

CS 224(M): Tutorial 9

ARP, ICMP, NAT & IPv6

Name: Dhruv Ilesh Shah

Roll No.: 150070016

1.

96	24.05050100f	14:dd:a9:37:fe:8f	Broadcast	ARP	42	Who has 192.168.1.3? Tell 192.168.1.73
135	43.59041100f	ac:e0:10:e0:2f:3f	WibhuTec_dc:55:f0	ARP	42	Who has 192.168.1.3? Tell 192.168.1.19
136	43.59190600f	WibhuTec_dc:55:f0	ac:e0:10:e0:2f:3f	ARP	42	192.168.1.3 is at 00:11:74:dc:55:f0
145	49.59656200f	WibhuTec_dc:55:f0	ac:e0:10:e0:2f:3f	ARP	42	Who has 192.168.1.19? Tell 192.168.1.3
146	49.59659000f	ac:e0:10:e0:2f:3f	WibhuTec_dc:55:f0	ARP	42	192.168.1.19 is at ac:e0:10:e0:2f:3f

2. Source MAC and Destination MAC can be seen on the left. IPs are a part of the message, seen on the right.

3.

dhruv @ {~} says \$ arp -n					
Address	HWtype	HWaddress	Flags	Mask	Iface
192.168.1.3	ether	00:11:74:dc:55:f0	C		wlan0

4.

The ARP packets are generated again, after the arp -d.

5.

2090	14.28318600f	ac:e0:10:e0:2f:3f	Broadcast	ARP	42	Who has 10.196.23.168? Tell 10.196.9.60
2105	14.37339500f	e4:f8:9c:e8:76:46	ac:e0:10:e0:2f:3f	ARP	42	10.196.23.168 is at e4:f8:9c:e8:76:46
2824	19.38877900f	e4:f8:9c:e8:76:46	ac:e0:10:e0:2f:3f	ARP	42	Who has 10.196.9.60? Tell 10.196.23.168
2825	19.38881600f	ac:e0:10:e0:2f:3f	e4:f8:9c:e8:76:46	ARP	42	10.196.9.60 is at ac:e0:10:e0:2f:3f
2106	14.37341000f	10.196.9.60	10.196.23.168	ICMP	98	Echo (ping) request id=0x4f19, seq=1/256, ttl=64 (reply in 2107)
2107	14.37564500f	10.196.23.168	10.196.9.60	ICMP	98	Echo (ping) reply id=0x4f19, seq=1/256, ttl=64 (request in 2106)
2236	15.28499500f	10.196.9.60	10.196.23.168	ICMP	98	Echo (ping) request id=0x4f19, seq=2/512, ttl=64 (reply in 2248)
2248	15.38540400f	10.196.23.168	10.196.9.60	ICMP	98	Echo (ping) reply id=0x4f19, seq=2/512, ttl=64 (request in 2236)
2345	16.28669700f	10.196.9.60	10.196.23.168	ICMP	98	Echo (ping) request id=0x4f19, seq=3/768, ttl=64 (reply in 2349)
2349	16.29119200f	10.196.23.168	10.196.9.60	ICMP	98	Echo (ping) reply id=0x4f19, seq=3/768, ttl=64 (request in 2345)
2511	17.28828100f	10.196.9.60	10.196.23.168	ICMP	98	Echo (ping) request id=0x4f19, seq=4/1024, ttl=64 (reply in 2512)
2512	17.29137500f	10.196.23.168	10.196.9.60	ICMP	98	Echo (ping) reply id=0x4f19, seq=4/1024, ttl=64 (request in 2511)

6.

```
dhruv @ {~} says $ ping 10.129.5.185
PING 10.129.5.185 (10.129.5.185) 56(84) bytes of data.
From 10.250.129.2 icmp_seq=5 Destination Host Unreachable
From 10.250.129.2 icmp_seq=10 Destination Host Unreachable
From 10.250.129.2 icmp_seq=15 Destination Host Unreachable
```

7.

38	3.996599000	ac:e0:10:e0:2f:3f	IntelCor_84:4e:23	ARP	42	Who has 10.42.0.66? Tell 10.42.0.1
39	3.997406000	IntelCor_84:4e:23	ac:e0:10:e0:2f:3f	ARP	42	10.42.0.66 is at d0:7e:35:84:4e:23

102	17.656121000	10.99.99.5	10.42.0.66	TCP	1514 [TCP segment of a reassembled PDU]
103	17.656409000	10.99.99.5	10.42.0.66	TCP	1514 [TCP segment of a reassembled PDU]
104	17.656413000	10.99.99.5	10.42.0.66	TCP	1514 [TCP segment of a reassembled PDU]
105	17.656638000	10.99.99.5	10.42.0.66	TCP	1514 [TCP segment of a reassembled PDU]
106	17.665832000	10.42.0.66	10.99.99.5	TCP	54 59197 > http [ACK] Seq=585 Ack=5841 Win=16384 Len=0
107	17.665906000	10.42.0.66	10.99.99.5	TCP	54 59197 > http [ACK] Seq=585 Ack=14601 Win=16384 Len=0
108	17.666652000	10.99.99.5	10.42.0.66	TCP	1514 [TCP segment of a reassembled PDU]
109	17.666687000	10.99.99.5	10.42.0.66	TCP	1514 [TCP segment of a reassembled PDU]
110	17.666709000	10.99.99.5	10.42.0.66	TCP	327 [TCP segment of a reassembled PDU]
111	17.668858000	10.42.0.66	10.99.99.5	TCP	54 59197 > http [ACK] Seq=585 Ack=17794 Win=16384 Len=0
112	17.682157000	10.99.99.5	10.42.0.66	TCP	1514 [TCP segment of a reassembled PDU]
113	17.682191000	10.99.99.5	10.42.0.66	TCP	1514 [TCP segment of a reassembled PDU]


```

Frame 113: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
Ethernet II, Src: ac:e0:10:e0:2f:3f (ac:e0:10:e0:2f:3f), Dst: IntelCor 84:4e:23 (d0:7e:35:84:4e:23)
Internet Protocol Version 4, Src: 10.99.99.5 (10.99.99.5), Dst: 10.42.0.66 (10.42.0.66)
Transmission Control Protocol, Src Port: http (80), Dst Port: 59197 (59197), Seq: 19254, Ack: 585, Len: 1460
  Source port: http (80)
  Destination port: 59197 (59197)
  [Stream index: 3]
  Sequence number: 19254 (relative sequence number)
  [Next sequence number: 20714 (relative sequence number)]
  Acknowledgment number: 585 (relative ack number)
  Header length: 20 bytes
  Flags: 0x010 (ACK)
  Window size value: 124
  [Calculated window size: 15872]
  [Window size scaling factor: 128]

```

(My machine was used as the NAT router. This is the traffic on my machine)

8.

IPv6 has a fixed header size of 40 bytes, unlike IPv4 which has a header of 20 bits + options. Since the given case has no options, and the total size is limited by the MTU, IPv4 would have more payload capacity than IPv6.

9.

IPv6 routers do not support fragmentation or the Don't Fragment option. For IPv6, Path MTU Discovery works by initially assuming the path MTU is the same as the MTU on the link layer interface where the traffic originates. Then, similar to IPv4, any device along the path whose MTU is smaller than the packet will drop the packet and send back an ICMPv6 Packet Too Big (Type 2) message containing its MTU, allowing the source host to reduce its Path MTU appropriately. The process is repeated until the MTU is small enough to traverse the entire path without fragmentation.