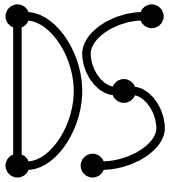


Research Group
Distributed Systems

Distributed Systems

Communication

Olaf Landsiedel

Last Time

- What did we discuss?

Today:
Tour through the protocol stack

- Refresh your networking background
- Make sure we speak the same language
- Focus on the “why”
  - and not so much on the “how”

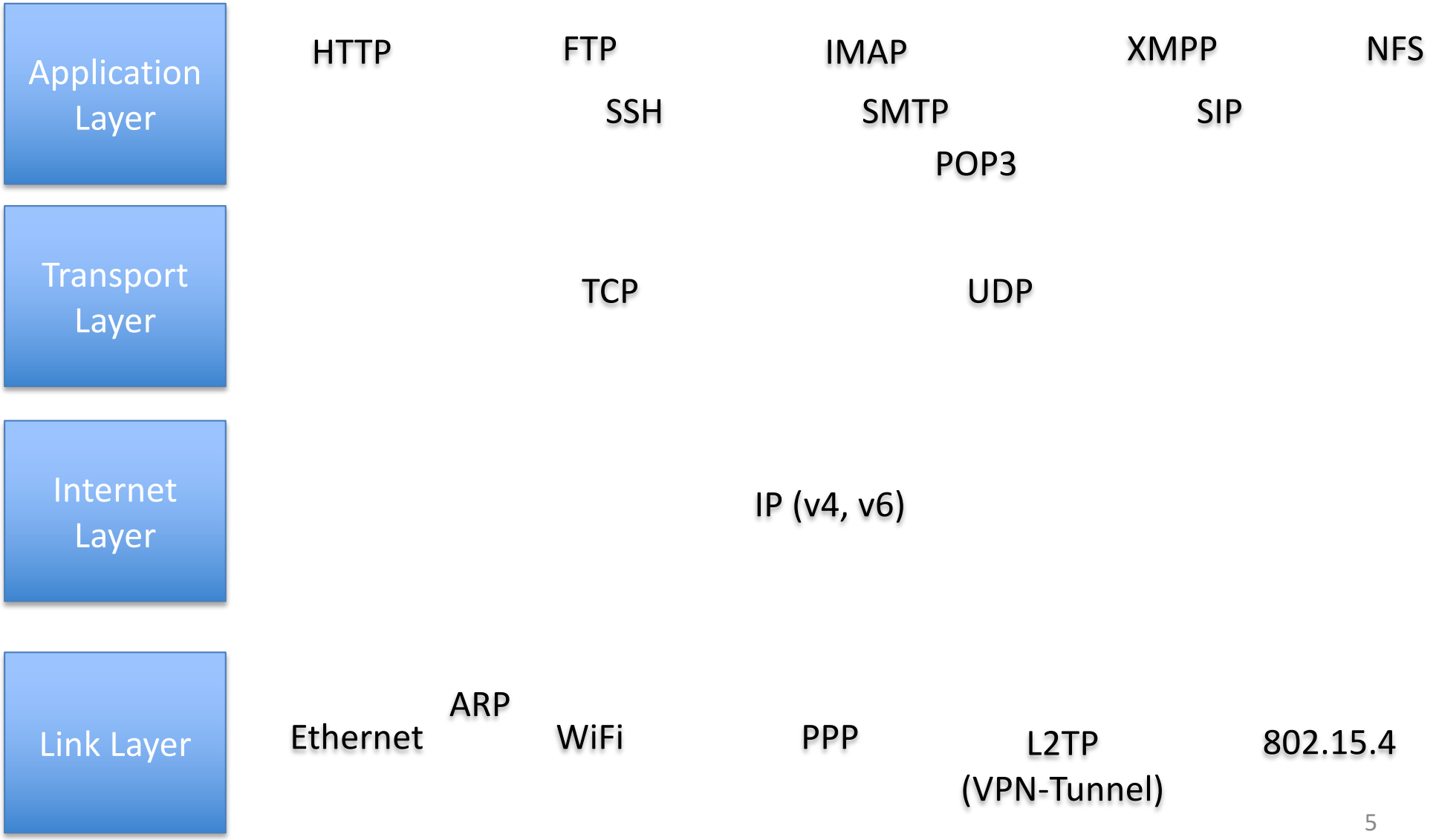
- Note
  - The chapter on communication in the book is large
    - And does not align well with the lecture
    - As this is only a recap lecture on communication

Warm-Up

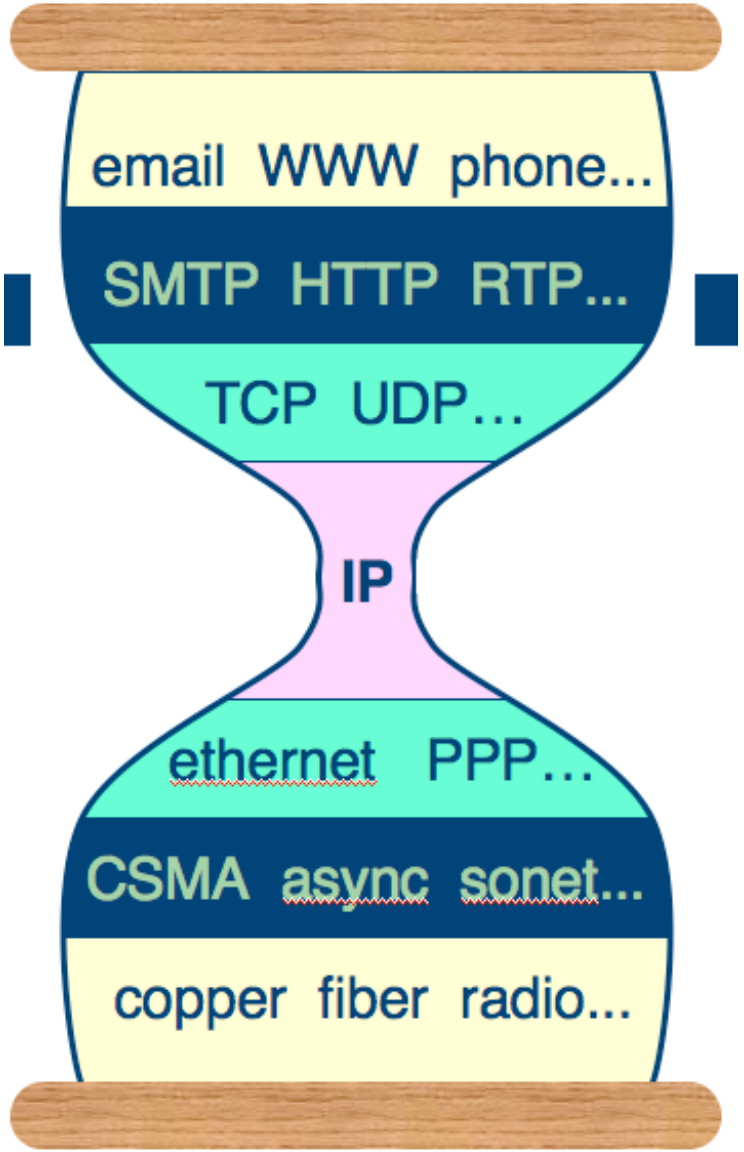
- What Internet Protocols do you know?
  - And what do they do?

- Layers
  - What layers do we have?
  - On which protocol layer does each protocol go?
  - What protocol does a protocol use on the layer below?

Warm-Up


Application Layer
HTTP
FTP
IMAP
XMPP
NFS
SSH
SMTP
SIP
POP3
Transport Layer
TCP
UDP
Internet Layer
IP (v4, v6)
Link Layer
Ethernet
ARP
WiFi
PPP
L2TP
(VPN-Tunnel)
802.15.4

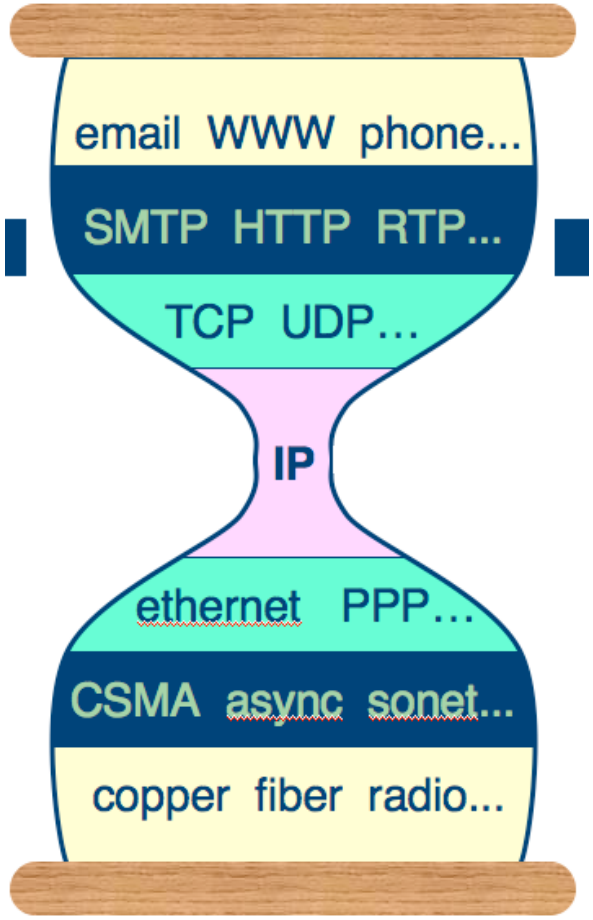
Internet Hourglass



Everything over IP

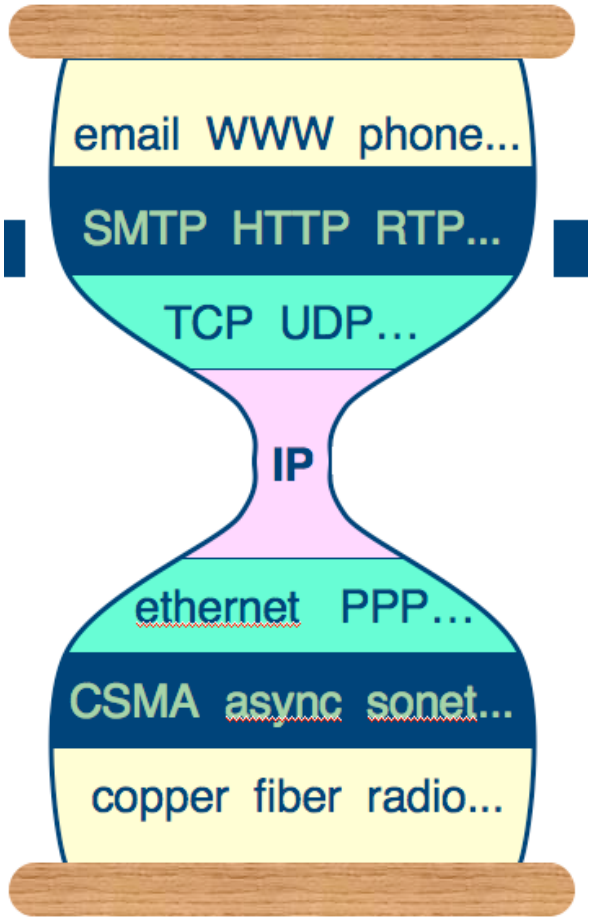
IP over everything

Internet Hourglass



- IP: core element
  - Internet design philosophy
- Why is IP a good choice for this core?
- IP is a simple protocol
  - Single main task
    - Forwarding: try to bring packet from source to destination (end-to-end)
    - Other services (reliability, congestion control, security, …) on layers above (and below)
    - All lower layers: do not provide globally valid addresses
  - Reason for success of the Internet
- But?
  - What happens when we need to change IP?

Example: IPv6 Deployment

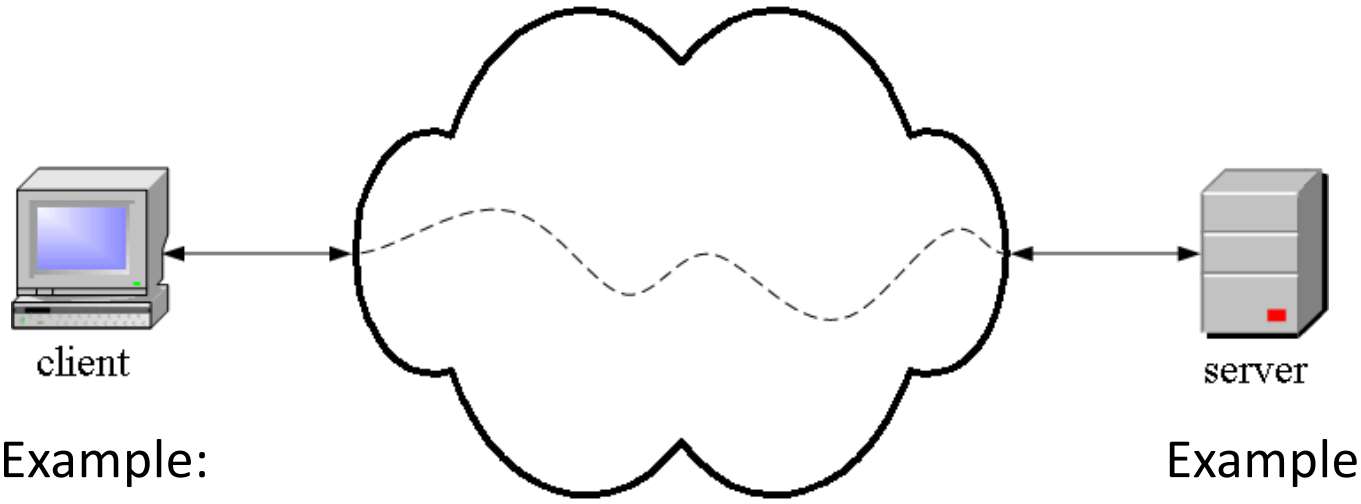


- Challenges for IPv6 deployment?
  - Concerns **all** devices on the Internet
- Approaches?
  - Switch day?
    - Difficult due to size
    - ARPANET: NCP -> TCP/IP protocols
      - January 1, 1983
  - Incremental deployment?
    - Requires translation (IPv4 <> IPv6)
    - Costly
    - Until today: Not enough pressure

Outline

- Warm-Up: Internet Hourglass
- Protocols and OSI Reference Model
- Internet Protocol Stack
- Selected Protocols

Internet from 2,000 feet up



Example:
Web Browser

Internet

Example:
Web server

Nodes exchange “data”

What is in the data?

- Humans: What do we need to understand each other?
  - Same language, same conventions (grammar, …)
- For computers?
  - Electrical encoding of data
  - Where is the start of the packet?
  - Which bits contain the length?
  - Is there a checksum? where is it? how is it computed?
  - What is the format of an address?
  - Byte ordering
  - …

Protocols

- These instructions and conventions are known as **protocols**

Protocols

- Exist at different levels

understand format of address and how to compute checksum

humans vs. whales
different wavelengths

versus

request web page

French vs. Hungarian

Layering

- Network protocols
  - generally organized in layers
- Reasons?
  - Ease software development?
    - Higher-level software does not have to know how to format an Ethernet packet
    - … or even know that Ethernet is being used
  - Maximize flexibility?
    - Replace one layer without replacing surrounding layers
    - But: still difficult on central layers
      - See hourglass model
    - Lower / higher layers: easier to replace

Layering: OSI Reference Model

- Most popular model of guiding (not specifying) protocol layers is
  - **OSI reference model**
  - OSI: Open Systems Interconnection
  - Reference model for communication networks
  - Not specific to the Internet
- Adopted and created by ISO
  - ISO: International Organization for Standardization
- 7 layers of protocols

Note

- In the Warm Up
  - We discussed the layering in the Internet
    - Less layers than on OSI reference model
- We will later discuss this in more detail

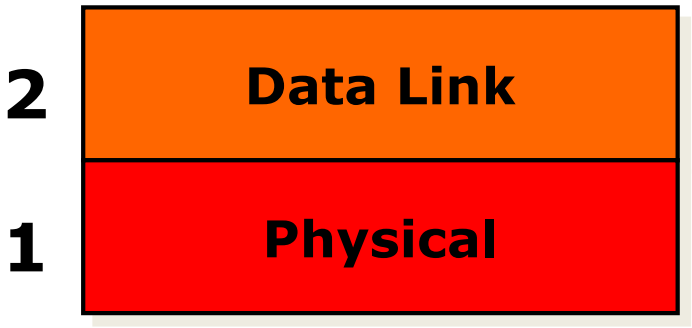
OSI Reference Model: Layer 1

- Transmits and receives raw data to communication medium
- Does not care about contents
- Voltage levels, speed, connectors, modulation, …

Examples: RS-232, 10BaseT

OSI Reference Model: Layer 2

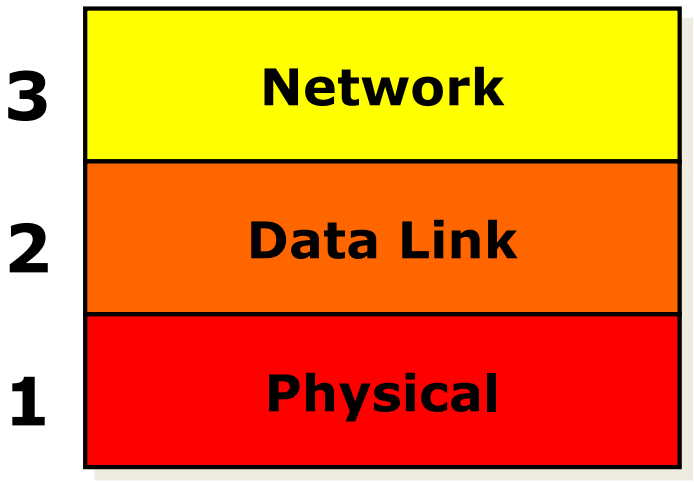
Examples: Ethernet MAC, PPP



- Detect, correct errors
- Organizes data into packets before passing it down
- Sequences packets (if necessary)
- Accepts acknowledgements from receiver

OSI Reference Model: Layer 3

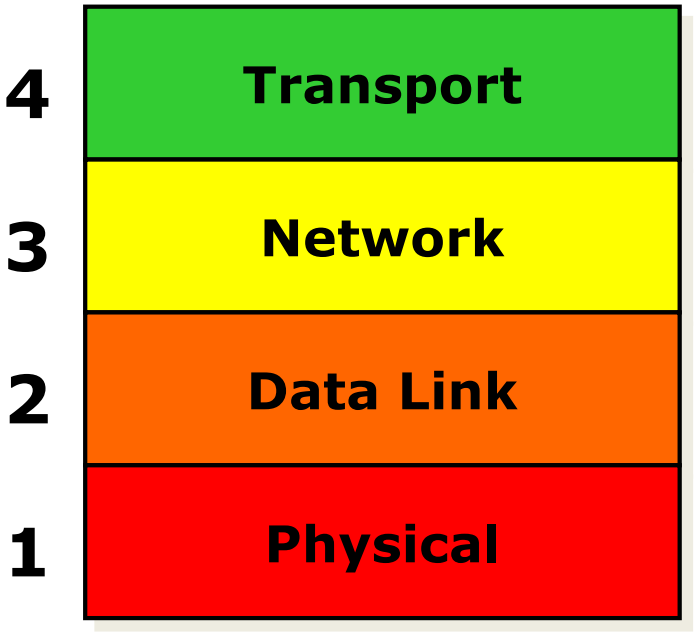
Examples: IP, X.25



- Relay and route information to destination
- Manage journey of packets and figure out intermediate hops (if needed)

OSI Reference Model: Layer 4

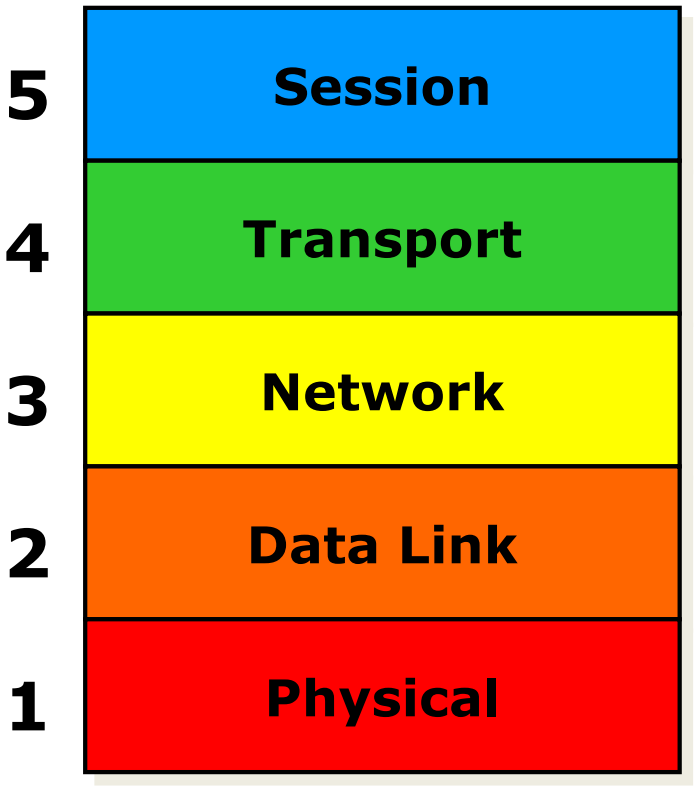
Examples: TCP, UDP



- Provides a consistent interface for end-to-end (application-to-application) communication
- Congestion, flow control
- Network interface is similar to a mailbox
  - "send this to X"

OSI Reference Model: Layer 5

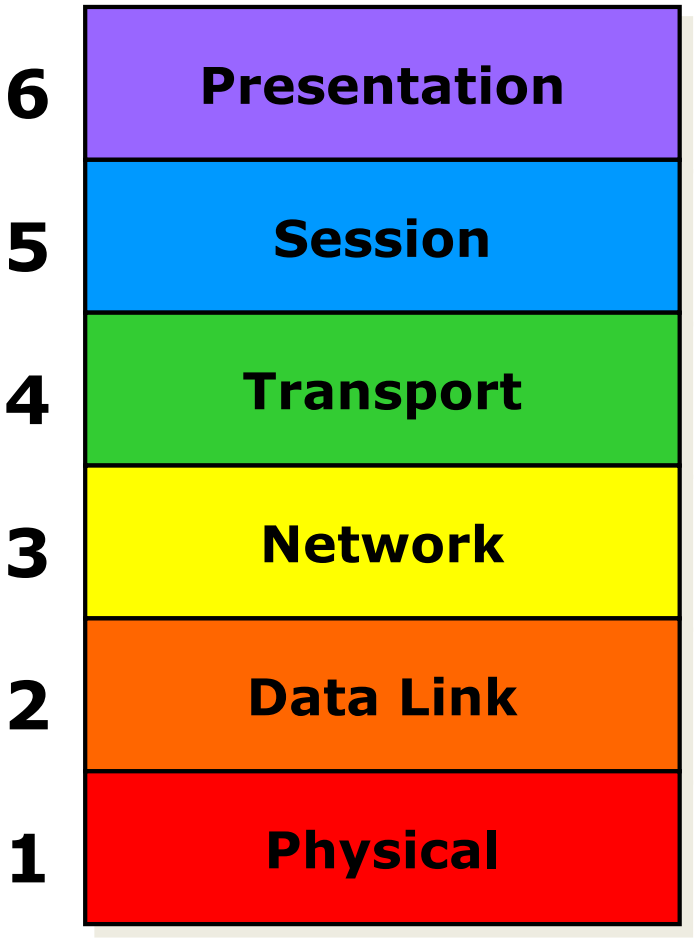
Examples: HTTP 1.1, SSL, NetBIOS



- Services to coordinate dialogue and manage data exchange
- Software implemented switch
- Manage multiple logical connections
- Keep track of who is talking: establish & end communications

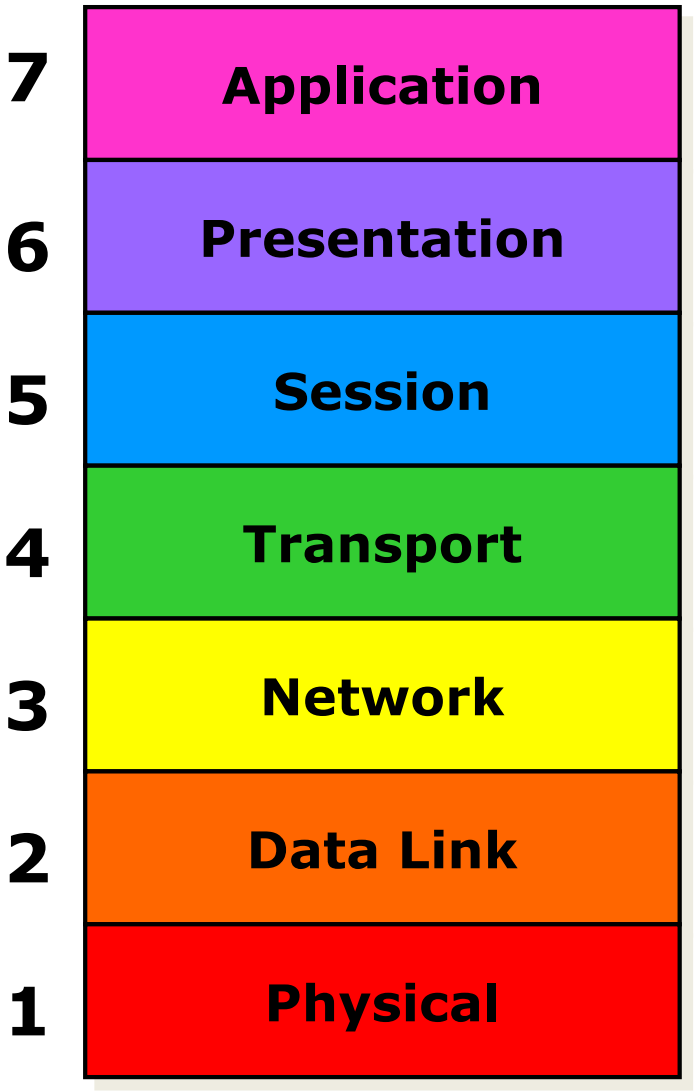
OSI Reference Model: Layer 6

Examples: XDR, ASN.1, MIME, MIDI



- Data representation
- Concerned with the meaning of data bits
- Convert between machine representations

OSI Reference Model: Layer 7



- Collection of application-specific protocols

Examples:
email (SMTP, POP, IMAP)
file transfer (FTP)
directory services (LDAP)

Discussion: OSI Model

- OSI Model: Reference model for
  - Layered communication systems
  - They should follow a **similar** design
- Internet
  - Has a similar structure
  - But: some layers are combined into one
    - Will show you later

Good / Bad Layering

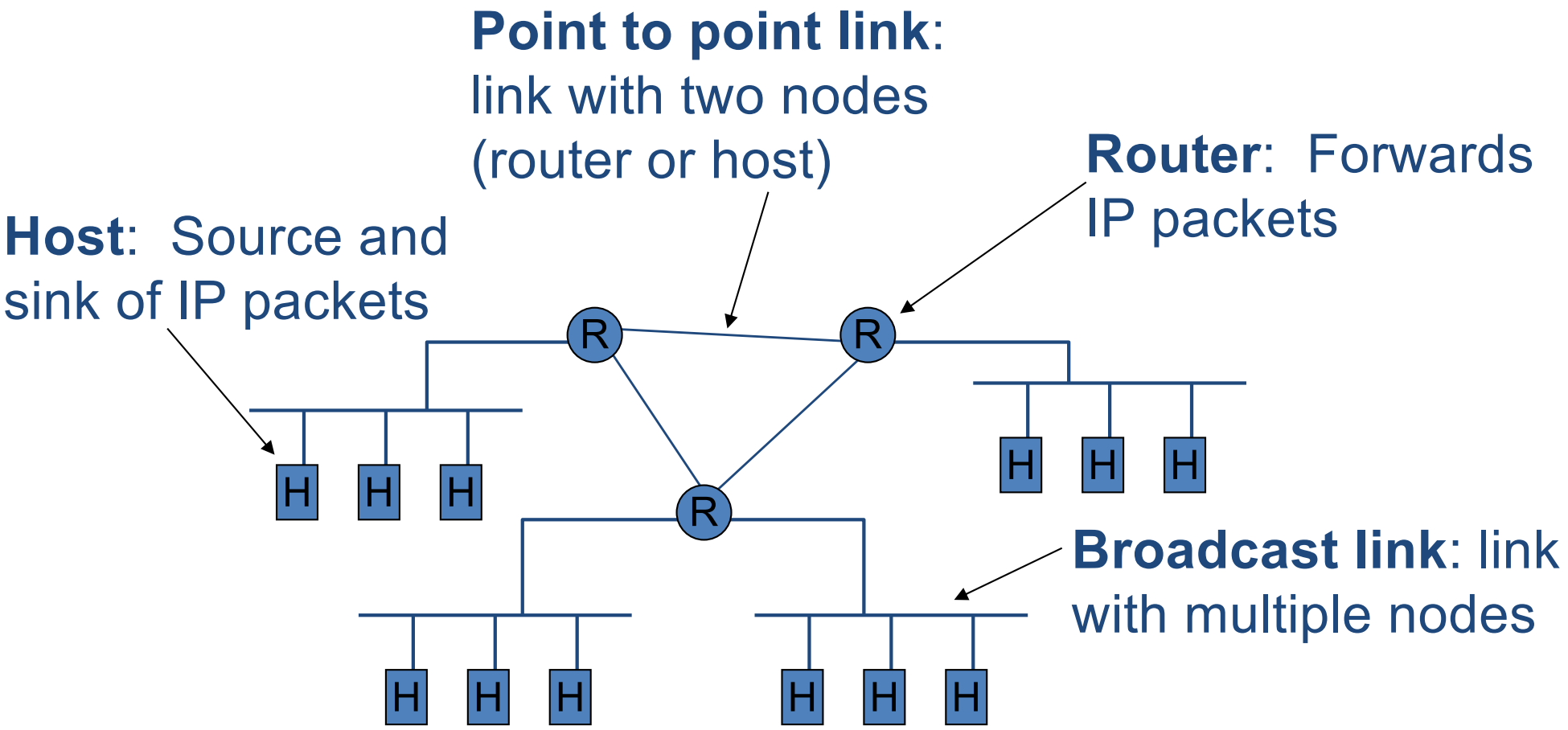
- Assume you have to define a layered architecture
  - How do you distinguish a “good” and “bad” layering?

- Good layering
  - Narrow, simple interfaces between the layers
  - Example: TCP / UDP socket programming
    - Very narrow, simple interfaces: small number of functions

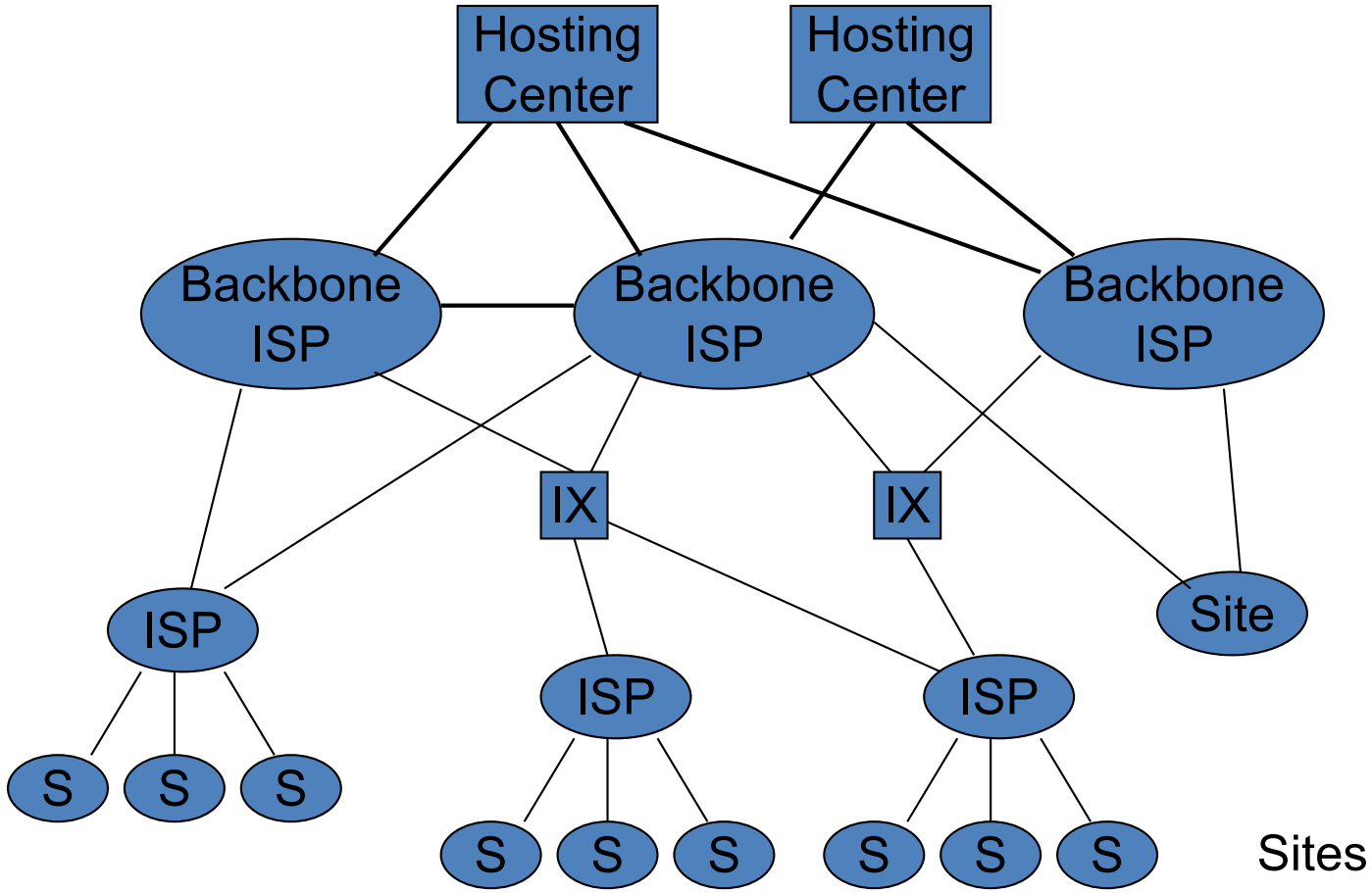
Outline

- Warm-Up: Internet Hourglass
- Protocols and OSI Reference Model
- Internet Protocol Stack
- Selected Protocols

Network Components: Local Network



Network Components: Internet



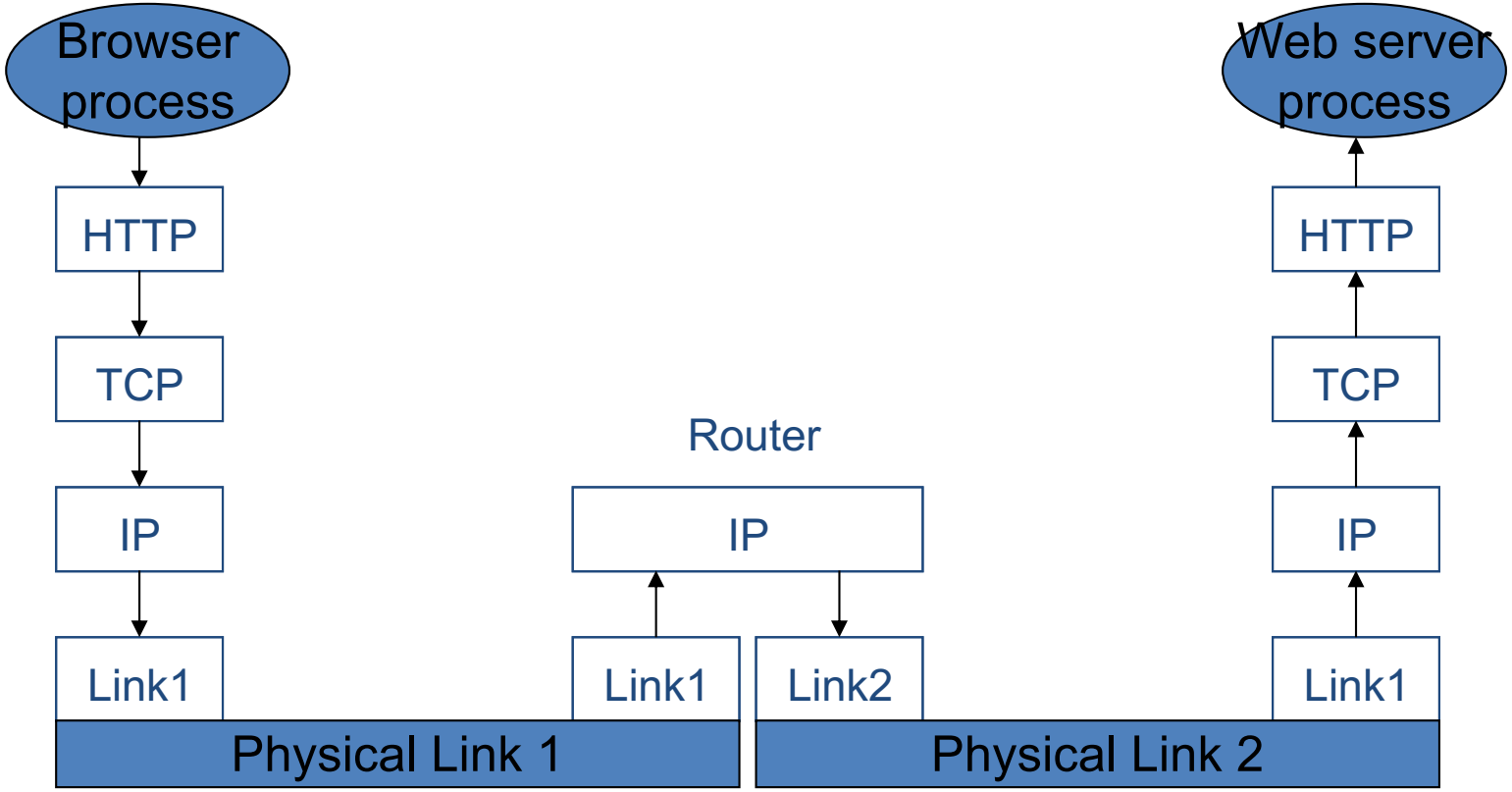
Network Components (for your reference)

- **Network**: Collection of hosts, links, and routers
- **Site**: Stub network, typically in one location and under control of one administration
- **Firewall/NAT**: Box between the site and ISP that provides filtering, security, and Network Address Translation
- **ISP**: Internet Service Provider. Transit network that provides IP connectivity for sites
- **Backbone ISP**: Transit network for regional ISPs and large sites
- **Inter-exchange (peering point, IX)**: Broadcast link where multiple ISPs connect and exchange routing information (peering)
- **Hosting center**: Stub network that supports lots of hosts (web services), typically with high speed connections to many backbone ISPs.
- **Bilateral peering**: Direct connection between two backbone ISPs

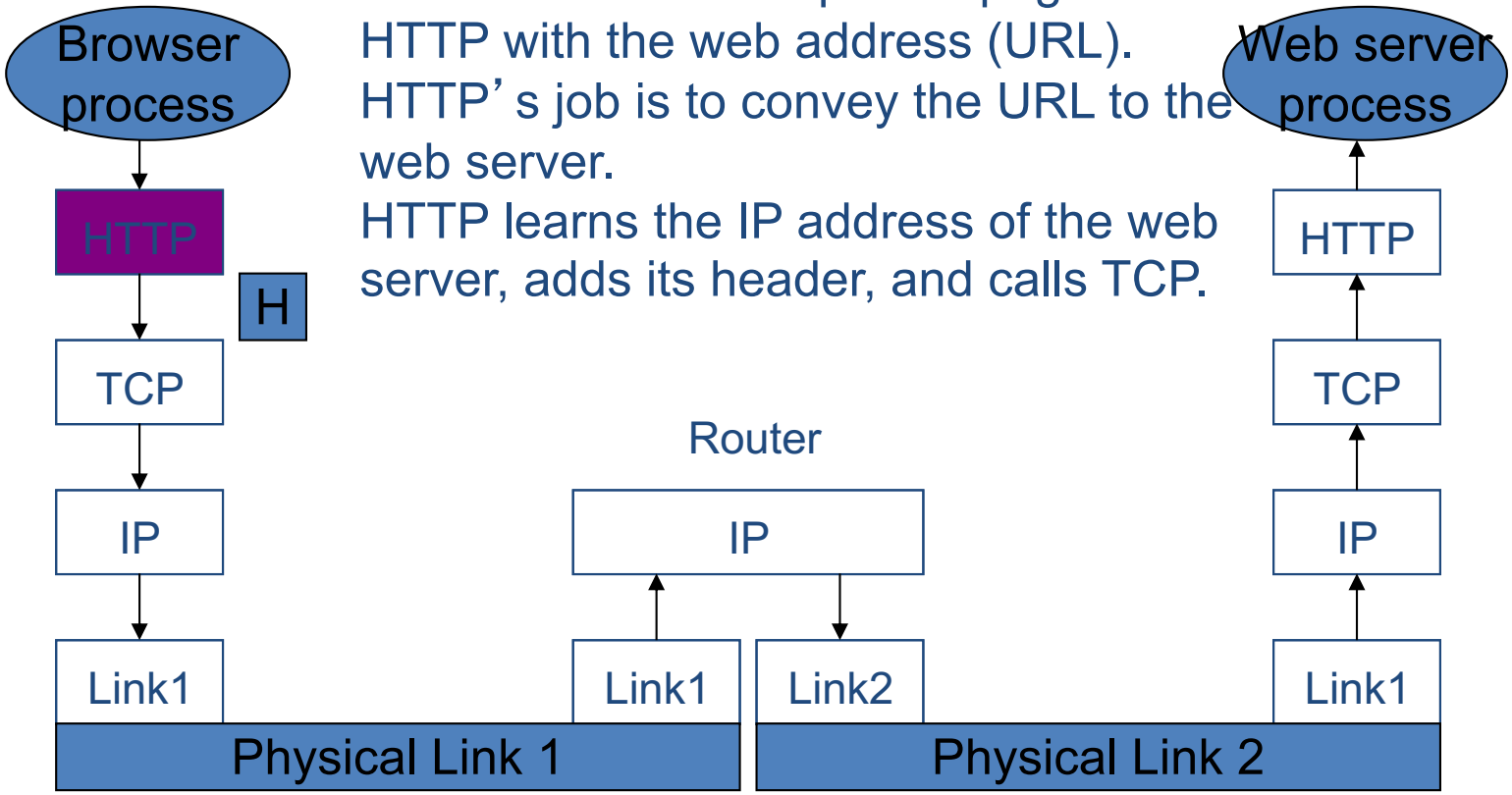
Protocol Layering

- Communications stack consists of a set of **services**, each providing a service to the **layer** above, and using services of the layer below
  - Each service has a programming **API**, just like any software module
- Each service has to convey information one or more **peers** across the network
- This information is contained in a **header**
  - The headers are transmitted in the same order as the layered services

Protocol layering example

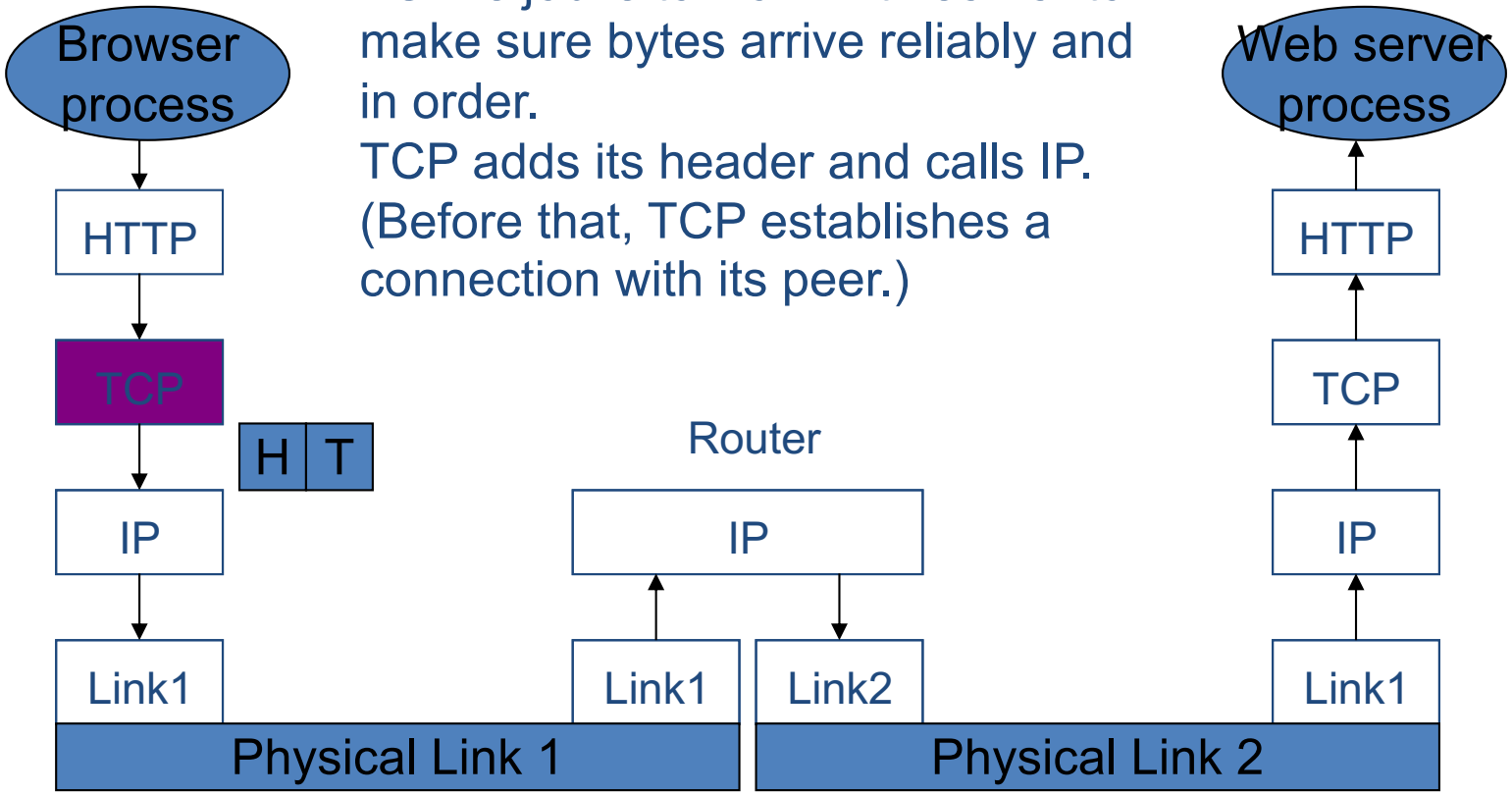


Protocol layering example

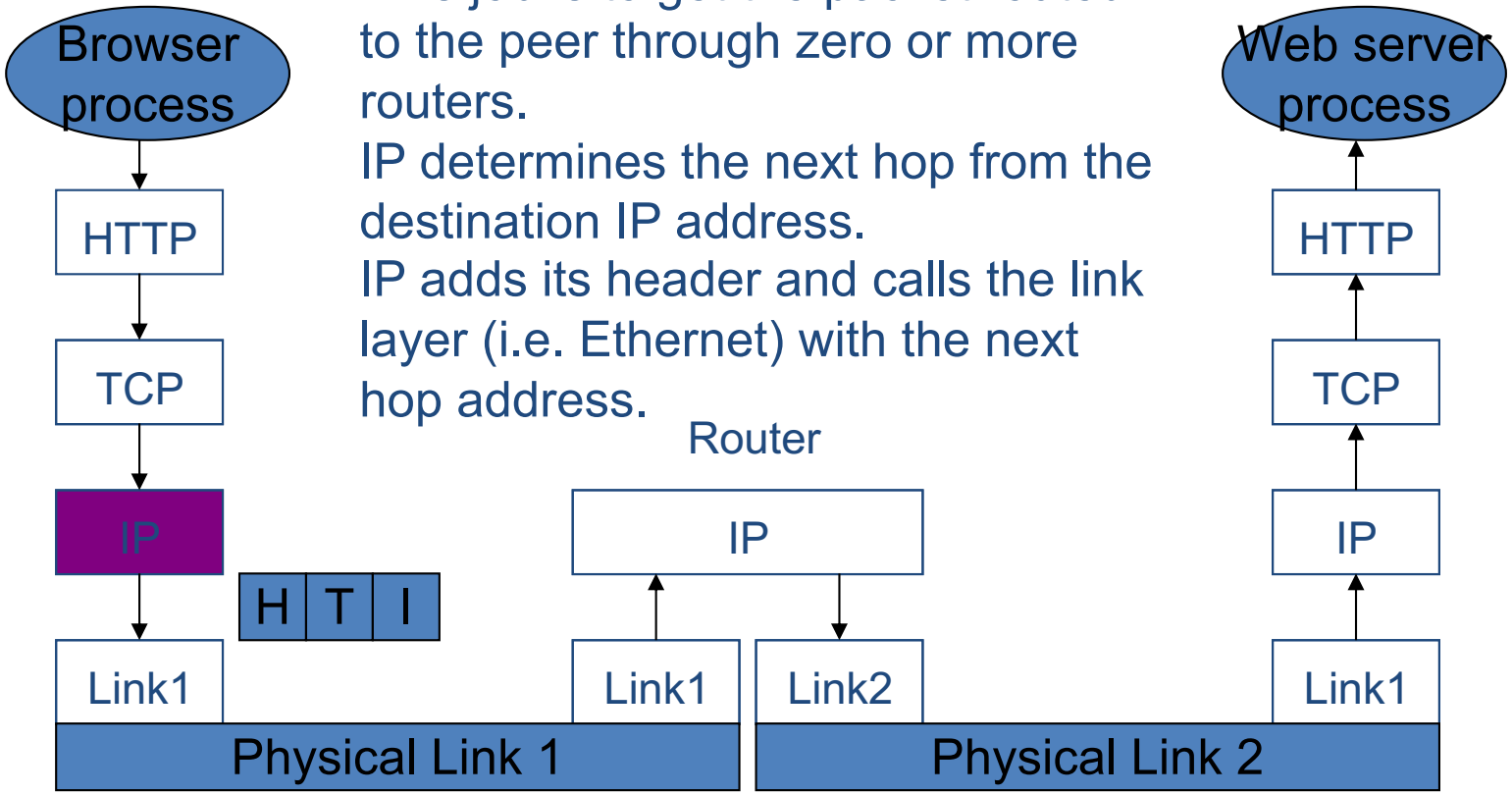


Browser wants to request a page. Calls HTTP with the web address (URL).
HTTP's job is to convey the URL to the web server.
HTTP learns the IP address of the web server, adds its header, and calls TCP.

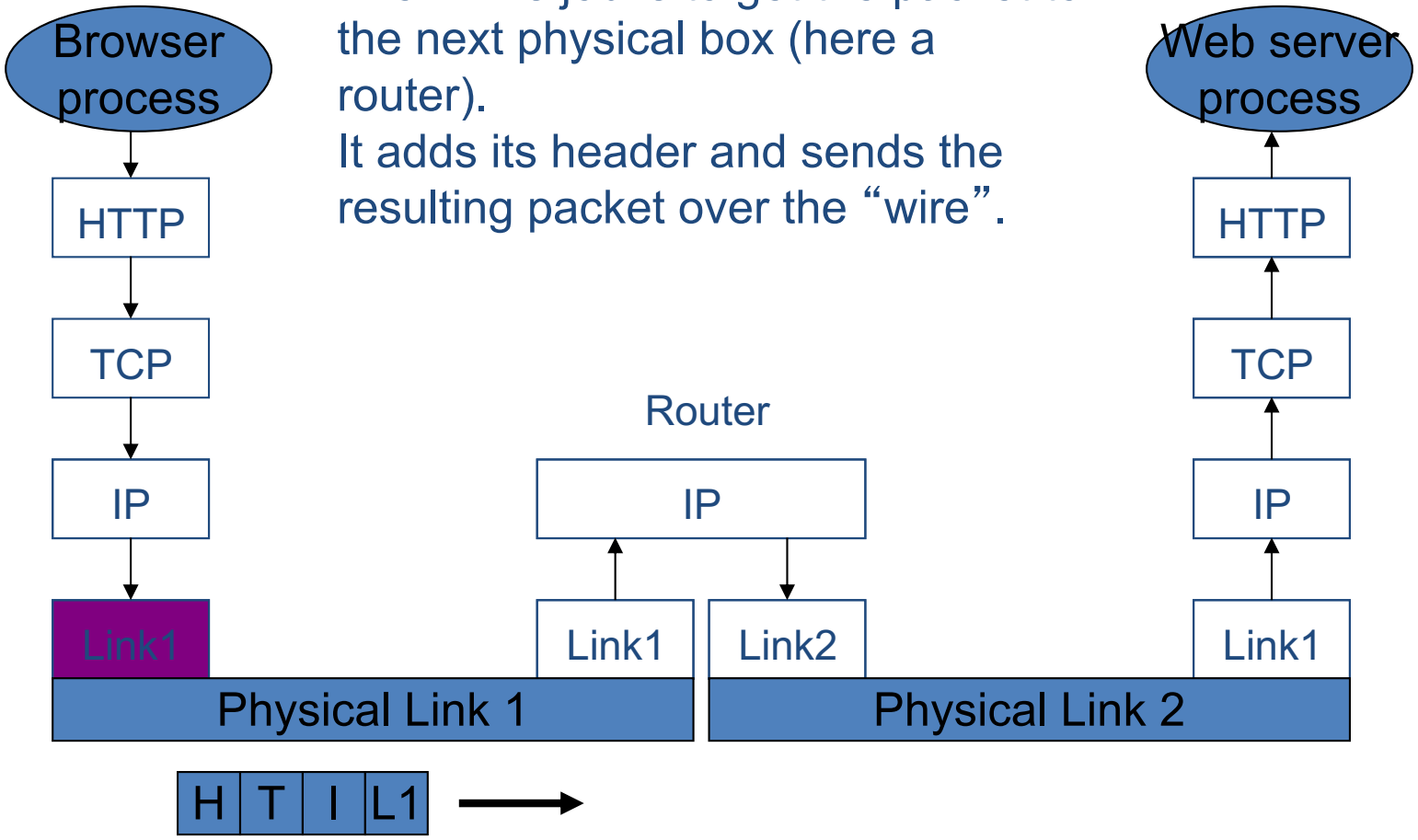
Protocol layering example



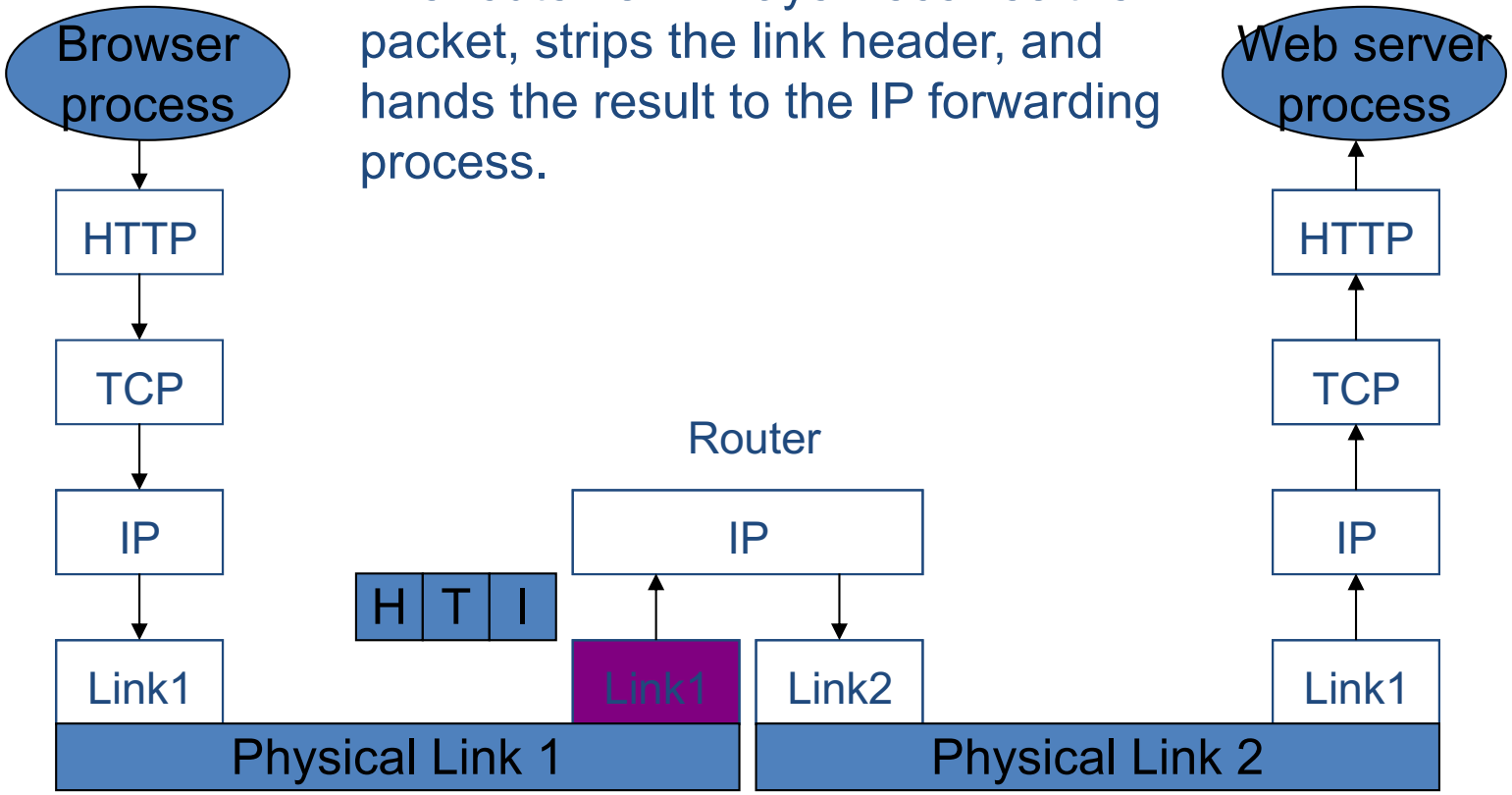
Protocol layering example



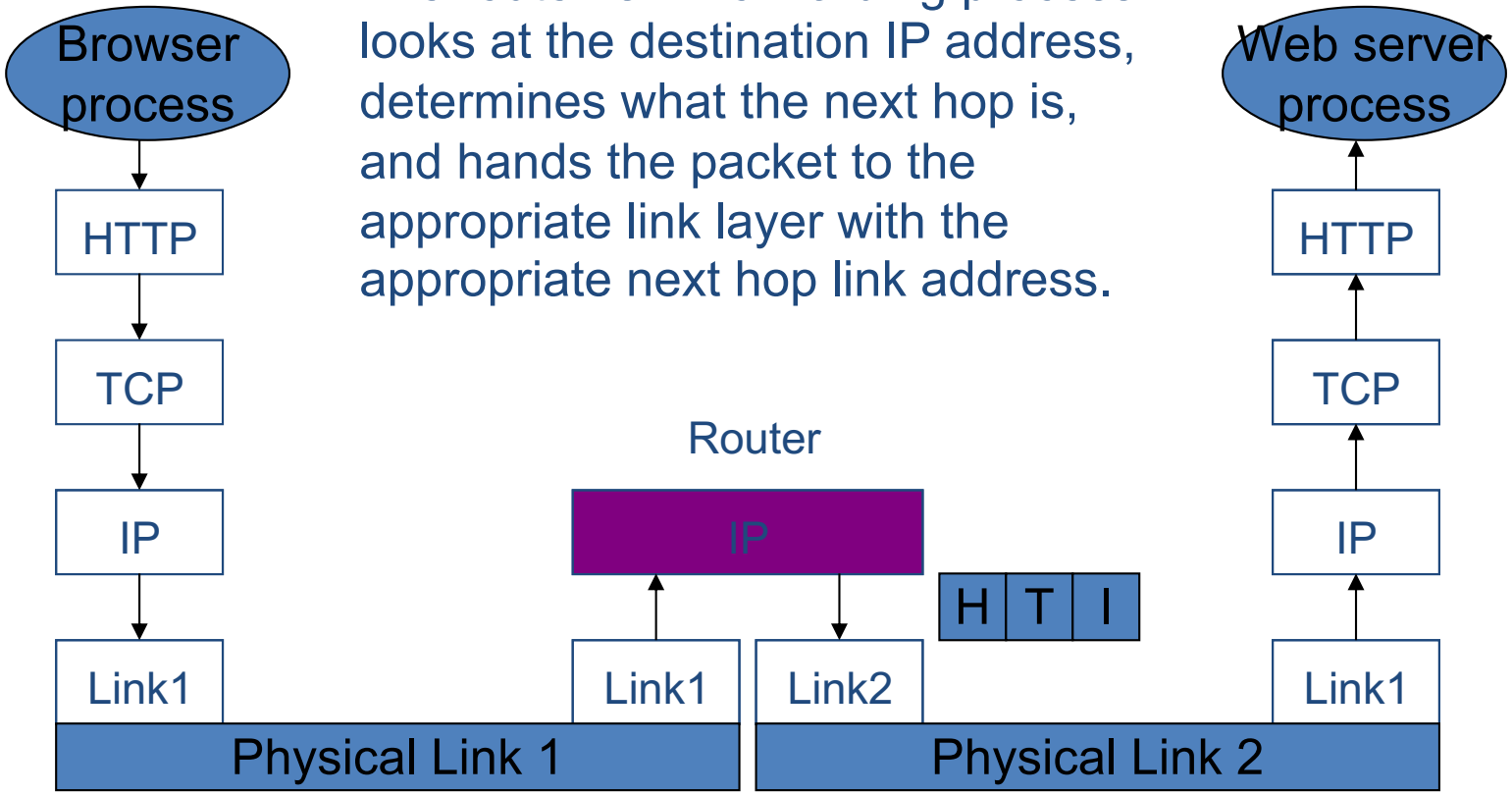
Protocol layering example



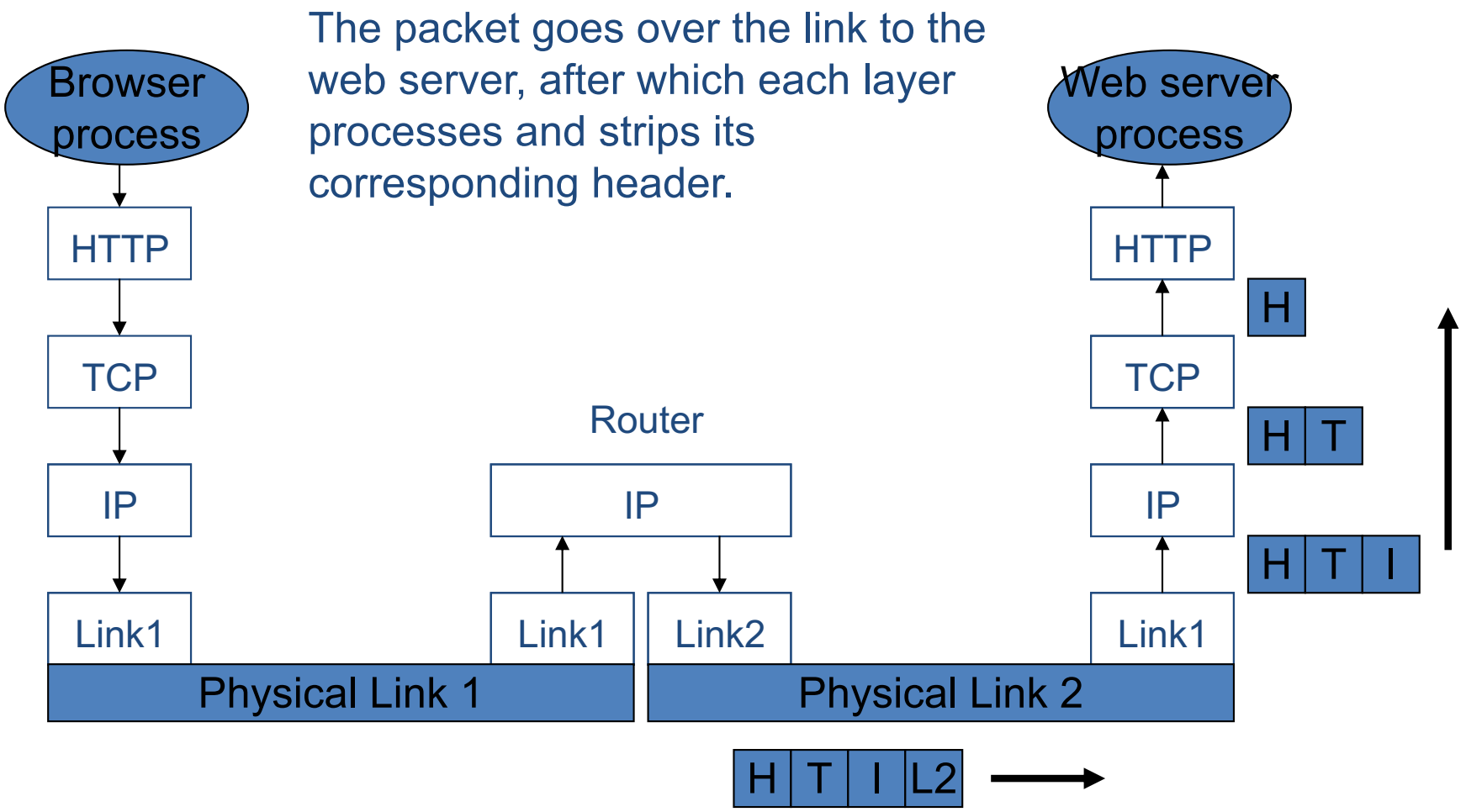
Protocol layering example



Protocol layering example



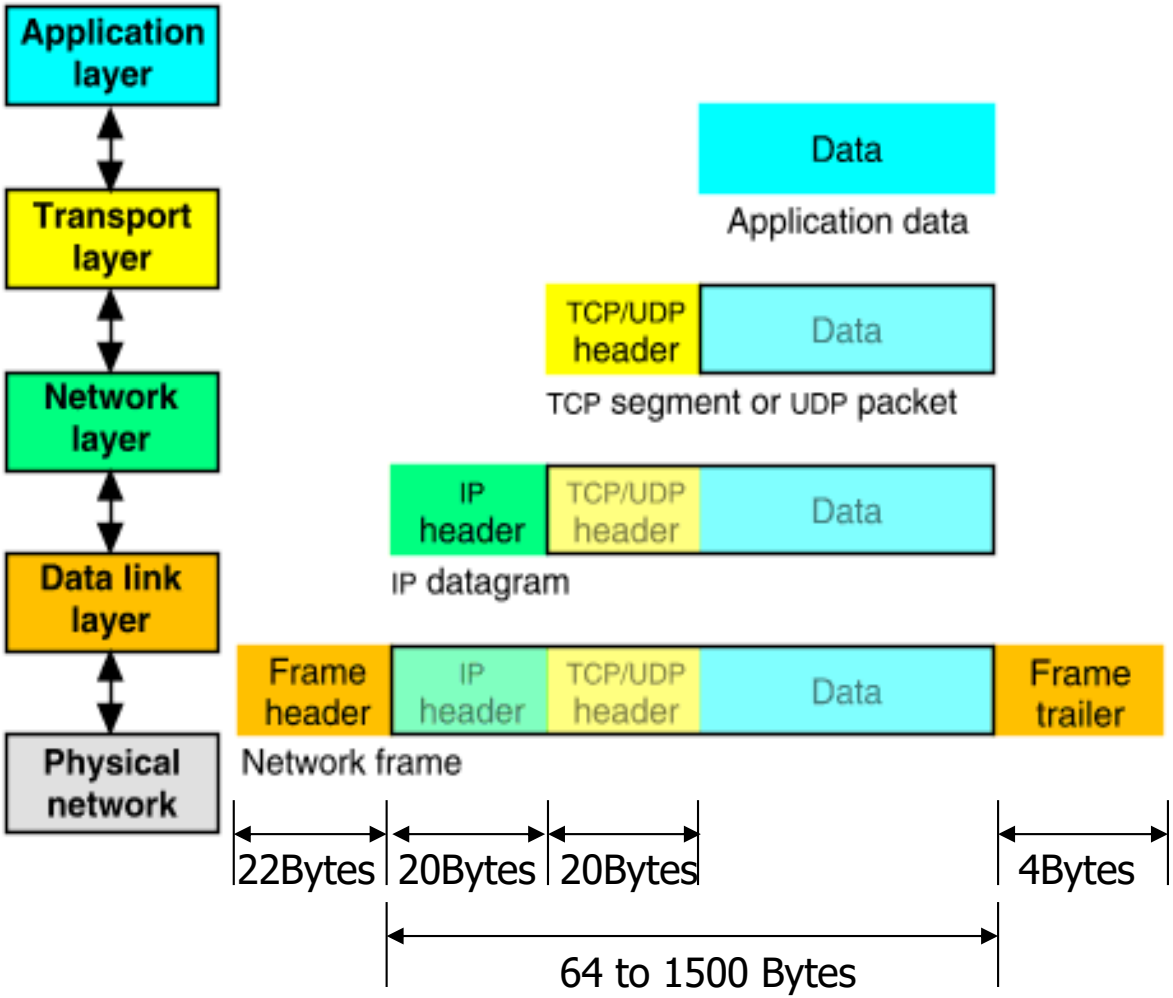
Protocol layering example



Address Resolution

- What about address resolution?
  - DNS
  - ARP
  - …
- Place them
  - On previous slides
- What protocols do DNS, ARP, etc. use to communicate?

Packet Encapsulation

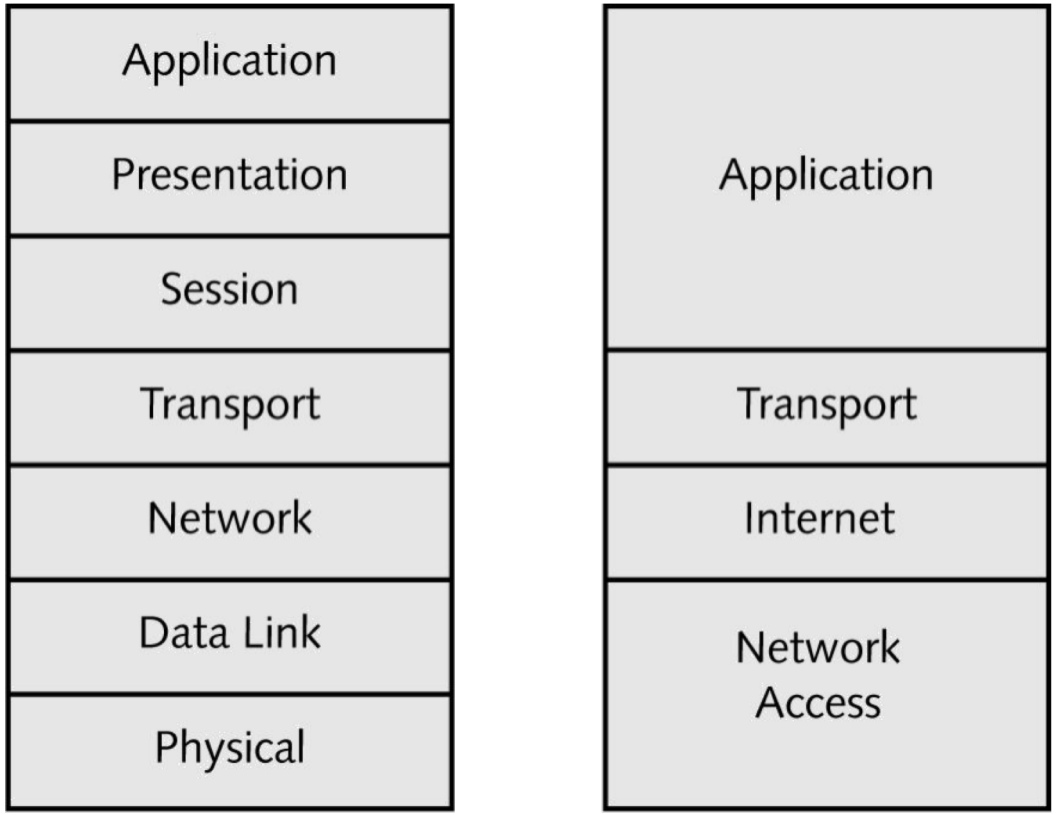


- The data is sent down the protocol stack
- Each layer adds to the data by prepending headers / trailers

Internet (for your reference)

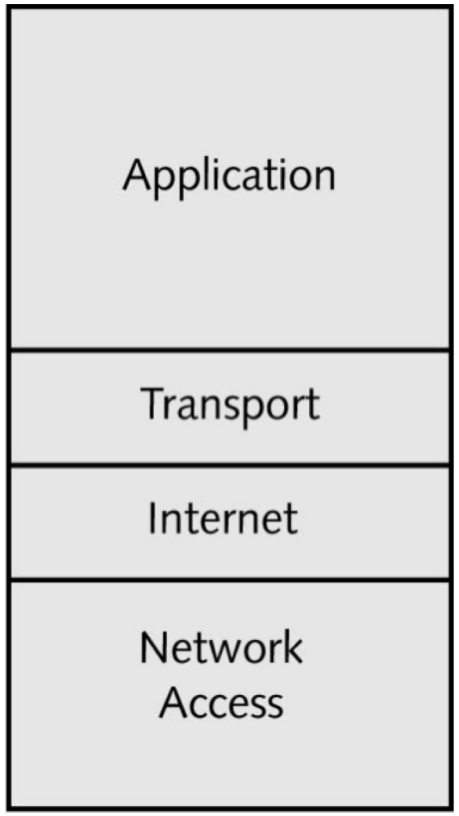
- The **link layer** (commonly Ethernet) contains communication technologies for a local network.
- The **internet layer** (IP) connects local networks, thus establishing internetworking.
- The **transport layer** (TCP, UDP) handles host-to-host communication.
- The **application layer** (for example HTTP) contains all protocols for specific data communications services on a process-to-process level (for example how a web browser communicates with a web server).

OSI vs. Internet Stack



Questions

- To which layer do the following protocols belong?
  - IP
  - TCP
  - UDP
  - WIFI
  - HTTP



http://olafland.polldaddy.com/s/protocols

Outline

- Warm-Up: Internet Hourglass
- Protocols and OSI Reference Model
- Internet Protocol Stack
- Selected Protocols

Selected Protocols

- IP, TCP, UDP

- Discussion
  - What is their job?
  - Which applications use them?
  - What features to they provide to the layer above?
  - How are the features realized?

Basic Elements of any Protocol Header

- **Demuxing** field
  - Indicates which is the next higher layer (or process, or context, etc.)
- **Length** field or header **delimiter**
  - For the header, optionally for the whole packet
- Header format may be **text** (HTTP, SMTP (email)) or **binary** (IP, TCP, Ethernet)
- When do I need this?
  - When going upwards in the stack
    - To which protocol to give a packet next

Demuxing fields

- Ethernet: Protocol Number
  - Indicates IPv4, IPv6, (old: Appletalk, SNA, Decnet, etc.)
- IP: Protocol Number
  - Indicates TCP, UDP, SCTP, …
- TCP and UDP: Port Number
  - Well known ports indicate FTP, SMTP, HTTP, SIP, many others
  - Dynamically negotiated ports indicate specific processes (for these and other protocols)
- HTTP: Host field
  - Indicates "virtual web server" within a physical web server
  - (Well, more like an identifier than a demuxing field)

IP (Internet Protocol)

- Three services:
  - Unicast: transmits a packet to a specific host
  - Multicast: transmits a packet to a group of hosts
  - Anycast: transmits a packet to one of a group of hosts (typically nearest)
- Destination and source identified by the IP address (32 bits for IPv4, 128 bits for IPv6)
- All services are unreliable
  - Packet may be dropped, duplicated, and received in a different order
  - When can each of these happen?

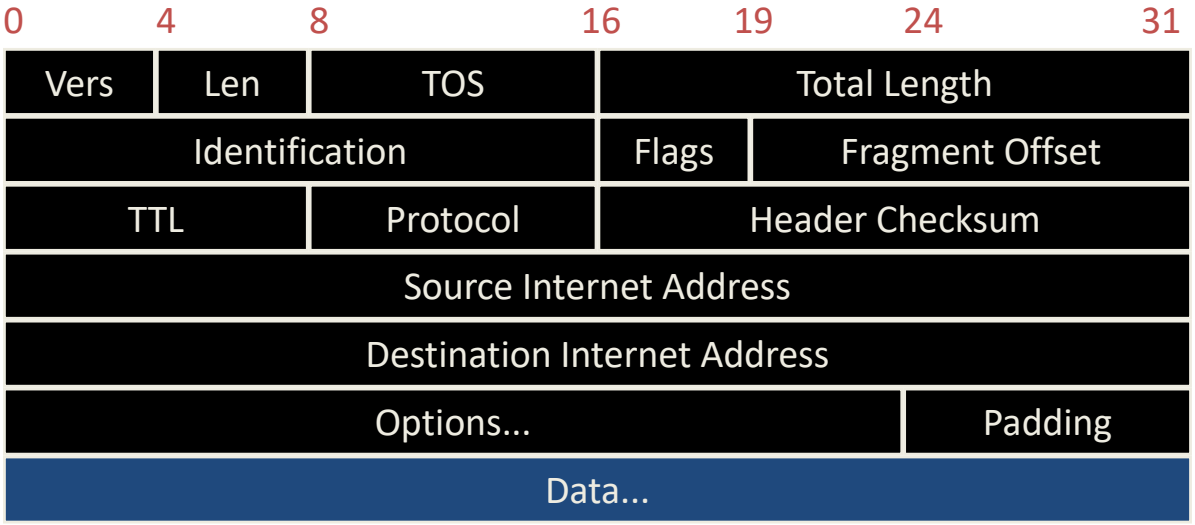
IP Address

- The “core” element of an IP Packet
- Both source and destination address may be modified in transit
  - By NAT boxes
  - But even so, sending a packet back to the source IP address will get the packet there
  - Unless source address is spoofed, which can easily be done
- IP (unicast) address is hierarchical, but host can treat it as a flat identifier
  - (almost…needs to know network mask)
  - Can’t tell how close or far a host is by looking at its IP address

IP(v4) Address Format

- In binary, a 32-bit integer
- In text, this: “128.52.7.243”
  - Each decimal digit represents 8 bits (0 – 255)
- “Private” addresses are not globally unique:
  - Used behind NAT boxes
  - 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
- Multicast addresses start with 1110 as the first 4 bits (Class D address)
  - 224.0.0.0/4
- Unicast and anycast addresses come from the same space

IP Datagram



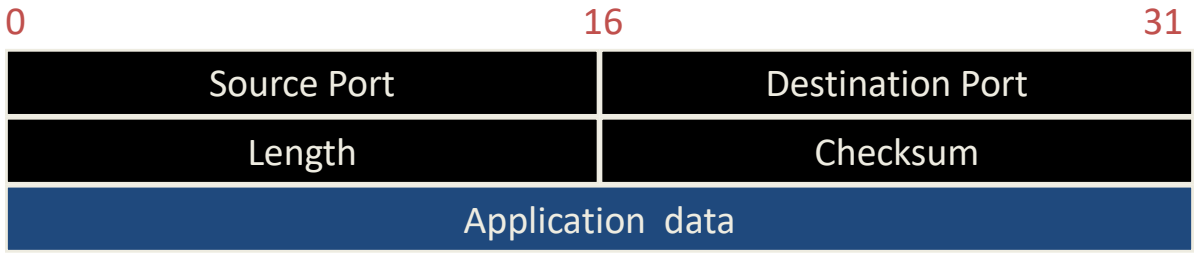
Field	**Purpose**
Vers	IP version number
Len	Length of IP header (4 octet units = 32bit words)
TOS	Type of Service
T. Length	Length of entire datagram (octets = bytes)
Ident.	IP datagram ID (for frag/reassembly)
Flags	Don't/More fragments
Frag Off	Fragment Offset

Field	**Purpose**
TTL	Time To Live - Max # of hops
Protocol	Higher level protocol (1=ICMP, 6=TCP, 17=UDP)
Checksum	Checksum for the IP header
Source IA	Originator's Internet Address
Dest. IA	Final Destination Internet Address
Options	Source route, time stamp, etc.
Data...	Higher level protocol data

UDP (User Datagram Protocol)

- Runs above IP
- Same unreliable service as IP
  - Packets can get lost anywhere:
    - Outgoing buffer at source
    - Router or link
    - Incoming buffer at destination
- But adds port numbers
  - Used to identify “application layer” protocols or processes
- Also a checksum, optional

UDP datagram



Field	Purpose
Source Port	16-bit port number identifying originating application
Destination Port	16-bit port number identifying destination application
Length	Length of UDP datagram (UDP header + data)
Checksum	Checksum of IP pseudo header, UDP header, and data

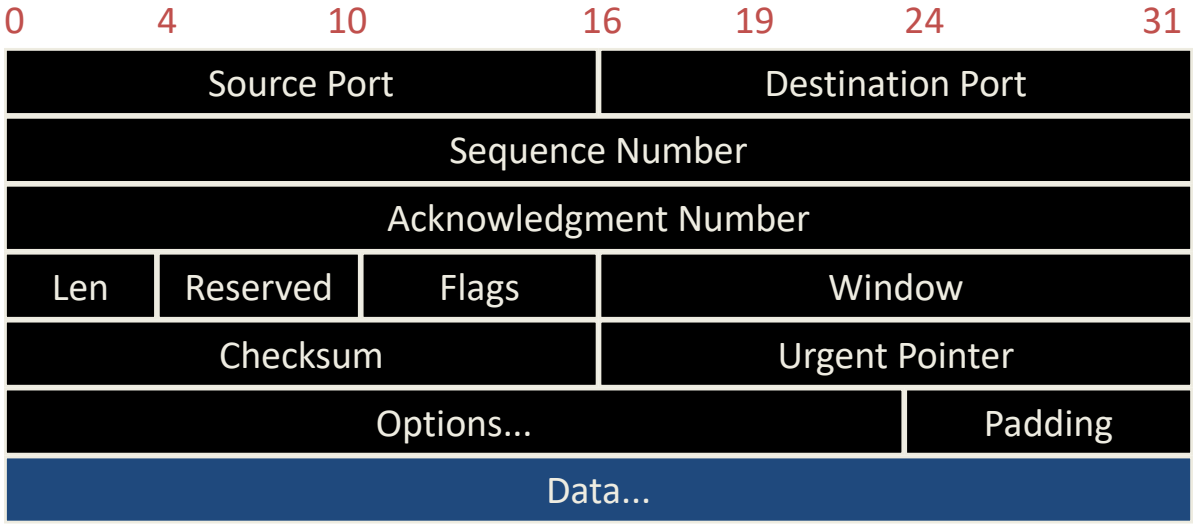
Typical applications of UDP

- Where packet loss etc is better handled by the application than the network stack
  - often: no transmission
    - Data is outdated anyway
- Where the overhead of setting up a connection isn't wanted
  - Where low-delay is critical
    - VOIP
    - Online gaming

TCP (Transmission Control Protocol)

- Runs above IP
  - Port number and checksum like UDP
- Service is in-order byte stream
  - Application does not absolutely know how the bytes are packaged in packets
- Flow control and congestion control
- Connection setup and teardown phases
- Can be considerable delay between bytes in at source and bytes out at destination
  - Because of timeouts and retransmissions
- Works only with unicast (not multicast or anycast)

TCP Segment



Field	Purpose
Source Port	Identifies originating application
Destination Port	Identifies destination application
Sequence Number	Sequence number of first octet in the segment
Acknowledgment #	Sequence number of the next expected octet (if ACK flag set)
Len	Length of TCP header in 4 octet units
Flags	TCP flags: SYN, FIN, RST, PSH, ACK, URG
Window	Number of octets from ACK that sender will accept
Checksum	Checksum of IP pseudo-header + TCP header + data
Urgent Pointer	Pointer to end of “urgent data”
Options	Special TCP options such as MSS and Window Scale

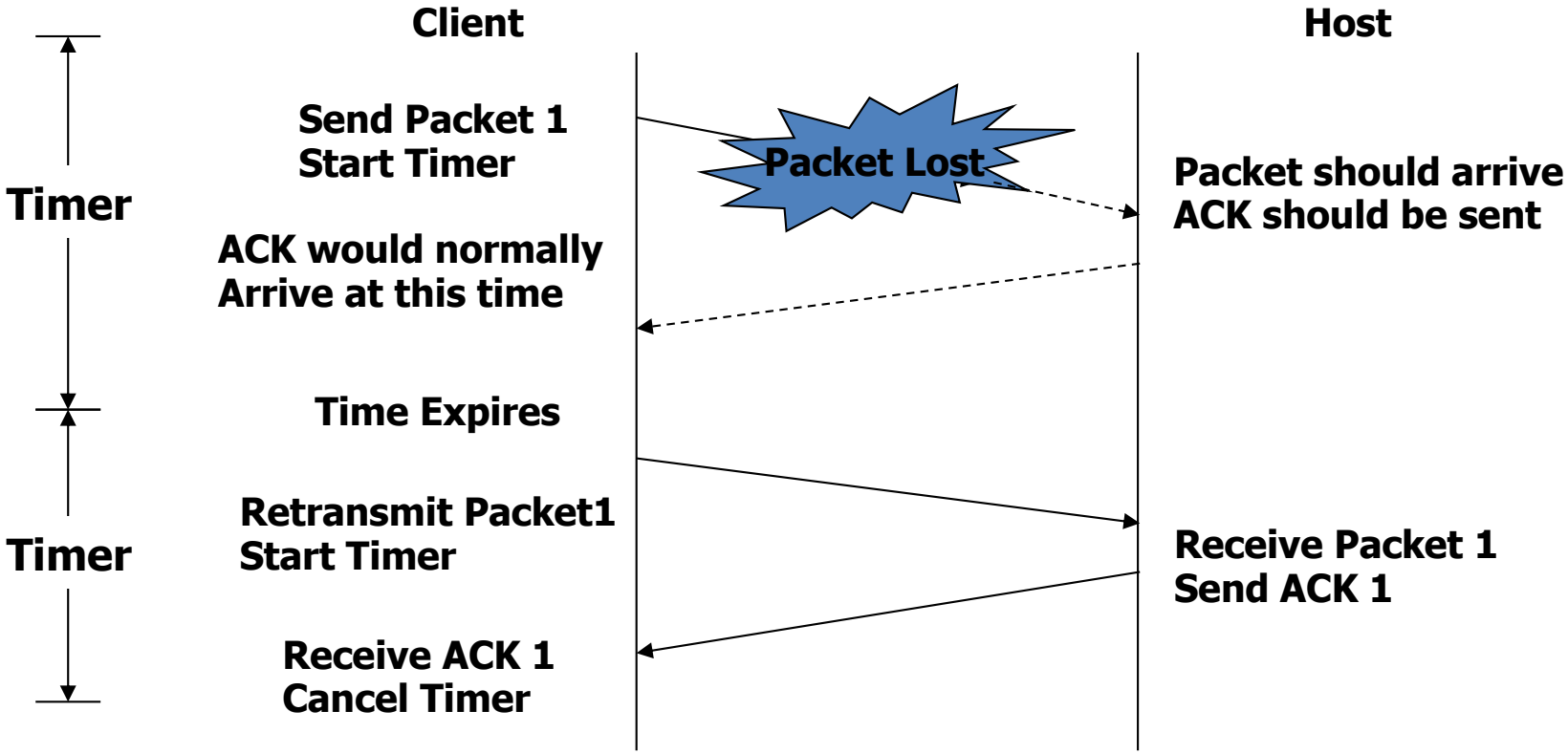
You just need to know port numbers, seq and ack are added

TCP: Reliability

- Reliability
  - How does it work?

- Acknowledgements
- Timeout
  - Retransmission
  - Close connection when too many retransmission
    - Notify application

TCP : Reliability

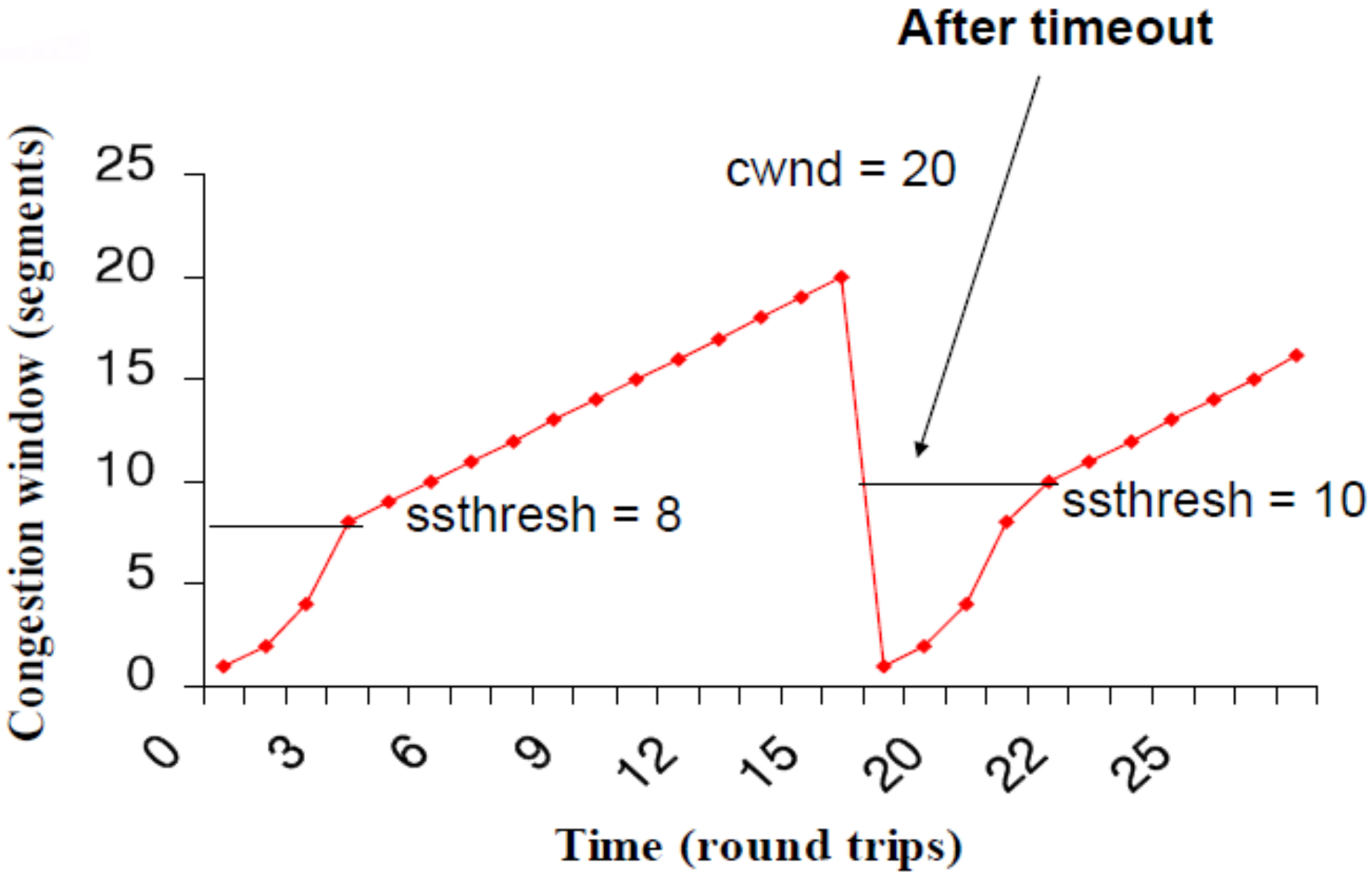


TCP: Congestion Control

- Congestion Control
  - How does it work?

- Try-And-Error
  - On successful delivery (ACK received)
    - Increase bandwidth
  - On packet loss
    - Decrease bandwidth

TCP: Congestion Control



UDP vs. TCP

- UDP is more real-time
  - Packet is sent or dropped, but is not delayed
- UDP has more of a "message" flavor
  - One packet = one message
  - But applications must add reliability mechanisms over it
- TCP is great for transferring a file or a bunch of email, but kind-of frustrating for messaging
  - Interrupts to application don't conform to message boundaries
  - No "Application Layer Framing"
- TCP is vulnerable to DoS (Denial of Service) attacks, because initial packet consumes resources at the receiver

Next Time

- Next time:
  - A bit on operating systems and concurrency
  - Architecture of distributed systems

Questions?

In part, inspired from / based on slides from Andrew Tanenbaum, Paul Krzyzanowski, Paul Francis, Ganesh Sittampalam, and many others