UMASS AMHERST

ECE 671
Introduction to Computer Networks

Review for Test #1

Topics

- Introduction
  - Lessons 1 & 2
- Application Layer Protocols (HTTP, SMTP)
  - Lesson 3
- Transport Layer Protocols (TCP, UDP)
  - Lesson 4
- Transport Layer (Congestion Control)
  - Lesson 5
- Network Layer Protocol (IP)
  - Lesson 6
- Data Link Layer Protocol (Ethernet)
  - Lesson 7
- Software Defined Networks (SDN)
  - Lesson 8

Distributed Applications

- Distributed computing applications use networks
- Distributed application requires data communication
  - Data are stored at different places
  - Data need to be exchanged between computers
- Computer networks enable data communications
  - Network connects geographically distributed nodes
  - Data can be sent between connected nodes
- What applications are inherently distributed?
  - Cannot be implemented on a single computer

Whiteboard: Classify the Applications

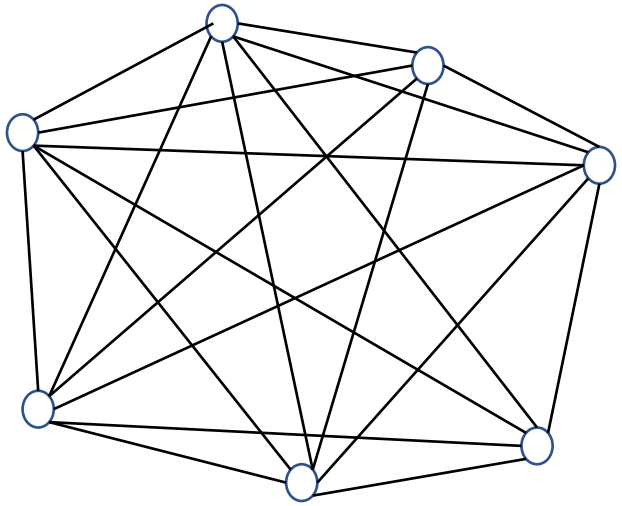
Applications	Distributed	NOT Distributed
Social networking		
Text processing		
Email		
Online shopping		
Computer gaming		
Scientific computing		
Web browsing		
Distance education		
TV/video distribution		

Distributed Applications

- Classify these example applications (distributed or not distributed)
  - Social networking (distributed)
  - Text processing (not distributed)
  - Email (distributed)
  - Online shopping (distributed)
  - Computer gaming (not distributed, distributed for multiplayer)
  - Scientific computing (not distributed)
  - Web browsing (distributed)
  - Distance education (distributed)
  - TV/video distribution (distributed)

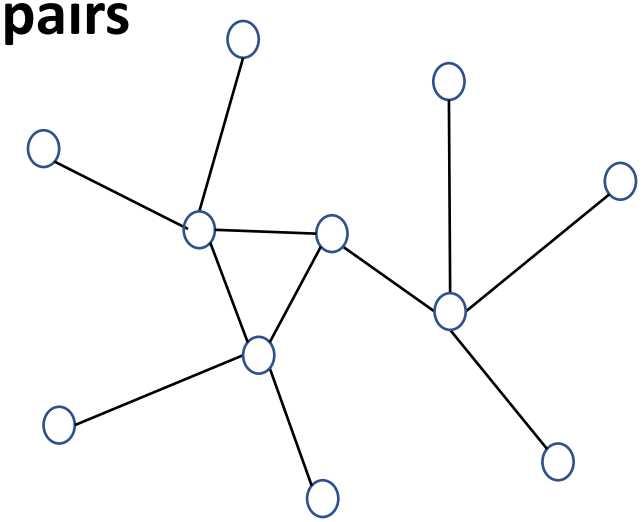
Group Discussion and Report Back (Pen): How to Connect Nodes

- How would you connect two nodes?
  - Three nodes? Four nodes? Five Nodes? …
- Networks can be structured differently
- Simplest approach
  - Connect all computers directly to each other
  - Very expensive for large networks
- What would be a better approach?



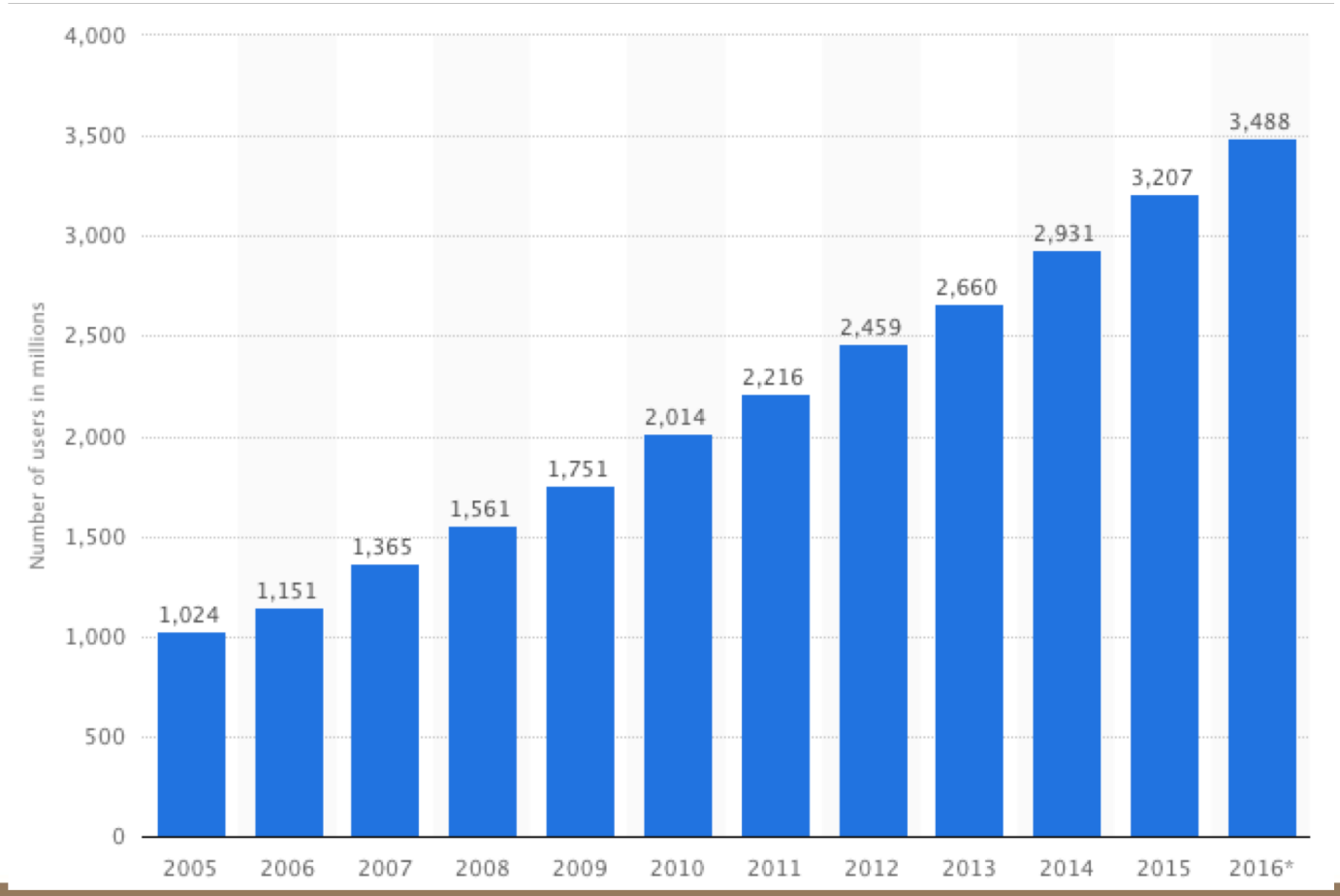
How to Connect Nodes

- Better approach to structuring networks
  - Connect computers via multi-hop network
  - Cheaper since links are used my many node pairs
- Network operation is more complex
  - Data relay from node to node
  - Set of protocols specifies rules
- This course looks at structure, operation, and performance of such networks with multiple hops



Size of Internet

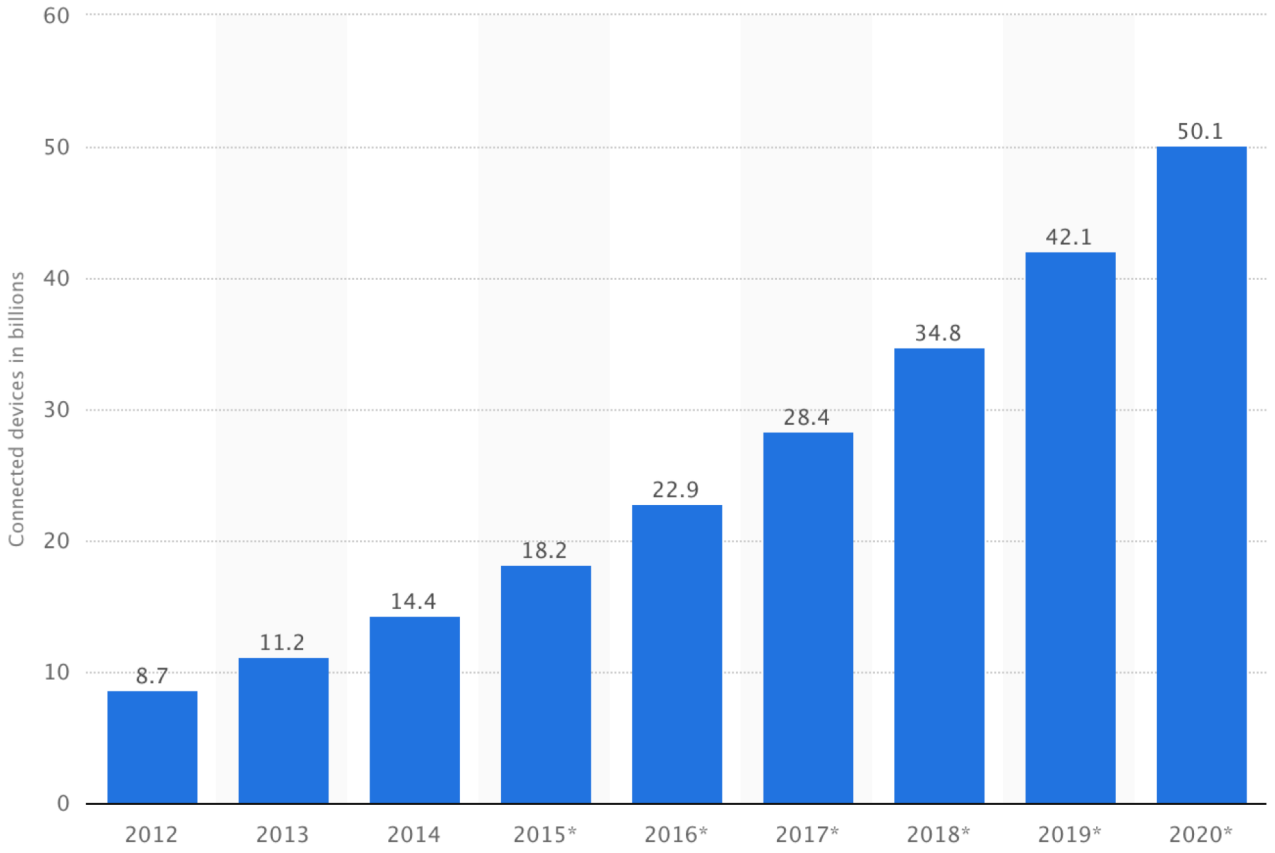
- **Number of Internet users worldwide in millions**



From:
statista.com

Orchestrated Discussion (Hand Raise): Size of Internet

- **Number of Internet-connected devices in millions**



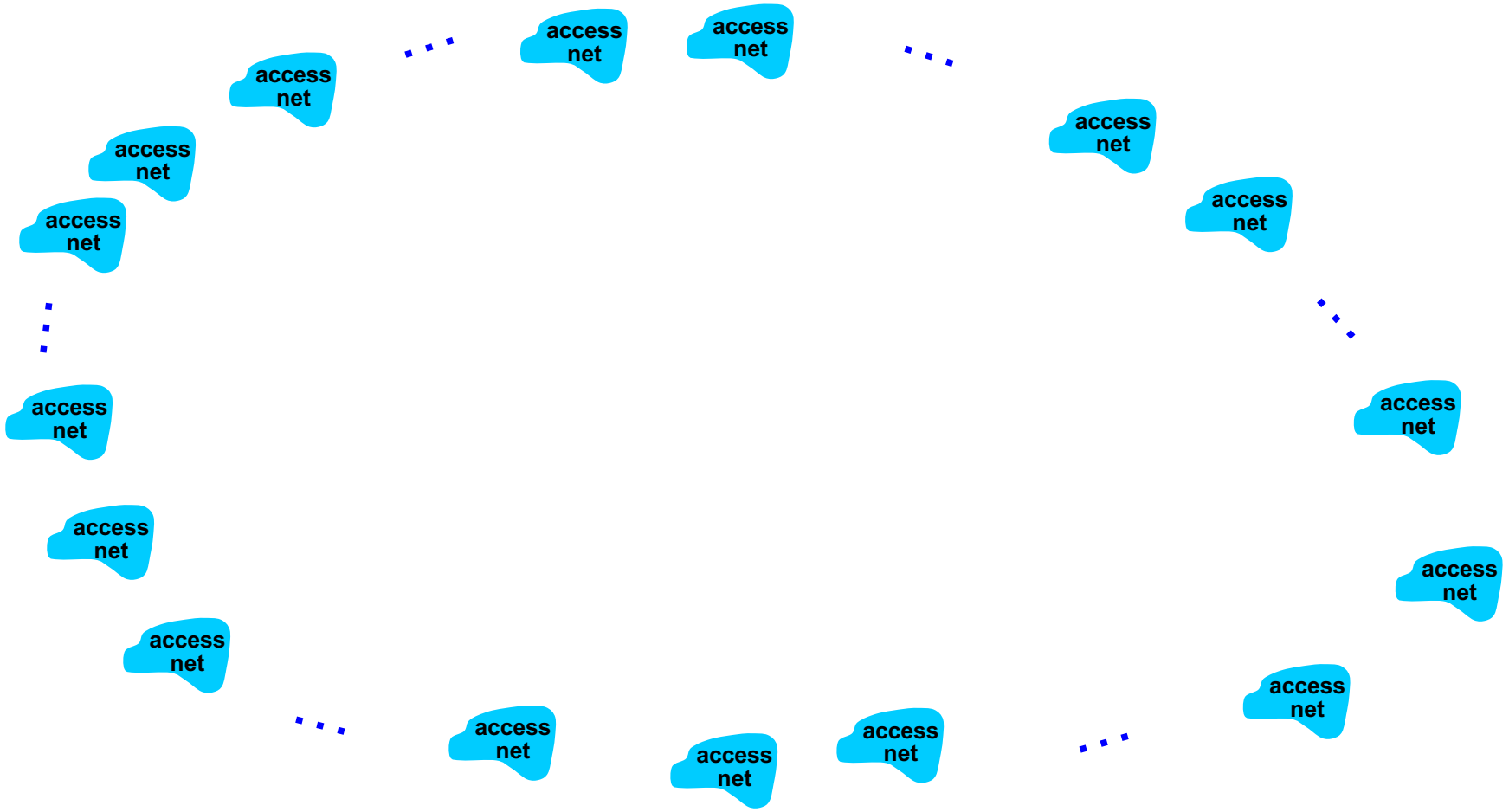
From:
statista.com

Internet Structure

- Very large number of nodes need to be connected
  - What is the structure of the Internet?
- Group nodes into networks (e.g., access network)
  - Easier to think of networks than of individual nodes
- Internet is structured as network of networks
  - Each network operates independently
  - Networks collaborate to connect devices across many networks
- Terminology: Internet Service Provider (ISP)
  - Network/entity that connects to Internet

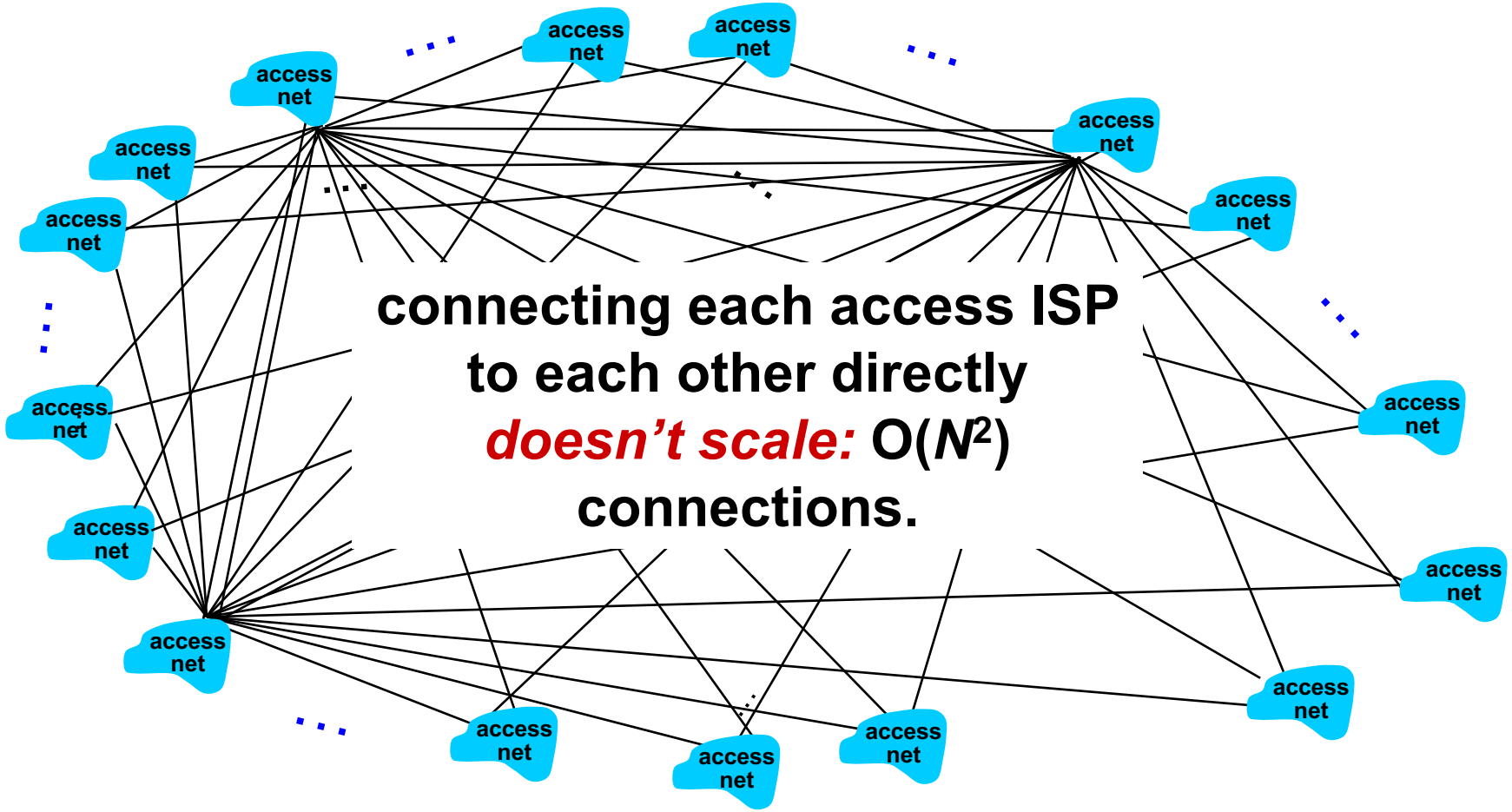
Internet Structure

- Same problem: how to connect millions of access networks?



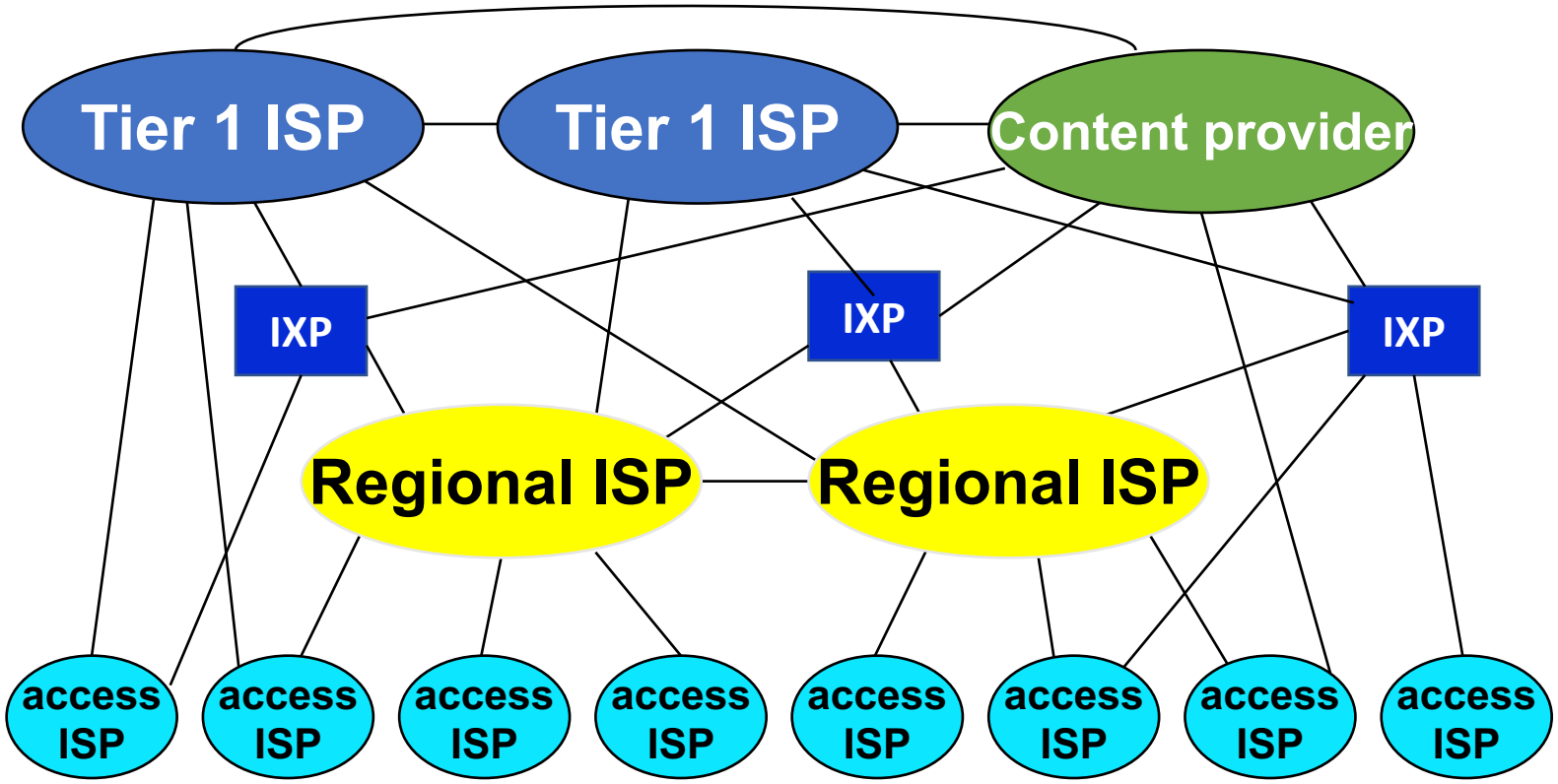
Internet Structure

- Connecting each ISP directly leads to scalability problem



Internet Structure

- Tiered view of Internet structure
  - Tier 1: ISPs with national and international coverage
  - Content providers may bypass Tier 1 ISPs

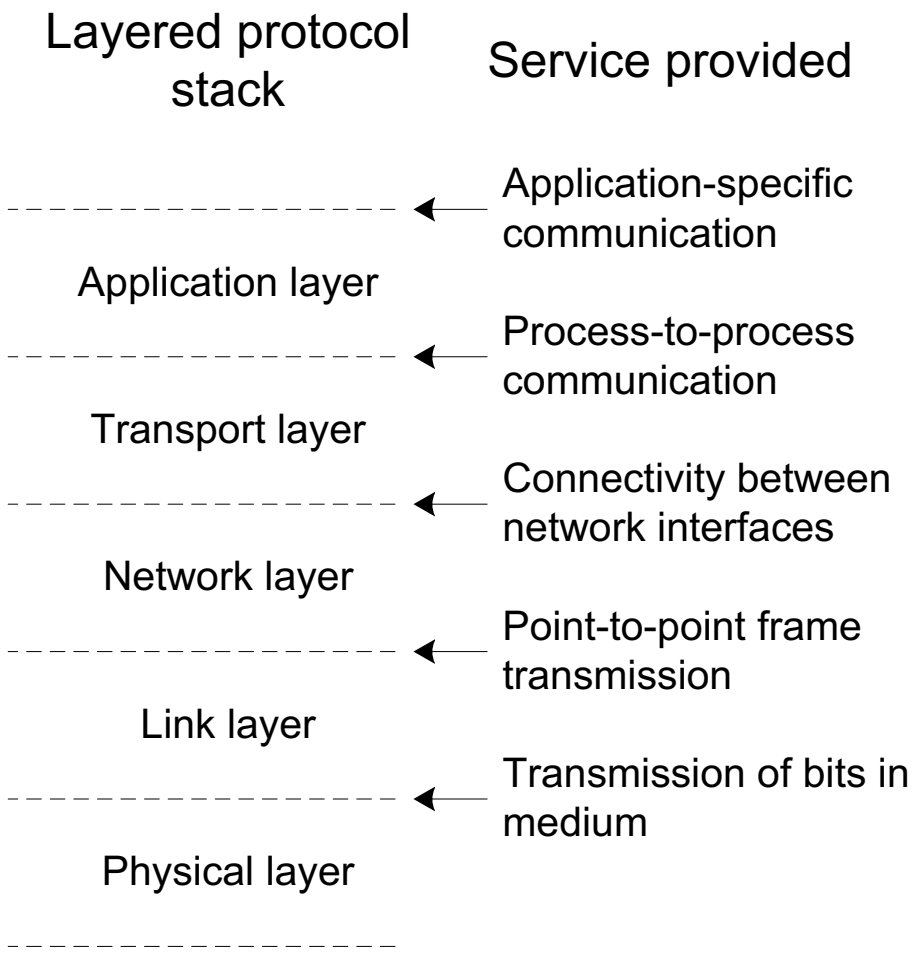


Modularization

- Breaking complex systems into smaller components
  - Modularization structures functionality
  - Interfaces describe interactions between components
- Module implementation is interchangeable
  - Change of implementation does not affect other modules
- Layered reference model of Internet
  - OSI reference model protocol stack (7 layers)
    - Session layer and presentation layer not used widely
  - Internet protocol stack (5 layers)

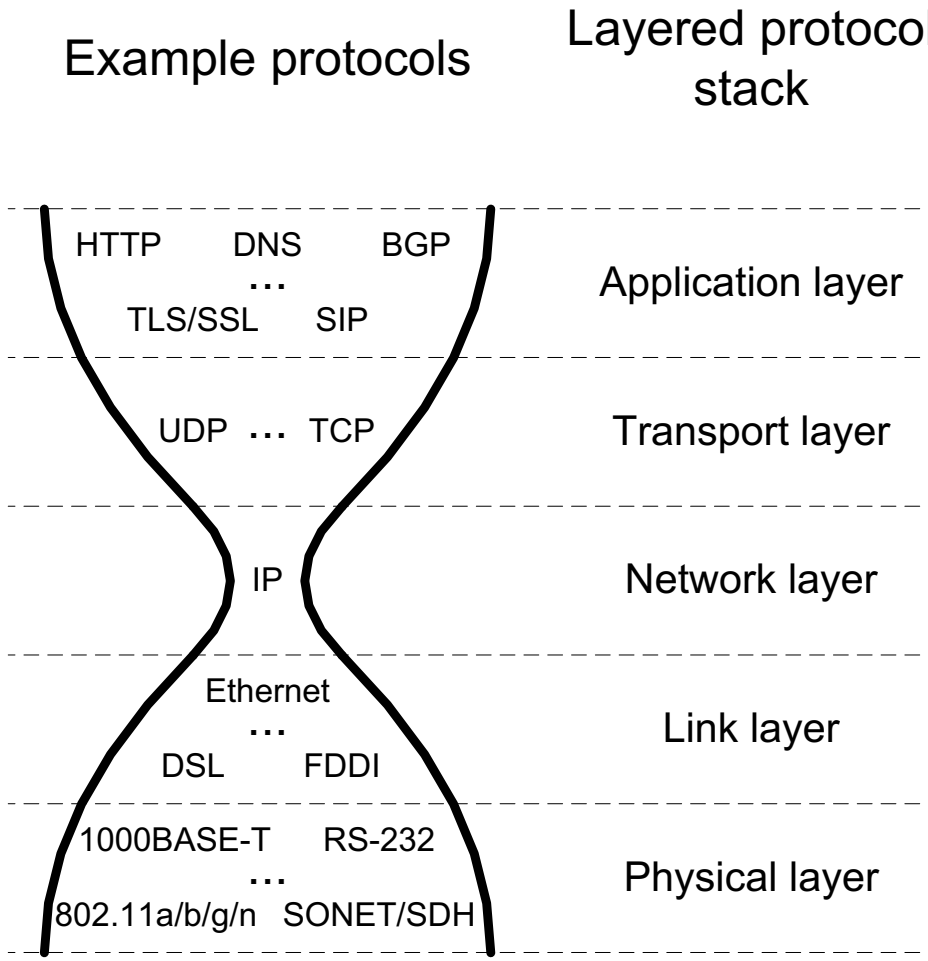
Interfaces between Layers

- Each layer in protocol stack provides a “service”
  - Uses service from lower layers
- Benefits of layering
  - Isolates complexity
  - Clearly defined interfaces
- Protocols implement functionality within layer
  - Different protocols can achieve module functionality



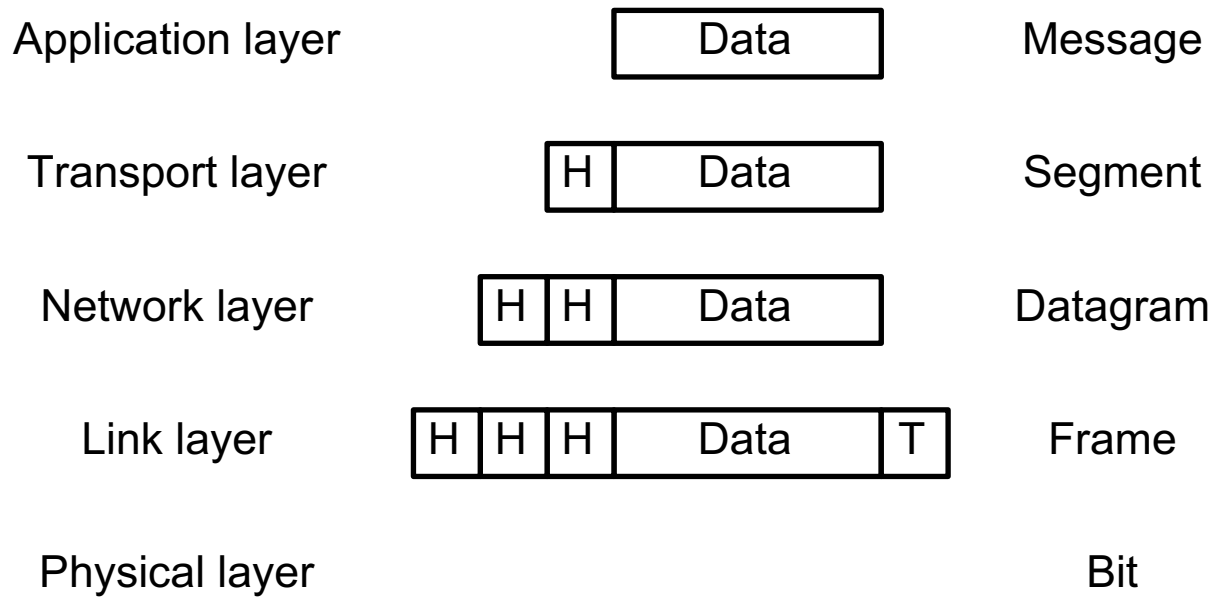
Internet Architecture

- "Hourglass architecture"
- Achieves interoperability
  - Single, common network layer protocol: Internet Protocol (IP)
  - All network nodes need to support this protocol
- Supports diversity
  - Different link/physical layer protocols below
  - Different transport/application layer protocols above



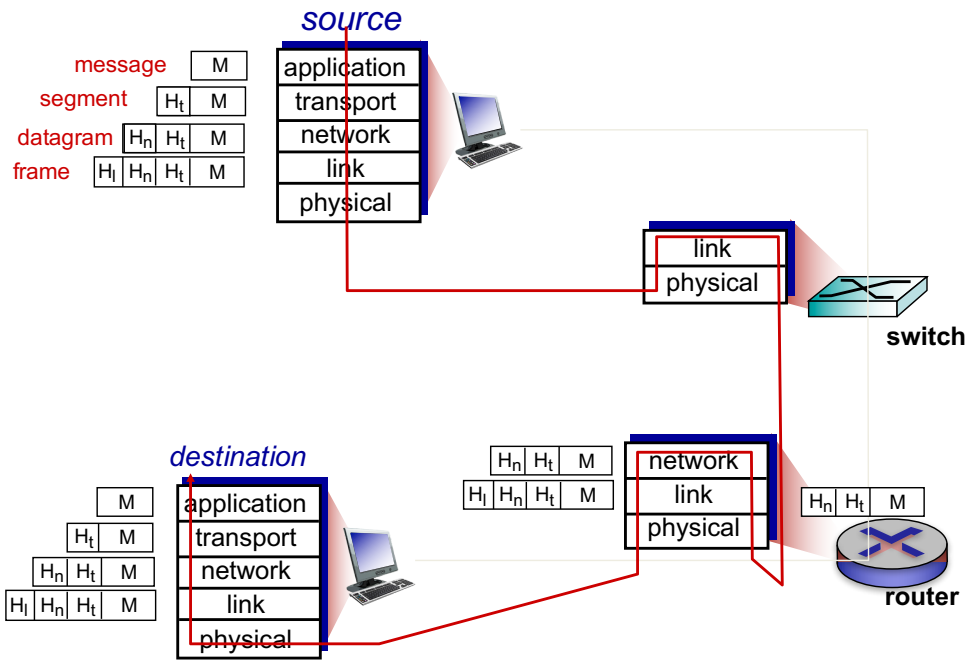
Protocols

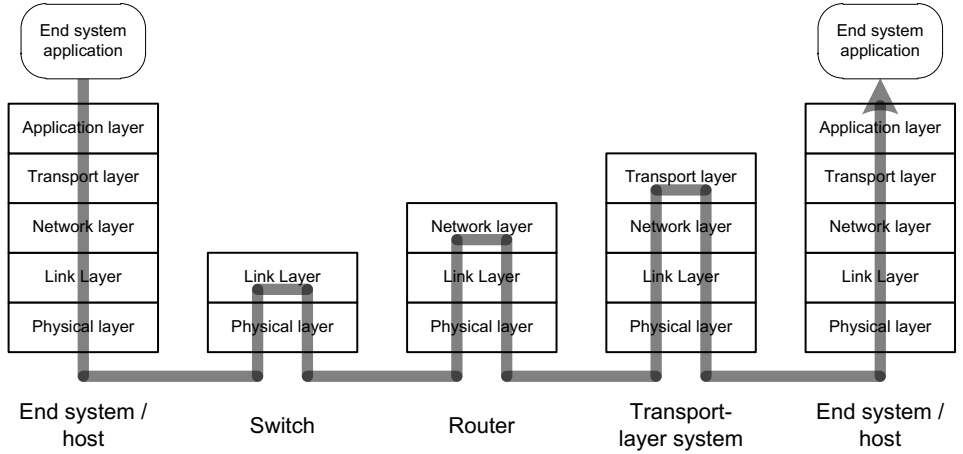
- Protocols define communication between entities
  - Format and order of messages
  - Actions taken on transmission and/or receipt of message or other event
- Protocols use headers (and trailers) for control information
  - Naming depends on layer



Network Devices

- Network devices differ by highest layer processed
  - Devices can process/modify headers up to that layer
  - Switches and routers are most common





Example Protocol Stack

Layer	Example protocols
Application layer	Hypertext Transfer Protocol (HTTP)
Transport layer	Transmission Control Protocol (TCP)
Network layer	Internet Protocol (IP)
Link layer	Ethernet
Physical layer	1000BASE-T

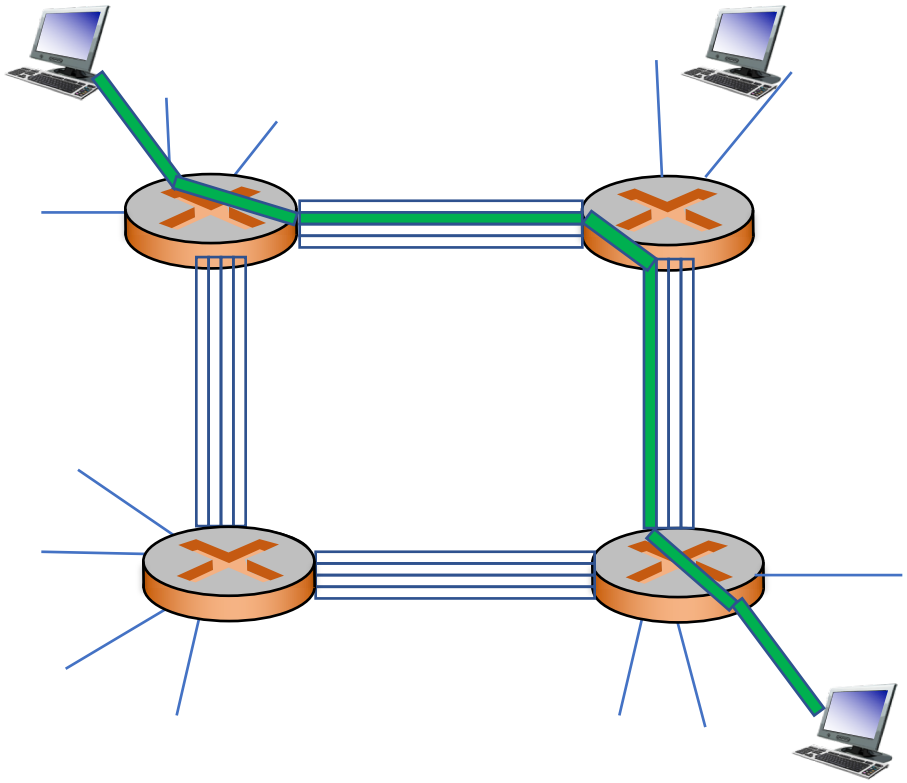
- **We will review Application Layer through Link Layer**

Link Sharing in the Internet

- Circuit switching
  - Fixed end-to-end connection
  - Link resources are reserved
  - Similar to telephony network
- Packet switching
  - Each data packet travels through network on its own
  - Buffering on nodes to handle contention for link
  - No guarantees for resources
- Hybrid approaches
  - E.g., virtual circuit switching

Circuit Switching in the Internet

- Links have multiple (four) circuits
  - Connection reserves a circuit during setup phase
- End-to-end connection by connecting circuits between switches
- Guaranteed access to reserved circuit
  - Cannot be used by other connection
- Same principle as telephony network



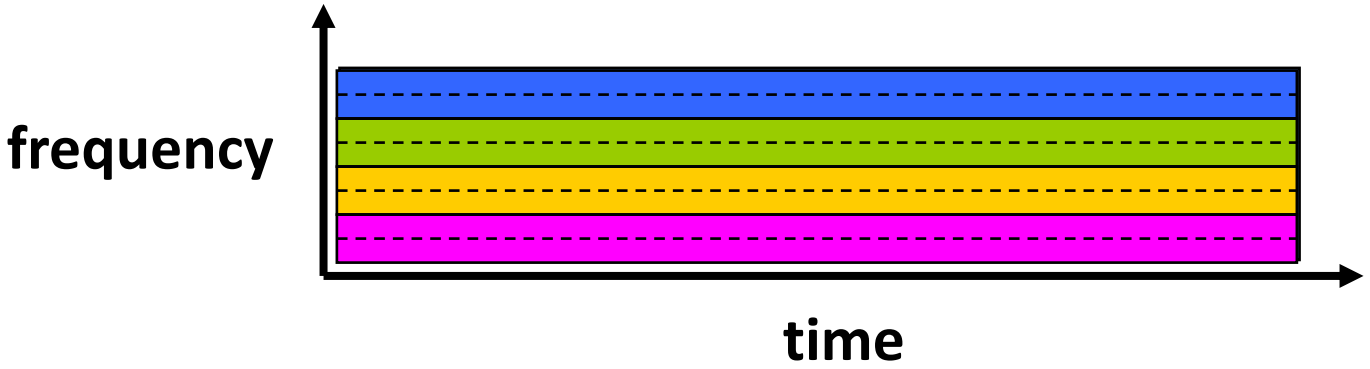
Link Sharing in Circuit Switching

- Approaches to divide link resources:

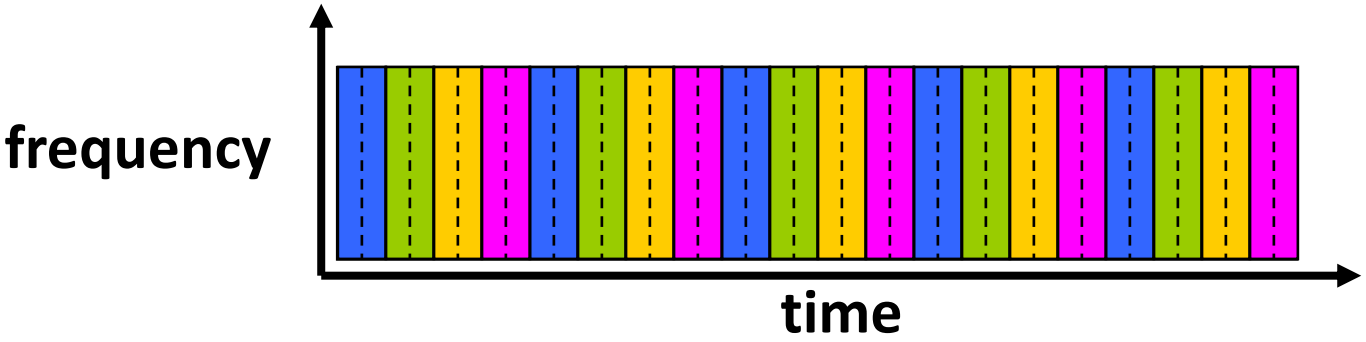
Example:

4 users

Frequency division multiplexing (FDM)

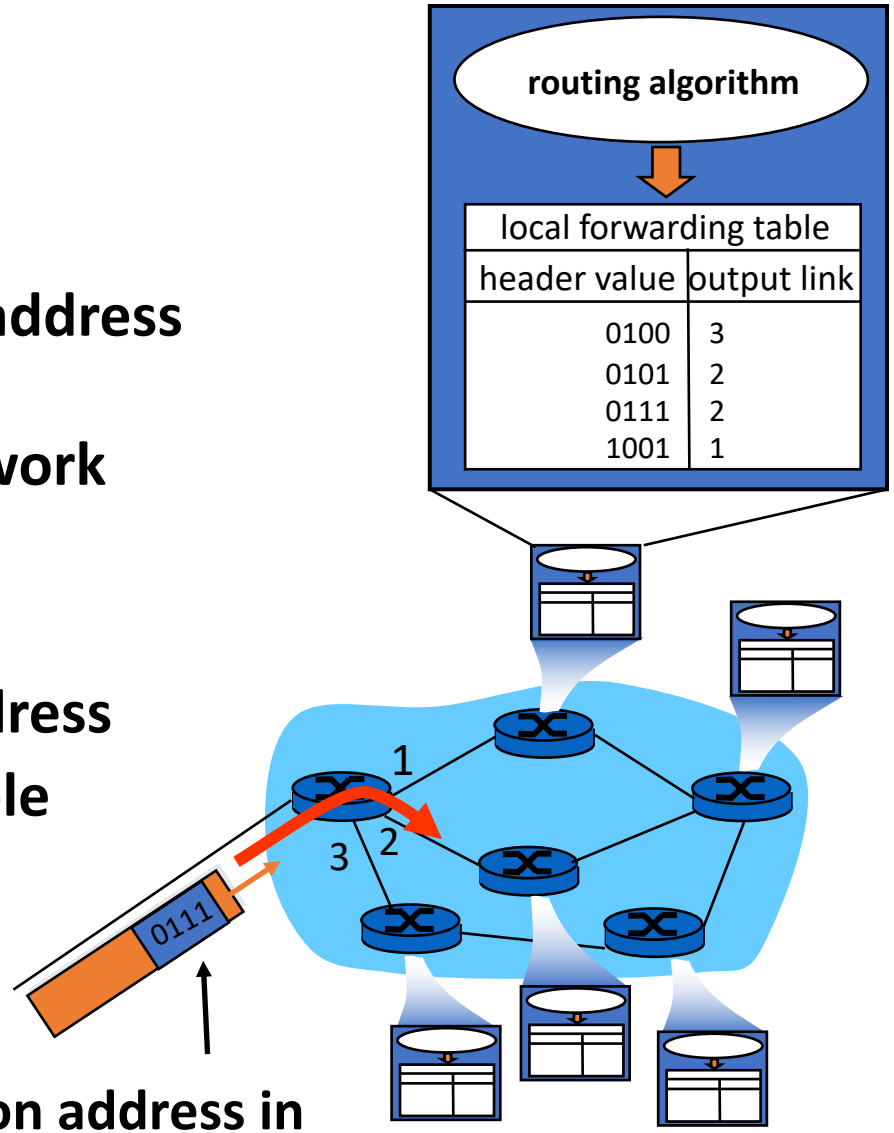


Time division multiplexing (TDM)



Packet Switching

- Traffic is divided into packets
  - Packets carry destination address in header
  - Each packet traverses network independently
- Nodes forward packets
  - Lookup on destination address
  - Each nodes needs to be able to forward to any destination address



Whiteboard: Circuit Switching vs. Packet Switching

Which property do you associate with circuit switching vs. packet switching? (Why?)

Property	Circuit Switching	Packet Switching	Depends
Better resource utilization			
Guaranteed bandwidth performance			
Easier to implement			
Less end-to-end delay variation			
Fairer			
More robust under failure			
Cheaper			

Circuit Switching vs. Packet Switching

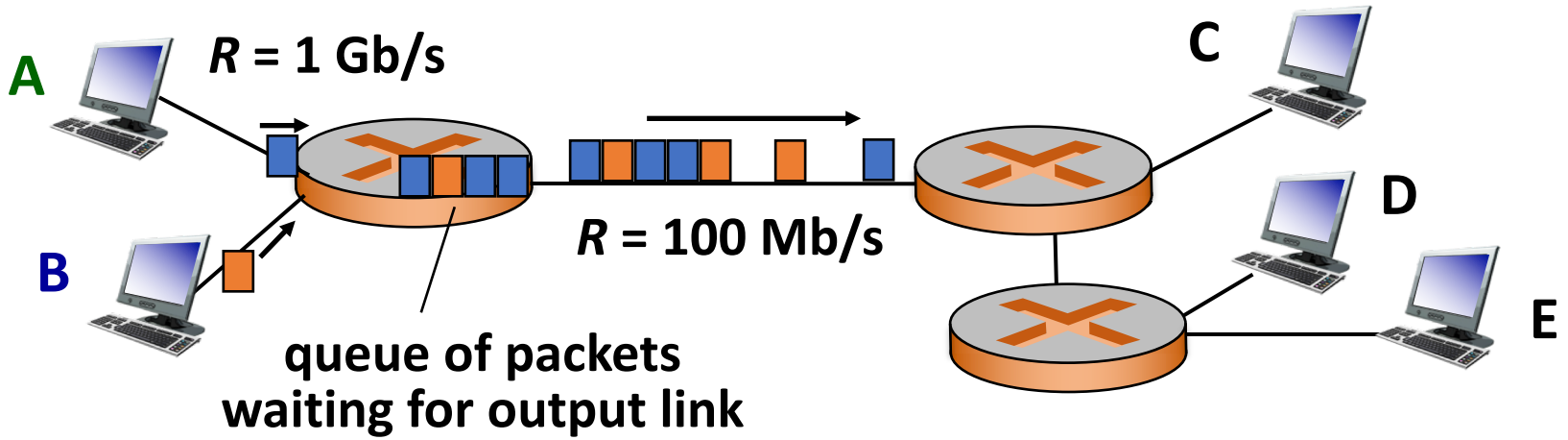
- Which property do you associate with circuit switching vs. packet switching? (Why?)
  - Better resource utilization (packet switching)
  - Guaranteed bandwidth performance (circuit switching)
  - Easier to implement (depends)
  - Less end-to-end delay variation (circuit switching)
  - Fairer (depends)
  - More robust under failure (packet switching)
  - Cheaper (depends)

Coordination Tradeoff

- Circuit switching vs. packet switching is example of engineering tradeoff
  - Coordination vs. best-effort
- Circuit switching requires more coordination, but provides better guarantees
  - Call setup necessary to reserve resources
- Packet switching requires no coordination, but provides no guarantees
  - Any end-system can send to any other at any time

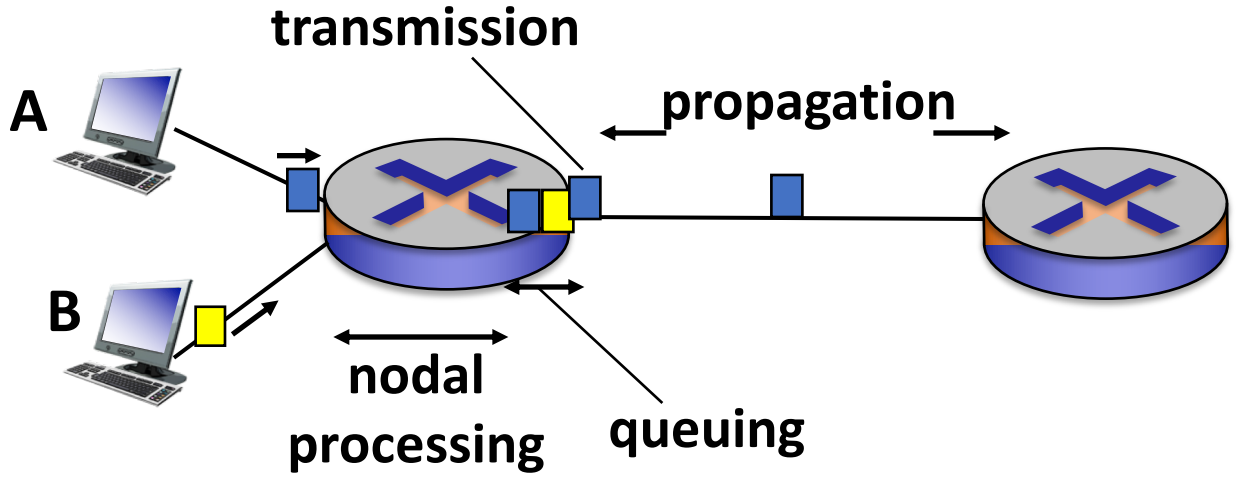
Packet Loss and Delays

- Packet switching can lead to packet loss and delays
- Traffic from A and from B have to share buffer on output link of router
  - Delay variation: depending on buffer level
  - Packet loss: if buffer is already full



Packet Delay

- Four sources of packet delay:



$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

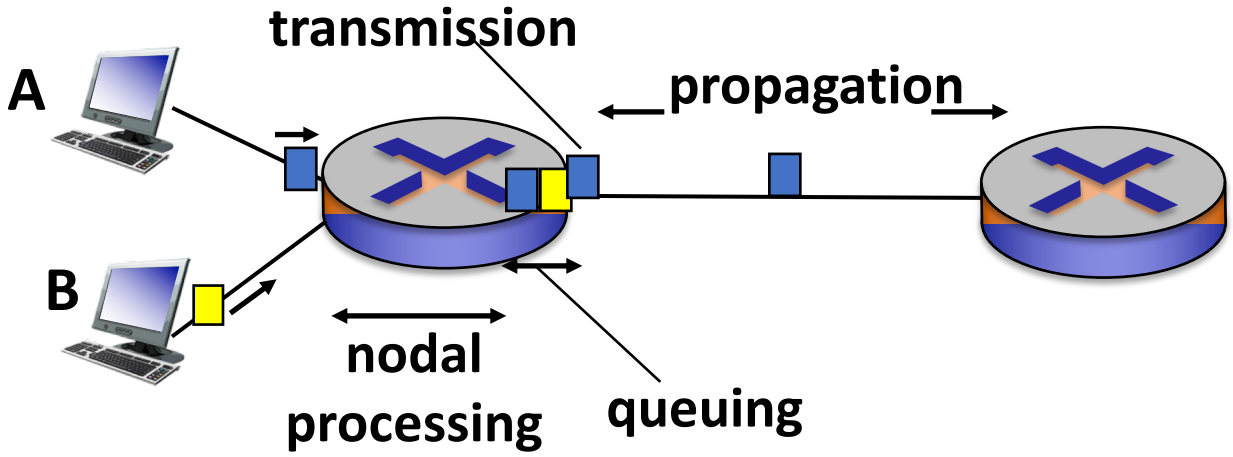
d_{proc}: nodal processing

- check bit errors
- determine output link
- typically < msec

d_{queue}: queuing delay

- time waiting at output link for transmission
- depends on congestion level of router

Packet Delay



$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

d_{trans}: transmission delay

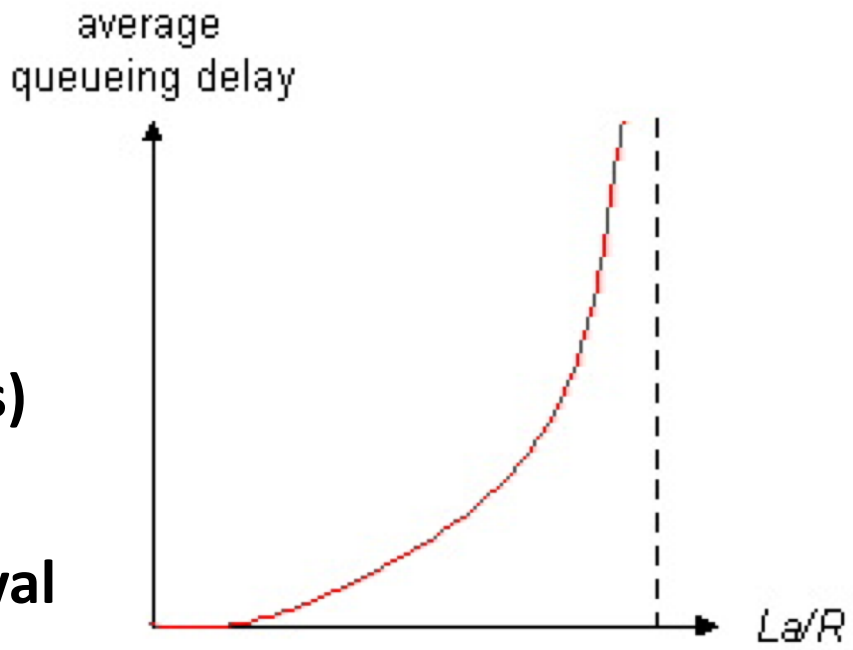
- L: packet length (bits)
- R: link *bandwidth (bps)*
- $d_{trans} = L/R$

d_{trans} and d_{prop} *very* different

d_{prop}: propagation delay

- d: length of physical link
- s: propagation speed (~2×10^8 m/sec)
- $d_{prop} = d/s$

Queuing Delay



- We will study queuing later in the course
  - *R:* link bandwidth (bps)
  - *L:* packet length (bits)
  - a: average packet arrival rate

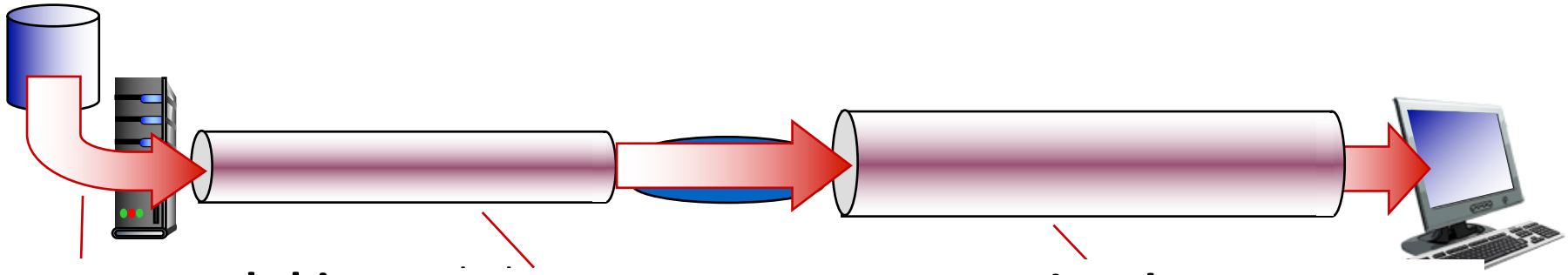
La/R ~ 0

La/R -> 1

- *La/R* ~ 0: avg. queueing delay small
- *La/R* -> 1: avg. queueing delay large
- *La/R* > 1: more "work" arriving than can be serviced, average delay infinite!

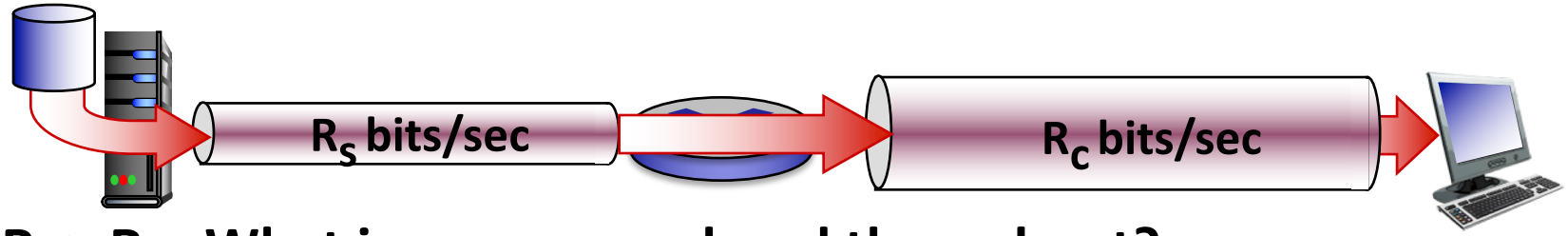
Throughput

- Throughput is rate (bits/time unit) at which data are transferred between sender/receiver
  - Instantaneous throughput: rate at given point in time
  - Average throughput: rate over longer period of time
- Fluid analogy:

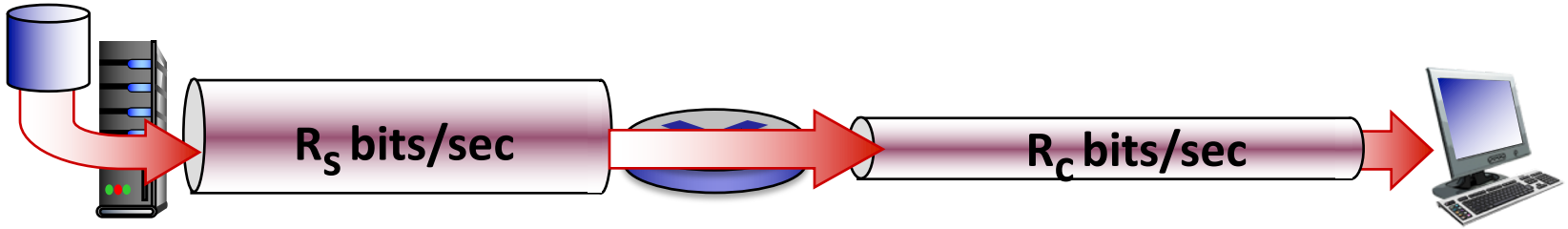


Throughput

- $R_s < R_c$: What is average end-end throughput?



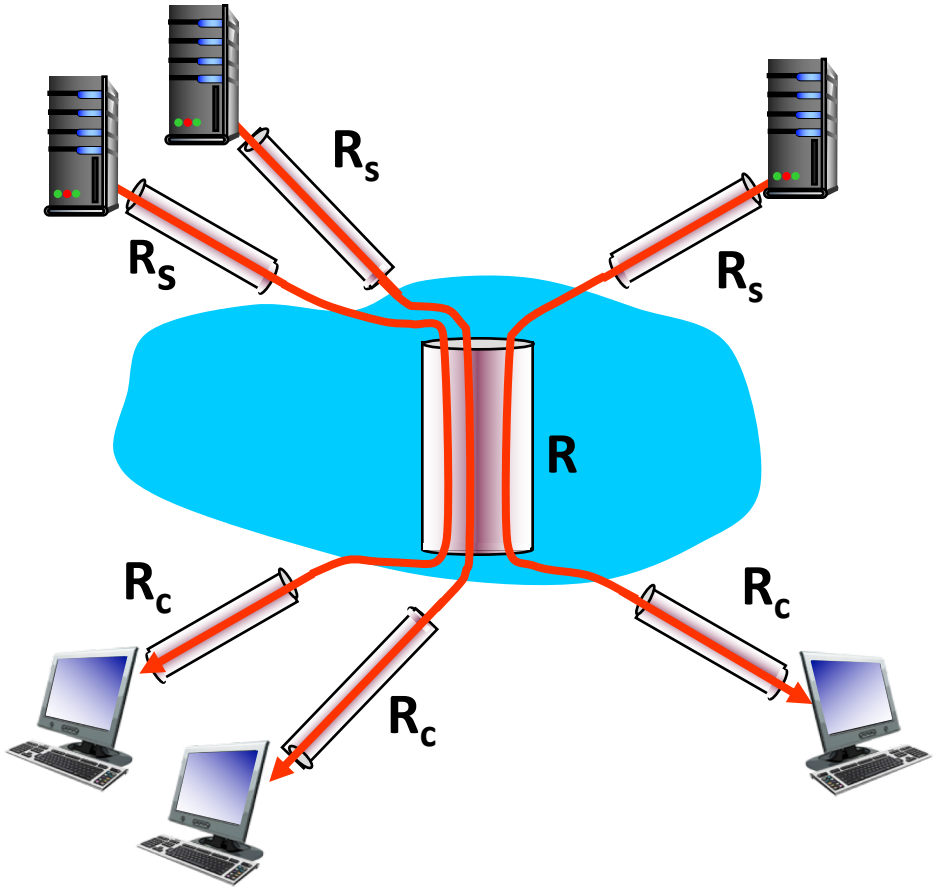
- $R_s > R_c$: What is average end-end throughput?



- Bottleneck link
  - Link on end-end path that constrains end-end throughput

Throughput and Fairness

- What if traffic from multiple connections share link?
  - Will fair sharing occur?
- No answer (yet)
  - We will see this again in congestion control and fair scheduling



Topics

- **Introduction**
  - **Lessons 1 & 2**
- **Application Layer Protocols (HTTP, SMTP)**
  - **Lesson 3**
- **Transport Layer Protocols (TCP, UDP)**
  - Lesson 4
- **Transport Layer (Congestion Control)**
  - Lesson 5
- **Network Layer Protocol (IP)**
  - Lesson 6
- **Data Link Layer Protocol (Ethernet)**
  - Lesson 7
- **Software Defined Networks (SDN)**
  - Lesson 8

Review of Specific Protocols

- **We will briefly review the top four protocols**
- **For full details**
  - **Textbook**
  - **Requests for comments (RFCs)**

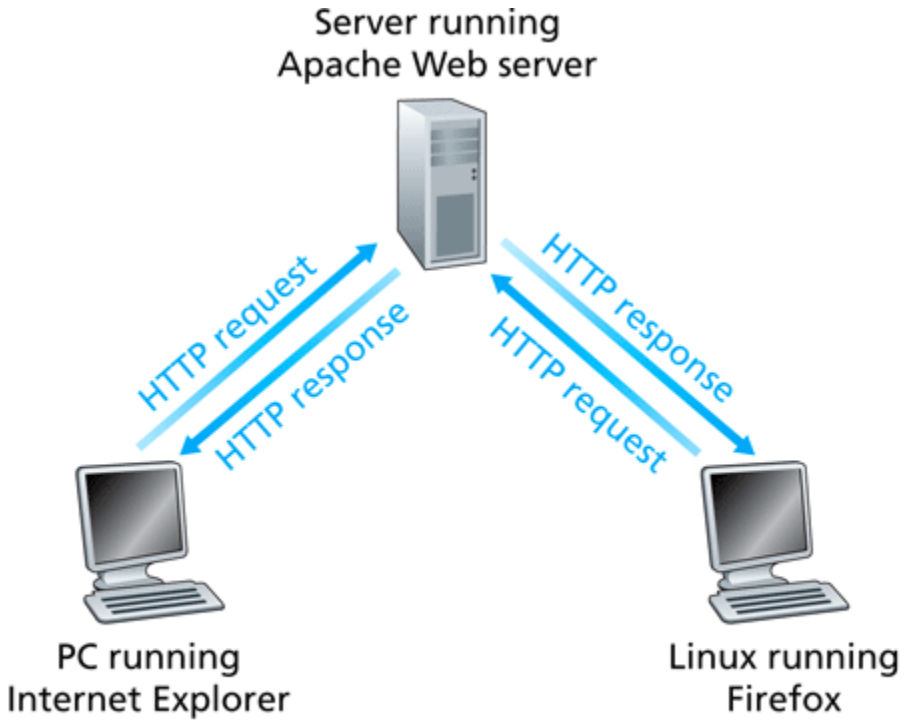
Layer	Example protocols
Application layer	Hypertext Transfer Protocol (HTTP)
Transport layer	Transmission Control Protocol (TCP)
Network layer	Internet Protocol (IP)
Link layer	Ethernet
Physical layer	1000BASE-T

Application Layer

- Client-server architecture (most common)
  - Server
    - Provides service (e.g., access to data)
    - Is always on and waits for requests
    - Has well known address
  - Client
    - Initiates communication with server
    - Often determines what is communicated
    - Often unknown to server
  - Communication via “socket”
- Peer-to-peer architecture
  - End-systems act as clients and servers
  - Indexing system figures out which end-system to connect to

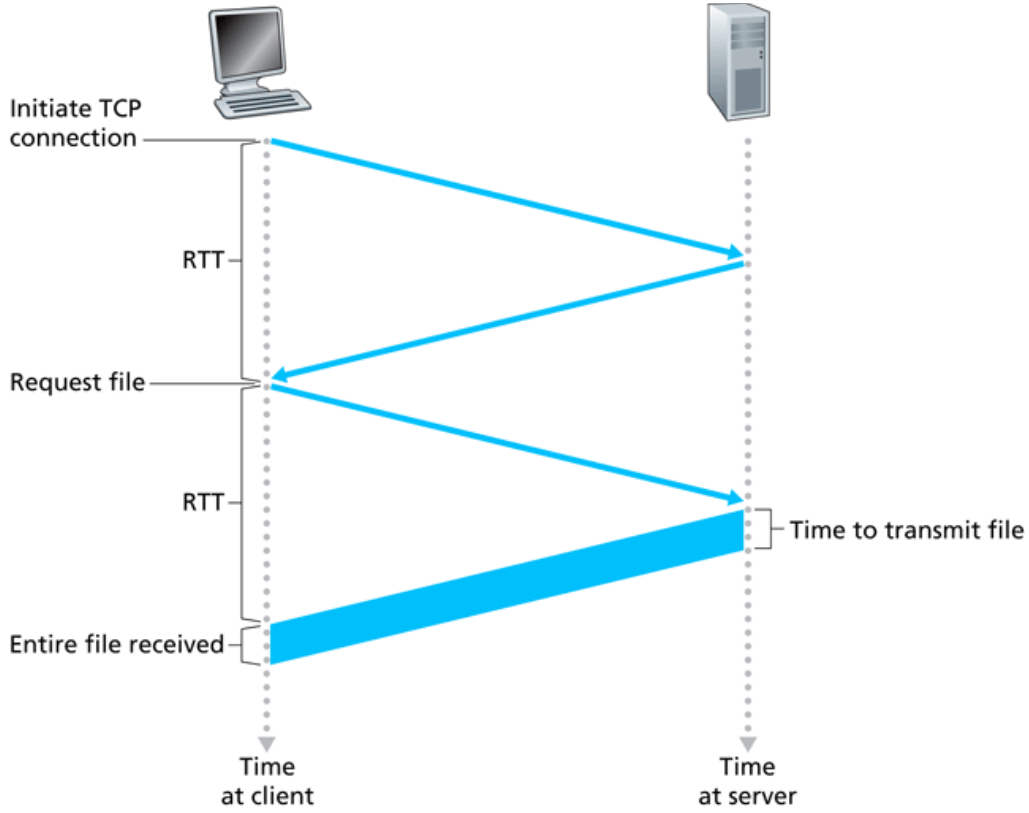
Hypertext Transfer Protocol

- Web browser and server use protocol to exchange web documents
  - Hypertext Transfer Protocol (HTTP)
- HTTP uses plain text for protocol exchange
  - Easy for humans to understand
- Remember: protocol
  - Definition of message format
    - Types of messages
    - Syntax of messages (fields and delineation)
    - Semantics of fields
  - Definition of message exchange
    - When and how to send messages
    - When and how to respond



Hypertext Transfer Protocol

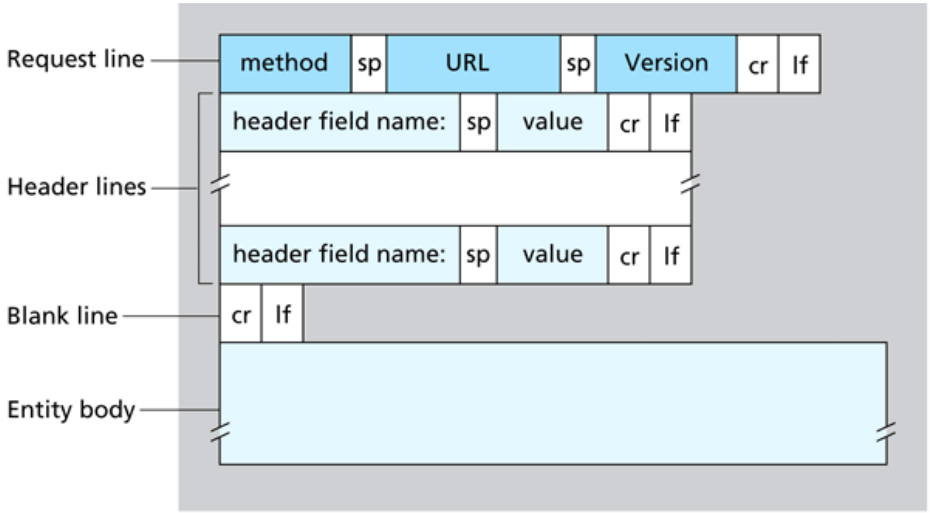
- Sequence of messages
  1. Establish connection (transport layer and below)
  2. Send request for document
  3. Receive document
  4. Repeat 2.-3. or close connection
- Format of messages
  - Plain text



Hypertext Transfer Protocol

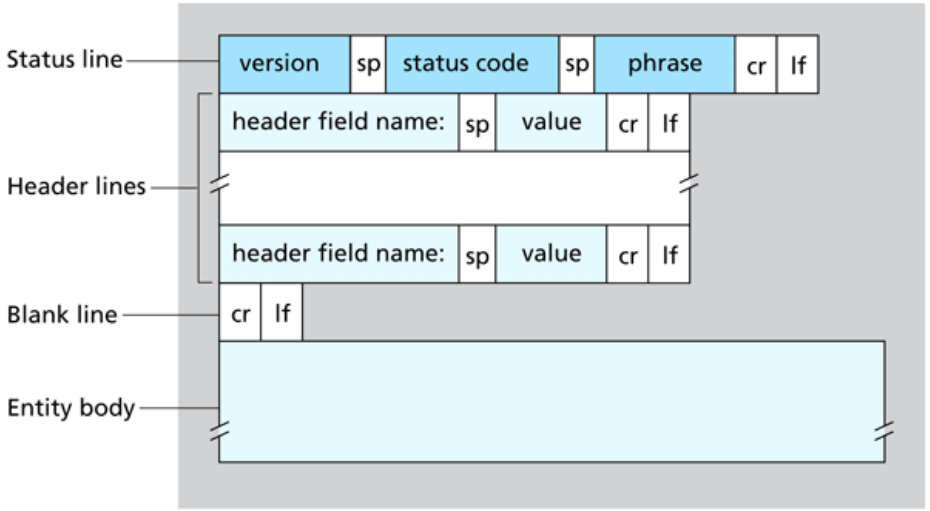
- **Example:**

Request




```
GET / HTTP/1.1
Host: engineering.umass.edu
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X
10_12_3) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/56.0.2924.87 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,
image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
```

Response

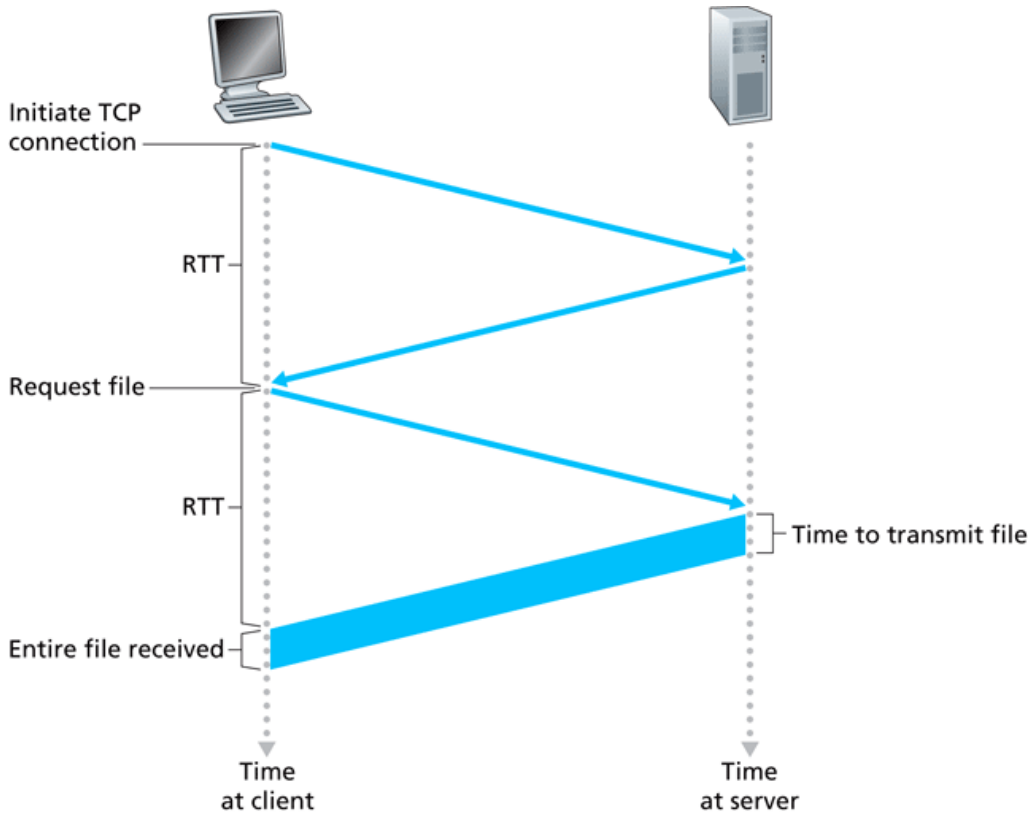



```
HTTP/1.1 200 OK
Server: Apache
Content-Language: en
Cache-Control: public, max-age=180
Last-Modified: Thu, 09 Feb 2017 23:18:58 GMT
Expires: Sun, 19 Nov 1978 05:00:00 GMT
Vary: Cookie,Accept-Encoding
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Content-Length: 10656
Accept-Ranges: bytes
Date: Fri, 10 Feb 2017 00:17:19 GMT
Connection: keep-alive

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; …
```

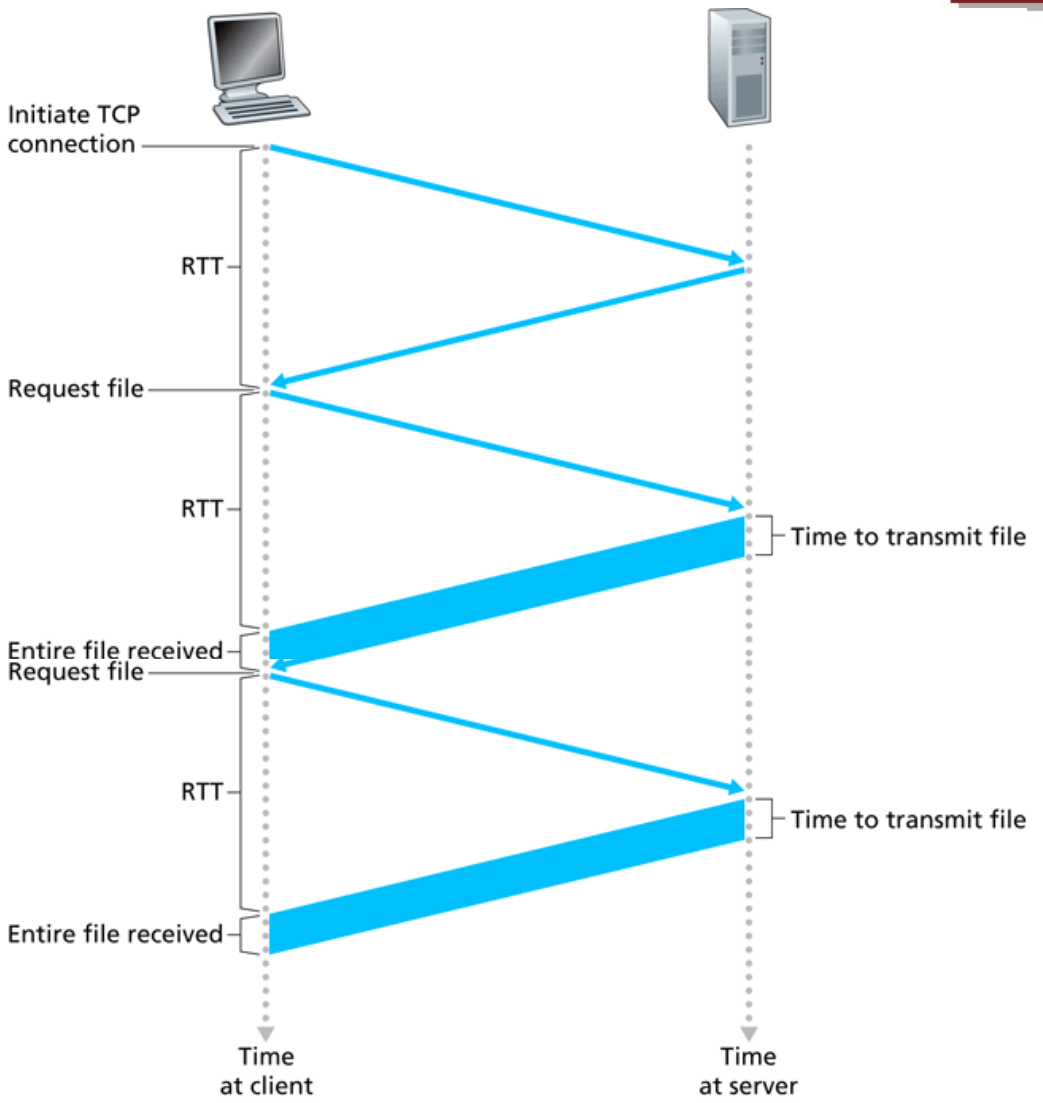
Hypertext Transfer Protocol

- What happens in network when requesting document?
  - Transport layer establishes connection
    - Requires one round-trip time (RTT)
  - HTTP requests document
    - Requires one RTT plus transmission time



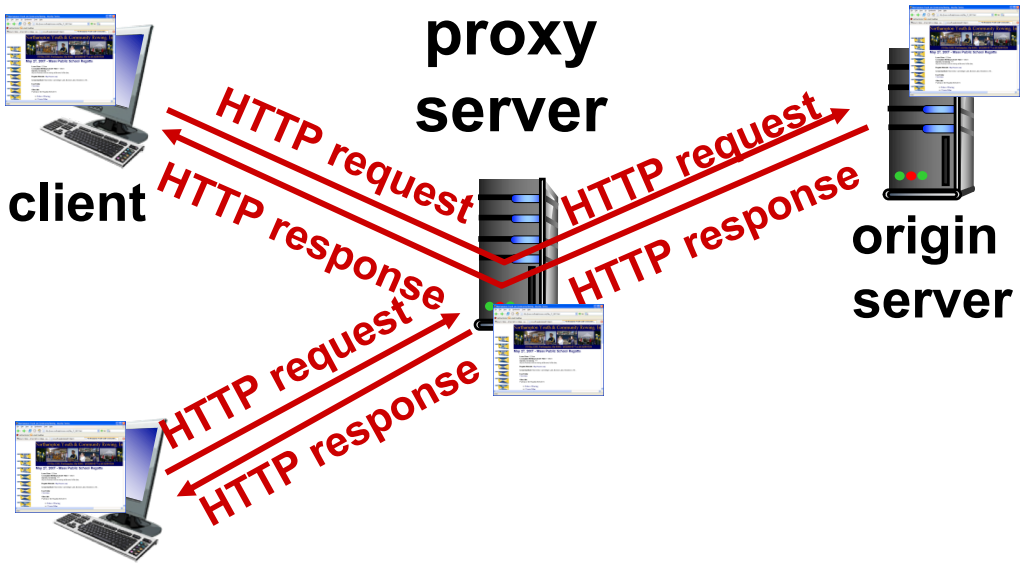
Persistent Connections

- Persistent HTTP connection reuses connection
  - Multiple files on same connection
  - “Pay” for RTT only once
- Example of how protocol design can affect performance



Content Caching

- Reduce RTT by fetching content from “closer” server
- Caching proxy
  - Keeps copy of fetched web documents
  - Serves content when other local users request document
- Caching works well for frequently requested content

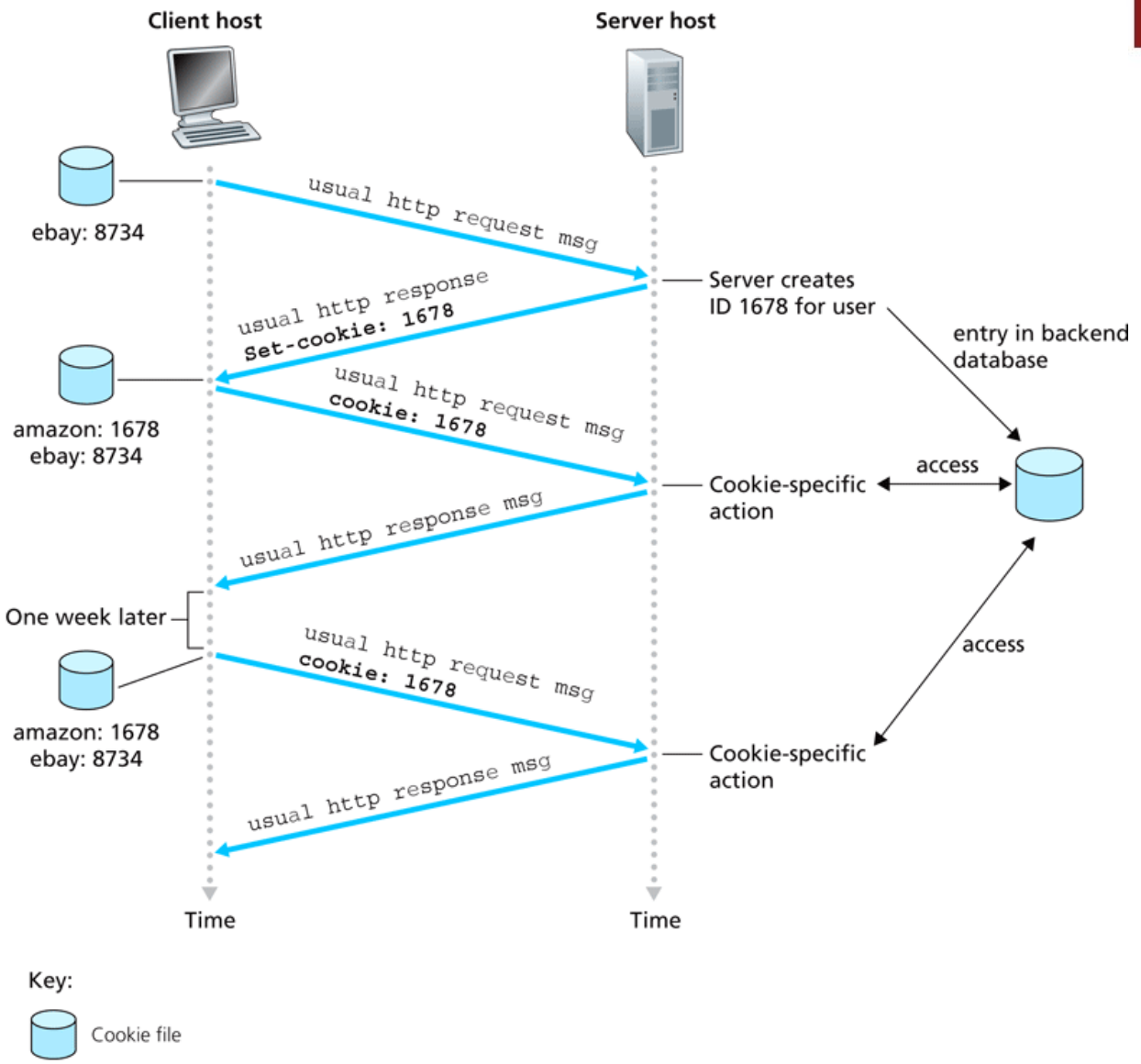


User State in HTTP

- HTTP is a “stateless” protocol
  - Any request is independent from any other
  - In practice, state is necessary for user experience
- How can a shopping site remember what I have in my shopping cart?
  - Identify browser / session with server-provided id
  - Associate shopping cart with session

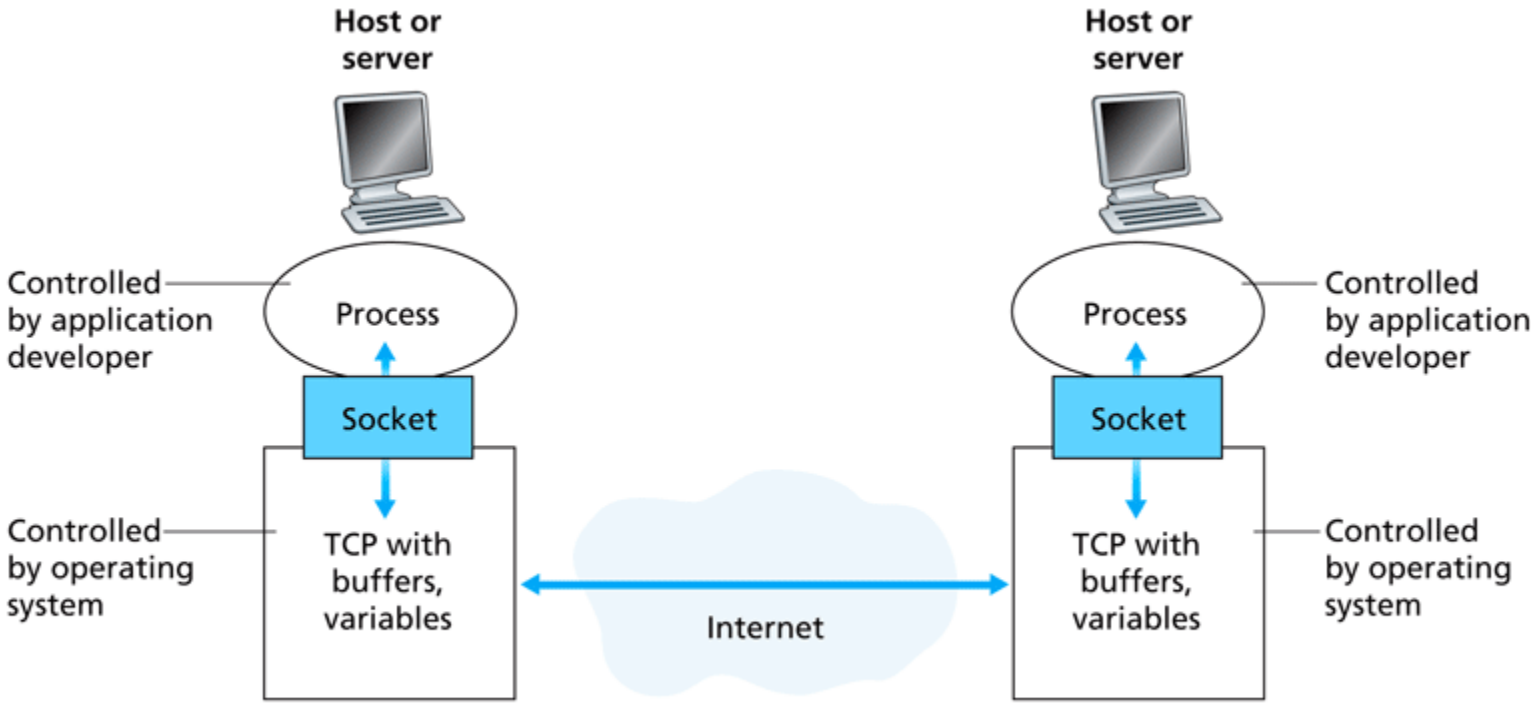
Connected Device: User State in HTTP

- Cookies in HTTP
  - Identifier that is sent with every request



Application Layer

- Connection provided layer below (transport layer):
  - UNIX socket (="bit pipe")
  - "telnet" is one protocol that provides this functionality



Transport Layer Service

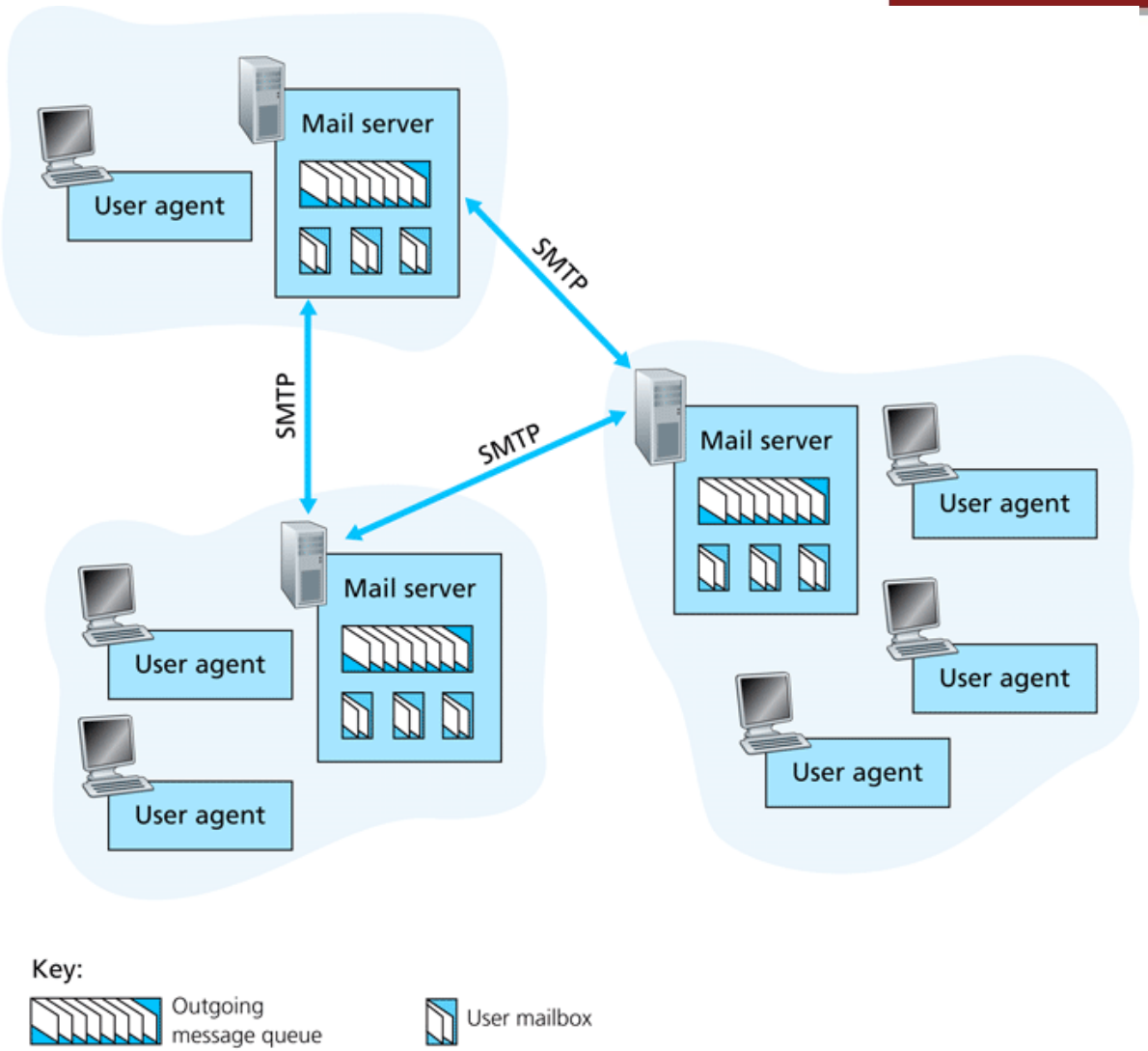
- TCP service:
  - reliable transport between sending and receiving process
  - flow control: sender won't overwhelm receiver
  - congestion control: throttle sender when network overloaded
  - does not provide: timing, minimum throughput guarantee, security
  - connection-oriented: setup required between client and server processes
- UDP service:
  - unreliable data transfer between sending and receiving process
  - does not provide: reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup
- Why bother with UDP at all?

Application Requirements

Application	Data Loss	Bandwidth	Time-Sensitive
File transfer	No loss	Elastic	No
E-mail	No loss	Elastic	No
Web documents	No loss	Elastic (few kbps)	No
Internet telephony/ Video conferencing	Loss-tolerant	Audio: few kbps–1Mbps Video: 10 kbps–5 Mbps	Yes: 100s of msec
Stored audio/video	Loss-tolerant	Same as above	Yes: few seconds
Interactive games	Loss-tolerant	Few kbps–10 kbps	Yes: 100s of msec
Instant messaging	No loss	Elastic	Yes and no

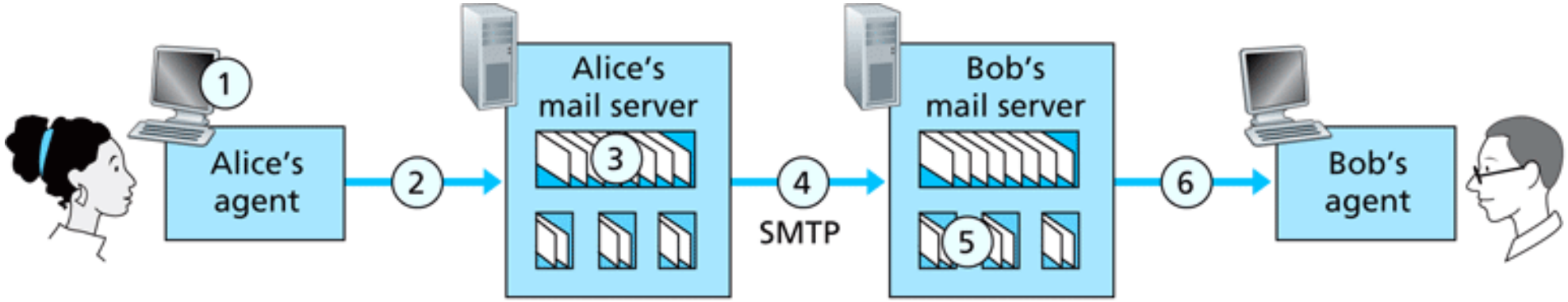
Electronic Mail

- Mail servers
  - Transfer mail
  - Store mail in
  - mailboxes
- User agents
  - Access mail from server
  - Transmit new mail
- Protocols
  - Simple Mail Transfer Protocol (SMTP)
  - Post Office Protocol (POP)
  - Internet Mail Access Protocol (IMAP)
  - Web-based email access (HTTP)



Electronic Mail

- Sending of email:



- SMTP "pushes" email to destination mail server
- POP/IMAP/ HTTP "pulls" email from mail server
- SMTP messages are in cleartext

Topics

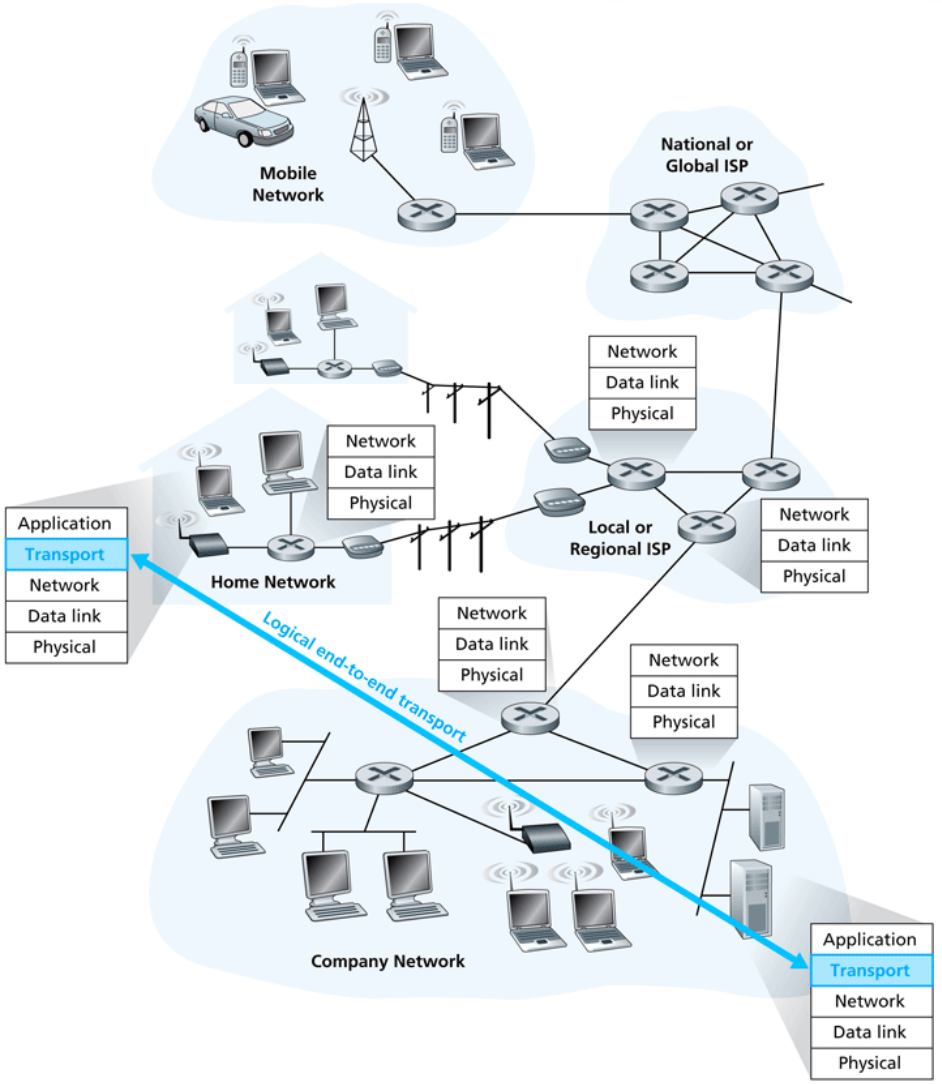
- **Introduction**
  - **Lessons 1 & 2**
- **Application Layer Protocols (HTTP, SMTP)**
  - Lesson 3
- **Transport Layer Protocols (TCP, UDP)**
  - Lesson 4
- **Transport Layer (Congestion Control)**
  - Lesson 5
- **Network Layer Protocol (IP)**
  - Lesson 6
- **Data Link Layer Protocol (Ethernet)**
  - Lesson 7
- **Software Defined Networks (SDN)**
  - Lesson 8

Securing TCP

- TCP & UDP
  - No encryption
  - Cleartext passwords sent into socket traverse Internet in cleartext
- Secure Socket Layer (SSL)
  - provides encrypted TCP connection
  - data integrity
  - end-point authentication
- SSL is at app layer
  - Apps use SSL libraries that “talk” to TCP
- SSL socket API
  - Cleartext passwords sent into socket traverse Internet encrypted
  - More later in course

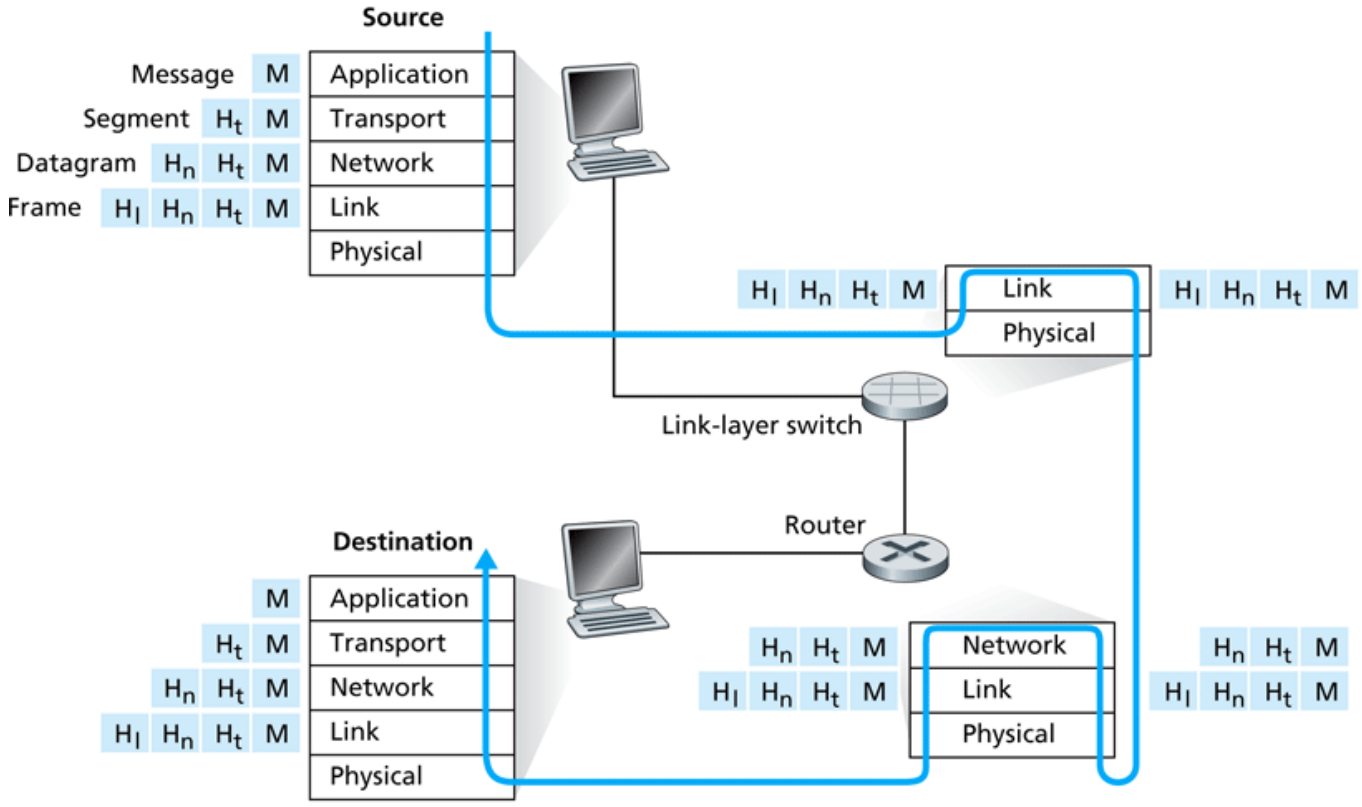
Transport Layer

- Transport layer provides logical communication
  - Service to layer above
    - Connects applications
  - Uses network layer
    - Data is “segmented” and encapsulated in packet
- Services
  - Transport-layer multiplexing and demultiplexing
    - Delivery to correct process
  - User Datagram Protocol (UDP)
    - Send and hope for the best
  - Transport Control Protocol (TCP)
    - Reliability
    - Flow control
    - Congestion control



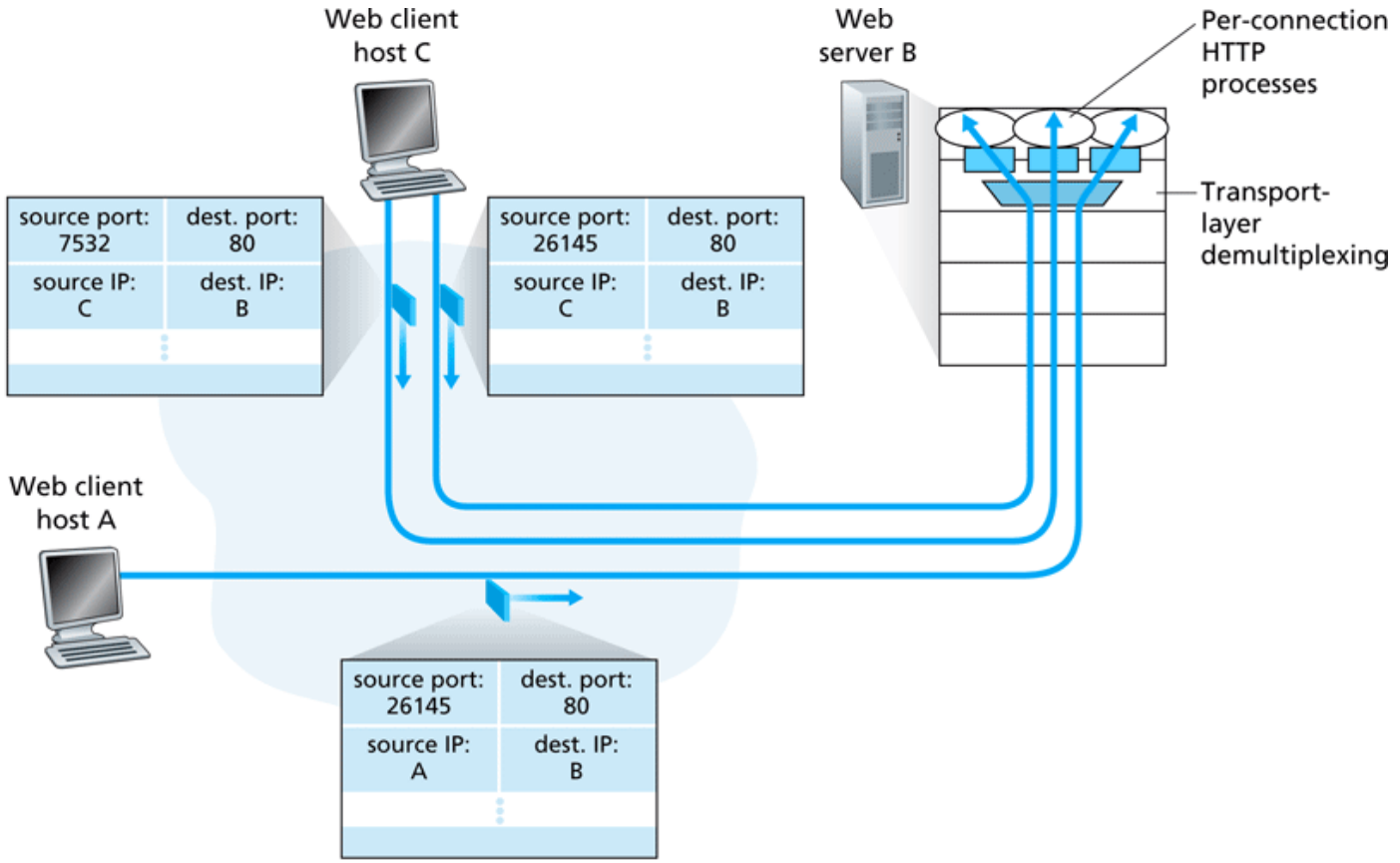
Layered Protocol Stack

- Recall naming of data in different layers
  - Message, segment, datagram, frame



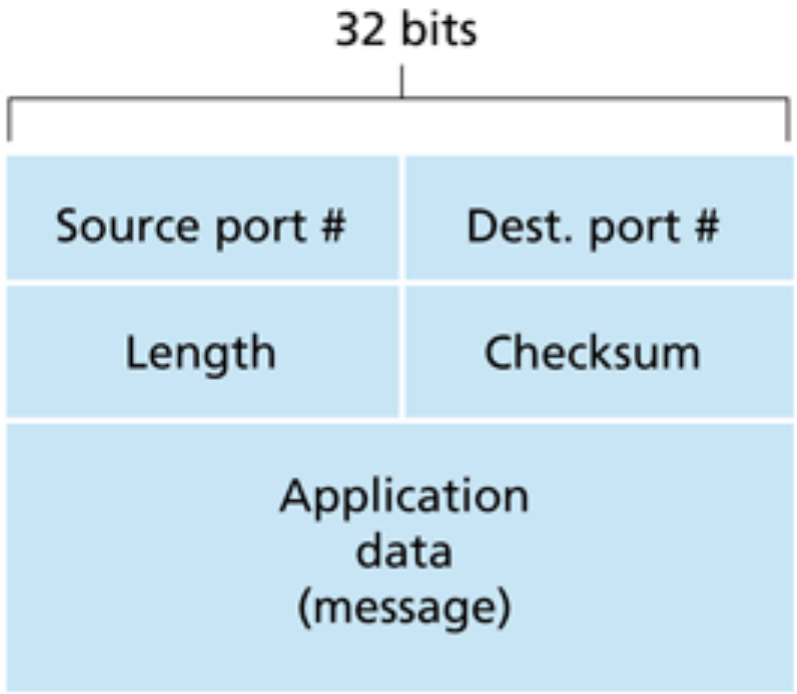
Document Cam: De-/Multiplexing

- **Works even if multiple connections go to the same port**



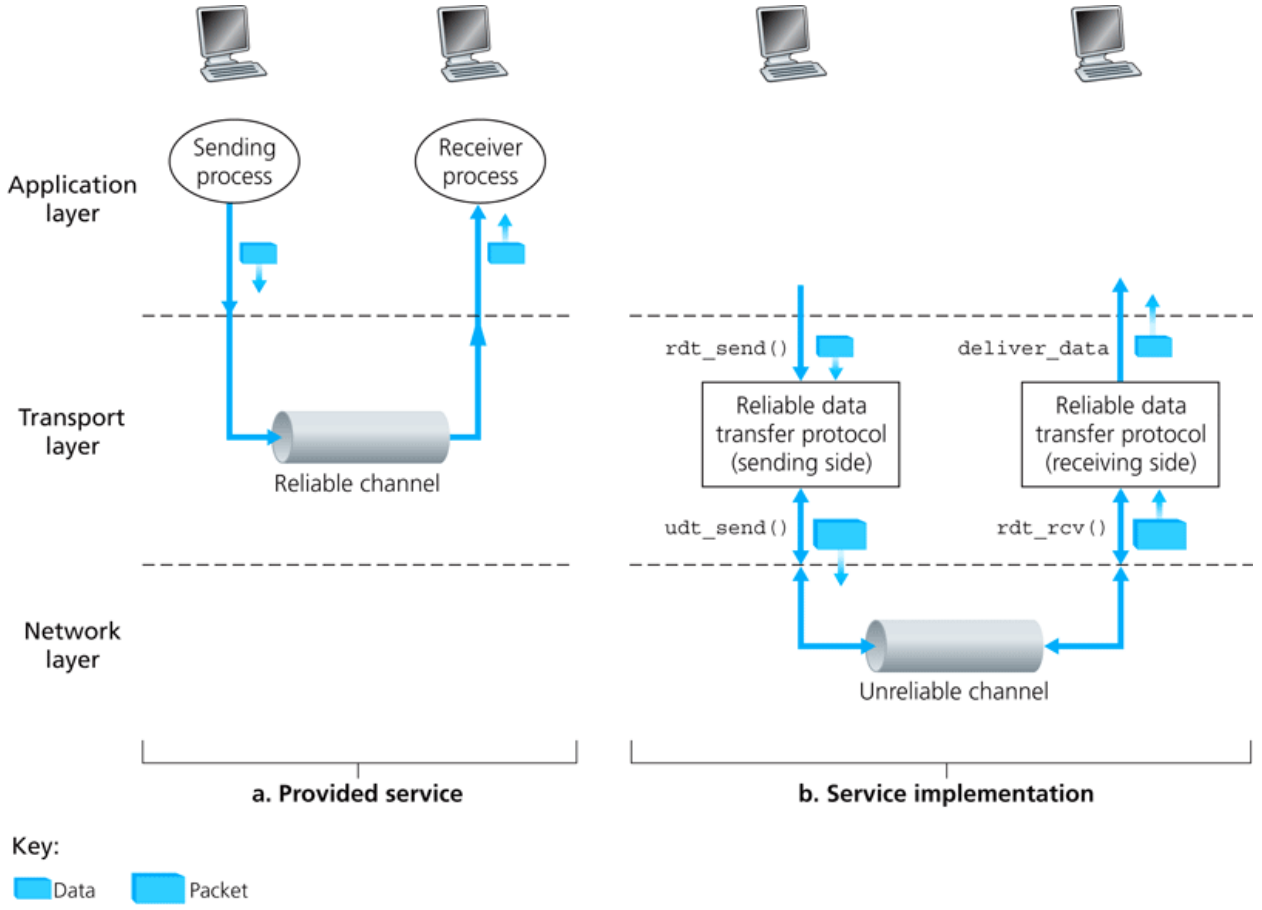
User Datagram Protocol

- Bare-bones transport protocol
  - Connectionless, state-free
  - Unreliable (because network layer is unreliable)
  - Low overhead
- UDP segment
  - Port numbers
  - Length field
  - Checksum field (optional!)
- UDP couldn't do less…



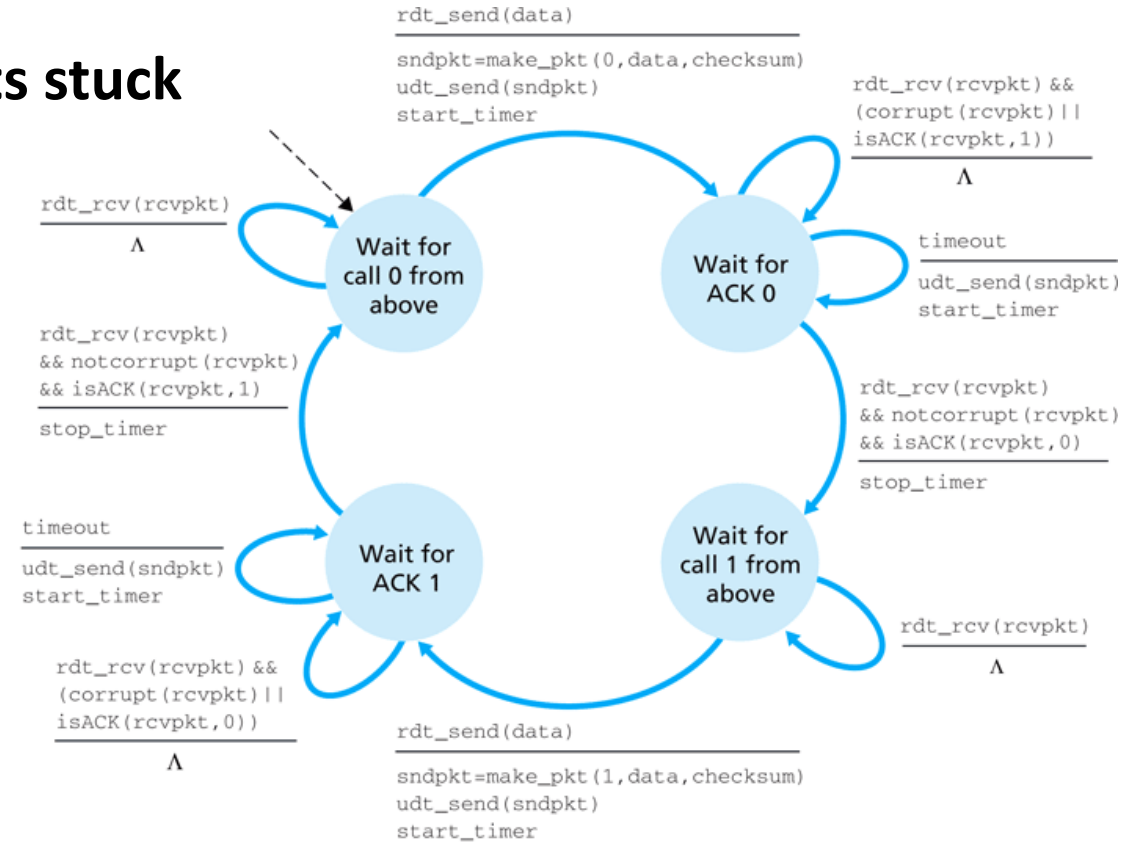
Reliable Data Transfer

- **Reliability is provided on top of unreliable channel**



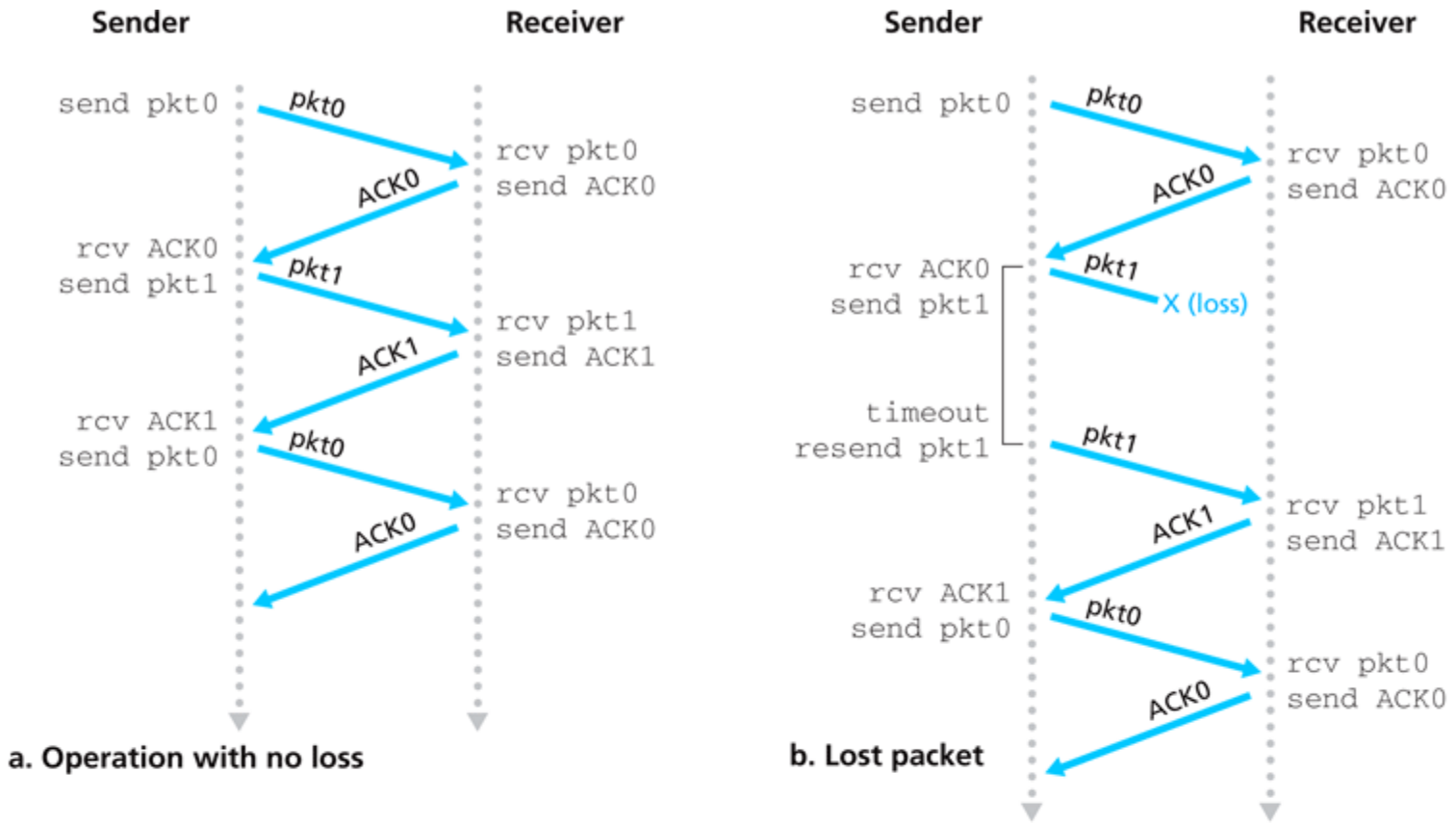
Whiteboard: Reliable Data Transfer

- Scenario 3: Channel with packet loss and bit errors
- What is the problem?
  - If packets get lost, transfer gets stuck
- Timer necessary
  - Starts on transmission
  - Retransmission on expiration
  - Sequence bit maintains order
- “Alternating-bit protocol”



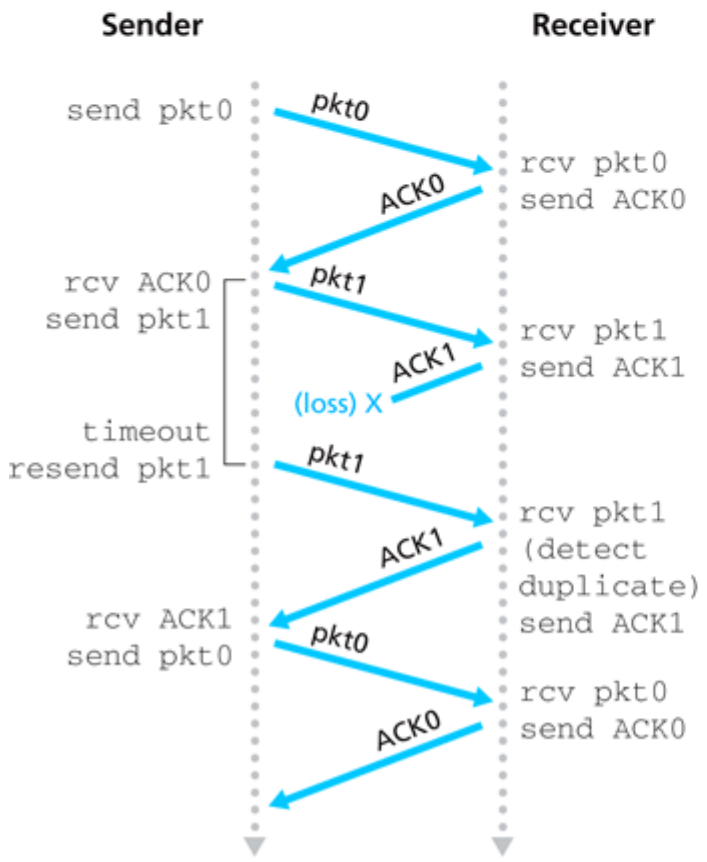
Alternating-Bit protocol

- Error scenarios are handled correctly

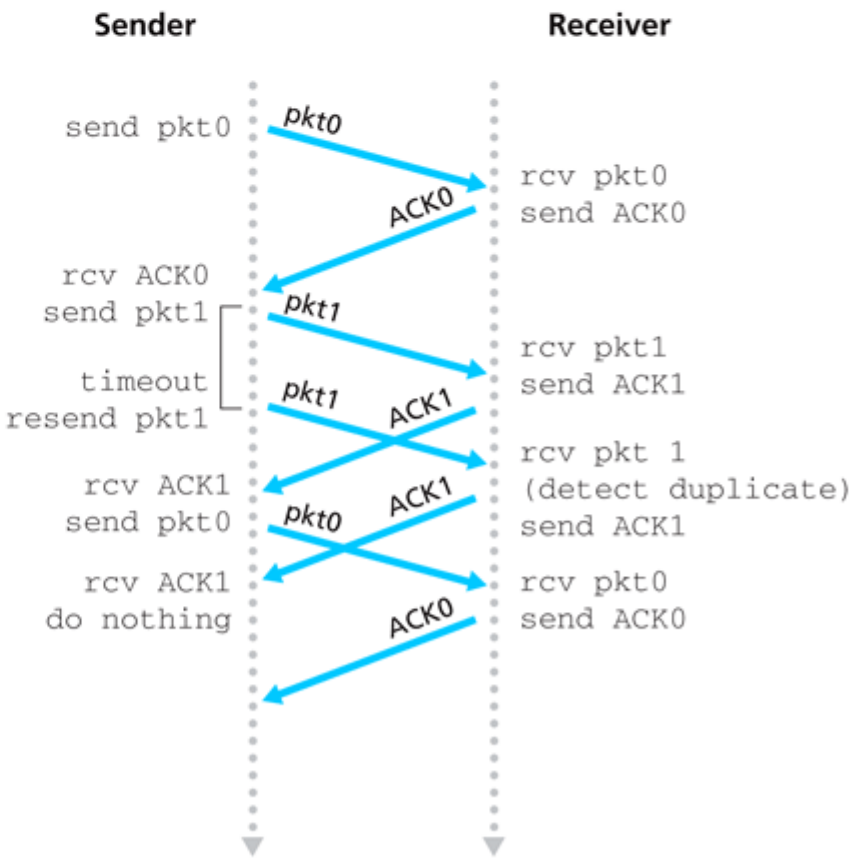


Alternating-Bit Protocol

- **Error scenarios are handled correctly**



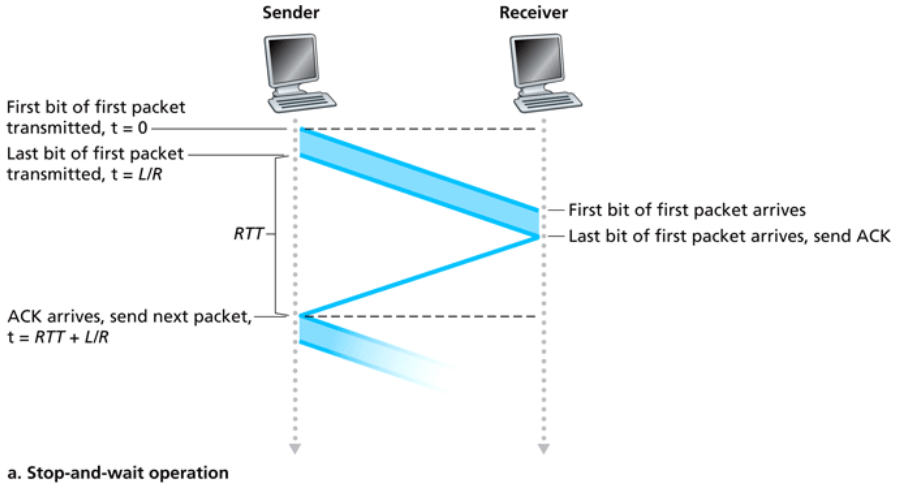
c. Lost ACK



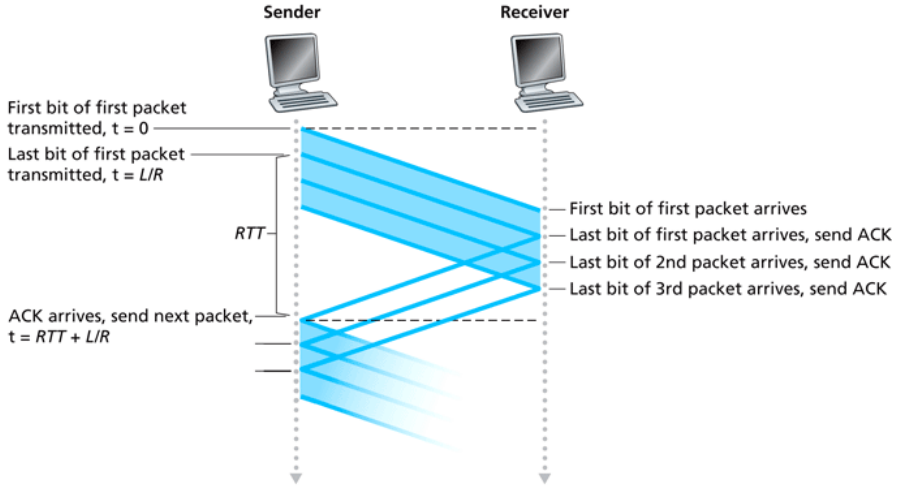
d. Premature timeout

Pipelined Transfer

- Multiple packets can be in-flight
  - Sliding window protocol
- What needs to change?
  - More sequence numbers
  - Larger buffers (retransmission or reassembly)
- Two versions:
  - Go-back-n
  - Selective Repeat



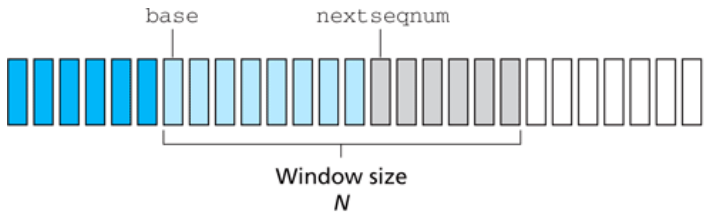
a. Stop-and-wait operation

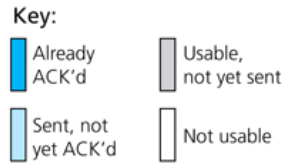


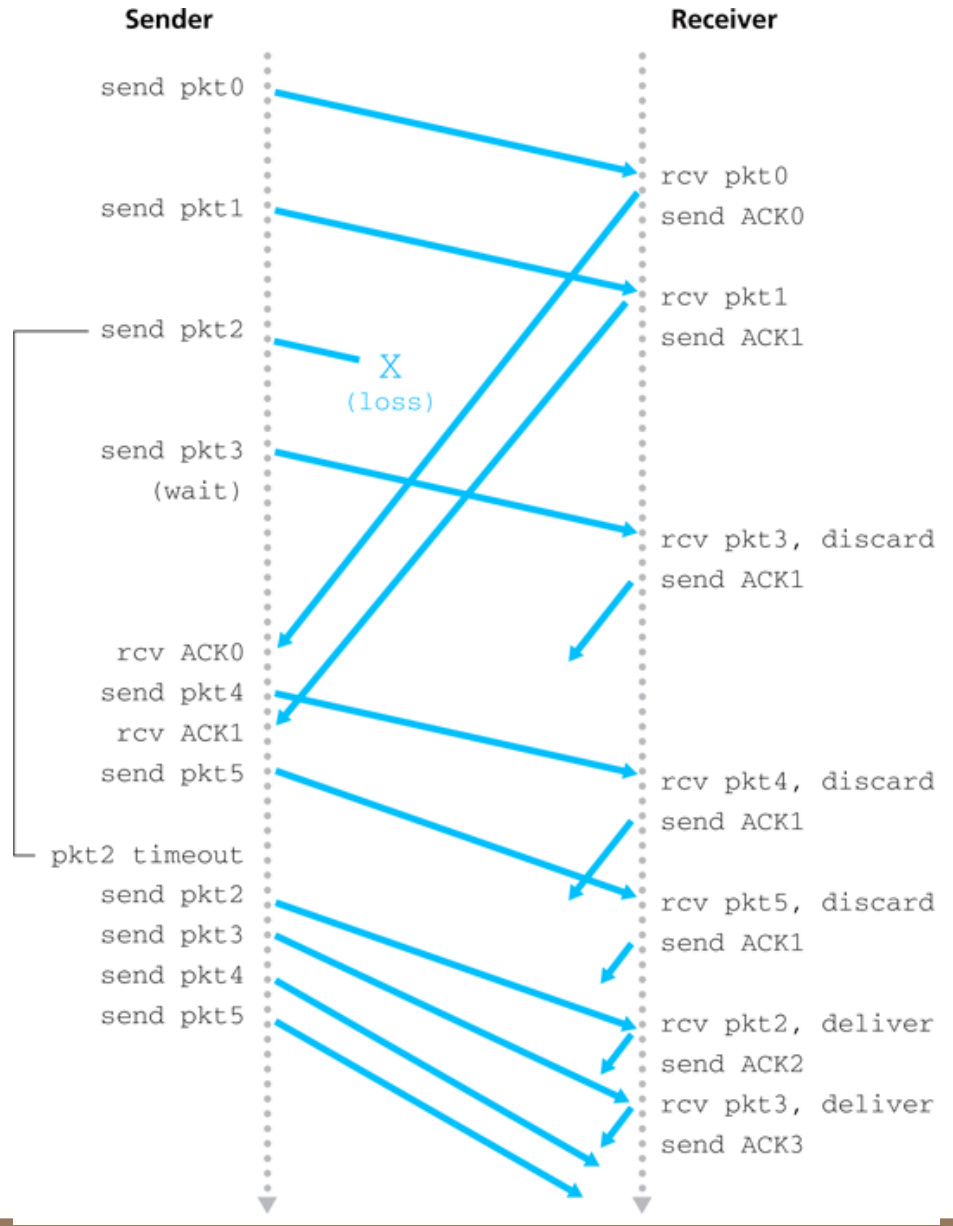
b. Pipelined operation

Go-Back-N

- Characteristics
  - ACK-based
  - Cumulative acknowledgement
  - On timeout, all outstanding packets are resent

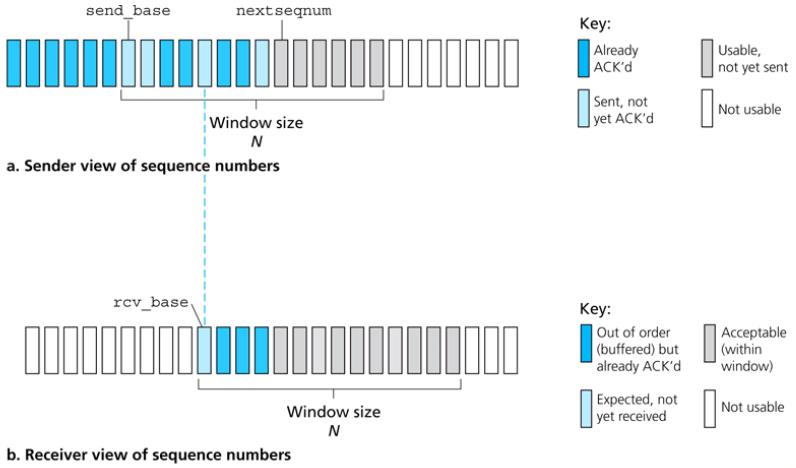


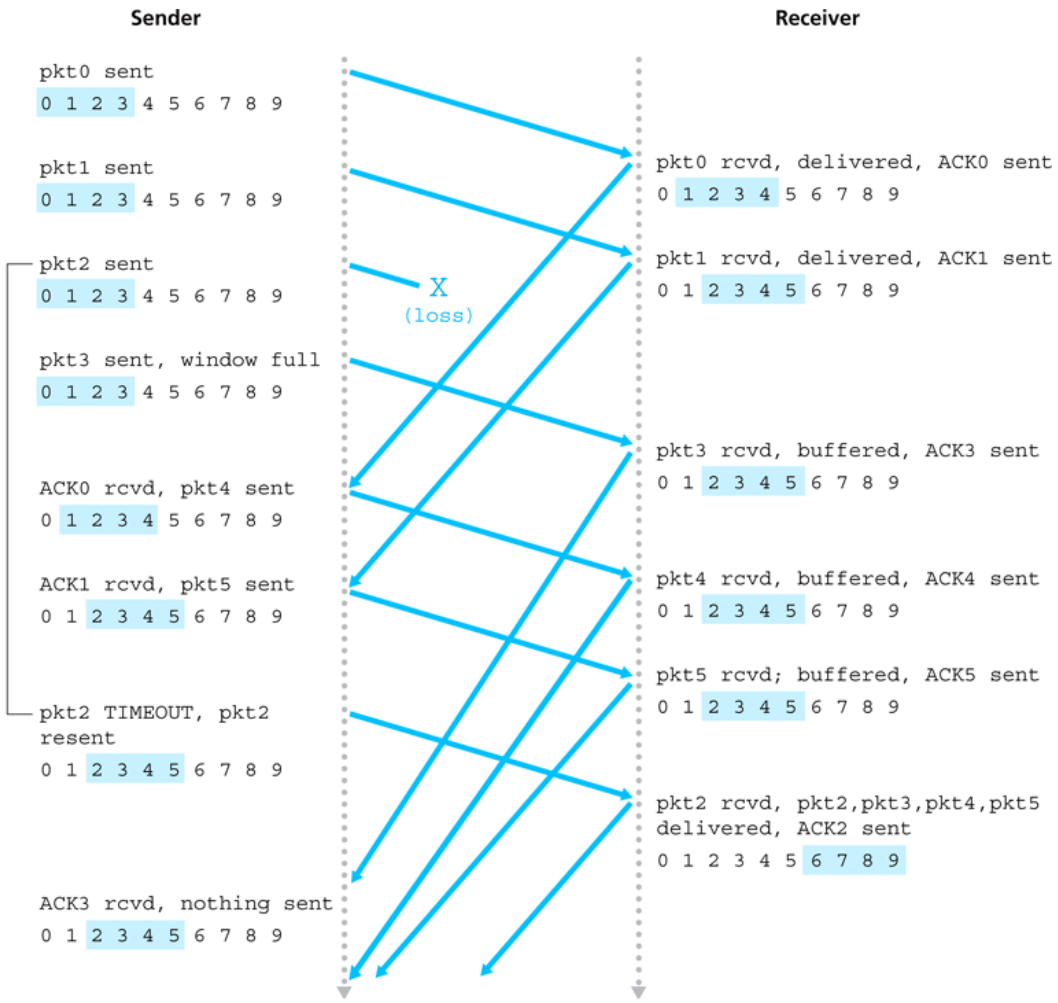




Selective Repeat

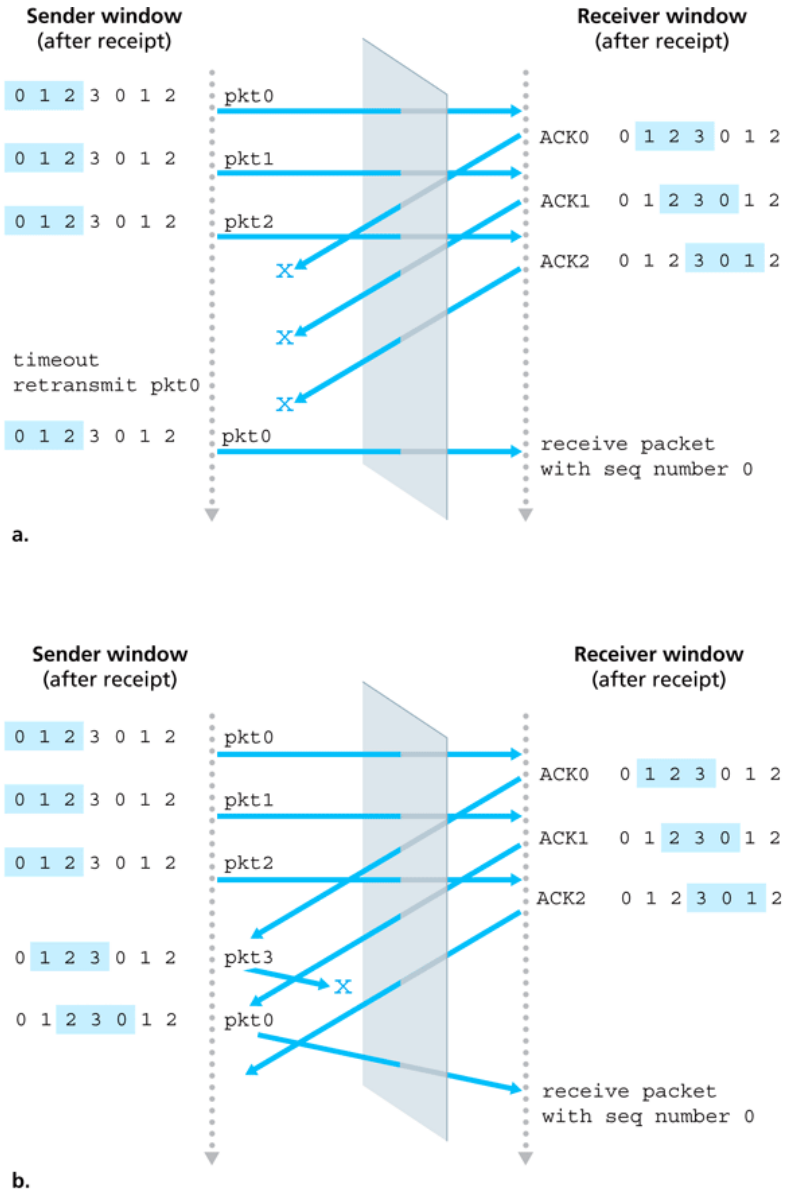
- Characteristics
  - ACK-based
  - Selective acknowledgements
  - On timeout, only unacknowledged packets are resent





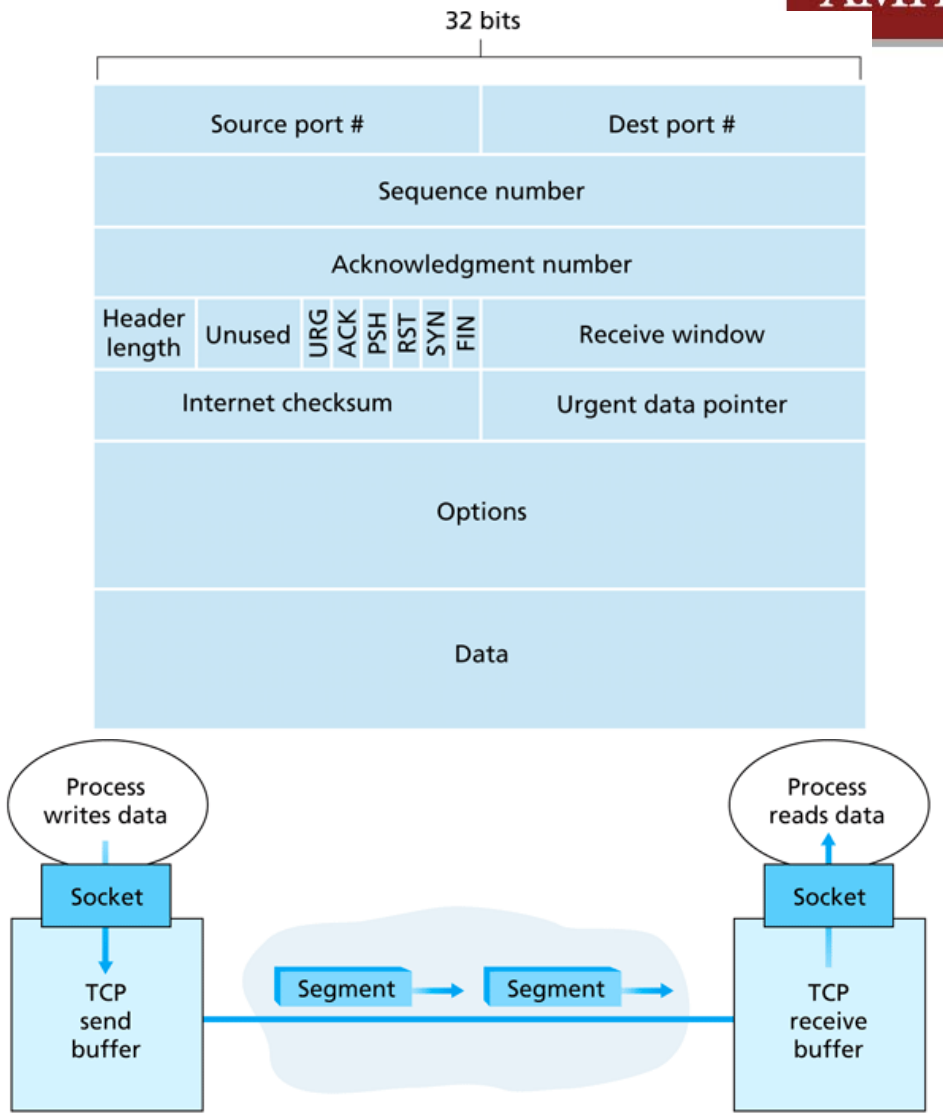
Sequence Numbers

- Sequence numbers are finite
  - Need to be carried in each packet
- Problem if window is too large
  - Packets with reused sequence numbers cannot be distinguished from originals
- Problem if indefinite packet lifetime is allowed
  - New packet cannot be distinguished from retransmission
  - Internet: ~3 minutes max



Transmission Control Protocol

- Implementation of sliding window protocol
  - Cumulative ACKs
  - ACKs are piggy-backed on data packets in other direction
  - If out-of-order segment
    - Option 1: discard
    - Option 2: wait and see if other segments show up

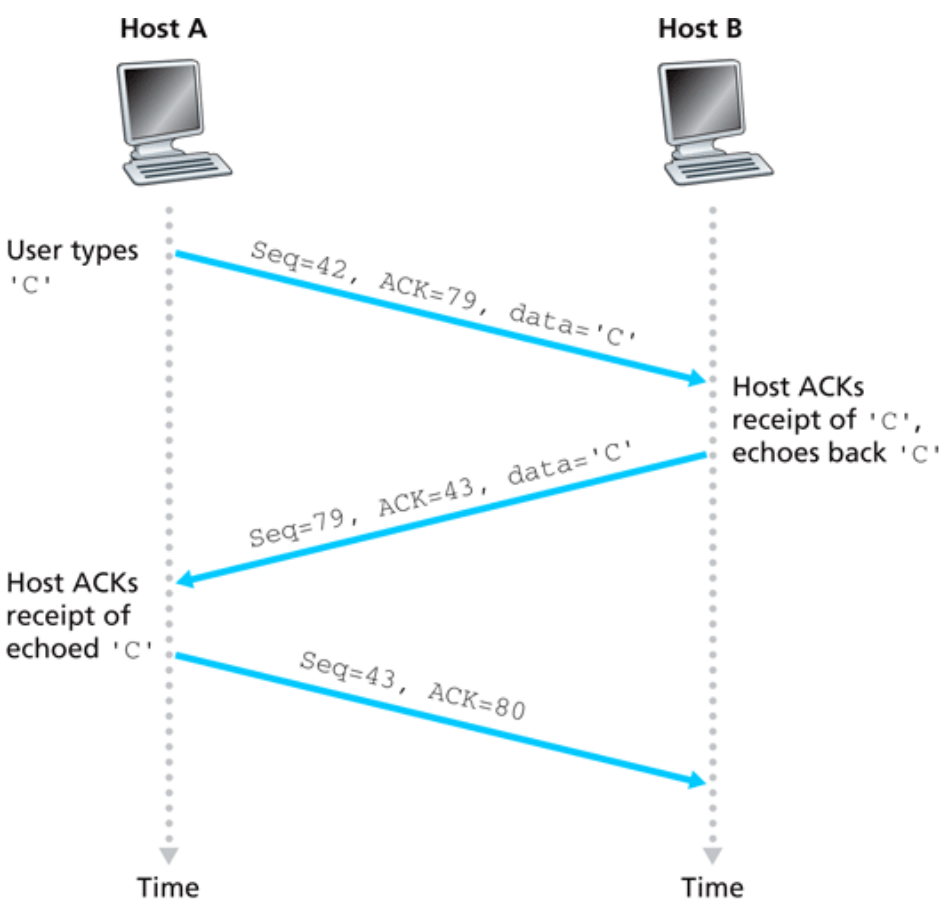


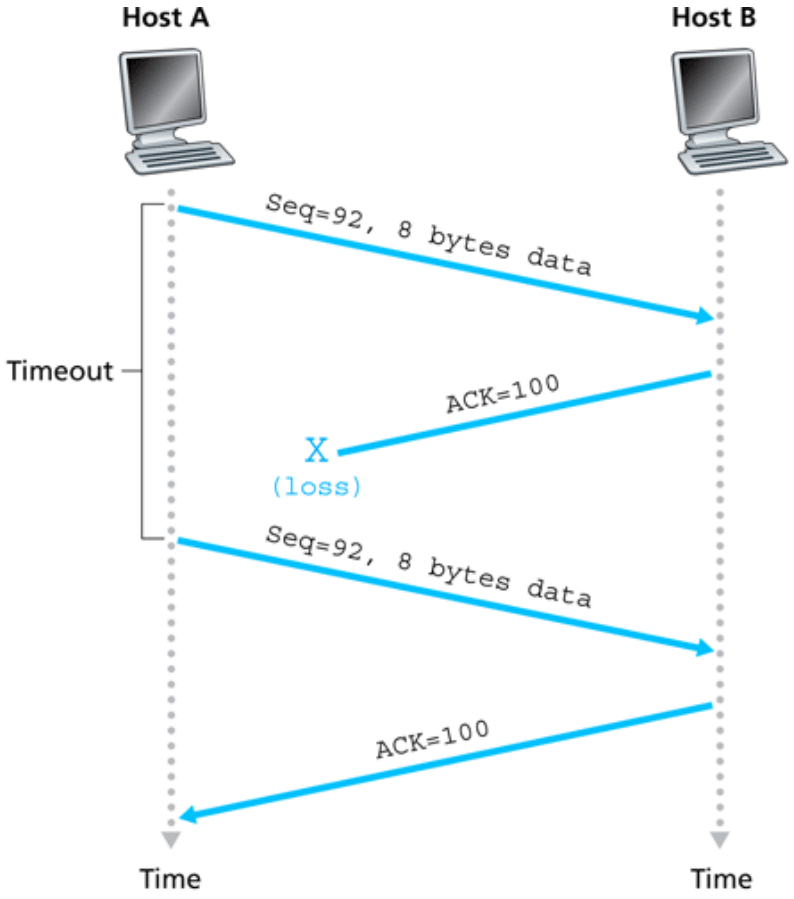
TCP Sequence Numbers

- Sequence numbers are position in byte-stream
  - 16bits = 64kByte window
- Acknowledgement number is next expected byte

TCP Operation

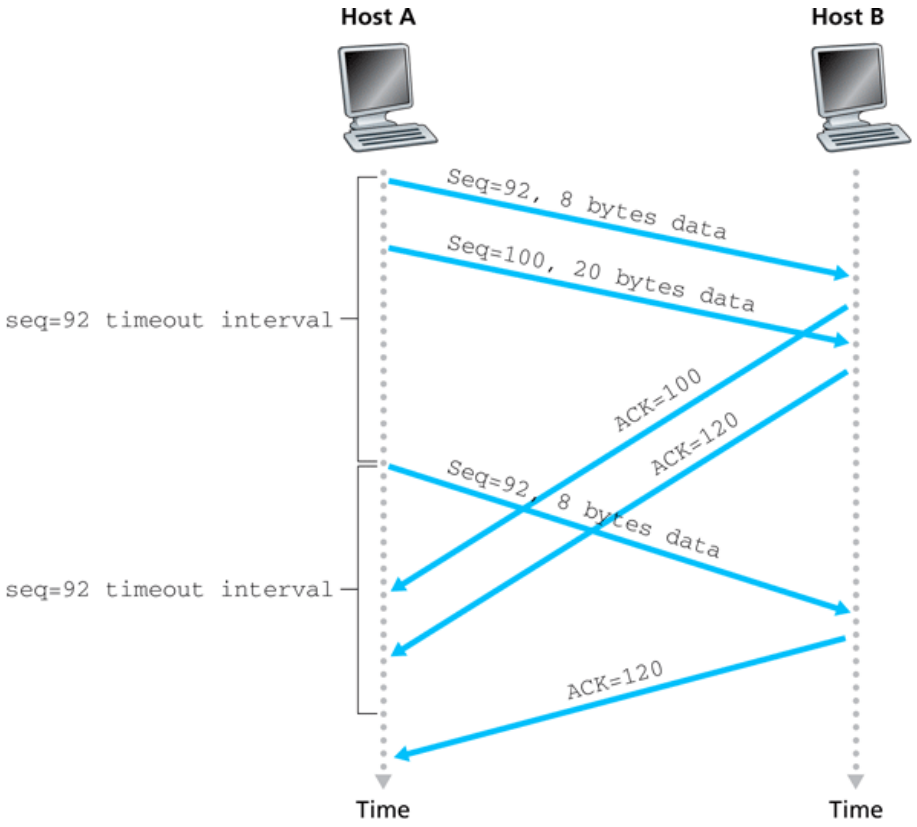
- **Acknowledgement loss and timeout**

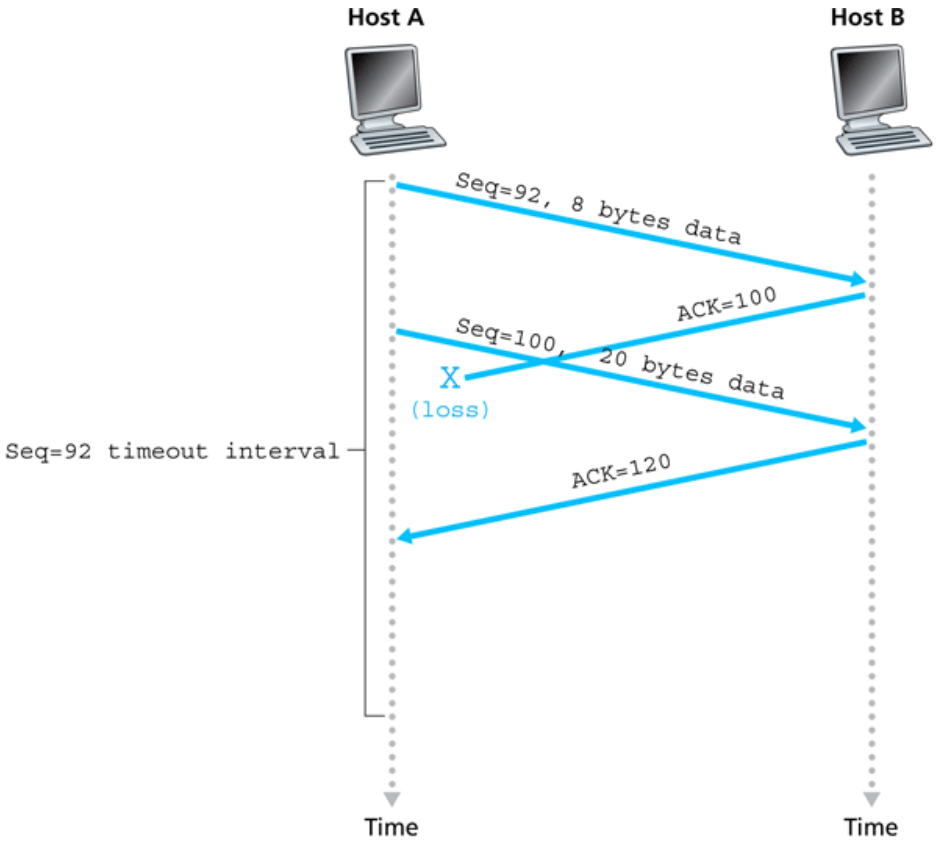




TCP Operation

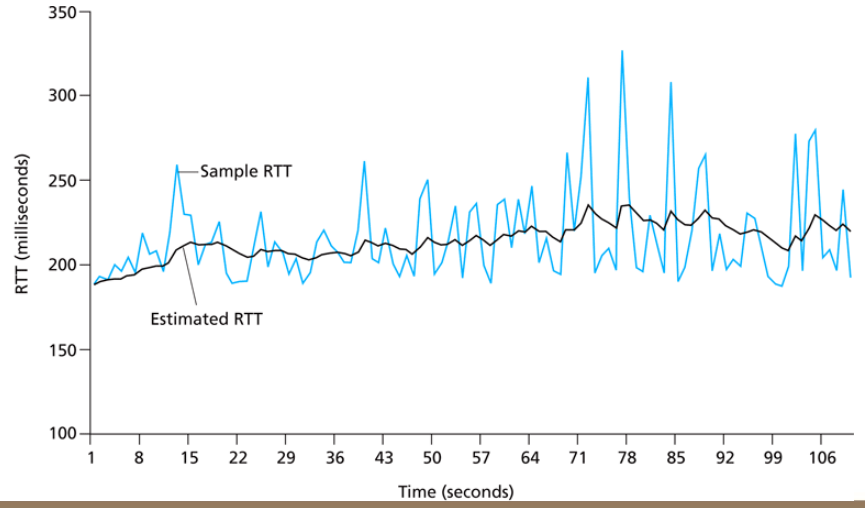
- **Timeout before acknowledgement received**





Round-Trip Time Estimation

- RTT estimate important for efficient operation
  - Too long: long delay before retransmission
  - Too short: unnecessary retransmission
- TCP RTT estimation
  - TCP keeps exponential weighted moving average of RTT
  - Timeout is set to estimation+4×deviation
- Example:

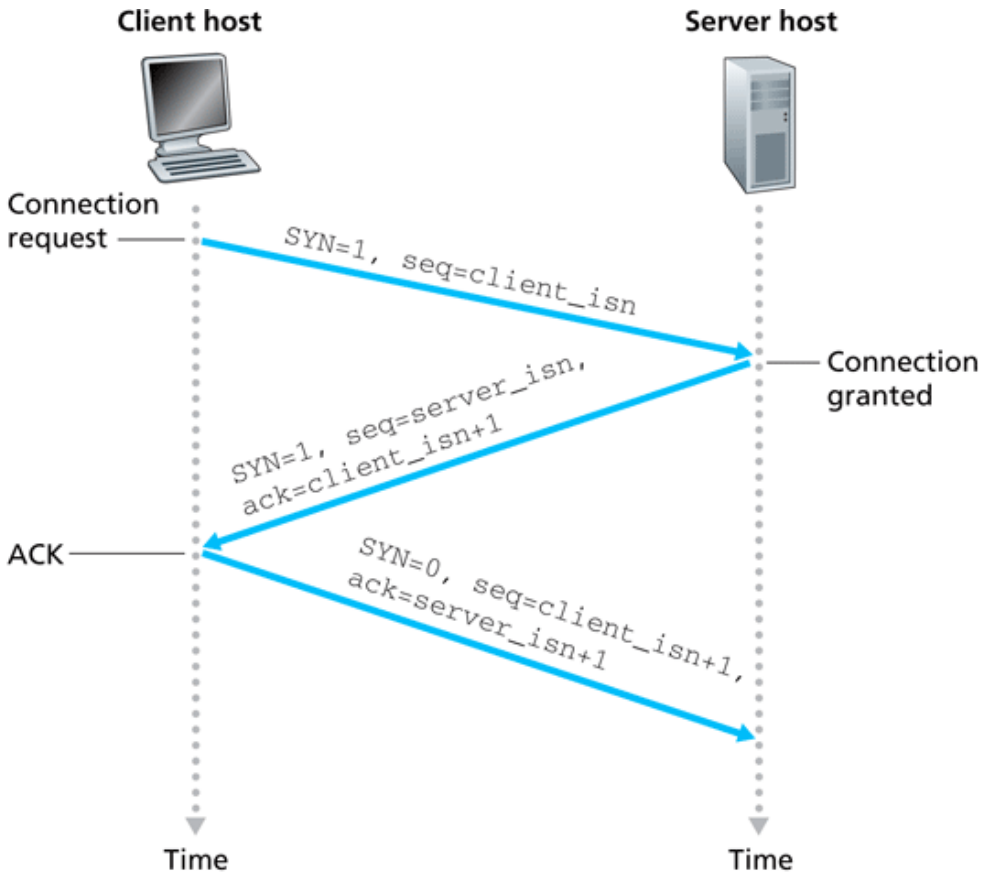


Topics

- **Introduction**
  - **Lessons 1 & 2**
- **Application Layer Protocols (HTTP, SMTP)**
  - Lesson 3
- **Transport Layer Protocols (TCP, UDP)**
  - Lesson 4
- **Transport Layer (Congestion Control)**
  - Lesson 5
- **Network Layer Protocol (IP)**
  - Lesson 6
- **Data Link Layer Protocol (Ethernet)**
  - Lesson 7
- **Software Defined Networks (SDN)**
  - Lesson 8

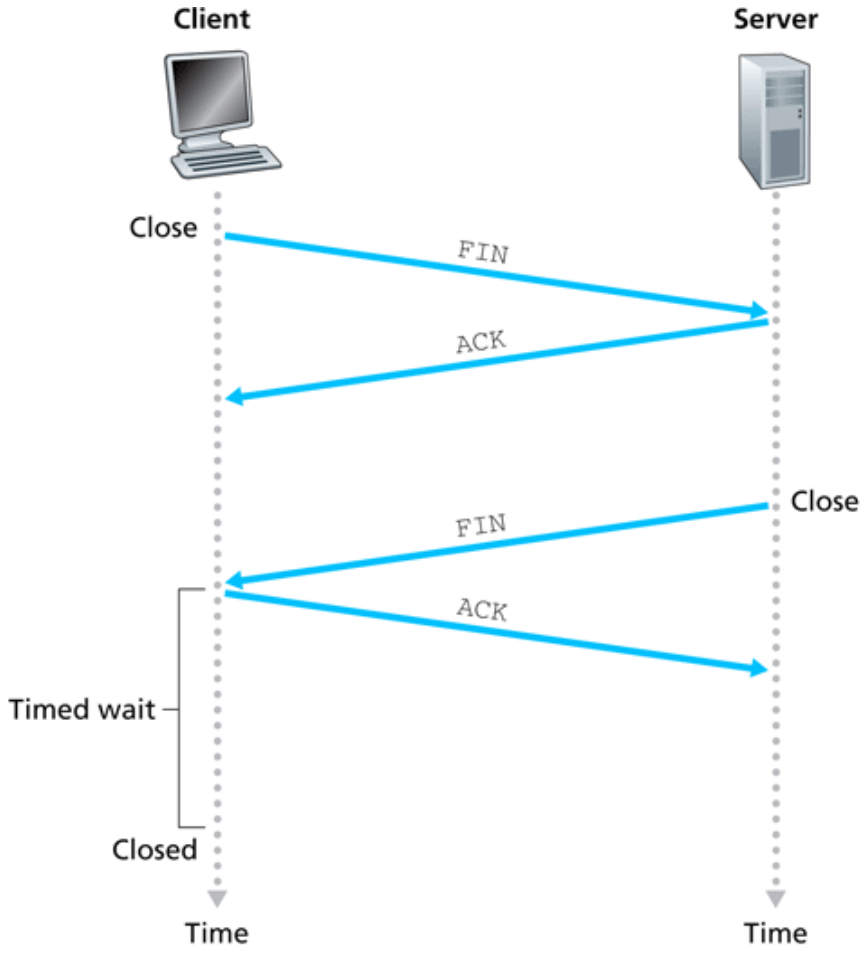
TCP Connection Management

- Connection setup
  - Three-way handshake
    - SYN
    - SYN/ACK
    - ACK
  - SYN counts as one byte
  - ACK may carry data already
  - Flag in header identifies SYN
    - Used in network systems to identify new connections



TCP Connection Management

- Connection teardown
  - Each side closes when transmission is complete
    - FIN
    - ACK
  - Final FIN or ACK may get lost
    - Need to be able to retransmit
  - Connection cleanup after timed wait

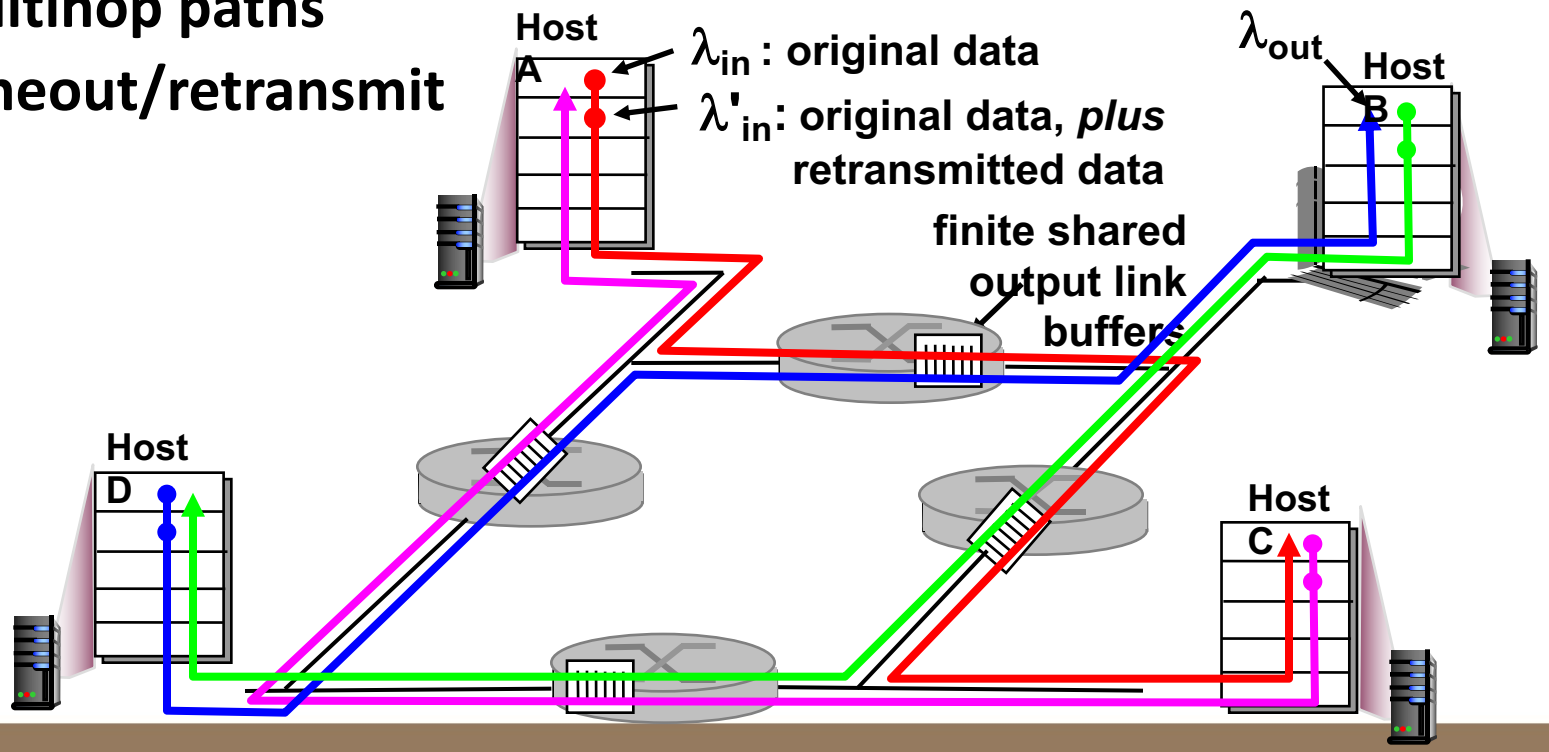


Cause of Congestion: Scenario 3

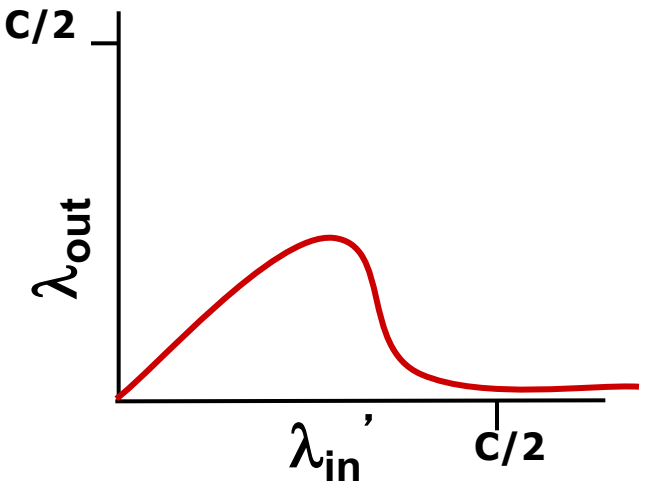
Q: What happens as λ_{in} and λ_{in}' increase ?

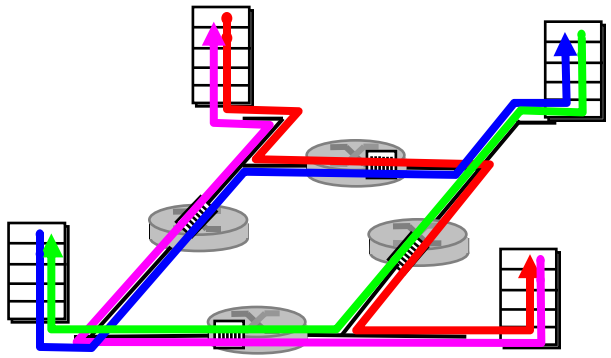
A: As red λ_{in}' increases, all arriving blue packets at upper queue are dropped, blue throughput → 0

- Scenario:
  - Four senders
  - Multihop paths
  - Timeout/retransmit



Cause of Congestion: Scenario 3





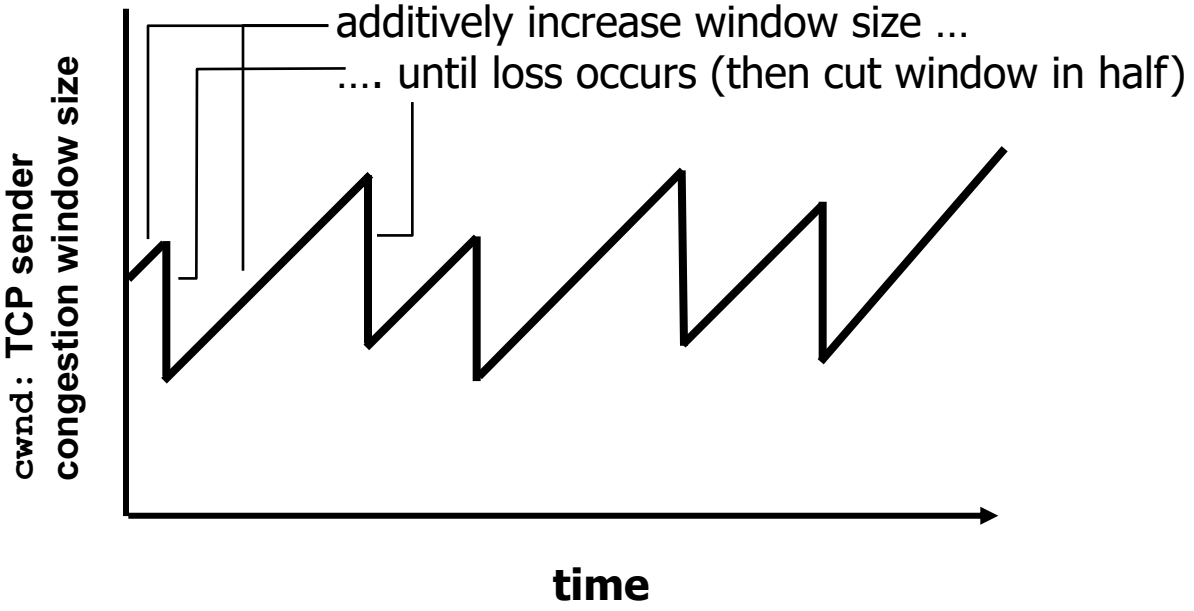
- Another "cost" of congestion:
  - When packet dropped, any "upstream transmission capacity used for that packet was wasted!

Congestion Control Approaches

- Reserving resources (cf. circuit switching)
  - As for permission before sending
  - Send at permitted rate
  - Congestion can be avoided through managing reservations
- Adapting sending rate (cf. packet switching)
  - Detect congestion (e.g., packet losses cause NAKs)
  - Reduce sending rate to avoid congestion collapse
- Tradeoff: central control vs. distributed approach

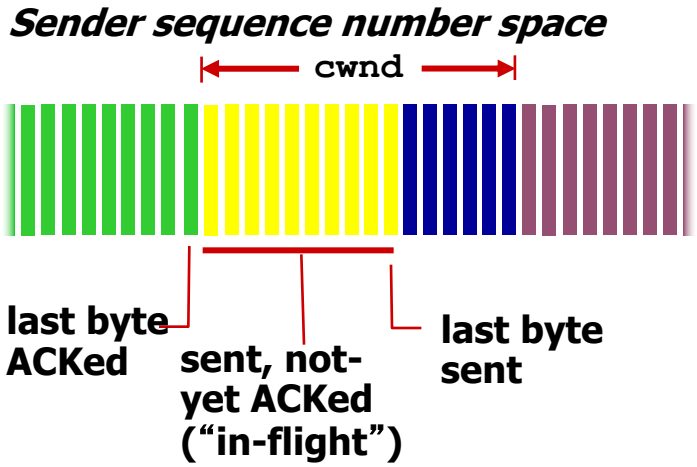
TCP Congestion Control

- Additive increase, multiplicative decrease (AIMD)
- Approach: sender increases transmission rate (window size), probing for usable bandwidth, until loss occurs
  - Additive increase: increase `cwnd` by 1 MSS every RTT until loss detected
  - Multiplicative decrease: cut `cwnd` in half after loss



AIMD saw tooth behavior: probing for bandwidth

TCP Congestion Control



- Sender limits transmission:

$$\texttt{LastByteSent} - \texttt{LastByteAcked} \leq \texttt{cwnd}$$

- `cwnd` is dynamic, function of perceived network congestion

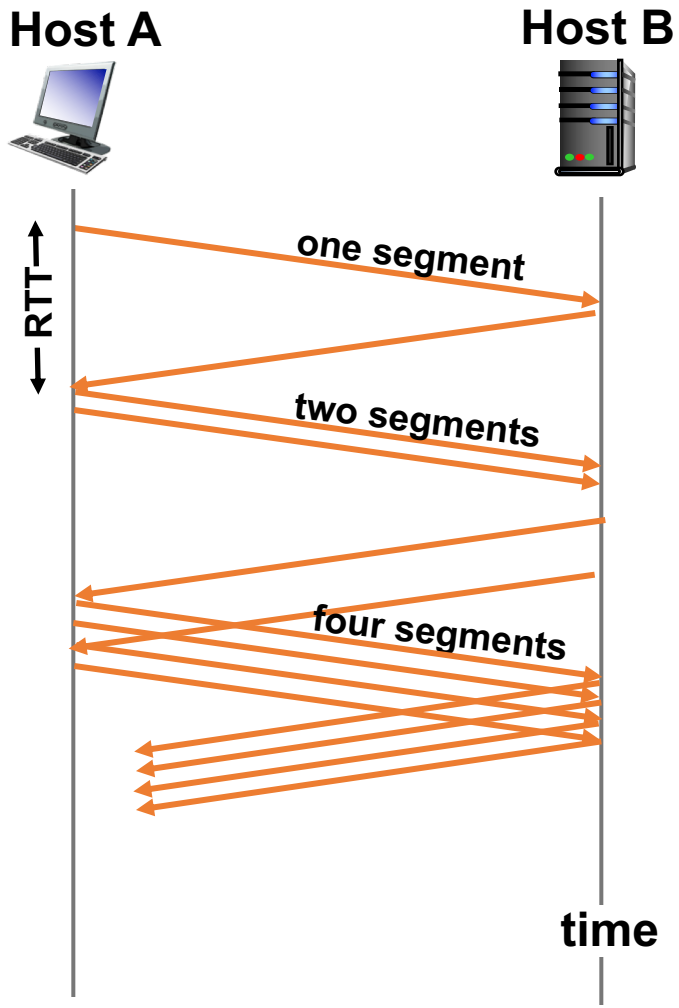
TCP sending rate:

- Roughly: send cwnd bytes, wait RTT for ACKS, then send more bytes

$$\text{rate} \approx \frac{\text{cwnd}}{\text{RTT}} \text{ bytes/sec}$$

TCP Slow Start

- When connection begins, increase rate exponentially until first loss event:
  - Initially `cwnd` = 1 MSS
  - Double `cwnd` every RTT
  - Done by incrementing `cwnd` for every ACK received
- <u>Summary:</u> initial rate is slow but ramps up exponentially fast



TCP Loss Detection and Reaction

- Loss indicated by timeout:
  - `cwnd` set to 1 MSS;
  - Window then grows exponentially (as in slow start) to threshold, then grows linearly
- Loss indicated by 3 duplicate ACKs: TCP RENO
  - Duplicate ACKs indicate the network is capable of delivering some segments
  - `cwnd` is cut in half, window then grows linearly
- TCP Tahoe always sets `cwnd` to 1 (timeout or 3 duplicate ACKs)

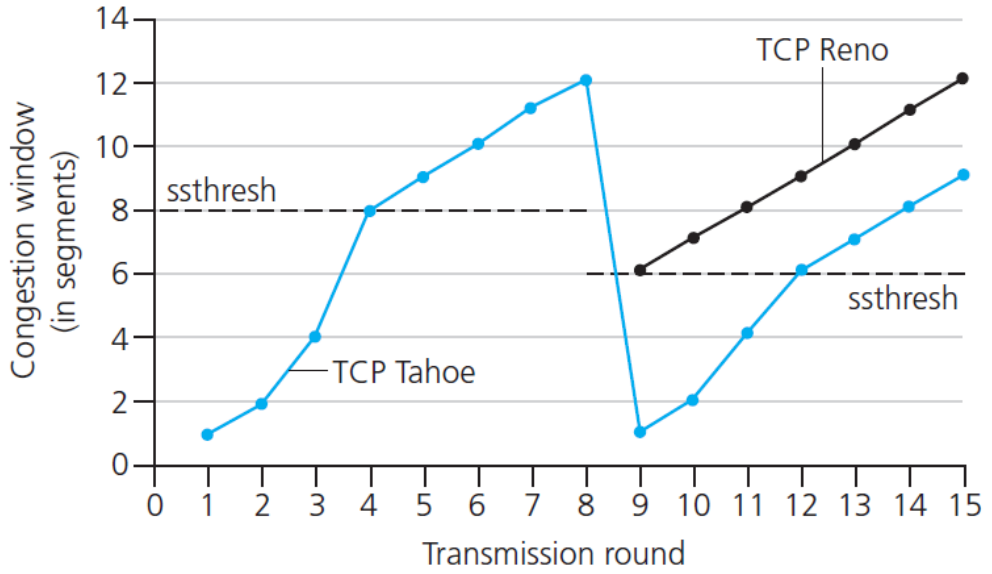
TCP Slow Start and Congestion Avoidance

Q: When should the exponential increase switch to linear?

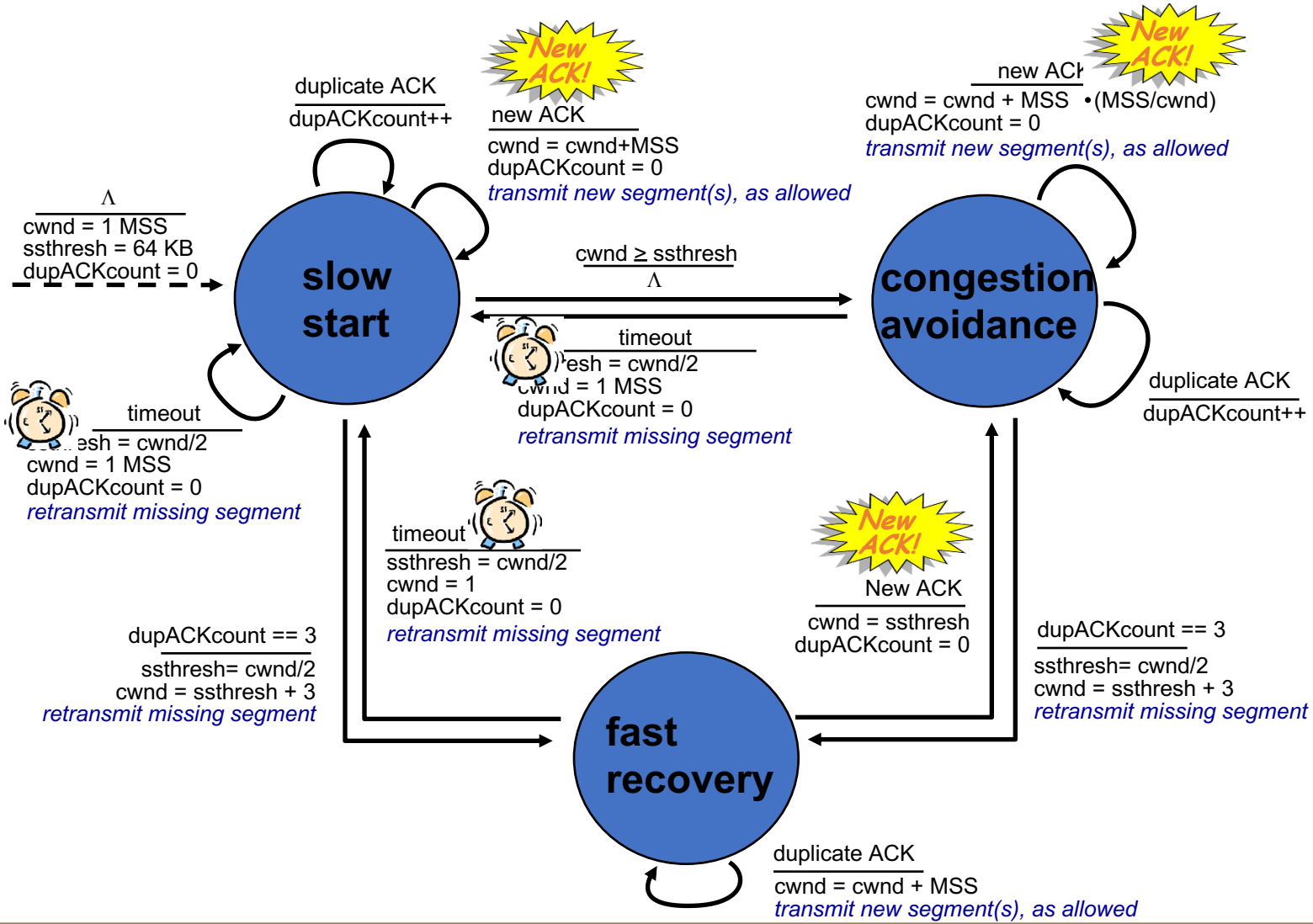
A: When `cwnd` gets to 1/2 of its value before timeout.

Implementation:

- Variable `ssthresh`
- On loss event, `ssthresh` is set to 1/2 of `cwnd` just before loss event

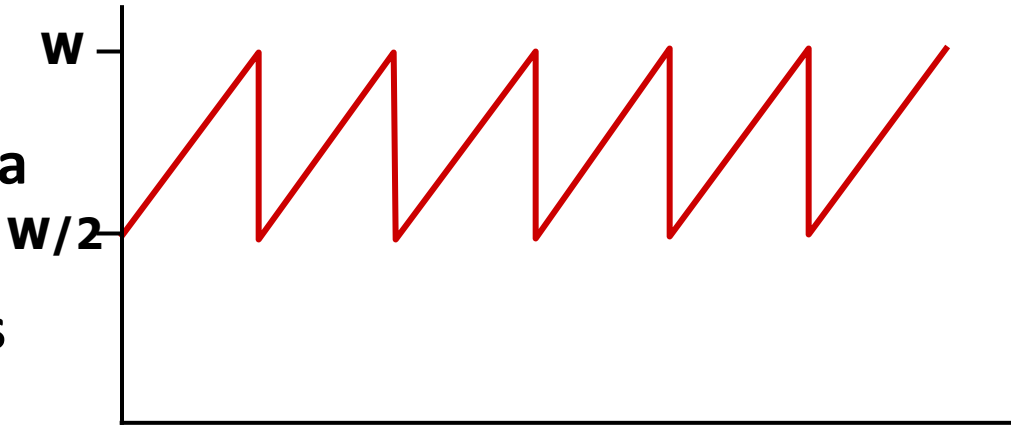


TCP Congestion Control Summary



TCP Throughput

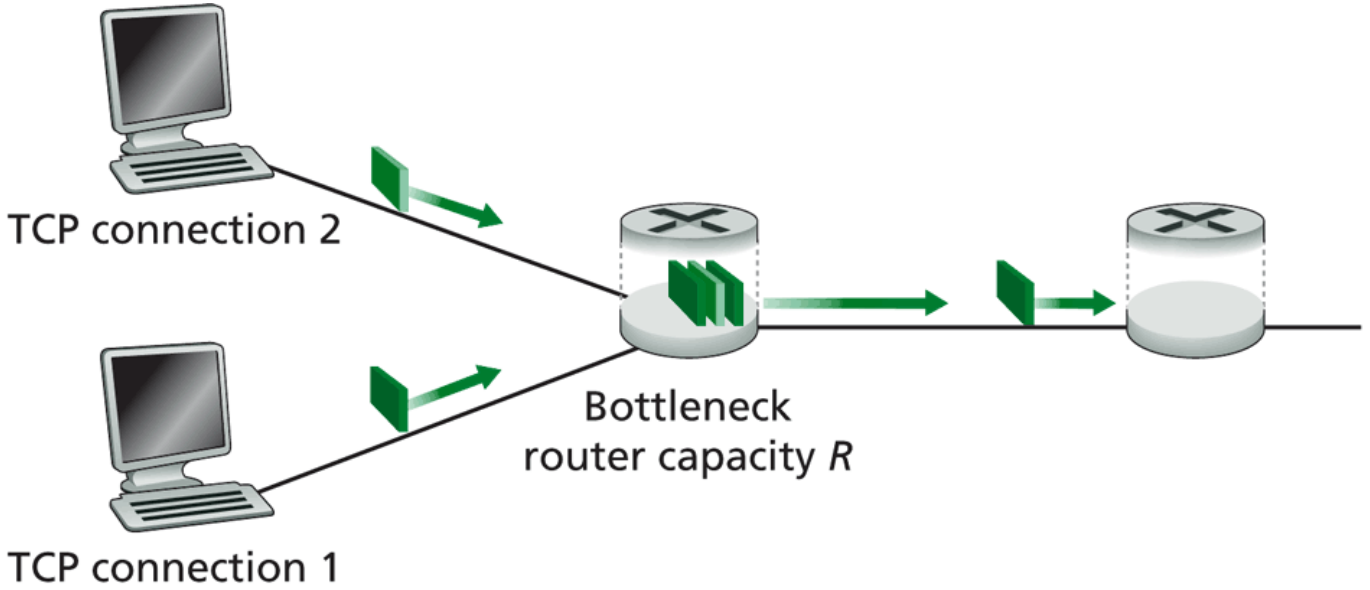
- What is the average TCP throughput as function of window size W and RTT?
  - Ignore slow start, assume always data to send
- W: window size (measured in bytes) where loss occurs
  - Average window size (# in-flight bytes) is ¾ W
  - Average throughput is 3/4W per RTT



$$\text{Avg. TCP throughput} = \frac{3}{4}\,\frac{W}{RTT}\ \text{bytes/sec}$$

Link Sharing with TCP

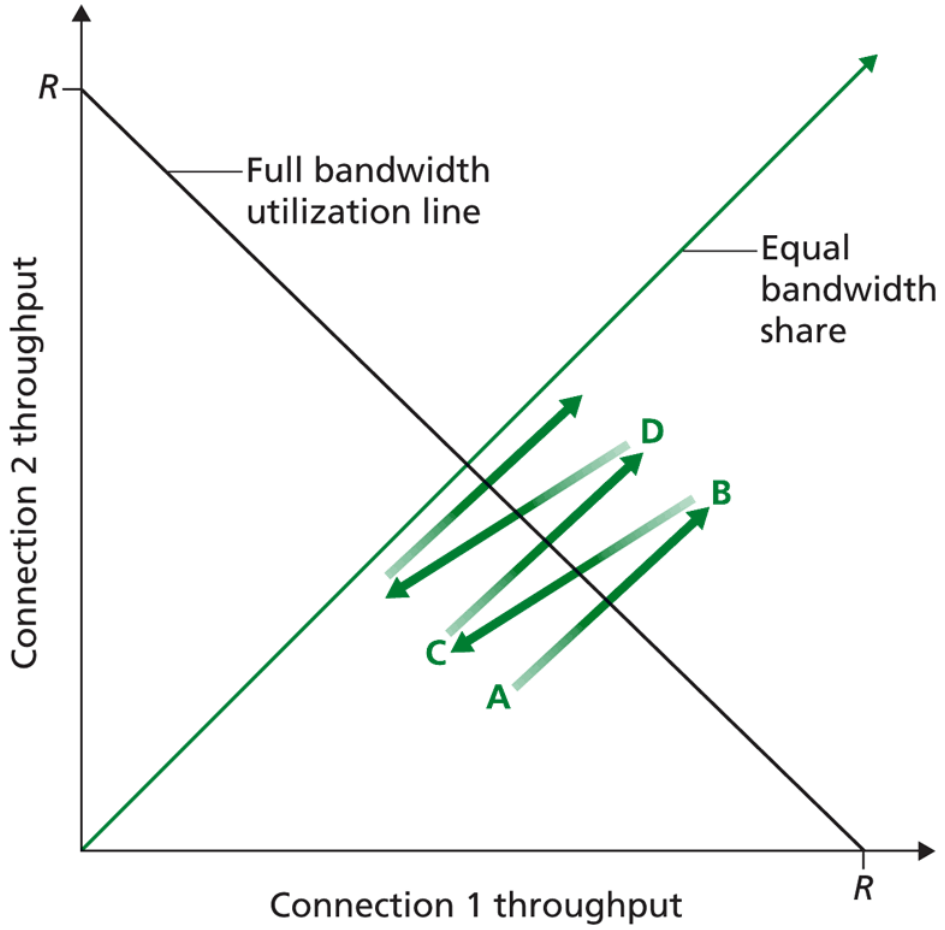
- Two TCP connections share a link:



- Eventually each connection receives a fair share
  - How can this be shown?

Link Sharing with TCP

- **Illustration of two connections**

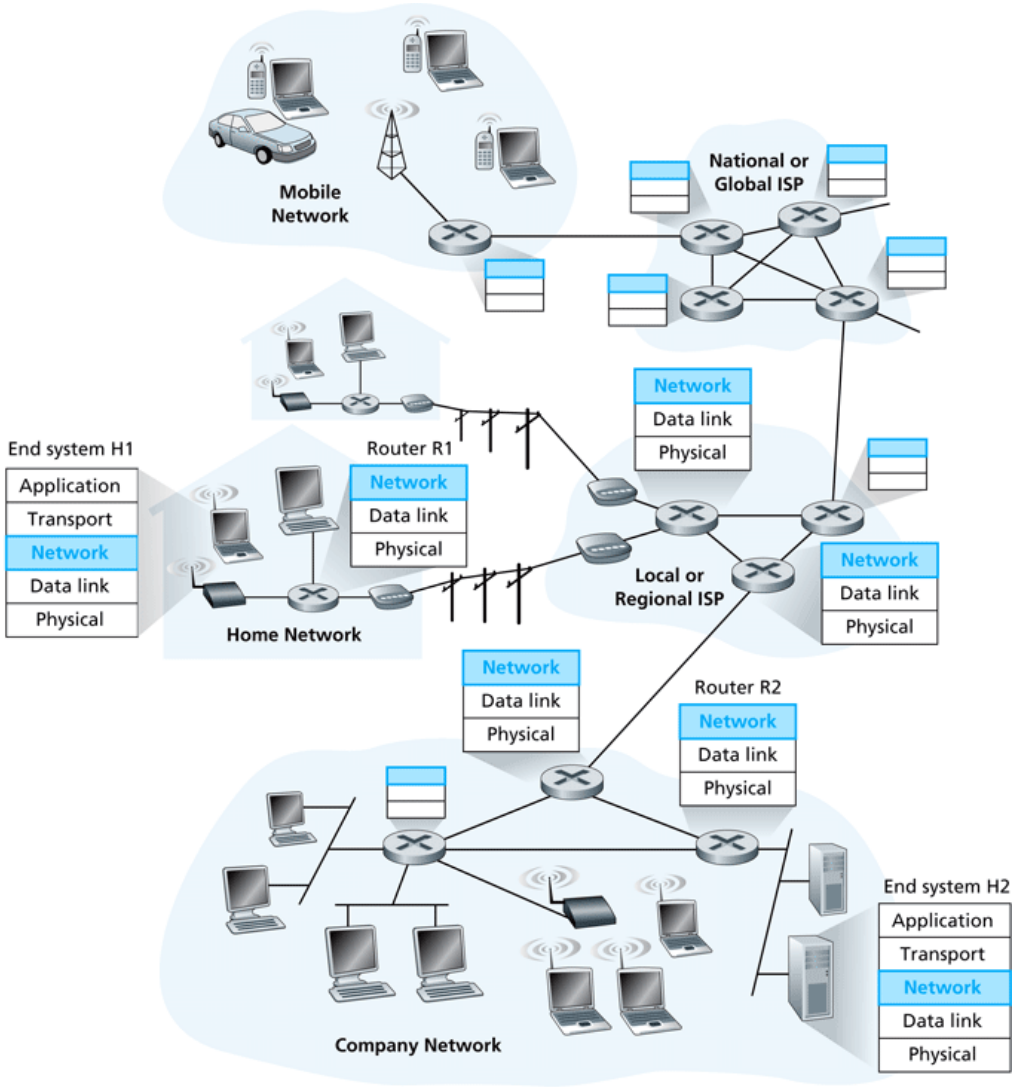


Topics

- **Introduction**
  - **Lessons 1 & 2**
- **Application Layer Protocols (HTTP, SMTP)**
  - Lesson 3
- **Transport Layer Protocols (TCP, UDP)**
  - Lesson 4
- **Transport Layer (Congestion Control)**
  - Lesson 5
- **Network Layer Protocol (IP)**
  - Lesson 6
- **Data Link Layer Protocol (Ethernet)**
  - Lesson 7
- **Software Defined Networks (SDN)**
  - Lesson 8

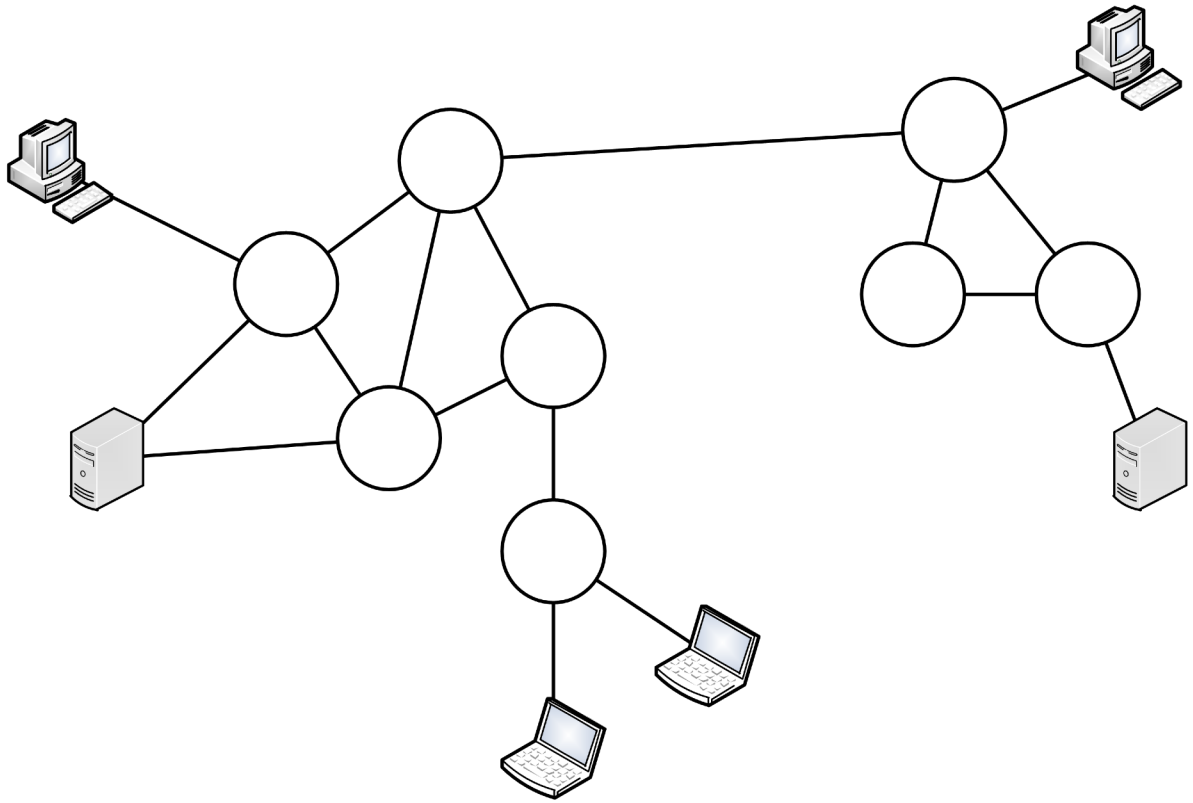
Network Layer

- Network layer provides end-to-end connectivity
  - Service to layer above
    - Unreliable channel
  - Uses link layer
    - Point-to-point links
- Connectivity is between end-system interfaces
  - Computers may have multiple interfaces



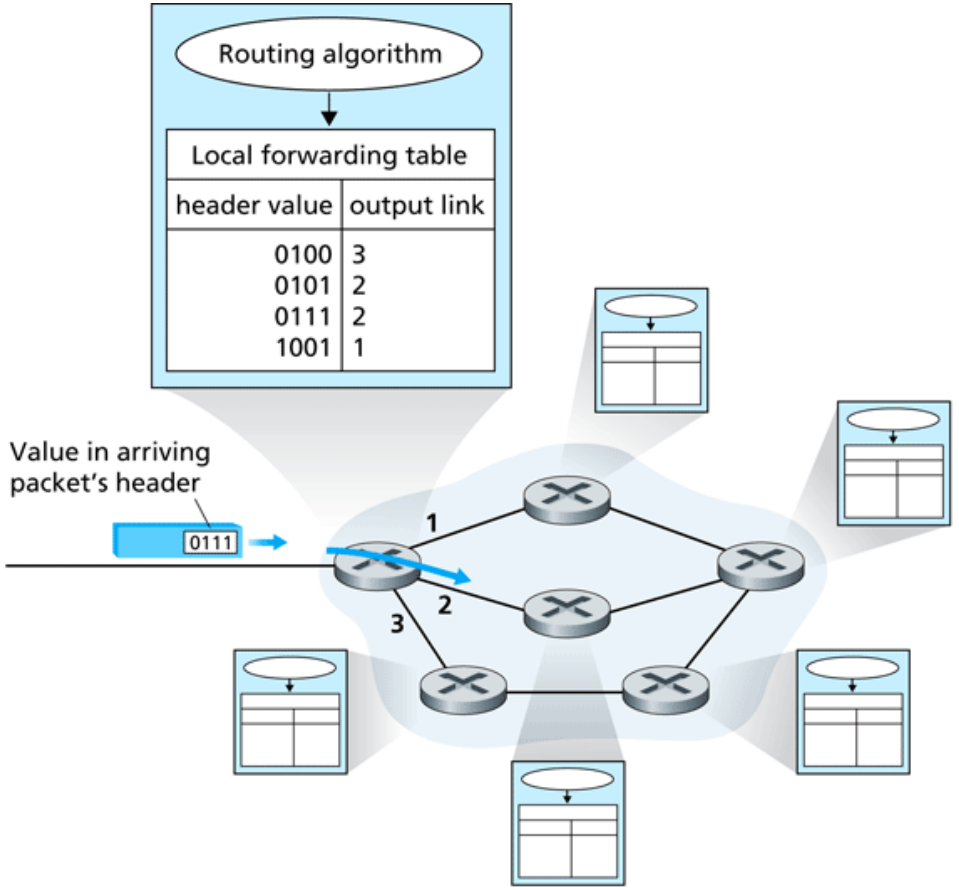
Interface-to-Interface Connectivity

- How can we provide end-to-end connectivity if we can only send link by link?
  - Need to have addresses to identify end-systems
  - Need to know which path to take



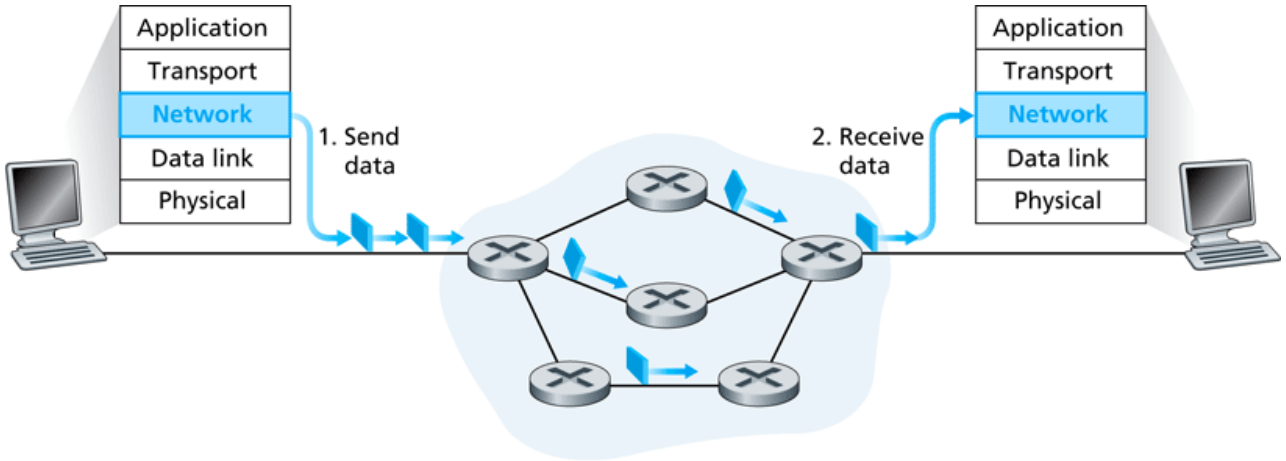
Connectivity

- Connectivity requires
  - Addressing
  - Routing and forwarding
- Internet is packet-switched network
- Two different approaches for connectivity
  - Datagram network
  - Virtual circuit network



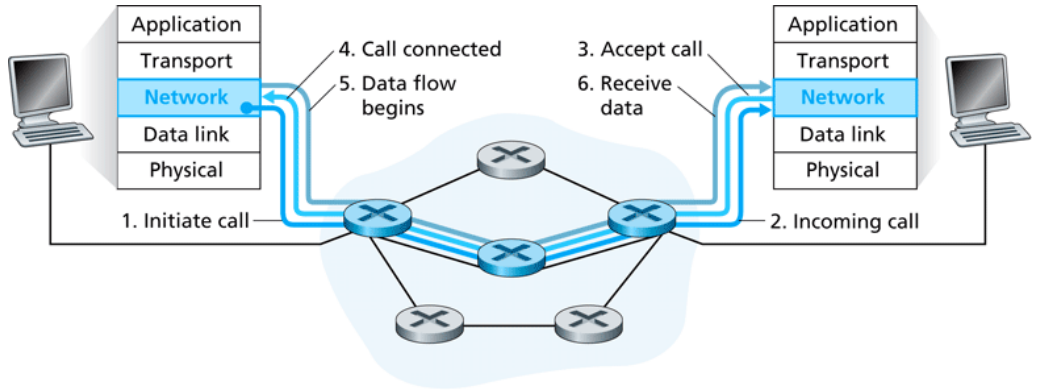
Datagram Networks

- Packets are sent independently of each other
  - Each packet has full set of control information
- Every switch needs to be able to handle any packet
  - No need for per-connection state



Virtual Circuit Networks

- End-to-end connectivity through virtual circuits
- Connection process
  - Connection setup
  - Data transfer
  - Connection teardown
- Router maintains state for every connection
- We will see that again later in context of software-defined networks

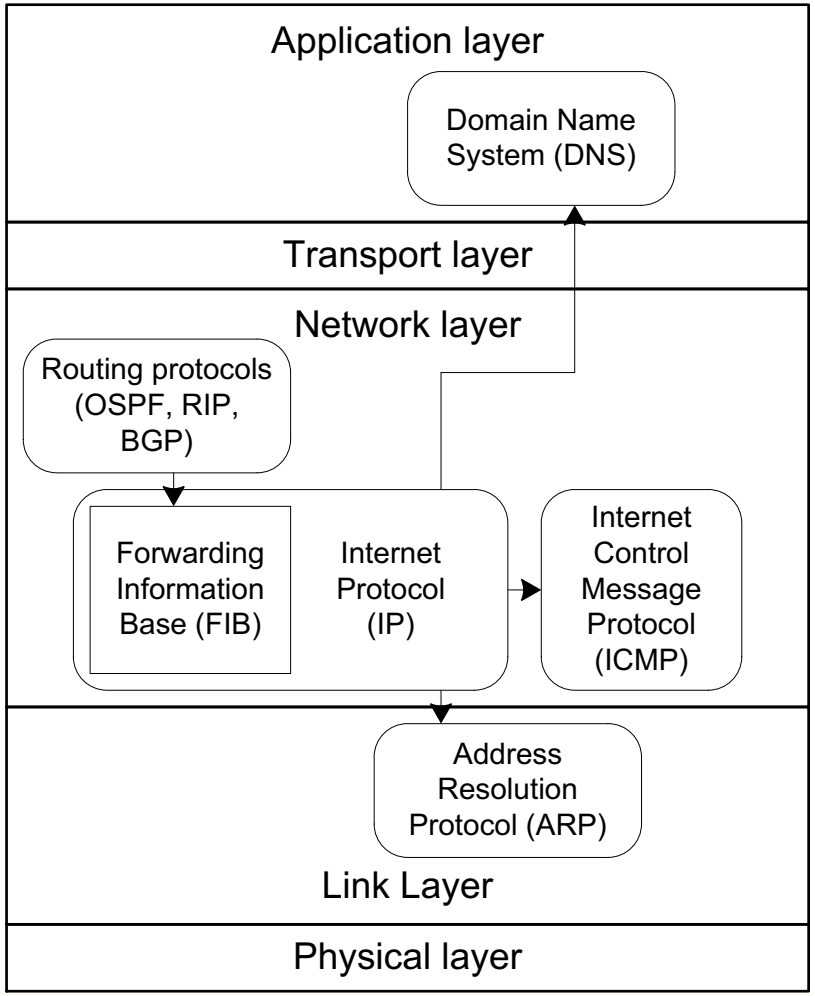


Internet Protocol

- Internet Protocol (IP)
  - Network layer of the Internet
- Data plane
  - Forwarding of IP packets on node
  - Local per-node functions
- Control plane
  - Routing from source to destination along end-to-end path
  - Network-wide functions

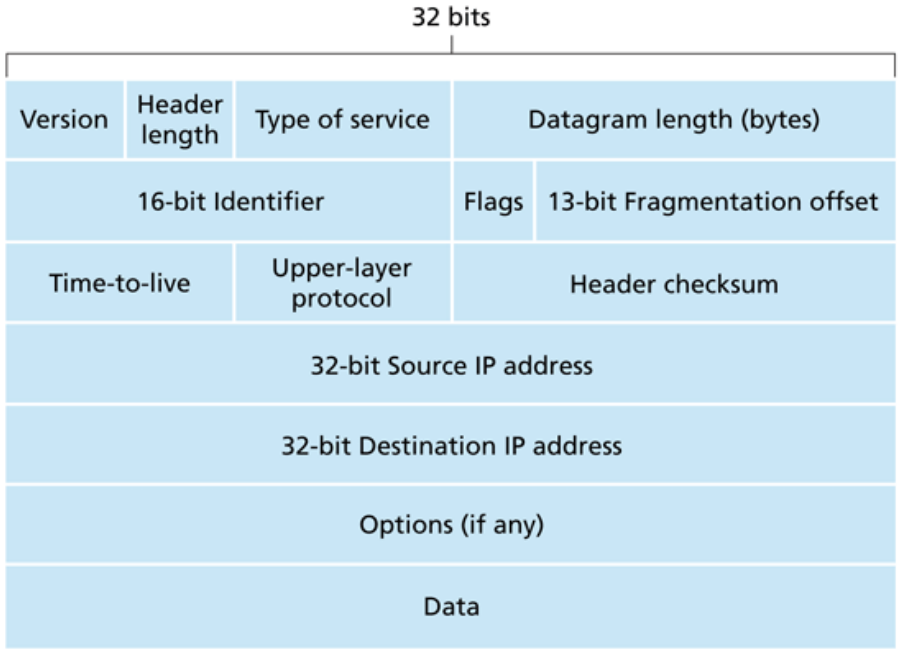
Other IP Aspects

- IP needs to interact with other components in protocol stack
- ICMP
  - Error handling
- Link layer
  - Address resolution (ARP)
- Application layer
  - Domain names (DNS)
  - Dynamic IP addresses (DHCP)
- Transport layer
  - Network address translation (NAT)



Internet Protocol

- IP (version 4) header format:
  - Source and destination address
  - Datagram length
  - Upper layer protocol
    - Identifies TCP, UDP, etc.
  - Time to live
    - Protection against accidental loops
  - Header checksum
    - Protection against bit errors
  - Fragmentation possible
    - Link layer limited to some datagram size (min. MTU is 576 bytes)

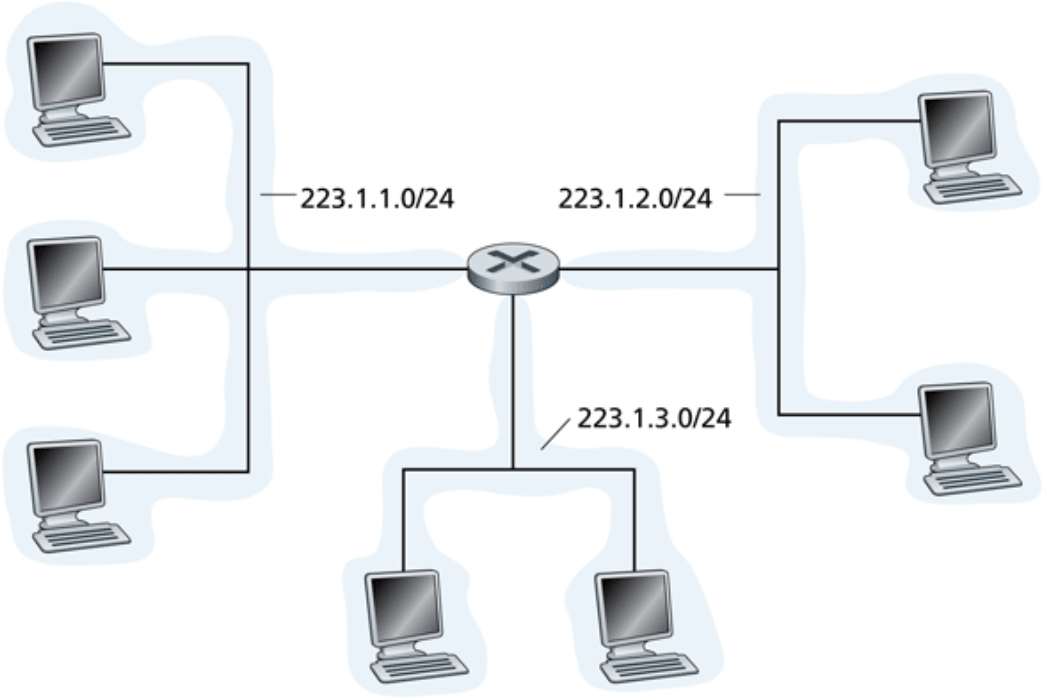


Whiteboard: IP Addressing

- A 32-bit globally unique identifier for an interface
  - Typically written in dotted-decimal notation: 192.168.0.1
- IP address assignment
  - In blocks of neighboring IP addresses: “subnets”
  - Notation: lowest address / prefix: 192.168.0.128/25
- Note: the address classes are dead!
  - Classless Interdomain Routing (CIDR)
- Allocation of addresses is crucial for routing
- Addresses assigned by Internet Assigned Numbers Authority (IANA)

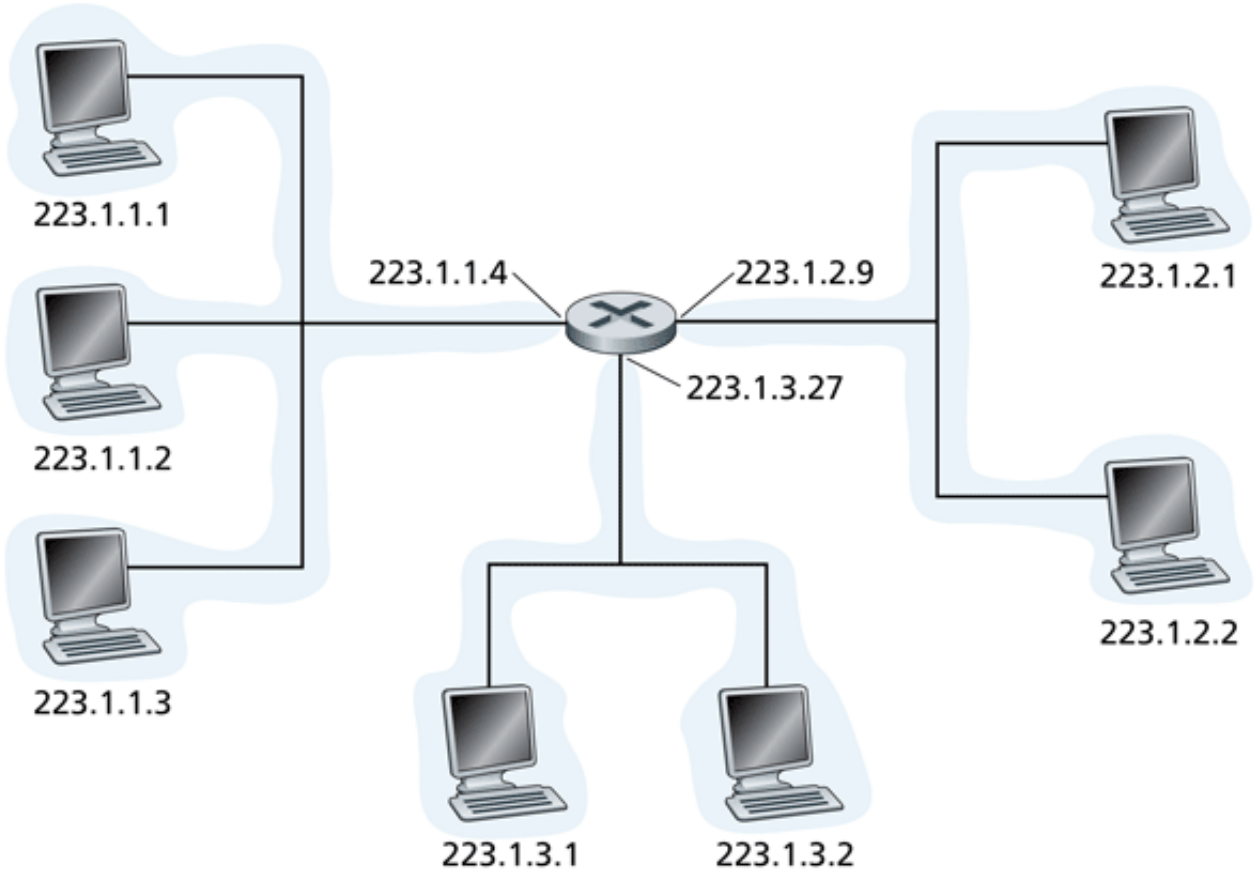
IP Address Allocation

- Address space is allocated to subnetwork
  - Each subnet represented by “prefix”
  - Notation indicates length of prefix
  - Example: 223.1.1.0/24



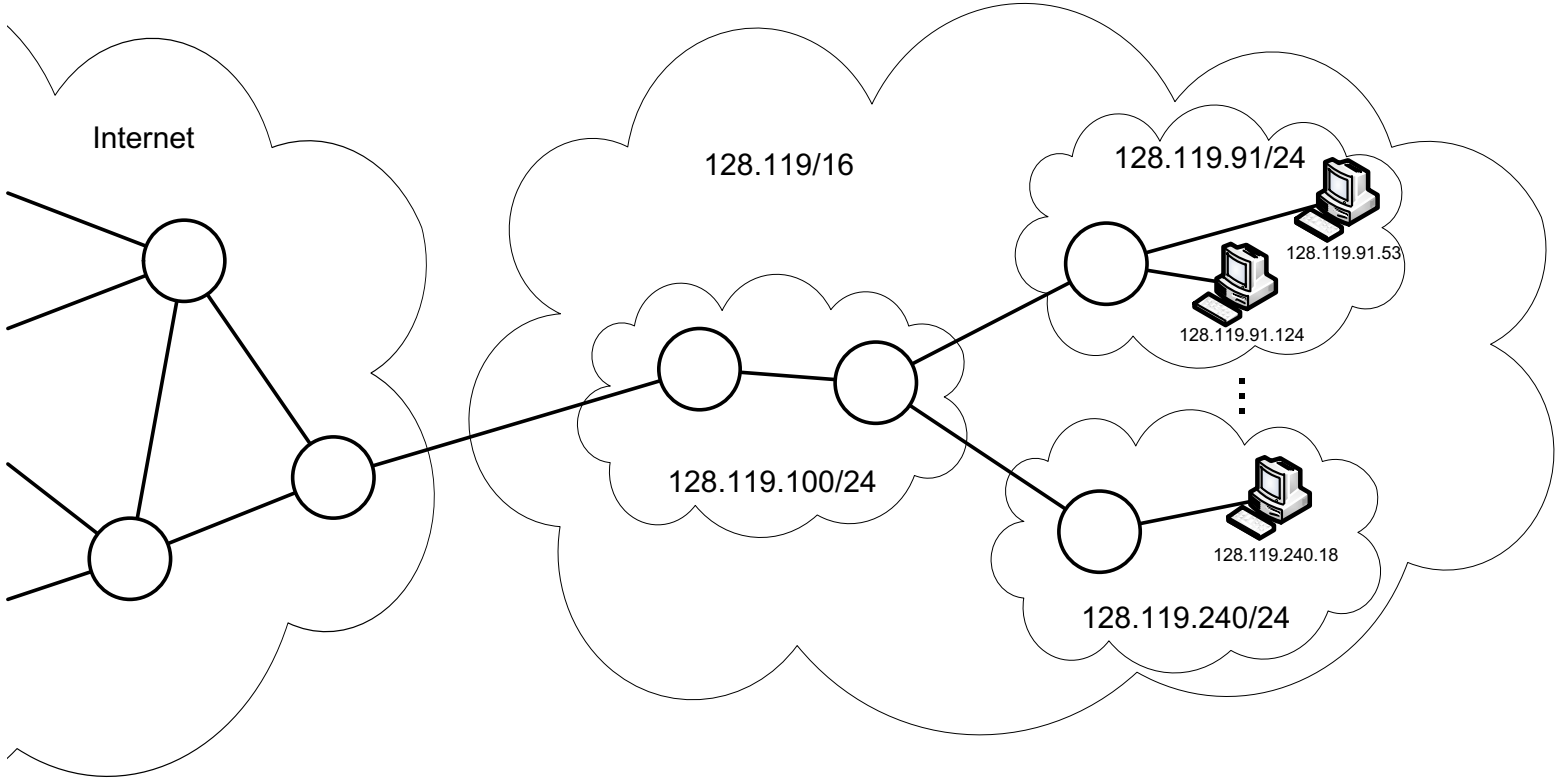
IP Address Allocation

- Example address allocation:
  - Address allocation determined by network administrator
  - Even point-to-point links require addresses for interfaces



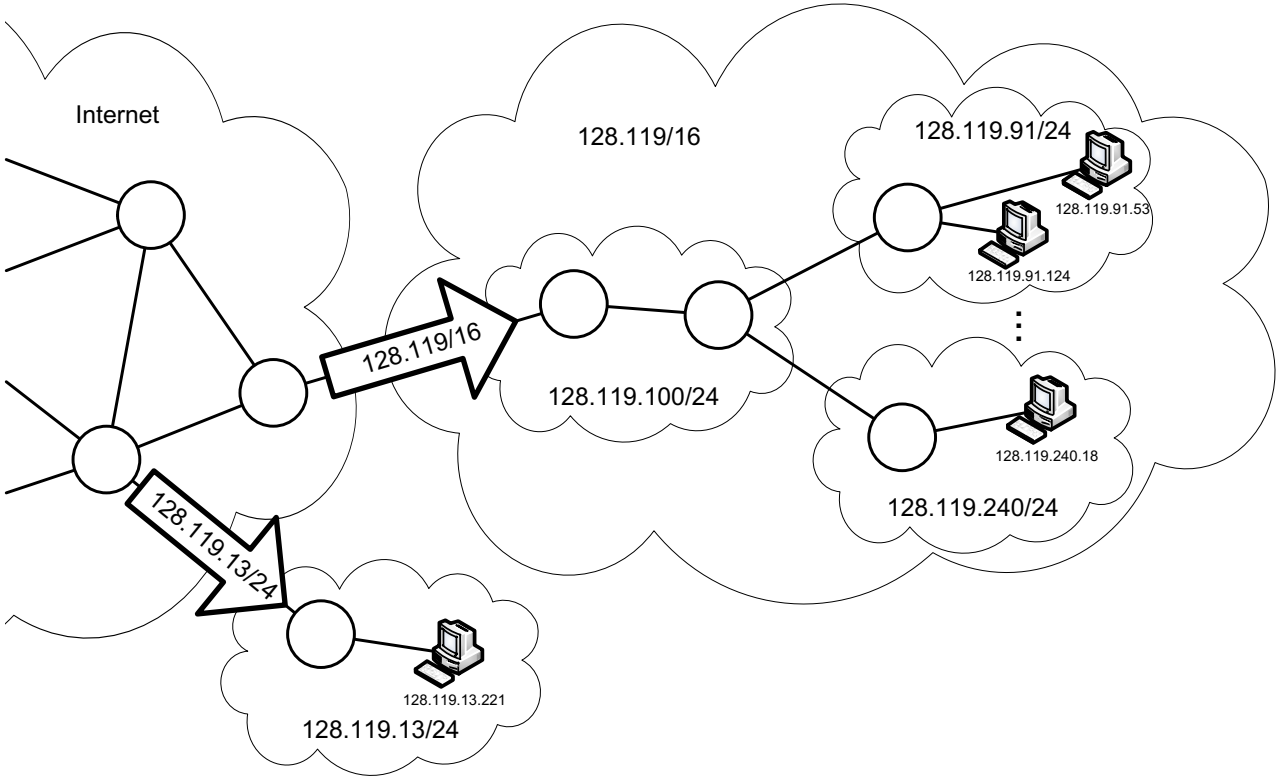
Hierarchical Address Allocation

- IP address allocation is hierarchical
  - Subnets within subnets have longer prefixes
- Rest of Internet only needs to know of highest layer in hierarchy for routing ("address aggregation")
- Example:



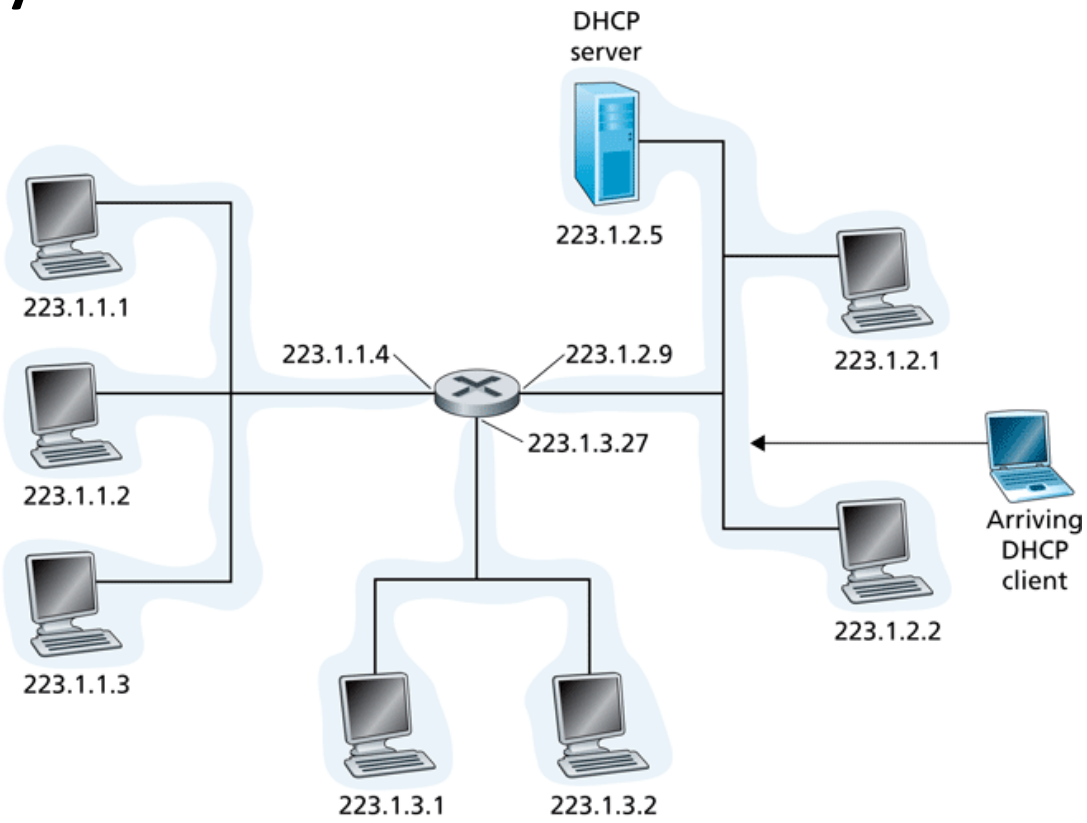
Address Aggregation

- Internet does allow addresses to move out of subnet
- Work-around to handle disaggregation:
  - Both aggregates are known to routers
  - Routers perform “longest prefix match”
    - Most specific route used for forwarding



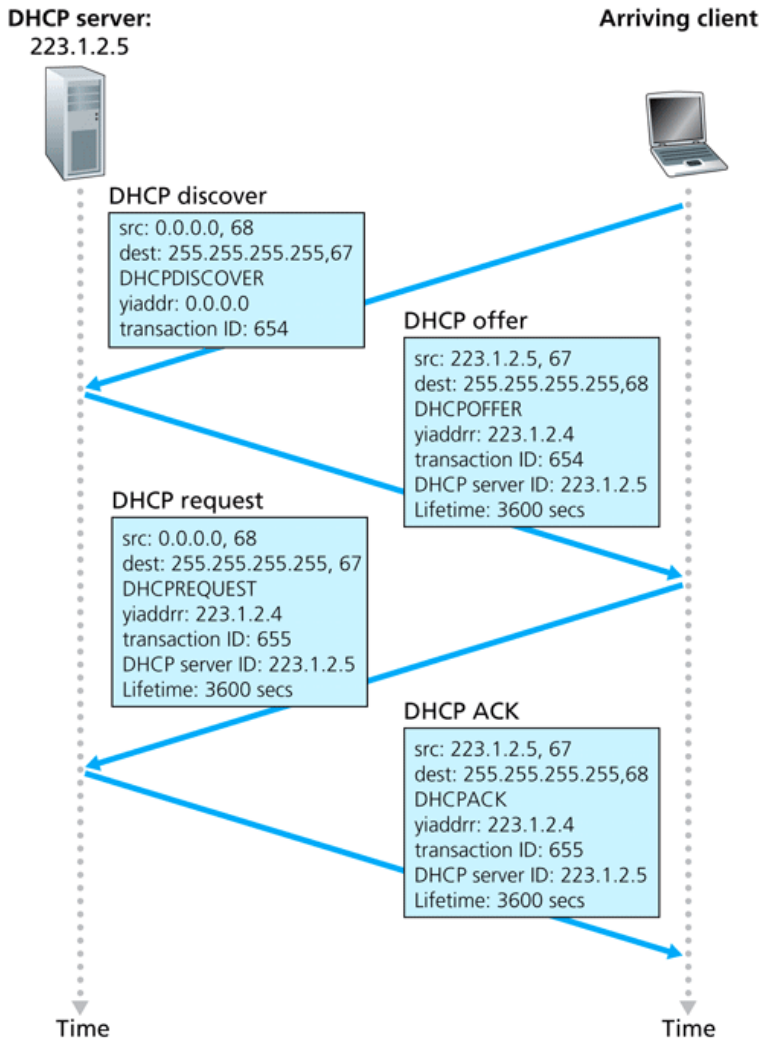
Dynamic Host Configuration Protocol

- DHCP provides IP addresses to end-systems in LAN
  - Example: using a laptop in a coffee shop
  - Manual configuration may be too complex



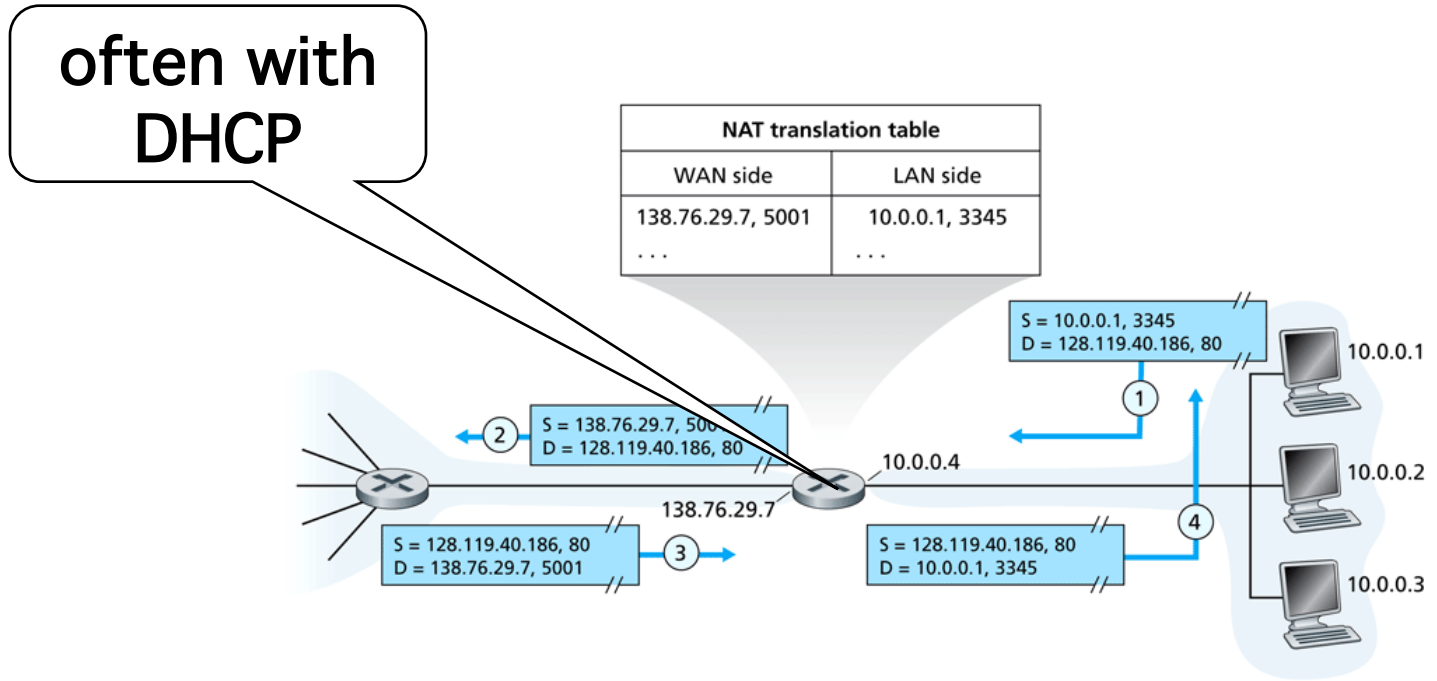
Dynamic Host Configuration Protocol

- DHCP protocol exchange
  - DHCP client tries to discover DHCP server
    - Communication possible without IP address
  - DHCP server offers IP address to client
  - DHCP client requests use of IP address
  - DHCP server assigns IP address to client
- Address lease for limited time



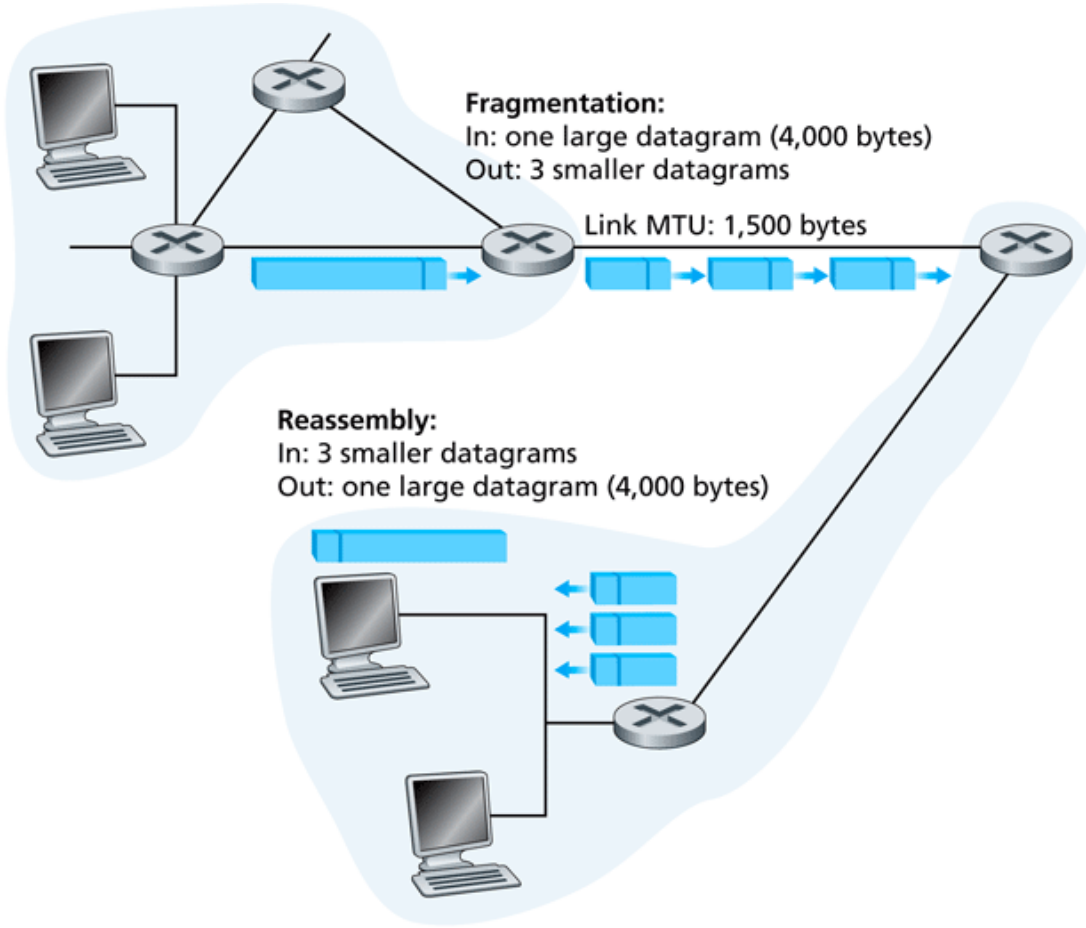
Network Address Translation (NAT)

- Limited number of IP addresses
  - Block of addresses reserved for “local” use
    - 10.*.*.* and 192.168.*.*
- Network address translator
  - Connects local net through single outside IP address



IP Fragmentation

- Data link layer determines Maximum Transfer Unit
  - E.g., 1500 bytes on Ethernet
- If packet encounters smaller MTU than size, then router fragments packet
- Reassembly on end-system
  - Not on router!
- Internet minimum MTU is 576 bytes



Internet Control Message Protocol

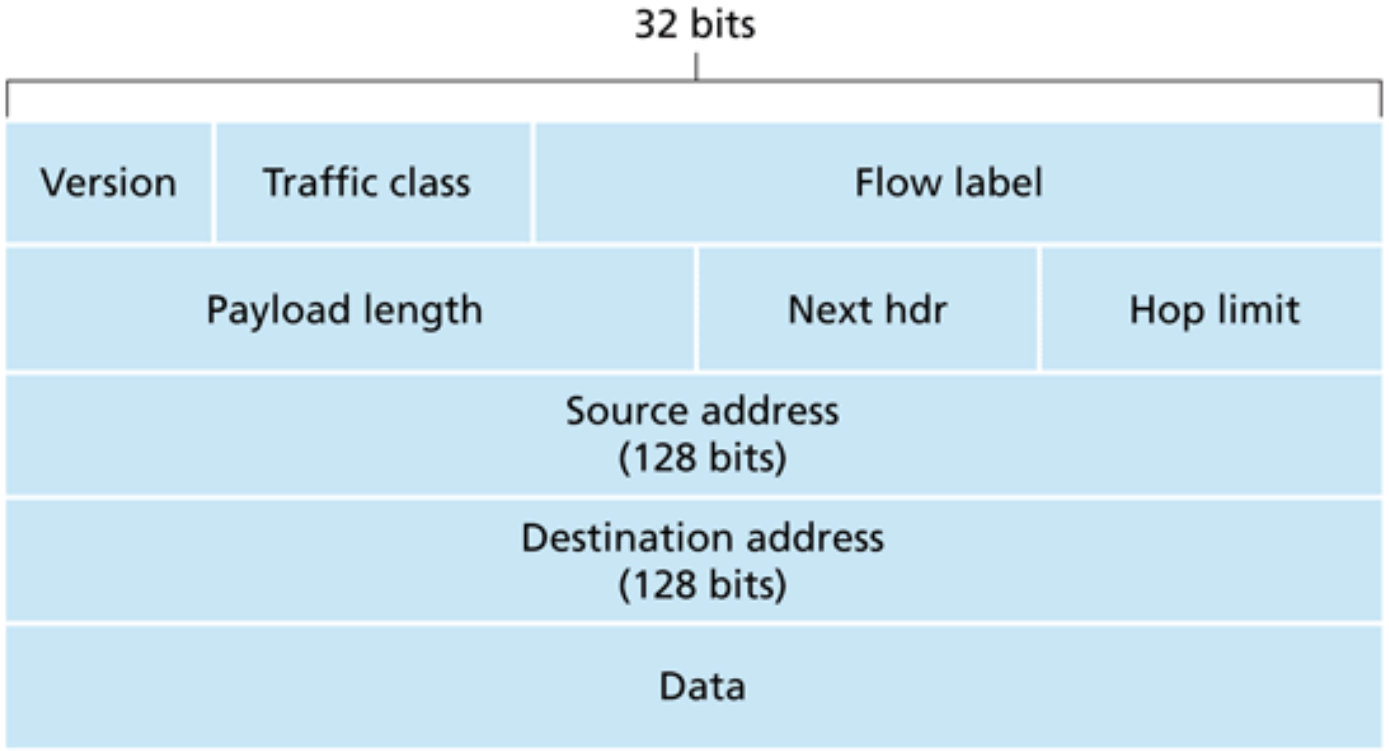
- ICMP provides control and error messages:
  - Echo request / reply
  - Destination unreachable (network, host, protocol, port)
  - Destination unknown (network, host)
  - Congestion control
  - Router advertisement
  - Router discovery
  - TTL expired
  - IP header bad
- Can be used (and abused) for many purposes
  - Intentional TTL expiration for route discovery

IP Version 6

- 32-bit address space of IPv4 almost completely allocated
  - Difficult to get IP addresses
  - More fragmentation of IP address space
  - Network address translation “obscures” network
- Version 6 of IP
  - 128-bit addresses
  - Improved header format for faster processing on routers

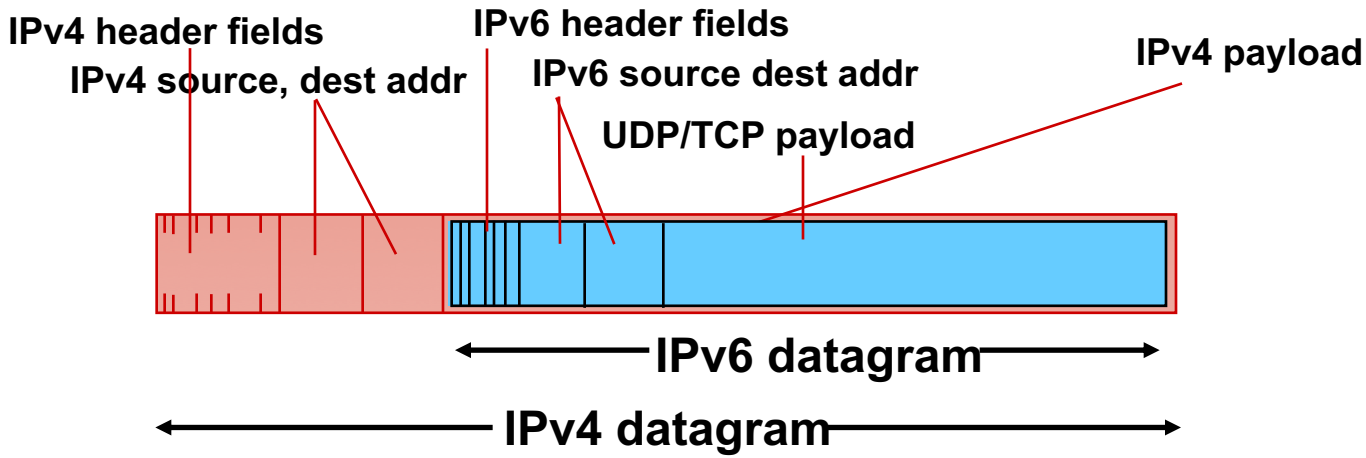
IP Version 6

- IP (version 6) header format:
  - 128-bit addresses (unicast, multicast, anycast)
  - Flow labeling and priority
  - No fragmentation
  - No header checksum
  - Option in next header

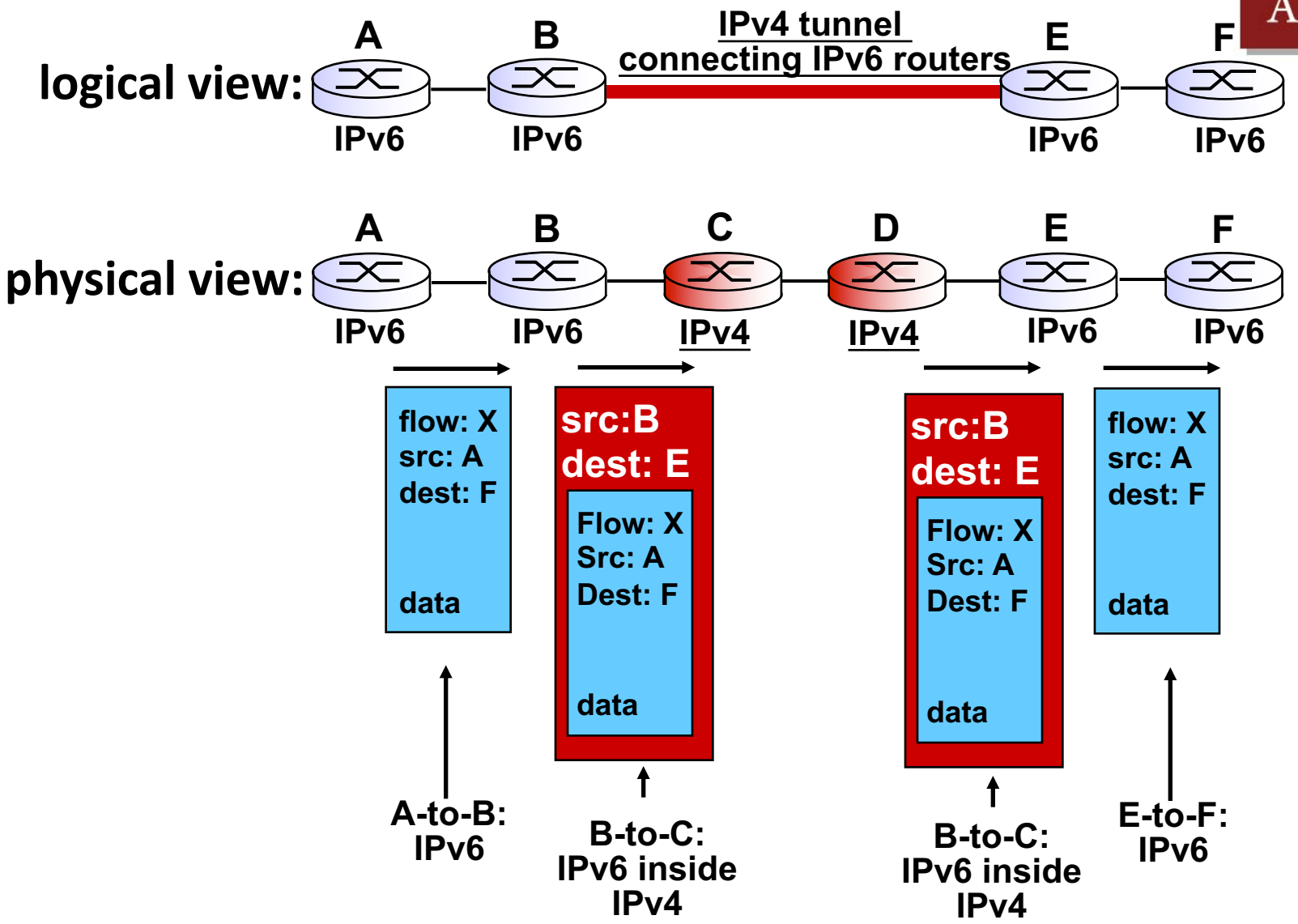


Transition to IPv6

- Not all routers can be upgraded simultaneously
  - No “flag day”
  - How will network operate with mixed IPv4 and IPv6 routers?
- Tunneling: IPv6 datagram carried as payload in IPv4 datagram among IPv4 routers



Tunneling

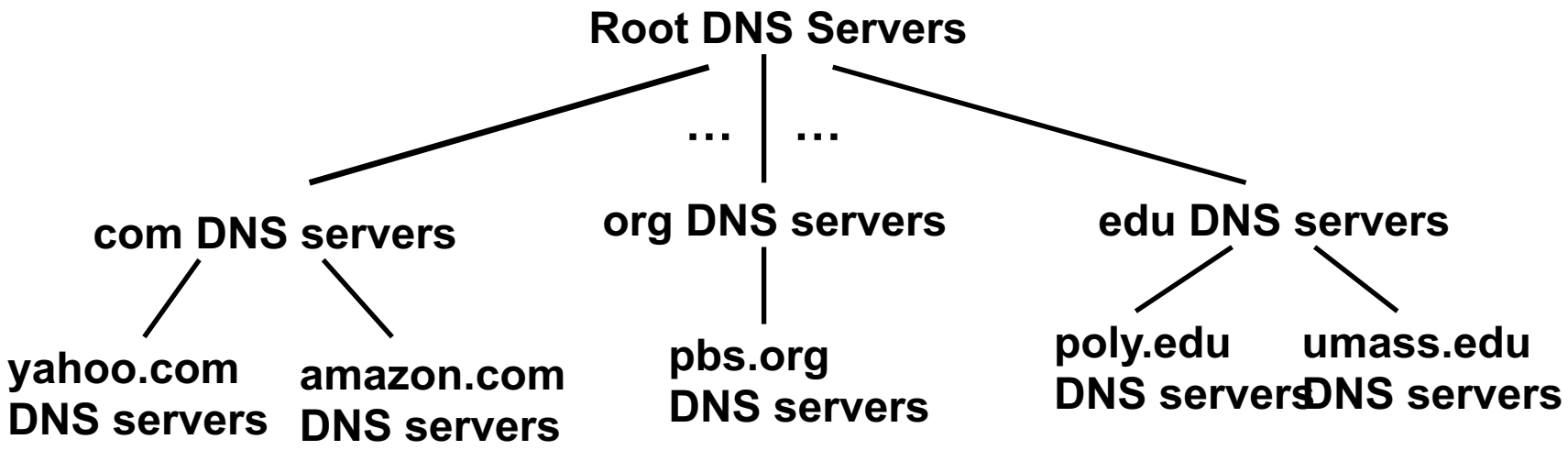


Domain Name System (DNS)

- Internet hosts and routers use IP addresses
  - Difficult for humans to remember and use
- "Names" are easier to use
  - Example: www.umass.edu, www.amazon.com, etc.
- Domain Name System maps from IP to name
  - Distributed database
  - Application layer protocol (used to find network layer IP)
  - Additional features: mail server info, load balancing, etc.

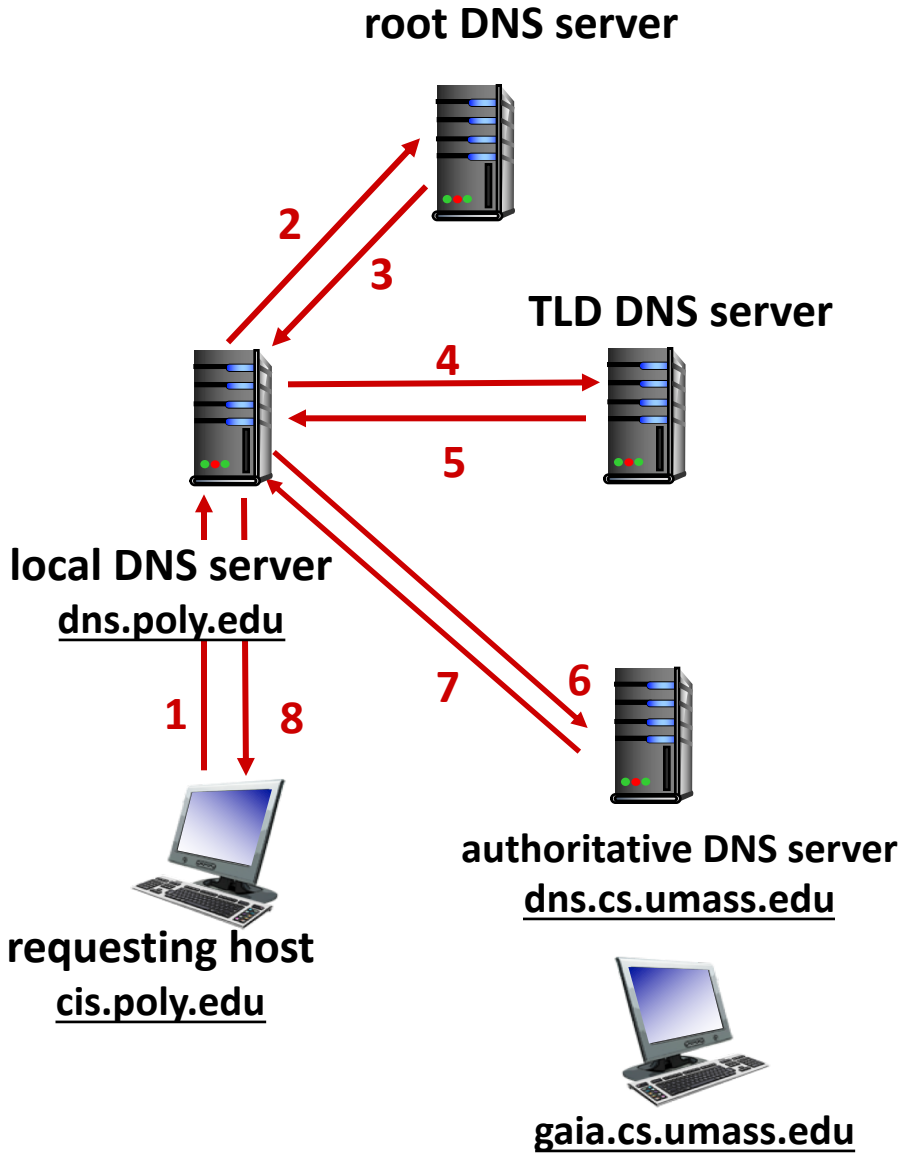
DNS Structure

- Hierarchical structure of names
  - Later parts of names are higher in hierarchy



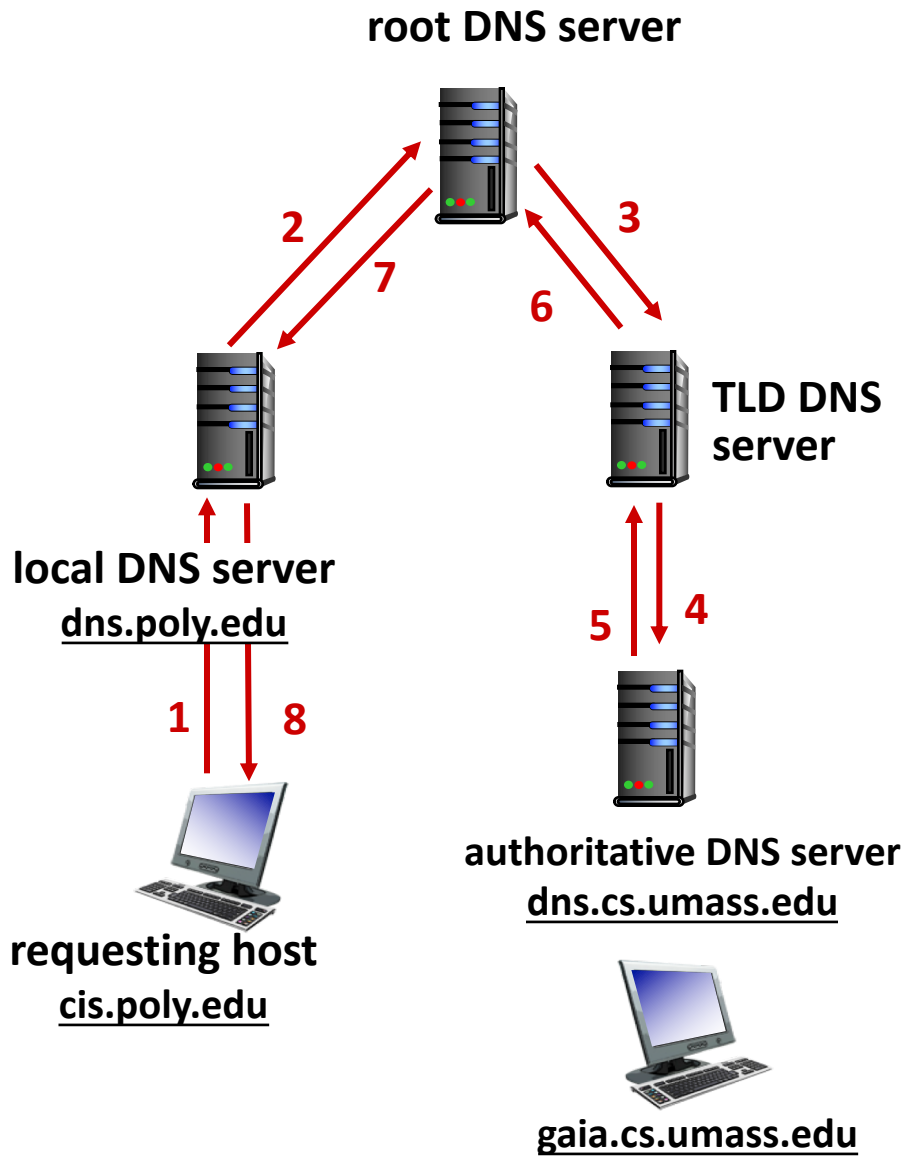
DNS Example

- DNS example: iterative query
  - Server responds: "I don't know, ask this server."
  - Assumes only authoritative name server knows answer
  - In practice, results may be cached



DNS Example

- DNS example: recursive query
  - Server relays query and responds when result available
  - Assumes only authoritative name server knows answer
  - In practice, results may be cached

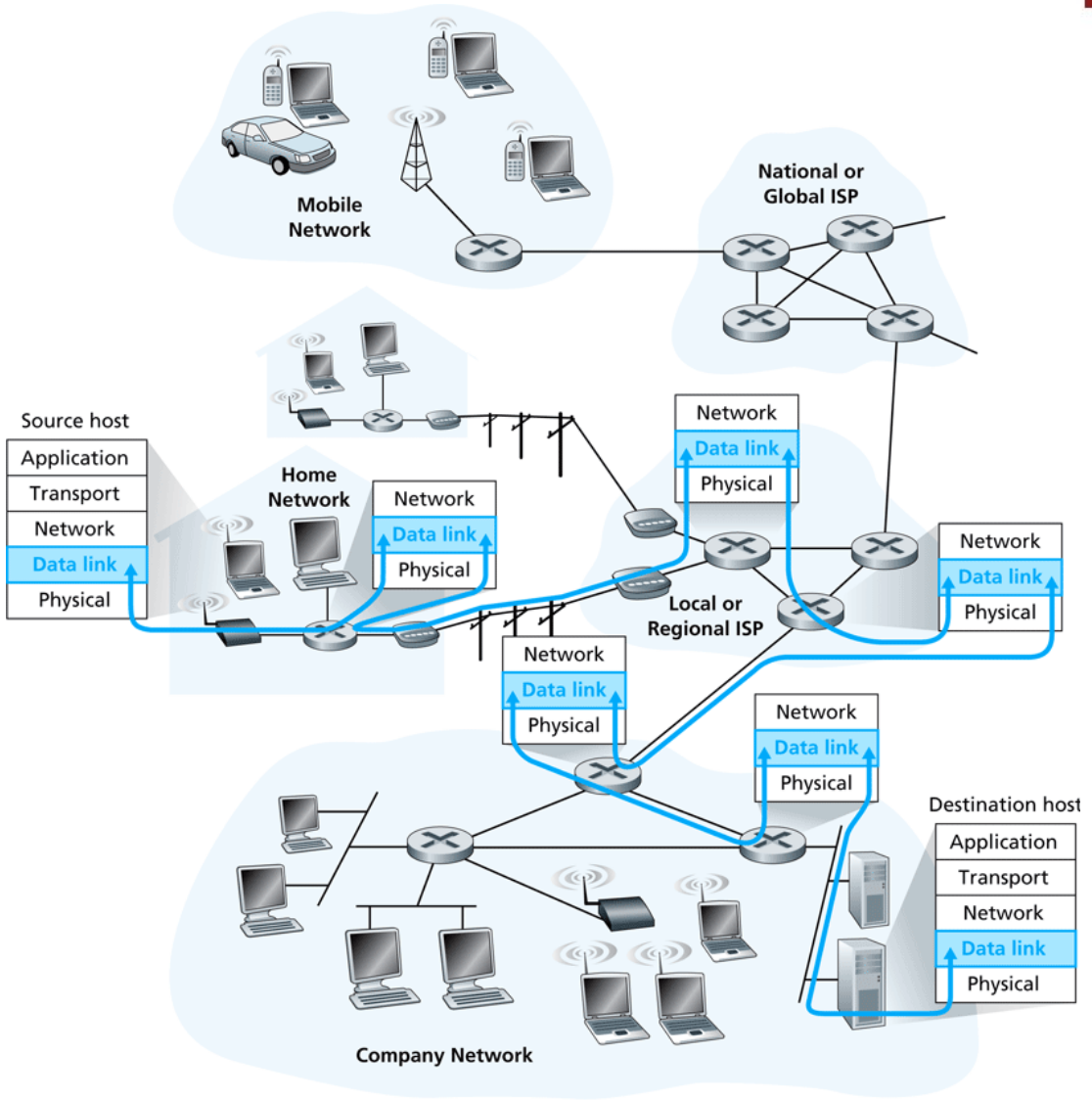


Topics

- **Introduction**
  - **Lessons 1 & 2**
- **Application Layer Protocols (HTTP, SMTP)**
  - Lesson 3
- **Transport Layer Protocols (TCP, UDP)**
  - Lesson 4
- **Transport Layer (Congestion Control)**
  - Lesson 5
- **Network Layer Protocol (IP)**
  - Lesson 6
- **Data Link Layer Protocol (Ethernet)**
  - Lesson 7
- **Software Defined Networks (SDN)**
  - Lesson 8

Data Link Layer

- Data links connect one device to another
  - Single network hop
  - “Neighboring” interfaces

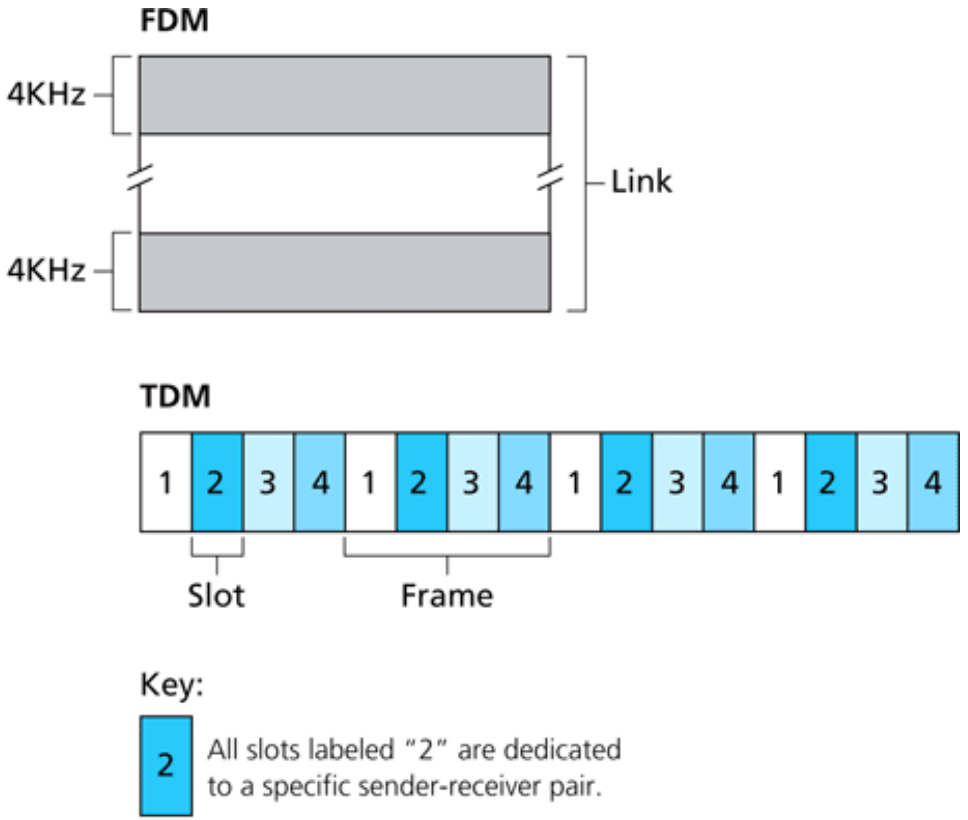


Link Access

- Point-to-point guided medium is straightforward
  - One side sends, other side receives (coding, timing, etc. is handled by physical layer)
  - Duplex operation by duplicating medium
- Multiple access case is more interesting
  - Multiple nodes share guided or unguided medium
  - Need to consider:
    - Naming
    - Medium access protocol
    - “Strange” cases (e.g., hidden terminal problem)

Multiple Access

- Medium Access Control Approaches:
  - Channel partitioning
    - TDM, FDM, CDMA
  - Random access protocols
    - ALOHA, CSMA/CD
  - Taking turns protocol
    - Polling, token-passing
- Channel partitioning
  - Discussed in circuit switching
- Random access used in Ethernet (CSMA/CD)
  - Widely used link layer protocol

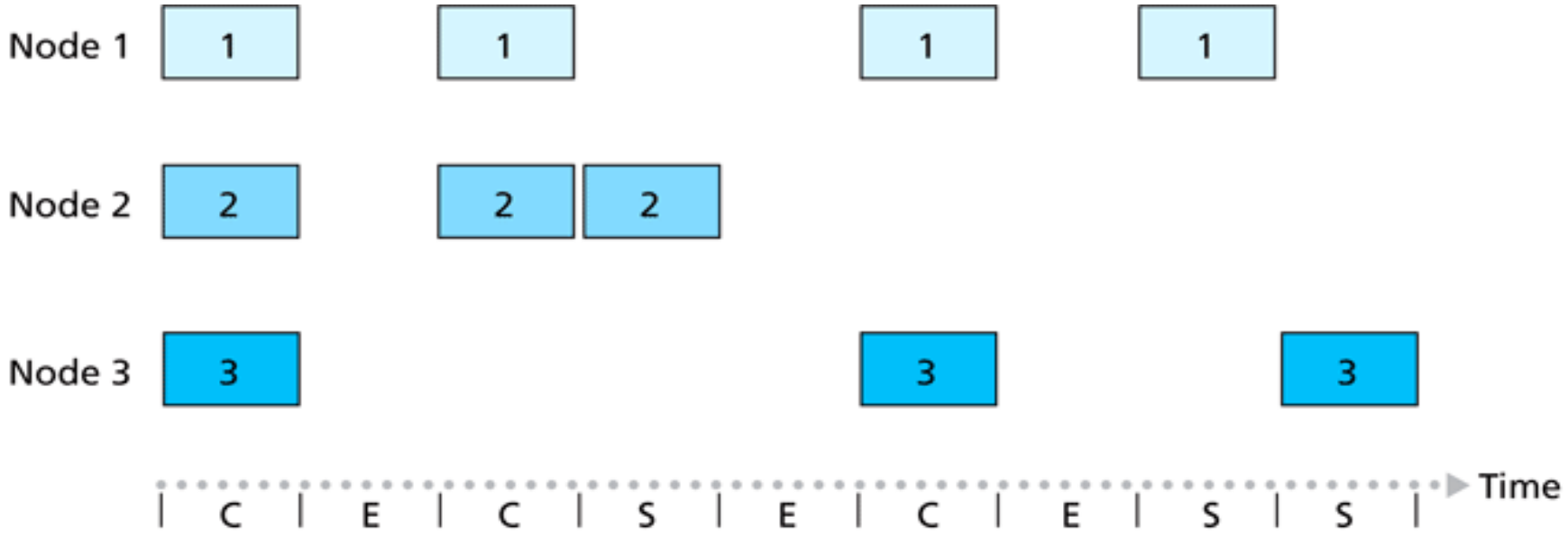


ALOHA Protocol

- ALOHA is one of first, simplest random access protocols
- “Slotted ALOHA” protocol
  - Discrete time slots
  - Each node makes random choice to transmit in slot or not
- “Pure ALOHA”
  - No time slots
  - Each note makes random choice to transmit at any time

Slotted ALOHA

- Multiple stations may send at the same time?
  - What happens if collision occurs?

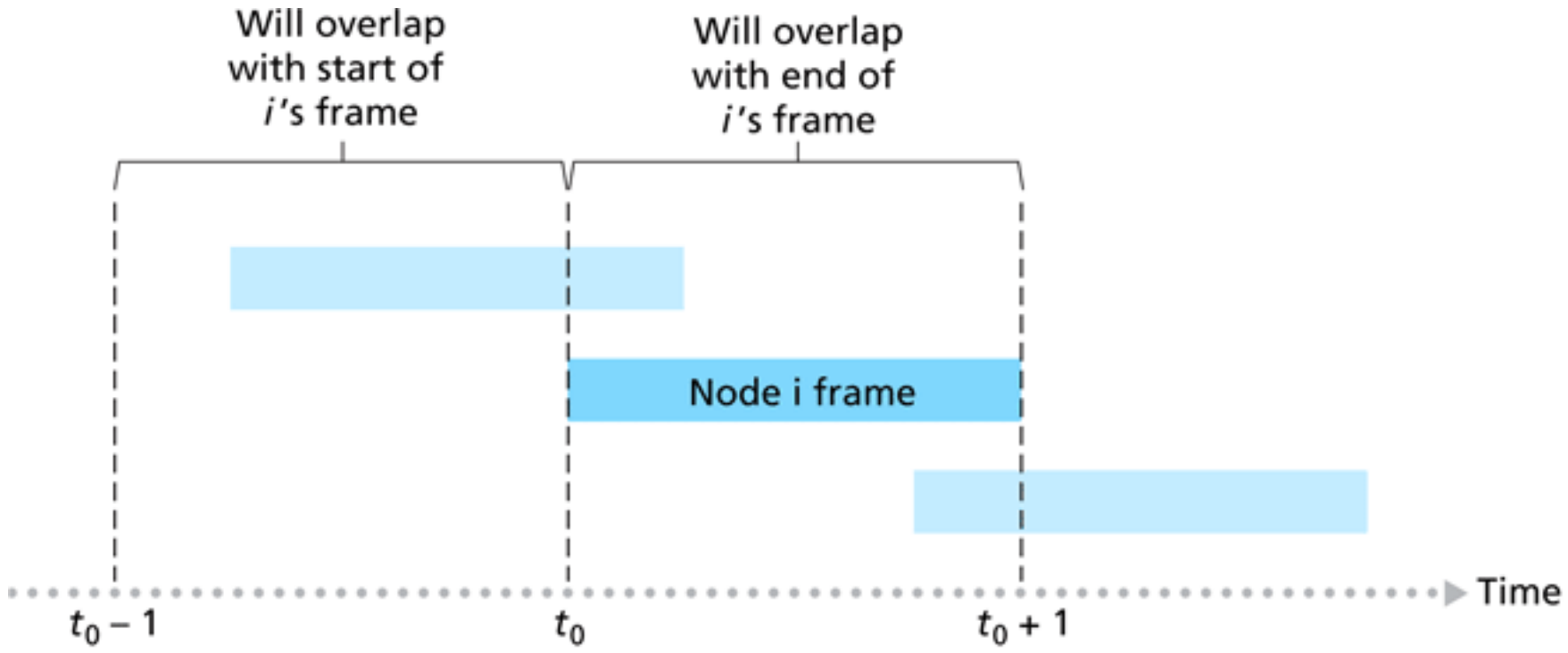


Key:

C = Collision slot
E = Empty slot
S = Successful slot

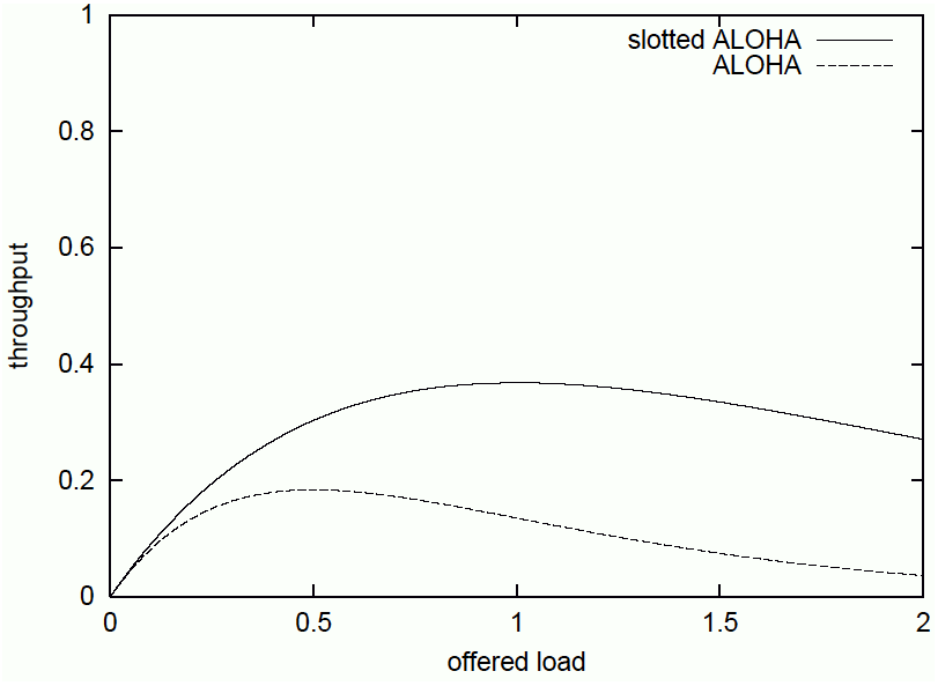
Pure ALOHA

- Station can send whenever it is ready
  - Avoids synchronization challenge in Slotted ALOHA



Analysis of ALOHA

- How does performance differ (pure vs. slotted)?
  - Maximum throughput:
    - Slotted ALOHA: 37%
      - At 100% network load
    - Pure ALOHA: 18%
      - At 50% network load
  - Note “collapse” at higher loads

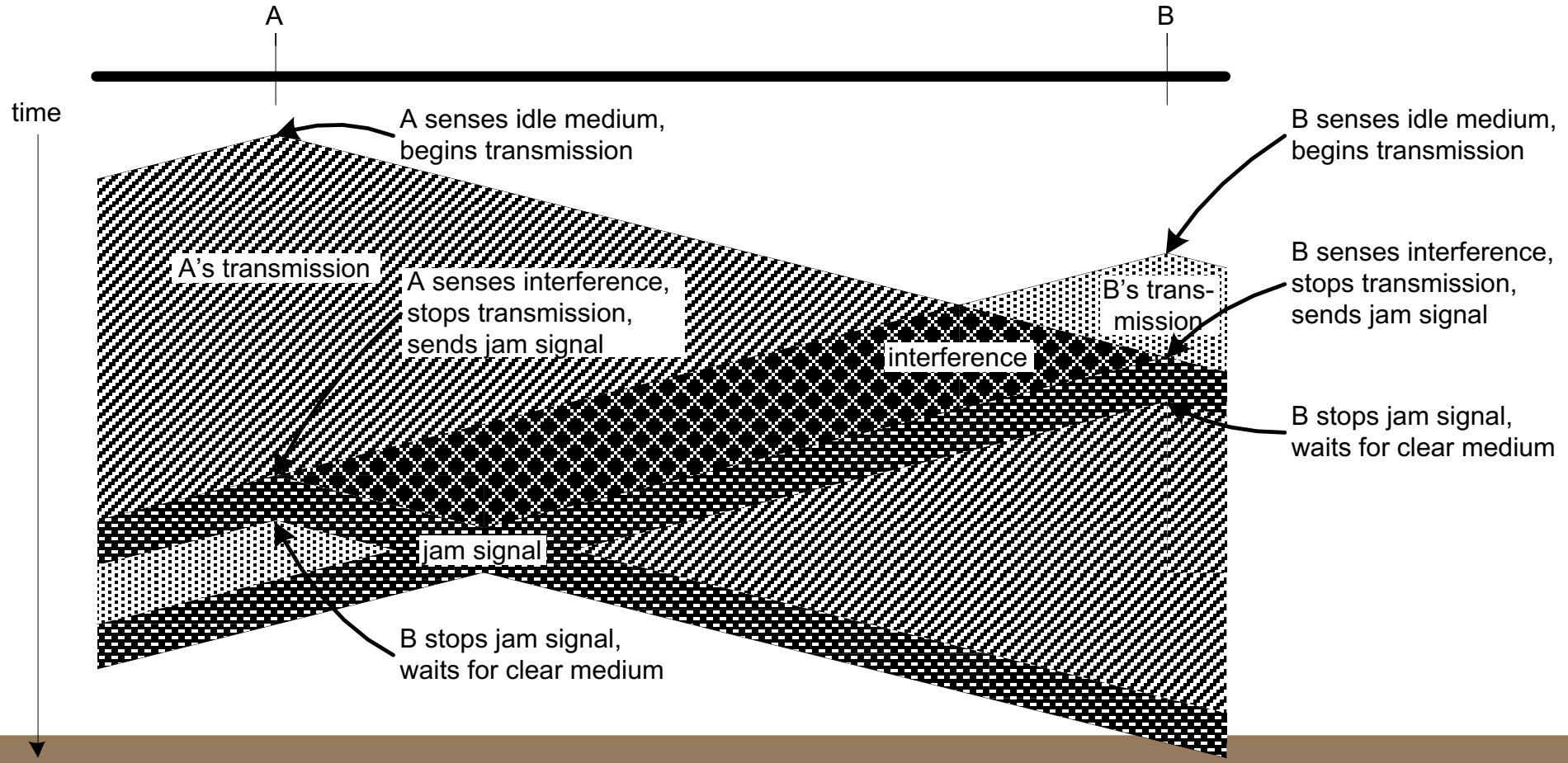


CSMA/CD Protocol

- How can we improve ALOHA?
  - Don’t send when somebody else has already started
  - Stop when interference is already happening
    - Why do we need to do that if we don’t start sending when somebody else sends?
- Carrier Sensing (CS)
  - Listen on channel
  - Only send when nobody else is transmitting
- Collision Detection (CD)
  - Listen to own transmission on channel
  - If garbled then stop transmitting

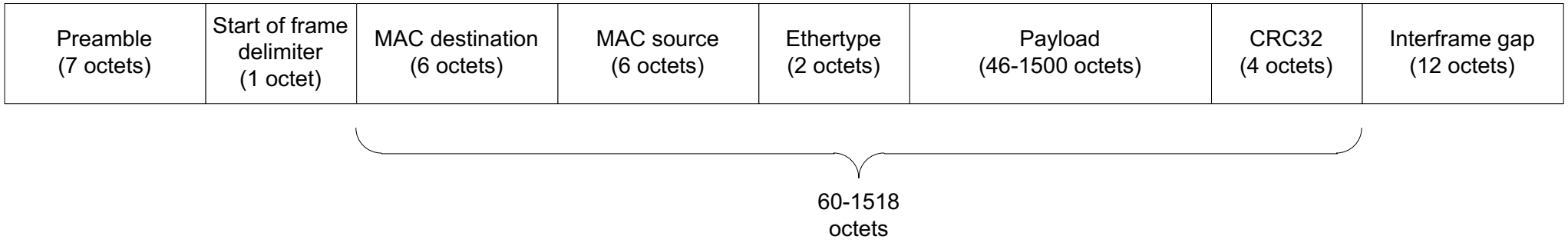
CSMA/CD Example

- **Complete example of CSMA/CD interaction:**



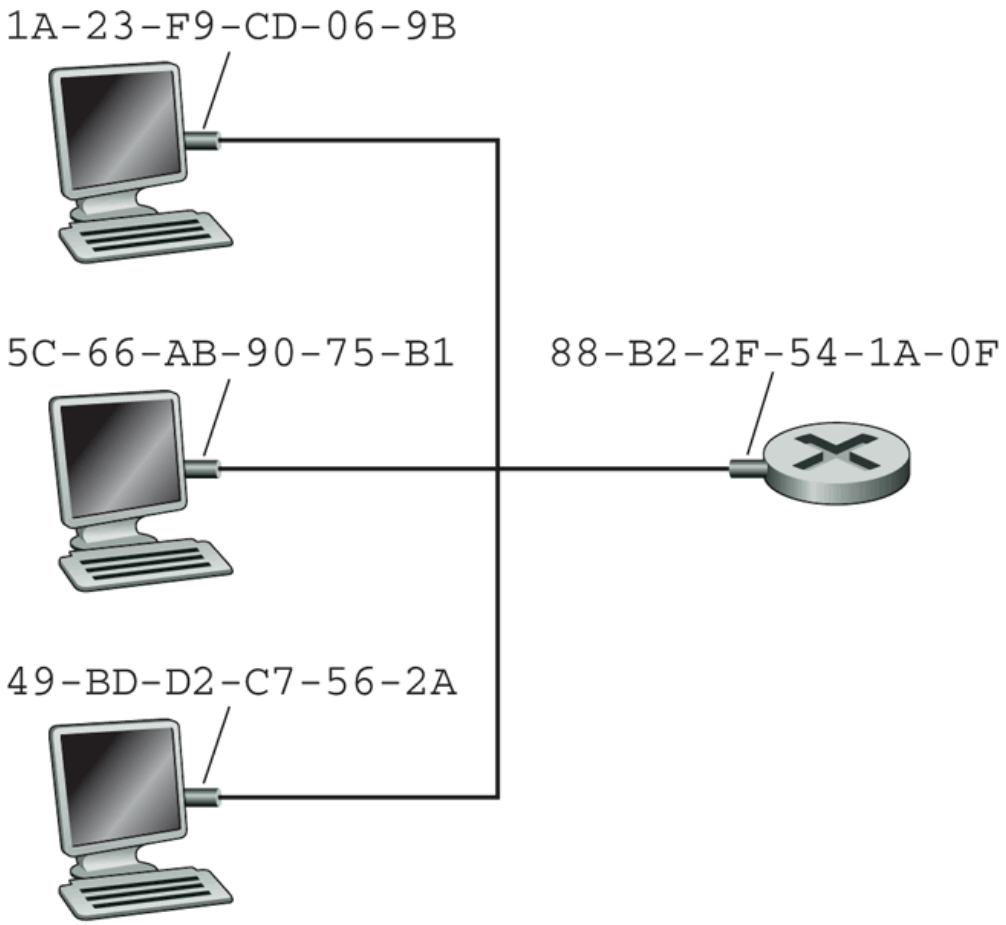
Ethernet

- Frame format:
  - Preamble synchronized receiver
  - Type identifies network layer protocol
  - CRC for error detection
- Addressing
  - Globally unique 48-bit address
  - First 24 bits identify vendor



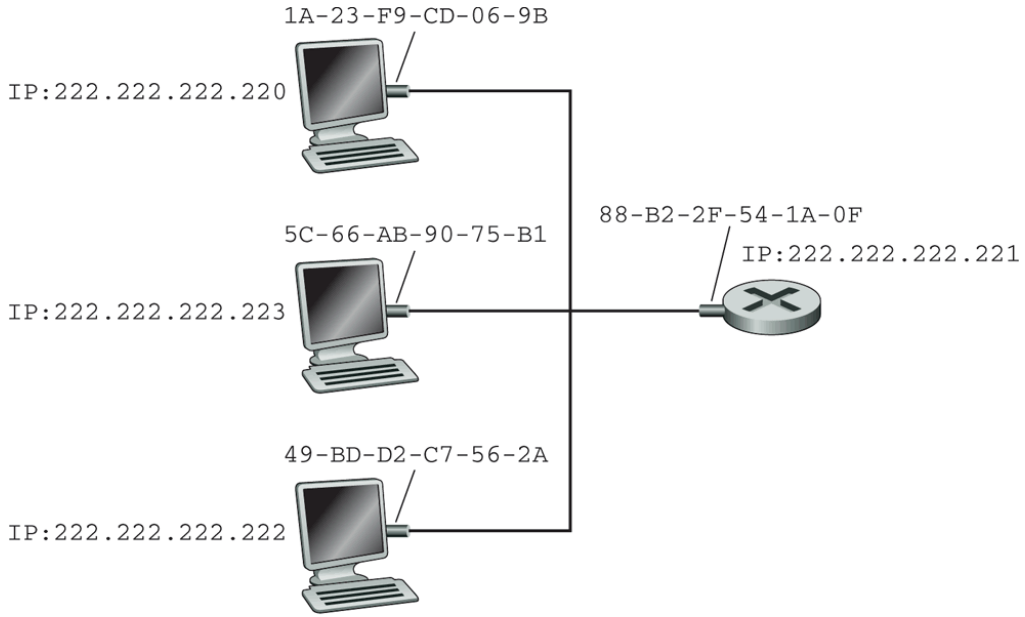
Ethernet

- Example of Ethernet addresses in network
  - Often referred to as “MAC addresses”
- How can node associate network layer (IP) addresses with MAC addresses?
  - MAC address structure does not follow hierarchical IP address allocation



Address Resolution Protocol

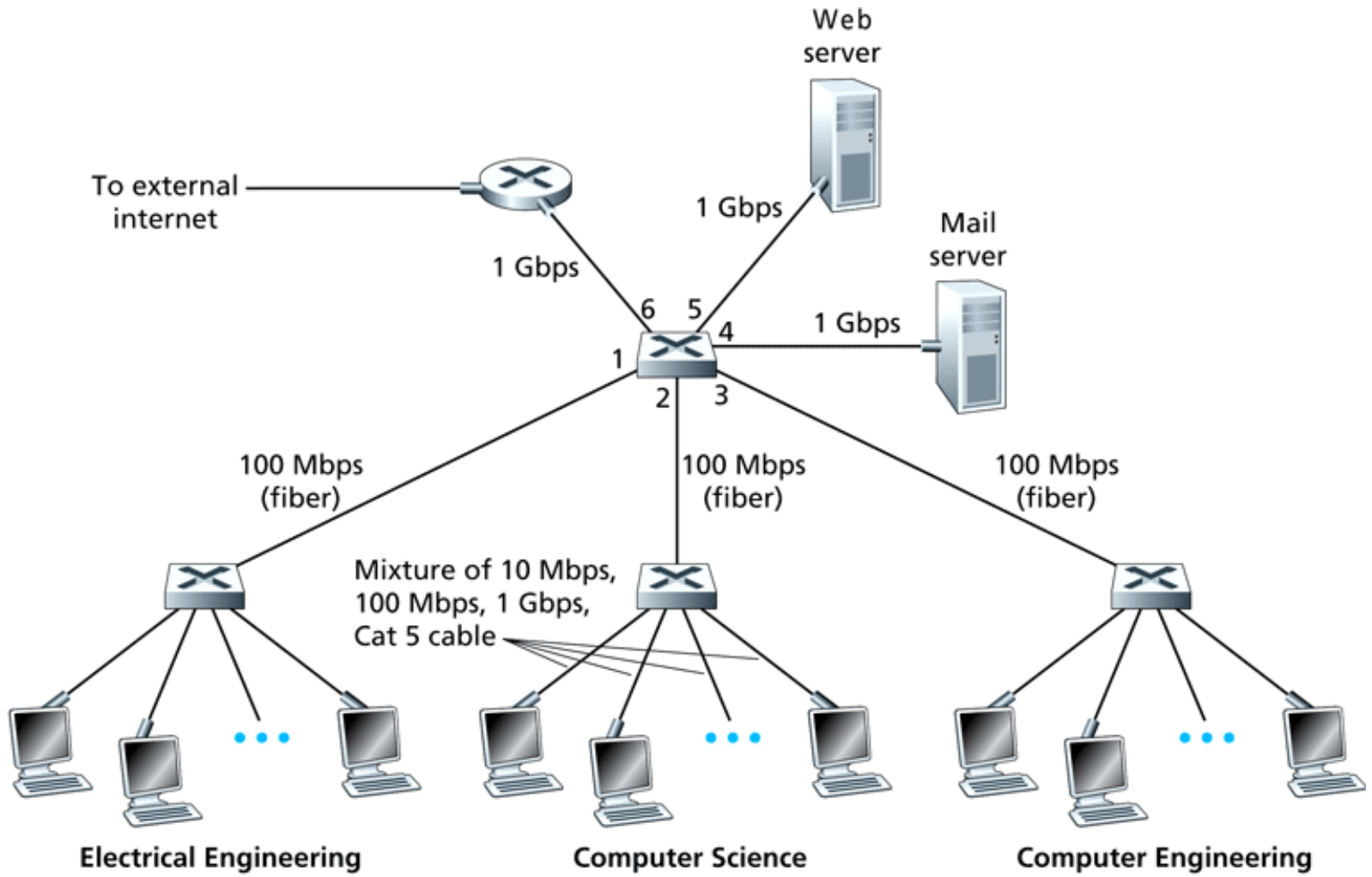
- Problem: what is Ethernet address of destination?
- ARP protocol
  - Broadcast of request for Ethernet address of given IP address
  - Response from anybody
- ARP table maintains entries
  - Timeout ensures adaptation to reconfigurations



IP Address	MAC Address	TTL
222.222.222.221	88-B2-2F-54-1A-0F	13:45:00
222.222.222.223	5C-66-AB-90-75-B1	13:52:00

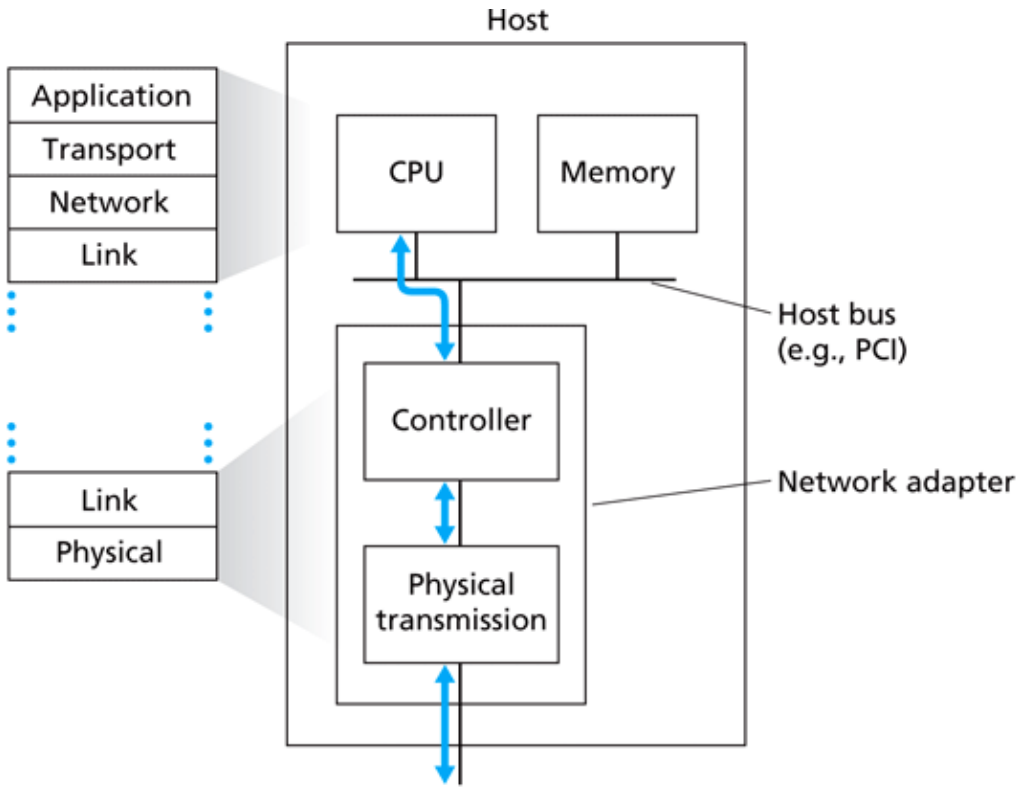
LAN Architecture

- **Example with heterogeneous links**



Network Interface Cards

- Implementation of NIC
  - Adapter connected to processor via bus
- NIC implements many functions in hardware
  - Error detection
  - Retransmission
  - Direct Memory Access (DMA)
- More later…



Topics

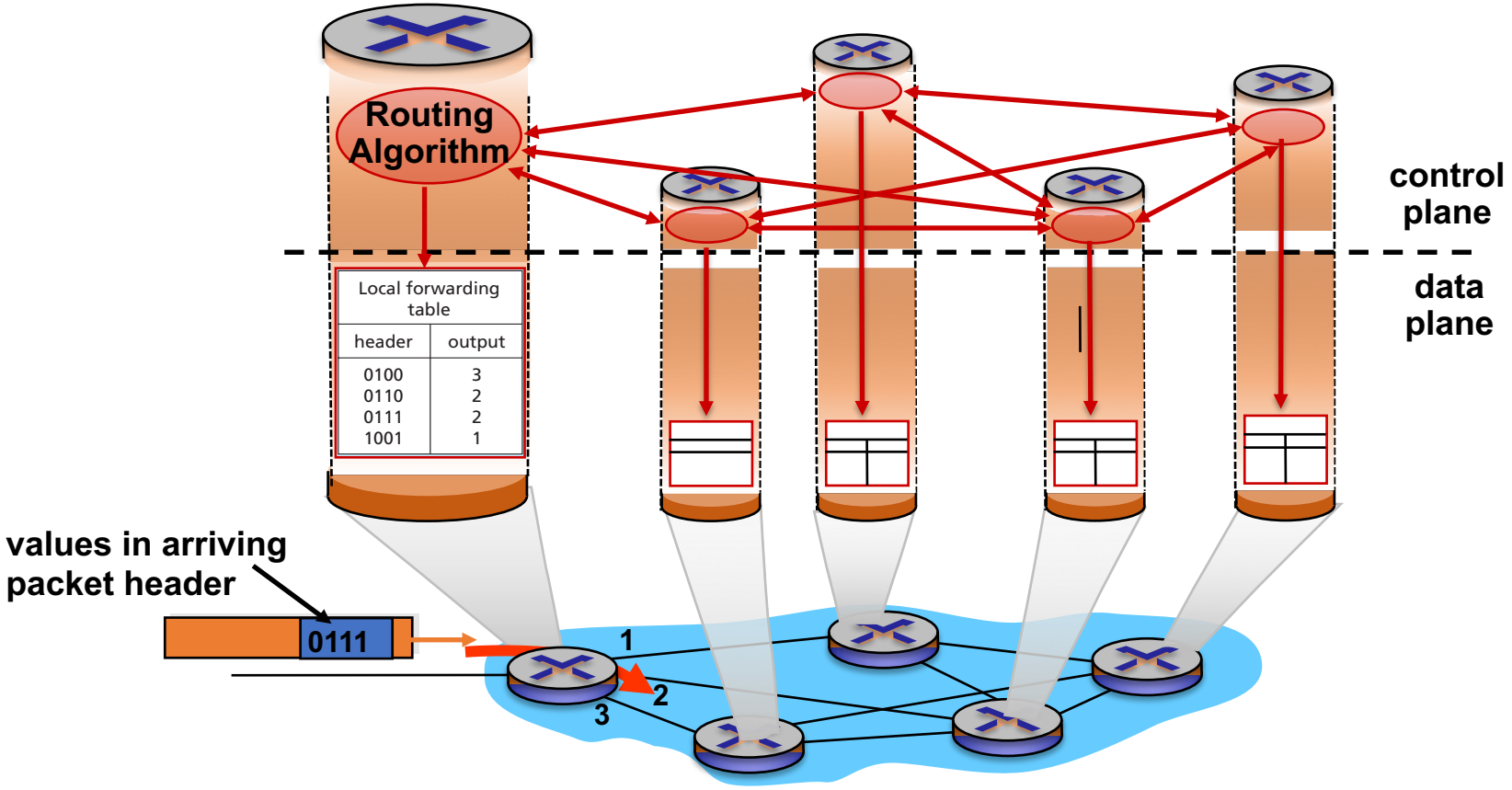
- **Introduction**
  - **Lessons 1 & 2**
- **Application Layer Protocols (HTTP, SMTP)**
  - Lesson 3
- **Transport Layer Protocols (TCP, UDP)**
  - Lesson 4
- **Transport Layer (Congestion Control)**
  - Lesson 5
- **Network Layer Protocol (IP)**
  - Lesson 6
- **Data Link Layer Protocols (Ethernet)**
  - Lesson 7
- **Software Defined Networks (SDN)**
  - Lesson 8

Software-Defined Networks

- Software-defined networking (SDN) is emerging technology
  - Different approach to controlling traffic in network
  - Crosses multiple layers in protocol stack
- Traditional IP networks
  - Distributed routing
- SDN
  - Logically centralized routing
  - Enables more control over network traffic

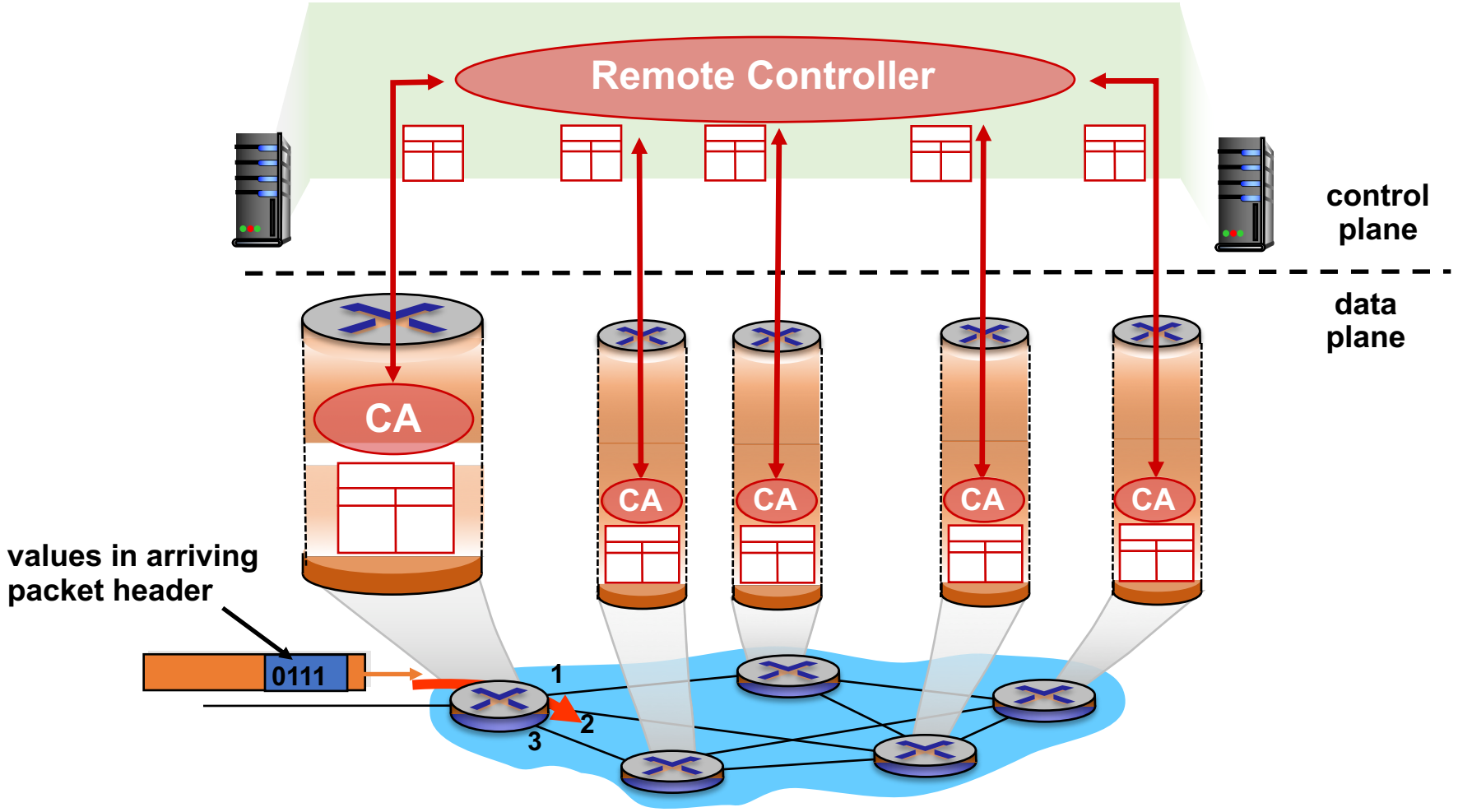
Per-Router Control Plane

- Individual routing algorithm components in each and every router interact in the control plane



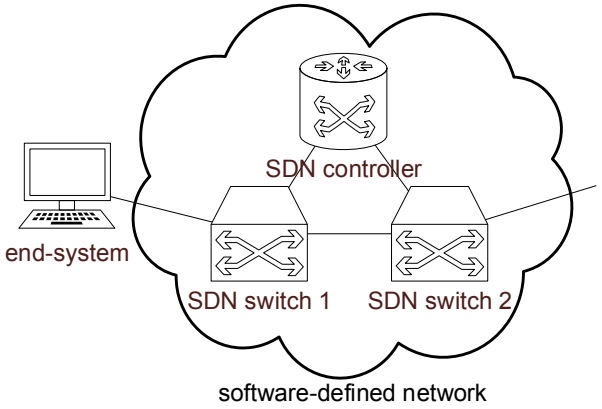
Logically Centralized Control Plane

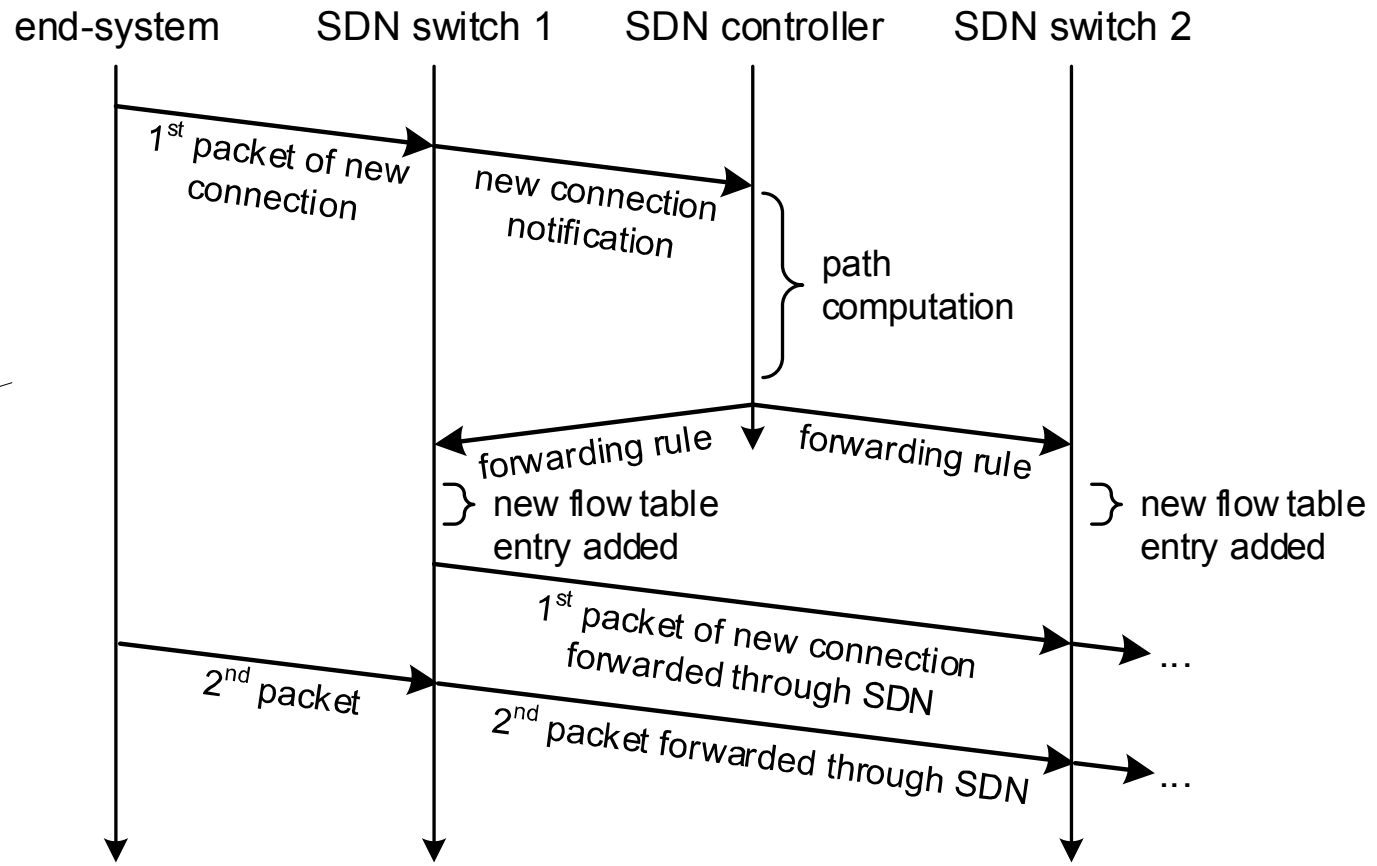
- A distinct (typically remote) controller interacts with local control agents (CAs)



Software-Define Networks

- **Space-time diagram of packet forwarding:**





Programmable Networks

- Network Operating System as extension of control
  - Different “applications” can control network traffic
  - SDN functionality can control individual flows