museU: A MEDIA STREAMING APPLICATION

by

Sharanyo Chatterjee, Subham Bhattacharjee and Dipan Ganguly

Under guidance of

Prof. Avishek Barman,

Assistant Professor

(Dept. of Computer Science)

A thesis submitted to the graduate faculty

In partial fulfillment of the requirements for the degree of

BACHELOR OF SCIENCE

Honours: Computer Science

Batch of 2015-2018

RAMAKRISHNA MISSION VIDYAMANDIRA

(Belur Math, Howrah)

Acknowledgements

We would like to place on record my deep sense of gratitude to **Asst. Prof. Avishek Barman, HOD of Dept. of Computer Science and Electronics,** Ramakrishna Mission Vidyamandira, Belur Math, Howrah-711202 for his generous guidance, help and useful suggestions.

We are extremely thankful to **Swami Shashtrajnananda, Principal,** Ramakrishna Mission Vidyamandira, Belur Math, Howrah-711202 for providing us infrastructural facilities to work in, without which this work would not have been possible.

We would like to thank our classmates and friends for their continuous moral support.

Place:- Belur Math, Howrah

Date:-

Sharanyo Chatterjee
Reg. No: - A04-1112-0148-15

Subham Bhattacharjee
Reg. No: -A04-1112-0143-15

Dipan Ganguly
Reg. No:-A04-1112-0152-15

CERTIFICATE

I hereby certify that the work which is being presented in the B.Sc Computer Science report entitled "**museU**", in practical fulfillment of the requirements for the award of the Bachelor of the Computer Science and submitted to the Department of Computer Science of Ramakrishna Mission Vidyamandira, Belur Math, Howrah is an authentic record of our own work carried out during a period from January 2017 to April 2017 (6th Semester) under the supervision of Abhishek Barman, Professor, Computer Science Department.

The matter presented in this Project report has not been submitted by me for award of any other degree elsewhere.

Place:- Belur Math, Howrah

Date:-

Sharanyo Chatterjee
Reg. No: - A04-1112-0148-15

Subham Bhattacharjee
Reg. No: - A04-1112-0143-15

Dipan Ganguly
Reg. No:- A04-1112-0152-15

This is to certify that the above statement made by students is correct to the best of my knowledge.

Place: -Belur Math, Howrah
Date: -

Signature of the Supervisor
Avishek Barman

Signature of HOD
Avhishek Barman

Signature of the External Examiner

Abstract

"museU" is a multiplatform media streaming application based on peer to peer communication. The main idea behind it was to save storage space of our mobile devices, reduce piracy, customer privacy, and of course to share media files. Unlike other messaging applications or file sharing applications it streams the file to you so you don't have to be afraid of your flooded SD cards anymore. The moment you are downloading a copyrighted media you are violating the norms and you are indulging in piracy, but in case of streaming you do not indulge in any kind of piracy. The big messaging apps these days all are concerned about security, but are WE? Even if we are we will have to trust the administrator of that app as they can see whatever they want!! But can they truly be trusted? We bring a solution to this by exchanging the data from user to user or peer to peer, no server in between. So, we do not know what you share.

Table of Contents

List of Figures

Chapter 1

Introduction:

Objectives

- Introduction to the goals of the project and justification of why it is important.

1.1 Project Overview:

Our report entitled "museU: A media streaming application" describes the features of our network-based application "museU" that shares files in peer to peer network environment. Some of the unique features of our application are data security, storage friendly, piracy free etc. This application only uses a well-known server to forward the IP addresses of two of the users to each other. Then those two users exchange the files between them without any interference of the server. Someone can watch the movies that are basically in his pc way back at his/her house or can listen to his friend's play list. But then of course both of them has to be online at that moment. Even if his/her storage space is not enough to watch that movie he can watch it by just streaming it through our application.

1.2 Objectives:

We focused on a few topics while designing our application: -

1. Piracy control.
2. Storage friendly.
3. Data security for users.
4. Reduce the burden of a huge server.

Chapter 2

System Analysis

Objectives

- Why we need a software like this.
- Drawbacks of existing system.
- What made us think of a project like this.
- Feasibility studies.

2.1 Identification of need:

2.1.1 Existing Systems:

The few close systems that already exists are messaging applications like whatsapp hike, line etc., or the end to end communication system like telnet. All these messaging apps can share media files and obviously messaging. The telnet can communicate between two end users of network. Both of these have some drawbacks.

Drawbacks:

- All the messaging applications that are already in the market share their media files by storing them into local machines, so your storage space is flooded.
- These applications don't check for copyrights of the media before sharing so piracy is inevitable.
- Always there is an upper limit of the size of the file that can be shared.
- The application admin receives all the data he/she requires of client and can sell it to anyone.
- And at last but not the least the admin doesn't have to carry the burden of a huge server storing all the data of all the clients.

2.1.2 Primary Investigation:

Primarily we over viewed all the existing applications to judge our application's need in the market. To do that we discussed about our future plan about this application with random people from various backgrounds and received their feedback. We present here a summary of those feed backs.

Feedbacks:

- Most of the people were having the above-mentioned problems (Drawbacks of existing systems) and they agreed that our application could help them solving those problems.

- A bunch of people suggested that they are happy with their application and with them they are well connected to their virtual world and won’t go for a new application.
- Some people suggested that they can just use a telnet service to do what our application does, so they do not need a new application.
- And some people thought it is absurd that the other user will be on line 24X7.

2.2 Motivation:

After all these Investigation and requirement analysis we went through a detailed study of the needs of people. What we ended up concluding was the number of people who admitted that they are having the problem is very large. Many of them and others though said that moving to a new application means another blockage to memory but they also admitted that the amount of storage space that will be saved is huge. Those who didn’t want to go for a new application leaving the old one also said that they could use those features of ours in their chat applications, and the streaming content also was very amusing to them. After last month of Mark Zuckerberg’s confession about data stealing also worried them a bit. And those who said that they would rather use a telnet service, well there is a lot of drawback of using a service like telnet like, in telnet you have to log in to other’s device to access the files, so you’ll have known his user name and password, you cannot categorize which people gets what, and telnet uses text stream to communicate which is very bad if you want a secured data transfer.

A lot of people to whom I presented the fact that they can move with a mobile device and watch the movies he have in his home PC they were much interested than sharing media files to others. As it is impossible to predict which movie they’d want to watch while going on a trip or at recess times at their respective work places. And not to mention the space crunch we face to fit a single HD movie in our mobile devices.

So, after a fruitful debate on this topic we came to this conclusion that our product should be accepted by majority of the people. But they all suggested that it may happen so that the less number of users may result in people's disinterest of using it. Here we face a great paradox if someone doesn't find their friends using the application they have no one from stream to they can only stream from his other devices, from the other side if they (the person's friend) doesn't find that person using that application then they won't use the application either. This a work of post-production and promotion, but here as most of the people ultimately said the features of this application is kind of useful one way or other we got our motivation to work on the above project.

2.3 Feasibility Study:

This section describes the judgement to how much our project is possible in accordance to certain circumstances process is to make changes in the current in the current system in order to achieve new effective system. The feasibility study includes complete initial analysis of all related system. The types of feasibility study we have undergone through and our project is.

2.3.1 Technical Feasibility:

We considered our resources at our laboratory and decided to construct the core structure in java, so that we can easily move to android studio any time as it is undoubtedly the dominating mobile OS. For server-side programming we used PHP for well-known server implementation. We received full access to our departmental router from our departmental HOD to implement port forwarding.

2.3.2 Schedule Feasibility:

We analyzed if the target we have can be completed within the given time limit. And after calculating our man power and lab hours we decided that it was feasible.

2.3.3 Economic Feasibility:

We barely needed any extra hardware support for the project of ours. Our computers and mobile devices allowed us to do all the coding and testing. The expense we had to carry was the cost of internet connection of different service providers for testing. The server we hosted needs to be a dedicated server so that it can become a well known one to all the clients at the same time. We couldn’t have afforded that kind of a server so we just hosted a server in one of our computers and set the I.P. manually to the clients. The rest of the work was feasible and cheap in terms of money.

2.3.4 Cultural Feasibility:

This project is culturally feasible as it helps people in their daily lives by releasing storage space, frees hazards of copying movies in mobile devices.

2.3.5 Legal Feasibility:

This project doesn’t violate any laws or regulation and discourages piracy.

2.3.6 Resource Feasibility:

Software resource required for this projects like java, Android Studio, Xampp etc. and hardware devices like computers, Androids were easily obtainable. Ans man power required to do this project was also efficiently managed to complete the work.

Chapter 3

System Design

Objectives

- Basic structures and designs of our software.
- Diagrams of functions that are present in the system.

3.1 Data Flow Diagrams:

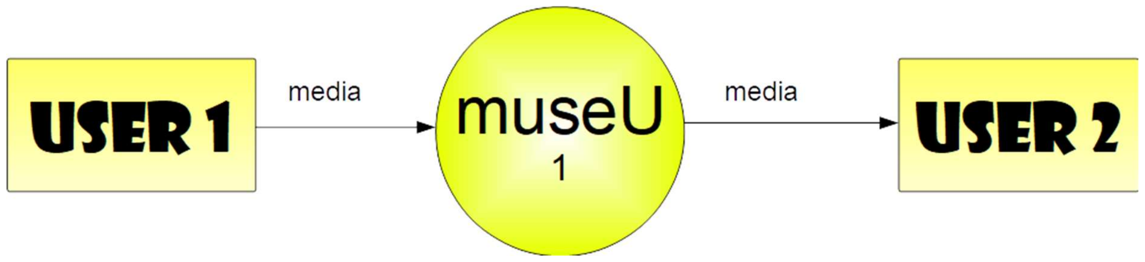


Figure3.1: Context Level DFD

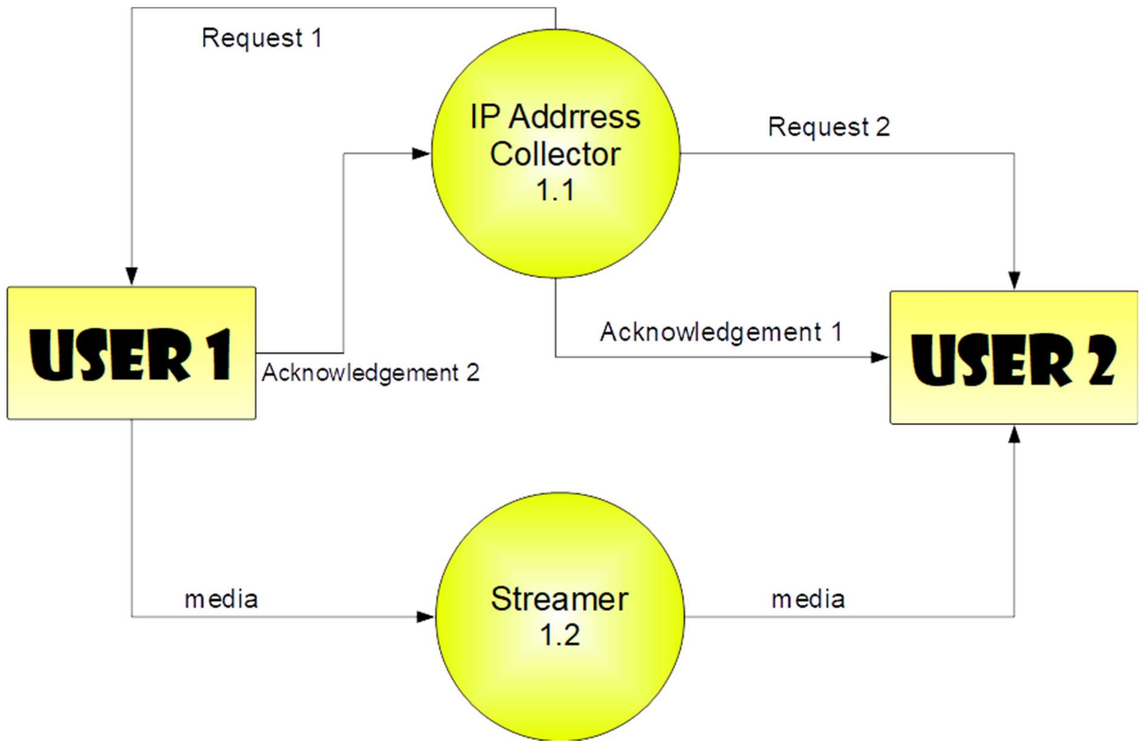


Figure3.2: Level 1 DFD

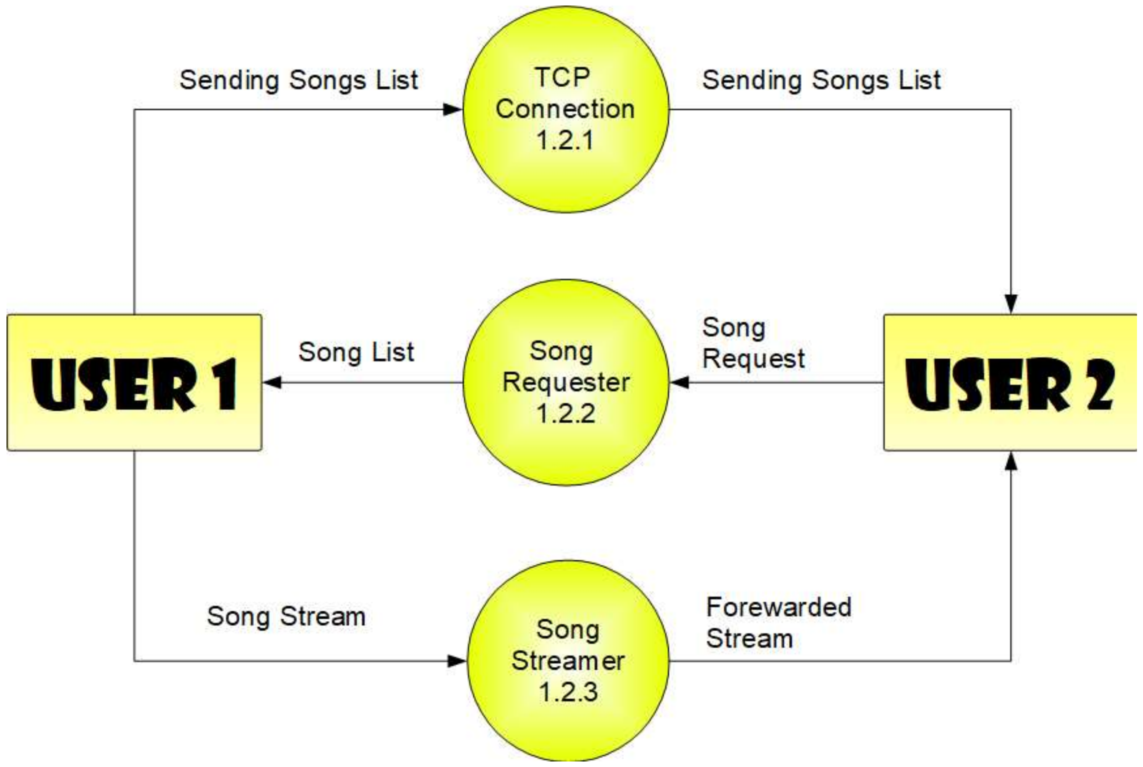


Figure3.3: Level2 DFD

3.1.1 Data Dictionary:

Media: The media that is going to be streamed

Request 1: User 1 request for IP address and port of the server.

Request 2: server request for IP address and port of the User2.
Acknowledgement1: User2 responds with its IP address and port to the server.
Acknowledgement1: Server forwards the IP address and port of User2 to the User1.
Audio: Audio is being streamed.
Song list: List of songs available for streaming

Stream: Data stream that communicates between two clients.

3.2 Use Case Diagram:

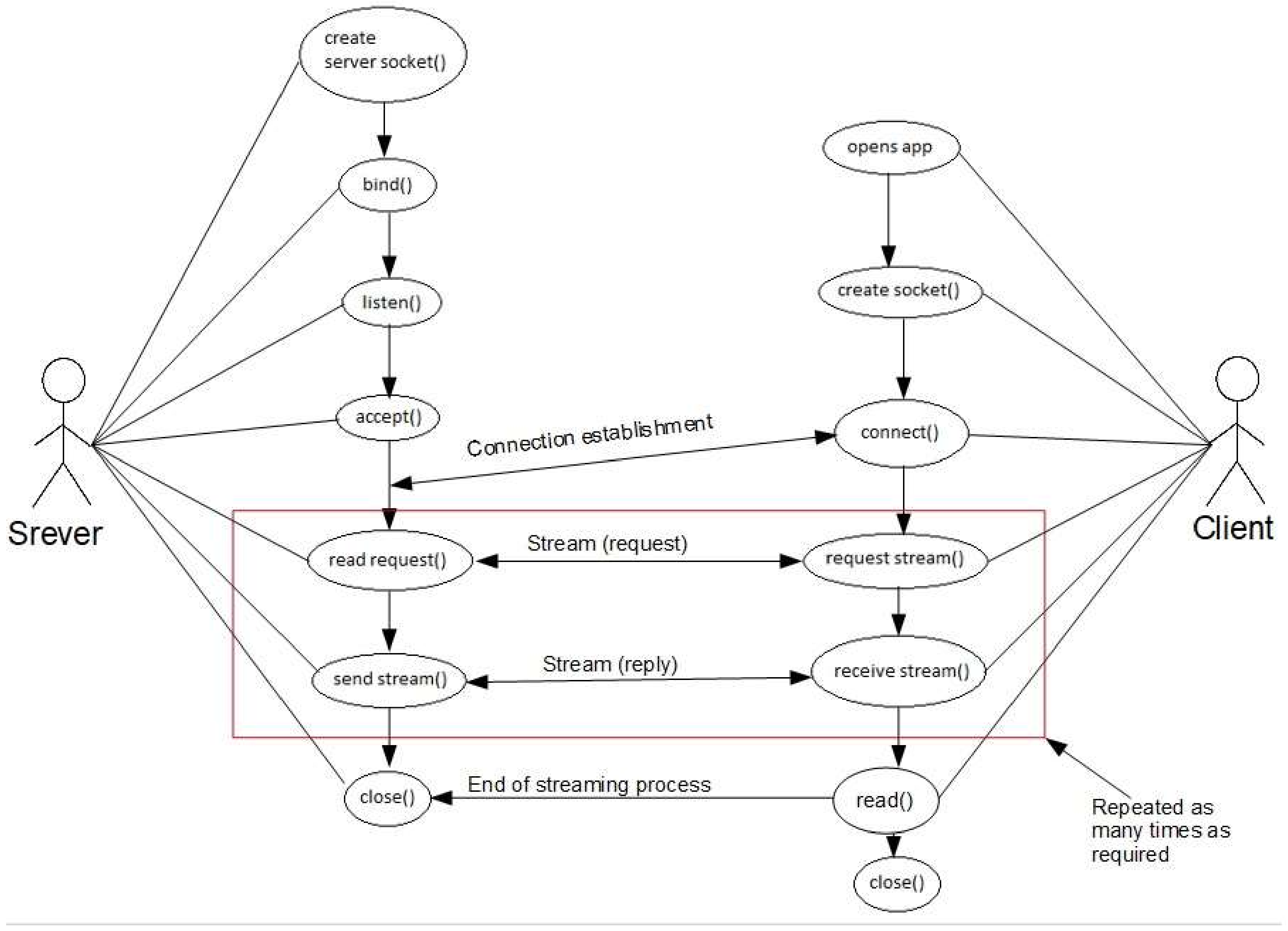


Figure 3.4: Use Case Diagram

3.3 Sequence Diagram:

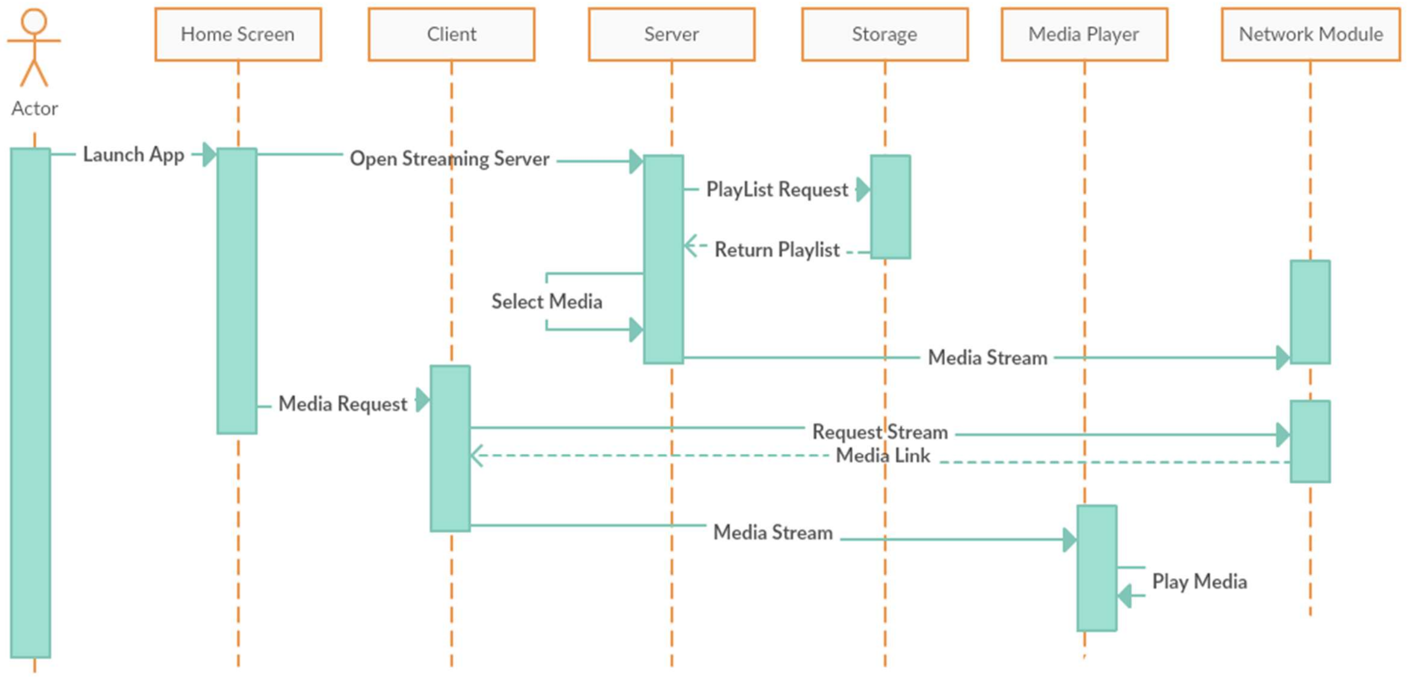
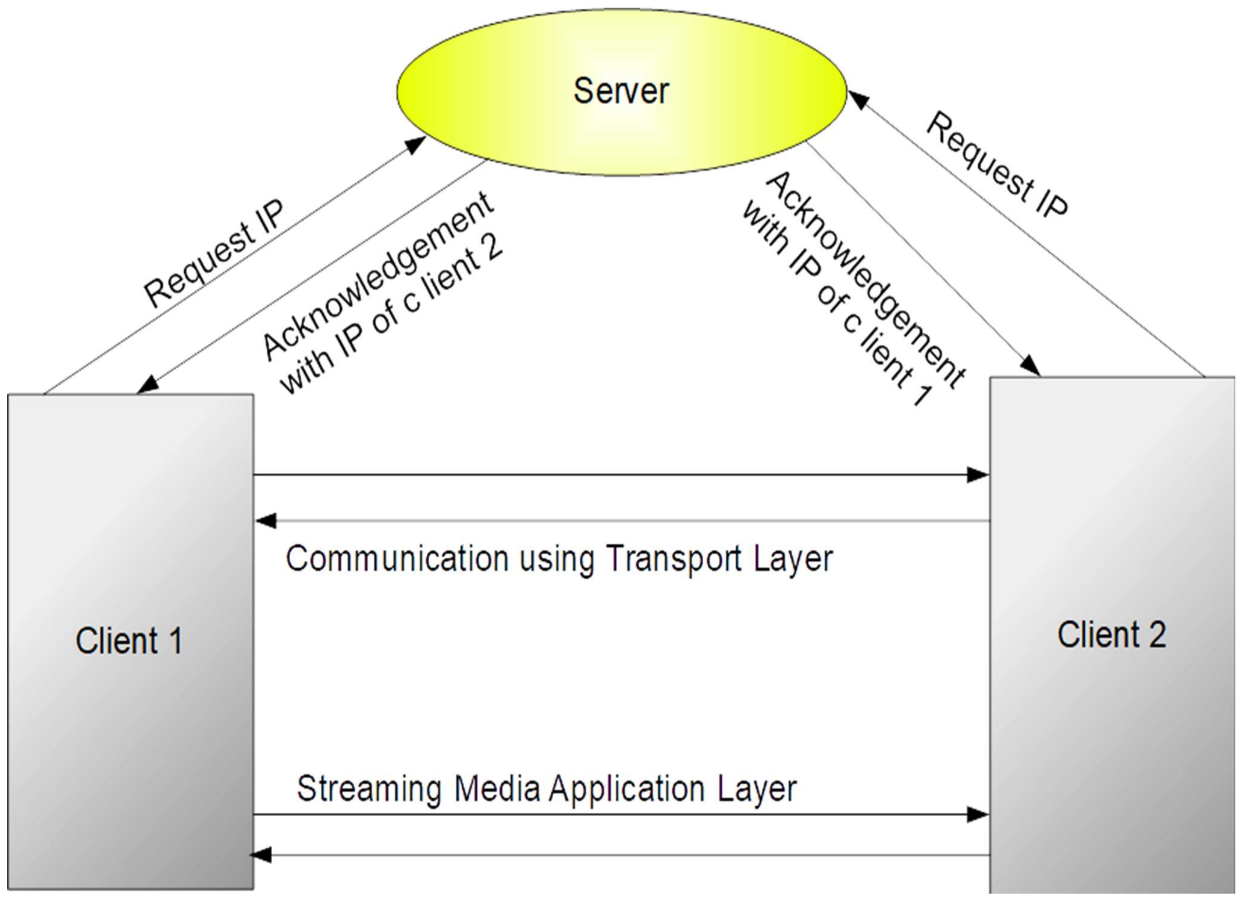


FIGURE3.5: SEQUENCE DIAGRAM

3.4 Block Diagram:



Server
Request IP
Acknowledgement with IP of c lient 2
Acknowledgement with IP of c lient 1
Request IP
Client 1
Client 2
Communication using Transport Layer
Streaming Media Application Layer

FIGURE 3.6: BLOCK DIAGRAM

Chapter 4

Implementation

Objectives

- Peer-to-Peer network, how does it work and what are the drawbacks.
- Well-known servers or rendezvous servers, and possible implementation techniques.
- Streaming tools can be used and has been used.

In the last chapter we discussed about our plan of designing the software from client and administrator side. Now we will talk mainly about how we gradually developed the application, what problems we faced developing the software, the tools we used to design, develop and debug etc. And what are the different modules we worked on.

Modules:

- Peer-to-Peer network
- Rendezvous server
- Live media streaming

4.1 Peer-to-Peer network:

A network is called peer-to-peer network when two or more computers are connected to each other in a robust form and communicate with each other without going via any server in the middle. It can be some dedicated wired or wireless connection between some computers who don't need any server computer in between to communicate among each other. Since last few decades number of devices connected to internet has increased massively. So, IPv4 was no longer enough to identify each device uniquely on the network. This is where the concept of NAT (Network Address Translator) comes into scene.

4.1.1 Network Address Translator(NAT):

Network address translation (NAT) is a method of remapping one IP address space into another by modifying network address information in IP header of packets while they are in transit across a traffic routing device. The technique was originally used as a shortcut to avoid the need to readdress every host when a network was moved. It has become a popular and essential tool in conserving global address space in the face of IPv4 address exhaustion. One Internet-routable IP address of a NAT gateway can be used for an entire private network.

There are several ways of implementing network address and port translation. In some application protocols that use IP address information, the application running on a node in the masqueraded network needs to determine the external address of the NAT, i.e., the address that its communication peers detect, and, furthermore, often needs to examine and categorize the type of mapping in use. Usually this is done because it is desired to set up a direct communications path (either to save the cost of taking the data via a server or to improve performance) between two clients both of which are behind separate NATs. As we can see in figure:1 different levels of NAT's are present. Nodes on private networks can connect to other nodes which are present in the same private network or it can connect to well-known nodes in the global network. These can be accomplished by casually opening TCP or UDP connections. For outgoing connection NAT allocates a temporary public endpoint. But it usually blocks all the incoming connections.

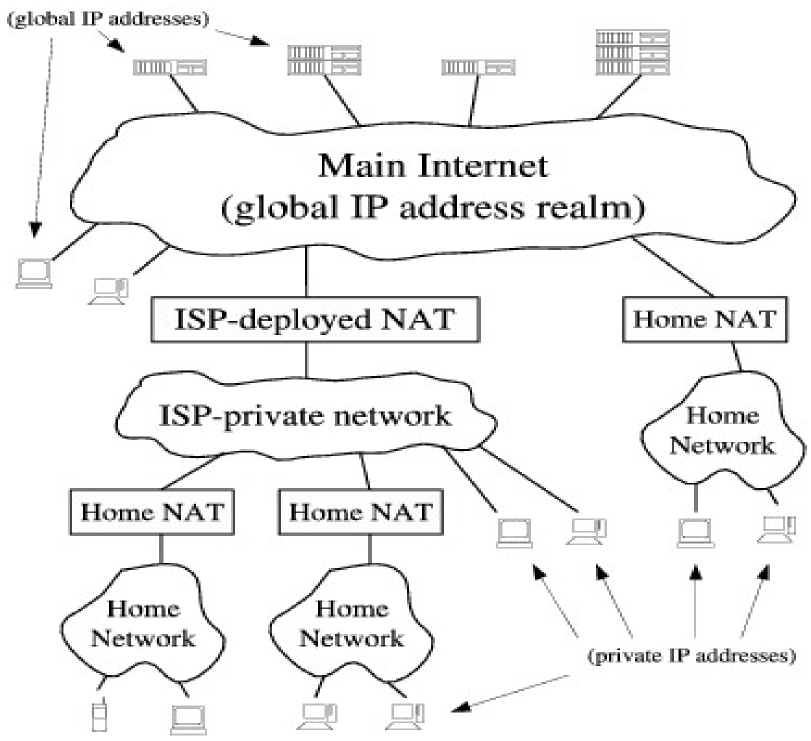


FIGURE4.1:PUBLIC AND PRIVATE IP ADDRESS DOMAINS

For this purpose, the Simple traversal of UDP over NATs (STUN) protocol was developed (RFC 3489, March 2003). It classified NAT implementation as full-cone NAT, (address) restricted-cone NAT, port-restricted cone NAT or symmetric NAT and proposed a methodology for testing a device accordingly.

➢ Full-cone NAT

Full-cone NAT, also known as *one-to-one NAT*

- Once an internal address (iAddr:iPort) is mapped to an external address (eAddr:ePort), any packets from iAddr:iPort are sent through eAddr:ePort.
- *Any external host* can send packets to iAddr:iPort by sending packets to eAddr:ePort.

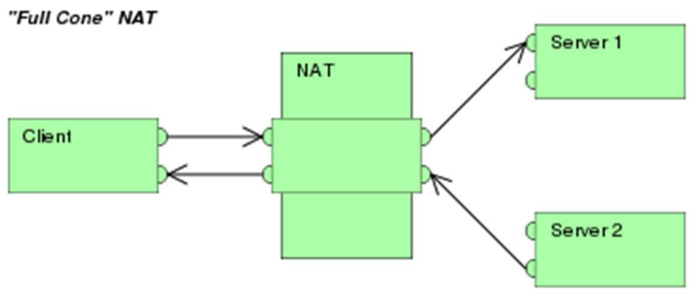


FIGURE 4.2:FULL CONE NAT

➢ (Address)-restricted-cone NAT

- Once an internal address (iAddr:iPort) is mapped to an external address (eAddr:ePort), any packets from iAddr:iPort are sent through eAddr:ePort.
- An external host (*hAddr:any*) can send packets to iAddr:iPort by sending packets to eAddr:ePort only if iAddr:iPort has previously

sent a packet to hAddr:*any*. "Any" means the port number doesn't matter.

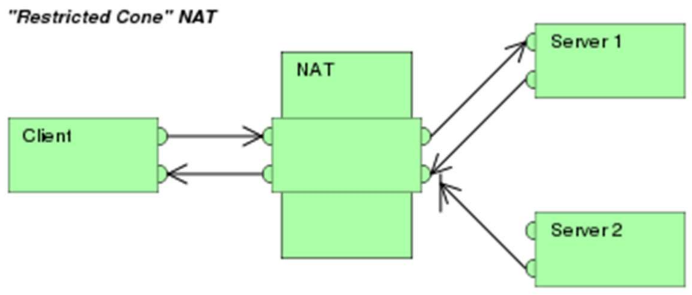


FIGURE 4.3:ADDRESS RESTRICTED CONE NAT

- Port-restricted cone NAT

Like an address restricted cone NAT, but the restriction includes port numbers

  - Once an internal address (iAddr:iPort) is mapped to an external address (eAddr:ePort), any packets from iAddr:iPort are sent through eAddr:ePort.
  - An external host (*hAddr:hPort*) can send packets to iAddr:iPort by sending packets to eAddr:ePort only if iAddr:iPort has previously sent a packet to hAddr:hPort.

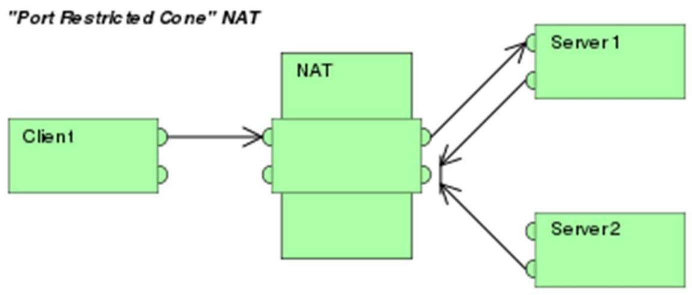


FIGURE4.4:PORT RESTRICTED CONE NAT

- Symmetric NAT
  - Each request from the same internal IP address and port to a specific destination IP address and port is mapped to a unique external source IP address and port; if the same internal host sends a packet even with the same source address and port but to a different destination, a different mapping is used.
  - Only an external host that receives a packet from an internal host can send a packet back.

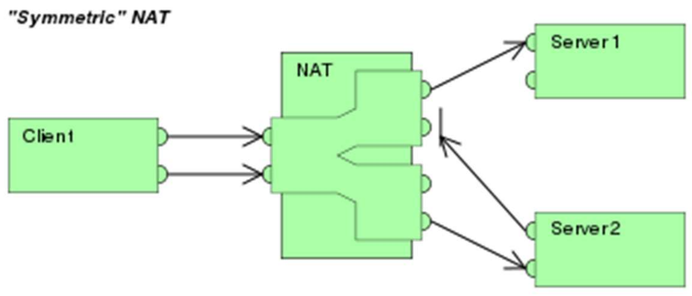


FIGURE 4.5:SYMMETRIC NAT

4.1.2 Communication Across Network Address Translators:

NAT indeed is a breakthrough of the problem that we were facing before with the inadequate no. of IPv4 addresses in the Internet today. But in terms of Peer-to-Peer communication it brings one of the biggest challenge. Two computers behind same NAT have same global IP, how we can uniquely identify these devices? When we say NAT, we can think of a router routing a number of computers. But the main question arises where the router would forward the packets it receives from the Internet at its external end? A few trivial solutions to this are –

4.1.2.1 Port Forwarding:

1. DMZ: DMZ stands for Demilitarized Zone, a lot of router now a day supports this feature. DMZ in routers is called FAUX-DMX as they fail to provide all the features of DMZ. What it does is the simplest kind of networking. It simply forwards all the incoming requests to a single node or computer. But this has a major problem when you want to setup two or more devices it is of no good.
2. Port forwarding: All the networks have a "port". It basically identifies where (specific process on a device) the packet is to be sent. As the device may have a number of process running it helps to direct the packet to right process. The concept of port forwarding is when you have access to your router you tell your router that any packet sent on this port has to be forwarded to this specific node. And any packet that is received by that router or NAT ion general will be forwarded to the device specified.
3. UPnP: UPnP or Universal Plug n Play works the same way as port forwarding but instead of setting the router manually by the user the software on a device can set up the forwarding it needs and can receive the packets received by the router on that port.

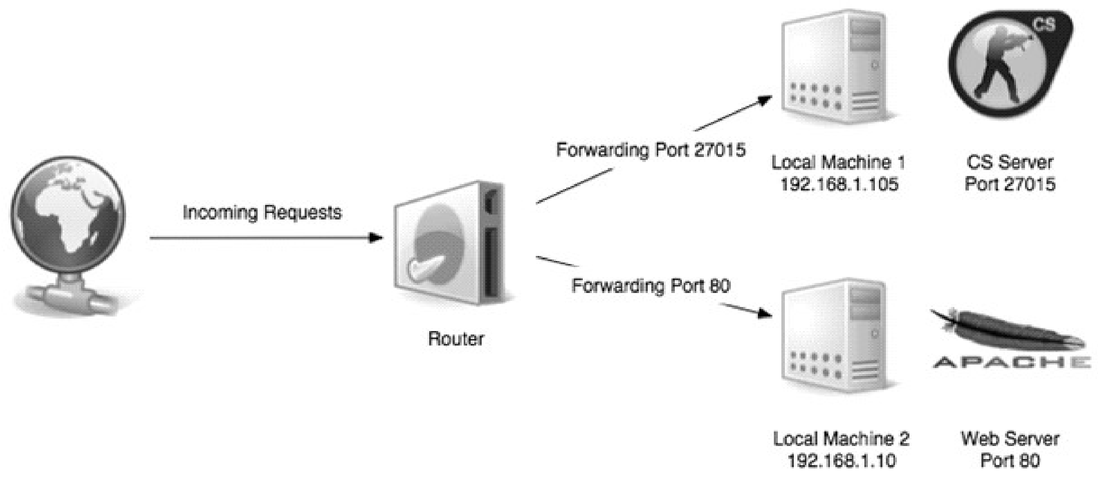


FIGURE 4.6: PORT FORWARDING

➢ Problems of Port Forwarding*:*

All of the above can ease the problems of NAT but there still remains some crucial problems. To forward a port first of all you need the router’s full access i.e. you must be the admin of the respective NAT. In many cases we consider mobile devices, for a mobile device there can be no router so there can be no port forwarded. The occasion UPnP however solve the router access problem granted that its on by default but it also has issues. UPnP is not supported still in many of routers, and if you give all the software to read packets from the internet as the will you have already lost the privacy to that software or malware may be. The FBI (Federal Bureau of Investigation America) at 2001 strongly recommended everybody to disable their router’s Plug and Play option for security purpose.

4.1.2.2 Relaying:

The most reliable but the least efficient method of P2P communication across NAT. The communication is presented to the network as if its performing a normal client/server communication, through relaying. Two clients say A and B initiates a TCP or UDP connection with the well-known server say S. A and B both contacts server S at its global IP address 18.181.0.31 in this case and port 1234. The clients reside in different private network and their respective NATs keep them separated from sending message directly. Instead of sending message directly, the two clients simply use the server S to exchange message.

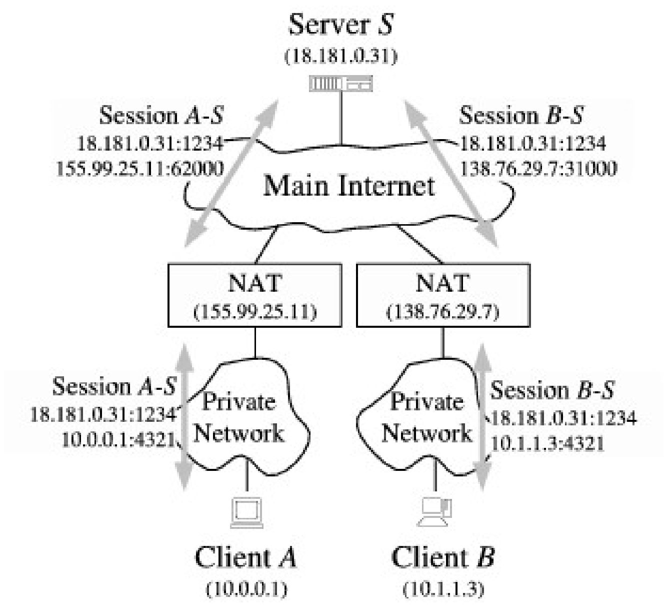


FIGURE 4.7 : RELAYING

Relaying works just fine as long as two the clients are connected to internet. Its main disadvantages are it consumes the server's processing power and network bandwidth, and the communication delay is significant even if the connection is well established. Nevertheless, as there is no other reliable method to communicate between two nodes, it still is used where maximum robustness is our primary obsession.

Most of the messaging applications use this method to communicate. They keep the messages or the files stored unless it is received by the intended receiver. This is why they require a massive server to keep all the user data.

4.1.2.3 Connection Reversal:

Some P2P connection uses a simple method called connection reversal. To communicate both the nodes have connections to the well-known rendezvous server S and only one of the peer is behind NAT as shown in the figure:8. If A wants to initiate a connection to B, then a direct connection attempt works automatically as B is not behind any NAT.A's NAT simply interprets the connection as an outgoing connection to a well-known address B. But in the case where B wants to initiate a connection to A it is not that simple, as any attempt to connect to A will be blocked by the NAT of A. B instead relays a connection request through the rendezvous server S, asking A to attempt a “REVERSE” connection to B.

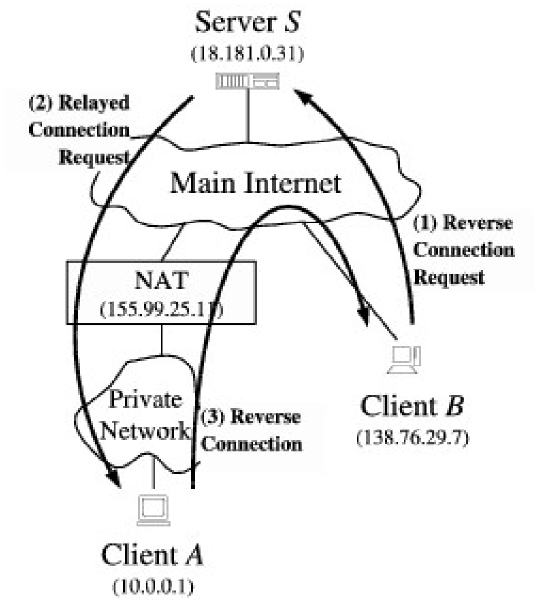


FIGURE4.8: CONNECTION REVERSAL

- It has some obvious limitation like at least one of the Client or node has to have a well-known address or connected directly to the Internet without any NAT in between.
- All the above-mentioned systems fail to deliver what is required for our system.
-

4.1.3 Hole Punching:

Hole Punching is a technique to connect two parties with each other directly in which one or both are behind a firewalls or routers that use Network Address Translation. To punch a hole both of the Nodes connect to a well-known server or rendezvous server which stores the internal and external address and the port of the incoming connections. Then it simply relays those information to the opposite nodes. Then the two of the nodes try to connect with each other respectively.

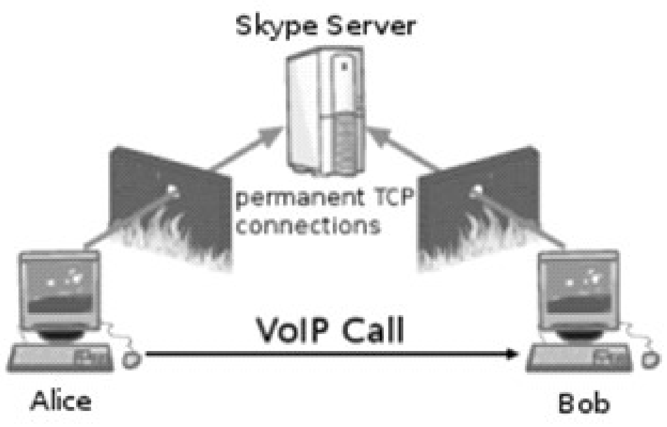


FIGURE 4.9: RENDEZVOUS SERVER

Let's assume that Alice wants to call her friend Bob. Alice contacts it’s server and tells sends its requirements to the rendezvous server. The server already knows about Alice’s details. From the incoming query it sees that Alice is currently registered at the IP address 1.1.1.1 and she always contacts the server or other client on a specific port say 1414. The Skype server passes this information on to Bob, which, according to its database, is currently registered at the IP address 2.2.2.2 and which, by preference uses port 2828.

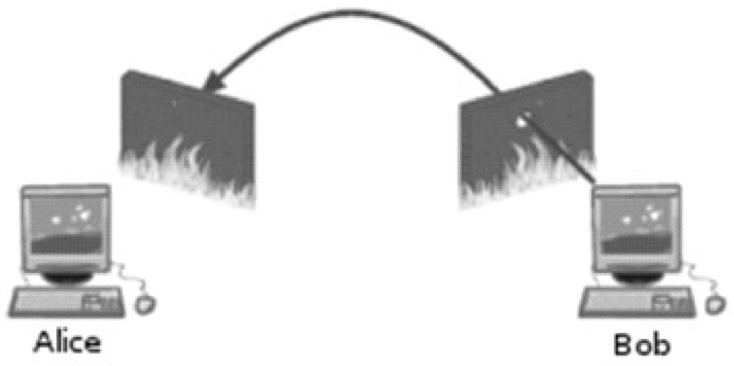


FIGURE 4.10: BOB PUNCHES HOLE

Bob then punches a hole in its own network firewall: It sends a packet to 1.1.1.1 port 1414. This is discarded by Alice's firewall, but Bob's firewall doesn't know that. It now thinks that anything which comes from 1.1.1.1 port 1414 and is addressed to Bob's IP address 2.2.2.2 and port 2828 is legitimate - it must be the response to the query which has just been sent.

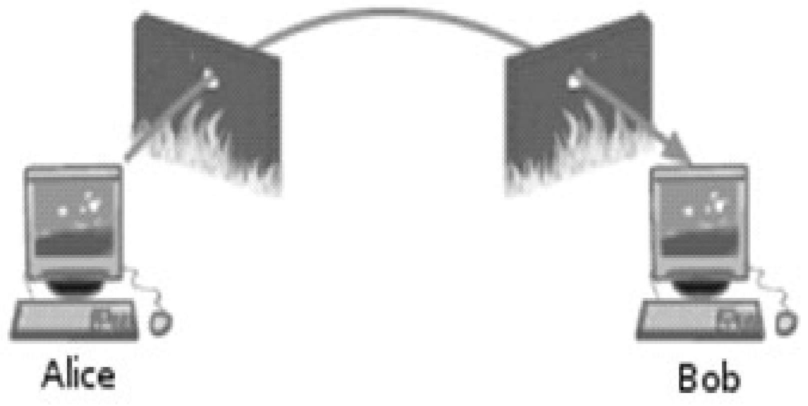


FIGURE 4.11: ALICE PUNCHES BACK

Now the server passes Bob's coordinates on to Alice, who attempts to contact Bob at 2.2.2.2:2828. Bob's firewall sees the recognized sender address and passes the apparent response on to Bob's PC - and his device receives some data.

4.1.4 UDP Hole Punching:

UDP hole punching allows two clients to set up a UDP connection between them directly. For hole punching mechanism we first need to know what rendezvous server is.

4.1.4.1 Rendezvous Server:

Rendezvous server is a well-known server to whom any device can reach out any time. Hole punching presumes that two clients say A and B, already have an active UDP session with the rendezvous server say S. Then the server S records two endpoint for that connection, firstly the endpoint on which server observes the incoming connection (IP address, UDP port), secondly the endpoint it believes it is in (IP address, UDP port). We refer the first pair as client’s public endpoint and the later one as private endpoint. If both of the IP, port pair is same then the node is not behind any kind of NAT. Rendezvous server is described further in this chapter later in section 4.2.

4.1.4.2 Establishing Connection:

Suppose A wants to establish an UDP connection B, the process goes as follows:

1. Initially A has no information how to reach B, so it reaches for the rendezvous server S.
2. S sends A the private and the public endpoints of B
3. At the same time S sets up an UDP connection with B and sends the endpoints of A.
4. When A received the B's endpoints A starts to send UDP packets to both of the endpoints of B and when it gets success sending packets locks on that one. B does the same for A locking the first end point that work. Messages are not critical as long as they are asynchronous.

There are three kinds of UDP hole punching we need to consider:

4.1.4.3 Peers behind a common NAT:

First, we consider the situation where both of the peer happens to reside behind same NAT and their fore client A establishes a UDP connection with server S. The common NAT assigns A with a public port 62000. The same works for B, and the NAT assigns it a port say 62005.

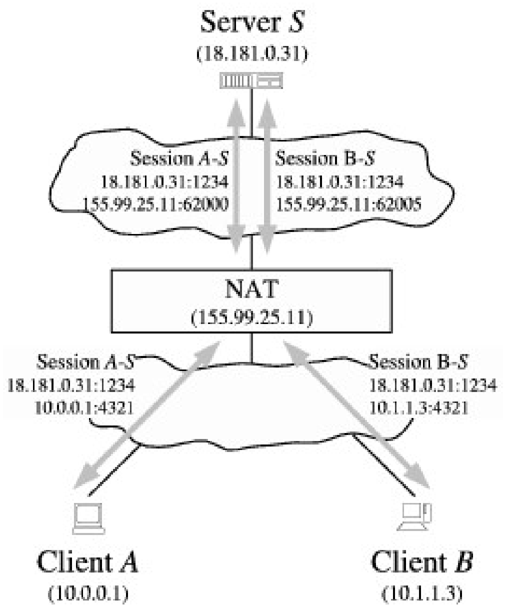


FIGURE4.12: CONNECTION TO RENDEZVOUS SERVER

Now, the server S sends the two end points of the opposite clients as described above. For this case the two end points should be the same. So, both of the clients try to send the datagrams directly to each of the end points. As they are inside same NAT the datagram should reach its destination.

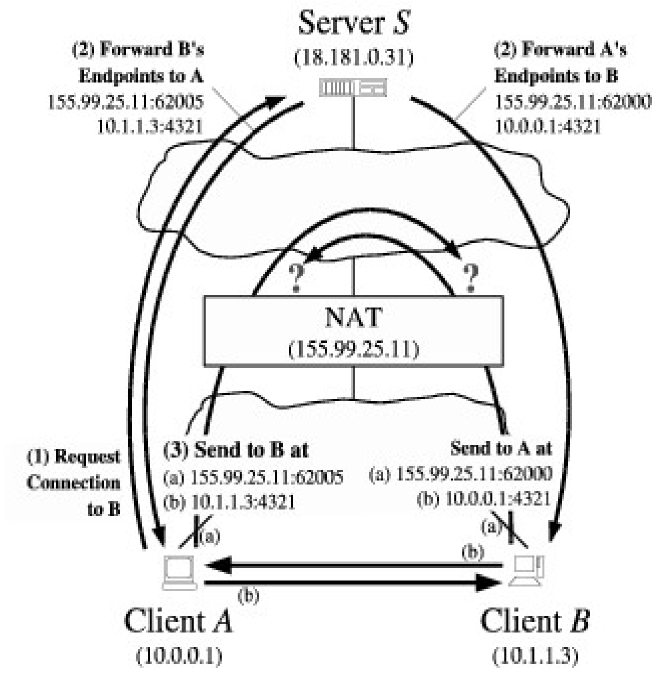


FIGURE 4.13: SENDING UDP DATAGRAMS

This is as simple as writing a UDP programming in local host. Only thing that is to be done is we have to maintain a rendezvous server rest of the part is trivial. The local port they used to communicate with that server is used here to open a Datagram Server and Datagrams are sent to that port by the other client and received by that client.

4.1.4.4 Peers behind different NATs:

Suppose clients A and B have private IP addresses behind different NATs. A and B have each initiated UDP communication sessions from their local port 4321 to port 1234 on server S. In handling these outbound sessions, NAT A has assigned port 62000 at its own public IP address, 155.99.25.11, for the use of A's session with S, and NAT B has assigned port 31000 at its IP address, 138.76.29.7, to B's session with S.

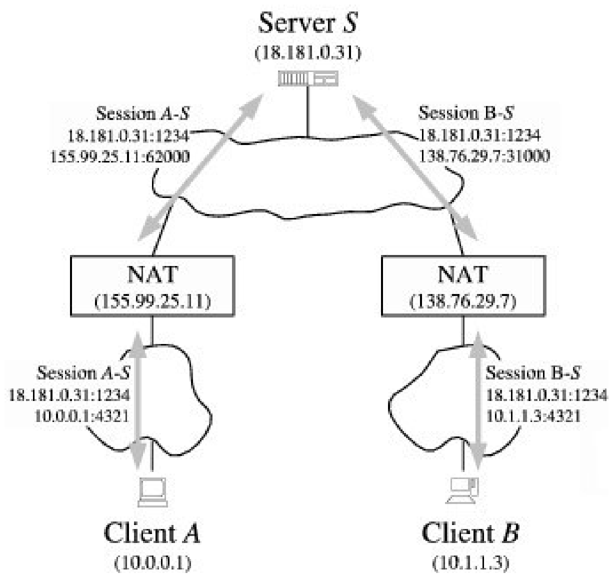


FIGURE4.14: BEFORE HOLE PUNCHING

In A’s registration message to S, A report it’s private endpoints to S a 10.0.0.1:4321, where 10.0.0.1 is A’s IP address on its own private network. S records A's private endpoint along with A's public endpoint as observed by S itself. A's public endpoint in this case is 155.99.25.11:6200, the temporary endpoints assigned to the session by the NAT. Similarly, when client B communicates with S, it records B's private endpoint from B's point of view and records the public endpoint of B by observing its occurrence. In this case B's private endpoint is 10.1.1.3:4321 and B's public endpoint is 138.76.29.7:31000.

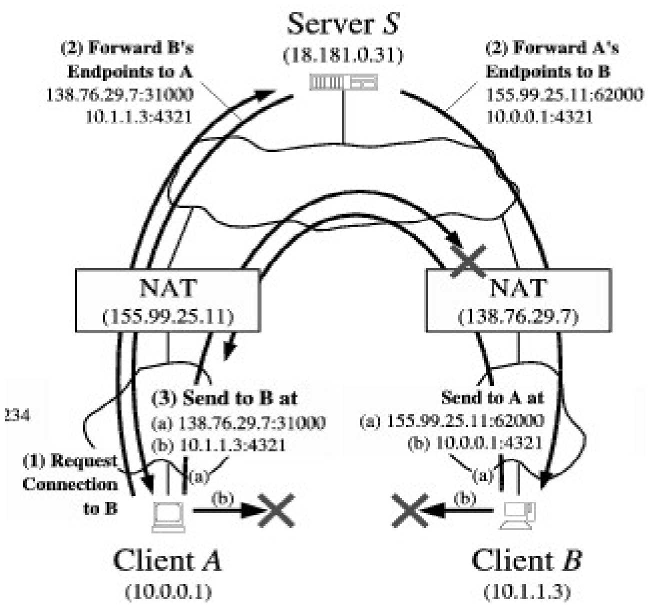


FIGURE4.15: CONNECTION TO RENDEZVOUS SERVER

Now client A follows the UDP hole punching technique as mentioned above to establish a UDP communication session directly with B. First, A sends a request message to S asking for help connecting with B. In response, S sends B's public and private endpoints to A, and send A's public and private End points to B. A and B starts trying to send UDP datagrams directly to each of these endpoints.

Since A and B are on different private networks and their respective private IP addresses are not globally routable, the messages sent to these endpoints will reach either the wrong host or no host at all. Because many NATs also act as

DHCP servers, handing out IP addresses in a fairly deterministic way from a private address pool usually determined by the NAT vendor by default, it is quite likely in practice that A's messages directed at B's private endpoint will reach *some* (incorrect) host on A's private network that happens to have the same private IP address as B does. Applications must therefore authenticate all messages in some way to filter out such stray traffic robustly. The messages might include application-specific names or cryptographic tokens, for example, or at least a random nonce pre-arranged through S.

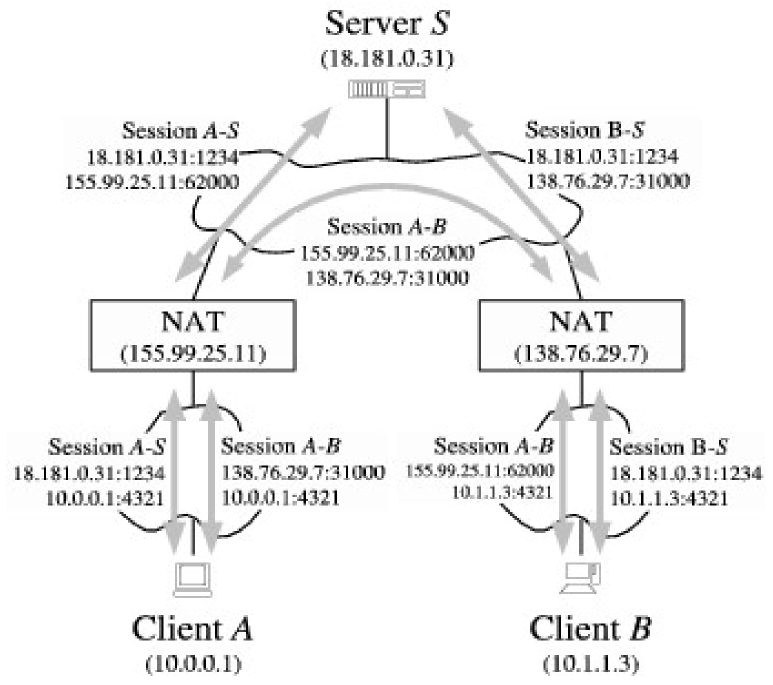


FIGURE 4.16: AFTER HOLE PUNCHING

Now consider A's first message sent to B's public endpoint, as shown in Figure 15. As this outbound message passes through A's NAT, this NAT notices that this is the first UDP packet in a new outgoing session. The new session's source endpoint (10.0.0.1:4321) is the same as that of the existing session between A and S, but its destination endpoint is different. If NAT A is well-behaved, it preserves the identity of A's private endpoint, consistently translating *all* outbound sessions from private source endpoint 10.0.0.1:4321 to the corresponding public source endpoint 155.99.25.11:62000. A's first outgoing message to B's public endpoint thus, in effect, “punches a hole” in A's NAT for a new UDP session identified by the endpoints (10.0.0.1:4321, 138.76.29.7:31000) on A's private network, and by the endpoints (155.99.25.11:62000, 138.76.29.7:31000) on the main Internet.

If A's message to B's public endpoint reaches B's NAT before B's first message to A has crossed B's own NAT, then B's NAT may interpret A's inbound message as unsolicited incoming traffic and drop it. B's first message to A's public address, however, similarly opens a hole in B's NAT, for a new UDP session identified by the endpoints (10.1.1.3:4321, 155.99.25.11:62000) on B's private network, and by the endpoints (138.76.29.7:31000, 155.99.25.11:62000) on the Internet. Once the first messages from A and B have crossed their respective NATs, holes are open in each direction and UDP communication can proceed normally. Once the clients have verified that the public endpoints work, they can stop sending messages to the alternative private endpoints.

4.1.4.5 Peers behind different Levels of NATs:

In some topologies involving multiple NAT devices, two clients cannot establish an “optimal” P2P route between them without specific knowledge of the topology. Consider a final scenario, depicted in Figure 16. Suppose NAT C is a large industrial NAT deployed by an internet service provider (ISP) to multiplex many customers onto a few public IP addresses, and NATs A and B are small consumer NAT routers deployed independently by two of the ISP's customers to multiplex their private home networks onto their respective ISP-provided IP addresses. Only server S and NAT C have globally routable IP addresses; the “public” IP addresses used by NAT A and NAT B are actually private to the ISP's address realm, while client A's and B's addresses in turn are private to the addressing realms of NAT A and NAT B, respectively. Each client initiates an outgoing connection to server S as before, causing NATs A and B each to create a single public/private translation, and causing NAT C to establish a public/private translation for each session.

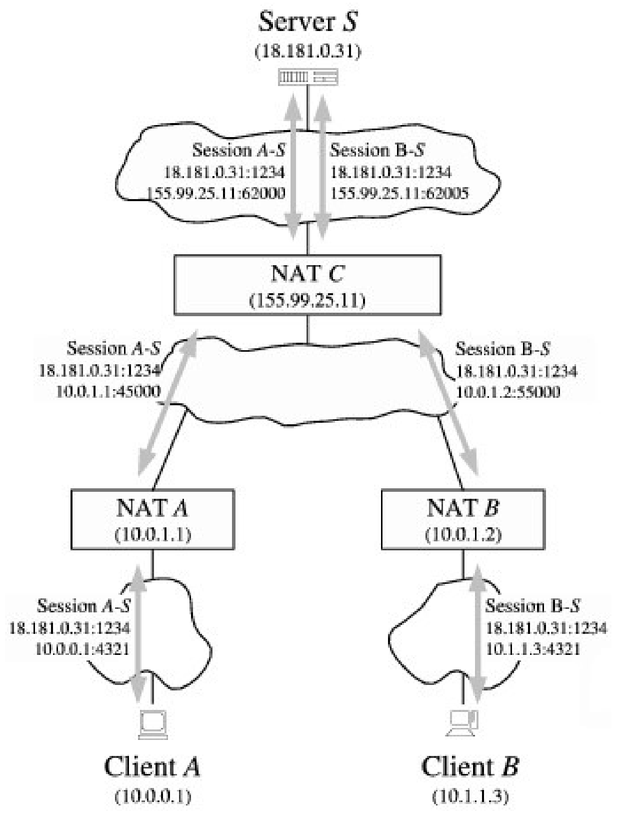


FIGURE4.17: BEFORE PUNCHING

Now suppose A and B attempt to establish a direct peer-to-peer UDP connection via hole punching. The optimal routing strategy would be for client A to send messages to client B's "semi-public" endpoint at NAT B, 10.0.1.2:55000 in the ISP's addressing realm, and for client *B* to send messages to A's "semi-public" endpoint at NAT B, namely 10.0.1.1:45000. Unfortunately, A and B have no way to learn these addresses, because server S only sees the truly global public endpoints of the clients, 155.99.25.11:62000 and 155.99.25.11:62005 respectively. Even if A and B had some way to learn these addresses, there is still no guarantee that they would be usable, because the address assignments in the ISP's private address realm might conflict with unrelated address assignments in the clients' private realms. (NAT A's IP address in NAT C's realm might just as easily have been 10.1.1.3, for example, the same as client B's private address in NAT B's realm.)

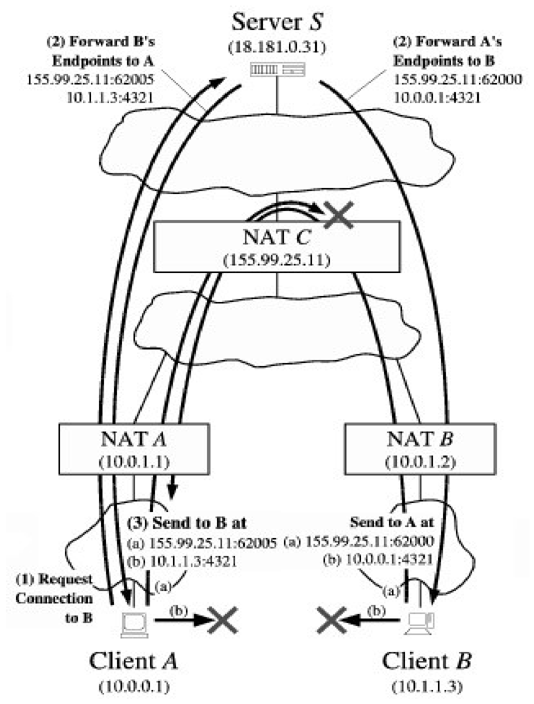


FIGURE 4.18: THE HOLE PUNCHING PROCESS

The clients therefore have no choice but to use their global public addresses as seen by S for their P2P communication and rely on NAT C providing hairpin or loopback translation. When A sends a UDP datagram to B's global endpoint, 155.99.25.11:62005, NAT A first translates the datagram's source endpoint from 10.0.0.1:4321 to 10.0.1.1:45000. The datagram now reaches NAT C, which recognizes that the datagram's destination address is one of NAT C's own translated public endpoints. If NAT C is well-behaved, it then translates both the source and destination addresses in the datagram and "loops" the datagram back onto the private network, now with a source endpoint of 155.99.25.11:62000 and a destination endpoint of 10.0.1.2:55000. NAT B finally translates the datagram's destination address as the datagram enters B's private network, and the datagram reaches B. The path back to A works similarly. Many NATs do not yet support hairpin translation, but it is becoming more common as NAT vendors become aware of this issue.

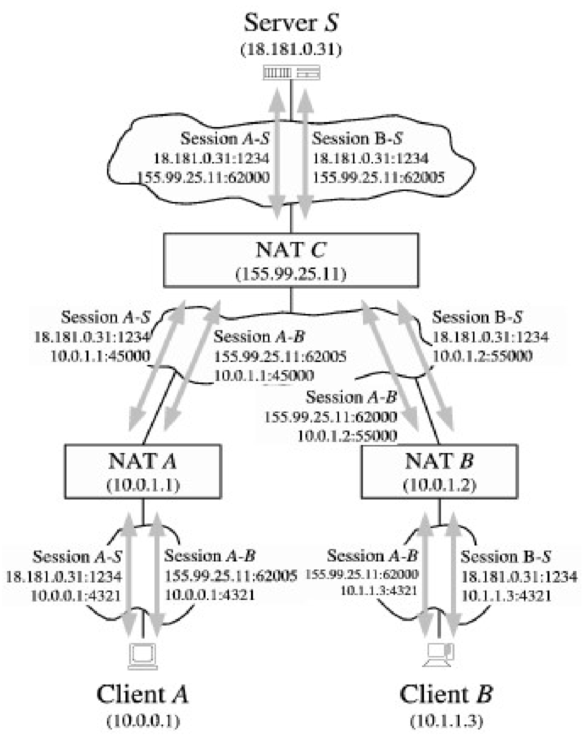


FIGURE 4.19: AFTER HOLE PUNCHING

4.1.5 TCP Hole Punching:

The main practical challenge to applications wishing to implement TCP hole punching is not a protocol issue but an application programming interface (API) issue. Because the standard Berkeley sockets API was designed around the client/server paradigm, the API allows a TCP stream socket to be used to initiate an outgoing connection via connect (), or to listen for incoming connections via listen() and accept(), but not both. Further, TCP sockets usually have a one-to-one correspondence to TCP port numbers on the local host: after the application binds one socket to a particular local TCP port, attempts to bind a second socket to the same TCP port fail.

For TCP hole punching to work, however, we need to use a single local TCP port to listen for incoming TCP connections and to initiate multiple outgoing TCP connections concurrently. Fortunately, all major operating systems support a special TCP socket option, commonly named SO_REUSEADDR, which allows the application to bind multiple sockets to the same local endpoint as long as this option is set on all of the sockets involved. BSD systems have introduced a SO_REUSEPORT option that controls port reuse separately from address reuse; on such systems both of these options must be set.

It may occur so that an application like this should use a TCP connection instead of UDP. It is true that we have a lot of problem when we implement the application via UDP but the case study shows from report by Bryan Ford in his iconic work named “Peer-to-Peer Communication Across Network Address Translators” that not all of the router supports the TCP NAT traversal and it is fairly complex procedure. And also, the hardware we used in this project gave us a negative result.

4.2 Rendezvous Server:

Rendezvous server is a well-known server which is accessible globally to any device that is connected to Internet. This enables P2P network peers to find each other. A rendezvous server uses a handshaking model unlike an eager protocol which directly copies the data. In a rendezvous protocol the data is sent when the destination says it is ready. But in eager protocol the server assumes that the host sends the data already assuming the peer is ready to handle the data.

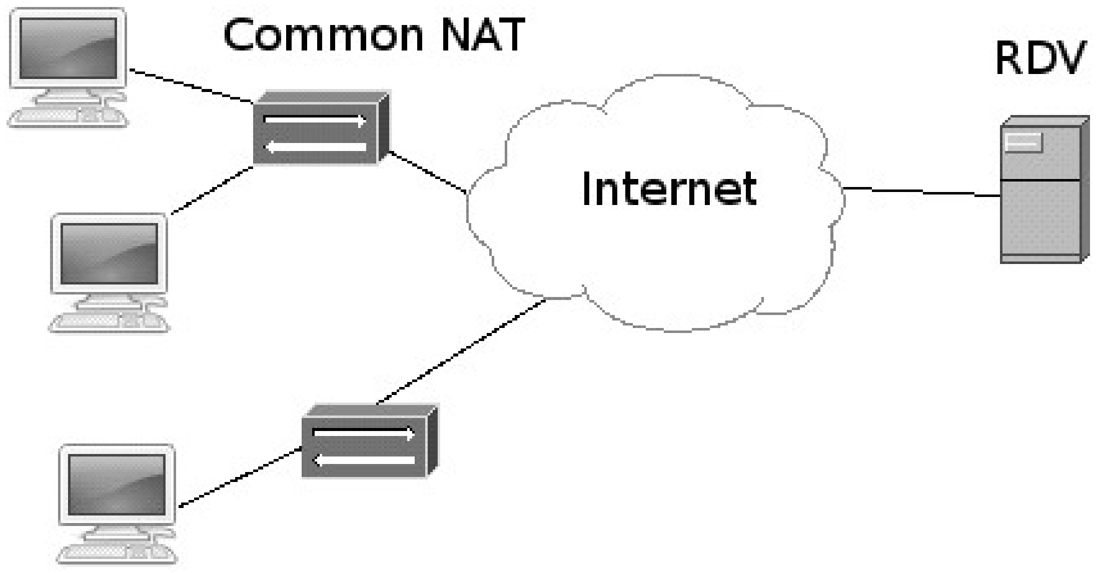


FIGURE 4.20: RDV(RENDEZVOUS SERVER)

In the fig:15 we can see that there are normal computers those are behind some NAT or directly connected to the Internet. And there is a server to which the peers are trying to make a connection or sends a request for other peer asking for their IP addresses or ports or what their requests are. If the server is capable of answering those questions it immediately replies, or in a more dynamic process like ours the server waits for the acknowledgement from the other end to give the updated result. It may happen so that the NAT is using a dynamic addressing mode like DHCP so change of IP address can be frequent. So, maintaining database of existing user may not always work taken granted that we don't have problem of hole punching and harsh firewalls.

To these requests our rendezvous server responds or sends acknowledgements in TCP or UDP connection the IP addresses and port should be used to connect to the opposite sides. The server in this process should be a well-known address. So either you need to get a dedicated line for your server or if you have access to your ISP (Internet Service Provider) you can ask him to forward the port or if you are the admin of that channel you can reduce your stress by simply using DMZ settings in your router. Any of which will make your IP globally accessible.

All the above-mentioned procedures are hardware specifications. To implement such a server, we need a server-side scripting language like PHP, JAVA, JavaScript, ASP, Pearl etc. And to run these scripts a suitable platform on that end is also mandatory.

4.3 Media Streaming:

For the time being we used a HTTP server to perform our basic testing in local area network or LAN. Because setting up an TCP connection in local area network is fairly simple and so is to stream a media using HTTP. But in case of Wide Area Network which is guided by a hole punching system a setting up a TCP connection is more than just complex. And so, the HTTP server won't work either. So, we used an UDP connection to transfer data bytes from device to device. This though requires packet verification and loss of packets, still it communicates just fine.

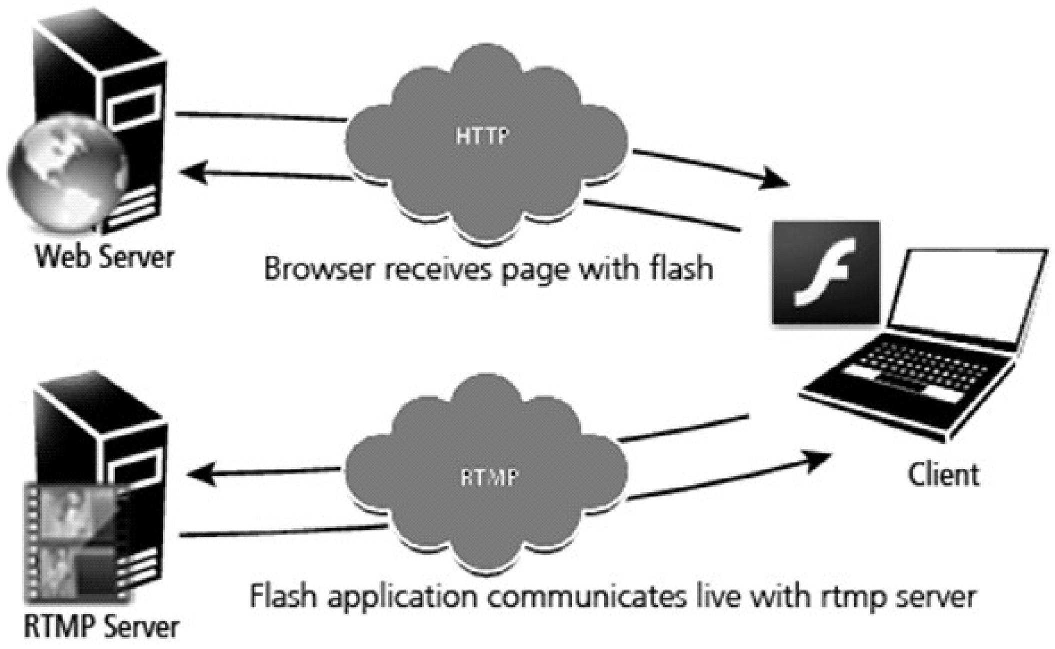

Web Server
HTTP
Browser receives page with flash
Client
RTMP
RTMP Server
Flash application communicates live with rtmp server

FIGURE 4.21: STREAMING MEDIA

In our case as we said earlier HTTP was used any device in that matter of fact has a browser that supports flash player (usually supported by any video LAN) can easily play this kind of stream. Other transport layer protocols like RTMP, Apple's HLS, Microsoft's Smooth Streaming are also available but for our transparency we used TCP/IP for transport layer and HTTP for application layer.

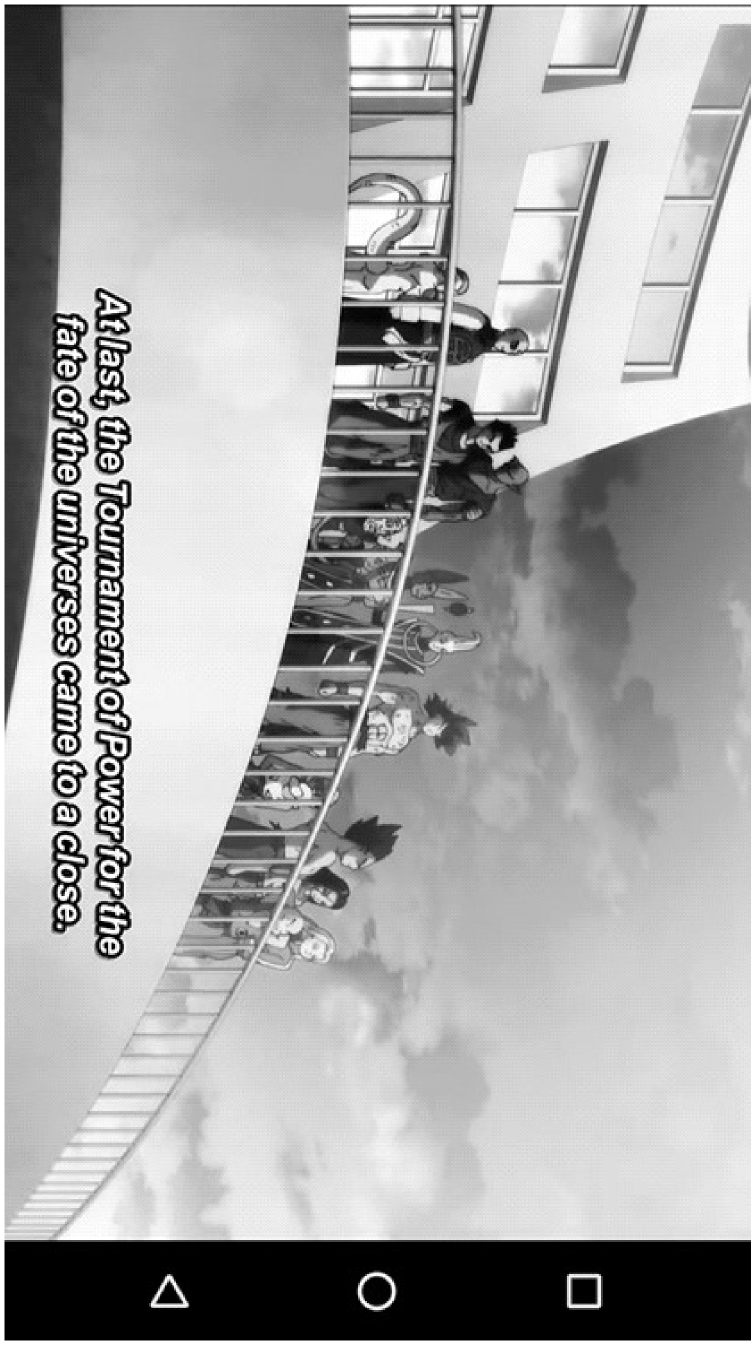

At last, the Tournament of Power for the fate of the universes came to a close.

FIGURE T_1: A VIDEO IS BEING STREAMED

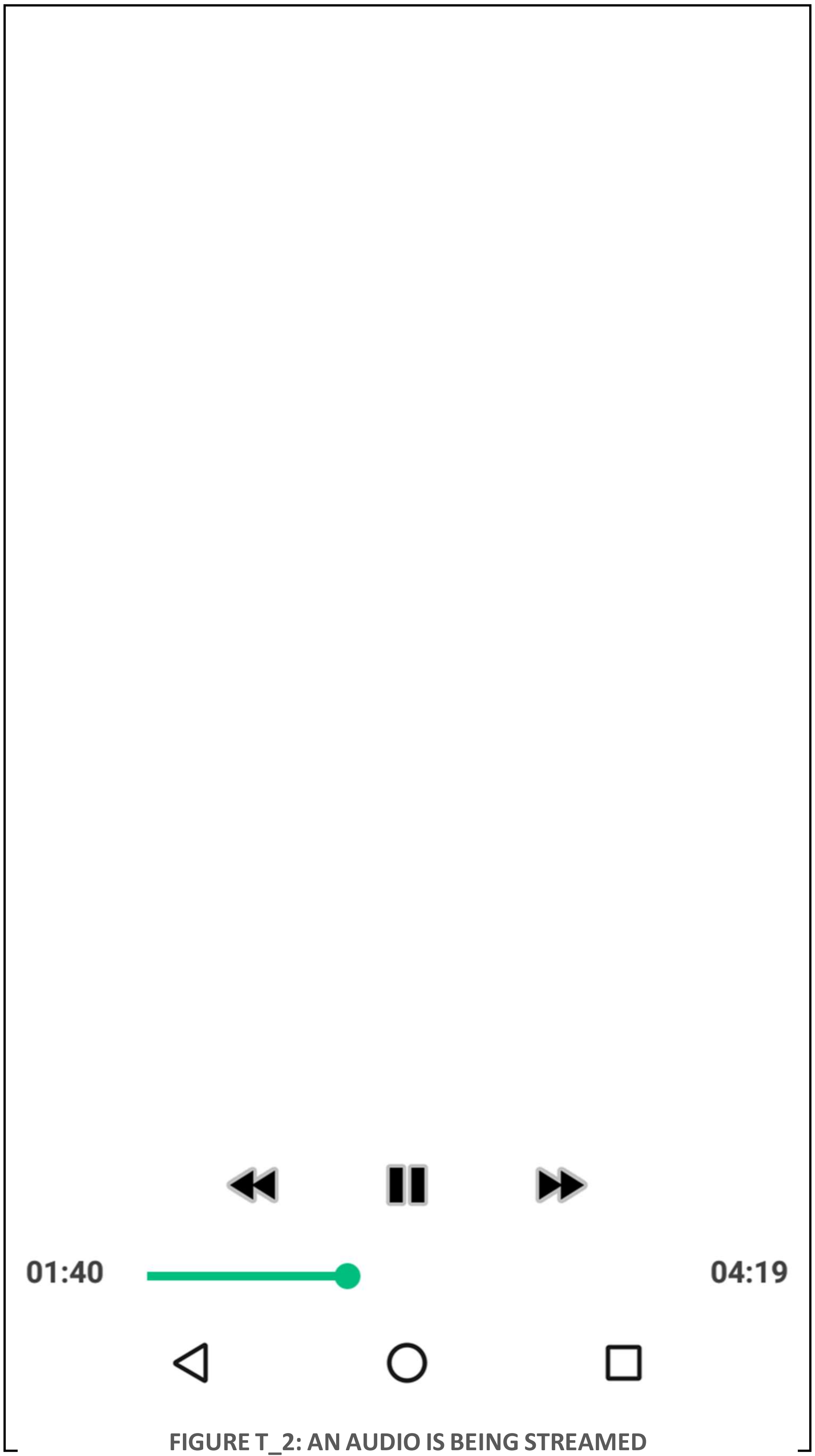


FIGURE T_2: AN AUDIO IS BEING STREAMED

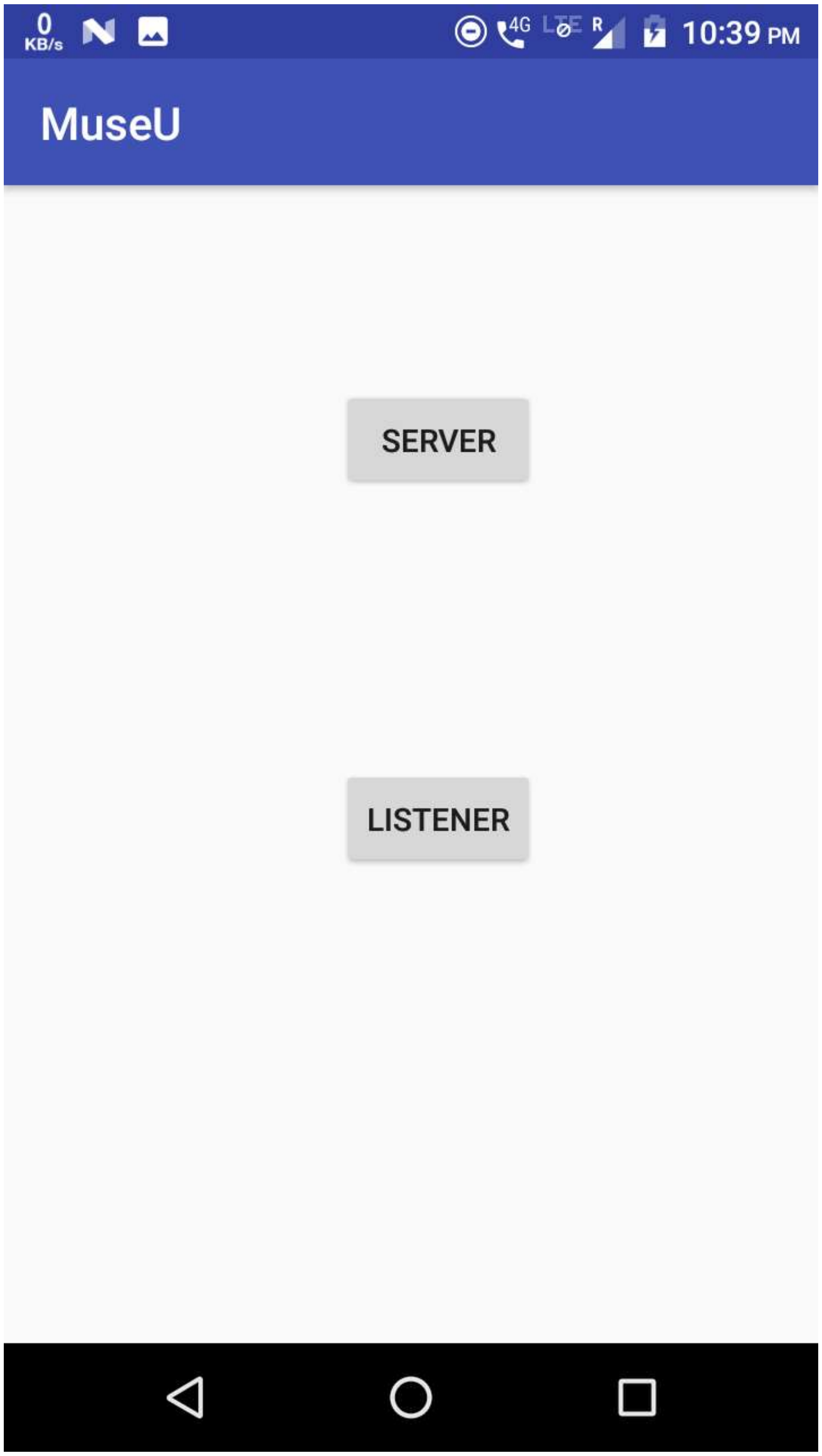
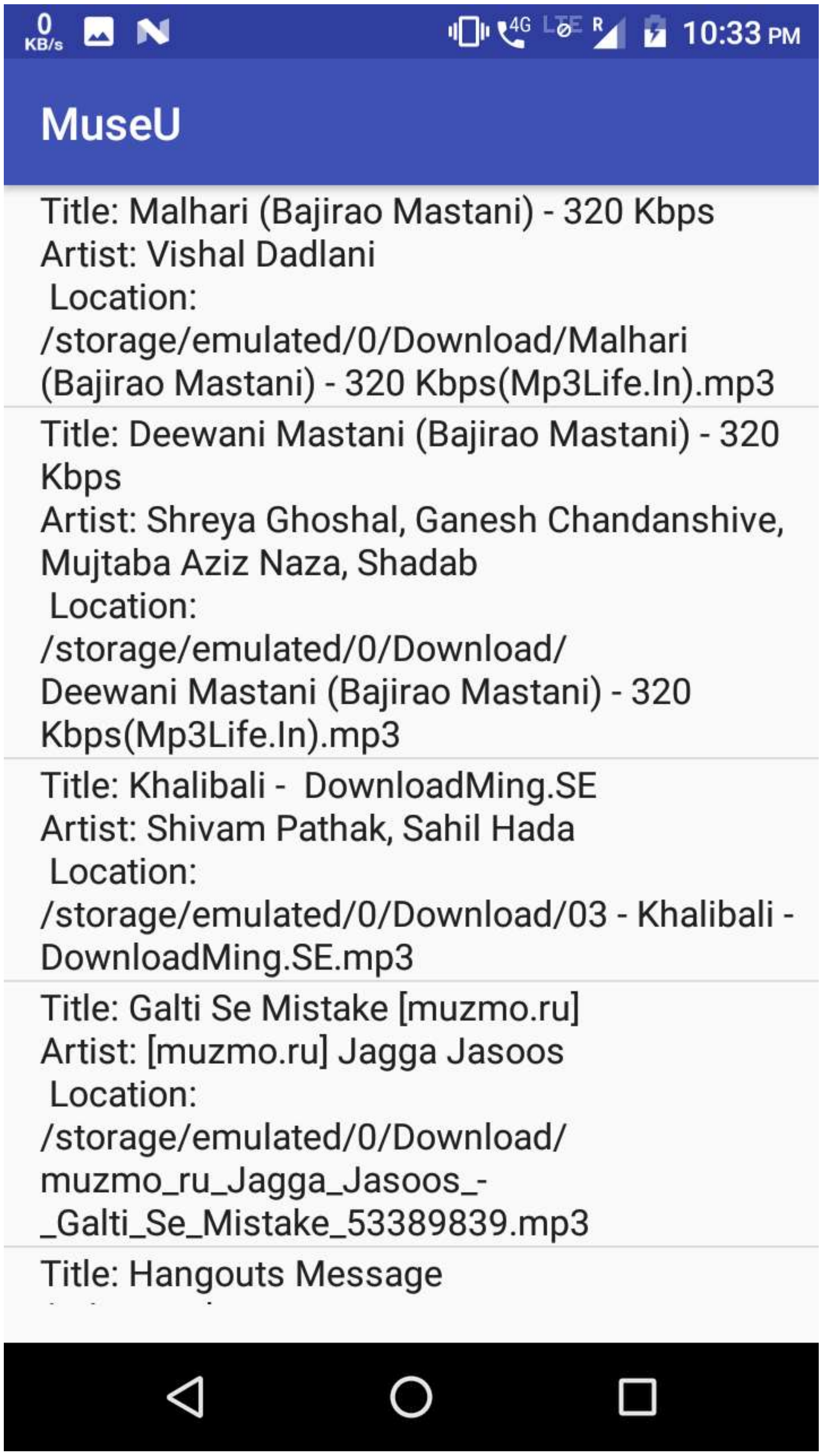


FIGURE T_3: HOME PAGE OF OUR APPLICATION



Title: Malhari (Bajirao Mastani) - 320 Kbps
Artist: Vishal Dadlani
Location:
/storage/emulated/0/Download/Malhari (Bajirao Mastani) - 320 Kbps(Mp3Life.In).mp3

Title: Deewani Mastani (Bajirao Mastani) - 320 Kbps
Artist: Shreya Ghoshal, Ganesh Chandanshive, Mujtaba Aziz Naza, Shadab
Location:
/storage/emulated/0/Download/Deewani Mastani (Bajirao Mastani) - 320 Kbps(Mp3Life.In).mp3

Title: Khalibali - DownloadMing.SE
Artist: Shivam Pathak, Sahil Hada
Location:
/storage/emulated/0/Download/03 - Khalibali - DownloadMing.SE.mp3

Title: Galti Se Mistake [muzmo.ru]
Artist: [muzmo.ru] Jagga Jasoos
Location:
/storage/emulated/0/Download/muzmo_ru_Jagga_Jasoos_-_Galti_Se_Mistake_53389839.mp3

Title: Hangouts Message

FIGURE T_4: LIST OF MEDIA IN A DEVICE

Future Scope

- As we have discussed earlier on, no other application in market does what we are doing or the way we are doing it. All the applications use a Huge server to send or receive files or steams. The way we are implementing our application could really be the future of file streaming between people.
- In near future when the network communities replace IPv4 by IPv6 our application will be full proof as the techniques like NAT and port forwarding Hole punching will no longer be needed. And every device will have a unique ID over the Internet. So, connecting every device by TCP will be much easier and streaming will much easy too.
- For the time being we used one of our computers as rendezvous server but for future if we had a well-known server or rendezvous server in the global space. So, we won't have to maintain a 24X7 uprunning server.
- For security reasons people can choose our app as the application they used mostly was controlled by the Facebook, Inc. And if some one keeps himself updated will know one of the biggest scam of this decade or data thievery was done by this group. We give people transparency about their data and it can never be read by the server of the application administrator.
- If we could further proceed and keep a tracker server for our application then we could enhance the speed significantly like torrents and other Peer-to-Peer applications does.
- This could be a file sharing that is legalized. As no body gets to download the contents no one is indulging in piracy. So, there are countries where torrent is already blocked and, in our country, it will be blocked soon so how will people share big files among them. Our application could serve this purpose for them.

Conclusion

The concept of museU comes from our daily basic needs. Flooded storage space and data security is at it's utmost importance these days. You can not deny the need of a file legal sharing application when the torrent goes down for good. It may not have a whole lot of things in it's basket but yet a song dedicated to you is always more soothing to ears than from a random play list.

Though our application has some trivial drawbacks like, both the devices have to be online to complete a transfer, Internet is required for both the devices, double data wears away for a single stream. UDP packets are not that trust worthy sending data stream. But then again Internet is something that became very cheap these days (after the jio invasion). And for UDP packets with 4G connections firmly available in India we can hope very soon the country will upgrade its communication system to IPv6 module and we could then easily move to TCP connection and make a well-established connection between two devices.

After these being said we conclude that the technology we are using or planning to use is not available in any of the available apps in market. The major problem though will be the NAME! If we want our application to be a social media and streaming application as we planned for we have a to fight big names like Facebook, WhatsApp, Hike, Line etc. So, the marketing of the application is never going to be easy! To make a lame man understand (as they are the majority) what is the small benefits of our application will be hard, and to change his mind about rejecting other unsafe applications with ours will be even harder. But still we hope for a high acceptance among people if we can have a good marketing technique and post production policy.

Bibliography

[1] **Wikipedia: Peer to Peer** – Wikipedia is a multilingual, web- based, free-content encyclopedia project supported by the Wikimedia Foundation and based on a model of openly editable content.

[2] **Wikipedia: Network Address Translation** – Wikipedia is a multilingual, web-based, free-content encyclopedia project supported by the Wikimedia Foundation and based on a model of openly editable content.

[3] **Wikipedia: Hole Punching (Networking)** – Wikipedia is a multilingual, web-based, free-content encyclopedia project supported by the Wikimedia Foundation and based on a model of openly editable content.

[4] **Wikipedia: IP Address** – Wikipedia is a multilingual, web-based, free-content encyclopedia project supported by the Wikimedia Foundation and based on a model of openly editable content.

[5] **Bryan Ford, Pyda Srisuresh, Dan Kegel** - Peer-to-Peer Communication Across Network Address Translators. ATEC '05 Proceedings of the annual conference on USENIX Annual Technical Conference

[6] "UDP Hole Punching, State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs)". ietf.org. 2008-03-01. Retrieved 2016-06-22.

[7] **Dan Kegel:** NAT and peer-to-peer networking, July 1999. http://www.alumni.caltech.edu/~dank/peer-nat.html.

[8] **Stack Overflow:** Networking in JAVA, Networking in Android Studio, UDP Hole Punching.

[9] **YouTube:** Tutorials on Android Studio, JAVA.

[10] **ciscopress.com:** Cisco Networking Academy's Introduction to Routing Dynamically.