UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science
Masrter of Software Engineering

L.D.Mohan D.Liyanage

Addressing Devices in Mobile Networks

Master Thesis (30 EAP)

Supervisor:Satish Narayana Srirama, PhD

TARTU, 2014

Addressing Devices in Mobile Networks

Abstract:

The emergence of mobile terminals with enhanced features like high processing power, more memory, inbuilt sensors, low power consumption, etc. have led to their extensive usage in different domains like mobile social networking, mobile cloud and Internet of Things (IoT). However, to successfully utilize these devices as information providing/processing entities, we need proper means of identification and addressing, so that the devices and their offered data/services are accessible also from outside the mobile network. But most of the times, when the peers connecting to the internet through cellular networks, peer devices locate behind the common components like firewalls and Network Address Translators (NATs) that prevent establishing direct connections. Setting up connection between peers in mobile networks has been examined extensively over the years and there are several solutions one can conceive. However, the most popular and widely used addressing mechanism for internet, IP address, is also being extensively used in mobile data networks (3G/4G) but ends up with barriers like their temporarily availability, known only within the mobile operators network, Network Address Translation (NAT) etc. To address such kind of limitations we proposed few different approaches such as Session Initiation Protocol (SIP), UDP/TCP hole punching with help from the Rendezvous server and UDP/TCP Relaying those can be applied to different types of mobile networks. In this thesis we discuss practical implementation, test results and evaluation of strengths and limitations of each approach.

Key words: Android, Mobile Networks, Addressing, Hole Punching

Seadmete adresseerimine mobiilivrkudes

Lhikokkuvte:

Mobiilterminalide arengust tingitud vhenenud energiakulu, sisseehitatud sensorite kasutusvimalus, suurenenud ttlusjudlus ja mlumaht vimaldavad mobiilide laialdase kasutuse erinevates domeenides nagu mobiilne sotsiaalvrgustik, mobiilne pilvandmettlus ja Internet of Things (IoT). Selleks, et antud seadmeid oleks vimalik edukalt informatsiooni pakkumise ja ttlemise vahenditena kasutada, on vaja identifitseerimiseks ja adresseerimiseks lesandele kohaseid vahendeid, mis vimaldaksid ligipsu seadmetele ja teenustele ka vljaspool mobiilsidevrku. Enamuse ajast, kui kasutajad kasutavad Internetiga hendamiseks mobiilivrke, paiknevad kasutajate seadmed tulemride ja vrguaadressi translaatorite (NAT ehk Network Address Translator) taga, mis takistavad otsese henduse loomist.

Kasutajate hendamist mobiilsetes vrkudes on aastaid phjalikult uuritud ja selle tulemusena on leitud mitmeid lahendusi. IP-aadress, mis on levinuim adresseerimise mehhanism Internetis, on htlasi laialdaselt kasutusel mobiilivrkudes (3G/4G), kuid sellel on omad piirangud: ajutine kttesaadavus, piiratud kasutus ainult mobiilioperaatorite vrkudes ja vrguaadresside tlkimine (NAT). Nende piirangute krvaldamiseks pakume vlja mned teistsugused lhenemised: Session Initiation Protocol (SIP), Rendezvous serveri toel toimiv UDP/TCP Hole Punching ja UDP/TCP Relaying. Neid saab kasutada erinevate mobiilsidevrkude tpide puhul. Kesolevas magistrits ksitletakse praktilist paigaldust, testide tulemusi ja iga lhenemise nrku ning tugevaid klgi.

Android, mobiilsed vrgud, adresseerimine, Hole Punching

Contents

List of Figures

1

Introduction

In recent years, smart phones have entered the market with enhanced features like high processing power, more memory, HD video, inbuilt sensors, touchscreen, low power consumption, etc. changed the ways in which people want to use it. It is also worth to mention that during past years usage of mobile phones has grown rapidly, according to the data from some market surveys. According to the latest report released by the ITU, the Mobile-cellular subscriptions will reach almost 7 billion by end 2014 that approaching the number of people on the earth. There are more than 6.8 billion mobile subscribers currently in the world, including 2.3 billion of mobile broadband subscriptions (1).

With those enhanced features and the development of the mobile infrastructure, there are many new applications and services developed for mobile users and the big demand for data intensive applications for mobile devices can be seen. More common examples of such kind of applications are Peer-to-Peer applications like mobile social networking, playing online games, sharing photos and videos with friends, teacher sharing blackboard with a group of students etc.

Apart from the traditional client/server architecture, recent studies showed that even mobile devices can be used to provide services to other mobile users. As an example, use of Mobile Host, the mobile device which can provide the Mobile web services for other clients (2), (3)over mobile network was demonstrated that the proof of Mobile Web Service Provisioning.

While the applications are interesting, for successful adoption of these mobile Peer-to-Peer and social applications, the mobile terminal, that is registered and connected

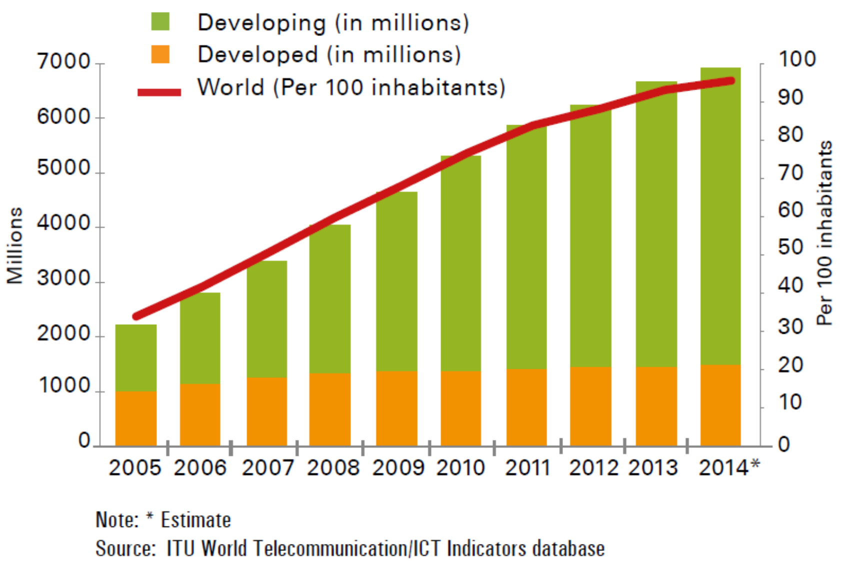


Figure 1.1: Worldwide Mobile-cellular subscriptions (1)

to the mobile operator network, requires some means of identification and addressing. When the device using such kind of applications, it is important that peers should be able to establish connections without going through the intermediate devices. But most of the times, when the peers connecting to the internet through cellular networks, mobile devices locate behind the common components like firewalls and Network Address Translators (NATs) which prevent establishing direct connections.

Addressing devices in mobile networks have been studied extensively over the years and there are several solutions one can consider. Technologies such as OpenID (4), Session Initiation Protocol (SIP) (5), Extensible Messaging and Presence Protocol (XMPP) (6), Zero-configuration networking (zeroconf) (7),and Universal Plug and Play (UPnP) (8) have offered solutions for addressing devices in different Peer-to-Peer setups. Although there are different addressing mechanisms proposed, the most popular and widely used addressing mechanism for internet is Internet Protocol Address (IP address). IP addressing provides a globally unique 32 bit address for each and every device connected to the internetwork. Network operators always have provided different means to address the devices with IP addresses, so that they could be accessed from the Internet (9).

We have extensively studied several solutions to addressing device, but most of the solutions designed for the device connected to the fixed networks or Wi-Fi networks.

Those solutions not applicable when addressing device in the mobile network and we could not find any open source solution that can apply to the mobile domain. There were some proprietary applications provide limited functionality when considering in the mobile domain, but not giving any workable generic solution.

Due to different configuration policies of mobile networks, the addressing approach also may be changed. This thesis provides few techniques that can use to address devices in mobile networks. Also, it provides an evaluation on each approach with relative to the applied configurations by the mobile operator.

1.1 Problem Statement

Addressing devices in networks have been studied extensively over the years and there are some addressing mechanisms available for fixed networks, but when considering mobile domain, developers still facing difficulties to address devices. However, to successfully utilize mobile devices as information providing/processing entities, we need proper means of identification and addressing, so that the devices and their offered data/services are accessible also from outside the mobile network. There have been several solutions over the years like Session Initiation Protocol (SIP), provide a Public IPs for mobile devices by operators, a Mobile IP that allows users to move from one network to another network without changing the permanent IP address, etc. but still having some limitations. When considering the latest technologies like 3G/4G mobile networks that provide services for millions of devices getting more complicated in addressing devices. However, the most popular and widely used addressing mechanism for internet, IP address, is also being extensively used in mobile data networks (3G/4G). Typically, the IP address assigned to mobile devices ends up with barriers like their temporarily availability, known only within the mobile operator's network, Network Address Translation (NAT) etc., which makes it difficult to use the addresses in IoT scenarios.

During the NAT process the NAT box maintains a map in order to determine which computer should receive the responses from outside clients. But because of the NAT process, devices in private network not visible to outside and it is impossible to initiate a connection from the outside device to the internal device. Whenever inbound traffic originated from the outside devices only allowed if those are being a part of an existing session that initiated from the internal devices. This process adds the additional security for the inside (private) network because internal devices are not visible to the outside network (public network) and no direct access from the public network. Due to the different technologies and some other limitations like security measures discussed above, still mobile devices facing difficulties when establishing direct connections. This thesis proposed few workable solutions to overcome those limitations.

1.2 Goal of the Thesis

The thesis focuses on developing and evaluating different approaches to address devices in mobile networks. Due the non-unique nature of the NAT configurations applied by the mobile operators, we developed few different applications to achieve the problems. After evaluating developed applications with different test scenarios thesis proposed the best traversal approach for the particular NAT configuration.

1.3 Outline

This thesis organized as follows

Chapter2 -This chapter discusses some related background information like basic protocols, concepts and related technologies to understand the mobile communications and addressing approaches.

Chapter3 - This chapter presents five addressing approaches that can use to address a device in different situations. Details of implementations, test results, strengths and limitations of each approach also discuss in this chapter

Chapter4 - Provides the conclusion of the results observed.

Chapter5 - Discuss the related works done in the UDP and TCP hole punching to address a device.

Chapter6 - Provides future research directions of the addressing a device in the mobile network.

2

Background

This chapter provides some related background information to understand the works done in this thesis. It explains basic protocols, concepts and related technologies about mobile communications and addressing domains.

2.1 Session Initiation Protocol (SIP)

SIP (Session Initiation Protocol) is text-based Peer-to-Peer signalling protocol used to create, manage and terminate sessions in IP-based networks. The origin of the SIP was in the 1996 by Henning Schulzrinne and Mark Handley (5). The latest version of the specification is RFC 3261 (10) published by the IETF Network Working Group in June 2002. It uses existing protocols like HTTP, SMTP, and DNS etc. during communication. For the SIP, it does not necessarily to have prior established connections with end device and not making any resources reservation during communication.

SIP Addressing

A main interesting feature of the SIP is allowing establish a user location (translates user's name into their current network address). Each SIP user agent uses logical address called uniform resource identifier (URI) which likes an e-mail address instead of location specific address like an IP address. Because of this feature, the user having mobility and it is increasing the efficiency of the signalling as well. It also supports both Internet and PSTN type addresses. The General format of the SIP URI is user@domain that's very similar to e-mail address like `'sip:username@domain.ee'`.

Some examples:

```
sip:ua@wcom.com
sip:12345@ut.ee
sip:Mohan<liyanage@ut.ee>
```

Basic Architecture of the SIP

In SIP there are four logical entities dealing with a session, namely User Agents, Registrars, Proxy and Redirect servers.

User Agents

User Agents (UA) installed in client devices and in proxy servers. Client UA can be IP phone or softphone software running on a PC or a smart phone. User Agents also referred as a User Agent Server (UAS) and User Agent Client (UAC) since both are at the same location and can have the ability to send requests and received responses. When needed, user agents generate the request and forward to the nearest proxy server to call another person over the network.

Registrars

The main responsibility of Registrar server is detecting the location of the user on the network and keeps track of users within the domain. Registrar gets registrations information from users (IP address, port and username, etc.) and stores in the location database. Later proxy server will use this information to locate the particular user. During the registration, registrar server assigned the specific time period for the user location. So users should update their current location with the registrar before this time period has expired.

Proxy Servers

Proxy servers are application layer routers installed in the IP PBX. This is the first point of contact of the user agent. When the user places request, proxy server forwards that request to the recipient, if the destination is known. If the location is unknown, then forwards that request to another proxy server, which may know the location of the recipient. A proxy server is handling the routing part to establish connection between the calling party and the receiving party.

Redirect Server

The redirect server returns the received request to the client that indicating client should use different path to reach the recipient. Redirect server not established the connection like proxy server; instead it returns the location of another SIP user agent or another proxy server where the recipient might be located (figure2.1).

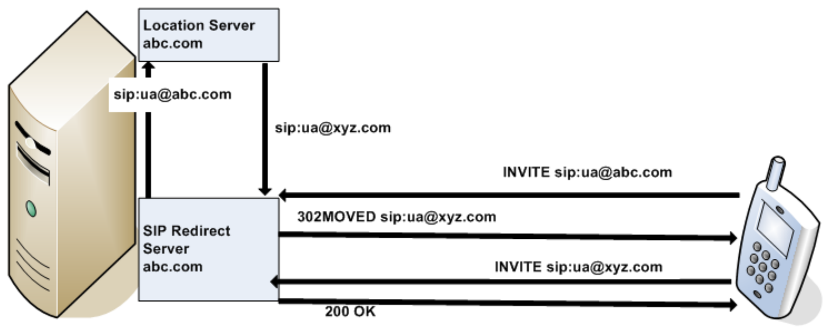


Figure 2.1: SIP Redirect server

SIP Sessions

Normally there are five methods using in SIP sessions. They are

REGISTER- to register contact information of the user (user's current location)

INVITE - to initiate call signalling process

ACK-providing reliable message exchange in INVITE

CANCLE- to terminate a call request

BYE to teardown a connection between clients

Registration Process in SIP

Every SIP user should register its current location to the SIP Registrar server by sending a request message called REGISTER. Then the Registrar server stores this information in a location server and later this information shares with other proxies to locate the user (figure 2.2).

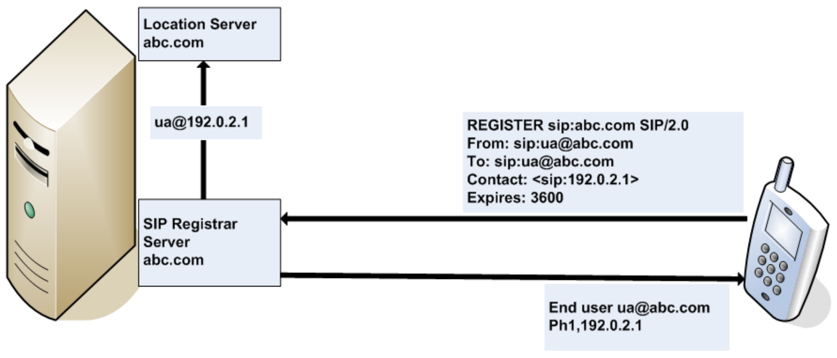


Figure 2.2: SIP Registration

Here user ’ua’ uses the IP 192.0.2.1 as its current location when registering with the Registrar server.

Initiate a Call Using SIP

Typically a user agent initiates a call by making INVITE request. But at this time user does not know the exact location of the recipient. Then this request passes to the proxy server. If the recipient is within the same domain, proxy server just forwards this request to the recipient. If the recipient is in another domain, then the current proxy server forwards that INVITE request to another proxy server who knows about the recipient. Same time Proxy1 sends back the TRYING response to the initiator(UA1 - figure 2.3). When the Proxy2 receive INVITE message and if the recipient is within its own domain it just forward the INVITE to this client and send back the TRYING response to the Proxy1 as well. If the Proxy2 does not know about the recipient it will forward the INVITE request to another server who may know about the recipient.

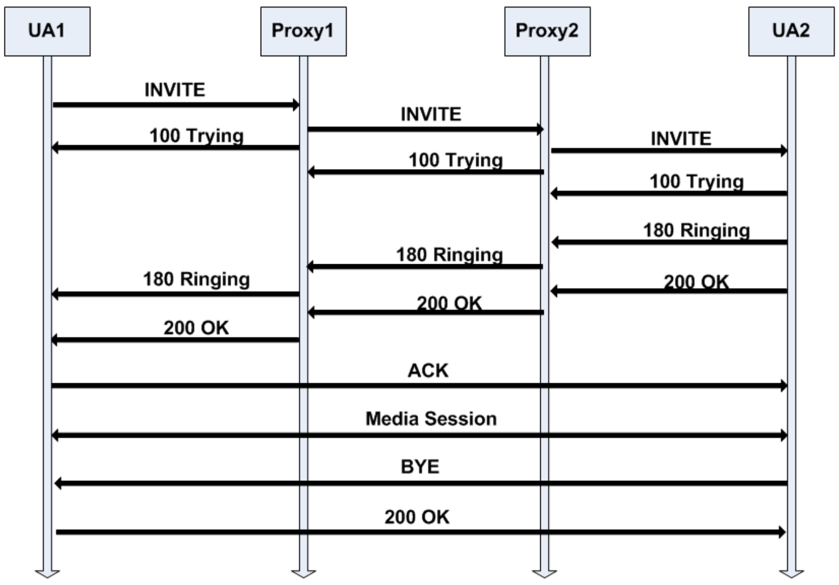


Figure 2.3: SIP Call initiation and termination

When the recipient (UA2) gets the INVITE request it sends RINGING response back to initiator through Proxy2, Proxy1 and then finally UA1. Then the recipient can accept or denied this call. If recipient accepted the call, OK response also sent back to the UA1 through Proxy2 and Proxy1. When the requested party received the OK response, it sends back ACK message to the recipient (UA2). This ACK message directly moves to the UA2 without going through proxy servers to confirm the setup of the call. After this step media exchange takes place between two phones. The proxy servers are not involved in this step and manage by different types of protocols like RTP (Real-Time Transport Protocol).

2.2 Zero Configuration Networking (Zeroconf)

Zero Configuration Networking (Zeroconf) (7)automatically creates network of devices without having to manually configure a DHCP server, DNS services, or network settings for each device. This reduces network configuration to zero (or near zero) in Internet Protocol (IP) networks where configuration and administration is difficult. Zeroconf use link-local addressing technology that's useful only in the small, isolated networks because link-local addresses and names cannot use in globally. Automatic link-local addressing mechanism already available in the IPv6, but can use "Dynamic Configuration of IPv4 Link-Local Addresses" for the IPv4 (10). A special IPv4 address block 169.254.0.0/16 is reserved for link-local use only.

Zero configuration networking is an ideal solution for environments where the administration is impractical or impossible, like wired or wireless home or small office networks, Ad hoc networks at meetings and conferences, setting up a network in disaster situation etc. but not suitable for enterprise level networks (11).

2.3 Extensible Messaging and Presence Protocol (XMPP)

Extensible messaging and presence protocol (XMPP) is an XML-based communication protocol, which was developed by the Jabber open-source community (6).It provides facilities for users to communicate with others in real time and also get their status information like available, away from the computer, offline, etc.

Addresses in XMPP

Every user on the particular network has a unique XMPP ID called Jabber ID (JID). The structure of the JID is similar to the email address like username and the domain name (username@abc.com). There is a possibility that a single user may login to multiple locations within the same domain. To identify multiple login locations for the single user, Jabber specify a resource for a particular address. As an example user Smith may have main JID as *smith@abc.com* and for mobile account that can be *smith@abc.com\mobile*. The *\mobile* mentioned the location of the client that user logged in. Sometimes Smith may login through his main terminal and his full JID for this instance will be *smith@abc.com\m_terminal*.

XMPP architecture

The XMPP communication is based on traditional client server architecture. That means clients do not communicate directly to each other. All communications are going through an XMPP server. But the architecture is decentralized like e-mail systems. That means anyone can host an XMPP server and can provide service to clients within the organization. Server to server communications enables clients to communicate in between organizations (domains). Domains can be connected through the Gateways that provide the facility by translating between foreign messaging domains and protocols (figure 2.4).

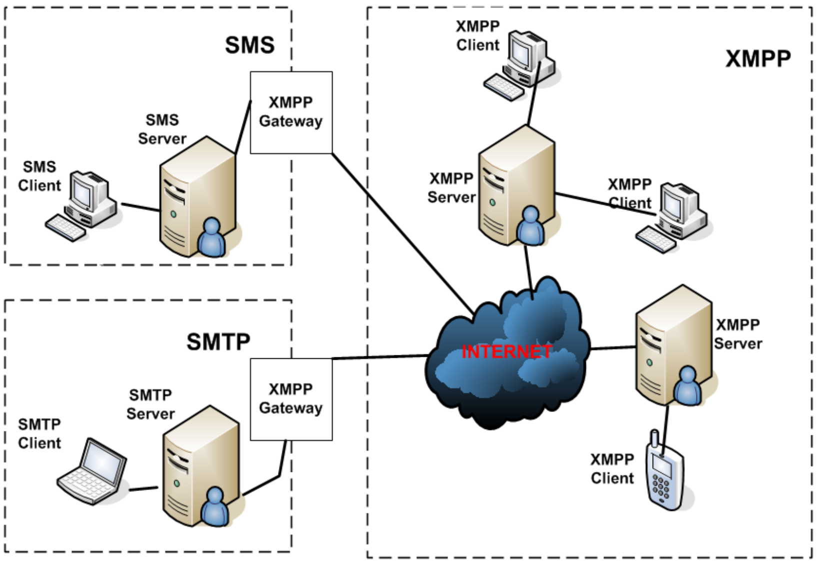


Figure 2.4: XMPP architecture

2.4 Universal Plug and Play (UPnP)

Universal Plug and Play (UPnP) is a set of networking protocols that allowed devices to set up a network and establish communications (8). UPnP mainly intended for home networks, small office networks that are not using enterprise level devices and technology. Devices that enable UPnP can dynamically connect to the network and obtain an IP address, discover other devices, share services and leave automatically from the network.

UPnP Addressing

Main addressing mechanism for the UPnP enabled device is IP addressing. Each device equipped with the DHCP client and search for the DHCP server when the device connected to the network. If a DHCP server is unavailable device assigned the IP address by itself.

2.5 Internet Protocol

Internet Protocol (IP) is considered as the main addressing protocol of the internet. It operates on the Internet Layer (12) incorporates with other protocols during data transmission. The main functions of IP include unique addressing, routing and connectionless communication. Data coming from the Transport Layer to the Internet Layer, encapsulated using IP header, makes a packet and hand over it to the Data Link Layer. Figure 2.5 illustrates the header of IP version 4.

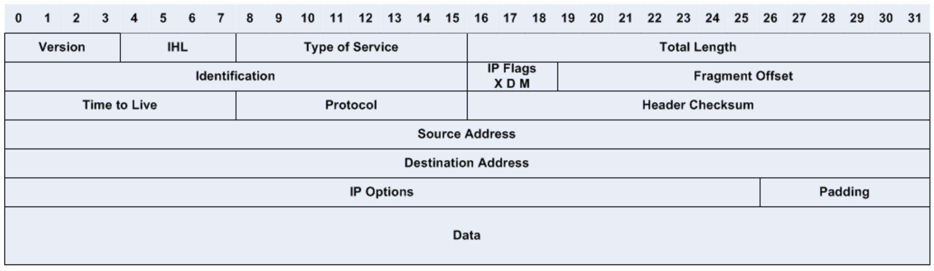


Figure 2.5: IPv4 Header

The data portion of the IP packet consists of the data coming from the Application and Transport Layers. Destination and source IP addresses indicating that from where the packet comes and to where it is going. Those addresses processed by the network nodes during packets traveling through the internet.

2.6 IP Address

When the device connected to the network, there should be a way to identify that device uniquely. If it is a small closed network like home or small office, we can assign names like Alpha, Beta, etc. for the identification. But when the network is growing and more devices are attached we face a problem when issuing and managing unique names for every device. When considering a large network like internetwork, there are millions of devices connecting together and it is impossible to use simple addressing mechanism. There must be a standard and easy to manage addressing mechanism which capable of issuing addresses for all devices without any conflict. In 1980 Internet Engineering Task Force (IETF) introduced Internet Protocol (IP) addressing for networks that use Internet protocol suite (TCP/IP) for their communication. The revised standard for the IP version 4 published in RFC 791 in September 1981 (12).

An IPv4 address is a 32 bit number store inside the networking device as serious of binary numbers. But represent using dotted decimal notation for human convenience. IPv4 provides theoretically unique 2^{32} numbers of addresses in the address space. But practically there are some reserved addresses for the special purposes, thus number of available addresses are less than the theoretical value. Due to the rapid growth of the internetwork, IPv4 addresses are insufficient and Internet Protocol Version 6 (IPv6) (13) is now working in action. But IPv4 play the main role in addressing devices because IPv6 still not acquired by the all parts of the internet.

2.6.1 Public and Private IP Address

Due to the enormous growth of the internet during past years, there was a problem of providing sufficient number of usable IPv4 addresses. As a permanent solution IETF introduced the next generation of the IPv4 that is IPv6 in the 1995. The size of the IPv6 address is 128 bits and there are logically 2^{128} possible unique addresses available. But moving from IPv4 to IPv6 is a complex task and taking years to adopt whole internetwork to IPv6.

Therefor as a temporary solution, IETF proposed private IP address space (14) which can use within an organization and allow devices to communicate locally but cannot routable over the internet. There are three private IP address blocks defined in RFC 1918. Table 2.1 shows the reserved private IP address blocks.

Block	Start	End	No. of addresses
24-bit block (/8 prefix, 1 A)	10.0.0.0	10.255.255.255	16777216
20-bit block (/12 prefix, 16 B)	172.16.0.0	172.31.255.255	1048576
16-bit block (/16 prefix, 256 C)	192.168.0.0	192.168.255.255	65536

Table 2.1: Private IP address blocks

Network administrators of the private network can use any of private IP blocks according to the requirements of the organization. The main objective of the private IP addressing is increasing the reusability of IP address. As an example organization A can use private IP address block 192.168.10.0/24 to assign addresses to their devices, but at the same time organization B also can use the 192.168.10.0/24 network to assign addresses for own devices within the organization B. Since the private networks isolated from the public networks, there is no issue of the IP address conflicts. Since these private addresses are not routable, there is a problem occurred when the device need to communicate with the devices on a public network. In order to establish such communications, private IP address translated into a routable public IP address. A particular organization may have few public IP addresses which work as reference points for the outside networks. Public IP addresses are unique and nowadays availability of the new address is also limited.

2.7 Network Address Translation

Network Address Translation (NAT) is being used by many service providers and organizations as a solution for the insufficient number of usable public IP addresses. Originally NAT was introduced as a short-term solution for the IP address shortage, but is prone to some problems in networking. There are few IP address blocks were defined as Private IP addresses. These IP addresses are also called as non routable because data packets with these addresses cant travel over internet. Since the scope is private many organizations can use same private IP addresses within their networks.

But when the device needs to communicate with the devices on public network this private IP address should be converted to the public IP address. At this point the NAT process is involved and maps internal private addresses to the external public addresses.

An internal IP address and the port (Internal Socket) is mapped to an external IP:port pair (External Socket). Whenever the NAT receives a packet to the external IP: port, it uses the map to reroute the packet back to the internal device which is using matched internal socket (illustrated in figure 2.6).

Using this method several private IP addresses can be mapped to one public IP address. Depending on the method of the translation it is called Network Address Translation (NAT) or Network Address Port Translation (15) .

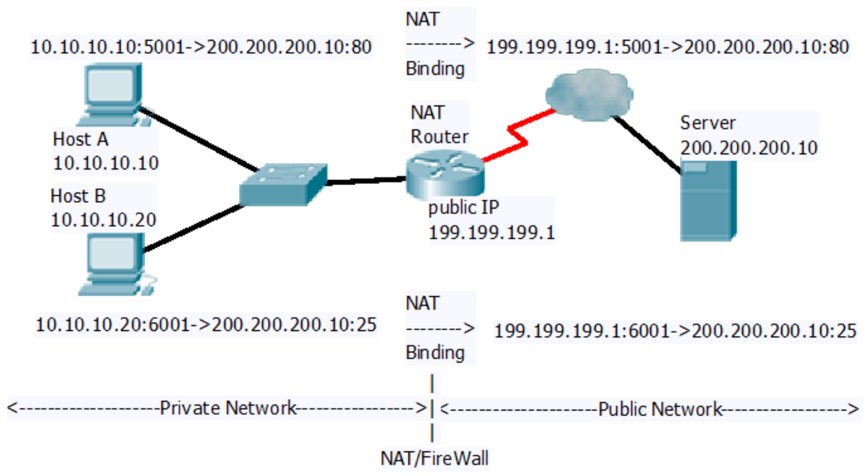
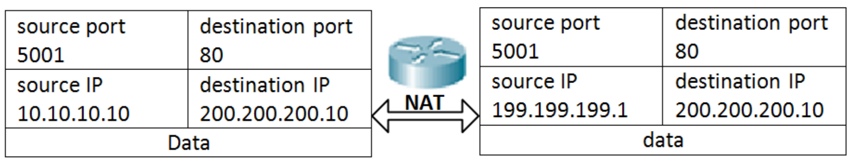


Figure 2.6: NAT process

As an example, in the figure2.6 host A which is located inside the private network use 10.10.10.10 as its private IP address. Now host A needs to send the message to the server, which on the public network. In this message source IP address stated as 10.10.10.10 and source port stated as 5001 (local socket address 10.10.10.10:5001). Also destination IP address stated as 200.200.200.10 and destination port as 80 (destination socket 200.200.200.10:80).



source port 5001	destination port 80
source IP 10.10.10.10	destination IP 200.200.200.10
Data	

source port 5001	destination port 80
source IP 199.199.199.1	destination IP 200.200.200.10
data	

Figure 2.7: Address translation in the packet

When this request moves to the NAT device (gateway router) private IP address translated into the public IP address, creates map in NAT table and keeps it for future mappings when the replies are back. In the above example (figure 2.7) NAT device translates host A's private IP address into one of the available public IP addresses. Thereafter packet moves from the private network into the public network with the translated source IP address. When the reply is coming from the server, the NAT device uses existing map in the NAT table to route the reply packet to the host A.

2.8 Commonly Used NAT Techniques

Although there is no particular standard defined for the NAT process, there are four commonly used translation techniques (16)in the industry.

2.8.1 Full Cone NAT

In full cone NAT, the internal IP address and port are mapped to an external IP and port. In this method, whenever inbound traffic comes to external port it is just forwarded to the matching internal device without doing any filtering. This is prone to some security risks as anybody from outside can send data to the internal device if he knows the external port of the mapping (illustrated in figure 2.8).

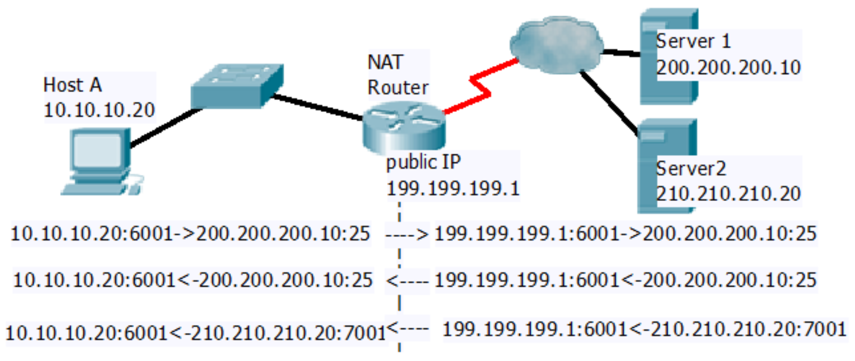


Figure 2.8: Full Cone NAT

In the above example, host A sends message to the server1. NAT device assigned 199.199.199.1 as source public IP address and create new map as

`[10.10.10.20:6001->200.200.200.10]<->[199.199.199.1:6001->200.200.200.10:25]`

This map added to the NAT table and forwards the packet to its destination where it is server1. Now server1 sends a reply to the destination 199.199.199.1:6001. In the full cone NAT it is only matching the destination port (6001) of the inbound packet. If the public port is matched, then the NAT router forwards the packet to the host A. In the above example, we can see server2 also sent a packet to the destination 199.199.199.1:6001. Here NAT device accepted this packet and forwards it to the host A. But there is not initial request sent from host A to the server2. This is a security threat because external devices can send packets to the internal devices without having a request in advance.

2.8.2 Address Restricted Cone NAT

In Address Restricted Cone NAT, internal IP and port are mapped to the external IP and port. But for inbound traffic, NAT box checks the source IP address of the packet. It allows the traffic coming from the source only if it is already addressed by the internal host. Any unsolicited packets from outside to external port are blocked. This provides additional security, but causes difficulty in connecting P-2-P applications (illustrated in figure 2.9).

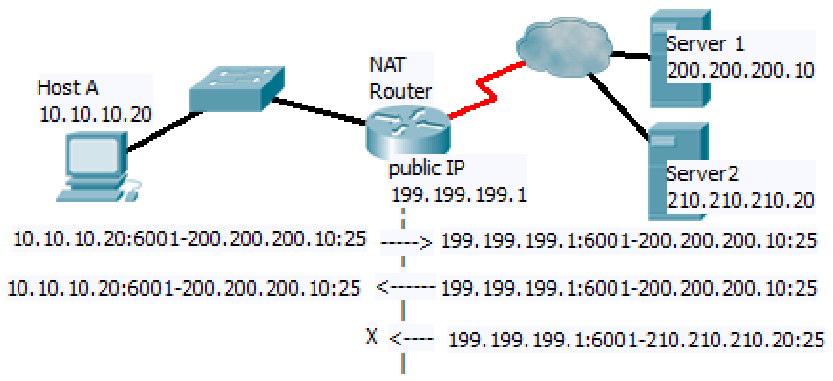


Figure 2.9: Address Restricted Cone NAT

In the figure 2.9 server1 sends packet to host A as a reply to the initial request. At the NAT router, it checks the source address of the inbound packet; it is 200.200.200.10 and match with the maps in the NAT table. Already there is a related map as

```
[10.10.10.20:6001->200.200.200.10:25]<->[199.199.199.1:6001->200.200.200.10:25]
```

which is an initial request sent by the host A. Then the NAT device accepts this inbound packet, and route it to the destination. But inbound packet from the server2 dropped. The reason is that in Address Restricted Cone NAT, inbound traffic filtered against their source address. Here NAT checked the map for the source address 210.210.210.20 in NAT table. But there is no existing map for the destination as 210.210.210.20 because no one in the private network initiated session with the server2. Because of that reason the NAT router dropped the inbound traffic from server2 to the private network.

2.8.3 Port Restricted Cone NAT

This case is almost similar to Address Restricted Cone NAT. In addition; it also checks the port number of the inbound traffic. It allows the packet only if the source IP and port of the packet match with the mapped public socket (illustrated in figure 2.10).

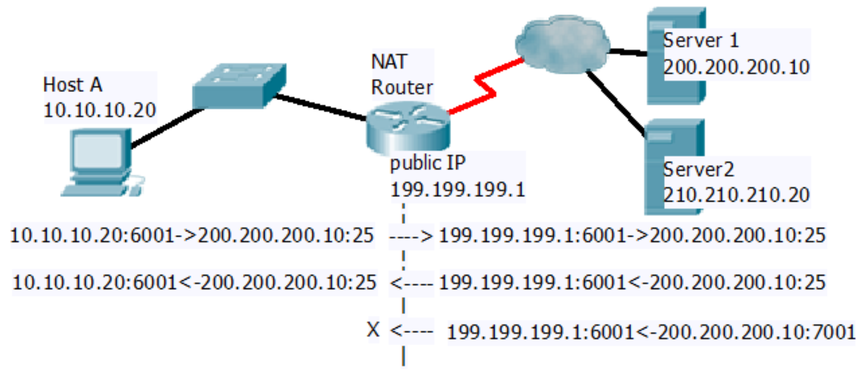


Figure 2.10: Port Restricted Cone NAT

In the figure 2.9 host A sends the request to the server1 and the NAT device create a map as

`[10.10.10.20:6001->200.200.200.10:25]<->[199.199.199.1:6001->200.200.200.10:25]`

When replying is coming from the server1, NAT device checked the inbound packets both source IP address and the port. This packet accepts because there is a map `(199.199.199.1:6001->200.200.200.10:25)` in the NAT table. But if server1 sends a packet with the source port set as 7001 this packet dropped. The reason is that although the source IP address is matched, but no matched existing map to the port 7001.

2.8.4 Symmetric NAT

This is the most restricted mapping method in comparison with other three methods. In all the above cases the internal socket port number is directly mapped to the public sockets same port number. But in symmetric NAT the internal socket is mapped to a completely different new external socket. If the internal device uses the same socket to send data to a different external host, the NAT device maps it with another new external socket. This prevents any type of unsolicited inbound traffic. But it is very hard to predict the port mappings in this NAT process even though we know the internal socket details. In Symmetric NAT only the external device that receives a packet from internal device can send a packet back to the internal host. The scenario is illustrated in figure 2.11

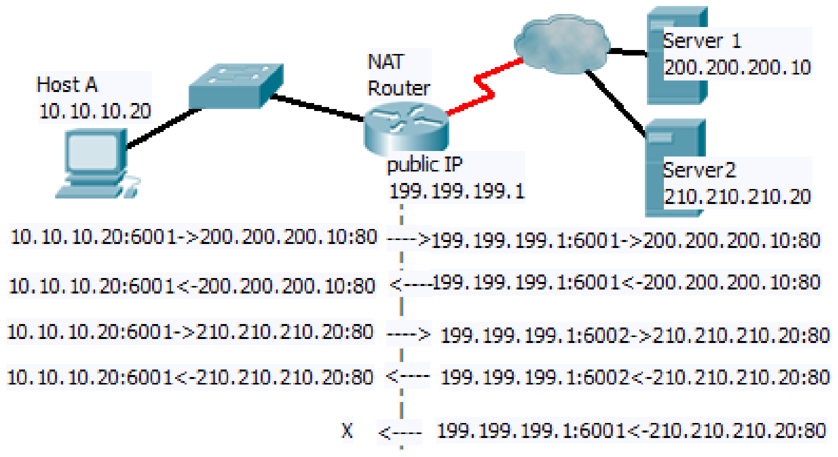


Figure 2.11: Symmetric NAT

We can see in the above example, host A sends the request to the server1. The map of this request in the NAT table is

`[10.10.10.20:6001->200.200.200.10:80]<->[199.199.199.1:6001->200.200.200.10:80]`

When the reply packet reaches to the NAT device, it checks the source address and the source port of this packet. This packet is accepted because there is map as

`[199.199.199.1:6001->200.200.200.10:80]`

in the NAT table.

Now host A sends request to the server2 using the same local port (6001) but NAT device uses different public port numbers for this transaction. Here the NAT mapping is

`[10.10.10.20:6001->210.210.210.20:80]<->[199.199.199.1:6002->210.210.210.20:80]`

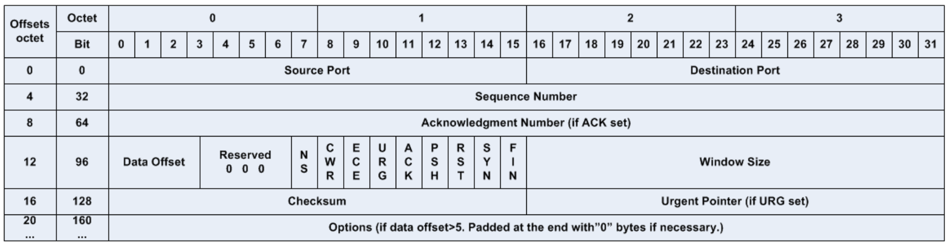
This is the main difference of the Symmetric NAT process. All NAT processes we discussed in the above were using same public port number when the traffic coming from the same local port. But in Symmetric NAT, although traffic coming from the same local socket (10.10.10.20:6001), to the different destination (210.210.210.20) it creates different map. Therefore, traffic from the server2 to the public port 6001 will drop. The reason, because there is no map for this traffic and only existing map for the server2 is

`[(10.10.10.20:6001->210.210.210.20:80)<->(199.199.199.1:6002->210.210.210.20:80]`

Because of this reason, in Symmetric NAT, only inbound traffic coming from the same source address and the same port to the same public port is accepted. NAT will drop all other unsolicited inbound traffic.

2.9 Transmission Control Protocol (TCP)

Transmission Control Protocol (TCP) is connection oriented and reliable protocol mainly used to transport data in a reliable manner (17). The size of the TCP header is 20 Bytes (figure 2.12). Some important fields of the TCP header which are interesting for the hole punching and the session establishment are described below.



Offsets octet	Octet	0								1								2								3							
	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source Port																Destination Port															
4	32	Sequence Number																															
8	64	Acknowledgment Number (if ACK set)																															
12	96	Data Offset				Reserved 0 0 0			NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
16	128	Checksum																Urgent Pointer (if URG set)															
20 …	160 …	Options (if data offset>5. Padded at the end with"0" bytes if necessary.)																															

Figure 2.12: TCP header

Source Port

The source port is the number that is assigned by the local host when it transmits data to the remote host. This is a random number which is normally higher than 1023 according to the Internet Assigned Numbers Authority (IANA) port numbering procedure (18). Source device can distinguish the sessions by referring the source port.

Destination Port

The destination port is normally a well-known port number which is used by the remote device to identify the service requested by the source device.

Sequence Number

It is a 32 bit number, which is indicating the number of data bytes transmitted. This number helps to maintain the ordered delivery of data at the receiving end.

Acknowledgment Number

This is a 32 bit number which is used to maintain the reliability of the data transmission. The receiver should acknowledge the transmitter about the received data.

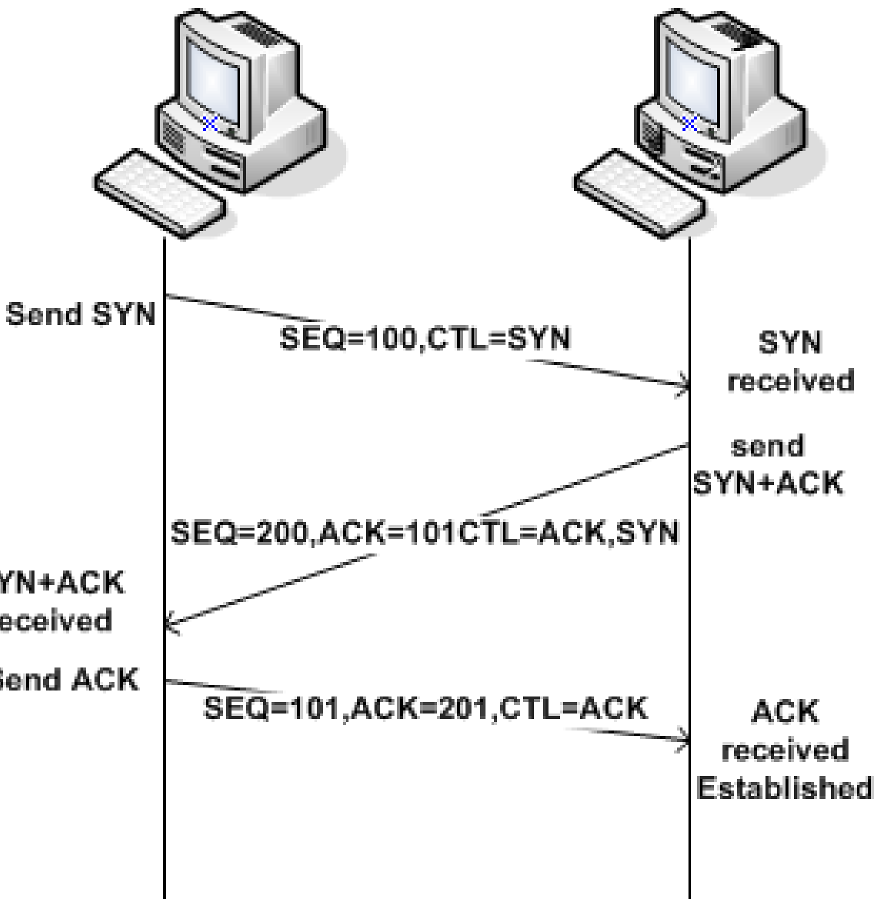


Figure 2.13: TCP Three-Way Handshaking

TCP is a connection oriented protocol. That means, first the transmitter and the receiver should establish the logical connection before the data transmission. Then data can send through this connection and finally should properly terminate the connection. The connection establishment process of the TCP is called as the Three-Way Handshaking (figure 2.13).

2.10 User Datagram Protocol (UDP)

User Datagram Protocol (UDP) is a simple Transport Layer protocol in the Internet protocol suite, which defined by the IETF RFC 768 (19). This is message oriented, unreliable and connectionless protocol mainly used to transport stream data. The UDP data packet is called as a Datagram. Applications which aren't concerned about the reliability, but need faster delivery may use UDP as their transport layer protocol. Figure 2.14 shows the structure of the UDP datagram.

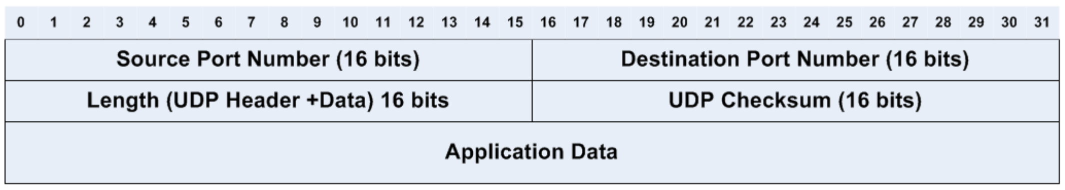


Figure 2.14: UDP Header

2.11 Mobile 3G Network

The Third Generation (3G) of the mobile networks is becoming very popular because of the ability to access the internet over mobile networks. 3G networks operate under the standards provided by the International Telecommunications Unions (ITUs) - International Mobile Telecommunications-2000 (IMT-2000) (20). The Universal Mobile Telecommunications System (UMTS) is one of the common 3G mobile communication systems widely being used in many countries. According to this standard minimum data rate for the stationary or walking users is 2Mbit/s and for a moving vehicle is 348 kbit/s. Mobile 3G also has high data rates up to 10Mbps with the HSDPA technology (High Speed Downlink Packet Access) (21). With higher data rates, 3G network enables voice and video calling, file transmission, internet surfing, online TV, viewing high definition videos, playing games and much more for their users.

2.12 IP Address Assignment in Mobile Networks

Hierarchical architecture is used to design the core network of the mobile communication systems. In UMTS the data traffic generated by each mobile traverses through the UMTS Terrestrial Radio Access Network (UTRAN) and moves to the packet-switched domain. From the packet-switched domain the data moves to the internet or other mobile networks (figure 2.15). In such a scenario, each mobile device gets the IP address from the Gateway GPRS Support Node (GGSN) located at the edge of the packet switched domain. GGSN works as an interface for the mobile network and the internet. It is impossible to provide a public IP address for each and every mobile device because there are thousands of subscribers registered under one mobile network operator. As a solution for this, GGSN assigns private un-routable IP address for the mobile and translates this private IP address into the public IP address when the user needs to access the internet.

2.13 Hole Punching

Hole Punching is one of the most effective and common method to establish Peer-to-Peer communication between devices on different private networks. This technique described in the section 5.1 of RFC 3027 (22) and the section 3.3 of RFC 5128 (23). When hosts are behind the NAT boxes it is impossible to establish a connection from the outside network because NAT boxes drop the incoming requests. Hole Punching allows a host located behind a firewall/NAT to send traffic to another host without the collaboration of the NAT itself (24), (25). With the help of the well-known Rendezvous server (RS), clients can establish these direct sessions. First hosts establish an initial session with the RS and the server later exchanges the connection details between both the hosts. Since each device knows the other peer's details they can establish a connection by sending an initial request. Figure 2.16 illustrates the hole punching process.

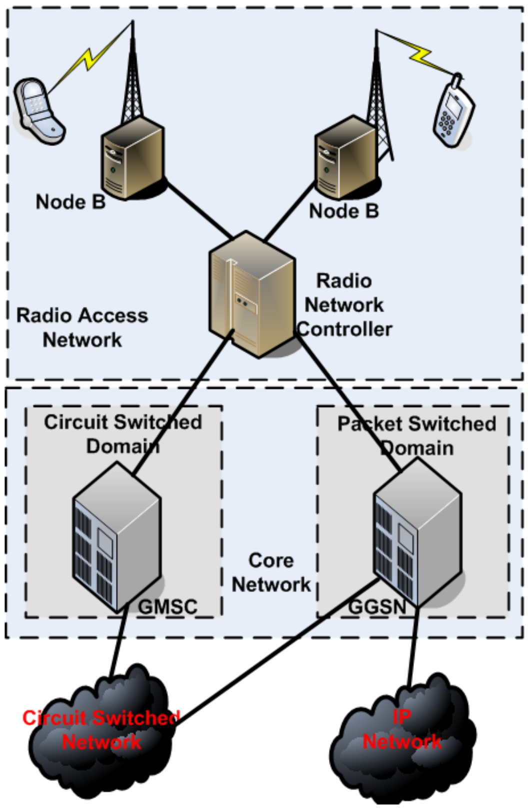


Figure 2.15: 3G Mobile network

As an example, suppose host A and B need to establish session with each other. But they can't establish the sessions directly because their NAT devices dropped the inbound requests. Then hosts first established connections with the server S. Now host A sends the connect request to the server S. Then the server learns the host A's public IP address, port number (public socket) and the private IP address, local port number (local socket) from this request. Now server forwards this information to the host B. Simultaneously server exchange the host B's information to the host A in the same manner. Since both hosts A and B know the other side public and private connection details they can start the session by sending messages to each other.

But in general first attempt will fail. When the host A sends the first message to the host B, NAT-B will drop this message because there is no map for this inbound message. But at the NAT-A it makes a map and waiting for the replies from the host B. We called NAT-A is making a hole in this communication. Host B also sends the message to the A and at this time at the NAT-A accepts the message because there is

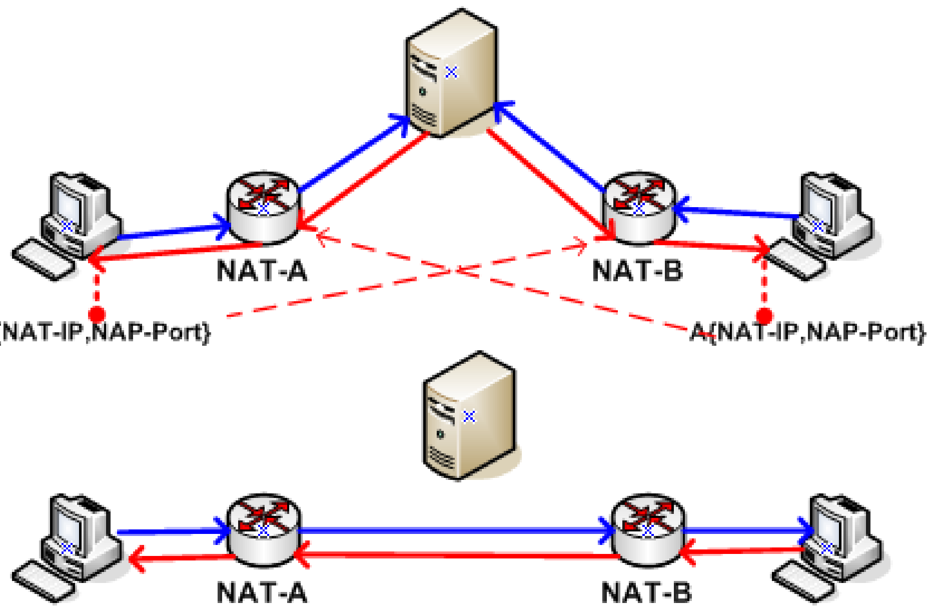


Figure 2.16: Hole punching with Rendezvous server

a map and waiting for replies from host B. When the A sends the next message to B it accepted by the NAT-B because it also waiting for replies for the existing request. Now both devices can exchange messages directly without going through the server S.

2.14 Summary

Although there are many addressing techniques available, in this thesis we focus mainly about the IP addressing for mobile devices. In this chapter, we discussed the basic features of the two main transport layer protocols namely TCP and UDP. Also discussed how it handled the problems caused due to insufficient number of IPv4 address by using private IP address and the behavior of different types of NAT configurations. Finally, this chapter summarizes about the current mobile technologies and basics of the common NAT traversal method called hole punching.

3

Addressing Device in Mobile Network

Although there were several solutions (24), (26), (27)already implemented, but most of those solutions targeted establish sessions between peers connected to the fixed networks. However, when we think of mobile networks the solutions become complicated. In this research five addressing approaches, namely Session Initiation Protocol (SIP), UDP hole punching, TCP hole punching, UDP relaying and TCP relaying implemented to address device in the mobile network. This chapter discusses all five approaches in detail and evaluates the obtained test results regarding few Estonian mobile networks, namely TELE 2 (28), Elisa (29), EMT (30) and Super (31). Initially implemented solutions are only for the Android devices (mobile phones and Tabs). Later it will develop for other types of mobiles devices.

3.1 Session Initiation Protocol (SIP) Approach to Address Devices

To implement the SIP based solution we need SIP PBX/Proxy server to register and exchange the details of SIP clients. For this task we configured open source Asterisk SIP server running on the Ubuntu server installed on Amazon Cloud as an EC2 micro instant. The server assigned with public IP address 107.20.232.29 . For the initial testing purpose we created four user accounts with names as Alpha, Beta, Gamma, Delta and numbered as 100,200,300, 400 respectively.

3.1.1 Configuring the System

In the Asterisk PBX system files like sip.conf and iax.conf contain the basic configurations for the channel drivers. The extensions.conf file contains users' details, dialling rules and their extension numbers. To setup the basic level communications, at least need to do some configurations in the sip.conf and the extensions.conf files as described in coming sections.

3.1.2 Channel Configuration (sip.conf)

The SIP channel module is the most important channel module in the Asterisk PBX. There are a few sections in this file and all section names enclose in square brackets ([ ]). Common information and default configurations about the channel drivers are in the [general] section at the top of the configuration file. Any configuration after the section name belongs to that section. Following configurations should added to the channel configuration file (sip.conf for SIP) in the /etc/asterisk/ directory.

```
[general]
context=unauthenticated ; default context for incoming calls
allowguest=no ; disable unauthenticated calls
srvlookup=yes ; enabled DNS SRV record lookup on outbound calls
bindaddr=0.0.0.0 ; listen for UDP requests on all interfaces in the server
port=5060  ; define the port for SIP

[office-phone] ; create a template for our devices
type=friend ; the channel driver will match on username first, IP second
context=incoming  ; this is where calls from the device will enter the Dialplan
host=dynamic  ; the device will register with asterisk
nat=yes  ; assume device is behind NAT

[alpha](office-phone); define user name under the office-phone template
secret=Alpha1; a secure password for this device

[beta](office-phone)
```

```
secret=Beta2

[gamma](office-phone)
secret=Gamma3

[delta](office-phone)
secret=Delta4
```

Here the ***[general]*** section is the standard section that appears at the top of the configuration file. This section contains general configuration options and can use to define default parameters as well. In the ***[office-phone]*** section it defines several options required to control the calls define under this template. The first option we configured *type* as *friend*. This tells the channel driver should first attempt to match name and then IP address. There are three types defined as

type = peer
Match incoming requests using the source IP address and port number.

type=user
Match incoming using the username in the header of the SIP request. This name should match to a section with the same name in square brackets ([]) in the sip.conf.

type=friend
This enables match both peer and user. Commonly use setting for the SIP phones.

When a request receives from a telephone it authenticates by the PBX server. Then requested extension number is handled by the context defined in the Dialplan. In the above configuration context named as ***[office-phone]***.

The *host* option can use to send a request to the telephone. When we set this value as *dynamic* Asterisk knows that the telephone will tell us its location. It is also possible to define as *static* with an IP address such as 192.168.1.50

The password for the device defines by the *secret* parameter. This can use to prevent unknown users (telephone) register with our system.

3.1.3 Configuring the Daialplan

The Dialplan is the most important part of the Asterisk PBX system. It shows how calls are flowing into and out of the system. Dialplan contains few main areas like contexts, extensions, priorities, and applications (32). Mainly Dialplan configuration divided into sections called contexts which define by using names in square brackets ([]). An extension which defines in one context is completely isolated from other extensions define in any other context. There are two special contexts called *[general]* and *[globals]* define in the beginning of the Dialplan configuration file.

When defining a channel in the sip.conf(or iax.conf, chan_dahdi.conf, etc.), one of the required parameters is context. It is the point that connections will begin. Here in sip.conf we defined a section called *[office-phone]* and under this section we define context as *incoming*. After that need to define context as *[incoming]* in the extensions.conf (Dialplan) as well.

We need to add the following configurations in the extensions.conf file located in */etc/asterisk/* directory.

```
[general]
autofallthrough=yes

[incoming]
exten=>100,1,Dial(SIP/alpha)
exten => 200,1,Dial(SIP/beta)
exten => 300,1,Dial(SIP/gamma)
exten => 400,1,Dial(SIP/delta)
```

In this dial plan we configured four extensions as 100,200,300 and 400 under the context *[incoming]*. The relevant information for this context is configured in the sip.conf as well.

To initiate SIP session, need to install SIP client application on mobile clients. IMSDroid (33)is one of the best SIP clients for the Android devices and we used it as our client application. The first step is clients should register with the PBX server. For this purpose clients must provide their local SIP name, global SIP name or number and password for the SIP account. After the successful registration clients can initiate SIP sessions with the peer clients.

3.1.4 Test Scenarios and the Results

Installed Asterisk PBX server on Amazon EC2 server and tested with client mobile devices located in TELE2, EMT, Elisa, Super mobile networks and also in the Institute of Computer Science Wi-Fi network. We obtained successful results when the client device on any mobile network mentioned above. But when the device was in the UT Wi-Fi network, connection couldnt establish. According to our observations, some ports important for the SIP communication blocked at the University gateway router. This prevents the client device (SIP-UA) to register with the SIP Registrar server and block all communications with the SIP PBX server.

3.1.5 Strengths and Limitations of the SIP

Strengths

When considering the SIP solution, it can address devices behind any type of NAT boxes because SIP PBX installed on public network where clients can access easily. Initial session establishment is done by the PBX and proxy servers. There after clients start exchanging data. There are no restrictions about the mobility of the client device. Clients can initiate sessions with peers as long as they connected to the internet without regarding the type of NAT or other configurations imposed by the mobile network. Another advantage is that simple installation and configurations in SIP. There are many SIP servers and softphone applications available freely with detailed user manuals which help administrators to set up the solution very easily.

Limitations

Mainly SIP designed to transport stream type data between peers. Applications like VoIP, Video Calling are examples for using streaming type data. But when we need to send data files over SIP, our approach might fail because most of the open source SIP proxy servers and clients do not support to transmit data files. Another drawback is application ports blocked by the firewalls. Associated TCP and UDP port numbers for the SIP communications are 5060 and 5061. But in some organizations security policy may blocking these range of port numbers which may badly impact on the SIP communications.

3.2 UDP Hole Punching using Rendezvous Server

UDP hole punching is the most common approach in NAT traversal. In the proposed solution we modified existing approach that suitable for addressing Android devices in mobile networks.

3.2.1 The Rendezvous Server

This server program is running on the Ubuntu server installed on Amazon Cloud as an EC2 micro instant. The server assigned a public IP address 107.20.232.29. Since the public IP address is assigned, any host can easily access the server even they located behind the NAT. When the server is running it is listening on local port 2020 to accept clients' requests. According to the proposed solution client should register with the server by providing clients mobile number in the IDD format in order to maintain a unique identification for the clients. Use this number as the URI for the particular client. The server is storing all registered clients' details (Mobile no, Public IP Address, Public Port, Local IP Address and Local Port) in the data structure for future matching. Figure 3.1 shows the class diagram of the Rendezvous server.

If the client registers again with the same URI, then the existing record updated with the new information automatically (figure 3.2).

The first step is client device needs to register with the server. When the client device register in the first time, the server program creates a client object and added it into the client list (figure 3.3). If the same client registers next time with the same number, then the existing client objects details updated with new details.

3.2.2 Establishing Peer Connections

The proposed solution should address the problems encountered in all different scenarios when connecting two mobile devices. As an example, sometimes both peer devices located in the same cell under one mobile operator, sometimes the peers are in two

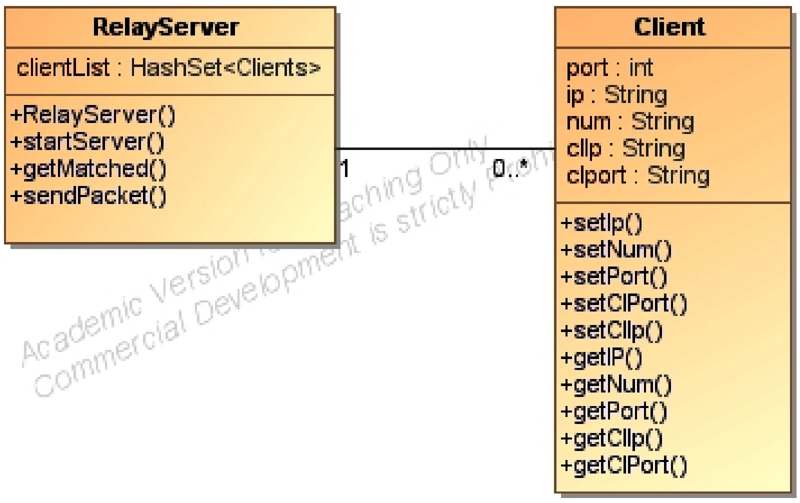


Figure 3.1: Class diagram of the server application

different locations under two mobile operators. The server will identify clients configurations and handle the situation accordingly. Following section describes the few possible scenarios and solutions from the proposed protocol.

3.2.2.1 Both Peers Located Behind the Same NAT Box

In this situation, most of the times both host devices are getting their IP configurations from the local interface of the same GGSN. So this GGSN device is working as the NAT interface for both hosts. In other words, both clients are behind the common NAT device and belong to the same private network. In this case, the connection will be established as follows.

When the host A needs to establish a connection with the host B to send some information, the first step is that both devices should register with the rendezvous server. For this they should start the installed client application and send the UDP request as "register" to the server endpoint 107.20.232.29:2020, as already explained, and shown in figure 3.4.

If the registration is successful, the server creates an object with A's registration details and adds it to the client list for later matching. If the recipient (host B) is not registered, host A can send the message to the host B by using existing communication methods (SMS, SIP message) for asking to register with the server. Extensions can also

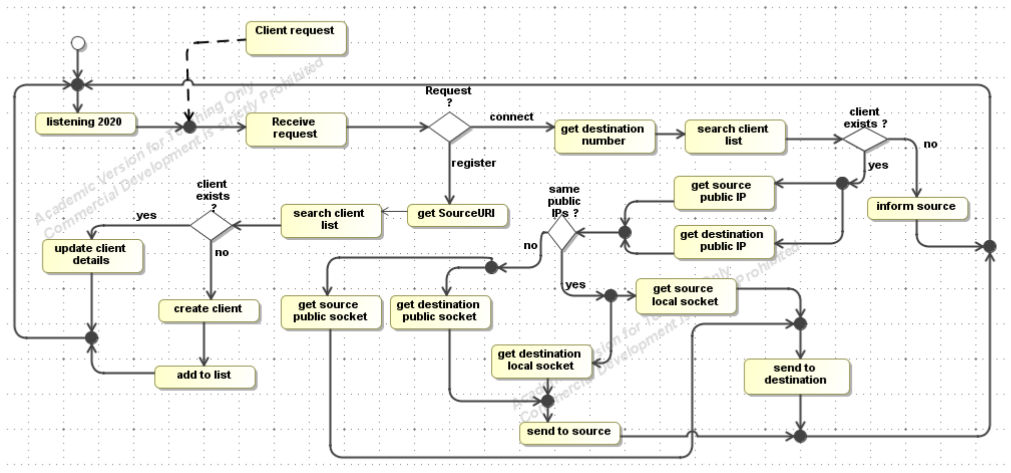
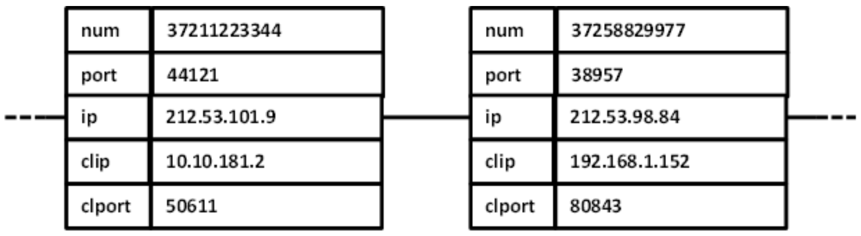


Figure 3.2: Flow chart of Rendezvous server -registration and matching



num	37211223344
port	44121
ip	212.53.101.9
clip	10.10.181.2
clport	50611

num	37258829977
port	38957
ip	212.53.98.84
clip	192.168.1.152
clport	80843

Figure 3.3: Client list

be introduced to the current approach, when it is widely adopted, such as the mobile networks themselves can register the devices to the public rendezvous servers once they show interest in exchanging packet data.

Once host B gets the invitation from the host A, it also registers with the server by sending UDP ”register” request. Registration details of the host A and B shown in table 3.1.

Now host A can send the request as ”connect” with the host B’s mobile number as the destination. At this point server processes this request and stores the information in a temporary object. Then the server gets the destination number of this temporary client object (37258226048) and looks for a matching client in the client list. Since host B is already registered in the client list the server can find the connection details of the B. Steps 1 and 2 in figure 3.5 represent the registration process.

To identify the scenario, whether both hosts are behind the same NAT, the server does further matching with public IP addresses of both hosts A and B. If the both

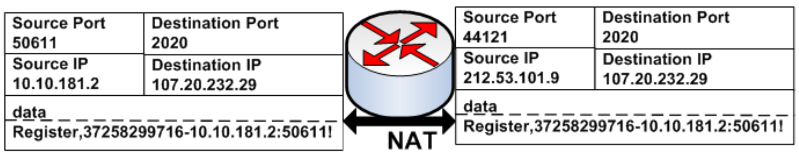


Figure 3.4: Details of the datagram in registration

	Host A	Host B
Mobile No	37258299716	37258226048
Local IP	10.10.181.2	10.10.181.10
Local Port	50611	60011
Public IP(at NAT box)	212.53.101.9	212.53.101.9
Public Port(at NAT box)	44121	80011

Table 3.1: Registration details of hosts A and B

public IP addresses of A and B are the same then they should be behind the same NAT box. In this situation hosts A and B only need to know their local endpoint details to initiate a connection.

Then the server crafts a Datagram and inserts host A's local endpoint details (10.10.181.2: 50611) and sends it to the host B. Also server crafts another Datagram and inserts host B's local socket details (10.10.181.10: 60011) and sends it to the host A. The procedure is illustrated in figure 3.5, steps 3, 4 and 5. Now both devices having each others connection information and can start the communication. In this scenario, packets route within the local network without going out through the NAT box (step 6 in the figure3.5).

3.2.2.2 Peers Located Behind Different NAT Boxes

One mobile operator can have a few GGSN devices and it is possible to get the IP details for mobile devices from the different GGCN s. Sometimes clients may be in two different mobile networks. In this situation, one client belongs to the one private network and the other client belongs to another private network. In order to initiate a connection, both clients should learn the public socket details of the other side. The

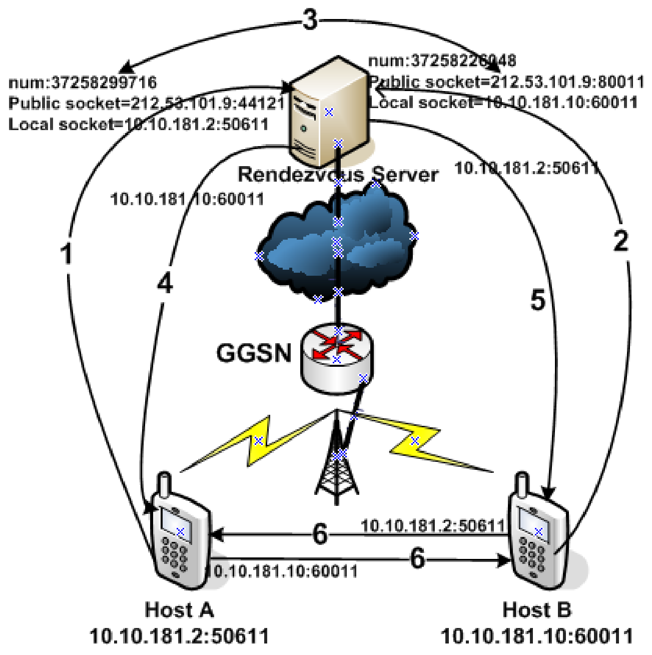


Figure 3.5: Establishing connection through hole punching when peers are behind the same NAT

registration phase is almost the same as previous scenario and table 3.2 shows the registration details of hosts A and B.

	Host A	Host B
Mobile No	37258299716	37258226048
Local IP	10.10.181.2	10.100.100.2
Local Port	50611	60011
Public IP(at NAT box)	212.53.101.9	89.18.102.10
Public Port(at NAT box)	44121	80011

Table 3.2: Registration details of hosts A and B

The server uses the same algorithm to match the destination device as described earlier. But in this case peers public IP addresses are not same because they are coming from different NAT boxes. Because of this reason, the server should exchange hosts public socket as well as private socket details between peers to initiate the sessions. At this point, server crafts a datagram and inserts host B's public and private socket details and sends it to the host A. Similarly, the server sends host A's public and private socket details to host B by using another datagram (figure 3.6, steps 3, 4 and 5).

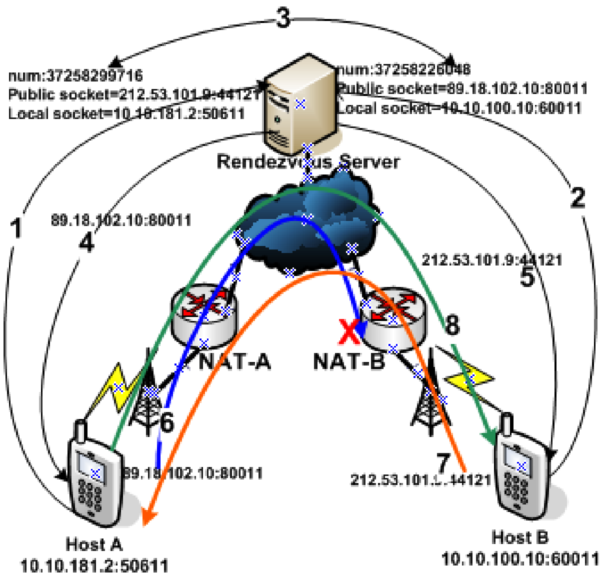


Figure 3.6: Establishing connection through hole punching when peers are behind different NATs

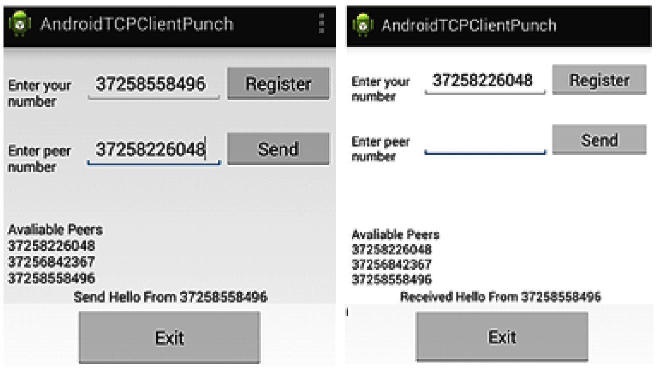
The next step is both peer initiates sessions simultaneously. Once A got the B's public socket details, then A will send a packet to the B. At the NAT-A it creates a map for this session and forwards it to the host B's public end point, here it is 89.18.102.10:80011. But at the NAT-B this packet is dropped because it is unknown inbound packet for the NAT-B (figure 3.6, step 6). But still NAT-A is waiting for the reply from host B (hole is punched).

Same time host B also sends the packet to the host Afis public socket, here it is 212.53.101.9:44121. NAT-B creates a map and forwards this packet to the A. Now this packet can reach the host A because NAT-A is waiting for a reply from the B (figure 3.6, step 7). In other words, this packet can travel through the punched hole in the NAT-A and reaches the host A. After this step, all other packets from host A to host B can travel through the NAT-B because there is a punched hole for traffic coming from host A (figure 3.6, step 8).

3.2.3 Test Scenarios and the Results

3.2.3.1 Test Scenario 1 - Both hosts are located in the same mobile network

The scenario is demonstrated without any troubles in TELE2 network. Here both mobile devices use TELE2 3G/4G internet connection. Host A is a Samsung Galaxy SIII phone with the mobile number 37258299716. Host B is a Nexus-5 phone with the number 37258226048. Both hosts A and B were registered with the server. After that host A sends "connect" request to the server to establish connection with host B. The server checks the hosts registration number to locate the destination. The connection details are later exchanged, as already explained. After this step, both devices were able to send and receive messages from each other. Figure 3.7 displays the output captured from two hosts when the host A sent the message and the host B received the message on our developed client application.


AndroidTCPClientPunch

Enter your number: 37258558496

Enter peer number: 37258226048

Avaliable Peers
37258226048
37256842367
37258558496

Send Hello From 37258558496

AndroidTCPClientPunch

Enter your number: 37258226048

Enter peer number:

Avaliable Peers
37258226048
37256842367
37258558496

Received Hello From 37258558496

Figure 3.7: Android application with the messages

Same test scenario we applied to the EMT mobile network. The result was the same as described above. For better understanding of the proposed approach we tested when mobile devices connected to the UT Wi-Fi network. The test result was also successful as expected. But when both mobile devices are on the Elisa mobile network our proposed solution was filled. We have done the extensive research on the NAT technique used by all mobile networks and observed results discuss in the later section. Table 3.3 summarizes the results

‘

Connection initiator	Receiving mobile	Result
TELE2	TELE2	Success
EMT	EMT	Success
UT Wi-Fi	UT Wi-Fi	Success
Elisa	Elisa	Fail

Table 3.3: Test results of the UDP hole punching approach - peers in the same mobile network

3.2.3.2 Test Scenario 2 - Hosts are located in different mobile networks

In this scenario the mobiles are connected to different mobile networks and tried to exchange messages. Consider the case where host A was in the TELE2 mobile network and the host B was in the EMT mobile network. As the first step both devices register with the server. Now host A sends a message to the host B. The server locates the host B (3725855496) and exchange connection details. Then the devices tried to exchange messages. In this case, both devices can send messages and received messages. Thats mean when devices in TELE2 and EMT, either device can send or receive data successfully. But when sending messages from host A in TELE2 to host B in Elisa was failed. However, when sending messages from host B (B initiates connection) to host A it was successful. Table 3.4 summarizes the results of different scenarios.

3.2.4 Strengths and Limitations of the UDP Hole Punching

According to the test results we observed that some hosts cannot establish connections properly. There are few internal and external factors that will limit the functionality of our implementation. Main external factor is the differences in behavior of the NAT techniques used by the mobile operators. To get a better idea about the behavior of the NAT box applied, we implemented a simple application which can retrieve devices own public socket details. This application sends two messages using same local socket to two different servers. These servers replied to the host with the public socket details assigned by the respective NAT for that communication. Table 3.5 summarizes the result of the NAT translation of few mobile operators in Estonia.

Strengths

Connection initiator	Receiving mobile	Result
TELE2	EMT	Success
	Elisa	Fail
	Super	Success
EMT	TELE2	Success
	Elisa	Fail
	Super	Success
Elisa	TELE2	Success
	EMT	Success
	Super	Success
Super	TELE2	Success
	EMT	Success
	Elisa	Fail

Table 3.4: Test results of the UDP hole punching approach - peers in different mobile networks

When considering the UDP hole punching approach it is easy to exchange messages between devices (peers, server, etc.) because of the connectionless nature of UDP. It is not necessary to maintain simultaneous open sessions that extensively utilize devices resources. Also, UDP minimize the network latency because there are no session establishment and termination stages.

Limitations

When analyzing the obtained results for UDP hole punching, if the connection initiator in the Elisa and receiving device in TELE2, EMT or Super mobile networks, success rate became 100%. The meaning of that TELE2, EMT and Super using simple type of NAT configurations which not filtered inbound traffic against source address or port. When analyzing the above table (table 3.5) we can see that TELE2 always assign the public IP address to a mobile device. Since, probably there is no NAT translation applied, devices from any mobile network can directly initiate session with a device that attached to the TELE2 mobile network.

In the EMT mobile, it assigns a private IP address to the clients and translates it into a public IP address when needed. During NAT it keeps the local port number as the same and only translates the private IP address into the public IP address. Since

Mobile network	Local socket	Public socket (from server1)	Public socket (from server2)
TELE2 (subscription 1)	213.101.198.63 :57952	213.101.198.63 :57952	213.101.198.63 :57952
TELE2 (subscription 2)	83.176.2.178 :40756	83.176.2.178 :40756	83.176.2.178 :40756
EMT (subscription 1)	10.145.20.13 :52142	217.71.46.13 :52142	217.71.46.13 :52142
EMT (subscription 2)	10.128.194.59 :58296	217.71.44.59 :58296	217.71.44.59 :58296
Elisa (subscription 1)	10.27.201.251 :51605	194.150.65.155 :55251	194.150.65.155 :55252
Elisa (subscription 2)	10.26.164.121 :33045	194.150.65.24 :32322	194.150.65.24 :32322
Super	10.26.164.121 :33045	194.150.65.24 :32322	194.150.65.24 :32322
UT Wi-Fi	10.26.164.121 :33045	194.150.65.24 :32322	194.150.65.24 :32322

Table 3.5: NAT across different mobile networks in Estonia

EMT use same port for the both client socket and the public socket hole punching approach worked.

But when considering Elisa mobile network address translation is more likely similar to the symmetric NAT translation. We found that Elisa translates local socket into a completely new public socket. In the UT Wi-Fi network, NAT configuration is same as the Elisa but assigned ports randomly for the public socket. Elisa assigns ports for public socket by an incremental manner for different destinations but in the UT it is completely unpredictable. In both networks even the request coming from same local socket to the different destination, NAT assigns a new socket for that session. It is very difficult to implementing hole-punching because of the port prediction involved in this case.

According to results obtained for the UDP hole punching we can see if the receiving mobile in the Elisa then no other hosts can initiate session with that mobile. Because in our approach we assume that NAT use the same port for both local and public sockets (cone type NATs). But Elisa use different port numbers for the public socket which prevent establishing session from the other devices. Even if both devices in the Elisa but no one can establish sessions.

UDP is unreliable protocol. That means we don't have any guarantee about the data we transmitted. This is ok for the applications like online games, some multimedia applications those are not necessary to receive all packets but not for the applications contain critical data.

3.3 TCP Hole Punching

There are some applications using TCP as a transport layer protocol. For those types of applications, TCP hole punching should use to establish TCP session between peers. There are several solutions (25), (26), (34)already implemented for the TCP hole punching, but target establishing sessions between peers connected to the fixed networks. To address these problems we propose to use UDP sessions with the Rendezvous server to send and receive peer information and TCP sessions in the hole punching process. In this approach functionality of the rendezvous server is same as the server used in UDP hole punching and can be use the same server.

3.3.1 Establishing TCP Peer Sessions

In the TCP hole punching also we use the help of rendezvous server, which can exchanges connection information between peers. In this approach functionality of the rendezvous server is same as the server used in the UDP hole punching and used the same server with the different port number. The reason is why we use UDP for registration instead of TCP because UDP is a connectionless protocol that can reduce the overhead of maintaining too many sessions for registration requests.

3.3.1.1 When both peers are behind the same NAT

We use the same algorithm for the registration and matching hosts. If the registration is successful host A can send the "connect" request to the server asking connection details about host B. Then the server exchange connection details between host A and host B.

	Host A	Host B
Mobile No	37258299716	37258226048
Local IP	10.10.181.2	10.10.181.10
Local Port	50611	60011
Public IP(at NAT box)	212.53.101.9	212.53.101.9
Public Port(at NAT box)	44121	80011

Table 3.6: Registration details of hosts A and B

Once the datagrams are received from the server, both hosts A and B should create a TCP listening socket according to their local port number which was sent to the other host. As an example host A and B should create listening TCP sockets with the ports 50611 and 60011, respectively. Host A then sends TCP SYN message to initiate the session. At the host B there is a TCP listening socket running on port 60011 to accept TCP SYN message from the host A. Then host B sends a SYN-ACK to Host A. Host A finishes the establishment of the TCP connection by confirming with an ACK. The complete scenario is shown in sequence diagram figure 3.8. Notice that, since the TCP listening sockets are running on the both host devices any device can initiate the connection procedure, after the registration.

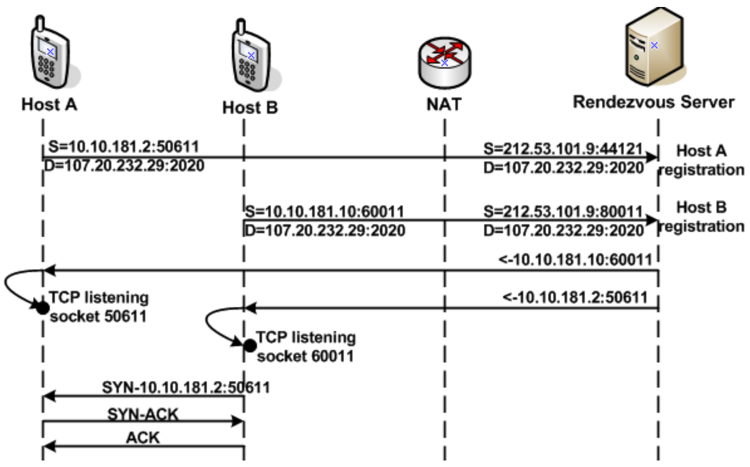


Figure 3.8: Sequence diagram showing the time-line of activities performed during the connection establishment between the devices

3.3.1.2 Peers are located behind different NAT boxes

When the peppers are in different mobile operators they are located behind different NAT boxes and hence are on different private networks. In order to initiate a connection, both clients should learn the public socket details about each other. The registration phase is almost the same as the previous scenario and table 3.7 shows the registration details of hosts A and B.

	Host A	Host B
Mobile No	37258299716	37258226048
Local IP	10.10.181.2	10.100.100.2
Local Port	50611	60011
Public IP(at NAT box)	212.53.101.9	89.18.102.10
Public Port(at NAT box)	44121	80011

Table 3.7: Registration details of hosts A and B

The server uses the same algorithm to match the destination device and exchanged information as described earlier in the UDP hole punching. Once the datagrams are received, the next step is starting a TCP session between the peers. Here both peers initiate sessions simultaneously. Host A starts the TCP listening socket on port 50611 and sends the TCP SYN message to the host B. At the NAT-A it creates a map for this session and forwards it to the host Bfis public end point, here it is 89.18.102.10:80011. But at the NAT-B this packet is dropped because it is unknown inbound packet for the NAT-B. But still NAT-A is waiting for the reply from B (hole is punched).

Same time host B also starts a TCP listen socket on the port 60011 and sends the TCP SYN packet to the host As public socket, here it is 212.53.101.9:44121. NAT-B creates a map and forwards this packet to the A. Now this packet can reach the host A because NAT-A is waiting for a reply from the B. Host A accepts this TCP SYN message as there is a TCP listen socket on port 50611 to accept SYN messages. Then host A sends SYN-ACK message to the B and NAT-B allows this packet because there is a hole punched during initial step. Now B accepts SYN-ACK and confirms to A using ACK message. Once the ACK received by the host A then both devices have established TCP session which can be used to exchange data between them (2 way communication).

It is interesting to note that, in the above scenario, host A wanted to initiate the session, however if one thinks from a pure TCP perspective, the scenario ended up as though host B initiated the TCP session.

3.3.2 Test Scenarios and the Results

3.3.2.1 Test Scenario 1 -Both hosts are located in same mobile network

The scenario is demonstrated without any troubles in TELE2 network. Both hosts A and B are registered with the server as already explained. After that the details are exchanged and both devices were able to send and receive messages from each other. Once the TCP connection was established, both hosts can send and receive messages because TCP connection is full duplex.

3.3.2.2 Test Scenario 2 - Hosts are located in different mobile network

In this scenario the mobiles are connected to different mobile networks and tried to exchange messages. Consider the case where host A was in the TELE2 mobile network and the host B was in the EMT mobile network. Server exchanged the connection details after the registration. Then the devices tried to exchange messages. Here we observed interesting results. When sending messages from host A (A initiates connection in TELE2) to host B (in EMT) our proposed solution failed. However, when sending messages from host B (B initiates connection) to host A it was successful. Similar results we observed when host A is in TELE2 and host B in Elisa.

Table 3.8 summarizes the results of different test scenarios with the three common Estonian mobile networks. So we observed when one of the devices is in the TELE2 mobile network, a connection can be established from any other mobile network, by making the second device to initiate a connection. It should be remembered that once the connection is established, both the devices can send the messages to each other as long as the connection persists (TCP connection is bidirectional).

3.3.3 Strengths and Limitations of the TCP Hole Punching

Strengths

TCP is a reliable transport layer protocol. TCP established logical connection before transmitting data and use sequence numbers to maintain the order of delivery.

Connection initiator	Receiving mobile	Result
TELE2	EMT	Fail
	Elisa	Fail
EMT	TELE2	Success
	Elisa	Fail
Elisa	TELE2	Success
	EMT	Fail

Table 3.8: Test results of the TCP hole punching approach across different mobile networks

When need to address a peer device with reliability we can use TCP hole punching. In the TCP hole punching, our approach is working if the receiving device in the TELE2 network.

Limitations

When implementing our solution we considered about the way that NAT maps local socket with the public socket. But in the TCP session, sequence numbers also play a major role. Some NAT boxes are also checking the sequence numbers of the incoming TCP packets. If this sequence number is invalid with the current session, NAT just closes the public socket (punched hole) that resulting impossible to establish a connection from the outside peer. To overcome this problem we should also learn the sequence numbers of end points TCP process. But to capture the sequence numbers we need the help of the raw sockets. Creating a raw socket is not difficult for a PC, but is very difficult for Android devices, which most often needs rooting the device. We have a solution tried based on this approach, but we think this approach is too invasive for the device (35). Another problem is that when the SYN packet dropped at the other side NAT device, it generated ICMP error message and send back to the source device. Some NAT devices sensitive to this ICMP error message and just close the public socket. This is also a limitation when applying our solution with that type of NAT configurations.

3.4 Establish UDP Sessions using Relay Server

Relaying is the most reliable NAT traversal method that works for any type of NAT configuration. It is working as standard client/server communication where clients always communicate with the server. But this method is more expensive than the hole punching. The server may freeze due to heavy work load if more users relaying data simultaneously.

3.4.1 UDP Relay Server

Relay server also running on the Amazon Cloud with the public IP address 107.20.232.29. It is listening on local port 9090 to accept clients' requests. The registration process of the Relay server is same as the Rendezvous server discussed above. Only difference is instead of exchanging connection information between hosts the Relay Server relaying messages between hosts.

3.4.2 Establishing UDP Relay Connections

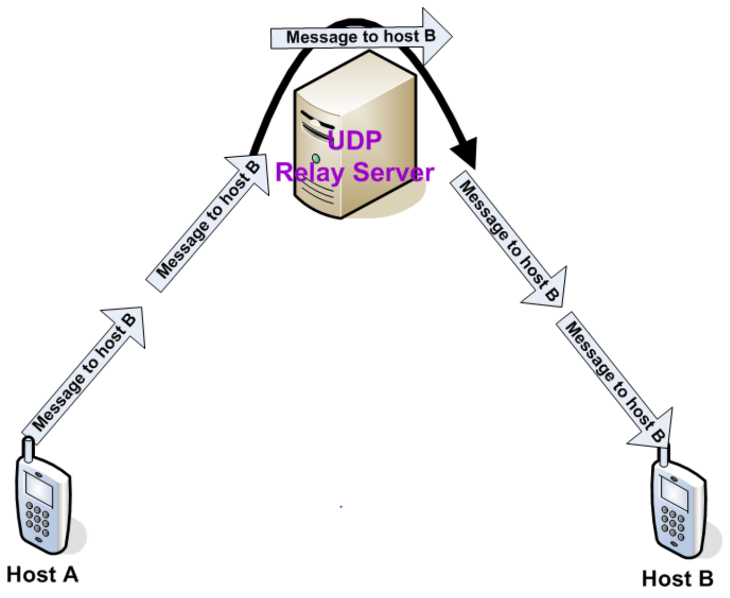


Figure 3.9: UDP Relay Server

As shown in the figure 3.9 both should establish the initial connection with the Relay server R to the R's global IP address 107.20.232.29 and port 9090. When the

A needs to send messages to the host B first host A sends the message to the server R asking forward it to the host B. Then the server extracts the data of that message, crafts a new datagram with this data and forwards it to the B using B's existing details. This datagram accepted by the NAT-B because there is map already at the NAT-B with the Relay server. Host B should follow the same procedure that host A did when it needs to send a message to host A. An interesting feature of this mechanism is it works with any type of NAT configuration.

3.4.3 Test Scenarios and Results

Hole punching approach was not working in some test scenarios which used restricted types of NATs and we applied UDP Relaying approach to address those challenges. In the first scenario host A was in the TELE2 and host B was in the Elisa mobile network. Both are done the registration with the relay server. Now host A send the message to the relay server asking forward it to the host B. Here message successfully received by the host B that located in the Elisa. We extended our test cases with other mobile networks and the UT Wi-Fi network that achieved successful result in all cases. Table 3.9 shows the results

3.4.4 Strengths and Limitations of the UDP Relaying

Strengths

According to the results we obtained that showed UDP relaying can establish sessions between peers in all cases. So we can apply relaying approach for restricted type NATs where the hole punching approach does not work.

Limitations

If there are more devices trying to exchange messages through the relay server then the performance of the server may be degraded. Also, UDP is unreliable and does not provide any guarantee for the data transported.

3.5 Establish TCP Sessions using Relay Server

As a solution to overcome some limitations in the TCP hole punching we propose TCP Relaying server. The functionality of the TCP relaying server is exactly same

Connection initiator	Receiving mobile	Result
TELE2	EMT	Success
	Elisa	
	UT Wi-Fi	
	Super	
EMT	TELE2	Success
	Elisa	
	UT Wi-Fi	
	Super	
Elisa	TELE2	Success
	EMT	
	UT Wi-Fi	
	Super	
Super	TELE2	Success
	EMT	
	UT Wi-Fi	
	Elisa	

Table 3.9: Test results of the UDP Relaying

as the UDP relay server, but use TCP as transport protocol which provides the reliability. With the TCP relaying peer can establish reliable TCP sessions with other peer through a relay server. When exchanging messages peers use existing TCP session which established during registration (figure 3.10).

3.5.1 Test Results

Host A was in the TELE2 and host B was in the Elisa mobile network. Host A sent the message to host B and it was successful. We tested TCP relay approach with all mobile networks and UT Wi-Fi also. As expected, we got successful results for all test cases.

3.5.2 Strengths and Limitations of the TCP Relaying

Strengths

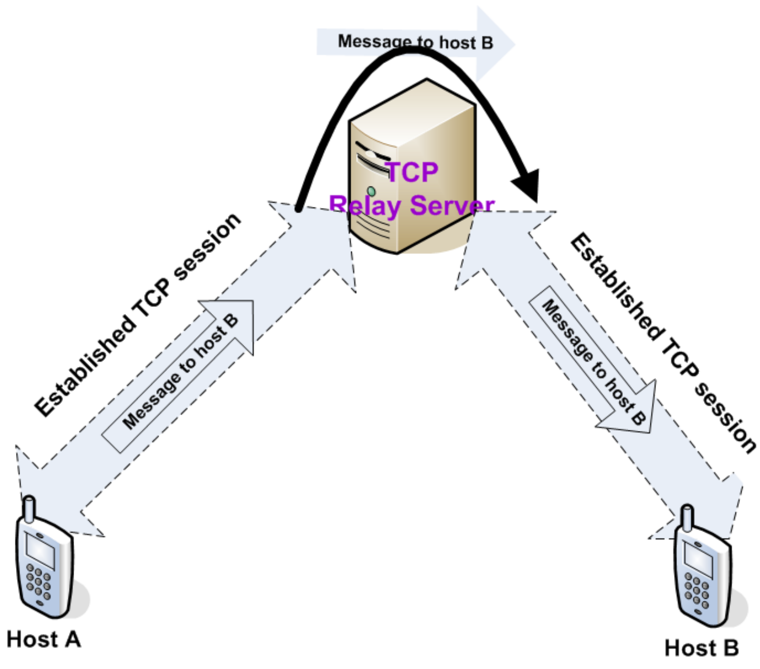


Figure 3.10: TCP Relay Server

When considering the reliability, TCP relaying is the best option because it can apply to any mobile network without considering any NAT configurations.

Limitations

TCP always maintains established sessions that may be expensive for the server processing. When the situation like a large number of users registered with the server and trying to exchange data, the server should handle all requests with the established sessions separately. This situation tender to degrade the server performance and sometimes may freeze the services.

3.6 Summary

This chapter discussed five addressing approaches, namely as SIP, UDP hole punching, TCP hole punching, UDP relaying and TCP relaying. All approaches are implemented and tested with the different mobile networks to evaluate their suitability/performance. SIP approach worked with all tested mobile networks, but it did not support to send data over SIP channel. It can be used to make VoIP sessions between peers. The UDP hole punching approach worked well with all mobile networks except Elisa, because Elisa used to restrict the NAT configuration. Since UDP is unreliable protocol there is no guarantee of data transmitted. TCP hole punching was more complex than UDP because there are some internal factors like sequence number, ICMP messages affected on the session establishment. In this test if the receiving device in the TELE2 mobile network, any device from other networks can establish TCP sessions successfully. In other situations result may be failed. In the relaying approach it worked for any mobile network irrespective of the configurations imposed by the mobile network. Test results showed that both TCP and UDP performed as same as by achieving 100% success rate. It is worth to mention that there is no unique approach can address all the scenarios because every approach has it's own advantages and limitations.

4

Conclusions

This thesis proposed solutions how to address device in mobile network by eliminating some limitations. Each approach proposed here was implemented and tested with few Estonian mobile networks. When discussing about the addressing for the mobile devices, it is impossible to provide all in one solution due to the different policies imposed by the mobile operators. Because of this reason here proposed, implemented and evaluated five different addressing approaches to address Android devices in mobile networks. The first approach was SIP based addressing implemented using Asterisk PBX server installed on Amazon EC2 instance and IMSDroid client installed on Android devices. Successful results obtained for all tested mobile networks, but failed in the UT Wi-Fi network due some useful ports blocked by the firewall. Then UDP hole punching approach implemented using rendezvous server with the developed android client application. The result was successful for all mobile networks except Elisa. In Elisa there was an inconvenient NAT policy that blocks the hole punching. Next approach was TCP hole punching for the applications using TCP as transport protocol. In this test scenario, it was only successful when the receiving device was in the TELE2 mobile network. To overcome the limitations of some unfair NAT configurations, we implemented relaying solution for the both TCP and UDP applications. Test results for the relaying approach was 100% successful for all tested mobile networks. According to the observed test results some approaches fail in some test scenarios. There are few internal and external factors affected to results of proposed addressing solutions. Main major external factor was the NAT configuration applied by the mobile operator. Since there is no unique standard for the NAT process, mobile operators applying different policies depending on the type of the NAT device installed. Because of this reason one approach may be working in one mobile network and may not in another mobile network. Also, when considering the TCP hole punching, management of the sequence numbers in the TCP session is very important and for that purpose mobile device needs root access. But normally using rooted devices is not fair for the normal users and the developed TCP hole punching application was not working as we expected. The next generation of the IP addressing is IPv6, which having enough number of addresses for every device connected to the Internet. With the IPv6 applications we can eliminate the difficulties imposed by the NAT devices and can establish a direct session with any device. With the IPv6 establish the machine-to-matching connections are easier than ever and can see the growth of many IPv6 compatible applications. But still IPv6 is not adopted by the all parts of the internet and may take few more years for full functionality.

5

Related Work

There were many studies were undergone and many different solutions proposed to address a device. But most of proposed protocols implemented to address a device attached to the fixed network. This chapter describes some related studies/implementations done to addressing devices during past years.

5.1 A NAT Traversal Protocol for Peer-to-Peer Networks

In this study authors presented a traversal protocol using existing Session Traversal Utilities for NAT (STUN). The authors developed and tested a software for the P-2-P applications to automate the NAT traversal (36). According to their protocol first host needs to join the P-2-P overlay network and should find own NAT type and the public socket details. In this step host can use STUN server that is available in the public network. After detecting the type of the applied NAT, the host makes a decision of the traversal type according to the own NAT type. Host host can make the decision of the traversal method by referring information available in the table 5.1.

NAT Type	FC	ARC	PRC	SYM
FC	Direct	Reversal	Reversal	Reversal
ARC	Direct	Hole Punch	Hole Punch	Hole Punch
PRC	Direct	Hole Punch	Hole Punch	Relaying
SYM	Direct	Hole Punch	Relaying	Relaying

Table 5.1: Traversal methods according to the NAT type

The advantage of this protocol is host can select the best traversal method that increases the efficiency of communication. As an example Relaying is more expensive than the Hole Punching and will use Relaying only in the most restricted NAT types like Port Restricted and Symmetric NAT. More details of the practical implementations and analysis of the results were not provided in this paper.

5.2 Network Address Translator (NAT) Traversal for Offer/Answer Protocols

Interactive Connectivity Establishment (ICE) is a protocol that used for NAT traversal in UDP based multimedia sessions (16). This protocol uses existing Session Traversal Utilities for NAT (STUN) and Traversal Using Relay NAT (TURN) protocols to get the behavior of the applied NAT device. STUN is a UDP based protocol that hosts can use to identify the configuration of their NAT and details of their public socket. The STUN protocol described in RFC 3489 (37). According to the authors the proposed method is mainly designed for the protocols which establish sessions using an offer / answer method like Session Imitation Protocol. According to the protocol description, connections can establish in any type of NAT configurations. But depending on the NAT configuration, it uses a different approach to make a connection. Clients can get the type of their NAT configuration by sending a STUN request to the STUN server located in the public network. In most cases this protocol works fine except the Symmetric NAT. In the Symmetric NAT ICE protocol use relay server to exchange data. Implementation of this protocol is mainly suitable for the devices in the fixed LAN environments.

5.3 Peer-to-Peer Communication Across Network Address Translators

The traversal method always changes according to the NAT configuration applied. The authors of this study (24)proposed different NAT traversal methods that used to establish Peer-to-Peer communication across NAT devices. As an example, when Connection Reversal can use if only one host is behind the NAT device (private network) and the other host located on the public network. The authors evaluated different approaches for the both UDP and TCP protocols with different NAT router hardware from 68 vendors. The test was basically done for the devices connected to the fixed network. The test result shows that 82% success rate for the UDP hole punching and 64% success rate for the TCP hole punching.

5.4 NatTrav a Software Approach for the TCP Hole Punching

Nattrav is a software approach to solve the NAT problems for P2P applications (25). First peers should register with a connection broker which provides current network address for the recipients and facilitate the NAT traversal if peers are behind the NAT. In the registration, peers register with the connection broker by providing the peer's URI. Connection broker uses this URI to send subsequent connection requests from other peers. When a peer device needs to connect with other peer device, it should send the lookup message to the connection broker. The connection broker eventually replies back with the IP address and the port number of the destination peer. Since peers know the other side endpoint details they can establish a connection by sending a SYN packet. If the peers are behind the NAT connection, broker helps the traversal by providing a public endpoint details of the destination peer. In the implementation they used a Java package called "nattrav" which provides classes for setting up TCP sockets. This solution is conceptually the most similar approach to the one proposed by us. However, our approach is specifically targeted for mobile devices, which makes it different from any proposed solutions.

5.5 STUNT

STUNT is an approach which proposed using an external STUNT server and simultaneous NAT traversal technique that is applicable to devices connected to the fixed network (27). According to this approach both end points send an initial SYN with a little TTL value that is just enough to cross their own NATs. This SYN is dropped in the middle of the network before reaching to its destination. End points can learn their initial sequence number by listening to their outbound SYN using PCAP or a RAW socket. Then both endpoints inform this sequence number to the STUN server hosted on the public network. The STUN server spoofs a SYNACK to each host with the properly settled sequence numbers. After that, both hosts send ACK to complete the handshaking process. There are some potential problems which can be seen when implementing this approach. We need to set the lower TTL value in the SYN packet that should be more enough to cross the own NAT but less enough to cross the destination NAT. But it is impossible to set that kind of the TTL value if both devices are located behind the same NAT. To set the TTL value we need to access the OS TCP process using a raw socket. But when considering android devices, creating a raw socket is still complicated and needs a rooted device.

5.6 NATBLASTER

Similarly, in NATBLASTER approach both devices use SYN packet with lower TTL value to initiate the session (26). This SYN packet is dropped in the middle of the network before reaching its destination. Then hosts learn the initial sequence numbers of the SYN packet and send it to the publicly accessible third party server. This server exchanges initial sequence number with the hosts. Instead of using server to forge SYNACK packets here each host forges SYNACK to the other host. After this step each host can establish the TCP session by sending ACK to the peer. Again, The approach ends up with the problem of setting a TTL value and making raw socket which needs root privileges.

6 Future Research Directions

As for the future work the first task will be develop a light weight client application that intelligent enough to identify the most suitable addressing approach according to the configurations of the mobile network. As an example if only one peer located behind the NAT box, connection reversal can use instead of relaying which is more expensive option than others. Same time some port prediction algorithms can included to the client application which may be useful during hole punching in the symmetric type NATs. To achieve this target additional servers should be installed which can provide required details for client application when making selecting the best approach. When considering the server programs used in this experiments were using default searching and sorting mechanisms. But when servicing to the large number of clients, server performances may be degraded. Using better algorithms for searching and maintaining data structures it can optimize the server performances.

Bibliography

[1] ITU, Ict facts and figures : International telecommunication union (may 2014).
URL `http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2014-e.pdf` ix, 1, 2

[2] S. N. Srirama, C. Paniagua, Mobile web service provisioning and discovery in android days, in: Proceedings of the 2013 IEEE Second International Conference on Mobile Services, IEEE Computer Society, 2013, pp. 15–22. 1

[3] S. N. Srirama, M. Jarke, W. Prinz, Mobile web service provisioning, in: AICT-ICIW '06: Advanced Int. Conf. on Telecommunications and Int. Conf. on Internet and Web Applications and Services, IEEE Computer Society, 2006, p. 120. 1

[4] D. Recordon, D. Reed, Openid 2.0: a platform for user-centric identity management, in: Proceedings of the second ACM workshop on Digital identity management, ACM, 2006, pp. 11–16. 2

[5] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, et al., Sip: session initiation protocol, Tech. rep., RFC 3261, Internet Engineering Task Force (2002). 2, 5

[6] P. Saint-Andre, Extensible messaging and presence protocol (xmpp), Internet Engineering Task Force (IETF) Standards Track ISSN: 2070-1721 Extensible Messaging and Presence Protocol (XMPP), March 2011. 2, 9

[7] Zeroconf, Zero configuration networking (zeroconf).
URL `http://www.zeroconf.org/` 2, 9

[8] U. Plug, P. Forum, Universal plug and play forum.
URL `http://www.upnp.org/forum/default.htm` 2, 10

[9] S. N. Srirama, M. Jarke, W. Prinz, Mobile host: A feasibility analysis of mobile web service provisioning., 4th International Workshop on Ubiquitous Mobile Information and Collaboration Systems (UMICS 2006) @ 18th Int. Conference on Advanced Information Systems Engineering (CAiSE'06). 2

[10] S. Cheshire, B. Aboba, E. Guttman, Dynamic configuration of ipv4 link-local addresses, Internet Engineering Task Force (IETF) Request for Comments (RFC) 3927. 9

[11] E. Guttman, Autoconfiguration for ip networking: Enabling local communication, Internet Computing, IEEE 5 (3) (2001) 81–86. 9

[12] J. Postel, Rfc 791: Internet protocol 1981 - september, Defense Advanced Research Projects Agency Information Processing Techniques Office 1400 Wilson Boulevard Arlington, Virginia 22209. 11, 12

[13] S. E. Deering, Internet protocol, version 6 (ipv6) specification, DRAFT STANDARD, Errata Exist, Standards Track December 1998. 12

[14] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. de Groot, E. Lear, Rfc 1918: Address allocation for private internets. february 1996 best current practice 8. 12

[15] P. Srisuresh, M. Holdrege, Ip network address translator (nat) terminology and considerations, Informational Lucent Technologies August 1999 IP Network Address Translator (NAT). 14

[16] B. Sterman, D. Schwartz, Nat traversal in sip, IEC Annual Review of Communications 56 (2002). 56. 15, 54

[17] J. Postel, Rfc 793: Transmission control protocol, september 1981, Status: Standard 88. 19

[18] M. Cotton, L. Eggert, J. Touch, M. Westerlund, S. cheshire," internet assigned numbers authority (iana) procedures for the management of the service name and transport protocol port number registry, Tech. rep., BCP 165, RFC 6335, August (2011). 19

[19] J. Postel, User datagram protocol, INTERNET STANDARD RFC 768 ,ISI 28 August 1980. 21

[20] I. global standard for international mobile telecommunications. [link].
URL `www.imt-2000.org` 21

[21] K. Richardson, Umts overview, Electronics & Communication Engineering Journal 12 (3) (2000) 93–100. 21

[22] M. Holdrege, P. Srisuresh, Protocol complications with the ip network address translator rfc 3027, january (2001). 22

[23] P. Srisuresh, B. Ford, D. Kegel, State of peer-to-peer (p2p) communication across network address translators (nats), Internet Engineering Task ForceRequest for Comments 5128 (2008) 1–32. 22

[24] B. Ford, P. Srisuresh, D. Kegel, Peer-to-peer communication across network address translators., in: USENIX Annual Technical Conference, General Track, 2005, pp. 179–192. 22, 25, 54

[25] J. L. Eppinger, Tcp connections for p2p apps: A software approach to solving the nat problem, Institute for Software Research (2005) 16. 22, 40, 55

[26] A. Biggadike, D. Ferullo, G. Wilson, A. Perrig, Natblaster: Establishing tcp connections between hosts behind nats, in: ACM SIGCOMM Asia Workshop, Vol. 5, 2005. 25, 40, 56

[27] S. Guha, Y. Takeda, P. Francis, Nutss: A sip-based approach to udp and tcp network connectivity, in: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture, ACM, 2004, pp. 43–48. 25, 55

[28] Tele2. [link].
URL `http://www.tele2.ee/index.html` 25

[29] Elisa. [link].
URL https://www.elisa.ee/ 25

[30] EMT. [link].
URL https://www.emt.ee/en/ 25

[31] Super. [link].
URL https://www.super.ee/ 25

[32] L. Madsen, J. Van Meggelen, R. Bryant, Asterisk: the definitive guide, O'Reilly Media, Inc., 2011. 28

[33] ImsDroid, D. telecom, "sip/ims client for android," 2011,.
URL http://code.google.com/p/imsdroid/ 29

[34] S. Guha, P. Francis, Characterization and measurement of tcp traversal through nats and firewalls, in: Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement, USENIX Association, 2005, pp. 18–18. 40

[35] K. Reinloo, Addressing smartphones located behind firewalls, Bachelor's thesis, Institute of computer science,University of Tartu (2013). 44

[36] A. Wacker, G. Schiele, S. Holzapfel, T. Weis, A nat traversal mechanism for peer-to-peer networks., in: Peer-to-Peer Computing, 2008, pp. 81–83. 53

[37] J. Rosenberg, H. Weinberger, R. Mahy, C. Huitema, Rfc 3489, STUN, IETF, March 2003. 54

Non-exclusive license to reproduce thesis