Homework 4

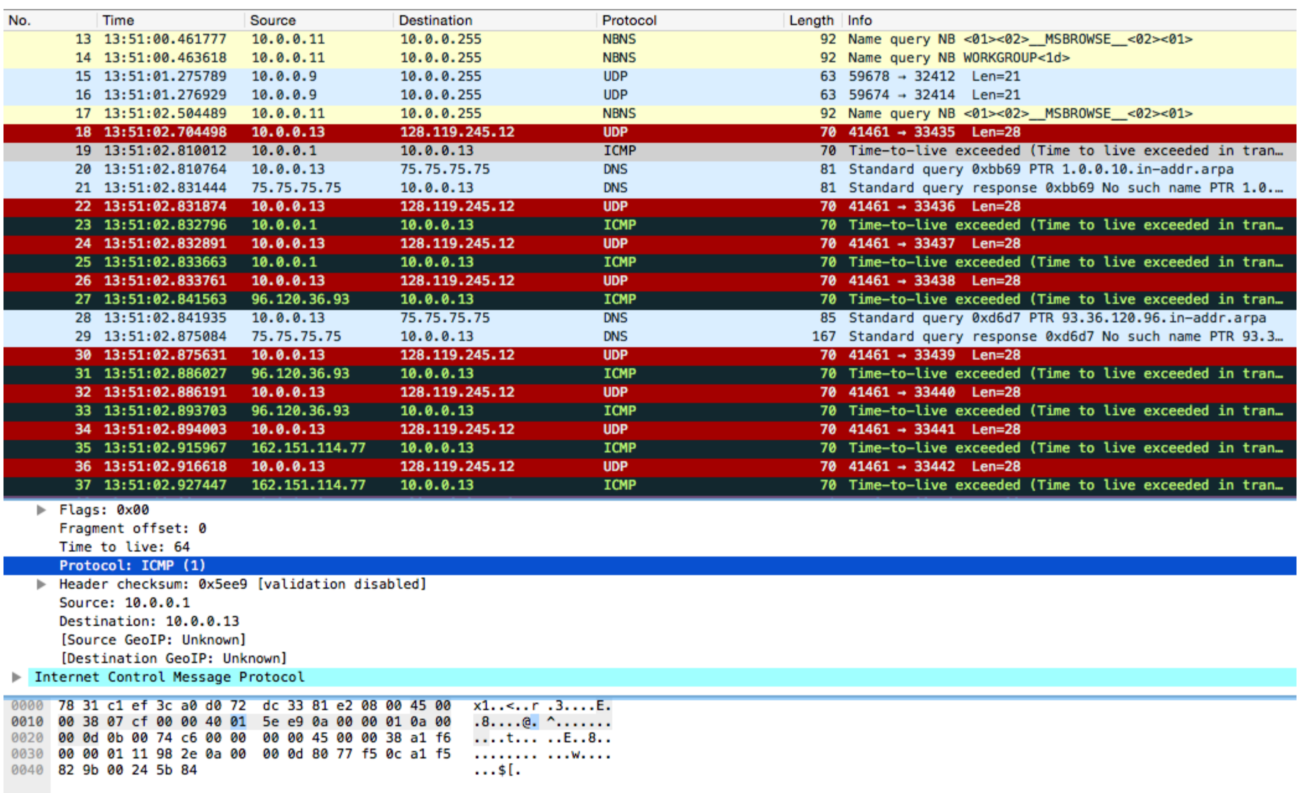
lukas_borges

Wireshark labs

~~http://www-net.cs.umass.edu/wireshark-labs/Wireshark_IP_v6.0.pdf~~
~~http://www-net.cs.umass.edu/wireshark-labs/Wireshark_ICMP_v6.0.pdf~~
~~http://www-net.cs.umass.edu/wireshark-labs/Wireshark_Ethernet_ARP_v6.01.pdf~~

Wireshark lab: IP v6.0

Captured Trace:



No.	Time	Source	Destination	Protocol	Length	Info
13	13:51:00.461777	10.0.0.11	10.0.0.255	NBNS	92	Name query NB <01><02>__MSBROWSE__<02><01>
14	13:51:00.463618	10.0.0.11	10.0.0.255	NBNS	92	Name query NB WORKGROUP<1d>
15	13:51:01.275789	10.0.0.9	10.0.0.255	UDP	63	59678 → 32412 Len=21
16	13:51:01.276929	10.0.0.9	10.0.0.255	UDP	63	59674 → 32414 Len=21
17	13:51:02.504489	10.0.0.11	10.0.0.255	NBNS	92	Name query NB <01><02>__MSBROWSE__<02><01>
18	13:51:02.704498	10.0.0.13	128.119.245.12	UDP	70	41461 → 33435 Len=28
19	13:51:02.810012	10.0.0.1	10.0.0.13	ICMP	70	Time-to-live exceeded (Time to live exceeded in tran…
20	13:51:02.810764	10.0.0.13	75.75.75.75	DNS	81	Standard query 0xbb69 PTR 1.0.0.10.in-addr.arpa
21	13:51:02.831444	75.75.75.75	10.0.0.13	DNS	81	Standard query response 0xbb69 No such name PTR 1.0.…
22	13:51:02.831874	10.0.0.13	128.119.245.12	UDP	70	41461 → 33436 Len=28
23	13:51:02.832796	10.0.0.1	10.0.0.13	ICMP	70	Time-to-live exceeded (Time to live exceeded in tran…
24	13:51:02.832891	10.0.0.13	128.119.245.12	UDP	70	41461 → 33437 Len=28
25	13:51:02.833663	10.0.0.1	10.0.0.13	ICMP	70	Time-to-live exceeded (Time to live exceeded in tran…
26	13:51:02.833761	10.0.0.13	128.119.245.12	UDP	70	41461 → 33438 Len=28
27	13:51:02.841563	96.120.36.93	10.0.0.13	ICMP	70	Time-to-live exceeded (Time to live exceeded in tran…
28	13:51:02.841935	10.0.0.13	75.75.75.75	DNS	85	Standard query 0xd6d7 PTR 93.36.120.96.in-addr.arpa
29	13:51:02.875084	75.75.75.75	10.0.0.13	DNS	167	Standard query response 0xd6d7 No such name PTR 93.3…
30	13:51:02.875631	10.0.0.13	128.119.245.12	UDP	70	41461 → 33439 Len=28
31	13:51:02.886027	96.120.36.93	10.0.0.13	ICMP	70	Time-to-live exceeded (Time to live exceeded in tran…
32	13:51:02.886191	10.0.0.13	128.119.245.12	UDP	70	41461 → 33440 Len=28
33	13:51:02.893703	96.120.36.93	10.0.0.13	ICMP	70	Time-to-live exceeded (Time to live exceeded in tran…
34	13:51:02.894003	10.0.0.13	128.119.245.12	UDP	70	41461 → 33441 Len=28
35	13:51:02.915967	162.151.114.77	10.0.0.13	ICMP	70	Time-to-live exceeded (Time to live exceeded in tran…
36	13:51:02.916618	10.0.0.13	128.119.245.12	UDP	70	41461 → 33442 Len=28
37	13:51:02.927447	162.151.114.77	10.0.0.13	ICMP	70	Time-to-live exceeded (Time to live exceeded in tran…

```
▶ Flags: 0x00
  Fragment offset: 0
  Time to live: 64
  Protocol: ICMP (1)
▶ Header checksum: 0x5ee9 [validation disabled]
  Source: 10.0.0.1
  Destination: 10.0.0.13
  [Source GeoIP: Unknown]
  [Destination GeoIP: Unknown]
▶ Internet Control Message Protocol

0000  78 31 c1 ef 3c a0 d0 72  dc 33 81 e2 08 00 45 00   x1..<..r .3....E.
0010  00 38 07 cf 00 00 40 01  5e e9 0a 00 00 01 0a 00   .8....@. ^.......
0020  00 0d 0b 00 74 c6 00 00  00 00 45 00 00 38 a1 f6   ....t... ..E..8..
0030  00 00 01 11 98 2e 0a 00  00 0d 80 77 f5 0c a1 f5   ........ ...w....
0040  82 9b 00 24 5b 84                                  ...$[.
```

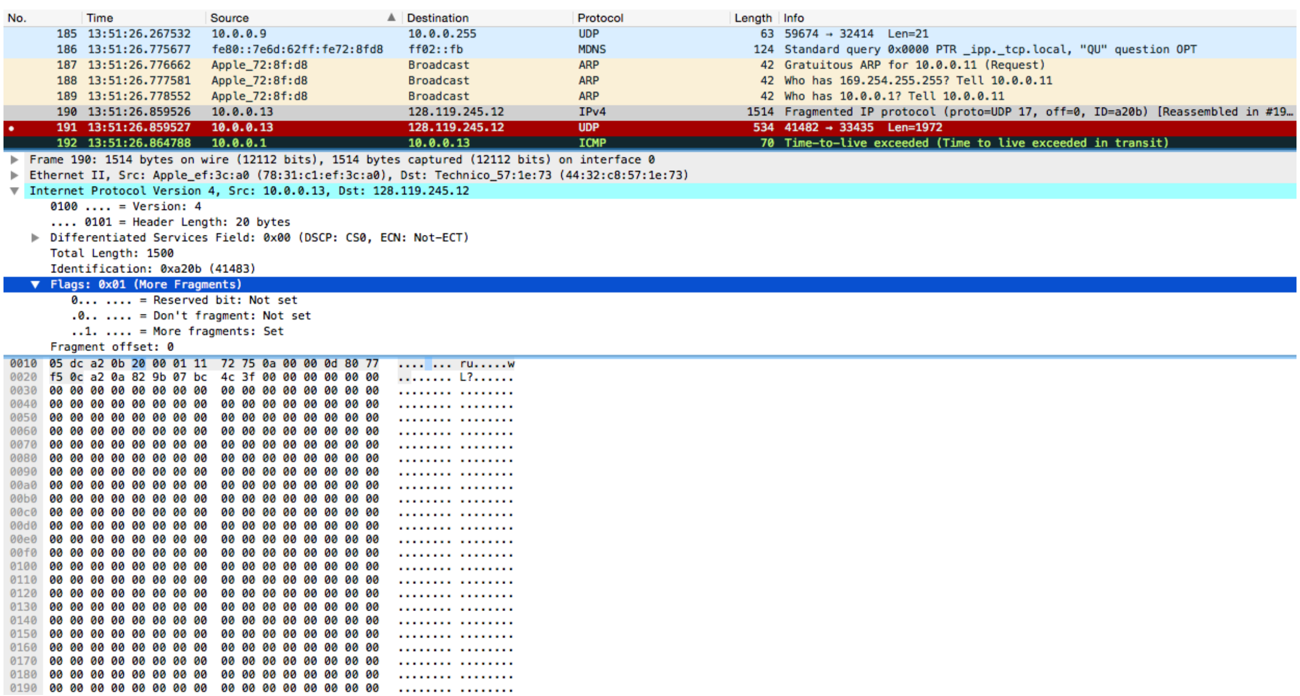
1. Computer: `10.0.0.13`
2. ICMP `(0x01)`
3. Header length: 20 bytes, Total Length: 56. $56 - 20 = 36$

   36 bytes in the payload of the IP datagram.
4. Fragment offset: 0. No fragmentation.

5. Identification, Time to live and Header checksum.
6. *Fields that remain constant:* Version (IPv4 for all), header length (ICMP packets), source IP (always sending from same source), destination, differentiated services (always ICMP type of service class), upper layer protocol.
   Fields that must stay constant match the ones that remained constant.
   Fields that must change:
   Identification (IP packets must have different ids), TTL (traceroute increments each subsequent packet), Header Checksum (if header changed, checksum changes too).
7. The IP header Identification fields increment with each ICMP Echo request.
8. Identification: `0x07cd` (1999) TTL: 64
9. TTL remains unchanged because the TTL for the first hop router is the same.

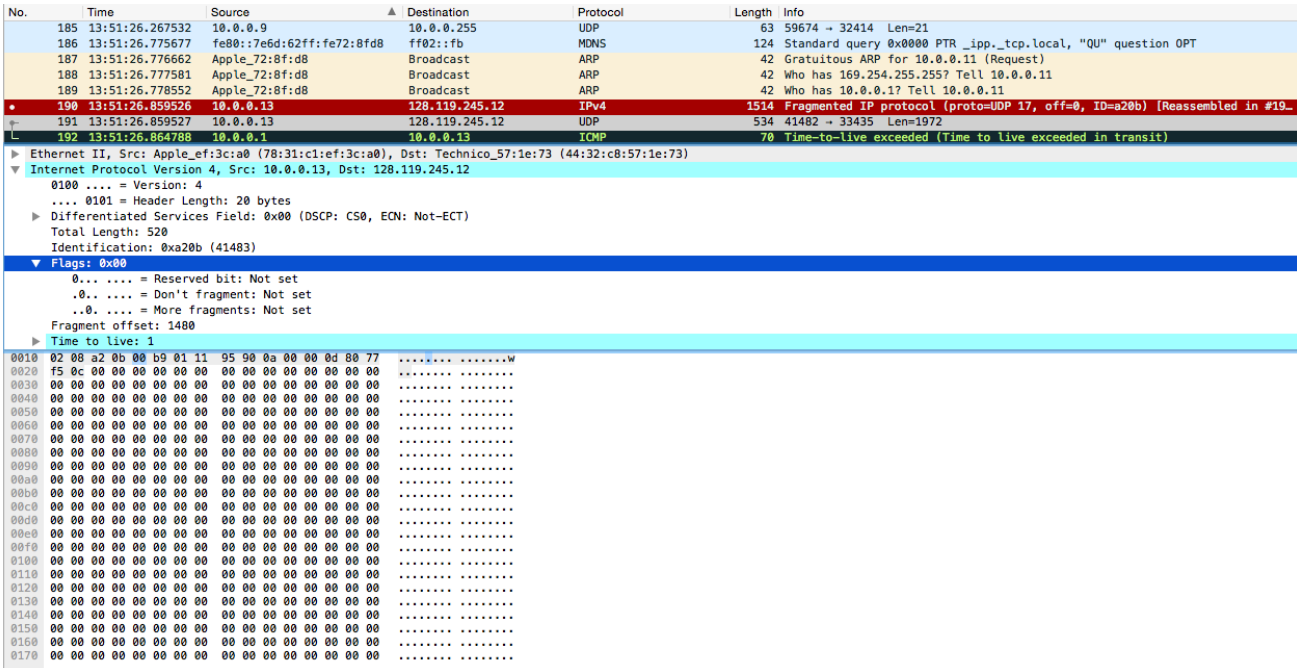
Fragmentation

10. Yes.
11.



The “More fragments” flag is set.
Length: 1500 bytes.

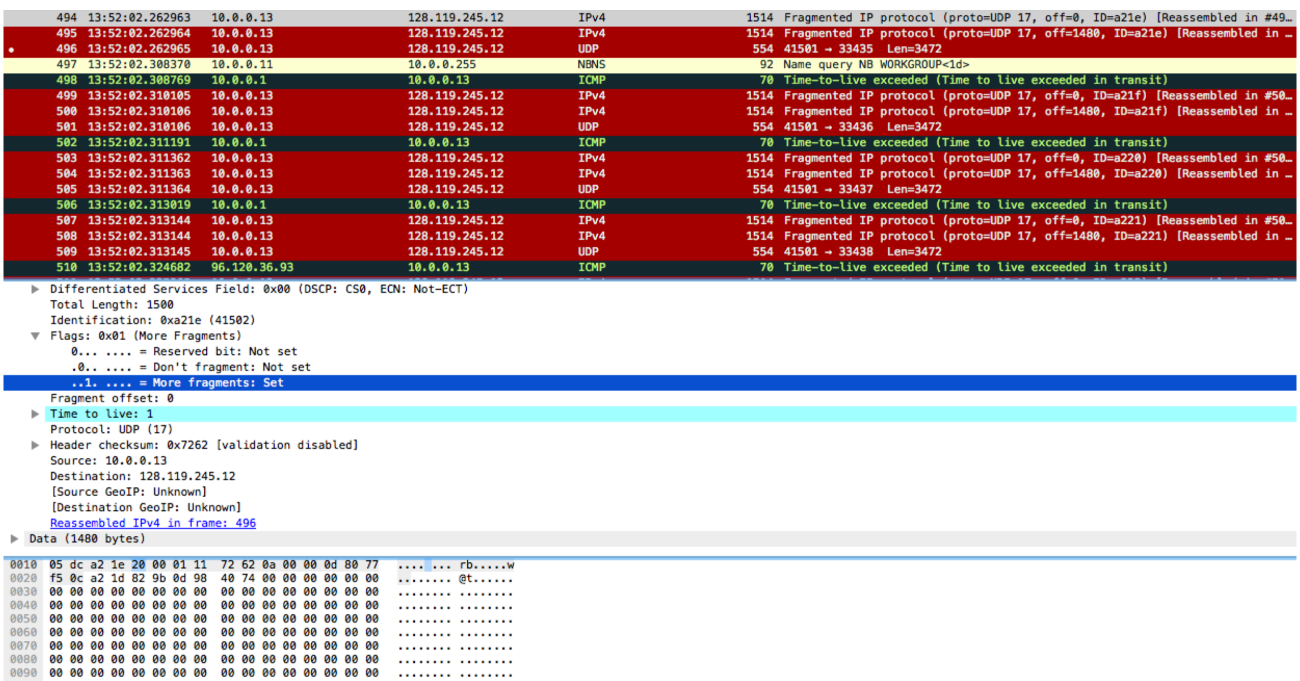
12.



The following has fragment offset of 0, “More fragments: not set” (there are no more fragments from this point onwards).

The offset 1480 indicates it is the remainder of the fragment.

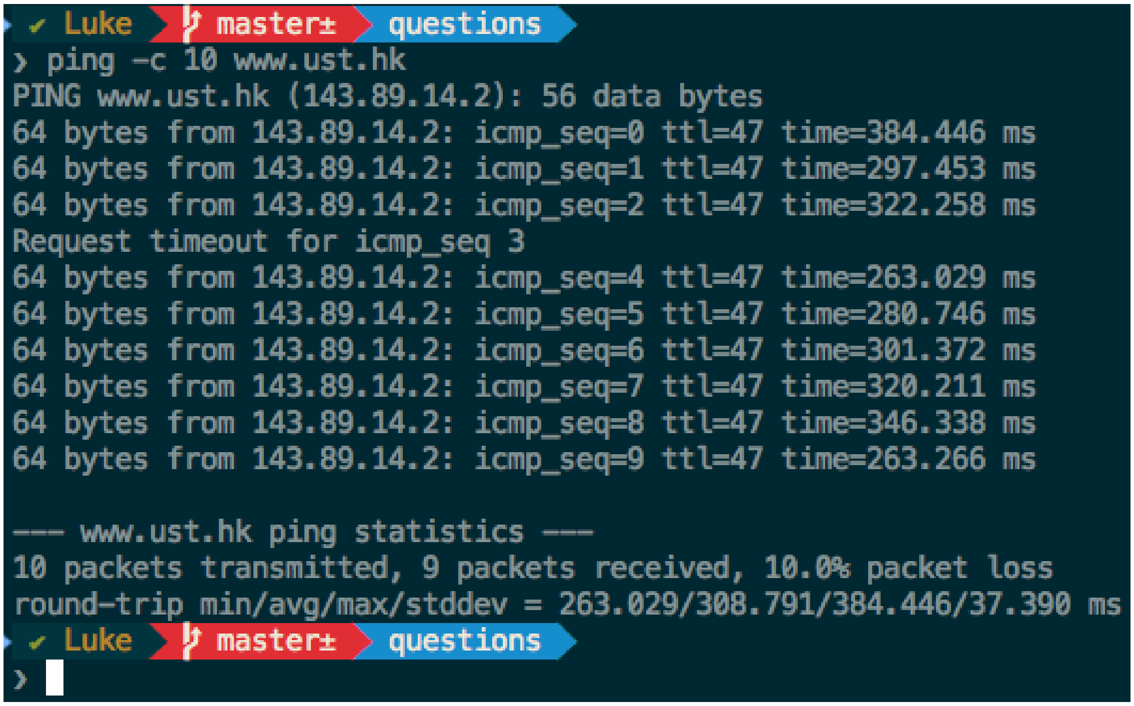
13. Total length, More Fragments and Fragment offset.



14. The original request created 2 fragments. 3 packets total.
15. From 1 to 2: fragment offset.

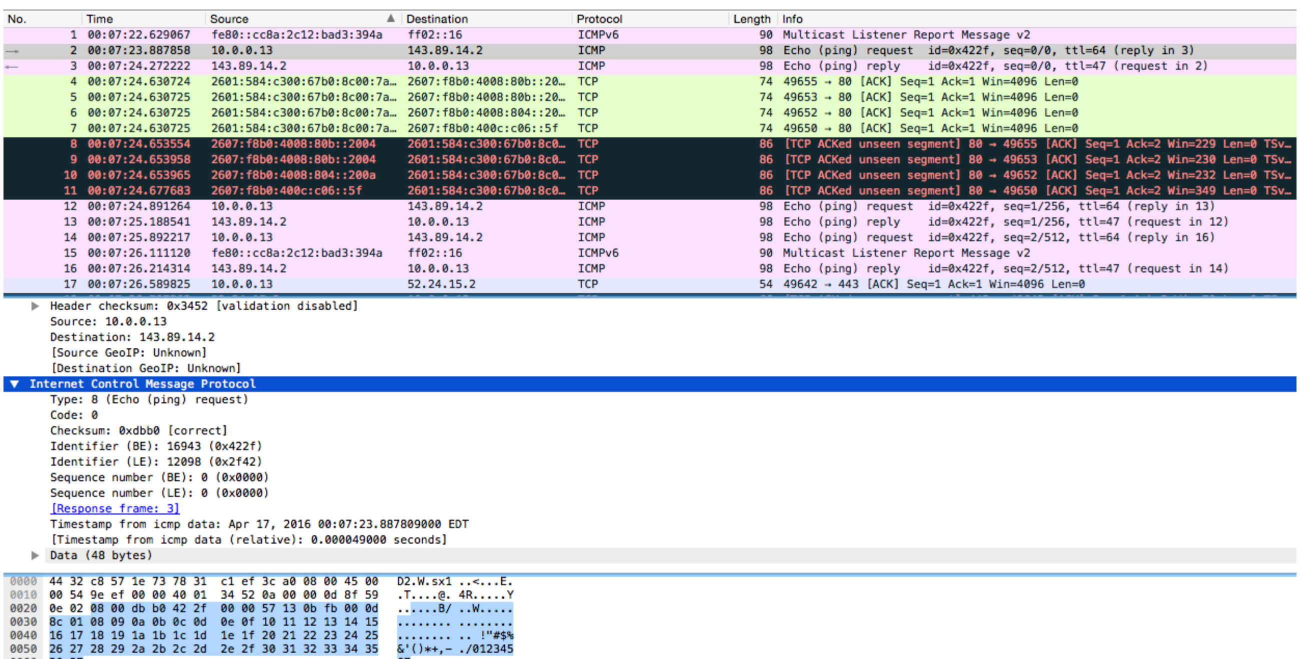
    From 2 to 3: fragment offset, more fragments, total length.

Wireshark lab: ICMP v6.0



```
✔ Luke  master±  questions
❯ ping -c 10 www.ust.hk
PING www.ust.hk (143.89.14.2): 56 data bytes
64 bytes from 143.89.14.2: icmp_seq=0 ttl=47 time=384.446 ms
64 bytes from 143.89.14.2: icmp_seq=1 ttl=47 time=297.453 ms
64 bytes from 143.89.14.2: icmp_seq=2 ttl=47 time=322.258 ms
Request timeout for icmp_seq 3
64 bytes from 143.89.14.2: icmp_seq=4 ttl=47 time=263.029 ms
64 bytes from 143.89.14.2: icmp_seq=5 ttl=47 time=280.746 ms
64 bytes from 143.89.14.2: icmp_seq=6 ttl=47 time=301.372 ms
64 bytes from 143.89.14.2: icmp_seq=7 ttl=47 time=320.211 ms
64 bytes from 143.89.14.2: icmp_seq=8 ttl=47 time=346.338 ms
64 bytes from 143.89.14.2: icmp_seq=9 ttl=47 time=263.266 ms

--- www.ust.hk ping statistics ---
10 packets transmitted, 9 packets received, 10.0% packet loss
round-trip min/avg/max/stddev = 263.029/308.791/384.446/37.390 ms
✔ Luke  master±  questions
❯
```

1. My IP: `10.0.0.13`
   Host: `143.89.14.2`
2. Because it is a very simple protocol designed as a session less protocol, not designed for applications.

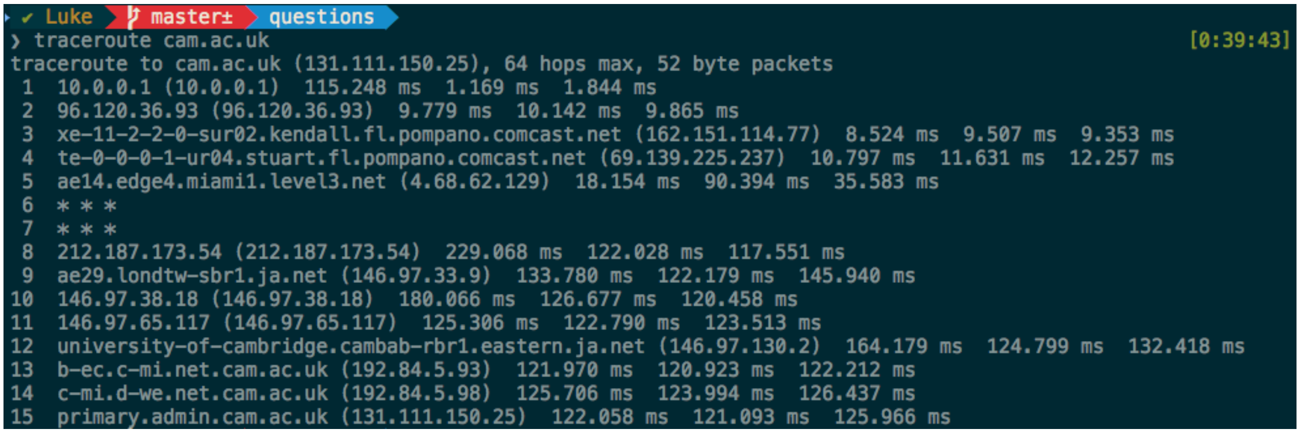


No.	Time	Source	Destination	Protocol	Length	Info
1	00:07:22.629067	fe80::cc8a:2c12:bad3:394a	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
2	00:07:23.887858	10.0.0.13	143.89.14.2	ICMP	98	Echo (ping) request id=0x422f, seq=0/0, ttl=64 (reply in 3)
3	00:07:24.272222	143.89.14.2	10.0.0.13	ICMP	98	Echo (ping) reply id=0x422f, seq=0/0, ttl=47 (request in 2)
4	00:07:24.630724	2601:584:c300:67b0:8c00:7a…	2607:f8b0:4008:80b::20…	TCP	74	49655 → 80 [ACK] Seq=1 Ack=1 Win=4096 Len=0
5	00:07:24.630725	2601:584:c300:67b0:8c00:7a…	2607:f8b0:4008:80b::20…	TCP	74	49653 → 80 [ACK] Seq=1 Ack=1 Win=4096 Len=0
6	00:07:24.630725	2601:584:c300:67b0:8c00:7a…	2607:f8b0:4008:804::20…	TCP	74	49652 → 80 [ACK] Seq=1 Ack=1 Win=4096 Len=0
7	00:07:24.630725	2601:584:c300:67b0:8c00:7a…	2607:f8b0:400c:c06::5f	TCP	74	49650 → 80 [ACK] Seq=1 Ack=1 Win=4096 Len=0
8	00:07:24.653554	2607:f8b0:4008:80b::2004	2601:584:c300:67b0:8c0…	TCP	86	[TCP ACKed unseen segment] 80 → 49655 [ACK] Seq=1 Ack=2 Win=229 Len=0 TSv…
9	00:07:24.653958	2607:f8b0:4008:80b::2004	2601:584:c300:67b0:8c0…	TCP	86	[TCP ACKed unseen segment] 80 → 49653 [ACK] Seq=1 Ack=2 Win=230 Len=0 TSv…
10	00:07:24.653965	2607:f8b0:4008:804::200a	2601:584:c300:67b0:8c0…	TCP	86	[TCP ACKed unseen segment] 80 → 49652 [ACK] Seq=1 Ack=2 Win=232 Len=0 TSv…
11	00:07:24.677683	2607:f8b0:400c:c06::5f	2601:584:c300:67b0:8c0…	TCP	86	[TCP ACKed unseen segment] 80 → 49650 [ACK] Seq=1 Ack=2 Win=349 Len=0 TSv…
12	00:07:24.891264	10.0.0.13	143.89.14.2	ICMP	98	Echo (ping) request id=0x422f, seq=1/256, ttl=64 (reply in 13)
13	00:07:25.188541	143.89.14.2	10.0.0.13	ICMP	98	Echo (ping) reply id=0x422f, seq=1/256, ttl=47 (request in 12)
14	00:07:25.892217	10.0.0.13	143.89.14.2	ICMP	98	Echo (ping) request id=0x422f, seq=2/512, ttl=64 (reply in 16)
15	00:07:26.111120	fe80::cc8a:2c12:bad3:394a	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
16	00:07:26.214314	143.89.14.2	10.0.0.13	ICMP	98	Echo (ping) reply id=0x422f, seq=2/512, ttl=47 (request in 14)
17	00:07:26.589825	10.0.0.13	52.24.15.2	TCP	54	49642 → 443 [ACK] Seq=1 Ack=1 Win=4096 Len=0

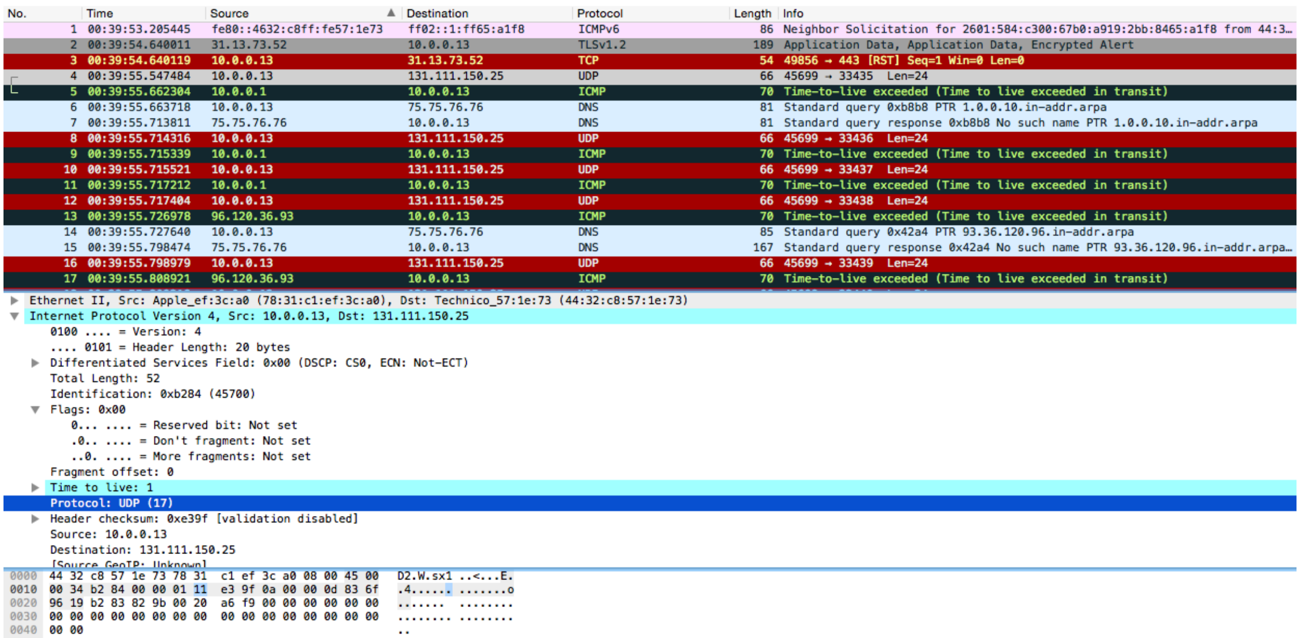
```
  ▶ Header checksum: 0x3452 [validation disabled]
    Source: 10.0.0.13
    Destination: 143.89.14.2
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▼ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xdbb0 [correct]
    Identifier (BE): 16943 (0x422f)
    Identifier (LE): 12098 (0x2f42)
    Sequence number (BE): 0 (0x0000)
    Sequence number (LE): 0 (0x0000)
    [Response frame: 3]
    Timestamp from icmp data: Apr 17, 2016 00:07:23.887809000 EDT
    [Timestamp from icmp data (relative): 0.000049000 seconds]
  ▶ Data (48 bytes)

0000  44 32 c8 57 1e 73 78 31  c1 ef 3c a0 08 00 45 00   D2.W.sx1 ..<...E.
0010  00 54 9e ef 00 00 40 01  34 52 0a 00 00 0d 8f 59   .T....@. 4R.....Y
0020  0e 02 08 00 db b0 42 2f  00 00 57 13 0b fb 00 0d   ......B/ ..W.....
0030  8c 01 08 09 0a 0b 0c 0d  0e 0f 10 11 12 13 14 15   ........ ........
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ........ .. !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67
```

3. Type: 8 Code: 0. Checksum, Identifier, Sequence Number, Timestamp and Data. 2 bytes for each checksum, sequence number and identifier field
4. Type and code both 0. Checksum, Identifier, Sequence Number, and Data. Also 2 bytes for each checksum, sequence number and identifier field.

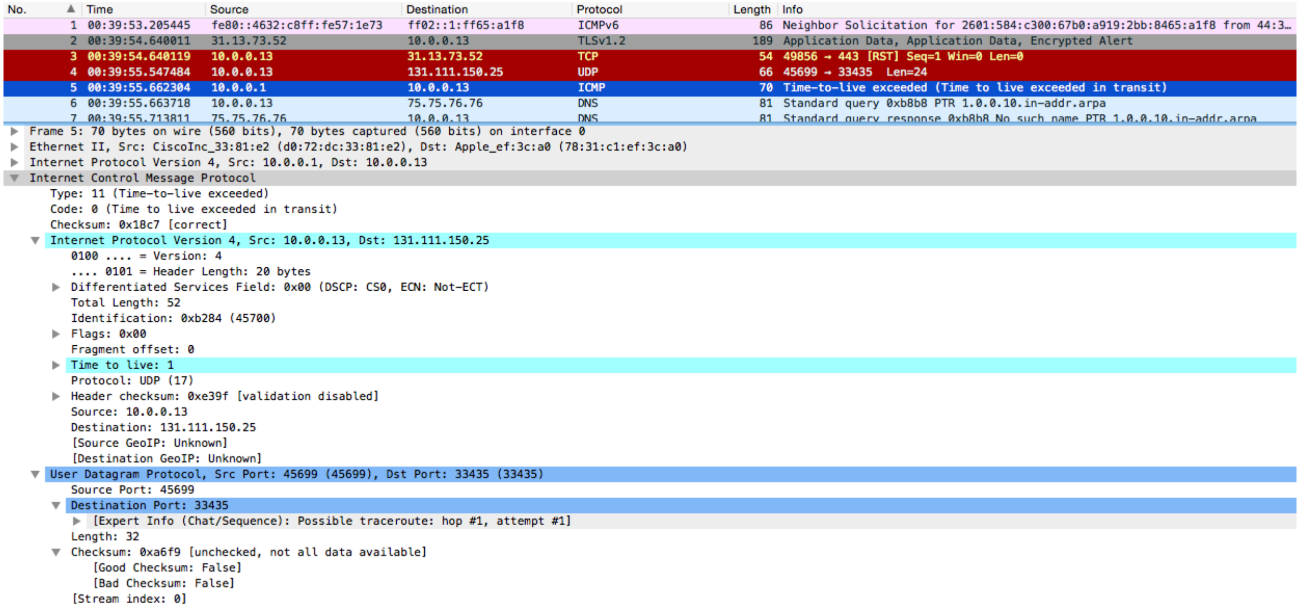
2. ICMP and Traceroute



5. Host: `10.0.0.13` Destination: `131.111.150.25`



6. Since I am running Unix, I can verify that the ICMP IP protocol number is 0x11
7. Yes, same fields.
8.

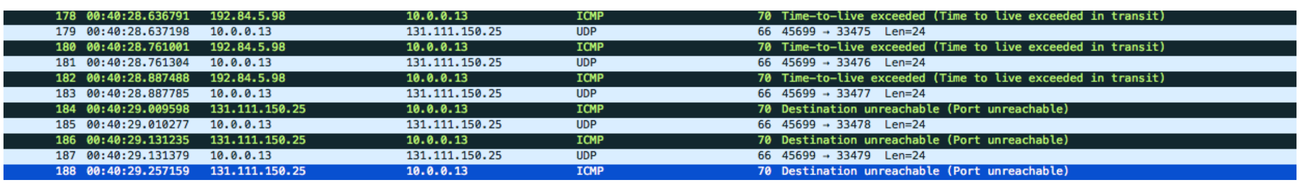


No.	Time	Source	Destination	Protocol	Length	Info
1	00:39:53.205445	fe80::4632:c8ff:fe57:1e73	ff02::1:ff65:a1f8	ICMPv6	86	Neighbor Solicitation for 2601:584:c300:67b0:a919:2bb:8465:a1f8 from 44:3…
2	00:39:54.640011	31.13.73.52	10.0.0.13	TLSv1.2	189	Application Data, Application Data, Encrypted Alert
3	00:39:54.640119	10.0.0.13	31.13.73.52	TCP	54	49856 → 443 [RST] Seq=1 Win=0 Len=0
4	00:39:55.547484	10.0.0.13	131.111.150.25	UDP	66	45699 → 33435 Len=24
5	00:39:55.662304	10.0.0.1	10.0.0.13	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
6	00:39:55.663718	10.0.0.13	75.75.76.76	DNS	81	Standard query 0xb8b8 PTR 1.0.0.10.in-addr.arpa
7	00:39:55.713811	75.75.76.76	10.0.0.13	DNS	81	Standard query response 0xb8b8 No such name PTR 1.0.0.10.in-addr.arpa

```
▶ Frame 5: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
▶ Ethernet II, Src: CiscoInc_33:81:e2 (d0:72:dc:33:81:e2), Dst: Apple_ef:3c:a0 (78:31:c1:ef:3c:a0)
▶ Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.13
▼ Internet Control Message Protocol
    Type: 11 (Time-to-live exceeded)
    Code: 0 (Time to live exceeded in transit)
    Checksum: 0x18c7 [correct]
  ▼ Internet Protocol Version 4, Src: 10.0.0.13, Dst: 131.111.150.25
      0100 .... = Version: 4
      .... 0101 = Header Length: 20 bytes
    ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      Total Length: 52
      Identification: 0xb284 (45700)
    ▶ Flags: 0x00
      Fragment offset: 0
    ▶ Time to live: 1
      Protocol: UDP (17)
    ▶ Header checksum: 0xe39f [validation disabled]
      Source: 10.0.0.13
      Destination: 131.111.150.25
      [Source GeoIP: Unknown]
      [Destination GeoIP: Unknown]
  ▼ User Datagram Protocol, Src Port: 45699 (45699), Dst Port: 33435 (33435)
      Source Port: 45699
    ▼ Destination Port: 33435
      ▶ [Expert Info (Chat/Sequence): Possible traceroute: hop #1, attempt #1]
      Length: 32
    ▼ Checksum: 0xa6f9 [unchecked, not all data available]
        [Good Checksum: False]
        [Bad Checksum: False]
      [Stream index: 0]
```

They are different packets. The ICMP contains both the IP header and the first 8 bytes of the original UDP packet request that the error is for.

9.



178	00:40:28.636791	192.84.5.98	10.0.0.13	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
179	00:40:28.637198	10.0.0.13	131.111.150.25	UDP	66	45699 → 33475 Len=24
180	00:40:28.761001	192.84.5.98	10.0.0.13	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
181	00:40:28.761304	10.0.0.13	131.111.150.25	UDP	66	45699 → 33476 Len=24
182	00:40:28.887488	192.84.5.98	10.0.0.13	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
183	00:40:28.887785	10.0.0.13	131.111.150.25	UDP	66	45699 → 33477 Len=24
184	00:40:29.009598	131.111.150.25	10.0.0.13	ICMP	70	Destination unreachable (Port unreachable)
185	00:40:29.010277	10.0.0.13	131.111.150.25	UDP	66	45699 → 33478 Len=24
186	00:40:29.131235	131.111.150.25	10.0.0.13	ICMP	70	Destination unreachable (Port unreachable)
187	00:40:29.131379	10.0.0.13	131.111.150.25	UDP	66	45699 → 33479 Len=24
188	00:40:29.257159	131.111.150.25	10.0.0.13	ICMP	70	Destination unreachable (Port unreachable)

They are different in the sense that the last 3 three ICMP packets are message type 0 (echo reply), instead of 11 (TTL expired). This means that for the last ICMP packets the datagrams traveled all the way to the destination host before the Time To Live expired.

10. Since we were tracerouting from North America to Europe (Cambridge - UK), we clearly have a long hop, which is also described by the `***` (meaning waiting time) shown on the console. We start from Miami - FL (kendall.fl.pompano.comcast.net), to Stuart (stuart.fl.pompano.comcast.net). From Stuart we move into Comcast's backbone, Level 3 Communication is going to forward our traffic.
    (network map available @ http://www.level3.com/~/media/files/maps/map_1115_interactive.pdf).
    According to Level 3's map we most likely travel up to New York (miami1.level3.net) and from there we cross the Atlantic to reach the United Kingdom and that is when the longer waiting time starts. The last two routers:

```
14  c-mi.d-we.net.cam.ac.uk (192.84.5.98)  125.706 ms  123.994 ms  126
.437 ms
15  primary.admin.cam.ac.uk (131.111.150.25)  122.058 ms  121.093 ms
125.966 ms
```

are most likely to be in the UK. (cam.ac.uk).

Wireshark lab: Ethernet ARP v6.01

1.

No.	Time	Source	Destination	Protocol	Length	Info
2	02:18:07.829647	fe80::7e6d:62ff:fe72:8fd8	ff02::fb	MDNS	217	Standard query
3	02:18:08.343172	Apple_72:8f:d8	Broadcast	0x0800	110	IPv4
4	02:18:10.099363	Apple_ef:3c:a0	Technico_57:1e:73	0x0800	78	IPv4
5	02:18:10.179861	Technico_57:1e:73	Apple_ef:3c:a0	0x0800	74	IPv4
6	02:18:10.179968	Apple_ef:3c:a0	Technico_57:1e:73	0x0800	66	IPv4
7	02:18:10.180358	Apple_ef:3c:a0	Technico_57:1e:73	0x0800	410	IPv4
8	02:18:10.233067	Technico_57:1e:73	Apple_ef:3c:a0	0x0800	66	IPv4
9	02:18:10.233596	Technico_57:1e:73	Apple_ef:3c:a0	0x0800	66	IPv4
10	02:18:10.233606	Technico_57:1e:73	Apple_ef:3c:a0	0x0800	1514	IPv4
11	02:18:10.233608	Technico_57:1e:73	Apple_ef:3c:a0	0x0800	1514	IPv4
12	02:18:10.233787	Apple_ef:3c:a0	Technico_57:1e:73	0x0800	66	IPv4
13	02:18:10.234242	Technico_57:1e:73	Apple_ef:3c:a0	0x0800	1514	IPv4
14	02:18:10.234247	Technico_57:1e:73	Apple_ef:3c:a0	0x0800	585	IPv4
15	02:18:10.234315	Apple_ef:3c:a0	Technico_57:1e:73	0x0800	66	IPv4

```
▶ Frame 7: 410 bytes on wire (3280 bits), 410 bytes captured (3280 bits) on interface 0
▶ Ethernet II, Src: Apple_ef:3c:a0 (78:31:c1:ef:3c:a0), Dst: Technico_57:1e:73 (44:32:c8:57:1e:73)
▼ Data (396 bytes)
    Data: 4500018c2f5a400040068a810a00000d8077f50cc3490050...
    [Length: 396]
```

```
0000  44 32 c8 57 1e 73 78 31  c1 ef 3c a0 08 00 45 00   D2.W.sx1 ..<...E.
0010  01 8c 2f 5a 40 00 40 06  8a 81 0a 00 00 0d 80 77   ../Z@.@. .......w
0020  f5 0c c3 49 00 50 76 cb  d8 84 cf 8c 0b 30 80 18   ...I.Pv. .....0..
0030  10 15 cc cc 00 00 01 01  08 0a 04 03 0d 15 9f 6e   ........ .......n
0040  b6 59 47 45 54 20 2f 77  69 72 65 73 68 61 72 6b   .YGET /w ireshark
0050  2d 6c 61 62 73 2f 48 54  54 50 2d 65 74 68 65 72   -labs/HT TP-ether
0060  65 61 6c 2d 6c 61 62 2d  66 69 6c 65 33 2e 68 74   eal-lab- file3.ht
0070  6d 6c 20 48 54 54 50 2f  31 2e 31 0d 0a 48 6f 73   ml HTTP/ 1.1..Hos
0080  74 3a 20 67 61 69 61 2e  63 73 2e 75 6d 61 73 73   t: gaia. cs.umass
0090  2e 65 64 75 0d 0a 55 73  65 72 2d 41 67 65 6e 74   .edu..Us er-Agent
00a0  3a 20 4d 6f 7a 69 6c 6c  61 2f 35 2e 30 20 28 4d   : Mozill a/5.0 (M
00b0  61 63 69 6e 74 6f 73 68  3b 20 49 6e 74 65 6c 20   acintosh ; Intel 
00c0  4d 61 63 20 4f 53 20 58  20 31 30 2e 31 30 3b 20   Mac OS X  10.10; 
00d0  72 76 3a 34 35 2e 30 29  20 47 65 63 6b 6f 2f 32   rv:45.0)  Gecko/2
00e0  30 31 30 30 31 30 31 20  46 69 72 65 66 6f 78 2f   0100101  Firefox/
00f0  34 35 2e 30 0d 0a 41 63  63 65 70 74 3a 20 74 65   45.0..Ac cept: te
0100  78 74 2f 68 74 6d 6c 2c  61 70 70 6c 69 63 61 74   xt/html, applicat
0110  69 6f 6e 2f 78 68 74 6d  6c 2b 78 6d 6c 2c 61 70   ion/xhtm l+xml,ap
0120  70 6c 69 63 61 74 69 6f  6e 2f 78 6d 6c 3b 71 3d   plicatio n/xml;q=
0130  30 2e 39 2c 2a 2f 2a 3b  71 3d 30 2e 38 0d 0a 41   0.9,*/*; q=0.8..A
0140  63 63 65 70 74 2d 4c 61  6e 67 75 61 67 65 3a 20   ccept-La nguage: 
0150  65 6e 2d 55 53 2c 65 6e  3b 71 3d 30 2e 35 0d 0a   en-US,en ;q=0.5..
0160  41 63 63 65 70 74 2d 45  6e 63 6f 64 69 6e 67 3a   Accept-E ncoding:
0170  20 67 7a 69 70 2c 20 64  65 66 6c 61 74 65 0d 0a    gzip, d eflate..
0180  43 6f 6e 6e 65 63 74 69  6f 6e 3a 20 6b 65 65 70   Connecti on: keep
0190  2d 61 6c 69 76 65 0d 0a  0d 0a                     -alive.. ..
```

```
▶ Frame 7: 410 bytes on wire (3280 bits), 410 bytes captured (3280 bits) on interface 0
▼ Ethernet II, Src: Apple_ef:3c:a0 (78:31:c1:ef:3c:a0), Dst: Technico_57:1e:73 (44:32:c8:57:1e:73)
  ▶ Destination: Technico_57:1e:73 (44:32:c8:57:1e:73)
  ▶ Source: Apple_ef:3c:a0 (78:31:c1:ef:3c:a0)
    Type: IPv4 (0x0800)
▶ Data (396 bytes)
```

Source: `78:31:c1:ef:3c:a0`

2.

Destination: `44:32:c8:57:1e:73`

No, it is the address of the Router.

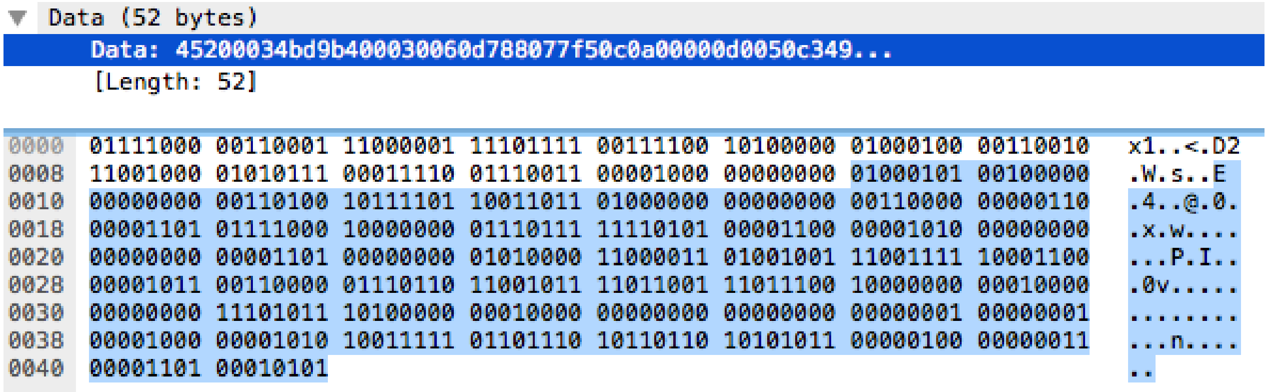
3. `0x0800`
4. Around 41 bytes from the start.
5.

No.	Time	Source	Destination	Protocol	Length	Info
3	02:18:08.343172	Apple_72:8f:d8	Broadcast	0x0800	110	IPv4
4	02:18:10.099363	Apple_ef:3c:a0	Technico_57:1e:73	0x0800	78	IPv4
5	02:18:10.179861	Technico_57:1e:73	Apple_ef:3c:a0	0x0800	74	IPv4
6	02:18:10.179968	Apple_ef:3c:a0	Technico_57:1e:73	0x0800	66	IPv4
7	02:18:10.180358	Apple_ef:3c:a0	Technico_57:1e:73	0x0800	410	IPv4
8	02:18:10.233067	Technico_57:1e:73	Apple_ef:3c:a0	0x0800	66	IPv4
9	02:18:10.233596	Technico_57:1e:73	Apple_ef:3c:a0	0x0800	66	IPv4
10	02:18:10.233606	Technico_57:1e:73	Apple_ef:3c:a0	0x0800	1514	IPv4
11	02:18:10.233608	Technico_57:1e:73	Apple_ef:3c:a0	0x0800	1514	IPv4

```
    Capture Length: 66 bytes (528 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:data]
▼ Ethernet II, Src: Technico_57:1e:73 (44:32:c8:57:1e:73), Dst: Apple_ef:3c:a0 (78:31:c1:ef:3c:a0)
  ▶ Destination: Apple_ef:3c:a0 (78:31:c1:ef:3c:a0)
  ▶ Source: Technico_57:1e:73 (44:32:c8:57:1e:73)
    Type: IPv4 (0x0800)
▼ Data (52 bytes)
    Data: 45200034bd9b400030060d788077f50c0a00000d0050c349...
    [Length: 52]

0000  01111000 00110001 11000001 11101111 00111100 10100000 01000100 00110010   x1..<.D2
0008  11001000 01010111 00011110 01110011 00001000 00000000 01000101 00100000   .W.s..E
0010  00000000 00110100 10111101 10011011 01000000 00000000 00110000 00000110   .4..@.0.
0018  00001101 01111000 10000000 01110111 11110101 00001100 00001010 00000000   .x.w....
0020  00000000 00001101 00000000 01010000 11000011 01001001 11001111 10001100   ...P.I..
0028  00001011 00110000 01110110 11001011 11011001 11011100 10000000 00010000   .0v.....
0030  00000000 11101011 10100000 00010000 00000000 00000000 00000001 00000001   ........
0038  00001000 00001010 10011111 01101110 10110110 10101011 00000100 00000011   ...n....
0040  00001101 00010101                                                         ..
```

`44:32:c8:57:1e:73`, address of my router.

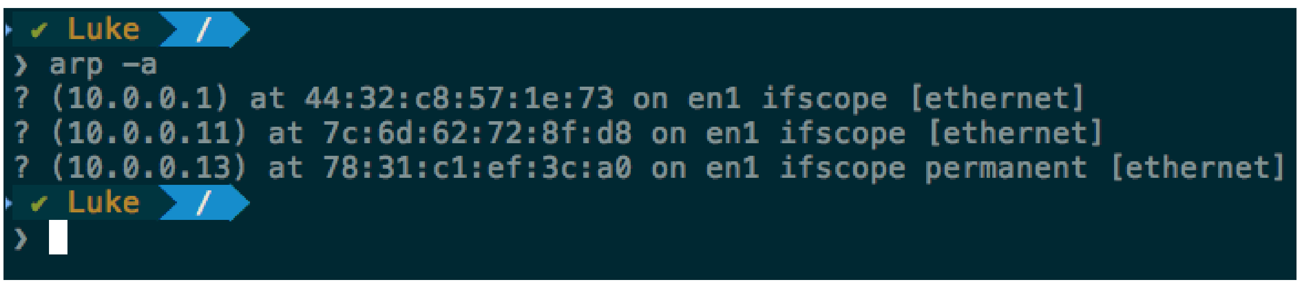
6. `78:31:c1:ef:3c:a0` address of my computer.
7. `0x0800` (IPv4)
8.



52 bytes

ARP Caching

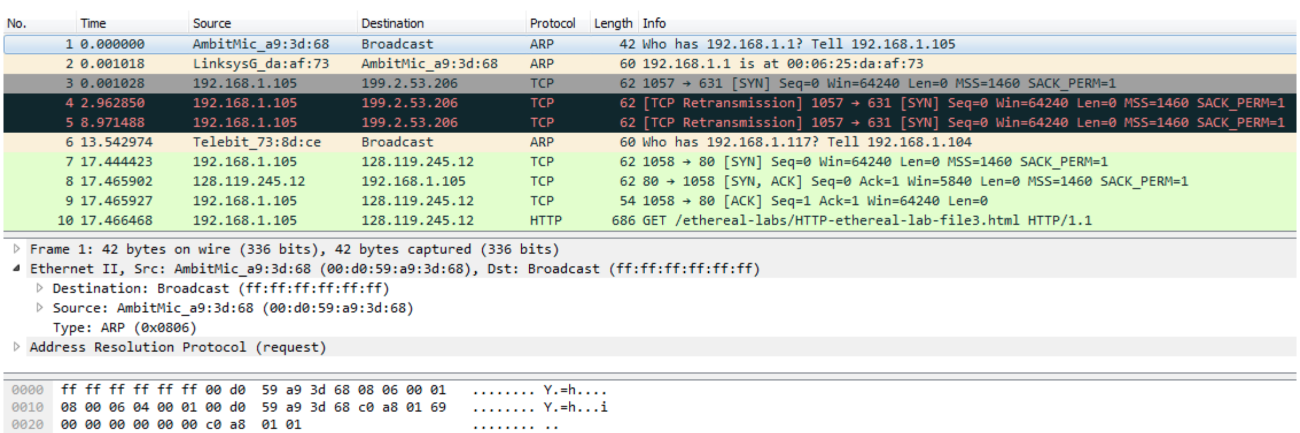
9.



IP address, Physical (MAC Address), port, interface scope, status (stale or not) and type (ethernet)

ARP in Action (from ethereal-trace)

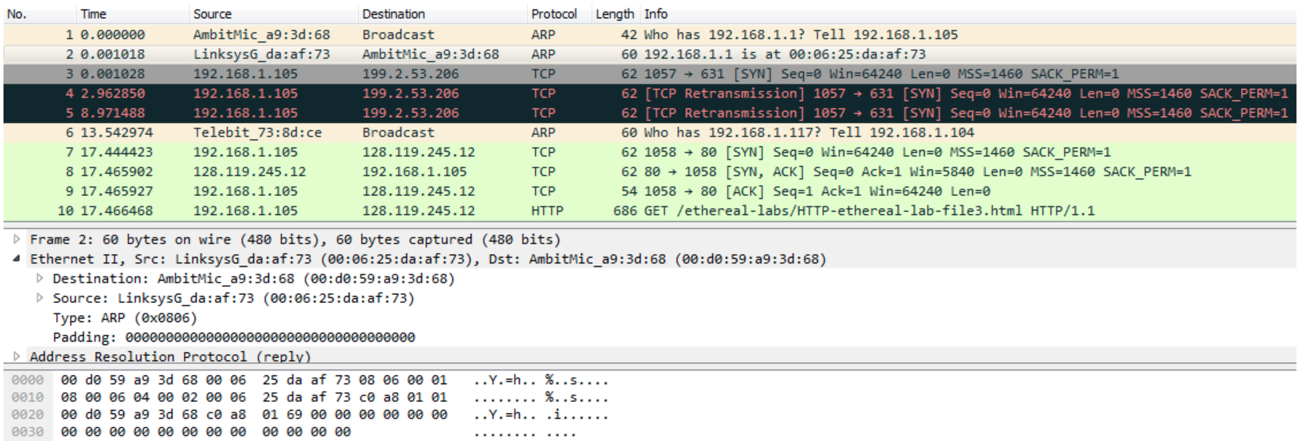
10.



Source: `00:d0:59:a9:3d:68`

Destination: `ff:ff:ff:ff:ff:ff`

11. `0x0806`, ARP
12. a) ARP opcode field begins 20 bytes from the beginning of the Ethernet frame.
    b) The hex value for opcode within the ARP-payload of the request is `0x0001` for request.
    c) Yes. The ARP message contains `192.168.1.105` for sender.
    d) The field "Target MAC address" set to `00:00:00:00:00:00` to question the machine whose corresponding IP address (192.168.1.1) is queried.
13.



a) 20 bytes from the beginning of the Ethernet frame.
b) reply (2), `0x0002`
c) Sender MAC Address field containing the Ethernet address `00:06:25:da:af:73` for sender with IP: `192.186.1.1`

14.

Source: `00:06:25:da:af:73`
Destination: `00:d0:59:a9:3d:68`

15.

There is no reply because the machine that captured this trace is not the machine that sent the request. The ARP request is broadcasted. However, ARP reply is sent back directly to sender's Ethernet address.

Protocol List:

> Bob connects to google.com via school's network.

- ARP so that his computer can be recognized within the local area network (using MAC address).
- DHCP provides a local IP address to his computer.
- NAT protocol allows his LAN IP to communicate with public internet IPs.
- DNS (via UDP messages) so he can convert `google.com` into an IP address.
- HTTP (via TCP) so that the page can be sent to the user.
- TCP messages are encapsulated in TCP packets and further on into IP packets.
- IP packets should follow routing protocols (RIP or OSPF) to eventually arrive at the router in an efficient manner.

Review Questions:

Chapter 5:
~~R3~~, ~~R5~~, ~~R8~~, ~~R9~~, ~~R14~~

R3:

- **Framing:** Each datagram received from the network layer is encapsulated in a frame by the Link-layer.
- **Link Access:** Link Layer specifies MAC protocol required for successful connection in case of multiple nodes using the same link.
- **Reliable Delivery:** Link-layer is responsible for having the network-layer datagram is delivered across the link without any errors.
- **Error detection and correction:** Link layer protocol is equipped with bit error detection potentially present in the frame. Link-layer is also able to correct such errors.

Corresponding IP services:
Framing, Link Access and Error detection and correction.

Corresponding TCP services:
Framing, Link Access, Reliable Delivery, Error detection and correction.

R5:
Slotted ALOHA:

1. Slotted ALOHA has a node to transmit continuously at maximum rate, all the time.

2. In slotted ALOHAm each of the nodes has a $\frac{R}{Mbps}$ throughput. Average transmission rate of $\frac{R}{M}$ for each node.
3. Each node detects collision and decides when to transmit independently (no clock). Partially decentralized.
4. Simple, efficient.

Token Passing:

1. Always has a node to transmit at Rbps rate.
2. Each node has throughput of $\frac{R}{Mbps}$. Average transmission rate of $\frac{R}{M}$ for each node.
3. Decentralized.
4. Simple, inexpensive.

R8:
In token-ring, a node can only send the frame when it has the token. In a large lan perimeter, each node will have to wait longer for its turn. Each node has to wait until its frame propagates around the entire ring before passing the token to the next node. Therefore, token-ring is an inefficient protocol when the LAN has a large perimeter.

Problems:

~~P1~~, ~~P5~~, ~~P11~~, ~~P31~~, ~~P32~~

P1:
Bit pattern: `1110 0110 1001 1101`
Parity: Even

1110	1
0110	0
1001	0
1101	1

We organize the bits from last table in columns and calculate the parity vertically:

1	1	1	0
0	1	1	0
1	0	0	1
1	1	0	1
parity: 1	parity: 1	parity: 0	parity: 0

Resulting in:
`1100`, which has horizontal (row) parity of: `0`

For the final result, we interleave the original data with its row parity.

`1110 1 0110 0 1001 0 1101 1`

Now we add the resulting value from the column parity calculation and again interleave with its row parity, leaving us with the final result of:

`1110 1 0110 0 1001 0 1101 1 1100 0`

P5:

$G = 10011$
$D = 1010101010$

We start by rewriting G in terms of a polynomial expression:

$$= (x^4 \cdot 1) + (x^3 \cdot 0) + (x^2 \cdot 0) + (x^1 \cdot 1) + (x^0 \cdot 1)$$

$$= (x^4 + x^1 + 1)$$

$G(x) = x^4 + x^1 + 1$
The degree of the expression is 4, therefore our $r = 4$
So, we will append 4 `0s` to D

$D = 1010101010$
$D + r = 1010101010\ 0000$

To calculate the value of R:

$$R = \frac{D + r}{G}$$

- CRC operation used for division is the XOR operator
- XOR operations results in 0 when both the bits are equal; 1 otherwise.
  We divide

$$\frac{10101010100000}{10011}$$

```
          1011011100
      ----------------
10011 ) 10101010100000
        10011
        -----
         0110010100000
          10011
          -----
           10100100000
           10011
           -----
            0111100000
             10011
             -----
              11010000
              10011
              -----
               1001000
               10011
               -----
                000100
```

$R = 0100$

P11:

number of packets at each node $= \infty$
probability required by each node to transmit packet in each slot is p
Assuming probability of success $= p$
number of failures $q = 1 - p$

Probability of A succeeding in a slot is:

$P(A) = (A\, transmits)(B\, not)(C\, not)(D\, not)$
$P(A) = p \cdot (1 - p) \cdot (1 - p) \cdot (1 - p)$
$P(A) = p(1 - p)^3$

a:

Probability of A succeeding for the first time in slot 5:

$$P = P(A\, fails\, slot\, 1) \cdot (A\, fails\, slot\, 2) \cdot (A\, fails\, slot\, 3) \cdot (A\, succeeds\, slot\, 5)$$
$$P = (1 - P(A))^4 \cdot P(A)$$
$$P = (1 - (p(1-p)^3))^4 \cdot p(1-p)^3$$

b:

Probability of A succeeding in a slot 4:

$$P(A) = (A\, not)(A\, not)(A\, not)(A\, succeeds)$$
$$P(A) = p \cdot (1-p) \cdot (1-p) \cdot (1-p)$$
$$P(A) = p \cdot (1-p)^3$$

Similarly;

Probability of B succeeding in slot 4:

$$P(B) = p \cdot (1-p)^3$$

Probability of C succeeding in slot 4:

$$P(C) = p \cdot (1-p)^3$$

Probability of D succeeding in slot 4:

$$P(C) = p \cdot (1-p)^3$$

b. *Probability of any node succeeding in slot 4:*

Since the nodes are mutually exclusive;

$$= [p \cdot (1-p)^3] + [p \cdot (1-p)^3] + [p \cdot (1-p)^3] + [p \cdot (1-p)^3]$$
$$= 4 \cdot p \cdot (1-p)^3$$

c.

Probability that any node succeed in a slot:

$$4p(1-p)^3$$

Probability that any node does not succeed in a slot:

$$1 - 4p(1-p)^3$$

Probability of success is first for slot 3 is:

$$P = P(fails\, first) \cdot P(fails\, second) \cdot P(success\, third)$$
$$P = (1 - (4p(1-p)^3)) \cdot (1 - (4p(1-p)^3)) \cdot (4p(1-p)^3)$$
$$P = (1 - (4p(1-p)^3))^2 (4p(1-p)^3)$$

$P = 1 - (4p(1-p)^3)^2(4p(1-p)^3)$

d.

Efficiency of the 4 node system:

Efficiency can be described as the probability of any node succeeding in a slot:
$P = 4p(1-p)^3$

P31:

1. Connect PC to the network using Ethernet Interface
2. DHCP provides an IP address to the PC (steps follow):
   1… PC creates an IP datagram with dest: `255.255.255.255` in DHCP's server discovery step.
   2… Datagram is placed in an Ethernet frame, sent and the router broadcasts it to the network.
   3… DHCP server residing in the DHCP provides the PC with a list of addresses of the routes with one hop, as well as subnet mask and subnet where the PC resides. Also a DNS server if it exists.
3. The ARP cache for the PC is starts empty. ARP protocol is used by the PC in order to obtain MAC address of first-hop routers and local DNS server.
4. PC first obtains IP address of the webpage requested. If local DNS server does not have it, DNS protocol is used so the computer can obtain the appropriate IP address.
5. Once this IP is obtained, PC will send an HTTP request using the first-hop router.
6. The PC then sends the Ethernet frames to the router.
7. The PC sends Ethernet frames destined to the router.
8. Upon receiving, the first-hop router passes the frames up to the IP layer, and checks its routing table. The router sends the packets to the right interface.
9. IP packets are routed through the internet until they arrive at the Web server.
10. Server hosting the Web page will send back the Web page to the PC using HTTP messages.
11. Such TCP messages are then encapsulated into TCP packets and further into IP packets.
12. IP packets then follow IP routes and eventually reach first-hop router.
13. Router forwards these IP packets to the user PC by encapsulating them into Ethernet frames.

P32:

Number of flow pairs $= 80$
Capacity of each link $= 10\,Gbps$
Capacity of the link between TOR switches and hosts $= 1\,Gbps$

Each flow traversing over a same link shares the capacity of the link with other flows because each link is shared.

Maximum flow rate determines link capacity required for each flow in the network.

a.
Maximum rate of flow:

$$= \frac{Link\ Cap}{Number\ of\ flow\ pairs}$$

$[1\,Gbps = 1000\,Mbps]$

$= \frac{10\,Gbps}{80} = \frac{10{,}000\,Mbps}{80} = 125\,Mbps$

b.
Highly interconnected topology:

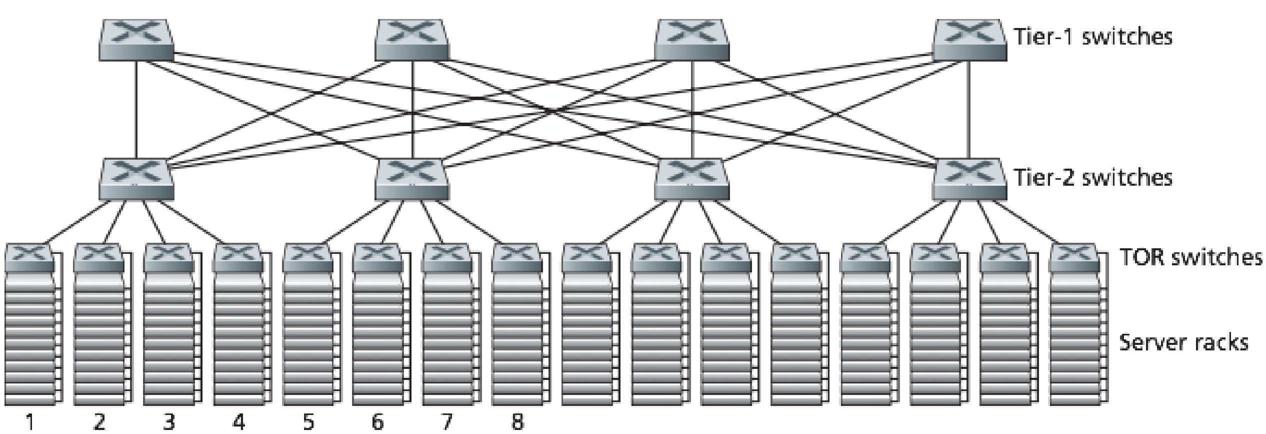


Figure 5.31 ♦ Highly-interconnected data network topology

Every switch in Tier-1 has connection with every switch in Tier-2. Since we only have 4 hosts involved in the network on each host, number of paths between a tier-1 to tier-2 switches is 4.

Maximum rate of flow determines the capacity at which all the data paths are transmitting.

Number of paths $= 4$
Capacity of each link $= 10\,Gbps$
Maximum rate of flow $= 4 \cdot 10\,Gbps = 40\,Gbps$

c.
Data center network using Hierarchical topology:

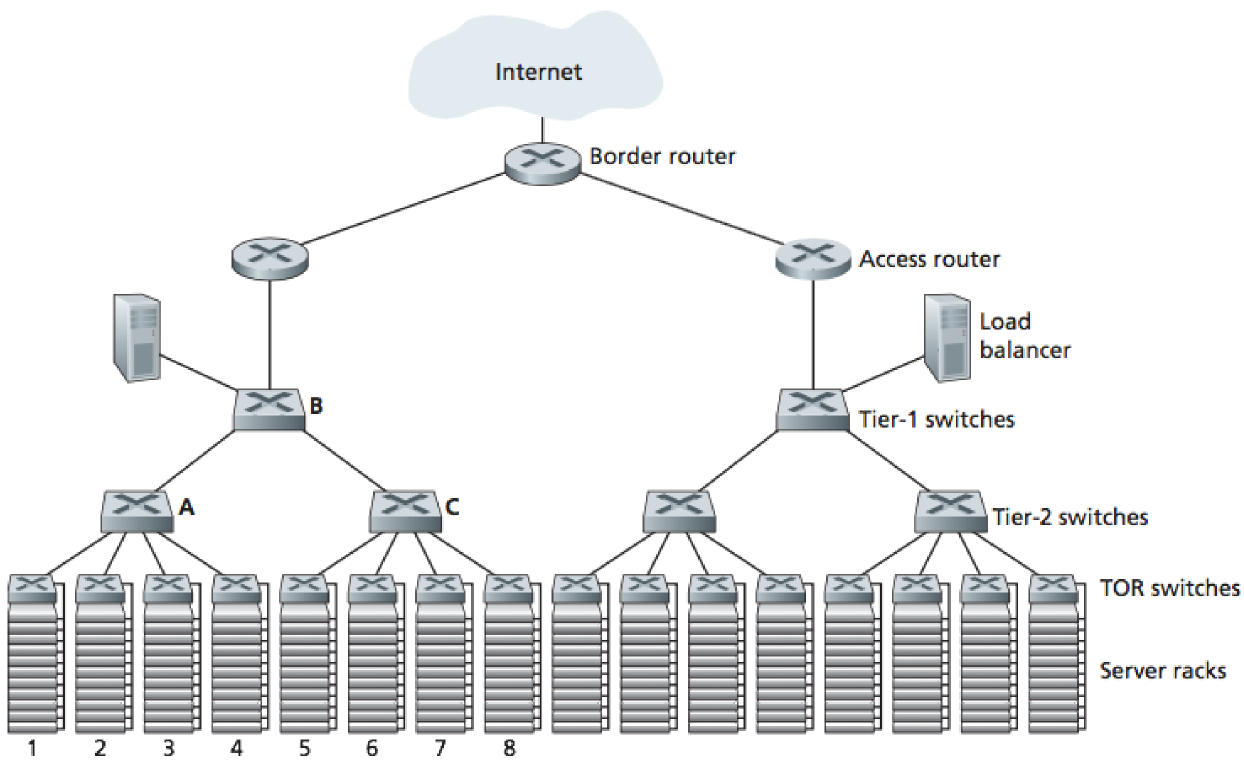


Figure 5.30 ♦ A data center network with a hierarchical topology

Number of flow pairs $= 160$
Capacity of each link $= 10\,Gbps$

Maximum rate of flow $= \frac{10\,Gbps}{160}$
$= \frac{10{,}000\,Mbps}{160} = 62.5\,Mbps$

Highly interconnected network:

Every switch in Tier-1 has a connection to every switch in Tier-2. The number of paths from a tier-1 switch to tier-2 switches is 20 because 20 hosts are involved in the network.

Number of paths $= 20$

Capacity of each link $= 10\,Gbps$
Maximum rate of flow $= 20 \cdot 10\,Gbps = 200\,Gbps$

With 160 flow pairs, the maximum rate of flow is $200\,Gbps$