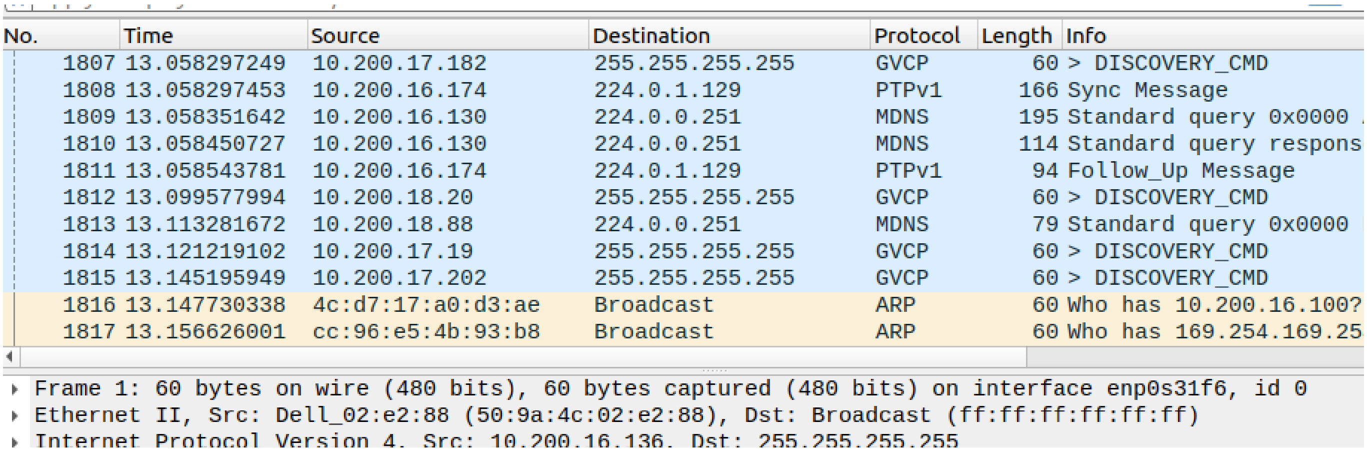
Traffic Capture

Q: What type of traffic has been captured?



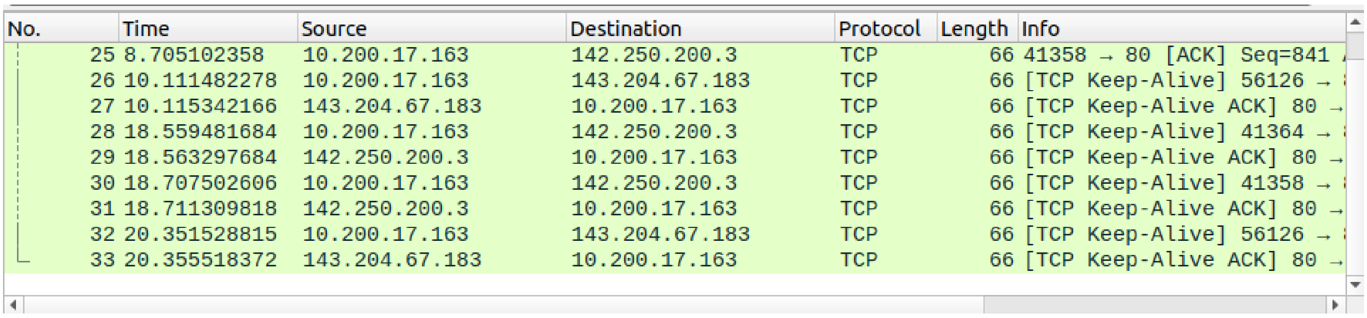
No.	Time	Source	Destination	Protocol	Length	Info
1807	13.058297249	10.200.17.182	255.255.255.255	GVCP	60	> DISCOVERY_CMD
1808	13.058297453	10.200.16.174	224.0.1.129	PTPv1	166	Sync Message
1809	13.058351642	10.200.16.130	224.0.0.251	MDNS	195	Standard query 0x0000
1810	13.058450727	10.200.16.130	224.0.0.251	MDNS	114	Standard query respons
1811	13.058543781	10.200.16.174	224.0.1.129	PTPv1	94	Follow_Up Message
1812	13.099577994	10.200.18.20	255.255.255.255	GVCP	60	> DISCOVERY_CMD
1813	13.113281672	10.200.18.88	224.0.0.251	MDNS	79	Standard query 0x0000
1814	13.121219102	10.200.17.19	255.255.255.255	GVCP	60	> DISCOVERY_CMD
1815	13.145195949	10.200.17.202	255.255.255.255	GVCP	60	> DISCOVERY_CMD
1816	13.147730338	4c:d7:17:a0:d3:ae	Broadcast	ARP	60	Who has 10.200.16.100?
1817	13.156626001	cc:96:e5:4b:93:b8	Broadcast	ARP	60	Who has 169.254.169.25

```
▸ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface enp0s31f6, id 0
▸ Ethernet II, Src: Dell_02:e2:88 (50:9a:4c:02:e2:88), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▸ Internet Protocol Version 4, Src: 10.200.16.136, Dst: 255.255.255.255
```

Figure 1: Departmental Computer Traffic

The types of traffic are various including ARP, HTTP, TCP, etc.

Q: How to capture http traffic?

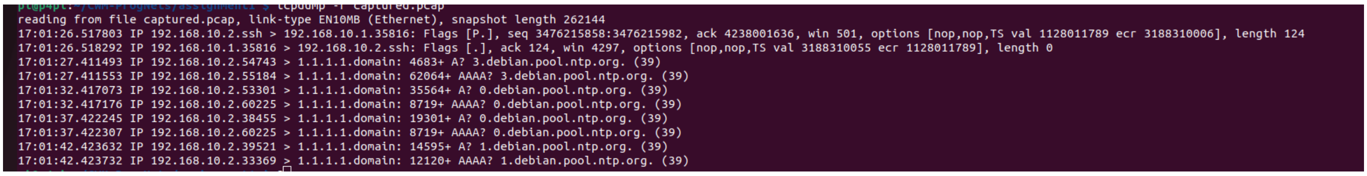


No.	Time	Source	Destination	Protocol	Length	Info
25	8.705102358	10.200.17.163	142.250.200.3	TCP	66	41358 → 80 [ACK] Seq=841
26	10.111482278	10.200.17.163	143.204.67.183	TCP	66	[TCP Keep-Alive] 56126 →
27	10.115342166	143.204.67.183	10.200.17.163	TCP	66	[TCP Keep-Alive ACK] 80 →
28	18.559481684	10.200.17.163	142.250.200.3	TCP	66	[TCP Keep-Alive] 41364 →
29	18.563297684	142.250.200.3	10.200.17.163	TCP	66	[TCP Keep-Alive ACK] 80 →
30	18.707502606	10.200.17.163	142.250.200.3	TCP	66	[TCP Keep-Alive] 41358 →
31	18.711309818	142.250.200.3	10.200.17.163	TCP	66	[TCP Keep-Alive ACK] 80 →
32	20.351528815	10.200.17.163	143.204.67.183	TCP	66	[TCP Keep-Alive] 56126 →
33	20.355518372	143.204.67.183	10.200.17.163	TCP	66	[TCP Keep-Alive ACK] 80 →

Figure 2: Departmental Computer HTTP Traffic

An http display filter with expression http had to be applied to shark.

Q: Rasberry Pi Capture?



```
pi@p4pi:~/CWM-ProgNets/assignment1 $ tcpdump -r captured.pcap
reading from file captured.pcap, link-type EN10MB (Ethernet), snapshot length 262144
17:01:26.517803 IP 192.168.10.2.ssh > 192.168.10.1.35816: Flags [P.], seq 3476215858:3476215982, ack 4238001636, win 501, options [nop,nop,TS val 1128011789 ecr 3188310006], length 124
17:01:26.518292 IP 192.168.10.1.35816 > 192.168.10.2.ssh: Flags [.], ack 124, win 4297, options [nop,nop,TS val 3188310055 ecr 1128011789], length 0
17:01:27.411493 IP 192.168.10.2.54743 > 1.1.1.1.domain: 4683+ A? 3.debian.pool.ntp.org. (39)
17:01:27.411553 IP 192.168.10.2.55184 > 1.1.1.1.domain: 62064+ AAAA? 3.debian.pool.ntp.org. (39)
17:01:32.417073 IP 192.168.10.2.53301 > 1.1.1.1.domain: 35564+ A? 0.debian.pool.ntp.org. (39)
17:01:32.417176 IP 192.168.10.2.60225 > 1.1.1.1.domain: 8719+ AAAA? 0.debian.pool.ntp.org. (39)
17:01:37.422245 IP 192.168.10.2.38455 > 1.1.1.1.domain: 19301+ A? 0.debian.pool.ntp.org. (39)
17:01:37.422307 IP 192.168.10.2.60225 > 1.1.1.1.domain: 8719+ AAAA? 0.debian.pool.ntp.org. (39)
17:01:42.423632 IP 192.168.10.2.39521 > 1.1.1.1.domain: 14595+ A? 1.debian.pool.ntp.org. (39)
17:01:42.423732 IP 192.168.10.2.33369 > 1.1.1.1.domain: 12120+ AAAA? 1.debian.pool.ntp.org. (39)
```

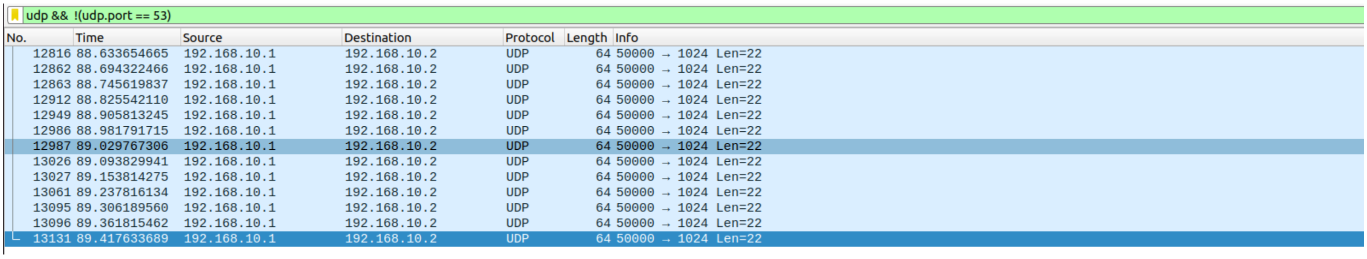
Figure 3: Rasberry Pi Traffic

Sending Traffic

Q: Can you define a filter to filter only the message you sent?

We observe that the traffic sent to the Rasberry Pi is all in UDP format. So this can be accomplished defining a filter with expression $$ udp && !(udp.port == 53)$$, with the second subformula to filter out DNS traffic which is also UDP protocol with port = 53.

Q: what is packet size and what format?

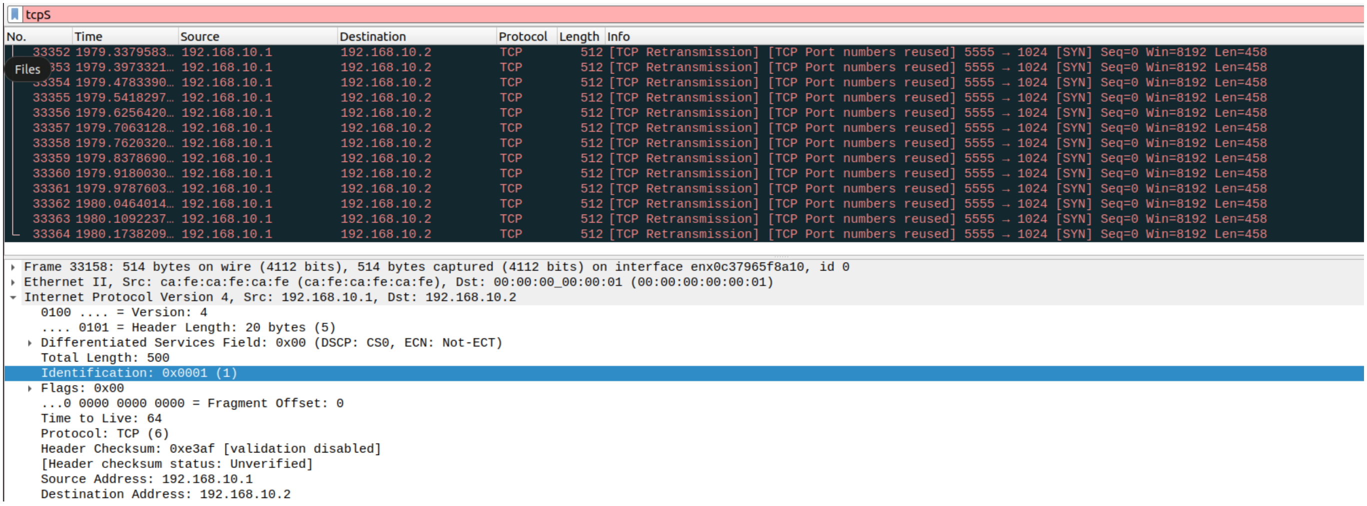


```
udp && !(udp.port == 53)
No.   Time         Source        Destination   Protocol Length Info
12816 88.633654665 192.168.10.1  192.168.10.2  UDP      64 50000 → 1024 Len=22
12862 88.694322466 192.168.10.1  192.168.10.2  UDP      64 50000 → 1024 Len=22
12863 88.745619837 192.168.10.1  192.168.10.2  UDP      64 50000 → 1024 Len=22
12912 88.825542110 192.168.10.1  192.168.10.2  UDP      64 50000 → 1024 Len=22
12949 88.905813245 192.168.10.1  192.168.10.2  UDP      64 50000 → 1024 Len=22
12986 88.981791715 192.168.10.1  192.168.10.2  UDP      64 50000 → 1024 Len=22
12987 89.029767306 192.168.10.1  192.168.10.2  UDP      64 50000 → 1024 Len=22
13026 89.093829941 192.168.10.1  192.168.10.2  UDP      64 50000 → 1024 Len=22
13027 89.153814275 192.168.10.1  192.168.10.2  UDP      64 50000 → 1024 Len=22
13061 89.237816134 192.168.10.1  192.168.10.2  UDP      64 50000 → 1024 Len=22
13095 89.306189560 192.168.10.1  192.168.10.2  UDP      64 50000 → 1024 Len=22
13096 89.361815462 192.168.10.1  192.168.10.2  UDP      64 50000 → 1024 Len=22
13131 89.417633689 192.168.10.1  192.168.10.2  UDP      64 50000 → 1024 Len=22
```

Figure 4: Sent Traffic Only

This type is UDP format. We see that although in the script the number of word is 22, but due to the header of the UDP format, we have total packet size of 64 bytes.

Q:Modify the Script?



```
tcpS
No.   Time            Source        Destination   Protocol Length Info
33352 1979.3379583…   192.168.10.1  192.168.10.2  TCP      512 [TCP Retransmission] [TCP Port numbers reused] 5555 → 1024 [SYN] Seq=0 Win=8192 Len=458
  353 1979.3973321…   192.168.10.1  192.168.10.2  TCP      512 [TCP Retransmission] [TCP Port numbers reused] 5555 → 1024 [SYN] Seq=0 Win=8192 Len=458
 3354 1979.4783390…   192.168.10.1  192.168.10.2  TCP      512 [TCP Retransmission] [TCP Port numbers reused] 5555 → 1024 [SYN] Seq=0 Win=8192 Len=458
33355 1979.5418297…   192.168.10.1  192.168.10.2  TCP      512 [TCP Retransmission] [TCP Port numbers reused] 5555 → 1024 [SYN] Seq=0 Win=8192 Len=458
33356 1979.6256420…   192.168.10.1  192.168.10.2  TCP      512 [TCP Retransmission] [TCP Port numbers reused] 5555 → 1024 [SYN] Seq=0 Win=8192 Len=458
33357 1979.7063128…   192.168.10.1  192.168.10.2  TCP      512 [TCP Retransmission] [TCP Port numbers reused] 5555 → 1024 [SYN] Seq=0 Win=8192 Len=458
33358 1979.7620320…   192.168.10.1  192.168.10.2  TCP      512 [TCP Retransmission] [TCP Port numbers reused] 5555 → 1024 [SYN] Seq=0 Win=8192 Len=458
33359 1979.8378690…   192.168.10.1  192.168.10.2  TCP      512 [TCP Retransmission] [TCP Port numbers reused] 5555 → 1024 [SYN] Seq=0 Win=8192 Len=458
33360 1979.9180030…   192.168.10.1  192.168.10.2  TCP      512 [TCP Retransmission] [TCP Port numbers reused] 5555 → 1024 [SYN] Seq=0 Win=8192 Len=458
33361 1979.9787603…   192.168.10.1  192.168.10.2  TCP      512 [TCP Retransmission] [TCP Port numbers reused] 5555 → 1024 [SYN] Seq=0 Win=8192 Len=458
33362 1980.0464014…   192.168.10.1  192.168.10.2  TCP      512 [TCP Retransmission] [TCP Port numbers reused] 5555 → 1024 [SYN] Seq=0 Win=8192 Len=458
33363 1980.1092237…   192.168.10.1  192.168.10.2  TCP      512 [TCP Retransmission] [TCP Port numbers reused] 5555 → 1024 [SYN] Seq=0 Win=8192 Len=458
33364 1980.1738209…   192.168.10.1  192.168.10.2  TCP      512 [TCP Retransmission] [TCP Port numbers reused] 5555 → 1024 [SYN] Seq=0 Win=8192 Len=458

Frame 33158: 514 bytes on wire (4112 bits), 514 bytes captured (4112 bits) on interface enx0c37965f8a10, id 0
Ethernet II, Src: ca:fe:ca:fe:ca:fe (ca:fe:ca:fe:ca:fe), Dst: 00:00:00_00:00:01 (00:00:00:00:00:01)
Internet Protocol Version 4, Src: 192.168.10.1, Dst: 192.168.10.2
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 500
  Identification: 0x0001 (1)
  Flags: 0x00
  ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 64
  Protocol: TCP (6)
  Header Checksum: 0xe3af [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 192.168.10.1
  Destination Address: 192.168.10.2
```

Figure 5: Modified-script-traffic

Note that in order to achieve 512byte size packet, we need to adjust the number of words so that after adding the header, the packet reaches the size requirement.