Network communication

Final exercise

by:
Amit Tubul 208710319
Keren Segev 207904244

System description:

- files included:
- client.py
- dhcp.py
- dns.py
- http_server.py
- whole_run_using_http_tcp.pcapng: is the wireshark capture if user chose 1
- whole_run_using_rudp_http.pcapng: is the wireshark capture if user chose 2
- explain_video.webm: is the desired video description

- running the program: to run this program we need four files:
    - client.py
    - dhcp.py
    - dns.py
    - http_server.py

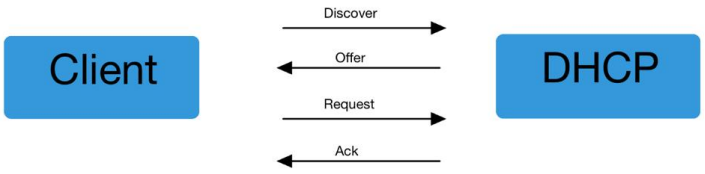
  steps:
    1. you need to open the terminal from the same path as all of the files are in.
    2. use “sudo python3 dhcp.py” command on the individual terminal.
    3. use “sudo python3 dns.py” command on the individual terminal.
    4. use “sudo python3 http_server.py” command on the individual terminal.
    5. use “sudo python3 client.py” command on the individual terminal.
    6. then, on the client terminal you will be asked to choose between TCP or RUDP

- what the program does:
  after the user will choose TCP or RUDP the program will execute the next steps:
    1. DHCP server connection to get an IP address for the client to communicate with the DNS and an IP for the DNS server.
    2. The client sends a query request to the DNS server and then the DNS server responds with the desired HTTP server address.
    3. The program will create a stable connection between the client and the server based on the user choice, then the client will send a request to the server and then it will respond with “redirect”.
    4. The client sends a DNS query with the new HTTP server address and then the client sends the same get request to the new HTTP server and in response the new server gives the desired data (in our case a creation of a “strong” password).

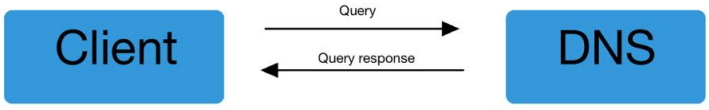
RUDP

Our RUDP is based on a normal UDP protocol with adjustments of reliability with syn, fin request and ack for each packet.

- state diagram:

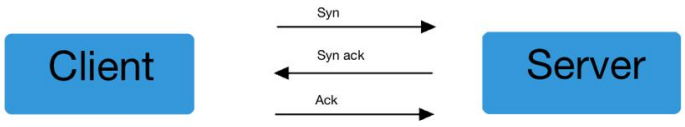



```
58.838193742   0.0.0.0          255.255.255.255    DHCP      286 DHCP Discover - Transaction ID 0x1
58.842429421   10.0.2.2         10.0.2.16          DHCP      590 DHCP Offer    - Transaction ID 0x1
59.877089635   10.0.0.10        255.255.255.255    DHCP      310 DHCP Offer    - Transaction ID 0x1
60.932347212   0.0.0.0          10.0.0.10          DHCP      286 DHCP Request  - Transaction ID 0x1
61.972930835   10.0.0.10        255.255.255.255    DHCP      310 DHCP ACK      - Transaction ID 0x1
```

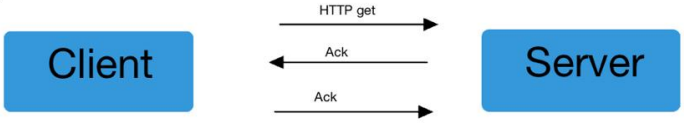



```
62.075306077  10.0.0.11          10.0.0.12          DNS        75 Standard query 0x0000 A http_server.com
62.163840326  10.0.0.12          10.0.0.11          DNS       106 Standard query response 0x0000 A http_server.com A 10.0.0.13
```

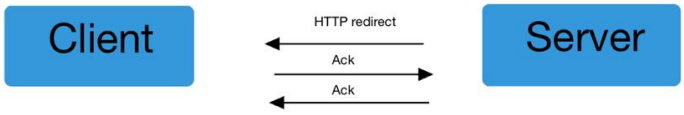
1.



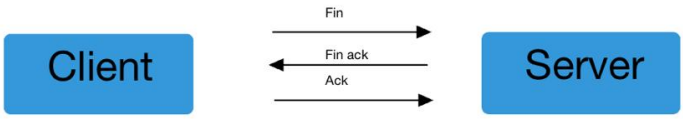
2.



3.



4.



*the application capturs at the end of the file.

- how the system handle packet loss:
- at the DHCP level if the client does not get an address it raises an error.
- at the DNS level if the client gets an empty packet it raises an error.
- at the Application level if the user chooses TCP the protocol already handles packet loss, if the user chooses RUDP we would want to raise an error after 5 seconds of no response (we didn't manage to finish this part of timings).

- how does the system handle latency issues?
  we would want to raise an error after 5 seconds of no response (we didn't manage to finish this part of timings).

Questions:

1. write at least 4 main differences between TCP protocol and QUIC
2. write at least 2 main differences between Cubic and Vegas
3. describe BGP protocol, what are the differences between BGP and OSPF and does it work by short paths?
4. fill the table based on your project code. describe how the messages will be affected if there is NAT between the user and the server and answer if you will use QUIC protocol.
5. describe the differences between ARP protocol and DNS.

Answers:

1.
   - TCP is a protocol that uses a reliable connection that ensures data delivery, QUIC is a UDP based protocol that provides a low-latency, encrypted, and reliable connection.
   - TCP uses a three-way handshake to get a connection between a client and a server, QUIC using a zero-round-trip connection that allows a client to send data to a server without waiting for the server response. This means that it reduces latency.
   - TCP uses slow-start congestion control which makes it work slower sometimes. QUIC uses a congestion control algorithm that works better in modern networks and can quickly adapt to changing network conditions.
   - Both TCP and QUIC can provide secure connections. However, QUIC was designed to improve the transport performance for encrypted traffic with faster session setup.

2.
   - Cubic assumes that there is a smooth curve function for that reason it works better with polynomial functions. On the other hand vegas randomly sampling points in the range and estimate the value of the function. For that reason

Vegas works better for high-dimensional integrals and for irregular functions or functions that are difficult to integrate using other methods.

- Cubic is a deterministic method. it will give the same results for the same input data. Vegas is a probabilistic method that can give different results each run because of the random sampling.

3. In BGP every router decides what path is best based on what it knows. In OSPF the routers share data to build network topology. OSPF is more used in LAN while BGP in WAN.
   BGP does not necessarily work by short paths, there are more factors it takes in mind such as network congestion and peering agreement.

4. *(point of view of the client)

Application	Port Src	Port Des	IP Src	IP Des	MAC Src	MAC Des
DHCP	68	67	0.0.0.0	255.255.255.255	02:42:02:8c:72:a8	ff:ff:ff:ff:ff:ff

Application	Port Src	Port Des	IP Src	IP Des	MAC Src	MAC Des
DNS	5000	53	10.0.0.11	10.0.0.12	02:42:02:8c:72:a8	02:42:02:8c:72:a8

Application	Port Src	Port Des	IP Src	IP Des	MAC Src	MAC Des
TCP	5005	80	10.0.2.15	10.0.2.15	02:42:02:8c:72:a8	02:42:02:8c:72:a8

Application	Port Src	Port Des	IP Src	IP Des	MAC Src	MAC Des
HTTP	5005	80	10.0.2.15	10.0.2.15	02:42:02:8c:72:a8	02:42:02:8c:72:a8

5. ARP(data link layer) and DNS(application layer) work on different layers. ARP protocol purpose is to map IP addresses to MAC addresses in LAN, using dynamic table to store the data for a limited time.
   DNS purpose is to map a domain name to an IP address.

wireshark captures:

if user choose TCP:

On rows 3-7 we can notice that there is connection and communication between the client and the DHCP server.

On row 10 there is the DNS query and on row 13 the server response.

On rows 17-37 there is the client-http_server connection (beside rows 25, 27 where there is DNS query and response).

3	1.690436806	0.0.0.0	255.255.255.255	DHCP	288	DHCP Discover - Transaction ID 0x1
4	1.690946473	10.0.2.2	10.0.2.16	DHCP	592	DHCP Offer - Transaction ID 0x1
5	1.817595991	10.0.0.10	255.255.255.255	DHCP	312	DHCP Offer - Transaction ID 0x1
6	2.849092719	0.0.0.0	10.0.0.10	DHCP	288	DHCP Request - Transaction ID 0x1
7	2.971273299	10.0.0.10	255.255.255.255	DHCP	312	DHCP ACK - Transaction ID 0x1
8	3.051114490	PcsCompu_e4:7d:89		ARP	44	Who has 10.0.2.2? Tell 10.0.2.15
9	3.051407635	RealtekU_12:35:02		ARP	62	10.0.2.2 is at 52:54:00:12:35:02
10	3.075349061	10.0.0.11	10.0.0.12	DNS	84	Standard query 0x0000 A http://http_server.com
11	3.291255472	PcsCompu_e4:7d:89		ARP	44	Who has 10.0.2.2? Tell 10.0.2.15
12	3.291835754	RealtekU_12:35:02		ARP	62	10.0.2.2 is at 52:54:00:12:35:02
13	3.307259146	10.0.0.12	10.0.0.11	DNS	122	Standard query response 0x0000 A http://http_server.com A 0.0.0.0
14	5.212336704	10.0.2.15	172.20.10.1	DNS	102	Standard query 0xb5a0 A connectivity-check.ubuntu.com OPT
15	5.212465570	10.0.2.15	172.20.10.1	DNS	102	Standard query 0xca76 AAAA connectivity-check.ubuntu.com OPT
16	5.241427994	10.0.2.15	0.0.0.0	FTP-DATA	57	0 FTP Data: 1 bytes
17	5.381538052	127.0.0.1	127.0.0.1	TCP	76	1 35636 → 80 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=294328113 TSecr=
18	5.381562949	127.0.0.1	127.0.0.1	TCP	76	1 80 → 35636 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM TSval=2943
19	5.382135636	127.0.0.1	127.0.0.1	TCP	68	1 35636 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=294328113 TSecr=294328113
20	5.382216051	127.0.0.1	127.0.0.1	HTTP	112	1 GET /example.txt HTTP/1.1
21	5.382224343	127.0.0.1	127.0.0.1	TCP	68	1 80 → 35636 [ACK] Seq=1 Ack=45 Win=65536 Len=0 TSval=294328114 TSecr=294328114
22	5.382661833	127.0.0.1	127.0.0.1	TCP	147	1 80 → 35636 [PSH, ACK] Seq=1 Ack=45 Win=65536 Len=79 TSval=294328115 TSecr=2943281
23	5.382674874	127.0.0.1	127.0.0.1	TCP	68	1 35636 → 80 [ACK] Seq=45 Ack=80 Win=65536 Len=0 TSval=294328115 TSecr=294328115
24	5.382703272	127.0.0.1	127.0.0.1	HTTP	68	1 HTTP/1.1 302 Found
25	5.388928899	10.0.0.11	10.0.0.12	DNS	107	Standard query 0x0000 A http://redirected_http_server.com/example.txt
26	5.431530761	127.0.0.1	127.0.0.1	TCP	68	1 35636 → 80 [ACK] Seq=45 Ack=81 Win=65536 Len=0 TSval=294328163 TSecr=294328115
27	5.551797226	10.0.0.12	10.0.0.11	DNS	168	Standard query response 0x0000 A http://redirected_http_server.com/example.txt A
28	5.574310133	127.0.0.1	127.0.0.1	TCP	68	1 35636 → 80 [FIN, ACK] Seq=45 Ack=81 Win=65536 Len=0 TSval=294328306 TSecr=2943281
29	5.574326561	127.0.0.1	127.0.0.1	TCP	68	1 80 → 35636 [ACK] Seq=81 Ack=46 Win=65536 Len=0 TSval=294328306 TSecr=294328306
30	5.574500465	10.0.2.15	10.0.2.15	TCP	76	2 57256 → 80 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=1227761463 TSecr
31	5.574512769	10.0.2.15	10.0.2.15	TCP	76	2 80 → 57256 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM TSval=12277
32	5.574522626	10.0.2.15	10.0.2.15	TCP	68	2 57256 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1227761463 TSecr=1227761463
33	5.574633903	10.0.2.15	10.0.2.15	HTTP	114	2 GET /example.txt HTTP/1.1
34	5.574641981	10.0.2.15	10.0.2.15	TCP	68	2 80 → 57256 [ACK] Seq=1 Ack=47 Win=65536 Len=0 TSval=1227761464 TSecr=1227761464
35	5.574685289	10.0.2.15	10.0.2.15	TCP	160	2 80 → 57256 [PSH, ACK] Seq=1 Ack=47 Win=65536 Len=92 TSval=1227761464 TSecr=122776
36	5.574696281	10.0.2.15	10.0.2.15	HTTP	68	2 HTTP/1.1 200 OK
37	5.575621913	10.0.2.15	10.0.2.15	TCP	68	2 57256 → 80 [ACK] Seq=47 Ack=93 Win=65536 Len=0 TSval=1227761465 TSecr=1227761464

if user choose RUDP:

On rows 3-7 we can notice that there is connection and communication between the client and the tcp server.

On row 10 there is the DNS query and on row 13 the server response.

On rows 17-42 there is the client-http_server connection (beside 29, 30 where there is DNS query and response).

3	1.851599396	0.0.0.0	255.255.255.255	DHCP	288	DHCP Discover - Transaction ID 0x1
4	1.852044847	10.0.2.2	10.0.2.16	DHCP	592	DHCP Offer - Transaction ID 0x1
5	1.971941488	10.0.0.10	255.255.255.255	DHCP	312	DHCP Offer - Transaction ID 0x1
6	3.002656894	0.0.0.0	10.0.0.10	DHCP	288	DHCP Request - Transaction ID 0x1
7	3.125279520	10.0.0.10	255.255.255.255	DHCP	312	DHCP ACK - Transaction ID 0x1
8	3.195307972	PcsCompu_e4:7d:89		ARP	44	Who has 10.0.2.2? Tell 10.0.2.15
9	3.195616371	RealtekU_12:35:02		ARP	62	10.0.2.2 is at 52:54:00:12:35:02
10	3.226522196	10.0.0.11	10.0.0.12	DNS	84	Standard query 0x0000 A http://http_server.com
11	3.384181838	PcsCompu_e4:7d:89		ARP	44	Who has 10.0.2.2? Tell 10.0.2.15
12	3.384873350	RealtekU_12:35:02		ARP	62	10.0.2.2 is at 52:54:00:12:35:02
13	3.406939083	10.0.0.12	10.0.0.11	DNS	122	Standard query response 0x0000 A http://http_server.com A 0.0.0.0
14	3.998848191	10.0.2.15	172.20.10.1	DNS	102	Standard query 0xe8b0 AAAA connectivity-check.ubuntu.com OPT
15	3.998969457	10.0.2.15	172.20.10.1	DNS	102	Standard query 0x042c A connectivity-check.ubuntu.com OPT
16	4.538890023	10.0.2.15	0.0.0.0	FTP-DATA	57	0 FTP Data: 1 bytes
17	4.669921348	127.0.0.1	127.0.0.1	UDP	47	37675 → 80 Len=3
18	4.670514940	127.0.0.1	127.0.0.1	UDP	52	80 → 37675 Len=8
19	4.671134990	127.0.0.1	127.0.0.1	UDP	47	37675 → 80 Len=3
20	4.671296987	127.0.0.1	127.0.0.1	UDP	86	37675 → 80 Len=42
21	4.671396781	127.0.0.1	127.0.0.1	UDP	47	80 → 37675 Len=3
22	4.671607205	127.0.0.1	127.0.0.1	UDP	47	37675 → 80 Len=3
23	4.671809582	127.0.0.1	127.0.0.1	UDP	123	80 → 37675 Len=79
24	4.672162416	127.0.0.1	127.0.0.1	UDP	47	37675 → 80 Len=3
25	4.672384898	127.0.0.1	127.0.0.1	UDP	47	80 → 37675 Len=3
26	4.672630345	127.0.0.1	127.0.0.1	UDP	47	37675 → 80 Len=3
27	4.672894879	127.0.0.1	127.0.0.1	UDP	52	80 → 37675 Len=8
28	4.673182490	127.0.0.1	127.0.0.1	UDP	47	37675 → 80 Len=3
29	4.705533319	10.0.0.11	10.0.0.12	DNS	107	Standard query 0x0000 A http://redirected_http_server.com/example.txt
30	4.872280189	10.0.0.12	10.0.0.11	DNS	168	Standard query response 0x0000 A http://redirected_http_server.com/exam
31	4.894904989	10.0.2.15	10.0.2.15	UDP	47	59046 → 80 Len=3
32	4.895143397	10.0.2.15	10.0.2.15	UDP	52	80 → 59046 Len=8
33	4.895197336	10.0.2.15	10.0.2.15	UDP	47	59046 → 80 Len=3
34	4.895229500	10.0.2.15	10.0.2.15	UDP	88	59046 → 80 Len=44
35	4.895243898	10.0.2.15	10.0.2.15	UDP	47	80 → 59046 Len=3
36	4.895275459	10.0.2.15	10.0.2.15	UDP	47	59046 → 80 Len=3
37	4.895290314	10.0.2.15	10.0.2.15	UDP	134	80 → 59046 Len=90
38	4.895490541	10.0.2.15	10.0.2.15	UDP	47	59046 → 80 Len=3
39	4.895520783	10.0.2.15	10.0.2.15	UDP	47	80 → 59046 Len=3
40	4.895642372	10.0.2.15	10.0.2.15	UDP	47	59046 → 80 Len=3
41	4.895664511	10.0.2.15	10.0.2.15	UDP	52	80 → 59046 Len=8
42	4.895689407	10.0.2.15	10.0.2.15	UDP	47	59046 → 80 Len=3

Bibliography:

- https://sudonull.com/post/30978-DHCP-Mysql-server-in-Python
- https://null-byte.wonderhowto.com/how-to/create-packets-from-scratch-with-scapy-for-scanning-dosing-0159231/
- python code bases from lectures