



COS 461: Computer Networks

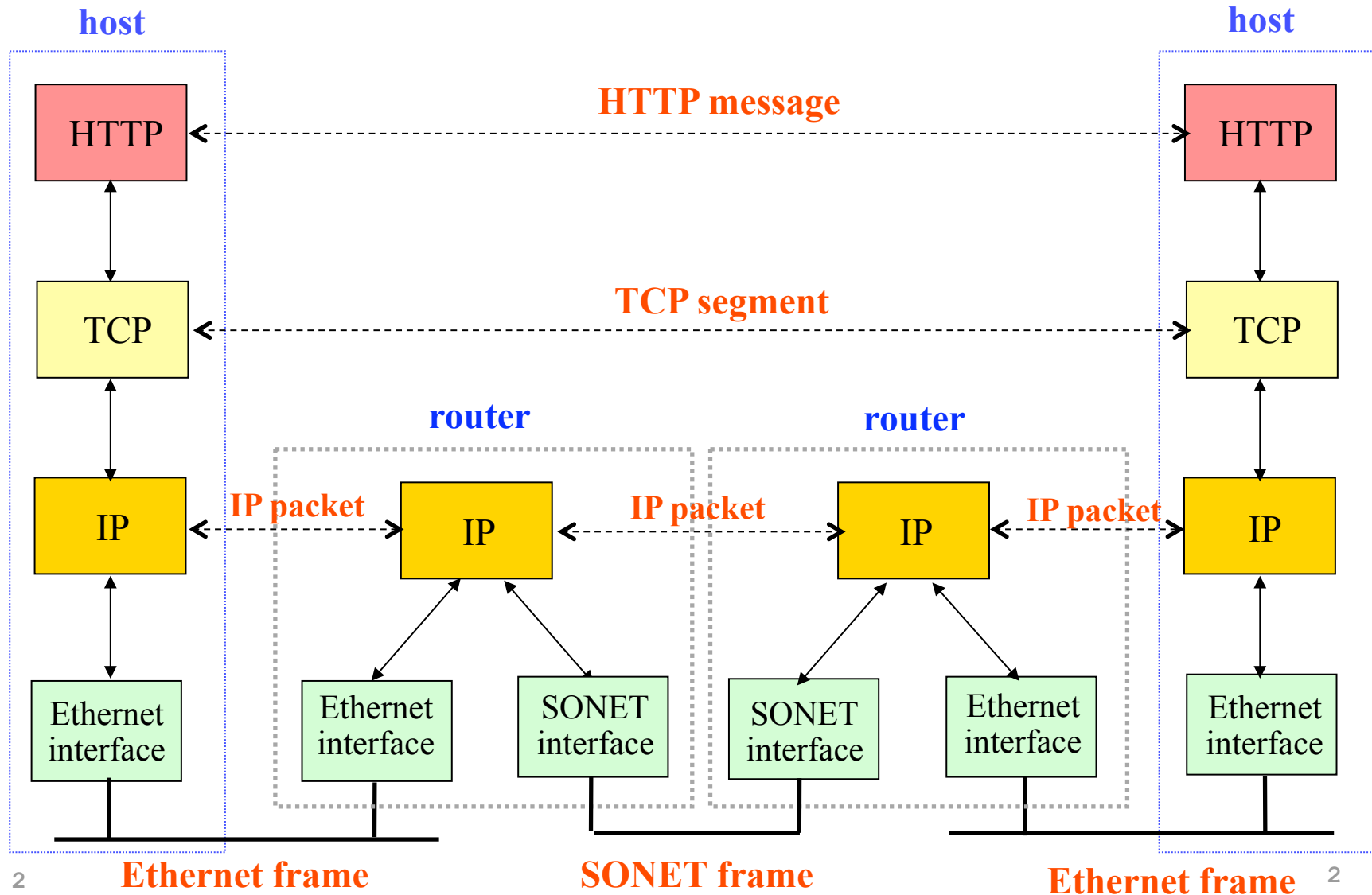
Midterm Review

Spring 2011

Mike Freedman

<http://www.cs.princeton.edu/courses/archive/spr11/cos461/>

Internet layering: Message, Segment, Packet, and Frame



Topics

- **Link layer (Sl.4)**
 - Sharing a link: TDMA, FDMA
 - Ethernet and CSMA/CD
 - Wireless and CSMA/CA
 - Spanning tree and switching
 - Translating addrs: DHCP / ARP
- **Network layer (Sl.25)**
 - IPv4 and addressing
 - IP forwarding
 - Middleboxes: NATs, firewalls, tunneling
- **Transport layer (Sl.38)**
 - Socket interface
 - UDP
 - TCP
 - Reliability
 - Congestion Control
 - Interactions w/ Active Queue Management
- **Application layer (Sl.68)**
 - Translating names: DNS
 - HTTP and CDNs
 - Overlay networks

Link Layer

Link-Layer Services

- **Encoding**
 - Representing the 0s and 1s
- **Framing**
 - Encapsulating packet into frame, adding header and trailer
 - Using MAC addresses, rather than IP addresses
- **Error detection**
 - Errors caused by signal attenuation, noise.
 - Receiver detecting presence of errors

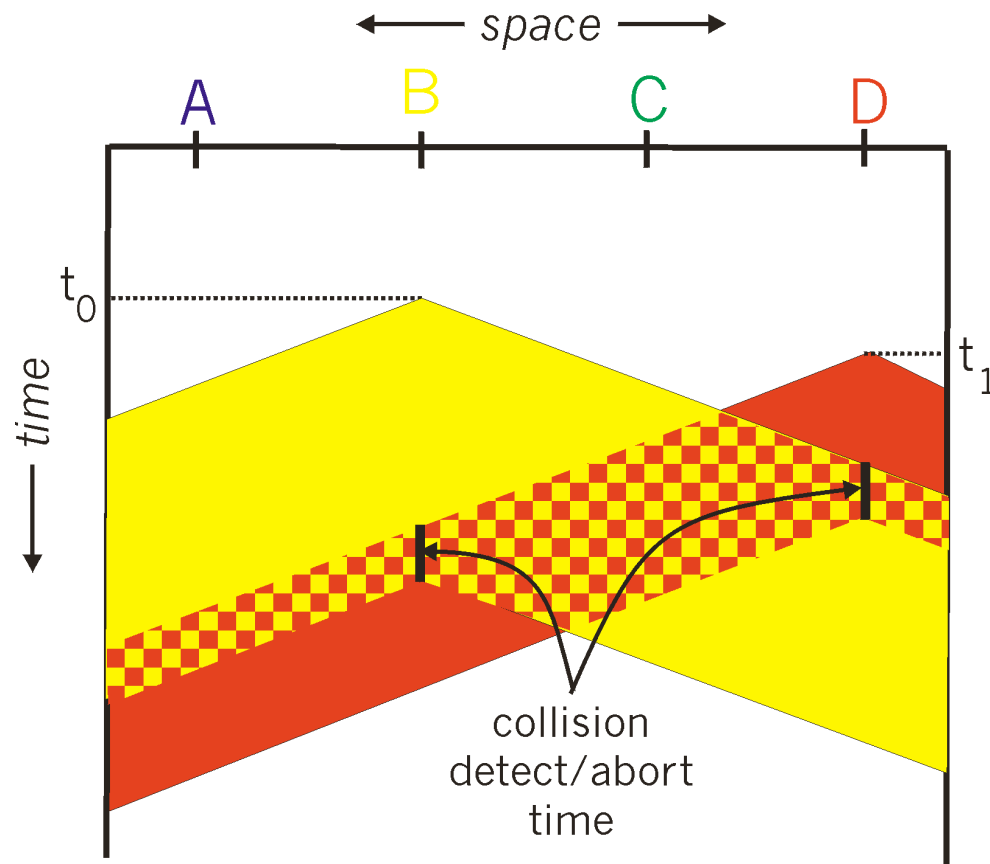
Multiple Access Protocol

- **Single shared broadcast channel**
 - Avoid having multiple nodes speaking at once
 - Otherwise, collisions lead to garbled data
- **Multiple access protocol**
 - Distributed algorithm for sharing the channel
 - Algorithm determines which node can transmit
- **Classes of techniques**
 - Channel partitioning: divide channel into pieces
 - Time-division multiplexing, frequency division multiplexing
 - Taking turns: passing a token for right to transmit
 - Random access: allow collisions, and then recover

Key Ideas of Random Access

- **Carrier Sense (CS)**
 - *Listen before speaking, and don't interrupt*
 - Checking if someone else is already sending data
 - ... and waiting till the other node is done
- **Collision Detection (CD)**
 - *If someone else starts talking at the same time, stop*
 - Realizing when two nodes are transmitting at once
 - ...by detecting that the data on the wire is garbled
- **Randomness**
 - *Don't start talking again right away*
 - Waiting for a random time before trying again

CSMA/CD Collision Detection



Medium Access Control in 802.11

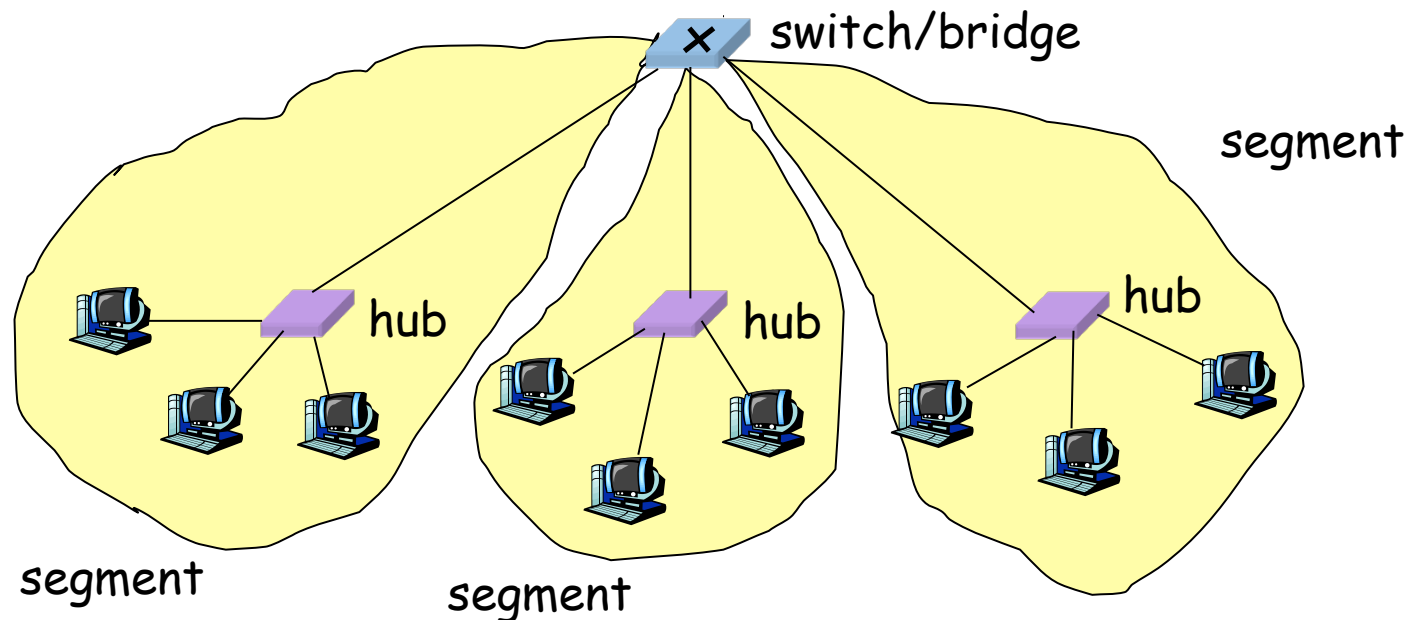
- **Collision avoidance, not detection**
 - First exchange control frames before transmitting data
 - Sender issues “Request to Send” (RTS), including length of data
 - Receiver responds with “Clear to Send” (CTS)
 - If sender sees CTS, transmits data (of specified length)
 - If other node sees CTS, will idle for specified period
 - If other node sees RTS but not CTS, free to send
- **Link-layer acknowledgment and retransmission**
 - CRC to detect errors
 - Receiving station sends an acknowledgment
 - Sending station retransmits if no ACK is received
 - Giving up after a few failed transmissions

Scaling the Link Layer

- Ethernet traditionally limited by fading signal strength in long wires
 - Introduction of hubs/repeaters to rebroadcast
- Still a maximum “length” for a Ethernet segment
 - Otherwise, two nodes might be too far for carrier sense to detect concurrent broadcasts
- Further, too many nodes in shorter Ethernet can yield low transmissions rates
 - Constantly conflict with one another

Bridges/Switches: Traffic Isolation

- Switch breaks subnet into LAN segments
- Switch filters packets
 - Frame only forwarded to the necessary segments
 - Segments can support separate transmissions

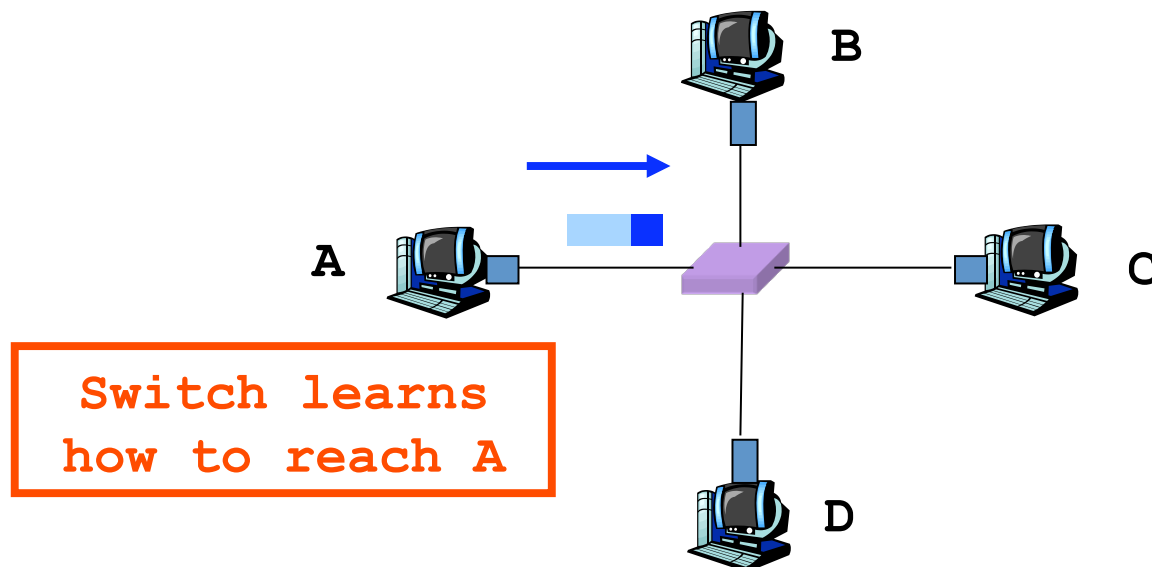


Comparing Hubs, Switches, Routers

	Hub/ Repeater	Bridge/ Switch	Router
Traffic isolation	no	yes	yes
Plug and Play	yes	yes	no
Efficient routing	no	no	yes
Cut through	yes	yes	no

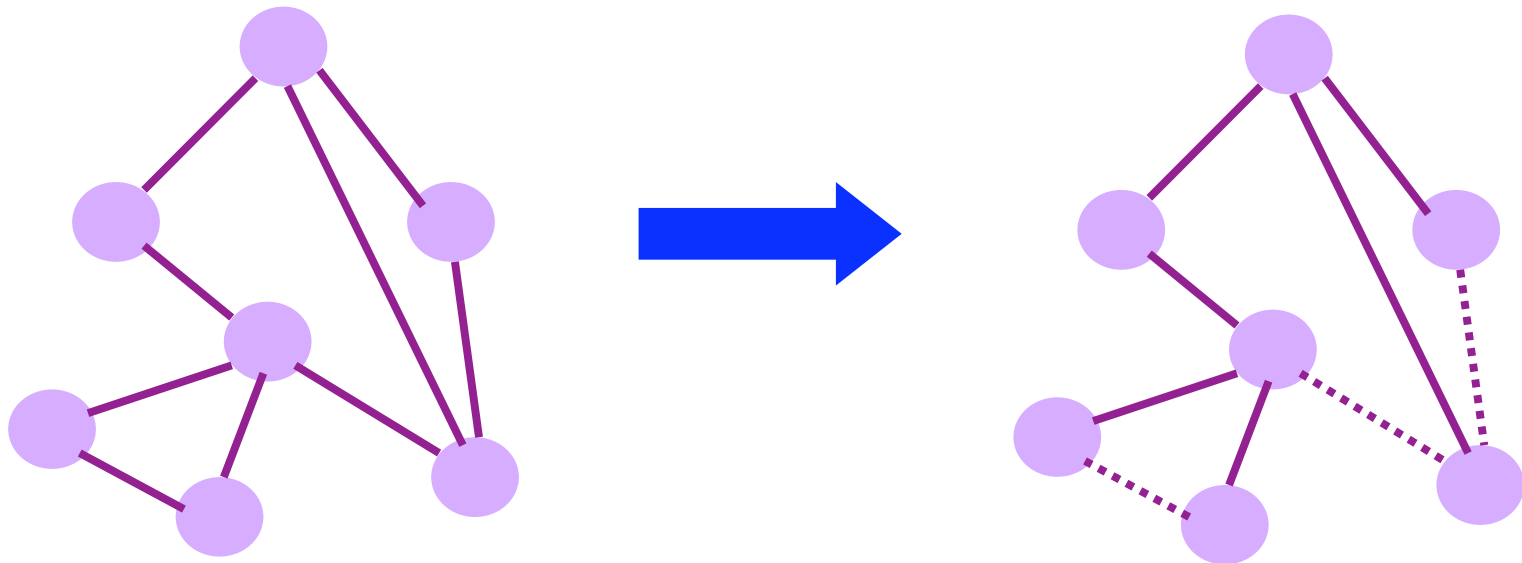
Self Learning: Building the Table

- When a frame arrives
 - Inspect the *source* MAC address
 - Associate the address with the *incoming* interface
 - Store the mapping in the switch table
 - Use a time-to-live field to eventually forget the mapping

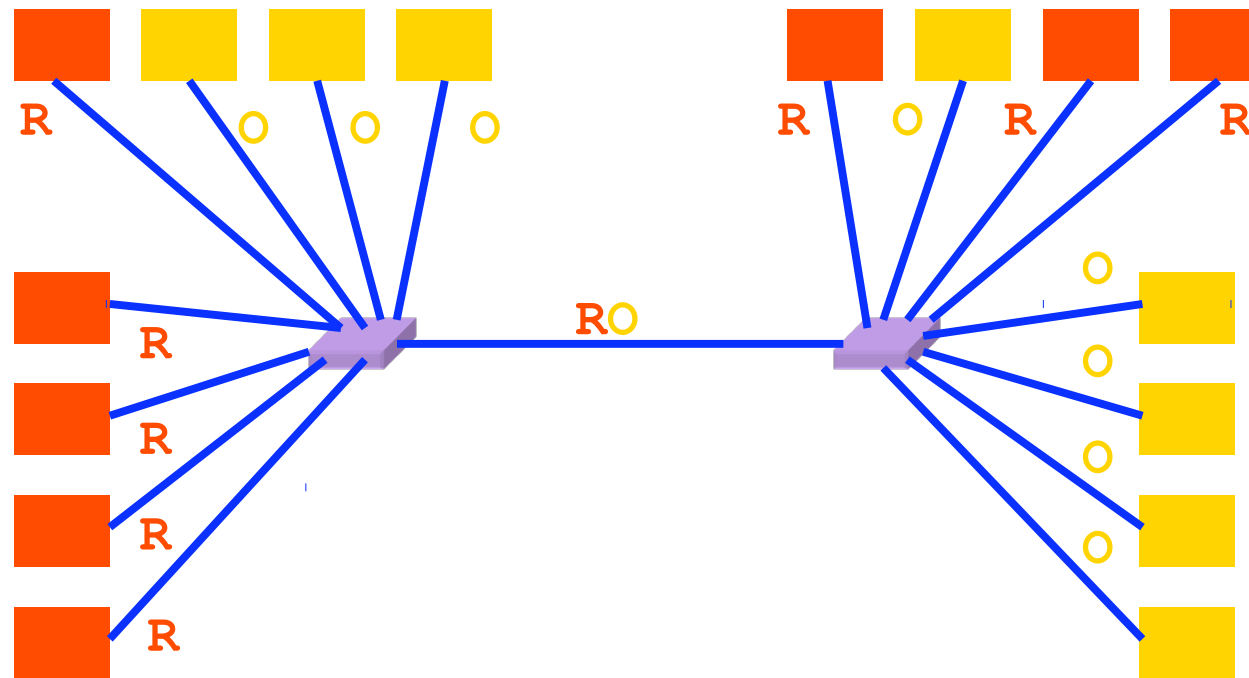


Solution: Spanning Trees

- Ensure the topology has no loops
 - Avoid using some of the links when flooding
 - ... to avoid forming a loop
- Spanning tree
 - Sub-graph that covers all vertices but contains no cycles
 - Links not in the spanning tree do not forward frames



Evolution Toward Virtual LANs



Red VLAN and Orange VLAN

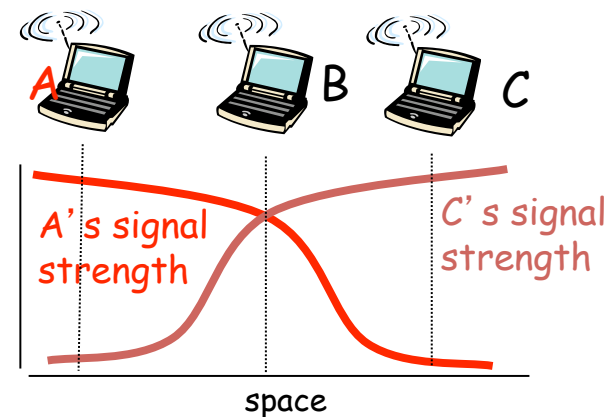
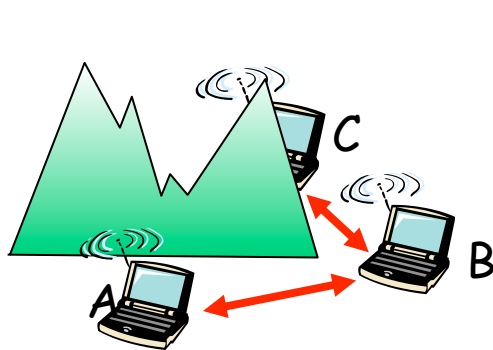
Switches forward traffic as needed

Group users based on organizational structure, rather than the physical layout of the building.

Wireless

CSMA: Carrier Sense, Multiple Access

- **Multiple access: channel is shared medium**
 - Station: wireless host or access point
 - Multiple stations may want to transmit at same time
- **Carrier sense: sense channel before sending**
 - Station doesn't send when channel is busy
 - To prevent collisions with ongoing transfers
 - But, detecting ongoing transfers isn't always possible



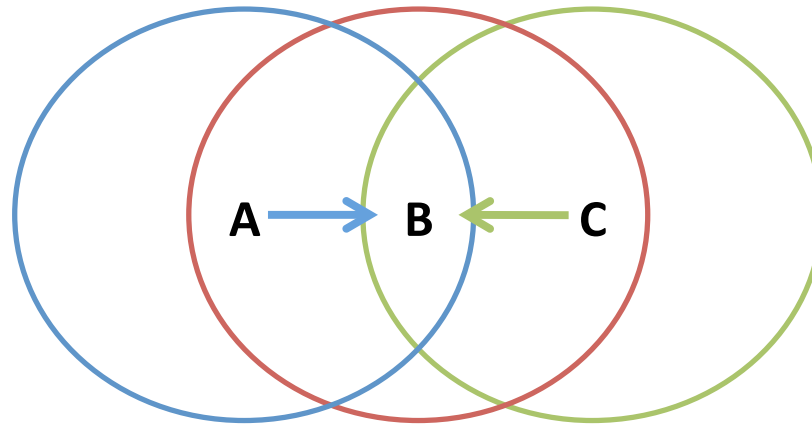
CA: Collision Avoidance, Not Detection

- **Collision detection in wired Ethernet**
 - Station listens while transmitting
 - Detects collision with other transmission
 - Aborts transmission and tries sending again
- **Problem #1: cannot detect all collisions**
 - Hidden terminal problem
 - Fading

CA: Collision Avoidance, Not Detection

- **Collision detection in wired Ethernet**
 - Station listens while transmitting
 - Detects collision with other transmission
 - Aborts transmission and tries sending again
- **Problem #1: cannot detect all collisions**
 - Hidden terminal problem
 - Fading
- **Problem #2: listening while sending**
 - Strength of received signal is much smaller
 - Expensive to build hardware that detects collisions
- **So, 802.11 does collision avoidance, not detection**

Hidden Terminal Problem

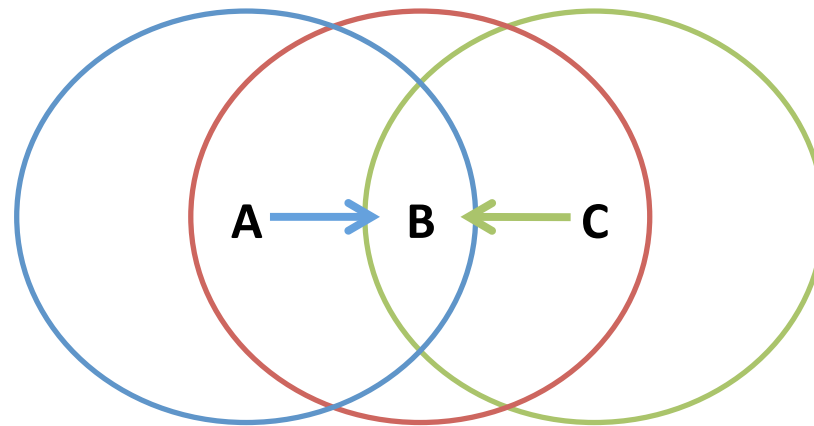


- A and C can't see each other, both send to B
- Occurs b/c 802.11 relies on physical carrier sensing, which is susceptible to hidden terminal problem

Virtual carrier sensing

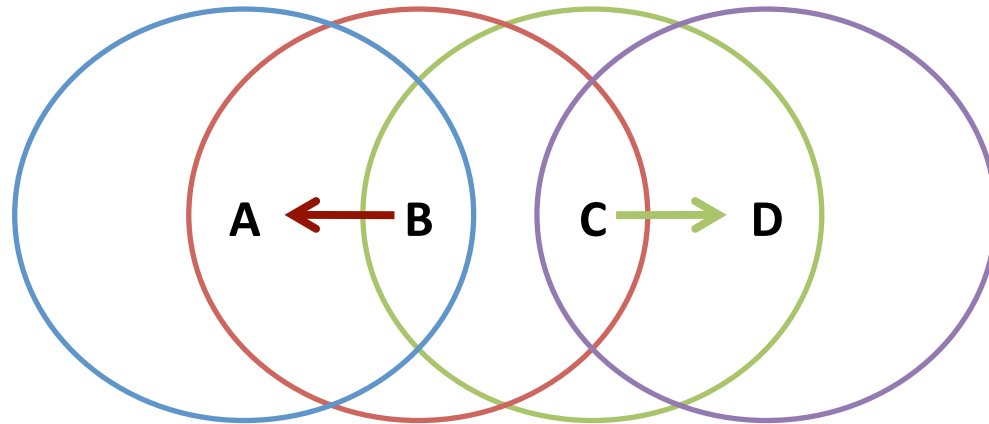
- First exchange control frames before transmitting data
 - Sender issues “Request to Send” (RTS), incl. length of data
 - Receiver responds with “Clear to Send” (CTS)
- If sender sees CTS, transmits data (of specified length)
- If other node sees CTS, will idle for specified period
- If other node sees RTS but not CTS, free to send

Hidden Terminal Problem



- A and C can't see each other, both send to B
- RTS/CTS can help
 - Both A and C would send RTS that B would see first
 - B only responds with one CTS (say, echo'ing A's RTS)
 - C detects that CTS doesn't match and won't send

Exposed Terminal Problem



- B sending to A, C wants to send to D
- As C receives B's packets, carrier sense would prevent it from sending to D, even though wouldn't interfere
- RTS/CTS can help
 - C hears RTS from B, but not CTS from A
 - C knows it's transmission will not interfere with A
 - C is safe to transmit to D

Impact on Higher-Layer Protocols

- **Wireless and mobility change path properties**
 - Wireless: higher packet loss, not from congestion
 - Mobility: transient disruptions, and changes in RTT
- **Logically, impact should be minimal ...**
 - Best-effort service model remains unchanged
 - TCP and UDP can (and do) run over wireless, mobile
- **But, performance definitely *is* affected**
 - TCP treats packet loss as a sign of congestion
 - TCP tries to estimate the RTT to drive retransmissions
 - TCP does not perform well under out-of-order packets
- **Internet not designed with these issues in mind**

Network Layer

IP Packet Structure

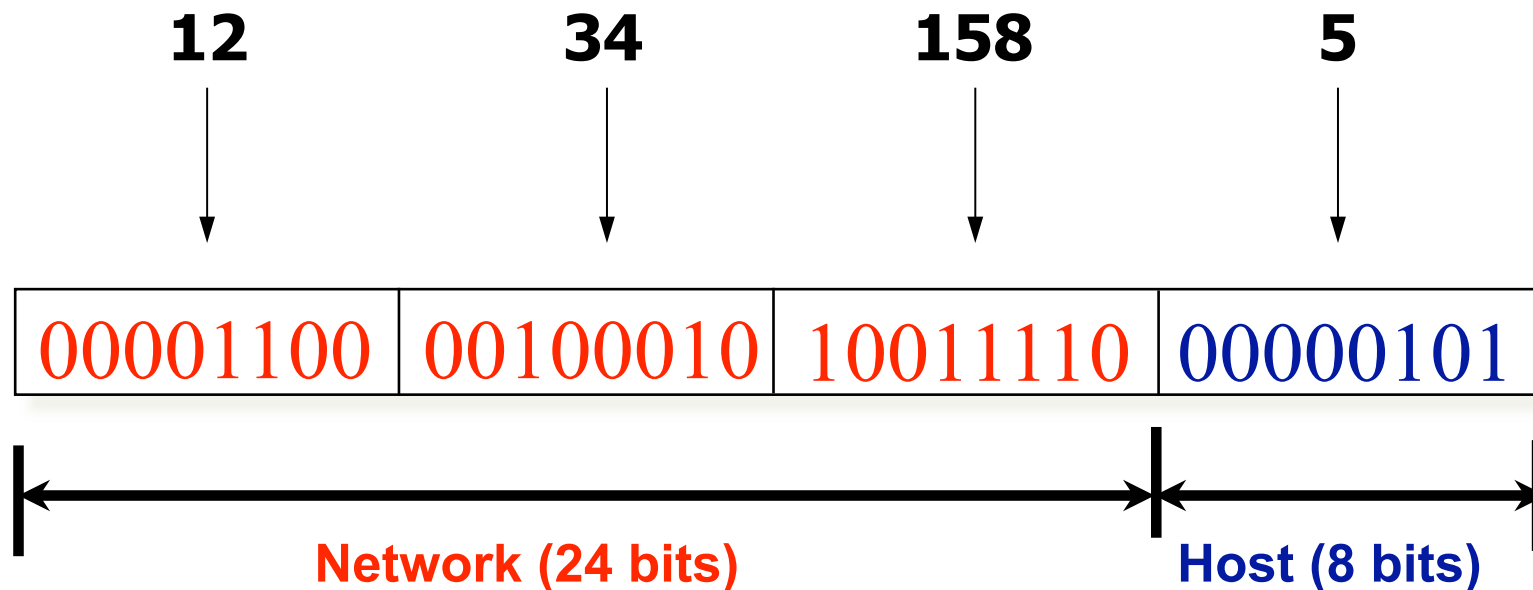
4-bit Version	4-bit Header Length	8-bit Type of Service (TOS)	16-bit Total Length (Bytes)	
16-bit Identification			3-bit Flags	13-bit Fragment Offset
8-bit Time to Live (TTL)	8-bit Protocol		16-bit Header Checksum	
32-bit Source IP Address				
32-bit Destination IP Address				
Options (if any)				
Payload				

Source Address: What if Source Lies?

- Source address should be the sending host
 - But, who's checking, anyway?
 - You could send packets with any source you want
- Why would someone want to do this?
 - Launch a denial-of-service attack
 - Send excessive packets to the destination
 - ... to overload the node, or the links leading to node
 - Evade detection by “spoofing”
 - But, the victim could identify you by the source address
 - So, you can put someone else's source address in packets
 - Also, an attack against the spoofed host
 - Spoofed host is wrongly blamed
 - Spoofed host may receive return traffic from receiver

Hierarchical Addressing: IP Prefixes

- IP addresses can be divided into two portions
 - Network (left) and host (right)
- 12.34.158.0/24 is a 24-bit **prefix**
 - Which covers 2^8 addresses (e.g., up to 255 hosts)

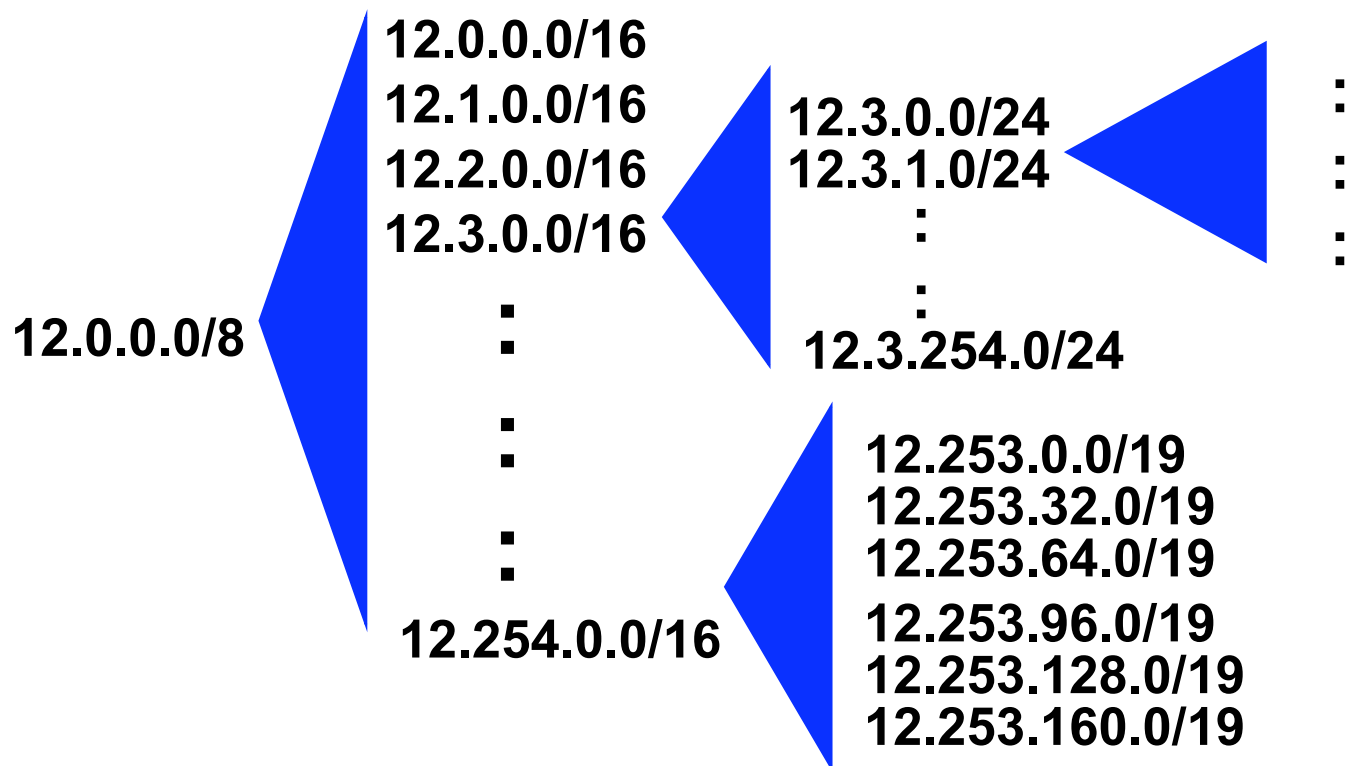


Classful Addressing

- In the olden days, only fixed allocation sizes
 - Class A: 0*
 - Very large /8 blocks (e.g., MIT has 18.0.0.0/8)
 - Class B: 10*
 - Large /16 blocks (e.g., Princeton has 128.112.0.0/16)
 - Class C: 110*
 - Small /24 blocks (e.g., AT&T Labs has 192.20.225.0/24)
 - Class D: 1110*
 - Multicast groups
 - Class E: 11110*
 - Reserved for future use
- This is why folks use dotted-quad notation!

CIDR: Hierarchal Address Allocation

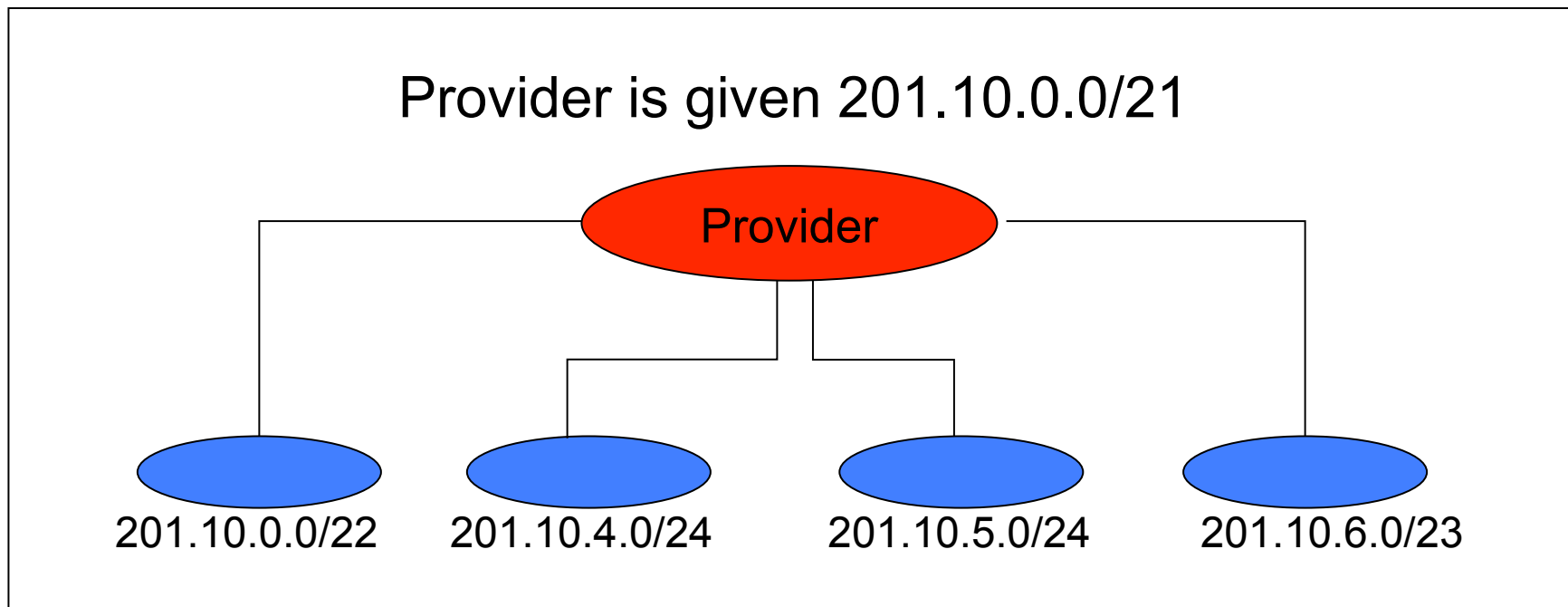
- **Prefixes are key to Internet scalability**
 - Address allocated in contiguous chunks (prefixes)
 - Routing protocols and packet forwarding based on prefixes
 - Today, routing tables contain ~200,000 prefixes (vs. 4B)



Two types of addresses

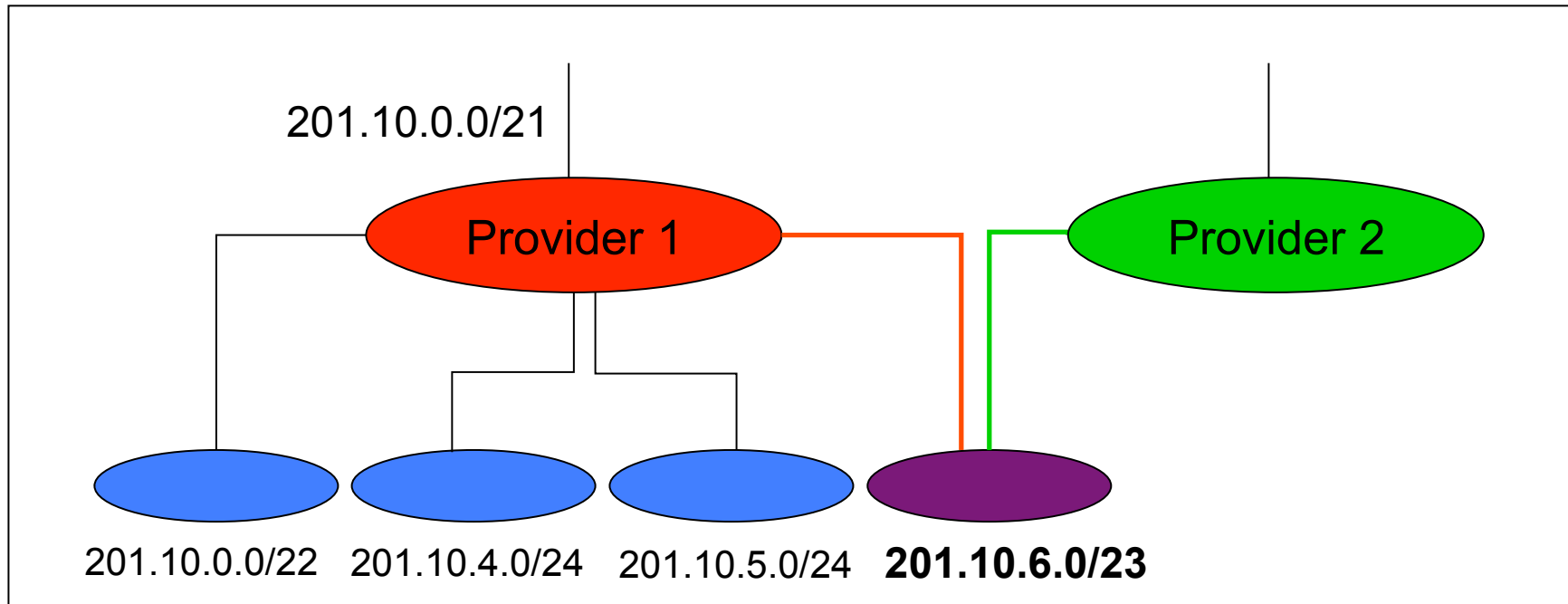
- Provider independent (from IANA)
- Provider allocated (from upstream ISP)
- Provider allocated addresses seem to offer more potential for aggregation (and reducing routing table size), but not always so...

Scalability: Address Aggregation



Routers in rest of Internet just need to know how to reach **201.10.0.0/21**. Provider can direct IP packets to appropriate **customer**.

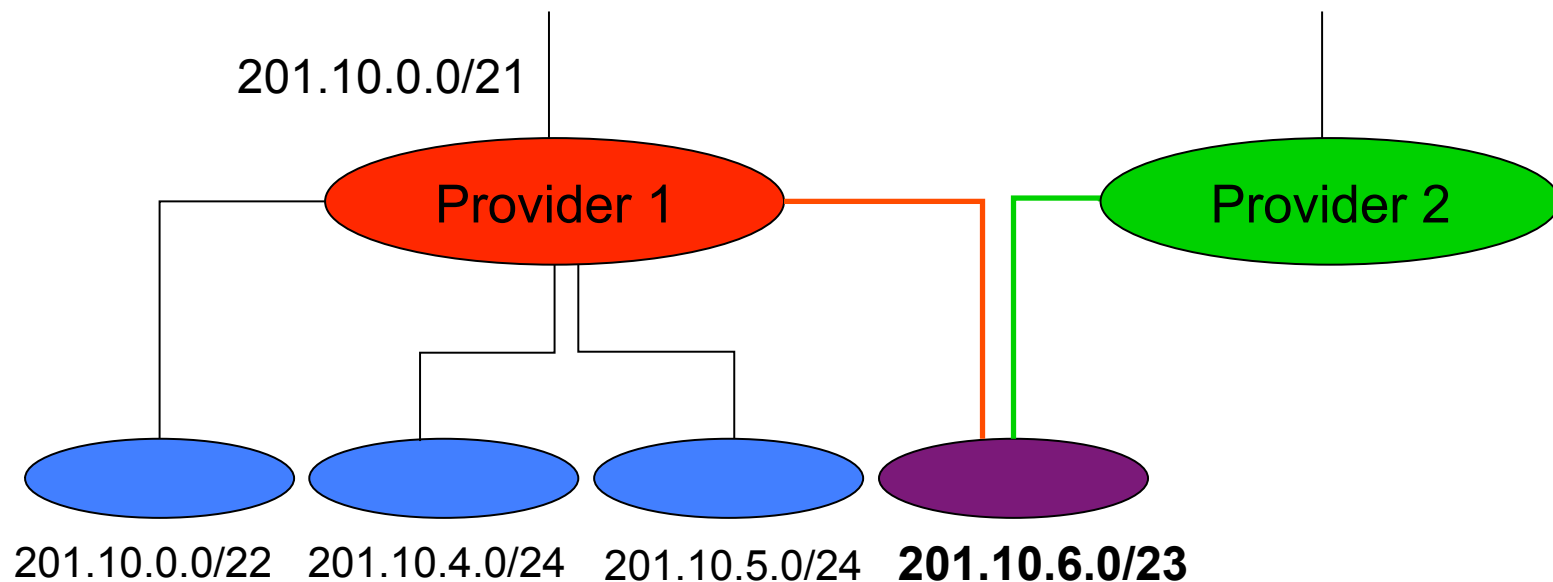
But, Aggregation Not Always Possible



Multi-homed customer (201.10.6.0/23) has two providers. Other parts of the Internet need to know how to reach these destinations through *both* providers.

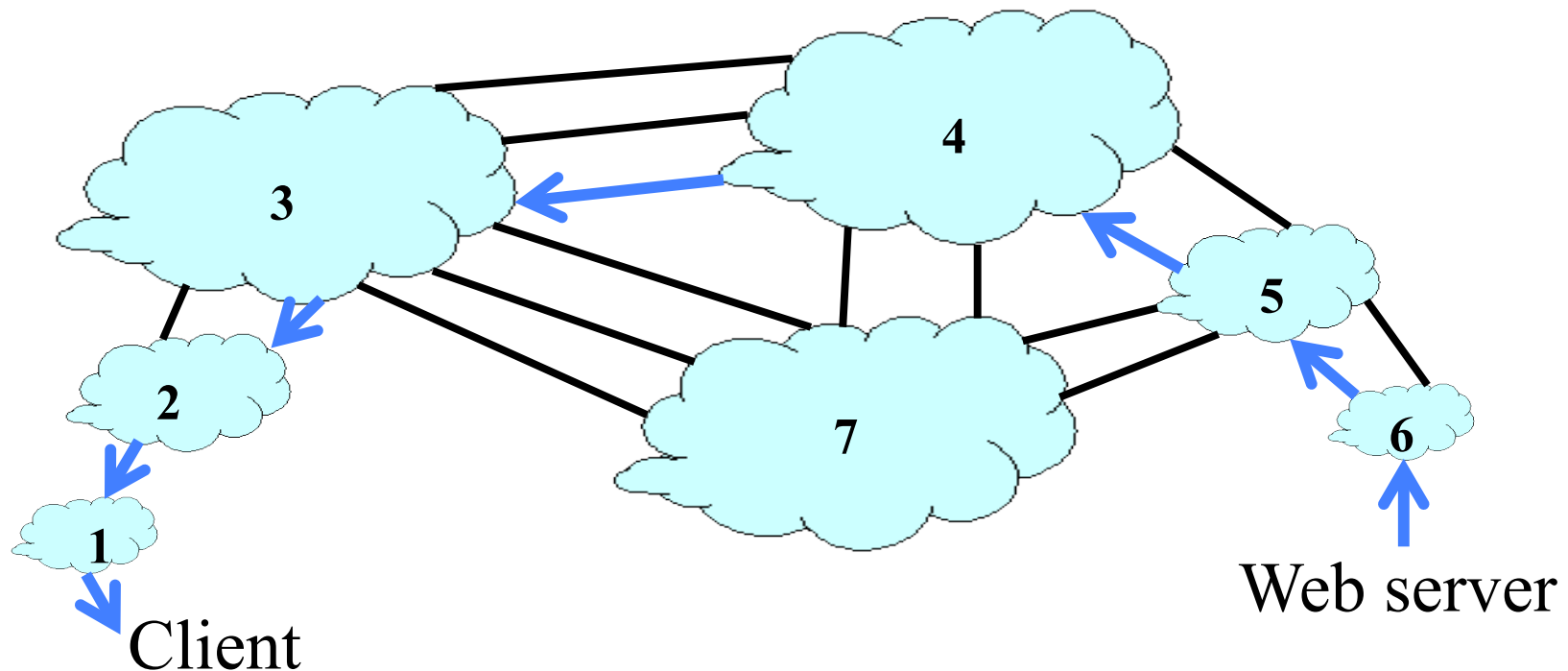
CIDR Makes Packet Forwarding Harder

- Forwarding table may have many matches
 - E.g., entries for 201.10.0.0/21 and 201.10.6.0/23
 - The IP address 201.10.6.17 would match *both*!
 - Use Longest Prefix Matching
- Can lead to routing table expansion
 - To satisfy LPM, need to announce /23 from both 1 and 2



Internet-wide Internet Routing

- **AS-level topology**
 - Destinations are IP prefixes (e.g., 12.0.0.0/8)
 - Nodes are Autonomous Systems (ASes)
 - Edges are links and business relationships



Middleboxes

- **Middleboxes are intermediaries**
 - Interposed in-between the communicating hosts
 - Often without knowledge of one or both parties

- **Myriad uses**
 - Network address translators
 - Firewalls
 - Tunnel endpoints
 - Traffic shapers
 - Intrusion detection systems
 - Transparent Web proxy caches
 - Application accelerators

“An abomination!”

- Violation of layering
- Hard to reason about
- Responsible for subtle bugs

“A practical necessity!”

- Solve real/pressing problems
- Needs not likely to go away

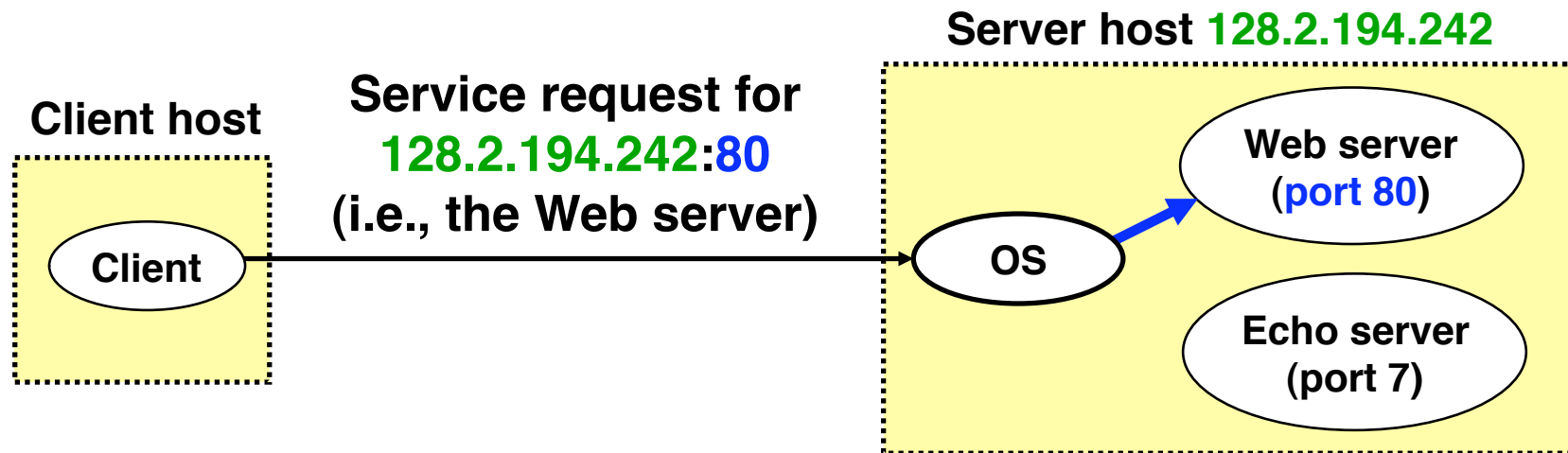
Port-Translating NAT

- **Map outgoing packets**
 - Replace source address with NAT address
 - Replace source port number with a new port number
 - Remote hosts respond using (NAT address, new port #)
- **Maintain a translation table**
 - Store map of (src addr, port #) to (NAT addr, new port #)
- **Map incoming packets**
 - Consult the translation table
 - Map the destination address and port number
 - Local host receives the incoming packet

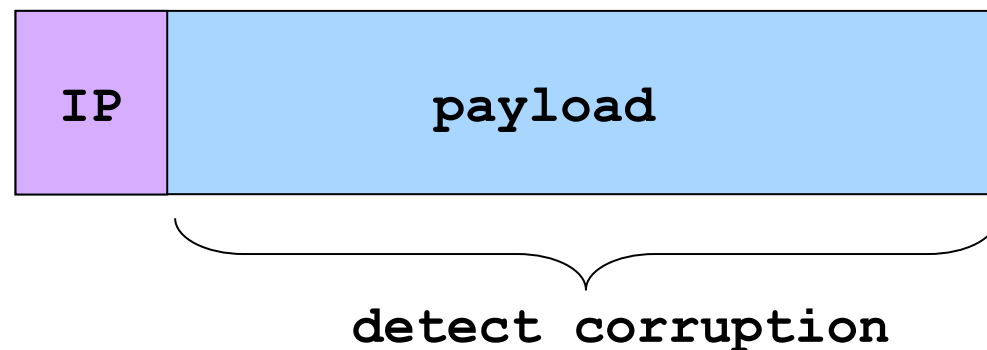
Transport Layer

Two Basic Transport Features

- **Demultiplexing: port numbers**

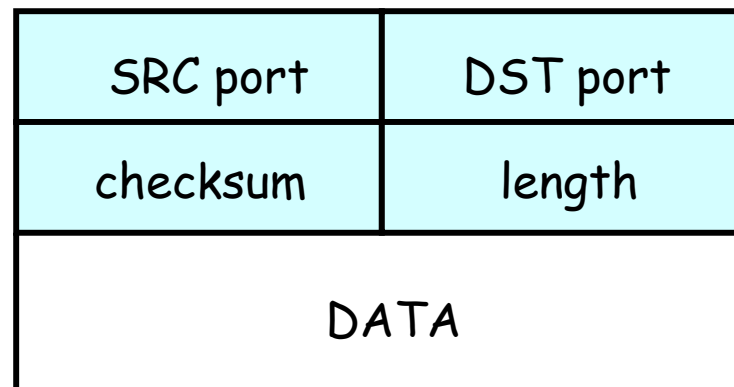


- **Error detection: checksums**



User Datagram Protocol (UDP)

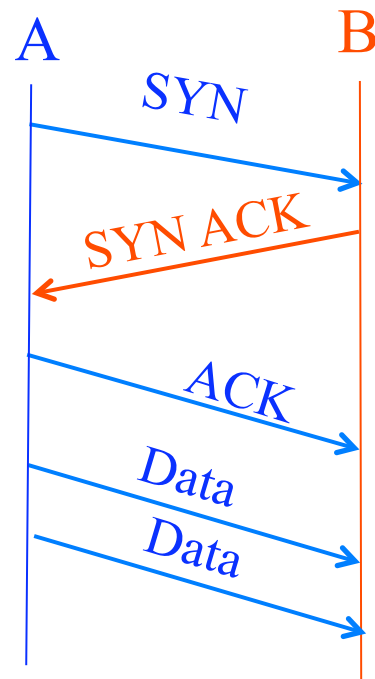
- **Datagram messaging service**
 - Demultiplexing of messages: port numbers
 - Detecting corrupted messages: checksum
- **Lightweight communication between processes**
 - Send messages to and receive them from a socket
 - Avoid overhead and delays of ordered, reliable delivery



Transmission Control Protocol (TCP)

- **Stream-of-bytes service**
 - Sends and receives a stream of bytes, not messages
- **Reliable, in-order delivery**
 - Checksums to detect corrupted data
 - Sequence numbers to detect losses and reorder data
 - Acknowledgments & retransmissions for reliable delivery
- **Connection oriented**
 - Explicit set-up and tear-down of TCP session
- **Flow control**
 - Prevent overflow of the receiver's buffer space
- **Congestion control**
 - Adapt to network congestion for the greater good

Establishing a TCP Connection

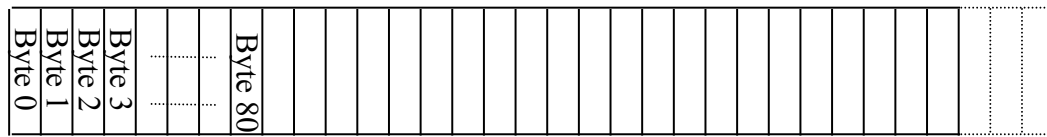


Each host tells its ISN to the other host.

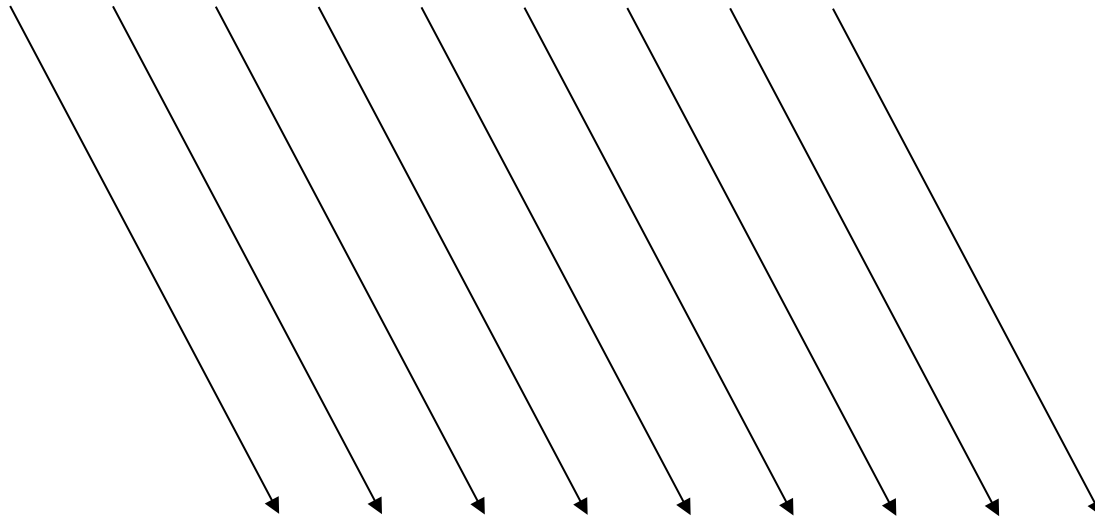
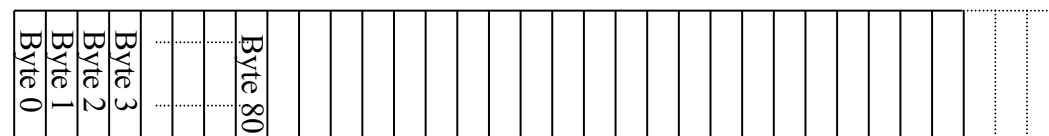
- Three-way handshake to establish connection
 - Host A sends a **SYN**chronize (open) to the host B
 - Host B returns a SYN **ACK**nowledgment (**SYN ACK**)
 - Host A sends an **ACK** to acknowledge the SYN ACK

TCP “Stream of Bytes” Service

Host A

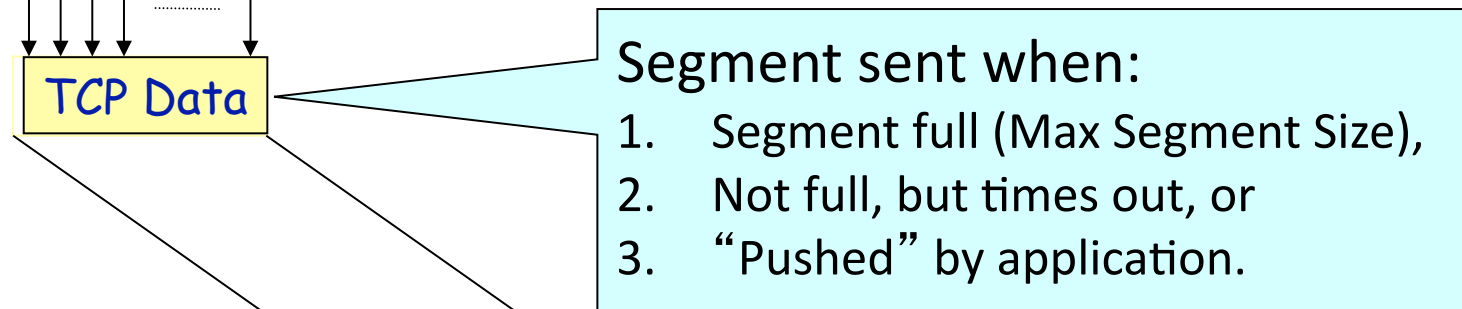
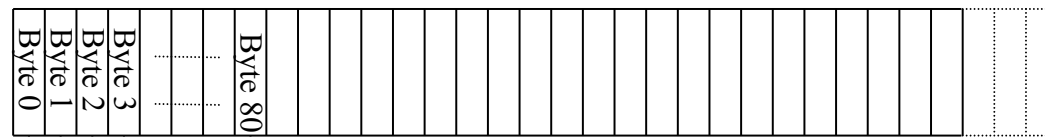


Host B

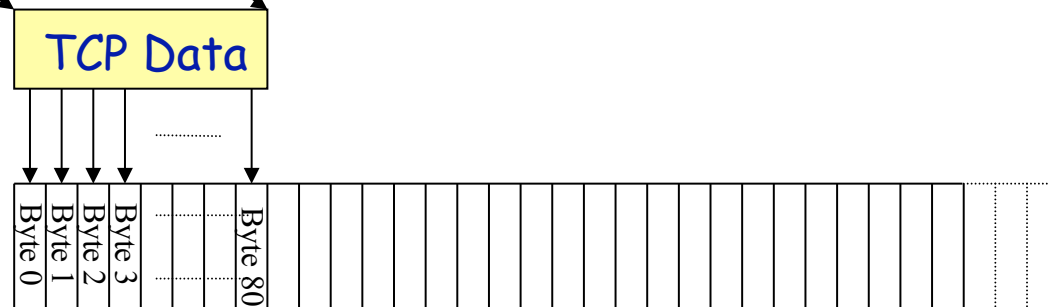


...Emulated Using TCP “Segments”

Host A

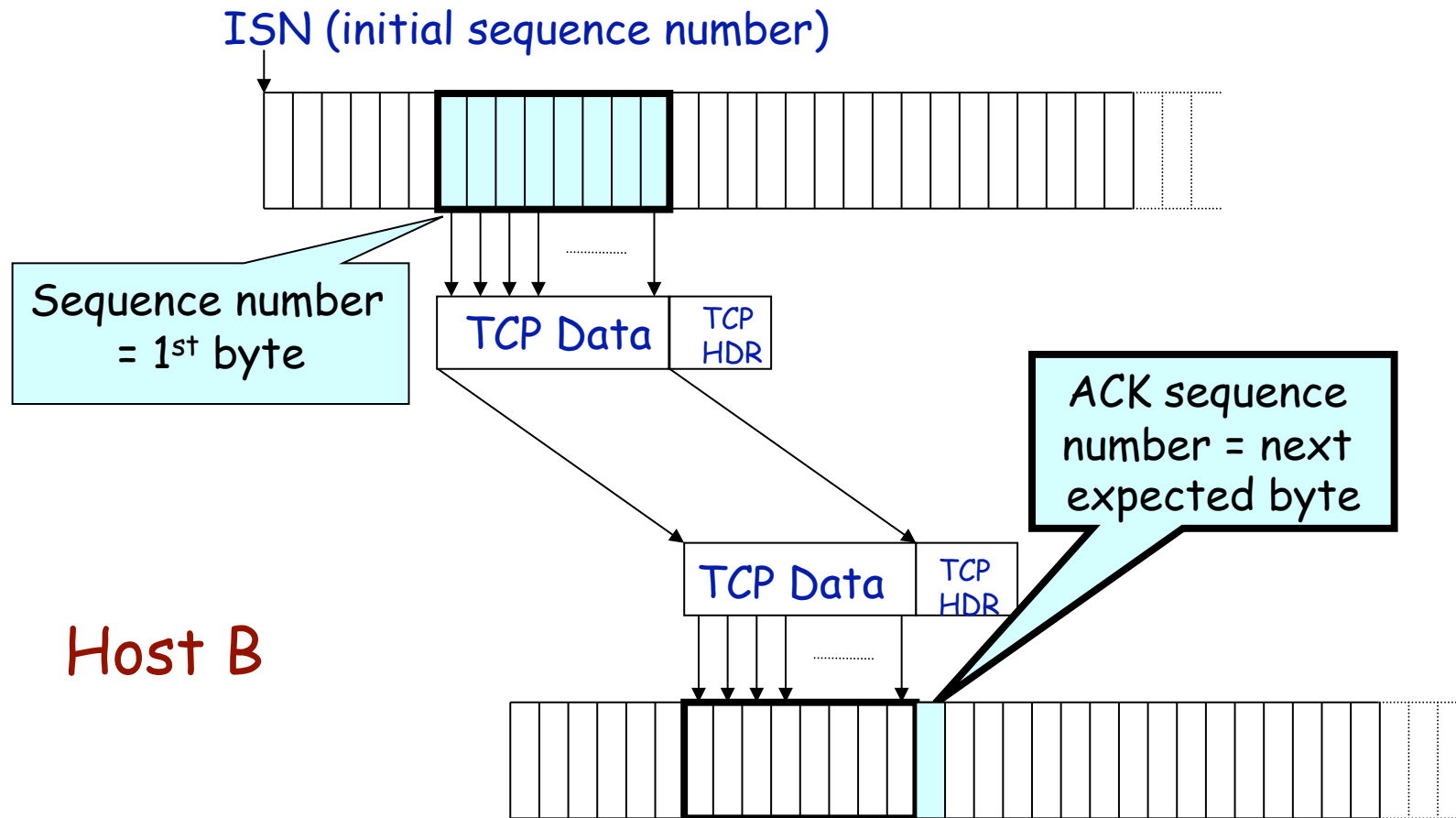


Host B

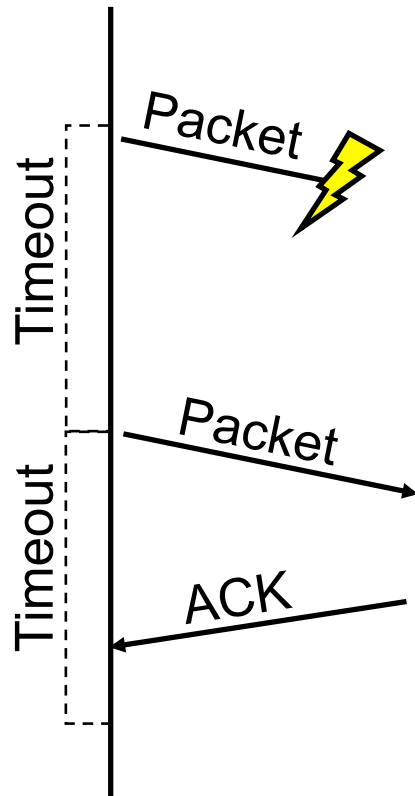


Reliability: TCP Acknowledgments

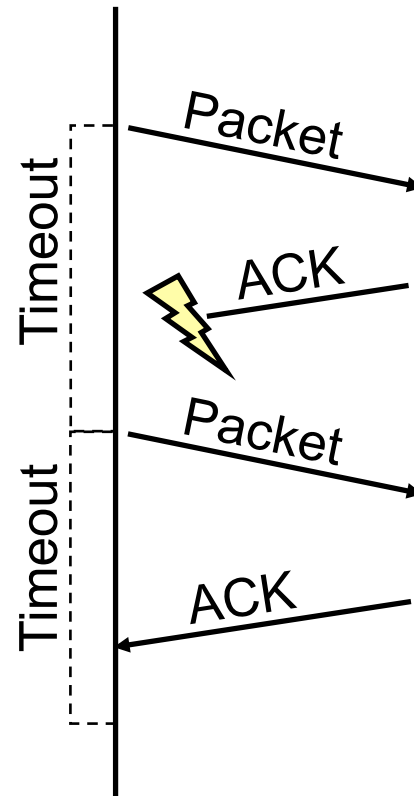
Host A



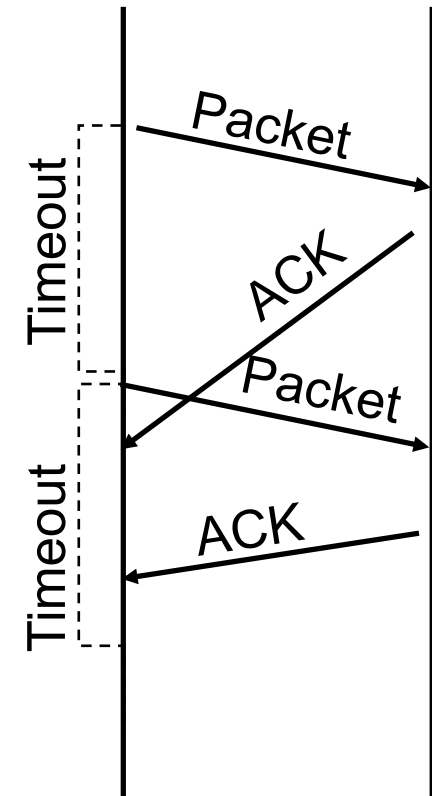
Detecting losses



Packet lost



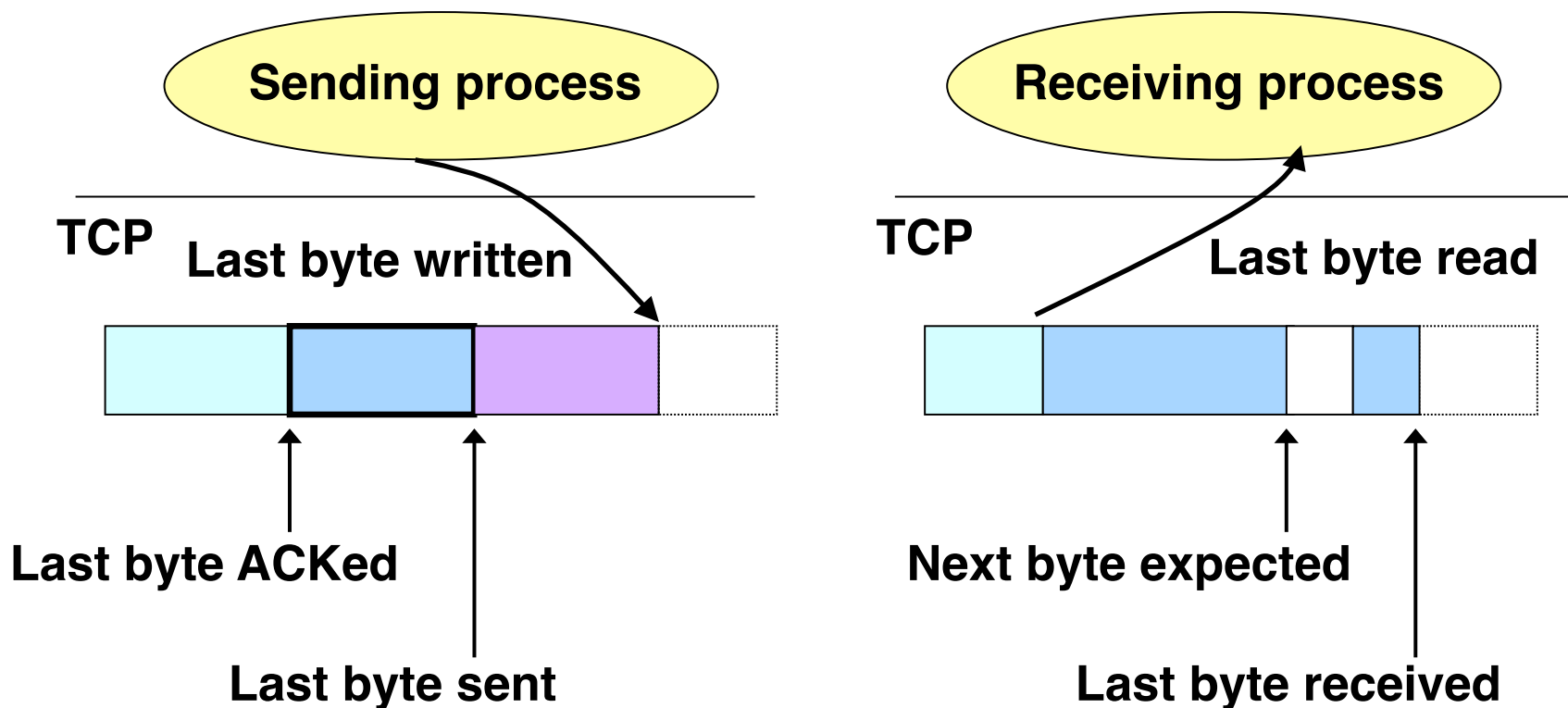
ACK lost
DUPLICATE
PACKET



Early timeout
DUPLICATE
PACKETS

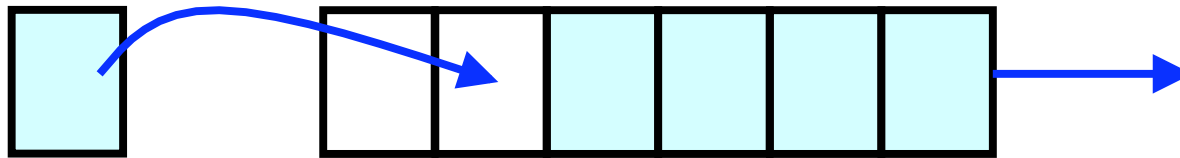
Flow control: Sliding window

- Allow a larger amount of data “in flight”
 - Allow sender to get ahead of the receiver
 - ... though not *too far* ahead

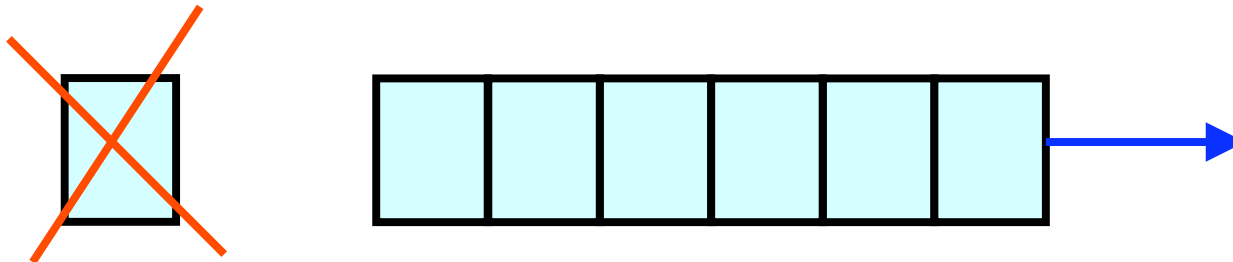


Where Congestion Happens: Links

- Simple resource allocation: FIFO queue & drop-tail
- Access to the bandwidth: first-in first-out queue
 - Packets transmitted in the order they arrive



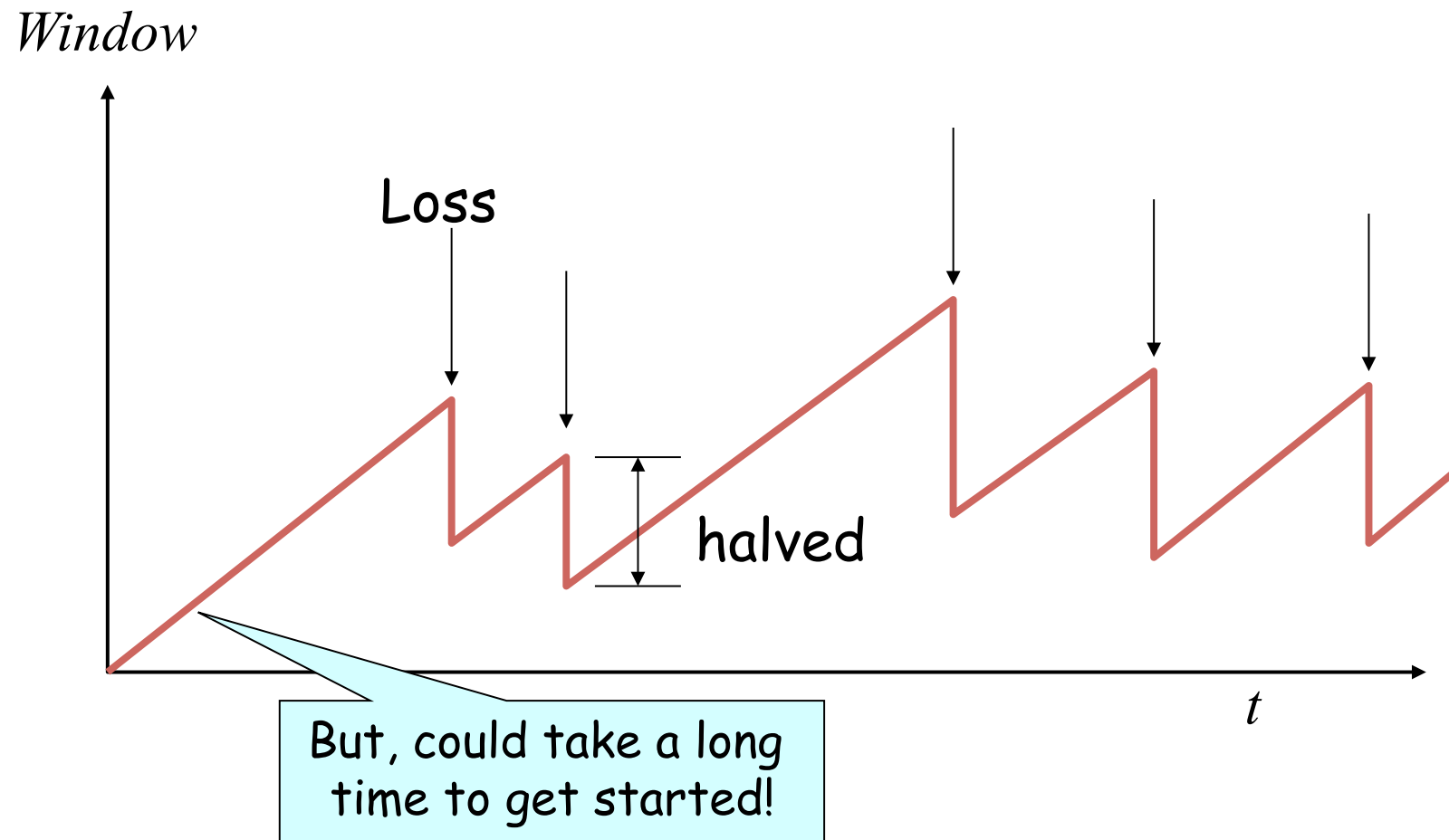
- Access to the buffer space: drop-tail queuing
 - If the queue is full, drop the incoming packet



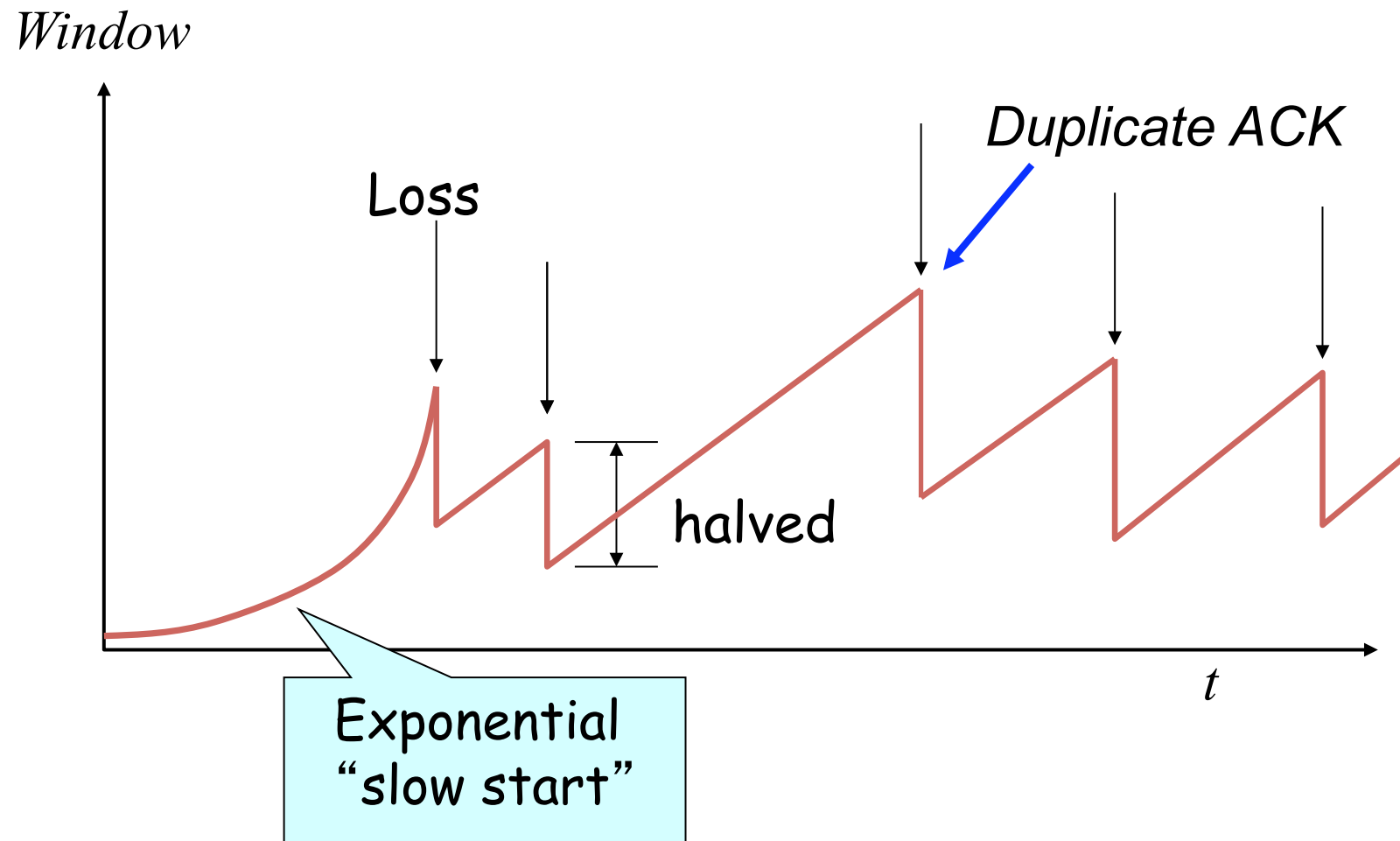
TCP Congestion Window

- Each TCP sender maintains a congestion window
 - Maximum number of bytes to have in transit
 - I.e., number of bytes still awaiting acknowledgments
- Adapting the congestion window
 - Decrease upon losing a packet: backing off
 - Increase upon success: optimistically exploring
 - Always struggling to find the right transfer rate
- Both good and bad
 - Pro: avoids having explicit feedback from network
 - Con: under-shooting and over-shooting the rate

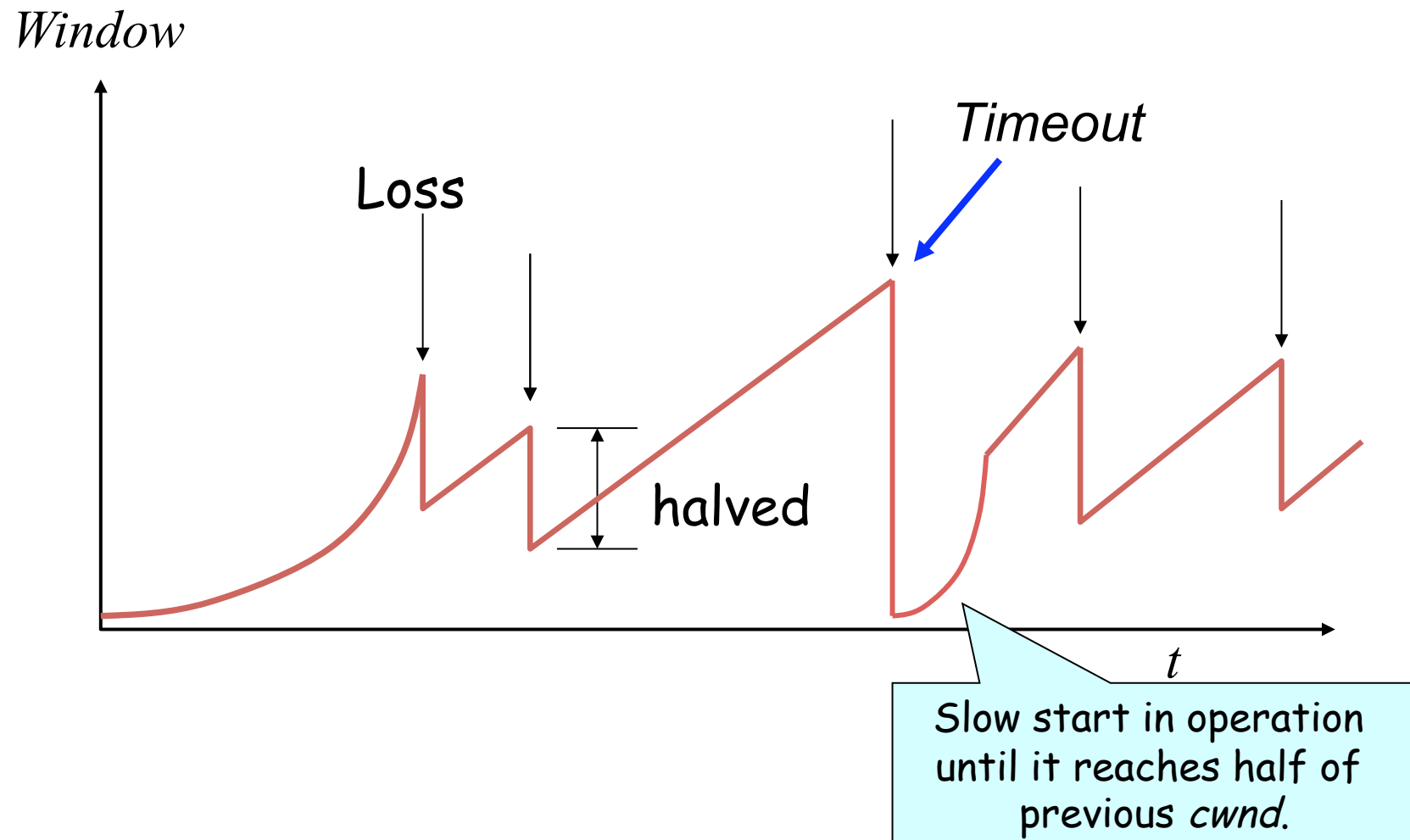
Leads to the TCP “Sawtooth”



Slow Start and the TCP Sawtooth



Repeating Slow Start After Timeout



Extensions

- Tail drop in routers lead to bursty loss and synchronization of senders
 - Led to Random Early Detection (RED)
- Packets dropped and retransmission when unnecessary
 - Led to Explicit Congestion Notification (ECN)

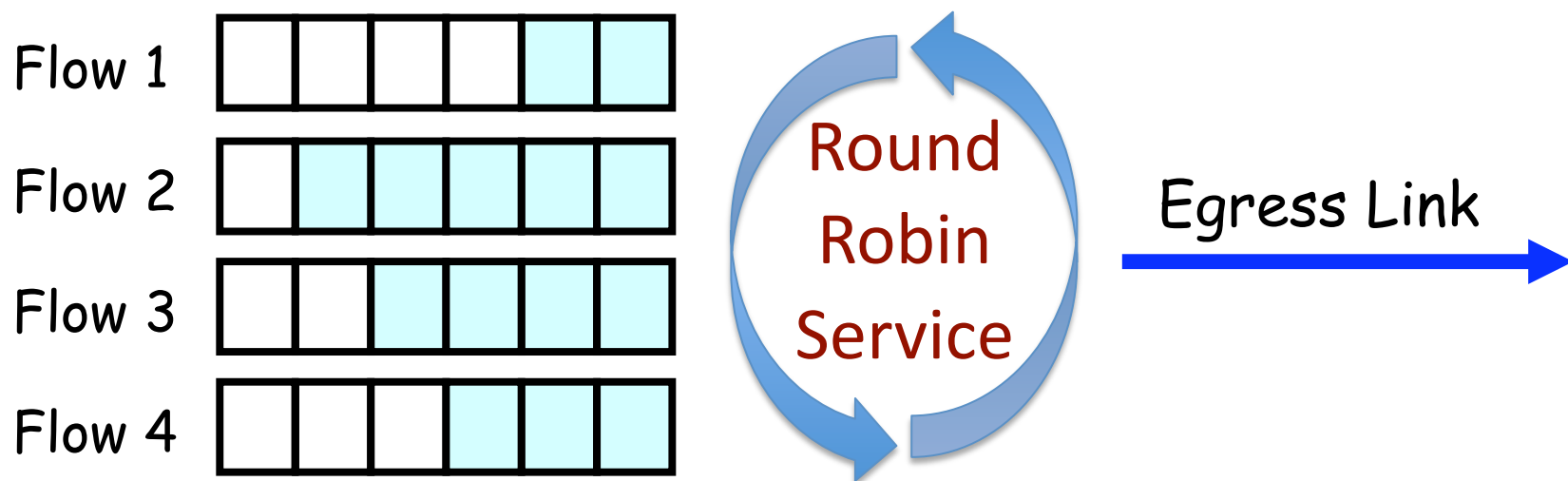
Problems with tail drop

- Under stable conditions, queue almost always full
 - Leads to high latency for all traffic
- Possibly unfair for flows with small windows
 - Larger flows may fast retransmit (detecting loss through Trip Dup ACKs), small flows may have to wait for timeout
- Window synchronization
 - More on this later...



Fair Queuing (FQ)

- Maintains separate queue per flow
- Ensures no flow consumes more than its $1/n$ share
 - Variation: weighted fair queuing (WFQ)
- If all packets were same length, would be easy
- If ***non-work-conserving*** (resources can go idle), also would be easy, yet lower utilization



Fair Queuing Basics

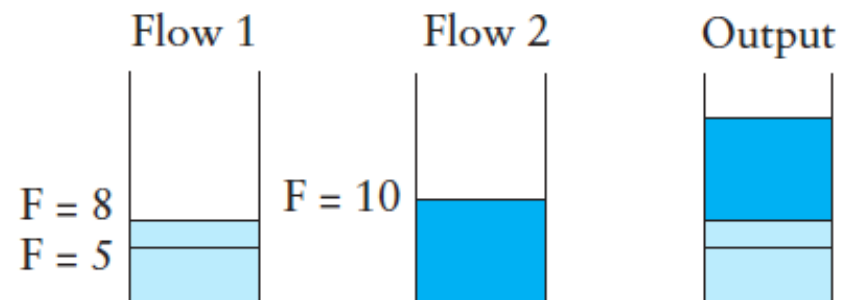
- Track how much time each flow has used link
 - Compute time used if it transmits next packet
- Send packet from flow that will have lowest use if it transmits
 - Why not flow with smallest use so far?
 - Because next packet may be huge!

FQ Algorithm

- Imagine clock tick per bit, then tx time \sim length

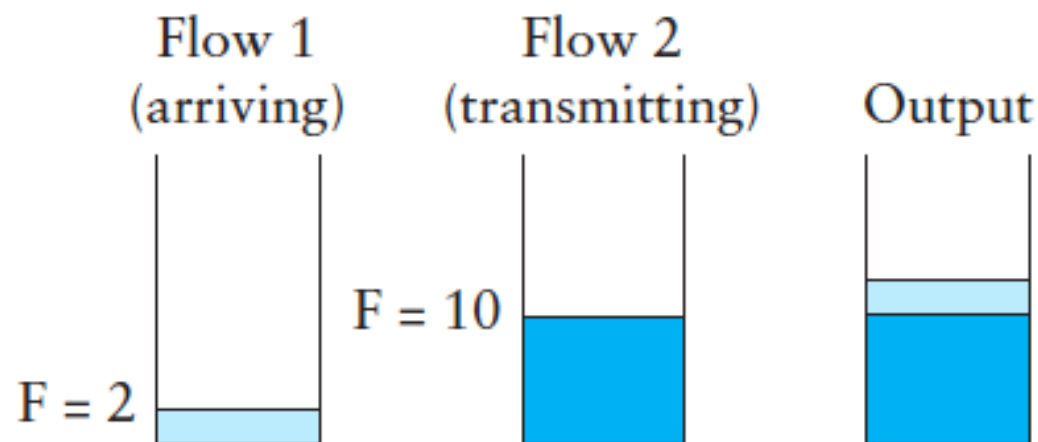
Finish time $F_i = \max(F_{i-1}, \text{Arrive time } A_i) + \text{Length } P_i$

- Calculate estimated F_i for all queued packets
- Transmit packet with lowest F_i next



FQ Algorithm (2)

- Problem: Can't preempt current tx packet
- Result: Inactive flows ($A_i > F_{i-1}$) are penalized
 - Standard algorithm considers no history
 - Each flow gets fair share only when packets queued



FQ Algorithm (3)

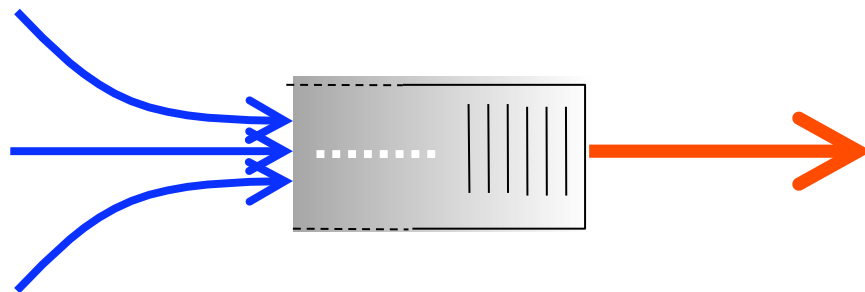
- Approach: give more *promptness* to flows utilizing less bandwidth historically
- Bid $B_i = \max (F_{i-1}, A_i - \delta) + P_i$
 - Intuition: with larger δ , scheduling decisions calculated by last tx time F_{i-1} more frequently, thus preferring slower flows
- FQ achieves max-min fairness
 - First priority: maximize the *minimum* rate of any active flows
 - Second priority: maximize the second min rate, etc.

Uses of (W)FQ

- **Scalability**
 - # queues must be equal to # flows
 - But, can be used on edge routers, low speed links, or shared end hosts
- **(W)FQ can be for classes of traffic, not just flows**
 - Use IP TOS bits to mark “importance”
 - Part of “Differentiated Services” architecture for “Quality-of-Service” (QoS)

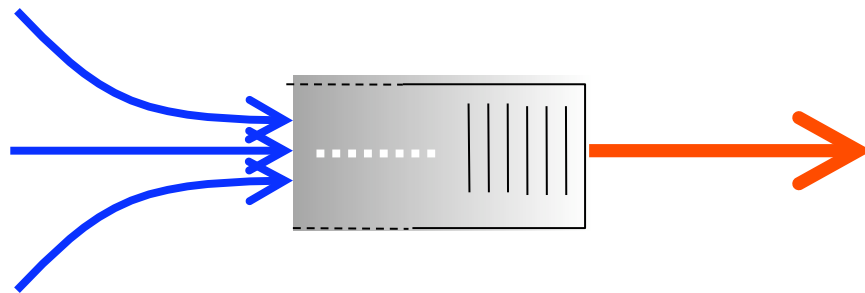
Bursty Loss From Drop-Tail Queuing

- TCP depends on packet loss
 - Packet loss is indication of congestion
 - And TCP *drives* network into loss by additive rate increase
- Drop-tail queuing leads to *bursty* loss
 - If link is congested, many packets encounter full queue
 - Thus, loss synchronization:
 - Many flows lose one or more packets
 - In response, many flows divide sending rate in half



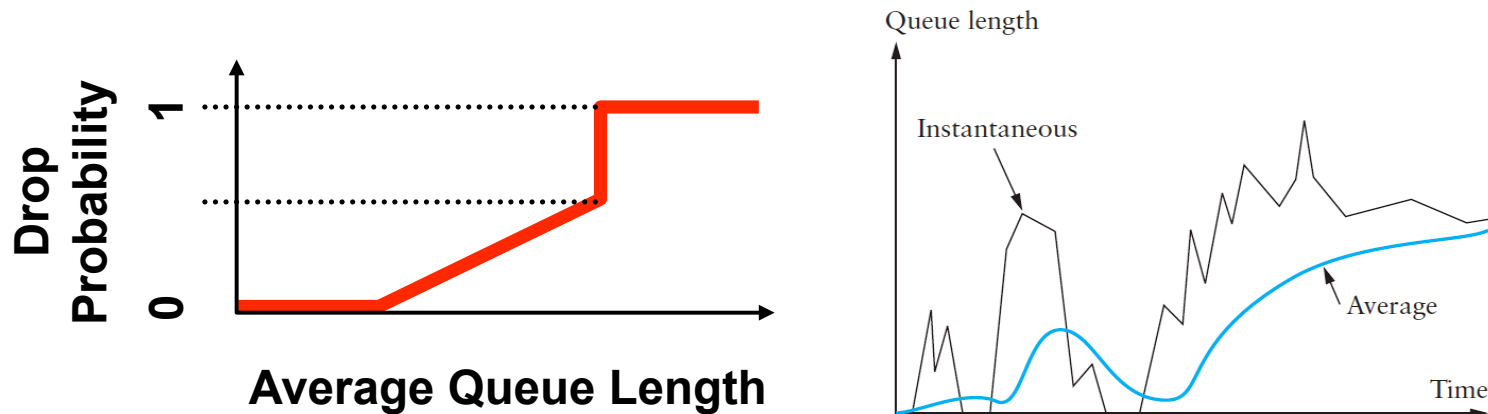
Slow Feedback from Drop Tail

- Feedback comes when buffer is completely full
 - ... even though the buffer has been filling for a while
- Plus, the filling buffer is increasing RTT
 - ... making detection even slower
- Might be better to give early feedback
 - And get 1-2 connections to slow down before it's too late



Random Early Detection (RED)

- **Basic idea of RED**
 - Router notices that queue is getting backlogged
 - ... and randomly drops packets to signal congestion
- **Packet drop probability**
 - Drop probability increases as queue length increases
 - Else, set drop probability as function of avg queue length and time since last drop



Properties of RED

- Drops packets before queue is full
 - In the hope of reducing the rates of some flows
- Drops packet in proportion to each flow's rate
 - High-rate flows have more packets
 - ... and, hence, a higher chance of being selected
- Drops are spaced out in time
 - Which should help desynchronize the TCP senders
- Tolerant of burstiness in the traffic
 - By basing the decisions on *average* queue length

Problems With RED

- **Hard to get tunable parameters just right**
 - How early to start dropping packets?
 - What slope for increase in drop probability?
 - What time scale for averaging queue length?
- **RED has mixed adoption in practice**
 - If parameters aren't set right, RED doesn't help
 - Hard to know how to set the parameters
- **Many other variations in research community**
 - Names like “Blue” (self-tuning), “FRED” ...

Feedback: From loss to notification

- **Early dropping of packets**
 - Good: gives early feedback
 - Bad: has to drop the packet to give the feedback
- **Explicit Congestion Notification**
 - Router marks the packet with an ECN bit
 - Sending host interprets as a sign of congestion

Explicit Congestion Notification

- **Must be supported by router, sender, AND receiver**
 - End-hosts determine if ECN-capable during TCP handshake
- **ECN involves all three parties (and 4 header bits)**
 1. Sender marks “ECN-capable” when sending
 2. If router sees “ECN-capable” and experiencing congestion, router marks packet as “ECN congestion experienced”
 3. If receiver sees “congestion experienced”, marks “ECN echo” flag in responses until congestion ACK’ d
 4. If sender sees “ECN echo”, reduces cwnd and marks “congestion window reduced” flag in next TCP packet
- **Why extra ECN flag?** Congestion could happen in either direction, want sender to react to forward direction
- **Why CRW ACK?** ECN-echo could be lost, but we ideally only respond to congestion in forward direction

Application layer

DNS

HTTP and CDNs

P2P and DHTs

Three Hierarchical Assignment Processes

- **Host name:** `www.cs.princeton.edu`
 - **Domain:** registrar for each top-level domain (e.g., .edu)
 - **Host name:** local administrator assigns to each host
- **IP addresses:** `128.112.7.156`
 - **Prefixes:** ICANN, regional Internet registries, and ISPs
 - **Hosts:** static configuration, or dynamic using DHCP
- **MAC addresses:** `00-15-C5-49-04-A9`
 - **Blocks:** assigned to vendors by the IEEE
 - **Adapters:** assigned by the vendor from its block

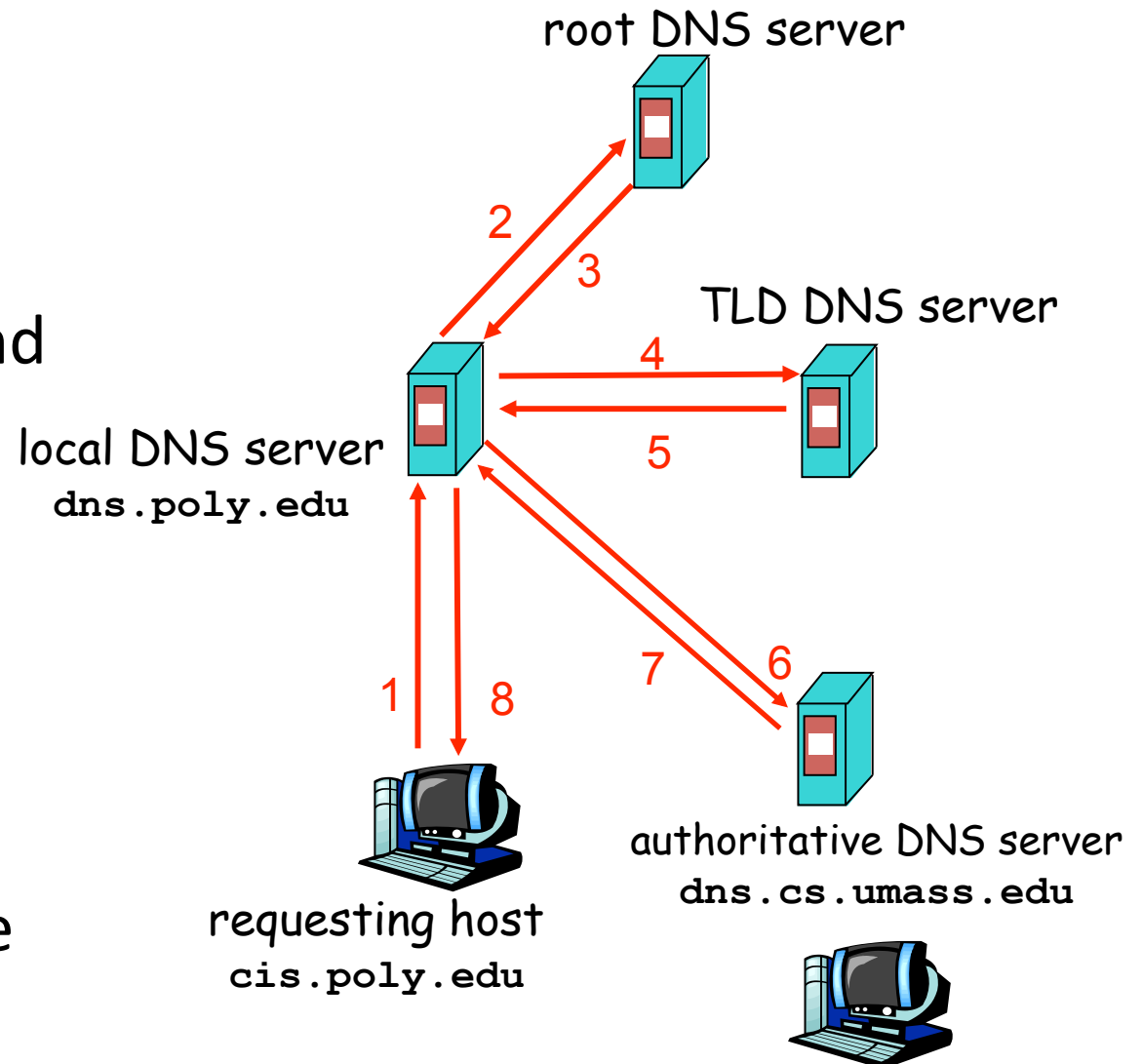
Mapping Between Identifiers

- **Domain Name System (DNS)**
 - Given a host name, provide the IP address
 - Given an IP address, provide the host name
- **Dynamic Host Configuration Protocol (DHCP)**
 - Given a MAC address, assign a unique IP address
 - ... and tell host other stuff about the Local Area Network
 - To automate the boot-strapping process
- **Address Resolution Protocol (ARP)**
 - Given an IP address, provide the MAC address
 - To enable communication within the Local Area Network

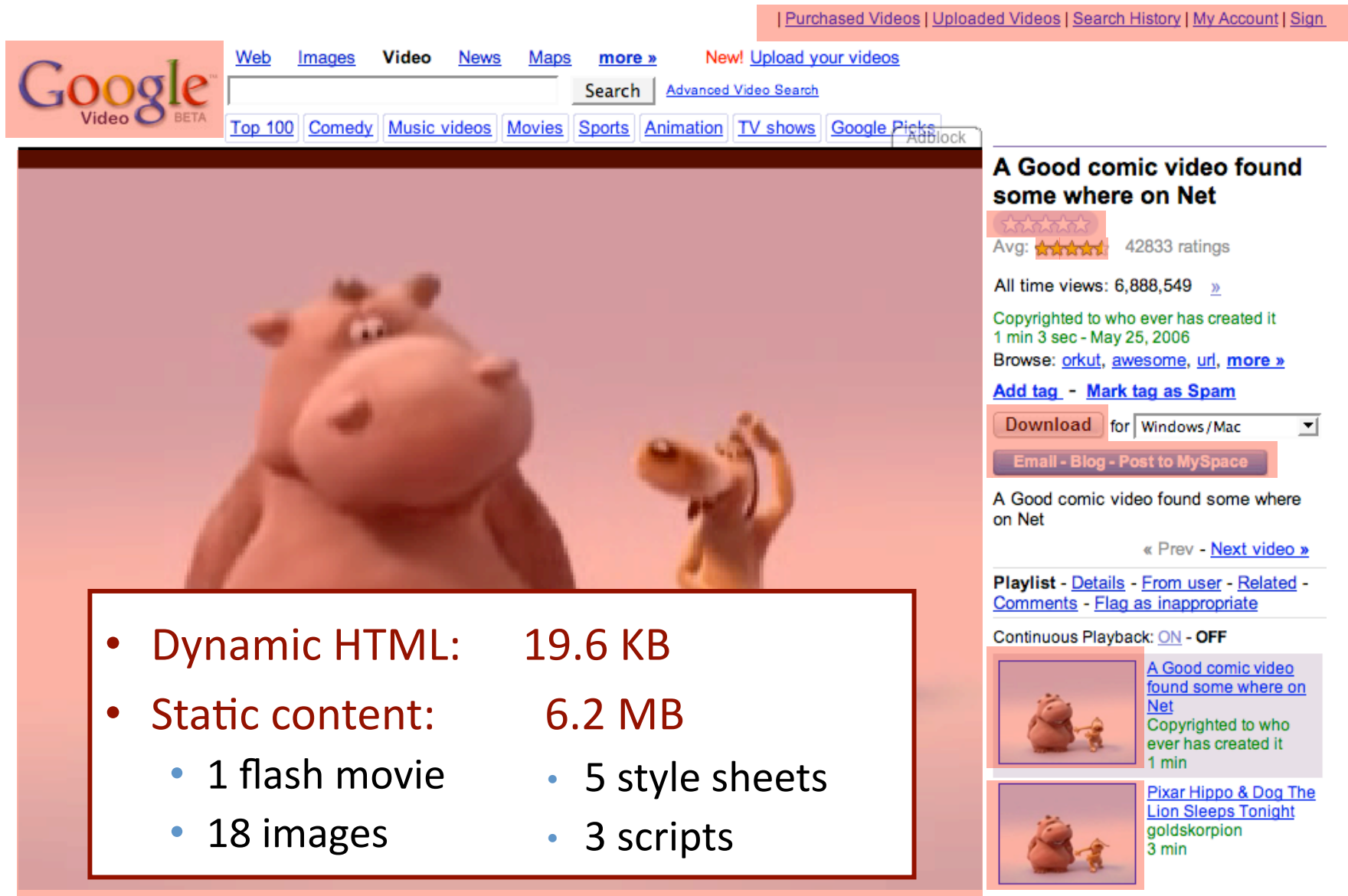
DHCP and ARP use L2 broadcast....DNS is app-layer protocol

Recursive vs. Iterative Queries

- **Recursive query**
 - Ask server to get answer for you
 - E.g., request 1 and response 8
- **Iterative query**
 - Ask server who to ask next
 - E.g., all other request-response pairs



One page, lots of objects



The screenshot shows a Google Video search result for a cartoon video. The page layout includes a top navigation bar with links for 'Purchased Videos', 'Uploaded Videos', 'Search History', 'My Account', and 'Sign'. Below this is the Google Video logo and navigation tabs for 'Web', 'Images', 'Video', 'News', 'Maps', and 'more'. A search bar is present with a 'Search' button and a link to 'Advanced Video Search'. Below the search bar are category tabs: 'Top 100', 'Comedy', 'Music videos', 'Movies', 'Sports', 'Animation', 'TV shows', and 'Google Picks'. The main content area features a large video player showing a cartoon hippo and a dog. To the right of the video player is a sidebar with the video title 'A Good comic video found some where on Net', a star rating of 4.5 out of 5, and 42833 ratings. It also shows 'All time views: 6,888,549' and 'Copyrighted to who ever has created it 1 min 3 sec - May 25, 2006'. Below this are links for 'Browse: orkut, awesome, url, more', 'Add tag - Mark tag as Spam', and a 'Download' button for Windows/Mac. There are also buttons for 'Email - Blog - Post to MySpace'. At the bottom of the sidebar are links for 'Playlist - Details - From user - Related - Comments - Flag as inappropriate' and 'Continuous Playback: ON - OFF'. Two smaller video thumbnails are shown at the bottom of the sidebar.

• Dynamic HTML: 19.6 KB

• Static content: 6.2 MB

- 1 flash movie
- 5 style sheets
- 18 images
- 3 scripts

TCP Interaction: Short Transfers

- **Multiple connection setups**
 - Three-way handshake each time
- **Round-trip time estimation**
 - Maybe large at the start of a connection (e.g., 3 seconds)
 - Leads to latency in detecting lost packets
- **Congestion window**
 - Small value at beginning of connection (e.g., 1 MSS)
 - May not reach a high value before transfer is done
- **Detecting packet loss**
 - Timeout: slow 😞
 - Duplicate ACK
 - Requires many packets in flight
 - Which doesn't happen for very short transfers 😞

Persistent HTTP

Non-persistent HTTP issues:

- Requires 2 RTTs per object
- OS must allocate resources for each TCP connection
- But browsers often open parallel TCP connections to fetch referenced objects

Persistent HTTP:

- Server leaves connection open after sending response
- Subsequent HTTP messages between same client/server are sent over connection

Persistent without pipelining:

- Client issues new request only when previous response has been received
- One RTT for each object

Persistent with pipelining:

- Default in HTTP/1.1
- Client sends requests as soon as it encounters referenced object
- As little as one RTT for all the referenced objects