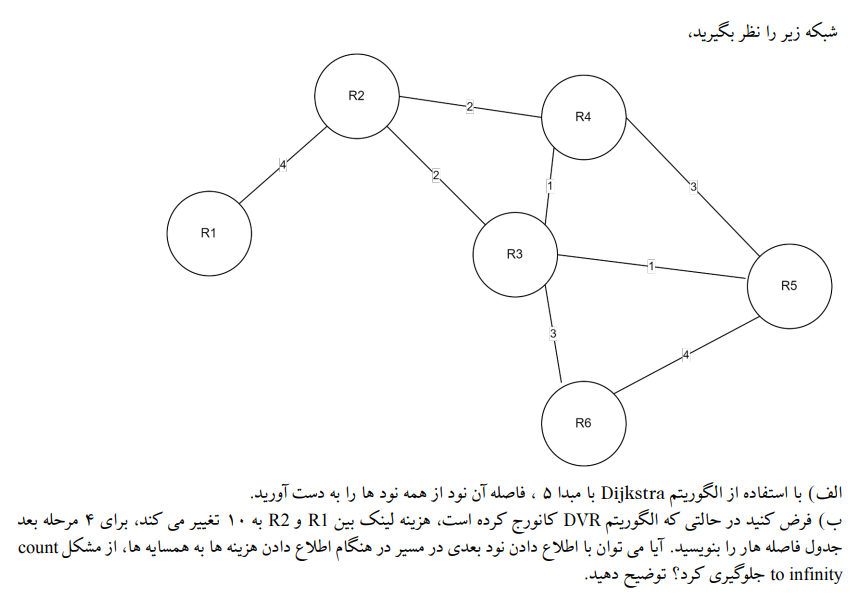


**تمرین سوم**

**سوال اول**

****

**پاسخ سوال اول:**

**الف)**

برای نود R5، الگوریتم Dijkstra را مرحله به مرحله اجرا می‌کنیم.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| D(R6), p(R6) | D(R4), p(R4) | D(R3), p(R3) | D(R2), p(R2) | D(R1), p(R1) | N’ | Step |
| 4, R5 | 3, R5 | 1, R5 | inf | inf | R5 | 0 |
|  |  |  |  |  |  | 1 |
|  |  |  |  |  |  | 2 |
|  |  |  |  |  |  | 3 |
|  |  |  |  |  |  | 4 |
|  |  |  |  |  |  | 5 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| D(R6), p(R6) | D(R4), p(R4) | D(R3), p(R3) | D(R2), p(R2) | D(R1), p(R1) | N’ | Step |
| 4, R5 | 3, R5 | 1, R5 | inf | inf | R5 | 0 |
| 4, R5 | 2, R3 |  | 3, R3 | inf | R5, R3 | 1 |
|  |  |  |  |  |  | 2 |
|  |  |  |  |  |  | 3 |
|  |  |  |  |  |  | 4 |
|  |  |  |  |  |  | 5 |

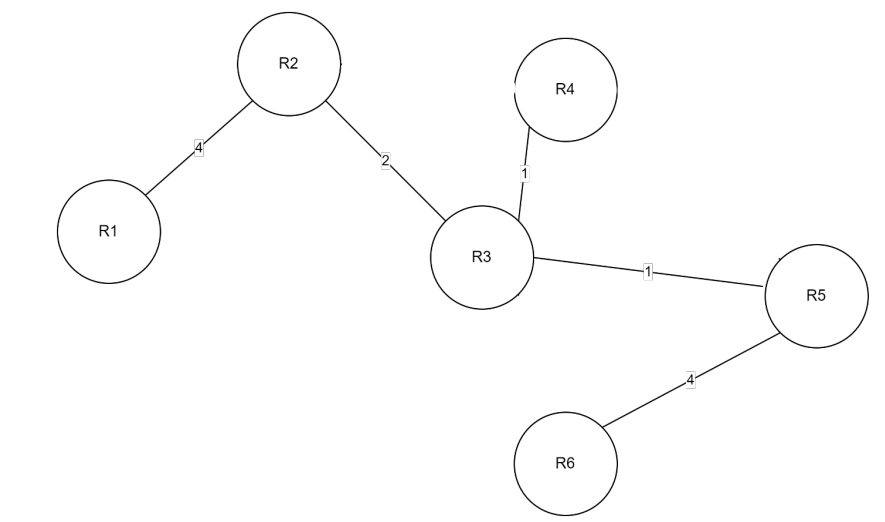
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| D(R6), p(R6) | D(R4), p(R4) | D(R3), p(R3) | D(R2), p(R2) | D(R1), p(R1) | N’ | Step |
| 4, R5 | 3, R5 | 1, R5 | inf | inf | R5 | 0 |
| 4, R5 | 2, R3 |  | 3, R3 | inf | R5, R3 | 1 |
| 4, R5 |  |  | 3, R3 | inf | R5, R3, R4 | 2 |
|  |  |  |  |  |  | 3 |
|  |  |  |  |  |  | 4 |
|  |  |  |  |  |  | 5 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| D(R6), p(R6) | D(R4), p(R4) | D(R3), p(R3) | D(R2), p(R2) | D(R1), p(R1) | N’ | Step |
| 4, R5 | 3, R5 | 1, R5 | inf | inf | R5 | 0 |
| 4, R5 | 2, R3 |  | 3, R3 | inf | R5, R3 | 1 |
| 4, R5 |  |  | 3, R3 | inf | R5, R3, R4 | 2 |
| 4, R5 |  |  |  | 7, R2 | R5, R3, R4, R2 | 3 |
|  |  |  |  |  |  | 4 |
|  |  |  |  |  |  | 5 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| D(R6), p(R6) | D(R4), p(R4) | D(R3), p(R3) | D(R2), p(R2) | D(R1), p(R1) | N’ | Step |
| 4, R5 | 3, R5 | 1, R5 | inf | inf | R5 | 0 |
| 4, R5 | 2, R3 |  | 3, R3 | inf | R5, R3 | 1 |
| 4, R5 |  |  | 3, R3 | inf | R5, R3, R4 | 2 |
| 4, R5 |  |  |  | 7, R2 | R5, R3, R4, R2 | 3 |
|  |  |  |  | 7, R2 | R5, R3, R4, R2, R6 | 4 |
|  |  |  |  |  |  | 5 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| D(R6), p(R6) | D(R4), p(R4) | D(R3), p(R3) | D(R2), p(R2) | D(R1), p(R1) | N’ | Step |
| 4, R5 | 3, R5 | 1, R5 | inf | inf | R5 | 0 |
| 4, R5 | 2, R3 |  | 3, R3 | inf | R5, R3 | 1 |
| 4, R5 |  |  | 3, R3 | inf | R5, R3, R4 | 2 |
| 4, R5 |  |  |  | 7, R2 | R5, R3, R4, R2 | 3 |
|  |  |  |  | 7, R2 | R5, R3, R4, R2, R6 | 4 |
| **4**, R5 | **2**, R3 | **1**, R5 | **3**, R3 | **7**, R2 | R5, R3, R4, R2, R6, R1 | 5 |

کمترین فاصله R5 تا هر نود را مشاهده می‌کنید. به شکل زیر رسیدیم:

****

**ب)**

جدول را هنگامی که الگوریتم کانورج کرده است نمایش می‌دهیم:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| R6 | R5 | R4 | R3 | R2 | R1 |  |
| 9 | 7 | 6 | 6 | 4 | 0 | R1 |
| 5 | 3 | 2 | 2 | 0 | 4 | R2 |
| 3 | 1 | 1 | 0 | 2 | 6 | R3 |
| 4 | 2 | 0 | 1 | 2 | 6 | R4 |
| 4 | 0 | 2 | 1 | 3 | 7 | R5 |
| 0 | 4 | 4 | 3 | 5 | 9 | R6 |

حال با تغییر هزینه لینک گفته شده، ابتدا جدول R1 و R2 آپدیت می‌شوند.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| R6 | R5 | R4 | R3 | R2 | R1 |  |
| 15 | 13 | 12 | 12 | 10 | 0 | R1 |
| 5 | 3 | 2 | 2 | 0 | 10 | R2 |
| 3 | 1 | 1 | 0 | 2 | 6 | R3 |
| 4 | 2 | 0 | 1 | 2 | 6 | R4 |
| 4 | 0 | 2 | 1 | 3 | 7 | R5 |
| 0 | 4 | 4 | 3 | 5 | 9 | R6 |

حال R2 به همسایه‌هایش خبر می‌دهد. در این حالت هر کدام از R3 و R4 درمی‌یابند که دیگری با هزینه 6 به R1 می‌رود و آنها با هزینه 1 بین خودشان می‌توانند با هزینه کمتر، به R1 بروند. همچنین دقت نمایید در این حالت R2 می‌بیند که خودش با هزینه 10 به R1 می‌رود. اما R3 هزینه 6 را معرفی کرده است. پس با هزینه 2 به R3 رفته و با هزینه 6 (قبل از آپدیت) عنوان شده از طرف R3، به R1 خواهد رفت. داریم:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| R6 | R5 | R4 | R3 | R2 | R1 |  |
| 15 | 13 | 12 | 12 | 10 | 0 | R1 |
| 5 | 3 | 2 | 2 | 0 | 8 | R2 |
| 3 | 1 | 1 | 0 | 2 | 7 | R3 |
| 4 | 2 | 0 | 1 | 2 | 7 | R4 |
| 4 | 0 | 2 | 1 | 3 | 7 | R5 |
| 0 | 4 | 4 | 3 | 5 | 9 | R6 |

حال نوبت آن است که R2 وR3 و R4 به همسایه‌های خود اطلاع بدهند. به همان دلیلی که در بخش قبل ذکر کردیم، مقدارها آپدیت می‌شوند. ( برای مثال R5 می‌بیند که R3 با هزینه 7 به R1 می‌رود و خودش هم با هزینه 1 به R3 می‌رود. بنابراین هزینه‌ش به 8 آپدیت می‌شود و ...):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| R6 | R5 | R4 | R3 | R2 | R1 |  |
| 15 | 13 | 12 | 12 | 10 | 0 | R1 |
| 5 | 3 | 2 | 2 | 0 | 9 | R2 |
| 3 | 1 | 1 | 0 | 2 | 8 | R3 |
| 4 | 2 | 0 | 1 | 2 | 8 | R4 |
| 4 | 0 | 2 | 1 | 3 | 8 | R5 |
| 0 | 4 | 4 | 3 | 5 | 10 | R6 |

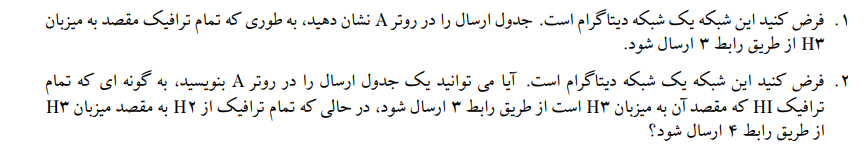
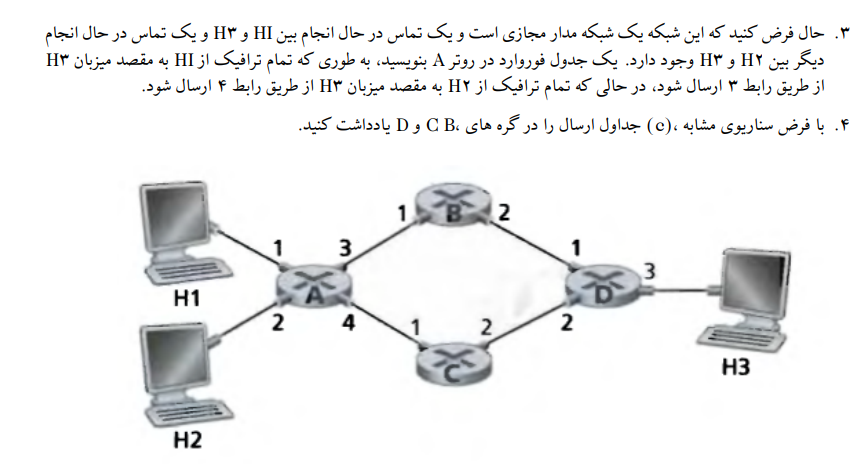
حال اگر ادامه بدهیم و 5 روتر نشانه گذاری شده به همسایه‌های خود خبر بدهند، با همان شرایط داریم (دقت

کنید R2 دیگر همان مسیر مستقیم به R1 را ترجیح می‌دهد.) :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| R6 | R5 | R4 | R3 | R2 | R1 |  |
| 15 | 13 | 12 | 12 | 10 | 0 | R1 |
| 5 | 3 | 2 | 2 | 0 | 10 | R2 |
| 3 | 1 | 1 | 0 | 2 | 9 | R3 |
| 4 | 2 | 0 | 1 | 2 | 9 | R4 |
| 4 | 0 | 2 | 1 | 3 | 9 | R5 |
| 0 | 4 | 4 | 3 | 5 | 11 | R6 |

دقت کنید از این مرحله به بعد R2 آپدیتی نخواهد داشت و روی 10 خواهد ماند. همانطور که مشاهده می‌کنید در این چند گام برای R3 و R4 به مشکل count to infinity برخورد کردیم.

بله اگر step lookahead 2 داشته باشیم R3 و R4 متوجه مسیر از R2 می‌شوند و در حالت count to infinity قرار نمی‌گیرند.

**سوال دوم**

**پاسخ سوال دوم:**



|  |  |
| --- | --- |
| Link Interface | Destination Address |
| 3 | H3 |



خیر امکان ندارد. زیرا forwarding rule تنها بر اساس آدرس مقصد است و این حالت را نمی‌توان انجام داد.



در این بخش می‌دانیم یک virtual circuit network داریم . یکی از جواب‌های ممکن، VC forwarding table زیر می‌باشد:

|  |  |  |  |
| --- | --- | --- | --- |
| VC # | Outgoing Interface | VC # | Incoming Interface |
| 22 | 3 | 12 | 1 |
| 18 | 4 | 63 | 2 |



مطابق مقدار vc در بخش قبلی، جداول هر 4 router را رسم می‌نماییم:

A:

|  |  |  |  |
| --- | --- | --- | --- |
| VC # | Outgoing Interface | VC # | Incoming Interface |
| 22 | 3 | 12 | 1 |
| 18 | 4 | 63 | 2 |

B:

|  |  |  |  |
| --- | --- | --- | --- |
| VC # | Outgoing Interface | VC # | Incoming Interface |
| 24 | 2 | 22 | 1 |

C:

|  |  |  |  |
| --- | --- | --- | --- |
| VC # | Outgoing Interface | VC # | Incoming Interface |
| 50 | 2 | 18 | 1 |

D:

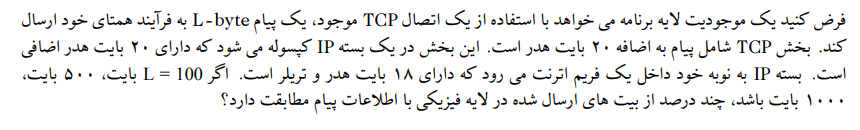
|  |  |  |  |
| --- | --- | --- | --- |
| VC # | Outgoing Interface | VC # | Incoming Interface |
| 70 | 3 | 24 | 1 |
| 76 | 3 | 50 | 2 |

همانطور که مشاهده می‌کنید، در روتر A فرض می‌کنیم مقدار vc از دو درگاه ورودی متفاوت است. حال اگر از درگاه اول ورودی گرفتیم، آن را از درگاه سوم با مقدار vc برابر با 22 خارج می‌نماییم. همچنین اگر درگاه دوم ورودی گرفتیم، آن را از درگاه چهارم با مقدار vc برابر با 18 خارج می‌نماییم.

حال برای روتر B، اگر از درگاه اول با مقدار vc برابر 22 که همان خروجی درگاه سوم روتر A هست، ورودی بگیریم، آن را از درگاه دوم با مقدار vc برابر 24 خارج می‌کنیم.

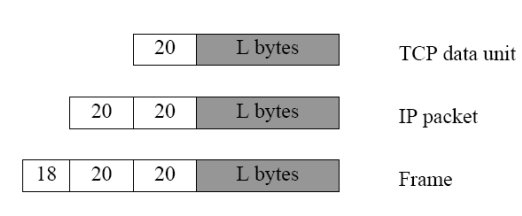
همین روند را برای روتر C داریم با این تفاوت که اگر از درگاه اول با مقدار vc برابر 18 که همان خروجی درگاه چهارم روتر A هست، ورودی بگیریم، آن را از درگاه دوم با مقدار vc برابر 50 خارج می‌کنیم.

حال برای روتر D، 2 ورودی داریم. یکی خروجی روتر B با مقدار vc برابر 24، و دیگری خروجی روتر C با مقدار vc برابر با 50. طبق جدول اگر این مقادیر را از درگاه‌های مربوط مطابق شکل دریافت نماییم، هر دو از تنها درگاه خروجی یعنی درگاه سوم خارج شده و به H3 می‌روند.

**سوال سوم**

**پاسخ سوال سوم:**

طبق اطلاعات صورت سوال، داریم:

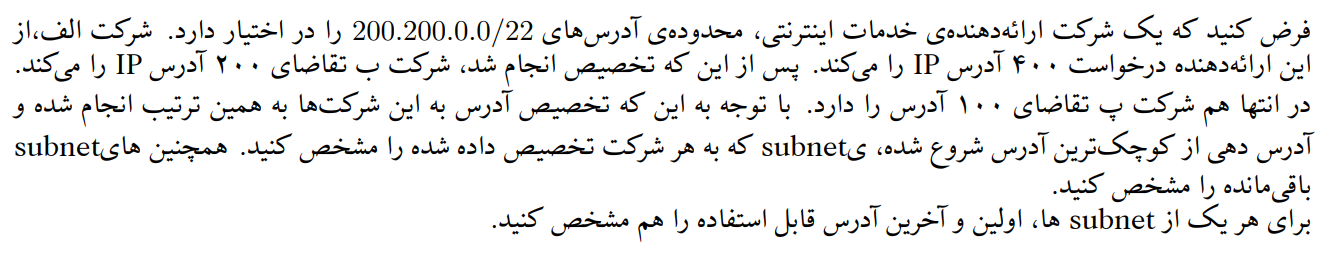


بنابراین:

حال برای مقادیر مختلف داریم:

|  |  |
| --- | --- |
| percentage of the transmitted bits in the physical layer corresponds to the message information | L (bytes) |
|  | 100 |
|  | 500 |
|  | 1000 |

**سوال چهارم**

****

**پاسخ سوال چهارم:**

ابتدا توجه کنید که داریم:

شرکت الف تقاضای 400 آدرس را دارد. چون توانی از 2 اختصاص می‌دهیم، 9 بیت برای شرکت الف نیاز است.

شرکت ب تقاضای 200 آدرس را دارد. چون توانی از 2 اختصاص می‌دهیم، 8 بیت برای شرکت ب نیاز است.

شرکت پ تقاضای 100 آدرس را دارد. چون توانی از 2 اختصاص می‌دهیم، 7 بیت برای شرکت پ نیاز است.

طبق محدوده آدرس داده شده، 10 بیت را در اختیار داریم که با توجه به نیاز شرکت‌ها می‌توانیم مانند جدول زیر عمل بنماییم:

|  |  |
| --- | --- |
|  | |
|  | شرکت الف |
|  | شرکت ب |
|  | شرکت پ |

بنابراین برای آدرس شروع و پایان داریم:

شرکت الف:

شرکت ب:

شرکت پ:

بنابراین برای subnetهای باقی‌مانده از آدرس 200.200.3.128/25 خواهیم داشت.

توجه کنید که مطابق گفته‌های تی‌ای در گروه تلگرامی، مشکلی مبنی بر توان 2 گرفتن برای آدرس‌ها نیست. برای همین شرکت الف که 400 آدرس میخواست، اکنون آدرس دارد.

هم‌چنین برای آدرس قابل استفاده دقت کنید که اولین آدرس و آخرین آدرس برای هر شرکت غیرقابل استفاده است. در واقع داریم:

شرکت الف:

اولین آدرس قابل استفاده: 200.200.0.1 (200.200.0.0 برای آدرس subnet رزرو شده است( آخرین آدرس قابل استفاده: 200.200.1.254 (200.200.1.255 برای broadcast رزرو شده است(

شرکت ب:

اولین آدرس قابل استفاده: 200.200.2.1

آخرین آدرس قابل استفاده 200.200.2.254 :

شرکت پ:

اولین آدرس قابل استفاده: 200.200.3.1

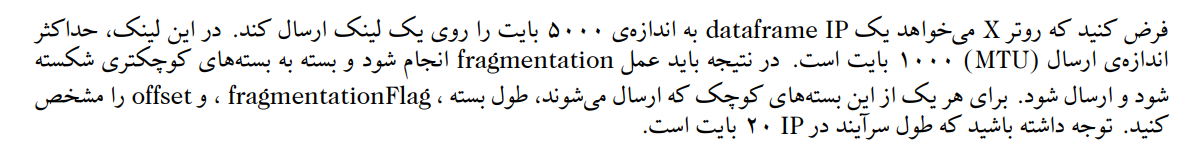
آخرین آدرس قابل استفاده 200.200.3.126 :

باقی‌مانده:

اولین آدرس قابل استفاده: 200.200.3.129

آخرین آدرس قابل استفاده 200.200.3.254 :

**سوال پنجم**

****

**پاسخ سوال پنجم:**

می‌دانیم اندازه dataframe برابر 5000 بایت و MTU برابر 1000 می‌باشد. همچنین طول سرآیند برابر 20 بایت است. داریم:

Large dataframe:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Length = 5000 | ID = x | fragFlag = 0 | offset = 0 | Data = 4980 |

حال fragmentation را انجام می‌دهیم:



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Length = 996 | ID = x | fragFlag = 1 | offset = 0 | Data = 976 |



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Length = 996 | ID = x | fragFlag = 1 | offset = 122 | Data = 976 |



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Length = 996 | ID = x | fragFlag = 1 | offset = 244 | Data = 976 |



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Length = 996 | ID = x | fragFlag = 1 | offset = 366 | Data = 976 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Length = 996 | ID = x | fragFlag = 1 | offset = 488 | Data = 976 |

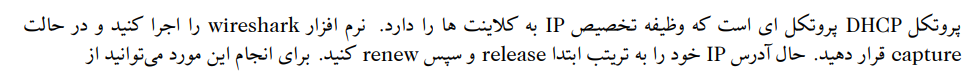
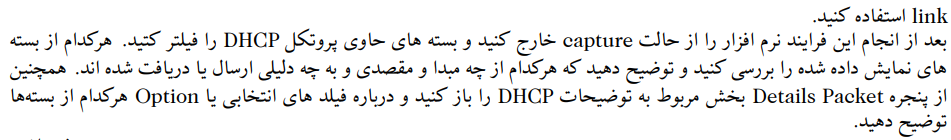


|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Length = 120 | ID = x | fragFlag = 0 | offset = 610 | Data = 100 |

دقت کنید اندازه دیتا باید حتما به 8 تقسیم‌پذیر باشه تا بتوانیم offset درستی ارائه دهیم. برای همین مقدار دیتا برابر 976 می‌باشد. همچنین مقدار دیتای تکه آخر از رابطه زیر به دست می‌آید:

همانطور که مشاهده می‌شود برای 5000 بایت اولیه، مجموع اندازه تکه‌ها برابر با شد و در واقع کارآیی برابر با می‌باشد.

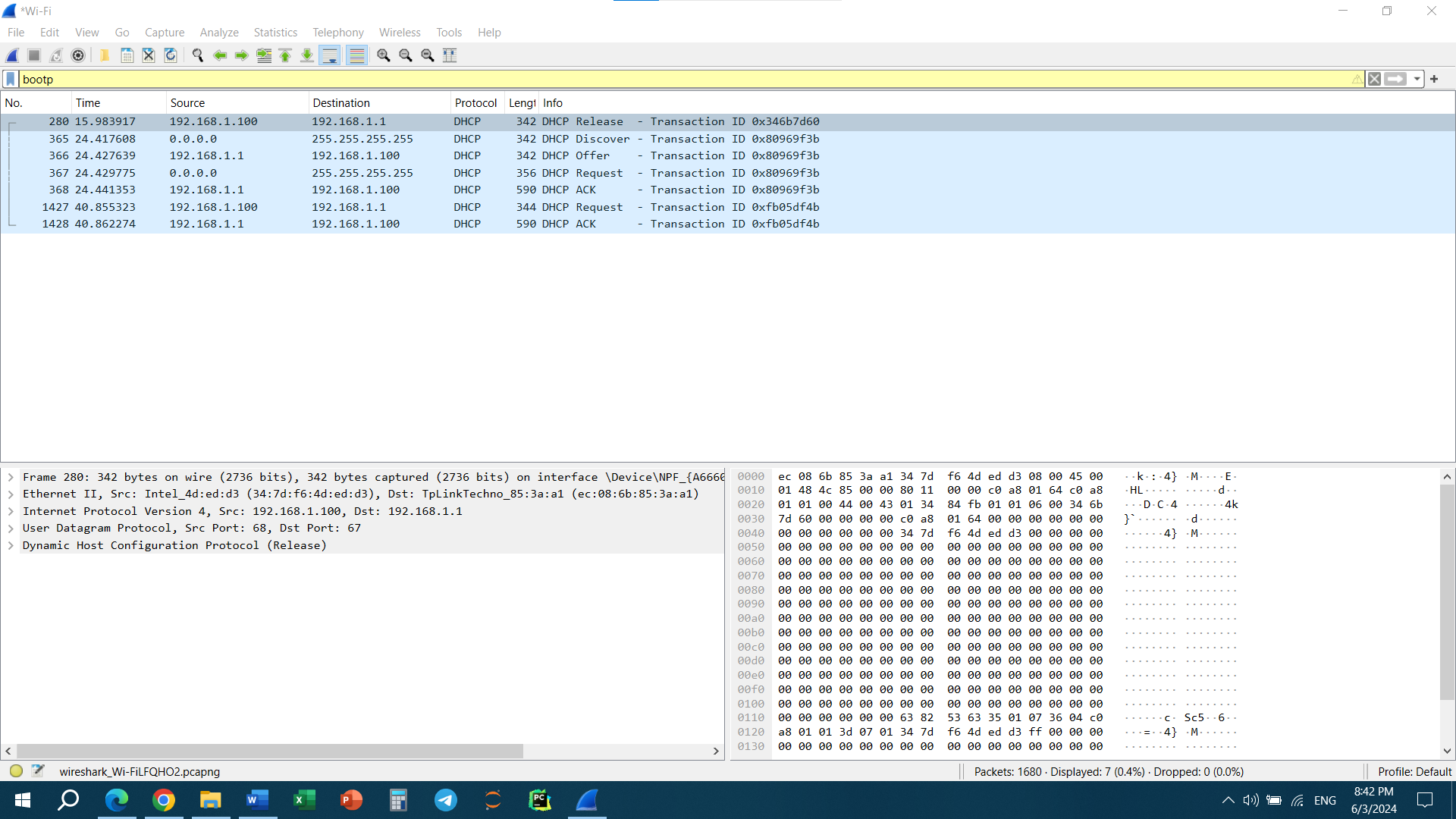
البته این بخش، خواسته مسئله نبوده است.

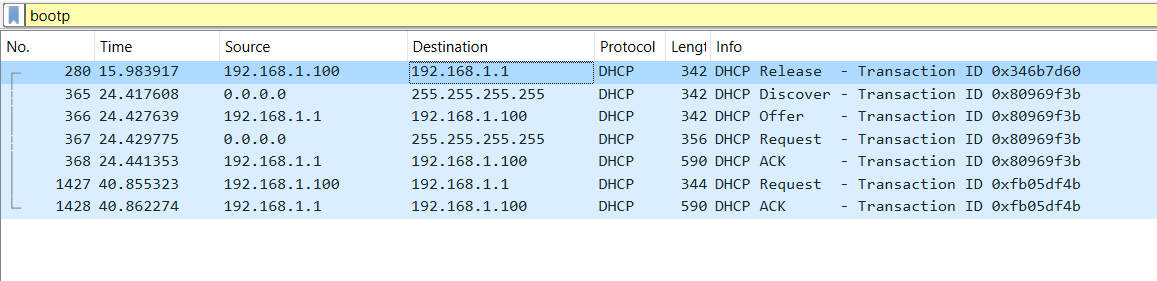
**سوال ششم:**

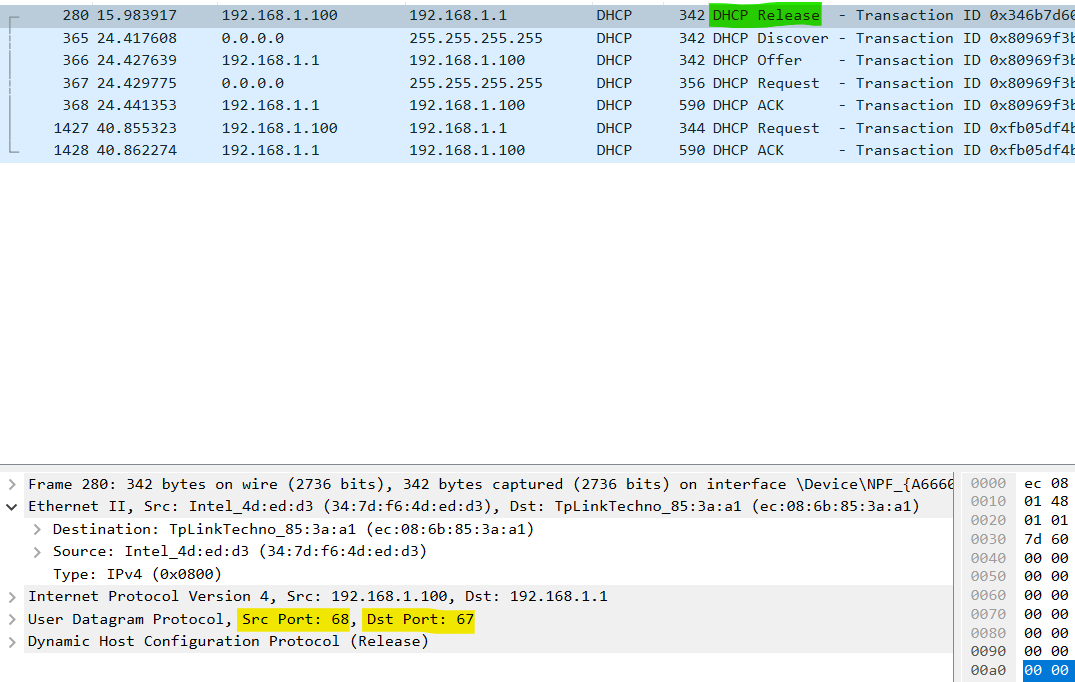
**پاسخ سوال ششم:**

ابتدا در ویندوز به کمک ipconfig/release و ipconfig /renew مراحل را انجام می‌دهیم.

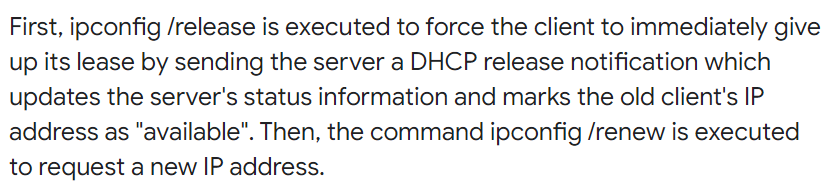
سپس از capture خارج کرده و به کمک bootp پیام‌های DHCP را فیلتر می‌کنیم. داریم:



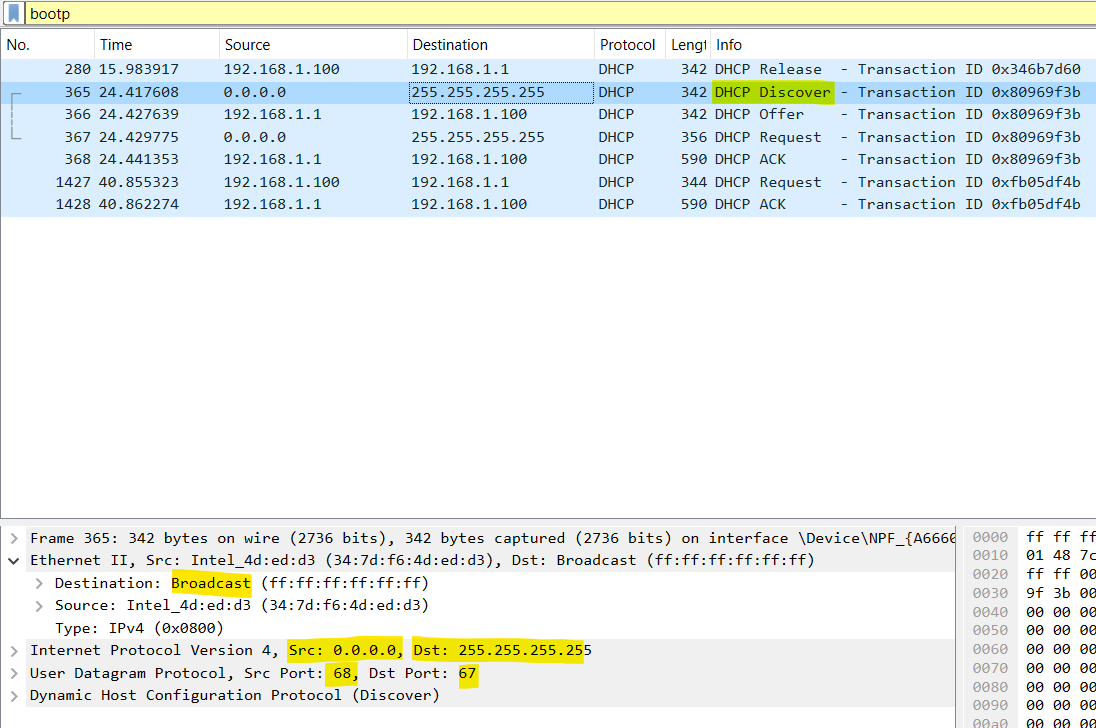
بسته اول:



همانطور که از شماره پورت مشخص است، از کلاینت به سرور است.

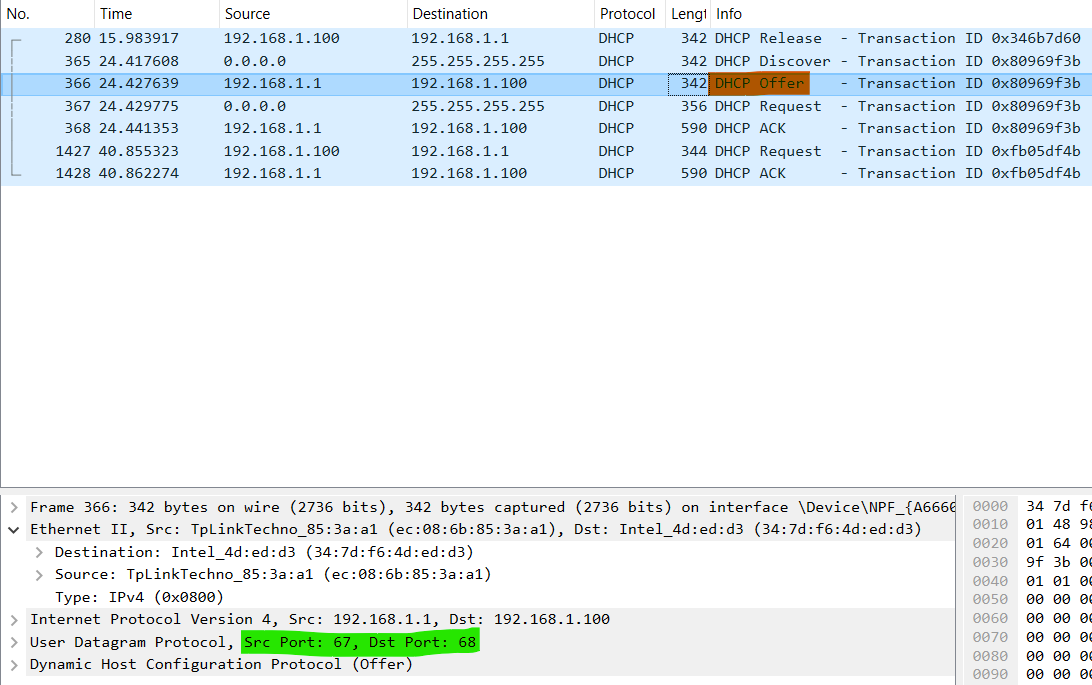


بسته دوم:



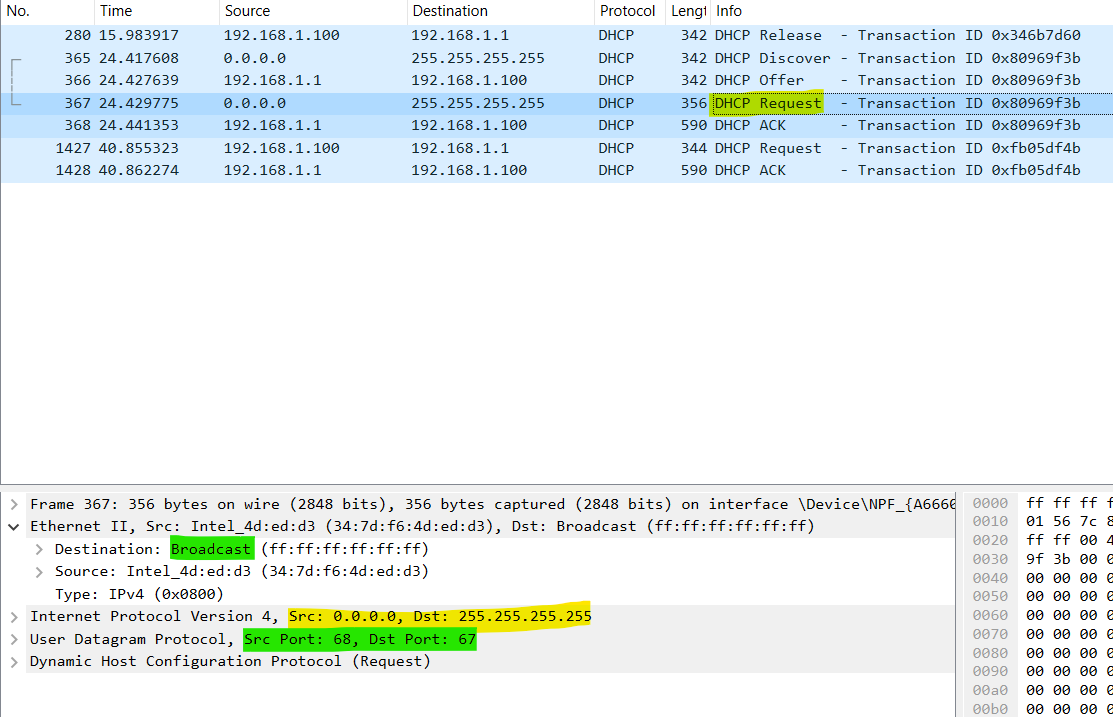
که از 0.0.0.0 source به 255.255.255.255 destination( حالت برادکست است) و به دنبال پیدا کردن DHCP سرور ها است و همانطور که در اسلایدها اشاره کردیم، در حالت برادکست و دنبال یافتن ادرس است.

بسته سوم:



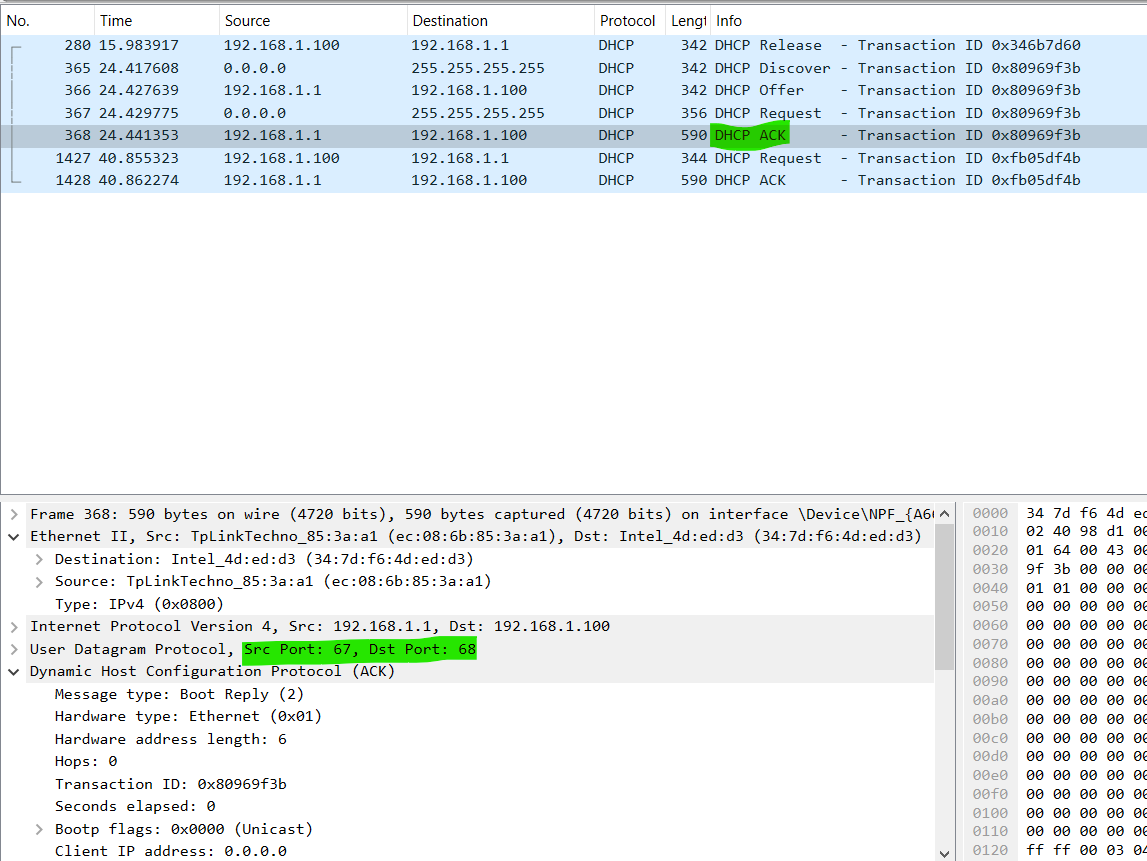
همانطور که از شماره پورت مشخص است، سرور پاسخ می‌دهد و ادرس موجود را آفر می‌کند.

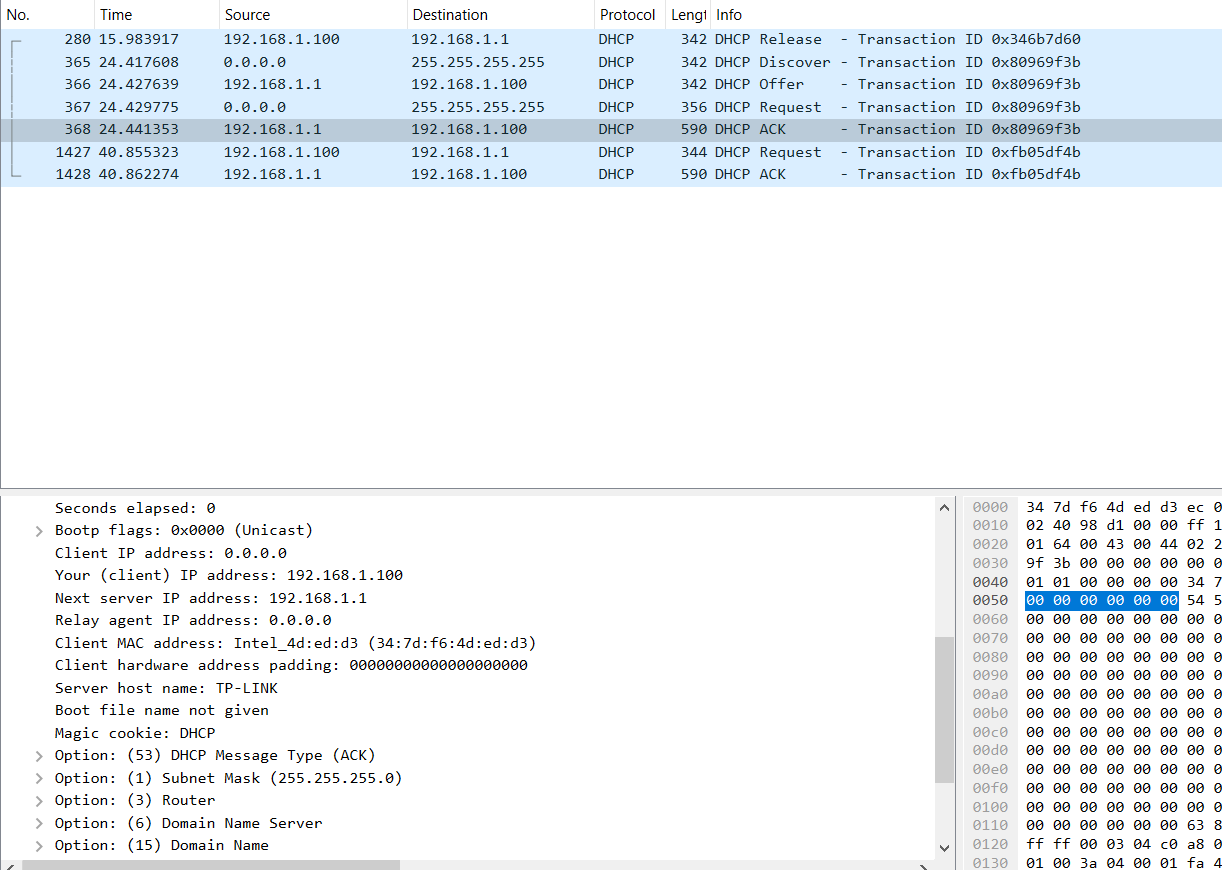
بسته چهارم:



حال مجددا کلاینت درخواست خود مبنی بر تایید ادرس گفته شده را می‌دهد.

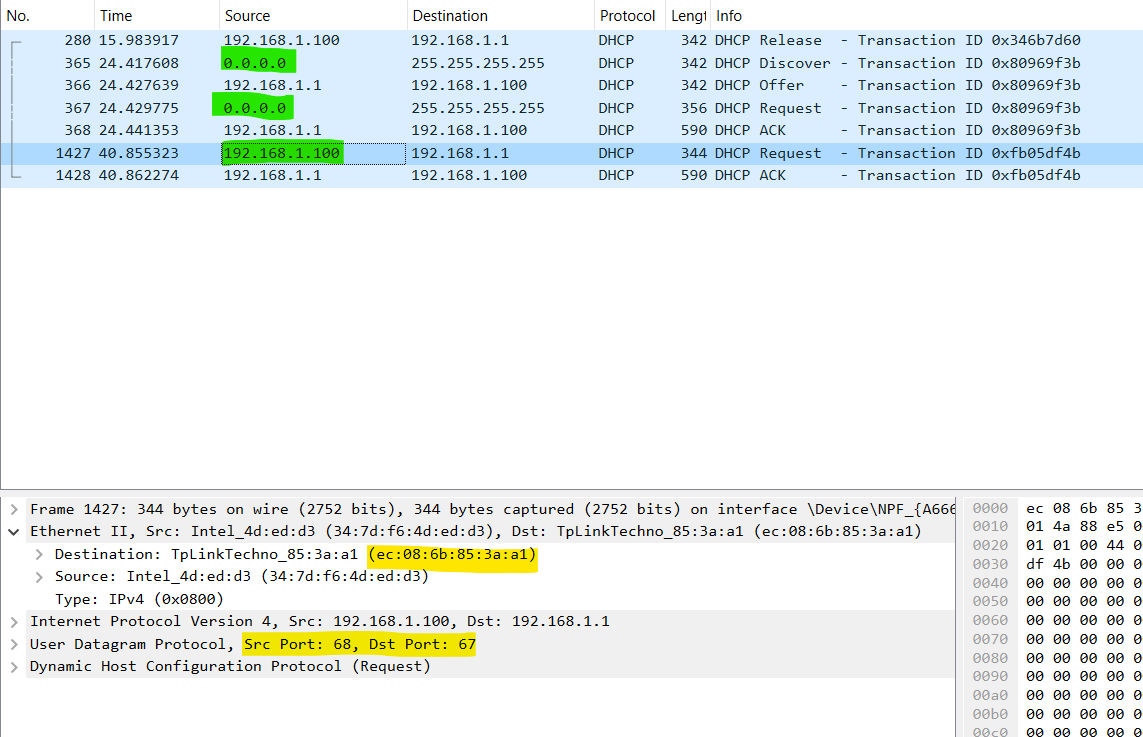
بسته پنجم:



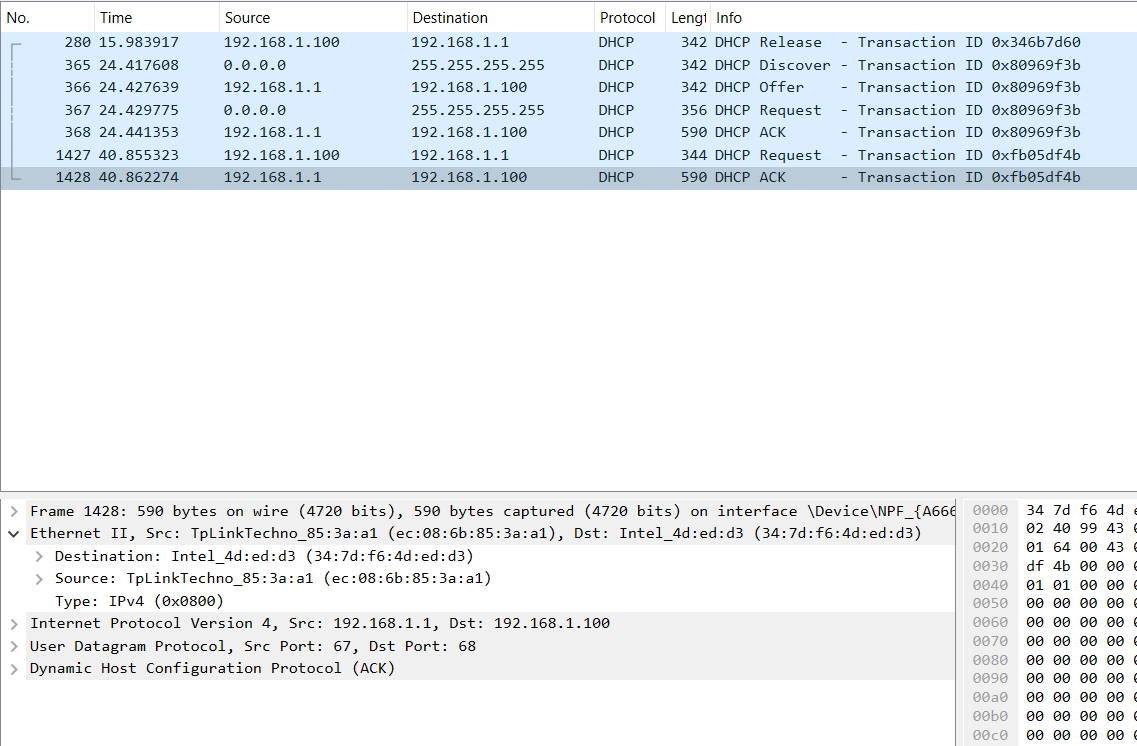


سرور ack مورد انتظار را خروجی می‌دهد تا کلاینت بتواند از آدرس استفاده نماید.

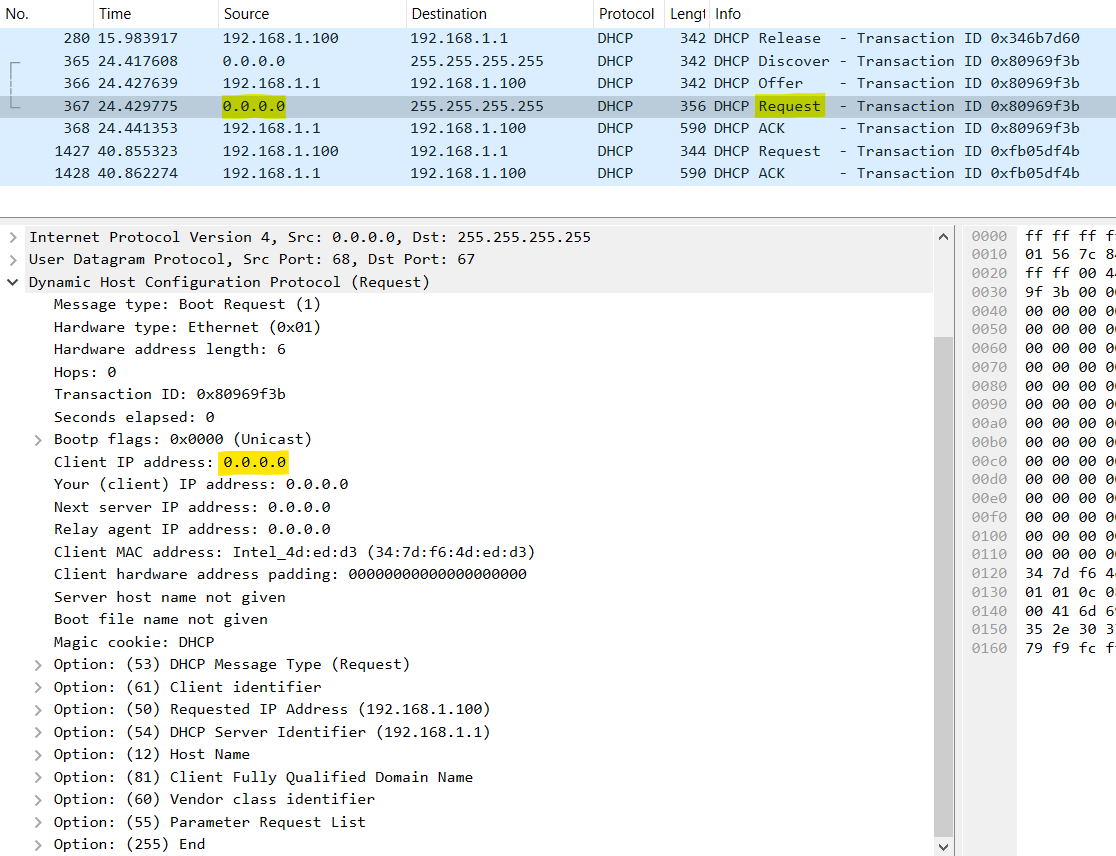
حال در دو بسته بعدی، کلاینت دیگر ادرس دارد و برابر 0.0.0.0 نیست.

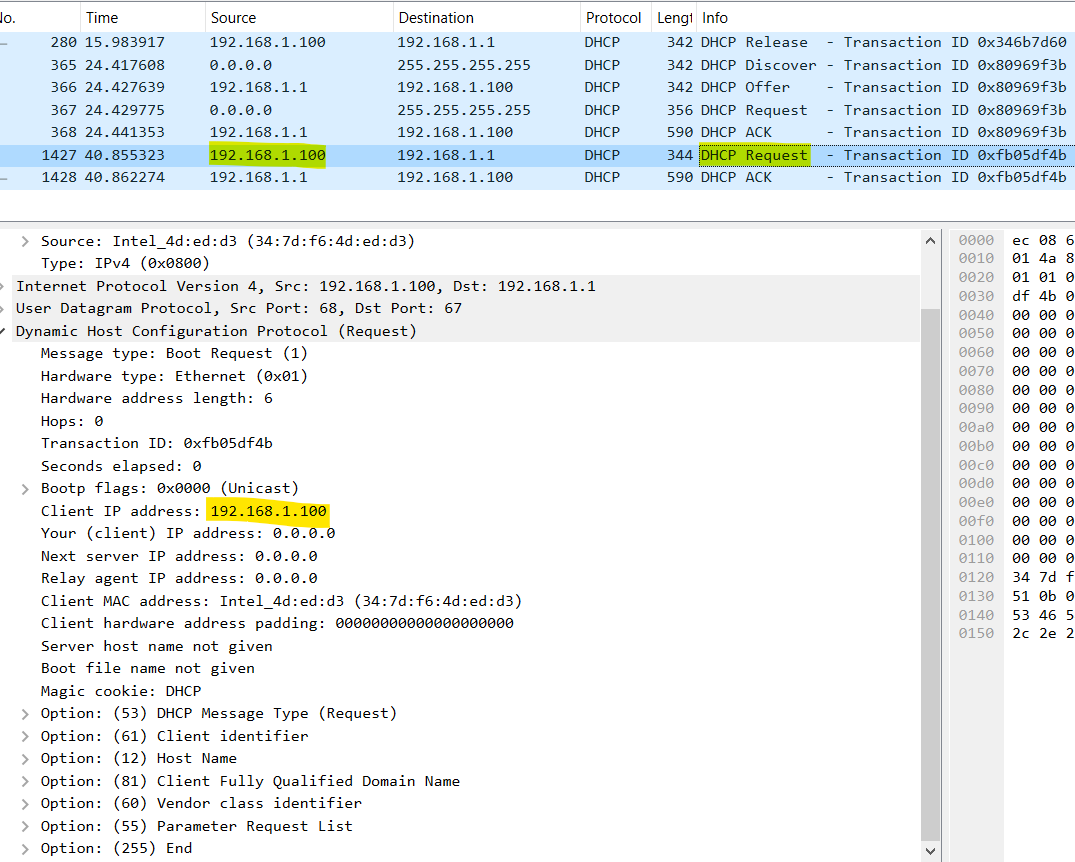


و در بسته بعدی دیگر سرور ادرس کلاینت را دارد:

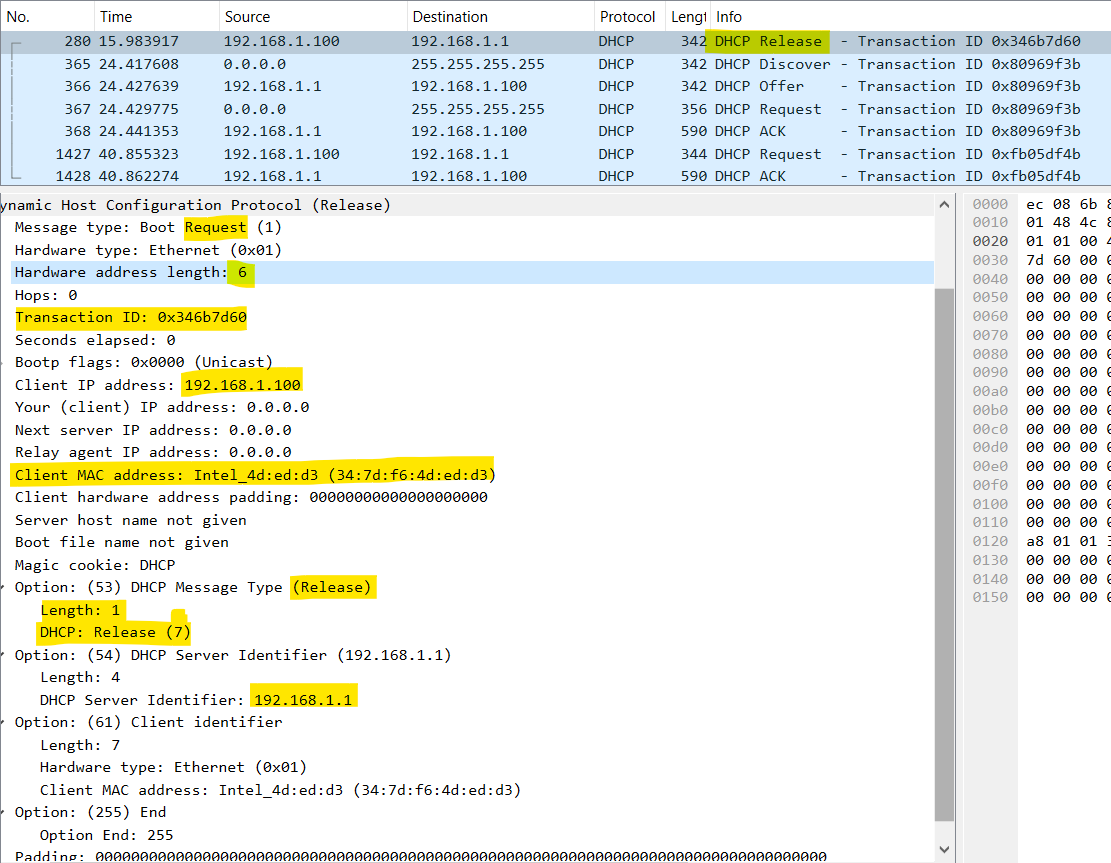


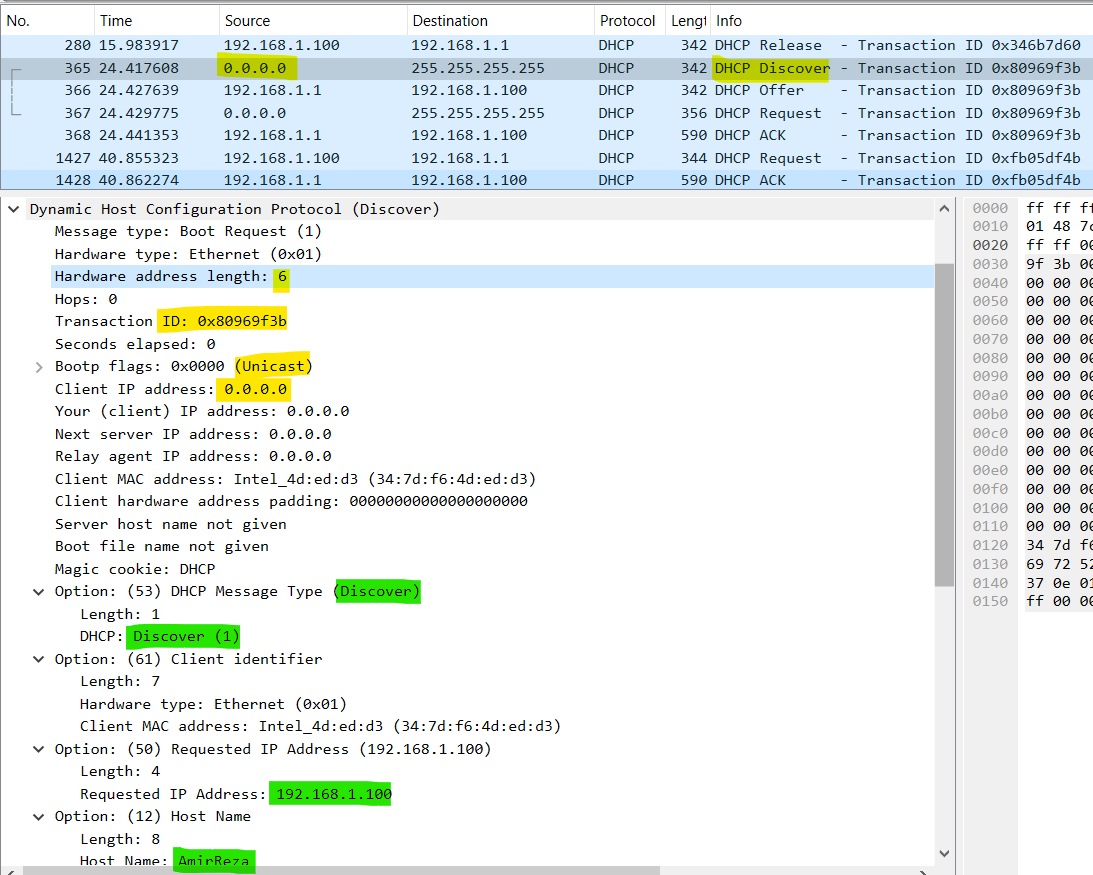
برای مشخص شدن بهتر، محتویات دو request نهایی را ببینید:



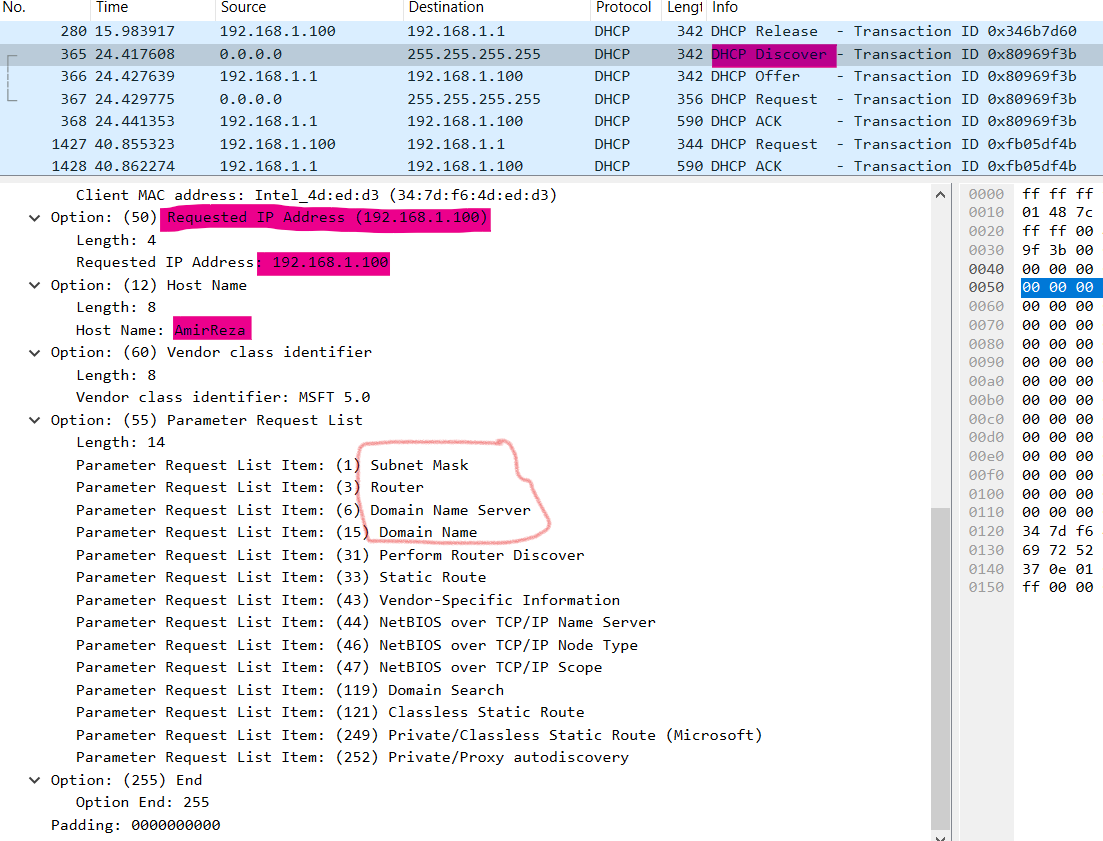


برای بخش details packet و بخش DHCP توضیحات را دادیم ولی تصاویر بیشتر را قرار می‌دهیم:

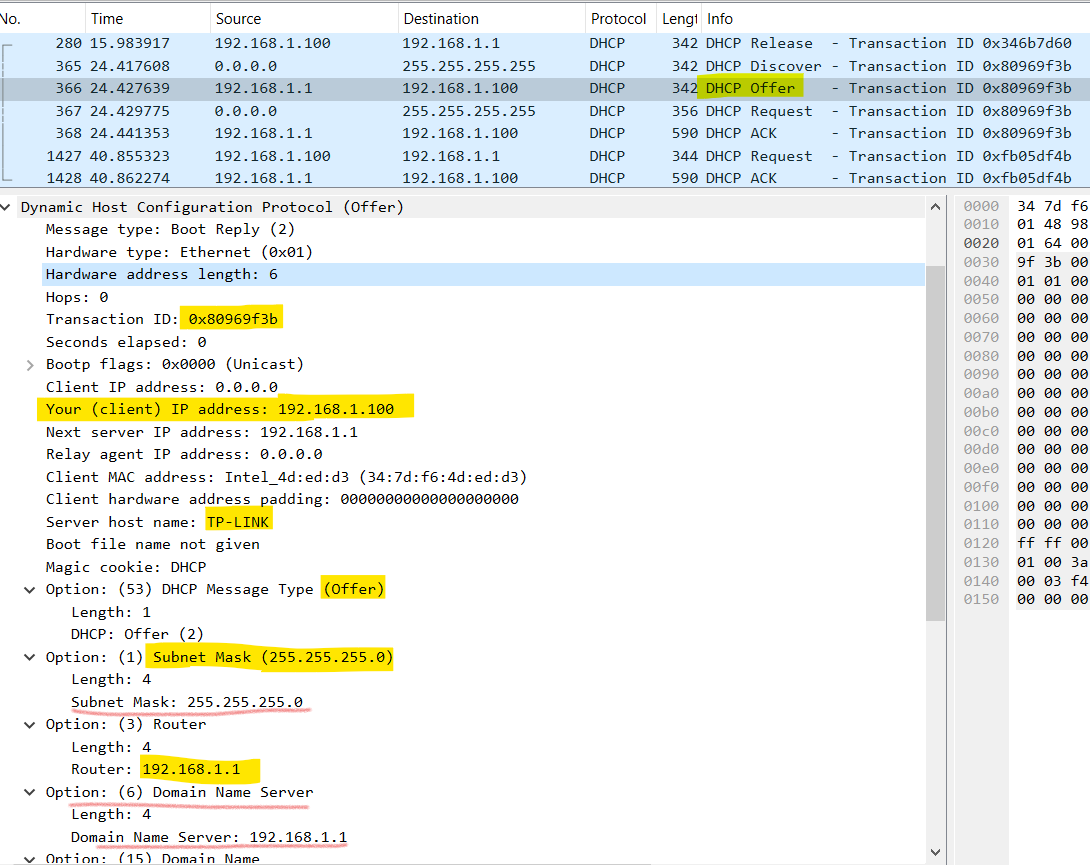


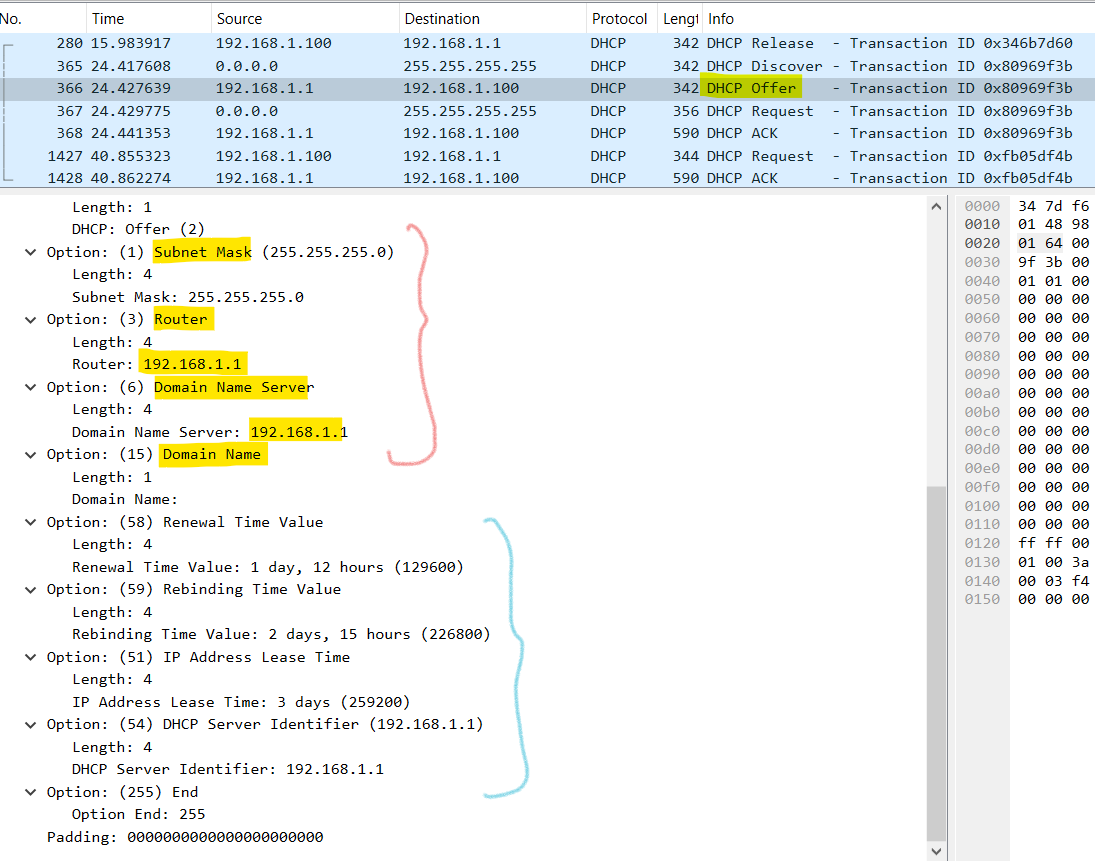


عکس زیر بسیار مهم است:

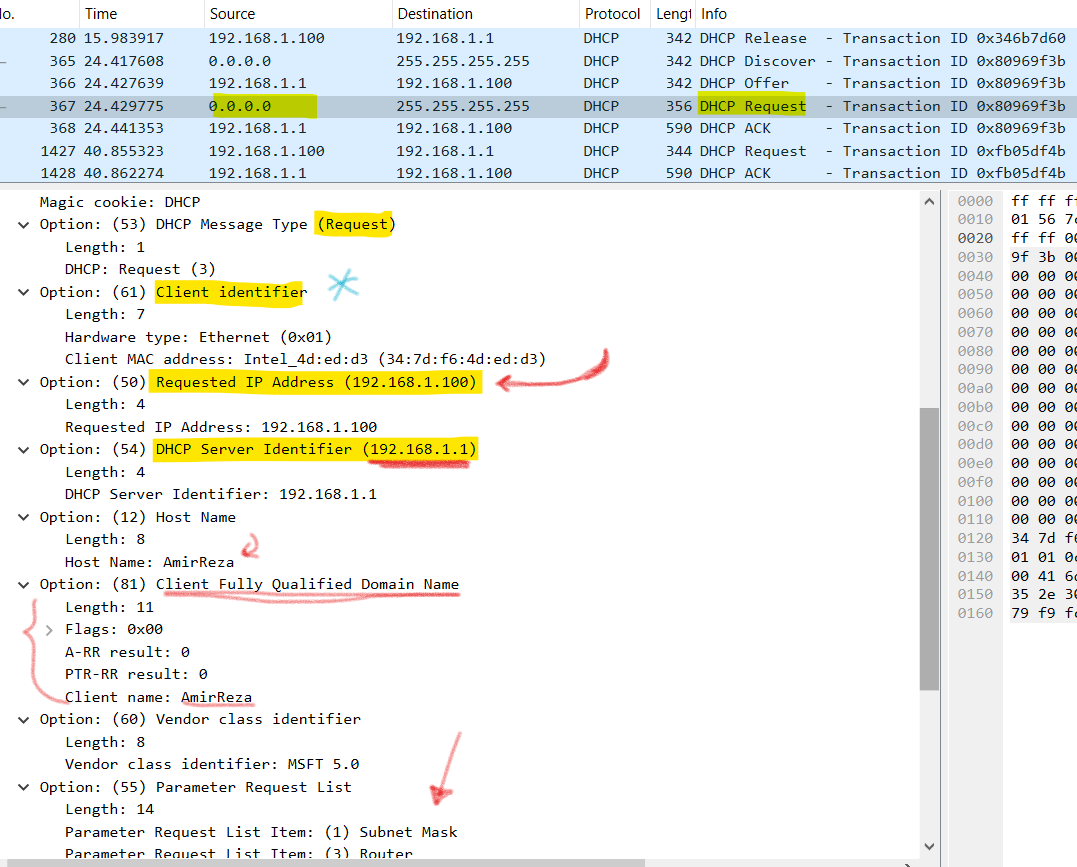


همانطور که میبینید در بخش discover، خواسته های مورد نیاز که DHCP فراهم می‌کند، لیست شده‌اند. در دو عکس بعدی و در offer سرور، داده ها داده شده است.





در بخش request داریم:



و ack سرور:

