

# Network Test Tool Report

**Name:** Chaoren Wang

**Student ID:** 122090513

**Date:** 2024-12-07

## 1. Testing Environment

- **Operating System:** MacOS 15.0
- **Network Interface:** Wi-Fi (en0)
- **Local IP Address:** 192.168.31.18
- **Router IP Address:** 192.168.31.1
- **DNS Server:** 8.8.8.8
- **Target IP Address:** 116.31.95.60
- **Target Domain:** cuhk.edu.cn
- **Tools:** curl, ping, nslookup, arp, netstat, traceroute, wireshark and tshark are also used for capturing and analyzing network packets.

## 2. Part A: Basic Network Test Commands [10 points]

### 2.1 ifconfig

The ifconfig command is used to display or configure network interfaces. It shows the IP addresses and other information, including the MAC address, subnet mask, and interface status on the system.

No protocol is used in this command.

**Command:**

```
1 | ifconfig
```

**Output:**

```
1 eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 65535
2         inet 172.17.0.2 netmask 255.255.0.0 broadcast 172.17.255.255
3             ether 02:42:ac:11:00:02 txqueuelen 0 (Ethernet)
4                 RX packets 1009 bytes 22958828 (22.9 MB)
5                 RX errors 0 dropped 0 overruns 0 frame 0
6                 TX packets 589 bytes 43403 (43.4 KB)
7                 TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
8
9 lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
10        inet 127.0.0.1 netmask 255.0.0.0
11        inet6 ::1 prefixlen 128 scopeid 0x10<host>
12             loop txqueuelen 1000 (Local Loopback)
13             RX packets 0 bytes 0 (0.0 B)
```

```
14      RX errors 0  dropped 0  overruns 0  frame 0
15      TX packets 0  bytes 0 (0.0 B)
16      TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```

## 2.2 ping

The `ping` command is used to check connectivity to a remote host. It sends ICMP Echo Request packets to the target and measures the round-trip time for each packet.

**Protocol Used:** ICMP (Internet Control Message Protocol), which operates at the Network Layer (Layer 3) of the OSI model.

### Key Information in Output:

- TTL (Time To Live): 63 - indicates how many more network hops the packet can traverse
- Round Trip Time: varies between 14.7-16.8 ms - shows network latency
- Packet Size: 56(84) bytes - 56 bytes of data plus 28 bytes of ICMP and IP headers

### Command:

```
1 | ping cuhk.edu.cn
```

### Output:

```
1 | PING cuhk.edu.cn (116.31.95.60) 56(84) bytes of data.
2 | 64 bytes from 116.31.95.60 (116.31.95.60): icmp_seq=1 ttl=63 time=16.1 ms
3 | 64 bytes from 116.31.95.60 (116.31.95.60): icmp_seq=2 ttl=63 time=15.3 ms
4 | 64 bytes from 116.31.95.60 (116.31.95.60): icmp_seq=3 ttl=63 time=16.8 ms
5 | 64 bytes from 116.31.95.60 (116.31.95.60): icmp_seq=4 ttl=63 time=14.7 ms
6 |
7 | --- cuhk.edu.cn ping statistics ---
8 | 4 packets transmitted, 4 received, 0% packet loss, time 3008ms
9 | rtt min/avg/max/mdev = 14.668/15.708/16.819/0.813 ms
```

## 2.3 nslookup

The `nslookup` command is used to query DNS to obtain domain name or IP address mapping.

### Protocol Details:

- Primary Protocol: DNS over UDP (port 53)
- Alternative Protocols:
  - DNS over TCP: Used for responses larger than 512 bytes
  - DNS over TLS (DoT): Encrypted DNS queries over port 853
  - DNS over HTTPS (DoH): Encrypted DNS queries over HTTPS

### Output Explanation:

- Server: The DNS server being queried (8.8.8.8 - Google's public DNS)

- Non-authoritative answer: Response comes from the server's cache rather than the authoritative server for the domain
- Address: The resolved IP address for the domain

#### Command:

```
1 | nslookup cuhk.edu.cn 8.8.8.8
```

#### Output:

```
1 | Server:          8.8.8.8
2 | Address:         8.8.8.8#53
3 |
4 | Non-authoritative answer:
5 | Name:    cuhk.edu.cn
6 | Address: 116.31.95.60
```

## 2.4 arp

The `arp` command is used to display or modify the ARP table, which maps IP addresses to MAC addresses. It uses the ARP protocol, which is a link layer protocol.

#### Command:

```
1 | arp -a
```

#### Output:

```
1 | ? (192.168.31.1) at 44:f7:70:4d:57:ee on en0 ifscope [ethernet]
2 | ? (192.168.31.18) at c4:35:d9:88:f:2a on en0 ifscope [ethernet]
3 | ? (192.168.31.83) at b2:7c:1e:8b:75:6e on en0 ifscope [ethernet]
4 | ? (192.168.31.99) at da:bd:44:a9:76:20 on en0 ifscope [ethernet]
5 | ? (192.168.31.253) at e:dc:28:a6:ab:2f on en0 ifscope [ethernet]
6 | ? (192.168.31.255) at ff:ff:ff:ff:ff:ff on en0 ifscope [ethernet]
7 | mdns.mcast.net (224.0.0.251) at 1:0:5e:0:0:fb on en0 ifscope permanent [ethernet]
```

## 2.5 netstat

The `netstat` command is used to display active network connections, listening ports, routing tables, and interface statistics.

#### Command:

```
1 | netstat
```

#### Output:

```
1 | Active Internet connections
```

	Proto	Recv-Q	Send-Q	Local Address	Foreign Address	(state)
2	tcp4	0	0	localhost.56641	localhost.55787	ESTABLISHED
3	tcp4	0	0	localhost.55787	localhost.56641	ESTABLISHED
4	tcp4	0	0	localhost.56641	localhost.55786	ESTABLISHED
5	tcp4	0	0	localhost.55786	localhost.56641	ESTABLISHED
6	tcp4	0	0	localhost.56641	localhost.55785	ESTABLISHED
7	tcp4	0	0	localhost.56641	localhost.55785	ESTABLISHED
8	tcp4	0	0	localhost.55785	localhost.56641	ESTABLISHED
9	tcp6	0	0	fe80::3c1f:67ff:.8771	fe80::403:9bff:f.62163	ESTABLISHED
10	tcp4	0	0	192.168.31.33.55692	52.182.143.214.https	ESTABLISHED
11	tcp4	0	0	192.168.31.33.55691	183.1.15.218.bro.17010	ESTABLISHED
12	tcp4	0	0	localhost.7890	localhost.55686	ESTABLISHED
13	tcp4	0	0	localhost.55686	localhost.7890	ESTABLISHED
14	tcp4	0	0	192.168.31.33.55670	183.1.15.218.bro.17010	ESTABLISHED
15	...					

## 2.6 traceroute

The `traceroute` command is used to trace the path that packets take from the local system to a remote host, each hop along the path is represented by an IP address shown in the output, along with the time taken for each hop. It uses the ICMP protocol, which is a network layer protocol.

**Command:**

```
1 | traceroute -P icmp www.cuhk.edu.cn
```

**Output:**

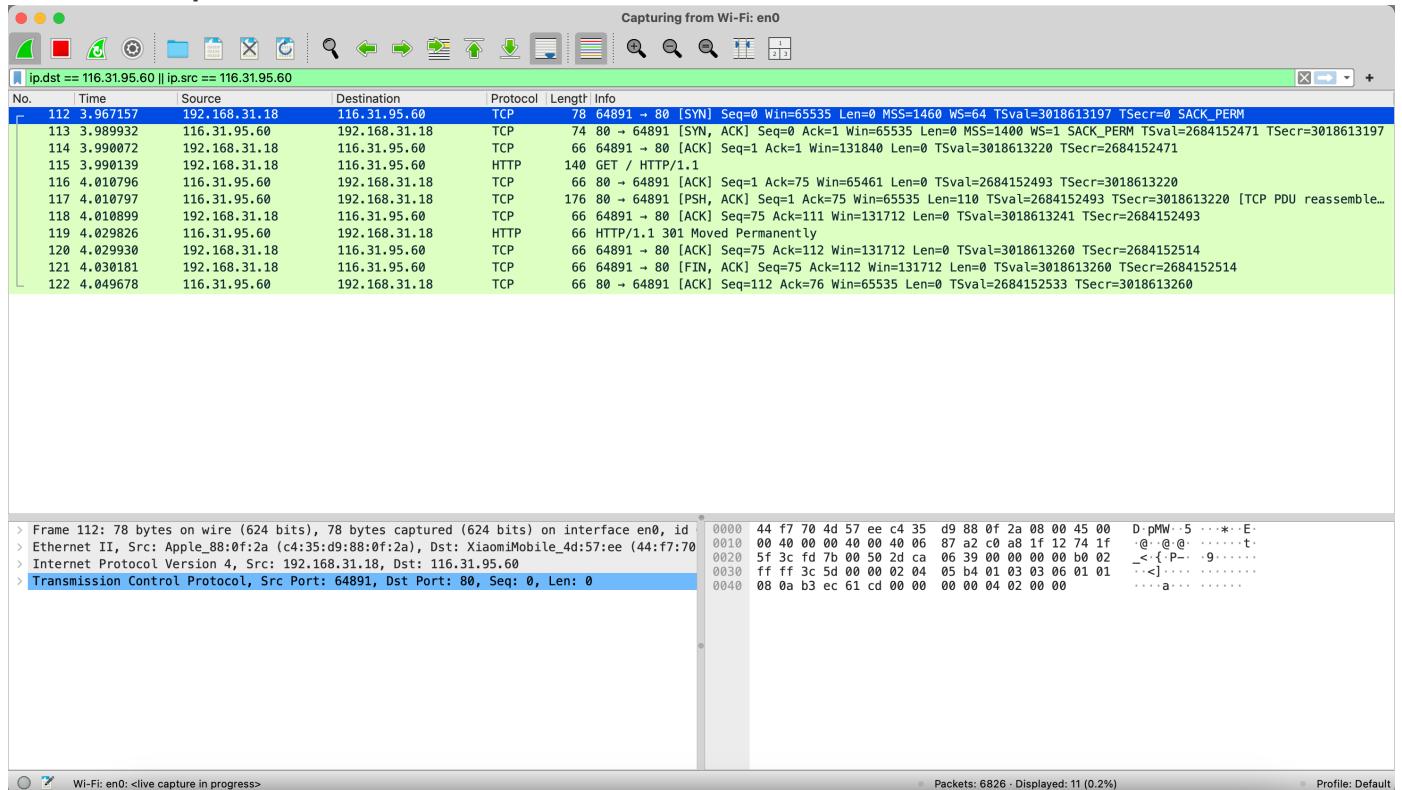
```
1 traceroute to www.cuhk.edu.cn (116.31.95.60), 64 hops max, 48 byte packets
2  1 xiaoliang (192.168.31.1)  6.236 ms  9.915 ms  23.824 ms
3  2 192.168.1.1 (192.168.1.1)  6.380 ms  8.252 ms  13.637 ms
4  3 10.141.64.1 (10.141.64.1)  16.561 ms  9.768 ms  10.083 ms
5  4 120.80.167.213 (120.80.167.213)  9.829 ms  9.568 ms  9.978 ms
6  5 * * *
7  6 219.158.110.54 (219.158.110.54)  14.892 ms  17.729 ms  9.902 ms
8  7 * * *
9  8 * 202.97.95.133 (202.97.95.133)  18.309 ms *
10 9 * 14.147.127.10 (14.147.127.10)  18.501 ms  12.918 ms
11 10 * * *
12 11 14.147.80.50 (14.147.80.50)  21.200 ms  24.399 ms  24.989 ms
13 12 116.31.95.60 (116.31.95.60)  19.429 ms  19.846 ms  20.017 ms
```

## 3. Part B: Capture TCP/UDP Packets and Connection Process [15 points]

### 3.1 TCP Connection Process

First, we use `curl` to get the HTTP request from the server `cuhk.edu.cn`, i.e. `curl http://cuhk.edu.cn`.

## Wireshark Capture:



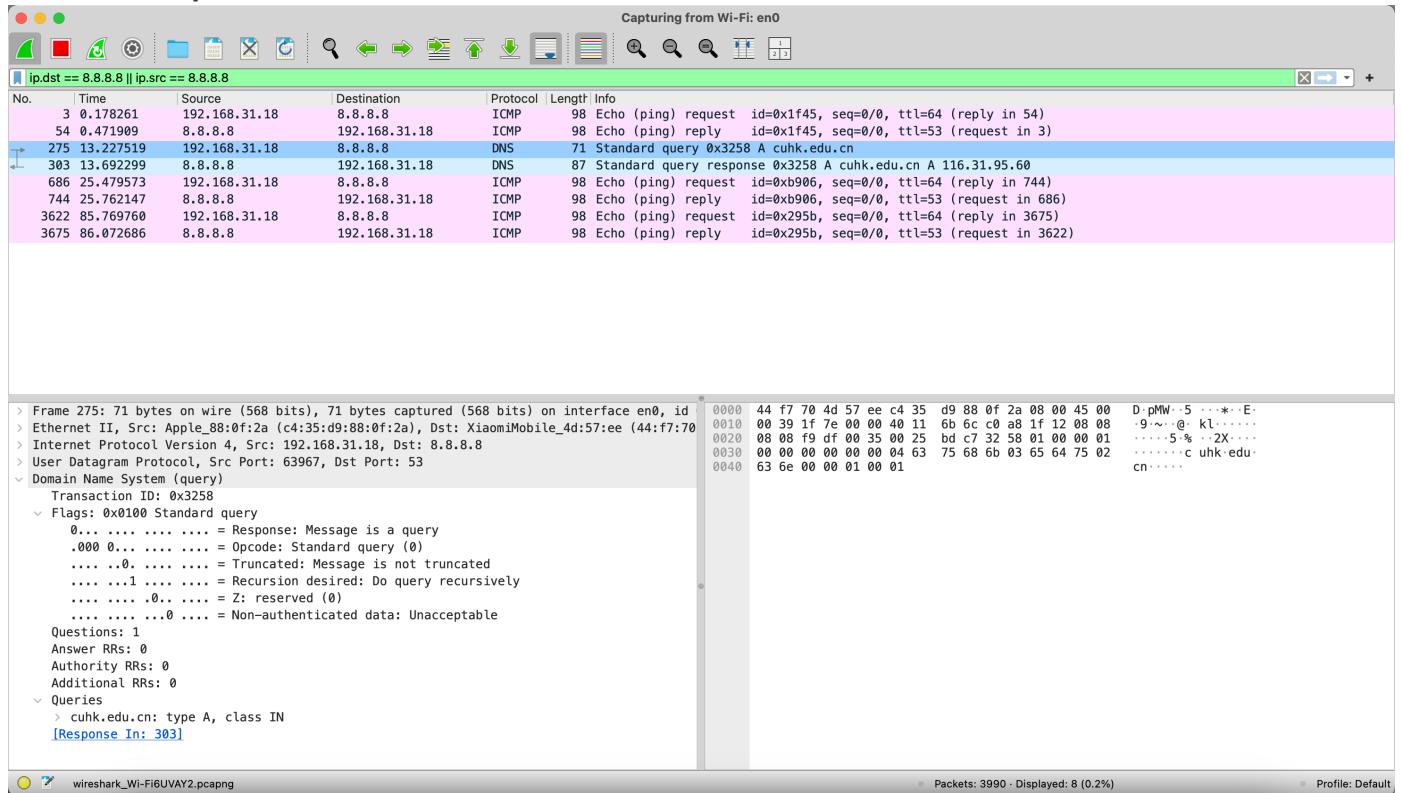
### Steps:

- SYN Packet #112** – Client (192.168.31.18) sends a SYN packet to the server to initiate a connection.
- SYN-ACK Packet #113** – Server (116.31.95.60) responds with a SYN-ACK packet as acknowledgement, confirming the connection request.
- ACK Packet #114** – Client (192.168.31.18) sends an ACK packet back to the server to acknowledge the receipt of the SYN-ACK packet, completing the three-way handshake.

## 3.2 UDP Connection Process

We use `nslookup` to query the DNS server `8.8.8.8` for the IP address of `cuhk.edu.cn`.

## Wireshark Capture:



## Steps:

1. **DNS Query** #275 – Client (192.168.31.18) sends a DNS query packet to the DNS server (8.8.8.8) to request the IP address of `cuhk.edu.cn`.
2. **DNS Response** #303 – DNS server (8.8.8.8) responds with a DNS response packet containing the IP address of `cuhk.edu.cn`.

## 4. Part C: Wireshark and Tshark Packet Capture Analysis [15 points]

### 4.1 Wireshark Capture

#### Command & Output (Tshark):

```
1 | tshark -i en0 -f "icmp or arp or udp"
```

```
tshark -i en0 -f "icmp or arp or udp"

X tshark (dumpcap)
  157 66.147456 118.249.109.101 -> 192.168.31.18 UDP 64 443 -> 63708 Len=22
  158 67.533317 192.160.31.99 -> 224.0.0.251 MDNS 272 Standard query 0x0000 PTR _companion-link.tcp.local, "QM" question PTR _rdlink.tcp.local, "QM" question SRV C64A264C-6C9D-4B83-A2CD-37F54A692A35.aaq
  159 67.533798 fe80::23:f91fa:7c3:15a0 -> ff02::fb MDNS 292 Standard query 0x0000 PTR _companion-link._tcp.local, "QM" question PTR _rdlink._tcp.local, "QM" question SRV C64A264C-6C9D-4B83-A2CD-37F54A692A35.aaq
  160 68.804018 192.168.31.18 -> 104.194.8.134 UDP 75 64542 -> 9993 Len=33
  161 68.804069 192.168.31.18 -> 104.194.8.134 UDP 75 64542 -> 9993 Len=33
  163 68.804971 192.168.31.18 -> 104.194.8.134 UDP 75 64542 -> 9993 Len=33
  164 68.804145 192.168.31.18 -> 35.208.170.228 UDP 183 64542 -> 21796 Len=141
  165 69.671269 192.168.31.18 -> 192.168.31.18 UDP 88 21796 -> 64542 Len=46
  166 69.885285 192.168.31.18 -> 8.8.8.8 ICMP 88 Echo (ping) request id=0xbpcf, seq=0/0, ttl=64
  167 69.885301 192.168.31.18 -> 8.8.8.8 ICMP 88 Echo (ping) request id=0xbpcf, seq=0/0, ttl=64
  168 71.014666 192.168.31.251 -> 224.0.0.251 MDNS 101 Standard query 0x0000 PTR _companion-link._tcp.local, "QM" question PTR _rdlink._tcp.local, "QM" question PTR _rdlink._tcp.local, "QM" question PTR Yuan'sMacBook Air._companion-link._tcp.local
  169 71.015087 fe80::142f:dca:566d:b9b2 -> ff02::fb MDNS 120 Standard query 0x0000 PTR _companion-link._tcp.local, "QM" question PTR _rdlink._tcp.local, "QM" question PTR _rdlink._tcp.local, "QM" question PTR Yuan'sMacBook Air._companion-link._tcp.local
  170 71.183445 192.168.31.18 -> 192.168.31.253 MDNS 376 Standard query response 0x0000 PTR Yuan'sMacBook Air._companion-link._tcp.local SRV, cache flush 0 &gt; 58700 YuandeMacBook Air.local TXT, cache flush T AAAA, cache flush fe80::8b5:42d0:492f:23c1 A, cache flush 192.168.31.18
  171 71.185178 b2:7c:1e:8b:75:6 -> Apple_88:0f:2a ARP 42 Who has 192.168.31.18? Tell 192.168.31.83
  172 71.185216 Apple_88:0f:2a -> b2:7c:1e:8b:75:6 ARP 42 Who has 192.168.31.18? at c4:35:9d:88:0f:2a
  173 72.038867 192.168.31.253 -> 224.0.0.251 MDNS 191 Standard query 0x0000 PTR _companion-link._tcp.local, "QM" question PTR _rdlink._tcp.local, "QM" question PTR MacBook Pro (2).companion-link._tcp.local PTR Zheka1ipad.companion-link._tcp.local
  174 72.039233 fe80::142f:dca:566d:b9b2 -> ff02::fb MDNS 211 Standard query 0x0000 PTR _companion-link._tcp.local, "QM" question PTR _rdlink._tcp.local, "QM" question PTR MacBook Pro (2).companion-link._tcp.local PTR Zheka1ipad.companion-link._tcp.local PTR Yuan'sMacBook Air._companion-link._tcp.local
  175 72.362446 35.208.170.228 -> 192.168.31.18 UDP 88 21796 -> 64542 Len=46
  176 73.836691 192.168.31.18 -> 104.194.8.134 UDP 75 64542 -> 9993 Len=33
  177 73.836746 192.168.31.18 -> 104.194.8.134 UDP 75 64542 -> 9993 Len=33
  178 73.836749 192.168.31.18 -> 104.194.8.134 UDP 75 64542 -> 9993 Len=33
  179 73.836753 192.168.31.18 -> 104.194.8.134 UDP 75 64542 -> 9993 Len=33
  180 73.836768 192.168.31.18 -> 35.208.170.228 UDP 183 64542 -> 21796 Len=141
  181 74.183440 35.208.170.228 -> 192.168.31.18 UDP 88 21796 -> 64542 Len=46
  182 74.969571 192.168.31.18 -> 239.255.255.250 SSDP 208 M-SEARCH * HTTP/1.1
  183 74.969805 172.24.63.45 -> 239.255.255.250 SSDP 208 M-SEARCH * HTTP/1.1
  184 74.969833 0.0.1.1 -> 239.255.255.250 SSDP 208 M-SEARCH * HTTP/1.1
  185 75.974789 192.168.31.18 -> 239.255.255.250 SSDP 208 M-SEARCH * HTTP/1.1
  186 75.974931 172.24.63.45 -> 239.255.255.250 SSDP 208 M-SEARCH * HTTP/1.1
  187 75.974935 0.0.1.1 -> 239.255.255.250 SSDP 208 M-SEARCH * HTTP/1.1
  188 76.979924 192.168.31.18 -> 239.255.255.250 SSDP 208 M-SEARCH * HTTP/1.1
  189 76.980027 172.24.63.45 -> 239.255.255.250 SSDP 208 M-SEARCH * HTTP/1.1
  190 76.980034 0.0.1.1 -> 239.255.255.250 SSDP 208 M-SEARCH * HTTP/1.1
  191 77.982627 192.168.31.18 -> 239.255.255.250 SSDP 208 M-SEARCH * HTTP/1.1
  192 77.982709 172.24.63.45 -> 239.255.255.250 SSDP 208 M-SEARCH * HTTP/1.1
  193 77.982714 0.0.1.1 -> 239.255.255.250 SSDP 208 M-SEARCH * HTTP/1.1

  Last login: Sat Dec 7 15:13:39 on ttys013
  nslookup cuhk.edu.cn 8.8.8.8
  Server: 8.8.8.8
  Address: 8.8.8.8#53
  ping cuhk.edu.cn
  PING cuhk.edu.cn (116.31.95.60): 56 data bytes
  64 bytes from 116.31.95.60: icmp_seq=0 ttl=17 time=14.080 ms
  64 bytes from 116.31.95.60: icmp_seq=1 ttl=17 time=18.385 ms
  64 bytes from 116.31.95.60: icmp_seq=2 ttl=17 time=13.417 ms
  --- cuhk.edu.cn ping statistics ---
  3 packets transmitted, 3 packets received, 0.0% packet loss
  round-trip min/avg/max/stddev = 13.417/15.294/18.385/2.202 ms
  curl -v cuhk.edu.cn
  * Trying 116.31.95.60:80...
  * Connected to cuhk.edu.cn (116.31.95.60) port 80
  > GET / HTTP/1.1
  > Host: cuhk.edu.cn
  > User-Agent: curl/8.4.0
  > Accept: */*
  > 
  < HTTP/1.1 301 Moved Permanently
  < Location: https://cuhk.edu.cn/
  < Content-Type: text/html
  < Connection: close
  < 
  * Closing connection
  
```

```
1 | Capturing on 'Wi-Fi: en0'
2 |
3 | 16 9.415227 192.168.31.18 -> 8.8.8.8           ICMP 98 Echo (ping) request id=0xc286, seq=0/0, ttl=64 # This is the ping request
4 | 17 9.711140          8.8.8.8 -> 192.168.31.18 ICMP 98 Echo (ping) reply      id=0xc286, seq=0/0, ttl=53 (request in 16) # This is the ping response
5 |
6 | 18 13.176259 192.168.31.18 -> 104.194.8.134 UDP 75 64542 -> 9993 Len=33 # This is the UDP packet
7 | 19 13.176323 192.168.31.18 -> 104.194.8.134 UDP 75 64542 -> 9993 Len=33
8 | 20 13.176326 192.168.31.18 -> 104.194.8.134 UDP 75 64542 -> 9993 Len=33
9 | 21 13.176354 192.168.31.18 -> 104.194.8.134 UDP 75 64542 -> 9993 Len=33
10 |
11 | 24 14.081061 192.168.31.18 -> 8.8.8.8           DNS 71 Standard query 0x1be4 A cuhk.edu.cn # This is the DNS query
12 | 25 14.560930          8.8.8.8 -> 192.168.31.18 DNS 87 Standard query response 0x1be4 A cuhk.edu.cn A 116.31.95.60 # This is the DNS response
13 |
14 | 59 28.197455 Apple_88:0f:2a -> 72:0b:5d:07:84:ef ARP 42 Who has 192.168.31.33? Tell 192.168.31.18 # This is the ARP request
15 | 60 28.202957 72:0b:5d:07:84:ef -> Apple_88:0f:2a ARP 42 192.168.31.33 is at 72:0b:5d:07:84:ef # This is the ARP response
16 | ...
```

```
1 | tshark -i en0 -f "tcp port 80"
```

```

1 Capturing on 'Wi-Fi: en0'
2     1 0.000000 192.168.31.18 → 116.31.95.60 TCP 78 51842 → 80 [SYN] Seq=0 Win=65535
Len=0 MSS=1460 WS=64 TSval=2521737059 TSecr=0 SACK_PERM # This is the SYN
3     2 0.021255 116.31.95.60 → 192.168.31.18 TCP 74 80 → 51842 [SYN, ACK] Seq=0
Ack=1 Win=65535 Len=0 MSS=1400 WS=1 SACK_PERM TSval=2685910185 TSecr=2521737059 #
SYN-ACK
4     3 0.021426 192.168.31.18 → 116.31.95.60 TCP 66 51842 → 80 [ACK] Seq=1 Ack=1
Win=131840 Len=0 TSval=2521737080 TSecr=2685910185 # ACK
5     4 0.021459 192.168.31.18 → 116.31.95.60 HTTP 140 GET / HTTP/1.1 # Connection
established, HTTP protocol now
6     5 0.039600 116.31.95.60 → 192.168.31.18 TCP 66 80 → 51842 [ACK] Seq=1 Ack=75
Win=65461 Len=0 TSval=2685910203 TSecr=2521737080
7     6 0.039601 116.31.95.60 → 192.168.31.18 TCP 176 HTTP/1.1 301 Moved Permanently
# 301 Moved Permanently
8     7 0.039651 192.168.31.18 → 116.31.95.60 TCP 66 51842 → 80 [ACK] Seq=75 Ack=111
Win=131712 Len=0 TSval=2521737098 TSecr=2685910203 # ACK
9     8 0.059510 116.31.95.60 → 192.168.31.18 HTTP 66 HTTP/1.1 301 Moved Permanently
# 301 Moved Permanently
10    9 0.059599 192.168.31.18 → 116.31.95.60 TCP 66 51842 → 80 [ACK] Seq=75 Ack=112
Win=131712 Len=0 TSval=2521737119 TSecr=2685910221 # ACK
11   10 0.062385 192.168.31.18 → 116.31.95.60 TCP 66 51842 → 80 [FIN, ACK] Seq=75
Ack=112 Win=131712 Len=0 TSval=2521737121 TSecr=2685910221 # FIN-ACK
12   11 0.080326 116.31.95.60 → 192.168.31.18 TCP 66 80 → 51842 [ACK] Seq=112 Ack=76
Win=65535 Len=0 TSval=2685910244 TSecr=2521737121 # ACK

```



## Wireshark Screenshot:

Capturing from Wi-Fi: en0

**arp**

No.	Time	Source	Destination	Protocol	Length	Info
202	10.397442	32:d8:f6:d2:b8:9b	Broadcast	ARP	42	Who has 192.168.31.1? Tell 192.168.31.228
1162	32.383334	XiaomiMobile_4d:57:..	Apple_88:0f:2a	ARP	42	192.168.31.1 is at 44:f7:70:4d:57:ee
1601	57.264525	Apple_88:0f:2a	0:dc:28:a6:ab:2f	ARP	42	Who has 192.168.31.253? Tell 192.168.31.18
1603	57.271674	0:dc:28:a6:ab:2f	Apple_88:0f:2a	ARP	42	192.168.31.253 is at 0:dc:28:a6:ab:2f
3198	71.429113	32:d8:f6:d2:b8:9b	Broadcast	ARP	42	Who has 192.168.31.1? Tell 192.168.31.228

Frame 202: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)  
 Ethernet II, Src: 32:d8:f6:d2:b8:9b (32:d8:f6:d2:b8:9b), Dst: Broadcast  
 Address Resolution Protocol (request)  
 Hardware type: Ethernet (1)  
 Protocol type: IPv4 (0x0800)  
 Hardware size: 6  
 Protocol size: 4  
 Opcode: request (1)  
 Sender MAC address: 32:d8:f6:d2:b8:9b (32:d8:f6:d2:b8:9b)  
 Sender IP address: 192.168.31.228  
 Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)  
 Target IP address: 192.168.31.1

Frame 1162: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)  
 Ethernet II, Src: XiaomiMobile\_4d:57:ee (44:f7:70:4d:57:ee), Dst: Ap  
 Address Resolution Protocol (reply)  
 Hardware type: Ethernet (1)  
 Protocol type: IPv4 (0x0800)  
 Hardware size: 6  
 Protocol size: 4  
 Opcode: reply (2)  
 Sender MAC address: XiaomiMobile\_4d:57:ee (44:f7:70:4d:57:ee)  
 Sender IP address: 192.168.31.1  
 Target MAC address: Apple\_88:0f:2a (c4:35:d9:88:0f:2a)  
 Target IP address: 192.168.31.18

Sender IP address (arp.dst.proto.ipv4), 4 bytes  
 Show packet bytes Layout: Vertical (Stacked)

Help Close

Address Resolution Protocol: Protocol

Capturing from Wi-Fi: en0

**icmp**

No.	Time	Source	Destination	Protocol	Length	Info
174	6.271953	192.168.31.18	116.31.95.60	ICMP	98	Echo (ping) request id=0xdb85, seq=0/0, ttl=64 (reply in 179)
179	6.291506	116.31.95.60	192.168.31.18	ICMP	98	Echo (ping) reply id=0xdb85, seq=0/0, ttl=117 (request in 174)

[Header checksum status: Unverified]  
 Source Address: 192.168.31.18  
 Destination Address: 116.31.95.60  
 [Stream index: 15]

Internet Control Message Protocol  
 Type: 8 (Echo (ping) request)  
 Code: 0  
 Checksum: 0xa1a9 [correct]  
 [Checksum Status: Good]  
 Identifier (BE): 56197 (0xdb85)  
 Identifier (LE): 34267 (0x85db)  
 Sequence Number (BE): 0 (0x0000)  
 Sequence Number (LE): 0 (0x0000)  
 [Response frame: 179]  
 Timestamp from icmp data: Dec 7, 2024 15:36:57.111723000 CST  
 [Timestamp from icmp data (relative): 0.000069000 seconds]  
 Data (48 bytes)  
 Data: 08090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435  
 [Length: 48]

Header Checksum: 0x76a2 [Validation disabled]  
 [Header checksum status: Unverified]  
 Source Address: 116.31.95.60  
 Destination Address: 192.168.31.18  
 [Stream index: 15]

Internet Control Message Protocol  
 Type: 0 (Echo (ping) reply)  
 Code: 0  
 Checksum: 0x229d [correct]  
 [Checksum Status: Good]  
 Identifier (BE): 56197 (0xdb85)  
 Identifier (LE): 34267 (0x85db)  
 Sequence Number (BE): 0 (0x0000)  
 Sequence Number (LE): 0 (0x0000)  
 [Request frame: 174]

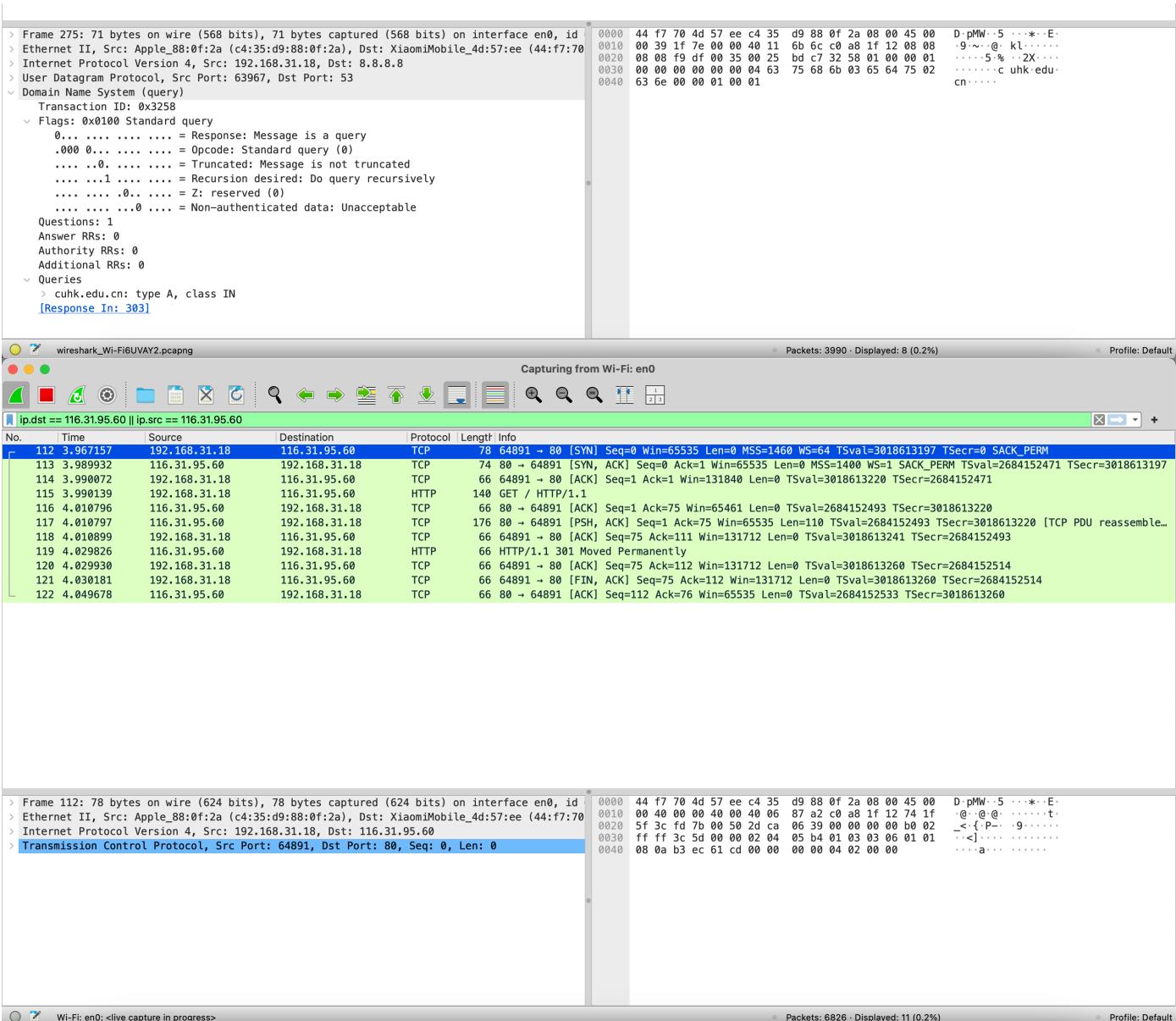
Destination Address (ip.dst), 4 bytes  
 Show packet bytes Layout: Vertical (Stacked)

Help Close

ip.dst == 8.8.8.8 || ip.src == 8.8.8.8

Capturing from Wi-Fi: en0

No.	Time	Source	Destination	Protocol	Length	Info
3	0.178261	192.168.31.18	8.8.8.8	ICMP	98	Echo (ping) request id=0x1f45, seq=0/0, ttl=64 (reply in 54)
54	0.471909	8.8.8.8	192.168.31.18	ICMP	98	Echo (ping) reply id=0x1f45, seq=0/0, ttl=53 (request in 3)
275	13.227519	192.168.31.18	8.8.8.8	DNS	71	Standard query 0x3258 A cuhk.edu.cn
303	13.692299	8.8.8.8	192.168.31.18	DNS	87	Standard query response 0x3258 A cuhk.edu.cn A 116.31.95.60
686	25.479573	192.168.31.18	8.8.8.8	ICMP	98	Echo (ping) request id=0xb06, seq=0/0, ttl=64 (reply in 744)
744	25.762147	8.8.8.8	192.168.31.18	ICMP	98	Echo (ping) reply id=0xb06, seq=0/0, ttl=53 (request in 686)
3622	85.769760	192.168.31.18	8.8.8.8	ICMP	98	Echo (ping) request id=0x295b, seq=0/0, ttl=64 (reply in 3675)
3675	86.072686	8.8.8.8	192.168.31.18	ICMP	98	Echo (ping) reply id=0x295b, seq=0/0, ttl=53 (request in 3622)



## Explanation:

- **ARP:** Address Resolution Protocol used to map an IP address to a MAC address. We can see the ARP request and response packets in the capture.
  - Sender: 192.168.31.228 -> Receiver: 00:00:00:00:00:00, asking for the IP address of 192.168.31.1
  - Sender: 44:f7:70:4d:57:ee (router) -> Receiver: 192.168.31.18, responding with the IP address of 192.168.31.1
- **ICMP:** Internet Control Message Protocol, used for diagnostic messages like ping. We can see the ping request and response packets in the capture.
  - Sender: 192.168.31.18 -> Receiver: 116.31.95.60, sending a ping request to the DNS server, data length is 48 bytes
  - Sender: 116.31.95.60 -> Receiver: 192.168.31.18, responding to the ping request, data length is 48 bytes
- **UDP:** User Datagram Protocol, a connectionless protocol for sending datagrams. We can see the DNS query and response packets in the capture.

- Sender: 192.168.31.18 -> Receiver: 8.8.8.8, sending a DNS query packet to the DNS server, data length is 71 bytes, asking for the IP address of cuhk.edu.cn
- Sender: 8.8.8.8 -> Receiver: 192.168.31.18, responding to the DNS query, data length is 87 bytes, answering with the IP address of cuhk.edu.cn is 116.31.95.60
- **TCP:** Transmission Control Protocol, a connection-oriented protocol for reliable data transfer. We can see the TCP connection establishment, data transfer, and termination in the capture.
  - Details are shown in the Part B
- **HTTP:** Hypertext Transfer Protocol, used for transferring web pages. We can see the HTTP request and response packets in the capture.
  - Client (192.168.31.18) send: GET / HTTP/1.1
  - Server (116.31.95.60) response: HTTP/1.1 301 Moved Permanently

**Question:** Why contents of the bag you caught may not be the same as what you learned?

This might because different versions of the same protocol have different default settings. As the test environment is MacOS, might be different from the Linux environment.

## 5. Part D: Encapsulation and Decapsulation Process [10 points]

The encapsulation process involves wrapping data with necessary protocol information at each layer of the OSI model, while decapsulation is the reverse process when receiving data.

### Encapsulation for Screenshot:

1. **Application Layer:** Data (e.g., HTTP request to cuhk.edu.cn).
2. **Transport Layer:** Data is encapsulated in TCP segments (or UDP segments in the DNS query), with the source port (51034) and destination port (80).
3. **Network Layer:** Data is encapsulated in IP packets, including the source IP address (192.168.31.18) and destination IP address (116.31.95.60).
4. **Data Link Layer:** Data is encapsulated in Ethernet frames, including the source MAC address (c4:35:d9:88:0f:2a) and destination MAC address (44:f7:70:4d:57:ee).

## Wireshark Screenshot:

Frame 15030: 140 bytes on wire (1120 bits), 140 bytes captured (1120 bits) on interface en0, id 0  
> Ethernet II, Src: Apple\_88:0f:2a (c4:35:d9:88:0f:2a), Dst: XiaomiMobile\_4d:57:ee (44:f7:70:4d:57:ee)  
> Internet Protocol Version 4, Src: 192.168.31.18, Dst: 116.31.95.60  
> Transmission Control Protocol, Src Port: 51034, Dst Port: 80, Seq: 1, Ack: 1, Len: 74  
Source Port: 51034  
Destination Port: 80  
[Stream index: 301]  
> [Conversation completeness: Complete, WITH\_DATA (31)]  
[TCP Segment Len: 74]  
Sequence Number: 1 (relative sequence number)  
Sequence Number (raw): 3084085645  
[Next Sequence Number: 75 (relative sequence number)]  
Acknowledgment Number: 1 (relative ack number)  
Acknowledgment number (raw): 852339266  
1000 .... = Header Length: 32 bytes (8)  
> Flags: 0x18 (PSH, ACK)  
Window: 2060  
[Calculated window size: 131840]  
[Window size scaling factor: 64]  
Checksum: 0xa143 [unverified]  
[Checksum Status: Unverified]  
Urgent Pointer: 0  
> Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps  
> [Timestamps]  
> [SEQ/ACK analysis]  
TCP payload (74 bytes)

< Hypertext Transfer Protocol

> GET / HTTP/1.1\r\nHost: cuhk.edu.cn\r\nUser-Agent: curl/8.4.0\r\nAccept: \*/\*\r\n\r\n[Response in frame: 15034]  
[Full request URI: http://cuhk.edu.cn/]

Hypertext Transfer Protocol (http), 74 bytes

Show packet bytes Layout: Horizontal (Side-by-side)

Help Close

## Decapsulation Process:

When a packet is received, it goes through the reverse process—Ethernet frame is stripped, IP packet is extracted, and TCP/UDP segment is processed at the transport layer.