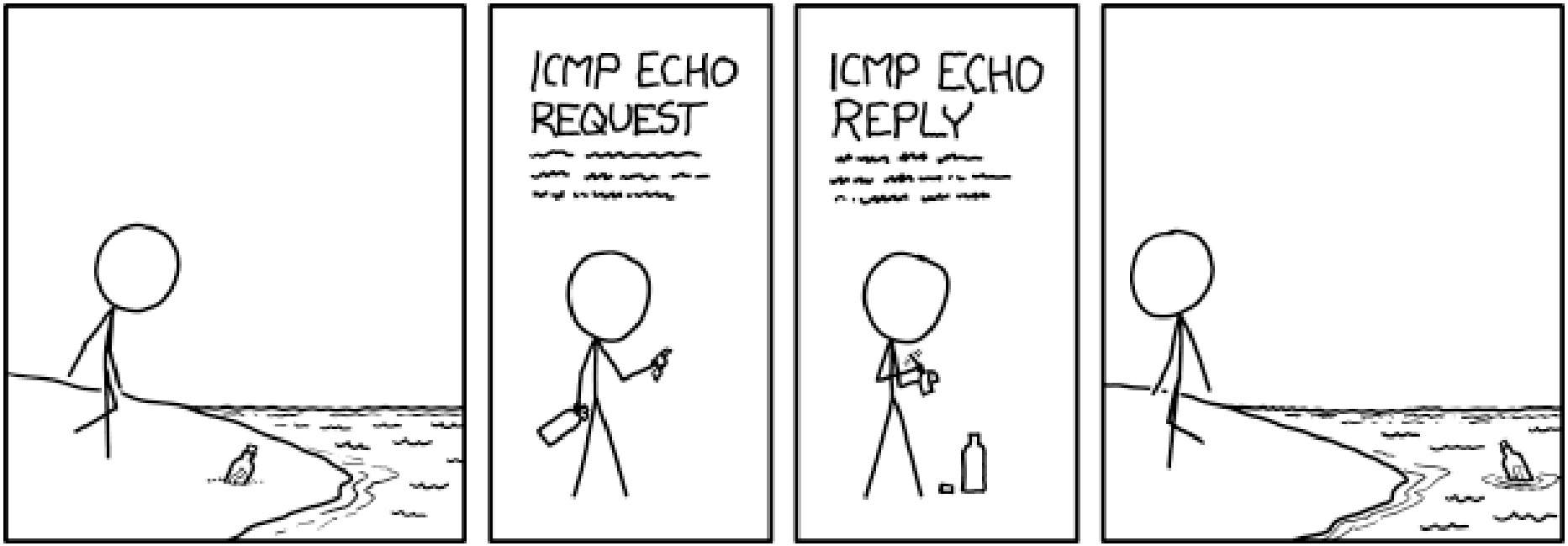


Source: https://xkcd.com/3150/

CS 60:
Computer Networks

Network layer: Data plane

Based on Computer Networking: A Top-Down Approach by Kurose and Ross unless otherwise noted

Review from last class

Layer 4: TCP is connection oriented

- Three-way handshake to establish connection – SYN, SYN ACK, ACK
- Sender can't be sure receiver got the Segment without a reply from the receiver
- Uses checksum, timer, and sequence numbers to handle
  - Bit flips
  - Dropped packets
  - Packets arriving out of order
- Flow control: receiver uses a receive window field in the header to keep sender and receiver in sync (e.g., make sure sender does not send too fast)

TCP streams bytes as a stream of data. Bytes from multiple segments that arrive before *recv* is called are concatenated. Use a length or delimiter between messages

We did not discuss it, but there is also a mechanism to slow down if the network is busy (e.g., the receiver can handle faster speed, but the network is slow)

TCP header is usually 20 bytes (although there can be optional info, so 20 is not guaranteed, check offset field)

- UDP header is always 8 bytes

Today's focus is on Layer 3: called the Network or IP Layer that routes packets

Conceptual network layers

7) Application	Interacts with application programs to send ***messages*** Applications assigned a port, multiple instances can run (many browser pages) Examples: HTTP, SSH, FTP, SMTP, DNS
4) Transport	Moves ***segments or datagrams*** May provide error control, flow control, application addressing (ports) Examples: TCP (connection-oriented), UDP (connectionless) TCP provides sequencing, dropped packet resend, traffic congestion routing
3) Network (IP)	Moves ***packets*** between local area networks (routing) Each computer on the Internet identified by an IP address (IP v4 or v6) **Also called Layer 3 or IP layer (ICMP Ping is here)**
2) Link (MAC)	Moves ***frames*** within a local area network (switching) Each computer identified by a MAC address on its Network Interface Card (NIC) Also called Layer 2, MAC layer, Data Link layer, or Ethernet layer
1) Physical	How data is physically transmitted • Transmitter converts logical 1 and 0 ***bits*** to electrical/light pulses or phase/amplitude of radio frequency (RF) and sends down wire or over air • Receiver converts electrical/light or RF back to logical 1 and 0 bits

https://www.ibm.com/think/topics/osi-model

Agenda



1. Review of network layers
2. A router has two jobs:
   - Forwarding **Today**
   - Routing
3. Network (IP) Layer header
4. Forwarding
5. Exercise
6. Supplemental material

Review: network layers

Network model	Critical information
7) Application	Application message to be sent across network
4) Transport	
3) Network (IP)	
2) Link (MAC)	
1) Physical	



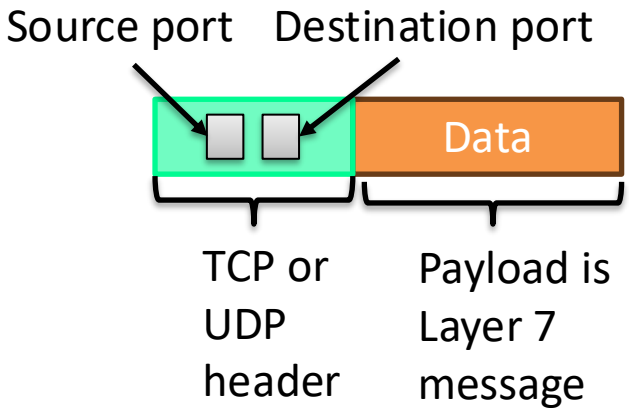
Layer 7 message

Layer 7 messages normally follow an RFC or web service description that tell:

- **How to format messages**
- **What to expect in reply**

Review: network layers

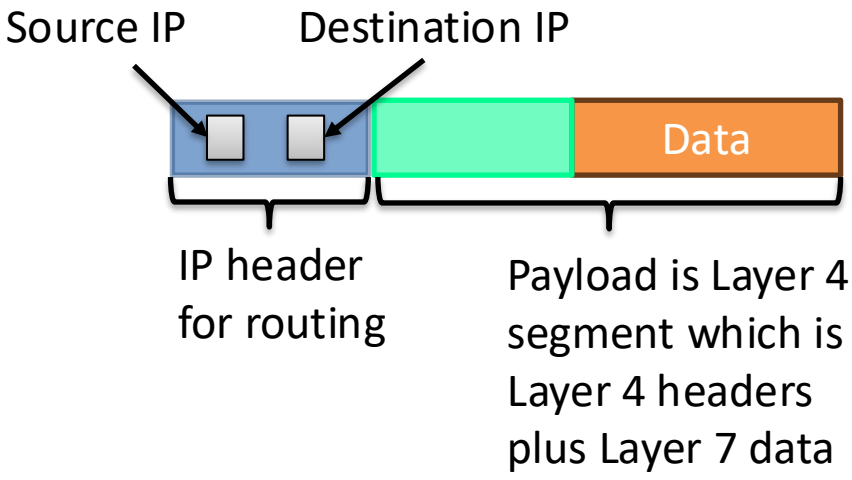
Network model	Critical information
7) Application	Application message to be sent across network
4) Transport	TCP or UDP src and dst port
3) Network (IP)	
2) Link (MAC)	
1) Physical	



Layer 4 connects *processes*
Ports used for mux/demux

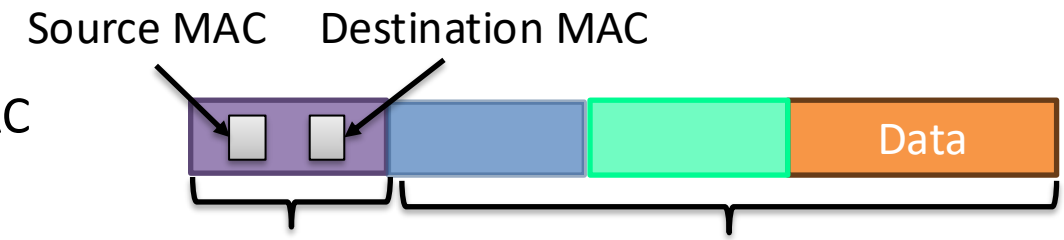
Review: network layers

Network model	Critical information
7) Application	Application message to be sent across network
4) Transport	TCP or UDP src and dst port
3) Network (IP)	src and dst IP address **Routing!**
2) Link (MAC)	**Packets routed across LANs using IP addresses**
1) Physical	**Layer 3 connects *hosts***



Review: network layers

Network model	Critical information
7) Application	Application message to be sent across network
4) Transport	TCP or UDP src and dst port
3) Network (IP)	src and dst IP address Routing!
2) Link (MAC)	src and dst MAC address **Switching!**
1) Physical	**Frames transmitted *within* a LAN based on MAC address**

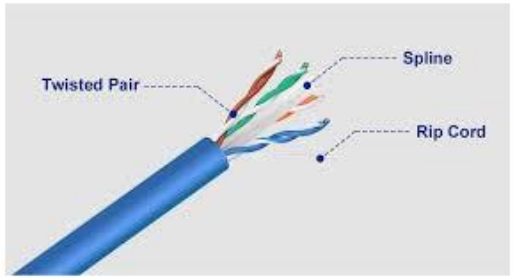


MAC header for switching

Payload is Layer 3 packet which is Layer 3 headers plus Layer 4 headers plus Layer 7 data

Review: network layers

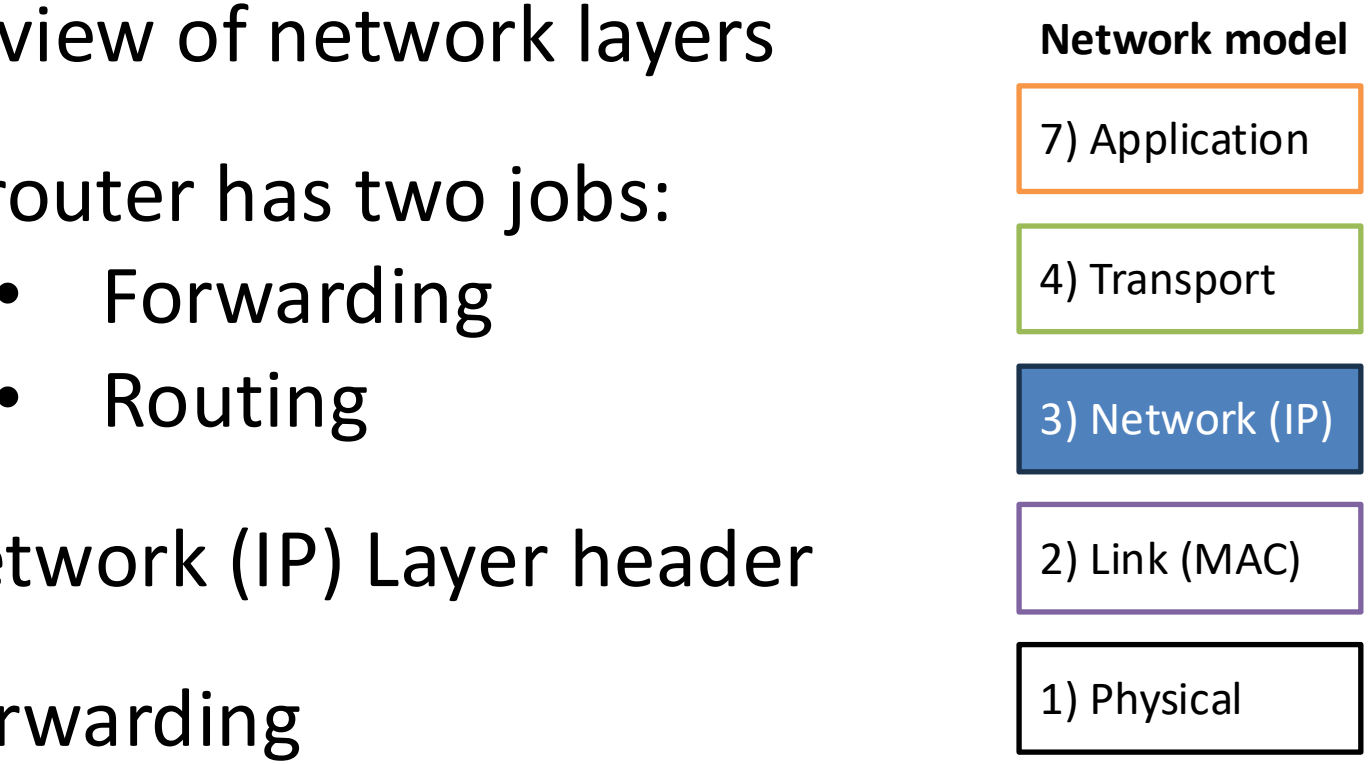
Network model	Critical information
7) Application	Application message to be sent across network
4) Transport	TCP or UDP src and dst port
3) Network (IP)	src and dst IP address Routing!
2) Link (MAC)	src and dst MAC address Switching!
1) Physical	**Sends frames to another host over cable or RF**





Agenda

1. Review of network layers
2. A router has two jobs:
   - Forwarding
   - Routing
3. Network (IP) Layer header
4. Forwarding
5. Exercise
6. Supplemental material

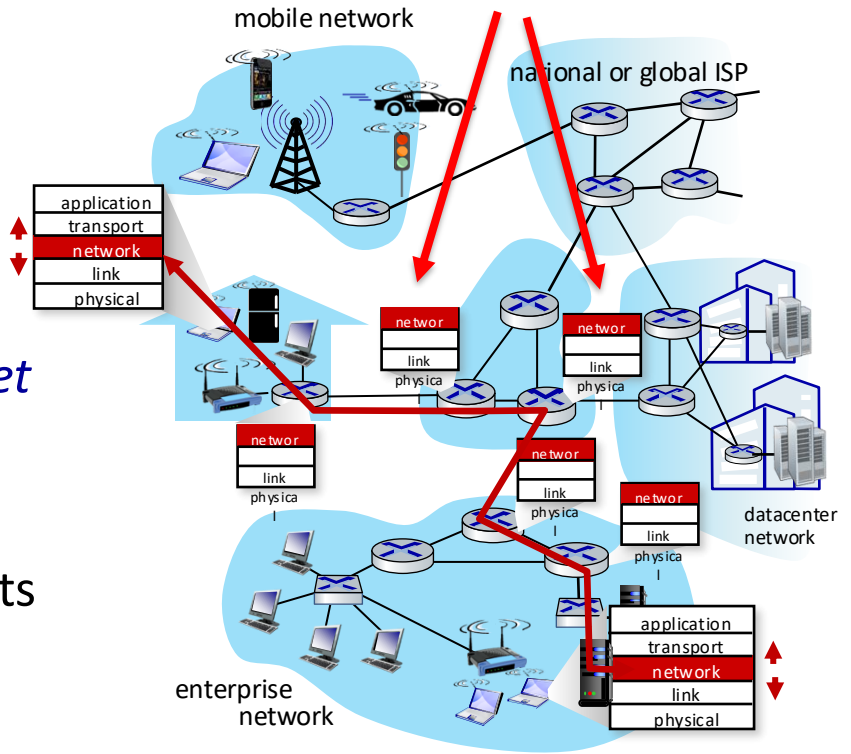


Network layer connects hosts

Network Layer

- Transport segments or datagrams from sending to receiving host
  - Sender: encapsulates segments, passes packet down to Link Layer
  - Receiver: delivers segments up to Transport Layer
- Network layer protocols in *every Internet device*: hosts and routers
- Routers:
  - Examines header fields in all IP packets passing through it
  - Moves packet from input ports to output ports to transfer packet along end-to-end path
  - Send traffic between LANs

Note: routers shown with truncated network stacks because they do not work at Layer 4 or above



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Two key network-layer functions: forwarding and routing

Network Layer functions:

1. *Forwarding:* *move packets from a router's input link to appropriate router output link* **Data plane (today)**
2. *Routing:* determine route taken by packets from source to destination

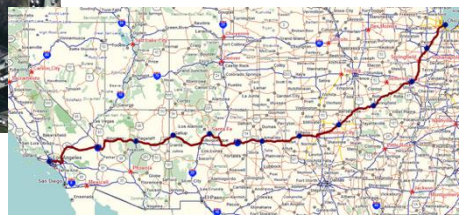
   Control plane (next class)

Analogy: taking a trip

- *Forwarding:* process of getting through single interchange
- *Routing:* process of planning trip from source to destination



forwarding

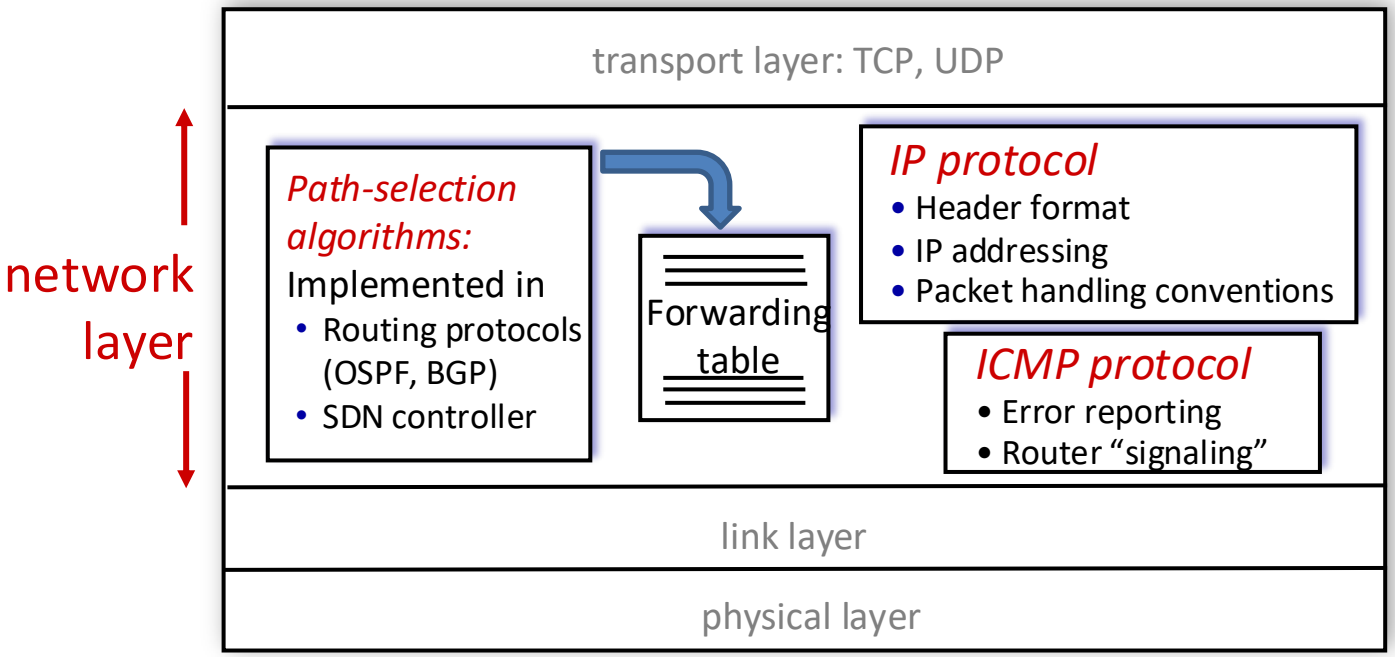


routing

Network Layer: Internet

Router network layer functions:

IP headers give source and destination IP addresses



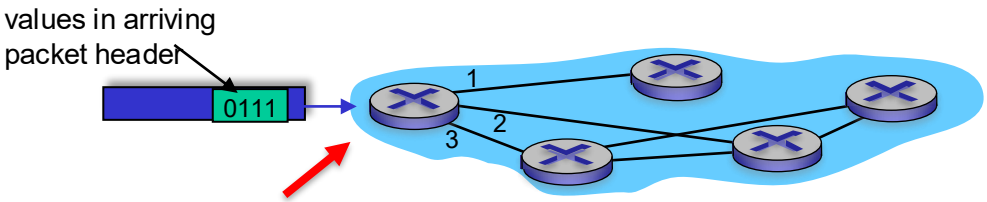
Control plane determines path from sender to receiver (next class)

Writes forwarding table that takes packets received on one port and sends them out another port (quickly!)

Why is ICMP considered Layer 3?
Deals with IP addresses (so Layer 3)
No port numbers (so not Layer 4)
Does not carry higher level message (so not Layer 7)

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Forwarding process



This router is connected to three other routers via three physical ports

Packet arrives

Goal: forward the packet to the next router or host (quickly!)

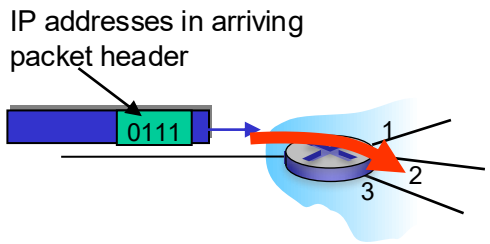
1. Router strips off Layer 2 header
2. If packet MAC matches router MAC, pass to this router's Layer 3, else drop
3. Layer 3 looks at destination IP address in header
4. Determine IP address of the host that is the next hop
5. Determine physical port connected to next hop
6. Re-write Layer 2 header with next hop host's MAC (using ARP to match MAC address with IP address)
7. Send packet to next hop host via physical port

Network layer: data plane and control plane

Data plane:

- *Local*, per-router function
- Determines how packet arriving on router input port is forwarded to router output port

Next hop may not be the final destination



Control plane

- *Network-wide* logic
- Determines how packet is routed among routers along end-end path from source host to destination host
- Writes forwarding table
  - Two control-plane approaches:
    1. *Traditional routing algorithms:* implemented in each router
    2. *Software-defined networking (SDN)*: implemented in (remote) servers, rules sent to each router

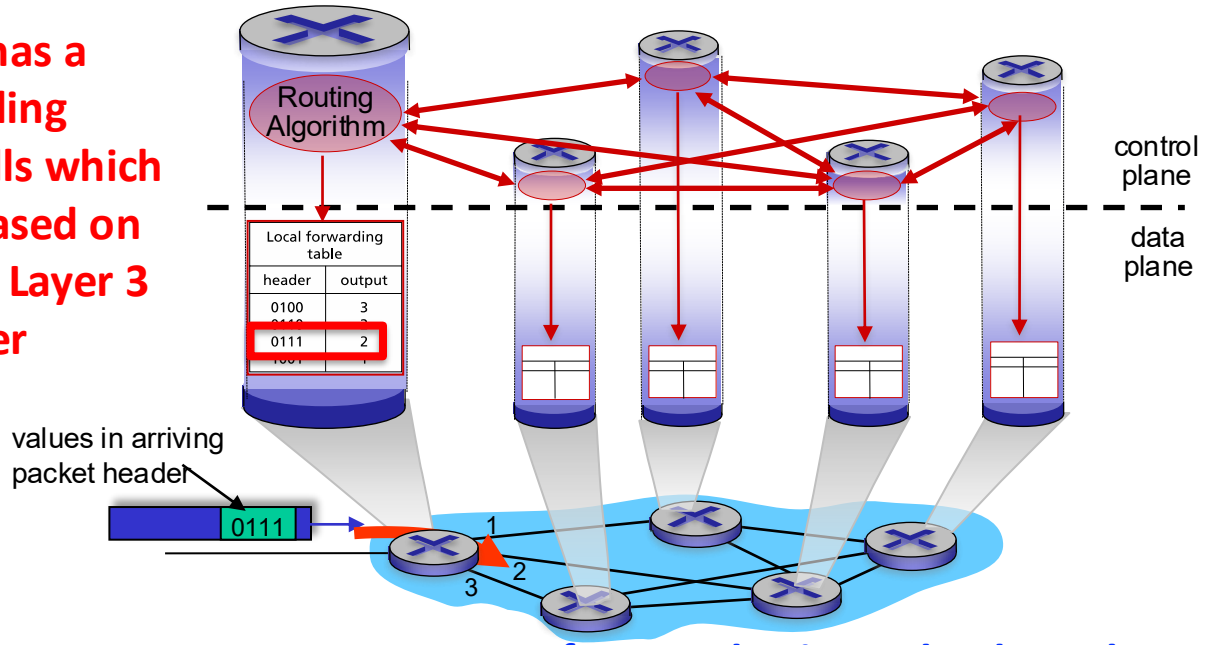
Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

1) Traditional routing has per-router control plane

Traditional routing

Individual routing algorithm components *in each and every router* interact in the control plane

Each router has a local forwarding table that tells which link to use based on IP address in Layer 3 packet header



Routers communicate with each other to set the forwarding table

Each router maintains its own forwarding table

Our focus today is on the data plane (how to forward from input port to output port)

Next class will focus on control plane (which output port has the next hop)

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

2) Software-Defined Networking (SDN) has a separate control plane

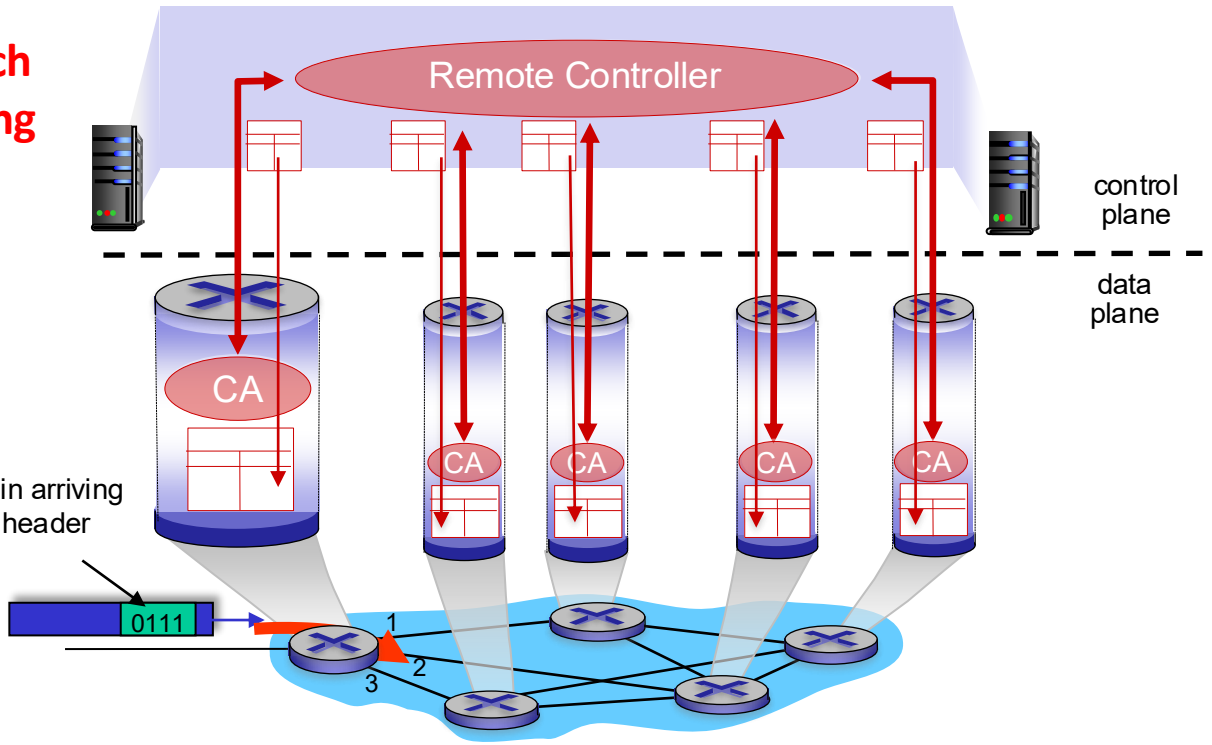
Software Defined Network (SDN)

Remote controller computes routes, installs forwarding tables in routers

SDN: a central controller gives each router its forwarding table

Each router has a Control Agent (CA) that talks with the SDN Controller

Data plane in SDN same as traditional



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

The network layer is a “best effort” model, no guarantees of performance

Network Architecture	Service Model	Quality of Service (QoS) Guarantees ?			
		Bandwidth	Loss	Order	Timing
Internet	best effort	none	no	no	no

Network Layer “best effort” service model

No guarantees of:

i. Successful delivery to destination
ii. Timing or order of delivery
iii. Bandwidth available to end-end flow

TCP at Layer 4 provides guarantees of non-loss and in-order delivery, but not bandwidth or timing

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Other protocols have been proposed to provide guaranteed performance

Network Architecture	Service Model	Quality of Service (QoS) Guarantees ?			
		Bandwidth	Loss	Order	Timing
Internet	best effort	none	no	no	no
ATM	Constant Bit Rate	Constant rate	yes	yes	yes
ATM	Available Bit Rate	Guaranteed min	no	yes	no
Internet	Intserv Guaranteed (RFC 1633)	yes	yes	yes	yes
Internet	Diffserv (RFC 2475)	possible	possibly	possibly	no

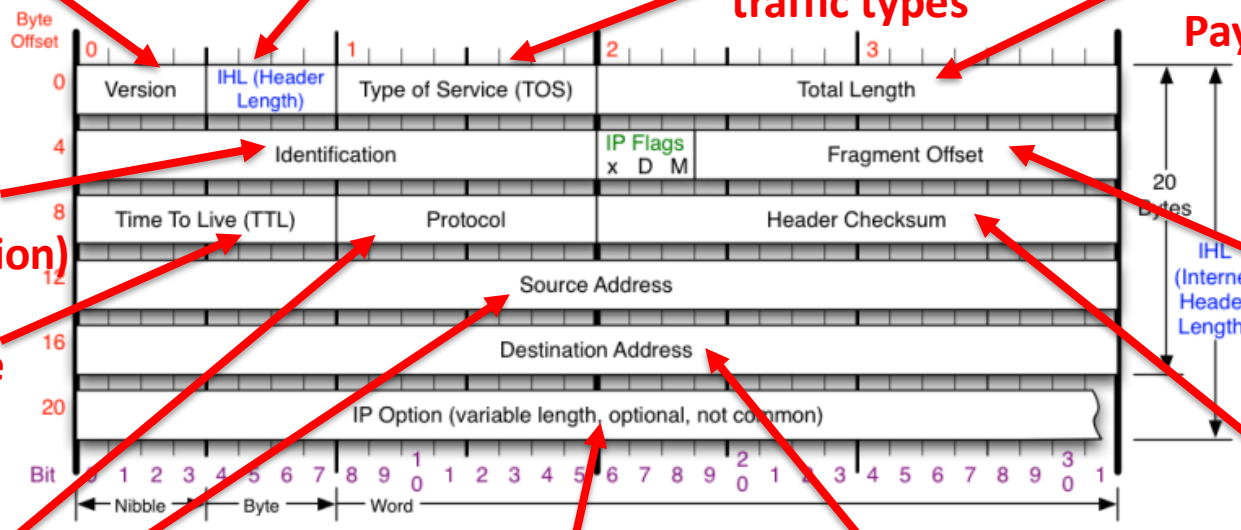
While other protocols have been proposed, the classic Internet best effort approach has been "good enough", even for demanding things like Netflix

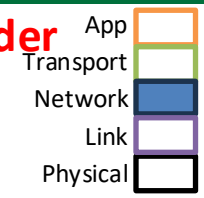
Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Agenda

1. Review of network layers
2. A router has two jobs:
    - Forwarding
    - Routing
3. Network (IP) Layer header
4. Forwarding
5. Exercise
6. Supplemental material

Layer 3 header format





Version
4 = IPv4
6 = IPv6

Header length in 32-bit words
(multiply by 4 to get bytes)

Normally 0
Can prioritize some traffic types

Length of header plus payload

Payload = UDP/TCP header + App data

Packet ID (used for fragmentation)

Fragmentation (not covering)

Time to live (recall traceroute)

Checksum (detect bit flips)
Only for header!
Removed for IPv6 (rely on Layer 2 and 4)

Higher level protocol
UDP: 17
TCP: 6
ICMP: 1
(Layer 3)

Source IP address

Destination IP address

Optional information (not commonly used)

Routers use this Layer 3 information to decide where to send packets

Remember, Layer 3 decoded by receiver before Layer 4

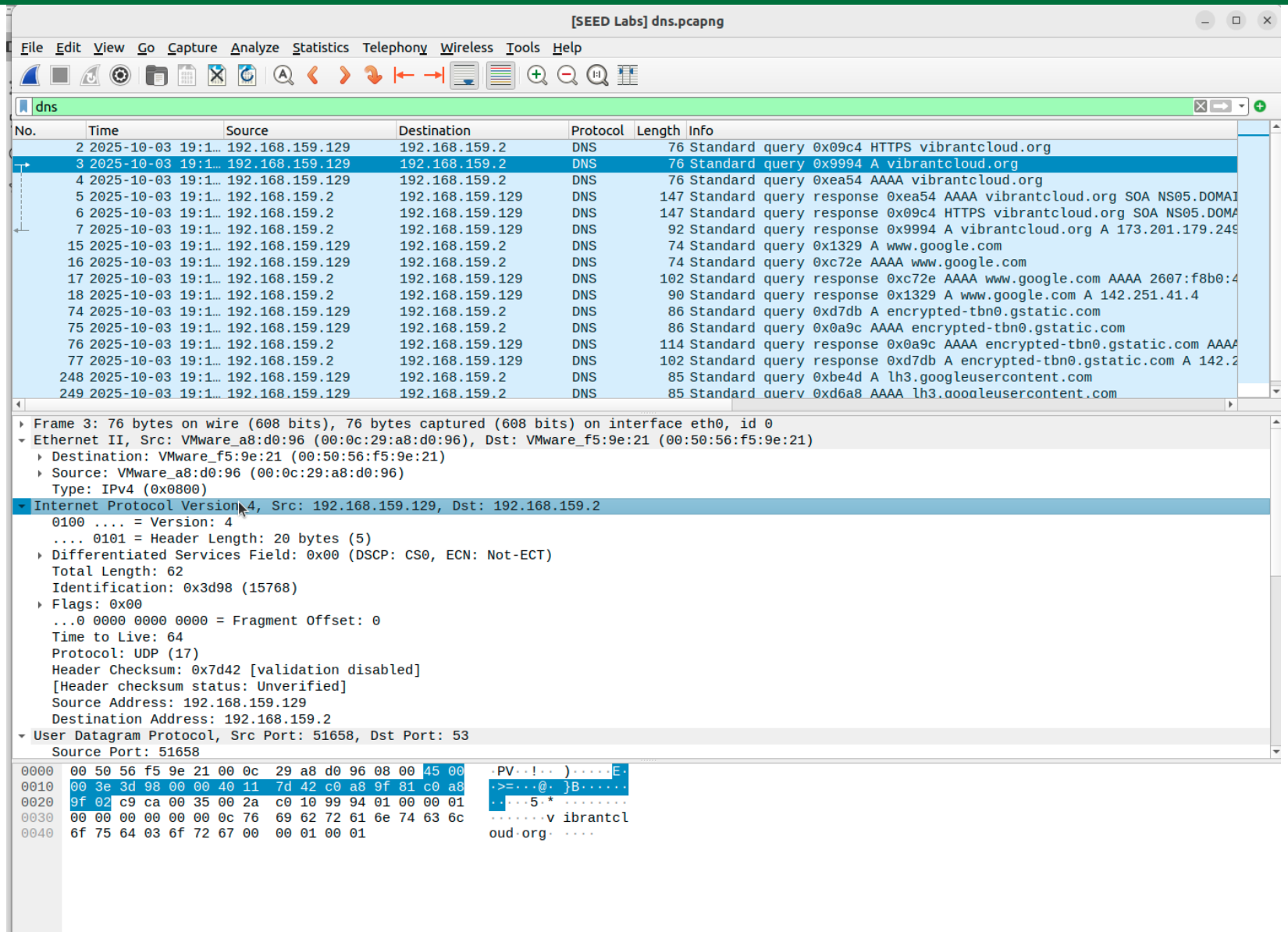
Today's topic

UDP and TCP requests

	UDP: DNS Request	TCP: HTTP Request
Application	Message format: Follow RFC 1034,1035	Message format: Follow RFC 7230-7235
Transport	UDP Port: 53	TCP Port: 80
Network	Request from DNS name server	Request from web server
Link	src and dst MAC	src and dst MAC
Physical	Bits	Bits

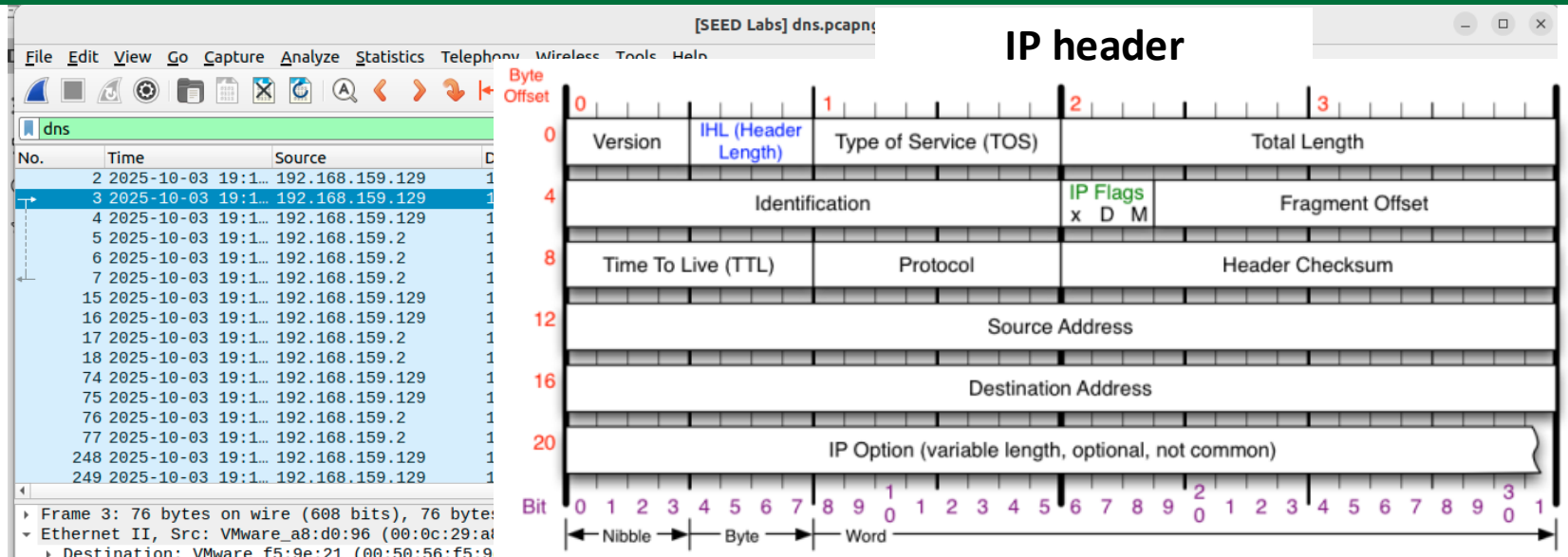
Supplemental slides walk through UDP DNS and TCP HTTP requests from Layer 7 down through Layer 2
Here we look at UDP DNS Request at Layer 3

DNS Layer 3

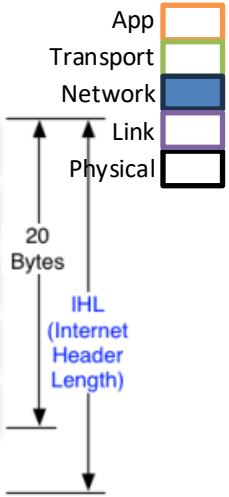


App
Transport
Network
Link
Physical

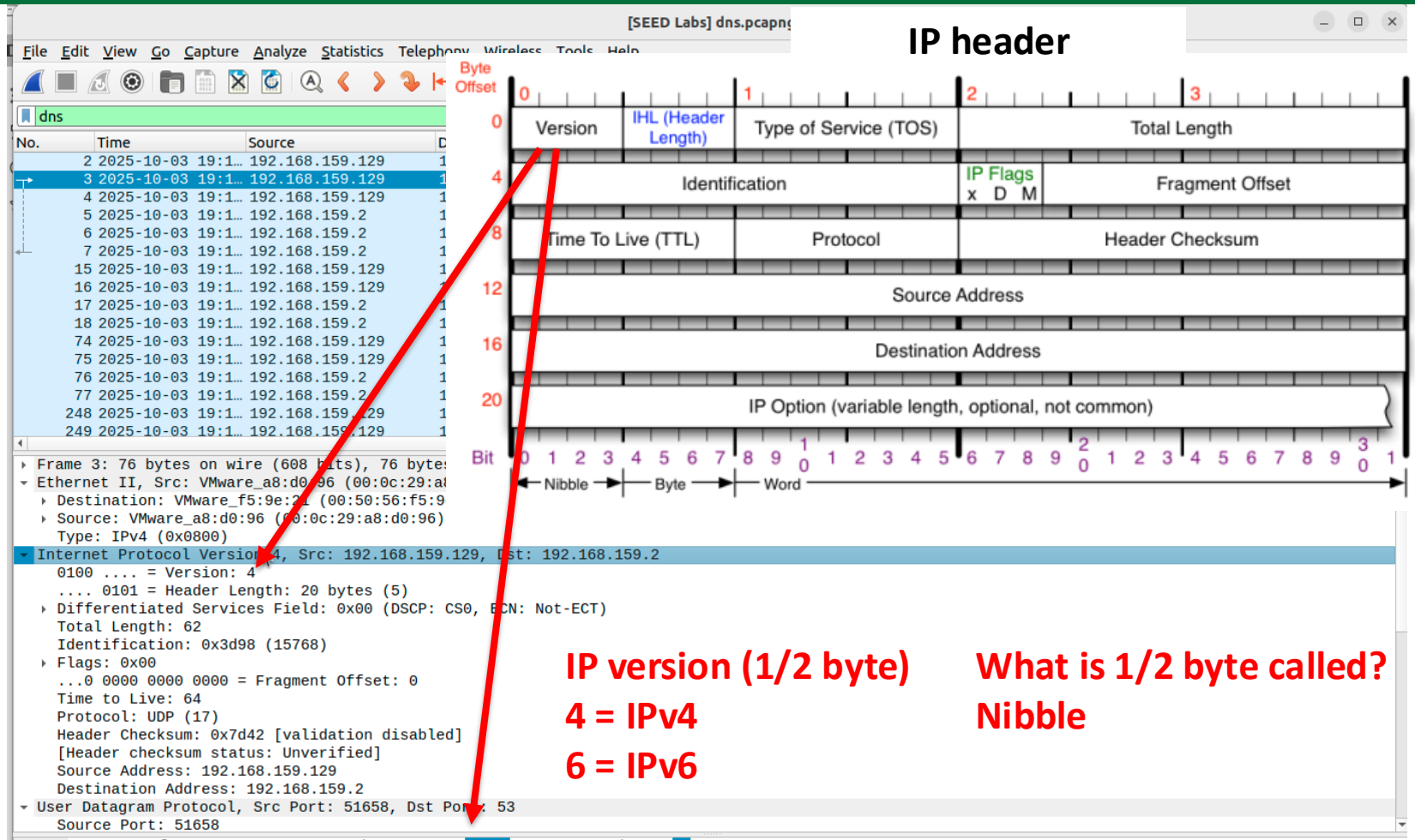
DNS Layer 3



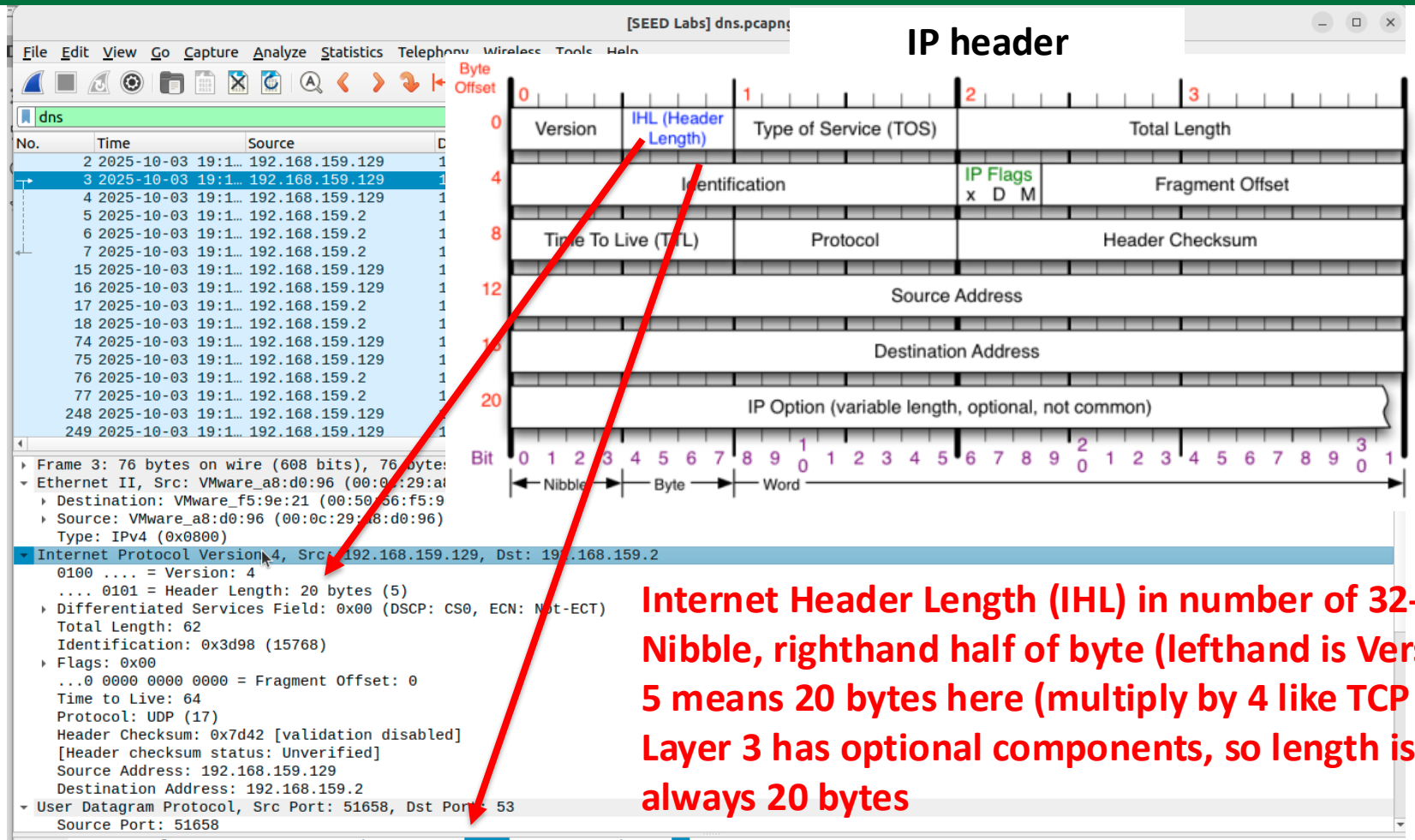
IP header



Version is IPv4

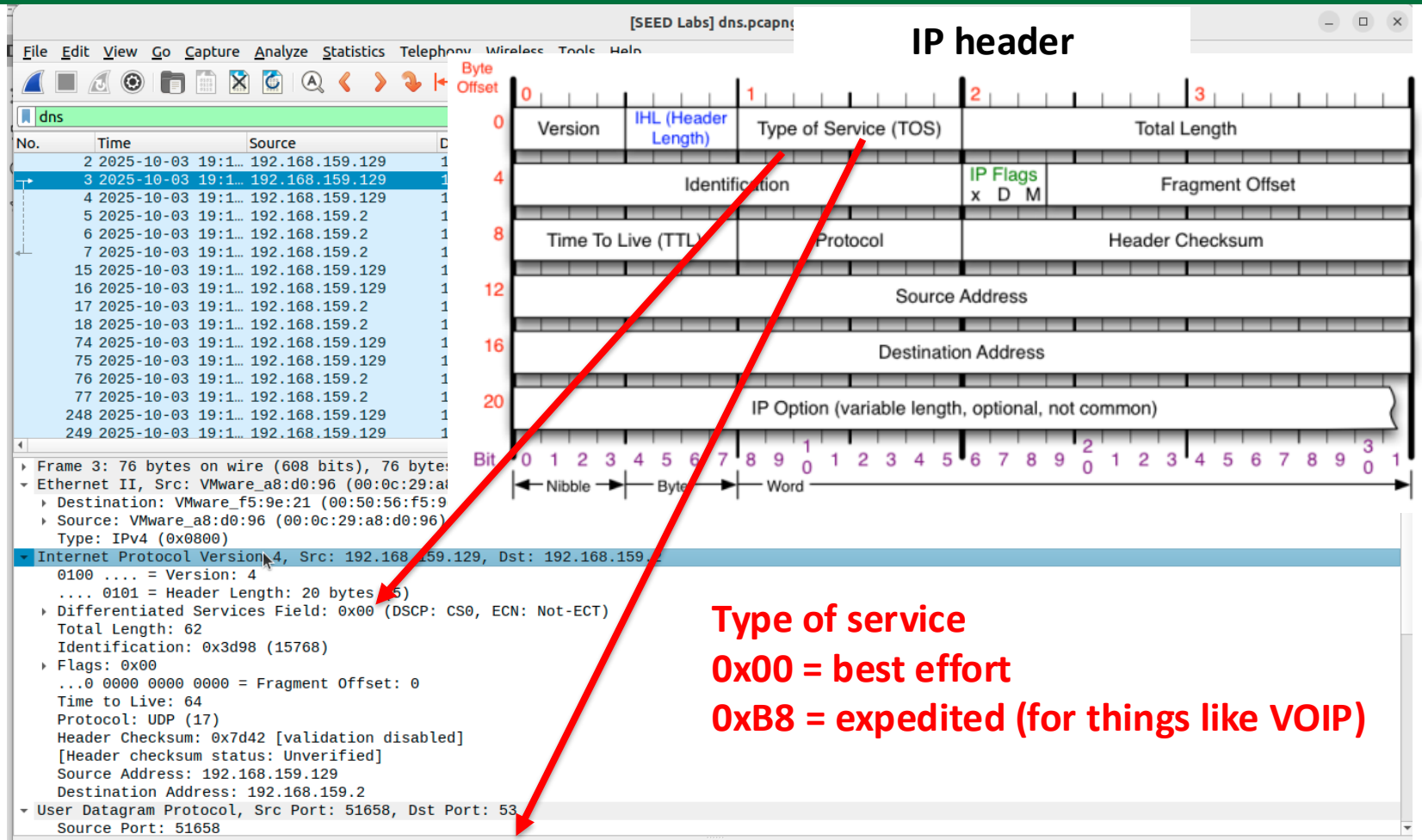


IP header length is 5 *4 = 20 bytes

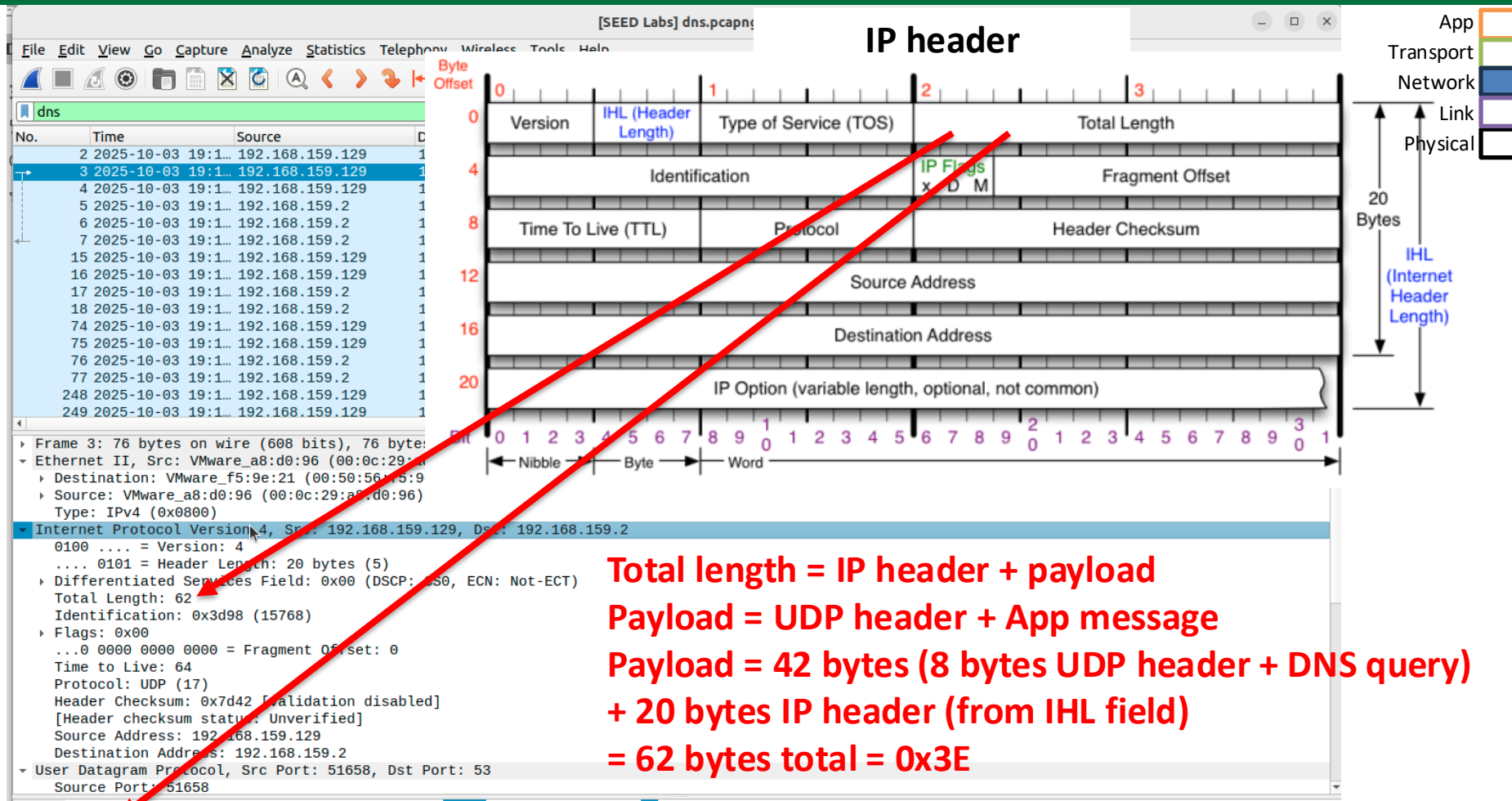


Internet Header Length (IHL) in number of 32-bit words
Nibble, righthand half of byte (lefthand is Version)
5 means 20 bytes here (multiply by 4 like TCP offset)
Layer 3 has optional components, so length is not always 20 bytes

TOS usually zero



Total length = IP header + payload



Total length = IP header + payload

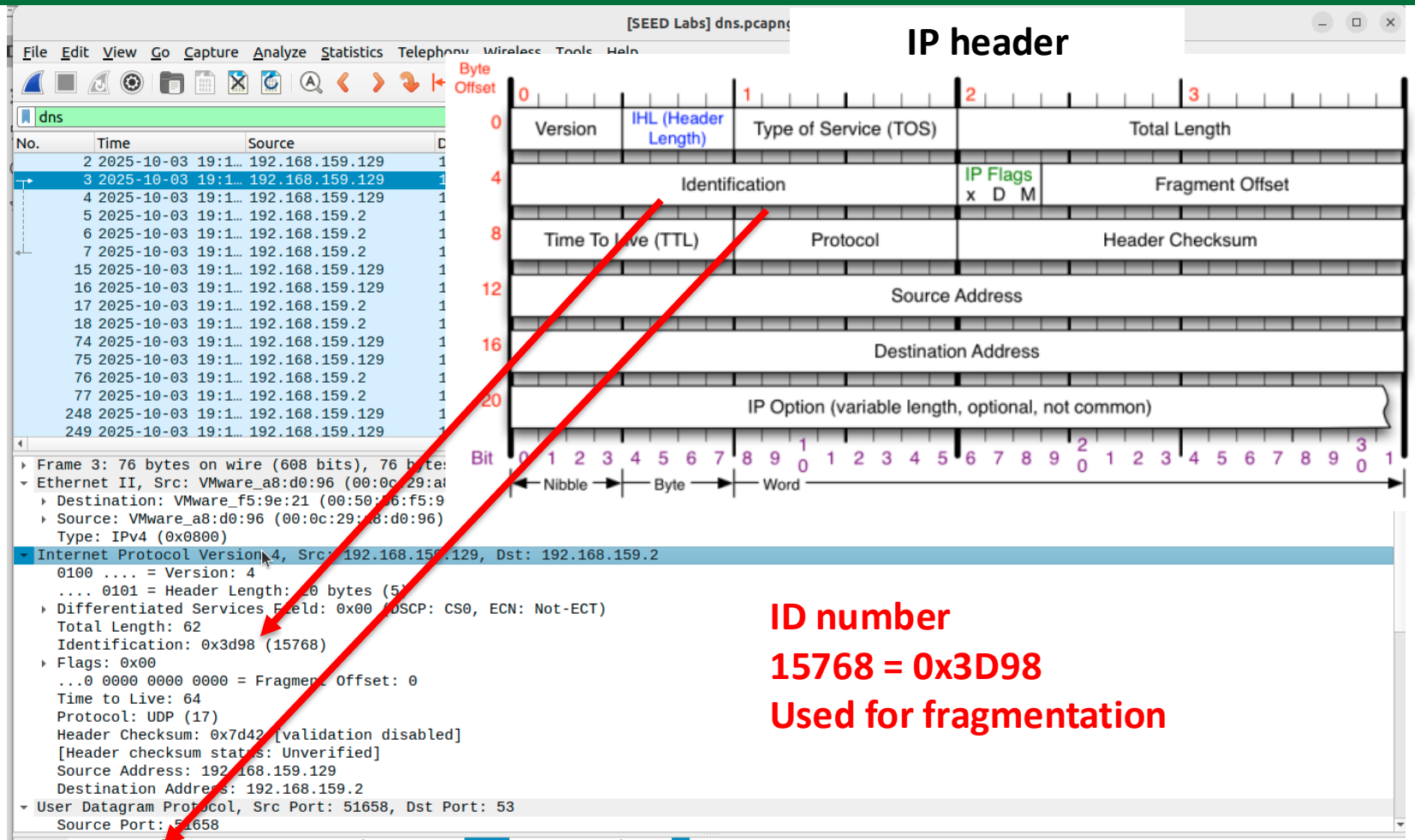
Payload = UDP header + App message

Payload = 42 bytes (8 bytes UDP header + DNS query)

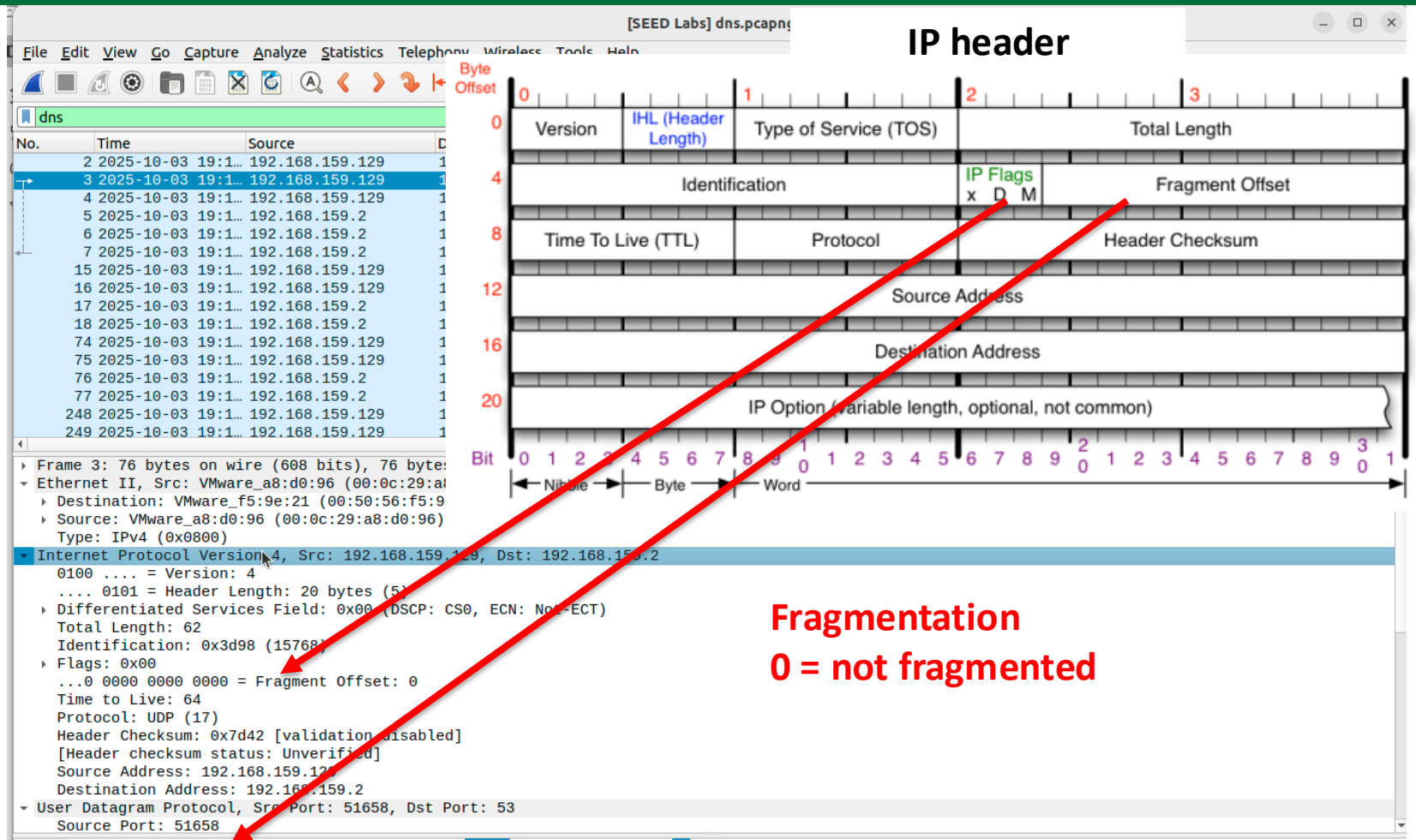
+ 20 bytes IP header (from IHL field)

= 62 bytes total = 0x3E

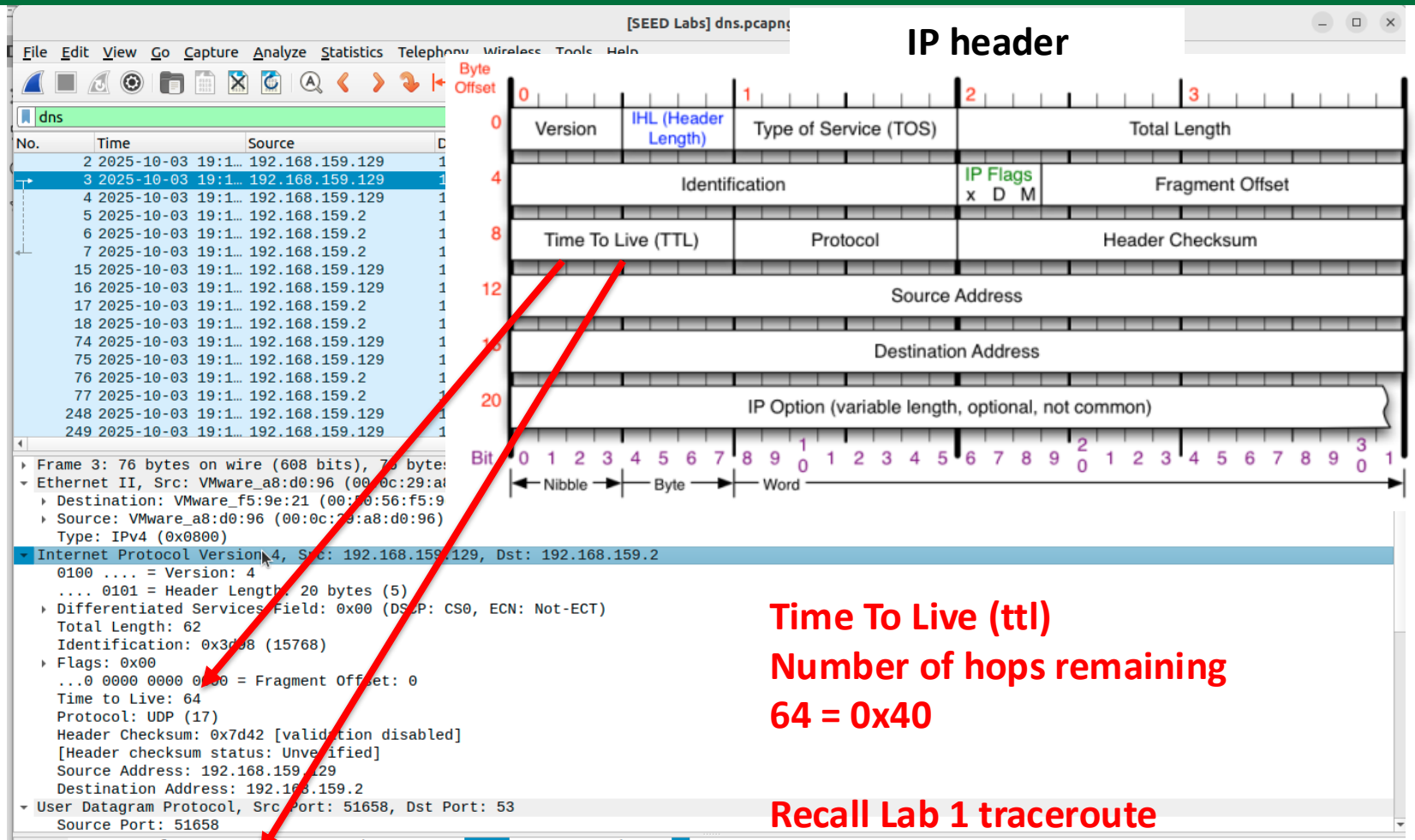
Packet has identification number for fragmentation



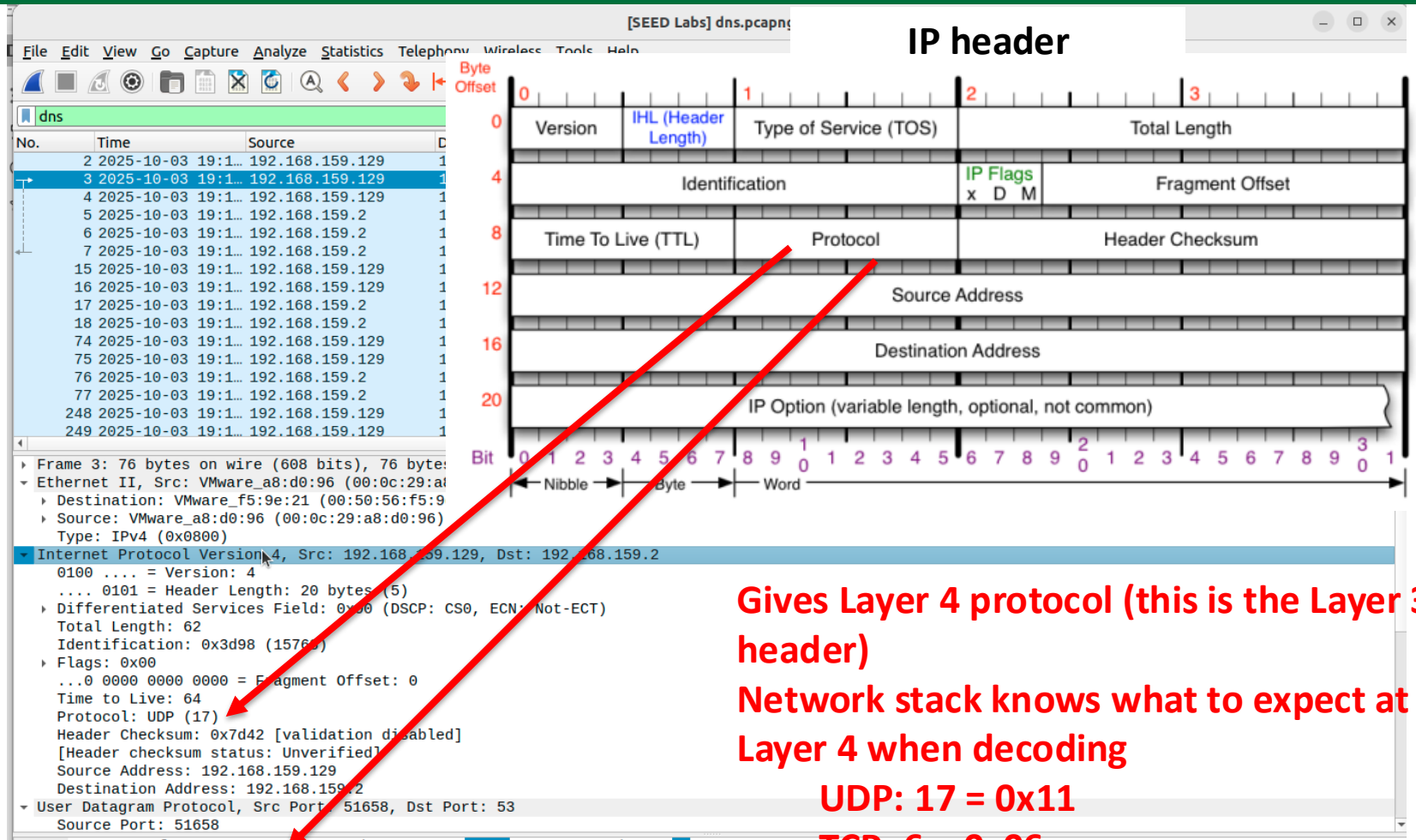
Flags and fragmentation offset



Time To Live (TTL) is 64 hops



Protocol is UDP (17 = 0x11)



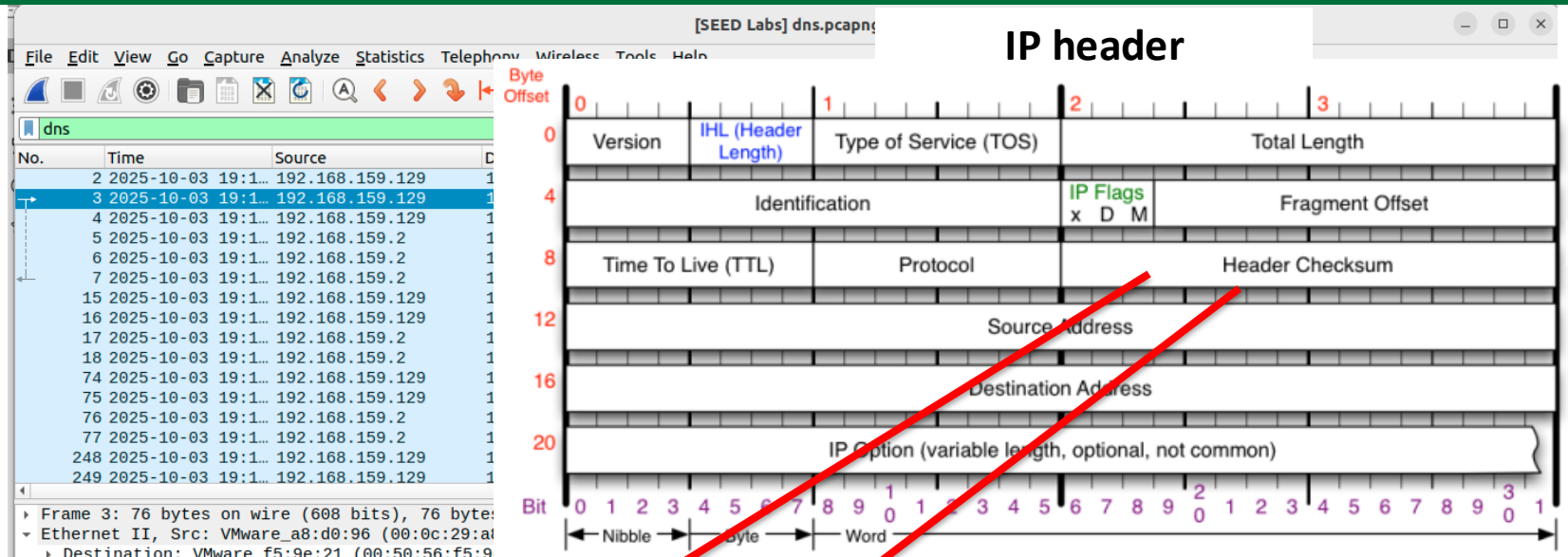
Gives Layer 4 protocol (this is the Layer 3 header)

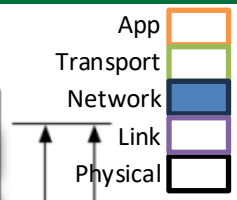
Network stack knows what to expect at Layer 4 when decoding

- **UDP: 17 = 0x11**
- **TCP: 6 = 0x06**
- **ICMP: 1 = 0x01**

We will see soon that there is a field at Layer 2 that tells if Layer 3 is IPv4 or IPv6

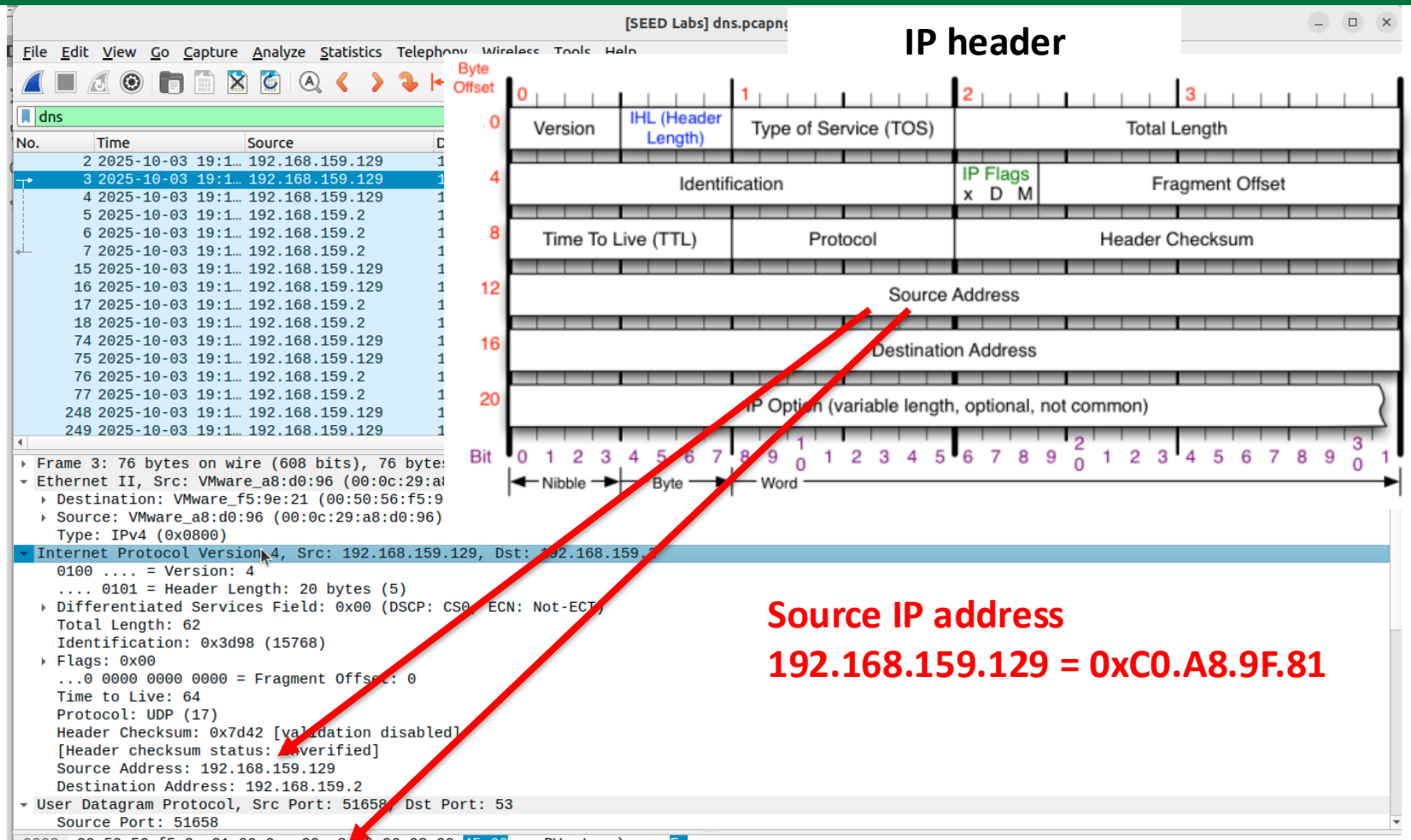
Checksum is for the header bytes only (unlike Layer 4)



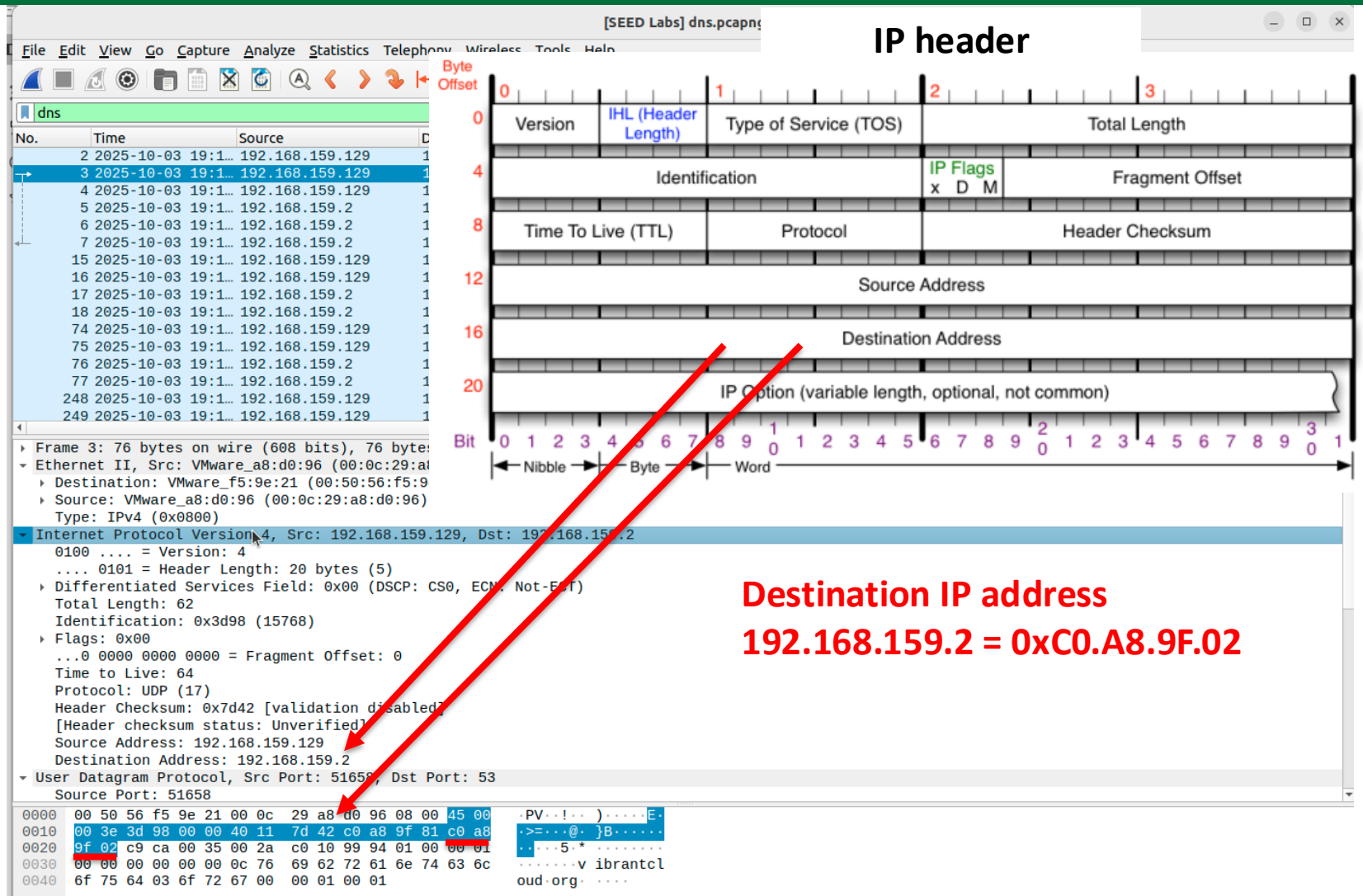


Checksum
16 bits, one's complement
Computed on header only
Not used for IPv6 (rely on Layer 2 and 4)

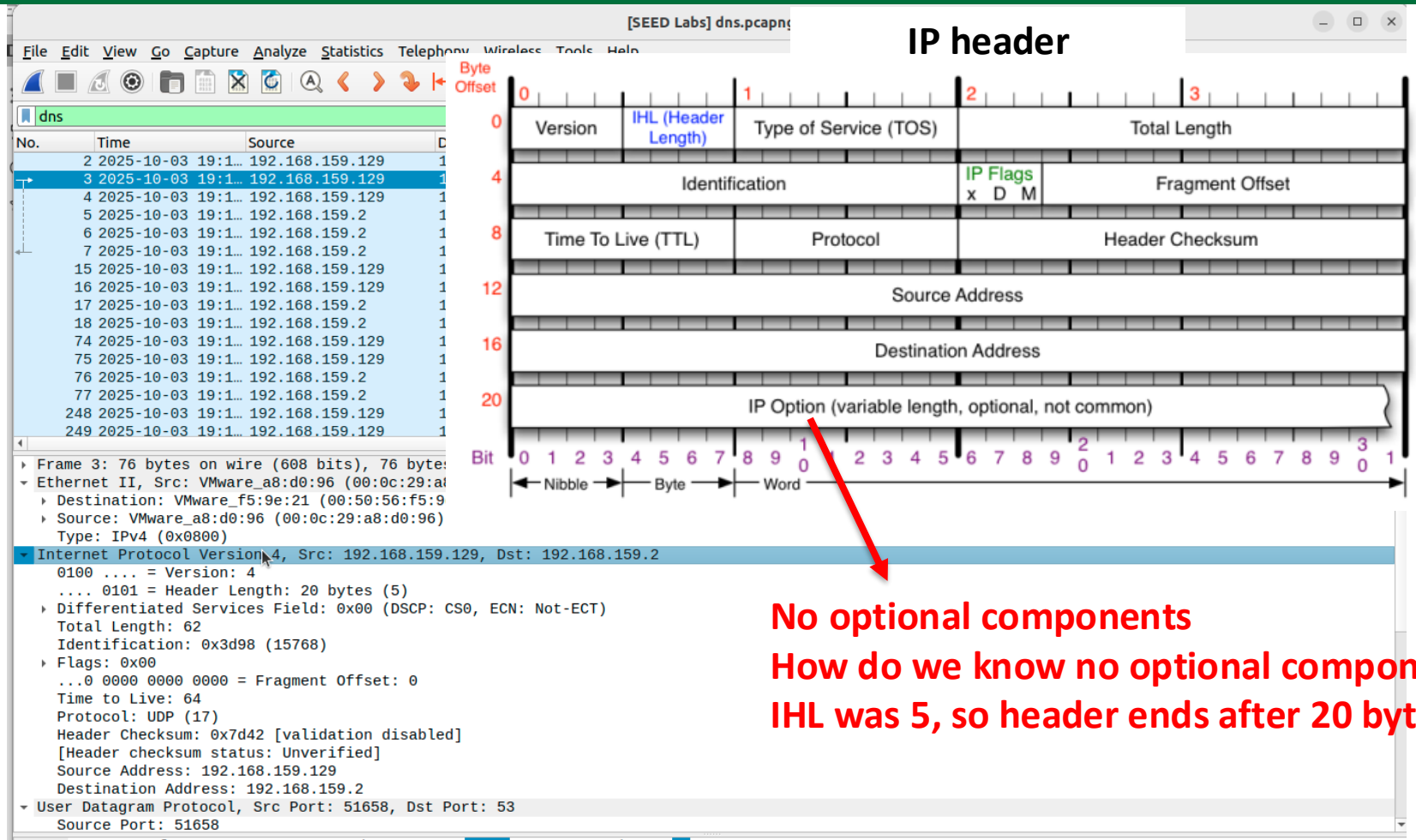
Source and destination IP address



Source and destination IP address



No optional values in this packet



No optional components

How do we know no optional components?

IHL was 5, so header ends after 20 bytes

Agenda

1. Review of network layers
2. A router has two jobs:
    - Forwarding
    - Routing
3. Network (IP) Layer header
4. Forwarding
5. Exercise
6. Supplemental material

IP addresses: how to get one?

That's actually two questions:

1. Q: How does a *host* get IP address within its network (host part of address)?
2. Q: How does a *network* get IP address for itself (network part of address)

How does *host* get IP address?

1. Hard-coded by sysadmin in config file (e.g., /etc/rc.config in UNIX)
2. DHCP: Dynamic Host Configuration Protocol: dynamically get address from a server
   - "plug-and-play"
   - Often part of a LAN's router

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

The network get an IP address from the ISP's range of addresses

Q: How does *network* get subnet part of IP address?

A: Gets allocated portion of its provider ISP's address space

ISP's block 11001000 00010111 00010000 00000000 200.23.16.0/20

32-20 = 12 bits the ISP *could* use to create up to 2^{12} = 4096 unique networks from this block

ISP can then allocate out its address space to clients:

But each network could have no hosts if it did!

Organization 0	11001000 00010111 00010000 00000000	200.23.16.0/23
Organization 1	11001000 00010111 00010010 00000000	200.23.18.0/23
Organization 2	11001000 00010111 00010100 00000000	200.23.20.0/23
…	….	….
Organization 7	11001000 00010111 00011110 00000000	200.23.30.0/23

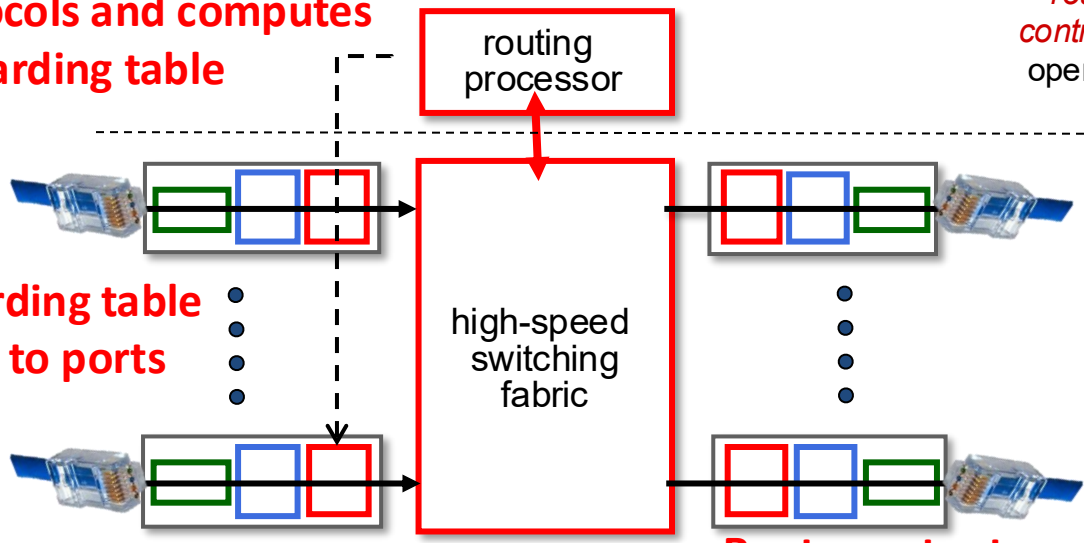
Here the ISP gives /23 subnet to each client
Uses 3 bits to uniquely identify each client's subnet
ISP can create 8 subnets with these 3 bits (2^3 = 8)
Each client can have $2^{32-23} = 2^9$ = 512 hosts (or could create subnets of smaller size)

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Router architecture overview

High-level view of generic router architecture:

Processor executes routing protocols and computes forwarding table



routing, management control plane (software) operates in millisecond time frame

forwarding data plane (hardware) operates in nanosecond timeframe

Forwarding table copied to ports

Router input ports

1. **Physical connection**
2. **Link layer functions**
3. **Lookup function to find output port**

Switching fabric connects input and output ports

Router output ports

Routing processor handles links going up and down

Link failure detection can operate at second scale

Implemented in software

Input, output ports and switching fabric implemented in hardware for speed

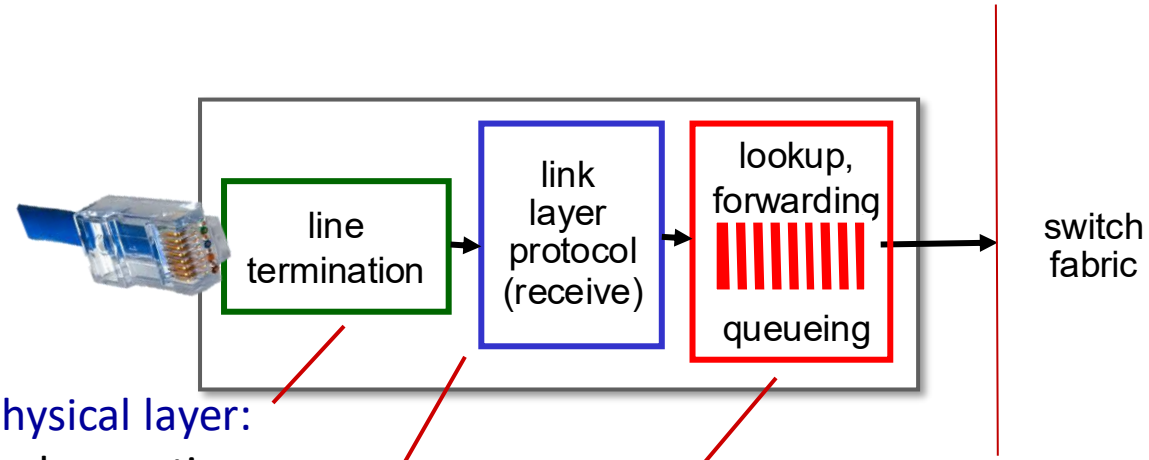
If 100 Gbps link

- **64-byte IP packet arrives**
- **5.12 nanosecs to process before next packet arrives**

If N ports, must be N times faster (or work in parallel!)

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Input port functions



Physical layer:
Bit-level reception
Convert electrical signal into bits

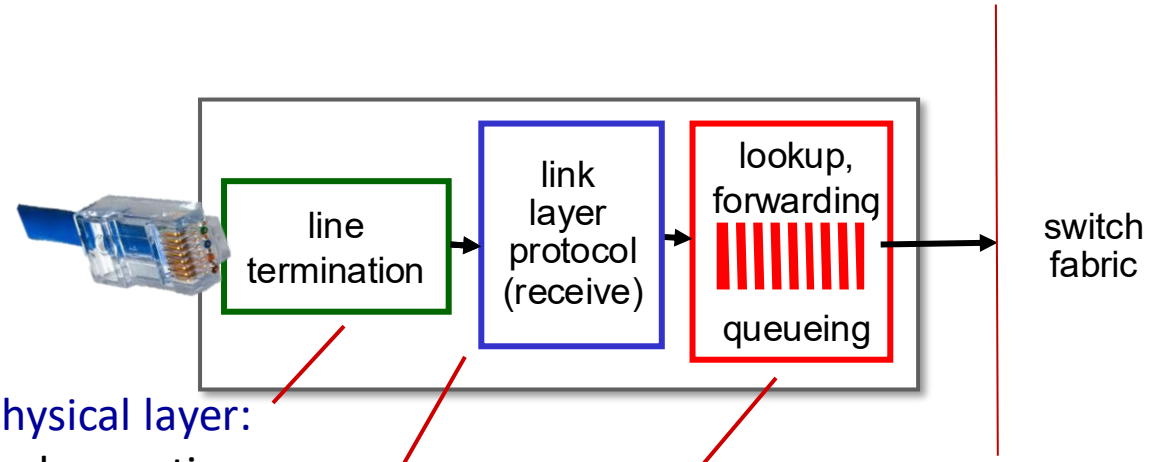
Link layer:
e.g., Ethernet (Layer 2)
Assemble into Ethernet frame (next class)
Check MAC address
Drop if not addressed to this router

Decentralized switching:

- Using header field values, lookup output port using forwarding table in input port memory *("match plus action")*
- Goal: complete input port processing at 'line speed'
- Input port queuing: if data arrive faster than forwarding rate into switch fabric, then queue for delivery

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Input port functions



Physical layer:
Bit-level reception
Convert electrical signal into bits

Link layer:
e.g., Ethernet (Layer 2)
Assemble into Ethernet frame (next class)
Check MAC address
Drop if not addressed to this router

Forwarding

- Destination-based forwarding: forward based only on destination IP address (traditional)
- Generalized forwarding: forward based on any set of header field values (SDN)

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Destination-based forwarding, look up destination from IP header

If packet passes version number, checksum and ttl

- Destination forwarding looks up output link based on Layer 3 destination IP address

So, we just use a hash table from CS10 to get O(1) look up, right?

- With 32-bit IP addresses there are 2^{32} = 4.2 billion possible addresses
- Can not have one entry per address!
- Instead create ranges of IP addresses for each output port

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Forwarding table gives output port for a range of destinations

Suppose a router has four output links
Numbered 0 to 3

If destination IP address in this range, go out this output port

forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Could have a look up table with only 4 entries
Just match the prefix to find the output link
But what if ranges do not fall out so nicely?

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Problem: Links may not fall out nicely to ranges of addresses

forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

This portion of the address range should go out port 3

Does not go out port 0 like the rest of the range does

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Match the longest prefix in the forwarding table

Longest prefix match

When looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

*** means match anything**

examples:

11001000 00010111 00010110 10100001	which interface?
11001000 00010111 00011000 10101010	which interface?

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Match the longest prefix in the forwarding table

Longest prefix match

When looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

match!

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Only the first range matches

Choose interface 0

Easy!

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Match the longest prefix in the forwarding table

Longest prefix match

When looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

match!

Matches interface 2

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Match the longest prefix in the forwarding table

Longest prefix match

When looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

match!

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Also matches interface 1

Choose longest (most specific)

Interface 1

Once output link found, ready to send to switch fabric to connect input link with output link

May queue here, however, if fabric is busy

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

TCAMS allow 1 clock cycle look up

- Longest prefix matching: often performed using Ternary Content Addressable Memories (TCAMs)
  - *Content addressable:* present address to TCAM: retrieve address in one clock cycle, regardless of table size
  - Cisco Catalyst: ~1M routing table entries in TCAM

- "Match plus action" protocol
  - Match: find output link from destination IP
  - Action: send packet to link

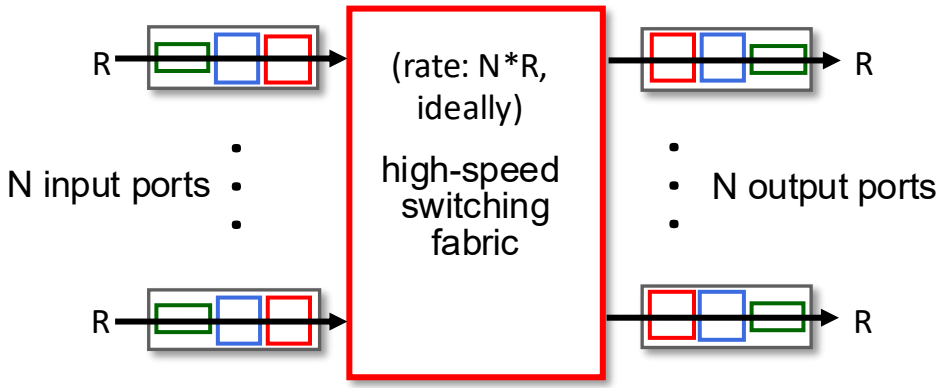
We will make "match plus action" more sophisticated later today

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Switching fabrics need to be fast to steer input data to output ports

Transfer packet from input link to appropriate output link

- Switching rate: rate at which packets can be transferred from inputs to outputs
  - Often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable

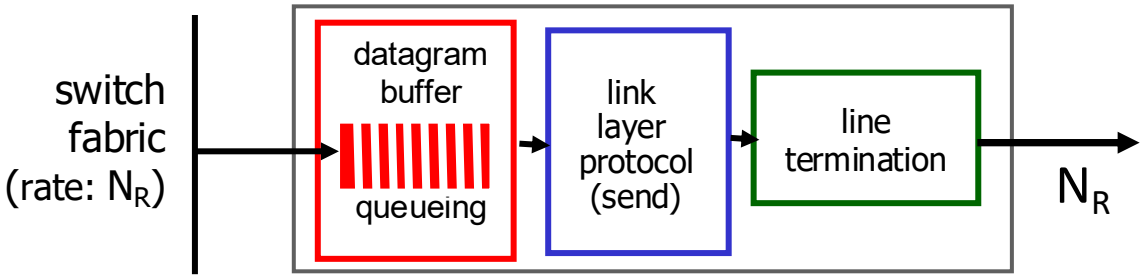


R = line speed
N = number of ports

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Output port queuing occurs if data arrives faster than it can be sent out a port

What if N packets arrive on N input ports, all destined for the same out port?
If $R_{line\ in} = R_{line\ out}$, then N-1 packets will buffer (at each time step!)



- *Buffering* required when datagrams arrive from fabric faster than outbound link transmission rate. *Drop policy:* which datagrams to drop if no free buffers?

  → Datagrams can be lost due to congestion, lack of buffers

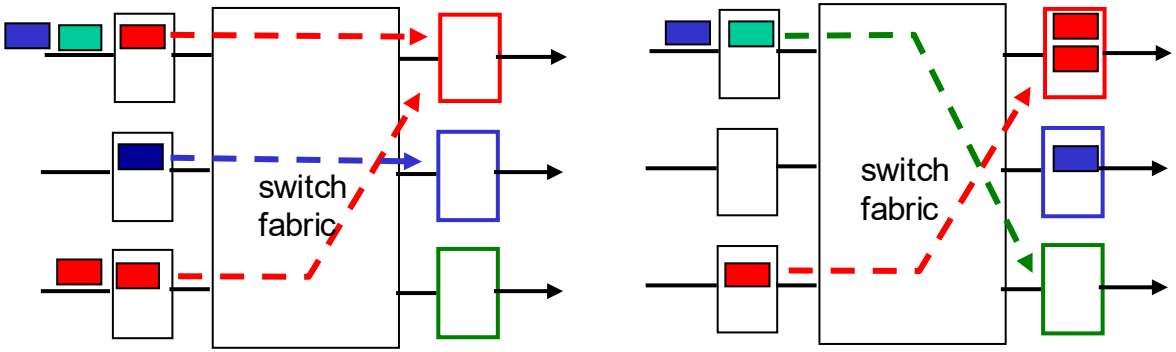
- *Scheduling discipline* chooses among queued datagrams for transmission

  → Priority scheduling – which datagrams get best performance (network neutrality issue)

- *Packet scheduler* chooses which buffered packet to send

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Output port queuing



at *t*, packets move from input to output

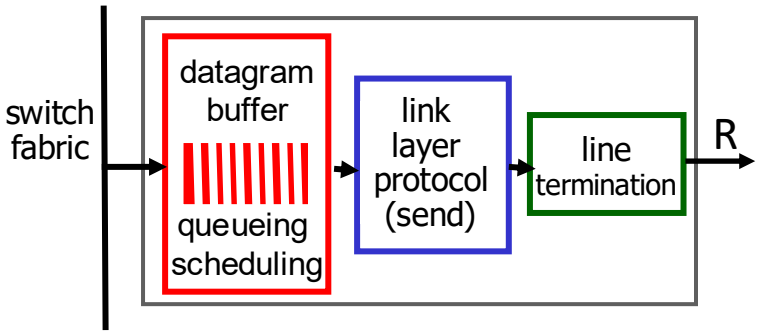
one packet time later

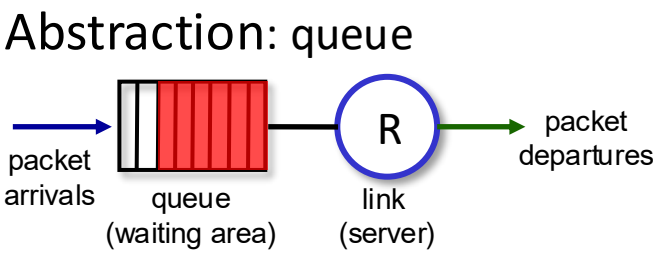
- Buffering when arrival rate via switch exceeds output line speed
- *Queueing (delay) and loss can occur due to output port buffer overflow!*

What algorithm to use to decide which packet to send?

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Buffer Management





Buffer management:

- Drop: which packet to add or drop when buffers are full
- Tail drop: drop arriving packet
- Priority: drop/remove on priority basis
- Marking: which packets to mark to signal congestion (we have not discussed Explicit Congestion Notification)

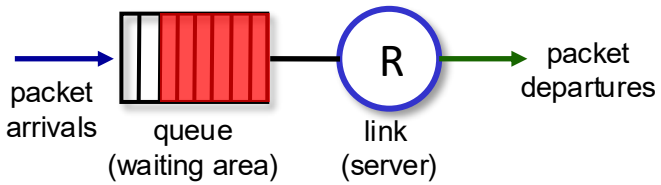
Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Packet Scheduling: FIFO

Packet scheduling: deciding which packet to send next on link

- First come, first served
- Priority
- Round robin
- Weighted fair queueing

Abstraction: queue



FIFO: packets transmitted in order of arrival to output port

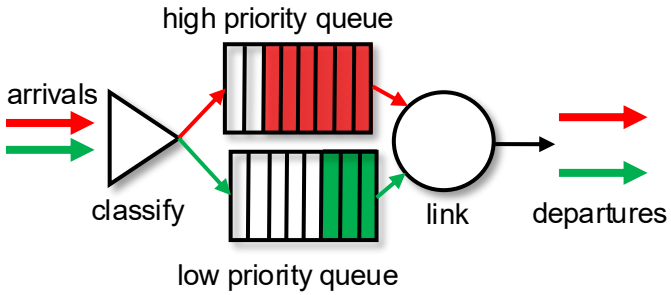
- Also known as: First Come First Served (FCFS)
- Recall Queues from CS10

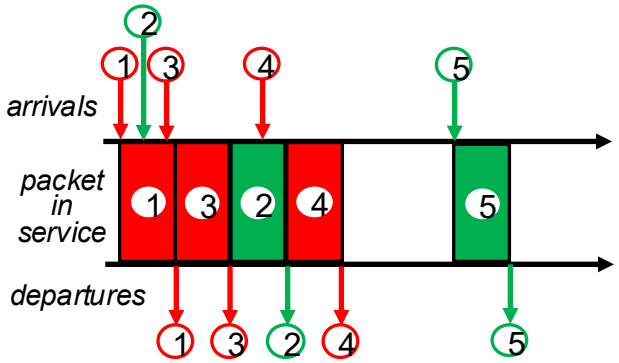
Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Scheduling policies: priority

Priority scheduling:

- Arriving traffic classified, queued by class
  - Any header field can be used for classification **Ex.: TOS field from IP layer**
    - Send packet from queue with highest priority that has buffered packets
      - FIFO within priority class





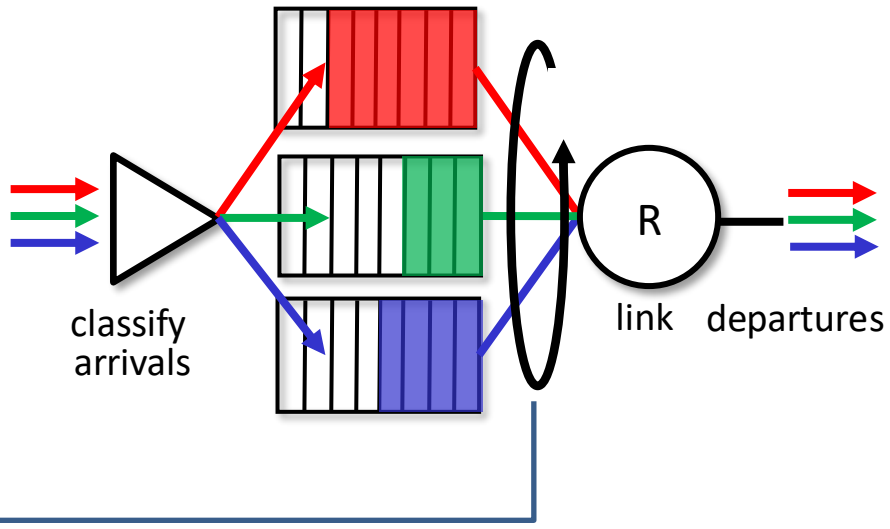
Packet 2 waits until high priority queue is empty

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Scheduling policies: round robin

Round Robin (RR) scheduling:

- Arriving traffic classified, queued by class
  - Any header fields can be used for classification
    - Server cyclically, repeatedly scans class queues, sending one complete packet from each class (if available) in turn



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

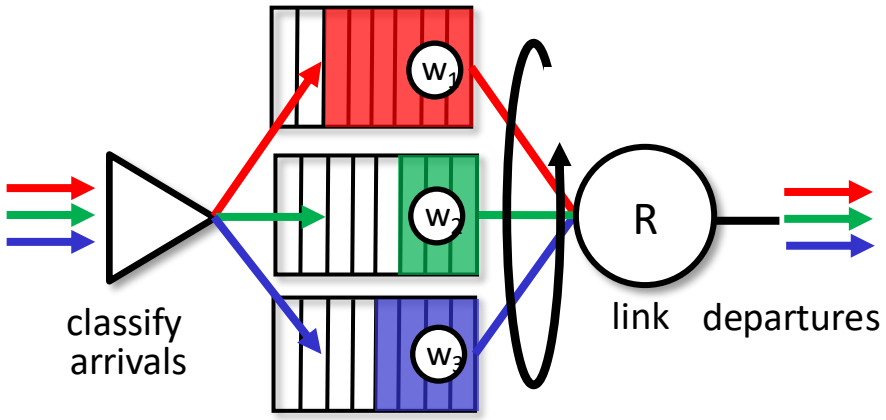
Scheduling policies: weighted fair queueing

Weighted Fair Queuing (WFQ):

- Generalized Round Robin
  - Each class, *i*, has weight, w_i, and gets weighted amount of service in each cycle:

$$\frac{w_i}{\sum_j w_j}$$

  - Minimum bandwidth guarantee (per-traffic-class)



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Generalized forwarding: match plus action

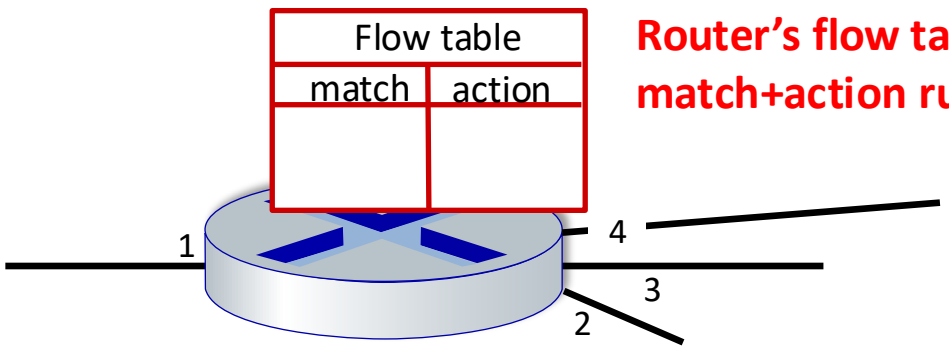
Review: each router contains a forwarding table (aka: flow table)

- "match plus action" abstraction: match bits in arriving packet, take action
  - *Destination-based forwarding:* forward based on destination IP address
  - *Generalized forwarding*:
    - Many header fields can determine action
    - Many actions possible: drop/copy/modify/log packet

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Generalized forwarding allows action on any header field (not just destination IP)

- Flow: defined by header field values (from link-, network-, transport-layer fields)
- Generalized forwarding: simple packet-handling rules
  - Match: pattern values in *any* packet header fields
  - Actions: for matched packet: drop, forward, modify, or send matched packet to controller
  - Priority: disambiguate overlapping patterns
  - Counters: #bytes and #packets

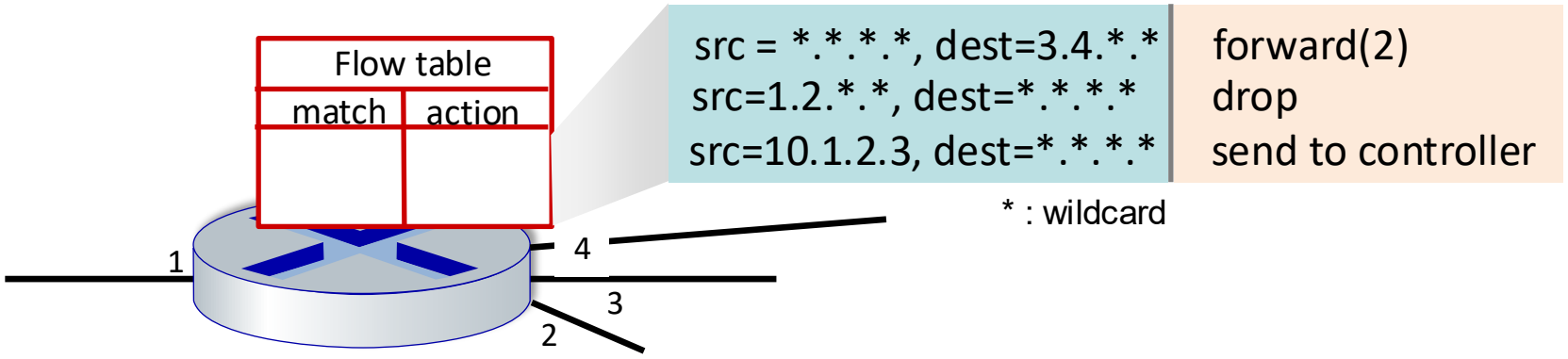


Router's flow table defines match+action rules

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

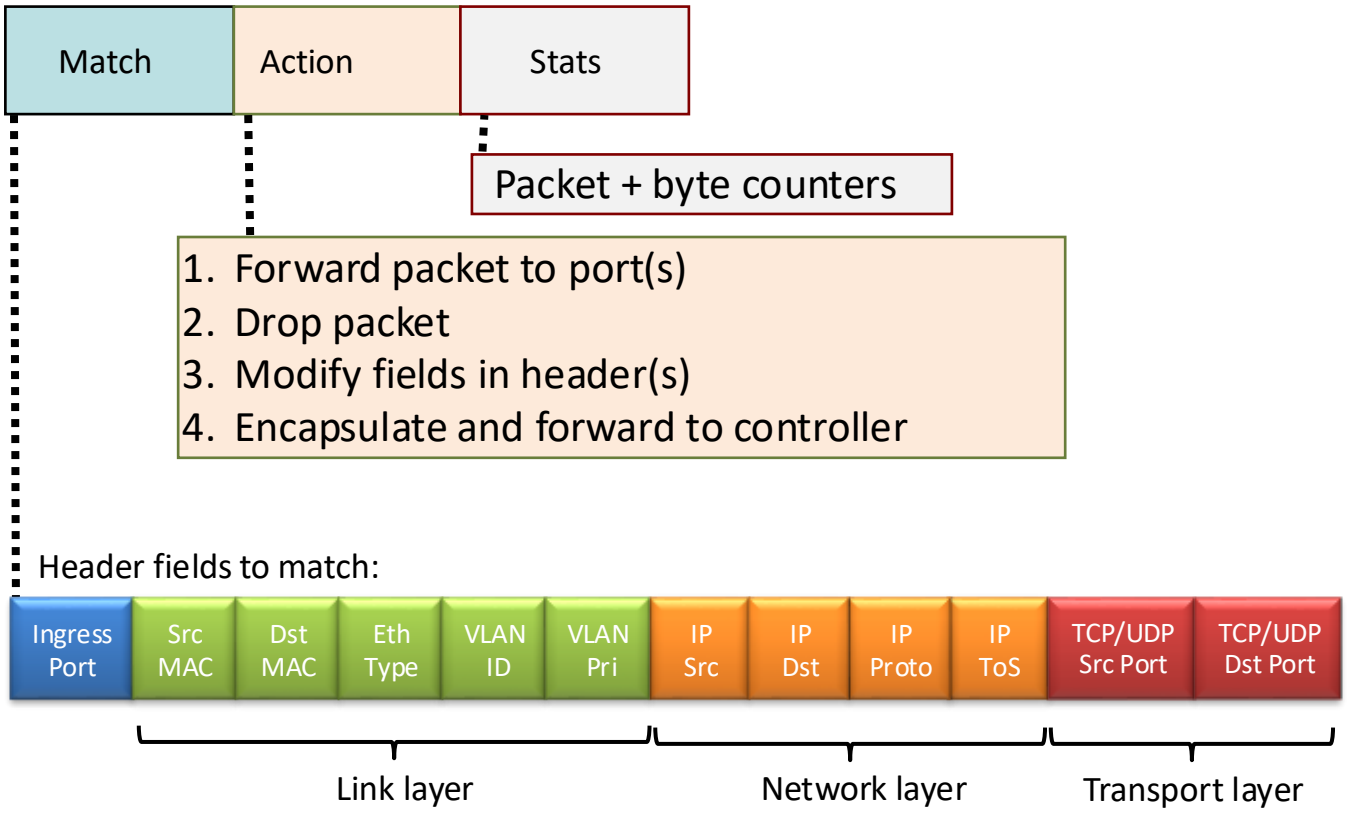
Flow table abstraction makes match plus action more powerful

- Flow: defined by header field values (from link-, network-, transport-layer fields)
- Generalized forwarding: simple packet-handling rules
  - Match: pattern values in *any* packet header fields
  - Actions: for matched packet: drop, forward, modify, or send matched packet to controller
  - Priority: disambiguate overlapping patterns
  - Counters: #bytes and #packets



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

OpenFlow: flow table entries allow actions other than just forward



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

OpenFlow: examples give a sense of the flexibility gained by generalized forwarding

Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	*	51.6.0.8	*	*	*	*	port 6

IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6

Firewall:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	*	*	*	*	*	22	drop

Block (do not forward) all datagrams destined to TCP port 22 (ssh port #)

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	128.119.1.1	*	*	*	*	*	drop

Block (do not forward) all datagrams sent by host 128.119.1.1

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

OpenFlow: examples give a sense of the flexibility gained by generalized forwarding

Layer 2 destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	22:A7:23:11:E1:02	*	*	*	*	*	*	*	*	*	port 3

Layer 2 frames with destination MAC address 22:A7:23:11:E1:02 should be forwarded to output port 3

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

OpenFlow abstraction allows a router to act like other types of network devices

- Match+action: abstraction unifies different kinds of devices

Router
- *Match:* longest destination IP prefix
- *Action:* forward out a link

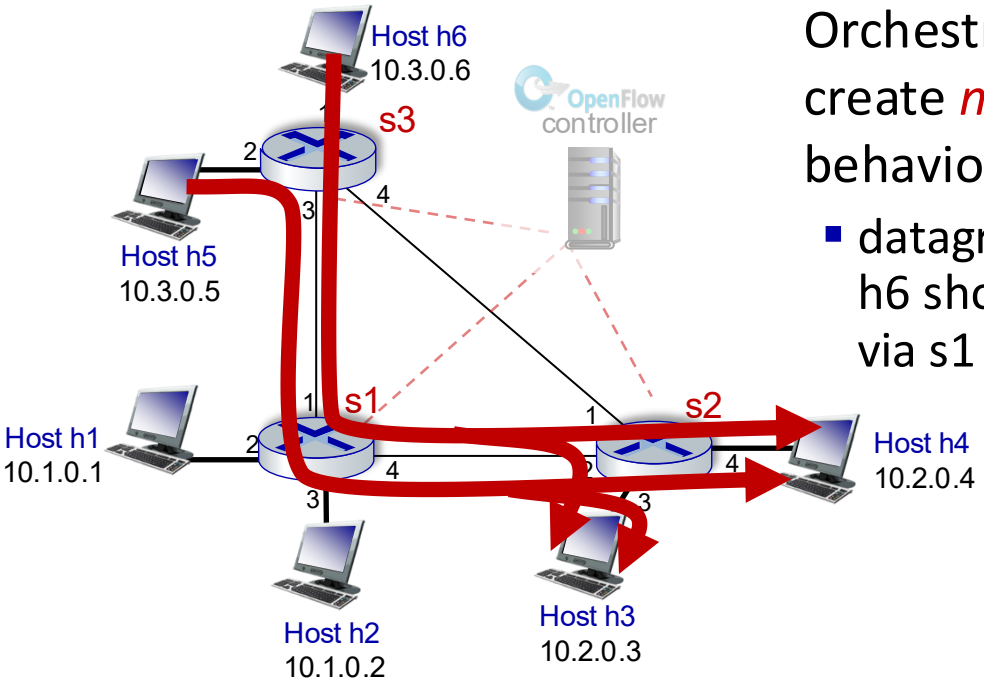
Switch
- *Match:* destination MAC address
- *Action:* forward or flood

Firewall
- *Match*: IP addresses and TCP/UDP port numbers
- *Action:* permit or deny

NAT
- *Match:* IP address and port
- *Action:* rewrite address and port

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

OpenFlow example

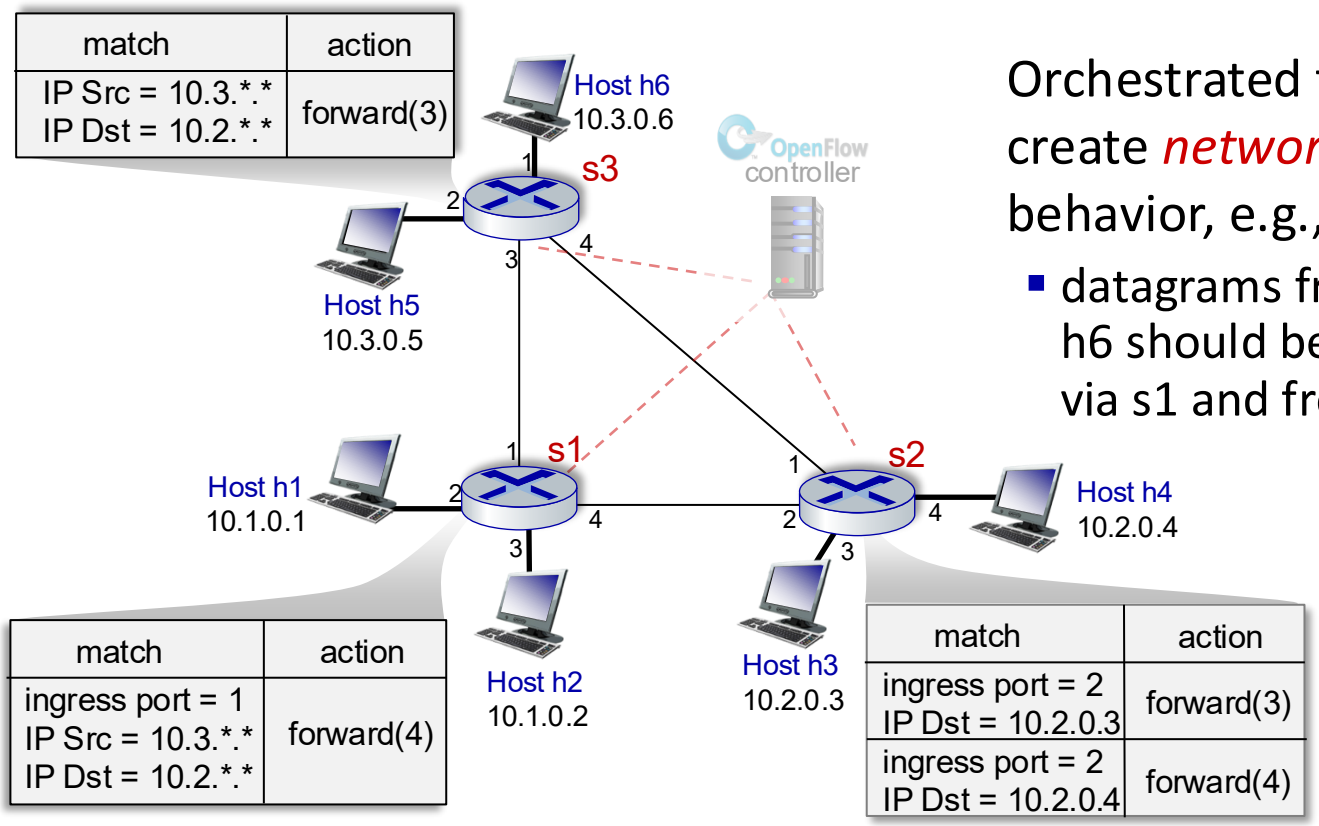


Orchestrated tables can create *network-wide* behavior, e.g.,:

- datagrams from hosts h5 and h6 should be sent to h3 or h4, via s1 and from there to s2

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

OpenFlow example



Orchestrated tables can create *network-wide* behavior, e.g.,:

- datagrams from hosts h5 and h6 should be sent to h3 or h4, via s1 and from there to s2

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Generalized forwarding: summary

- "Match plus action" abstraction: match bits in arriving packet header(s) in any layers, take action
  - Matching over many fields (link-, network-, transport-layer)
  - Local actions: drop, forward, modify, or send matched packet to controller
  - "Program" n*etwork-wide* behaviors
- Simple form of "network programmability"
  - Programmable, per-packet "processing"

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Agenda

1. Review of network layers
2. A router has two jobs:
    - Forwarding
    - Routing
3. Network (IP) Layer header
4. Forwarding
5. Exercise
6. Supplemental material

Exercise: Parse IP fields

parse_ip_starter.py

```
def parse_ipv4_packet(packet_bytes):
    #TODO: decode layer 3 fields


if __name__ == "__main__":
    try:
        # AF_PACKET: Sniff directly at the device driver level (Linux)
        # socket.SOCK_DGRAM: Drop Layer 2 headers, but keep Layer 3 headers
        # socket.htons(3): ETH_P_ALL captures ALL protocols (IP, ARP, IPv6, etc.) incoming & outgoing
        sniffer = socket.socket(socket.AF_PACKET, socket.SOCK_DGRAM, socket.htons(3))
        # sniffer.bind(("tun0", 0)) #capture only on this interface, otherwise capture on all interfaces


        print("Starting to sniff... (Ctrl+C to stop)")
    except PermissionError:
        print("You must run this script with 'sudo'")
        sys.exit(1)

    try:
        while True:
            # Receive IP packets (Max size 65535 bytes)
            raw_data, _ = sniffer.recvfrom(65535)
            if len(raw_data) < 20:
                print("Not an IP packet")
                continue


            # Extract packet fields
            packet_info = parse_ipv4_packet(raw_data)


    except KeyboardInterrupt:
        print("\nSniffer terminated by Ctrl+C")
```

Layer 3 header starts at packet_bytes
DGRAM removed Layer 2

Create socket

- **AF_PACKET: sniff directly at device driver**
- **SOCK_DGRAM: strip Layer 2 headers**
- **socket.htons(3): Capture all protocols**

Receive packets including Layer 3 and above headers (but not Layer 2 due to DGRAM)

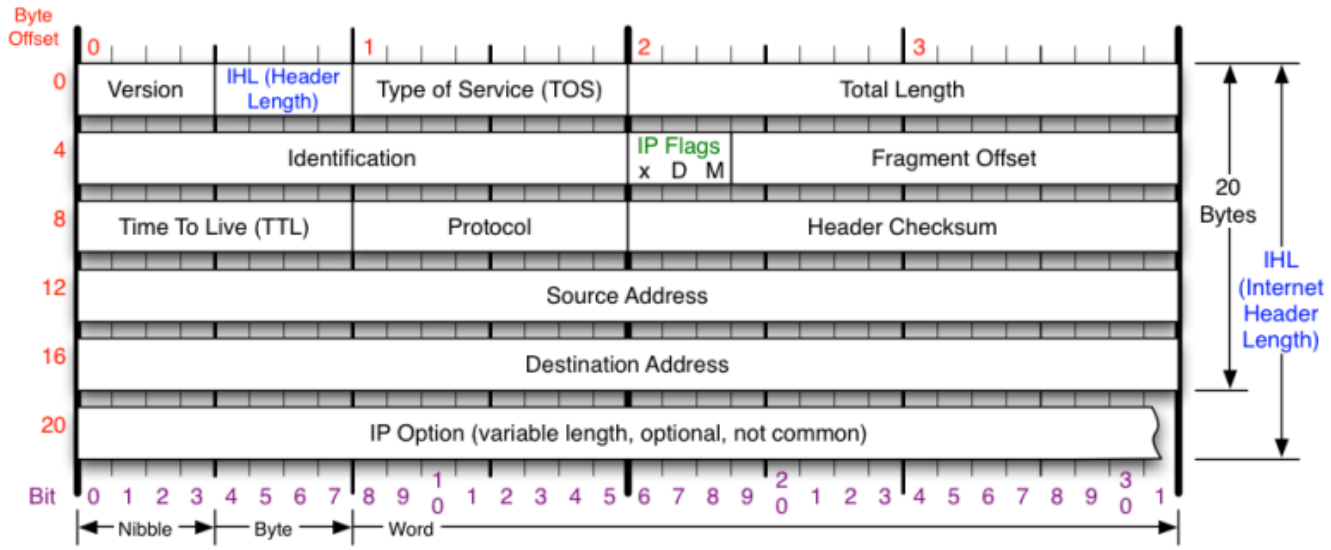
Parse bytes to extract header fields

Exercise: Parse IP fields

Begin with parse_ip_starter.py on the course web page for today

Parse Layer 3 bytes extracting:

- IP version (4 or 6)
- Header length (probably 20 bytes)
- Upper layer protocol:
  - ICMP =1
  - TCP = 6
  - UDP = 17
- Time to live
- Source IP
- Destination IP



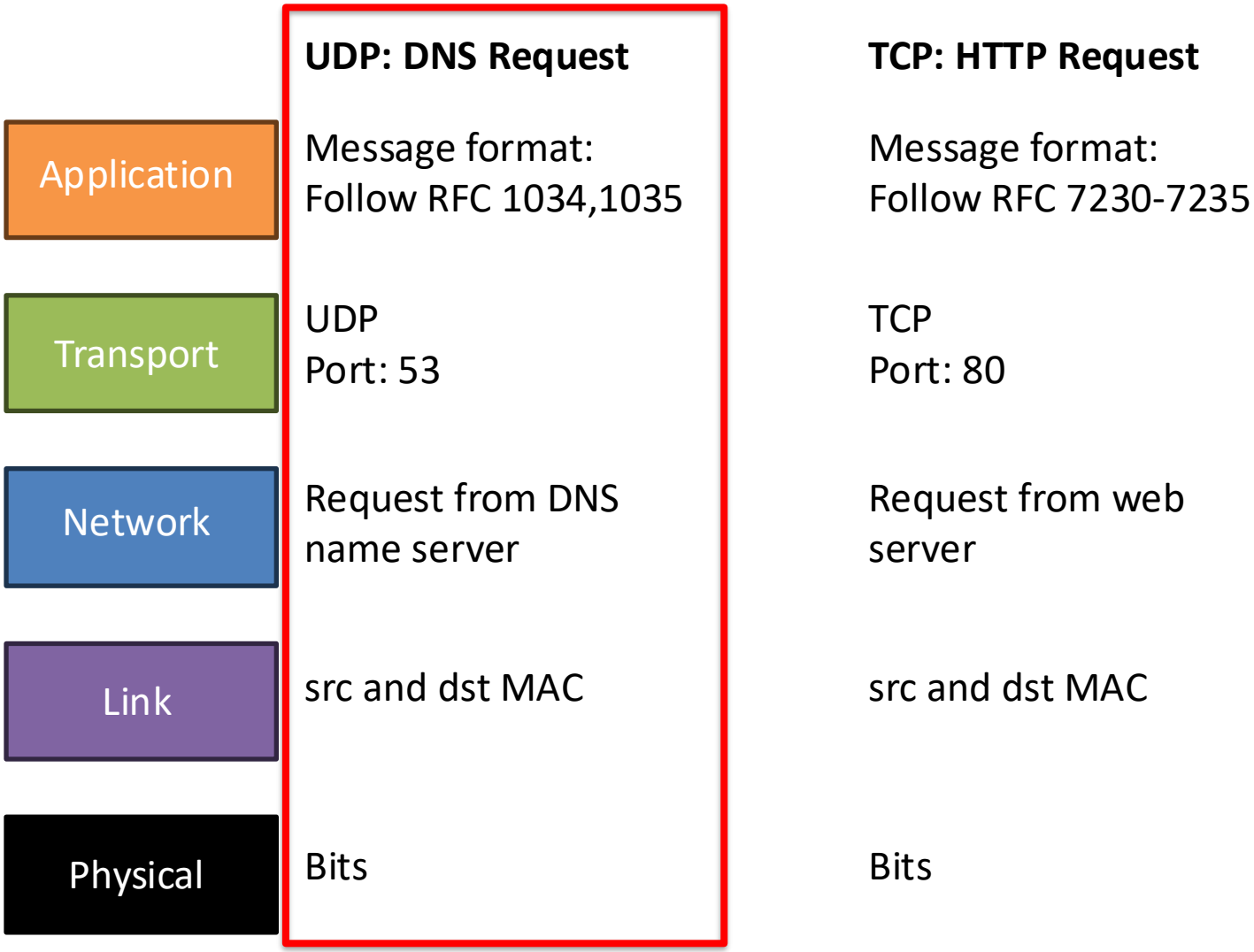
Start program running
Ping 8.8.8.8

Watch with Wireshark to confirm packets

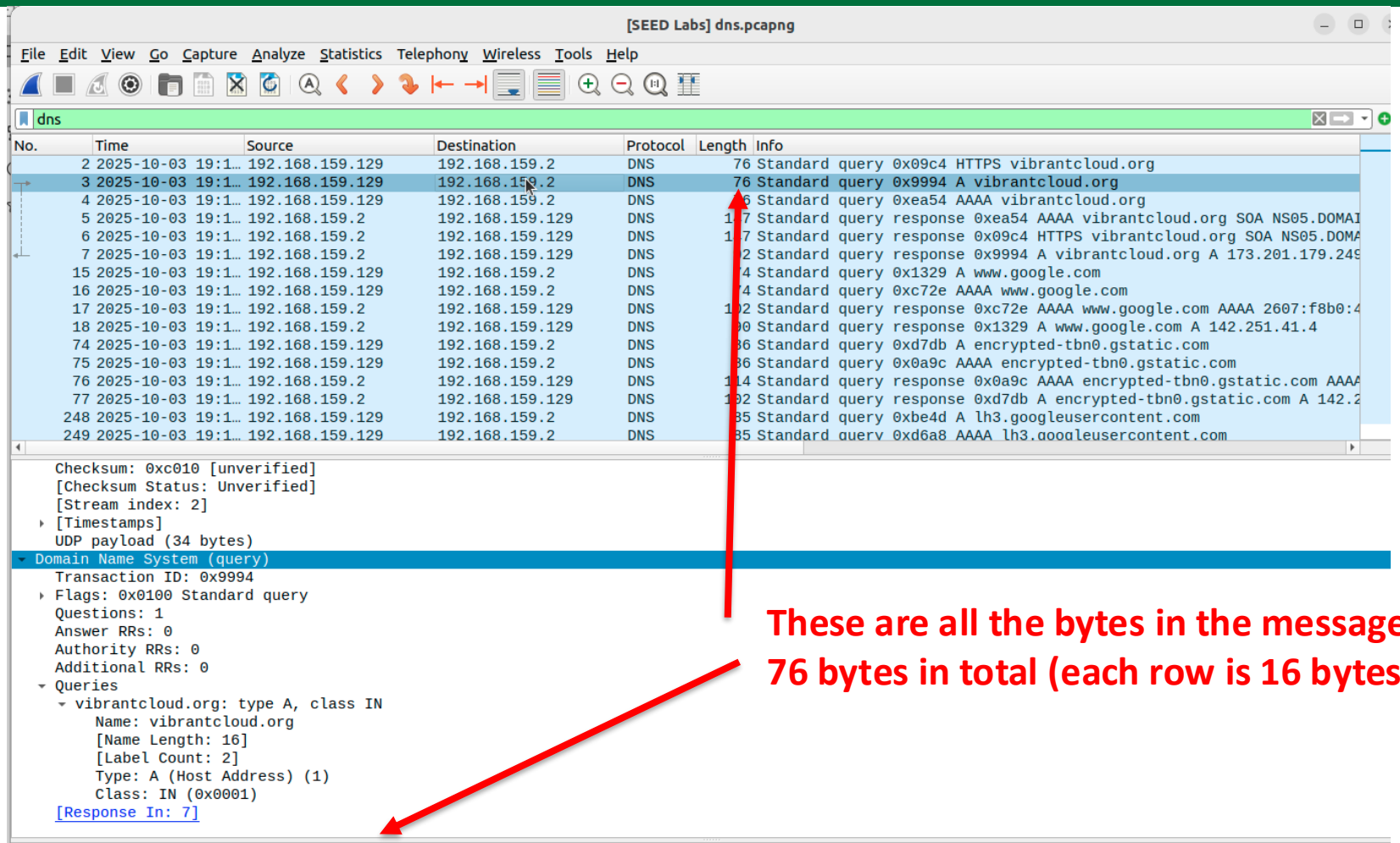
Agenda

1. Review of network layers
2. A router has two jobs:
    - Forwarding
    - Routing
3. Network (IP) Layer header
4. Forwarding
5. Exercise
6. Supplemental material

UDP and TCP requests



UDP: DNS request

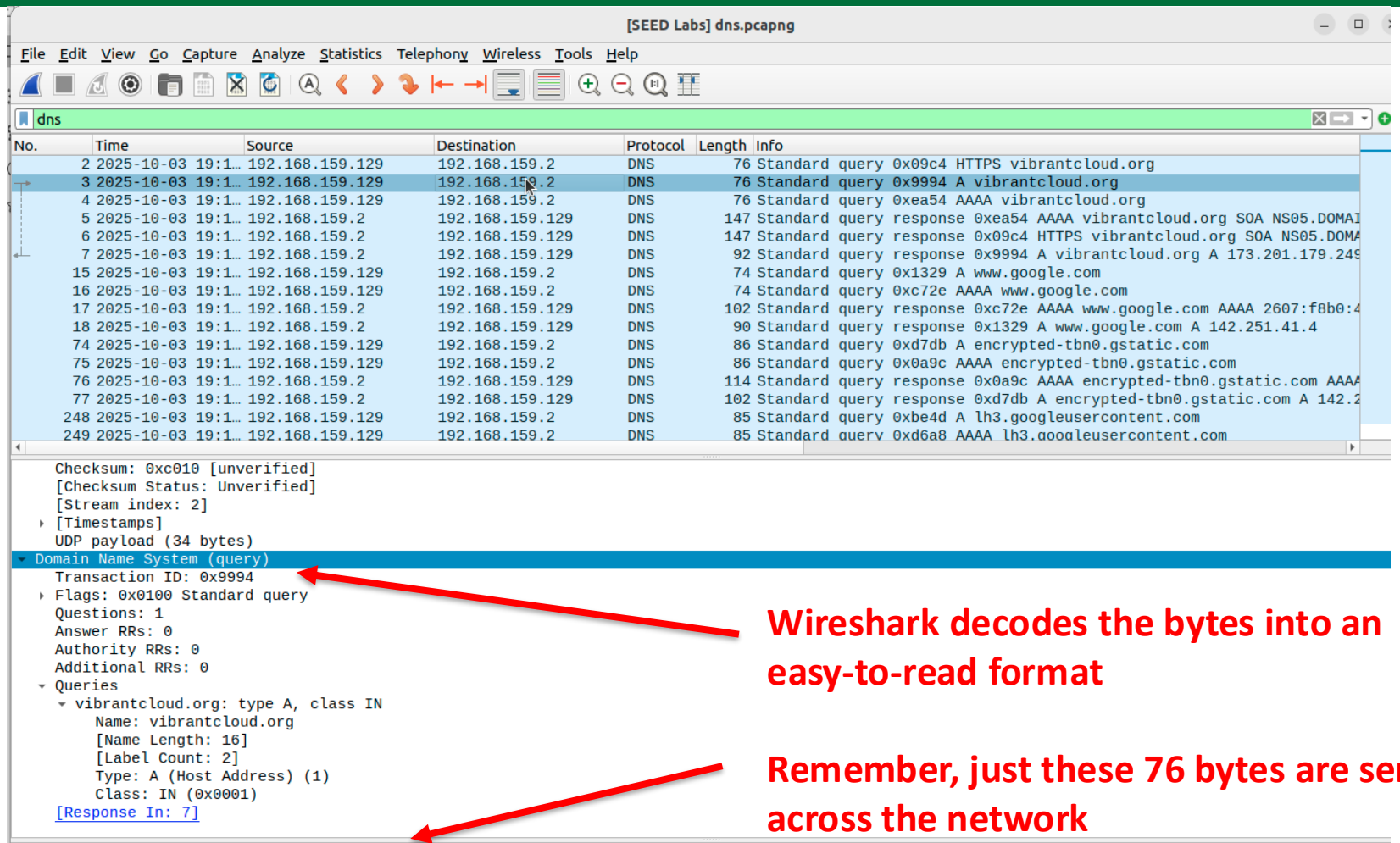


These are all the bytes in the message
76 bytes in total (each row is 16 bytes)

Layer 7 in this packet is a DNS query

App
Transport
Network
Link
Physical

UDP: DNS request

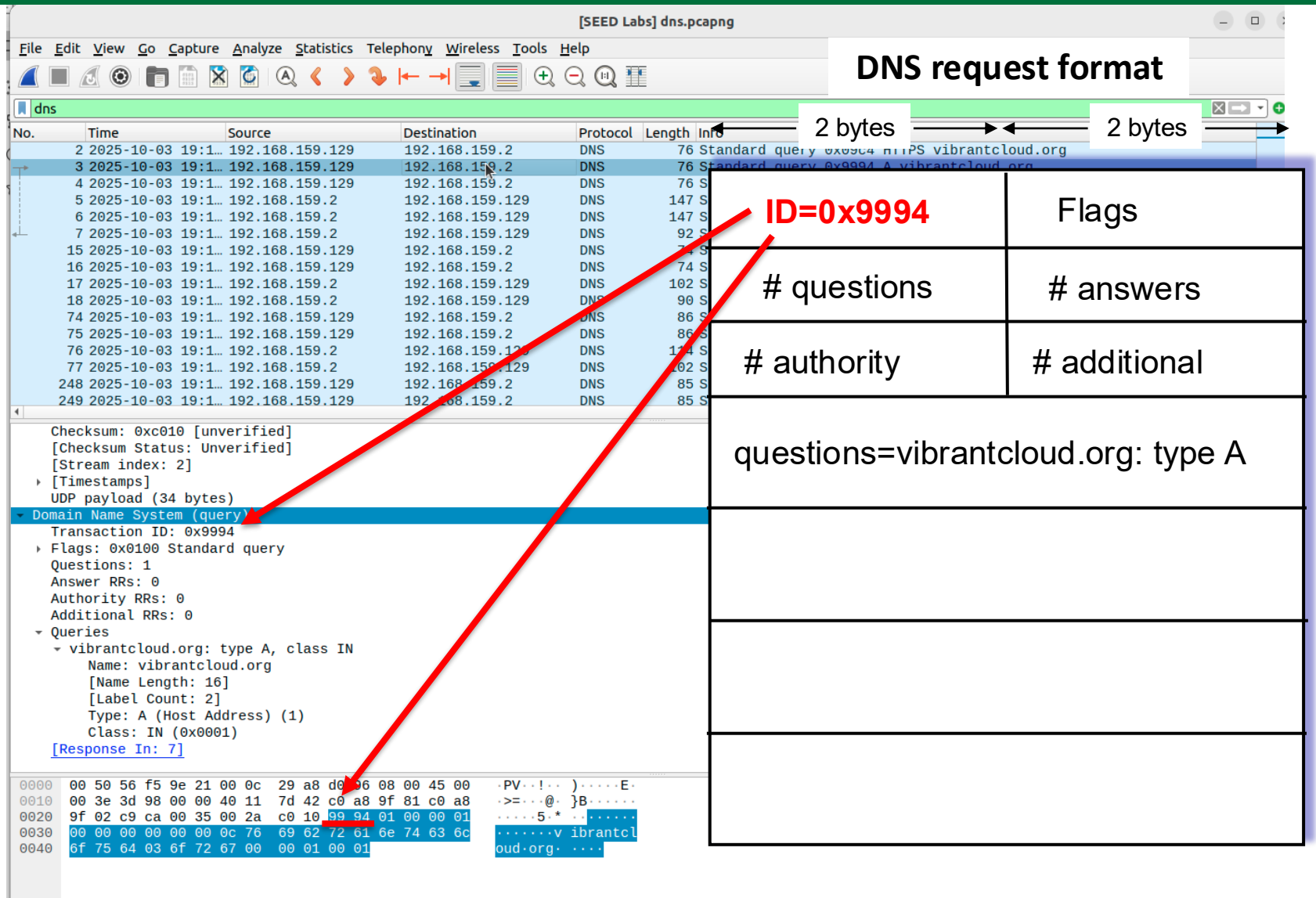


Wireshark decodes the bytes into an easy-to-read format

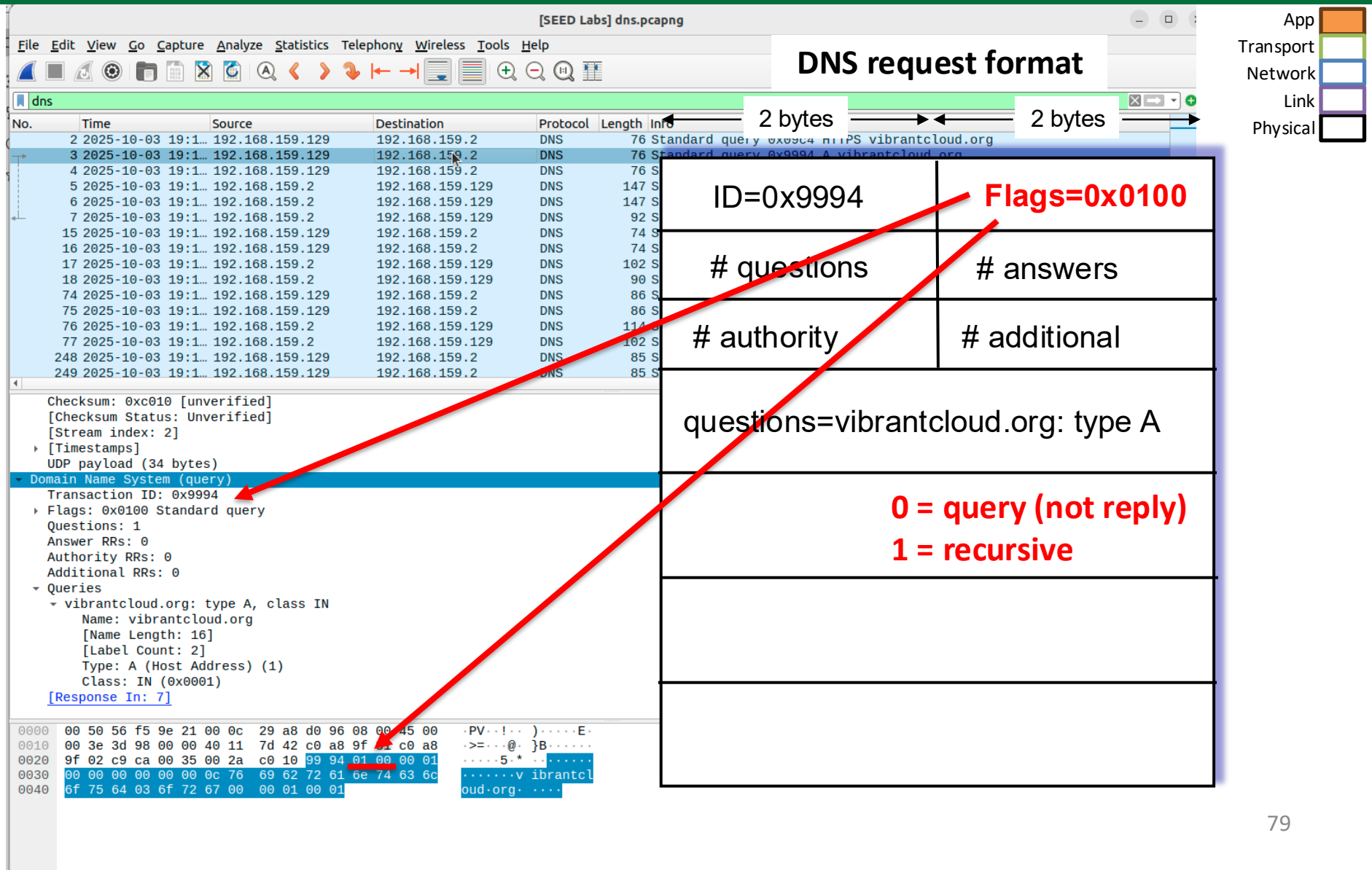
Remember, just these 76 bytes are sent across the network

App
Transport
Network
Link
Physical

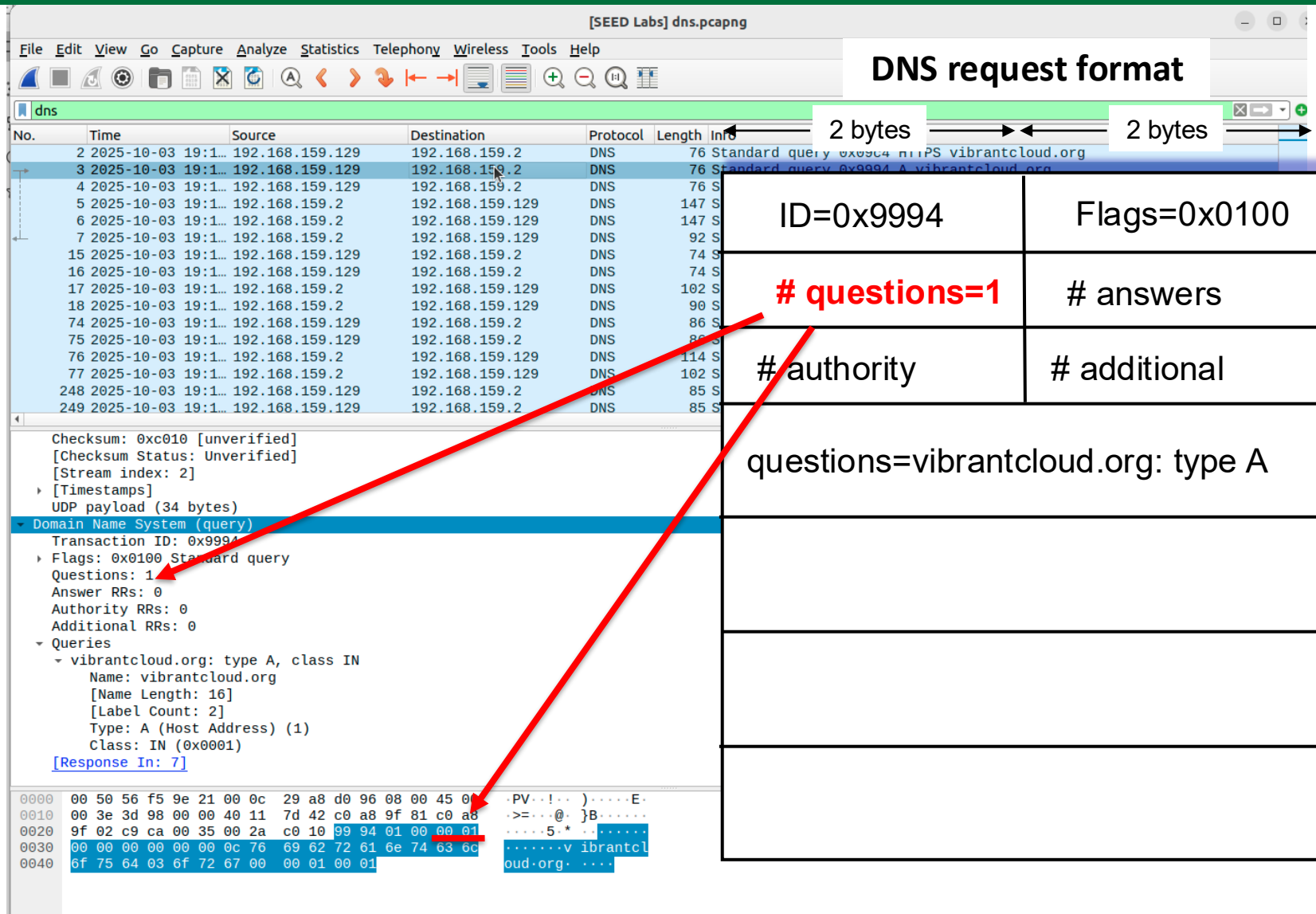
DNS Layer 7


[SEED Labs] dns.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
No. Time Source Destination Protocol Length Info
2 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 76 Standard query 0x09c4 HTTPS vibrantcloud.org
3 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 76 Standard query 0x9994 A vibrantcloud.org
4 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 76
5 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 147
6 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 147
7 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 92
15 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 74
16 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 74
17 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 102
18 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 90
74 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 86
75 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 86
76 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 114
77 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 102
248 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 85
249 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 85
Checksum: 0xc010 [unverified]
[Checksum Status: Unverified]
[Stream index: 2]
[Timestamps]
UDP payload (34 bytes)
Domain Name System (query)
Transaction ID: 0x9994
Flags: 0x0100 Standard query
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Queries
vibrantcloud.org: type A, class IN
Name: vibrantcloud.org
[Name Length: 16]
[Label Count: 2]
Type: A (Host Address) (1)
Class: IN (0x0001)
[Response In: 7]
0000 00 50 56 f5 9e 21 00 0c 29 a8 d0 96 08 00 45 00
0010 00 3e 3d 98 00 00 40 11 7d 42 c0 a8 9f 81 c0 a8
0020 9f 02 c9 ca 00 35 00 2a c0 10 99 94 01 00 00 01
0030 00 00 00 00 00 00 0c 76 69 62 72 61 6e 74 63 6c
0040 6f 75 64 03 6f 72 67 00 00 01 00 01
DNS request format
2 bytes
2 bytes
ID=0x9994
Flags
# questions
# answers
# authority
# additional
questions=vibrantcloud.org: type A
App
Transport
Network
Link
Physical

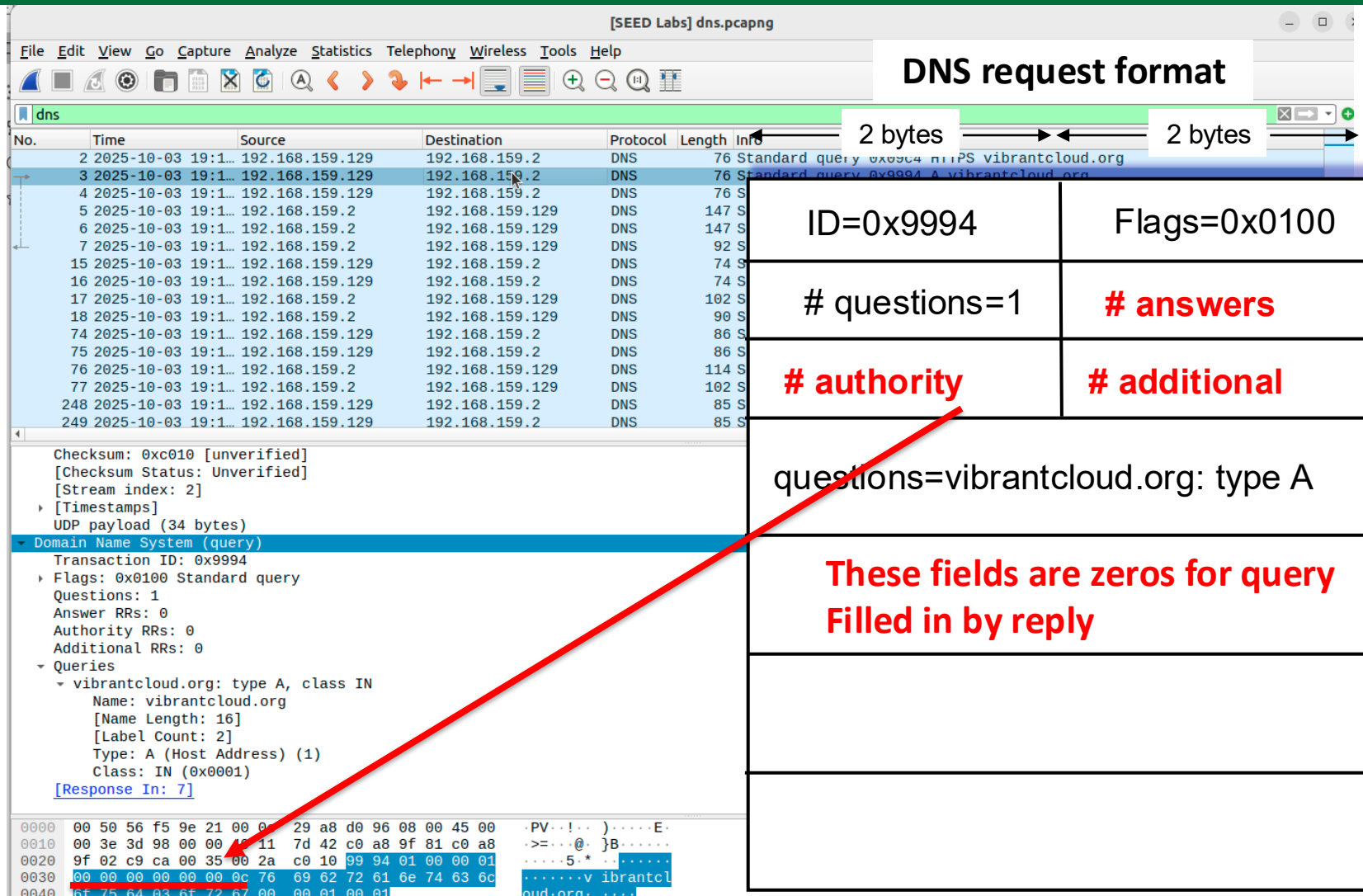
DNS Layer 7


[SEED Labs] dns.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
No. Time Source Destination Protocol Length Info
2 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 76 Standard query 0x0904 HTTPS vibrantcloud.org
3 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 76 Standard query 0x9994 A vibrantcloud.org
4 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 76 S
5 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 147 S
6 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 147 S
7 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 92 S
15 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 74 S
16 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 74 S
17 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 102 S
18 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 90 S
74 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 86 S
75 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 86 S
76 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 114 S
77 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 102 S
248 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 85 S
249 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 85 S
Checksum: 0xc010 [unverified]
[Checksum Status: Unverified]
[Stream index: 2]
[Timestamps]
UDP payload (34 bytes)
Domain Name System (query)
Transaction ID: 0x9994
Flags: 0x0100 Standard query
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Queries
vibrantcloud.org: type A, class IN
Name: vibrantcloud.org
[Name Length: 16]
[Label Count: 2]
Type: A (Host Address) (1)
Class: IN (0x0001)
[Response In: 7]
0000 00 50 56 f5 9e 21 00 0c 29 a8 d0 96 08 00 45 00 ·PV··!··)·····E·
0010 00 3e 3d 98 00 00 40 11 7d 42 c0 a8 9f 01 c0 a8 ·>=···@· }B····
0020 9f 02 c9 ca 00 35 00 2a c0 10 99 94 01 00 00 01 ·····5·* ····
0030 00 00 00 00 00 00 0c 76 69 62 72 61 6e 74 63 6c ·····v ibrantcl
0040 6f 75 64 03 6f 72 67 00 00 01 00 01 oud·org· ····
DNS request format
2 bytes
2 bytes
ID=0x9994
Flags=0x0100
# questions
# answers
# authority
# additional
questions=vibrantcloud.org: type A
0 = query (not reply)
1 = recursive
App
Transport
Network
Link
Physical

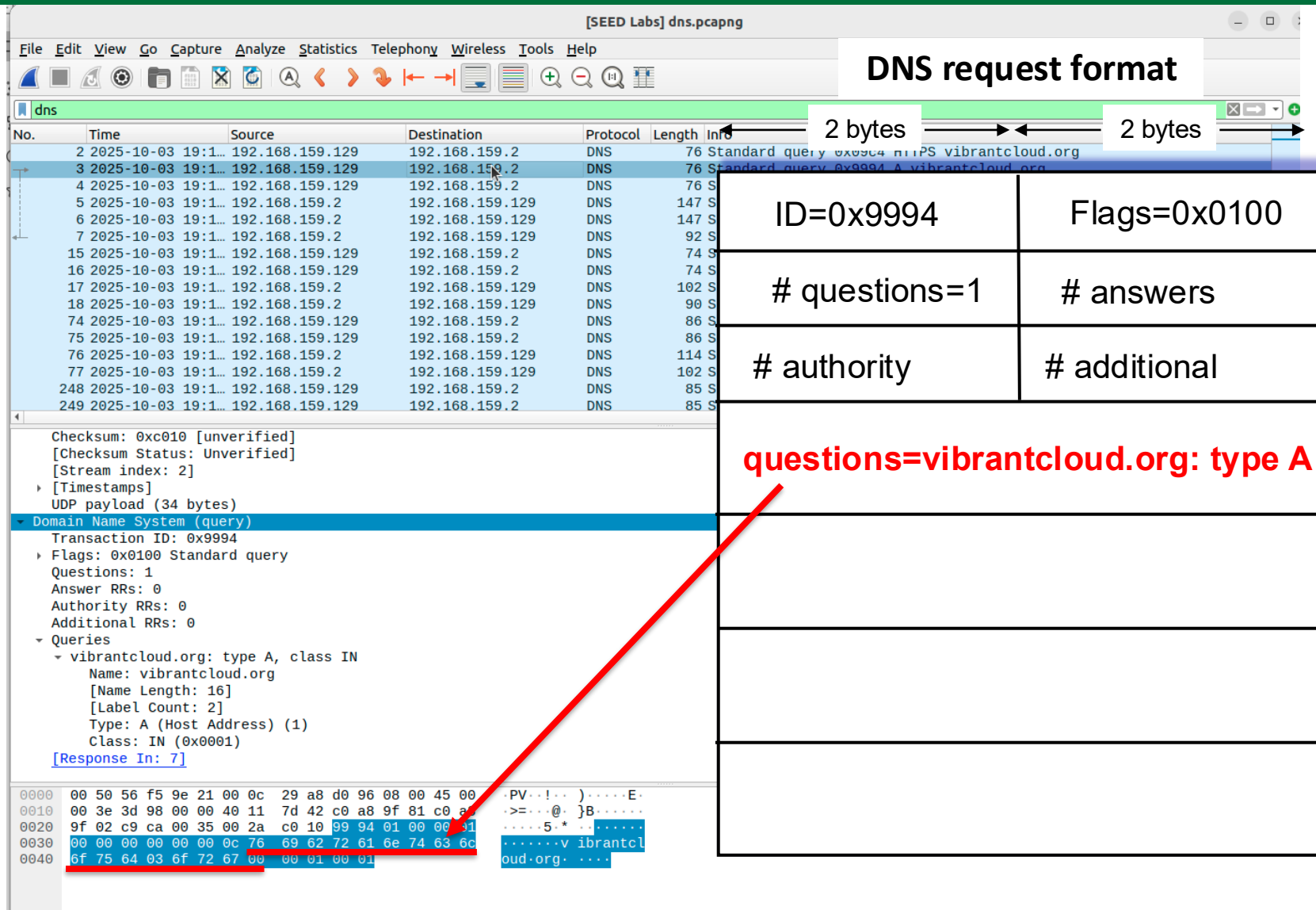
DNS Layer 7


[SEED Labs] dns.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
No. Time Source Destination Protocol Length Info
2 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 76 Standard query 0x0904 HTTPS vibrantcloud.org
3 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 76 Standard query 0x9994 A vibrantcloud.org
4 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 76 S
5 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 147 S
6 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 147 S
7 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 92 S
15 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 74 S
16 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 74 S
17 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 102 S
18 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 90 S
74 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 86 S
75 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 80 S
76 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 114 S
77 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 102 S
248 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 85 S
249 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 85 S
Checksum: 0xc010 [unverified]
[Checksum Status: Unverified]
[Stream index: 2]
[Timestamps]
UDP payload (34 bytes)
Domain Name System (query)
Transaction ID: 0x9994
Flags: 0x0100 Standard query
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Queries
vibrantcloud.org: type A, class IN
Name: vibrantcloud.org
[Name Length: 16]
[Label Count: 2]
Type: A (Host Address) (1)
Class: IN (0x0001)
[Response In: 7]
0000 00 50 56 f5 9e 21 00 0c 29 a8 d0 96 08 00 45 00
0010 00 3e 3d 98 00 00 40 11 7d 42 c0 a8 9f 81 c0 a8
0020 9f 02 c9 ca 00 35 00 2a c0 10 99 94 01 00 00 01
0030 00 00 00 00 00 00 0c 76 69 62 72 61 6e 74 63 6c
0040 6f 75 64 03 6f 72 67 00 00 01 00 01
DNS request format
2 bytes
2 bytes
ID=0x9994
Flags=0x0100
# questions=1
# answers
# authority
# additional
questions=vibrantcloud.org: type A
App
Transport
Network
Link
Physical

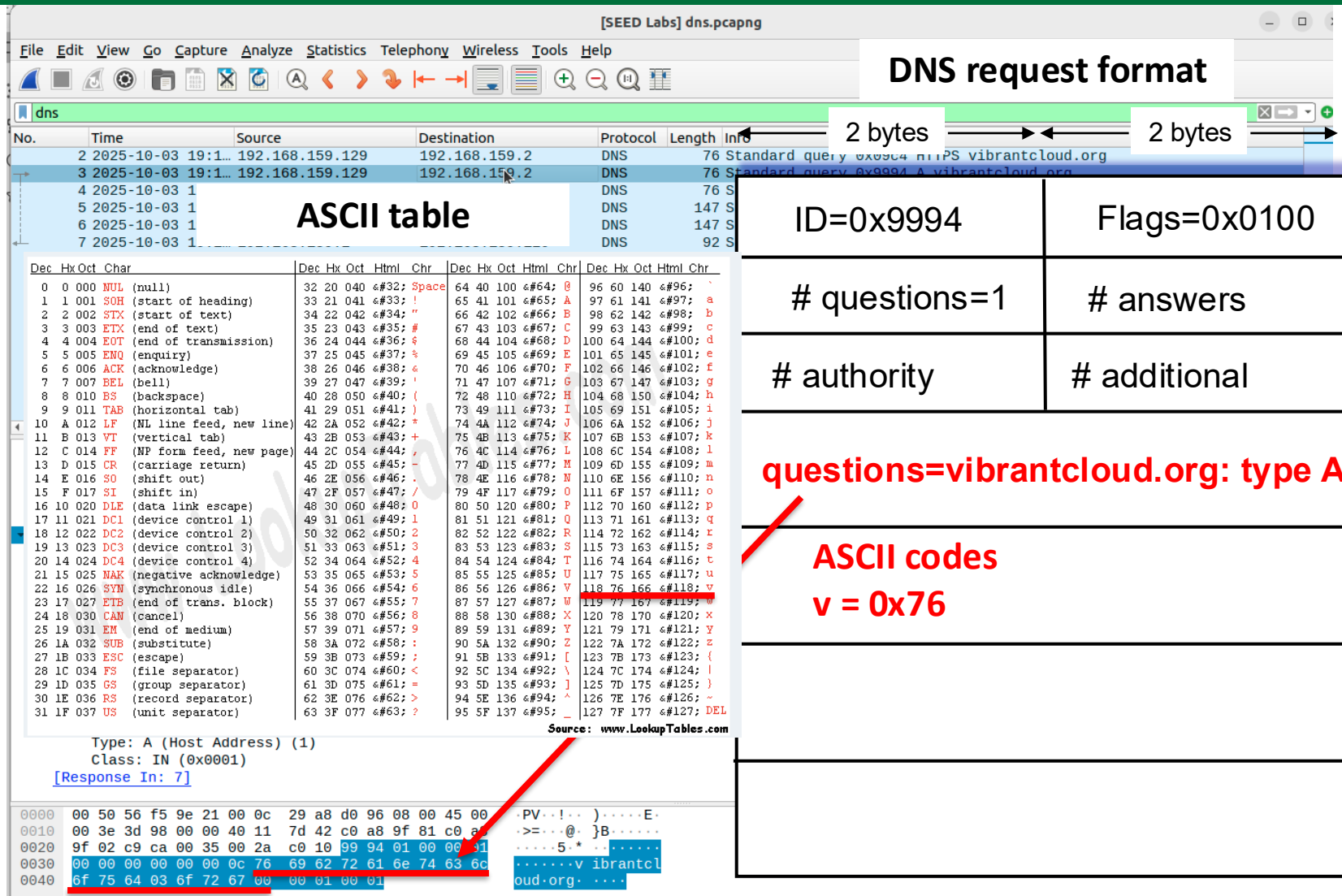
DNS Layer 7


[SEED Labs] dns.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
No. Time Source Destination Protocol Length Info
2 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 76 Standard query 0x0904 HTTPS vibrantcloud.org
3 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 76 Standard query 0x9994 A vibrantcloud.org
4 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 76 S
5 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 147 S
6 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 147 S
7 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 92 S
15 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 74 S
16 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 74 S
17 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 102 S
18 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 90 S
74 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 86 S
75 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 86 S
76 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 114 S
77 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 102 S
248 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 85 S
249 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 85 S
Checksum: 0xc010 [unverified]
[Checksum Status: Unverified]
[Stream index: 2]
[Timestamps]
UDP payload (34 bytes)
Domain Name System (query)
Transaction ID: 0x9994
Flags: 0x0100 Standard query
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Queries
vibrantcloud.org: type A, class IN
Name: vibrantcloud.org
[Name Length: 16]
[Label Count: 2]
Type: A (Host Address) (1)
Class: IN (0x0001)
[Response In: 7]
0000 00 50 56 f5 9e 21 00 0c 29 a8 d0 96 08 00 45 00
0010 00 3e 3d 98 00 00 40 11 7d 42 c0 a8 9f 81 c0 a8
0020 9f 02 c9 ca 00 35 00 2a c0 10 99 94 01 00 00 01
0030 00 00 00 00 00 00 0c 76 69 62 72 61 6e 74 63 6c
0040 6f 75 64 03 6f 72 67 00 00 01 00 01
DNS request format
2 bytes
2 bytes
ID=0x9994
Flags=0x0100
# questions=1
# answers
# authority
# additional
questions=vibrantcloud.org: type A
These fields are zeros for query
Filled in by reply
App
Transport
Network
Link
Physical

DNS Layer 7


[SEED Labs] dns.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
No. Time Source Destination Protocol Length Info
2 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 76 Standard query 0x09c4 HTTPS vibrantcloud.org
3 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 76 Standard query 0x9994 A vibrantcloud.org
4 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 76 S
5 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 147 S
6 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 147 S
7 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 92 S
15 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 74 S
16 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 74 S
17 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 102 S
18 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 90 S
74 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 86 S
75 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 86 S
76 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 114 S
77 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 102 S
248 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 85 S
249 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 85 S
Checksum: 0xc010 [unverified]
[Checksum Status: Unverified]
[Stream index: 2]
[Timestamps]
UDP payload (34 bytes)
Domain Name System (query)
Transaction ID: 0x9994
Flags: 0x0100 Standard query
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Queries
vibrantcloud.org: type A, class IN
Name: vibrantcloud.org
[Name Length: 16]
[Label Count: 2]
Type: A (Host Address) (1)
Class: IN (0x0001)
[Response In: 7]
0000 00 50 56 f5 9e 21 00 0c 29 a8 d0 96 08 00 45 00
0010 00 3e 3d 98 00 00 40 11 7d 42 c0 a8 9f 81 c0 a8
0020 9f 02 c9 ca 00 35 00 2a c0 10 99 94 01 00 00 01
0030 00 00 00 00 00 00 0c 76 69 62 72 61 6e 74 63 6c
0040 6f 75 64 03 6f 72 67 00 00 01 00 01
DNS request format
2 bytes
2 bytes
ID=0x9994
Flags=0x0100
# questions=1
# answers
# authority
# additional
questions=vibrantcloud.org: type A
App
Transport
Network
Link
Physical

DNS Layer 7



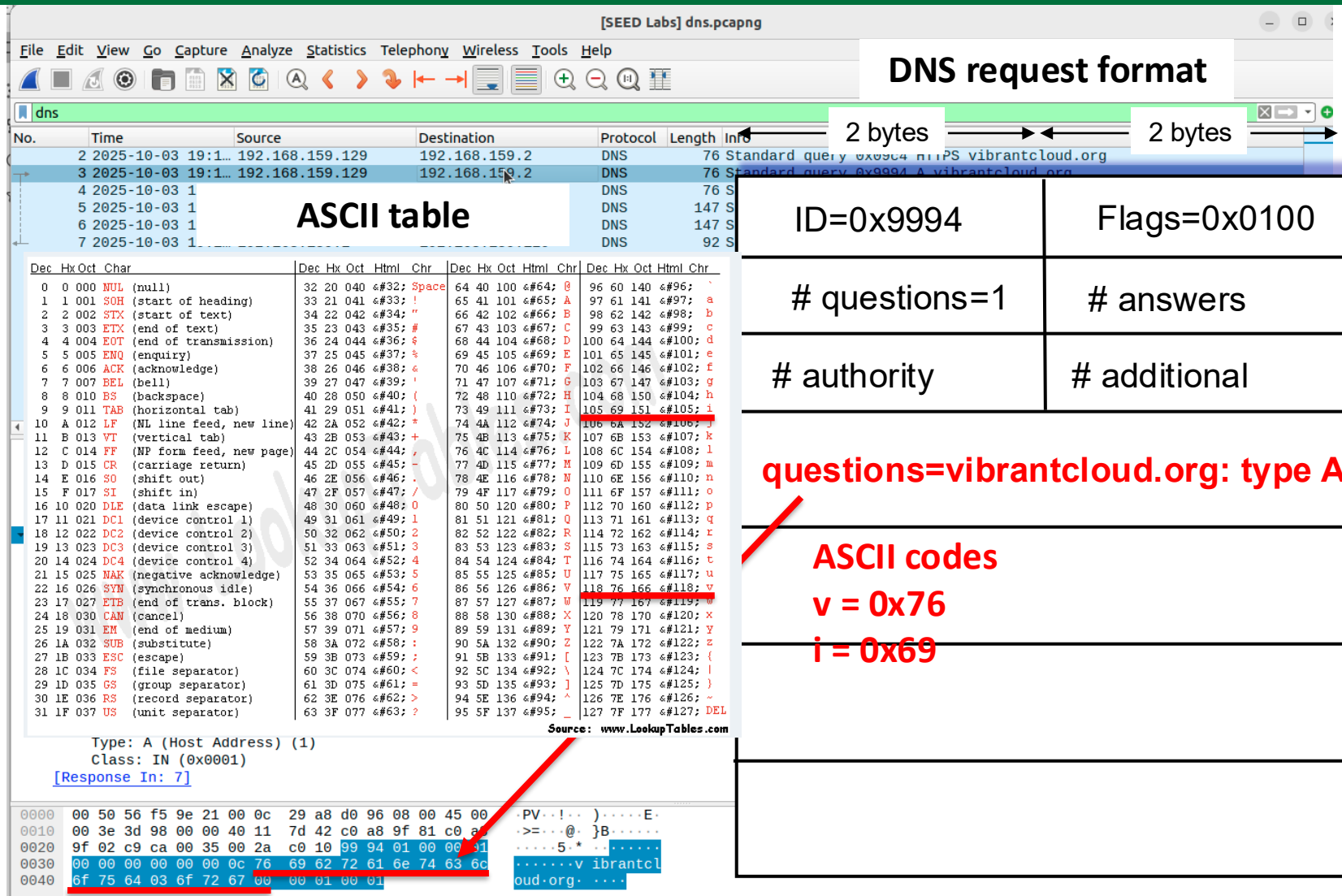
DNS request format

2 bytes | 2 bytes

ID=0x9994	Flags=0x0100
# questions=1	# answers
# authority	# additional
questions=vibrantcloud.org: type A	
ASCII codes v = 0x76	

App
Transport
Network
Link
Physical

DNS Layer 7



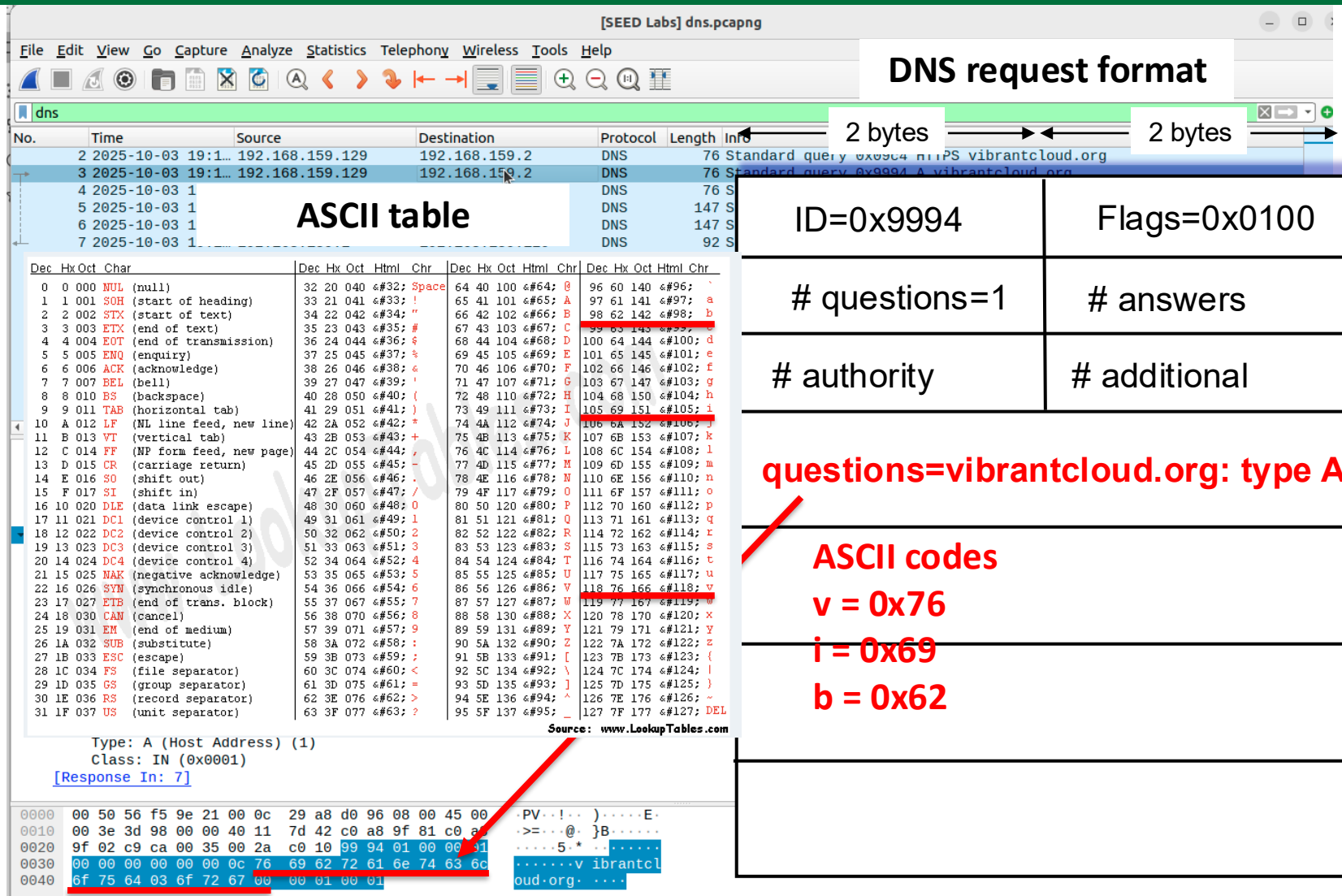
DNS request format

2 bytes | 2 bytes

ID=0x9994	Flags=0x0100
# questions=1	# answers
# authority	# additional
questions=vibrantcloud.org: type A	
ASCII codes v = 0x76 i = 0x69	

App
Transport
Network
Link
Physical

DNS Layer 7



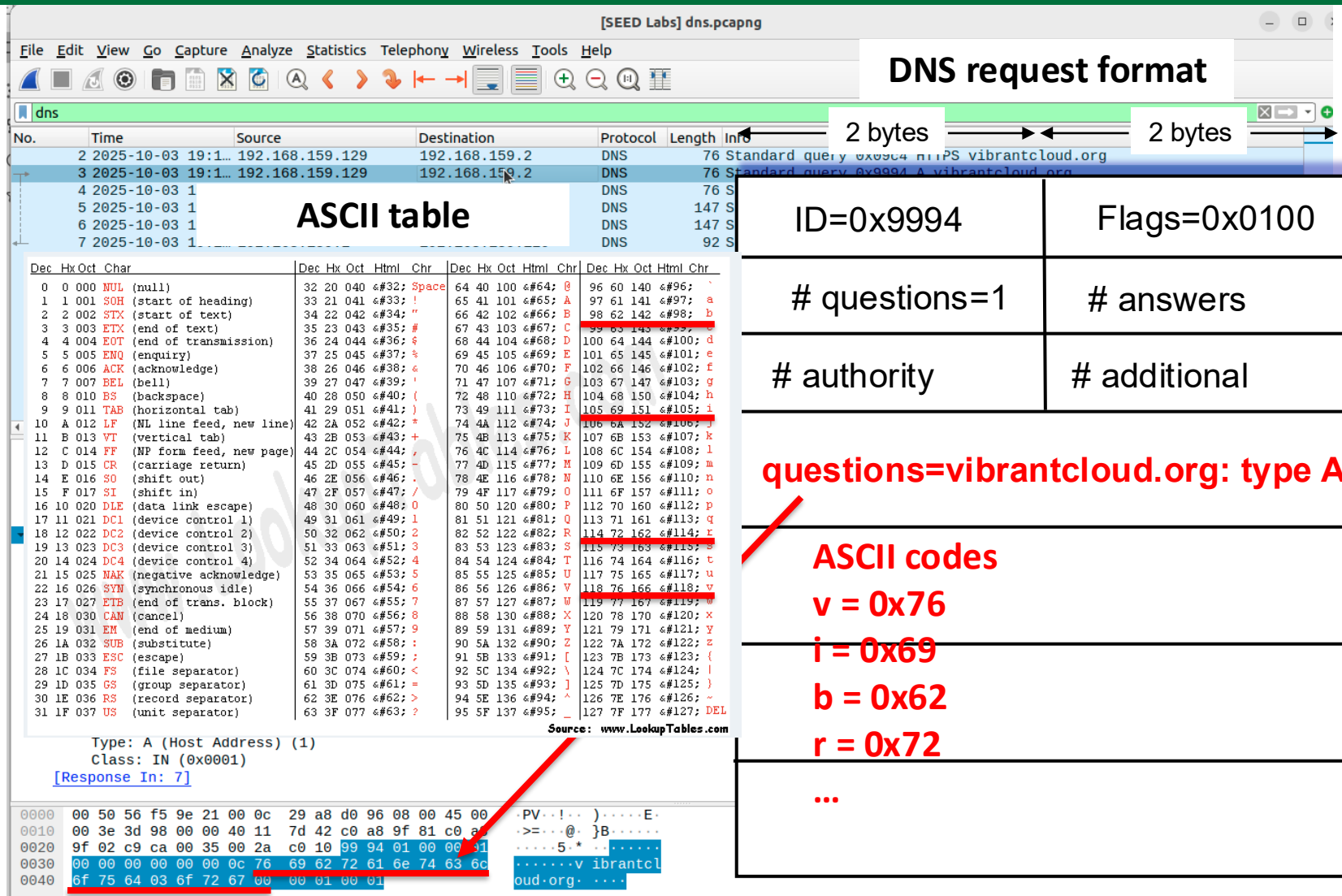
DNS request format

2 bytes | 2 bytes

ID=0x9994	Flags=0x0100
# questions=1	# answers
# authority	# additional
questions=vibrantcloud.org: type A	
ASCII codes v = 0x76 i = 0x69 b = 0x62	

App
Transport
Network
Link
Physical

DNS Layer 7



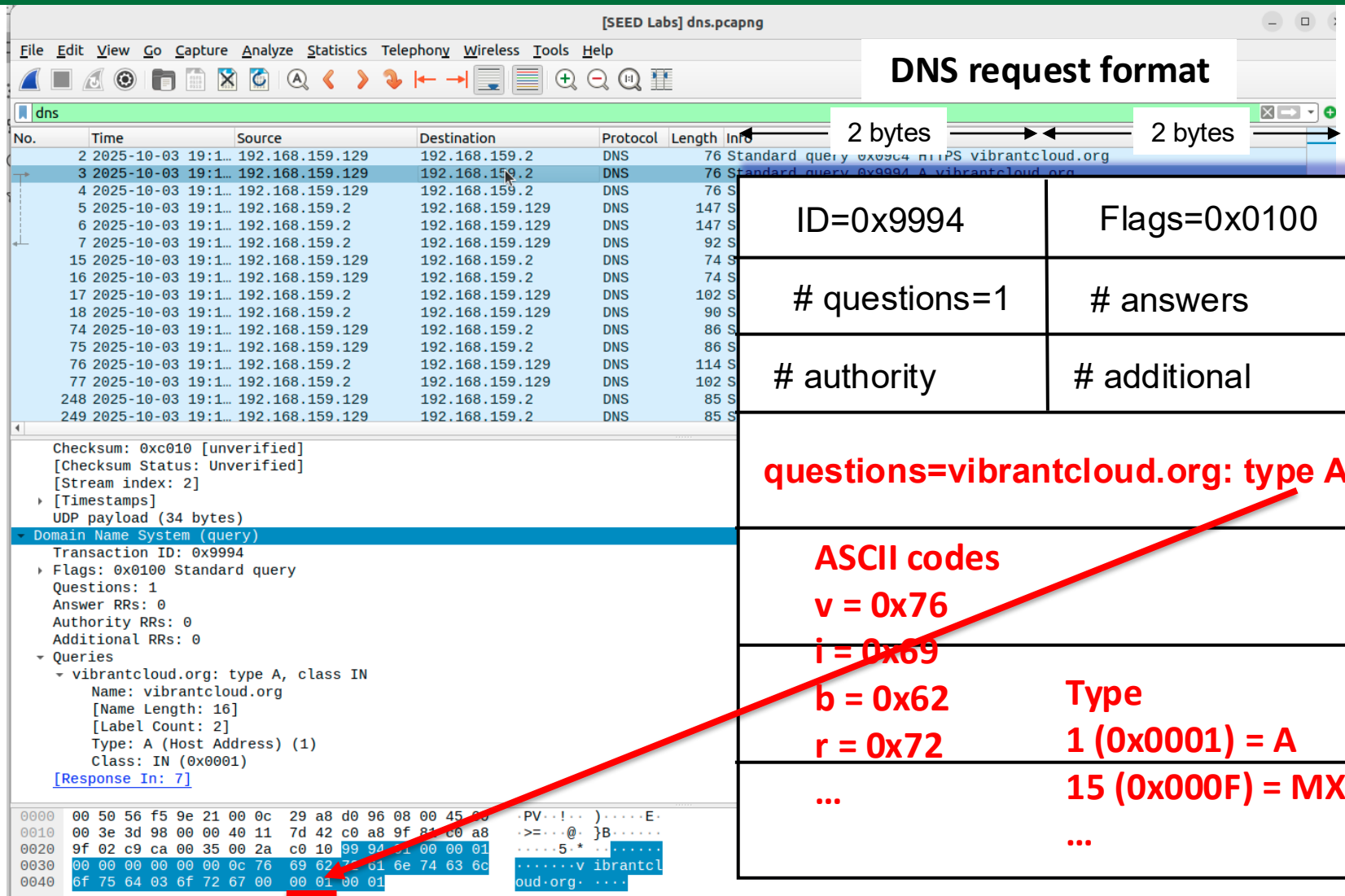
DNS request format

2 bytes | 2 bytes

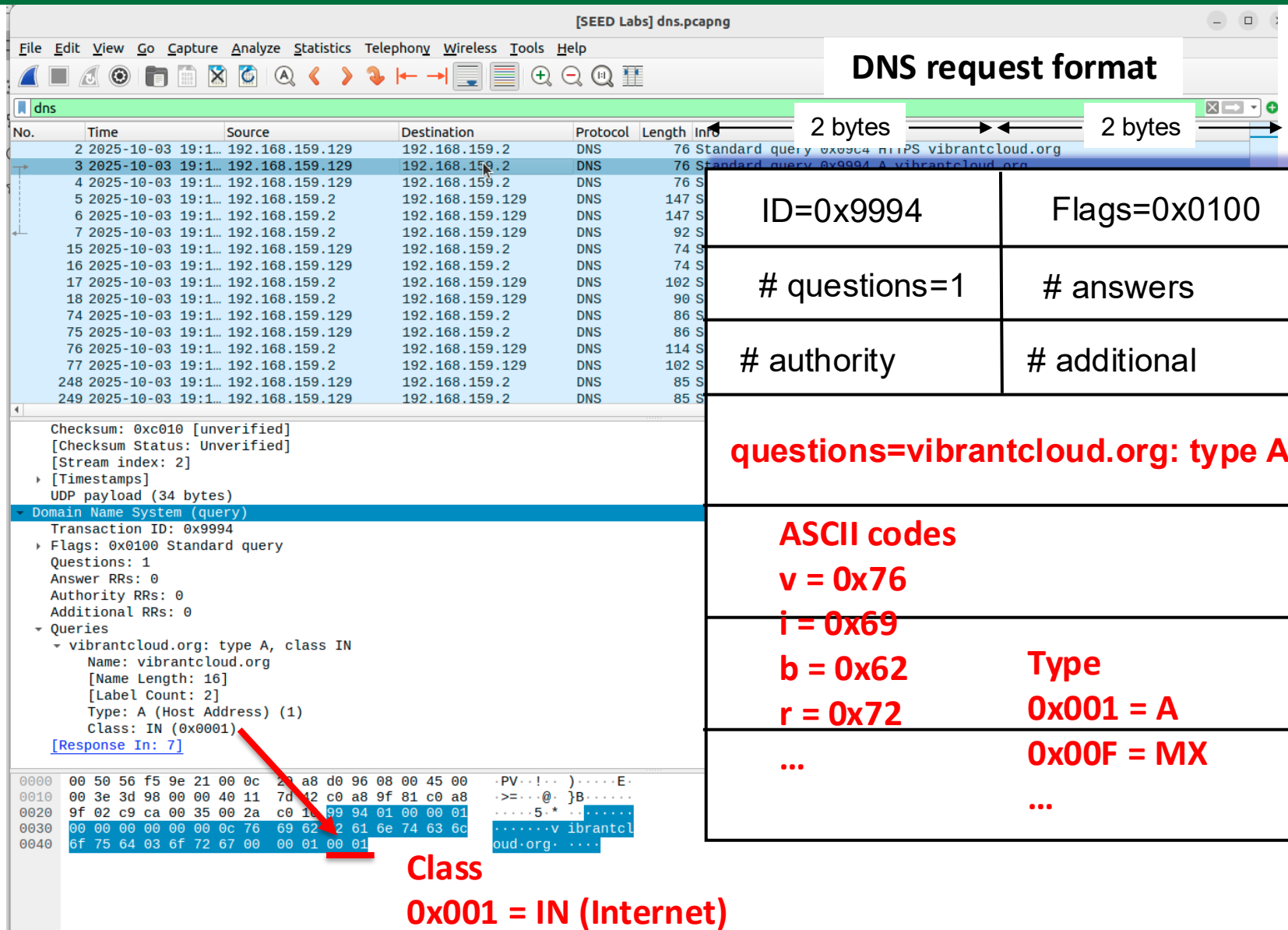
ID=0x9994	Flags=0x0100
# questions=1	# answers
# authority	# additional
questions=vibrantcloud.org: type A	
ASCII codes v = 0x76 i = 0x69 b = 0x62 r = 0x72 …	

App
Transport
Network
Link
Physical

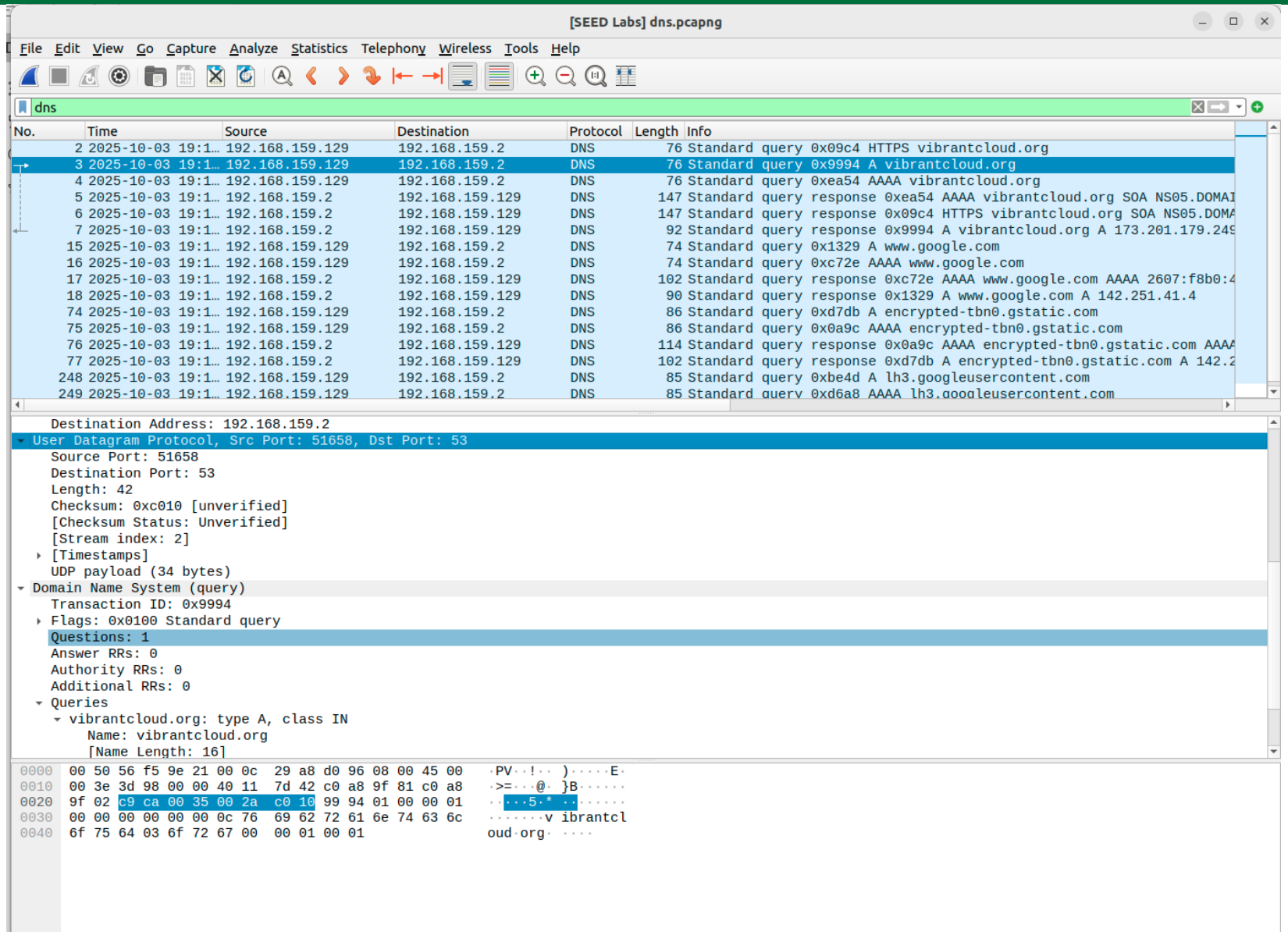
DNS Layer 7


[SEED Labs] dns.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
No. Time Source Destination Protocol Length Info
2 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 76 Standard query 0x09c4 HTTPS vibrantcloud.org
3 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 76 Standard query 0x9994 A vibrantcloud.org
4 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 76 S
5 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 147 S
6 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 147 S
7 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 92 S
15 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 74 S
16 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 74 S
17 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 102 S
18 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 90 S
74 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 86 S
75 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 86 S
76 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 114 S
77 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 102 S
248 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 85 S
249 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 85 S
Checksum: 0xc010 [unverified]
[Checksum Status: Unverified]
[Stream index: 2]
[Timestamps]
UDP payload (34 bytes)
Domain Name System (query)
Transaction ID: 0x9994
Flags: 0x0100 Standard query
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Queries
vibrantcloud.org: type A, class IN
Name: vibrantcloud.org
[Name Length: 16]
[Label Count: 2]
Type: A (Host Address) (1)
Class: IN (0x0001)
[Response In: 7]
0000 00 50 56 f5 9e 21 00 0c 29 a8 d0 96 08 00 45 c0
0010 00 3e 3d 98 00 00 40 11 7d 42 c0 a8 9f 81 c0 a8
0020 9f 02 c9 ca 00 35 00 2a c0 10 99 94 01 00 00 01
0030 00 00 00 00 00 00 0c 76 69 62 72 61 6e 74 63 6c
0040 6f 75 64 03 6f 72 67 00 00 01 00 01
DNS request format
2 bytes
2 bytes
ID=0x9994
Flags=0x0100
# questions=1
# answers
# authority
# additional
questions=vibrantcloud.org: type A
ASCII codes
v = 0x76
i = 0x69
b = 0x62
r = 0x72
...
Type
1 (0x0001) = A
15 (0x000F) = MX
...
App
Transport
Network
Link
Physical

DNS Layer 7

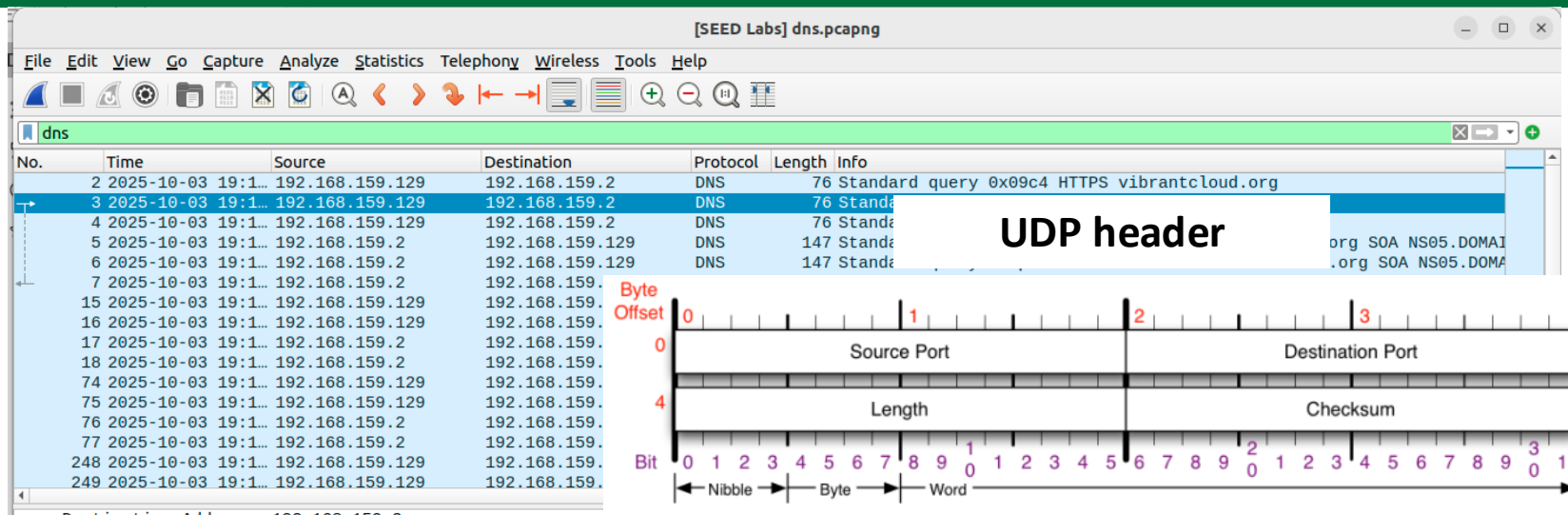

[SEED Labs] dns.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
No. Time Source Destination Protocol Length Info
2 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 76 Standard query 0x9904 HTTPS vibrantcloud.org
3 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 76 Standard query 0x9994 A vibrantcloud.org
4 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 76 S
5 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 147 S
6 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 147 S
7 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 92 S
15 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 74 S
16 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 74 S
17 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 102 S
18 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 90 S
74 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 86 S
75 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 86 S
76 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 114 S
77 2025-10-03 19:1… 192.168.159.2 192.168.159.129 DNS 102 S
248 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 85 S
249 2025-10-03 19:1… 192.168.159.129 192.168.159.2 DNS 85 S
Checksum: 0xc010 [unverified]
[Checksum Status: Unverified]
[Stream index: 2]
[Timestamps]
UDP payload (34 bytes)
Domain Name System (query)
Transaction ID: 0x9994
Flags: 0x0100 Standard query
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Queries
vibrantcloud.org: type A, class IN
Name: vibrantcloud.org
[Name Length: 16]
[Label Count: 2]
Type: A (Host Address) (1)
Class: IN (0x0001)
[Response In: 7]
0000 00 50 56 f5 9e 21 00 0c 29 a8 d0 96 08 00 45 00 ·PV··!··)·····E·
0010 00 3e 3d 98 00 00 40 11 7d 42 c0 a8 9f 81 c0 a8 ·>=···@· }B····
0020 9f 02 c9 ca 00 35 00 2a c0 1e 99 94 01 00 00 01 ·····5·* ····
0030 00 00 00 00 00 00 0c 76 69 62 72 61 6e 74 63 6c ·····v ibrantcl
0040 6f 75 64 03 6f 72 67 00 00 01 00 01 oud·org· ····
Class
0x001 = IN (Internet)
DNS request format
2 bytes
2 bytes
ID=0x9994
Flags=0x0100
# questions=1
# answers
# authority
# additional
questions=vibrantcloud.org: type A
ASCII codes
v = 0x76
i = 0x69
b = 0x62
r = 0x72
…
Type
0x001 = A
0x00F = MX
…
App
Transport
Network
Link
Physical

DNS Layer 4: UDP on port 53



App
Transport
Network
Link
Physical

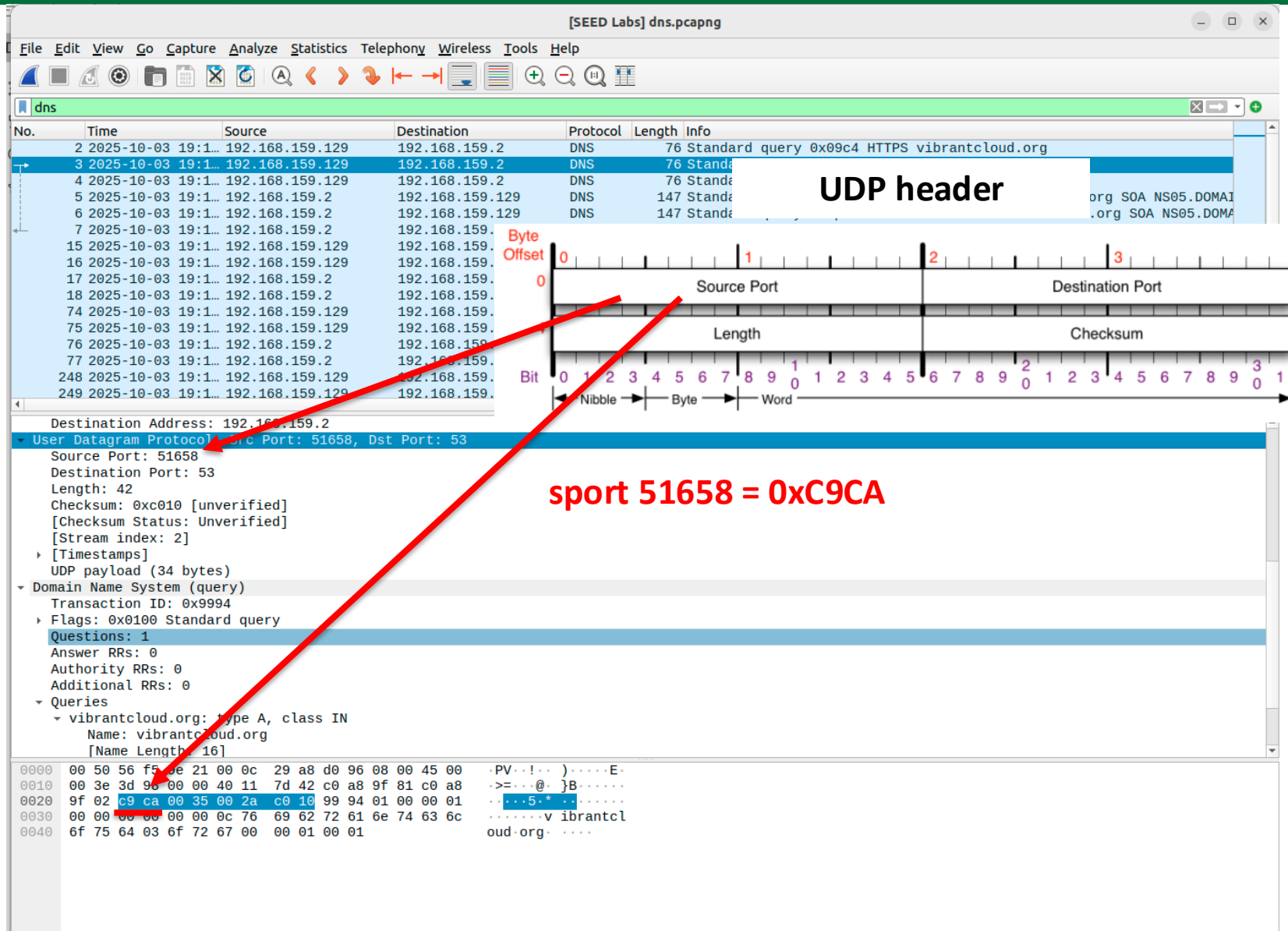
DNS Layer 4: UDP on port 53



App
Transport
Network
Link
Physical

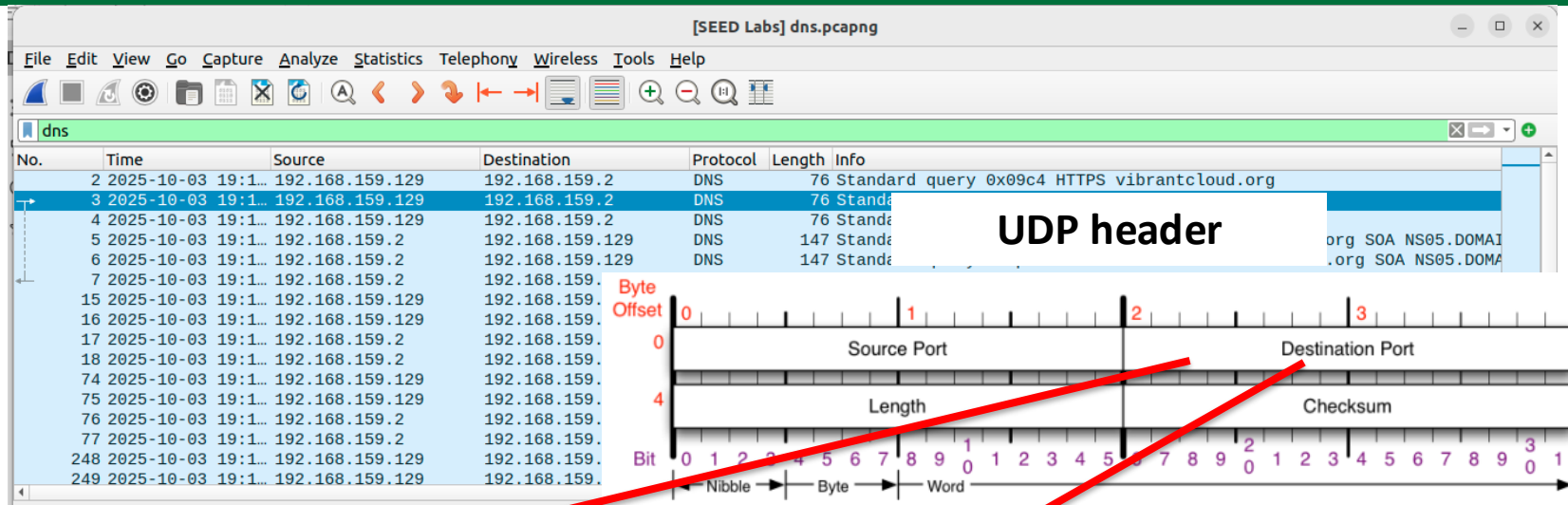
UDP header is always 8 bytes
TCP is variable length

DNS Layer 4: UDP on port 53



App
Transport
Network
Link
Physical

DNS Layer 4: UDP on port 53



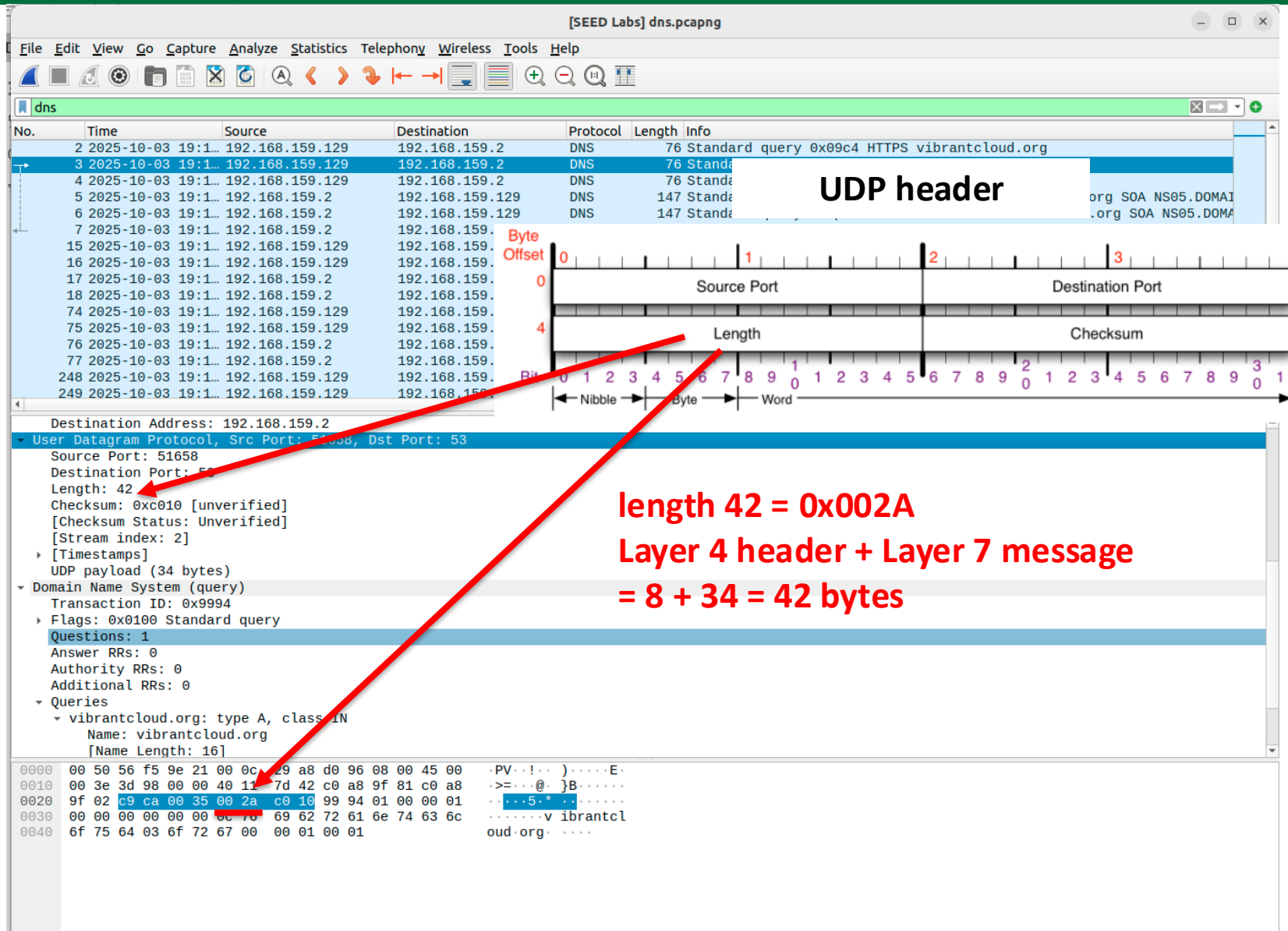
App
Transport
Network
Link
Physical

dport 53 = 0x0035
Recall DNS operates on port 53

How did the receiver know this is a DNS query at Layer 7?

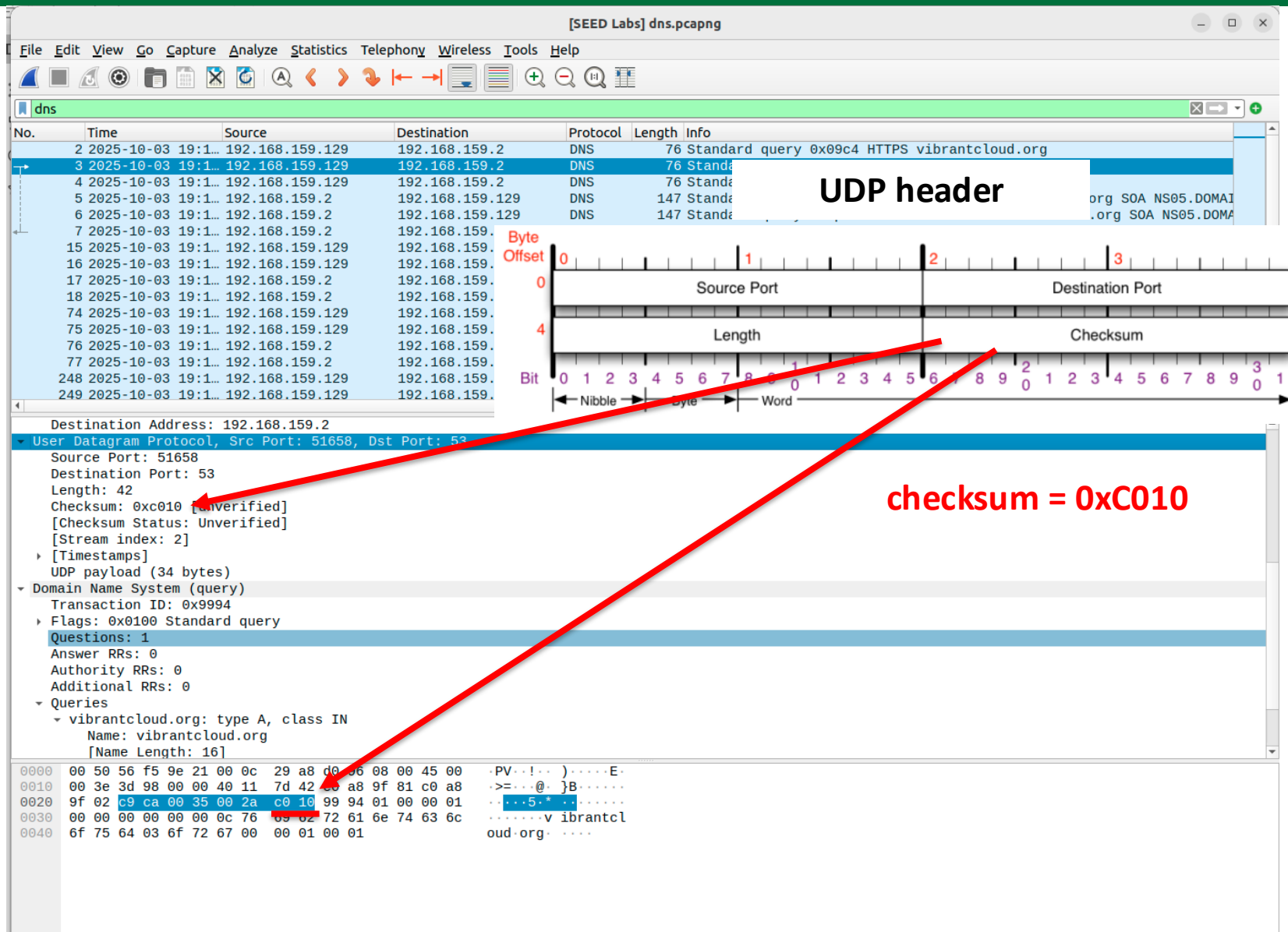
When decoding Layer 4, it saw the datagram was sent to dport 53, so demultiplexed to that port (where DNS listens)

DNS Layer 4: UDP on port 53



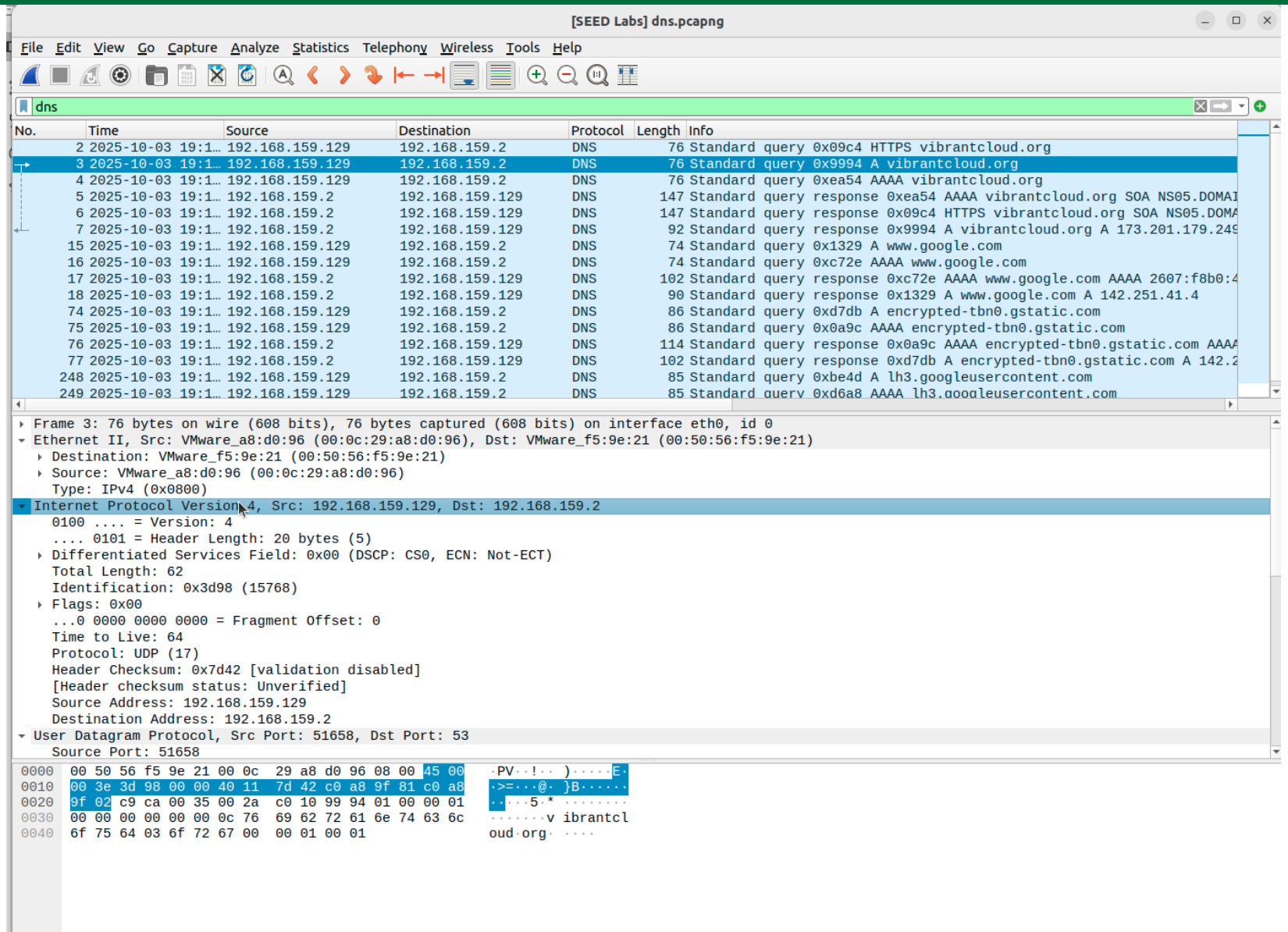
App
Transport
Network
Link
Physical

DNS Layer 4: UDP on port 53



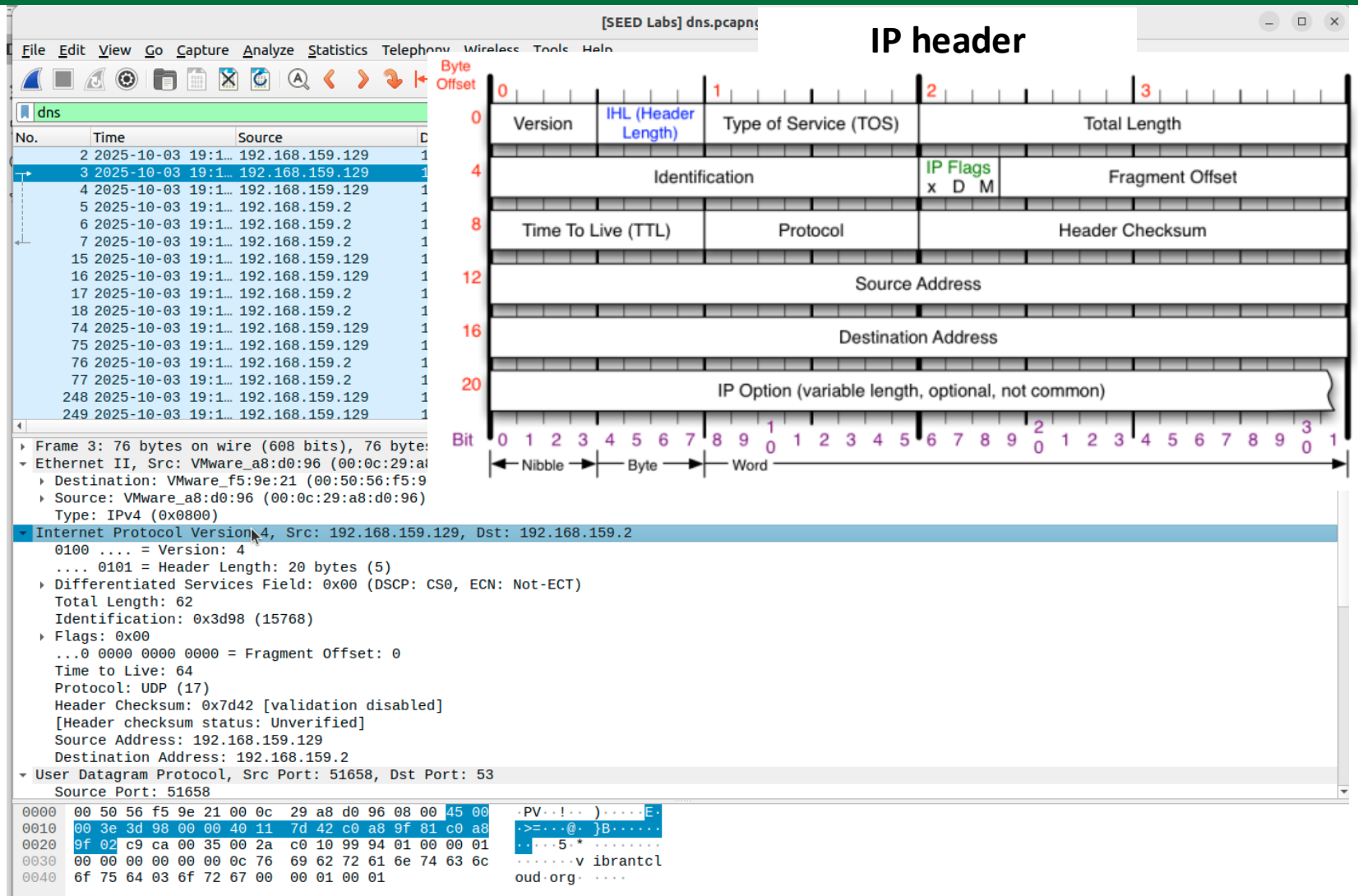
App
Transport
Network
Link
Physical

DNS Layer 3

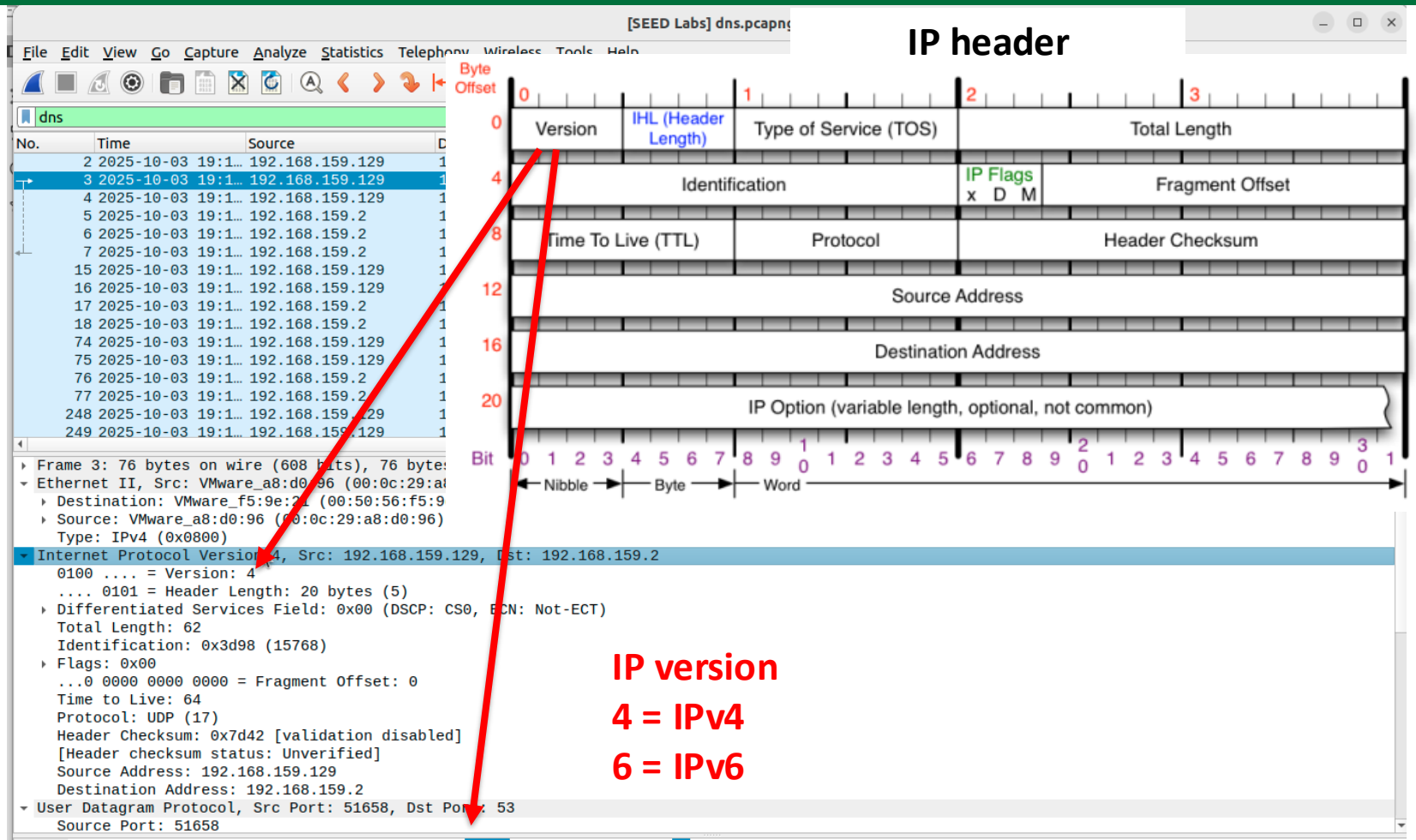


App
Transport
Network
Link
Physical

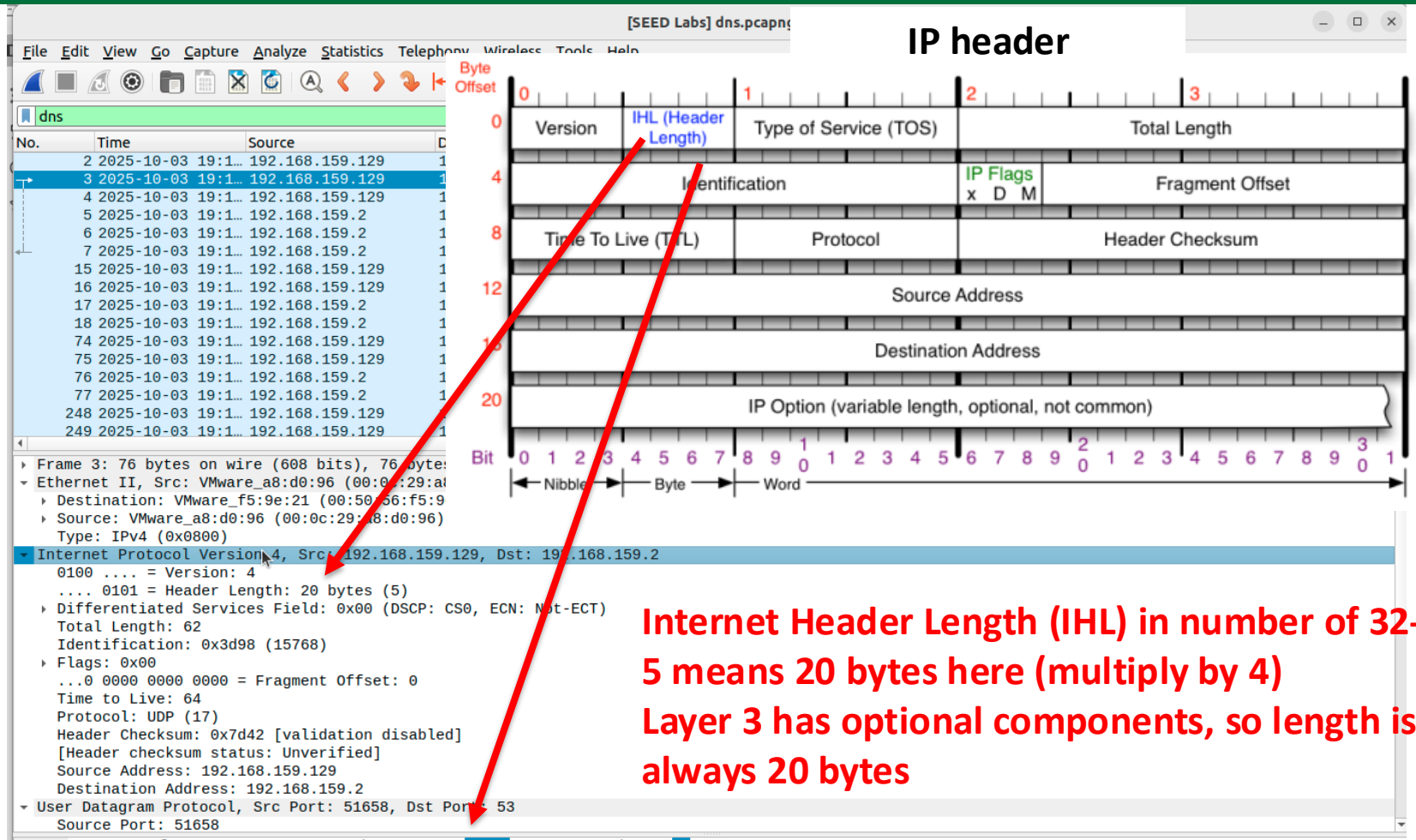
DNS Layer 3


[SEED Labs] dns.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
No. Time Source
2 2025-10-03 19:1… 192.168.159.129
3 2025-10-03 19:1… 192.168.159.129
4 2025-10-03 19:1… 192.168.159.129
5 2025-10-03 19:1… 192.168.159.2
6 2025-10-03 19:1… 192.168.159.2
7 2025-10-03 19:1… 192.168.159.2
15 2025-10-03 19:1… 192.168.159.129
16 2025-10-03 19:1… 192.168.159.129
17 2025-10-03 19:1… 192.168.159.2
18 2025-10-03 19:1… 192.168.159.2
74 2025-10-03 19:1… 192.168.159.129
75 2025-10-03 19:1… 192.168.159.129
76 2025-10-03 19:1… 192.168.159.2
77 2025-10-03 19:1… 192.168.159.2
248 2025-10-03 19:1… 192.168.159.129
249 2025-10-03 19:1… 192.168.159.129
Frame 3: 76 bytes on wire (608 bits), 76 bytes
Ethernet II, Src: VMware_a8:d0:96 (00:0c:29:a8
Destination: VMware_f5:9e:21 (00:50:56:f5:9
Source: VMware_a8:d0:96 (00:0c:29:a8:d0:96)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.159.129, Dst: 192.168.159.2
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 62
Identification: 0x3d98 (15768)
Flags: 0x00
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 64
Protocol: UDP (17)
Header Checksum: 0x7d42 [validation disabled]
[Header checksum status: Unverified]
Source Address: 192.168.159.129
Destination Address: 192.168.159.2
User Datagram Protocol, Src Port: 51658, Dst Port: 53
Source Port: 51658
0000 00 50 56 f5 9e 21 00 0c 29 a8 d0 96 08 00 45 00
0010 00 3e 3d 98 00 00 40 11 7d 42 c0 a8 9f 81 c0 a8
0020 9f 02 c9 ca 00 35 00 2a c0 10 99 94 01 00 00 01
0030 00 00 00 00 00 00 0c 76 69 62 72 61 6e 74 63 6c
0040 6f 75 64 03 6f 72 67 00 00 01 00 01
IP header
Byte Offset
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags x D M
Fragment Offset
Time To Live (TTL)
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
Bit
Nibble
Byte
Word
20 Bytes
IHL (Internet Header Length)
App
Transport
Network
Link
Physical

DNS Layer 3

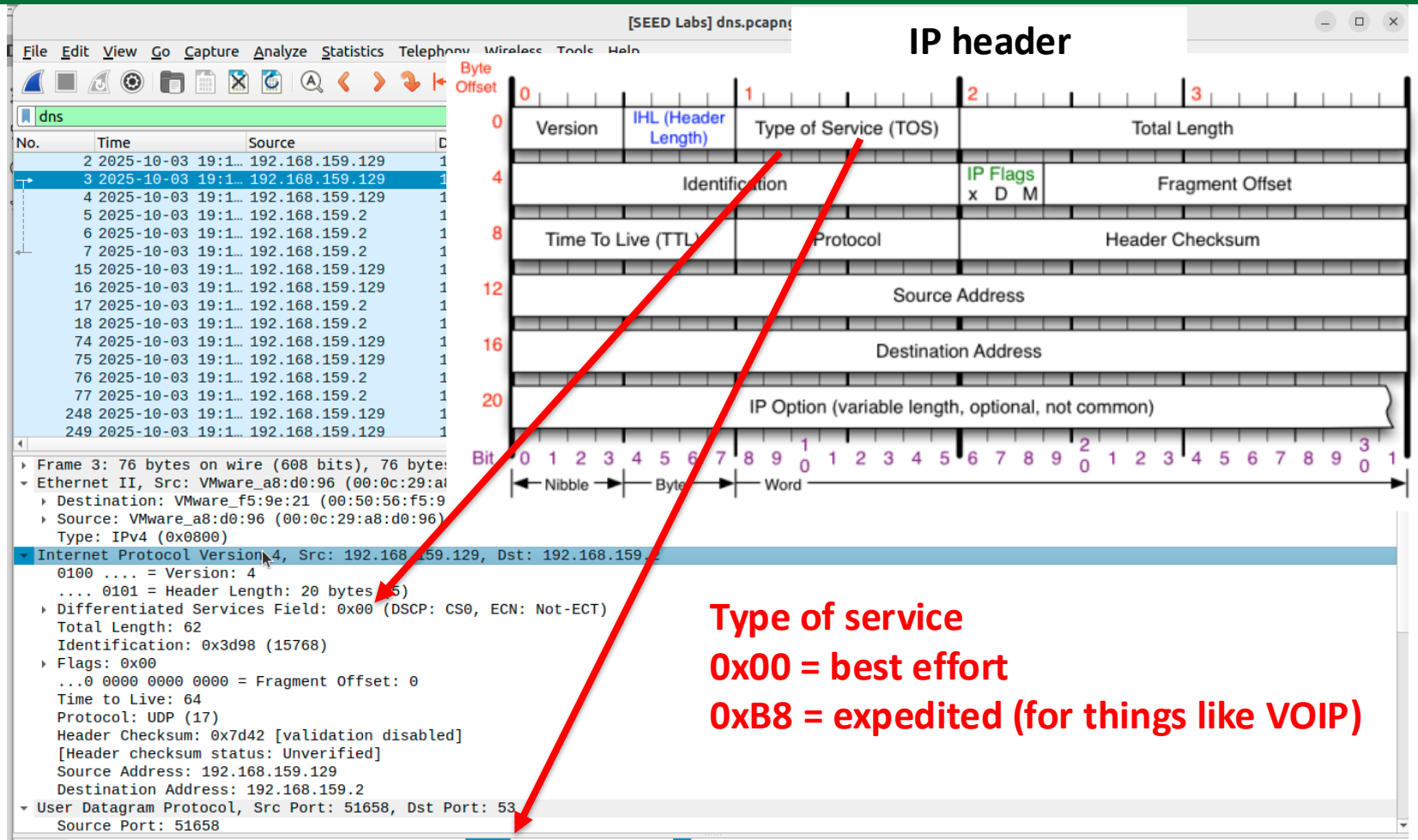

[SEED Labs] dns.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
No. Time Source
2 2025-10-03 19:1… 192.168.159.129
3 2025-10-03 19:1… 192.168.159.129
4 2025-10-03 19:1… 192.168.159.129
5 2025-10-03 19:1… 192.168.159.2
6 2025-10-03 19:1… 192.168.159.2
7 2025-10-03 19:1… 192.168.159.2
15 2025-10-03 19:1… 192.168.159.129
16 2025-10-03 19:1… 192.168.159.129
17 2025-10-03 19:1… 192.168.159.2
18 2025-10-03 19:1… 192.168.159.2
74 2025-10-03 19:1… 192.168.159.129
75 2025-10-03 19:1… 192.168.159.129
76 2025-10-03 19:1… 192.168.159.2
77 2025-10-03 19:1… 192.168.159.2
248 2025-10-03 19:1… 192.168.159.129
249 2025-10-03 19:1… 192.168.159.129
Frame 3: 76 bytes on wire (608 bits), 76 bytes
Ethernet II, Src: VMware_a8:d0:96 (00:0c:29:a8:d0:96)
Destination: VMware_f5:9e:21 (00:50:56:f5:9e:21)
Source: VMware_a8:d0:96 (00:0c:29:a8:d0:96)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.159.129, Dst: 192.168.159.2
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 62
Identification: 0x3d98 (15768)
Flags: 0x00
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 64
Protocol: UDP (17)
Header Checksum: 0x7d42 [validation disabled]
[Header checksum status: Unverified]
Source Address: 192.168.159.129
Destination Address: 192.168.159.2
User Datagram Protocol, Src Port: 51658, Dst Port: 53
Source Port: 51658
0000 00 50 56 f5 9e 21 00 0c 29 a8 d0 96 08 00 45 00
0010 00 3e 3d 98 00 00 40 11 7d 42 c0 a8 9f 81 c0 a8
0020 9f 02 c9 ca 00 35 00 2a c0 10 99 94 01 00 00 01
0030 00 00 00 00 00 00 0c 76 69 62 72 61 6e 74 63 6c
0040 6f 75 64 03 6f 72 67 00 00 01 00 01
IP header
Byte Offset
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags x D M
Fragment Offset
Time To Live (TTL)
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
Bit
Nibble
Byte
Word
20 Bytes
IHL (Internet Header Length)
App
Transport
Network
Link
Physical
IP version
4 = IPv4
6 = IPv6

DNS Layer 3

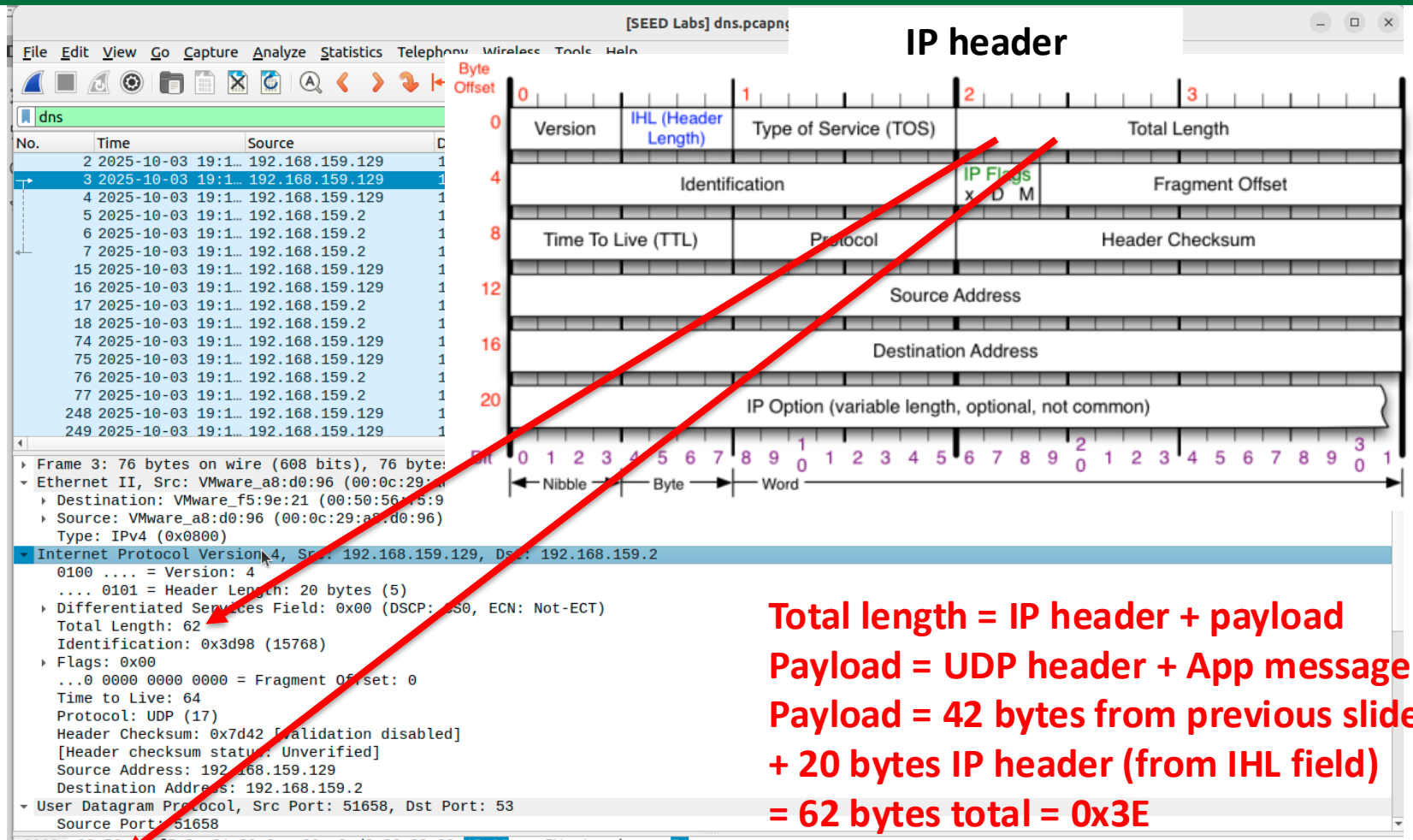


Internet Header Length (IHL) in number of 32-bit words
5 means 20 bytes here (multiply by 4)
Layer 3 has optional components, so length is not always 20 bytes

DNS Layer 3

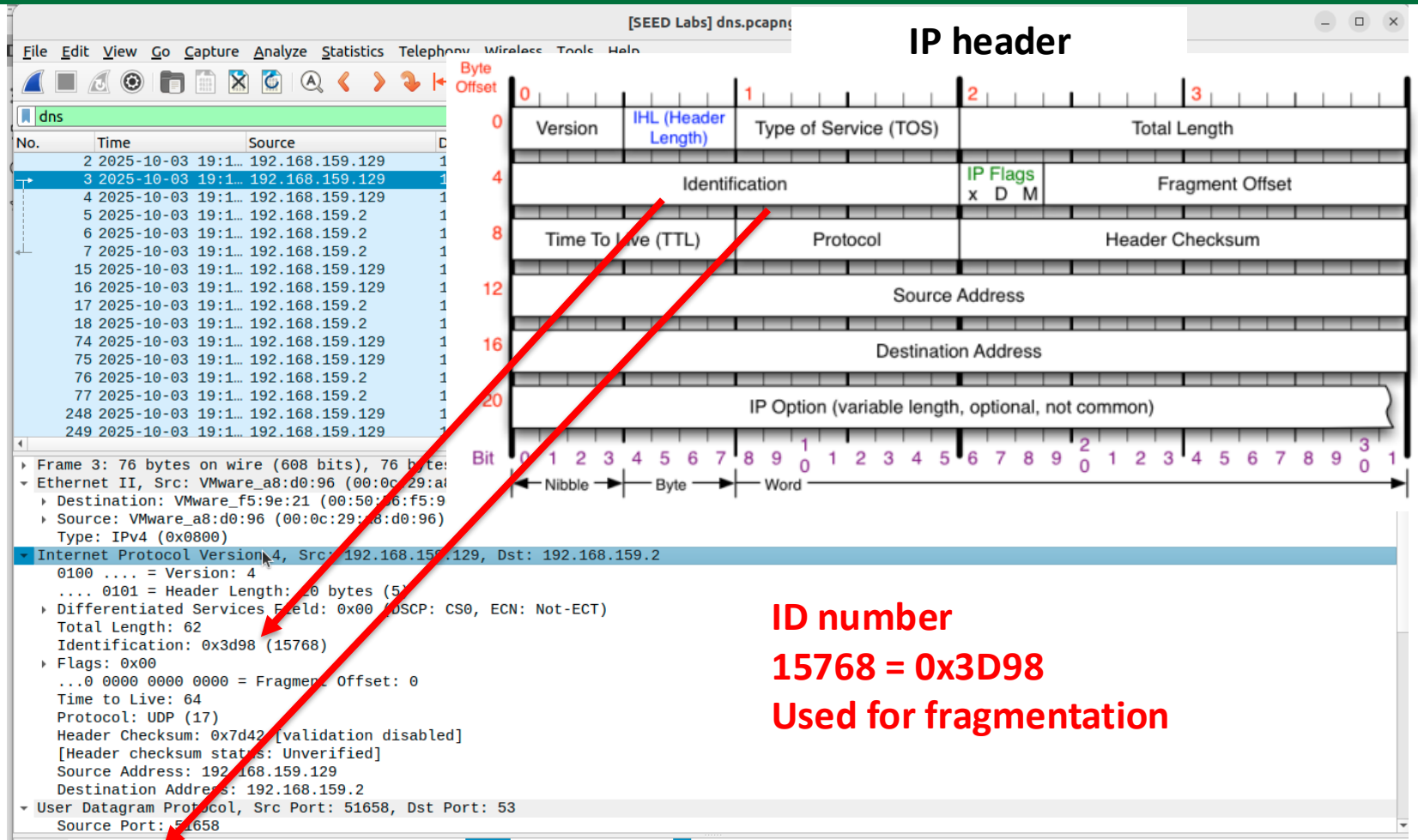

[SEED Labs] dns.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
No. Time Source
2 2025-10-03 19:1… 192.168.159.129
3 2025-10-03 19:1… 192.168.159.129
4 2025-10-03 19:1… 192.168.159.129
5 2025-10-03 19:1… 192.168.159.2
6 2025-10-03 19:1… 192.168.159.2
7 2025-10-03 19:1… 192.168.159.2
15 2025-10-03 19:1… 192.168.159.129
16 2025-10-03 19:1… 192.168.159.129
17 2025-10-03 19:1… 192.168.159.2
18 2025-10-03 19:1… 192.168.159.2
74 2025-10-03 19:1… 192.168.159.129
75 2025-10-03 19:1… 192.168.159.129
76 2025-10-03 19:1… 192.168.159.2
77 2025-10-03 19:1… 192.168.159.2
248 2025-10-03 19:1… 192.168.159.129
249 2025-10-03 19:1… 192.168.159.129
Frame 3: 76 bytes on wire (608 bits), 76 bytes
Ethernet II, Src: VMware_a8:d0:96 (00:0c:29:a8
Destination: VMware_f5:9e:21 (00:50:56:f5:9
Source: VMware_a8:d0:96 (00:0c:29:a8:d0:96)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.159.129, Dst: 192.168.159.2
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 62
Identification: 0x3d98 (15768)
Flags: 0x00
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 64
Protocol: UDP (17)
Header Checksum: 0x7d42 [validation disabled]
[Header checksum status: Unverified]
Source Address: 192.168.159.129
Destination Address: 192.168.159.2
User Datagram Protocol, Src Port: 51658, Dst Port: 53
Source Port: 51658
0000 00 50 56 f5 9e 21 00 0c 29 a8 d0 96 08 00 45 00
0010 00 3e 3d 98 00 00 40 11 7d 42 c0 a8 9f 81 c0 a8
0020 9f 02 c9 ca 00 35 00 2a c0 10 99 94 01 00 00 01
0030 00 00 00 00 00 00 0c 76 69 62 72 61 6e 74 63 6c
0040 6f 75 64 03 6f 72 67 00 00 01 00 01
IP header
Byte Offset
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags x D M
Fragment Offset
Time To Live (TTL)
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
Bit
Nibble
Byte
Word
20 Bytes
IHL (Internet Header Length)
App
Transport
Network
Link
Physical
Type of service
0x00 = best effort
0xB8 = expedited (for things like VOIP)

DNS Layer 3

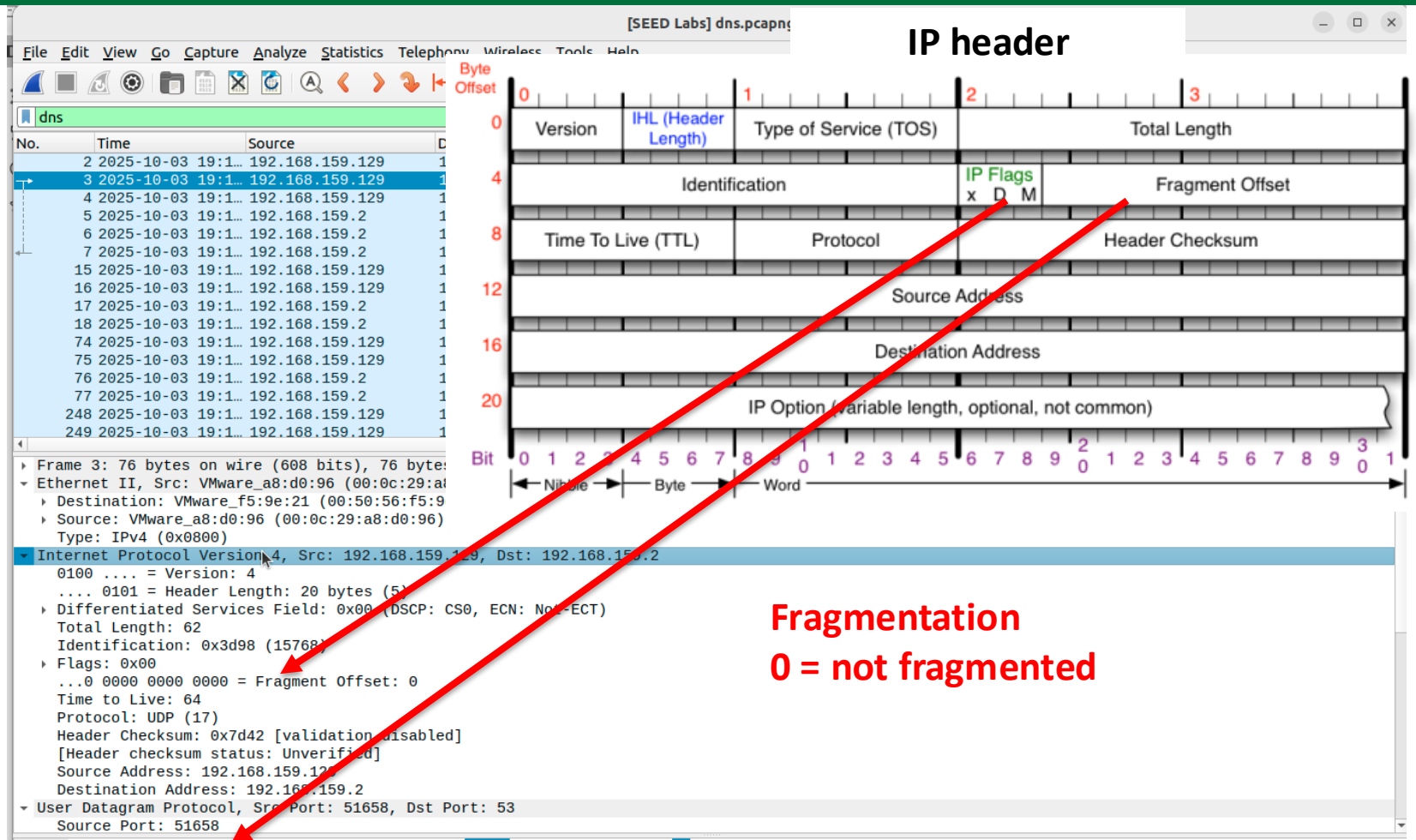


Total length = IP header + payload
Payload = UDP header + App message
Payload = 42 bytes from previous slide
+ 20 bytes IP header (from IHL field)
= 62 bytes total = 0x3E

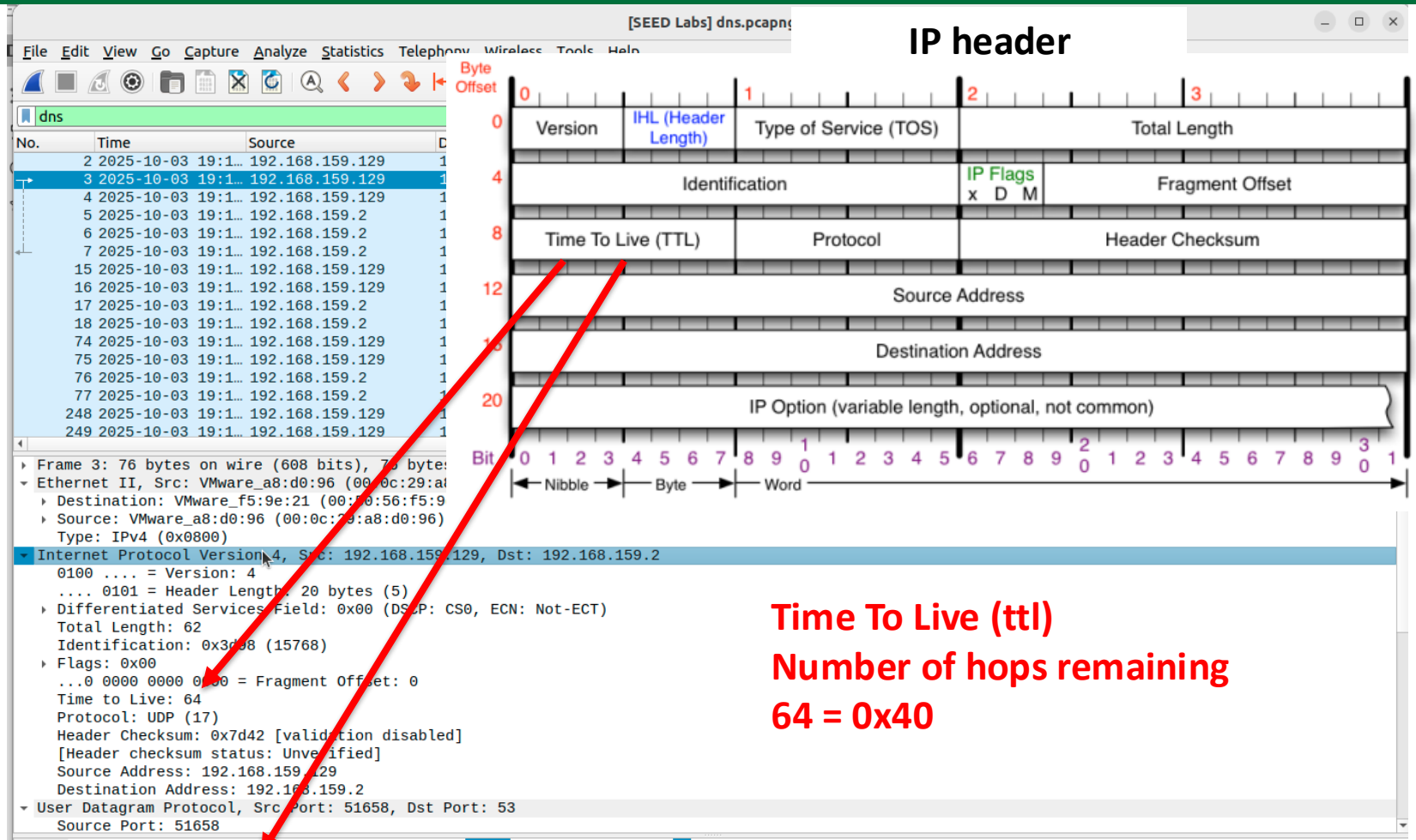
DNS Layer 3


[SEED Labs] dns.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
No. Time Source
2 2025-10-03 19:1… 192.168.159.129
3 2025-10-03 19:1… 192.168.159.129
4 2025-10-03 19:1… 192.168.159.129
5 2025-10-03 19:1… 192.168.159.2
6 2025-10-03 19:1… 192.168.159.2
7 2025-10-03 19:1… 192.168.159.2
15 2025-10-03 19:1… 192.168.159.129
16 2025-10-03 19:1… 192.168.159.129
17 2025-10-03 19:1… 192.168.159.2
18 2025-10-03 19:1… 192.168.159.2
74 2025-10-03 19:1… 192.168.159.129
75 2025-10-03 19:1… 192.168.159.129
76 2025-10-03 19:1… 192.168.159.2
77 2025-10-03 19:1… 192.168.159.2
248 2025-10-03 19:1… 192.168.159.129
249 2025-10-03 19:1… 192.168.159.129
Frame 3: 76 bytes on wire (608 bits), 76 bytes
Ethernet II, Src: VMware_a8:d0:96 (00:0c:29:a8:d0:96)
Destination: VMware_f5:9e:21 (00:50:56:f5:9e:21)
Source: VMware_a8:d0:96 (00:0c:29:a8:d0:96)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.159.129, Dst: 192.168.159.2
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 62
Identification: 0x3d98 (15768)
Flags: 0x00
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 64
Protocol: UDP (17)
Header Checksum: 0x7d42 [validation disabled]
[Header checksum status: Unverified]
Source Address: 192.168.159.129
Destination Address: 192.168.159.2
User Datagram Protocol, Src Port: 51658, Dst Port: 53
Source Port: 51658
0000 00 50 56 f5 9e 21 00 0c 29 a8 d0 96 08 00 45 00
0010 00 3e 3d 98 00 00 40 11 7d 42 c0 a8 9f 81 c0 a8
0020 9f 02 c9 ca 00 35 00 2a c0 10 99 94 01 00 00 01
0030 00 00 00 00 00 00 0c 76 69 62 72 61 6e 74 63 6c
0040 6f 75 64 03 6f 72 67 00 00 01 00 01
IP header
Byte Offset
0 1 2 3
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags x D M
Fragment Offset
Time To Live (TTL)
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
Bit
Nibble
Byte
Word
20 Bytes
IHL (Internet Header Length)
App
Transport
Network
Link
Physical
ID number
15768 = 0x3D98
Used for fragmentation

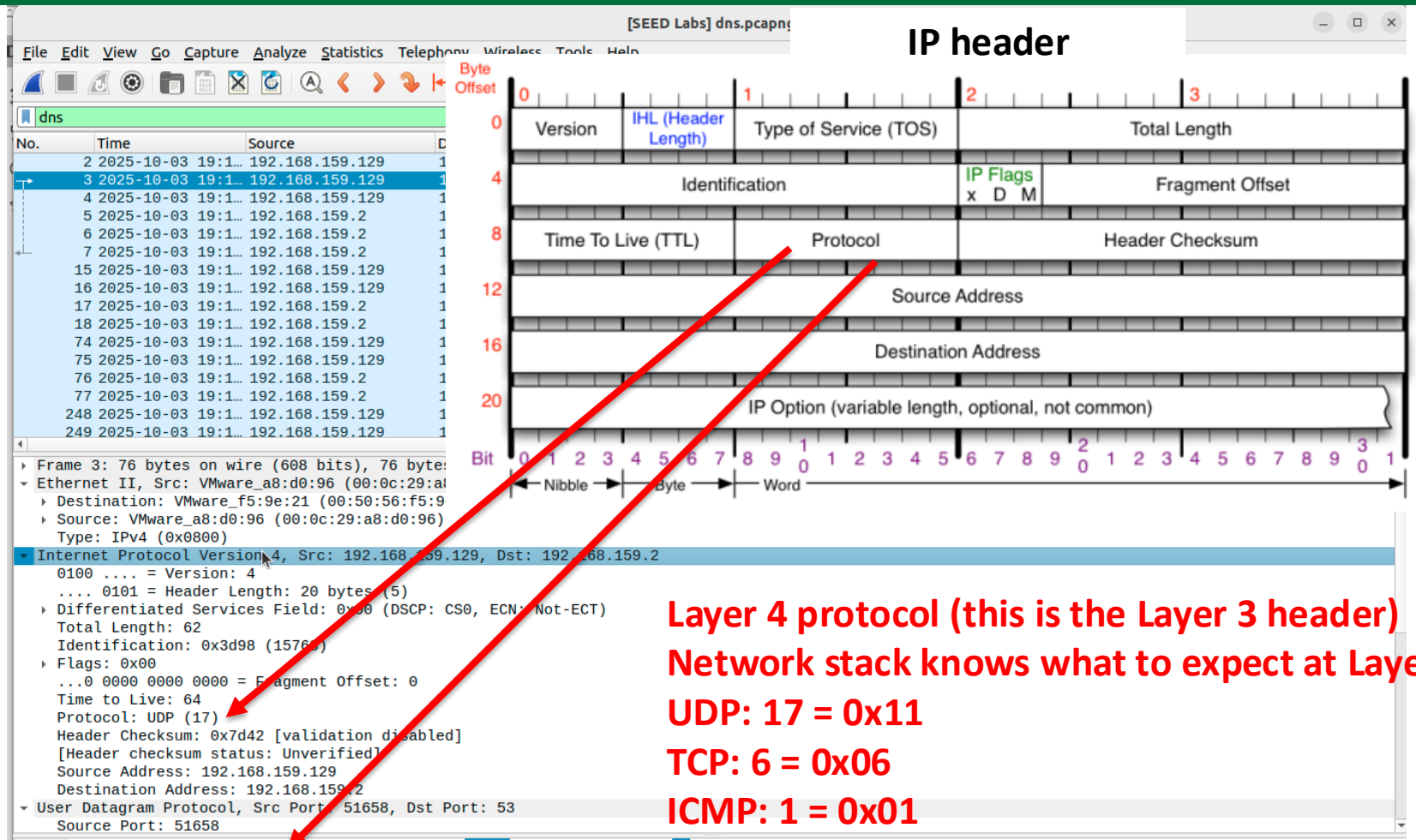
DNS Layer 3


[SEED Labs] dns.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
No. Time Source
2 2025-10-03 19:1… 192.168.159.129
3 2025-10-03 19:1… 192.168.159.129
4 2025-10-03 19:1… 192.168.159.129
5 2025-10-03 19:1… 192.168.159.2
6 2025-10-03 19:1… 192.168.159.2
7 2025-10-03 19:1… 192.168.159.2
15 2025-10-03 19:1… 192.168.159.129
16 2025-10-03 19:1… 192.168.159.129
17 2025-10-03 19:1… 192.168.159.2
18 2025-10-03 19:1… 192.168.159.2
74 2025-10-03 19:1… 192.168.159.129
75 2025-10-03 19:1… 192.168.159.129
76 2025-10-03 19:1… 192.168.159.2
77 2025-10-03 19:1… 192.168.159.2
248 2025-10-03 19:1… 192.168.159.129
249 2025-10-03 19:1… 192.168.159.129
Frame 3: 76 bytes on wire (608 bits), 76 bytes
Ethernet II, Src: VMware_a8:d0:96 (00:0c:29:a8
Destination: VMware_f5:9e:21 (00:50:56:f5:9
Source: VMware_a8:d0:96 (00:0c:29:a8:d0:96)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.159.129, Dst: 192.168.159.2
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 62
Identification: 0x3d98 (15768)
Flags: 0x00
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 64
Protocol: UDP (17)
Header Checksum: 0x7d42 [validation disabled]
[Header checksum status: Unverified]
Source Address: 192.168.159.129
Destination Address: 192.168.159.2
User Datagram Protocol, Src Port: 51658, Dst Port: 53
Source Port: 51658
0000 00 50 56 f5 9e 21 00 0c 29 a8 d0 96 08 00 45 00
0010 00 3e 3d 98 00 00 40 11 7d 42 c0 a8 9f 81 c0 a8
0020 9f 02 c9 ca 00 35 00 2a c0 10 99 94 01 00 00 01
0030 00 00 00 00 00 00 0c 76 69 62 72 61 6e 74 63 6c
0040 6f 75 64 03 6f 72 67 00 00 01 00 01
IP header
Byte Offset
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags x D M
Fragment Offset
Time To Live (TTL)
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
Bit
Nibble
Byte
Word
20 Bytes
IHL (Internet Header Length)
App
Transport
Network
Link
Physical
Fragmentation
0 = not fragmented

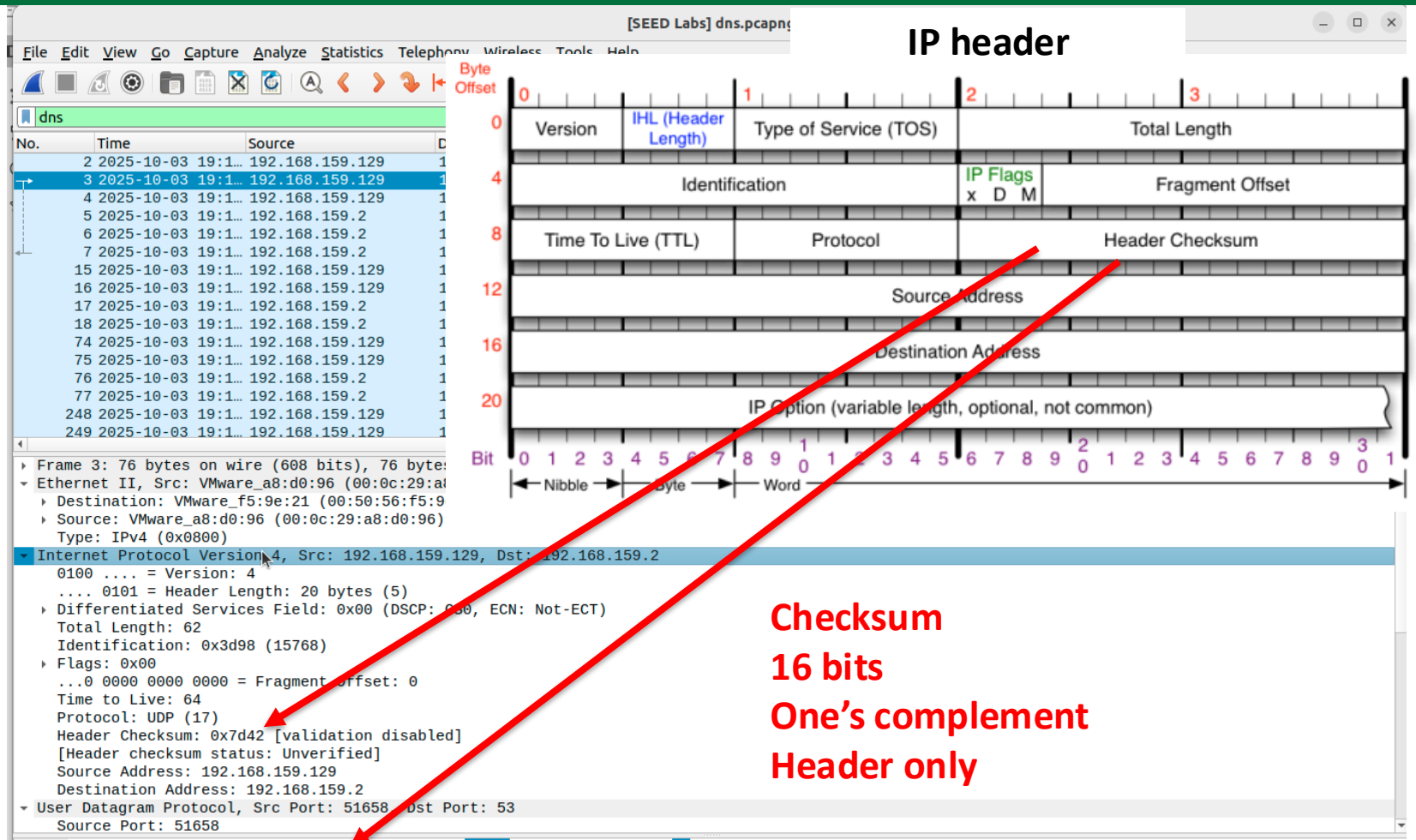
DNS Layer 3


[SEED Labs] dns.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
No. Time Source
2 2025-10-03 19:1… 192.168.159.129
3 2025-10-03 19:1… 192.168.159.129
4 2025-10-03 19:1… 192.168.159.129
5 2025-10-03 19:1… 192.168.159.2
6 2025-10-03 19:1… 192.168.159.2
7 2025-10-03 19:1… 192.168.159.2
15 2025-10-03 19:1… 192.168.159.129
16 2025-10-03 19:1… 192.168.159.129
17 2025-10-03 19:1… 192.168.159.2
18 2025-10-03 19:1… 192.168.159.2
74 2025-10-03 19:1… 192.168.159.129
75 2025-10-03 19:1… 192.168.159.129
76 2025-10-03 19:1… 192.168.159.2
77 2025-10-03 19:1… 192.168.159.2
248 2025-10-03 19:1… 192.168.159.129
249 2025-10-03 19:1… 192.168.159.129
Frame 3: 76 bytes on wire (608 bits), 76 bytes
Ethernet II, Src: VMware_a8:d0:96 (00:0c:29:a8:d0:96)
Destination: VMware_f5:9e:21 (00:50:56:f5:9e:21)
Source: VMware_a8:d0:96 (00:0c:29:a8:d0:96)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.159.129, Dst: 192.168.159.2
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 62
Identification: 0x3d98 (15768)
Flags: 0x00
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 64
Protocol: UDP (17)
Header Checksum: 0x7d42 [validation disabled]
[Header checksum status: Unverified]
Source Address: 192.168.159.129
Destination Address: 192.168.159.2
User Datagram Protocol, Src Port: 51658, Dst Port: 53
Source Port: 51658
0000 00 50 56 f5 9e 21 00 0c 29 a8 d0 96 08 00 45 00
0010 00 3e 3d 98 00 00 40 11 7d 42 c0 a8 9f 81 c0 a8
0020 9f 02 c9 ca 00 35 00 2a c0 10 99 94 01 00 00 01
0030 00 00 00 00 00 00 0c 76 69 62 72 61 6e 74 63 6c
0040 6f 75 64 03 6f 72 67 00 00 01 00 01
IP header
Byte Offset
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags x D M
Fragment Offset
Time To Live (TTL)
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
Bit
Nibble
Byte
Word
20 Bytes
IHL (Internet Header Length)
App
Transport
Network
Link
Physical
Time To Live (ttl)
Number of hops remaining
64 = 0x40

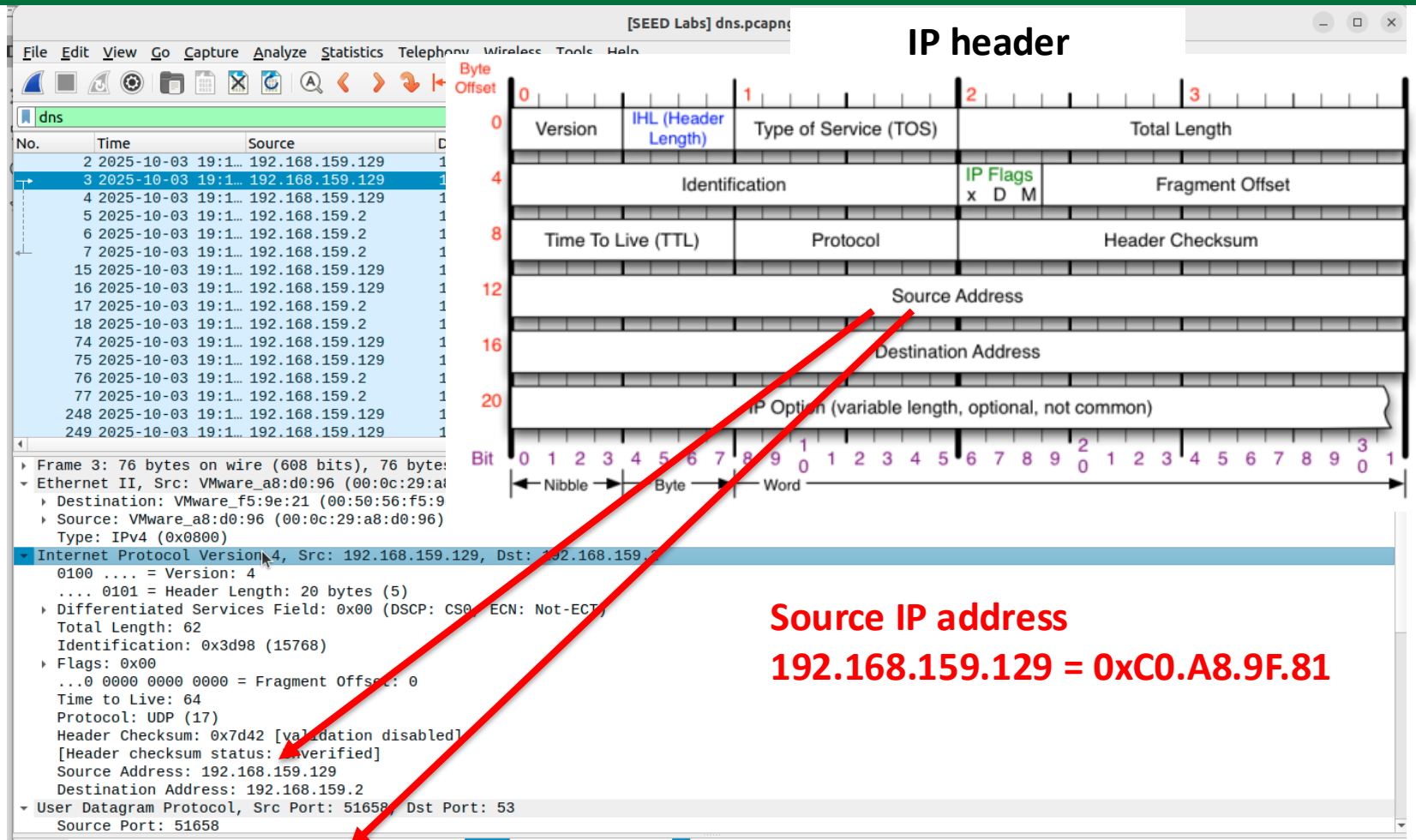
DNS Layer 3


[SEED Labs] dns.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
No. Time Source
2 2025-10-03 19:1… 192.168.159.129
3 2025-10-03 19:1… 192.168.159.129
4 2025-10-03 19:1… 192.168.159.129
5 2025-10-03 19:1… 192.168.159.2
6 2025-10-03 19:1… 192.168.159.2
7 2025-10-03 19:1… 192.168.159.2
15 2025-10-03 19:1… 192.168.159.129
16 2025-10-03 19:1… 192.168.159.129
17 2025-10-03 19:1… 192.168.159.2
18 2025-10-03 19:1… 192.168.159.2
74 2025-10-03 19:1… 192.168.159.129
75 2025-10-03 19:1… 192.168.159.129
76 2025-10-03 19:1… 192.168.159.2
77 2025-10-03 19:1… 192.168.159.2
248 2025-10-03 19:1… 192.168.159.129
249 2025-10-03 19:1… 192.168.159.129
Frame 3: 76 bytes on wire (608 bits), 76 bytes
Ethernet II, Src: VMware_a8:d0:96 (00:0c:29:a8:d0:96)
Destination: VMware_f5:9e:21 (00:50:56:f5:9
Source: VMware_a8:d0:96 (00:0c:29:a8:d0:96)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.159.129, Dst: 192.168.159.2
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 62
Identification: 0x3d98 (15768)
Flags: 0x00
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 64
Protocol: UDP (17)
Header Checksum: 0x7d42 [validation disabled]
[Header checksum status: Unverified]
Source Address: 192.168.159.129
Destination Address: 192.168.159.2
User Datagram Protocol, Src Port: 51658, Dst Port: 53
Source Port: 51658
0000 00 50 56 f5 9e 21 00 0c 29 a8 d0 96 08 00 45 00
0010 00 3e 3d 98 00 00 40 11 7d 42 c0 a8 9f 81 c0 a8
0020 9f 02 c9 ca 00 35 00 2a c0 10 99 94 01 00 00 01
0030 00 00 00 00 00 00 0c 76 69 62 72 61 6e 74 63 6c
0040 6f 75 64 03 6f 72 67 00 00 01 00 01
IP header
Byte Offset
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags x D M
Fragment Offset
Time To Live (TTL)
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
Bit
Nibble
Byte
Word
20 Bytes
IHL (Internet Header Length)
App
Transport
Network
Link
Physical
Layer 4 protocol (this is the Layer 3 header)
Network stack knows what to expect at Layer 4
UDP: 17 = 0x11
TCP: 6 = 0x06
ICMP: 1 = 0x01

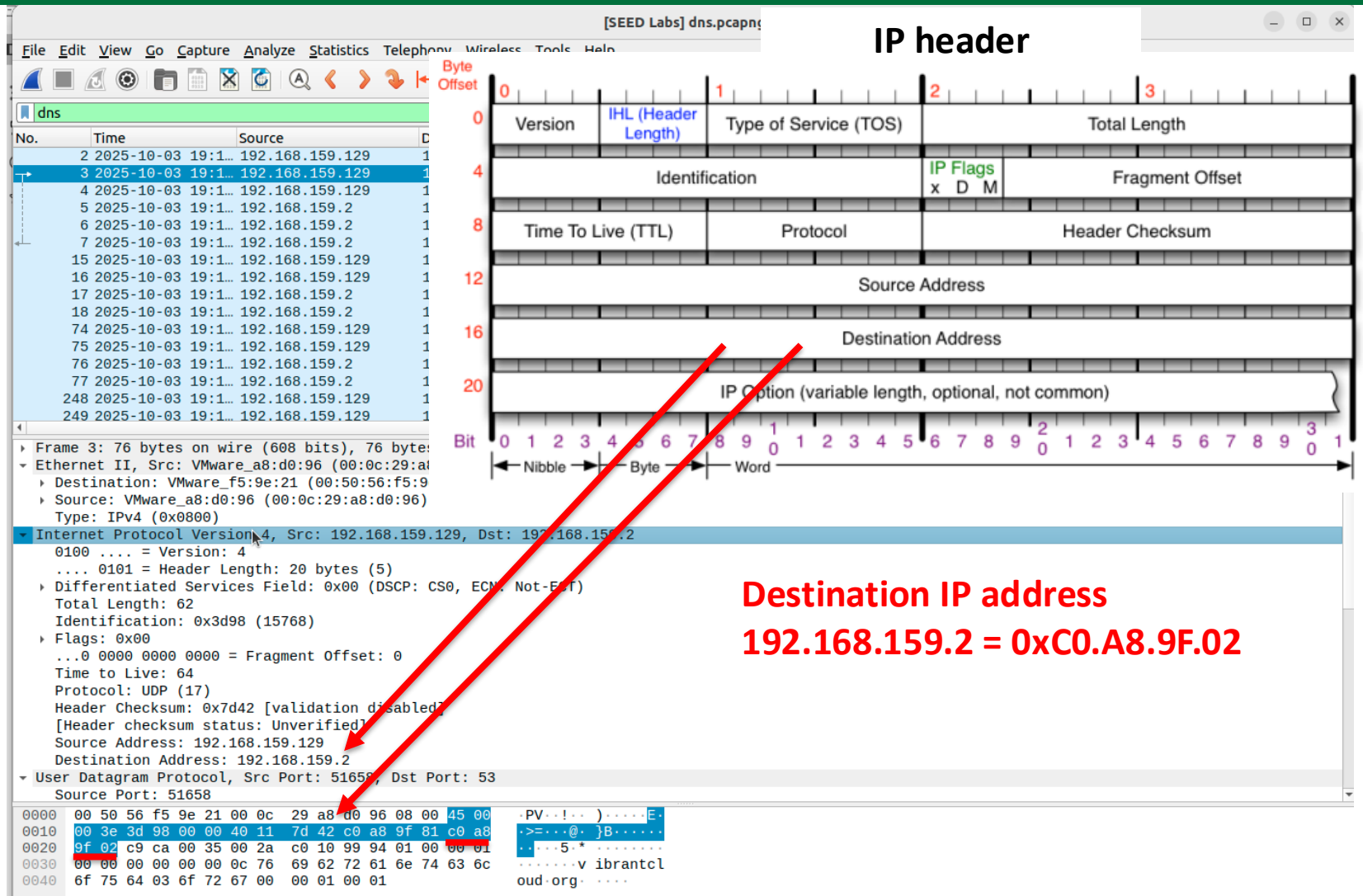
DNS Layer 3


[SEED Labs] dns.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
No. Time Source
2 2025-10-03 19:1… 192.168.159.129
3 2025-10-03 19:1… 192.168.159.129
4 2025-10-03 19:1… 192.168.159.129
5 2025-10-03 19:1… 192.168.159.2
6 2025-10-03 19:1… 192.168.159.2
7 2025-10-03 19:1… 192.168.159.2
15 2025-10-03 19:1… 192.168.159.129
16 2025-10-03 19:1… 192.168.159.129
17 2025-10-03 19:1… 192.168.159.2
18 2025-10-03 19:1… 192.168.159.2
74 2025-10-03 19:1… 192.168.159.129
75 2025-10-03 19:1… 192.168.159.129
76 2025-10-03 19:1… 192.168.159.2
77 2025-10-03 19:1… 192.168.159.2
248 2025-10-03 19:1… 192.168.159.129
249 2025-10-03 19:1… 192.168.159.129
Frame 3: 76 bytes on wire (608 bits), 76 bytes
Ethernet II, Src: VMware_a8:d0:96 (00:0c:29:a8:d0:96)
Destination: VMware_f5:9e:21 (00:50:56:f5:9
Source: VMware_a8:d0:96 (00:0c:29:a8:d0:96)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.159.129, Dst: 192.168.159.2
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 62
Identification: 0x3d98 (15768)
Flags: 0x00
...0 0000 0000 0000 = Fragment offset: 0
Time to Live: 64
Protocol: UDP (17)
Header Checksum: 0x7d42 [validation disabled]
[Header checksum status: Unverified]
Source Address: 192.168.159.129
Destination Address: 192.168.159.2
User Datagram Protocol, Src Port: 51658, Dst Port: 53
Source Port: 51658
0000 00 50 56 f5 9e 21 00 0c 29 a8 d0 96 08 00 45 00
0010 00 3e 3d 98 00 00 40 11 7d 42 c0 a8 9f 81 c0 a8
0020 9f 02 c9 ca 00 35 00 2a c0 10 99 94 01 00 00 01
0030 00 00 00 00 00 00 0c 76 69 62 72 61 6e 74 63 6c
0040 6f 75 64 03 6f 72 67 00 00 01 00 01
IP header
Byte Offset
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags x D M
Fragment Offset
Time To Live (TTL)
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
Bit
Nibble
Byte
Word
20 Bytes
IHL (Internet Header Length)
App
Transport
Network
Link
Physical
Checksum
16 bits
One's complement
Header only

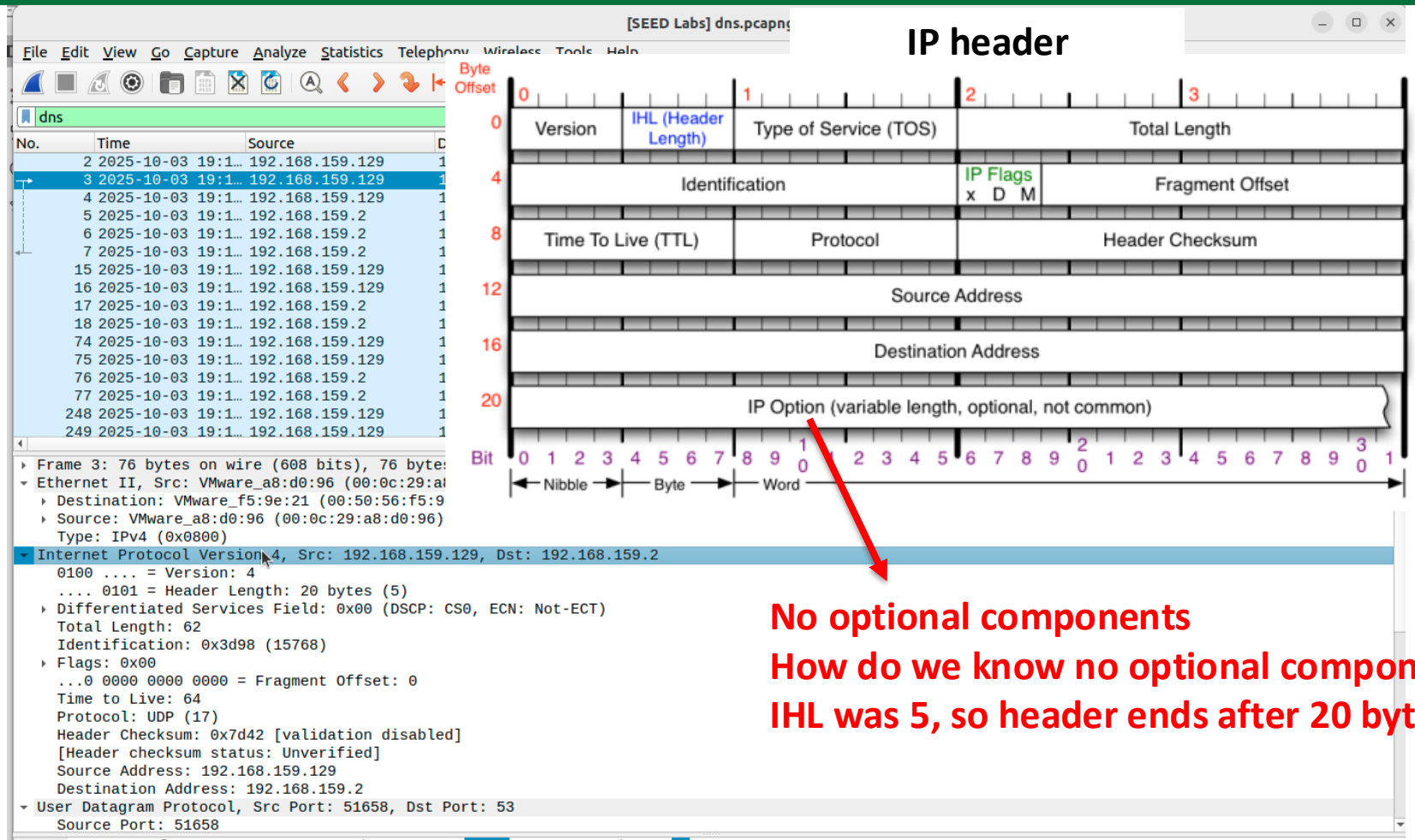
DNS Layer 3


[SEED Labs] dns.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
No. Time Source
2 2025-10-03 19:1… 192.168.159.129
3 2025-10-03 19:1… 192.168.159.129
4 2025-10-03 19:1… 192.168.159.129
5 2025-10-03 19:1… 192.168.159.2
6 2025-10-03 19:1… 192.168.159.2
7 2025-10-03 19:1… 192.168.159.2
15 2025-10-03 19:1… 192.168.159.129
16 2025-10-03 19:1… 192.168.159.129
17 2025-10-03 19:1… 192.168.159.2
18 2025-10-03 19:1… 192.168.159.2
74 2025-10-03 19:1… 192.168.159.129
75 2025-10-03 19:1… 192.168.159.129
76 2025-10-03 19:1… 192.168.159.2
77 2025-10-03 19:1… 192.168.159.2
248 2025-10-03 19:1… 192.168.159.129
249 2025-10-03 19:1… 192.168.159.129
Frame 3: 76 bytes on wire (608 bits), 76 bytes
Ethernet II, Src: VMware_a8:d0:96 (00:0c:29:a8:d0:96)
Destination: VMware_f5:9e:21 (00:50:56:f5:9
Source: VMware_a8:d0:96 (00:0c:29:a8:d0:96)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.159.129, Dst:
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 62
Identification: 0x3d98 (15768)
Flags: 0x00
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 64
Protocol: UDP (17)
Header Checksum: 0x7d42 [validation disabled]
[Header checksum status: Unverified]
Source Address: 192.168.159.129
Destination Address: 192.168.159.2
User Datagram Protocol, Src Port: 51658, Dst Port: 53
Source Port: 51658
0000 00 50 56 f5 9e 21 00 0c 29 a8 d0 96 08 00 45 00
0010 00 3e 3d 98 00 00 40 11 7d 42 c0 a8 9f 81 c0 a8
0020 9f 02 c9 ca 00 35 00 2a c0 10 99 94 01 00 00 01
0030 00 00 00 00 00 00 0c 76 69 62 72 61 6e 74 63 6c
0040 6f 75 64 03 6f 72 67 00 00 01 00 01
IP header
Byte Offset
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags x D M
Fragment Offset
Time To Live (TTL)
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
Bit
Nibble
Byte
Word
20 Bytes
IHL (Internet Header Length)
App
Transport
Network
Link
Physical
Source IP address
192.168.159.129 = 0xC0.A8.9F.81

DNS Layer 3


[SEED Labs] dns.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
No. Time Source
2 2025-10-03 19:1… 192.168.159.129
3 2025-10-03 19:1… 192.168.159.129
4 2025-10-03 19:1… 192.168.159.129
5 2025-10-03 19:1… 192.168.159.2
6 2025-10-03 19:1… 192.168.159.2
7 2025-10-03 19:1… 192.168.159.2
15 2025-10-03 19:1… 192.168.159.129
16 2025-10-03 19:1… 192.168.159.129
17 2025-10-03 19:1… 192.168.159.2
18 2025-10-03 19:1… 192.168.159.2
74 2025-10-03 19:1… 192.168.159.129
75 2025-10-03 19:1… 192.168.159.129
76 2025-10-03 19:1… 192.168.159.2
77 2025-10-03 19:1… 192.168.159.2
248 2025-10-03 19:1… 192.168.159.129
249 2025-10-03 19:1… 192.168.159.129
Frame 3: 76 bytes on wire (608 bits), 76 bytes
Ethernet II, Src: VMware_a8:d0:96 (00:0c:29:a8:d0:96)
Destination: VMware_f5:9e:21 (00:50:56:f5:9
Source: VMware_a8:d0:96 (00:0c:29:a8:d0:96)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.159.129, Dst: 192.168.159.2
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 62
Identification: 0x3d98 (15768)
Flags: 0x00
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 64
Protocol: UDP (17)
Header Checksum: 0x7d42 [validation disabled]
[Header checksum status: Unverified]
Source Address: 192.168.159.129
Destination Address: 192.168.159.2
User Datagram Protocol, Src Port: 51658, Dst Port: 53
Source Port: 51658
0000 00 50 56 f5 9e 21 00 0c 29 a8 d0 96 08 00 45 00
0010 00 3e 3d 98 00 00 40 11 7d 42 c0 a8 9f 81 c0 a8
0020 9f 02 c9 ca 00 35 00 2a c0 10 99 94 01 00 00 01
0030 00 00 00 00 00 00 0c 76 69 62 72 61 6e 74 63 6c
0040 6f 75 64 03 6f 72 67 00 00 01 00 01
IP header
Byte Offset
0 1 2 3
Version
IHL (Header Length)
Type of Service (TOS)
Total Length
Identification
IP Flags x D M
Fragment Offset
Time To Live (TTL)
Protocol
Header Checksum
Source Address
Destination Address
IP Option (variable length, optional, not common)
Bit
Nibble
Byte
Word
20 Bytes
IHL (Internet Header Length)
App
Transport
Network
Link
Physical
Destination IP address
192.168.159.2 = 0xC0.A8.9F.02

DNS Layer 3

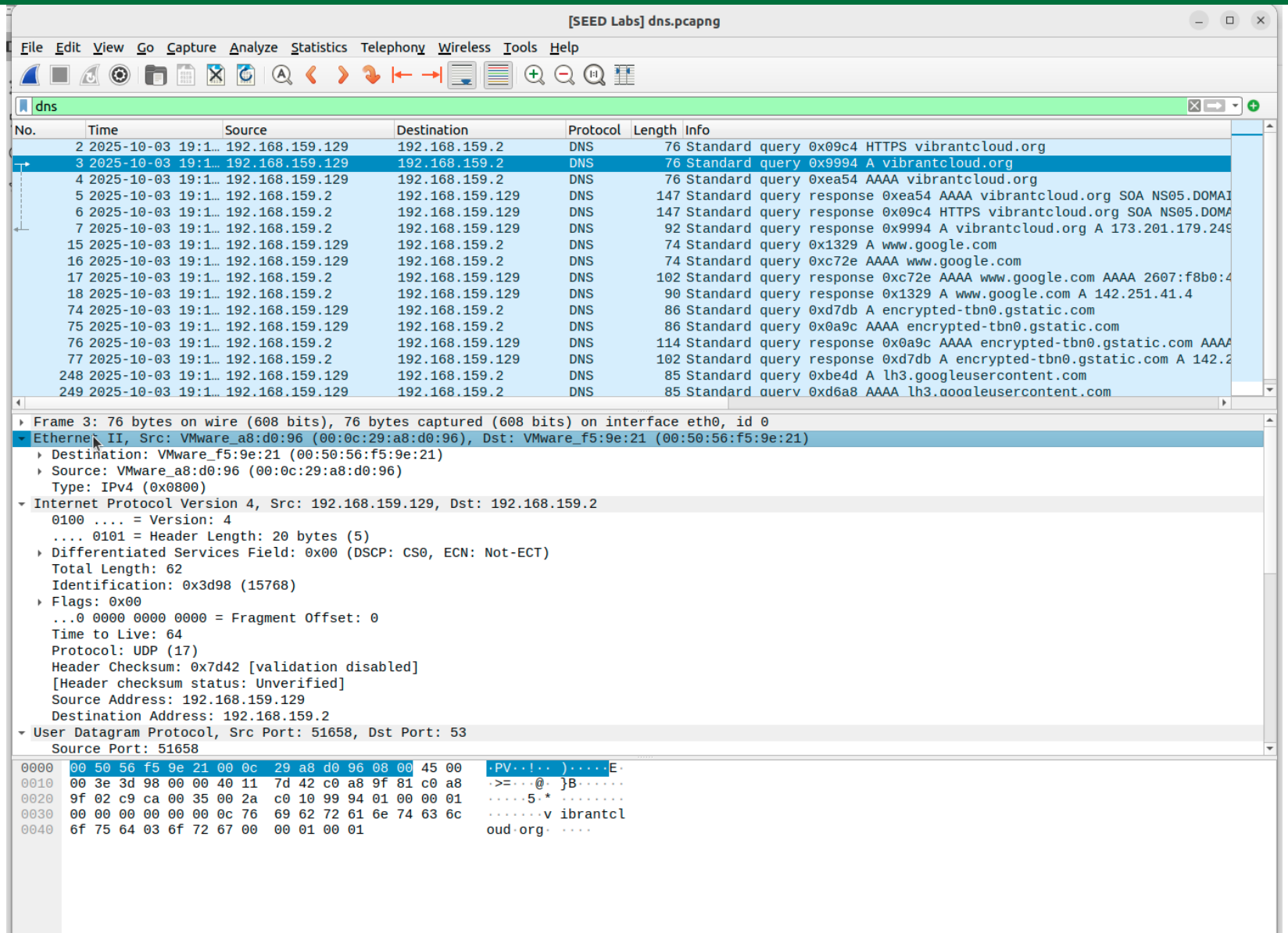


No optional components

How do we know no optional components?

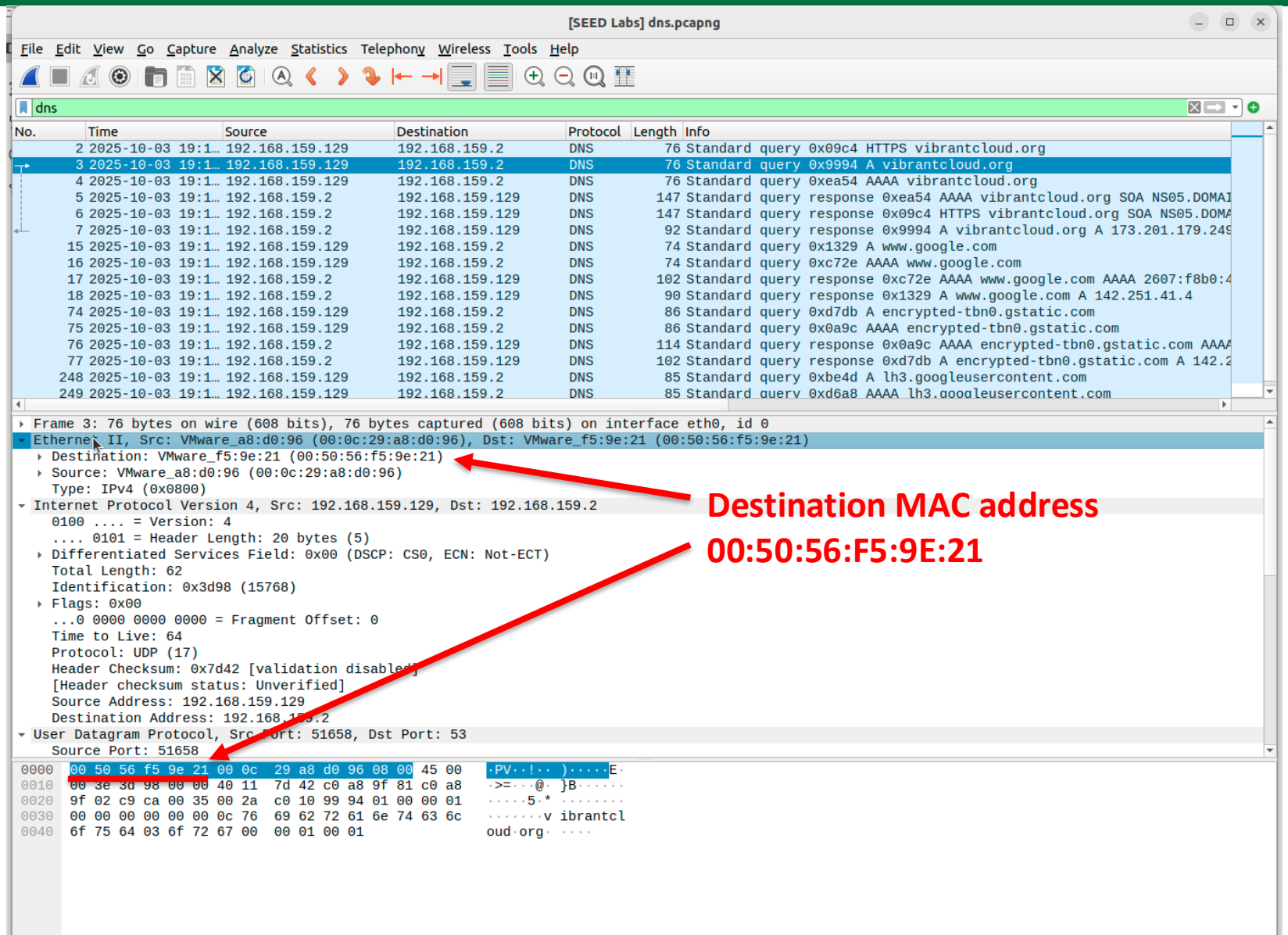
IHL was 5, so header ends after 20 bytes

DNS Layer 2



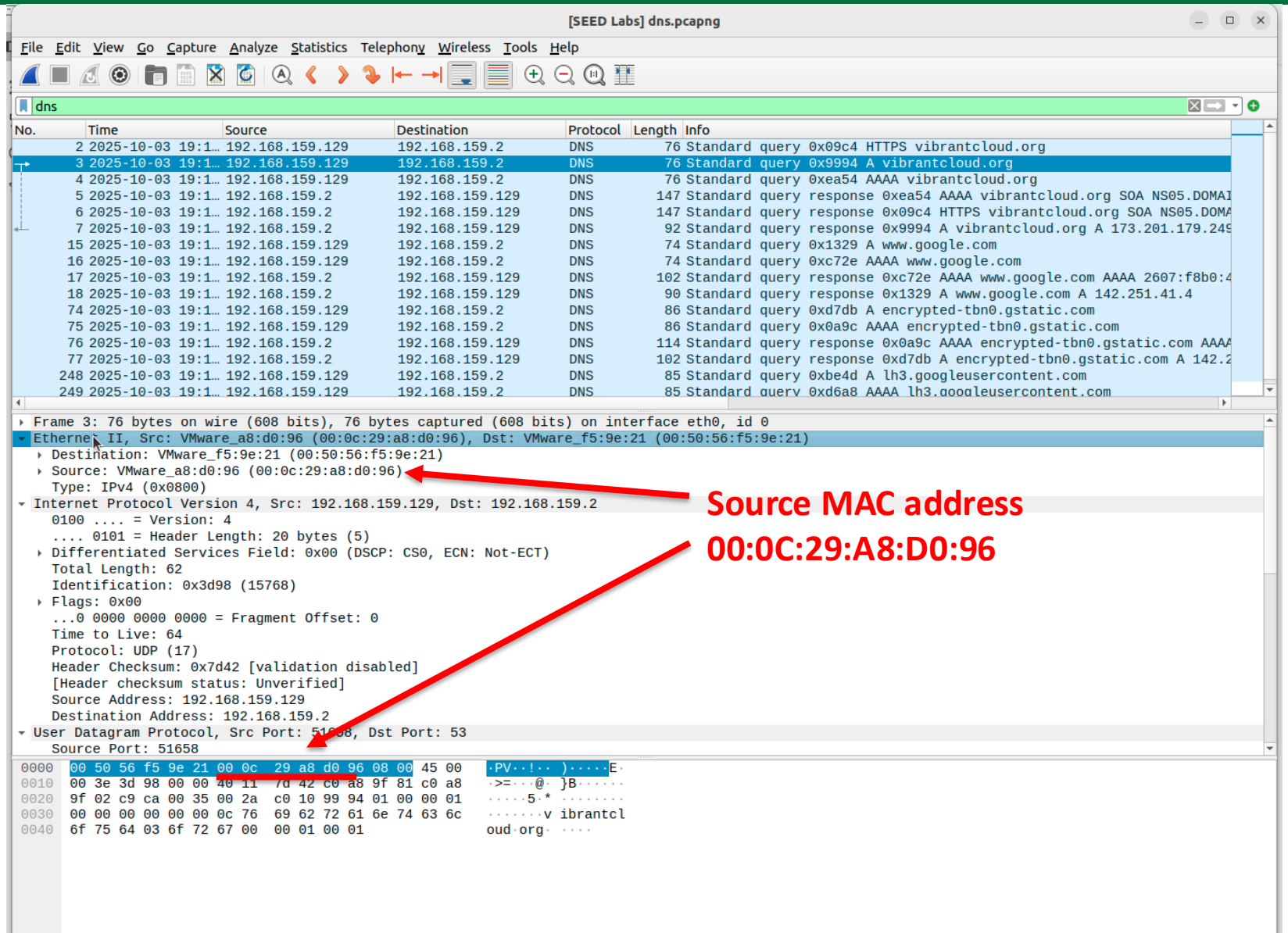
App
Transport
Network
Link
Physical

DNS Layer 2



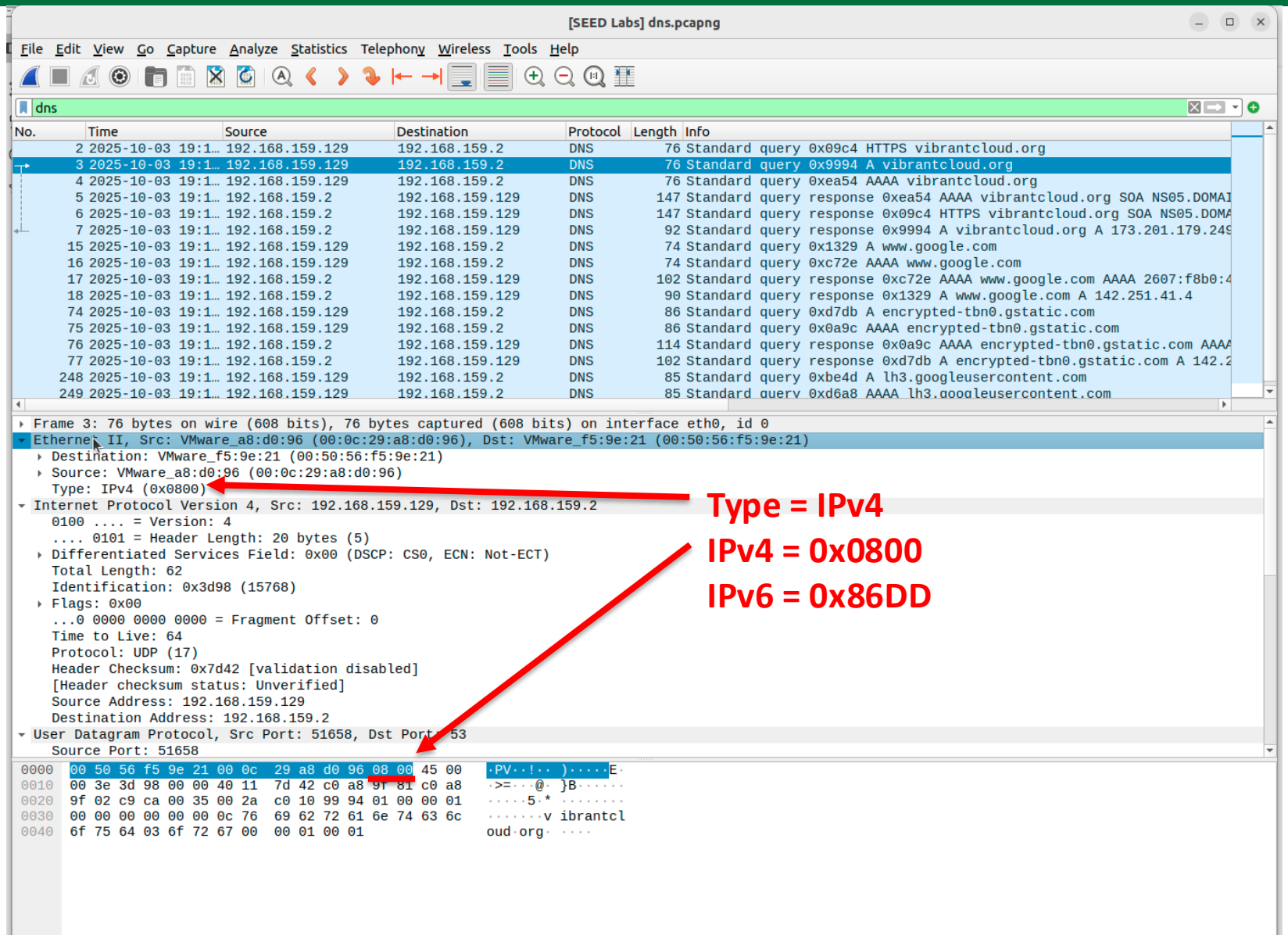
App
Transport
Network
Link
Physical

DNS Layer 2



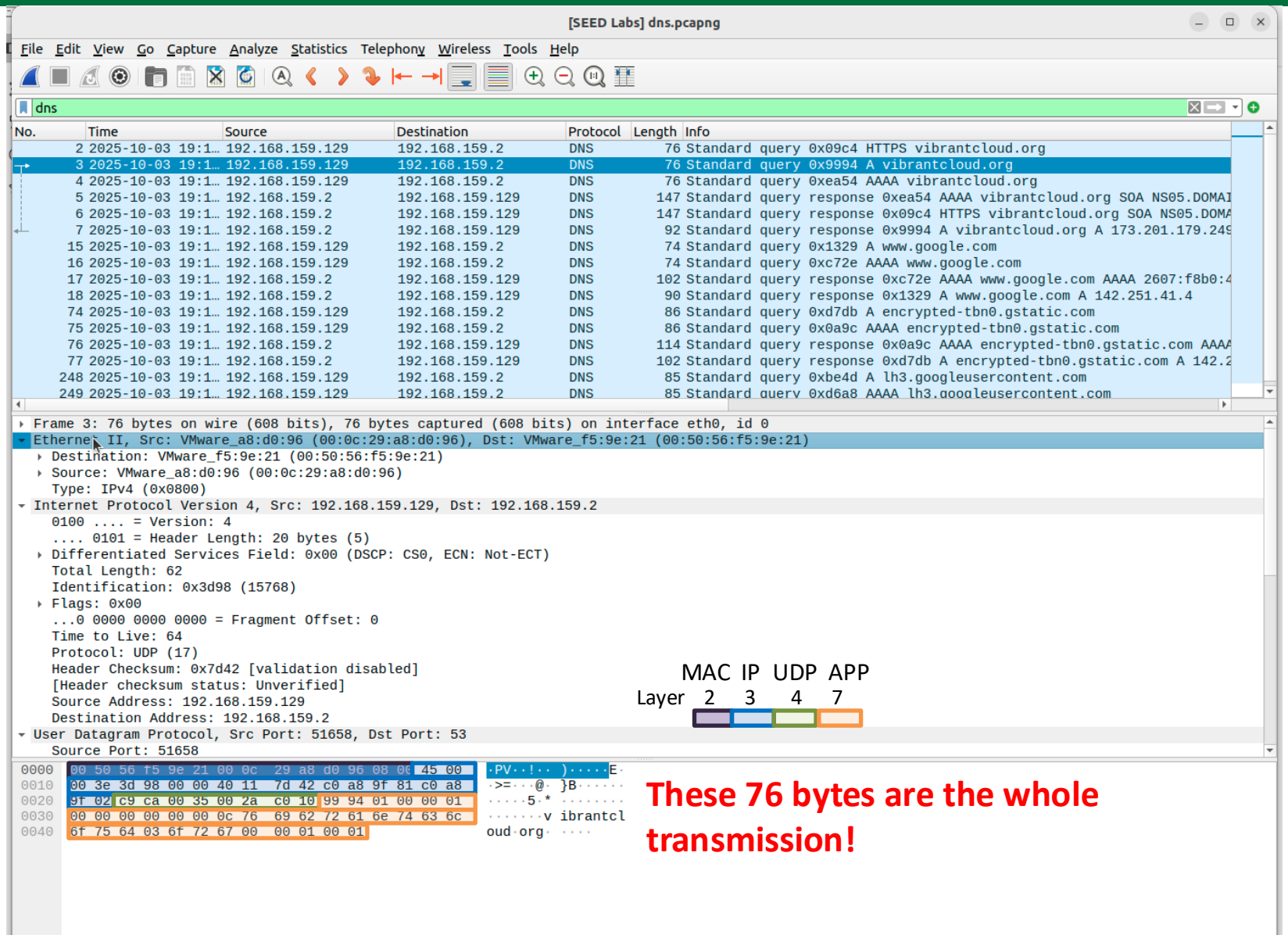
App
Transport
Network
Link
Physical

DNS Layer 2



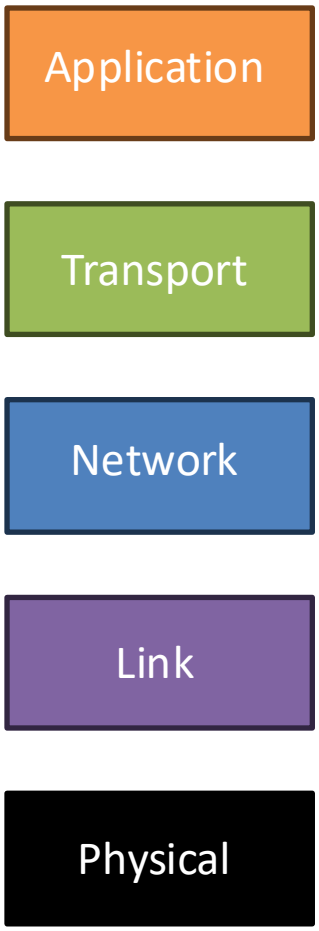
App
Transport
Network
Link
Physical

That is all 76 bytes!



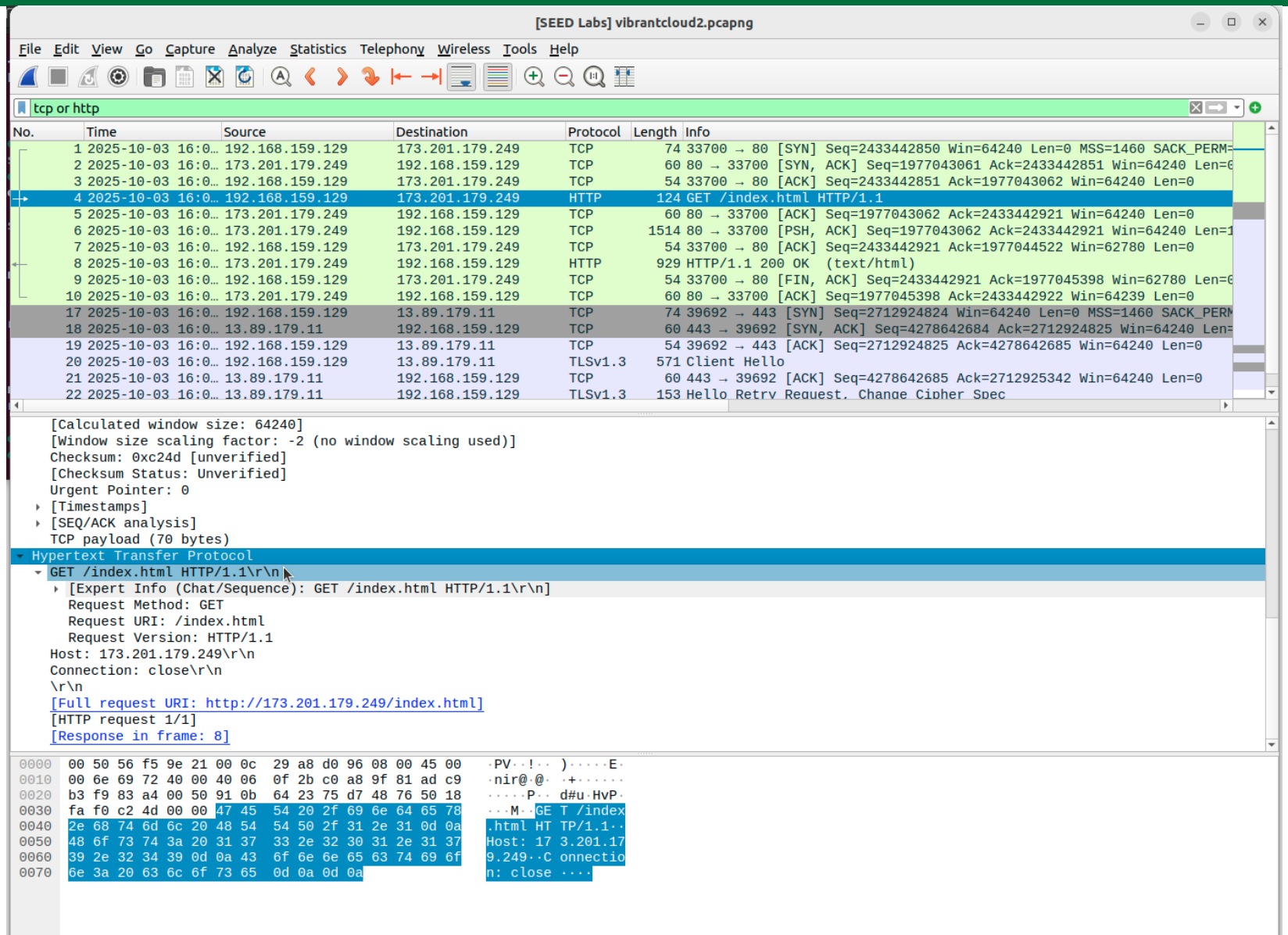
App
Transport
Network
Link
Physical

UDP and TCP requests

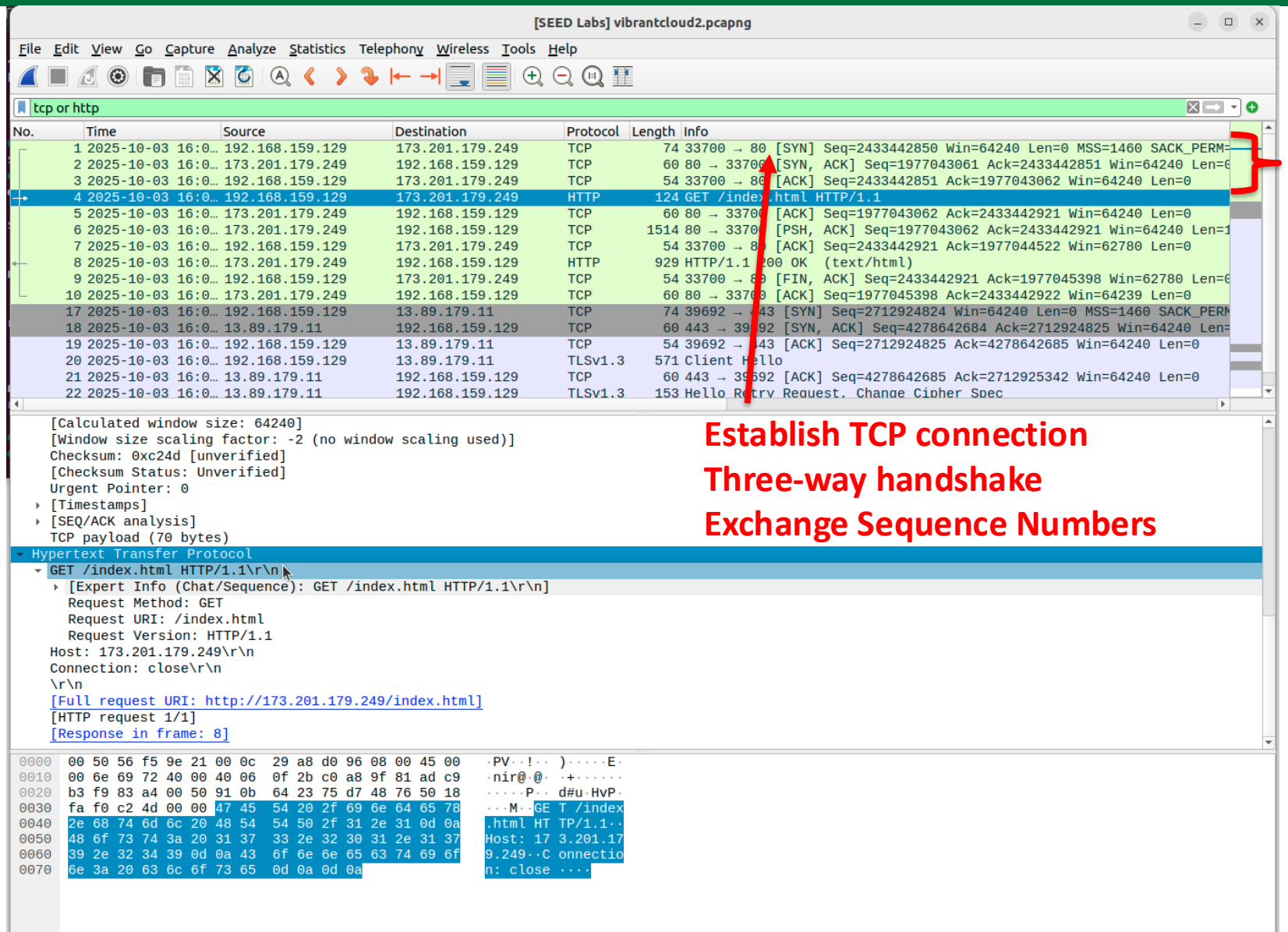


	UDP: DNS Request	**TCP: HTTP Request**
Application	Message format: Follow RFC 1034,1035	Message format: Follow RFC 7230-7235
Transport	UDP Port: 53	TCP Port: 80
Network	Request from DNS name server	Request from web server
Link	src and dst MAC	src and dst MAC
Physical	Bits	Bits

TCP: HTTP request



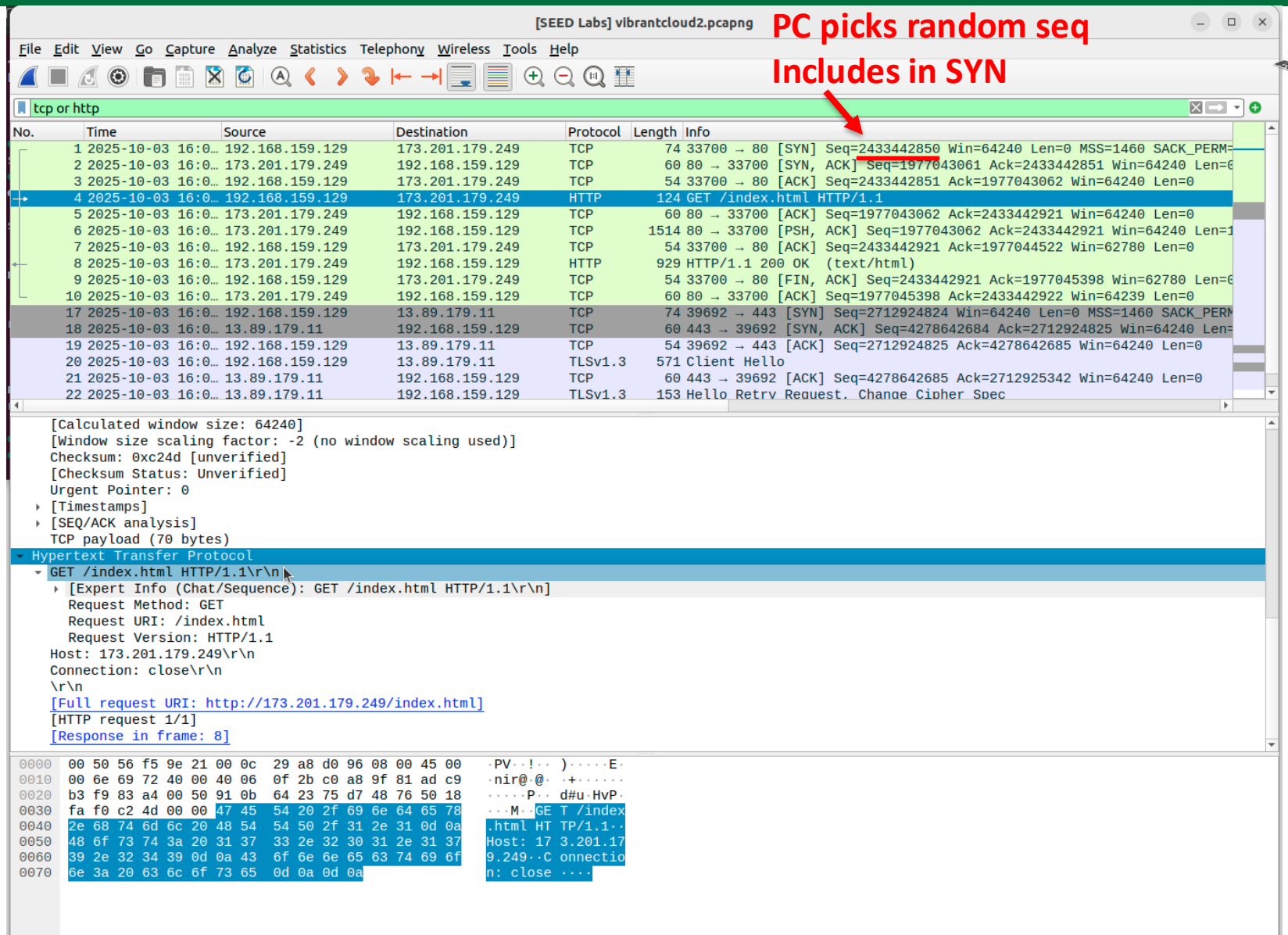
Begin with a TCP three-way handshake



SYN
SYN ACK
ACK

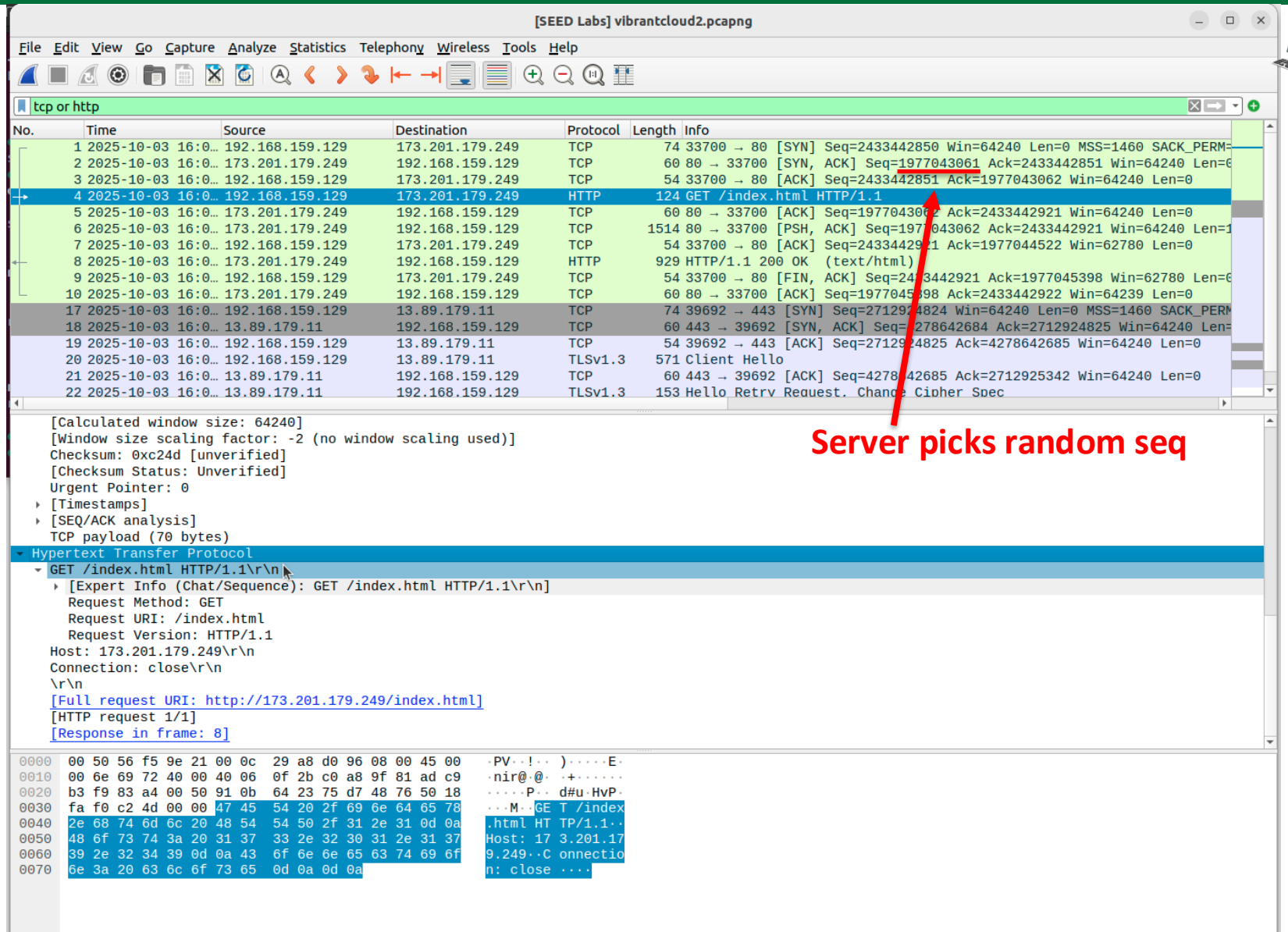
Establish TCP connection
Three-way handshake
Exchange Sequence Numbers

Begin with a TCP three-way handshake

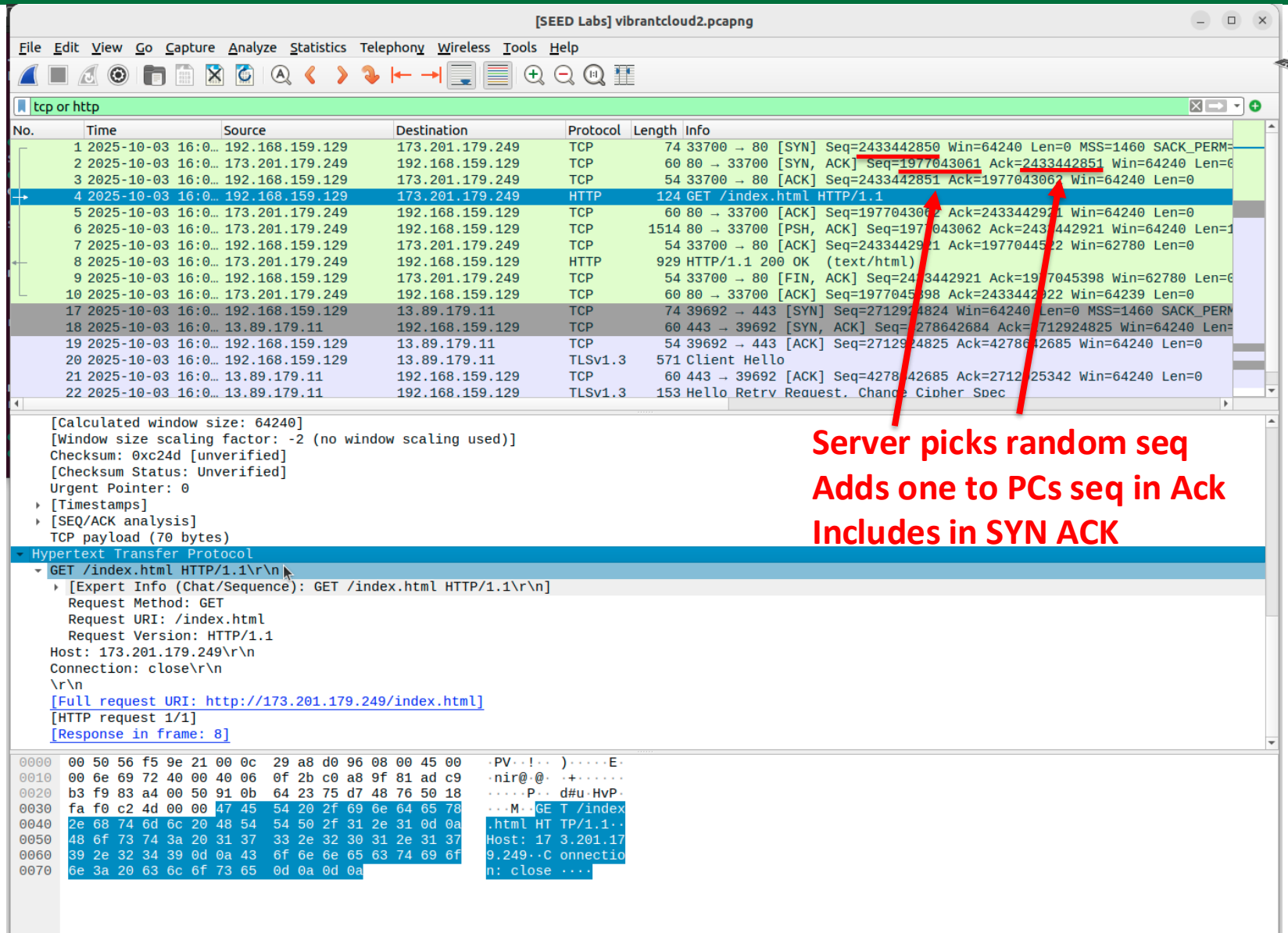




Begin with a TCP three-way handshake



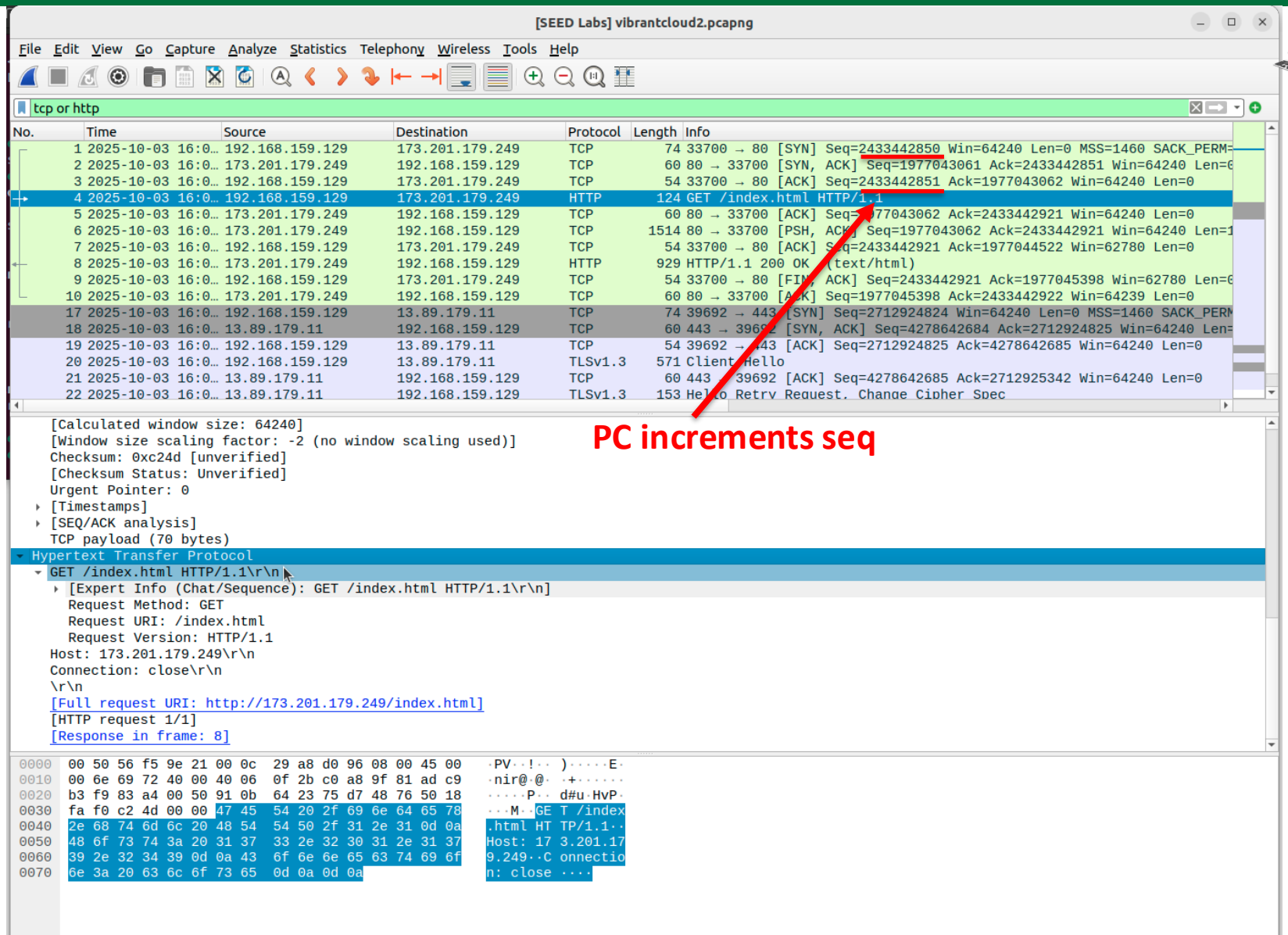
Begin with a TCP three-way handshake



SYN ACK

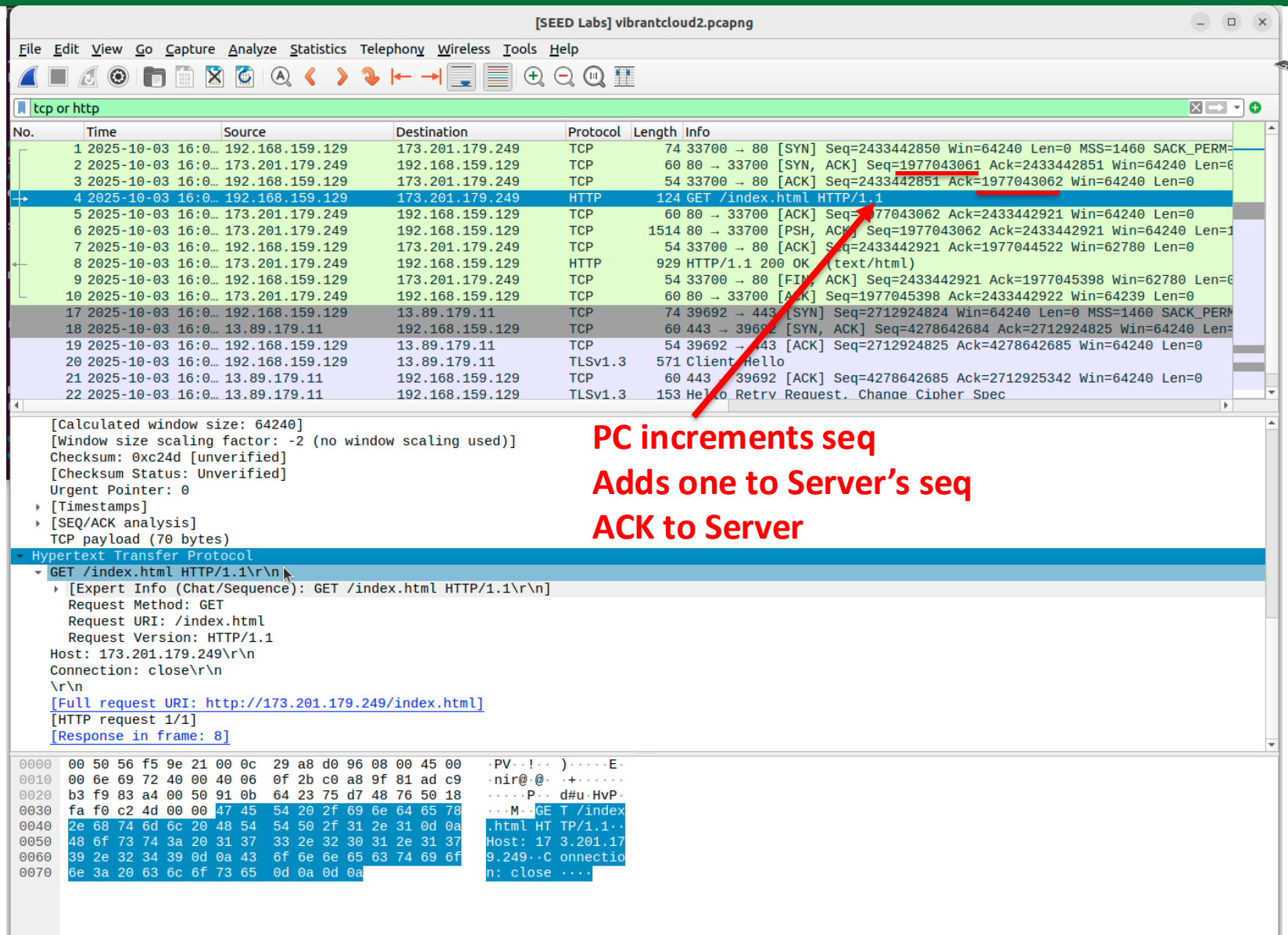
Server picks random seq
Adds one to PCs seq in Ack
Includes in SYN ACK

Begin with a TCP three-way handshake





Begin with a TCP three-way handshake



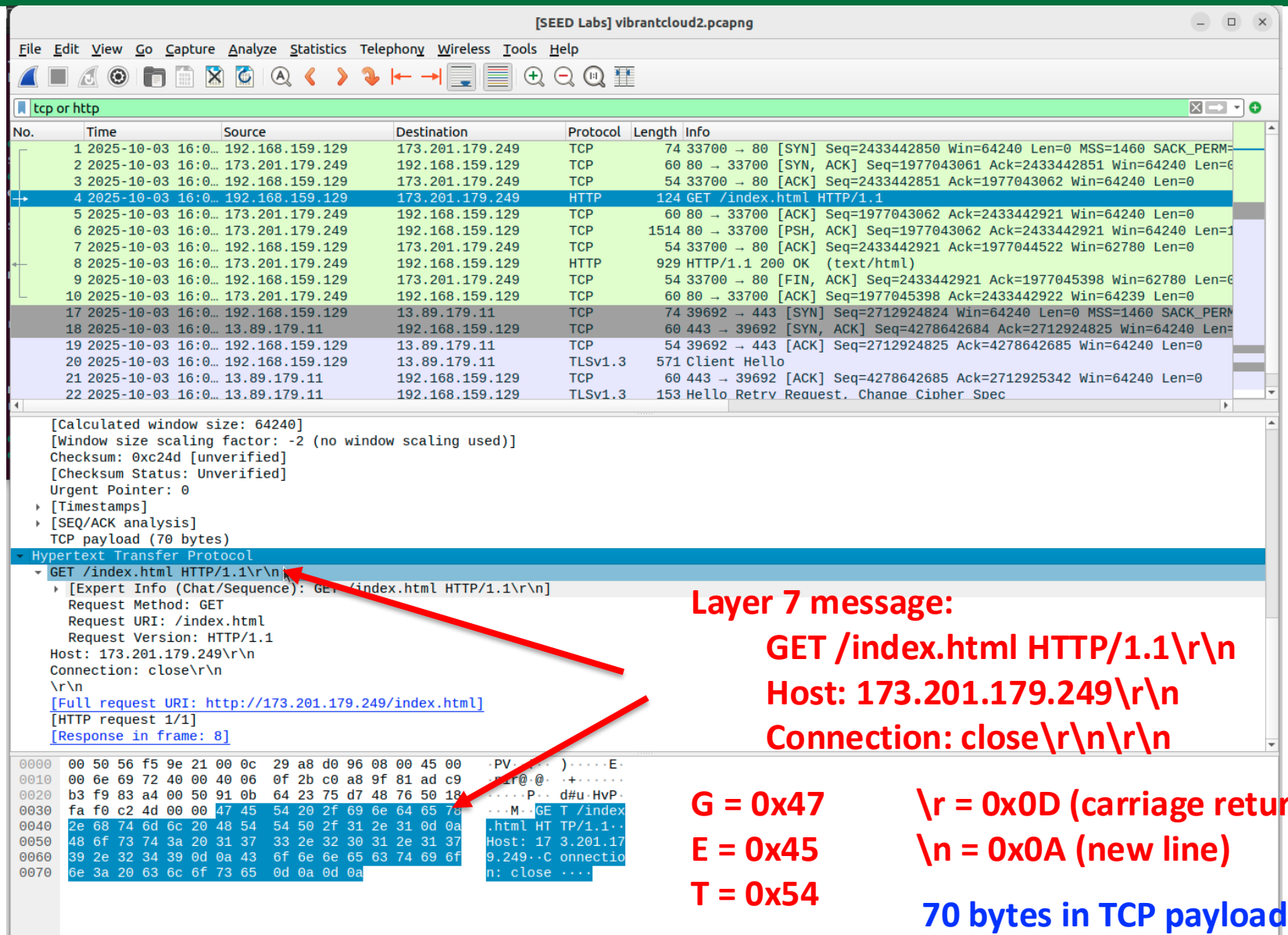
PC increments seq

Adds one to Server's seq

ACK to Server



Layer 7: HTTP request



App
Transport
Network
Link
Physical

Layer 7 message:

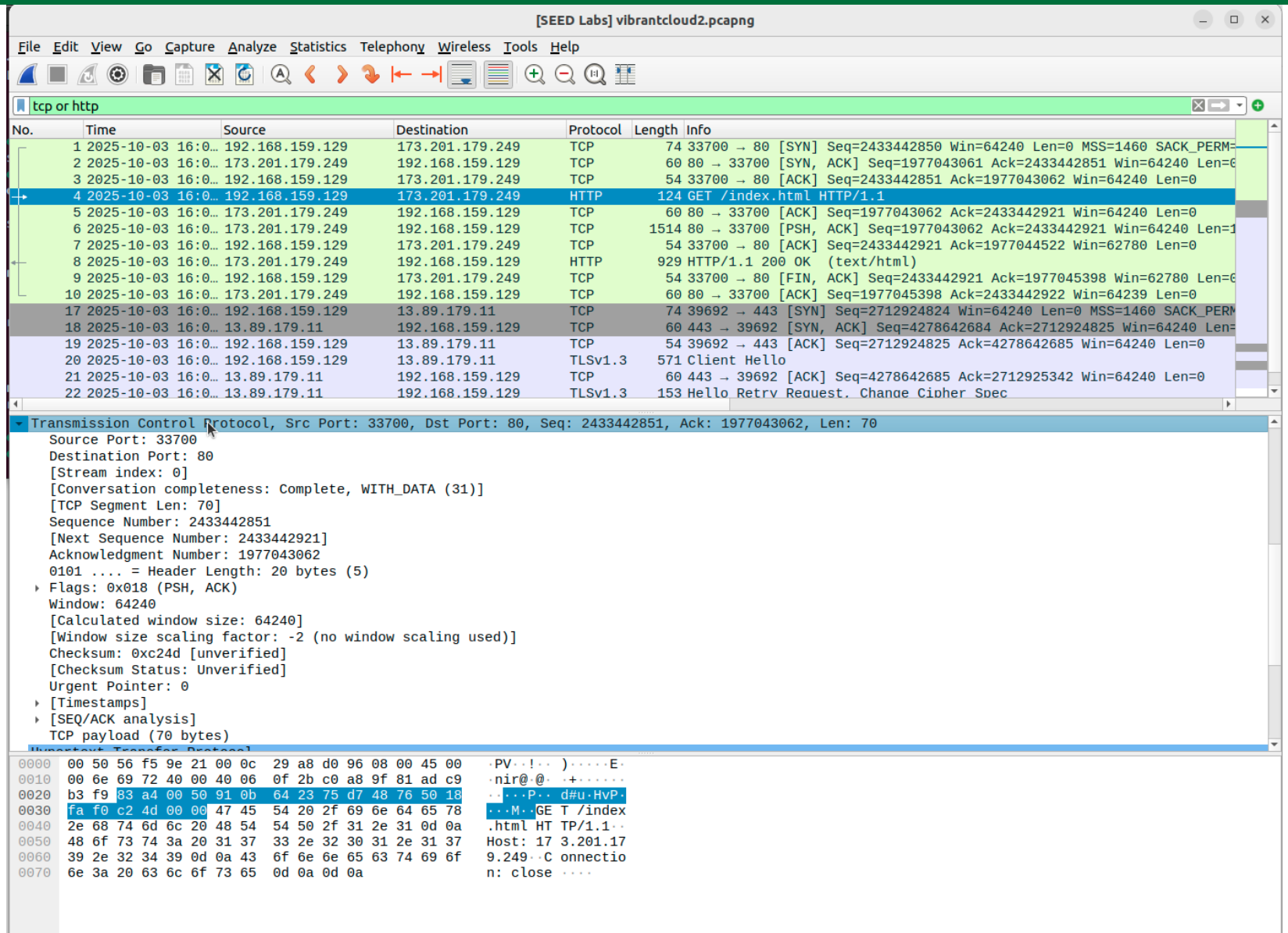
GET /index.html HTTP/1.1\r\n
Host: 173.201.179.249\r\n
Connection: close\r\n\r\n

G = 0x47
E = 0x45
T = 0x54

\r = 0x0D (carriage return)
\n = 0x0A (new line)

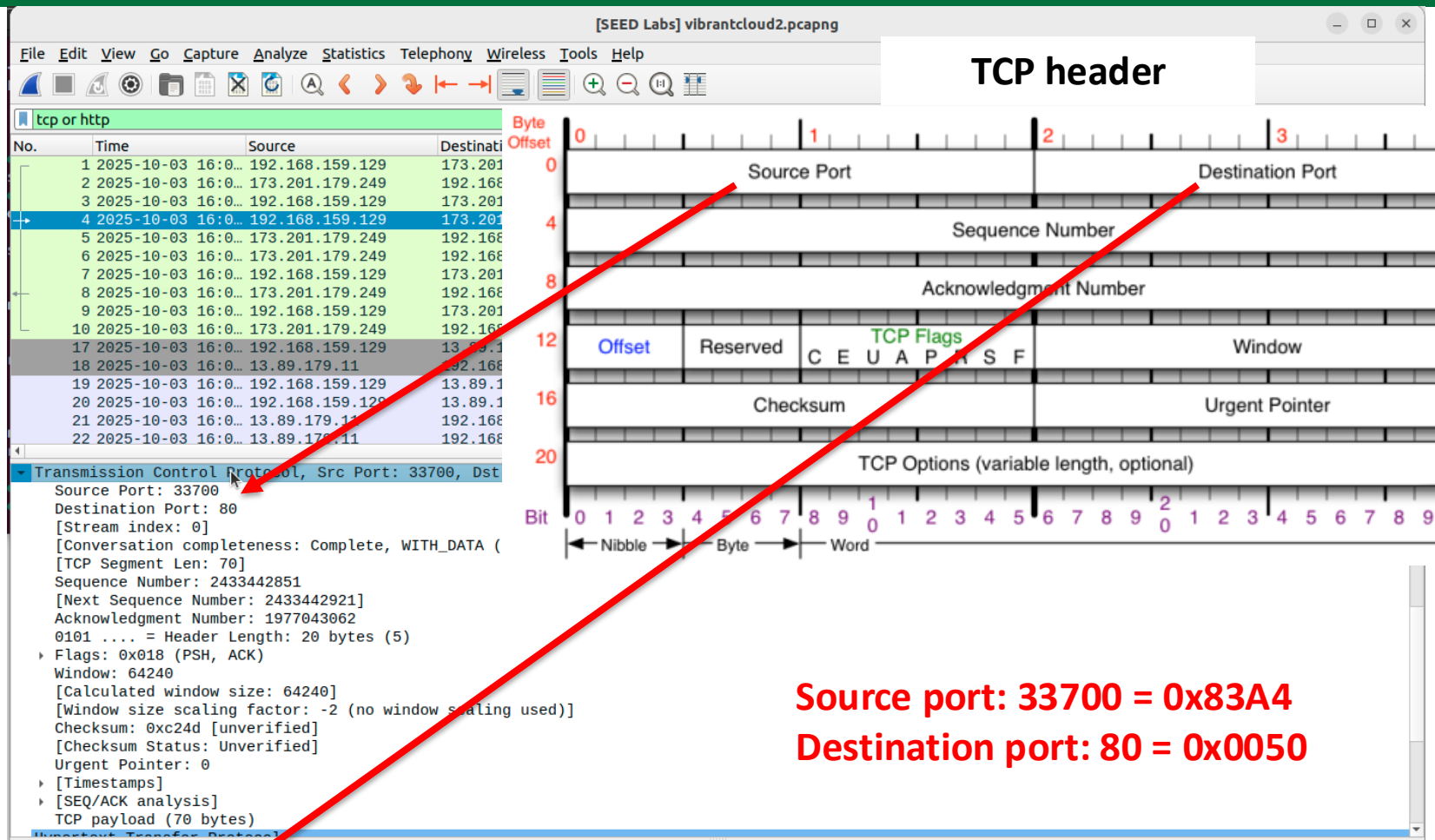
70 bytes in TCP payload

Layer 4: TCP header



App
Transport
Network
Link
Physical

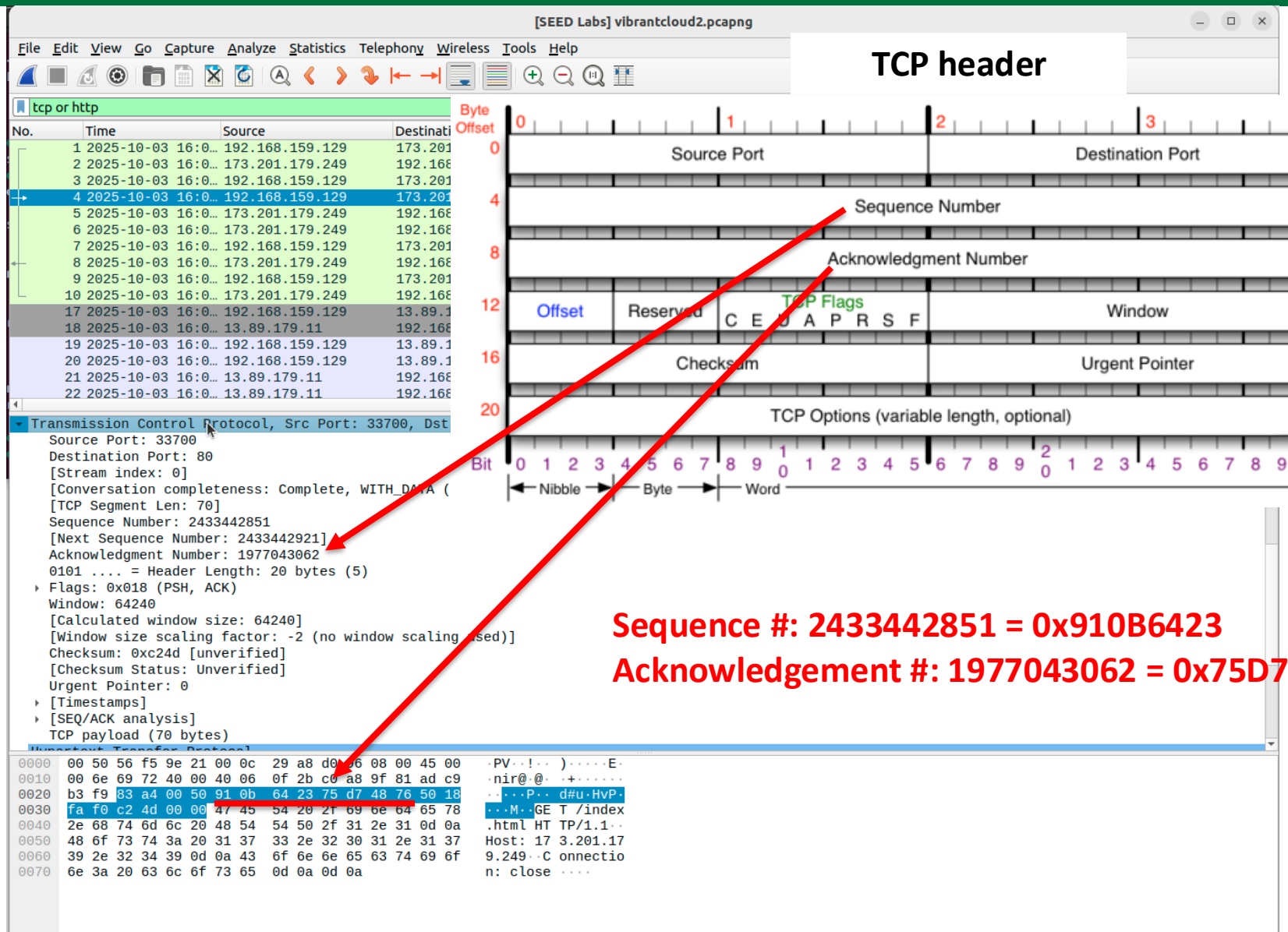
Layer 4: TCP header



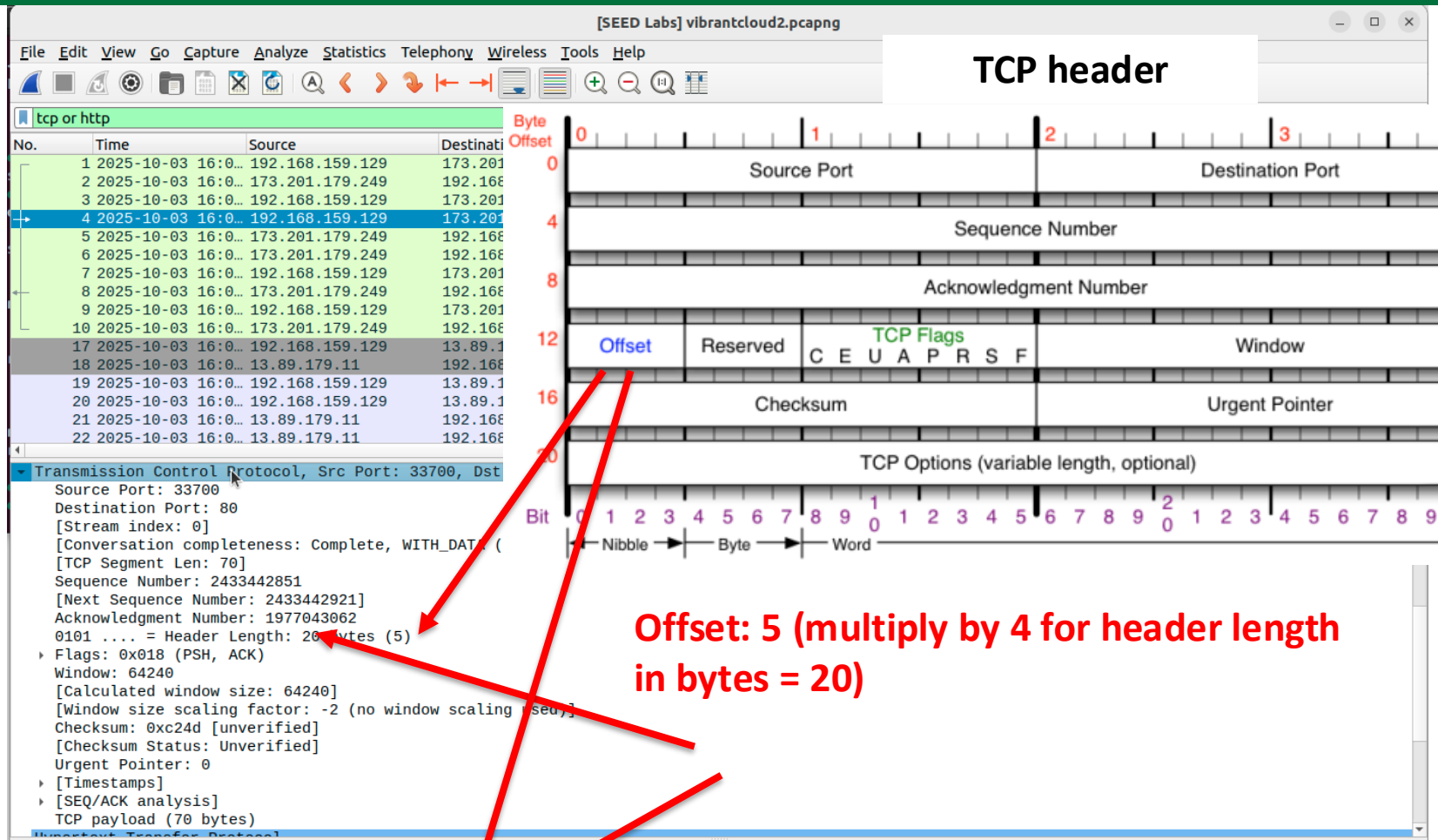
Source port: 33700 = 0x83A4

Destination port: 80 = 0x0050

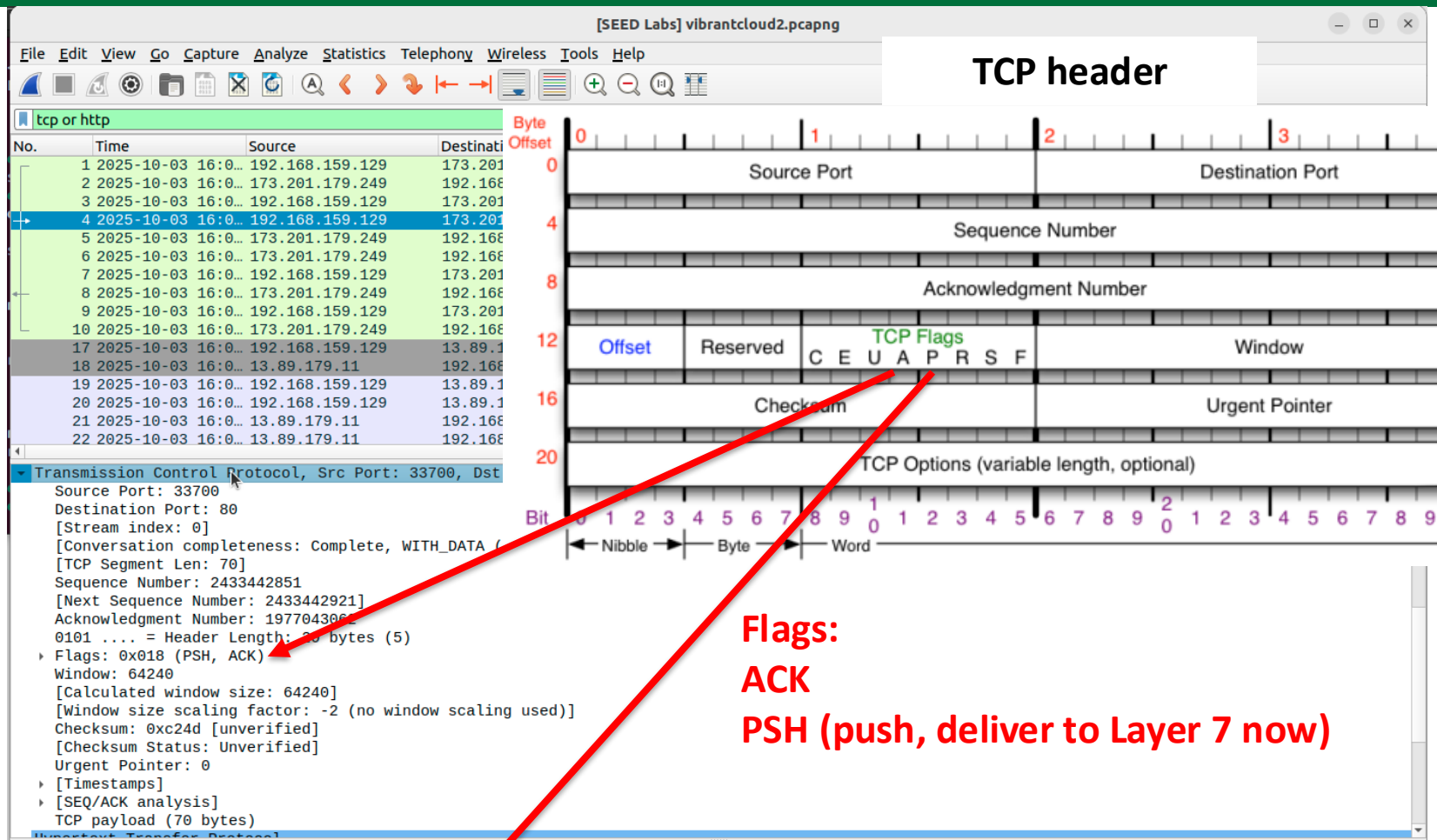
Layer 4: TCP header



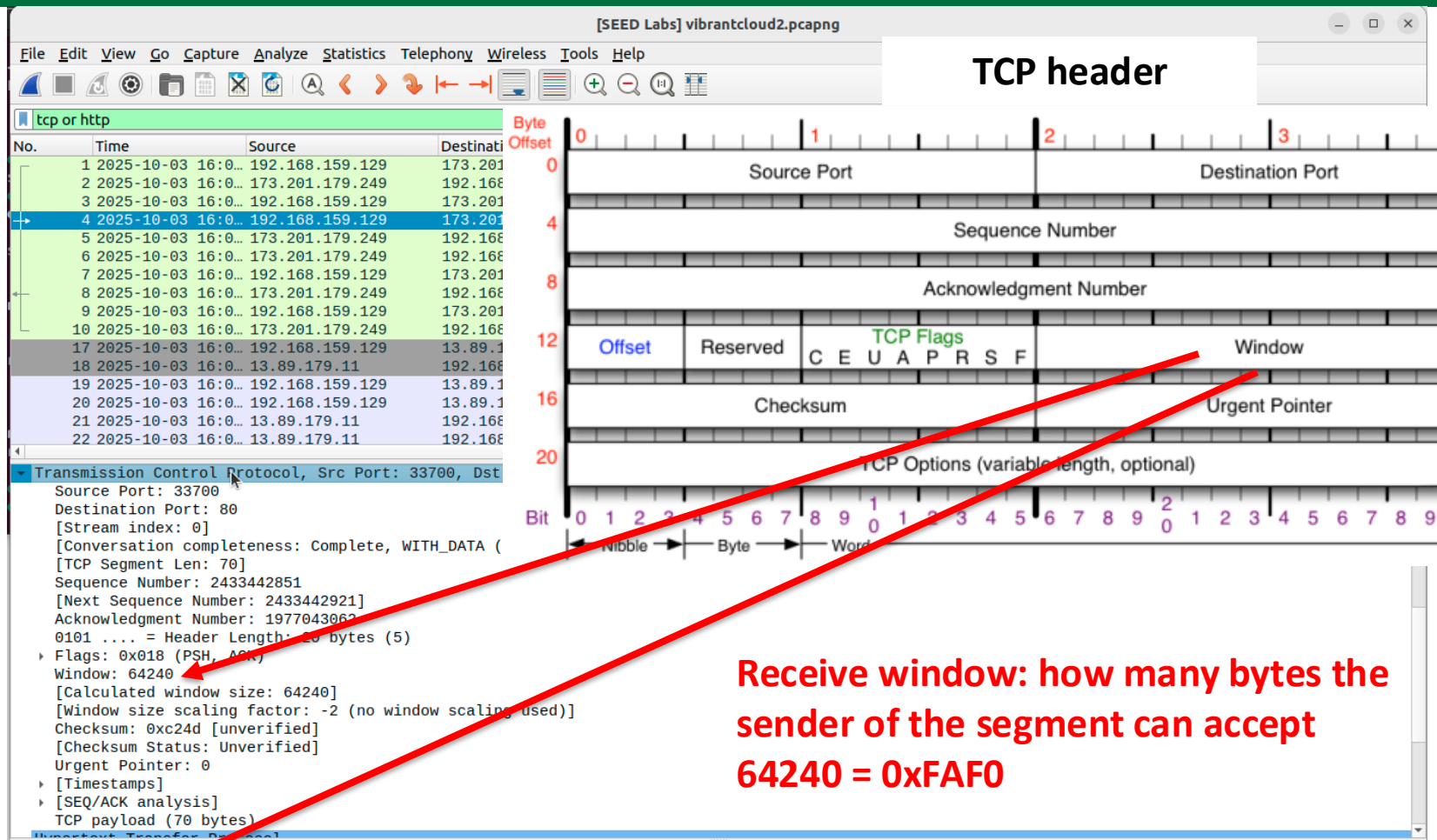
Layer 4: TCP header



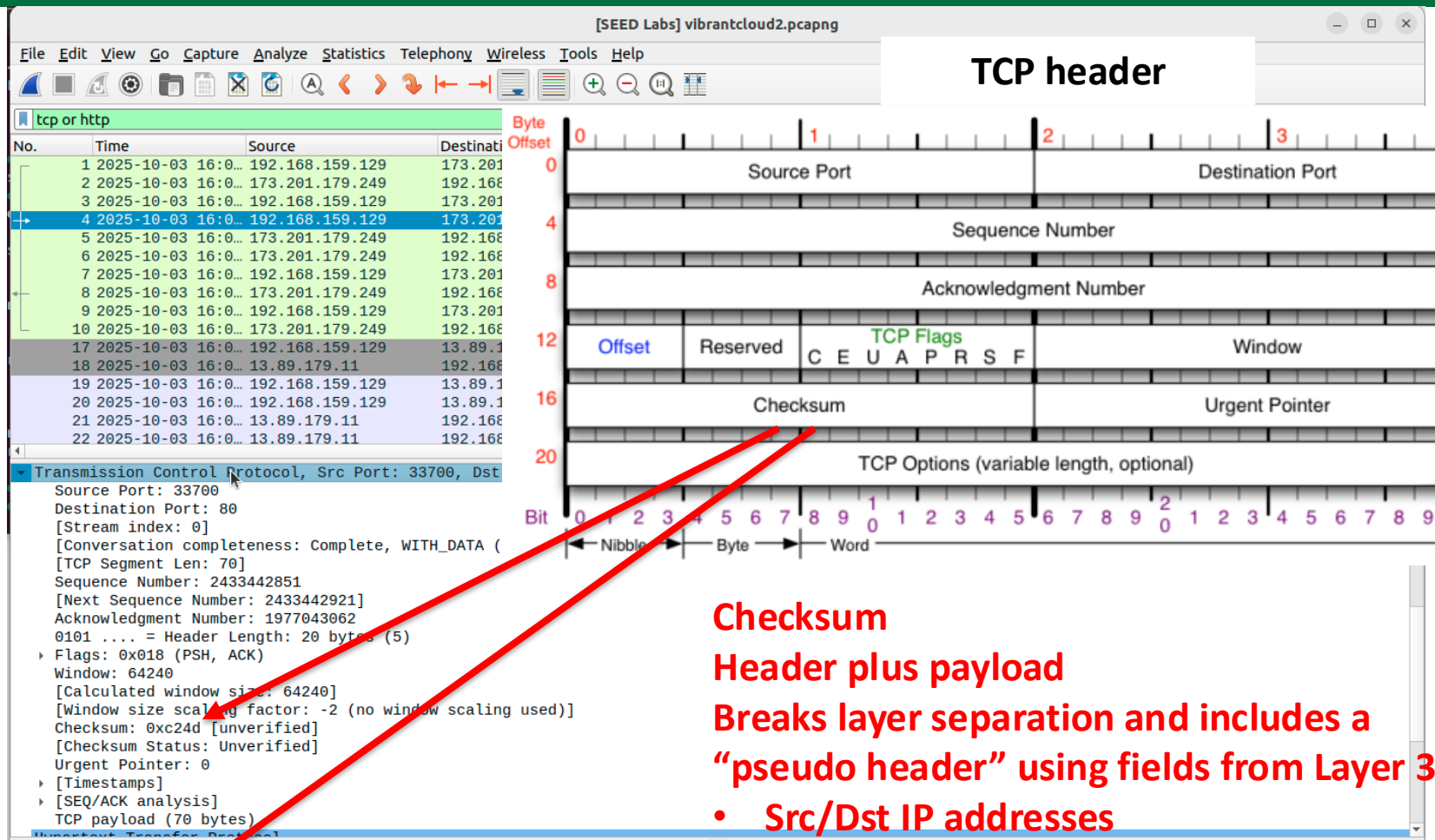
Layer 4: TCP header



Layer 4: TCP header



Layer 4: TCP header



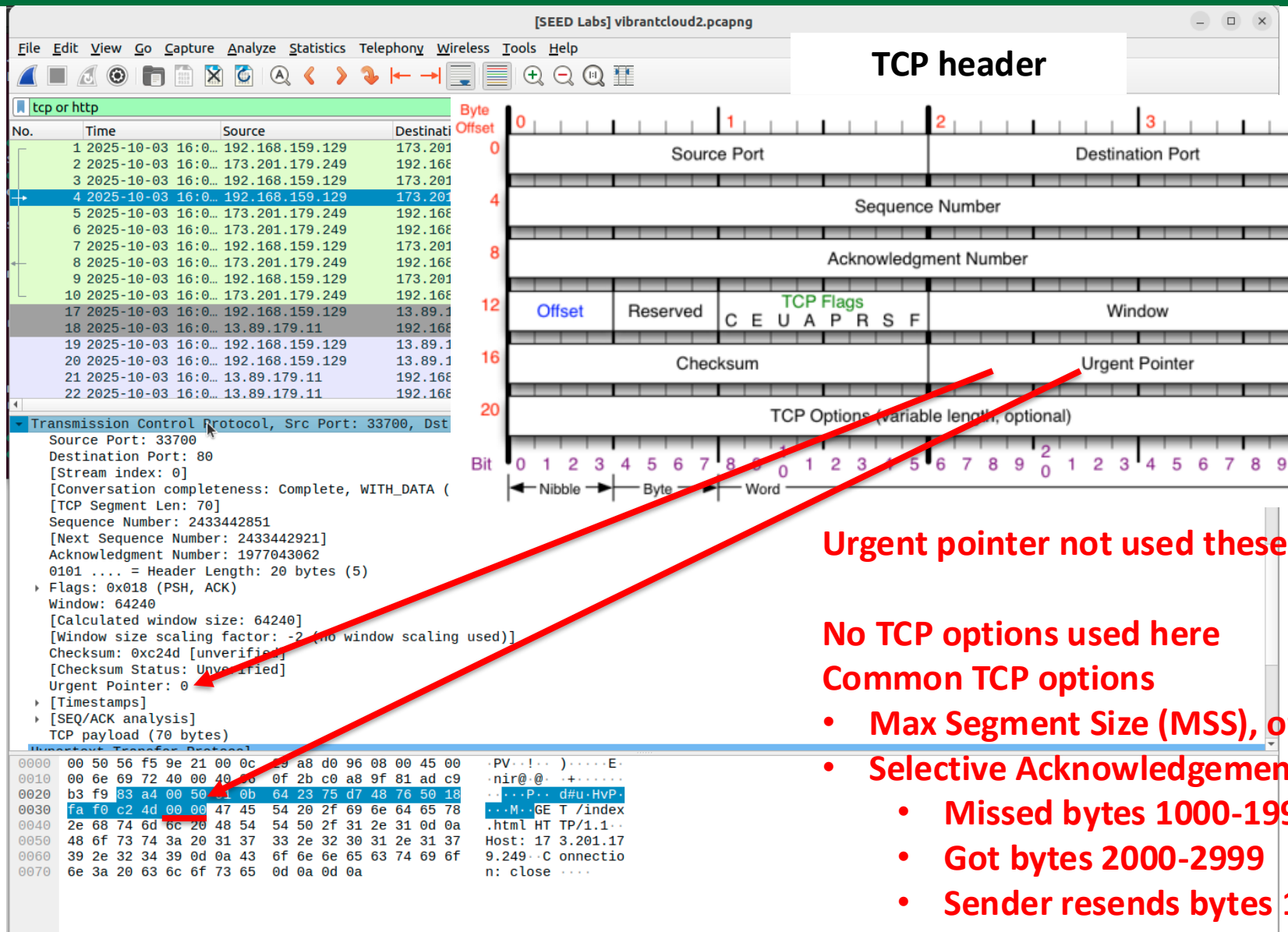
Checksum

Header plus payload

Breaks layer separation and includes a "pseudo header" using fields from Layer 3:

- **Src/Dst IP addresses**
- **Protocol**
- **TCP segment length (header + payload)**

Layer 4: TCP header



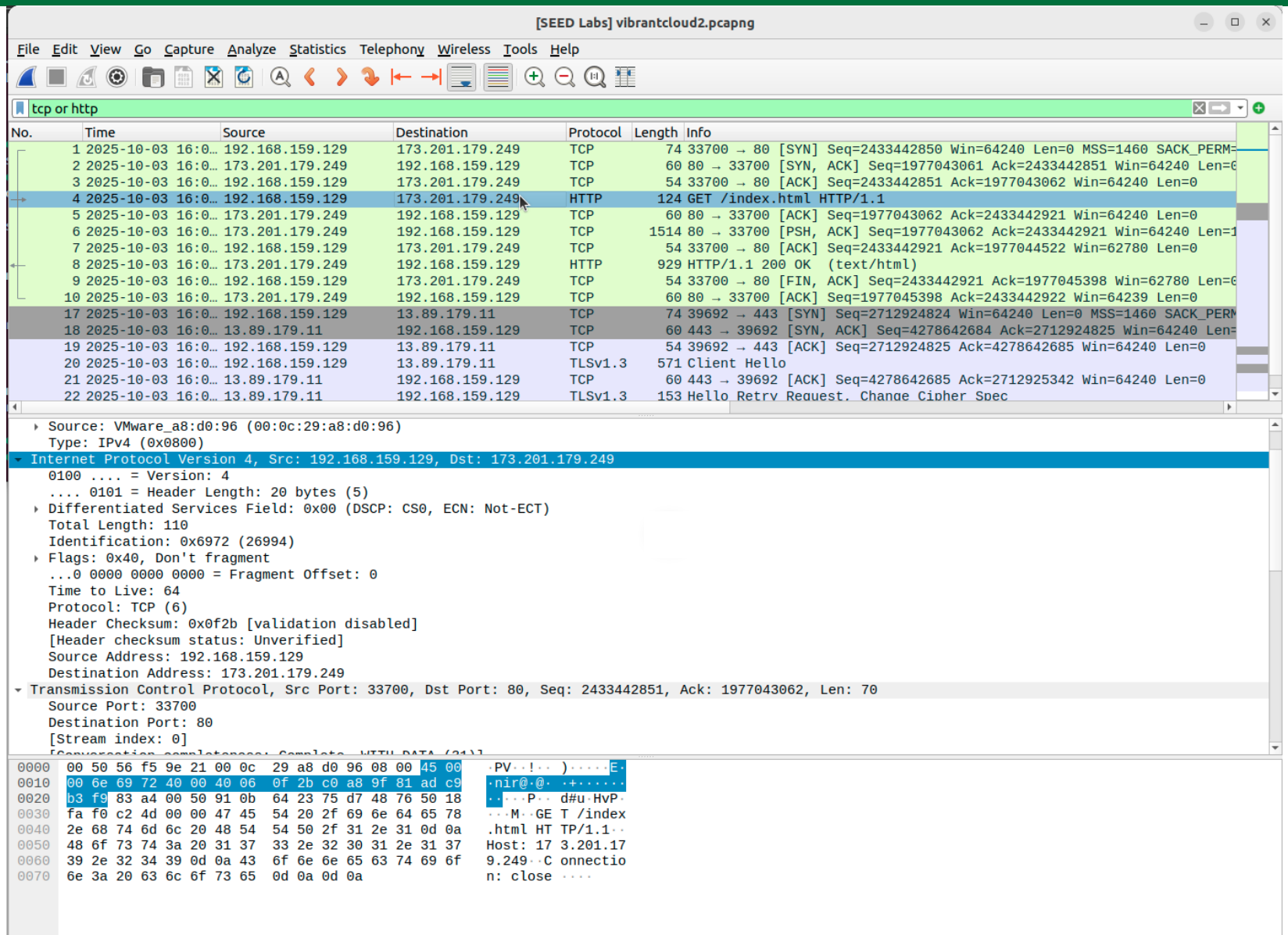
Urgent pointer not used these days

No TCP options used here

Common TCP options

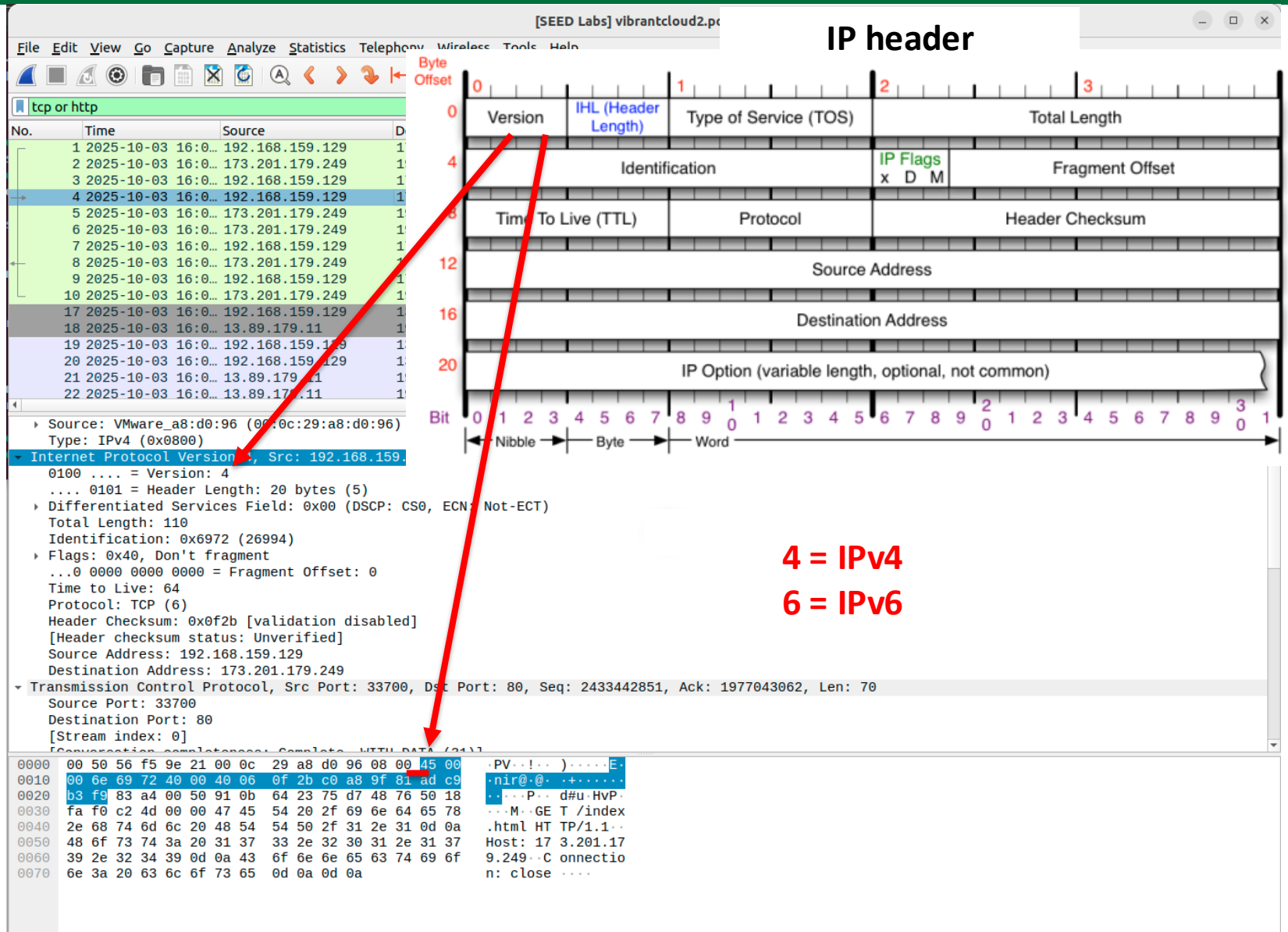
- **Max Segment Size (MSS), often 1460**
- **Selective Acknowledgements (SACK)**
  - **Missed bytes 1000-1999**
  - **Got bytes 2000-2999**
  - **Sender resends bytes 1000-1999**

Layer 3: IP header

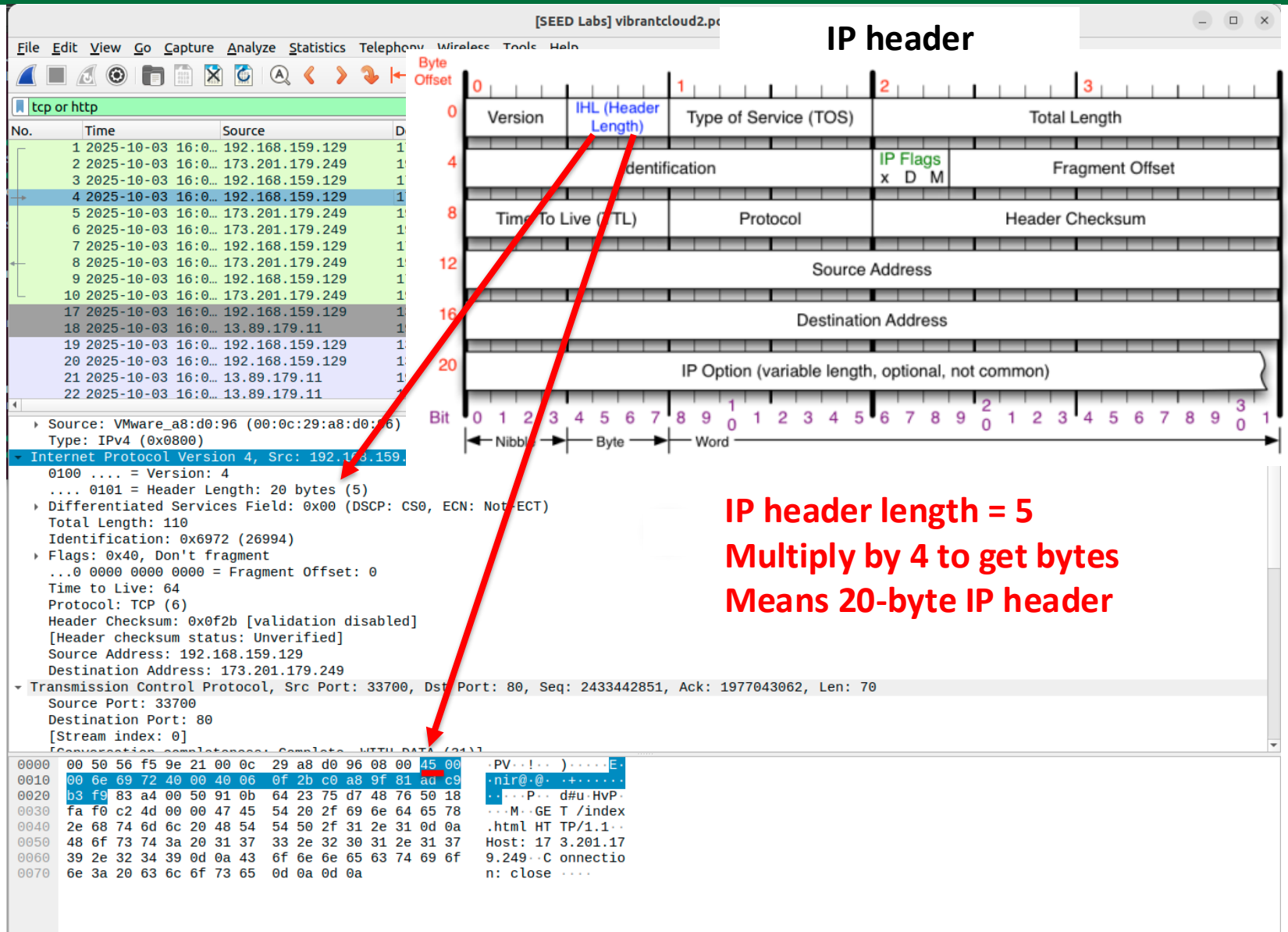


App
Transport
Network
Link
Physical

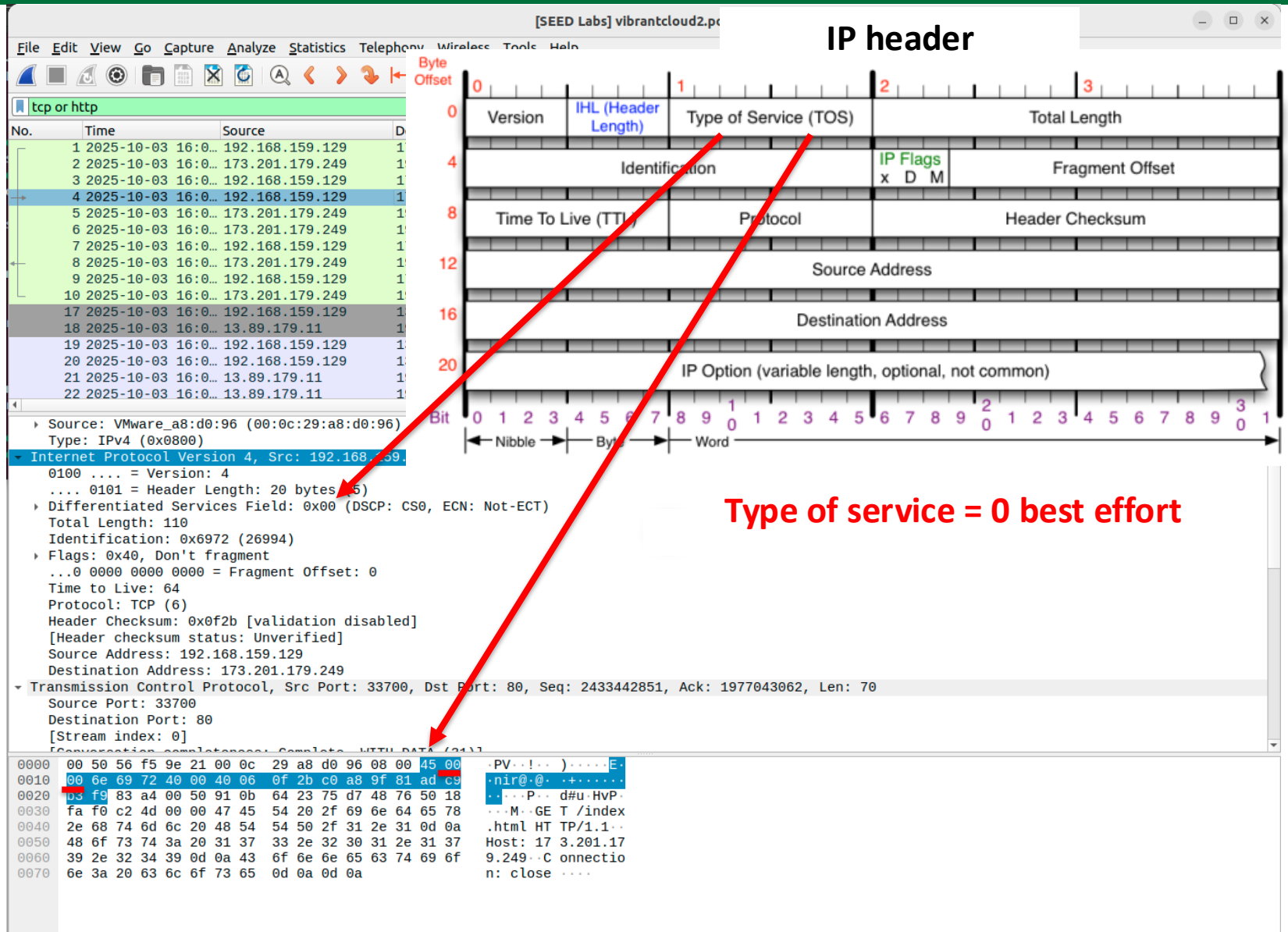
Layer 3: IP header



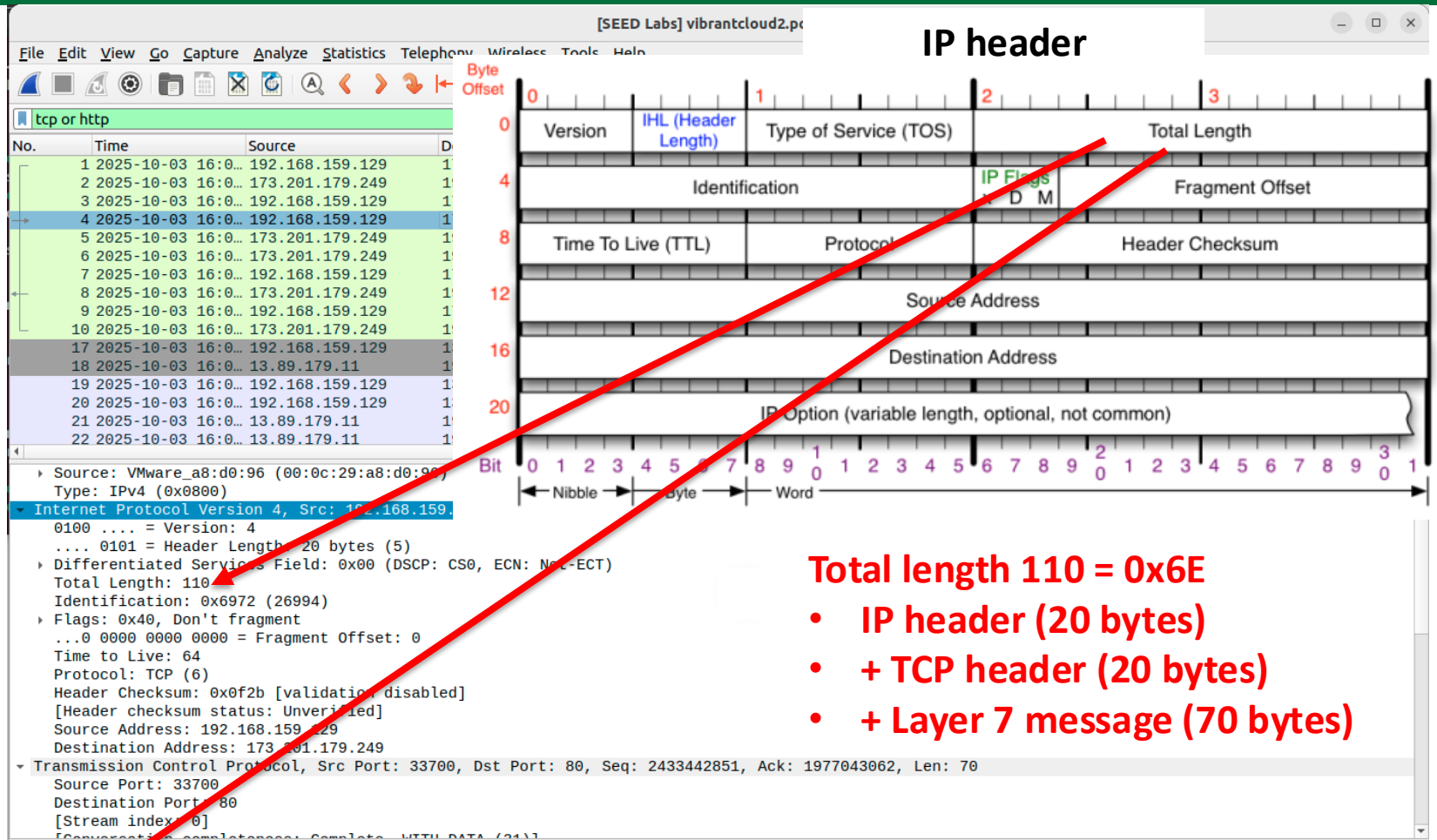
Layer 3: IP header



Layer 3: IP header



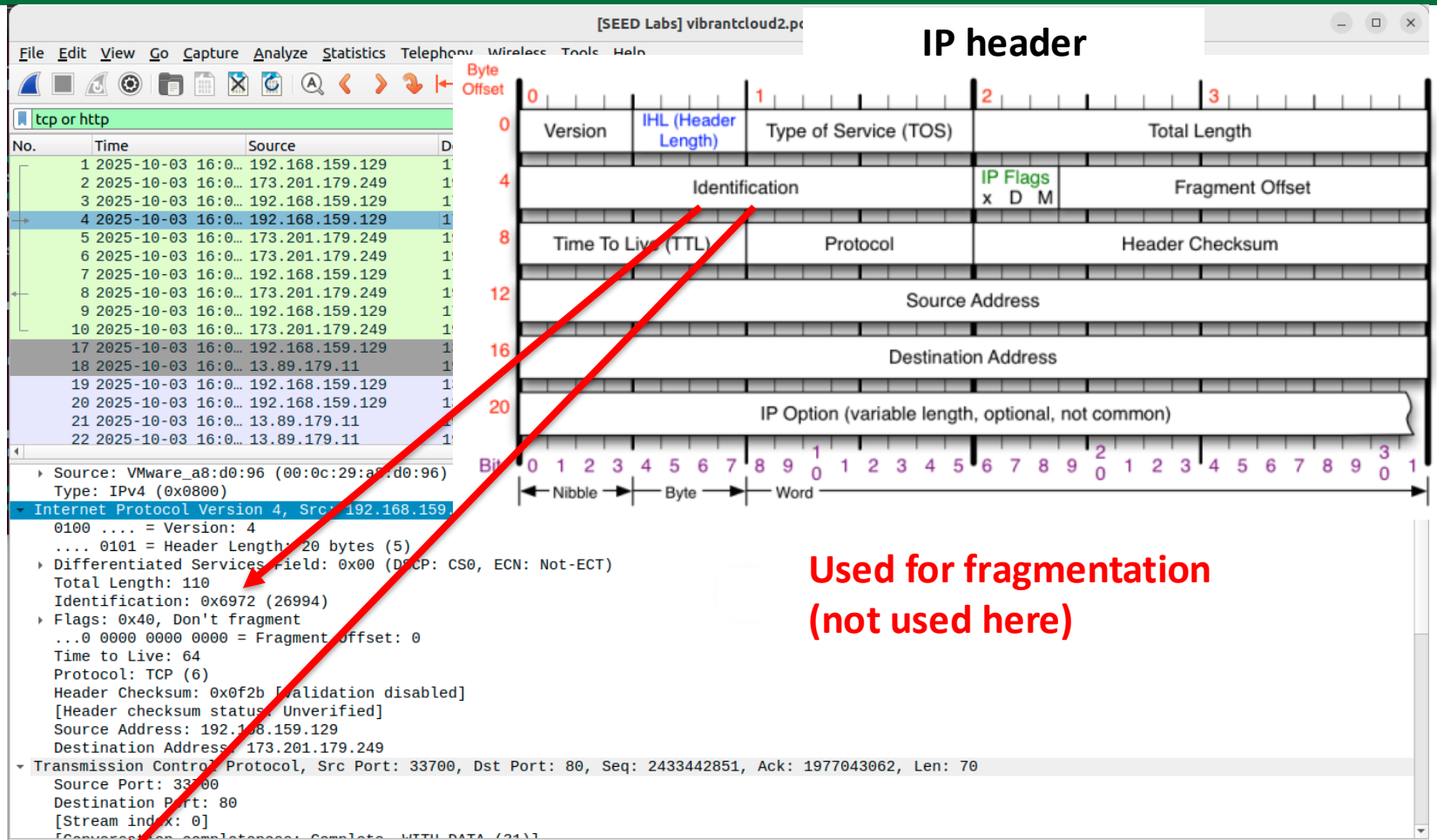
Layer 3: IP header



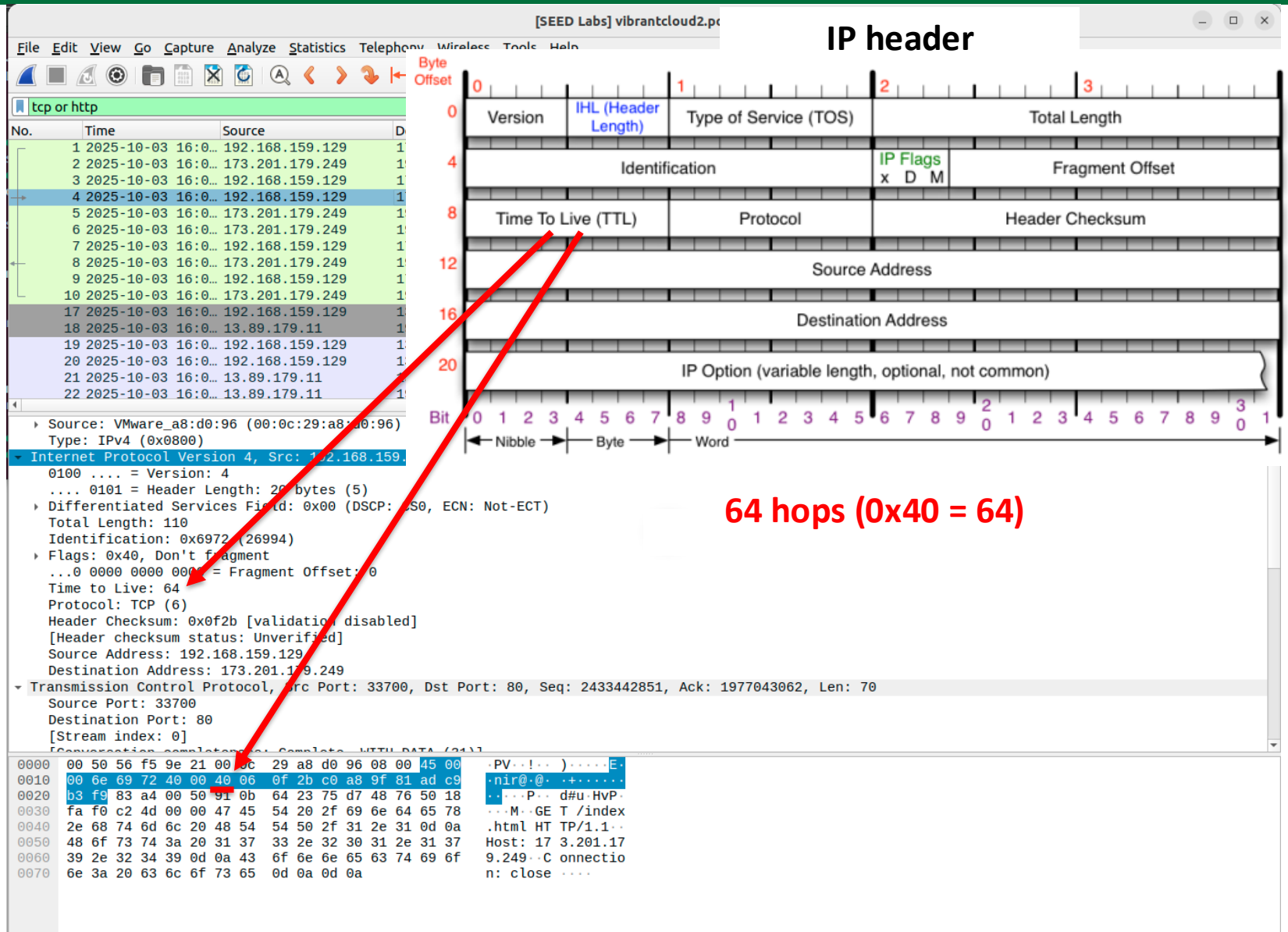
Total length 110 = 0x6E

- **IP header (20 bytes)**
- **+ TCP header (20 bytes)**
- **+ Layer 7 message (70 bytes)**

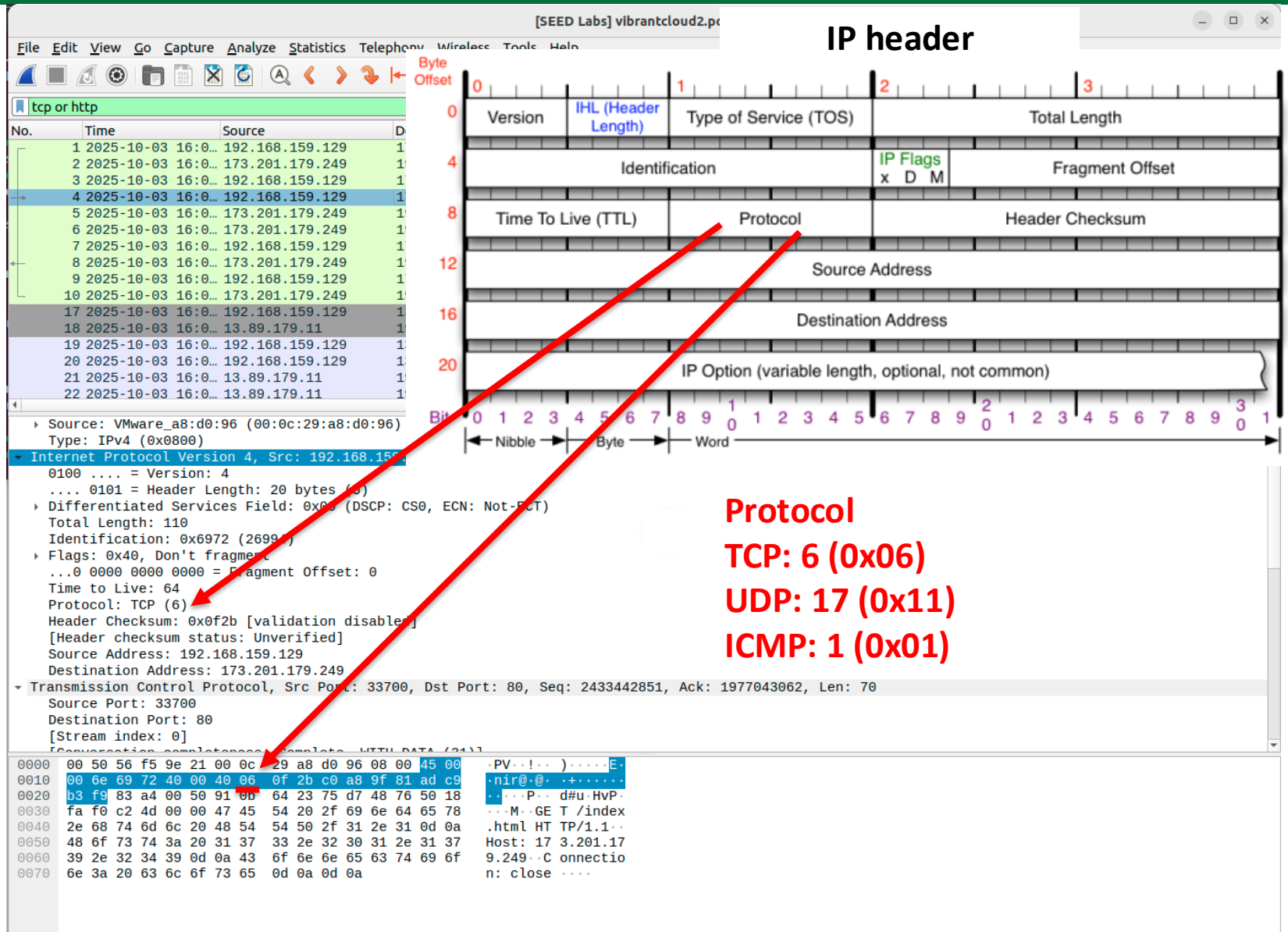
Layer 3: IP header



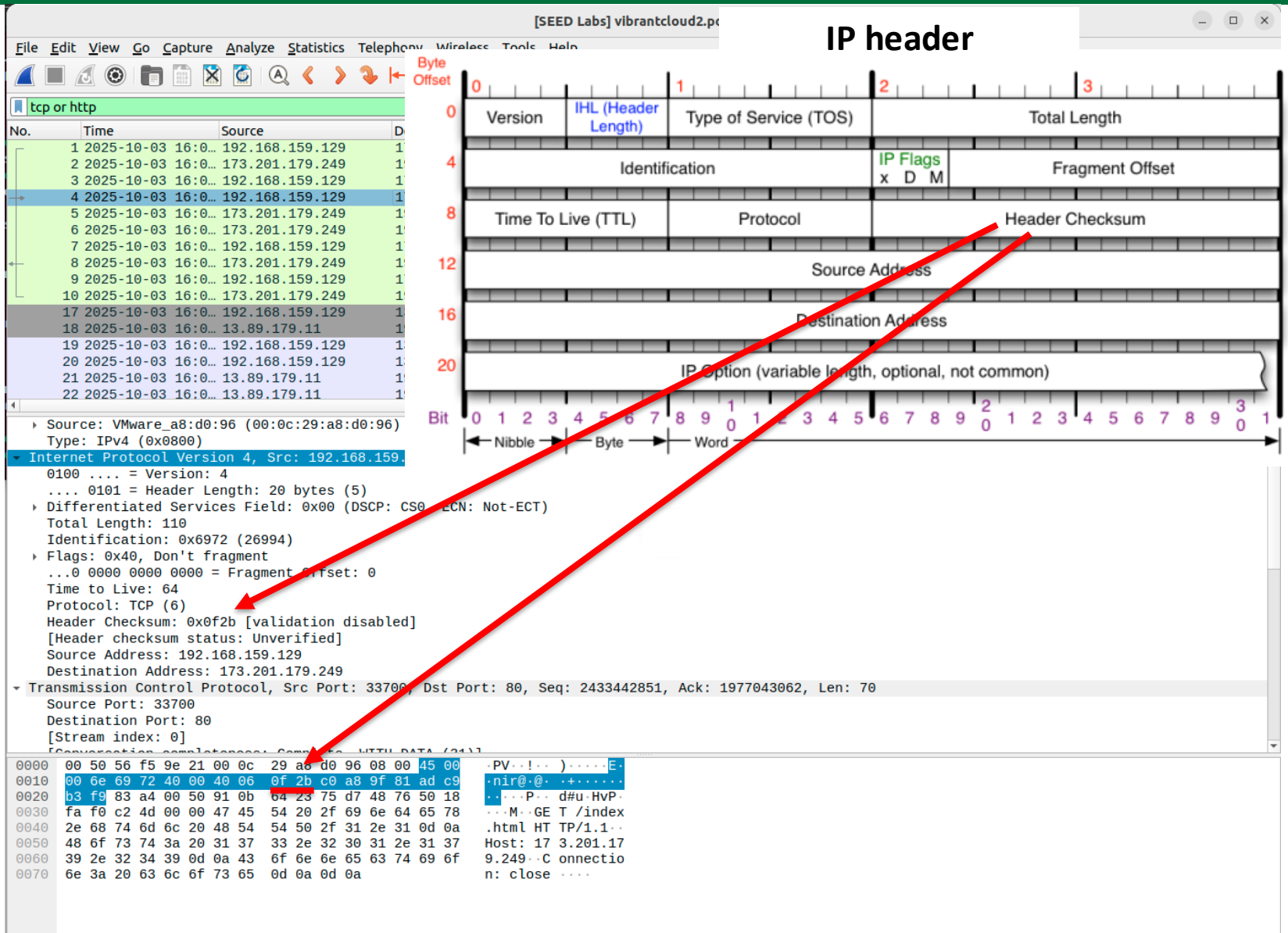
Layer 3: IP header

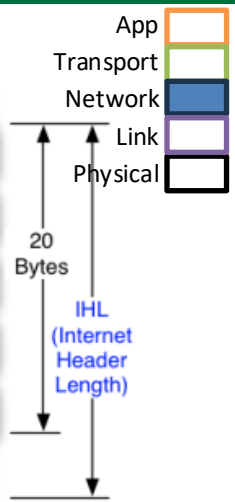


Layer 3: IP header

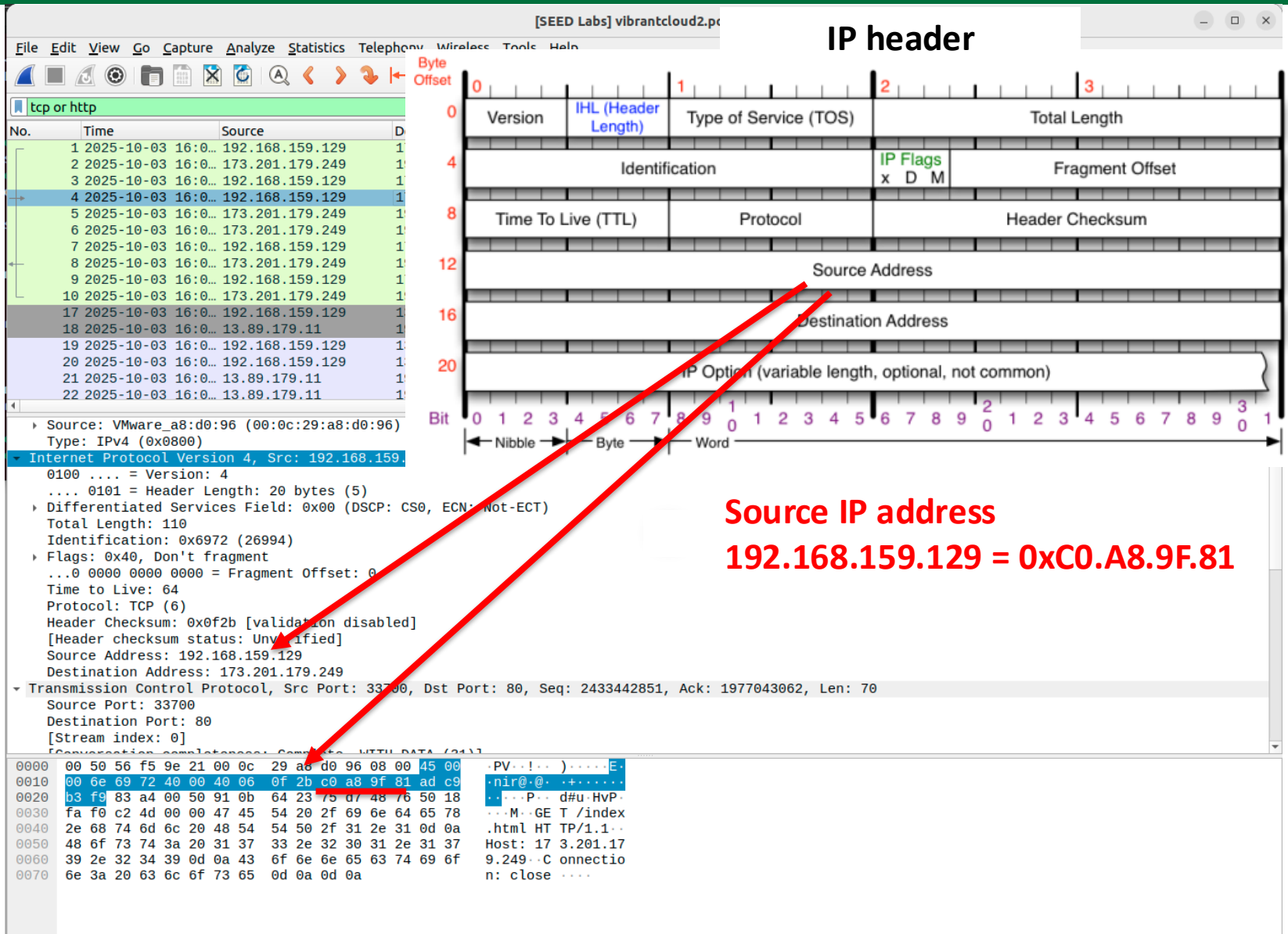


Layer 3: IP header

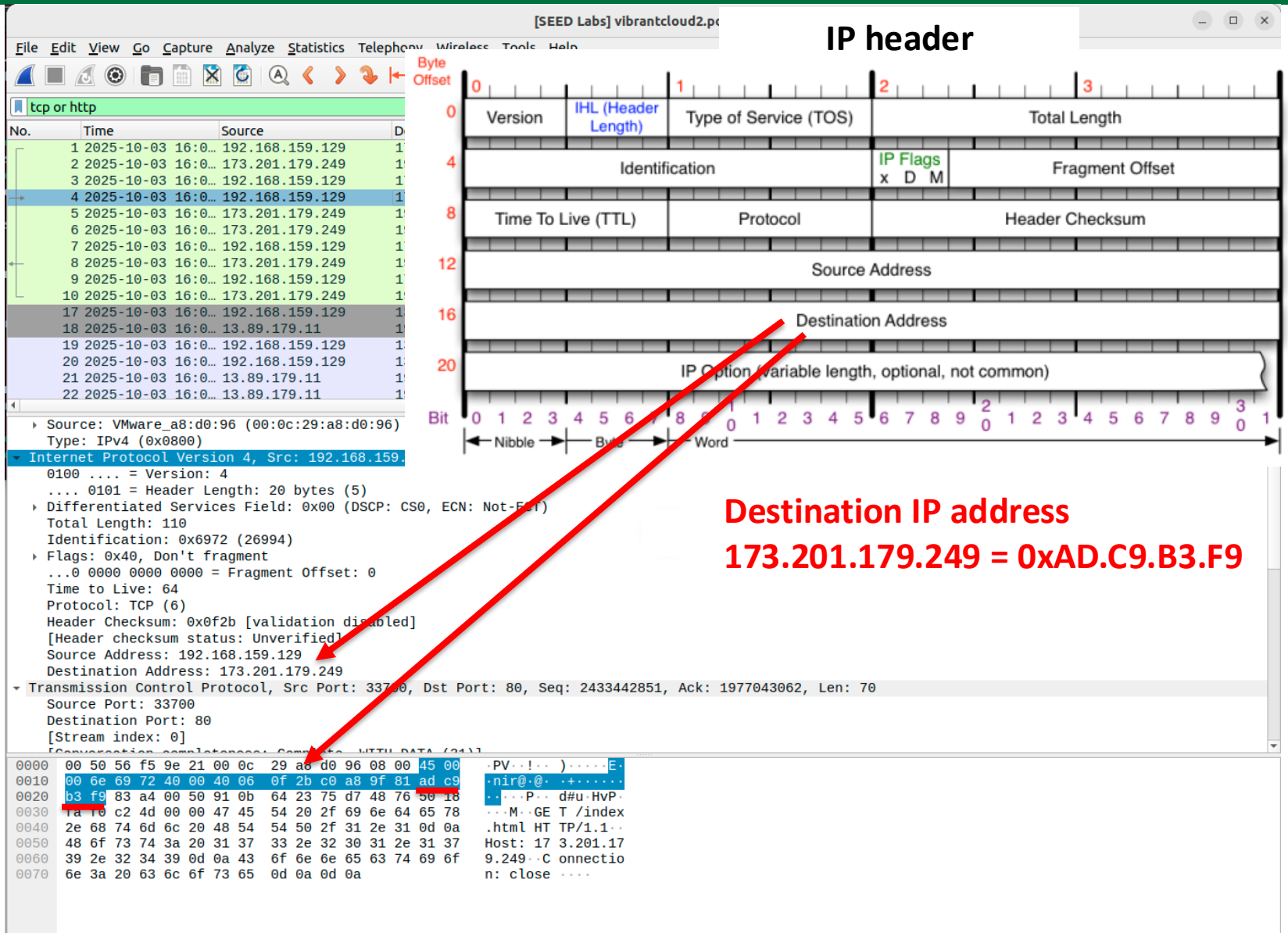




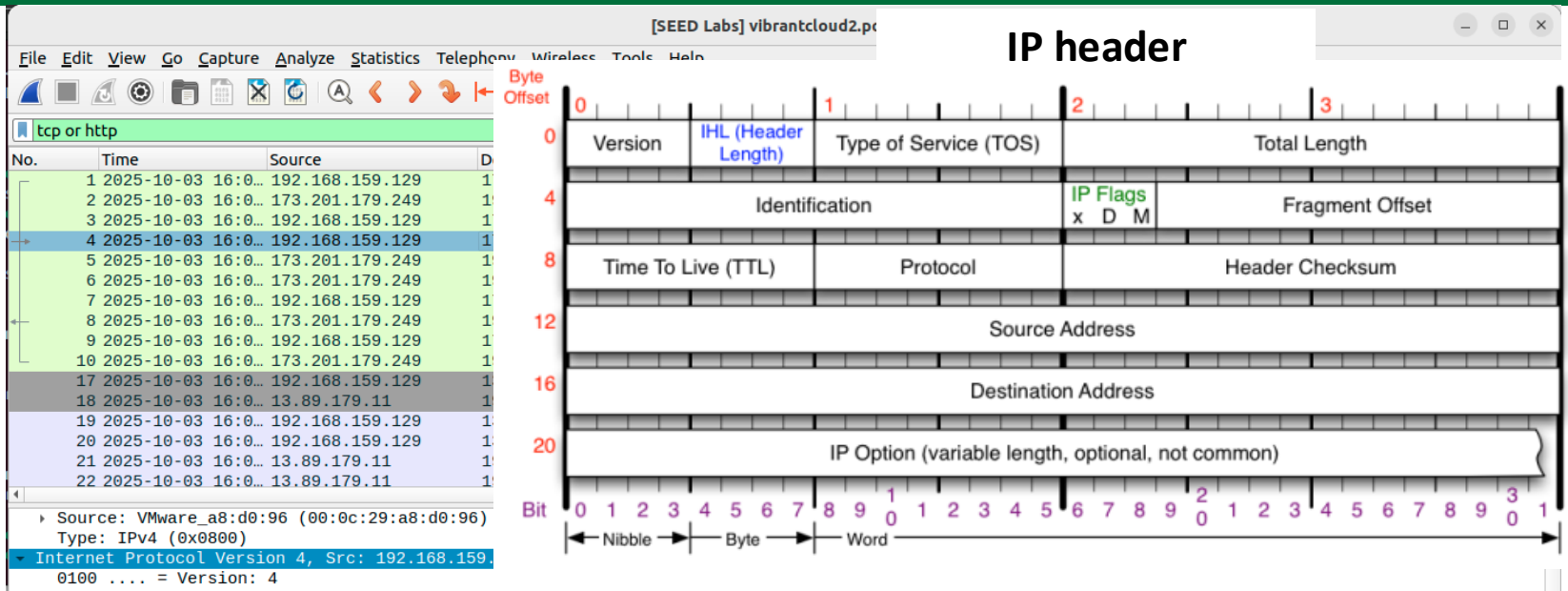
Layer 3: IP header

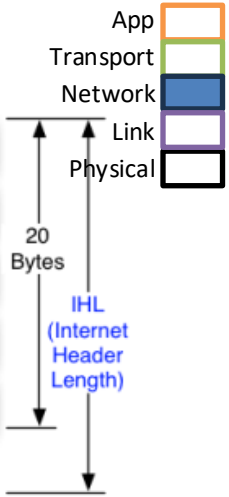


Layer 3: IP header



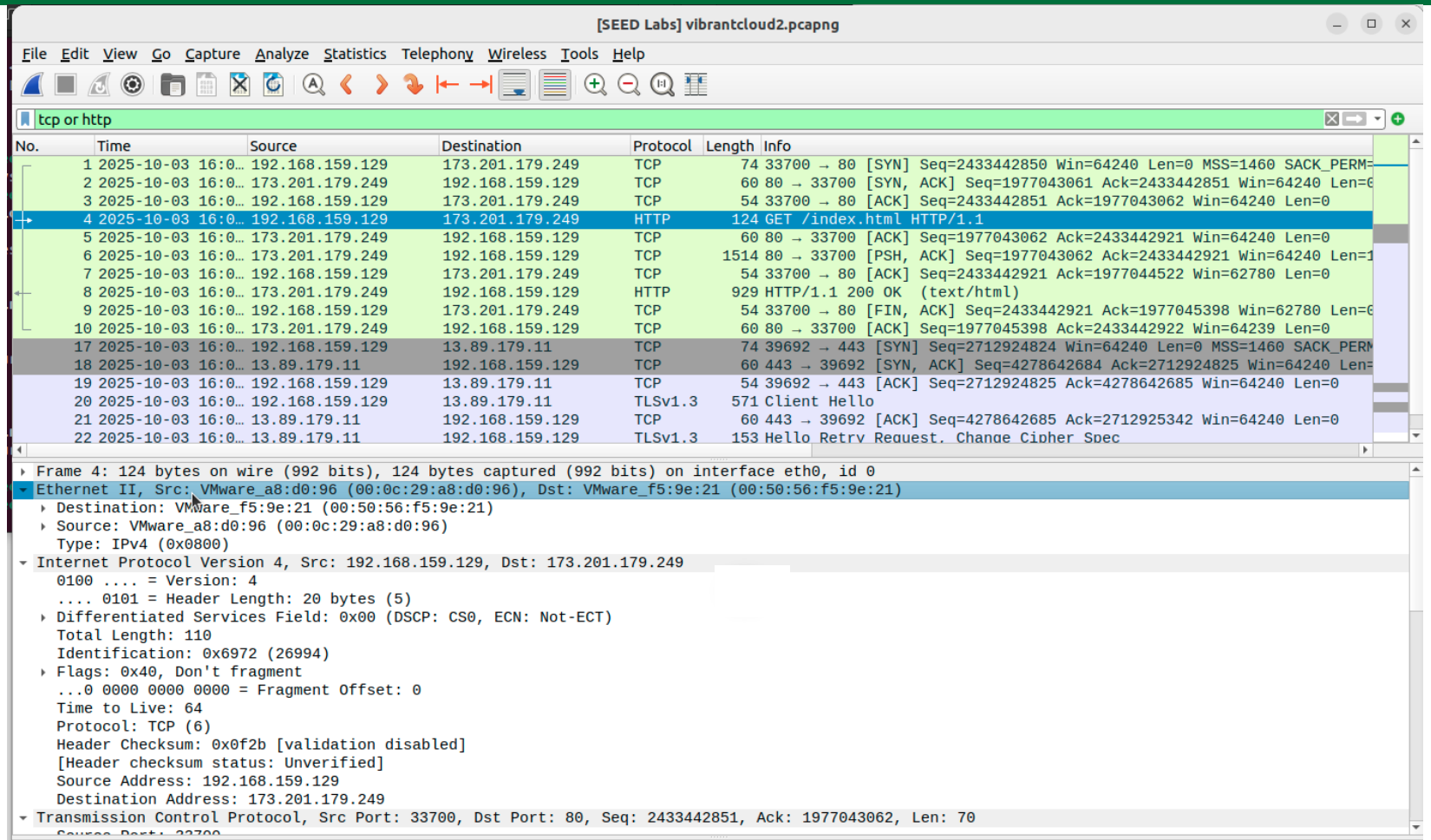
Layer 3: IP header





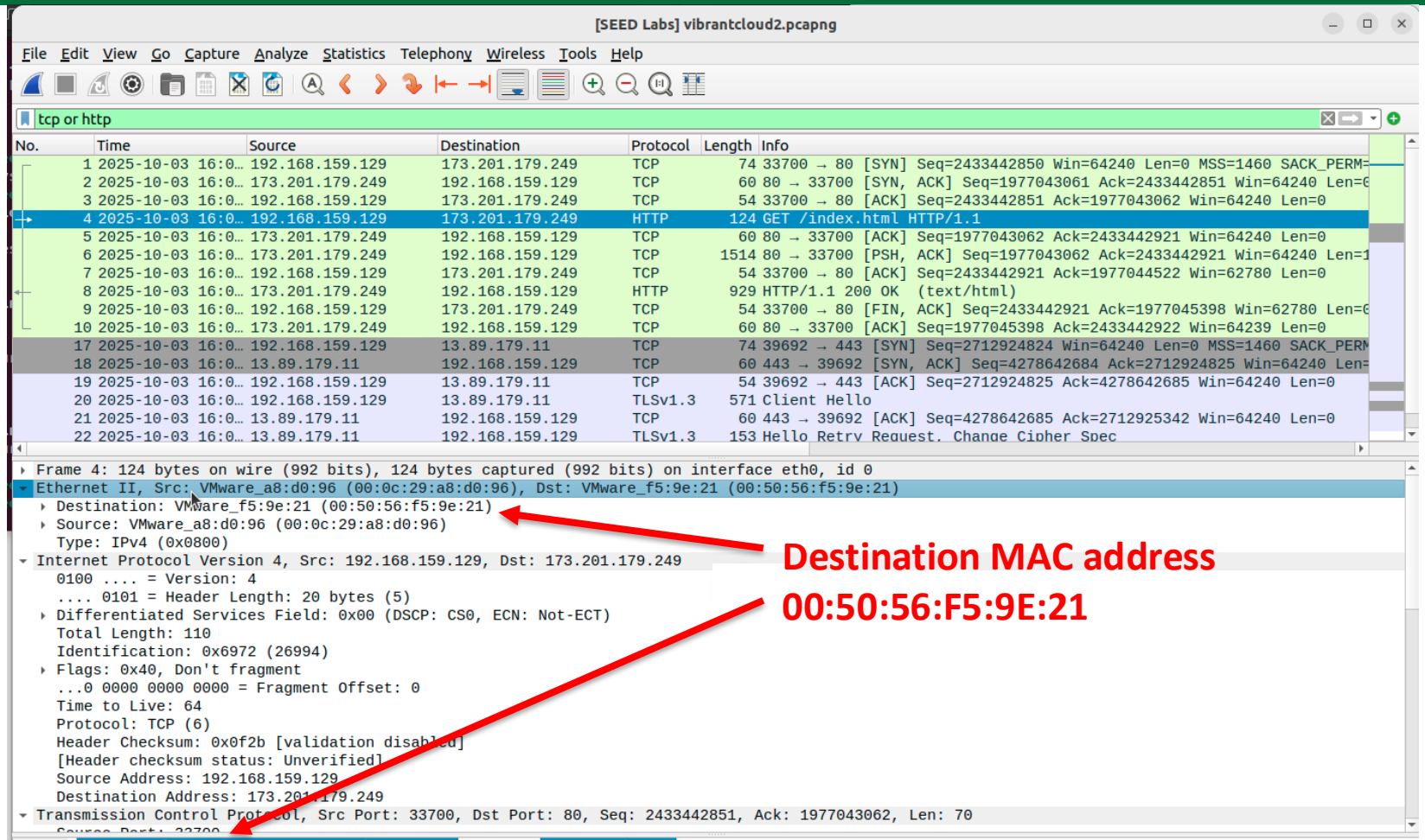
No optional fields here

Layer 2: MAC header

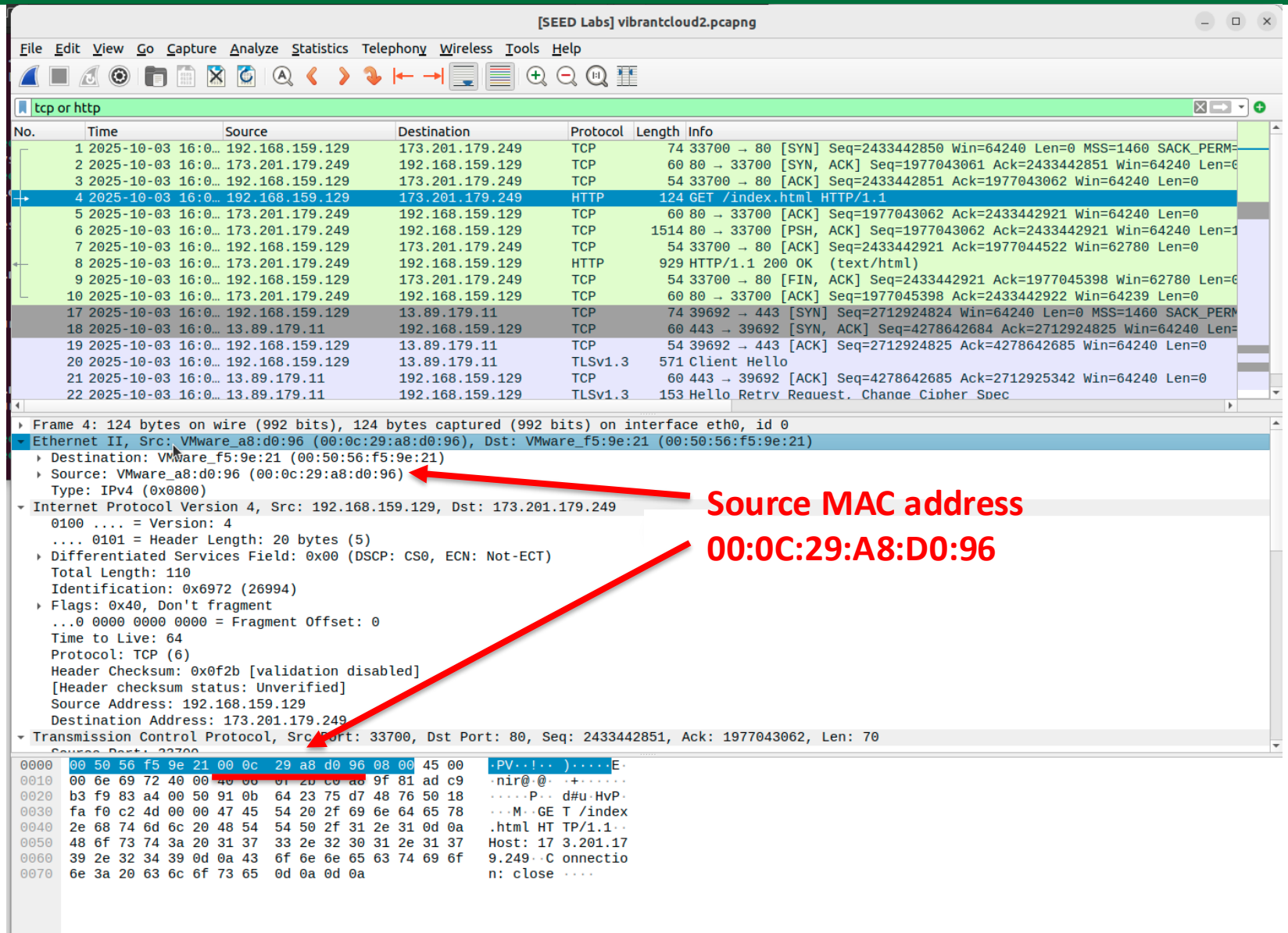


App
Transport
Network
Link
Physical

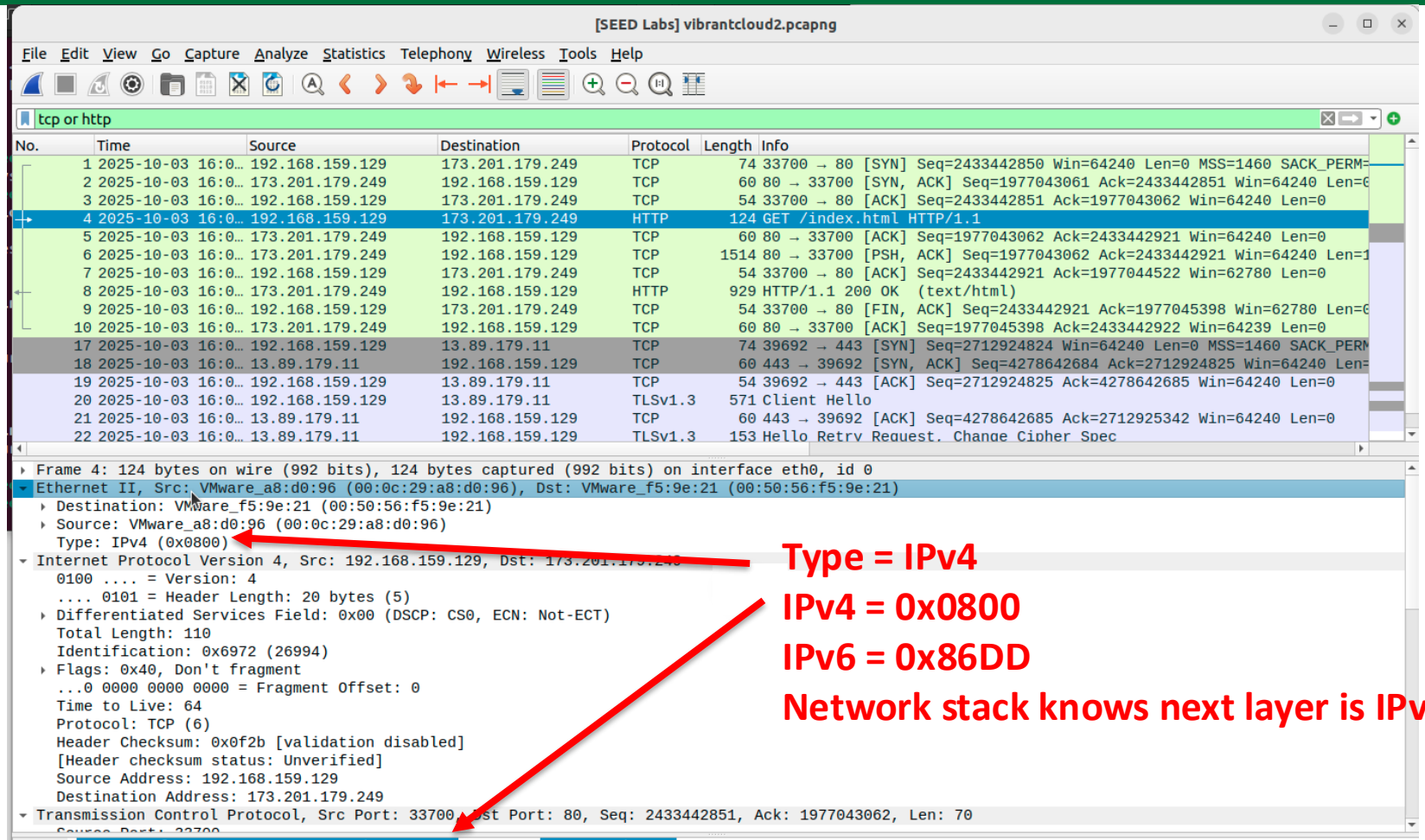
Layer 2: MAC header



Layer 2: MAC header



Layer 2: MAC header



Layer 2: MAC header

