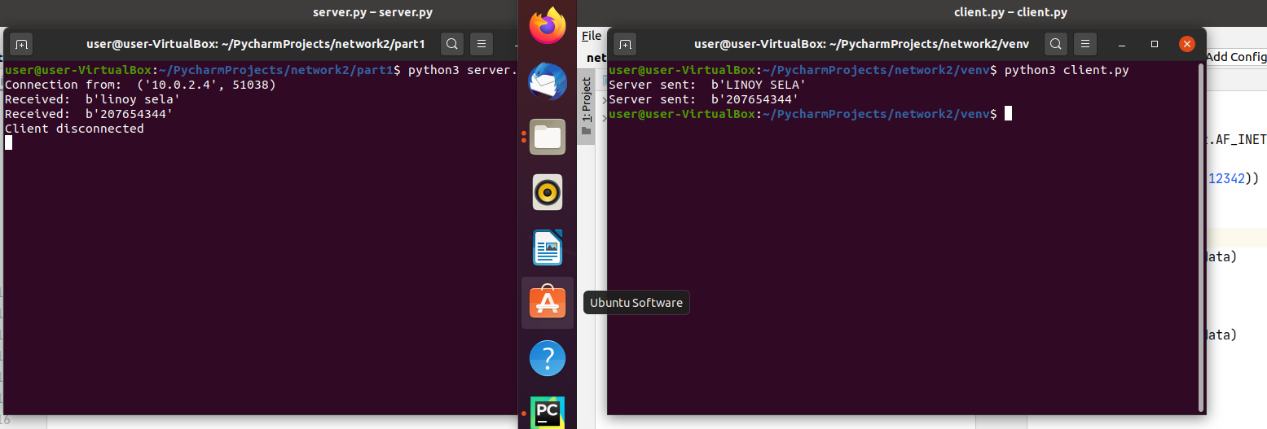
**רשתות תרגיל 2 – חלק 1**

**חלק א'**

ראשית נריץ את השרת והלקוח על מנת לקבל את השם באותיות גדולות ואת התעודת הזהות, בלקוח.

הפעלנו 2 מכונות כאשר מצאנו את הip של השרת עם הפקודה ifconfig, וגילנו שכתובותו היא 10.0.2.15, לכן בלקוח כאשר ניצור socket, נרצה להגיע ל Ip של השרת. כתובת הip של הלקוח היא 10.0.2.4 וניתן לגלות זאת גם ע"י הדפסת השרת שמדפיס מאיפה הגיע הבקשה. הport שהלקוח מעוניין להאזין לו הוא 12342.

ניתן לראות כי התוכנית אכן עובדת כרצוי:

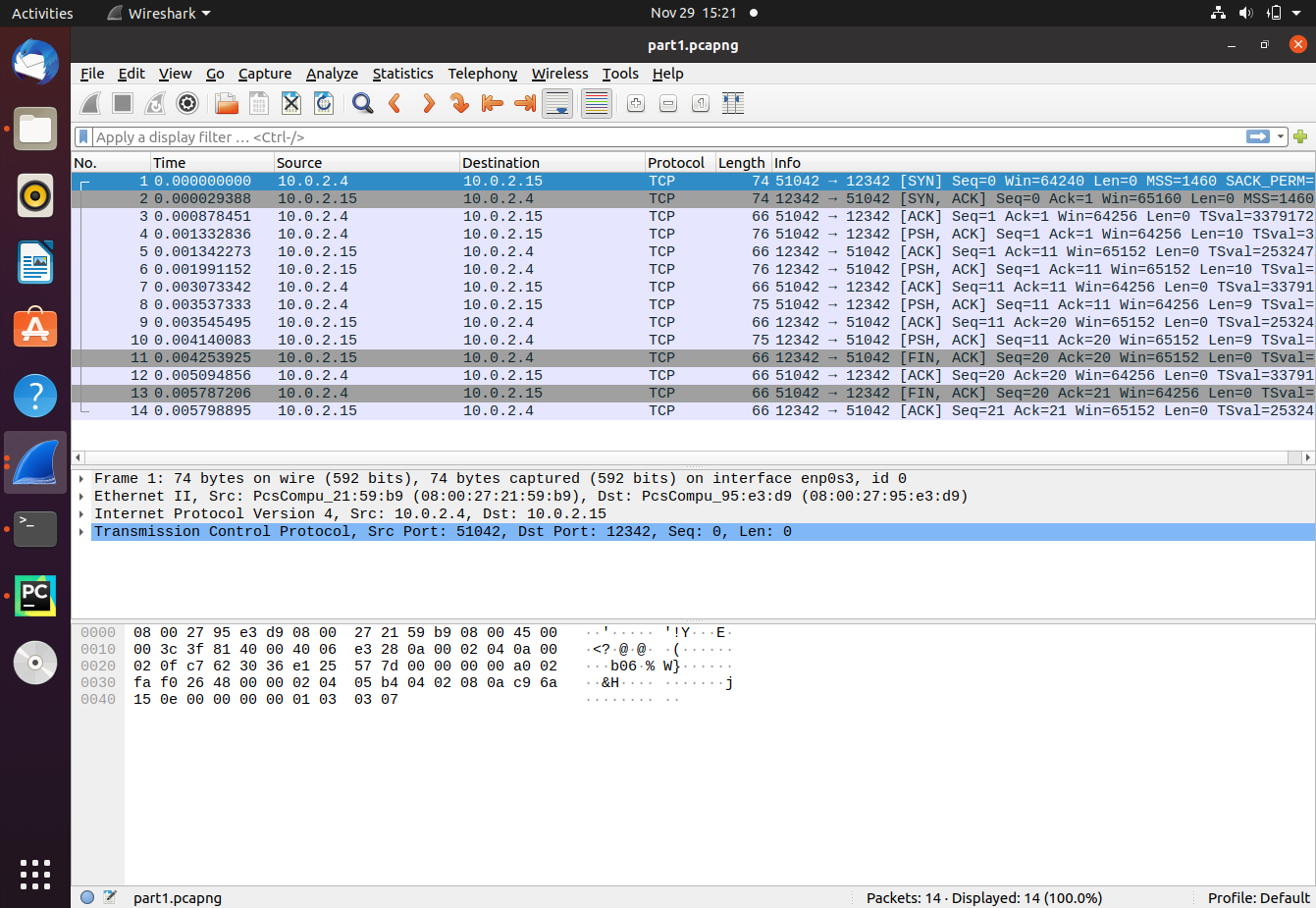


נריץ את התוכנית שוב וננתח את הנתונים:

נפעיל whireshark ונסנן את החיפוש ע"י כתובות הip של המקור ושל היעד ע"י:

((ip.dst == 10.0.2.4) && (ip.src==10.0.2.15)) || ((ip.src == 10.0.2.4) && (ip.dst==10.0.2.15))

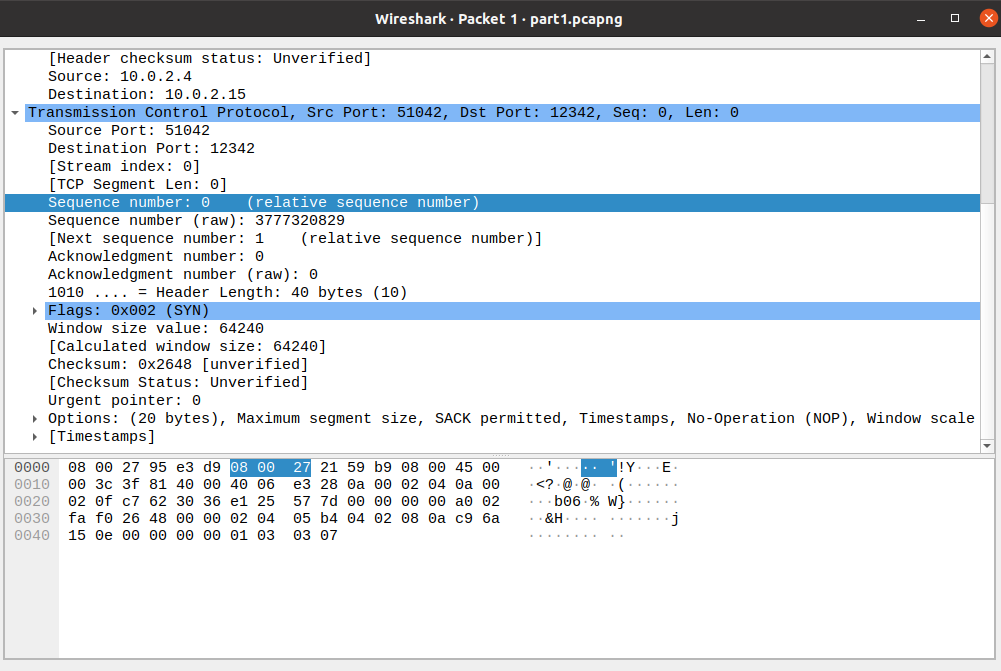
סינון זה שמור בpart1.

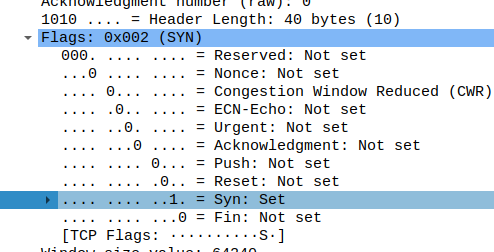


מכיוון שאנו עובדים עם פרוטוקול TCP, ראשית יש לבצע חיבור בין 2 המכונות, השלב הראשון הוא שלב לחיצת ידיים המתבטא בהקמת החיבור שנראה זאת ע"י הדגל SYN, יחד עם מספר סידורי, היעד ישלח גם SYN יחד עם ACK , כלומר אישר את הקמת החיבור מבחינתו – לצורך סנכרון, וישלח גם את הack number יחד עם מספר סידורי.

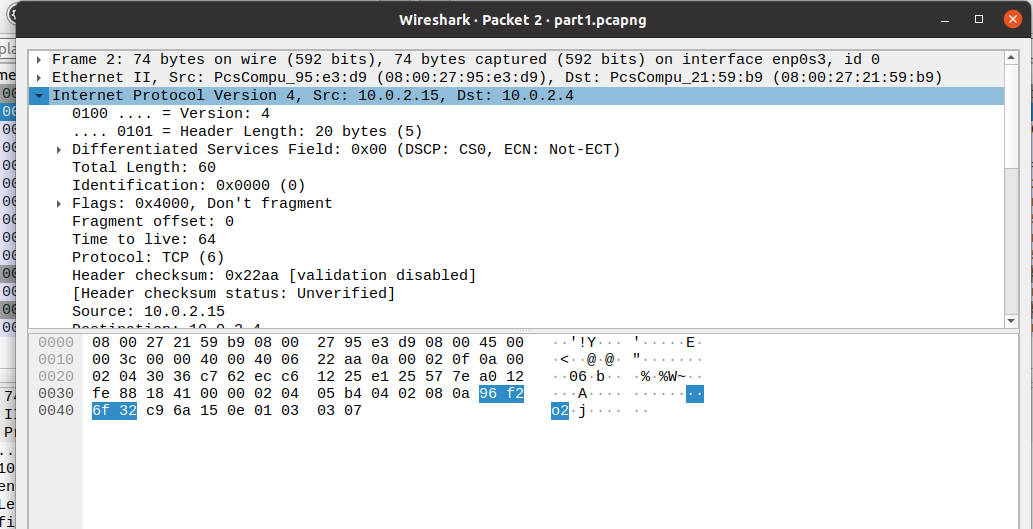
נראה זאת בwhireshark:

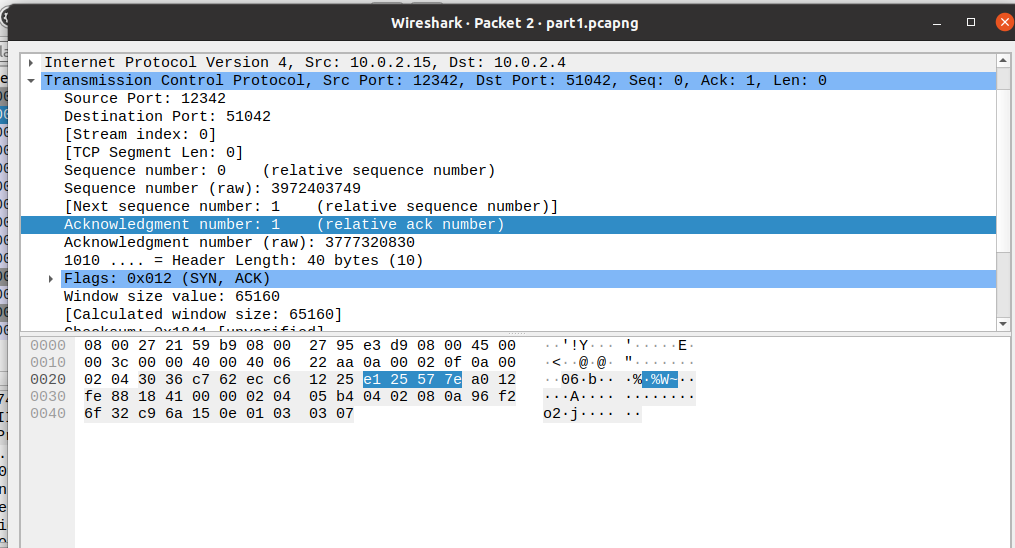
בחבילה הראשונה של התקשורת, port המקור הוא 51042, כלומר הלקוח, וport השרת הוא 12342, כלומר השרת. תחת שכבת התעבורה ניתן לראות כי דגל הSYN דלוק, המספר הסידורי (נסתכל על האמיתיים ולא היחסי) הוא 3777320829 והack number גם 0.

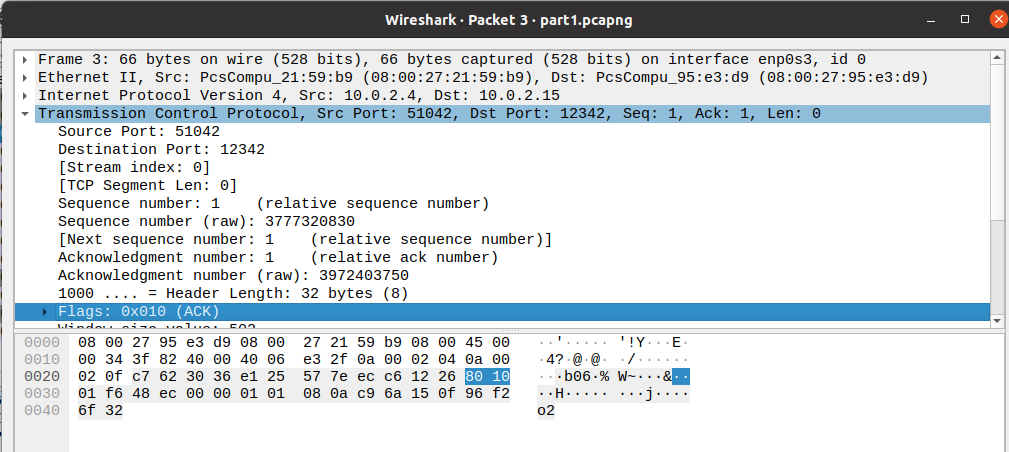




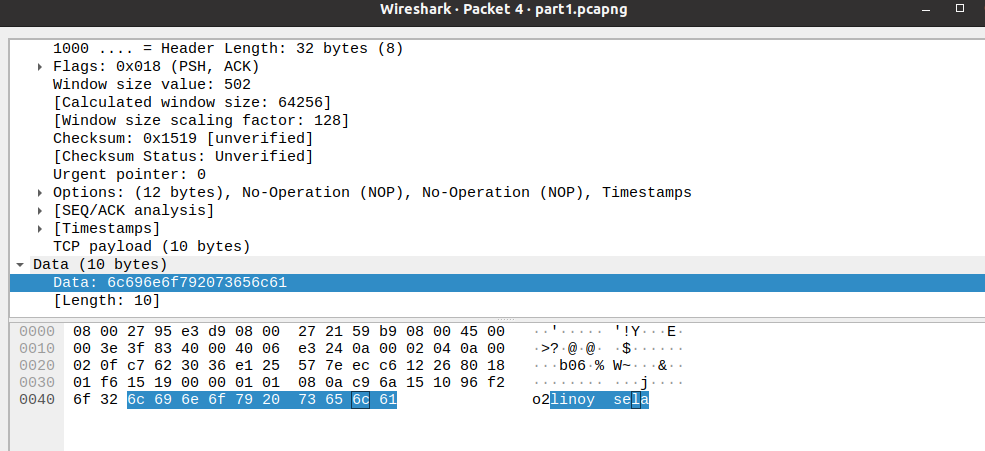
בחבילה ההבאה, ניתן לראות לדוגמה על פי שכבת הרשת כי ip המקור הוא של השרת 10.0.2.15, כלומר הוא משיב כעת לבקשת הלקוח ליצור חיבור, נראה זאת בפירוט בשכבת התעבורה. כעת ניתן לראות כי הדגלים הדלוקים הם SYN על הרצון לסנכרון הקמת החיבור מצד השרת, ודגל ACK על מנת לאשר את החבילה הקודמת. ניתן לראות כי המספר הסידורי של השרת הוא 3972403749, בחר אותו אקראי, אך הACK הוא 3777320830 מכיוון שהמספר הסידורי של הלקוח היה 3777320829 + 1 עבור הACK – יחד 3777320830.  
נראת זאת:





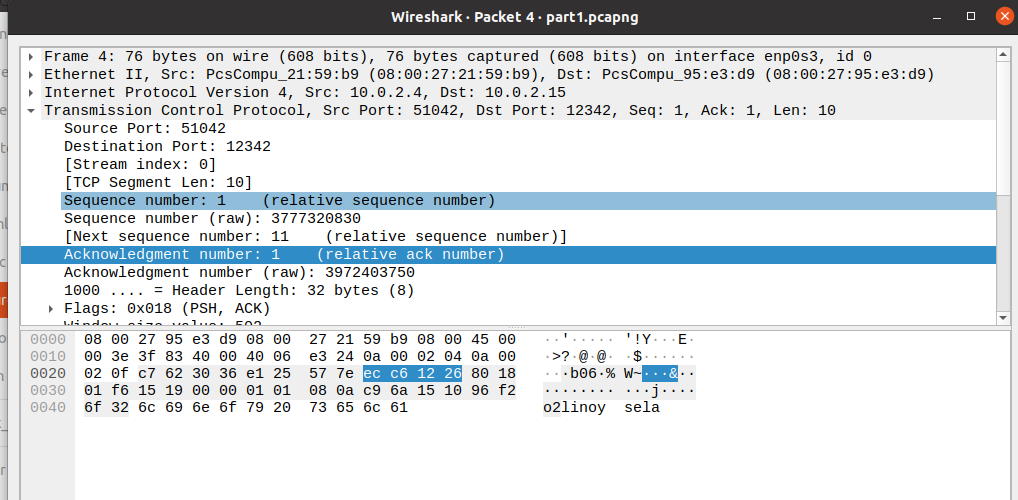
החבילה ההבאה היא שליחת ack על ידי הלקוח, מאשר את זאת שהחיבור קם וקיים. לכן רואים כי הack הוא 3972403750 (מכיוון שהמספר הסידורי היה 3972403749 + 1 לACK), והמספר הסידורי הפך להיות 3777320830 , בהתאם לACK הקודם שנשלח והיה זהה לכך, הוא לא גדל כי לא נוסף מידע.

עד כאן עבור שלב הקמת החיבור, כעת נעבור לשלב מימוש החיבור, כלומר שליחת מידע.  
לפי התוכנית, ראשית הלקוח שולח לשרת את השם linoy sela ואכן ניתן לראות זאת בdata:

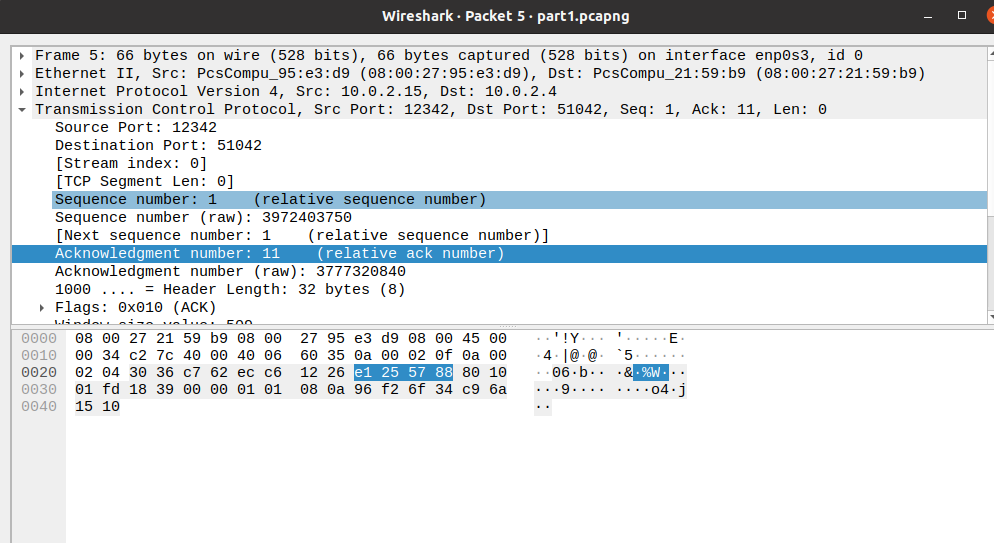


בשכבת התעבורה של חיבור זה ניתן לראות כי הack שווה ל3972403750, לא התקבל מידע חדש לכן נשאר כך, ולכן גם המספר הסידורי שווה ל3777320830, מכייון שהוא שולח מידע "מהמקום" הזה והלאה.

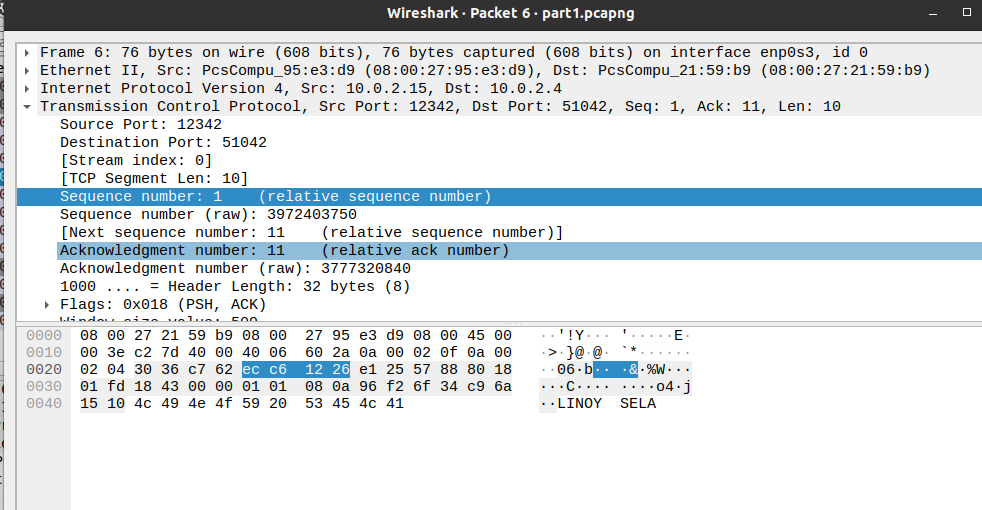
(אם נסתכל על הACK היחסי, ניתן לשים לב כי הnext seq number הוא 11, וזאת מכייוון שהוספנו את כמות הביטים של השם לינוי סלה למספר הסידורי שאנו נמצאים עליו כעת(1), כלומר דיווח על כך שבפעם הבאה אני אהיה פנוי מ11. ניתן גם לראות בחץ הכחול כי גודל המידע שנשלח הוא 10, אכן ככמות האותיות שבשם יחד עם רווח.)

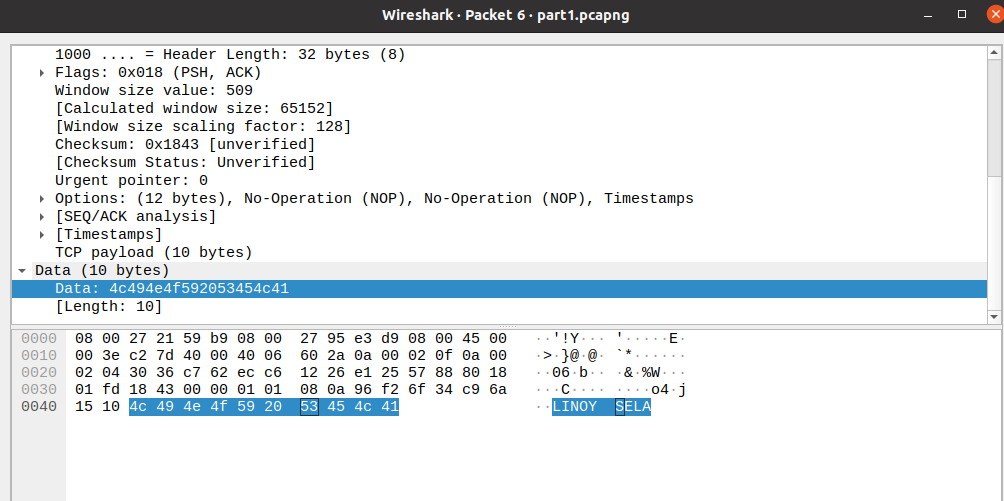


בחבילה הבאה, נצפה שהשרת יתן על אישור על כך שקיבל את החבילה הקודמת עם השם, כלומר ישלח עליה ACK, לפי חישובנו עליו להיות המספר הסידורי הקודם 3777320830 + גודל המידע - פלוס 10 תווים של השם. סה"כ 3777320840, נראה אם זה מתקיים:

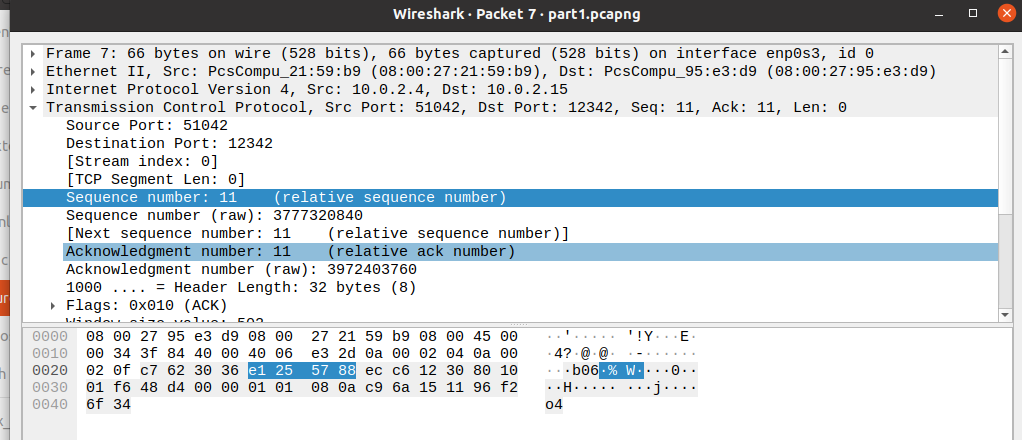


אכן ניתן לראות כי חבילה זו מגיעה מהשרת ע"פ הport השולח שהוא 12342, אכן דגל הACK דלוק והוא שולח ACK 3777320840, הוא לא שולח מידע נוסף לכן האורך len הוא 0, והמספר הסידורי הוא 3972403750, זהה לACK הקודם. כעת, לאחר שהשרת אישר את קבלת השם, לפי הקוד עליו לבצע פעולה והיא להחזיר את השם באותיות גדולות. לכן נצפה בחבילה הבאה שהשרת ישלח מידע שהוא LINOY SELA, נראה זאת:

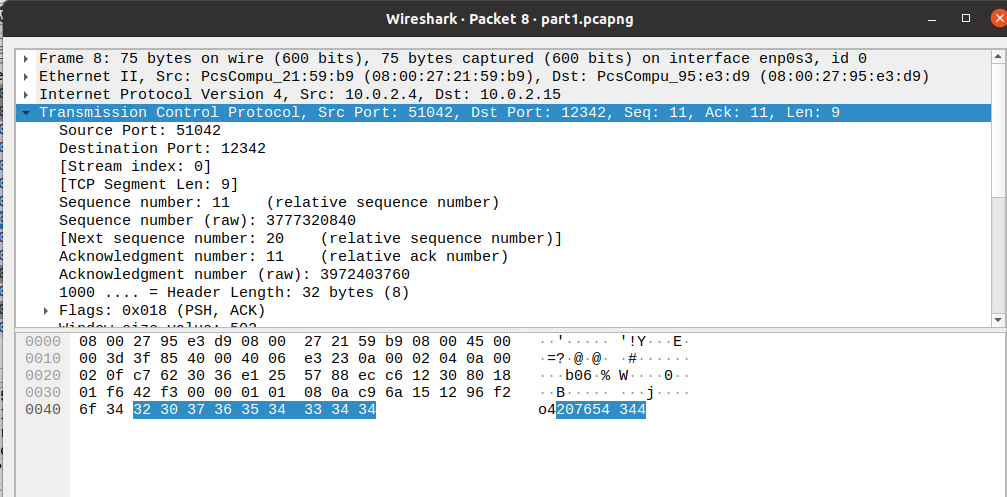




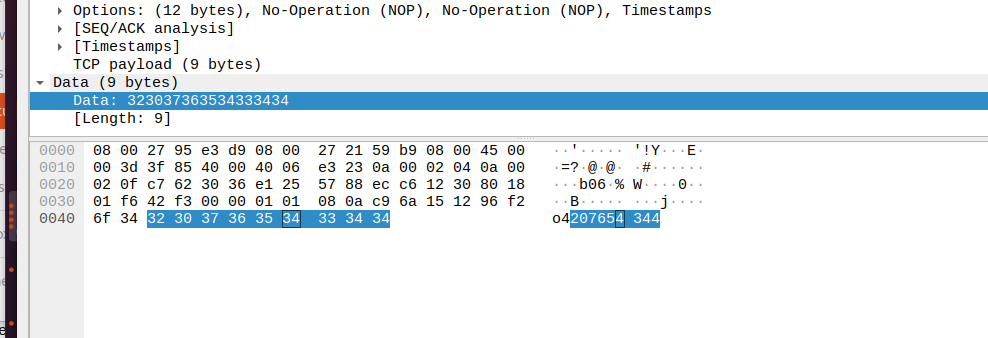
אכן השרת שולח מידע ללקוח, שהוא LINOY SELA, המיר את השם לאותיות גדולות, מידע באורך 10 תווים=בתים. המספר הסידורי שלו הוא 3972403750 והack נשאר 3777320840 כי הוא לא קיבל מידע חדש.

לאחר מכן נצפה שתגיע חבילה מהלקוח, עליו לקבל מידע זה ולאשרו, ואכן אנו רואים כי הוא שולח ACK 3972403750 כי המספר הסידורי היה3972403750 + 10 על המידע החדש , את השם באותיות גדולות, סהכ 3972403760, והמספר הסידורי שלו 3777320840 בדומה לACK הקודם(לא שולח מידע חדש- נשאר זהה).

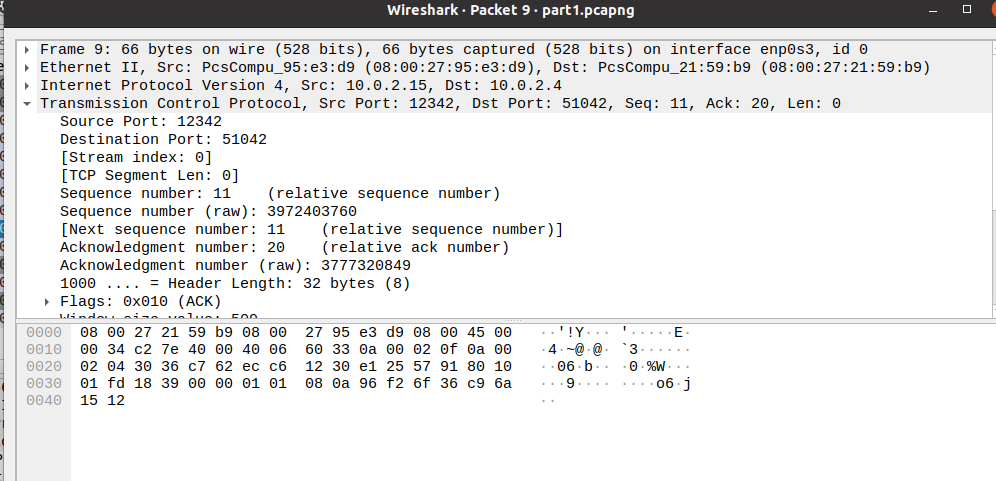
כעת, יתבצע אותו תהליך שתיארנו עבור שליחת שם ללקוח וקבלתו באותיות גדולות, הפעם הלקוח ישלך ת"ז ויקבלו בחזרה.



ואכן השרת שולח את הת"ז, מידע באורך 9 בתים, יחד עם ACK 3972403760 ומספר סידורי 3777320840, המספרים האחרונים שעליהם היה, ממשיך מהם והלאה. נראה את המידע בשכבת האפליקציה:

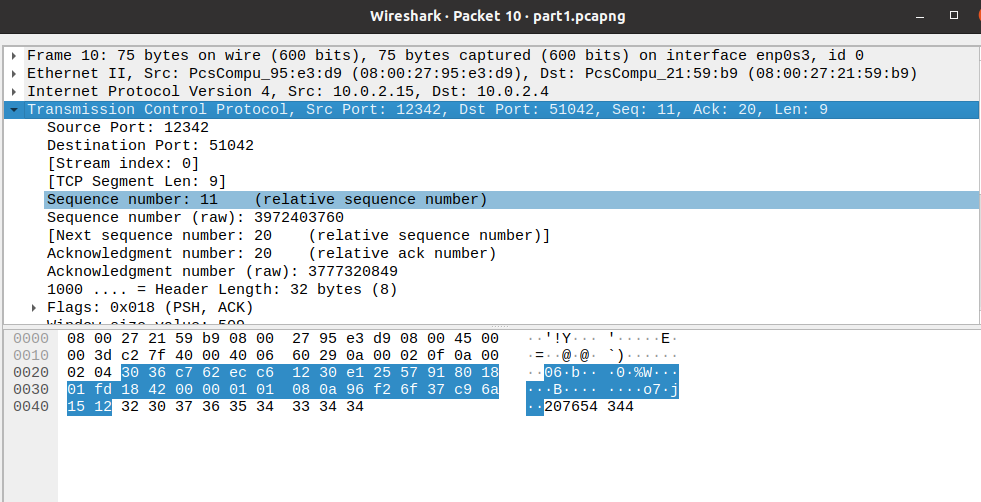


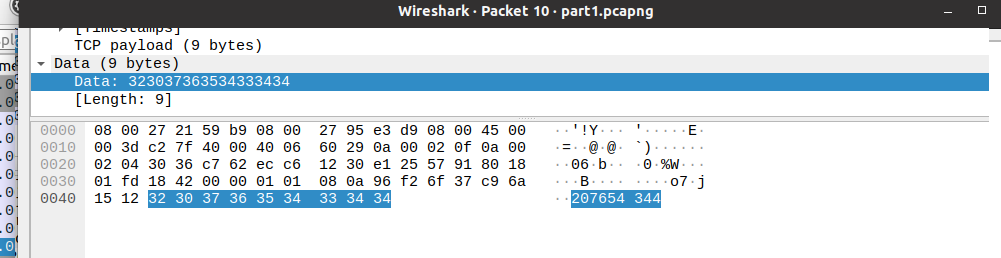
כעת נצפה לקבל ACK על חבילה זו מצד הלקוח:



ואכן, השרת מאשר את קבלת המידע על ידיACK 3777320849, המספר הסידורי הקודם+9בתים על המידע החדש. בחבילה זו הוא לא מקבל מידע לכן האורך הוא 0, והמספר הסידורי הוא 3972403760, בדומה לACK האחרון, ל נשלח מיד חדש לכן נשאר זהה.

כעת, עליו להחזיר את המידע ללקוח, להחזיר את הת"ז, נראה שאכן הוא שולח מידע בגודל 9 בתים, עם ACK 3777320849 ומספר סידורי 3972403760 - כמו במקרה האחרון בו הוא רק שלח ACK.



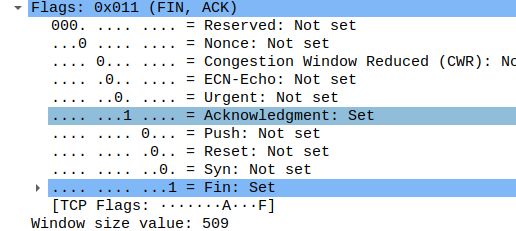


לאחר השרת שלח את המידע האחרון שהיה לו לשלוח ללקוח, כעת, הוא יכריז על כך שאין לו יותר מידע להעביר ולכן מבחינתו ניתן לסיים קשר זה. לכן השרת ישלח ללקוח FIN, כמוש ניתן לראות בתצלומים יחד עם ACK:

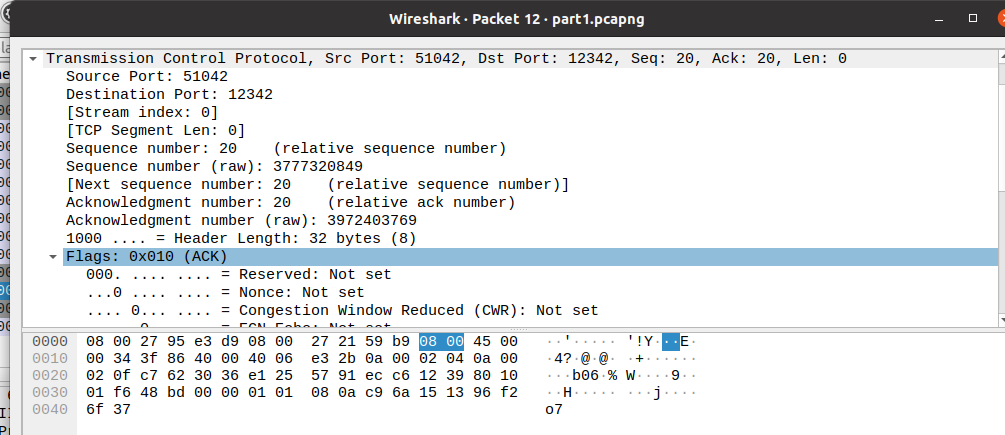


נשים לב כי כעת המספר הסידורי עלה ל3972403769 מכיוון שהשרת שלח חבילה קודמת בגודל 9 בתים, המספר הסידורי שלו יהיה 3972403760, המספר הקודם, ועוד 9 – הבתים שקיבל – כלומר סה"כ 3972403769. בנוסף, הACK שלו עדיין 3777320849 כי הוא לא שולח מידע חדש כאן.

החידוש כאן הוא דגל הFIN (שמצהיר על כוונת סיום הקשר) שמודלק יחד עם הACK:

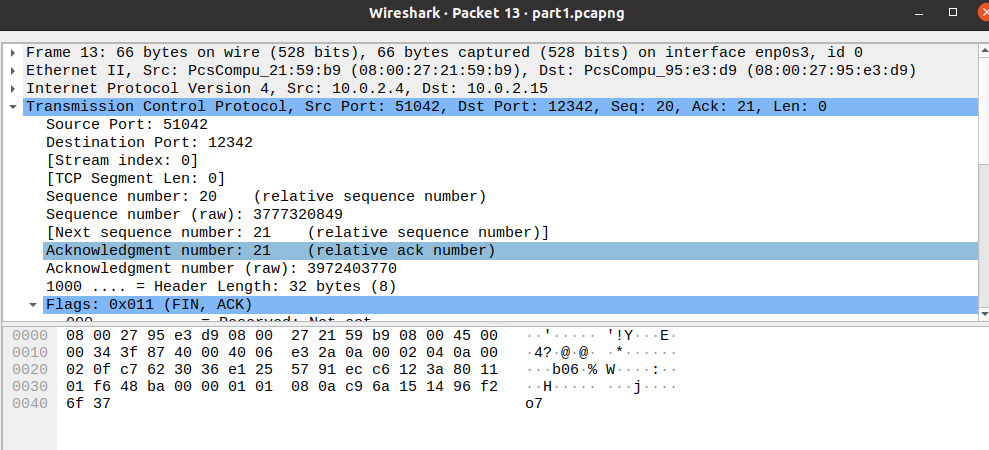


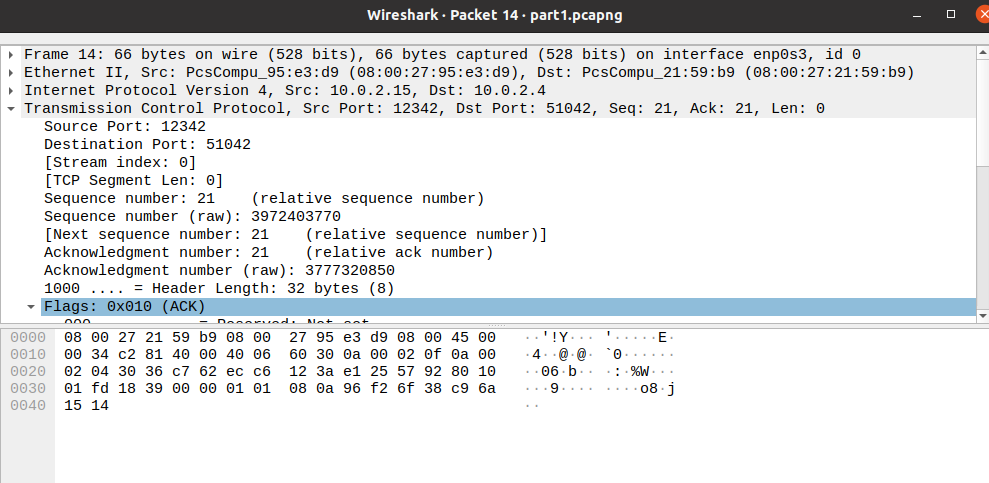
כעת, הלקוח יסכים עם השרת וישלח על בקשת השרת ACK :



נראה כי המספר הסידורי של הלקוח הוא 3777320849 זהה לACK הקודם ללא תוספת מידע לכן נשאר כך. הACK הוא 3972403769, בדומה למספר הסידורי הקודם.

ועכשיו, הלקוח יצהיר גם על כוונתו לניתוק הקשר, וישלח גם כן ACK עם FIN ללקוח, כאשר המספר הסידורי שלו נשאר 3777320849, אך הפעם ה הACK הוא 3972403770 (3972403769 + 1 על ACK).



לבסוף, בחבילה האחרונה, השרת מאשר ושולח ACK. נשים לב שהשרת מנתק את הקשר עם הלקח הזה ספציפי, אך פנוי לקבלת לקוחות חדשים, השרת לא נסגר. מספרו הסידורי יהיה הACK האחרון שבוצע-3972403770 וישלח ACK עם המספר הסידורי האחרון 777320849 + 1 עבור הACK, סהכ 3777320850.

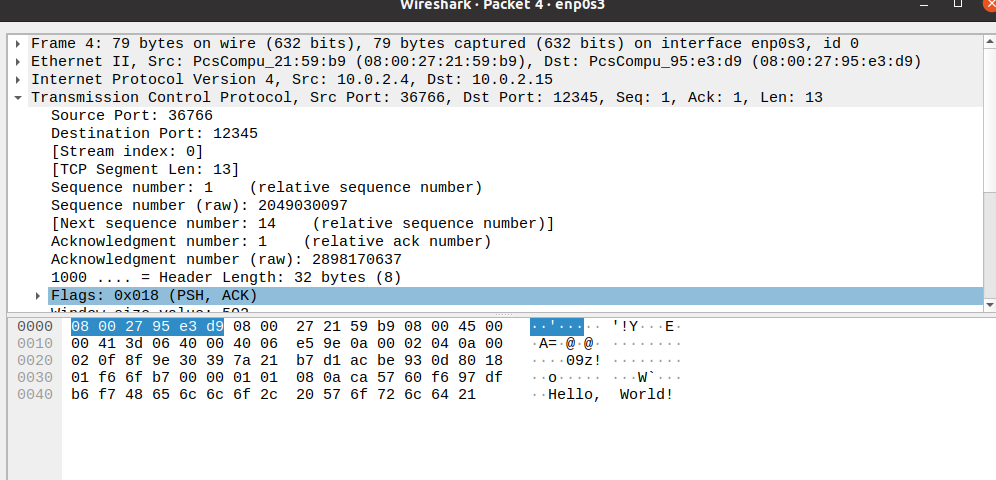
**חלק ב'**

נריץ את **v1**, נפתח זאת בwhireshark ונסנן לפי

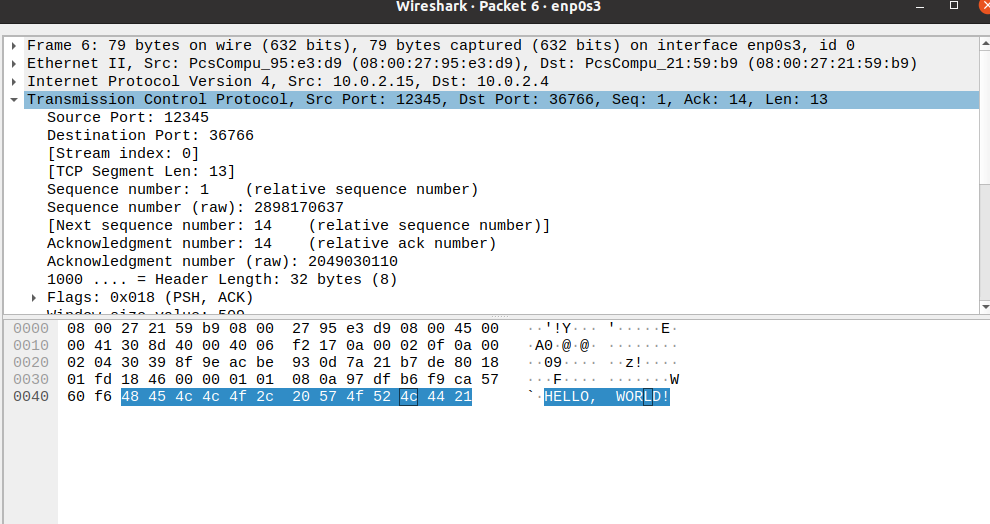
((ip.dst == 10.0.2.4) && (ip.src==10.0.2.15)) || ((ip.src == 10.0.2.4) && (ip.dst==10.0.2.15))

בחלק זה, Port השרת נקבע ע"י ארגומנט שאנו מכניסים – נבחר 12345, וכשאר נריץ את הלקוח ניתן את הארומגנטים 12345 וגם 10.0.2.15 שזהו ip של השרת.

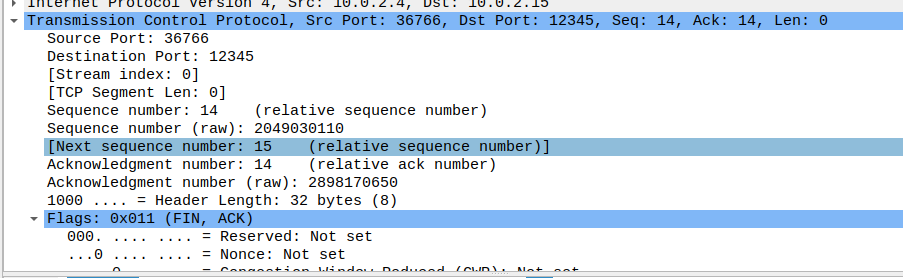
הלקוח שולח Hello, World לשרת והשרת יחזיר הודעה זו באותיות גדולות – HELLO, WORLD. כלומר זה דיי זהה לכל הקודים שראינו עד בה, אנו עובדים עם TCP, לכן החיבור יתבצע ע"י 3 שלבים-הקמה (דגל ה SYN), שליחת מידע וניתוק החיבור (דגל הFIN ).

הלקוח מעביר את המידע - 

והשרת מחזיר זאת באותיות גדולות-



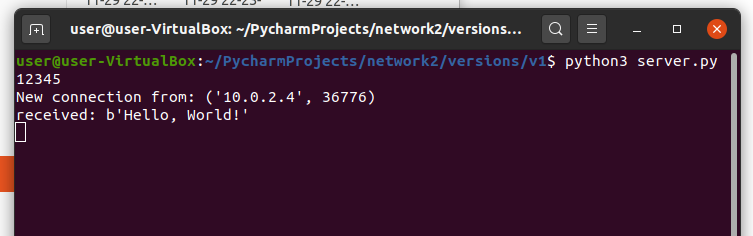
מבחינת ניתוק, הלקוח שולח בקשה כזו ראשונה, כאשר הFIN והACK דלוקים.

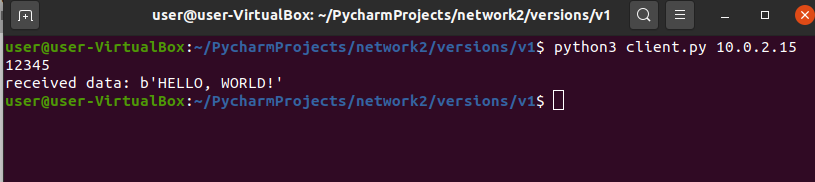
.



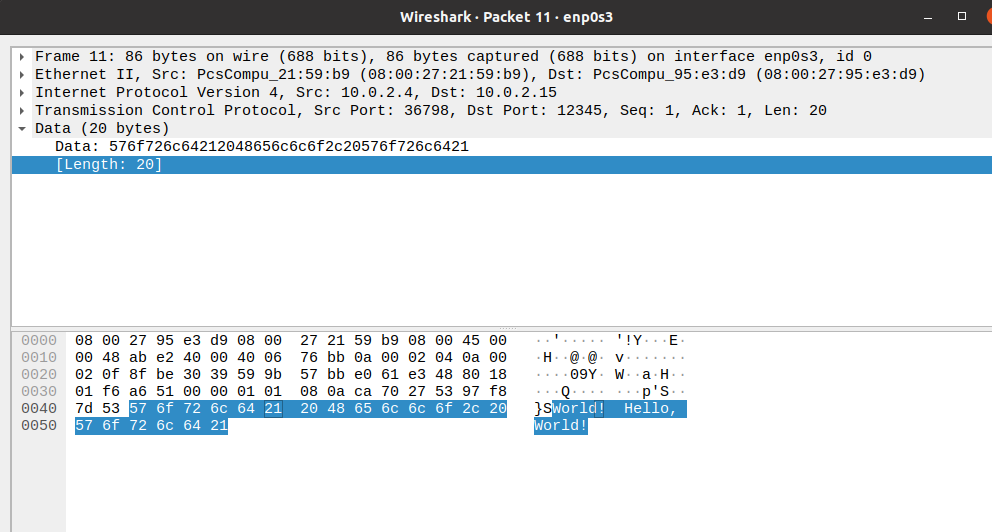
אך השרת אינו מעוניין בסגירת הניתוק, הוא פנוי לקבלת לקוחות נוספים.

אכן מתקיים:

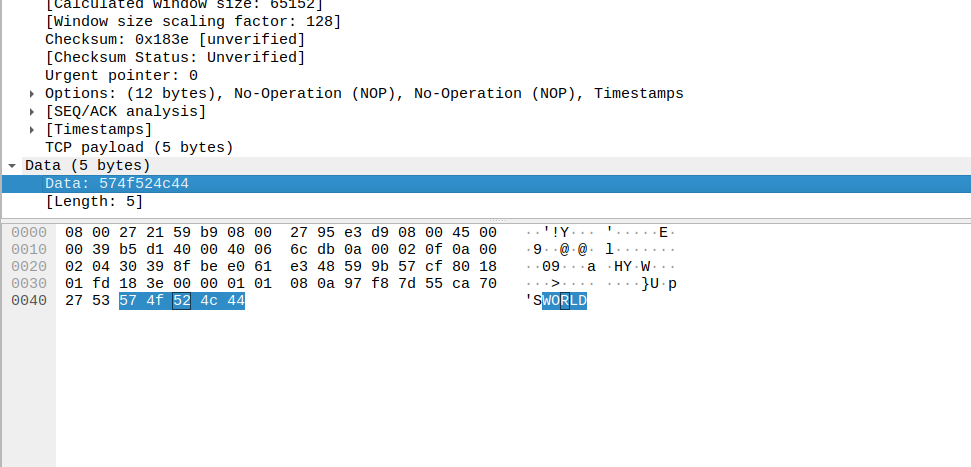




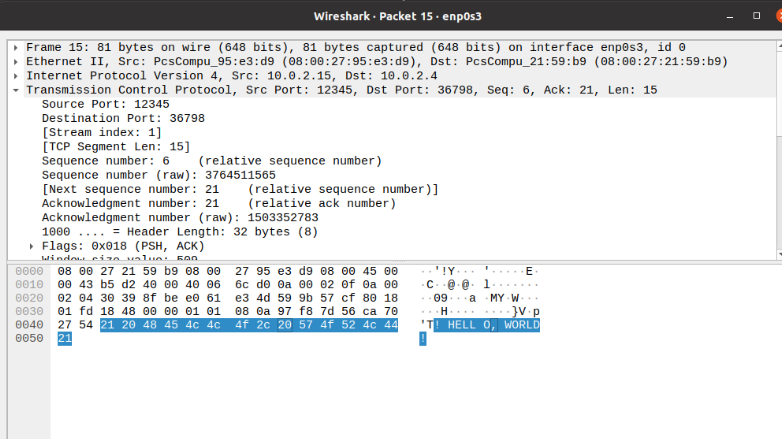
נעבור **לv2**, כבר מהקוד עצמו ניתן לראות כי גודל הBUFFER הוקטן משמעותית בשרת וכרגע הוא 5. הפעם, הלקוח שולח World! Hello, World! ומצפה לקבל זאת מהשרת באותיות גדולות. שוב אנו בTCP לכן השלבים הכללים זהים. המידע שהלקוח שולח בגודל 20 בתים – במספר התווים בביטוי:



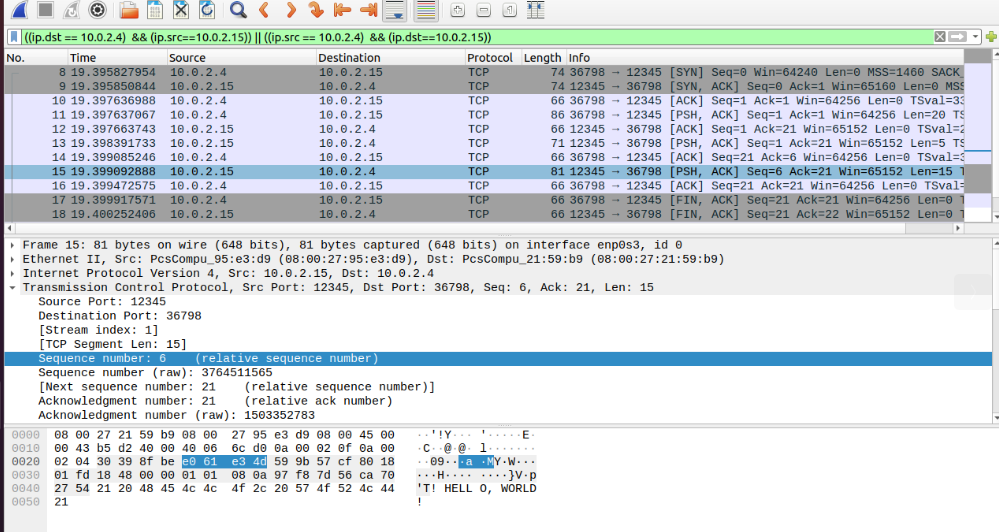
כעת, מכיוון שהשרת יכול לקבל עד 5 בתים, לפי הבאפר, הוא יצליח לקרוא כרגע רק את 5 הבתים הראשונים, כלומר World.



כעת, מכייון שיש עוד מידע לקרוא והשרת לא נסגר, הוא ימשיך לקבל 5 בתים הבאים, אך במקרה הבא, הוא עוד לא קיבל ACK על החבילה הנוכחית לכן הוא ימתין וביינתים ימשיך לקרוא עוד 5 בתים ועוד 5 בתים, לאחר מכן יקבל ACK על כולם יחד, לכן סה"כ נראה זאת:

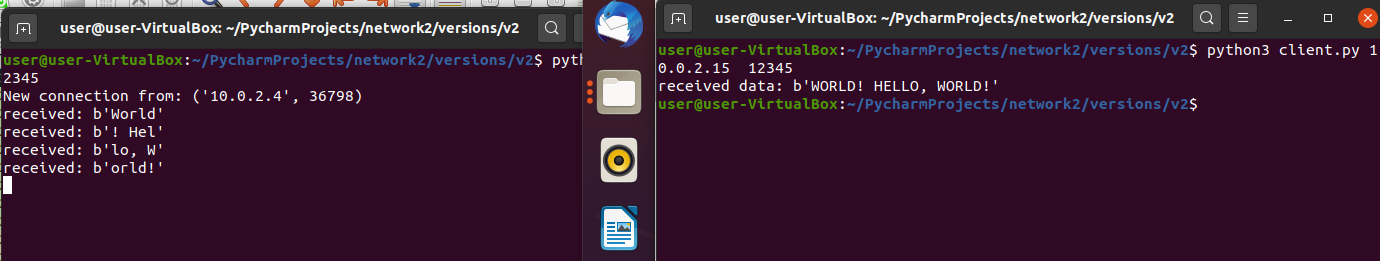


החבילה המסומנת בכחול היא החבילה עם שאר המידע:

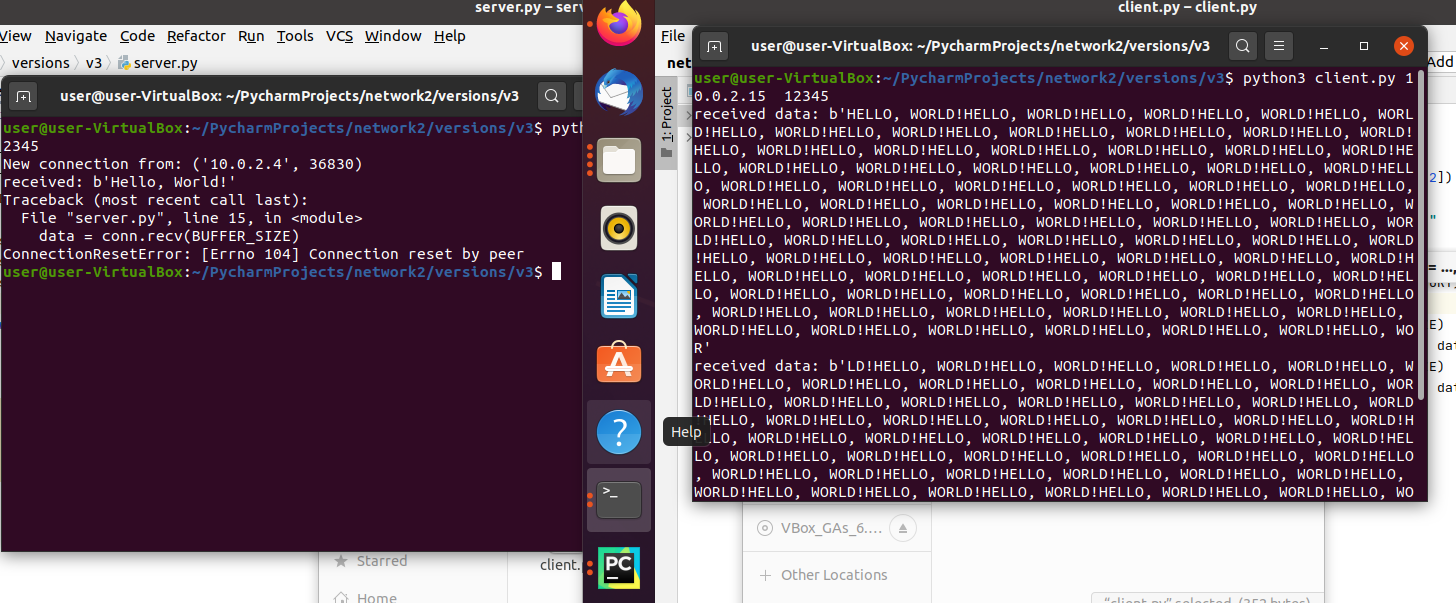


כך שניתן לראות שהיא מקבלת את שאר הביטוי ואין עליהם ACK בין לבין, רק אחד סופי, אחרי חבילה זו. לבסוף. מתבצע הניתוק עבור הלקוח, השרת עדיין מחובר.

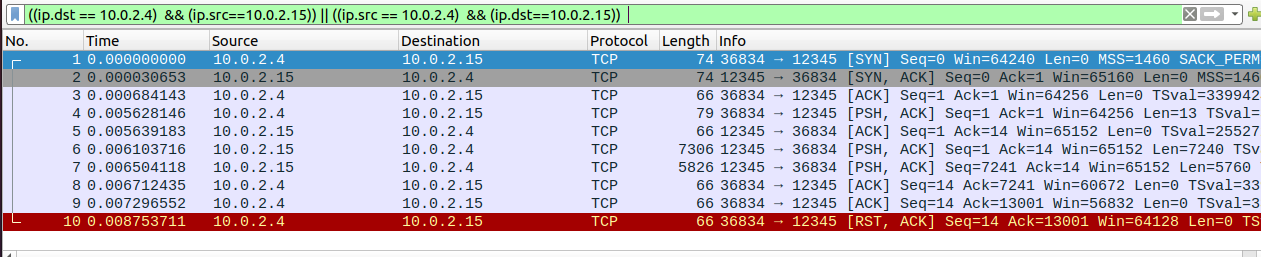
סה"כ, נקבל:



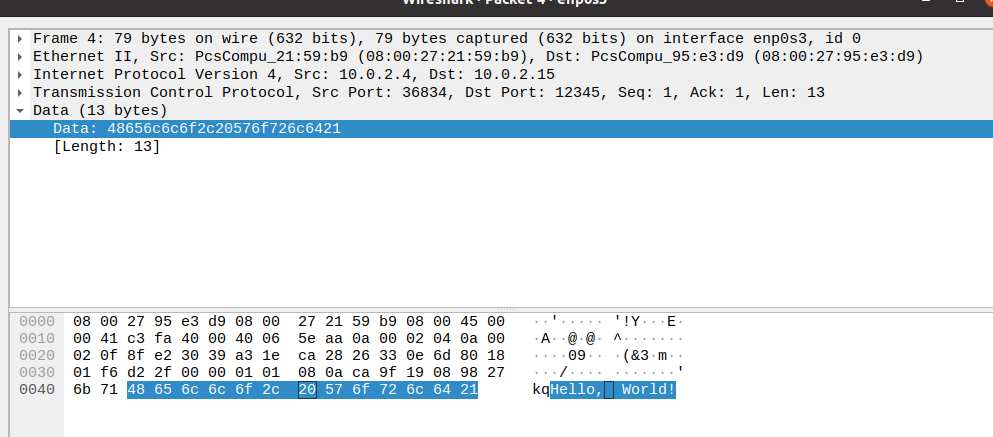
נעבור **לv3**, כעת גודל הבאפר הוא 1024.  
הלקוח בפעם הראשונה שולח "Hello, World!" לשרת ומצפה לקבל זאת באותיות גדולות. נשים לב כי בקוד השרת, בשונה ממה שקיבלנו עד כה, הוא מבצע conn.send(data.upper()\*1000)

כלומר הוא מכפיל את כמות המידע שמחזיר ללקוח באותיות גדולות ואכן נקבל

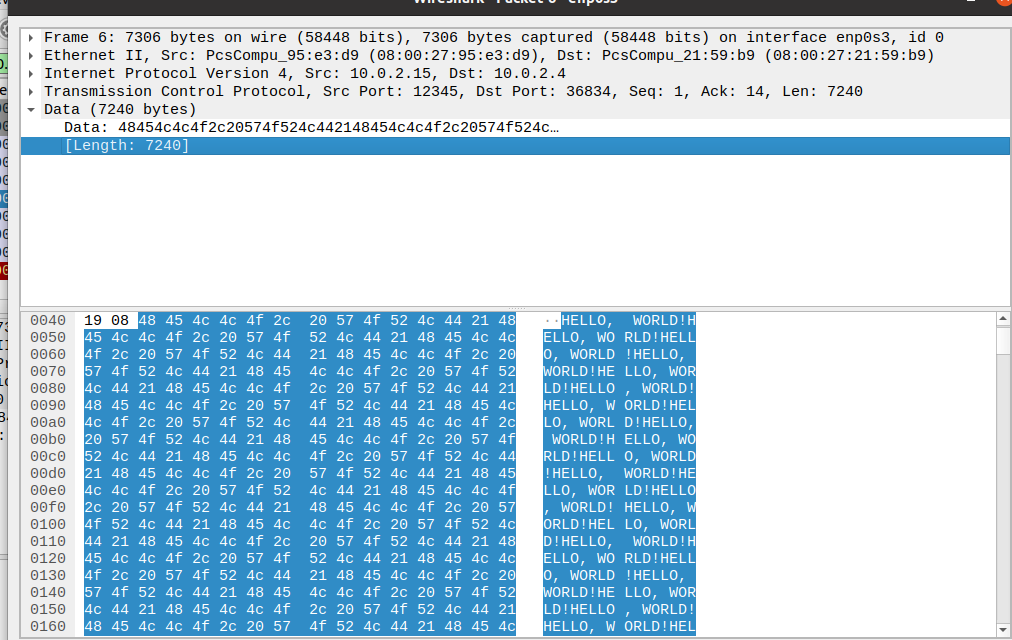
השגיאה נובעת מכך שאנו חוצים את הגבולות המותרות לבפאר. נסתכל על כך בwhireshark

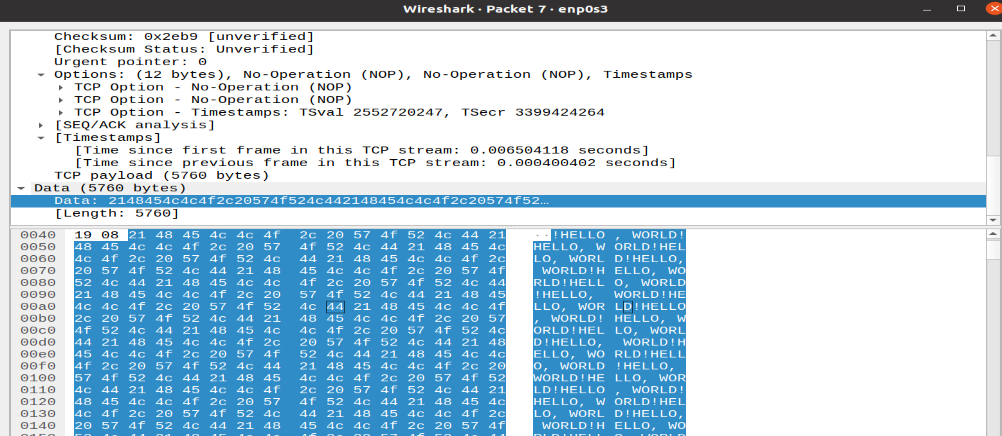


הקמת החיבור מתבצעת כרגיל עבור TCP, לאחר מכן, נראה כי הלקוח שולח Hello World

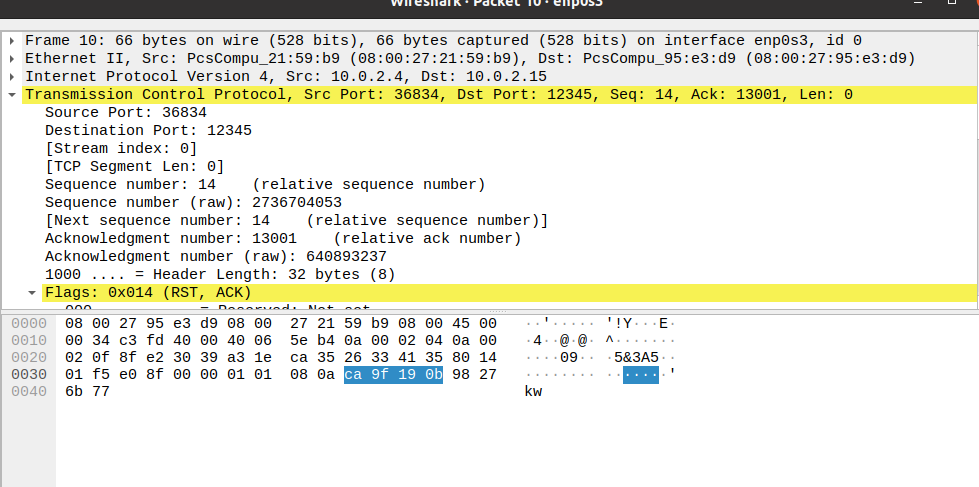


לאחר קבלת הACK עבור חבילה זו, נקבל חבילה שמקורה מהשרת-ששולחת את ה Hello World באותיות גדולות ברצף. גודל המידע הוא 7240.

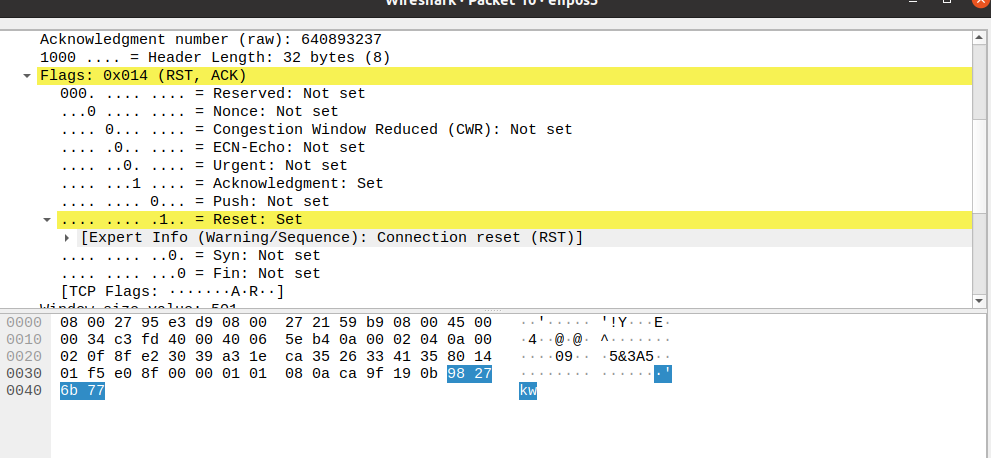


ולאחר מכן חבילה נוספת, נשים לב שגודלה 5760-

לבסוף, בחבילה האחרונה המסומנת באדום:

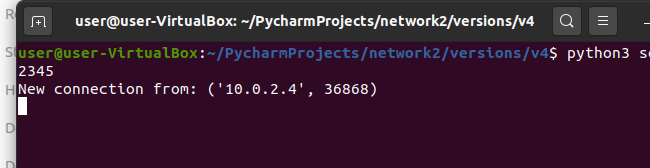


שבה מודלק הדגל של RST יחד עם ACK – RST אומר שהשרת מפסיק לטפל בלקוח הזה, הוא לא יכול לקבל יותר מידע ויש לחדש-לרסט את הקשר.



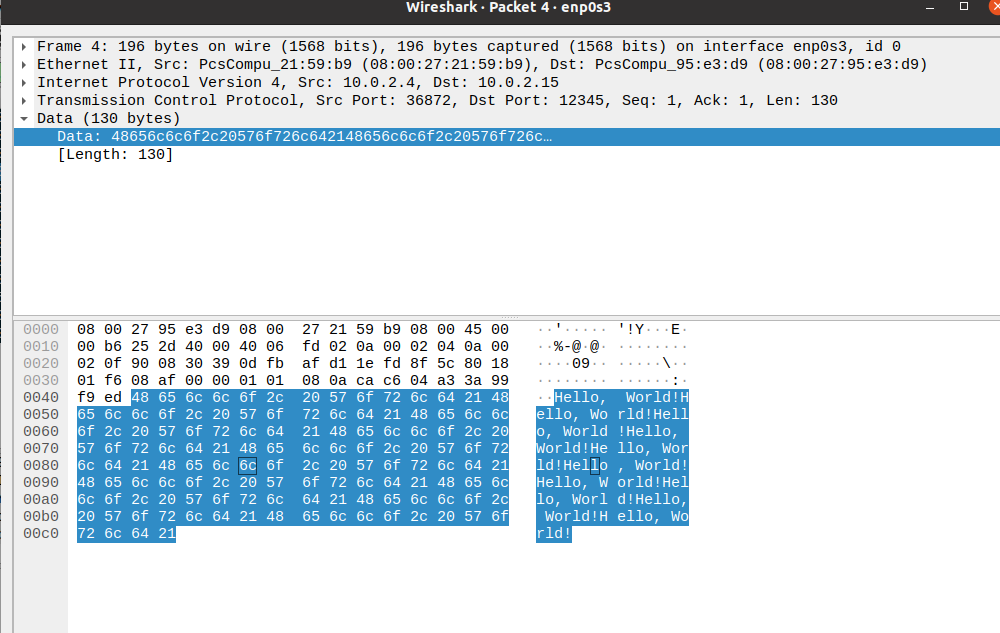
נעבור ל**v4**, בחלק זה ניתן לראות כי הלקוח שולח 4 פעמים את המידע שלו שזה Hello World כאשר כל פעם שהוא שולח את המידע הוא מכפיל זאת ב10: s.send((MESSAGE\*10).

בנוסף, נשים לב כי בקוד השרת נוספה שורה time.sleep(5), כלומר עליו להמתין 5 שניות לפני שהוא מקבל מידע, לכן השרת ישאר על מצב זה, בדומה לתמונה, למשך 5 שניות:

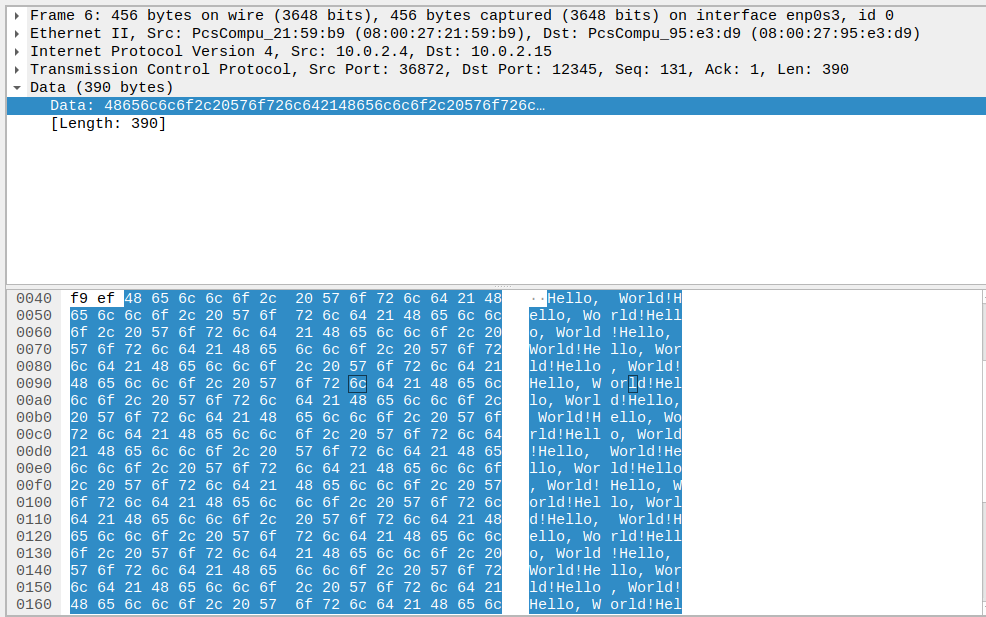


לבסוף, נקבל:

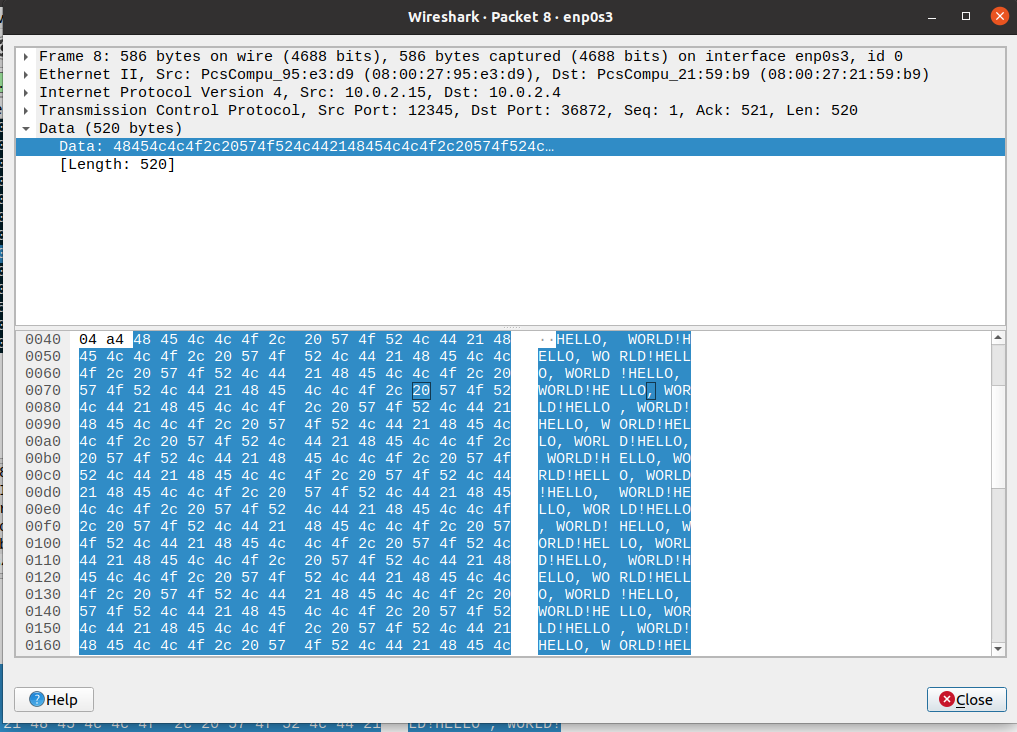
כעת ננתח זאת. מכיוון שאנו בפרוטוקול TCP, השלב של החיבור מתקיים כרגיל.  
הפעם, הלקוח שולח רצף של הודעות של Hello World , בניגוד לv3, בגודל 130:



לאחר מכן יתקבל על כך ACK והשרת ימשיך לשלוח, הפעם בגודל 390 בתים.



ואז השרת מתחיל להחזיר הודעות אלו באותיות גדולות, בגודל 520:



הלקוח מאשר את קבלת החבילה האחרונה בACK ואז מתחיל שלב ניתוק החיבור. סהכ, המידע שמועבר כאן קטן יותר מהמידע שהועבר בv3 וגרם להגעת גבול הבאפר. לכן כאן החיבור יסתיים כהרגלו.