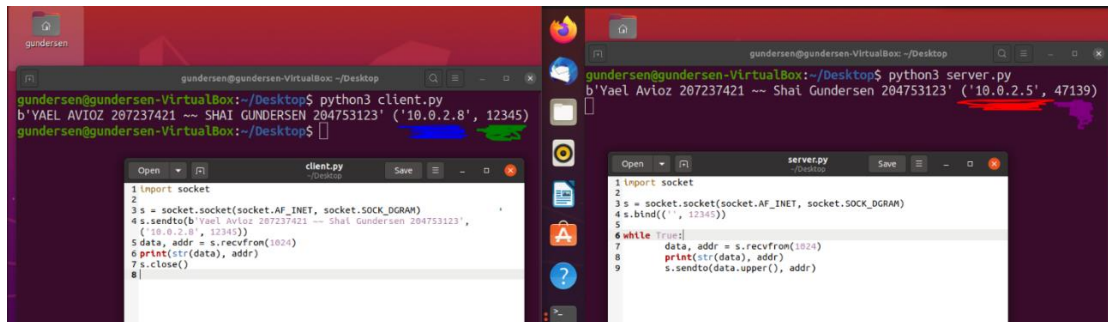


## מבוא לרשתות תקשורת – תרגיל 1

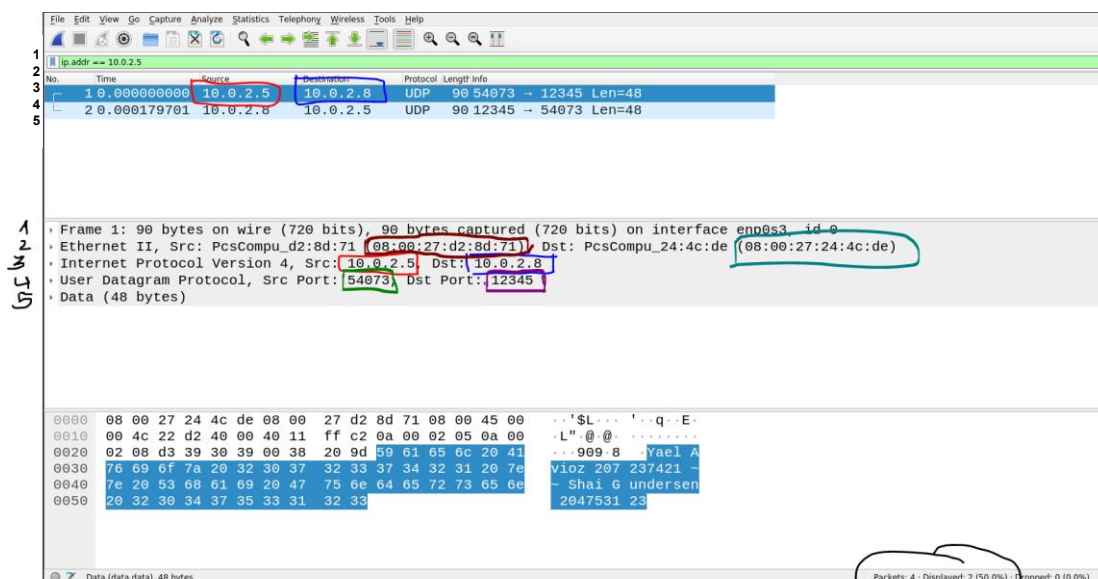
### חלק א

1. בצילום מסך זה ניתן לראות כיצד השרת והלקוח מתקשרים ע"י שליחת הודעה מהלקוח לשרת המכילה את השם והת"ז שלנו. מצד שמאל רואים את המכונה המריצה את הלקוח ומצד ימין את השרת. ניתן לראות שהתוכנה של השרת ממשיכה להמתין כדי לקבל חבילות נוספות.  
צבע כחול – כתובת ה IP של השרת.  
צבע אדום – כתובת ה IP של הלקוח.  
צבע ירוק וסגול הם מספרי הפורט של השרת והלקוח בהתאמה.



2. בצילום זה ניתן לראות את הסנפת המידע באמצעות תוכנת Wireshark – השתמשנו בפרוטוקול UDP, ועל מנת לסנן את החבילות השתמשנו בסנן ip.addr כדי לראות רק את החבילות הרלוונטיות לנו. מקרא-  
עיגול שחור למטה – ניתן לראות שהתבצע סינון בו אנחנו מראים רק 2 חבילות מתוך 4 שהוסנפו.  
שורה 2 – שכבת הערוץ. צבע חום כתובת ה MAC של הלקוח, צבע טורקיז כתובת ה MAC של השרת.  
שורה 3 – שכבת הרשת. צבע אדום כתובת ה IP של הלקוח, צבע כחול כתובת ה IP של השרת.  
שורה 4 – שכבת התעבורה. צבע ירוק מספר הפורט ממנו נשלחה החבילה (של הלקוח בעצם) ובצבע סגול מספר הפורט של השרת בו הוא מחכה לקבל בקשות חדשות.  
שורה 5 – שכבת האפליקציה שם נמצא המידע שאנחנו רוצים להעביר.

3. בקוד השרת והלקוח השתמשנו במספרי פורט על מנת לתקשר בין השרת ללקוח.  
בעצם, תפקידו של הפורט הוא לסמן לשכבת התעבורה לאיזו אפליקציה במחשב המידע מיועד.  
בחלק 1 ניתן לראות את הצילום הרלוונטי וההסבר.



4. בחלון הימני ניתן לראות את המכונה המריצה את הלקוח וניתן לראות בסימון הכחול את כתובת ה IP שלו ובצבע אדום את כתובת ה MAC.
- בחלון השמאלי ניתן לראות את המכונה המריצה את השרת וניתן לראות בסימון הירוק את כתובת ה IP שלו ובצבע סגול את כתובת ה MAC.
- ניתן לראות כיצד באמצעות הפקודה ifconfig בדקנו זאת בטרמינל וקיבלנו כי הן זהות לכתובות המופיעות בתוכנת Wireshark.

```
gundersen@gundersen-VirtualBox: ~/Desktop
gundersen@gundersen-VirtualBox:~/Desktop$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.5 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::f360:ae09:96c2:bb3b prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:d2:8d:71 txqueuelen 1000 (Ethernet)
    RX packets 104487 bytes 149197120 (149.1 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 29471 bytes 2026717 (2.0 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 1811 bytes 256434 (256.4 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1811 bytes 256434 (256.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

gundersen@gundersen-VirtualBox:~/Desktop$

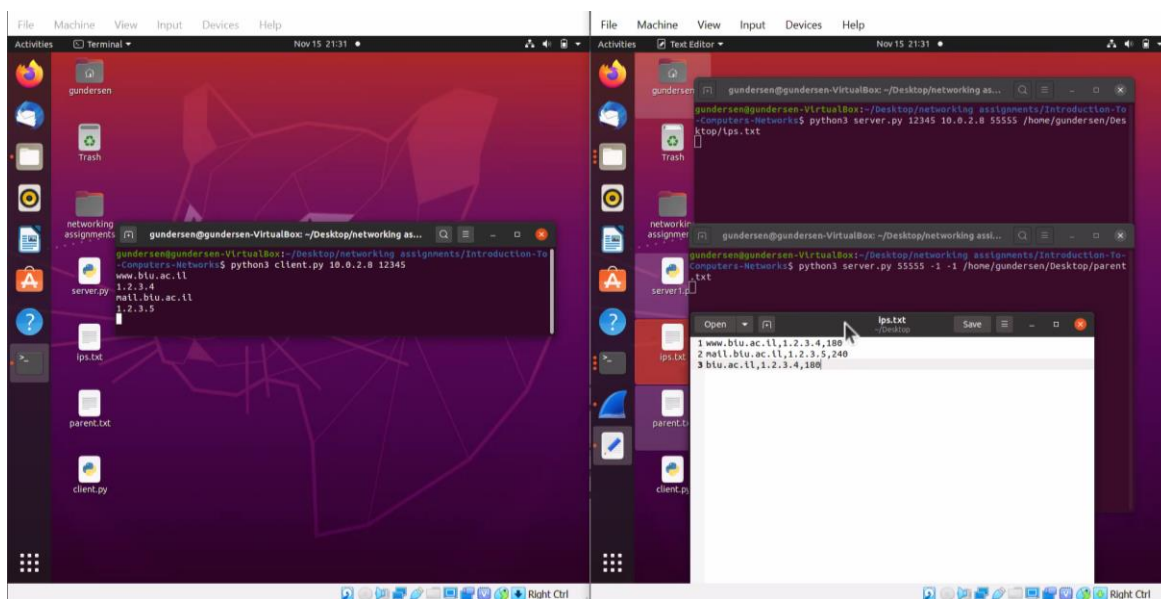
gundersen@gundersen-VirtualBox:~/Desktop$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.8 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::b1f3:25c4:1eec:2b62 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:24:4c:de txqueuelen 1000 (Ethernet)
    RX packets 104504 bytes 149635555 (149.6 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 29035 bytes 2029130 (2.0 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 3058 bytes 348178 (348.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3058 bytes 348178 (348.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

gundersen@gundersen-VirtualBox:~/Desktop$
```

## חלק ב

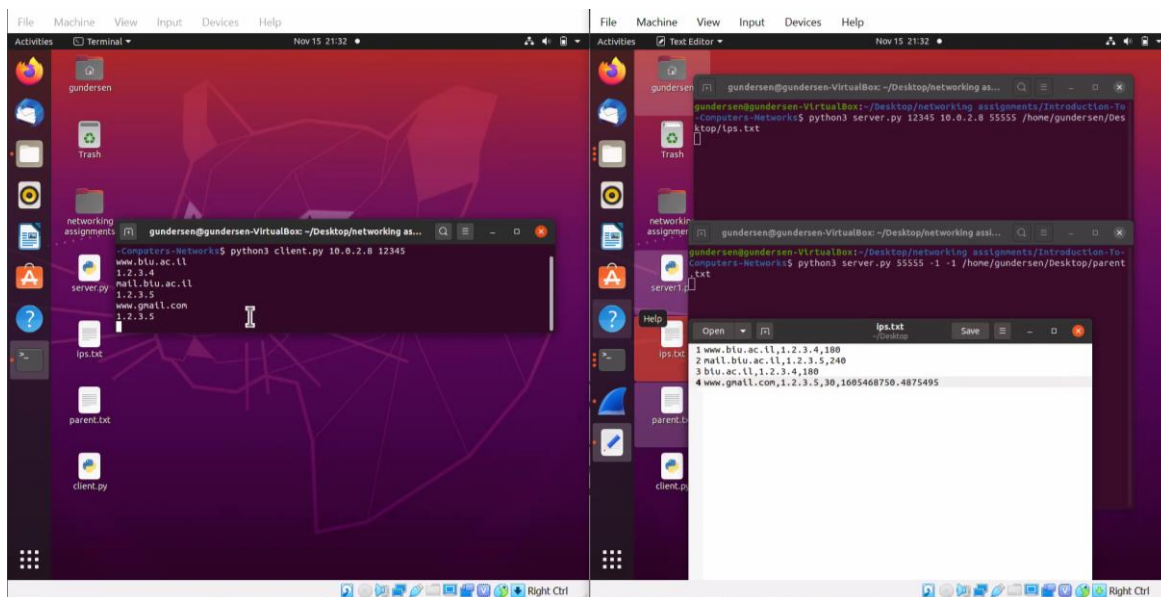
### צילום מסך 1:



בצד ימין של צילום מסך זה ניתן לראות את המידע ששמור על השרת שלנו ובצד שמאל מופיע צד הלקוח. בצד הלקוח בשורה הראשונה הלקוח מבקש מהשרת את כתובת ה IP של הכתובת [www.biu.ac.il](http://www.biu.ac.il) השרת מחזיר לו את הכתובת ה IP אשר שמורה בזיכרון שלו ומשויכת לכתובת זו. ניתן לראות כי הפלט שקיבלנו בצד הלקוח (1.2.3.4) אכן תואם את כתובת ה IP שמופיעה בעבור הכתובת המבוקשת בשרת.

הלקוח פונה אל השרת פעם נוספת ומבקש את כתובת ה IP של הכתובת mail.biu.ac.il, גם כתובת זו כמו הכתובת הקודמת כבר שמורה בזיכרון השרת ולכן הוא מחזיר את כתובת ה IP המשוויכת לכתובת זו והפלט המוצג הוא 1.2.3.5 בדיוק כפי שמופיע במידע ששמור בשרת (ניתן לראות בשורה 2 בצד השרת).

### צילום מסך 2:

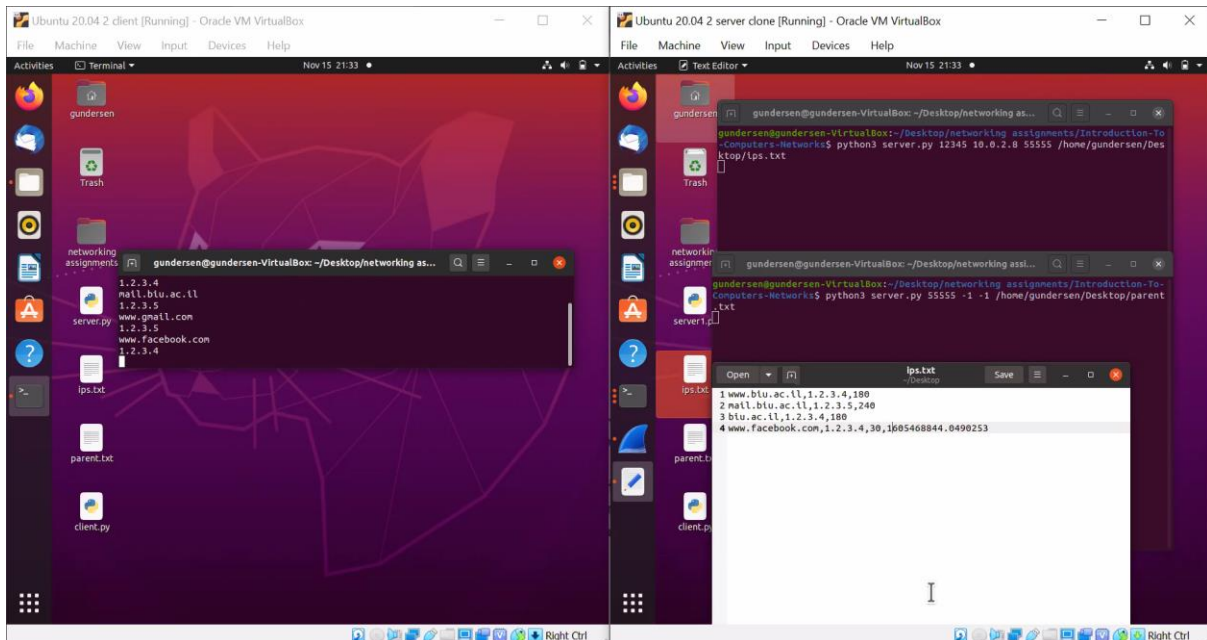


כעת הלקוח מבקש מהשרת את הכתובת [www.gmail.com](http://www.gmail.com). בצילום המסך הקודם ניתן לראות שכתובת זו אינה שמורה בזיכרון השרת ולכן השרת פונה לשרת האב ומבקש ממנו את הכתובת. שרת האב מחזיר לשרת את הכתובת כאשר הTTL שווה 30 שניות, כלומר כתובת זו תופיע בזיכרון השרת רק ל-30 שניות הבאות ולאחר מכן תימחק מזיכרון השרת ובכל פניה נוספת אשר תבקש את כתובת ה IP של כתובת זו ותחרוג מזמן זה נצטרך לפנות שוב לשרת האב על מנת לקבל את המידע הדרוש.

\*המספר שמופיע ליד TTL זהו ייצוג של השעה הנוכחית בשניות בתוספת הTTL שהתקבל על ידי שרת האב, כלומר

זוהי השעה בה המידע שקיבלנו משרת האב אמור להימחק מזיכרון השרת שלנו ואת השעה הזו ייצגו באמצעות שניות. בצד הלקוח ניתן לראות כי השרת החזיר ללקוח את כתובת ה IP המבוקשת (אותה כתובת שקיבל משרת האב).

### צילום מסך 3:

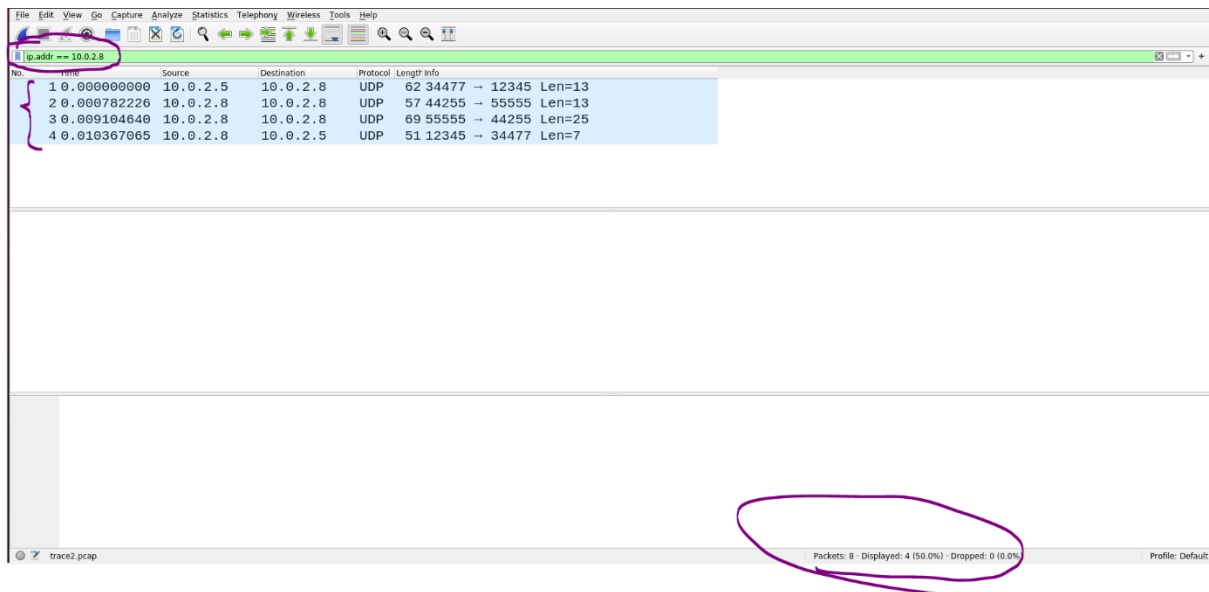


צילום מסך זה בוצע לאחר 30 שניות מרגע קבלת המידע עבור כתובת [www.gmail.com](http://www.gmail.com) ואכן ניתן לראות כי המידע שהתקבל בשרת משרת האב נמחק לאחר שהזמן שהוגדר לו חלף ( $TTL=30$ ). כעת הלקוח ביקש את כתובת ה IP של הכתובת [www.facebook.com](http://www.facebook.com) ושוב כמו קודם הכתובת לא הופיעה בזיכרון השרת ולכן הוא פנה לשרת האב בבקשה לקבל מידע עבור כתובת זו, המידע שהתקבל מכיל  $TTL=30$  כלומר גם פה לאחר 30 שניות המידע ימחק מהשרת שלנו. השרת החזיר ללקוח את כתובת ה IP המתאימה עבור הבקשה של הלקוח.

**\*כעת נראה כיצד נראת תעבורת הרשת כאשר הלקוח ביקש כתובת מסוימת אשר השרת לא מכיר. (לא נראה תעבורה של כתובת שהשרת כן מכיר כי זה מקרה פרטי של כתובת שהוא לא מכיר)**

### **תחילה נראה צילום מסך כללי ולאחריו נסביר כל שורה בה:**

בצילום זה ניתן לראות כי סיננו את החבילות שהוסנפו לפי כתובת ה־ip של השרת (אנחנו יודעים שזהו השרת לפי הצילום האחרון בחלק א') ולכן מוצגות לנו רק 4 מתוך 8 חבילות (סימון בצד ימין למטה). חשוב לציין כי ההסנפה התבצעה על any ב־wireshark וזאת מכיוון שאם היינו עושים על loopback היינו רואים רק תעבורה בין השרת לשרת האב (כי הם חולקים את אותה כתובת IP) ואם היינו מסניפים לפי enp0s3 היינו מקבלים רק את התעבורה בין השרת ללקוח (כי הם ברשתות נפרדות)



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.5	10.0.2.8	UDP	62	34477 → 12345 Len=13
2	0.000782226	10.0.2.8	10.0.2.8	UDP	57	44255 → 55555 Len=13
3	0.009104640	10.0.2.8	10.0.2.8	UDP	69	55555 → 44255 Len=25
4	0.010367065	10.0.2.8	10.0.2.5	UDP	51	12345 → 34477 Len=7

Packets: 8 - Displayed: 4 (50.0%) - Dropped: 0 (0.0%)

## כעת נצלול לעומק כל שורה:

### שורה 1: לקוח - < שרת

Frame 1: 62 bytes on wire (496 bits), 62 bytes captured (496 bits)

Linux cooked capture

Internet Protocol Version 4, Src: 10.0.2.5, Dst: 10.0.2.8

User Datagram Protocol, Src Port: 34477, Dst Port: 12345

Data (13 bytes)

Data: 7777772e676d61696c2e636f6d

[Length: 13]

VSS Monitoring Ethernet trailer, Source Port: 0

0000 00 00 00 01 00 06 08 00 27 d2 8d 71 00 00 08 00 .....q....  
 0010 45 00 00 29 71 d4 40 00 40 11 b0 e3 0a 00 02 05 E..).q.@. @..  
 0020 0a 00 02 08 86 ad 30 39 00 15 3c b6 77 77 77 2e ..www..  
 0030 67 6d 61 69 6c 2e 63 6f 6d 00 00 00 00 00 00 gmail.co m

חבילה המגיעה מכתובת ip=10.0.2.5 שזוהי הכתובת של הלקוח בשכבת הרשת מהפורט 34477 בשכבת התעבורה. נשלחת אל הכתובת ip=10.0.2.8 שזוהי הכתובת של השרת בשכבת הרשת אל הפורט 12345 בשכבת התעבורה. נראה כי בשכבת האפליקציה, המידע שהלקוח שלח נמצא בשדה DATA והוא מכיל את www.gmail.com, כפי שניתן לראות בסימון השחור מצד ימין למטה. כעת, השרת מעבד את המידע ומבין כי אין ברשותו את הכתובת של הלקוח מבקש ולכן יעביר את הבקשה לשרת האב שלו.

### שורה 2: שרת - < שרת אב

Frame 2: 57 bytes on wire (456 bits), 57 bytes captured (456 bits)

Linux cooked capture

Internet Protocol Version 4, Src: 10.0.2.8, Dst: 10.0.2.8

User Datagram Protocol, Src Port: 44255, Dst Port: 55555

Data (13 bytes)

Data: 7777772e676d61696c2e636f6d

[Length: 13]

VSS Monitoring Ethernet trailer, Source Port: 0

0000 00 00 03 04 00 06 00 00 00 00 00 00 00 08 00 .....  
 0010 45 00 00 29 92 ad 40 00 40 11 90 07 0a 00 02 08 E..).q.@. @..  
 0020 0a 00 02 08 ac df d9 03 00 15 18 36 77 77 77 2e ..www..  
 0030 67 6d 61 69 6c 2e 63 6f 6d 00 00 00 00 00 00 gmail.co m

בחלק זה, מכיוון והשרת לא מצא את הכתובת אצלו הוא נאלץ לשאול את אביו לגבי הכתובת. למעשה, השרת הפך להיות "לקוח" של שרת האב. נשים לב כי כתובת ה IP של שניהם זהה וזאת אינה הפתעה היות והן שתי פרוצדורות הקורות מאותו המחשב. דבר נוסף שחשוב לשים אליו לב הוא לכך שמספר הפורט של השרת הוא 44255 בשונה ממקודם וזאת מכיוון שבפורט הזה נעשה שימוש על מנת לשלוח חבילות אל פורט שהוא לא מי שביקש מאיתנו את הבקשה האחרונה (הלקוח) ובפורט הקודם נעשה שימוש כדי להאזין לחבילות שמגיעות.



אפשר לראות בצד ימין למטה כי הDATA שהשרת מעביר הלאה אל שרת האב הוא בדיוק כתובת האינטרנט שהלקוח ביקש ולא הצליח למצוא לבד.

### שורה 3: שרת אב - < שרת

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.2.5	10.0.2.8	UDP	62	34477 → 12345 Len=13
2	0.000782	10.0.2.8	10.0.2.8	UDP	57	44255 → 55555 Len=13
3	0.009105	10.0.2.8	10.0.2.8	UDP	69	55555 → 44255 Len=25
4	0.010367	10.0.2.8	10.0.2.5	UDP	51	12345 → 34477 Len=7

Frame 3: 69 bytes on wire (552 bits), 69 bytes captured (552 bits)  
 Linux cooked capture  
 Internet Protocol Version 4, Src: 10.0.2.8, Dst: 10.0.2.8  
 User Datagram Protocol, Src Port: 55555, Dst Port: 44255  
 Data (25 bytes)

```

0000  00 00 03 04 00 06 00 00 00 00 00 00 00 00 08 00  .....
0010  45 00 00 35 92 af 40 00 40 11 8f f9 0a 00 02 08  E..5..@. @.
0020  0a 00 02 08 d9 03 ac df 00 21 18 42 77 77 77 2e  .....Bwww.
0030  67 6d 61 69 6c 2e 63 6f 6d 2c 31 2e 32 2e 33 2e  gmail.co m,1.2.3.
0040  35 2c 33 30 0a                                     5,30.
  
```

בחלק זה, שרת האב מצא את הכתובת המדוברת והוא מחזיר תשובה ל"לקוח" (בעצם לשרת) ששלח את הבקשה. נשים לב כי שוב כתובת הIP של השניים זהה, וגם כתובות הפורטים זהים, שכן שרת האב רוצה לשלוח את החבילה בדיוק חזרה לפורט שביקש ממנו את החבילה. ניתן לראות כי הDATA ששרת האב שולח הוא www.gmail.com,1.2.3.5,30. דבר המעיד כי הוא מצא את הכתובת המבוקשת.

### שורה 4: שרת - < לקוח

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.2.5	10.0.2.8	UDP	62	34477 → 12345 Len=13
2	0.000782	10.0.2.8	10.0.2.8	UDP	57	44255 → 55555 Len=13
3	0.009105	10.0.2.8	10.0.2.8	UDP	69	55555 → 44255 Len=25
4	0.010367	10.0.2.8	10.0.2.5	UDP	51	12345 → 34477 Len=7

Frame 4: 51 bytes on wire (408 bits), 51 bytes captured (408 bits)  
 Linux cooked capture  
 Internet Protocol Version 4, Src: 10.0.2.8, Dst: 10.0.2.5  
 User Datagram Protocol, Src Port: 12345, Dst Port: 34477  
 Data (7 bytes)  
 Data: 312e322e332e35  
 [Length: 7]

```

0000  00 04 00 01 00 06 08 00 27 24 4c de 00 00 08 00  ..... '$L...
0010  45 00 00 23 f2 98 40 00 40 11 30 25 0a 00 02 08  E..#..@. @.0%
0020  0a 00 02 05 30 39 86 ad 00 0f 18 2d 31 2e 32 2e  .....09...1.2.
0030  33 2e 35                                           3.5
  
```

בחלק זה השרת מחזיר תשובה ללקוח עבור אותה הבקשה שנשלחה בשורה 1. נשים לב כי כעת כתובת הIP של השולח היא 10.0.2.8 שזוהי בדיוק כתובת השרת ואילו כתובת הIP של המקבל היא 10.0.2.5 שזוהי בדיוק כתובת הIP של הלקוח (כמו שורה 1 רק עם החלפת תפקידים) נשים לב בנוסף כי השרת זוכר את הפורט ממנו התקבלה ההודעה 34477 שכן רק כך הלקוח ידע בשכבת התעבורה לאיזו אפליקציה להעביר את הDATA. נראה כי הDATA הנשלח הוא רק כתובת IP של אתר האינטרנט המבוקש 1.2.3.5, כלומר השרת קיבל משרת האב את השורה המלאה www.gmail.com,1.2.3.5,30 ובשכבת האפליקציה, ביצע תהליך של שמירת המידע לפי TTL ושלח ללקוח רק את המידע שרלוונטי אליו.

