Laboratorinio darbo ataskaita

VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

INFORMACINIŲ SISTEMŲ INŽINERIJA, 3 KURSAS

Laboratorinis darbas 3 – Kompiuterių tinklo duomenų srauto analizė

KOMPIUTERIŲ TINKLAI

Atliko: Martynas Jašinskas

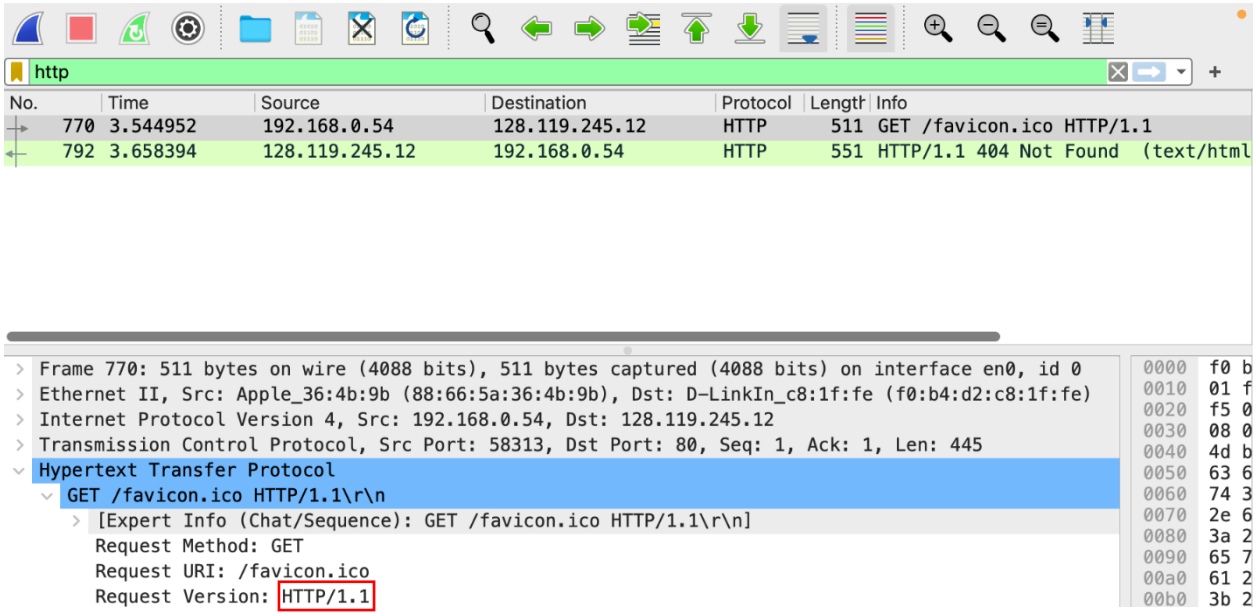
VU el. p: martynas.jasinskas@mif.stud.vu.lt

Vilnius, 2022

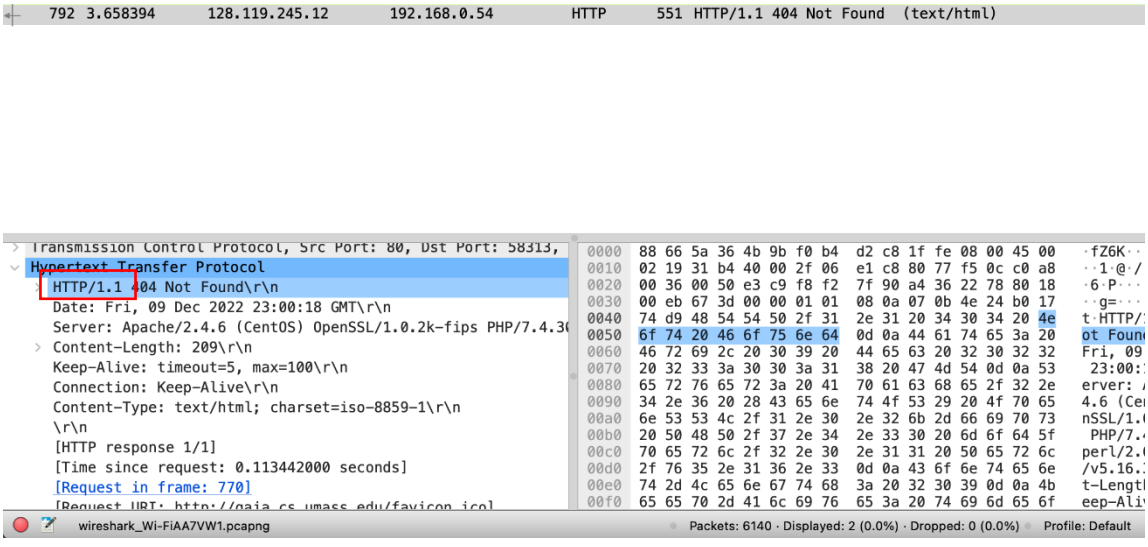
HTTP paketų filtravimas

1. Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running?

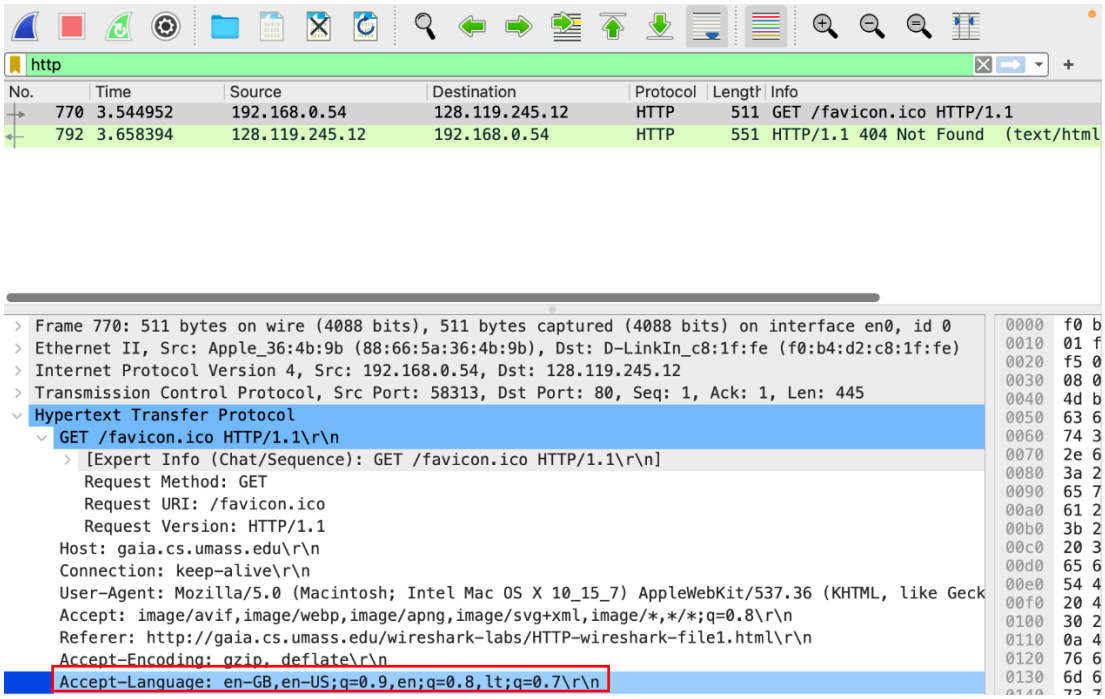
My browser:



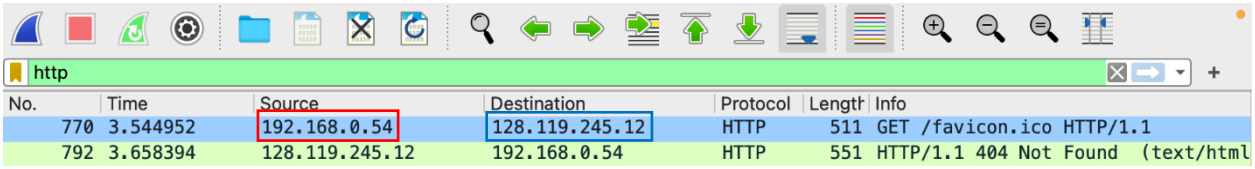
Server:



2. What languages (if any) does your browser indicate that it can accept to the server?



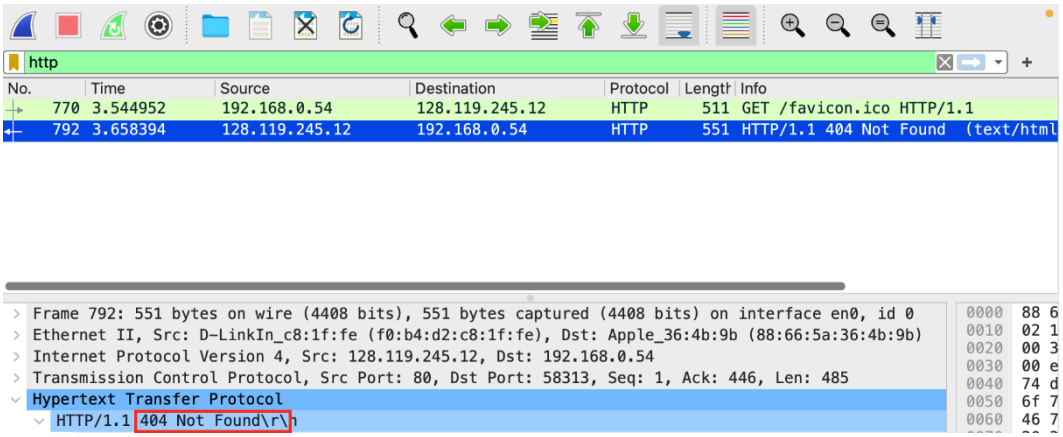
3. What is the IP address of your computer? Of the gaia.cs.umass.edu server?



192.168.0.54 – Mano kompiuterio adresas.

128.119.245.12 – *Gaia.cs.umas.edu* adresas.

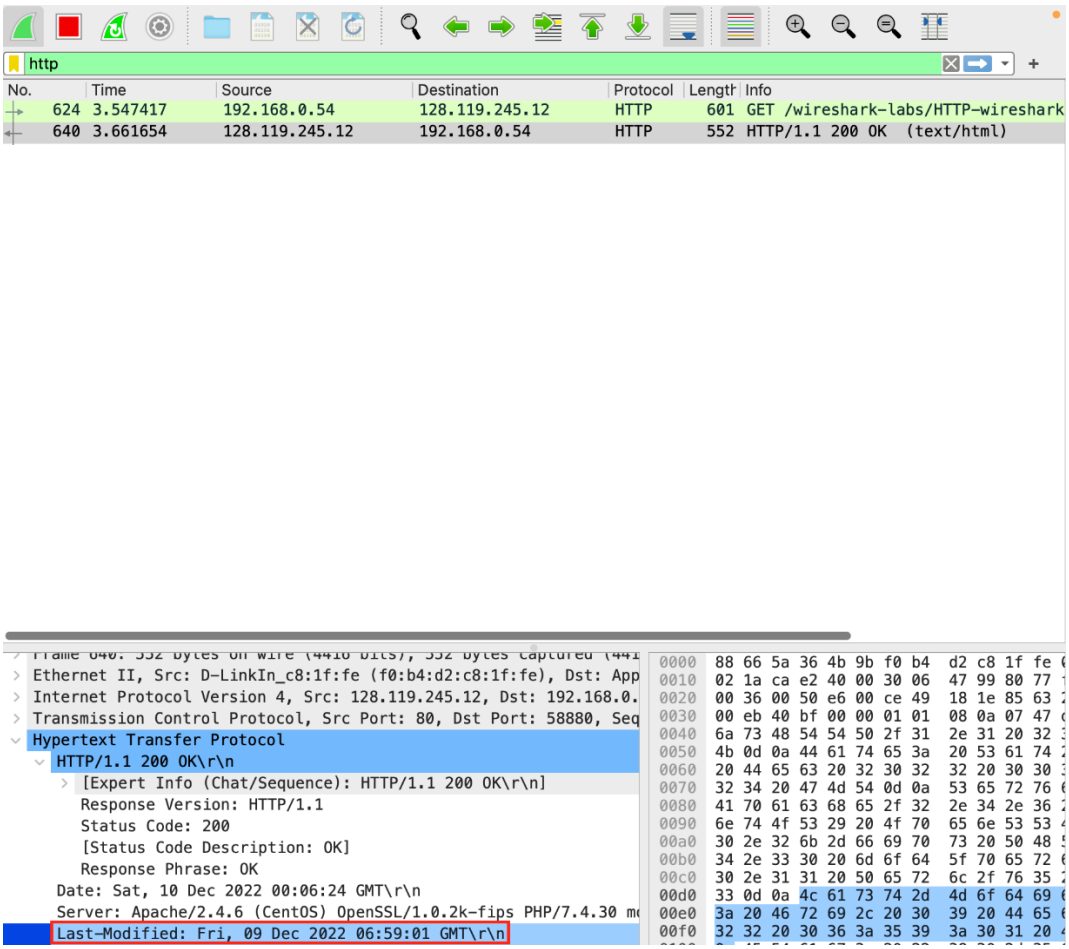
4. What is the status code returned from the server to your browser?



Čia buvo gautas 404 kodas, kuris rodo, kad toks resursas nebuvo rastas. Iš requesto galime matyti, kad mano browseris bandė prašyti kažkokios ikonėlės, vietoje reikiamo html failo. Galiausiai pasisekė surasti tuos paketus, todėl tolimesnėse užduotyse matytis paketai su reikiamu resursu.

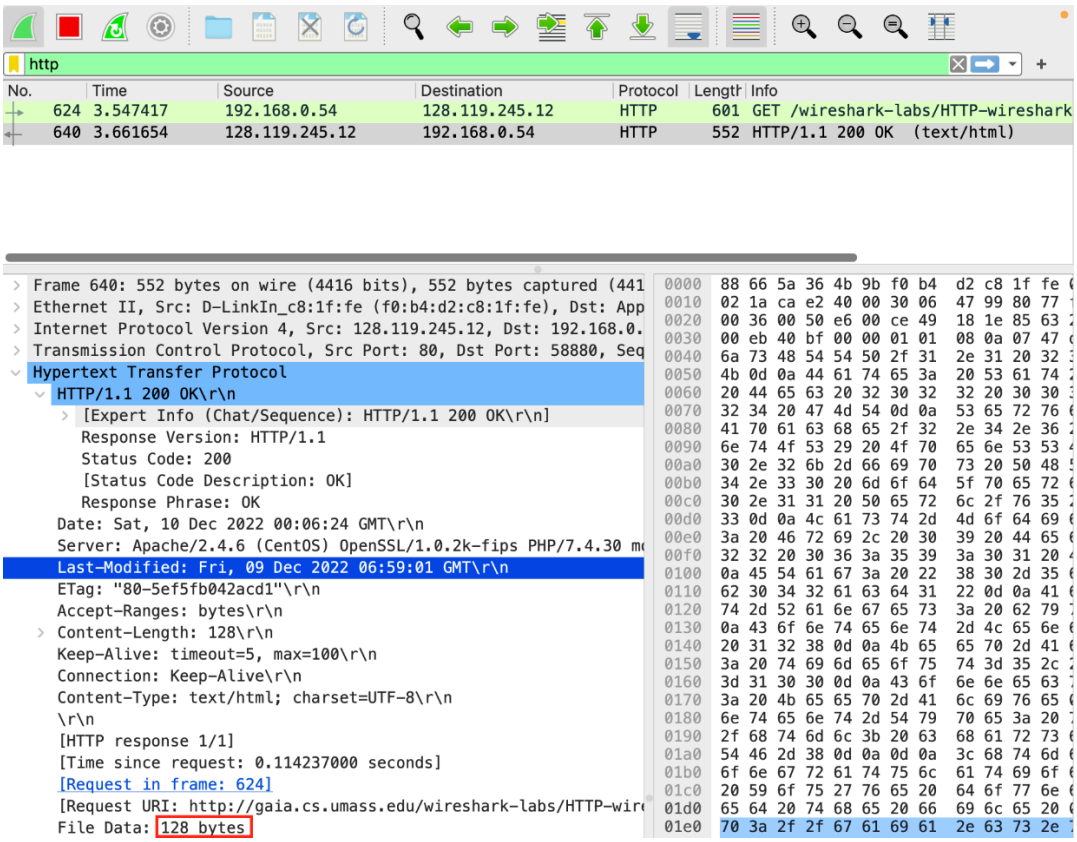
Kadangi norimas resursas paskiausiai buvo rastas, galime sakyti kad status kode buvo 200 OK.

5. When was the HTML file that you are retrieving last modified at the server?

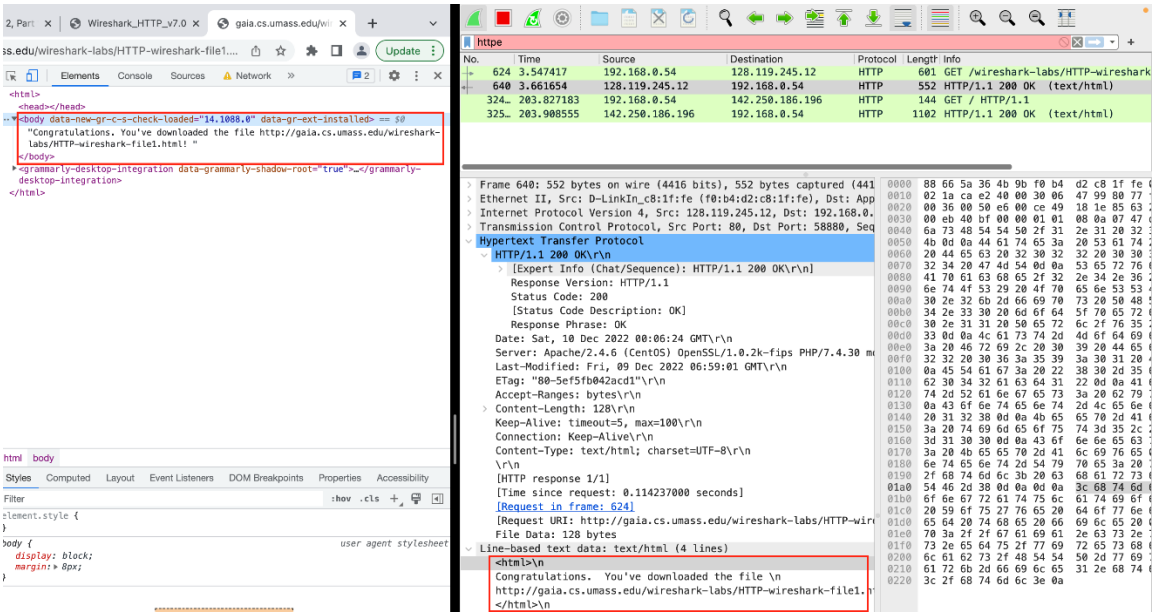


Data sutampa maždaug su puslapio atidarymo data (čia taip specialiai buvo padaryta).

6. How many bytes of content are being returned to your browser?



7. By inspecting the raw data in the packet content window, do you see any headers within the data that are not displayed in the packet-listing window? If so, name one.

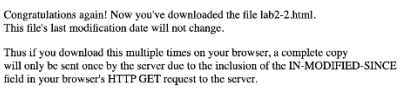


Galime pastebėti, kad originaliame HTML faile yra papildomos <*body*> ir <*head*> žymės.

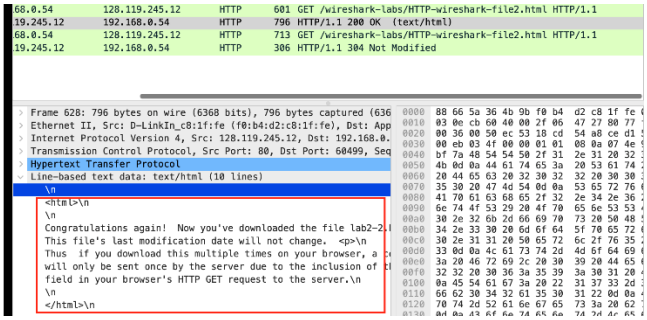
8. Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an “IF-MODIFIED-SINCE” line in the HTTP GET?

Pirmame GET requeste nieko nematome, kadangi mūsų cache buvo išvalytas ir naršyklė „pirmą“ kartą atsisiuntė šitą HTML dokumentą.

9. Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell?

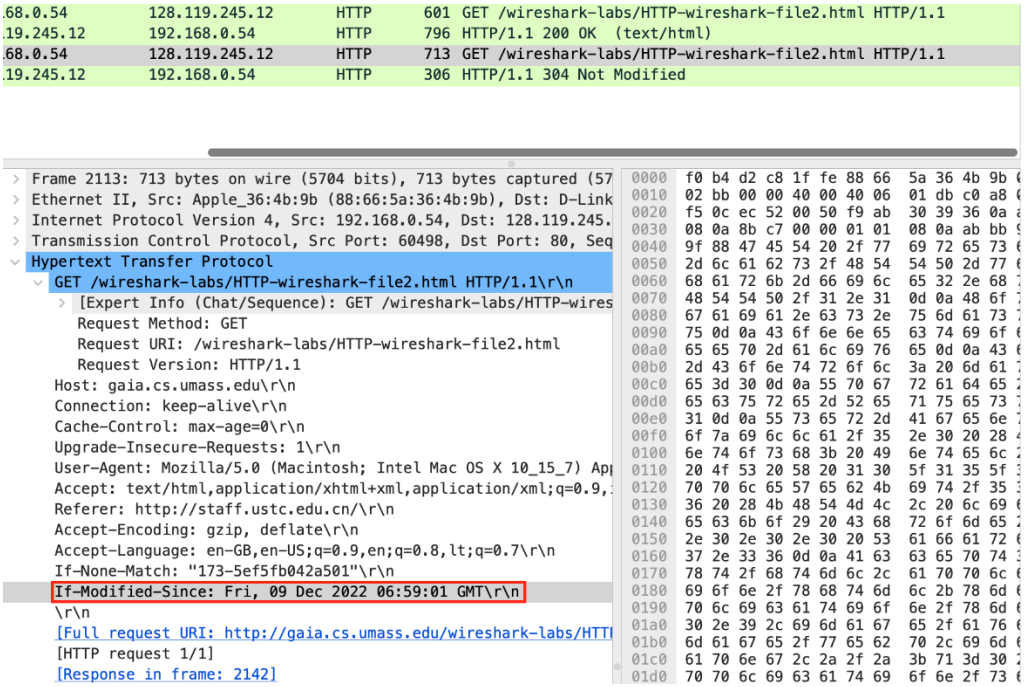

Congratulations again! Now you've downloaded the file lab2-2.html.
This file's last modification date will not change.

Thus if you download this multiple times on your browser, a complete copy
will only be sent once by the server due to the inclusion of the IN-MODIFIED-SINCE
field in your browser's HTTP GET request to the server.



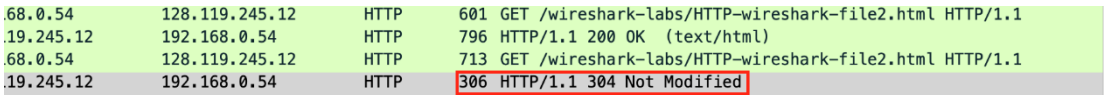
Taip, serveris atgal gražino visą informaciją, kuri sutampa su naršyklėje rodomu tekstu.

10. Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an “IF-MODIFIED-SINCE:” line in the HTTP GET? If so, what information follows the “IF-MODIFIED-SINCE:” header?



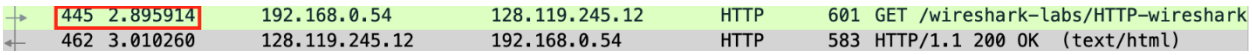
Antrą karta siunčiant GET requestą jau yra matomas „IF-MODIFIED-SINCE“ header‘is. Juo yra pateikiamas laikas kuomet paskutinį kartą buvo atsisiųstas HTML dokumentas. Jeigu dokumentas nepakitęs nuo to laiko, serveris nesiųs nieko atgal.

11. What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Did the server explicitly return the contents of the file? Explain.



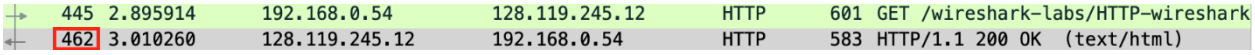
Kadangi failas nepakito, tai serveris mum duomenų pilnų negražino ir todėl naršyklė pasiėmė šį resursą iš cache.

12. How many HTTP GET request messages did your browser send? Which packet number in the trace contains the GET message for the Bill or Rights?

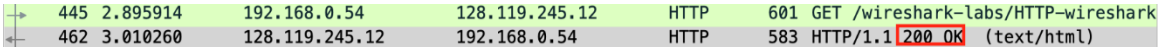


Buvo išsiųstas 1 HTTP GET requestas, jo numeris yra 445.

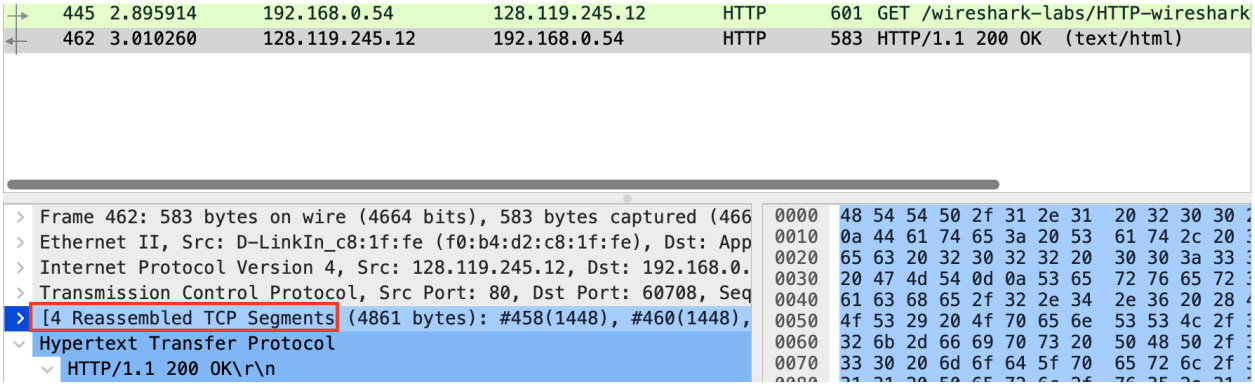
13. Which packet number in the trace contains the status code and phrase associated with the response to the HTTP GET request?



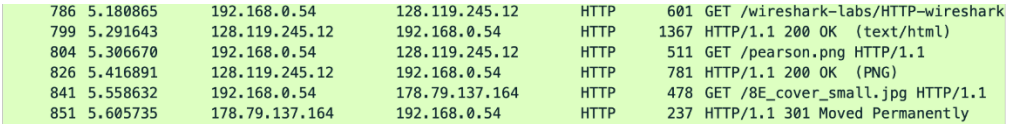
14. What is the status code and phrase in the response?



15. How many data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights?



16. How many HTTP GET request messages did your browser send? To which Internet addresses were these GET requests sent?

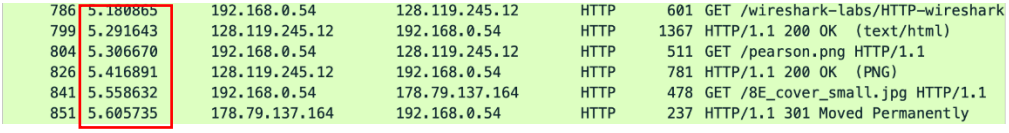


786	5.180865	192.168.0.54	128.119.245.12	HTTP	601	GET /wireshark-labs/HTTP-wireshark
799	5.291643	128.119.245.12	192.168.0.54	HTTP	1367	HTTP/1.1 200 OK (text/html)
804	5.306670	192.168.0.54	128.119.245.12	HTTP	511	GET /pearson.png HTTP/1.1
826	5.416891	128.119.245.12	192.168.0.54	HTTP	781	HTTP/1.1 200 OK (PNG)
841	5.558632	192.168.0.54	178.79.137.164	HTTP	478	GET /8E_cover_small.jpg HTTP/1.1
851	5.605735	178.79.137.164	192.168.0.54	HTTP	237	HTTP/1.1 301 Moved Permanently

Buvo siųsti 3 GET request'ai. Pirmi du kreipėsi į tą patį puslapio adresą (128.119.245.12), kol paskutinis į 178.79.137.164

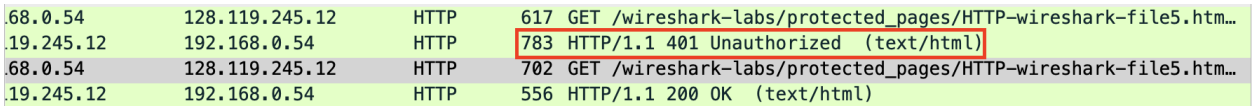
17. Can you tell whether your browser downloaded the two images serially, or whether they were downloaded from the two web sites in parallel? Explain.

Kadangi šių paketų laikas yra skirtingas ir didėja monotoniškai, tai darau prielaida kad juos atsisiuntė vieną po kito.



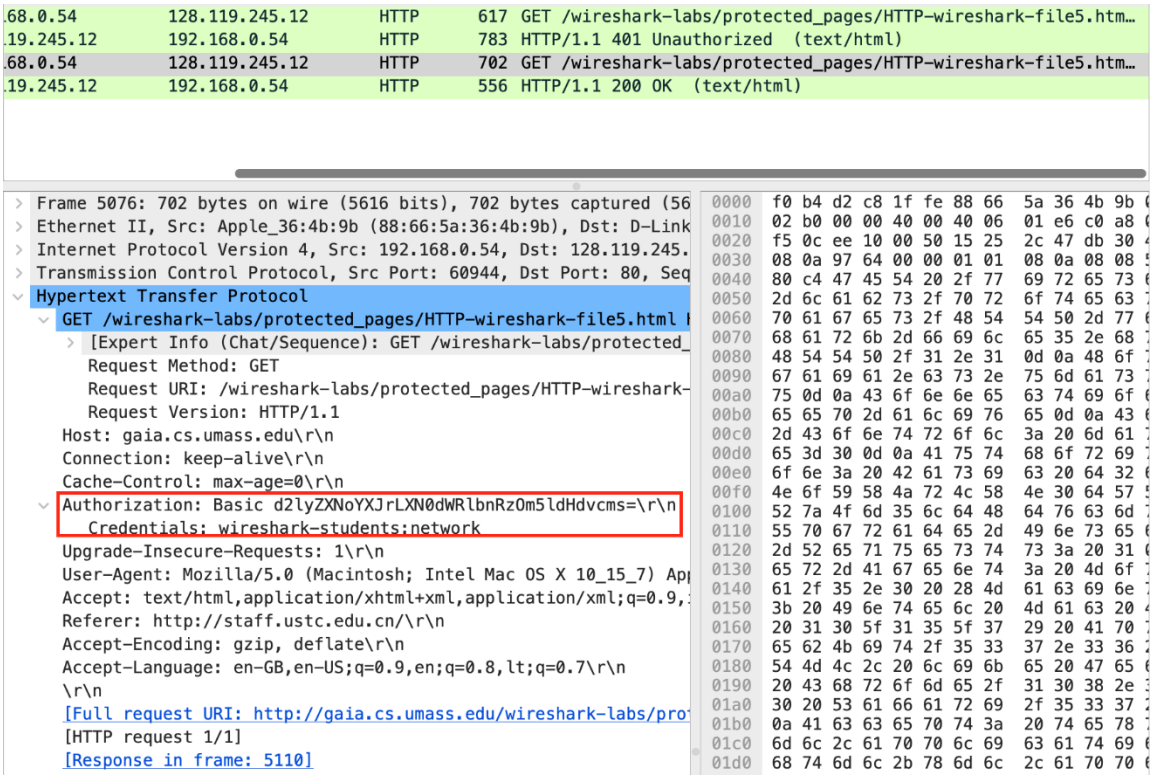
786	5.180865	192.168.0.54	128.119.245.12	HTTP	601	GET /wireshark-labs/HTTP-wireshark
799	5.291643	128.119.245.12	192.168.0.54	HTTP	1367	HTTP/1.1 200 OK (text/html)
804	5.306670	192.168.0.54	128.119.245.12	HTTP	511	GET /pearson.png HTTP/1.1
826	5.416891	128.119.245.12	192.168.0.54	HTTP	781	HTTP/1.1 200 OK (PNG)
841	5.558632	192.168.0.54	178.79.137.164	HTTP	478	GET /8E_cover_small.jpg HTTP/1.1
851	5.605735	178.79.137.164	192.168.0.54	HTTP	237	HTTP/1.1 301 Moved Permanently

18. What is the server's response (status code and phrase) in response to the initial HTTP GET message from your browser?



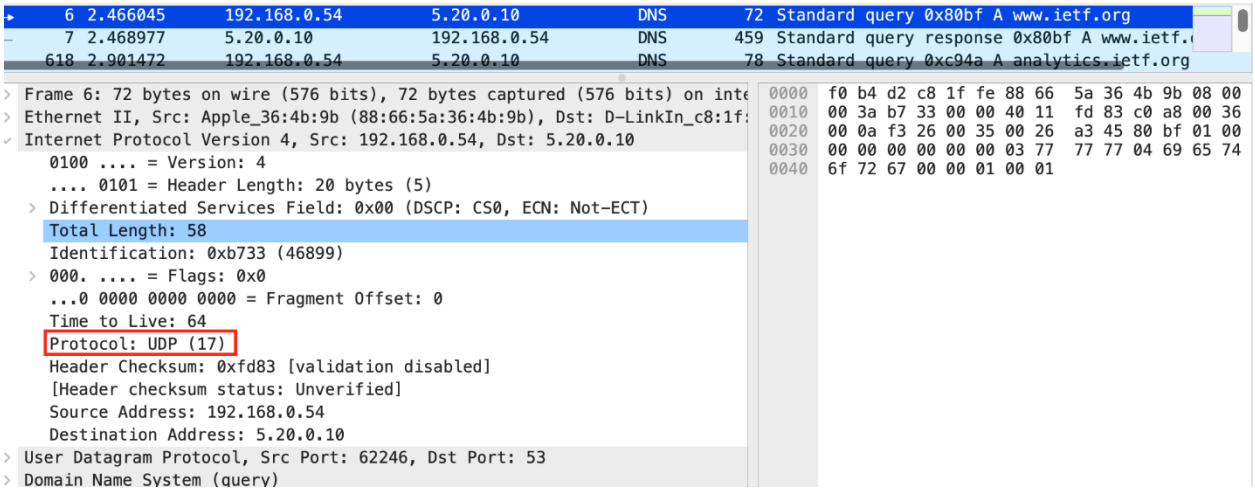
68.0.54	128.119.245.12	HTTP	617 GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.htm…
19.245.12	192.168.0.54	HTTP	783 HTTP/1.1 401 Unauthorized (text/html)
68.0.54	128.119.245.12	HTTP	702 GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.htm…
19.245.12	192.168.0.54	HTTP	556 HTTP/1.1 200 OK (text/html)

19. When your browser's sends the HTTP GET message for the second time, what new field is included in the HTTP GET message?

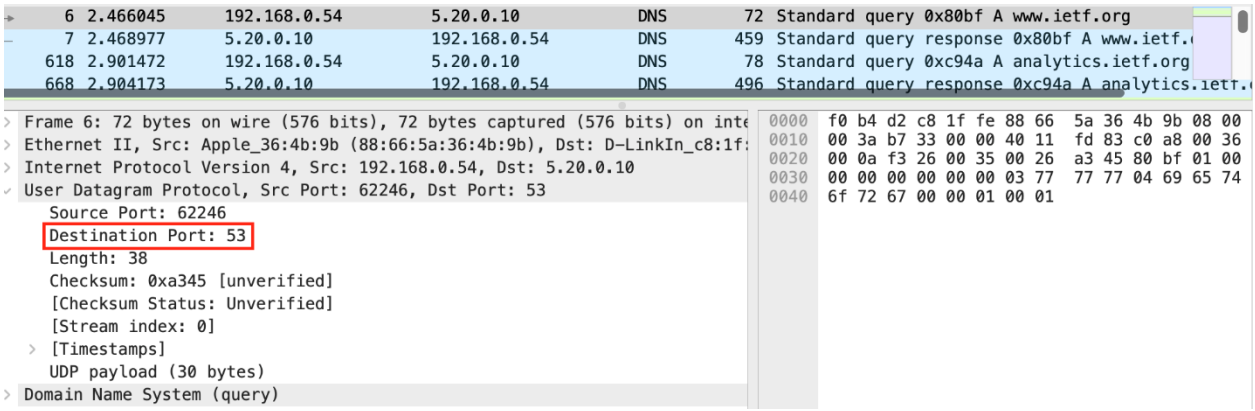


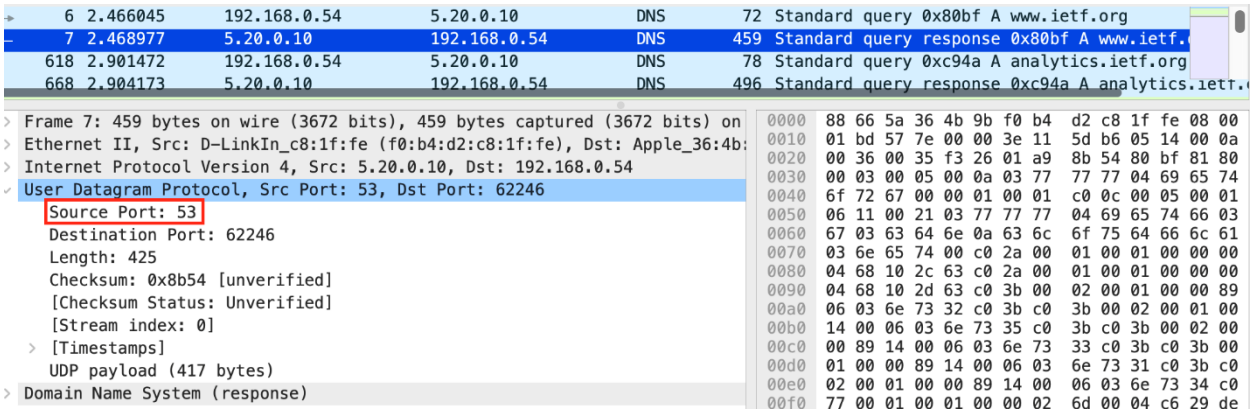
DNS paketų filtravimas

4. Locate the DNS query and response messages. Are they sent over UDP or TCP?

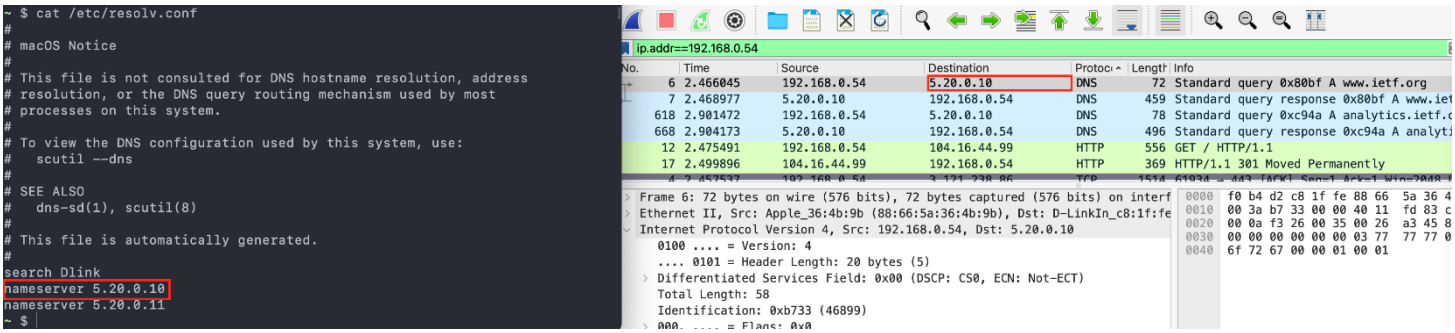


5. What is the destination port for the DNS query message? What is the source port of DNS response message?

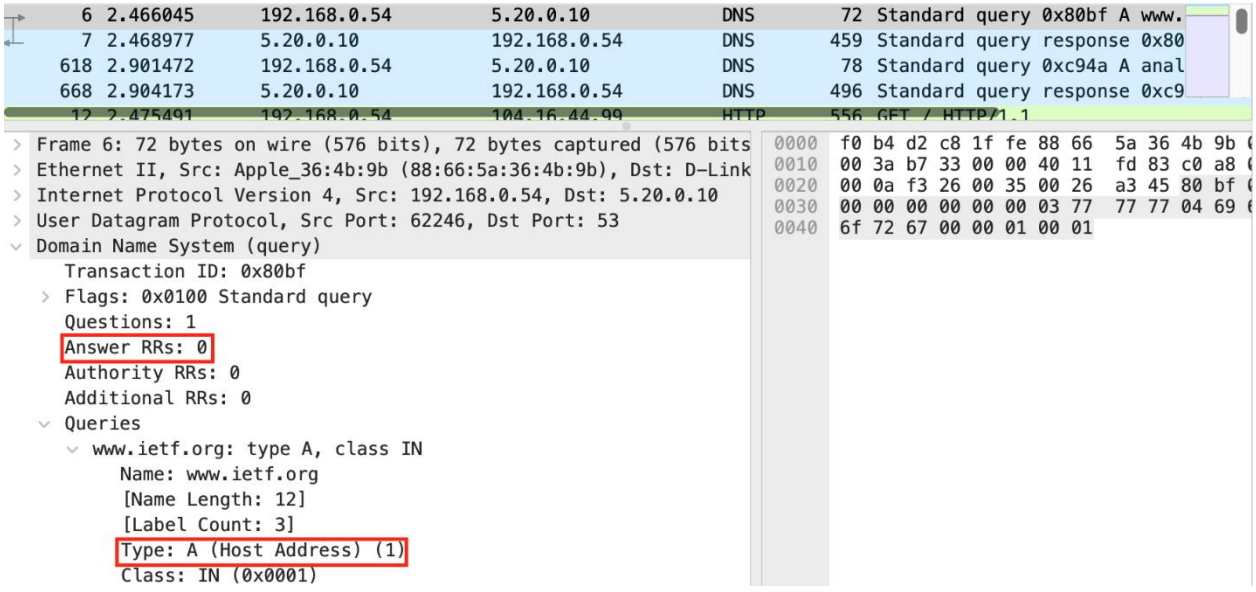




6. To what IP address is the DNS query message sent? Use ipconfig to determine the IP address of your local DNS server. Are these two IP addresses the same?

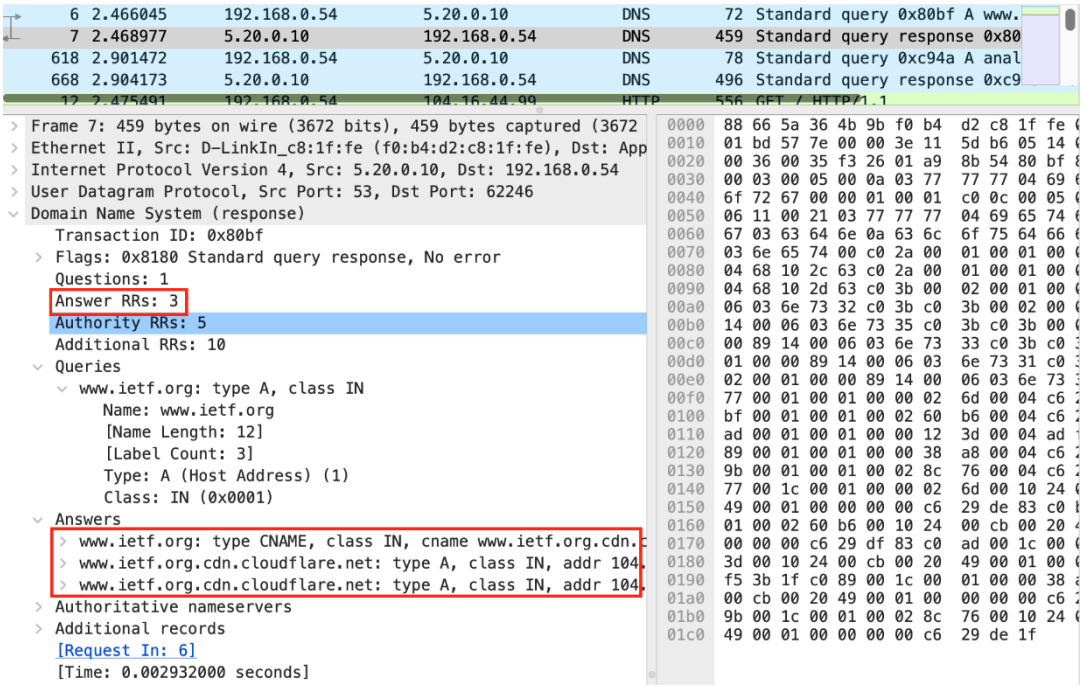


7. Examine the DNS query message. What “Type” of DNS query is it? Does the query message contain any “answers”?



Čia yra „A“ tipo užklausa. Negražintas nei vienas „answer“

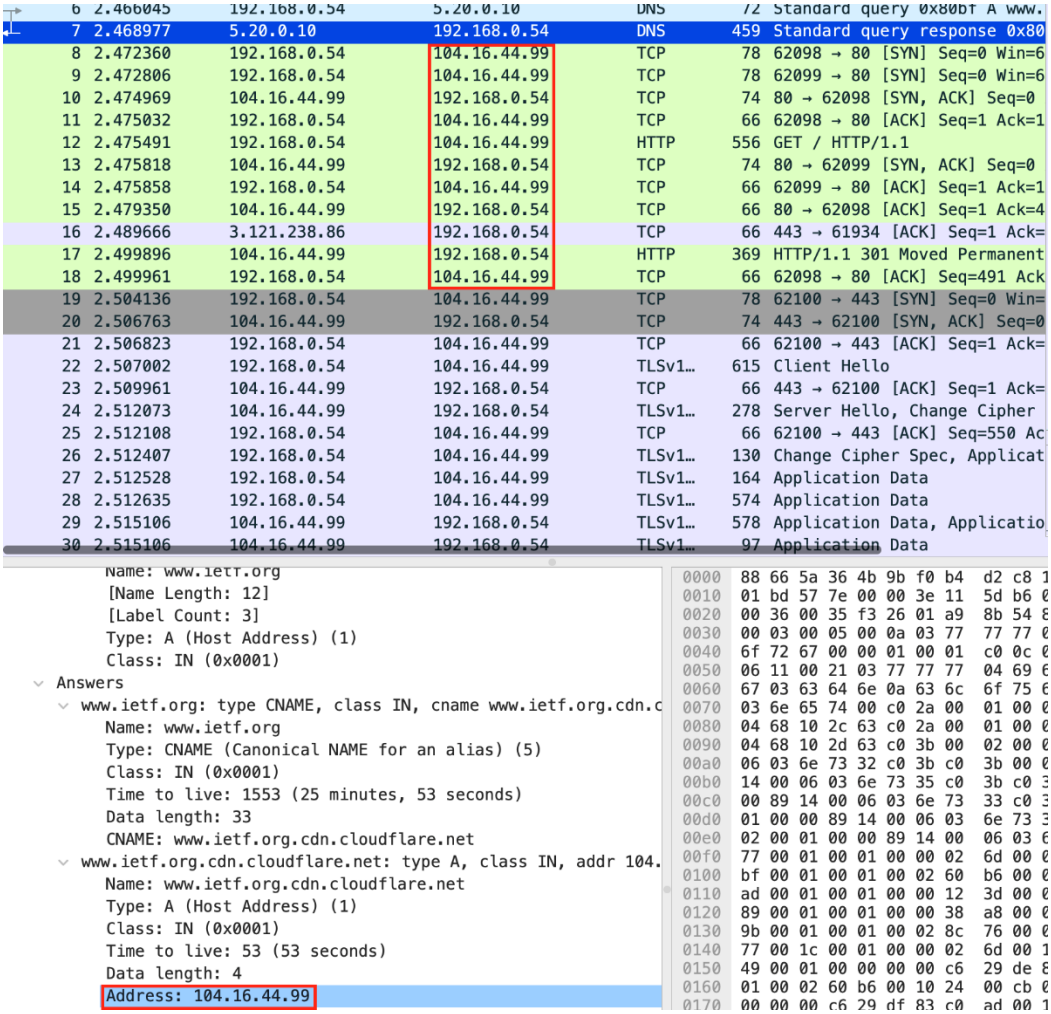
8. Examine the DNS response message. How many “answers” are provided? What do each of these answers contain?



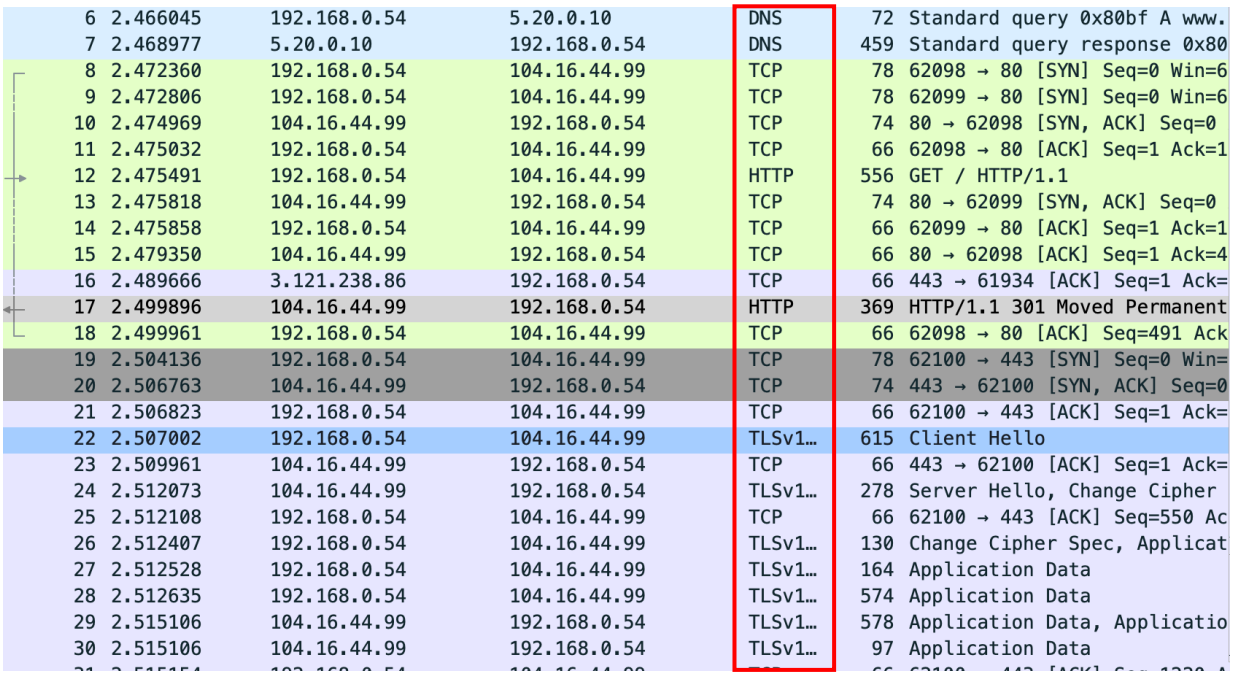
Kiekviename atsakyme yra gražinama:

- Vardas, pagal kūrį yra kuriama DNS užklausa
- DNS įrašo tipas
- TTL – time to live
- Duomenų ilgis
- IP adresas (jeigu tankamas DNS įrašo tipas)

9. Consider the subsequent TCP SYN packet sent by your host. Does the destination IP address of the SYN packet correspond to any of the IP addresses provided in the DNS response message?



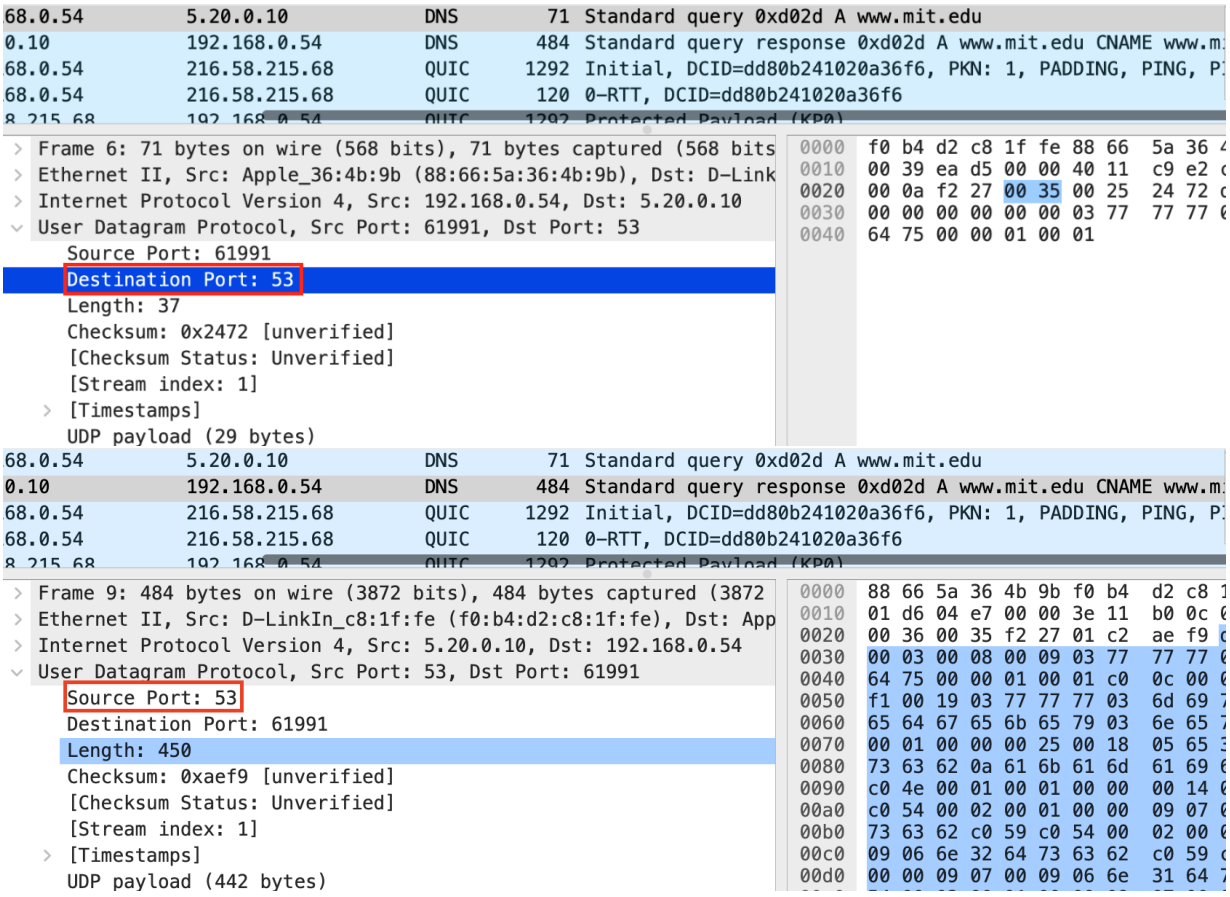
10. This web page contains images. Before retrieving each image, does your host issue new DNS queries?



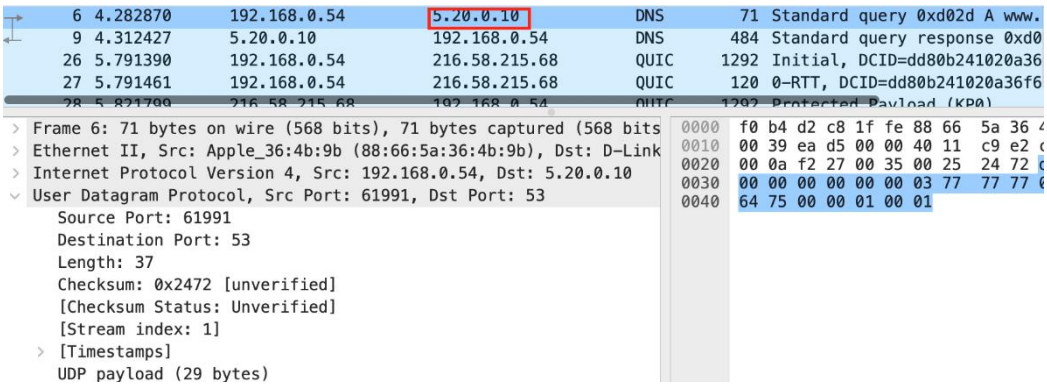
6	2.466045	192.168.0.54	5.20.0.10	DNS	72	Standard query 0x80bf A www.
7	2.468977	5.20.0.10	192.168.0.54	DNS	459	Standard query response 0x80
8	2.472360	192.168.0.54	104.16.44.99	TCP	78	62098 → 80 [SYN] Seq=0 Win=6
9	2.472806	192.168.0.54	104.16.44.99	TCP	78	62099 → 80 [SYN] Seq=0 Win=6
10	2.474969	104.16.44.99	192.168.0.54	TCP	74	80 → 62098 [SYN, ACK] Seq=0
11	2.475032	192.168.0.54	104.16.44.99	TCP	66	62098 → 80 [ACK] Seq=1 Ack=1
12	2.475491	192.168.0.54	104.16.44.99	HTTP	556	GET / HTTP/1.1
13	2.475818	104.16.44.99	192.168.0.54	TCP	74	80 → 62099 [SYN, ACK] Seq=0
14	2.475858	192.168.0.54	104.16.44.99	TCP	66	62099 → 80 [ACK] Seq=1 Ack=1
15	2.479350	104.16.44.99	192.168.0.54	TCP	66	80 → 62098 [ACK] Seq=1 Ack=4
16	2.489666	3.121.238.86	192.168.0.54	TCP	66	443 → 61934 [ACK] Seq=1 Ack=
17	2.499896	104.16.44.99	192.168.0.54	HTTP	369	HTTP/1.1 301 Moved Permanent
18	2.499961	192.168.0.54	104.16.44.99	TCP	66	62098 → 80 [ACK] Seq=491 Ack
19	2.504136	192.168.0.54	104.16.44.99	TCP	78	62100 → 443 [SYN] Seq=0 Win=
20	2.506763	104.16.44.99	192.168.0.54	TCP	74	443 → 62100 [SYN, ACK] Seq=0
21	2.506823	192.168.0.54	104.16.44.99	TCP	66	62100 → 443 [ACK] Seq=1 Ack=
22	2.507002	192.168.0.54	104.16.44.99	TLSv1…	615	Client Hello
23	2.509961	104.16.44.99	192.168.0.54	TCP	66	443 → 62100 [ACK] Seq=1 Ack=
24	2.512073	104.16.44.99	192.168.0.54	TLSv1…	278	Server Hello, Change Cipher
25	2.512108	192.168.0.54	104.16.44.99	TCP	66	62100 → 443 [ACK] Seq=550 Ac
26	2.512407	192.168.0.54	104.16.44.99	TLSv1…	130	Change Cipher Spec, Applicat
27	2.512528	192.168.0.54	104.16.44.99	TLSv1…	164	Application Data
28	2.512635	192.168.0.54	104.16.44.99	TLSv1…	574	Application Data
29	2.515106	104.16.44.99	192.168.0.54	TLSv1…	578	Application Data, Applicatio
30	2.515106	104.16.44.99	192.168.0.54	TLSv1…	97	Application Data

Iš paviršutiniško paketų vaizdo tas nebuvo pastebėta.

11. What is the destination port for the DNS query message? What is the source port of DNS response message?

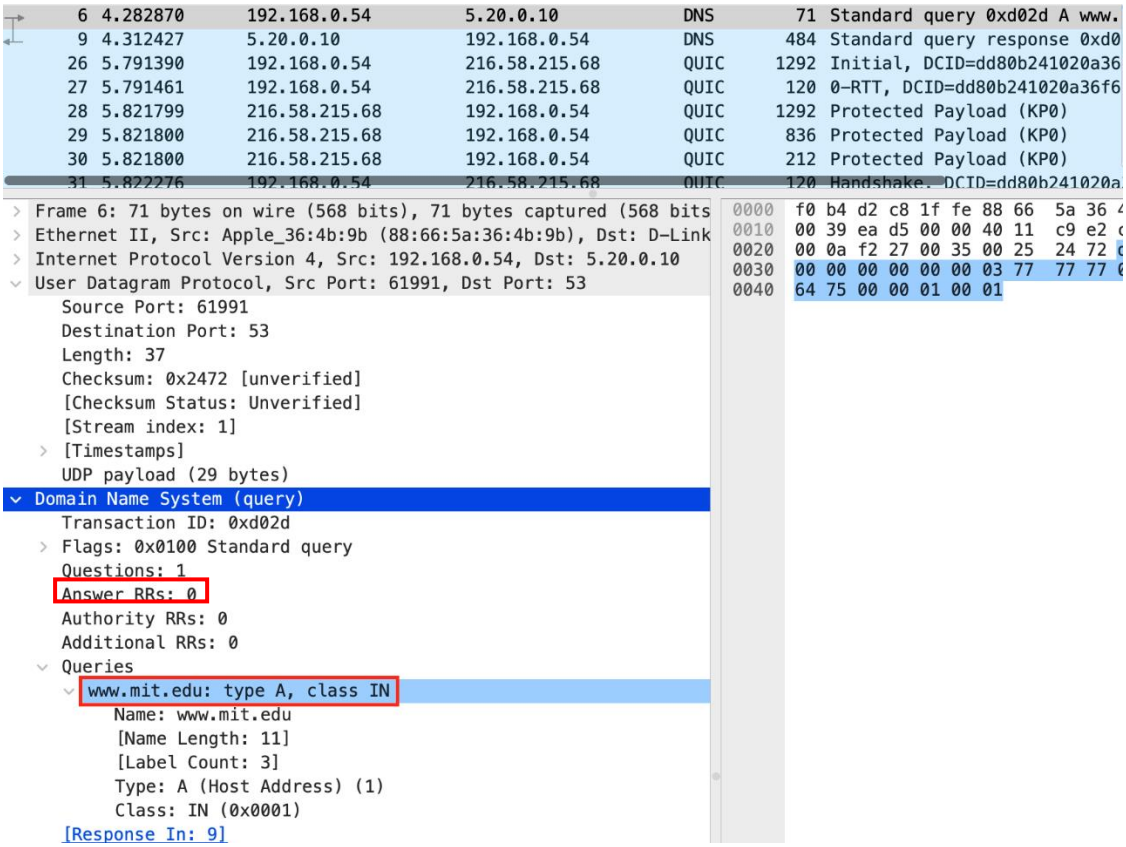


12. To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server?

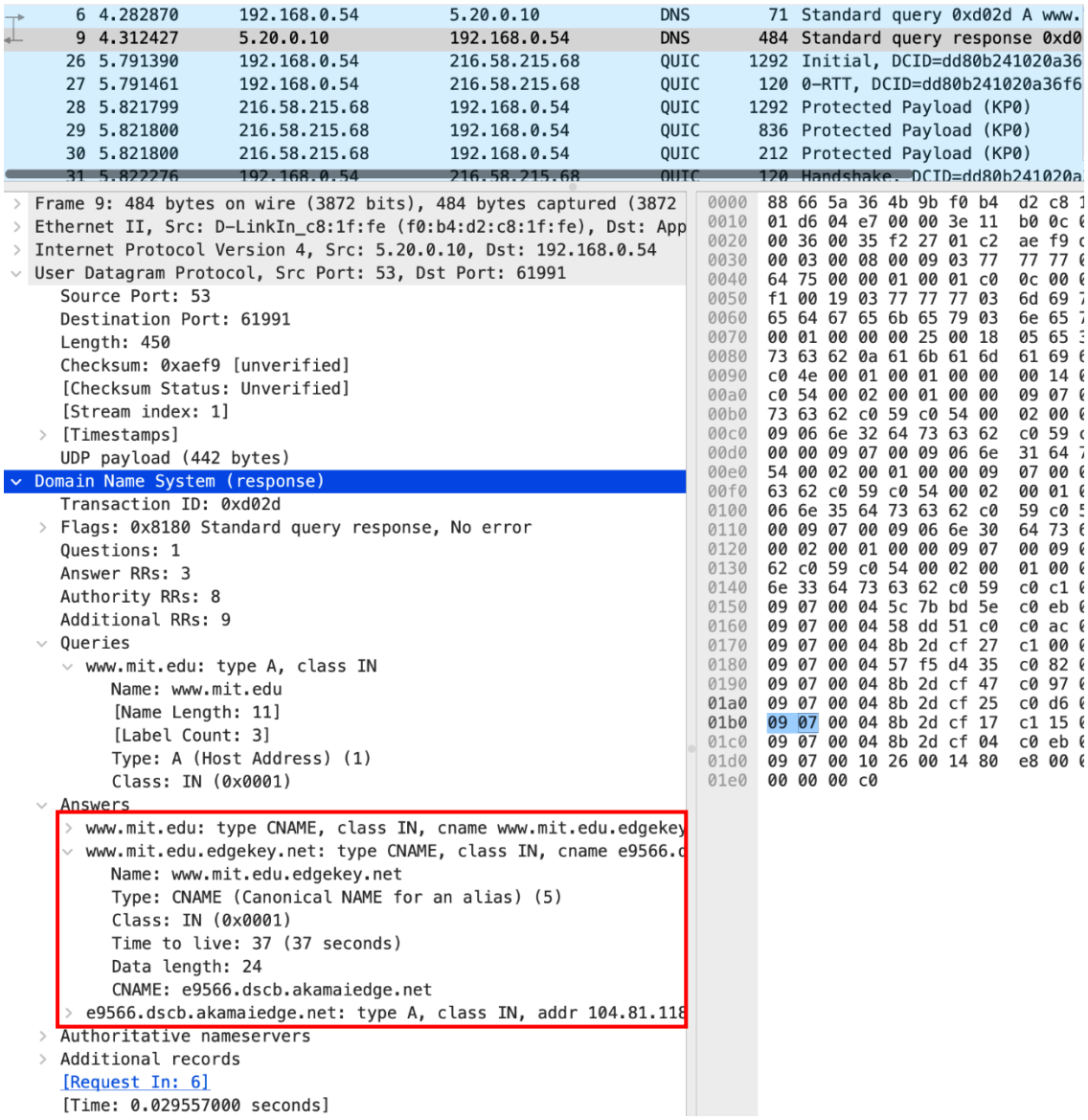


Taip, čia yra mano lokalus DNS serverio adresas (žiūrėti 6 užduotį)

13. Examine the DNS query message. What “Type” of DNS query is it? Does the query message contain any “answers”?



14. Examine the DNS response message. How many “answers” are provided? What do each of these answers contain?

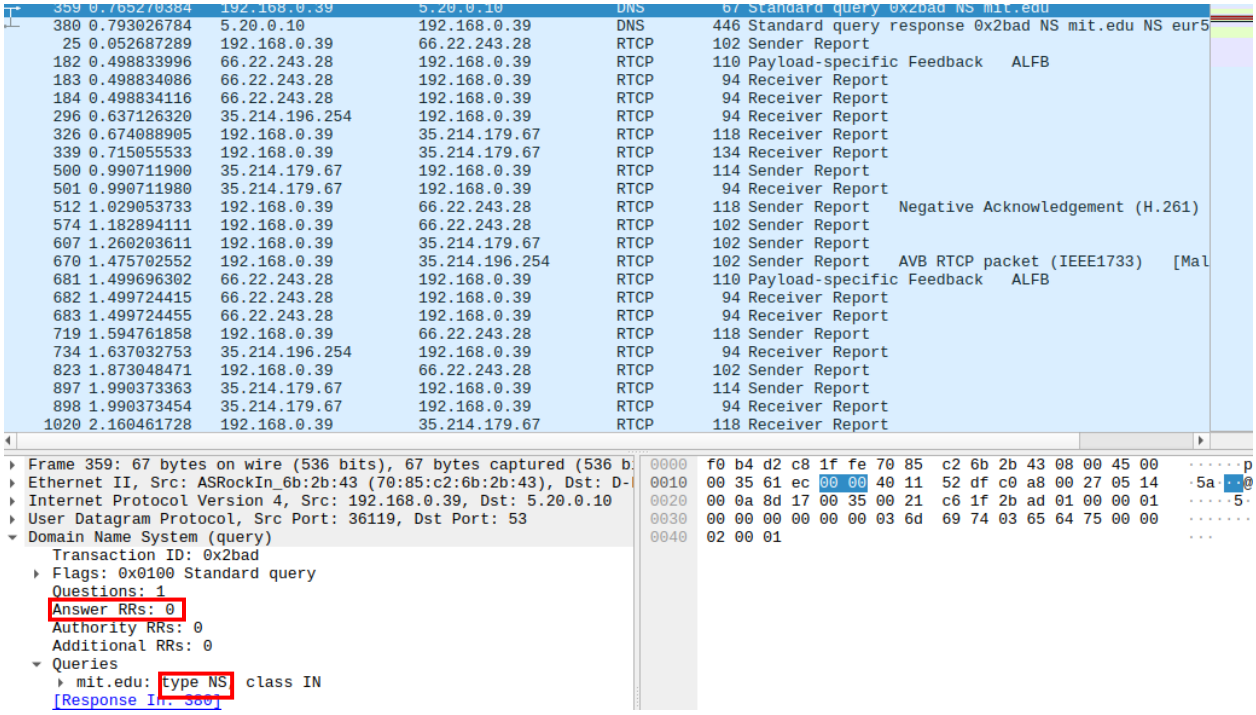


Gauti 3 atsakymai. Žiūrėti 8 užduotį, nes joje pateikiama informacija yra identiška šiam atasakymui.

16. To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server?

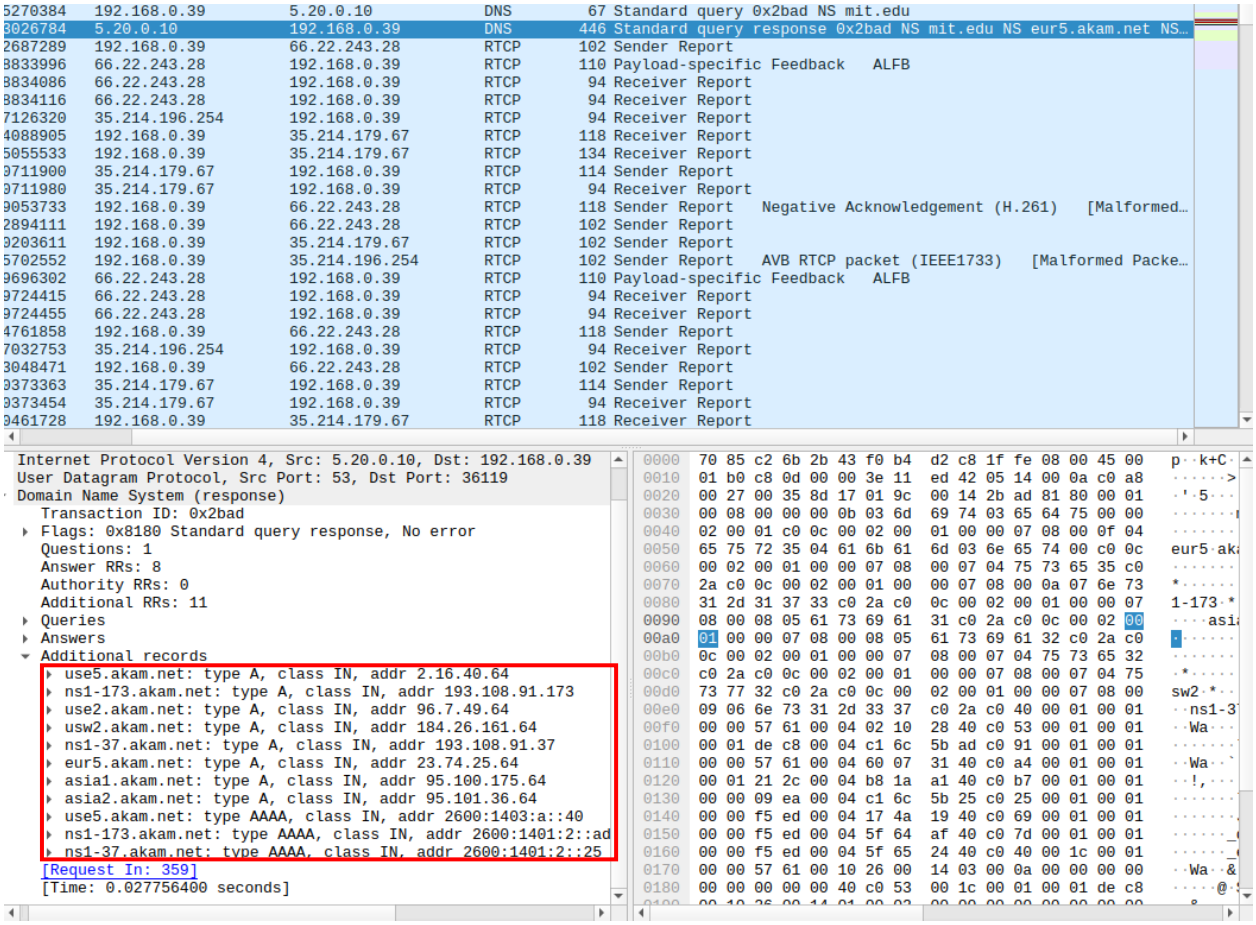
Taip, čia yra mano lokalaus DNS serverio adresas (žiūrėti 6 užduotį)

17. Examine the DNS query message. What “Type” of DNS query is it? Does the query message contain any “answers”?



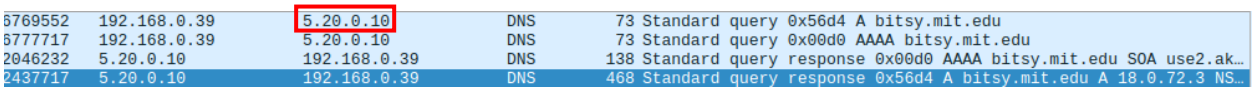
Šį kartą tai yra Name Server DNS užklausa. Tačiau ir vėl gražinama 0 atsakymų.

18. Examine the DNS response message. What MIT nameservers does the response message provide? Does this response message also provide the IP addresses of the MIT namesers?



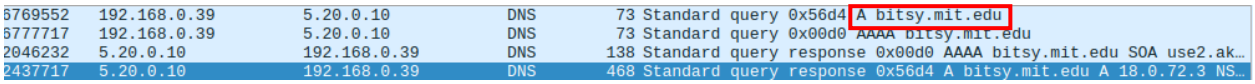
Pateikiami 8 „A“ tipo DNS įrašai. Šalia jų yra pateikiami ir jų adresai.

20. To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server? If not, what does the IP address correspond to?



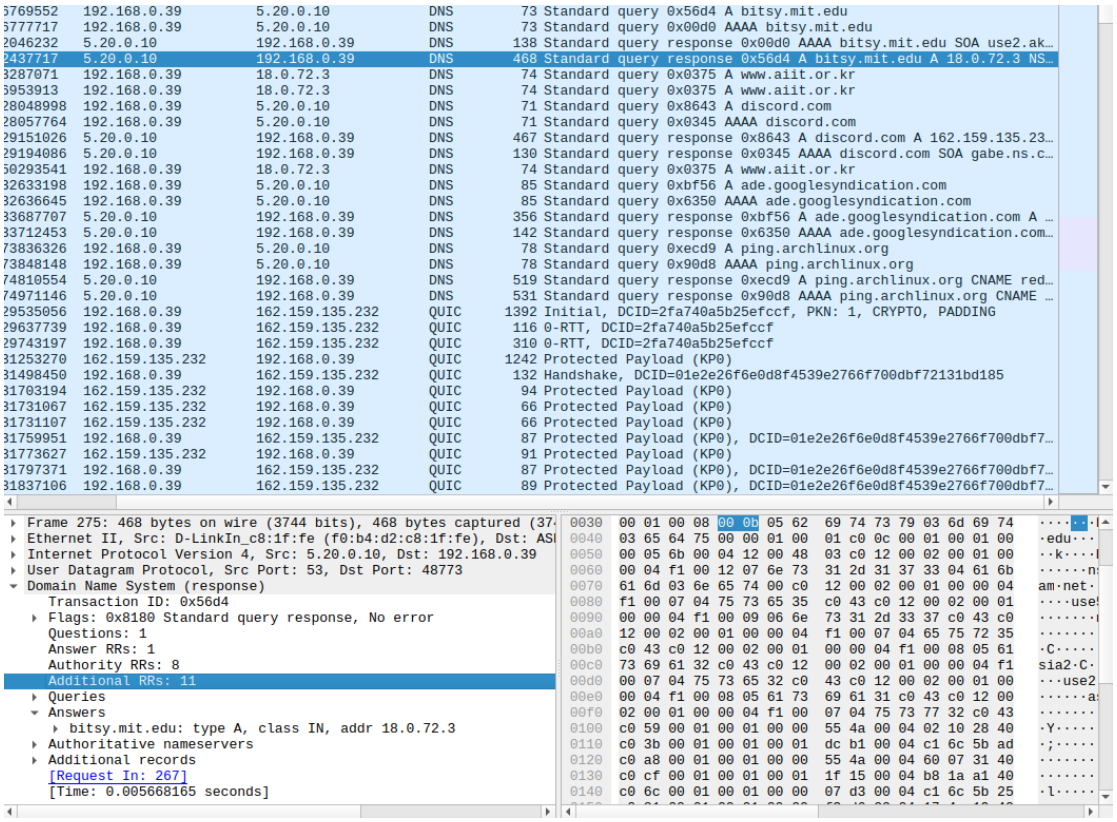
Taip, čia yra mano lokalus DNS serverio adresas (žiūrėti 6 užduotį).

21. Examine the DNS query message. What “Type” of DNS query is it? Does the query message contain any “answers”?



Tipas – A, 0 „atsakymų“.

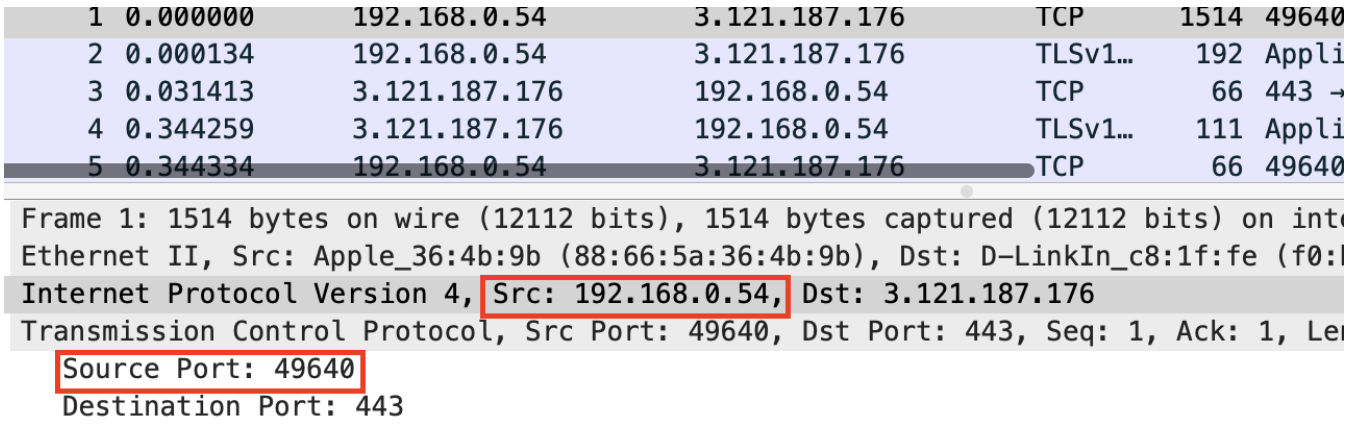
22. Examine the DNS response message. How many “answers” are provided? What does each of these answers contain?



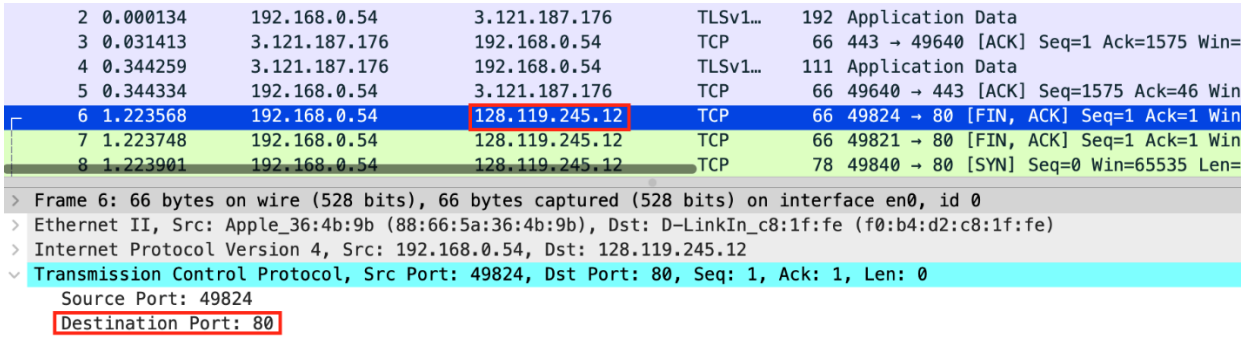
Pateikiamas 1 atsakymas. Informacija identiškai kaip ir 8 uždavinyje.

TCP paketų filtravimas

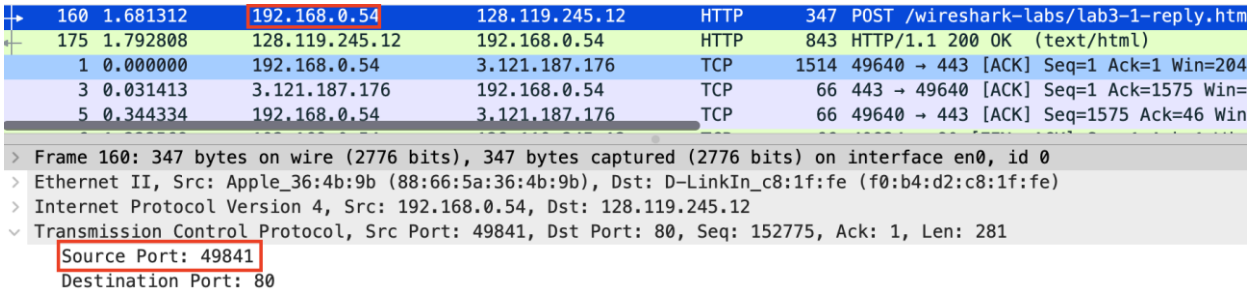
1. What is the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu?



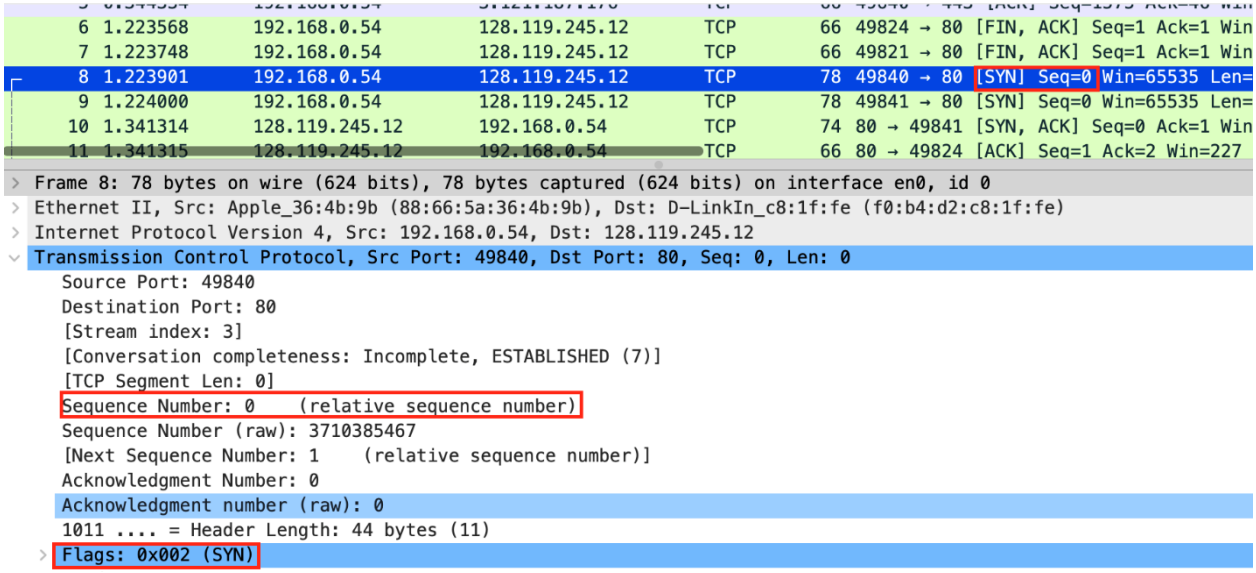
2. What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?



3. What is the IP address and TCP port number used by your client computer (source) to transfer the file to gaia.cs.umass.edu?

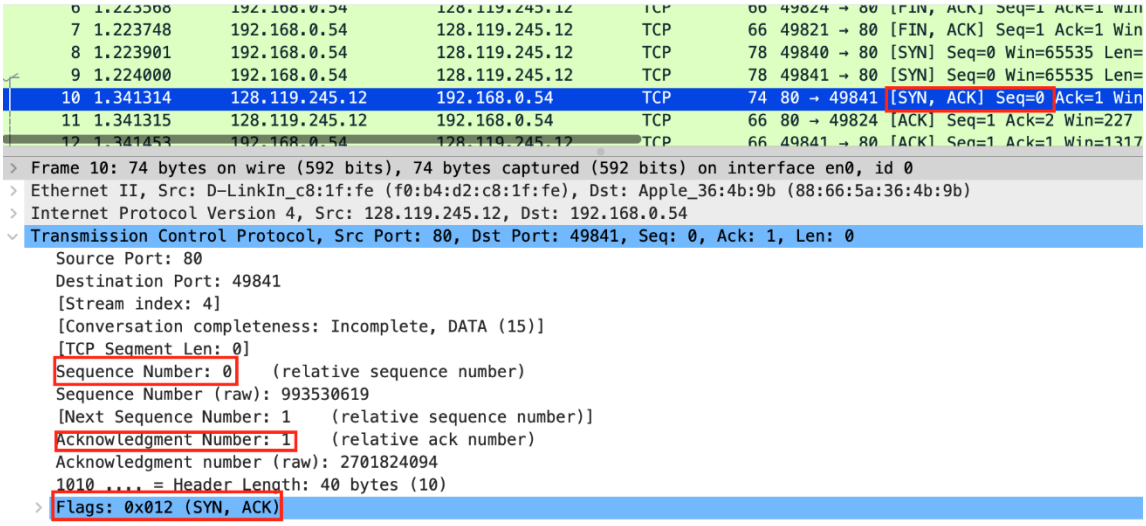


4. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?



Kad čia yra SYN segmentas galime pagal Flag‘us, kuriose yra matoma ši žymė.

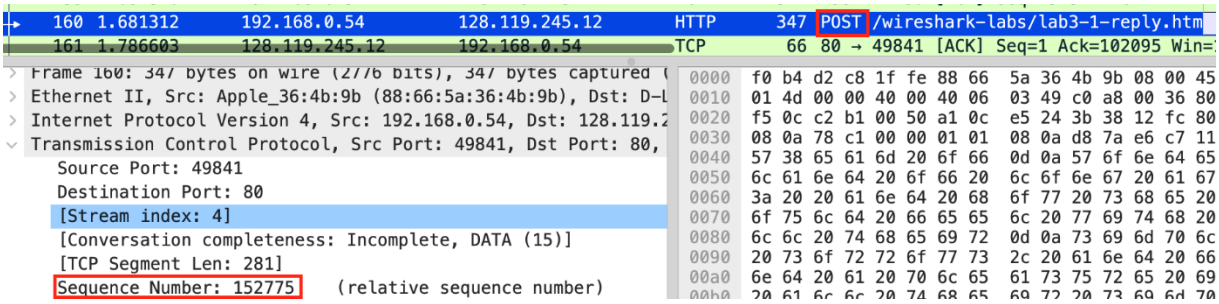
5. What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK segment?



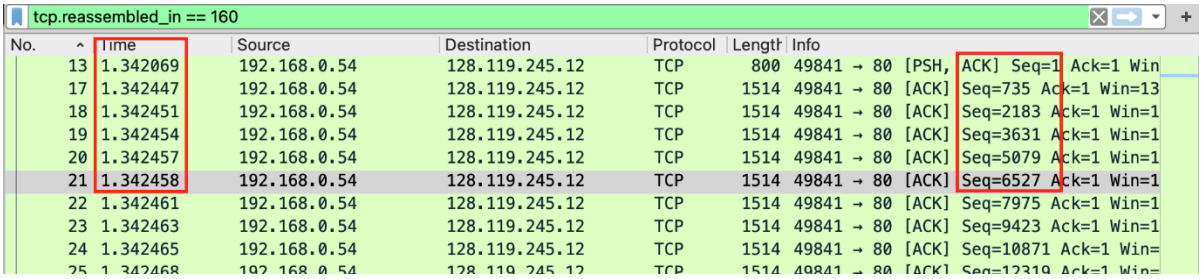
„Aknowledgement“ skaičius pasako iki kokio baito gavėjas yra gavęs informaciją. Jeigu tikėtinas sequence skaičius iš siuntėjo ir aknowledgement skaičius ant gavėjo nesutampa po vieno paketų exchange, tai gali signalizuoti informacijos neatitikimą.

Kad šis segmentas yra SYN ACK, galime matyti vėl gi iš flag‘ų.

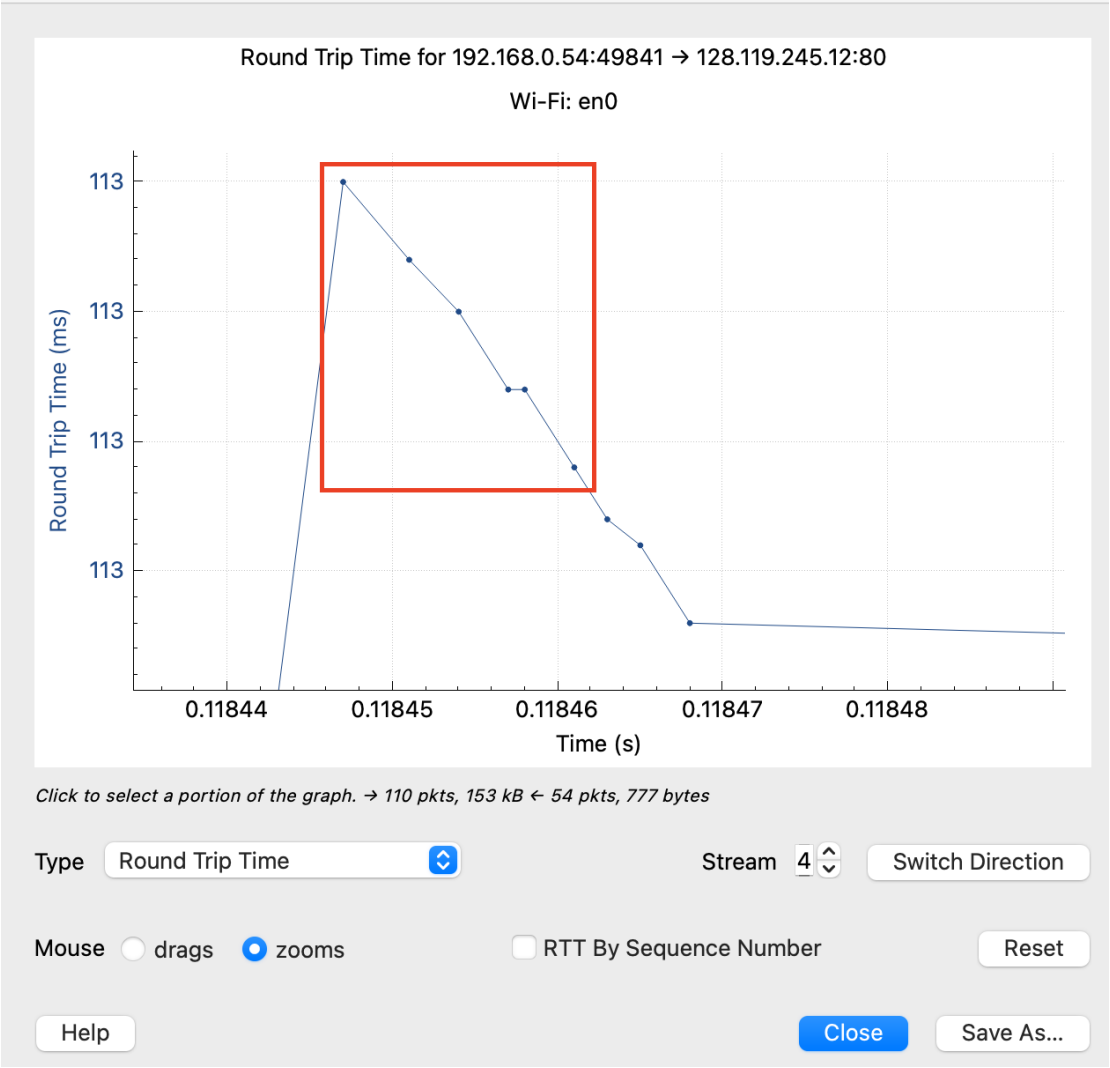
6. What is the sequence number of the TCP segment containing the HTTP POST command? Note that in order to find the POST command, you'll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a "POST" within its DATA field.



7. Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? At what time was each segment sent? When was the ACK for each segment received? Given the difference between when each TCP segment was sent, and when its acknowledgement was received, what is the RTT value for each of the six segments? What is the EstimatedRTT value (see Section 3.5.3, page 242 in text) after the receipt of each ACK?



Sequence number	Sent Time	Received time
1	1.342090	1.453099
735	1.342447	1.455372
2183	1.342451	1.455373
3631	1.342454	1.455374
5079	1.342457	1.455374
6527	1.342458	1.455375

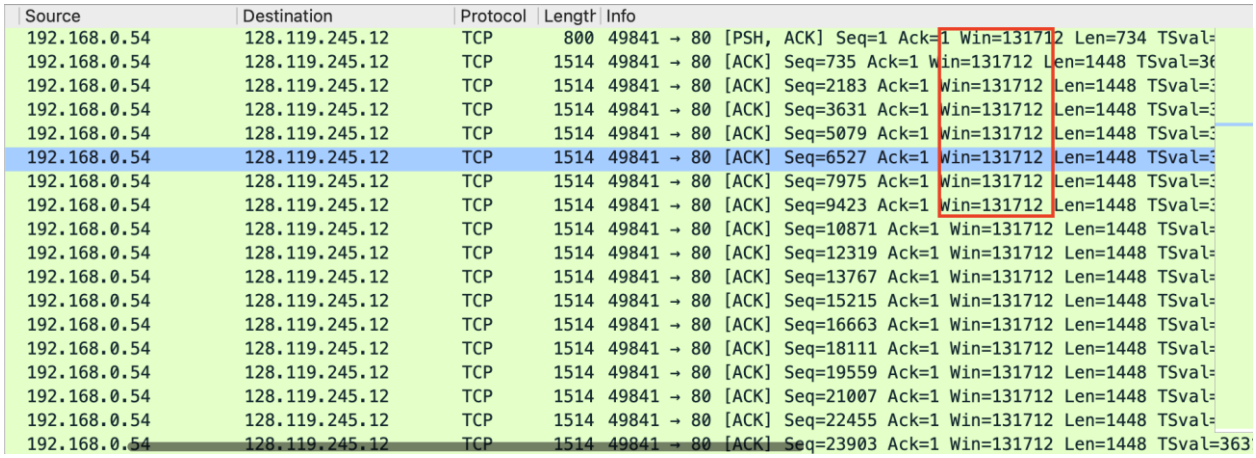


8. What is the length of each of the first six TCP segments?

13	1.342069	192.168.0.54	128.119.245.12	TCP	800	49841 → 80 [PSH, ACK] Seq=1 Ack=1 Win
17	1.342447	192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=735 Ack=1 Win=13
18	1.342451	192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=2183 Ack=1 Win=1
19	1.342454	192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=3631 Ack=1 Win=1
20	1.342457	192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=5079 Ack=1 Win=1
21	1.342458	192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=6527 Ack=1 Win=1
22	1.342461	192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=7975 Ack=1 Win=1
23	1.342463	192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=9423 Ack=1 Win=1
24	1.342465	192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=10871 Ack=1 Win=
25	1.342468	192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=12319 Ack=1 Win=
27	1.453210	192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=13767 Ack=1 Win=
37	1.455499	192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=15215 Ack=1 Win=
38	1.455504	192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=16663 Ack=1 Win=
39	1.455577	192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=18111 Ack=1 Win=
40	1.455579	192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=19559 Ack=1 Win=
41	1.455677	192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=21007 Ack=1 Win=
42	1.455683	192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=22455 Ack=1 Win=
43	1.455739	192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=23903 Ack=1 Win=1317

1514, išskyrus pradinį segmentą - 800.

9. What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?

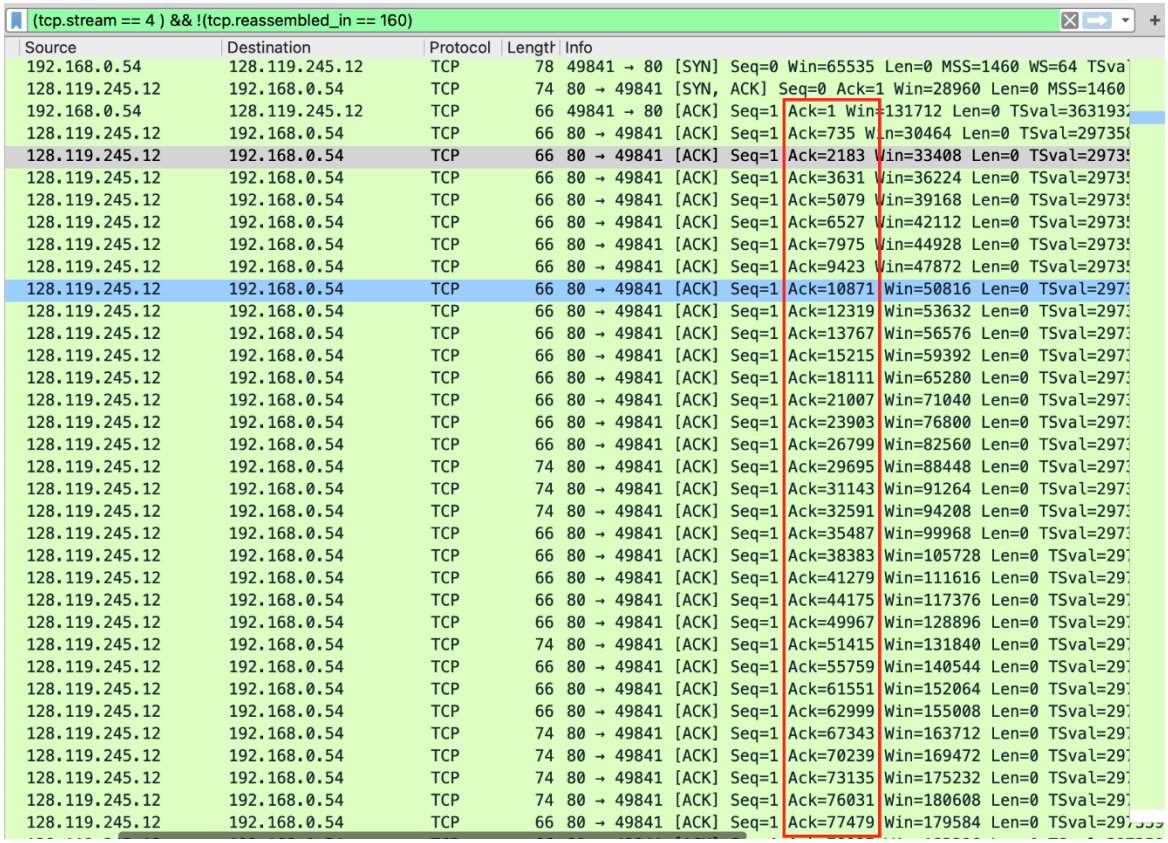


Source	Destination	Protocol	Length	Info
192.168.0.54	128.119.245.12	TCP	800	49841 → 80 [PSH, ACK] Seq=1 Ack=1 Win=131712 Len=734 TSval=
192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=735 Ack=1 Win=131712 Len=1448 TSval=36
192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=2183 Ack=1 Win=131712 Len=1448 TSval=3
192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=3631 Ack=1 Win=131712 Len=1448 TSval=3
192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=5079 Ack=1 Win=131712 Len=1448 TSval=3
192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=6527 Ack=1 Win=131712 Len=1448 TSval=3
192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=7975 Ack=1 Win=131712 Len=1448 TSval=3
192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=9423 Ack=1 Win=131712 Len=1448 TSval=3
192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=10871 Ack=1 Win=131712 Len=1448 TSval=
192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=12319 Ack=1 Win=131712 Len=1448 TSval=
192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=13767 Ack=1 Win=131712 Len=1448 TSval=
192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=15215 Ack=1 Win=131712 Len=1448 TSval=
192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=16663 Ack=1 Win=131712 Len=1448 TSval=
192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=18111 Ack=1 Win=131712 Len=1448 TSval=
192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=19559 Ack=1 Win=131712 Len=1448 TSval=
192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=21007 Ack=1 Win=131712 Len=1448 TSval=
192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=22455 Ack=1 Win=131712 Len=1448 TSval=
192.168.0.54	128.119.245.12	TCP	1514	49841 → 80 [ACK] Seq=23903 Ack=1 Win=131712 Len=1448 TSval=363

Window size – 131712, ir jis tikrai nelimituoja siuntimo, žinant, kad vidutinis segmentas buvo vos apie 1514 baitų.

10. Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?

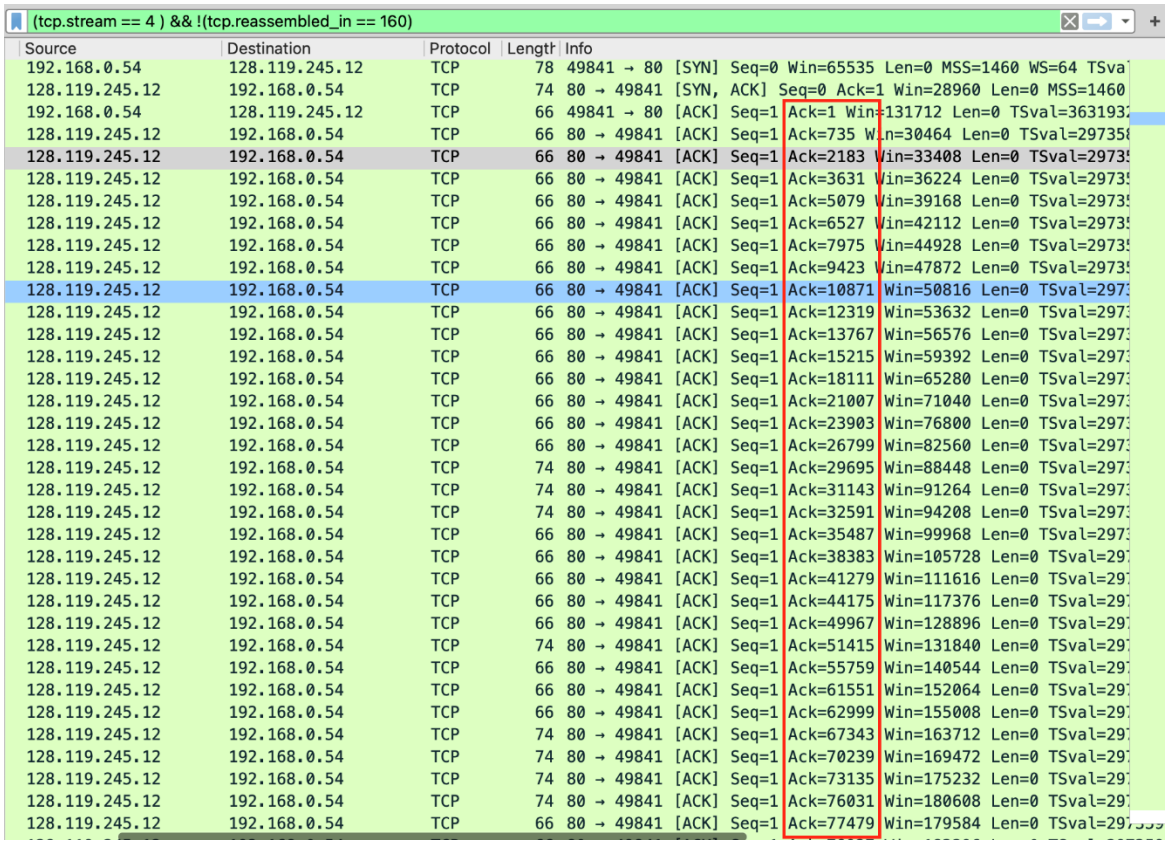
Kadangi iš atsakų „aknowledgement“ skaičiai nesikartodavo, tai galime sakyti, kad segmentai nebuvo persiųsti iš naujo.



(tcp.stream == 4) && !(tcp.reassembled_in == 160)

Source	Destination	Protocol	Length	Info
192.168.0.54	128.119.245.12	TCP	78	49841 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSva
128.119.245.12	192.168.0.54	TCP	74	80 → 49841 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460
192.168.0.54	128.119.245.12	TCP	66	49841 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=363193
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=735 Win=30464 Len=0 TSval=29735
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=2183 Win=33408 Len=0 TSval=2973
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=3631 Win=36224 Len=0 TSval=2973
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=5079 Win=39168 Len=0 TSval=2973
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=6527 Win=42112 Len=0 TSval=2973
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=7975 Win=44928 Len=0 TSval=2973
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=9423 Win=47872 Len=0 TSval=2973
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=10871 Win=50816 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=12319 Win=53632 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=13767 Win=56576 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=15215 Win=59392 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=18111 Win=65280 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=21007 Win=71040 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=23903 Win=76800 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=26799 Win=82560 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	74	80 → 49841 [ACK] Seq=1 Ack=29695 Win=88448 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	74	80 → 49841 [ACK] Seq=1 Ack=31143 Win=91264 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	74	80 → 49841 [ACK] Seq=1 Ack=32591 Win=94208 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=35487 Win=99968 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=38383 Win=105728 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=41279 Win=111616 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=44175 Win=117376 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=49967 Win=128896 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	74	80 → 49841 [ACK] Seq=1 Ack=51415 Win=131840 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=55759 Win=140544 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=61551 Win=152064 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=62999 Win=155008 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	74	80 → 49841 [ACK] Seq=1 Ack=67343 Win=163712 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	74	80 → 49841 [ACK] Seq=1 Ack=70239 Win=169472 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	74	80 → 49841 [ACK] Seq=1 Ack=73135 Win=175232 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	74	80 → 49841 [ACK] Seq=1 Ack=76031 Win=180608 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=77479 Win=179584 Len=0 TSval=29

11. How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment (see Table 3.2 on page 250 in the text).

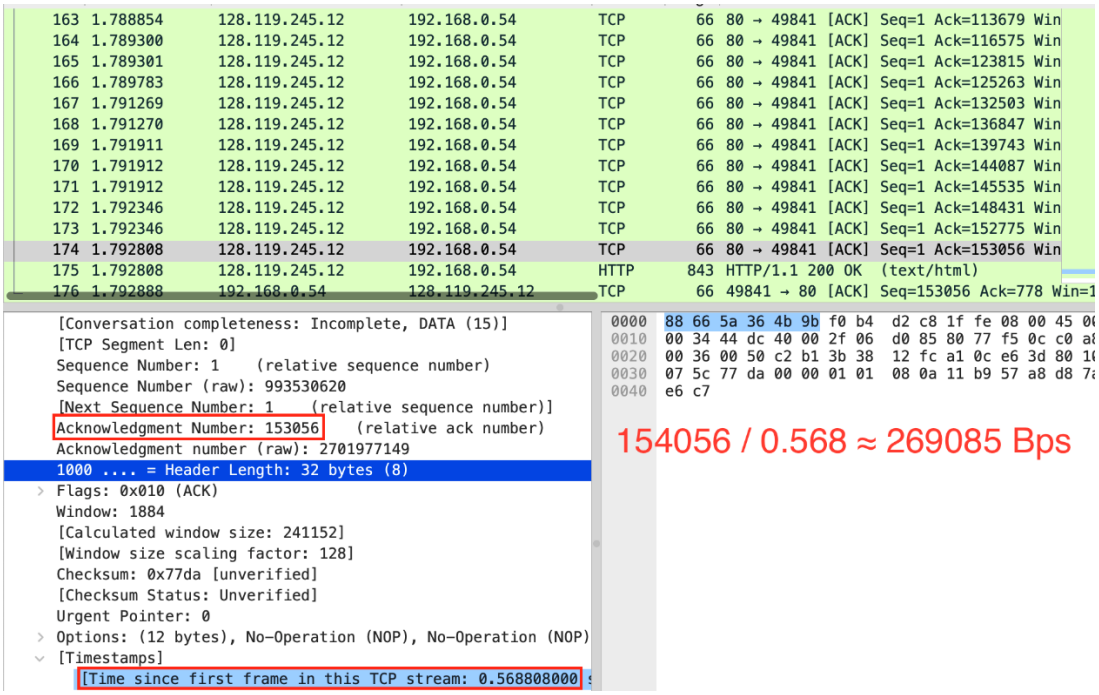


(tcp.stream == 4) && !(tcp.reassembled_in == 160)

Source	Destination	Protocol	Length	Info
192.168.0.54	128.119.245.12	TCP	78	49841 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSva
128.119.245.12	192.168.0.54	TCP	74	80 → 49841 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460
192.168.0.54	128.119.245.12	TCP	66	49841 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=363193
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=735 Win=30464 Len=0 TSval=29735
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=2183 Win=33408 Len=0 TSval=2973
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=3631 Win=36224 Len=0 TSval=2973
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=5079 Win=39168 Len=0 TSval=2973
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=6527 Win=42112 Len=0 TSval=2973
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=7975 Win=44928 Len=0 TSval=2973
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=9423 Win=47872 Len=0 TSval=2973
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=10871 Win=50816 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=12319 Win=53632 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=13767 Win=56576 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=15215 Win=59392 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=18111 Win=65280 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=21007 Win=71040 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=23903 Win=76800 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=26799 Win=82560 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	74	80 → 49841 [ACK] Seq=1 Ack=29695 Win=88448 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	74	80 → 49841 [ACK] Seq=1 Ack=31143 Win=91264 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	74	80 → 49841 [ACK] Seq=1 Ack=32591 Win=94208 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=35487 Win=99968 Len=0 TSval=297
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=38383 Win=105728 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=41279 Win=111616 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=44175 Win=117376 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=49967 Win=128896 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	74	80 → 49841 [ACK] Seq=1 Ack=51415 Win=131840 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=55759 Win=140544 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=61551 Win=152064 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=62999 Win=155008 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	74	80 → 49841 [ACK] Seq=1 Ack=67343 Win=163712 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	74	80 → 49841 [ACK] Seq=1 Ack=70239 Win=169472 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	74	80 → 49841 [ACK] Seq=1 Ack=73135 Win=175232 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	74	80 → 49841 [ACK] Seq=1 Ack=76031 Win=180608 Len=0 TSval=29
128.119.245.12	192.168.0.54	TCP	66	80 → 49841 [ACK] Seq=1 Ack=77479 Win=179584 Len=0 TSval=29

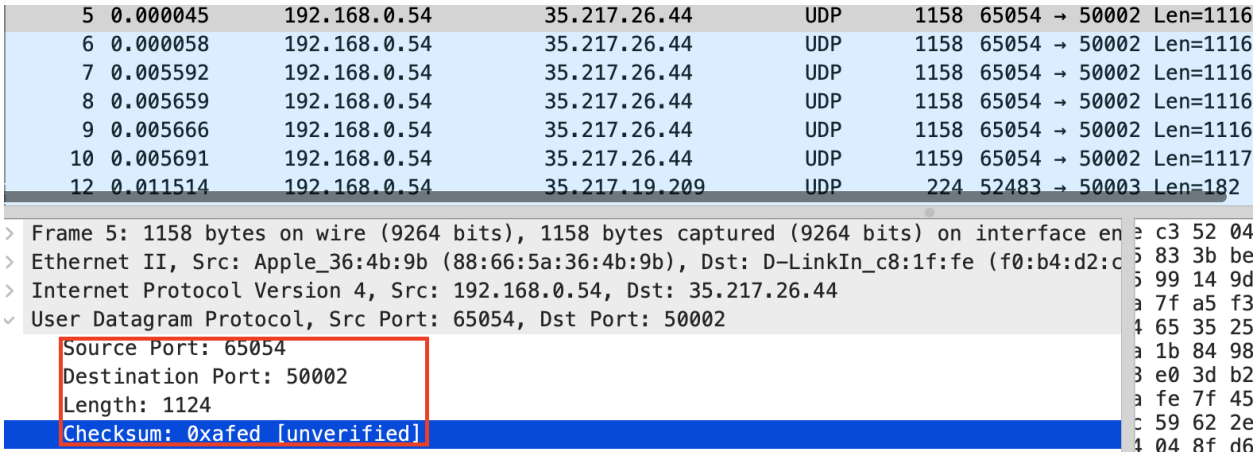
Vidutiniškai *aknolege'ina* apie 1448 baitų. (inkrementas tarp ACK segmentų).

12. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.



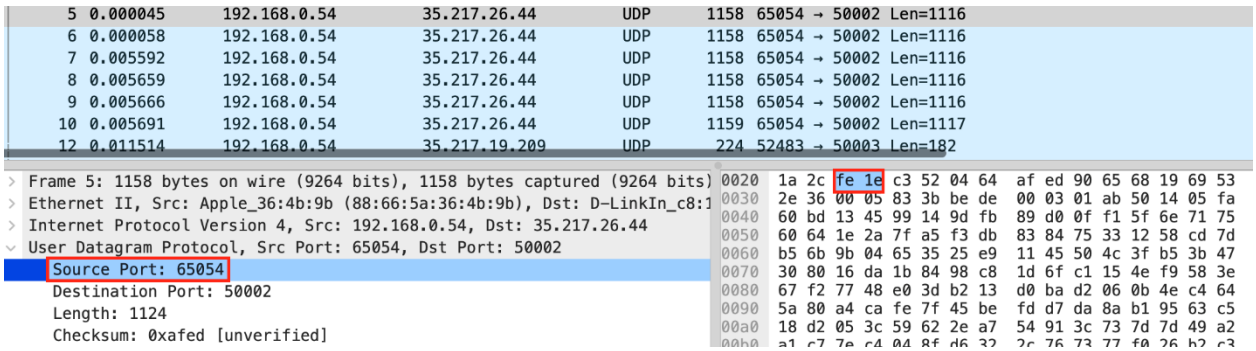
UDP paketų filtravimas

1. Select one UDP packet from your trace. From this packet, determine how many fields there are in the UDP header. (You shouldn't look in the textbook! Answer these questions directly from what you observe in the packet trace.) Name these fields.



- source port
- destination port
- length
- checksum.

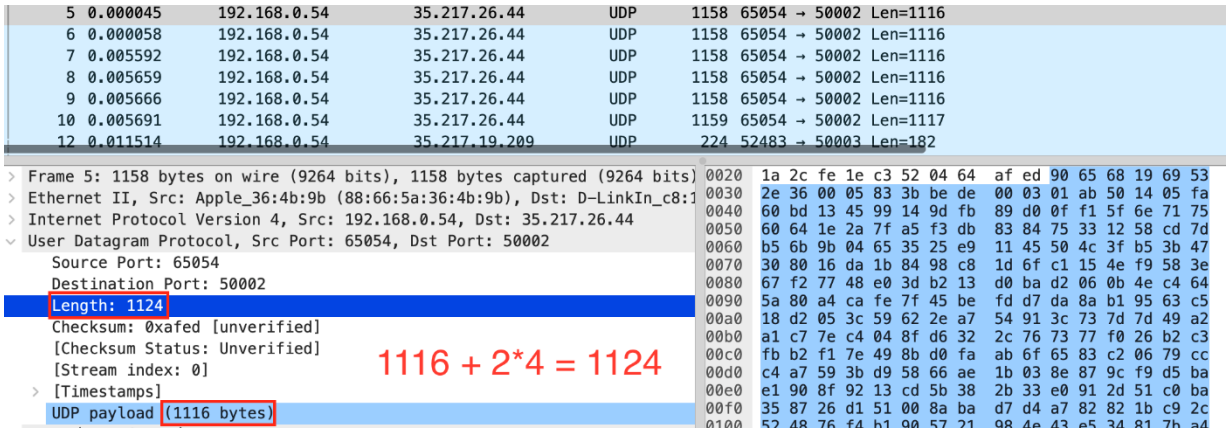
2. By consulting the displayed information in Wireshark's packet content field for this packet, determine the length (in bytes) of each of the UDP header fields.



2 baitai, nes čia yra *„hexadecimal characters"*, kurie kiekvienas užima 4 bitus, tai jų pora yra oktetas, arba vienas baitas.

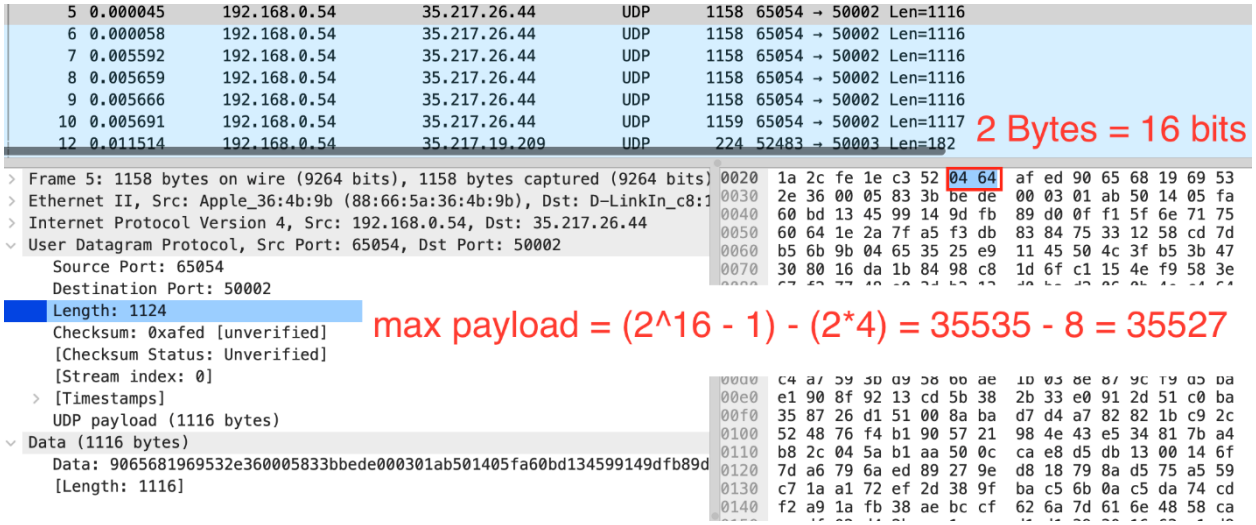
3. The value in the Length field is the length of what? (You can consult the text for this answer). Verify your claim with your captured UDP packet.

Paketo dydis (ar ilgis) susideda iš antraštės (*header*) ir siunčiamų duomenų. Taigi, imant tą patį pavyzdį:



Kadangi headeryje yra 4 laukai, ir kiekvienas užima po 2 baitus, pats headeris sudaro 8 baitus. Likę baitai yra skirti payload'ui. O „length" rodo visą segmento dydį (i.e. headerio ir payload sumą)

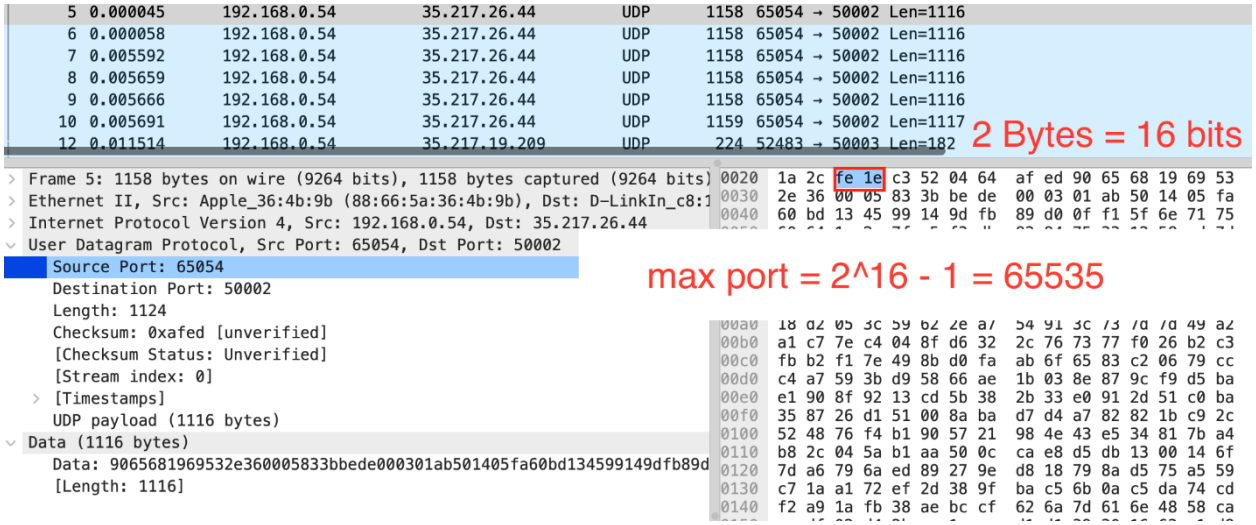
4. What is the maximum number of bytes that can be included in a UDP payload? (Hint: the answer to this question can be determined by your answer to 2. above)



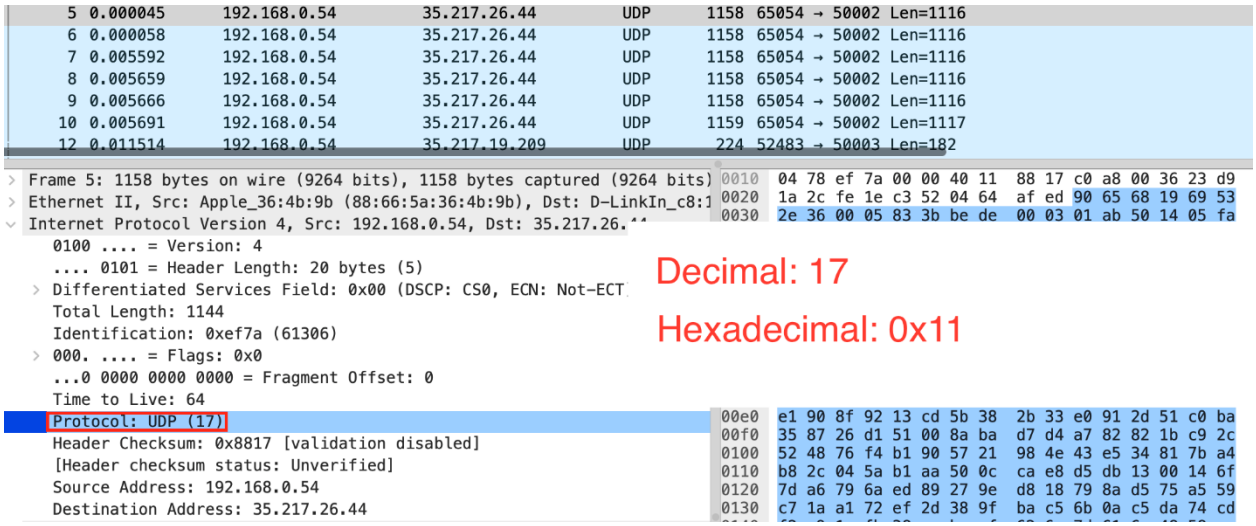
Pirma, suskaičiuojame kokį didžiausią skaičių gali laikyti „length" field – $2^{16} - 1$ (nes 0 įskaičiuojame). Tuomet iš gauto skaičiaus atimame headerio dydį – 8 baitus, ir gauname max payload dydį.

P.S. Čia yra skaičiavimo klaida, vietoj 3, turėtų būti 6. Teisingas atsakymas 65527 baitai.

5. What is the largest possible source port number? (Hint: see the hint in 4.)

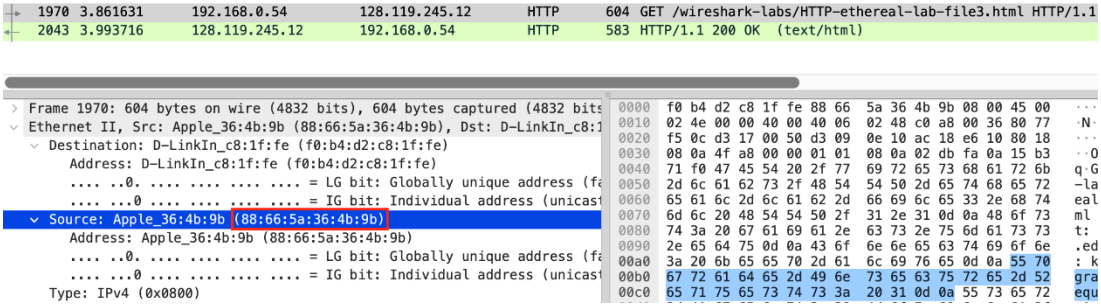


6. What is the protocol number for UDP? Give your answer in both hexadecimal and decimal notation. To answer this question, you'll need to look into the Protocol field of the IP datagram containing this UDP segment (see Figure 4.13 in the text, and the discussion of IP header fields).

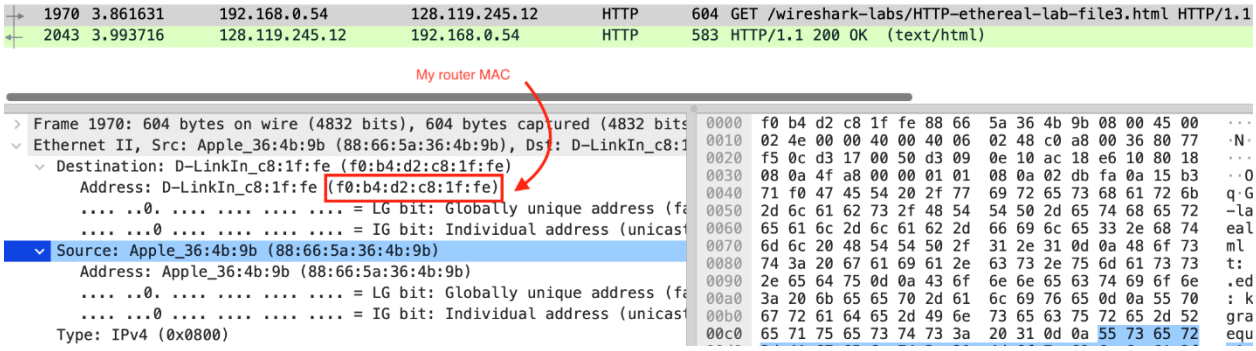


Ethernet ir ARP paketų filtravimas

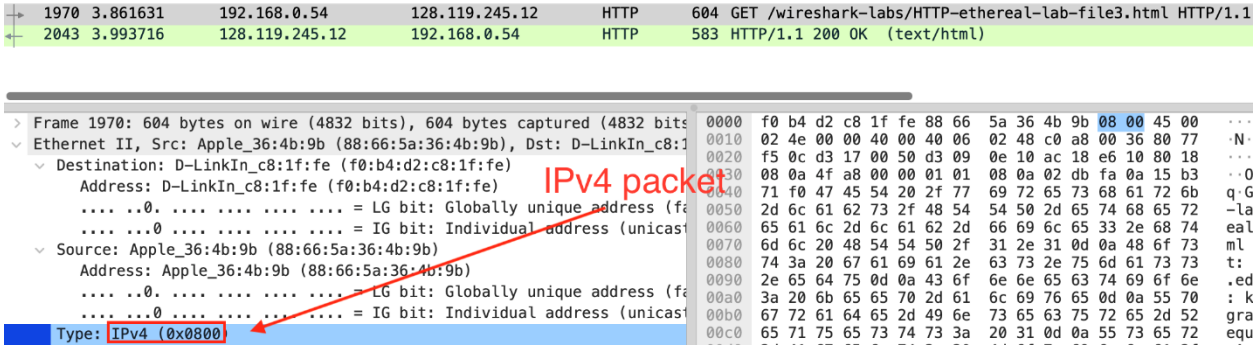
1. What is the 48-bit Ethernet address of your computer?



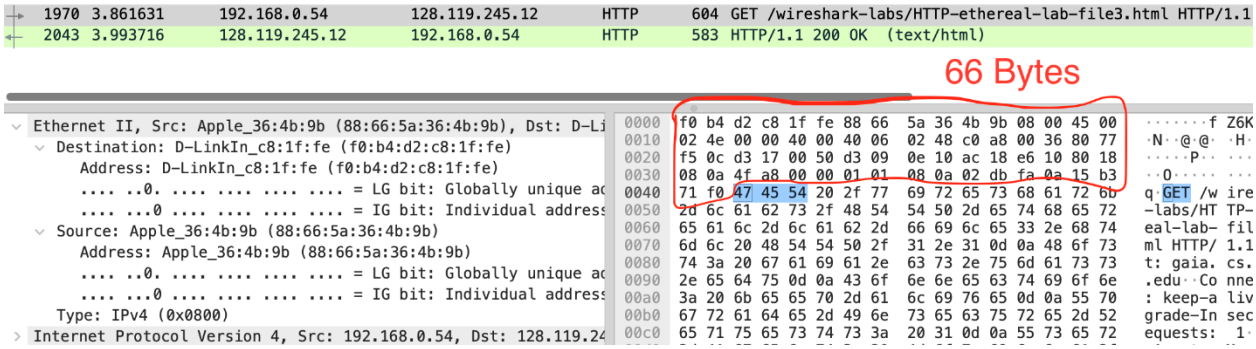
2. What is the 48-bit destination address in the Ethernet frame? Is this the Ethernet address of gaia.cs.umass.edu? (Hint: the answer is no). What device has this as its Ethernet address? [Note: this is an important question, and one that students sometimes get wrong. Re-read pages 468-469 in the text and make sure you understand the answer here.]



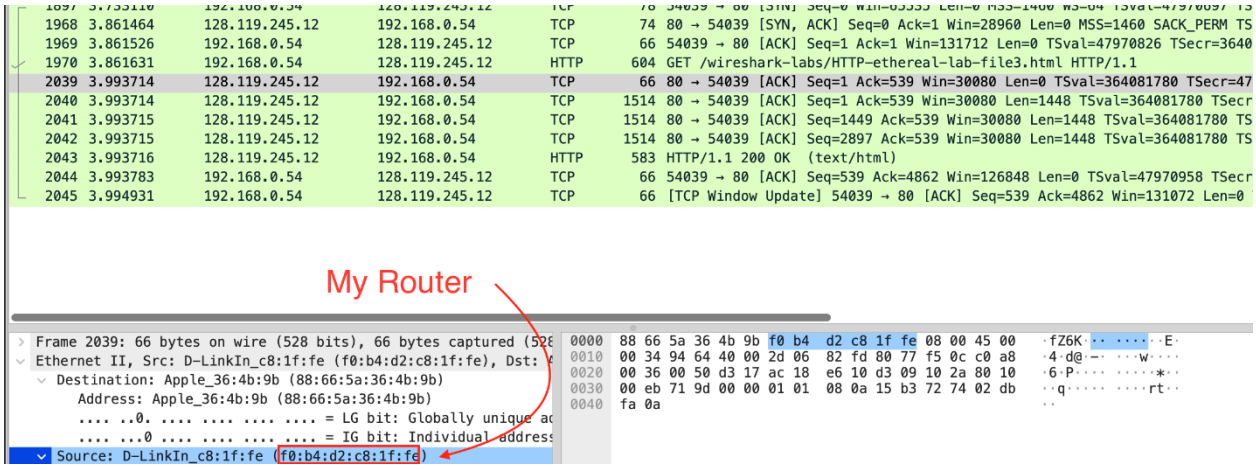
3. Give the hexadecimal value for the two-byte Frame type field. What upper layer protocol does this correspond to?



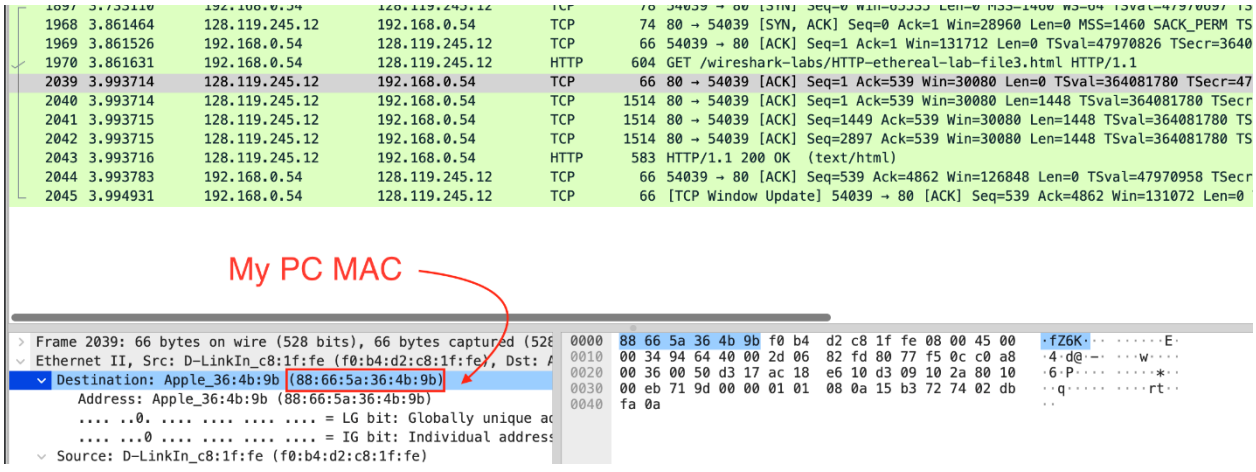
4. How many bytes from the very start of the Ethernet frame does the ASCII “G” in “GET” appear in the Ethernet frame?



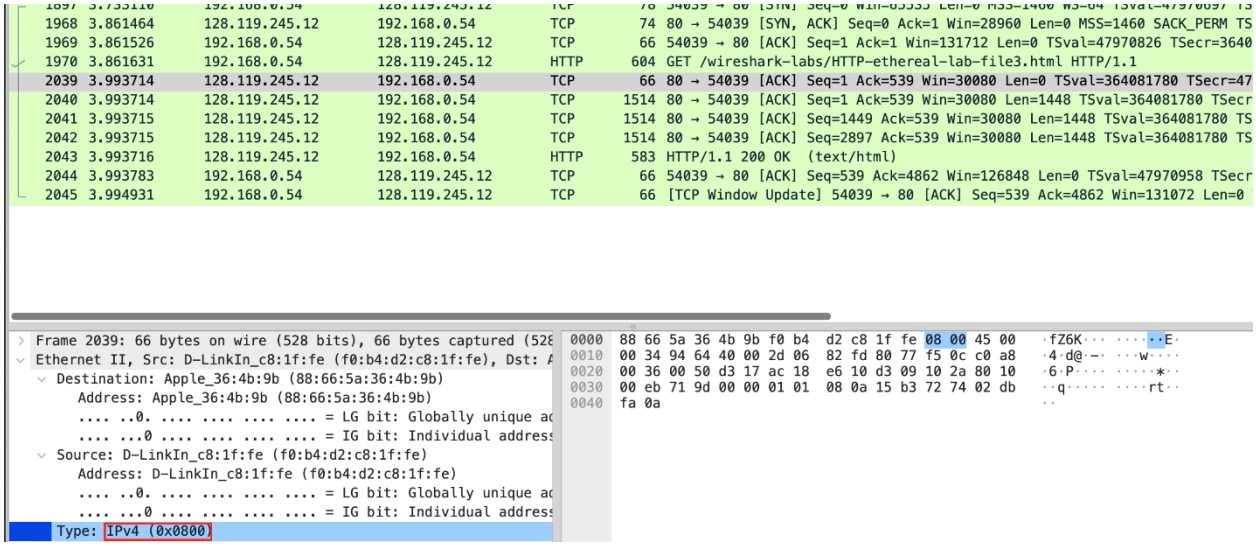
5. What is the value of the Ethernet source address? Is this the address of your computer, or of gaia.cs.umass.edu (Hint: the answer is no). What device has this as its Ethernet address?



6. What is the destination address in the Ethernet frame? Is this the Ethernet address of your computer?



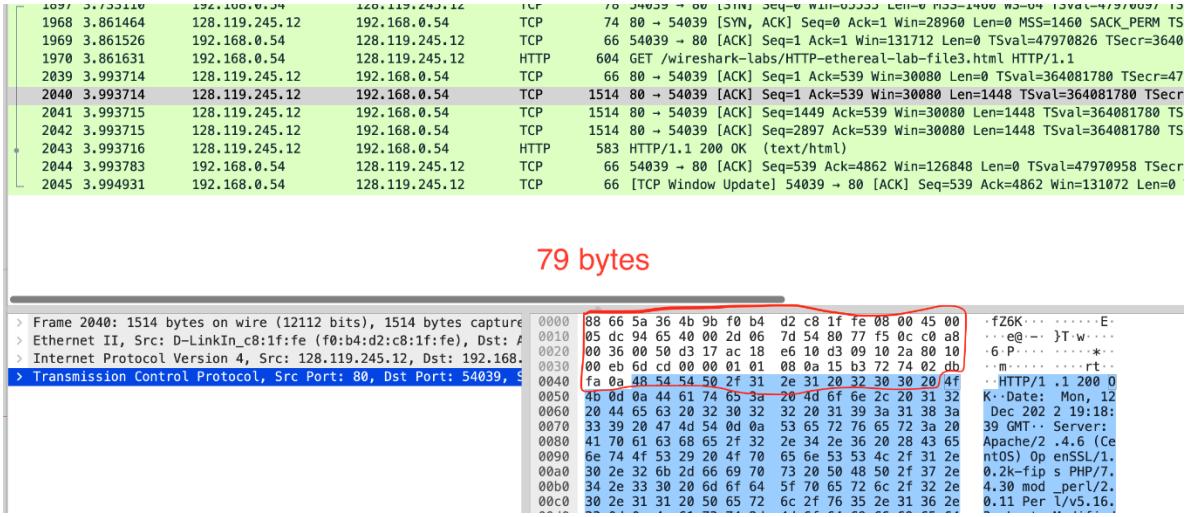
7. Give the hexadecimal value for the two-byte Frame type field. What upper layer protocol does this correspond to?



Vėl IPv4 protokolo kodas

.

8. How many bytes from the very start of the Ethernet frame does the ASCII “O” in “OK” (i.e., the HTTP response code) appear in the Ethernet frame?



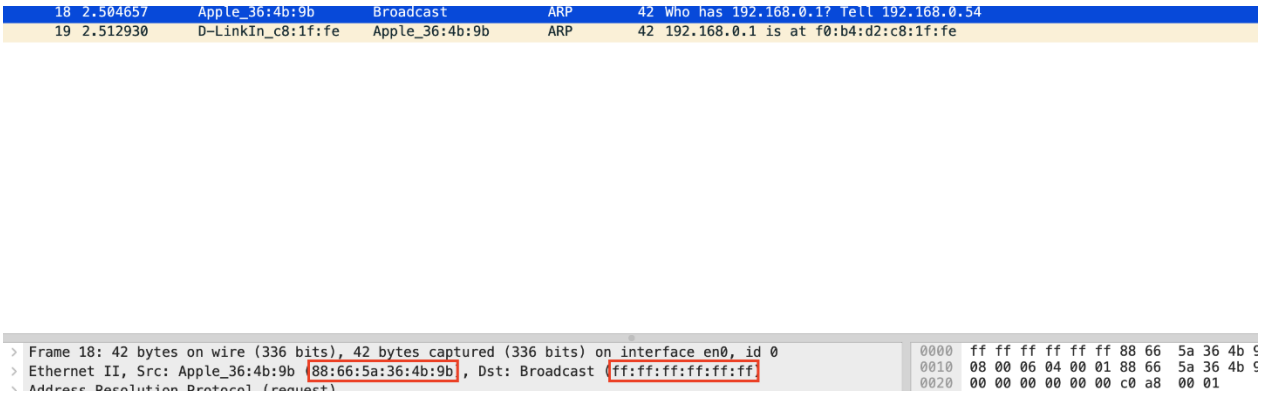
9. Write down the contents of your computer's ARP cache. What is the meaning of each column value?

```
~ $ arp -a
? (192.168.0.1) at f0:b4:d2:c8:1f:fe on en0 ifscope [ethernet]
? (224.0.0.251) at 1:0:5e:0:0:fb on en0 ifscope permanent [ethernet]
? (239.255.255.250) at 1:0:5e:7f:ff:fa on en0 ifscope permanent [ethernet]
```

Stulpeliai:

- IP adresas
- MAC adresas
- NIC interface, kuriam priklauso šis įrašas
- Įrašo tipas: „permanent" ar ne

10. What are the hexadecimal values for the source and destination addresses in the Ethernet frame containing the ARP request message?



11. Give the hexadecimal value for the two-byte Ethernet Frame type field. What upper layer protocol does this correspond to?