COMPUTER NETWORKING

Student name: Son Cao

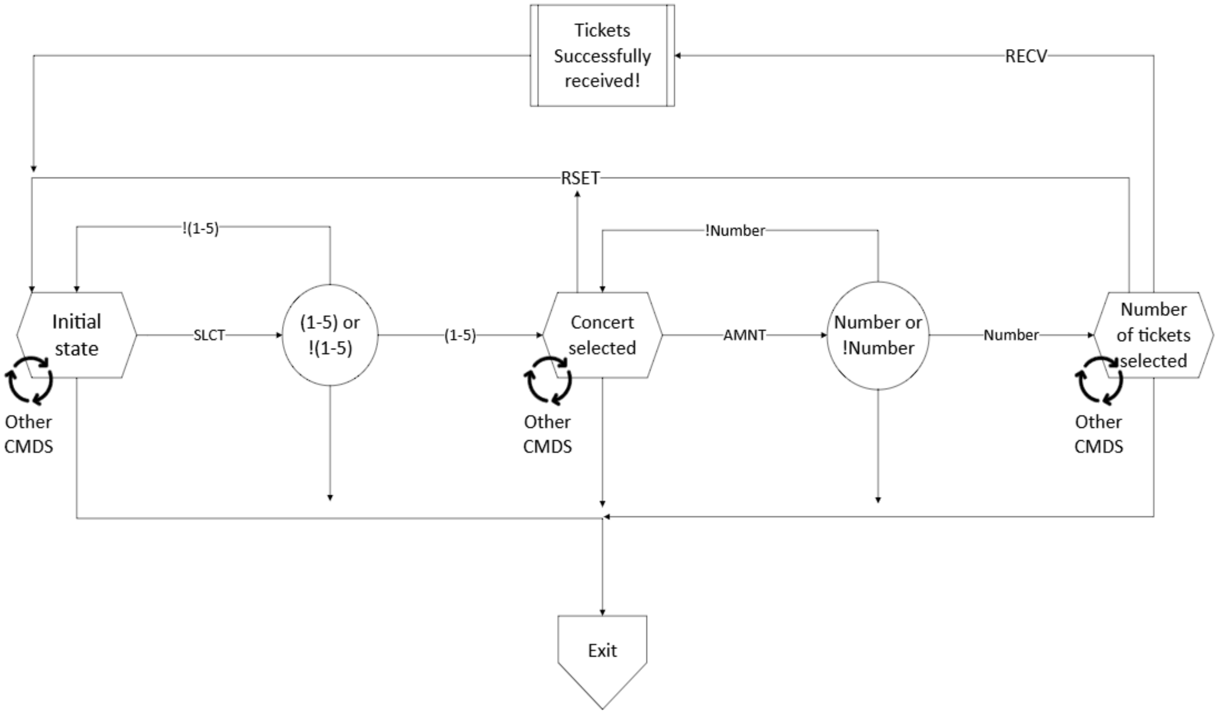
Student number: 570135

assignment week 1: protocol analysis

Commands to open blobmac

```
Ns localhost 5107
```

- HELP – List all commands
- RECV – Receives concert tickets
- STAT – Lists all selected tickets
- LIST – Lists all possible concerts
- SLCT – Selects concert
- AMNT – Selects amount of tickets
- RSET – Clears concerts and the corresponding tickets
- EXIT – Closes program



assignment week 2 & 3: Wireshark

How did the DNS lookup find the web address?

1. My computer sent a DNS query to the configured DNS server (e.g., Saxion's DNS server or my home router).
2. The DNS server resolved the domain name neverssl.com to its corresponding IP address (34.223.124.45).
3. The resolved IP address was returned to my computer, allowing it to establish a connection to the web server.

How long did it take to find and establish the connection and download the webpage + image(s)?

The total time was approximately 200-300 milliseconds:

- DNS lookup: ~50 ms
- TCP connection establishment (3-way handshake): ~100 ms
- HTTP request and response (including webpage and images): ~100-150 ms

Approximately how many TCP packets, IP packets, and Ethernet frames were used? What were the average sizes?

- **TCP packets**: ~20-30 packets (average size: ~500-1500 bytes)
- **IP packets**: ~20-30 packets (average size: ~500-1500 bytes)
- **Ethernet frames**: ~20-30 frames (average size: ~500-1500 bytes)
  I determined this by filtering the Wireshark capture for TCP, IP, and Ethernet traffic and using the **Statistics** tool to calculate the average sizes.

Background Network Traffic

I started a Wireshark capture while running my daily applications (Outlook, Spotify, OneDrive, WhatsApp, Discord, Browser, etc.) and observed the following types of network traffic:

- **DNS queries**: Resolving domain names for various applications.
- **HTTP/HTTPS traffic**: Web browsing, API calls, and updates.
- **TCP connections**: Establishing and maintaining connections for applications like Outlook and Discord.
- **UDP traffic**: Used by Spotify for streaming and by Discord for voice chat.
- **ARP requests**: Resolving MAC addresses for devices on the local network.
- **ICMP packets**: Ping requests and responses for network diagnostics.
- **Background OS traffic**: Windows Update checks, NTP (time synchronization), and other system-related communications.

DHCP in Saxion vs. Home Network

- **Saxion Network**:
    1. My device sends a **DHCP Discover** message broadcast to the network.
    2. The Saxion DHCP server responds with a **DHCP Offer**, providing an available IP address.
    3. My device sends a **DHCP Request** to confirm the offered IP address.
    4. The DHCP server sends a **DHCP Acknowledgment**, finalizing the IP assignment.

    - The subnet mask provided by Saxion is typically **255.255.252.0** which defines the range of IP addresses available in the subnet.

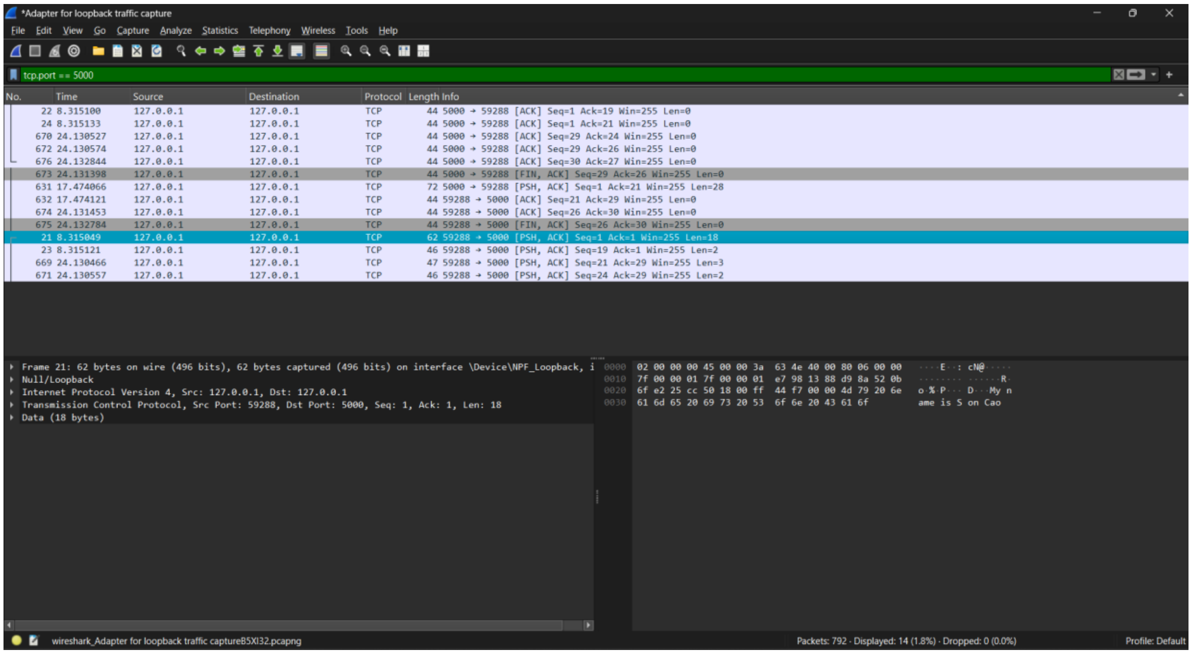
- **Home Network**:

    1. My device sends a **DHCP Discover** message to the home router.
    2. The router responds with a **DHCP Offer**, providing an IP address from the home network's range
    3. My device sends a **DHCP Request** to confirm the IP address.
    4. The router sends a **DHCP Acknowledgment**, finalizing the assignment.

    - The subnet mask at home is **255.255.240.0**, which allows for 240 devices in the subnet.

- **Function of a Subnet Mask**:
  The subnet mask defines the network and host portions of an IP address. It helps devices determine whether a destination IP address is within the same local network or requires routing through a gateway.

assignment week 4 & 5: TCP client & server


*Adapter for loopback traffic capture
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.port == 5000

```
No.   Time       Source      Destination  Protocol Length Info
 22 8.315100     127.0.0.1   127.0.0.1    TCP      44 5000 → 59288 [ACK] Seq=1 Ack=19 Win=255 Len=0
 24 8.315133     127.0.0.1   127.0.0.1    TCP      44 5000 → 59288 [ACK] Seq=1 Ack=21 Win=255 Len=0
670 24.130527    127.0.0.1   127.0.0.1    TCP      44 5000 → 59288 [ACK] Seq=29 Ack=24 Win=255 Len=0
672 24.130574    127.0.0.1   127.0.0.1    TCP      44 5000 → 59288 [ACK] Seq=29 Ack=26 Win=255 Len=0
676 24.132844    127.0.0.1   127.0.0.1    TCP      44 5000 → 59288 [ACK] Seq=30 Ack=27 Win=255 Len=0
673 24.131398    127.0.0.1   127.0.0.1    TCP      44 5000 → 59288 [FIN, ACK] Seq=29 Ack=26 Win=255 Len=0
631 17.474066    127.0.0.1   127.0.0.1    TCP      72 5000 → 59288 [PSH, ACK] Seq=1 Ack=21 Win=255 Len=28
632 17.474121    127.0.0.1   127.0.0.1    TCP      44 59288 → 5000 [ACK] Seq=21 Ack=29 Win=255 Len=0
674 24.131453    127.0.0.1   127.0.0.1    TCP      44 59288 → 5000 [ACK] Seq=26 Ack=30 Win=255 Len=0
675 24.132784    127.0.0.1   127.0.0.1    TCP      44 59288 → 5000 [FIN, ACK] Seq=26 Ack=30 Win=255 Len=0
 21 8.315049     127.0.0.1   127.0.0.1    TCP      62 59288 → 5000 [PSH, ACK] Seq=1 Ack=1 Win=255 Len=18
 23 8.315121     127.0.0.1   127.0.0.1    TCP      46 59288 → 5000 [PSH, ACK] Seq=19 Ack=1 Win=255 Len=2
669 24.130466    127.0.0.1   127.0.0.1    TCP      47 59288 → 5000 [PSH, ACK] Seq=21 Ack=29 Win=255 Len=3
671 24.130557    127.0.0.1   127.0.0.1    TCP      46 59288 → 5000 [PSH, ACK] Seq=24 Ack=29 Win=255 Len=2
```

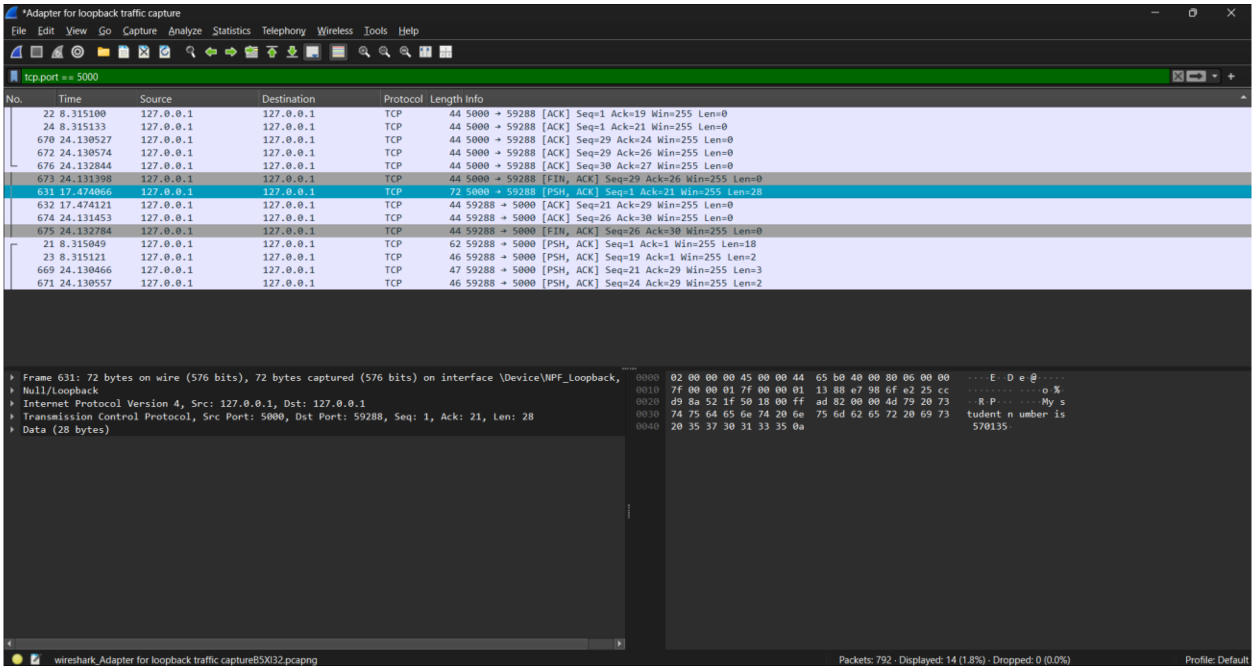
```
› Frame 21: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface \Device\NPF_Loopback, i
› Null/Loopback
› Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
› Transmission Control Protocol, Src Port: 59288, Dst Port: 5000, Seq: 1, Ack: 1, Len: 18
› Data (18 bytes)
```

```
0000  02 00 00 00 45 00 00 3a  63 4e 40 00 80 06 00 00   ····E··: cN@·····
0010  7f 00 00 01 7f 00 00 01  e7 98 13 88 d9 8a 52 0b   ········ ······R·
0020  6f e2 25 cc 50 18 00 ff  44 f7 00 00 4d 79 20 6e   o·%·P··· D···My n
0030  61 6d 65 20 69 73 20 53  6f 6e 20 43 61 6f         ame is S on Cao
```

wireshark_Adapter for loopback traffic captureB5XI32.pcapng

Packets: 792 · Displayed: 14 (1.8%) · Dropped: 0 (0.0%)

Profile: Default


*Adapter for loopback traffic capture
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.port == 5000

```
No.   Time       Source      Destination  Protocol Length Info
 22 8.315100     127.0.0.1   127.0.0.1    TCP      44 5000 → 59288 [ACK] Seq=1 Ack=19 Win=255 Len=0
 24 8.315133     127.0.0.1   127.0.0.1    TCP      44 5000 → 59288 [ACK] Seq=1 Ack=21 Win=255 Len=0
670 24.130527    127.0.0.1   127.0.0.1    TCP      44 5000 → 59288 [ACK] Seq=29 Ack=24 Win=255 Len=0
672 24.130574    127.0.0.1   127.0.0.1    TCP      44 5000 → 59288 [ACK] Seq=29 Ack=26 Win=255 Len=0
676 24.132844    127.0.0.1   127.0.0.1    TCP      44 5000 → 59288 [ACK] Seq=30 Ack=27 Win=255 Len=0
673 24.131398    127.0.0.1   127.0.0.1    TCP      44 5000 → 59288 [FIN, ACK] Seq=29 Ack=26 Win=255 Len=0
631 17.474066    127.0.0.1   127.0.0.1    TCP      72 5000 → 59288 [PSH, ACK] Seq=1 Ack=21 Win=255 Len=28
632 17.474121    127.0.0.1   127.0.0.1    TCP      44 59288 → 5000 [ACK] Seq=21 Ack=29 Win=255 Len=0
674 24.131453    127.0.0.1   127.0.0.1    TCP      44 59288 → 5000 [ACK] Seq=26 Ack=30 Win=255 Len=0
675 24.132784    127.0.0.1   127.0.0.1    TCP      44 59288 → 5000 [FIN, ACK] Seq=26 Ack=30 Win=255 Len=0
 21 8.315049     127.0.0.1   127.0.0.1    TCP      62 59288 → 5000 [PSH, ACK] Seq=1 Ack=1 Win=255 Len=18
 23 8.315121     127.0.0.1   127.0.0.1    TCP      46 59288 → 5000 [PSH, ACK] Seq=19 Ack=1 Win=255 Len=2
669 24.130466    127.0.0.1   127.0.0.1    TCP      47 59288 → 5000 [PSH, ACK] Seq=21 Ack=29 Win=255 Len=3
671 24.130557    127.0.0.1   127.0.0.1    TCP      46 59288 → 5000 [PSH, ACK] Seq=24 Ack=29 Win=255 Len=2
```

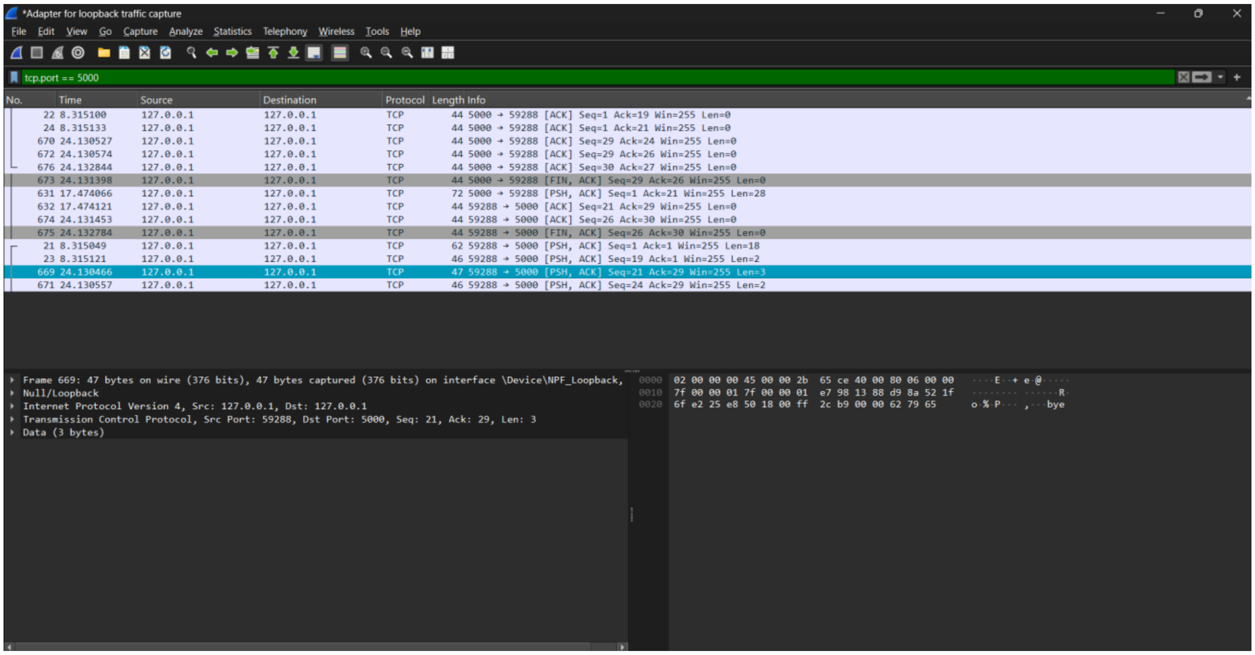
```
› Frame 631: 72 bytes on wire (576 bits), 72 bytes captured (576 bits) on interface \Device\NPF_Loopback,
› Null/Loopback
› Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
› Transmission Control Protocol, Src Port: 5000, Dst Port: 59288, Seq: 1, Ack: 21, Len: 28
› Data (28 bytes)
```

```
0000  02 00 00 00 45 00 00 44  65 b0 40 00 80 06 00 00   ····E··D e·@·····
0010  7f 00 00 01 7f 00 00 01  13 88 e7 98 6f e2 25 cc   ········ ····o·%·
0020  d9 8a 52 1f 50 18 00 ff  ad 82 00 00 4d 79 20 73   ··R·P··· ····My s
0030  74 75 64 65 6e 74 20 6e  75 6d 62 65 72 20 69 73   tudent n umber is
0040  20 35 37 30 31 33 35 0a                             570135·
```

wireshark_Adapter for loopback traffic captureB5XI32.pcapng

Packets: 792 · Displayed: 14 (1.8%) · Dropped: 0 (0.0%)

Profile: Default



*Adapter for loopback traffic capture
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

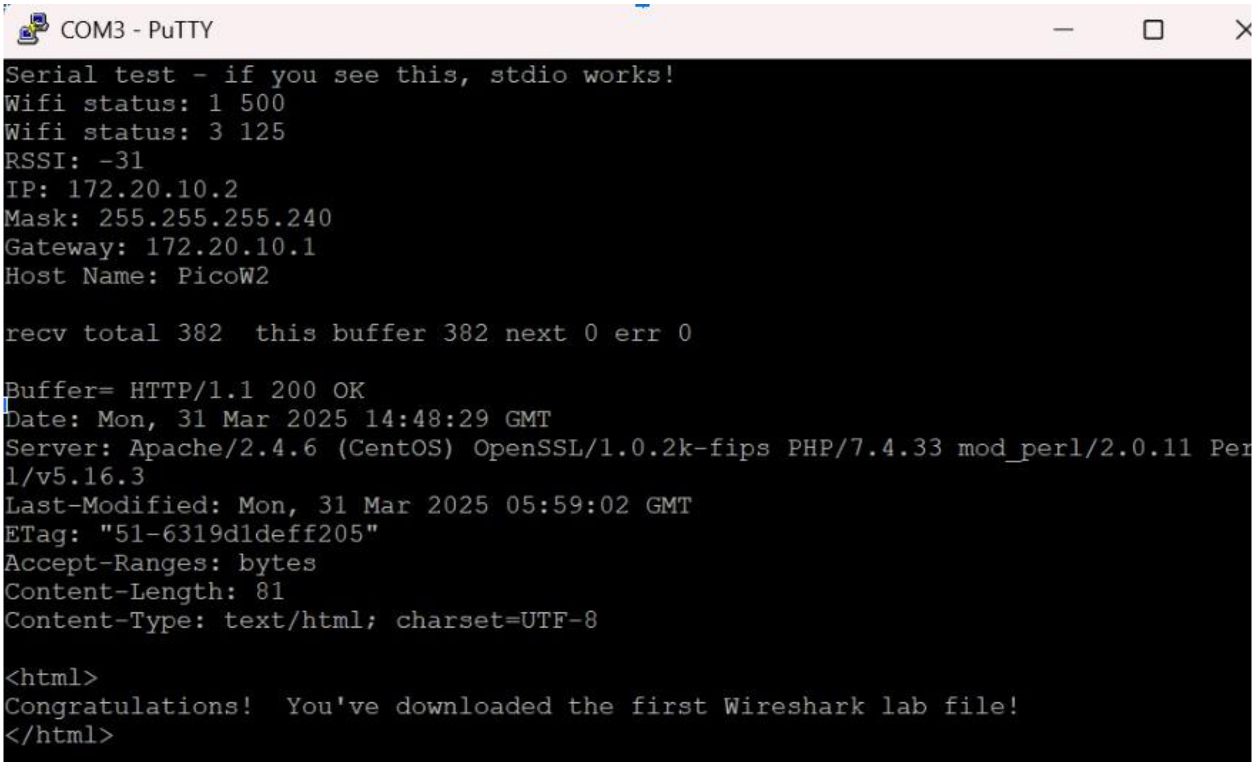
tcp.port == 5000

```
No.   Time       Source      Destination  Protocol Length Info
   22 8.315100   127.0.0.1   127.0.0.1    TCP      44 5000 → 59288 [ACK] Seq=1 Ack=19 Win=255 Len=0
   24 8.315133   127.0.0.1   127.0.0.1    TCP      44 5000 → 59288 [ACK] Seq=1 Ack=21 Win=255 Len=0
  670 24.130527  127.0.0.1   127.0.0.1    TCP      44 5000 → 59288 [ACK] Seq=29 Ack=24 Win=255 Len=0
  672 24.130574  127.0.0.1   127.0.0.1    TCP      44 5000 → 59288 [ACK] Seq=29 Ack=26 Win=255 Len=0
  676 24.132844  127.0.0.1   127.0.0.1    TCP      44 5000 → 59288 [ACK] Seq=30 Ack=27 Win=255 Len=0
  673 24.131398  127.0.0.1   127.0.0.1    TCP      44 5000 → 59288 [FIN, ACK] Seq=29 Ack=26 Win=255 Len=0
  631 17.474066  127.0.0.1   127.0.0.1    TCP      72 5000 → 59288 [PSH, ACK] Seq=1 Ack=21 Win=255 Len=28
  632 17.474121  127.0.0.1   127.0.0.1    TCP      44 59288 → 5000 [ACK] Seq=21 Ack=29 Win=255 Len=0
  674 24.131453  127.0.0.1   127.0.0.1    TCP      44 59288 → 5000 [ACK] Seq=26 Ack=30 Win=255 Len=0
  675 24.132784  127.0.0.1   127.0.0.1    TCP      44 59288 → 5000 [FIN, ACK] Seq=26 Ack=30 Win=255 Len=0
   21 8.315049   127.0.0.1   127.0.0.1    TCP      62 59288 → 5000 [PSH, ACK] Seq=1 Ack=1 Win=255 Len=18
   23 8.315121   127.0.0.1   127.0.0.1    TCP      46 59288 → 5000 [PSH, ACK] Seq=19 Ack=1 Win=255 Len=2
  669 24.130466  127.0.0.1   127.0.0.1    TCP      47 59288 → 5000 [PSH, ACK] Seq=21 Ack=29 Win=255 Len=3
  671 24.130557  127.0.0.1   127.0.0.1    TCP      46 59288 → 5000 [PSH, ACK] Seq=24 Ack=29 Win=255 Len=2
```

```
› Frame 669: 47 bytes on wire (376 bits), 47 bytes captured (376 bits) on interface \Device\NPF_Loopback,
› Null/Loopback
› Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
› Transmission Control Protocol, Src Port: 59288, Dst Port: 5000, Seq: 21, Ack: 29, Len: 3
› Data (3 bytes)
```

```
0000  02 00 00 00 45 00 00 2b  65 ce 40 00 80 06 00 00   ····E··+ e·@·····
0010  7f 00 00 01 7f 00 00 01  e7 98 13 88 d9 8a 52 1f   ········ ······R·
0020  6f e2 25 e8 50 18 00 ff  2c b9 00 00 62 79 65      o·%·P··· ,···bye
```

assignment week 6 & 7: pi pico networking



```
Serial test - if you see this, stdio works!
Wifi status: 1 500
Wifi status: 3 125
RSSI: -31
IP: 172.20.10.2
Mask: 255.255.255.240
Gateway: 172.20.10.1
Host Name: PicoW2

recv total 382  this buffer 382 next 0 err 0

Buffer= HTTP/1.1 200 OK
Date: Mon, 31 Mar 2025 14:48:29 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.33 mod_perl/2.0.11 Per
1/v5.16.3
Last-Modified: Mon, 31 Mar 2025 05:59:02 GMT
ETag: "51-6319d1deff205"
Accept-Ranges: bytes
Content-Length: 81
Content-Type: text/html; charset=UTF-8

<html>
Congratulations!  You've downloaded the first Wireshark lab file!
</html>
```

For this assignment, I successfully set up WiFi connectivity on my Raspberry Pi Pico W and retrieved a webpage from a remote server.

Setup Process:

1. Downloaded and extracted the provided pico-wifi-example.zip files
2. Modified credentials.h with my WiFi network details (SSID and password)

3. Set up the project structure with the necessary CMake configuration
4. Built and flashed the program to my Pico W

Challenges Faced:

1. Initially had trouble with the Pico not connecting - realized I needed to update the SDK and firmware
2. First attempt failed because I hadn't properly configured the PICO_BOARD variable in CMake