**שאלה 2:**

**א.**

ניתן לראות בתרגיל זה כי לא ניתן לשלוח פקטות גדולות מדי בבת אחת, דוגמא כרגע – 15,000 פעמים את האות A בפעם אחת. מכך, פרוטוקול TCP יפרק את המידע למספר פקטות שונות.

על מנת לדעת שכל הפקטות הן חלק מאותה ההודעה ולוודא שאכן קיבלנו בדיוק את המידע שרצינו, הוספנו לבסוף את ההודעה \*FINISH\* וחיברנו את תוכן ההודעות. במידה והמידע נכון, הוחזר B.

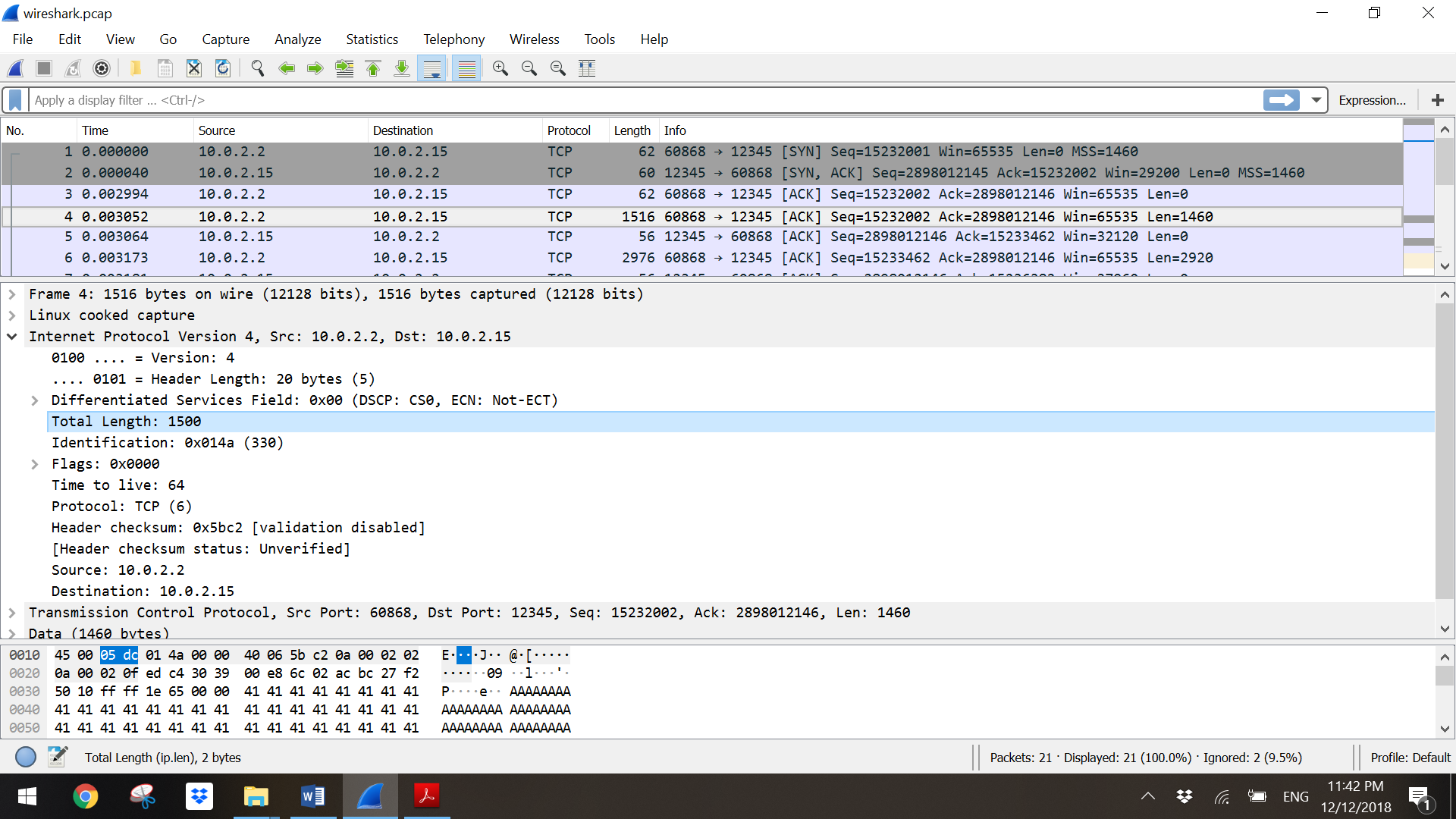
ניתן לראות בפקטה הראשונה שכמות המידע הוא 1500 תווים.

הפורט המוגדר עבור המקור הינו 60686

הפורט המוגדר עבור היעד הינו 12345

Sequence number: 15232002

Acknowledgment number: 2898012146



ניתן לראות בפקטה הבאה שכמות המידע הוא 2960 תווים.

Sequence number: 15233462

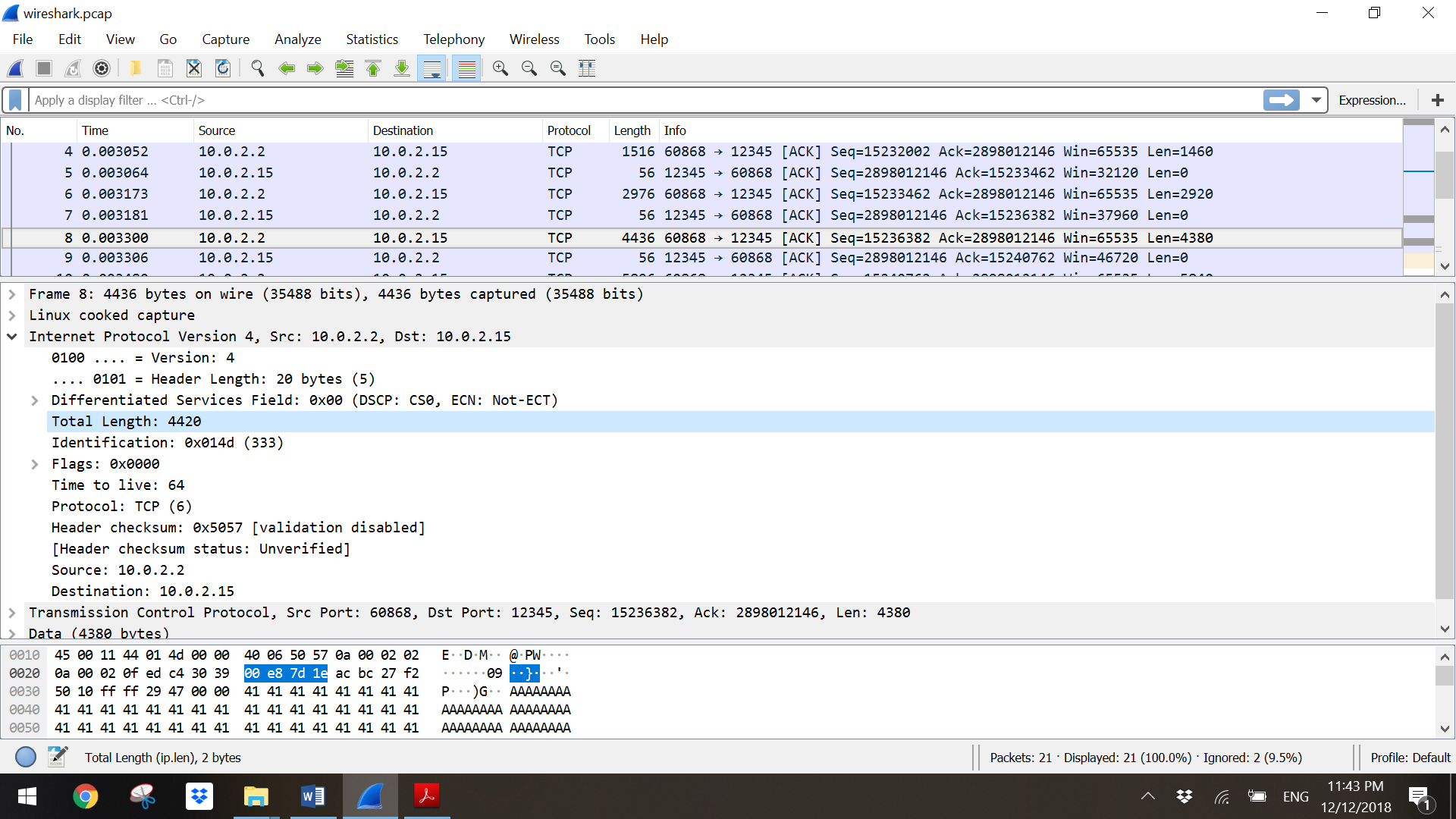
Acknowledgment number: 2898012146



ניתן לראות בפקטה הבאה שכמות המידע הוא 4420 תווים.

Sequence number: 15236382

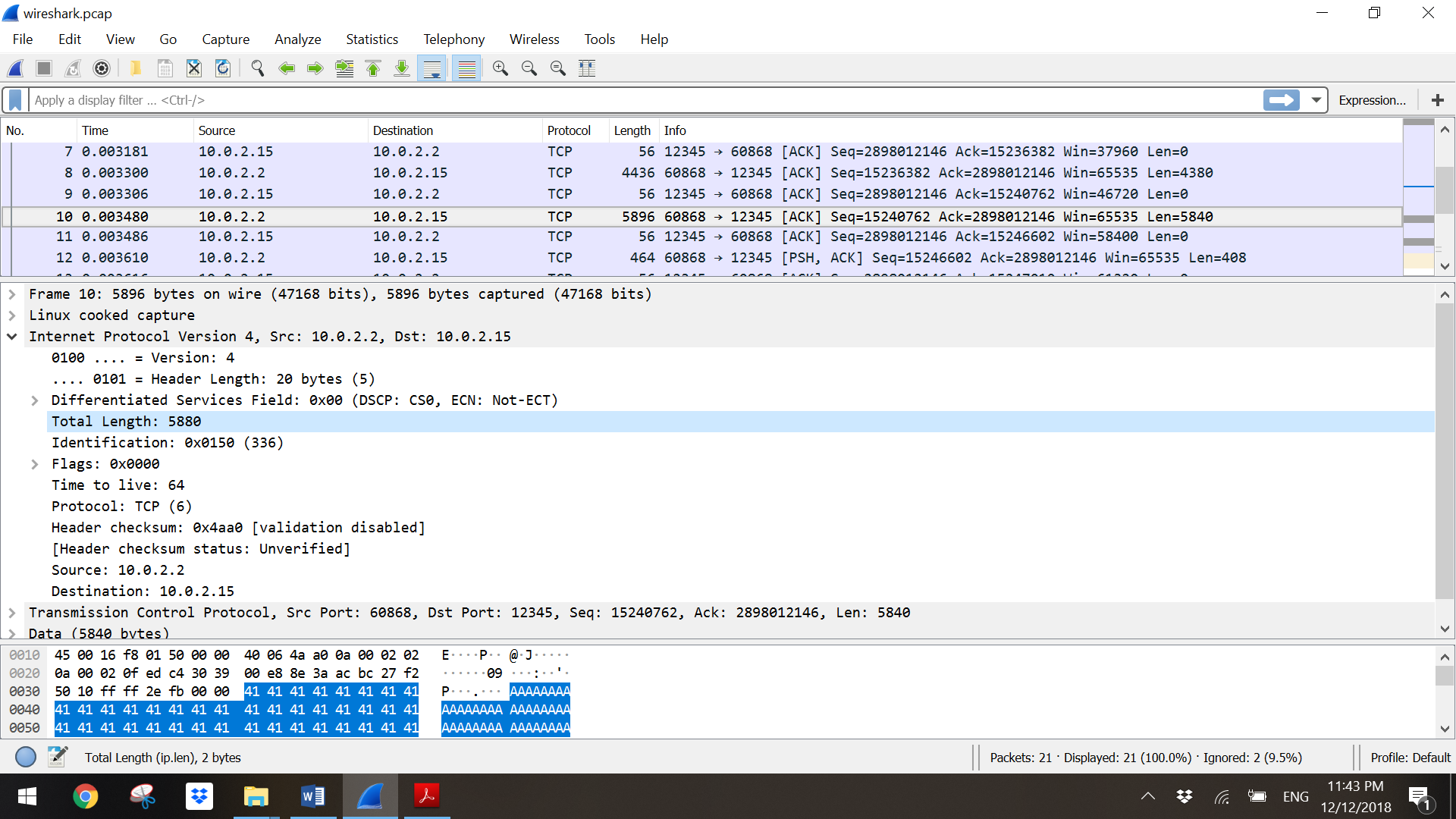
Acknowledgment number: 2898012146



ניתן לראות בפקטה הבאה שכמות המידע הוא 5880 תווים.

Sequence number: 15240762

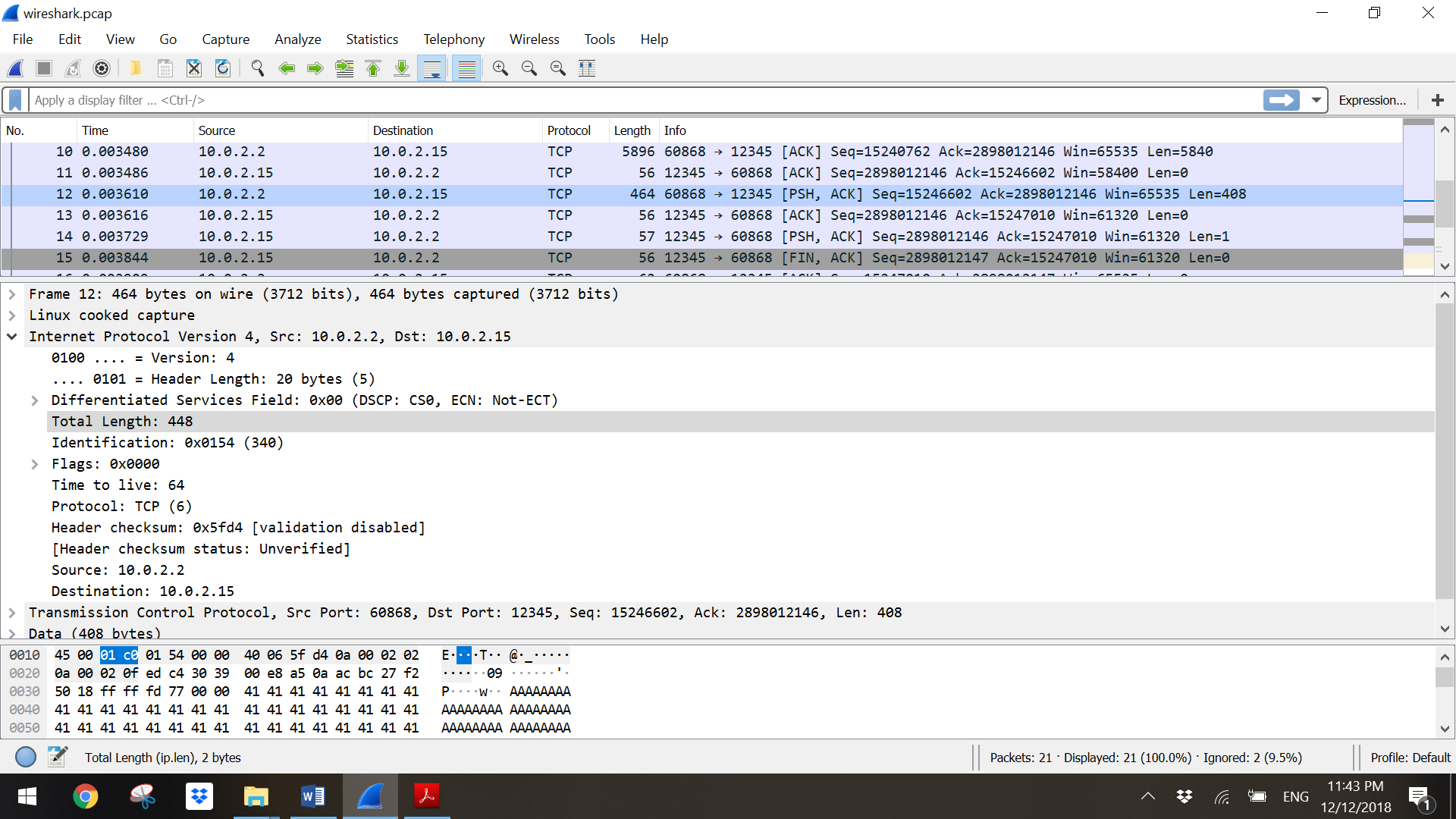
Acknowledgment number: 2898012146



ניתן לראות בפקטה הבאה שכמות המידע הוא 464 תווים.

Sequence number: 15246602

Acknowledgment number: 2898012146

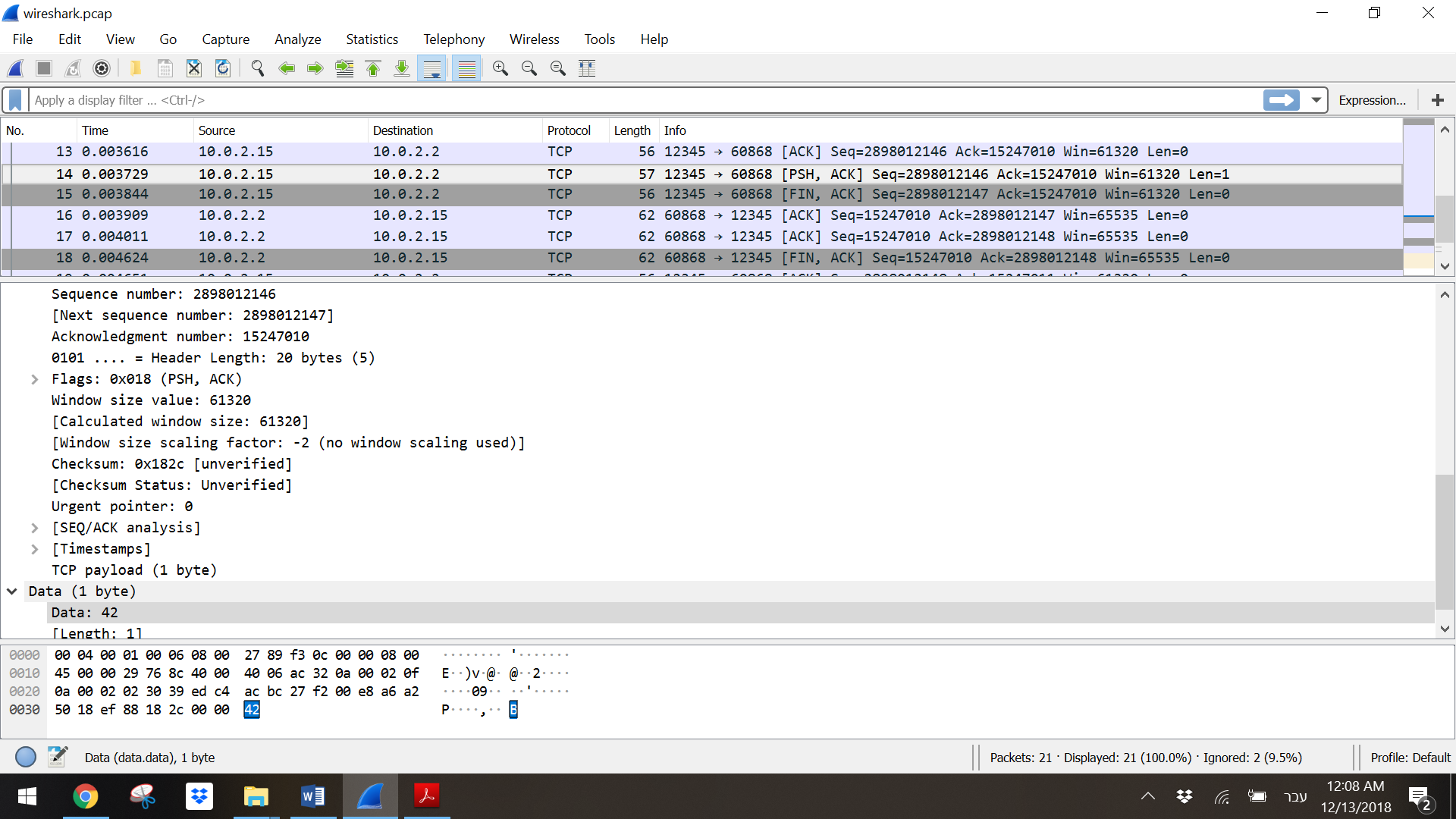


מחיבור כמות המידע המועבר על ידי הפקטות, קיבלנו את כמות התווים A 15,000. ( בנוסף לתווי ה \*FINISH\*)

לאחר מכן, השרת בודק האם המידע הוא מידע נכון. מפני שהמידע שהועבר נכון, השרת מחזיר אל הלקוח את ההודעה B בתמונה הבאה:

Sequence number: 2898012146

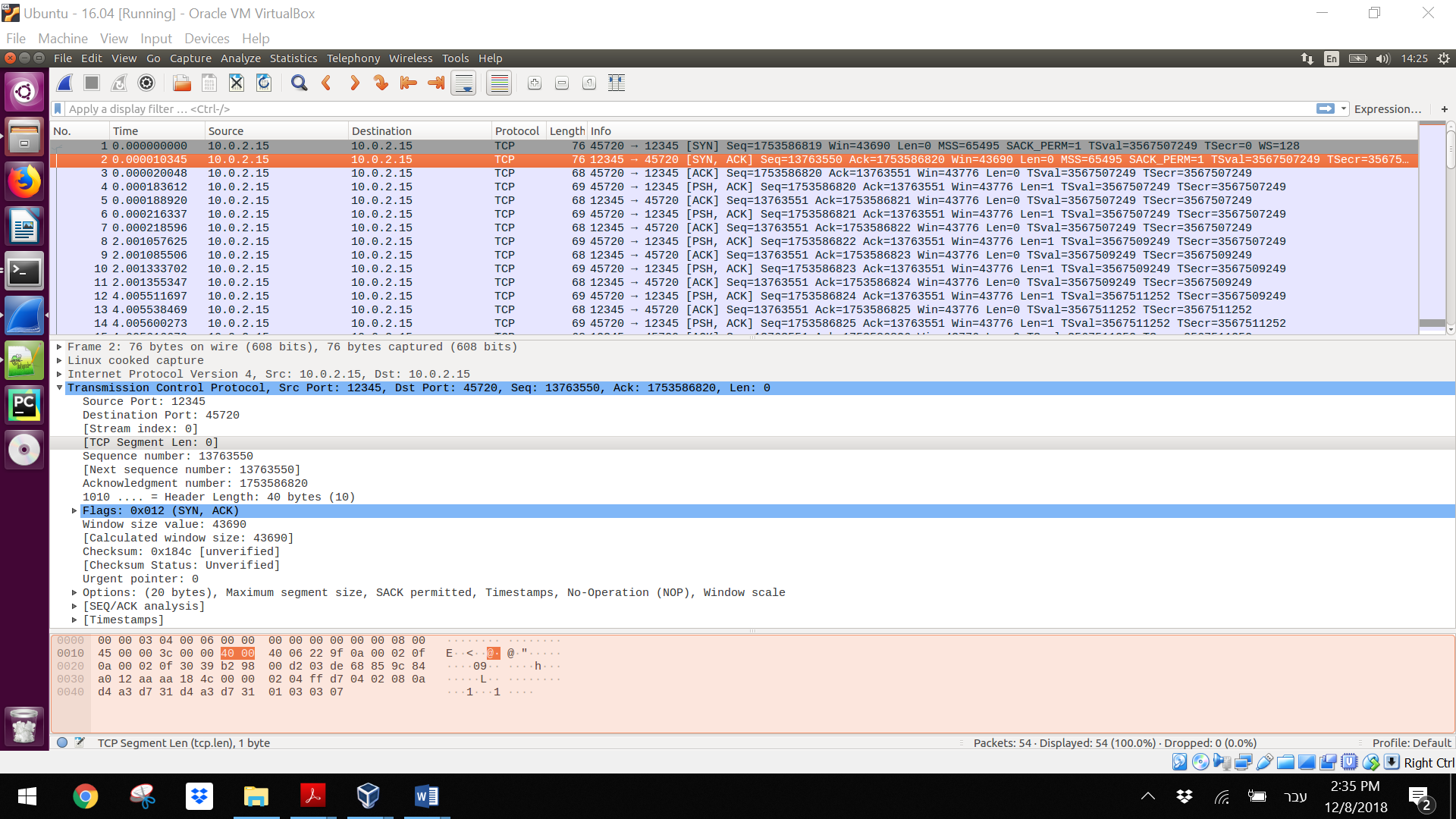
Acknowledgment number: 15247010



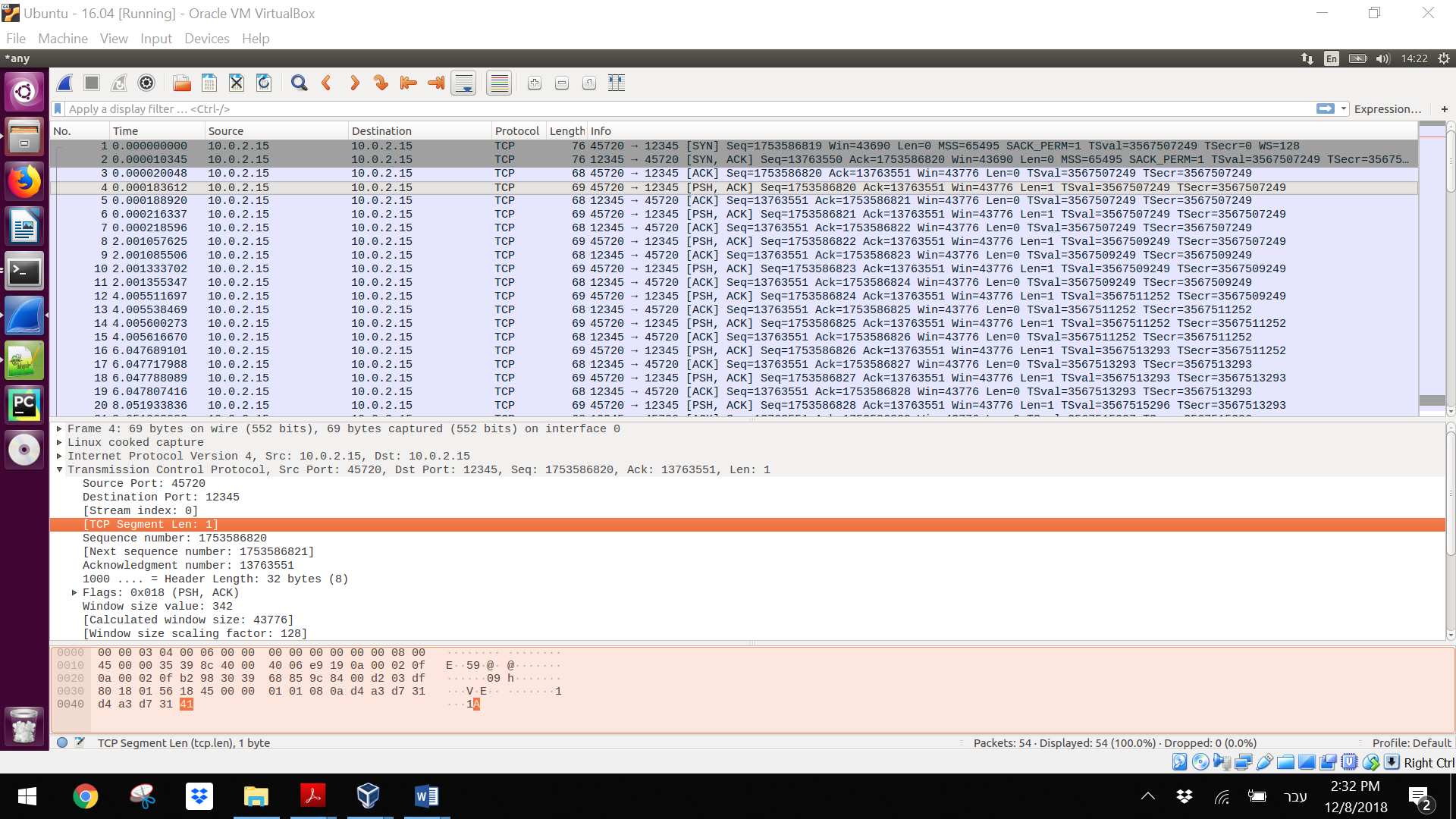
**ב.**

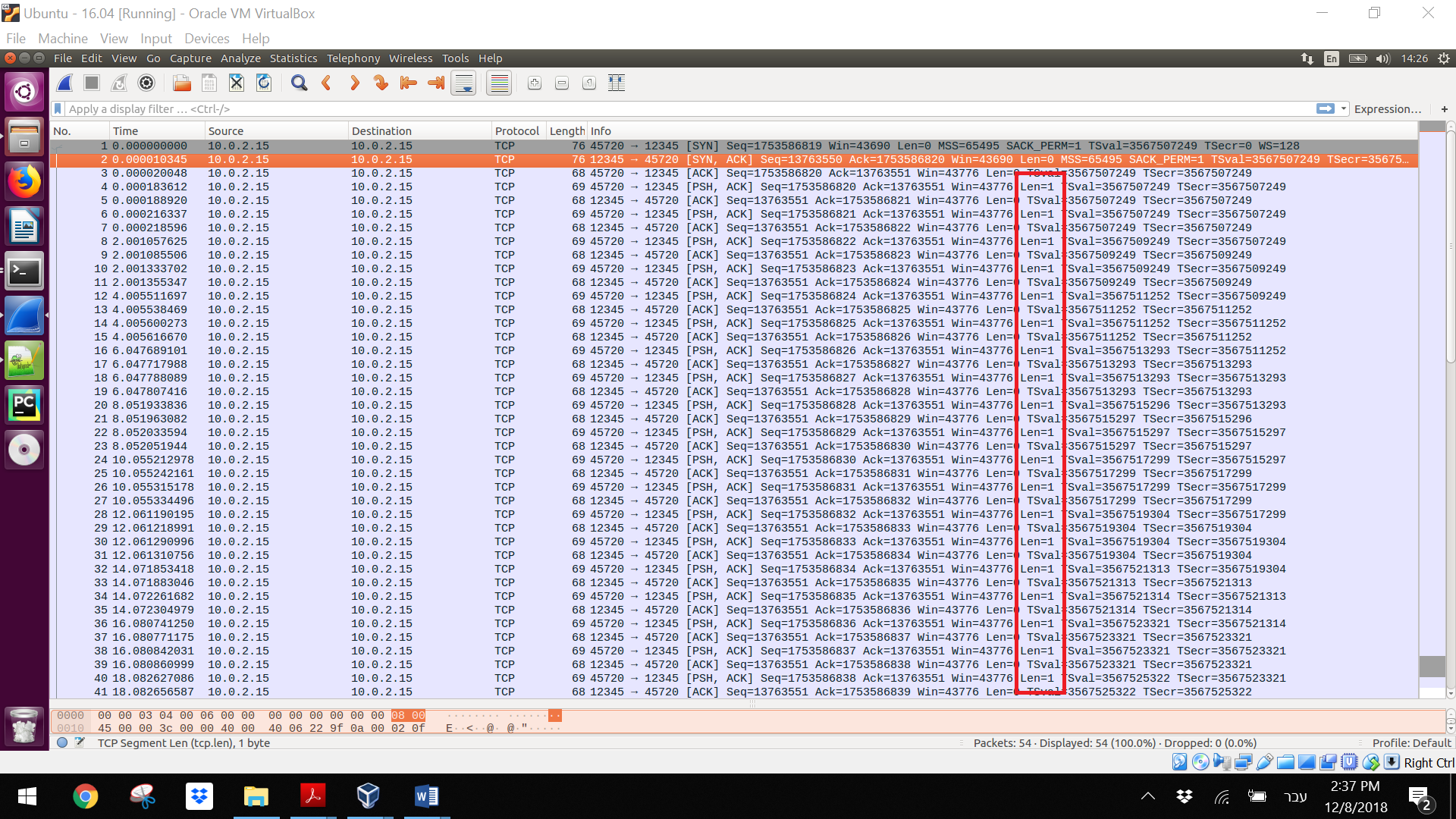
כפי שהתבקשנו בשאלה, אנו צריכים לשלוח את התו A פעמיים ברצף מהלקוח אל השרת, להמתין 2 שניות ולאחר מכן לחזור על פעולה זו במשך 10 פעמים נוספות.

תחילה מתבצעת לחיצת הידיים:



ממעבר על כל ההודעות שנשלחו, ניתן לראות שכל הודעה הכילה לכל היותר תו A אחד. זאת מפני שניתן לראות ב Len כי המספר הגדול ביותר של תווים הוא 1 כלומר כל הודעה נשלחה בנפרד עד שנגיע לסוף ההודעה ושליחת \*FINISH\* אשר מכיל 8 תווים.





דוגמא לנתונים משליחת ההודעה A מהלקוח אל השרת:

Transmission Control Protocol, Src Port: 45720, Dst Port: 12345

Sequence number: 1753586821

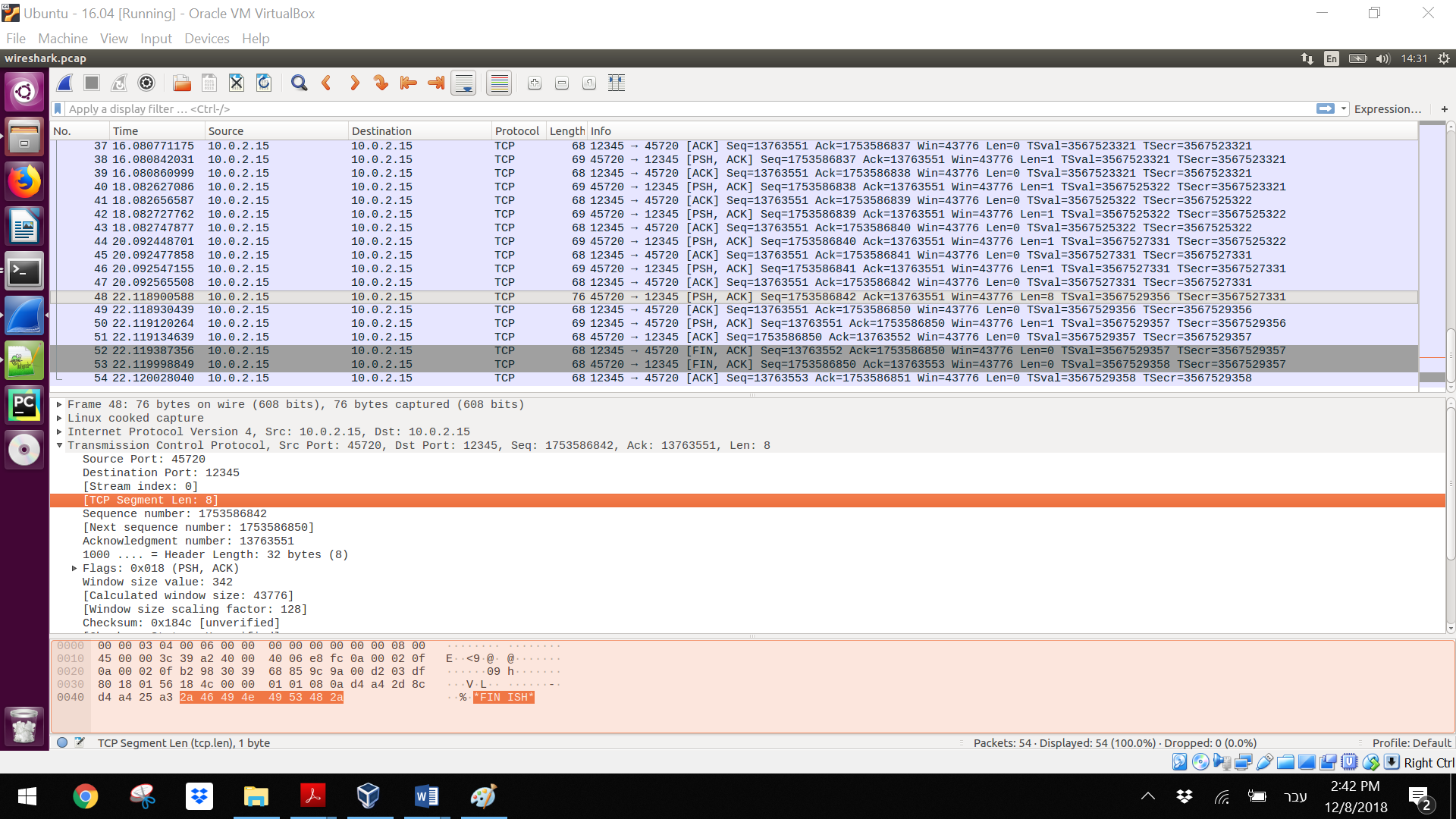
[TCP Segment Len: 1]

Acknowledgment number: 13763551

כעת, הלקוח שולח הודעה נוספת אל השרת.

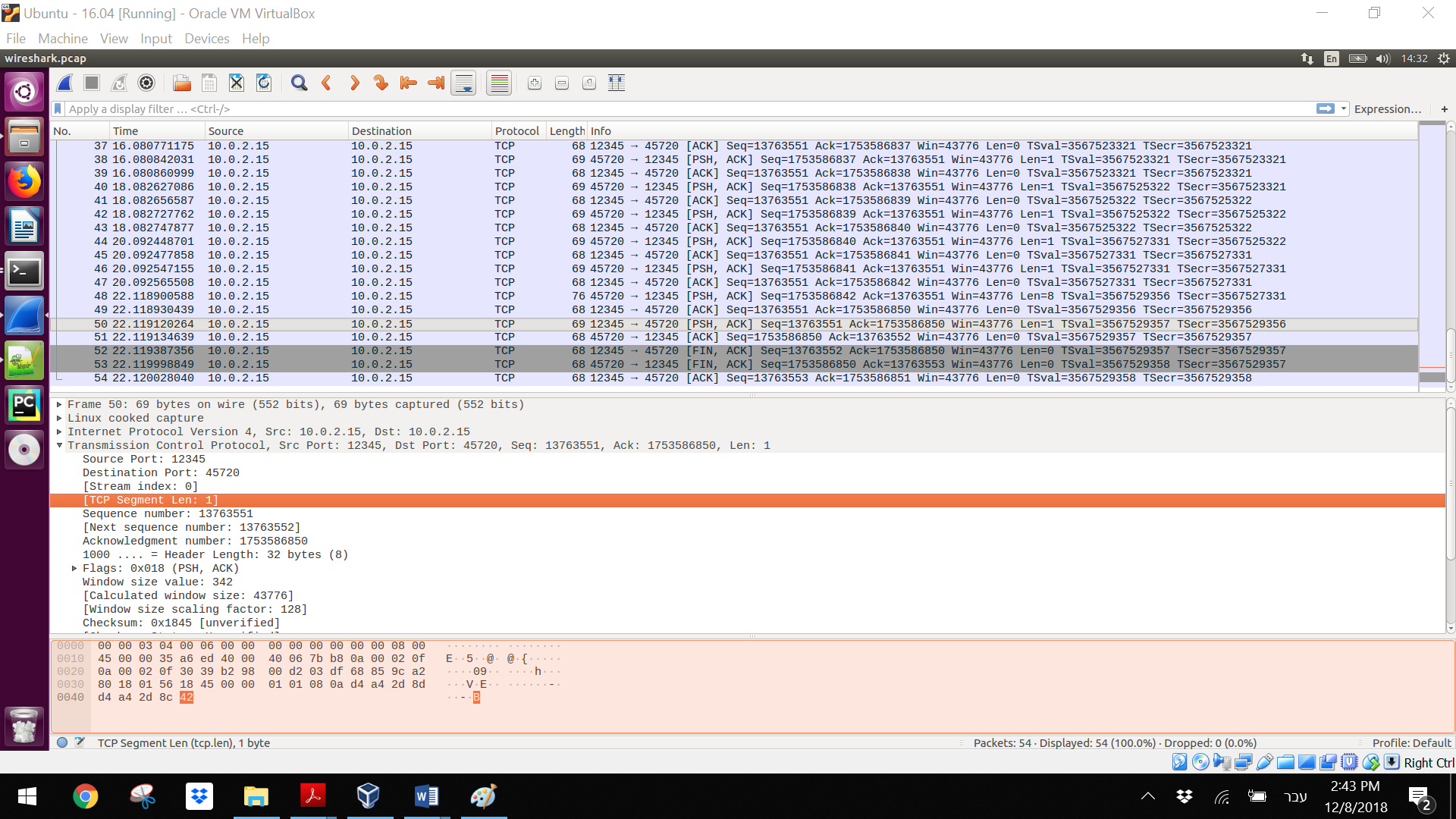
שלחנו בנוסף את ההודעה \*FINISH\* כדי להודיע על סיום ההודעה ולהודיע לשרת לבצע את הבדיקה.

Transmission Control Protocol: Seq: 1753586842, Ack: 13763551, Len: 8



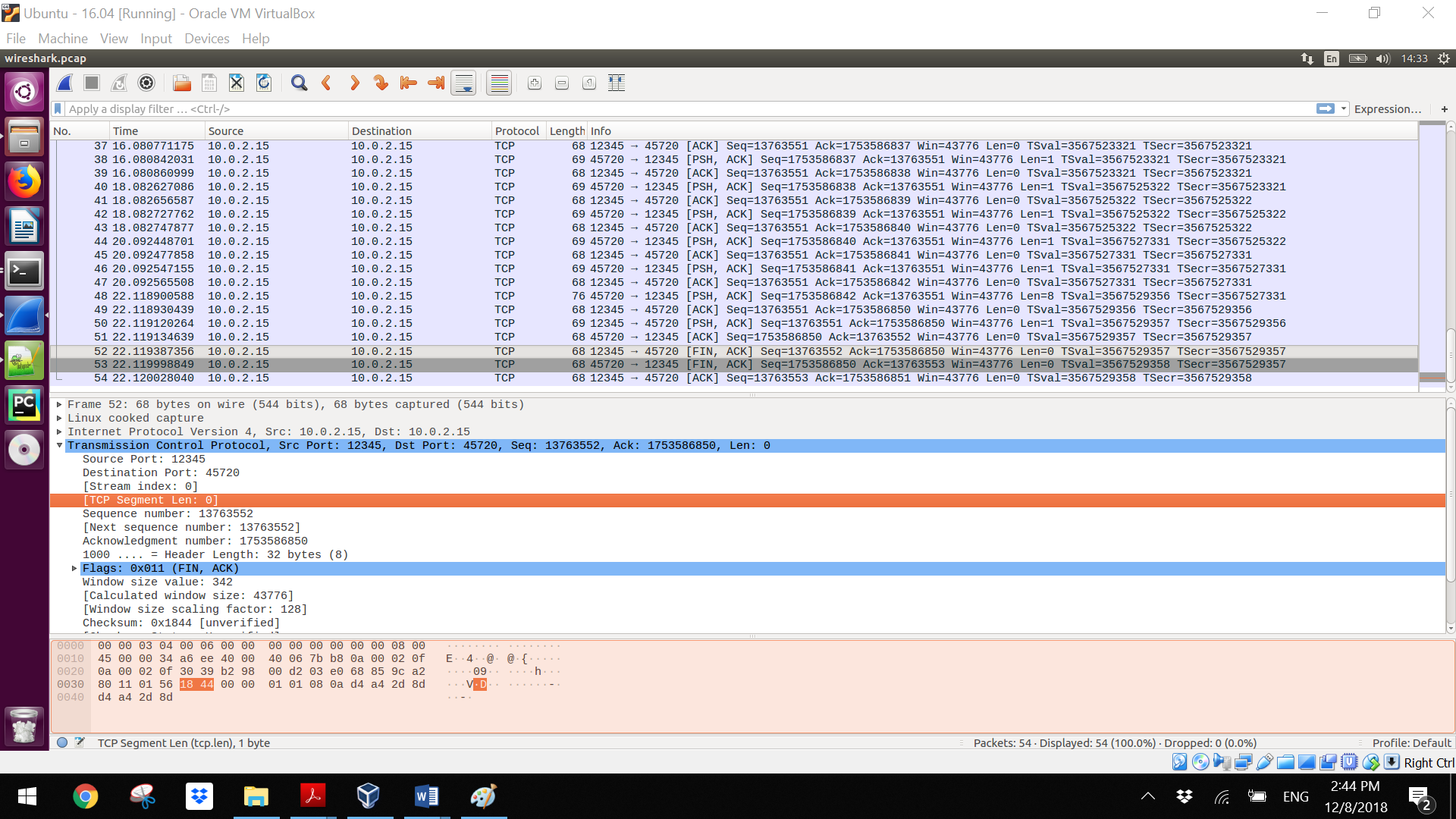
לאחר מכן, השרת בודק האם התקבלה ההודעה הרצוייה במלואה, במידה וכן, מוחזרת אל הלקוח ההודעה B.

Sequence number: 13763551, [TCP Segment Len: 1]



כעת השרת סוגר את התקשורת עם הלקוח:

Sequence number: 13763552



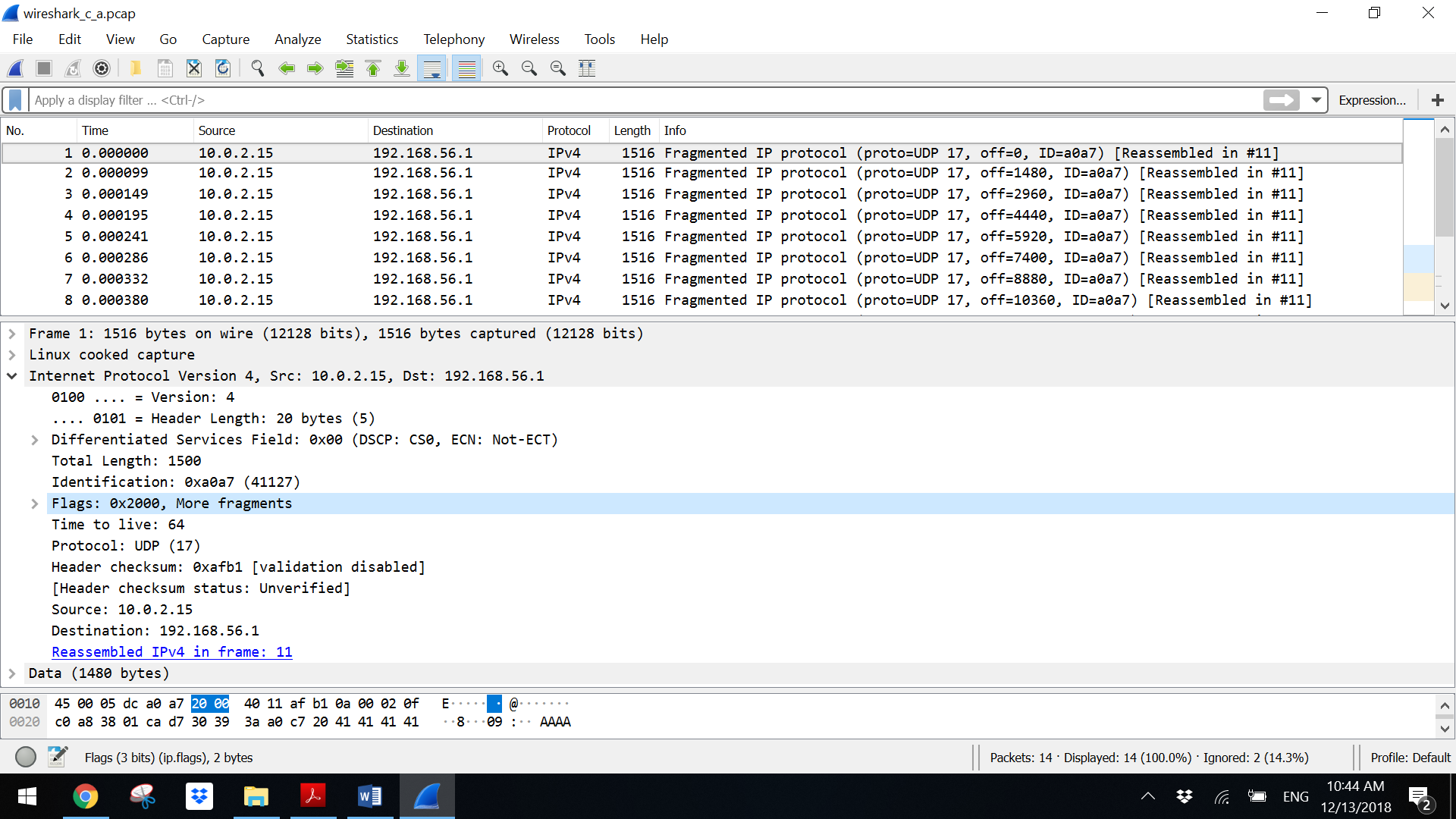
**ג.**

סעיף א:

מפני ששוב, אורך ההודעה – 15,000 פעמים A, גדול מדי עבור פקטה אחת ומפני שאנו עושים שימוש ב UDP, מתבצעת כעת פרגמנטציה ולא סיגמנטציה. הפרוטוקול מחלק בכל פעם את שארית ההודעה להודעה הגדולה ביותר שניתן לשלוח בבת אחת ושתוכל לעבור לערוץ הקרוב. לבסוף, כל מרכיבי ההודעה מורכבים להודעה הכוללת במחשב היעד.

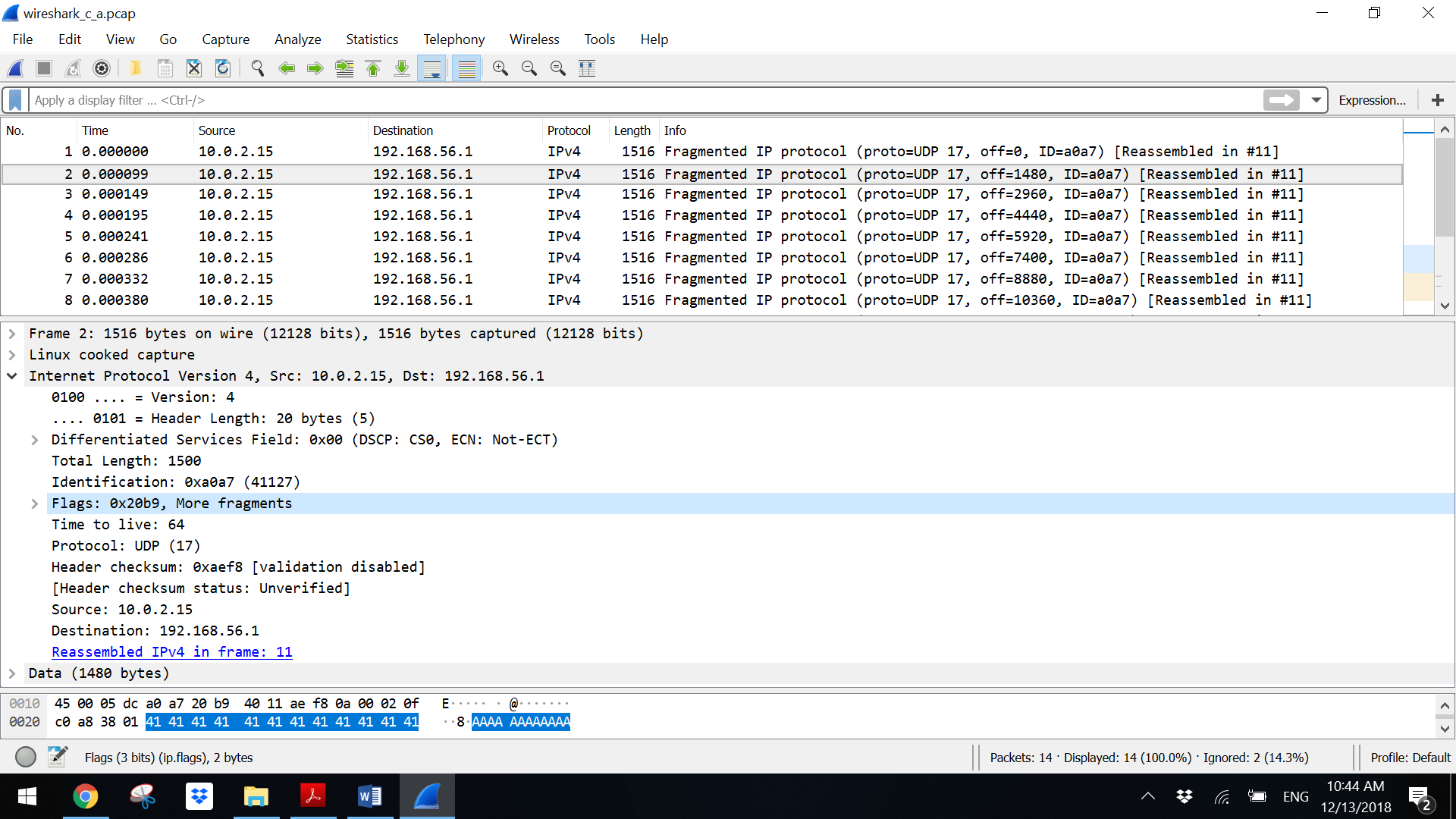
ניתן לראות בפקטה הראשונה שהדגל MoreFragments דולק.

ה offset שאנו רואים הוא מספר הבייט בו הפקטה עצרה. זאת על מנת שנוכל לספור את כמו הנתונים שעברו מתחילת שליחת ההודעה. כך הפרוטוקול מסוגל לספור באמצעות הדגל More Fragments.

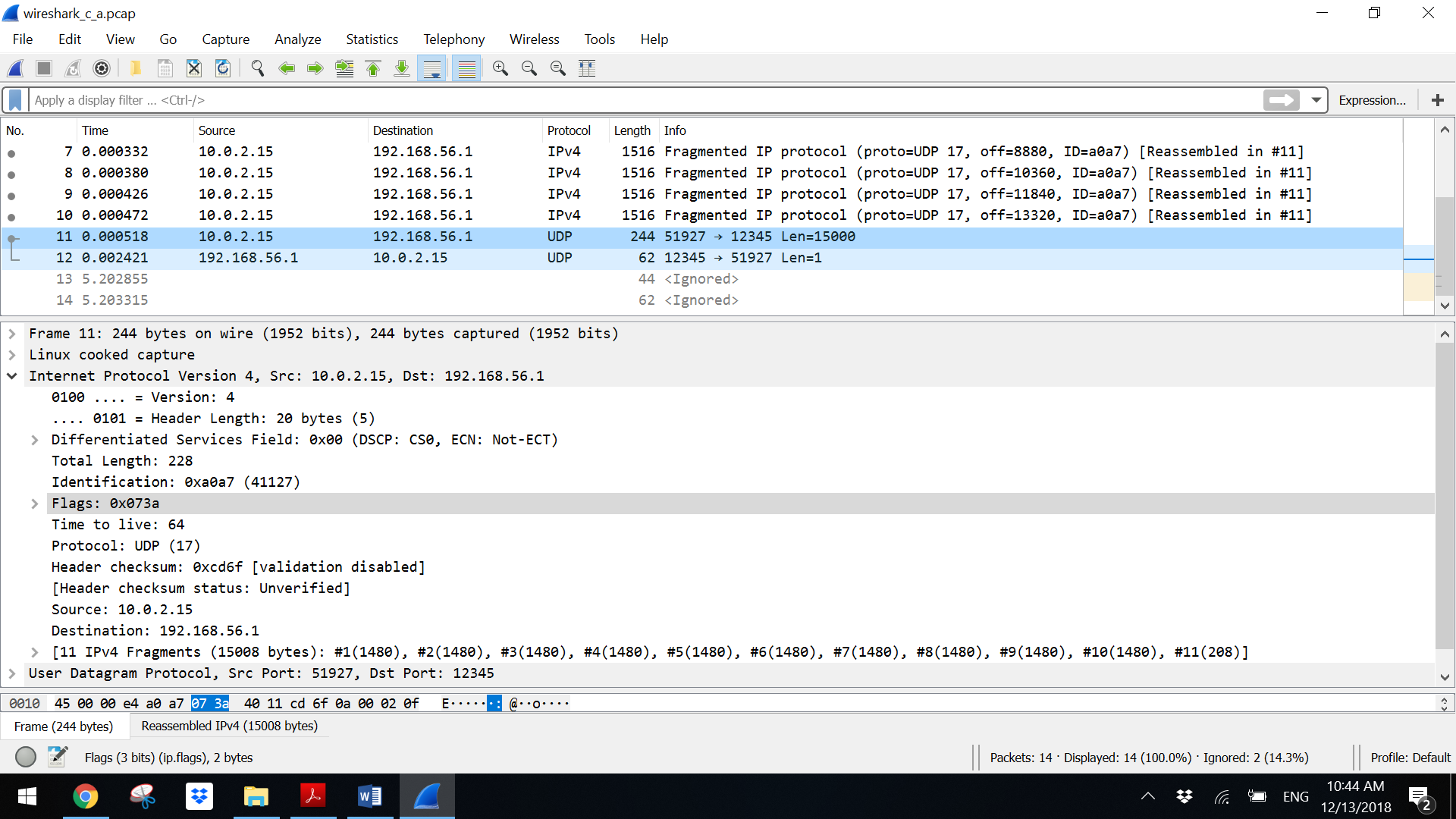


נראה דוגמא לפקטה שמועברת:

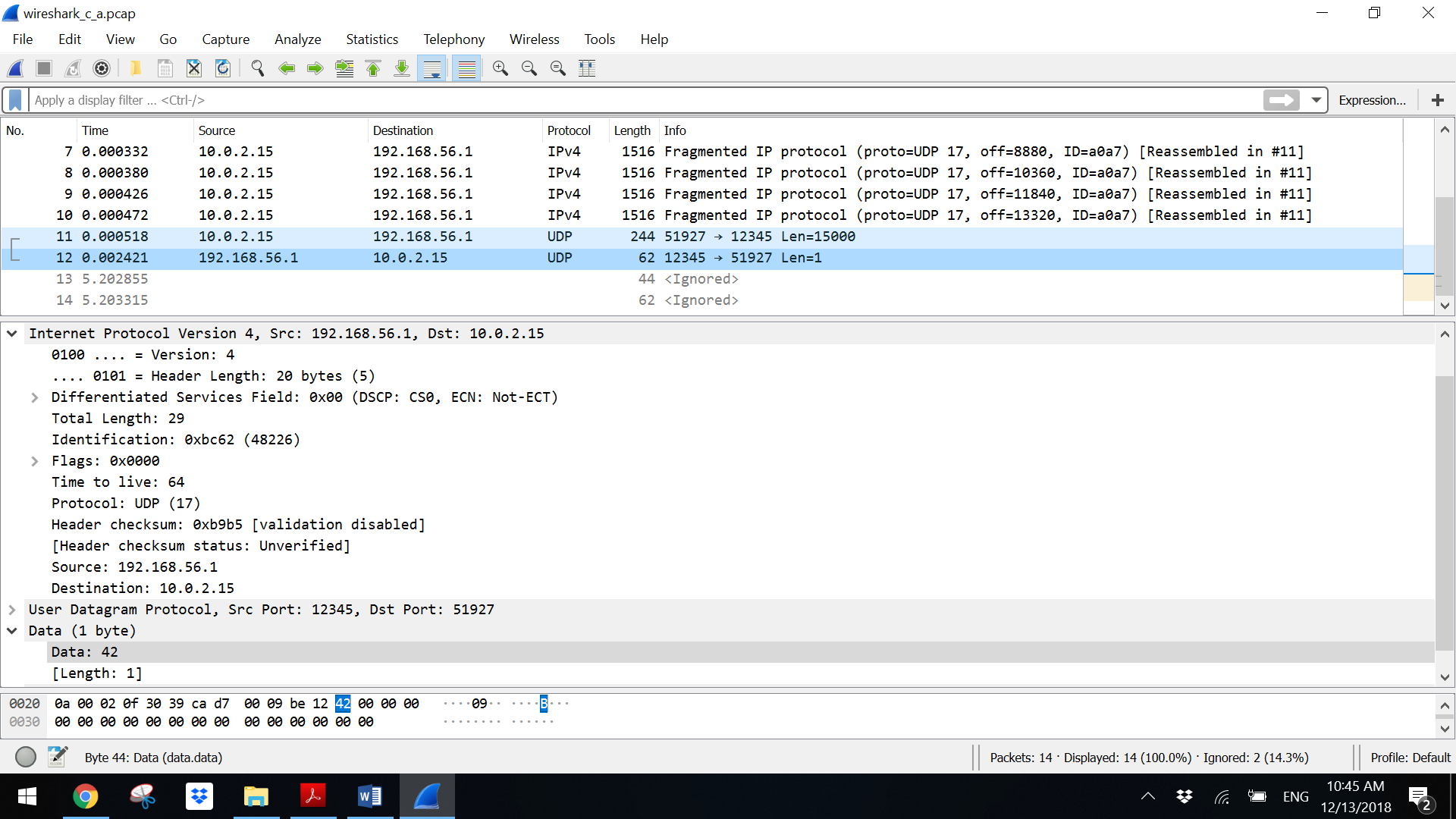
אורך הפקטה הארוך ביותר שהפרוטוקול החליט לשלוח הוא 1516, שזהו פירוק של האורך שאנו רוצים לשלוח מלכתחילה, של 15,000 פעמים A.



בפקטה האחרונה ניתן לראות שהדגל More Fragments אינו דולק, מפני שזוהי הפקטה האחרונה של המידע.



לאחר שהשרת בדק את המידע שהועבר אליו, הוא מוודא שהמידע הוא בדיוק כמו המידע הרצוי, מפני שזהו המידע, ההודעה שמועברת בחזרה היא הודעה קצרה B, ללא צורך בפרגמנטצייה.

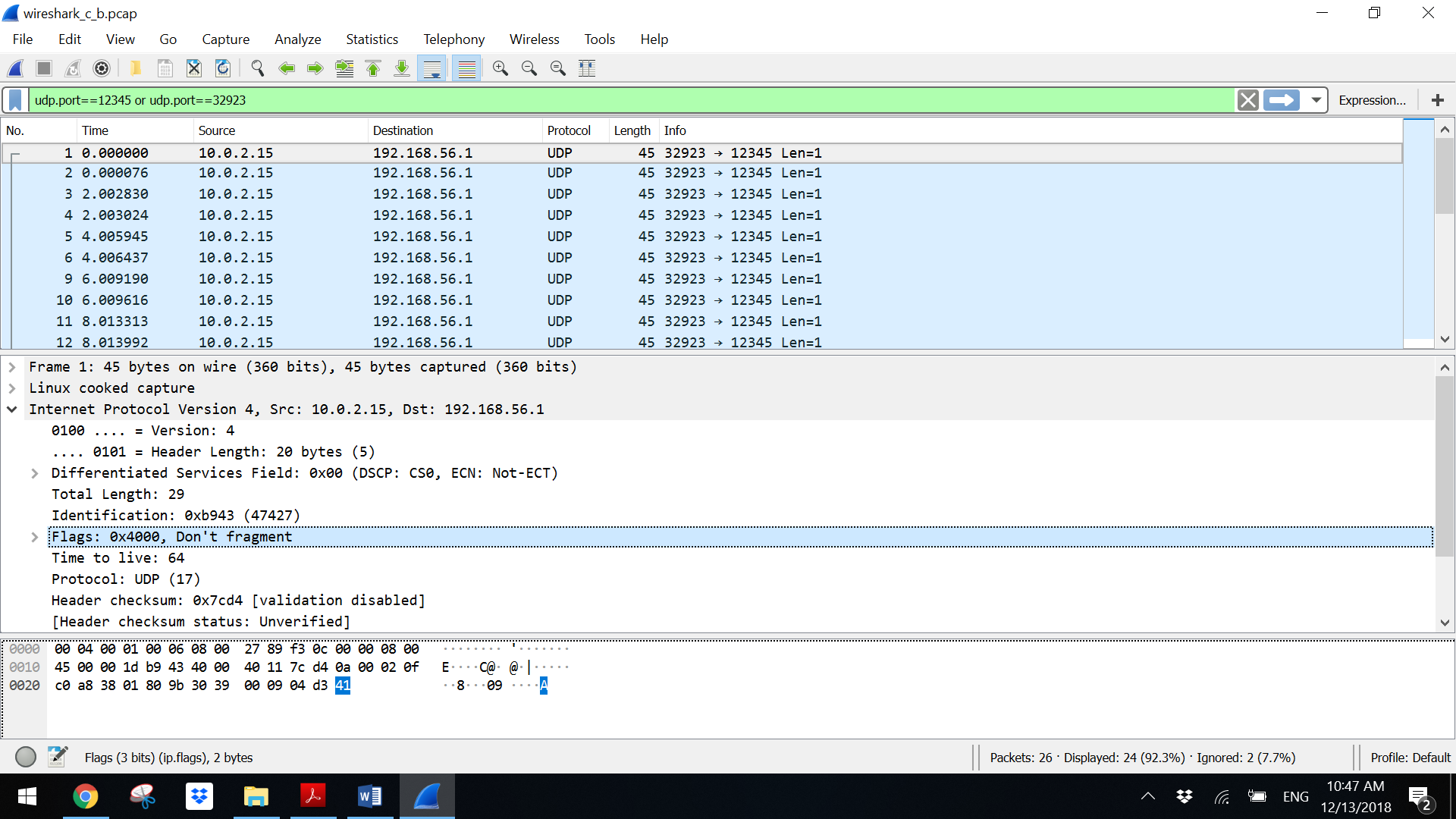


סעיף ב:

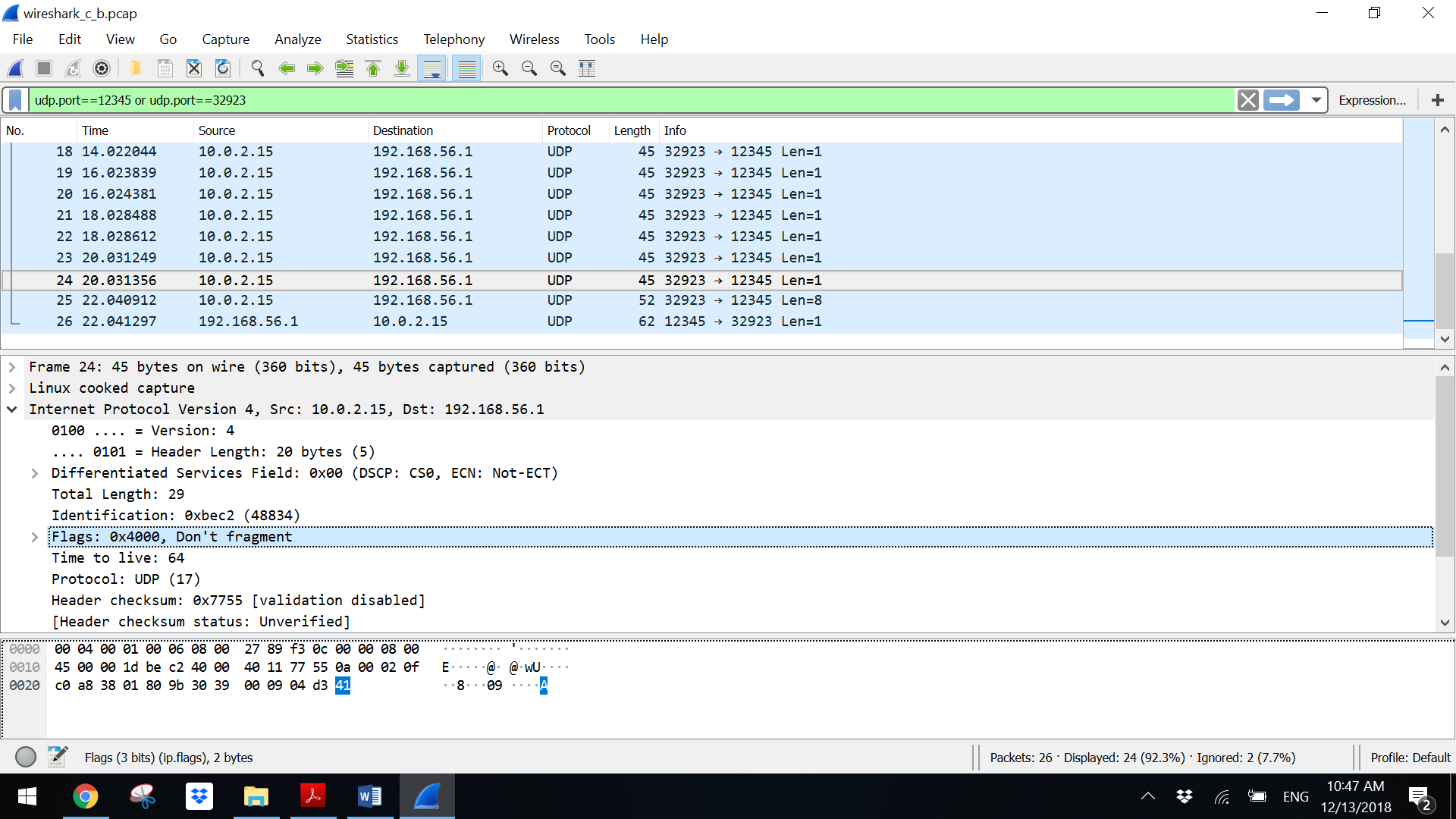
בסעיף זה אין צורך בפרגמנטצייה בשום שלב, זאת מפני שכל הודעה היא הודעה באורך קצר. לכן הפרוטוקול אינו מבצע פרגמנטצייה ושולח את המידע בהודעה כמו שהוא ללא שינוי.

ניתן לראות בהודעה הראשונה ששלחנו מהקליינט לשרת ובהודעה האחרונה מהקליינט לשרת, כי הדגל

Don’t Fragment דלוק, וזאת מפני שהפרוטוקול אל פירק אף אחד מהההודעות להודעות קטנות יותר (נבדק גם עבור כל ההודעות בין ההודעה הראשונה לאחרונה הדומות ביניהן)



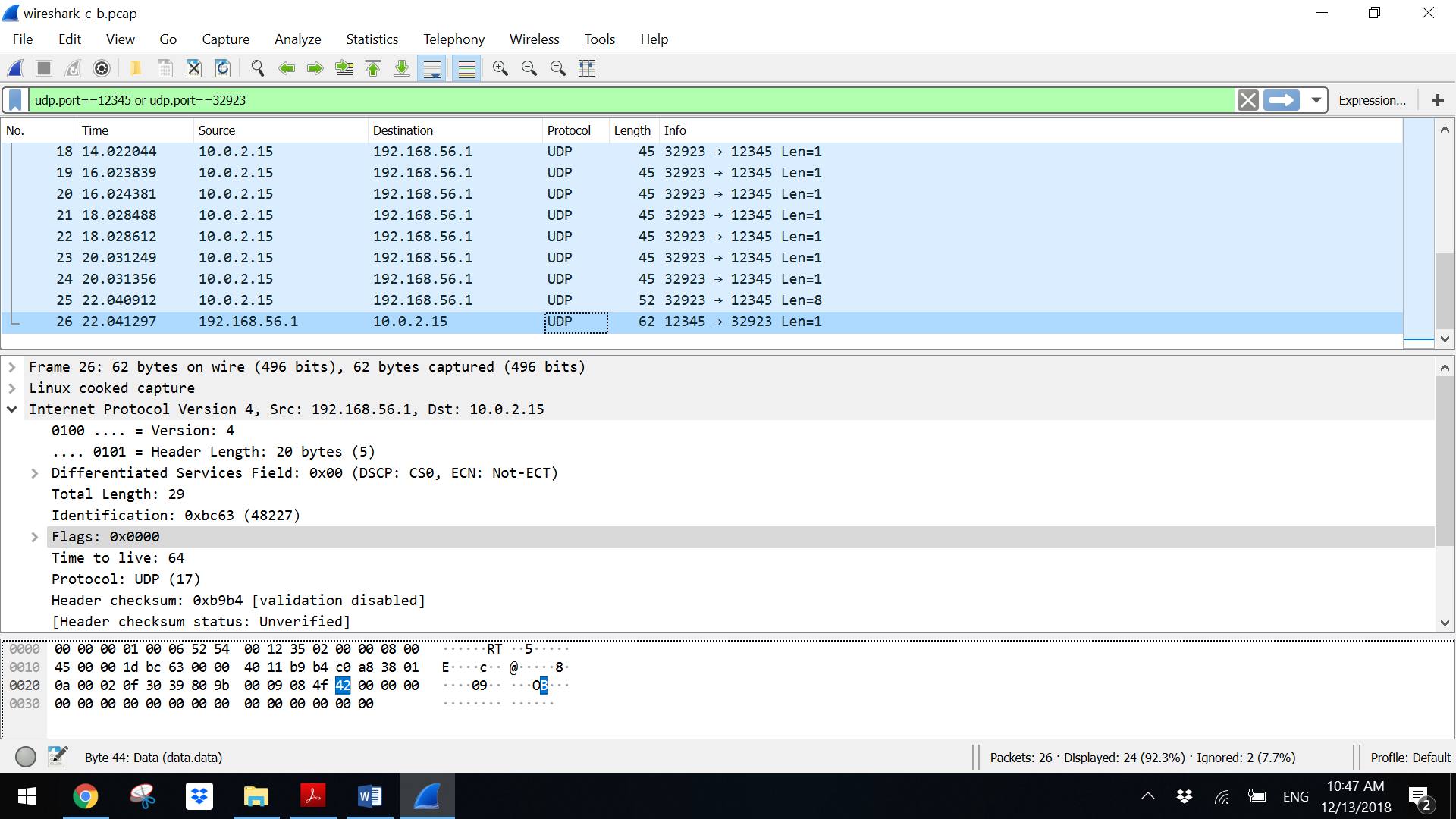
ההודעה האחרונה מהקליינט לשרת כפי שהסברנו:



הודעת \*FINISH\* ששלחנו על מנת להודיע לשרת כי סיימנו לשלוח את המידע הרצוי וכי הוא יכול להתחיל לבדוק את המידע.



לאחר שהשרת בדק את המידע שהועבר אליו, הוא מוודא שהמידע הוא בדיוק כמו המידע הרצוי, מפני שזהו המידע, ההודעה שמועברת בחזרה היא הודעה קצרה B, ללא צורך בפרגמנטצייה.



**ההבדלים בין TCP ל UDP:**

פרוטוקול TCP הינו פרוטוקול אשר מבצע סגמנטציה בשכבת התעבורה וזאת על מנת למנוע פרגמנטציה בשכב הרשת. לכן, כאשר אנו עושים שימוש בפרוטוקול TCP, לא נצפה לביצוע פרגמנטציה.

פרוטוקול TCP לא שולח רצף של הודעות, אלא רצף של בתים. לכן, צורת ההגעה של המידע בפרוטוקול TCP אינה משנה (מספר החבילות), מפני שהאפליקציה בשרת שלנו, תקרא בתים ולא תקרא הודעות. זאת בשונה מפרוטוקול UDP אשר בשימוש בו, האפליקציה תקרא הודעות ולא בתים.

בשימוש בפרוטוקול UDP, כל הודעה מחולקת לחבילה נפרדת, לכן השרת שלנו קורא את ההודעות בנפקד כברירת מחדל.

בשימוש בפרוטוקול TCP, אנו יודעים כי ניתן לבצע חיבור בקריאת מידע מתוך סוקט מפני שהתבצעה לחיצת ידיים. כך שאנו יכולים לקרוא מספר בתים בכל פעם בשל החיבור שהתבצע.

בשימוש בפרוטוקול UDP אשר לא מבצע חיבור, לקוחות שונים יכולים לשלוח מידע לאותו הסוקט, ללא בירור כמה בתים עלינו לקרוא בכל פעם. לכן, על פרוטוקול UDP לקרוא את כל ההודעה.