

CONGESTION CONTROL TCP

יoud תמר
213451818

ליאור וינמן
213081763

25 בדצמבר 2022

תוכן עניינים

פרק א' - תיאור המערכת	1
הקדמה	1.1
קבץ הפרויקט והרצתו	1.2
ממשק משתמש	1.3
ממשק Sender	1.3.1
ממשק Receiver	1.3.2
הקובץ היוצר Makefile	1.4
קובץ הצירות myHeader.h	1.5
קובץ הטקסט msg.txt	1.6
קובץ השולח Sender.c	1.7
קובץ המתקבל Receiver.c	1.8
פרק ב' - המחבר על אלגוריתמי בקרת העומס	2
הקדמה	2.1
ניסוי 1 - 0% איבוד חבילות	2.2
שליחת התבילהות	2.2.1
מספר חבילות שנשלחו	2.2.2
תוצאות	2.2.3
מסקנות	2.2.4
ניסוי 2 - 10% איבוד חבילות	2.3
שליחת התבילהות	2.3.1
מספר חבילות שנשלחו	2.3.2
תוצאות	2.3.3
מסקנות	2.3.4
ניסוי 3 - 15% איבוד חבילות	2.4
שליחת התבילהות	2.4.1
מספר חבילות שנשלחו	2.4.2
תוצאות	2.4.3
מסקנות	2.4.4
ניסוי 4 - 20% איבוד חבילות	2.5
שליחת התבילהות	2.5.1
מספר חבילות שנשלחו	2.5.2
תוצאות	2.5.3
מסקנות	2.5.4
מסקנות מקייפות בביבליוגרפיה	2.6

1 פרק א' - תיאור המערכת

1.1 הקדמה

בפרק זה עוסוק בתיאור כלל המערכת שבנינו. נתאר אילו קבצים משתתפים בפרויקט, איך כל אחד מהם ממומש ומה הרציאנו והמשמעות מהורי כל אחד מהם. בנוסף, נסביר כיצד להריץ את התוכנית.

1.2 קבצי הפרויקט והרכתו

את הפרויקט יש להריץ על מערכת הפעלה UBUNTU LTS גרסה 22.04 בלבד! הפרויקט מכיל את הקבצים להן:

1. *Makefile* - הקובץ היוצר של התוכנית.

2. *myHeader.h* - קובץ הצהרות.

3. *Sender.c* - שלוח ההודעות.

4. *Receiver.c* - מקבל ההודעות.

5. *msg.txt* - ההודעה שאותה השולח ישלח למקבל.

על כל הקבצים נהיב במשך(!) את כלל הקבצים האל, יש להוריד ולשמור באותו התקיה. ישנה אפשרות להוריד ישרות ממה-GITHUB, בקישור הנ"ל:

https://github.com/liorvi35/TCP_CongestionControl_C

לאחר ההורדה והשמירה של כלל קבצי הפרויקט, נפתח חלון טרמינל במקומות של התקיה שבה נמצאים כל הקבצים ונכתב אט הפקודה: "make all". לאחר שכתבנו את הפקודה הנ"ל יוצרו בתיקיה הקבצים הבאים:

1. קובץ הצהרות מהודר *myHeader.h.gch*.

2. קובץ אובייקט ביןארי של *Sender.c*, ז"א זה הקובץ המתתקבל לאחר היזור שלו.

3. קובץ אובייקט ביןארי של *Receiver.c*.

4. קובץ הרצת התוכנית של *Sender.c*, ז"א ההרצה מתבצעת על ידי הקובץ הנ"ל.

5. קובץ הרצת התוכנית של *Receiver.c*.

כעת לאחר סיוםו את תהליך היצירה, נפתח שני חלונות טרמינל על מיקום התקיה עם כלל הקבצים (ניתן לשימוש בטרמינל שבו יצרנו את הקבצים, אך רצוי קודם לנקוט אותו עם הפקודה "clear"). בטרמינל הראשון כתוב את הפקודה "./Sender" ובטרמינל השני נכתב את הפקודה "./Receiver" (סדר הפקודות קרייטי, כיון שעליינו קודם כל להפעיל את השרת - מקבל ההודעות לפני שנitin להפעיל את הלקוח - שלוח ההודעות).

לאחר שהרכנו את שתי הפקודות האלו, בחלון אחד רצתה התוכנית *Receiver.c* ובחלון השני רצתה התוכנית *Sender.c* (בשתייה נפתח משק המשתמש ונitin להכניס קלט ולקבל פלט בהתאם). נחזר בקרה על הדברים שנאמרו ונסכם:

- קבצי הפרויקט הם: *Makefile*, *myHeader.h*, *Sender.c*, *Receiver.c*, *msg.txt* יש להוריד ולשמור אותם באותו התקיה.

- נפתח שני טרמינלים על התקיה עם הקבצים, רץ הפקודות הבא מרים את הפרויקט:

1. בטרמינל הראשון נרשם "make all" ולאחר מכן "clear", הדבר יגרום יצירת הפרויקט ונקיי הפלט הסטנדרטי.

2. בטרמינל הראשון נרשם "./Receiver", הדבר יגרום להרצה התוכנית של מקבל הקבצים.

3. בטרמינל השני (על שניהם להיות פתוחים במקביל!) נרשם "./Sender", הדבר יגרום להרצה התוכנית של שלוח הקבצים.

4. כעת נפתח בפנינו משק המשתמש, ניתן להזין קלט ולקבל פלט רצוי.

1.3 ממשקי המשתמש

לאחר שביצעו את כל השלבים מהנושא הקודם,icut נרצה להשתמש בתוכנית (**עד כה ורק הרצנו...**). בשתי התוכניות נפתח בפנינו ממשק נוח למשתמש אשר מדפיס למשתמש קלט ברור על הפעולות (רק אלו הרלוונטיות לידעו המשתמש) שקרו וכמוון גם מבקש מאיתנו קלט על מנת להמשיך את הרצת התוכנית. בנושא זה, נסקור את המשקים של שתי התוכניות הרצות (שהן, תזכורת: *Sender* ו- *Receiver*) ונראה כיצד ניתן להגיע ולקבל את הפלט הרצוי.

1.3.1 ממשק ה-*Sender*

מיד בעית הפעלת התוכנית, הקובץ כבר מבצע מספר רב של פעולות (הפעם הראשונה מבוצעת אוטומטית והחל מופעם השניה התוכנית תשאל אותנו האם ברצוננו לחזור על פעולות אלו). פלט המשיק להלן:

מעבר על הפלט שאנו רואים בתמונה:

```
liorvi35@liorvi-VirtualBox:~/Desktop/TCP_CongestionControl_C$ ./Sender
file opened!
socket created!
connected to receiver!
first half has been sent!
waiting for authentication...
authentication match!
CC algorithm has been changed to: "reno"
second half has been sent!
send file again? (y/n)
■
```

1. ראשית, אנו רואים כי הקובץ שאנו שולחים למקבל נפתח בהצלחה.
2. אנו רואים כי נוצר שקע והתקשרות (*socket*) בהצלחה.
3. השולח התחבר בהצלחה למקבל.
4. החלק הראשון של הקובץ נשלח.
5. ממתחין לקבלת הסימון המוסכם לאישור (*authentication*) מהשרת.
6. קיבל את האישור והוא אכן נכון.
7. שונה האלגוריתם לבקרת העומס (*CC algorithm*) מ-*cubic* ל-*reno*.
8. החלק השני של הקובץ נשלח.
9. שאלת למשתמש - האם לחזור על התהליך הנ"ל ולשלוח את הקובץ מחדש?
10. המתנה של המשתמש להזנה של התו המתאים. יש להזין '*y*' אם ברצוננו לשЛОח מחדש ויש להזין '*n*' אם ברצוננו ליצא.

נציין כי שורות 3-1 הן שורות אשר יהיו מודפסות רק בעית הפעלה הראשונה (שהרי אם כבר הקובץ נפתח, השקע נוצר והשולח מחובר למקבל - אין צורך לחזור על פעולות אלו).icut נעבור על שתי האפשרויות המתאפשרות לאחר מכן נראה גם פלטים בהתאם למקבל.

אם המשתמש בקיש ליצאת מהתוכנית (ז"א היזן את התו '*n*') אז נקבל את הפלט הבא:

```
liorvi35@liorvi-VirtualBox:~/Desktop/TCP_CongestionControl_CS$ ./Sender
file opened!
socket created!
connected to receiver!
first half has been sent!
waiting for authentication...
authentication match!
CC algorithm has been changed to: "reno"
second half has been sent!
send file again? (y/n)
n
exit message has been sent
liorvi35@liorvi-VirtualBox:~/Desktop/TCP_CongestionControl_CS$ █
```

נשים לב כי קיבלנו פלט שאומר שנשלחה הודעה יציאה ולאחר מכן התוכנית הסיימה, בambilים אחרות, אם המשתמש מזין '*n*' אז הלקוח שלח הודעה יציאה לשרת, סגור את התקשרות ומסיים לזר.

כעת נסתכל על המקרה השני, אם המשתמש בקיש לשЛОוח את הקובץ מחדש (ז"א היזן את התו '*y*') נקבל את הפלט הבא:

```
liorvi35@liorvi-VirtualBox:~/Desktop/TCP_CongestionControl_CS$ ./Sender
file opened!
socket created!
connected to receiver!
first half has been sent!
waiting for authentication...
authentication match!
CC algorithm has been changed to: "reno"
second half has been sent!
send file again? (y/n)
y
CC algorithm has been changed to: "cubic"
first half has been sent!
waiting for authentication...
authentication match!
CC algorithm has been changed to: "reno"
second half has been sent!
send file again? (y/n)
█
```

מעבר על השורות שקיבלו לאחר שהמשתמש היזן '*y*:

1. שונה אלגוריתם בקרת העומס מ-'reno' ל-'cubic'
2. החלק הראשון של הקובץ נשלח.
3. ממותין לקבלת הסימן המוסכם מוהשרת.
4. קיבל את האישור והוא אכן נכון.
5. שונה האלגוריתם לברכת העומס מ-'cubic' ל-'reno'
6. החלק השני של הקובץ נשלח.
7. כעת, ישנה חזרה על התהליך - ישנה שוב שאלת משתמש האם לשЛОוח מחדש את הקובץ או האם לצאת מהתוכנית.

כך התוכנית ממשיכת לעבוד בכל בחירה של המשתמש, ככלומר אם המשתמש מזין כעת '*y*' הלקוח שוב משנה את האלגוריתם, שלוח את החלק הראשון, מוחכה ומתקבל אישור, משנה את האלגוריתם, שלוח את החלק השני ולאחר מכן שואל את המשתמש האם לחזור על התהליך ואם המשתמש מזין כעת '*n*' אז תישלח הודעה יציאה לשרת, התקשרות תיסגר והתוכנית תיעצר.
בזאת סיימו לפרט על הממשק של השולח.

1.3.2 ממשק ה-Receiver

כעת, בנושא זה, נverb על הממשק של המקלט ונראה בהתאם כיצד הפעולות שהשלח מבצע משפיעות על המקלט בצורה ישירה.
הפלט המתקבל מהרצת השרת:

```
liorvi35@liorvi-VirtualBox:~/Desktop/TCP_CongestionControl_C$ ./Receiver
socket created!
socket bound!
waiting for connection...
```

נשים לב כי הודפסו שלוש שורות, נverb עליהן:

1. שקע התקשרות נוצר בהצלחה.
 2. שקע התקשרות נקשר לכתובת IP (אנו משתמשים בכתובת המארח המקומי שהיא "127.0.0.1") ונקשר לכתובת PORT (אנו משתמשים בשער "8395").
 3. השרת ממתין לחיבור מוחמשתמש.
- שוב, נציין כי שלוש השורות האלו יודפסו רק פעם אחת ויחידה והיא בתחילת הריצת המקלט. כעת, נפעיל את השולח ונראה כיצד הדבר משפיע על המקלט:

```
liorvi35@liorvi-VirtualBox:~/Desktop/TCP_CongestionControl_C$ ./Receiver
socket created!
socket bound!
waiting for connection...
sender connected!
first message has been received!
authentication has been sent!
CC algorithm has been changed to: "reno"
second message has been received!
CC algorithm has been changed to: "cubic"
```

נverb על השורות חדשות שהודפסו:

1. השולח התחבר בהצלחה.
2. החלק הראשון של הקובץ התקבל.
3. נשלח אישור על קבלת הקובץ.
4. שונה האלגוריתם לבקרת העומס מ-reno ל-cubic.
5. החלק השני של הקובץ התקבל.
6. שונה אלגוריתם בקרת העומס מ-cubic ל-reno.
7. השרת ממתין לאישור להמשך התהיליך (או סיומו) מוחמשתמש.

כעת, ברצונו להסתכל מה קורה לשרת כאשר המשתמש מבקש ליצאת (ז"א מין 'n'):

```
liorvi35@liorvi-VirtualBox:~/Desktop/TCP_CongestionControl_C$ ./Receiver
socket created!
socket bound!
waiting for connection...
sender connected!
first message has been received!
authentication has been sent!
CC algorithm has been changed to: "reno"
the receiver only received 550005 of 550005 bytes of the second partCC algorithm has been changed to: "cubic"
exit message has been received!

#####
collected DATASET is:

time to receive part 1 of file 1 is 0.389000 [ms]
time to receive part 2 of file 1 is 0.263000 [ms]

#####
the times of avg part:
the average of sending first file - by "cubic" - is: 0.389 [ms]
the average of sending second file - by "reno" - is: 0.263 [ms]

#####
liorvi35@liorvi-VirtualBox:~/Desktop/TCP_CongestionControl_C$
```

מעבר על השורות החדשות:

1. התקבלה הודעה יציאה מהלכות.

2. הודפסו הזמן שלקח לקבל כל חלק מהלכות.

3. הודפסו הזמן הממוצעים לקבל כל אחד מהחלקים.

כלומר, אם המשתמש מבקש ליצאת גם השרת מסיים לעבוד ופולט את המידע שאסף מאז תחילת תהליך ההרצה (להלן *Dataset*).

כעת נראה מה קורה כאשר המשתמש מבקש לשלוח שוב את הקובץ:

```
liorvi35@liorvi-VirtualBox:~/Desktop/TCP_CongestionControl_C$ ./Receiver
socket created!
socket bound!
waiting for connection...
sender connected!
first message has been received!
authentication has been sent!
CC algorithm has been changed to: "reno"
second message has been received!
CC algorithm has been changed to: "cubic"
first message has been received!
authentication has been sent!
CC algorithm has been changed to: "reno"
second message has been received!
CC algorithm has been changed to: "cubic"
```

נשים לב כי קורה בדיק אותו תהליך כמו בתחילת ההרצה. כעת נניח כי המשתמש ביקש ליצאת, נראה מה קורה מבנית הדפסות הזמן:

```

liorvi35@liorvi-VirtualBox:~/Desktop/TCP_CongestionControl_C$ ./Receiver
socket created!
socket bound!
waiting for connection...
sender connected!
first message has been received!
authentication has been sent!
CC algorithm has been changed to: "reno"
the receiver only received 550005 of 550005 bytes of the second partCC algorithm has been changed to: "cubic"
first message has been received!
authentication has been sent!
CC algorithm has been changed to: "reno"
the receiver only received 550005 of 550005 bytes of the second partCC algorithm has been changed to: "cubic"
exit message has been received!

#####
collected DATASET is:

time to receive part 1 of file 1 is 0.162000 [ms]
time to receive part 2 of file 1 is 0.139000 [ms]
time to receive part 1 of file 2 is 0.270000 [ms]
time to receive part 2 of file 2 is 0.316000 [ms]

#####
the times of avg part:
the average of sending first file - by "cubic" - is: 0.216 [ms]
the average of sending second file - by "reno" - is: 0.228 [ms]

#####
liorvi35@liorvi-VirtualBox:~/Desktop/TCP_CongestionControl_C$
```

נשים לב כי קורה כאן אותו תהליך כמו בפעם הראשונה כאשר המשתמש בקש לצאת אבל הפעם, מודפסים יותר זמנים (שורי הפעם, הקובי נשלח פעם נוספה).

אם כאן התוכנית ממשיכה לעבוד באותה דרך כמו שראינו כתה, כלומר, אם המשתמש מזמן t איז השרת מקבל הודעה יציאה, מדפיס זמנים לקבלת כל חלק ואת הזמן המומוצעים ומסיים לעבוד. ואם המשתמש מזמן y (נניח שהמשתמש הzin y מס' $t \in \mathbb{N}$ כלשהו של פעמים) איז יודפסו $2 \cdot t$ שורות של זמנים לקבלת החלקים של הקבצים שנשלחו ועוד שתי שורות להדפסת הזמן המומוצעים.

בזאת סיימנו לפרט על המשך של המקלט.

1.4 הקובץ היוצר - Makefile

נזכיר חזרה לבסיס, הקוד אשר כתבנו נכתב בשפה (העילית) *C*. המחשב אינו יודע ישירות להריץ את קוד זה ולכן ולכון לבצע מספר תהליכיים לפני שנוכל בכלל להריץ את התוכנית שכתבנו, התהליכים הללו הם הידור ו קישור. נסביר, המחשב (המעבד) כלל אינו יודע מה זו השפה שאנו מותבים בה ומה המשמעות שלה, הוא מבין רק את השפה הבינארית, לכן כדי שנוכל להריץ עליו לתת למחשב שהוא יודע לעובד אותו. ישנו שני תהליכים עיקריים אשר קובץ *makefile* מטפל בהם וهم:

1. הידור (קומpileציה) - תהליך זה ממיר את קבצי שפת התוכנות שלנו (קובצים עם הסיומת *".c"* ו *".h."*) לקובצי אובייקטיבים בשפה הבינארית (קובצים עם סיומת *".o"* ו *".h.gch"*), בעת יש בידינו קבצים אשר המחשב יודע לתפעל ונitin להמשיך לעובד עימם.

2. קישור (לינקוג') - לאחר שהשכנו קבצים בשפה הבינארית, נוכל שליטר ממה קבצי הריצה לתוכניות שלנו (שכן, בעת המחשב כן מבין אם נגיד לו להריץ קבצים בינאריים) תהליך זה ממיר את הקבצים הבינאריים שהשכנו בתהליך הקודם לקבי הריצה (הערה: במערכות ההפעלה UBUNTU LTS לא כל קובץ חייב להיות להיות סיימת, אבל לדוגמה במערכות הפעלה WINDOWS קובץ הריצה חייב להיות בעל סיומת *".exe"*), אותן נוכל להריץ ולהפעיל ולממש אתם יחסית קלט-פלט.

על מנת לייצר בקלות את קבצי ההרצה ולהריץ את התוכנית, יצרנו קובץ הנקרא *Makefile* ועל ידי פקודת בת שורה אחת הקובץ מರיץ (אם ישנו בכך צורך) את שני התהליכים שציגו מלעיל ומיציר לנו בקלות קבצי הריצה: מעבור על הקוד שלו ונסביר כיצד הוא עבד:

```
1 # macros - for more dynamic Makefile
2 CC = gcc # compiler
3 FLAGS = -Wall -g # compilation flags
4
5 # for not creating 'all' and 'clean' files
6 .PHONY: all clean
7
8 # final targets
9 all: Receiver Sender
10
11 # (linkage) making executables from objects:
12 Receiver: Receiver.o
13         $(CC) $(FLAGS) -o Receiver Receiver.o
14
15 Sender: Sender.o
16         $(CC) $(FLAGS) -o Sender Sender.o
17
18 # (compilation) making object files from '.c' and '.h' files:
19 Receiver.o: Receiver.c myHeader.h
20         $(CC) $(FLAGS) -c Receiver.c myHeader.h
21
22 Sender.o: Sender.c myHeader.h
23         $(CC) $(FLAGS) -c Sender.c myHeader.h
24
25 # delete all files that created after 'make' command (not deleting '.c' files)
26 clean:
27         rm -f *.o *.h.gch Receiver Sender
```

1. שורות 3 – 2: בשורות אלו אנו יוצרים קיצורי כתיבה ("מקרו"). זאת על מנת להפוך את הכתיבה שלנו ליותר קרייה ודינאמית. אנו יוצרים מאקרו אחד עבור המהדר (קומפיילר) שישמש אותנו בתהליך ההמרה (אנו משתמשים ב- *Gnu Compiler Collection*, *gcc*, במלחינים אחרים זהו המהדר הסטנדרטי - *Wall*, – *g*) ובנוסף, אנו יוצרים מאקרו שעבוד דגלי הקומפלציה, הדגלים שעימם אנו נקملם הם " – *Wall*, – *g*" משמעות הדגל הראשון היא להציג לנו רק שגיאות קומפלציה אלא גם אחריות קומפלציה שמהוות סיכון לתוכנית, הדגל השני עוזר לנו למןפה (להלן " – *debugger*") למצוא אזהרות ושגיאות.

שילוב שני הדגלים האלו מהדר את הקוד שלנו בצורה יותר קפנית ובכך אנו מקבלים תוכנית תקינה ונקייה יותר.

2. שורה 6: בשורה זו אנו מצהירים כי התווiotת (*Makefile*) היא שפת תווiotת עם מושימות לביצוע) "clean" ו- "all" – הן אין קבצים ואין לייצר קבצים הנקראים כך. הסיבה לכך כי אין לנו צורך ושימוש בהם, אילו רק תווiotת לצורך נוחות השימוש (למשתמש – כי הוא מזין בכל פעם טה"כ שורה אחת כדי לבצע הרבה) ועל כן אין לנו באמות צורך, כמו כן אין.

3. שורה 9: בשורה זו אנו יוצרים תווiotת אשר תשתמש אותן ליצור שני קבצים של השולח ושל המקלט, בו זמנית. תווiotת זו היא למטרת נוחות, כדי שנוכל בפקודה אחת לייצר שני קבצים ולא נדרש לייצר כל אחד ידנית בנפרד.

4. שורות 14 – 12 ו- שורות 16 – 15: בשורות אלו מטבחע תהליך הקישור, התווiotת משתמש בקובץ אובייקט ביןאי של השולח או המקלט ויוצרת לו קובץ הרצה בהתאם.

5. שורות 20 – 19 ו- שורות 23 – 22: בשורות אלו מטבחע תהליך ההידור, התווiotת משתמש בקובץ קוד הכתוב בשפת C ובקובץ הצהרות, משלבת את שניהם ומקבלת קובץ ביןאי.

6. שורות 27 – 26: בשורות אלו אנו יוצרים תווiotת אשר בעת הרצתה תשמש אותנו למחיקת כל הקבצים אשר נוצרו. תווiotת זו גם היא למטרת נוחות וסדר, כדי שבקודה אחת נוכל למחוק את כל ה "בלאגן" שנוצר לנו.

1.5 קובץ ה章בות - myHeader.h

כאשר אנו מתעסקים בתכנות בשפת C, יש לנו כמה וכמה סוגים של קבצים. ראשית, ישנו (מן הסתם) קוד הכתובים בשפה הנ"ל וממניחים את הלוגיקה ואת הפקודות השונות. עם זאת, אלה לא הקבצים היחידים, ישנו גם קבצי "cotracts"/"ספריות" ("header"), אשר מהווים בסיס ויסוד לקובץ הקוד. קבצי הספריות מכילים ה章בות על קבועים וה章בות על פונקציות (ה章בות בלבד) כך שאם נזכיר את קובץ הכותרת לקובץ הקוד, הקוד יהיה מודע לכל הקבועים והפונקציות ויכול להשתמש בהם (אם קיים מימוש), למעשה קבציים אלה הם עיבוד מראש ("pre-processed").
כיוון ש כדי למשם את הלקוח והשרת, היינו צריכים להשתמש בכליה של קבצי ספריות רבים ויצירת מספר קבועים חשובים גם בלקוח וגם בשרת, החלפנו לצורך ספריה משלהו על מנת לשמר על הסדר, הנוחות והארגון בקוד.
מעבר על הקוד של הספריה שלנו ונסביר את הכתוב:

```
1  /* including libraries */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <stdbool.h>
5  #include <string.h>
6  #include <sys/types.h>
7  #include <sys/socket.h>
8  #include <arpa/inet.h>
9  #include <netinet/in.h>
10 #include <netinet/tcp.h>
11 #include <errno.h>
12 #include <unistd.h>
13 #include <time.h>
14 #include <sys/time.h>
15
16 /* defining constants */
17 #define IP "127.0.0.1" // server's ip address
18 #define PORT 8395 // connection port
19 #define BUFSIZE 8192 // size of the buffer
20 #define AUTH "0000000111111001" // authentication code
21 #define OK "ok" // ready message
22 #define CONNECTIONS 50 // number of clients that server can listen simultaneously
```

בלי הגבלת הכלליות, קודם כל נסביר את הקבועים שהגדכנו:

1. הגדרנו את כתובות ה-IP אשר תהיה בשימוש בעת שליחה לשרת (זהי הכתובת של השרת והלקוח שולח את המידע לכטוב האז).

2. הגדרנו את מספר ה-PORT שיהיה בשימוש וישמש לתקשורת בין הלקוח לשרת. המספר 8395 הוא אקרוי
ואין סיבה מיוחדת לבחירה זו (במילים אחרות השתמשנו במספר PORT כרצונו).

3. הגדרנו את גודל החוץ ("buffer") אשר יהיה בשימוש, החוץ יחזק בכל פעם את ההודעות המתקבלות
והשלוחות מהלקוח לשרת וההפך.

4. הגדרנו את הסימן המוסכם (האישור) שהמקבל שולח לשרת. הגענו למספר זהה כך:
ארבעת הספרות האחרונות של ת"ז של ליאור ה'ן - 1763, נעביר מספר זה לבינארית ונציג בתור 16 ביטים -
נקבל: 0000011011100011. ארבעת הספרות האחרונות של ת"ז של יועד ה'ן - 1818 נמיר לבינארית - נקבל:
.010000111111001. בצע, נבצע פעולה xor בין שתי התוצאות, נקבל בדיקות: 00000000111111001.

5. הגדרנו הודעה אישור על התחלה התקשורת בין השולח למქבל.

6. הגדרנו הודעה יציאה אשר נשלחת מהשולח למქבל כאשר משתמש מעוניין להפסיק לשולח קבצים ולצאת

7. הגדרנו את המספר המקסימלי של לקוחות שהמקבל יכול להזין בו זמני (ספקטיבית כאן, נדרש להזין לפחות אחד בלבד שהוא שלוח אבל באופן כללי בכלל שניתן, הגדרנו יותר).

כעת, נסביר את הספריות הספריות שהשתמשנו בהן:

1. שורות 3 – 1: ספריות קלט ופלט סטנדרטיות – שימוש חובה בכל תוכנית. הספריה השלישייה יוצרת אובייקט חדש עבור בוליאנים, לא השתמשנו בספריה זו אבל בכל מקרה החלטנו להוסיף כיוון שהוא ספרייה סטנדרטית ולמקרה שתוכל לבוא לעזרתינו.

2. שורה 5: ספריה עם פעולות רבות לטיפול במחזוזות, תורמת רבות לנוחות השימוש במחזוזות.

3. שורות 10 – 6 ושורה 12: אלו ספריות עבור תכנות שקעים ("SOCKET PROGRAMMING"). ספריות חובה בשבייל המטלה.

4. שורה 11: זהה ספריה בעל אובייקט אשר מחייב את שגיאת המערכת האחרונות שקרה, שימושית מאוד כיוון שבמקרים של תקלות החזינו למערכת ההפעלה את מספר השגיאה (פתרון אלגנטיבי יותר מאשר להחזיר 1 גנרי).

5. שורות 13 – 14: אלו ספריות אשר מוחזיקות זמנים, השימוש שלנו איתן היה בחישוב הזמנים של קובלות החלקים של הקובץ.

1.6 קובי הטעסט - msg.txt

הקובץ שבחרנו לשלוח מלהלכו לשרת הוא קובי גני לחלוין המכיל את הספרות 9–2. גודל הקובי הוא 1.1MB (הקובץ עומד בדרישות כיון שהוא גדול מ 1MB).
תמונה להמחשת הקובי (כמובן שזה לא כל הקובי, הוא גדול מידי ולא ניתן להציג את כלו):

```
1 2935893598593769278873924793356643562559997779752758933838982328793292872438822644742896253675338947967845  
259954667588875526952336242549985329223858674568989734823997336485476222469564352553234766495786527438526  
7967368877378366949262234345223792289386342236467265484734628887823774766595957357437275999825554426397274  
56955478766729983522593963727368626452239988445663298754975745598775825768297679823235496764296966424358  
8625397865924357548882476599986266249864522348629695272836372987889327493786929438477973358986626274729638  
3632339242582284337478743739923929489395583383954226329329794727565332772869463948588276683379783444663932  
944989362536694762748743294679822785996783383675897743228323387439565552647625639543445658454595426888775  
6663247878854622452432654926675752773832987624252279864395995426357679656496355772265368455768947264363447  
2833689747884854545246734689644379983679656945722236842369249656287523498358945323229653363753393535684988  
3685562593994948323956625987988628982455558765242923786882423972936954792437964597846875992295664243932482  
2483893668696538839722273674474349682894787968232686762675932285728644425326732225832895659479994645888725  
299822435386873834683792759843632396629478662294877834572539874339772445695829739535784654439346767799696  
583652938672456845822628847257449399767982737528767746897326493329843262653873833736682779742494665448925  
822346883937955632643387586235665249572366277487943874934936597975997698497826477523255336745723275376966  
7938848944844827234986539558599863574636636742692249994347297629854523986998829664939379383484962394897674  
864655593663999858459447737599866495242728882482385259353292963854598558734272934467693438793329526784548  
7253649688323459472363225672657755824274586995739342483236524995295749956948338563264682427664293258266744  
9434929948848382269268973487699458564772669729856793363383555655368473233662772367567954867557452828822882  
723296966885979994529568285327826466286283495578737329894724467574272963566395674868644378597958425488553  
459736353465796666999967476577357847985489432544827632488858646322957826633998552895325923658759625392395  
2772857837924544537383447727724855758728234356437648353868869647942445736652384263994336637695629233688653  
9426382933582984352877556574354978578849786462488825537737926793573873745594836536346949227728446649482375  
4829289458288735387879949946287395292823887838998664397626937697537438837337686627772449936678556575767582  
7686587278884286378426568443727923358844399598224536399454576527786556226742662866395538994592983979739794  
3873688855599688363782576546846552685585846327663526769248826589864643384485484999963644345383286927865259
```

אין יותר מידי מה להסביר בחלק זה, לכן נעבור לחלק הבא.

1.7 קובץ השולח - Sender.c

בקובץ זה כתבנו ל Koha (שלוח) אשר שולח קבצים לשרת (מקבל). נعبر על הקוד אשר כתבנו:

```
32  /* (1) opening and reading the file */
33  file = fopen("msg.txt" , "r");
34  if(file == NULL) // checking if file exists
35  {
36      perror("fopen() failed");
37      exit(errno);
38  }
39  printf("file opened!\n");
40
41  fseek(file , SEEK_SET , SEEK_END); //calculating the size of file
42  size = ftell(file);
43  rewind(file);
44  half = size / 2;
45
46  partA = (char*)malloc((half + 1 ) * sizeof(char)); // allocating and resetting memory for saving parts of file
47  partB = (char*)malloc((size - half + 1 ) * sizeof(char));
48  memset(partA, '\0' ,(half+1)*sizeof(char));
49  memset(partB, '\0' ,(size - half+1)*sizeof(char));
50
51  fread(partA, 1, half + 1, file); // coping the parts to allocated memory
52  fread(partB , 1, size - half + 1, file);
53
54  size1 = strlen(partA) + 1; // saving size of each part
55  size2 = strlen(partB) + 1;
56
57  fclose(file); //closing the file
```

כأن אנחנו רואים פותחים את הקובץ ובודקים שהוא אכן פתוח, אם לא אנו מוצאים מהתוכנית ומוחזרים את מספר השגיאה. לאחר מכן, אנו מחשבים את גודל הקובץ על מנת לחלק אותו לשני חצאים כדי שנוכל בהמשך לשולח כל אחד מהם. עברו כל אחד, אנו מזמנים זיכרון דינמי (גודל החלק) ומפסיקים אותו. לאחר מכן אנו אכן קוראים את החלק הרלוונטי מהקובץ הגדל לכל אחד מהזיכרון שהוקצו קודם לכן ואז אנו מחשבים את גודל המחרוזת שהועתקה ובסיום העבודה עם הקובץ אנו סוררים אותו.

```
59  /* (2) creating TCP connection */
60  sock = socket(AF_INET, SOCK_STREAM, 0); // creating the communication socket
61  if(sock <= 0) // checking if socket created
62  {
63      perror("socket() failed");
64      close(sock);
65      free(partA);
66      free(partB);
67      exit(errno);
68  }
69  printf("socket created!\n");
70
71  addr.sin_family = AF_INET; //setting up socket's used protocol, port and ip
72  addr.sin_port = htons(PORT);
73  addr.sin_addr.s_addr = inet_addr(IP);
74
75  if(connect(sock, (struct sockaddr*)&addr, sizeof(addr)) < 0) // checking if could connect to server
76  {
77      perror("connect() failed");
78      close(sock);
79      free(partA);
80      free(partB);
81      exit(errno);
82  }
83  printf("connected to server!\n");
```

כעת אנחנו פותחים שקע כדי לבצע תקשורת עם השרת, בודקים שהוא תקין ואם לא נמצא מהתוכנית. אם הוא תקין נדפיס הודעה מתאימה ונבצע התabbrות בפועל לשרת. לאחר מכן נדפיס למשתמש שהתחברנו בהצלחה או שנמצא מהתוכנית אם לא.

כאן אנו שולחים את הגודל של כל חלק לשרת על מנת שידע כמה לקבל (הקוד לשילוח של החלק השני זהה).

```
85     bytesSent = send(sock, &size1, sizeof(size1), 0); // sending the size of each part to the Receiver
86     if(bytesSent < 0)
87     {
88         perror("send() failed");
89         close(sock);
90         free(partA);
91         free(partB);
92         exit(errno);
93     }
94     else if(bytesSent == 0)
95     {
96         printf("peer has closed the TCP connection prior to send()\n");
97         close(sock);
98         free(partA);
99         free(partB);
100        exit(EXIT_FAILURE);
101    }
```

כאן אנו יוצרים תווית לחקירה במקרה שהמשתמש ירצה לחזור על התהליך, לאחר מכן אנו שולחים הודעה אוקי' לשרת ומצביעים שהוא יחזיר לנו הודעה אוקי', משמעו שהיא שחריבור קיים והשרת מוכן להתחילה לקבל נתונים.

```
123     REPEAT: // label for repeating process
124
125     bytesSent = send(sock, OK, strlen(OK) + 1, 0); // send ok message to the Receiver
126     if(bytesSent < 0)
127     {
128         perror("send() failed");
129         close(sock);
130         free(partA);
131         free(partB);
132         exit(errno);
133     }
134     else if(bytesSent == 0)
135     {
136         printf("peer has closed the TCP connection prior to send()\n");
137         close(sock);
138         free(partA);
139         free(partB);
140         exit(EXIT_FAILURE);
141     }
142
143     if(recv(sock, buffer, BUFSIZE, 0) <= 0) // receiving Receiver's answer
144     {
145         perror("recv() failed");
146         close(sock);
147         free(partA);
148         free(partB);
149         exit(errno);
150     }
151
152     if(strcmp(OK, buffer) != 0) // checking if receiver returns answers ok
153     {
154         printf("error occurred\n");
155         close(sock);
156         free(partA);
157         free(partB);
158         exit(EXIT_FAILURE);
159     }
160 }
```

כאן אנו שולחים את החלק הראשון של הקובץ ובודקים האם הוא נשלח בהצלחה, אם לא אז נצא מהתוכנית.
ואם רק חלק נשלח אנו מחשבים כמה אוחזים נשלחו. בכל מקרה אנו מדפיסים הודעה מותאמת.

```
161     /* (3) sending the first part */
162     bytesSent = send(sock, partA, size1, 0);
163     if(bytesSent < 0)
164     {
165         perror("send() failed");
166         close(sock);
167         free(partA);
168         free(partB);
169         exit(errno);
170     }
171     else if (bytesSent == 0) //check if the part sent clearly
172     {
173         printf("peer has closed the TCP connection prior to send()\n");
174         close(sock);
175         free(partA);
176         free(partB);
177         exit(EXIT_FAILURE);
178     }
179     else if (bytesSent < size1)
180     {
181         loss = (bytesSent / size1) * 100.0;
182         printf("could sent only %ld bytes from the required %ld bytes.\nlost %.2f %%bytes.", bytesSent, size1, loss);
183     }
184     else
185     {
186         printf("first half has been sent!\n");
187     }
```

כעת אנו ממתינים לאישור (אוטנטיקציה) מהשרת על קבלת החלק הראשון.

```
189     /* (4) checking for authentication */
190     printf("waiting for authentication...\n");
191     if(recv(sock, buffer, BUFSIZE, 0) <= 0) // receiving the authentication
192     {
193         perror("recv() failed");
194         close(sock);
195         free(partA);
196         free(partB);
197         exit(errno);
198     }
199     if(strcmp(AUTH, buffer) != 0) // checking if the authentication is correct
200     {
201         printf("authentication does not match\n");
202         close(sock);
203         free(partA);
204         free(partB);
205         exit(EXIT_FAILURE);
206     }
207     printf("authentication match!\n");
208 }
```

כאן אנו משנים את אלגוריתם בקרת העומס.

```
209     /* (5) changing CC algorithm */
210     CC = (strcmp(CC, "reno") == 0 ? "cubic" : "reno"); // determinating congestion control algorithm
211     if(setsockopt(sock, IPPROTO_TCP, TCP_CONGESTION, CC, sizeof(CC)) < 0) //changing congestion control algorithm
212     {
213         perror("setsockopt() failed");
214         close(sock);
215         free(partA);
216         free(partB);
217         exit(errno);
218     }
219     printf("CC algorithm has been changed to: \"%s\"\n", CC);
```

כעת, אנו שולחים את החלק השני של הודעה.

```
221     /* (6) sending the second part */
222     bytesSent = send(sock, partB, size2, 0); // send the second part
223     if(bytesSent < 0)
224     {
225         perror("send() failed");
226         close(sock);
227         free(partA);
228         free(partB);
229         exit(errno);
230     }
231     else if (bytesSent == 0) //check if the part sent clearly
232     {
233         printf("peer has closed the TCP connection prior to send()\n");
234         close(sock);
235         free(partA);
236         free(partB);
237         exit(EXIT_FAILURE);
238     }
239     else if (bytesSent < size2)
240     {
241         loss = (bytesSent / size2) * 100.0;
242         printf("could sent only %ld bytes from the required %ld bytes.\nlost %.2f %%bytes.", bytesSent, size2, loss);
243     }
244     else
245     {
246         printf("second half has been sent!\n");
247     }
```

כאן אנו שואלים את המשתמש האם ברצונו לחזור על התהיליך, אם לא איז שולחים הודעה יציאה לשרת.

```
249     /* (7) user decision */
250     printf("send file again? (y/n)\n"); // asking the user if send again
251     scanf(" %c", &d);
252     while(d != 'y' && d != 'n')
253     {
254         printf("please enter \'y\' to send or \'n\' to exit!\n");
255         scanf(" %c", &d);
256     }
257     if(d == 'n') // (7-b-i) user chose to exit
258     { /* (7-b-ii) sending exit message */
259         bytesSent = send(sock, EXIT, strlen(EXIT) + 1, 0); // send an exit message to the receiver
260         if(bytesSent < 0)
261         {
262             perror("send() failed");
263             close(sock);
264             free(partA);
265             free(partB);
266             exit(errno);
267         }
268         else if (bytesSent == 0) //check if the part sent clearly
269         {
270             printf("peer has closed the TCP connection prior to send()\n");
271             close(sock);
272             free(partA);
273             free(partB);
274             exit(EXIT_FAILURE);
275         }
276         else
277         {
278             printf("exit message has been sent\n");
279         }
280     }
281 }
```

אם כן איז אנו משנים שוב את אלגוריתם בקרת העומס וקופצים לתוויות - כלומר חוזרים על התהיליך.

```
282     else // (7-a-i) user chose to send file again
283     {
284         /* (7-a-ii) changing back the CC algorithm */
285         CC = (strcmp(CC, "reno") == 0 ? "cubic" : "reno"); // determining congestion control algorithm
286         if(setsockopt(sock, IPPROTO_TCP, TCP_CONGESTION, CC, sizeof(CC)) < 0) //changing congestion control algorithm
287         {
288             perror("setsockopt() failed");
289             close(sock);
290             free(partA);
291             free(partB);
292             exit(errno);
293         }
294         printf("CC algorithm has been changed to: \"%s\"\n", CC);
295         goto REPEAT; // (7-a-iii) repeating process
296     }
```

1.8 קובי המקלט - *Receiver.c*

נעבור על הקוד של המקלט:

כאן אנחנו ראשית יוצרים שקע שבו תבוצע התקשרות, לאחר מכן אנחנו בודקים האם הכתובת והפורט בשימוש ואם הכל תקין עד כה אנחנו מושרים את הכתובת ופורט לשקע. לאחר מכן מוכן השירות מאיין לתקשרות נכנסת אליו.

```
56     /* (1) creating TCP connection*/
57     server_sock = socket(AF_INET, SOCK_STREAM, 0); // creating the listener socket
58     if(server_sock <= 0) // checking if socket created
59     {
60         perror("socket() failed");
61         close(server_sock);
62         free(timeArray);
63         exit(errno);
64     }
65     printf("socket created!\n");
66
67     if(setsockopt(server_sock, SOL_SOCKET, SO_REUSEADDR, &er, sizeof(er)) < 0) // // checking if ip and port are reusable
68     {
69         perror("setsockopt() failed");
70         close(server_sock);
71         free(timeArray);
72         exit(errno);
73     }
74
75     server_addr.sin_family = AF_INET; //setting up socket's used protocol, port and ip
76     server_addr.sin_port = htons(PORT);
77     server_addr.sin_addr.s_addr = inet_addr(IP);
78
79     if(bind(server_sock, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) // binding socket with settings
80     {
81         perror("bind() failed");
82         close(server_sock);
83         free(timeArray);
84         exit(errno);
85     }
86     printf("socket bound!\n");
87
88     if(listen(server_sock, CONNECTIONS) < 0) //listen to incoming connections
89     {
90         perror("listen() failed");
91         close(server_sock);
92         free(timeArray);
93         exit(errno);
94     }
95     printf("waiting for connection...\\n");
```

כאן אנחנו מאשרים את ההתחברות של הלקוח לשרת.

```
97     /* (2) getting connection from the sender */
98     addr_size = sizeof(client_addr);
99     client_sock = accept(server_sock, (struct sockaddr*)&client_addr, &addr_size); //accept a connection
100    if(client_sock <= 0) // checking if accepted
101    {
102        perror("bind() failed");
103        close(client_sock);
104        close(server_sock);
105        free(timeArray);
106        exit(errno);
107    }
108    printf("sender connected!\n");
```

כאן אנחנו מקבלים את הגודל של כל אחד מהחקלים (הקבלת חלקו השני זהה)

```
112    byteRecv = recv(client_sock , &sizepart1 , sizeof(size_t) , 0); // receiving the size of first part of file
113    if(byteRecv < 0) // checkig if received
114    {
115        perror("send() failed");
116        close(client_sock);
117        close(server_sock);
118        free(timeArray);
119        exit(errno);
120    }
121    else if(byteRecv == 0)
122    {
123        printf("peer has closed the TCP connection prior to send()\n");
124        close(client_sock);
125        close(server_sock);
126        free(timeArray);
127        exit(EXIT_FAILURE);
128    }
```

כאן אנחנו מקבלים הודעה אוקי או הודעה יציאה.

```
150    memset(buffer, '\0', BUFSIZE);
151    byteRecv = recv(client_sock, buffer, BUFSIZE, 0); // receiving an ok or exit message
152    if(byteRecv < 0) // checkig if received
153    {
154        perror("send() failed");
155        close(client_sock);
156        close(server_sock);
157        free(timeArray);
158        exit(errno);
159    }
160    else if(byteRecv == 0)
161    {
162        printf("peer has closed the TCP connection prior to send()\n");
163        close(client_sock);
164        close(server_sock);
165        free(timeArray);
166        exit(EXIT_FAILURE);
167    }
168 }
```

במקרה של הودעת יציאה, נdfs_is למשתמש את הזמן ומצא מהתוכנית.

```
169     /* (10) if received exit message */
170     else if(strcmp(buffer, EXIT) == 0)
171     {
172         printf("exit message has been received!\n");
173         printf("\n\n");
174         printf("#####\n");
175         printf("\n");
176         printf("collected DATASET is:\n\n");
177
178         /* (10-i) printing out times */
179         for(i = 0; i < sizeTime - 2; i++)
180         {
181             if(i % 2 == 0)
182             {
183                 pf++;
184                 avg1 += timeArray[i];
185             }
186             else
187             {
188                 avg2 += timeArray[i];
189             }
190             printf("time to receive part %d of file %d is %f [ms]\n", (i % 2 == 0 ? 1 : 2), pf, timeArray[i]);
191         }
192
193         printf("\n");
194         printf("#####\n");
195         printf("\n");
196         printf("the times of avg part: \n");
197         avg1 /= (file - 1);
198         avg2 /= (file - 1);
199         /* (10-iii) printing the average times */
200         printf("the average of sending first file - by \"cubic\" - is: %.3f [ms]\n", avg1);
201         printf("the average of sending second file - by \"reno\" - is: %.3f [ms]\n", avg2);
202         printf("\n");
203         printf("#####\n");
204     }
205 }
```

נבדוק האם הלקוח שלח אוקי ונחזיר לו אוקי בחזרה אם נכון.

```
207     if(send(client_sock, OK, strlen(OK) + 1, 0) <= 0) // sending ok message to the Sender
208     {
209         perror("send() failed");
210         close(client_sock);
211         close(server_sock);
212         free(timeArray);
213         exit(errno);
214     }
```

כאן אנו מקבלים את החלק הראשון של הקובץ.

```
216     /* (3) receiving the first part */
217     memset(buffer, '\0', BUFSIZE); // clearing buffer
218     memset(&start, 0, sizeof(start)); // clearing times
219     memset(&start, 0, sizeof(end));
220     memset(&start, 0, sizeof(calc));
221     sum = 0;
222     gettimeofday(&start, NULL); // get start time
223     while (sum != sizepart1) // receive the first part
224     {   byteRecv = recv(client_sock, buffer, BUFSIZE, 0);
225         if(byteRecv < 0)
226         {
227             perror("send() failed");
228             close(client_sock);
229             close(server_sock);
230             free(timeArray);
231             exit(errno);
232         }
233         else if(byteRecv == 0)
234         {
235             printf("peer has closed the TCP connection prior to send()\n");
236             close(client_sock);
237             close(server_sock);
238             free(timeArray);
239             exit(EXIT_FAILURE);
240         }
241         sum += byteRecv;
242     }
243     gettimeofday(&end, NULL); // get end time
244     total+= sum;
245     if(sum < sizepart1)
246     {
247         printf("the receiver only received %ld of %ld bytes of the first part" , sum , sizepart1);
248     }
249     else
250     {
251         printf("first message has been received!\n");
252     }
```

לאחר מכן אנו מוחשבים את זמן הקבלה ושמורים אותו.

```
255     /* (4) measuring the time */
256     timersub(&end, &start, &calc); // calculate the difference
257
258     /* (5) saving the time */
259     timeArray[last] = (((calc.tv_sec * 1000000.0) + calc.tv_usec)/1000.0); //save the time in mili-seconds
260     last++;
```

כאן אנו שולחים אוטנטיקציה ללקוח.

```
262     /* (6) sending back the authentication */
263     if(send(client_sock, AUTH, strlen(AUTH) + 1, 0) <= 0) //send the authentication
264     {
265         perror("send() failed");
266         close(client_sock);
267         close(server_sock);
268         free(timeArray);
269         exit(errno);
270     }
271     else
272     {
273         printf("authentication has been sent!\n");
274     }
```

כאן אנחנו משנים את אלגוריתם בקרת העומס

```
276     /* (6-i) changing the CC algorithm */
277     CC = (strcmp(CC, "reno") == 0 ? "cubic" : "reno"); // determinating congestion control algorithm
278     if(setsockopt(client_sock, IPPROTO_TCP, TCP_CONGESTION, CC, sizeof(CC)) < 0) //changing congestion control algorithm
279     {
280         perror("setsockopt() failed");
281         close(client_sock);
282         close(server_sock);
283         free(timeArray);
284         exit(errno);
285     }
286     printf("CC algorithm has been changed to: \"%s\"\n", CC);
```

כאן אנו מקבלים את החלק השני של הקובץ.

```
288     /* (7) receiving the second part */
289     memset(buffer, '\0', BUFSIZE); // clearing buffer
290     memset(&start, 0, sizeof(start)); // clearing times
291     memset(&start, 0, sizeof(end));
292     memset(&start, 0, sizeof(calc));
293     sum = 0;
294     gettimeofday(&start, NULL); //set the start time
295     while (sum != sizepart2)
296     {
297         byteRecv = recv(client_sock, buffer, BUFSIZE, 0);
298         if(byteRecv < 0)
299         {
300             perror("send() failed");
301             close(client_sock);
302             close(server_sock);
303             free(timeArray);
304             exit(errno);
305         }
306         else if(byteRecv == 0)
307         {
308             printf("peer has closed the TCP connection prior to send()\n");
309             close(client_sock);
310             close(server_sock);
311             free(timeArray);
312             exit(EXIT_FAILURE);
313         }
314         sum += byteRecv;
315     }
316     gettimeofday(&end, NULL);
317     total+= sum;
318     if(sum < sizepart1)
319     {
320         printf("the receiver only received %ld of %ld bytes of the second part" , sum , sizepart2);
321     }
322     else
323     {
324         printf("second message has been received!\n");
325     }
```

כאן אנו שוב שומרים את הזמן של קבלת החלק ומגדילים את המערך (הдинמי) של הזמן.

```
327     /* (8) measuring the time */
328     timersub(&end, &start, &calc);
329
330     /* (9) saving the time*/
331     timeArray[last] = (((calc.tv_sec * 1000000.0) + calc.tv_usec)/1000.0); // set the end time
332     last++;
333     file++;
334     sizeTime = 2 * file;
335     timeArray = (double*)realloc(timeArray, sizeTime * sizeof(double));
```

כאן אנו שוב מושנים את אלגוריתם בקרת העומס (כדי שהיה סנכרו).

```
343     CC = (strcmp(CC, "reno") == 0 ? "cubic" : "reno"); // determining congestion control algorithm
344     if(setsockopt(client_sock, IPPROTO_TCP, TCP_CONGESTION, CC, sizeof(CC)) < 0) //changing congestion control algorithm failed
345     {
346         perror("setsockopt() failed");
347         close(client_sock);
348         close(server_sock);
349         free(timeArray);
350         exit(errno);
351     }
352     printf("CC algorithm has been changed to: \"%s\"\n", CC);
```

2 פרק ב' - המחבר על אלגוריתמי בקרת העומס

2.1 הקדמה

במיטלה זו נדרש לכתוב שני קבצים, האחד ל Koh (שלוח) והשני Shart (מקבל) ולבצע בהם פועלות של תעבורות רשות, שהן שליחת וקבלת של הודעות. פועלות אלו של תעבורות הרשות, בוצעו באמצעות פרוטוקול התקשורת *TCP* אשר הוא פרוטוקול תקשורת אמין אשר מתמקד בהונגות על ידי חילוק רוחב הפס של תעבורות הרשות (הפרוטוקול מחלק את רוחב הפס באופן שווה בין כל המשתמשים המוחברים בו-זמנית). השגת החלוקה השווה זו מותאפשרת באמצעות אלגוריתמי בקרת ותוכנו עומס (*Congestion Control*). בפרק זה עוסק כיצד איבוד חבילות (ובעරת שימוש באלגוריתמי הבקרה) משפיע על זמני השיליחה והקבלת של קבצים ברשות בפרוטוקול זה.

את הסקירה של תעבורות הרשות והחבילות הנשלחות והתקבלות, נבצע באמצעות התוכנה "Wireshark", נשימוש בה באופן הבא:

1. נבחר אחו איבוד חבילות מסוימים.
2. נפעיל את סריקת ה-"Wireshark".
3. נריץ את המקלט.
4. נריץ את השולח.
5. נשלח את הקובץ 5 פעמים.
6. נעצור את שתי התוכניות.
7. נתבונן בתוצאות שקיבלנו - תעבורות הרשות ומימי הקבלה שחושו.
8. נסיק מסקנות מותאיימות.

נפעיל את ה-"Wireshark", נבחר את האפשרות לרחרח חבילות ב- "lo : LOOPBACK", שזו רשות תקשורת עם המארח המקומי ("localhost : 127.0.0.1"). כיוון שהתוכנות שכתבנו רצויות על המחשב שלנו עצמו ולא מתחברות לשאר רשת האינטרנט, נבחר באפשרות זו.
בחירה זו תהיה זהה לכל הבדיקות!

Capture

...using this filter: Enter a capture filter ...

enp0s3	—
enp0s8	—
any	—
Loopback: lo	—
bluetooth-monitor	—
nflog	—
nfqueue	—
dbus-system	—
dbus-session	—
Cisco remote capture: ciscodump	—
DisplayPort AUX channel monitor capture: dpauxmon	—

לאחר שהחרנו, נפעיל את הלקוק והשרת ונכפה לקבל התובורה. בעת שימוש ב프וטוקול "TCP", על הלקוק קודם כל לוודא שהשרת זמין ומוכן לתקשורת (כיון שהוא פרוטוקול אמין), הבדיקה הזאת תבוצע על ידי "חיצות ידים משולשת", הבדיקה מבוצעת כך: נשלחת חבילה ראשונית מהשולח למטרך שהיא חביבת "SYN" חביבו זו אינה מכילה מידע. לאחר מכן, נשלחת חביבה שנייה מטרך לשלוח חביבה זו היא חביבת "SYN-ACK" פירושה שהשרת זמין ומאשר שקיבל את החביבה הראשונה שנשלחה. לסיום, נשלחת חביבה שלישית "ACK", חביבה זו כבר יכולה (להכיל מידע רלוונטי שהלקוק ורוצה לשולח).

לאחר שפתחנו קשר עם הרשת, נתחיל לבצע את הבדיקות שלנו עם אחוזי איבוד הפקחות:

No.	Time	Source	Destination	Protocol	Length	Info
9	1.792721300	127.0.0.1	127.0.0.1	TCP	74	484598 8395 [SYN] Seq=0 Win=65536 Len=0 TStamp=1824806925 TSrcr=0 WS=128
10	1.792721300	127.0.0.1	127.0.0.1	TCP	74	484598 8395 [SYN] Seq=0 Win=65536 Len=0 TStamp=1824806925 TSrcr=0 WS=128
11	1.792724450	127.0.0.1	127.0.0.1	TCP	66	484598 8395 [ACK] Seq=1 Win=65536 Len=0 TStamp=1824806925 TSrcr=0 WS=128
12	1.792779229	127.0.0.1	127.0.0.1	TCP	74	484598 8395 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=8 TStamp=1824806925 TSrcr=1824806925
13	1.792783754	127.0.0.1	127.0.0.1	TCP	66	8395 - 48598 [ACK] Seq=1 Ack=9 Win=65536 Len=0 TStamp=1824806925 TSrcr=1824806925
14	1.792789773	127.0.0.1	127.0.0.1	TCP	74	484598 8395 [PSH, ACK] Seq=9 Ack=1 Win=65536 Len=8 TStamp=1824806925 TSrcr=1824806925
15	1.792801465	127.0.0.1	127.0.0.1	TCP	66	8395 - 48598 [ACK] Seq=1 Ack=17 Win=65536 Len=0 TStamp=1824806925 TSrcr=1824806925
16	1.792811933	127.0.0.1	127.0.0.1	TCP	69	484598 8395 [PSH, ACK] Seq=17 Ack=1 Win=65536 Len=3 TStamp=1824806925 TSrcr=1824806925
17	1.792822388	127.0.0.1	127.0.0.1	TCP	66	8395 - 48598 [ACK] Seq=1 Ack=20 Win=65536 Len=0 TStamp=1824806925 TSrcr=1824806925
18	1.792888472	127.0.0.1	127.0.0.1	TCP	69	8395 - 48598 [PSH, ACK] Seq=20 Ack=20 Win=65536 Len=3 TStamp=1824806925 TSrcr=1824806925

ראשית, נשלח לשרת את הגדול של החלק הראשון של הקובץ ולאחר מכן גם את החלק השני של הקובץ (התמונה היא של השילחה של החלק הראשון, החלק השני יהיה לחולוטין):

12	1.792778220	127.0.0.1	127.0.0.1	TCP	74	48598	- 8395	[PSH, ACK]	Seq=1	Ack=1	Win=65536	Len=9	Tsval=1824806925	Tserr=1824806925
13	1.792811893	127.0.0.1	127.0.0.1	TCP	74	48598	- 8395	[ACK]	Seq=1	Ack=9	Win=65536	Len=9	Tsval=1824806925	Tserr=1824806925
14	1.792811893	127.0.0.1	127.0.0.1	TCP	74	48598	- 8395	[ACK]	Seq=1	Ack=1	Win=65536	Len=8	Tsval=1824806925	Tserr=1824806925
15	1.792801405	127.0.0.1	127.0.0.1	TCP	66	8395	- 8395	[ACK]	Seq=1	Ack=17	Win=65536	Len=9	Tsval=1824806925	Tserr=1824806925
16	1.792811893	127.0.0.1	127.0.0.1	TCP	69	48598	- 8395	[PSH, ACK]	Seq=17	Ack=1	Win=65536	Len=3	Tsval=1824806925	Tserr=1824806925
17	1.792822396	127.0.0.1	127.0.0.1	TCP	66	8395	- 8395	[ACK]	Seq=1	Ack=20	Win=65536	Len=9	Tsval=1824806925	Tserr=1824806925
18	1.79288847	127.0.0.1	127.0.0.1	TCP	69	8395	- 8395	[PSH, ACK]	Seq=1	Ack=20	Win=65536	Len=9	Tsval=1824806925	Tserr=1824806925

כעת, כדי להודיע לשרת שאנחנו מוכנים להתחילה לשלוח לו את הקובץ, אנו שולחים לו הודעה "ok" ומצביעים שיחזיר לנו אותה, נראה:

```

16 1.7920111893 127.0.0.1      127.0.0.1      TCP    69 48598 - 8395 [PSH, ACK] Seq=17 Ack=1 Win=65536 Len=3 Tsvl=1824806925 Tsecr=1824806925
17 1.7928223390 127.0.0.1      127.0.0.1      TCP    69 8395 - 48598 [ACK] Seq=1 Win=65536 Len=9 Tsvl=1824806925 Tsecr=1824806925
18 1.792888847 127.0.0.1       127.0.0.1      TCP    69 8395 - 48598 [PSH, ACK] Seq=1 Ack=28 Win=65536 Len=3 Tsvl=1824806925 Tsecr=1824806925

Frame 16: 69 bytes on wire (552 bits), 69 bytes captured (552 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 48598, Dst Port: 8395, Seq: 17, Ack: 1, Len: 3
Data (3 bytes)

```

0020 00 01 bd d6 20 cb ea f0 79 a3 d1 34 f5 1e 80 18 y 4 ...
0030 02 00 fe 2b 00 00 01 01 08 0a 6c c4 58 0d 6c c4 ... + . . . x l x l .
0040 58 0d 6f b5 00 X-Ok.

לאחר מכן, ניתן לראות בתבילה 17 כי השרת קיבל את הודעה ואכן חביבה 18 היא התשובה שאנו מצפים לה:

17 1.792822380 127.0.0.1 127.0.0.1 TCP 66 8395 → 48598 [ACK] Seq=1 Ack=20 Win=65536 Len=0 TStamp=1824806925 TSectr=1824806925
18 1.792888847 127.0.0.1 127.0.0.1 TCP 69 8395 → 48598 [PSH, ACK] Seq=1 Ack=20 Win=3 Len=3 TStamp=1824806925 TSectr=1824806925
> Frame 18: 69 bytes on wire (552 bits), 69 bytes captured (552 bits) on interface lo, id 0
> Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 8395, Dst Port: 48598, Seq: 1, Ack: 20, Len: 3
. Data (3 bytes)

0029 00 01 29 cb bd d6 ad 34 f5 1e ea f0 79 a6 88 18 4 Y
0030 02 00 fe 2b 00 00 01 81 08 0a 6c c4 58 0d 6c c4 1 X 1
0040 58 0d 6f 6b 00 X ok:

החל מנקודה זו, יהיה רלוונטי להסתכל על ההשלכות של איבוד החבילות בראשת.

2.2 ניסוי 1 - 0% איבוד חבילות

ראשית נפתח חלון טרמינל ונרשום את הפקודה הבאה על מנת לאפס את איבוד החבילות (במקרה והיה דלקה):

”sudo tc qdisc del dev lo root netem”

כעת נסתכל מהו קורה בתעבורה.

2.2.1 שליחת החבילות

כעת, נשלחות החבילות עם המידע הרלוונטי באמת, ז"א נשלחות מהלכמה לשרת החבילות עם המידע מהקובץ ונשלחות מהשרת ללקוח חבילות של אישורים למיניהם. הקובץ גדול, אך הלוקוח שלוח אותו לשרת במקטעים ("סגננטים"). נראה כי בתבילה זו נשלח הסגמנט הראשון (לדוגמה):

20 1.792957232 127.0.0.1 127.0.0.1 TCP 328 48598 → 8395 [ACK] Seq=20 Ack=4 Win=65536 Len=32768 TStamp=1824806926 TSectr=1824806925
21 1.793986284 127.0.0.1 127.0.0.1 TCP 66 48598 → 8395 [ACK] Seq=4 Ack=20 Win=65536 Len=0 TStamp=1824806926 TSectr=1824806926
22 1.793986284 127.0.0.1 127.0.0.1 TCP 328 48598 → 8395 [ACK] Seq=5 Ack=21 Win=65536 Len=0 TStamp=1824806926 TSectr=1824806926
23 1.793924970 127.0.0.1 127.0.0.1 TCP 66 8395 → 48598 [ACK] Seq=4 Ack=65536 Win=65536 Len=0 TStamp=1824806926 TSectr=1824806926
24 1.793175336 127.0.0.1 127.0.0.1 TCP 328 48598 → 8395 [ACK] Seq=65536 Ack=4 Win=65536 Len=32768 TStamp=1824806926 TSectr=1824806926
> Frame 20: 32834 bytes on wire (262672 bits), 32834 bytes captured (262672 bits) on interface lo, id 0
> Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 48598, Dst Port: 8395, Seq: 20, Ack: 4, Len: 32768
. Data (32768 bytes)

0040 58 0d 32 39 33 35 38 39 33 35 39 38 35 39 33 37 X 293589 35985937
0050 36 39 32 37 38 38 37 33 39 32 34 37 39 33 35 69278873 92479335
0060 36 36 34 33 35 36 32 35 35 39 39 37 37 37 39 66435625 59997779

כיוון שאין איבוד חבילות מופעל, אין נשים לב כי גם כל החבילות מתחת לחביבה 20 הן גם חבילות של שליחה וקיבלה ואין חבילות אשר נאבדות באמצעותם.

cut נרצה לקבל אימות (אוטנטיקציה) מהשרת על השילוח של החלק הראשון, נראה:

49 1.793974995 127.0.0.1	127.0.0.1	TCP	83 8395 - 48598 [PSH, ACK]	Sent=4 Ack=550027 Win=1244288 TStamp=1824806926 TSectr=1824806926
50 1.794088970 127.0.0.1	127.0.0.1	TCP	655..48598 - 8395 [ACK]	Seq=550027 Ack=21 Win=65536 Len=65483 TStamp=1824806927 TSectr=1824806926
Frame 49: 83 bytes on wire (664 bits), 83 bytes captured (664 bits) on interface lo, id 0				
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)				
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1				
Transmission Control Protocol, Src Port: 8395, Dst Port: 48598, Seq: 4, Ack: 550027, Len: 17				
Data (17 bytes)				

00:50 25 f9 fe 39 00 00 01 01 08 08 c4 58 0e 6c c4 % ..9LX.1.	X 000000 0111110
00:40 50 0e 30 30 30 30 30 30 31 31 31 31 31 31 30	51
00:50 30 31 00	

ושוב כיוון שאין איבוד חבילות מופעל אין חבילות שנאבדות. cut נשלח את החלק השני של הקובץ (להלן סגמנט לדוגמה):

50 1.793974995 127.0.0.1	127.0.0.1	TCP	655..48598 - 8395 [ACK]	Seq=550027 Ack=21 Win=65536 Len=65483 TStamp=1824806927 TSectr=1824806926
51 1.794088970 127.0.0.1	127.0.0.1	TCP	655..48598 - 8395 [ACK]	Seq=615510 Ack=21 Win=65536 Len=65483 TStamp=1824806927 TSectr=1824806926
Frame 50: 65549 bytes on wire (524392 bits), 65549 bytes captured (524392 bits) on interface lo, id 0				
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)				
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1				
Transmission Control Protocol, Src Port: 8395, Dst Port: 48598, Seq: 550027, Ack: 21, Len: 65483				
Data (65483 bytes)				

00000040 58 0e 39 32 33 36 37 36 35 37 36 37 35 37 37 32 X 093676 57675772	
00000050 36 36 38 32 38 37 37 36 35 37 32 33 33 35 38	66828776 57233358
00000060 34 34 35 34 32 36 36 34 34 34 33 32 36 33 32 38	44542664 44326328

כעת, מכיוון שאנו חווים על התהיליך (לפי הדרישות צריך להריץ לפחות 5 פעמים), אנו שולחים הודעה "ok" לשרת וגם מצפים שהוא יחזיר לנו אותה, נראה זאת בתמונות הבאות:

64.3.5.000669255 127.0.0.1	127.0.0.1	TCP	69 48598 - 8395 [PSH, ACK] Seq=1100032 Ack=21 Win=65536 Len=3 TSval=1824808633 TSecr=1824808633
65.3.5.000678624 127.0.0.1	127.0.0.1	TCP	69 48595 - 48598 [ACK] Seq=21 Ack=1100035 Win=2422912 Len=0 TSval=1824808633 TSecr=1824808633
66.3.5.000718299 127.0.0.1	127.0.0.1	TCP	69 48395 - 48598 [PSH, ACK] Seq=21 Ack=1100035 Win=2422912 Len=3 TSval=1824808633 TSecr=1824808633
67.3.5.000824737 127.0.0.1	127.0.0.1	TCP	655.48598 - 8395 [ACK] Seq=1100035 Ack=24 Win=65536 Len=65483 TSval=1824808633 TSecr=1824808633
68.3.5.000861382 127.0.0.1	127.0.0.1	TCP	655.48598 - 8395 [ACK] Seq=1165518 Ack=24 Win=65536 Len=65483 TSval=1824808633 TSecr=1824808633
69.3.5.000873081 127.0.0.1	127.0.0.1	TCP	66 48395 - 48598 [ACK] Seq=24 Ack=1231001 Win=2422912 Len=0 TSval=1824808633 TSecr=1824808633

Frame 64: 69 bytes on wire (552 bits), 69 bytes captured (552 bits) on interface lo, id 0

Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

Transmission Control Protocol, Src Port: 48598, Dst Port: 8395, Seq: 1100032, Ack: 21, Len: 3

Data (3 bytes)

0020 00 01 bd d6 29 cb eb 01 42 92 ad 34 f5 32 80 18 B . 4 . 2 .
0030 02 00 fe 2d 00 00 01 01 08 0a 6c c4 5e b9 6c c4	... + . . . l ^ . l
0040 58 3b 0f bb 08	X;ok.

66.3.5.000716290 127.0.0.1	127.0.0.1	TCP	69 8395 - 48598 [PSH, ACK] Seq=21 Ack=1100035 Win=2422912 Len=3 TSval=1824808633 TSecr=1824808633
67.3.5.000861382 127.0.0.1	127.0.0.1	TCP	655.48598 - 8395 [ACK] Seq=1100035 Ack=24 Win=65536 Len=65483 TSval=1824808633 TSecr=1824808633
68.3.5.00088513801 127.0.0.1	127.0.0.1	TCP	655.48598 - 8395 [ACK] Seq=1165518 Ack=24 Win=65536 Len=65483 TSval=1824808633 TSecr=1824808633
69.3.5.0008873081 127.0.0.1	127.0.0.1	TCP	66 8395 - 48598 [ACK] Seq=24 Ack=1231001 Win=2422912 Len=0 TSval=1824808633 TSecr=1824808633

Frame 66: 69 bytes on wire (552 bits), 69 bytes captured (552 bits) on interface lo, id 0

Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

Transmission Control Protocol, Src Port: 8395, Dst Port: 48598, Seq: 21, Ack: 1100035, Len: 3

Data (3 bytes)

0020 00 01 20 cb bd d6 ad 34 f5 32 eb 01 42 95 80 18 4 . 2 . B .
0030 49 f1 fe 2b 00 00 01 01 08 0a 6c c4 5e b9 6c c4	I . + . . . l ^ . l
0040 5e b9 6f bb 08	A;ok.

כעת, אנו חווים על כל התהיליך, הפלטים שאנו מקבלים זהים לחלוין וכן נסתפק בצלומים שראינו עד כה. כעת, כאשר הלקוח מבקש לסייע את התהיליך, נשלחת הודעה יציאה נראה:

604.0.351593895 127.0.0.4	127.0.0.4	TCP	74 [TCP Previous segment not captured] 48598 - 8395 [PSH, ACK] Seq=5260092 Ack=102 Win=65536 Len=6 TS
162.0.3.351633889 127.0.0.1	127.0.0.1	TCP	66 [TCP ACKed unsolicited segment] 8395 - 48598 [ACK] Seq=101 Ack=5500097 Win=3132032 Len=0 TSval=1824813484 TSecr=1824813484
163.0.3.351630157 127.0.0.1	127.0.0.1	TCP	66 48598 - 8395 [FIN, ACK] Seq=5500097 Ack=101 Win=65536 Len=0 TSval=1824813484 TSecr=1824813484
164.0.3.351720590 127.0.0.1	127.0.0.1	TCP	66 8395 - 48598 [FIN, ACK] Seq=101 Ack=5500098 Win=3132032 Len=0 TSval=1824813484 TSecr=1824813484
165.0.3.351727590 127.0.0.1	127.0.0.1	TCP	66 48598 - 8395 [ACK] Seq=5500098 Ack=102 Win=65536 Len=0 TSval=1824813484 TSecr=1824813484

Frame 161: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface lo, id 0

Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

Transmission Control Protocol, Src Port: 48598, Dst Port: 8395, Seq: 5500098, Ack: 101, Len: 5

Data (5 bytes)

0020 00 01 20 cb bd d6 29 cb eb 44 66 4e ad 34 f5 82 80 18 0 fN 4 . . .
0030 02 00 fe 2d 00 00 01 01 08 0a 6c c4 71 ac 6c c4 l q l
0040 6d 17 05 78 69 74 08	m;exit.

2.2.2 מס' חבילות שנשלחו

נשלחו בסך הכל: 165 חבילות.

	159.10.199.227	127.0.0.1	127.0.0.1	TCP	66	48598 - 8395 [ACK] Seq=101 Win=0 TSval=1824813484 TSecr=1824813484	65536 - 8395 [ACK] Seq=101 Win=0 TSval=1824813484 TSecr=1824813484
166	7.17.8771610	127.0.0.1	127.0.0.1	TCP	66	[TCP ACKed Unseen segment] 8395 - 48598 [ACK] Seq=101 Ack=55009029 Win=3123023 Len=0 TSval=18248123..	
161	8.351593636	127.0.0.1	127.0.0.1	TCP	71	[TCP Previous segment not captured] 48598 - 8395 [PSH, ACK] Seq=55009029 Ack=101 Win=65536 Len=5 TS..	
162	8.351613389	127.0.0.1	127.0.0.1	TCP	66	[TCP ACKed Unseen segment] 8395 - 48598 [ACK] Seq=101 Ack=55009097 Win=3123023 Len=0 TSval=18248134..	
163	8.351639157	127.0.0.1	127.0.0.1	TCP	66	48598 - 8395 [FIN, ACK] Seq=55009097 Ack=101 Win=65536 Len=0 TSval=1824813484 TSecr=1824813484	
164	8.351720590	127.0.0.1	127.0.0.1	TCP	66	8395 - 48598 [FIN, ACK] Seq=101 Ack=55009098 Win=3123023 Len=0 TSval=1824813484 TSecr=1824813484	
165	8.351727590	127.0.0.1	127.0.0.1	TCP	66	48598 - 8395 [ACK] Seq=55009098 Ack=102 Win=65536 Len=0 TSval=1824813484 TSecr=1824813484	

2.2.3 תוצאות

עד כה ראיינו את תובורת הרשות בתחום התקשורת, בעת נרצה כיצד אפס אחוז של איבוד פקודות, ושני אלגוריתמי בקרת העומס משפיעים על הזמן.

ראשית, נראה את הפלט של התוכניות בסוף הריצה. מצד ימין זהו פלט בצד הלוקה ובצד ימין זהו הפלט בצד השרת:

```

authentication has been sent!
CC algorithm has been changed to: "reno"
second message has been received!
CC algorithm has been changed to: "cubic"
first message has been received!
authentication has been sent!
CC algorithm has been changed to: "reno"
second message has been received!
CC algorithm has been changed to: "cubic"
exit message has been received!

#####
collected DATASET is:

time to receive part 1 of file 1 is 0.272000 [ms]
time to receive part 2 of file 1 is 0.258000 [ms]
time to receive part 1 of file 2 is 0.180000 [ms]
time to receive part 2 of file 2 is 0.133000 [ms]
time to receive part 1 of file 3 is 0.202000 [ms]
time to receive part 2 of file 3 is 0.130000 [ms]
time to receive part 1 of file 4 is 0.302000 [ms]
time to receive part 2 of file 4 is 0.202000 [ms]
time to receive part 1 of file 5 is 0.252000 [ms]
time to receive part 2 of file 5 is 0.175000 [ms]

#####
the times of avg part:
the average of sending first file - by "cubic" - is: 0.242 [ms]
the average of sending second file - by "reno" - is: 0.180 [ms]

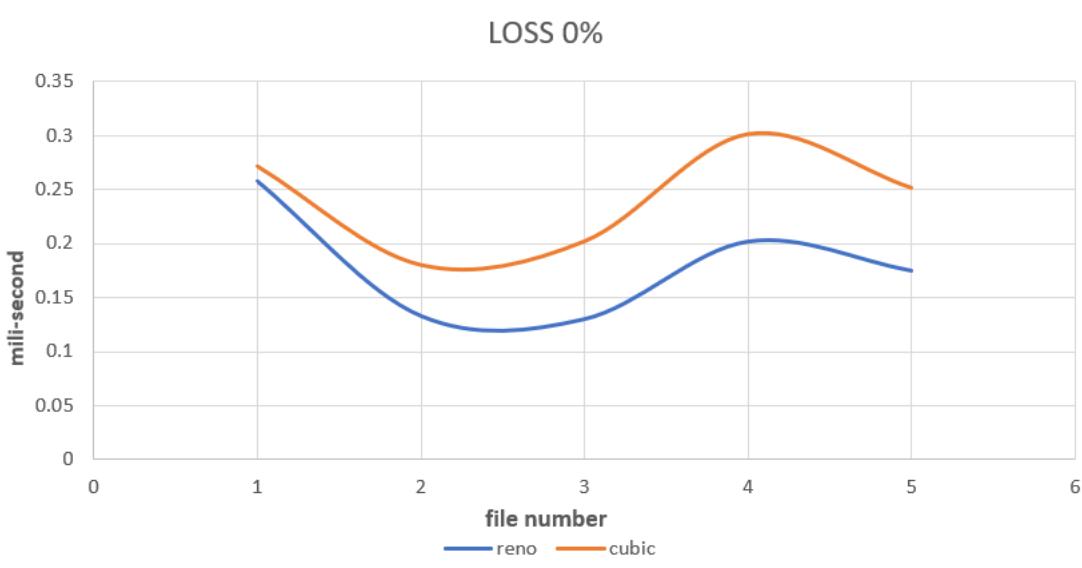
#####
yoad@yoad-VirtualBox:~/Desktop/test$ 
```

```

y
CC algorithm has been changed to: "cubic"
first half has been sent!
waiting for authentication...
authentication match!
CC algorithm has been changed to: "reno"
second half has been sent!
send file again? (y/n)
y
CC algorithm has been changed to: "cubic"
first half has been sent!
waiting for authentication...
authentication match!
CC algorithm has been changed to: "reno"
second half has been sent!
send file again? (y/n)
y
CC algorithm has been changed to: "cubic"
first half has been sent!
waiting for authentication...
authentication match!
CC algorithm has been changed to: "reno"
second half has been sent!
send file again? (y/n)
y
CC algorithm has been changed to: "cubic"
first half has been sent!
waiting for authentication...
authentication match!
CC algorithm has been changed to: "reno"
second half has been sent!
send file again? (y/n)
n
exit message has been sent
yoad@yoad-VirtualBox:~/Desktop/test$ 
```

נסכם את התוצאות (הפלט הרלוונטי) בטבלה הבאה ונציג את גרפ' התוצאות:

	אלגוריתם/מספר-קובץ	CUBIC	RENO
1	0.272[MS]	0.258[MS]	
2	0.180[MS]	0.133[MS]	
3	0.202[MS]	0.130[MS]	
4	0.302[MS]	0.202[MS]	
5	0.252[MS]	0.175[MS]	



האמינים המומוצעים:

1. אלגוריתם $0.242[ms]$:*cubic*

2. אלגוריתם $0.180[ms]$:*reno*

2.2.4 מסקנות

המסקנה המתבקשת - אלגוריתם ה-*reno* מהיר יותר.

2.3 ניסוי 2 - 10% איבוד חבילות

הפעם, נפעיל את איבוד חבילות התקשרות על ידי הפקודה הבאה:

`"sudo tc qdisc del dev lo root netem 10%"`

נסתכל על התעבורת:

2.3.1 שליחת החבילות

מכיוון שיש לנו איבוד חבילות, כאשר הלקות שולח לשרת לא בהכרח שבזאת התהילך השילוח והקבלת יסתאים. מכיוון שפרוטוקול "TCP" הוא פרוטוקול SMBTIEkus לו אמינותו הוא חייב לבדוק האם כל פקטה הגיעה בשלום מה כתוביה היעד שלו, אך אם פקטה לא הגיעה בשלהי - הפרוטוקול יחייב לשלוח אותה מחדש. נראה דוגמה לאיבוד חבילות:

```
| 10 0.282635067 127.0.0.1      127.0.0.1      TCP    328... [TCP Window Full]  [TCP Previous segment not captured] 53016 .. 8395 [PSH, ACK] Seq=32788 Ack=4 Win=..  
| 11 0.28267336 127.0.0.1       127.0.0.1      TCP    78 [TCP Dup ACK 7#1] 8395 .. 53016 [ACK] Seq=4 Ack=20 Win=65536 Len=0 Tsvl=1828864363 Tscr=1828864363  
| 12 0.282673336 127.0.0.1      127.0.0.1      TCP    328... [TCP Retransmission] 53016 .. 8395 [ACK] Seq=20 Ack=4 Win=65536 Len=32768 Tsvl=1828864363 Tscr=18...  
| 13 0.282685524 127.0.0.1      127.0.0.1      TCP    66 [TCP ZeroWindow] 8395 .. 53016 [ACK] Seq=4 Ack=65556 Win=0 Len=0 Tsvl=1828864363 Tscr=1828864363  
  
Window: 512  
[Calculated window size: 65536]  
[Window size scaling factor: 128]  
Checksum: 0x7e29 [unverified]  
[Checksum Status: Unverified]  
Urgent Pointer: 0  
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps  
[Timestamps]  
[SEQ/ACK analysis]  
[iRTT: 0.000036467 seconds]  
- [TCP Analysis Flags]  
- [Expert Info (Warning/Sequence): Previous segment(s) not captured (common at capture start)]  
  [Previous segment(s) not captured (common at capture start)]  
  [Severity level: Warning]  
  [Group: Sequence]  
- [Expert Info (Warning/Sequence): TCP window specified by the receiver is now completely full]  
  [TCP window specified by the receiver is now completely full]  
  [Severity level: Warning]  
  [Group: Sequence]  
  TCP payload (32768 bytes)  
, Data (32768 bytes)  
  
0040  41 5c 32 34 34 32 36 35 35 33 32 34 34 35 32 33  A\244265 58244523  
0050  33 34 39 36 34 35 35 39 35 34 34 35 37 35 35 37  34964559 54457557  
0060  39 35 37 32 32 36 32 35 37 34 35 38 35 34 36  9572625 74585446
```

כאן אנו רואים שהחטילה הולכת לאיבוד וה-"Wireshark" לא הצליח לתפוס אותה. כמו כן, ניתן לראות כי checksum = [unverified] המשמעות של הדבר כי הפרוטוקול לא יכול לאמת את קבלת החטילה במלואה. עשויה מה שקרה זה שפרוטוקול ה-"TCP" ידרוש לשלוח את החטילה מחדש, נראה בתמונה הבאה:

```
| 11 0.282659676 127.0.0.1      127.0.0.1      TCP    78 [TCP Dup ACK 7#1] 8395 .. 53016 [ACK] Seq=4 Ack=20 Win=65536 Len=0 Tsvl=1828864363 Tscr=1828864363..  
| 12 0.282673336 127.0.0.1      127.0.0.1      TCP    328... [TCP Retransmission] 53016 .. 8395 [ACK] Seq=20 Ack=4 Win=65536 Len=32768 Tsvl=1828864363 Tscr=18...  
| 13 0.282685524 127.0.0.1      127.0.0.1      TCP    66 [TCP ZeroWindow] 8395 .. 53016 [ACK] Seq=4 Ack=65556 Win=0 Len=0 Tsvl=1828864363 Tscr=1828864363  
| 14 0.282725755 127.0.0.1      127.0.0.1      TCP    66 [TCP Window Update] 8395 .. 53016 [ACK] Seq=4 Ack=65556 Win=48512 Len=0 Tsvl=1828864363 Tscr=1828...  
| 15 0.282736032 127.0.0.1      127.0.0.1      TCP    328... 53016 .. 8395 [ACK] Seq=65556 Ack=4 Win=65536 Len=32768 Tsvl=1828864363 Tscr=1828864363  
| 16 0.2827353564 127.0.0.1     127.0.0.1      TCP    66 8395 .. 53016 [ACK] Seq=4 Ack=98324 Win=65536 Len=0 Tsvl=1828864363 Tscr=1828864363  
  
Acknowledgment Number: 20 (relative ack number)  
Acknowledge number (raw): 2484109310  
1011 .... = Header Length: 44 bytes (11)  
Flags: 0x010 (ACK)  
Window: 512  
[Calculated window size: 65536]  
[Window size scaling factor: 128]  
Checksum: 0xfe34 [unverified]  
[Checksum Status: Unverified]  
Urgent Pointer: 0  
Options: (24 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps, No-Operation (NOP), No-Operation (NOP), SACK  
[Timestamps]  
[SEQ/ACK analysis]  
[iRTT: 0.000036467 seconds]  
- [TCP Analysis Flags]  
- [Duplicate ACK #: 1]  
- [Duplicate to the ACK in frame: ?]  
  [Expert Info (Note/Sequence): Duplicate ACK (#1)]  
  [Duplicate ACK (#1)]  
  [Severity level: Note]  
  [Group: Sequence]  
  
0020  00 01 20 cb cf 18 e7 fa b2 5a 94 10 7f fe b0 10  ... .Z. ....  
0030  02 00 fe 34 00 00 01 01 08 0a 6d 02 41 6b 6d 02  ... 4. .... m Akm.  
0040  41 5c 01 01 05 0a 94 10 ff fe 94 11 7f fe A\.....
```

כאן הוא אומר שהוא פפס חבילה "TPC Dup ACK" וכעת הוא אומר שיש לחזור ל- *ACK* המסומן ולשלוח אותה מחדש, נראה:

11 0.282659676 127.0.0.1	127.0.0.1	TCP	78 [TCP Dup ACK 7#1] 8395 → 53016 [ACK] Seq=4 Ack=20 Win=65536 Len=0 TSval=1828864363 TSeср=182886434...
12 0.282673336 127.0.0.1	127.0.0.1	TCP	328... [TCP Retransmission] 53016 → 8395 [ACK] Seq=20 Ack=4 Win=65536 Len=32768 TSval=1828864363 TSeср=182...
13 0.282685524 127.0.0.1	127.0.0.1	TCP	66 [TCP ZeroWindow] 8395 → 53016 [ACK] Seq=4 Ack=65556 Win=0 Len=0 TSval=1828864363 TSeср=1828864363
14 0.282725755 127.0.0.1	127.0.0.1	TCP	66 [TCP Window Update] 8395 → 53016 [ACK] Seq=4 Ack=65556 Win=48512 Len=0 TSval=1828864363 TSeср=1828...
15 0.282736032 127.0.0.1	127.0.0.1	TCP	328... 53016 → 8395 [ACK] Seq=65556 Ack=4 Win=65536 Len=32768 TSval=1828864363 TSeср=1828864363
16 0.282753564 127.0.0.1	127.0.0.1	TCP	66 8395 → 53016 [ACK] Seq=4 Ack=98324 Win=65536 Len=0 TSval=1828864363 TSeср=1828864363
Acknowledgment Number: 20 (relative ack number)			
Acknowledgment number (raw): 2484109310			
1011 = Header Length: 44 bytes (11)			
Flags: 0x010 (ACK)			
Window: 512			
[Calculated window size: 65536]			
[Window size scaling factor: 128]			
Checksum: 0xfe34 [unverified]			
[Checksum Status: Unverified]			
Urgent Pointer: 0			
Options: (24 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps, No-Operation (NOP), No-Operation (NOP), SACK			
[Timestamps]			
[SEQ/ACK analysis]			
[RTT: 0.000036467 seconds]			
+ [TCP Analysis Flags]			
+ [Duplicate ACK #: 1]			
- [Duplicate to the ACK in frame: 7]			
- [Expert Info (Note/Sequence): Duplicate ACK (#1)]			
- [Duplicate ACK (#1)]			
- [Severity level: Note]			
- [Group: Sequence]			
0020 00 01 20 cb cf 18 e7 fa b2 5a b4 10 7f fe b0 10 Z			
0030 02 00 fe 34 00 00 01 01 08 0a 6d 02 41 6b 6d 02 ... 4 m Akm			
0040 41 5c 01 01 05 0a 94 10 ff fe 94 11 7f fe A\			

כאן הпротокול מפס את חלון הקבלה של החבילות, זו הпрוטוקול מצמצם את התגובה הנוכחית ויצת בכלל איבוד החבילות.

13 0.282685524 127.0.0.1	127.0.0.1	TCP	66 [TCP ZeroWindow] 8395 → 53016 [ACK] Seq=4 Ack=65556 Win=0 Len=0 TSval=1828864363 TSeср=1828864363
14 0.282725755 127.0.0.1	127.0.0.1	TCP	66 [TCP Window Update] 8395 → 53016 [ACK] Seq=4 Ack=65556 Win=48512 Len=0 TSval=1828864363 TSeср=1828...
15 0.282736032 127.0.0.1	127.0.0.1	TCP	328... 53016 → 8395 [ACK] Seq=65556 Ack=4 Win=65536 Len=32768 TSval=1828864363 TSeср=1828864363
16 0.282753564 127.0.0.1	127.0.0.1	TCP	66 8395 → 53016 [ACK] Seq=4 Ack=98324 Win=65536 Len=0 TSval=1828864363 TSeср=1828864363
Acknowledgment Number: 65556 (relative ack number)			
Acknowledgment number (raw): 2484174846			
1000 = Header Length: 32 bytes (8)			
Flags: 0x010 (ACK)			
Window: 0			
[Calculated window size: 0]			
[Window size scaling factor: 128]			
Checksum: 0xfe28 [unverified]			
[Checksum Status: Unverified]			
Urgent Pointer: 0			
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps			
[Timestamps]			
[SEQ/ACK analysis]			
[This is an ACK to the segment in frame: 10]			
[The RTT to ACK the segment was: 0.000050457 seconds]			
[RTT: 0.000036467 seconds]			
+ [TCP Analysis Flags]			
- [Expert Info (Warning/Sequence): TCP Zero Window segment]			
- [TCP Zero Window segment]			
- [Severity level: Warning]			
- [Group: Sequence]			
0020 00 01 20 cb cf 18 e7 fa b2 5a 94 11 7f fe 80 10 Z			
0030 00 00 fe 28 00 00 01 01 08 0a 6d 02 41 6b 6d 02 ... (. . . . m Akm			
0040 41 6b			

כאן הпротокол מעדכן מחדש התעבורה ומרחיב אותו כדי שוב על מנת לקבל ולהעביר חבילות.

14 0.282725755 127.0.0.1	127.0.0.1	TCP	66 [TCP Window Update] 8395 → 53016 [ACK] Seq=4 Ack=65556 Win=49512 Len=0 TSval=1828864363 TSecr=1828864363
15 0.282736032 127.0.0.1	127.0.0.1	TCP	328... 53016 → 8395 [ACK] Seq=65556 Ack=4 Win=65536 Len=32768 TSval=1828864363 TSecr=1828864363
16 0.282753564 127.0.0.1	127.0.0.1	TCP	66 8395 → 53016 [ACK] Seq=4 Ack=98324 Win=65536 Len=0 TSval=1828864363 TSecr=1828864363
Sequence Number (raw): 3891966554			
[Next Sequence Number: 4 (relative sequence number)]			
Acknowledgment Number: 65556 (relative ack number)			
Acknowledgment number (raw): 2484174846			
1000 = Header Length: 32 bytes (8)			
Flags: 0x010 (ACK)			
Window: 379			
[Calculated window size: 48512]			
[Window size scaling factor: 128]			
Checksum: 0xfe28 [unverified]			
[Checksum Status: Unverified]			
Urgent Pointer: 0			
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps			
[Timestamps]			
[SEQ/ACK analysis]			
- [iRTT: 0.000036467 seconds]			
- [TCP Analysis Flags]			
- [Expert Info (Chat/Sequence): TCP window update]			
[TCP window update]			
[Severity level: Chat]			
[Group: Sequence]			
0020 00 01 20 cb cf 18 e7 fa b2 5a 94 11 7f fe 80 10 Z			
0030 01 7b fe 28 00 00 01 01 08 0a 6d 62 41 6b 6d 02 .{ m Akm			
0040 41 6b Ak			

2.3.2 מספר חבילות שנשלחו

שלחו בסך הכל: 170 חבילות.

104 0.359230000 127.0.0.1	127.0.0.1	TCP	655... [TCP Retransmission] 53016 → 8395 [ACK] Seq=5408468 Ack=101 Win=65536 Len=65483 TSval=1828870447 TSecr=1828870447
165 6.366693345 127.0.0.1	127.0.0.1	TCP	655... 53016 → 8395 [ACK] Seq=101 Ack=5500092 Win=3098752 Len=0 TSval=1828870498 TSecr=1828870447
166 6.417865295 127.0.0.1	127.0.0.1	TCP	66 8395 → 53016 [ACK] Seq=101 Ack=5500092 Win=3098752 Len=0 TSval=1828870498 TSecr=1828870447
167 7.123544622 127.0.0.1	127.0.0.1	TCP	71 53016 → 8395 [PSH, ACK] Seq=5500092 Ack=101 Win=65536 Len=5 TSval=1828871204 TSecr=1828870498
168 7.123579548 127.0.0.1	127.0.0.1	TCP	66 53016 → 8395 [FIN, ACK] Seq=5500097 Ack=101 Win=65536 Len=0 TSval=1828871204 TSecr=1828870498
169 7.123713234 127.0.0.1	127.0.0.1	TCP	66 8395 → 53016 [FIN, ACK] Seq=101 Ack=5500098 Win=3145728 Len=0 TSval=1828871204 TSecr=1828871204
170 7.123723151 127.0.0.1	127.0.0.1	TCP	66 53016 → 8395 [ACK] Seq=5500098 Ack=102 Win=65536 Len=0 TSval=1828871204 TSecr=1828871204

2.3.3 תוצאות

פלט התוכנית בסוף הרצה:

```

authentication has been sent!
CC algorithm has been changed to: "reno"
second message has been received!
CC algorithm has been changed to: "cubic"
first message has been received!
authentication has been sent!
CC algorithm has been changed to: "reno"
second message has been received!
CC algorithm has been changed to: "cubic"
exit message has been received!

#####
collected DATASET is:

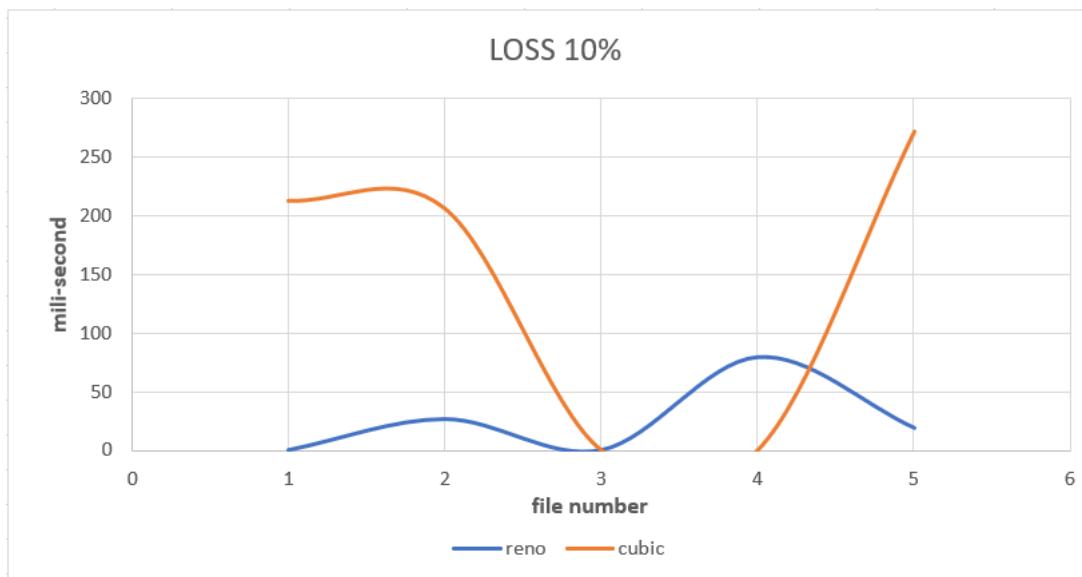
time to receive part 1 of file 1 is 213.524000 [ms]
time to receive part 2 of file 1 is 0.281000 [ms]
time to receive part 1 of file 2 is 206.853000 [ms]
time to receive part 2 of file 2 is 26.812000 [ms]
time to receive part 1 of file 3 is 0.376000 [ms]
time to receive part 2 of file 3 is 0.312000 [ms]
time to receive part 1 of file 4 is 0.564000 [ms]
time to receive part 2 of file 4 is 79.823000 [ms]
time to receive part 1 of file 5 is 272.538000 [ms]
time to receive part 2 of file 5 is 19.216000 [ms]

#####
the times of avg part:
the average of sending first file - by "cubic" - is: 138.771 [ms]
the average of sending second file - by "reno" - is: 25.289 [ms]

#####
y
CC algorithm has been changed to: "cubic"
first half has been sent!
waiting for authentication...
authentication match!
CC algorithm has been changed to: "reno"
second half has been sent!
send file again? (y/n)
y
CC algorithm has been changed to: "cubic"
first half has been sent!
waiting for authentication...
authentication match!
CC algorithm has been changed to: "reno"
second half has been sent!
send file again? (y/n)
y
CC algorithm has been changed to: "cubic"
first half has been sent!
waiting for authentication...
authentication match!
CC algorithm has been changed to: "reno"
second half has been sent!
send file again? (y/n)
y
CC algorithm has been changed to: "cubic"
first half has been sent!
waiting for authentication...
authentication match!
CC algorithm has been changed to: "reno"
second half has been sent!
send file again? (y/n)
n
exit message has been sent
yoadyoad-VirtualBox:~/Desktop/test$ 
```

סיכום התוצאות בטבלה ובגרף הבאים להלן:

אלגוריתם/מספר-קובץ	CUBIC	RENO
1	213.524[MS]	0.281[MS]
2	206.853[MS]	26.812[MS]
3	0.376[MS]	0.312[MS]
4	0.564[MS]	79.823[MS]
5	272.538[MS]	19.216[MS]



המינים המומוצעים:

1. אלגוריתם cubic : $138.771[ms]$
2. אלגוריתם reno : $25.289[ms]$

2.3.4 מסקנות

נשים לב כי חלק מהזמינים עלו משמעותית, וזאת כיוון שהפעלו איבוד חבילות ולכן בכל פעם שקרה איבוד הפרוטוקול דרש שהחבילות ישלחו שוב ושוב עד שיגיעו בשלוםונן, דבר אשר פגע בזמןי הקבלה (אך זה לא בהכרח משחו רע אם בסוף כל המידע מגיע שלם - אמינות). כמו כן, גם בניסוי זה אלגוריתם *reno* הוא המהיר יותר.

2.4 ניסוי 3 - 15% איבוד חבילות

עכשו ברכינו שוב לשנות את התעבורה, נשתמש בפקודה הבאה:

”`sudo tc qdisc del dev lo root netem 15%`”

2.4.1 שליחת החבילות

כעת, נשים לב לתופעה חדשה עם החבילות הנשלחות:

```

28 0.763472204 127.0.0.1      127.0.0.1      TCP      328... [TCP Out-Of-Order] 52684 - 8395 [PSH, ACK] Seq=163860 Ack=4 Win=65536 Len=32768 TSval=1829662911 TSecr=1829662911
29 0.763478037 127.0.0.1      127.0.0.1      TCP      78 [TCP Window Update] 8395 - 52684 [ACK] Seq=4 Ack=163860 Win=458496 Len=0 TSval=1829662911 TSecr=1829662911
30 0.763487886 127.0.0.1      127.0.0.1      TCP      66 8395 - 52684 [ACK] Seq=4 Ack=294932 Win=729384 Len=0 TSval=1829662911 TSecr=1829662911
31 0.763496588 127.0.0.1      127.0.0.1      TCP      328.. 52684 - 8395 [PSH, ACK] Seq=294932 Ack=4 Win=65536 Len=32768 TSval=1829662911 TSecr=1829662911
32 0.763501838 127.0.0.1      127.0.0.1      TCP      328.. 52684 - 8395 [ACK] Seq=327700 Ack=4 Win=65536 Len=32768 TSval=1829662911 TSecr=1829662911
33 0.763507168 127.0.0.1      127.0.0.1      TCP      328.. 52684 - 8395 [PSH, ACK] Seq=360468 Ack=4 Win=65536 Len=32768 TSval=1829662911 TSecr=1829662911
34 0.763518349 127.0.0.1      127.0.0.1      TCP      66 8395 - 52684 [ACK] Seq=4 Ack=360468 Win=982272 Len=0 TSval=1829662911 TSecr=1829662911
35 0.763527122 127.0.0.1      127.0.0.1      TCP      328.. 52684 - 8395 [ACK] Seq=393236 Ack=4 Win=65536 Len=32768 TSval=1829662911 TSecr=1829662911
Acknowledgment number (raw): 56276414
1000 .... = Header Length: 32 bytes (8)
> Flags: 0x018 (PSH, ACK)
Window: 512
[Calculated window size: 65536]
[Window size scaling factor: 128]
Checksum: 0xe7e29 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
[Timestamps]
[SEQ/ACK analysis]
  [RTT: 0.000031715 seconds]
  [Bytes in flight: 98364]
  [Bytes sent since last PSH flag: 65536]
- [TCP Analysis Flags]
  - [Expert Info (Warning/Sequence): This frame is a (suspected) out-of-order segment]
    [This frame is a (suspected) out-of-order segment]
    [Severity level: Warning]
    [Group: Sequence]
  TCP payload (32768 bytes)

0030 02 00 7e 29 00 00 00 01 01 08 0a 6d 0e 70 bf 6d 0e  ..-~. .: .m.p.m.
0040 70 bf 37 33 37 39 36 33 32 35 35 34 33 33 36 35 p-737963 25543365
0050 34 36 33 32 37 34 37 35 35 33 38 33 33 38 39 38 46327475 53833898

```

החבילה מסומנת כ-”*TCP Out Of Order*”, המשמעות היא שאם לדוגמה אנו שולחים מספר חבילות רצוף והשרת מזכה לקבל אותן לפי הסדר אז בשגיאה זו אחת מהחבילות הגיעה לא בסדר שלו (ז”א אחת מהחבילות שלחנו הגעה לפני שהגיע סדר השליחה שלה). גם במקרה זה ה프וטוקול מטתקש על אמינות התעבורה ושולח מחדש את החבילות שלא נשלחו.

לאחר מכן הפרוטוקול שוב מעדכן את חלון התעבורה:

```

29 0.763478037 127.0.0.1      127.0.0.1      TCP      78 [TCP Window Update] 8395 - 52684 [ACK] Seq=4 Ack=163860 Win=458496 Len=0 TSval=1829662911 TSecr=1829662911
30 0.763496588 127.0.0.1      127.0.0.1      TCP      66 8395 - 52684 [ACK] Seq=4 Ack=294932 Win=729384 Len=0 TSval=1829662911 TSecr=1829662911
31 0.763507168 127.0.0.1      127.0.0.1      TCP      328.. 52684 - 8395 [PSH, ACK] Seq=294932 Ack=4 Win=65536 Len=32768 TSval=1829662911 TSecr=1829662911
32 0.763518349 127.0.0.1      127.0.0.1      TCP      328.. 52684 - 8395 [ACK] Seq=327700 Ack=4 Win=65536 Len=32768 TSval=1829662911 TSecr=1829662911
33 0.763527122 127.0.0.1      127.0.0.1      TCP      66 8395 - 52684 [ACK] Seq=4 Ack=360468 Win=982272 Len=0 TSval=1829662911 TSecr=1829662911
34 0.763537000 127.0.0.1      127.0.0.1      TCP      328.. 52684 - 8395 [ACK] Seq=393236 Ack=4 Win=65536 Len=32768 TSval=1829662911 TSecr=1829662911
Sequence Number (raw): 56276414
[Next Sequence Number: 4 (relative sequence number)]
Acknowledgment Number: 163860 (relative ack number)
Acknowledgment number (raw): 1928093627
1011 .... = Header Length: 44 bytes (11)
> Flags: 0x010 (ACK)
Window: 3582
[Calculated window size: 458496]
[Window size scaling factor: 128]
Checksum: 0xfe34 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
Options: (24 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps, No-Operation (NOP), No-Operation (NOP), SACK
[Timestamps]
[SEQ/ACK analysis]
  [RTT: 0.000031715 seconds]
- [TCP Analysis Flags]
  - [Expert Info (Chat/Sequence): TCP window update]
    [TCP window update]
    [Severity level: Chat]
    [Group: Sequence]

0029 00 01 20 cb cd cc 03 5a b5 be 72 ec 5f bb b0 10 ..-....Z ..r. ....
0030 0d fe fe 34 00 00 01 01 08 0a 6d 0e 70 bf 6d 0e ..-4.... .m.p.m.
0040 70 bf 01 01 05 04 72 ec df bb 72 ed df bb p.....r. ....

```

כעת שוב אנחנו נתקלים באיבוד של סגמנטים של חבילות:

```

65 5.00291794 127.0.0.1 127.0.0.1 TCP 655.. [TCP Previous segment not captured] 52684 - 8395 [ACK] Seq=1296484 Ack=24 Win=6536 Len=65483 TSval=1829667151 TSecl=1829667151
66 5.002923592 127.0.0.1 127.0.0.1 TCP 78 [TCP Window Update] 8395 - 52684 [ACK] Seq=24 Ack=1231001 Win=3112448 Len=0 TSval=1829667151 TSecl=1829667151
67 5.002976788 127.0.0.1 127.0.0.1 TCP 655.. 52684 - 8395 [ACK] Seq=1361967 Ack=24 Win=65536 Len=65483 TSval=1829667151 TSecl=1829667151
68 5.002991283 127.0.0.1 127.0.0.1 TCP 78 [TCP Dup ACK 64#1] 8395 - 52684 [ACK] Seq=24 Ack=1231001 Win=3112448 Len=0 TSval=1829667151 TSecl=1829667151
69 6.7606626281 127.0.0.1 127.0.0.1 TCP 655.. [TCP Retransmission] 52684 - 8395 [ACK] Seq=1231001 Ack=24 Win=65536 Len=65483 TSval=1829668988 TSecl=1829668988
70 6.760681807 127.0.0.1 127.0.0.1 TCP 66 8395 - 52684 [ACK] Seq=24 Ack=1427456 Win=3046272 Len=0 TSval=1829668989 TSecl=1829668989
71 6.760681972 127.0.0.1 127.0.0.1 TCP 655.. 52684 - 8395 [ACK] Seq=1427456 Ack=24 Win=65536 Len=65483 TSval=1829668999 TSecl=1829668999
1000 .... = Header Length: 32 bytes (8)
> Flags: 0x010 (ACK)
Window: 512
[Calculated window size: 65536]
[Window size scaling factor: 128]
Checksum: 0xffff4 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
> Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
[timestamps]
-> [SEQ/ACK analysis]
  [iRTT: 0.000031715 seconds]
  -> [TCP Analysis Flags]
    -> [Expert Info (Warning/Sequence): Previous segment(s) not captured (common at capture start)]
      [Previous segment(s) not captured (common at capture start)]
      [Severity level: Warning]
      [Group: Sequence]
    TCP payload (65483 bytes)
-> Data (65483 bytes)
  Data: 373333363839343938343332333638393432343439363839335393537383934363737...
  [Length: 65483]

```

וכrangle גם בהזה הפרטוקול מטפל על ידי שליחת *ACK* המציין את מספר הרץ' הרצוי. כאן ישנה שוב בעיה בקבלת החבילות. (שתי בעיות אחת אחרי השניה), החבילות לא הגיעו שלמות ואבדו סגמנטניים ולאחר מכן הוא מטפל בבעיות על ידי בקשה מחדש.

```

183.19.2195874.127.0.0.1 127.0.0.1 TCP 655... [TCP Previous segment not captured] 52684 -> 8395 [ACK] Seq=5146536 Ack=101 Win=65536 Len=65483 TSv...
184.19.2196012.127.0.0.1 127.0.0.1 TCP 78 [TCP dup ACK 182#1] 8395 -> 52684 [ACK] Seq=101 Ack=5081053 Win=311248 Len=0 TSval=1829681367 TSec...
185.19.2196192.127.0.0.1 127.0.0.1 TCP 655... [TCP Previous segment not captured] 52684 -> 8395 [ACK] Seq=5277502 Ack=101 Win=65536 Len=65483 TS...
186.19.2196286.127.0.0.1 127.0.0.1 TCP 86 [TCP dup ACK 182#2] 8395 -> 52684 [ACK] Seq=101 Ack=5081053 Win=311248 Len=0 TSval=1829681368 TSec...
187.19.2196414.127.0.0.1 127.0.0.1 TCP 655... [TCP Fast Retransmission] 52684 -> 8395 [ACK] Seq=5081053 Ack=101 Win=65536 Len=65483 TSval=1829681368 SLE=527...
188.19.2196580.127.0.0.1 127.0.0.1 TCP 78 8395 -> 52684 [ACK] Seq=101 Ack=5212019 Win=3045632 Len=0 TSval=1829681368 TSecr=1829681368 SLE=527...
189.19.2196728.127.0.0.1 127.0.0.1 TCP 655... [TCP Retransmission] 52684 -> 8395 [ACK] Seq=5212019 Ack=101 Win=65536 Len=65483 TSval=1829681368 TSecr=1829681368 SLE=527...

Acknowledgment number (raw): 56276511
1000... = Header length: 32 bytes (8)
> Flags: 0x010 (ACK)
Window: 512
[Calculated window size: 65536]
[Window size scaling factor: 128]
Checksum: 0xffff4 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
> Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
> [Timestamps]
-> [SEQ/ACK analysis]
  [RTT: 0.0000031715 seconds]
  + [TCP Analysis Flags]
    - [Expert Info (Warning/Sequence): Previous segment(s) not captured (common at capture start)]
      [Previous segment(s) not captured (common at capture start)]
      [Severity level: Warning]
      [Group: Sequence]
    TCP payload (65483 bytes)
-> Data: (65483 bytes)
  Data: 33373639363235323833434373933343432383638373734333436333832373436343332...
  Length: 65483

00000030 02 06 fd ff 00 00 01 01 08 0a 6d 0e b8 d7 6d 0e .. . . . . . . . .
00000040 b8 d7 33 37 36 39 36 32 35 32 38 34 34 37 39 33 .. 376962 52844793
00000050 34 34 32 38 36 38 37 37 37 34 33 34 36 33 38 32 44286877 74346382

```

2.4.2 מס' חבילות שנשלחו

נשלחו בסך הכל: 200 חבילות.

2.4.3 תוצאות

פלט התוכנית בסוף הרצה:

```

CC algorithm has been changed to: "cubic"
first message has been received!
authentication has been sent!
CC algorithm has been changed to: "reno"
second message has been received!
CC algorithm has been changed to: "cubic"
first message has been received!
authentication has been sent!
CC algorithm has been changed to: "reno"
second message has been received!
CC algorithm has been changed to: "cubic"
exit message has been received!

#####
collected DATASET is:

time to receive part 1 of file 1 is 920.153000 [ms]
time to receive part 2 of file 1 is 463.924000 [ms]
time to receive part 1 of file 2 is 1758.306000 [ms]
time to receive part 2 of file 2 is 1110.862000 [ms]
time to receive part 1 of file 3 is 172.412000 [ms]
time to receive part 2 of file 3 is 83.571000 [ms]
time to receive part 1 of file 4 is 4456.977000 [ms]
time to receive part 2 of file 4 is 0.513000 [ms]
time to receive part 1 of file 5 is 15.182000 [ms]
time to receive part 2 of file 5 is 0.588000 [ms]

#####
the times of avg part:
the average of sending first file - by "cubic" - is: 1464.606 [ms]
the average of sending second file - by "reno" - is: 331.892 [ms]

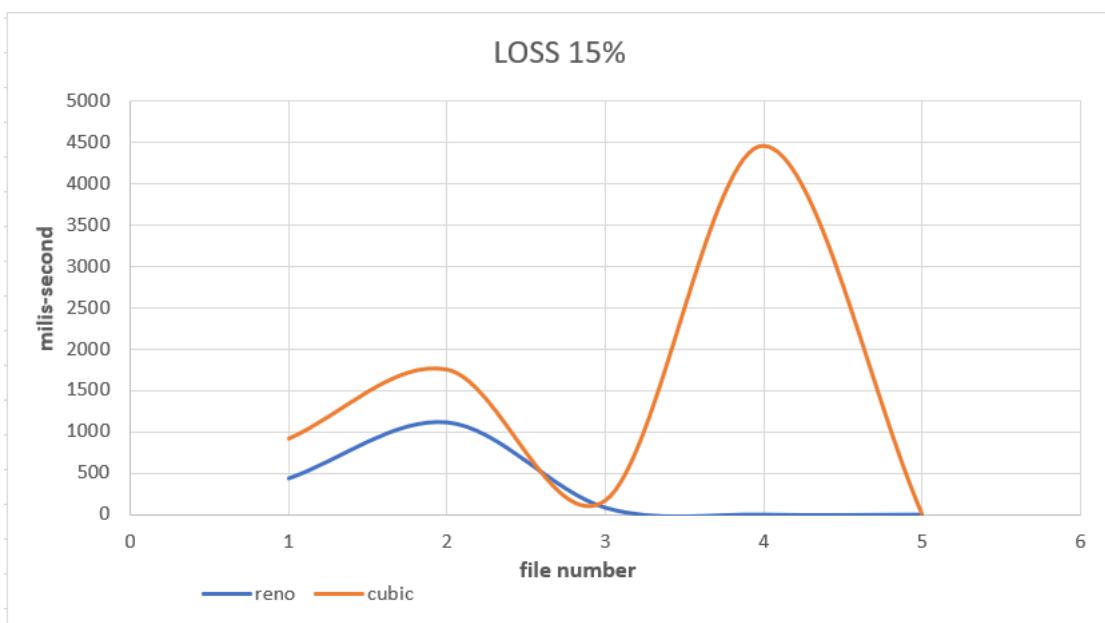
#####
yoad@yoad-VirtualBox:~/Desktop/test$ 
```

```

yoad@yoad-VirtualBox:~ $ sudo tc qdisc add dev lo root
yoad@yoad-VirtualBox:~ $ netem loss 15%
yoad@yoad-VirtualBox:~ $ 
```

סיכום התוצאות בטבלה ובגרף הבאים להלן:

אלגוריתם/מספר קובץ	CUBIC	RENO
1	920.153[MS]	463.924[MS]
2	1758.306[MS]	1110.862[MS]
3	172.412[MS]	83.571[MS]
4	4456.977[MS]	0.513[MS]
5	15.182[MS]	0.588[MS]



הזמןים המומוצעים:

1. אלגוריתם $1464.606[ms]$:*cubic*

2. אלגוריתם $331.892[ms]$:*reno*

2.4.4 מסקנות

שוב נקבל עלייה מאוד משמעותית בזמן הקבלה של החלקים. כמו שראינו כבר - אלגוריתם *reno* מנצת.

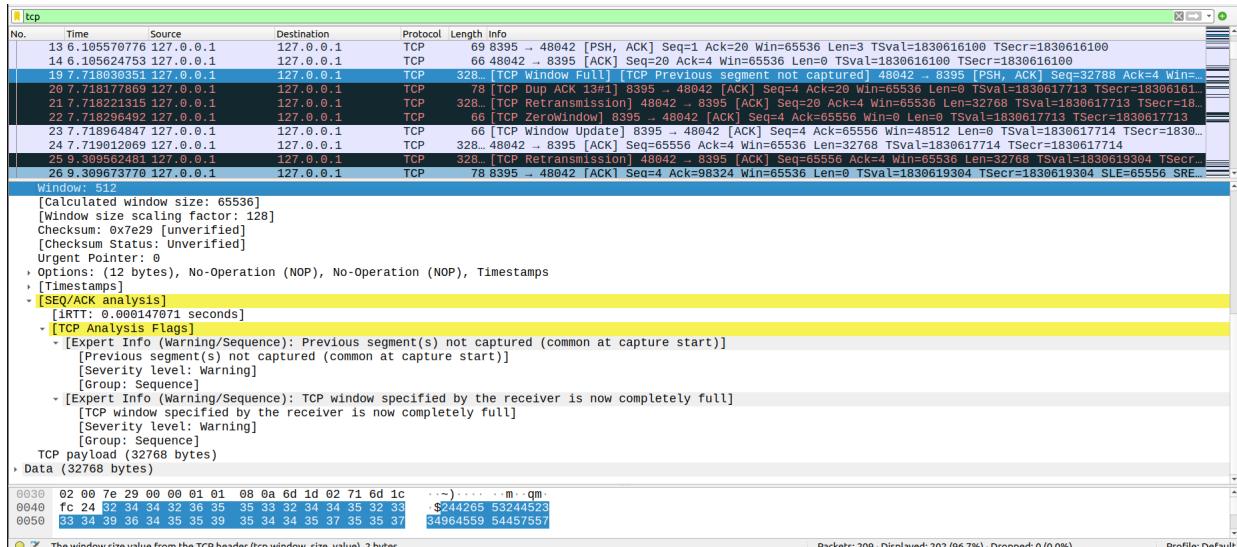
2.5 ניסוי 4 - 20% איבוד חבילות

כעת עלינו שוב לשנות את אחוזי איבוד חבילות בתעבורת הרשת. נשתמש בפקודה:

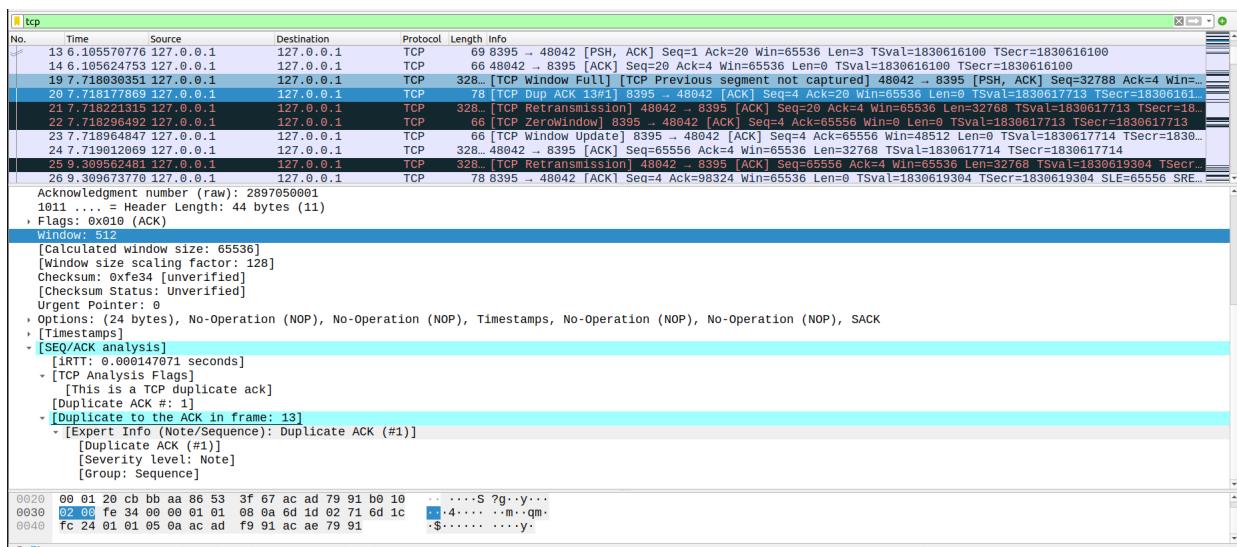
"sudo tc del dev lo root netem loss 20%"

2.5.1 שליחת חבילות

בניסויים הקודמים ראיינו המונ שגיאות תעבורת מסווגים שונים, הפעם גם נציג אותן ונסביר בקצרה:
 כאן שוב חלון התעבורת מותמאל "TCP Window Full"
 כמו כן גם מופפסים סגמנטיים של חבילות "TCP Previous Segment Not Captured"



לאחר מכן, שוב מגיעות חבילות כפולות "TCP Dup ACK"



בנוסף, הוא שוב חוזר לדורש לקבלת תבילות תקינות "TCP Retransmission"

וכעת הוא שוב מאפס את חלון התעבורת "TCP ZeroWindow"

```

tcp
No. Time Source Destination Protocol Length Info
13 6.185570776 127.0.0.1 127.0.0.1 TCP 69 8395 > 48042 [PSH, ACK] Seq=1 Ack=20 Win=65536 Len=3 TSval=1830616100 TSecr=1830616100
14 6.185624753 127.0.0.1 127.0.0.1 TCP 66 48042 > 8395 [ACK] Seq=20 Ack=4 Win=65536 Len=0 TSval=1830616100 TSecr=1830616100
19? 7.718030351 127.0.0.1 127.0.0.1 TCP 328... [TCP Window Full] [TCP Previous segment not captured] 48042 > 8395 [PSH, ACK] Seq=32788 Ack=4 Wins=1 TSval=1830617713 TSecr=1830617713
20? 7.718177769 127.0.0.1 127.0.0.1 TCP 78 [TCP Dup ACK 13#1] 8395 > 48042 [ACK] Seq=4 Ack=20 Win=65536 Len=0 TSval=1830617713 TSecr=1830616101
21? 7.718221315 127.0.0.1 127.0.0.1 TCP 328... [TCP Retransmission] 48042 > 8395 [ACK] Seq=20 Ack=4 Win=65536 Len=32768 TSval=1830617713 TSecr=1830617713
22? 7.718296492 127.0.0.1 127.0.0.1 TCP 66 [TCP ZeroWindow] 8395 > 48042 [ACK] Seq=4 Ack=65556 Win=0 Len=0 TSval=1830617713 TSecr=1830617713
23? 7.7183964847 127.0.0.1 127.0.0.1 TCP 66 [TCP Window Update] 8395 > 48042 [ACK] Seq=4 Ack=65556 Win=48512 Len=0 TSval=1830617714 TSecr=1830617714
24? 7.719032069 127.0.0.1 127.0.0.1 TCP 328... 48042 > 8395 [ACK] Seq=65556 Ack=4 Win=65536 Len=32768 TSval=1830617714 TSecr=1830617714
25? 9.3089562481 127.0.0.1 127.0.0.1 TCP 328... [TCP Retransmission] 48042 > 8395 [ACK] Seq=65556 Ack=4 Win=65536 Len=32768 TSval=1830617714 TSecr=1830617714
26? 9.309967373/760 127.0.0.1 127.0.0.1 TCP 78 8395 > 48042 [ACK] Seq=4 Ack=98324 Win=65536 Len=0 TSval=1830619304 TSecr=1830619304 SLE=65556 SRE=65556
Acknowledgment Number: 65556 (relative ack number)
Acknowlegment number (raw): 2897115537
1000 .... = Header Length: 32 bytes (8)
Flags: 0x010 (ACK)
Window: 0
[Calculated window size: 0]
[Window size scaling factor: 128]
Checksum: 0xfe28 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
[Timestamps]
[SEQ/ACK analysis]
[This is an ACK to the segment in frame: 19]
[The RTT to ACK the segment was: 0.000266141 seconds]
[RTT: 0.000147071 seconds]
[TCP Analysis Flags]
- [Expert Info (Warning/Sequence): TCP Zero Window segment]
  [TCP Zero Window segment]
  [Severity level: Warning]
  [Group: Sequence]

0020 00 01 20 cb bb aa 86 53 3f 67 ac ae 79 91 80 10 .. . . . ?g..y...
0030 00 06 fe 28 00 00 01 01 08 0a 6d 0d 92 71 6d 1d .. . . . . . . . . . . . q
0040 02 71

```

2.5.2 מס' חבילות שנשלחו

נשלחו בסך הכל: 209 חבילות

205 34 .0.122.95...	127.0.0.1	127.0.0.1	TCP	70 8395 -> 4094 [SYN] Seq=101 Ack=350092 Win=134172 Len=0 TSVal=1830648708 TSerr=1830648708 SLE=350...
204 36 .88803170...	127.0.0.1	127.0.0.1	TCP	71 480402 -> 8395 [PSH, ACK] Seq=5500092 Ack=101 Win=65536 Len=5 TSVal=1830646875 TSerr=1830644807
205 36 .88803577...	127.0.0.1	127.0.0.1	TCP	66 8395 -> 48042 [ACK] Seq=101 Ack=5500097 Win=3145728 Len=0 TSVal=1830646875 TSerr=1830646875
206 36 .88804238...	127.0.0.1	127.0.0.1	TCP	66 480402 -> 8395 [FIN, ACK] Seq=5500097 Ack=101 Win=65536 Len=0 TSVal=1830646875 TSerr=1830646875
207 36 .88806124...	127.0.0.1	127.0.0.1	TCP	66 8395 -> 48042 [FIN, ACK] Seq=101 Ack=5500098 Win=3145728 Len=0 TSVal=1830646875 TSerr=1830646875
208 37 .0882444...	127.0.0.1	127.0.0.1	TCP	66 [TCP Retransmission] 8395 -> 48042 [FIN, ACK] Seq=101 Ack=5500098 Win=3145728 Len=0 TSVal=183064708...
209 37 .0882815...	127.0.0.1	127.0.0.1	TCP	66 480402 -> 8395 [ACK] Seq=5500098 Ack=102 Win=65536 Len=0 TSVal=1830646875 TSerr=1830646875

2.5.3 תוצאות

פלט התוכנית בסוף הריצה:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
CC algorithm has been changed to: "cubic"
first message has been received!
authentication has been sent!
CC algorithm has been changed to: "reno"
second message has been received!
CC algorithm has been changed to: "cubic"
first message has been received!
authentication has been sent!
CC algorithm has been changed to: "reno"
second message has been received!
CC algorithm has been changed to: "cubic"
exit message has been received!

#####
collected DATASET is:
time to receive part 1 of file 1 is 3204.856000 [ms]
time to receive part 2 of file 1 is 2.124000 [ms]
time to receive part 1 of file 2 is 2121.429000 [ms]
time to receive part 2 of file 2 is 3543.439000 [ms]
time to receive part 1 of file 3 is 418.397000 [ms]
time to receive part 2 of file 3 is 721.564000 [ms]
time to receive part 1 of file 4 is 48.780000 [ms]
time to receive part 2 of file 4 is 209.822000 [ms]
time to receive part 1 of file 5 is 271.262000 [ms]
time to receive part 2 of file 5 is 2.719000 [ms]

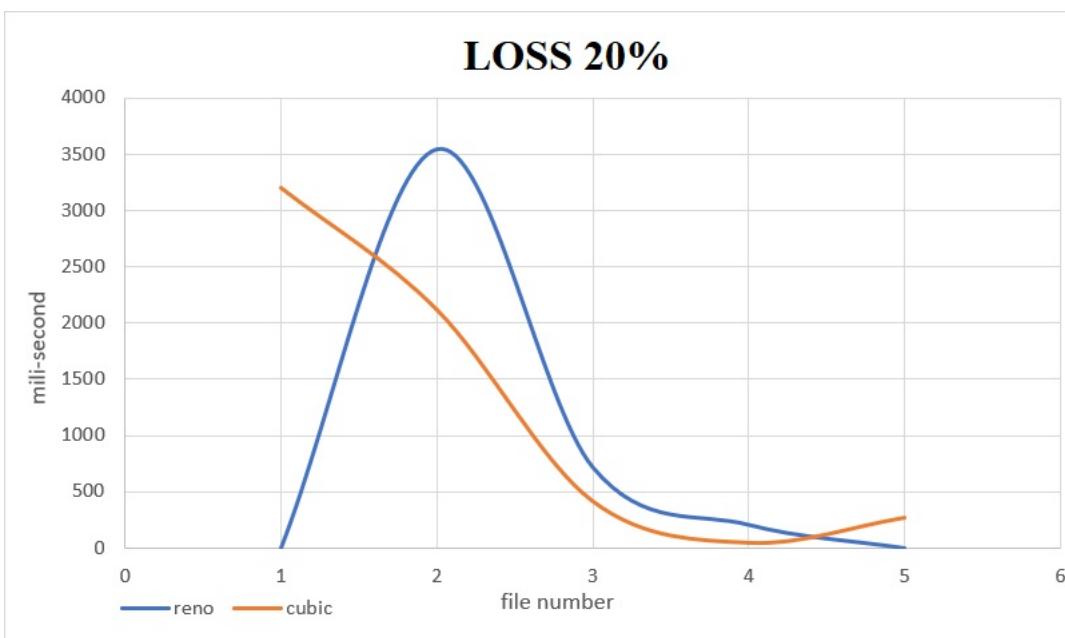
#####
the times of avg part:
the average of sending first file - by "cubic" - is: 1212.945 [ms]
the average of sending second file - by "reno" - is: 895.934 [ms]
#####

yoad@yoad-VirtualBox:~/Desktop/test$ 

```

סיכום התוצאות בטבלה ובגרף הבאים להלן:

אלגוריתם/מספר-קובץ	CUBIC	RENO
1	3204.856[MS]	2.124[MS]
2	2121.429[MS]	3543.439[MS]
3	418.397[MS]	721.564[MS]
4	48.78[MS]	209.822[MS]
5	271.262[MS]	2.719[MS]



הזמןים הממוצעים:

1. אלגוריתם **1212.945**[*ms*] :*cubic*

2. אלגוריתם **895.934**[*ms*] :*reno*

2.5.4 מסקנות

גם כאן אלגוריתם *reno* מנצח בזמנים.

2.6 מסקנות מקיפות

במהלך המהקר על תעבורת הרשת, הגיעו למספר מסקנות.

1. ככל שיש עלייה באיבוד החבילות בראשת, יש עלייה בזמן הקבלה - כיוון שפרוטוקול התקשרות אמין, לכן אם חבילות נאבדות הוא צריך לשולח אותן מחדש, דבר הגוזל זמן.
2. ככל שיש עלייה באיבוד החבילות, יש עלייה בכמות החבילות שנשלחו - מאוותה סיבה כמו 1 כיון שאם חבילות נאבדות הוא צריך לשולח מחדש ולאחר מכן עוד ועוד חבילות.
3. אלגוריתם בקרת העומס *reno* עדיף על *cubic* - ניתן לראות מרובעת הניסויים כי זמן הקבלה אצל *reno* נמוכים יותר מאשר אצל המתחילה (בזמנים הממוצעים).

3 ביבליוגרפיה

רשימת מקורות בהם השתמשנו בעת מימוש וכתיבת הפרויקט:

1. *Woodman, P.(2017) . Cubic TCP, Wikipedia*
"https://en.m.wikipedia.org/wiki/CUBIC_TCP"

2. *Frenchytheasia.(2005) TCP congestion control, Wikipedia*
"https://en.m.wikipedia.org/wiki/TCP_congestion_control"

3. *pintusaini.(2022) TCP Reno With Example, GeeksforGeeks*
"https://www.geeksforgeeks.org/tcp-reno-with-example/"

4. מצגות הקורס "תכנות מערכות 1" של ד"ר חגי אסף.

5. מצגות וחוברת הקורס "רשתות תקשורת".