Network Service Models

CS457

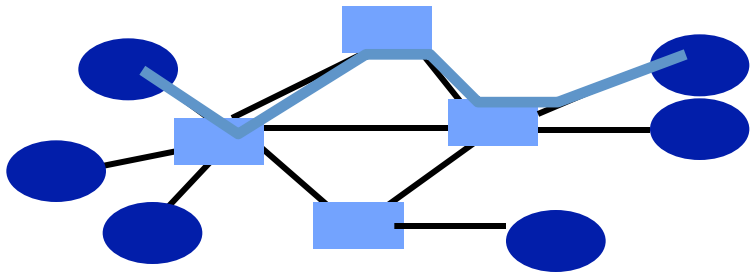
Fall 2014

Topics

- Circuit vs. Packet Switching
- Best Effort Internet Model
- Multiplexing/de-multiplexing
  - Socket and ports
- (chapters 1 and 2.5 onwards)
- Link Layer Protocols

Circuit Switching
(e.g., Phone Network)

- Step 1: Source establishes **connection** to destination
  - Connection setup signaling from src to dst
  - Routers en route store path connection info (state)
- Step 2: Source sends data over the connection
  - (Sometimes packets are called cells - cells are fixed size)
  - No dst address in packets, since routers know path
- Step 3: When done, source tears down connection



Virtual Circuits (VC)

- Need call setup/teardown for each call *before* data can flow
- Each packet carries VC identifier (not destination host address)
- *Every* router on src-dst path maintains "state" for each passing connection
- Link, router resources (bandwidth, buffers) may be allocated to VC
- Example network: ATM (Asynchronous Transfer Mode)

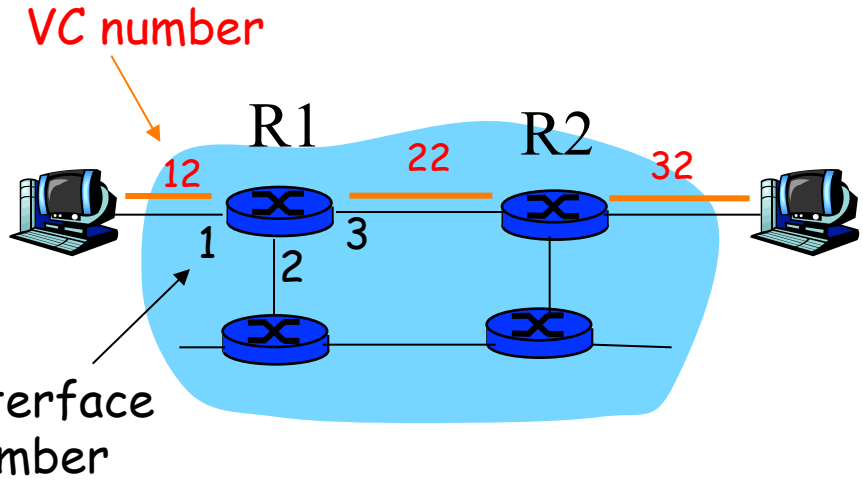
VC Implementation

A VC consists of:

1. Path from source to destination
2. VC numbers, one number for each link along path
3. Entries in forwarding tables in routers along path

- Packet belonging to VC carries a VC number in the header
- VC number may change (and usually does) on each link.
  - VCs are negotiated between neighboring routers

Forwarding Tables



Forwarding table in R1:

	Incoming interface	Incoming VC #	Outgoing interface	Outgoing VC #
C1:	1	12	3	22
	2	63	1	18
	3	7	2	17
	1	97	3	87
	…	…	…	…

Routers maintain connection state information!

What is the forwarding state in R2 for C1?

Advantages of Circuit Switching

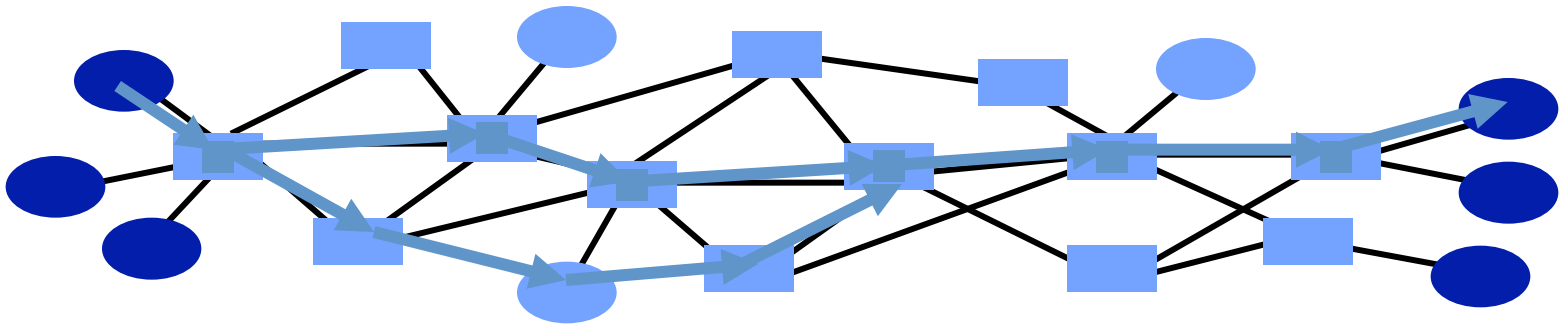
- Easy to provide performance guarantees
  - Bandwidth can be reserved along the entire path
  - Fixed path means virtually constant latency
- Simple abstraction
  - Reliable communication channel between hosts
  - No worries about lost or out-of-order packets
- Simple forwarding
  - Based on time slot, frequency or label
  - No need for complex packet header
  - Low per-packet overhead

Disadvantages of Circuit Switching

- With peak reservations, wasted BW
  - Idle resources during silent periods
  - Unable to achieve gains from statistical multiplexing
- Blocked connections
  - Connection refused when resources are not sufficient
  - Unable to offer "okay" service to everybody
- Connection set-up delay
  - No communication until a connection is set up (1 RTT)
  - Unable to avoid extra latency for small data transfers
- Network state
  - Network nodes must store per-connection state
  - Unable to avoid per-connection storage and state

Packet Switching (e.g., Internet)

- *Messages* divided into globally addressable *packets* (aka datagrams)
  - Each packet's header contains a destination address
- Packets may travel separately through network
  - Packet forwarding based on the header
  - Network nodes may store packets temporarily
- Destination reconstructs the message



IP Service Model: Why Packet Switching?

- In one word: Flexibility!
- Data traffic is bursty
  - Remote login, email, video, voice, etc.
- Packets don't waste reserved bandwidth
  - No traffic exchanged during idle periods
- Packets better for multiplexing
  - Different transfers share access to same links
- Packets can be delivered by almost anything
  - Best effort service
  - RFC 2549: IP over Avian Carriers (aka birds)
- … still, packet switching can be inefficient
  - Extra header bits

Network Architecture: Internet vs. POTS*

- There is a fundamental architectural difference between Internet and telephone network
- **POTS: Intelligent network, dumb terminals**
  - Reliable, in-sequence, guaranteed delivery (bandwidth and delay)
- **Internet: Dumb network, intelligent endpoints**
  - Best effort delivery (unreliable, packets may arrive out of sequence and duplicated, no bandwidth or delay guarantees)

(*POTS: Plain Old Telephone System)

IP Service Model: Why Best-Effort?

- Flexibility: Network does not dictate applications
- IP means never having to say you're sorry…
  - Don't need to reserve bandwidth and memory
  - Don't need to do error detection & correction
  - Don't need to remember from one packet to next
  - *Can't get any simpler than that!*
- Easier to survive failures
  - Transient disruptions are okay during failover
- … but, applications *do* want efficient, accurate transfer of data in order, in a timely fashion
  - IP pushes these to the higher layers

IP Service: Is Best-Effort Enough?

- No error detection or correction
  - Higher-level protocol can provide error checking
- Successive packets may not follow the same path
  - Usually not a problem as long as packets get there
- Packets can be delivered out-of-order
  - Number packets so they can be put back in order
- Packets may be lost or arbitrarily delayed
  - Sender can send the packets again (if desired)
- No network congestion signal (beyond "drop")
  - Sender can slow down in response to loss or delay (but this is a *really* hard problem..)

To Think About..

- Think about the diametrically opposing architectural difference:
  - Smart Network, dumb endpoints, vs. Stupid Network, Intelligent Endpoints.
- Which one provides more flexibility?
- Which one allows more future innovation?
- What do you think about the KISS principle? (Keep It Simple, Stupid)

Multiplexing/De-multiplexing

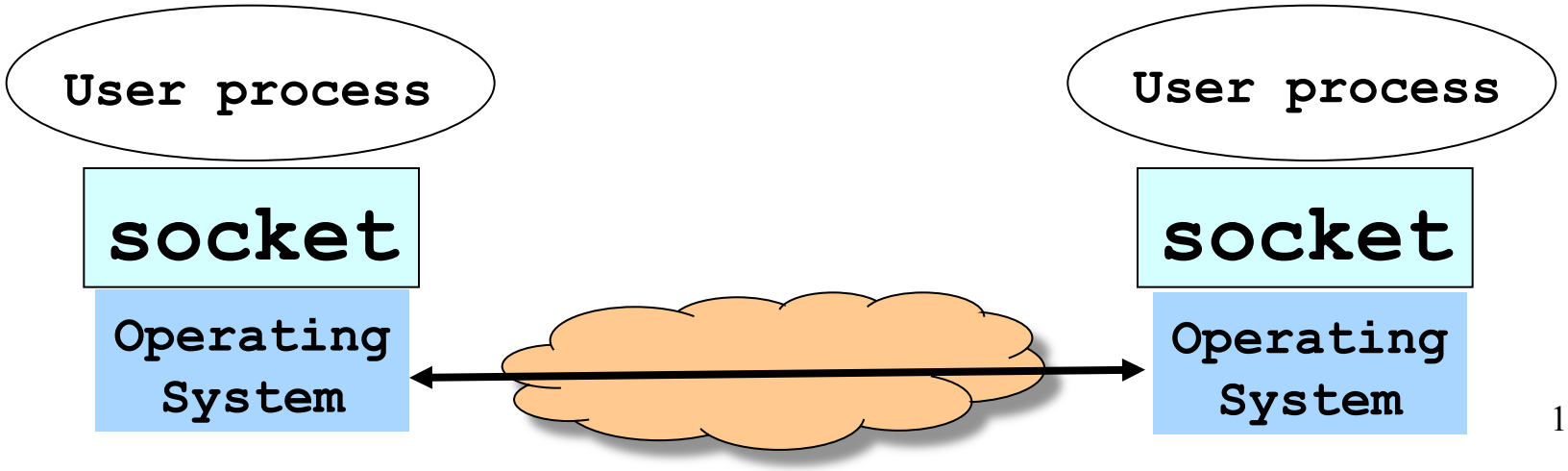
Sockets and Ports

Multiplexing/De-multiplexing

- Communication is really between end-**processes**, not just hosts
- How to identify the sending and receiving process?
- IP addresses are not enough – they only identify interfaces (not hosts! A host may have multiple interfaces, wired, wireless)
- Solution: **sockets and ports**

Socket: End Point of Communication

- Socket is an Application Programming Interface
  - Supports the creation of network applications
- Used for Inter-Process Communication (IPC)
  - Locally or over the network
- Process sends and receives through a "socket"
  - In essence, the doorway leading in/out of the house



UNIX Socket API

- Socket interface
  - Originally provided in Berkeley UNIX
  - Later adopted by all popular operating systems
  - Simplifies porting applications to different OSes
- In UNIX, everything is like a file
  - All input is like reading a file
  - All output is like writing a file
  - File is represented by an integer file descriptor
- New system calls for sockets
  - Client: create, connect, write, read, close
  - Server: create, bind, listen, accept, read, write, close
  - (covered at recitation)

Delivering the Data: Division of Labor

- Network
  - Deliver data packet to the destination **host**
  - Based on the destination IP address
- Operating system
  - Deliver data to the destination **socket**
  - Based on the protocol and destination port #
- Application
  - Read data from the socket
  - Interpret the data (e.g., render a Web page)

Ports: Identifying the Receiving Process

- Sending process must specify the receiver
  - Specify the host, and
  - Specify the receiving process
- Receiving host
  - Identified by its IP address (32 bits)
- But the receiving process?
  - **Problem**: host may be running many different processes
  - **Solution**: port: uniquely identifies the receiver's socket
  - A port number is a 16-bit quantity
- Similarly, the sender specifies its own socket using its own port number
- Thus, for networked applications a socket must be *bound* to a port (See the bind() syscall)

Multiplexing/De-multiplexing

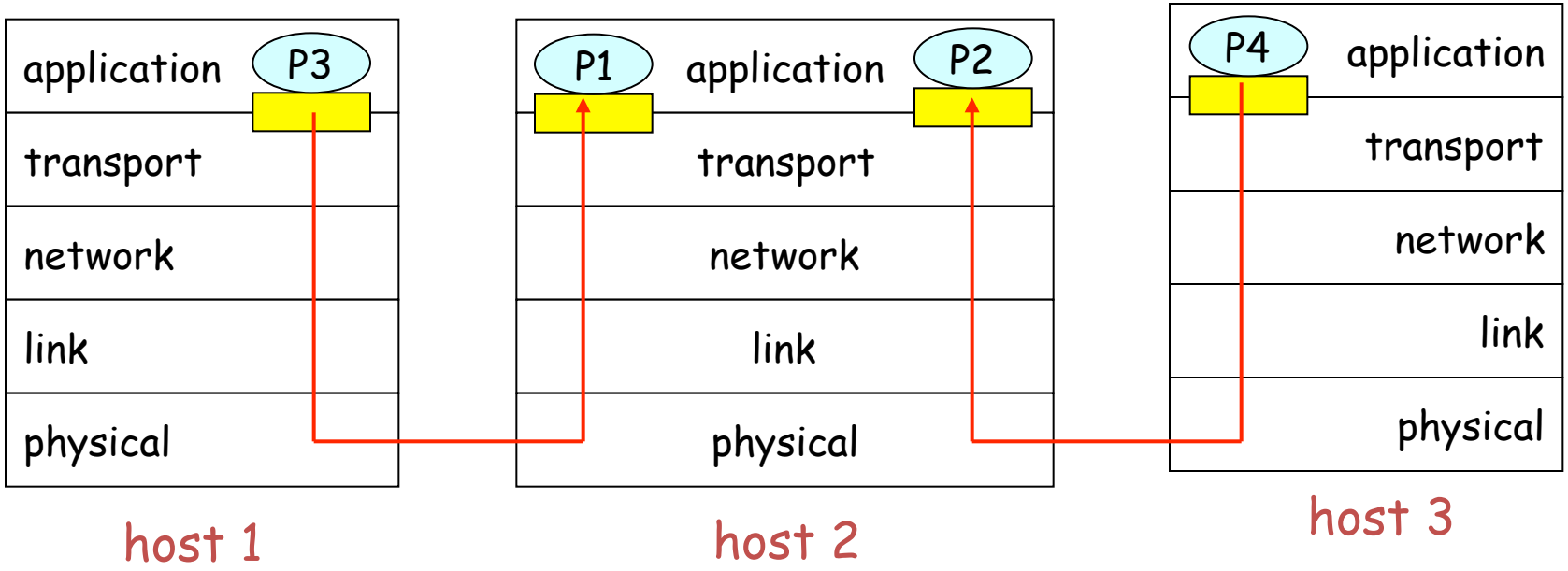
De-multiplexing at recv host

deliver received segments to correct socket

Multiplexing at send host

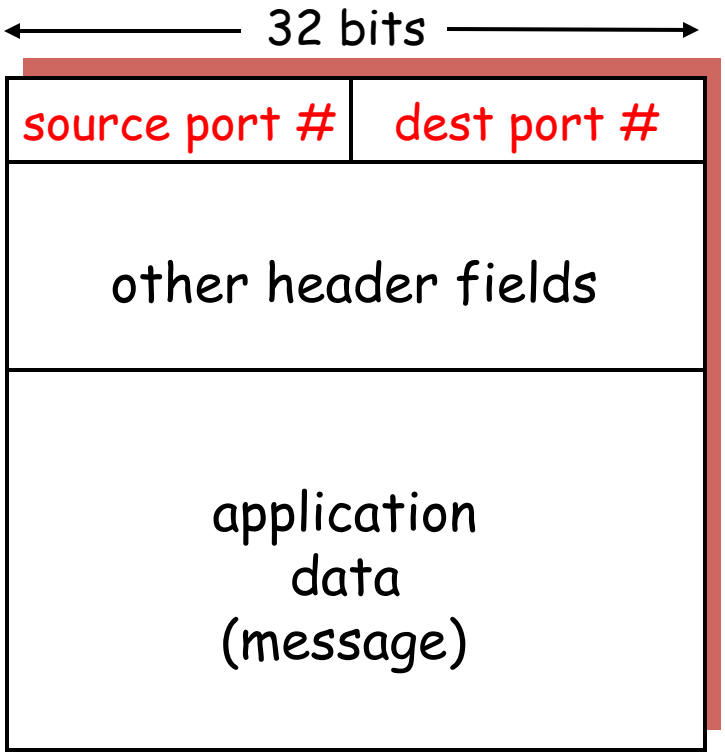
gathering data from multiple sockets, enveloping data with header (later used for De-multiplexing)

= socket = process



How De-multiplexing Works

- Network layer delivers IP datagrams
  - each datagram (packet) has source IP address, destination IP address
  - each datagram carries one transport-layer segment
  - each segment has source, destination port number (recall: well-known port numbers for specific applications)
- host uses IP addresses & port numbers to direct segment to appropriate socket



Simplified TCP/UDP segment format

Part 1: Connectionless De-multiplexing

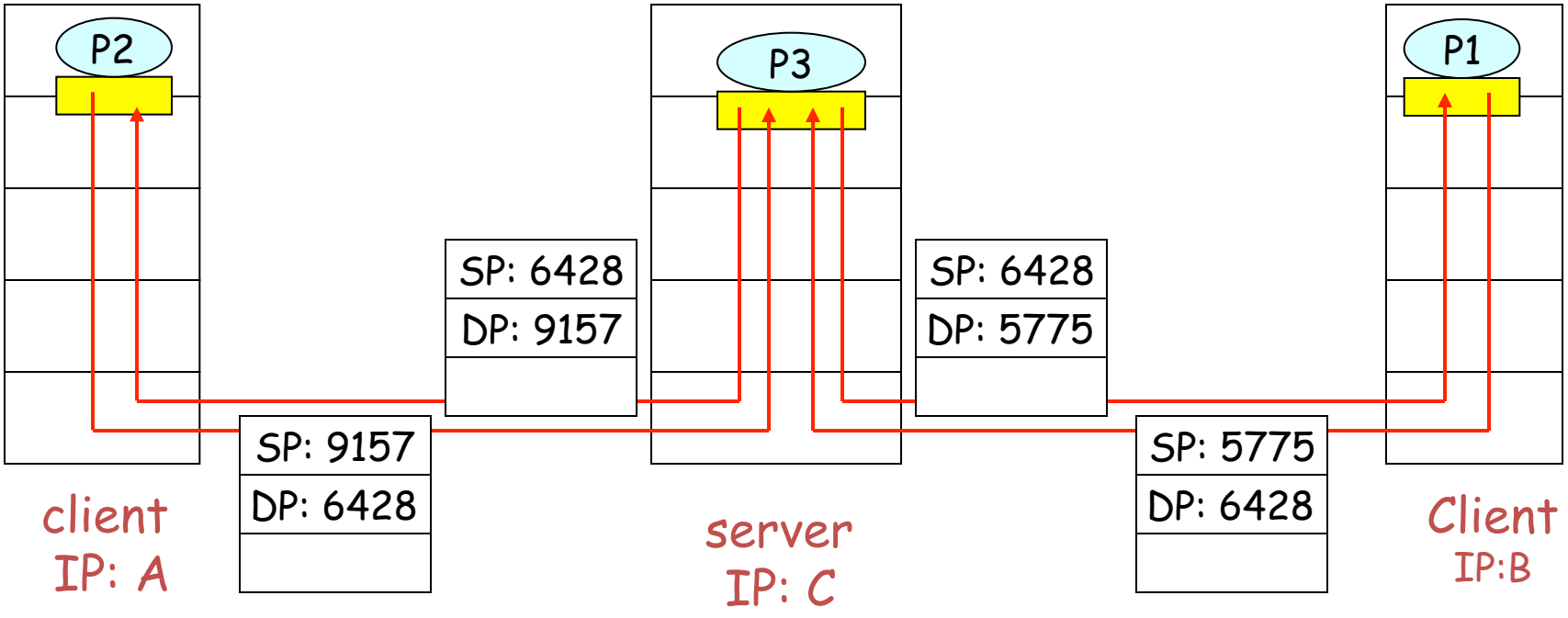
- Connectionless socket identified by two-tuple:

  <dest IP address, dest port>

- For communication two such sockets are needed – one at each end
- In the Internet, implemented by the UDP transport protocol

- When host receives segment:
  - checks destination port number in segment
  - directs segment to socket with that port number
- IP datagrams with different source IP addresses and/or source port numbers directed to same socket

Connectionless Demux (cont.)



SP provides "return address"

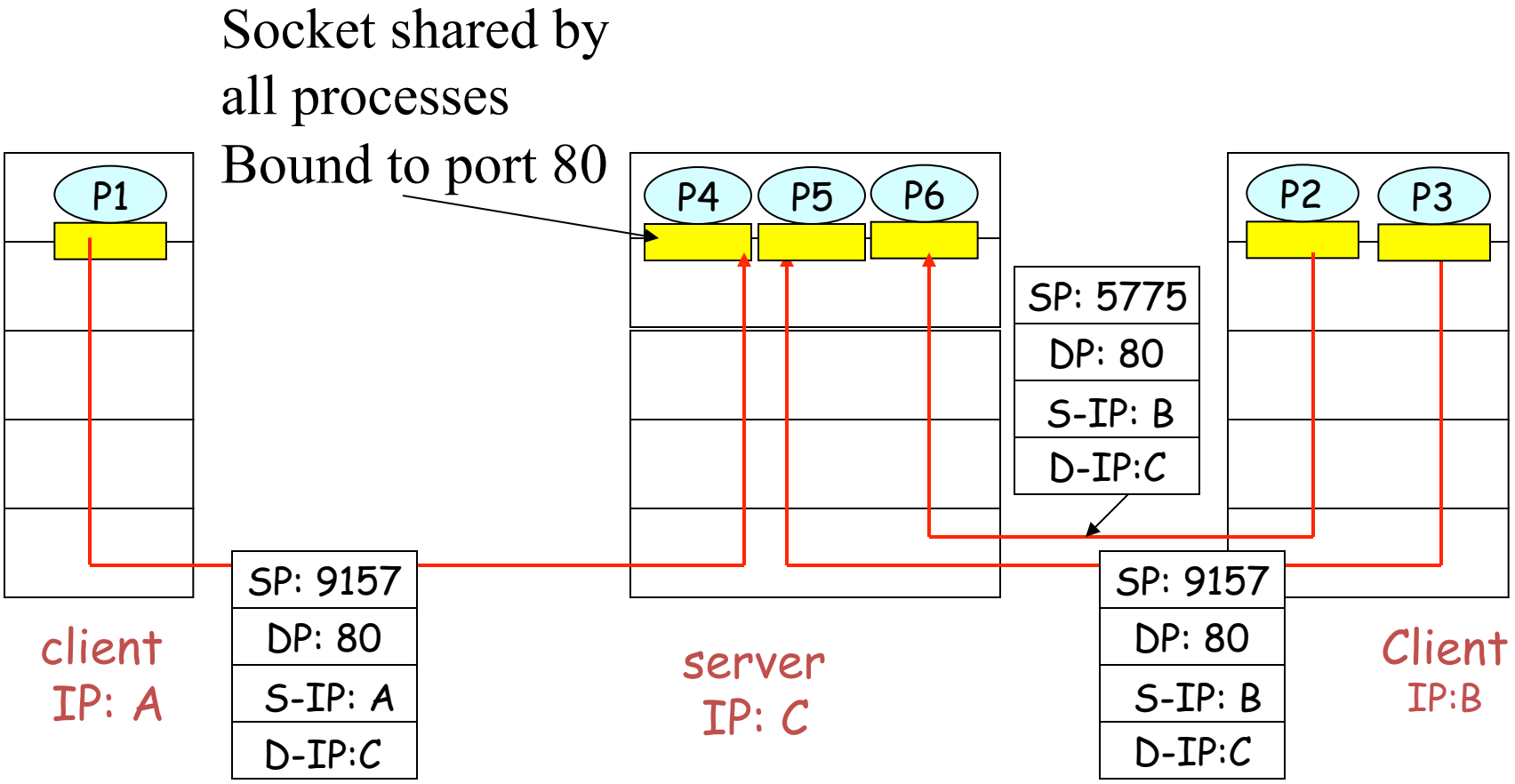
Part 2: Connection-oriented Demux

- Connection identified by 4-tuple:

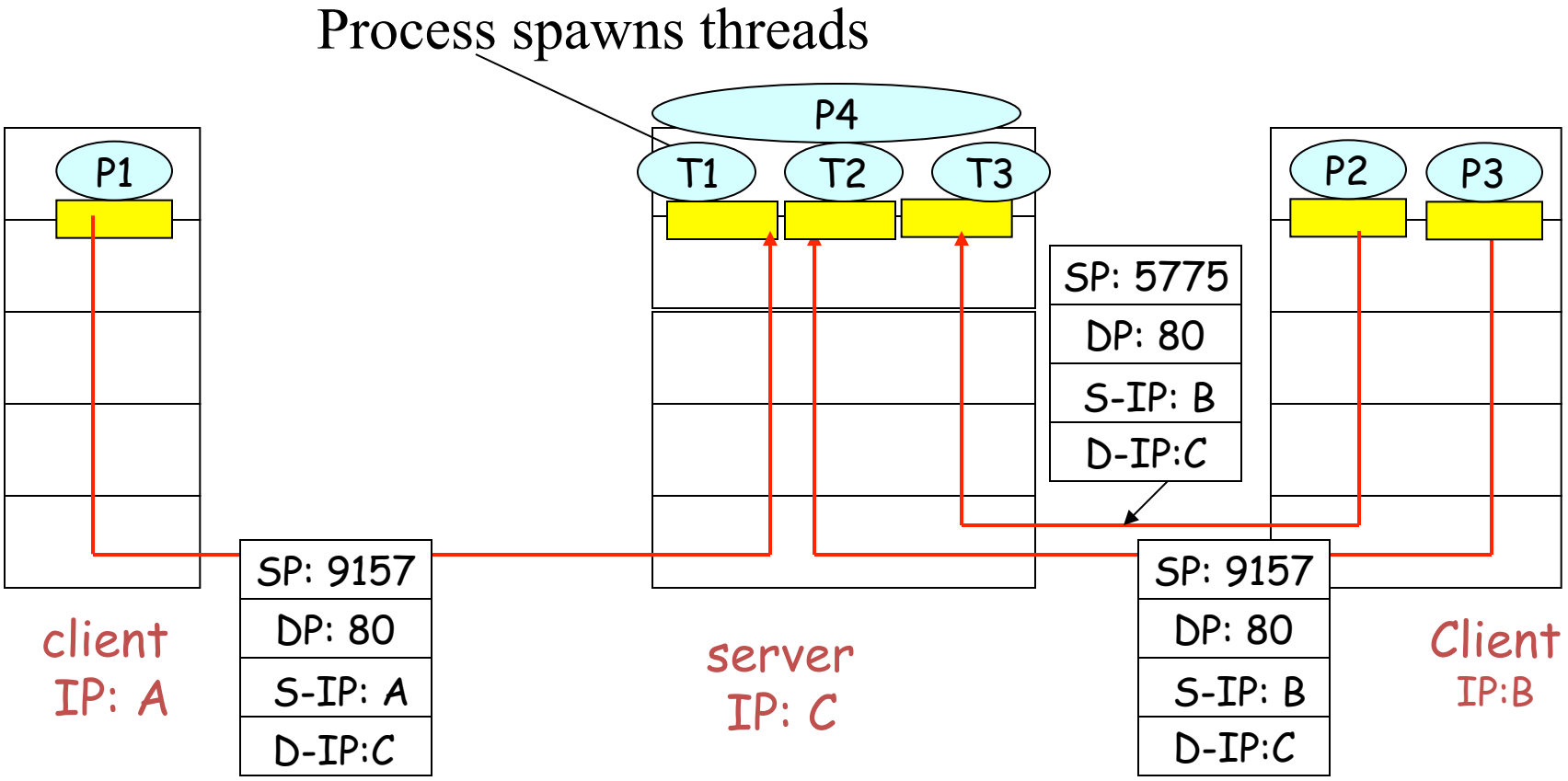
<srsIP, src port, dstIP, dst port>

- Receiving host uses all four values to direct segment to appropriate socket
- In the Internet, implemented by the TCP transport protocol
- Server host may support many simultaneous sockets:
  - each socket identified by its own 4-tuple
- Web servers have different sockets for each connecting client
  - non-persistent HTTP will have different socket for each request

Connection-oriented Demux (cont.)



Example: Threaded Web Server



What Port Number To Use?

- Popular applications have well-known ports
  - E.g., port 80 for Web and port 25 for e-mail
  - Well-known ports listed at http://www.iana.org
- Well-known vs. ephemeral ports
  - Server has a well-known port (e.g., port 80)
    - Between 0 and 1023 (low or privileged ports, OS assigned)
  - Client picks an unused ephemeral (i.e., temporary) port
    - Between 1024 and 65535
- Uniquely identifying the traffic between the hosts
  - Two IP addresses and two port numbers
  - Underlying transport protocol (e.g., TCP or UDP)
  - The so-called **5-tuple <saddr, sp, daddr, dp, proto>**

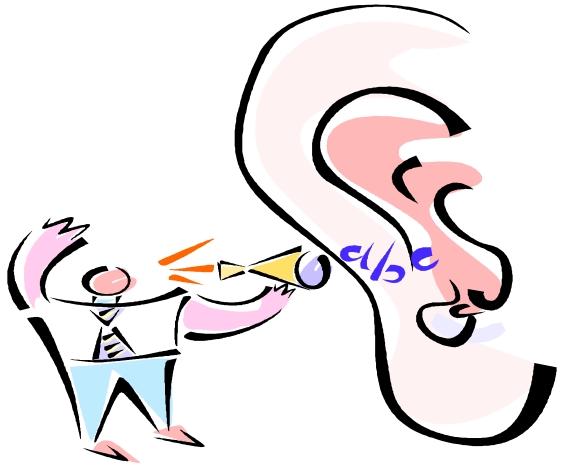
Typical Client Program

- Prepare to communicate
  - Create a socket
  - Determine server address and port number
  - Initiate the connection to the server
- Exchange data with the server
  - Write data to the socket
  - Read data from the socket
  - Do stuff with the data (e.g., render a Web page)
- Close the socket

Typical Server Program

- Prepare to communicate
  - Create a socket
  - Associate local address and port with the socket
- Wait to hear from a client (passive open)
  - Indicate how many clients-in-waiting to permit
  - Accept an incoming connection from a client
- Exchange data with the client over new socket
  - Receive data from the socket
  - Do stuff to handle the request (e.g., get a file)
  - Send data to the socket
  - Close the socket
- Repeat with the next connection request

Servers Differ From Clients



- Passive open
  - Prepare to accept connections
  - … but don't actually establish one
  - … until hearing from a client
- Hearing from multiple clients
  - Allow a backlog of waiting clients
  - ... in case several try to start a connection at once
- Create a socket for each client
  - Upon accepting a new client
  - … create a *new* socket for the communication

Want to See Real Clients and Servers?

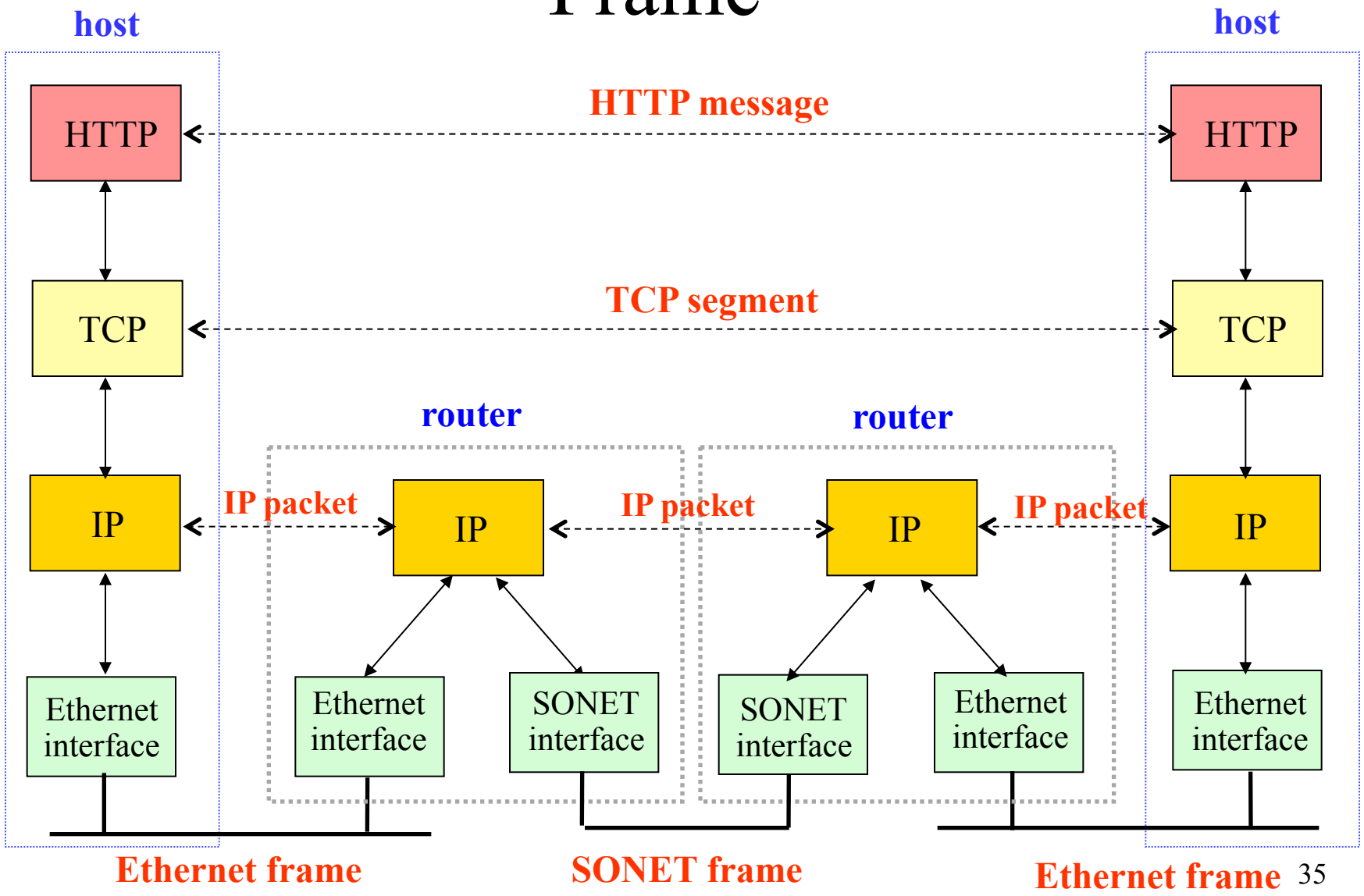
- Apache Web server
  - Open source server first released in 1995
  - Name derives from "a patchy server" ;-)
  - Software available online at http://www.apache.org
- Mozilla Web browser
  - http://www.mozilla.org/developer/
- Sendmail
  - http://www.sendmail.org/
- BIND Domain Name System
  - Client resolver and DNS server
  - http://www.isc.org/index.pl?/sw/bind/
- …

Link Layer Protocols

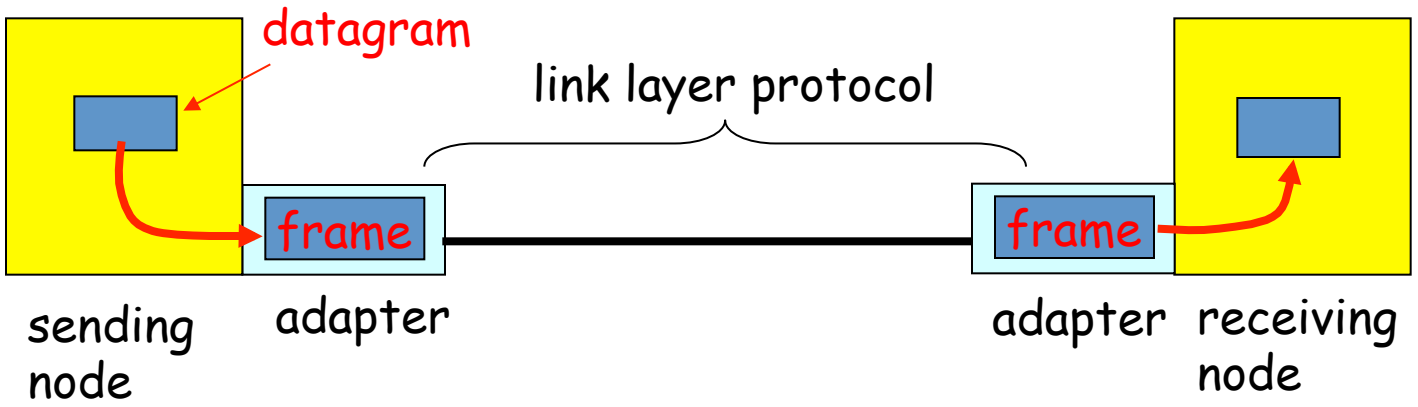
Topics

- Link-layer services (ch2 – 2.4: read on your own)
  - Encoding, framing, and error detection
  - Error correction and flow control
- Sharing a media
  - Channel partitioning
  - Taking turns
  - Random access
- Ethernet protocol
  - Carrier sense, collision detection, and random access
  - Frame structure
  - Hubs and switches

Message, Segment, Packet, and Frame



Adaptors Communicating



- Link layer implemented in adaptor (network interface card)
  - Ethernet card, PCMCIA card, 802.11 card
- Sending side:
  - Encapsulates datagram in a frame
  - Adds error checking bits, flow control, etc.
- Receiving side
  - Looks for errors, flow control, etc.
  - Extracts datagram and passes to receiving node

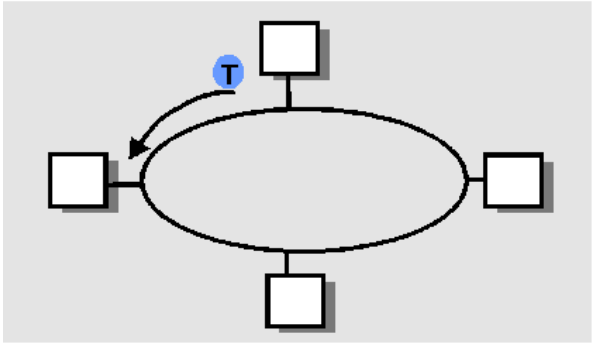
"Taking Turns" MAC protocols

Polling

- Master node "invites" slave nodes to transmit in turn
- Concerns:
  - Polling overhead
  - Latency
  - Single point of failure (master)

Token passing

- token passed from one node to next sequentially
- Concerns:
  - Token overhead
  - Latency
  - Single point of failure (token)
  - Token regeneration



Random Access Protocols

- When node has packet to send
  - Transmit at full channel data rate R.
  - No a priori coordination among nodes
- Two or more transmit: collision!
- Random access MAC protocol specifies:
  - How to detect collisions
  - How to recover from collisions
- Examples
  - ALOHA and Slotted ALOHA
  - CSMA, CSMA/CD, CSMA/CA

Key Ideas of Random Access

- Carrier sense
  - *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
  - *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

Slotted ALOHA

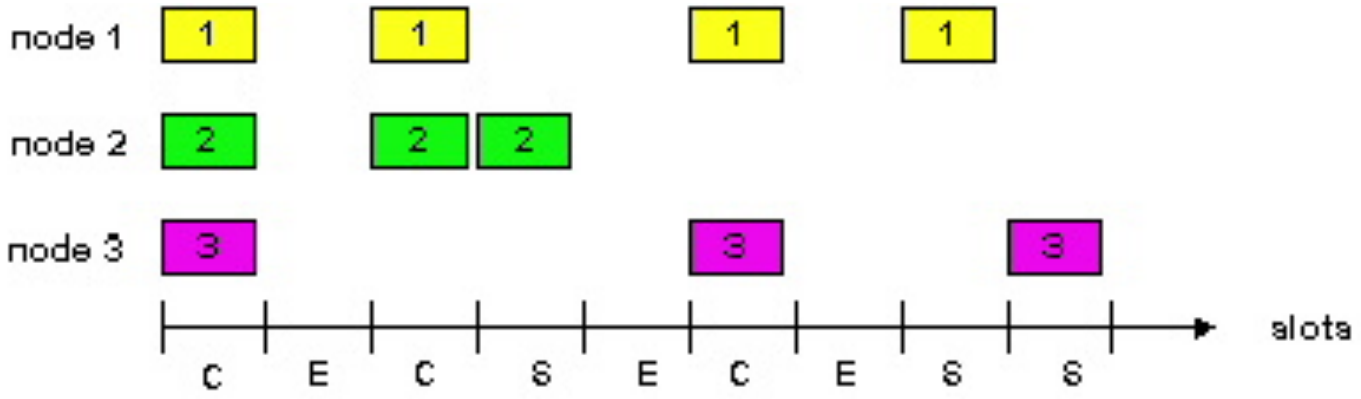
Assumptions

- All frames same size
- Time divided into equal slots (time to transmit a frame)
- Nodes start to transmit frames only at start of slots
- Nodes are synchronized
- If two or more nodes transmit, all nodes detect collision

Operation

- When node obtains fresh frame, transmits in next slot (no carrier sense)
- No collision: node can send new frame in next slot
- Collision: node retransmits frame in each subsequent slot with probability p until success

Slotted ALOHA



Pros

- Single active node can continuously transmit at full rate of channel
- Highly decentralized: only slots in nodes need to be in sync
- Simple

Cons

- Collisions, wasting slots
- Idle slots
- Nodes may be able to detect collision in less than time to transmit packet
- Clock synchronization

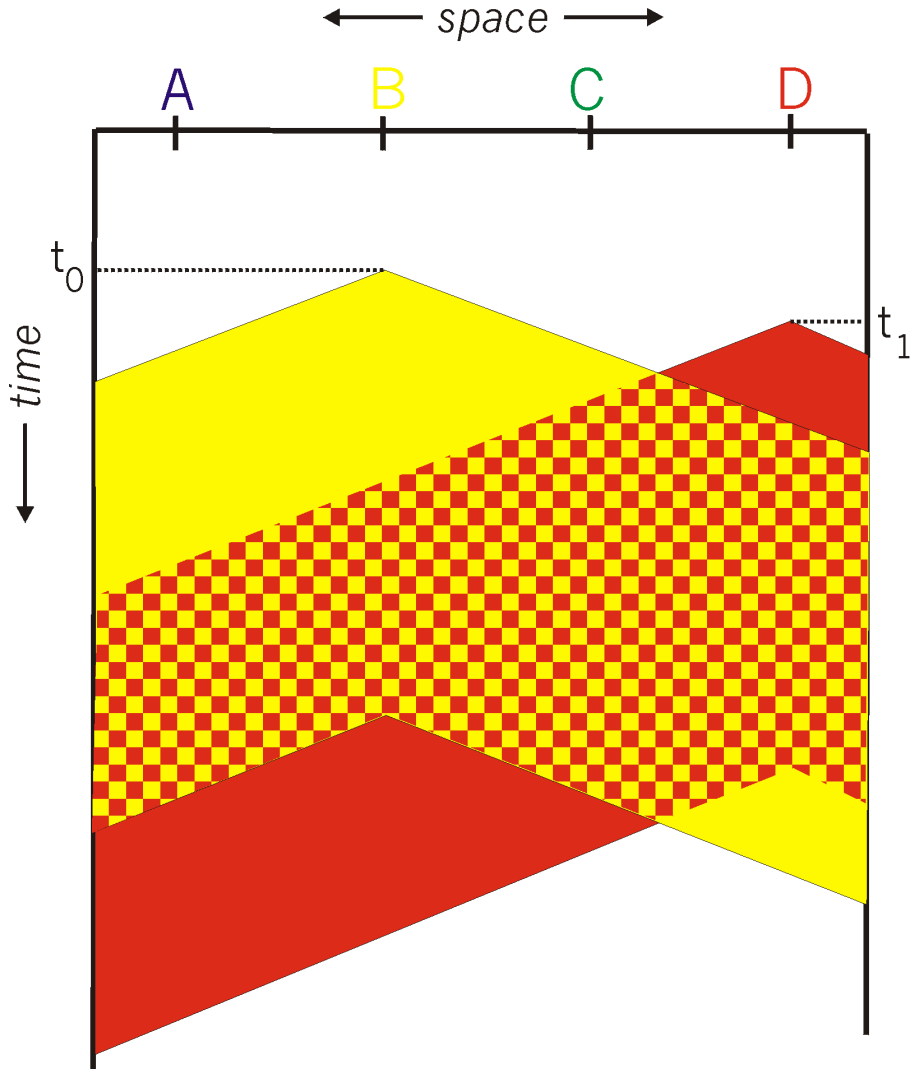
CSMA (Carrier Sense Multiple Access)

- Collisions hurt the efficiency of ALOHA protocol
  - At best, channel is useful 37% of the time
- CSMA: listen before transmit
  - If channel sensed idle: transmit entire frame
  - If channel sensed busy, defer transmission
- Human analogy: don't interrupt others!

CSMA Collisions

Collisions *can* still occur:
propagation delay means
two nodes may not hear
each other's transmission

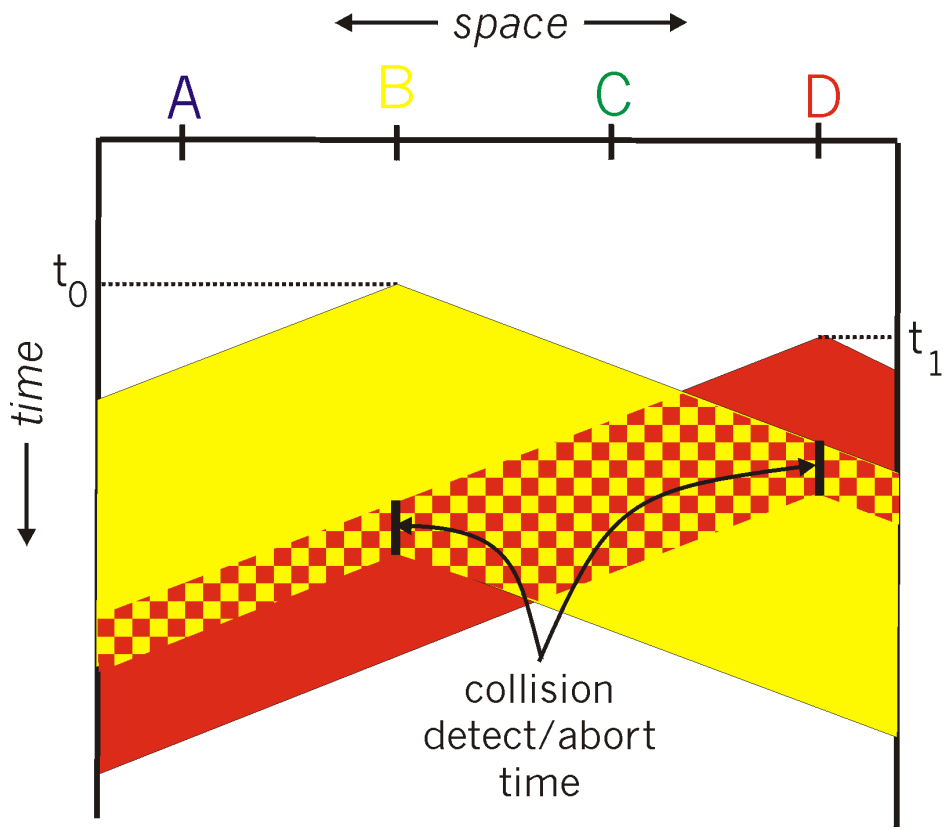
Collision:
entire packet transmission
time wasted



CSMA/CD (Collision Detection)

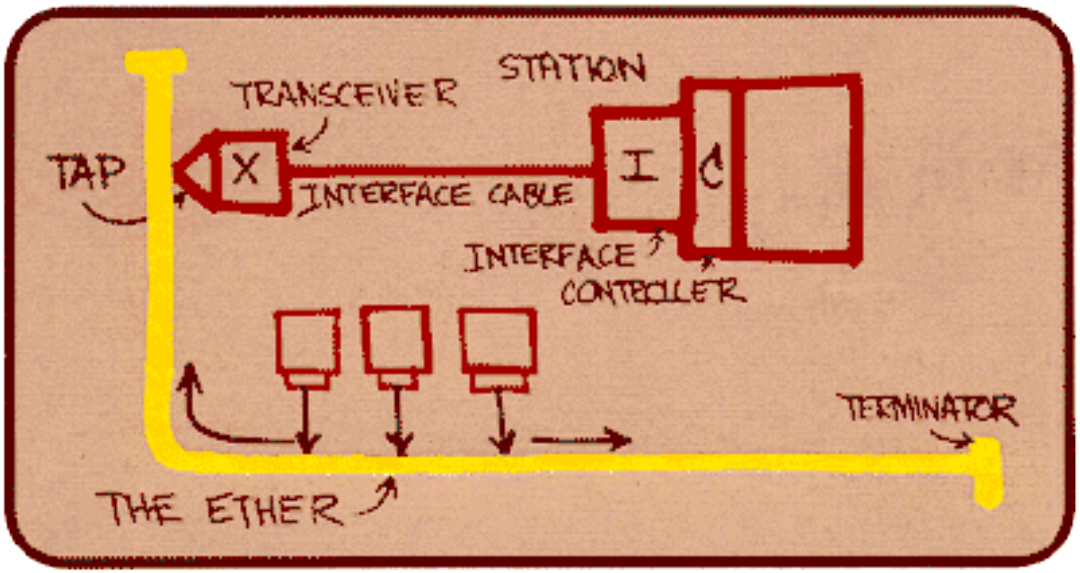
- CSMA/CD: carrier sensing, deferral as in CSMA
  - Collisions detected within short time
  - Colliding transmissions aborted, reducing wastage
- Collision detection
  - Easy in wired LANs: measure signal strengths, compare transmitted, received signals
  - Difficult in wireless LANs: receiver shut off while transmitting
- Human analogy: the polite conversationalist

CSMA/CD Collision Detection



Ethernet

- Dominant wired LAN technology:
- First widely used LAN technology
- Simpler, cheaper than token LANs and ATM
- Kept up with speed race: 10 Mbps – 10 Gbps or more

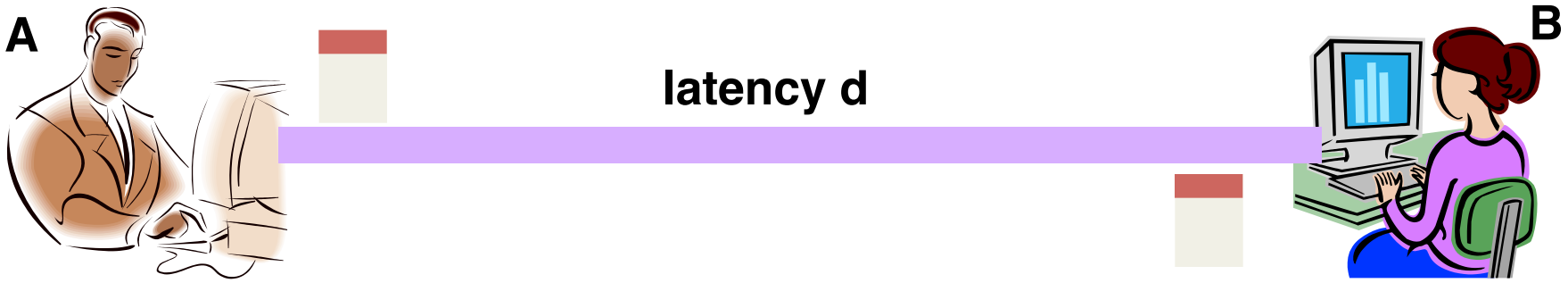


Metcalfe's Ethernet sketch

Ethernet Uses CSMA/CD

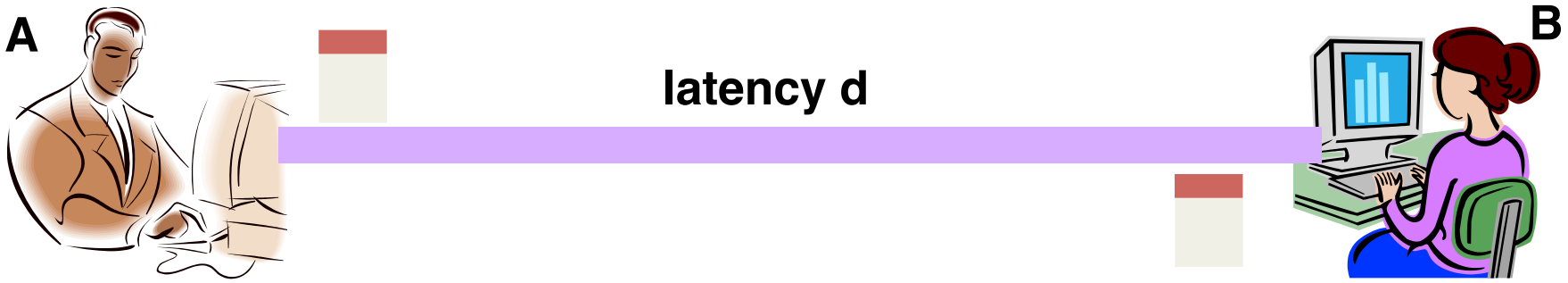
- Carrier sense: wait for link to be idle
  - Channel idle: start transmitting
  - Channel busy: wait until idle
- Collision detection: listen while transmitting
  - No collision: transmission is complete
  - Collision: abort transmission, send jam signal
- Random access: exponential back-off
  - After collision, wait a random time before trying again
  - After m^{th} collision, pick K randomly from $\{0, \ldots, 2^m-1\}$
  - … and wait for K*512 bit times before trying again

Limitations on Ethernet Length



- Latency depends on physical length of link
  - Time to propagate a packet from one end to the other
- Suppose A sends a packet at time t
  - And B sees an idle line at a time just before t+d
  - … so B happily starts transmitting a packet
- B detects a collision, and sends jamming signal
  - But A doesn't see collision till t+2d

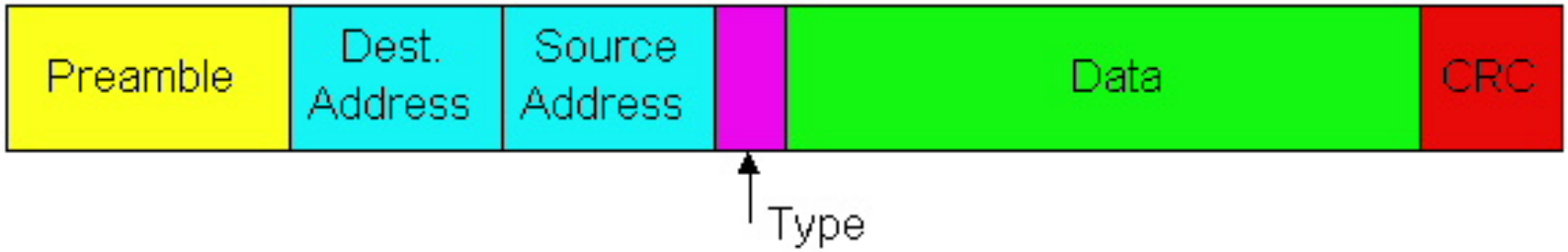
Limitations on Ethernet Length



- A needs to wait for time 2d to detect collision
  - So, A keeps transmitting during this period
  - … and keep an eye out for a possible collision
- Imposes restrictions on Ethernet
  - Maximum length of the wire: 2500 meters
  - Minimum length of the packet: 512 bits (64 bytes)
- Limitations less relevant with switched networks?
  - Still have to do broadcast..

Ethernet Frame Structure

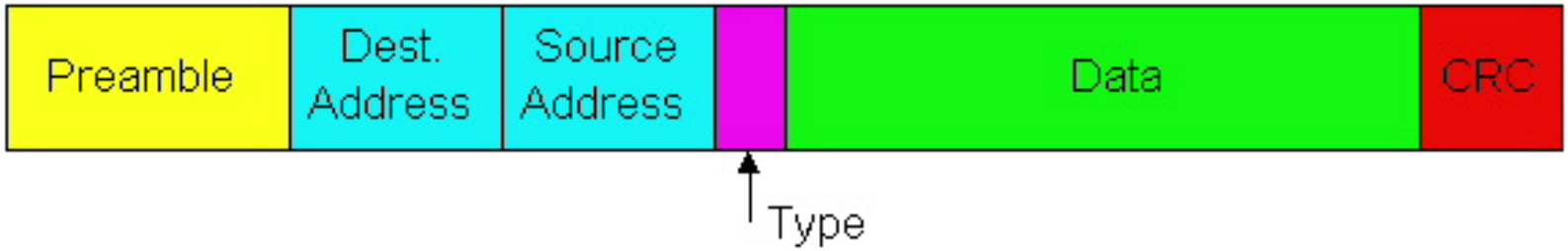
- Sending adapter encapsulates packet in frame



- Preamble: synchronization
  - Seven bytes with pattern 10101010, followed by one byte with pattern 10101011
  - Used to synchronize receiver, sender clock rates

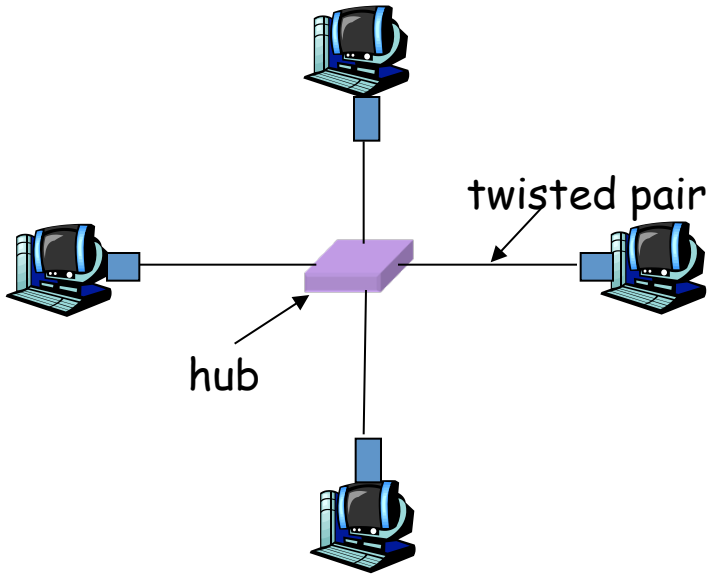
Ethernet Frame Structure (Cont.)

- Addresses: source and destination MAC addresses
  - Adaptor passes frame to network-level protocol
    - If destination address matches the adaptor
    - Or the destination address is the broadcast address
  - Otherwise, adapter discards frame
- Type: indicates the higher layer protocol
  - Usually IP
  - But also Novell IPX, AppleTalk, …
- CRC: cyclic redundancy check
  - Checked at receiver



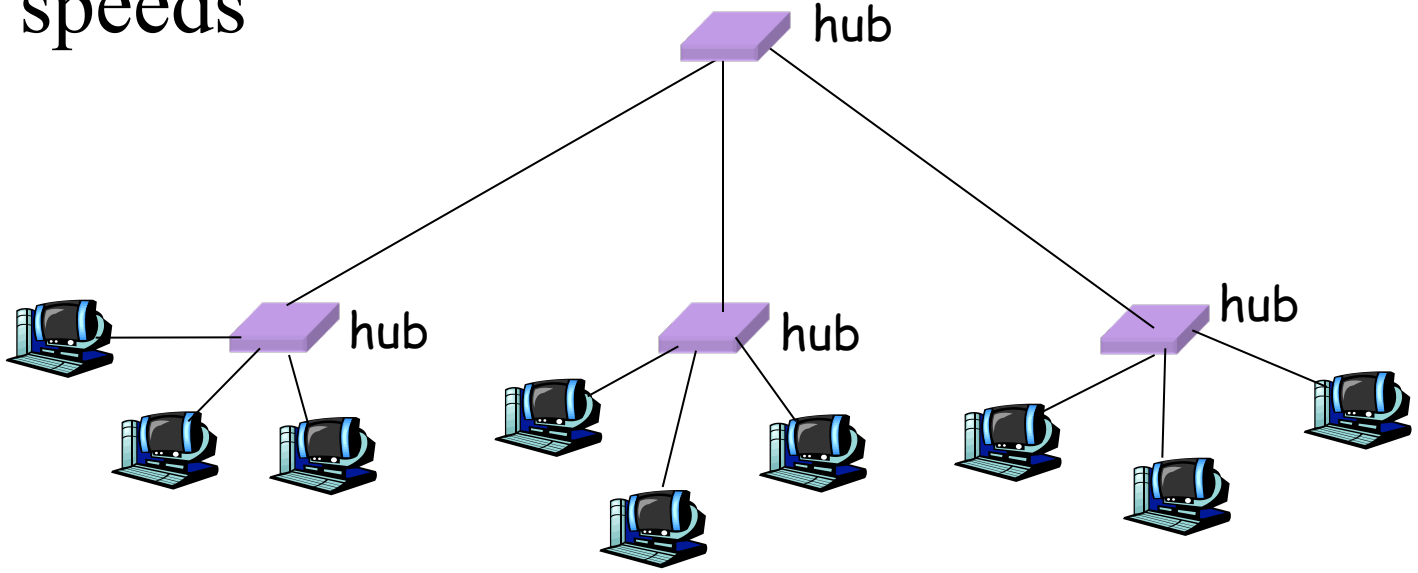
Hubs: Physical-Layer Repeaters

- Hubs are physical-layer repeaters
  - Bits coming from one link go out all other links
  - At the same rate, with no frame buffering
  - No CSMA/CD at hub: adapters detect collisions



Interconnecting with Hubs

- Backbone hub interconnects LAN segments
- All packets seen everywhere, forming one large collision domain
- Can't interconnect Ethernets of different speeds

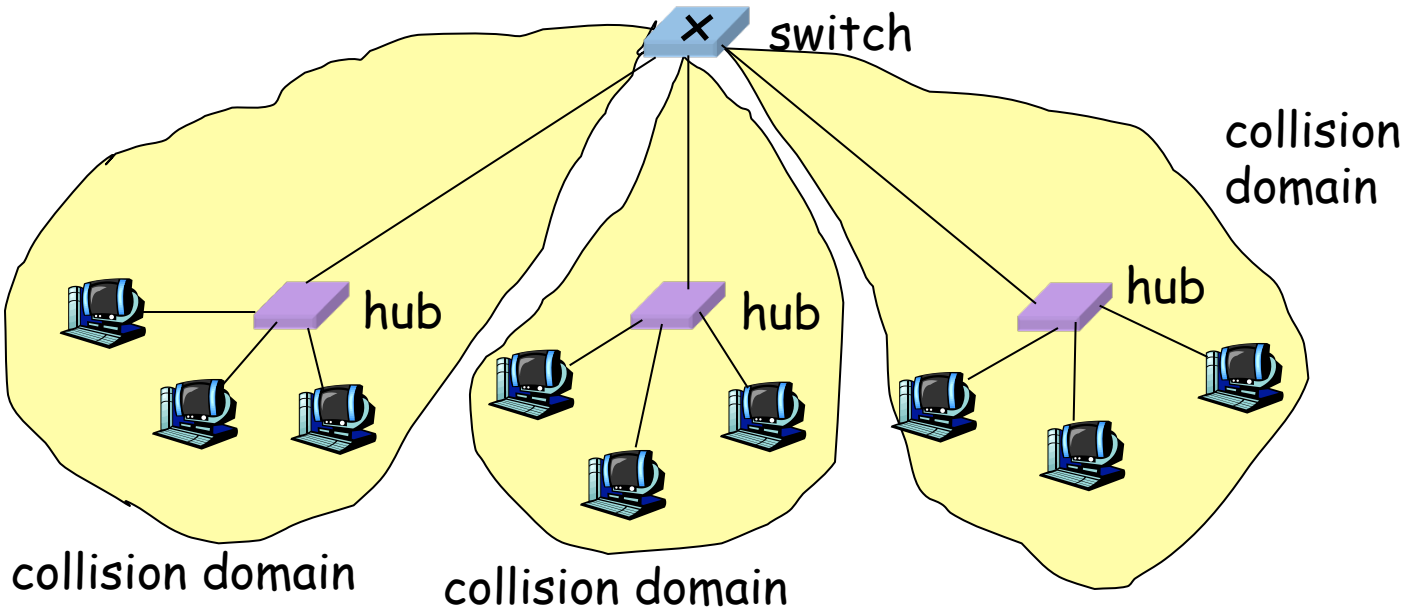


Switch

- Link layer device
  - Stores and forwards Ethernet frames
  - Examines frame header and selectively forwards frame based on MAC dest address
  - When frame is to be forwarded on segment, uses CSMA/CD to access segment
- Transparent
  - Hosts are unaware of presence of switches
- Plug-and-play, self-learning
  - Switches do not need to be configured

Switch: Traffic Isolation

- Switch breaks subnet into LAN segments
- Switch filters packets
  - Same-LAN-segment frames not usually forwarded onto other LAN segments
  - Segments become separate collision domains



Benefits of Ethernet

- Easy to administer and maintain
- Inexpensive
- Increasingly higher speed
- Moved from shared media to switches
  - Change everything except the frame format
  - A good general lesson for evolving the Internet

Conclusions

- IP runs on a variety of link layer technologies
  - Point-to-point links vs. shared media
  - Wide varieties within each class
- Link layer performs key services
  - Encoding, framing, and error detection
  - Optionally error correction and flow control
- Shared media introduce interesting challenges
  - Decentralized control over resource sharing
  - Partitioned channel, taking turns, and random access
  - Ethernet as a wildly popular example