network hw4
dr. arshadi

aeirya mohammadi
97103779

سوالات نظری

1]

a)

Dest Addr Range				Link Interface
11100000	00xxxxxx	xxxxxxxx	xxxxxxxx	0
11100000	01000000	xxxxxxxx	xxxxxxxx	1
1110000x	xxxxxxxx	xxxxxxxx	xxxxxxxx	2
11100001	1xxxxxxx	xxxxxxxx	xxxxxxxx	3
otherwise				3

b)

Address				Link Interface
11001000	10010001	01010001	01010101	3
11100001	01000000	11000011	00111100	1
11100001	10000000	00010001	01110111	3

2]

Subnet+Host				[Sub]network
10000000	01110111	00101000	0000000	All
10000000	01110111	00101000	0000000	A
10000000	01110111	00101001	0000000	B
10000000	01110111	00101001	1000000	C
10000000	01110111	00101001	1111100	D
10000000	01110111	00101001	1111110	E

Net ID	Sub-network
128.119.40.0/24	A
128.119.41.0/25	B
128.119.41.127/26	C
128.119.41.252/31	D
128.119.41.254/31	E

3]

a) the packet splits into ceil[(14000-20)/(3300-20)] = 5 parts, each of maximum size 3300 which are small enough to be also passed by the second router without fragmentation.

total payload: 14000 - 20 = 13980
# of parts = ceil(13980/3280) = 5
last frag size = 13980 - 3280 * 4 + 20 = 860 + 20 (overhead) = 880
4 fragment of size 3300, 1 of size 880

5 fragments exiting router 2

b) the first router splits the packet to packets of size 4480 (plus 20 overhead of header), making ceil((14000-20)/4500) = 4 packets of the following sizes:

last datagram fragment size = 14000 - 20 - 3*(4500-20) + 20 = 560
3 of size 4500, 1 of size 560

the former 3 will be dividend again at the router two. into fragments of following size:

packet 1 = 3300
packet 2 = (4500 - 20) - (3300 - 20) + 20 = 1220

then the packets exiting router 2 will be:
1 of size 560, 3 of size 3300, 3 of size 1220
adding up to 7

4]

350 - 20 (header) = 330 Bytes for payload
nearest number divisible by 8 -> 328

#. of fragments = 900 / 328= 2.74 -> 3

last frag size = 900 - 2*328 + 20 = 244 + 20

part 1 offset = 328/8 = 41

frag #.	1	2	3
length	348	348	264
frag flag	1	1	0
offset	0	41	82

5]

192.168.1.192/28 <-> 11111111 11111111 11111111 11110000 (subnet mask)

NAT translation table

WAN side	LAN side
129.119.112.235: 5010	192.168.1.192: 3000
129.119.112.235: 5011	192.168.1.192: 3001
129.119.112.235: 5020	192.168.1.193: 3000
129.119.112.235: 5021	192.168.1.193: 3001
129.119.112.235: 5030	192.168.1.194: 4000
129.119.112.235: 5031	192.168.1.194: 4001

a sample packet P entering the router from internet:

source ip	source port	destination ip	destination port
176.213.40.12	80	129.119.112.235	5010

P will be forwarded to host A (listening at 192.168.1.192: 3000)

6]

a) not necessarily. there's a chance the broadcast doesn't reach some nodes (and is lost). even if it does, there's always some delay and until the state is updated, packets are sent to "wrong" routers.

b) still some are lost. so the routes aren't **always** correct but most of the time they are. there's still delay.

c) the bigger the network gets there's more chance flooding causes heavy traffic and infinite looping so we can't really have such network (with no loss) but if we did, then this protocol works more correctly but still some bad routing could happen because of change of link states (change of costs/broken links) as we're broadcasting the previous change. but if the broadcast happens immediately (and without other change of network) this would be a usable protocol and there will probably be no need to rebroadcast states.

سوالات عملی

1- IPv4

1]

source ip: 192.168.43.29

2]

TTL: 1
protocol: UDP (17)

3]

header len: 20 bytes
payload len: 56-20 = 36 bytes

4]

no, it is exactly 56 bytes (datagram size)

6]

TTL changes (incrementally) and that's to let the packet traverse further into internet (hop one more router each time). Dest port may vary. Identification value and checksum (obviously) also differ. Other labels stay the same since all sent packets (which are of the same size and content) are supposed to reach 'sharif.ir'.

7]

starting from 50889 and going up by one for each UDP packet sent.

9]

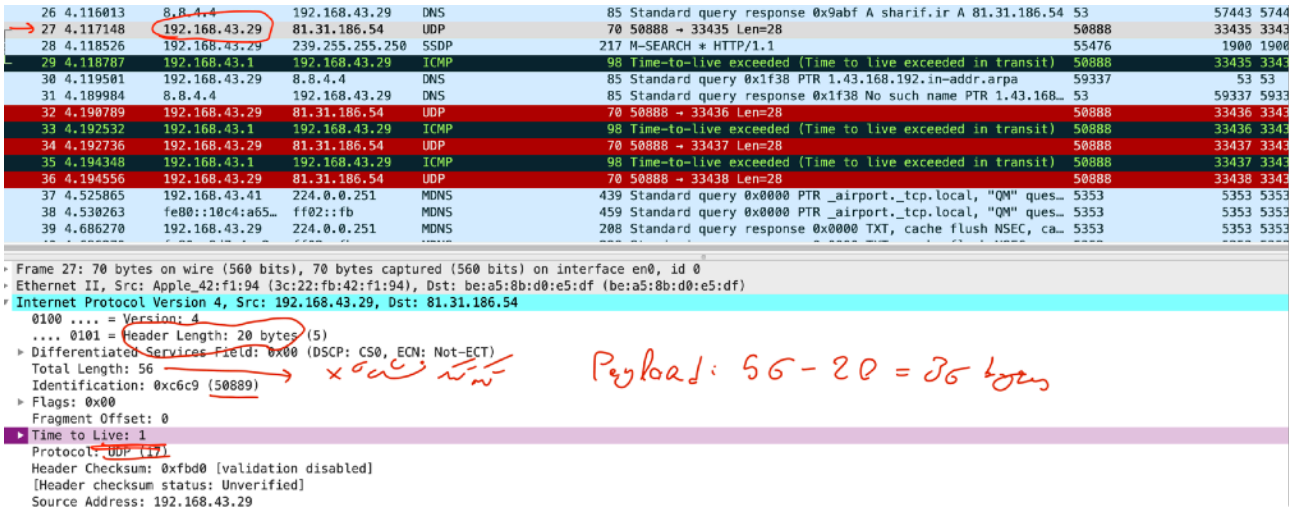
protocol: ICMP (1)

10]

no, since they are probably sent by different routers.

11]

no, neither they share the same identification as the sent datagram nor they necessarily follow up each other numerically (though it's increasing, but not by one).



2- fragmentation

12]

datagram length: 1500

13]

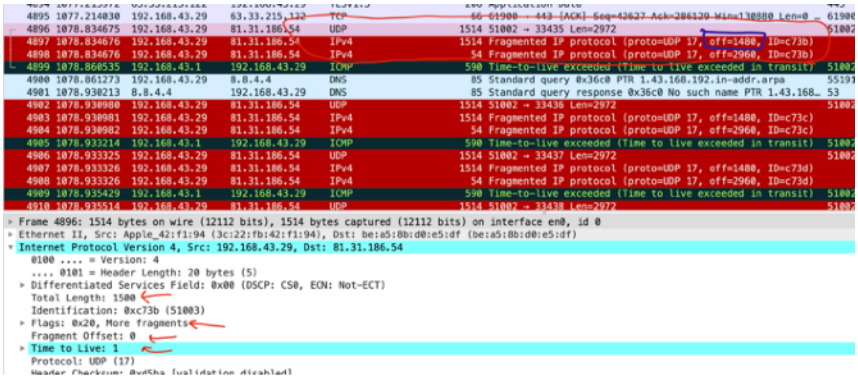
“more fragments” flag

14]

fragment offset being 0

15]

1500 bytes



16]

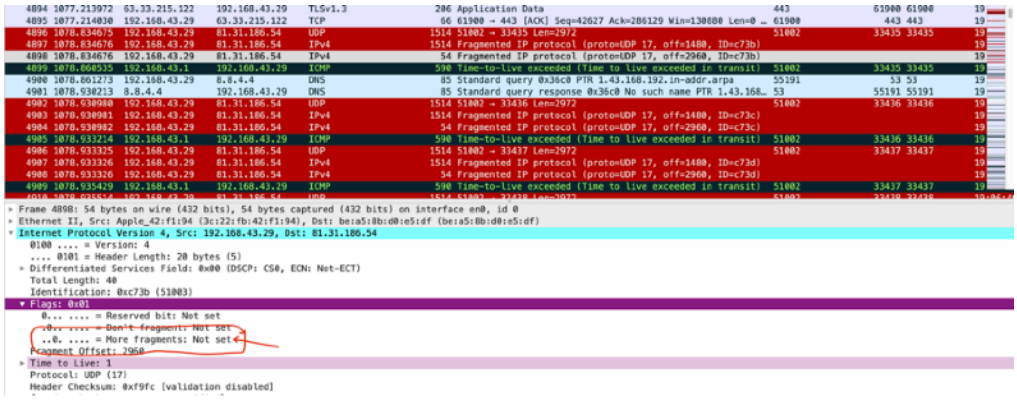
fragment offset = 1480 > 0
(which means the first byte of data is not the first byte of the assembled packet’s payload)

17]

fragment offset and header checksum

18]

because of “more fragments" flag being equal to “not set”.



3- IPv6

19]

src addr: 2601:193:8302:4620:215c:f5ae:8b40:a27a
dest addr: 2001:558:feed::1

20]

flow label: 0x00063ed0

21]

payload len: 37B

22]

user datagram protocol (UDP)

```
▸ Frame 20: 91 bytes on wire (728 bits), 91 bytes captured (728 bits) on interface en0, id 0
▸ Ethernet II, Src: Apple_98:d9:27 (78:4f:43:98:d9:27), Dst: Technico_81:74:5a (44:1c:12:81:74:5a)
▾ Internet Protocol Version 6, Src: 2601:193:8302:4620:215c:f5ae:8b40:a27a, Dst: 2001:558:feed::1
    0110 .... = Version: 6
  ▸ .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... .... .... 0110 0011 1110 1101 0000 = Flow Label: 0x63ed0
    Payload Length: 37
    Next Header: UDP (17)
    Hop Limit: 255
    Source Address: 2601:193:8302:4620:215c:f5ae:8b40:a27a
    Destination Address: 2001:558:feed::1
▸ User Datagram Protocol, Src Port: 64430, Dst Port: 53
▸ Domain Name System (query)
```

24]

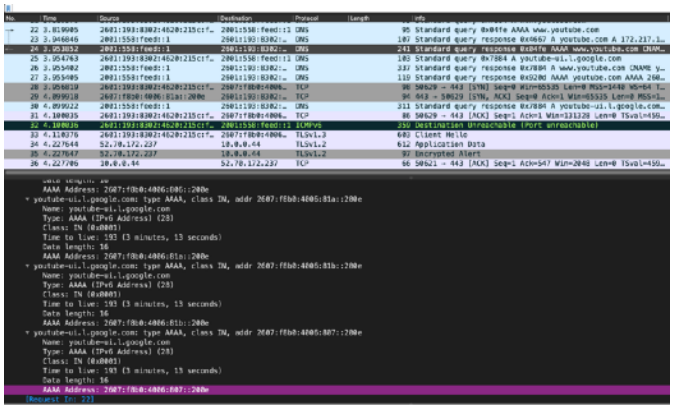
one

25]

2607:f8b0:4006:815::200e

```
▾ Queries
  ▾ youtube.com: type AAAA, class IN
      Name: youtube.com
      [Name Length: 11]
      [Label Count: 2]
      Type: AAAA (IPv6 Address) (28)
      Class: IN (0x0001)
▾ Answers
  ▾ youtube.com: type AAAA, class IN, addr 2607:f8b0:4006:815::200e
      Name: youtube.com
      Type: AAAA (IPv6 Address) (28)
      Class: IN (0x0001)
      Time to live: 201 (3 minutes, 21 seconds)
      Data length: 16
      AAAA Address: 2607:f8b0:4006:815::200e
    [Request In: 20]
    [Time: 0.140916000 seconds]
```

there are also other queries which they had received more answers. pictures of one is shown below:



4- ICMP, Ping

26]

source ip: 192.168.43.29 dest ip: 81.31.186.54

27]

since ICMP is a network layer protocol and unlike TCP and UDP it doesn't need ports to functions.

28]

request:

type: 8 code: 0

field	type	code	checksum	identifier	seq. #	timestamp
size(B)	1	1	2	2	2	8

29]

reply:
type: 0 code: 0

not only the reply packet's ICMP header shares the same fields as the request but it also has the same field values (except for type). meaning that the only difference between ICMP header of a request message and its reply would be the "type" value (either 0 or 8) and checksum.