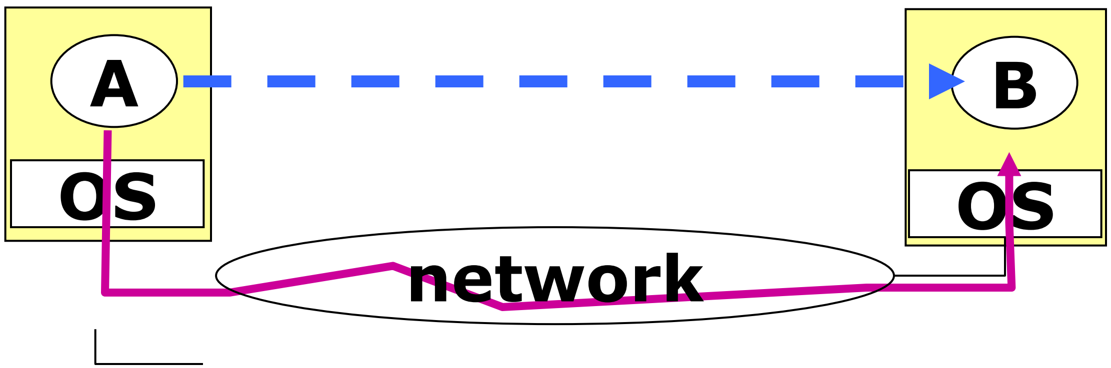
Chapter 3:
Networking and Internetworking

- Concepts
- Switching
- Routing (IP)
- End-to-End Protocols (UDP/TCP)

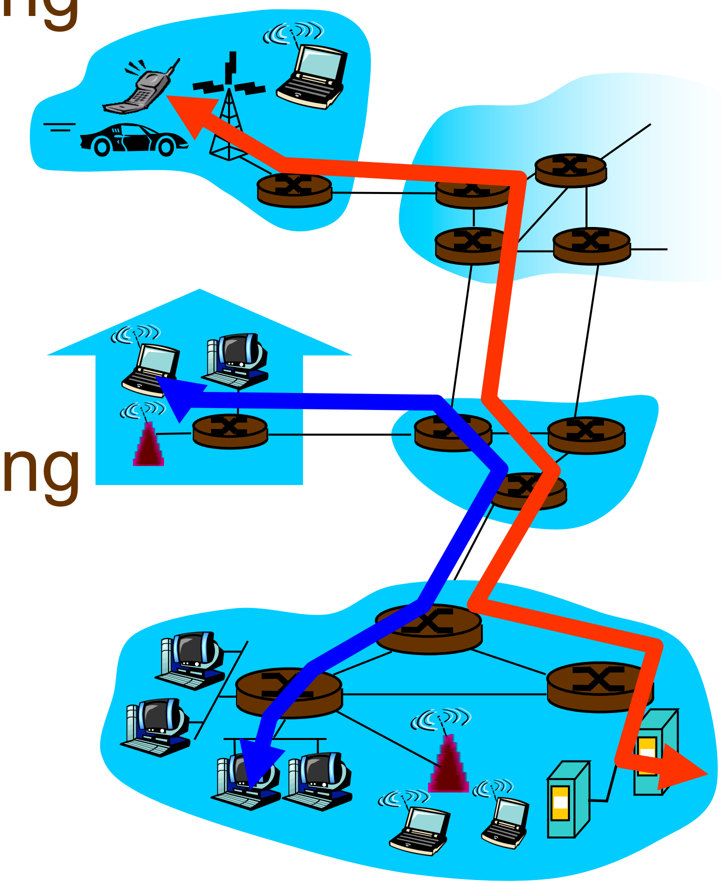
How can A and B communicate?



- Many different agreements (protocols) are needed at various levels
- **Application-level agreements**
  - Bit representation to meaning of each message
- **Other-levels and agreements**
  - How to actually transmit messages through a network
  - Addressing, performance, scalability, reliability, security

What's Network (the Internet)?
To learn more, take CECS 303

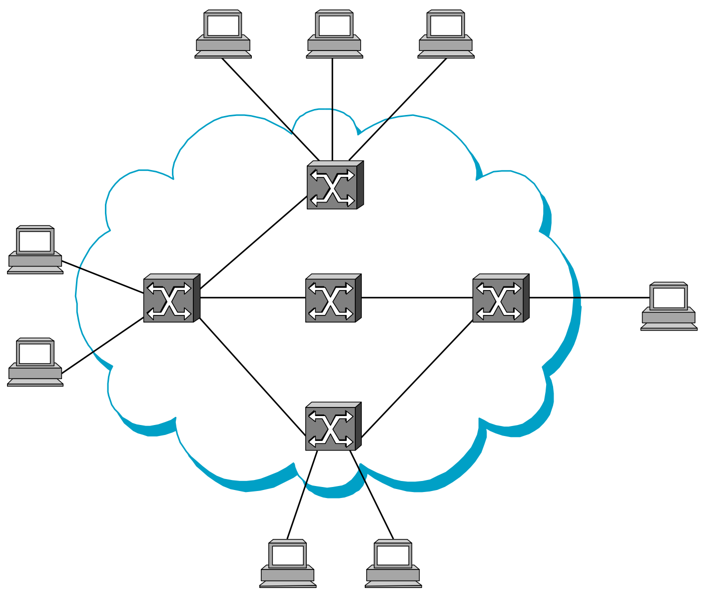
- Network of networks connecting millions of devices:
  - Hosts (end systems)
  - Links (fiber to satellite)
  - Routers and switches
- Collection of protocols providing communication services to distributed applications
- Networks are complex!

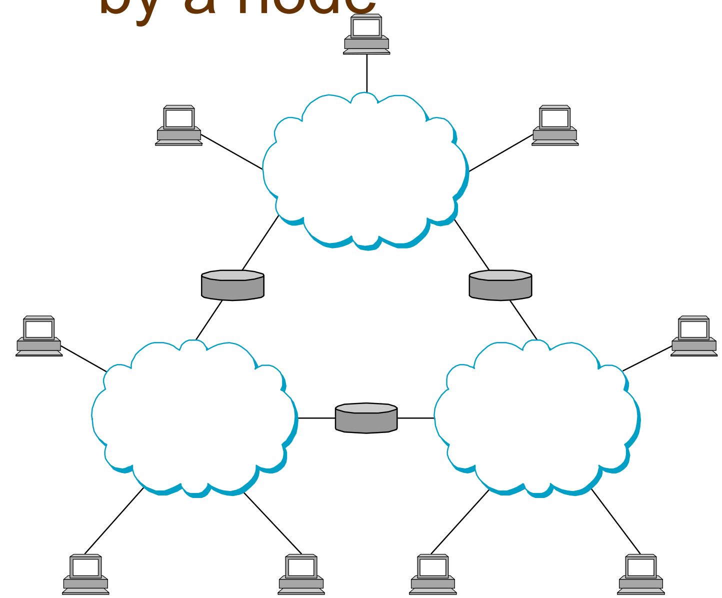


From Computer Networking by Kurose and Ross.

Switched Networks

- A network can be defined recursively as...
  - two or more nodes connected by a link, or
  - two or more networks connected by a node





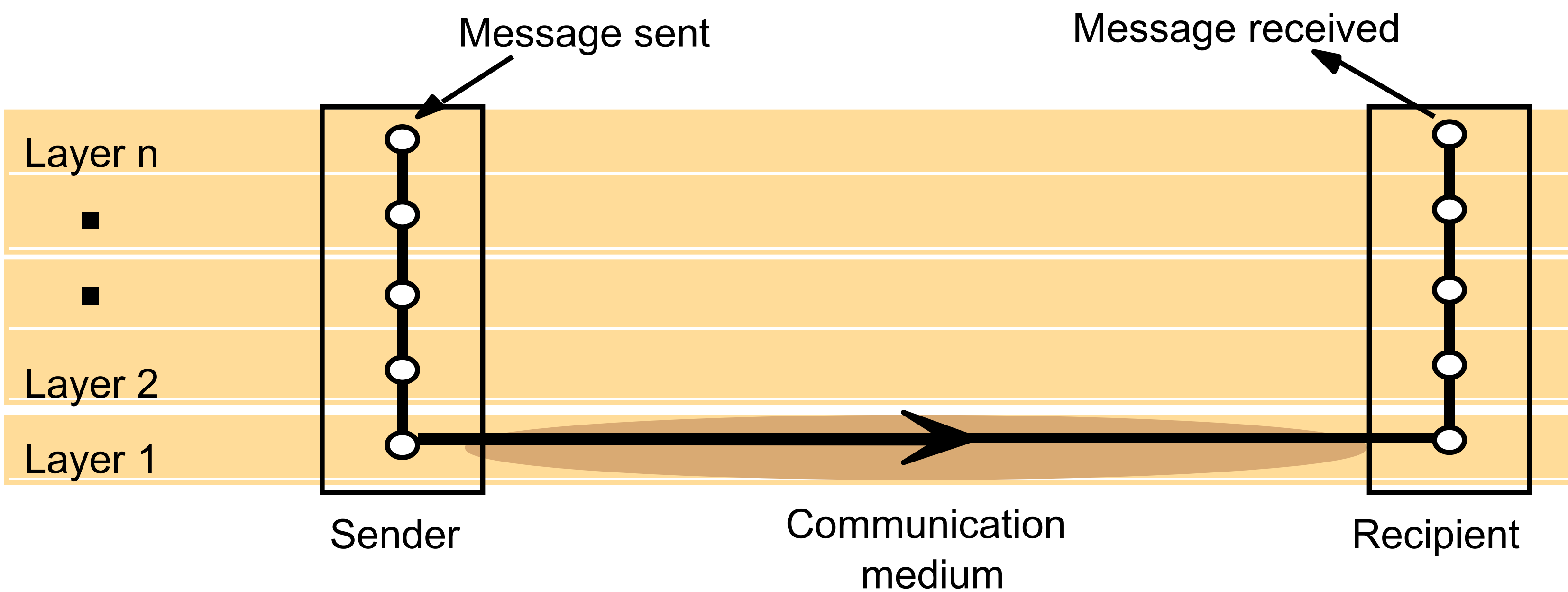
Types of Networks

	Example	*Range*	*Bandwidth (Mbps)*	*Latency (ms)*
Wired:				
LAN	Ethernet	1–2 kms	10–10,000	1–10
WAN	IP routing	worldwide	0.010–600	100–500
MAN	ATM	2–50 kms	1–600	10
Internetwork	Internet	worldwide	0.5–600	100–500
Wireless:				
WPAN	Bluetooth (IEEE 802.15.1)	10–30m	0.5–2	5–20
WLAN	WiFi (IEEE 802.11)	0.15–1.5 km	11–108	5–20
WMAN	WiMAX (IEEE 802.16)	5–50 km	1.5–20	5–20
WWAN	3G phone	cell: 1—5 km	348–14.4	100–500

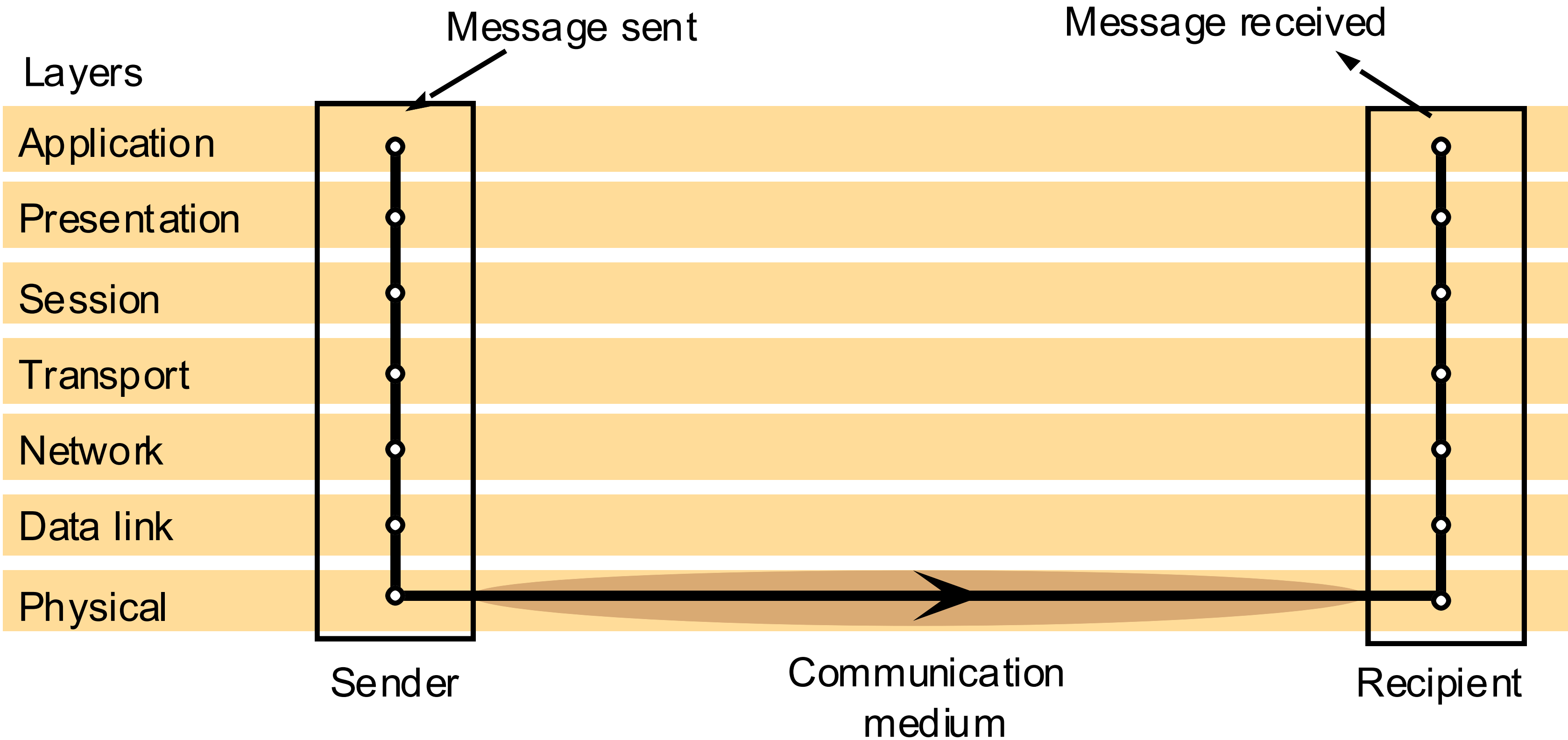
https://www.youtube.com/watch?v=HLziLmaYsO0

Protocols and Layers

Conceptual layering of protocol software

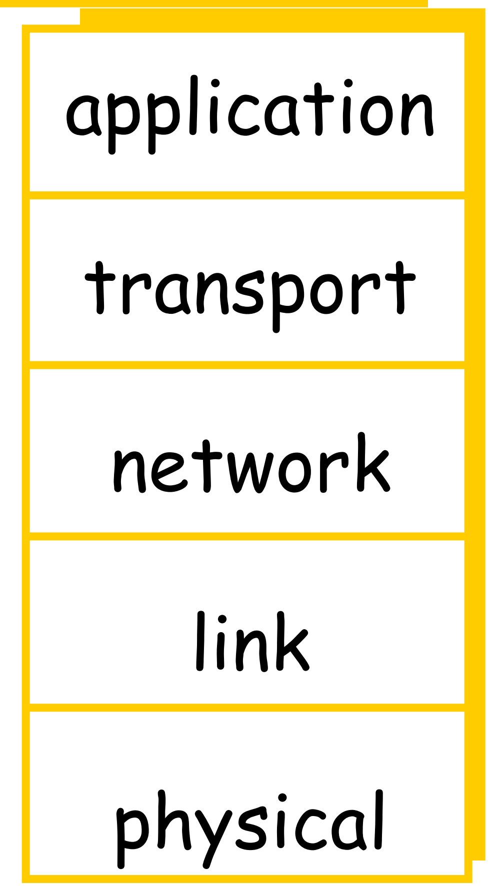


Protocol layers in the ISO Open Systems Interconnection (OSI) model



Internet protocol stack

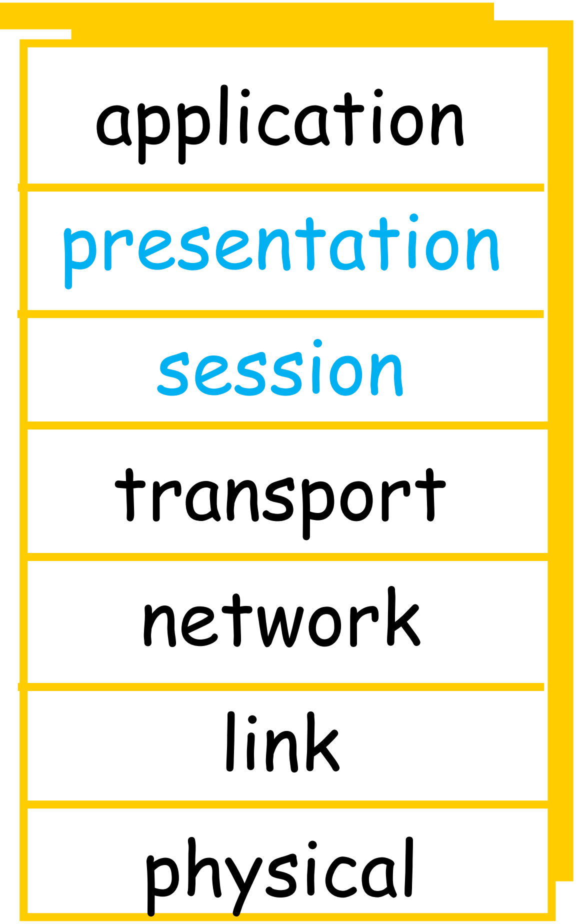
- application: Protocols that are designed to meet the communication requirements of specific applications, often defining the interface to a service. (FTP, HTTP)
- transport: process-to-process data transfer (TCP, UDP)
- network: routing of datagrams from source to destination (IP, OSPF, BGP)
- link: data transfer between neighboring network elements (PPP, Ethernet)
- physical: transmission of bits on a link (electrical signals on cable, light signals on fibre or other electromagnetic signals on radio)



From Computer Networking by Kurose and Ross.

ISO/OSI reference model

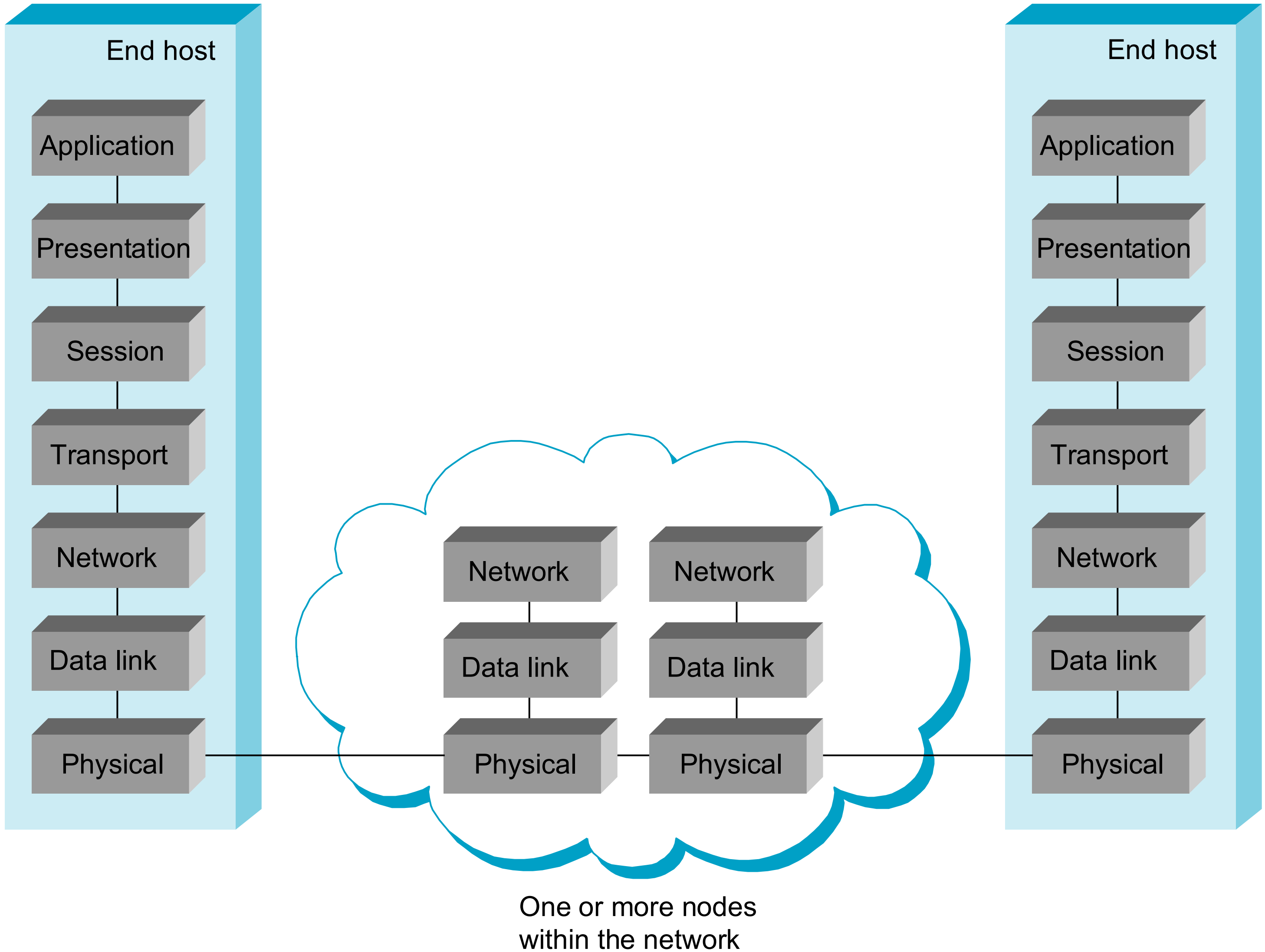
- presentation: allow applications to interpret meaning of data, e.g., encryption, compression, machine-specific conventions
- *session:* synchronization, check pointing, recovery of data exchange



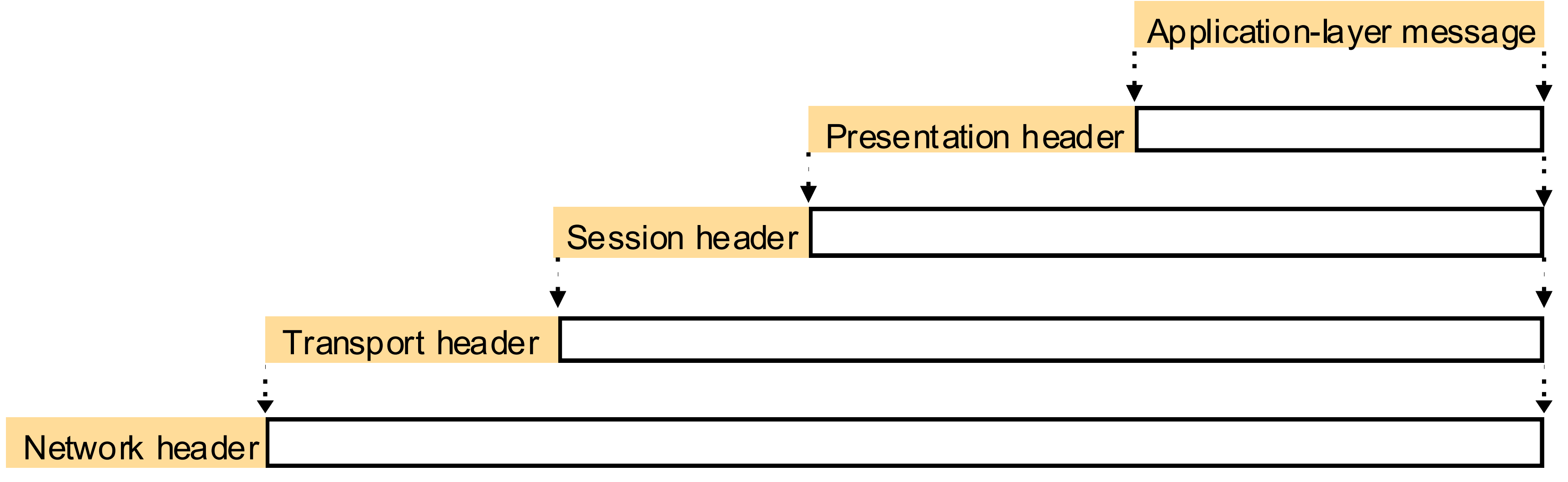
What is OSI mode?

https://www.youtube.com/watch?v=Ilk7UXzV_Qc

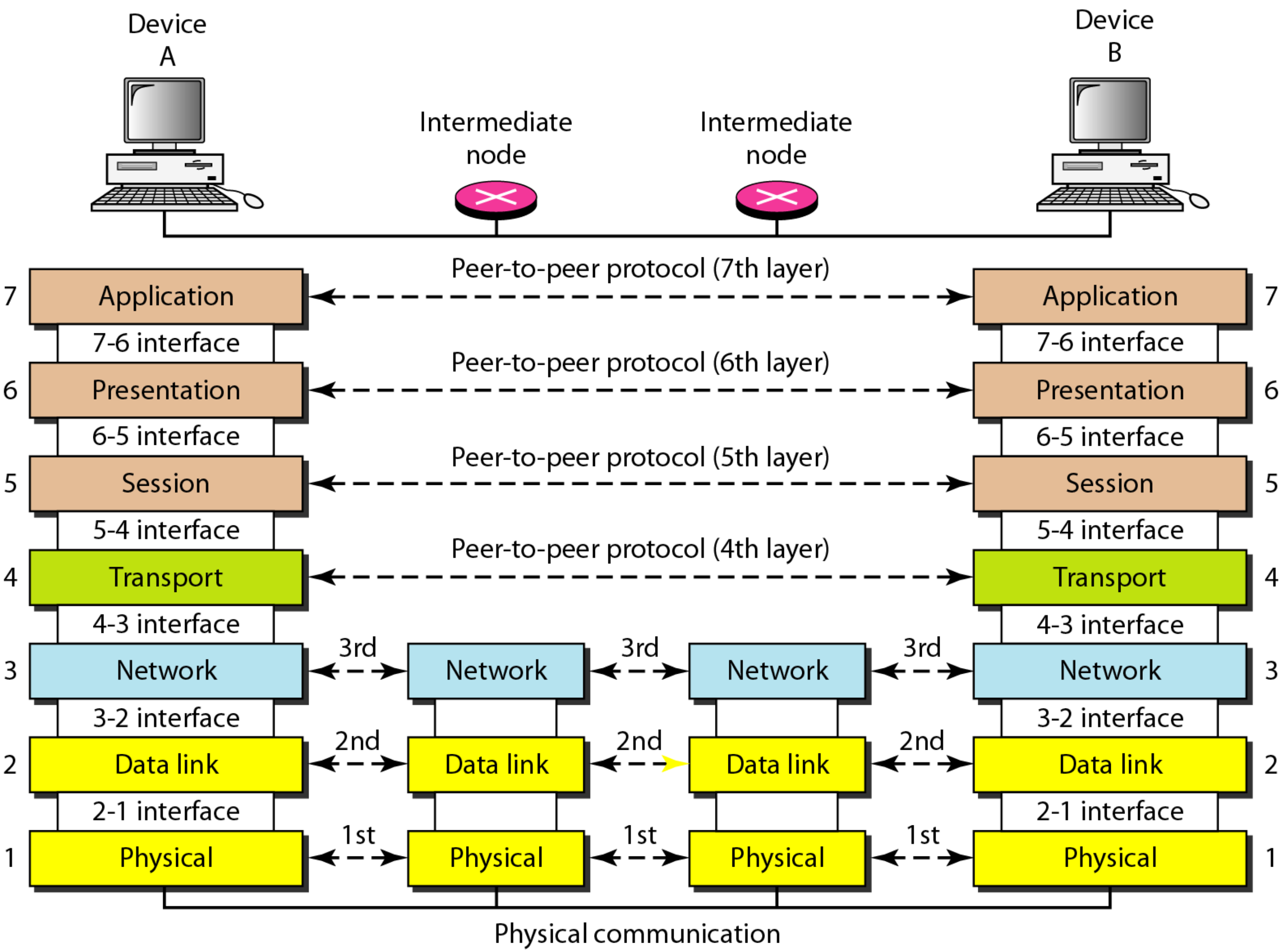
ISO Architecture



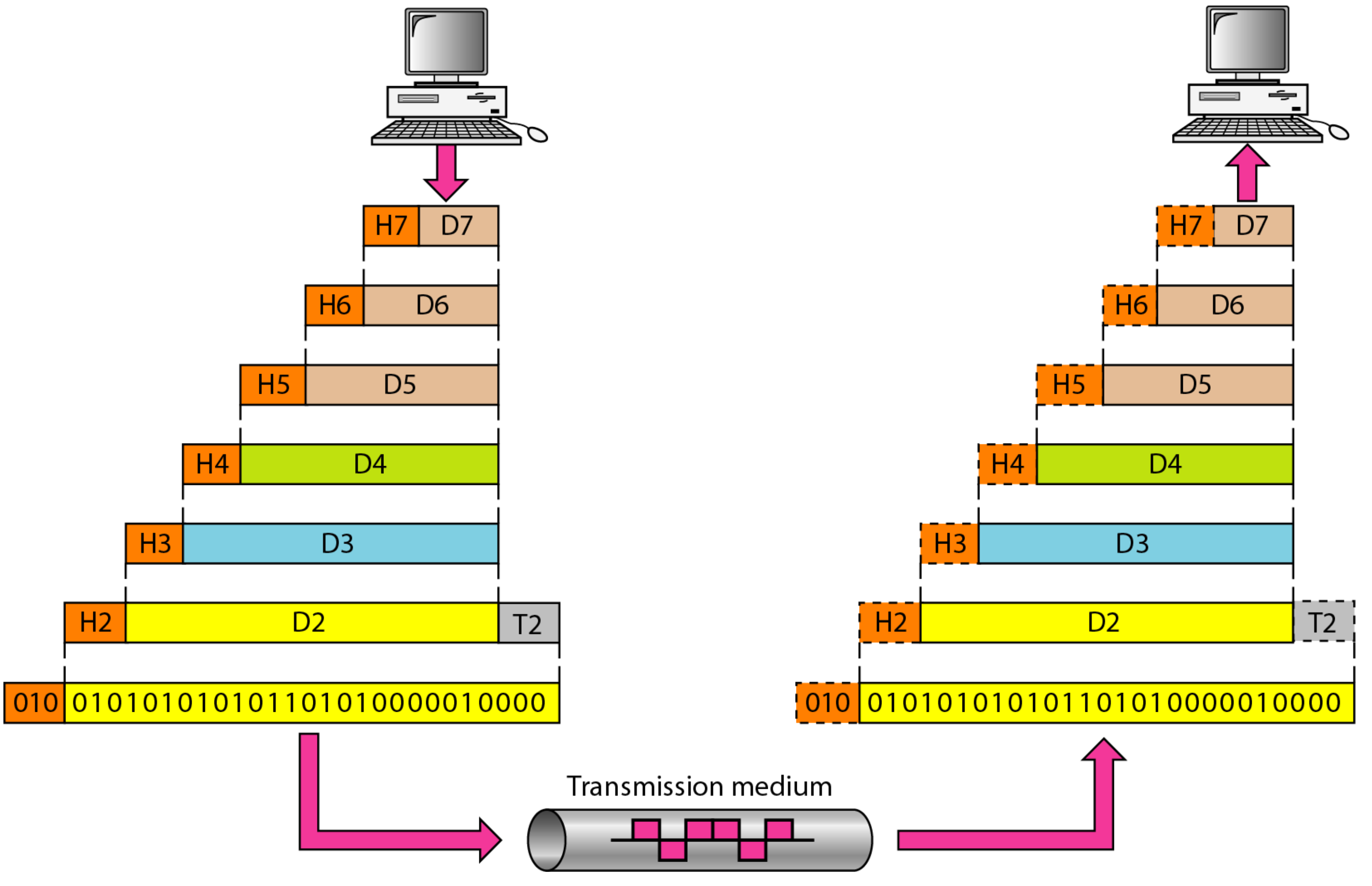
Encapsulation as it is applied in layered protocols



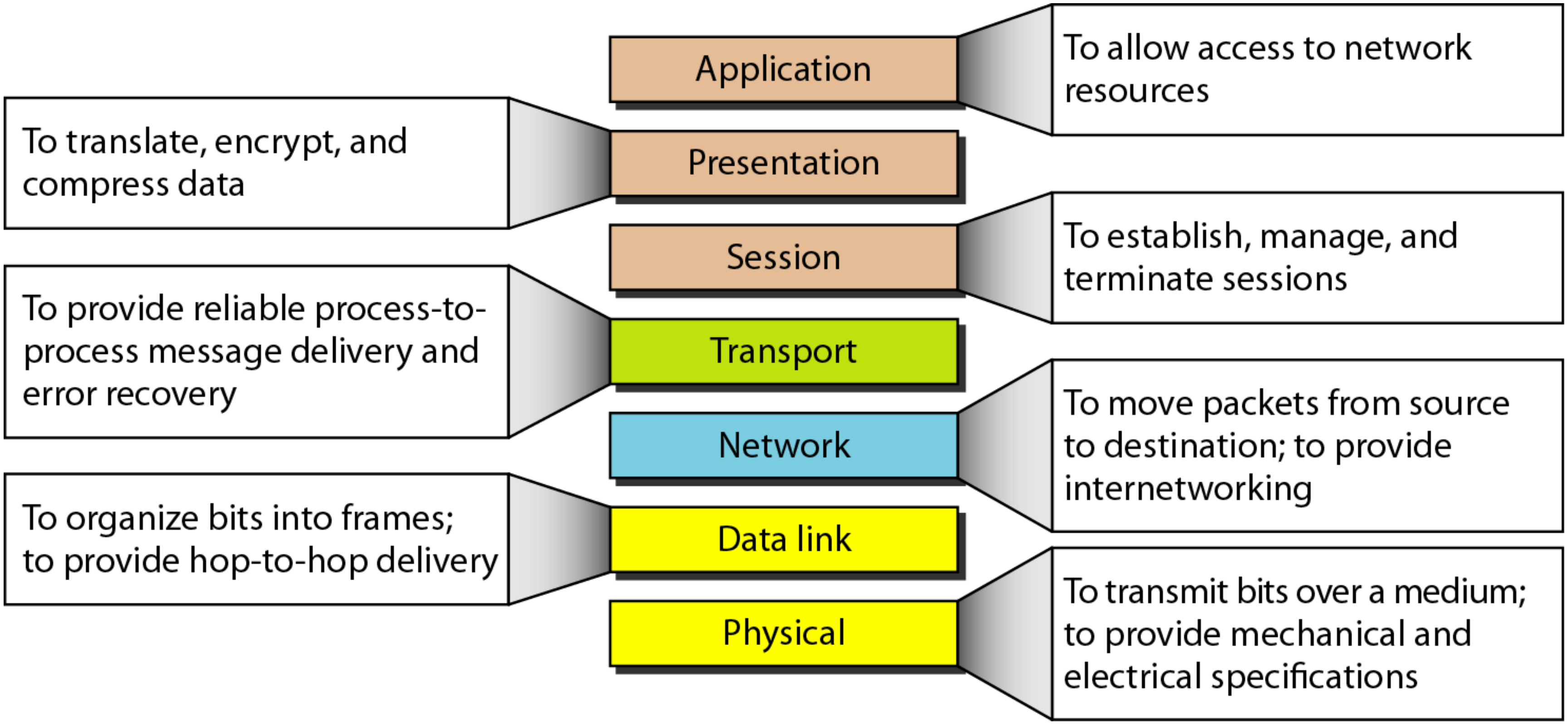
The interaction between layers in the OSI model



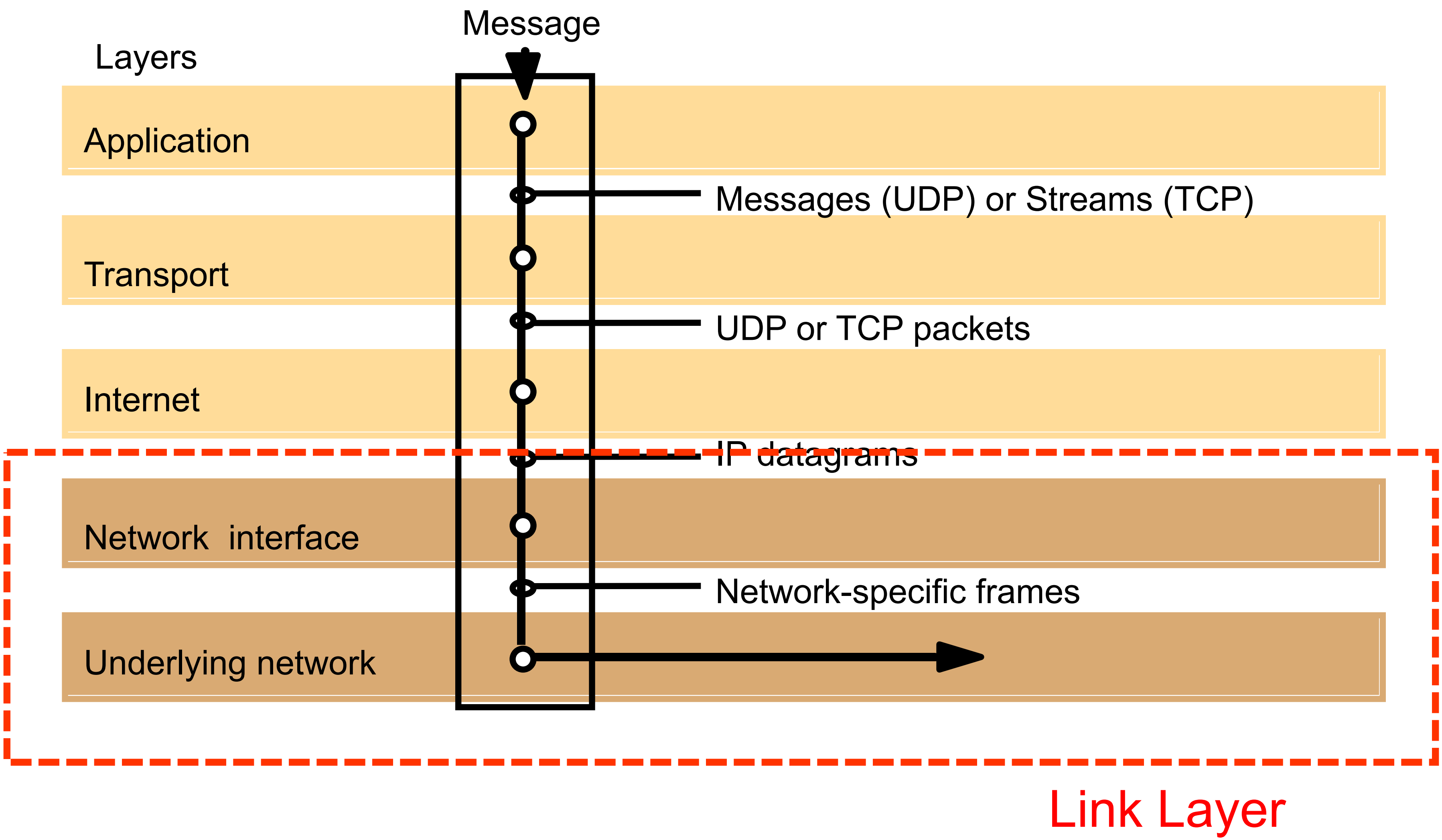
An exchange using the OSI model



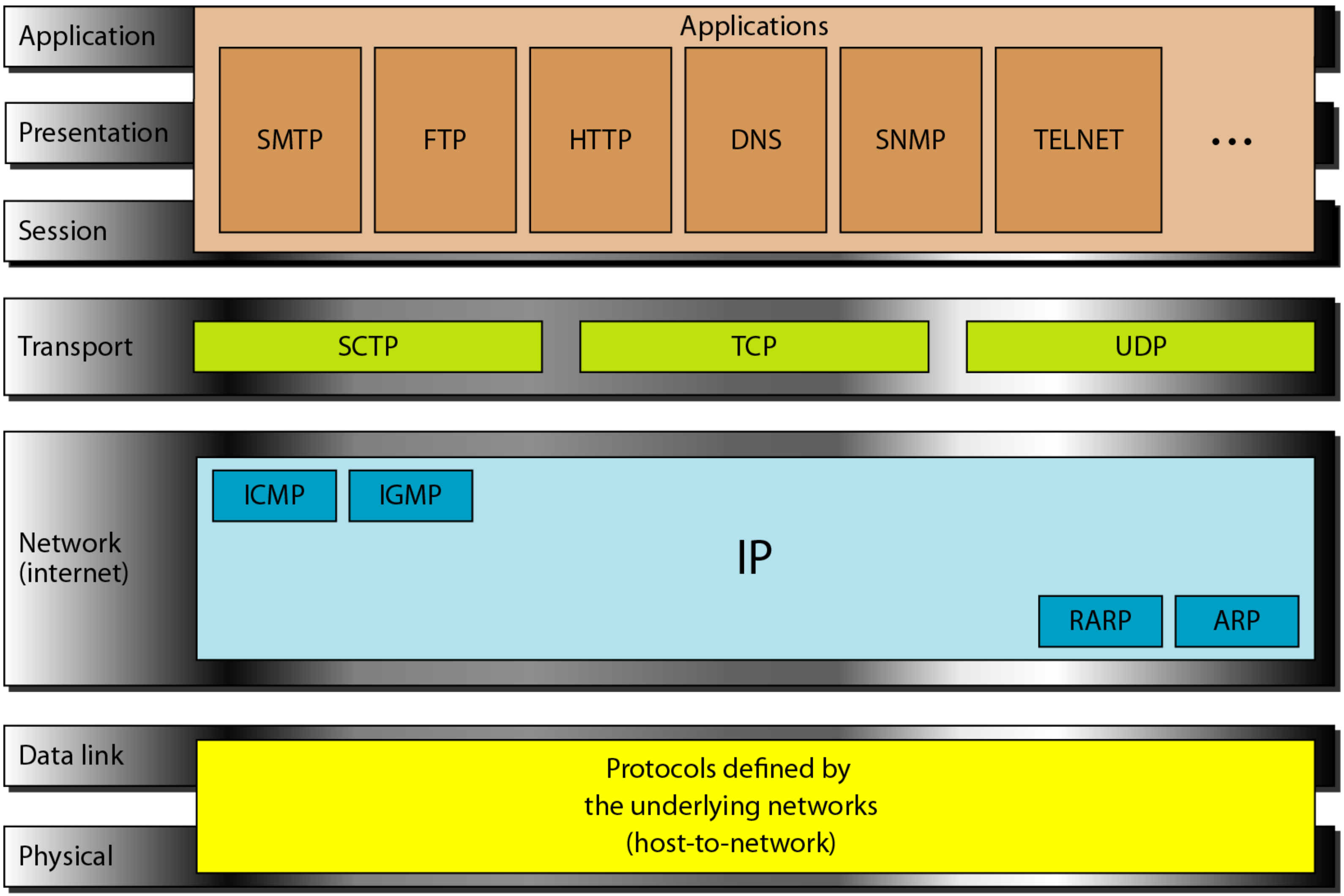
Summary of layers



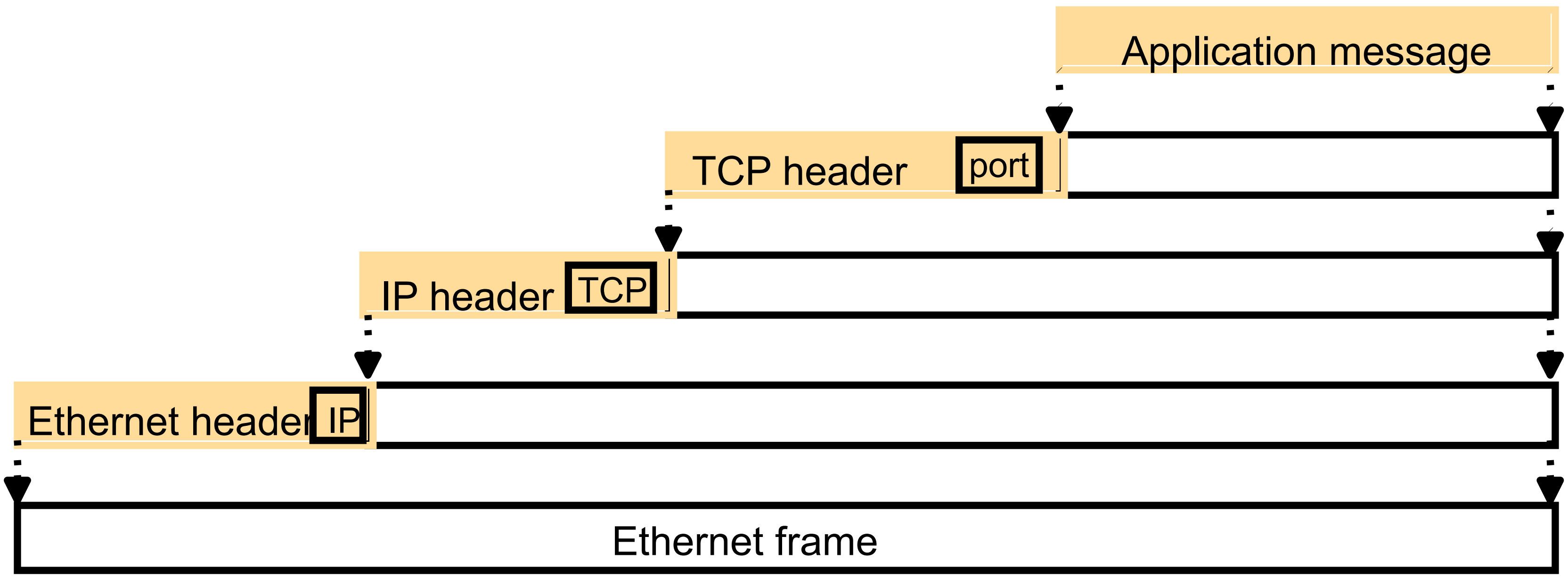
TCP/IP layers



TCP/IP and OSI model

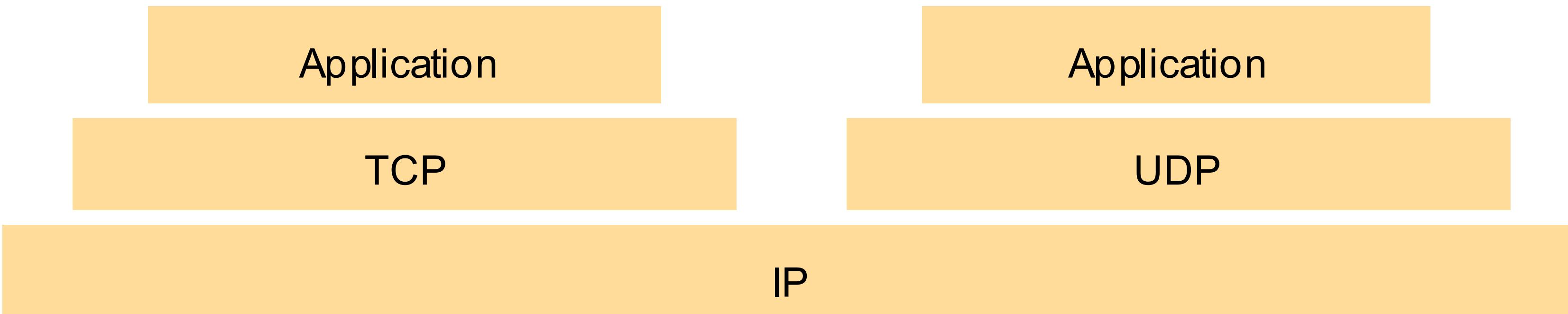


Encapsulation in a message transmitted via TCP over an Ethernet

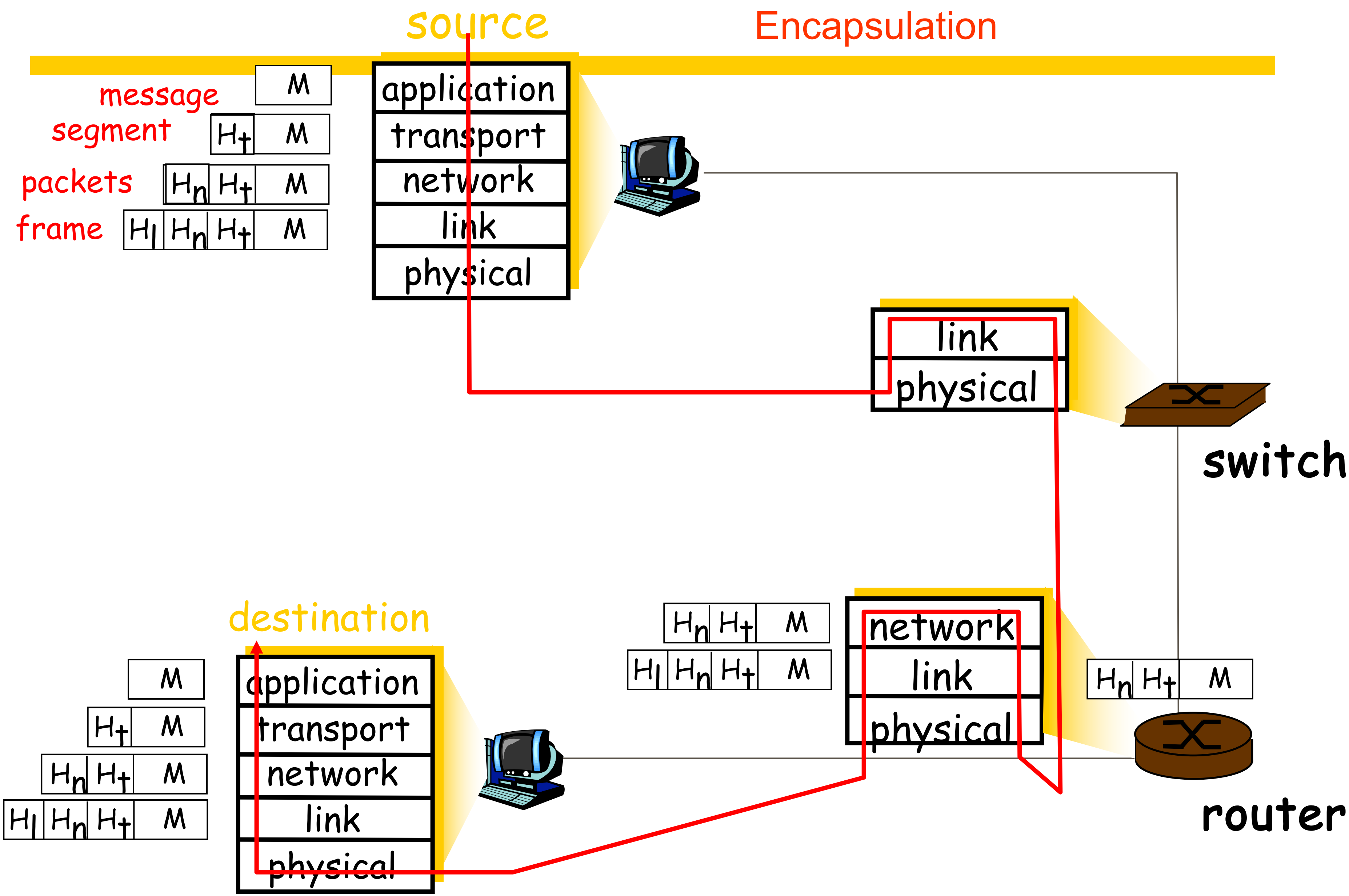


https://www.youtube.com/watch?v=OTwp3xtd4dg

The programmer's conceptual view of a TCP/IP Internet



Encapsulation



From Computer Networking by Kurose and Ross.

OSI vs TCP/IP

OSI Model	TCP/IP model
It is developed by ISO (International Standard Organization)	It is developed by ARPANET (Advanced Research Project Agency Network).
OSI model provides a clear distinction between interfaces, services, and protocols.	TCP/IP doesn't have any clear distinguishing points between services, interfaces, and protocols.
OSI refers to Open Systems Interconnection.	TCP refers to Transmission Control Protocol.
OSI uses the network layer to define routing standards and protocols.	TCP/IP uses only the Internet layer.
OSI follows a vertical approach.	TCP/IP follows a horizontal approach.
OSI use two separate layers physical and data link to define the functionality of the bottom layers.	TCP/IP uses only one layer (link).
OSI layers have seven layers.	TCP/IP has five layers.
OSI model, the transport layer is only connection-oriented.	A layer of the TCP/IP model is both connection-oriented and connectionless.
In the OSI model, the data link layer and physical are separate layers.	In TCP, physical and data link are both combined as a single host-to-network layer.
Session and presentation layers are not a part of the TCP model.	There is no session and presentation layer in TCP model.
It is defined after the advent of the Internet.	It is defined before the advent of the internet.
The minimum size of the OSI header is 5 bytes.	Minimum header size is 20 bytes.

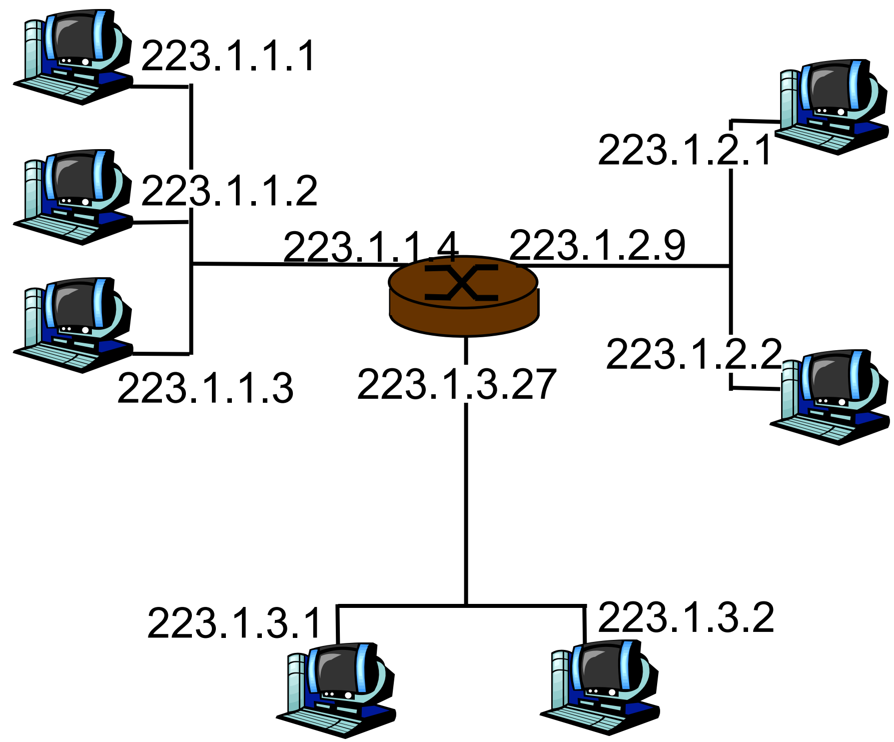
Why layering?

- Explicit structure allows identification, relationship of complex system's pieces
- Each layer
  - gets a service from the one below,
  - performs a specific task, and
  - provides a service to the one above
- Modularization eases maintenance and updating of system
  - We can change the implementation of a layer without affecting the rest of the system as long as the interfaces between the layer are kept the same!
- In some cases, layering considered harmful! Why?

Routing (IP)

IP Addressing: introduction

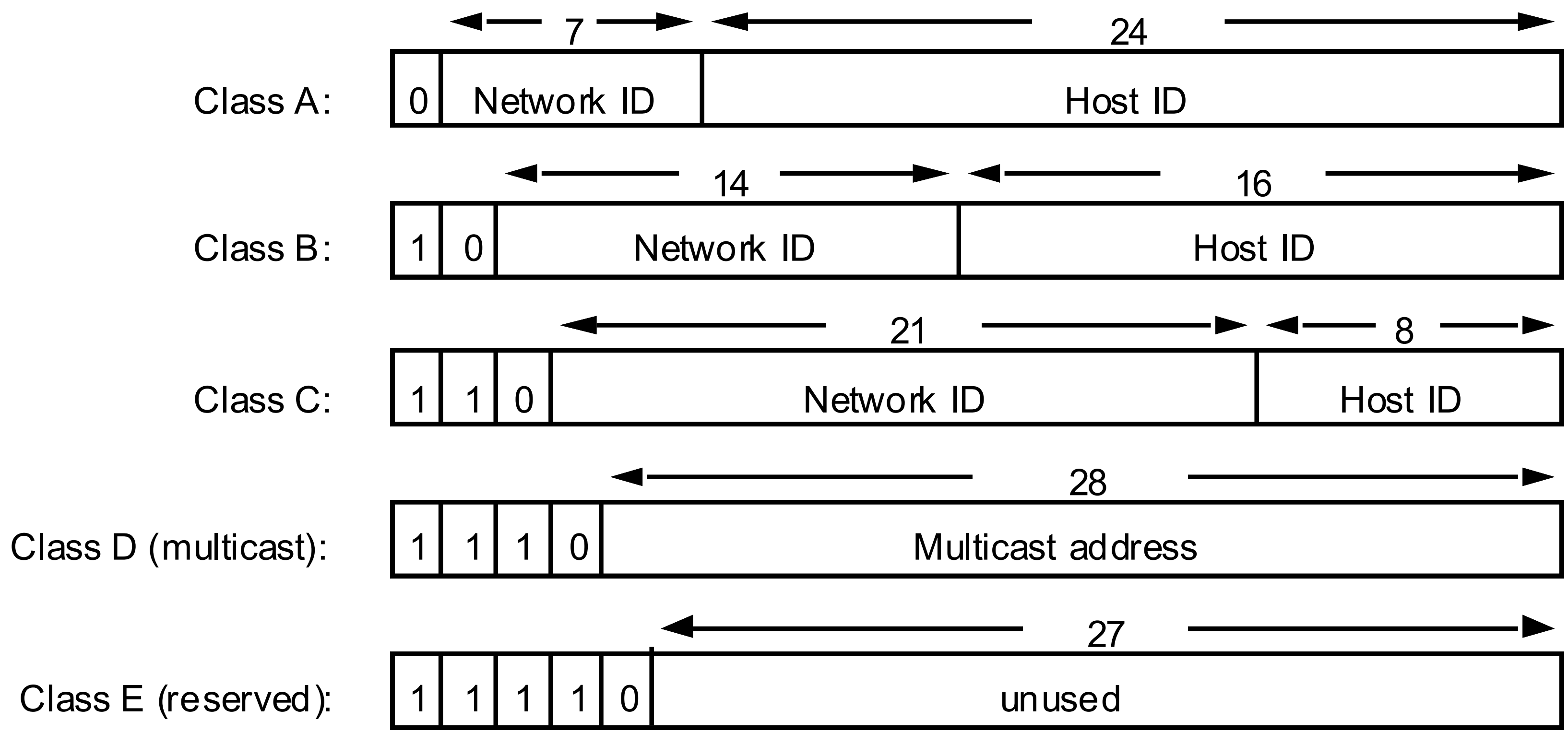
- IP address: 32-bit unique identifier for host, router *interface*
- *interface:* connection between host/router and physical link
  - router's typically have multiple interfaces
  - host typically has one interface
  - IP addresses associated with each interface



$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$

From Computer Networking by Kurose and Ross.

Internet address structure, showing field sizes in bits



Decimal representation of Internet addresses

	octet 1	octet 2	octet 3		Range of addresses
	Network ID	Host ID			
Class A:	1 to 127	0 to 255	0 to 255	0 to 255	1.0.0.0 to 127.255.255.255
	Network ID		Host ID		
Class B:	128 to 191	0 to 255	0 to 255	0 to 255	128.0.0.0 to 191.255.255.255
	Network ID			Host ID	
Class C:	192 to 223	0 to 255	0 to 255	1 to 254	192.0.0.0 to 223.255.255.255
	Multicast address				
Class D (multicast):	224 to 239	0 to 255	0 to 255	1 to 254	224.0.0.0 to 239.255.255.255
Class E (reserved):	240 to 255	0 to 255	0 to 255	1 to 254	240.0.0.0 to 255.255.255.255

IPv4 Packet Header

Version (4 bits)	Header length (4 bits)	Priority and Type of Service (8 bits)	Total length (16 bits)
Identification (16 bits)		Flags (3 bits)	Fragmented offset (13 bits)
Time to live (8 bits)	Protocol (8 bits)	Header checksum (16 bits)	
Source IP address (32 bits)			
Destination IP address (32 bits)			
Options (up to 32 bits)			

- **Version** (always set to the value 4 for IPv4)
- **IP Header Length** (number of 32 -bit words forming the header, usually five)
- **Size of Datagram** (in bytes, header + data)
- **Flags** 3 bits: R (reserved bit set to 0) DF (Don't fragment) MF (More fragments)
- **Time To Live** (Number of hops /links which the packet may be routed over, decremented by most routers - used to prevent accidental routing loops)
- **Protocol** (the type of transport packet being carried (e.g. 1 = ICMP; 6 = TCP; 17= UDP).
- ***Header* Checksum** (A 1's complement checksum of IP header, updated whenever the packet header is modified by a node. Packets with an invalid checksum are discarded by all nodes in an IP network)
- **Source Address / Destination Address**

Example of IP Packet

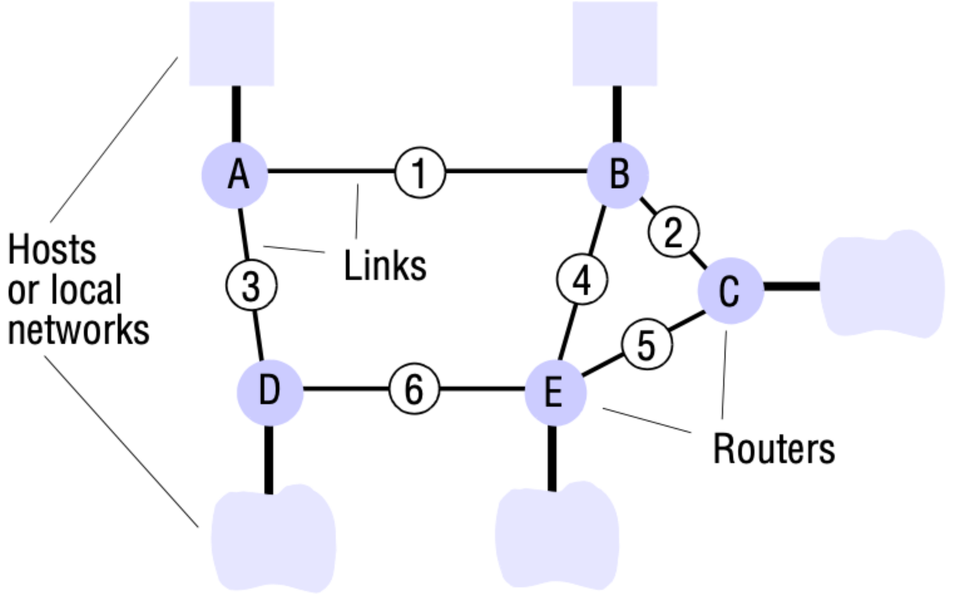
```
⊟ Internet Protocol Version 4, Src: 192.168.82.147 (192.168.82.147), Dst: 192.243.232.2 (192.243.232.2)
    Version: 4
    Header Length: 20 bytes
  ⊟ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
      0000 00.. = Differentiated Services Codepoint: Default (0x00)
      .... ..00 = Explicit Congestion Notification: Not-ECT (Not ECN-Capable Transport) (0x00)
    Total Length: 1155
    Identification: 0x69de (27102)
  ⊟ Flags: 0x02 (Don't Fragment)
      0... .... = Reserved bit: Not set
      .1.. .... = Don't fragment: Set
      ..0. .... = More fragments: Not set
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (6)
  ⊟ Header checksum: 0xd064 [validation disabled]
      [Good: False]
      [Bad: False]
    Source: 192.168.82.147 (192.168.82.147)
    Destination: 192.243.232.2 (192.243.232.2)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
⊞ Transmission Control Protocol, Src Port: 57487 (57487), Dst Port: 80 (80), Seq: 1102, Ack: 883, Len: 1115
```

Datagram Forwarding Strategy

- Every datagram packet contains destination's address
- if connected to destination network, then forward to the host in LAN
  - If network number of destination IP == my network number
- if not directly connected, then forward to the host's default router
- Each router maintains a forwarding table
  - forwarding table maps **network number** (rather than host address) into next hop or interface number (if directly connected)

An Example of Routing

- A packet is submitted to Router A and destination is C, how to routing?



Routings from A		
To	*Link*	*Cost*
A	local	0
B	1	1
C	1	2
D	3	1
E	1	2

Routings from B		
To	*Link*	*Cost*
A	1	1
B	local	0
C	2	1
D	1	2
E	4	1

Routings from C		
To	*Link*	*Cost*
A	2	2
B	2	1
C	local	0
D	5	2
E	5	1

Addressing and Routing

- Address: byte-string that identifies a node
  - usually unique
- Routing: process of forwarding messages to the destination node based on its address
- Types of addresses
  - unicast: node-specific
  - broadcast: all nodes on the network
  - multicast: some subset of nodes on the network

https://www.youtube.com/watch?v=gQtgtKtvRdo

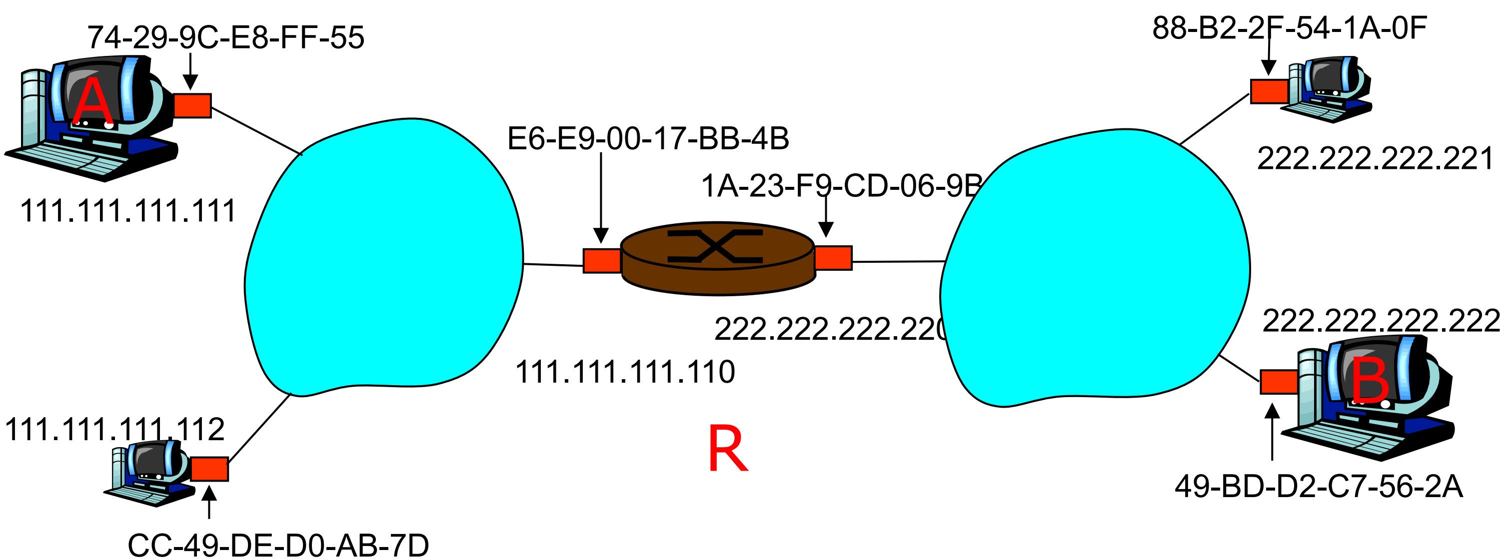
Address Translation in LAN

- Map IP addresses into physical addresses of the destination host (if connected directly) or the next hop router
- ARP (Address Resolution Protocol)
  - Each host caches its table of IP to physical address bindings
  - table entries are discarded if not refreshed
    - timeout in about 10 minutes
  - broadcast request if IP address not in table
  - target machine send its physical address to the sender
  - target machine also updates add entry of the source in its table
    - It is likely that the target will send IP packets to the source later on.
  - Other hosts (who receives the broadcasted request) update table if already have an entry

Addressing: routing to another LAN

walkthrough: send datagram from A to B via R

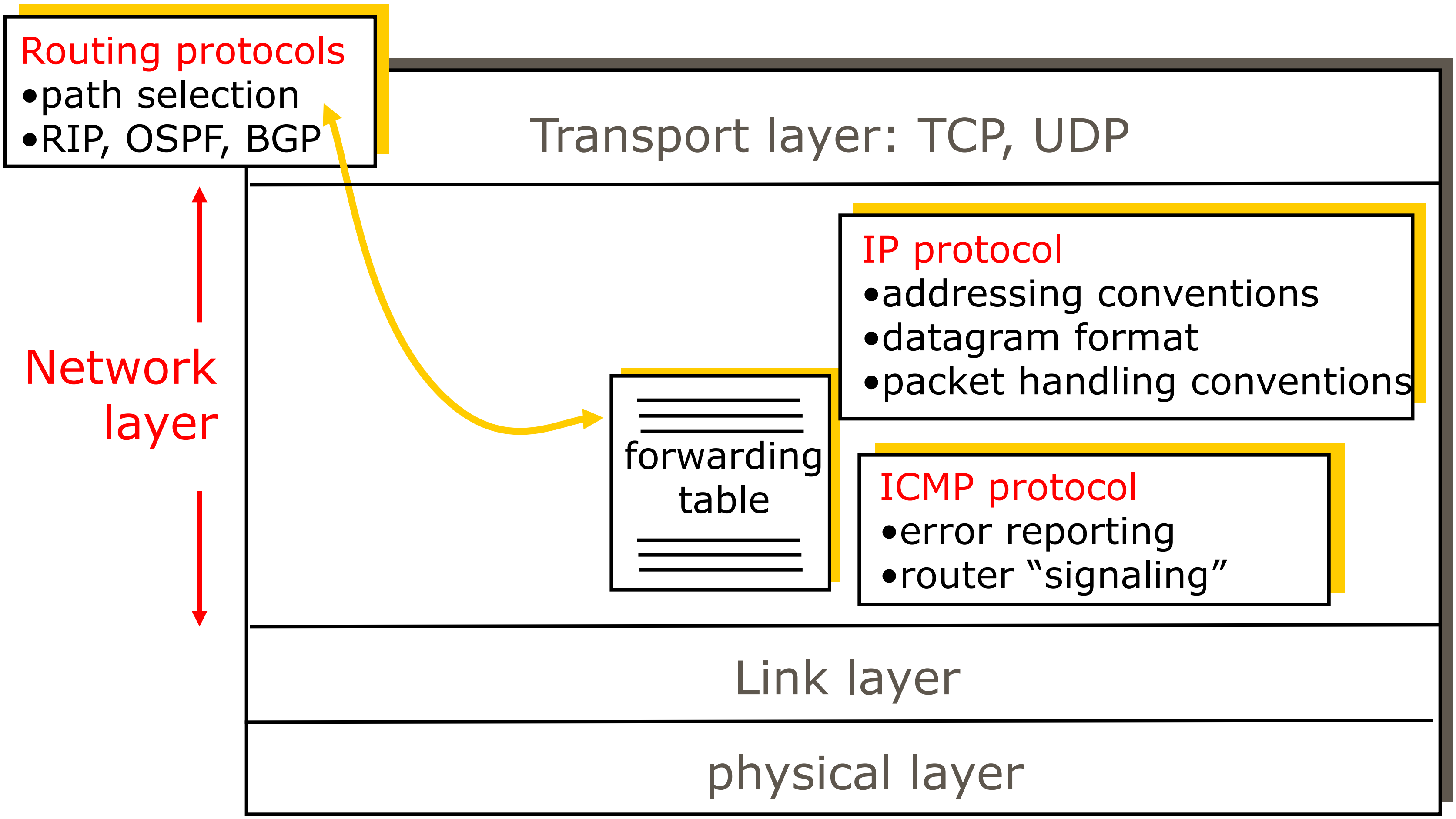
assume A knows B's IP address



- two ARP tables in router R, one for each IP network (LAN)

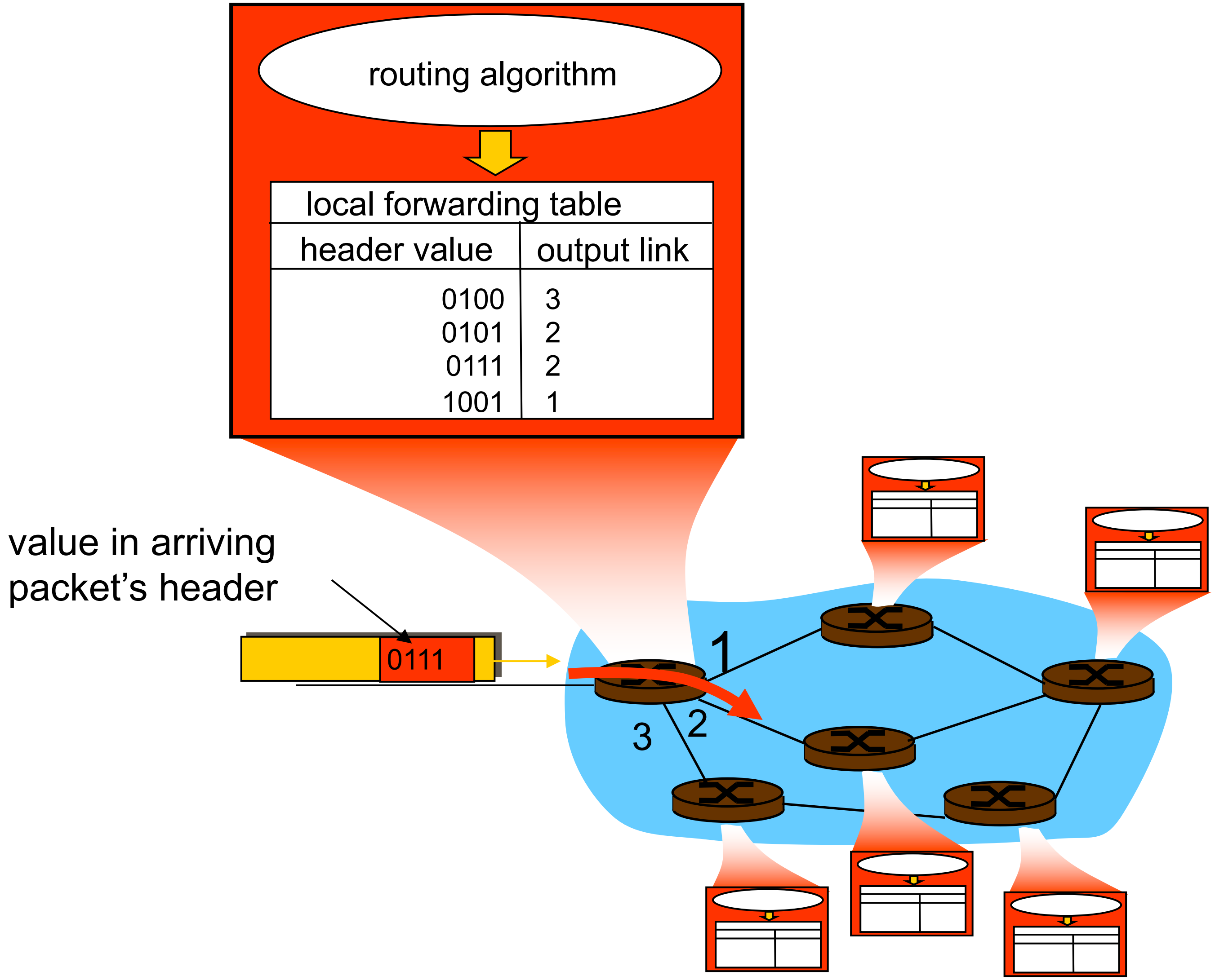
Network layer

Host, router network layer functions:



From Computer Networking by Kurose and Ross.

Forwarding Problem: Where to Send Next?



From Computer Networking by Kurose and Ross.

Only IPv4? No, we need more!

https://www.youtube.com/watch?v=bNmnRvZW3HU

IPv6

Version (4 bits)	Traffic class (8 bits)	Flow label (20 bits)
Payload length (16 bits)	Next header (8 bits)	Hop limit (8 bits)
Source address (128 bits)		
Destination address (128 bits)		

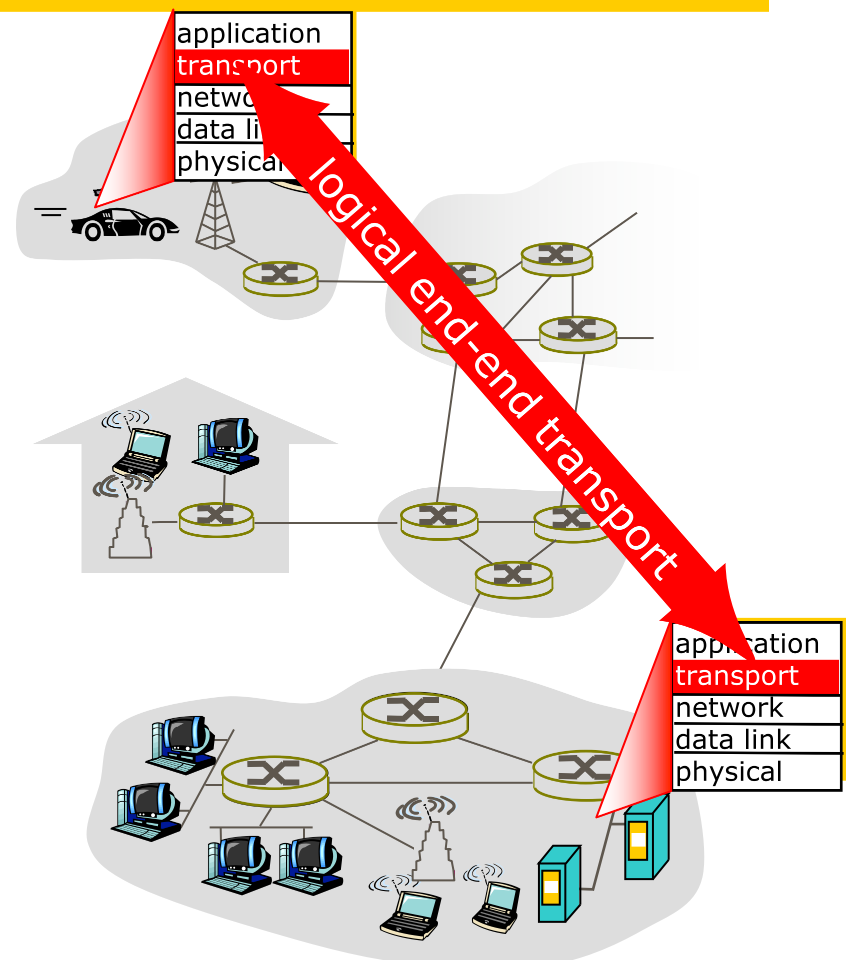
End-to-End Protocols
(UDP/TCP)

End-to-End Protocols

- Underlying best-effort network
  - drop messages
  - re-orders messages
  - delivers duplicate copies of a given message
  - limits packet (not message) to some finite size
  - delivers messages after an arbitrarily long delay
- Common end-to-end services
  - guarantee message delivery
  - deliver messages in the same order they are sent
  - deliver at most one copy of each message
  - support arbitrarily large messages
  - support synchronization between sender and receiver
  - allow the receiver to flow control the sender
  - support multiple application processes on each host

Transport Layer

- provide *logical communication* between app processes running on different hosts
- transport protocols run in end systems
  - sender side: breaks app messages into segments, passes to network layer
  - receiver side: reassembles segments into messages, passes to app layer
- more than one transport protocol available to apps
  - Internet: TCP and UDP



From Computer Networking by Kurose and Ross.

Internet transport protocols services

Transmission Control Protocol (TCP) service:

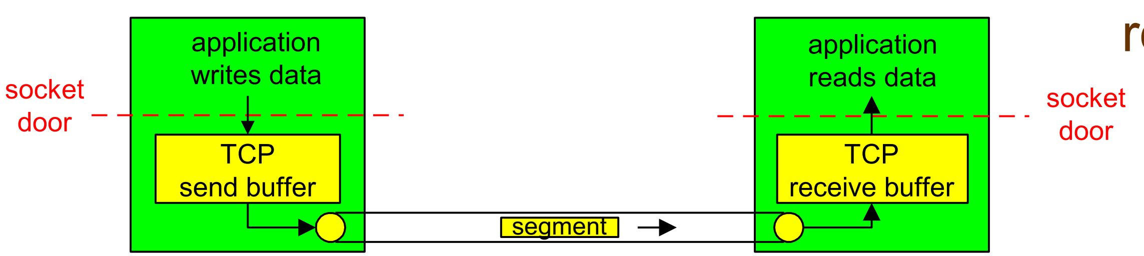
- *connection-oriented:* setup required between client and server processes
- *reliable transport* between sending and receiving process
- *flow control:* sender won't overwhelm receiver
- *does not provide:* timing, minimum throughput guarantees, security

User Datagram Protocol (UDP) service:

- unreliable data transfer between sending and receiving process
- does not provide: connection setup, reliability, flow control, congestion control, timing, throughput guarantee, or security

TCP: Overview

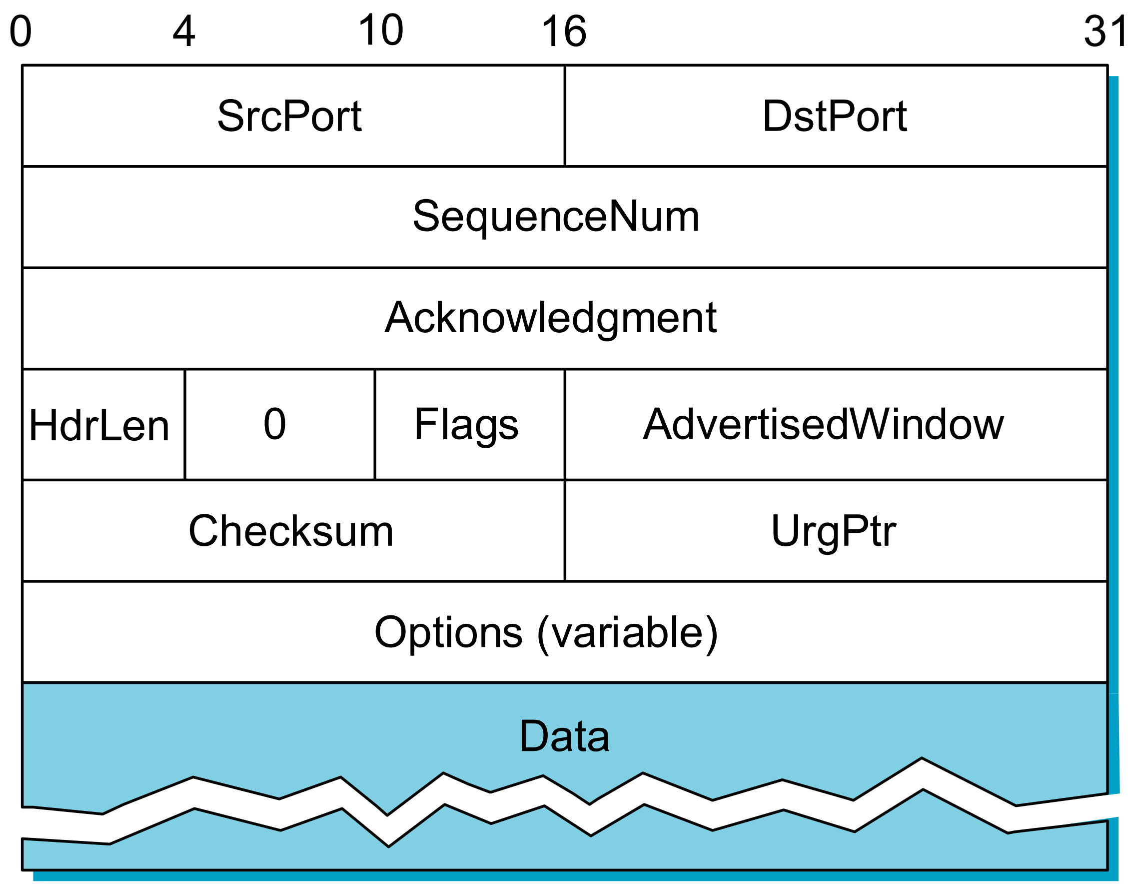
- point-to-point
  - one sender, one receiver
- reliable, in-order *byte steam*
- Pipelined
- *send & receive buffers*



- full duplex data:
  - bi-directional data flow in same connection
- connection-oriented:
  - handshaking (exchange of control msgs)
- flow controlled:
  - sender will not overwhelm receiver

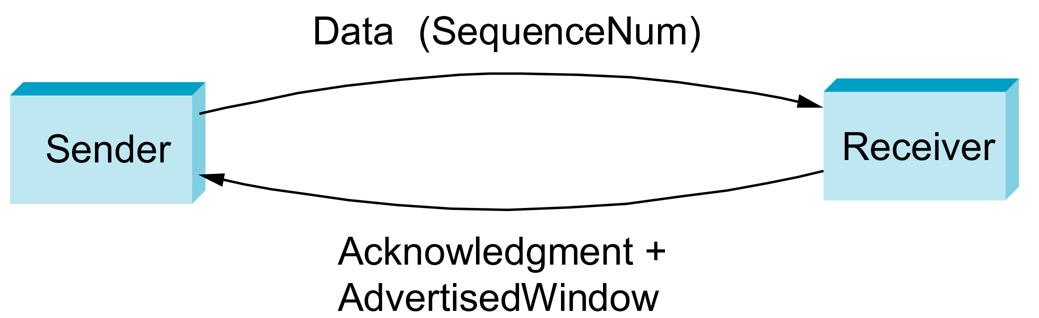
From Computer Networking by Kurose and Ross.

TCP Segment Format



Segment Format (cont)

- Each connection identified with 4-tuple:
  - **`(SrcPort, SrcIPAddr, DsrPort, DstIPAddr)`**
- Sliding window + flow control
  - **`acknowledgment, SequenceNum, AdvertisedWinow`**



- Flags
  - **`SYN, FIN, RESET, PUSH, URG, ACK`**
- Checksum
  - pseudo header + TCP header + data

TCP Connection Management

Recall: TCP sender, receiver establish "connection" before exchanging data segments

- initialize TCP variables:
  - seq. #s
  - buffers, flow control info
- *client:* connection initiator
- *server:* contacted by client

Three way handshake:

Step 1: client host sends TCP SYN segment to server
- specifies initial seq #
- no data

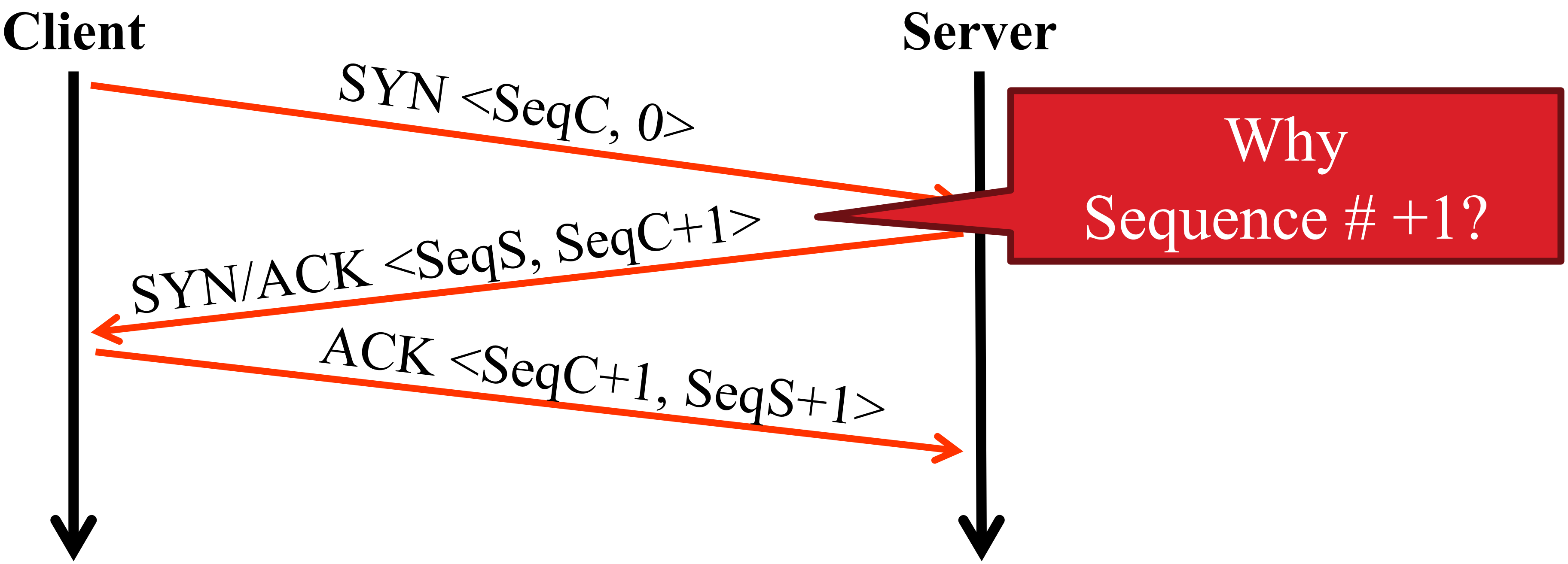
Step 2: server host receives SYN, replies with SYNACK segment
- specifies server initial seq. #

Step 3: client receives SYNACK, replies with ACK segment, which may contain data

Connection Setup

- Why do we need connection setup?
  - To establish state on both hosts
  - Most important state: sequence numbers
    - Count the number of bytes that have been sent
    - Initial value chosen at random
    - Why?
- Important TCP flags (1 bit each)
  - SYN – synchronization, used for connection setup
  - ACK – acknowledge received data
  - FIN – finish, used to tear down connection

Three Way Handshake



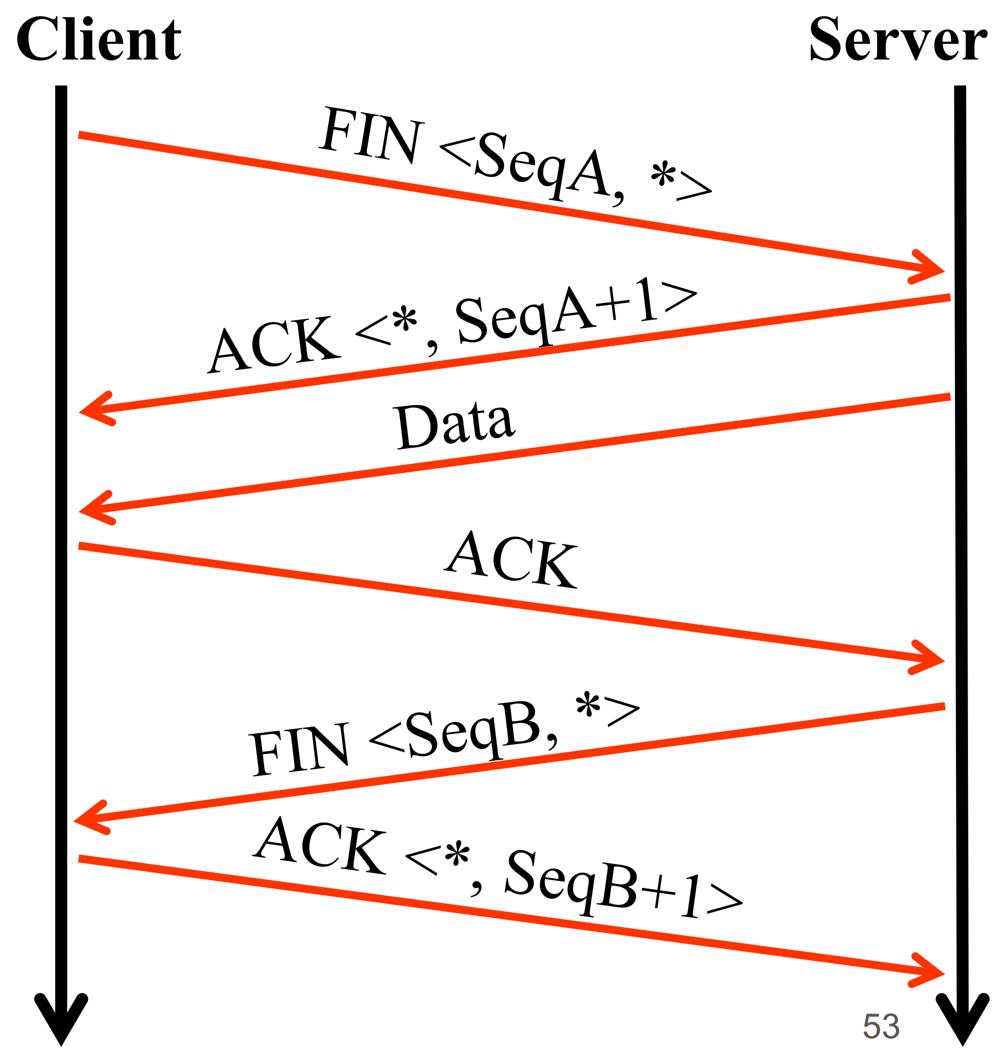
- Each side:
  - Notifies the other of starting sequence number
  - ACKs the other side's starting sequence number

Connection Setup Issues

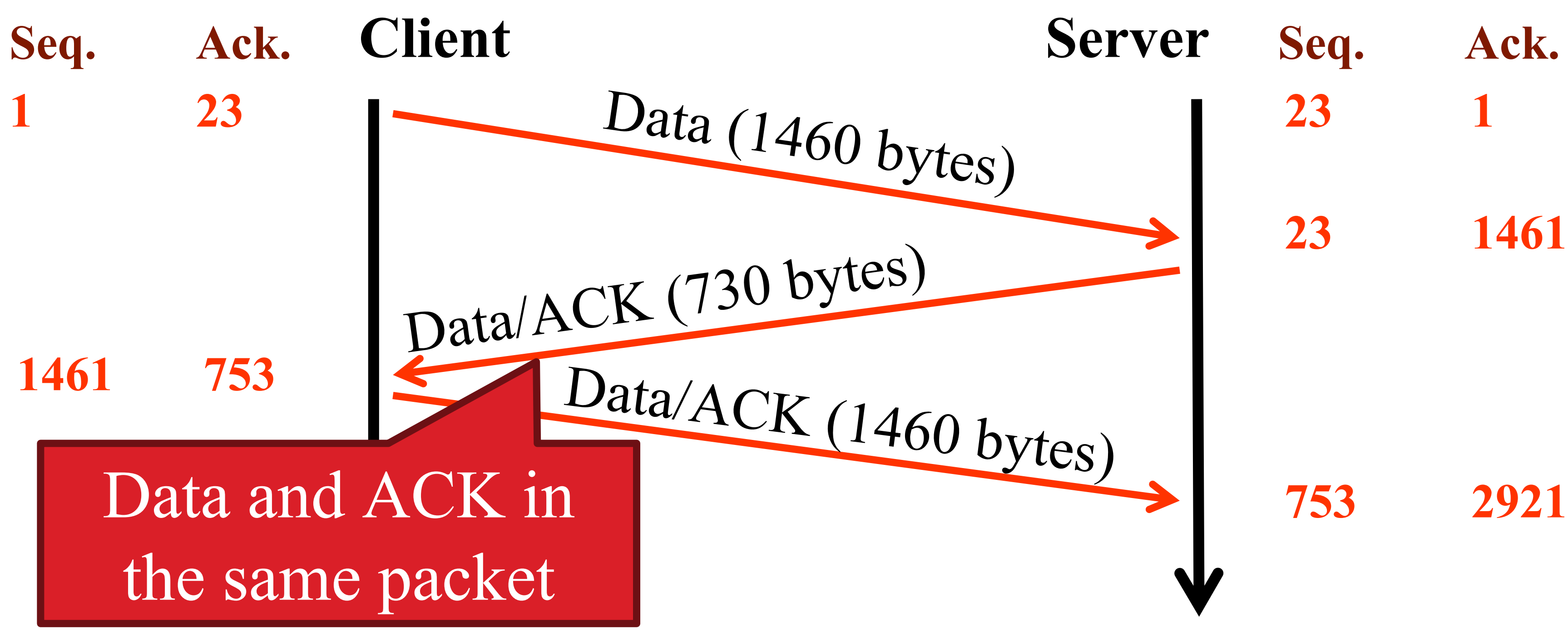
- Connection confusion
  - How to disambiguate connections from the same host?
  - Random sequence numbers
- Source spoofing
  - Need good random number generators!

Connection Tear Down

- Either side can initiate tear down
- Other side may continue sending data
  - Half open connection
  - *shutdown()*
- Acknowledge the last FIN
  - Sequence number + 1
- What happens if 2nd FIN is lost?



Bidirectional Communication



- Each side of the connection can send and receive
  - Different sequence numbers for each direction

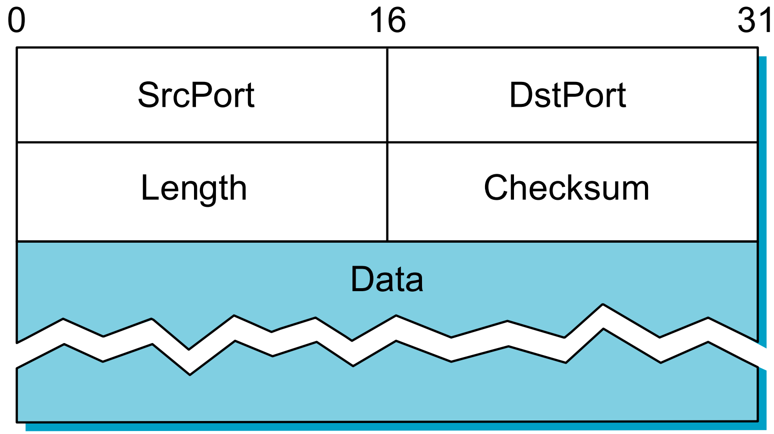
User Datagram Protocol (UDP)

0	16	31
Source Port	Destination Port	
Message Length	Checksum	

- Simple, connectionless datagram
- Port numbers enable demultiplexing
  - 16 bits = 65535 possible ports
  - Port 0 is invalid
- Checksum for error detection
  - Detects (some) corrupt packets
  - Does not detect dropped, duplicated, or reordered packets

Simple Demultiplexor (UDP)

- Unreliable and unordered datagram service
- No flow control or error control
  - no need for sender-side buffer
- Endpoints identified by ports
- Header format



- Optional checksum
  - psuedo header (IP.src, IP.dsest, IP.proto, UDP.len) + UDP header + data

Uses for UDP

- Invented after TCP
  - Why?
- Not all applications can tolerate TCP
- Custom protocols can be built on top of UDP
  - Reliability? Strict ordering?
  - Flow control? Congestion control?
- Examples
  - Live media streaming (e.g. voice, video)
  - Facebook datacenter protocol

TCP vs UDP

https://www.youtube.com/watch?v=cA9ZJdqzOoU