

OSI Model,TCP/IP Model and Wireshark

Packet Analysis

1. OSI Layer Explanations

Layer 1 – Physical Layer

This layer is the quiet workhorse of the network world. It deals with the raw signals—electrical pulses, light flashes, or radio waves—that travel through cables and air. No understanding of data happens here; it's just bits marching in a steady rhythm. Ethernet cables, fiber optics, Wi-Fi radio signals, voltage levels, connectors, and physical topologies belong here. You can imagine it like the actual road on which vehicles travel: without the asphalt or the rails, nothing else can move.

Layer 2 – Data Link Layer

Here the network gains awareness of neighbors. The Data Link Layer shapes raw bits into frames, attaches MAC addresses, and checks for errors using CRC. Switches, NICs, VLANs, ARP, and protocols like Ethernet and PPP live here. It's the traffic manager inside a city block: it ensures local deliveries reach the right door, even if the rest of the world remains a mystery. Collisions are handled, and corrupted frames are quietly filtered out so higher layers don't panic.

Layer 3 – Network Layer

This is the layer where a device learns how to travel beyond its local area. The Network Layer handles logical addressing and routing—IP addresses, subnets, routers, and routing protocols like OSPF, RIP, and BGP. Packets may hop through multiple networks, each router choosing the next turn based on routing tables. It's like the postal service's long-distance system: envelopes move from city to city through a maze of interconnected routes until they find the right region.

Layer 4 – Transport Layer

Here the network learns to care about reliability and order. This layer decides whether data should arrive carefully (TCP) or quickly (UDP). It forms segments, tracks port numbers, handles re-transmissions, flow control, and ensures that scattered pieces of data get reassembled like puzzle tiles. It resembles a courier service that either delivers fragile items with great caution or drops off parcels rapidly with no signatures—depending on what the sender wants.

Layer 5 – Session Layer

The Session Layer is like the host of a long conversation. It creates, manages, and gracefully ends communication sessions between devices. It keeps track of who is speaking, when, and ensures that if a connection drops, it can be resumed. Protocols such as RPC and NetBIOS sit here. Think of it as a moderator in a meeting room who ensures that the discussion keeps flowing without two people talking over each other.

Layer 6 – Presentation Layer

This layer acts as the translator and stylist of the network. It converts data into formats that applications can understand—handling encryption, compression, and encoding. SSL/TLS (for encryption), JPEG, MP3, GIF, and text encoding schemes like ASCII and UTF-8 show up here. Picture a multilingual editor who reformats and translates a document so the final reader receives it in the exact style and language they expect.

Layer 7 – Application Layer

At the top sits the layer users actually interact with—even if they don’t realize it. It provides interfaces for email, browsing, file transfers, and other network services. Protocols like HTTP, HTTPS, FTP, SMTP, DNS, and DHCP

operate here. It's like the front counter of a service desk: people walk up, make requests, and receive what they need, while the deeper layers do the hidden heavy lifting behind the wall.

2. OSI Mnemonic

“Please Don’t Nap Too Soon, People Are Watching.”

Word	OSI Layer
Please	Physical
Don’t	Data Link
Nap	Network
Too	Transport
Soon	Session
People	Presentation
Are Watching	Application

3. OSI vs TCP/IP Model Comparison

The OSI model is a seven-layer staircase, each step carefully separating responsibilities—from raw signals at the bottom to user-facing applications at the top. It’s a detailed blueprint, almost theoretical in places, offering a clean way to study how networks behave. The TCP/IP model, meanwhile, is the practical four-layer framework the internet actually runs on. Instead of slicing tasks into fine layers, it folds related functions together—much like packing fewer but sturdier bags for a long trip.

Where OSI gives a highly structured, “textbook-perfect” perspective, TCP/IP reflects real-world protocol stacks. The upper OSI layers (Application, Presentation, Session) merge into a single Application layer in TCP/IP; the lower layers (Data Link and Physical) condense into a Network Access layer. Both models

describe the same journey of data, just with different levels of granularity.

OSI Layer	TCP/IP Layer	Explanation
Application (L7)	Application Layer	TCP/IP combines user-facing services here.
Presentation (L6)	Application Layer	Formatting, encryption, compression are also handled by the TCP/IP Application layer.
Session (L5)	Application Layer	Session management is included within TCP/IP Application protocols.
Transport (L4)	Transport Layer	TCP/UDP control reliability and ports.
Network (L3)	Internet Layer	IP, routing, addressing.
Data Link (L2)	Network Access Layer	Frames, MAC addresses, NICs.
Physical (L1)	Network Access Layer	Cables, signals, physical transmission.

4. Protocol Data Units (PDUs)

OSI Layer	PDU Name	Notes
Layer 4 – Transport	Segment (TCP) / Datagram (UDP)	TCP behaves like a careful courier (segments), while UDP tosses lightweight datagrams without ceremony.
Layer 3 – Network	Packet	Carries IP addresses and travels across multiple networks.
Layer 2 – Data Link	Frame	Wrapped with MAC addresses; perfect for local delivery.
Layer 1 – Physical	Bits	Raw 1s and 0s racing through cables or airwaves.

5. Addressing Concepts

1. MAC Address – used at Layer 2 (Data Link)

A MAC address is the hardware identifier burned into a network interface card. It's like a permanent name tag that a device carries on a local network. Layer 2 uses this address to deliver frames from one device to another within the same LAN. Switches rely on MAC addresses to decide which port should receive a frame, keeping the local traffic flowing neatly.

2. IP Address – used at Layer 3 (Network)

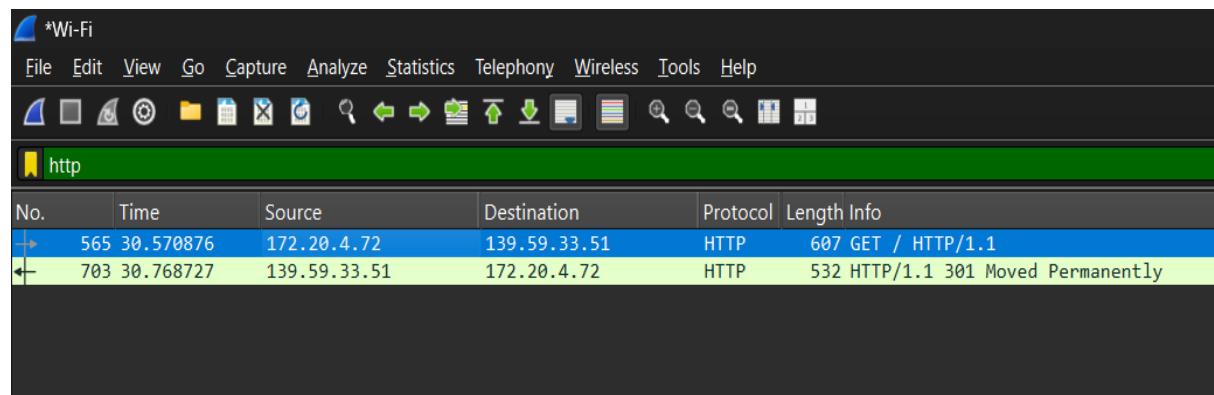
An IP address is a logical address assigned to a device so it can be located across different networks. Routers use IP addresses to move packets from one network to another, choosing paths and forwarding packets hop by hop. While MAC handles "who is next door," the IP address handles "where in the world should this go?"

3. Port Number – used at Layer 4 (Transport)

A port number identifies a specific application or service running on a device. The Transport layer uses ports to make sure data meant for a web browser doesn't accidentally land in a video player. TCP and UDP both use port numbers like tiny door labels, helping the device deliver segments or datagrams to the right process.

Part B – Wireshark Practical

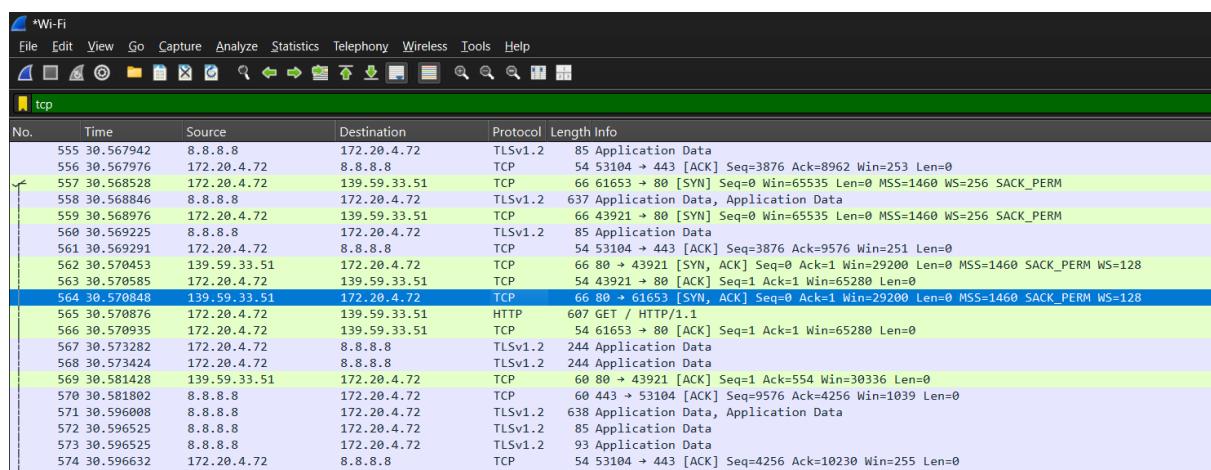
1. HTTP Traffic



No.	Time	Source	Destination	Protocol	Length	Info
565	30.570876	172.20.4.72	139.59.33.51	HTTP	607	GET / HTTP/1.1
703	30.768727	139.59.33.51	172.20.4.72	HTTP	532	HTTP/1.1 301 Moved Permanently

Frame 565: Packet, 607 bytes on wire (4856 bits), 607 bytes captured (4856 bits) on interface \Device\NPF_Ethernet II, Src: AzureWaveTec_50:52:92 (28:d0:43:50:52:92), Dst: Sophos_d1:3f:f0 (7c:5a:1c:d1:3f:f0)
Internet Protocol Version 4, Src: 172.20.4.72, Dst: 139.59.33.51
Transmission Control Protocol, Src Port: 43921, Dst Port: 80, Seq: 1, Ack: 1, Len: 553
Hypertext Transfer Protocol

2. TCP Packets



No.	Time	Source	Destination	Protocol	Length	Info
555	30.567942	8.8.8.8	172.20.4.72	TLSv1.2	85	Application Data
556	30.567976	172.20.4.72	8.8.8.8	TCP	54	53104 → 443 [ACK] Seq=3876 Ack=8962 Win=253 Len=0
557	30.568528	172.20.4.72	139.59.33.51	TCP	66	61653 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
558	30.568846	8.8.8.8	172.20.4.72	TLSv1.2	637	Application Data, Application Data
559	30.568976	172.20.4.72	139.59.33.51	TCP	66	43921 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
560	30.569225	8.8.8.8	172.20.4.72	TLSv1.2	85	Application Data
561	30.569291	172.20.4.72	8.8.8.8	TCP	54	53104 → 443 [ACK] Seq=3876 Ack=9576 Win=251 Len=0
562	30.570453	139.59.33.51	172.20.4.72	TCP	66	80 → 43921 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM WS=128
563	30.570585	172.20.4.72	139.59.33.51	TCP	54	43921 → 80 [ACK] Seq=1 Ack=1 Win=65280 Len=0
564	30.570848	139.59.33.51	172.20.4.72	TCP	66	80 → 61653 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM WS=128
565	30.570876	172.20.4.72	139.59.33.51	HTTP	607	GET / HTTP/1.1
566	30.570935	172.20.4.72	139.59.33.51	TCP	54	61653 → 80 [ACK] Seq=1 Ack=1 Win=65280 Len=0
567	30.573282	172.20.4.72	8.8.8.8	TLSv1.2	244	Application Data
568	30.573424	172.20.4.72	8.8.8.8	TLSv1.2	244	Application Data
569	30.581428	139.59.33.51	172.20.4.72	TCP	60	80 → 43921 [ACK] Seq=1 Ack=554 Win=30336 Len=0
570	30.581802	8.8.8.8	172.20.4.72	TCP	60	443 → 53104 [ACK] Seq=9576 Ack=4256 Win=1039 Len=0
571	30.596008	8.8.8.8	172.20.4.72	TLSv1.2	638	Application Data, Application Data
572	30.596525	8.8.8.8	172.20.4.72	TLSv1.2	85	Application Data
573	30.596525	8.8.8.8	172.20.4.72	TLSv1.2	93	Application Data
574	30.596632	172.20.4.72	8.8.8.8	TCP	54	53104 → 443 [ACK] Seq=4256 Ack=10230 Win=255 Len=0

```

▶ Frame 525: Packet, 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{...}
└─ Ethernet II, Src: AzureWaveTec_50:52:92 (28:d0:43:50:52:92), Dst: Sophos_d1:3f:f0 (7c:5a:1c:d1:3f:f0)
    └─ Destination: Sophos_d1:3f:f0 (7c:5a:1c:d1:3f:f0)
    └─ Source: AzureWaveTec_50:52:92 (28:d0:43:50:52:92)
        Type: IPv4 (0x0800)
        [Stream index: 0]
    ┌─ Internet Protocol Version 4, Src: 172.20.4.72, Dst: 8.8.8.8
    └─ Transmission Control Protocol, Src Port: 53104, Dst Port: 443, Seq: 3422, Ack: 7701, Len: 0
        Source Port: 53104
        Destination Port: 443
        [Stream index: 14]
        [Stream Packet Number: 98]
        └─ [Conversation completeness: Incomplete (12)]
            [TCP Segment Len: 0]
            Sequence Number: 3422      (relative sequence number)
            Sequence Number (raw): 1096974818
            [Next Sequence Number: 3422      (relative sequence number)]
            Acknowledgment Number: 7701      (relative ack number)
            Acknowledgment number (raw): 2560504998
            0101 .... = Header Length: 20 bytes (5)
        └─ Flags: 0x010 (ACK)
            Window: 251

```

3. UDP Packets

```

▶ Frame 529: Packet, 146 bytes on wire (1168 bits), 146 bytes captured (1168 bits) on interface \Device\NPF_{...}
└─ Ethernet II, Src: Sophos_d1:3f:f0 (7c:5a:1c:d1:3f:f0), Dst: AzureWaveTec_50:52:92 (28:d0:43:50:52:92)
    └─ Destination: AzureWaveTec_50:52:92 (28:d0:43:50:52:92)
    └─ Source: Sophos_d1:3f:f0 (7c:5a:1c:d1:3f:f0)
        Type: IPv4 (0x0800)
        [Stream index: 0]
    ┌─ Internet Protocol Version 4, Src: 8.8.8.8, Dst: 172.20.4.72
    └─ User Datagram Protocol, Src Port: 53, Dst Port: 55637
        Source Port: 53
        Destination Port: 55637
        Length: 112
        Checksum: 0xb0e1 [unverified]
        [Checksum Status: Unverified]
        [Stream index: 4]
        [Stream Packet Number: 2]
        └─ [Timestamps]
            UDP payload (104 bytes)
    └─ Domain Name System (response)

```

*Wi-Fi

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

udp

No.	Time	Source	Destination	Protocol	Length Info
278	29.124279	142.250.206.74	172.20.4.72	QUIC	876 Protected Payload (K0)
282	29.124279	142.250.206.74	172.20.4.72	QUIC	876 Protected Payload (K0)
288	29.125428	172.20.4.72	142.250.206.74	QUIC	77 Protected Payload (K0), DCID=ea30771573235df6
289	29.125617	172.20.4.72	142.250.206.74	QUIC	83 Protected Payload (K0), DCID=ea30771573235df6
290	29.125698	172.20.4.72	142.250.206.74	QUIC	77 Protected Payload (K0), DCID=ea30771573235df6
300	29.187517	172.20.4.72	142.250.206.74	QUIC	72 Protected Payload (K0), DCID=ea30771573235df6
301	29.194787	142.250.206.74	172.20.4.72	QUIC	66 Protected Payload (K0)
313	29.212862	142.250.206.74	172.20.4.72	QUIC	67 Protected Payload (K0)
518	30.533320	172.20.4.72	8.8.8.8	DNS	70 Standard query 0xbe18 HTTPS dns.google
519	30.533802	172.20.4.72	8.8.8.8	DNS	70 Standard query 0x9475 A dns.google
529	30.552797	8.8.8.8	172.20.4.72	DNS	146 Standard query response 0xbe18 HTTPS dns.google
530	30.553434	8.8.8.8	172.20.4.72	DNS	102 Standard query response 0x9475 A dns.google A 8.
531	30.554708	172.20.4.72	8.8.8.8	QUIC	1292 Initial, DCID=f67894753a713c0a, PKN: 1, CRYPTO
532	30.554807	172.20.4.72	8.8.8.8	QUIC	1292 Initial, DCID=f67894753a713c0a, PKN: 2, PADDING,
610	30.627549	172.20.4.72	104.18.11.207	QUIC	1292 Initial, DCID=333b8ed653585a6f, PKN: 1, PING, PA
611	30.627661	172.20.4.72	104.18.11.207	QUIC	1292 Initial, DCID=333b8ed653585a6f, PKN: 2, CRYPTO,
614	30.628966	172.20.4.72	142.251.221.106	QUIC	1292 Initial, DCID=2b79d179ec199e95, PKN: 1, CRYPTO,
615	30.629015	172.20.4.72	142.251.221.106	QUIC	1292 Initial, DCID=2b79d179ec199e95, PKN: 2, CRYPTO,
635	30.655028	142.251.221.106	172.20.4.72	QUIC	82 Initial, SCID=eb79d179ec199e95, PKN: 1, ACK
636	30.655028	142.251.221.106	172.20.4.72	QUIC	1292 Initial, SCID=eb79d179ec199e95, PKN: 2, ACK, PAD

4. ICMP Packets

No.	Time	Source	Destination	Protocol	Length Info
→	354 17.664227	172.20.4.72	8.8.8.8	ICMP	74 Echo (ping) request id=0x0001, seq=13/3328, ttl=128 (reply in 355)
←	355 17.690251	8.8.8.8	172.20.4.72	ICMP	74 Echo (ping) reply id=0x0001, seq=13/3328, ttl=118 (request in 354)
	365 18.693840	172.20.4.72	8.8.8.8	ICMP	74 Echo (ping) request id=0x0001, seq=14/3584, ttl=128 (reply in 367)
	367 18.749762	8.8.8.8	172.20.4.72	ICMP	74 Echo (ping) reply id=0x0001, seq=14/3584, ttl=118 (request in 365)
	370 19.701415	172.20.4.72	8.8.8.8	ICMP	74 Echo (ping) request id=0x0001, seq=15/3840, ttl=128 (reply in 372)
	372 19.749081	8.8.8.8	172.20.4.72	ICMP	74 Echo (ping) reply id=0x0001, seq=15/3840, ttl=118 (request in 370)
	375 20.712659	172.20.4.72	8.8.8.8	ICMP	74 Echo (ping) request id=0x0001, seq=16/4096, ttl=128 (reply in 376)
	376 20.750195	8.8.8.8	172.20.4.72	ICMP	74 Echo (ping) reply id=0x0001, seq=16/4096, ttl=118 (request in 375)

```
▶ Frame 354: Packet, 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF_{3
└─ Ethernet II, Src: AzureWaveTec_50:52:92 (28:d0:43:50:52:92), Dst: Sophos_d1:3f:f0 (7c:5a:1c:d1:3f:f0)
    └─ Destination: Sophos_d1:3f:f0 (7c:5a:1c:d1:3f:f0)
    └─ Source: AzureWaveTec_50:52:92 (28:d0:43:50:52:92)
        Type: IPv4 (0x0800)
        [Stream index: 3]
    └─ Internet Protocol Version 4, Src: 172.20.4.72, Dst: 8.8.8.8
        └─ Internet Control Message Protocol
            Type: Echo (ping) request (8)
            Code: 0
            Checksum: 0x4d4e [correct]
            [Checksum Status: Good]
            Identifier (BE): 1 (0x0001)
            Identifier (LE): 256 (0x0100)
            Sequence Number (BE): 13 (0x000d)
            Sequence Number (LE): 3328 (0x0d00)
            [Response frame: 355]
        └─ Data (32 bytes)
```

```
▶ Frame 355: Packet, 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF_{3
└─ Ethernet II, Src: Sophos_d1:3f:f0 (7c:5a:1c:d1:3f:f0), Dst: AzureWaveTec_50:52:92 (28:d0:43:50:52:92)
    └─ Destination: AzureWaveTec_50:52:92 (28:d0:43:50:52:92)
    └─ Source: Sophos_d1:3f:f0 (7c:5a:1c:d1:3f:f0)
        Type: IPv4 (0x0800)
        [Stream index: 3]
    └─ Internet Protocol Version 4, Src: 8.8.8.8, Dst: 172.20.4.72
        └─ Internet Control Message Protocol
            Type: Echo (ping) reply (0)
            Code: 0
            Checksum: 0x554e [correct]
            [Checksum Status: Good]
            Identifier (BE): 1 (0x0001)
            Identifier (LE): 256 (0x0100)
            Sequence Number (BE): 13 (0x000d)
            Sequence Number (LE): 3328 (0x0d00)
            [Request frame: 354]
            [Response time: 26.024 ms]
        └─ Data (32 bytes)
```

5. ARP Frames

The figure shows a Wi-Fi traffic capture interface with the following details:

- Interface:** *Wi-Fi
- File Menu:** File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, Help
- Toolbar:** Includes icons for file operations (New, Open, Save, Print), search, and various analysis tools.
- Selected Item:** arp
- Table Headers:** No., Time, Source, Destination, Protocol, Length Info
- Data Rows:** The table lists 416 rows of ARP traffic. Key entries include:
 - Row 1: 0.000000, 32:83:d3:1a:20:ce, Broadcast, ARP, 60 ARP Announcement for 172.20.9.23
 - Row 291: 12.454192, 1e:da:0e:9b:8f:81, Broadcast, ARP, 60 Gratuitous ARP for 172.20.4.135 (Request)
 - Row 322: 14.194878, 0:a:ce:e2:81:b4:1f, Broadcast, ARP, 60 Gratuitous ARP for 172.20.5.28 (Reply)
 - Row 369: 18.803412, c2:0e:43:8f:a8:d3, Broadcast, ARP, 60 Gratuitous ARP for 172.20.2.65 (Reply)
 - Row 371: 19.724593, 36:70:bc:9a:e6:d4, Broadcast, ARP, 60 ARP Announcement for 172.20.2.205
 - Row 377: 21.158001, 36:70:bc:9a:e6:d4, Broadcast, ARP, 60 ARP Announcement for 172.20.2.205
 - Row 412: 28.531207, AzureWaveTec_86:6a:..., Broadcast, ARP, 60 ARP Announcement for 172.20.9.60
 - Row 413: 29.658339, ba:0:a:ac:5b:71:24, Broadcast, ARP, 60 Gratuitous ARP for 172.20.6.234 (Request)
 - Row 414: 29.658339, ba:0:a:ac:5b:71:24, Broadcast, ARP, 60 Gratuitous ARP for 172.20.6.234 (Request)
 - Row 415: 29.658339, ba:0:a:ac:5b:71:24, Broadcast, ARP, 60 Gratuitous ARP for 172.20.6.234 (Request)
 - Row 416: 30.170150, 32:83:d3:1a:20:ce, Broadcast, ARP, 60 ARP Announcement for 172.20.9.23

```
▶ Frame 291: Packet, 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF_{3
  ▶ Ethernet II, Src: 1e:da:0e:9b:8f:81 (1e:da:0e:9b:8f:81), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
    ► Destination: Broadcast (ff:ff:ff:ff:ff:ff)
    ► Source: 1e:da:0e:9b:8f:81 (1e:da:0e:9b:8f:81)
    Type: ARP (0x0806)
      [Stream index: 5]
      Padding: 0000000000000000000000000000000000000000000000000000000000000000
  ▶ Address Resolution Protocol (request/gratuitous ARP)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
      [Is gratuitous: True]
    Sender MAC address: 1e:da:0e:9b:8f:81 (1e:da:0e:9b:8f:81)
    Sender IP address: 172.20.4.135
    Target MAC address: Broadcast (ff:ff:ff:ff:ff:ff)
    Target IP address: 172.20.4.135
```