ISE306 Computer Networks

Homework

March 28, 2022

Overview

Implement a simplified DNS system using TCP sockets and Python that contains a primary DNS server and a secondary DNS server. Both DNS servers get the list of host-name and IP address associations from a file whose name is supplied as a command line argument when they start running. For example;

```
$python primary_server.py primary_port secondary_ip secondary_port addresses.txt
```

starts **primary DNS server** on port **primary_port** which gets IP addresses of host names from the file **addresses.txt**. The addresses.txt contains host names and IP addresses separated by space characters. An example addresses.txt file is as follows:

```
python.org 192.168.2.1
ninova.itu.edu.tr 192.168.10.2
youtube.com 192.168.18.1
bbf.itu.edu.tr 10.0.0.2
```

A **client** starts running as follows:

```
$python client.py ip_address port_number host_name
```

where **ip_address** and **port_number** are IP address and port number of the **primary DNS server**, respectively. The **host_name** argument is the name of the host whose IP address will be found by first contacting to primary DNS server. If the searched host name is available in the host name list of the primary DNS server, it returns the IP address. Otherwise, the primary DNS server returns the IP address of the **secondary DNS server** and the client contacts secondary DNS server for IP address of the host.
Please note that, you have to use exactly the same message format specified below.

Primary DNS Server

1. The primary DNS server starts running with the command:

```
$python primary_server.py primary_port secondary_ip secondary_port addresses_file_name
```

where primary_port argument is the port number on which primary DNS server waits, secondar_ip argument is the IP address of the secondary DNS server in dotted-decimal format, secondary_port argument is the port number of the secondary DNS server, and addresses_file_name argument is name of the file which contains host-name and IP address associations, as exemplified above with the addresses.txt file.

2. The primary DNS server waits for a request from the client. The client supplies its request with the following message format: GET HOST-NAME where HOST-NAME is the name of the host.

3. The primary DNS server looks for the HOST-NAME in the addresses file.

a) If the primary DNS server finds the HOST-NAME in the file, it returns back the IP address of the host with the following message format: FOUND HOST-NAME IP-ADDRESS

b) If the primary DNS server CANNOT find the HOST-NAME in the file, it returns back the address of the secondary DNS server with the following message format: REDIR HOST-NAME SECONDARY_IP SECONDARY_PORT
where HOST-NAME is the requested (but could not be found) host name, SECONDARY_IP_ADDRESS is the IP address of the secondary DNS server in dotted-decimal format, SECONDARY_PORT is the port number of the secondary DNS server.

4. The primary DNS server continues with Step (2) to get another client request.

Secondary DNS Server

1. The secondary DNS server starts running with the command:

```
$python secondary_server.py port_number addresses_file_name
```

where port_number argument is the port number on which secondary DNS server waits and addresses_file_name argument is name of the file which contains host-name and IP address associations, as exemplified above with the addresses.txt file.

2. The secondary DNS server waits for a request from the client. The client supplies its request with the following message format: GET HOST-NAME
where HOST-NAME is the name of the host.

3. The secondary DNS server looks for the HOST-NAME in the addresses file.

a) If the secondary DNS server finds the HOST-NAME in the file, it returns back the IP address of the host with the following message format: FOUND HOST-NAME IP-ADDRESS where HOST-NAME is the requested host name and IP-ADDRESS is the IP address of the HOST-NAME in dotted-decimal notation.

b) If the primary DNS server CANNOT find the HOST-NAME in the file, it returns back an error message in the following message format: ERROR HOST-NAME
where HOST-NAME is the requested (but could not be found) host name.

4. The secondary DNS server continues with Step (2) to get another client request.

Client

1. The client starts running with the command:

```
$python client.py ip_address port_number host_name
```

where ip_address and port_number are IP address and port number of the primary DNS server, respectively. The host_name argument is the name of the host whose IP address will be requested.

2. The client contacts with primary DNS server and asks for the IP address of the host with the following message format: GET HOST-NAME
where HOST-NAME is the name of the host.

3. If the primary DNS server responds with a message: FOUND HOST-NAME IP-ADDRESS
then the IP address is found. The client prints out the HOST-NAME and IP-ADDRESS (e.g., "The IP address of the host HOST-NAME is IP-ADDRESS") and **terminates the program**.

4. If the primary DNS server responds with a message: REDIR HOST-NAME SECONDARY_IP SECONDARY_PORT
then the IP address is NOT found on the primary DNS server.

5. The client contacts with secondary DNS server using SECONDARY_IP and SECONDARY_PORT information supplied by the primary DNS server and asks for the IP address of the host with the following message format: GET HOST-NAME
where HOST-NAME is the name of the host.

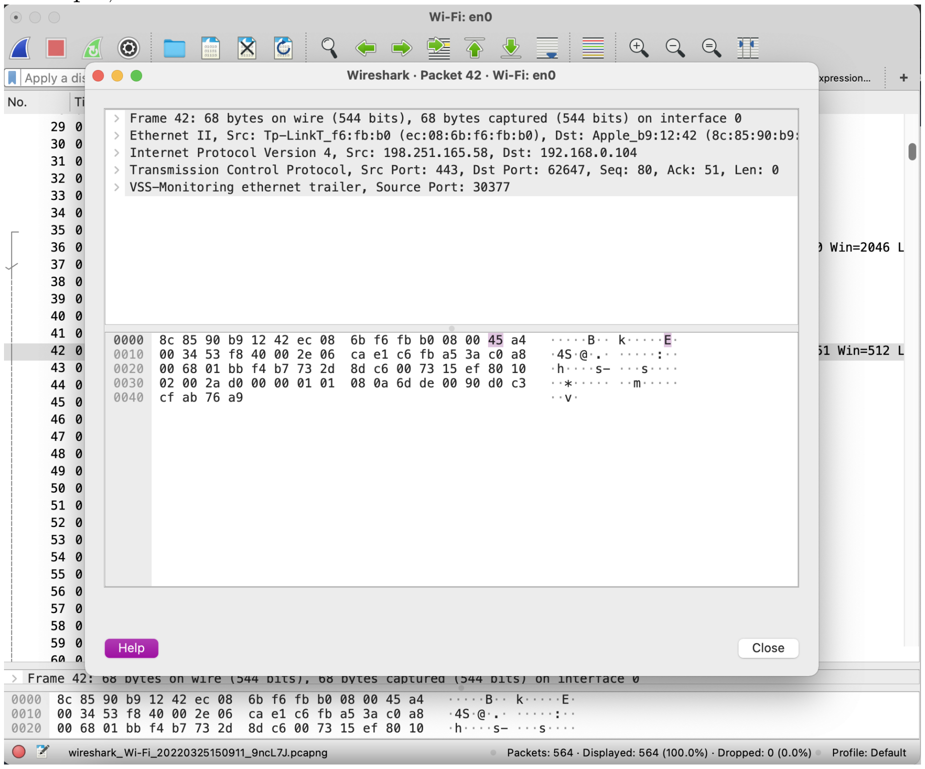
6. If the secondary DNS server responds with a message: FOUND HOST-NAME IP-ADDRESS
then the IP address is found. The client prints out the HOST-NAME and IP-ADDRESS (e.g., "The IP address of the host HOST-NAME is IP-ADDRESS") and **terminates the program**.

7. If the secondary DNS server responds with a message: ERROR HOST-NAME

then the IP address could not be found. The client prints out an error message including the HOST-NAME (e.g., “The IP address of the host HOST-NAME could not be found, sorry!”) and **terminates the program**.

Wireshark Report

Prepare a report about your program. Explain your program flow and add the Wireshark captures explained below:
While running your program, start run Wireshark Network Analyzer. Try to capture a packet of your program and your local port and take a screen shot. For example;



```
> Frame 42: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface 0
> Ethernet II, Src: Tp-LinkT_f6:fb:b0 (ec:08:6b:f6:fb:b0), Dst: Apple_b9:12:42 (8c:85:90:b9:
> Internet Protocol Version 4, Src: 198.251.165.58, Dst: 192.168.0.104
> Transmission Control Protocol, Src Port: 443, Dst Port: 62647, Seq: 80, Ack: 51, Len: 0
> VSS-Monitoring ethernet trailer, Source Port: 30377

0000  8c 85 90 b9 12 42 ec 08  6b f6 fb b0 08 00 45 a4   ·····B·· k·····E·
0010  00 34 53 f8 40 00 2e 06  ca e1 c6 fb a5 3a c0 a8   ·4S·@·.· ·····:··
0020  00 68 01 bb f4 b7 73 2d  8d c6 00 73 15 ef 80 10   ·h····s- ···s····
0030  02 00 2a d0 00 00 01 01  08 0a 6d de 00 90 d0 c3   ··*····· ··m·····
0040  cf ab 76 a9                                        ··v·
```

```
wireshark_Wi-Fi_20220325150911_9ncL7J.pcapng    Packets: 564 · Displayed: 564 (100.0%) · Dropped: 0 (0.0%)    Profile: Default
```

Secondly, from ”Statistics/TCP Stream Graphs” get **throughput** and **round trip time** graphs and take screenshots. Include the screenshots and graphs in your report. Evaluate and give details about throughput and round trip time concepts according to your graphs.

Submission Rules

-Submit a .zip file includes your source codes and report pdf.
-Any type of plagiarism will not be tolerated.
-If you have any questions, please e-mail Sultan Çoğay (cogay@itu.edu.tr)