***מטלה 3 – רשתות תקשורת***

***מגישים: אביב תורג'מן – 208007351***

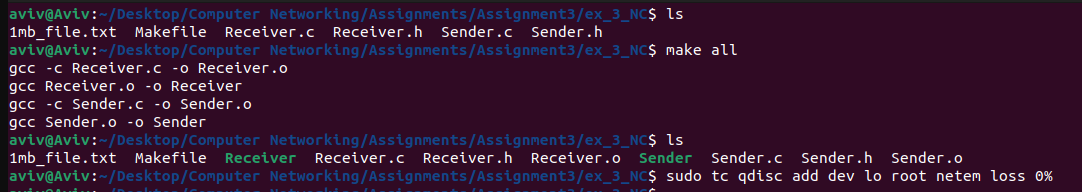
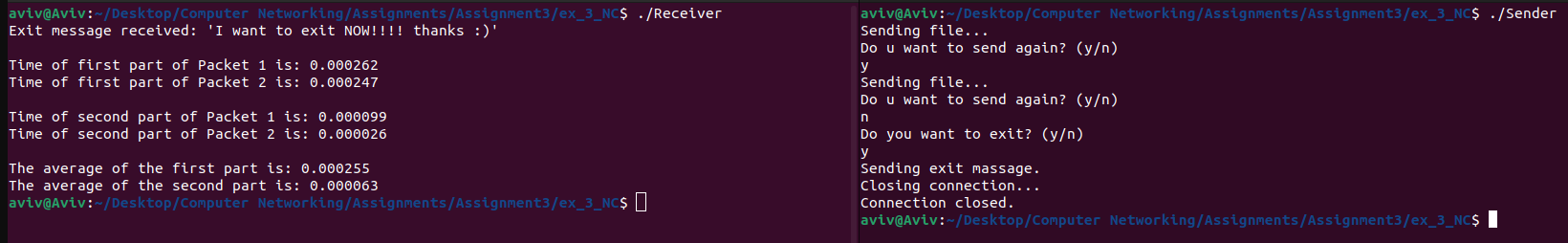
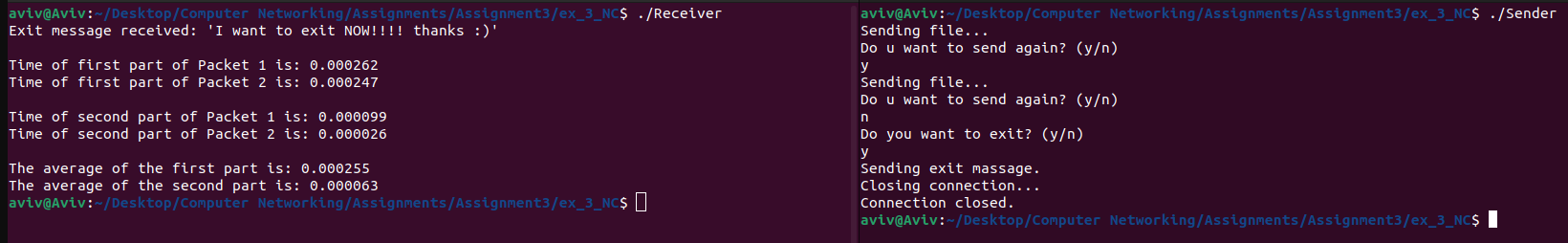
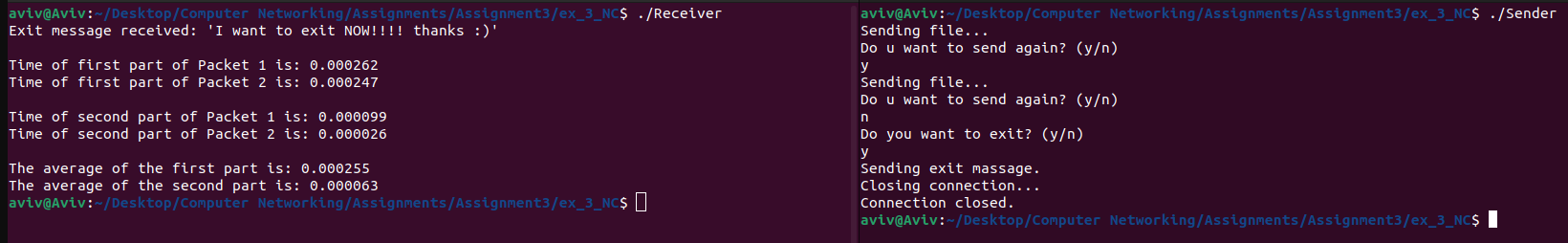
***אלון סויסה – 211344015***

***תוכן עניינים:***

הוראות קימפול והרצה.................................................................. 3-6  
  
מטרת התוכנה............................................................................. 7  
  
מבנה התוכנה.............................................................................. 7  
  
הסבר מפורט על הקוד................................................................... 8

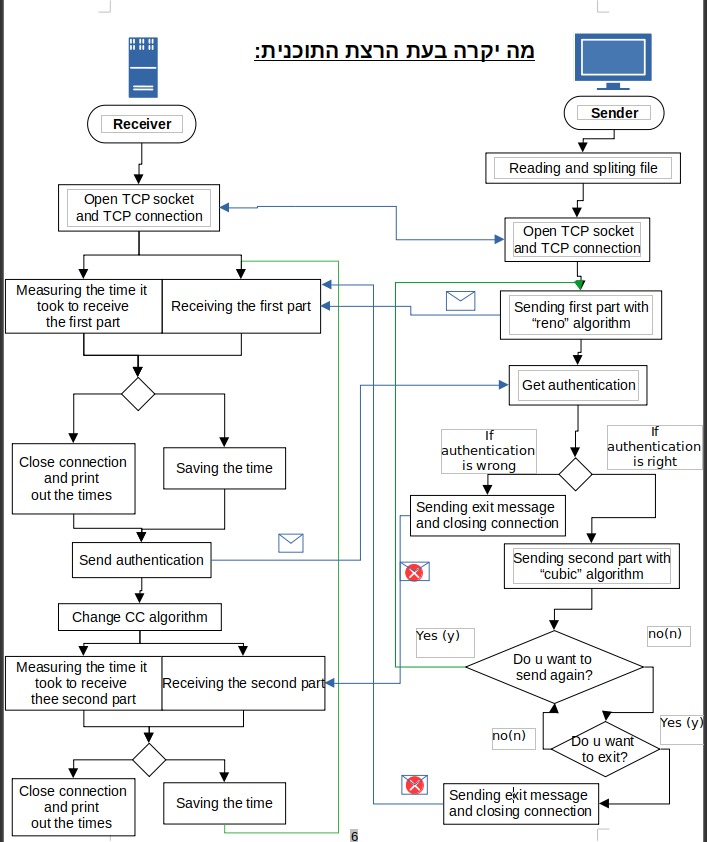
Sender........................................................................................ 8-13

**הוראות קימפול והרצה**

1. קימפול:
   1. תחילה נפתח את הטרמינל בתוך התיקייה של הקובץ. (נראה עם פקודת "ls" שאכן הקבצים הנ"ל נמצאים בתיקייה)
   2. נריץ את פקודת "make all" על מנת לקמפל את הקוד כדי שיהיה מוכן להרצה.
   3. נראה שוב עם פקודת "ls" שאכן נמצאים בתיקייה הקבצים "Receiver" ו- "Sender"
   4. נריץ את הפקודה:  
      "sudo tc qdisc add dev lo root netem loss 0%"  
      על מנת שבהרצה הראשונה יקבע אחוז האיבוד   
      ל-0%. על מנת לשנות זאת בהמשך נריץ את הפקודה:  
      "sudo tc qdisc change dev lo root netem XX%"
   5. לאחר שנסיים את כל ההרצות, נריץ את הפקודה:  
      "sudo tc qdisc del dev lo root netem"  
      על מנת להפסיק את איבוד הפאקטות.
2. הרצה:
   1. תחילה נפתח שני טרמינלים שונים בתיקיית הקובץ.
   2. נריץ **תחילה** את פקודת: "./Receiver"
   3. בטרמינל השני נריץ את פקודת: "./Sender" (**אחרי שהרצנו את Receiver)**
   4. אם נרצה לשלוח שוב, נזין את התו 'y' אחרת נזין את התו 'n'. (אם נזין תו אחר, נקבל הודעה: "Please enter (y/n)."
      1. 'y':
      2. 'n': נקבל הודעה "do you want to exit? (y/n)"

אם נזין 'y', הקשר ייסגר והתוכנית תפסיק לרוץ.  
אם נזין 'n', נחזור לשלב **d)**.

אחרת נקבל הודעה: "Please enter (y/n)."

****

Receiving the first part or an exit message

Got an exit message

Got the first part

Receiving the first part or an exit message

\*

**Please enter (y/n)**

Got an exit message

\*

Got the second part

**\***  if input is different than y/n/Y/N

**מטרת התוכנה:**

מטרת התוכנה היא לחקור איזה CC algorithm מהיר יותר בשליחת קובץ טקסט (של 0.5 מגה בייט) מבין שני האלגוריתמים "reno” ו- “cubic” , באמצעות TCP סוקט , תוך דימוי של איבוד מידע (סגמנטים) ברשת.

בכל שליחת קובץ שנבקש מהתוכנית לעשות ישלח חצי קובץ(0.5 מגה בייט) באלגוריתם "reno” וחצי באלגוריתם "cubic”.

התוכנית תמדוד כמה זמן לקח לכל חצי להגיע (עד כדי זמן של סגמנט אחד) ובסוף הריצה תדפיס את הזמנים שלקח לכל חצי בכל אחת מהשליחות ואת הזמן הממוצע של כל חצי.

**מבנה התוכנה:**

התוכנה מורכבת משישה קבצים:

* Makefile – קובץ שאחראי לעזור לנו לקמפל בקלות  
   הקבצים
* Sender.c – מימוש צד הלקוח
* Sender.h – בו כל ההצהרות על הפונקציות והקבועים

בהם הקובץ Sender.c משתמש

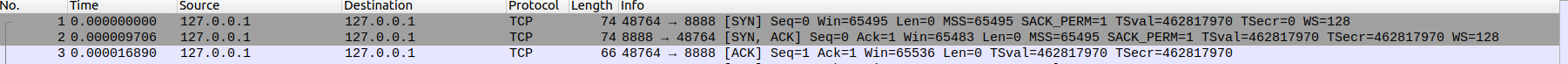
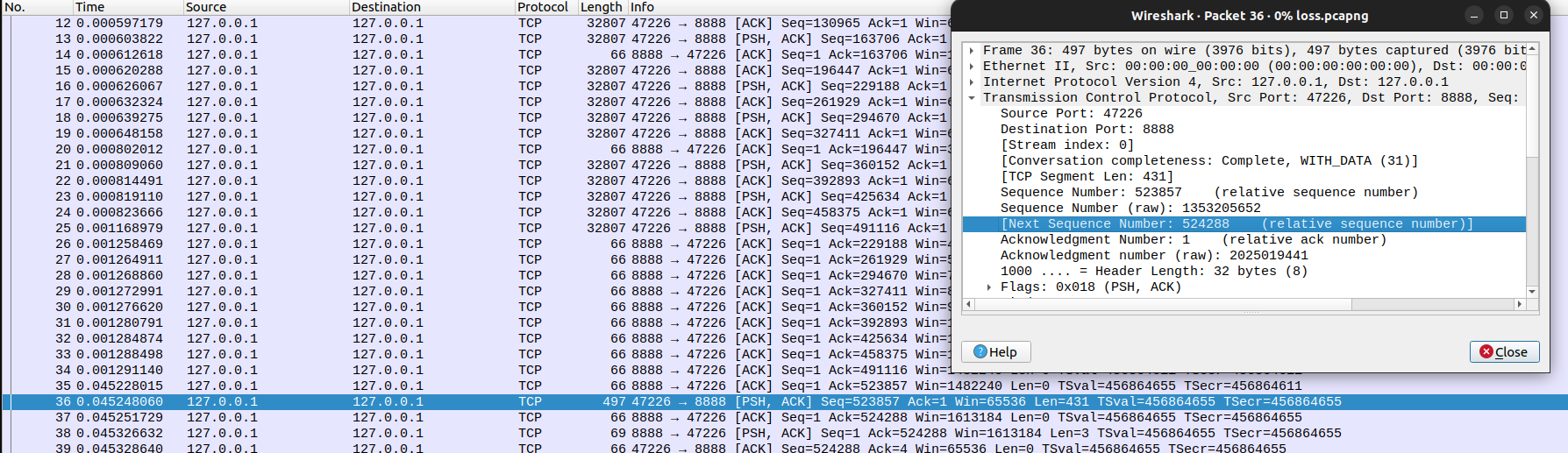
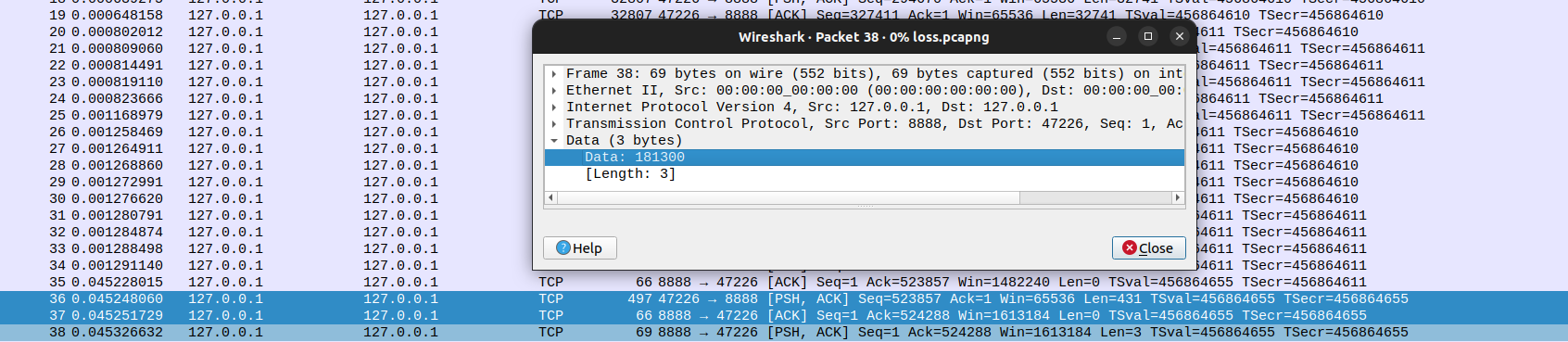
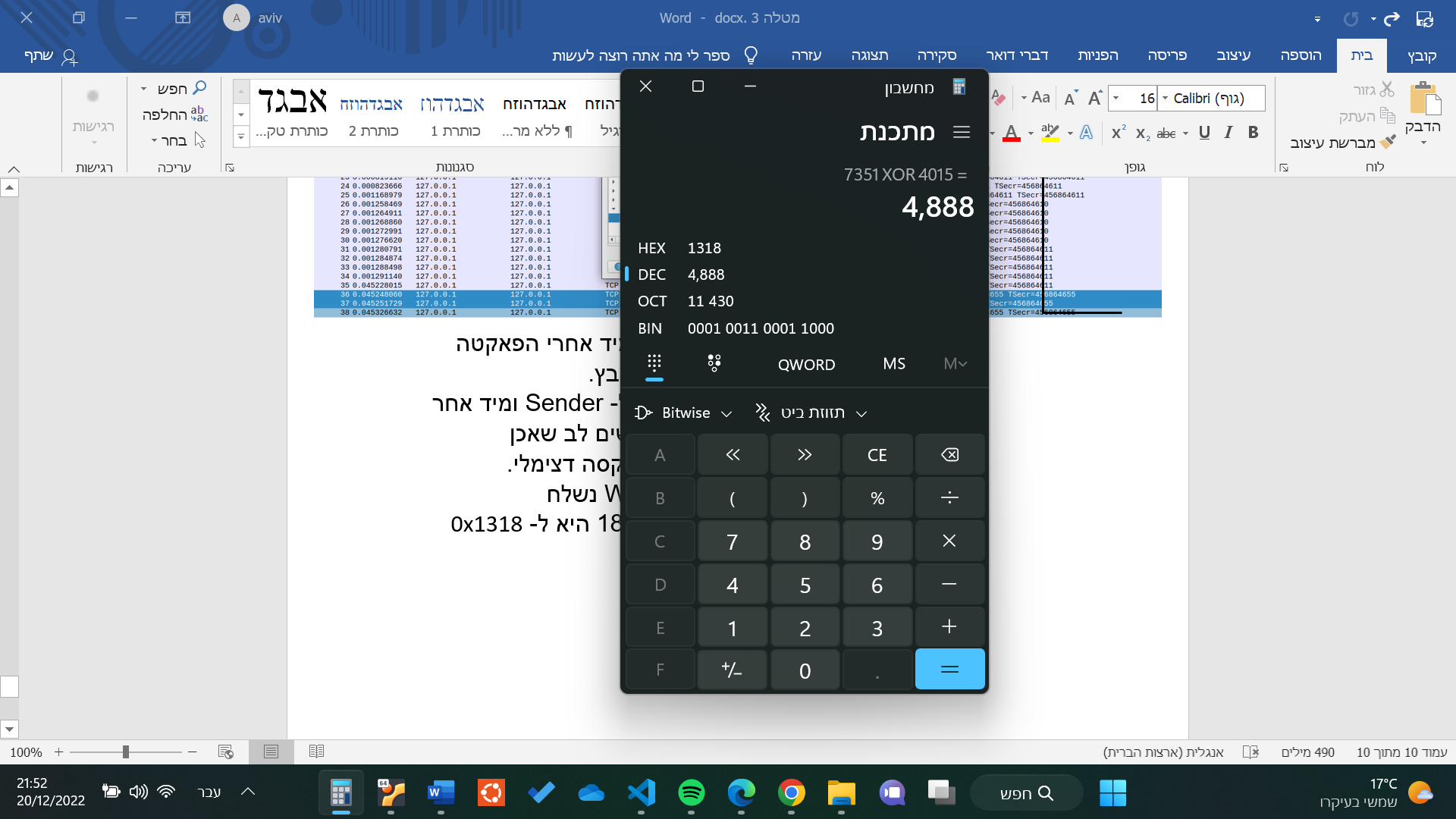
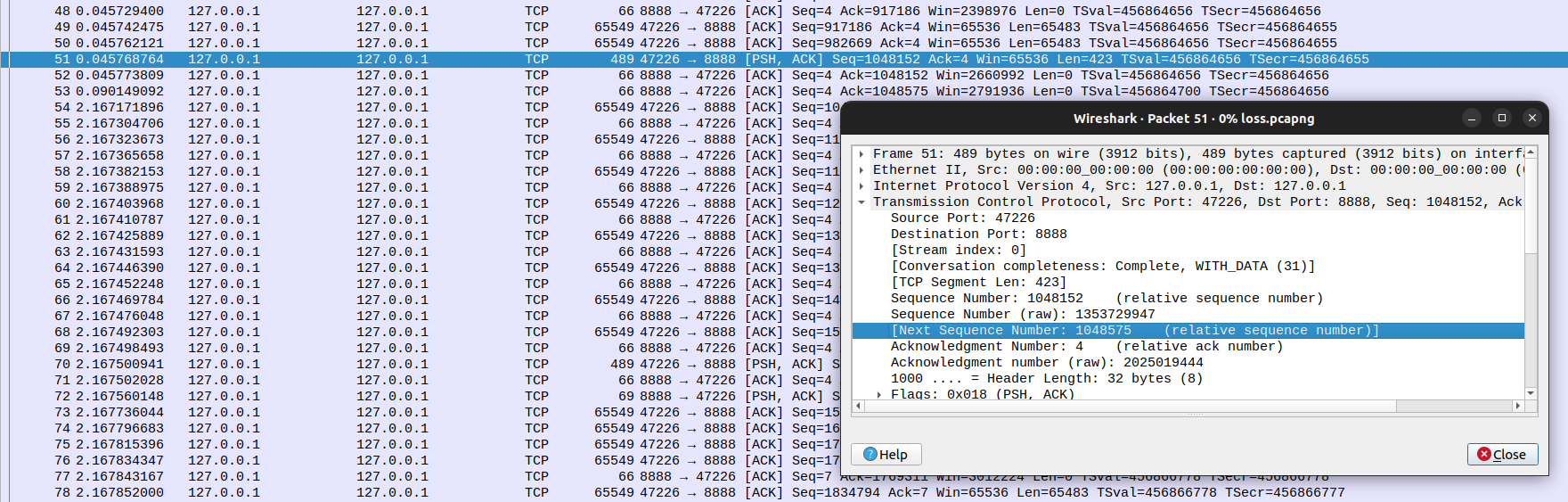
* Receiver.c – מימוש צד השרת
* Receiver.h – בו כל ההצהרות על הפונקציות והקבועים   
   בהם הקובץ Receiver.c משתמש
* 1mb\_file.txt - קובץ טקסט בגודל 1 מגה בייט.

**הסבר מפורט על הקוד:**

**Sender**

**Receiver**

**Wireshark**

1. **0% איבוד:**
   1. פתיחת קשר:  
      ניתן לראות, פקטת SYN, פקטת SYN,ACK לפתיחת קשר, ופקטת ACK המאשרת זאת.
   2. שליחת חצי ראשון:  
      ניתן לראות שגודל הבתים שקיבלנו עד כה הוא 524287, כיוון שה-byte הבא שנקבל הוא byte מספר 524288, ונשים לב כי גודל חצי הקובץ ששלחנו הוא אכן 524287.  
        
        
      (1048574 / 2 = 524287)
   3. שליחת אוטנטיקציה:  
      נשים לב כי האוטנטיקציה נשלחת מיד אחרי הפאקטה האחרונה של החצי הראשון של הקובץ.  
      קודם נשלח ACK מה-Receiver ל- Sender ומיד אחר כך נשלחת האוטנטיקציה. בנוסף נשים לב שאכן האוטנטיקציה נכונה אך מופיעה בהקסה דצימלי.  
      בנוסף נבחין שה-Data ב-WireShark נשלחת   
      כ-Little Endian ולכן הכוונה ב- 181300 היא ל- 0x1318.
   4. שליחת חצי שני:  
      ניתן לראות שגודל הבתים שקיבלנו עד כה הוא 1048574, כיוון שה-byte הבא שנקבל הוא byte מספר 1048575, ונשים לב כי הגודל המלא של הקובץ ששלחנו הוא אכן 1048574.
   5. סגירת קשר:  
        
      ניתן לראות פקטות FIN,ACK לסגירת קשר ופקטת ACK המאשרת זאת.
2. **10% איבוד:**