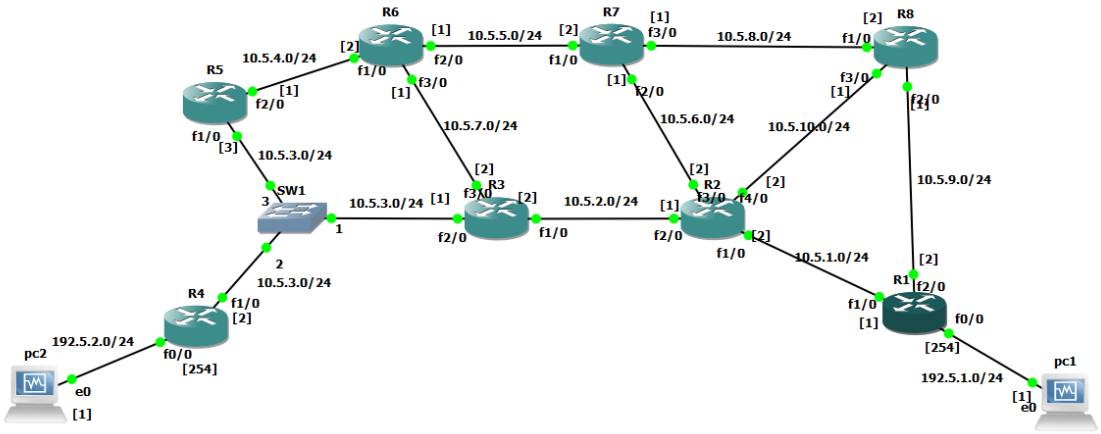
Lab 8

Shir Moshe - 318492667

Nadav Biran - 316468834

1. Network Configuration – Topology 11 (base on topology 9)

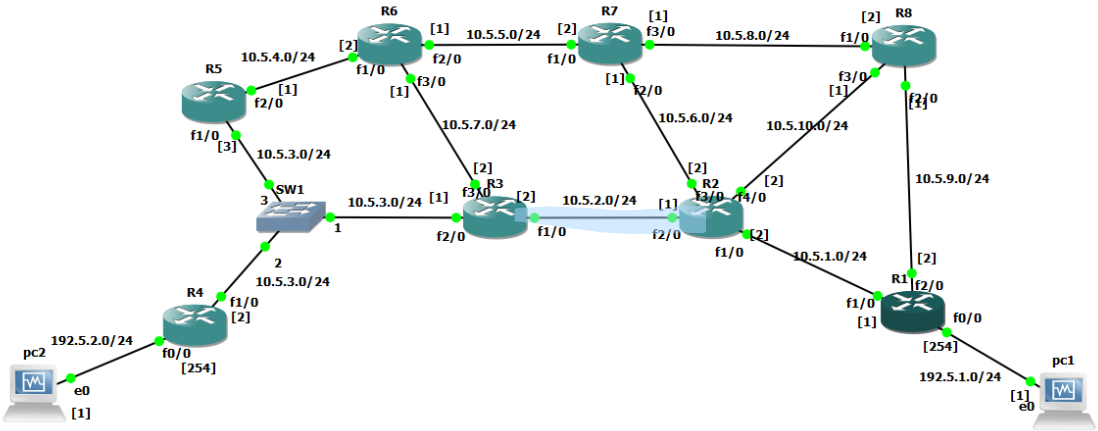


1.1.

Pc1	*route add default gw 192.5.1.254*
Pc2	*route add default gw 192.5.2.254*
	configure terminal *interface FastEthernet0/0* *shutdown* *no ip route 192.168.2.0 255.255.255.0* *exit* *configure terminal* *interface FastEthernet0/0* *ip address 192.5.1.254 255.255.255.0* *no shutdown* *end*
	configure terminal router ospf 1 network 192.5.1.0 0.0.0.255 area 1 passive-interface FastEthernet 0/0 interface FastEthernet 0/0 ip address 192.5.1.254 255.255.255.0 no shutdown end
	configure terminal router ospf 1 network 192.5.2.0 0.0.0.255 area 1 passive-interface FastEthernet 0/0 interface FastEthernet 0/0 ip address 192.5.2.254 255.255.255.0 no shutdown end

2. Transmitting data with UDP

2.1.



2.2.

2.3. Start Wireshark on R3 – R2, set filter to display packets that include :

Ip == 192.5.2.1

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

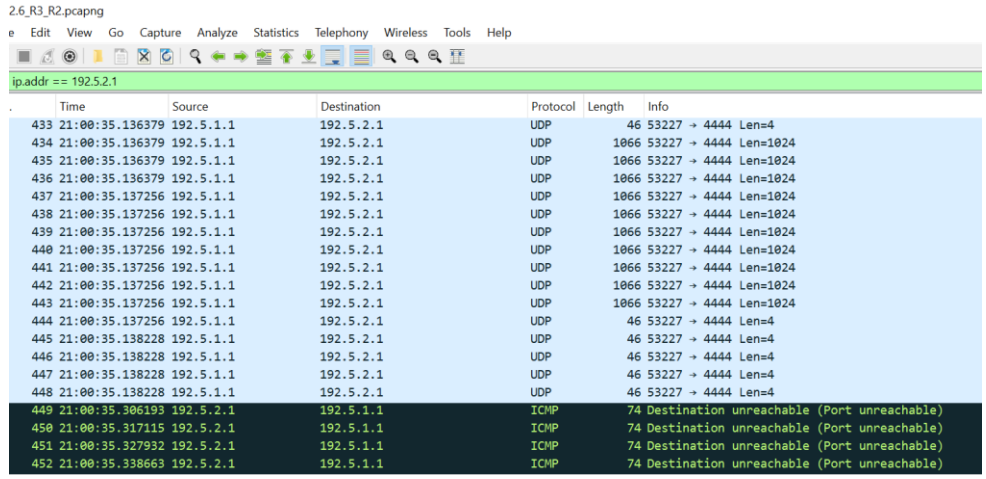
ttcp –r –l1024 –n10 –p4444 –u

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

ttcp –t –l1024 –n10 –p4444 –u 192.5.2.1

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

2.7.

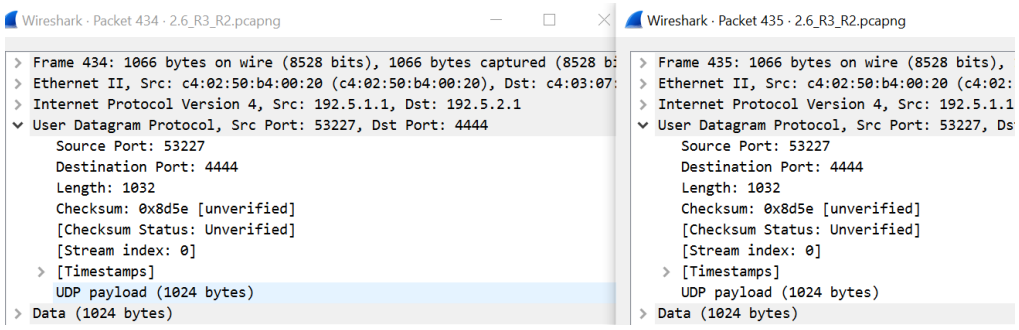


	Time	Source	Destination	Protocol	Length	Info
433	21:00:35.136379	192.5.1.1	192.5.2.1	UDP	46	53227 → 4444 Len=4
434	21:00:35.136379	192.5.1.1	192.5.2.1	UDP	1066	53227 → 4444 Len=1024
435	21:00:35.136379	192.5.1.1	192.5.2.1	UDP	1066	53227 → 4444 Len=1024
436	21:00:35.136379	192.5.1.1	192.5.2.1	UDP	1066	53227 → 4444 Len=1024
437	21:00:35.137256	192.5.1.1	192.5.2.1	UDP	1066	53227 → 4444 Len=1024
438	21:00:35.137256	192.5.1.1	192.5.2.1	UDP	1066	53227 → 4444 Len=1024
439	21:00:35.137256	192.5.1.1	192.5.2.1	UDP	1066	53227 → 4444 Len=1024
440	21:00:35.137256	192.5.1.1	192.5.2.1	UDP	1066	53227 → 4444 Len=1024
441	21:00:35.137256	192.5.1.1	192.5.2.1	UDP	1066	53227 → 4444 Len=1024
442	21:00:35.137256	192.5.1.1	192.5.2.1	UDP	1066	53227 → 4444 Len=1024
443	21:00:35.137256	192.5.1.1	192.5.2.1	UDP	1066	53227 → 4444 Len=1024
444	21:00:35.137256	192.5.1.1	192.5.2.1	UDP	46	53227 → 4444 Len=4
445	21:00:35.138228	192.5.1.1	192.5.2.1	UDP	46	53227 → 4444 Len=4
446	21:00:35.138228	192.5.1.1	192.5.2.1	UDP	46	53227 → 4444 Len=4
447	21:00:35.138228	192.5.1.1	192.5.2.1	UDP	46	53227 → 4444 Len=4
448	21:00:35.138228	192.5.1.1	192.5.2.1	UDP	46	53227 → 4444 Len=4
449	21:00:35.306193	192.5.2.1	192.5.1.1	ICMP	74	Destination unreachable (Port unreachable)
450	21:00:35.317115	192.5.2.1	192.5.1.1	ICMP	74	Destination unreachable (Port unreachable)
451	21:00:35.327932	192.5.2.1	192.5.1.1	ICMP	74	Destination unreachable (Port unreachable)
452	21:00:35.338663	192.5.2.1	192.5.1.1	ICMP	74	Destination unreachable (Port unreachable)

גודל ה payload = 1024 bytes, נשלחו בסה"כ 10 (לא כולל הפתיח/סוף) הודעות UDP.
לכל datagram נשלחה הודעה אחת.

2.8.

כל ההדרים אותו דבר : 2 פקטות לדוגמא:



2.9.

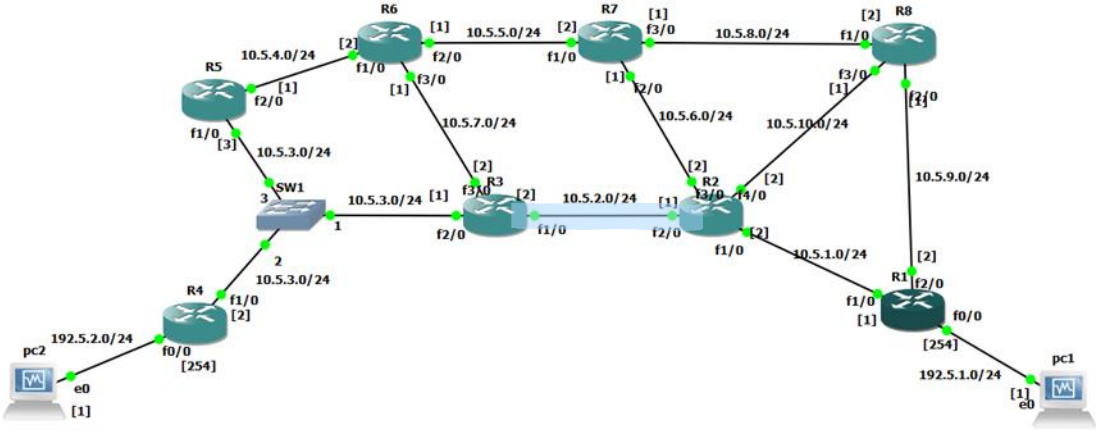
הפקודה ttcp מגרילה מספר.

2.10.

סה"כ יש בהודעה 1066 byte , מתוכם יש 1024 byte של data.
כלומר כל ההדר (ip, ethernet, udp) מוסיפים $1066 - 1024 = 42_{byte}$
ויש 10 הודעות כאלה זהות לכל המידע.

3. Transmitting data with TCP

3.1.



3.2. Start Wireshark on R3 – R2, set filter to display packets that include:

Ip == 192.5.2.1

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

ttcp –r –l1024 –n10 –p4444

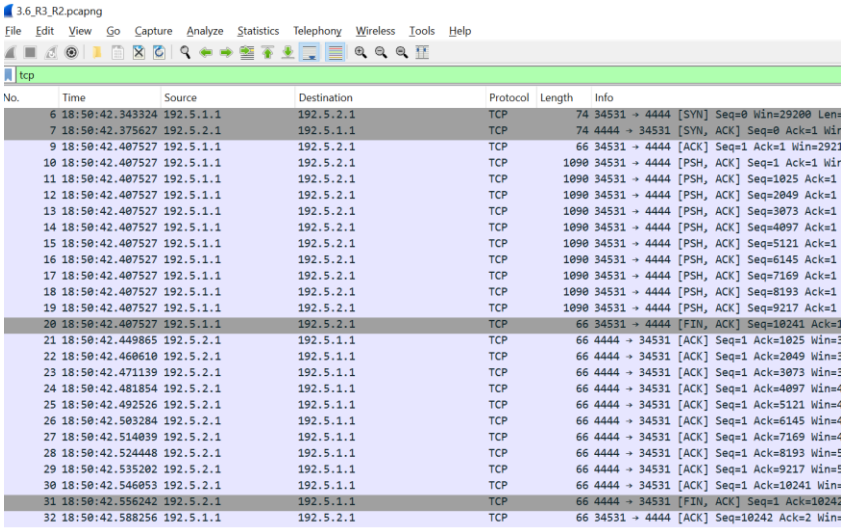
3.4. **On PC1**, start a ttcp sender that transmits UDP traffic by typing:

ttcp –t –l1024 –n10 –p4444 –D 192.5.2.1

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

3.6.

3 לחיצת ידיים, 10 פקטות מידע, ועוד 10 פקטות ACK ועוד 2 של ACK-FIN לסגירה TCP נשלחו



No.	Time	Source	Destination	Protocol	Length	Info
6	18:50:42.343324	192.5.1.1	192.5.2.1	TCP	74	34531 → 4444 [SYN] Seq=0 Win=29200 Len=
7	18:50:42.375627	192.5.2.1	192.5.1.1	TCP	74	4444 → 34531 [SYN, ACK] Seq=0 Ack=1 Wi
9	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	66	34531 → 4444 [ACK] Seq=1 Ack=1 Win=292
10	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	1090	34531 → 4444 [PSH, ACK] Seq=1 Ack=1 Wi
11	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	1090	34531 → 4444 [PSH, ACK] Seq=1025 Ack=1
12	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	1090	34531 → 4444 [PSH, ACK] Seq=2049 Ack=1
13	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	1090	34531 → 4444 [PSH, ACK] Seq=3073 Ack=1
14	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	1090	34531 → 4444 [PSH, ACK] Seq=4097 Ack=1
15	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	1090	34531 → 4444 [PSH, ACK] Seq=5121 Ack=1
16	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	1090	34531 → 4444 [PSH, ACK] Seq=6145 Ack=1
17	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	1090	34531 → 4444 [PSH, ACK] Seq=7169 Ack=1
18	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	1090	34531 → 4444 [PSH, ACK] Seq=8193 Ack=1
19	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	1090	34531 → 4444 [PSH, ACK] Seq=9217 Ack=1
20	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	66	34531 → 4444 [FIN, ACK] Seq=10241 Ack=
21	18:50:42.449865	192.5.2.1	192.5.1.1	TCP	66	4444 → 34531 [ACK] Seq=1 Ack=1025 Win=
22	18:50:42.460610	192.5.2.1	192.5.1.1	TCP	66	4444 → 34531 [ACK] Seq=1 Ack=2049 Win=
23	18:50:42.471139	192.5.2.1	192.5.1.1	TCP	66	4444 → 34531 [ACK] Seq=1 Ack=3073 Win=
24	18:50:42.481854	192.5.2.1	192.5.1.1	TCP	66	4444 → 34531 [ACK] Seq=1 Ack=4097 Win=
25	18:50:42.492526	192.5.2.1	192.5.1.1	TCP	66	4444 → 34531 [ACK] Seq=1 Ack=5121 Win=
26	18:50:42.503284	192.5.2.1	192.5.1.1	TCP	66	4444 → 34531 [ACK] Seq=1 Ack=6145 Win=
27	18:50:42.514039	192.5.2.1	192.5.1.1	TCP	66	4444 → 34531 [ACK] Seq=1 Ack=7169 Win=
28	18:50:42.524448	192.5.2.1	192.5.1.1	TCP	66	4444 → 34531 [ACK] Seq=1 Ack=8193 Win=
29	18:50:42.535202	192.5.2.1	192.5.1.1	TCP	66	4444 → 34531 [ACK] Seq=1 Ack=9217 Win=
30	18:50:42.546053	192.5.2.1	192.5.1.1	TCP	66	4444 → 34531 [ACK] Seq=1 Ack=10241 Win=
31	18:50:42.556242	192.5.2.1	192.5.1.1	TCP	66	4444 → 34531 [FIN, ACK] Seq=1 Ack=10242
32	18:50:42.588256	192.5.1.1	192.5.2.1	TCP	66	34531 → 4444 [ACK] Seq=10242 Ack=2 Win=

3.7. Sender – initial sequence number = 34 28 ce eb = 875089643

```
Sequence Number: 0    (relative sequence number)
Sequence Number (raw): 875089643
[Next Sequence Number: 1    (relative sequence number)]
Acknowledgment Number: 0

c4 03 07 1c 00 10 c4 02  50 b4 00 20 08 00 45 00   ........ P... .
00 3c ce 30 40 00 3e 06  eb 7e c0 05 01 01 c0 05   .<.0@.>. .~....
02 01 86 e3 11 5c 34 28  ce eb 00 00 00 00 a0 02   ......\4( ......
72 10 b6 b5 00 00 02 04  05 b4 04 02 08 0a 00 01   r....... ......
00 db 00 00 00 00 01 03  03 05                     ........ ..
```

Receiver - initial sequence number = 1f 94 ec 9d = 529853597

```
Sequence Number: 0    (relative sequence number)
Sequence Number (raw): 529853597
[Next Sequence Number: 1    (relative sequence number)]
Acknowledgment Number: 1    (relative ack number)
Acknowledgment number (raw): 875089644

c4 02 50 b4 00 20 c4 03  07 1c 00 10 08 00 45 00   ..P... ...... E
00 3c 00 00 40 00 3e 06  b9 af c0 05 02 01 c0 05   .<..@.>. .......
01 01 11 5c 86 e3 1f 94  ec 9d 34 28 ce ec a0 12   ...\.... ..4(....
71 20 7e c2 00 00 02 04  05 b4 04 02 08 0a 00 01   q ~..... .......
2c 9f 00 01 00 db 01 03  03 05                     ,....... ..
```

The rage: $[0, 2^{32} - 1],\ 32\ bit$

3.8. sequence number = 875089644

זה לא 0 בגלל שהתחלנו מ ISN שהגרלנו בהודעת SYN הראשונה ואז התקדמנו בהתאם לכמות הבייטים שנשלחו בחבילה

```
Sequence Number: 1    (relative sequence number)
Sequence Number (raw): 875089644
[Next Sequence Number: 1025    (relative sequence number)]
Acknowledgment Number: 1    (relative ack number)
Acknowledgment number (raw): 529853598
1000 .... = Header Length: 32 bytes (8)

0020  02 01 86 e3 11 5c 34 28  ce ec 1f 94 ec 9e 80 18   ......\4( ......
```

3.9. 1. sender – SIN

השולח שולח הודעה שהוא רוצה להתחבר למקבל

2. receiver – SIN ACK

המקבל מאשר את הבקשה מהשולח, ושולח בקשת התחברות מעצמו

3. sender - ACK

השולח שולח אישור על הצעת החברות מהמקבל

3.10. packets do not carry a payload:

$$3_{3\ hand\ shake} + 10_{ack} + 3_{fin} = 16$$

3.11. flag:

SYN – בקשת התחברות

ACK – אישור על קבלת הודעה

PSH – להגיד למקבל שיעביר את המידע ישר לאפליקציה

FIN - בקשת התנתקות

	Time	Source	Destination	Protocol	Length	Info
6	18:50:42.343324	192.5.1.1	192.5.2.1	TCP	74	34531 → 4444 [SYN] Seq=0
7	18:50:42.375627	192.5.2.1	192.5.1.1	TCP	74	4444 → 34531 [SYN, ACK]
9	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	66	34531 → 4444 [ACK] Seq=1
10	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	1090	34531 → 4444 [PSH, ACK]
11	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	1090	34531 → 4444 [PSH, ACK]
12	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	1090	34531 → 4444 [PSH, ACK]
13	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	1090	34531 → 4444 [PSH, ACK]
14	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	1090	34531 → 4444 [PSH, ACK]
15	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	1090	34531 → 4444 [PSH, ACK]
16	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	1090	34531 → 4444 [PSH, ACK]
17	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	1090	34531 → 4444 [PSH, ACK]
18	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	1090	34531 → 4444 [PSH, ACK]
19	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	1090	34531 → 4444 [PSH, ACK]
20	18:50:42.407527	192.5.1.1	192.5.2.1	TCP	66	34531 → 4444 [FIN, ACK]
21	18:50:42.449865	192.5.2.1	192.5.1.1	TCP	66	4444 → 34531 [ACK] Seq=1
22	18:50:42.460610	192.5.2.1	192.5.1.1	TCP	66	4444 → 34531 [ACK] Seq=1
23	18:50:42.471139	192.5.2.1	192.5.1.1	TCP	66	4444 → 34531 [ACK] Seq=1
24	18:50:42.481854	192.5.2.1	192.5.1.1	TCP	66	4444 → 34531 [ACK] Seq=1
25	18:50:42.492526	192.5.2.1	192.5.1.1	TCP	66	4444 → 34531 [ACK] Seq=1
26	18:50:42.503284	192.5.2.1	192.5.1.1	TCP	66	4444 → 34531 [ACK] Seq=1
27	18:50:42.514039	192.5.2.1	192.5.1.1	TCP	66	4444 → 34531 [ACK] Seq=1
28	18:50:42.524448	192.5.2.1	192.5.1.1	TCP	66	4444 → 34531 [ACK] Seq=1
29	18:50:42.535202	192.5.2.1	192.5.1.1	TCP	66	4444 → 34531 [ACK] Seq=1
30	18:50:42.546053	192.5.2.1	192.5.1.1	TCP	66	4444 → 34531 [ACK] Seq=1
31	18:50:42.556242	192.5.2.1	192.5.1.1	TCP	66	4444 → 34531 [FIN, ACK]
32	18:50:42.588256	192.5.1.1	192.5.2.1	TCP	66	34531 → 4444 [ACK] Seq=1

3.12. overhead:

$$74 * 2 + 1090 * 10 + 66 * 14 = 11{,}972_{total\ byte}$$

$$1024 * 10 = 10{,}240_{data\ byte}$$

$$11{,}972 - 10{,}240 = 1732_{overhead}$$

3.13. sender:

התחלנו ב 875,089,643 והגענו ל 875,099,885 – ההפרש: 10,242.

המספר הסידורי מצביע בכל פעם על הבייט הבא ברצף בייטי המידע שצריכים להישלח

נשים לב שהודעות FIN,SYN נספרות כל אחת כ 1 בייט מרצף הדתא.

בהתחלה מצביע על הבייט הראשון, שהוא SYN

אז אחרי הודעת SYN המספר יצביע על הבייט הבא, כלומר יעלה באחד

אחרי הודעת דתא, שנשלחו 1024 בייטים המספר יעלה ב1024 ובסוף להודעת הפין יעלה באחד שוב

```
Acknowledgment Number: 10242    (relative ack number)
Acknowledgment number (raw): 875099885
1000 .... = Header Length: 32 bytes (8)
> Flags: 0x011 (FIN, ACK)

0000  c4 02 50 b4 00 20 c4 03  07 1c 00 10 08 00 45 00   ··P·· ·· ····· E·
0010  00 34 c0 3b 40 00 3e 06  f9 7b c0 05 02 01 c0 05   ·4·;@·>· ·{····· 
0020  01 01 11 5c 86 e3 1f 94  ec 9e 34 28 f6 ed 80 11   ···\···· ··4(····
```

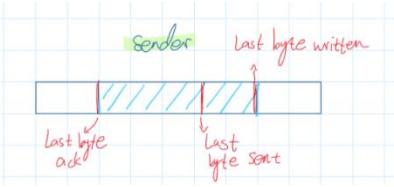
Receiver:

המקבל אין לו דתא לשלוח, אז הוא מגריל מספר בהתחלה, ואז שולח SYN ומעלה ב 1 בייט את המספר שלו, ואז שולח אקים שלא מעלים כלום ובסוף שולח FIN שמסיים את הכל, אז סה"כ רק עלינו במספר אחד.

```
Sequence Number (raw): 529853598
```

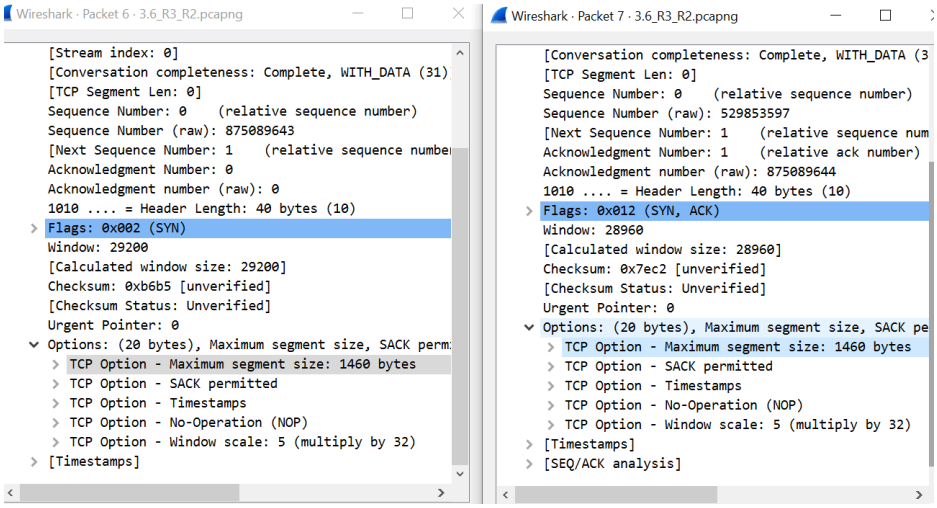
3.14. Window:

Window- כל אחד מפרסם את גודל החלון שנשאר לו, כלומר כמה מקום עוד יש לו בבאפר לקבל מידע. (עד שלא יגיע ACK החלון לא יזוז ולאט לאט יגמרו החבילות שניתן לשלוח בחלון)



3.15.

בשלב הלחיצת ידיים שולחים כל אחד מהמחשבים את גודל הסיגמנט שהם מוכנים לקבל בכל חבילת TCP.



4. Comparing basic UDP and TCP data transfers

4.1.

4.2.

ב udp – 420

ב tcp - 1676

כלומר ב TCP יש יות מידע מסביב

4.3.

Tcp – 1090 אורך ה segment 1056 בייטים: 1. Source port – 2 2. Destination port – 2 3. Sequence number– 4 4. Ack – 4 5. Header length – 1 6. Flag- 1 7. Window – 2 8. Checksum – 2 9. Urgent pointer – 2 10. Options – 12 11. Payload– 1024	Udp – 1066 אורך ה datagram הוא 1032 בייטים: 1. Source port – 2 2. Destination port – 2 3. Length – 2 4. Checksum- 2 5. Payload – 1024

4.4. Flow control

TCP:

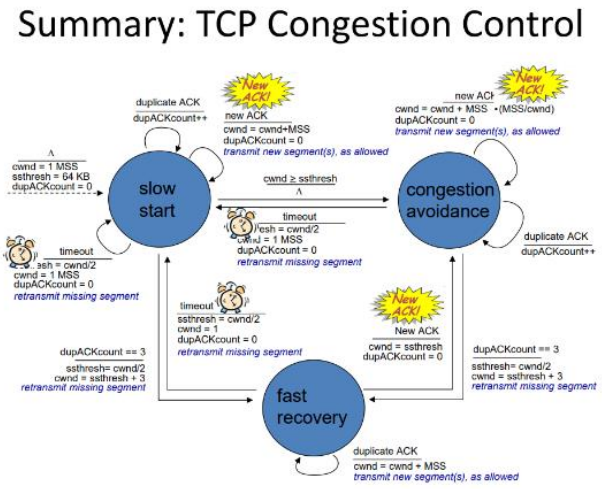
Flow control זה מה שקובע את גודל הפקטות ששולחים, מושפע מהגודל חלון, ונקבע בהתאם אליו כדי למנוע גודש על הערוץ, או גודש על הבאפר של המקבל.

משתמש ב slow start, congestion avoideance, fast retransmit

Slow start– מעלה בתחילה את גודל החלון בצורה אקספוננציאלית על כל ACK שמגיע עד שיש התנגשות או שמגיע ל טרש הולד (איזה גבול עליון של גודל חלון)

Congestion avoidance – אם הגענו לטרש הולד, אז נעבור לעליה עדינה בצורה ליניארית

Fast recovery – אם הייתה התנגשות, נגיע למצב זה בו נקטין את החלון בחצי, ואז נחזור לעליה ליניארית.



UDP:

אין לו flow control, אז הוא פשוט שולח הודעות לפי ה MTU – גודל מקסימלי של הודעות בהינתן רשת מסוימת. מחלק מראש את כל החבילות לגודל הזה.

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

5. TCP Fairness and advanced Wireshark statistics

5.1.

5.2.

5.3. The first PC - **PC_RCV** will be used to receive two TCP streams.
The two other PCs - **PC_TX1, PC_TX2** are used to send them.

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

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 (לתת הרשאות)

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

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.