Lab 8

Computer Communication Networks

TCP & UDP

The lab's schedule

Published date:	02/06/22
Quiz date:	07,09/06/22
Deadline for the final report:	18/06/21

BEN-GURION UNIVERSITY OF THE NEGEV - COMMUNICATION SYSTEMS ENGINEERING

General Instructions

- **The final report** will be based on the answers of this document.
- **Remember** to pay attention to the "Final report submission" in the Introduction Lab.
- Most of the laboratory experiments based on the book: "*Mastering Networks: An Internet Lab Manual*"
- For each exercise, it is recommended to **read it all to understand the main idea before you do it**.
- IP address is composed of four octets ("octet1.octet2.octet3.octet4").
  In our Labs all IP addresses will be according to the Network Figure, except *octet2*.
  ***Octet2* will be according to the pair's number ("10.*X*.0.1", *X = pair's number*).**
- **Write the number *X*** clearly **on the title page** of your final report.

Reading Material

As always, focus on the subjects found in the ***Preliminary Questions*** section, **not** all subjects found in the links!

For the quiz you can read only from:

- Read about UDP: https://en.wikipedia.org/wiki/User_Datagram_Protocol
- Read about TCP: https://en.wikipedia.org/wiki/Transmission_Control_Protocol

For the rest of the lab (no need to read for the quiz):

- A nice example of the Sliding window mechanism (Selective Repeat):
  http://www.ccs-labs.org/teaching/rn/animations/gbn_sr/ \
- Read about the command/program ttcp: http://linux.die.net/man/1/ttcp
- Read about TCP stream graphs (you will use those in part 2 of this lab)
  https://www.wireshark.org/docs/wsug_html_chunked/ChStatTCPStreamGraphs.html

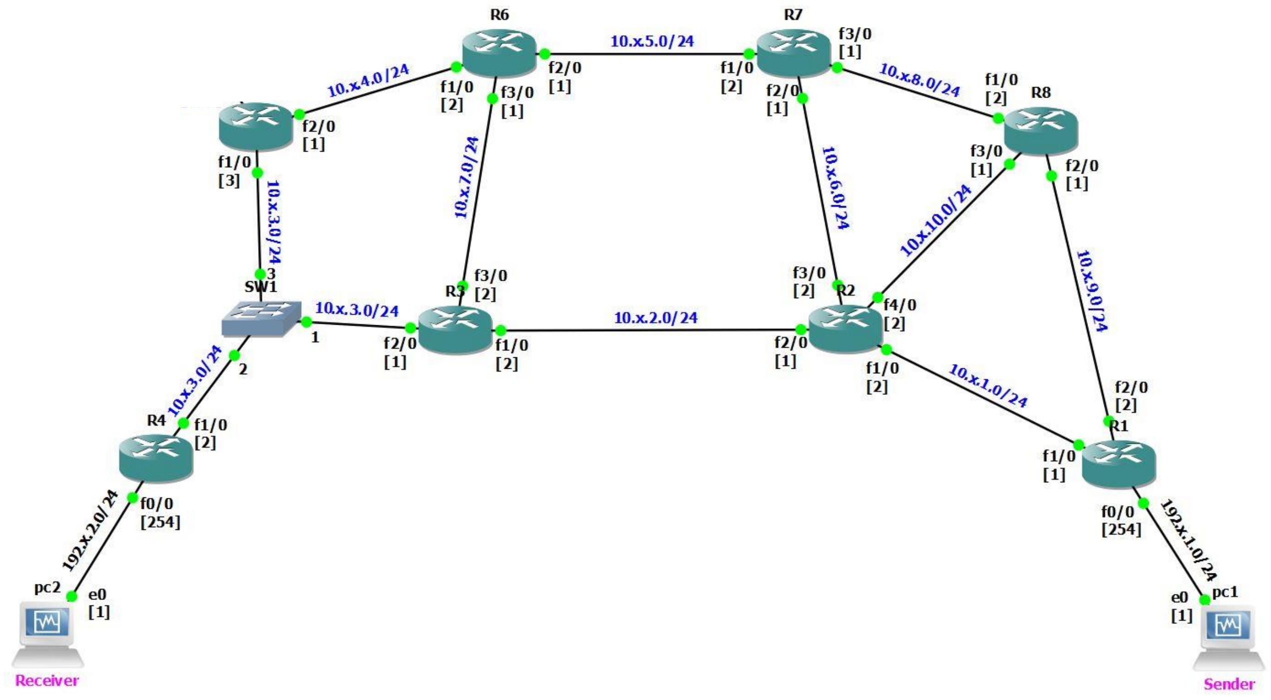
Preliminary Questions

- Describe the fields of a TCP segment and a UDP datagram.
- Describe the error detection method of TCP and UDP protocols.
- Describe the process of 3-way handshake.
  - In your answer consider various situations such that the connection is not successful.
- Briefly explain what are the main differences between TCP and UDP?
- Briefly explain about the fairness considerations that each protocol keeps.
  - For TCP, two examples of these are: Congestion control and Flow control.

The practical section

TCP and UDP (Part A) : Basic connection management

1. Network Configuration



Pre

1.1. The typology is based on one of the following:

Topology 8, *Topology 9* or *Topology 10*.

1.2. Use the topology assigned to your pair.

1.3. But first, save as a new project before making any changes.

Do

1.4. Add and configure the PCs according to the figure.

1.5. Configure all the necessary on the routers for a proper communication between the PCs.

1.6. Test your network configuration using "ping" utility

1.7. Perform "`write`" on each router, save the topology and ZIP it.

2. Transmitting data with UDP

This exercise consists of setting up a UDP data transfer between two hosts (PC1 and PC2) and observing the UDP traffic.

Before starting this exercise, please read these notes:

- ttcp, in the UDP mode, sends 1 short UDP packet before sending the data, and 5 short UDP packets after the data was sent. These short packets have a payload of 4 byte. They are not a part of the UDP protocol, but a part of the ttcp application, i.e., they are not necessary for the UDP data transfer. So, just ignore these short packets in the experiment.
- Since the data transfer will occur at a relatively high rate, sometimes Wireshark fails to capture all the packets. If you see that there are not enough packets in the Wireshark capture (after you stop the capture), you have to perform the experiment again until you are lucky to capture all the expected packets.

Pre

2.1. The exercise is based on the topology you had just saved in the first exercise.

2.2. For this part you will use ttcp as once as a receiver of UDP messages and once as a sender.

Do

2.3. Start Wireshark on R3 interface f1/0 and set filter to display only packets that include IP address *192.x.2.1*.

2.4. On PC2, start a *ttcp* receiver that receives UDP traffic with the following command:

```
ttcp -r -l1024 -n10 -p4444 -u
```

2.5. On PC1, start a *ttcp* sender that transmits UDP traffic by typing:

```
ttcp -t -l1024 -n10 -p4444 -u 192.x.2.1
```

2.6. Stop the Wireshark capture and save all the captured traffic for the exercise below.

Attach to your report

2.7. How many packets are exchanged in the data transfer? How many packets are transmitted for each UDP datagram? What is the size of the UDP payload of these packets?

2.8. Inspect the fields in the UDP headers. Which fields in the headers do not change in different packets?

2.9. Observe the port numbers in the UDP header. How did the sender select the source port number?

2.10. Compare the total number of bytes transmitted, including Ethernet, IP, and UDP headers, to the amount of application data transmitted. How much overhead did the protocol use?

3. Transmitting data with TCP

Here, you repeat the previous exercise, but use TCP for data transfer.

Pre

3.1. Use the saved topology from the previous exercise.

Do

3.2. Start Wireshark on R3 interface f1/0 and set filter to display only packets that include IP address 192.x.2.1.

3.3. On PC2, start a *ttcp* receiver that receives packets sent by PC1:

```
ttcp -r -l1024 -n10 -p4444
```

3.4. On PC1, start a *ttcp* sender that transmits packets from PC1 to PC2:

```
ttcp -t -l1024 -n10 -p4444 -D 192.x.2.1
```

3.5. Stop the Wireshark capture and save all the captured traffic for the exercise below.

Attach to your report

3.6. How many packets are exchanged in the data transfer? What are the sizes of the TCP segments?

3.7. What are the initial sequence numbers of the sender and the receiver? What is the range of the sequence numbers?
*Note that Wireshark shows us a relative sequence numbers. Look for the real number in the packet's bits.

3.8. Identify the first packet that contains application data. What is the relative sequence number used in the first byte of application data sent from the TCP sender to the TCP receiver? Why it is not zero, even though this is the first data packet?

3.9. Identify the packets of three-way handshake. Which flags are set in the TCP headers? Explain how these flags are interpreted by the receiving TCP server or TCP client(the sender and receiver)?

3.10. How many packets do not carry a payload, that is, how many packets are control packets(messages with no application data such as a SYN messages)?

3.11. Inspect the TCP headers. Which packets contain flags in the TCP header? Which types of flags do you observe?

3.12. Compare the total number of bytes transmitted, in both directions, including Ethernet, IP, and TCP headers, to the amount of application data transmitted. How much overhead did the protocol use?

3.13. What are the finale ack numbers in both directions? Explain each number, how did we get those numbers?

3.14. Determine the values of window sizes for the sender and the receiver.

3.15. What is the MSS value that is negotiated between the sender and receiver?

4. Comparing basic UDP and TCP data transfers

Here, we will compare the utilization of each of the protocols.

Pre

4.1. Use the saved Wireshark pcaps from the last two exercises to answer the fallowing questions.

Attach to your report

4.2. Compare the amount of data transmitted in the TCP and the UDP data transfers. Which protocol has less overhead? In your answer relate to the number of packets and the number of bytes in both TCP and UDP sessions.

4.3. Take the biggest UDP datagram and the biggest TCP segment that you observed. Compare the amount of application data that is transmitted in the UDP datagram and the TCP segment. Explain why we get exactly these sizes.

4.4. What are the underlying mechanisms that decide how many bytes are sent in each TCP packet and each in UDP packet.

TCP and UDP (Part B): Real world connection management

This part is done under the physical Ubuntu PCs, and not on GNS3

Login with your personal BGU user.

5. TCP Fairness and advanced Wireshark statistics

In this exercise we'll use the *ttcp* program in order to generate TCP and UDP data. The program is located at the course website. This program is similar in nature to the programs you wrote in lab 5.

Note: The ***Windows*** *version is called* `pcattcp.exe`*, you may use it if you are unable to run on Ubuntu for any reason*

Pre

5.1. The exercise is done using 3 physical PCs directly connected to a physical switch. (not on GNS3!)

5.2. In order to succeed in the experiment, we recommend that you perform the exercise in the course lab using Ubuntu.

5.3. The first PC (will be called *PC_RCV*) will be used to receive two TCP streams, and the two other PCs (will be called *PC_TX1* and *PC_TX2*) are used to send them.

5.4. Copy `ttcp` to your desktop of all the PCs.

Do

5.5. Check the IP addresses of all the PCs you are using. Make a screen capture of the IP address using ifconfig.

5.6. On *PC_RCV* start Wireshark with capture filter exclusively for *PC_TX1* and *PC_TX2*.

5.7. On *PC_RVC* run two separate terminals and change directory (`cd`) to the Desktop folder (where you put the file).

5.8. Execute the "`chmod 777 ttcp`" if necessary.

5.9. On *PC_RVC* run ***ttcp*** in listening mode on each terminal, as follows:

```
./ttcp -r -l1024 -n200000 -p4444
./ttcp -r -l1024 -n200000 -p4445
```

Note: in "-l1024" that's an *L not a 1!*

5.10. On *PC_TX1* run:

```
./ttcp -t -l1024 -n200000 -p4444 IP_of_PC_RCV
```

5.11. On *PC_TX2*, **wait 3 seconds** and run:

```
./ttcp -t -l1024 -n200000 -p4445 IP_of_PC_RCV
```

5.12. Stop capture traffic and generate the following graphs:

5.12.1. An *IO Graph*, where the download stream from *PC_TX1* is green and the download stream from *PC_TX2* is blue.

5.12.2. Two Stevens graphs (one for each download stream).

5.12.3. Create graphs as described in the appendix.

Attach to your report

5.13. Attach print screens of the two Stevens graphs and the IO graph along with the print screens of the IPs.

5.14. Look at each of the commands you used on the PCs (from parts 9 to 11). Explain the meaning of each part of these commands (i.e. ,what do `-t`, `-l1024`, `-r` etc. mean?).

5.15. What TCP source ports were used by *PC_TX1* and *PC_TX2*? How these numbers were chosen?

5.16. What are the sequence numbers of the TCP SYN segments that are used to initiate the TCP connections between sending PCs and the receiving PC? How did you find these numbers? Are they relative or absolute?
Note*: sequence numbers showed by Wireshark in the main window are relative numbers. To find out the absolute sequence numbers, you have to investigate the TCP header.(Note: Newer versions show both)*

5.17. What is the MSS value? How did you find it? What are the sizes of "*Receiver Window*" reported by the sender and the receiver?

5.18. How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing other amount of data?

5.19. What is the average throughput (bytes transferred per unit time) for each of the TCP connections? Explain how you calculated this value.

Hint: you can use this filter: "`tcp.flags.syn==1 or tcp.flags.fin==1`". If you can't see "fin" packets for each connection, think of some other filter that will help you to see where the connection starts and ends

5.20. Notice that there are two different slopes in the Stevens graph. Calculate (on your graph) the value of each slope $\left(\frac{\Delta y}{\Delta x}\right)$.

Hints:Near the graph you will see the "Graph Control Window". Go to it's "cross" tab and turn the cross on. Now you can easily see the (x,y) points on the graph(note that this feature might be *on* by default).
Don't "click" on the graph area. The "click" will activate the "zoom" feature which we don't need here.

5.21. Why are there two different slopes? What is the meaning of their values?

5.22. Using the above results, what can you say regarding the fairness of the TCP protocol? I.e., how does TCP share the bandwidth between several connections?

5.23. What was the "bottleneck" for the network traffic in this experiment, and why? (*PC_RVC*'s interface, *PC_TX1*'s interface, *PC_TX2*'s interface or the switch)

6. TCP vs. UDP "competition"

We now want to see how TCP competes with UDP. In order to do it, we'll use our best friend - ***ttcp***. The first PC will be a receiver *PC_RCV*, we will generate a TCP transmission from *PC_TX1* and a UDP transmission from *PC_TX2*.

Pre

6.1. Again, the exercise is done using 3 physical PCs directly connected to a physical switch. (not on GNS3!)

6.2. In order to succeed in the experiment, we recommend that you perform the exercise in the lab using Ubuntu.

6.3. The first PC (will be called *PC_RCV*) will be used to receive two streams (one of TCP and one of UDP), and the two other PCs (will be called *PC_TX1* and *PC_TX2*) are used to send them.

6.4. Copy `ttcp` to the desktop of all the PCs.

Do

6.5. On *PC_RVC* run the *ttcp* in listening:

```
./ttcp -r -l1024 -n200000 -p4444
./ttcp -r -l1024 -n200000 -u -p4445
```

6.6. On *PC_TX1* start TCP transmission:

```
./ttcp -t -l1024 -n200000 -p4444 IP_of_PC_RCV
```

6.7. Wait for 3 seconds.

6.8. On *PC_TX2* start UDP transmission:

```
./ttcp -t -l1024 -n200000 -u -p4445 IP_of_PC_RCV
```

6.9. Stop capture traffic and generate the following graphs:

6.9.1. An *IO Graph*

- Download stream from *PC_TX1* (TCP) is green
- Download stream from *PC_TX2* (UDP) is red.
- Total Download stream is black.

Note: *X axis tick interval is 0.1 seconds and Y axis units are Bytes/Tick.*

6.9.2. One Stevens graphs for the TCP download stream.

Attach to your report

6.10. Attach print screens of the two Stevens graphs and the IO graph.

6.11. In step 6.5 we used different ports for TCP and UDP. Was this really necessary? Why?

6.12. Explain the graphs from step (6.9). What can you tell about the TCP/UDP "competition"? Explain, what general conclusion can you draw from each.

6.13. Can you see retransmissions in the TCP traffic? If so, highlight at least one retransmission on a graph and attach a screenshot to your report.

6.14. What was the "bottleneck" for the network traffic in this experiment, and why? (*PC_RVC*'s interface, *PC_TX1*'s interface, *PC_TX2*'s interface or the switch)

6.15. What would be changed in the experiment result if we would send the TCP and the UDP traffic from a single PC and not from two separate PCs?
Will the "bottleneck" change?

Good Luck!

7. Appendix: Wireshark Statistics.

How to create a Time-Sequence Graph (Stevens)

Open the Wireshark pcap and go to:
`Statistics->Conversations->Label: "TCP".`
A window will open with all the TCP flows that the Wireshark captured. A single flow is the collection of all the packet traffic between to applications in the net, back and forth. Each flow defined by destination IP+port and source IP+port (4 fields for each flow).

7.1. Find the IP address of the server against whom the test was done and add it to the lab report.
Hint: most of the traffic the Wireshark recorded belongs to the test.

7.2. Leave the "Conversations" window open for the questions below.

7.3. Go back to the main window in Wireshark and apply a filter that will leave only the packets that addressed to/from the IP address you found.
In the "Conversations" window mark "Limit to display filter". Make sure that you are left only with the relevant flows.

7.1. In our case, Find and mark a packet, that belongs to the largest flow in the pcap (largest number of **Bytes**), and is sent form the "*far host*" server to your *local host*.
Tip: right click on the flow line->Find Packet->Find Packet-> A←B.

7.2. **Pay attention, it is very important that the relevant packet will be marked** (click on it). After you chose the relevant packet, in the main window go to:
`Statistics->TCP Stream Graph->Time-Sequence Graph (Stevens).`

7.3. In the control window, go to "Origin"(Or on newer versions, just set the view to show the axis origin .i.e. {0,0}) and choose "beginning of capture". Then go to "Cross" and choose "on". In order to move inside the Graph you need to right-click+hold+drag, in order to zoom in/out use "i","o" letters on the keyboard.

7.4. Leave all the windows open and answer the questions for the report.

How to create an IO graph

7.5. To better see how these TCP sessions with the server behave, create an IO graph (`Statistics->IO graph`).

7.6. Plot the graph so that X axis is **time** and Y axis is **bytes/sec** for each of the conversations. Present a red graph for the direction from server to local host and a blue graph for direction from the local host to the server.
Make sure your plot is easily readable, play with the tick interval and “pixels per tick” properties to change its scale.