

Network Security: Background and Start on Attacks

CS 161: Computer Security

Prof. Raluca Ada Popa

February 26, 2019

Some slides credit David Wagner.

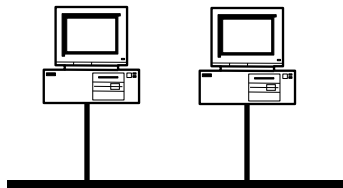
Announcements

- Project collaboration policy: every teammate must do every part
- Midterm 1 grades will be out by hopefully end of this week, one week for regrades
- Project 2 due March 11, get started
- Switching order of network security with web security in syllabus

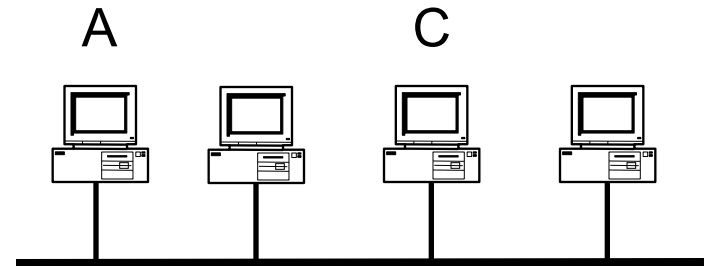
Networking overview

Pay attention to this material (part of 168) because you will need this to understand it for the class

Local-Area Networks (LAN)



point-to-point



shared

If two computers transmit at the same time, they interfere

How does computer A send a message to computer C?

Packets

Source: A

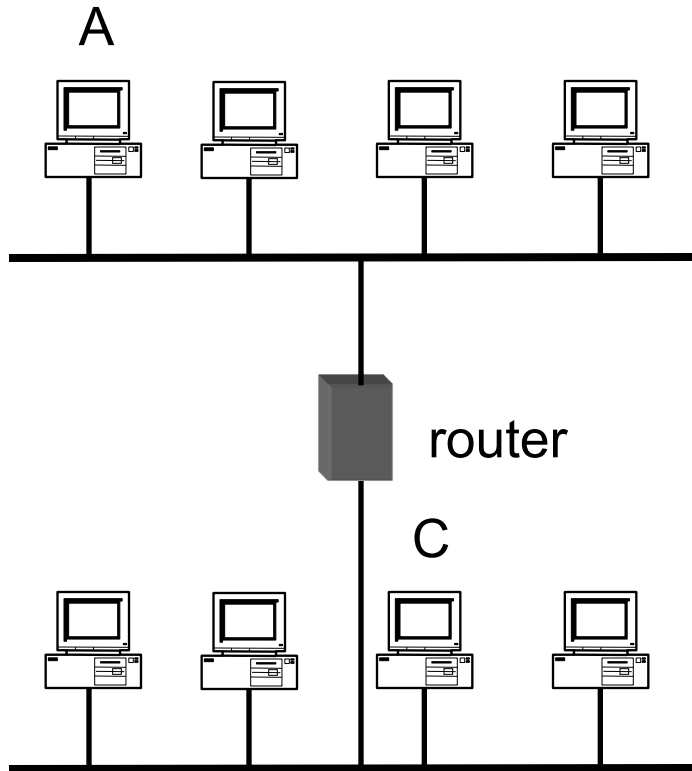
Destination: C

Message: Hello world!

A	C	Hello world!
---	---	--------------

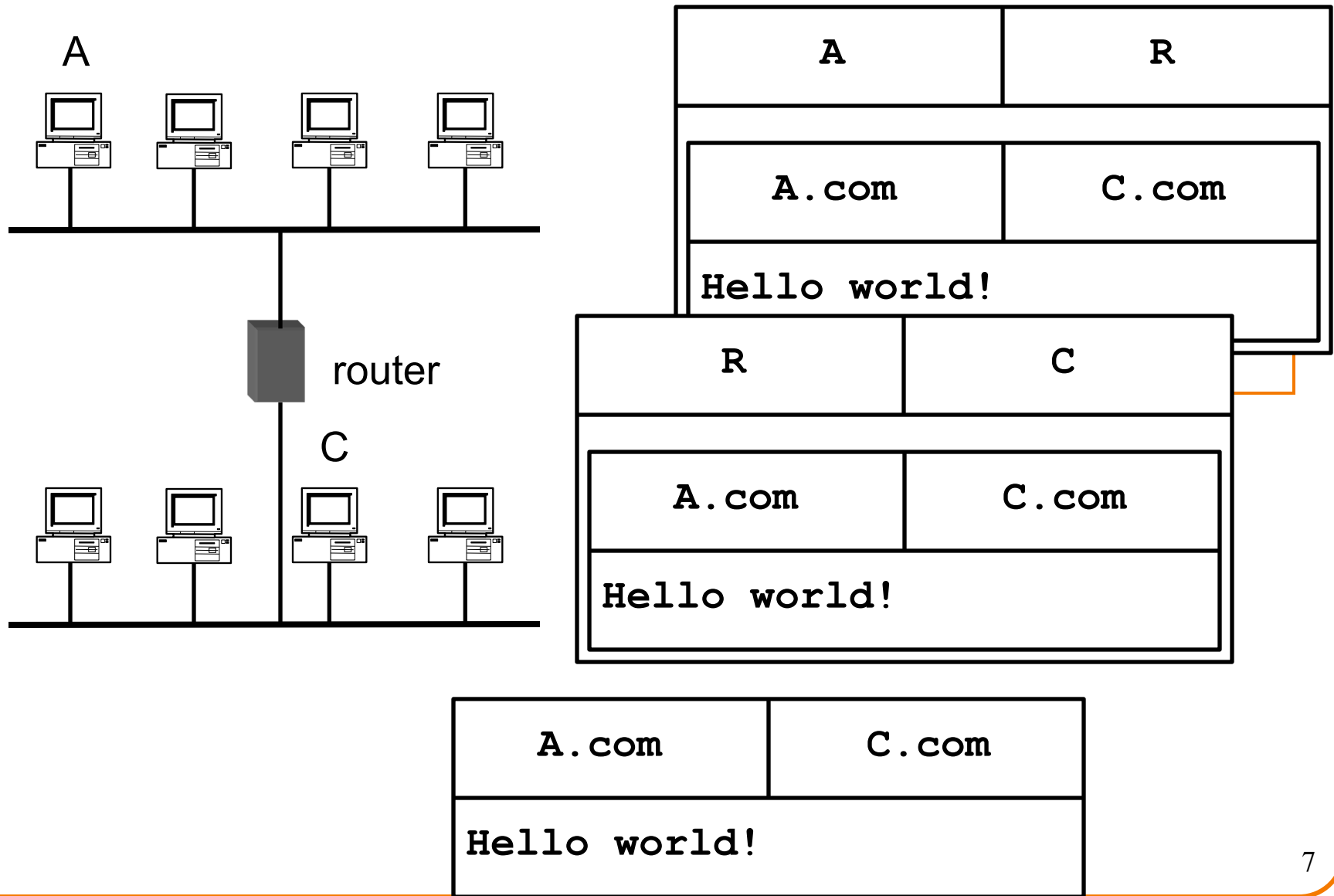
A	C
Hello world!	

Wide-Area Networks



How do we connect two LANs?

Wide-Area Networks



Key Concept #1: *Protocols*

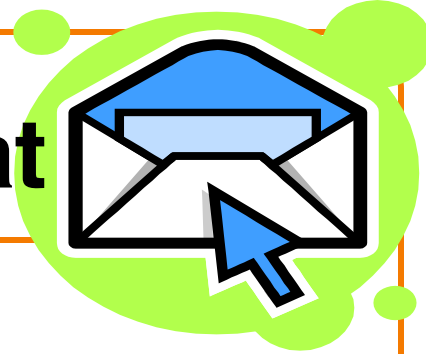
- A protocol is an **agreement on how to communicate**
- Includes **syntax** and **semantics**
 - How a communication is specified & structured
 - o Format, order messages are sent and received
 - What a communication means
 - o Actions taken when transmitting, receiving, or timer expires
- Example: making a comment in lecture?
 1. Raise your hand.
 2. Wait to be called on.
 3. Or: wait for speaker to **pause** and vocalize
 4. If unrecognized (after **timeout**): say “excuse me”

Key Concept #2: *Dumb Network*

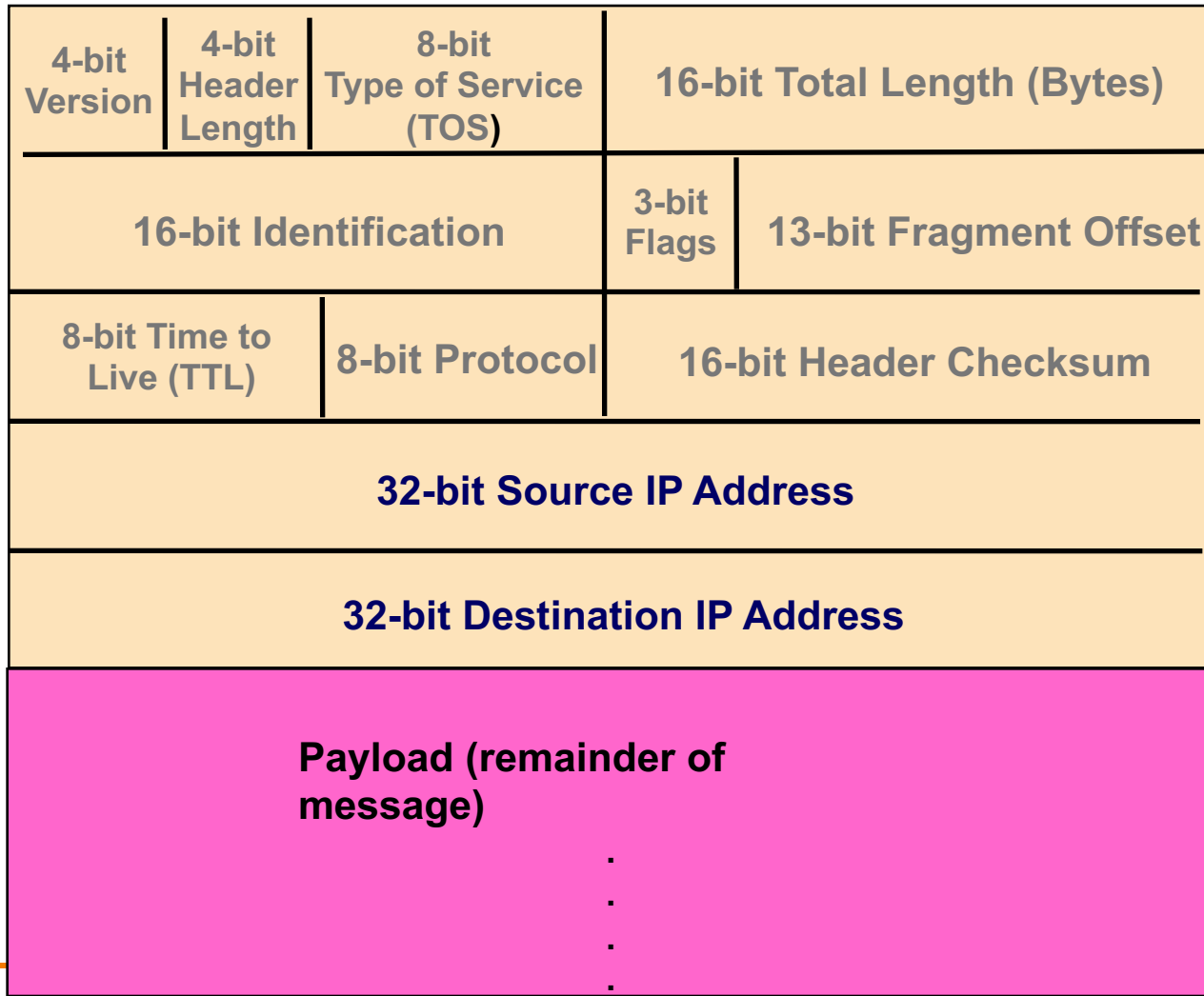
- Original Internet design: interior nodes (“**routers**”) have no knowledge* of ongoing connections going through them
- **Not** how you the telephone system works
 - Which internally tracks all of the active voice calls
- Instead: the **postal system!**
 - Each Internet message (“**packet**”) self-contained

* Today's Internet is full of hacks that violate this

Self-Contained IP Packet Format



IP = Internet *Protocol*



***Header* is like a letter envelope: contains all info needed for delivery**

Key Concept #2: *Dumb Network*

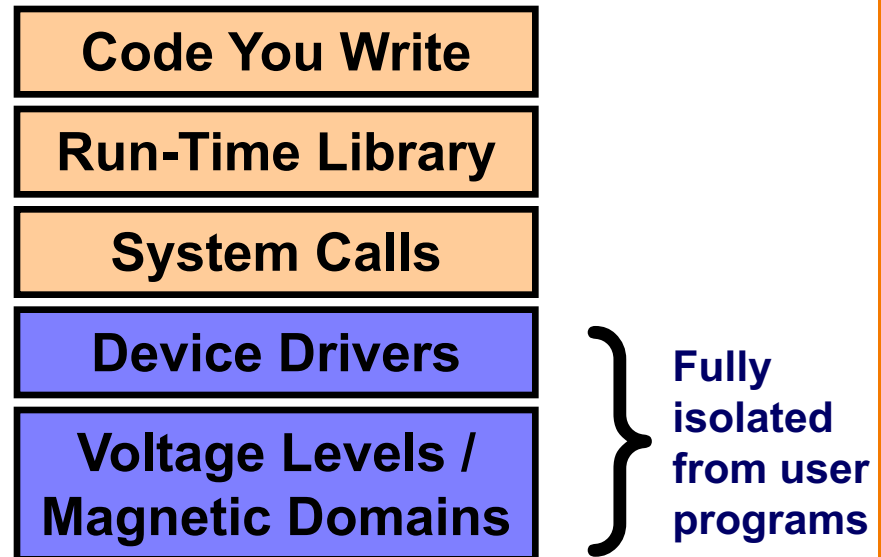
- Original Internet design: interior nodes (“**routers**”) have no knowledge* of ongoing connections going through them
- **Not:** how you picture the telephone system works
 - Which internally tracks all of the active voice calls
- Instead: the **postal system!**
 - Each Internet message (“**packet**”) self-contained
 - Interior routers look at destination address to forward

* Today's Internet is full of hacks that violate this

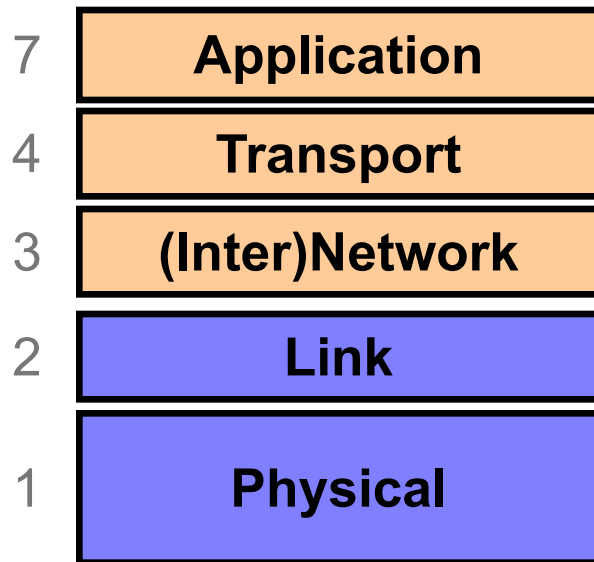
Key Concept #3: *Layering*

- The Internet design is strongly partitioned into layers
 - Each layer relies on services provided by next layer below ...
 - ... and provides services to layer above it

- Analogy:
 - Consider structure of an application you've written and the “services” each layer relies on / provides



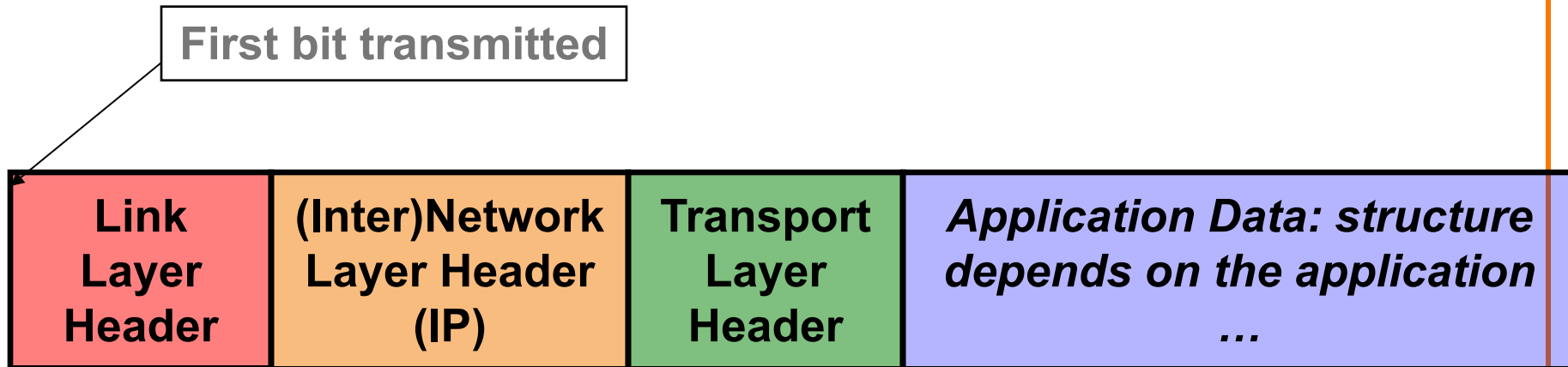
Internet Layering (“Protocol Stack”)



Note on a point of potential confusion: these diagrams are always drawn with lower layers **below** higher layers ...

But diagrams showing the layouts of packets are often the *opposite*, with the lower layers at the **top** since their headers precede those for higher layers

Horizontal View of a Single Packet



The diagram illustrates the seven layers of the OSI model, represented as a vertical stack of colored boxes. From top to bottom, the layers are: Link Layer Header (red), (Inter)Network Layer Header (IP) (orange), Transport Layer Header (green), and Application Data: structure depends on the application (blue). The bottom three layers are grouped by a bracket on the right, labeled 'Data'. A callout box on the left, labeled 'First bit transmitted', has an arrow pointing to the top of the Link Layer Header box. Below the Application Data box, there are seven vertical dots indicating further layers or data units.

Layer	Header/Content
7	Link Layer Header
6	(Inter)Network Layer Header (IP)
5	Transport Layer Header
4	Application Data: structure depends on the application
3	
2	
1	

First bit transmitted

15

First bit transmitted

Link Layer Header

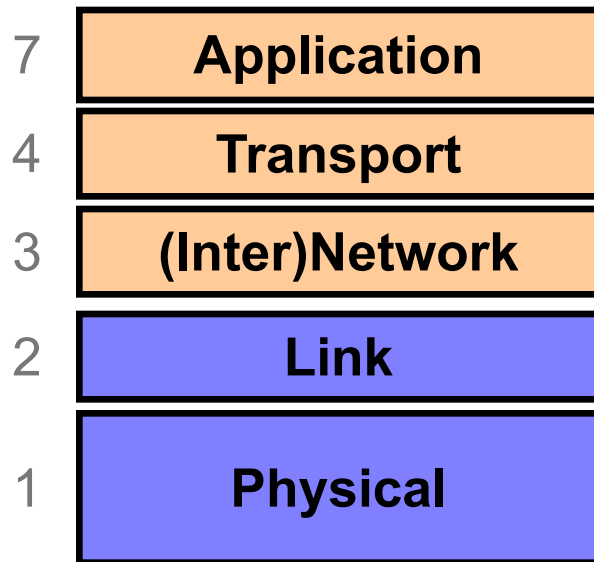
(Inter)Network Layer Header (IP)

Transport Layer Header

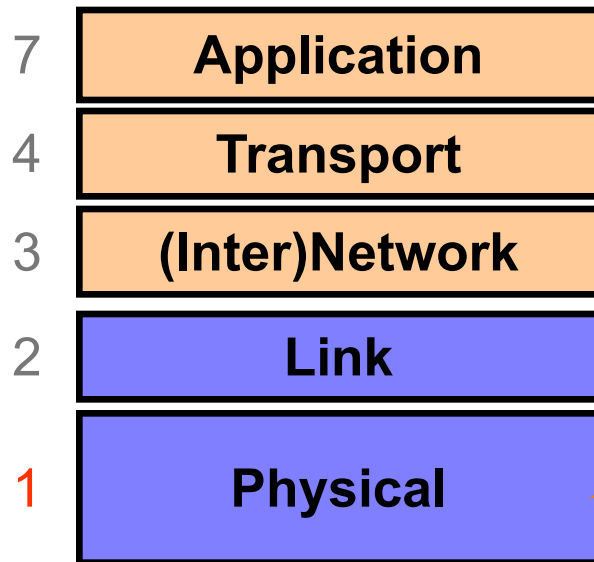
***Application Data:
structure depends on the
application***

- .
- .
- .
- .
- .
- .
- .

Internet Layering (“Protocol Stack”)

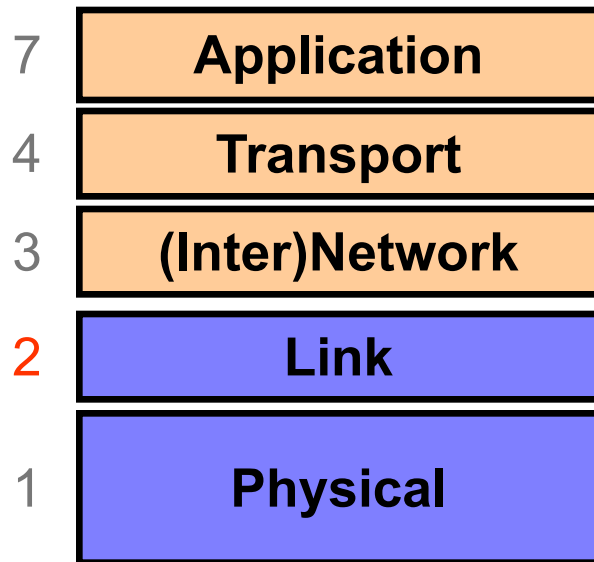


Layer 1: Physical Layer



Encoding **bits** to send them over a single **physical link**
e.g. patterns of
*voltage levels /
photon intensities /
RF modulation*

Layer 2: Link Layer

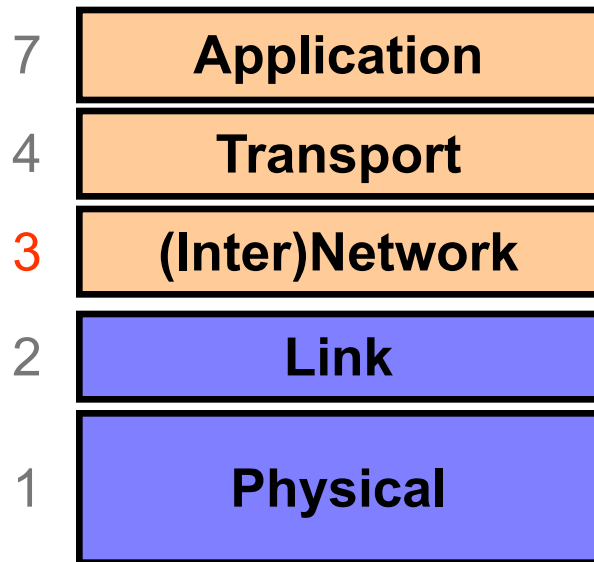


Framing and transmission of a collection of bits into individual **messages** sent across a single “subnetwork” (one physical technology)

Might involve multiple *physical links* (e.g., modern Ethernet)

Often technology supports **broadcast** transmission (**every** “node” connected to subnet receives)

Layer 3: (Inter)Network Layer (*IP*)



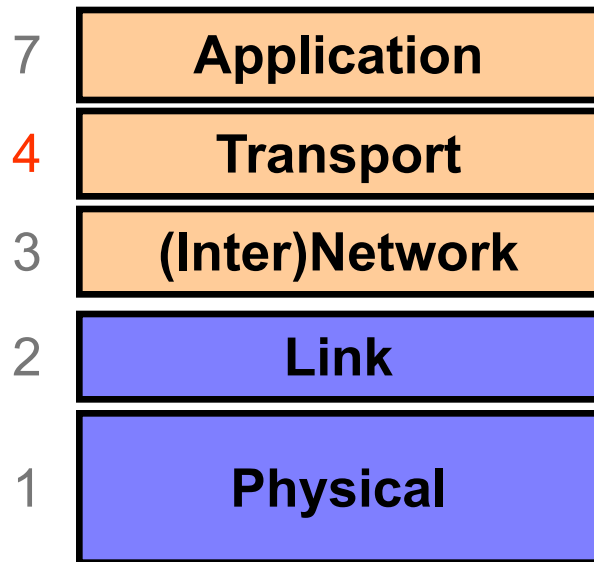
Bridges multiple “subnets” to provide *end-to-end* internet connectivity between nodes

- Provides global addressing

Works across different link technologies

Different for each Internet “hop”

Layer 4: Transport Layer

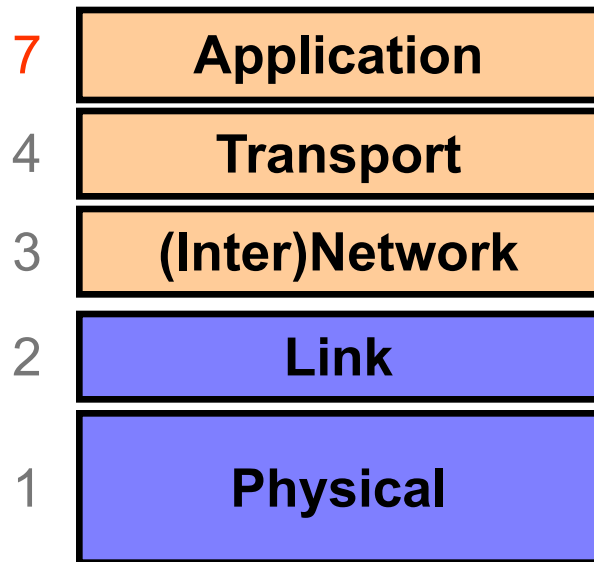


*End-to-end communication
between processes*

Different services provided:
TCP = reliable *byte stream*
UDP = unreliable *datagrams*

(Datagram = single packet message)

Layer 7: Application Layer



Communication of whatever you wish

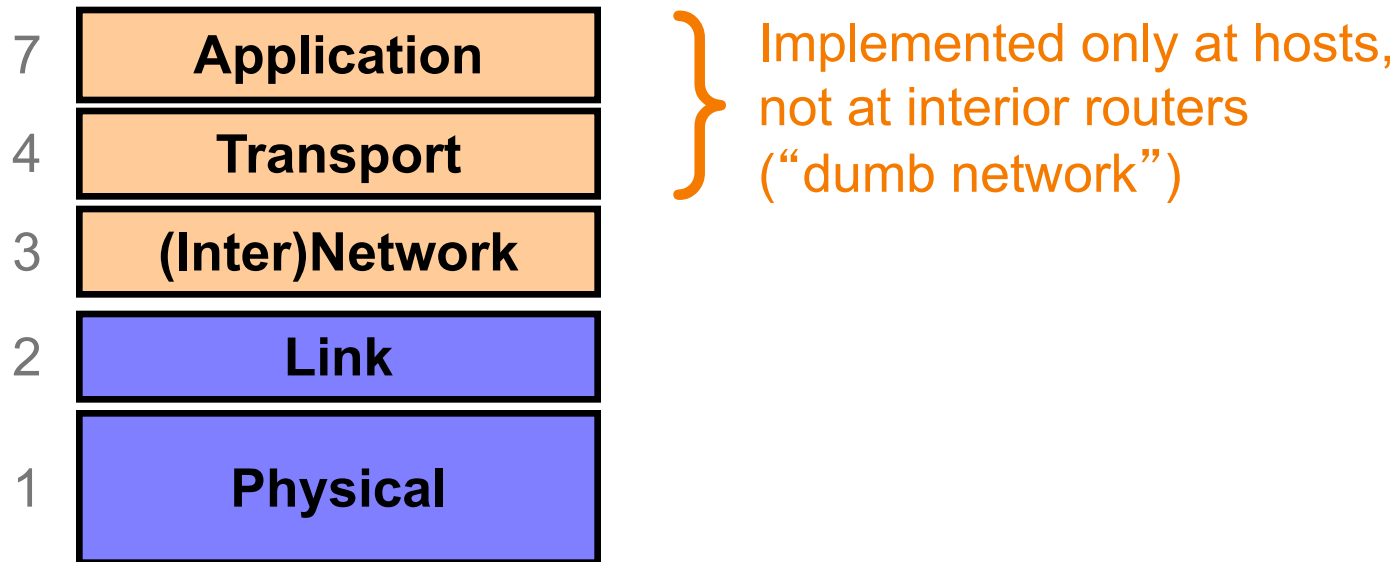
Can use whatever transport(s) is convenient

Freely structured

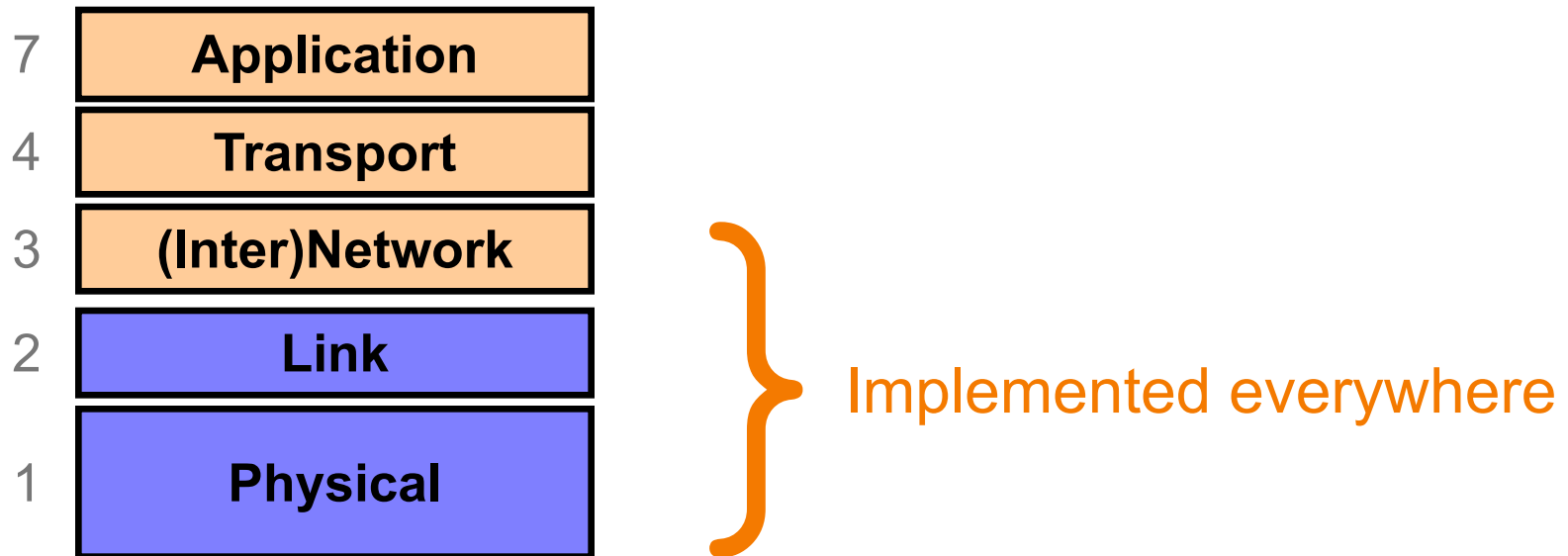
E.g.:

Skype, SMTP (email),
HTTP (Web), Halo, BitTorrent

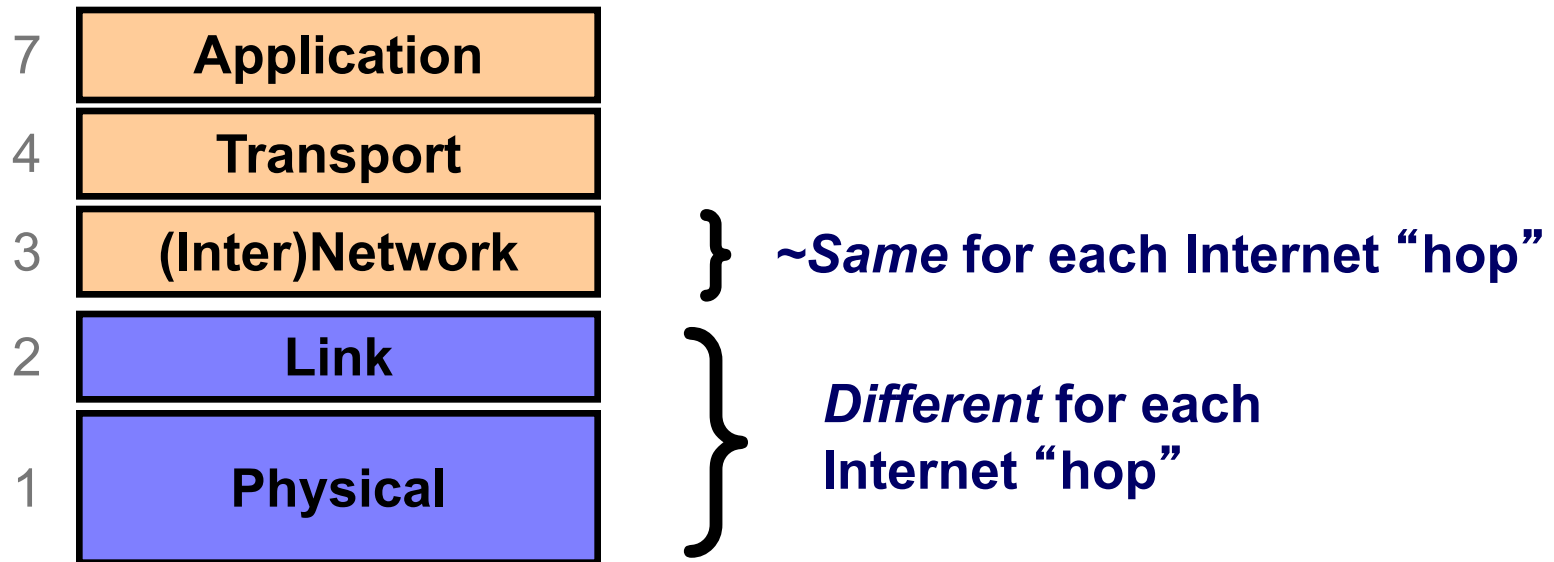
Internet Layering (“Protocol Stack”)



Internet Layering (“Protocol Stack”)

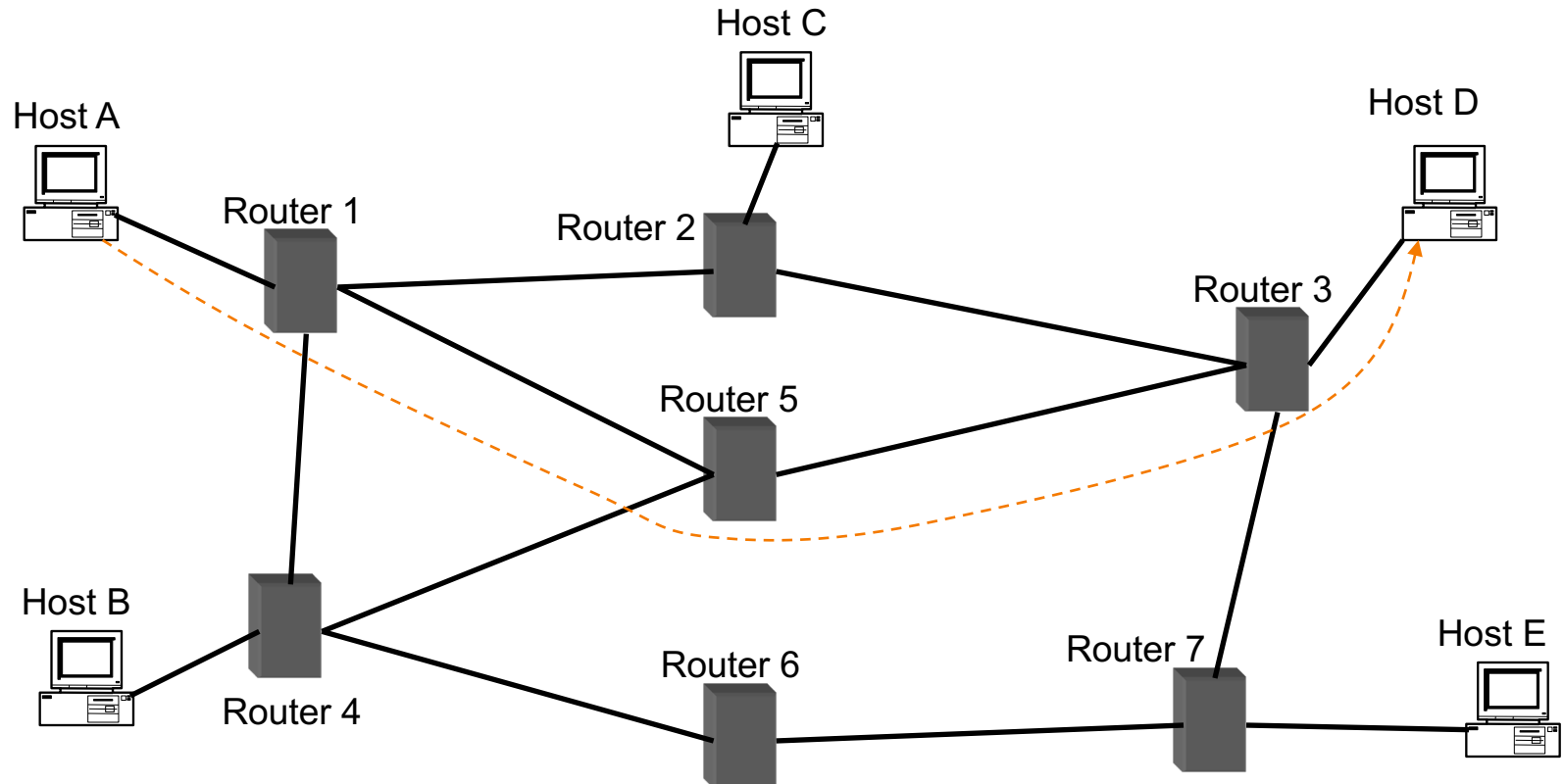


Internet Layering (“Protocol Stack”)



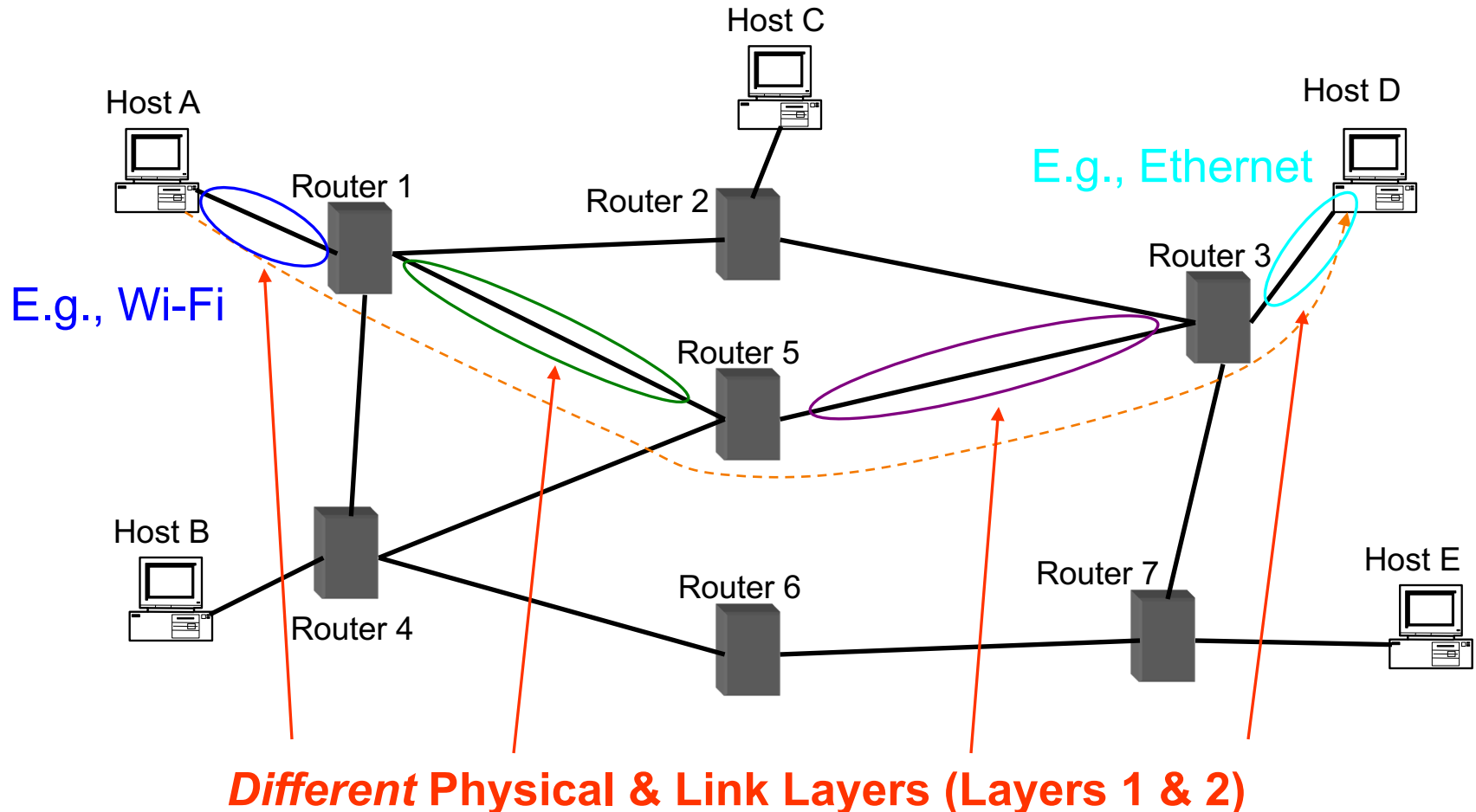
Hop-By-Hop vs. End-to-End Layers

Host A communicates with Host D



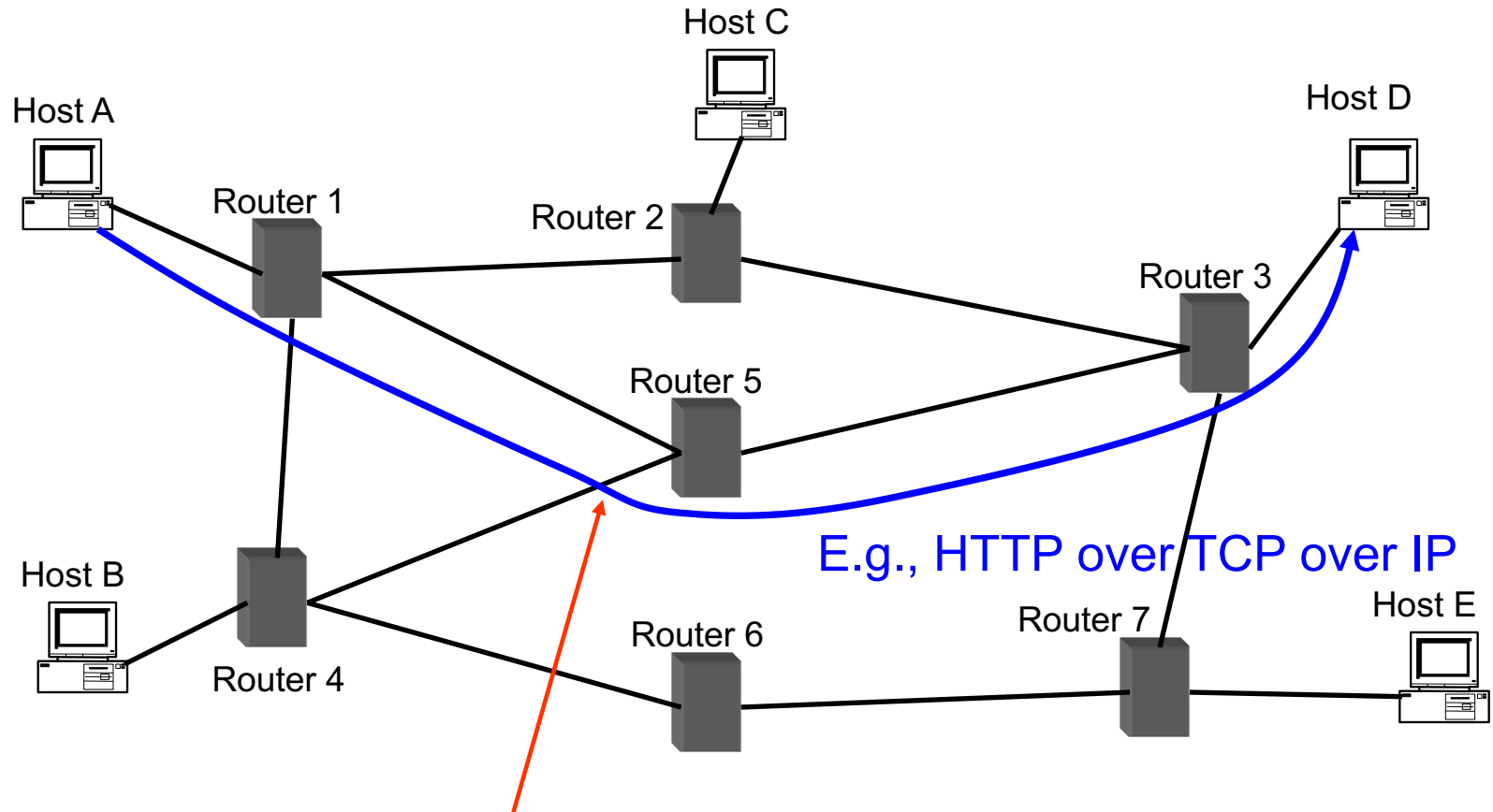
Hop-By-Hop vs. End-to-End Layers

Host A communicates with Host D



Hop-By-Hop vs. End-to-End Layers

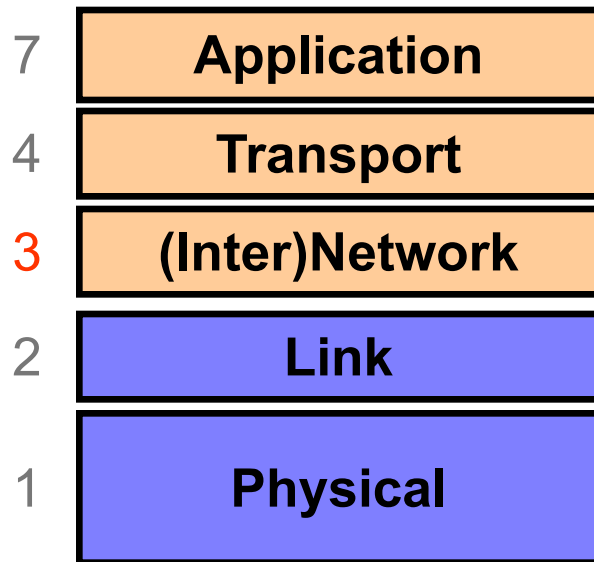
Host A communicates with Host D



E.g., HTTP over TCP over IP

Same Network / Transport / Application Layers
(Routers **ignore** Transport & Application layers)

Layer 3: (Inter)Network Layer (*IP*)



Bridges multiple “subnets” to provide *end-to-end* internet connectivity between nodes

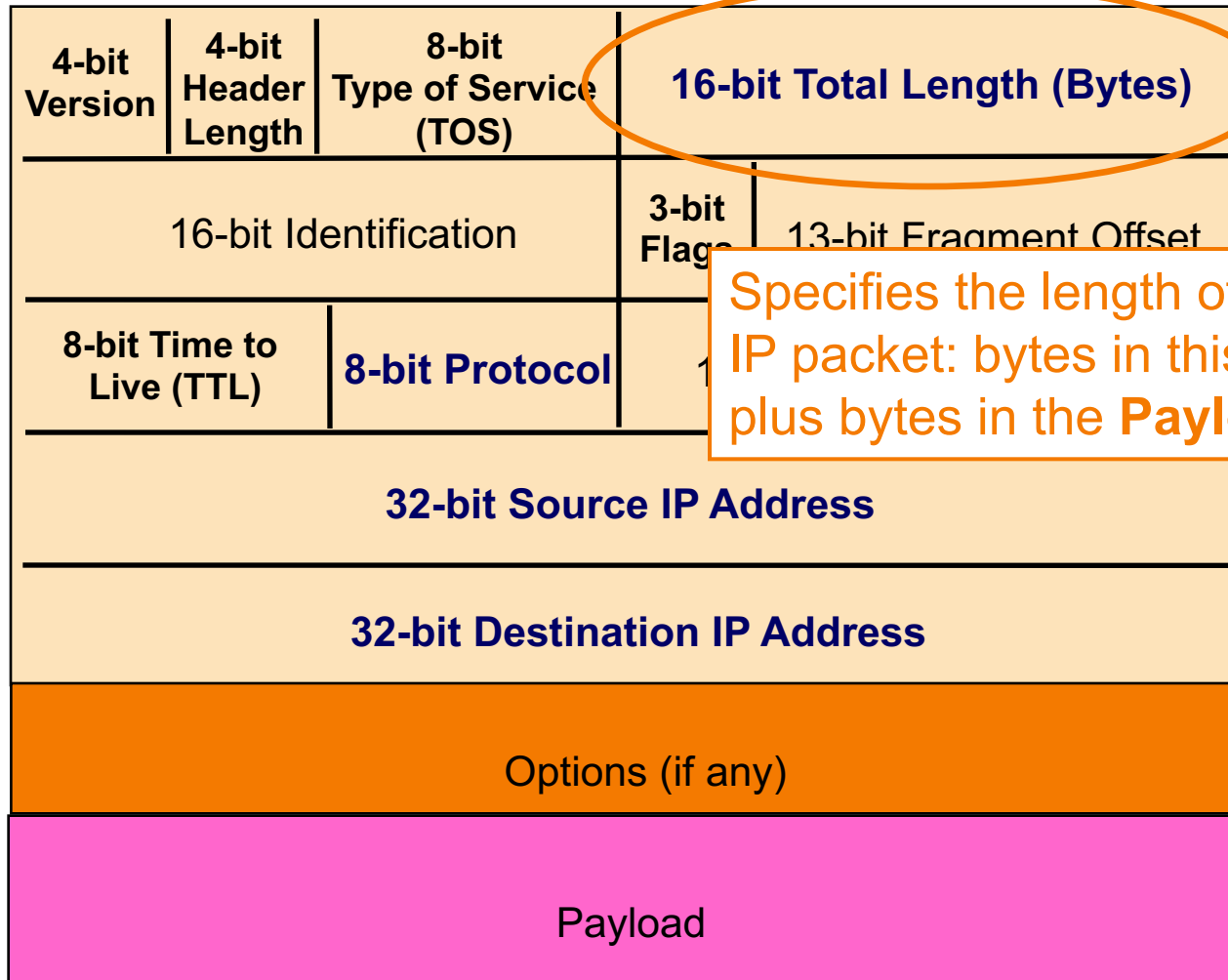
- Provides global addressing

Works across different link technologies

IP Packet Structure

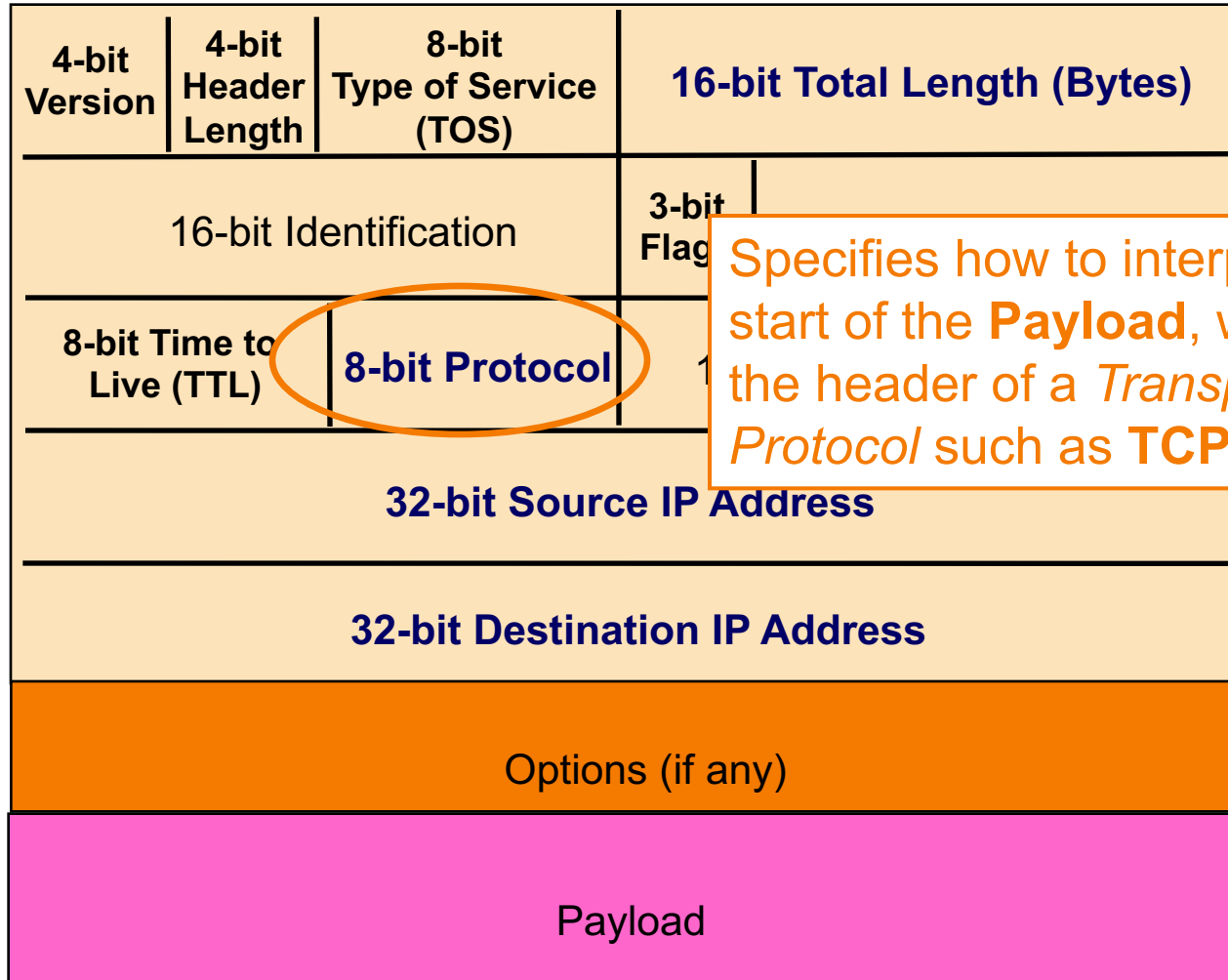
4-bit Version	4-bit Header Length	8-bit Type of Service (TOS)	16-bit Total Length (Bytes)	
16-bit Identification			3-bit Flags	13-bit Fragment Offset
8-bit Time to Live (TTL)	8-bit Protocol		16-bit Header Checksum	
32-bit Source IP Address				
32-bit Destination IP Address				
Options (if any)				
Payload				

IP Packet Structure



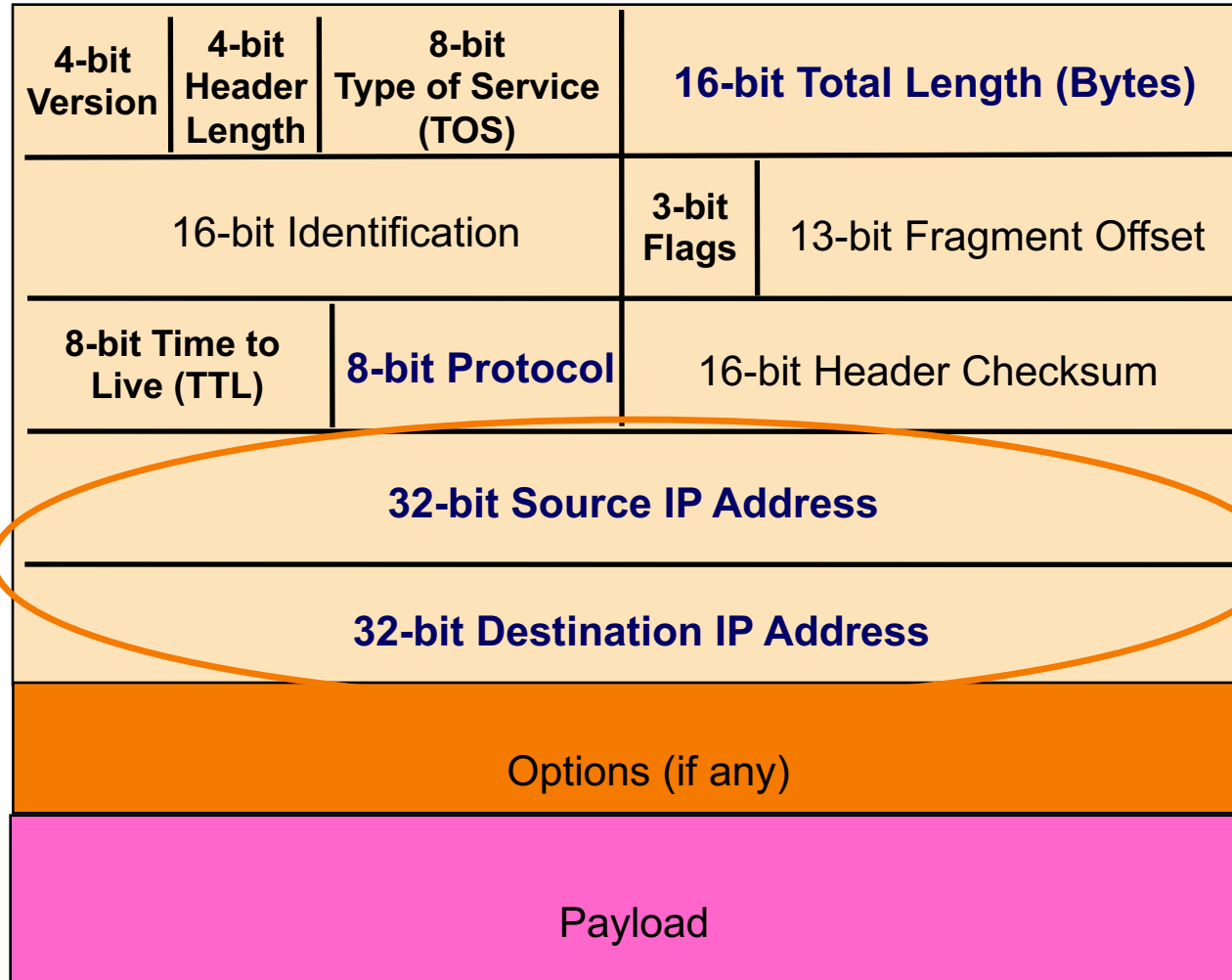
Specifies the length of the entire IP packet: bytes in this header plus bytes in the **Payload**

IP Packet Structure



Specifies how to interpret the start of the **Payload**, which is the header of a *Transport Protocol* such as **TCP** or **UDP**

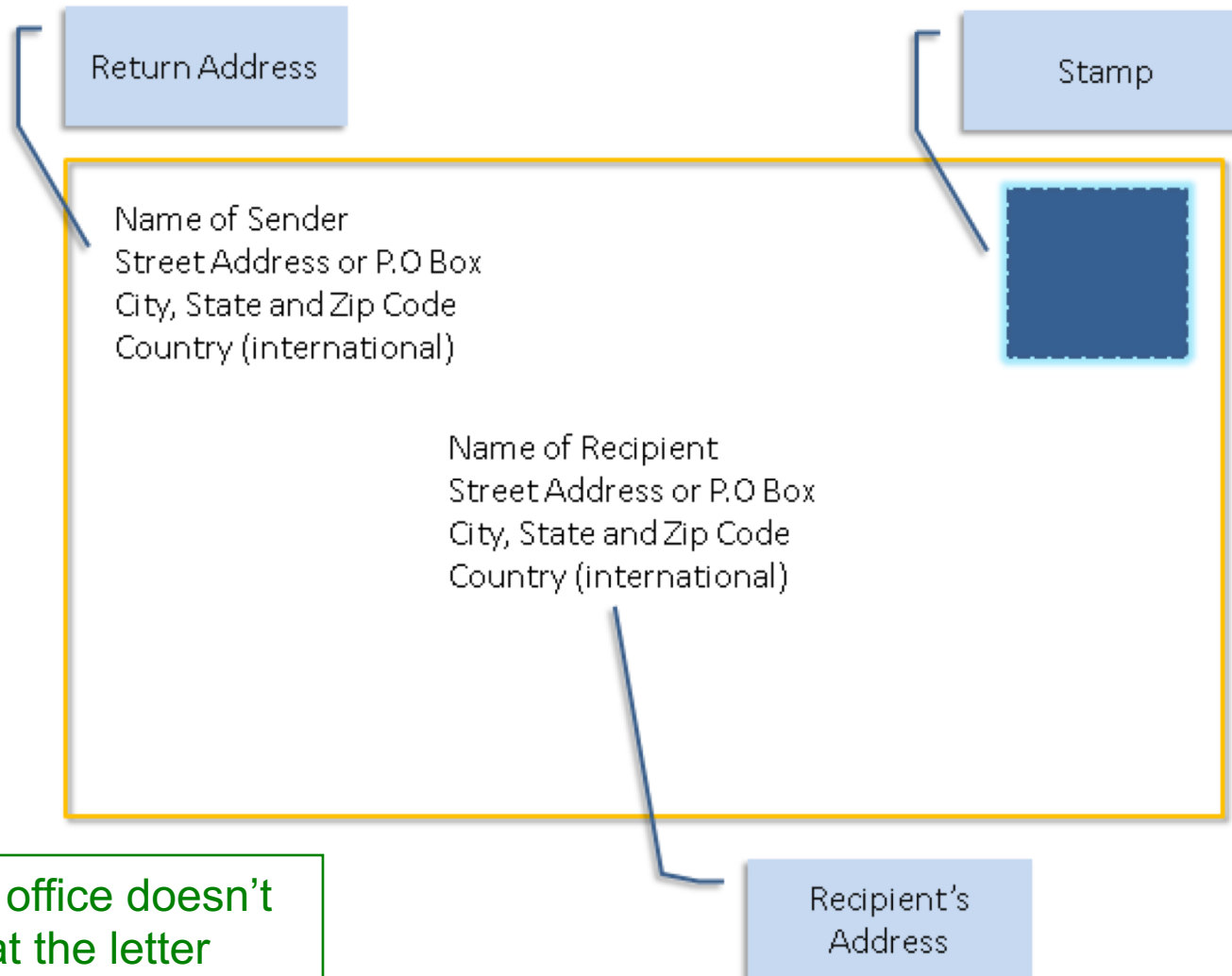
IP Packet Structure



IP Packet Header (Continued)

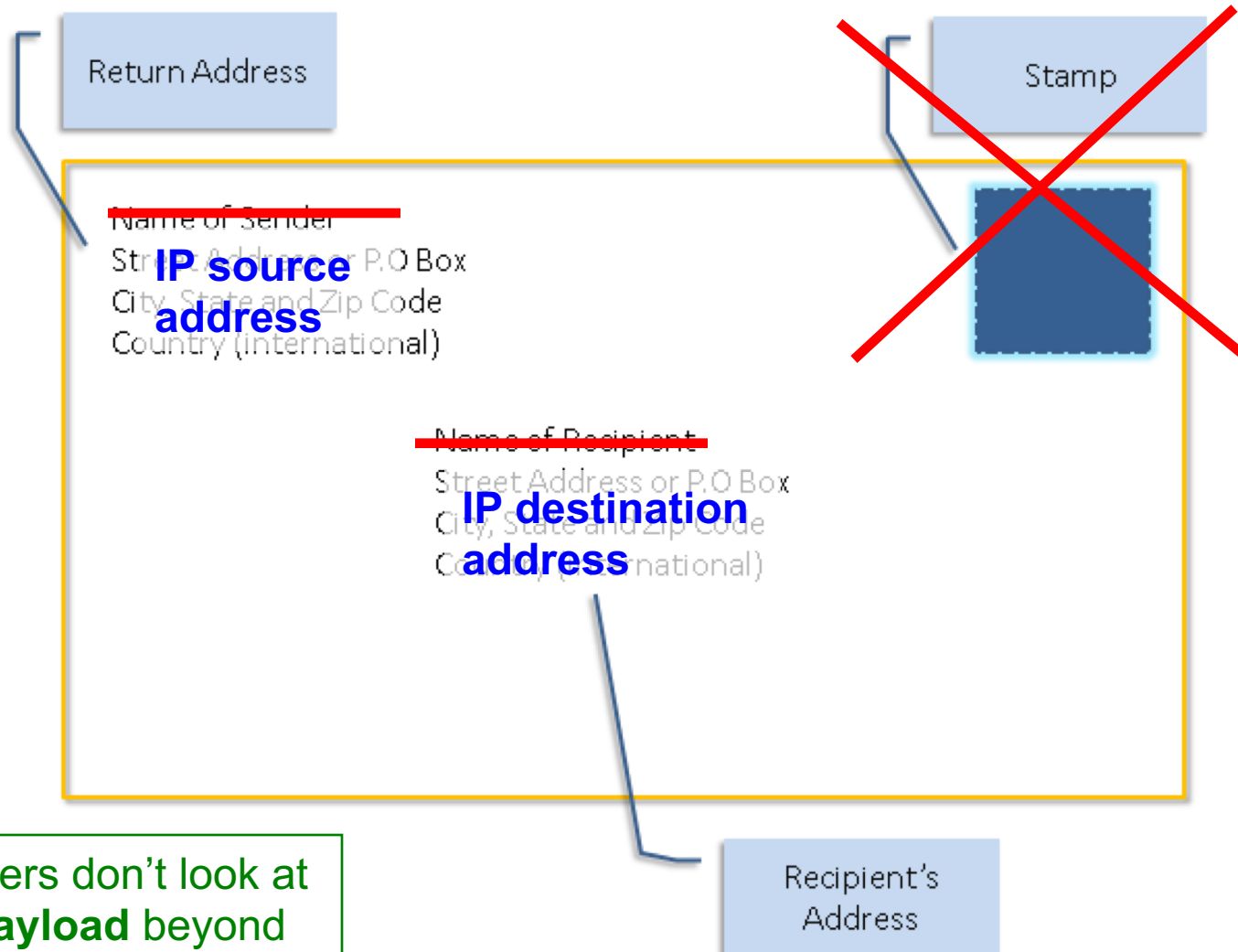
- Two IP addresses
 - Source IP address (32 bits)
 - Destination IP address (32 bits)
- Destination address
 - Unique **identifier/locator** for the receiving host
 - Allows each node to make forwarding decisions
- Source address
 - Unique identifier/locator for the sending host
 - Recipient can decide whether to accept packet
 - Enables recipient to send a reply back to source

Postal Envelopes:



(Post office doesn't
look at the letter
inside the envelope)

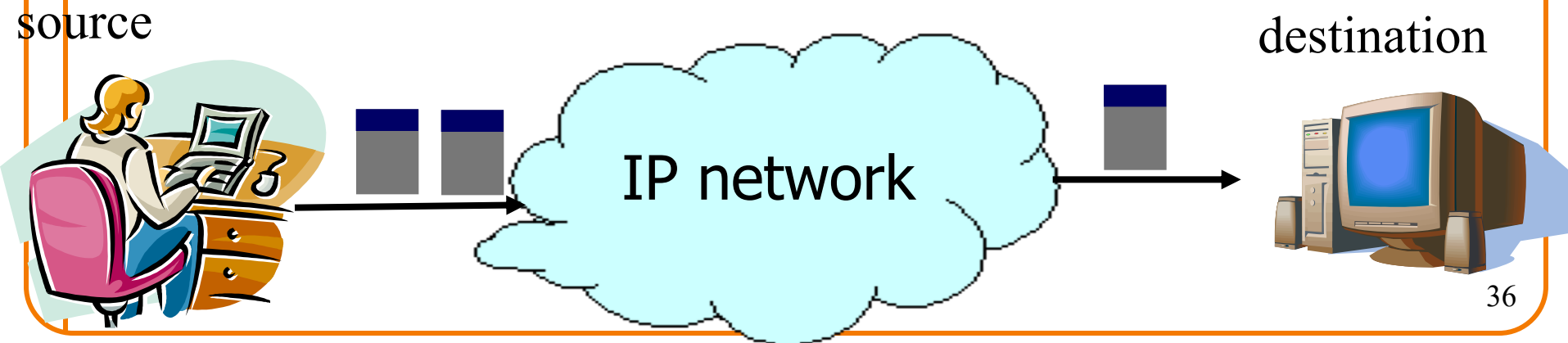
Analogy of IP to Postal Envelopes:



(Routers don't look at the **payload** beyond the IP header)

IP: “*Best Effort*” Packet Delivery

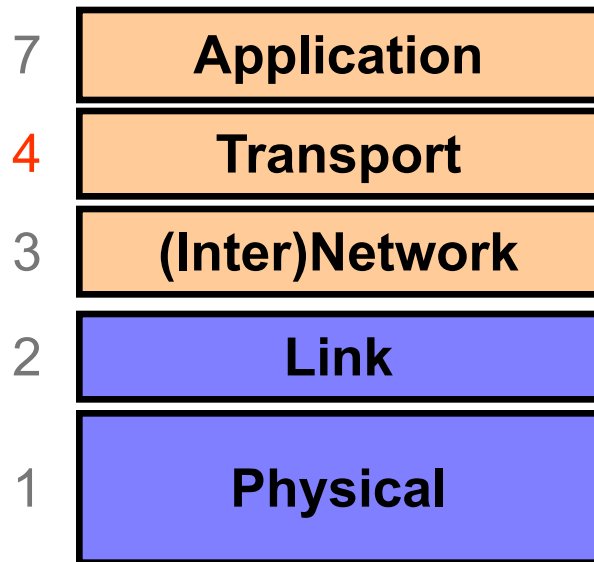
- Routers inspect destination address, locate “next hop” in forwarding table
 - Address = ~unique **identifier/locator** for the receiving host
- Only provides a “*I’ll give it a try*” delivery service:
 - Packets may be lost
 - Packets may be corrupted
 - Packets may be delivered out of order



“Best Effort” is Lame! What to do?

- It's the job of our Transport (layer 4) protocols to build services our apps need out of IP's modest layer-3 service

Layer 4: Transport Layer



*End-to-end communication
between processes*

Different services provided:
TCP = reliable *byte stream*
UDP = unreliable *datagrams*

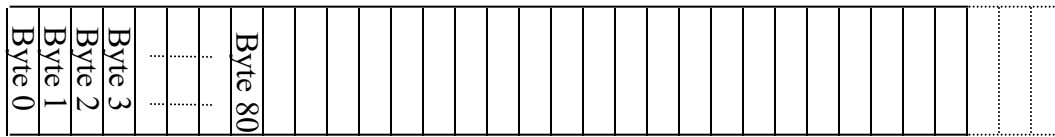
(Datagram = single packet message)

“Best Effort” is Lame! What to do?

- It's the job of our Transport (layer 4) protocols to build services our apps need out of IP's modest layer-3 service
- #1 workhorse: TCP (Transmission Control Protocol)
- Service provided by TCP:
 - Connection oriented (explicit set-up / tear-down)
 - o End hosts (processes) can have multiple concurrent long-lived communication
 - **Reliable**, in-order, *byte-stream* delivery
 - o Robust detection & retransmission of lost data

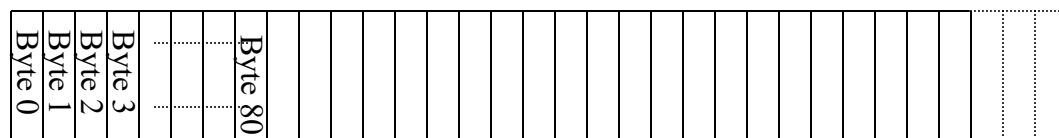
TCP “Bytestream” Service

Process A on host H1



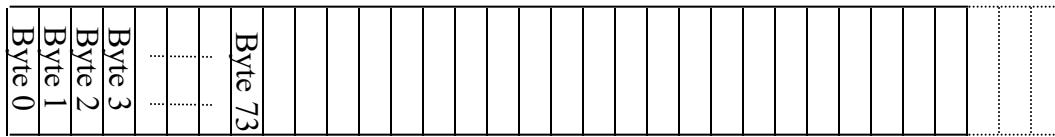
Hosts don't ever see packet boundaries, lost or corrupted packets, retransmissions, etc.

Process B
on host H2



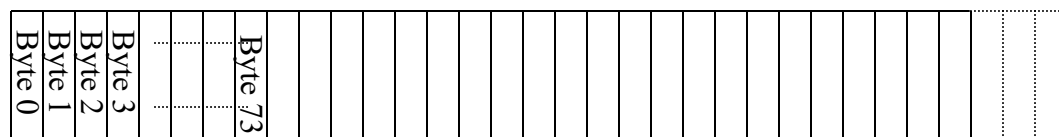
Bidirectional communication:

Process B on host H2

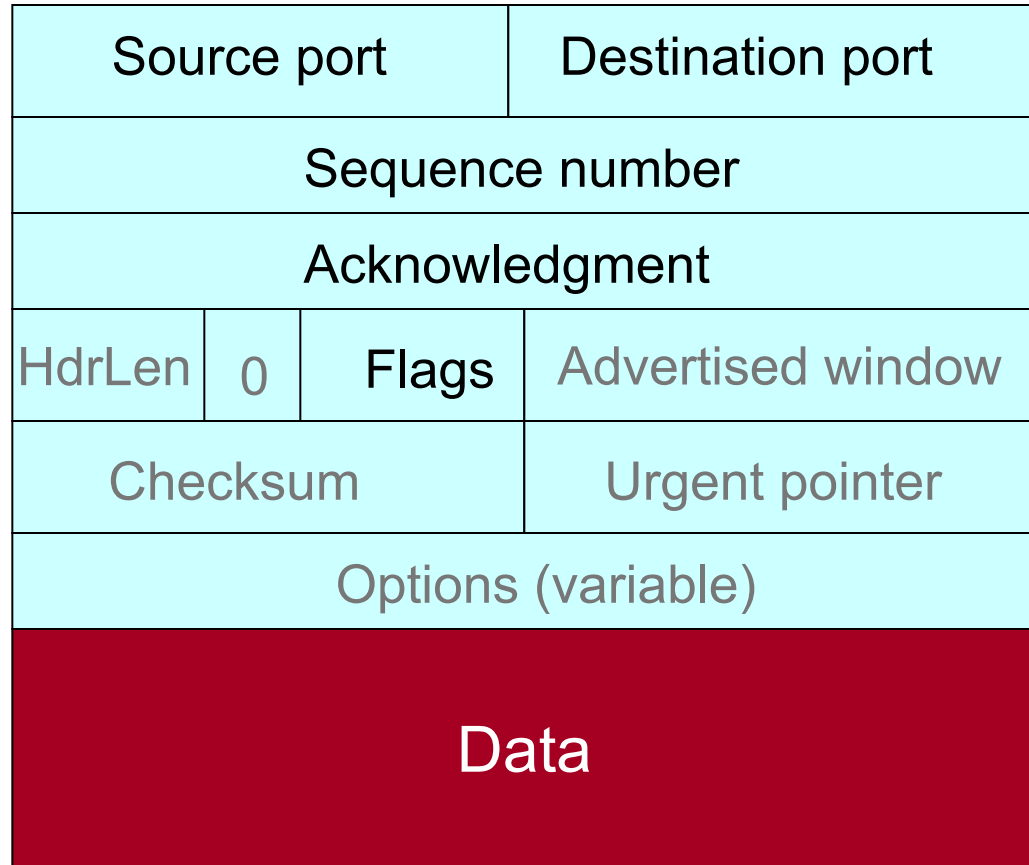


There are two separate **bytestreams**, one in each direction

Process A
on host H1

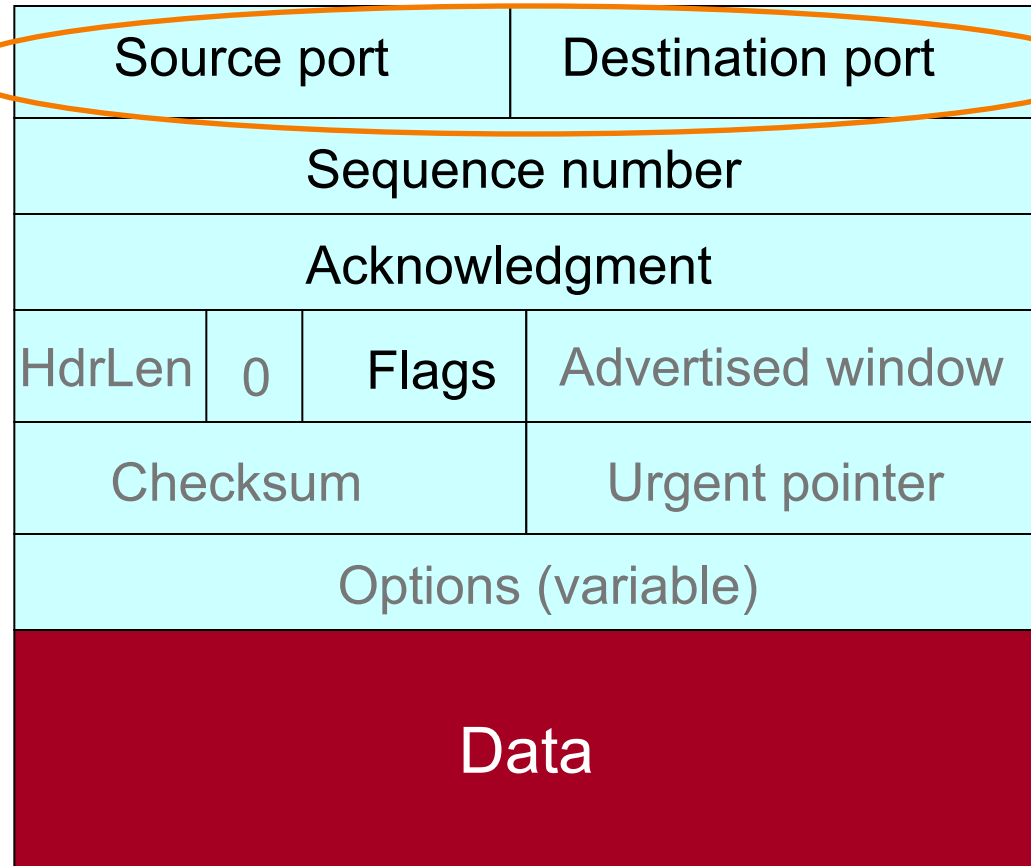


TCP Header



TCP Header

Ports are associated with OS processes



TCP Header

Ports are associated with OS processes

IP source & destination addresses plus TCP source and destination ports uniquely identifies a TCP connection

(Link Layer Header)

(IP Header)

Source port

Destination port

Sequence number

Acknowledgment

HdrLen

0

Flags

Advertised window

Checksum

Urgent pointer

Options (variable)

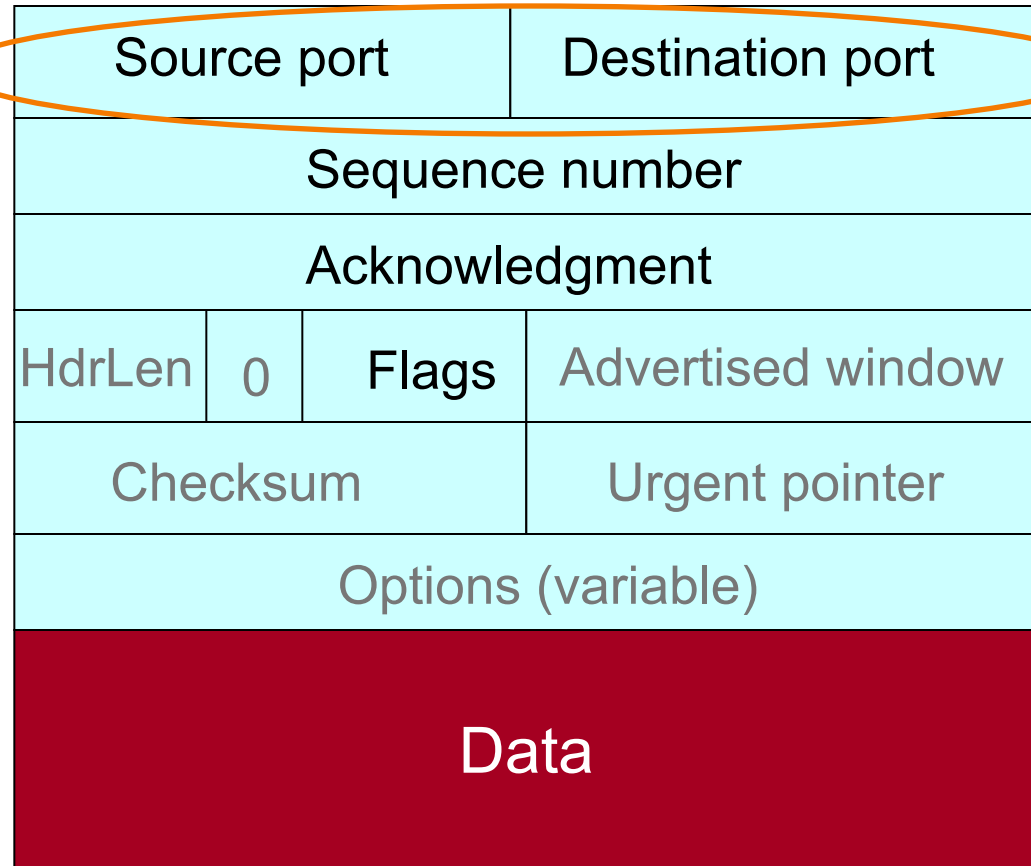
Data

TCP Header

Ports are associated with OS processes

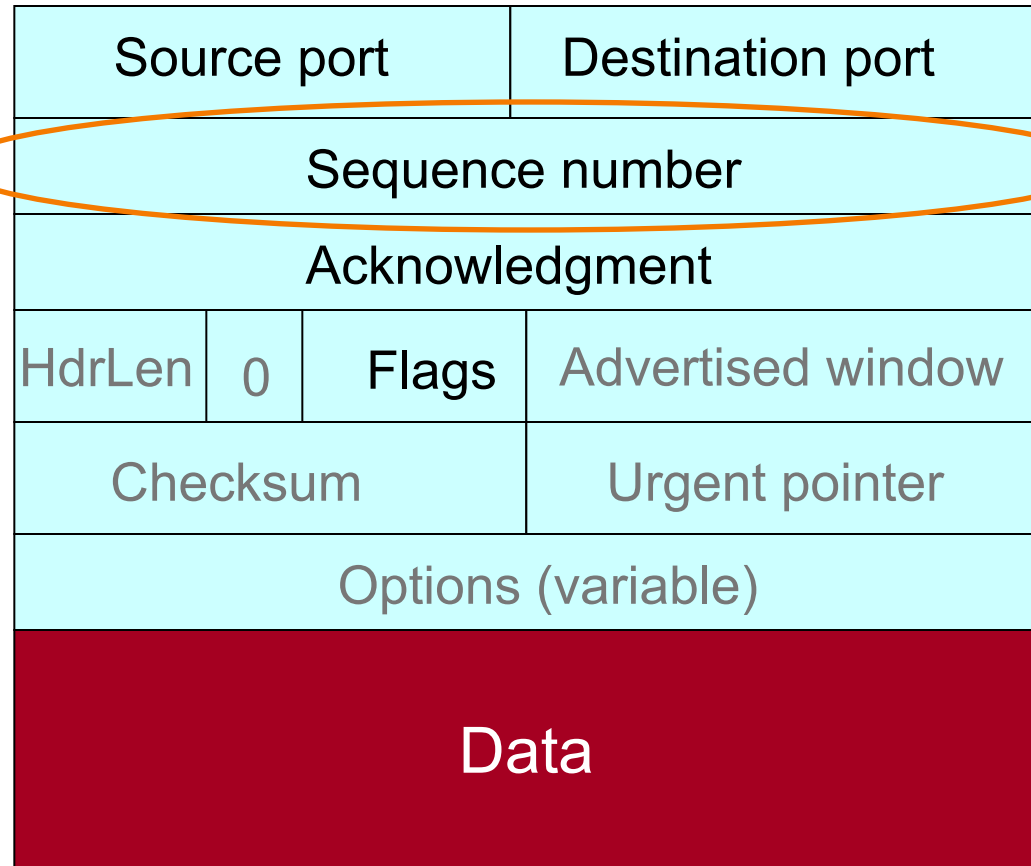
IP source & destination addresses plus TCP source and destination ports uniquely identifies a TCP connection

Some port numbers are “well known” / reserved
e.g. port 80 = HTTP



TCP Header

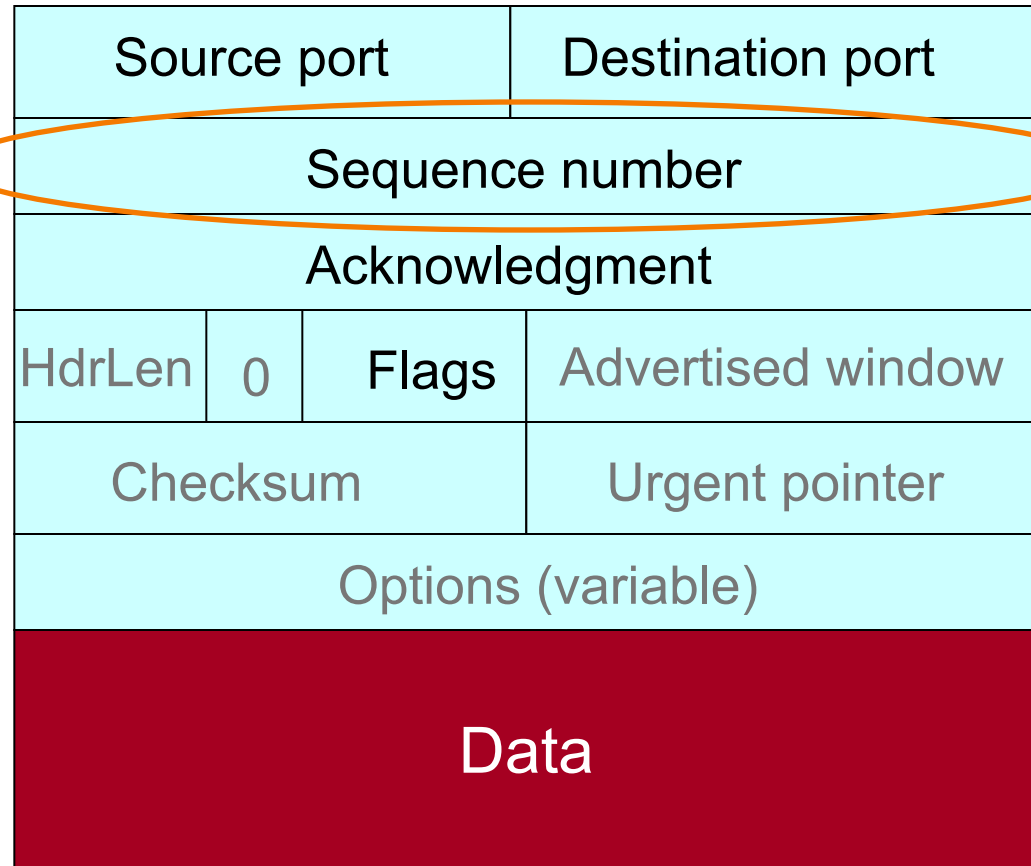
Starting sequence number (byte offset) of data carried in this packet



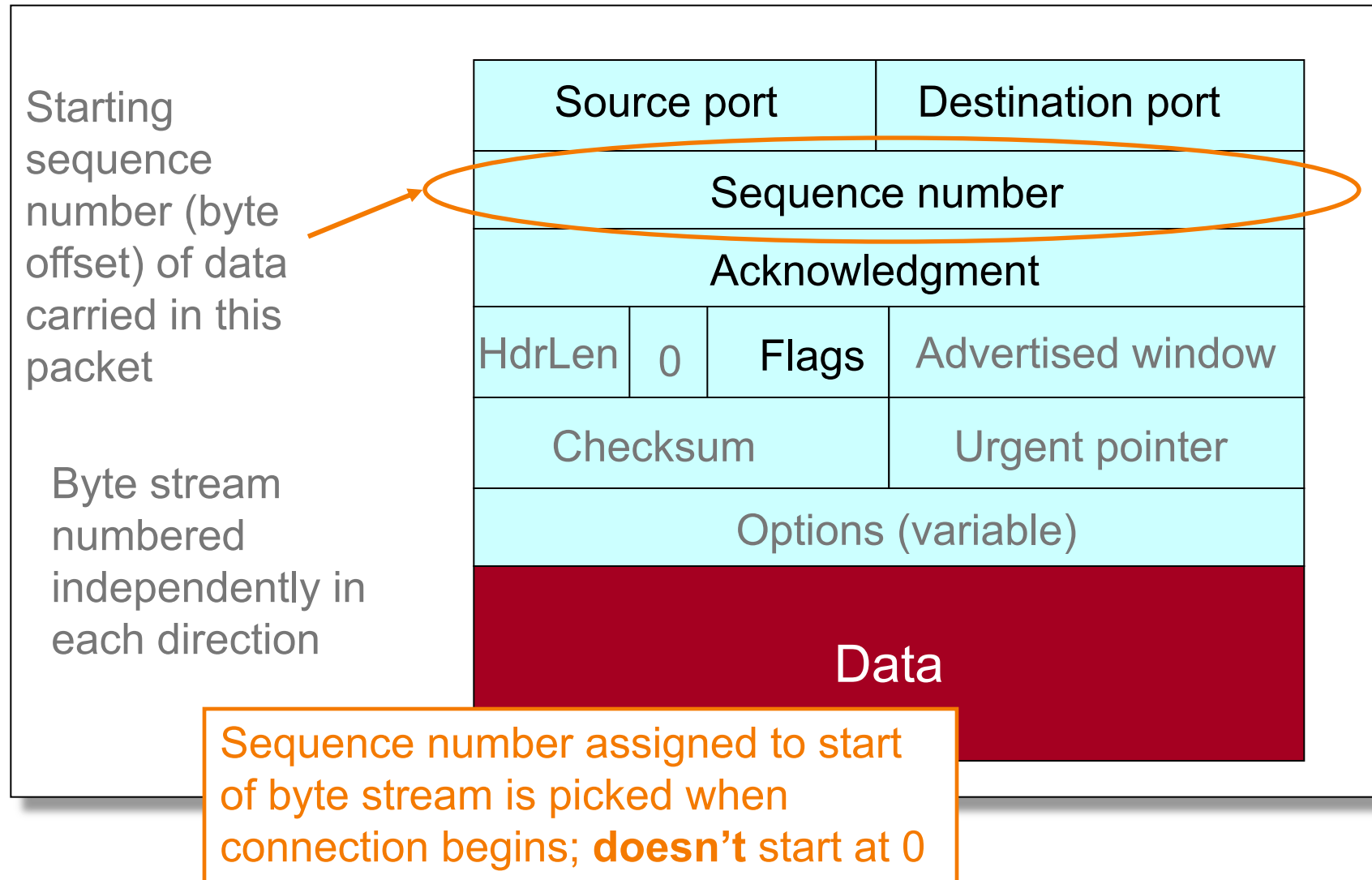
TCP Header

Starting sequence number (byte offset) of data carried in this packet

Byte streams numbered independently in each direction



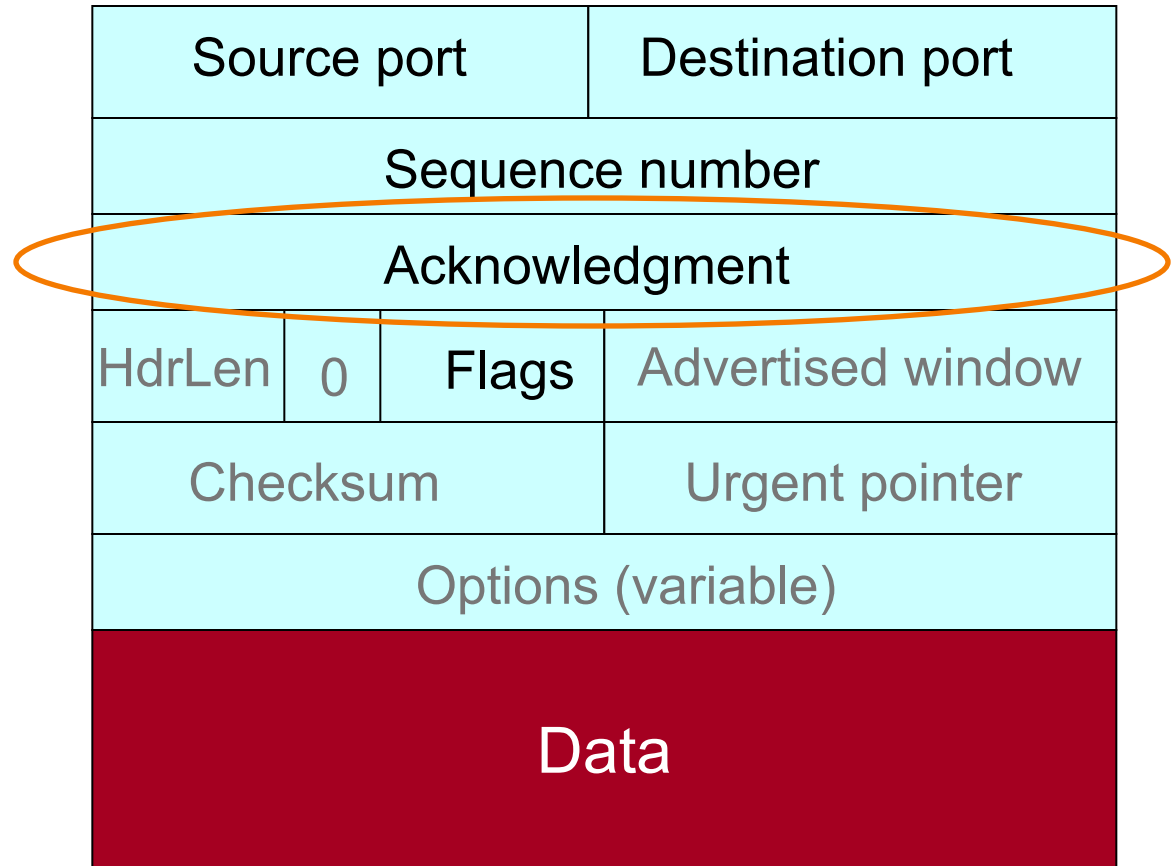
TCP Header



TCP Header

Acknowledgment gives seq # **just beyond** highest seq. received **in order**.

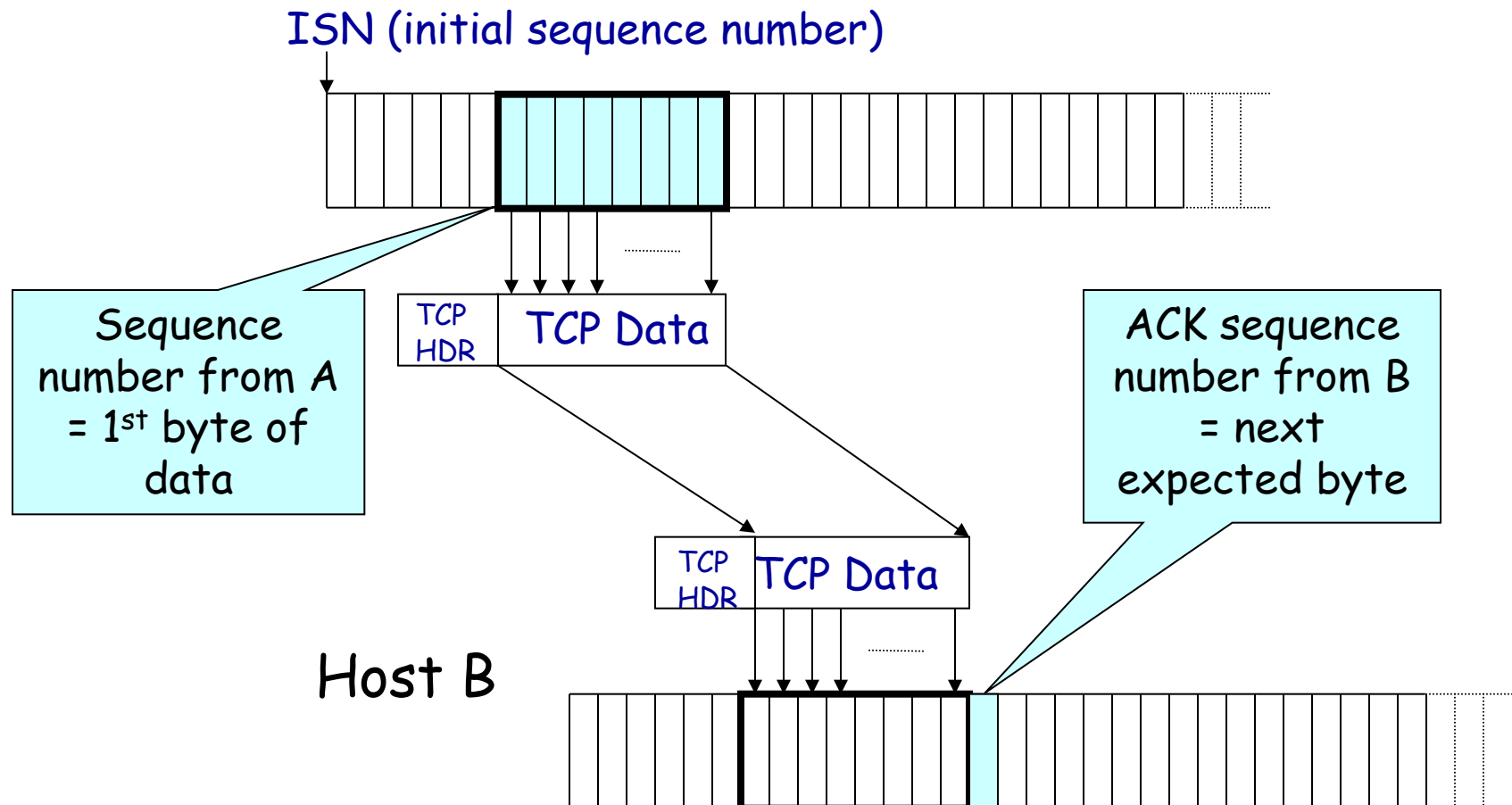
If sender sends **N** bytestream bytes starting at seq **S** then “ack” for it will be **S+N**.



Sequence Numbers

Host A

ISN (initial sequence number)

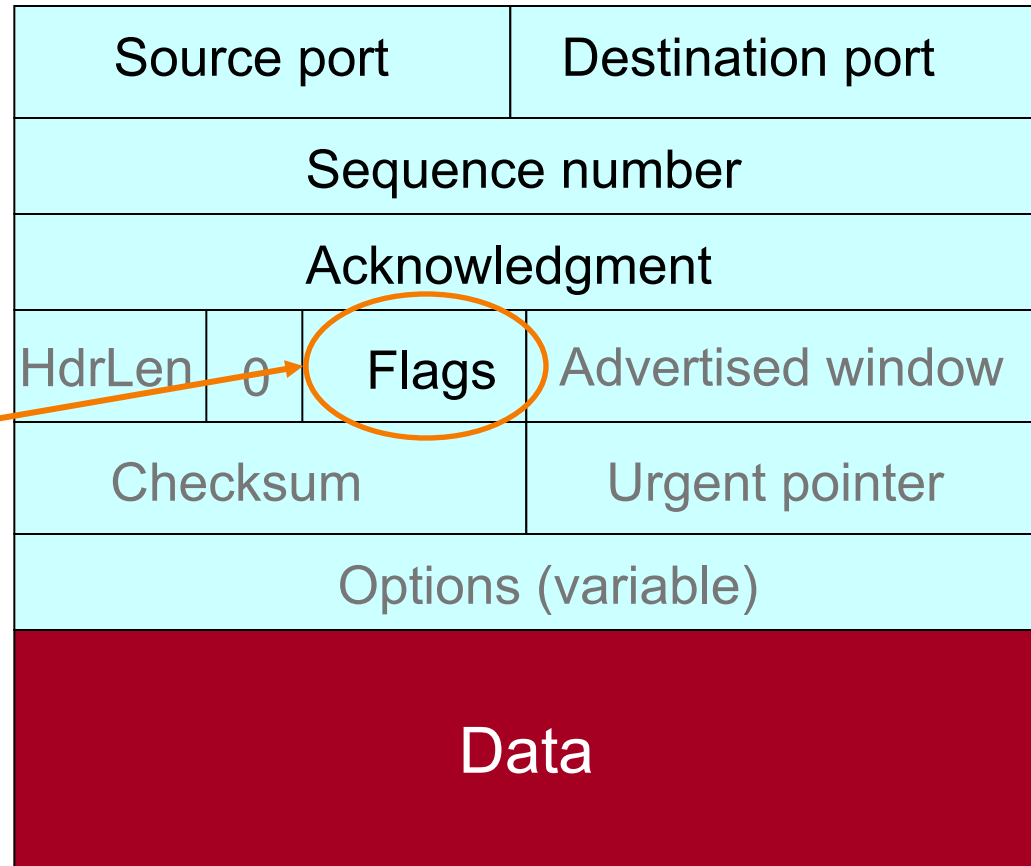


TCP Header

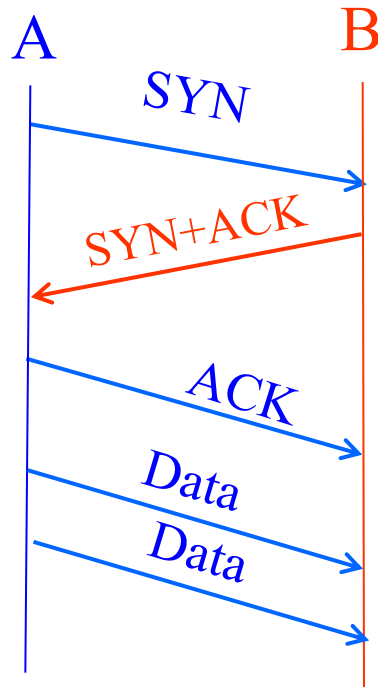
Uses include:

acknowledging
data (“**ACK**”)

setting up (“**SYN**”)
and closing
connections
 (“**FIN**” and
 “**RST**”)



Establishing a TCP Connection

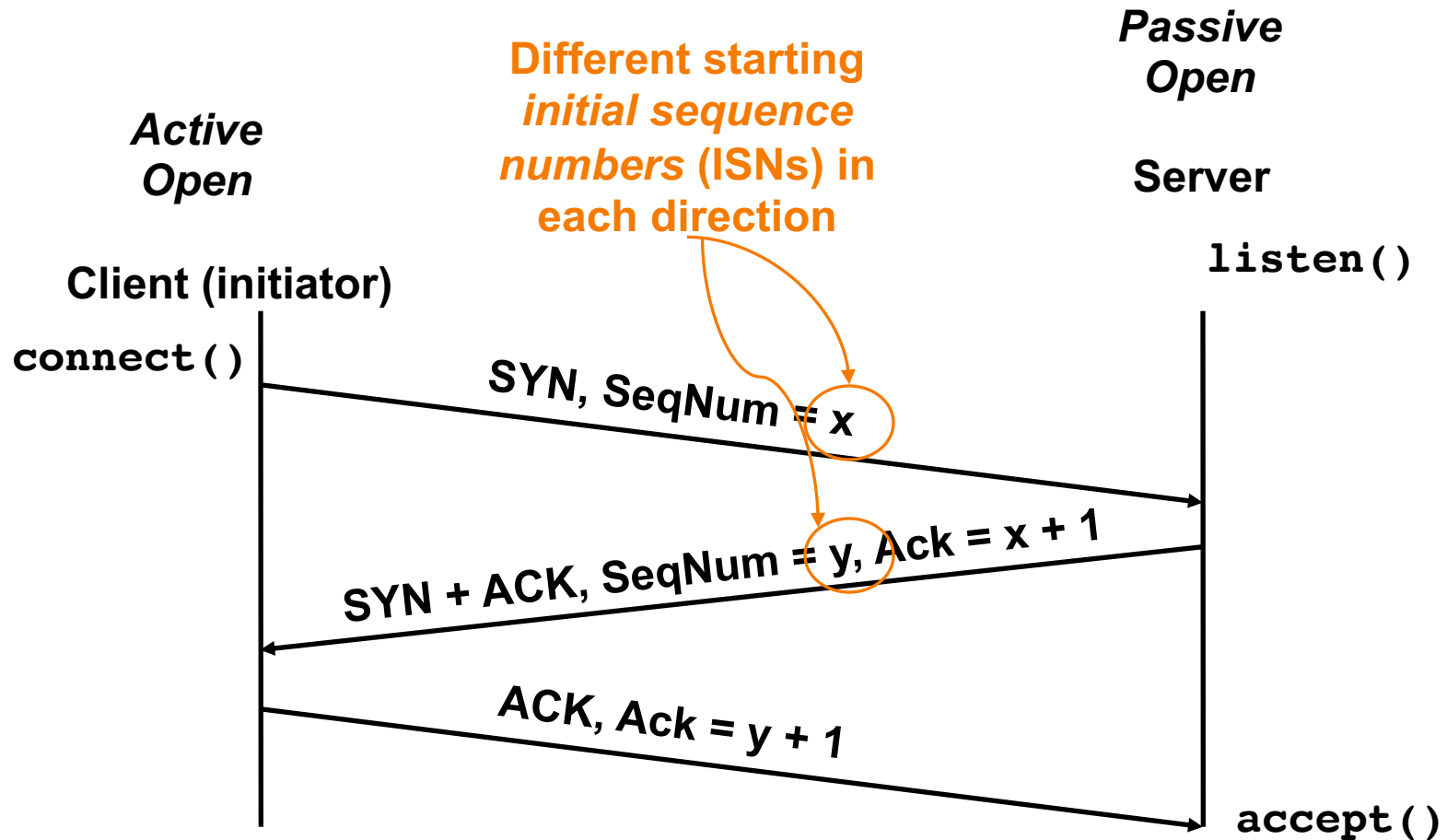


Each host tells its *Initial Sequence Number* (ISN) to the other host.

(Spec says to pick based on local clock)

- Three-way handshake to establish connection
 - Host A sends a **SYN** (open; “synchronize sequence numbers”) to host B
 - Host B returns a SYN acknowledgment (**SYN+ACK**)
 - Host A sends an **ACK** to acknowledge the SYN+ACK

Timing Diagram: 3-Way Handshaking



Fact about ...

Dave Patterson



- He taught various computer architecture courses
- Patterson & Henessy: classical computer architecture textbook
- Won the Turing award in 2017 (RAID, RISC-V)

UC Berkeley has the largest number of Turing award winners if you count by where they did their Turing award work

In his Turing lecture, Dave tried to distinguish himself from other Turing award winners:

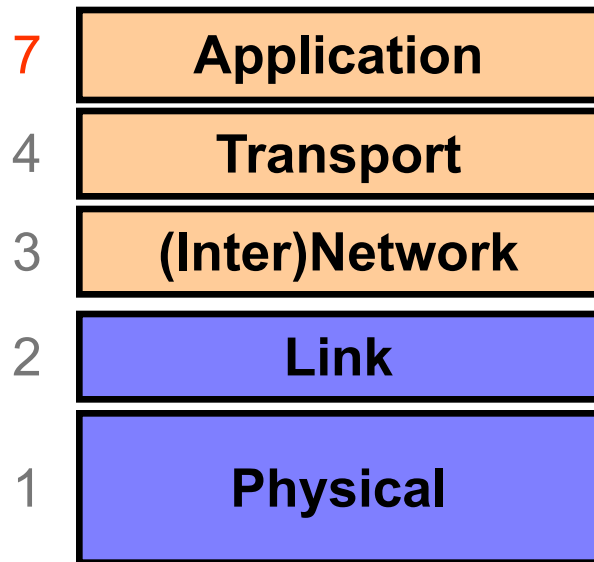
Dave is the strongest of them (literally).

Won California powerlifting championship in 2013 for his age range.



2min break

Layer 7: Application Layer



Communication of whatever you wish

Can use whatever transport(s) is convenient

Freely structured

E.g.:

Skype, SMTP (email),
HTTP (Web), Halo, BitTorrent

Web (HTTP) Request

Method **Resource** **HTTP version** **Headers**

↓ ↓ ↓ ↙

```
GET /index.html HTTP/1.1
Accept: image/gif, image/x-bitmap, image/jpeg, */*
Accept-Language: en
Connection: Keep-Alive
User-Agent: Mozilla/1.22 (compatible; MSIE 2.0; Windows 95)
Host: www.example.com
Referer: http://www.google.com?q=dingbats
```

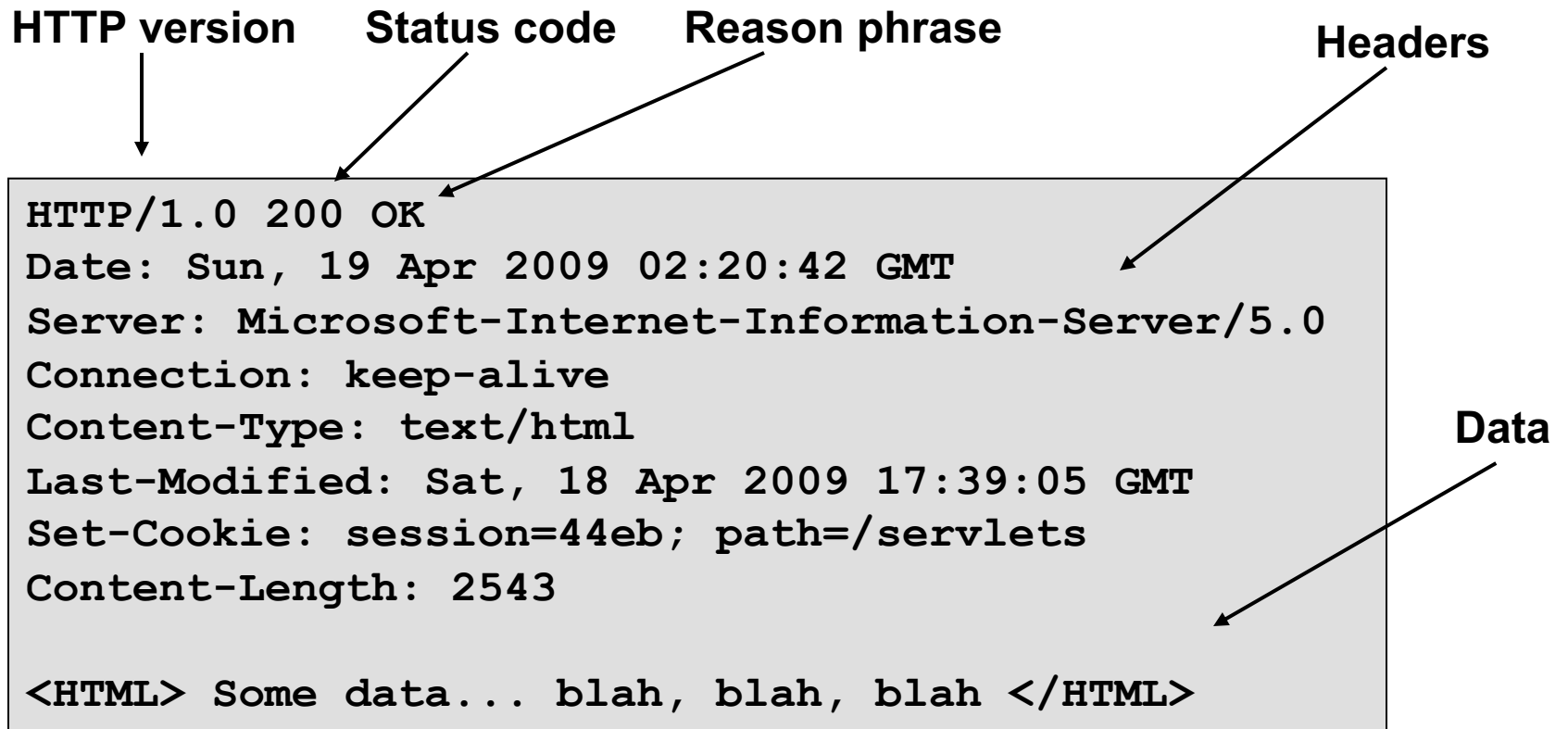
Blank line

Data (if POST; none for GET)

GET: download data.

POST: upload data.

Web (HTTP) Response



Host Names vs. IP addresses

- Host names
 - Examples: `www.cnn.com` and `bbc.co.uk`
 - Mnemonic name appreciated by **humans**
 - Variable length, full alphabet of characters
 - Provide little (if any) information about location
- IP addresses
 - Examples: `64.236.16.20` and `212.58.224.131`
 - Numerical address appreciated by **routers**
 - Fixed length, binary number
 - Hierarchical, related to host location

Networking Attacks: Link-, IP-, and TCP-layer attacks

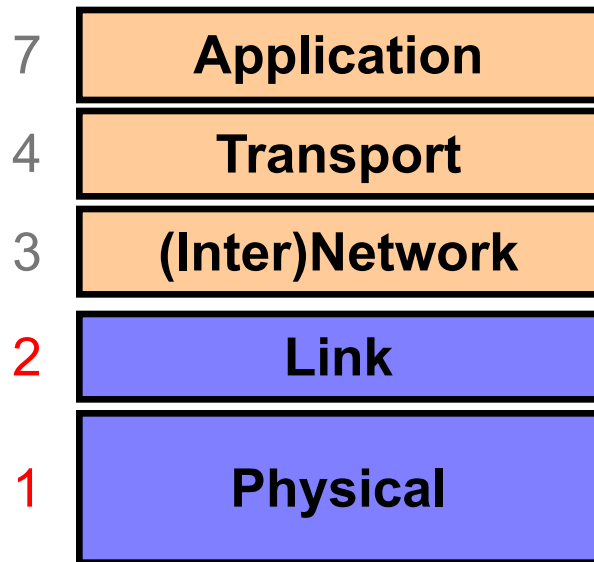
General Communication Security Goals: CIA

- Confidentiality:
 - No one can *read* our *data* / *communication* unless we want them to
- Integrity
 - No one can *manipulate* our *data* / *processing* / *communication* unless we want them to
- Availability
 - We can *access* our *data* / conduct our *processing* / use our *communication* capabilities when we want to

No security built in at the network level

- Everything you have seen in this lecture is just plaintext, to integrity attached to it so an attacker can easily spoof packets at multiple levels
- TLS will give application level security

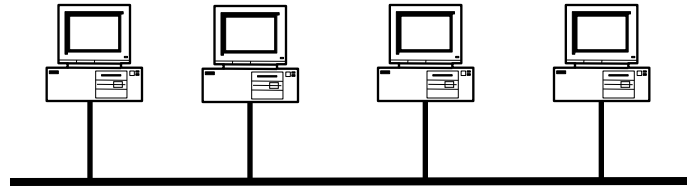
Layers 1 & 2: General Threats?



Framing and transmission of a collection of bits into individual **messages** sent across a single “subnetwork” (one physical technology)

Encoding **bits** to send them over a single **physical link**
e.g. patterns of
*voltage levels /
photon intensities /
RF modulation*

Link-layer threats



- Confidentiality: eavesdropping (aka sniffing)
- Integrity: injection of spoofed packets
- Availability: delete legit packets (e.g., jamming)

Eavesdropping

- For subnets using **broadcast** technologies (e.g., WiFi, some types of Ethernet), eavesdropping comes for “free”
 - Each attached system’s NIC (= Network Interface Card) can capture any communication on the subnet
 - Some handy tools for doing so
 - o tcpdump / windump (low-level ASCII printout)
 - o Wireshark (GUI for displaying 800+ protocols)

TCPDUMP: Packet Capture & ASCII Dumper

```
demo 2 % tcpdump -r all.trace2
```

```
reading from file all.trace2, link-type EN10MB (Ethernet)
```

```
21:39:37.772367 IP 10.0.1.9.60627 > 10.0.1.255.canon-bjnp2: UDP, length 16
```

```
21:39:37.772565 IP 10.0.1.9.62137 > all-systems.mcast.net.canon-bjnp2: UDP, length 16
```

```
21:39:39.923030 IP 10.0.1.9.17500 > broadcasthost.17500: UDP, length 130
```

```
21:39:39.923305 IP 10.0.1.9.17500 > 10.0.1.255.17500: UDP, length 130
```

```
21:39:42.286770 IP 10.0.1.13.61901 > star-01-02-pao1.facebook.com.http: Flags [S], seq 2523449627, win 65535, options [mss 1460,nop,wscale 3,nop,nop,TS val 429017455 ecr 0,sackOK,eol], length 0
```

```
21:39:42.309138 IP star-01-02-pao1.facebook.com.http > 10.0.1.13.61901: Flags [S.], seq 3585654832, ack 2523449628, win 14480, options [mss 1460,sackOK,TS val 1765826995 ecr 429017455,nop,wscale 9], length 0
```

```
21:39:42.309263 IP 10.0.1.13.61901 > star-01-02-pao1.facebook.com.http: Flags [.], ack 1, win 65535, options [nop,nop,TS val 429017456 ecr 1765826995], length 0
```

```
21:39:42.309796 IP 10.0.1.13.61901 > star-01-02-pao1.facebook.com.http: Flags [P.], seq 1:525, ack 1, win 65535, options [nop,nop,TS val 429017456 ecr 1765826995], length 524
```

```
21:39:42.326314 IP star-01-02-pao1.facebook.com.http > 10.0.1.13.61901: Flags [.], ack 525, win 31, options [nop,nop,TS val 1765827012 ecr 429017456], length 0
```

```
21:39:42.398814 IP star-01-02-pao1.facebook.com.http > 10.0.1.13.61901: Flags [P.], seq 1:535, ack 525, win 31, options [nop,nop,TS val 1765827083 ecr 429017456], length 534
```

```
21:39:42.398946 IP 10.0.1.13.61901 > star-01-02-pao1.facebook.com.http: Flags [.], ack 535, win 65535, options [nop,nop,TS val 429017457 ecr 1765827083], length 0
```

```
21:39:44.838031 IP 10.0.1.9.54277 > 10.0.1.255.canon-bjnp2: UDP, length 16
```

```
21:39:44.838213 IP 10.0.1.9.62896 > all-systems.mcast.net.canon-bjnp2: UDP, length 16
```

Wireshark: GUI for Packet Capture/Exam.

The image shows the Wireshark 1.6.2 interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Tools, Internals, and Help. Below the menu is a toolbar with various icons for file operations, packet capture, and analysis. A filter bar is present with a dropdown menu and buttons for 'Expression...', 'Clear', and 'Apply'. The main packet list table displays 13 captured packets. Packet 10 is selected, showing its details in the packet details pane. The packet details pane shows the following structure:

- Frame 10: 600 bytes on wire (4800 bits), 600 bytes captured (4800 bits)
- Ethernet II, Src: Apple_fe:aa:41 (00:25:00:fe:aa:41), Dst: Apple_41:eb:00 (e4:ce:8f:41:eb:00)
- Internet Protocol Version 4, Src: 31.13.75.23 (31.13.75.23), Dst: 10.0.1.13 (10.0.1.13)
- Transmission Control Protocol, Src Port: http (80), Dst Port: 61901 (61901), Seq: 1, Ack: 525, Len: 534
- Hypertext Transfer Protocol

The packet bytes pane at the bottom shows the raw data in hexadecimal and ASCII. The ASCII column shows the following text:

```
...A...% ...A..E
.Jg...X. ....K...
...P.... .l.h.(.
.../.... .i@b...
IpHTTP/1 .1 302 F
```

The status bar at the bottom indicates: File: "/Users/vern/tmp/all.trace2" 23... Packets: 13 Displayed: 13 Marked: 0 Load time: 0:00.109 Profile: Default

Wireshark: GUI for Packet Capture/Exam.

The image displays the Wireshark 1.6.2 graphical user interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Tools, Internals, and Help. Below the menu is a toolbar with various icons for file operations, network analysis, and packet capture. The main window is divided into three panes:

- Packet List Pane:** Shows a list of captured packets. The selected packet is packet 10, which is an HTTP GET request from 31.13.75.23 to 10.0.1.13.
- Packet Details Pane:** Provides a hierarchical view of the selected packet's structure. It shows the Ethernet II header, Internet Protocol Version 4 header, and the Transmission Control Protocol (TCP) segment. The TCP segment is an ACK with sequence number 535 and acknowledgment number 525.
- Packet Bytes Pane:** Displays the raw data of the selected packet in hexadecimal and ASCII. The data is the body of the HTTP response, starting with "IpHTTP/1.1 302 F".

The status bar at the bottom indicates that 13 packets are displayed, with a load time of 0:00.109 and a default profile.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.1.9	10.0.1.255	BJNP	58	Printer Command: Unknown code (2)
2	0.000198	10.0.1.9	224.0.0.1	BJNP	58	Printer Command: Unknown code (2)
3	2.150663	10.0.1.9	255.255.255.255	DB-LSP-D	172	Dropbox LAN sync Discovery Protocol
4	2.150938	10.0.1.9	10.0.1.255	DB-LSP-D	172	Dropbox LAN sync Discovery Protocol
5	4.514403	10.0.1.13	31.13.75.23	TCP	78	61901 > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=8 TSval=4290
6	4.536771	31.13.75.23	10.0.1.13	TCP	74	http > 61901 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK
7	4.536896	10.0.1.13	31.13.75.23	TCP	66	61901 > http [ACK] Seq=1 Ack=1 Win=524280 Len=0 TSval=429017456 T
8	4.537429	10.0.1.13	31.13.75.23	HTTP	590	GET / HTTP/1.1
9	4.553947	31.13.75.23	10.0.1.13	TCP	66	http > 61901 [ACK] Seq=1 Ack=525 Win=15872 Len=0 TSval=1765827012
10	4.626447	31.13.75.23	10.0.1.13	HTTP	600	HTTP/1.1 302 Found
11	4.626579	10.0.1.13	31.13.75.23	TCP	66	61901 > http [ACK] Seq=525 Ack=535 Win=524280 Len=0 TSval=4290174
12	7.065664	10.0.1.9	10.0.1.255	BJNP	58	Printer Command: Unknown code (2)
13	7.065846	10.0.1.9	224.0.0.1	BJNP	58	Printer Command: Unknown code (2)

Frame 10: 600 bytes on wire (4800 bits), 600 bytes captured (4800 bits)

Ethernet II, Src: Apple_fe:aa:41 (00:25:00:fe:aa:41), Dst: Apple_41:eb:00 (e4:ce:8f:41:eb:00)

Internet Protocol Version 4, Src: 31.13.75.23 (31.13.75.23), Dst: 10.0.1.13 (10.0.1.13)

Transmission Control Protocol, Src Port: http (80), Dst Port: 61901 (61901), Seq: 1, Ack: 525, Len: 534

Source port: http (80)

Destination port: 61901 (61901)

[Stream index: 0]

Sequence number: 1 (relative sequence number)

[Next sequence number: 535 (relative sequence number)]

Acknowledgement number: 525 (relative ack number)

Header length: 32 bytes

Flags: 0x18 (PSH, ACK)

Window size value: 31

[Calculated window size: 15872]

[Window size scaling factor: 512]

Checksum: 0xf42f [validation disabled]

0000 e4 ce 8f 41 eb 00 00 25 00 fe aa 41 08 00 45 20 ...A...% ...A..E

0010 02 4a 67 be 00 00 58 06 83 9f 1f 0d 4b 17 0a 00 ...Jg...X.K...

0020 01 0d 00 50 f1 cd d5 b8 c0 31 96 68 cb 28 80 18 ...P....l.h.(...

0030 00 1f f4 2f 00 00 01 01 08 0a 69 40 62 0b 19 92 .../....i@b...

0040 49 70 48 54 54 50 2f 31 2e 31 20 33 30 32 20 46 IpHTTP/1.1 302 F

Frame (frame), 600 bytes | Packets: 13 Displayed: 13 Marked: 0 Load time: 0:00.109 | Profile: Default

Wireshark: GUI for Packet Capture/Exam.

all.trace2 [Wireshark 1.6.2]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.1.9	10.0.1.255	BJNP	58	Printer Command: Unknown code (2)
2	0.000198	10.0.1.9	224.0.0.1	BJNP	58	Printer Command: Unknown code (2)
3	2.150663	10.0.1.9	255.255.255.255	DB-LSP-D	172	Dropbox LAN sync Discovery Protocol
4	2.150938	10.0.1.9	10.0.1.255	DB-LSP-D	172	Dropbox LAN sync Discovery Protocol
5	4.514403	10.0.1.13	31.13.75.23	TCP	78	61901 > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=8 TSval=4290
6	4.536771	31.13.75.23	10.0.1.13	TCP	74	http > 61901 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK
7	4.536896	10.0.1.13	31.13.75.23	TCP	66	61901 > http [ACK] Seq=1 Ack=1 Win=524280 Len=0 TSval=429017456 T
8	4.537429	10.0.1.13	31.13.75.23	HTTP	590	GET / HTTP/1.1
9	4.553947	31.13.75.23	10.0.1.13	TCP	66	http > 61901 [ACK] Seq=1 Ack=525 Win=15872 Len=0 TSval=1765827012
10	4.626447	31.13.75.23	10.0.1.13	HTTP	600	HTTP/1.1 302 Found
11	4.626579	10.0.1.13	31.13.75.23	TCP	66	61901 > http [ACK] Seq=525 Ack=535 Win=524280 Len=0 TSval=4290174
12	7.065664	10.0.1.9	10.0.1.255	BJNP	58	Printer Command: Unknown code (2)
13	7.065846	10.0.1.9	224.0.0.1	BJNP	58	Printer Command: Unknown code (2)

Frame 10: 600 bytes on wire (4800 bits), 600 bytes captured (4800 bits)

Ethernet II, Src: Apple_fe:aa:41 (00:25:00:fe:aa:41), Dst: Apple_41:eb:00 (e4:ce:8f:41:eb:00)

Internet Protocol Version 4, Src: 31.13.75.23 (31.13.75.23), Dst: 10.0.1.13 (10.0.1.13)

Transmission Control Protocol, Src Port: http (80), Dst Port: 61901 (61901), Seq: 1, Ack: 525, Len: 534

Source port: http (80)

Destination port: 61901 (61901)

[Stream index: 0]

Sequence number: 1 (relative sequence number)

[Next sequence number: 535 (relative sequence number)]

Acknowledgement number: 525 (relative ack number)

Header length: 32 bytes

Flags: 0x18 (PSH, ACK)

Window size value: 31

[Calculated window size: 15872]

[Window size scaling factor: 512]

Checksum: 0xf42f [validation disabled]

0000 e4 ce 8f 41 eb 00 00 25 00 fe aa 41 08 00 45 20 ...A...% ...A..E

0010 02 4a 67 be 00 00 58 06 83 9f 1f 0d 4b 17 0a 00 .Jg...X.K...

0020 01 0d 00 50 f1 cd d5 b8 c0 31 96 68 cb 28 80 18 ...P.... .l.h.(...

0030 00 1f f4 2f 00 00 01 01 08 0a 69 40 62 0b 19 92 .../.... .i@b...

0040 49 70 48 54 54 50 2f 31 2e 31 20 33 30 32 20 46 IpHTTP/1 .l 302 F

Frame (frame), 600 bytes

Packets: 13 Displayed: 13 Marked: 0 Load time: 0:00.109

Profile: Default

Link-Layer Threat: Disruption

- If attacker sees a packet he doesn't like, he can jam it (integrity)
- Attacker can also **overwhelm** link-layer signaling, e.g., jam WiFi's RF (denial-of-service)

Link-Layer Threat: Disruption

- If attacker sees a packet he doesn't like, he can jam it (integrity)
- Attacker can also **overwhelm** link-layer signaling, e.g., jam WiFi's RF (denial-of-service)
- There's also the heavy-handed approach ...

Sabotage attacks knock out phone service

Nanette Asimov, Ryan Kim, Kevin Fagan, Chronicle Staff Writers
Friday, April 10, 2009

PRINT E-MAIL SHARE COMMENTS (477) FONT | SIZE: - +

(04-10) 04:00 PDT SAN JOSE --

Police are hunting for vandals who chopped fiber-optic cables and killed landlines, cell phones and Internet service for tens of thousands of people in Santa Clara, Santa Cruz and San Benito counties on Thursday.

IMAGES



View More Images

MORE NEWS

- Toyota seeks damage control, in public and private 02.09.10
- Snow shuts down federal government, life goes on 02.09.10
- Iran boosts nuclear enrichment, drawing warnings 02.09.10

"I pity the individuals who have done this," said San Jose Police Chief Rob Davis.

Ten fiber-optic cables carrying were cut at four locations in the predawn darkness. Residential and business customers quickly found that telephone service was perhaps more laced into their everyday needs than they thought. Suddenly they couldn't draw out money, send text messages, check e-mail or Web sites, call anyone for help, or even check on friends or relatives down the road.

Several people had to be driven to hospitals because they were unable to summon ambulances. Many businesses lapsed into idleness for hours, without the ability to contact associates or customers.

More than 50,000 landline customers lost service - some were residential, others were business lines that needed the connections for ATMs, Internet and bank card transactions. One line alone could affect hundreds of users.

The sabotage essentially froze operations in parts of the three counties at hospitals, stores, banks and police and fire departments that rely on 911 calls, computerized medical records, ATMs and credit and debit cards.

The full extent of the havoc might not be known for days, emergency officials said as they finished repairing the damage late Thursday.

Whatever the final toll, one thing is certain: Whoever did this is in a world of trouble if he, she or they get caught.

NEWS | LOCAL BEAT

\$250K Reward Out for Vandals Who Cut AT&T Lines

Local emergency declared during outage

By LORI PREUITT

Updated 2:12 PM PST, Fri, Apr 10, 2009

PRINT EMAIL SHARE BUZZ UP! TWITTER FACEBOOK



AT&T is now offering a \$250,000 reward for information leading to the arrest of whoever is responsible for severing lines fiber optic cables in San Jose tha left much of the area without phone or cell service Thursday.

John Britton of AT&T said the reward is the largest ever offered by the company.

Link-Layer Threat: Spoofing

- Attacker can inject spoofed packets, and lie about the source address

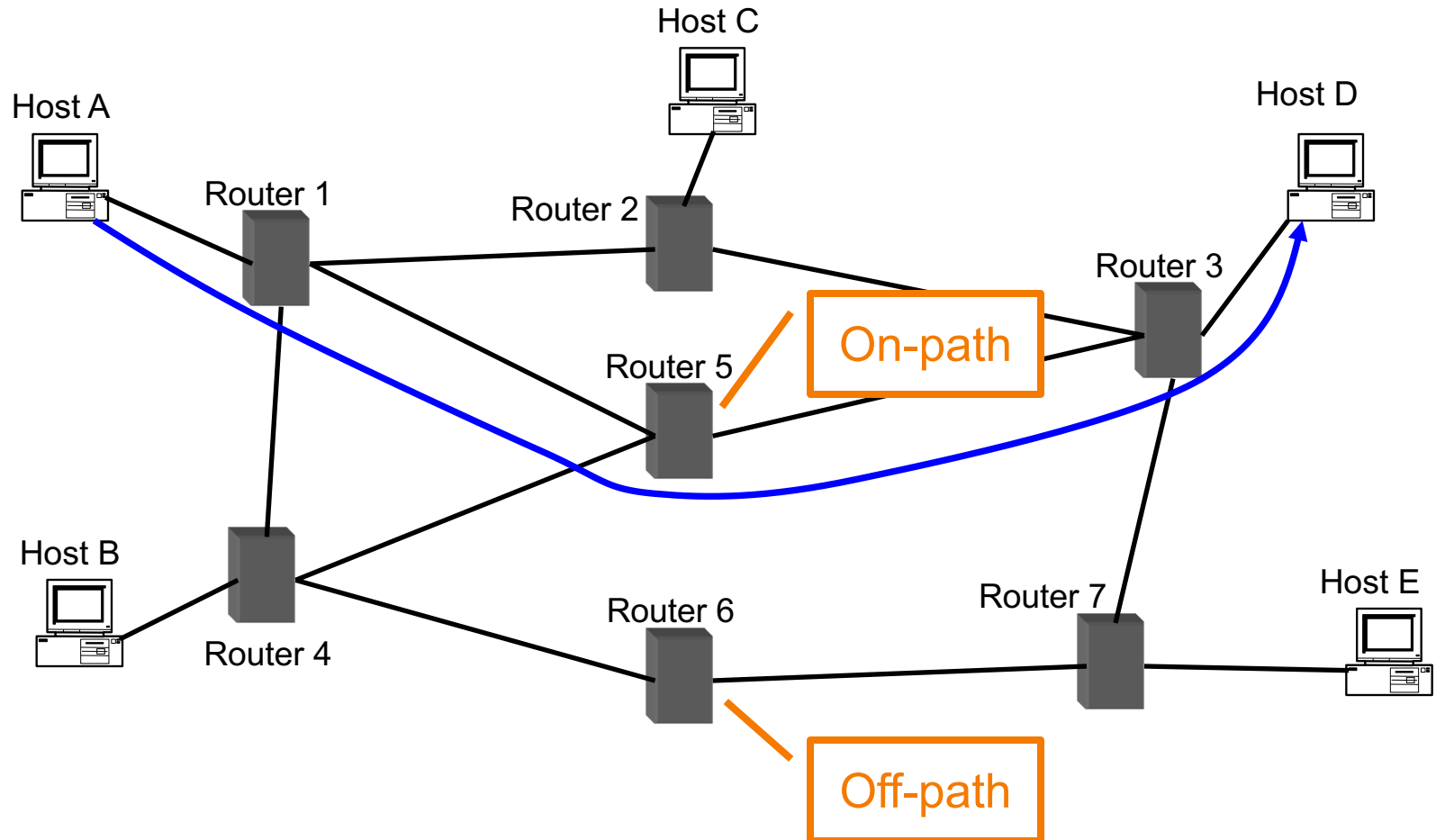
D	C	Hello world!
---	---	--------------

Physical/Link-Layer Threats: *Spoofing*

- With physical access to a local network, attacker can create any message they like
 - When with a bogus source address: *spoofing*
- When using a typical computer, may require root/administrator to have full freedom
- Particularly powerful when combined with *eavesdropping*
 - Because attacker can understand exact state of victim's communication and craft their spoofed traffic to match it
 - Spoofing w/o eavesdropping = *blind spoofing*

On-path vs Off-path Spoofing

Host A communicates with Host D



Spoofing on the Internet

- On-path attackers can see victim's traffic \Rightarrow spoofing is easy
- Off-path attackers can't see victim's traffic
 - They have to resort to blind spoofing
 - Often must **guess/infer** header values to succeed
 - o We then care about work factor: how hard is this
 - But sometimes they can just **brute force**
 - o E.g., 16-bit value: just try all 65,536 possibilities!
- When we say an attacker “can spoof”, we usually mean “w/ reasonable chance of success”

IP-Layer Threats

- Can set arbitrary source address
 - “**Spoofing**” - receiver has no idea who you are
 - Could be **blind**, or could be coupled w/ **sniffing**
 - Note: many attacks require **two-way communication**
 - o So successful off-path/blind spoofing might not suffice
- Can set arbitrary destination address
 - Enables “**scanning**” – brute force searching for hosts
- Can *send like crazy* (**flooding**)
 - IP has no general mechanism for tracking **overuse**
 - IP has no general mechanism for tracking **consent**
 - Very hard to tell where a spoofed flood comes from!
- **If** attacker can **manipulate routing**, can bring traffic to themselves for *eavesdropping* (not easy)