# רשתות תקשורת מחשבים - תרגיל בית תיאורטי 3

1

1

## שאלה 1

1

### i

1

1.5

1

1

1

(כל צומת ירוק בטבלה הוא השולח, והטור תחתיו הוא המרחק שהוא שולח בינו לבין הצומת TO)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TO | A | B | C | D | E | F | G | H |
| A | 0 | 1 | ∞ | ∞ | ∞ | ∞ | ∞ | 1 |
| B | 1 | 0 | 1 | ∞ | ∞ | ∞ | ∞ | ∞ |
| C | ∞ | 1 | 0 | 1 | ∞ | ∞ | ∞ | ∞ |
| D | ∞ | ∞ | 1 | 0 | 1.5 | ∞ | ∞ | ∞ |
| E | ∞ | ∞ | ∞ | 1.5 | 0 | 1 | ∞ | ∞ |
| F | ∞ | ∞ | ∞ | ∞ | 1 | 0 | 1 | ∞ |
| G | ∞ | ∞ | ∞ | ∞ | ∞ | 1 | 0 | 1 |
| H | 1 | ∞ | ∞ | ∞ | ∞ | ∞ | 1 | 0 |

### ii

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TO | A | B | C | D | E | F | G | H |
| A | 0 | 1 | 2 | 3 | 4 | 3 | 2 | 1 |
| B | 1 | 0 | 1 | 2 | 3.5 | 4 | 3 | 2 |
| C | 2 | 1 | 0 | 1 | 2.5 | 3.5 | 4 | 3 |
| D | 3 | 2 | 1 | 0 | 1.5 | 2.5 | 3.5 | 4 |
| E | 4 | 3.5 | 2.5 | 1.5 | 0 | 1 | 2 | 3 |
| F | 3 | 4 | 3.5 | 2.5 | 1 | 0 | 1 | 2 |
| G | 2 | 3 | 4 | 3.5 | 2 | 1 | 0 | 1 |
| H | 1 | 2 | 3 | 4 | 3 | 2 | 1 | 0 |

נחלק את הבעיה ל-2:   
1. צמתים שהמידע הטוב שלהם תלוי בהעברת מידע על קשת E-D  
מיכוון שצמתים אלו זקוקים למידע שנמצא שלכל היותר במרחק 2 קשתות מהם (בפועל המידע הוא על יעדים שנמצאים במרחק 3 קשתות מהם, אבל ב-t=0 המידע עליהם כבר שמור אצל שכניהם, לכן המידע המבוקש אודותיהם נמצא בצומת במרחק 2), לצמתים במרחק 4 קשתות הנתיב שלא עובר דרך E-D יהיה טוב יותר (מרחק 4 לא דרך DE, לעומת מרחק 4.5 דרך DE), 🡸 הזמן שיקח הוא קשת אחת של 1 מילישניה, וקשת אחת של 1.5 מילי שניות. 🡸 בזמן t=2.5 הצמתים יקבלו את כל המידע.

2. צמתים שלא צריכים לקבל מידע שעובר על קשת D-E  
צמתים אלו צריכים בזמן t=0 מידע שנמצא לכל היותר מרחק 3 קשתות מהם, ולכן יקח לו להגיע 3 מילי שניות.

🡸 סהכ קיבלנו בזמן t=3 כל הצמתים יקבלו את המידע הסופי שלהם לצורך עדכון הטבלאות.

## שאלה 2

### ניתן לייצר 4 עצי יעד: (העצים בכתום) 1.

8

6

7

2

4

1

3

5

2.

8

6

7

2

4

1

3

5

3.

8

7

2

1

3

5

6

4

4.

8

6

7

2

4

1

3

5

נשלחות **5 פקטות ללא תלות בעץ**. מספר הפקטות שנשלחות הוא כמספר הקשתות בעץ, מיכוון שעץ בהגדרתו מכיל |V|-1 קשתות (כאשר |V| הוא מספר הצמתים בגרף), כל עץ שנייצר יכיל אותו מספר קשתות.

### C.

V1 – שולח 2 פקטות.  
V2 – שולח 2 פקטות.  
V3- שולח 3 פקטות.  
V4 – שולח 3 פקטות.  
V5 – שולח פקטה אחת.  
V6 – לא שולח פקטות.  
סה"כ קיבלנו **11 פקטות.**

אין זה משנה באיזה עץ נבחר מכיוון שלכל אחד מהצמתים יהיה בדיוק קשת אחת (הקשת הכתומה שמובילה ל-V1) שממנה הוא מעביר את המידע הלאה לכל שאר הקשתות(נקרא לה "קשת העברה"), ומכל שאר הקשתות מידע שמגיע לא יעבור הלאה. בנוסף, בהכרח הוא יקבל את המידע בקשת ההעברה (מיכוון שהעץ הכתום עובר דרך כל הצמתים) 🡸   
כל צומת תקבל את המידע מ"קשת העברה"(שהיא יחידה), ולכן תשלח אותה בכל שאר הקשתות בדיוק פעם אחת.  
מכך נובע שמספר הפקטות יהיה זהה לא משנה באיזה מבנה של עץ נבחר.

## שאלה 3

|  |  |
| --- | --- |
| ארגון | הקצאת כתובות לארגון |
| A | 193.15.0.0/19 (8192-2 address) |
| B | 193.15.32.0/20 (4096-2 address) |
| C | 193.15.48.0/21 (2048-2 address) |
| D | 193.15.56.0/21 (2048-2 address)  193.15.64.0/21 (2048-2 address) |

## שאלה 4

נניח שהsubnet mask בכל רשתות תחת הוא 255.255.255.0. פרט לרשת המחשבים תחת R2 הוא 255.255.254.0.

בנוסף, נניח שמשקל כל הקשתות זהה(לא נתון משקל), ולכן ננתב בדרך הקצרה ביותר בגרף.(בעקבות כך גם אין התייחסות למשקלים בטבלאות מטה)

### R3

|  |  |  |
| --- | --- | --- |
| Next Hop | Interface(=port) | Mask |
|  | 130.132.2.3 | 130.132.2.0/24 |
| 130.132.5.2 | 130.132.5.1 | 130.132.9.0/23 |
| 130.132.4.3 | 130.132.4.2 | 130.132.1.0/24 |

### R2

|  |  |  |
| --- | --- | --- |
| Next Hop | Interface(=port) | Mask |
| 130.132.5.1 | 130.132.5.2 | 130.132.2.0/24 |
|  | 130.132.9.7 | 130.132.9.0/23 |
| 130.132.6.1 | 130.132.6.2 | 130.132.1.0/24 |

### R1

|  |  |  |
| --- | --- | --- |
| Next Hop | Interface(=port) | Mask |
| 130.132.4.2 | 130.132.4.3 | 130.132.2.0/24 |
| 130.132.6.2 | 130.132.6.1 | 130.132.9.0/23 |
|  | 130.132.1.5 | 130.132.1.0/24 |

שליחת הודעה מ-A ל-B (בהתבסס של שכבות Ethernet ו-IP – רק את פרטי היעד של הפקטה)

מחשב A מעוניין לשלוח פקטה למחשב B בכתובת 130.132.9.3. הוא הולך לטבלת הניתוב שלו שמכוונת אותו לdefault gateway שלו - לראטור R1. (הרחבה: בטבלת הניתוב בא לידי ביטוי הsubnet השונה של מחשב B לעומת מחשב A – כתובות בתוך הsubnet מוגדרות להיות On-link(טרמינולוגיה של windows) במקרה שלנו כתובות ברשת הם: 130.132.1.0 עם mask של 255.255.255.0. וכל שאר הכתובות מופנות לdefault gateway.)  
לכן, מחשב A מייצר את הפקטה כאשר כתובת destination mac היא כתובת הmac של R1, וכתובת destination ip היא הכתובת של B (כתובת ip זו תשאר לאורך כל השליחה, לכן לא נזכיר אותו מחדש בבנית הפקטה).

כעת הפקטה הגיעה לנתב R1, אם נסתכל בטבלת הניתוב שלו הממשק הוא 130.132.6.1 וnext hop הוא R2.  
לכן הפקטה תשלח שוב רק כעת כתובת ה- destination mac תהיה של R2

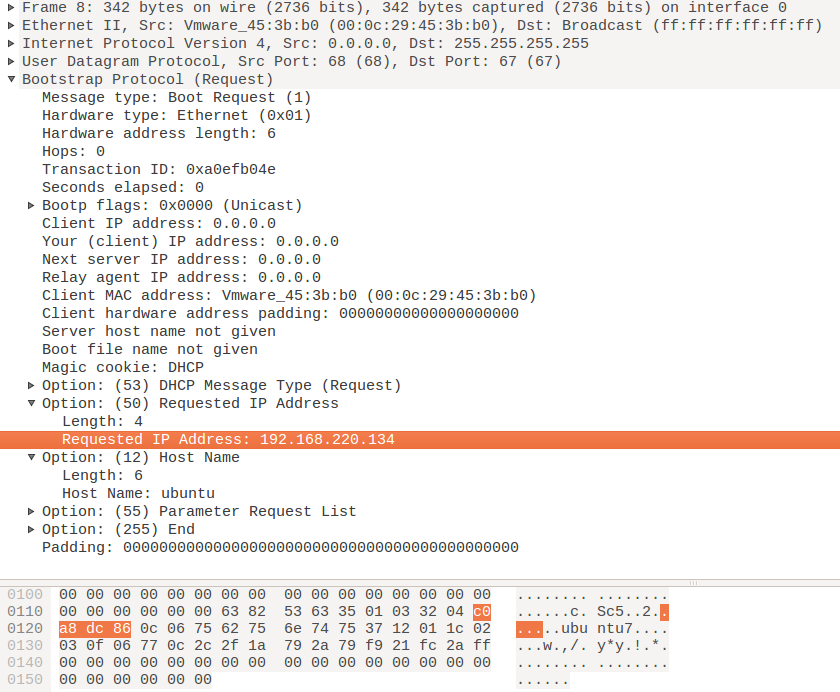
הפקטע כעת בנתב R2, בטבלה שלו מוגדר לו שהפקטה תעבור לממשק 130.132.9.7 ובנוסף אין לו next hop, כלומר הגענו לרשת היעד. כעת הנתב ישנה את destination mac להיות כתובת הmac של B.

הפקטה הגיע לB בהצלחה.

הערות:  
1) בכל פעם שמחשב/מעוניין למצוא את כתובת הmac של היעד שלו הוא מסתכל בטבלת ה-arp, במידה והיא לא קיימת שם הוא שולח פקטת arp.   
2) באופן כללי במידה והתקשורת לא מבוצעת מעל הפרוטוקולים ip ו- Ethernet, ניתן לשנות בתשובתנו את mac ל- כתובות פיסית, וip לכתובת רשת.

## שאלה 5

בחרנו פקטת DHCP Request – הפקטה ששולח הלקוח כדי לבקש כתובת IP מהשרת.



1. השדה Transaction ID שנועד לשייך את הבקשה עם התשובה בין השרת ללקוח, אמור להיות ייחודי והוא אכן נראה אקראי. באמת בפקטת ACK שהשרת שולח השדה הזה זהה.

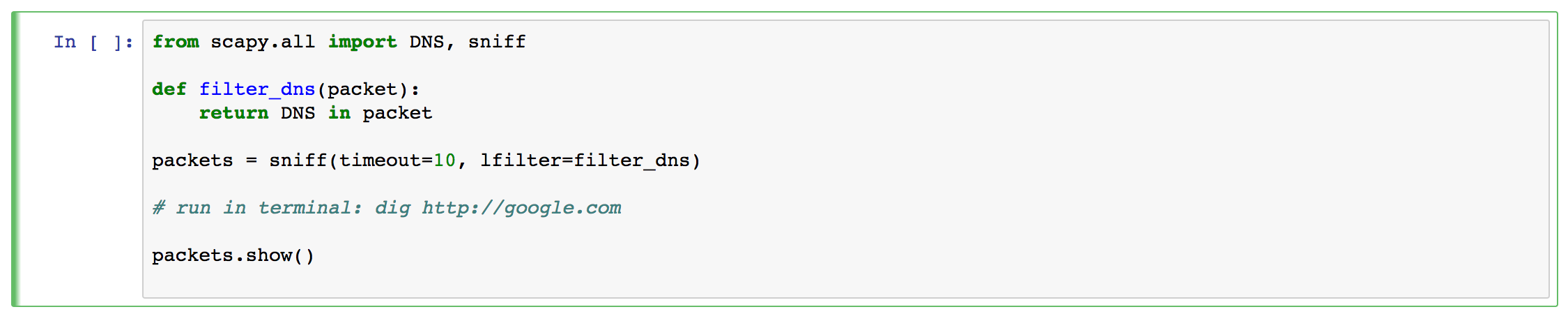
השדה Requested IP Address אמור להכיל את ה-IP שהלקוח מבקש מהשרת. זאת כתובת שהשרת הציע (בהודעת DHCP Offer) והיא באמת שייכת לרשת שאליה התחברנו.

1. בעזרת Wireshark אפשר לראות את השכבות הנמוכות יותר (אפשר לעשות את זה ידנית גם ע"י מעבר ידני של המידע). זה תואם את מה שלמדנו.

DHCP הוא פרוטוקול בשכבת האפליקציה. UDP הוא פרוטוקול בשכבת ה- Transport. IP הוא בשכבת ה- Network ו- Ethernet הוא בשכבת ה- Link.

למרות שעדיין ללקוח אין כתובת IP אפשר לראות שאנחנו עדיין מסתמכים על הפרוטוקול כאשר אנחנו משתמשים בכתובות Broadcast כדי להגיע לשרת DHCP.

## **בונוס**

**We used the following code to sniff DNS packets:**

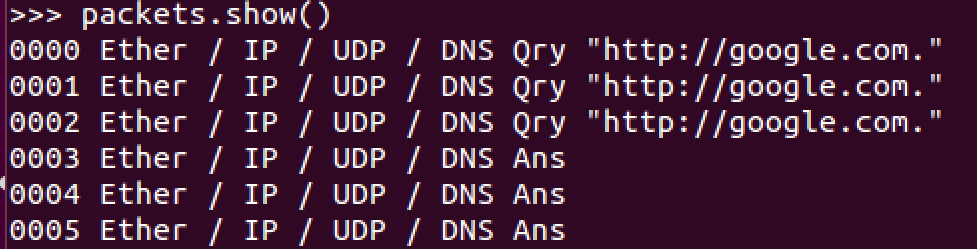
The code defines a packets filter, to filter in only DNS packets.

Then, we create a new sniff which will catch packets that are going in or out of our network card. Using the filter, the sniff will save only the DNS packets it sees.



Now we run:

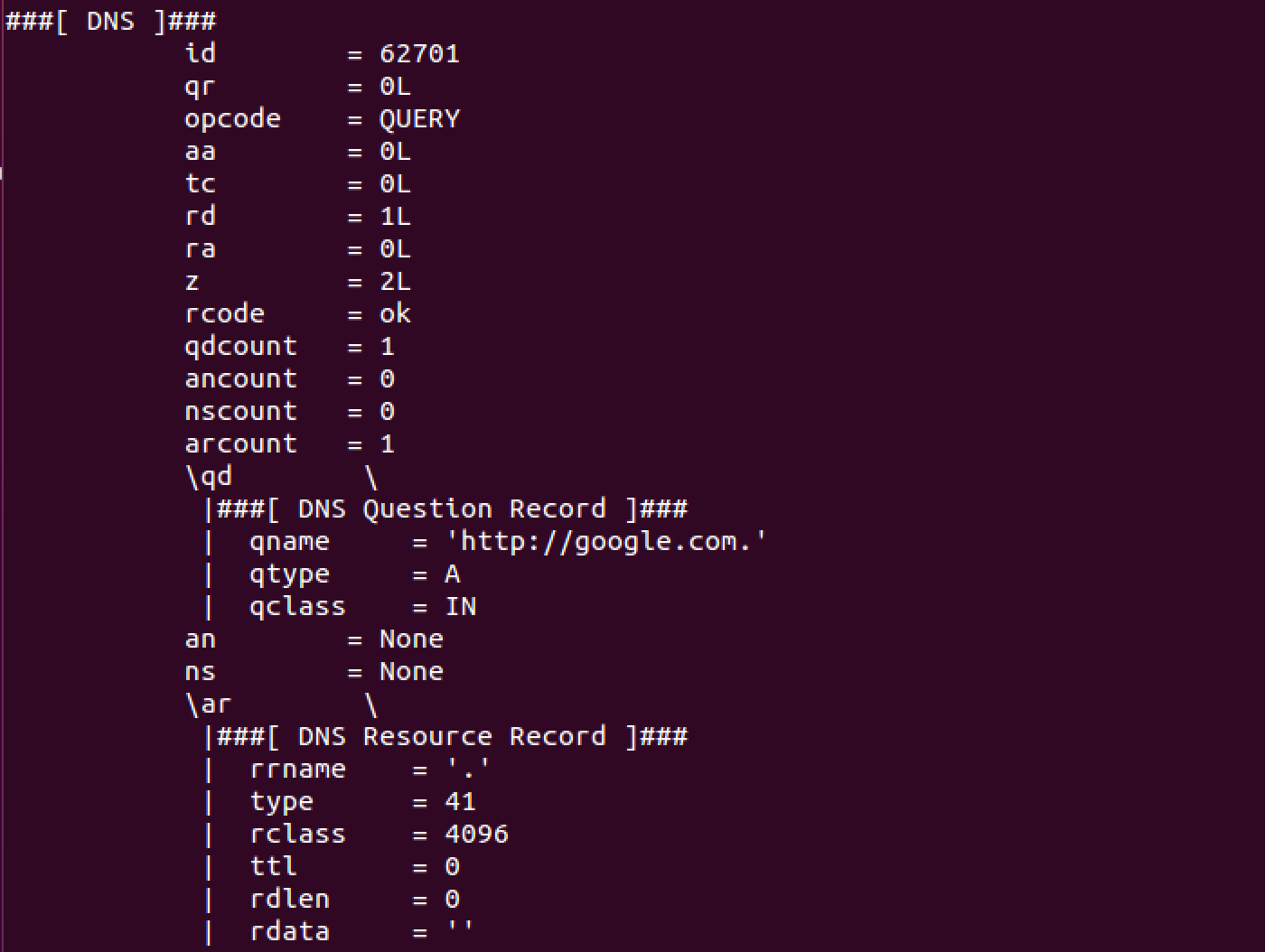
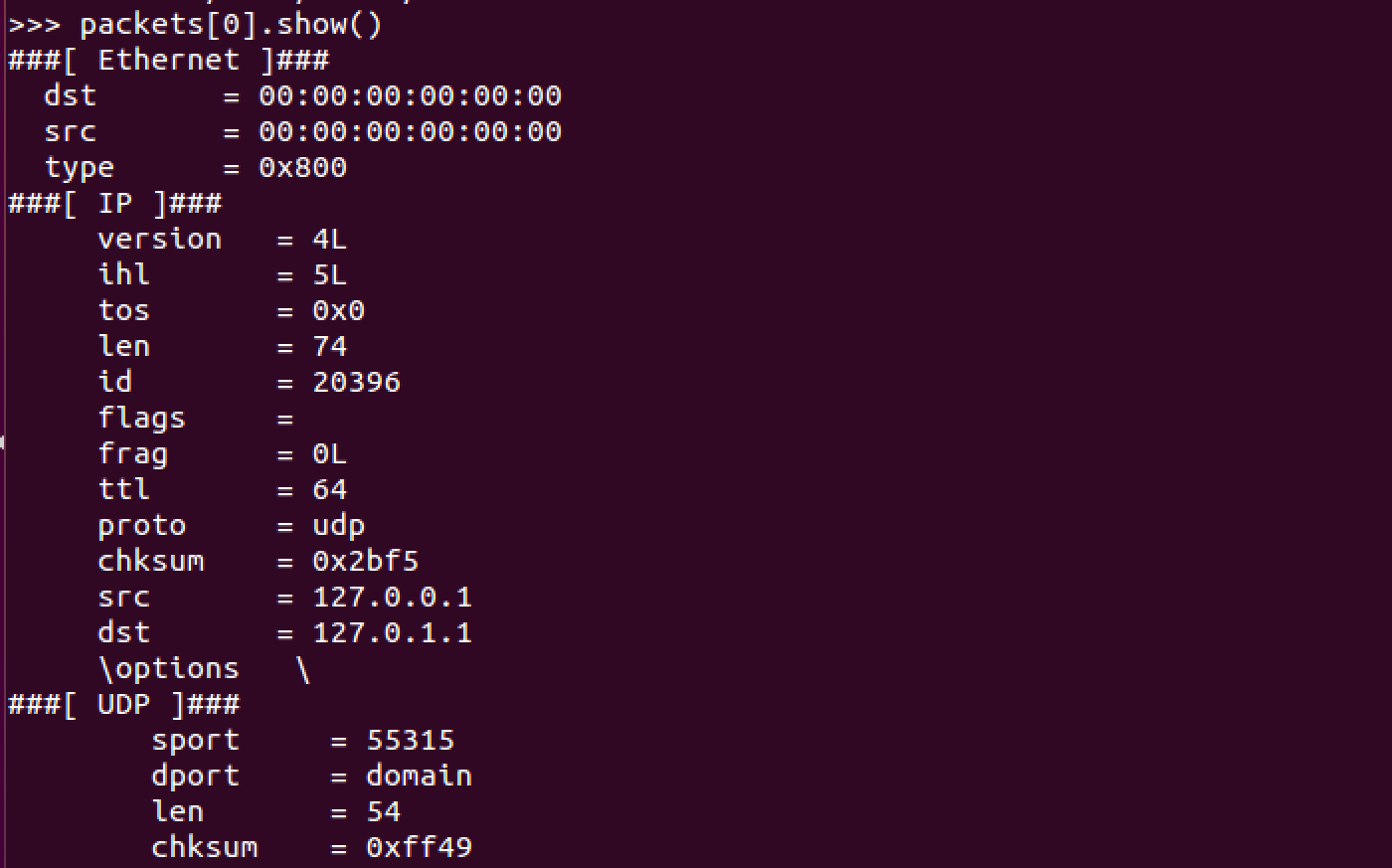
(Which is the linux (and osx) equivalent to nslookup)

And we get:

6 packets, 3 of them are request packets, and 3 of them answers from the DNS server.

Explanation: We see 3 requests probably because we didn’t got answer after a predefined timeout and the OS assumed the request packet was lost and tried again. We can also see that those request packets were not lost, and we got answers for all of them.

**Now we will review one of the captured packets in detail:**



**We can see the different layers here:** **Ethernet** in the **Link Layer**, **IP** in the **Network Layer**, **UDP** in the **Transport Layer** and finally **DNS** (Domain Name Service) in the **Application Layer**.

**Note**: We used sniff with ‘timeout’ instead of ‘count’ since we wanted to show only one flow of dns request-response. In general the count=x argument cause the sniffer to keep running until x packets are sniffed.