Computer Networking

-

Final Project

Alon Barak

213487598

Idan Philosoph

324076066

Part B:

In this part we will analyze the system we've created so far.

In the section below we explain how the system overcome packet loss:

Our system is using two protocols for different missions.

In case the client wants to connect to the server to just to send a broadcast/private message in the chat, the protocol that being used is TCP protocol. In that case, if a packet loss occurs then the Sender resend the packet and slows down sending data into the network by shrinking down the window size.

In case a client asks for a file download, the process is being handled by a UDP protocol. But in this case, we implement a RDT over UDP.

Therefore, if the sender notices a packet loss, he sends n packets starting from the first packet that didn’t receive an ack.
the value of n is determined by the congestion control algorithm we implemented.

For example, if the Server sends 5 packets to the server and receives an Ack until the 3rd packet then the server will re-send the 4th and 5th and so on in the next batch of packets.

With these methods our system is handling packet loss.

In the section below we will explain the system algorithm for latency issues:

We implemented a go-back-n RDT algorithm, the way the algorithm handles latency is by resending n packets starting one after the last packet that received a duplicate ack or if all the acks were received correctly, we will transmit n new packets.

On the client's side, the way we handle latency is by sending only the ack of the last packet received in order. And the server will start transmitting from one after the first duplicate ack.

So as an example, if the server sends the first 5 packets, but they were received by the client in the order 1, 2, 4, 3, 5, the client will send the acks in the following order: 1, 2, 2, 2, 2. And the server will realize that the client didn't receive packet 3 (at least) and transmit the next n packets starting from packet 3.

In the section below we will explain the system algorithm for congestion control:

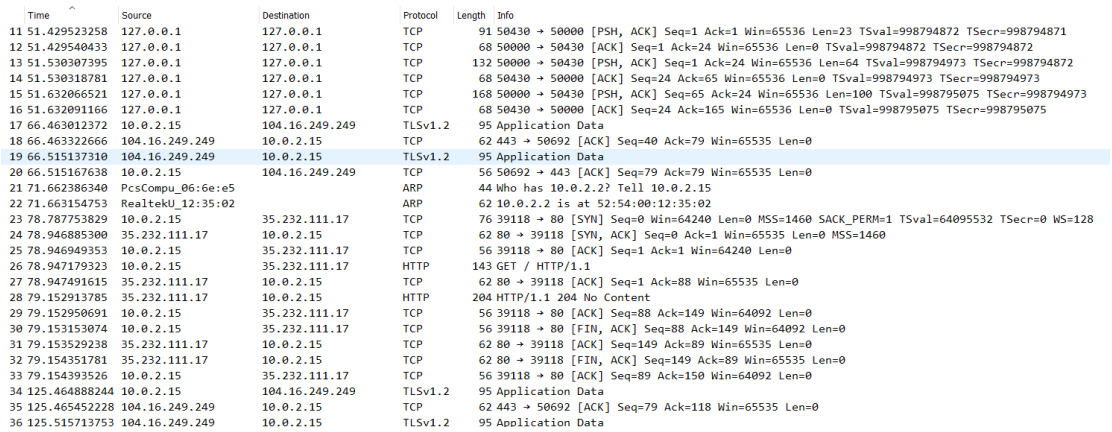
In our system we've implemented something similar to Cubic congestion control algorithm.

It will start slow and will increase the amount of packets being sent each round if the network is stable.

In case of a packet loss, the system will increase the window size in a linear growth instead of exponential as it was in the beginning.

In case of a triple duplicate Ack the algorithm cut the window size in half.

In case of a Timeout, the window size is being set back to one and start the algorithm all over again. (Expo. Growth)



	Time	Source	Destination	Protocol	Length	Info
11	51.429523258	127.0.0.1	127.0.0.1	TCP	91	50430 → 50000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=23 TSval=998794872 TSecr=998794871
12	51.429540433	127.0.0.1	127.0.0.1	TCP	68	50000 → 50430 [ACK] Seq=1 Ack=24 Win=65536 Len=0 TSval=998794872 TSecr=998794872
13	51.530307395	127.0.0.1	127.0.0.1	TCP	132	50000 → 50430 [PSH, ACK] Seq=1 Ack=24 Win=65536 Len=64 TSval=998794973 TSecr=998794872
14	51.530318781	127.0.0.1	127.0.0.1	TCP	68	50430 → 50000 [ACK] Seq=24 Ack=65 Win=65536 Len=0 TSval=998794973 TSecr=998794973
15	51.632066521	127.0.0.1	127.0.0.1	TCP	168	50000 → 50430 [PSH, ACK] Seq=65 Ack=24 Win=65536 Len=100 TSval=998795075 TSecr=998794973
16	51.632091166	127.0.0.1	127.0.0.1	TCP	68	50430 → 50000 [ACK] Seq=24 Ack=165 Win=65536 Len=0 TSval=998795075 TSecr=998795075
17	66.463012372	10.0.2.15	104.16.249.249	TLSv1.2	95	Application Data
18	66.463322666	104.16.249.249	10.0.2.15	TCP	62	443 → 50692 [ACK] Seq=40 Ack=79 Win=65535 Len=0
19	66.515137310	104.16.249.249	10.0.2.15	TLSv1.2	95	Application Data
20	66.515167638	10.0.2.15	104.16.249.249	TCP	56	50692 → 443 [ACK] Seq=79 Ack=79 Win=65535 Len=0
21	71.662386340	PcsCompu_06:6e:e5		ARP	44	Who has 10.0.2.2? Tell 10.0.2.15
22	71.663154753	RealtekU_12:35:02		ARP	62	10.0.2.2 is at 52:54:00:12:35:02
23	78.787753829	10.0.2.15	35.232.111.17	TCP	76	39118 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=64095532 TSecr=0 WS=128
24	78.946885300	35.232.111.17	10.0.2.15	TCP	62	80 → 39118 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
25	78.946949353	10.0.2.15	35.232.111.17	TCP	56	39118 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
26	78.947179323	10.0.2.15	35.232.111.17	HTTP	143	GET / HTTP/1.1
27	78.947491615	35.232.111.17	10.0.2.15	TCP	62	80 → 39118 [ACK] Seq=1 Ack=88 Win=65535 Len=0
28	79.152913785	35.232.111.17	10.0.2.15	HTTP	204	HTTP/1.1 204 No Content
29	79.152950691	10.0.2.15	35.232.111.17	TCP	56	39118 → 80 [ACK] Seq=88 Ack=149 Win=64092 Len=0
30	79.153153074	10.0.2.15	35.232.111.17	TCP	56	39118 → 80 [FIN, ACK] Seq=88 Ack=149 Win=64092 Len=0
31	79.153529238	35.232.111.17	10.0.2.15	TCP	62	80 → 39118 [ACK] Seq=149 Ack=89 Win=65535 Len=0
32	79.154351781	35.232.111.17	10.0.2.15	TCP	62	80 → 39118 [FIN, ACK] Seq=149 Ack=89 Win=65535 Len=0
33	79.154393526	10.0.2.15	35.232.111.17	TCP	56	39118 → 80 [ACK] Seq=89 Ack=150 Win=64092 Len=0
34	125.464888244	10.0.2.15	104.16.249.249	TLSv1.2	95	Application Data
35	125.465452228	104.16.249.249	10.0.2.15	TCP	62	443 → 50692 [ACK] Seq=79 Ack=118 Win=65535 Len=0
36	125.515713753	104.16.249.249	10.0.2.15	TLSv1.2	95	Application Data

In the first line the client sends a connection request to the server.

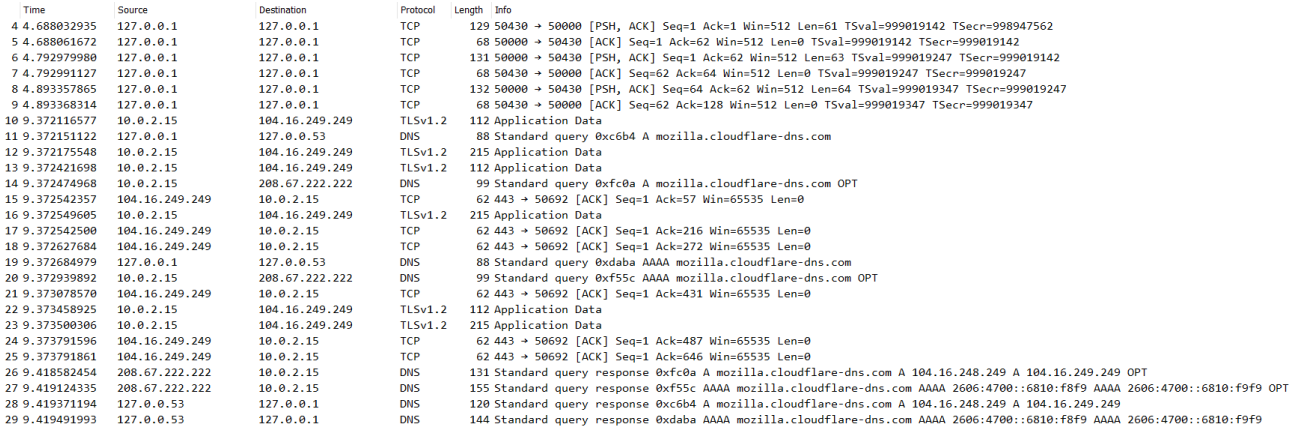
In the second line the server sends an ack to the client.

In the third line the server sends a connection response with the content “True” to the client.

In the fourth line the client sends an ack.

In the fifth line the server sends a broadcast message to all the users (which is only the one who just joined) stating that a new user has joined.

In the sixth line the client sends an ack to the server.



	Time	Source	Destination	Protocol	Length	Info
4	4.688032935	127.0.0.1	127.0.0.1	TCP	129	50430 → 50000 [PSH, ACK] Seq=1 Ack=1 Win=512 Len=61 TSval=999019142 TSecr=998947562
5	4.688061672	127.0.0.1	127.0.0.1	TCP	68	50000 → 50430 [ACK] Seq=1 Ack=62 Win=512 Len=0 TSval=999019142 TSecr=999019142
6	4.792979980	127.0.0.1	127.0.0.1	TCP	131	50000 → 50430 [PSH, ACK] Seq=1 Ack=62 Win=512 Len=63 TSval=999019247 TSecr=999019142
7	4.792991127	127.0.0.1	127.0.0.1	TCP	68	50430 → 50000 [ACK] Seq=62 Ack=64 Win=512 Len=0 TSval=999019247 TSecr=999019247
8	4.893357865	127.0.0.1	127.0.0.1	TCP	132	50000 → 50430 [PSH, ACK] Seq=64 Ack=62 Win=512 Len=64 TSval=999019347 TSecr=999019247
9	4.893368314	127.0.0.1	127.0.0.1	TCP	68	50430 → 50000 [ACK] Seq=62 Ack=128 Win=512 Len=0 TSval=999019347 TSecr=999019347
10	9.372116577	10.0.2.15	104.16.249.249	TLSv1.2	112	Application Data
11	9.372151122	127.0.0.1	127.0.0.53	DNS	88	Standard query 0xc6b4 A mozilla.cloudflare-dns.com
12	9.372175548	10.0.2.15	104.16.249.249	TLSv1.2	215	Application Data
13	9.372421698	10.0.2.15	104.16.249.249	TLSv1.2	112	Application Data
14	9.372474968	10.0.2.15	208.67.222.222	DNS	99	Standard query 0xfc0a A mozilla.cloudflare-dns.com OPT
15	9.372542357	104.16.249.249	10.0.2.15	TCP	62	443 → 50692 [ACK] Seq=1 Ack=57 Win=65535 Len=0
16	9.372549605	10.0.2.15	104.16.249.249	TLSv1.2	215	Application Data
17	9.372542500	104.16.249.249	10.0.2.15	TCP	62	443 → 50692 [ACK] Seq=1 Ack=216 Win=65535 Len=0
18	9.372627684	104.16.249.249	10.0.2.15	TCP	62	443 → 50692 [ACK] Seq=1 Ack=272 Win=65535 Len=0
19	9.372684979	127.0.0.1	127.0.0.53	DNS	88	Standard query 0xdaba AAAA mozilla.cloudflare-dns.com
20	9.372939892	10.0.2.15	208.67.222.222	DNS	99	Standard query 0xf55c AAAA mozilla.cloudflare-dns.com OPT
21	9.373078570	104.16.249.249	10.0.2.15	TCP	62	443 → 50692 [ACK] Seq=1 Ack=431 Win=65535 Len=0
22	9.373458925	10.0.2.15	104.16.249.249	TLSv1.2	112	Application Data
23	9.373500306	10.0.2.15	104.16.249.249	TLSv1.2	215	Application Data
24	9.373791596	104.16.249.249	10.0.2.15	TCP	62	443 → 50692 [ACK] Seq=1 Ack=487 Win=65535 Len=0
25	9.373791861	104.16.249.249	10.0.2.15	TCP	62	443 → 50692 [ACK] Seq=1 Ack=646 Win=65535 Len=0
26	9.418582454	208.67.222.222	10.0.2.15	DNS	131	Standard query response 0xfc0a A mozilla.cloudflare-dns.com A 104.16.248.249 A 104.16.249.249 OPT
27	9.419124335	208.67.222.222	10.0.2.15	DNS	155	Standard query response 0xf55c AAAA mozilla.cloudflare-dns.com AAAA 2606:4700::6810:f8f9 AAAA 2606:4700::6810:f9f9 OPT
28	9.419371194	127.0.0.53	127.0.0.1	DNS	120	Standard query response 0xc6b4 A mozilla.cloudflare-dns.com A 104.16.248.249 A 104.16.249.249
29	9.419491993	127.0.0.53	127.0.0.1	DNS	144	Standard query response 0xdaba AAAA mozilla.cloudflare-dns.com AAAA 2606:4700::6810:f8f9 AAAA 2606:4700::6810:f9f9

In the first line a used named “Alon” sent a broadcast message to all the clients saying “Hello everyone!”.

In the second line the server sent an ack back to Alon.

In the third line the server sent Alon the message that he sent as the broadcast message as he is the only user that is connected to the server at the time, he is the only one to receive it.

In the fourth line the client sent an ack back to the server.

In the fifth line the server sends the word True back to the client, meaning the message was successfully sent to the recipient.

And in the sixth line the client sends an ack to the server.

In both screenshots you are unable to see the content of the package as it is not written in the wireshark table, but you can see both the captures and a few more in the github page under “wireshark captures” and open each packet for yourself.

Part C

Q1:

Host joins the Network:

1. Host broadcast DHCP discover message.
   Src: 0.0.0.0 , 67, MAC:
   Dest: 172.16.0.235 , 68, MAC:
   Protocol: UDP
2. DHCP server response with DHCP offer message.
   Src: 172.16.0.235 , 68 , MAC:
   Dest: 0.0.0.0, 67, MAC:
   Protocol: UDP
3. Host request UP address: DHCP request message.
   Src: 0.0.0.0, 67, MAC:
   Dest: 172.16.0.235, 68, MAC:
   Protocol: UDP
4. DHCP sends IP address: DHCP Ack message.
   Src; 172.16.0.235, 68, MAC:
   Dest: 0.0.0.0, 67,MAC:
   Protocol: UDP

Client joins the Server:

1. Client sends a Connect request to the Server.
   Src: 192.168.56.1, 55147, MAC:
   Dest: 192.168.56.1, 55000, MAC:
   Protocol: TCP
2. Server response with an accept response.
   Src: 192.168.56.1, 55000, MAC:
   Dest: 192.168.56.1, 55147, MAC:
   Protocol: TCP

3. Client sends a 'Login' request to the Server via the TCP Socket.
   Src: 192.168.56.1, 55147, MAC:
   Dest: 192.168.56.1, 55000, MAC:
   Protocol: TCP
4. Server sends a 'Login' response back to the Client.
   Src: 192.168.56.1, 55000, MAC:
   Dest: 192.168.56.1, 55147, MAC:
   Protocol: TCP

Server inform other Clients about the new Client:

1. Server sends a Broadcast message about the new Client.
   Server actually sends private message to each Client.
   Src: 192.168.56.1, 55000, MAC:
   Dest: all clients connected to the server.
   Protocol: TCP.

Q2:

CRC (cyclic redundancy check) is an error detecting code used to determine if a certain packet/block of data has been corrupted. Its commonly used in network protocols such as TCP/IP for data verification reasons and more. The CRC algorithm computes a checksum for each specific data set that is sent through it. This algorithm utilizes a polynomial key and the transferred data.

Q3:

In this question we will present the differences between http 1.0, http 1.1, http 2.0 and QUIC.

First, we will present the different versions of HTTP and their differences, later we will compare the three with the QUIC protocol.

HTTP/1.1 builds on top of HTTP/1.0 with the addition of persistent connections and parallelization. Persistent connections allow a client to perform multiple consecutive requests over a single connection rather than having to create a new TCP connection for every request.

Parallelization adds the ability to create multiple TCP connections to a webserver and simultaneously request resources on each of these connections.

The features introduced in HTTP/1.1 are beneficial, but the protocol still has some drawbacks including increased network congestion due to multiple independent TCP connections being created, the inability to carry more than 1 HTTP request or response in a TCP segment, and the inability to compress HTTP headers.

HTTP/2 addresses the problems of HTTP/1.1 and introduces some new features as well. HTTP/2 relies on multiplexing which allows multiple requests and response to be sent over a single TCP connection. This means HTTP/2 only needs to open a single connection with the webserver which addresses the congestion issue of HTTP/1.1. Additionally, HTTP/2 adds priorities to requests in order to give them more bandwidth within the connection.

A downside to the implementation of multiplexing over a single TCP connection is the issue of head of line blocking. TCP relies on in order delivery, so if a TCP segment is lost then the whole TCP connection will be held up waiting for the missing segment. This means all requests on the HTTP/2 connection will be blocked until the lost segment is retransmitted which can add request latency. HTTP/2 also adds the ability for the server to push data to the client and introduces the compression of HTTP headers making messages more compact. The idea with server push is the server can anticipate what the client is going to request and start sending it ahead of time. This isn't possible in HTTP/1.1 as only the client can initiate the transfer of data. The final change

HTTP/2 introduces is the connection must be over SSL which is an added security benefit.

QUIC is a new application layer protocol proposed by Google to be used for transferring web pages. QUIC is a novel approach and is starting from a fresh slate unlike HTTP.

The biggest difference between QUIC and HTTP is 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. For example, one of the features of QUIC is its congestion control algorithm is pluggable thus you can choose whichever congestion control algorithm you like without having to change the underlying network stack. QUIC offers the same multiplexing capabilities as HTTP/2 without the issue of head of line blocking. QUIC creates multiple independent streams over its single UDP connection and a loss on one of the streams doesn't cause the other streams to become blocked. QUIC also reduces the connection setup time by introducing a 1 RTT handshake unlike TCP which relies on a 3 RTT handshake.

Q4:

The simplest explanation is that ports identify what process on the host the received traffic should be sent to. A computer can have multiple simultaneous connections, all receiving data for different processes (mail, web, database) on that computer.

When the computer receives data, the port information allows it to give the data to the correct process. For example, data with port 80 should go to the http process. Data with port 25 should go to the mail process, and so on.

In other words, the IP address identifies the computer host, and the port number specifies what specific process is running on that host. It's like a street address identifies a building, but the room number identifies a particular apartment or office.

Q5:

A subnet is a network inside a network. Subnets make networks more efficient. Through subnetting, network traffic can travel a shorter distance without passing through unnecessary routers to reach its destination.

In a network, there could be millions of connected devices, and it could take some time for the data to find the right device. This is where subnetting comes in handy. subnetting narrows down the IP address to usage within a range of devices and shorten the path for a specific packet because of it.

Q6:

MAC address is a global unique address for a specific device.

MAC address is useful and efficient for local communication. unlike the MAC address, the IP address is flexible, and changes based on the network the computer is in given moment.

MAC addresses and IP addresses operate on different layers of the Internet protocol suite. MAC addresses are used to identify machines within the same broadcast network on layer 2, while IP addresses are used on layer 3 to identify machines throughout different networks.

Even if your computer has an IP address, it still needs a MAC address to find other machines on the same network (especially the router/gateway to the rest of the network/internet), since every layer is using underlying layers.

Q7:

First, we will discuss the differences between a Router and a Switch.

A Router performs in the Network layer and its job is to connect various networks, while the Switch is on the Link layer and its job is to connect various devices on a network. Router store IP address in the

routing table whereas Switch store MAC address in a lookup table. Routers supports NAT while Switches does not.

Now we will discuss the difference between NAT and the two above.

Network Address Translation (NAT) is a process in which one or more local IP address is translated into one or more Global IP address and vice versa in order to provide Internet access to the local hosts. Also, it does the translation of port numbers in the packet that will be routed to the destination. It then makes the corresponding entries of IP address and port number in the NAT table. NAT generally operates on a router.

So basically, NAT is an algorithm that allow many hosts to use one global IP address instead of many addresses as usual. The main difference is that NAT is an algorithm and Routers and Switches are computers the allow communication between networks and devices. Another difference is that NAT is a platform that builds on a Router since its works on Layer 3 (the Network layer) and Switches works in Layer 2 (the Link layer).

To summarize, Routers and Switches are computers that allow communication and NAT is a platform that improves the communication.

Q8:

The most common ways to deal with the lake of IPv4 addresses is to use NAT.

When a Router is using NAT, its basically allow various hosts on a local network to be represented with only one global IPv4 address and generate for each host on the network a "local" IPv4 address that can only be recognize in the local network itself.

This platform saves a lot of IPv4 and reduce the need in a real and global IP address for each host.

Q9:

1. eBGP
2. iBGP
3. eBGP
4. iBGP