IERG 4130 – 2026 Fall

香港中文大學
CUHK

Introduction to Networking

Yajin ZHOU

https://yajin.org

Credits: the slides are from CS161 of UC Berkeley

Final exam

- Two A4 notes are allowed (double side, hand-writing or printed are fine)
- We have two review sessions on Dec 2 and Dec 3. You only need to attend one
- Please be well-prepared for the final exam

What's the Internet?



- **Network: A set of connected machines that can communicate with each other**
  - Machines on the network agree on a **protocol**, a set of rules for communication
- **Internet: A global network of computers**
  - The web sends data between browsers and servers using the Internet
  - The Internet can be used for more than the web (e.g., SSH)

Protocols

- A **protocol** is an agreement on how to communicate that specifies syntax and semantics
  - *Syntax:* How a communication is specified and structured (format, order of messages)
  - *Semantics*: What a communication means (actions taken when sending/receiving messages)
- **Example: Protocol for asking a question in lecture?**
  1. **The student should raise their hand**
  2. The student should wait to be called on by the speaker or wait for the speaker to pause
  3. **The student should speak the question after being called on or after waiting**
  4. **If the student has been unrecognized after some time: Vocalize with “Excuse me!”**



Internet Analogy: Mail

What's the goal of the Internet?



- **Move data from one location to another**
- Analogy: I write a message on a piece of paper. How do I send this message to you?
- **Solution: Postal system**

Building block 1: Something That Moves Data

- **Mailman, Pony Express, carrier pigeon, etc.**

IP over Avian Carriers

From Wikipedia, the free encyclopedia

In computer networking, **IP over Avian Carriers** (**IPoAC**) is a proposal to carry Internet Protocol (IP) traffic by birds such as homing pigeons. IP over Avian Carriers was initially described in RFC 1149, a Request for Comments (RFC) issued by the Internet Engineering Task Force (IETF), written by D. Waitzman, and released on April 1, 1990. It is one of several April Fools' Day Request for Comments.

Waitzman described an improvement of his protocol in RFC 2549, *IP over Avian Carriers with Quality of Service* (1 April 1999). Later, in RFC 6214—released on 1 April 2011, and 13 years after the introduction of IPv6—Brian Carpenter and Robert Hinden published *Adaptation of RFC 1149 for IPv6*.[1]

IPoAC has been successfully implemented, but for only nine packets of data, with a packet loss ratio of 55% (due to operator error),[2] and a response time ranging from 3,000 seconds (≈50 minutes) to over 6,000 seconds (≈1.77 hours). Thus, this technology suffers from poor latency. Nevertheless, for large transfers, avian carriers are capable of high average throughput when carrying flash memory devices, effectively implementing a sneakernet. During the last 20 years, the information density of storage media and thus the bandwidth of an avian carrier has increased 3 times as fast as the bandwidth of the Internet.[3] IPoAC may achieve bandwidth peaks of orders of magnitude more than the Internet when used with multiple avian carriers in rural areas. For example: If 16 homing pigeons are given eight 512 GB SD cards each, and take an hour to reach their destination, the throughput of the transfer would be 145.6 Gbit/s, excluding transfer to and from the SD cards.



Under RFC 1149, a homing pigeon (exemplar in Scheßlitz) can carry Internet Protocol traffic.

Are pigeons faster than the Internet?

Building block 1: Something That Moves Data

- The Internet is built on technology that moves bits across space
- Voltages on wires, wireless technology, radio waves, etc.

Risks [edit]

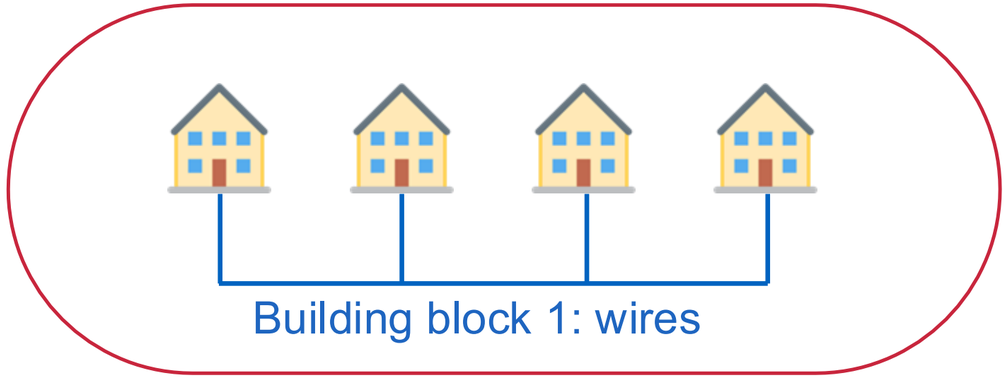
Although collisions are unlikely, packets can be lost, particularly to raptors.



An example of packet loss.

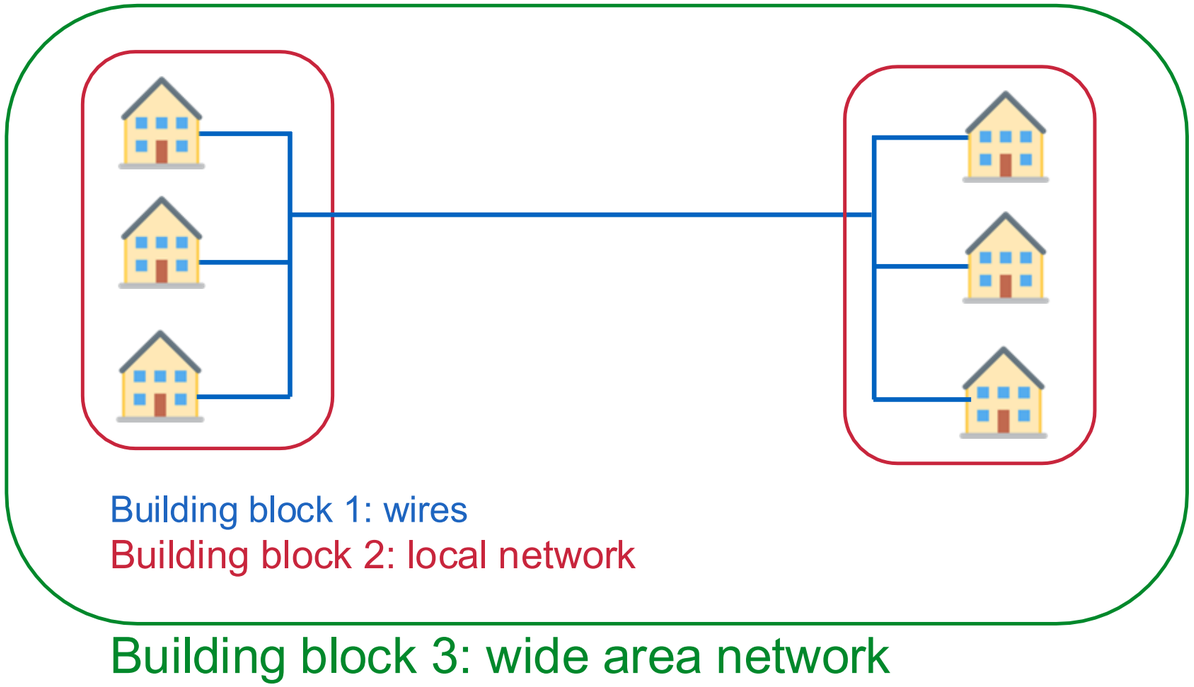
Building block 2: Talking to the Apartment Complex

- **Using building block 1, we can link up people within a local apartment complex**
- **Forms a local area network (LAN)**



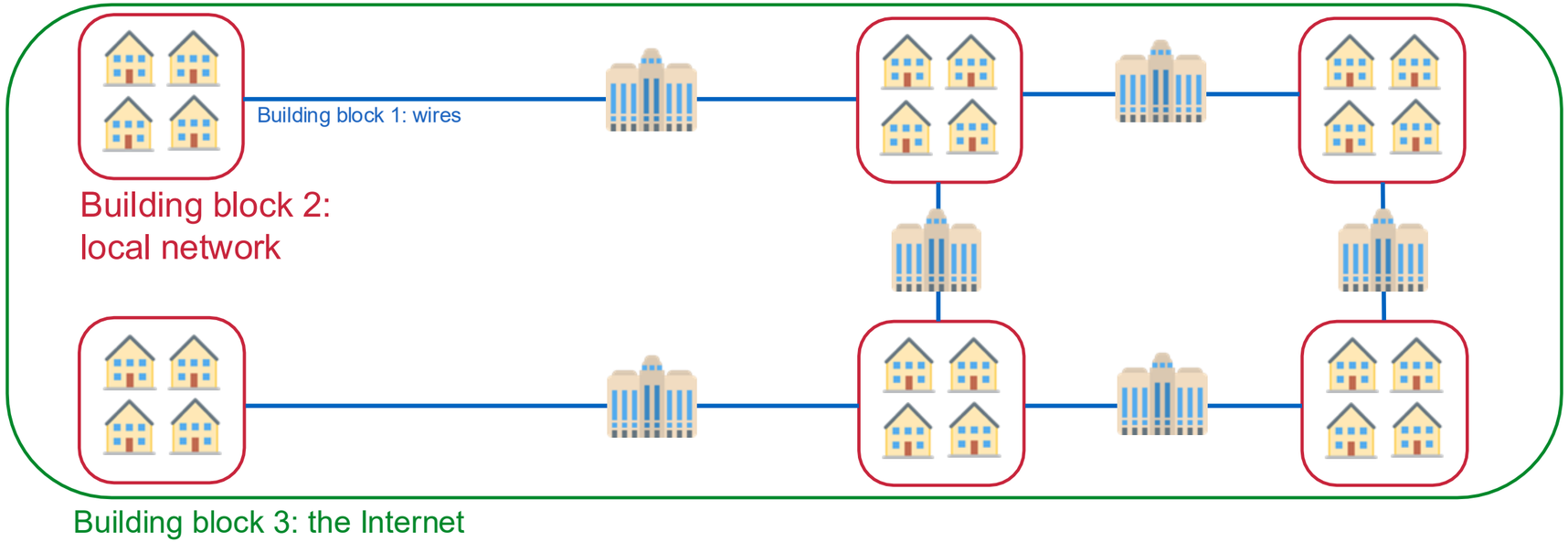
Building block 3: Post offices

- A post office connects two or more apartment complexes
- Forms a wide area network



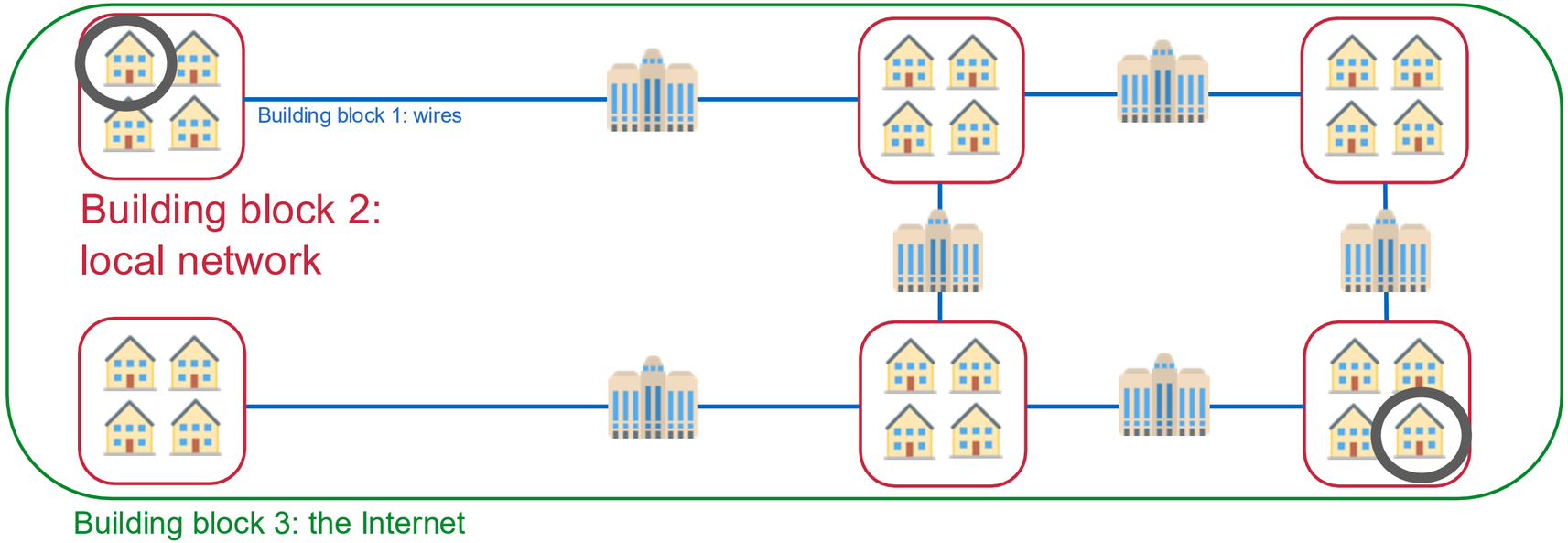
Building block 3: The Internet

- Connect the entire world using post offices
- Messages may pass through multiple post offices before reaching destination



Building block 3: The Internet

- Connect the entire world using post offices
- Messages may pass through multiple post offices before reaching destination



Layers of abstraction



Layer 3: Connect many local networks to form a global network

Layer 2: Create links in a local area

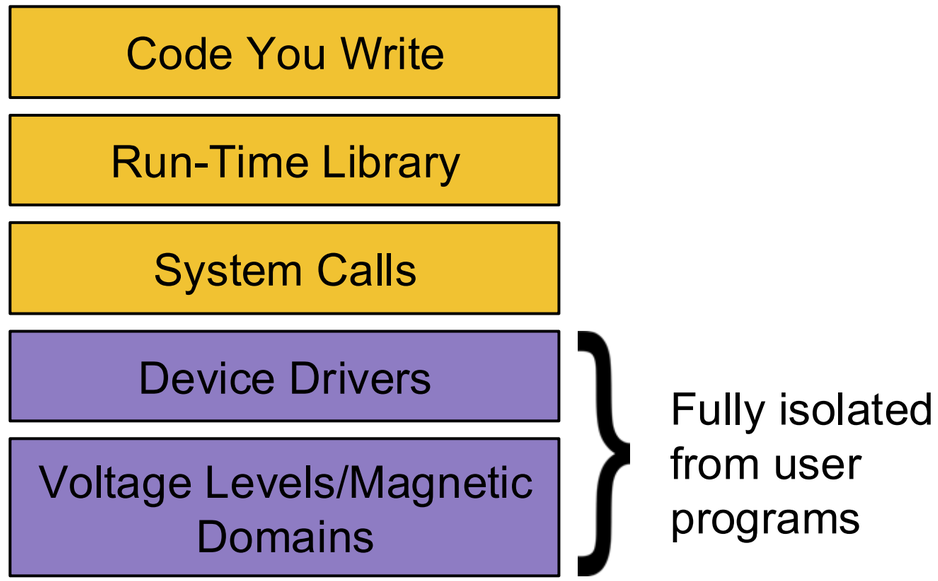
Layer 1: Move bits across space

- A change in layer 1 implementation (wireless instead of wires) doesn't affect the other layers
- **A change in layer 2 protocols doesn't affect the other layers**

Layering: The OSI Model

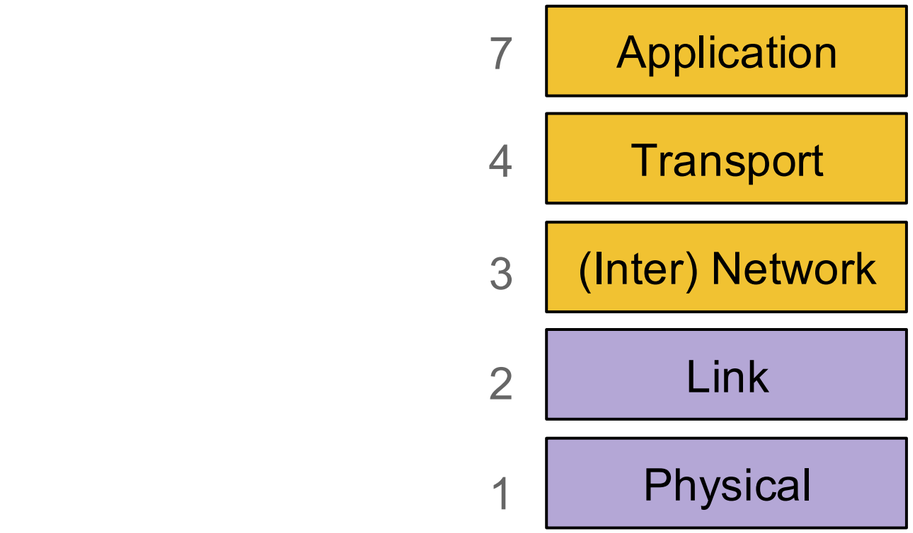
Layering

- **Internet design is partitioned into various layers. Each layer…**
  - Has a protocol
  - Relies on services provided by the layer below it
  - Provides services to the layer above it
- Analogous to the structure of an application and the “services” that each layer relies on and provides



OSI Model

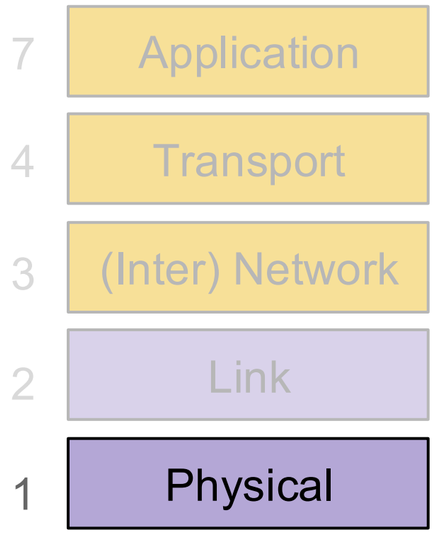
- **OSI model:** Open Systems Interconnection model, a layered model of Internet communication
  - Originally divided into 7 layers
    - But layers 5 and 6 aren't used in the real world, so we ignore them
- **Same reliance upon abstraction**
  - A layer can be implemented in different ways without affecting other layers
  - A layer's protocol can be substituted with another protocol without affecting other layers





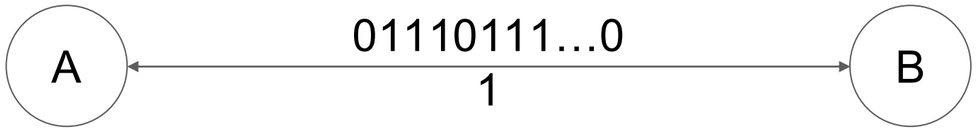
Layer 1: Physical Layer

- **Provides: Sending bits from one device to another**
  - Encodes bits to send them over a physical link
    - Patterns of voltage levels
    - Photon intensities
    - RF modulation
- **Examples**
  - Wi-Fi radios (IEEE 802.11)
  - Ethernet voltages (IEEE 802.3)

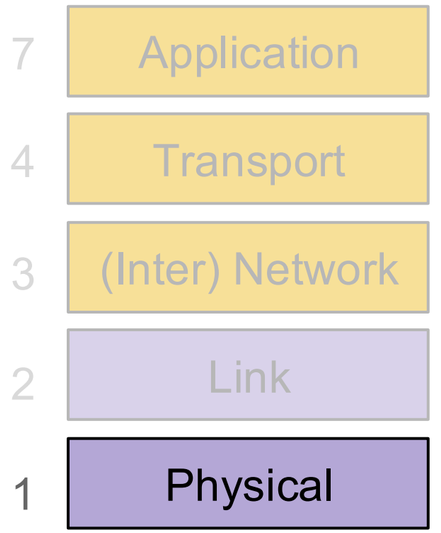


Layer 1: Physical Layer

Physical layer: “How do I transmit this sequence of 0’s and 1’s from A to B?”

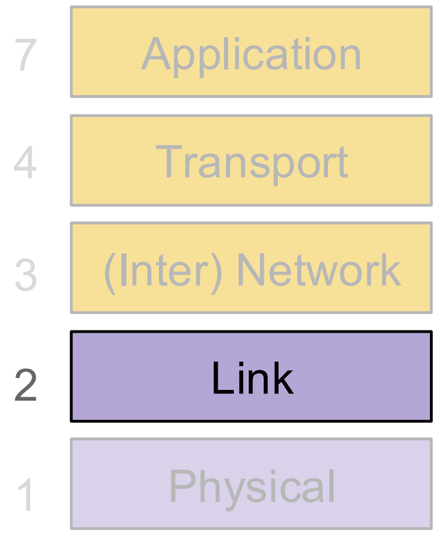


Next: How do we talk to more than one device?



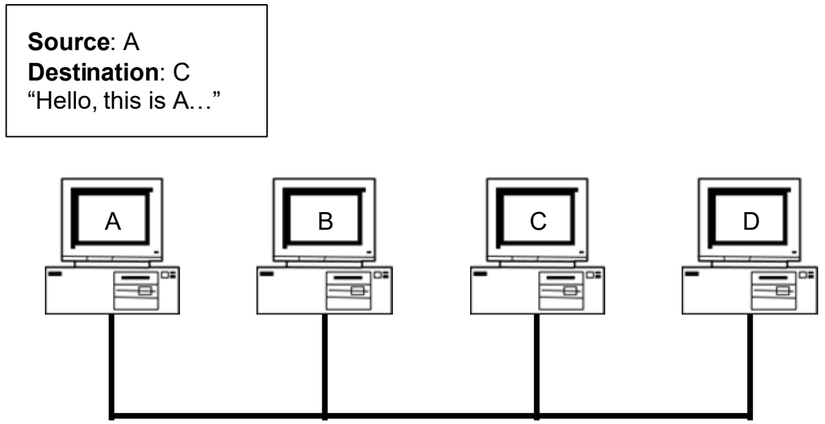
Layer 2: Link Layer

- **Provides**: Sending frames directly from one device to another (on a local network)
  - **Relies upon**: Sending bits from one device to another
  - Encodes messages into groups of bits called “frames”
- **Examples**
  - Ethernet frames (IEEE 802.3)



Layer 2: Link Layer

- **Local area network (LAN)**: A set of computers on a shared network that can directly address one another
  - Consists of multiple physical links
- **Frames must consist of at least 3 things:**
  - Source (“Who is this message coming from?”)
  - Destination (“Who is this message going to?”)
  - Data (“What does this message say?”)



Ethernet and MAC Addresses



- **Ethernet: A common layer 2 protocol that most endpoint devices use**
- **MAC address**: A 6-byte address that identifies a piece of network equipment (e.g., your phone's Wi-Fi antenna)
  - Typically represented as 6 hex bytes: **13:37:ca:fe:f0:0d**
  - The first 3 bytes are assigned to manufacturers (i.e. who made the equipment)
    - This is useful in identifying a device
  - The last 3 bytes are device-specific

Ethernet II header, real bytes (frame 4 of http-get.pcap)

0	1	2	3	4	5	6	7	8	9	10	11	12	13
6c	36	ff	4a	32	92	20	61	84	3a	a9	52	08	00

Destination MAC (6 bytes)
6c:36:ff:4a:32:92 — next hop: the router

Source MAC (6 bytes)
20:61:84:3a:a9:52 — Alice's network card

Type (2 bytes)
0x0800 = IPv4

First 3 bytes of a MAC = OUI (manufacturer ID). Lowest bit of byte 0 = 0 → unicast, 1 → multicast/broadcast.
On the wire the destination comes first, so a switch can start forwarding as soon as it has read 6 bytes.

Layer 2 He

http-get.pcap — Wireshark-style view (filter: tcp.stream eq 0)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	1.2.3.4	5.6.7.8	TCP	66	1234 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
2	0.000579	5.6.7.8	1.2.3.4	TCP	66	80 → 1234 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460
3	0.001006	1.2.3.4	5.6.7.8	TCP	54	1234 → 80 [ACK] Seq=1 Ack=1 Win=502 Len=0
4	0.001438	1.2.3.4	5.6.7.8	HTTP	124	GET / HTTP/1.1
5	0.001925	5.6.7.8	1.2.3.4	TCP	54	80 → 1234 [ACK] Seq=1 Ack=71 Win=509 Len=0
6	0.002336	5.6.7.8	1.2.3.4	TCP	1514	80 → 1234 [ACK] Seq=1 Ack=71 Len=1460 [TCP segment of a reassembled PDU]
7	0.002927	5.6.7.8	1.2.3.4	TCP	1514	80 → 1234 [ACK] Seq=1461 Ack=71 Len=1460 [TCP segment of a reassembled PDU]
8	0.003405	5.6.7.8	1.2.3.4	HTTP	797	HTTP/1.1 200 OK (text/html)
9	0.004037	1.2.3.4	5.6.7.8	TCP	54	1234 → 80 [ACK] Seq=71 Ack=3664 Win=502 Len=0

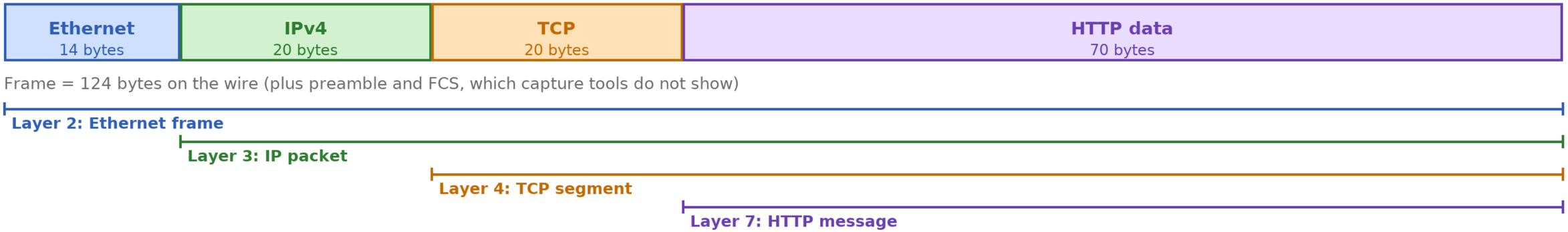
```
▸ Frame 4: 124 bytes on wire (992 bits), 124 bytes captured
▾ Ethernet II, Src: 20:61:84:3a:a9:52, Dst: 6c:36:ff:4a:32:92
    Destination: 6c:36:ff:4a:32:92
    Source: 20:61:84:3a:a9:52
    Type: IPv4 (0x0800)
▾ Internet Protocol Version 4, Src: 1.2.3.4, Dst: 5.6.7.8
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 110
    Identification: 0x1c45 (7237)
    010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: TCP (6)
    Header Checksum: 0x0e32 [correct]
    Source Address: 1.2.3.4
    Destination Address: 5.6.7.8
▾ Transmission Control Protocol, Src Port: 1234, Dst Port: 80, Seq: 1, Ack: 1, Len: 70
    Source Port: 1234
    Destination Port: 80
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 3052814922
    Acknowledgment Number: 1    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
    Flags: 0x018 (PSH, ACK)
    Window: 502
    Checksum: 0x0073 [correct]
    Urgent Pointer: 0
▾ Hypertext Transfer Protocol
    GET / HTTP/1.1\r\n
    Host: 5.6.7.8\r\n
    User-Agent: curl/8.5.0\r\n
    Accept: */*\r\n
    \r\n
```

```
0000  6c 36 ff 4a 32 92 20 61  84 3a a9 52 08 00 45 00   l6·J2· a·:·R··E·
0010  00 6e 1c 45 40 00 40 06  0e 32 01 02 03 04 05 06   ·n·E@·@··2······
0020  07 08 04 d2 00 50 b5 f6  42 4a 71 80 30 d6 50 18   ·····P··BJq·0·P·
0030  01 f6 00 73 00 00 47 45  54 20 2f 20 48 54 54 50   ···s··GET / HTTP
0040  2f 31 2e 31 0d 0a 48 6f  73 74 3a 20 35 2e 36 2e   /1.1··Ho st: 5.6.
0050  37 2e 38 0d 0a 55 73 65  72 2d 41 67 65 6e 74 3a   7.8··Use r-Agent:
0060  20 63 75 72 6c 2f 38 2e  35 2e 30 0d 0a 41 63 63    curl/8.5 .0··Acc
0070  65 70 74 3a 20 2a 2f 2a  0d 0a 0d 0a               ept: */* ····
```

- Ethernet header (14 bytes)
- IPv4 header (20 bytes)
- TCP header (20 bytes)
- HTTP data (70 bytes)

Layer 2 Header

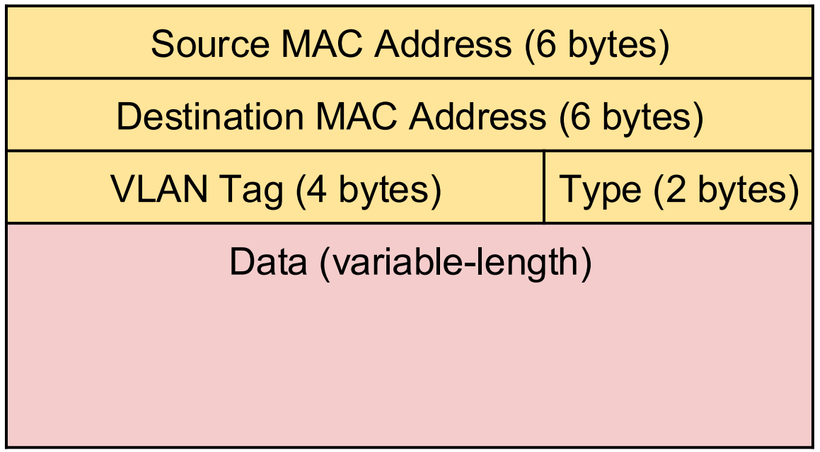
One real frame: an HTTP request wrapped in TCP, IP and Ethernet headers




```
0000  6c 36 ff 4a 32 92 20 61  84 3a a9 52 08 00 45 00   l6·J2· a·:·R··E·
0010  00 6e 1c 45 40 00 40 06  0e 32 01 02 03 04 05 06   ·n·E@·@··2······
0020  07 08 04 d2 00 50 b5 f6  42 4a 71 80 30 d6 50 18   ·····P··BJq·0·P·
0030  01 f6 00 73 00 00 47 45  54 20 2f 20 48 54 54 50   ···s··GET / HTTP
0040  2f 31 2e 31 0d 0a 48 6f  73 74 3a 20 35 2e 36 2e   /1.1··Host: 5.6.
0050  37 2e 38 0d 0a 55 73 65  72 2d 41 67 65 6e 74 3a   7.8··User-Agent:
0060  20 63 75 72 6c 2f 38 2e  35 2e 30 0d 0a 41 63 63    curl/8.5.0··Acc
0070  65 70 74 3a 20 2a 2f 2a  0d 0a 0d 0a               ept: */*····
```

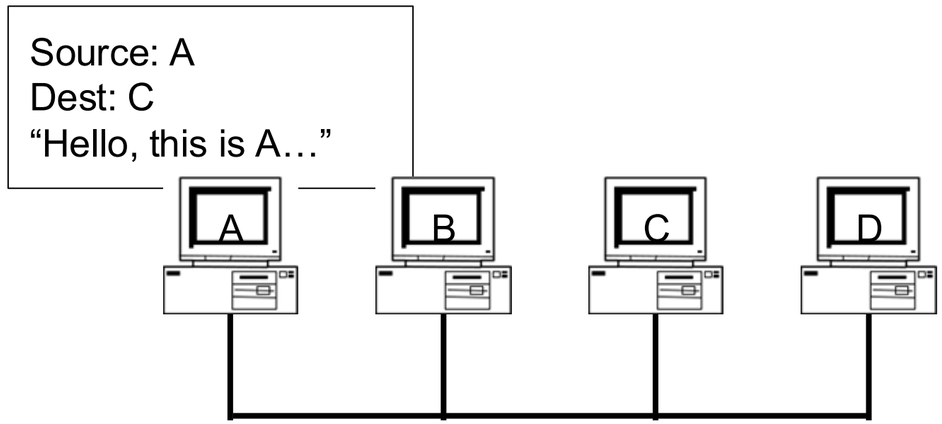
```
6c 36 ff 4a 32 92  dst MAC (router)
20 61 84 3a a9 52  src MAC (Alice)
08 00  type = IPv4
45  version 4, header 20 B
40  TTL = 64,  06  protocol = TCP
01 02 03 04  src IP 1.2.3.4
05 06 07 08  dst IP 5.6.7.8
04 d2  src port 1234
00 50  dst port 80
47 45 54 20 …  "GET " in ASCII
```

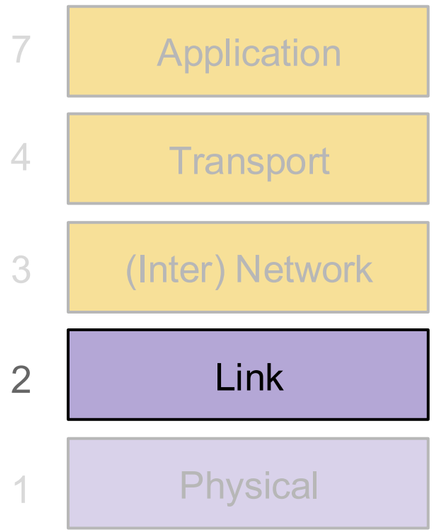
Layer 2 Header



Layer 2: Link Layer

Link layer: “How do I transmit this frame from A to C, making sure that no one else thinks the message is for them?”

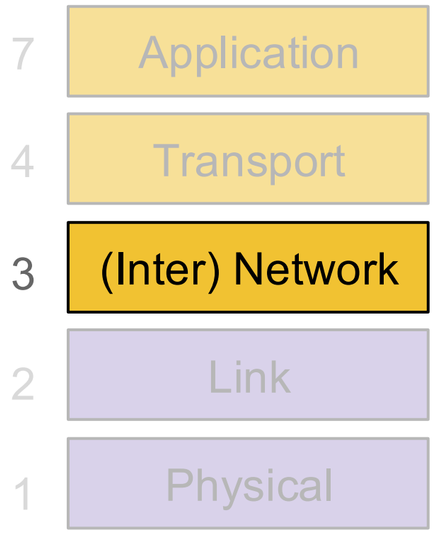




Next: How do we address every device in existence?

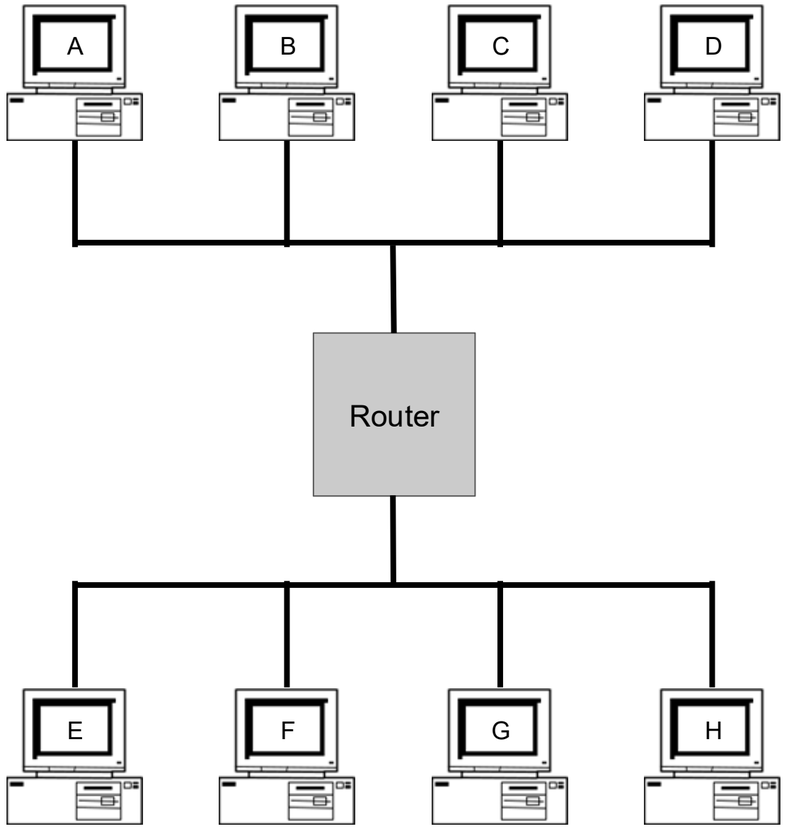
Layer 3: Network Layer

- **Provides: Sending packets from any device to any other device**
  - **Relies upon**: Sending frames directly from one device to another
  - Encodes messages into groups of bits called “packets”
  - Bridges multiple LANs to provide global addressing
- **Examples**
  - Internet Protocol (IP)

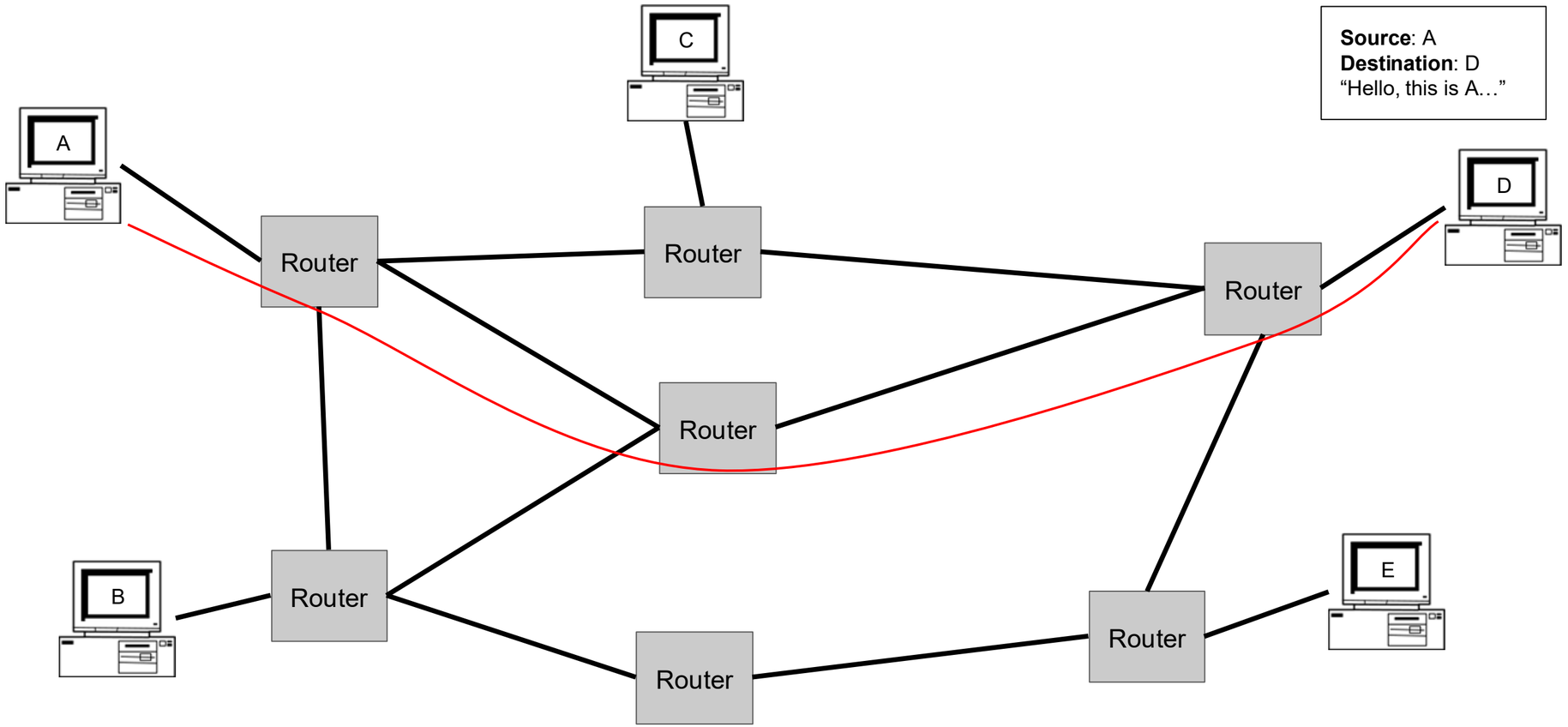


Layer 3: Network Layer

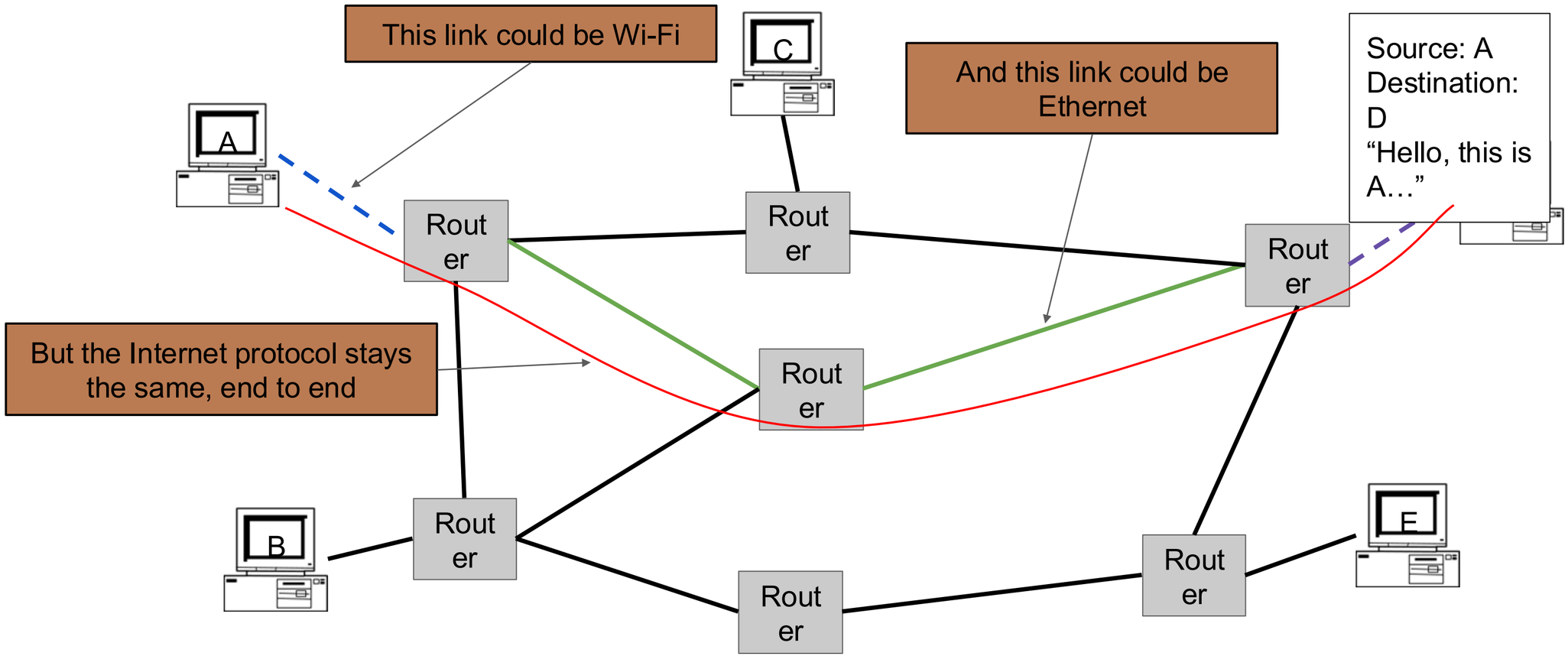
- Recall the ideal layer 2 model: All devices can directly address all other devices
  - This would not scale to the size of the Internet!
- Instead, allow packets to be **routed** across different devices to reach the destination
  - Each hop is allowed to use its own physical and link layers!



Layer 3: Network Layer



Layer 3: Network Layer



Layer 3: Network Layer



- **Packets must consist of at least 3 things:**
  - Source (“Who is this message coming from?”)
  - Destination (“Who is this message going to?”)
  - Data (“What does this message say?”)
  - Similar to frames (layer 2)
- **Packets may be fragmented into smaller packets**
  - Different links might support different maximum packet sizes
  - Up to the recipient to reassemble fragments into the original packet
- **Each router forwards a given packet to the next hop**
  - We will cover how a router knows how to forward—and attacks on it—in the future
- **Packets are not guaranteed to take a given route**
  - Two packets with the same source and destination may take different routes

Internet Protocol (IP)



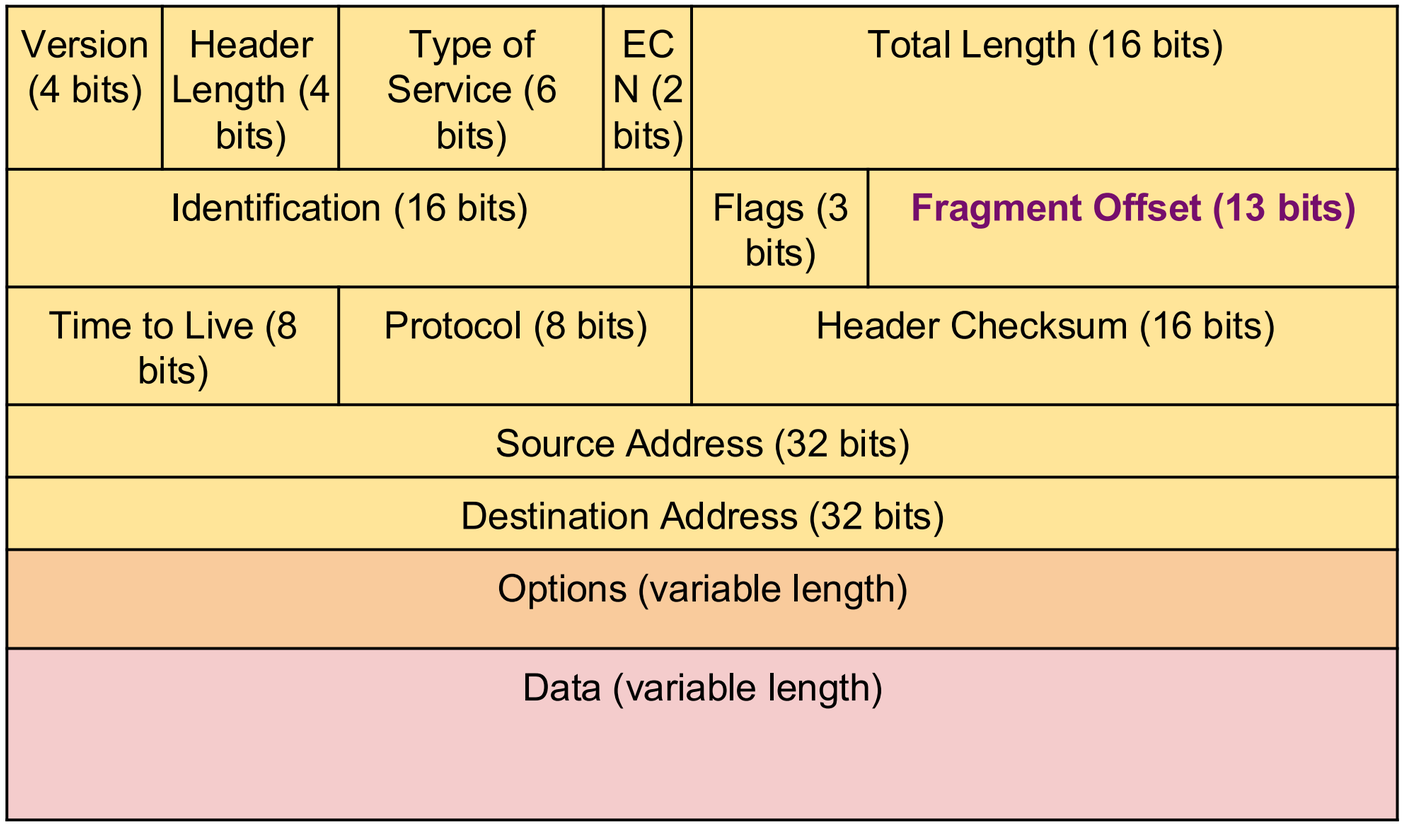
- **Internet Protocol (IP)**: The universal layer-3 protocol that all devices use to transmit data over the Internet
- **IP address: An address that identifies a device on the Internet**
  - IPv4 is 32 bits, typically written as 4 decimal octets, e.g. **35.163.72.93**
  - IPv6 is 128 bits, typically written as 8 groups of 2 hex bytes: **2607:f140:8801::1:23**
    - If digits or groups are missing, fill with 0's, so **2607:f140:8801:0000:0000:0000:0001:0023**
  - Globally unique from any single perspective
    - For now, you can think of them as just being globally unique
  - IP addresses help nodes make decisions on where to forward the packet

Reliability

- **Reliability** ensures that packets are received correctly or, if random errors occur, not at all
  - This is implemented with a checksum
  - However, there is no cryptographic MAC, so there are no guarantees if an attacker modifies packets
- **IP is *unreliable* and only provides a best effort delivery service, which means:**
  - Packets may be lost (“dropped”)
  - Packets may be corrupted
  - Packets may be delivered out of order
- **It is up to higher level protocols to ensure that the connection is reliable**

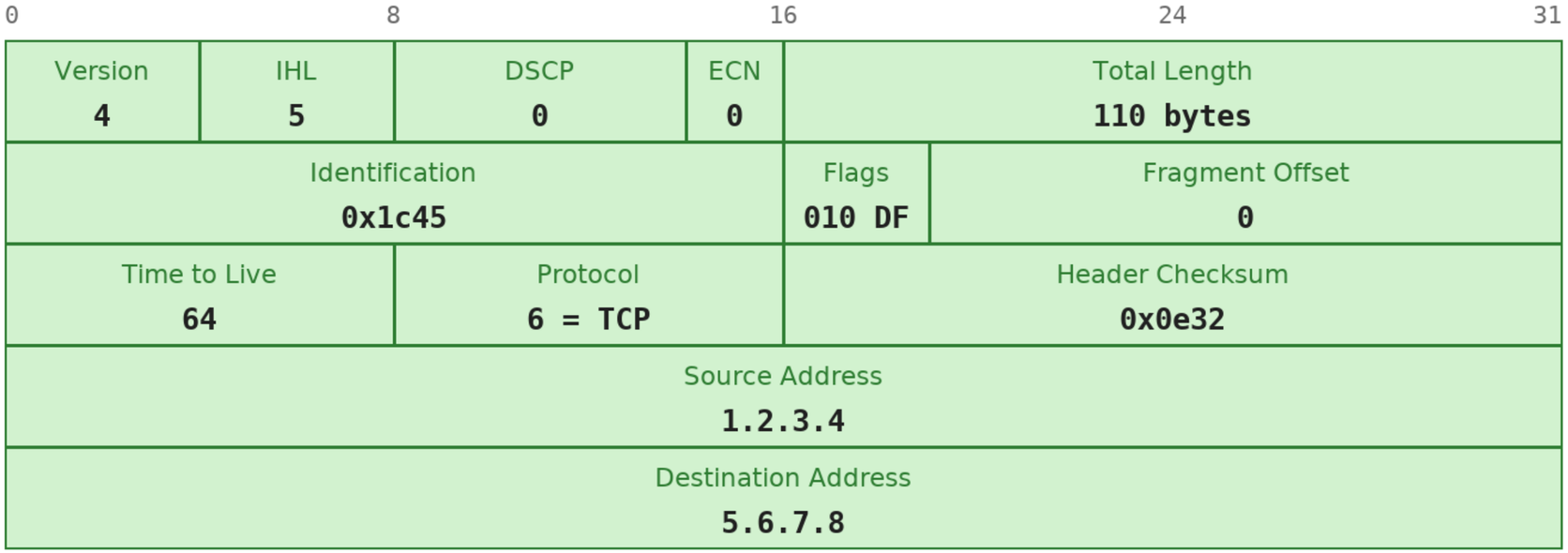
Layer 3 Header

IPv4 header



Layer 3 Header

IPv4 header, real values (frame 4 of http-get.pcap)



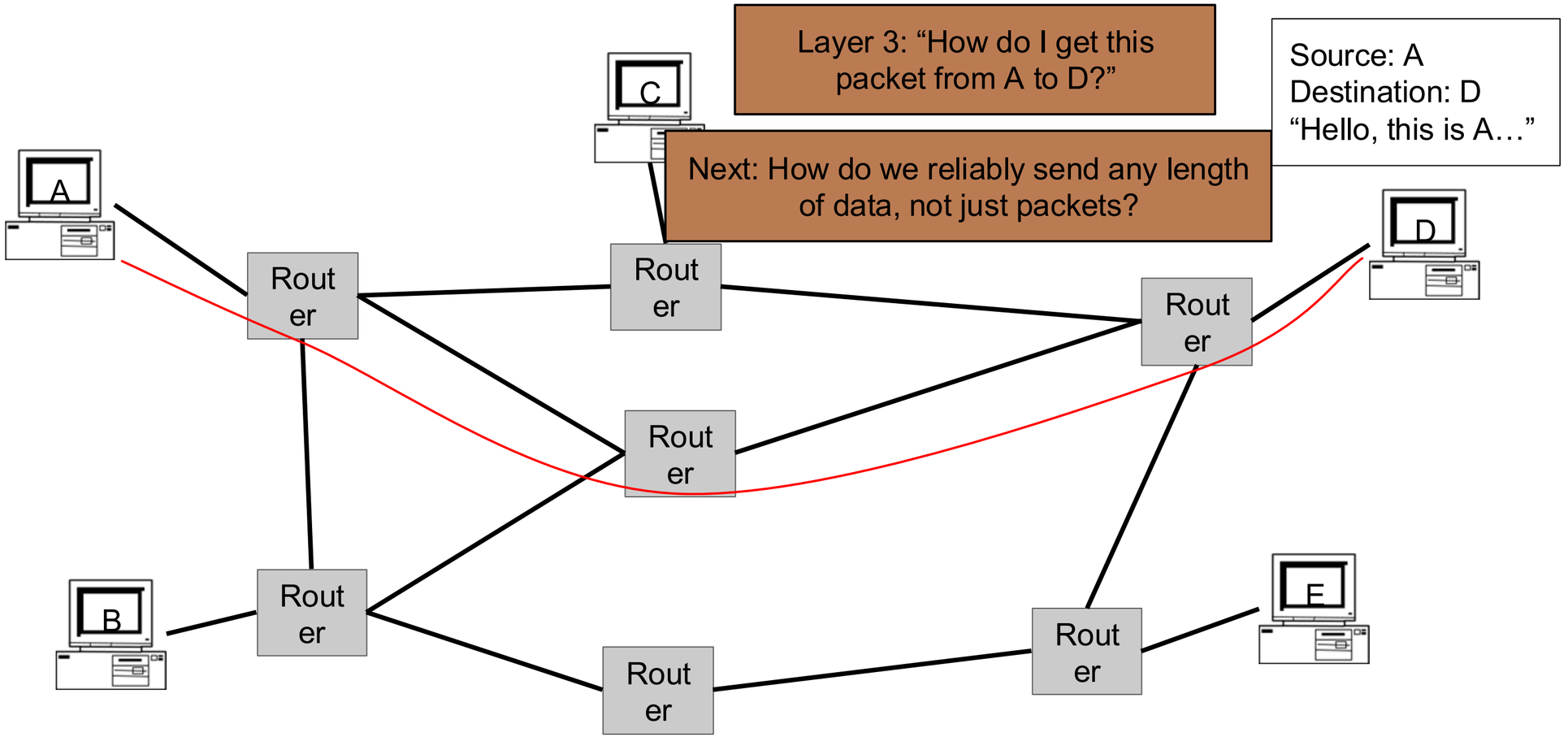
bytes 14–33 of the frame

```
45 00 00 6e
1c 45 40 00
40 06 0e 32
01 02 03 04
05 06 07 08
```

IHL = 5 means 5 × 4 = 20 bytes of header. Total Length 110 = 20 (IP header) + 20 (TCP header) + 70 (HTTP data). The Ethernet header is not counted.

Flags DF = "don't fragment"; Identification and Fragment Offset matter only when a packet is fragmented.

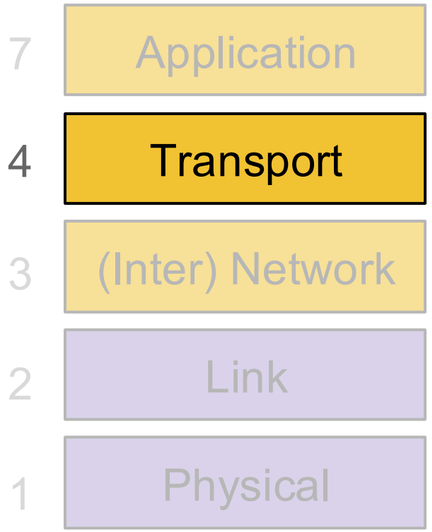
Layer 3: Network Layer



Layer 4: Transport Layer



- **Provides**: Transportation of **variable-length data** stream from any application to any other application
  - **Relies upon**: Sending packets from any device to any other device
  - Builds abstractions that are useful to applications on top of layer 3 packets
- **Useful abstractions**
  - **Reliability**: Transmit data reliably, in order
  - **Ports**: Provide multiple “addresses” per real IP address
- **Examples**
  - **TCP**: Provides reliability and ports
  - **UDP**: Provides ports, but no reliability



Layer 4: Transport Layer

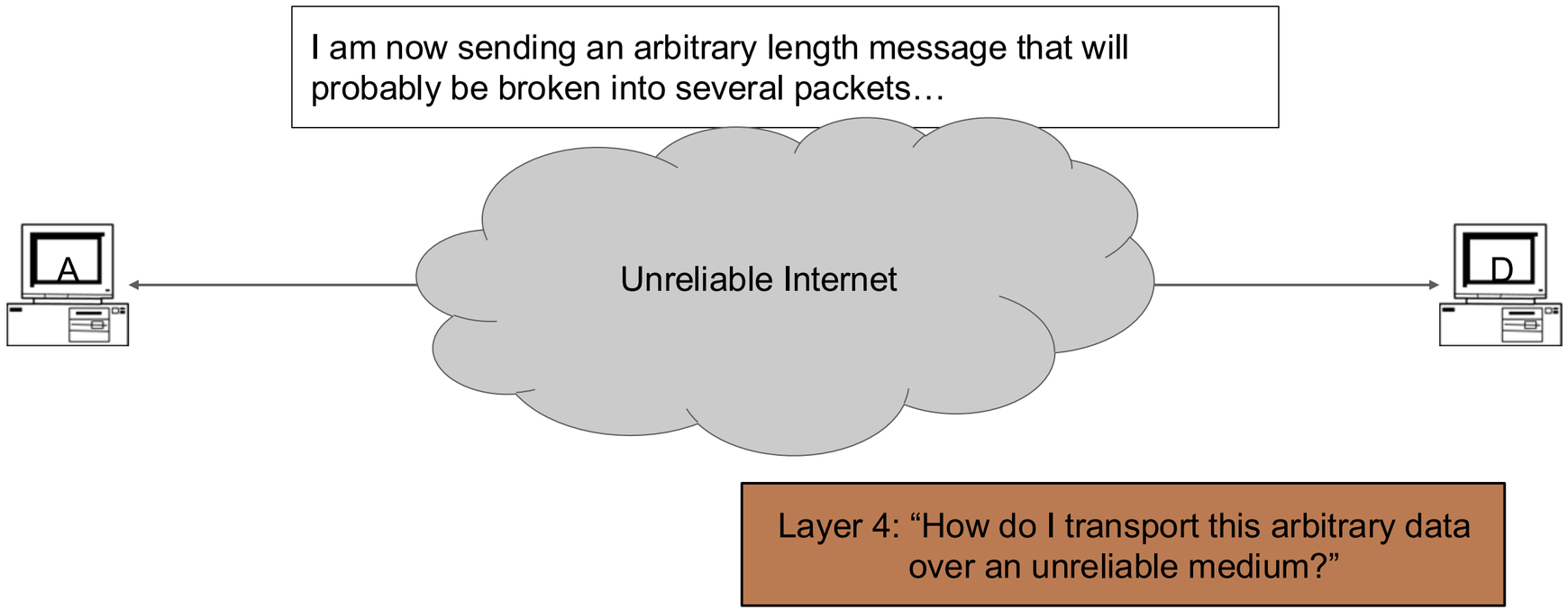
TCP header, real values (frame 4 of http-get.pcap)

0	8	16	24	31	bytes 34–53 of the frame
Source Port **1234**		Destination Port **80 (HTTP)**			**04 d2 00 50**
Sequence Number **3052814922 (relative: 1)**					**b5 f6 42 4a**
Acknowledgment Number **1904226518 (relative: 1)**					**71 80 30 d6**
Offset **5 (20 B)**	Reserved **0**	Flags **PSH ACK**	Window **502**		**50 18 01 f6**
Checksum **0x0073**		Urgent Pointer **0**			**00 73 00 00**

Sequence numbers count bytes, starting from a random initial value (ISN). Wireshark shows them relative to the ISN.

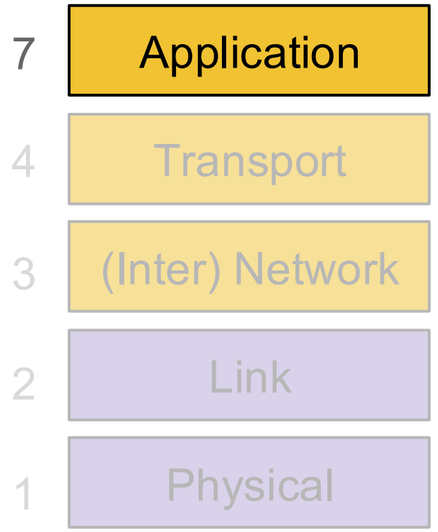
Ports identify the applications; IP addresses identify the machines.

Layer 4: Transport Layer



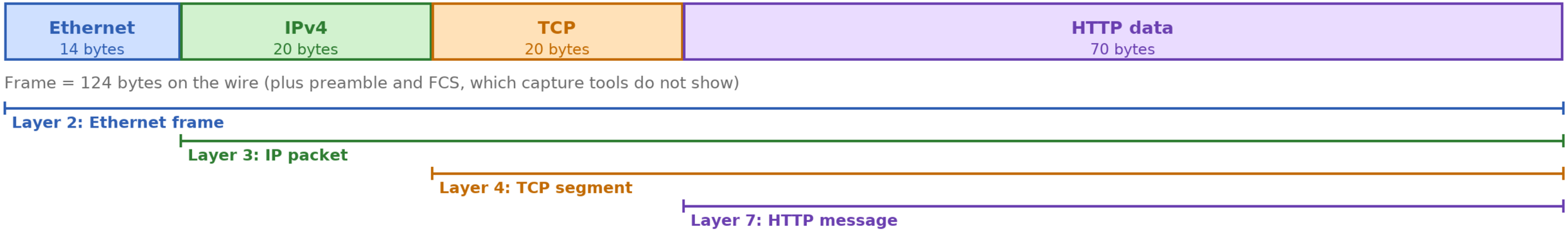
Layer 7: Application Layer

- **Provides: Applications and services to users**
  - **Relies upon**: Transportation of **variable-length data** from any application to any other application
- **Every online application is Layer 7**
  - Web browsing
  - Online video games
  - Messaging services
  - Video calls (Zoom)



Layer 7: Application Layer

One real frame: an HTTP request wrapped in TCP, IP and Ethernet headers




```
0000  6c 36 ff 4a 32 92 20 61  84 3a a9 52 08 00 45 00   l6·J2· a·:·R··E·
0010  00 6e 1c 45 40 00 40 06  0e 32 01 02 03 04 05 06   ·n·E@·@··2······
0020  07 08 04 d2 00 50 b5 f6  42 4a 71 80 30 d6 50 18   ·····P··BJq·0·P·
0030  01 f6 00 73 00 00 47 45  54 20 2f 20 48 54 54 50   ···s··GET / HTTP
0040  2f 31 2e 31 0d 0a 48 6f  73 74 3a 20 35 2e 36 2e   /1.1··Ho st: 5.6.
0050  37 2e 38 0d 0a 55 73 65  72 2d 41 67 65 6e 74 3a   7.8··Use r-Agent:
0060  20 63 75 72 6c 2f 38 2e  35 2e 30 0d 0a 41 63 63    curl/8. 5.0··Acc
0070  65 70 74 3a 20 2a 2f 2a  0d 0a 0d 0a               ept: */* ····
```

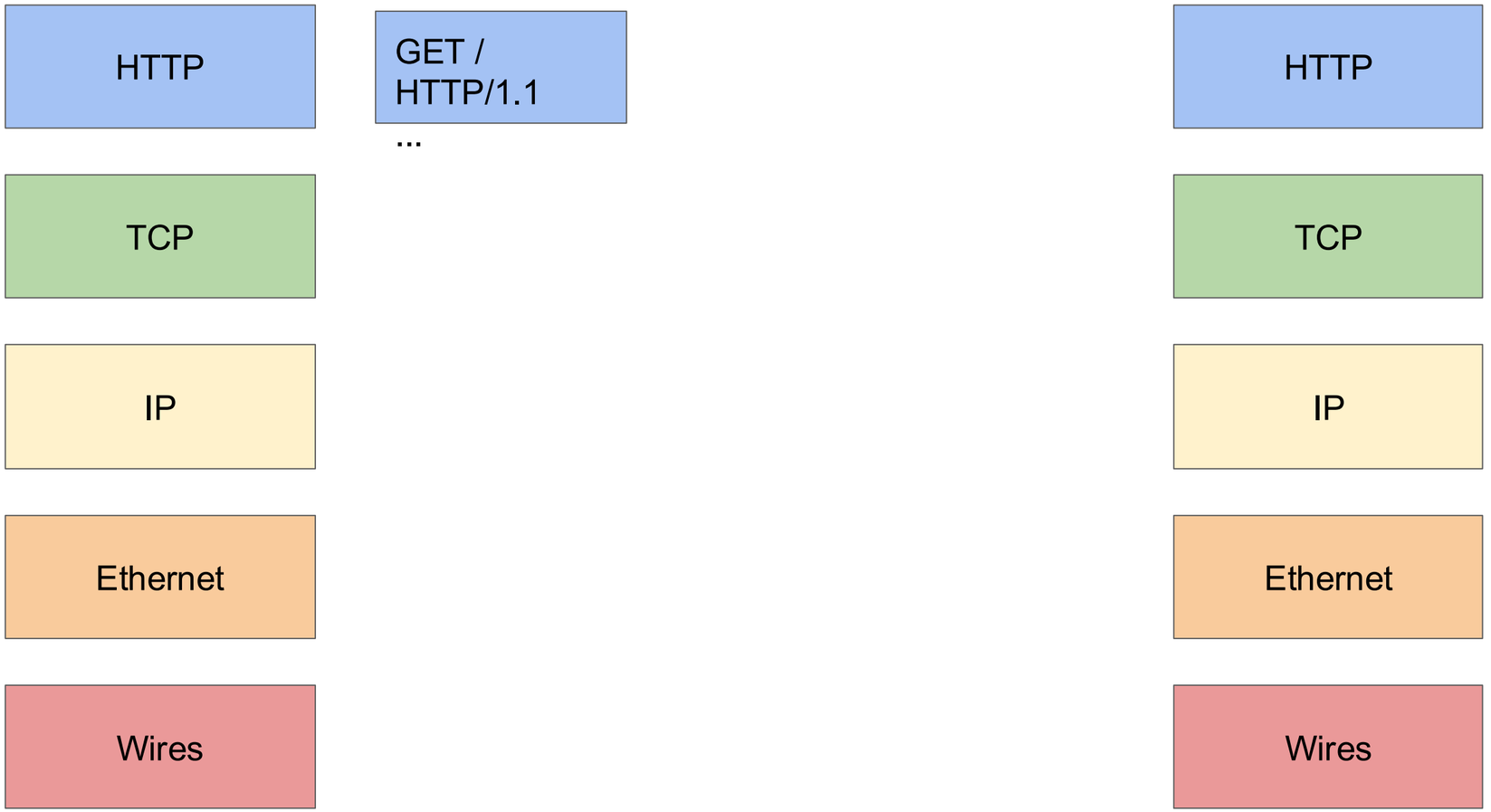
```
6c 36 ff 4a 32 92  dst MAC (router)
20 61 84 3a a9 52  src MAC (Alice)
08 00  type = IPv4
45  version 4, header 20 B
40  TTL = 64,  06  protocol = TCP
01 02 03 04  src IP 1.2.3.4
05 06 07 08  dst IP 5.6.7.8
04 d2  src port 1234
00 50  dst port 80
47 45 54 20 …  "GET " in ASCII
```

Layers of Abstraction and Headers

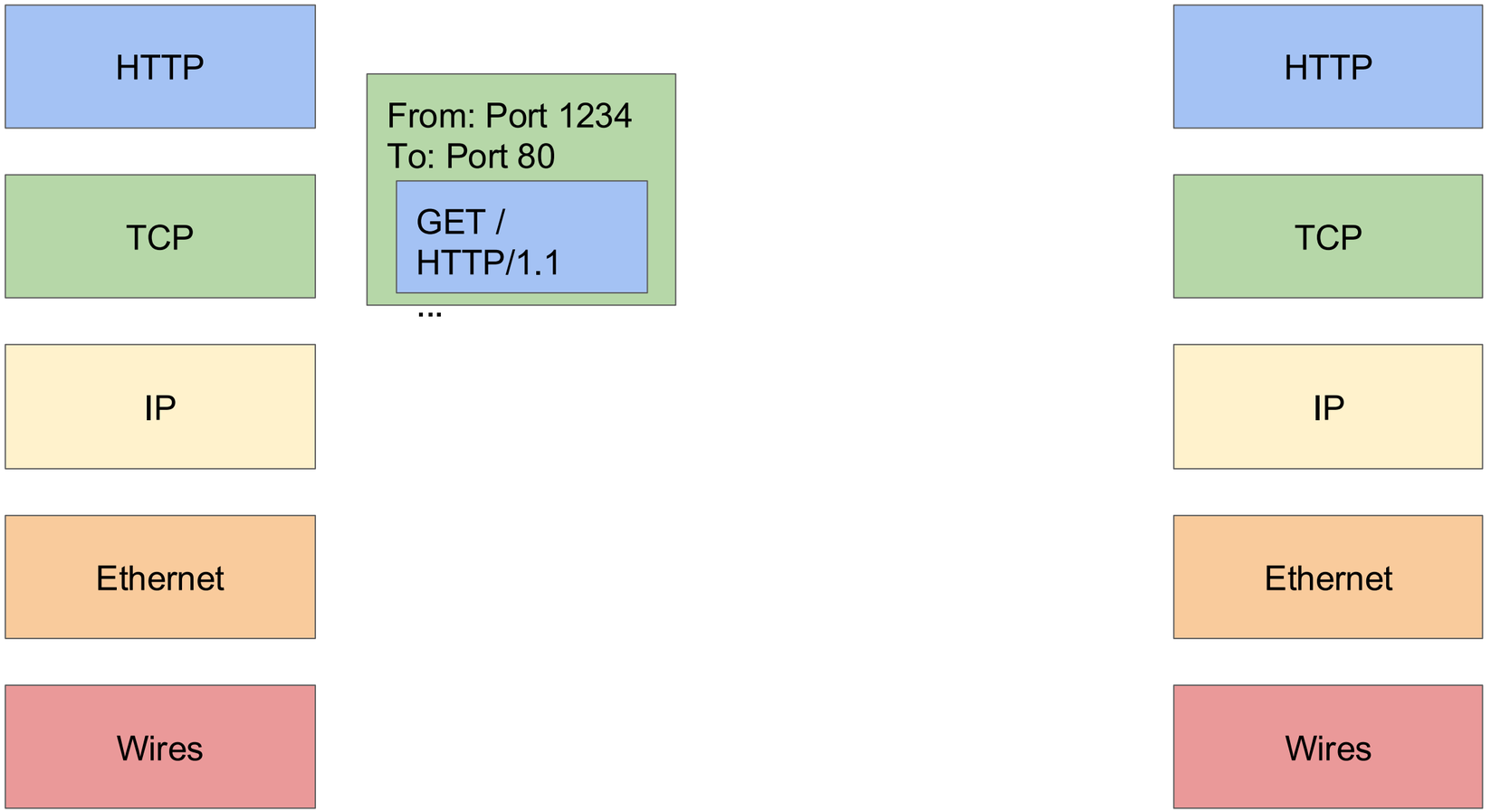


- As you move to lower layers, you wrap additional headers around the message
- As you move to higher layers, you peel off headers around the message
- When sending a message we go from the highest to the lowest layer
- When receiving a message we go from the lowest to highest layer

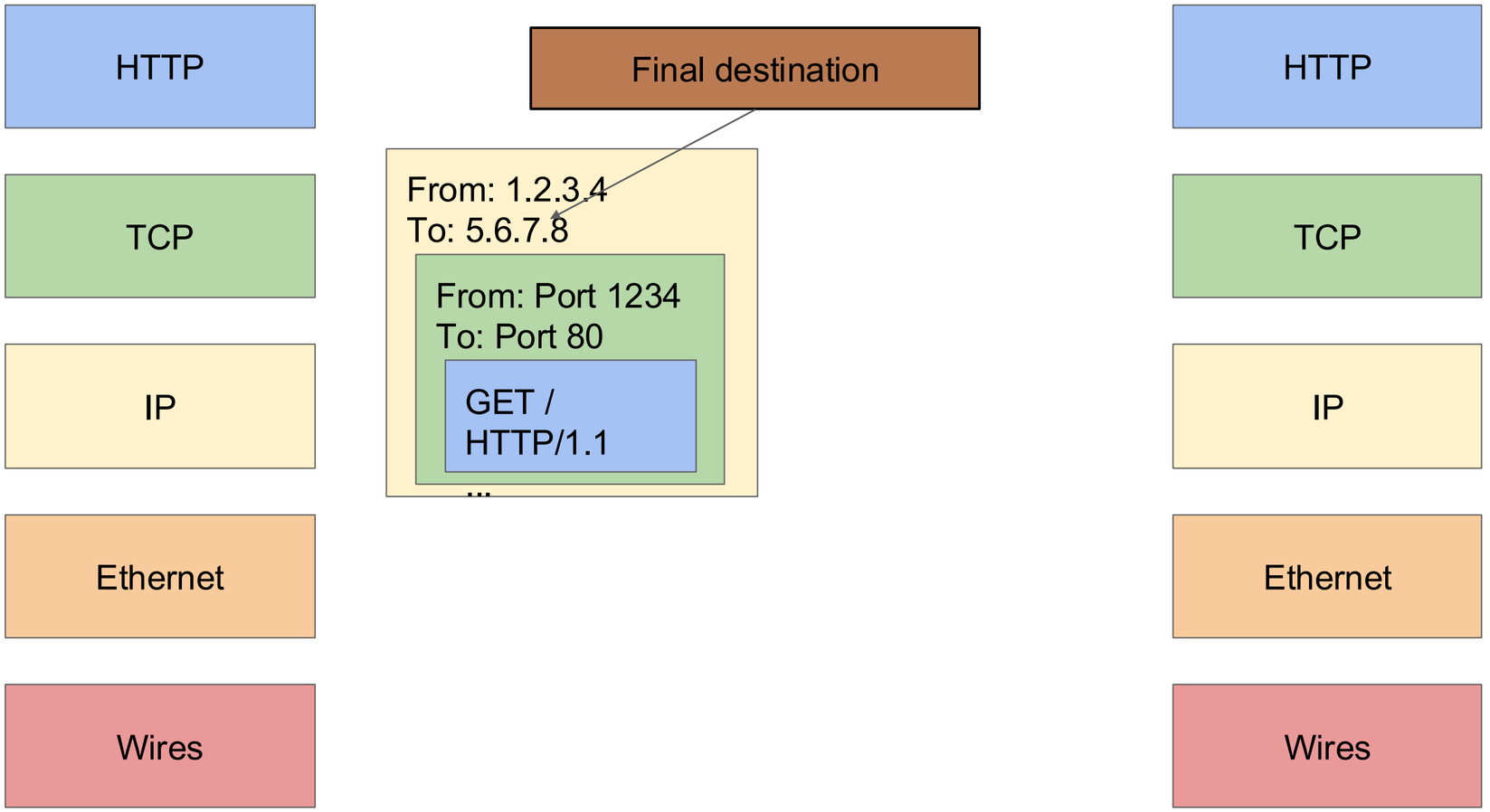
Example: HTTP Request



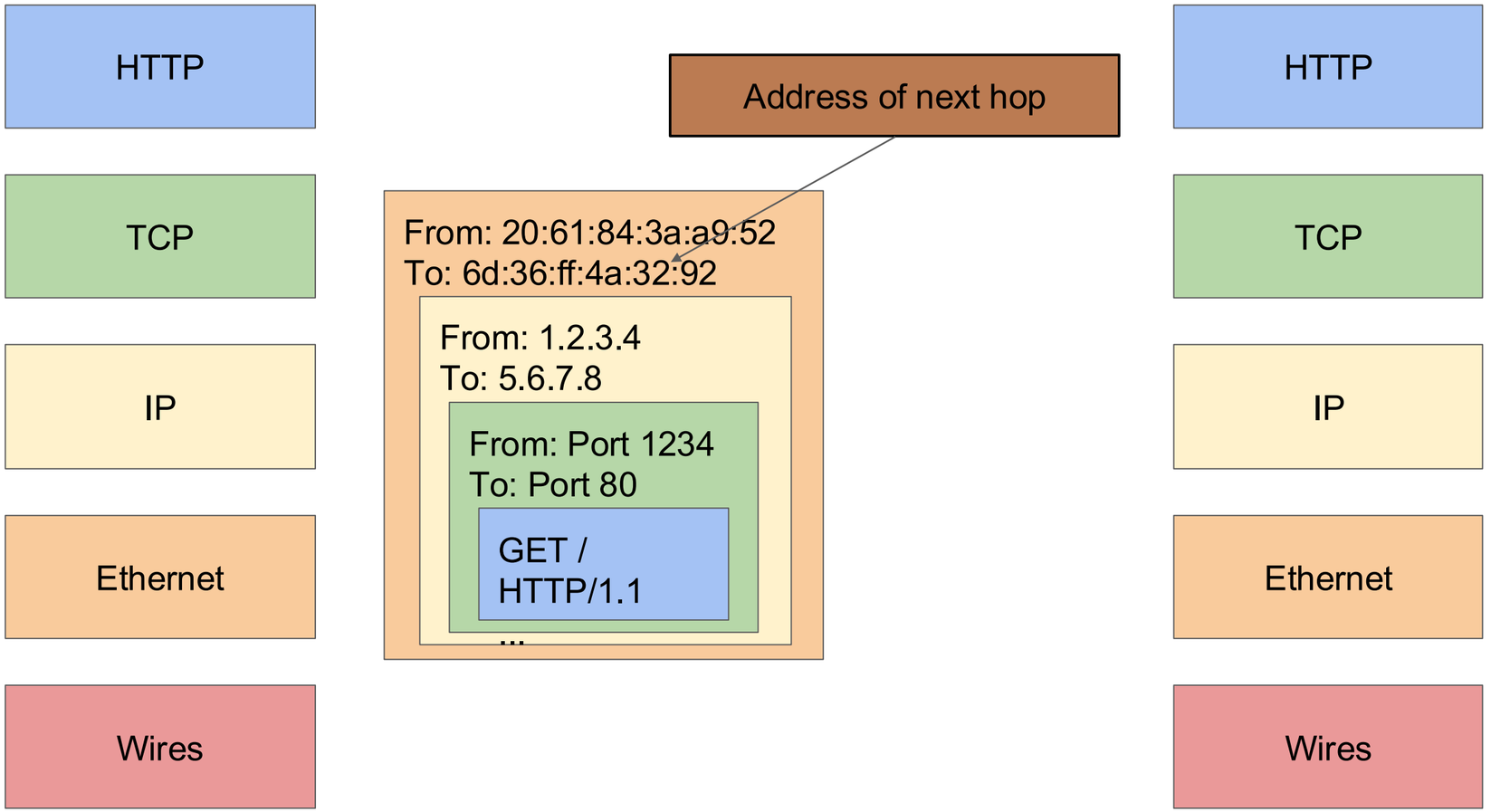
Example: HTTP Request



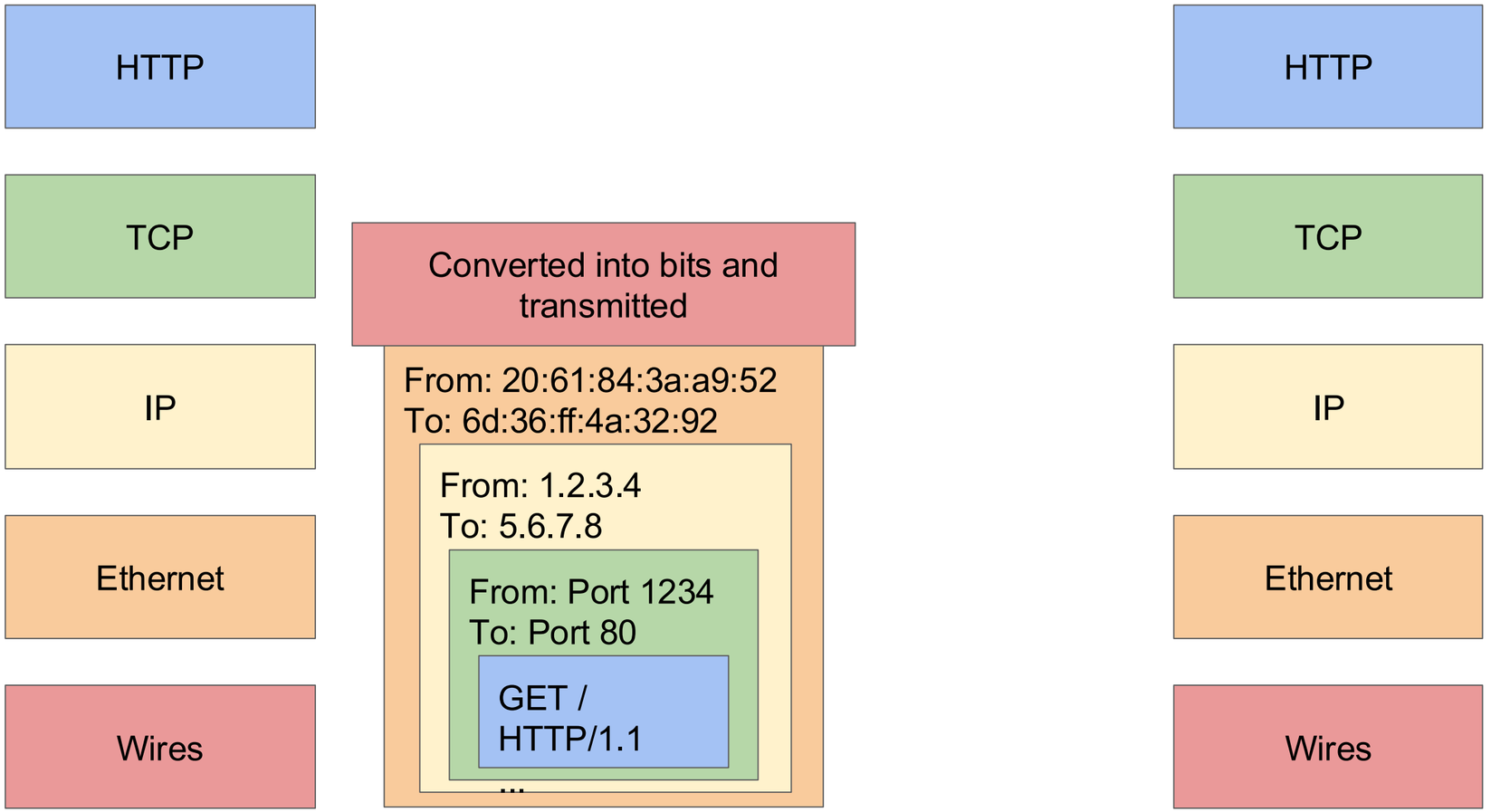
Example: HTTP Request



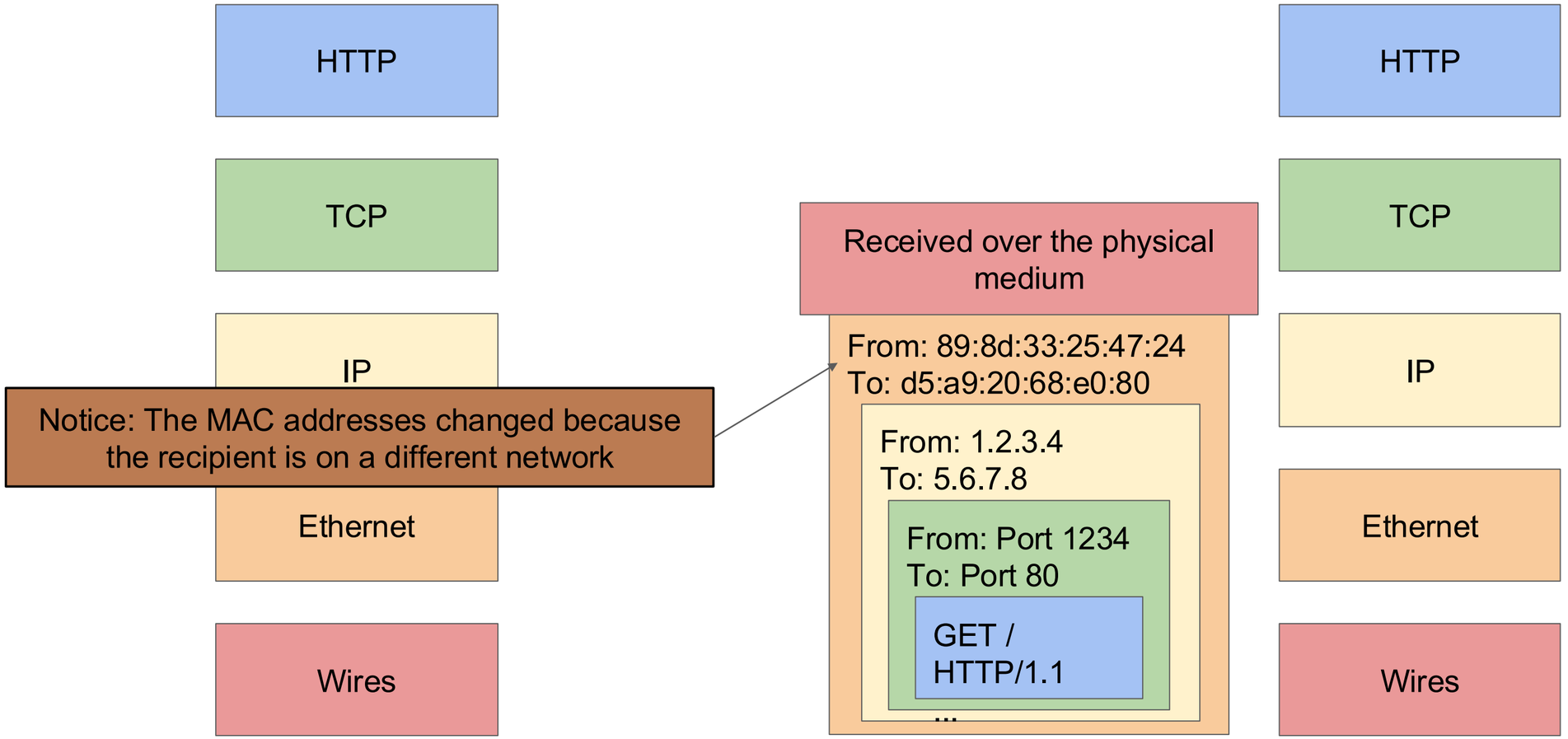
Example: HTTP Request



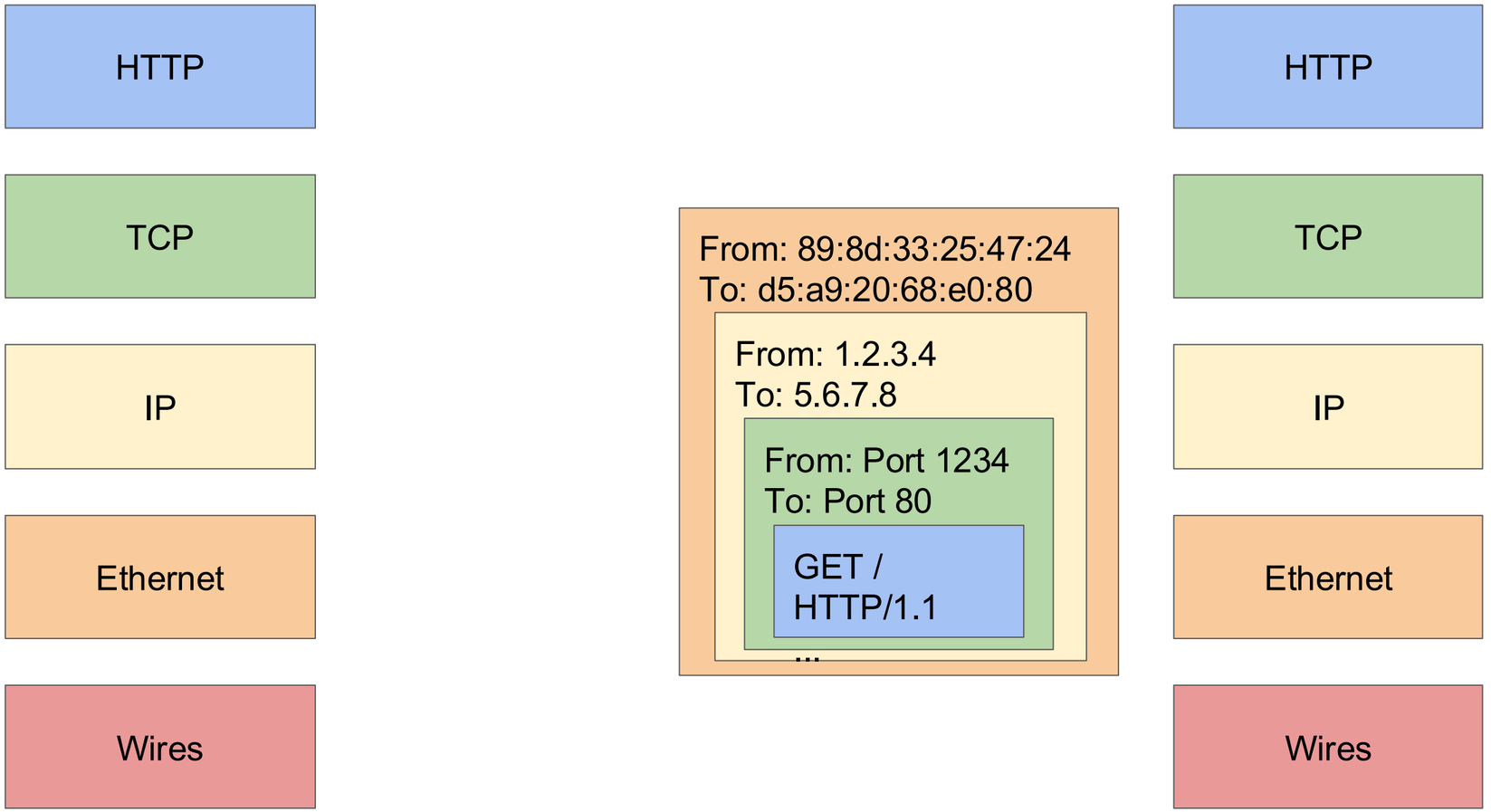
Example: HTTP Request



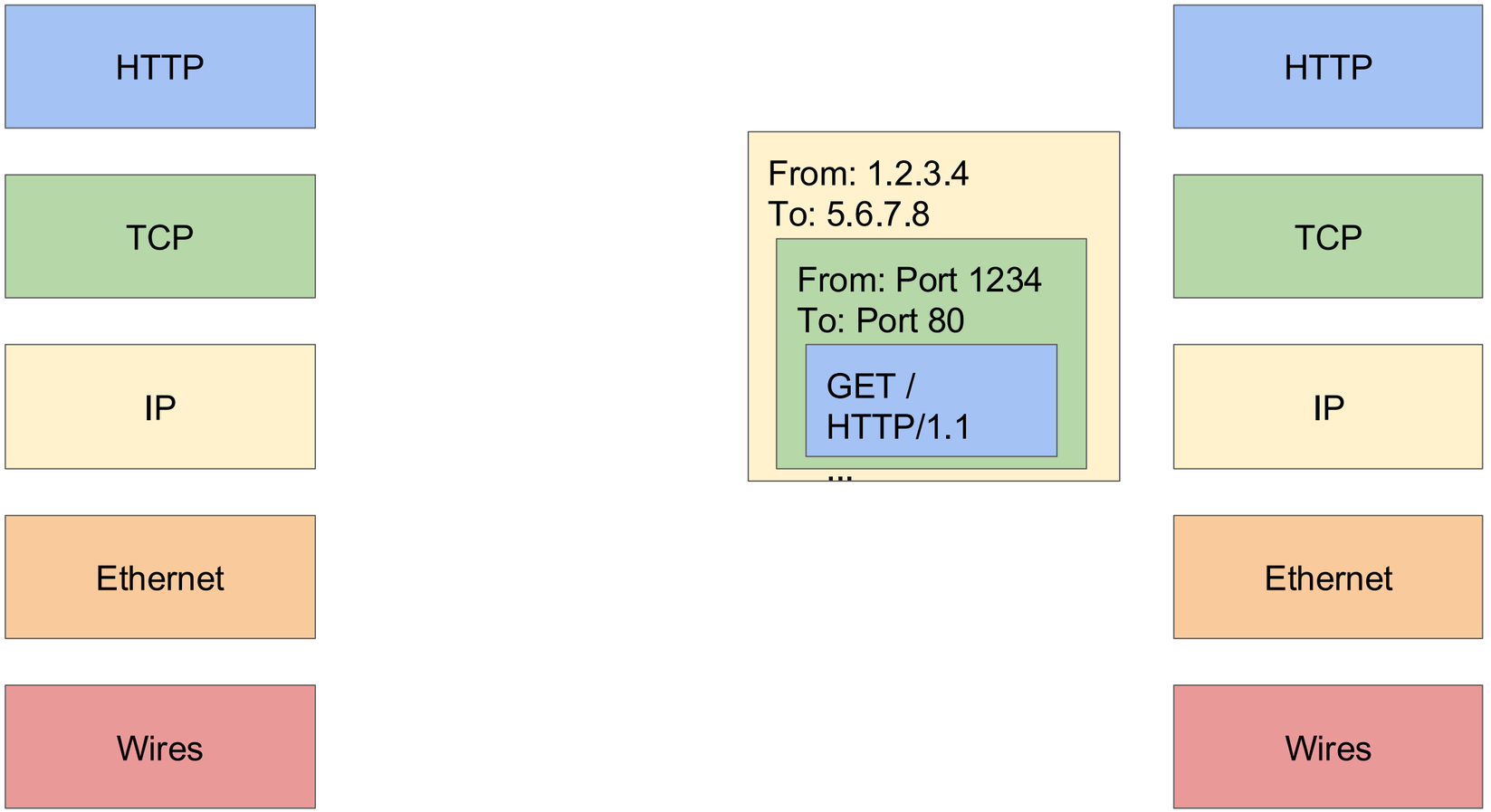
Example: HTTP Request



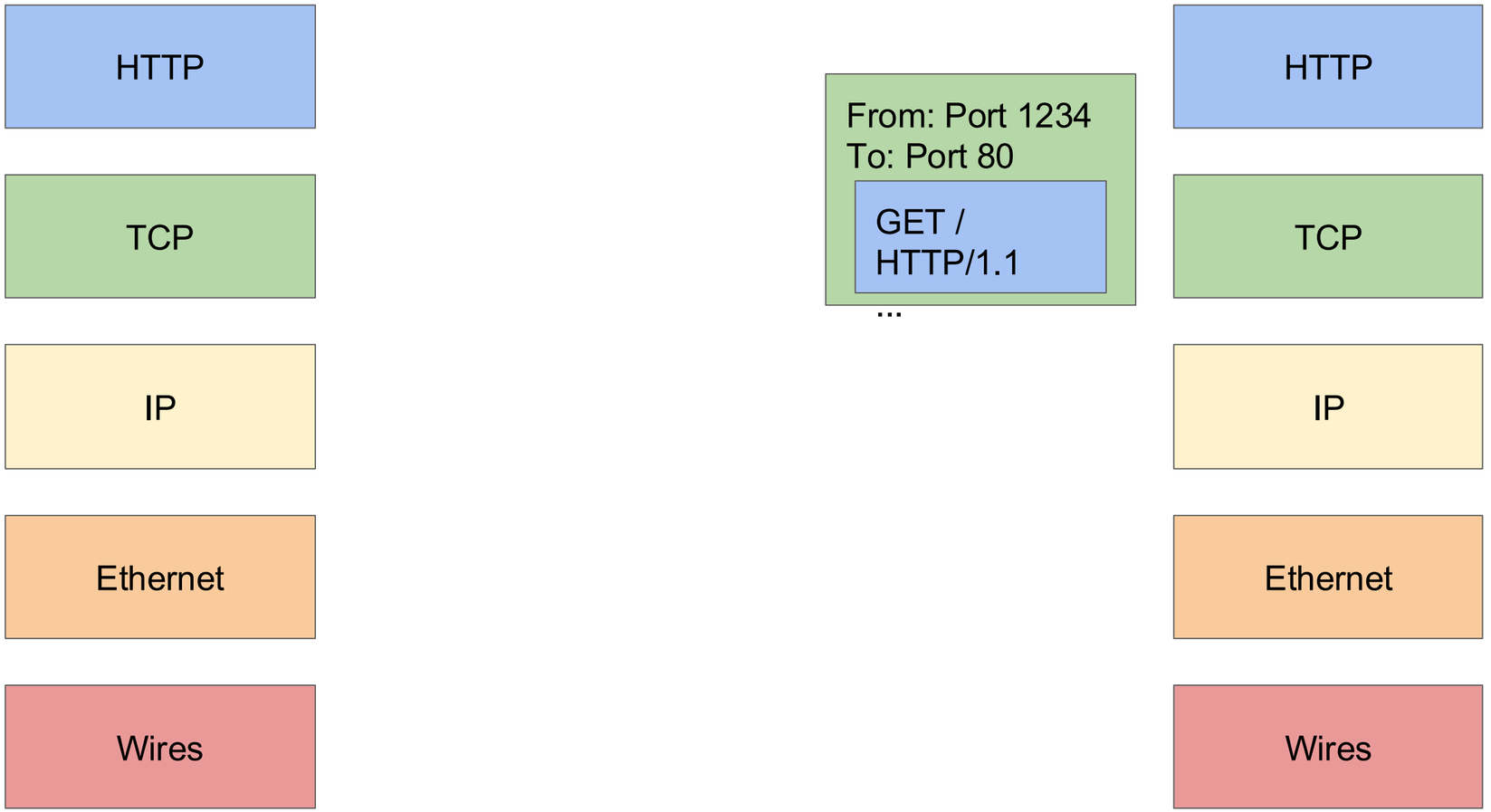
Example: HTTP Request



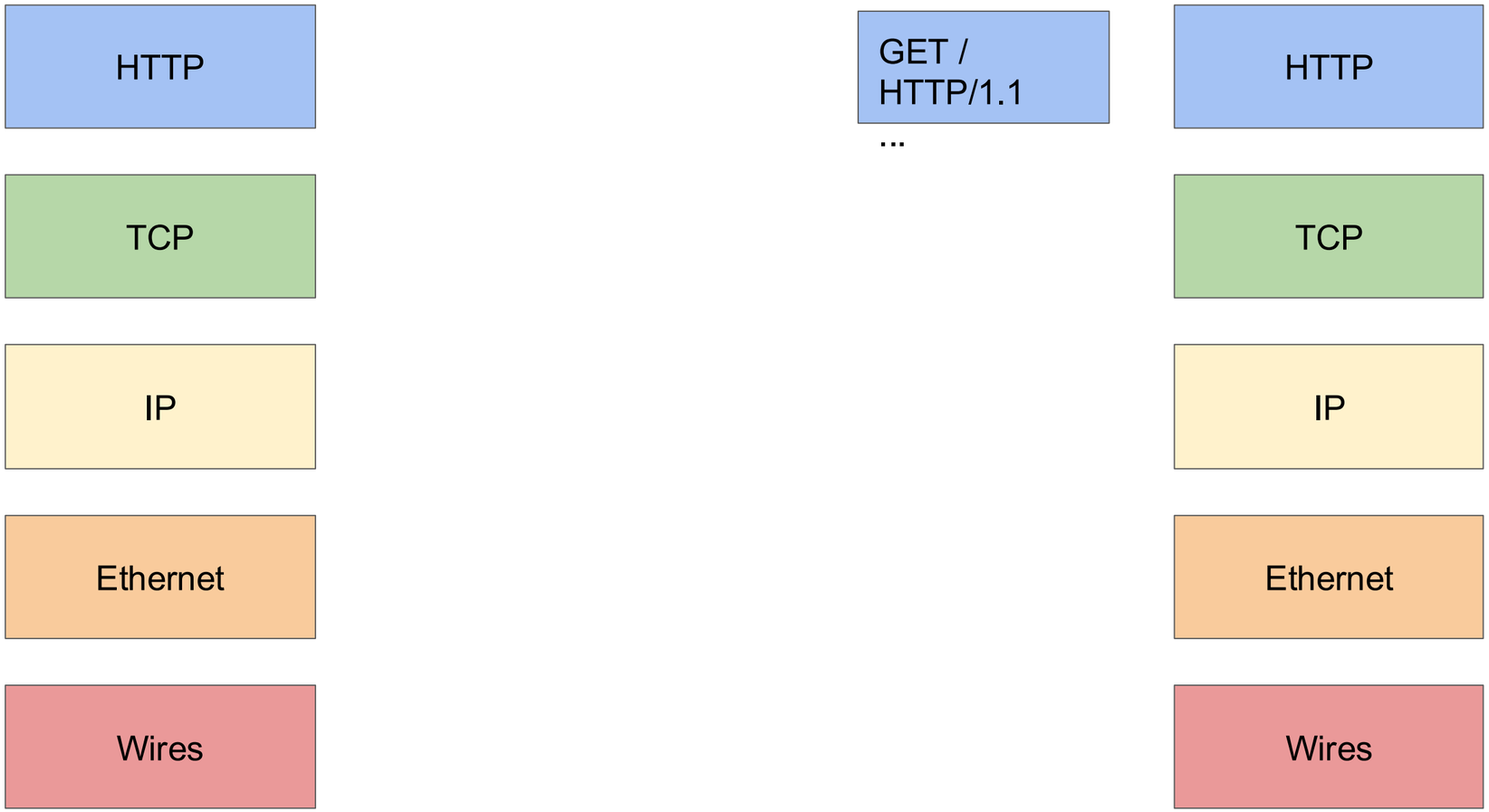
Example: HTTP Request



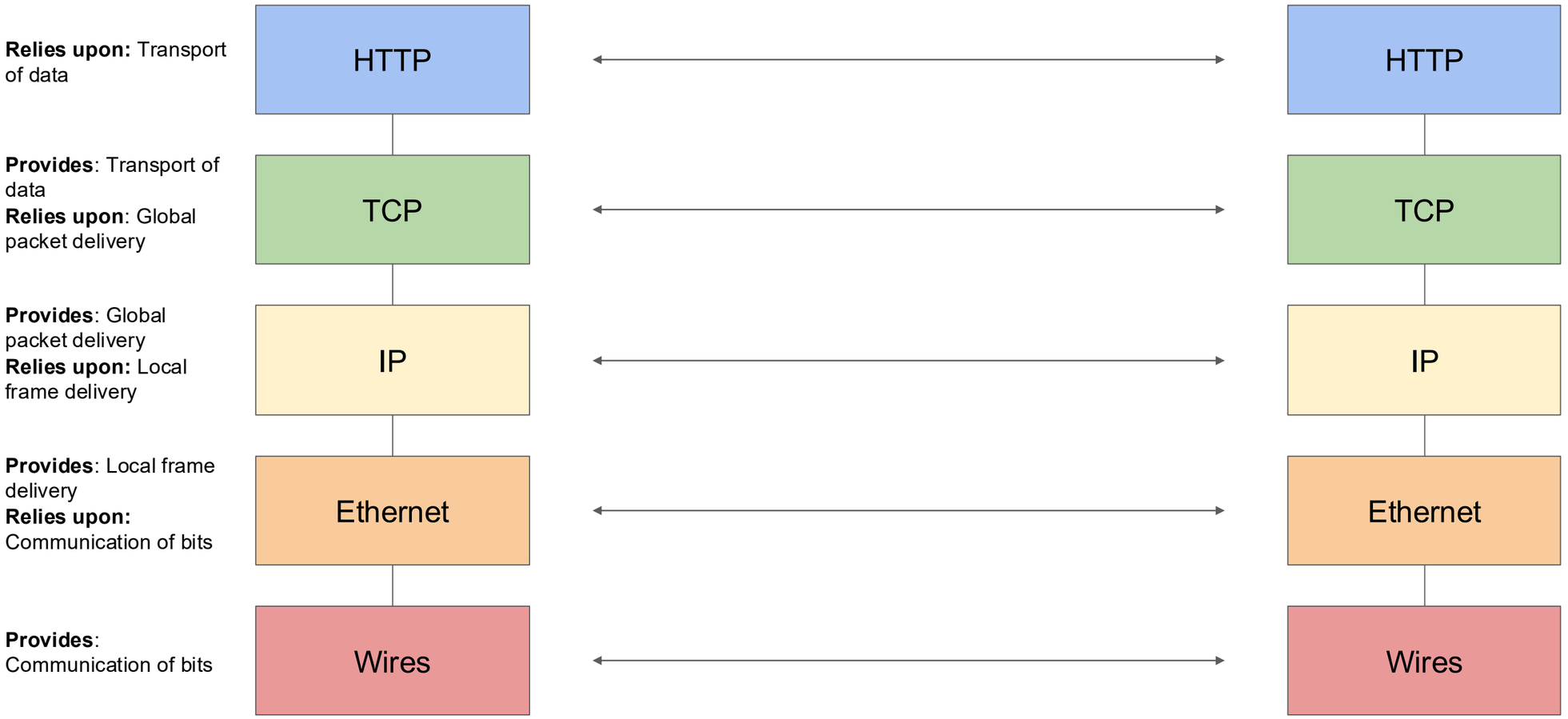
Example: HTTP Request



Example: HTTP Request



Example: HTTP Request

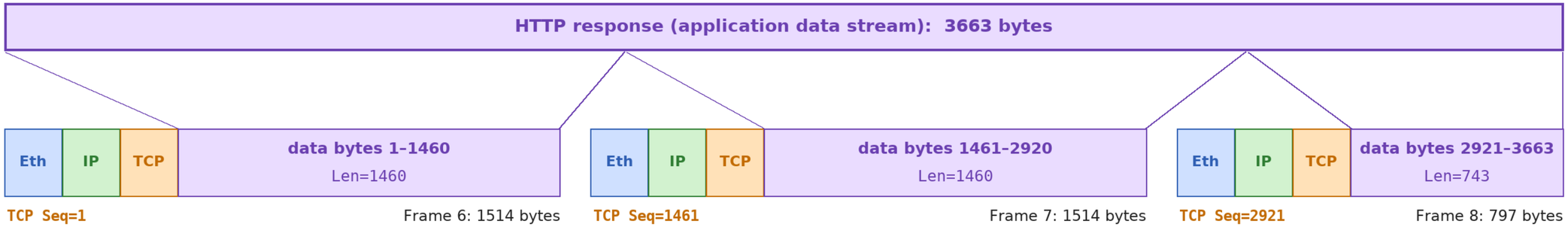


What about application writes lots of data

- TCP will do segmentation according to MSS (maximum segment size)

TCP segmentation: one 3663-byte HTTP response, three TCP segments

The server's application writes 3663 bytes. TCP cuts them into pieces of at most MSS = 1460 bytes (1500-byte Ethernet MTU − 20 B IP − 20 B TCP).



What Wireshark shows (http-get.pcap):

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	1.2.3.4	5.6.7.8	TCP	66	1234 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
2	0.000579	5.6.7.8	1.2.3.4	TCP	66	80 → 1234 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460
3	0.001006	1.2.3.4	5.6.7.8	TCP	54	1234 → 80 [ACK] Seq=1 Ack=1 Win=502 Len=0
4	0.001438	1.2.3.4	5.6.7.8	HTTP	124	GET / HTTP/1.1
5	0.001925	5.6.7.8	1.2.3.4	TCP	54	80 → 1234 [ACK] Seq=1 Ack=71 Win=509 Len=0
6	0.002336	5.6.7.8	1.2.3.4	TCP	1514	80 → 1234 [ACK] Seq=1 Ack=71 Len=1460 [TCP segment of a reassembled PDU]
7	0.002927	5.6.7.8	1.2.3.4	TCP	1514	80 → 1234 [ACK] Seq=1461 Ack=71 Len=1460 [TCP segment of a reassembled PDU]
8	0.003405	5.6.7.8	1.2.3.4	HTTP	797	HTTP/1.1 200 OK (text/html)
9	0.004037	1.2.3.4	5.6.7.8	TCP	54	1234 → 80 [ACK] Seq=71 Ack=3664 Win=502 Len=0

Seq of each segment = Seq of the previous one + its Len (1 → 1461 → 2921). The client's Ack=3664 means "I have every byte up to 3663".

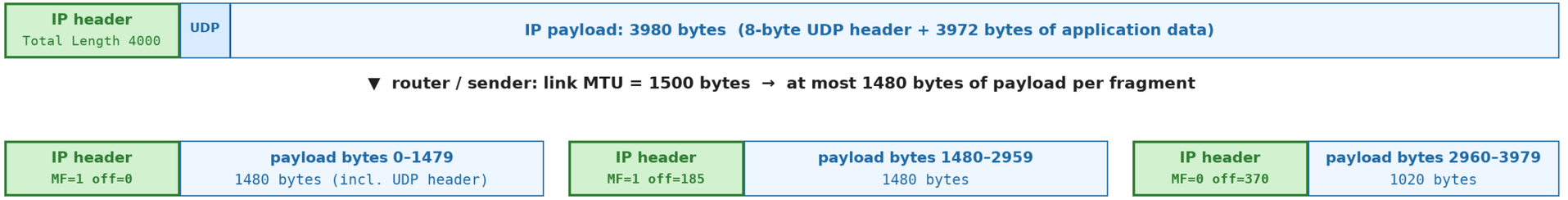
Wireshark labels the first two "TCP segment of a reassembled PDU" and shows the whole HTTP response at frame 8, after reassembly.

What about application writes lots of data

- UDP does not do segmentation, the IP layer will take care of that

IP fragmentation: a UDP datagram carrying 3972 bytes of data, on a 1500-byte link

The packet is larger than the link MTU, so it is split into fragments. Each fragment gets its own IP header; only the first one carries the UDP header (the ports).



The three fragments' IP headers:

Fragment	Total Length	Identification	MF (more fragments)	Fragment Offset field	= byte offset	Frame on the wire
1	1500	**0x3a5c**	**1**	**0**	0	1514 bytes
2	1500	**0x3a5c**	**1**	**185**	1480	1514 bytes
3	1040	**0x3a5c**	**0**	**370**	2960	1054 bytes

- Same Identification (0x3a5c) → the receiver knows the fragments belong together.
- Fragment Offset is in units of 8 bytes: 185 × 8 = 1480, 370 × 8 = 2960.
- MF = 1 on every fragment but the last. Only the destination reassembles; routers in between do not.
- Fragments 2 and 3 have no UDP header, so no port numbers: a firewall cannot tell which application they belong to.
- Compare TCP (figure 06): TCP cuts data into segments that fit (MSS) and sets DF. UDP does not, so IP has to fragment.

What Wireshark shows (ip-fragmentation.pcap):

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	1.2.3.4	5.6.7.8	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=3a5c) [Reassembled in #3]
2	0.000004	1.2.3.4	5.6.7.8	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=1480, ID=3a5c) [Reassembled in #3]
3	0.000008	1.2.3.4	5.6.7.8	UDP	1054	40000 → 5000 Len=3972

Intro to Networking: Summary



- **Internet: A global network of computers**
  - Protocols: Agreed-upon systems of communication
- **OSI model: A layered model of protocols**
  - Layer 1: Communication of bits
  - Layer 2: Local frame delivery
    - Ethernet: The most common Layer 2 protocol
    - MAC addresses: 6-byte addressing system used by Ethernet
  - Layer 3: Global packet delivery
    - IP: The universal Layer 3 protocol
    - IP addresses: 4-byte (or 16-byte) addressing system used by IP
  - Layer 4: Transport of data (more on this next time)
  - Layer 7: Applications and services (the web)

