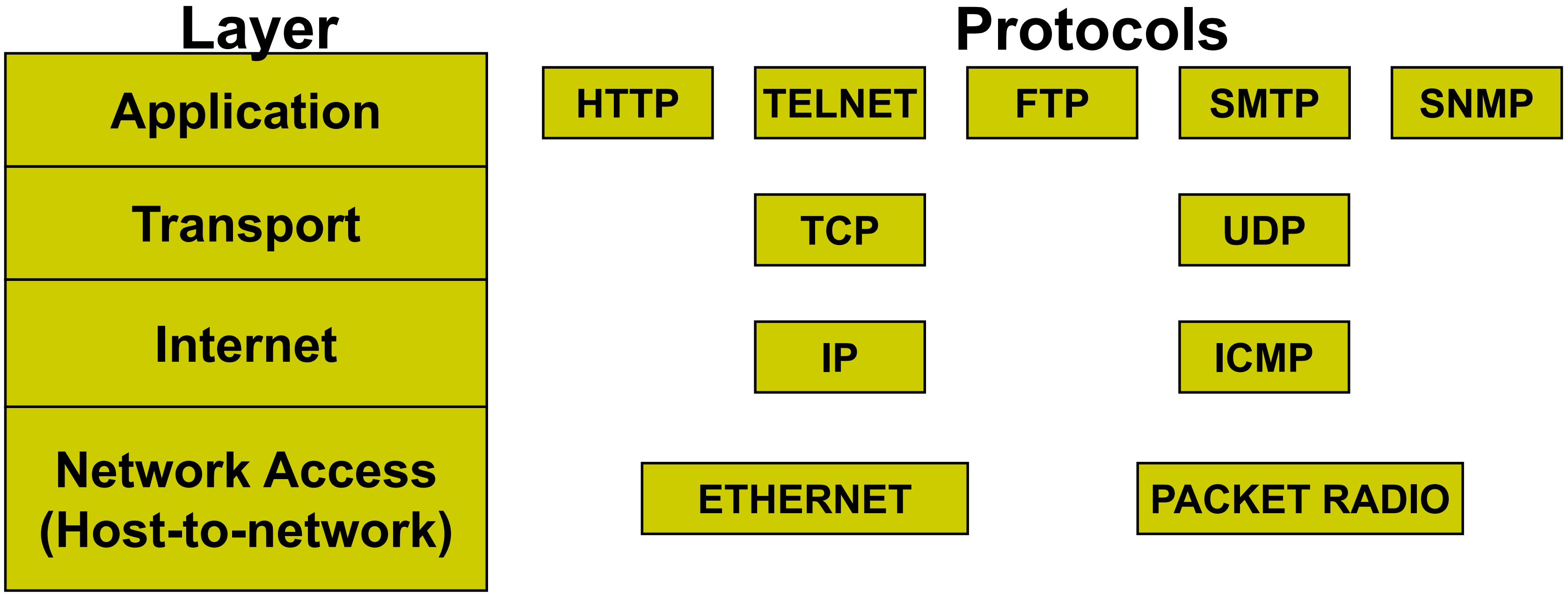
CS 161: Computer Security

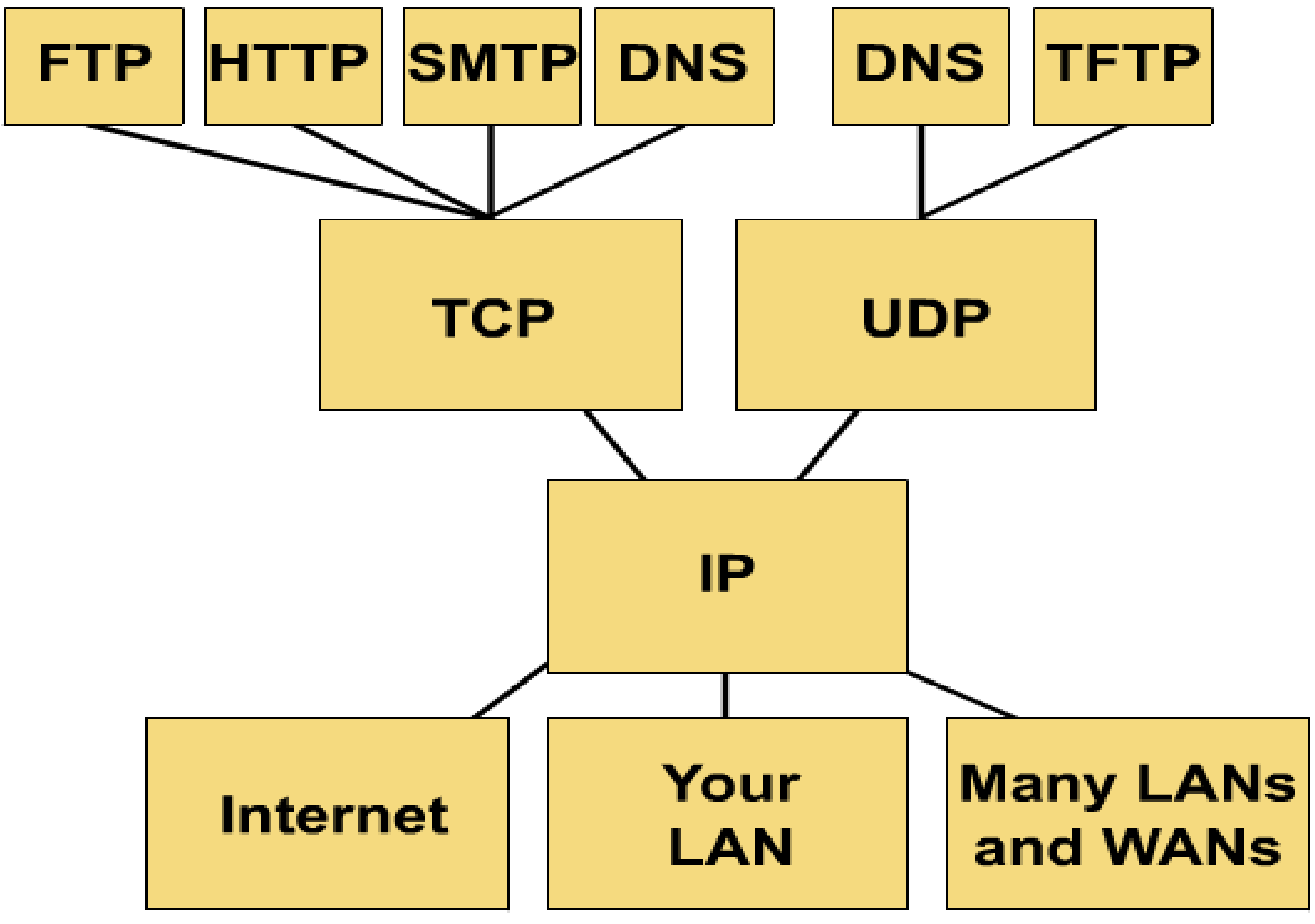
Lecture 11

October 13, 2015

TCP/IP Reference Model



TCP/IP protocol stack



TCP/IP versus OSI

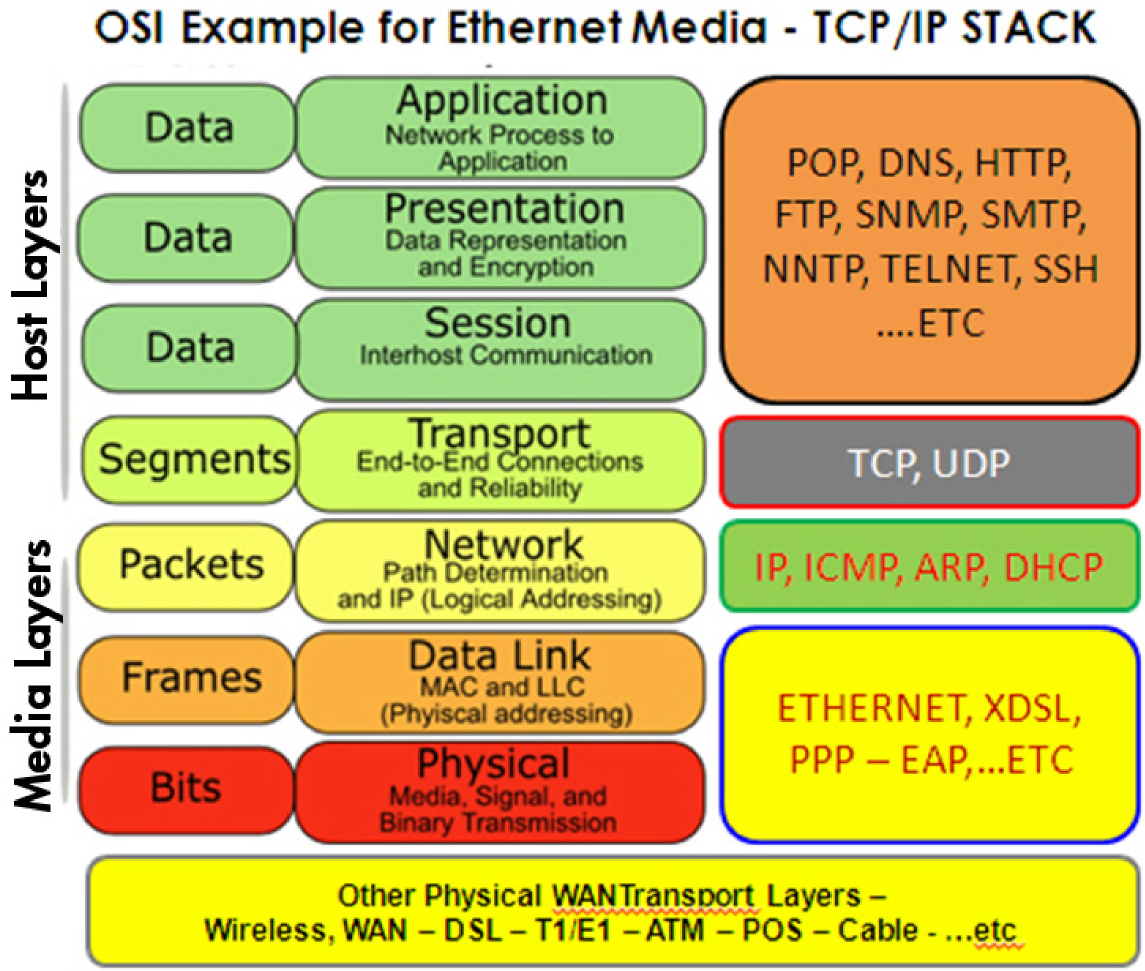
OSI Model	TCP/IP Hierarchy	Protocols				
7 Application Layer	Application Layer	HTTP	SMTP	POP3	FTP	
6 Presentation Layer						
5 Session Layer						
4 Transport Layer	Transport Layer	TCP		UDP		
3 Network Layer	Network Layer	IP				ICMP
2 Link Layer	Link Layer	ARP RARP Ethernet		PPP		
1 Physical Layer						

Link Layer : includes device driver and network interface card
Network Layer : handles the movement of packets, i.e. routing
Transport Layer : provides a reliable flow of data between two hosts
Application Layer : handles the details of the particular application

Mnemonics

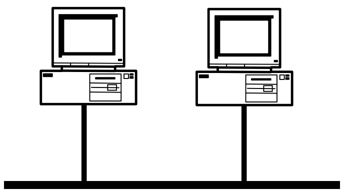
- All People Seem To Need Data Processing
    - Application, Presentation, Session, Transport, Network, Data link, Physical
- Please Do Not Throw Sausage Pizza Away
    - Physical, Data link, Network, Transport, Session, Presentation, Application
- All People Should Try New Dr. Pepper
    - Application, Presentation, Session, Transport, Network, Data link, Physical

TCP/IP versus OSI

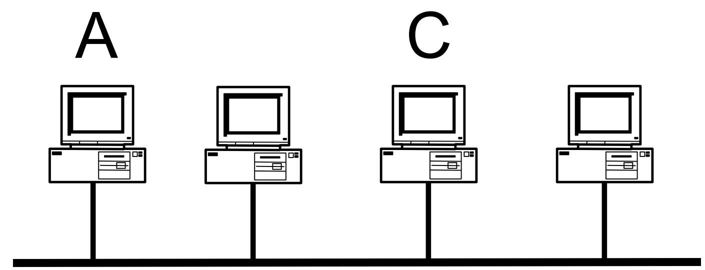


Link Layer : includes device driver and network interface card
Network Layer : handles the movement of packets, i.e. routing
Transport Layer : provides a reliable flow of data between two hosts
Application Layer : handles the details of the particular application

Local-Area Networks



point-to-point



shared

How does computer A send a message to computer C?

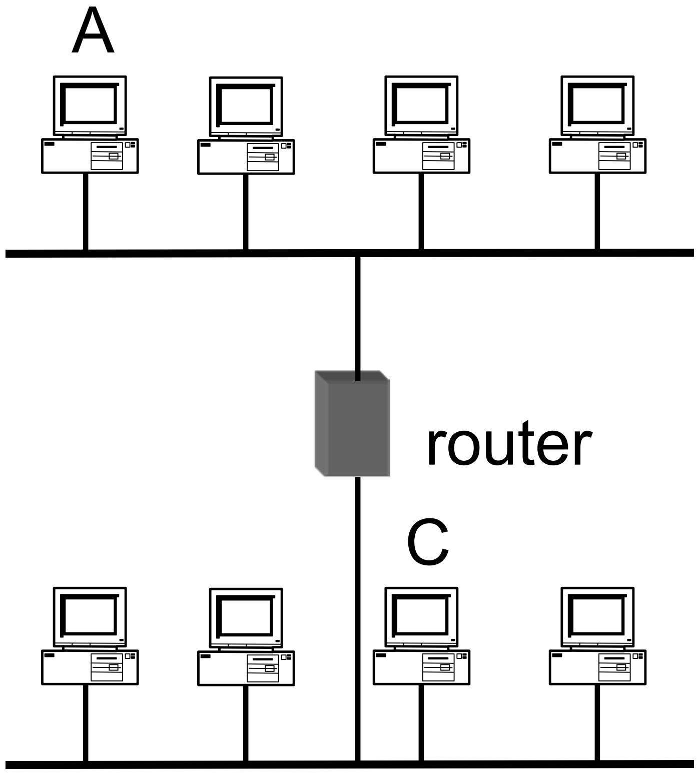
Local-Area Networks: Packets

```
From: A
To: C
Message: Hello world!
```

A	C	Hello world!

<table>
<tr><td>A</td><td>C</td></tr>
<tr><td colspan="2">Hello world!</td></tr>
</table>

Wide-Area Networks



How do we connect two LANs?

Key Concept #1: *Protocols*

- A protocol is an agreement on how to communicate
- Includes syntax and semantics
  - How a communication is specified & structured
    - Format, order messages are sent and received
  - What a communication means
    - 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
  - * Today's Internet is full of hacks that violate this
- **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
  - If you want smarts, build it "*end-to-end*", not "hop-by-hop"
  - Buys simplicity & robustness at the cost of shifting complexity into end systems

Self-Contained IP Packet Format

IP = Internet *Protocol*

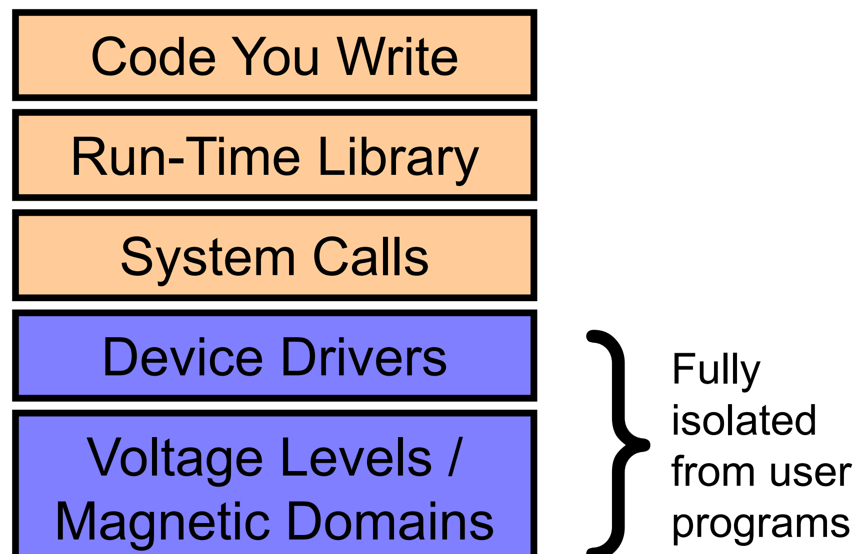
4-bit Version	4-bit Header Length	8-bit Type of Service (TOS)	16-bit Total Length (Bytes)	
16-bit Identification			3-bit Flags	13-bit Fragment Offset
8-bit Time to Live (TTL)	8-bit Protocol		16-bit Header Checksum	
32-bit Source IP Address				
32-bit Destination IP Address				

Payload (remainder of message)
.
.
.
.
.

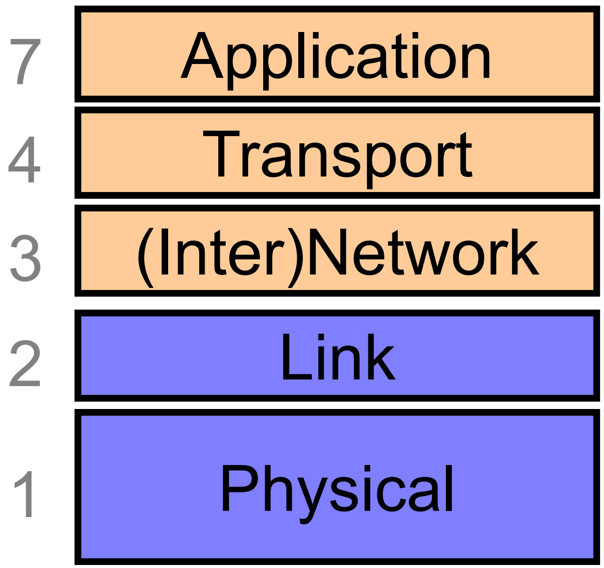
Header **is like a letter envelope: contains all info needed for delivery**

Key Concept #3: *Layering*

- 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



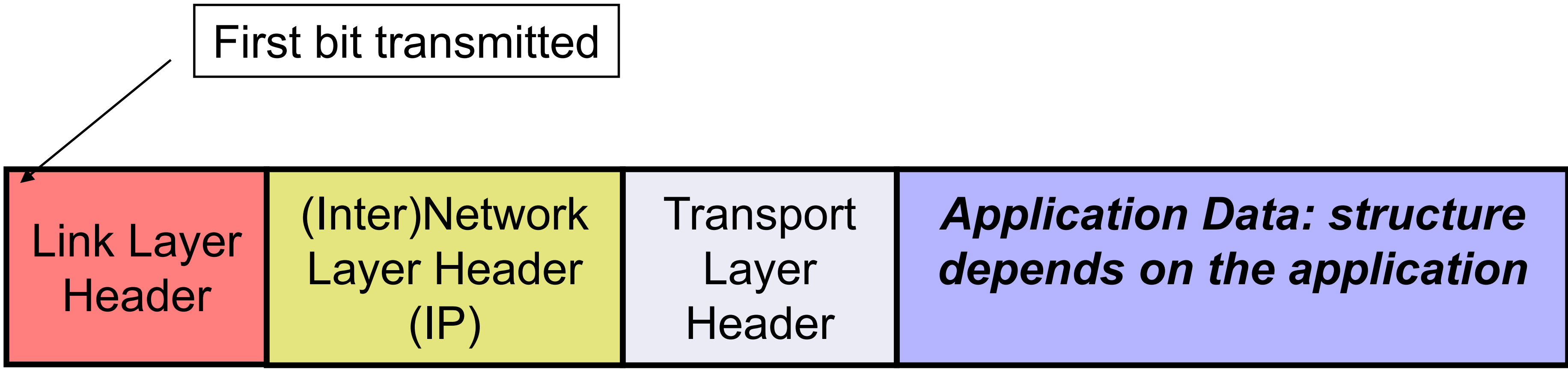
Internet Layering (“Protocol Stack”)



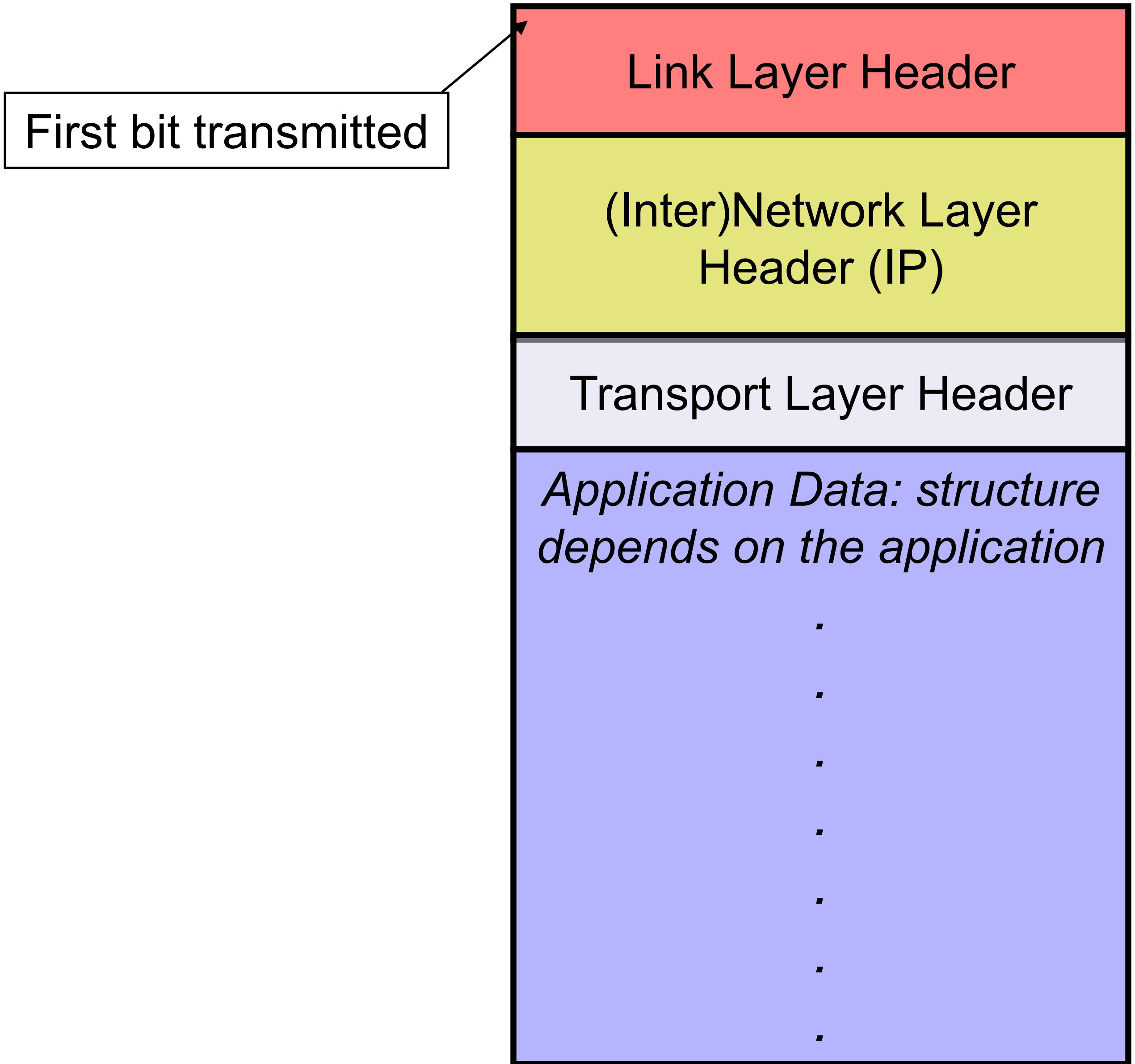
Note on a point of potential confusion: these diagrams are always drawn with lower layers **below** higher layers

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

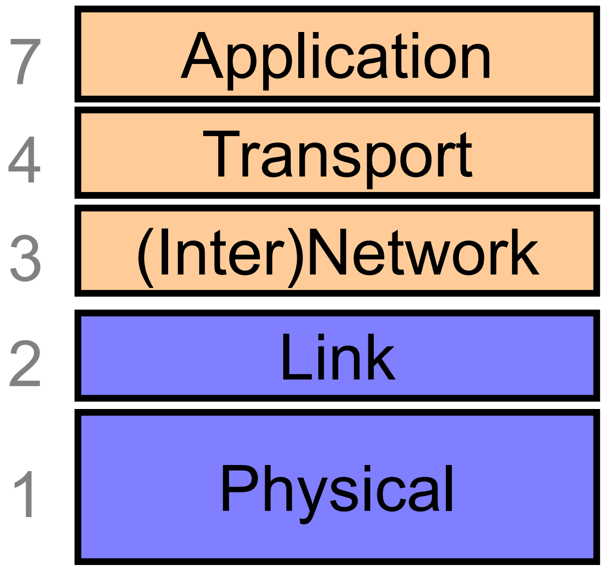
Horizontal View of a Single Packet



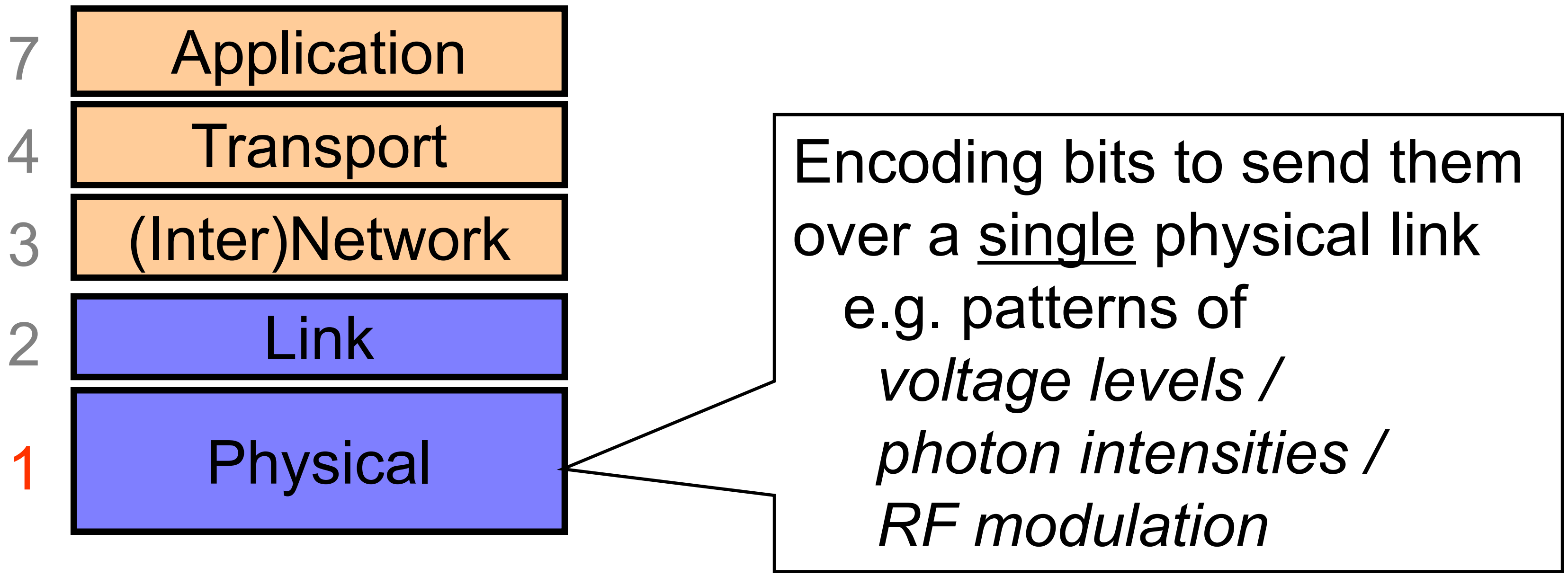
Vertical View of a Single Packet



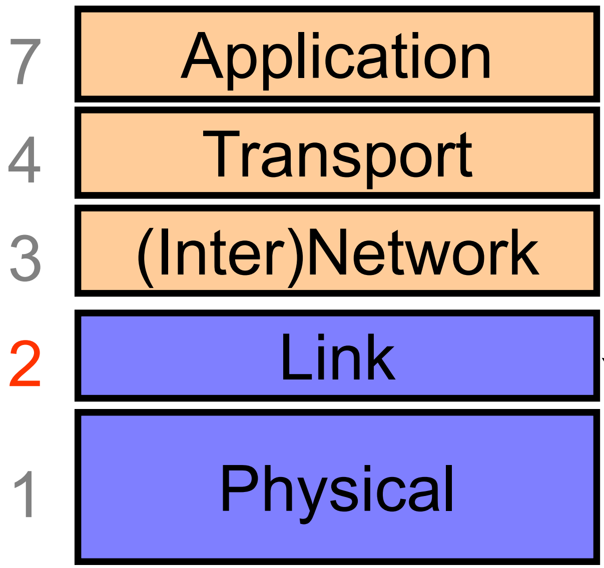
Internet Layering ("Protocol Stack")



Layer 1: Physical Layer



Layer 2: Link Layer

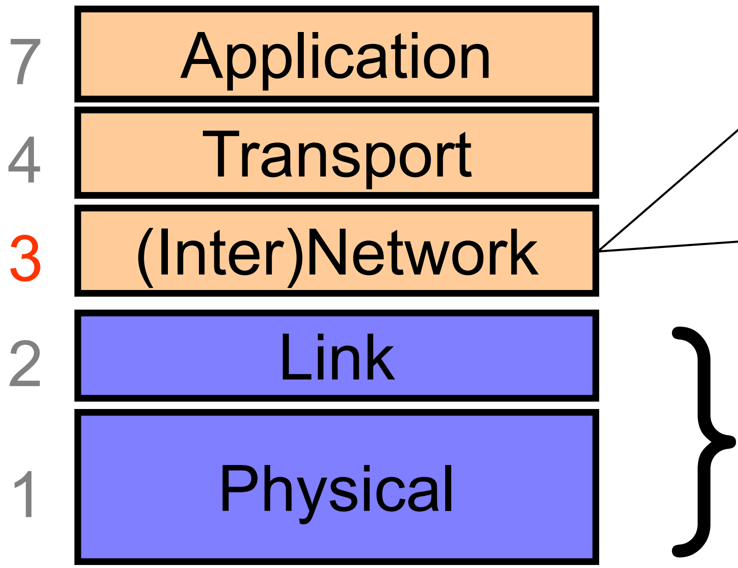


Framing and transmission of a collection of bits into individual messages sent across a single "subnetwork" (one physical technology)

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)*



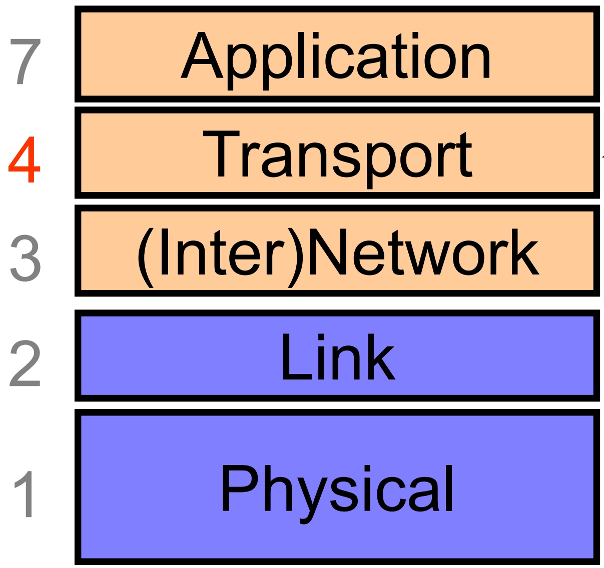
Bridges multiple "subnets" to provide end-to-end internet connectivity between nodes

- Provides global addressing

Works across different link technologies

***Different* for each Internet "hop"**

Layer 4: Transport Layer

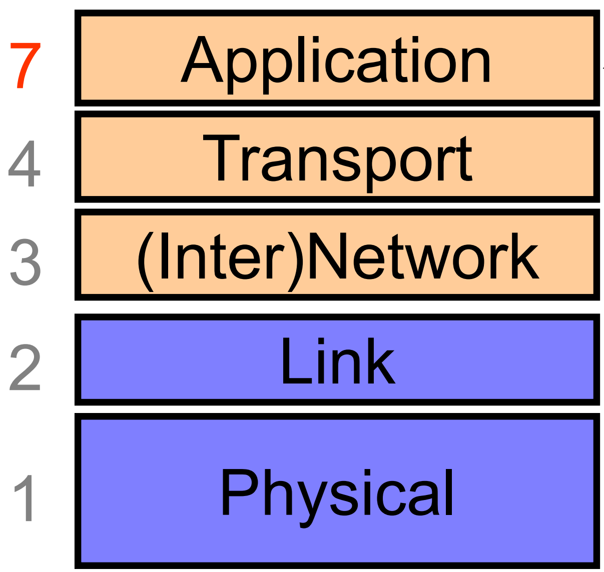


End-to-end communication between processes

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

(Datagram = single packet message)

Layer 7: Application Layer



Communication of whatever you wish

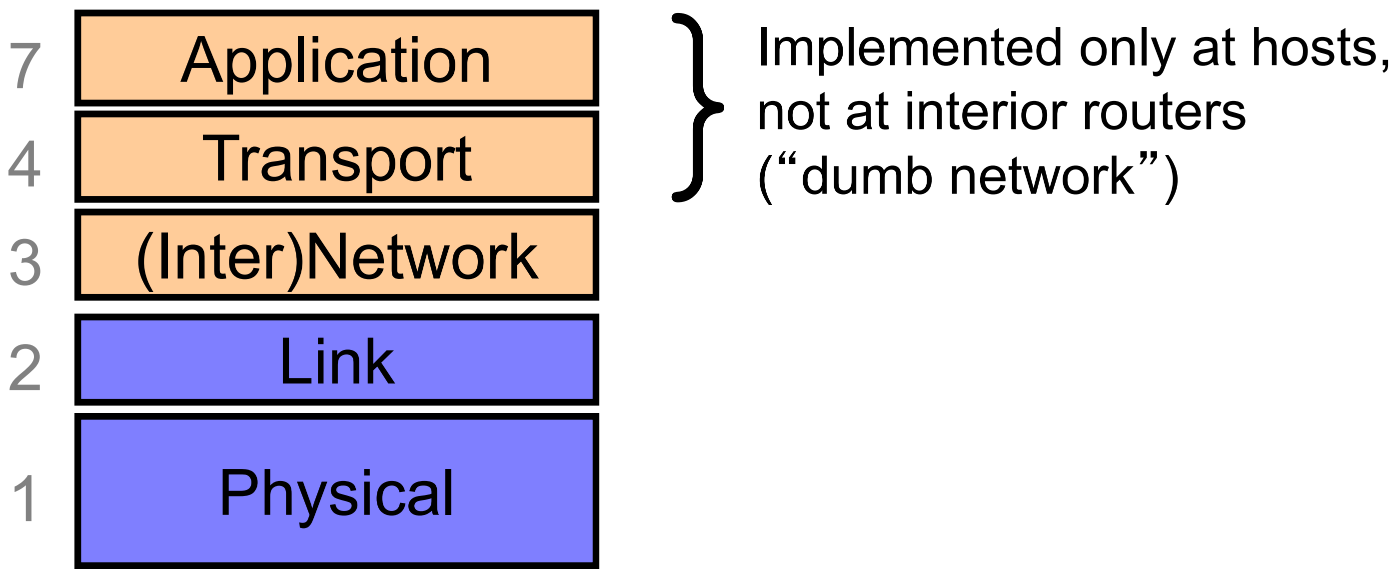
Can use whatever transport(s) is convenient

Freely structured

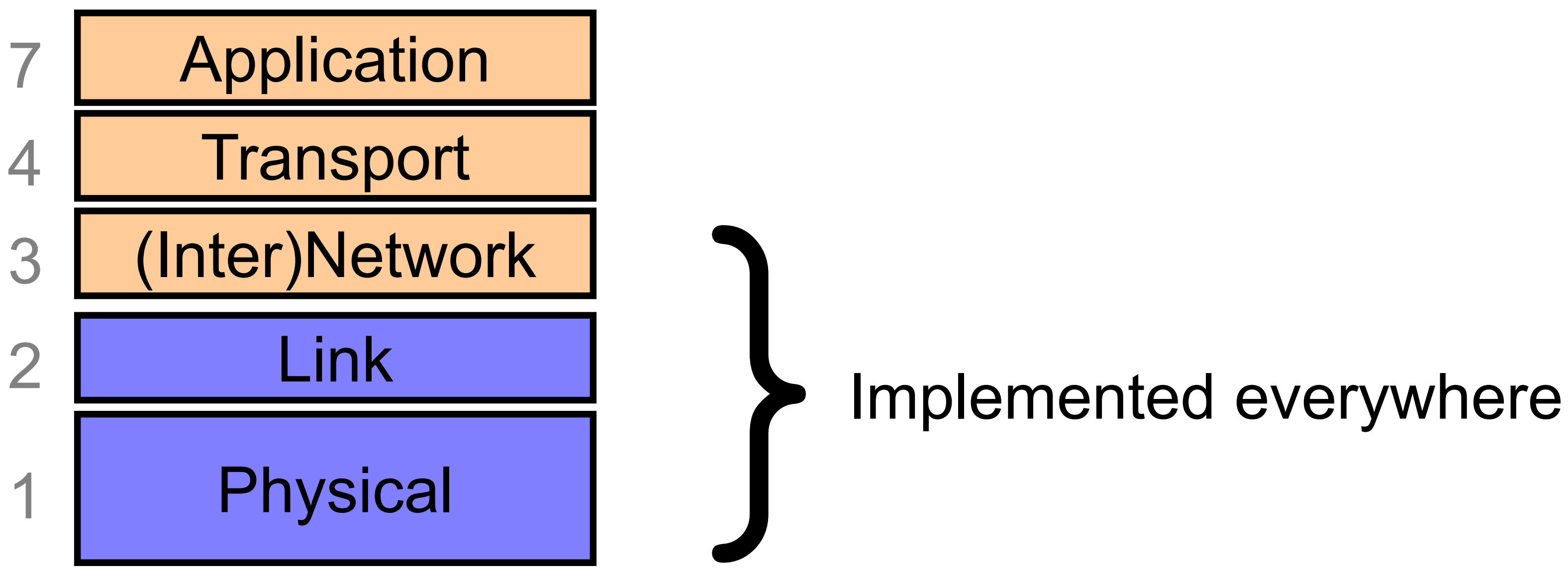
E.g.:

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

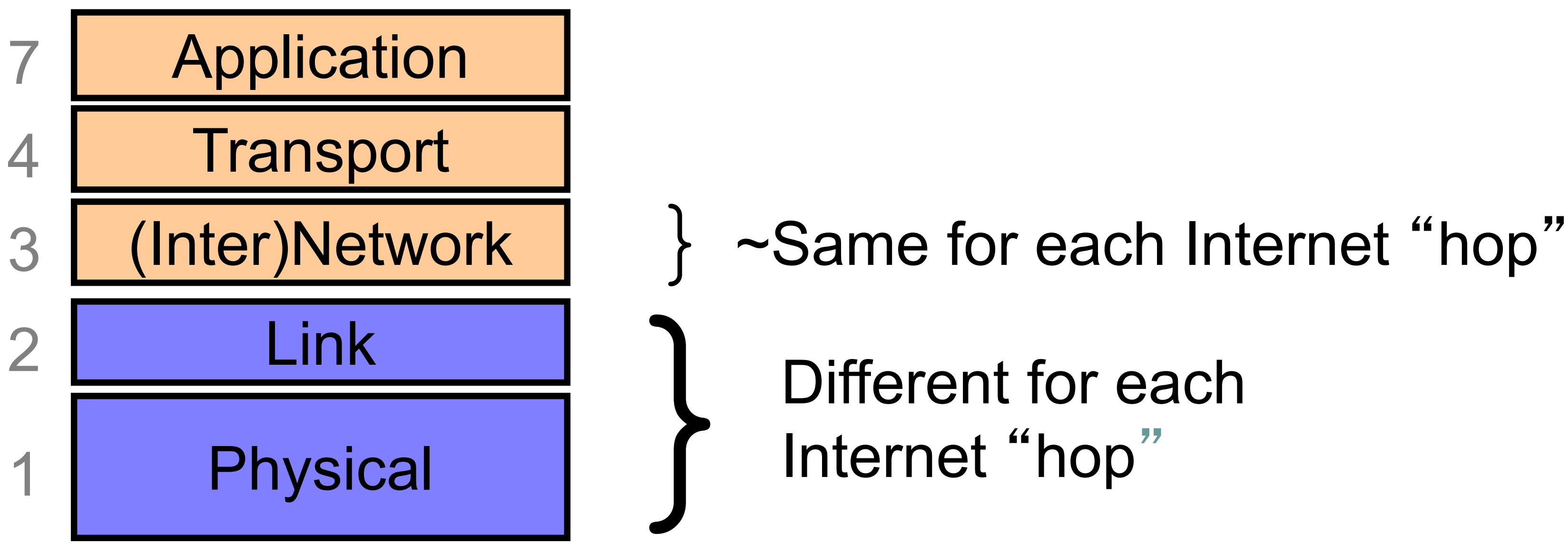
Internet Layering ("Protocol Stack")



Internet Layering ("Protocol Stack")

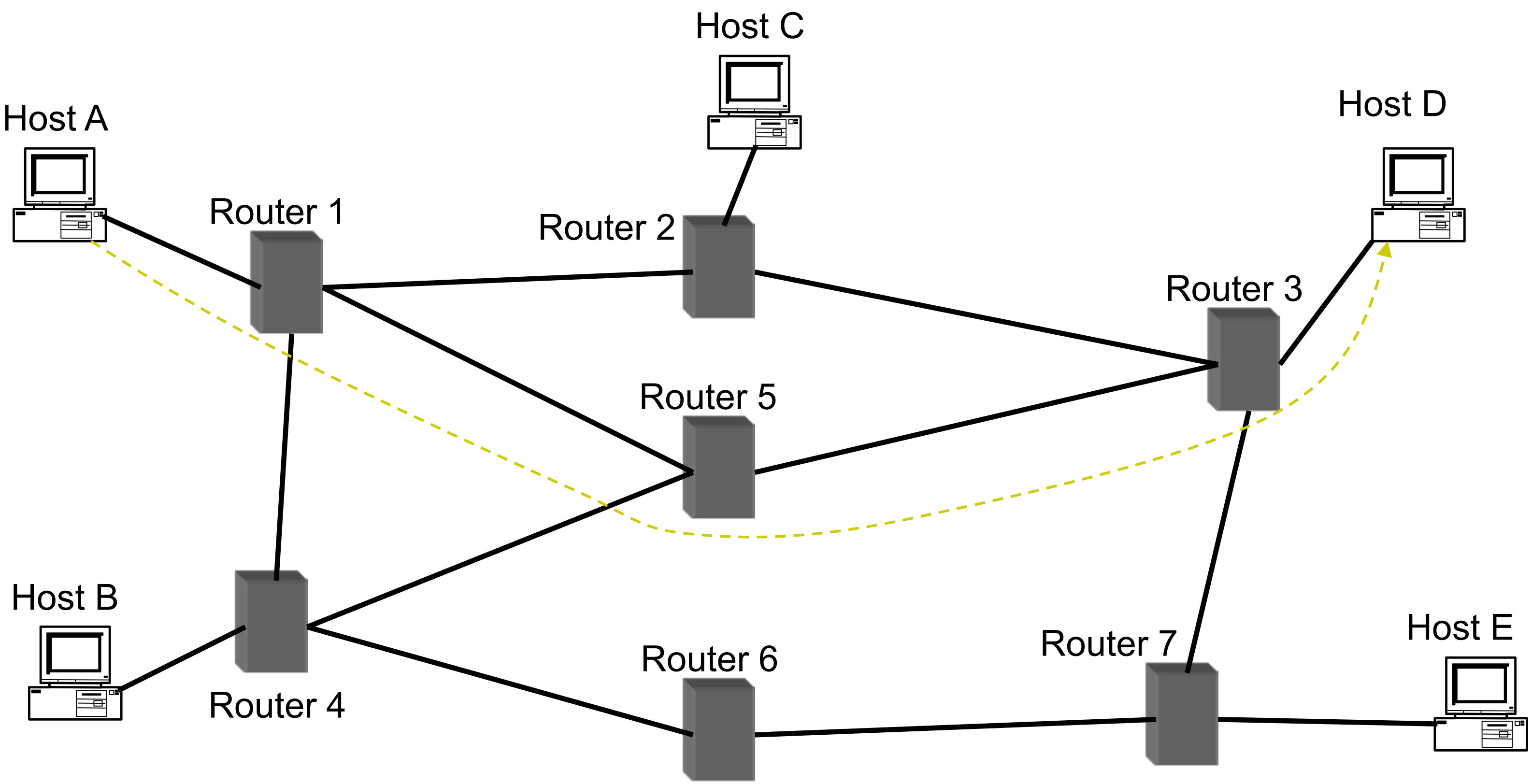


Internet Layering ("Protocol Stack")



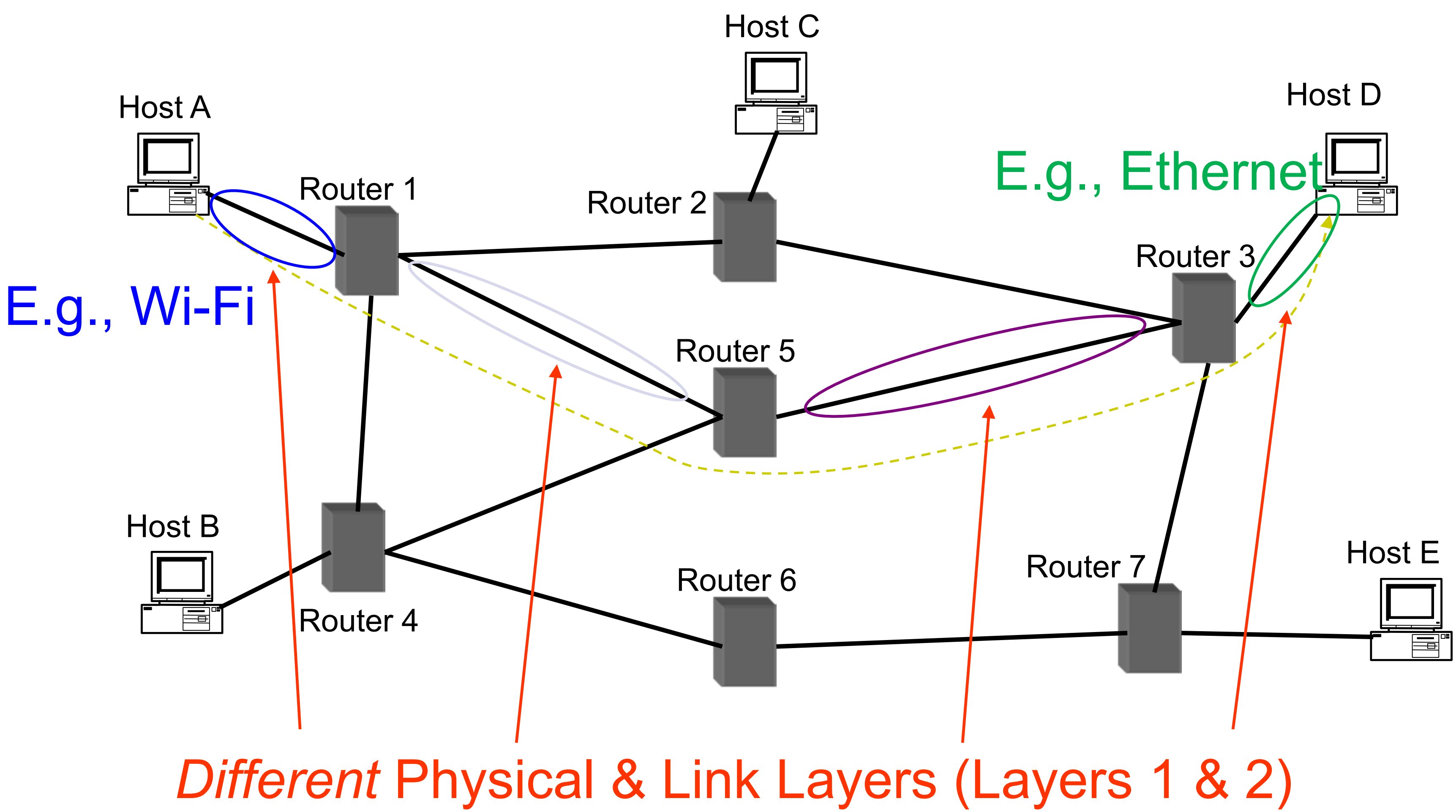
Hop-By-Hop vs. End-to-End Layers

Host A communicates with Host D



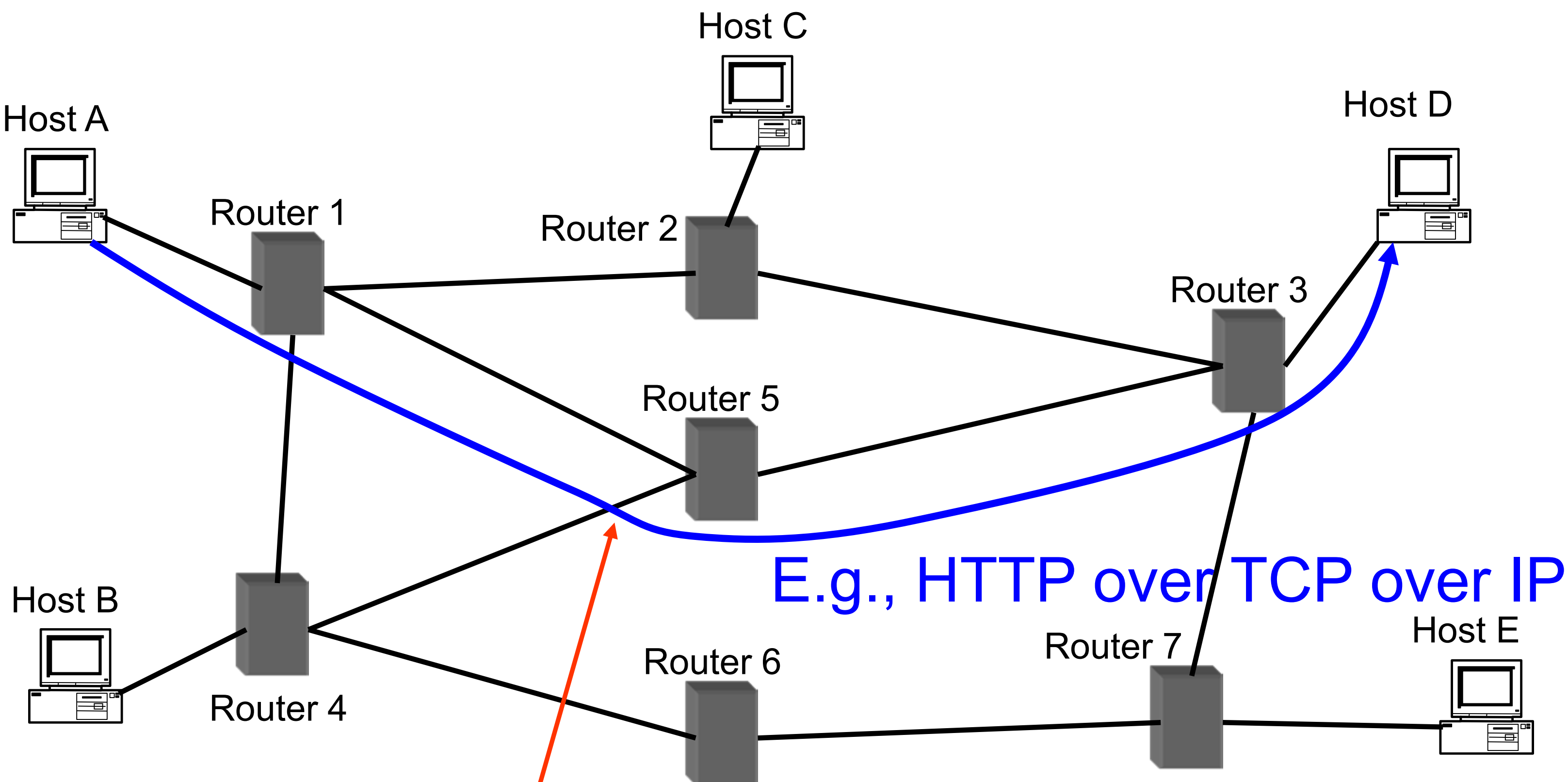
Hop-By-Hop vs. End-to-End Layers

Host A communicates with Host D



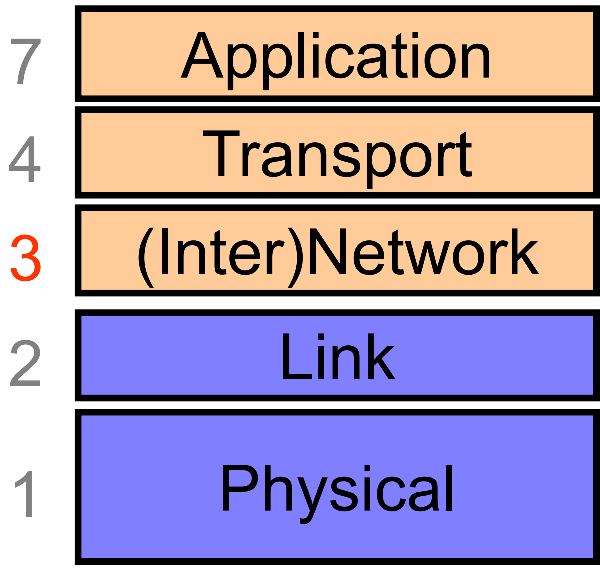
Hop-By-Hop vs. End-to-End Layers

Host A communicates with Host D



Same Network / Transport / Application Layers (3/4/7)
(Routers ignore Transport & Application layers)

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

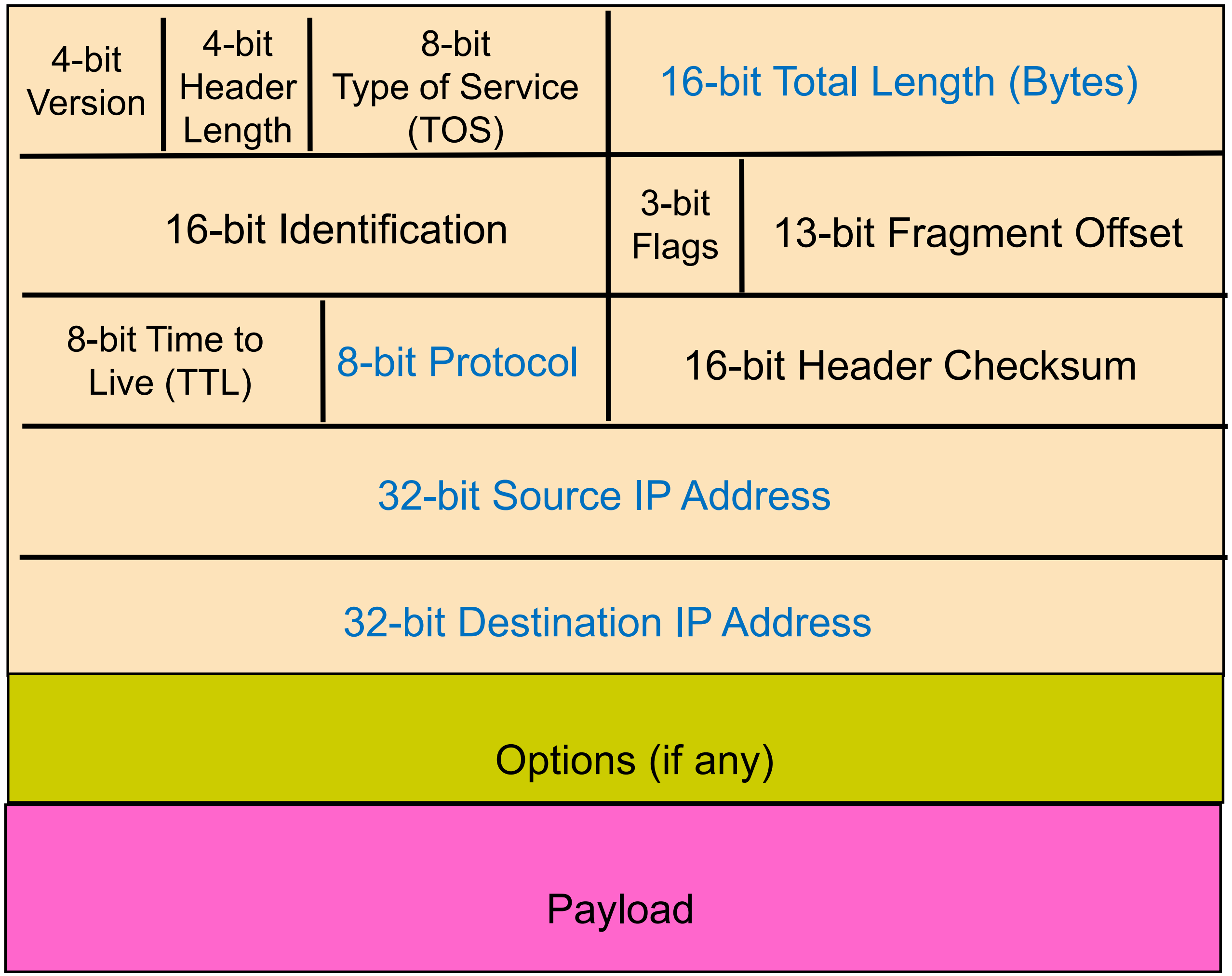


Bridges multiple "subnets" to provide end-to-end internet connectivity between nodes

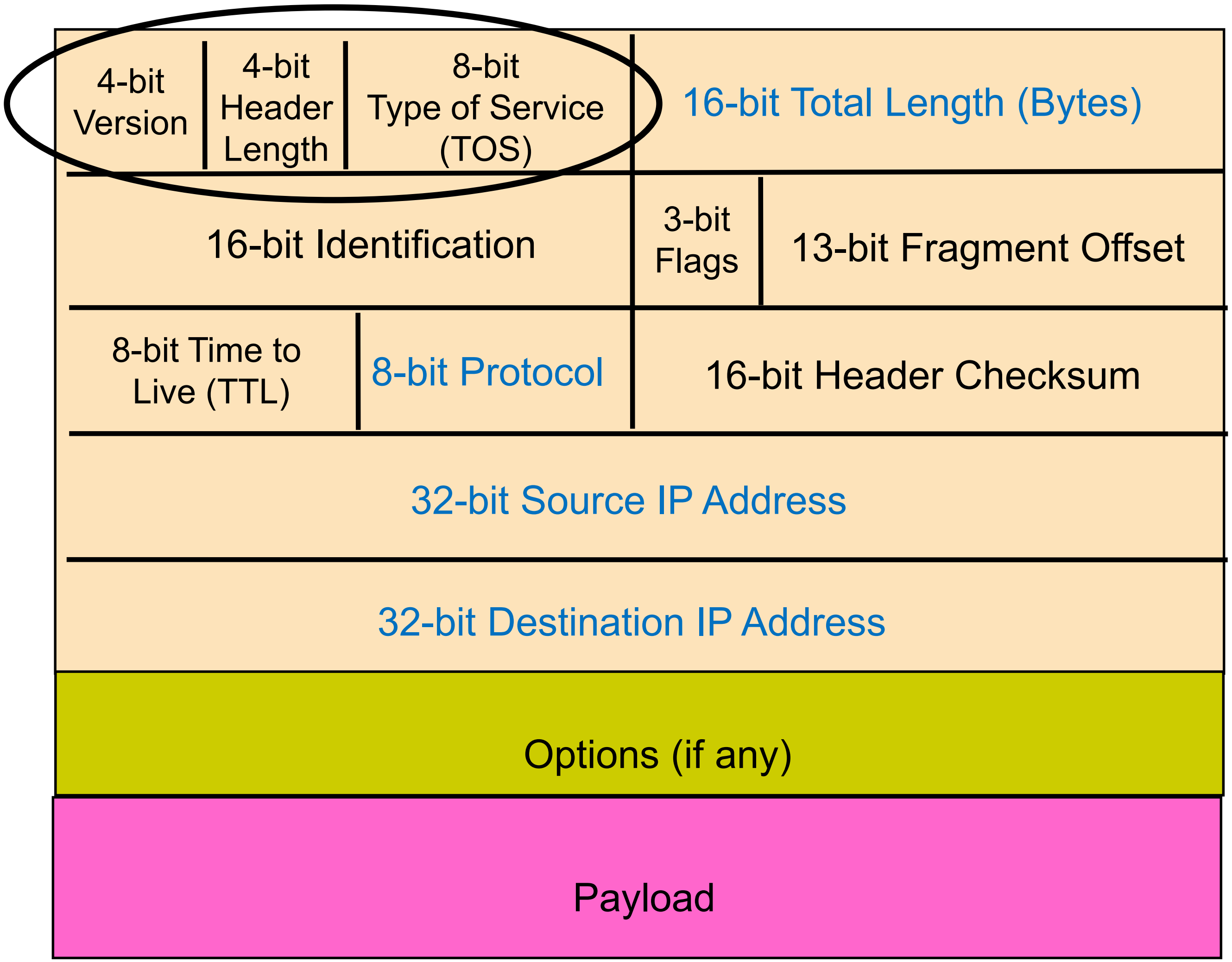
- Provides global addressing

Works across different link technologies

IP Packet Structure



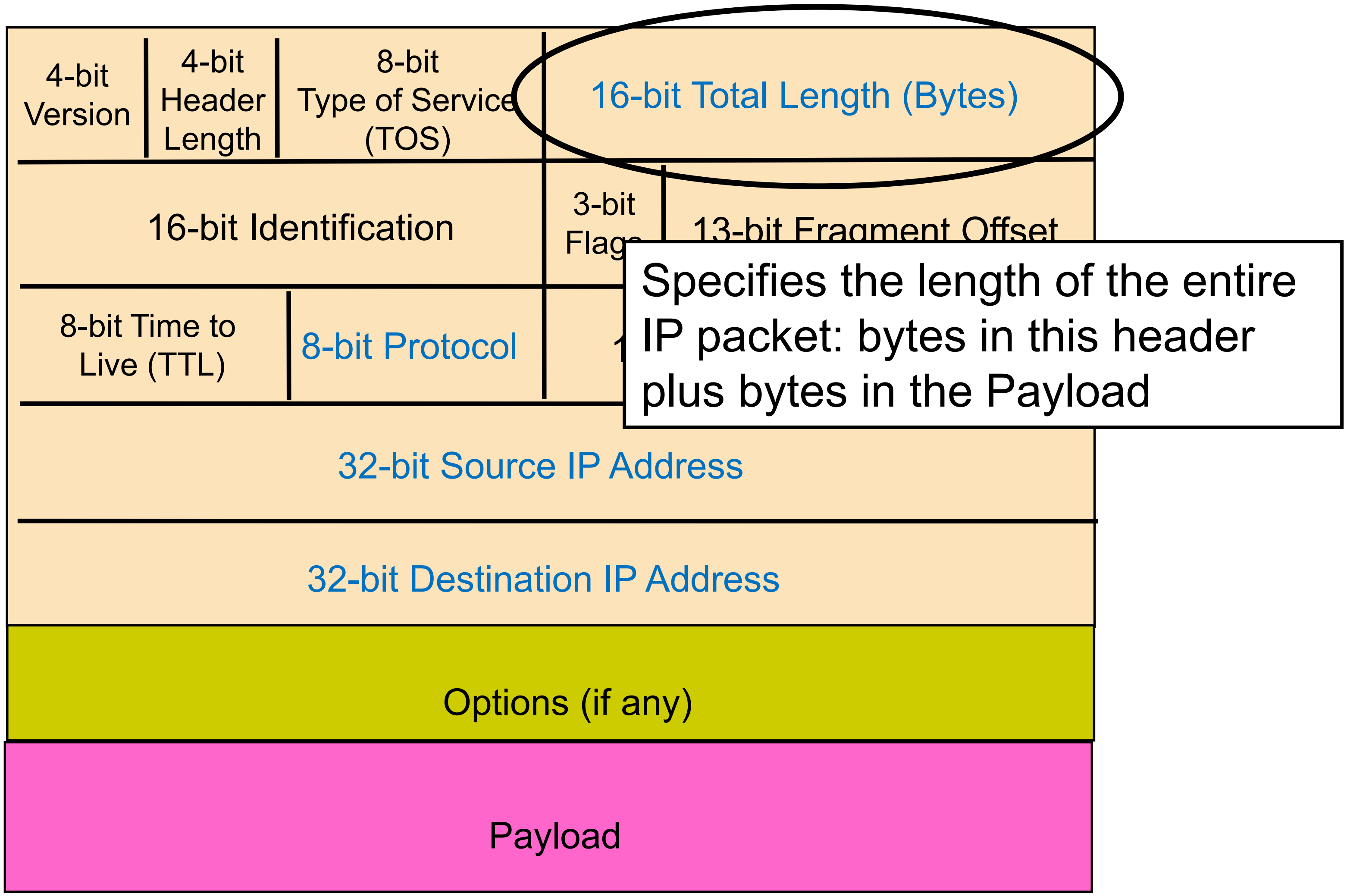
IP Packet Structure



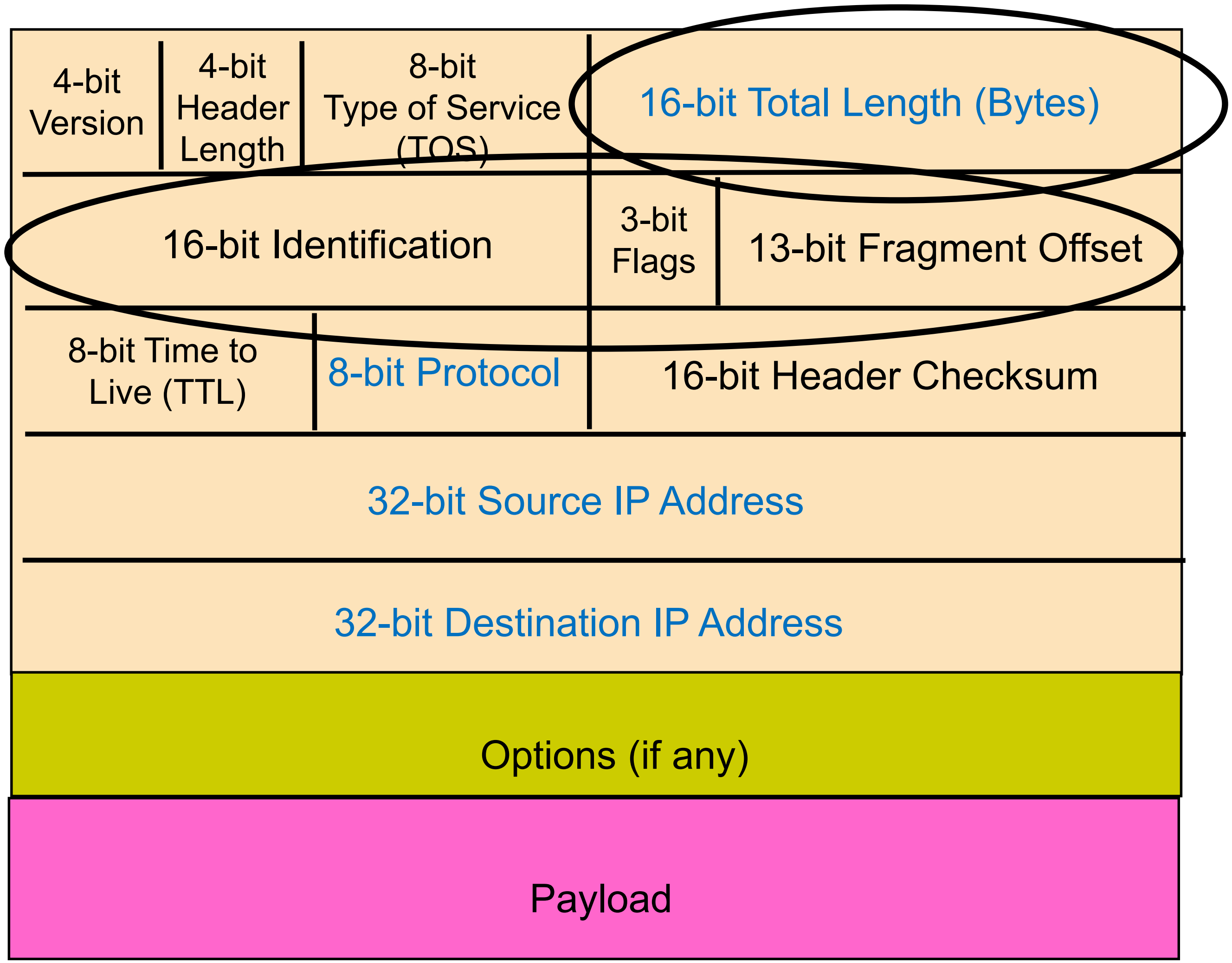
IP Packet Header

- Version number (4 bits)
  - Indicates the version of the IP protocol
  - Necessary to know what other fields to expect
  - Typically "4" (for IPv4), and sometimes "6" (for IPv6)
- Header length (4 bits)
  - Number of 32-bit words in the header
  - Typically "5" (for a 20-byte IPv4 header)
  - Can be more when IP options are used
- Type-of-Service (8 bits)
  - Allow packets to be treated differently based on needs
  - E.g., low delay for audio, high bandwidth for bulk transfer

IP Packet Structure



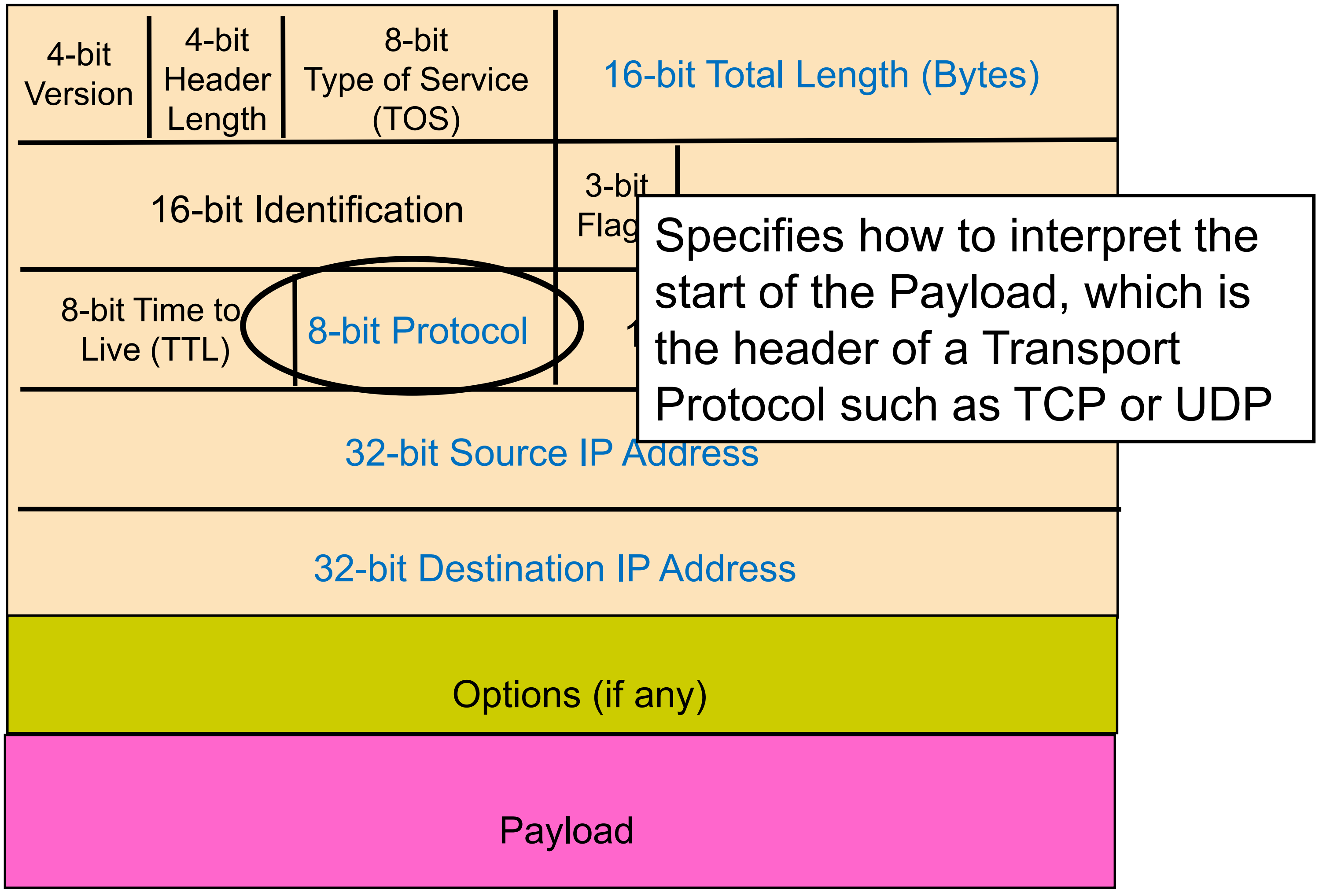
IP Packet Structure



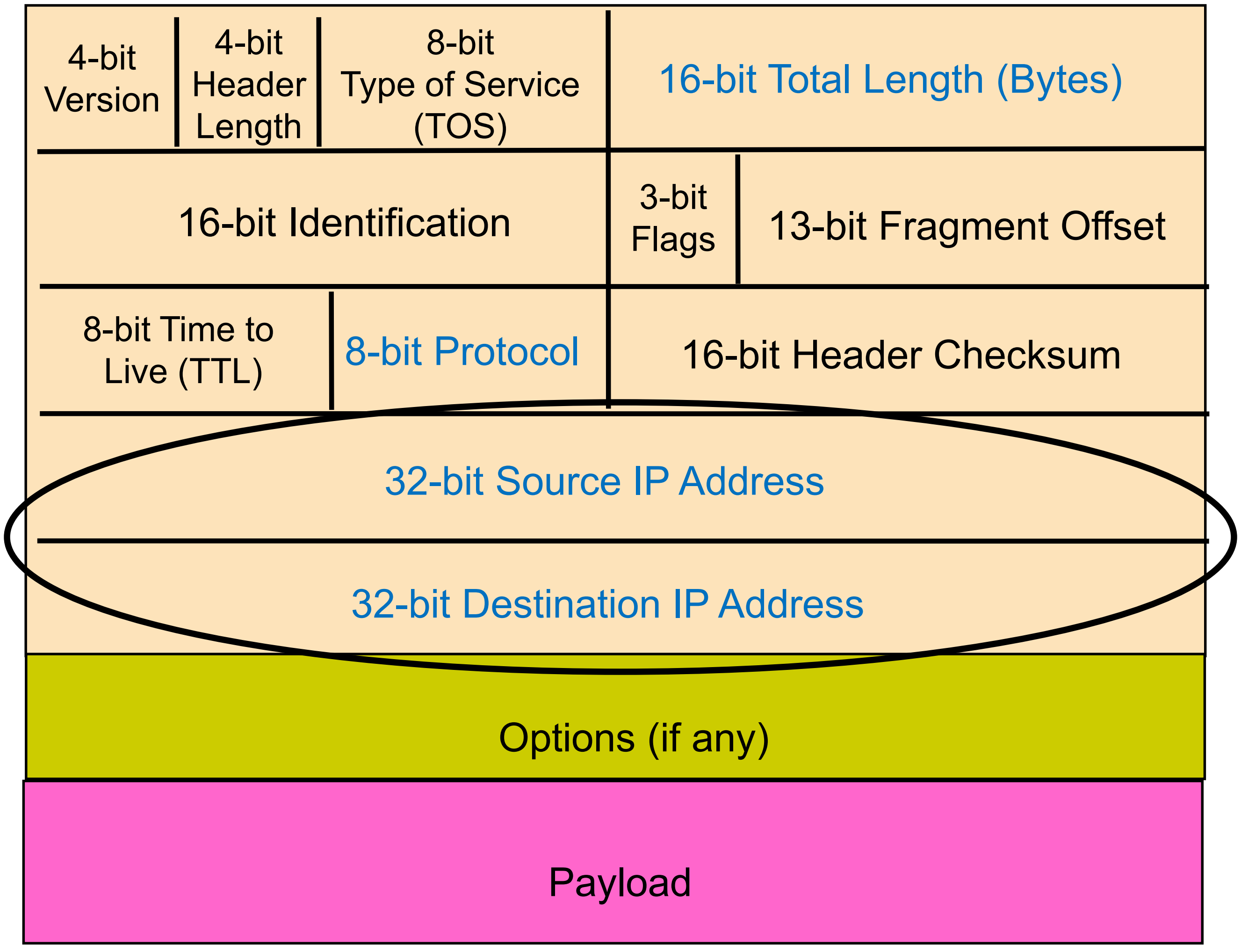
IP Packet Header (Continued)

- Total length (16 bits)
  - Number of bytes in the packet
  - Maximum size is 65,535 bytes (2^{16} -1)
  - though underlying links may impose smaller limits
- Fragmentation: when forwarding a packet, an Internet router can split it into multiple pieces ("fragments") if too big for next hop link
- End host reassembles to recover original packet
- Fragmentation information (32 bits)
  - Packet identifier, flags, and fragment offset
  - Supports dividing a large IP packet into fragments
  - in case a link cannot handle a large IP packet

IP Packet Structure



IP Packet Structure



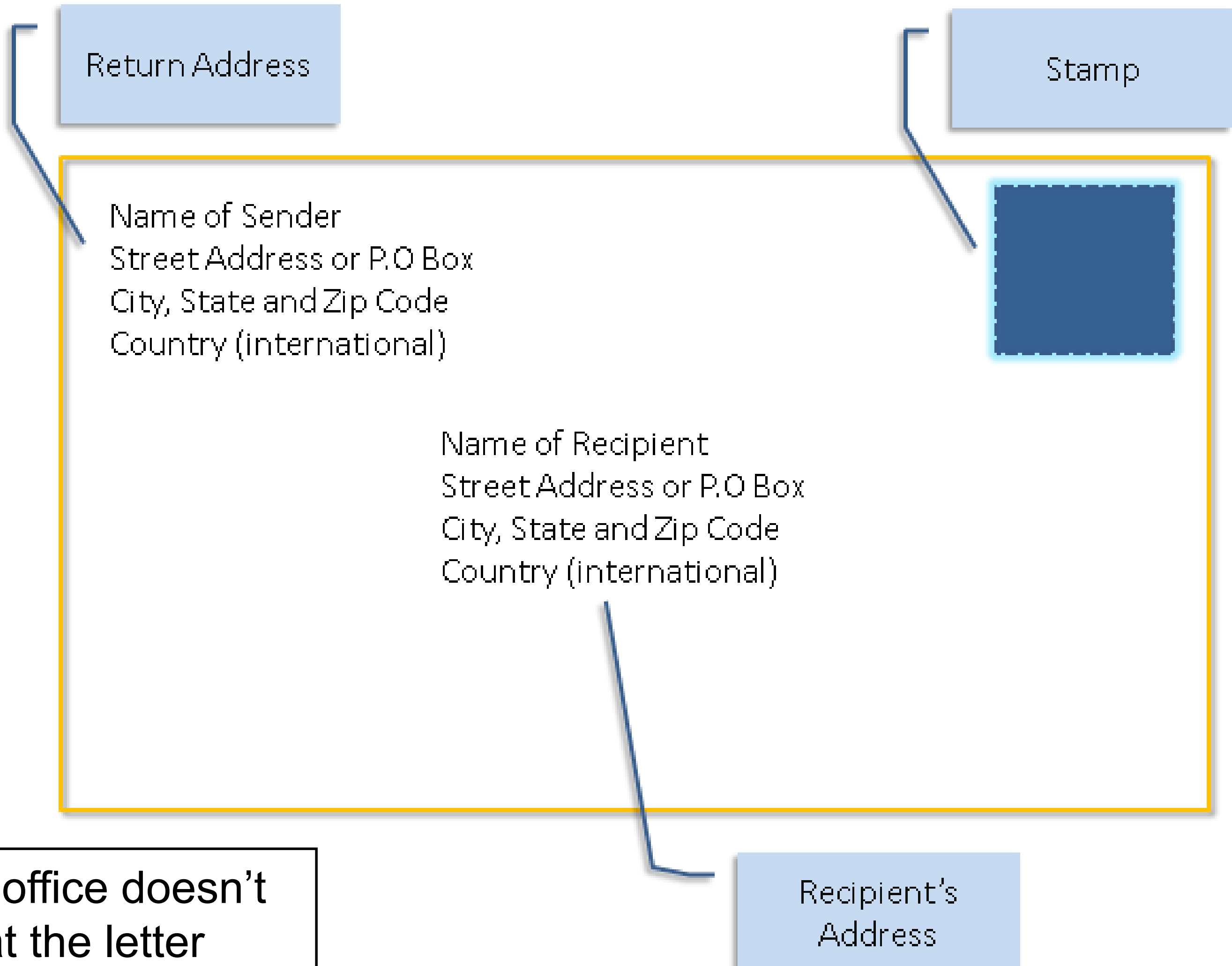
IP Packet Header (Continued)

- 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

Host Names vs. IP addresses

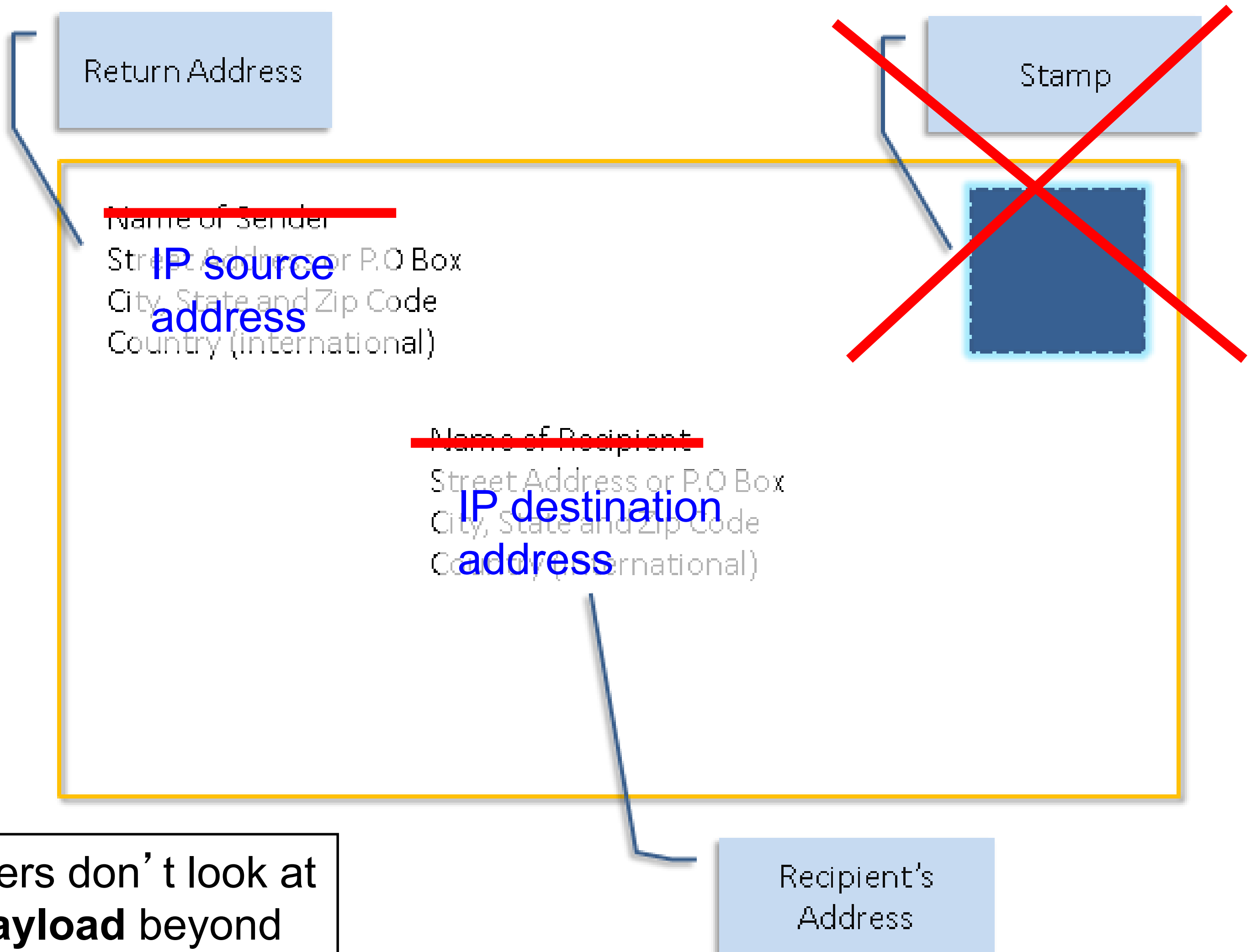
- Host names
  - Examples: `www.cnn.com` and `bbc.co.uk`
  - Mnemonic name appreciated by humans
  - Variable length, full alphabet of characters
  - Provide little (if any) information about location
- IP addresses
  - Examples: `64.236.16.20` and `212.58.224.131`
  - Numerical address appreciated by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

Postal Envelopes:



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

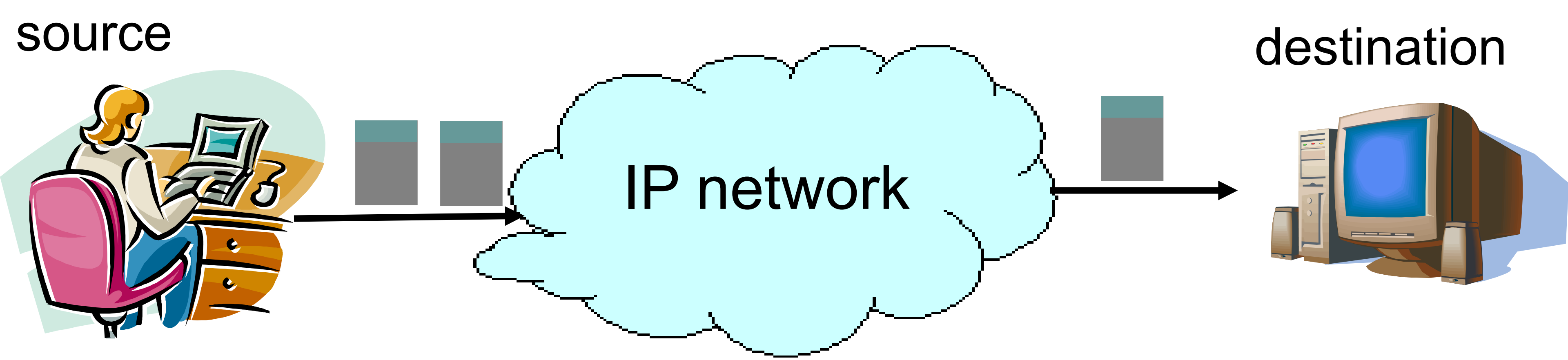
Analogy of IP to Postal Envelopes:



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

IP: Best Effort Packet Delivery

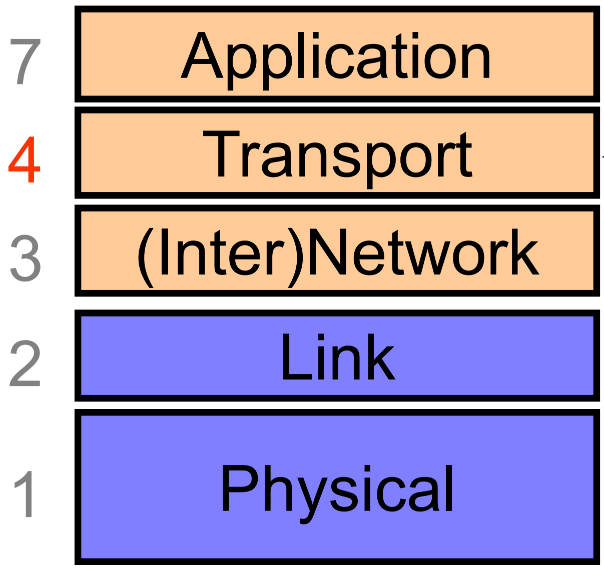
- Routers inspect destination address, locate "next hop" in forwarding table
  - Address = ~unique identifier/locator for the receiving host
- Only provides a "*I'll give it a try*" delivery service:
  - Packets may be lost
  - Packets may be corrupted
  - Packets may be delivered out of order



Going Beyond Best Effort

- Transport (layer 4) protocols build services our apps need
  - Layered on IP’s modest layer-3 service

Layer 4: Transport Layer



End-to-end communication between processes

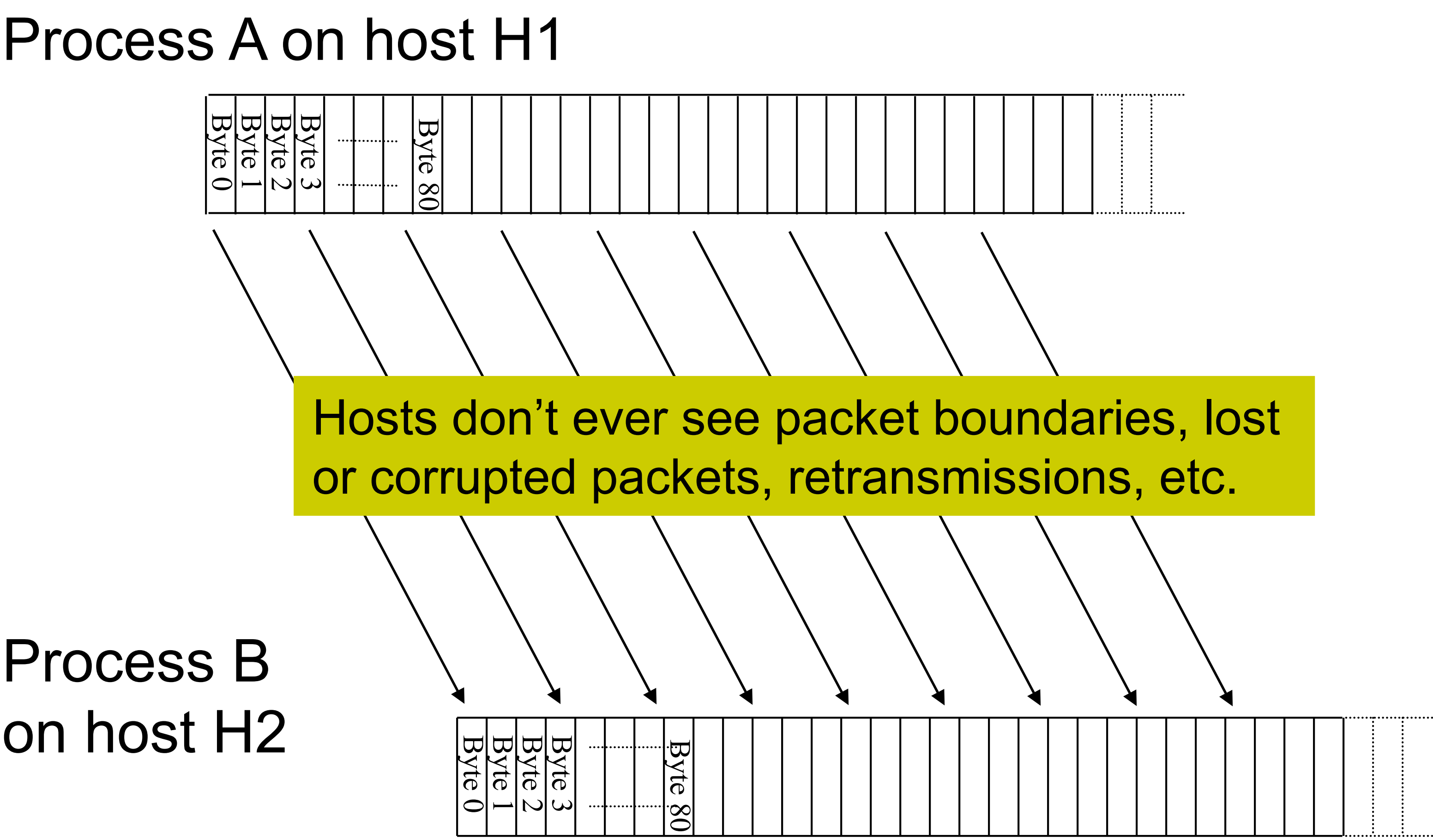
Different services provided:
TCP = reliable byte stream
UDP = unreliable datagrams

(Datagram = single packet message)

Going Beyond Best Effort

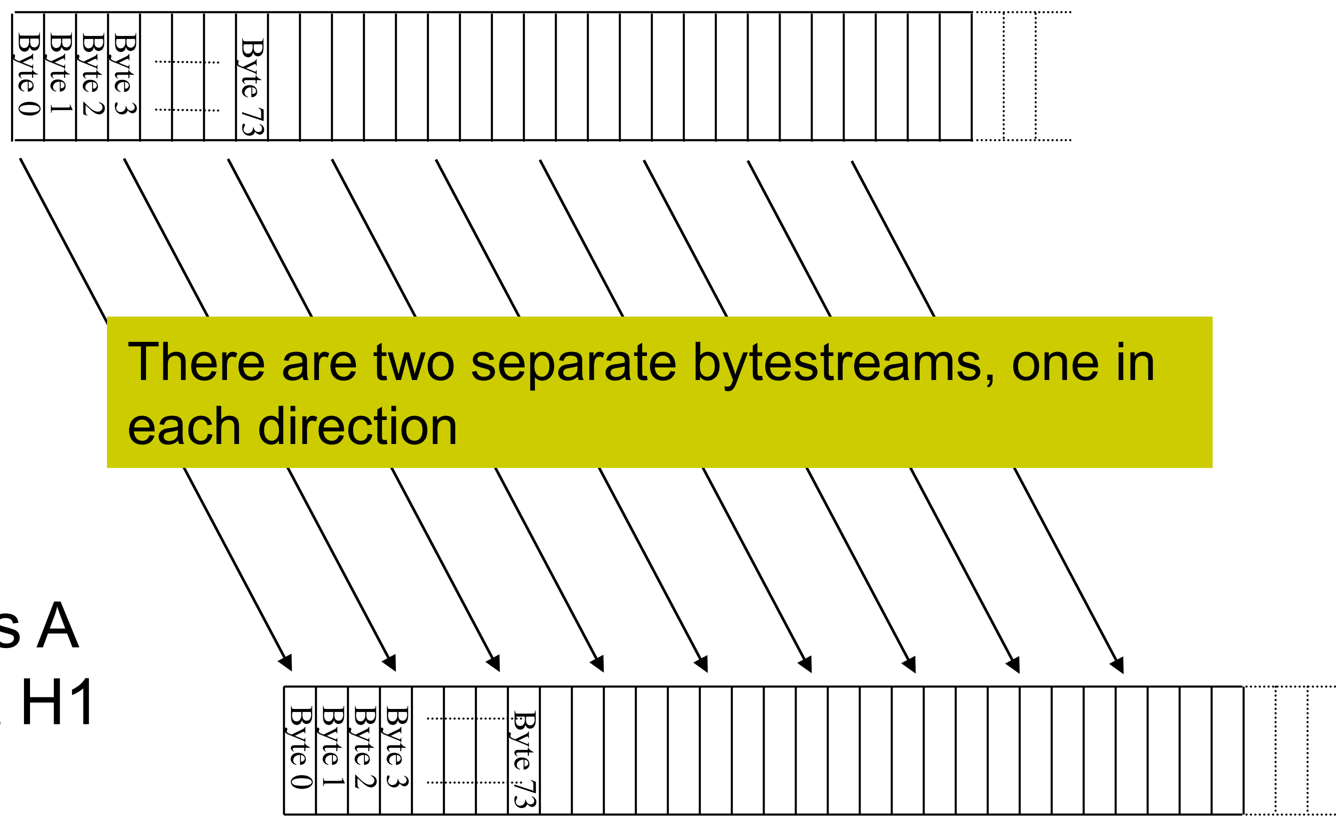
- Transport (layer 4) protocols build services our apps need
  - Layered on IP's modest layer-3 service
- #1 workhorse: TCP (Transmission Control Protocol)
- Service provided by TCP:
  - Connection oriented (explicit set-up / tear-down)
    - End hosts (processes) can have multiple concurrent long-lived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

TCP “Bytestream” Service



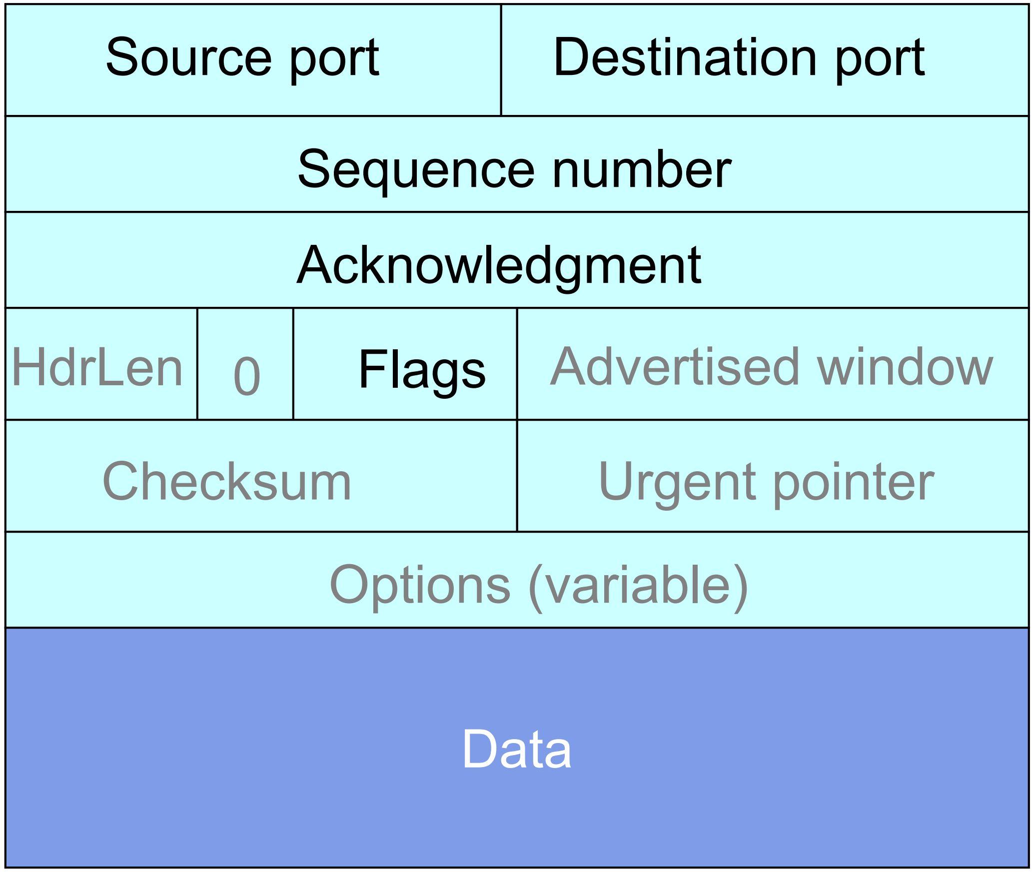
Bidirectional communication:

Process B on host H2



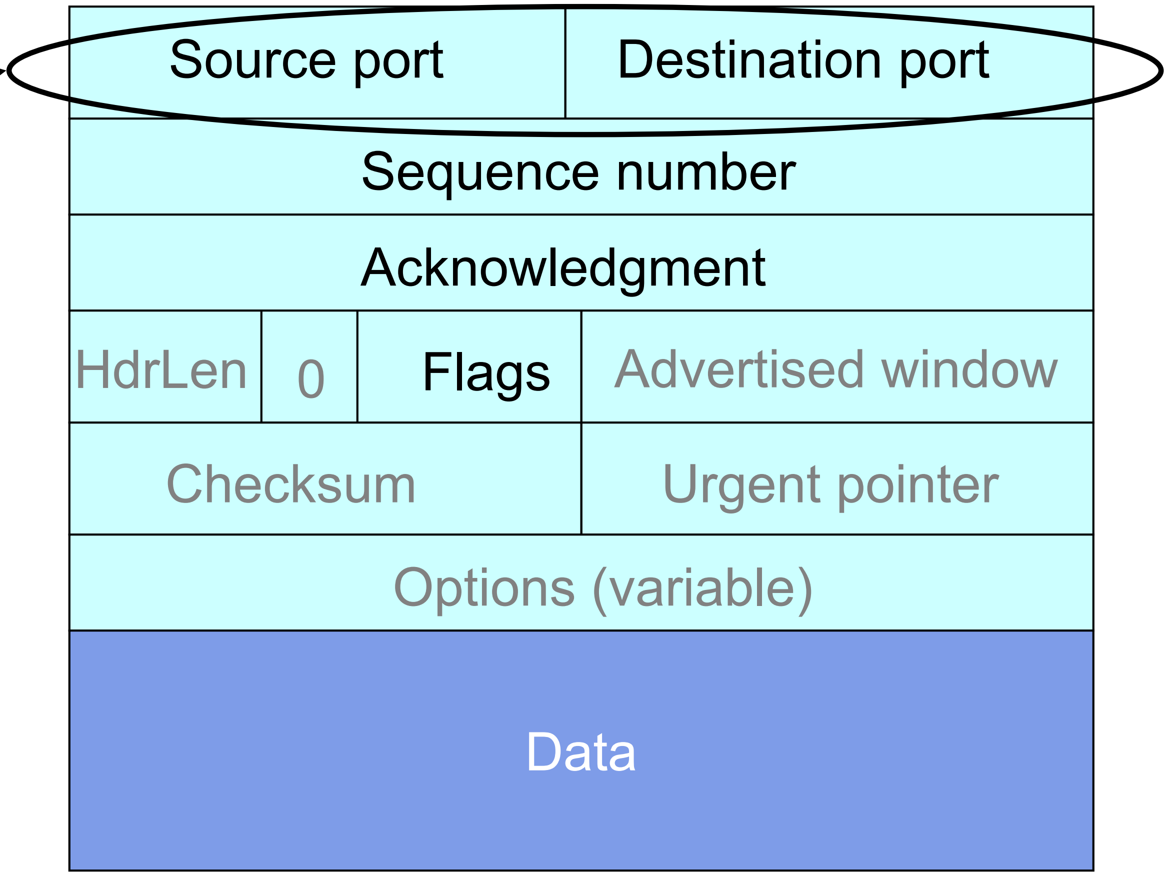
Process A
on host H1

TCP Header


Source port
Destination port
Sequence number
Acknowledgment
HdrLen
0
Flags
Advertised window
Checksum
Urgent pointer
Options (variable)
Data

TCP Header

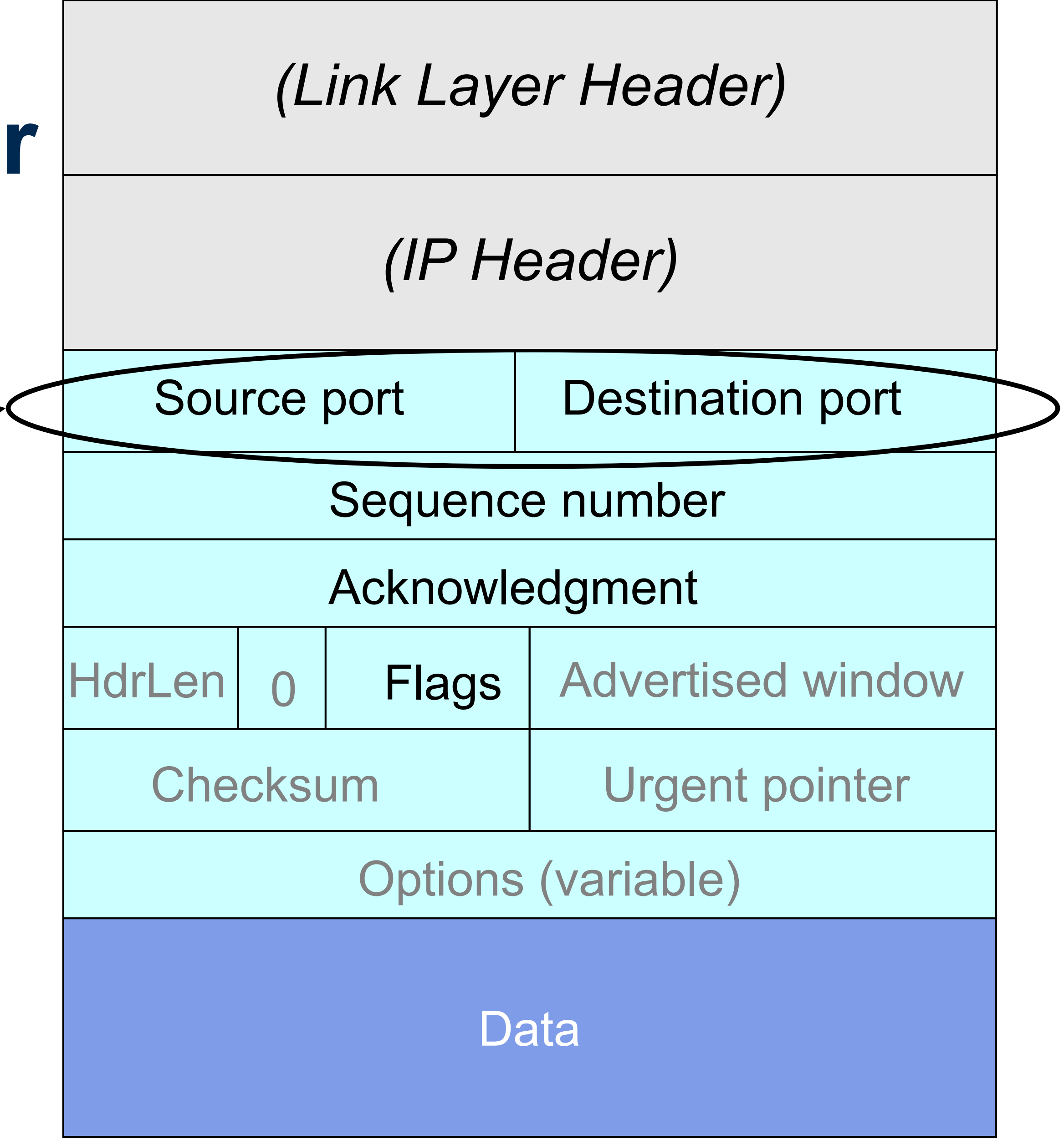
Ports are associated with OS processes



TCP Header

Ports are associated with OS processes

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

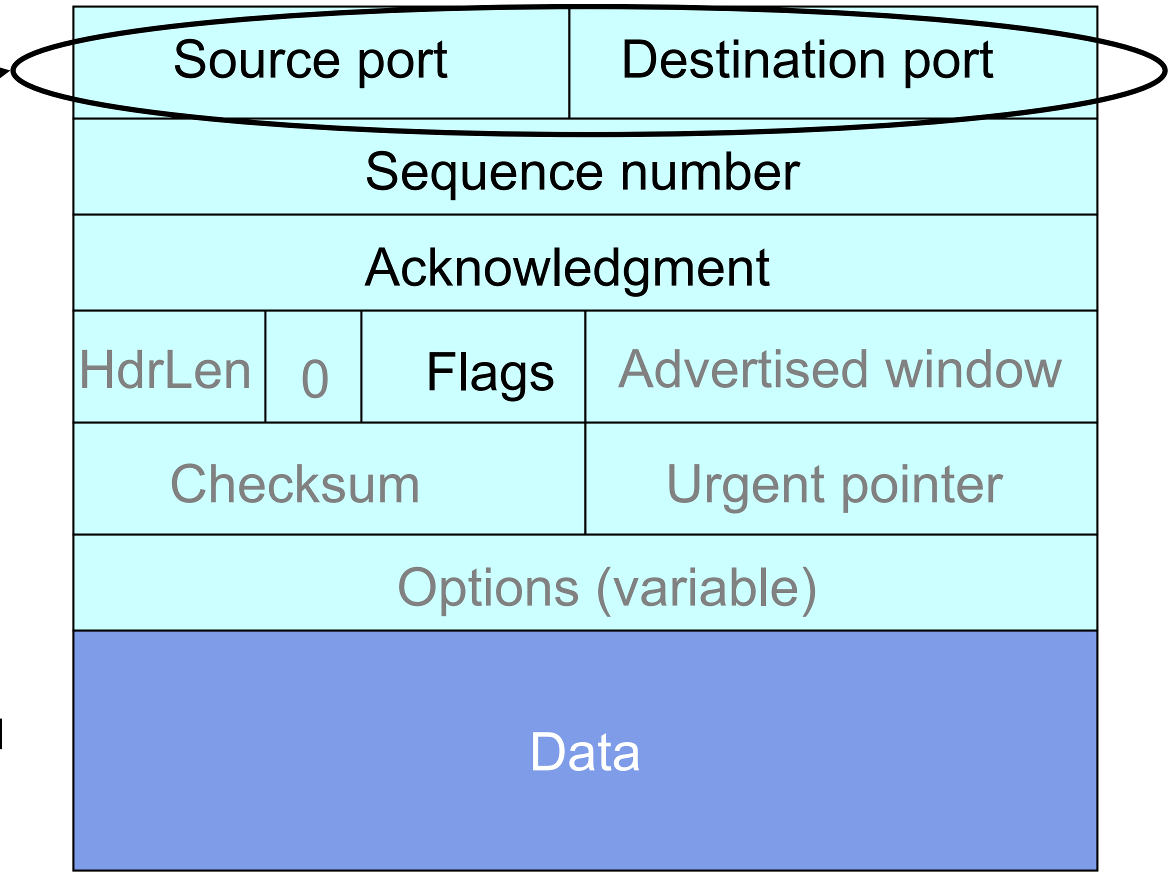


TCP Header

Ports are associated with OS processes

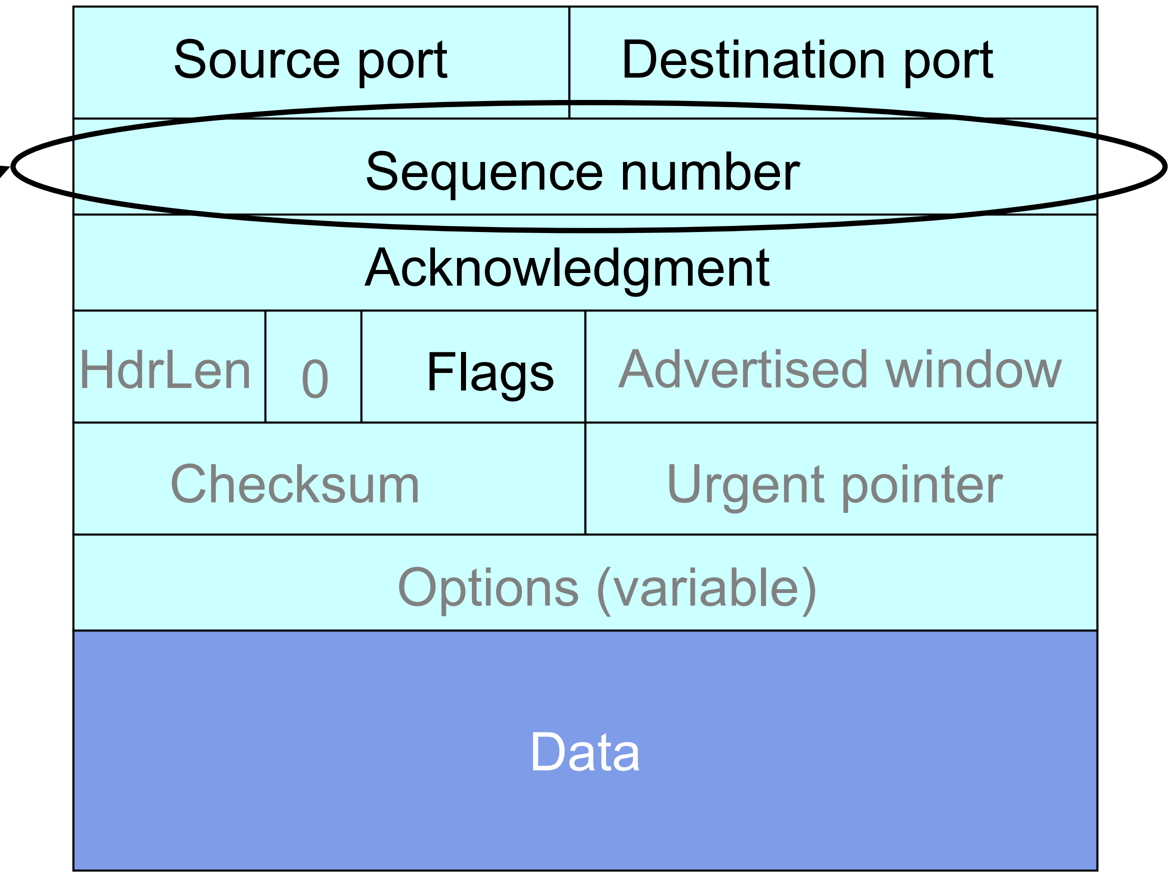
IP source & destination addresses plus TCP source and destination ports uniquely identifies a TCP connection

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



TCP Header

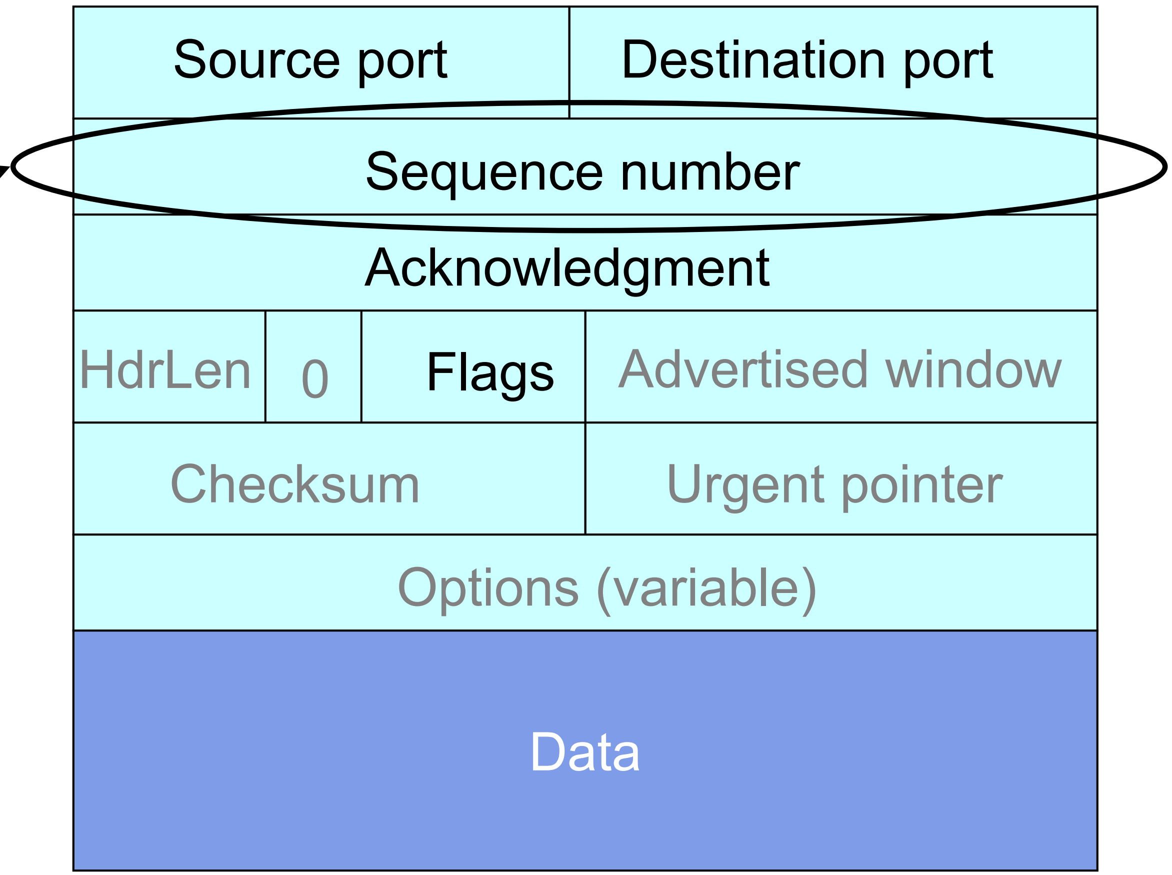
Starting sequence number (byte offset) of data carried in this packet



TCP Header

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

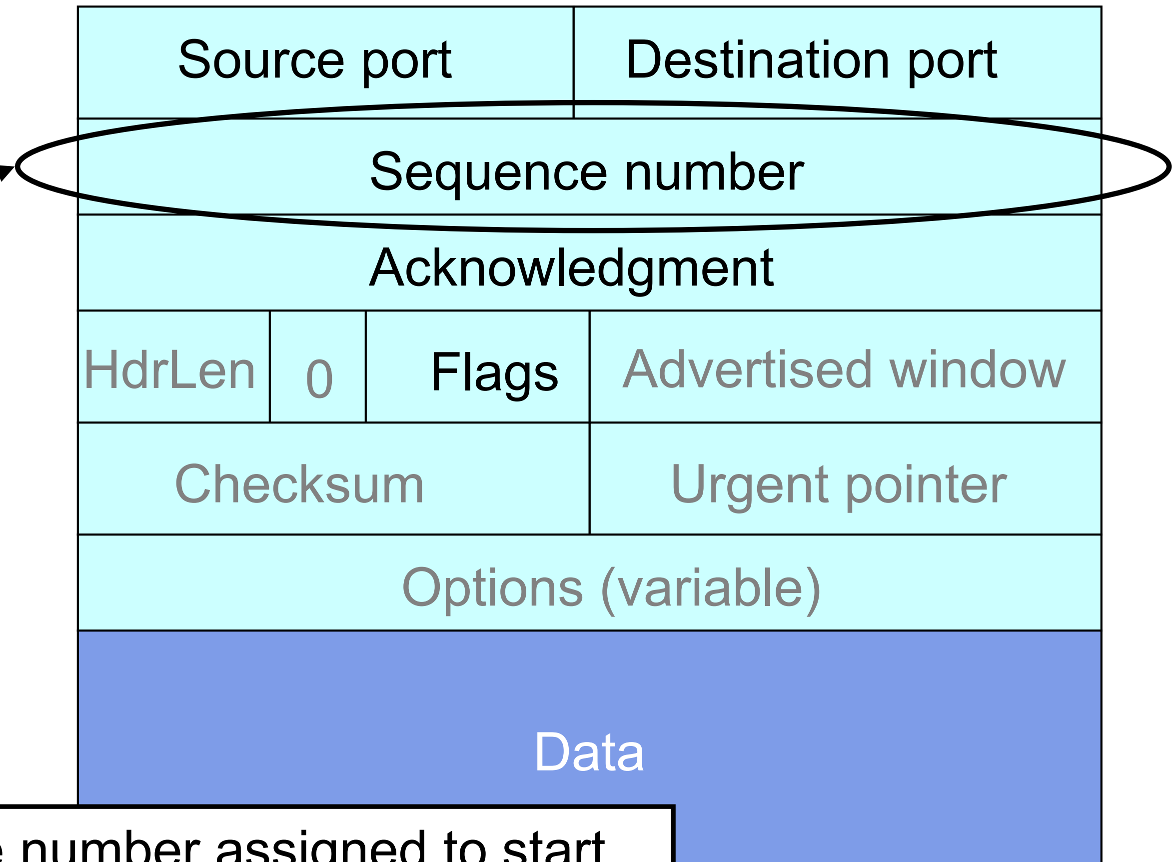
Byte streams numbered independently in each direction



TCP Header

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

Byte streams numbered independently in each direction

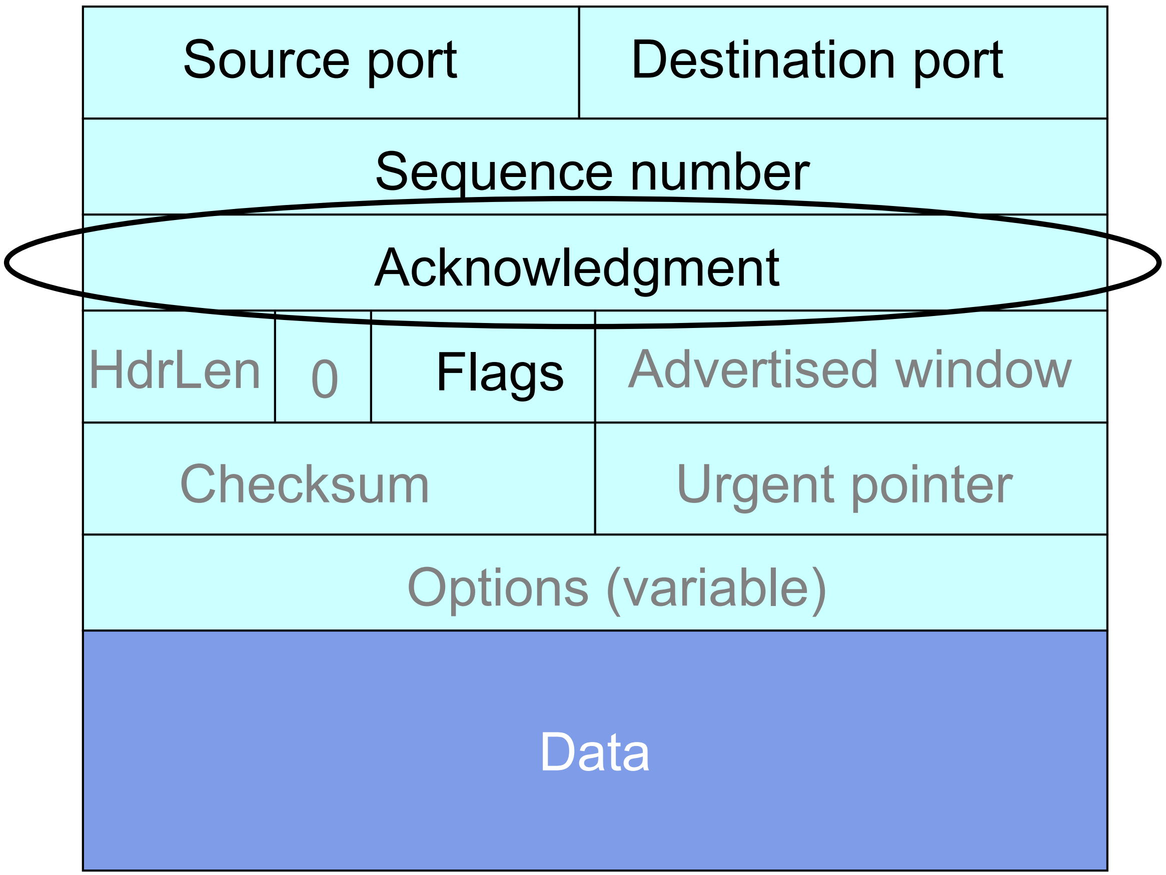


Sequence number assigned to start of byte stream is picked when connection begins; **doesn't** start at 0

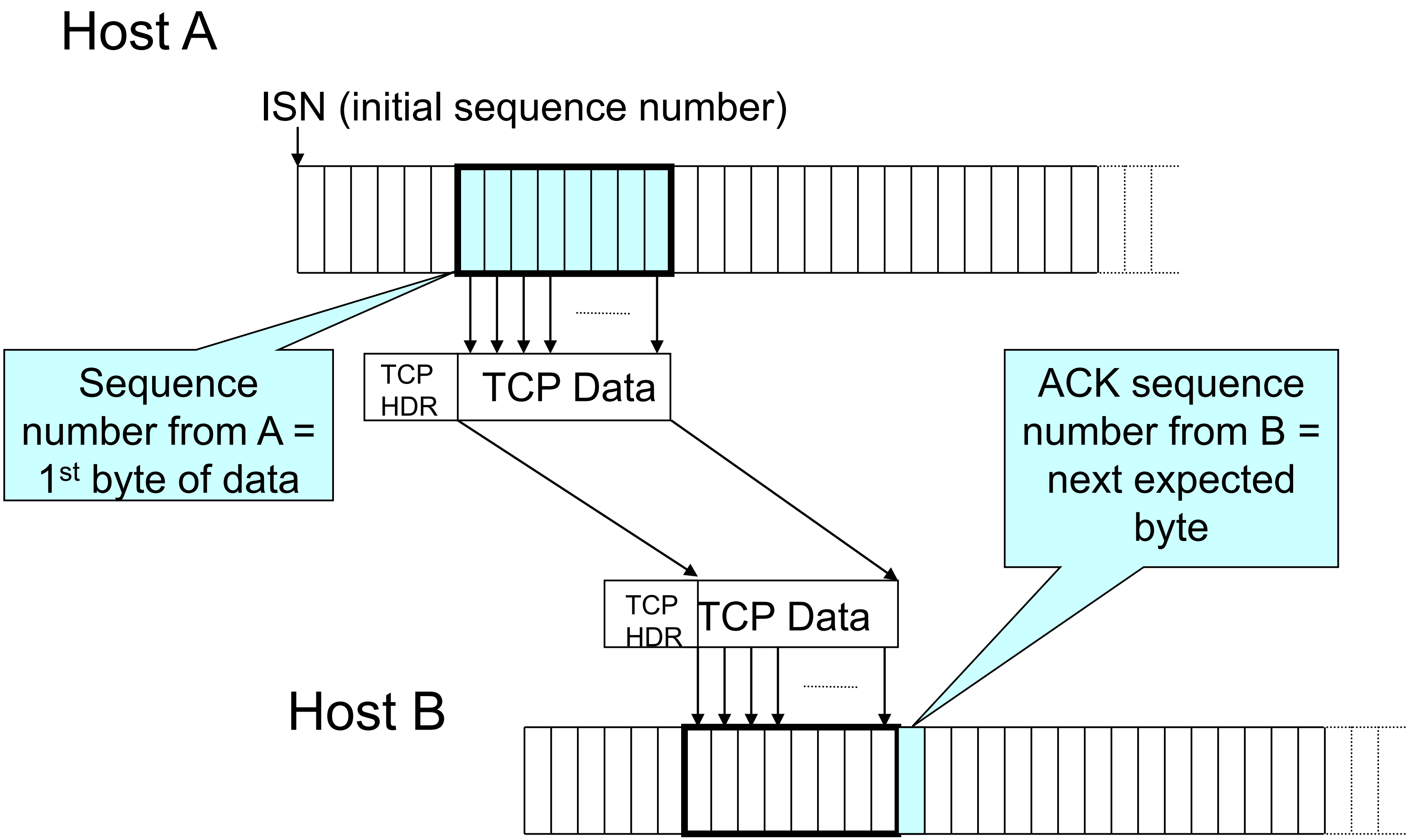
TCP Header

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

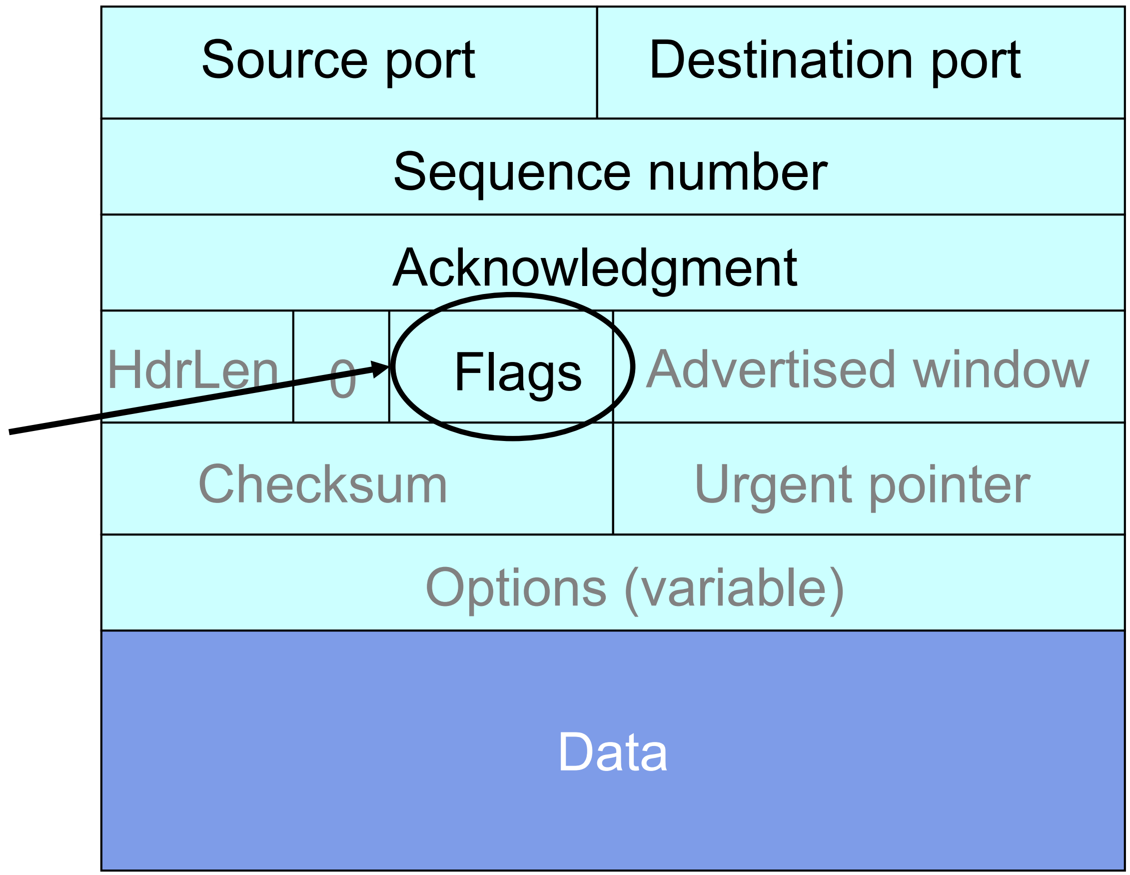


TCP Header

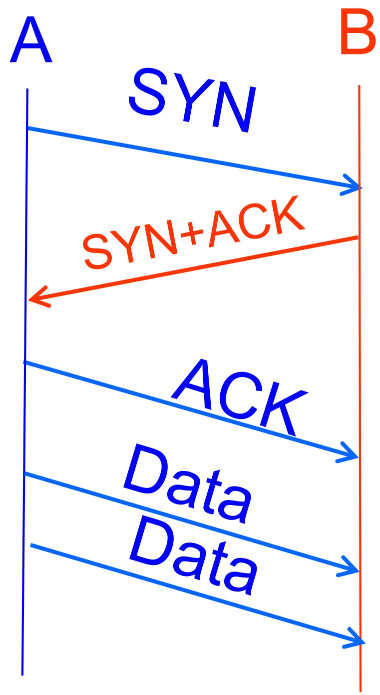
Uses include:

acknowledging data ("**ACK**")

setting up ("**SYN**") and closing connections ("**FIN**" and "**RST**")



Establishing a TCP Connection

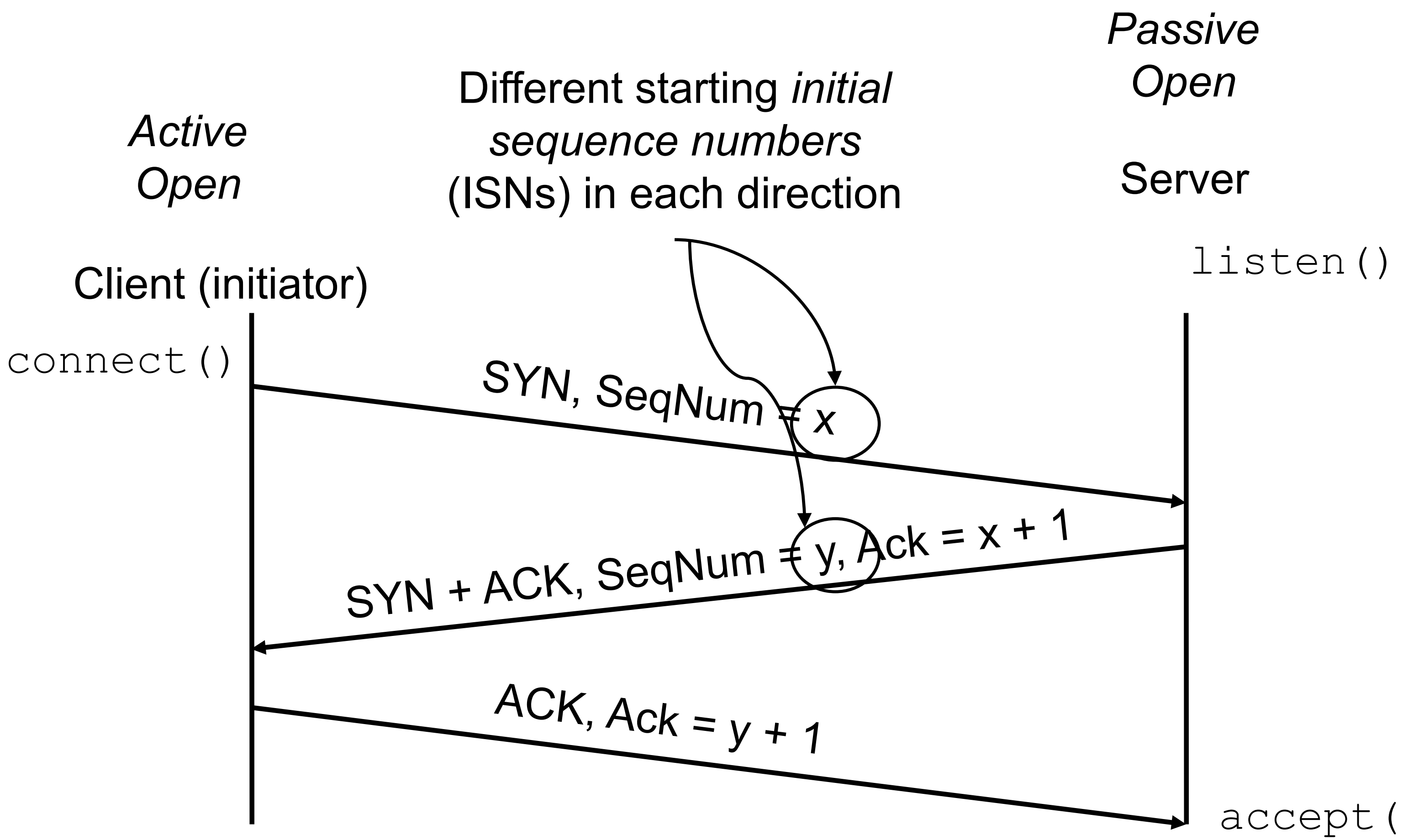


Each host tells its *Initial Sequence Number* (ISN) to the other host.

(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



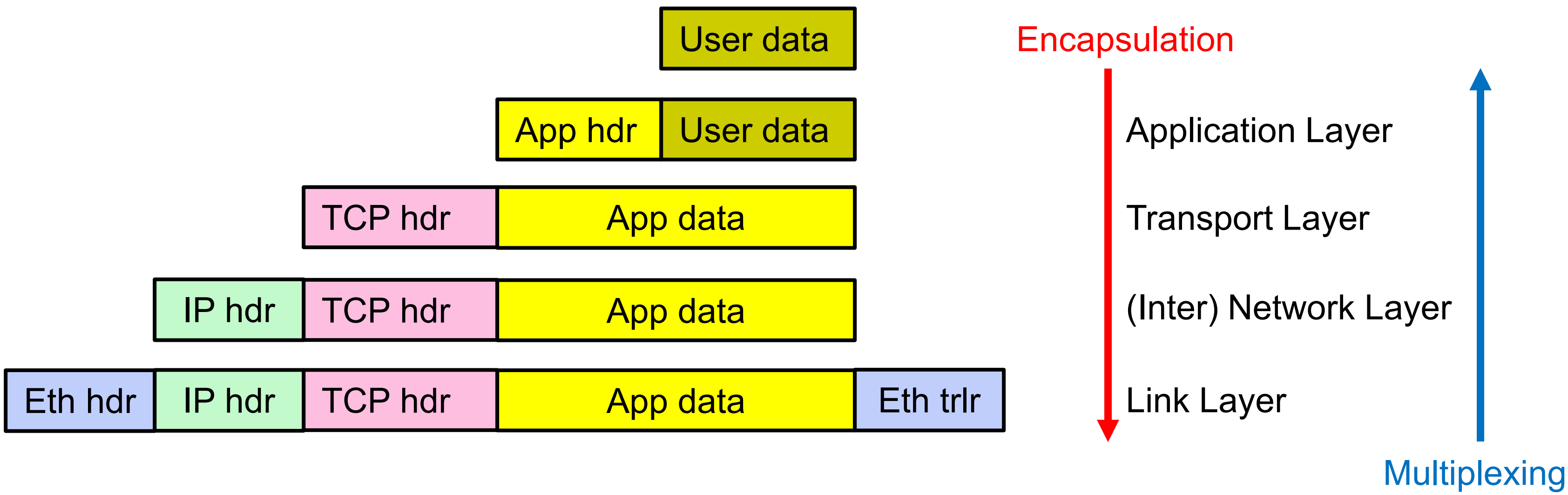
Example attack

- SYN flooding
  - Use anonymized source address
- If it comes from many sources
  - Distributed Denial of Service attack
- Will discuss in greater detail shortly

pcap: library for reading communications

- Packet Capturing Library: libpcap
  - OS independent
  - Easy support for both big-endian and little-endian
    - Network order is always big-endian
    - **`ntohl()`**
  - Simple and powerful user level library
- TCPDUMP & Wireshark implemented w/pcap
- Most commercial IDS systems use pcap
  - Intrusion Detection Systems
- For intro, read http://www.tcpdump.org

Packet capture



- Packet capturing (sniffing) does not affect data transfer
- The packet captured by libpcap is called raw packet and demultiplexing is required to analyze the packet

Initializing Packet Capturing APIs

- **`pcap_t *pcap_open_live(const char *device, int snaplen, int promisc, int to_ms, char *errbuf)`**
  - obtain a packet capture descriptor to look at packets on the network
  - **`snaplen`**: maximum number of *bytes* to capture
  - **`promisc`**: true, set the interface into *promiscuous mode*; false, only bring packets intended for you
  - **`to_ms`**: *read timeout* in milliseconds; *zero*, cause a read to wait forever to allow enough packets to arrive
  - return *NULL* indicates an error

Initializing Packet Capturing APIs

- **`pcap_t *pcap_open_offline(const char *fname, char *errbuf);`**
  - open a pcap file for reading
  - *fname*: the name of the file to open
  - return a *pcap_t ** on success and *NULL* on failure

pcap_pkthdr

```
struct pcap_pkthdr {
   struct timeval ts; /* time stamp */
   bpf_u_int32 caplen; /* length of portion present */
   bpf_u_int32 len; /* length this packet (off wire) */
};
```

Ethernet header

```
/* Ethernet header */
struct sniff_ethernet {
        u_char ether_dhost[ETHER_ADDR_LEN]; /* Dest host addr */
        u_char ether_shost[ETHER_ADDR_LEN]; /* Src host addr */
        u_short ether_type; /* IP? ARP? RARP? etc */
};
```

IP header

```
/* IP header */
struct sniff_ip {
        u_char ip_vhl; /* version << 4 | header length >> 2 */
        u_char ip_tos; /* type of service */
        u_short ip_len; /* total length */
        u_short ip_id; /* identification */
        u_short ip_off; /* fragment offset field */
#define IP_RF 0x8000 /* reserved fragment flag */
#define IP_DF 0x4000 /* dont fragment flag */
#define IP_MF 0x2000 /* more fragments flag */
#define IP_OFFMASK 0x1fff /* mask for fragmenting bits */
        u_char ip_ttl; /* time to live */
        u_char ip_p; /* protocol */
        u_short ip_sum; /* checksum */
        struct in_addr ip_src,ip_dst; /* src & dest address */
};
#define IP_HL(ip) (((ip)->ip_vhl) & 0x0f)
#define IP_V(ip) (((ip)->ip_vhl) >> 4)
```

TCP header (part 1)

```
#define TH_FIN 0x01
#define TH_SYN 0x02
#define TH_RST 0x04
#define TH_PUSH 0x08
#define TH_ACK 0x10
#define TH_URG 0x20
#define TH_ECE 0x40
#define TH_CWR 0x80
#define TH_FLAGS \
  (TH_FIN|TH_SYN|TH_RST|TH_ACK|TH_URG|TH_ECE|TH_CWR)
#define TH_OFF(th) (((th)->th_offx2 & 0xf0) >> 4)

typedef u_int tcp_seq;
```

TCP header (part 2)

```
/* TCP header */
struct sniff_tcp {
        u_short th_sport; /* source port */
        u_short th_dport; /* destination port */
        tcp_seq th_seq; /* sequence number */
        tcp_seq th_ack; /* acknowledgement number */
        u_char th_offx2; /* data offset, rsvd */
        u_char th_flags; /* see previous page */
        u_short th_win; /* window */
        u_short th_sum; /* checksum */
        u_short th_urp; /* urgent pointer */
};
```

Packet Read Related APIs

- `const u_char *pcap_next(pcap_t *p, struct pcap_pkthdr *h)`
  - read the next packet
  - return *NULL* indicates an error
- `int pcap_loop(pcap_t *p, int cnt, pcap_handler callback, u_char *user)`
  - processes packets from a live capture or "savefile'" until *cnt* packets are processed
  - A value of -1 or 0 for *cnt* is equivalent to *infinity*
  - *callback* specifies a routine to be called

Software based on Libpcap

- ntop - network top
  - network traffic probe; shows network usage
  - http://www.ntop.org/overview.html
- snort
  - intrusion prevention & detection system
  - http://www.snort.org/
- ethereal
  - network protocol analyzer
  - http://www.ethereal.com/
- wireshark
  - http://www.wireshark.org/

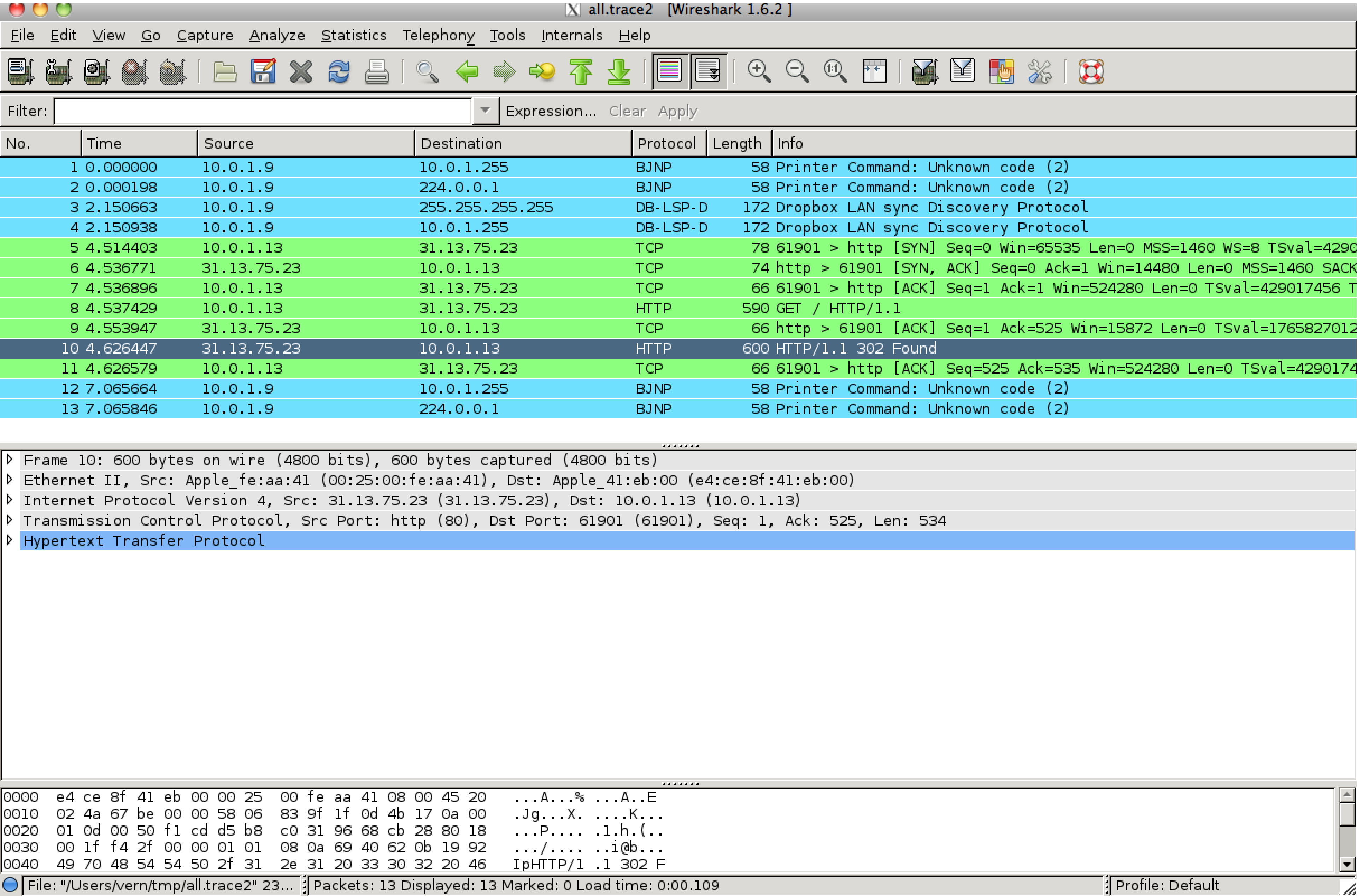
Eavesdropping

- Subnets using broadcasting technology get eavesdropping for free
  - WiFi
  - Ethernet
- Network interface card can capture all packets on net
- Tools
  - tcpdump, windump (low level)
  - Wireshark (GUI supporting 800+ protocols)

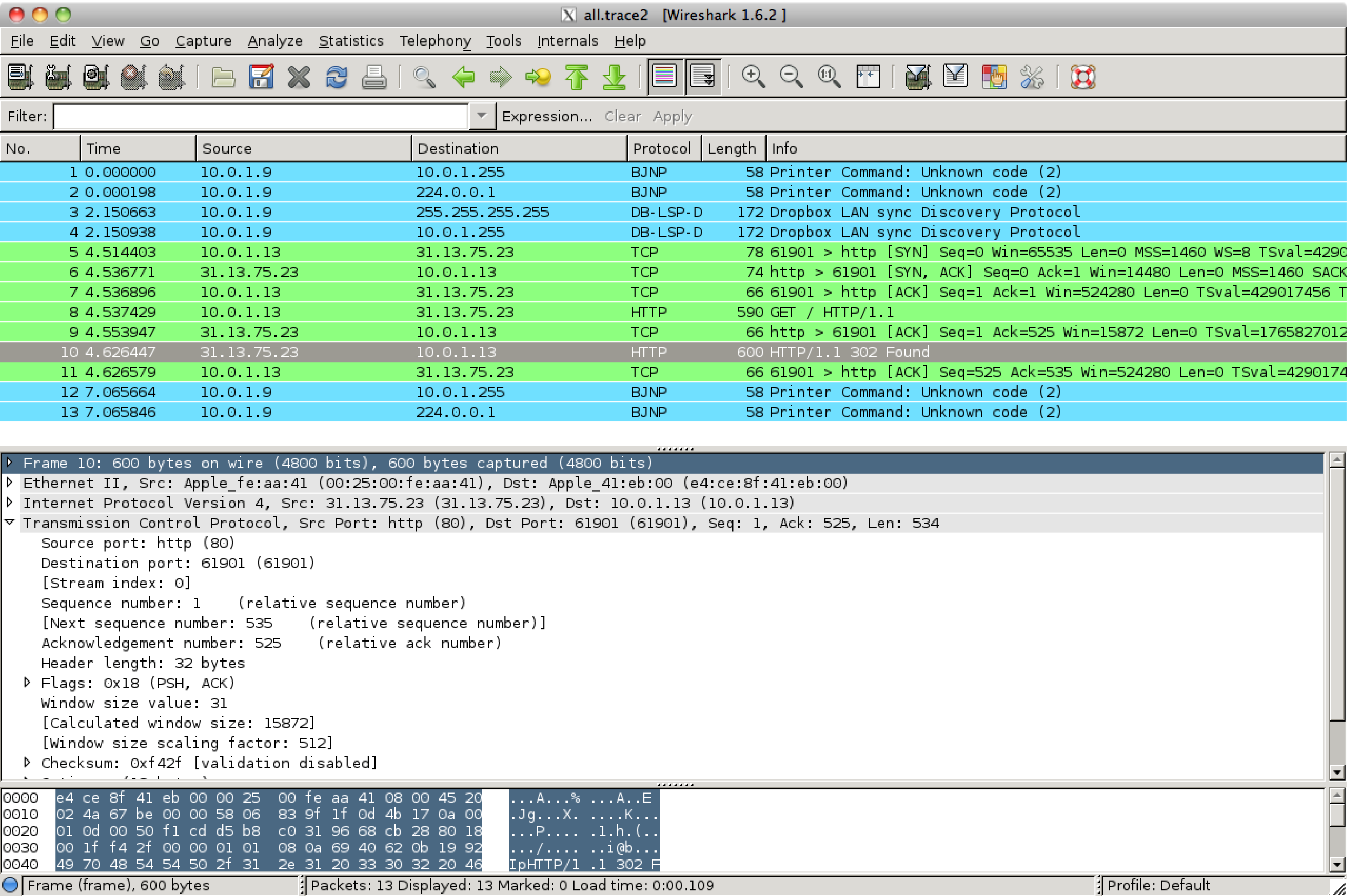
TCPDUMP: Packet Capture & ASCII Dumper

```
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 2
523449627, win 65535, options [mss 1460,nop,wscale 3,nop,nop,TS val 429017455 ecr 0,sack
OK,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 42
9017455,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 5
25, 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 5
35, 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
```

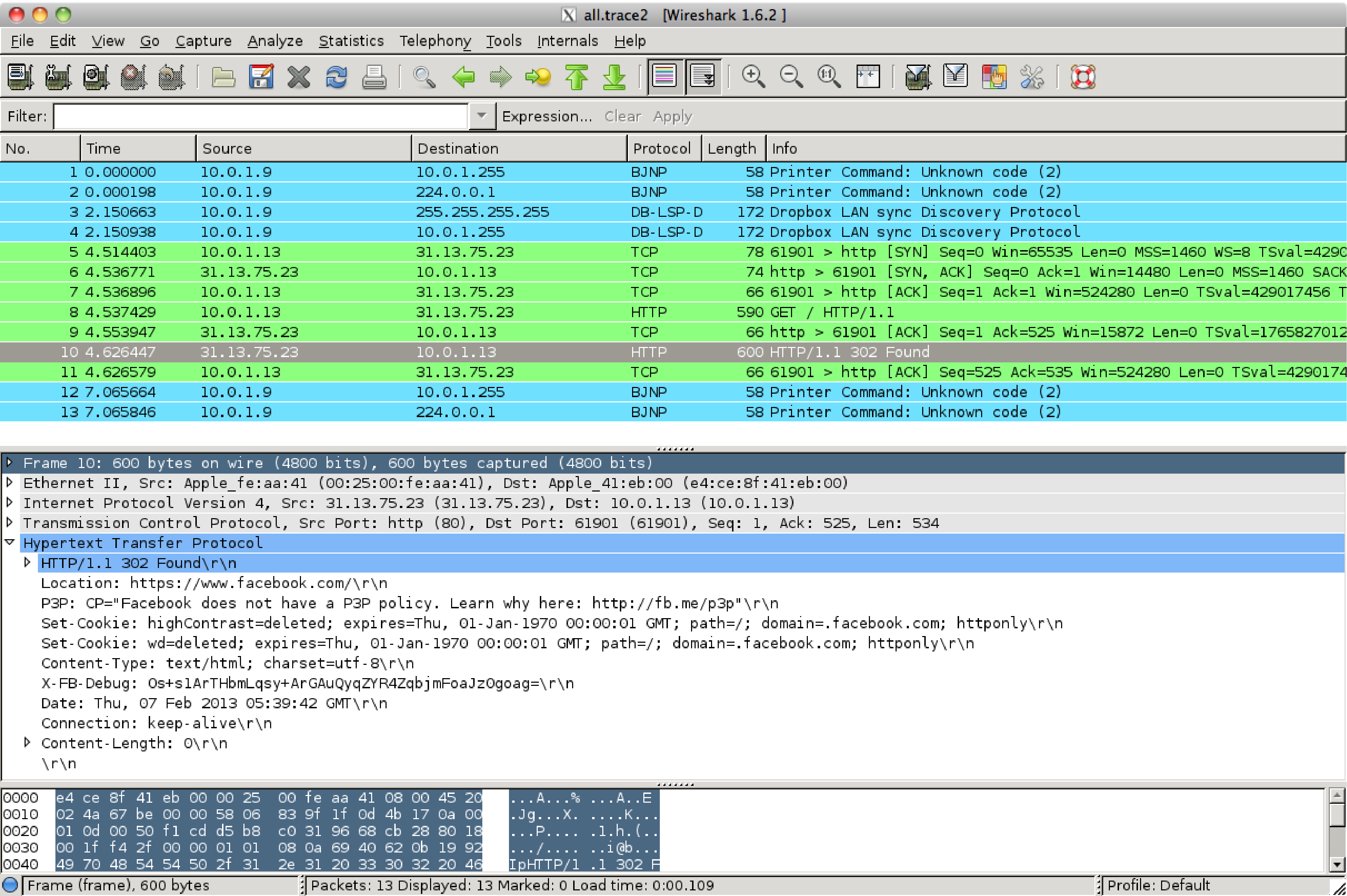
Wireshark: GUI for Packet Capture/Exam.



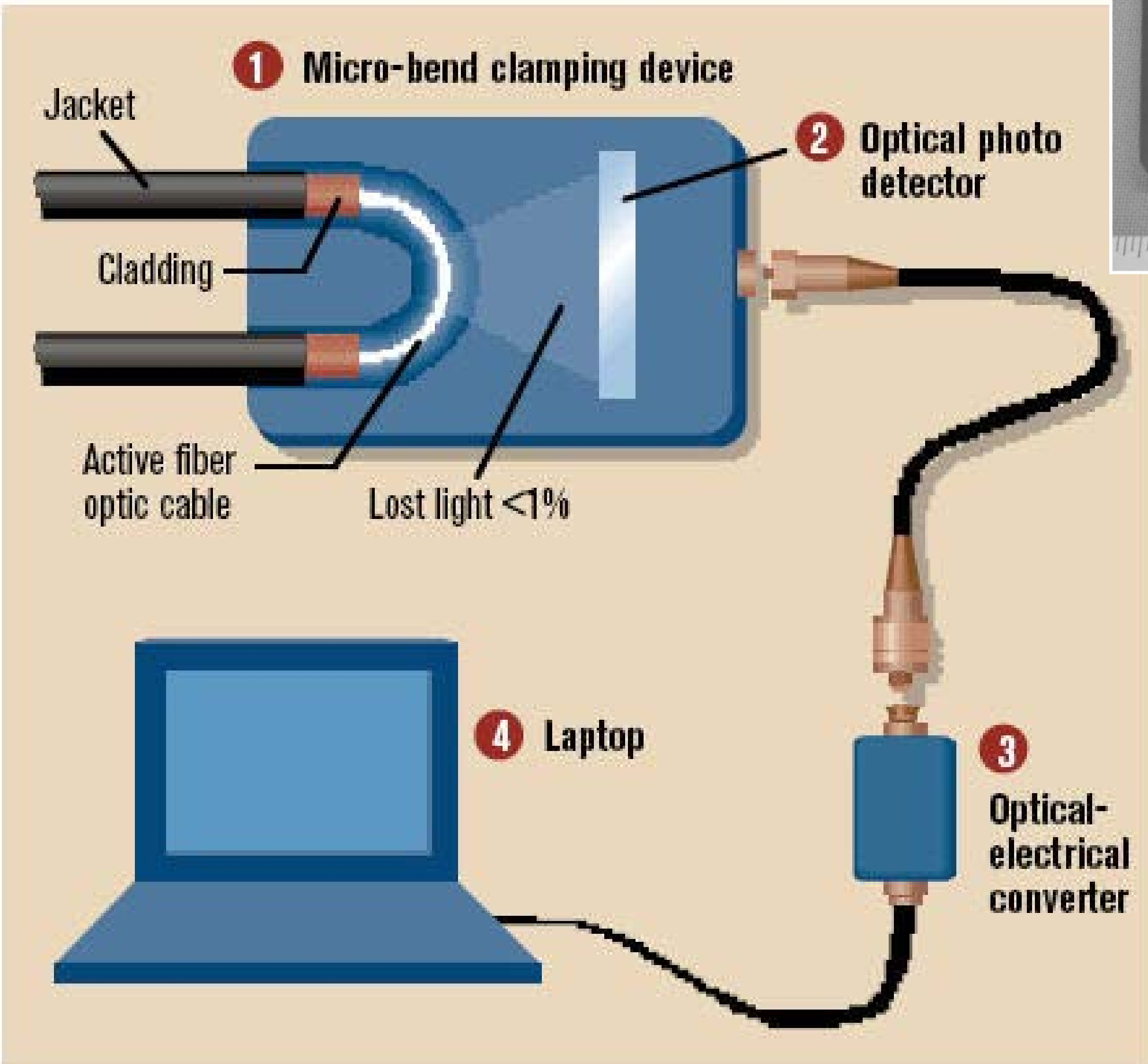
Wireshark: GUI for Packet Capture/Exam.



Wireshark: GUI for Packet Capture/Exam.

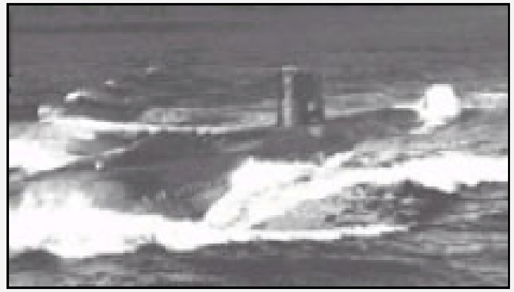


Stealing Photons



Operation Ivy Bells

By Matthew Carle
Military.com



The Regulus guided missile submarine, USS Halibut (SSN 587) which carried out Operation Ivy Bells.

At the beginning of the 1970's, divers from the specially-equipped submarine, USS Halibut (SSN 587), left their decompression chamber to start a bold and dangerous mission, code named "Ivy Bells".

In an effort to alter the balance of Cold War, these men scoured the ocean floor for a five-inch diameter cable carry secret Soviet communications between military bases.

The divers found the cable and installed a 20-foot long listening device on the cable. designed to attach to the cable without piercing the casing, the device recorded all communications that occurred. If the cable malfunctioned and the Soviets raised it for repair, the bug, by design, would fall to the bottom of the ocean. Each month Navy divers retrieved the recordings and installed a new set of tapes.

Upon their return to the United States, intelligence agents from the NSA analyzed the recordings and tried to decipher any encrypted information. The Soviets apparently were confident in the security of their communications lines, as a surprising amount of sensitive information traveled through the lines without encryption.

prison. The original tap that was discovered by the Soviets is now on exhibit at the KGB museum in Moscow.

Link Layer Threat Disruption

- Attackers can “jam” packets they don’t like (integrity)
- Attackers can overwhelm signaling (e.g., WiFi radio signals)

- There’s also the heavy handed approach

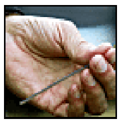
Sabotage attacks knock out phone service

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

(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

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.

"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.

$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

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

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

D	C	Hello world!

Physical/Link-Layer Threats: Spoofing

- Physical access allows attacker to create any message it likes
  - With a bogus source address: spoofing
  - May require root/administrative access
- Particularly powerful combined w/eavesdropping
  - Attacker understands state of victim's communication
  - Crafts communication to exploit it
- Spoofing w/o eavesdropping
  - *Blind spoofing*