

Networking Final Exercise

Submitted by:

Dana Zorohov 207817529

Noam Vanunu 318995156

Link to the project on github:

<https://github.com/noamv2/Chat-app-with-Reliable-file-transfer-over-UDP.git>

- All the project is on github: the codes, this pdf with explanations and the wireshark record.

Part I: How to run the program

Graphical User Interface

1. Open the terminal and run the server:

```
Terminal: Local x + v
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\danaz\PycharmProjects\Chat-app-with-Reliable-file-transfer-over-UDP> python server.py
```

Choose the percentage of packet loss you would like it to be, 0 for normal state:

```
Terminal: Local x + v
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

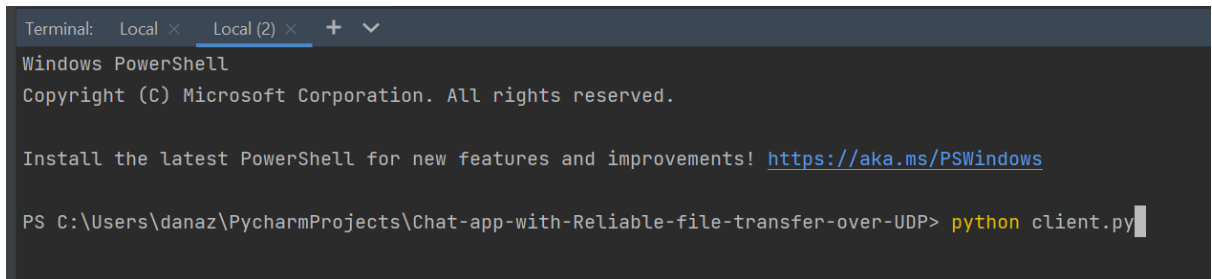
PS C:\Users\danaz\PycharmProjects\Chat-app-with-Reliable-file-transfer-over-UDP> python server.py
insert packet loss percentage, (0 for normal use, more for loss simulation):
0
```

Choose the percentage of packet corruption you would like, 0 for normal state:

```
PS C:\Users\danaz\PycharmProjects\Chat-app-with-Reliable-file-transfer-over-UDP> python server.py
insert packet loss percentage, (0 for normal use, more for loss simulation):
0
insert packet corruption percentage, (0 for normal use, more for simulation):
0
Server is up and listening.
```

If the message "Server is up and listening." Shows up in your terminal than you can move forward to the next step.

2. Open new terminal and run the client:



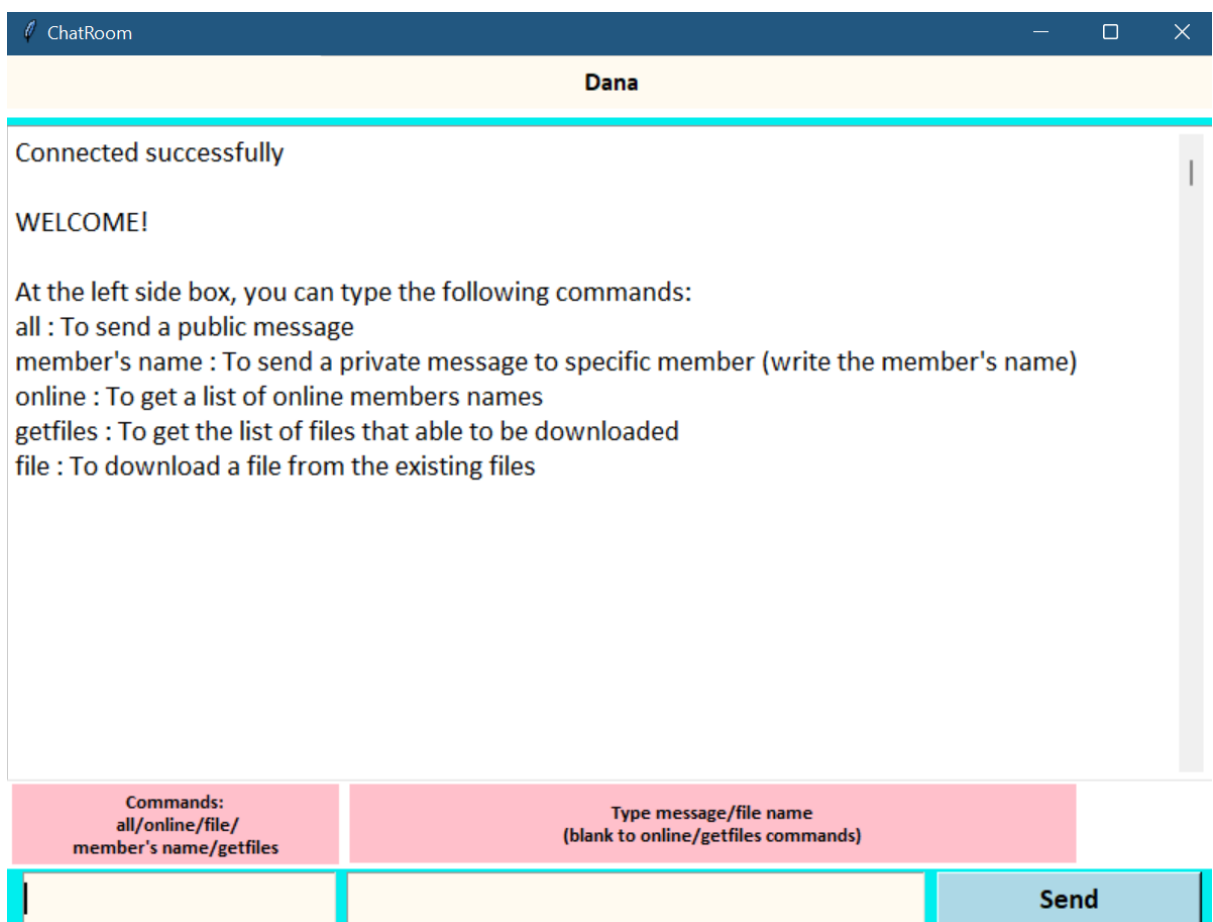
```
Terminal: Local x Local (2) x + v
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\danaz\PycharmProjects\Chat-app-with-Reliable-file-transfer-over-UDP> python client.py
```

3. Now a login window will appear, enter your name in the box, and press continue:
4. Now you entered the chat window.

The commands entry box is the smaller one at the left side, and the bigger one is where you type your message, of the file name you wish to download:



ChatRoom

Dana

Connected successfully

WELCOME!

At the left side box, you can type the following commands:

- all : To send a public message
- member's name : To send a private message to specific member (write the member's name)
- online : To get a list of online members names
- getfiles : To get the list of files that able to be downloaded
- file : To download a file from the existing files

Commands: all/online/file/ member's name/getfiles	Type message/file name (blank to online/getfiles commands)	Send
---	---	------

- If you want to see who's online, write online in the commands box, and nothing in the message box. Press the button send and the list of online members will appear in your chat window:

ChatRoom

Dana

Connected successfully

WELCOME!

At the left side box, you can type the following commands:
all : To send a public message
member's name : To send a private message to specific member (write the member's name)
online : To get a list of online members names
getfiles : To get the list of files that able to be downloaded
file : To download a file from the existing files

Commands:
all/online/file/
member's name/getfiles

Type message/file name
(blank to online/getfiles commands)

online

Send

ChatRoom

Dana

Connected successfully

WELCOME!

At the left side box, you can type the following commands:
all : To send a public message
member's name : To send a private message to specific member (write the member's name)
online : To get a list of online members names
getfiles : To get the list of files that able to be downloaded
file : To download a file from the existing files

Users Online: Dana

Commands:
all/online/file/
member's name/getfiles

Type message/file name
(blank to online/getfiles commands)

Send

- If you want to send private message, write name of online member in the commands box, and the message in the message box. Press the button send and the message will be sent to this member:

ChatRoom
— □ ×

another

Connected successfully

WELCOME!

At the left side box, you can type the following commands:

all : To send a public message

member's name : To send a private message to specific member (write the member's name)

online : To get a list of online members names

getfiles : To get the list of files that able to be downloaded

file : To download a file from the existing files

Users Online: someone, another

Commands:
all/online/file/
member's name/getfiles

Type message/file name
(blank to online/getfiles commands)

someone
hi
Send

ChatRoom
— □ ×

someone

Connected successfully

WELCOME!

At the left side box, you can type the following commands:

all : To send a public message

member's name : To send a private message to specific member (write the member's name)

online : To get a list of online members names

getfiles : To get the list of files that able to be downloaded

file : To download a file from the existing files

Unknown user or command

:> another joined the conversation

another[DM]: hi

Commands:
all/online/file/
member's name/getfiles

Type message/file name
(blank to online/getfiles commands)

Send

- If you want to send message to everyone who's online, write all in the commands box, and the message in the message box. Press the button send and the message will be sent to all the members:

ChatRoom

test

Connected successfully

WELCOME!

At the left side box, you can type the following commands:
all : To send a public message
member's name : To send a private message to specific member (write the member's name)
online : To get a list of online members names
getfiles : To get the list of files that able to be downloaded
file : To download a file from the existing files

Users Online: someone, another, test

Commands:
all/online/file/
member's name/getfiles

Type message/file name
(blank to online/getfiles commands)

all

test

Send

ChatRoom

another

Connected successfully

WELCOME!

At the left side box, you can type the following commands:
all : To send a public message
member's name : To send a private message to specific member (write the member's name)
online : To get a list of online members names
getfiles : To get the list of files that able to be downloaded
file : To download a file from the existing files

Users Online: someone, another

:> test joined the conversation

test:test

Commands:
all/online/file/
member's name/getfiles

Type message/file name
(blank to online/getfiles commands)

Send

ChatRoom

someone

At the left side box, you can type the following commands:
all : To send a public message
member's name : To send a private message to specific member (write the member's name)
online : To get a list of online members names
getfiles : To get the list of files that able to be downloaded
file : To download a file from the existing files

Unknown user or command

:> another joined the conversation

another[DM]: hi

:> test joined the conversation

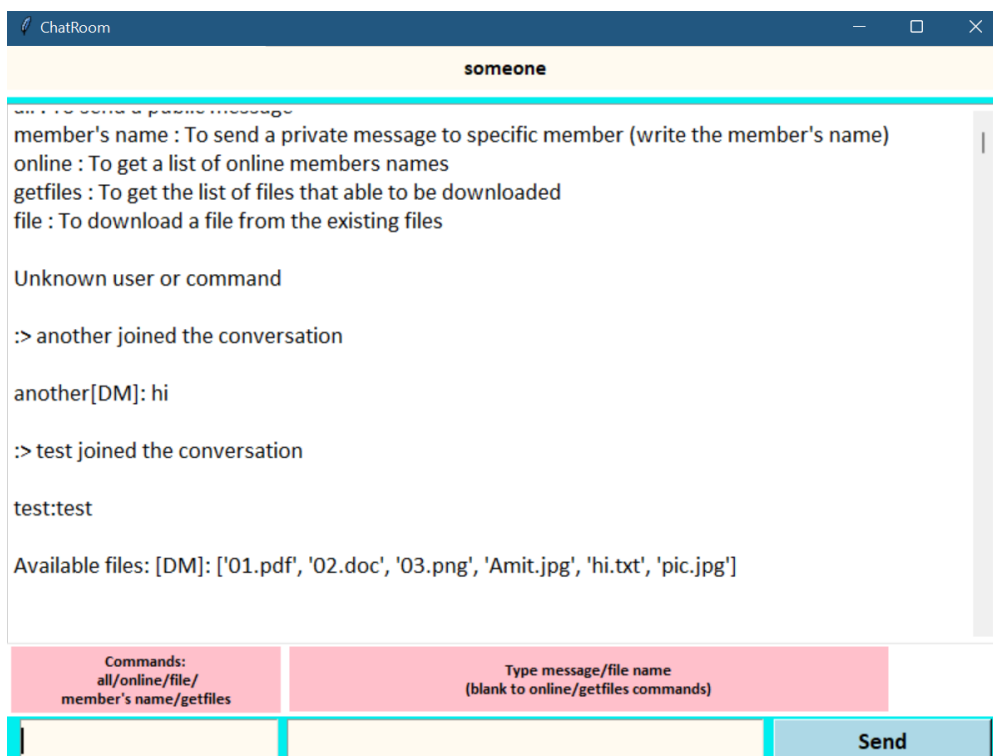
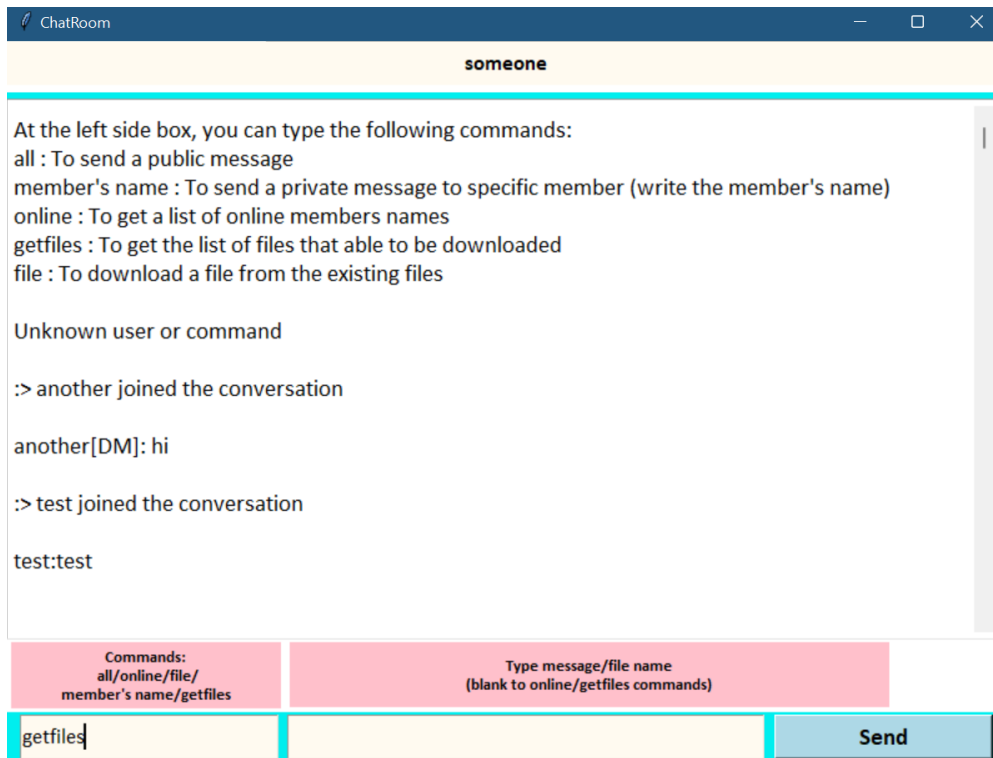
test:test

Commands:
all/online/file/
member's name/getfiles

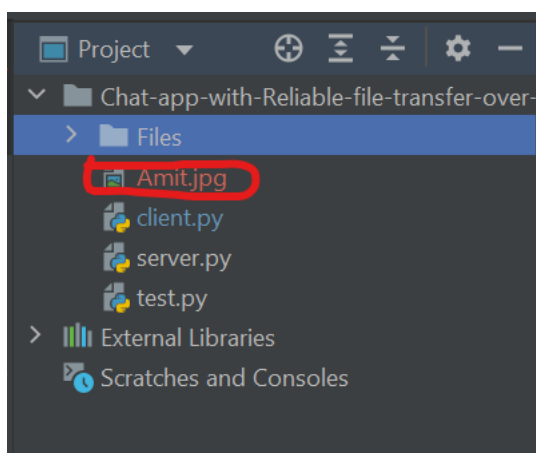
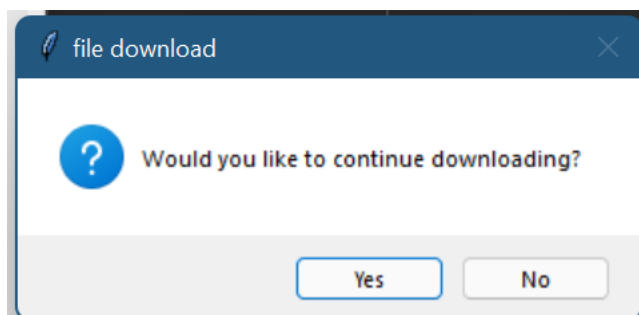
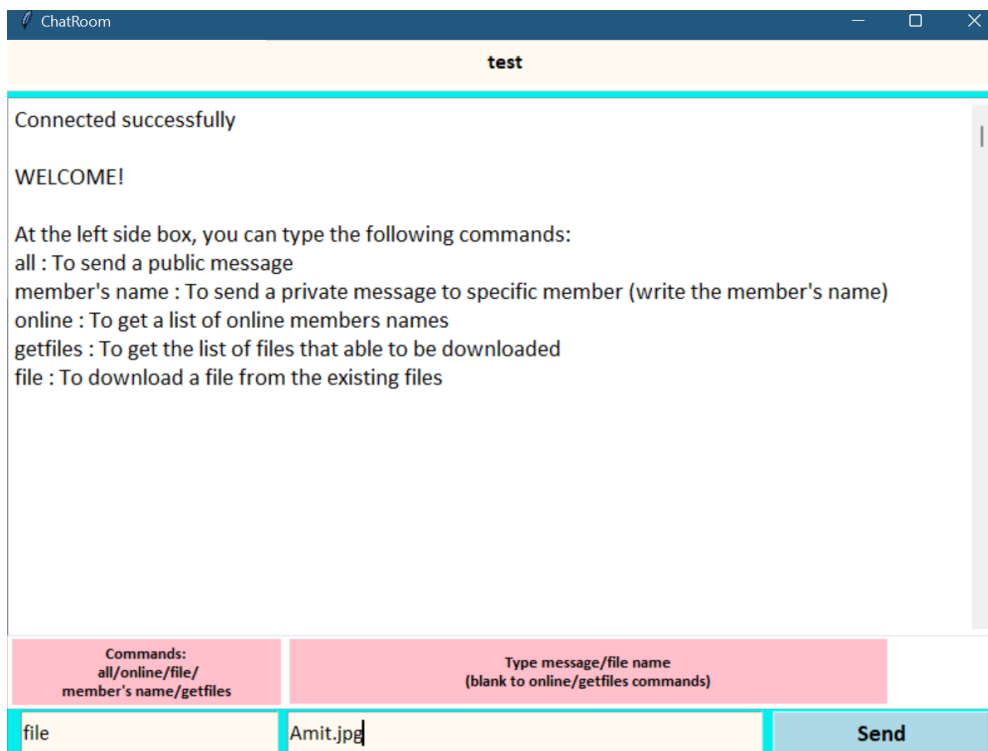
Type message/file name
(blank to online/getfiles commands)

Send

- If you want to see which files are available for download, write getfiles in the commands box, and nothing in the message box. Press the button send and the list of files will appear at your chat window:



- If you want to download a file, write file in the commands box, and the name of the file in the message box. Press the button send and the file will start downloading. At the middle of the downloading progress a pop-up message will appear and it will ask you if you wish to continue downloading:



- If you want to exit the chat you can press the red X button at the upper right side of the window.

Wireshark:

fe80::c97b:93d7:dc08:...	fe80::c97b:93d7:dc08:...	ICMPv6	178 Destination Unreachable (Address unreachable)
10.100.102.12	10.100.102.12	TCP	55 57852 → 50000 [PSH, ACK] Seq=1 Ack=1 Win=10233 Len=...
10.100.102.12	10.100.102.12	TCP	44 50000 → 57852 [ACK] Seq=1 Ack=12 Win=10233 Len=0
10.100.102.12	10.100.102.12	UDP	38 50000 → 57852 Len=6 ¹
10.100.102.12	10.100.102.12	UDP	35 50000 → 57852 Len=3 ²
10.100.102.12	10.100.102.12	UDP	568 50000 → 57852 Len=536
10.100.102.12	10.100.102.12	UDP	568 50000 → 57852 Len=536
10.100.102.12	10.100.102.12	UDP	568 50000 → 57852 Len=536
10.100.102.12	10.100.102.12	UDP	568 50000 → 57852 Len=536
10.100.102.12	10.100.102.12	UDP	568 50000 → 57852 Len=536
10.100.102.12	10.100.102.12	UDP	568 50000 → 57852 Len=536
10.100.102.12	10.100.102.12	UDP	568 50000 → 57852 Len=536
10.100.102.12	10.100.102.12	UDP	568 50000 → 57852 Len=536
10.100.102.12	10.100.102.12	UDP	568 50000 → 57852 Len=536
10.100.102.12	10.100.102.12	UDP	568 50000 → 57852 Len=536
10.100.102.12	10.100.102.12	UDP	568 50000 → 57852 Len=536
10.100.102.12	10.100.102.12	UDP	568 50000 → 57852 Len=536

On the picture above, 1,2 represents that the server sends the name of the file.

All the packets below it are the sending process of the file (the file split to packets).

10.100.102.12	UDP	568 50000 → 57852	Len=536
10.100.102.12	UDP	568 50000 → 57852	Len=536
10.100.102.12	UDP	568 50000 → 57852	Len=536
10.100.102.12	UDP	568 50000 → 57852	Len=536
10.100.102.12	UDP	568 50000 → 57852	Len=536
10.100.102.12	UDP	568 50000 → 57852	Len=536
10.100.102.12	UDP	36 50000 → 57852	Len=4 - ³
10.100.102.12	UDP	110 57852 → 50000	Len=78 ⁴
10.100.102.12	UDP	568 50000 → 57852	Len=536
10.100.102.12	UDP	568 50000 → 57852	Len=536
10.100.102.12	UDP	568 50000 → 57852	Len=536
10.100.102.12	UDP	568 50000 → 57852	Len=536

On the picture above, 3 represents req_Ack, server asks the client to send how many packets it received , 4 is the client's response – which packets it received.

Part II: Fast RUDP – safe file transfer over UDP

Fast VS TCP

The Tcp protocol take care for many things behind the scenes, providing a reliable data transfer between two host. However, our “Fast” protocol is based on UDP, so it has to provide those functions Itself in the application level.

RDT elements

Here are the staple functions that our protocol should provide for it to be called RDT:

- detection and correction of corrupted packets
- Loss recovery – make sure all the packets reaches the destination
- Latency – make sure retransmitted or delayed packets won't create a mess
- In order delivery
- Congestion control

In this section we will review the solutions we implemented to tackle these problems.

Packet loss recovery:

Arguably the most important attribute of any RDT protocol. In the early days of the internet, networks tended to be unreliable, and packets got lost frequently. Even today lossy networks or full buffers can cause a packet to get thrown or lost.

One of the major drawbacks of Tcp is the big overhead it generates to ensure arrival of packets (all the ACKs make the network crowded and slow down things even more).

Our aim is to reduce the number of unnecessary packets to minimum, our protocol follows these steps:

Sender side:

- 1) open a connection between two hosts
- 2) **The sender parses the file and split it to 500 bytes packets** – this step is done to avoid IP fragmentation. If we send a file that can't fit into lower protocols it will be split to smaller packets, and since we don't know the exact way it works, if one of these little packets get lost it will be hard to determine the exact data that was lost. **Sending 500 bytes packets ensures that fragmentation won't be a problem.**
- 3) **Sending the number of packets to the client – the client allocates an array with size equal to the number of incoming packets**, that way it can know if something missing.
- 4) The sender appends Sequence number and CRC remainder to the packets (see diagram below) and send all the packets to the client.
- 5) After all the packets were sent, **he sends ACK_REQ for the client, requesting to know which packets received correctly the response arrives in one message for all the packets** – hence saves lots of redundant ACK'S that would be sent in other protocols. **The server retransmit all the packets that got lost**

Receiver side:

- 1) The receiver gets the number of packets from the server and **create an empty array with that size**
- 2) When getting an uncorrupted packet, **the receiver checks if it has it already, if not he buffers it in the array** and mark the packet number as ACK'ed.

- 3) **After getting an ACK_REQ, the Receiver sends a list of all the ack'ed packets to the sender.**
- 4) When the buffer array is full the receiver ends the connection and **write the binary array to file with the same name as the requested file**

The FAST-UDP packet header:

seqNo	Data – up to 500 Bytes	CRC remainder
4 Bytes		32 Bytes

seqNo – is a value between 0 to PACKETS_NUM – 1

Error detection:

The Udp provide an error detection mechanism – checksum.

However, he can miss lot of errors, so we chose a more robust error detection algorithm – Cycle Redundancy Check – CRC we won't explain how it works since it was taught on class.

Before sending a packet, we compute its remainder and append it to the packet.

When receiving a packet, the Receiver compute it again and compare the two values

If they don't match, the packet discarded, and will be asked to be retransmitted again later.

Latency:

Lossy or slow Networks can cause a packet to be received more then once, in order to solve that, we check the seqNo of each incoming packet, if we already have the packet buffered, we discard it. In case that we got message that doesn't belong to the current state of the connection we discard it as well (example: open connection request, while it's already open).

In order delivery:

This one is simple, since our protocol aim is file transfer, we just wait for all the data to arrive while buffering it in the correct order(determined by the packet seq numbers) in our array, when we got all of it, we just write it from the start to the end of the array.

Congestion control:

Our protocol is already doing a good job, by reducing the number packets the moving around. Regardless, we should control adjust our sending speed to the environment and to the Receiver.

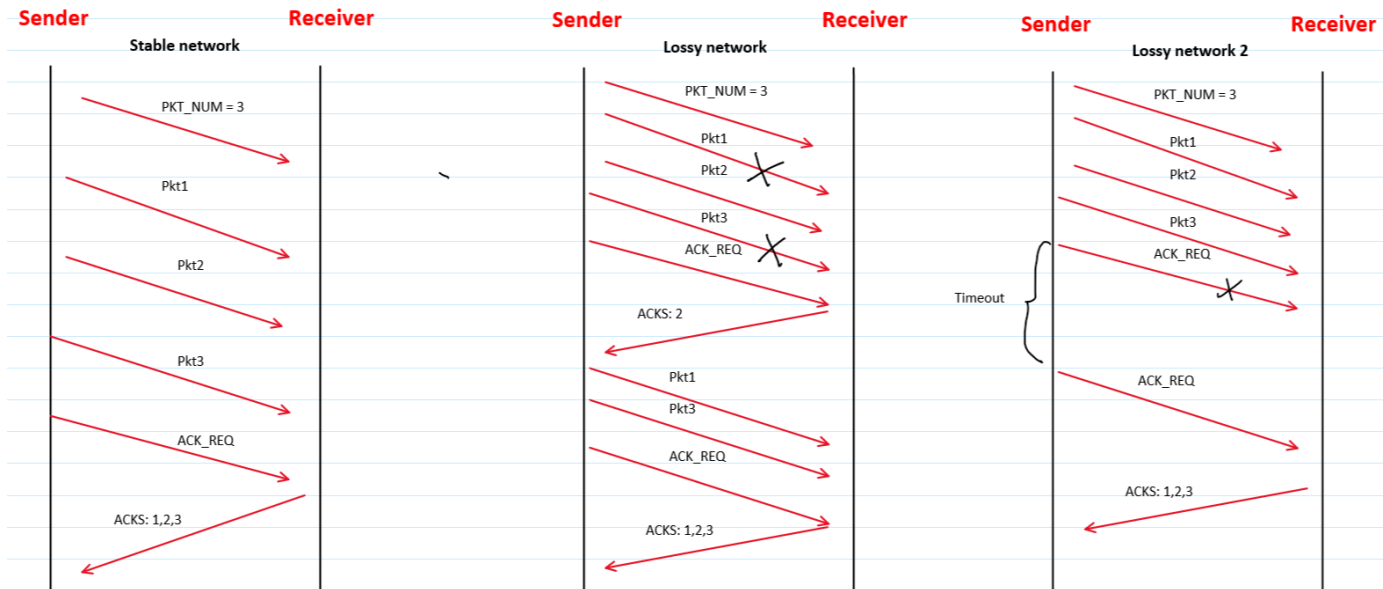
For this mission we employed couple of well-known elements.

We created a sending window with a size of 10 bytes – meaning, we send 10 bytes and wait for the cumulative acknowledgement message.

After creating the window we use the **slow-start** algorithm to enlarge it until we encounter the first congestion control event or we reached the maximum window size (file size in our case)

In case of congestion control event, the window size decreased, and it goes into **Fast recovery mode**

State Diagram:



Part III

- 1) At the beginning the computer knows only the MAC address of its own Network Card.

Connecting computer needs to get its own IP address: use *DHCP* (Dynamic Host Configuration Protocol).

First the computer sends DHCP Discover via broadcast to all the entities in the network over LAN.

The message manages to arrive to the DHCP server only if the computer is in the same broadcast Domain with it (means that the message should arrive without passing any router on its way).

The DHCP server sees the request and returns DHCP Offer, message which contains all the network details offered to the computer's network card: IP address, relevant DNS server and more. Since our network has one DHCP server, there won't be more offers, and the computer will send DHCP request message that notifies the DHCP server that the computer asks to receive the offer. At the end, the DHCP server sends DHCP ACK, and afterwards the computer can start using the IP address it was given to it.

Now the computer needs to learn about the IP address on the other side of the chat.

For that the computer needs to use DNS protocol and to ask its DNS server for the IP address of the other side of the chat.

So the computer needs to check if the IP address of the DNS server is located with it at the same subnet.

If it's not on the same subnet (can know by the mask), the OS needs to contact the Default Gateway (The computer knows its address using the DHCP process from before).

Since IP is a logical address, and the computer's network card knows only physical addresses – the computer needs to discover the MAC address of the default gateway.

ARP Protocol (Address Resolution Protocol) is needed for that. We assume that there are no record for the gateway in the ARP Cache of the computer, so the computer sends message via broadcast over LAN using Ethernet frames to all the network entities, asking for the MAC address (ARP request). The ARP reply sent to the switch that forwards it only to the computer that asked for it. The computer receives ARP reply with the gateway's MAC address.

At this point the computer has all the information it needs to send a packet to the DNS server.

The computer sends DNS query via LAN from computer to the gateway. The packet that sent contains computer's IP and destination IP – the DNS server (got it in DHCP process) and server name the computer needs.

The DNS reply a DNS packet with the IP address of the requested server.

If this DNS server don't have the requested IP it asks other DNS servers for this IP.

Two scenarios may happen at this point – whether the IP address will be found or it's doesn't exist.

Last step, computer has the destination IP, so now it can start the chat. Since HTTP protocol will be used, we need to pick up a TCP link with the server of the chat.

The port of the computer will be a random number that the OS draws.

To set up connection with the remote computer we use the Three Way Handshake.

The first packet is SYN. It indicates the start of the connection, so the SYN flag is on, ACK is off (because it's the first packet in the connection).

In the second packet the SYN flag is on since it's the first packet at the connection side between the server and the computer. ACK is on to provide approval that the SYN packet arrived.

The third and the last packet has the SYN flag off, ACK on, to approve the previous packet arrived.

Now the TCP connection is up. We can send now the HTTP request to the server.

The HTTP request sequence : GET -> OK.

And this is the end.

Transport Layer Protocol	Dest IP Dest Port Dest MAC	Source IP Source Port Source MAC	Message Info	Message Type
DHCP Protocol	All entities in the network MAC: FF:FF:FF:FF (indicates broadcast)	Our computer	Sent via Broadcast to receive network details	DHCP Discover
DHCP Protocol	Our computer	DHCP server	DHCP server sees the request above and returns DHCP Offer, the message contains: Our computer's IP address, relevant DNS server and more.	DHCP Offer
DHCP Protocol	DHCP server	Our computer	Notice to DHCP server that we accept the offer.	DHCP request
DHCP Protocol	Our computer	DHCP server	The DHCP server sends ack request so now our computer can use the IP	DHCP ACK

			address it received.	
ARP Protocol	All entities in the network	Our computer	Sent via broadcast to ask for the physical address of the gateway.	ARP request
ARP Protocol	Our computer	Gateway	Switch forwards ARP reply contains the MAC address of the default gateway only to our computer.	ARP reply
DNS Protocol	Dest IP: DNS server Dest MAC: default gateway	Source IP: Our Computer's IP Source MAC: our computer's MAC	Type A: translate domain name to IP address	DNS Query
DNS Protocol	Default gateway	DNS server	DNS packet over UDP with requested information.	DNS reply
TCP Protocol	server	Our computer	Indicates the start of the connection.	SYN
TCP Protocol	Our computer	server	Provides approval that the SYN packet arrived.	SYN ACK
TCP Protocol	server	Our computer	Approve the previous packet arrived.	ACK
HTTP/HTTPS Protocol	server	Our computer	To connect the chat.	GET
HTTP/HTTPS Protocol	Our computer	server	OK 200 Chat is willing to send it's page.	OK

2) "CRC" – Cyclic Redundancy Check

It's a way of checking if a packet was corrupted while it was sent, means that the bytes order have been changed, or some byte have been changed. It's based on binary division.

How CRC works:

Additional data added to a message in order to detect if any error has occurred.

The additional data:

Sender Side:

- Choosing a CRC key, it's format is usually x^3+1 (1001).
- We append to the end of the data zeroes with the length of the CRC key Size-1 (if the key is 1001, than we add 3 zeroes).
- We divide the data byte stream(+zeroes we appended) with our key. It will be a modulo 2 division.
- If the leftmost bit of the dividend (or the part used in each step) is 0, the step cannot use the regular divisor; we need to use an all-0s divisor.
- So in this case we perform XOR on the dividend with zeroes.
- If the leftmost bit of the dividend is 1, replace the dividend by the result of XOR and pull 1 bit down.
- For the last n bits, we must carry it out normally as increased value of pick will cause Index Out of Bounds.

Receiver Side:

- At the receiver side, the incoming data unit from the sender is divided by the same CRC key.
- If at this step there is no remainder, the data unit is assumed to be correct and is therefore accepted.
- A remainder indicates that the data unit has been damaged in transit and therefore must be rejected.

3) here are the key differences between the protocols

HTTP 1 – for each object requested, a separate connection is opened. which mean that for each object, we have to wait 2 RTT's to get it. For 10 object we wait 20 RTT's /

HTTP 1.1 – introduced support for persistent connection in which we open a connection one time, and any following request is sent over the same connection. Now for getting 10 objects we only have to wait 11 RTT's (ignoring size, handshakes, and other delays).

HTTP 2 – this version introduced several improves to HTTP 1.1 while keeping its structure

- Headers are now represented as binary information instead text – making the packages lighter
- Can break down large packets into smaller frames and send them in between other packets/ This helps to counter Head Of the Line problem (HOL) where a single big packet can delay The requests of multiple other users – think of waiting in line at the supermarket with only milk and bread, behind a family of 6 doing its monthly groceries.
- Allow “push” – sending objects that we know that the user will probably ask for latter. e.g: When user ask for the html body of the site, we know that a follow up requests of the jpeg objects in it are highly likely to come soon.

QUIC – the biggest difference between QUIC and the previous protocol one the is the transport protocol it uses. While HTTP 1/1.1/2 use TCP, QUIC use UDP.

Transport protocols are implemented on the kernel level, so creating a new one will require a hard work and many years until every unit in the network support it. So google decided to use the already implemented UDP and add RDTP elements on the app level. Choosing the UDP protocol over the TCP provide several advantages such as: no need for the 3-way handshake ,can add TCP-like features that works on every hardware since they are implemented on the App layer.

4) Ports help us to differentiate between different protocol and services, the transport layer protocols use them and distribute and collect packet to and from the different app layer services.

5) subnet is a logical division of ip addresses, computer within the same subnet share a number of MSB's. The shared bits identify the subnet, while the rest of the address is used to identify a particular host within it.

Subnetting helps with routing packets around, as well as allowing organizations, states, ISP's etc. To have their own administrative network associated with them.

- 5) - MAC addresses are physical addresses only used in LAN to identify a device.
- However, local IP addresses can also identify a device on a local network as it is unique. So why to use both? It's simple, the local IP address will dynamically change when we disconnect from the WIFI network or ethernet. So, MAC address is a unique address for our network card. We can uniquely identify someone's device from it. It cannot be changed. We know that a packet can sure get to the target device with an IP table instead of MAC table, but with dynamic IP, the packet may finally get to the wrong device - when the old device get offline and a new device takes the same IP. The new device cannot decide if the packet is sent to itself without the an identifier, that's when the MAC comes in.

- 6) Router and Switch are both network connecting devices. Router works at network layer and is responsible to find the shortest path for a packet whereas Switch connects various devices in a network. Router connects devices across multiple networks.

The main differences between these two:

1. Router main objective is to connect **various networks** while Switch main objective is to connect **various devices** in a network.
2. Router works in Network Layer while Switch works in Data Link Layer.
3. Router is used in LAN and MAN while Switch is used only in LAN.
4. Router sends data in form of packets while Switch sends data in form of packets and frames.
5. Routing type is Adaptive and Non-adaptive routing while Switching type is Circuit, Packet and Message switching.
6. Router will offer NAT, NetFlow and QoS Services, while Switch will not offer such services.
7. Router Store **IP address** in the routing table and maintain an address on its own, while Switch Store **MAC address** in a lookup table and maintain an address on its own. However, Switch can learn the MAC address.
8. Router Networking device 2/4/8 ports while A switch is a multi-port bridge. 24/48 ports.
9. Router's speed limit is 1-10 Mbps for wireless and 100 Mbps for wired connection while Switch's speed limit for the switch is 10/100Mbps.
10. Router Helps users to take the faster routing decision while Switch is Likely to take a more complicated routing decision.
11. In Router, every port has its own broadcast domain while The switch has one broadcast domain except VLAN implemented.
12. Routers can work within both wired and wireless network situations while Switches are restricted to wired network connections.

Difference between router and NAT:

NAT is translating a private IP to a Public IP or Vice Versa. On the other hand, routing is the process of moving a packet of data from source to destination.

- 8) There are 2 main methods created to handle the IPV4 addresses shortage

- IPV6 – a newer protocol which support **much** larger number of addresses. The transition to IPV4 is painfully slow and complicated, so to keep the internet running in the meantime, the next method was created
- Nat – Nat allows many computers on the same subnet to use the same ipv4 address.

The gateway to the subnet is assigned an IP address, and all the communication from within it is performed under this address. Each computer inside the subnet receives a local ipv4 address that replaced with the global one by the NAT when he is contacting host outside.

- 9) **e.** The router 3c, which is at AS3 and has eBGP connectivity to router 4c which is at AS4, 3c learns about sub-net x which is at AS4 with the use of **protocol eBGP**:

- The router 4c learns about X with the use of protocol RIP.
- When AS4 gateway 4c advertises path AS4, x to AS3 gateway 3c – AS4 promises to AS3 it will forward datagram towards x.

- f.** The router 3a, which is at AS3, learns about sub-net x which is at AS4 with the use of **protocol OSPF**:

- AS3 router 3c receives path advertisement AS4,x (via eBGP) from AS4 router 4c.
- Based on AS3 policy, AS3 router 3c accepts path AS4,x, propagates (via OSPF) to all AS3 routers.
- So now 3a learns about x.

- g.** The router 1c, which is at AS1, learns about sub-net x which is at AS4 with the use of **protocol eBGP**:

- AS3 router 3c receives path advertisement AS4,x (via eBGP) from AS4 router 4c.
- Based on AS3 policy, AS3 router 3c accepts path AS4,x, propagates (via OSPF) to all AS3 routers.
- Based on AS3 policy, AS3 router 3a advertises (via eBGP) path AS3, AS4, x, to AS1 gateway router 1c.

- h.** The router 2c , which is at AS2, learns about sub-net x which is at AS4 with the use of **protocol OSPF**:

- AS3 router 3c receives path advertisement AS4,x (via eBGP) from AS4 router 4c.
- Based on AS3 policy, AS3 router 3c accepts path AS4,x, propagates (via OSPF) to all AS3 routers.
- Based on AS3 policy, AS3 router 3a advertises (via eBGP) path AS3, AS4, x, to AS1 gateway router 1c.
- Based on AS1 policy, AS1 router 1c accepts path AS3, AS4, x, propagates (via RIP) to all AS1 routes.
- Based on AS1 policy, AS1 router 1b advertises (via eBGP) path AS1, AS3, AS4, x to AS2 gateway router 2a.
- Based on AS2 policy, AS2 router 2a accepts path AS1, AS3, AS4, x, propagates (via OSPF) to all AS2 routers.
- So now 2c knows about x.