

CSE 123b

Communications Software

Spring 2003

Lecture 1: Introduction & Review

Stefan Savage
savage@cs.ucsd.edu

Class Overview

- Course Material
 - ♦ Class lectures, textbook readings, and handouts
- Course Assignments
 - ♦ Homework questions from book and handouts
 - » Handed out on Tuesdays due the following Tuesday
 - » Roughly every 2-3 weeks
 - ♦ Network protocol programming projects (3-5)
 - » We will implement routing protocols, transport protocols, etc.
- Exams
 - ♦ Midterm and Final
 - ♦ I will be explicit about what is covered in each

Grading

- Homework 25%
- Projects 25%
- Midterm 20%
- Final 30%

- Extra credit for class participation

Some hints

- Come to lecture
 - ◆ Yes, I will distribute the slides online, and yes the material is in the book
 - ◆ However, lecture materials are the basis for exams
- Do the homework
 - ◆ You will have a hard time with the exams without doing the homework
 - ◆ Its 25% of your grade (easily the difference between an A and C)

Some hints (2)

- Ask questions
 - ◆ In class, via e-mail and at office hours
 - ◆ Inevitably you won't understand something... that's my fault, but you need to help
- Start assignments early
 - ◆ There is a statistical relationship between when you start and what grade you get
- Sleep

Administrativa

- Web page
 - <http://www-cse.ucsd.edu/classes/sp03/cse123B/>
(will be up shortly)
- Textbook (required)
 - *Computer Networks: A Systems Approach* (2nd ed) by Peterson and Davie
- TA's
 - Yuchung Cheng, Cristian Estan and Alvin Auyoung
- Mailing list, office hours, discussion section (TBA)

Common questions

- Can I take this class concurrently with X? (where X is typically 120)
 - ♦ Yes, but this may be challenging. We assume basic knowledge about OS structure and some of the issues that are discussed in 120 that related to networking. Fair warning.
- How much programming is there?
 - ♦ The projects will require that you can understand and write code in C. If you're a proficient programmer and don't know C, you should be able to pick it up quickly. If you've done almost no programming, then this class may be painful.

Course material

- The key aspects of modern computer networks and network services
 - ◆ Reliable communication
 - ◆ Congestion control
 - ◆ Routing (intradomain and interdomain)
 - ◆ Naming
 - ◆ Mobility
 - ◆ Web service, caching, load balancing, CDNs
 - ◆ E-mail
 - ◆ Peer-to-peer networks
 - ◆ Security

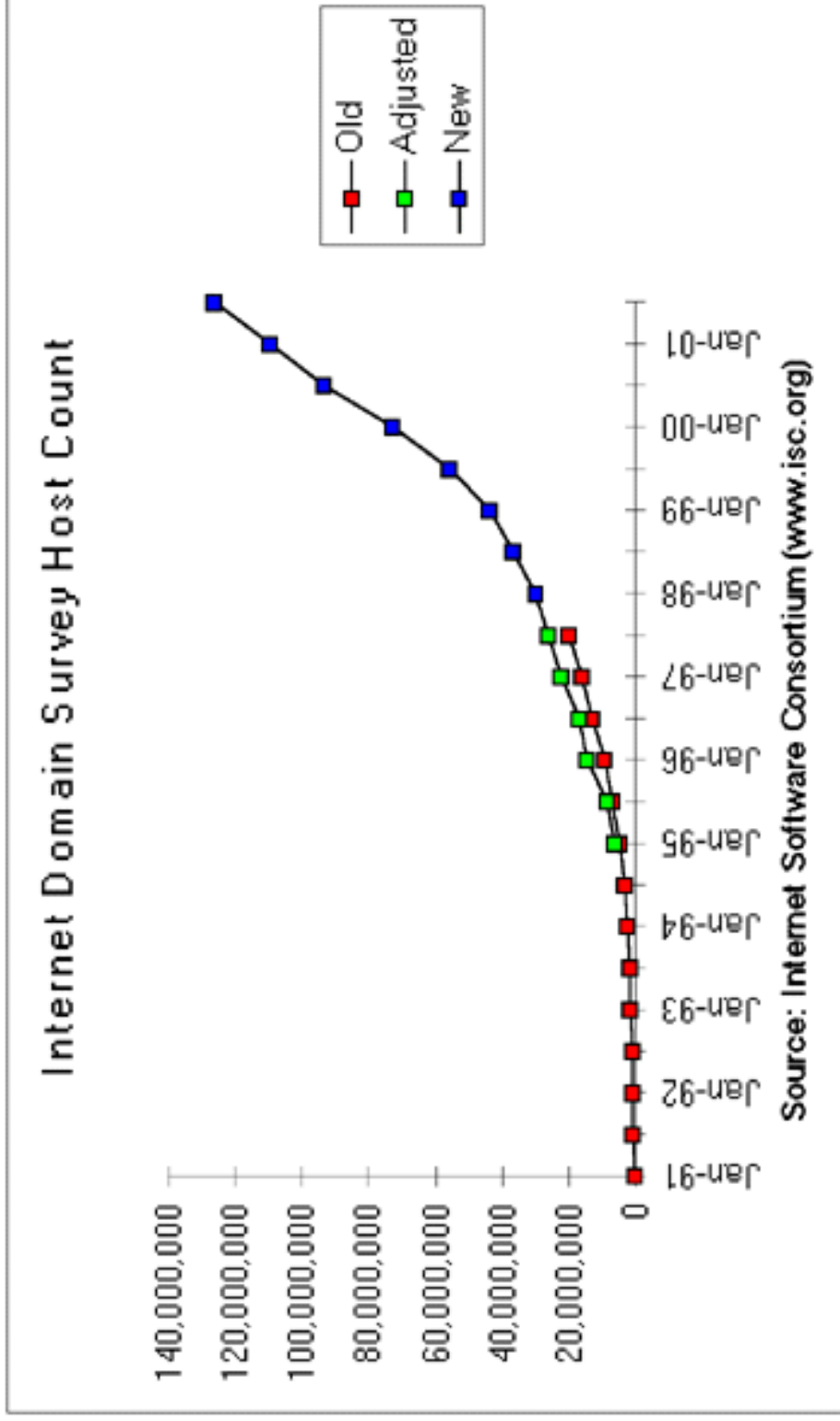
We will not cover

- Queuing theory
- Signals
- Hardware design
- Switching design
- Physical/data link layers

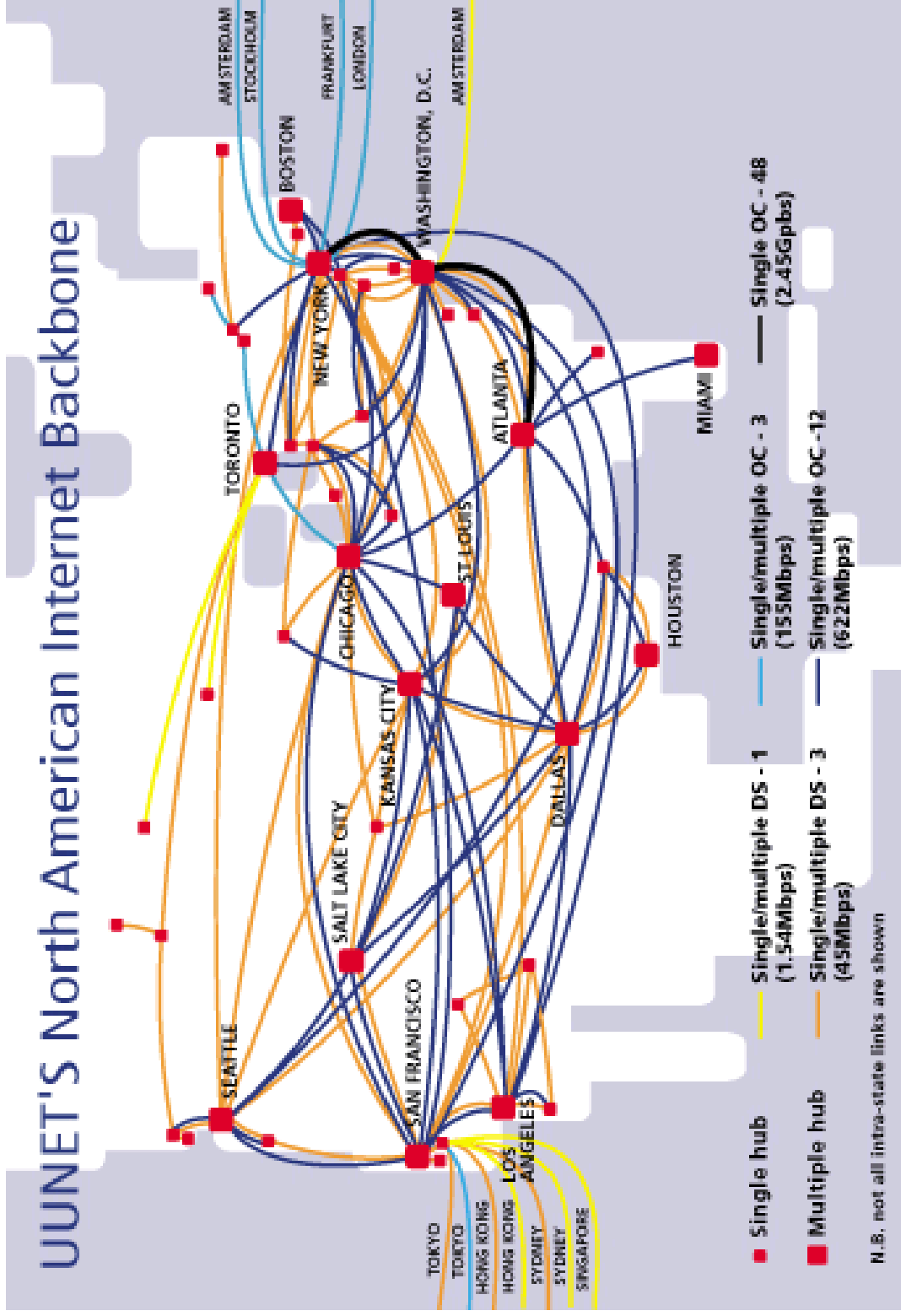
Overall goals

- Understand how to large scale, heterogeneous distributed networks are built
 - ◆ Fundamental problems
 - ◆ Established design principles
 - ◆ Standard Internet protocols and implementations

Large scale?



Large scale? (2)



Heterogeneous?

- Homogenous network: the telephone system
 - ◆ Designed for making phone calls
 - ◆ Known call duration distribution, bandwidth, service constraints, service model
- Heterogenous: the Internet
 - ◆ Supports E-mail, web, e-commerce, audio, video, multi-player games...
 - ◆ Few underlying assumptions – a strength and a weakness

Distributed?

- Decentralized components
 - ◆ Must update/manage changes in state
- Long communication latency
 - ◆ Actions take time
- Partial failures
 - ◆ Must tolerate failures

“A distributed system is a system in which I can’t do my work because some computer has filed that I’ve never even heard of”
– Leslie Lamport

Some review

- Elementary components
- Circuit switching vs packet switching
- Basic network model/metrics
- Layering/protocols
 - ◆ Layering by example: fetching a Web page

Network components

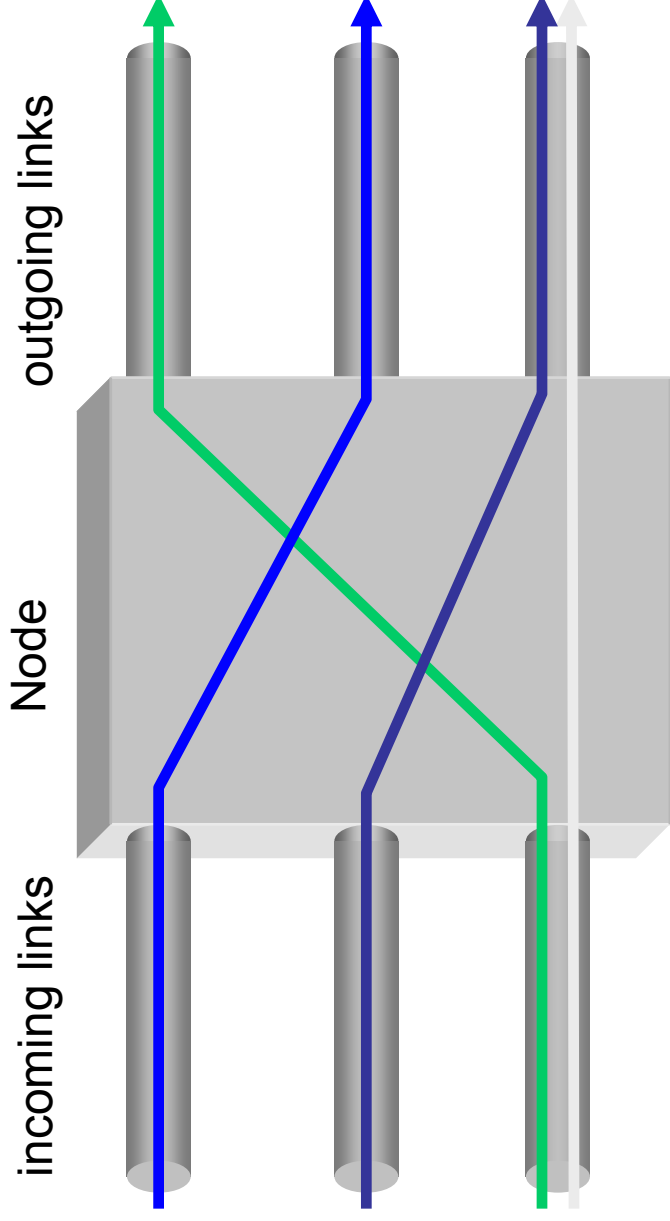
- **Hosts:** endpoints that communicate
 - ◆ e.g. workstation, server, PDA
- **Links:** transmission medium
 - ◆ e.g. Ethernet, 802.11b, FDDI
- **Routers/Switches:** moves bits between links
 - ◆ Circuit switching: guaranteed channel for a session (Telephone system)
 - ◆ Packet switching: statistical multiplexing of independent pieces of data (Internet)

Circuit Switching

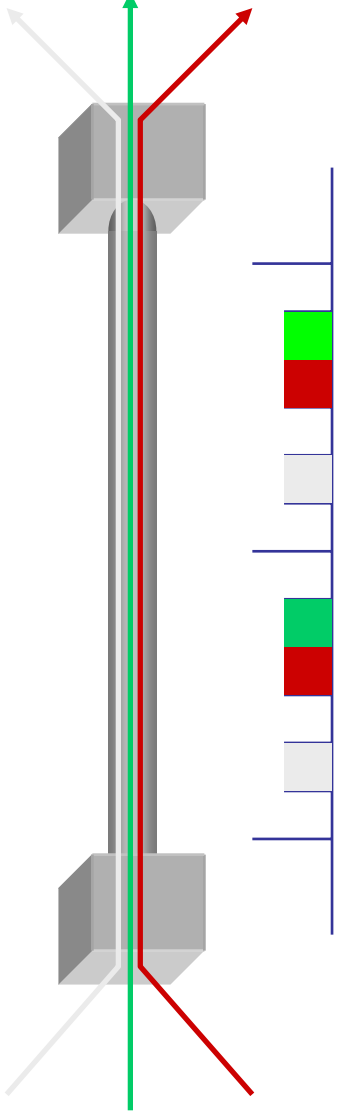
- Three phases
 1. circuit establishment (dial)
 2. data transfer (talk)
 3. circuit termination (hang up)
- If circuit not available: “Busy signal”
- Examples
 - ◆ Telephone networks
 - ◆ ISDN (Integrated Services Digital Networks)

Circuit Switching

- A node (switch) in a circuit switching network



Circuit switching: time division multiplexing



- Time divided in frames and frames divided in slots
 - ♦ Relative slot position inside a frame determines which conversation the data belongs to
 - ♦ Needs synchronization between sender and receiver
- In case of non-permanent conversations
 - ♦ Need to dynamically bind a slot to a conversation
 - ♦ Signaling protocol

Packet Switching

- Data is sent in a bundle of bit-sequences, called a packet.
- Packets have the following structure:

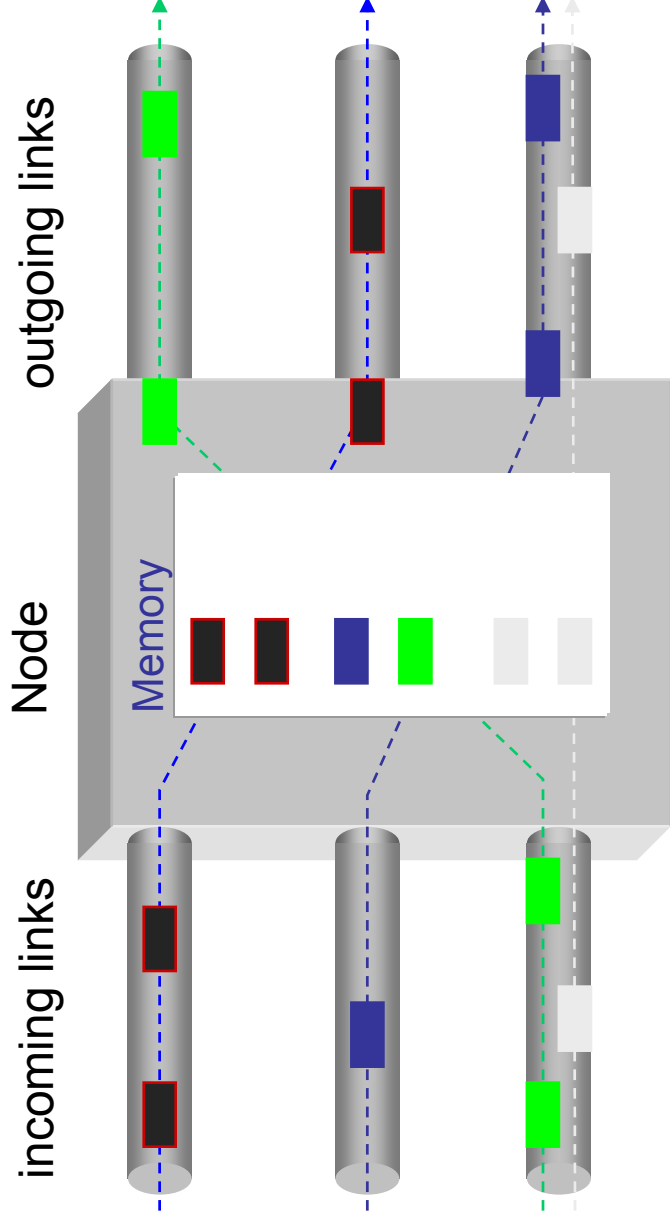


- » Header and Trailer carry control information (e.g., destination address, check sum)
- Each packet is passed through the network from node to node along some path (**Routing**)
- At each node the entire packet is received, stored briefly, and then forwarded to the next node (**Store-and-Forward Networks**)
- Typically no capacity is pre-allocated for packets

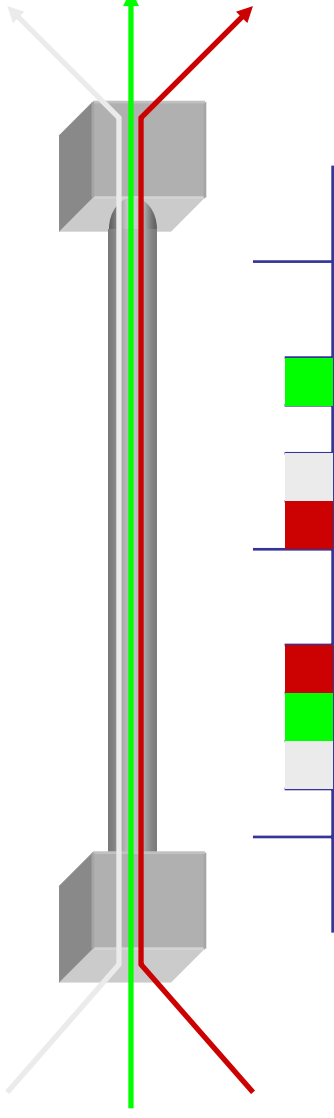
Slide courtesy Ion Stoica

Packet Switching

- A node in a packet switching network



Packet Switching: Statistical multiplexing



- Data from any conversation can be transmitted at any given time
- How to tell them apart?
 - ◆ use **header** to describe data

Pro/cons of packet switching

- Efficiency
 - ◆ Can share network up to its capacity – no overhead for reserving bandwidth that is unused
 - ◆ Can support many different service types
- Low complexity
 - ◆ Don't need to maintain state about each “call”
- Harder to guarantee bandwidth/delay

We will focus on packet switching in this class

Simple network model

Network is a pipe connection two computers



Basic Metrics

- ◆ Bandwidth, delay, overhead, error rate and message size

Network metrics

- Bandwidth
 - ◆ Data transmitted at a rate of R bits/sec
- Delay or Latency
 - ◆ Takes D seconds for bit to propagate down wire
- Overhead
 - ◆ takes O secs for CPU to put message on wire
- Error rate
 - ◆ Probability P that message will not arrive intact
- Message size
 - ◆ Size M of data being transmitted

How long to send a message?

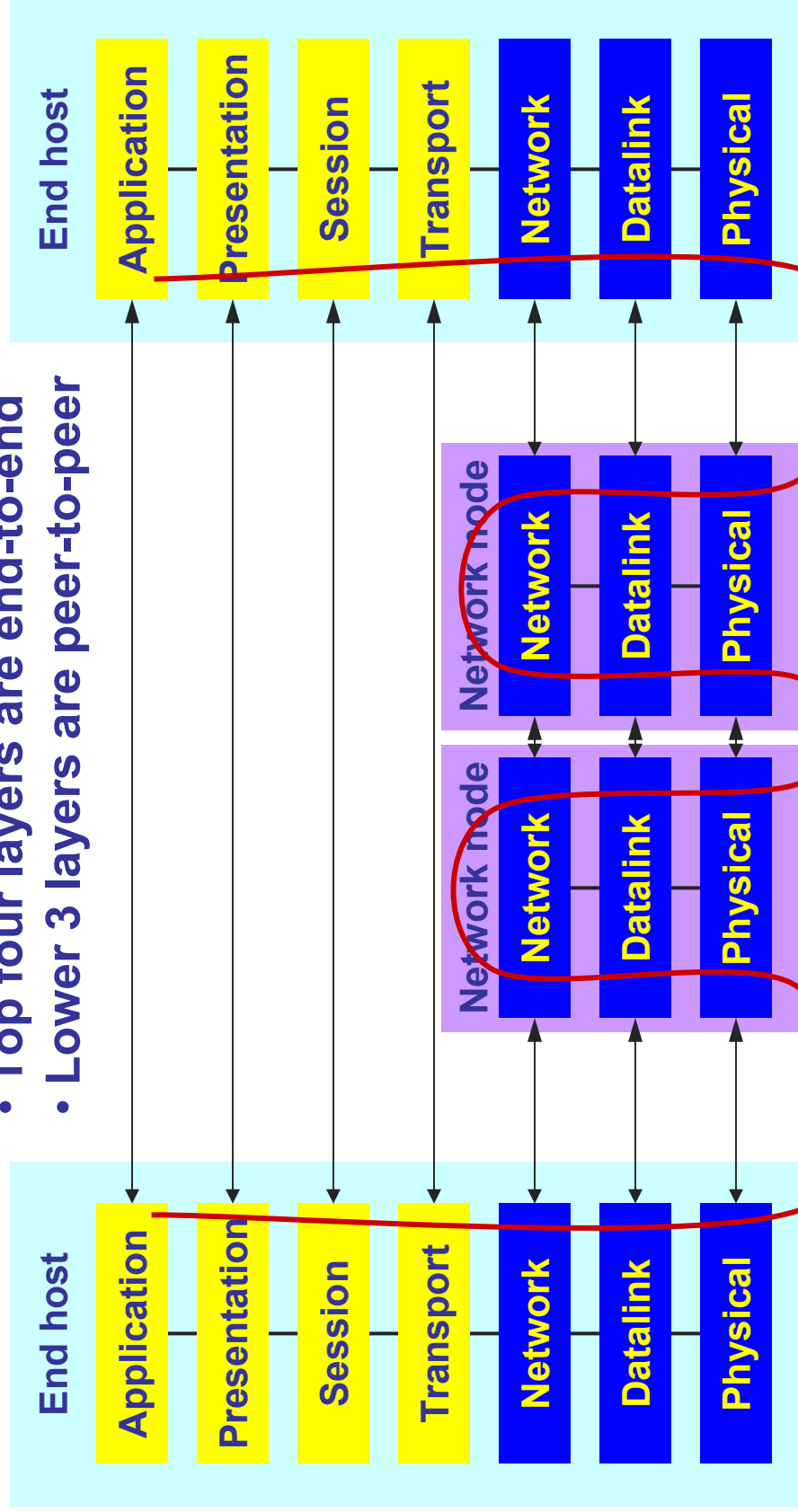
- Transmit time $T = M/R + D$
 - ♦ 10Mbps Ethernet LAN ($M=1\text{KBbyte}$, or 8000bits)
 - » $M/R=0.8\text{ms}$, $D \sim 5\mu\text{s}$
 - ♦ 155Mbps cross country ATM ($M=1\text{KB}$)
 - » $M/R = \sim 51\mu\text{s}$, $D \sim 40\text{-}100\text{ms}$
- $R \cdot D$ is the “storage” of pipe
(also called *bandwidth delay product*)

Layering

- What is layering?
 - ♦ Decomposition of a complex system into an **ordered** series of distinct abstractions
 - ♦ The services provided by a layer depend **only** on the services provided by the previous, less abstract, layer
- Layering in networking
 - ♦ **Service**: what a layer **does** (e.g. message delivery)
 - ♦ **Interface**: how to **use** the service (e.g. packet format)
 - ♦ **Protocol**: how the service is **implemented** (e.g. TCP)
 - ♦ **Protocol stack**: collection of protocols implementing a series of layers (e.g. Ethernet/IP/TCP/Web)

The OSI layering Model

- Top four layers are end-to-end
- Lower 3 layers are peer-to-peer



What the layers are for?

- **Application:** any service (e.g. WWW, SMTP)
- **Presentation:** data format conversion (e.g. XDR)
- **Session:** connection management, synchronization (e.g. SMIL)
- **Transport:** error-control, flow-control, channel multiplexing (e.g. TCP, UDP)
- **Network:** Routing (e.g. IP)
- **Datalink:** Framing, media access (e.g. Ethernet, FDDI, SONET)
- **Physical:** Transmission/modulation (e.g. 100BaseT)

Benefits of layering

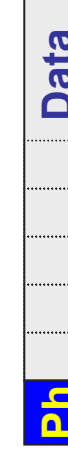
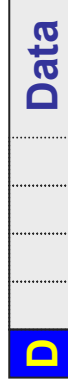
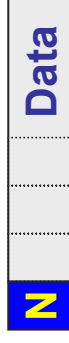
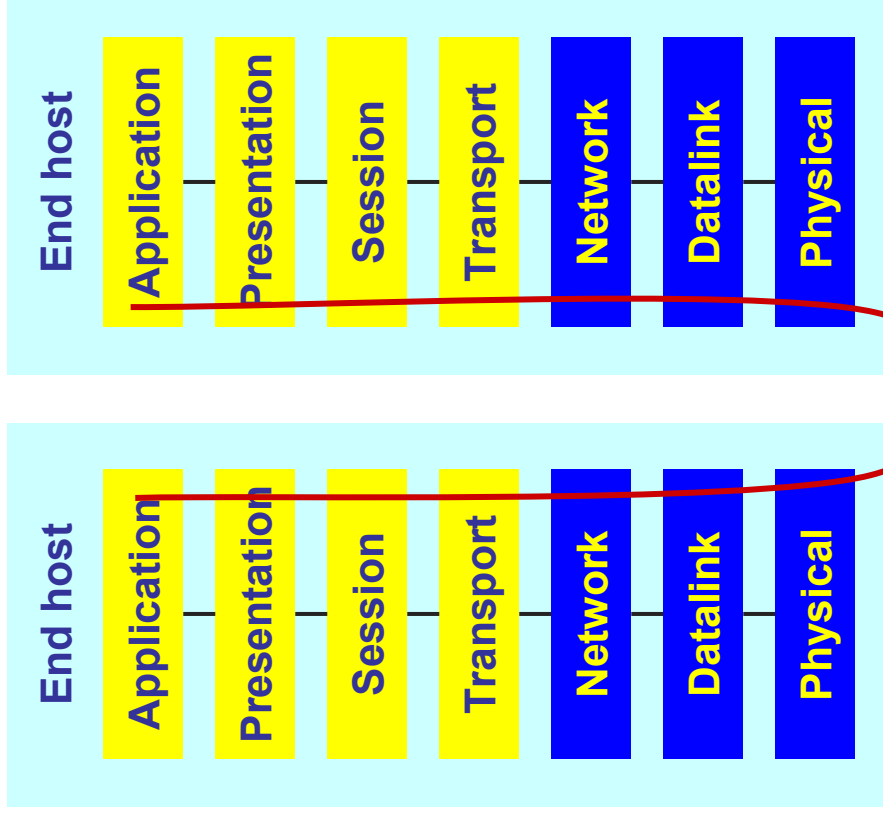
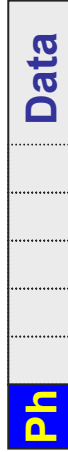
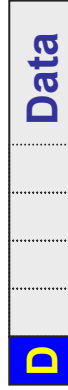
- **Encapsulation**
 - ◆ Functionality inside a layer is self-contained; one layer doesn't need to reason about other layers
- **Modularity**
 - ◆ Can replace a layer without impacting other layers
 - ◆ Lower layers can be reused by higher layers (e.g. TCP and UDP both are layered upon IP)
- One obvious drawback
 - ◆ Information hiding can produce **inefficient implementations**

Layer encapsulation

Layer N+1 packet

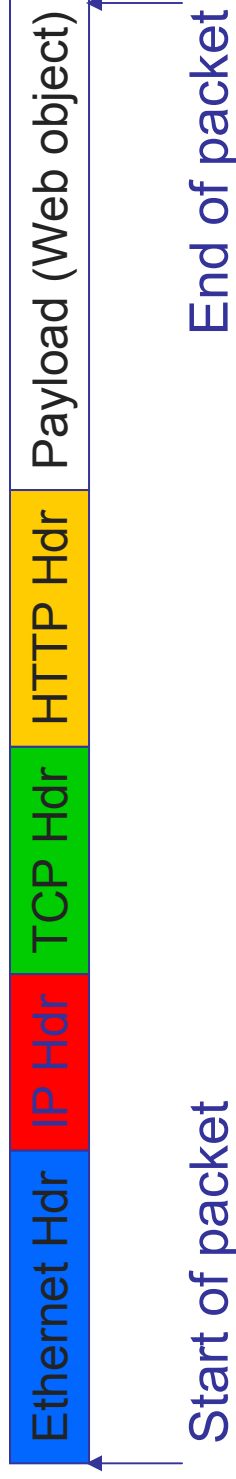
becomes

Layer N data



Layer Encapsulation (2)

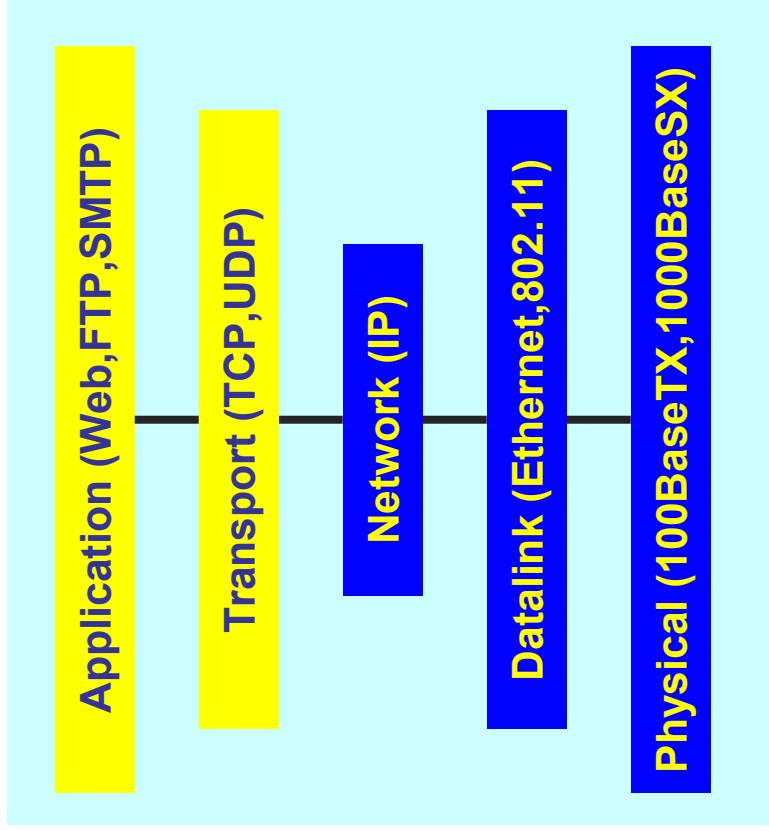
- Typical Web packet



- Notice that layers add overhead
 - ◆ Space (headers), effective bandwidth
 - ◆ Time (processing headers, peeling the onion), latency

The Internet layering model

- So-called “hourglass” model
 - One network layer protocol
 - Significant diversity at other layers
- No presentation or session layers
- Implementations more important than interfaces



Layering by example...

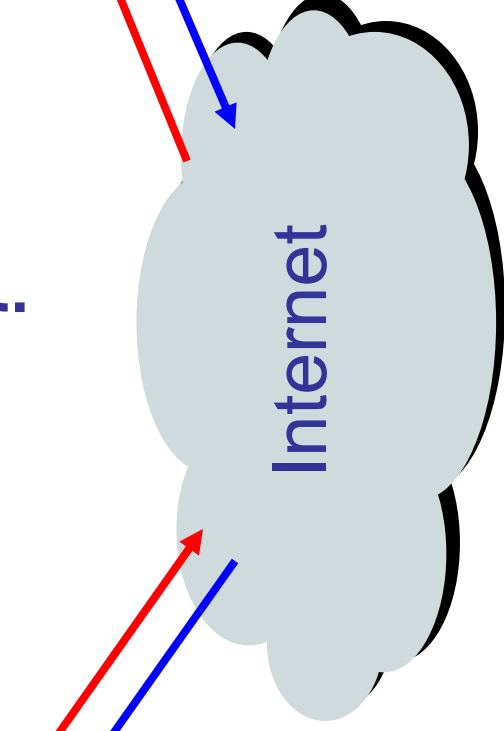
- **ROUGHLY**, what happens when I click on a Web page from UCSD?

My computer



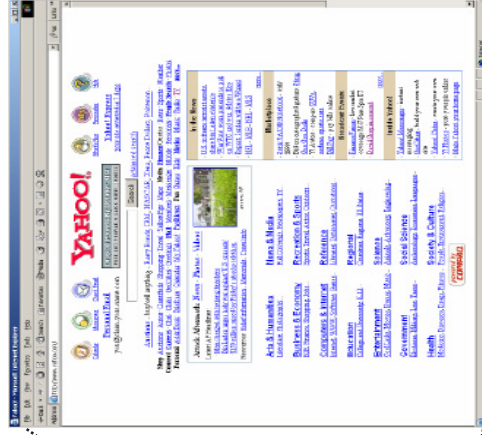
?

www.yahoo.com



Application layer (HTTP)

- Turn click into HTTP request



GET <http://www.yahoo.com/r/mp> **HTTP/1.1**
Host: www.yahoo.com
Connection:keep-alive
...



Application layer?

Name resolution (DNS)

- Where is www.yahoo.com?

My computer
(132.239.9.64)



What's the address for www.yahoo.com



Oh, you can find it at 64.58.76.177



Local DNS server
(132.239.51.18)



Transport layer (TCP)

- Break message into packets (TCP segments)
- Should be delivered reliably & in-order

```
GET http://www.yahoo.com/r/mp HTTP/1.1
Host: www.yahoo.com
Connection:keep-alive
...
```



3 yahoo.c 2 p://www. 1 GET htt

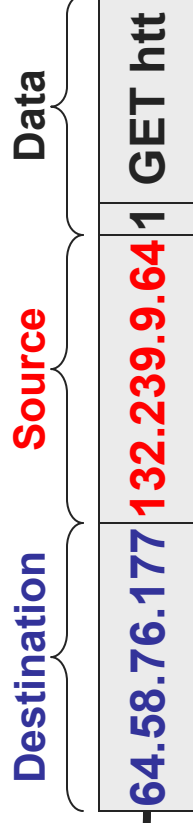
“and let me know when they got there”



Network layer: IP Addressing

- Address each packet so it can traverse network and arrive at host

My computer
(132.239.9.64)

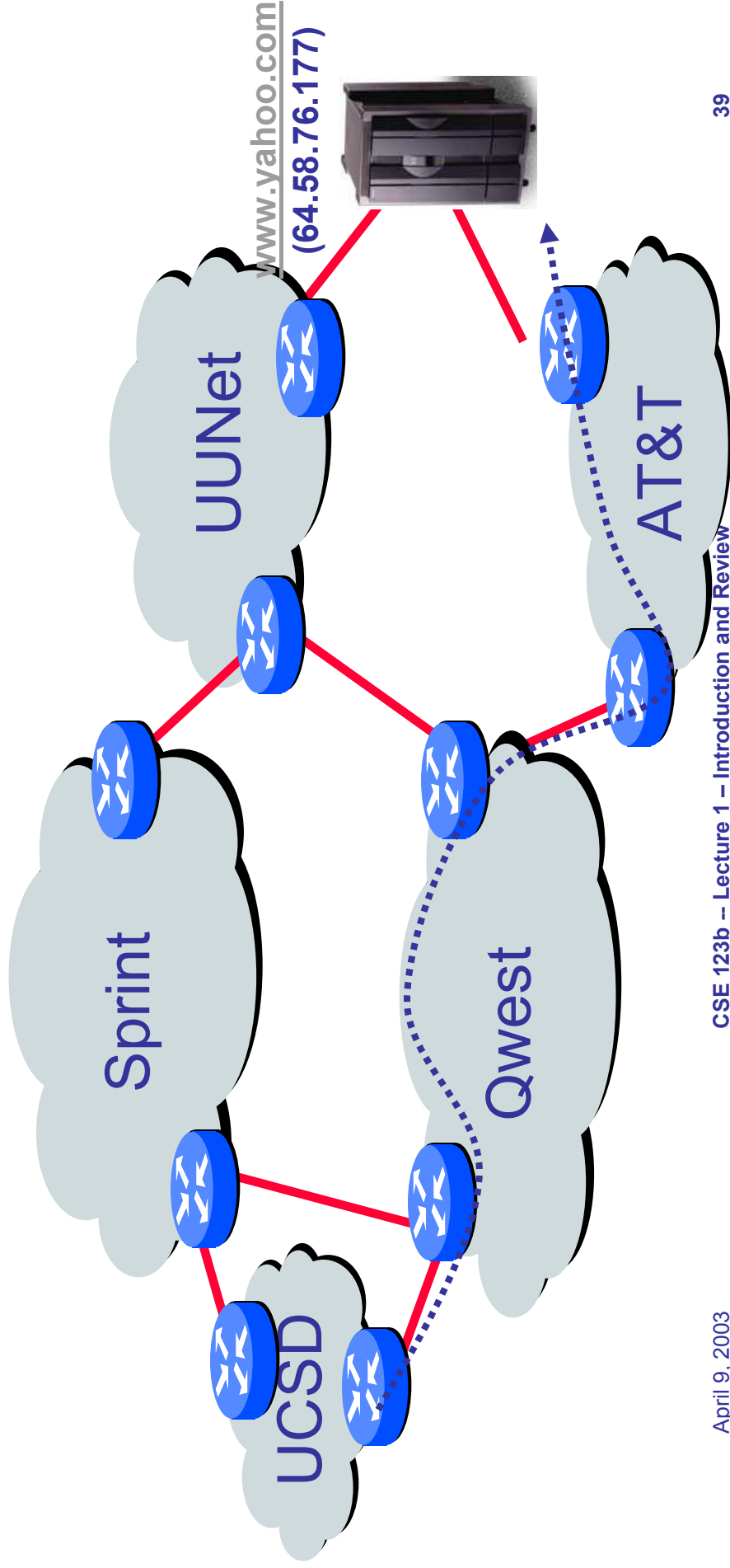


www.yahoo.com
(64.58.76.177)



Network layer: IP Routing

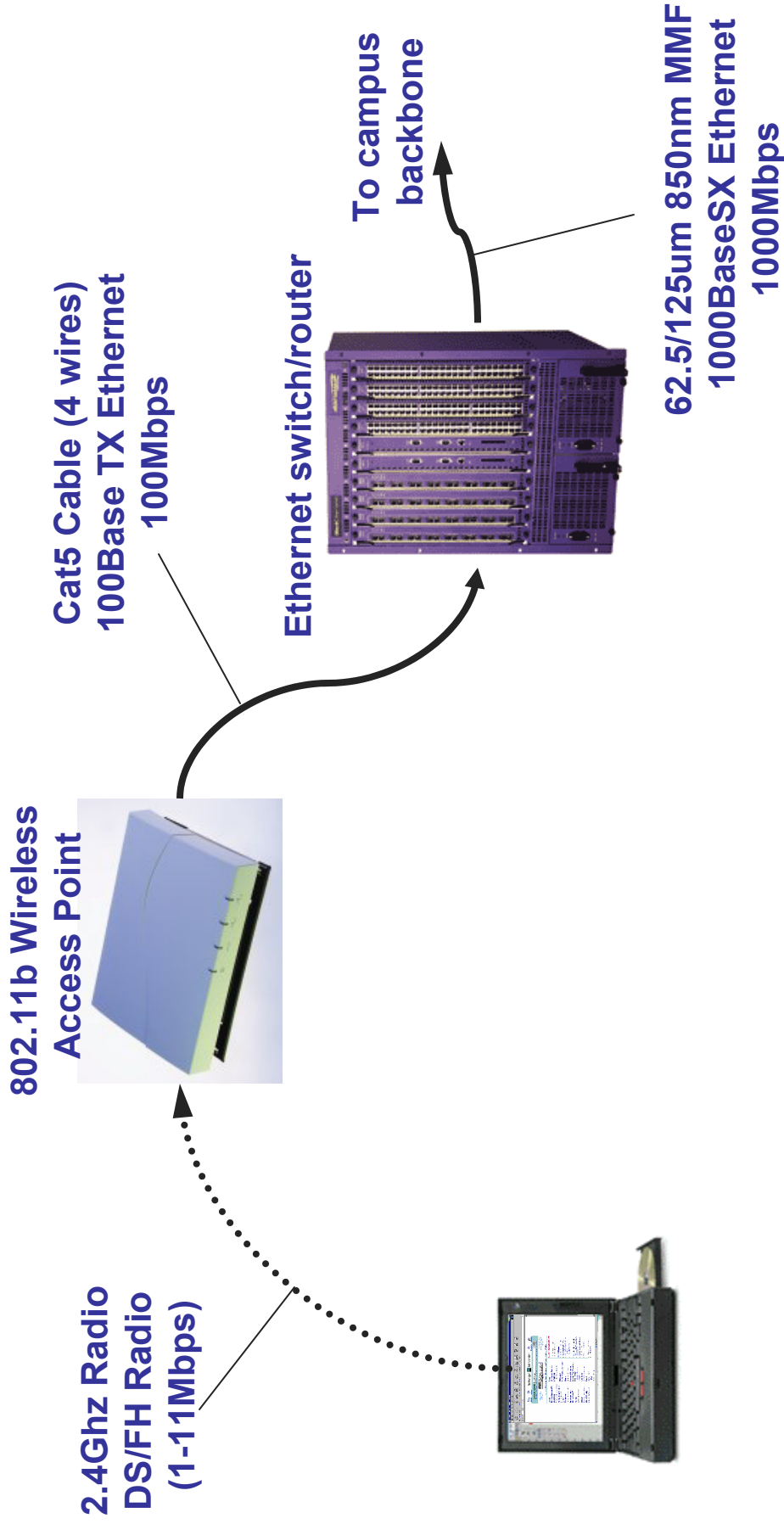
- Each router forwards packet towards destination



Datalink layer (Ethernet)

- Too boring for a picture (sorry)
- Break message into frames
- Media Access Control (MAC)
- Send frame

Physical layer



Summary

- Packets switching is an efficient and simple architecture for data communications
 - ◆ Gives up guarantees on service
- Layering is a technique for managing complexity in systems
 - ◆ Encapsulate related functionality in a layer and provide an interface to upper and lower layers
 - ◆ A **model**: implementations do not necessarily respect layers

For Next Time...

- **ATTENTION – Wake up!**
 - ♦ **Thursday’s class is cancelled**
 - ♦ **The next class will be Tuesday April 8th**
- **For then:**
 - ♦ **Get the textbook**
 - ♦ **Review Patterson&Davie Chap1**
 - ♦ **Read Chap 4.1 - 4.1.4**