רשתות תקשורת מטלה 3 מגישים:

322770629314990938

נוסיף את החלק הראשון של המטלה:

Part A

Algorithm: Reno - sender & receiver

0% packet loss

```
barye@barye-VirtualBox:~/from git/CN_EX3/communications_networbarye@barye-VirtualBox:~/from git/CN_EX3/communications_networbarye@barye-VirtualBox:~/from git/CN_EX3/communications_networbarye@barye-VirtualBox:~/from git/CN_EX3/communications_networbarye@barye-VirtualBox:~/from git/CN_EX3/communications_networbarye@barye-VirtualBox:~/from git/CN_EX3/communications_networbarye@barye-VirtualBox:~/from git/CN_EX3/communications_networbarye@barye-VirtualBox:~/from git/CN_EX3/communications_networbarye@barye-VirtualBox:~/from git/CN_EX3/communications_networparye@barye-VirtualBox:~/from git/CN_EX3/communications_networparyedorparye-VirtualBox:~/from git/
```

- * Statistics * -

Run #1 Data: Time=1.568000 S; Speed=1275.510204 MB/S

Run #2 Data: Time=0.534000 S; Speed=3745.318352 MB/S

Run #3 Data: Time=0.785000 S; Speed=2547.770701 MB/S

Run #4 Data: Time=0.697000 S; Speed=2869.440459 MB/S

Run #5 Data: Time=0.651000 S; Speed=3072.196621 MB/S

Average time: 0.847000 S

Average speed: 2702.047267 S

Sender 127.0.0.1:45026 disconnected

```
barye@barye-VirtualBox:-/from git/CN_EX3/communications_networ

k EX3$ sudo to qdisc change dev lo root netem loss 2%
barye@barye-VirtualBox:-/from git/CN_EX3/communications_networ

k EX3$.7/CP Receiver -p 5060 -algo reno

Stating Receiver...
Waiting for TCP connection...
Sender 127.0.0.1:34156 connected, beginning to receive file...
File transfer completed.
File trans
```

Statistics

Run #1 Data: Time=1.731000 S; Speed=1155.401502 MB/S

Run #2 Data: Time=0.599000 S; Speed=3338.898164 MB/S

Run #3 Data: Time=0.764000 S; Speed=2617.801047 MB/S

Run #4 Data: Time=4.465000 S; Speed=447.928331 MB/S

Run #5 Data: Time=0.524000 S; Speed=3816.793893 MB/S

Average time: 1.616600 S

Average speed: 2275.364587 S

Sender 127.0.0.1:34156 disconnected

```
barye@barye-VirtualBox:-/from git/CN_EX3/communications_network EX3s sudo to a disc change dev lo root netem loss 5% barye@barye-VirtualBox:-/from git/CN_EX3/communications_network EX3s ./TCP_Receiver .p 5060 -algo reno
Stating Receiver..

Sender 127.0.0.1:46214 connected, beginning to receive file...
File transfer completed.
File transfer comp
```

- * Statistics * -

Run #1 Data: Time=2.574000 S; Speed=777.000777 MB/S

Run #2 Data: Time=0.471000 S; Speed=4246.284501 MB/S

Run #3 Data: Time=0.419000 S; Speed=4773.269690 MB/S

Run #4 Data: Time=1.180000 S; Speed=1694.915254 MB/S

Run #5 Data: Time=0.530000 S; Speed=3773.584906 MB/S

Average time: 1.034800 S

Average speed: 3053.011026 S

Sender 127.0.0.1:46214 disconnected

```
barye@barye-VirtualBox:-/from git/CN_EX3/communications_networ

k_EX3$ sudo tc qdisc change dev lo root netem loss 10%
barye@barye-VirtualBox:-/from git/CN_EX3/communications_networ

k_EX3$ ./TCP_Receiver .p 5060 -algo reno

Stating Receiver ..
Waiting for TCP connection...
Sender 127.0.0.1:59202 connected, beginning to receive file...
File transfer completed.
File tra
```

- * Statistics * -

Run #1 Data: Time=1.165000 S; Speed=1716.738197 MB/S

Run #2 Data: Time=0.473000 S; Speed=4228.329810 MB/S

Run #3 Data: Time=0.838000 S; Speed=2386.634845 MB/S

Run #4 Data: Time=0.833000 S; Speed=2400.960384 MB/S

Run #5 Data: Time=0.757000 S; Speed=2642.007926 MB/S

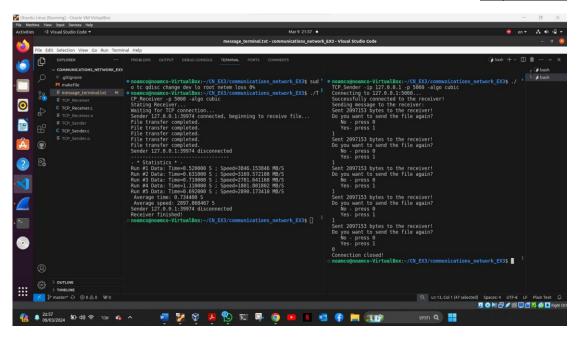
Average time: 0.813200 S

Average speed: 2674.934232 S

Sender 127.0.0.1:59202 disconnected

Algorithm: cubic - sender & receiver

0% packet loss



- * Statistics * -

Run #1 Data: Time=0.520000 S; Speed=3846.153846 MB/S

Run #2 Data: Time=0.631000 S; Speed=3169.572108 MB/S

Run #3 Data: Time=0.719000 S; Speed=2781.641168 MB/S

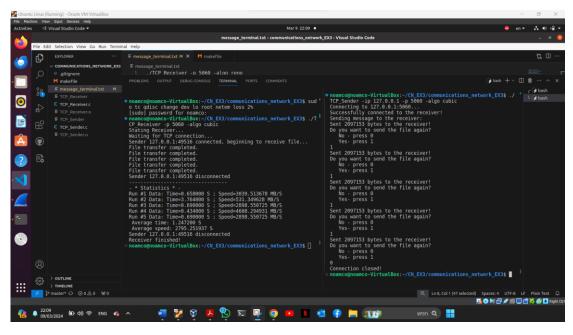
Run #4 Data: Time=1.110000 S; Speed=1801.801802 MB/S

Run #5 Data: Time=0.692000 S; Speed=2890.173410 MB/S

Average time: 0.734400 S

Average speed: 2897.868467 S

Sender 127.0.0.1:39974 disconnected



- * Statistics * -

Run #1 Data: Time=0.658000 S; Speed=3039.513678 MB/S

Run #2 Data: Time=3.764000 S; Speed=531.349628 MB/S

Run #3 Data: Time=0.690000 S; Speed=2898.550725 MB/S

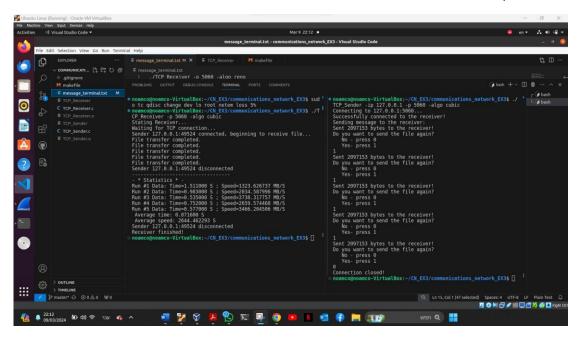
Run #4 Data: Time=0.434000 S; Speed=4608.294931 MB/S

Run #5 Data: Time=0.690000 S; Speed=2898.550725 MB/S

Average time: 1.247200 S

Average speed: 2795.251937 S

Sender 127.0.0.1:49516 disconnected



- * Statistics * -

Run #1 Data: Time=1.511000 S; Speed=1323.626737 MB/S

Run #2 Data: Time=0.983000 S; Speed=2034.587996 MB/S

Run #3 Data: Time=0.535000 S; Speed=3738.317757 MB/S

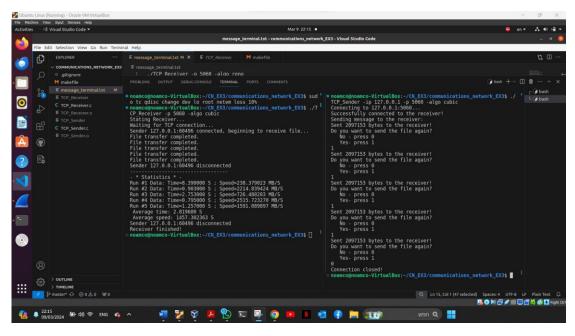
Run #4 Data: Time=0.752000 S; Speed=2659.574468 MB/S

Run #5 Data: Time=0.577000 S; Speed=3466.204506 MB/S

Average time: 0.871600 S

Average speed: 2644.462293 S

Sender 127.0.0.1:49524 disconnected



- * Statistics * -

Run #1 Data: Time=8.390000 S; Speed=238.379023 MB/S

Run #2 Data: Time=0.903000 S; Speed=2214.839424 MB/S

Run #3 Data: Time=2.753000 S; Speed=726.480203 MB/S

Run #4 Data: Time=0.795000 S; Speed=2515.723270 MB/S

Run #5 Data: Time=1.257000 S; Speed=1591.089897 MB/S

Average time: 2.819600 S

Average speed: 1457.302363 S

Sender 127.0.0.1:60496 disconnected

Bonus:

Algorithm: receiver- cubic, - sender- reno

0% packet loss

```
barye@barye-VirtualBox:-/from git/CN_EX3/communications_network_
EX35 sudo tc daise change dev lo root netem loss 0%
barye@barye-VirtualBox:-/from git/CN_EX3/communications_network_
EX35 ./TCP.Receiver -p 5060 -algo cubic
Stating Receiver...
Waiting for TCP connection...
Sender 127.0.0.1:49866 connected, beginning to receive file...
File transfer completed.
File transf
```

- * Statistics * -

Run #1 Data: Time=0.576000 S; Speed=3472.222222 MB/S

Run #2 Data: Time=0.711000 S; Speed=2812.939522 MB/S

Run #3 Data: Time=0.571000 S; Speed=3502.626970 MB/S

Run #4 Data: Time=0.833000 S; Speed=2400.960384 MB/S

Run #5 Data: Time=1.009000 S; Speed=1982.160555 MB/S

Average time: 0.740000 S

Average speed: 2834.181931 S

Sender 127.0.0.1:49866 disconnected

```
barye@barye-VirtualBox:-/from git/CN_EX3/communications_network_
EX3$ sudo tc qdisc change dev lo root netem loss 2% barye@barye-VirtualBox:-/from git/CN_EX3/communications_network_
EX3$ ./TCP Receiver -, p 5060 -algo cubic
Stating Receiver ...
Waiting for TCP connection...
Sender 127.0.0.1:56106 connected, beginning to receive file...
File transfer completed.
File tra
```

- * Statistics * -

Run #1 Data: Time=2.128000 S; Speed=939.849624 MB/S

Run #2 Data: Time=0.384000 S; Speed=5208.333333 MB/S

Run #3 Data: Time=0.546000 S; Speed=3663.003663 MB/S

Run #4 Data: Time=0.488000 S; Speed=4098.360656 MB/S

Run #5 Data: Time=0.436000 S; Speed=4587.155963 MB/S

Average time: 0.796400 S

Average speed: 3699.340648 S

Sender 127.0.0.1:56106 disconnected

- * Statistics * -

Run #1 Data: Time=0.573000 S; Speed=3490.401396 MB/S

Run #2 Data: Time=0.911000 S; Speed=2195.389682 MB/S

Run #3 Data: Time=0.407000 S; Speed=4914.004914 MB/S

Run #4 Data: Time=0.408000 S; Speed=4901.960784 MB/S

Run #5 Data: Time=0.504000 S; Speed=3968.253968 MB/S

Average time: 0.560600 S

Average speed: 3894.002149 S

Sender 127.0.0.1:57512 disconnected

```
barye@barye-VirtualBox:-/from git/CN_EX3/communications_network_barye@barye-VirtualBox:-/from git/CN_EX3/communications_network_barye@barye-VirtualBox:-/from git/CN_EX3/communications_network_Stating Receiver ... Waiting for TCP connection...
Waiting for TCP connection...
File transfer completed.
File transfer completed
```

- * Statistics * -

Run #1 Data: Time=1.988000 S; Speed=1006.036217 MB/S

Run #2 Data: Time=0.279000 S; Speed=7168.458781 MB/S

Run #3 Data: Time=0.807000 S; Speed=2478.314746 MB/S

Run #4 Data: Time=0.632000 S; Speed=3164.556962 MB/S

Run #5 Data: Time=0.410000 S; Speed=4878.048780 MB/S

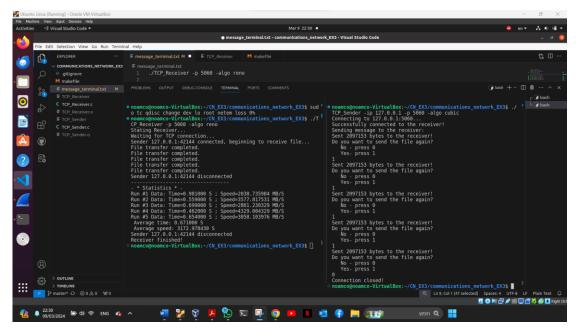
Average time: 0.823200 S

Average speed: 3739.083097 S

Sender 127.0.0.1:56976 disconnected

Algorithm: receiver- reno, - sender- cubic

0% packet loss



- * Statistics * -

Run #1 Data: Time=0.981000 S; Speed=2038.735984 MB/S

Run #2 Data: Time=0.559000 S; Speed=3577.817531 MB/S

Run #3 Data: Time=0.699000 S; Speed=2861.230329 MB/S

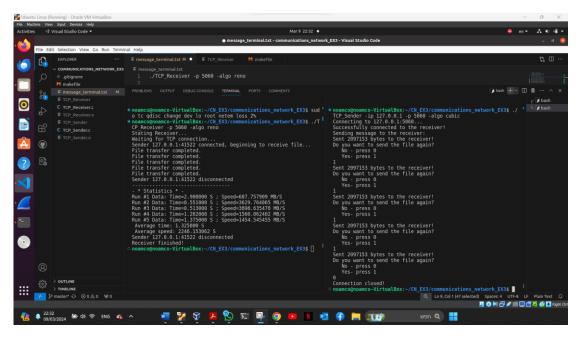
Run #4 Data: Time=0.462000 S; Speed=4329.004329 MB/S

Run #5 Data: Time=0.654000 S; Speed=3058.103976 MB/S

Average time: 0.671000 S

Average speed: 3172.978430 S

Sender 127.0.0.1:42144 disconnected



- * Statistics * -

Run #1 Data: Time=2.908000 S; Speed=687.757909 MB/S

Run #2 Data: Time=0.551000 S; Speed=3629.764065 MB/S

Run #3 Data: Time=0.513000 S; Speed=3898.635478 MB/S

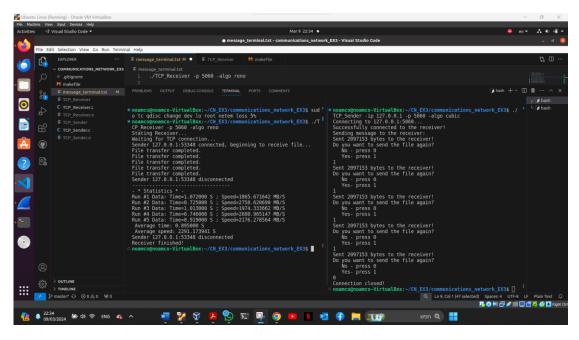
Run #4 Data: Time=1.282000 S; Speed=1560.062402 MB/S

Run #5 Data: Time=1.375000 S; Speed=1454.545455 MB/S

Average time: 1.325800 S

Average speed: 2246.153062 S

Sender 127.0.0.1:41522 disconnected



- * Statistics * -

Run #1 Data: Time=1.072000 S; Speed=1865.671642 MB/S

Run #2 Data: Time=0.725000 S; Speed=2758.620690 MB/S

Run #3 Data: Time=1.013000 S; Speed=1974.333662 MB/S

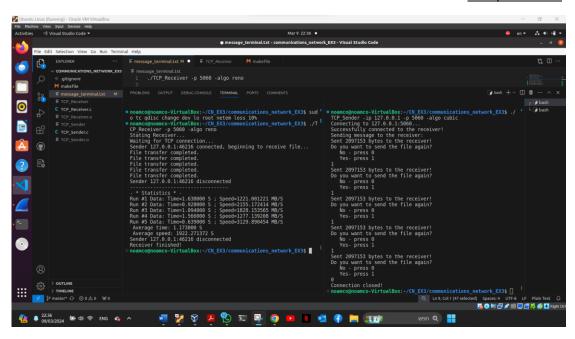
Run #4 Data: Time=0.746000 S; Speed=2680.965147 MB/S

Run #5 Data: Time=0.919000 S; Speed=2176.278564 MB/S

Average time: 0.895000 S

Average speed: 2291.173941 S

Sender 127.0.0.1:53348 disconnected



- * Statistics * -

Run #1 Data: Time=1.638000 S; Speed=1221.001221 MB/S

Run #2 Data: Time=0.928000 S; Speed=2155.172414 MB/S

Run #3 Data: Time=1.094000 S; Speed=1828.153565 MB/S

Run #4 Data: Time=1.566000 S; Speed=1277.139208 MB/S

Run #5 Data: Time=0.639000 S; Speed=3129.890454 MB/S

Average time: 1.173000 S

Average speed: 1922.271372 S

Sender 127.0.0.1:46216 disconnected

Part B

Packet loss 0%:

```
recived packet number 31
-send ack for 31
recived packet number 32
-send ack for 32
recived END packet
-send ack for packet END
got total recv: 2097153
recived FIN packet
send packet FIN ACK

-** Statistics *-
Run #1 Data: Time= 126.441000 ms; Speed= 15.817662 MB/S
Run #2 Data: Time= 28.506000 ms; Speed= 140.321403 MB/S
Run #3 Data: Time= 28.547000 ms; Speed= 2597.403836 MB/S
Run #5 Data: Time= 3.8350000 ms; Speed= 2597.403836 MB/S
Average speed: 783.462822 MB/s
Sender ip: 127.0.0.1 port: 5060 disconnected
Receiver finished!

barye@ubuntu22:-/Communication network/communications_networ

k EX3/Part Bs | 

-got ACK for packet number: 28
send packet 29
-got ACK for packet number: 30
send packet 31
-got ACK for packet number: 30
send packet 32
-got ACK for packet number: 30
send packet 131
-got ACK for packet number: 30
send packet 132
-got ACK for packet number: 30
send packet 130
-got ACK for packet number: 30
send packet 130
-got ACK for packet number: 30
send packet 130
-got ACK for packet number: 29
send packet 28
-got ACK for packet number: 29
send packet 130
-got ACK for packet number: 29
send packet 130
-got ACK for packet number: 30
send packet 130
-got ACK for packet number: 29
send packet 130
-got ACK for packet number: 29
send packet 130
-got ACK for packet number: 30
send packet 130
-got ACK for packet number: 29
send packet 130
-got ACK for packet number: 29
send packet 130
-got ACK for packet number: 30
send packet 130
-got ACK for packet number: 29
send packet 130
-got ACK for packet number: 29
send packet 130
-got ACK for packet number: 28
send packet 130
-got ACK for packet number: 28
send packet 130
-got ACK for packet number: 30
send packet 30
-got ACK for packet number: 30
send packet 130
-got ACK for packet number: 28
send packet 28
-got ACK for packet number: 30
send packet 30
-got ACK for packet number: 29
send packet 30
-got ACK for packet number: 29
send packet 30
-got ACK for packet number: 30
send packet 30
-got ACK for packet number: 30
send packet 30
-got ACK for packet number: 30
send
```

- * Statistics *

Run #1 Data: Time= 126.441000 ms; Speed= 15.817662 MB/S

Run #2 Data: Time= 28.506000 ms; Speed= 140.321403 MB/S

Run #3 Data: Time= 26.547000 ms; Speed= 226.014347 MB/S

Run #4 Data: Time= 8.531000 ms; Speed= 937.756865 MB/S

Run #5 Data: Time= 3.850000 ms; Speed= 2597.403836 MB/S

Average time: 38.775000 ms

Average speed: 783.462822 MB/S

Sender ip: 127.0.0.1 port: 5060 disconnected

Packet loss 2%:

* -Statistics- *

Run #1 Data: Time= 5.958000 ms; Speed= 335.683275 MB/S

Run #2 Data: Time= 18.283000 ms; Speed= 218.782580 MB/S

Run #3 Data: Time= 16.667000 ms ; Speed= 359.992972 MB/S

Run #4 Data: Time= 12.113000 ms; Speed= 660.447768 MB/S

Run #5 Data: Time= 14.049000 ms; Speed= 711.794773 MB/S

Average time: 13.414000 ms

Average speed: 457.340274 MB/s

Packet loss 5%:

```
barye@ubuntu22:~/Communication network/communications_networ

LEXA/Part B$ sudo tc qdisc change dev lo root netem loss 5%
barye@ubuntu22:~/Communication network/communications_network_EX3/Part B$  

//Communication network/communications_network_EX3/Part B$  

//Communication network/communications_network_EX3/Part B/RUDP_API.o・Untracked
```

```
-send ack for packet END
got total recv: 2097153
recived FIN packet
send packet FIN ACK
-** Statistics *-
Run #1 Data: Time= 40.875000 ms; Speed= 48.9296
87 MB/S
Run #2 Data: Time= 58.324000 ms; Speed= 68.5824
34 MB/S
Run #3 Data: Time= 18.888000 ms; Speed= 317.662
159 MB/S
Run #4 Data: Time= 11.445000 ms; Speed= 698.995
528 MB/S
mmunication network/communications_network_EX3/Part B/RUDP_Sender.c

Average time: 32.755800 ms
Average speed: 285.233265 MB/s
Sender ip: 127.0.0.1 port: 5060 disconnected
Receiver finished!
barye@ubuntu22:-/Communication network/communication
tions_network_EX3/Part B$

send packet 28
-got ACK for packet number: 28
send packet 29
recivefrom() FAILED1send packet 29
-got ACK for packet number: 30
send packet 30
-got ACK for packet number: 31
send packet 32
-got ACK for packet number: 32
send packet END
Send packet END
Send packet FIN
Sol ACK for packet number: 30
send packet 31
-got ACK for packet number: 30
send packet 31
-got ACK for packet number: 30
send packet 31
-got ACK for packet number: 30
send packet 31
-got ACK for packet number: 30
send packet FIN
Send packet Sall FIN
Send packet Sall
```

* -Statistics- *

Run #1 Data: Time= 40.875000 ms; Speed= 48.929687 MB/S

Run #2 Data: Time= 58.324000 ms; Speed= 68.582434 MB/S

Run #3 Data: Time= 18.888000 ms; Speed= 317.662159 MB/S

Run #4 Data: Time= 11.445000 ms; Speed= 698.995528 MB/S

Run #5 Data: Time= 34.247000 ms; Speed= 291.996518 MB/S MB/S

Average time: 32.755800 ms

Average speed: 285.233265 MB/S

Sender ip: 127.0.0.1 port: 5060 disconnected

Packet loss 10%:

```
-send ack for 23
reclived packet number 24
reclived packet number 25
-send ack for 25
reclived packet number 26
-send ack for 25
-send ack for 25
-send ack for 26
-send ack for 26
-send ack for 26
-send ack for 27
-reclived packet number 26
-send ack for 27
-reclived packet number 27
-send ack for 27
-send ack for 27
-reclived packet number 27
-send ack for 27
-reclived packet number 28
-reclived packet number 27
-send ack for 27
-reclived packet number 28
-reclived packet number 27
-send ack for 27
-reclived packet number 28
-reclived packet number 29
-send ack for 29
-reclived packet number 30
-send ack for 30
-reclived packet number 30
-send ack for 30
-reclived packet number 31
-send ack for 30
-reclived packet number 32
-send ack for 30
-reclived packet 180
-reclived packet number 32
-send ack for 30
-reclived packet number 32
-send ack for 30
-reclived packet number 32
-send ack for 30
-reclived packet 180
-reclived packet number 32
-send ack for 30
-reclived packet 180
-reclived packet number 32
-send ack for 30
-reclived packet 180
-reclived packet 180
-reclived packet 180
-reclived packet 180
-reclived packet number 32
-send ack for 30
-reclived packet 180
-reclived packet 18
```

*Statistics- *

Run #1 Data: Time= 21.517000 ms; Speed= 92.949805 MB/S

Run #2 Data: Time= 17.946000 ms; Speed= 222.891001 MB/S

Run #3 Data: Time= 7.469000 ms; Speed= 803.320774 MB/S

Run #4 Data: Time= 18.211000 ms; Speed= 439.295141 MB/S

Run #5 Data: Time= 17.411000 ms; Speed= 574.349823 MB/S

Average time: 16.510800 ms

Average speed: 426.561309 MB/s

שונים ראינו הבדלים בזמנים.

After you've gathered the data set for both TCP and Reliable UDP, answer the following questions:

- 1) In TCP, which congestion control algorithm gave better results overall? TCP Reno or TCP Cubic? Which one gave better results on high packet loss? Explain how you came to this conclusion based on the data set you've gathered. (5 points)
- 2) How did your implementation of Reliable UDP perform overall compared to regular TCP? Which one is better for high packet loss? (5 points)
- 3) According to the data you've gathered, in which scenarios and applications would we prefer to use TCP and in which situations would we prefer to use reliable UDP? Explain your answer. (5 points)
- 1. לפי ההרצות שביצענו ניתן לראות שבאחוז איבוד פקטות הכי גבוה (10%) יש הבדל משמעותי במהירות בין cubic ,tcp cubic מסוג tcp reno לסוג cubic ,tcp cubic מהיר יותר. וככל שהאחוזי איבוד יורדים ההבדלים יותר קטנים.
 באחוז איבוד של 5% ראינו מגמה הפוכה אך אנו חושבים שבגלל שההרצות בוצעו ממחשבים
- 2. מימשנו rudp על ידי פרוטוקול stop&wait (תמיד מחכים לack). לפי ההרצות שביצענו ניתן לראות שמהירות הtcp גבוה יותר ממהירות הrudp הן ע"י שימוש באלגוריתם reno והן ע"י שימוש באלגוריתם cubic.
- נראה שrudp שמימשנו איטי יותר מקסד כיוון שכתבנו rudp בצורה פשוטה ויחסית מנוונת cubic ו reno הממומשים בקסד ומבצעים בקרות זרימה. (לדוגמה הגדלת חלון השליחה והקטנתו בהתאם למצב התעבורה ברשת)
 - 3. כשמסתכלים על תוצאות ההרצות שביצענו רואים שtcp מהיר יותר מהקוח. בנוסף העלחות אמינות (לדוגמה חישוב rudp שמימשנו ממומש בצורה פשוטה וללא בדיקות המבטיחות אמינות (לדוגמה חישוב rudp rudp בצורה מתוחכמת יותר ובקרה על המידע המועבר). נוסף על כך, מימוש מסובך יותר. לכן במקרה זה נעדיף להתשמש בTCP.

בהתאם לנלמד בקורס נעדיף את הrudp כיוון שהוא מורכב מdpu שבד"כ מהיר יותר מקבהתאם לנלמד בקורס נעדיף את הreliable) הוא גם נהיה אמין כמו tcp ולכן בסה"כ אמין כמו tcp וכאשר מוסיפים לו את האמינות (rudp). rudp הוא גם נהיה אמין כמו

בחרנו בתוכנית שלנו לתמוך <u>בipv4</u>

נפרט על הקוד:

פירוט על RUDP_API כפי שמופיע בקובץ.

"הלקוח והשרת משתמשים בפקודות אלה כמו "בקופסה שחורה

```
* @return the socket number if the socket was created successfully,
* -1 if it failed.
int rudp_sockets();
* opening connection between two peers.
* @return 1 if the connection was successful, -1 otherwise.
int RUDP_connect_reciever(int sock, int port);
int RUDP_connect_sender(int sock, char* ip ,int port);
*Sending data to the peer. The function should wait for an
int rudp send(int sock, const void *user data, size t size D,int*arr);
int rudp recv(int sock, int data_size,int*arr);
int rudp close(int sock);
```

header structure:

```
int length_data;
int sequence_number;
int checksum;
int flags;
char data[Buffer];
}header , *pheader;
```

: עם 5 שדות header בחרנו לממש

בחרנו שיהיה גודל קבוע של 65000 כגודל packet גודל המידע בכל -Length_data.1 והדבל המידע בכל BUFFER האחרונה שנישלח תהיה בגודל שנשאר לאחר הפחתת הפקטות שנשלחו מגודל הfile.

packets בקרה על הequence number.2 כעין מספר סידורי על כל packet שמאפשר בקרה על -sequence number.2 מיוחדת שנתנו להם שהגיעו לצד השני וגם על סדר שליחת וקבלת ack בנוסף יש הpackets מיוחדת שנתנו להם packet fin מיוחד כמו packet fin כשהקליינט רוצה לסיים הקשר.

packet finack- אישור של הרסיבר שקיבל את ההודעה לסיום הקשר. packet end – פאקטה המציינת שסיימנו לקבל את הקובץ.

- לקחנו את פונקציית הבדיקה שניתנה לנו data שלנו(לקחנו את פונקציית הבדיקה שניתנה לנו -checksum.3 במטלה)בודקת האם המידע עבר נכון ובשלמותו.
- -flags.4 כמה סוגי דגלים שהגדרנו שיעזרו לנו בלוגיקת החיבור בין שני צדדים ברשת.
 - .buffer המידע עצמו שמועבר בכל פאקטה כגודל--data.5

:flags הסבר על

```
#define SYN 1
#define ACK 2
#define DATA 3
#define FIN 4
#define FIN_ACK 5
#define END 6
```

של פתיחת קשר. -SYN של פתיחת קשר.

. של אישור על קבלת מידע -ACK של המורה על packet דגל המורה

.data עם packet דגל המורה על -<u>DATA</u>

. של סיום קשר של צד אחד -FIN packet דגל המורה על

. דגל המורה על packet שמסכמת את סיום הקשר של הצד השני - FIN_ACK

. שמסיימת את שליחת הacket בסיבוב זה -END

Timeout

```
int set_timeout(int socket, int time) {\frac{1}{2}}// set timeout for the socket
    struct timeval timeout;
    timeout.tv_sec = time;
    timeout.tv_usec = 0;

if (setsockopt(socket, SOL_SOCKET, SO_RCVTIMEO, &timeout, sizeof(timeout)) < 0) {
        perror("setsockopt() FAILED");
        return FAIL;
    }
    return SUCCESS;</pre>
```

השתמשנו בפונקציית set timeout) קראנו באינטרנט עליה (קראנו באינטרנט timeout) מכניסה לtimeout למשל בפונקציית receive from שהיא "פונקציה תוקעת" אפשר להגדיר שאחרי זמן מסוים היא תצא מהתקיעה

למשל כאן בפונקציית rudp_recv כאשר התקבל כל המדשל מחנו נותנים למשתמש זמן ארוך לבחור האם הוא רוצה לסיים את הקשר(בלחיצת 0 מתחיל תהליך הסגירה) או לקבל את הדאטה שוב ע"י לחיצת 1.

```
int rudp_send(int sock, const void* user_data, size_t size_D, int* arr)

set_timeout(sock, TIME_OUT * 3);
  int sq = 0;
  int total_bytes_sent = 0;
  int packet_count = (size_D / Buffer);
```

ack של packet זה עזר לנו למשל כאשר rudp_send כאן הגדרנו time_out בפונקציית timeout בפונקציית נקבל timeout נשלחה ולא הגיע יש

```
if (packetRecv.flags == SYN)
{
    header send_ack;
    memset(&send_ack, 0, sizeof(send_ack));
    send_ack.flags = ACK;
    int sendResult = sendto(sock, &send_ack, sizeof(header), 0, NULL, 0);
    if (sendResult == -1)
    {
        printf("sendto() FAILED");
        return FAIL;
    }
    set_timeout(sock, TIME_OUT * 10);
}
return 1;
```

RUDP_connect_reciever

```
int RUDP_connect_sender(int sock, char* ip, int port)
{
    // setup a timeout for the socket
    set_timeout(sock, TIME_OUT * 7);
    struct sockaddr_in reciver_address;
    memset(&reciver address, 0, sizeof(reciver address));
```

RUDP_connect_sender

השתמשנו גם בפונקציות אלו כדי לא לתקוע את התהליך בתחילת דרכו דהיינו בחיבור בין sender receiver

פונקציה wait_for_ack מחכה לack ואם הגענו שבזה כדי timeout) פונקציה לבצע שליחה מחדש של פקטה אם היא אובדת בדרך.

```
int wait_for_ACK(int socket, int seq_num, clock_t start_time, int timeout)
{
    header packetRCV;
    while ((double)(clock() - start_time) / CLOCKS_PER_SEC < timeout)//if time finish got timeout
    {
        int bytes_recive = recvfrom(socket, &packetRCV, sizeof(packetRCV), 0, NULL, 0);
        if (bytes_recive == -1)
        {
            printf("recivefrom() FAILED1");
            return FAIL;
        }
        if (packetRCV.sequence_number == seq_num && packetRCV.flags == ACK)
        {
            printf("-got ACK for packet number: %d\n", seq_num);
            return SUCCESS;
        }
    }
    printf("got timeout\n");
    return -1; // timout!
}</pre>
```

```
int rudp_send(int sock, const void* user_data, size_t size_D, int* arr)
   set_timeout(sock, TIME_OUT * 3);
   int sq = 0;
   int total_bytes_sent = 0;
   int packet_count = (size_D / Buffer);
   int rest_data = size_D - packet_count * Buffer;
   for (int i = 0; i < packet_count; i++)</pre>
       header packet;
       packet.length_data = Buffer;
       packet.flags = DATA;
       packet.sequence_number = sq++;
       memcpy(packet.data, user_data + i * Buffer, Buffer);
       packet.checksum = calculate_checksum(packet.data, packet.length_data);
            // Try to send the message to the server using the created socket and the server structure.
           int bytes_sent = sendto(sock, &packet, sizeof(header), 0, NULL, 0);
           printf("send packet %d\n", packet.sequence_number);
            if (bytes_sent == -1)
               printf("sendto() FAILED");
            if (arr[packet.sequence_number] == 0)
               total_bytes_sent += packet.length_data;
               arr[packet.sequence_number] = 1;
        } while (wait_for_ACK(sock, packet.sequence_number, clock(), TIME_OUT) < 0);</pre>
```

Handshake

מתבצע ע"י 2 פונקציות עיקריות

receiver משמשת לחיבור בצד RUDP_connect_reciever.1

clienta משמשת לחיבור בצד RUDP connect sender.2

פישטנו את התהליך שיהיה נוח למימוש

עוד לפני הhandshake נסביר תהליכים קצרים:

```
int RUDP_connect_reciever(int sock, int port)
{

    // Setup the server address structure.
    struct sockaddr_in serverAddress;
    memset(&serverAddress, 0, sizeof(serverAddress));
    serverAddress.sin_family = AF_INET;
    serverAddress.sin_port = htons(port);
    serverAddress.sin_addr.s_addr = htonl(INADDR_ANY);
    int bind_Sock = bind(sock, (struct sockaddr*)&serverAddress, sizeof(serverAddress));
}
```

socket porti ip עי פונקציית bind עי פונקציית socket בצד socket בצד בפתיחת האכליכים פנימיים 127.0.0.1 ולקשור שהשתמשנו בו הוא לתהליכים פנימיים 127.0.0.1 ולקשור שהשתמשנו בו הוא לתהליכים פנימיים 127.0.0.1

```
if [connect(sock, (struct sockaddr*)&reciver_address, sizeof(reciver_address)) == -1
{
    printf("connect() FAILED");
    return FAIL;
}
```

```
if (connect(sock, (struct sockaddr*)&sender_address, sizeof(sender_address)) == -1)
{
    perror("connect() FAILED");
    return FAIL;
}
```

השתמשנו בפונקציית connect בצד הsender ובreciever אשתמשנו בפונקציית udp אומנם זו לא פונקציה של udp (עליו בעצם בנינו את הrudp) אבל יש אפשרות כזאת שהיא מקשרת בין הצדדים ולא צריך לומר במפורש לאן אנו שולחים אחרי החיבור הזה

```
header packetSYN;
memset(&packetSYN, 0, sizeof(packetSYN));
packetSYN.flags = SYN;
header packetAck;
memset(&packetAck, 0, sizeof(packetAck));

if [connect(sock, (struct sockaddr*)&reciver_address, sizeof(reciver_address)) == -1]
{
    printf("connect() FAILED");
    return FAIL;
}
int bytes_SYN_sent = sendto(sock, &packetSYN, sizeof(packetSYN), 0, NULL, 0);
```

sendera לפתיחת קשר מצד SYN שלחנו פאקטת

```
if (packetRecv.flags == SYN)
{
    header send_ack;
    memset(&send_ack, 0, sizeof(send_ack));
    send_ack.flags = ACK;
    int sendResult = sendto(sock, &send_ack, sizeof(header), 0, NULL, 0);
    if (sendResult == -1)
    {
```

לאישור ack ושלחנו פאקטת reciever בצד הSYN קיבלנו את פאקטת

```
}
if [[packetAck.flags == ACK[]]
{
    printf("connected complited\n");
    return SUCCESS;
}
return SUCCESS;
}
```

אם הsender קיבל את פאקטת הack קיבל את sender

שאלות תיאורטיות

.1

ענו על השאלות הבאות (5 נקודות):

- ב"רשת אמינה" הכוונה היא לרשת שבה מעט מאוד חבילות הולכות לאיבוד. ב"קשר ארוך" הכוונה היא לקשר TCP שיש בו הרבה מאוד מידע לשלוח. מוצע להגדיל ב-TCP את SSThreshold בתחילת הקשר.
 באיזה מהמקרים הבאים השינוי הזה עשוי להועיל במידה המירבית? נמק.
 - 1. בקשר ארוך על גבי רשת אמינה עם RTT גדול.
 - 2. בקשר קצר על גבי רשת לא אמינה עם RTT גדול.
 - 3. בקשר ארוך על גבי רשת לא אמינה עם RTT גדול.
 - 4. בקשר קצר על גבי רשת אמינה עם RTT קטן.
 - .5. בקשר ארוך על גבי רשת אמינה עם RTT קטן.
 - .6. בקשר קצר על גבי רשת לא אמינה עם RTT קטן.
 - .7. בקשר ארוך על גבי רשת לא אמינה עם RTT קטן.
 - 8. בקשר קצר על גבי רשת אמינה עם RTT גדול.

במקרה 5 השינוי הזה יועיל במידה המירבית כלומר בקשר ארוך – קשר TCP שיש לו הרבה מידע במקרה 5 השינוי הזה יועיל במידה המירבית כלומר בקשר ארוך – קשר RTT גדול. נסביר-לשלוח, על גבי רשת אמינה – רשת שבה מעט מאוד חבילות הולכות לאיבוד ,עם RTT אנו יודעים שיש מידע רב לשלוח ושהרשת אמינה אם SSThreshold ישאר כמו שהוא אז בגלל שTCP אנו יודעים שיש מידע רב לשלוח ושהרשת אמינה אבלל שהלח ובזמן הזה הוא לא שולח פקטות נוספות בזמן ההמתנה הזה ובגלל שהרשת אמינה לא נגיע לSSThreshold ולכן נרוויח בהגדלת הSSThreshold.

.2

שולח ל-B חבילות מידע באמצעות קשר TCP. נתון כי הקשר מתחיל עם חלון בגודל MSS1. הקשר מסתיים בפעם הראשונה שהוא מגיע ל-SSthresh. במהלך הקשר לא אובדות חבילות. נסמן בקיצור: SSthresh=S*MSS.

נתון כי לאורך כל הקשר S*MSS < rwnd נתון כי לאורך

מהי התפוקה של הקשר מ-A ל-B (ב-Bytes / sec) בזמן הנ"ל? נמקי.

- [S*MSS / [lg (S) * RTT*2 בערך.1
- [S*MSS / [S^0.5 * RTT*2 בערך. 2
 - S*MSS / RTT בערך.3
 - S^2)*MSS / RTT) בערך.4

בתחילה נתון כי הקשר מתחיל עם חלון בגודל 1mss בתחילה נתון כי הקשר מתחיל עם חלון בגודל 2ms. בכל שליחה נכפיל את גודל החלון פי 2c, כיוון שאין איבוד חבילות, גודל השליחה לא יצטמצם אלא רק יגדל. ומכאן כמות ב $\sum_{i=0}^{\log s} 2^i = 2(s-1)$ לכן גודל החבילה הוא בערך 2s.

 $\frac{2s}{\log s} \frac{MSS}{RTT}$ ומכאן התשובה הנכונה היא 1 בערך

.3

• שתי תחנות מתקשרות באופן אלחוטי בשיטת Go Back N.

נסמן ב-X את ספרת הביקורת של הת.ז. שלך. אם שני שותפים מגישים את התרגיל יחד, X הוא ספרת הביקורת בעלת הערך הנמוך יותר מבין השתיים.

קצב התקשורת הוא Gbps8, וגודל כל חבילה הוא X*KByte. גדלי ה-headers וה-Acks זניחים.

,Km1 קצב ההתפשטות הוא $2\cdot 10^8\ m/sec$, והמרחק בין התחנות הוא זמן עיבוד הנתונים בתחנות זניח. אף חבילה ואף Ack לא הולכים לאיבוד.

- ?שלך? מהו X שלך?
- מה צריך להיות גודל חלון המשלוח כדי להבטיח תפוקה מקסימלית?

ספרת הביקורת הנמוכה מבין תעודות הזהות שלנו היא X=8.8

נחשב את זמן העיכוב (מרחק לחלק לקצב התפשטות) נחשב את זמן העיכוב (מרחק לחלק לקצב התפשטות) (2*10^8) (m/sec) = 0.5*10^(-5)(sec)

$$rac{1000}{2*10^8} = rac{1}{2*10^5}$$
שניות

:זמן שידור

$$rac{8*10^3*8}{8*10^9}=rac{8}{10^6}$$
שניות שניות
$$2*\left(rac{1}{2*10^5}+rac{8}{10^6}
ight)=0.000026$$
ולכן סה"כ RTT שנקבל:

גודל כל חבילה הוא 8kbps

קצב התקשורת 8gbps

אין אובדן פקטות ולכן יתקבל ack על כל פקטה ולא יהיו שידורים חוזרים. ולכן ניתן להגדיל את החלון כמה שיותר

$$window\ size\ = rac{RTT*}{}$$
גודל חבילה = $rac{8*10^9*0.000026}{2*8*10^3}=13$