK` with SDP towards the server. The message contains the information: "`SIP/2.0 200 OK, Via: SIP/2.0/UDP sip.cesb.ces:50600, c=IN IP4 b`".
29. CES_B forwards `200 OK` with SDP to the server. The message contains the information: "`SIP/2.0 200 OK, Via: SIP/2.0/UDP sip.cesb.ces:50600, c=IN IP4 hostb.cesb.ces`".
30. The server sends `ACK` towards Host_B. This is software specific behaviour.
31. CES_B forwards `ACK` to Host_B.
32. The SIP server in the CES_B network sends `200 OK` with SDP towards the server in the CES_A network. `200 OK` with SDP is forwarded to the server. The message contains the information: "`SIP/2.0 200 OK, Via: SIP/2.0/UDP sip.cesa.ces:50600, c=IN IP4 sB/sip.cesb.ces`". Processing is similar to message 11.
33. The server sends `200 OK` with SDP towards Host_A. The message contains the information: "`SIP/2.0 200 OK, Via: SIP/2.0/UDP hosta.cesa.ces:50600, c=IN IP4 s_a`".
34. CESA forwards `200 OK` with SDP to Host_A. The message contains the information: "`SIP/2.0 200 OK, Via: SIP/2.0/UDP hosta.cesa.ces:50600, c=IN IP4 sip.cesa.ces`".
35. Host_A sends `ACK` towards the server.
36. CES_A forwards `ACK` to the server.

37. SIP server in CES_A network sends ACK towards the server in the CES_B network. ACK is forwarded to the server.
38. RTP traffic flows between the hosts.
39. Host_B sends BYE towards the server.
40. CES_B forwards BYE to the server.
41. The server sends 200 OK towards Host_B.
42. CES_B forwards 200 OK to Host_B.
43. The SIP server in the CES_B network sends BYE towards the server in the CES_A network. BYE is forwarded to the server.
44. The server sends BYE towards Host_A.
45. CESA forwards BYE to Host_A.
46. Host_A sends 200 OK towards the server.
47. CES_A forwards 200 OK to the server.
48. The SIP server in the CES_A network sends 200 OK towards the server in the CES_B network. 200 OK is forwarded to the server.

The addresses used in the previous test scenario can be found in the tables below. The table 6.12 represents the addresses and the mappings in CES_A, while the Table 6.13 shows similar information about CES_B. As signaling and RTP traffic both go via the servers there is no need for more than one mapping for CES-to-CES communication.

TABLE 6.12 SIP TRUNK CONNECTION CESA

NAT TABLE CES-to-CES				
Source		Destination		Type
IP	Proxy IP	IP	Proxy IP/ID	
a	Pa-s	sA	Ps-a	local
sA	Ps-a	a	Pa-s	local
sA	Pa-b	IP CES_B	IDb	CES-to-CES

TABLE 6.13 SIP TRUNK CONNECTION CESB

NAT TABLE CES-to-CES				
Source		Destination		Type
IP	Proxy IP	IP	Proxy IP	
b	Pb-s	sB	Ps-b	local
sB	Ps-b	b	Pb-s	local
sB	Pb-a	IP CES_A	IDa	CES-to-CES

If we perform the same tricks as with previous scenarios with the trunk communication, it is possible to shorten the message flow by 22 messages. Again, this amount of packets is not that much, when it is used to set up and tear down a RTP communication.

6.6 Address and port changes

As a chapter conclusion, different changes affecting the addresses and ports within the packet in each individual case are presented. Both inbound and outbound are considered in the relevant examples. At first, the structure of the notation is presented in Figure 6.17. This structure is used through this section.

$$The\ structure\ of\ the\ vector: \begin{bmatrix} IP_{src} \\ Port_{src} \\ IP_{dst} \\ Port_{dst} \\ RTP_{IP} \\ RTP_{Port} \end{bmatrix}$$

Figure 6.17: Structure of the address and port notation

Communication between the private network and the public network

With public addresses the ports are also used in addition to the IP addresses. If one client is located in the public network and the server with another client are located in the private network, only one public address needs to be allocated. This address is used to represent the client in the private network for the client in the public network.

This matches the situation shown in the Figure 6.6. Temporary address marked with t is released when the `200 OK` with SDP is transferred to the public network. Arrows in the Figures represent the processing in the CES. Figure 6.18 shows how `INVITE` is changed. Then Figure 6.19 presents how `200 OK` with SDP is changed.

$$\begin{matrix} & Caller\ out & & Server\ in & & Server\ out & & Callee\ in \\ Caller & \begin{bmatrix} IP_d \\ Port_d \\ IP_{Rs} \\ Port_{Rs} \\ IP_d \\ Port_d \end{bmatrix} & \rightarrow & \begin{bmatrix} IP_d \\ Port_d \\ IP_s \\ Port_s \\ IP_d \\ Port_d \end{bmatrix} & server & \begin{bmatrix} IP_s \\ Port_s \\ IP_{Pa} \\ Port_{Pa} \\ IP_d \\ Port_d \end{bmatrix} & \rightarrow & \begin{bmatrix} IP_{Ps} \\ Port_{Ps} \\ IP_a \\ Port_a \\ IP_d \\ Port_d \end{bmatrix} \end{matrix}$$

Figure 6.18: Caller sending INVITE in public communication

$$\begin{matrix} & Callee\ out & & Server\ in & & Server\ out & & Caller\ in \\ Callee & \begin{bmatrix} IP_a \\ Port_a \\ IP_{Ps} \\ Port_{Ps} \\ IP_a \\ Port_a \end{bmatrix} & \rightarrow & \begin{bmatrix} IP_{Pa} \\ Port_{Pa} \\ IP_s \\ Port_s \\ IP_t \\ Port_t \end{bmatrix} & server & \begin{bmatrix} IP_s \\ Port_s \\ IP_d \\ Port_d \\ IP_t \\ Port_t \end{bmatrix} & \rightarrow & \begin{bmatrix} IP_{Rs} \\ Port_{Rs} \\ IP_d \\ Port_d \\ IP_{Ra} \\ Port_{Ra} \end{bmatrix} \end{matrix}$$

Figure 6.19: Callee responding with 200 OK with SDP in public communication

Communication in the private network

In a communication within a private network the source IP and proxy IP representing the destination are changed to proxy IP representing the source and remote IP. Ports are not allocated or changed by any way. This scenario relates to the example run shown in the Figure 6.10.

RTP traffic requires one address per host. Thus, two addresses and one mapping are needed. These mappings are created when the hosts perform DNS queries. Arrows again represent the CES devices. Figure 6.20 visualizes the changes of an `INVITE`. Then Figure 6.21 shows how `200 OK` with SDP is processed.

$$\begin{matrix} & Caller\ out & & Server\ in & & Server\ out & & Callee\ in \\ Caller & \begin{bmatrix} IP_a \\ Port_a \\ IP_{Pa\_s} \\ Port_{Pa\_s} \\ IP_a \\ Port_a \end{bmatrix} & \rightarrow & \begin{bmatrix} IP_{Ps\_a} \\ Port_{Ps\_a} \\ IP_s \\ Port_s \\ FQDN_a \\ Port_a \end{bmatrix} & server & \begin{bmatrix} IP_s \\ Port_s \\ IP_{Ps\_c} \\ Port_{Ps\_c} \\ FQDN_a \\ Port_a \end{bmatrix} & \rightarrow & \begin{bmatrix} IP_{Pc\_s} \\ Port_{Pc\_s} \\ IP_c \\ Port_c \\ FQDN_a \\ Port_a \end{bmatrix} \end{matrix}$$

Figure 6.20: Caller sending INVITE in private communication

$$\begin{matrix} & Callee\ out & & Server\ in & & Server\ out & & Caller\ in \\ Callee & \begin{bmatrix} IP_c \\ Port_c \\ IP_{Pc\_s} \\ Port_{Pc\_s} \\ IP_c \\ Port_c \end{bmatrix} & \rightarrow & \begin{bmatrix} IP_{Ps\_c} \\ Port_{Ps\_c} \\ IP_s \\ Port_s \\ FQDN_c \\ Port_c \end{bmatrix} & server & \begin{bmatrix} IP_s \\ Port_s \\ IP_{Ps\_a} \\ Port_{Ps\_a} \\ FQDN_c \\ Port_c \end{bmatrix} & \rightarrow & \begin{bmatrix} IP_{Pa\_s} \\ Port_{Pa\_s} \\ IP_a \\ Port_a \\ FQDN_c \\ Port_c \end{bmatrix} \end{matrix}$$

Figure 6.21: Callee responding with 200 OK with SDP in private communication

CES-to-CES communication

The communication between two CES devices uses IDs instead of IP addresses. As with the earlier examples, this case is also shown in earlier Figure 6.15. For the RTP traffic additional mappings are created and addresses allocated in both CES devices. As with the previous cases the arrows show where the CES devices change the information. This case also uses FQDN.

Figure 6.22 demonstrates how an `INVITE` message is processed in CES-to-CES communication. After that, Figure 6.22 explains how `200 OK` with SDP, an answer to the previous `INVITE` is handled.

$$\begin{matrix} & Caller\ out & & Server\ in & & Server\ out & & Core & & Callee\ in \\ Caller & \begin{bmatrix} IP_a \\ Port_a \\ IP_{Pa\_s} \\ Port_{Pa\_s} \\ IP_a \\ Port_a \end{bmatrix} & \rightarrow & \begin{bmatrix} IP_{Ps\_a} \\ Port_{Ps\_a} \\ IP_s \\ Port_s \\ FQDN_a \\ Port_a \end{bmatrix} & server & \begin{bmatrix} IP_s \\ Port_s \\ IP_{Ps\_b} \\ Port_{Ps\_b} \\ FQDN_a \\ Port_a \end{bmatrix} & \rightarrow & \begin{bmatrix} IP_{IDs} \\ Port_{IDs} \\ IP_{IDb} \\ Port_{IDb} \\ FQDN_a \\ Port_a \end{bmatrix} & \rightarrow & \begin{bmatrix} IP_{Pb\_s} \\ Port_{Pb\_s} \\ IP_b \\ Port_b \\ FQDN_a \\ Port_a \end{bmatrix} \end{matrix}$$

Figure 6.22: Caller sending INVITE in CES-to-CES communication

$$\begin{matrix} & Callee\ out & & Core & & Server\ in & & Server\ out & & Caller\ in \\ Callee & \begin{bmatrix} IP_b \\ Port_b \\ IP_{Pb\_s} \\ Port_{Pb\_s} \\ IP_b \\ Port_b \end{bmatrix} & \rightarrow & \begin{bmatrix} IP_{IDb} \\ Port_{IDb} \\ IP_{IDs} \\ Port_{IDs} \\ FQDN_b \\ Port_b \end{bmatrix} & \rightarrow & \begin{bmatrix} IP_{Ps\_b} \\ Port_{Ps\_b} \\ IP_s \\ Port_s \\ FQDN_b \\ Port_b \end{bmatrix} & server & \begin{bmatrix} IP_s \\ Port_s \\ IP_{Ps\_a} \\ Port_{Ps\_a} \\ FQDN_b \\ Port_b \end{bmatrix} & \rightarrow & \begin{bmatrix} IP_{Pa\_s} \\ Port_{Pa\_s} \\ IP_a \\ Port_a \\ FQDN_b \\ Port_b \end{bmatrix} \end{matrix}$$

Figure 6.23: Callee responding with 200 OK with SDP in CES-to-CES communication

7. FTP Application Layer Gateway

This chapter explains how FTP is processed in the current prototype. At the beginning of the chapter FTP is described from a higher level. The second part focuses in more detail on local behavior of FTP, both behind the same CES. The last part of the chapter describes how FTP is handled when a client and a server are located behind different CES devices.

7.1 FTP ALG Briefly

Application Layer Gateway for FTP is created to support FTP with the current prototype. The previous version of the prototype did not offer suitable algorithms that could enable FTP communication between a client and a server. As mentioned in the Section 4.2.1 the ALG can affect to the length of the packet. Because of this the ALG stores connection independent offset in order to make right changes in the packets. Calculation of the offset is shown in the Figure 7.1.

$$Offset_{new} = Offset_{old} + Length_{original} - Length_{modified}$$

Figure 7.1: Calculation of the ALG offset from the length of the packets

The offset is used to change the sequence number and acknowledgement number in the packets so that they match the algorithms presented in Section 4.2.1 even after the CES has processed the packet. First of the changes affects the sequence number and it is visualized in the Figure 7.2.

$$Seq_{sent} = Seq_{received} + Offset_{con}$$

Figure 7.2: Calculation of the new sequence number

The acknowledgement number is changed in similar way. The biggest difference is that it does not use the same offset as with the sequence number. The offset used for this change is related to the traffic between the same hosts but other direction. This is presented in Figure 7.3.

$$Ack_{sent} = Ack_{received} - Offset_{other}$$

Figure 7.3: Calculation of the new acknowledgement number

7.2 FTP communication in a private network

FTP local algorithm is designed to support situations in which both the client and the server are located in the same private network. This means that communication happens behind one CES device.

Like in the case of Session Initiation Protocol we have designed to allow many private address realms behind one CES. Therefore all communication between the client and the server flows through the CES.

7.2.1 ALG algorithm for FTP communication in a private network

The algorithm, shown in the Figure 7.4, checks first if the packet is initializing an active connection by sending a `PORT` message. In a case where is no state yet, the algorithm creates a new state. The state is constructed in the following way: (local_ip, local_proxy_ip, remote_ip, remote_proxy_ip) in active mode. This structure can be used to retrieve the offset related to the connection or increase it.

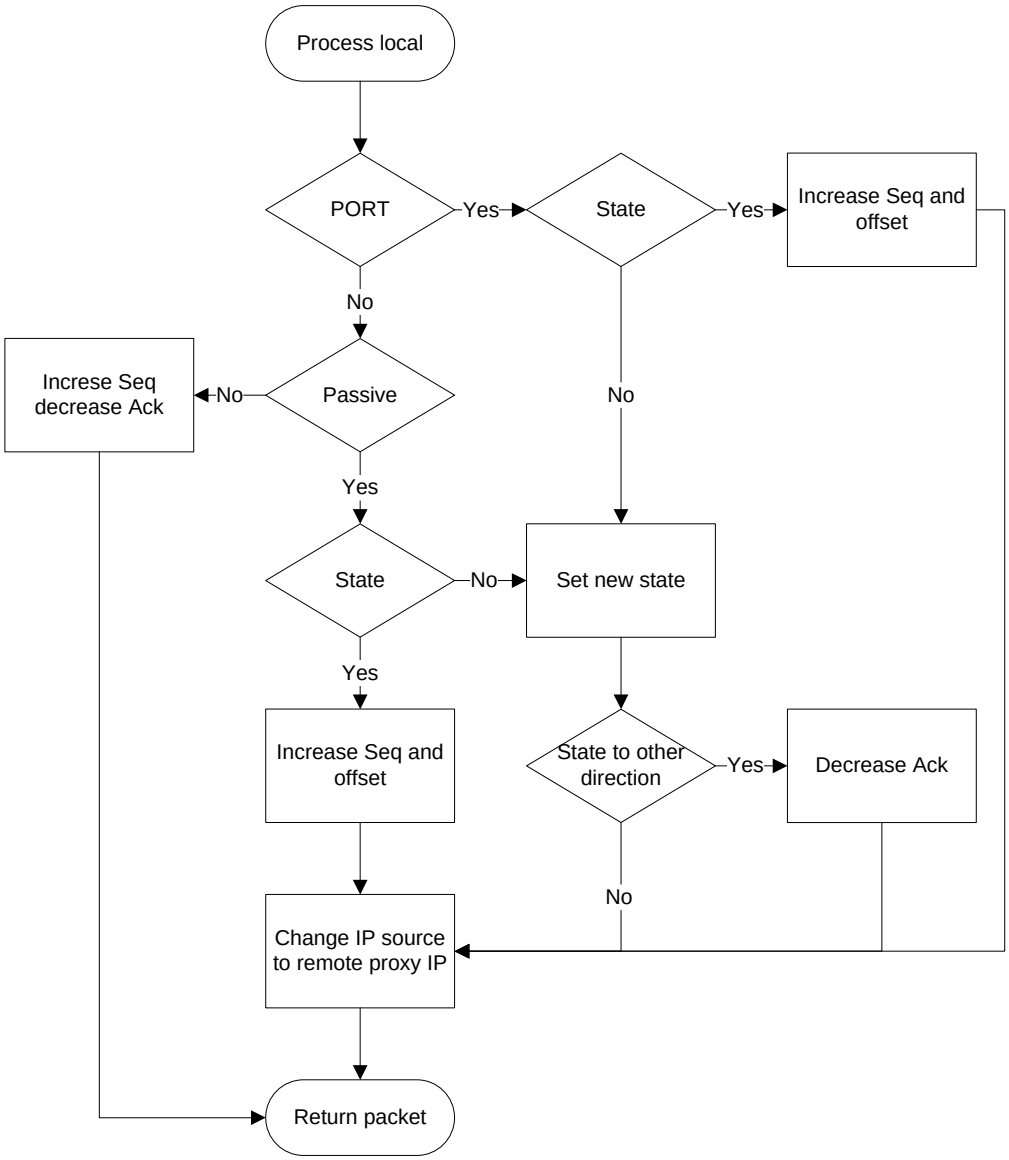

Figure 7.4: FTP ALG algorithm for Local Private Network communication

Then the algorithm checks state for traffic to another direction in passive mode (remote_ip, remote_proxy_ip, local_ip, local_proxy_ip). This is done to change Ack when needed. When the state exists Seq and offset are increased.

If the message informs about initiation of passive communication, “227 Entering Passive Mode”, the state is checked in the same way as in the PORT. If there is no previous state, the algorithm creates a new state. After that, the state related to the other direction, is checked to ensure whether Ack needs to be decreased. If the PASV is not the first packet requiring the passive connection, the Seq and offset are increased.

In both of the situations, described in the previous paragraphs, the algorithm changes IP source, contained within the message, to the remote proxy IP. By doing this the packet's source matches the information contained in it.

When the packet does not relate to either one, active or passive, the algorithm just updates Seq and Ack depending on the situation. The seq is increased according to the offset related to transfer in the direction that the packet is going. Ack is decreased if there is state bound for traffic to the other direction. The offset relates always to the offset in the ALG not the offset in the TCP header.

7.2.2 Example run of FTP communication in a private network

In the local scenario both the client and the server are located behind the same CES device, CES_A. Actual structure of the test environment related to this scenario and also for the CES-to-CES communication, explained later in this chapter, is presented in the Chapter 9.

Because the client and the server are behind the same CES, there is no need for an additional DNS server. DNS queries are directed to CES_A and it responds to those.

At the beginning of the message flow the client and the server perform a three-way handshake. More information about this is presented in the Chapter 4.

The messages, related to the login and authorization, are only represented as one line as they are not as important for processing the packets as the packets that follow it.

Last TCP ACK coming from the client as a response for login and authorization is included in the following figure to show the Seq/Ack that are related to the following messages. In the messages Sequence number (Seq), Acknowledgement number (Ack) and the length of the packet (Len) are demoted as S, A and L.

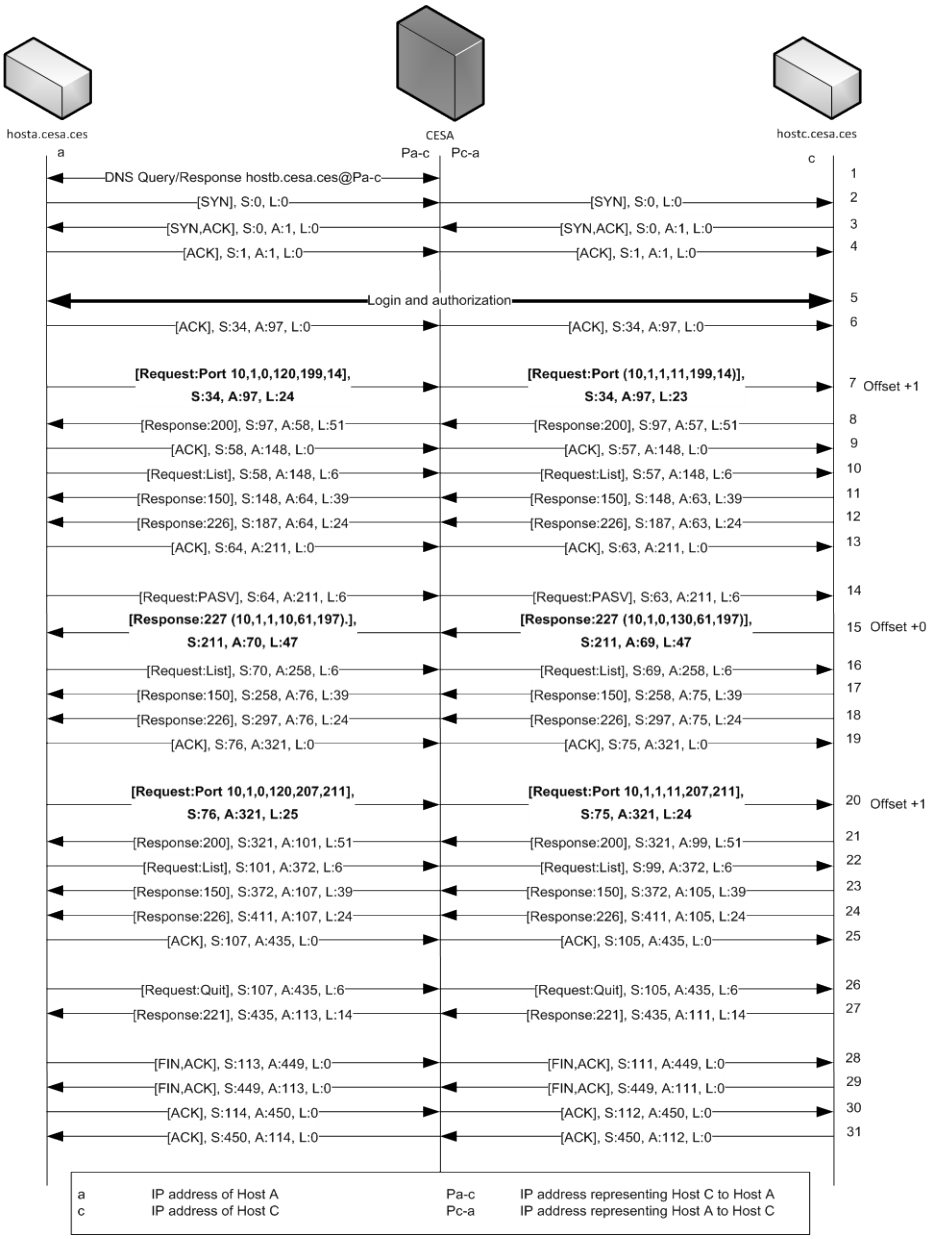

Figure 7.5: FTP communication in private network

In local FTP communication traffic flows between IP address of Host_A and allocated proxy address to reach Host_C. The same thing applies naturally for traffic between Host_C and CES_A. Traffic flow presented in the Figure 7.5 is explained next. Requesting the listing is only analyzed once as it is similar in all of the cases. The traces captured with Wireshark can be found in Appendix B.

1. Host_A queries for the address of the Host_C. CES_A responds with allocated proxy address.
2. Host_A initiates the three-way handshake.
3. Host_C responds with SYN-ACK.

4. Host_A responds with ACK.
5. All login and system information is compressed into one message flow as it is not critical information regarding the processing of the messages.
6. Last ACK sent by hosta gives starting sequence number 34 and acknowledgement number 97.
7. Host_A requests active connection with PORT command. Because the messages have different length the offset is increased by one (now +1).
8. Host_C responds with Response 200 (PORT command successful).
9. Host_A acknowledges with ACK.
10. Host_A requests for a file list.
11. Host_C sends the requested listing.
12. Host_C indicates that sending the data succeeded.
13. Host_A acknowledges with ACK.
14. Host_A requests passive connection with PASV command.
15. Host_C responds with Response 227 (PASV command successful). Because the messages have the same length the offset is unchanged (still +1).
16. Host_A acknowledges with ACK.
20. Host_A requests active connection with PORT command. Because the messages have different length the offset is increased by one (now +2).
21. Host_C responds with Response 200 (PORT command successful).
26. Host_A indicates that it wishes to terminate the communication.
27. Host_C responds with Response 221 (Goodbye).
28. Host_A is closing the application and sends [FIN, ACK]
29. Host_C is closing the application and sends [FIN, ACK]
30. Host_A received [FIN, ACK] and responds with [ACK]
31. Host_C received [FIN, ACK] and responds with [ACK]

The Table 7.1 gives information how the addresses are mapped to allow communication between two hosts in a same network. The table also includes the information about the final Offsets.

TABLE 7.1 FTP PRIVATE CONNECTION CESA

APP LAYER FTP STATUS				
Source		Destination		Offset
IP	Proxy IP	IP	Proxy IP	
a	Pa-c	c	Pc-a	-2
c	Pc-a	a	Pa-c	0

If the hosts in a private network were able to communicate directly with one another, the amount of the packets would be cut in half. Depending from the situation it could be even desirable to have this kind of option. Other than that, the TCP and FTP allow quite little room for changes.

7.3 CES-to-CES FTP communication

In addition to offering support for local FTP communication, the prototype also supports CES-to-CES communication. When the client and the server are located in different networks, the prototype needs to use two specific algorithms to handle the processing of outbound and inbound traffic.

Both of these algorithms are quite similar with the local private algorithm. Thus, they are explained in not so deep detail. Only the biggest differences are analyzed.

An important aspect that is good to remember is that in the CES-to-CES communication both CES devices have their own offsets. As with the private network this offset is related to ALG algorithm and should not be mistaken with the offset contained in the TCP header. This means that there are maximum four offsets. Two offsets in each CES device, one for each direction.

7.3.1 ALG algorithm for CES-to-CES FTP communication

The outbound algorithm performs in a similar way as the local private algorithm. The biggest difference is that the outbound algorithm replaces local IP with ID related to that IP address. For more detailed explanation of the other parts of the algorithm it is recommended to check the local private algorithm in section 7.2.

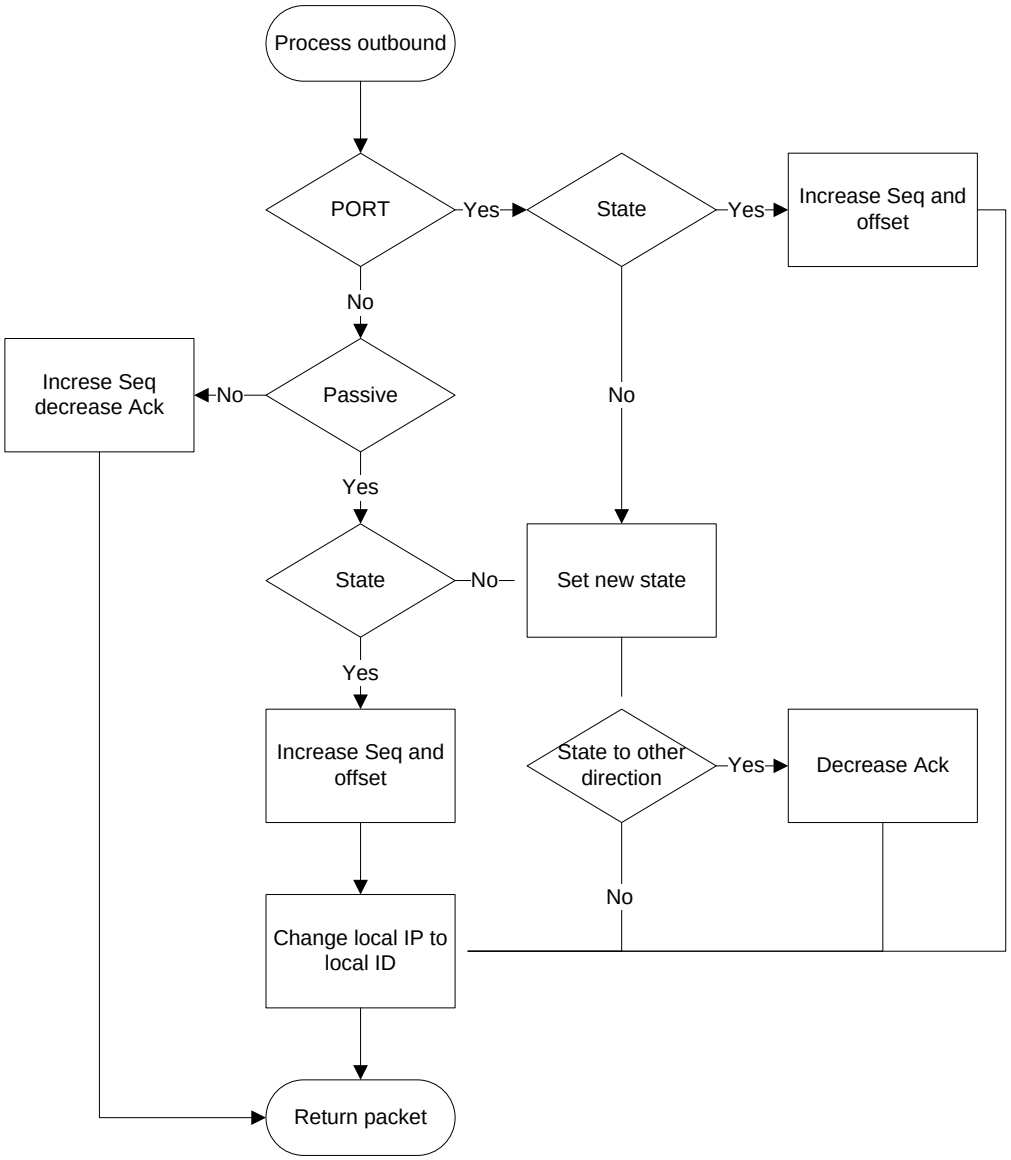

Figure 7.6: FTP ALG algorithm for outbound CES-to-CES communication

As with the outbound algorithm, the inbound algorithm, presented in Figure 7.7, reminds the local private algorithm. If some parts of the algorithm seem unclear, it is possible to check those at the beginning of this chapter.

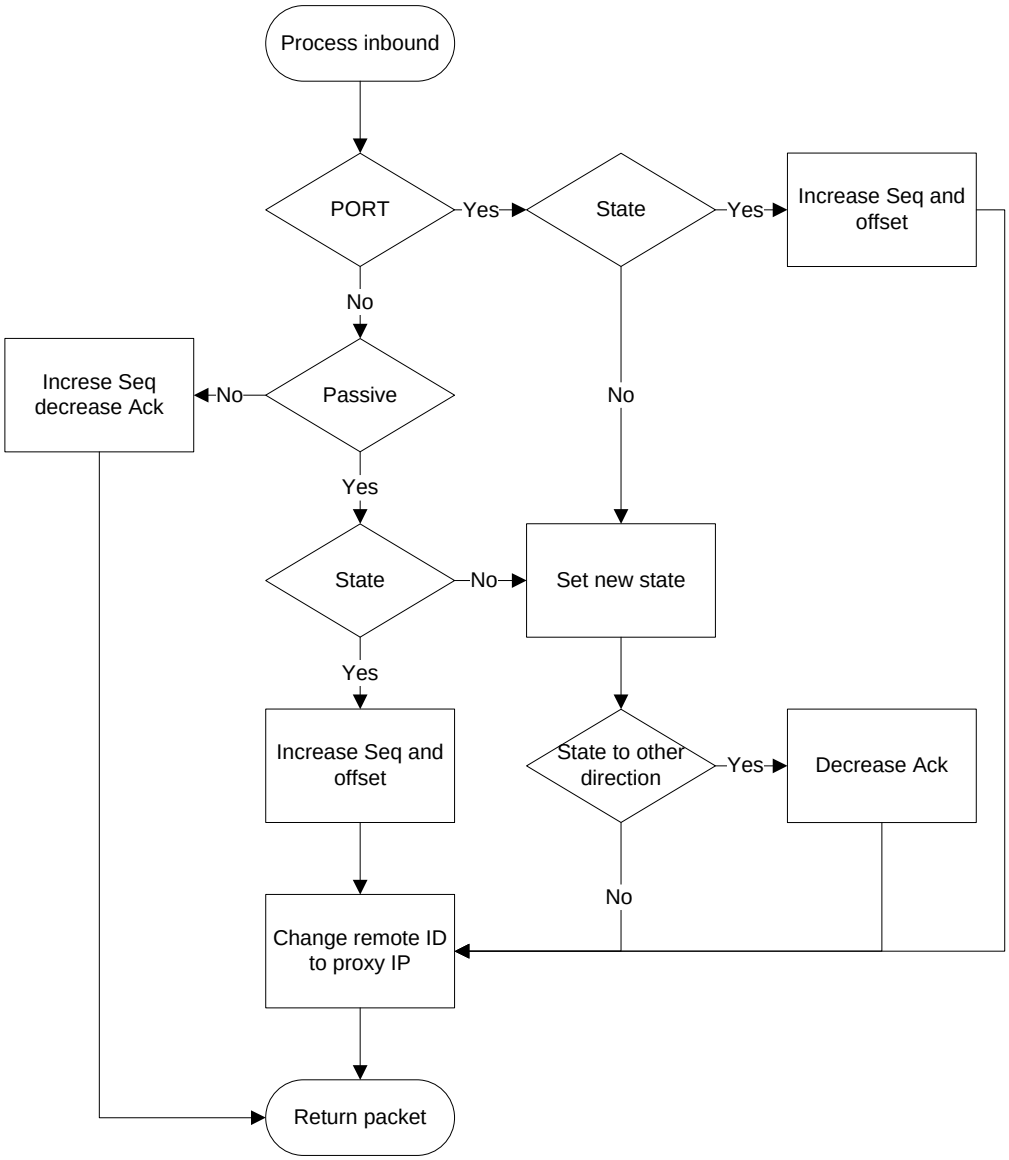

Figure 7.7: FTP ALG algorithm for inbound CES-to-CES communication

The biggest difference between this and other algorithms is at the end of the packet processing. Because the traffic is inbound, the remote ID, representing the source, is replaced by the proxy IP that is also related to the source.

7.3.2 Example run of FTP CES-to-CES communication

At the beginning of the communication the client queries for the address of the server. Then communication continues with three-way handshake. Login and authorization is not vital for message flow so it is compressed to just one line.

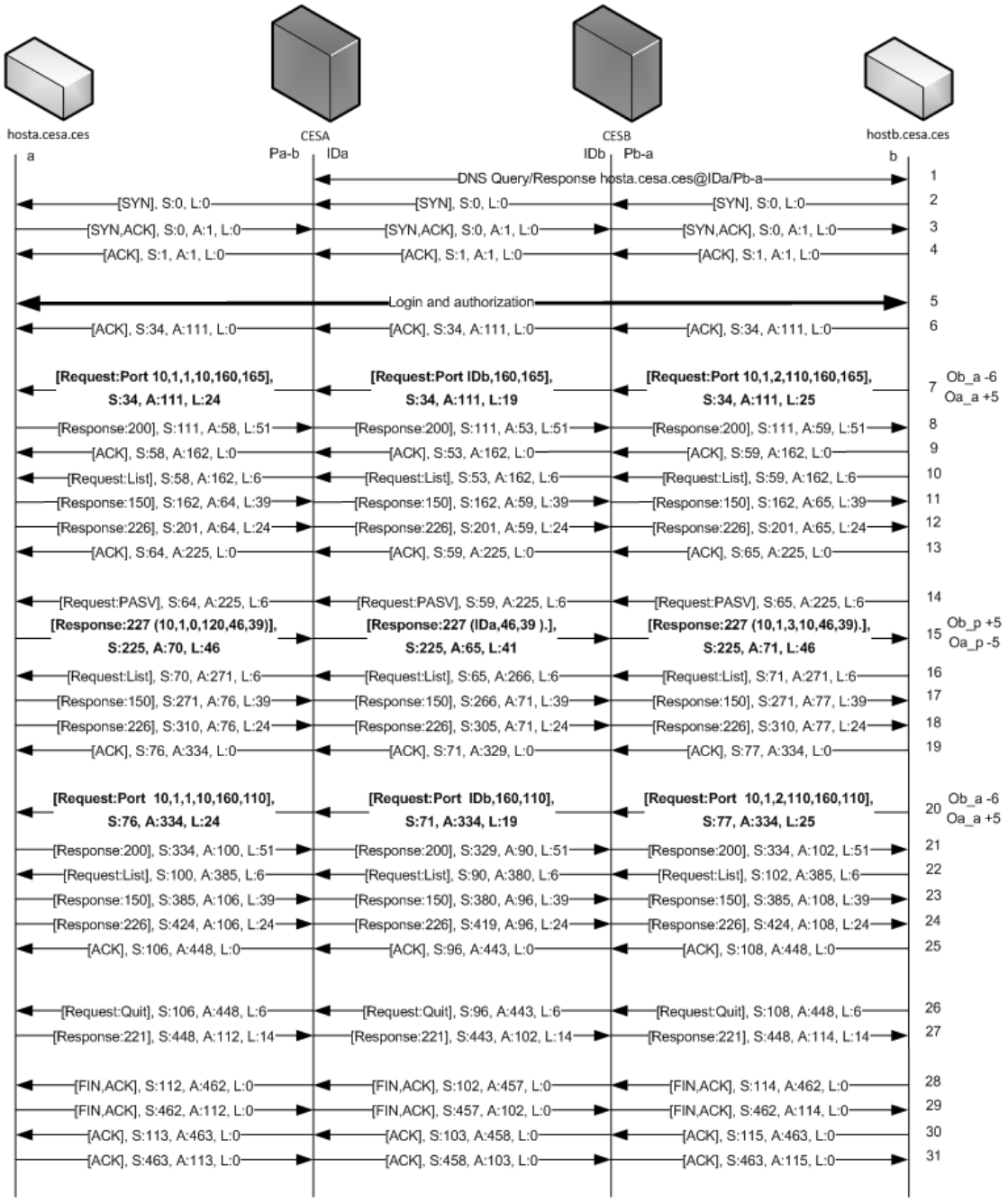

Figure 7.5: FTP communication CES-to-CES

FTP traffic between two hosts residing behind different CES devices requires more complex processing than the previously introduced local communication. In addition to proxy addresses IDs are needed because of the traffic between CESA and CESB.

Traffic flow from the example scenario shown in the Figure 7.5 is now systematically analyzed. It is important to notice that the DNS server, located in the public network, is not visualized in the figure. Four messages regarding List request are explained only once. In CES-to-CES communication there are four different offsets that need be tracked. One offset in each CES for each direction. Tables 7.2 and 7.3, after the explanation of the messages, show how mappings are created.

1. Host_B queries for the address of the hosta. CES_A responds with ID of the Host_A then CES_B responds to user with allocated proxy address.
2. Host_B initiates the three-way handshake.
3. Host_A responds with `SYN-ACK`.
4. Host_B responds with `ACK`.
5. All login and system information is compressed into one message flow as it is not critical information regarding the processing of the messages.
6. Last ACK sent by hostb gives starting sequence and acknowledge numbers.
7. Host_B requests active connection with `PORT` command. Because the messages have different length, the offset related to traffic in CES_B towards Host_A, (Ob_a) is decreased by six (now -6). For the same reason offset in it CES_A (Oa_A) is increased by five (now +5).
8. Host_A responds with `Response 200` (`PORT` command successful).
9. Host_B acknowledges with `ACK`.
10. Host_B requests for a file list.
11. Host_A sends the requested listing.
12. Host_A indicates that sending the data succeeded.
13. Host_B acknowledges with `ACK`.
14. Host_B requests passive connection with `PASV` command.
15. Host_A responds with `Response 227` (`PASV` command successful). Because the messages have different length, the offset related to traffic in CES_A towards Host_B, (Oa_p) is decreased by five (now -5). For same reason offset in CES_B (Ob_p) is increased by five (now +5).
16. Host_B acknowledges with `ACK`.
20. Host_B requests active connection with `PORT` command. Because the messages have different length, the offset related to traffic in CES_B towards Host_A, (Ob_a) is decreased by six (now -12). For same reason offset in CES_A (Oa_A) is increased by five (now +10).
21. Host_A responds with `Response 200` (`PORT` command successful).
26. Host_B indicates that it wishes to terminate the communication.
27. Host_A responds with `Response 221` (Goodbye).

28. Host_B is closing the application and sends [FIN, ACK]
29. Host_A is closing the application and sends [FIN, ACK]
30. Host_B received [FIN, ACK] and responds with [ACK]
31. Host_A received [FIN, ACK] and responds with [ACK]

TABLE 7.2 FTP CES-TO-CES CONNECTION CESA

APP LAYER FTP STATUS				
Source		Destination		Offset
IP/ID	Proxy IP/ID	IP/ID	Proxy IP/ID	
IDa	IDb	a	Pa-b	+10
a	Pa-b	IDa	IDb	-5

TABLE 7.3 FTP CES-TO-CES CONNECTION CESB

APP LAYER FTP STATUS				
Source		Destination		Offset
IP/ID	Proxy IP/ID	IP/ID	Proxy IP/ID	
b	Pb-a	IDb	IDa	-12
IDb	IDa	b	Pb-a	+5

In this scenario there are no packets that could be removed without changing the actual flow. The packets are transferred between the host and the CES device, then between the CES devices and finally from the CES to another host. Because of this the current implementation of the FTP Application Layer Gateway seems to work really well.

8. Test Environments

The tests to check whether the code is working properly are run in both virtual and physical environment. The virtual test environment is tested first to check any initial errors. After the prototype worked with the virtual setting, the testing continues with the physical test environment. This chapter also includes test settings from the clients and the servers.

8.1 Virtual Test Environment for SIP

A powerful desktop computer can support many virtual machines in parallel as is visualized in Figure 8.1. However, in most test cases the number of running virtual machines is between six and eight.

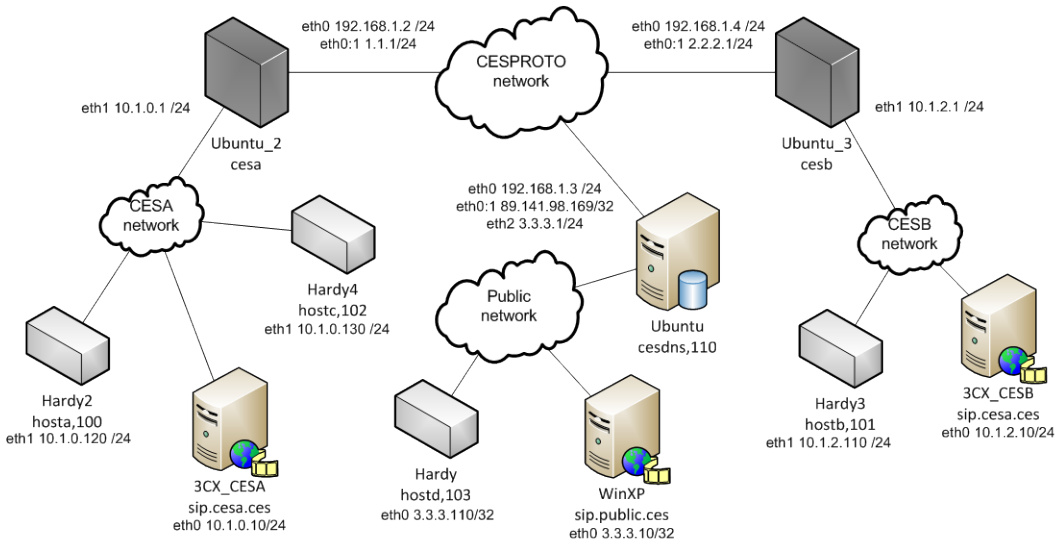

Figure 8.1: Structure of the SIP Virtual Test Environment

8.1.1 Clients

Clients in the virtual environment run on Ubuntu 8.04 Hardy Heron. This operating system allows tracking of the RTP traffic in VirtualBox while doing the testing. The client application in this case is Twinkle 1.1. The application supports configurations needed for various tests. In addition Twinkle performs with reasonable predictability which helps redoing the tests.

The tests done within this thesis require only two clients at a time. This explains the placement of the clients in the Figure 8.1. Two clients located in CESA network are used together for local tests. Different combinations of clients in CESA, CESB and public network can be applied to cover other test cases.

8.1.2 Servers

Servers in the virtual environment run on Windows XP. The 3CX phone system is used as software to provide necessary functionality. Browser based configuration of the server proved to be a really powerful tool for managing extensions.

The test environment uses three servers to provide enough options for different test settings. Usually only one server is running at a time.

8.1.3 Trunks

3CX has an option to create trunks between two servers. This means that the clients in one network can communicate with the clients residing in other network. In order for this to work both servers need to be preconfigured to allow an incoming trunk connection. Authentication ID selected to distinguish the connection between the servers must match. Thus, authentication IDs are also unique.

The problem with the trunks is that they do not support SIP packets containing "`MESSAGE`". Clients can communicate only via voice calls. This is a software specific restriction and even the prototype is still able to transfer these `MESSAGE`s towards the server, the servers just drop the packets.

8.2 Physical Test Environment for SIP

Because of limited budget the actual physical test environment is a bit smaller than the virtual environment, as we can see in Figure 8.2. As an additional feature, a laptop, not shown in the figure, was plugged in the proper network to allow similar tests as with the virtual environment.

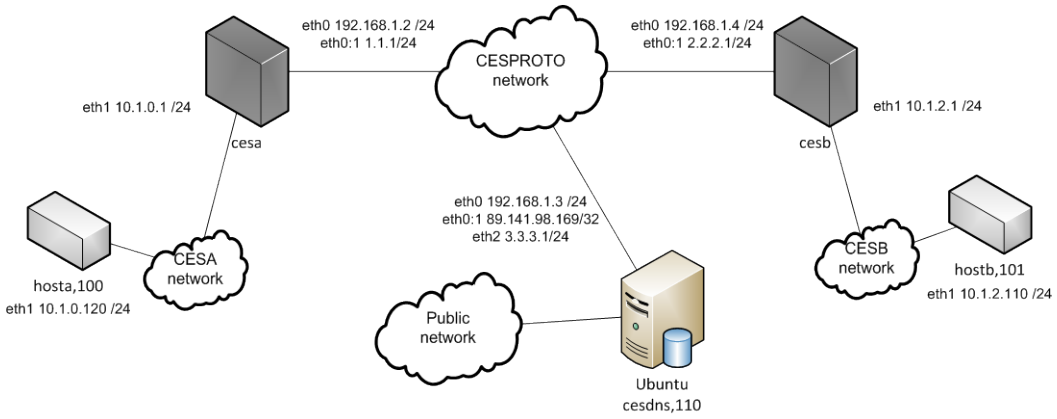

Figure 8.2: Structure of the SIP Physical Test Environment

8.2.1 Clients

The operating system in the physical test environment clients is Ubuntu 10.04 Lucid Lynx. Lucid performs a bit better than Hardy in the physical scenario. Because of the different operating system, the clients run a newer version of the application. Twinkle is still used, but the version is 1.4.

One of the clients was run on a laptop and connected to the network according to the on-going test. Software in the client was 3CX phone. This application is used, because it supports Windows and allows rather straight forward configuration. The only problem with this client is that registration and deregistration is not as simple as in Twinkle.

All of the clients are tested with microphone and speakers/headphones. In this test environment it is possible to receive actual RTP traffic in the form of speech. The testing is in that sense slightly easier than with the virtual environment.

8.2.2 Servers

Because of lack of physical machines to support thorough testing, virtual servers are connected to the proper network. This means connecting one server to one of the three possible networks to support the test that is currently in progress. Virtual machines running 3CX servers actually have two interfaces, one for the virtual and another for the physical network.

Thus, all the settings in the servers could be kept the same and this eased moving from the virtual testing to the physical testing. This solution provided an answer for the problem caused by the limited amount of physical machines.

8.2.3 Trunks

Trunks work in a similar way as in the virtual test environment because, as previously described, the settings in the servers match the ones used for the virtual testing. Therefore they are not discussed in detail here. Rechecking chapter 7.1.3 might be useful for more specific details.

8.3 SIP test scenarios

In the following table 8.1 there are the test scenarios related to SIP Application layer gateway testing that have been executed. The tests are done in both the virtual and in the physical environment. Such tests that mirror one of the scenarios already listed in the table are skipped. This is done because the same prototype is anyway run in both CES devices.

All of the tests can be done using either the algorithms performing operations with IP addresses or with the algorithms using FQDNs. More information about the algorithms used by the SIP ALG can be found in the chapter 6.

In the table a couple of the test scenarios are marked with the asterisk. These test scenarios have some problems and the prototype does not offer full support for those. The results of the tests are discussed in more detail in the chapter 9. That chapter also considers these problems encountered during the testing.

TABLE 8.1 DIFFERENT TEST SCENARIOS FOR SIP

Number	Caller	Registered	Callee	Registered	First test	Second test	Description
1	hosta	CESA	hostc	CESA	Virtual	Physical	Private
2	hostc	CESA	hosta	CESA	Virtual	Physical	Private
3	hosta	CESA	hostb	CESA	Virtual	Physical	CES-to-CES
4	hostb	CESA	hosta	CESA	Virtual	Physical	CES-to-CES
5	hosta	Public	hostb	Public	Virtual	Physical	Legacy
6	hostb	Public	hosta	Public	Virtual	Physical	Legacy
7	hosta	Public	hostc	Public	Virtual	Physical	Legacy
8	hostc	Public	hosta	Public	Virtual	Physical	Legacy
9	hosta	Public	hostd	Public	Virtual	Physical	Legacy
10	hostd	Public	hosta	Public	Virtual	Physical	Legacy
11	hosta	CESA	hostd	CESA	Virtual	Physical	Private/legacy
12	hostd	CESA	hosta	CESA	Virtual	Physical	Private/legacy

13	hostd	CESA	hostb	CESA	Virtual	Physical	C2C/legacy*
14	hostb	CESA	hostd	CESA	Virtual	Physical	C2C/legacy*
15	hosta	CESA	hostb	CESB	Virtual	Physical	Trunk
16	hostb	CESB	hosta	CESA	Virtual	Physical	Trunk
17	hosta	CESA	hostd	Public	Virtual	Physical	Trunk
18	hostd	Public	hosta	CESA	Virtual	Physical	Trunk
19	hosta	CESA	hostc	Public	Virtual	Physical	Trunk
20	hostc	Public	hosta	CESA	Virtual	Physical	Trunk
21	hosta	CESA	hostc	CESB	Virtual	Physical	Trunk
22	hostc	CESB	hosta	CESA	Virtual	Physical	Trunk
23	hosta	CESB	hostb	CESA	Virtual	Physical	Trunk*
24	hostb	CESA	hosta	CESB	Virtual	Physical	Trunk*

8.4 Virtual Test Environment for FTP

FTP uses a part of the virtual environment used for SIP testing. Because the tests are either local or CES-to-CES, only some of the virtual machines need to be run compared to the tests related to the SIP.

Actual structure of the virtual FTP test environment is presented in the Figure 8.3. Because the FTP Application Layer Gateway related to this thesis only considered FTP traffic in private network and between a client and a server located behind different CES devices, there is no need for a host in the public network. This kind of public test scenario would just repeat what current NATs do today.

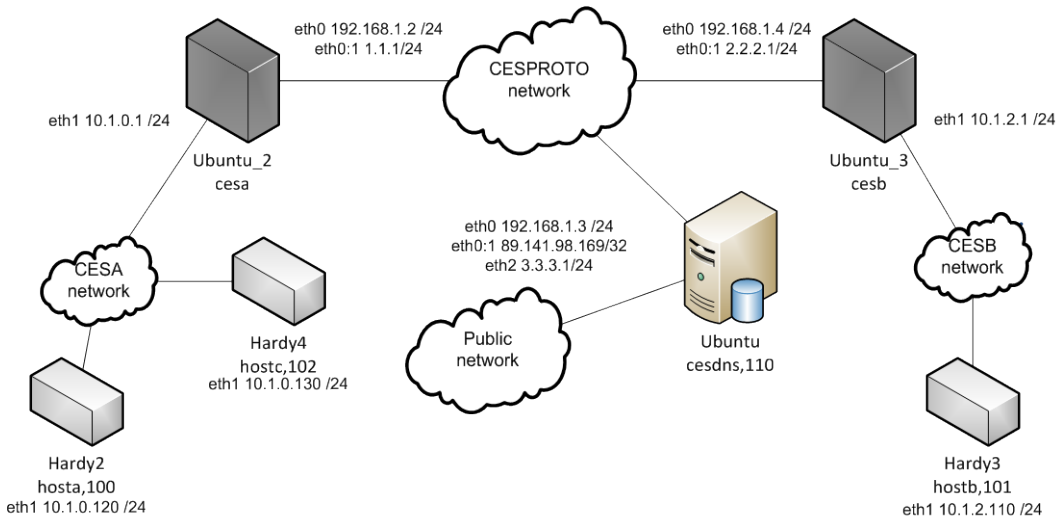

Figure 8.3: Structure of the FTP Virtual Test Environment

8.5 Physical Test Environment for FTP

Physical test environment is the same in FTP as in SIP testing. It can be seen in Figure 8.2. Setting up client and server for FTP is even easier than with SIP, so it is natural that the same environment is used for both protocols.

8.6 FTP test scenarios

Testing the FTP Application layer does not need as many different tests as SIP. Tests related to FTP are listed in the table 8.2.

TABLE 8.2 DIFFERENT TEST SCENARIOS FOR FTP

Number	Client	Located	Server	Located	First test	Second test	Description
1	hosta	CESA	hostc	CESA	Virtual	Physical	Private
2	hostc	CESA	hosta	CESA	Virtual	Physical	Private
3	hosta	CESA	hostb	CESB	Virtual	Physical	CES-to-CES
4	hostb	CESB	hosta	CESA	Virtual	Physical	CES-to-CES

9. Evaluation of the Results

In this chapter the results from the testing are analyzed. Results from all of the tests are first checked to get an overall picture from the testing. Then some of the cases are described in more detail. Problems encountered during coding and testing are also analyzed in this chapter. At the end of the chapter the suitability of the solution for Customer Edge Switching (CES) is considered.

9.1 Results of the SIP Test Cases

The following table presents results of the different test cases that were run in both virtual and physical environment.

TABLE 9.1 RESULTS OF THE DIFFERENT TEST CASES USING IP ADDRESSES

Number	Caller	Registered	Callee	Registered	Type	Algorithms	Success	Description
1	hosta	CESA	hostc	CESA	Both	IP	YES	Private
2	hostc	CESA	hosta	CESA	Both	IP	YES	Private
3	hosta	CESA	hostb	CESA	Both	IP	YES	CES-to-CES
4	hostb	CESA	hosta	CESA	Both	IP	YES	CES-to-CES
5	hosta	CESB	hostc	CESB	Both	IP	NO	CES-to-CES
6	hostc	CESB	hosta	CESB	Both	IP	NO	CES-to-CES
7	hosta	Public	hostb	Public	Both	IP	YES	Public
8	hostb	Public	hosta	Public	Both	IP	YES	Public
9	hosta	Public	hostc	Public	Both	IP	YES	Public
10	hostc	Public	hosta	Public	Both	IP	YES	Public
11	hosta	Public	hostd	Public	Both	IP	YES	Public
12	hostd	Public	hosta	Public	Both	IP	YES	Public
13	hosta	CESA	hostd	CESA	Both	IP	YES	Private/ Public
14	hostd	CESA	hosta	CESA	Both	IP	YES	Private/ Public
15	hostd	CESA	hostb	CESA	Both	IP	NO	C2C/Public
16	hostb	CESA	hostd	CESA	Both	IP	NO	C2C/Public
17	hosta	CESA	hostb	CESB	Both	IP	YES	Trunk
18	hostb	CESB	hosta	CESA	Both	IP	YES	Trunk
19	hosta	CESA	hostd	Public	Both	IP	YES	Trunk
20	hostd	Public	hosta	CESA	Both	IP	YES	Trunk
21	hosta	CESA	hostc	Public	Both	IP	YES	Trunk
22	hostc	Public	hosta	CESA	Both	IP	YES	Trunk
23	hosta	CESA	hostc	CESB	Both	IP	YES	Trunk

24	hostc	CESB	hosta	CESA	Both	IP	YES	Trunk
25	hosta	CESB	hostb	CESA	Both	IP	NO	Trunk
26	hostb	CESA	hosta	CESB	Both	IP	NO	Trunk

TABLE 9.2 RESULTS OF THE DIFFERENT TEST CASES USING FQDNS

Number	Caller	Registered	Callee	Registered	Type	Algorithms	Success	Description
1	hosta	CESA	hostc	CESA	Both	FQDN	YES	Private
2	hostc	CESA	hosta	CESA	Both	FQDN	YES	Private
3	hosta	CESA	hostb	CESA	Both	FQDN	YES	CES-to-CES
4	hostb	CESA	hosta	CESA	Both	FQDN	YES	CES-to-CES
5	hosta	CESB	hostc	CESB	Both	FQDN	YES	CES-to-CES
6	hostc	CESB	hosta	CESB	Both	FQDN	YES	CES-to-CES
7	hosta	Public	hostb	Public	Both	FQDN	YES	Public
8	hostb	Public	hosta	Public	Both	FQDN	YES	Public
9	hosta	Public	hostc	Public	Both	FQDN	YES	Public
10	hostc	Public	hosta	Public	Both	FQDN	YES	Public
11	hosta	Public	hostd	Public	Both	FQDN	YES	Public
12	hostd	Public	hosta	Public	Both	FQDN	YES	Public
13	hosta	CESA	hostd	CESA	Both	FQDN	YES	Private/ Public
14	hostd	CESA	hosta	CESA	Both	FQDN	YES	Private/ Public
15	hostd	CESA	hostb	CESA	Both	FQDN	NO	C2C/ Public
16	hostb	CESA	hostd	CESA	Both	FQDN	NO	C2C/Public
17	hosta	CESA	hostb	CESB	Both	FQDN	YES	Trunk
18	hostb	CESB	hosta	CESA	Both	FQDN	YES	Trunk
19	hosta	CESA	hostd	Public	Both	FQDN	YES	Trunk
20	hostd	Public	hosta	CESA	Both	FQDN	YES	Trunk
21	hosta	CESA	hostc	Public	Both	FQDN	YES	Trunk
22	hostc	Public	hosta	CESA	Both	FQDN	YES	Trunk
23	hosta	CESA	hostc	CESB	Both	FQDN	YES	Trunk
24	hostc	CESB	hosta	CESA	Both	FQDN	YES	Trunk
23	hosta	CESB	hostb	CESA	Both	FQDN	NO	Trunk
24	hostb	CESA	hosta	CESB	Both	FQDN	NO	Trunk

The biggest difference in the results is in the test cases 5 and 6. These tests show that FQDN algorithms succeed better in different situations than the algorithms operating with the IP addresses. The FQDN algorithms are also easier to understand as can be seen in the Chapter 6.

9.2 SIP Test Cases in More Detail

In this section some of the test scenarios are presented in more detail. Because the whole network is not used in any of the tests, it is easier to understand individual tests when they are shown with the machines needed for the testing.

One figure can provide information about more than one specific test. As shown in the previous tables some of the test scenarios, particularly public and trunk, have many different configurations. In these situations, the figure is analyzed in more detail.

9.2.1 Public

The case of public communication was tested by three different scenarios. Figure 9.1 explains how individual devices are located in the tests. In the first test Host_A located in CES_A network, and Host_B located in the CES_B network, registered to the SIP server located in the public network.

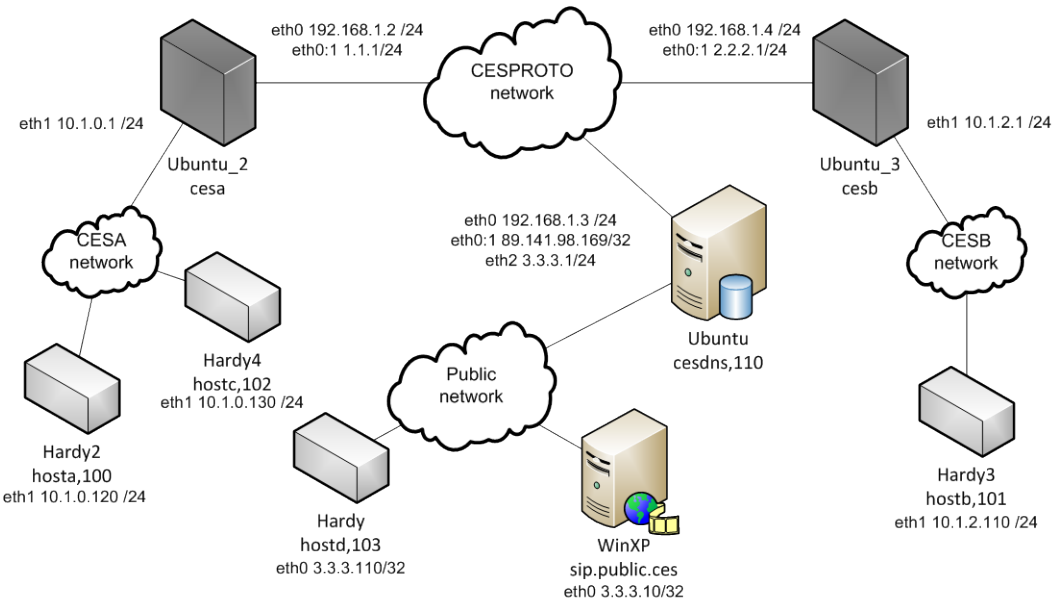

Figure 9.1: Structure of public test setting

The second test had both clients in the same network. In this case, Host_A and Host_C were in the CES_A network. Both clients registered again to the public server. In the third test scenario, the communication between a local and a public

client was tested. This means using Host_A and Host_D. Registration is similar to previous scenarios.

Each of the public test scenarios worked without any problems. There was no unnecessary resource allocation or problems with the real time traffic flowing end to end. Cancelling, rejecting and terminating the call did not have any issues either.

9.2.2 Private network

Private connections were tested in the CES_A network with Host_A and Host_C registering to SIP server in the same network. The structure of this relatively simple test scenario is shown in the Figure 9.2. No problems were detected during the tests.

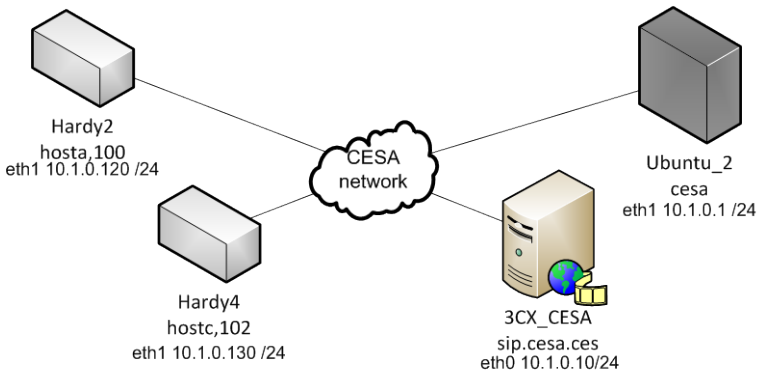

Figure 9.2: Structure of private network test setting

9.2.3 Private-Public

The scenario, combining the local private and the public algorithms, has one client in the private network and another in the public network. The setting for this particular test is presented in the Figure 9.3.

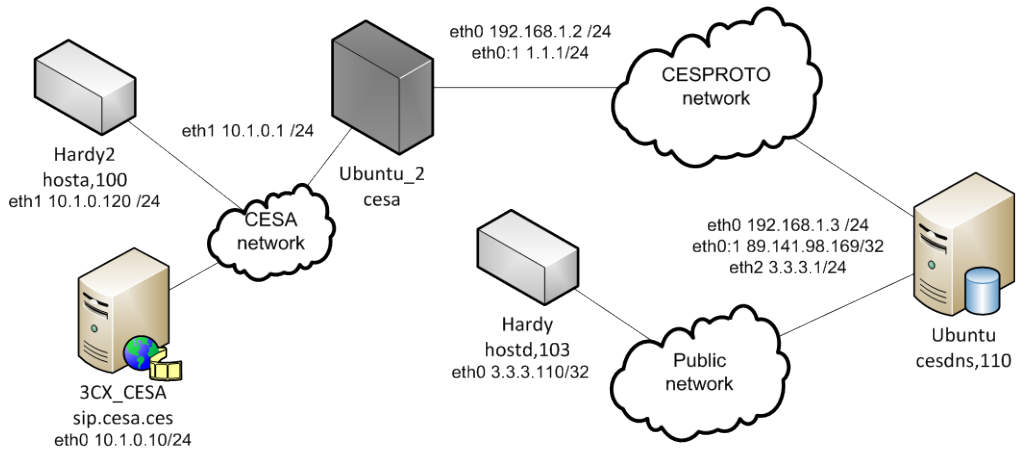

Figure 9.3: Structure of private-public test setting

Host_A registers locally to server in the CES_A network. Other client, Host_D, is located to public network, but also registers to the server in the CES_A network. Actual testing did not reveal any problems.

9.2.4 CES-to-CES

Communication between clients located behind different CES devices was a really important test for this thesis. Figure 9.4 explains the situation. In this scenario a SIP server is located in CES_A network. Clients from CES_A network and from CES_B network register to it. The actual testing was performed without any major issues.

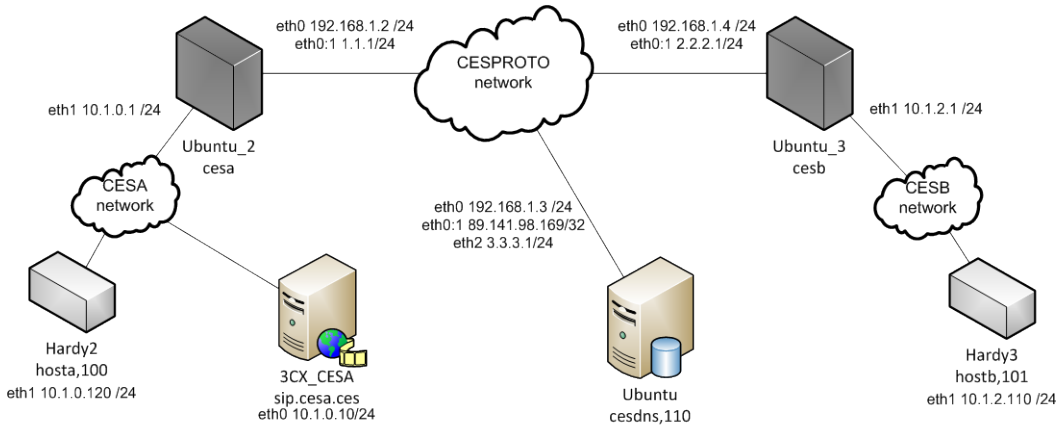

Figure 9.4: Structure of CES-to-CES test setting

Only aspect causing problems was configuring the IDs. In a case that CES devices have, for some reason, the same ID configured to their database, the communication naturally does not work. However, it should be remembered that this happened only when updating the prototype and does not occur in the final version.

FQDN approach offers support for wider range of the test cases. This can be verified from the tables 9.1 and 9.2. Tests 5 and 6 were successful with FQDN as the hosts can query for the addresses when they receive messages containing domains.

9.2.5 Trunks

For a thorough testing, the trunks had several test scenarios. Following tests match the order shown in the table 8.1. Devices used for the testing are visualized in the Figure 9.5. The tests were done with a host acting first as a caller then as a callee and vice versa.

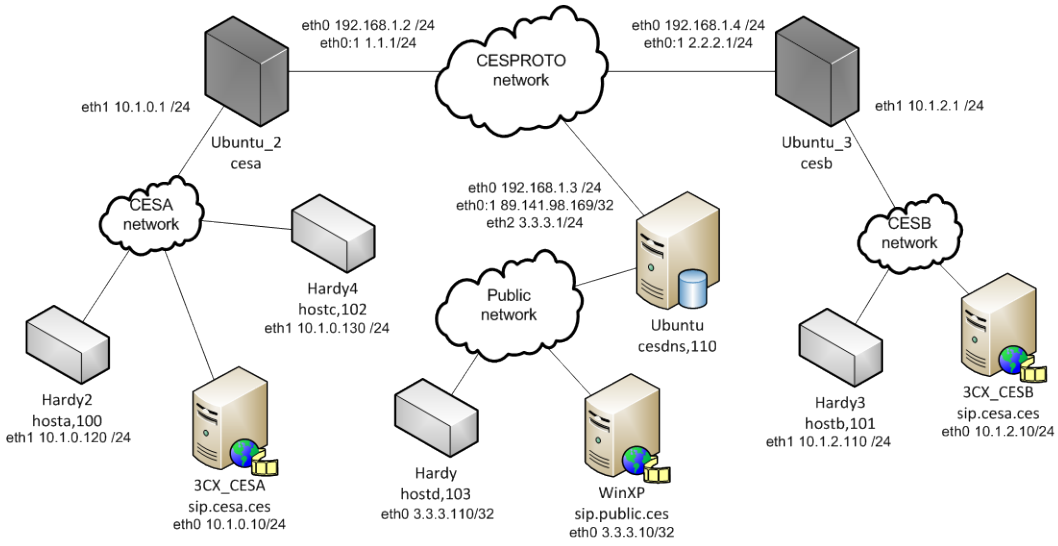

Figure 9.5: Structure of Trunk test setting

At first the clients and the servers were placed in the same network. Host_A registered to the server in CES_A network and Host_B registered to the server in CES_B network. The first session was initiated by the Host_A, after this the test was repeated with Host_B as a caller. This test did not cause any problems.

In the second trunk test, the trunk between private and public server was tested. Host_A registered again to the server in CES_A network, while Host_D registered to the public SIP server. The test succeeded without any errors.

The third scenario had clients in the same network registering to different servers. Host_A registered to the SIP server behind CES_A and Host_C registered to the public server. This test was also successful.

The test scenario that was tested as fourth option had again clients in the same network. The only difference between this and the previous test was that Host_D was registered to the server in the CES_B network. This scenario proved to work.

The fifth scenario was partly done to check how much traffic the prototype can currently support. Registering Host_A to the server in CES_B network and registering Host_B to server in the CES_A network might not be a common situation. However, by doing this the traffic caused by the communication between the clients was much more than in any of the earlier scenarios. Each RTP packet needs to be processed three times in both CES devices before it reaches other end. The sound quality in this test was really bad and therefore it was considered as failed. The signaling still worked and the session could be terminated.

This kind of scenario can happen in the real world only after bad configurations. Other than that, it is not likely that traffic is routed and voice is carried between the servers. Usually the client transmits the messages to the public SIP proxy that sends the messages to another client. The RTP traffic flows directly between the hosts.

9.2.5 CES-to-CES/Public

Even one more unlikely scenario was tested. In this test setting, shown in the Figure 9.6, the server was placed in the CES_A network. One client was in the public network and another behind CES_B.

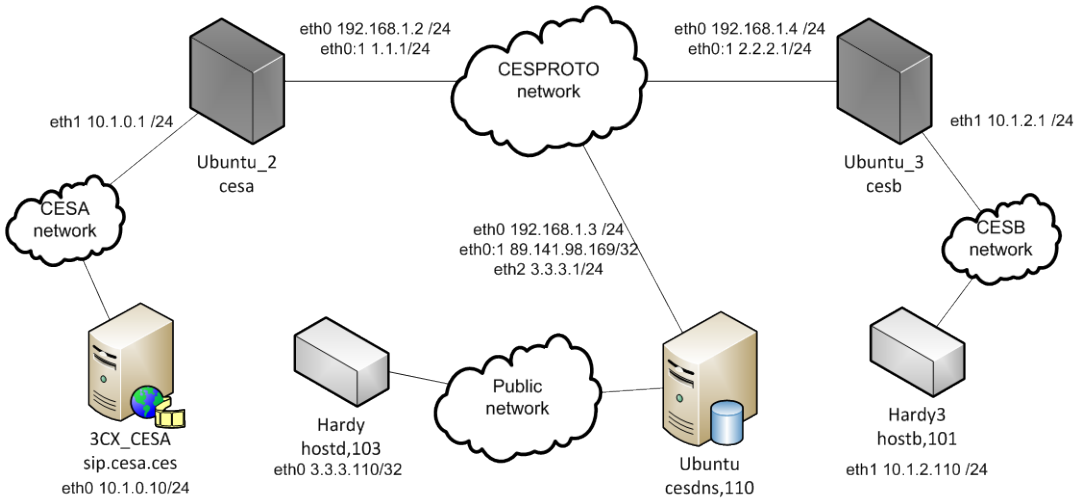

Figure 9.6: Structure of CES-to-CES/Public test setting

This test failed, because the RTP traffic did not flow end to end in both cases. Meaning that, when Host_D initiated the call, everything worked fine. When Host_B initiated the call, the address used to transfer RTP traffic from Host_B to Host_D was different than the address contained in the DNS response. Thus, RTP worked one way but was blocked when trying to enter CES_B network.

The previous result happened with the algorithms using FQDNs. The algorithms operating with the IP addresses do not perform as well. Therefore, at the moment the prototype does not offer support for the scenarios like this. When the prototype is developed to support mobility, this kind of scenarios might occur more often and will need to be studied further.

9.3 Results of FTP test cases

TABLE 9.3 RESULTS OF THE DIFFERENT FTP TEST CASES

Client	Located	Server	Located	Type	Result	Description
hosta	CESA	hostc	CESA	Virtual	OK	Local
hosta	CESA	hostc	CESA	Physical	OK	Local
hosta	CESA	hostb	CESB	Virtual	OK	CES-to-CES
hosta	CESA	hostb	CESB	Physical	OK	CES-to-CES

9.3.1 Local

Test setting for the local testing is demonstrated in the Figure 9.7. One of the hosts acts as a client and another as a server. DNS server is not needed as the queries are directed to CES_A.

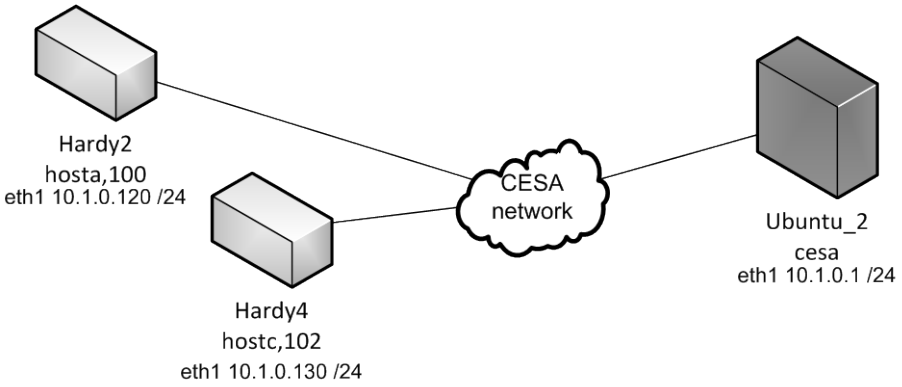

Figure 9.7: Structure of FTP local test setting

Local FTP communication worked with both active and passive. Even swapping from one to another did not provide any problems. It is also possible to have multiple connections between two hosts.

9.3.1 CES-to-CES

The machines related to the FTP Application Layer Gateway testing in CES-to-CES communication are shown in the Figure 9.8. The hosts are located behind different CES devices. DNS server is used in the scenario to handle DNS queries performed by the CES_A and CES_B. This gives a host located in the private network a proxy address to communicate with the host in the other private network.

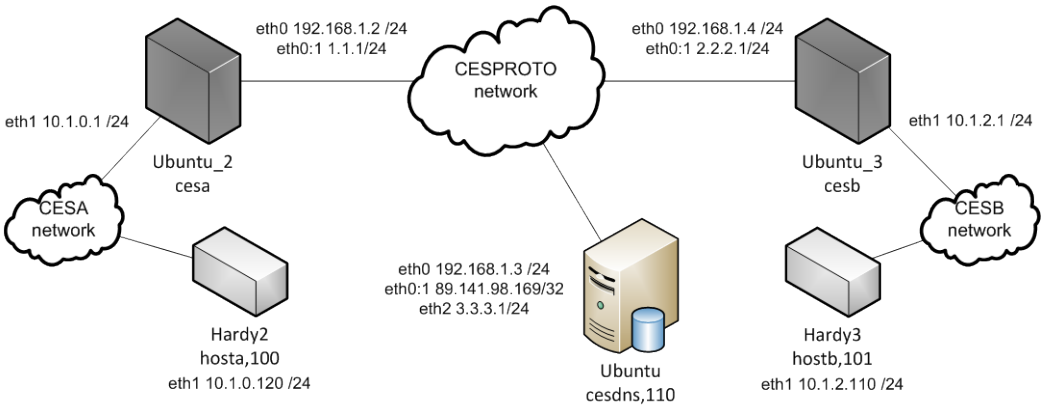

Figure 9.8: Structure of FTP CES-to-CES test setting

As with the local FTP communication the CES-to-CES communication also works with both active and passive. Changing from one method to another does not introduce any errors. Simultaneous connections work too without problems.

9.4 Problems encountered

Using VirtualBox as a test environment was rather challenging at the beginning. The biggest issue, which was relatively hard to spot, was that configuring audio drivers in virtual machines in a certain way caused virtual machines to crash when running SIP clients. One setting, that worked well, was when the host driver was set to Null Audio Driver and controller as ICH AC97. This prevented the problem from happening.

Partly related to the above problem, also the behavior of RTP traffic seems to differ in the virtual environment, which is caused by the operating system that is running in the client. The client software actually had different versions for both of the operating systems. The difference between Lucid and Hardy is described more thoroughly in Chapter 8. Briefly, running tests on Lucid provides no visible RTP traffic, which naturally causes challenges for testing and troubleshooting.

Even though Twinkle was adopted as the main client softphone, testing was performed with other software too. Using different clients proved to be quite a task as they do not have similar configurations. More importantly, they process RTP traffic differently. Twinkle, as an example, sends RTP straight after sending 200 OK, while Ekiga waits for ACK before sending any voice. However, spotting this is useful as the prototype can offer better support for various clients.

When updating the code in virtual or physical machines, the configuration file of the CES needs to be changed. This can cause mysterious errors, if it is forgotten. As an example one error that occurred was the usage of identical IDs. CES was trying to locate a client with a specific ID in a wrong network, because of this configuration error. At least it gives a warning how important it is to have large a enough pool of IDs.

Some difficulties emerged when implementing algortihms to change IP addresses within SDP. Normal find() methods in Python can find suitable matches even when they are not desirable. As an example when searching matches for 10.1.0.1 it is possible to get a match with first part of 10.1.0.100. If the first part is replaced, the result is something not even close to target value. After tuning the algorithms to take into account this and similar situations it is possible to avoid these mismatches.

Creating the algorithm for SIP communication in a private network, when using IP addresses, was probably the most complicated task. Allocating addresses at a certain point and creating tables to record ongoing `INVITE`s proved to be quite challenging. The biggest problem was that when an outbound CES receives the first `INVITE` it cannot know if the message is later going towards another CES or towards another user in the same network. Because of this, allocating resources had to be done with utmost care.

When a caller sends `CANCEL` or callees send `REJECT`, all the previously allocated resources and lists need to be released and cleared. Otherwise, every following call attempt will allocate an extra address and therefore consume resources needlessly. This can also be seen as a light approach for security improvement against Denial of Service (DoS) attacks. The attacker cannot consume all the resources if they are released straight away. Usually the timeout regarding the SIP communication is set to 3600 seconds.

FTP was a bit easier than SIP, but still had some minor issues. Both passive and active performed as expected, but when switching from one to another the initial version stopped working. It required some drawings and extensive analysis of the traces to finally solve the problem. The problem was that the offset information from the previous connection and seq/ack related to it were still important with the following connection. In addition to increasing seq, CES needs to check if there has been traffic to another direction, because it gives information whether ack needs to be changed.

As the previous paragraph mentioned, managing seq/ack was not a very easy task. It gets even more complicated when communication happens between two CES devices.

However, similar approach to change both seq/ack is just required twice. Both of these aspects of FTP are more thoroughly presented in Chapter 7.

9.5 Security considerations

Both Application Layer Gateways presented here modify existing data within packets. This is against the idea of end-to-end security. However, if a user regards it's serving CES device as a trusted entity, there should be no issues with end-to-end. If a CES needs to provide support for various application layer protocols, it is required to do processing that tampers with payload.

Returning addresses to the address pool prevents consuming resources by flooding SIP INVITEs. Also shorter timeouts with addresses allocated for RTP communications help to prevent abusing allocation of the IP addresses.

10. Conclusions

The objective of this thesis was to develop Application Layer Gateways for both Session Initiation Protocol and File Transfer Protocol. According to the evaluation the ALGs developed were highly successful. The developed prototype shows that by designing Application Layer Gateways for protocols that use IP addresses as identifiers, such as SIP or FTP, existing applications can work over a network where Customer Edge Switching has been deployed. Because the testing was done first in virtual then in the physical scenario, the test results are undisputed.

Customer Edge Switching as a method of network edge traversal allows communication across address realm boundaries. We can use one CES device to serve a large number of private address realms while both clients and servers can be placed and reached behind a CES. In fact, we chose to place each host into a separate private address space. This means that they can communicate with each other only through the CES. It is easy to support simpler scenarios where some of the hosts can communicate directly if this is preferred.

We have assumed that CES needs to support the initial set of end to end and network monitoring protocols. These include: FTP, HTTP, HTTPS, ICMP, SIP, SSH, telnet and the usual e-mail protocols, POP and IMAP. This thesis contributes toward that goal by providing a solution for FTP and SIP.

The target of the Customer Edge Switching is to provide also connectivity for mobile and wireless devices located behind a CES. Thus, CES acts as a new kind of collaborative firewall provides interrupt driven access for battery powered devices. Providing methods to support mobility still need to solve problems like the one encountered in the SIP test scenario 9.2.5. Because SIP ALG was done with two types of algorithms, it can offer useful information if other protocols need to be implemented in the prototype.

This thesis provided information about one aspect of the CES prototype. The current version is not integrated with the latest development of the Customer Edge Traversal Protocol (CETP). A future integration opens new tasks for the study of security and trust in connection with SIP and FTP. For this alternative policies need to be designed and tested for SIP and FTP communication.

The code developed for the prototype has some comments in it to help understanding the functionality of the individual algorithms. The algorithms are not tuned for high

performance and therefore there might be some room for improvements. This task might become reasonable if the coding of the prototype is changed from Python to C/C++.

Another thing that could be implemented is the caching of the messages. As seen in some of the SIP test scenarios, the amount of DNS queries can be quite high. Caching the information could prevent sending at least some of these queries further than the CES device serving the private network.

As this thesis was a part of the bigger project, adapting it to the Customer Edge Traversal Protocol might require some additions. This could mean adding another TLV to CETP for supporting the media identity. However, this aspect needs to be solved later.

References

[1] Audet, F. & Jennings, C. 2007 Network Address Translation (NAT) Behavioral Requirements for Unicast UDP. RFC 4787

[2] Clark, D. 30-31, 2007 MIT Communications Futures Program, Bi-annual meeting. Philadelphia. PA.

[3] Farinacci, D. Fuller, V. Meyer, D. Lewis, D. 2009 Locator/ID Separation Protocol (LISP). draft-farinacci-lisp-12.txt, Work-in-progress

[4] Handley, M. & Jacobson, V. & Perkins, C. 2006 Session Description Protocol (SDP). RFC 4566.

[5] Information Sciences Institute, University of Southern California. 1981. Transmission Control Protocol. RFC 793

[6] Johnston, A. & Schutzer, D. 2009 SIP: Understanding the Session Initiation Protocol (3rd edition) Norwood, MA, USA: Artech House.

[7] Kantola, R. 2009. Customer Edge Switching – A Trust-to-Trust Architecture for the Internet. Helsinki University of Technology, Department of Communications and Networking.

[8] Kantola, R. 2010. Implementing Trust-to-Trust with Customer Edge Switching, in press for AMCA in connection with AINA.

[9] Kantola, R. & Beijar, N. & Yan, Z. & Pahlevan, M. 2012. Customer Edge Traversal Protocol (CETP), [Retrieved 29.3.2012], Available: http://www.re2eee.org

[10] Llorente, J. 2012. Private Realm Gateway, M.Sc thesis, Comnet/Aalto. Work-in-progress.

[11] Luoma, M. & Kantola, R. & Manner, J. 2010. Future Internet is by Ethernet. World Computer Congress, Brisbane, Australia

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

[13] Pahlevan, M. 2012 Customer Edge Traversal Protocol, M.Sc thesis, Comnet/Aalto. Work-in-progress

[14] Postel, J. & Reynolds, J. 1985. File Transfer Protocol (FTP). RFC 959

[I5] Rekhter, Y. & Moskowitz, B. & Karrenberg, D. & de Groot, G. J. & Lear, E. 1996 Address Allocation for Private Internets RFC 1918 (et al.)

[16] Rosenberg, J., 2006. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245

[17] Rosenberg, J. & Mahy, R. & Matthews, P. 2010 Traversal Using Relays around NAT (TURN). RFC 5766

[18] Rosenberg, J. & Schulzrinne, H. & Camarillo, G. & Johnston, A. & Peterson, J. & Sparks, R & Handley, M. & Schooler, E. 2002 SIP: Session Initiation Protocol. RFC 3161

[19] Rosenberg, J. & Mahy, R. & Matthews, P. & Wing, D. 2008 Session Traversal Utilities for NAT (STUN). RFC 5389

[20] Ryynänen, J. 2008 Routed End-to-End Ethernet Network – Proof of Concept. M.Sc. Thesis Comnet/TKK, Espoo

[21] Srisuresh, P. & Egevang, K. 2001 Traditional IP Network Address Translator (Traditional NAT). RFC 3022

[22] Srisuresh, P Holdrege, M. 1999 IP Network Address Translator (NAT) Terminology Considerations. RFC 2663

[23] Virtanen, L. 2009 Communicating Globally Using Private IP Addresses, M.Sc thesis, Comnet/TKK, Espoo

Appendix A

SIP CES-to-CES connection

Some packets containing RTP data have been removed from the following traces to keep the size of the tables reasonable. Also any possible messages regarding ARP are not shown as they are not directly related to the SIP. Real-time traffic is visible in the tables with dark gray background. DNS queries are shown with lighter shade.

TABLE A.1 SIP CES-TO-CES CONNECTION CESA

No.	Source	Destination	Protocol	Info
1	10.1.0.120	10.1.0.1	DNS	Standard query A sip.cesa.ces
2	10.1.0.1	10.1.0.120	DNS	Standard query response A 10.1.1.10
3	10.1.0.120	10.1.1.10	SIP/SDP	Request: INVITE sip:101@sip.cesa.ces:50600, with SDP
4	10.1.1.11	10.1.0.10	SIP/SDP	Request: INVITE sip:101@sip.cesa.ces:50600, with SDP
5	10.1.0.10	10.1.1.11	SIP	Status: 407 Proxy Authentication Required
6	10.1.1.10	10.1.0.120	SIP	Status: 407 Proxy Authentication Required
7	10.1.0.120	10.1.1.10	SIP	Request: ACK sip:101@sip.cesa.ces:50600
8	10.1.0.120	10.1.1.10	SIP/SDP	Request: INVITE sip:101@sip.cesa.ces:50600, with SDP
9	10.1.1.11	10.1.0.10	SIP	Request: ACK sip:101@sip.cesa.ces:50600
10	10.1.1.11	10.1.0.10	SIP/SDP	Request: INVITE sip:101@sip.cesa.ces:50600, with SDP
11	10.1.0.10	10.1.1.11	SIP	Status: 100 Trying
12	10.1.1.10	10.1.0.120	SIP	Status: 100 Trying
13	10.1.0.10	10.1.0.1	DNS	Standard query A hardy2.cesa.ces
14	10.1.0.1	10.1.0.10	DNS	Standard query response A 10.1.1.11
15	10.1.0.10	10.1.0.1	DNS	Standard query A hardy.cesb.ces
16	10.1.0.1	10.1.0.10	DNS	Standard query response A 10.1.1.12
17	10.1.0.10	10.1.1.12	SIP/SDP	Request: INVITE sip:101@hardy.cesb.ces:50600, with SDP
18	10.1.1.12	10.1.0.10	SIP	Status: 100 Trying
19	10.1.1.12	10.1.0.10	SIP	Status: 180 Ringing
20	10.1.0.10	10.1.1.11	SIP	Status: 180 Ringing
21	10.1.1.10	10.1.0.120	SIP	Status: 180 Ringing
22	10.1.1.13	10.1.0.120	RTCP	Receiver Report Source description
23	10.1.0.120	10.1.1.13	ICMP	Destination unreachable (Port unreachable)
24	10.1.1.12	10.1.0.10	SIP/SDP	Status: 200 OK, with session description
25	10.1.0.10	10.1.0.1	DNS	Standard query A hardy.cesb.ces
26	10.1.1.13	10.1.0.120	RTP	PT=speex, SSRC=0x482E1A02, Seq=2625, Time=3319250273, Mark
27	10.1.0.120	10.1.1.13	ICMP	Destination unreachable (Port unreachable)
28	10.1.0.1	10.1.0.10	DNS	Standard query response A 10.1.1.12
29	10.1.1.13	10.1.0.120	RTP	PT=speex, SSRC=0x482E1A02, Seq=2626, Time=3319250433

30	10.1.0.120	10.1.1.13	ICMP	Destination unreachable (Port unreachable)
31	10.1.1.13	10.1.0.120	RTP	PT=speex, SSRC=0x482E1A02, Seq=2627, Time=3319250593
32	10.1.0.120	10.1.1.13	ICMP	Destination unreachable (Port unreachable)
33	10.1.1.13	10.1.0.120	RTP	PT=speex, SSRC=0x482E1A02, Seq=2628, Time=3319250753
34	10.1.0.120	10.1.1.13	ICMP	Destination unreachable (Port unreachable)
35	10.1.1.13	10.1.0.120	RTP	PT=speex, SSRC=0x482E1A02, Seq=2629, Time=3319250913
36	10.1.0.120	10.1.1.13	ICMP	Destination unreachable (Port unreachable)
37	10.1.0.10	10.1.1.11	SIP/SDP	Status: 200 OK, with session description
38	10.1.0.10	10.1.1.12	SIP	Request: ACK sip:101@hardy.cesb.ces:50600
39	10.1.1.10	10.1.0.120	SIP/SDP	Status: 200 OK, with session description
40	10.1.0.120	10.1.0.1	DNS	Standard query A hardy.cesb.ces
41	10.1.1.13	10.1.0.120	RTP	PT=speex, SSRC=0x482E1A02, Seq=2630, Time=3319251073
42	10.1.0.1	10.1.0.120	DNS	Standard query response A 10.1.1.13
43	10.1.0.120	10.1.1.13	RTCP	Receiver Report Source description
44	10.1.0.120	10.1.0.1	DNS	Standard query A sip.cesa.ces
45	10.1.0.1	10.1.0.120	DNS	Standard query response A 10.1.1.10
46	10.1.0.120	10.1.1.10	SIP	Request: ACK sip:101@sip.cesa.ces:50600
47	10.1.1.11	10.1.0.10	SIP	Request: ACK sip:101@sip.cesa.ces:50600
48	10.1.1.13	10.1.0.120	RTP	PT=speex, SSRC=0x482E1A02, Seq=2631, Time=3319251233
49	10.1.0.120	10.1.1.13	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20686, Time=1714857987, Mark
50	10.1.1.13	10.1.0.120	RTP	PT=speex, SSRC=0x482E1A02, Seq=2632, Time=3319251393
51	10.1.0.120	10.1.1.13	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20687, Time=1714858147
117	10.1.1.12	10.1.0.10	SIP	Request: BYE sip:100@sip.cesa.ces:50600
118	10.1.0.120	10.1.1.13	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20720, Time=1714863427
119	10.1.1.13	10.1.0.120	RTP	PT=speex, SSRC=0x482E1A02, Seq=2666, Time=3319256833
120	10.1.0.120	10.1.1.13	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20721, Time=1714863587
121	10.1.1.13	10.1.0.120	RTCP	Receiver Report Goodbye
122	10.1.0.120	10.1.1.13	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20722, Time=1714863747
123	10.1.0.120	10.1.1.13	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20723, Time=1714863907
124	10.1.0.120	10.1.1.13	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20724, Time=1714864067
125	10.1.0.120	10.1.1.13	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20725, Time=1714864227
126	10.1.0.10	10.1.0.1	DNS	Standard query A hardy2.cesa.ces
127	10.1.0.10	10.1.1.12	SIP	Status: 200 OK
128	10.1.0.1	10.1.0.10	DNS	Standard query response A 10.1.1.11
129	10.1.0.10	10.1.1.11	SIP	Request: BYE sip:100@hardy2.cesa.ces:50600
130	10.1.1.10	10.1.0.120	SIP	Request: BYE sip:100@hardy2.cesa.ces:50600
131	10.1.0.120	10.1.1.10	SIP	Status: 200 OK
132	10.1.0.120	10.1.1.13	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20726, Time=1714864387
133	10.1.1.11	10.1.0.10	SIP	Status: 200 OK
134	10.1.0.120	10.1.1.13	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20727, Time=1714864547
135	10.1.0.120	10.1.1.13	RTCP	Receiver Report Goodbye

TABLE A.2 SIP CES-TO-CES CONNECTION CESB

No.	Source	Destination	Protocol	Info
1	192.168.1.3	192.168.1.4	DNS	Standard query NAPTR hardy.cesb.ces
2	192.168.1.4	192.168.1.3	DNS	Standard query response NAPTR 100 10 U
3	56.48.48.57	52.52.52.52	SIP/SDP	Request: INVITE sip:101@hardy.cesb.ces:50600, with SDP
4	52.52.52.52	56.48.48.57	SIP	Status: 100 Trying
5	52.52.52.52	56.48.48.57	SIP	Status: 180 Ringing
6	192.168.1.4	192.168.1.3	DNS	Standard query NAPTR hardy2.cesa.ces
7	192.168.1.3	192.168.1.4	DNS	Standard query response NAPTR 100 10 U
8	52.52.52.52	53.53.53.53	UDP	Source port: vcom-tunnel Destination port: vcom-tunnel
9	52.52.52.52	56.48.48.57	SIP/SDP	Status: 200 OK, with session description
10	52.52.52.52	53.53.53.53	UDP	Source port: 8000 Destination port: 8000
11	53.53.53.53	52.52.52.52	ICMP	Destination unreachable (Port unreachable)
12	192.168.1.3	192.168.1.4	DNS	Standard query NAPTR hardy.cesb.ces
13	192.168.1.4	192.168.1.3	DNS	Standard query response NAPTR 100 10 U
14	52.52.52.52	53.53.53.53	UDP	Source port: 8000 Destination port: 8000
15	53.53.53.53	52.52.52.52	ICMP	Destination unreachable (Port unreachable)
16	53.53.53.53	52.52.52.52	ICMP	Destination unreachable (Port unreachable)
17	52.52.52.52	53.53.53.53	UDP	Source port: 8000 Destination port: 8000
18	53.53.53.53	52.52.52.52	ICMP	Destination unreachable (Port unreachable)
19	52.52.52.52	53.53.53.53	UDP	Source port: 8000 Destination port: 8000
20	53.53.53.53	52.52.52.52	ICMP	Destination unreachable (Port unreachable)
21	52.52.52.52	53.53.53.53	UDP	Source port: 8000 Destination port: 8000
22	53.53.53.53	52.52.52.52	ICMP	Destination unreachable (Port unreachable)
23	52.52.52.52	53.53.53.53	UDP	Source port: 8000 Destination port: 8000
24	56.48.48.57	52.52.52.52	SIP	Request: ACK sip:101@hardy.cesb.ces:50600
25	192.168.1.3	192.168.1.4	DNS	Standard query NAPTR hardy.cesb.ces
26	192.168.1.4	192.168.1.3	DNS	Standard query response NAPTR 100 10 U
27	53.53.53.53	52.52.52.52	UDP	Source port: vcom-tunnel Destination port: vcom-tunnel
28	52.52.52.52	53.53.53.53	UDP	Source port: 8000 Destination port: 8000
29	53.53.53.53	52.52.52.52	UDP	Source port: 8000 Destination port: 8000
30	52.52.52.52	53.53.53.53	UDP	Source port: 8000 Destination port: 8000
31	53.53.53.53	52.52.52.52	UDP	Source port: 8000 Destination port: 8000
32	52.52.52.52	53.53.53.53	UDP	Source port: 8000 Destination port: 8000
33	53.53.53.53	52.52.52.52	UDP	Source port: 8000 Destination port: 8000
34	52.52.52.52	53.53.53.53	UDP	Source port: 8000 Destination port: 8000
93	53.53.53.53	52.52.52.52	UDP	Source port: 8000 Destination port: 8000
94	52.52.52.52	53.53.53.53	UDP	Source port: 8000 Destination port: 8000
95	192.168.1.4	192.168.1.3	DNS	Standard query NAPTR sip.cesa.ces
96	192.168.1.3	192.168.1.4	DNS	Standard query response NAPTR 100 10 U
97	53.53.53.53	52.52.52.52	UDP	Source port: 8000 Destination port: 8000
98	52.52.52.52	53.53.53.53	UDP	Source port: 8000 Destination port: 8000
99	52.52.52.52	56.48.48.57	SIP	Request: BYE sip:100@sip.cesa.ces:50600
100	53.53.53.53	52.52.52.52	UDP	Source port: 8000 Destination port: 8000
101	52.52.52.52	53.53.53.53	UDP	Source port: 8000 Destination port: 8000

102	52.52.52.52	53.53.53.53	UDP	Source port: vcom-tunnel Destination port: vcom-tunnel
103	53.53.53.53	52.52.52.52	UDP	Source port: 8000 Destination port: 8000
104	52.52.52.52	53.53.53.53	ICMP	Destination unreachable (Port unreachable)
105	53.53.53.53	52.52.52.52	UDP	Source port: 8000 Destination port: 8000
106	52.52.52.52	53.53.53.53	ICMP	Destination unreachable (Port unreachable)
107	53.53.53.53	52.52.52.52	UDP	Source port: 8000 Destination port: 8000
108	53.53.53.53	52.52.52.52	UDP	Source port: 8000 Destination port: 8000
109	52.52.52.52	53.53.53.53	ICMP	Destination unreachable (Port unreachable)
110	52.52.52.52	53.53.53.53	ICMP	Destination unreachable (Port unreachable)
111	53.53.53.53	52.52.52.52	UDP	Source port: 8000 Destination port: 8000
112	56.48.48.57	52.52.52.52	SIP	Status: 200 OK
113	52.52.52.52	53.53.53.53	ICMP	Destination unreachable (Port unreachable)
114	53.53.53.53	52.52.52.52	UDP	Source port: 8000 Destination port: 8000
115	52.52.52.52	53.53.53.53	ICMP	Destination unreachable (Port unreachable)
116	53.53.53.53	52.52.52.52	UDP	Source port: 8000 Destination port: 8000
117	53.53.53.53	52.52.52.52	UDP	Source port: vcom-tunnel Destination port: vcom-tunnel

TABLE A.3 SIP CES-TO-CES CONNECTION HOSTB

No.	Source	Destination	Protocol	Info
1	10.1.3.10	10.1.2.110	SIP/SDP	Request: INVITE sip:101@hardy.cesb.ces:50600, with SDP
2	10.1.2.110	10.1.3.10	SIP	Status: 100 Trying
3	10.1.2.110	10.1.3.10	SIP	Status: 180 Ringing
4	10.1.2.110	10.1.2.1	DNS	Standard query A hardy2.cesa.ces
5	10.1.2.1	10.1.2.110	DNS	Standard query response A 10.1.3.11
6	10.1.2.110	10.1.3.11	RTCP	Receiver Report Source description
7	10.1.2.110	10.1.3.10	SIP/SDP	Status: 200 OK, with session description
8	10.1.2.110	10.1.3.11	RTP	PT=speex, SSRC=0x482E1A02, Seq=2625, Time=3319250273, Mark
9	10.1.2.110	10.1.3.11	RTP	PT=speex, SSRC=0x482E1A02, Seq=2626, Time=3319250433
10	10.1.2.110	10.1.3.11	RTP	PT=speex, SSRC=0x482E1A02, Seq=2627, Time=3319250593
11	10.1.2.110	10.1.3.11	RTP	PT=speex, SSRC=0x482E1A02, Seq=2628, Time=3319250753
12	10.1.2.110	10.1.3.11	RTP	PT=speex, SSRC=0x482E1A02, Seq=2629, Time=3319250913
13	10.1.2.110	10.1.3.11	RTP	PT=speex, SSRC=0x482E1A02, Seq=2630, Time=3319251073
14	10.1.3.10	10.1.2.110	SIP	Request: ACK sip:101@hardy.cesb.ces:50600
15	10.1.2.110	10.1.3.11	RTP	PT=speex, SSRC=0x482E1A02, Seq=2631, Time=3319251233
16	10.1.3.11	10.1.2.110	RTCP	Receiver Report Source description
17	10.1.2.110	10.1.3.11	RTP	PT=speex, SSRC=0x482E1A02, Seq=2632, Time=3319251393
18	10.1.3.11	10.1.2.110	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20686, Time=1714857987, Mark
19	10.1.2.110	10.1.3.11	RTP	PT=speex, SSRC=0x482E1A02, Seq=2633, Time=3319251553
20	10.1.3.11	10.1.2.110	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20687, Time=1714858147
21	10.1.2.110	10.1.3.11	RTP	PT=speex, SSRC=0x482E1A02, Seq=2634, Time=3319251713
22	10.1.3.11	10.1.2.110	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20688, Time=1714858307
23	10.1.2.110	10.1.3.11	RTP	PT=speex, SSRC=0x482E1A02, Seq=2635, Time=3319251873
81	10.1.3.11	10.1.2.110	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20718, Time=1714863107

82	10.1.2.110	10.1.3.11	RTP	PT=speex, SSRC=0x482E1A02, Seq=2664, Time=3319256513
83	10.1.2.110	10.1.2.1	DNS	Standard query A sip.cesa.ces
84	10.1.2.110	10.1.3.11	RTP	PT=speex, SSRC=0x482E1A02, Seq=2665, Time=3319256673
85	10.1.2.1	10.1.2.110	DNS	Standard query response A 10.1.3.10
86	10.1.2.110	10.1.3.10	SIP	Request: BYE sip:100@sip.cesa.ces:50600
87	10.1.3.11	10.1.2.110	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20719, Time=1714863267
88	10.1.2.110	10.1.3.11	RTP	PT=speex, SSRC=0x482E1A02, Seq=2666, Time=3319256833
89	10.1.3.11	10.1.2.110	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20720, Time=1714863427
90	10.1.2.110	10.1.3.11	RTCP	Receiver Report Goodbye [Malformed Packet]
91	10.1.3.11	10.1.2.110	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20721, Time=1714863587
92	10.1.2.110	10.1.3.11	ICMP	Destination unreachable (Port unreachable)
93	10.1.3.11	10.1.2.110	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20722, Time=1714863747
94	10.1.2.110	10.1.3.11	ICMP	Destination unreachable (Port unreachable)
95	10.1.3.11	10.1.2.110	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20723, Time=1714863907
96	10.1.2.110	10.1.3.11	ICMP	Destination unreachable (Port unreachable)
97	10.1.3.11	10.1.2.110	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20724, Time=1714864067
98	10.1.2.110	10.1.3.11	ICMP	Destination unreachable (Port unreachable)
99	10.1.3.11	10.1.2.110	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20725, Time=1714864227
100	10.1.2.110	10.1.3.11	ICMP	Destination unreachable (Port unreachable)
101	10.1.3.10	10.1.2.110	SIP	Status: 200 OK
102	10.1.3.11	10.1.2.110	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20726, Time=1714864387
103	10.1.2.110	10.1.3.11	ICMP	Destination unreachable (Port unreachable)
104	10.1.3.11	10.1.2.110	RTP	PT=speex, SSRC=0xEA3927B8, Seq=20727, Time=1714864547
105	10.1.3.11	10.1.2.110	RTCP	Receiver Report Goodbye [Malformed Packet]

Appendix B

FTP local connection

ARP messages are removed from the following tables. DNS queries are shown in lighter shade. Dark gray represents the data communication.

TABLE B.1 LOCAL FTP CONNECTION HOSTA

No.	Source	Destination	Protocol	Info
1	10.1.0.120	10.1.0.1	DNS	Standard query A hardy4.cesa.ces
2	10.1.0.1	10.1.0.120	DNS	Standard query response A 10.1.1.10
3	10.1.0.120	10.1.1.10	TCP	37521 > ftp [SYN] Seq=0 ...
7	10.1.1.10	10.1.0.120	TCP	ftp > 37521 [SYN, ACK] Seq=0 Ack=1 ...
8	10.1.0.120	10.1.1.10	TCP	37521 > ftp [ACK] Seq=1 Ack=1 ...
9	10.1.1.10	10.1.0.120	FTP	Response: 220 (vsFTPd 2.0.6)
10	10.1.0.120	10.1.1.10	TCP	37521 > ftp [ACK] Seq=1 Ack=21 ...
11	10.1.0.120	10.1.1.10	FTP	Request: USER tester
12	10.1.1.10	10.1.0.120	TCP	ftp > 37521 [ACK] Seq=21 Ack=14 ...
13	10.1.1.10	10.1.0.120	FTP	Response: 331 Please specify the password.
14	10.1.0.120	10.1.1.10	TCP	37521 > ftp [ACK] Seq=14 Ack=55 ...
15	10.1.0.120	10.1.1.10	FTP	Request: PASS cestest
16	10.1.1.10	10.1.0.120	FTP	Response: 230 Login successful.
17	10.1.0.120	10.1.1.10	TCP	37521 > ftp [ACK] Seq=28 Ack=78 ...
18	10.1.0.120	10.1.1.10	FTP	Request: SYST
19	10.1.1.10	10.1.0.120	FTP	Response: 215 UNIX Type: L8
20	10.1.0.120	10.1.1.10	TCP	37521 > ftp [ACK] Seq=34 Ack=97 ...
21	10.1.0.120	10.1.1.10	FTP	Request: PORT 10,1,0,120,231,158
22	10.1.1.10	10.1.0.120	FTP	Response: 200 PORT command successful. Consider using PASV.
23	10.1.0.120	10.1.1.10	TCP	37521 > ftp [ACK] Seq=59 Ack=148 ...
24	10.1.0.120	10.1.1.10	FTP	Request: LIST
25	10.1.1.10	10.1.0.120	TCP	ftp-data > 59294 [SYN] Seq=0 ...
26	10.1.0.120	10.1.1.10	TCP	59294 > ftp-data [SYN, ACK] Seq=0 Ack=1 ...
27	10.1.1.10	10.1.0.120	TCP	ftp-data > 59294 [ACK] Seq=1 Ack=1 ...
28	10.1.1.10	10.1.0.120	FTP	Response: 150 Here comes the directory listing.
29	10.1.1.10	10.1.0.120	FTPDATA	FTP Data: 1103 bytes
30	10.1.0.120	10.1.1.10	TCP	59294 > ftp-data [ACK] Seq=1 Ack=1104 ...
31	10.1.1.10	10.1.0.120	TCP	ftp-data > 59294 [FIN, ACK] Seq=1104 Ack=1 ...
32	10.1.0.120	10.1.1.10	TCP	59294 > ftp-data [FIN, ACK] Seq=1 Ack=1105 ...
33	10.1.1.10	10.1.0.120	TCP	ftp-data > 59294 [ACK] Seq=1105 Ack=2 ...
34	10.1.1.10	10.1.0.120	FTP	Response: 226 Directory send OK.
35	10.1.0.120	10.1.1.10	TCP	37521 > ftp [ACK] Seq=65 Ack=211 ...
36	10.1.0.120	10.1.1.10	FTP	Request: PASV
37	10.1.1.10	10.1.0.120	FTP	Response: 227 Entering Passive Mode (10,1,1,10,185,112).
38	10.1.0.120	10.1.1.10	TCP	57411 > 47472 [SYN] Seq=3342737855 ...
39	10.1.1.10	10.1.0.120	TCP	47472 > 57411 [SYN, ACK] Seq=3325353149 Ack=3342737856 ...
40	10.1.0.120	10.1.1.10	TCP	57411 > 47472 [ACK] Seq=3342737856 Ack=3325353150 ...
41	10.1.0.120	10.1.1.10	FTP	Request: LIST
42	10.1.1.10	10.1.0.120	FTP	Response: 150 Here comes the directory listing.
43	10.1.1.10	10.1.0.120	FTPDATA	FTP Data: 1103 bytes
44	10.1.0.120	10.1.1.10	TCP	57411 > 47472 [ACK] Seq=3342737856 Ack=3325354253 ...
45	10.1.1.10	10.1.0.120	TCP	47472 > 57411 [FIN, ACK] Seq=3325354253 Ack=3342737856 ...

46	10.1.0.120	10.1.1.10	TCP	57411 > 47472 [FIN, ACK] Seq=3342737856 Ack=3325354254 ...
47	10.1.1.10	10.1.0.120	TCP	47472 > 57411 [ACK] Seq=3325354254 Ack=3342737857 ...
48	10.1.1.10	10.1.0.120	FTP	Response: 226 Directory send OK.
49	10.1.0.120	10.1.1.10	TCP	37521 > ftp [ACK] Seq=77 Ack=322 ...
50	10.1.0.120	10.1.1.10	FTP	Request: PORT 10,1,0,120,152,118
51	10.1.1.10	10.1.0.120	FTP	Response: 200 PORT command successful. Consider using PASV.
52	10.1.0.120	10.1.1.10	FTP	Request: LIST
53	10.1.1.10	10.1.0.120	TCP	ftp-data > 39030 [SYN] Seq=0 ...
54	10.1.0.120	10.1.1.10	TCP	39030 > ftp-data [SYN, ACK] Seq=0 Ack=1 ...
55	10.1.1.10	10.1.0.120	TCP	ftp-data > 39030 [ACK] Seq=1 Ack=1 ...
56	10.1.1.10	10.1.0.120	FTP	Response: 150 Here comes the directory listing.
57	10.1.1.10	10.1.0.120	FTPDATA	FTP Data: 1103 bytes
58	10.1.0.120	10.1.1.10	TCP	39030 > ftp-data [ACK] Seq=1 Ack=1104 ...
59	10.1.1.10	10.1.0.120	TCP	ftp-data > 39030 [FIN, ACK] Seq=1104 Ack=1 ...
60	10.1.0.120	10.1.1.10	TCP	39030 > ftp-data [FIN, ACK] Seq=1 Ack=1105 ...
61	10.1.1.10	10.1.0.120	TCP	ftp-data > 39030 [ACK] Seq=1105 Ack=2 ...
62	10.1.1.10	10.1.0.120	FTP	Response: 226 Directory send OK.
63	10.1.0.120	10.1.1.10	TCP	37521 > ftp [ACK] Seq=108 Ack=436 ...
64	10.1.0.120	10.1.1.10	FTP	Request: QUIT
65	10.1.1.10	10.1.0.120	FTP	Response: 221 Goodbye.
66	10.1.0.120	10.1.1.10	TCP	37521 > ftp [FIN, ACK] Seq=114 Ack=450 ...
67	10.1.1.10	10.1.0.120	TCP	ftp > 37521 [FIN, ACK] Seq=450 Ack=114 ...
68	10.1.0.120	10.1.1.10	TCP	37521 > ftp [ACK] Seq=115 Ack=451 ...

TABLE B.2 LOCAL FTP CONNECTION HOSTC

No.	Source	Destination	Protocol	Info
3	10.1.1.11	10.1.0.130	TCP	37521 > ftp [SYN] Seq=0 ...
4	10.1.0.130	10.1.1.11	TCP	ftp > 37521 [SYN, ACK] Seq=0 Ack=1 ...
6	10.1.1.11	10.1.0.130	TCP	37521 > ftp [ACK] Seq=1 Ack=1 ...
7	10.1.0.130	10.1.1.11	FTP	Response: 220 (vsFTPd 2.0.6)
8	10.1.1.11	10.1.0.130	TCP	37521 > ftp [ACK] Seq=1 Ack=21 ...
9	10.1.1.11	10.1.0.130	FTP	Request: USER tester
10	10.1.0.130	10.1.1.11	TCP	ftp > 37521 [ACK] Seq=21 Ack=14 ...
11	10.1.0.130	10.1.1.11	FTP	Response: 331 Please specify the password.
12	10.1.1.11	10.1.0.130	TCP	37521 > ftp [ACK] Seq=14 Ack=55 ...
13	10.1.1.11	10.1.0.130	FTP	Request: PASS cestest
14	10.1.0.130	10.1.1.11	FTP	Response: 230 Login successful.
15	10.1.1.11	10.1.0.130	TCP	37521 > ftp [ACK] Seq=28 Ack=78 ...
16	10.1.1.11	10.1.0.130	FTP	Request: SYST
17	10.1.0.130	10.1.1.11	FTP	Response: 215 UNIX Type: L8
18	10.1.1.11	10.1.0.130	TCP	37521 > ftp [ACK] Seq=34 Ack=97 ...
19	10.1.1.11	10.1.0.130	FTP	Request: PORT 10,1,1,11,231,158
20	10.1.0.130	10.1.1.11	FTP	Response: 200 PORT command successful. Consider using PASV.
21	10.1.1.11	10.1.0.130	TCP	37521 > ftp [ACK] Seq=58 Ack=148 ...
22	10.1.1.11	10.1.0.130	FTP	Request: LIST
23	10.1.0.130	10.1.1.11	TCP	ftp-data > 59294 [SYN] Seq=0 ...
24	10.1.1.11	10.1.0.130	TCP	59294 > ftp-data [SYN, ACK] Seq=0 Ack=1 ...
25	10.1.0.130	10.1.1.11	TCP	ftp-data > 59294 [ACK] Seq=1 Ack=1 ...
26	10.1.0.130	10.1.1.11	FTP	Response: 150 Here comes the directory listing.
27	10.1.0.130	10.1.1.11	FTPDATA	FTP Data: 1103 bytes
28	10.1.0.130	10.1.1.11	TCP	ftp-data > 59294 [FIN, ACK] Seq=1104 Ack=1 ...
29	10.1.1.11	10.1.0.130	TCP	59294 > ftp-data [ACK] Seq=1 Ack=1104 ...

30	10.1.1.11	10.1.0.130	TCP	59294 > ftp-data [FIN, ACK] Seq=1 Ack=1105 ...
31	10.1.0.130	10.1.1.11	TCP	ftp-data > 59294 [ACK] Seq=1105 Ack=2 ...
32	10.1.0.130	10.1.1.11	FTP	Response: 226 Directory send OK.
33	10.1.1.11	10.1.0.130	TCP	37521 > ftp [ACK] Seq=64 Ack=211 ...
34	10.1.1.11	10.1.0.130	FTP	Request: PASV
35	10.1.0.130	10.1.1.11	FTP	Response: 227 Entering Passive Mode (10,1,0,130,185,112)
36	10.1.1.11	10.1.0.130	TCP	57411 > 47472 [SYN] Seq=3342737855 ...
37	10.1.0.130	10.1.1.11	TCP	47472 > 57411 [SYN, ACK] Seq=3325353149 Ack=3342737856 ...
38	10.1.1.11	10.1.0.130	TCP	57411 > 47472 [ACK] Seq=3342737856 Ack=3325353150 ...
39	10.1.1.11	10.1.0.130	FTP	Request: LIST
40	10.1.0.130	10.1.1.11	FTP	Response: 150 Here comes the directory listing.
41	10.1.0.130	10.1.1.11	FTPDATA	FTP Data: 1103 bytes
42	10.1.0.130	10.1.1.11	TCP	47472 > 57411 [FIN, ACK] Seq=3325354253 Ack=3342737856 ...
43	10.1.1.11	10.1.0.130	TCP	57411 > 47472 [ACK] Seq=3342737856 Ack=3325354253 ...
44	10.1.1.11	10.1.0.130	TCP	57411 > 47472 [FIN, ACK] Seq=3342737856 Ack=3325354254 ...
45	10.1.0.130	10.1.1.11	TCP	47472 > 57411 [ACK] Seq=3325354254 Ack=3342737857 ...
46	10.1.0.130	10.1.1.11	FTP	Response: 226 Directory send OK.
47	10.1.1.11	10.1.0.130	TCP	37521 > ftp [ACK] Seq=76 Ack=322 ...
48	10.1.1.11	10.1.0.130	FTP	Request: PORT 10,1,1,11,152,118
49	10.1.0.130	10.1.1.11	FTP	Response: 200 PORT command successful. Consider using PASV.
50	10.1.1.11	10.1.0.130	FTP	Request: LIST
51	10.1.0.130	10.1.1.11	TCP	ftp-data > 39030 [SYN] Seq=0 ...
52	10.1.1.11	10.1.0.130	TCP	39030 > ftp-data [SYN, ACK] Seq=0 Ack=1 ...
53	10.1.0.130	10.1.1.11	TCP	ftp-data > 39030 [ACK] Seq=1 Ack=1 ...
54	10.1.0.130	10.1.1.11	FTP	Response: 150 Here comes the directory listing.
55	10.1.0.130	10.1.1.11	FTPDATA	FTP Data: 1103 bytes
56	10.1.0.130	10.1.1.11	TCP	ftp-data > 39030 [FIN, ACK] Seq=1104 Ack=1 ...
57	10.1.1.11	10.1.0.130	TCP	39030 > ftp-data [ACK] Seq=1 Ack=1104 ...
58	10.1.1.11	10.1.0.130	TCP	39030 > ftp-data [FIN, ACK] Seq=1 Ack=1105 ...
59	10.1.0.130	10.1.1.11	TCP	ftp-data > 39030 [ACK] Seq=1105 Ack=2 ...
60	10.1.0.130	10.1.1.11	FTP	Response: 226 Directory send OK.
61	10.1.1.11	10.1.0.130	TCP	37521 > ftp [ACK] Seq=106 Ack=436 ...
62	10.1.1.11	10.1.0.130	FTP	Request: QUIT
63	10.1.0.130	10.1.1.11	FTP	Response: 221 Goodbye.
64	10.1.0.130	10.1.1.11	TCP	ftp > 37521 [FIN, ACK] Seq=450 Ack=112 ...
65	10.1.1.11	10.1.0.130	TCP	37521 > ftp [FIN, ACK] Seq=112 Ack=450 ...
66	10.1.0.130	10.1.1.11	TCP	ftp > 37521 [ACK] Seq=451 Ack=113 ...
67	10.1.1.11	10.1.0.130	TCP	37521 > ftp [ACK] Seq=113 Ack=451 ...

Appendix C

First this appendix has information about Twinkle configurations.

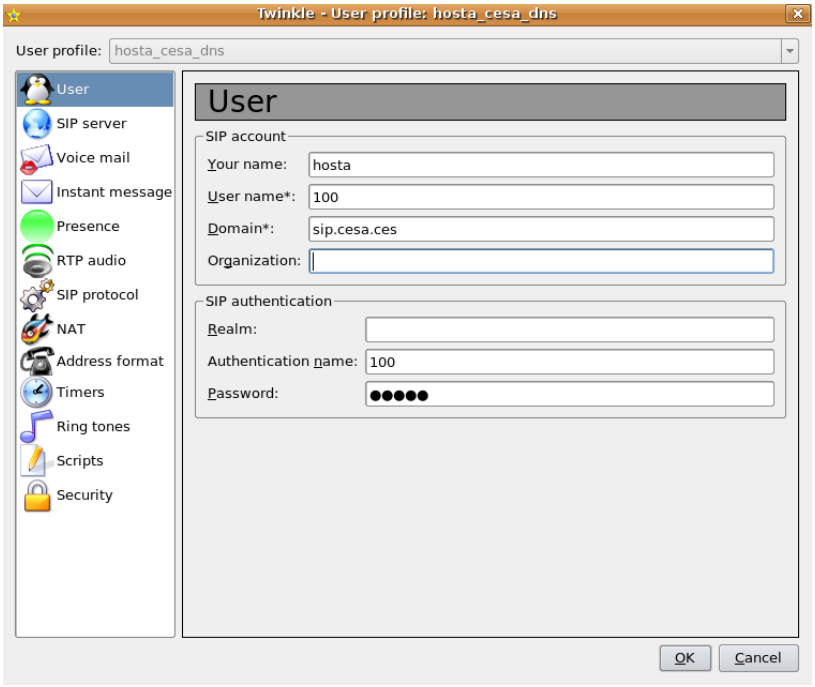

Figure C.1: Twinkle - User profile settings

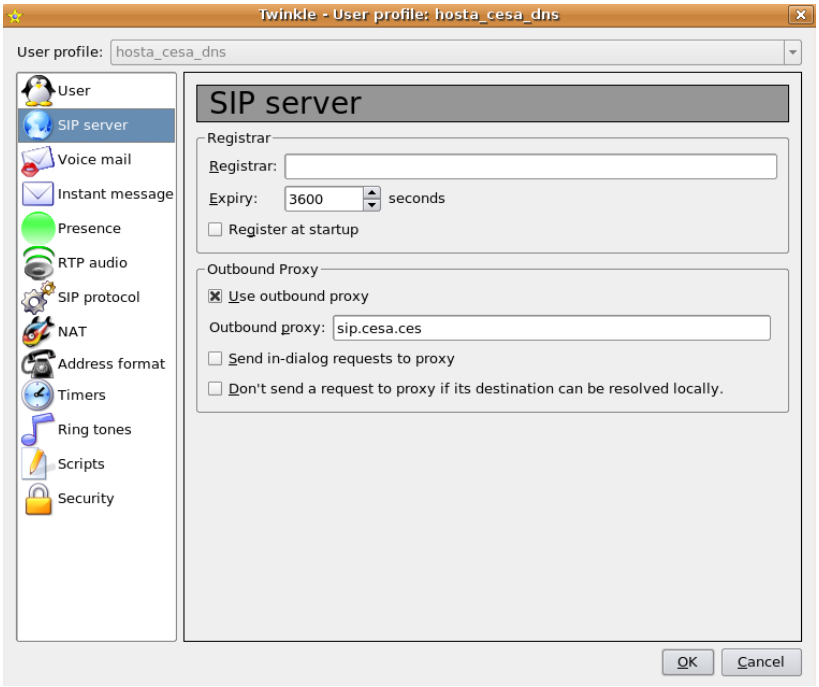

Figure C.2: Twinkle - SIP server settings

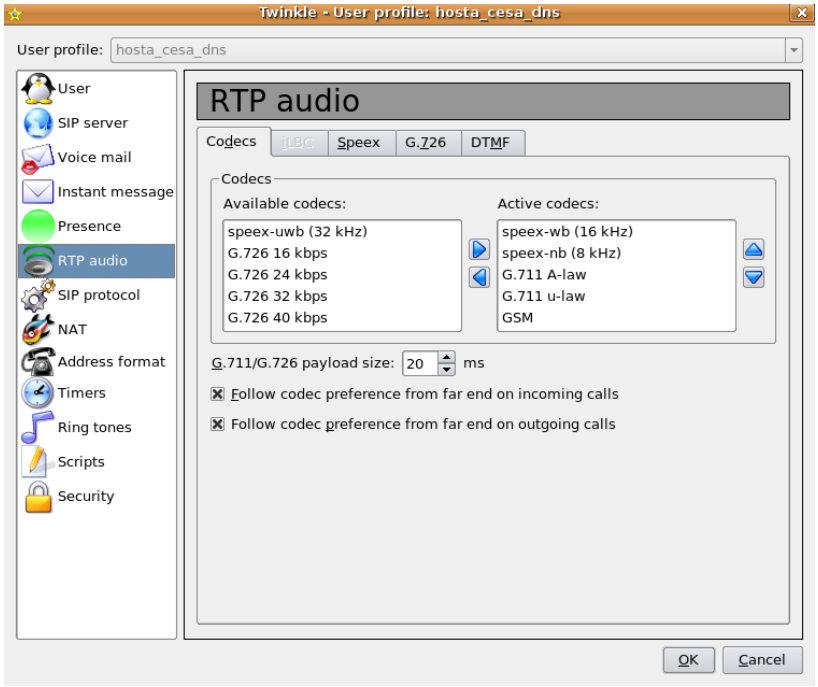

Figure C.3: Twinkle - RTP audio settings

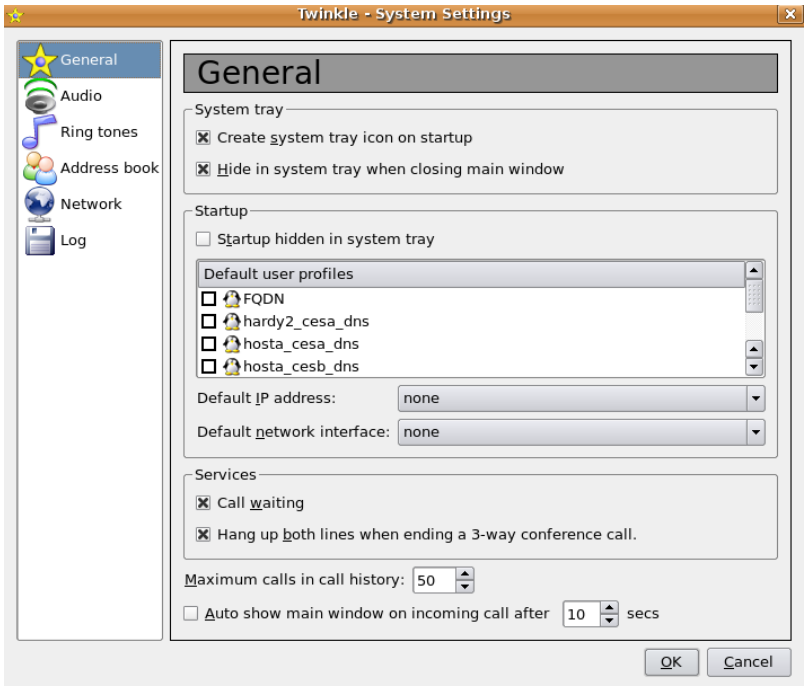

Figure C.4: Twinkle - General settings

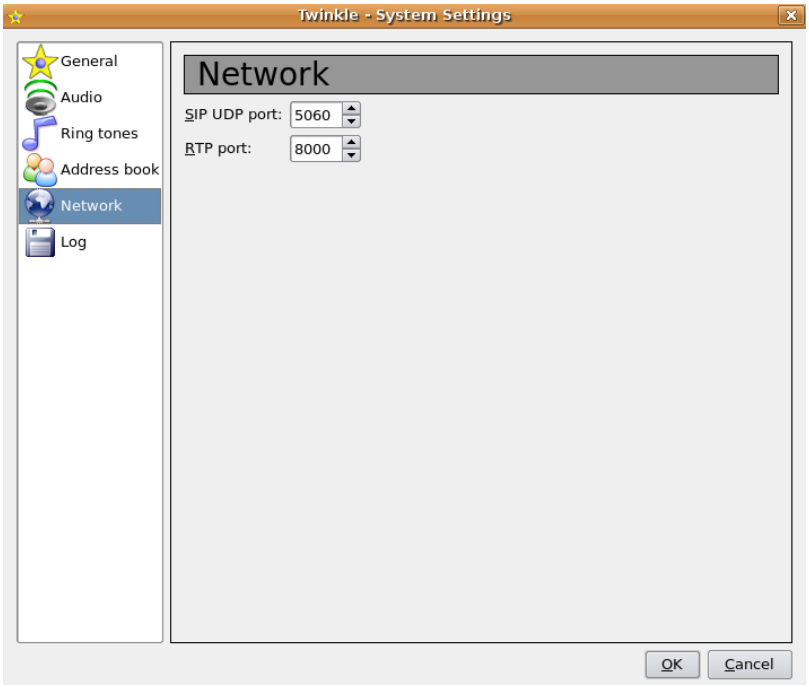

Figure C.5: Twinkle - Network settings

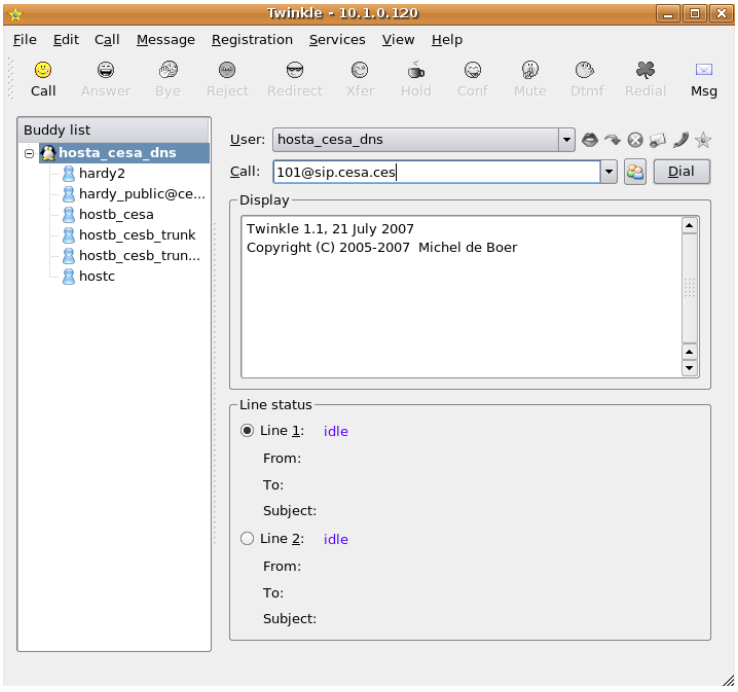

Figure C6: Twinkle - Main window

Finally, here is the extension configuration in the 3CX SIP server.

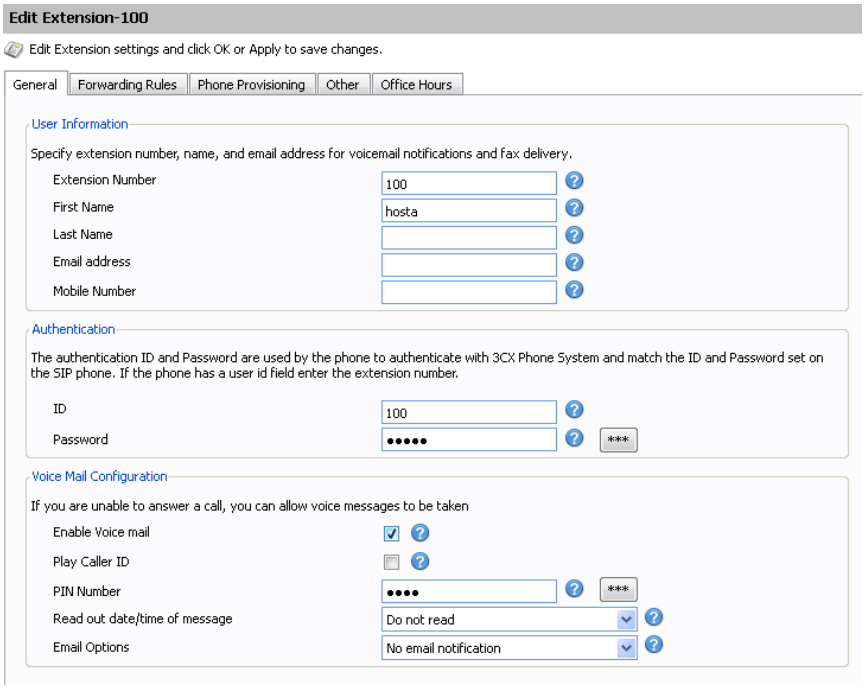

Figure C.7: 3CX – Extension settings