Networking Ex5

208925578 and 314789264

Myping.cpp

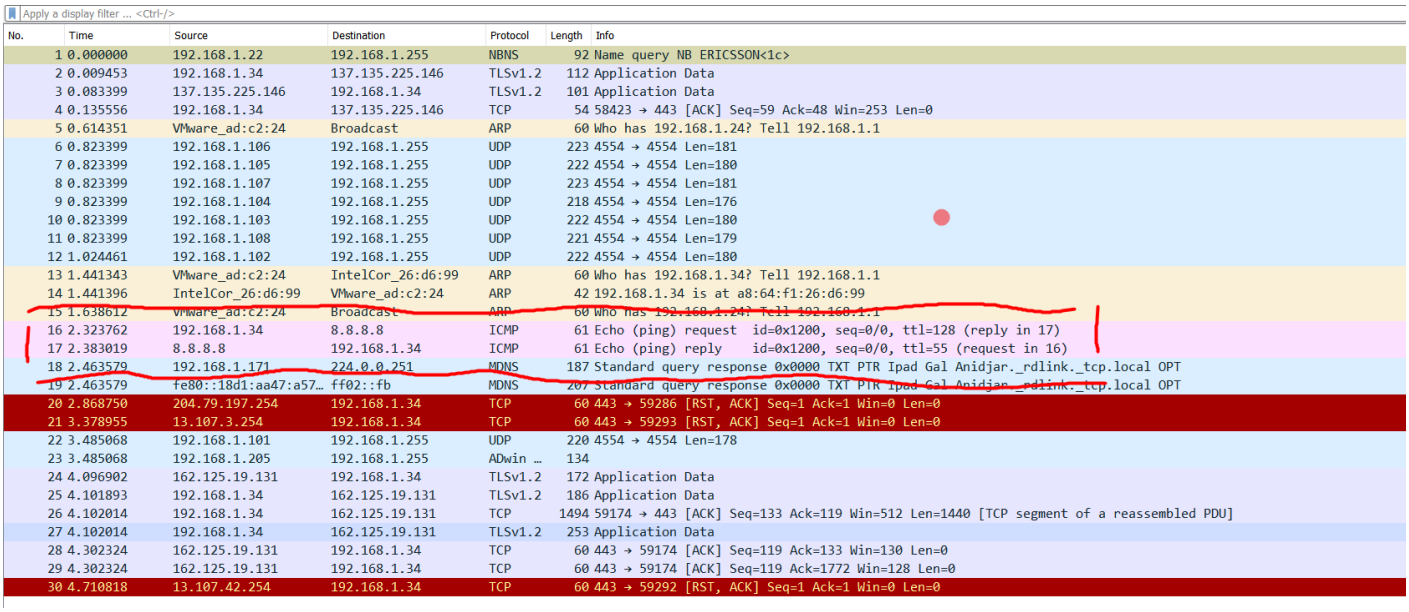
In this part, we update the ICMP.cpp skeleton to get myping.cpp. We created sender and receiver, and compute the RTT in milliseconds as can be seen below and in the Wireshark records.

```
itamark@DESKTOP-V5HJUST:~/network_course_ex5/networking-Ex5$ sudo ./myping
sent a packet:
        size: 27 bytes
        data: This is the ping.

recived:
        size: 47 bytes
        data: E

RTT time in miliseconds: 60.000000
RTT time in microseconds: 59468
```

As can be seen, the ping was sent to 8.8.8.8, and a few milliseconds afterward (59468 milliseconds) a reply was received.



Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.22	192.168.1.255	NBNS	92	Name query NB ERICSSON<1c>
2	0.009453	192.168.1.34	137.135.225.146	TLSv1.2	112	Application Data
3	0.083399	137.135.225.146	192.168.1.34	TLSv1.2	101	Application Data
4	0.135556	192.168.1.34	137.135.225.146	TCP	54	58423 → 443 [ACK] Seq=59 Ack=48 Win=253 Len=0
5	0.614351	VMware_ad:c2:24	Broadcast	ARP	60	Who has 192.168.1.24? Tell 192.168.1.1
6	0.823399	192.168.1.106	192.168.1.255	UDP	223	4554 → 4554 Len=181
7	0.823399	192.168.1.105	192.168.1.255	UDP	222	4554 → 4554 Len=180
8	0.823399	192.168.1.107	192.168.1.255	UDP	223	4554 → 4554 Len=181
9	0.823399	192.168.1.104	192.168.1.255	UDP	218	4554 → 4554 Len=176
10	0.823399	192.168.1.103	192.168.1.255	UDP	222	4554 → 4554 Len=180
11	0.823399	192.168.1.108	192.168.1.255	UDP	221	4554 → 4554 Len=179
12	1.024461	192.168.1.102	192.168.1.255	UDP	222	4554 → 4554 Len=180
13	1.441343	VMware_ad:c2:24	IntelCor_26:d6:99	ARP	60	Who has 192.168.1.34? Tell 192.168.1.1
14	1.441396	IntelCor_26:d6:99	VMware_ad:c2:24	ARP	42	192.168.1.34 is at a8:64:f1:26:d6:99
15	1.638612	VMware_ad:c2:24	Broadcast	ARP	60	Who has 192.168.1.24? Tell 192.168.1.1
16	2.323762	192.168.1.34	8.8.8.8	ICMP	61	Echo (ping) request id=0x1200, seq=0/0, ttl=128 (reply in 17)
17	2.383019	8.8.8.8	192.168.1.34	ICMP	61	Echo (ping) reply id=0x1200, seq=0/0, ttl=55 (request in 16)
18	2.463579	192.168.1.171	224.0.0.251	MDNS	187	Standard query response 0x0000 TXT PTR Ipad Gal Anidjar._rdlink._tcp.local OPT
19	2.463579	fe80::18d1:aa47:a57…	ff02::fb	MDNS	207	Standard query response 0x0000 TXT PTR Ipad Gal Anidjar._rdlink._tcp.local OPT
20	2.868750	204.79.197.254	192.168.1.34	TCP	60	443 → 59286 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
21	3.378955	13.107.3.254	192.168.1.34	TCP	60	443 → 59293 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
22	3.485068	192.168.1.101	192.168.1.255	UDP	220	4554 → 4554 Len=178
23	3.485068	192.168.1.205	192.168.1.255	ADwin …	134	
24	4.096902	162.125.19.131	192.168.1.34	TLSv1.2	172	Application Data
25	4.101893	192.168.1.34	162.125.19.131	TLSv1.2	186	Application Data
26	4.102014	192.168.1.34	162.125.19.131	TCP	1494	59174 → 443 [ACK] Seq=133 Ack=119 Win=512 Len=1440 [TCP segment of a reassembled PDU]
27	4.102014	192.168.1.34	162.125.19.131	TLSv1.2	253	Application Data
28	4.302324	162.125.19.131	192.168.1.34	TCP	60	443 → 59174 [ACK] Seq=119 Ack=133 Win=130 Len=0
29	4.302324	162.125.19.131	192.168.1.34	TCP	60	443 → 59174 [ACK] Seq=119 Ack=1772 Win=128 Len=0
30	4.710818	13.107.42.254	192.168.1.34	TCP	60	443 → 59292 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Sniffer.cpp

In this part, we built a sniffer. The sniffer should “sniff” ICMP transportation, and print to the screen the CODE, TYPE, IP_DST, and IP_SRC for each packet. The result is been shown in the screenshot below. In addition, we attach the Wireshark records to show the transportation.