

רשותת תקשורת - מטלה 3 - חקר Congestion Control

מגישים:

ארד בן מנשה 207083353

עפרוי לוייפר 206391054

מצורף לקובץ pdf ארבעה הקלותות וירשאך

חלק ראשון

1. תיאור המערכת

1.1. תיאור התוכנה -

התוכנה מחולקת לשני קבצי קוד של השולח ושל הרשת.

בהרצת קבצי הקוד הלקוח שלוח לשרת קובץ טקסט בגודל אחד מגה ביט, הוא עונה זאת בשני חלקים,

בחלק הראשון הוא שולח חצי ראשון של הקובץ עם אלגוריתם דחיסה מסוג TCP CUBIC, לאחר מכן לקובץ

מחכה לקבל אישור מהשרת שהוא קיבל את כל החצי הראשון, לאחר מכן מכוון להlkוח משנים את

אלגוריתם הדחיסה ל TCP RENO ואז הלקוח שלוח לשרת את החצי השני של הקובץ.

במהלך השילוחות של הקובץ הרשת ימודד זמנים ויאסוף את המידע למטרת ייעוד התוכנה.

לבסוף ניצבת החלטה בידי הלקוח האם לחזור על התהילה או לסיים את התוכנה, הלקוח שלוח את

ההחלטה של הלקוח לשרת ובהתאם לכך הלקוח והשרת חוזרים חלילה או שקוד הלקוח מסיים את הרצתו

והשרת מחכה ללקוח הבא.

ייעוד התוכנה הוא לאסוף מידע לגבי אלגוריתמי דחיסה והשוואתם אחד מול השני, בנוסף נפעיל איבוד פקודות רנדומלי אשר משפייע על מדדיות הזמן.

1.2. הפעלת התוכנה -

- יש להוריד את כל הקבצים הרלוונטיים למטלת ולודא שהם נמצאים באותה תיקייה, הקבצים

sender.c , reciever.c , makefile , 0.txt

סה"כ 4 קבצים.

- יש לפתח 2 טרמינלים בתיקייה בה נמצאים הקבצים.

- יש להריץ את הפקודות הבאות בטרמינל הראשון :

make all

./receiver

- יש להריץ את את הפקודה הבאה בטרמינל השני :

./sender

- לבסוף בהתאם לתיאור התוכנה, הלקוח התבוקש להכנס קלט האם להריץ מחדש את הקוד או לסיים להריץ, בטרמינל יהיה רשום את הודעה הבאה:

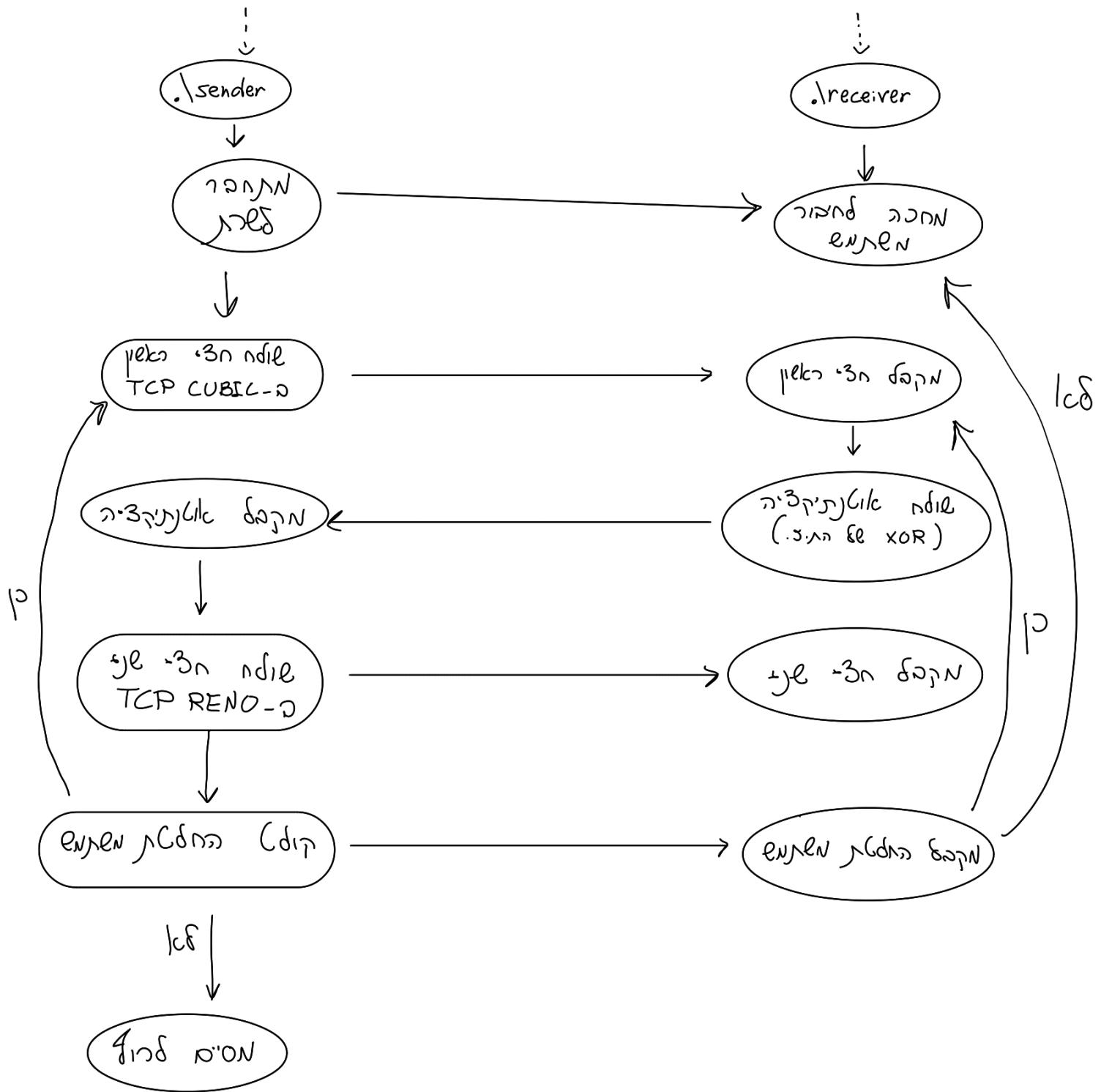
"Enter "stop" to end session"

ואכן בהתאם להניל על מנת לעצור יש לכתוב "stop" , כל קלט אחר יריץ את הקוד שוב ובזמן ישלח את הקובץ מחדש.

- תלויות - 1.3

- gcc
- makefile
- wireshark
- iproute

- תרשימים זרימה - 1.4



הסביר על שינוי האלגוריתם בקוד:
בשורות להלן, יש החלק שמשנה את אלגוריתם של ה sender ל

```
51 // Changing to cubic CC algorithm
52 char ccBuffer[256];
53 strcpy(ccBuffer, "cubic");
54 socklen_t socklen = strlen(ccBuffer);
55 if (setsockopt(SendingSocket, IPPROTO_TCP, TCP_CONGESTION, ccBuffer, socklen) != 0)
56 {
57     perror("ERROR! socket setting failed!");
58     goto end;
59 }
60 socklen = sizeof(ccBuffer);
61 if (getsockopt(SendingSocket, IPPROTO_TCP, TCP_CONGESTION, ccBuffer, &socklen) != 0)
62 {
63     perror("ERROR! socket getting failed!");
64     goto end;
65 }
66 printf("Changed Congestion Control to Cubic\n");
67
```

בשורות להלן יש החלק שמשנה את האלגוריתם של ה sender ל

```
129 // Changing to reno CC algorithm
130 strcpy(ccBuffer, "reno");
131 socklen = strlen(ccBuffer);
132 if (setsockopt(SendingSocket, IPPROTO_TCP, TCP_CONGESTION, ccBuffer, socklen) != 0)
133 {
134     printf("ERROR! socket setting failed!\n");
135     return -1;
136 }
137 socklen = sizeof(ccBuffer);
138 if (getsockopt(SendingSocket, IPPROTO_TCP, TCP_CONGESTION, ccBuffer, &socklen) != 0)
139 {
140     printf("ERROR! socket getting failed!\n");
141     goto end;
142 }
143 printf("Changed Congestion Control to Reno\n");
```

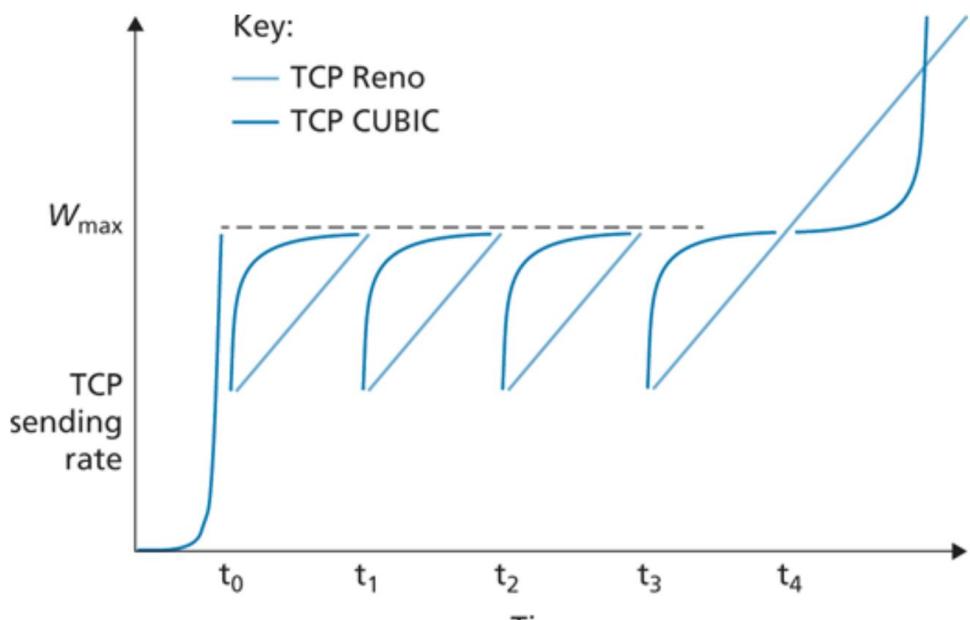
בנוסף ל-sender הוספנו ב-receiver שינוי של האלגוריתם.
בשורות 100 - 115 ב receiver.c יש שינוי אלגוריתם ל TCP CUBIC
בשורות 161 - 175 ב receiver.c יש שינוי אלגוריתם ל TCP RENO
אין השפעה משמעותית על התוכנה בשינוי האלגוריתם ב reciever כפי שנאמר בתרגול, עשוינו זאת ליתר ביטחון מכיוון שהוראות המטלה הציבינו שיש לעשות זאת.

חלק שני

בעובדה זו אנו נרצה להראות את ההבדל בין CC אלגוריתמים שונים ואיך זמן שליחת קובץ מסוים יושפע מהאלגוריתם הנבחר. לשם כך, נריץ את התוכנה שלנו במספר פעמים, בכל פעם עם אחוז שונה של איבוד פקודות בעזרת iproute2 וனראה מה הבדל בין האלגוריתמים.

אנו משתמשים באלגוריתם cubic לשילוח החצאי הראשון של הקובץ ואז משנים לאלגוריתם reno לחצאי השני(אין הבדל בגודל, ובדרך כלל השילוח של שני החצאים קבצים חוץ מסוג האלגוריתם). TCP reno = Slow Start + AIMD + Fast Retransmit + Fast Recovery אלגוריתם reno מורכב מ-AIMD ואנו משתמש באלגוריתם שונה叫做 threshold. האלגוריתם הנ"ל הוא בעצם עוקמה שmagieה באוטו זמן כמו AIMD לthreshold. האלגוריתם cubic דומה לו-reno רק שבמקרה של cubic נשתמש באוטו זמן(threshold) במקום threshold. העוקמה העולה מהר יותר, ושה"כ קיבל שהחלה לשילוח פקודות יהיה יותר גדול וישלחו בסופו שמאלית) אבל העוקמה עולה מהר יותר, ושה"כ קיבל שהחלה לשילוח פקודות יהיה יותר גדול וישלחו בסופו של דבר יותר פקודות בטוח זמן. העוקמה הזה היא ביצוע של binary search, במקומות עלייה לינארית, כדי להגיע אל ה-threshold. לכן, נצפה שהאלגוריתם cubic יהיה מהר יותר, וכאשר איבוד הפקודות יהיה גדול יותר, הבדל בזמן יתיר(כאשר אין איבוד פקודות שלא יהיה הבדל בזמןים כי לא יהיה איבוד מייד או timeout, ועוד החלה פשוט עלה בעזרת slow start).

Figure 3.54 TCP congestion avoidance sending rates: TCP Reno and TCP CUBIC

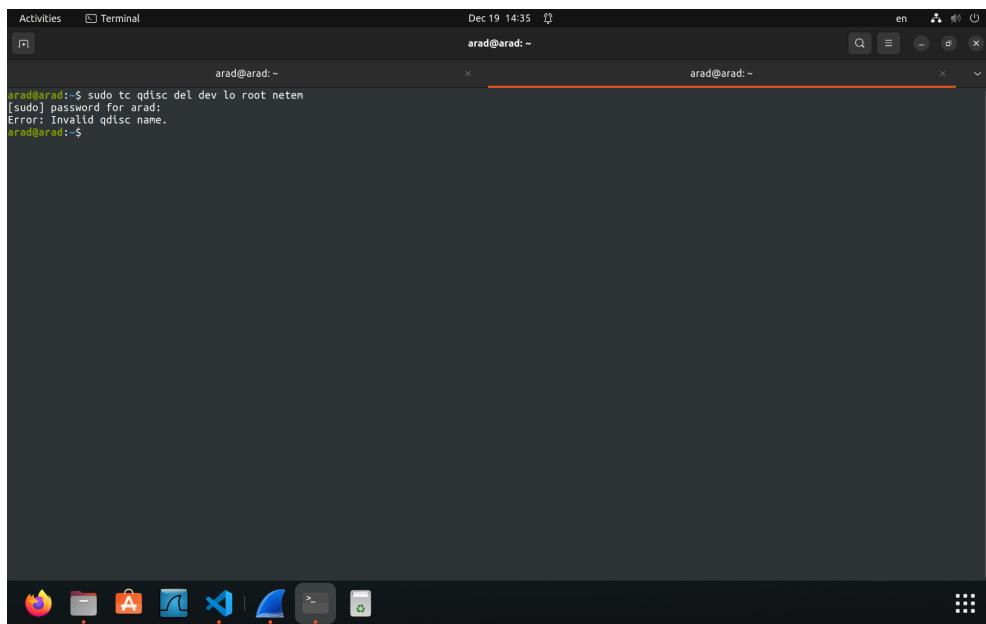


(תמונה מהספרקורס עמוד 272)

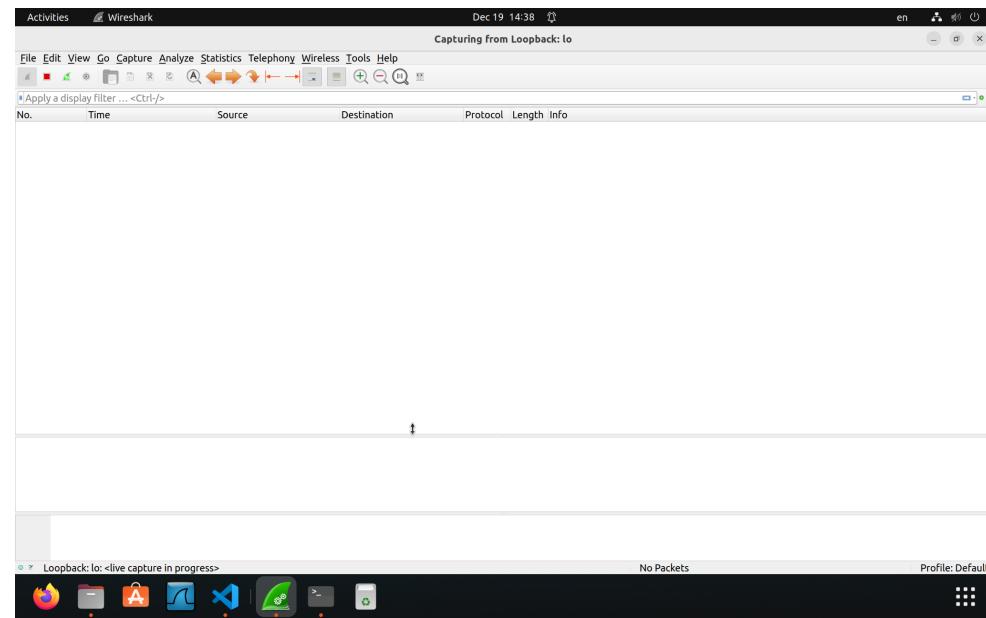
- בעמודים הבאים נראה את ההרצאות לכל אחוז איבוד פקודות והסביר لماذا שאנו רואים בוירשארק.
- נריץ 20 פעמים כל תוכנית כדי לקבל מדגם גדול בכל בדיקה, ונסתכל על הזמן הממוצע לקבלת חבילת מידע בכל ריצה עם איבוד פקודות שנע בין 0 ל-20 אחוז (כאשר אנו מרכיבים את החבילה מספר פעמים אנו יוצרים data set שנוןע לבדוק את התוכנית של ולקבל סטטיסטיקה עם מדגם יותר גדול).
- עיקר הנition מתבצע ב-0 ו-ב-10 אחוז איבוד פקודות, כיוון שב-15 ו-ב-20 קורים אותם הדברים כמו ב-10 אחוז איבוד פקודות, ורק בתדרות יותר גבוהה.

0% איבוד פקודות

נודה תחילה שאין איבוד פקודות בתכנית:



לפנינו כל ריצה נריצה ווירשאך ונסתכל על loopback, שומרה לנו את הפקודות שנשלחו בטור המחשב.



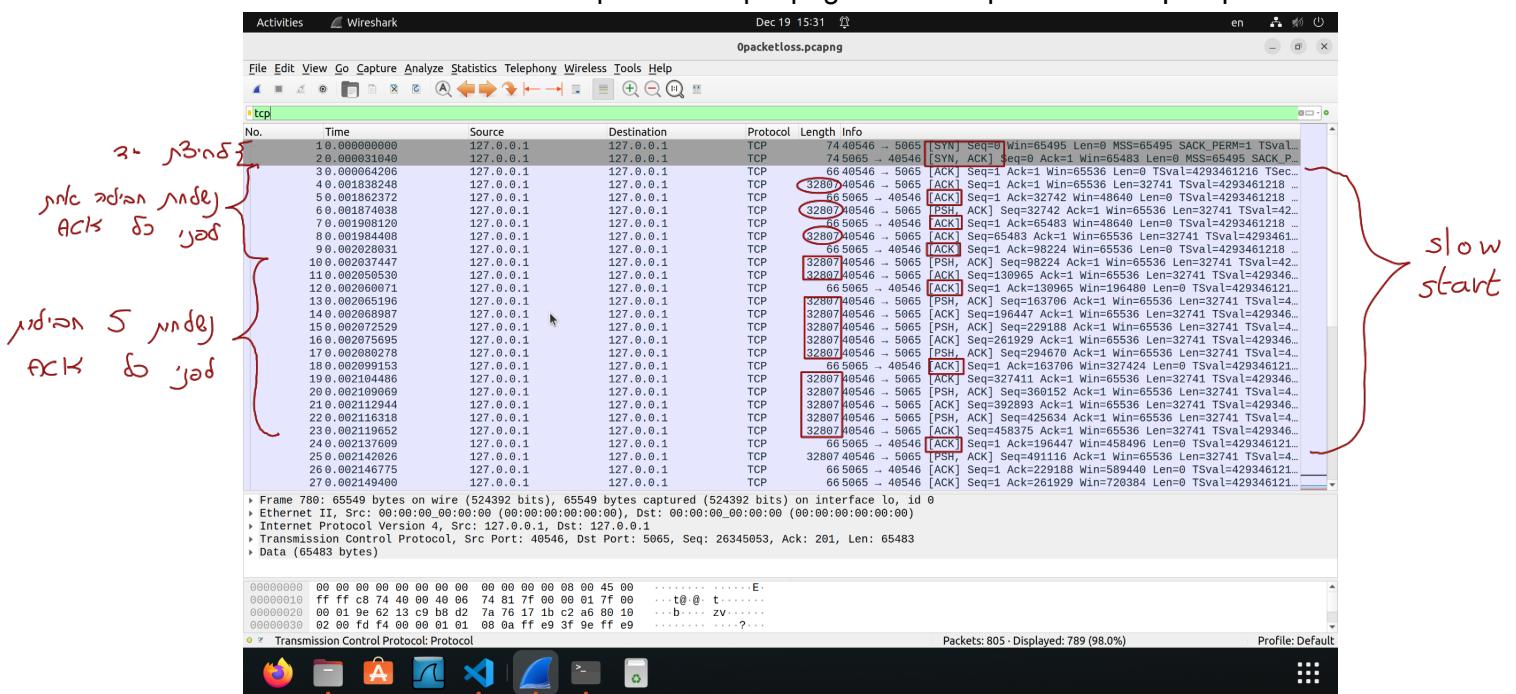
לאחר הריצה הראשונה, קיבלנו:

```

Activities Visual Studio Code Dec 19 14:44
sender.c - network_assignment3 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
receiver
sender
received 8 bytes from server. reply: 1740887
Changed Congestion Control to Reno
read 524287 bytes
sent 524287 bytes
Sent in half 2 524287 bytes
Enter "stop" to end session:
asd
sent msg successfully: asd
*****
read 524287 bytes.
sent 524287 bytes.
received 8 bytes from server. reply: 1740887
Changed Congestion Control to Reno
read 524287 bytes.
sent 524287 bytes.
Sent in half 2 524287 bytes
Enter "stop" to end session:
vr
sent msg successfully: vr
*****
read 524287 bytes.
sent 524287 bytes.
received 8 bytes from server. reply: 1740887
Changed Congestion Control to Reno
read 524287 bytes.
sent 524287 bytes.
Sent in half 2 524287 bytes
Enter "stop" to end session:
stop
sent msg successfully: stop
*****
```

המזהה הראשון הוא 1740887
המזהה השני הוא 1740888
המזהה השלישי הוא 1740889
המזהה הרביעי הוא 1740890
המזהה החמישי הוא 1740891
המזהה השישי הוא 1740892
המזהה השביעי הוא 1740893
המזהה השמיני הוא 1740894
המזהה העשירי הוא 1740895
המזהה ה-11 הוא 1740896
המזהה ה-12 הוא 1740897
המזהה ה-13 הוא 1740898
המזהה ה-14 הוא 1740899
המזהה ה-15 הוא 1740900
המזהה ה-16 הוא 1740901
המזהה ה-17 הוא 1740902
המזהה ה-18 הוא 1740903
המזהה ה-19 הוא 1740904
המזהה ה-20 הוא 1740905
המזהה ה-21 הוא 1740906
המזהה ה-22 הוא 1740907
המזהה ה-23 הוא 1740908
המזהה ה-24 הוא 1740909
המזהה ה-25 הוא 1740910

המוצע לשילוח החצי הראשון הוא 480 מילישניות ולחצי השני הוא 534 ל-25 שילוחות של הקובץ.
ההבדל המוצע הוא 50 מילישניות, זהה מספר זנich מאד, ונניח שההבדל בין גורמים פנימיים במחשב כיון
שה-0 אוחז איבוד פקודות אין הבדל בין אלגוריתם הריצה כי החלון גדול באופן קצב. בנוסף ניתן לראות
בוירשאך שאין בכלל איבוד פקודות בחבילה 0packetloss.pcapng



בתמונה הנ"ל אנו רואים שלאחר הלחיצת ידיהם, נשלחות חבילות בגודל 32807 וישראל מכך הסרבר שלו ack ולאחר מכן נשלחות שני חבילות כללה עד שהסרבר שולח ack, ואז נשלחו 5 חבילות לפני ack - זה האלגוריתם slow start שמוגנה גם ב-reno וגם ב-tahoe.

10% איבוד פקודות

נשנה את איבוד הפקודות מ-10% לנכפה להבדל בין הממוצע של כל חלק קובץ.

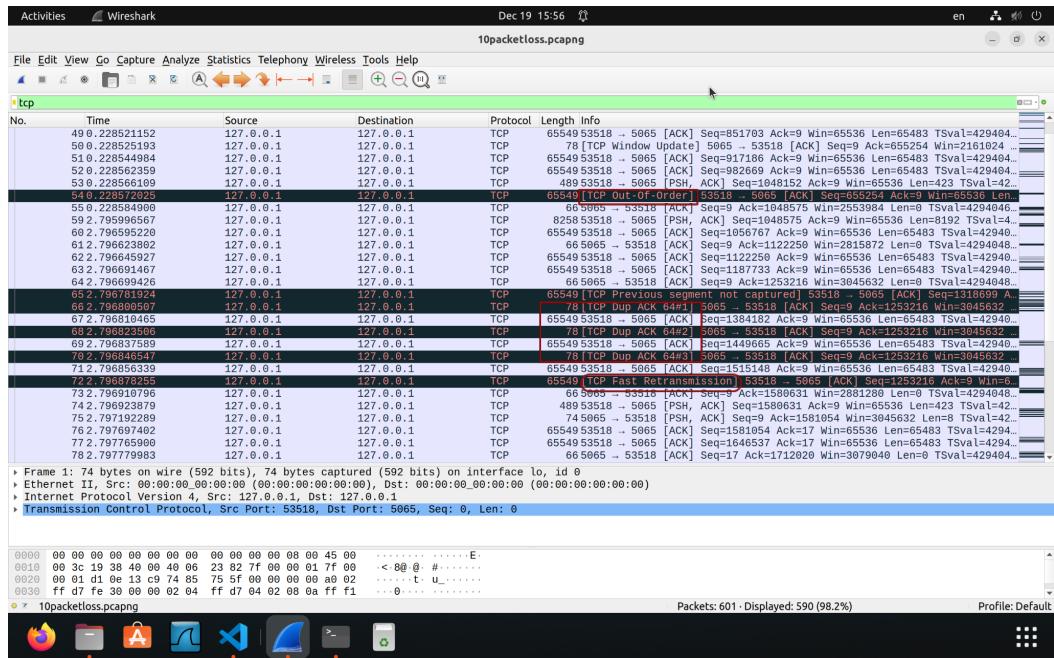
```
arad@arad:~$ sudo tc qdisc del dev lo root netem
[sudo] password for arad:
Error: Invalid qdisc name.
arad@arad:~$ sudo tc qdisc add dev lo root netem loss 15%
[sudo] password for arad:
arad@arad:~$ sudo tc qdisc del dev lo root netem
arad@arad:~$ sudo tc qdisc add dev lo root netem loss 10%
arad@arad:~$
```

תמונה של terminal בו אנו מרים את התוכנית, בצד ימין ה-sender ובצד שמאל ה-receiver. כאשר אני שולח "stop" נסגר החיבור וה-receiver מראה לי סיכום של הזמן.

```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
receiver
receives in total for second half: 524287 bytes
waiting for user decision...
Received 8192 bytes from client. decision is stop
Client has decided to end the session. State:
time taken in micro seconds for first half: 227381
time taken in micro seconds for second half: 941
time taken in micro seconds for first half: 976
time taken in micro seconds for second half: 961
time taken in micro seconds for first half: 952
time taken in micro seconds for second half: 907
time taken in micro seconds for first half: 13060
time taken in micro seconds for second half: 646
time taken in micro seconds for first half: 159
time taken in micro seconds for second half: 255768
time taken in micro seconds for first half: 993
time taken in micro seconds for second half: 754
time taken in micro seconds for first half: 62721
time taken in micro seconds for second half: 371
time taken in micro seconds for first half: 215221
time taken in micro seconds for second half: 394
time taken in micro seconds for first half: 16409
time taken in micro seconds for second half: 833
time taken in micro seconds for first half: 922
time taken in micro seconds for second half: 424849
time taken in micro seconds for first half: 230855
time taken in micro seconds for second half: 61574
time taken in micro seconds for first half: 140
time taken in micro seconds for second half: 642660
time taken in micro seconds for first half: 183
time taken in micro seconds for second half: 270943
time taken in micro seconds for first half: 170
time taken in micro seconds for second half: 813
time taken in micro seconds for first half: 785
time taken in micro seconds for second half: 3275
time taken in micro seconds for first half: 10882
time taken in micro seconds for second half: 43885
time taken in micro seconds for first half: 615
time taken in micro seconds for second half: 449
time taken in micro seconds for first half: 1088
time taken in micro seconds for second half: 207854
Total of time for first half in file[18] is: 982723 for 18 times for an average of 54595.
Total of time for second half in file[18] is: 1915877 for 18 times for an average of 106437.
Waiting for incoming TCP-connections...
arad@arad:~/Desktop/network_assignment3$
```

הממוצע לשילוח החצי הראשון הוא 54596 מלישניות ולחצי השני הוא 106437 לשוה' 18 שליחות של הקובץ.

במבחן מהיר על הזמן של חצאי הקבצים ניתן לראות כמה זמנים חריגים גדולים מאוד לשילוח חצי קובץ, שנൻ לוירשאך נראה את אותם מקרים כך:



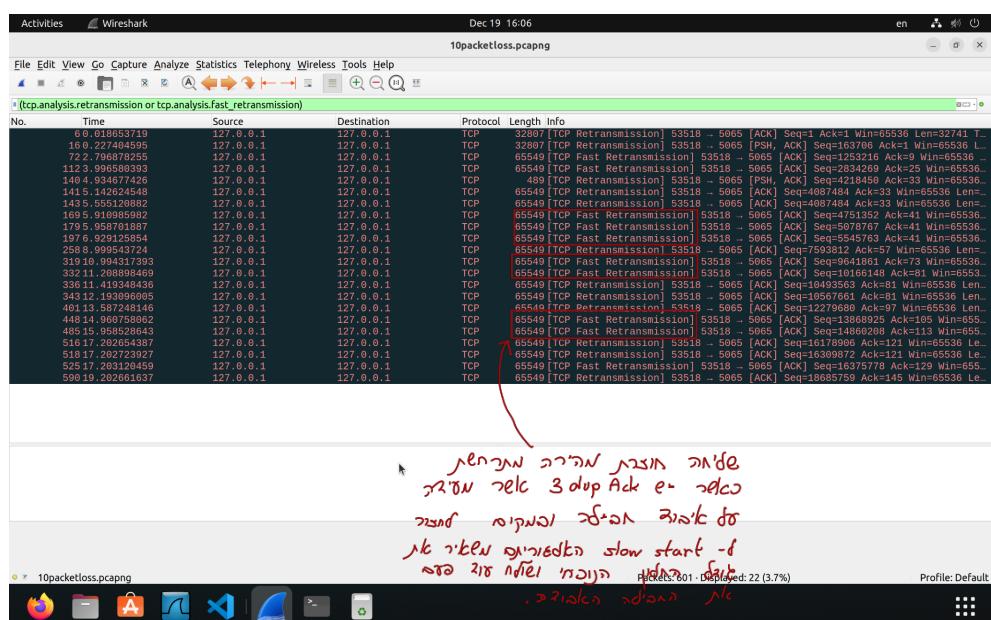
פקודות מסווגות בצד ימין מכמה סוגים:

- פקטה מס' 72 למשל היא שליחה חוזרת של פקטה שלא התקבל עליה ACK במסגרת ה TIMEOUT של TCP, ניתן לראות שהיא מסומנת בשחור וורוד.

x3 acks duo - מתריעה על שליחת 3 אקים חזרה מהשרת ללקוח, זה קורה משני סיבות אפשריות, הראשונה מכיוון שהשרת קיבל חבילות שלא לפ' הסדר, השנייה היא איבוד חビלה, בהודעה זו השרת יידע את הלוקה על מספר הסגמנט המוצופה אשר הוא מחכה לו.

out-of-order - כאשר יותר מדי חבילות מקבלות שלא לפ' הסדר, TCP יגרום לשידור חוזר בדומה למקרה שבו יש איבוד חבילות, על מנת למנוע תעבותה מידע ללא תועלת שכן החבילות שמקבלות לא לפ' הסדר שליהם, TCP מתעלם מהם.

בתמונה הבאה מסונן retansmitions מההריצה הנ"ל. אנו רואים שיש הבדל לעומת הריצה ביל' איבוד פקודות, שיש לנו.packet loss בין השאר בגל הטעות שנסלחות בגל האיבוד פקודות.



15% איבוד פקודות

נשנה את איבוד הפקודות מ-15% ל-15% ונראה לבדל בין הממוצע לבין כל חלק קובץ.

```
arad@arad:~$ sudo tc qdisc del dev lo root netem
[sudo] password for arad:
Error: Invalid qdisc name.
arad@arad:~$ sudo tc qdisc add dev lo root netem loss 15%
[sudo] password for arad:
arad@arad:~$ sudo tc qdisc del dev lo root netem
arad@arad:~$ sudo tc qdisc add dev lo root netem loss 10% ←
arad@arad:~$ sudo tc qdisc change dev lo root netem loss 15%
arad@arad:~$
```

תמונה של הTerminal בו אנו מryanם את התוכנית, בצד ימין הsender ובצד שמאל receiver בו אנו רואים את הסיכון של הזמן הקודמי: (15packetloss.pacpng)

```
474818
483810
491802
500824
507586
515778
523864
52407
524287 bytes in total for second half: 524287 bytes
waiting for user decision...
Received 8192 bytes from client. decision is stop
Client has decided to end the session. Stats:
time taken in micro seconds for second half: 3808149
time taken in micro seconds for second half: 3808149
time taken in micro seconds for first half: 1533374
time taken in micro seconds for second half: 6999098
time taken in micro seconds for first half: 736772
time taken in micro seconds for second half: 6999031
time taken in micro seconds for first half: 557757
time taken in micro seconds for second half: 2261444
time taken in micro seconds for first half: 110
time taken in micro seconds for second half: 1257075
time taken in micro seconds for first half: 99406
time taken in micro seconds for second half: 1003731
time taken in micro seconds for first half: 251
time taken in micro seconds for second half: 4088652
time taken in micro seconds for first half: 714920
time taken in micro seconds for second half: 573730
time taken in micro seconds for first half: 815
time taken in micro seconds for second half: 104851
time taken in micro seconds for first half: 1110
time taken in micro seconds for second half: 1136
time taken in micro seconds for first half: 209426
time taken in micro seconds for second half: 365
time taken in micro seconds for first half: 2000
time taken in micro seconds for second half: 764
time taken in micro seconds for first half: 64470
time taken in micro seconds for second half: 213142
time taken in micro seconds for first half: 931
time taken in micro seconds for second half: 103096
Total of time for first half[14] is: 3961675 for 14 times for an average of 282976.
Total of time for second half in file[14] is: 12894964 for 14 times for an average of 863926.
Waiting for incoming TCP connections... ←
```

הממוצע לשליית החצי הראשון הוא 282976 מילישניות ולחצי השני הוא 863926 לשליות של קובץ.

20% איבוד פקודות

נשנה את איבוד הפקודות מ-20% ל-0% ונראה הבדל בין הממוצע של שילוחה של כל חלק קובץ.

The screenshot shows two terminal windows side-by-side. The left terminal window displays a series of commands related to traffic control (qdisc) on a network interface (lo). The right terminal window shows the same commands but with a different output, indicating reduced packet loss. Red handwritten annotations include '20%' with an arrow pointing to the left terminal and '0%' with an arrow pointing to the right terminal. The title bar of the left terminal says 'Activities Terminal' and the right one says 'arad@arad: ~'. The bottom taskbar includes icons for a browser, file manager, terminal, code editor, and system status.

תמונה של הTerminal שבו מרייצים את התוכנית, בצד ימין ה-sender ובצד שמאל ה-receiver בו אנו רואים את הסיכון של הזמן הקודמים (20packetloss.pcapng הוא יירשאך) (שם הקובץ וירשאך הוא)

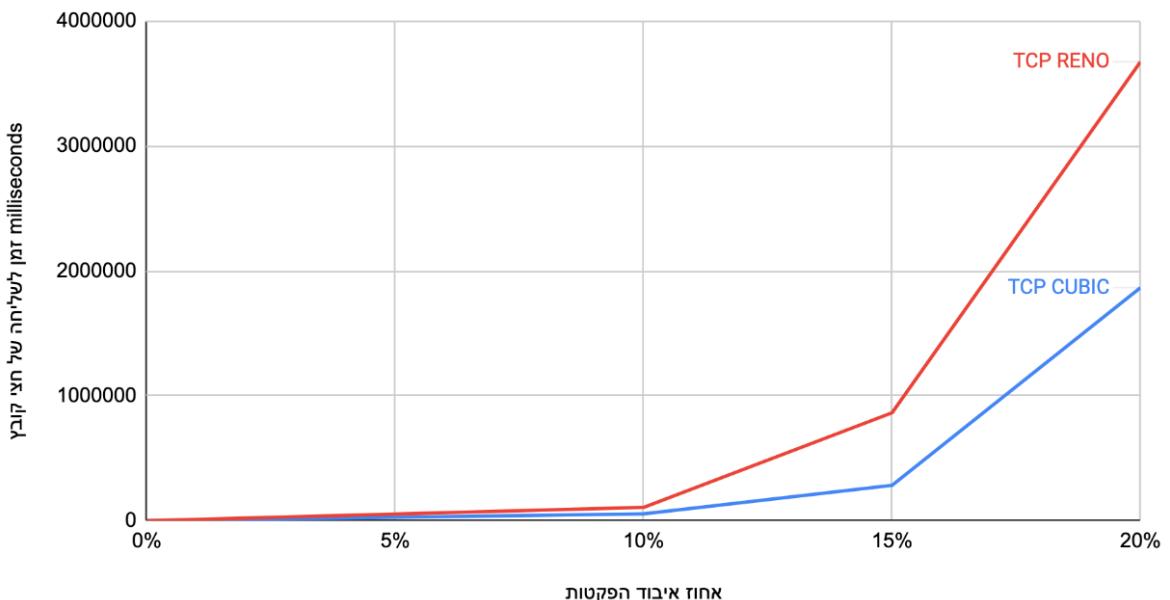
The screenshot shows a terminal window within Visual Studio Code. The code being run is 'sender.c' from a project named 'network_assignment3'. The terminal output shows a network session between a client and a server, with various statistics and congestion control information. Red handwritten annotations include 'אנו' with an arrow pointing to the first few lines of the output, 'את' with an arrow pointing to the middle of the session log, and 'המוצע' with an arrow pointing to the final statistics at the bottom. The title bar says 'Activities Visual Studio Code' and the file path is 'sender.c - network_assignment3 - Visual Studio Code'. The bottom taskbar includes icons for a browser, file manager, code editor, terminal, and system status.

הממוצע לשילוחת החצי הראשון הוא 282976 מילישניות ולחצי השני הוא 863926 לש"כ ~~7~~ 7 שליחות של הקובץ.

לסיום, לאחר כל ההרצאות קיבלנו את הטבלה הבאה:

סוג אלגוריתם/אחוז איבוד פקודות	0%	10%	15%	20%
TCP CUBIC	480	54595	282976	1865753
TCP RENO	534	106437	863926	3673558

נסתכל על גраф שקיבliśmy (ציר ה-x: אחוז איבוד הפקודות, ציר ה-y: כמה זמן לוקח לשולח חצי קובץ בmmo) (צ'יר ה-א: אחוז איבוד הפקודות, צ'יר ה-ב: כמה זמן לוקח לשולח חצי קובץ בmmo)



הgraf תואם למה שציפינו לקבל. ככל שאיבוד הפקודות גדול יותר, הזמן לשילוח של כל חצי קובץ בעזרת האלגוריתם cubic יהיה מהר יותר קבצים לעומת האלגוריתם reno. בזמן ש-reno משתמש ב-polling ומגדיל לינארית את החילון, cubic מגדיל את החילון מהר יותר בהתחלה(העיקונה עלייה הסברנו בהתחלה) וסה"כ הבעיות שנשחלות הוא יותר מהר, אז הזמן לס"ים את השילוח הקובץ הוא מהיר יותר כמו שניתן להסיק מהgraf כי ההבדל בזמןים גדול ככל שהוא איבוד פקודות עולה.

ביבליוגרפיה:

- <https://www.geeksforgeeks.org/tcp-tahoe-and-tcp-reno/>
- <https://witestlab.poly.edu/blog/tcp-congestion-control-basics/#:~:text=The%20main%20difference%20between%20TCP,in%20the%20TCP%20Cubic%20paper.>
- <https://www.researchgate.net/>