**מבוא לתקשורת מחשבים – תרגיל 2**

**מסמך נלווה :**

**צד לקוח – Client:**

בלולאה בפונקציית ה main – שם הקריאות מתחלקות בצורות שונות. את מס' הבקשה בכדי שהשרת ידע איזו קריאה הוא קיבל אנו שולחים כמחרוזת.

ישנן הקריאות לפעולות 1, 2, 3, 6, 7, 8, 9, 10, 11, 13 – פעולות אלו הן "פשוטות" ועובדות בצורת שליחת בקשה וקבלת תגובה ישירה (פעולה פשוטה).

בפעולה 4 – נשלחות 100 בקשות רציפות ולאחר מכן נקלטות 100 תגובות ברציפות.

בפעולה 5 – הלולאה עובדת 100 פעמים, בכל איטרציה נשלחת בקשה אחת ומתקבלת תגובה אחת.

בפעולה 12 – המערכת מבצעת שליחה של מידע פעמיים מהלקוח לשרת, בפעם הראשונה נשלח 12 ובעצם מודיע ללקוח שאנו מעוניינים לבצע זיהוי על עיר ובפעם השנייה הלקוח מודיע לאיזו עיר הוא מעוניין לקבל את המידע.

בפעולה 0 – הלקוח פשוט סוגר את האפליקציה ולא נשלחת אף בקשה.

הלוקח פועל ע"י המרה של הפקודה מ int ל ascii ושומר במשתנה sendBuff.

את התשובות הלוקח מקבל ב recvBuff ומציג אותם.

**צד שרת – Server:**

בפונקציית ה main של השרת קיימת לולאת while אשר תפקידה להשאיר את השרת דלוק תמידית.

השרת מחכה לבקשות וכאשר הן מגיעות הן יגיעו בפורמט של מחרוזת המכילה מס' בין 1 ל 13 המציין את בקשת הלקוח, לכל בקשה השרת עונה עם תשובה ייעודית, לבקשות 4 ו 5 השרת מתייחס כאל בקשה 1 ומפנה לאותה הפונקציה.

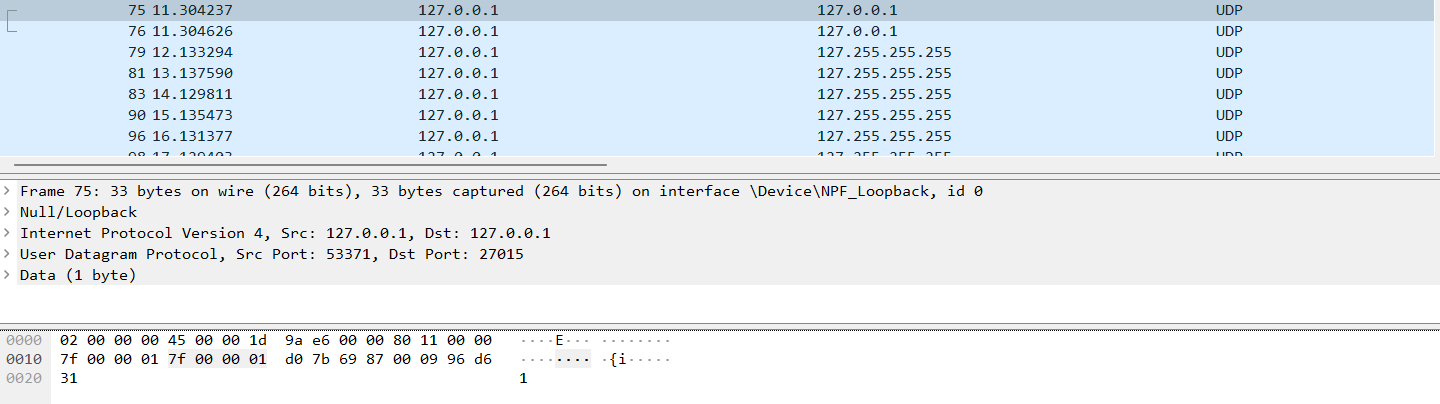
בקשה 13 היא החריגה מבין הבקשות מבחינת השרת – בכל תחילת איטרציה של לולאת ה while מתקיימת בדיקה אם אנחנו נמצאים באמצע של הקפת זמן (Time Lap), במידה ואנחנו אכן באמצע והזמן חרג מהתווך המותר (3 דקות) השרת יאפס את מצב המדידה.

השרת פועלה ע"י המרה של המשתנה recvBuff מ ascii ל int.

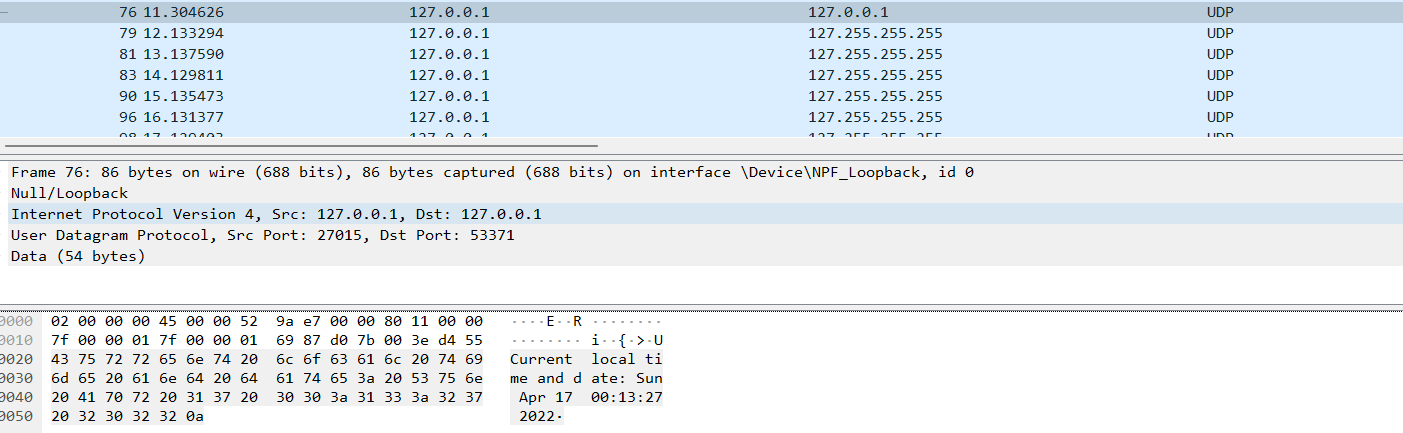
את התשובות השרת שומר כמחרוזות ב sendBuff אשר ישלח חזרה ללקוח כתגובה מהשרת.

**Wireshark:**

1. בקשה:



תגובה:



1. בקשה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תגובה:

תמונה שמכילה טקסט

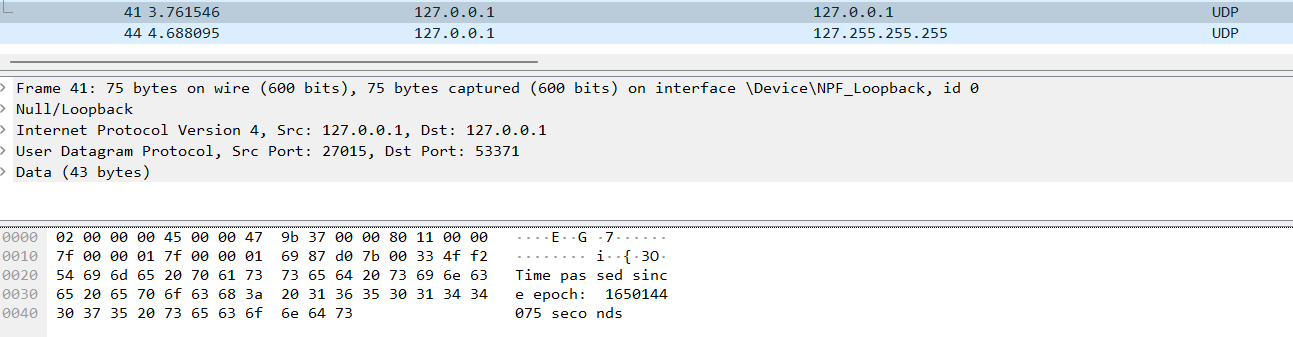
התיאור נוצר באופן אוטומטי

1. בקשה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תגובה:



1. בקשה:

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי

תגובה:

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי

1. בקשה:

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי

תגובה:

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי

1. בקשה:

תמונה שמכילה טקסט, צילום מסך, מקורה

התיאור נוצר באופן אוטומטי

תגובה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

1. בקשה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תגובה:

תמונה שמכילה טקסט

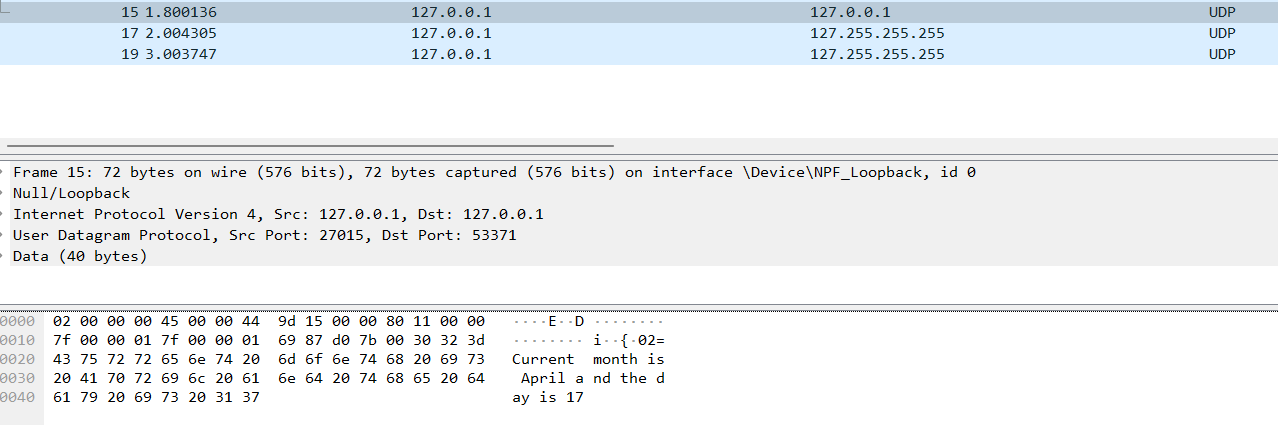
התיאור נוצר באופן אוטומטי

1. בקשה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תגובה:

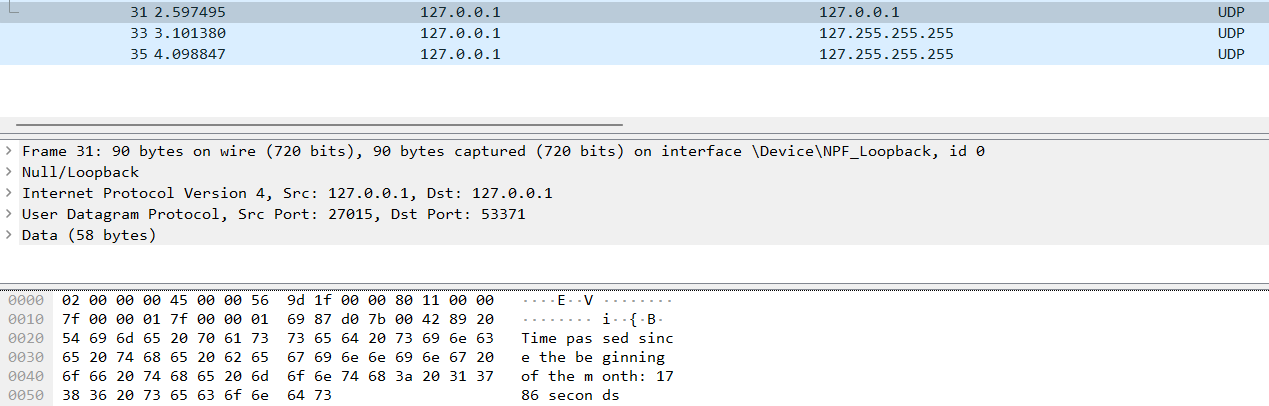


1. בקשה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תגובה:

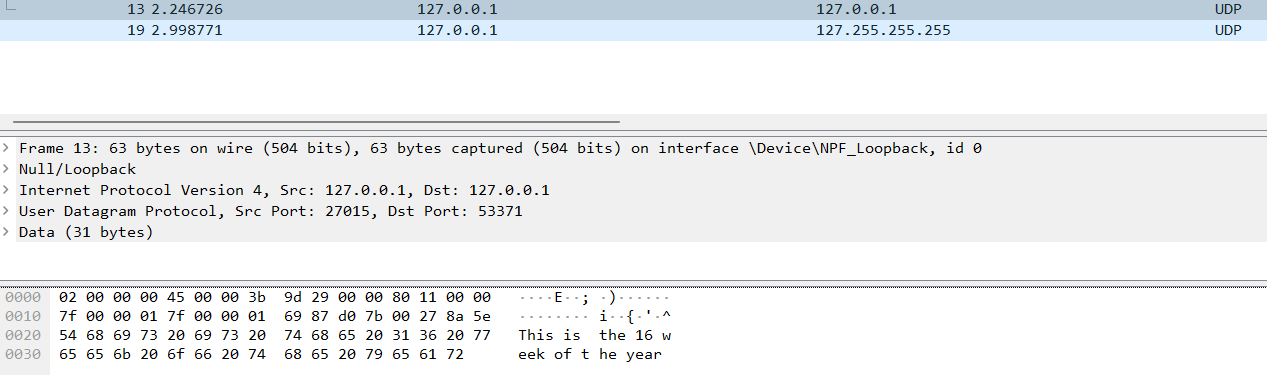


1. בקשה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תגובה:

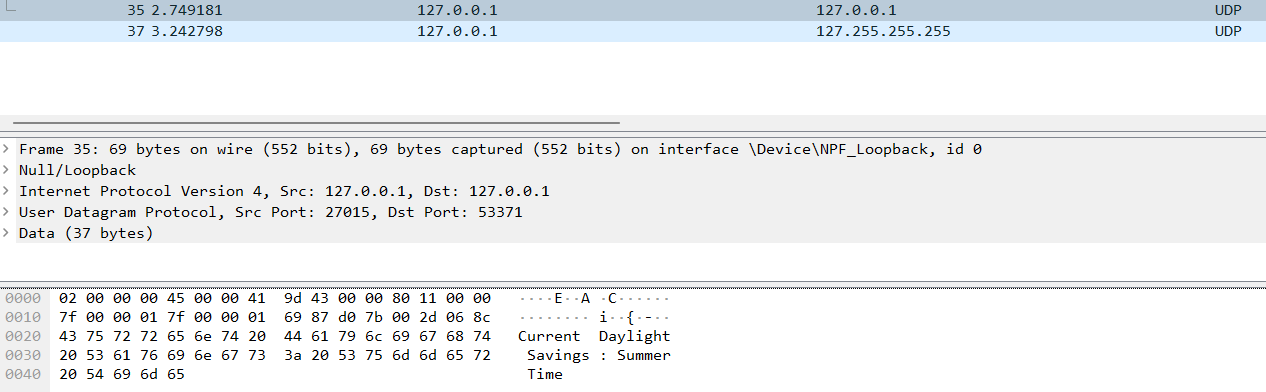


1. בקשה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תגובה:



1. בקשה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תגובה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

1. בקשה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

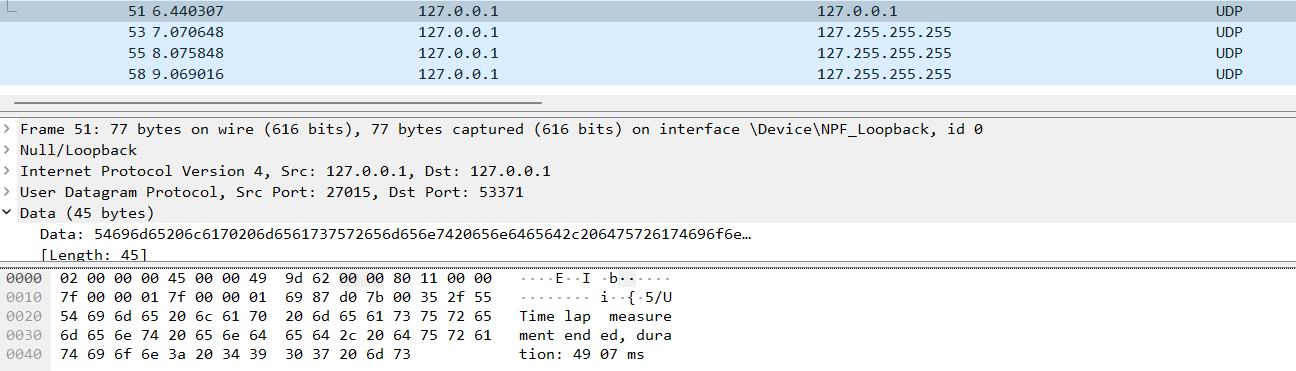
תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תגובה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי



**שאלות :**

* הריצו את התרגיל ובקשו בקשות GetClientToServerDelayEstimation רבות. האם ייתכן שהלקוח או השרת "ייתקעו"? אם לא, נמקו. אם כן, נמקו והציעו תיקון (ללא מימוש) בתוכנית שלכם בכדי שהלקוח והשרת לא "יתקעו".
* אכן ייתכן שהלקוח או השרת "ייתקעו", הפרוטוקול UDP איננו נחשב לפרוטוקול אמין להעברת מידע ולכן אין הבטחה שכל ההודעות הנשלחות יגיעו ליעדן. בעקבות זו קיים מצב בו אחת הבקשות של הלקוח לא תגיע ליעד אך הלקוח עדין יחכה לתשובה מהשרת ולכן "ייתקע". אפשרות נוספת אם לשרת יקבל בקשות רבות מלקוחות רבים הוא יוכל להגיע למצב של עומס וייתקע גם הוא. בכדי לפתור את המצב ניתן להקים מנגנון timeout בצד הלקוח, מנגנון זה ישמש כטיימר ויפעל לאחר שליחת כל בקשה, אם תגובה מהשרת לא הגיעה לאחר זמן מסוים תשלח בקשה נוספת, אם לאחר מכן שוב לא תגיע תגובה המנגנון יסמן ללקוח לסגור את החיבור אל השרת.
* מבחינת ההשהיות השונות שאנו כבר מכירים (אותן 4 השהיות שנלמדו בכיתה), מה מבטא הגודל הממוצע אותו הלקוח מחשב על סמך תשובות השרת בבקשת GetClientToServerDelayEstimation? (המלצה: נתחו גודל זה מבחינה תאורטית ולאו דווקא על סמך התוצאות המתקבלות מההרצה בפועל שעלולות להטעות, בפרט כאשר מבוצעות על אותו המחשב)