inst.eecs.berkeley.edu/~cs61c

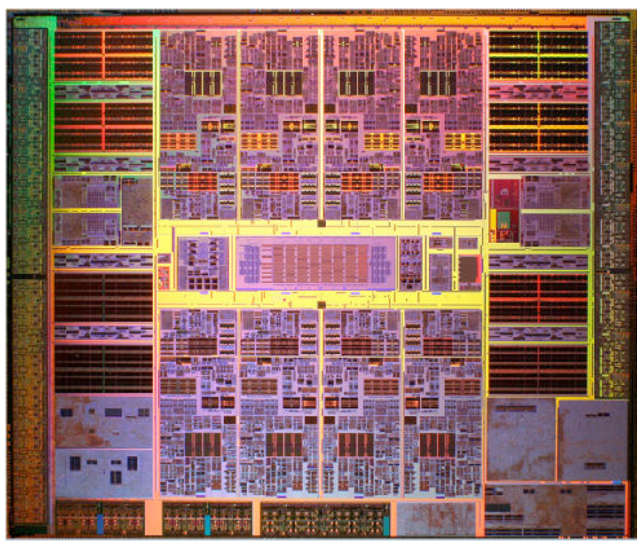
CS61C : Machine Structures

Lecture #27 I/O Basics & Networking

2007-8-9

Scott Beamer, Instructor

Sun Releases New Processor

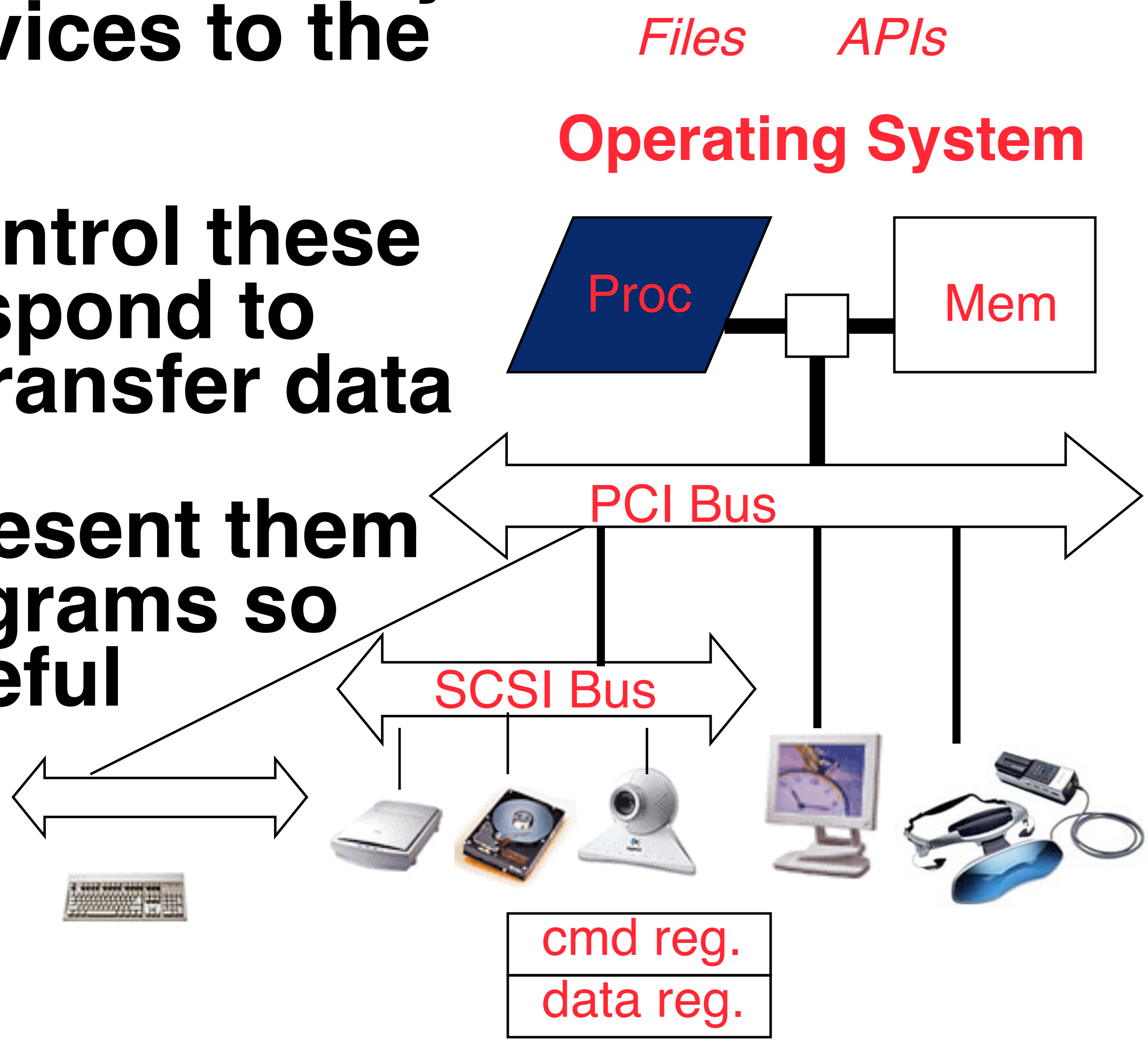


64 Threads in 1 Package!!

www.sun.com

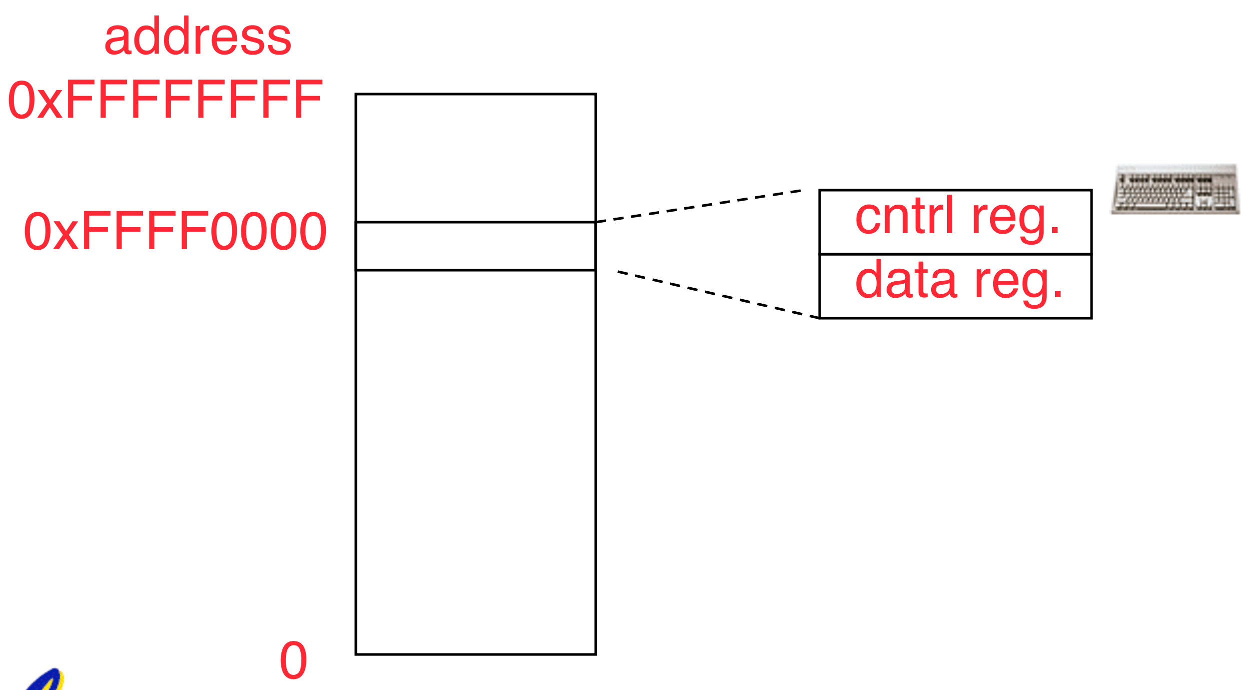
What do we need to make I/O work?

- A way to connect many types of devices to the Proc-Mem
- A way to control these devices, respond to them, and transfer data
- A way to present them to user programs so they are useful



Memory Mapped I/O

- Certain addresses are not regular memory
- Instead, they correspond to registers in I/O devices

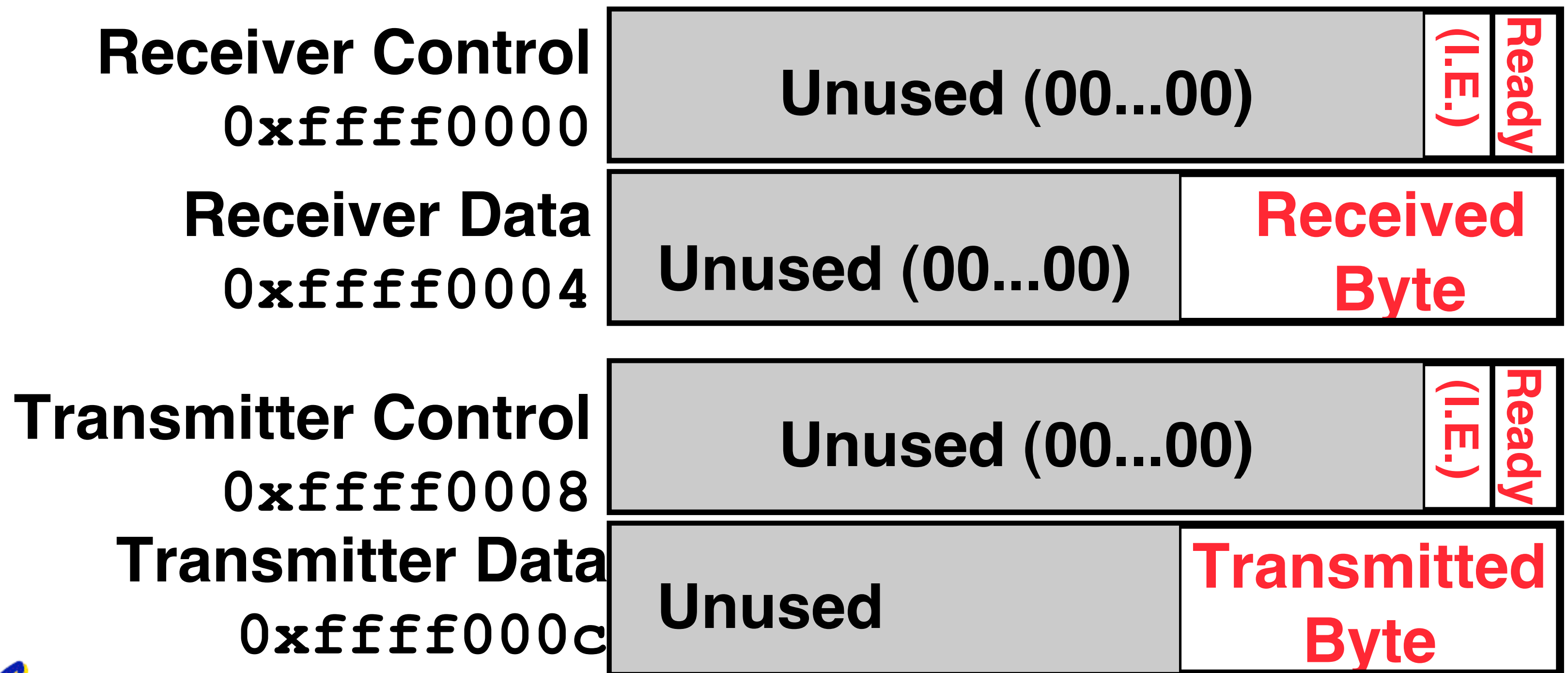


Processor Checks Status before Acting

- Path to device generally has 2 registers:
  - Control Register, says it's OK to read/write (I/O ready) [think of a flagman on a road]
  - Data Register, contains data
- Processor reads from Control Register in loop, waiting for device to set Ready bit in Control reg (0 ⇒ 1) to say its OK
- Processor then loads from (input) or writes to (output) data register
  - Load from or Store into Data Register resets Ready bit (1 ⇒ 0) of Control Register

SPIM I/O Simulation

- SPIM simulates 1 I/O device: memory-mapped terminal (keyboard + display)
  - Read from keyboard (receiver); 2 device regs
  - Writes to terminal (transmitter); 2 device regs



SPIM I/O

- Control register rightmost bit (0): Ready
  - Receiver: Ready==1 means character in Data Register not yet been read;
    1 ⇒ 0 when data is read from Data Reg
  - Transmitter: Ready==1 means transmitter is ready to accept a new character;
    0 ⇒ Transmitter still busy writing last char
    - I.E. bit discussed later
- Data register rightmost byte has data
  - Receiver: last char from keyboard; rest = 0
  - Transmitter: when write rightmost byte, writes char to display

I/O Example

- **Input: Read from keyboard into `$v0`**

```
Waitloop:   lui  $t0, 0xffff #ffff0000
            lw   $t1, 0($t0) #control
            andi $t1,$t1,0x1
            beq  $t1,$zero, Waitloop
            lw   $v0, 4($t0) #data
```

- **Output: Write to display from `$a0`**

```
Waitloop:   lui  $t0, 0xffff #ffff0000
            lw   $t1, 8($t0) #control
            andi $t1,$t1,0x1
            beq  $t1,$zero, Waitloop
            sw   $a0, 12($t0) #data
```

- **Processor waiting for I/O called "Polling"**
- **"Ready" bit is from processor's point of view!**

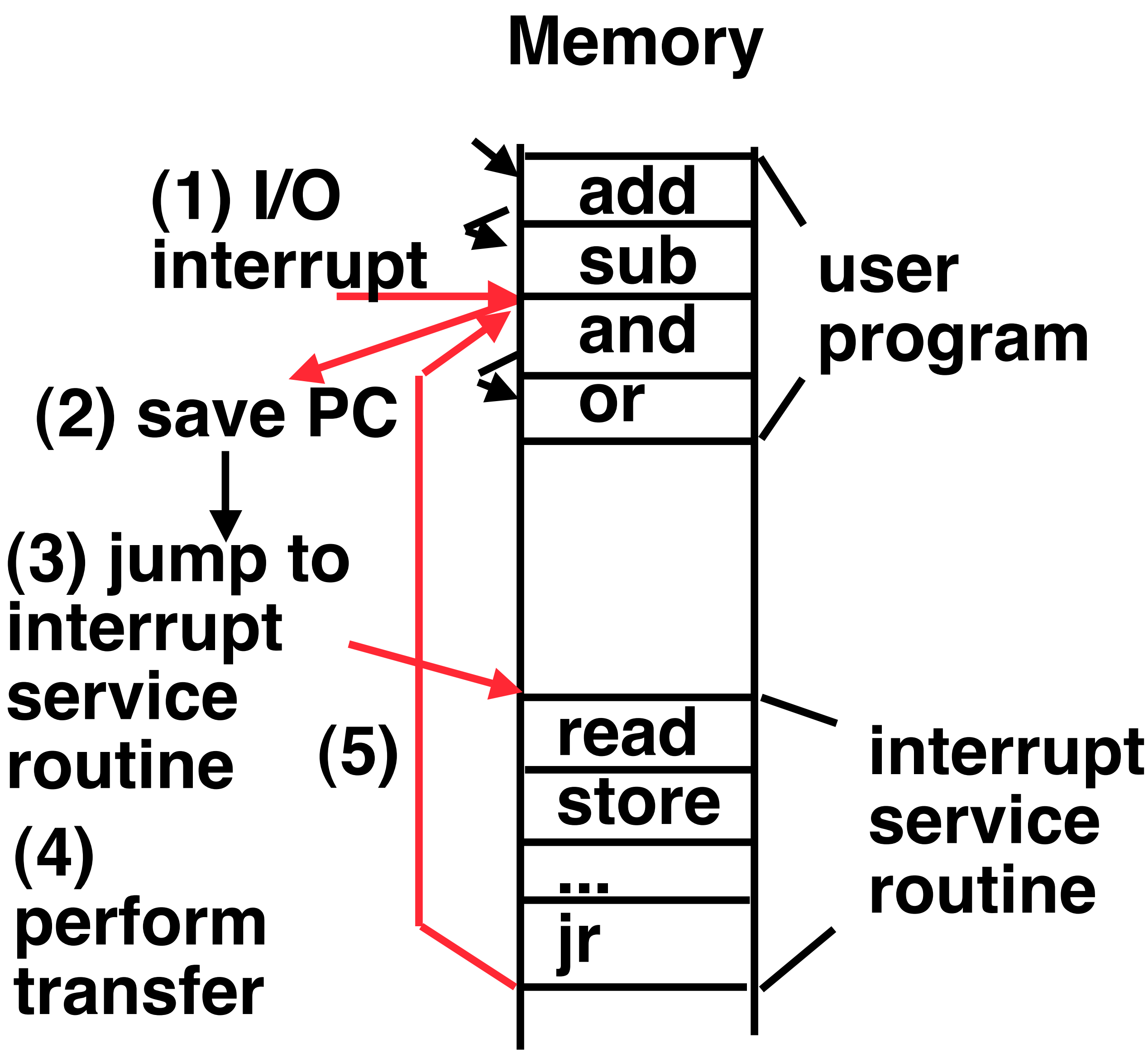
What is the alternative to polling?

- Wasteful to have processor spend most of its time "spin-waiting" for I/O to be ready
- Would like an unplanned procedure call that would be invoked only when I/O device is ready
- Solution: use exception mechanism to help I/O. Interrupt program when I/O ready, return when done with data transfer

I/O Interrupt

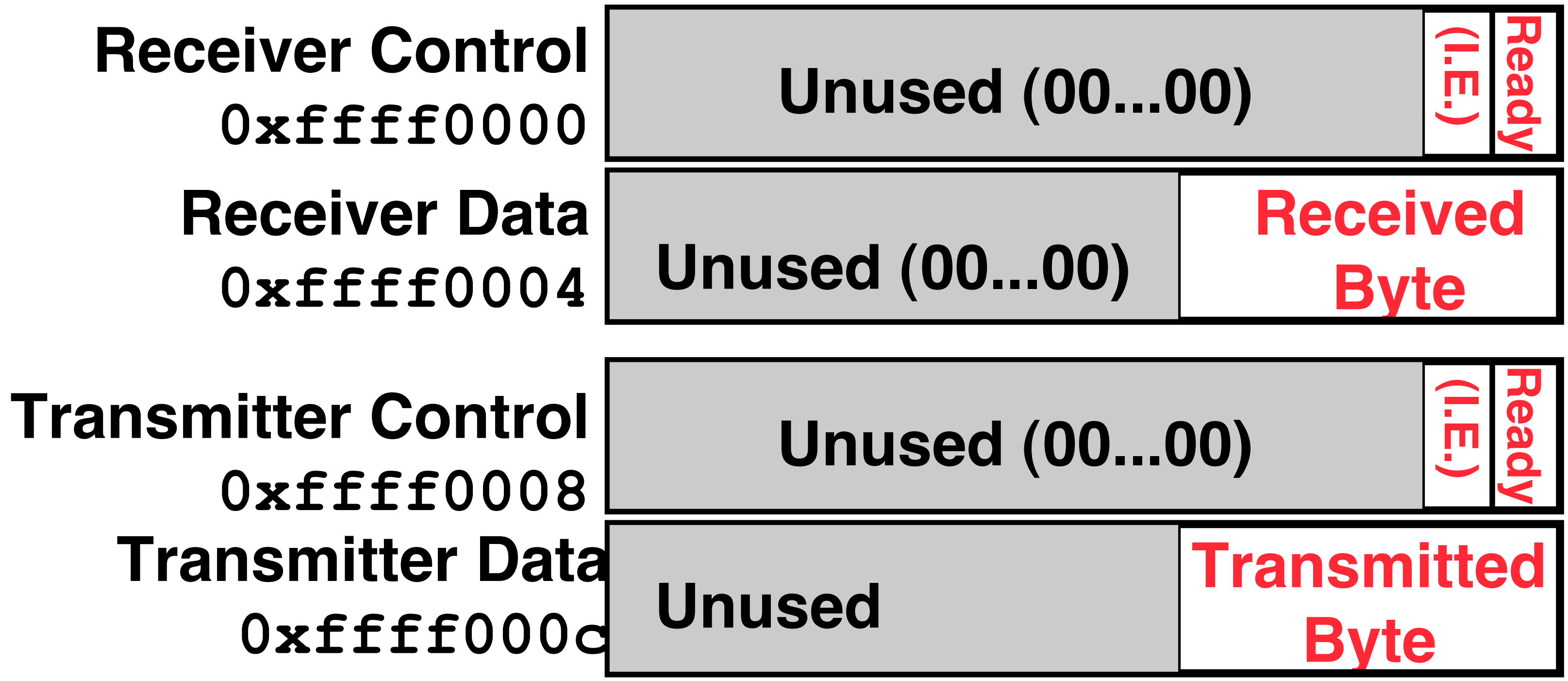
- An I/O interrupt is like overflow exceptions except:
  - An I/O interrupt is "asynchronous"
  - More information needs to be conveyed
- An I/O interrupt is asynchronous with respect to instruction execution:
  - I/O interrupt is not associated with any instruction, but it can happen in the middle of any given instruction
  - I/O interrupt does not prevent any instruction from completion

Interrupt-Driven Data Transfer



SPIM I/O Simulation: Interrupt Driven I/O

- I.E. stands for <u>Interrupt Enable</u>
- Set Interrupt Enable bit to 1 have interrupt occur whenever Ready bit is set



Peer Instruction

A. A faster CPU will result in faster I/O.

B. Hardware designers handle mouse input with interrupts since it is better than polling in almost all cases.

C. Low-level I/O is actually quite simple, as it's really only reading and writing bytes.

	ABC
0:	FFF
1:	FFT
2:	FTF
3:	FTT
4:	TFF
5:	TFT
6:	TTF
7:	TTT

Peer Instruction Answer

A. Less sync data idle time

B. Because mouse has low I/O rate polling often used

C. Concurrency, device requirements vary!

A. A faster CPU will result in faster I/O. **TRUE**

B. Hardware designers handle mouse input with interrupts since it is better than polling in almost all cases. **FALSE**

C. Low-level I/O is actually quite simple, as it's really only reading and writing bytes. **FALSE**

	ABC
0:	FFF
1:	FFT
2:	FTF
3:	FTT
4:	**TFF**
5:	TFT
6:	TTF
7:	TTT

"And in early conclusion…"

- I/O gives computers their 5 senses
- I/O speed range is 100-million to one
- Processor speed means must synchronize with I/O devices before use
- Polling works, but expensive
  - processor repeatedly queries devices
- Interrupts works, more complex
  - devices causes an exception, causing OS to run and deal with the device
- I/O control leads to Operating Systems

Administrivia

- Assignments
  - Proj4 due 8/12
  - HW8 due 8/14
- Final Review Session probable on Monday
- Course Survey during last lecture
  - 2 points extra added for taking survey (still anonymous)
- Grading done for HW1-4 & Proj1

Why Networks?

- Originally *sharing I/O devices* between **computers**
  - ex: printers
- Then *communicating* between **computers**
  - ex: file transfer protocol
- Then *communicating* between **people**
  ex: e-mail
- Then *communicating* between **networks of computers**
  - ex: file sharing, www, …

How Big is the Network (2007)?

~30 in **273 Soda**

~525 in **inst.cs.berkeley.edu**

~6,400 in **eecs & cs .berkeley.edu**

(1999) ~50,000 in **berkeley.edu**

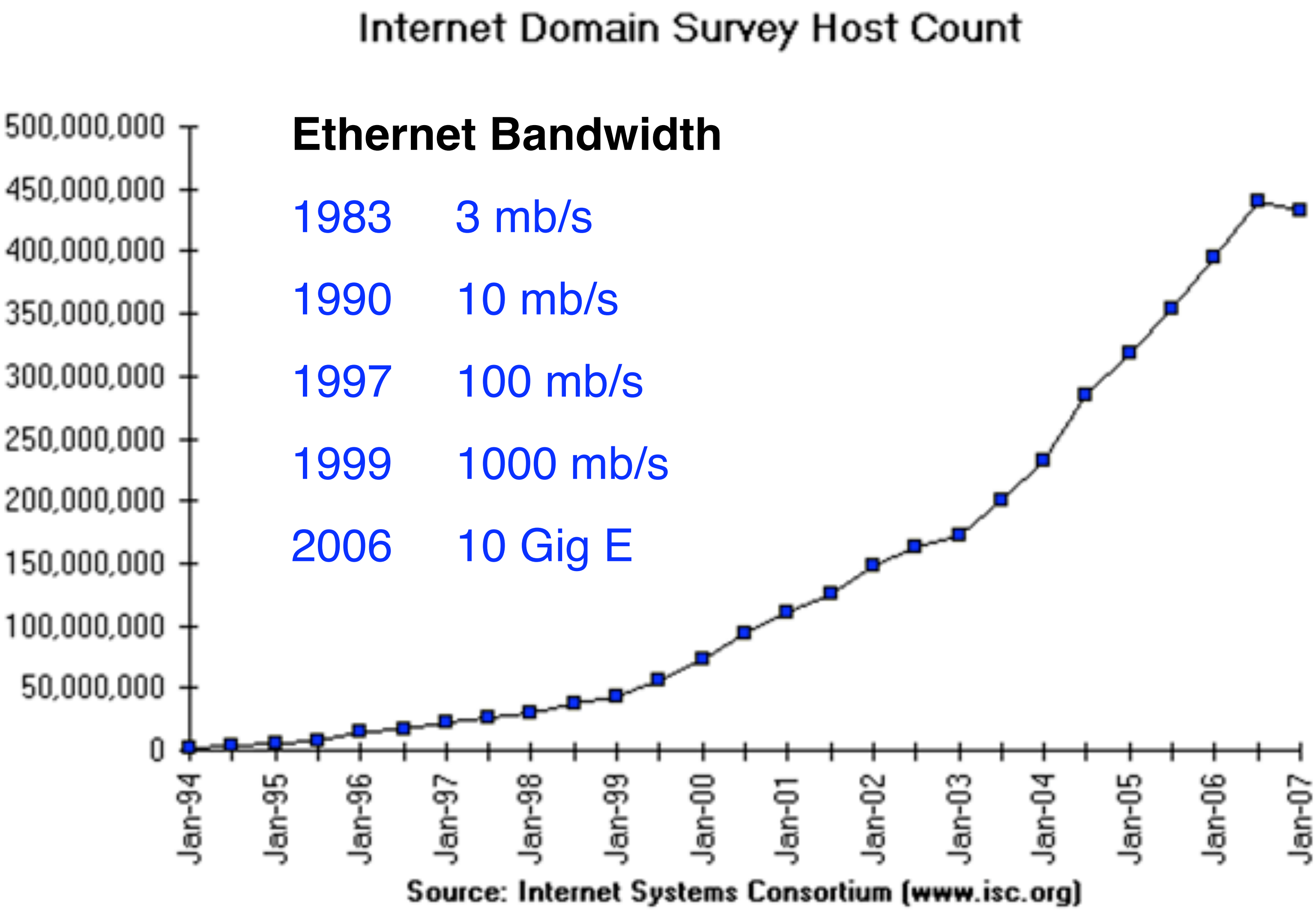
~10,000,000 in **.edu** **(2005: ~9,000,000)**

~258,941,310 in **US** **(2005: ~217,000,000, 2006: ~286.5E6)**
(.net .com .edu .arpa .us .mil .org .gov)

~433,190,000 in **the world**
(2005:~317,000,000, 2006: ~439,000,000)

Source: Internet Software Consortium: www.isc.org

Growth Rate



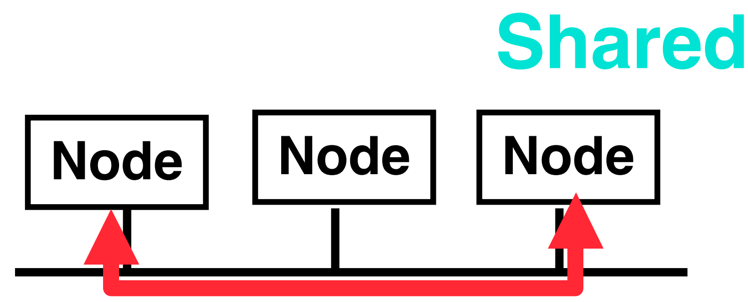
Ethernet Bandwidth

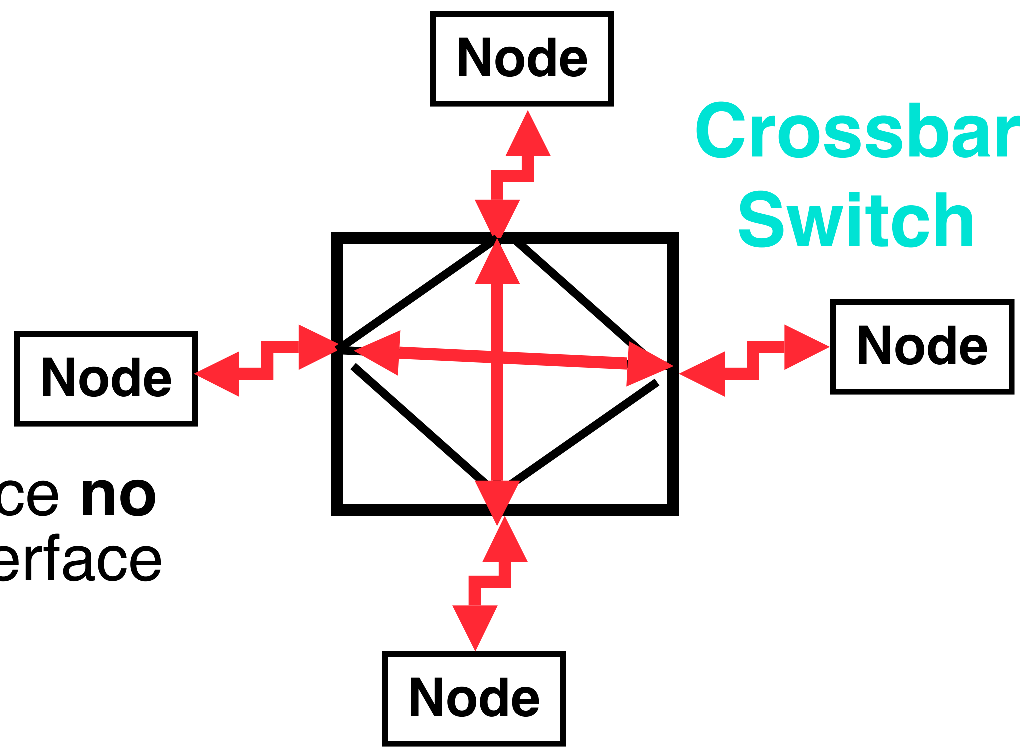
1983	3 mb/s
1990	10 mb/s
1997	100 mb/s
1999	1000 mb/s
2006	10 Gig E

en.wikipedia.org/wiki/10_gigabit_ethernet

Shared vs. Switched Based Networks

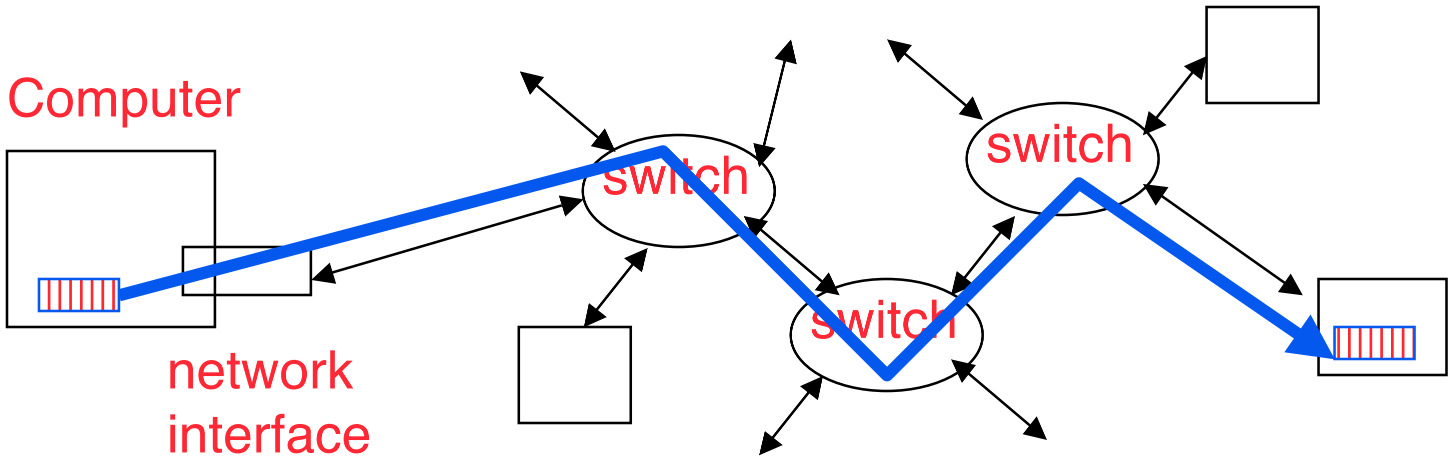
- **Shared vs. Switched:**
  - **Switched:** pairs ("point-to-point" connections) communicate at same time
  - **Shared:** 1 at a time (CSMA/CD)
- **Aggregate bandwidth (BW) in switched network is many times shared:**
  - point-to-point faster since **no arbitration**, simpler interface





What makes networks work?

- **links** connecting **switches** to each other and to computers or devices



- ability to **name** the components and to **route** packets of information - messages - from a source to a destination

- Layering, redundancy, protocols, and encapsulation as means of
  abstraction (61C big idea)

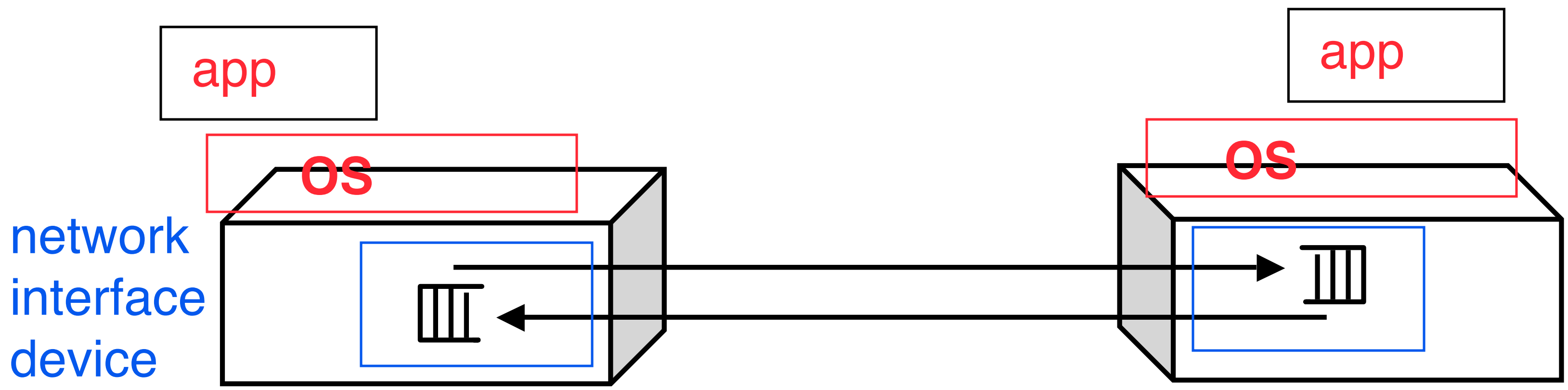
Typical Types of Networks

- Local Area Network (Ethernet)
  - Inside a building: Up to 1 km
  - (peak) Data Rate: 10 Mbits/sec, 100 Mbits /sec,1000 Mbits/sec (1.25, 12.5, 125 MBytes/s)
  - Run, installed by network administrators
- Wide Area Network
  - Across a continent (10km to 10000 km)
  - (peak) Data Rate: 1.5 Mb/s to 40000 Mb/s
  - Run, installed by telecommunications companies (Sprint, UUNet[MCI], AT&T)
- Wireless Networks (LAN), ...

The Sprint U.S. Topology (2001)

ABCs of Networks: 2 Computers

- **Starting Point: Send bits between 2 computers**



- **Queue (First In First Out) on each end**
- **Can send both ways ("Full Duplex")**
  - **One-way information is called "Half Duplex"**
- **Information sent called a "message"**
  - **Note: Messages also called packets**

A Simple Example: 2 Computers

- What is Message Format?
  - Similar idea to Instruction Format
  - Fixed size? Number bits?

- Header (Trailer): information to deliver message
- Payload: data in message
- What can be in the data?
  - anything that you can represent as bits
  - values, chars, commands, addresses...

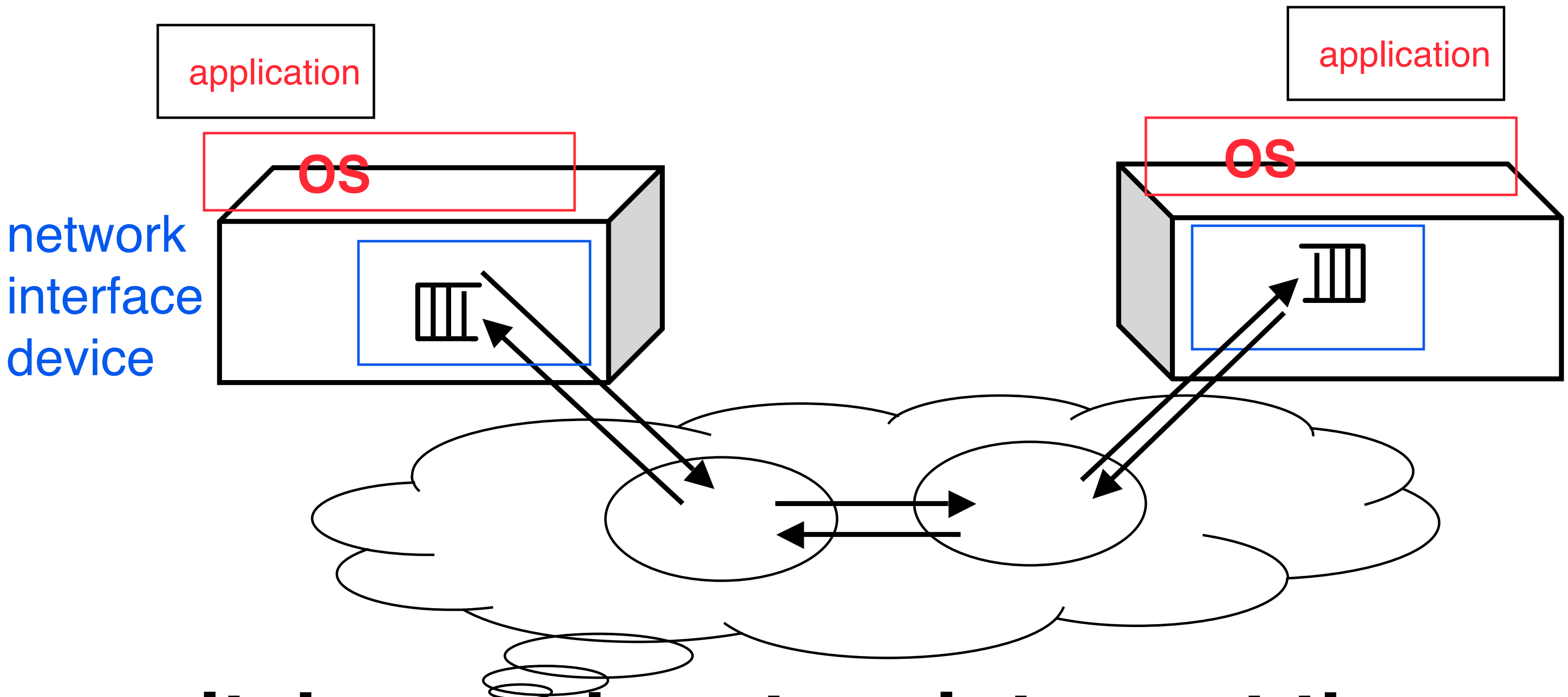
Questions About Simple Example

- What if more than 2 computers want to communicate?
  - Need computer "address field" in packet to know:
    - which computer should receive it (destination)
    - which computer to reply to (source)
  - Just like envelopes!

Dest.	Source	Len	
Net ID	Net ID		CMD/ Address /Data
8 bits	8 bits	8 bits	32*n bits

Header — Payload

ABCs: many computers



- switches and routers interpret the header in order to deliver the packet
- source encodes and destination decodes content of the payload

Questions About Simple Example

- **What if message is garbled in transit?**
- **Add redundant information that is checked when message arrives to be sure it is OK**
- **8-bit sum of other bytes: called "Check sum"; upon arrival compare check sum to sum of rest of information in message. `xor` also popular.**

Learn about Checksums in Math 55/CS 70…

Questions About Simple Example

- **What if message never arrives?**
- **Receiver tells sender when it arrives**
  - **Send an ACK (ACKnowledgement) [like registered mail]**
  - **Sender retries if waits too long**
- **Don't discard message until it is ACK'ed**
- **If check sum fails, don't send ACK**

Observations About Simple Example

- **Simple questions (like those on the previous slides) lead to:**
    - **more complex procedures to send/receive message**
    - **more complex message formats**

- **Protocol: algorithm for properly sending and receiving messages (packets)**

…an agreement on how to communicate

Software Protocol to Send and Receive

- **SW Send steps**
  - 1: Application copies data to OS buffer
  - 2: OS calculates checksum, starts timer
  - 3: OS sends data to network interface HW and says start
- **SW Receive steps**
  - 3: OS copies data from network interface HW to OS buffer
  - 2: OS calculates checksum, if OK, send ACK; if not, delete message (sender resends when timer expires)
  - 1: If OK, OS copies data to user address space, & signals application to continue

Protocol for Networks of Networks?

- Abstraction to cope with complexity of communication
- Networks are like onions
  - Hierarchy of layers:
    - Application (chat client, game, etc.)
    - Transport (TCP, UDP)
    - Network (IP)
    - Physical Link (wired, wireless, etc.)

Networks are like onions.

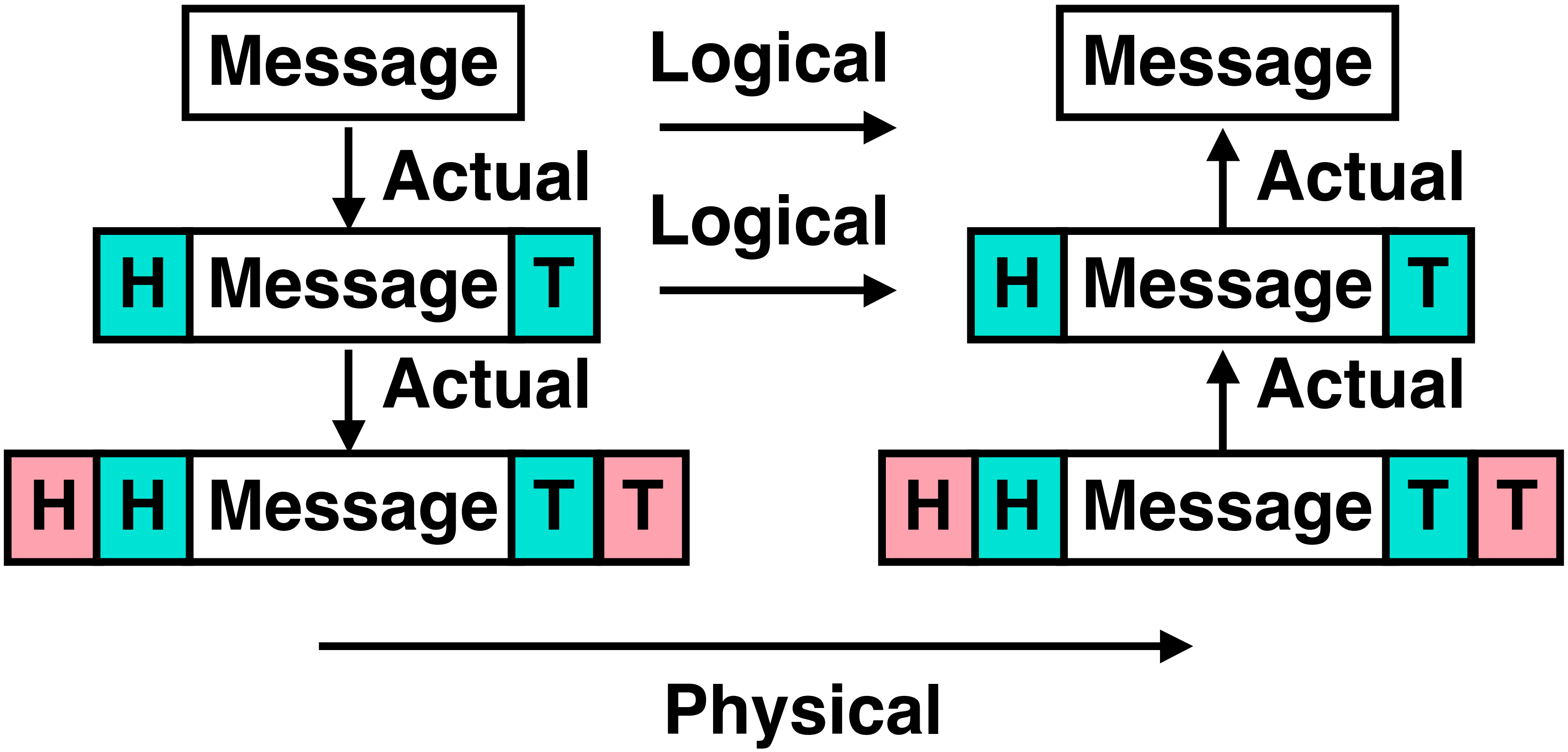
They stink?

Yes. No!

Oh, they make you cry.

No!… Layers. Onions have layers. Networks have layers.

Protocol Family Concept



Protocol Family Concept

- **Key to protocol families is that communication occurs logically at the same level of the protocol, called peer-to-peer…**

  …but is implemented via services at the next lower level

- **Encapsulation: carry higher level information within lower level "envelope"**

- **Fragmentation: break packet into multiple smaller packets and reassemble**

Protocol for Network of Networks

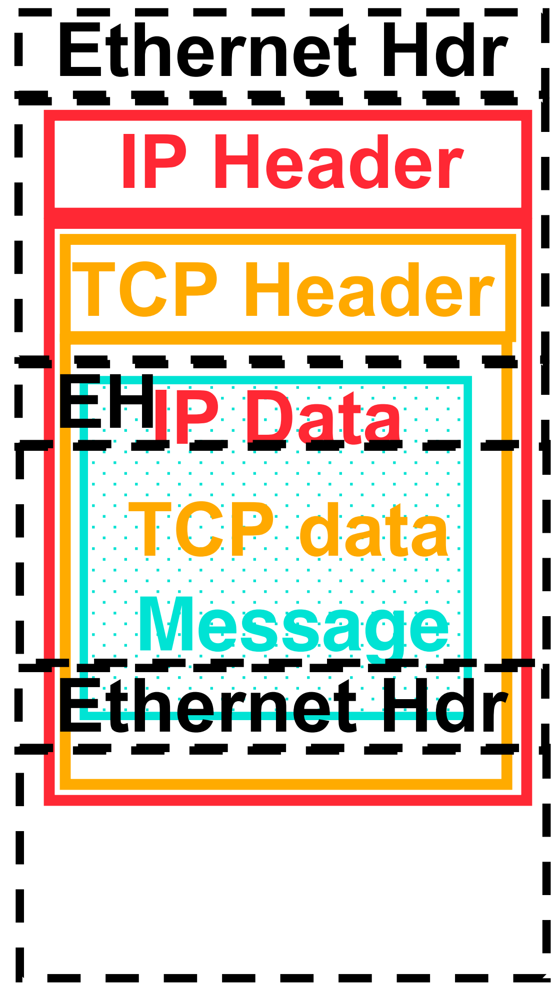
- **IP: Best-Effort Packet Delivery (Network Layer)**
- Packet switching
  - Send data in packets
  - Header with source & destination address
- "Best effort" delivery
  - Packets may be lost
  - Packets may be corrupted
  - Packets may be delivered out of order

Protocol for Network of Networks

- Transmission Control Protocol/Internet Protocol (TCP/IP) (TCP :: a Transport Layer)
  - This protocol family is the basis of the Internet, a WAN protocol
  - IP makes best effort to deliver
  - TCP guarantees delivery
  - TCP/IP so popular it is used even when communicating locally: even across homogeneous LAN

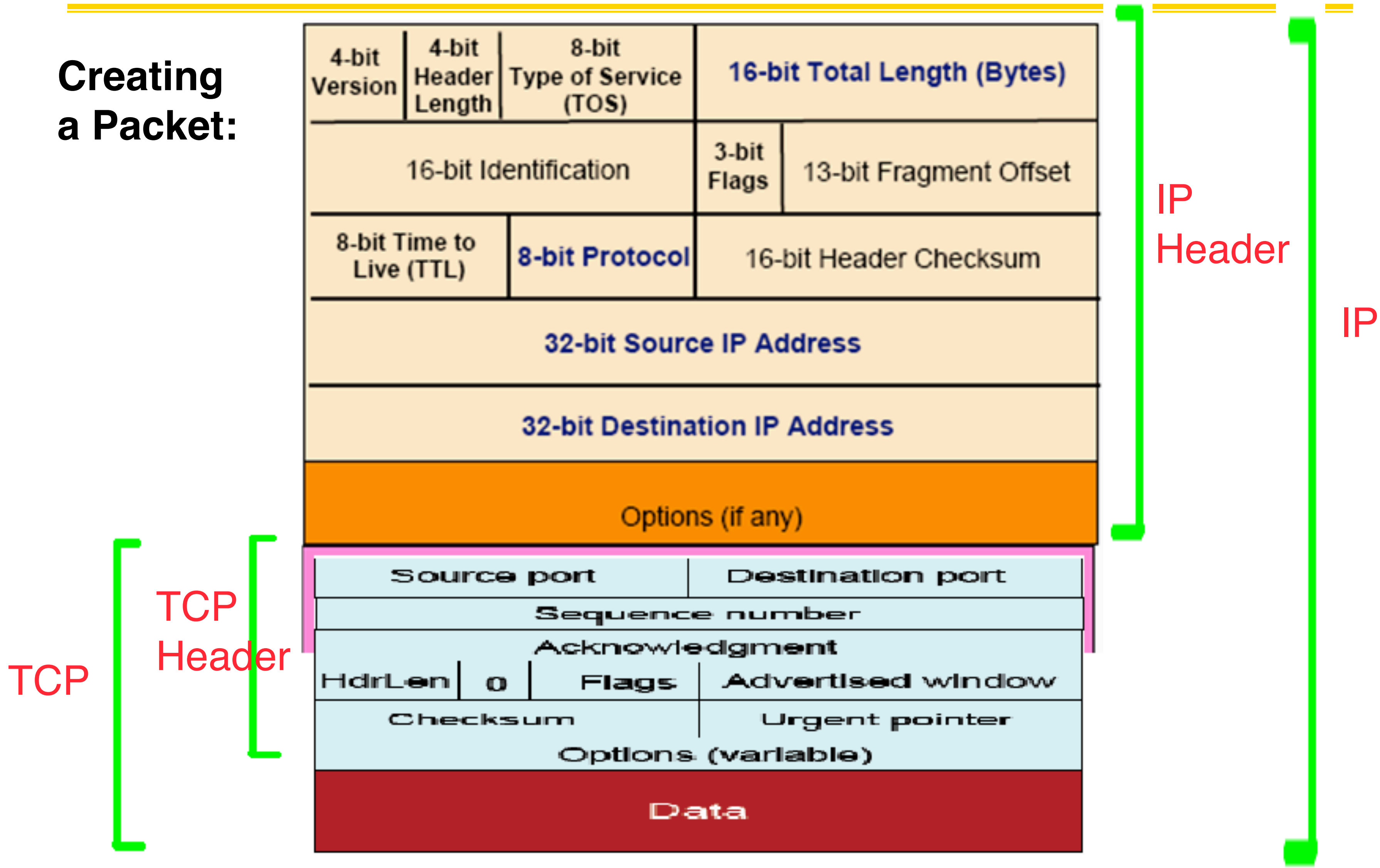
TCP/IP packet, Ethernet packet, protocols

- Application sends message
- TCP breaks into 64KiB segments, adds 20B header
- IP adds 20B header, sends to network
- If Ethernet, broken into 1500B packets with headers, trailers (24B)
- All Headers, trailers have length field, destination, ...



TCP/IP in action

Creating a Packet:



Overhead vs. Bandwidth

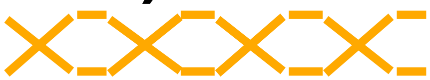
- Networks are typically advertised using peak bandwidth of network link: e.g., 100 Mbits/sec Ethernet ("100 base T")
- Software overhead to put message into network or get message out of network often limits useful bandwidth
- Assume overhead to send and receive = 320 microseconds (μs), want to send 1000 Bytes over "100 Mbit/s" Ethernet
  - Network transmission time: 1000Bx8b/B /100Mb/s = 8000b / (100b/μs) = 80 μs
  - Effective bandwidth: 8000b/(320+80)μs = 20 Mb/s

And in conclusion…

- Protocol suites allow networking of heterogeneous components
  - Another form of principle of abstraction
  - Protocols ⇒ operation in presence of failures
  - Standardization key for LAN, WAN
- Integrated circuit ("Moore's Law") revolutionizing network switches as well as processors
  - Switch just a specialized computer
- Trend from shared to switched networks to get faster links and scalable bandwidth
- Interested?
  - EE122 (CS-based in Fall, EE –based in Spring)

[Bonus] Example: Network Media

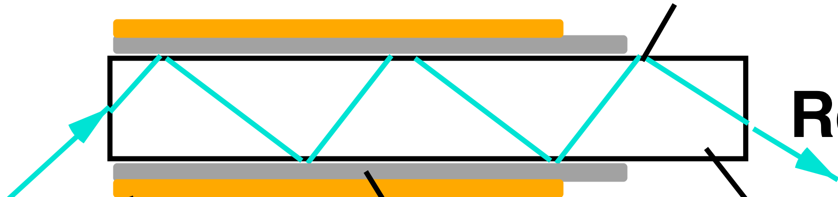
Twisted Pair ("Cat 5"):



Copper, 1mm think, twisted to avoid antenna effect

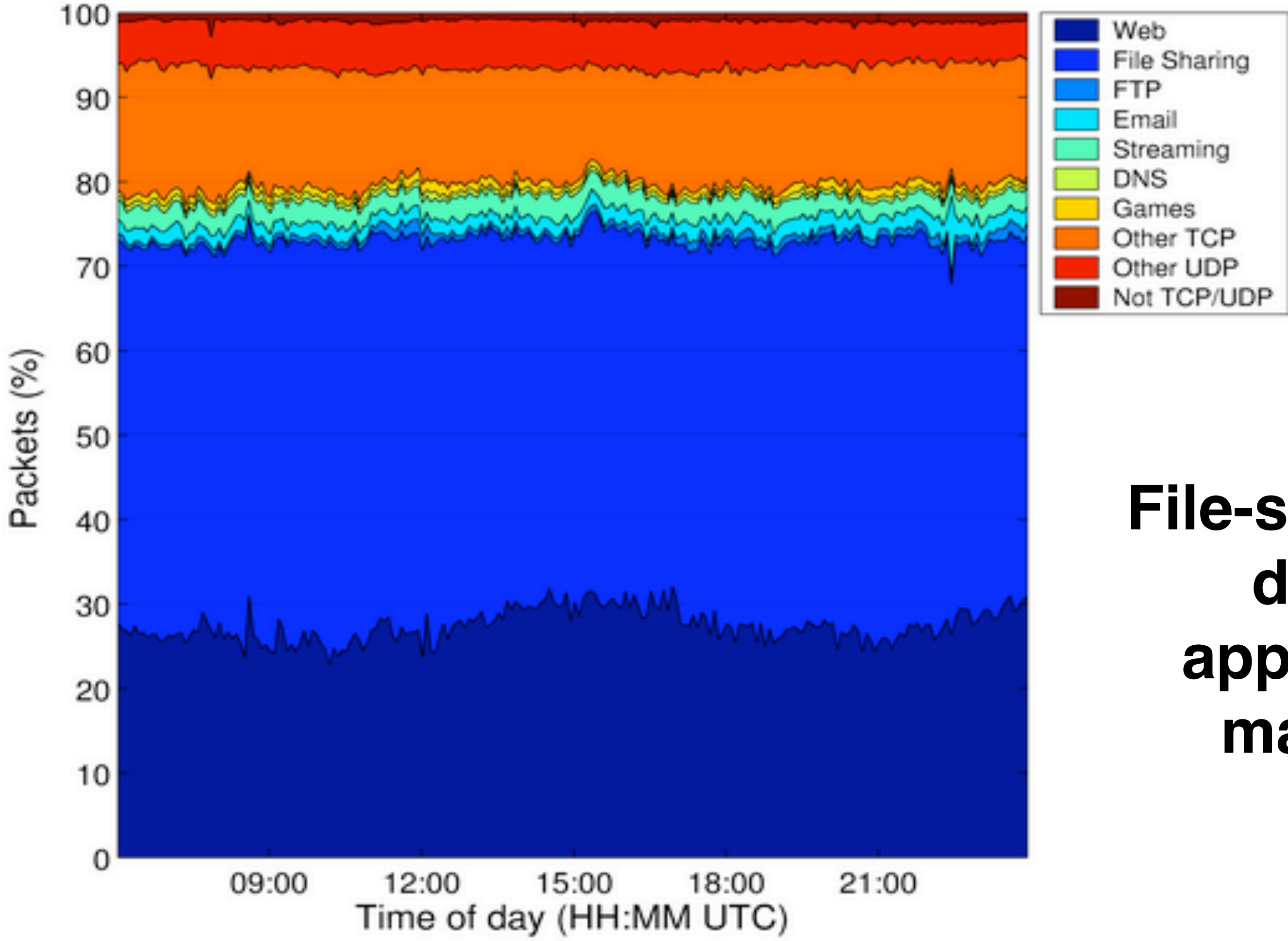
Light: 3 parts are cable, light source, light detector

Fiber Optics



Silica: glass or plastic; actually < 1/10 diameter of copper

[Bonus] Backbone Link App Composition



File-sharing is the dominant application on many links!