**מבוא לרשתות תקשורת**

**תרגיל 1**

**עידו ברקאי ויוגב אברבאנל**

**חלק א**

1) ראשית, שינינו את הודעת השליחה לשמות והת"ז בקובץ הלקוח:

A screenshot of a computer

Description automatically generated with medium confidence

לאחר מכן הרצנו משני טרמינלים את קוד השרת והלקוח והסנפנו את התעבורה על כרטיס ה loopback (זאת מאחר והשרת והלקוח הורצו בשני טרמינלים שונים על גבי אותה המכונה):

Graphical user interface, text, application, email

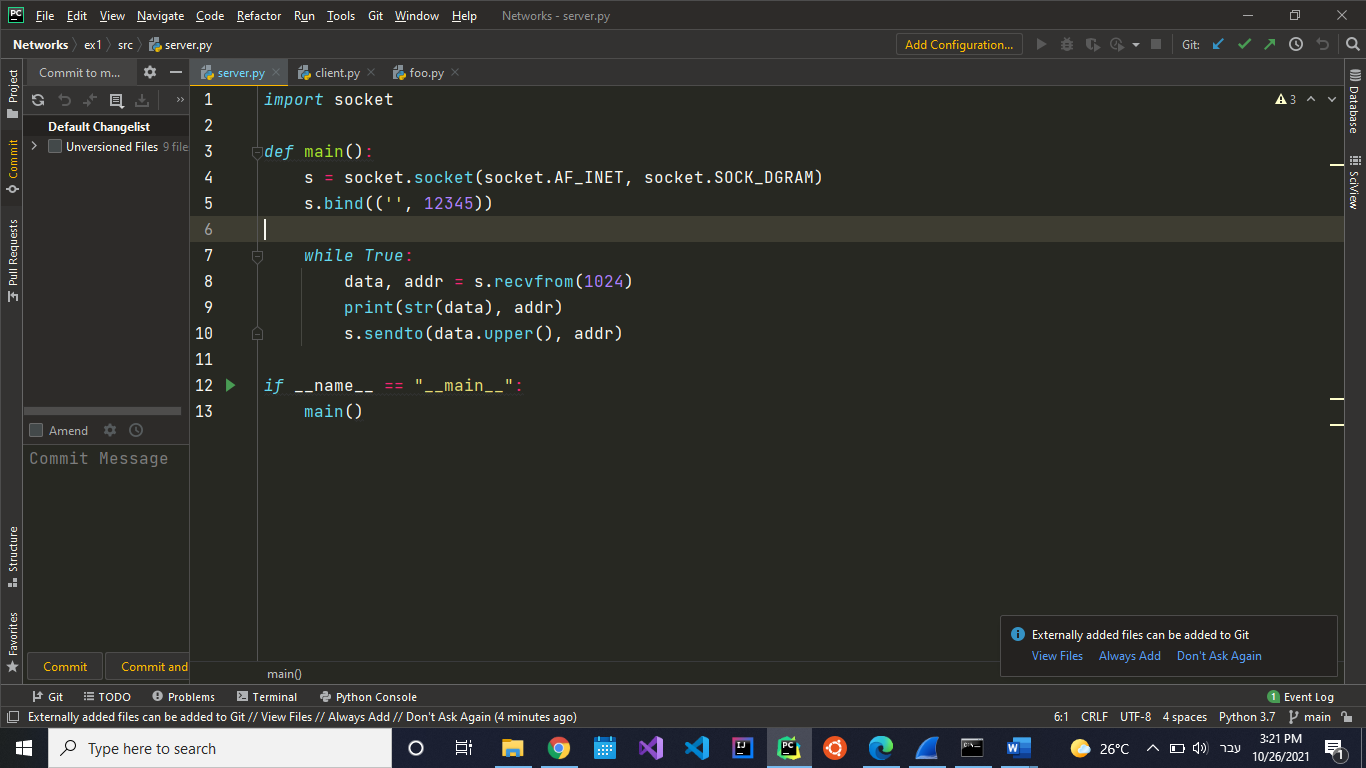
Description automatically generated

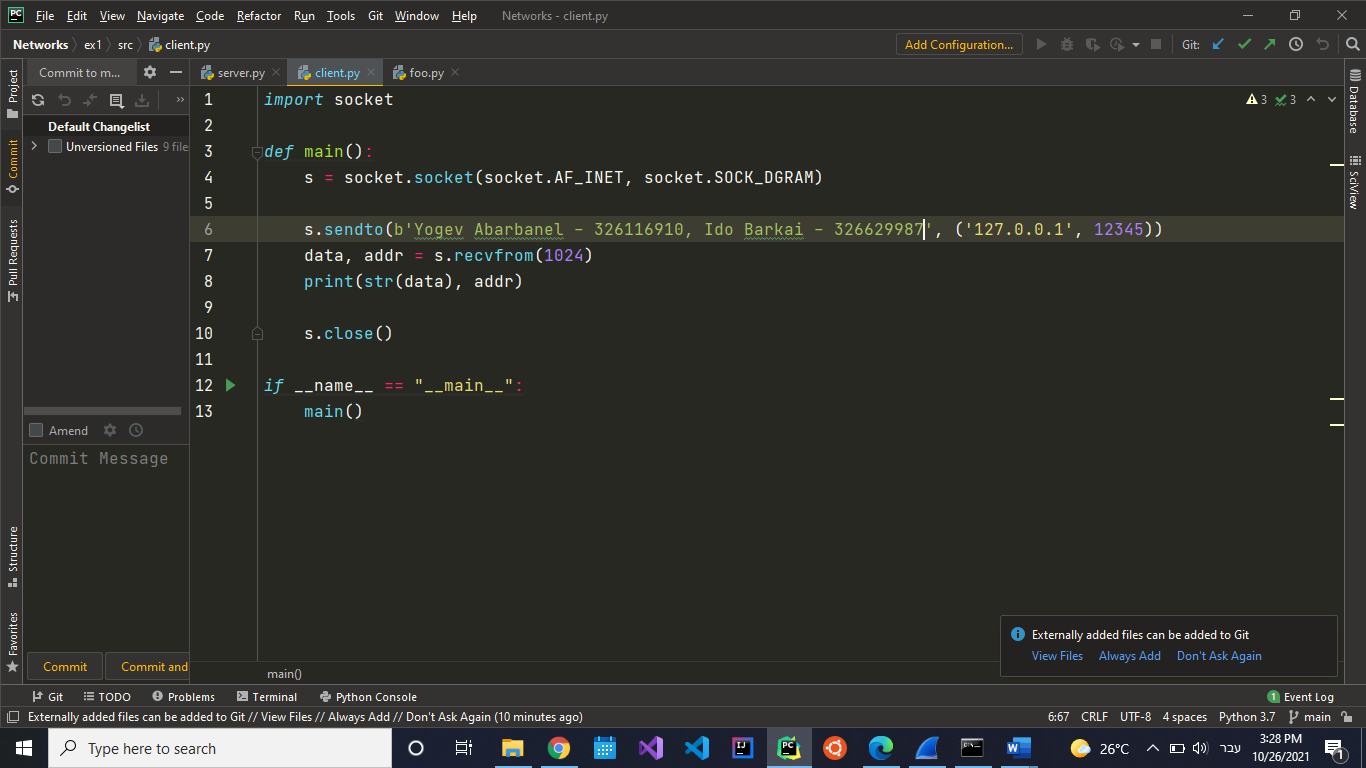
2) היה צריך לסנן את החבילות שהן לא של הלקוח והשרת, אז הוספנו את השורה הבאה ב-filter:

udp && udp.port == 12345 && ip.addr == 127.0.0.1

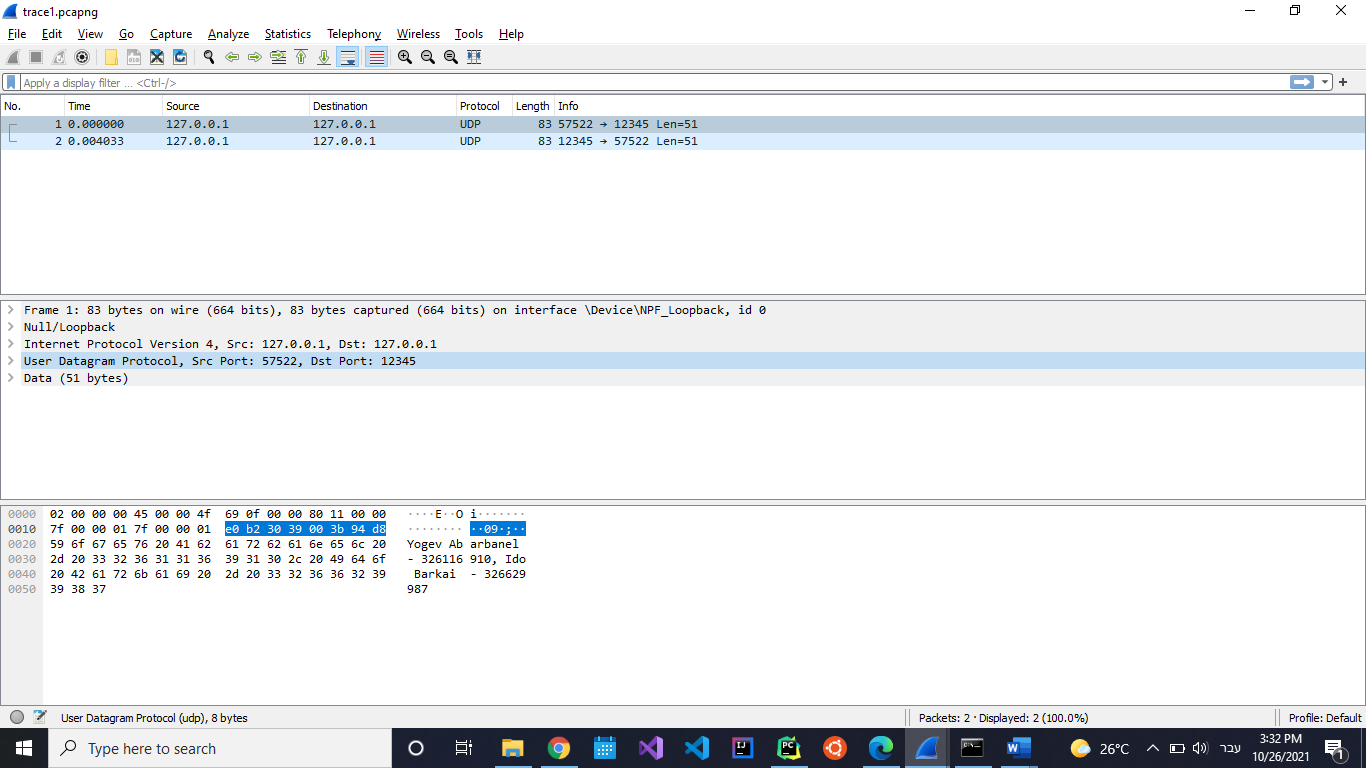
* מסנן חבילות שלא בפרוטוקול תקשורת של השרת והלקוח
* מסנן חבילות מסוג מסויים שגם בפרוטוקול הנ"ל אבל לא של השרת והלקוח
* מסנן לפי כתובת ה-ip של הlopback

3) בשרת יש שימוש במספר פורט כאן:



השימוש הוא בתפיסת הפורט במספר הנ"ל, כך שכאשר מישהו ירצה לשלוח הודעה לשרת, הוא יוכל לשלוח למחשב ולפי מספר הפורט הוא ידע לאיזה אפליקציה לשלוח את ה-pacet. בקוד השרת הוא נמצא בשליחת ההודעה, כדי שהיא תגיע לאפליקציה הנכונה כפי שהסברנו: 

הפורט נמצא בשכבת התעבורה, מספר הפורט הוא חלק מההדר של שכבת התעבורה ונמצא בהודעה:



הפורטים מסומנים בכחול, זאת הודעה שנשלחה מהלקוח לשרת, כיוון שבלקוח אין bind לפורט מסוים, מערכת ההפעלה הקצתה לו אחד שהיה פנוי (src port: 57522), לעומת זאת כפי שעשינו בbind הפורט של השרת הוא האחד שהקצאנו לו (dst port:12345).

4) כפי שניתן לראות בכחול :

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

החבילה הזאת הגיעה מה-ip 127.0.0.1 ונשלחה ל-ip 127.0.0.1. (בדומה החבילה השנייה)

Graphical user interface, text

Description automatically generated

כפי שניתן לראות בכרטיסי הרשת, זהו הכרטיס של lopback, כלומר ההודעה נשלחה מהמחשב הזה 127.0.0.1 מציין זאת, ונשלחה אל עצמו 127.0.0.1.

**פרוטוקול שכבת האפליקציה**

כדי להתמודד עם זריקת החבילות והדילי, הפרוטוקול שלנו הוסיף מספר סידורי לחבילה וטיימאאוט בלקוח.

**בשרת**, אנו מתחזקים מונה, בהגעת חבילה:  
1.בודק האם המספר הסידורי של החבילה תואם למונה:  
 א. אם כן,

-מדפיס אותה  
-מעלה את המונה ב1  
-שולח ack (החבילה עצמה)  
ב. אם לא, שולח ack (החבילה עצמה)  
  
  
לקוח:  
1. שולח חבילה ומפעיל טיימאאוט  
2. מחכה לack:  
א. אם לא קיבל והטיימאאוט נגמר אז הוא שולח אותה שוב (1)  
ב. אם קיבל, בודק את מספרה הסידורי:  
-אם שונה מהמונה, זורק אותה  
-אחרת:  
\*מעלה את המונה  
\*שולח את החבילה הבאה (1 שוב)  
  
  
  
  
  
נכונות הפרוטוקול:  
1. במקרה של חבילות עד גודל 100 טפלנו בעת יצירת ההודעה.  
2. עם זריקת פקטות טפלנו על ידי הטיימאאוט, במקרה שהודעה נזרקה הטיימאאוט יגיע לסיומו והיא תשלח שוב.  
כיוון שיתכן ממקרה שבו, הטיימאאוט יסתיים בגלל שהack עדיין לא הגיע (מקצב השליחה ברשת או זריקת ה-ack על ידי foo ), ואז תקרה שליחה חוזרת של הודעה אשר לא נזרקה,

הוספנו את המספר הסידורי כדי שהשרת יוכל למנוע שליחת ack כפול על חבילה יחידה.

3. במקרים של עיכוב, טפלנו בדיעבד על ידי טיפול ב-2, אם יש עיכוב של חבילה, כלומר הסדר לא נכון, אנו נתעלם בשרת מכל הפקטות עם מספר סידורי לא תואם ורק נחזיר עליהן ack (למקרה שה-ack הקודם נזרק ולא הגיע ללקוח), וגם בקוד לקוח אם ה-ack לא תואם למונה נתעלם.

כיוון שאנו עובדים בשיטת stop & wait, יכולות להגיע פקטות רק מהמספר הסידורי הדרוש ומטה, לכן בהחזרת ack, אם הוא קטן יותר זה לא ישפיע כי הלקוח יתעלם אבל אם זאת החבילה הנוכחית הלקוח יתיחס ויקדם את המונה.

**חלק ב**

נפרט על פעולתו של הקוד בכל אחד מן השלבים[[1]](#footnote-1):

לאורך כל ההסבר שלושת התוכנות רצו על טרמינלים שונים כמתואר בטבלה הבאה:

|  |  |  |  |
| --- | --- | --- | --- |
| client | foo | server |  |
| רנדומלי, מוקצה ע"י מערכת ההפעלה | 12345 | 23456 | port |
| 127.0.0.1 | 127.0.0.1 | 127.0.0.1 | ip |

**שלב 1**

בשלב זה foo "מתנהג יפה" ומעביר את החבילות כמו שצריך בתנאי שאינן מכילות יותר מ-100 בתים.

נצרף דוגמת הרצאה של התכנית:

**server**

>> python3 server.py 23456

**foo**

**>>** python3 foo.py 12345 127.0.0.1 23456 1

**client**

>> python3 client.py 127.0.0.1 12345 input.txt

נסניף את התעבורה המתאימה ב wireshark המגיעה ויוצאת מן foo[[2]](#footnote-2):

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי

ונתסכל על השכבות השונות בפקטה:

שכבת התעבורה – בשכבת התעבורה עבדנו עם פרוטוקול UDP, לכן ניתן לראות בשכבה זו פורט מקור ופורט יעד:



**תמונה 1.1** : פקטה מ client אל foo



**תמונה 1.2** : הפקטה מועברת ל server



**תמונה 1.3** : פקטת תגובה חוזרת מ server ל foo

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

**תמונה 1.4**: הפקטה חוזרת ל client

כמו שניתן לראות, פורט המקור ופורט היעד משתנים בהתאם למסלול הפקטה

שכבת הרשת – בשכבת הרשת נראה את כתובות ה-ip, מאחר והרצנו את התוכנות בטרמינלים שונים באותה המכונה נראה כי כולם יהיו 127.0.0.1



**תמונה 1.5** : כתובת המקור והיעד בפקטות

שכבת אפליקציה

בשכבת האפליקציה ניתן לראות את המידע שמועבר בפקטות, הכתוב בפרוטוקול שהגדרנו לעיל[[3]](#footnote-3). לדוגמה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

**תמונה 1.6** : הפקטה הראשונה שנשלחה מ client ל foo

**שלב 2**

בשלב זה foo בנוסף לתנאי של גודל החבילות זורק באקראי אחוז מסויים של החבילות בשני כיוונים. על כן נראה תעבורה יותר מסיבית – הרבה יותר פקטות יישלחו.

נצרף דוגמת הרצאה של התכנית:

**server**

>> python3 server.py 23456

**foo**

**>>** python3 foo.py 12345 127.0.0.1 23456 2

**client**

>> python3 client.py 127.0.0.1 12345 input.txt

נסניף את התעבורה המתאימה ב wireshark המגיעה ויוצאת מן foo:

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי

התעבורה אכן יותר גדולה – 114 פקטות לעומת 68 בשלב 1

נתסכל על השכבות השונות בפקטה:

שכבת התעבורה – בשכבת התעבורה לא השתנה הפרוטוקול. עבדנו עם פרוטוקול UDP, לכן ניתן לראות בשכבה זו פורט מקור ופורט יעד:



**תמונה 1.1** : פקטה מ client אל foo



**תמונה 1.2** : הפקטה מועברת ל server



**תמונה 1.3** : פקטת תגובה חוזרת מ server ל foo

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

**תמונה 1.4**: הפקטה חוזרת ל client

כמו שניתן לראות, פורט המקור ופורט היעד משתנים בהתאם למסלול הפקטה

שכבת הרשת – גם בשכבת הרשת לא השתנה הפרוטוקול. נראה את כתובות ה-ip, מאחר והרצנו את התוכנות בטרמינלים שונים באותה המכונה נראה כי כולם יהיו 127.0.0.1



**תמונה 1.5** : כתובת המקור והיעד בפקטות

שכבת אפליקציה

בשכבת האפליקציה ניתן לראות את המידע שמועבר בפקטות, הכתוב בפרוטוקול שהגדרנו לעיל – בשלב זה כבר מספרנו כל פקטה, נוכל לראות זאת בתחילת חלק המידע. לדוגמה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

**תמונה 1.6** : הפקטה הראשונה שנשלחה מ client ל foo

**שלב 3+4**

שלבים אלו עובדים באותו פרוטוקול של שלב 2 בשכבת האפליקציה (כמפורט לעיל), הרשת והתעבורה ולכן נראה מידע דומה בפקטות.

אכן נשים לב שהתעבורה תהיה גדולה ככל שפרמטרי העיכוב וזריקה יגדלו.

1. קובץ הטקסט מכיל חלק מן הסיפור "אליסה בארץ הפלאות" [↑](#footnote-ref-1)
2. הסינון בוצע ע"י הפילטר udp&&udp.port==12345 [↑](#footnote-ref-2)
3. בשלב זה עוד לא היה צריך למספר את הפקטות [↑](#footnote-ref-3)