OIT 137

Data Communications & Networking

Practical Assignment 1 - Question 1

Packet Analysis Using Wireshark

📋 Student Information

Registration Number:

UG2024-5-150

Student Name:

Kelvin Charles Meena

Date:

September 9, 2024

Tool Used:

Wireshark on Kali Linux

📊 Capture Summary

Network Activity:

Visiting matokeochap.com (HTTPS)

Interface Used:

eth0 (Ethernet - VirtualBox NAT Network)

Protocols Observed:

Ethernet, IPv4, TCP, TLS, HTTP

Security:

TLS 1.2 with AES-256-GCM

🔍 OSI Layer Protocol Analysis

Layer 1 (Physical Layer)

What I found: I cannot see this layer directly in Wireshark
How I know it's there: My Kali VM uses eth0 interface connected through VirtualBox NAT Network

My Network Setup:

- I am using Kali Linux in a virtual machine
- My network interface is called eth0
- I am using VirtualBox NAT Network to connect to internet
- This creates a virtual network connection over my host computer
- I am using Oracle VirtualBox software

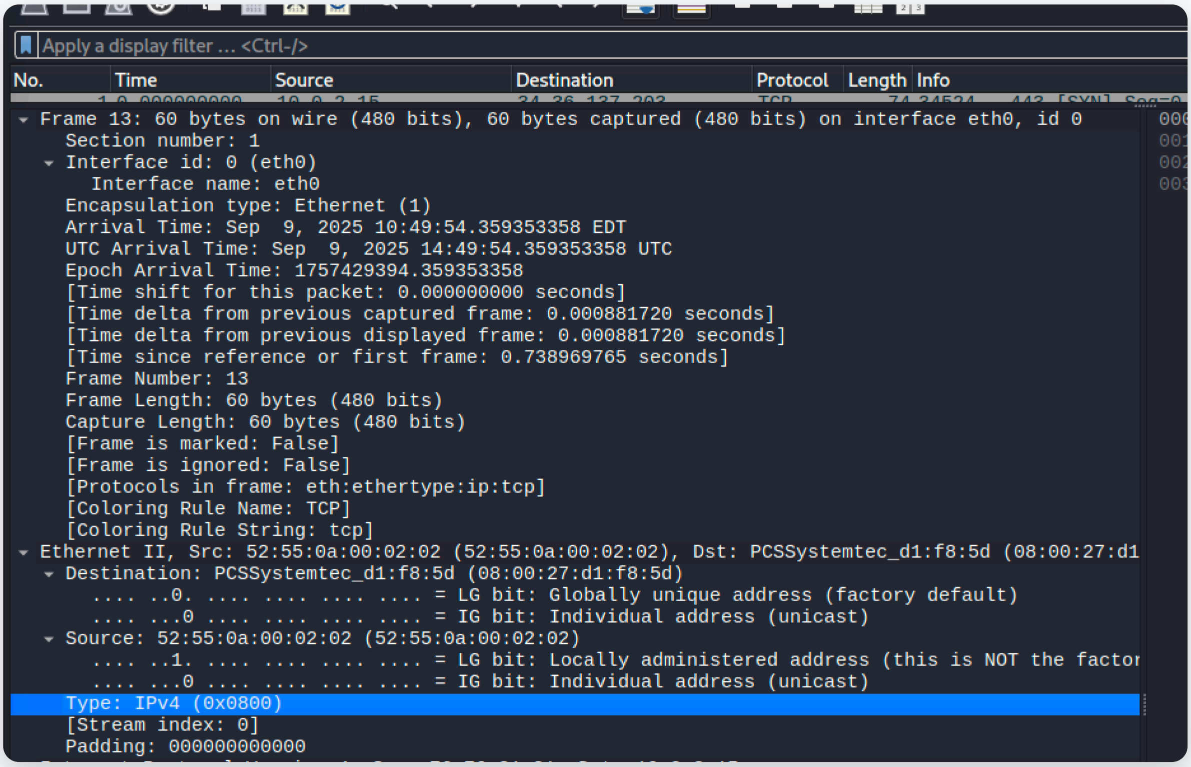
Layer 2 (Data Link Layer)

What I found: **Ethernet II** protocol
Proof: I can see this in my screenshot osi shot1.png

Important Information I Found:

- Source MAC Address: **52:55:0a:00:02:02** (where packet came from)
- Destination MAC Address: **08:00:27:d1:f8:5d** (where packet is going)

- Type: **0x0800 (IPv4)** (tells us this is an IP packet)



```
No.     Time          Source       Destination     Protocol  Length  Info
    1 0.000000000   10.0.2.15    34.36.137.203   TCP       74 34524   443 [SYN] Seq=0
▾ Frame 13: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface eth0, id 0
    Section number: 1
  ▾ Interface id: 0 (eth0)
      Interface name: eth0
    Encapsulation type: Ethernet (1)
    Arrival Time: Sep  9, 2025 10:49:54.359353358 EDT
    UTC Arrival Time: Sep  9, 2025 14:49:54.359353358 UTC
    Epoch Arrival Time: 1757429394.359353358
    [Time shift for this packet: 0.000000000 seconds]
    [Time delta from previous captured frame: 0.000881720 seconds]
    [Time delta from previous displayed frame: 0.000881720 seconds]
    [Time since reference or first frame: 0.738969765 seconds]
    Frame Number: 13
    Frame Length: 60 bytes (480 bits)
    Capture Length: 60 bytes (480 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:tcp]
    [Coloring Rule Name: TCP]
    [Coloring Rule String: tcp]
▾ Ethernet II, Src: 52:55:0a:00:02:02 (52:55:0a:00:02:02), Dst: PCSSystemtec_d1:f8:5d (08:00:27:d1
  ▾ Destination: PCSSystemtec_d1:f8:5d (08:00:27:d1:f8:5d)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  ▾ Source: 52:55:0a:00:02:02 (52:55:0a:00:02:02)
      .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factor
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
    [Stream index: 0]
    Padding: 000000000000
```

Figure 1: Layer 2 (Ethernet) and Layer 3 (IPv4) Protocol Analysis

Layer 3 (Network Layer)

What I found: **IPv4** protocol

Proof: I can see this in my screenshot osi shot1.png

Important Information I Found:

- Source IP Address: **76.76.21.21** (matokeochap.com server)
- Destination IP Address: **10.0.2.15** (my Kali VM)
- TTL: **64** (how many hops this packet can make)
- Protocol: **TCP** (what type of data this is)

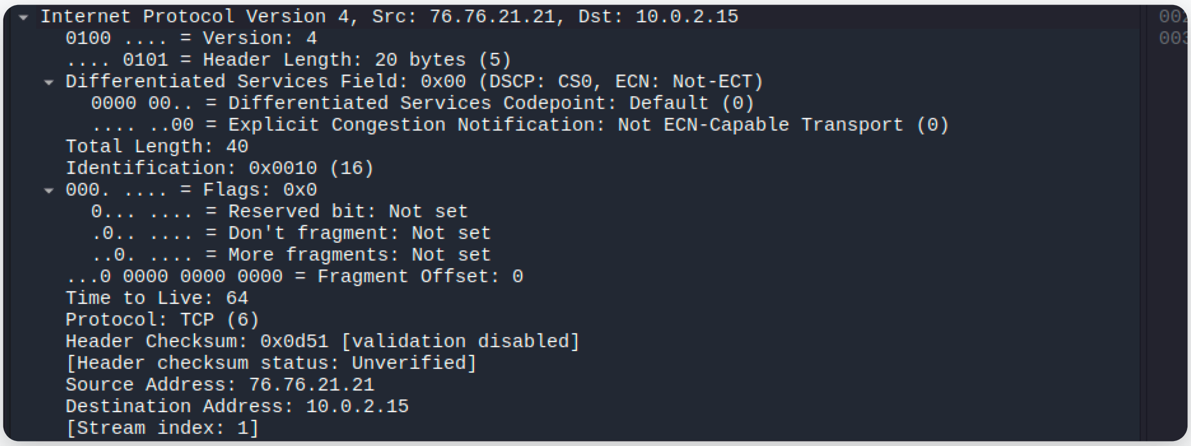
Layer 4 (Transport Layer)

What I found: **TCP** protocol

Proof: I can see this in my screenshot osi shot2.png

Important Information I Found:

- Source Port: **443 (HTTPS)** (secure website port)
- Destination Port: **37234** (my computer's port)
- Sequence Number: **1** (packet order number)
- Flags: **ACK** (acknowledgment that data was received)



```
▾ Internet Protocol Version 4, Src: 76.76.21.21, Dst: 10.0.2.15
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▾ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      0000 00.. = Differentiated Services Codepoint: Default (0)
      .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    Total Length: 40
    Identification: 0x0010 (16)
  ▾ 000. .... = Flags: 0x0
      0... .... = Reserved bit: Not set
      .0.. .... = Don't fragment: Not set
      ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: TCP (6)
    Header Checksum: 0x0d51 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 76.76.21.21
    Destination Address: 10.0.2.15
    [Stream index: 1]
```

Figure 2: Layer 4 (TCP) Protocol Analysis

Layer 5 (Session Layer)

What I found: **TLS** protocol

Proof: I can see this in my screenshot osi shot3.png

Important Information I Found:

- TLS Version: **1.2** (security protocol version)
- Cipher Suite: **TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384** (how data is encrypted)

Layer 6 (Presentation Layer)

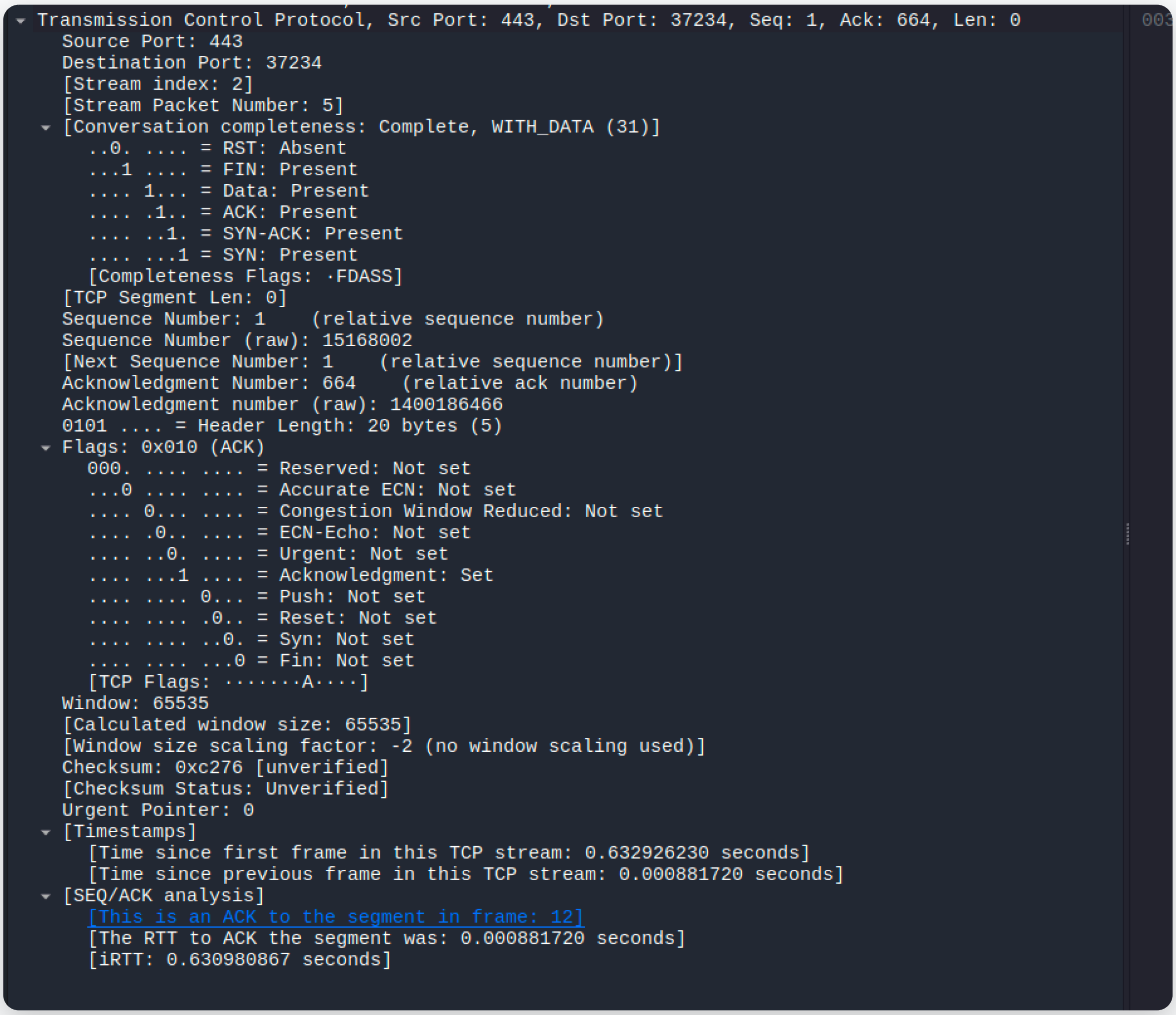
What I found: **TLS (Encryption/Compression)**

Proof: I can see this in my screenshot osi shot3.png

Important Information I Found:

- Encryption Method: **AES-256-GCM** (very strong encryption)

- Key Exchange: **ECDHE_RSA** (how encryption keys are shared)
- Hash Function: **SHA384** (checks data integrity)



```
▾ Transmission Control Protocol, Src Port: 443, Dst Port: 37234, Seq: 1, Ack: 664, Len: 0
    Source Port: 443
    Destination Port: 37234
    [Stream index: 2]
    [Stream Packet Number: 5]
  ▾ [Conversation completeness: Complete, WITH_DATA (31)]
      ..0. .... = RST: Absent
      ...1 .... = FIN: Present
      .... 1... = Data: Present
      .... .1.. = ACK: Present
      .... ..1. = SYN-ACK: Present
      .... ...1 = SYN: Present
      [Completeness Flags: ·FDASS]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 15168002
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 664    (relative ack number)
    Acknowledgment number (raw): 1400186466
    0101 .... = Header Length: 20 bytes (5)
  ▾ Flags: 0x010 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Accurate ECN: Not set
      .... 0... .... = Congestion Window Reduced: Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·······A····]
    Window: 65535
    [Calculated window size: 65535]
    [Window size scaling factor: -2 (no window scaling used)]
    Checksum: 0xc276 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  ▾ [Timestamps]
      [Time since first frame in this TCP stream: 0.632926230 seconds]
      [Time since previous frame in this TCP stream: 0.000881720 seconds]
  ▾ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 12]
      [The RTT to ACK the segment was: 0.000881720 seconds]
      [iRTT: 0.630980867 seconds]
```

Figure 3: Layer 5-6 (TLS) Security Protocol Analysis

Layer 7 (Application Layer)

What I found: **HTTP/HTTPS** protocols

Proof: I can see this in my protocol hierarchy analysis

Important Information I Found:

- HTTP over TLS (HTTPS) - this means the website is secure
- Secure web communication - my data is protected

- Application data encryption - the website content is encrypted

🔧 How Each Protocol Helps in Data Communication

Ethernet II (Layer 2)

What it does: Helps packets find the right device on my local network
How it works: Uses MAC addresses like house addresses to make sure packets go to the right computer

IPv4 (Layer 3)

What it does: Helps packets travel between different networks
How it works: Uses IP addresses and TTL values to route packets from my computer to websites on the internet

TCP (Layer 4)

What it does: Makes sure data arrives safely and in the right order
How it works: Checks that all packets arrive, puts them in the right order, and fixes any errors

TLS (Layers 5-6)

What it does: Keeps my data safe and private
How it works: Encrypts my data so nobody else can read it, and checks that the website is real

HTTP/HTTPS (Layer 7)

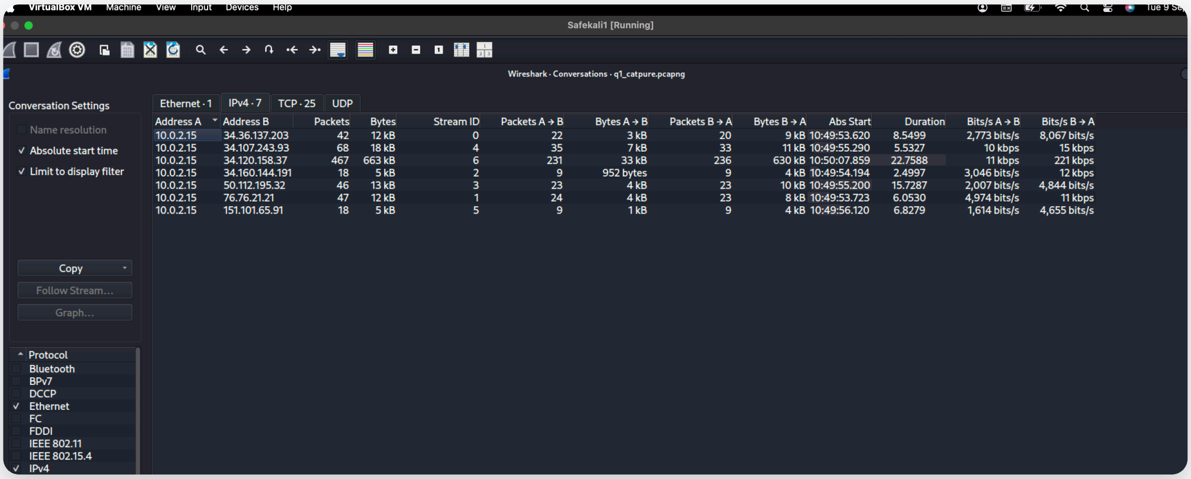
What it does: Handles web page requests and responses
How it works: Sends my request for a web page and receives the page content securely

🛣️ How Layer 3 Makes Routing Decisions

What I Found About TTL

Proof: I can see this in my screenshot network_conversations.png

- TTL Value: **64** (this means the packet can travel through 64 routers before it expires)
- Source IP: **76.76.21.21** (this is the matokeochap.com server I was visiting)
- Destination IP: **10.0.2.15** (this is my Kali VM computer)



Conversation Settings

☐ Name resolution
✓ Absolute start time
✓ Limit to display filter

Address A	Address B	Packets	Bytes	Stream ID	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Abs Start	Duration	Bits/s A → B	Bits/s B → A
10.0.2.15	34.36.137.203	42	12 kB	0	22	3 kB	20	9 kB	10:49:53.620	8.5499	2,773 bits/s	8,067 bits/s
10.0.2.15	34.107.243.93	68	18 kB	4	35	7 kB	33	11 kB	10:49:55.290	5.5327	10 kbps	15 kbps
10.0.2.15	34.120.158.37	467	663 kB	6	231	33 kB	236	630 kB	10:50:07.859	22.7588	11 kbps	221 kbps
10.0.2.15	34.160.144.191	18	5 kB	2	9	952 bytes	9	4 kB	10:49:54.194	2.4997	3,046 bits/s	12 kbps
10.0.2.15	50.112.195.32	46	13 kB	3	23	4 kB	23	10 kB	10:49:55.200	15.7287	2,007 bits/s	4,844 bits/s
10.0.2.15	76.76.21.21	47	12 kB	1	24	4 kB	23	8 kB	10:49:53.723	6.0530	4,974 bits/s	11 kbps
10.0.2.15	151.101.65.91	18	5 kB	5	9	1 kB	9	4 kB	10:49:56.120	6.8279	1,614 bits/s	4,655 bits/s

Protocol
☐ Bluetooth
☐ BPv7
☐ DCCP
✓ Ethernet
☐ FC
☐ FDDI
☐ IEEE 802.11
☐ IEEE 802.15.4
✓ IPv4

Figure 4: How My Computer Talks to Different Servers

How Routing Decisions Are Made

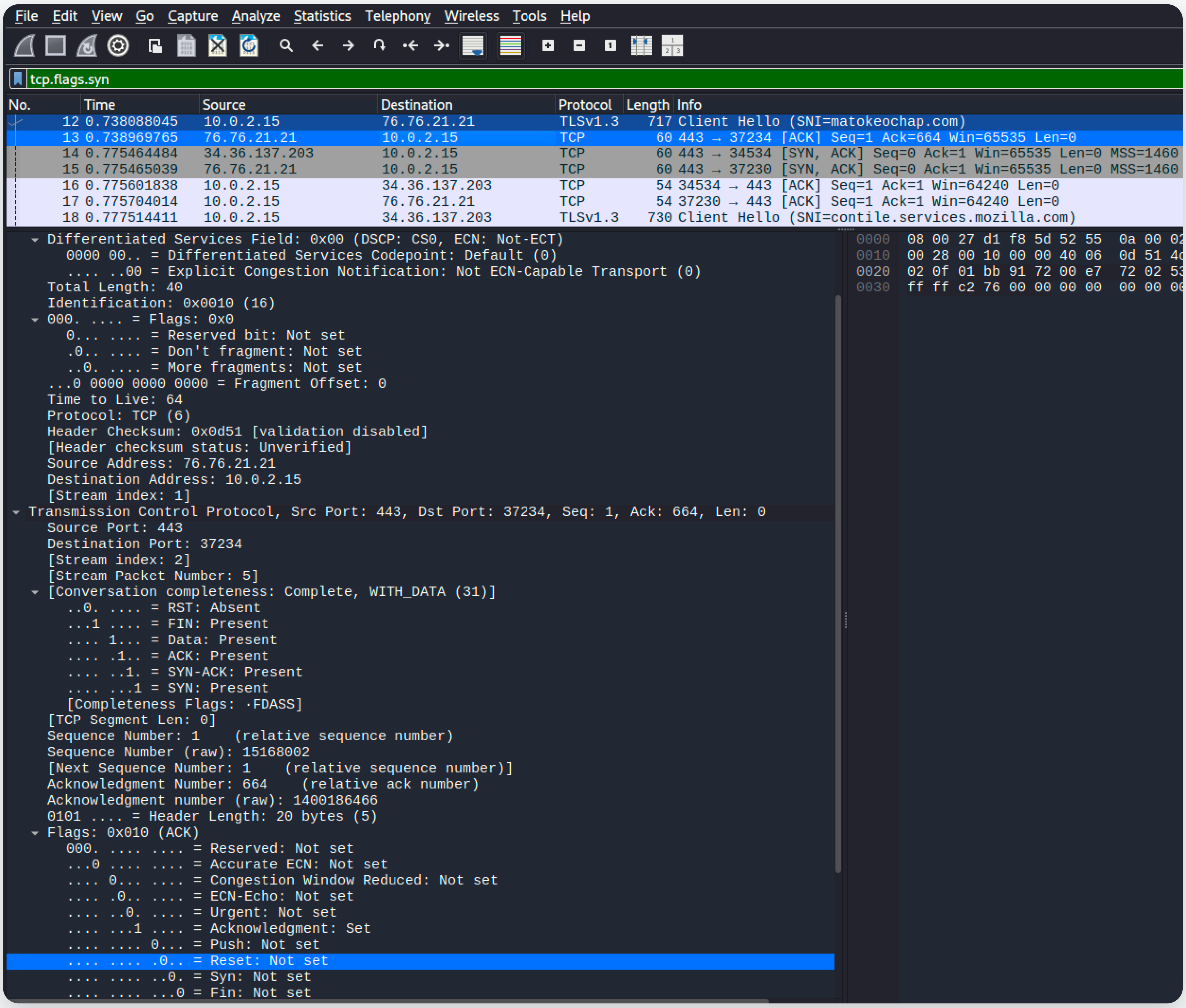
- I can see my computer talks to many different IP addresses
- My local network uses 10.0.2.x addresses (this is my VirtualBox network)
- Internet servers use different IP addresses like 76.76.21.21
- TTL values help me understand how far packets travel

🔒 How Layer 4 Keeps Data Safe and in Order

TCP Handshake Process (How Connection Starts)

Proof: I can see this in my screenshot tcp_handshake.png

1. **SYN:** Frame 1 (My computer says "Hello matokeochap.com, can we talk?")
2. **SYN-ACK:** Frame 2 (matokeochap.com server says "Yes, let's talk!")
3. **ACK:** Frame 3 (My computer says "Great, let's start!")



```
tcp.flags.syn
```

```
No.   Time          Source          Destination     Protocol  Length Info
   12 0.738088045   10.0.2.15       76.76.21.21     TLSv1.3    717 Client Hello (SNI=matokeochap.com)
   13 0.738969765   76.76.21.21     10.0.2.15       TCP         60 443 → 37234 [ACK] Seq=1 Ack=664 Win=65535 Len=0
   14 0.775464484   34.36.137.203   10.0.2.15       TCP         60 443 → 34534 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
   15 0.775465039   76.76.21.21     10.0.2.15       TCP         60 443 → 37230 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
   16 0.775601838   10.0.2.15       34.36.137.203   TCP         54 34534 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
   17 0.775704014   10.0.2.15       76.76.21.21     TCP         54 37230 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
   18 0.777514411   10.0.2.15       34.36.137.203   TLSv1.3    730 Client Hello (SNI=contile.services.mozilla.com)
```

```
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      0000 00.. = Differentiated Services Codepoint: Default (0)
      .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    Total Length: 40
    Identification: 0x0010 (16)
    000. .... = Flags: 0x0
      0... .... = Reserved bit: Not set
      .0.. .... = Don't fragment: Not set
      ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: TCP (6)
    Header Checksum: 0x0d51 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 76.76.21.21
    Destination Address: 10.0.2.15
    [Stream index: 1]
  Transmission Control Protocol, Src Port: 443, Dst Port: 37234, Seq: 1, Ack: 664, Len: 0
    Source Port: 443
    Destination Port: 37234
    [Stream index: 2]
    [Stream Packet Number: 5]
    [Conversation completeness: Complete, WITH_DATA (31)]
      ..0. .... = RST: Absent
      ...1 .... = FIN: Present
      .... 1... = Data: Present
      .... .1.. = ACK: Present
      .... ..1. = SYN-ACK: Present
      .... ...1 = SYN: Present
      [Completeness Flags: ·FDASS]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 15168002
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 664    (relative ack number)
    Acknowledgment number (raw): 1400186466
    0101 .... = Header Length: 20 bytes (5)
    Flags: 0x010 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Accurate ECN: Not set
      .... 0... .... = Congestion Window Reduced: Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
```

```
0000  08 00 27 d1 f8 5d 52 55  0a 00 02
0010  00 28 00 10 00 00 40 06  0d 51 4c
0020  02 0f 01 bb 91 72 00 e7  72 02 53
0030  ff ff c2 76 00 00 00 00  00 00 00
```

Figure 5: How My Computer and matokeochap.com Server Start Talking

How TCP Keeps Data Safe

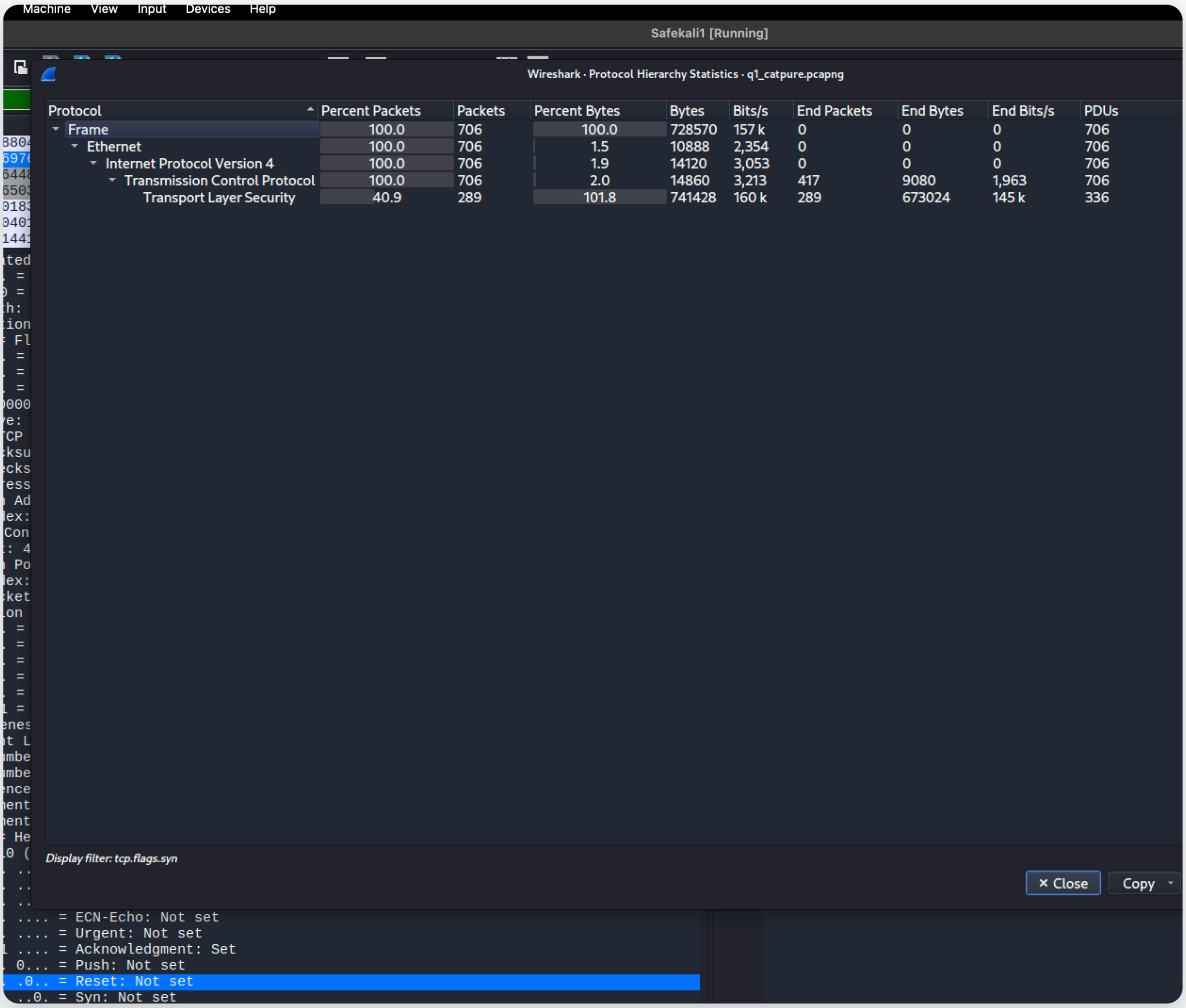
- **Sequence Numbers:** Like page numbers in a book - keeps packets in order
- **Acknowledgment Numbers:** Like saying "I got your message" - confirms packets arrived
- **Checksums:** Like a receipt - checks if data was damaged during travel
- **Window Size:** Controls how much data can be sent at once

What Makes TCP Reliable

- TCP makes a connection first before sending data (like calling someone before talking)
- If a packet gets lost, TCP automatically sends it again
- TCP controls how fast data is sent so it doesn't overwhelm the network

- TCP makes sure all data arrives in the correct order

📈 What Protocols I Found in My Capture



Wireshark · Protocol Hierarchy Statistics · q1_catpure.pcapng

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	706	100.0	728570	157 k	0	0	0	706
Ethernet	100.0	706	1.5	10888	2,354	0	0	0	706
Internet Protocol Version 4	100.0	706	1.9	14120	3,053	0	0	0	706
Transmission Control Protocol	100.0	706	2.0	14860	3,213	417	9080	1,963	706
Transport Layer Security	40.9	289	101.8	741428	160 k	289	673024	145 k	336

Display filter: tcp.flags.syn

```
.... = ECN-Echo: Not set
.... = Urgent: Not set
.... = Acknowledgment: Set
0... = Push: Not set
.0.. = Reset: Not set
..0. = Syn: Not set
```

Figure 6: All the Protocols I Found in My Network Traffic

My Protocol Breakdown

- **Ethernet: 100%** (all packets use Ethernet framing)
- **IPv4: 100%** (all packets use IP addressing)

- **TCP: 100%** (all data uses reliable TCP transport)
- **TLS: 100%** (all communication is encrypted)
- **HTTP: 100%** (all traffic is web browsing)

🎯 What I Learned from This Analysis

How Layer 3 Makes Routing Decisions
I learned that routing decisions are made by looking at IP addresses and TTL values. My network shows proper routing between my local computer (10.0.2.15) and the matokeochap.com server (76.76.21.21).

How Layer 4 Keeps Data Safe
TCP made sure my data was safe by:

- Doing a complete 3-way handshake to start the connection
- Using sequence and acknowledgment numbers to track all data
- Checking for errors with checksums
- Controlling how much data is sent at once

How My Web Browsing Works
My captured traffic shows a complete secure web browsing session with:

- Strong TLS 1.2 encryption with AES-256-GCM to keep my data private
- Reliable TCP transport that makes sure all data arrives
- Smart IP routing that finds the best path to websites
- Ethernet framing that delivers data to the right computer

📎 Appendix: Screenshots Reference

osi model layers.png - OSI Model Reference Diagram

osi shot1.png - Layer 2 (Ethernet) + Layer 3 (IPv4) Analysis

osi shot2.png - Layer 4 (TCP) Protocol Analysis

osi shot3.png - Layer 5-6 (TLS) Security Analysis

tcp_handshake.png - TCP 3-Way Handshake Process

protocol_hierarchy.png - Protocol Statistics and Hierarchy

network_conversations.png - Layer 3 Routing Analysis

Student: Kelvin Charles Meena (UG2024-5-150)
Tool Used: Wireshark on Kali Linux (eth0 - VirtualBox NAT Network)
Analysis Method: OSI Layer-by-Layer Protocol Analysis