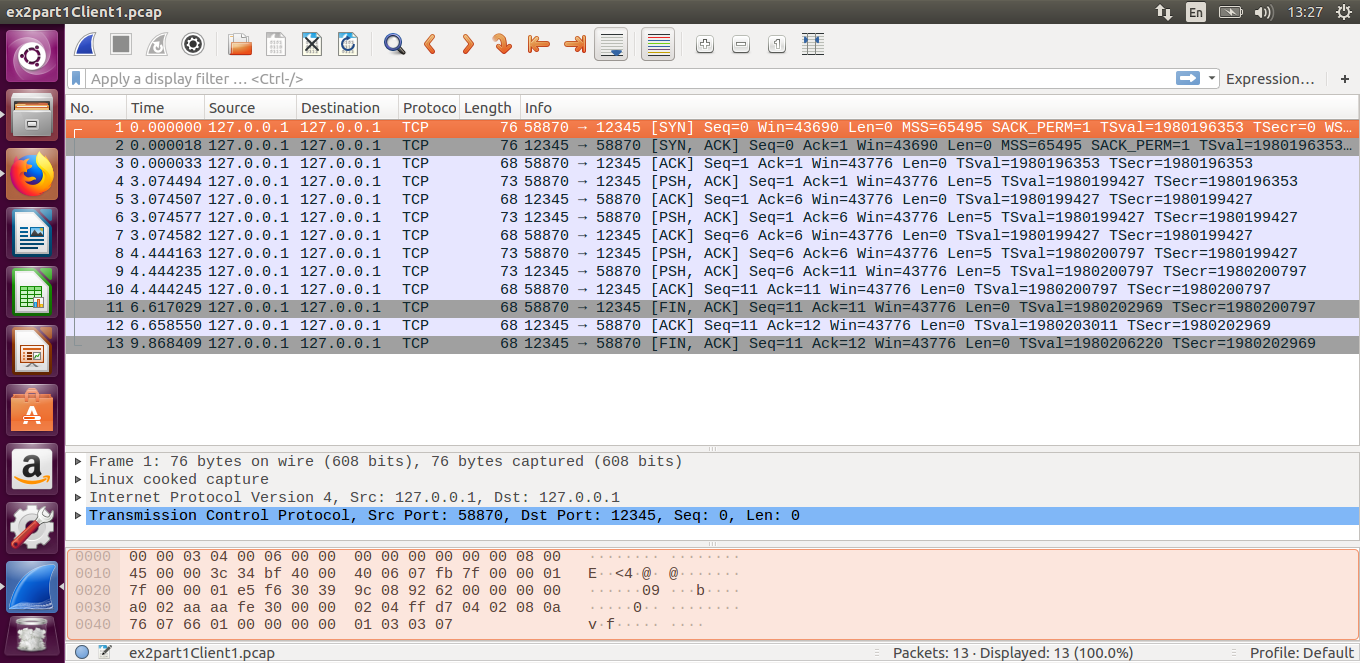
**רשתות תקשורת – תרגיל מס' 2**

**שאלה 1**

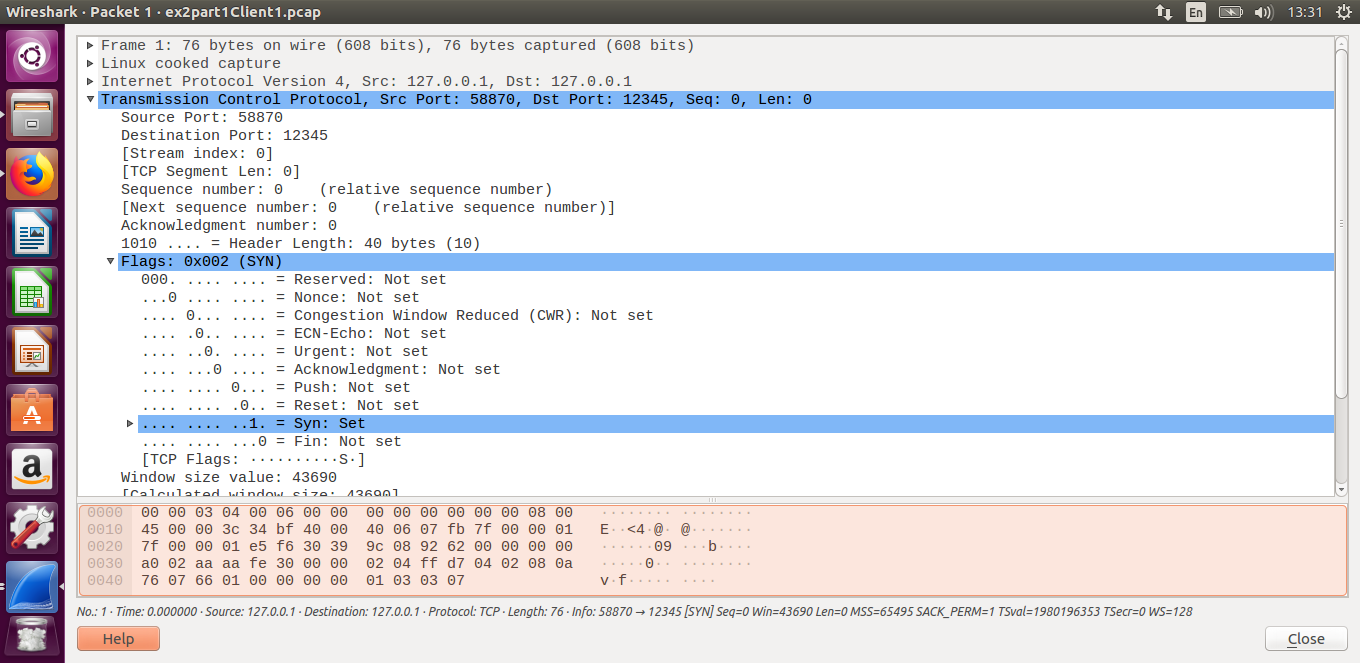
**א.**

תהליך הhandshake עם הלקוח הראשון:



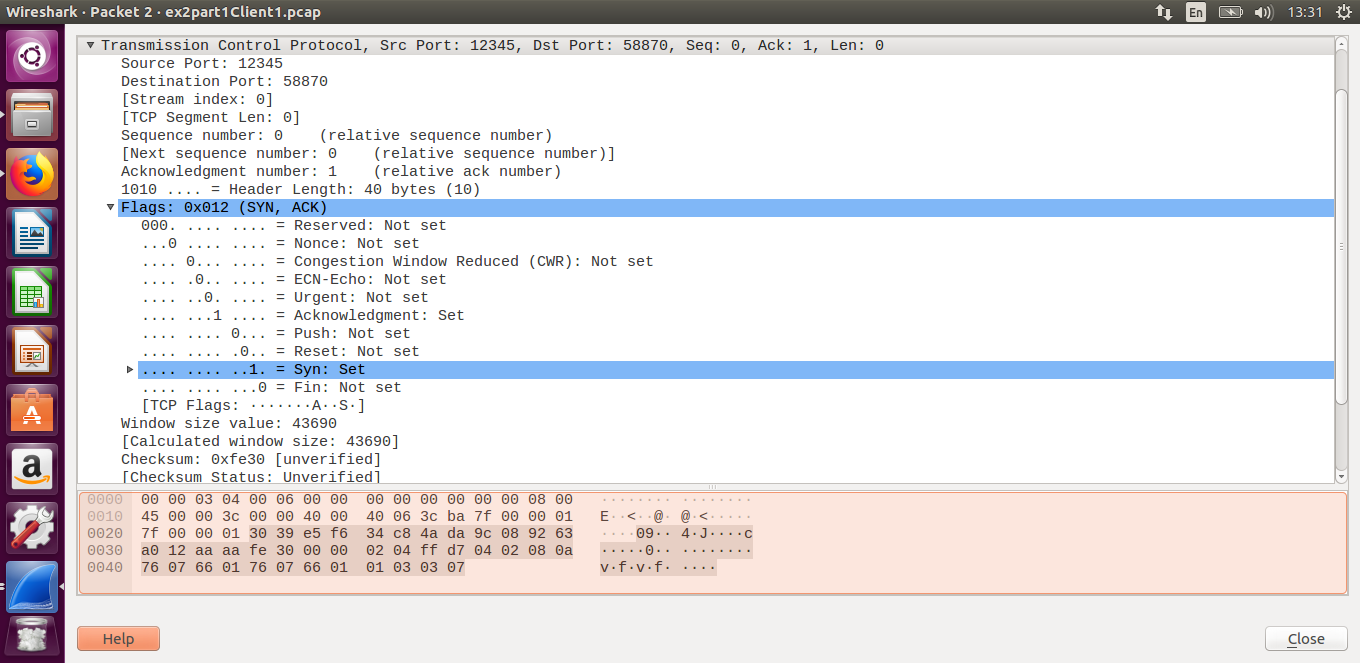
ניתן לראות כי הפורט בו הלקוח הראשון משתמש הוא פורט מס' 58870.

כעת נפרט את כל התהליך:



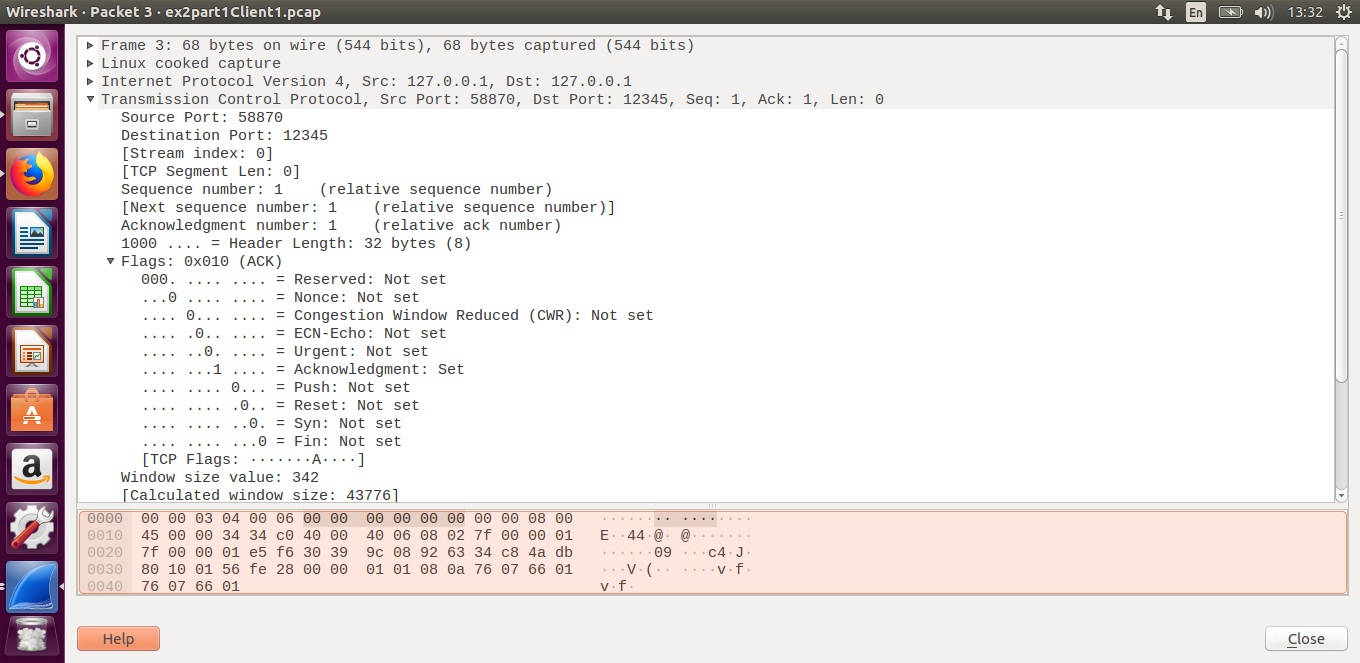
ניתן לראות כי החבילה הראשונה בhandshake שנשלחת היא מהלקוח לסרבר לצורך הקמת שיחה.

ניתן לאות כי הסרבר משתמש בפורט 12345 והלקוח בפורט 58870 , כמו כן ניתן לראות בדגלים כי דגל הsyn דולק.

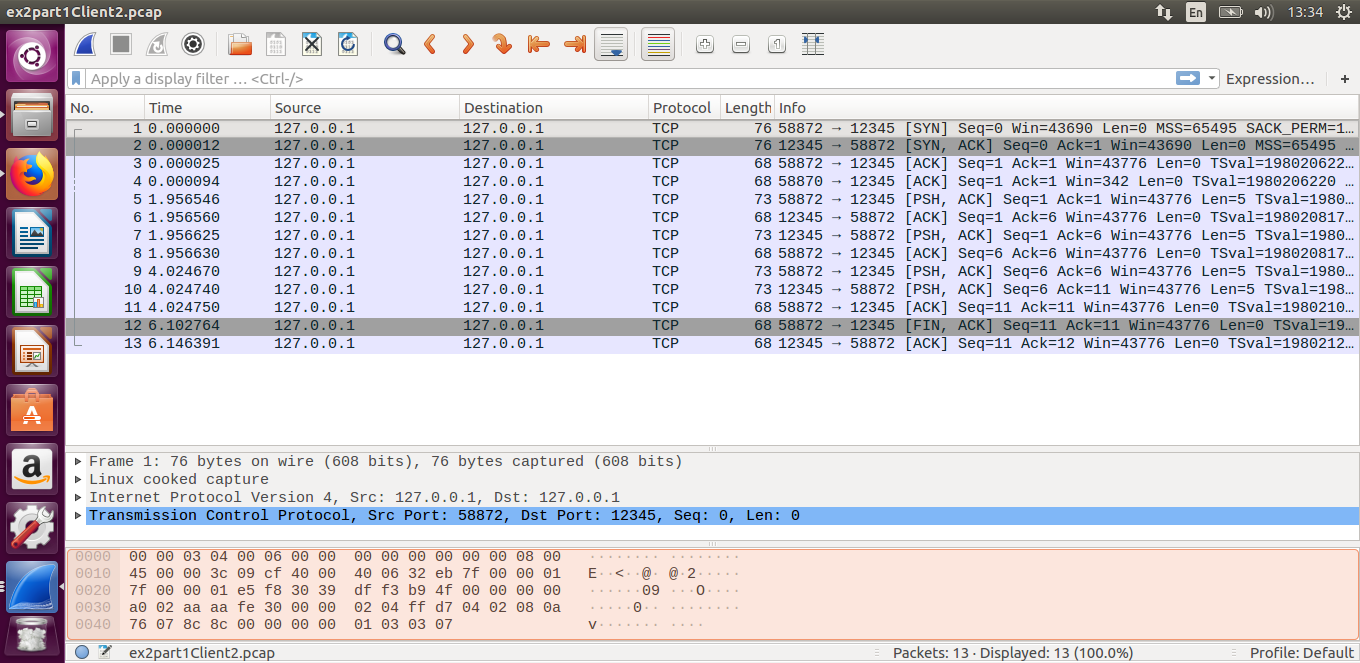


כעת הסרבר שולח הודעה חזרה ללקוח שמצהירה ללקוח למעשה שהוא פנוי , והם יכולים להתחיל לנהל שיחה.

שוב ניתן לראות כי דגל הsyn דלוק.

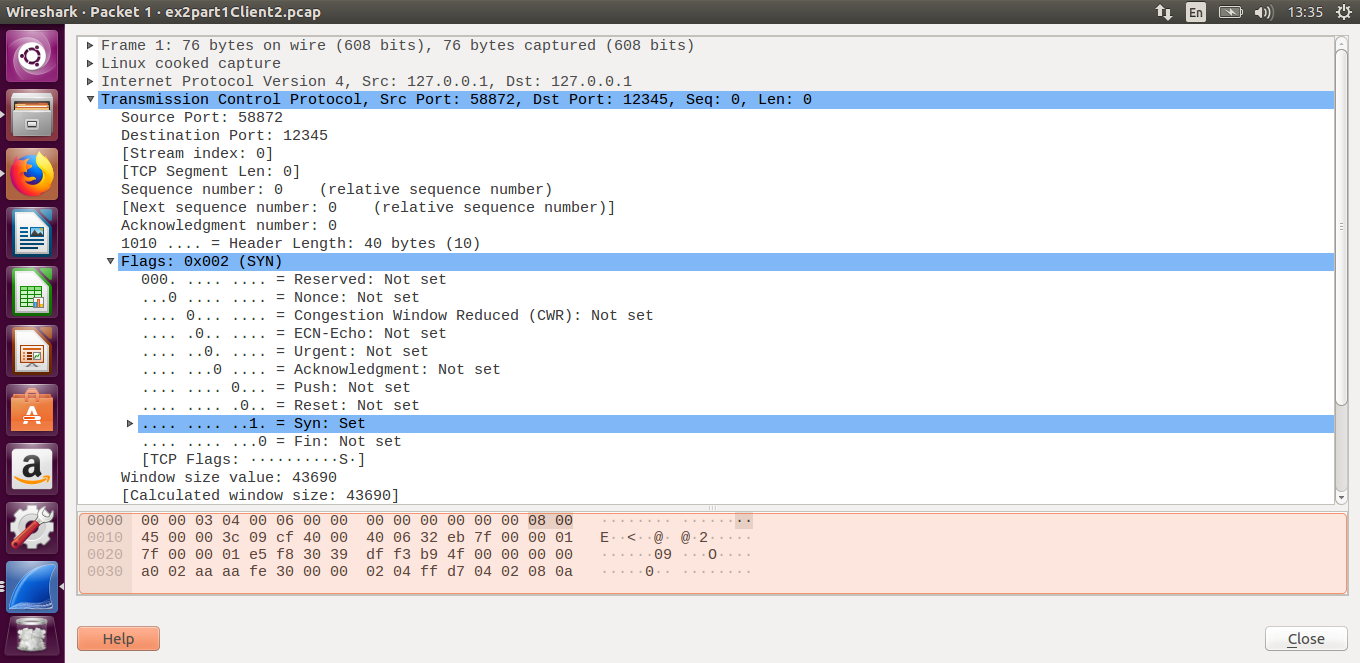
כעת הלקוח מקבל את ההודעה מהסרבר , וכעת הוא יודע שהוא יכול להתחיל במטרת השיחה מבחינתו שאצלנו זה שליחת ההודעות hello וworld.

**כעת נראה את תהליך הhandshake עם הלקוח השני:**

****

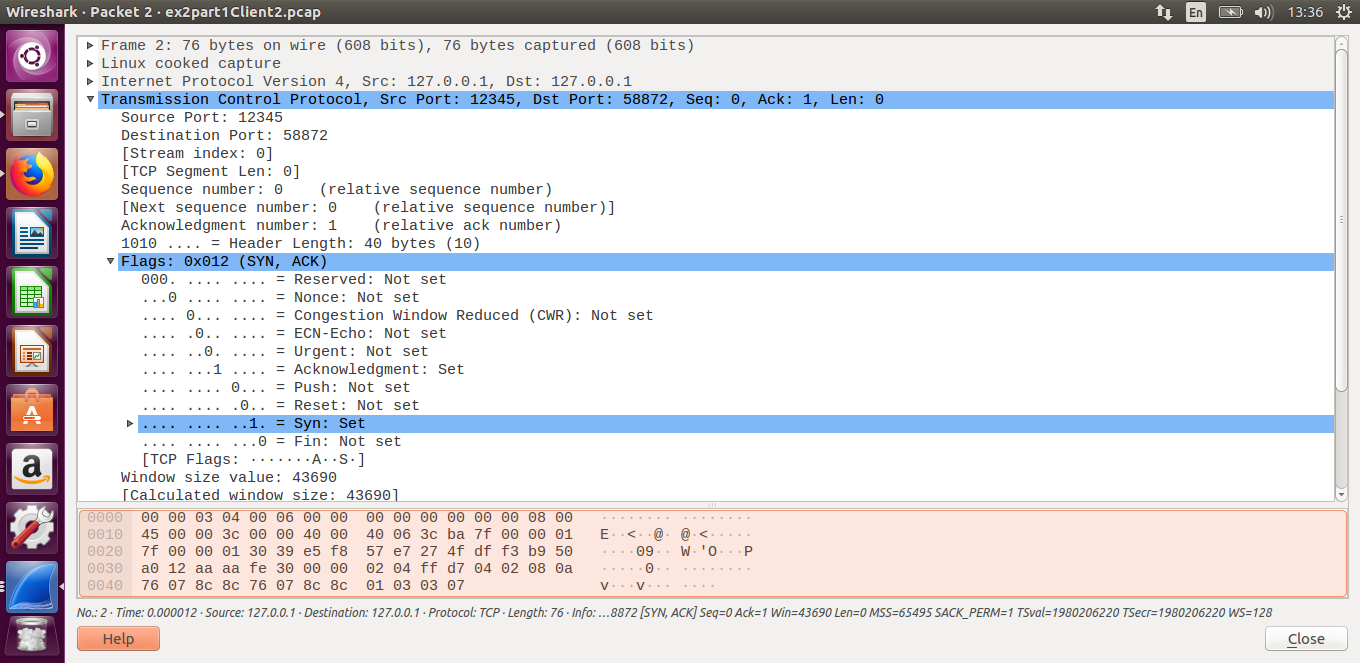
ניתן לראות כי הפורט בו הלקוח הראשון משתמש הוא פורט מס' 58872.

כעת נפרט את כל התהליך:



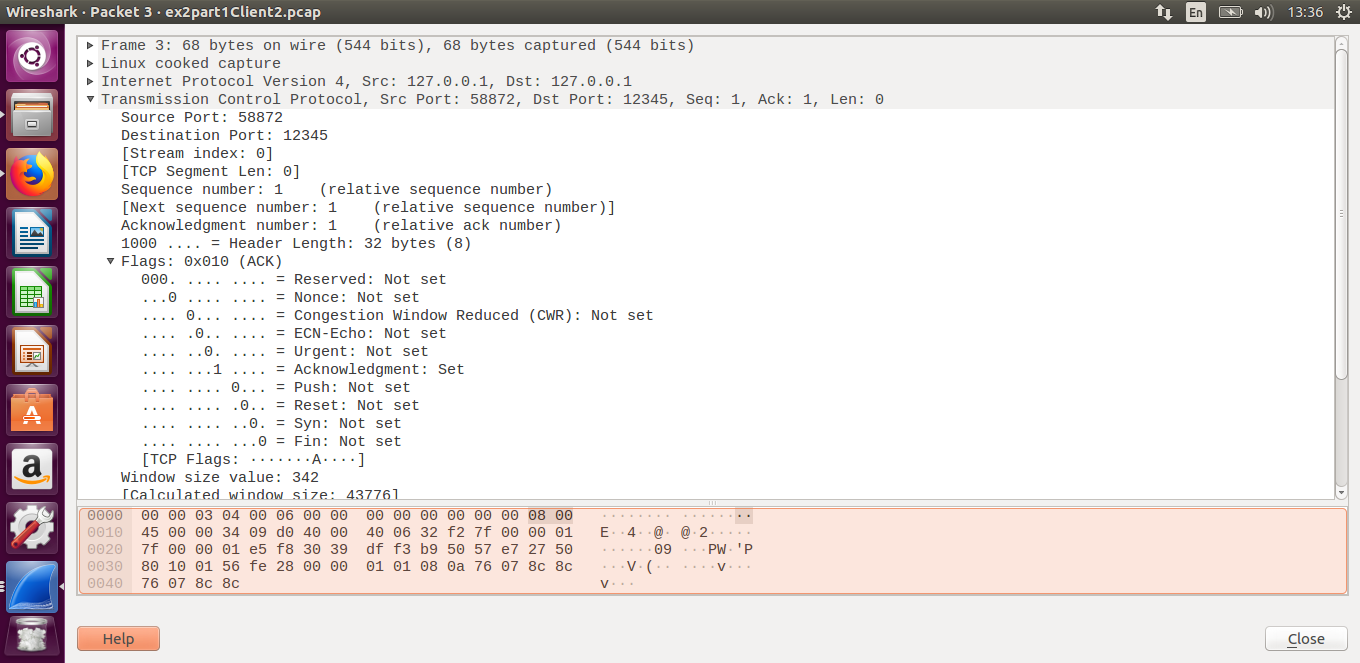
ניתן לראות כי החבילה הראשונה בhandshake שנשלחת היא מהלקוח לסרבר לצורך הקמת שיחה.

ניתן לאות כי הסרבר משתמש בפורט 12345 והלקוח בפורט 58872 , כמו כן ניתן לראות בדגלים כי דגל הsyn דולק.



כעת הסרבר שולח הודעה חזרה ללקוח שמצהירה ללקוח למעשה שהוא פנוי , והם יכולים להתחיל לנהל שיחה.

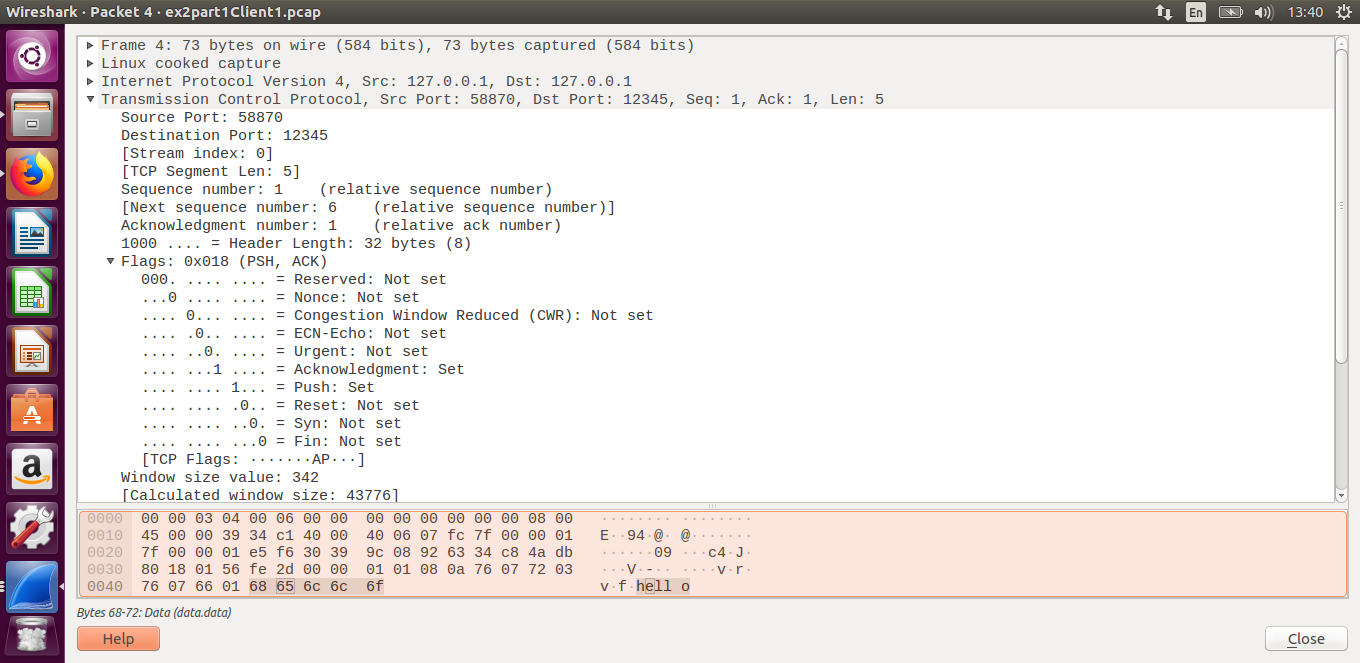
שוב ניתן לראות כי דגל הsyn דלוק.



כעת הלקוח מקבל את ההודעה מהסרבר , וכעת הוא יודע שהוא יכול להתחיל במטרת השיחה מבחינתו שאצלנו זה שליחת ההודעות hello וworld.

**סעיף ב':**

**כעת נפרט את החבילות שהלקוח הראשון שלח לשרת והשרת החזיר להם בהקשר של ההודעות hello וworld. (וכמובן שבסוף הסעיף נדגיש את ההבדלים שיש בין הלקוח הראשון לשני מכיוון שברוב הדברים הם זהים).**

****

**ניתן לראות כי הלקוח שולח לסרבר כאן את הודעת הhello.**

**פרטים על שכבת התעבורה בpacket הנ"ל:**

**Source port: 58870 (הפורט של הלקוח הראשון)**

**Dest port: 12345 (הסרבר שלנו)**

**ניתן לראות שהדגלים ACK ו- PSH דלוקים:**

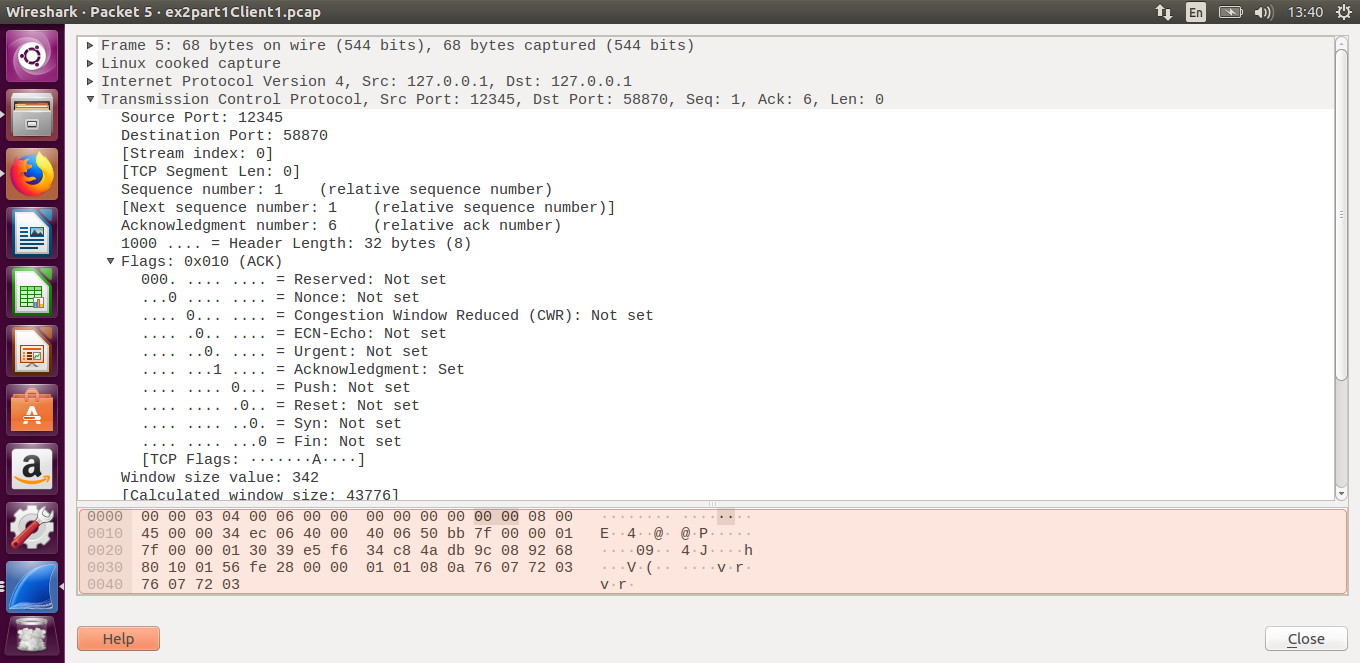
**Sequence number רלטיבי : 1**

**הערך האמיתי של ה Sequence number :** **2617807459**

**Ack numberרלטיבי :1**

**הערך האמיתי של הAck number :** **885541595**

**גודל הheader: 5**

****

**כעת הסרבר שולח ללקוח הודעת ack על ה-hello.**

**פרטים על שכבת התעבורה בpacket הנ"ל:**

**Source port: 12345 (הסרבר שלנו)**

**Dest port: 58870 (הסרבר שלנו)**

**ניתן לראות שהדגל ACK דלוק.**

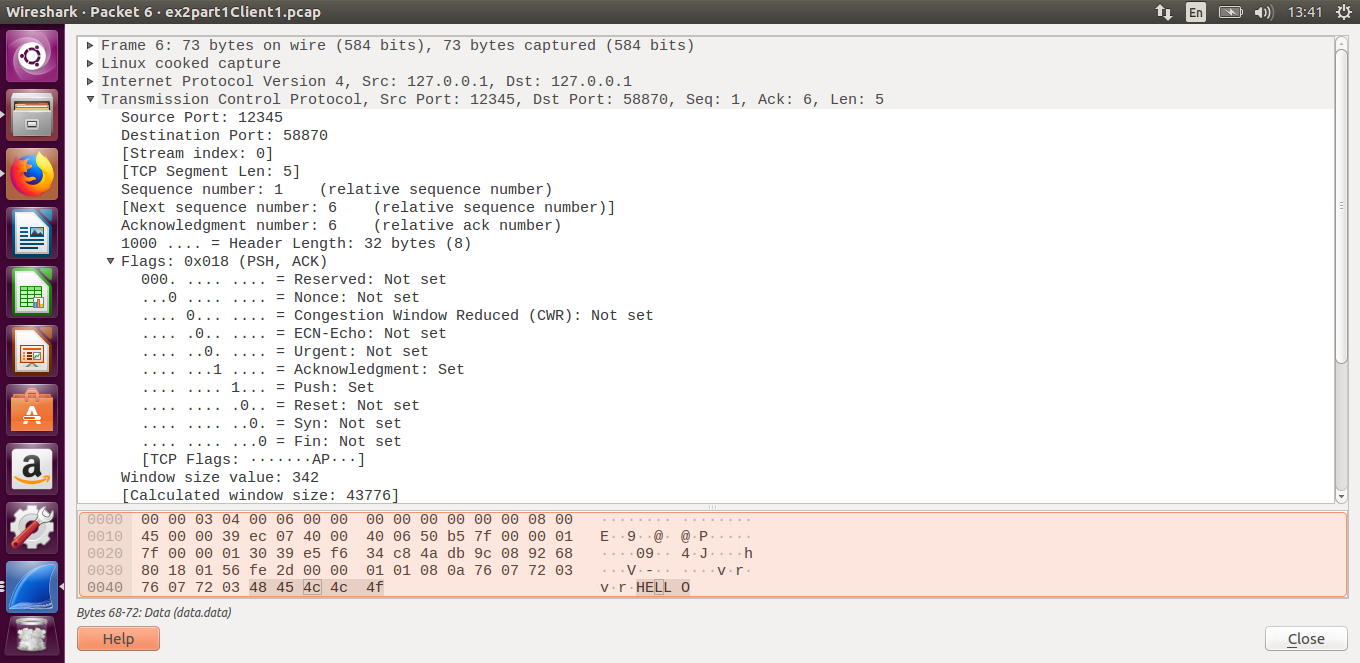
**Sequence number רלטיבי : 1**

**הערך האמיתי של ה Sequence number :** **885541595**

**Ack numberרלטיבי :6**

**הערך האמיתי של הAck number :** **2617807464**

**גודל הheader: 5**

****

**כעת הסרבר מחזיר ללקוח את הודעת הhello.**

**פרטים על שכבת התעבורה בpacket הנ"ל:**

**Source port: 12345 (הסרבר שלנו)**

**Dest port: 58870 (הסרבר שלנו)**

**ניתן לראות שדגלי ה-ACK וה-PSH דלוקים.**

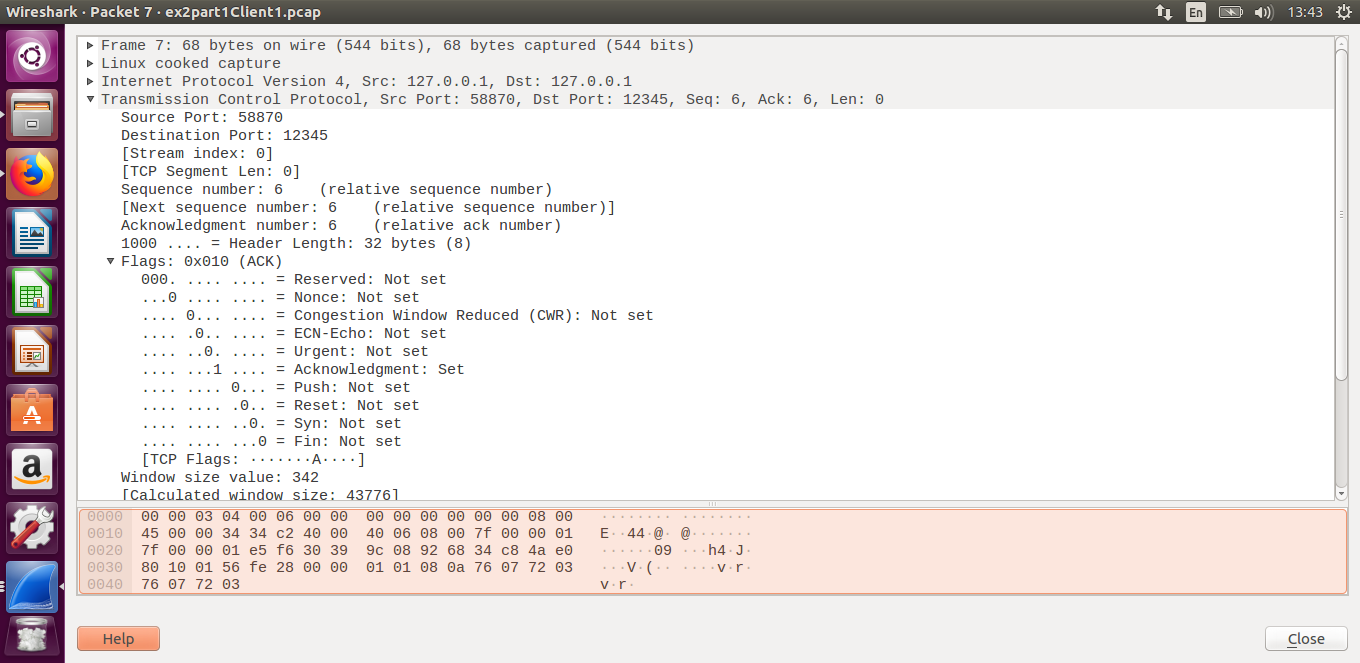
**Sequence number רלטיבי : 1**

**הערך האמיתי של ה Sequence number :** **885541595**

**Ack numberרלטיבי :6**

**הערך האמיתי של הAck number :** **2617807464**

**גודל הheader: 5**

****

**כעת הלקוח שולח ack על קבלת הודעת הhello מהסרבר.**

**פרטים על שכבת התעבורה בpacket הנ"ל:**

**Source port: 58870 (הפורט של הלקוח הראשון)**

**Dest port: 12345 (הסרבר שלנו)**

**ניתן לראות שדגל ה-ACK דלוק:**

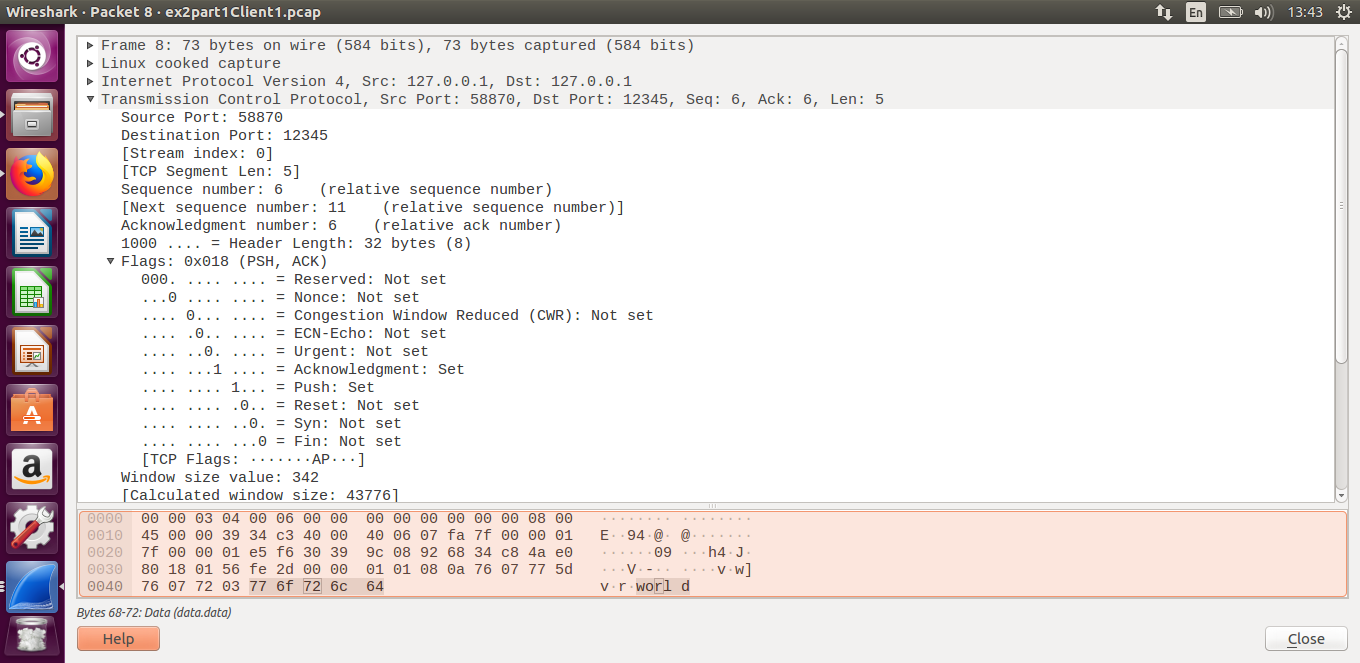
**Sequence number רלטיבי : 6**

**הערך האמיתי של ה Sequence number :** **2617807464**

**Ack numberרלטיבי :6**

**הערך האמיתי של הAck number :** **885541600**

**גודל הheader: 5**

**כעת הלקוח שורת לשרת את הודעת הworld.**

**פרטים על שכבת התעבורה בpacket הנ"ל:**

**Source port: 58870 (הפורט של הלקוח הראשון)**

**Dest port: 12345 (הסרבר שלנו)**

**ניתן לראות שהדגלים ACK ו- PSH דלוקים:**

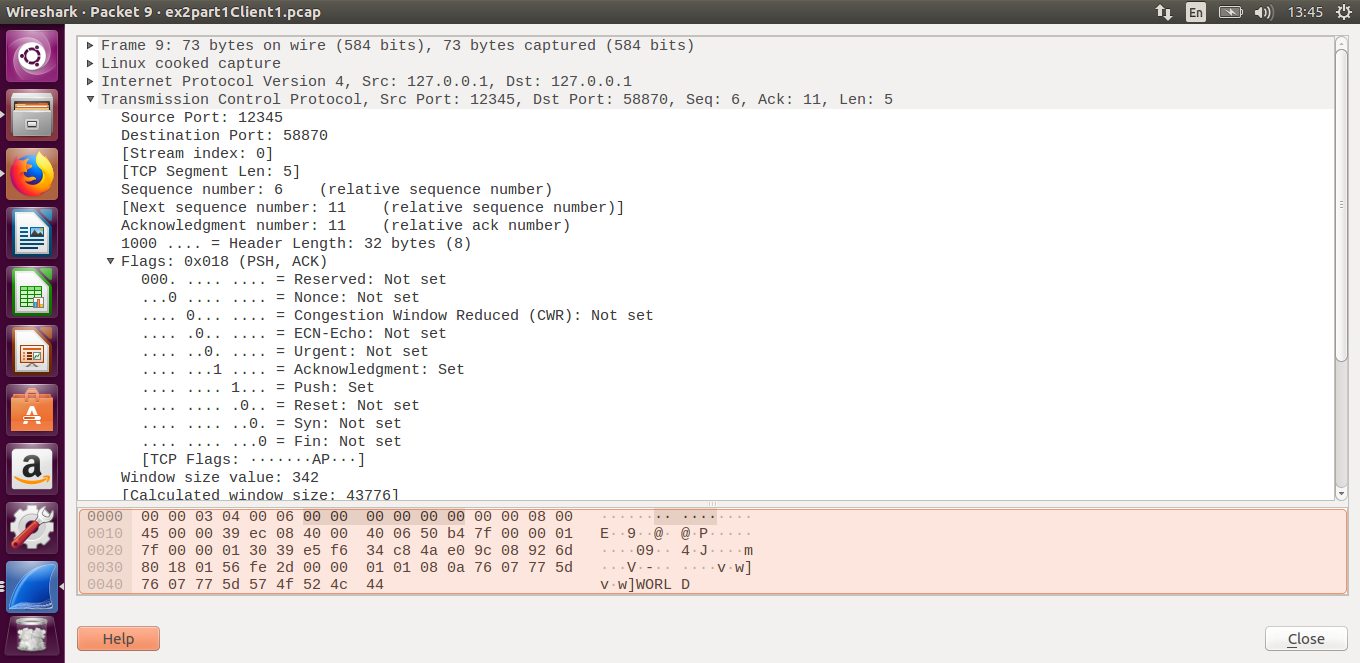
**Sequence number רלטיבי : 6**

**הערך האמיתי של ה Sequence number :** **2617807464**

**Ack numberרלטיבי :6**

**הערך האמיתי של הAck number :** **885541600**

**גודל הheader: 5**

**כעת הסרבר בפאקטה אחת שולח גם הודעת ACK על הworld וגם הודעת PSH על הworld.(שולח ללקוח את הודעת הworld).**

**פרטים על שכבת התעבורה בpacket הנ"ל:**

**Source port: 12345 (הסרבר שלנו)**

**Dest port: 58870 (הסרבר שלנו)**

**ניתן לראות שדגלי ה-ACK וה-PSH דלוקים.**

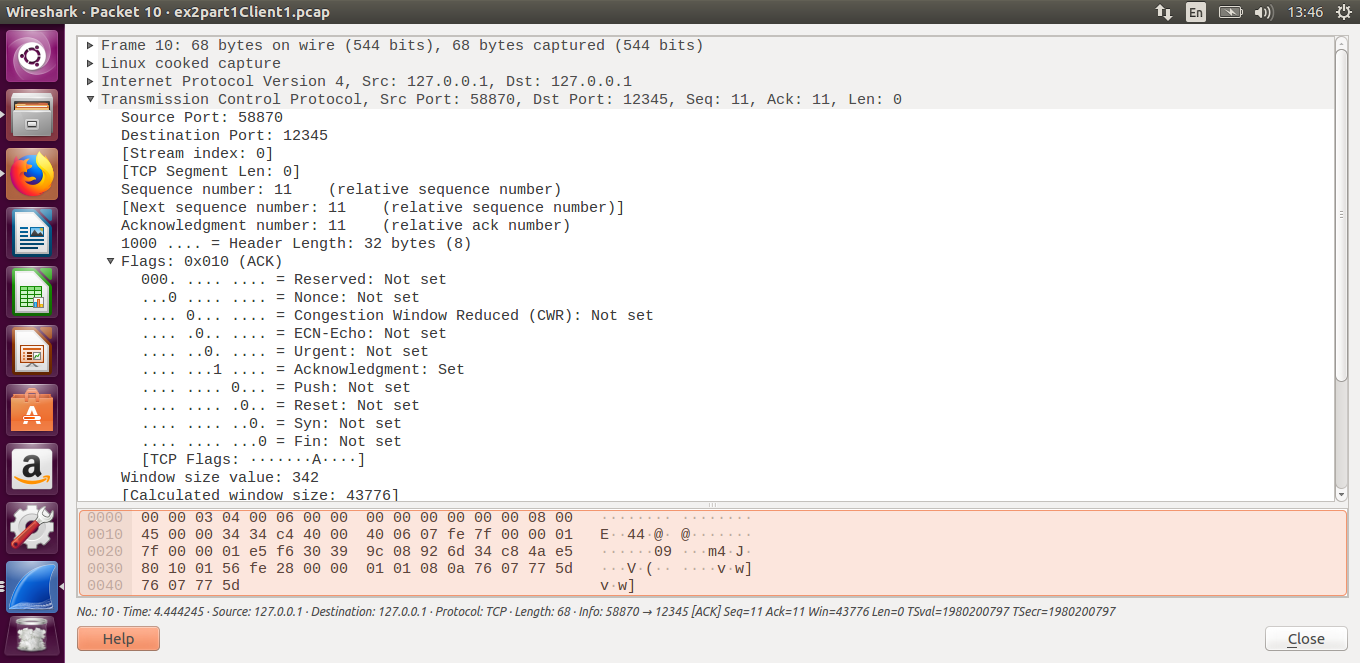
**Sequence number רלטיבי : 6**

**הערך האמיתי של ה Sequence number :** **885541600**

**Ack numberרלטיבי :11**

**הערך האמיתי של הAck number :** **2617807469**

**גודל הheader: 5**

****

**כעת הלקוח מחזיר לשרת ACK על הודעת הworld ובכך מסתיימת התקשורת מבחינת ההעברת ההודעות hello וworld בין הלקוח הראשון לשרת.**

**פרטים על שכבת התעבורה בpacket הנ"ל:**

**Source port: 58870 (הפורט של הלקוח הראשון)**

**Dest port: 12345 (הסרבר שלנו)**

**ניתן לראות שהדגל ACK דלוק.**

**Sequence number רלטיבי : 11**

**הערך האמיתי של ה Sequence number :** **2617807469**

**Ack numberרלטיבי :11**

**הערך האמיתי של הAck number :** **885541605**

**גודל הheader: 5**

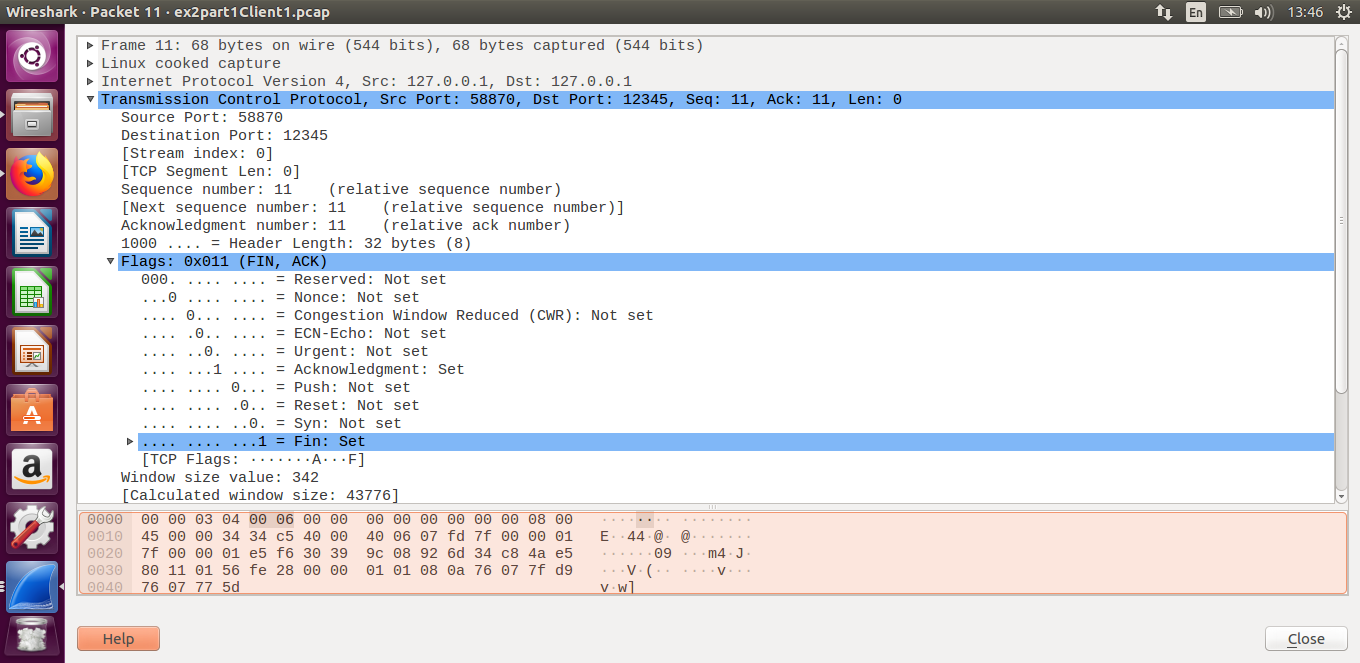
**עבור הלקוח השני קיבלנו את אותם נתונים בשכבת התעבורה רק הport של הקליינט השתנה מ- 58870 ל58872**

**כמו כן הsequence number ההתחלתי (הרלטיבי = 0 ) של הקליינט (הלא רלטיבי) הוא: 3757291855**

**כמו כן הsequence number ההתחלתי (הרלטיבי = 0 ) של הסרבר (הלא רלטיבי) הוא: 1474766671**

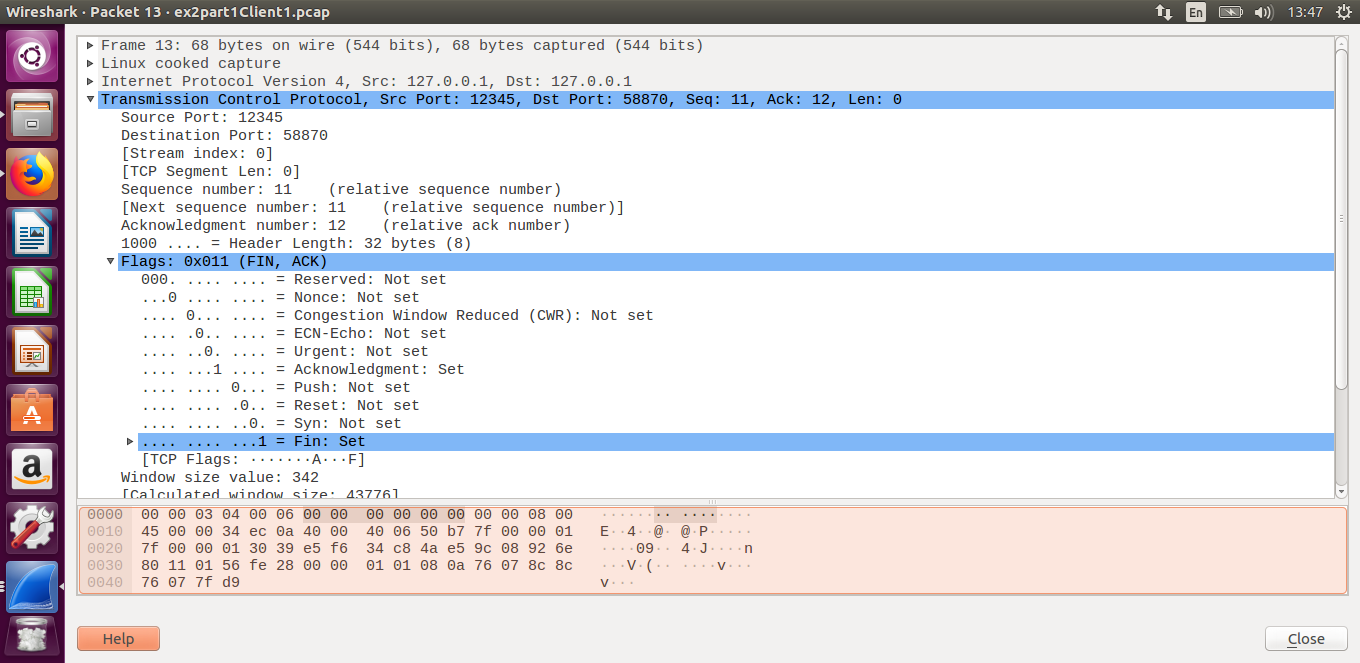
**סעיף ג'**

**כעת נדגים את תהליך הteardown בין הלקוח הראשון לשרת.**

**כאן ניתן לראות כי הלקוח שולח בקשת סגירת שיחה לשרת , כמו כן דגל הack וה-fin דלוקים.**

****

**כאן ניתן לראות כי השרת קיבל את בקשת הסגירה ושלח ACK ללקוח.**

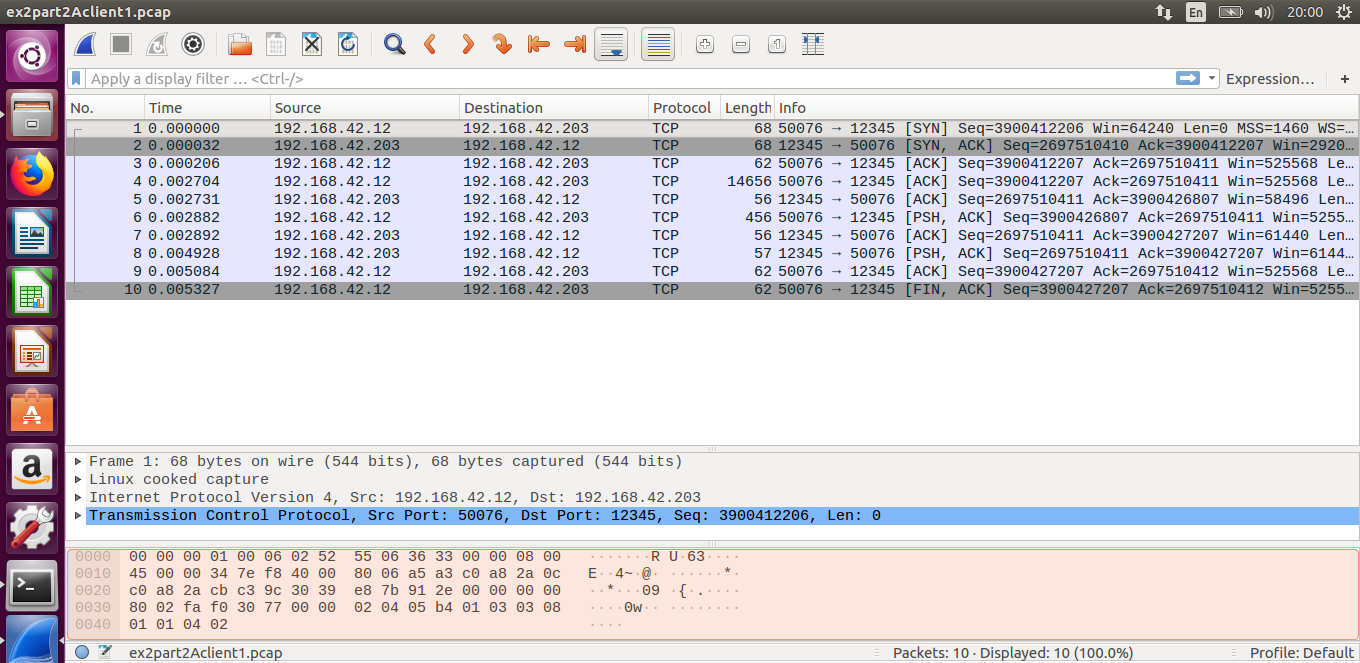
**וכאן השרת שולח את בקשת הסגירה ללקוח , ניתן לראות כי דגלי ACK וfin דלוקים.**

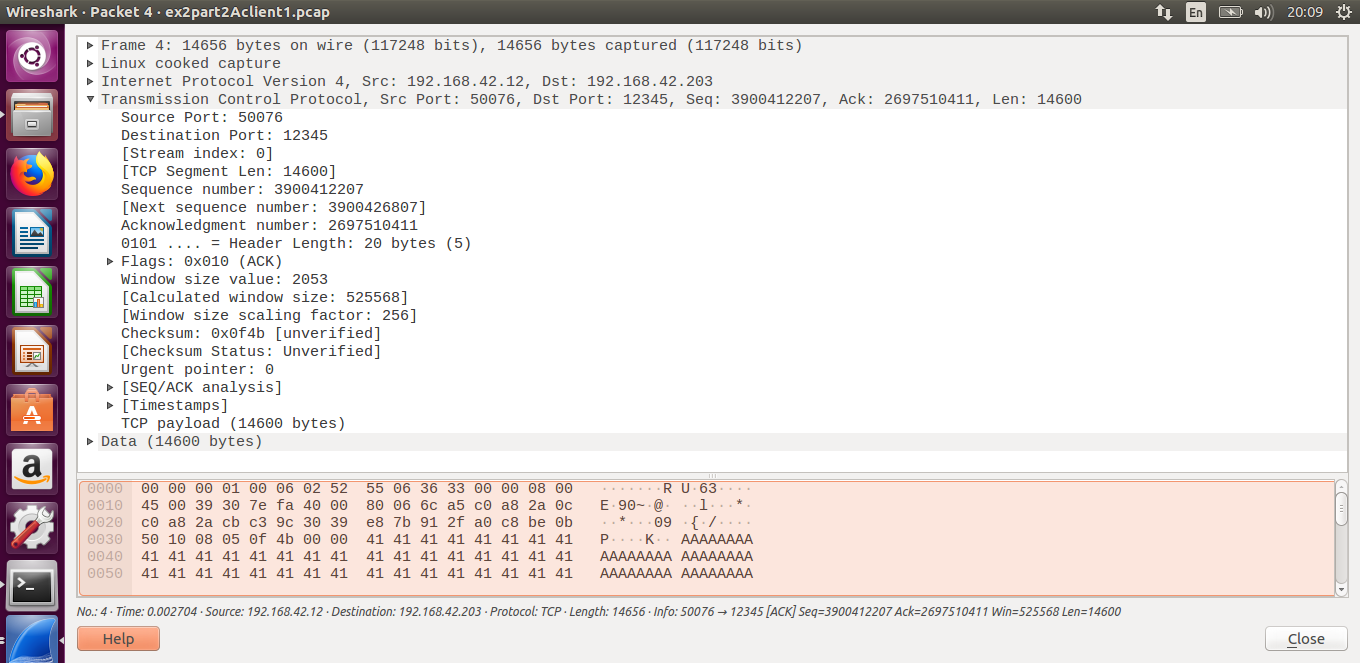
**אותו דבר קורה עם הלקוח השני , וכך למעשה מתבצע תהליך הteardown בסעיף הזה.**

**שאלה 2:**

**סעיף א':**

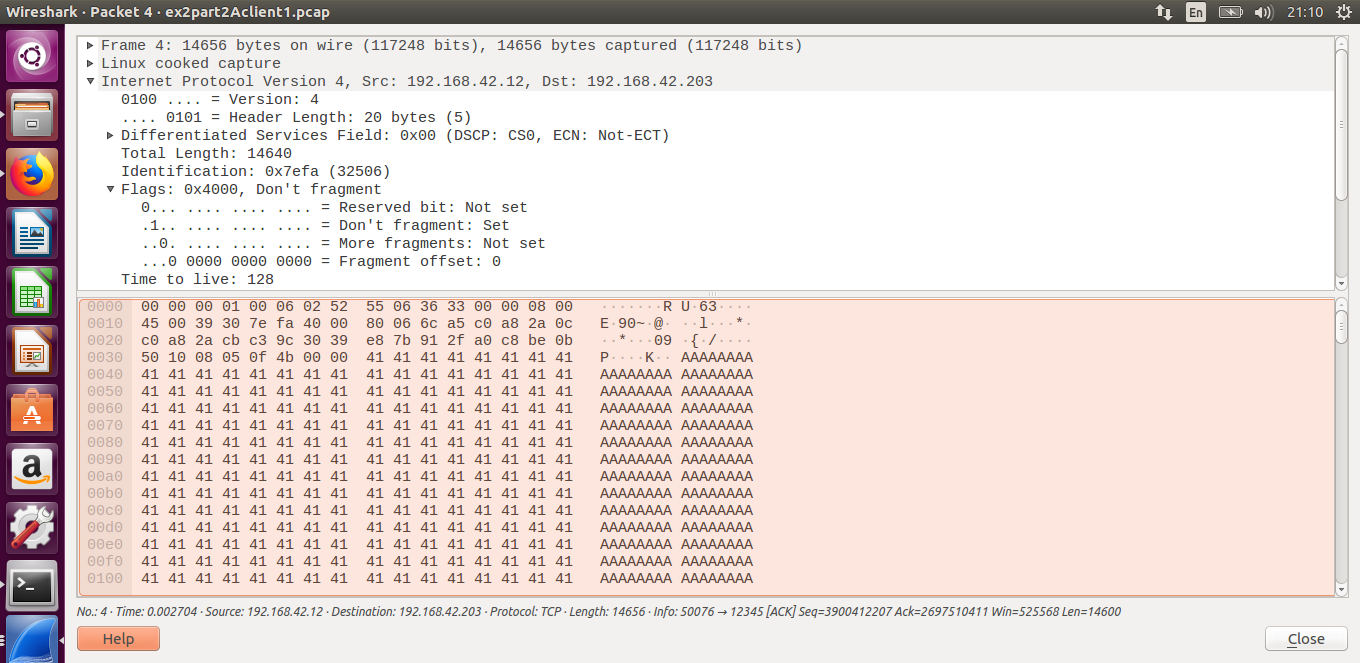
נתבונן בחבילה שהלקוח שלח לשרת:

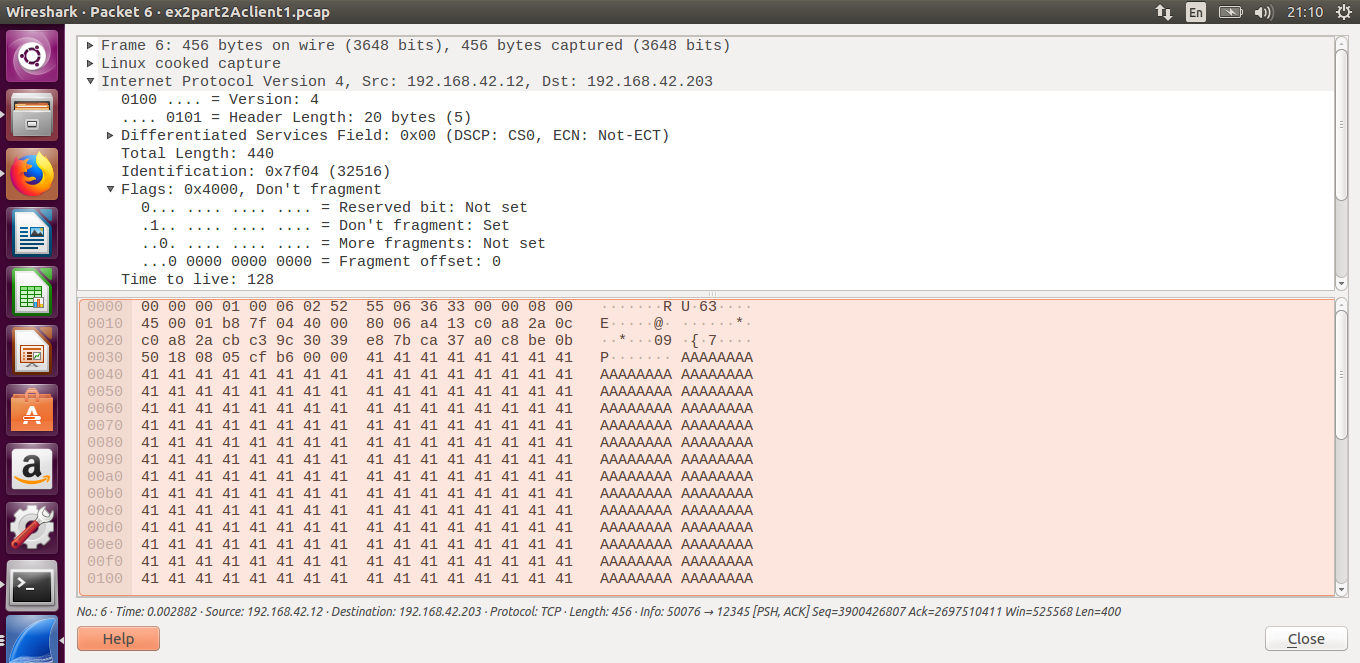




סך הכל הלקוח צריך לשלוח 15000 בתים, בהודעה הראשונה נשים לב שהלקוח שולח לשרת מידע של 14600 בתים (סה"כ 14656 עם header-ים) , כאשר כל המידע הזה הוא A, נשים לב שההבדל בseq number בין ההודעה הזאת(הודעה מס' 4) לבין ההודעה הבאה(הודעה מס' 6) הוא 14,600 (כי הלקוח הגדיל את ה-seq number שלו בעקבות ה-ack של השרת), נשים לב שיש את הדגל don’t fragment בשכבת הרשת.

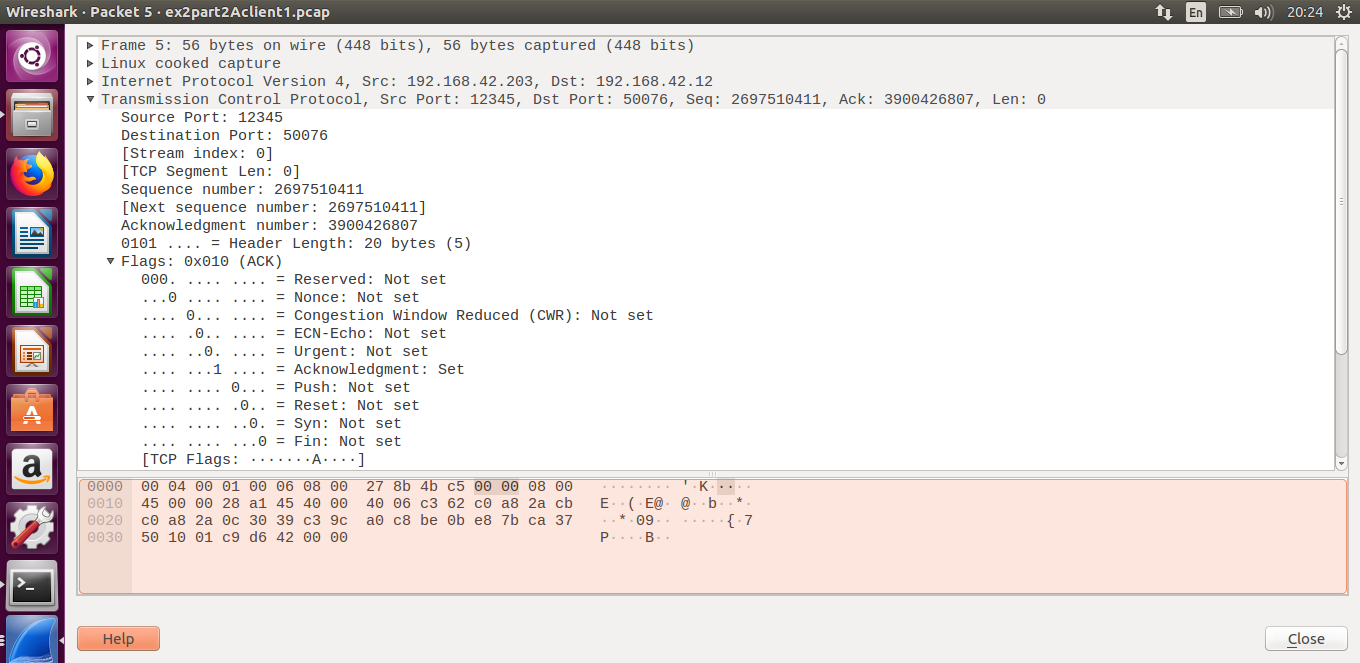
נשים לב שבאף אחת מההודעות לא הייתה פרגמנטציה בשכבת הרשת:





בכל ההודעות של רצפי A בשכבת הרשת גם הדגל don't fragment דולק וגם ה-fragment offset הוא 0 כי אין פרגמנטציה.

התשובה שהתקבלה מהשרת היא:



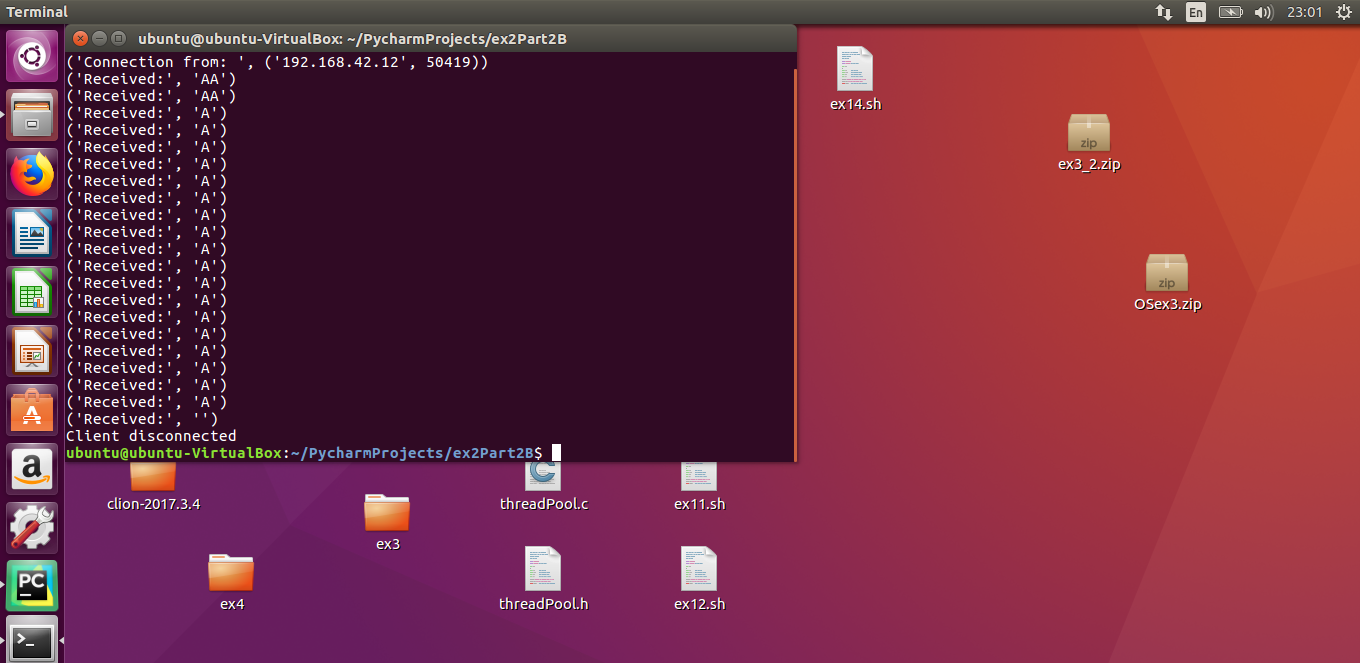
השרת שלח ACK על ההודעה שקיבל , ניתן לראות שה-ACK NUMBER שהשרת שלח גדול ב14600 מהseq number ששלח לו הלקוח עם ההודעה , כלומר השרת קיבל את כל ההודעה(שאורכה 14600).

באופן זה ניתן לראות שהלקוח שלח לשרת מספר הודעות בהן רצפים של 14600 A ולא הודעה אחת של 15000 A כלומר בוצע תהליך סגמנטציה ע"י שכבת התעבורה.

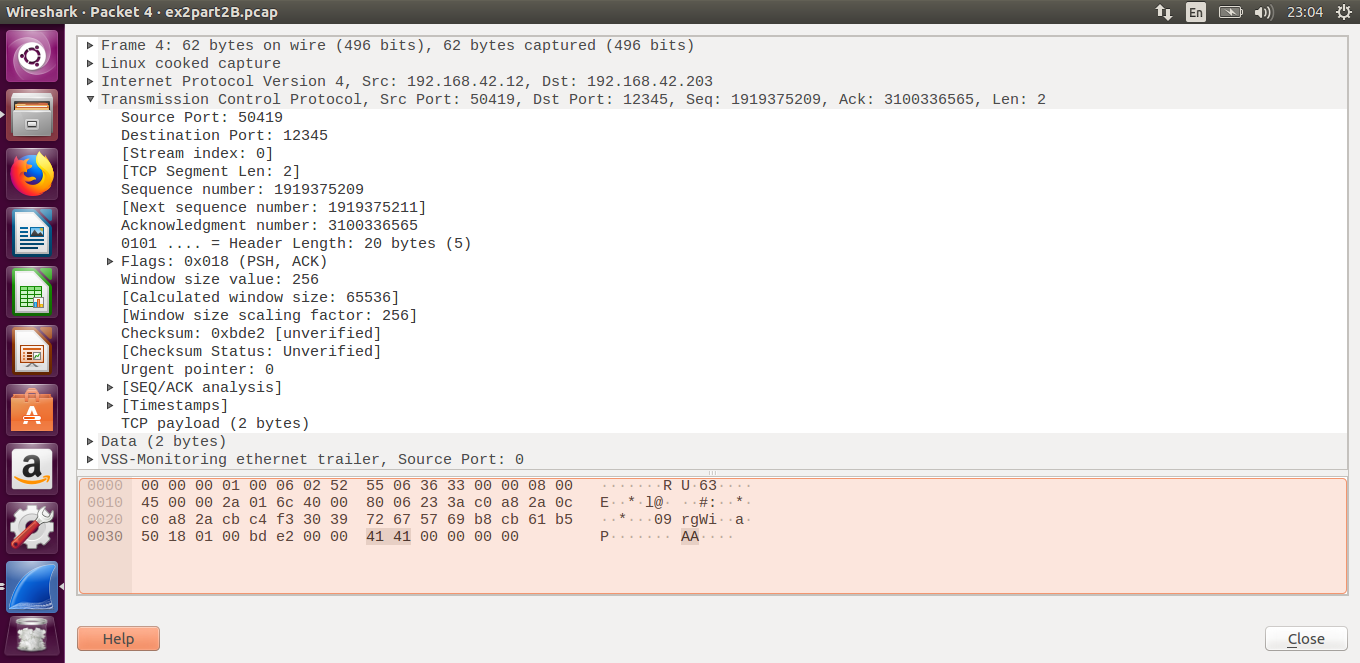
**סעיף ב':**

הip של השרת הוא 192.168.42.203 , הip של הלקוח הוא 192.168.42.12 , הפורט של הסרבר הוא 12345 ופורט הלקוח הוא 50419.

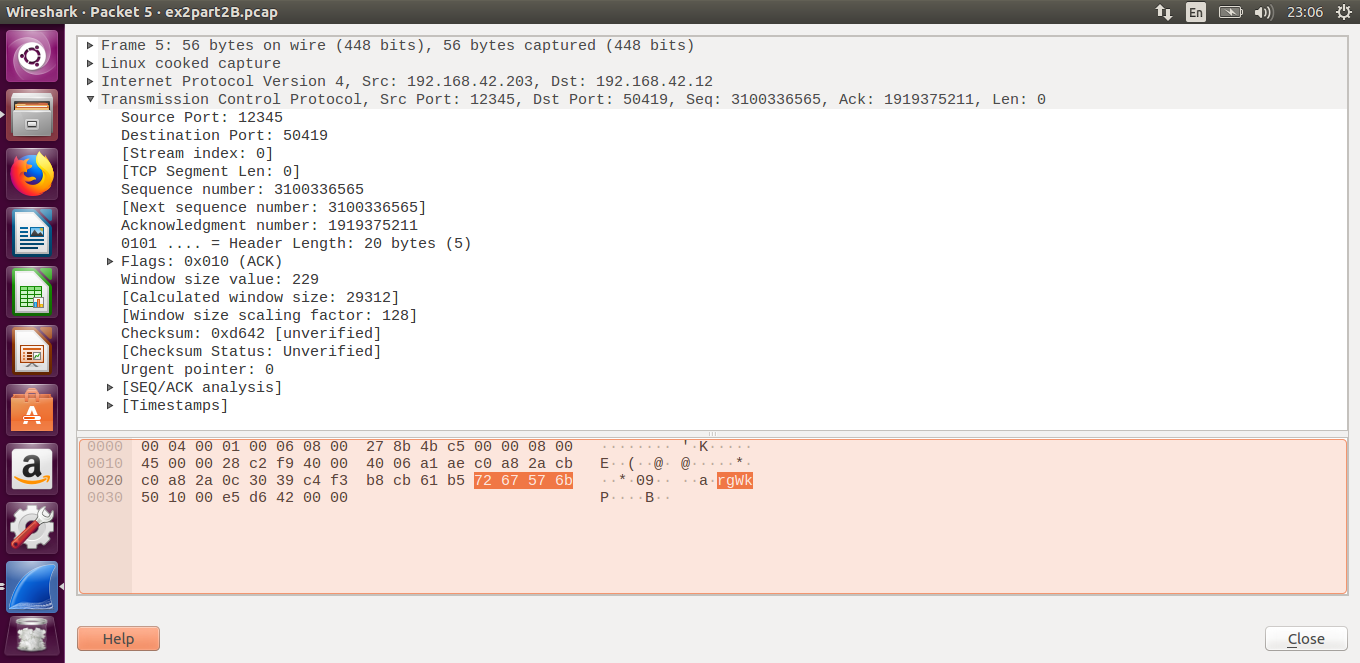
בתהליך התקשורת בין הלקוח לשרת, השרת קיבל פעמיים הודעת AA (שני A רצופים באותה הודעה) ובשאר ההודעות התקבל כל A בנפרד.



כאן רואים את החבילה הראשונה שנשלחה לשרת: AA (ואכן המידע הוא 2 בתים – 4141 שבascii זה AA)

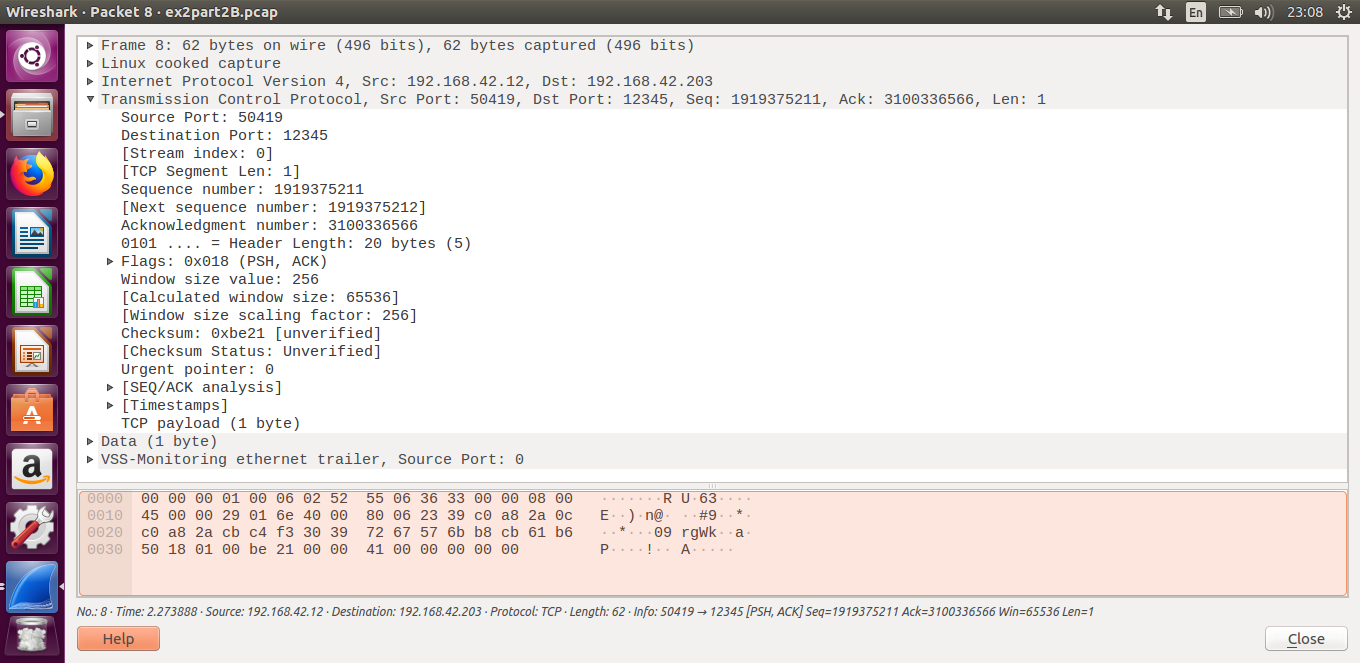


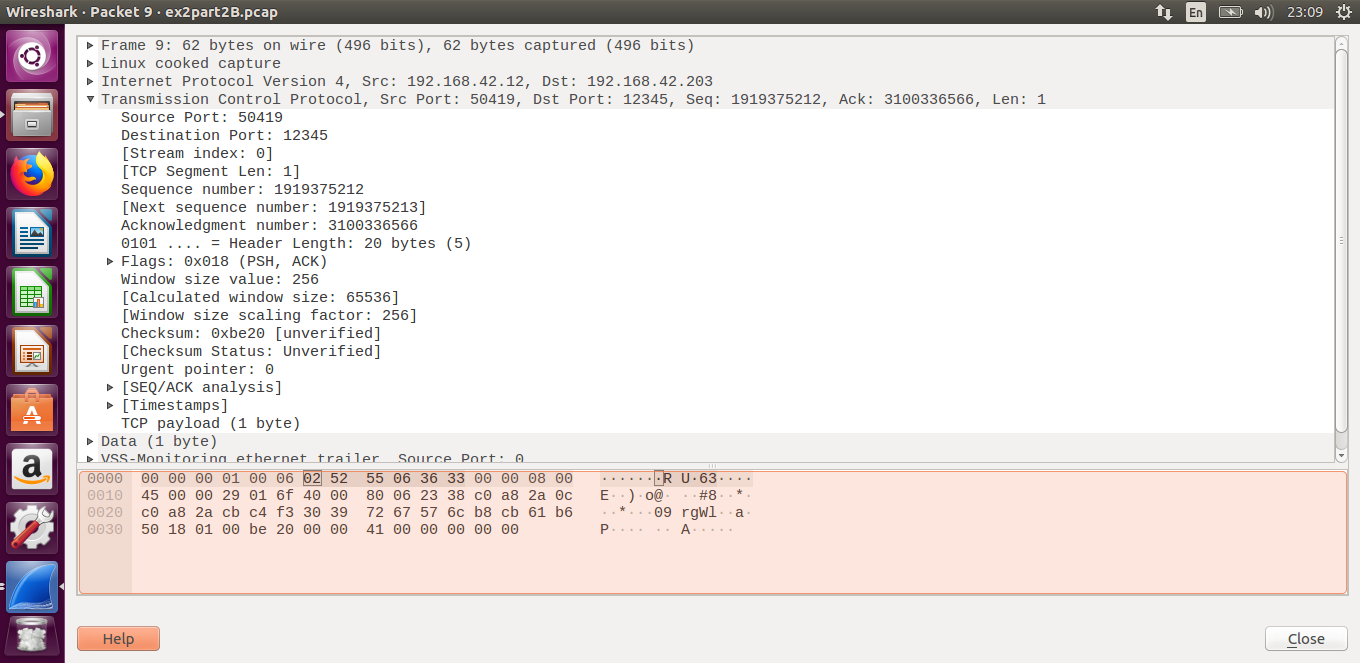
בהודעה זו ניתן לראות את שליחת ההודעה AA מהלקוח לשרת. אנחנו יודעים שבקוד הלקוח ה send מבוצע פעמיים בנפרד, אך כמו שניתן לראות ב data של ההודעה השליחות "התלכדו" להודעה אחת והמידע נשלח הפעם נשלח בהודעה אחת. נשים לב ש ה number seq הוא 1919375209 ,ונסתכל בצילום הבא:



רואים שהשרת שולח ללקוח ACK עם number ack שהוא 199375211 כלומר גדול ב 2 מ ה seq number של הודעת ה AA שאורכה 2 - כלומר ההודעה התקבלה במלואה בשרת, והתקבלה כהודעה אחת כנראה הרי נשלח עליה ACK נפרד.

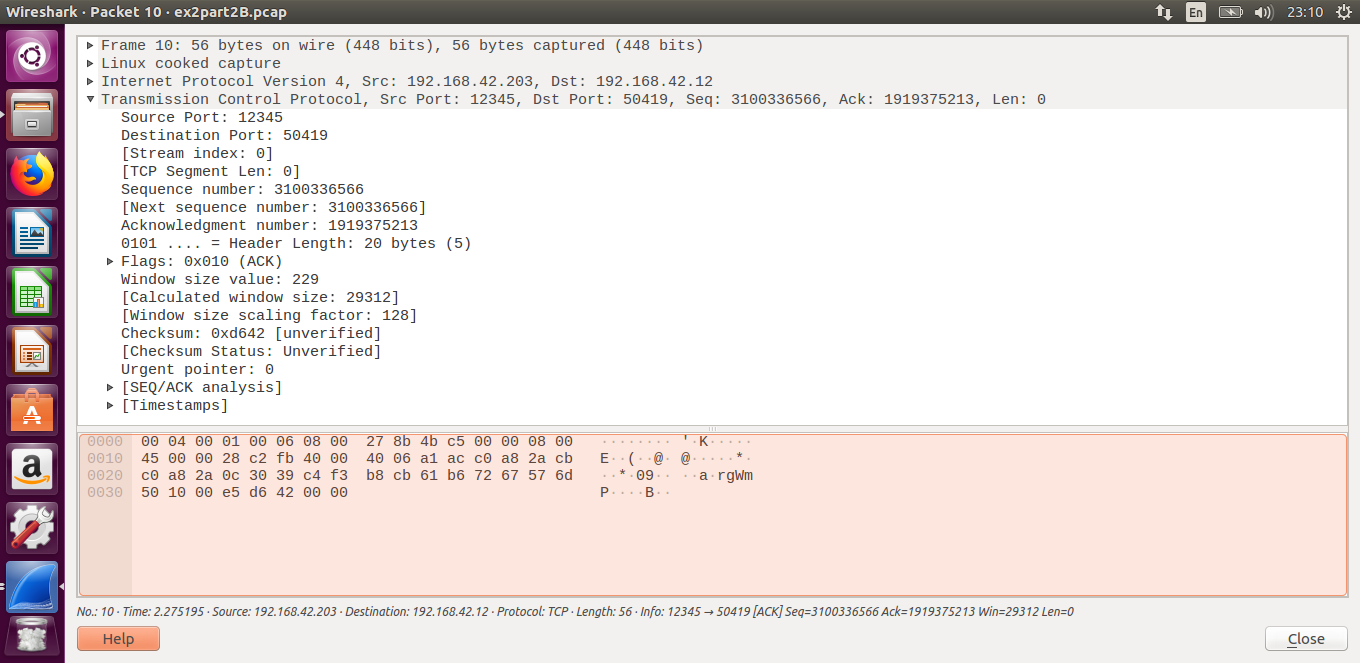
נתבונן בחבילות הבאות שנשלחו מהלקוח לשרת שמכילה גם שני AA.





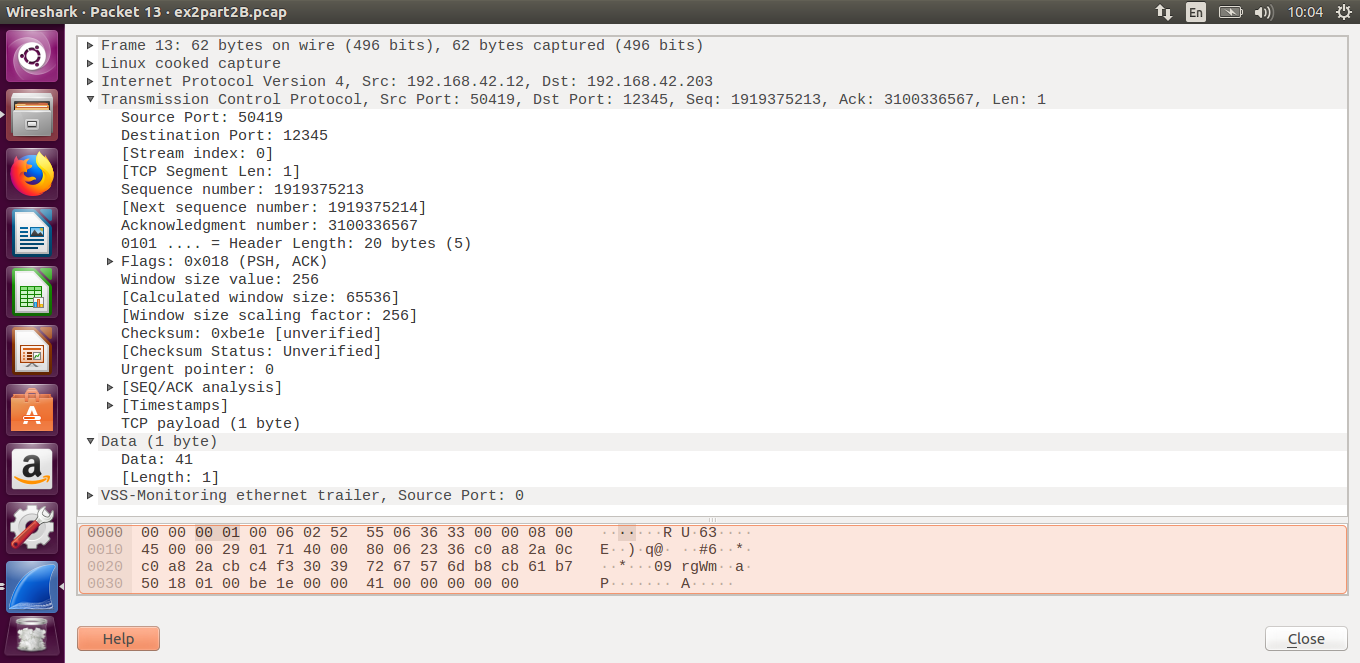
נשים לב שגודל כל הודעה הוא 1, מתקיים שנשלח A מהלקוח לשרת(גודל ה-data הוא 1), גודל הtcp header הוא 20, ה-seq number של ההודעה הראשונה של הלקוח הוא 1919375211 והשנייה 1919375212 והack הוא הseq number של השרת שהוא לא השתנה מהשליחה הקודמת של השרת(כיוון שנשלח ACK).

נתבונן ב-ack של השרת ללקוח:



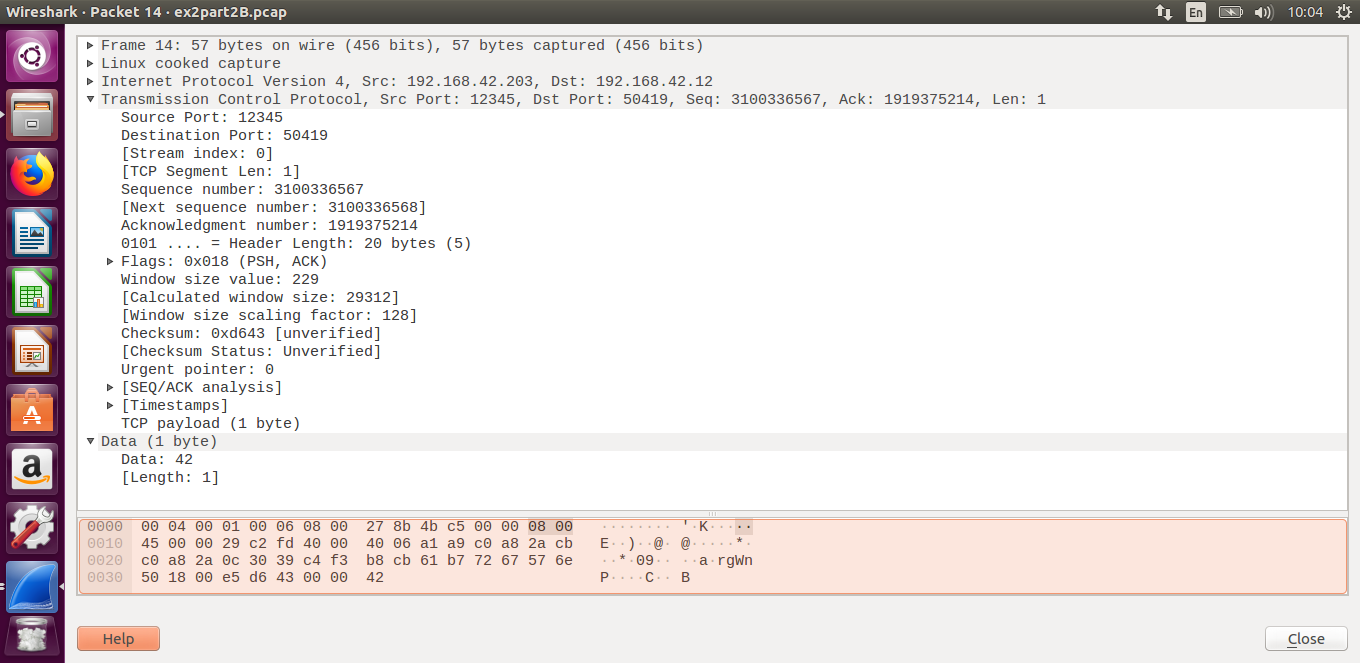
נשים לב שהack הוא 1919375213 , כלומר הseq number של הלקוח גדול ב2 ממקודם , כך השרת מאשר ללקוח שהוא קרא מידע של שני ביטים(AA) , נשים לב שהגודל של הtcp header הוא 20 ודגל הack דלוק.

כעת נתבונן בחבילה שהלקוח שלח לשרת A בודד והשרת תפס אותו כa בודד(שזה למעשה כל החבילות שנשארו).



נשים לב שגודל ההודעה הוא 1, מתקיים שנשלח A מהלקוח לשרת(גודל ה-data הוא 1), גודל הtcp header הוא 20, ה-seq number של ההודעה הראשונה של הלקוח הוא 1919375213 והack הוא הseq number של השרת שהוא לא השתנה מהשליחה הקודמת של השרת(כיוון שנשלח ACK).

נתבונן בack של השרת ללקוח:



נשים לב שהack הוא 1919375214 , כלומר הseq number של הלקוח גדול ב1 ממקודם , כך השרת מאשר ללקוח שהוא קרא מידע של ביט אחד(A) , נשים לב שהגודל של הtcp header הוא 20 ודגל הack דלוק.

כל החבילות הבאות זהות בdata שלהם(כלומר שולחות רק A אחד , חוץ מה2 הראשונות ששולחות 2 A) ולכן אין צורך להראות את כולן , נדגיש כי בהודעה האחרונה הseq number של הלקוח יהיה גדול ב22 ממה שהיה בהתחלה, מה שמוכיח שאכן נשלחו הכל ההודעות ביחד 22 A.

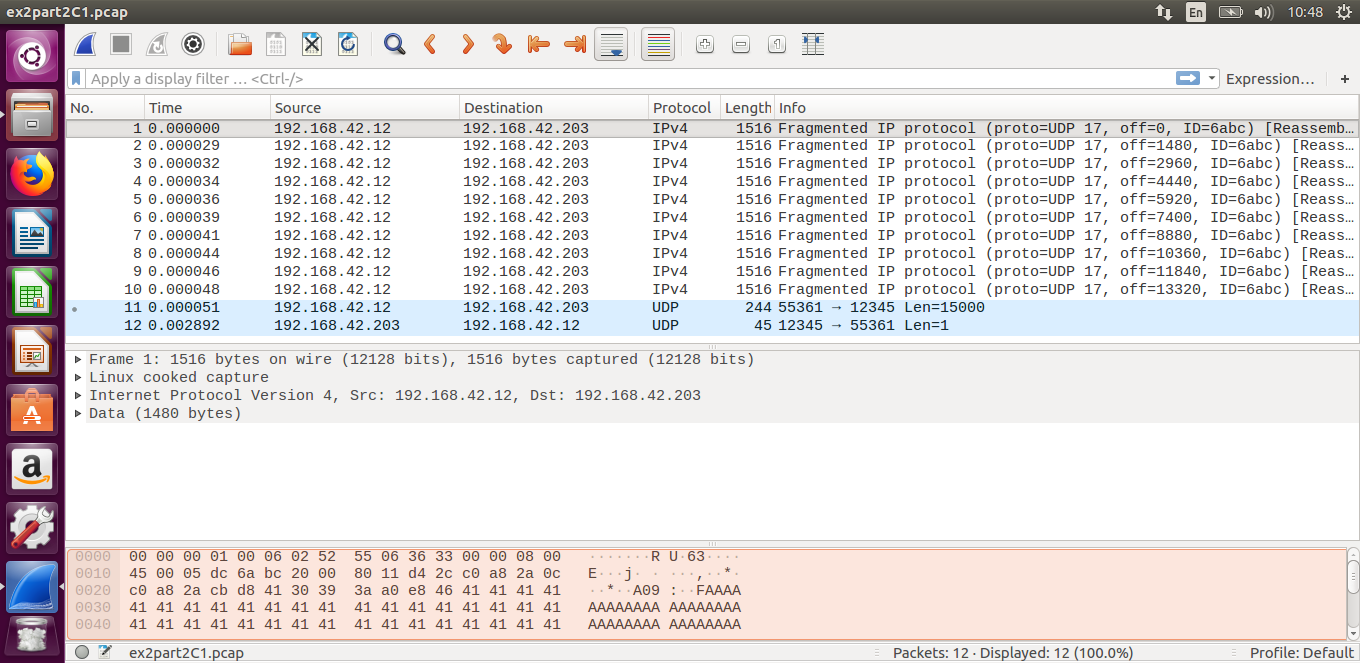
תוצאה זו צפויה להתקבל לא משנה אם שני הודעות send מתלכדות או נשלחות בנפרד שהרי בכל מקרה יישלחו 22 בתים בסופו של דבר.

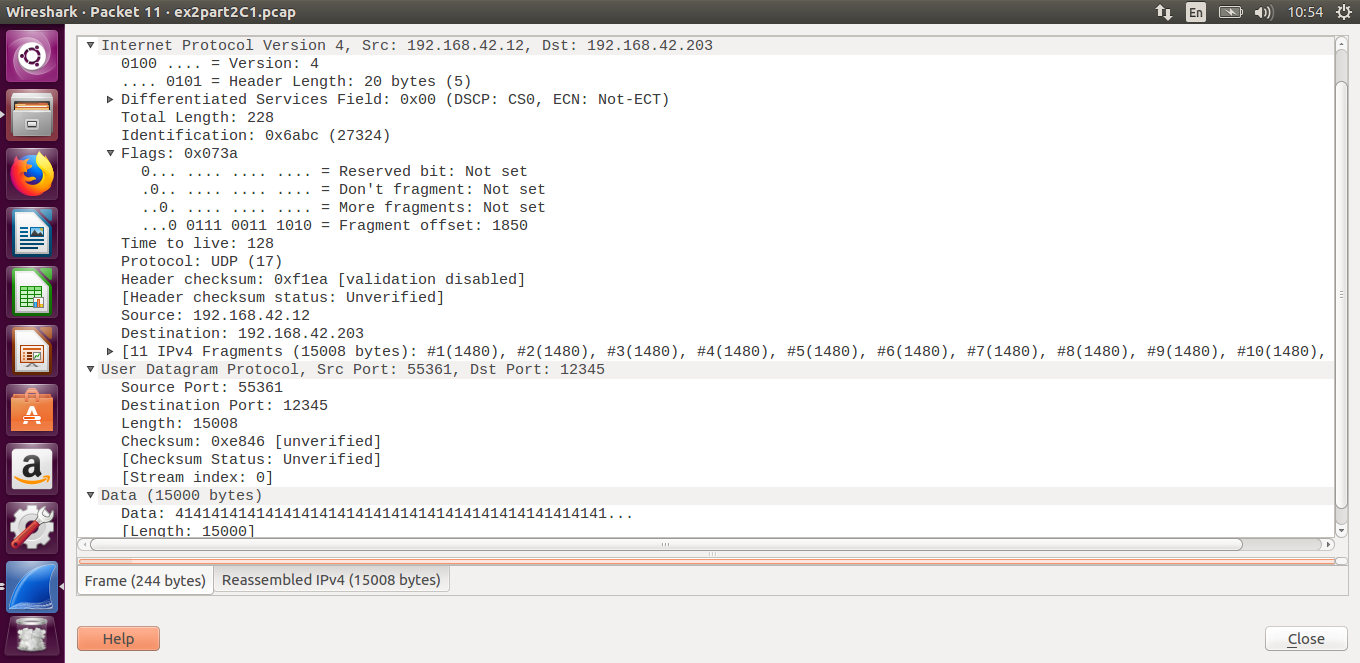
בתשובת השרת ללקוח נשלח פעם אחת B ,ניתן לראות שבהודעה הבאה יחזיר לו הלקוח ACK שערך ה ack number שלו גדול ב 1 מערך ה seq של השרת בהודעה הקודמת כמצופה.

**שאלה 2 סעיף ג':**

**1.**

שליחת 15,000 תווי A מהלקוח לשרת בudp:



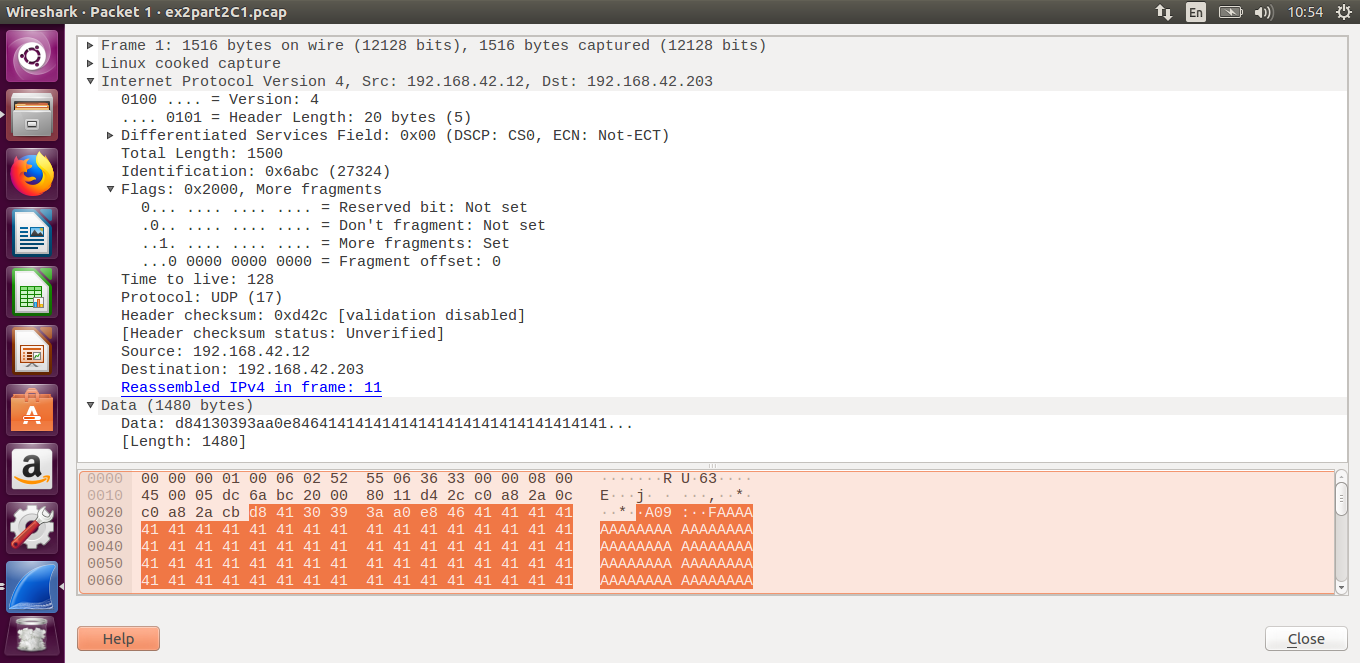


ההודעה המצולמת כאן היא ההודעה האחרונה שהלקוח שלח לשרת.

נשים לב לתהליך שמתרחש כאן, הudp מוריד לשכבת הרשת בקשה לשלוח 15000 בתים של התו A , אפשר לראות בהודעות כי שכבת הרשת עושה פרגמנטציה, כלומר מפרקת את החבילה לפי הmtu, שניתן לראות כאן שהוא 1516.

כיוון שudp אינו פרוטוקול אמין , הוא לא מבצע חלוקה של החבילות.

נסתכל על ההודעה הראשונה שנשלחה:



נשים לב שגודל המידע המקסימלי(MSS) שיכול להישלח לא כולל הheaders הוא 1480(רואים את זה לפי האורך של הdata) , עבורו מתווסף header של שכבת הרשת(20 בתים) וheader של שאר שכבות(16 בתים) וסך הכל גודל של 1516.

נתבונן בכל החבילות חוץ מהחבילה האחרונה כיוון שזו חבילה שנשלחה מהשרת ללקוח , ומה שמעניין אותנו זה החבילות שנשלחו מהלקוח לשרת.

לכל חבילה הinfo כולל fragmented ip protocol, כלומר שהיא חלק מחבילה שעברה פרגמנטציה.

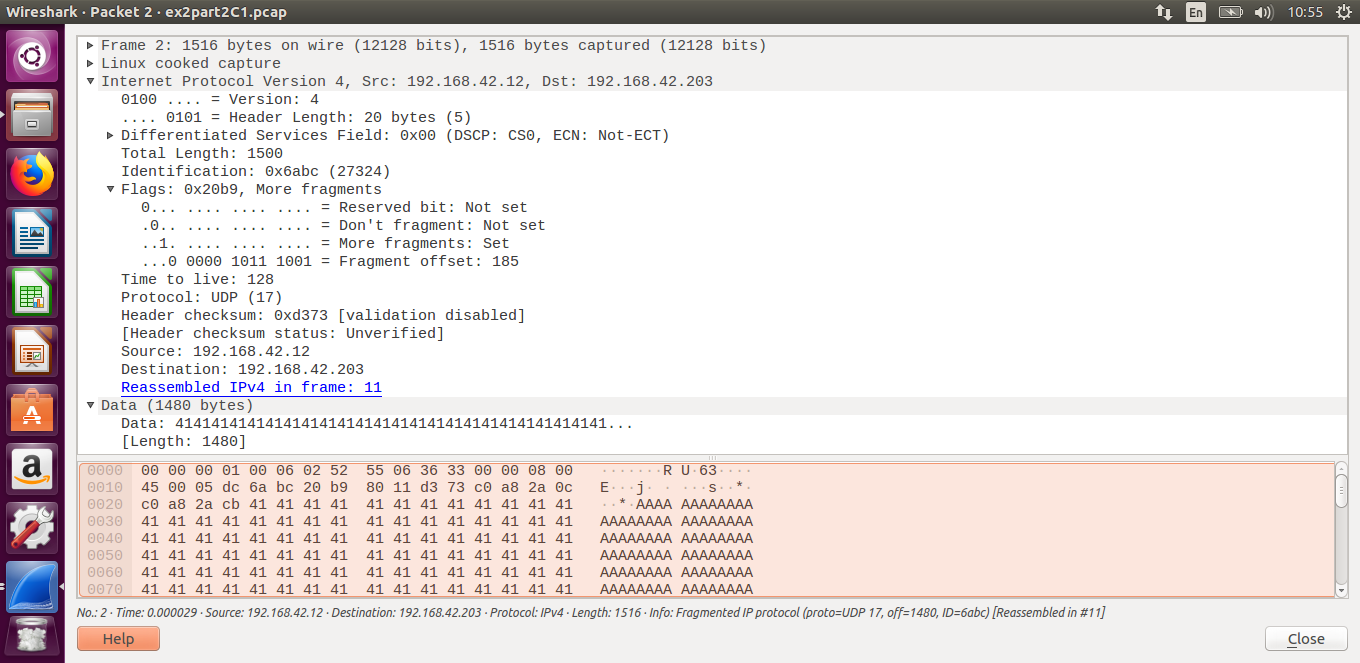
נשים לב שעבור החבילה הראשונה והחבילות שאחריה דלוק הדגל mf(כלומר, בוצעה פרגמנטציה, ויש עוד חבילות אחרי החבילה הזו), נשים לב שבtcp היה דלוק הדגל df בשכבת הרשת.

בכל החבילות חוץ מחבילה אחת לא יהיה header של udp, כיוון ששכבת הרשת מבצעת פרגמנטציה, לא משנה לה איך המידע שהיא קיבלה הגיע, היא סך הכל מחלקת אותו לחבילות בגדלים המתאימים ומוסיפה את ה-headers הרלוונטים שלה, ולכן קיים רק header בודד של udp בכל החבילות שנשלחו.

נשים לב שהidentification(שדה שחשוב בפרגמנטציה על מנת שנוכל לדעת שכל החבילות קשורות אחת לשנייה) הוא זהה בכל החבילות והוא 0x6abc.

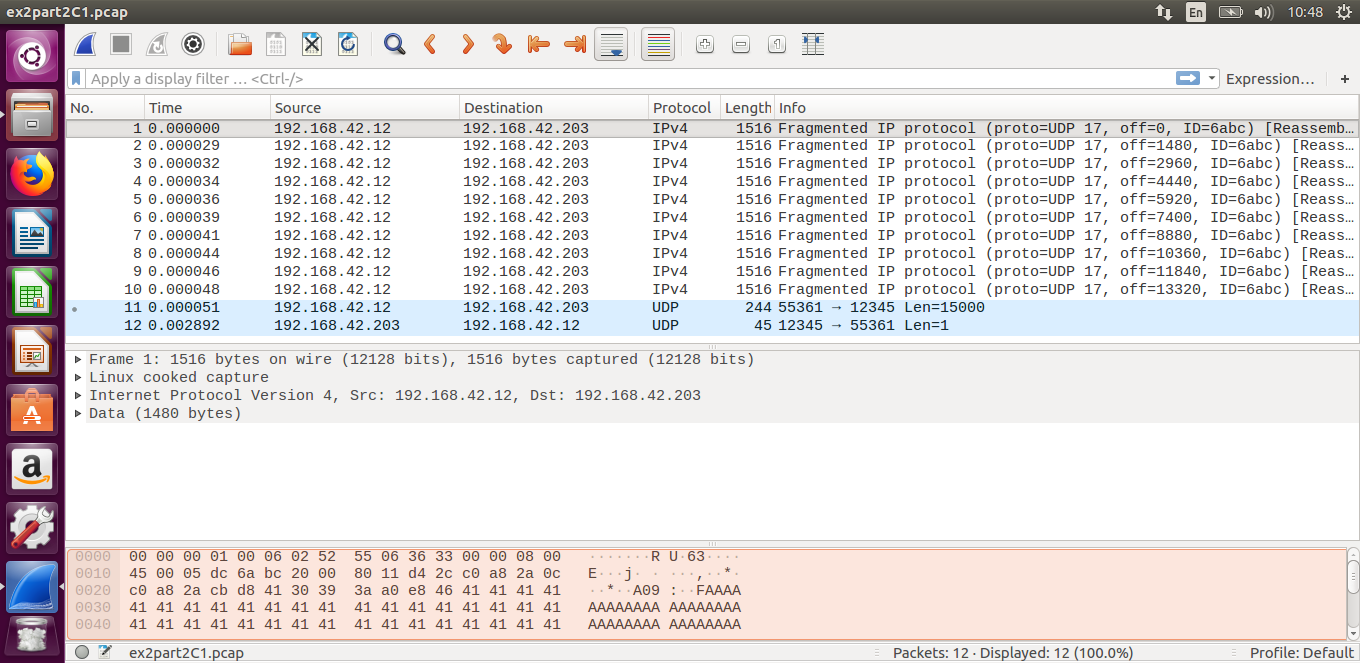
נשים לב כי הoffset של החבילה הראשונה הוא 0.

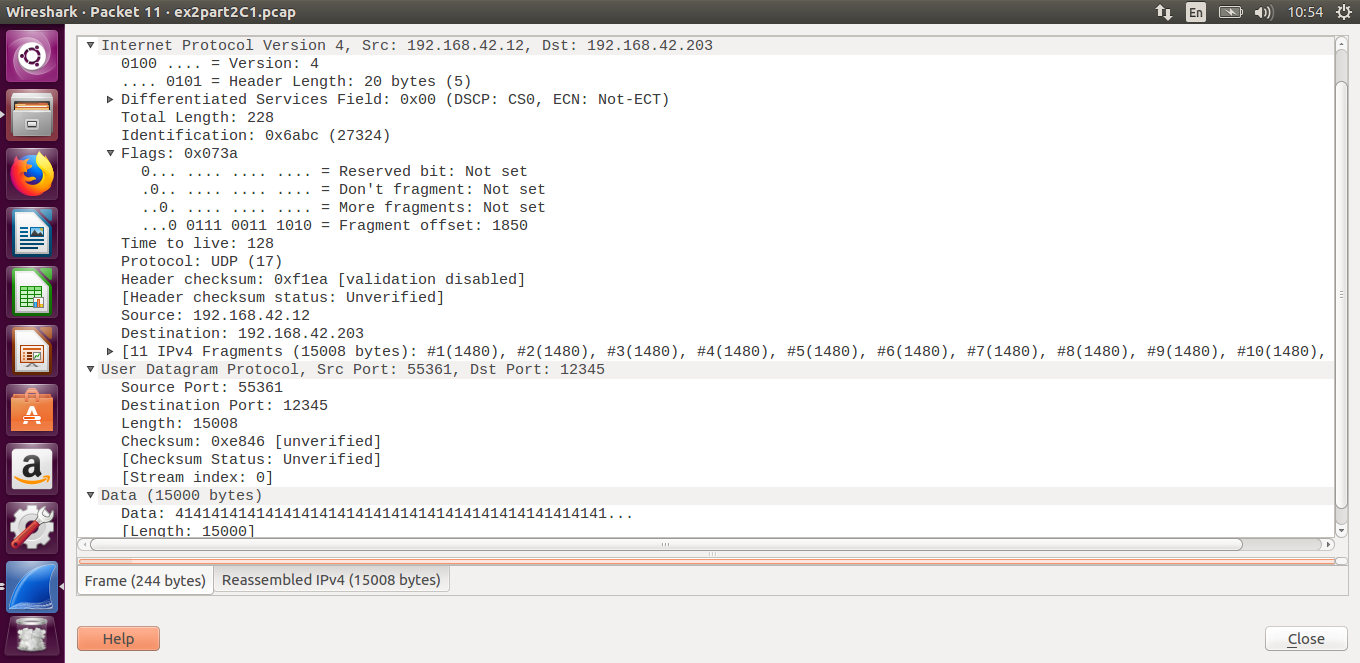
**כעת נתבונן בחבילה השנייה שנשלחה:**



נשים לב שכאן ה-offset הוא 185 , ז"א המידע מחולק ל8 הודעות ולכן זה בעצם 8\*185=1480 , שכן החבילה הראשונה שנשלחה כוללת 1480 bytes מהחבילה המקורית, והחבילה השנייה תתחיל עם המידע שנמצא אחרי אותם 1480 בתים , נשים לב שהidentification כאן זהה להודעה הקודמת וגם כאן דלוק דגל הmf.

**כעת נתבונן בחבילה האחרונה שהלקוח שלח אל השרת:**

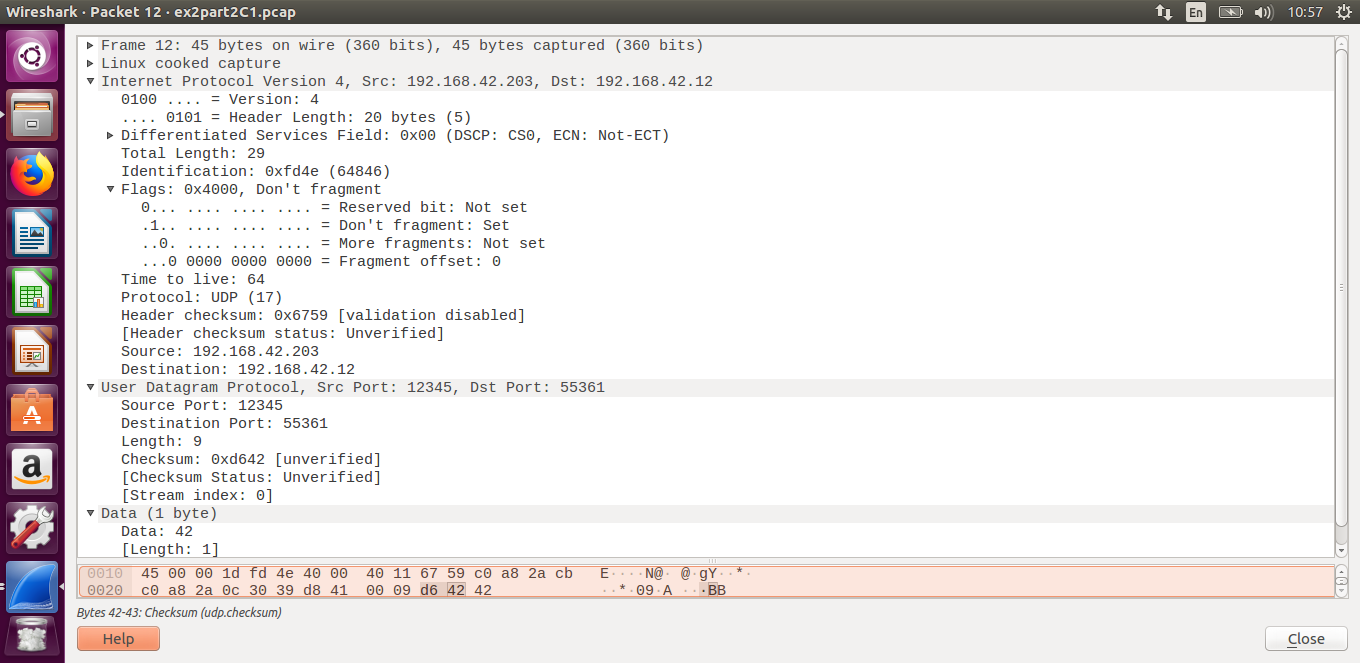


****

גודל החבילה צריך להיות 15000 פחות 1480 \* מספר החבילות שנשלחו , מחישוב פשוט אנו יודעים כי זה שווה ל200 , וכיוון שיש גם את הudp header, ואת שאר הheaders שביחד הם 44 בתים נקבל שגודל החבילה הזו הוא 244 בתים.

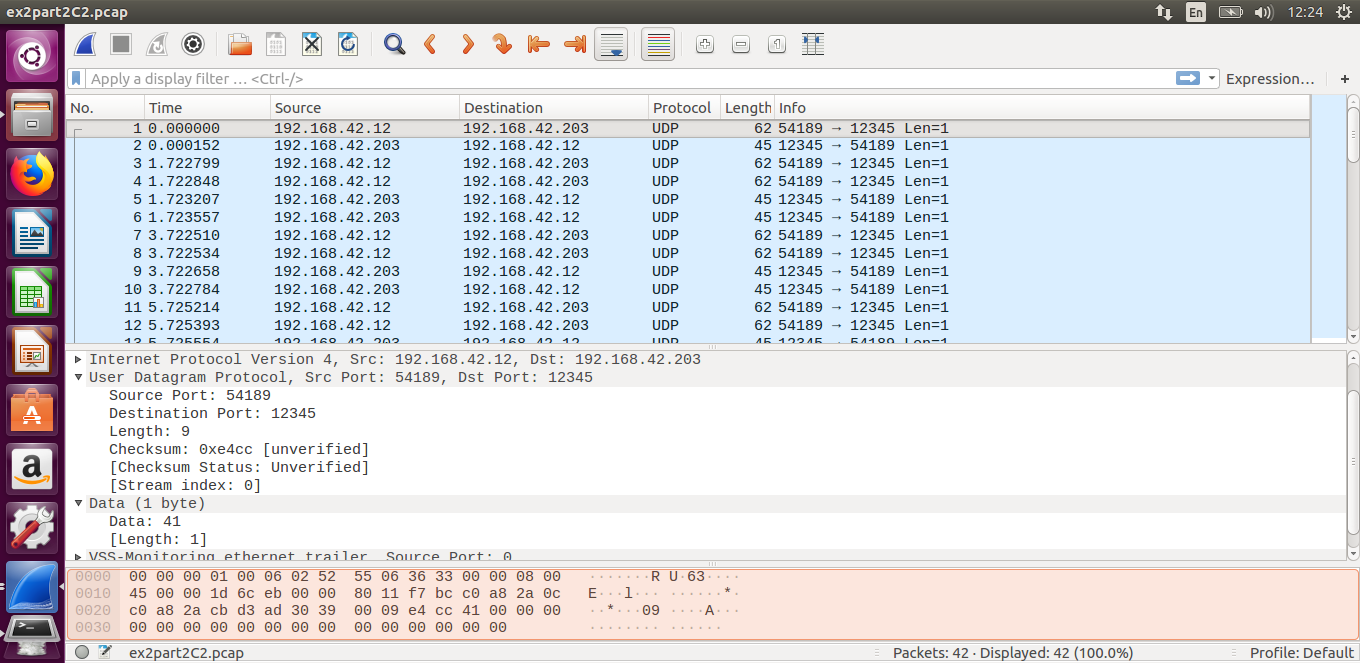
נשים לב שהidentification כאן זהה להודעות הקודמות ודגל הmf הוא 0 כיוון שזו החבילה האחרונה שנשלחה מהלקוח לשרת בתהליך הפרגמנטציה הנ"ל, אפשר לראות שהמידע בשכבת הudp הוא מאוד מצומצם , כיוון שהיא נותנת לשכבת הרשת למעשה לעשות את רוב העבודה.

**נתבונן בחבילה(היחידה) שהשרת שלח ללקוח:**

**ניתן לזהות כי זו ההודעה שהסרבר שלח ללקוח ע"פ הsource port שהוא 12345.**

נשים לב שלחבילה זו אין דגלים בחבילת הרשת שקשורים לפרגמנטציה(כיוון שהיא לא צריכה לעבור פרגמנטציה), וכאן נעביר רק ביט אחד של מידע בתוספת הheaders משאר השכבות ונגיע ל45 בתים של מידע בסך הכל.

**חלק ב' של סעיף ג':**

****

נשים לב שבמקרה זה כל ההודעות של תווי A יתקבלו בנפרד (גודל כל הודעה כאן היא 1) , במקרה של TCP ראינו לפעמים שתי הודעות A רצופות שמתחברות להודעת AA אחת.

ידוע לנו שtcp מעביר את המידע כרצף של בתים , זאת אומרת שהוא לא מודע לגבולות של הודעה, הוא יודע רק שיש לו רצף של בתים שעליו להעביר הלאה בסדר הזה.

כאשר עשינו ב-tcp שני send רצופים, אז הבתים שהבאפר היה צריך להעביר נכנסו אליו ביחד שני ה aהאלו ו-tcp כמו שהואו יודע העביר אותם כbyte stream.

כלומר , tcp לא הספיק לשלוח את הa הראשון לפני שהשני כבר נכנס, וtcp כאמור שולח את כל הבתים שיש לו בבאפר בלי מודעות למאיפה הגיעה ההודעה ולכן לפעמים(כאשר tcp לא הספיק לשלוח את הa הראשון לפני השני נכנס) הם יתמזגו להודעה אחת(הרי בבאפר ישב המידע AA).

לעומת זאת , בudp אין את אותו רצף בתים שמועברים מקצה לקצה בudp תמיד כתבנו:

Socket.sendto(msg,(ip\_dest,port\_dest))

כלומר את ההודעה הוא היה שולח מיד ליעד, ולא בהתבסס על החיבור ביניהם.

לכן כל שליחת הודעה הייתה בלתי תלויה בשליחות ההודעות האחרות, ולכן כל הודעת A נשלחה בפני עצמה והתקבלה אצל השרת בנפרד.