**שאלות תיאורטיות:**

1. כן, ייתכן שהלקוח או השרת "ייתקעו" במצב שבו יש כמות גדולה של בקשות GetClientToServerDelayEstimation . הסיבה לכך היא שכל בקשה כזו דורשת שליחת 100 בקשות וקבלת 100 תגובות, וללא תזמון נכון, הלקוח או השרת עלולים להיתקע בעיבוד הבקשות הרבות. לדוגמה :

* חוסר תיזמון נכון בעיבוד הבקשות: אם הלקוח או השרת לא מתזמנים כראוי את עיבוד הבקשות, הם עלולים להגיע למצב שבו יש יותר מדי בקשות פעילות במקביל ללא משאבים פנויים לטפל בהן.
* חוסר התחשבות בקצב העיבוד: אם הלקוח שולח בקשות בקצב גבוה מדי ללא התחשבות בקצב העיבוד של השרת, השרת עלול להיתקע בניסיון לעבד את כל הבקשות הרבות במקביל.
* חוסר ניהול נכון של משאבים: אם הלקוח או השרת לא מנהלים כראוי את המשאבים הזמינים (כגון זיכרון, חיבורי רשת וכדומה), הם עלולים לצרוך יותר מדי משאבים בעיבוד הבקשות הרבות ולהיתקע כתוצאה מכך.

תיקון אפשרי (ללא מימוש): ניתן למנוע "היתקעות" על ידי הגבלת מספר הבקשות GetClientToServerDelayEstimation שיכולות להתבצע במקביל. למשל, ניתן להגביל את מספר הבקשות הפעילות בכל נקודת זמן נתונה לערך מסוים (למשל, 5 בקשות במקביל). כאשר לקוח שולח בקשה GetClientToServerDelayEstimation, השרת יבדוק אם מספר הבקשות הפעילות נמוך מהמקסימום המותר. אם כן, הבקשה תתקבל לעיבוד, אחרת הבקשה תידחה או תועמד בתור המתנה. בצד הלקוח, ניתן ליישם מנגנון דומה להגבלת מספר הבקשות הפעילות במקביל.

1. מבחינה תיאורטית, הגודל הממוצע שהלקוח מחשב על סמך תשובות השרת בבקשת GetClientToServerDelayEstimation מבטא את סך ארבע ההשהיות הבאות:
2. השהיית שידור (Transmission Delay) - הזמן הדרוש לשידור החבילה מהלקוח לרשת.
3. השהיית תפוגה (Propagation Delay) - הזמן הדרוש לחבילה לעבור דרך הרשת מהלקוח לשרת.
4. השהיית תור (Queueing Delay) - הזמן שהחבילה ממתינה בתורים במעבדי הרשת לאורך הדרך מהלקוח לשרת.
5. זמן העיבוד בשרת (Processing Delay) - הזמן שלוקח לשרת לעבד את החבילה ולהכין תשובה.

במהלך המדידה, הלקוח שולח 100 בקשות "מה הזמן?" לשרת ומקבל 100 תשובות. השרת שולח בתשובה את חותמת הזמן הנוכחית כפי שנמדדת על ידי GetTickCount(). הלקוח מחשב את ההפרש בין חותמת הזמן שהשרת שלח לבין הזמן הנוכחי בלקוח עבור כל בקשה ותשובה, ומחשב את הממוצע עבור ההפרשים הללו על פני 100 הבקשות.

לכן, הגודל הממוצע שמחושב על ידי הלקוח מבטא בפועל את סך ארבע ההשהיות הללו בדרך מהלקוח לשרת וחזרה. יש לציין שבמצב שהלקוח והשרת רצים על אותו מחשב, השהיית התפוגה והשהיית התור יהיו זניחות, ולכן הגודל הממוצע ישקף בעיקר את השהיית השידור והשהיית העיבוד בשרת. אך מבחינה תיאורטית, הגודל הממוצע כולל את כל ארבע ההשהיות הללו.

תיאור הפרוטוקול :

אפליקציית הלקוח מאפשרת למשתמש לבחור מתוך רשימה של בקשות ולשלוח אותן לשרת. הלקוח שולח את הבקשה לשרת באמצעות פרוטוקול UDP ומחכה לתשובה מהשרת. כל הודעה היא מחרוזת (string) שמייצגת פעולה מסוימת שהשרת צריך לבצע. השרת מפענח את ההודעה ומחזיר תשובה בהתאם, ההודעות הן מחרוזות פשוטות שמכילות את שם הפעולה. בחלק מהמקרים יש צורך בפרמטר נוסף (כגון שם עיר), שמופרד מהפעולה באמצעות רווח. התשובות הן גם מחרוזות שמכילות את המידע המבוקש. כל הודעה נבדקת בעזרת פונקציה מרכזית שמפנה לפונקציות עזר לפי סוג הבקשה. פונקציות העזר מבצעות את הפעולות המתאימות ומחזירות את התשובות ללקוח

בחרתי להשתמש במחרוזת (string) להעברת הבקשה מהלקוח לשרת ממספר סיבות עיקריות:

**1-פשטות וקלות מימוש**

מחרוזות הן סוג נתונים פשוט ונוח לשימוש. המחרוזות מאפשרות לנו להגדיר ולשלוח בקשות בצורה קריאה וברורה. שימוש במחרוזות מאפשר לנו לבצע פעולות כמו השוואת בקשות והעתקתן בקלות רבה.

**2-גמישות**

מחרוזות מאפשרות גמישות רבה בהגדרת הבקשות. ניתן להוסיף, לשנות ולהרחיב את הבקשות בקלות רבה, מבלי להידרש לשינויים משמעותיים בקוד.

**3-תאימות לפורמט קריא לבני אדם**

מחרוזות הן פורמט קריא לבני אדם, מה שמקל על בדיקה (debugging) . כאשר נשלחת בקשה במחרוזת, ניתן לראות בקלות מה נשלח ומה התקבל, דבר שמקל מאוד על תהליך הפיתוח והתחזוקה של המערכת.

מבנה הבקשה :

בפרוטוקול שתוכנן למערכת זו, ההודעות מועברות במחרוזות טקסטואליות בפורמט ASCII .

**גודל הודעה מקסימלי:**

* בקשה: 255 תווים

|  |  |
| --- | --- |
| **parameters** | **OpCode** |

* **OpCode** קוד פעולה המתאר את סוג הבקשה לדוגמה, "GetTime" לבקשת השעה הנוכחית.
* **Parameters** פרמטרים נוספים הנדרשים לבקשה, אם ישנם (לדוגמה, שם העיר בבקשה של שעה בעיר מסוימת).
* תשובה : תשובה: 255 תווים

|  |  |
| --- | --- |
| **responseMessage** | **OpCode** |

* **OpCode**: קוד הפעולה שהלקוח שלח בבקשה.
* **responseMessage**: הודעת התשובה המלאה שהשרת מחזיר ללקוח.

**Wireshark Screenshots:**

בקשה של זמן ותאריך :

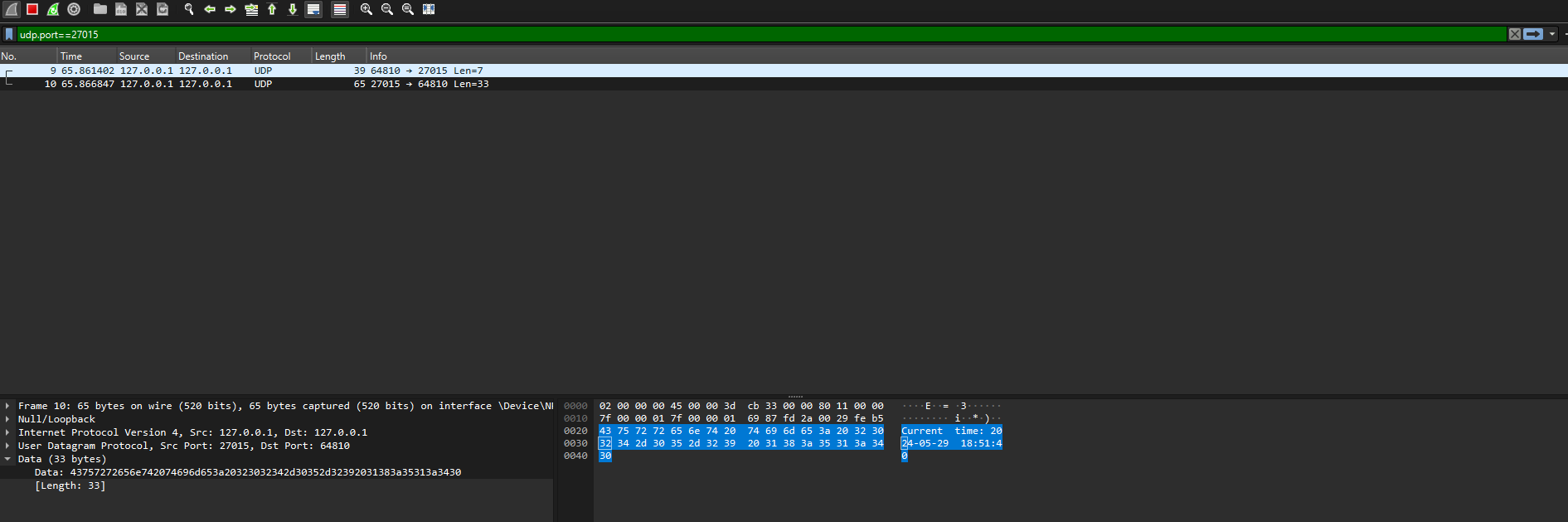
בקשה: "GetTime"

תמונה שמכילה צילום מסך, טקסט, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי

תשובה :

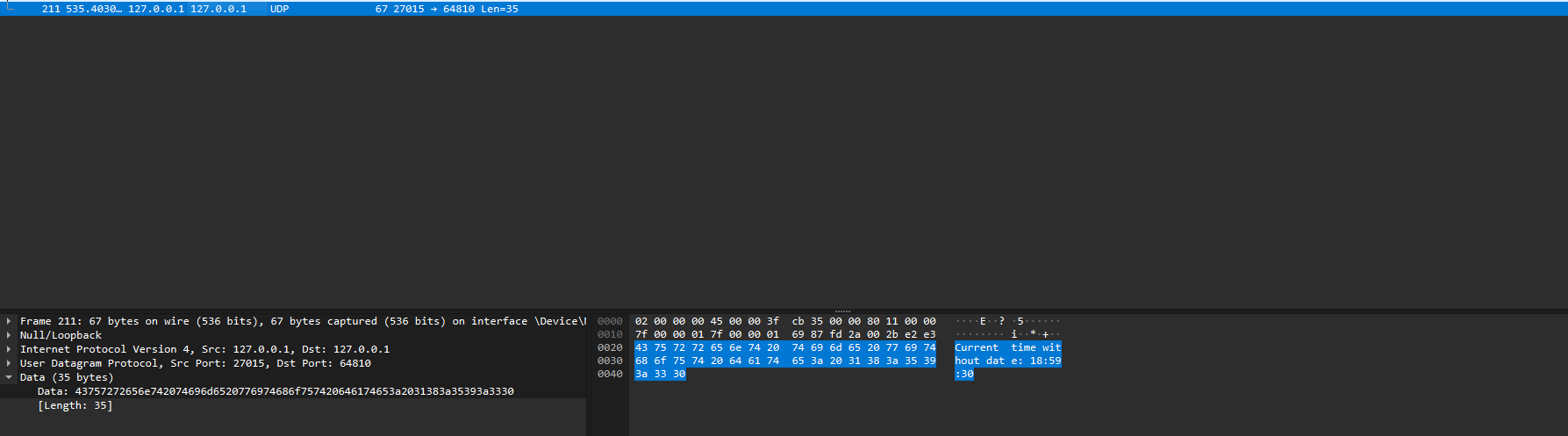
Current time: 2024-05-29 18:51:40



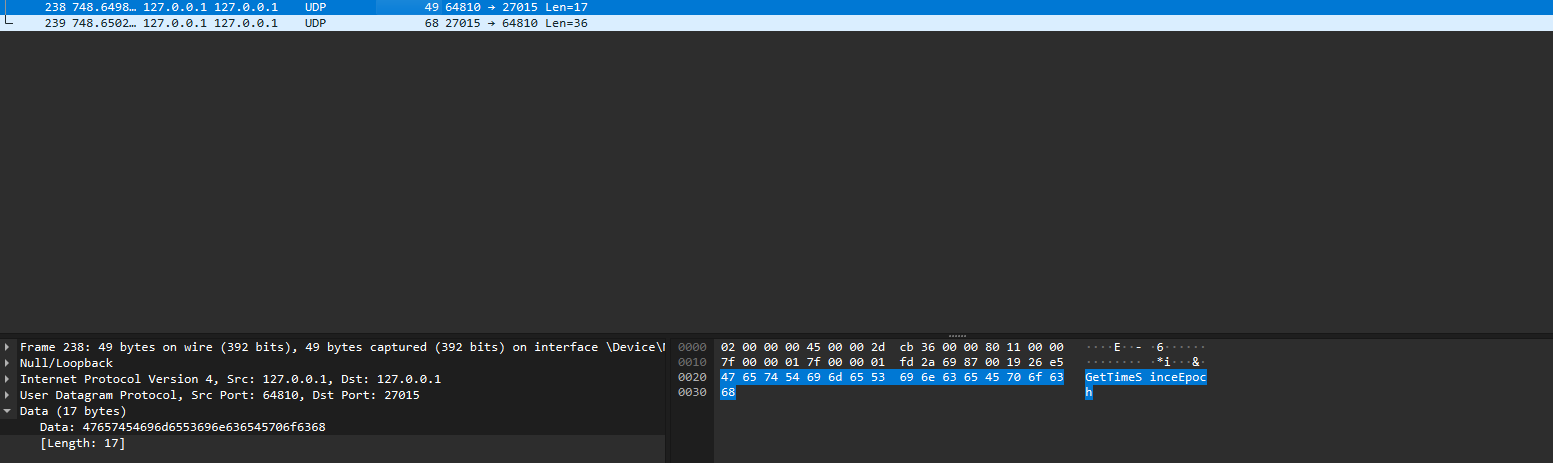
בקשה של זמן בלי תאריך :

תמונה שמכילה צילום מסך, טקסט, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטיבקשה: GetTimeWithoutDate"

תשובה : "Current time without date: 18:59:30"

בקשה של זמן בשניות לפי GetTimeSinceEpoch:

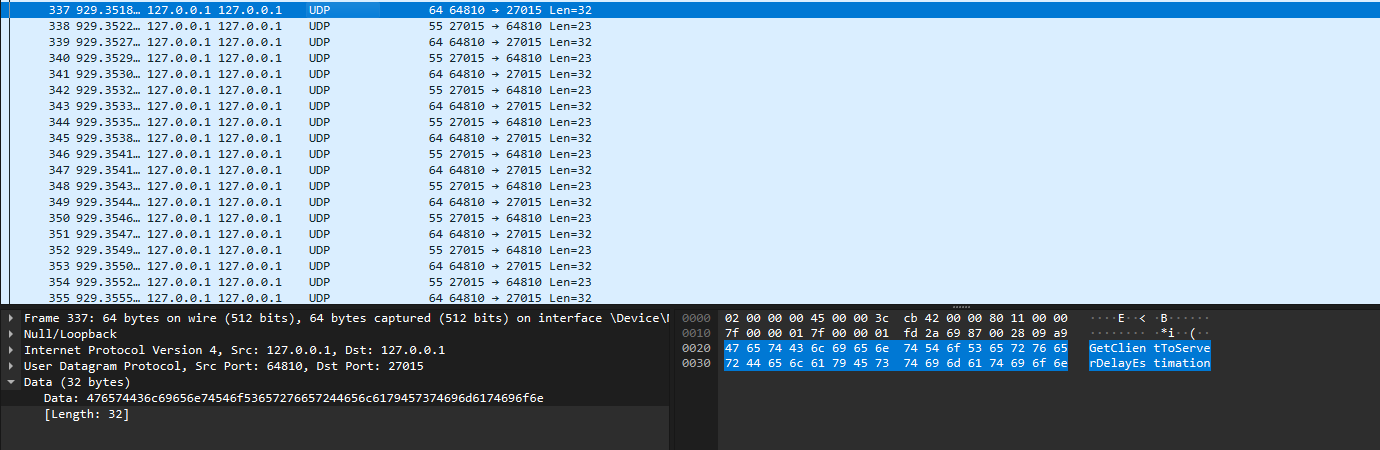
בקשה : **"GetTimeSinceEpoch"**

תמונה שמכילה צילום מסך, טקסט, תכונות מולטימדיה, תוכנה

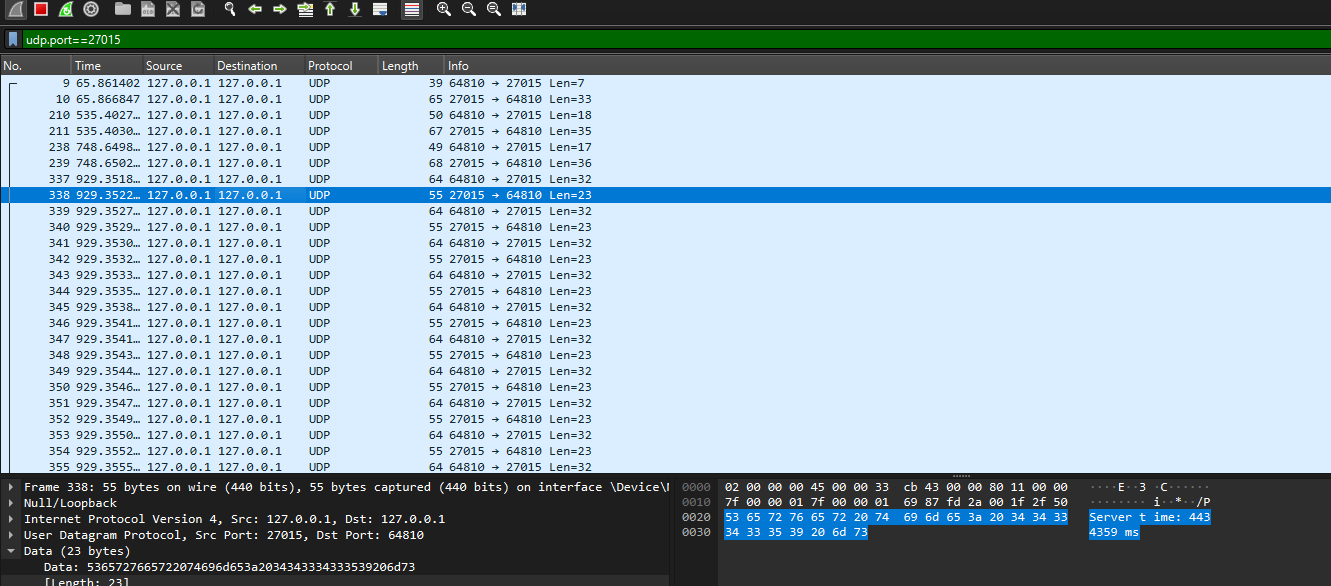
התיאור נוצר באופן אוטומטיתשובה : "Time since epoch: 1716998583 seconds"

בקשה של מיצוע הזמן בין הודעות העוברות מהלקוח לשרת:

בקשה : נשלח 100 בקשות רצופות "GetClientToServerDelayEstimation"



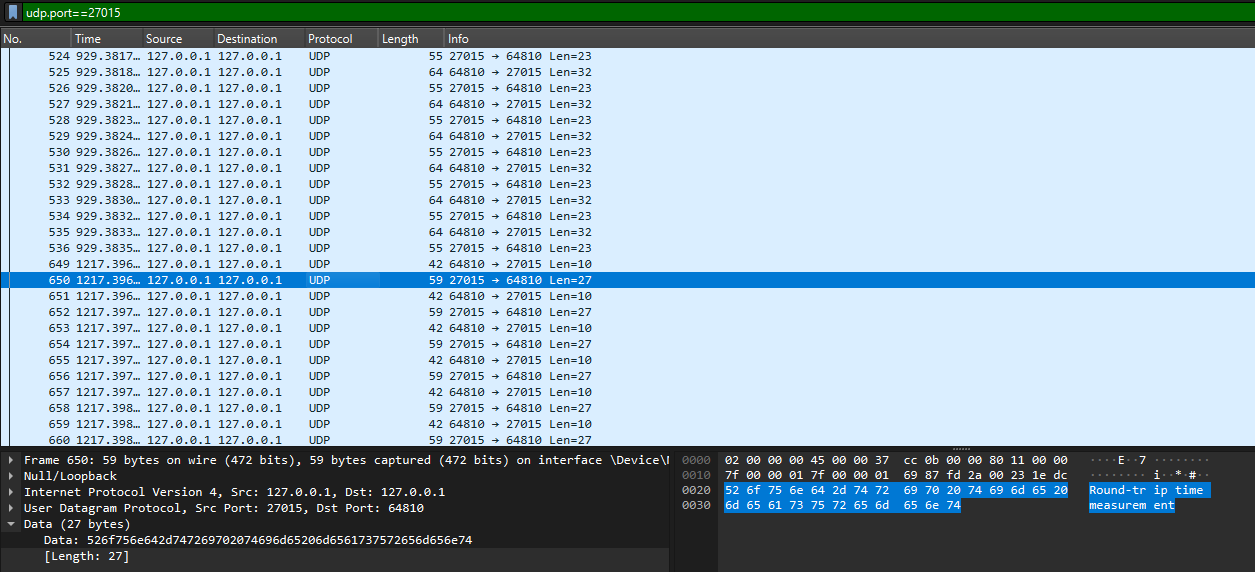
תשובה : "Server time: 4434359 ms" ( דוגמא עבור ההודעה הראשונה)



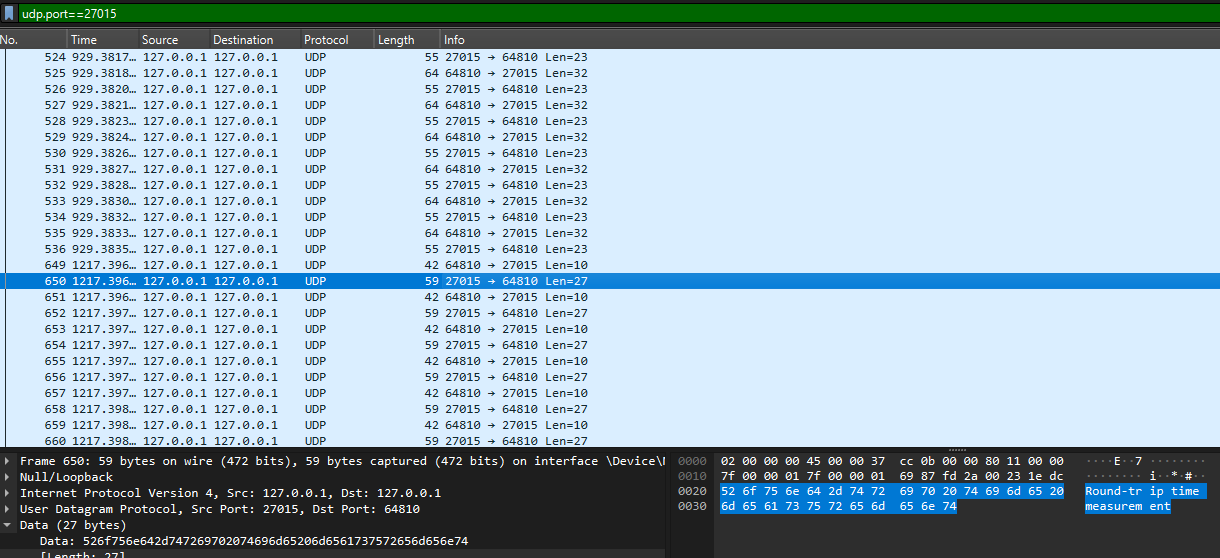
החישוב יקרה לאחר קבלת 100 התשובות אצל הלקוח.

בקשה של מיצוע של ה- RTT בין הלקוח לשרת:

בקשה: נשלח 100 בקשות **MeasureRTT**

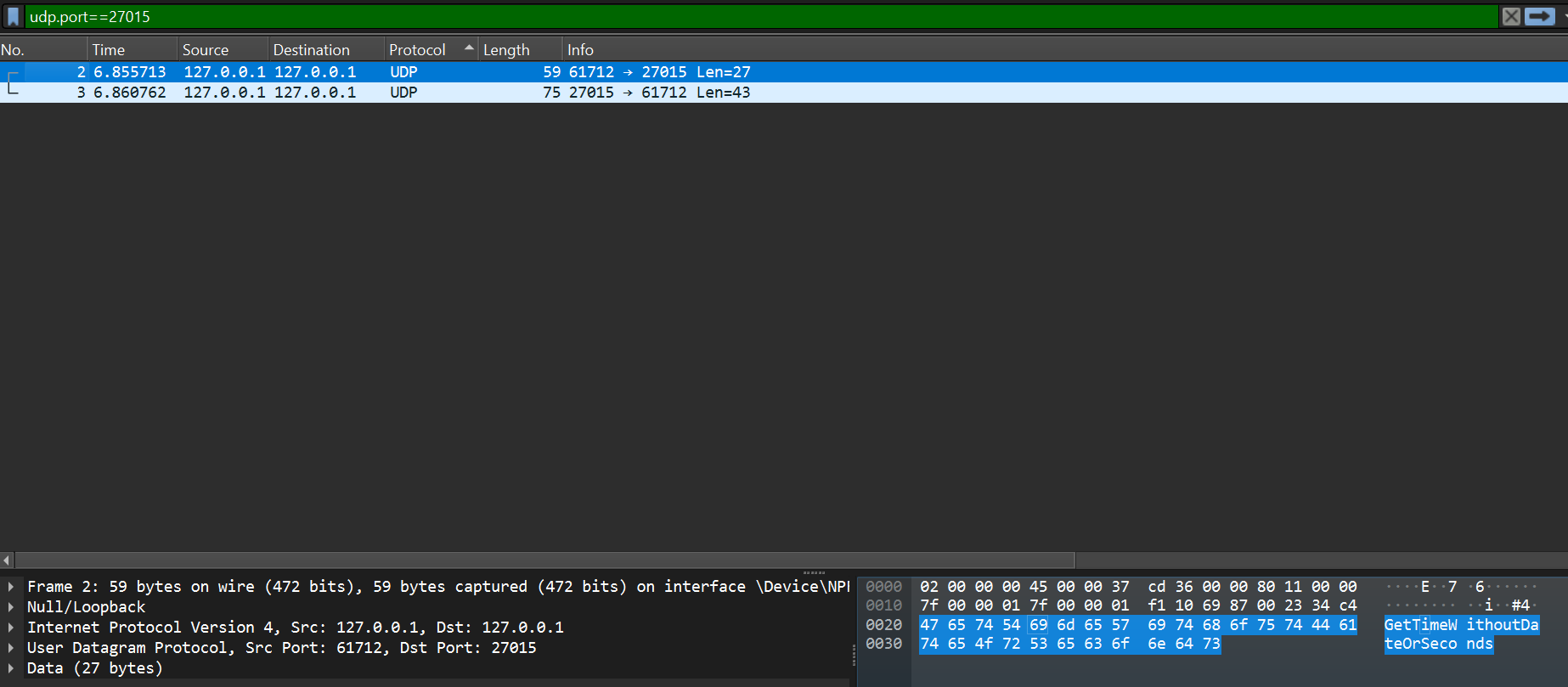


תשובה : 100 Round-trip time measurement

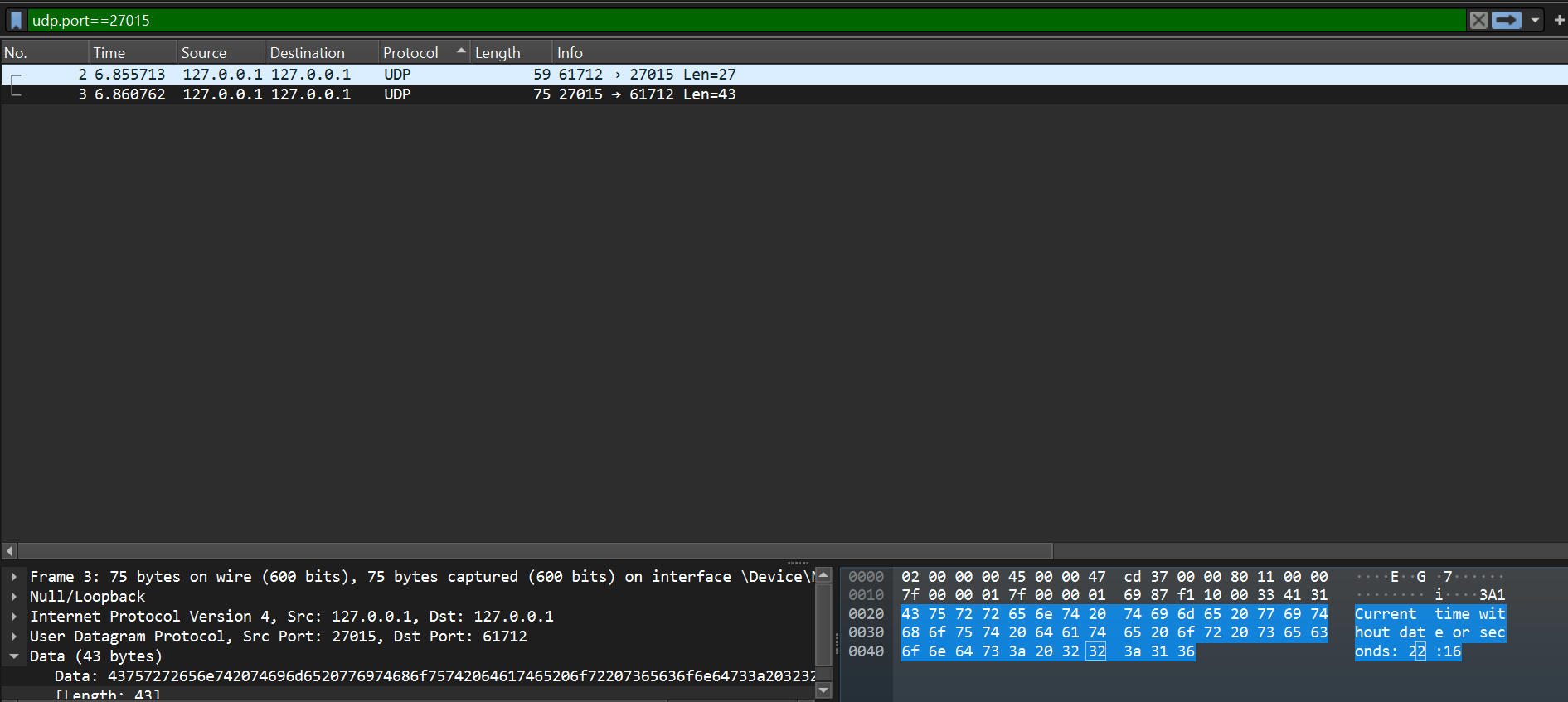


* המדידה מתבצעת אצל הלקוח מרגע שליחת הבקשה עד קבלת התשובה.  
  לכן תוכן התשובה של השרת אינה חשובה לנו והממוצע יחושב אצל הלקוח.

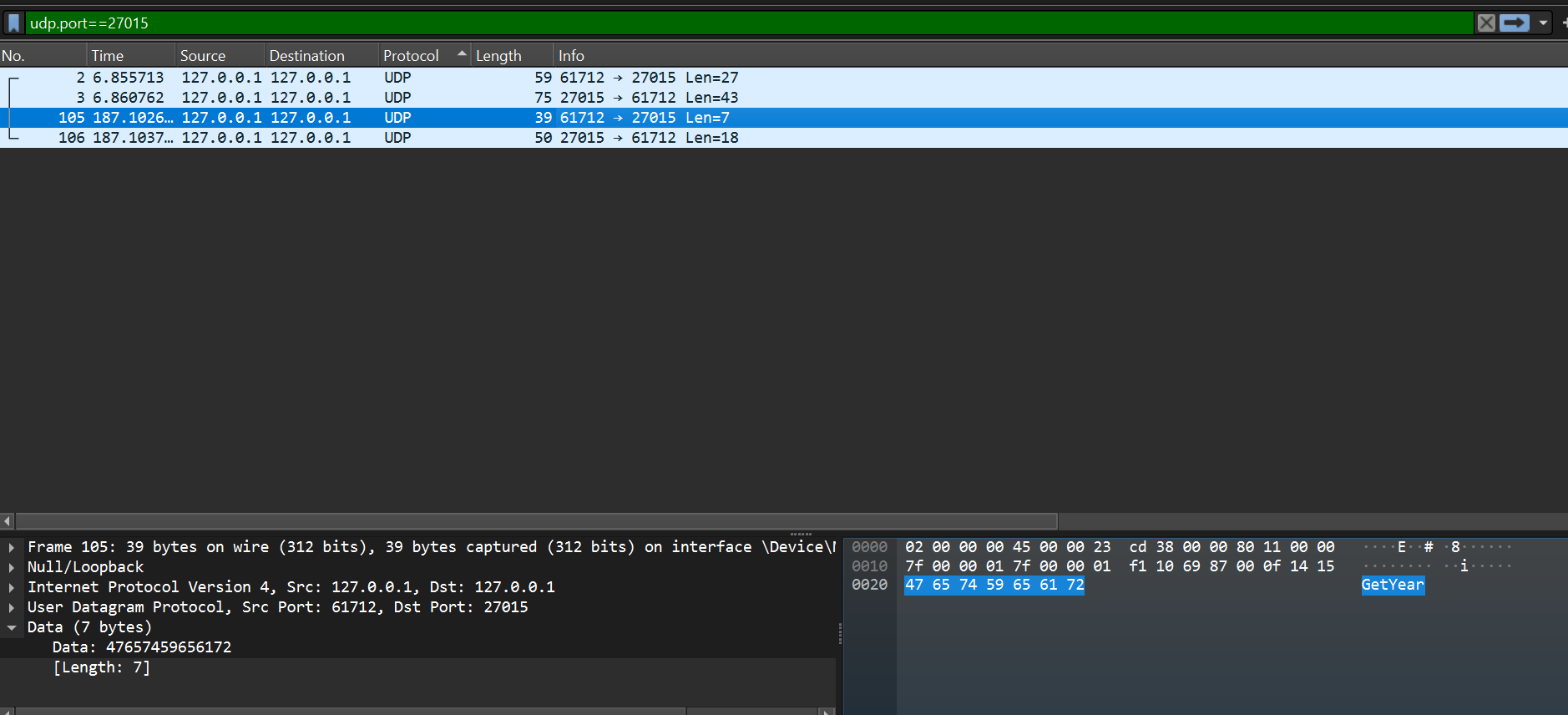
בקשה של הזמן ללא שניות וללא תאריך:

בקשה: נשלח "GetTimeWithoutDateOrSeconds"

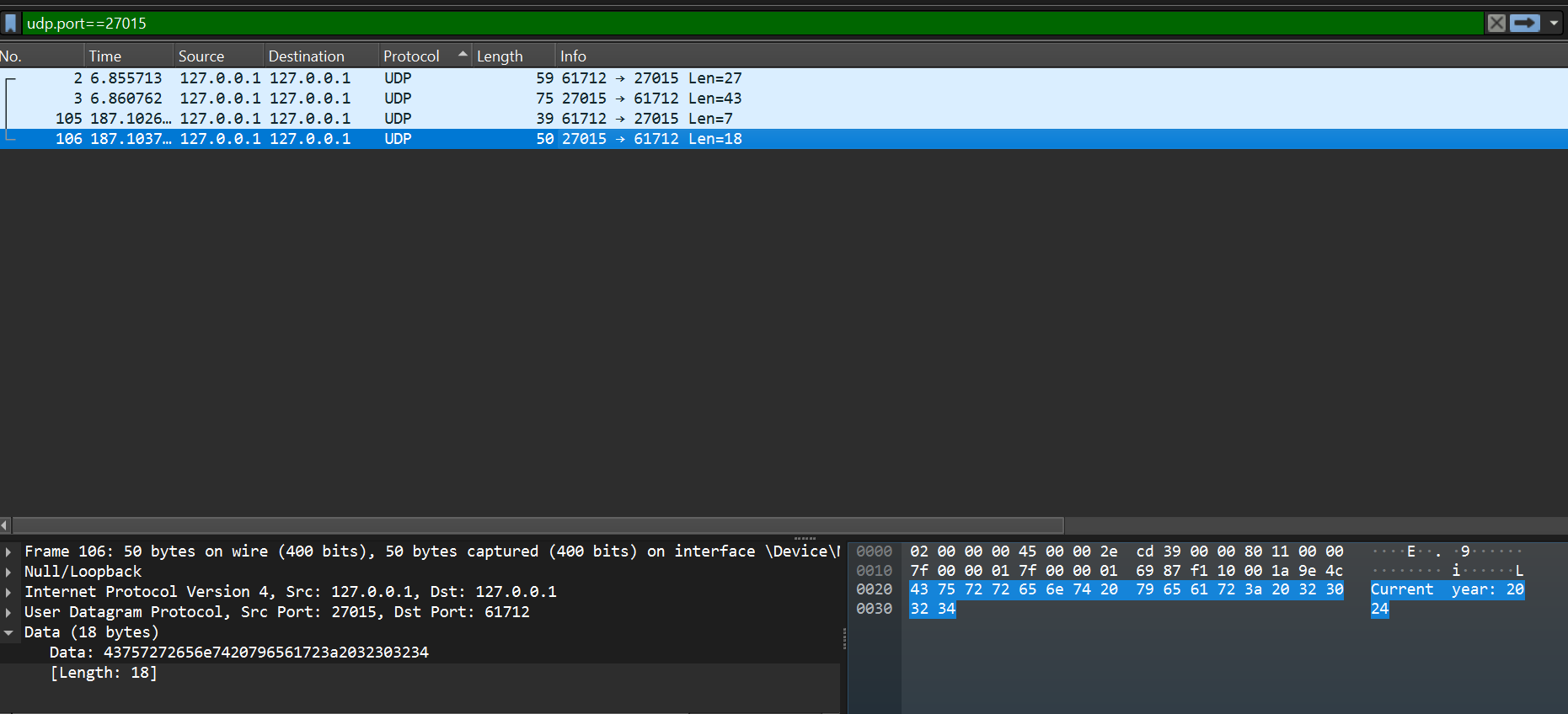
תשובה : "Current time without date or seconds: 22:16"



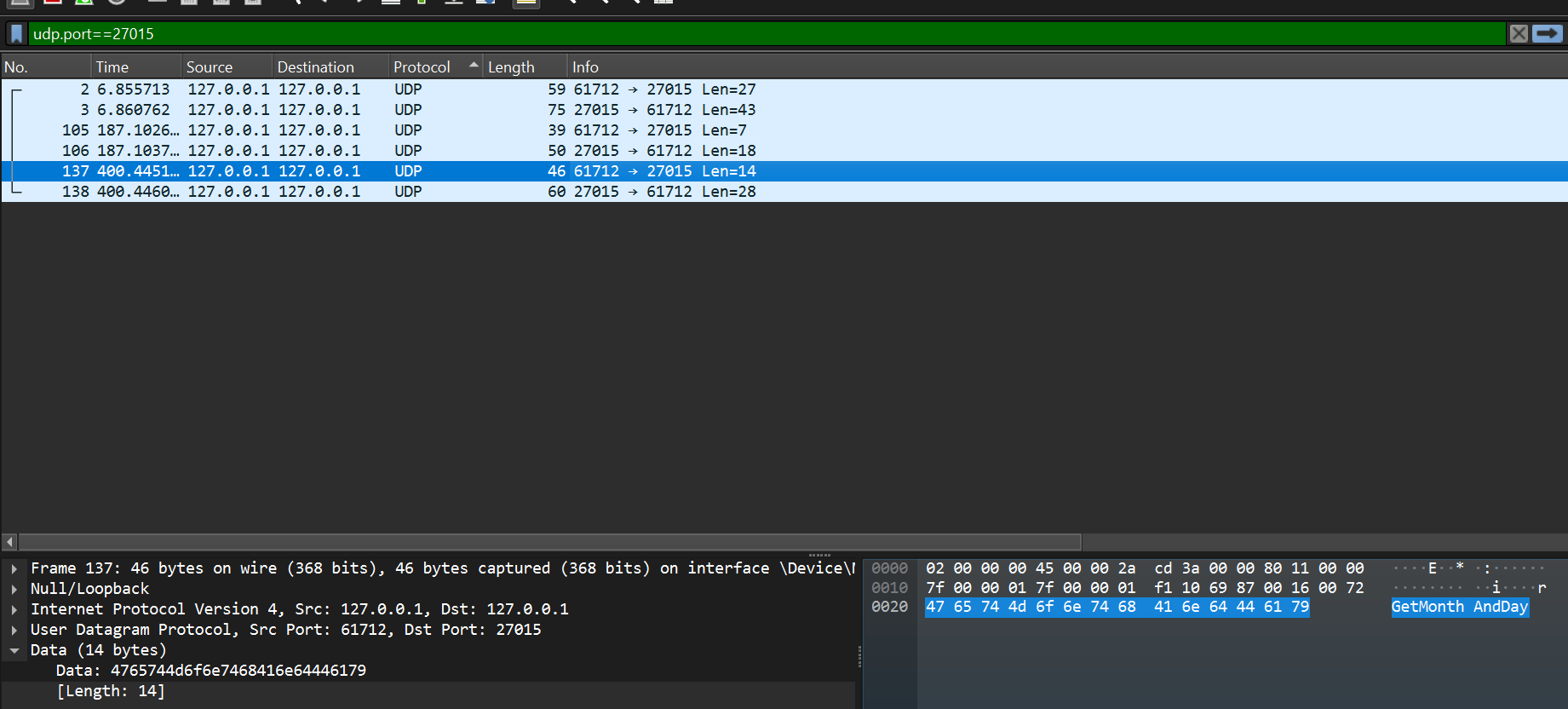
בקשה של השנה:

בקשה: נשלח " "GetYear

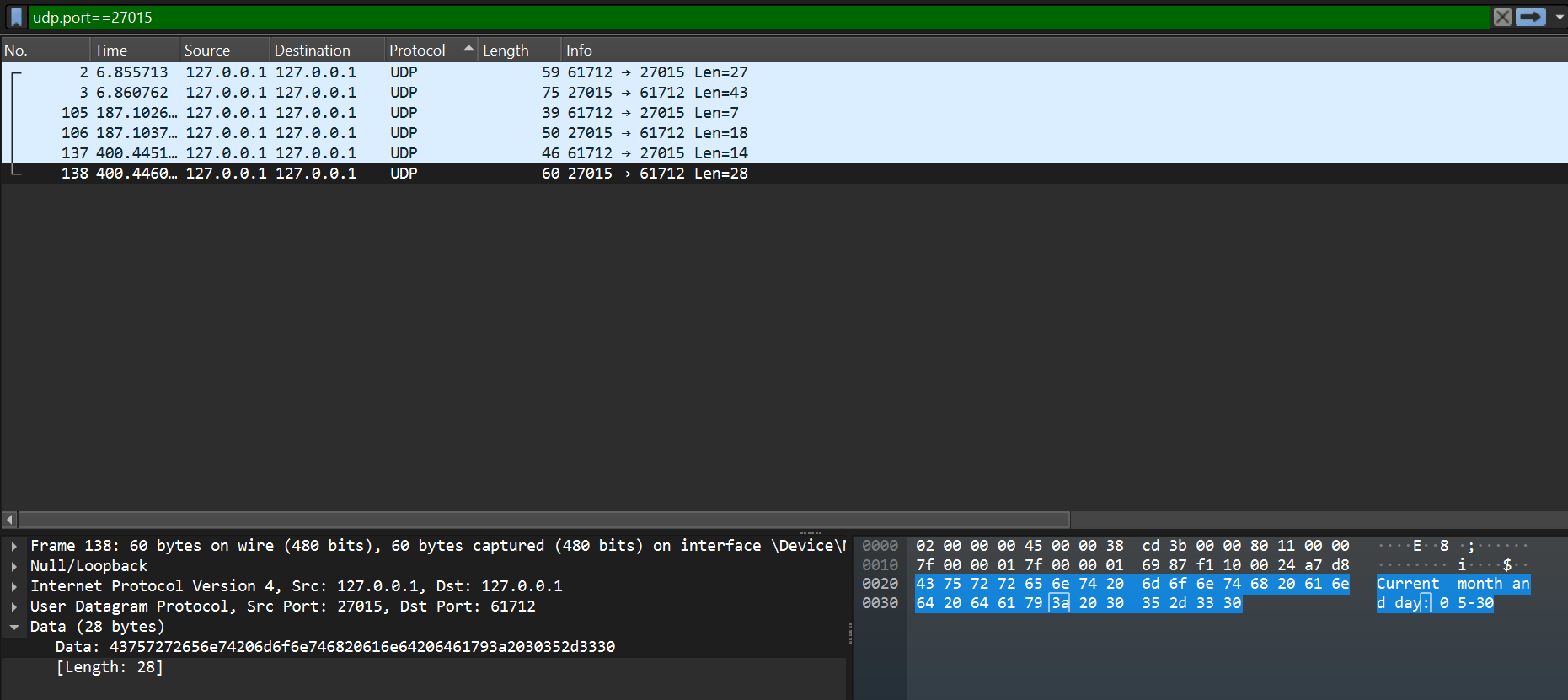
תשובה : "Current year: 2024"



בקשה של חודש ויום:

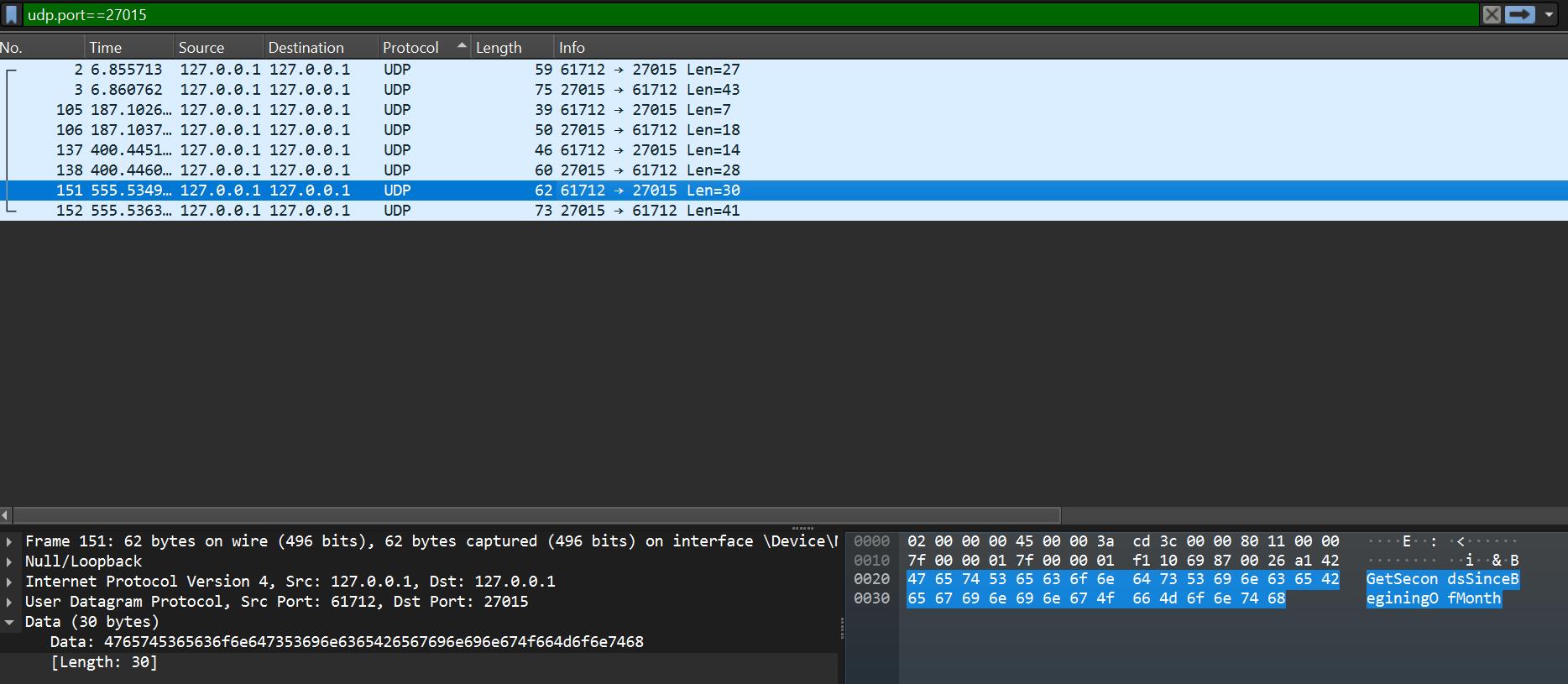
בקשה: נשלח "GetMonthAndDay" 

תשובה : "Current month and day: 05-30"

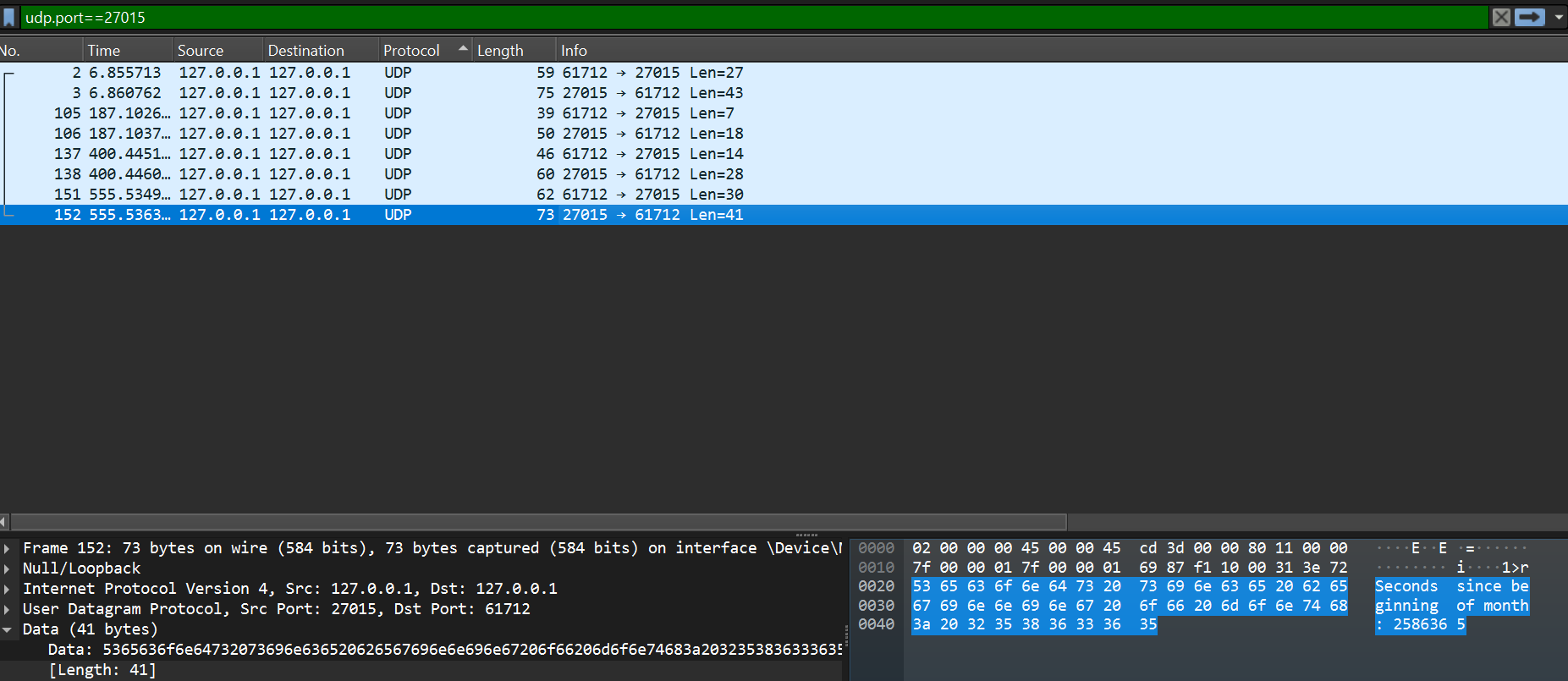


בקשה של הזמן בשניות שעבר מתחילת החודש:

הבקשה: "GetSecondsSinceBeginingOfMonth"

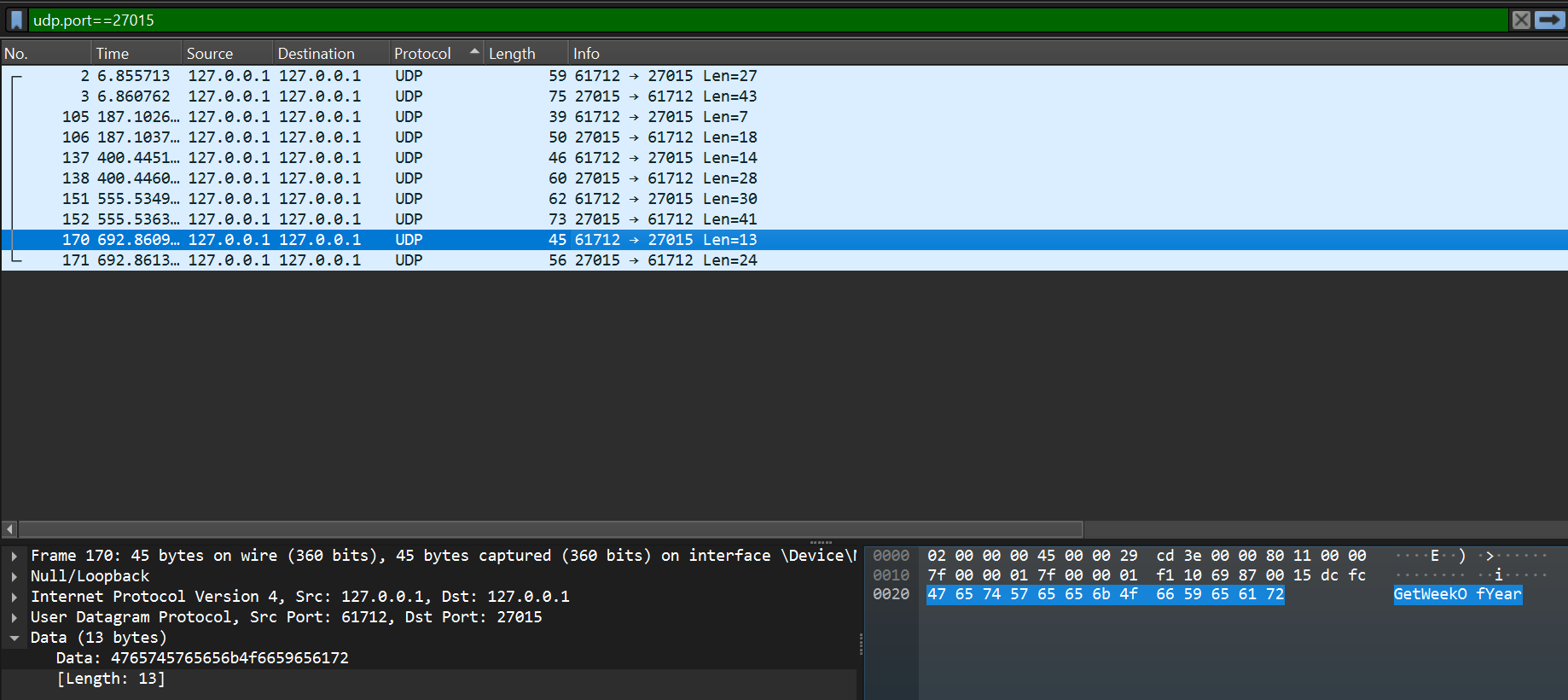


תשובה : "Seconds since beginning of month: 2586365"

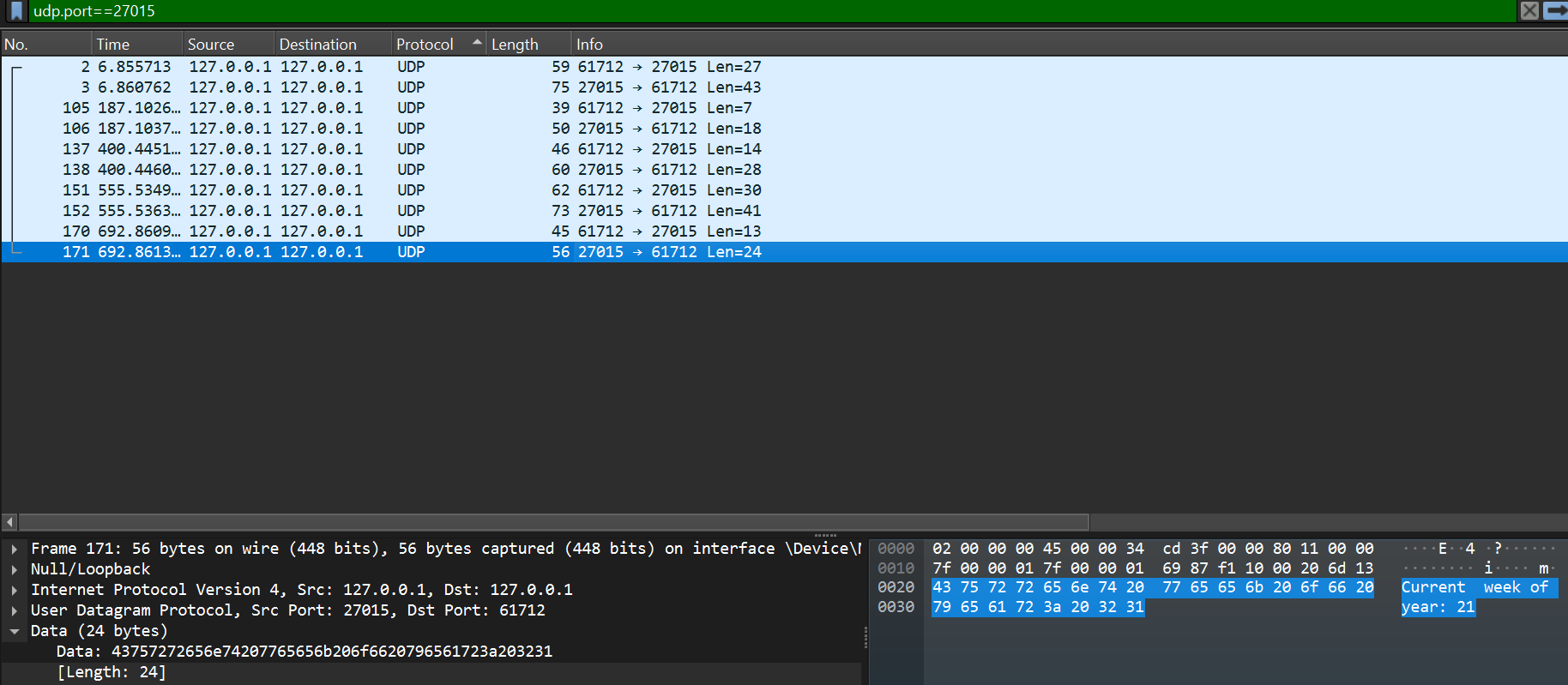


בקשה של מספר השבוע בשנה:

הבקשה: "GetWeekOfYear"

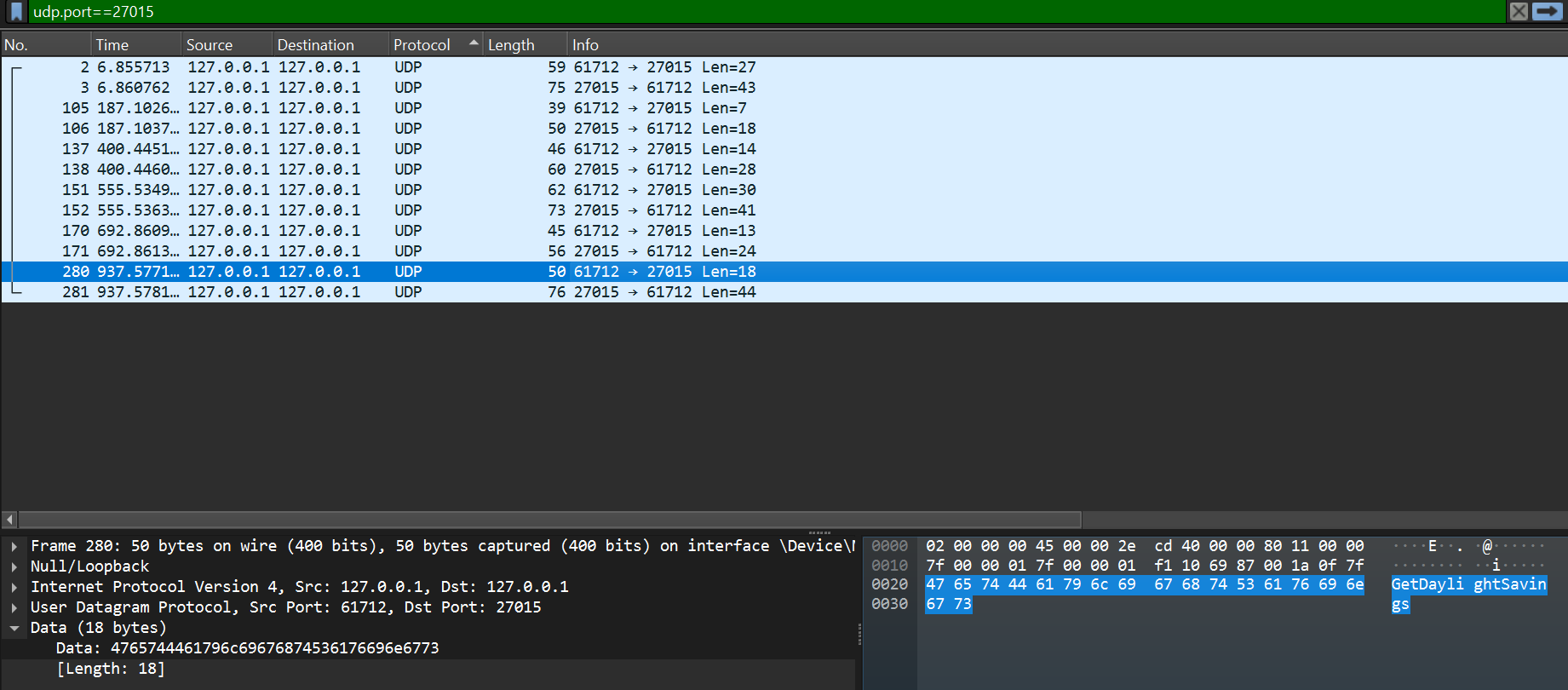


תשובה: "Current week of year: 21"

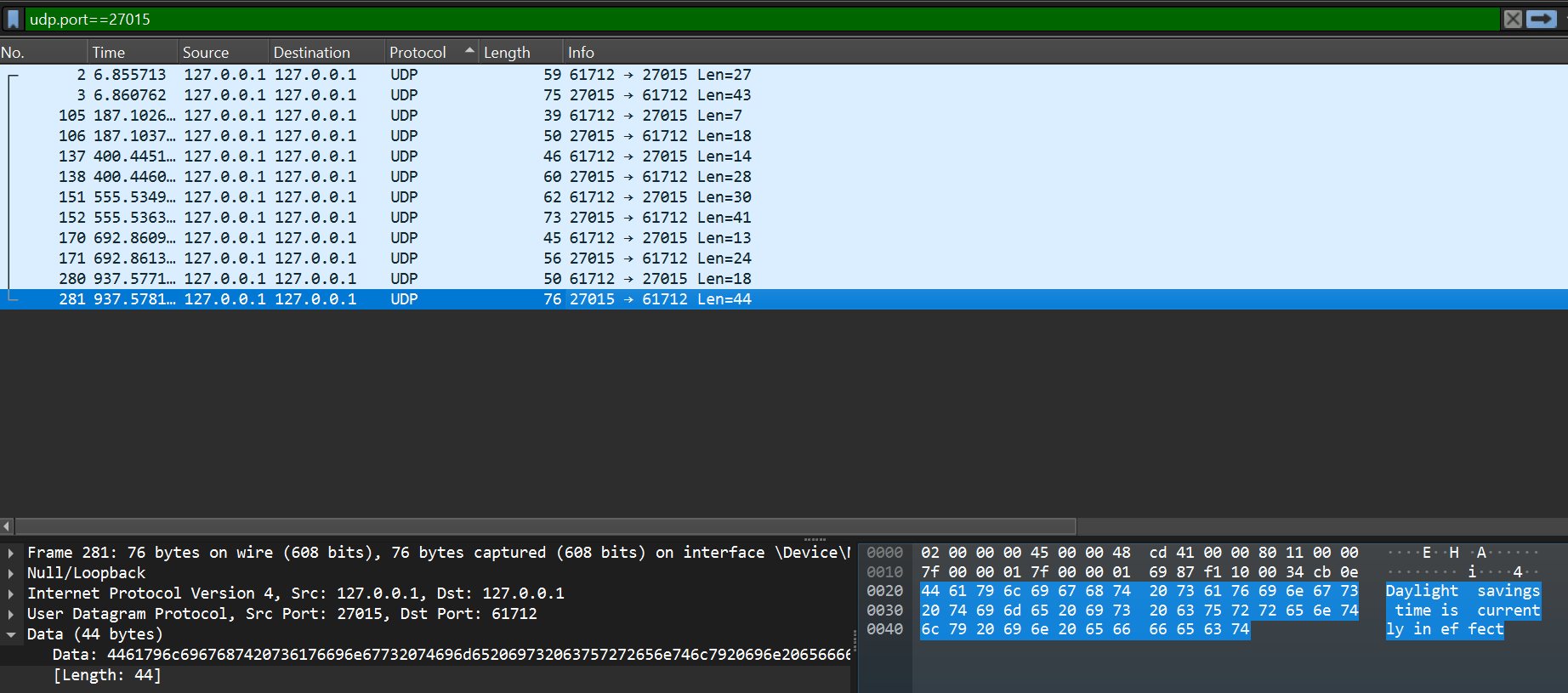


בקשה של סטטוס שעון חורף\קיץ:

הבקשה : "GetDaylightSavings"



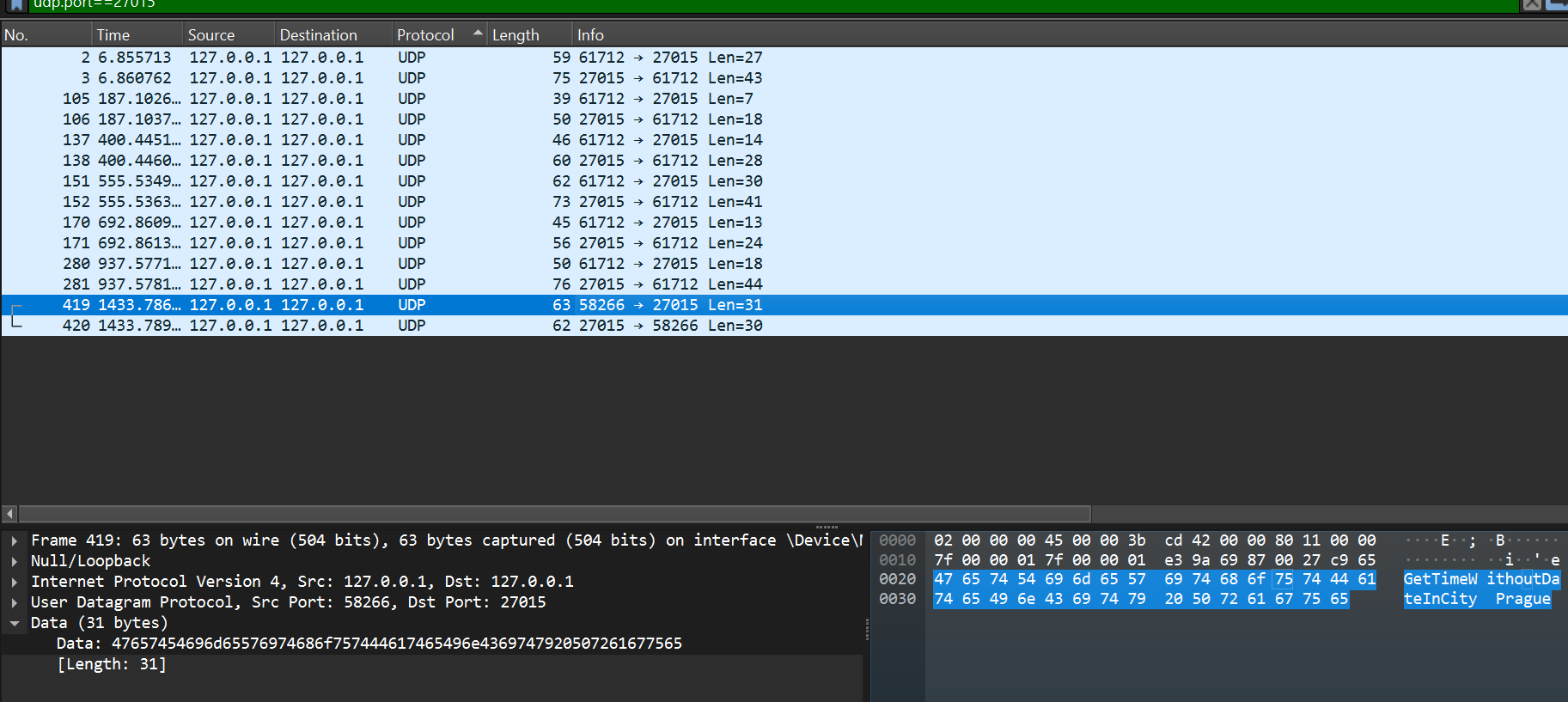
תשובה : "Daylight savings time is currently in effect"



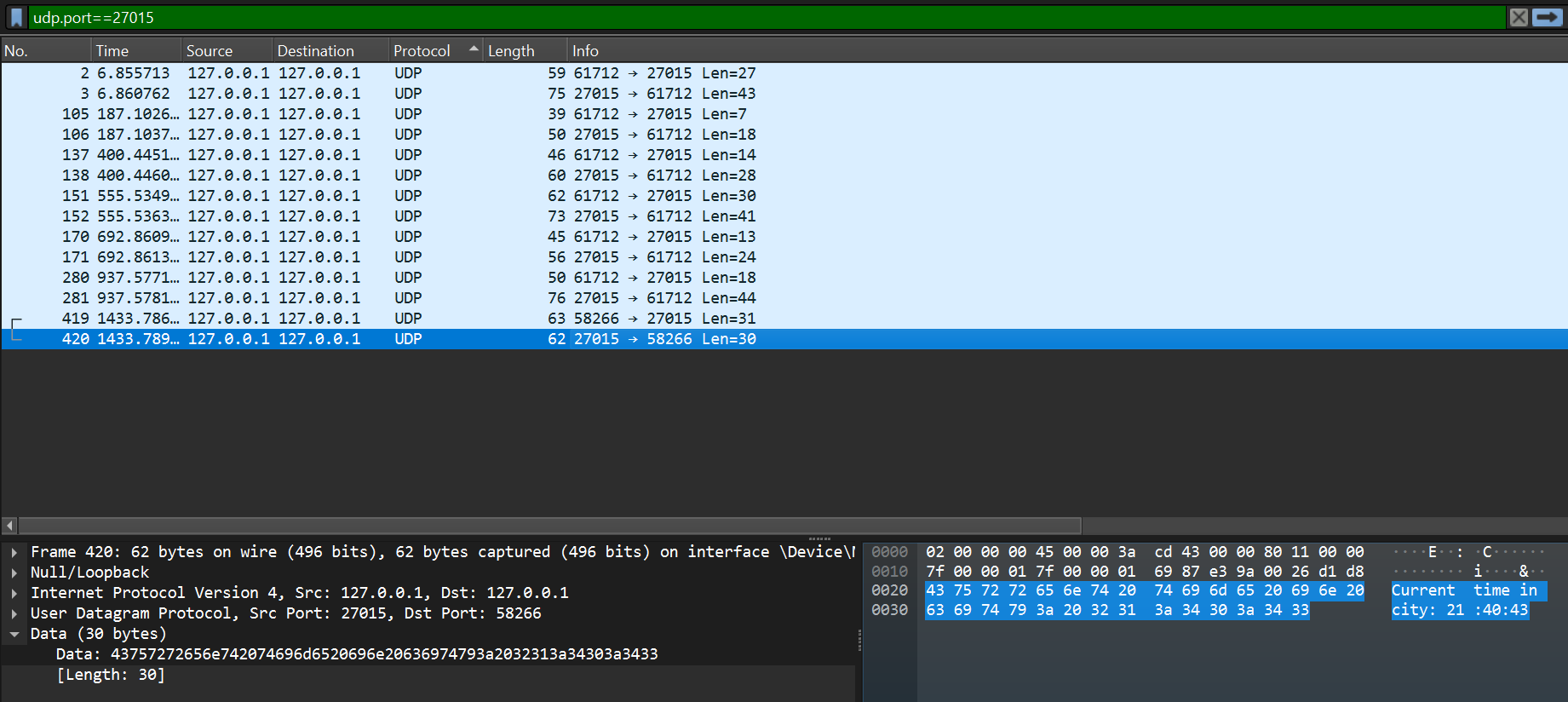
בקשה של שעה ללא תאריך לפי ערים מתוך רשימת אפשרויות:

הבקשה : "GetTimeWithoutDateInCity Prague" (דוגמה אחת מתוך הערים)

|  |  |
| --- | --- |
| Prague(parameters) | GetTimeWithoutDateInCity(opcode) |



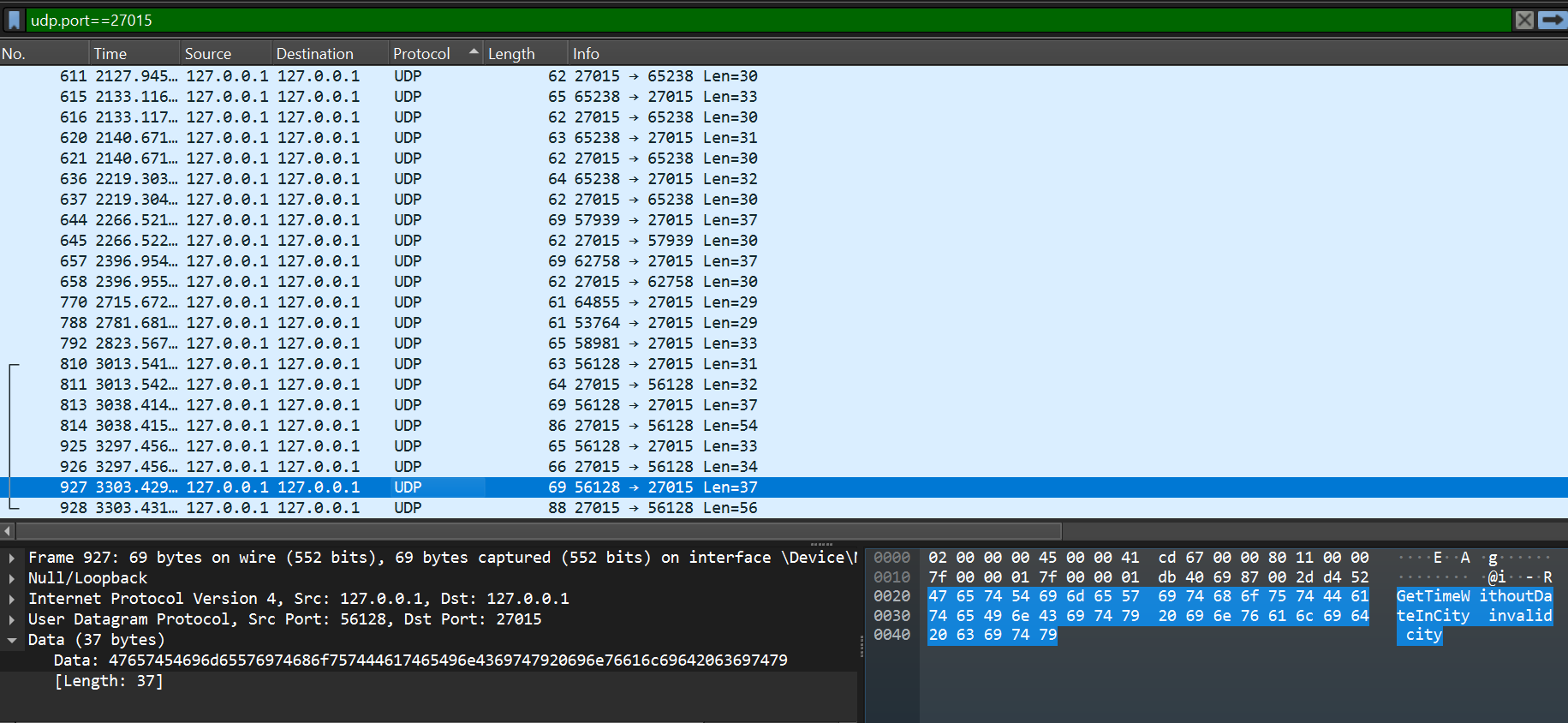
תשובה : Current time in city: 21:40:43



בקשה של שעה ללא תאריך לפי ערים מתוך רשימת אפשרויות:

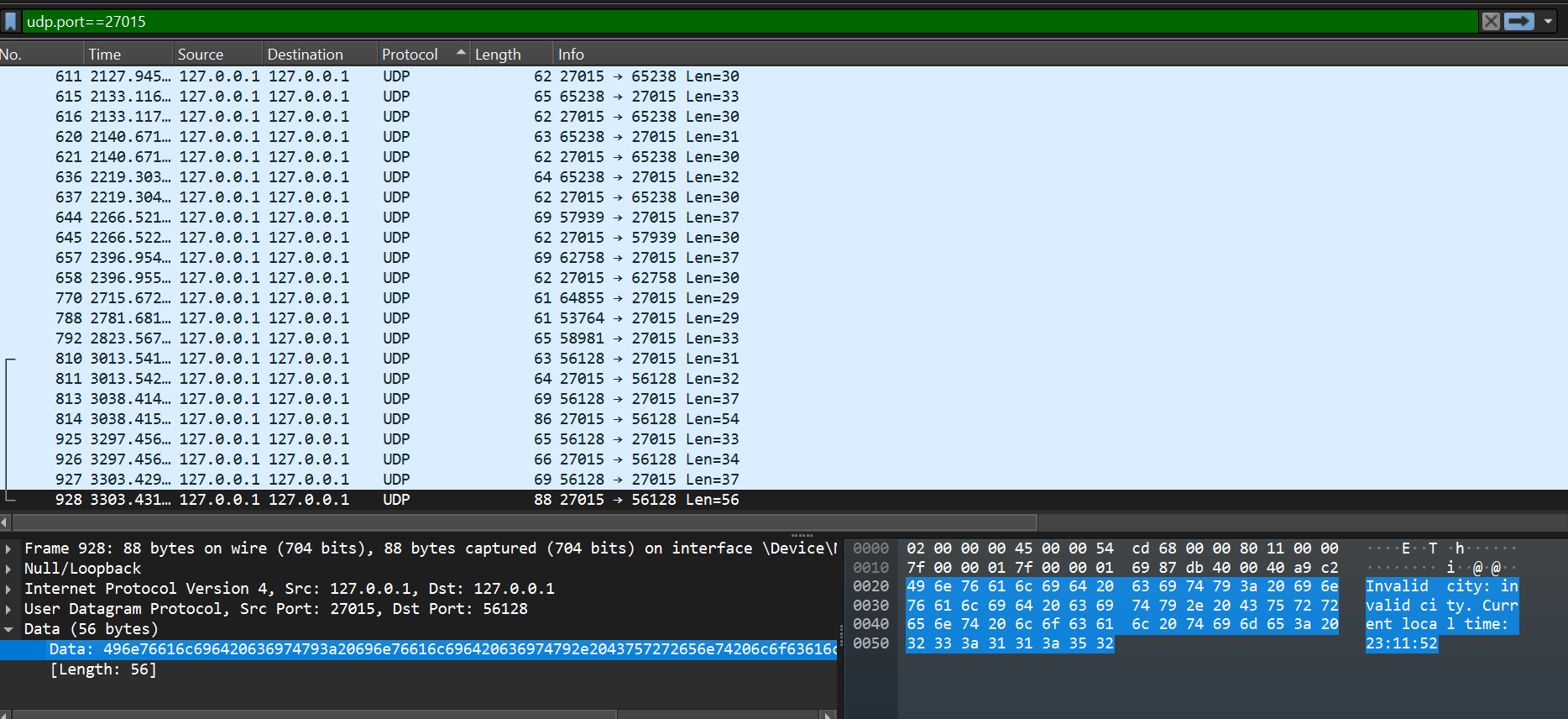
הבקשה : "GetTimeWithoutDateInCity invalid city"

|  |  |
| --- | --- |
| invalid city(parameters) | GetTimeWithoutDateInCity(opcode) |



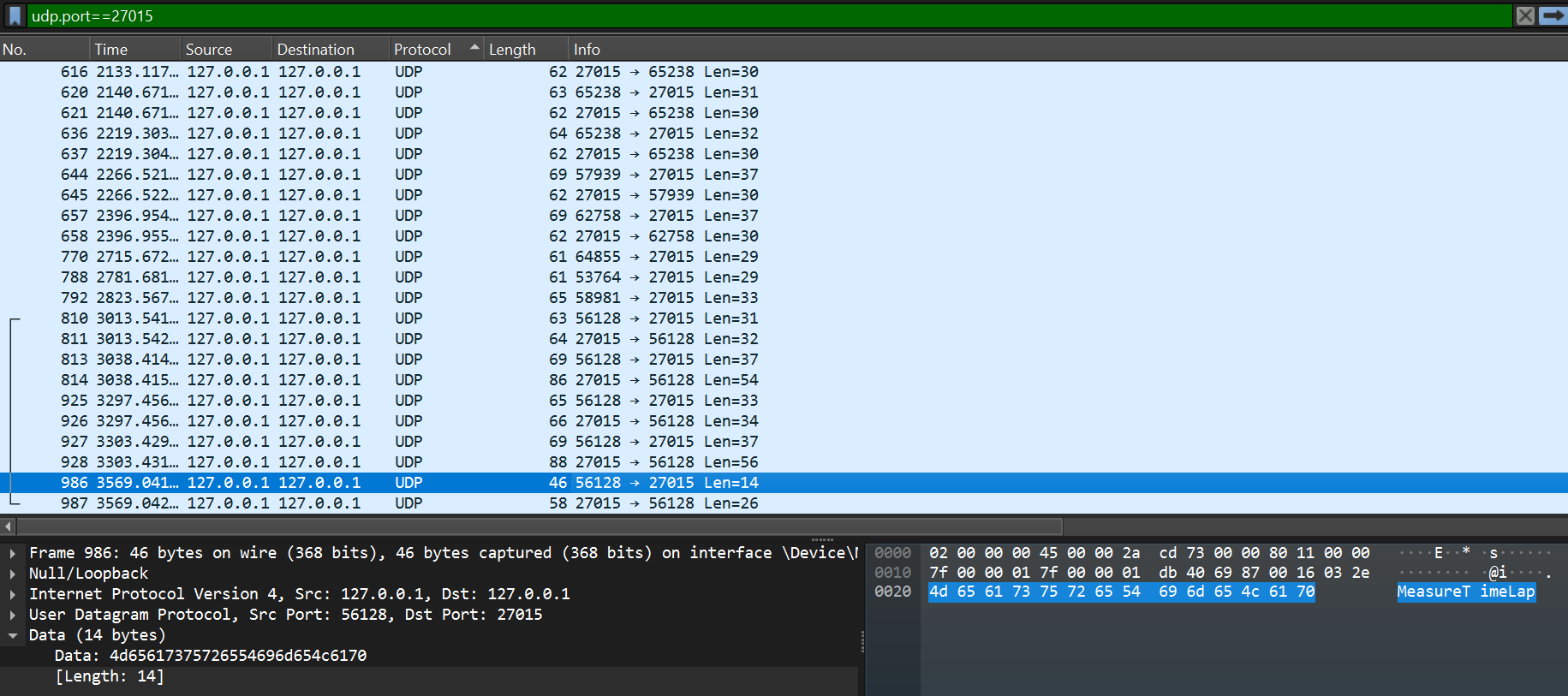
תשובה : "Invalid city: invalid city. Current local time: 23:11:52

מוחזרת הודעה מתאימה עם הזמן האוניברסלי המתואם (UTC).



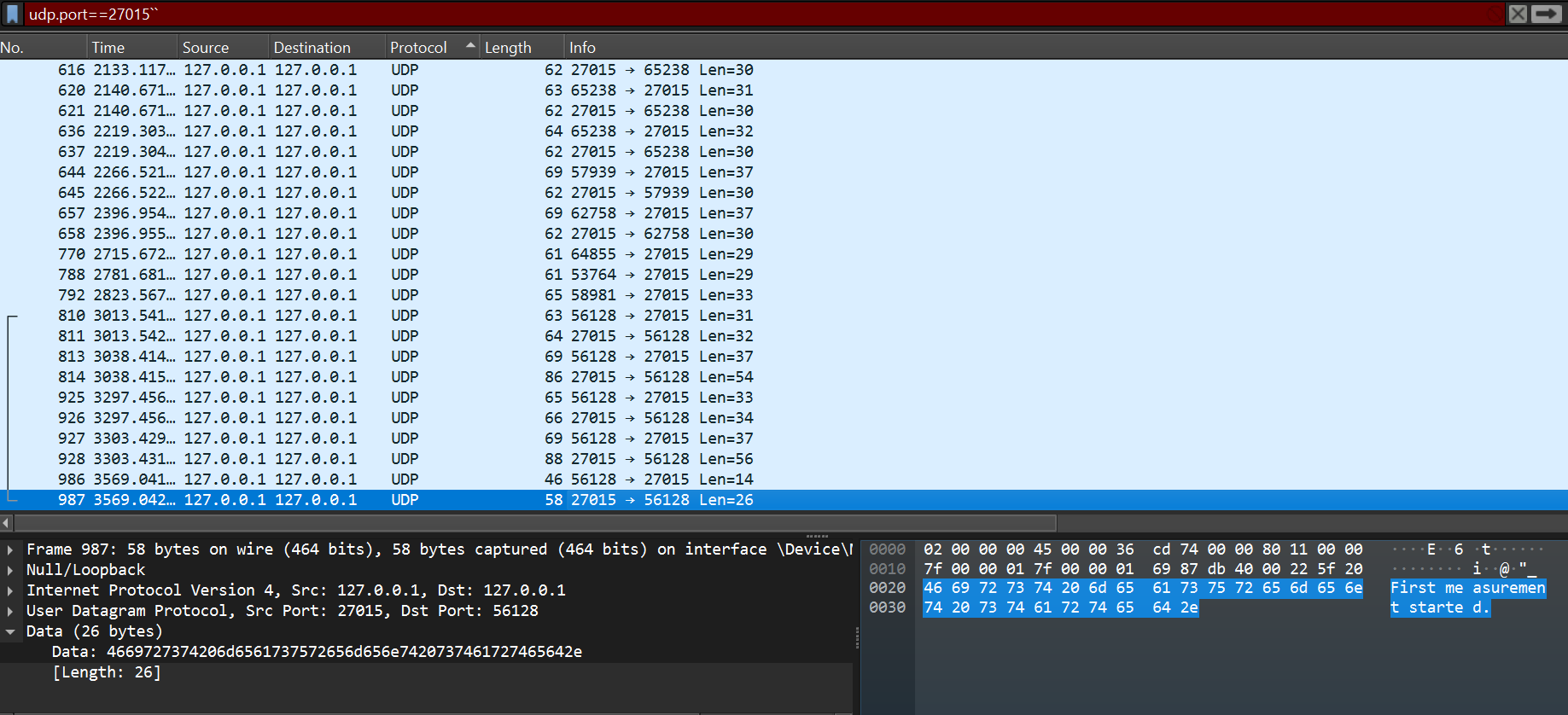
בקשה להתחלה \ סיום של מדידת זמן:

בקשה : MeasureTimeLap”"

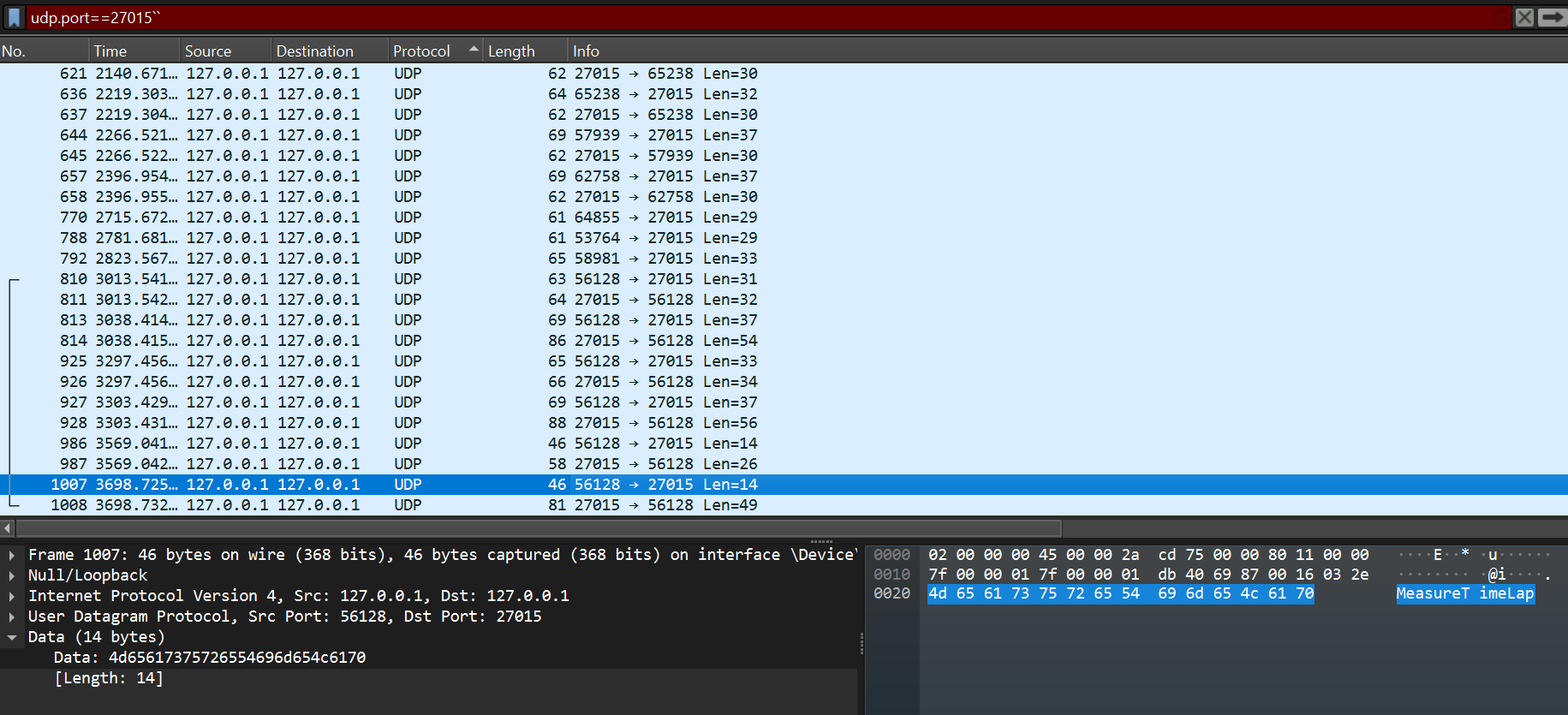


תשובה :

עבור המדידה הראשונה : "First measurement started."



בקשה שנייה : MeasureTimeLap”"



תשובה : "Time lapse since last measurement: 130.00 seconds"

תמונה שמכילה טקסט, צילום מסך, תוכנה, תצוגה

התיאור נוצר באופן אוטומטי