CHAT ROOM

Part 1:

Functions:

1. Connecting

2. Disconnecting

3. Getting the server connected client list

4. Getting the server existing files list

5. Downloading a file

6. Broadcast a message

7. Send messages to a specific client

8. Clear

We constructed a simple GUI that make it easy and comfortable for the user.

Every client has its own GUI window!
To use one of the functions one should only press the appropriate button in case of a simple request or input the desired data to the text window and press the appropriate button as well.

For **example**, if one wish to speak to a specific person, he will enter his name and press the Private button which will enable a client to speak directly to it friend. Then, all his messages will be directed to his friend only until he will ask otherwise.

GUI tutorial:
Before we start, please open a CMD and enter "ipconfig" and change the server IP and the client ip to your IP address. Example:

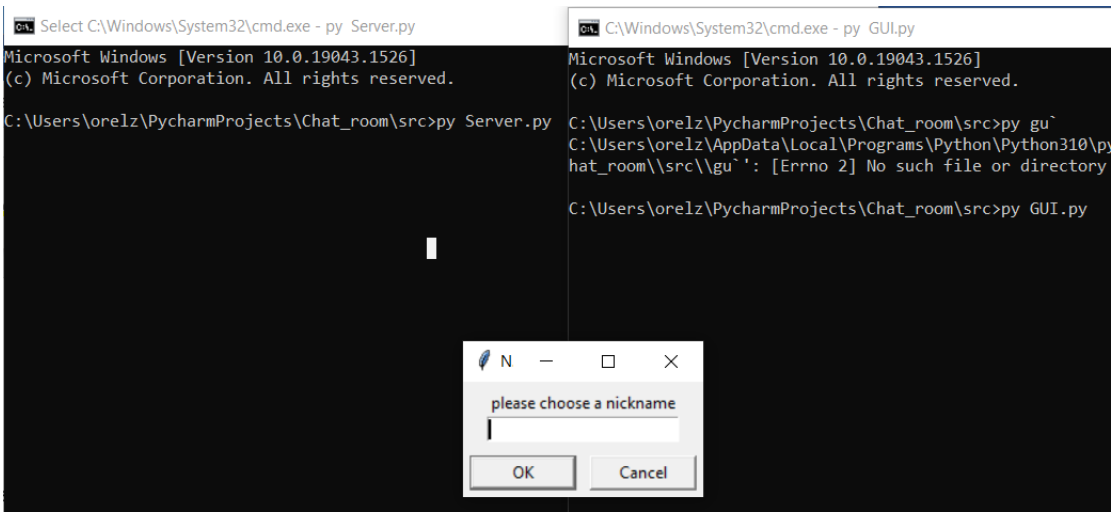
```
C:\Users\orelz\PycharmProjects\Chat_room\src>ipconfig

Windows IP Configuration


Ethernet adapter VirtualBox Host-Only Network:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::e816:2b3b:4cd3:efa9%5
   IPv4 Address. . . . . . . . . . . : 192.168.56.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter ⎫⎫ □□□□□ □□□□□* 11:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter ⎫⎫ □□□□□ □□□□□* 12:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter ⎫⎫ Wi-Fi:

   Connection-specific DNS Suffix  . : Home
   Link-local IPv6 Address . . . . . : fe80::cc7e:b9ad:a053:610b%2
   IPv4 Address. . . . . . . . . . . : 10.0.0.9
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.0.0.138
```

CHAT ROOM

First step: open CMD, enter the project folder and enter <py Server.py>

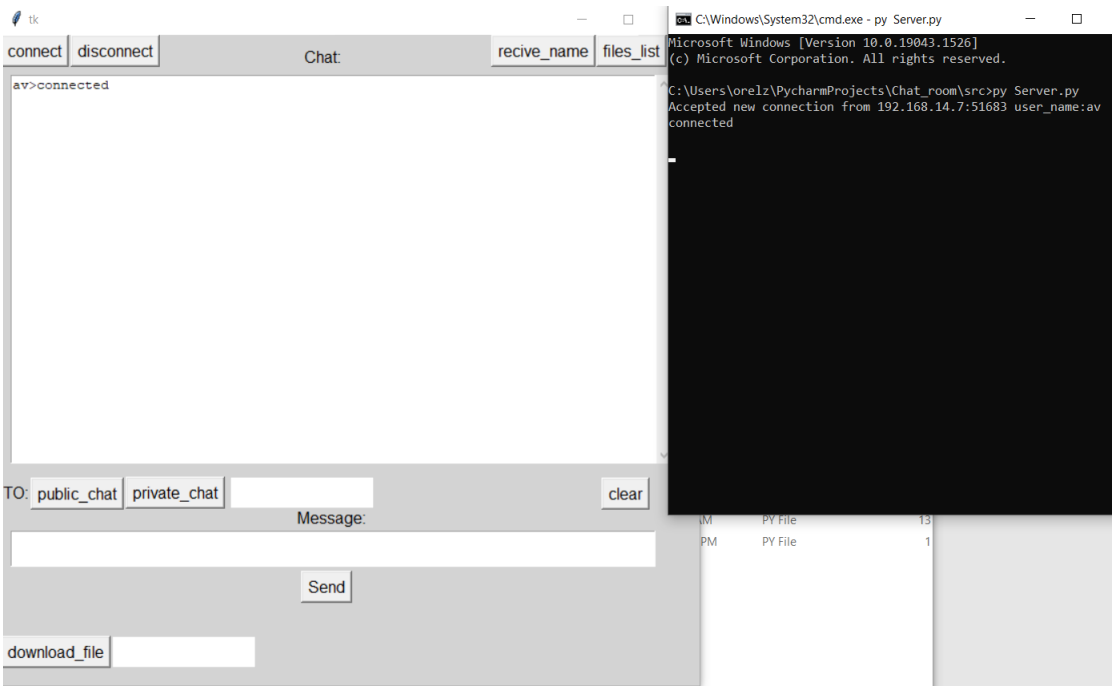
Second step: open CMD, enter the project folder and enter <py GUI.py>

Now, a GUI window will pop, **Example**:



Third step: enter our desired nickname and press OK.

A new GUI window will show:



All finished, Enjoy!

<u>How it works:</u>

CHAT ROOM

Note: The GUI class has a Client object. therefore, every time we activate the GUI from the terminal a new client is formed.
The GUI class divides into two main parts, one controls the requests which we send to the server and the other control the receiving part.

To do so, the two of them operate on spared threads in addition to the main one. There is no need for locks or synchronization because both are working on there part where there are no collisions.

Thread 1: "requests handler" is connected to the GUI button, always listening. When given an action, it calls the function that hands this type of request.
Thread 2: "receiving handler" always listening for the current action response. When a request has been made, it changes the class object named "action" to the current action Hence the "receiving handler knows what to expect.

Chat:

Protocols:

We handle the chat using the Transmission Control Protocol (**TCP**) protocol.

Both the client & server creating TCP & UDP socket accordingly, to make

the connection. Obviously, the server binds are TCP socket to a specific address (IP, PORT).

If one wish to connect, he will send a connect request to this address.

How the server & clients corresponds:

When activated, the Server awaits (listen) requests from clients.

Connect – The moment he gets a connection request from a new client he will accept if there are less than 6 clients connected already, assign him a specific port, and saves an important information about him. The server will broadcast our new chat friend connection.

All the requests are sent with the TCP protocol. But the file request use both TCP & UDP protocols. When requesting it will be transferred using the TCP socket, if the file exist, we will start a new communication throw the UDP socket for the transfer\download of the file itself.

As can see below in part B of this project in the diagram.

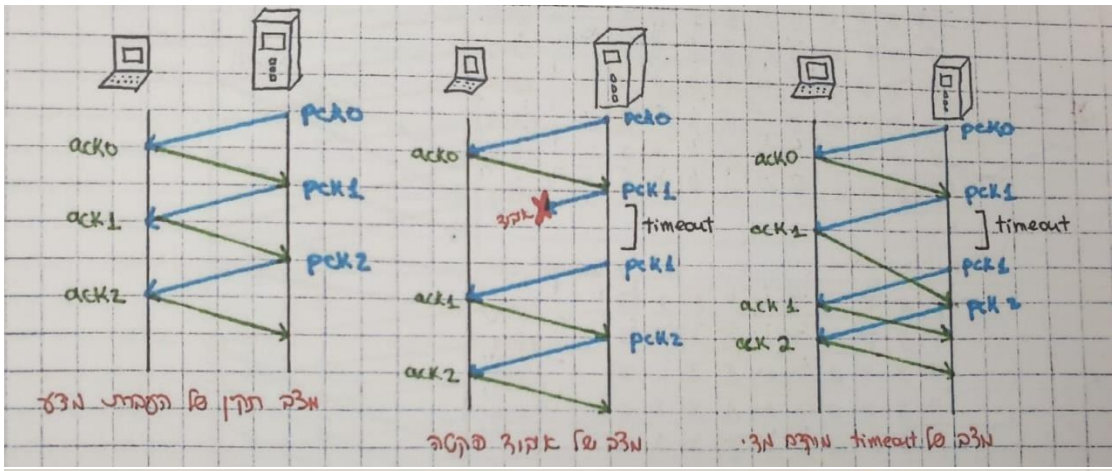
Now, the server is running, and we can use all the above functions.

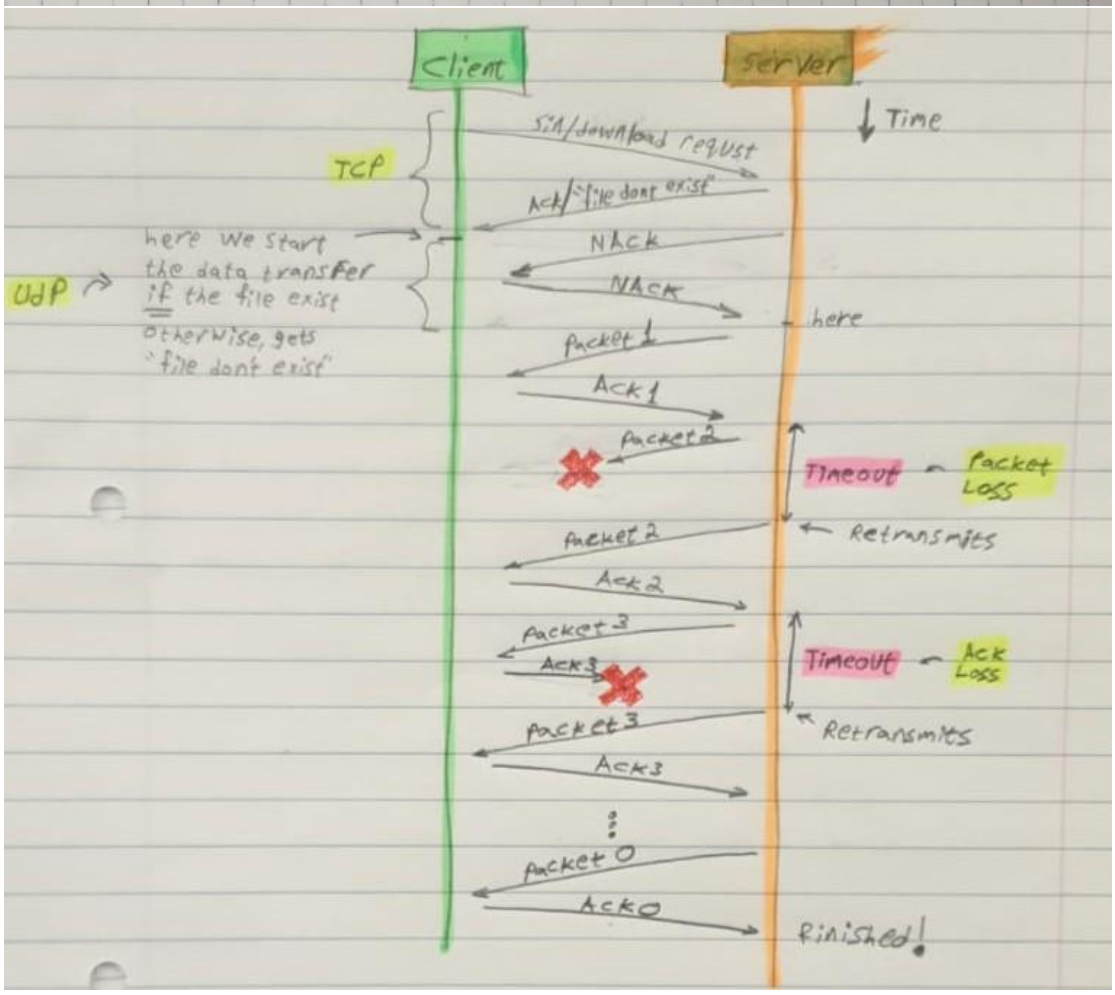
CHAT ROOM

Part B:

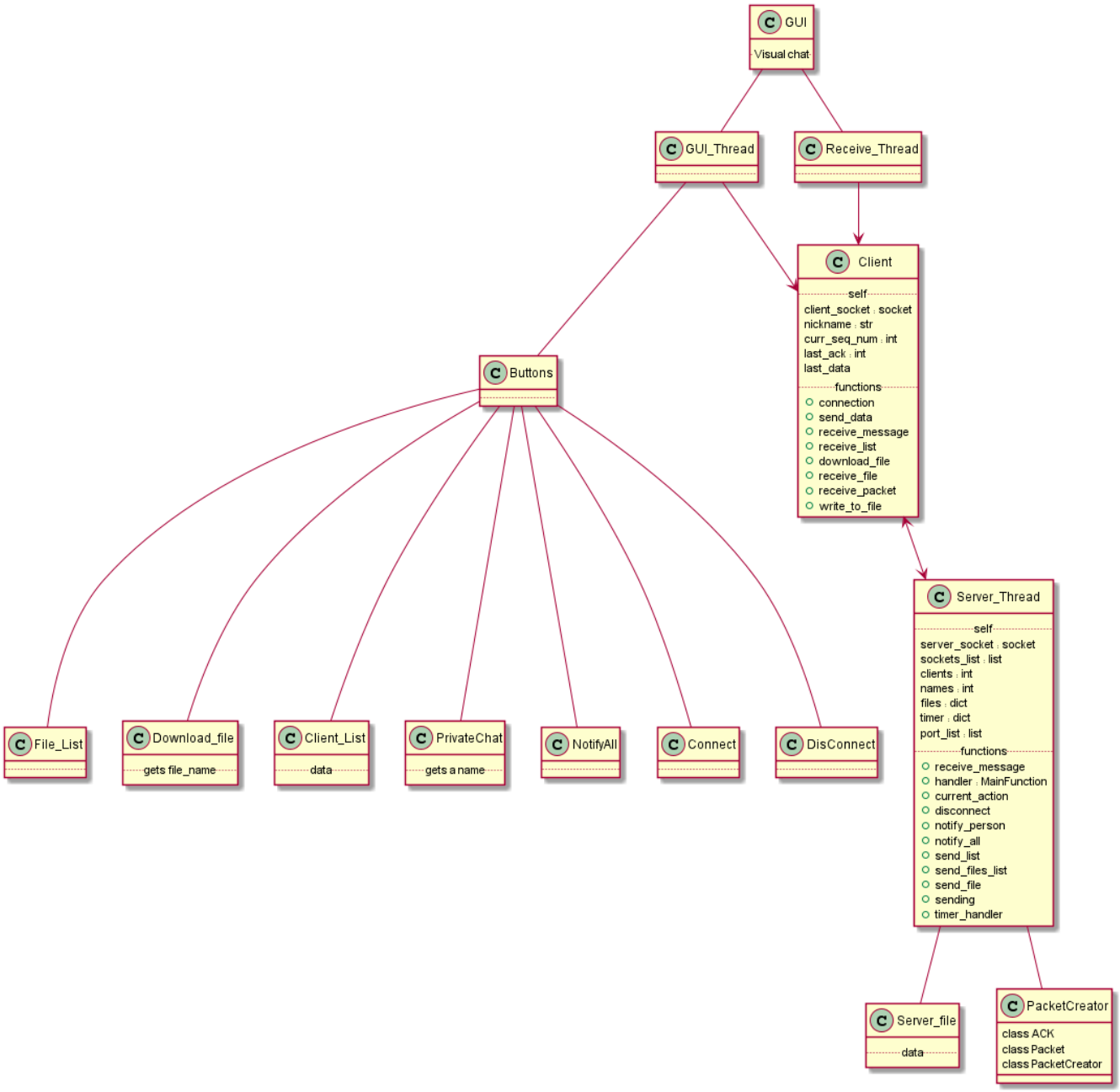
Our project is based on the Stop and Wait protocol. We implemented it in our way, but the fundamentals are the same.

Diagram of this protocol: (The first is from an old summery – stop and wait diagram, the second is our diagram**)**





CHAT ROOM



In addition, the above is some sort of UML for visualization..

CHAT ROOM

Second question:

What if the data packet has been lost?

After a fixed (constant) time (called **timeout**) timer expires, the sender will **assume that the data has been lost**, but the acknowledgment is lost! Every time we send a new packet the timer resets. By assuming it, the sender will **send the data packet again**. However, according to receiver it is a new data packet, hence it will give rise to duplicate packet problem. To eliminate duplicate packet situation sequence number is added to the data packet. Using this method, we can easily detect and deal with the duplicate packets.

The fixed timeout is extremely important. Its purpose is to stop the program from running forever waiting to receive the acknowledgment for a packet! Timer resets every time we send a packet for that exact reason. Weather the receiver didn't get the packet or the acknowledgment was lost, the sender will consider the packet as lost and will try to send it again with the same sequence number of course.

Third question:

Latency: (from the web)

Network latency, sometimes called lag, is the term used to describe delays in communication over a network. Latency meaning in networking is best thought of as the amount of time it takes for a packet of data to be captured, transmitted, processed through multiple devices, then received at its destination and decoded.

When delays in transmission are small, it's referred to as a low-latency network (desirable) and longer delays are called a high-latency network (not so desirable).

Long delays that occur in high-latency networks create bottlenecks in communication. In the worst cases, it's like traffic on a four-lane highway trying to merge into a single lane. High latency decreases communication bandwidth, and can be temporary or permanent, depending on the source of the delays.

Latency is measured in milliseconds, or during speed tests, it's referred to as a ping rate(1. Obviously, zero to low latency in communication is what we all want. However, standard latency for a network is explained slightly differently in various contexts, and latency issues also vary from one network to another.

Answer: Our project is based on the stop and wait protocol, we implemented in our own way, a bit different but the fundamentals are the same. The program does not deal with latency problem head high but, the program does notify us if a packet out reached its deadline(timeout) which means there were a transfer problem, or the latency is high and the therefore the packet didn't reach in time and a warning is printed ("Timeout").
Moreover, the timer makes sure the program will not run infinitely. It will not wait for a packet forever nor will transfer an "infinite file", the socket has a timeout as well.

CHAT ROOM

<u>Question 1:</u> We added another word file just for this question. We didn't have enough time to rewrite it to English. Sorry for the inconvenience.

<u>question 2:</u> (CRC - cyclic redundancy check**)**

(CRC) is a popular error-detecting code.
Its purpose is to detect accidental changes to raw data, specifically designed to protect against common types of errors on communication channels. CRCs encodes messages by adding a fixed length check value. Blocks of data entering systems get a short check value (codeword) attached. On retrieval, the calculation is repeated (checksum...) and if the values do not match then the block contains a data error. Therefore, need to be corrected or sent again. (CRCs can be used for error correction).

Specification of a CRC code requires definition of a so-called polynomial generator. This polynomial becomes the divisor in a polynomial long division, the remainder of that division becomes the result. The polynomial coefficients are calculated according to the arithmetic of a finite field, so the addition operation can always be performed bitwise-parallel (there is no carry between digits)

Its popularity comes form the fact that it's very easy to implement and analyze. Moreover, it particularly well suited for the detection of common errors in transmission. However, they are not suitable for protecting against intentional alteration of data.

The CRC called this way because the information it adds to the message is redundant, its sole purpose to verify the data is not corrupted. After the check its redundant.

Example of CRC calculation:

Given generated polynomial with a degree R and message M:
1. Adding R zeros to the right of the message.
2. Do long division (mod 2)
3. Subtract the reminder with XOR instead of regular subtraction.
4. Adding the result to the right of the message.
5. Sending the message

The receiver will do 1 & 2 and make sure that the R last bytes are identical to the result.

<u>Question 3:</u>

HTTP 1.0:

It is a generic, stateless protocol that can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes, and headers.

Usernames and passwords are not encrypted, leaving them susceptible to spying. Moreover, there are no time constraints. any information obtained by spying can be used for a long time after it was obtained. Each appeal needs to pay for the expense of establishing a new TCP connection (Some version1.0 implementations utilized a “keep-alive”)

The server must reply in the same sequence as the matching request for a given connection. However, the client does not have to wait for the response to its previous request before submitting another over the same connection (pipelining). It reduces network round-trip latency while also optimizing the TCP protocol's capabilities

HTTP 1.1:

The latest version of Hypertext Transfer Protocol (HTTP), protocol that runs on top of the Internet's TCP / IP suite of protocols. HTTP 1.1 provides faster delivery of Web pages than the original HTTP and reduces Web traffic.

Connection of TCP should be **"kept alive"** unless explicitly requested otherwise. It allows the client to submit numerous appeals over the same connection without having to wait for each one to be answered, **significantly increasing performance over HTTP 1.0.**

The number of competing TCP connections between a client and a server is limited, and **each new connection consumes a lot of resources**.

Flow control in HTTP / 1.1 is based on TCP.

HTTP 2.0:

Motivation: reduce web page load latency by using techniques such as compression, multiplexing, and prioritization.

A major revision of the HTTP network protocol used by the World Wide Web. It was derived from the earlier experimental SPDY protocol, originally developed by Google.

The difference between HTTP/1.1 and HTTP/2 is the binary framing layer.

HTTP 1.1 keeps all requests and responses in plain text format. However, HTTP 2 uses the binary framing layer to encapsulate all messages in binary format, while still maintaining HTTP semantics. This ensures that web applications created before HTTP 2 can continue functioning as normal when interacting with the new protocol.

QUIC:

It's a transport network protocol.

Motivation: QUIC aims to be nearly equivalent to a TCP connection but with much-reduced latency. Think of QUIC as TCP + TLS + HTTP / 2 implemented in UDP.

QUIC greatly reduces overhead during connection setup. When a client opens a connection, the response packet includes the data needed for future packets to use encryption (Combining multiple steps into a single request-response).

Use UDP rather than TCP as its basis, which does not include loss recovery. If an error occurs in one stream, the protocol stack can continue servicing other streams independently.

CHAT ROOM

Packets are encrypted individually.
improves performance during network-switch events.

		Summary
http 1.0	http 1.1	HTTP 1.0 is mostly in the header, it has some bandwidth waste, each server needs to bind a distinct IP address, only one request and answer for each TCP connection, support caching to serve websites quicker. However, HTTP 1.1 use more sophisticated cache management approach, less bandwidth waste, enables connection reuse, provides persistent connections, which implies that many responses and requests can be sent over the same HTTP connection.
http 1.0	http 2.0	HTTP 1.0 is mostly in the header, it has some bandwidth waste, each server needs to bind a distinct IP address, only one request and answer for each TCP connection, support caching to serve websites quicker. While HTTP2 avoids network delay by using multiplexing and **much faster and more reliable** HTTP is a network delay sensitive protocol in the sense that if there is less network delay, then the page loads faster.
http 1.0 1.1 2.0	QUIC	The biggest difference between QUIC and HTTP is that **QUIC runs on top of UDP** whereas HTTP runs on top of TCP. QUIC moves the TCP reliability, congestion control, and in order delivery to the application layer thus giving it more control and allowing for changes to be applied easier. QUIC has an error detector, it encrypts the data always, there is a flow control. Connect a bit faster QUIC creates multiple independent streams over its single UDP connection. If one will be loss, it doesn't cause the other streams to become blocked unlike HTTP. QUIC also reduces the connection setup time by introducing a 1 RTT handshake unlike TCP with HTTP uses. which relies on a 3 RTT handshake.
http 1.1	http 2.0	Multiplexing: HTTP 1.1 **loads resources** one after the other, so if one resource cannot be loaded, it blocks all the other resources behind it. In contrast, HTTP 2 can use a single TCP connection to send multiple streams of data at once so that no one resource blocks any other resource. Moreover, HTTP 2 uses the binary framing layer to encapsulate all messages in binary format, while still maintaining HTTP semantics. This conversion of messages into binary allows HTTP/2 to try new approaches to data delivery not available in HTTP/1.1, a contrast that is at the root of the practical differences between the two protocols.

<u>Question 4:</u>
A port is simply an "inside virtual address", with it the transport layer knows which application it needs to serve. Its a channel of communication which is numbered between 1 and 65000. All network devices use ports, and most can change them when needed.
They were originally created to allow multiple programs to use the same IP address.

CHAT ROOM

The most common reason for needing to use ports is for remote access.
It allows us to access more then one "application" simultaneously with a single IP amazing!!
One would only need to use separate ports to do so.

Question 5:

A subnet, or subnetwork, is a network inside a network.
Motivation:
If a network has millions of connected devices, it can take a lot of time before the data will "find" the right device. Therefore, **subnetting** comes in handy, subnetting narrows the IP address to usage within a range of devices.

Networks are more efficient when messages travel as directly as possible. When a network receives data packets from another network, it will sort and route those packets by subnet so that the packets do not take an inefficient route to their destination. Subnet allows us to do so. Subnets make networks more efficient. Through subnetting, network traffic can travel a shorter distance without passing through unnecessary routers to reach its destination. Routers within a network use something called a **subnet mask** to sort data into subnetworks. Routers use subnet masks to route data packets to the right place.

Example, suppose an IP packet is addressed to the IP address 192.0.8.12. The network is identified by "192.0.8" (192.0.8.0/24). Network routers forward the packet to a host on the network indicated by "192.0.8." When it arrives at that network, a router within the network consults its routing table. It does some binary mathematics using its subnet mask of 255.255.255.0, sees the device address "12" (the rest of the IP address indicates the network), and calculates which subnet the packet should go to. It forwards the packet to the router or switch responsible for delivering packets within that subnet, and the packet arrives at IP address 192.0.8.12

Question 6:

Why each computer has both an IP Address and a MAC Address assigned to it?

Motivation: If to communicate with each other every computer needs to be physically connected to the other, all networks either would become very complicated quickly or they would stay small. To get around this physical limitation, they separated computer communication to also contain a routable component, the IP Address.

For instance, let's suppose we have two groups of computers A and B as well. In the same group everyone knows each other. Let's also suppose only one couple <x from A, y from B> knows each other as well. If so, every computer can communicate with any other from these groups, simply by forwarding the message to the one who has the connection to the other group, and he will do the rest.
Now, not every computer can send to multiple groups of people, but the ones that can are known as **ROUTERS**.

Now, Computer x could potentially learn the IP Address of Computer y. **However, MAC Addresses** are physical and are NOT routable. So, Computer x could not really learn the MAC Address of Computer y. That's why computers have both MAC and IP Addresses. MAC Addresses handle the physical connection from computer to computer, while IP Addresses

handle the logical routable connection from both computer to computer AND network to network.

Question 7:

Router and Switch are networks connecting devices.
Router works at network layer and is responsible to find the shortest path for a packet.
They connect devices across multiple networks.
Switch connects various devices in a network.

Differences between Router and Switch:

Router main objective is to connect various networks. It works in the Network Layer, sends data in form of packets, follows duplex mode of transmission. It has Less collision. Compatible with NAT and its type is Adaptive and Non-adaptive routing.
However, **switch** main objective is to connect various devices in a network. It works in the Data Link Layer. Switch sends data in form of packets and frames. It also follows duplex mode of transmission, in full duplex mode, no collision happens in switch too. It's not compatible with NAT and its type is Circuit, Packet and Message switching.

Question 8:

convert to ipv6:

By using IPv6 together with IPv4, as IPv6 uses a 128-bit address, allowing 2 ^ 128, or approximately 3.4 × 10 ^ 38 addresses, more than 7.9 × 10 ^ 28 times as many as IPv4, which uses 32-bit addresses and provides approximately 4.3 billion addresses.

Network Address Translation (NAT)

NAT or IP masquerading allows ISPs and companies to use special private IP addresses to connect computer networks to the Internet using a single public IP address. This way, they can use an IPv4 for an entire private network instead of an IP address per network device. NAT has also been key for slowing down IPv4 exhaustion

Classless Inter-Domain Routing (CIDR):

In the CIDR notation, addresses are written with a suffix, introduced by a slash, that indicates the number of bits of the prefix. For instance, the suffix / 16 means that the first 16 bits out of the 32 bits of an IPv4 address are defined by the network and the other 16 bits left are defined by the host. This method divided the IP address space for IPv4 into five classes (from A to E), based on the leading 4 bits of the address.

Question 9:

(wiki)

CHAT ROOM

OSPF: Open Shortest Path First (OSPF) is a routing protocol for Internet Protocol (IP) networks. It uses a link state routing (LSR) algorithm and falls into the group of interior gateway protocols (IGPs), operating within a single autonomous system (AS).

OSPF gathers link state information from available routers and constructs a topology map of the network. The topology is presented as a routing table to the Internet Layer for routing packets by their destination IP address. OSPF supports Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6) networks and supports the Classless Inter-Domain Routing (CIDR) addressing model.

OSPF is widely used in large enterprise networks. IS-IS, another LSR-based protocol, is more common in large service provider networks.

RIP:

The Routing Information Protocol (RIP) is one of the oldest distance-vector routing protocols which employs the hop count as a routing metric. RIP prevents routing loops by implementing a limit on the number of hops allowed in a path from source to destination. The largest number of hops allowed for RIP is 15, which limits the size of networks that RIP can support.

RIP implements the split horizon, route poisoning, and hold down mechanisms to prevent incorrect routing information from being propagated.

In RIPv1 routers broadcast updates with their routing table every 30 seconds. In the early deployments, routing tables were small enough that the traffic was not significant. As networks grew in size, however, it became evident there could be a massive traffic burst every 30 seconds, even if the routers had been initialized at random times.

In most networking environments, RIP is not the preferred choice of routing protocol, as its time to converge and scalability are poor compared to EIGRP, OSPF, or IS-IS. However, it is easy to configure, because RIP does not require any parameters, unlike other protocols.

BGP:

Border Gateway Protocol (BGP) is a standardized exterior gateway protocol designed to exchange routing and reachability information among autonomous systems (AS) on the Internet. [2] BGP is classified as a path-vector routing protocol, [3] and it makes routing decisions based on paths, network policies, or rulesets configured by a network administrator.

BGP used for routing within an autonomous system is called Interior Border Gateway Protocol, Internal BGP (**iBGP**). In contrast, the Internet application of the protocol is called Exterior Border Gateway Protocol, External BGP (**eBGP**).

BGP neighbors, called peers, are established by manual configuration among routers to create a TCP session on port 179. A BGP speaker sends 19-byte keep-alive messages every 30 seconds (protocol default value, tunable) to maintain the connection. Among routing protocols, BGP is unique in using TCP as its transport protocol.

When BGP runs between two peers in the same autonomous system (AS), it is referred to as Internal BGP (iBGP or Interior Border Gateway Protocol). When it runs between different autonomous systems, it is called External BGP (eBGP or Exterior Border Gateway Protocol).

CHAT ROOM

Routers on the boundary of one AS exchanging information with another AS are called border or edge routers or simply eBGP peers and are typically connected directly, while iBGP peers can be interconnected through other intermediate routers. Other deployment topologies are also possible, such as running eBGP peering inside a VPN tunnel, allowing two remote sites to exchange routing information in a secure and isolated manner.

The main difference between iBGP and eBGP peering is in the way routes that were received from one peer are typically propagated by default to other peers:

New routes learned from an eBGP peer are re-advertised to all iBGP and eBGP peers.

New routes learned from an iBGP peer are re-advertised to all eBGP peers only.

These route-propagation rules effectively require that all iBGP peers inside an AS are interconnected in a full mesh with iBGP sessions.

How routes are propagated can be controlled in detail via the route-maps mechanism. This mechanism consists of a set of rules. Each rule describes, for routes matching some given criteria, what action should be taken. The action could be to drop the route, or it could be to modify some attributes of the route before inserting it into the routing table.

iBGP: A routing protocol between two BGP routers within the same AS. The purpose of the protocol is to provide information to internal routers on the network, such as a list of IP addresses that are outside the AS.

Answers:

e. The c3 network router learns about the x subnet through the eBGP protocol, trivial because it is a router between two BGP routers belonging to different but neighboring Ass.

f. Network router a3 learns about subnet X through iBGP protocol i.e., internal BGP because

it is a router between two BGP routers belonging to the same AS Subnet (3c,3a). a3 belongs to AS3 which operates with OSPF protocol which means it will choose the lowest cost to its destination. Unfortunately, all its neighbors operate with RIP protocol so he will not exchange the routing tables in the same way he would with an OSPF neighbor. But once one of the AS3 parties learns from a different subnet it will notify its iBGP and eBGP neighbors.

g. Network router 1c learns about subnet X through the eBGP protocol because it is a router between two BGP routers belonging to different but neighboring Ass. C1 is in a RIP meaning it will choose the current closest neighbor (by distance not cost) therefore, AS3. Hence EBGP.

h. The c2 network router learns about the x subnet through the iBGP protocol, because of the distance.