

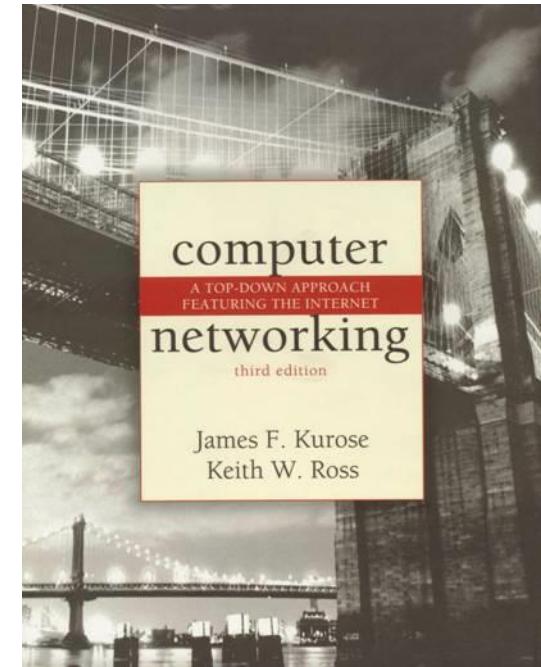
Computer Networks

SE3CN11

Module Overview

Professor R Simon Sherratt

Adapted and updated from the original work
of Dr. Michael Evans



*Computer
Networking: A Top
Down Approach*

Jim Kurose, Keith
Ross
Addison-Wesley

A MODULE OF TWO HALVES

Autumn term

- Lecture based, no coursework.
- How everything in the Internet, and networks in general, actually work

Spring term

- Group work
- How to put everything together to create a working network for a client

TEACHING MODES AND ASSESSMENT

Autumn Term

- Teaching: Weekly lectures, taught by me
- Coursework: none

Second half

- Teaching: none
- Group coursework

Exam

- One two-hour exam in Summer term

FIRST HALF IN DEPTH

“How everything in the Internet, and networks in general, actually works...”

Equipment

- Routers, switches, bridges, Network Interface Cards, modems, servers, clients

Protocols

- TCP, IP(v4 and v6), BGP, OSPF, RIP, ATM, DNS, DHCP, Ethernet, ARP, CIDR, NAT, BitTorrent, HTTP...

Applications

- The Web, YouTube, Email, FTP, Skype, BitTorrent

Concepts

- Layers, services, architecture, protocols, multimedia networking, LANs, WANs, routing, addressing, congestion, flow control, error checking, connections, connectionless, QoS, internetworking, subnetting, protocol encapsulation, multiplexing, jitter, delay...

SECOND HALF

ALL Coursework

You:

- Are split into teams,
- Are given a network scenario
 - Contains current network
 - And requirements to improve it
- Create a proposal for new network that meets requirements
- Present your work, write-up your network proposal.

CARROT AND STICK

Carrot:

- Computer Networking is demanded skill with high salary and good job prospects



Stick:

- Module is worth 20 credits – that's 1/6th this year's credits.
- Very technical – lots of acronyms and protocols
- Some people always fail!
- You know group work can be challenging.



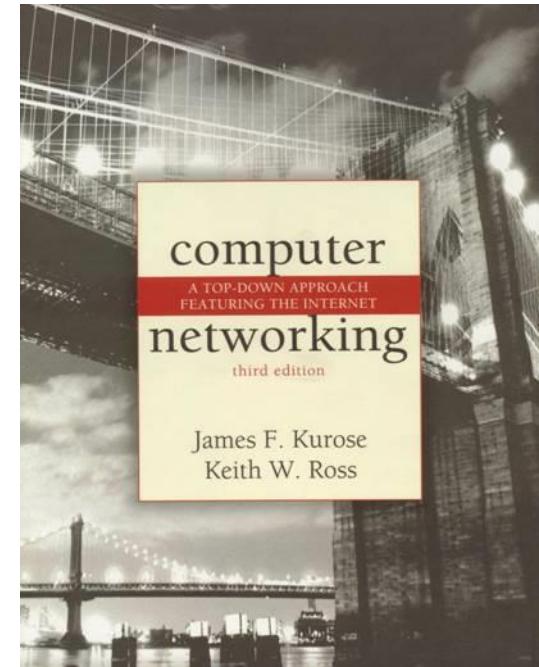
Computer Networks

SE3CN11

Chapter 1 Introduction

Professor R Simon Sherratt

Adapted and updated from the original work
of Dr. Michael Evans



*Computer
Networking: A Top
Down Approach
Featuring the
Internet,
Jim Kurose, Keith
Ross
Addison-Wesley*

Course Overview

- ❑ Chapter 1 – Introduction to the Internet
- ❑ Chapter 2 – The Application Layer
- ❑ Chapter 3 – The Transport Layer
- ❑ Chapter 4 – The Network Layer
- ❑ Chapter 5 – The Data Link Layer

Chapter 1: Introduction

Our goal:

- Get “feel” and terminology
- More depth and detail *later* in module
- Approach:
 - ❖ use Internet as example

Overview:

- What makes up the Internet?
- What is a protocol?
- Network edge
- Network core
- Access network
- Internet/ISP structure
- Protocol layers

Chapter 1: roadmap

1.1 What *is* the Internet?

1.2 Network Structure

- ❑ End systems, access networks, links
- ❑ Network Edge
- ❑ Access Network
- ❑ Network Core

1.3 Network Core in detail

- ❑ circuit switching, packet switching, network structure

1.4 Internet Structure and ISPs

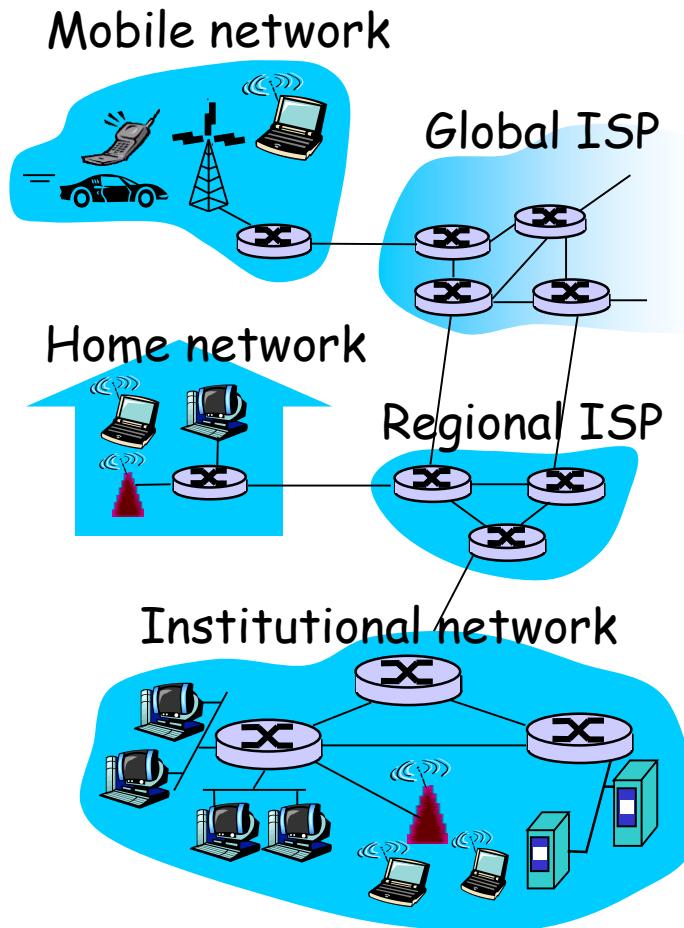
1.5 Protocol layers, service models

1.6 History

1.1 What *is* the Internet: a “nuts and bolts” view

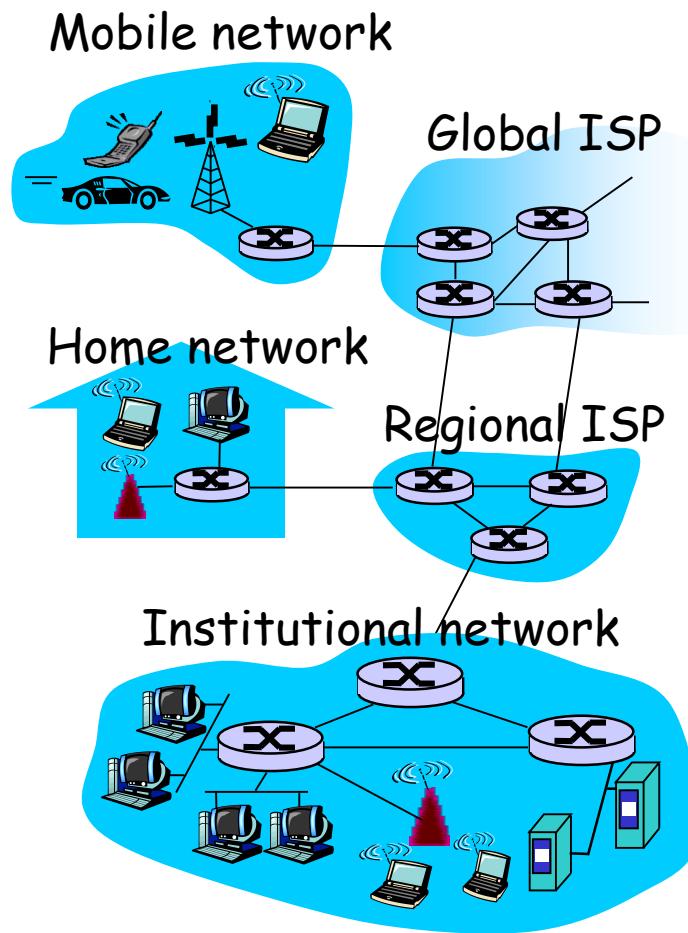


- Billions of connected computing devices: *hosts = end systems*
 - ❖ running *network apps*
- *communication links*
 - ❖ fiber, copper, radio, satellite
 - ❖ transmission rate = *bandwidth (bits/second)*
- *routers*: forward packets (chunks of data)



What is the Internet: a “nuts and bolts” view

- ❑ **Protocols** control the sending, receiving of messages
 - ❖ e.g., TCP, IP, HTTP, Skype, Ethernet
- ❑ **Internet: “network of networks”**
 - ❖ loosely hierarchical
 - ❖ public Internet versus private intranet
- ❑ Internet standards
 - ❖ RFC: Request For Comments
 - ❖ IETF: Internet Engineering Task Force



Protocols – the rules

Human protocols:

- “what’s the time?”
- “I have a question”
- introductions

... messages sent

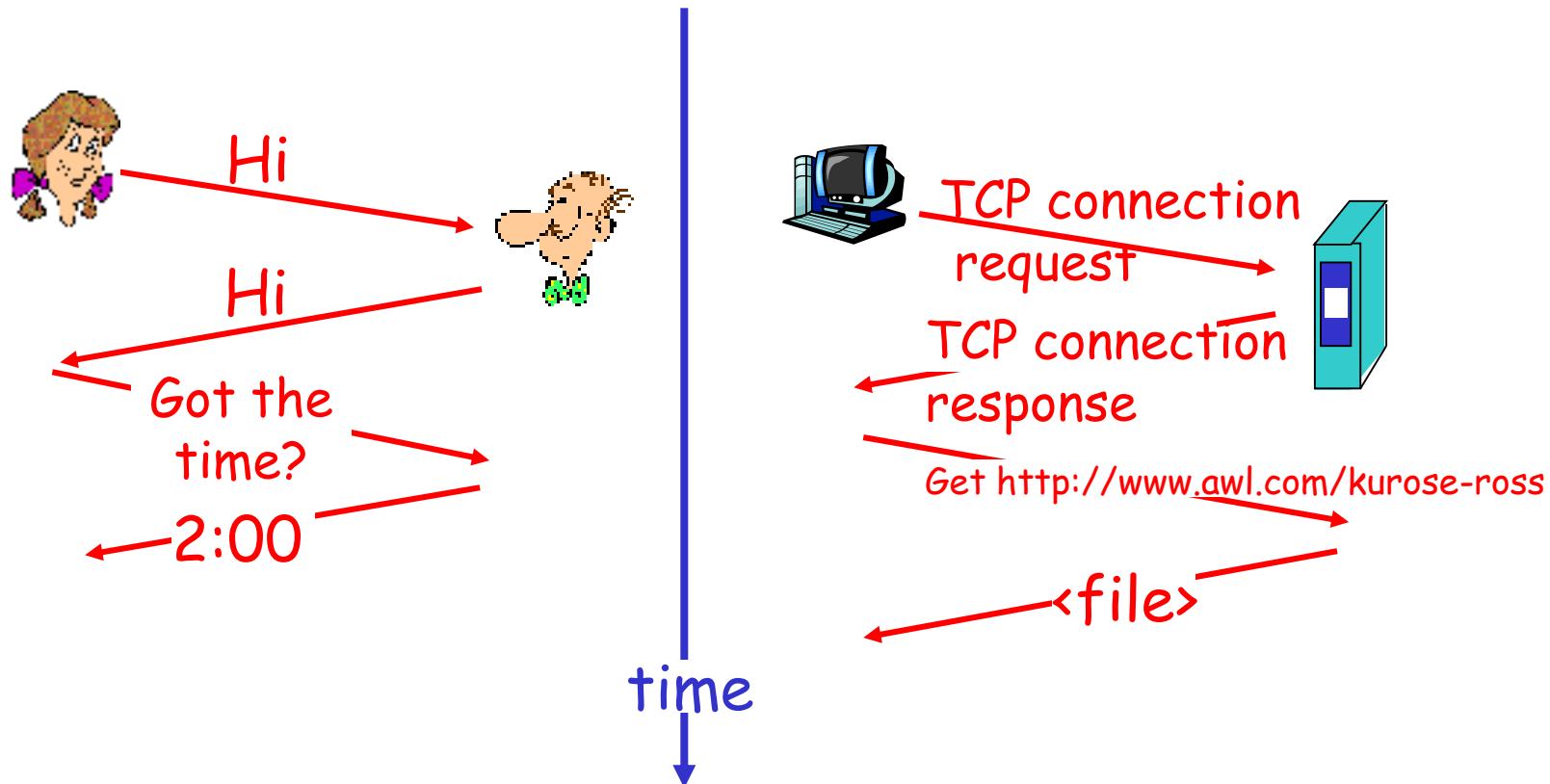
... actions taken based
on messages

Network protocols:

- Machines rather than
humans
- All** communication
activity in Internet
governed by protocols

***Protocols define format,
order of messages sent
and received among
network entities, and
actions taken on
transmission and receipt***

A human protocol and a computer network protocol



Chapter 1: roadmap

1.1 What *is* the Internet?

1.2 Network Structure

- ❑ End systems, access networks, links
- ❑ Network Edge
- ❑ Access Network
- ❑ Network Core

1.3 Network Core in detail

- ❑ circuit switching, packet switching, network structure

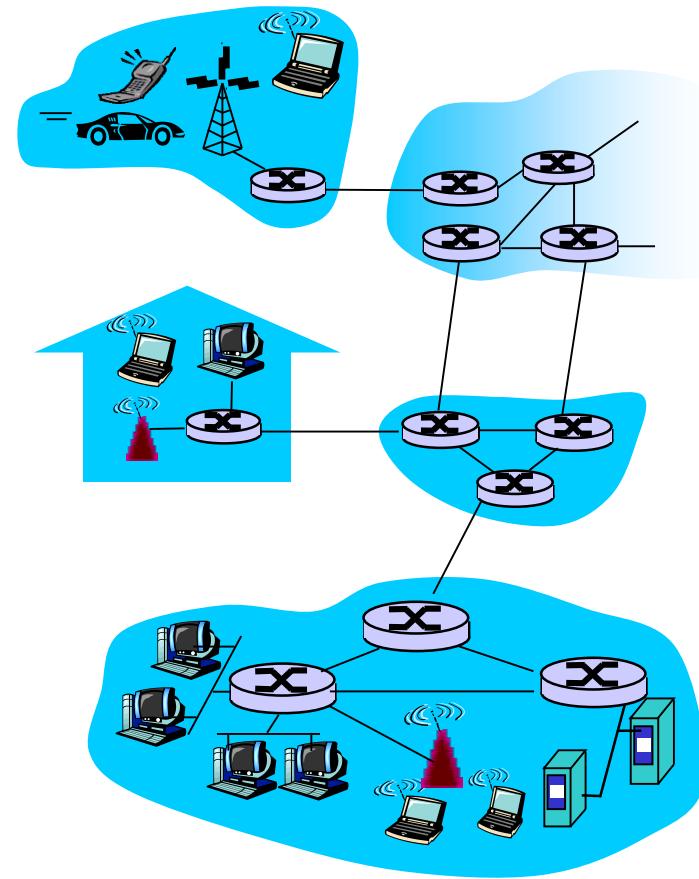
1.4 Internet Structure and ISPs

1.5 Protocol layers, service models

1.6 History

1.2 Network Structure

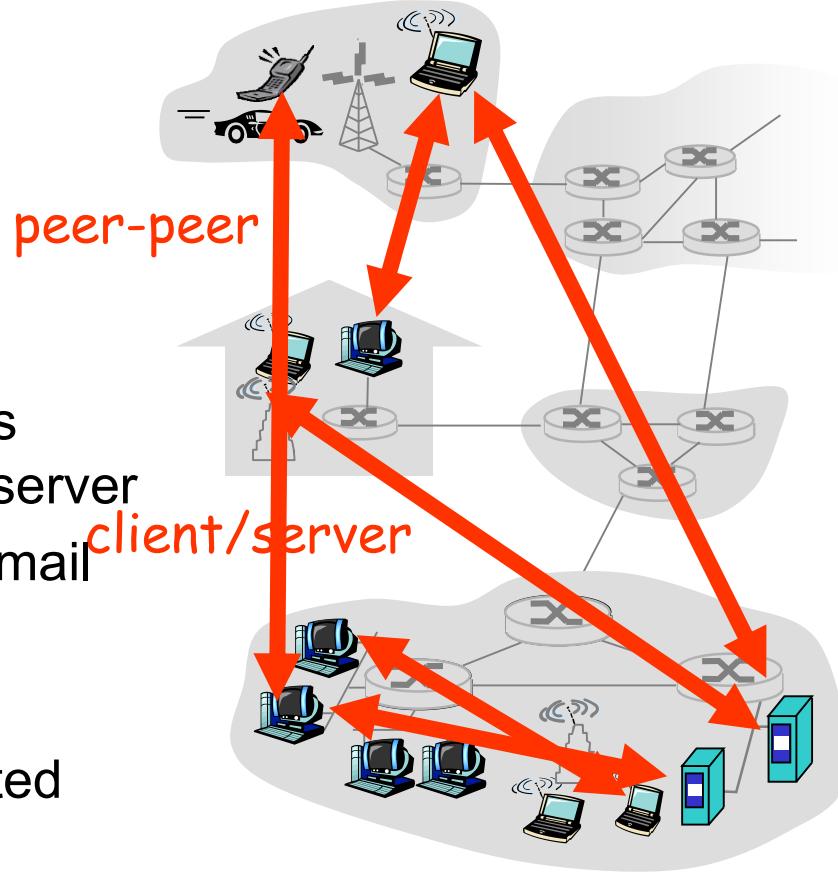
- **network edge:**
applications and hosts
- **access networks, physical media:**
wired, wireless communication links
- **network core:**
 - ❖ interconnected routers
 - ❖ network of networks



1.2.1 The Network Edge

□ end systems (hosts):

- ❖ run application programs
- ❖ e.g. Web, email
- ❖ at “edge of network”



□ client/server model

- ❖ client requests and receives service from an always-on server
- ❖ e.g. Web browser/server; email client/server

□ peer-peer model:

- ❖ has minimal (or no) dedicated servers
- ❖ e.g. Skype, BitTorrent

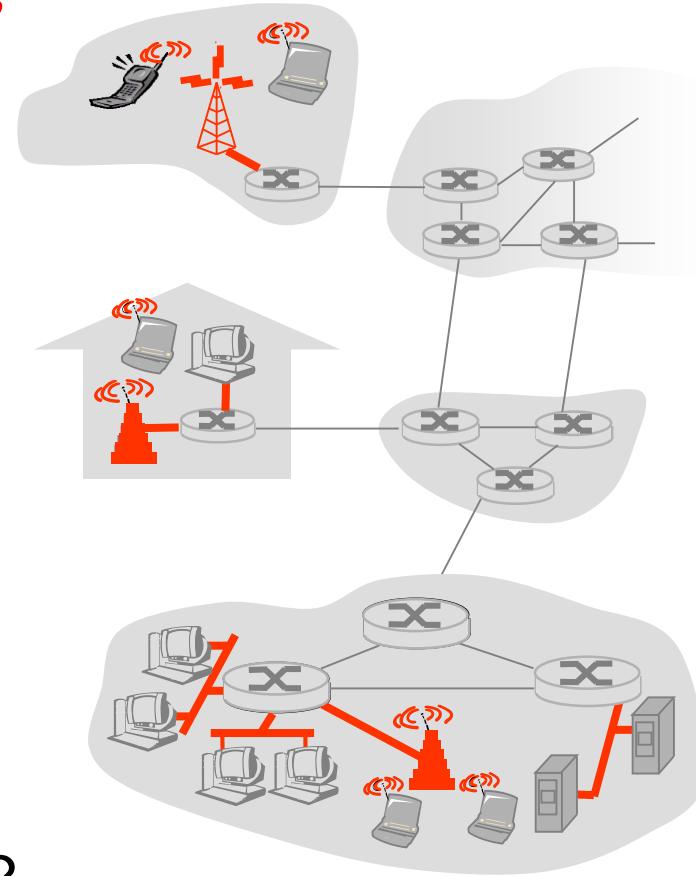
1.2.2 Access networks and physical media

Q: How to connect end systems to edge router?

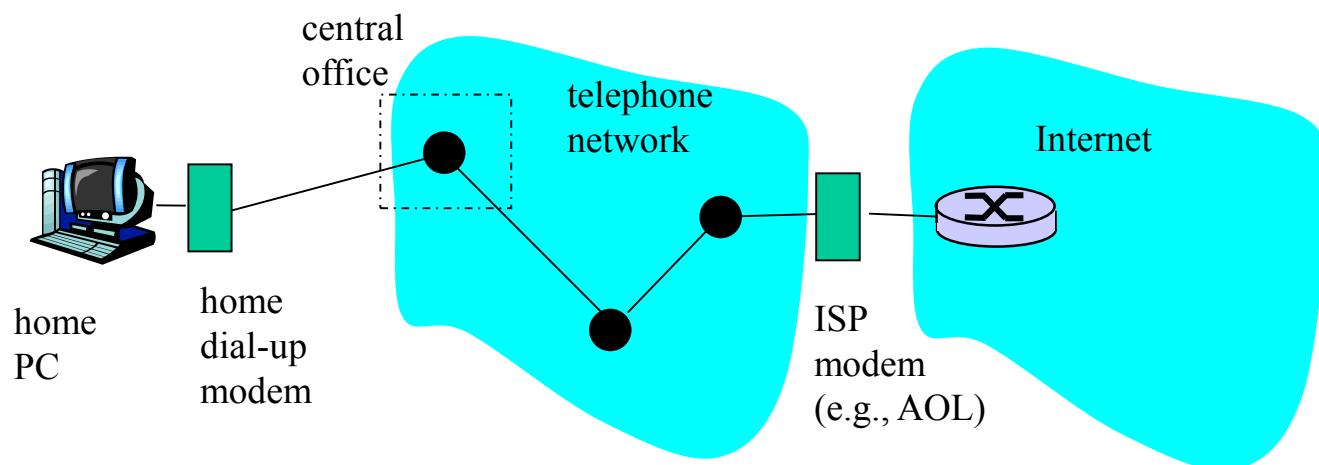
- residential access nets
- institutional access networks (school, company)
 - ❖ Intranet
- mobile access networks

Keep in mind:

- bandwidth (bits per second) of access network?
- shared medium or dedicated?

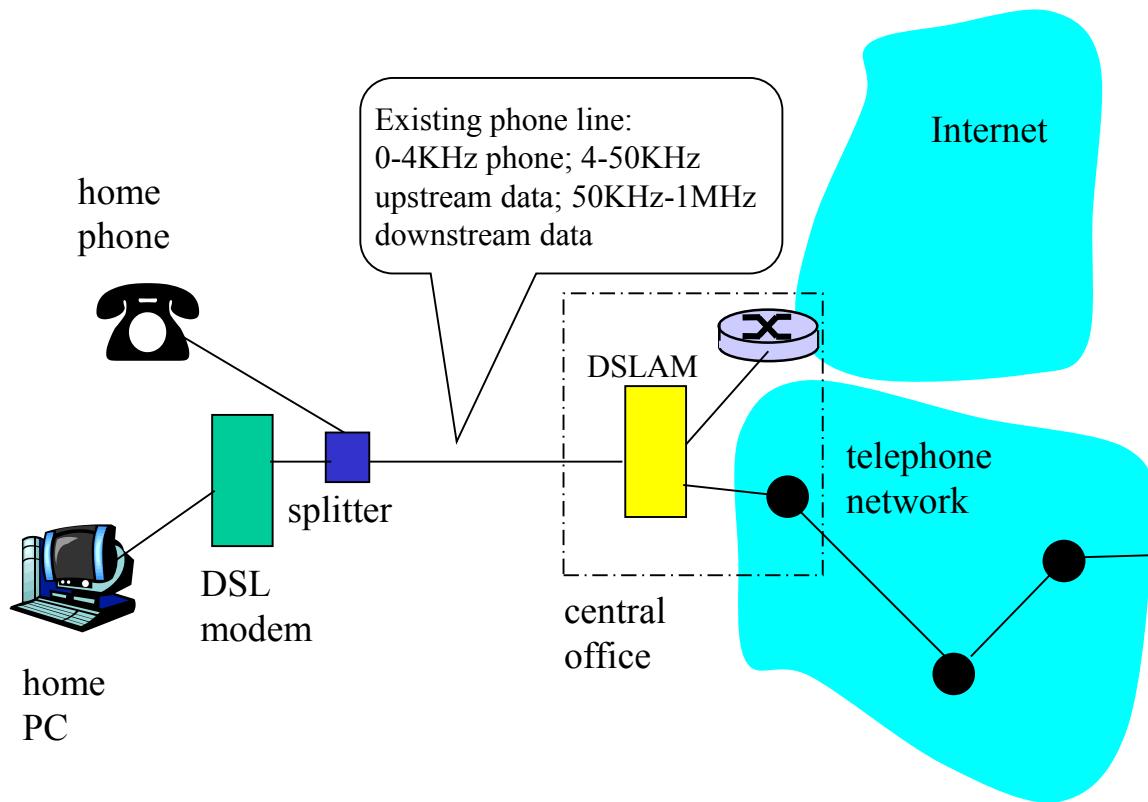


Old Fashioned Dial-up Modem



- ❖ Uses existing telephony infrastructure
 - ❖ Home is connected to **central office**
- ❖ up to 56Kbps direct access to router (often less)
- ❖ Can't surf and phone at same time: not "**always on**"

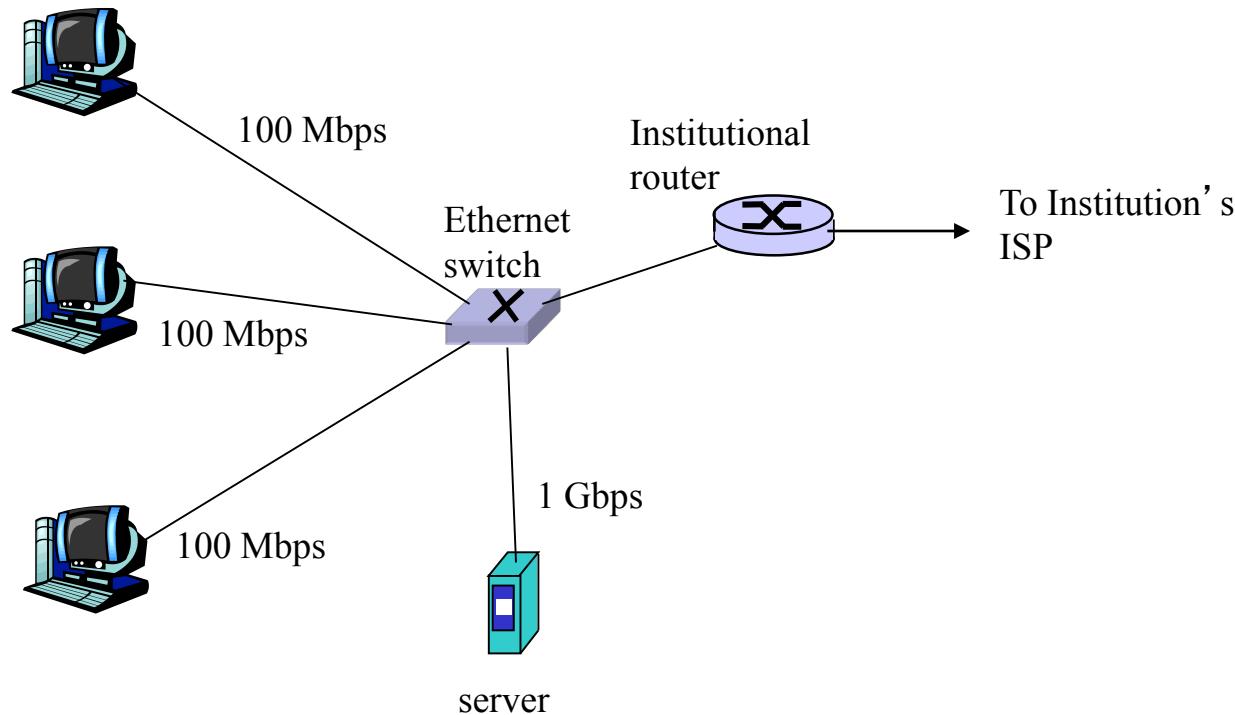
Digital Subscriber Line (DSL)



Q - Why do we have a faster downstream?

- ❖ Also uses existing telephone infrastructure
- ❖ Typically 3.3 Mbps upstream
- ❖ up to 24 Mbps downstream
- ❖ dedicated physical line to telephone central office

Ethernet Internet access



- Typically used in companies, offices, universities, etc
- 10 Mbs, 100Mbps, 1Gbps, even 10Gbps Ethernet
- Today, end systems typically connect into Ethernet switch
- Switch can isolate sub-networks

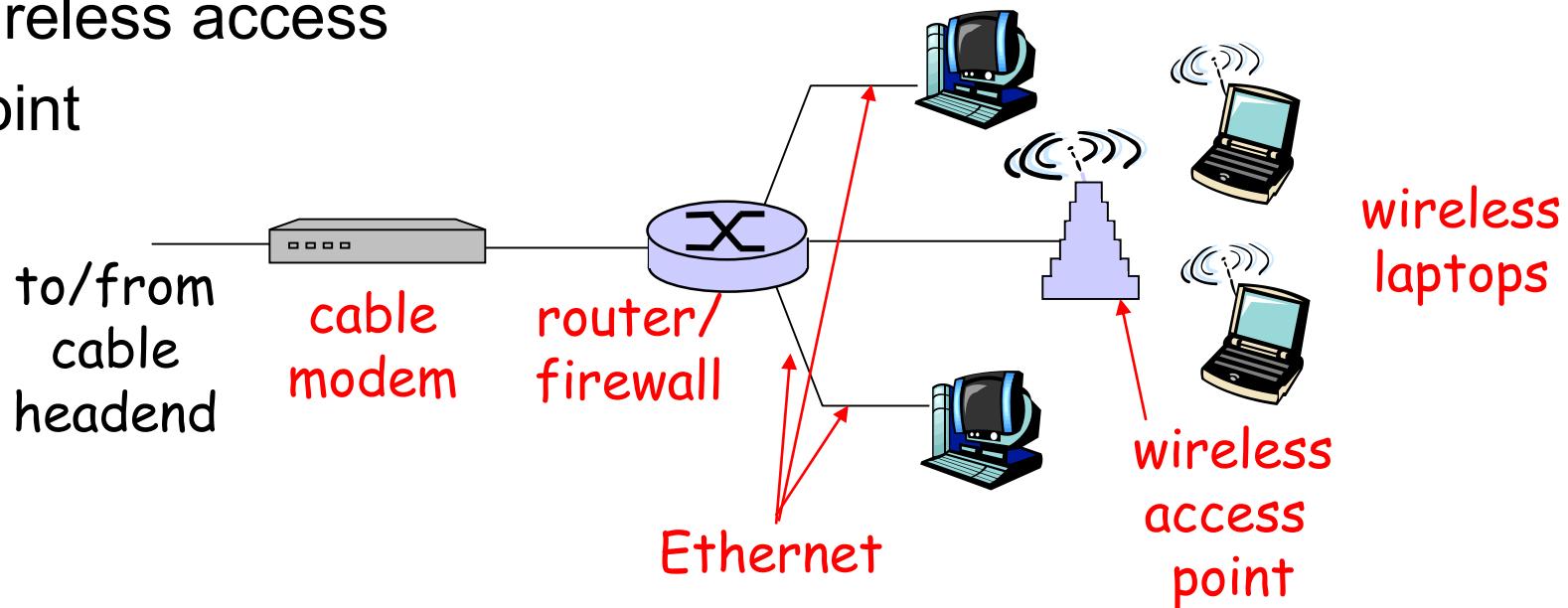
Home networks

Typical home network components:

- DSL or cable modem (DOCSIS)
- router/firewall/NAT
- Ethernet
- wireless access point

Q – given 2^{32} Internet addresses, how can we have more than one connected device per person?

Look up NAT and IPV6...



Chapter 1: roadmap

1.1 What *is* the Internet?

1.2 Network Structure

- ❑ End systems, access networks, links
- ❑ Network Edge
- ❑ Access Network
- ❑ Network Core

1.3 Network core

- ❑ circuit switching, packet switching, network structure

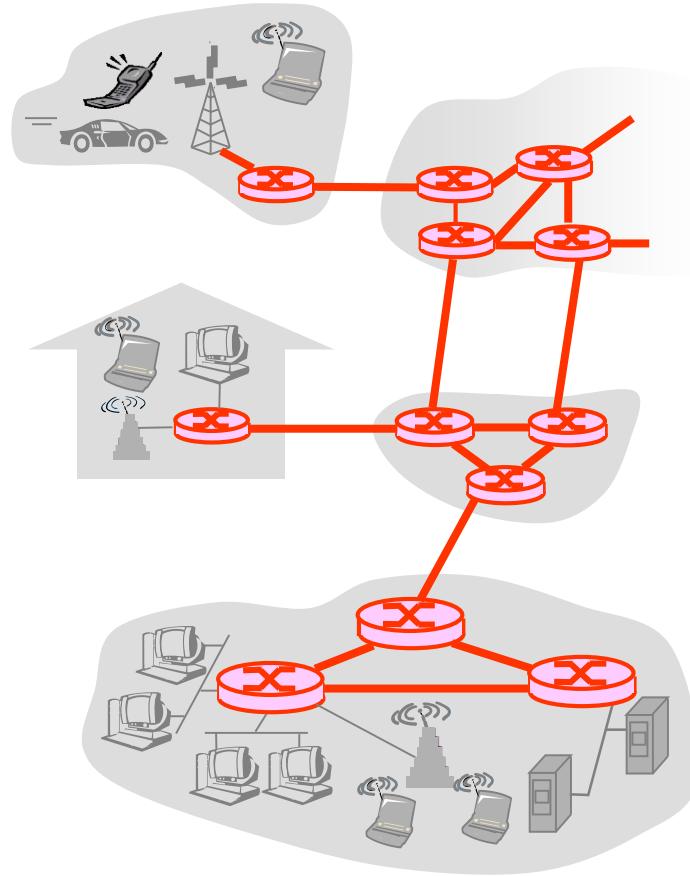
1.4 Internet Structure and ISPs

1.5 Protocol layers, service models

1.6 History

1.3 The Network Core

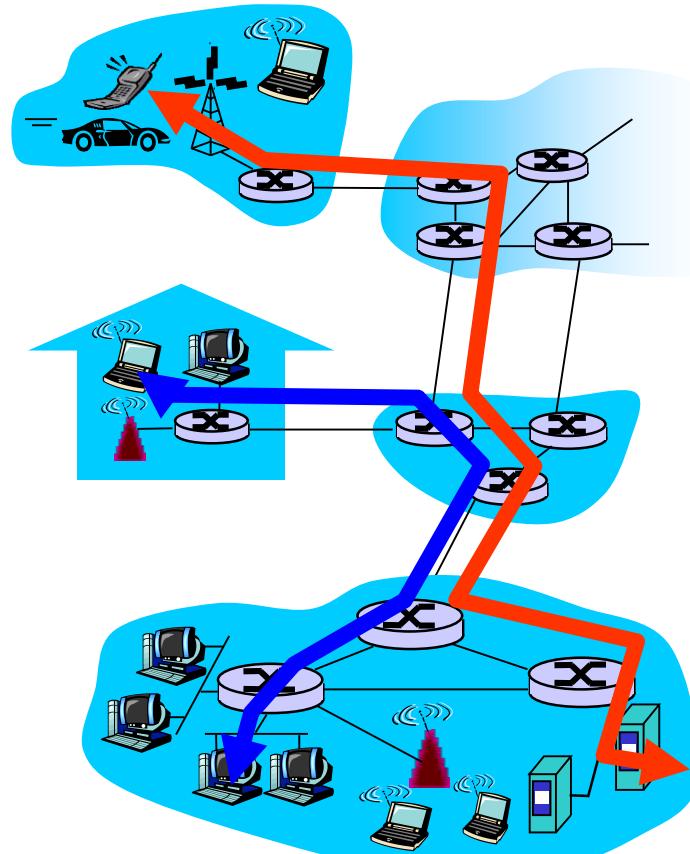
- Core is a Mesh of interconnected routers
- The fundamental question: how is data transferred through the network?
 - ❖ a). **circuit switching:** dedicated circuit per call: telephone net
 - ❖ b). **packet-switching:** data sent in discrete “chunks”



1.3.1 Circuit Switching

End-end resources reserved for “call”

- Resources: link bandwidth, switch capacity
- Dedicated resources: no sharing
- Circuit-like (guaranteed) performance
- Call setup required
- Can run out of links.
- Resource used even not communicating.



SE3CN11 - 1. Introduction

Circuit Switching (cont.)

- ❑ Links may be shared...
- ❑ Divide link bandwidth into “pieces”
 - ❖ a). Frequency division (e.g. radio channels)
 - ❖ b). Time division (time slots)

Problem: Inefficiency remains

- ❑ Multiple connections can be made over the same link
- ❑ Resources are now shared
- ❑ But each connection “owns” its part of the link
- ❑ What if a connection stops transferring data?
 - ❖ Bandwidth or time is wasted
- ❑ What if the link goes down?
- ❑ ***Solution: Packet switching***

1.3.2 Packet Switching

Each end-end data stream
divided into *packets*

- ❑ user A, B packets *share* network resources
- ❑ each packet uses full link bandwidth
- ❑ resources used as *needed*

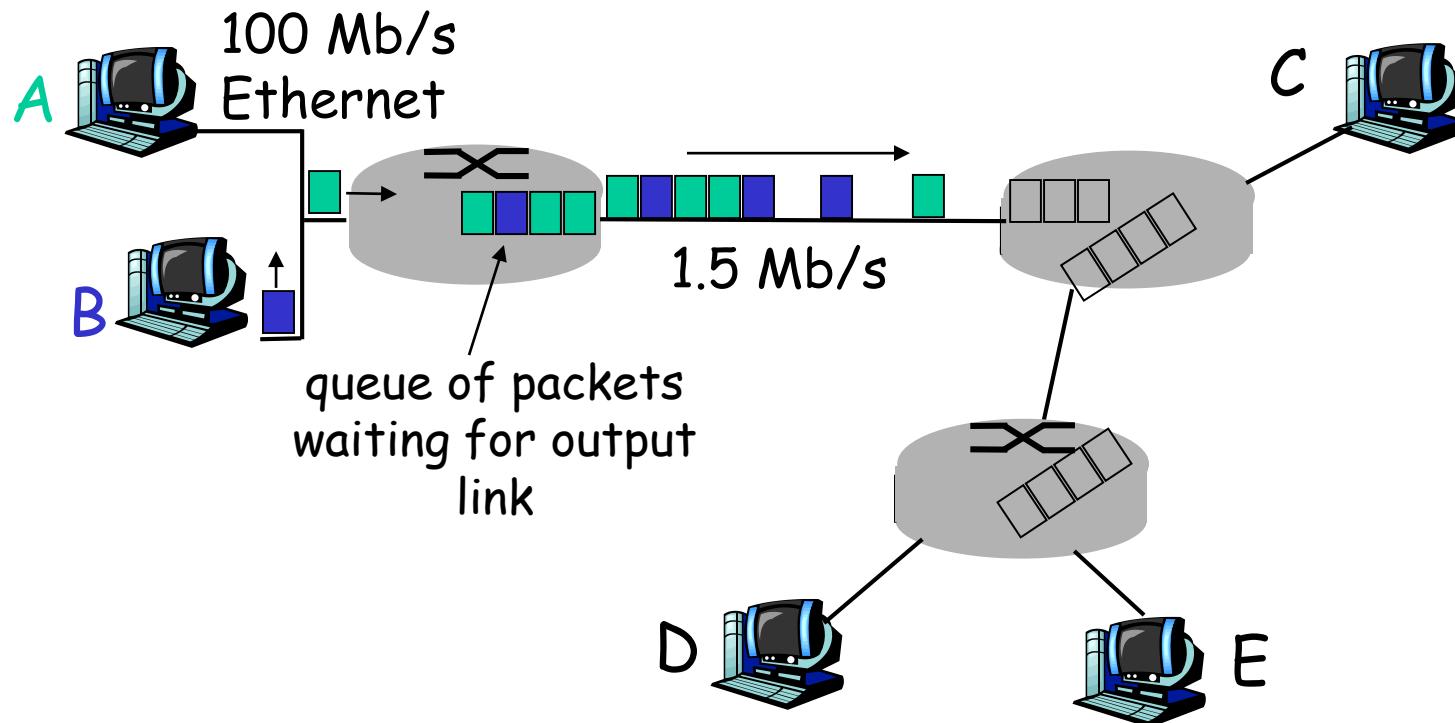
Bandwidth division into “pieces”
Dedicated allocation
Resource reservation



Resource contention:

- ❑ aggregate resource demand can exceed amount available
- ❑ congestion: packets queue, wait for link use
- ❑ store and forward:
packets move one hop at a time
 - ❖ Node receives complete packet before forwarding

Packet Switching in action



- ❑ Note: On-demand sharing of resources (i.e. packet switching) sometimes called *statistical multiplexing*

Packet-switching: store-and-forward



- ❑ Takes L/R seconds to transmit (push out) packet of L bits on to link of R bps
- ❑ *store and forward*:
 - ❖ Entire packet must arrive at router before it can be transmitted on next link
- ❑ Delay = $3L/R$ (assuming zero propagation delay)

Example:

- ❑ $L = 7.5$ Mbits
- ❑ $R = 1.5$ Mbps
- ❑ delay = 15 sec

Q – What if data arrives faster than it can be served?
look up Erlang and queueing theory

Is Packet switching always better than circuit switching?

- **Packet switching is great for bursty data**
 - ❖ E.g. Ethernet, the web, etc.
 - ❖ resource sharing
 - ❖ simpler, no call setup
- **BUT: Needs mechanisms to cope with excessive congestion**
 - ❖ Congestion = packet delay and loss
 - ❖ protocols needed for reliable data transfer, congestion control
- **Q: How to provide circuit-like behavior?**
 - ❖ bandwidth guarantees are needed for audio/video apps

Chapter 1: roadmap

1.1 What *is* the Internet?

1.2 Network Structure

- ❑ End systems, access networks, links
- ❑ Network Edge
- ❑ Access Network
- ❑ Network Core

1.3 Network Core in detail

- ❑ circuit switching, packet switching, network structure

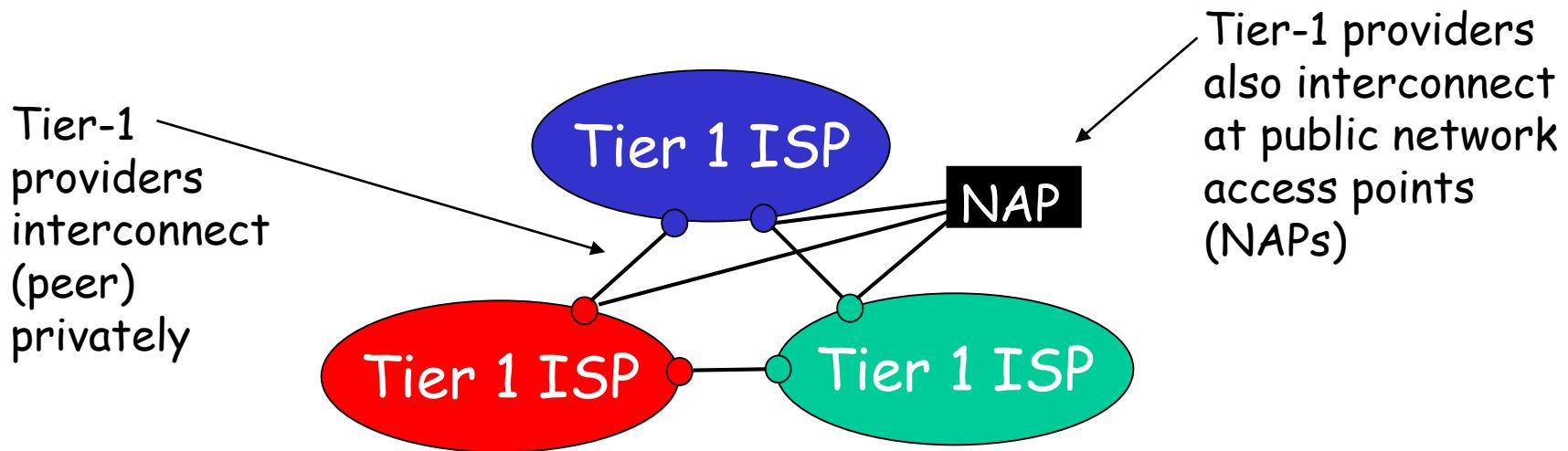
1.4 Internet Structure and ISPs

1.5 Protocol layers, service models

1.6 History

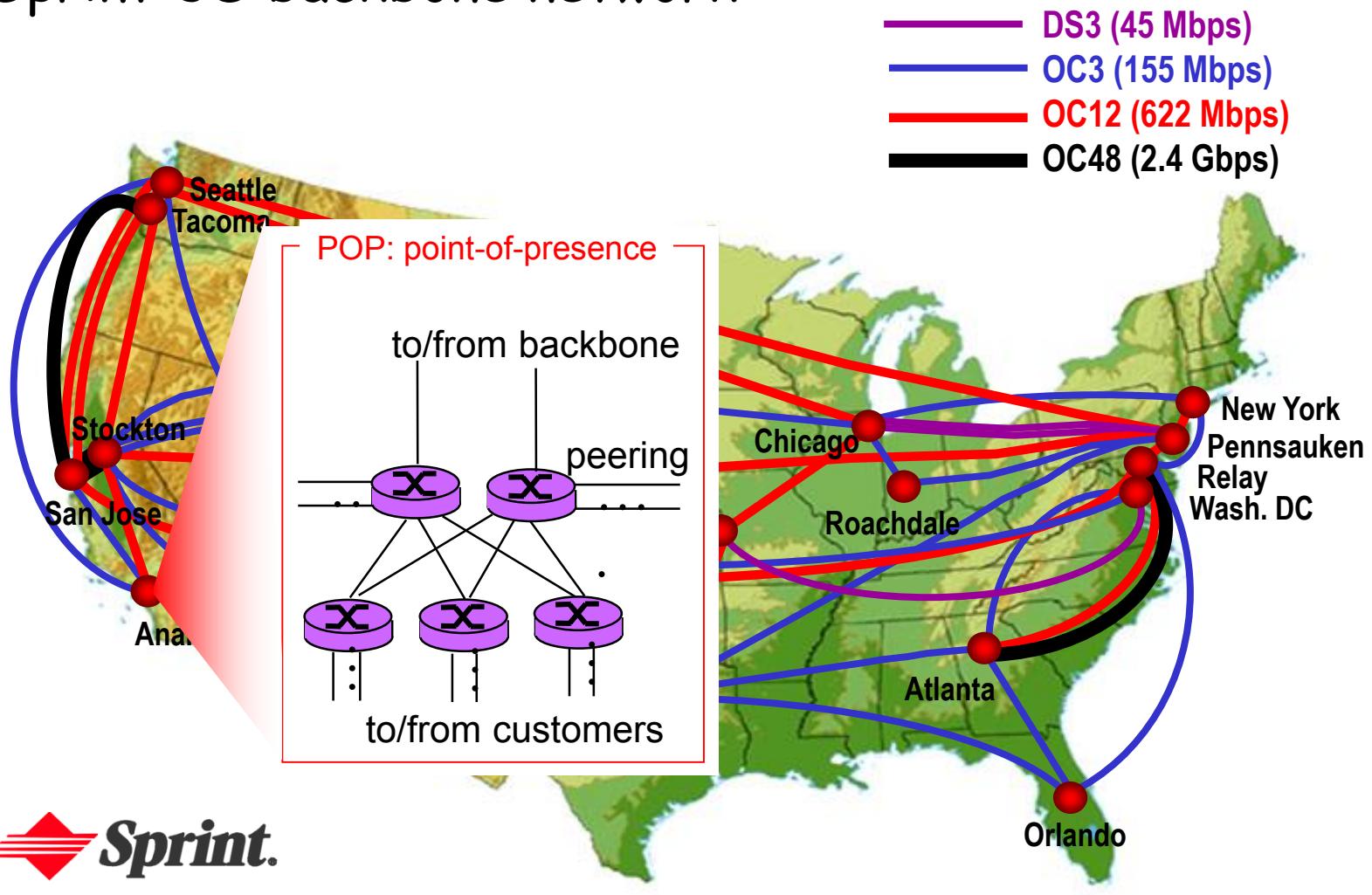
1.4 Internet structure: a network of networks

- Roughly hierarchical
- At center: “tier-1” ISPs (e.g., MCI, Sprint, AT&T, Cable and Wireless), national/international coverage
 - ❖ treat each other as equals



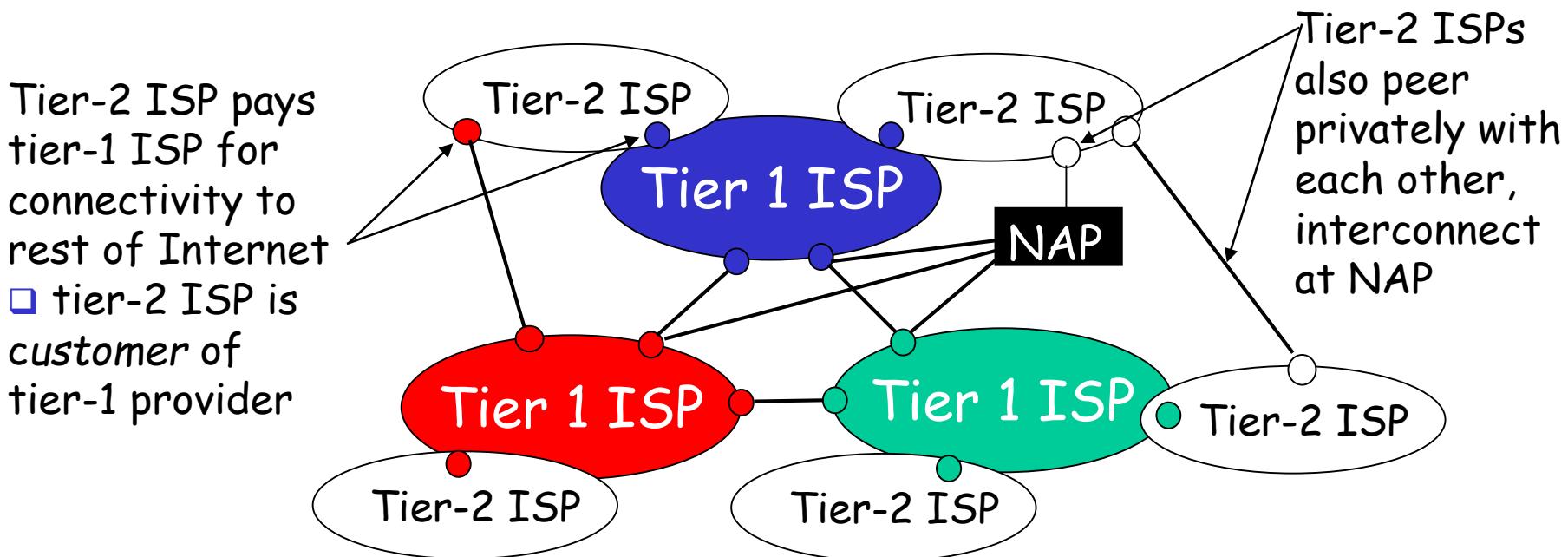
Tier-1 ISP: e.g., Sprint

Sprint US backbone network



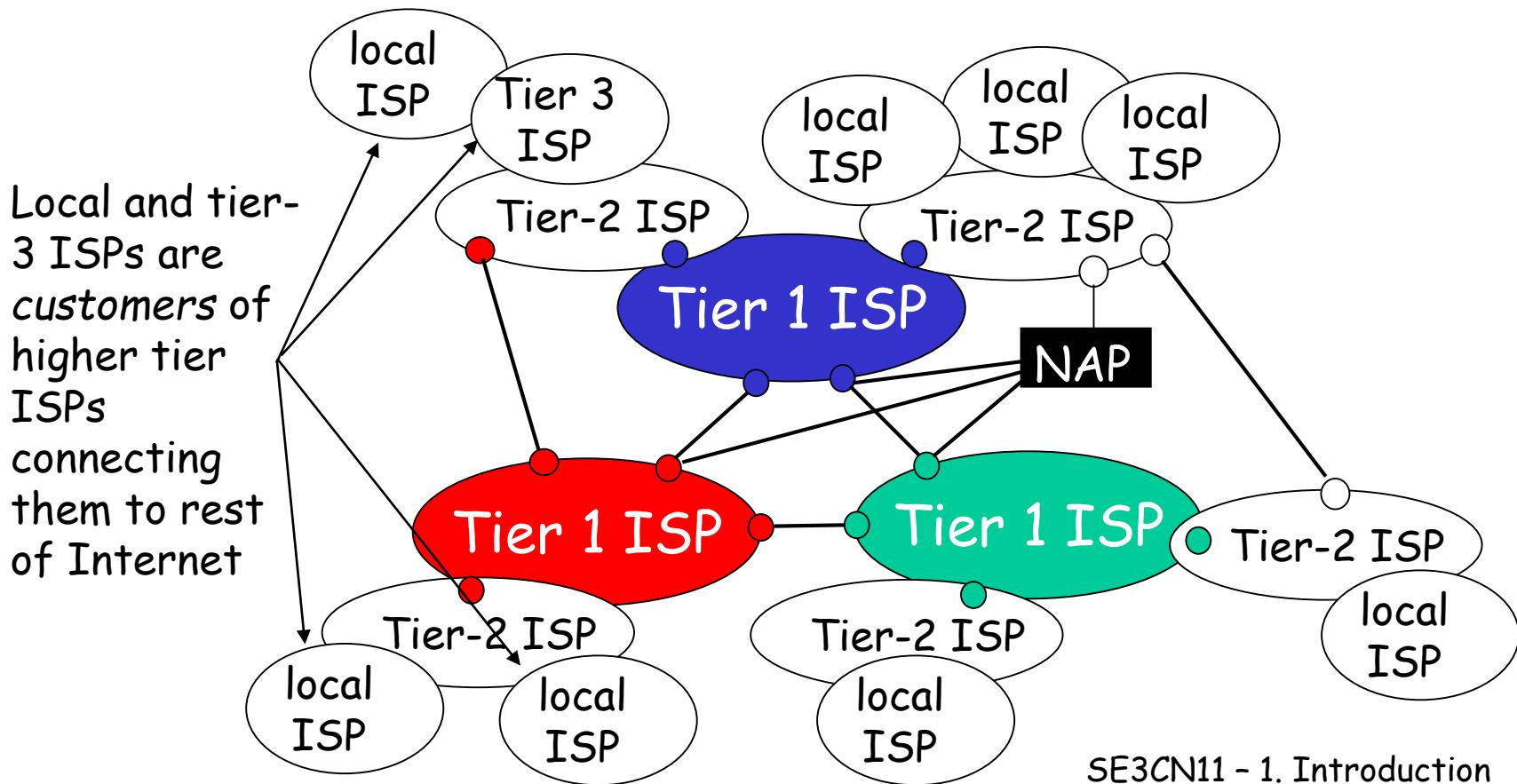
Internet structure: network of networks

- “Tier-2” ISPs: smaller (often regional) ISPs
 - ❖ Connect to one or more tier-1 ISPs, possibly other tier-2 ISPs



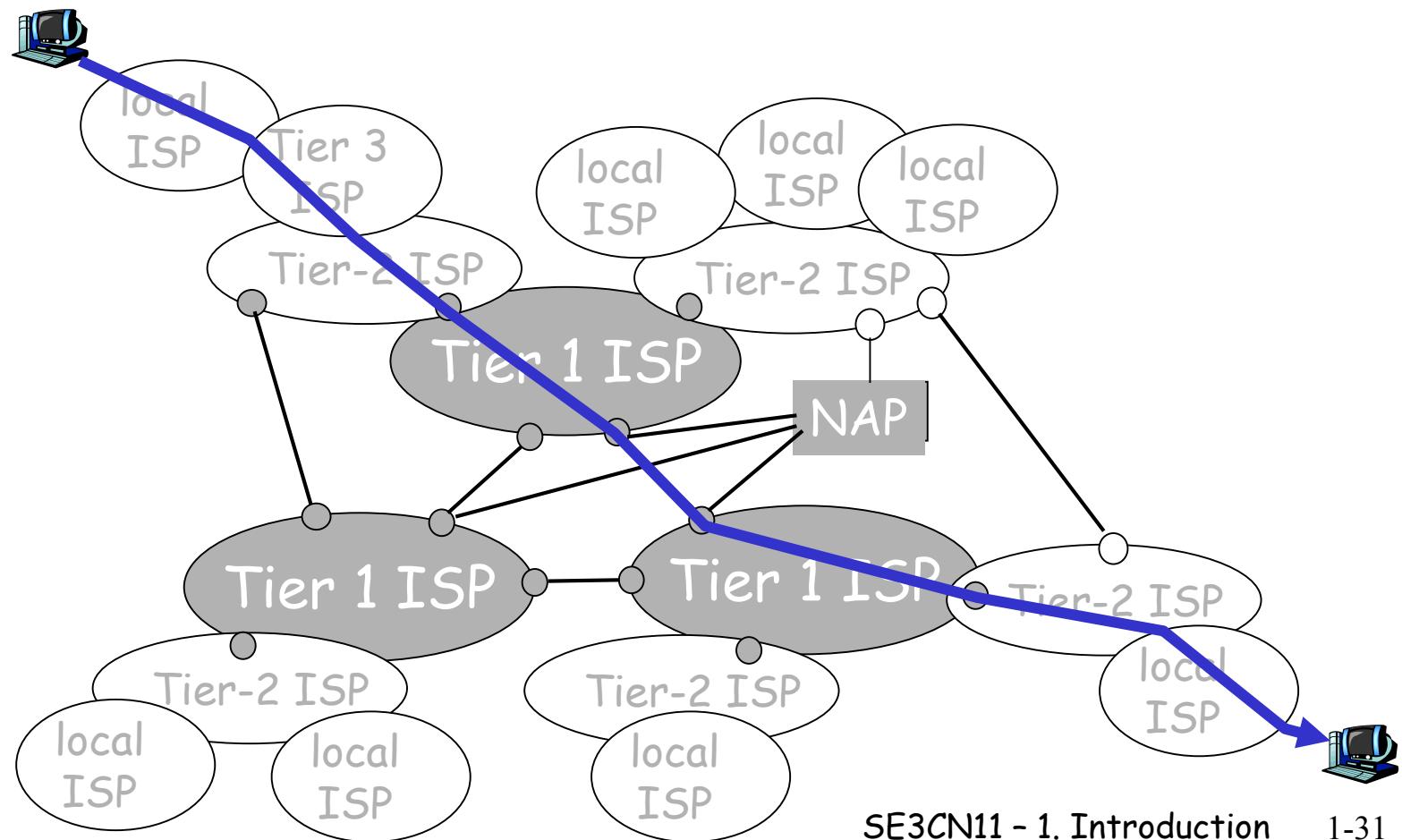
Internet structure: network of networks

- “Tier-3” ISPs and local ISPs
 - ❖ last hop (“access”) network (closest to end systems)



Internet structure: network of networks

- a packet passes through many networks!



Net neutrality

- ❑ We cannot discuss the Internet without dealing with neutrality
 - ❖ *All animals are equal, but some animals are more equal than others...*
- ❑ Modern issue – should some services, e.g. Netflix be given priority over others? Indeed who decides?
- ❑ 2007 - Senator Barack Obama pledges support for net neutrality to protect a free and open Internet if elected President...

THE WHITE HOUSE

WASHINGTON

February 26, 2015

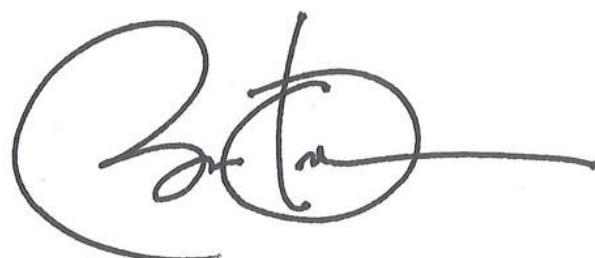
Today's FCC decision will protect innovation and create a level playing field for the next generation of entrepreneurs—and it wouldn't have happened without Americans like you.

More than 4 million people wrote in to the FCC, overwhelmingly in support of a free and fair internet. Countless others spoke out on social media, petitioned their government, and stood up for what they believe.

I ran for office because I believed that nothing can stand in the way of millions of voices calling for change. That's the backbone of our democracy—and you've proven that this timeless principle is alive and well in our digital age.

So to all the people who participated in this conversation, I have a simple message:

Thank you,

A handwritten signature in black ink, appearing to read "Barack Obama". The signature is fluid and cursive, with a large, stylized 'B' and 'O'.

□ See

1. <https://www.whitehouse.gov/net-neutrality>
2. http://www.circleid.com/posts/20140717_connectivity_policy_and_the_open_internet/
3. <http://frankston.com/public/?n=beyondneutrality>

Chapter 1: roadmap

1.1 What *is* the Internet?

1.2 Network Structure

- ❑ End systems, access networks, links
- ❑ Network Edge
- ❑ Access Network
- ❑ Network Core

1.3 Network Core in detail

- ❑ circuit switching, packet switching, network structure

1.4 Internet Structure and ISPs

1.5 Protocol layers, service models

1.6 History

1.5 Protocol Layers

Networks are complex!

❑ many “pieces”:

- ❖ hosts
- ❖ routers
- ❖ links of various media
- ❖ applications
- ❖ protocols
- ❖ hardware, software

Question:

Is there any hope of *organizing* structure of network?

Or at least our discussion of networks?

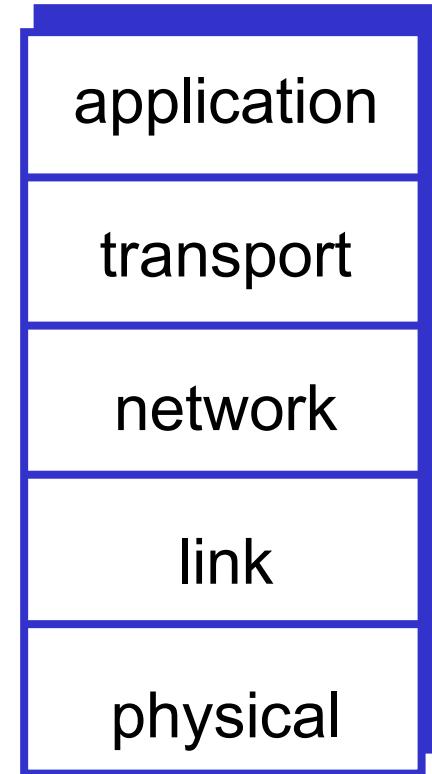
Layered Architecture

When dealing with complex systems:

- explicit structure allows identification and relationship of complex system's pieces
 - ❖ i.e. a **layered reference model**
- **modularization** eases maintenance and updating of system
 - ❖ change of implementation of layer's service is transparent to rest of system
- **Networks thus employ Protocol Layering**
- Result: a **Protocol Stack**

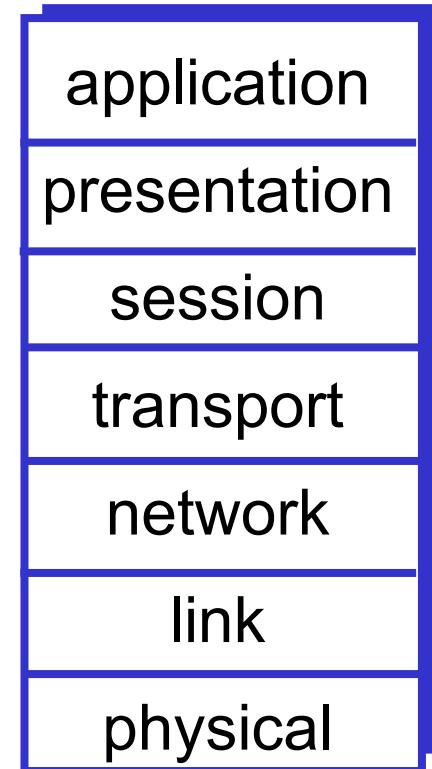
Internet Protocol Stack

- **application**: supporting network applications
 - ❖ FTP, SMTP, HTTP
- **transport**: process-process data transfer
 - ❖ TCP, UDP
- **network**: routing of datagrams from source to destination
 - ❖ IP, routing protocols
- **link**: data transfer between neighboring network elements
 - ❖ Ethernet, Wi-Fi
- **physical**: bits “on the wire” or “radio”

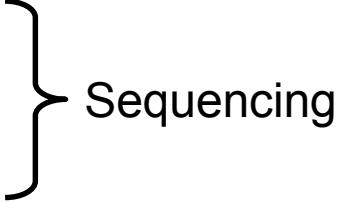


ISO/OSI reference model

- ❑ **presentation**: allow applications to interpret meaning of data, e.g., encryption, compression, machine-specific conventions
- ❑ **session**: synchronization, checkpointing, recovery of data exchange
- ❑ Internet stack “missing” these layers!
 - ❖ these services, *if needed*, must be implemented in application
 - ❖ needed?



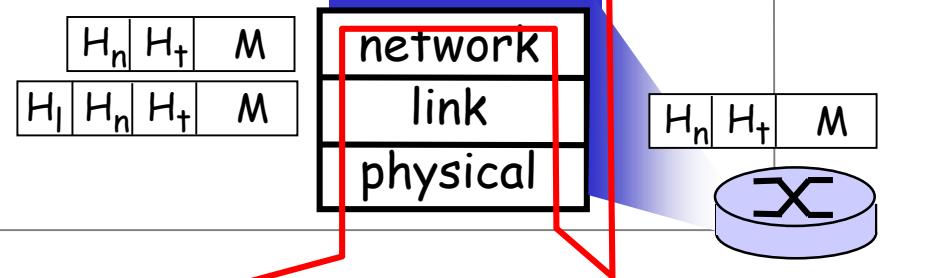
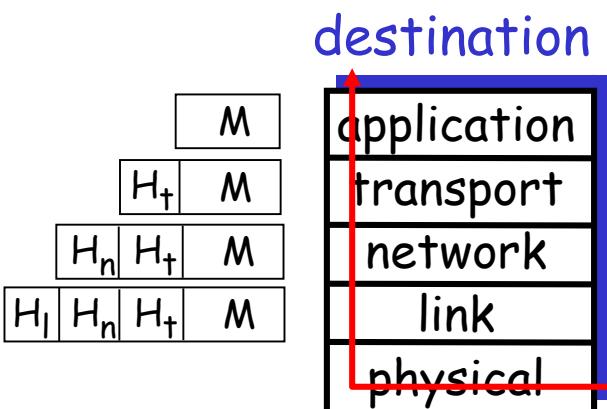
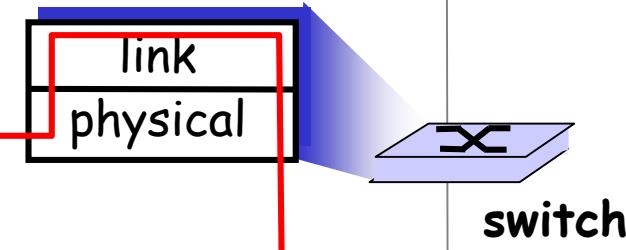
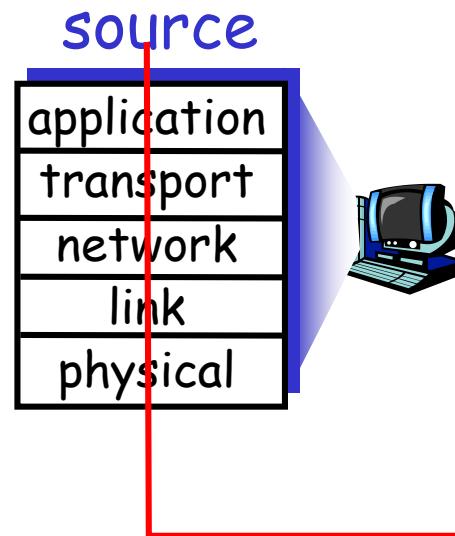
Protocol Functions

- ❑ Small set of functions that form basis of **all** protocols
- ❑ Not all protocols have all functions
 - ❖ May have same type of function in protocols at different levels
- ❑ **Functions:**
 1. Encapsulation
 2. Fragmentation and reassembly
 3. Connection control
 4. Ordered delivery
 5. Flow control
 6. Error control
 7. Addressing
 8. Multiplexing
 9. Transmission services

Sequencing

Encapsulation

message	M
segment	H_t M
datagram	H_n H_t M
frame	H_l H_n H_t M



Chapter 1: roadmap

1.1 What *is* the Internet?

1.2 Network Structure

- ❑ End systems, access networks, links
- ❑ Network Edge
- ❑ Access Network
- ❑ Network Core

1.3 Network Core in detail

- ❑ circuit switching, packet switching, network structure

1.4 Internet Structure and ISPs

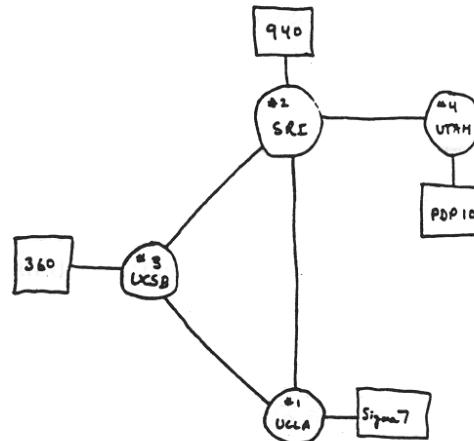
1.5 Protocol layers, service models

1.6 History

1.7 Internet History

1961-1972: Early packet-switching principles

- 1961: Kleinrock - queueing theory shows effectiveness of packet-switching
- 1964: Baran - packet-switching in military nets
- 1967: ARPAnet conceived by Advanced Research Projects Agency
- 1969: first ARPAnet node operational
- 1972:
 - ❖ ARPAnet public demonstration
 - ❖ NCP (Network Control Protocol) first host-host protocol
 - ❖ first e-mail program
 - ❖ ARPAnet has 15 nodes



Internet History

1972-1980: Internetworking, new and proprietary nets

- 1970: ALOHAnet satellite network in Hawaii
- 1974: Cerf and Kahn - architecture for interconnecting networks
- 1976: Ethernet at Xerox PARC
- late 70' s: proprietary architectures: DECnet, SNA, XNA
- late 70' s: switching fixed length packets (ATM precursor)
- 1979: ARPAnet has 200 nodes

Cerf and Kahn's internetworking principles:

- ❖ minimalism, autonomy - no internal changes required to interconnect networks
- ❖ ***best effort*** service model
- ❖ stateless routers
- ❖ decentralized control

define today's Internet architecture

Internet History

1980-1990: new protocols, a proliferation of networks

- 1983: deployment of TCP/IP
- 1982: SMTP e-mail protocol defined
- 1983: DNS defined for name-to-IP-address translation
- 1985: ftp protocol defined
- 1988: TCP congestion control
- new national networks: Csnet, BITnet, NSFnet, Minitel
- 100,000 hosts connected to confederation of networks

Internet History

1990, 2000's: commercialization, the Web, new apps

- Early 1990's: ARPAnet decommissioned
- 1991: NSF lifts restrictions on commercial use of NSFnet (decommissioned, 1995)
- early 1990s: Web
 - ❖ hypertext [Bush 1945, Nelson 1960's]
 - ❖ December 25th, 1990: First Web Server goes live [Berners-Lee]
 - ❖ 1994: Mosaic, later Netscape
 - ❖ late 1990's: commercialization of the Web

Late 1990's – 2000's:

- DotCom boom
- DotCom crash
- more killer apps: instant messaging, P2P file sharing, blogs, wikis, social networking
- Web 2.0
- network security to forefront
- est. 50 million host, 100 million+ users
- backbone links running at Gbps

Now...

Don't fall into the networking trap – go mobile...

- Cisco:
 - ❖ Global mobile data traffic grew 69 percent in 2014 consuming 2.5 exabytes per month at the end of 2014, up from 1.5 exabytes per month at the end of 2013.
 - ❖ In 2013, mobile data traffic was nearly 30 times the size of the entire global Internet in 2000.
 - ❖ Mobile video traffic exceeded 50 percent of total mobile data traffic for the first time in 2012
 - ❖ 497 million mobile devices and connections were added in 2014.
 - ❖ Global mobile devices and connections in 2014 grew to 7.4 billion, up from 6.9 billion in 2013. Smartphones accounted for 88 percent of that growth, with 439 million net additions in 2014.

Interesting future

□ Cisco:

- ❖ By 2018, more internet traffic will be generated by wireless internet-connected things than wired.
- ❖ The amount of internet traffic generated by **PCs** will have decreased to 43 percent, down from 67 percent in 2013.
- ❖ Majority of traffic generated by smartphones, tablets, TVs and machine to machine” M2M.
- ❖ Check out the **Internet of Things (IoT)** – 50 Billion connected devices by 2020.

Bob's video for us...

- ❑ Bob Frankston is an Internet pioneer. He created many technologies, including the spreadsheet and NAT.
 - ❖ This year he celebrated 50 years of an Internet connection to his house – Internet is not new.
- ❑ He was happy to do a video for us (videochat timings may not work out) on **wired, wireless, should it matter? (on Blackboard)**
 - ❖ Important point is to enable the data to route from end to end, using any medium.

Summary

Covered a “ton” of material!

- ❑ Internet overview
- ❑ What’s a protocol?
- ❑ Network edge, core, access network, edge router
 - ❖ packet-switching versus circuit-switching
- ❑ Communication services
 - ❖ Connection-oriented vs connectionless
- ❑ Internet/ISP structure
- ❑ Layering and service models
- ❑ History, now and future

You now have:

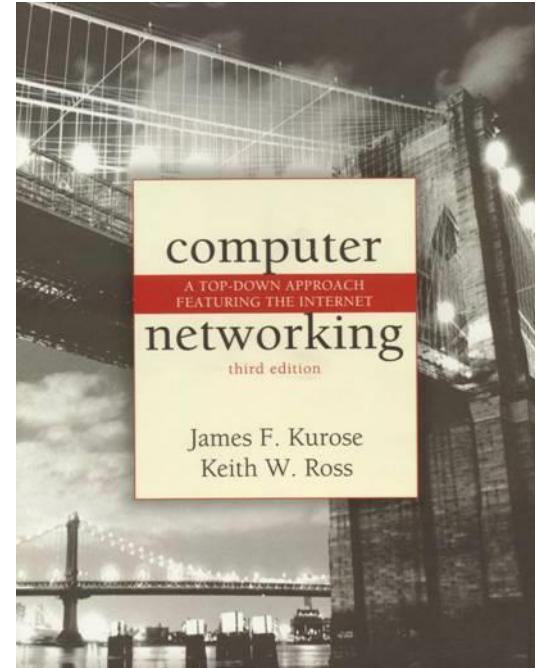
- ❑ Context, overview, “feel” of networking
- ❑ more depth and detail *to follow!*

Computer Networks

Chapter 2 Application Layer

Professor R Simon Sherratt

Adapted and updated from the original work
of Dr. Michael Evans



*Computer
Networking: A Top
Down Approach
Featuring the
Internet,
Jim Kurose, Keith
Ross
Addison-Wesley*

Chapter 2: Application Layer

Our goals:

- ❑ Conceptual, implementation aspects of network application protocols
 - ❖ transport-layer service models
 - ❖ client-server paradigm
 - ❖ peer-to-peer paradigm
- ❑ Learn about protocols by examining popular application-level protocols
 - ❖ HTTP
 - ❖ DNS
- ❑ Programming network applications
 - ❖ Socket API

Chapter 2: Application layer

2.1 Principles of network applications

2.2 Web and HTTP

2.3 DNS

2.4 P2P file sharing

2.5 Socket programming with TCP

2.6 Socket programming with UDP

Introduction: Some network apps

- ❑ e-mail
- ❑ web
- ❑ instant messaging
- ❑ remote login
- ❑ P2P file sharing
- ❑ multi-user network games
- ❑ streaming stored video clips
- ❑ social networks
- ❑ voice over IP
- ❑ real-time video conferencing
- ❑ grid computing

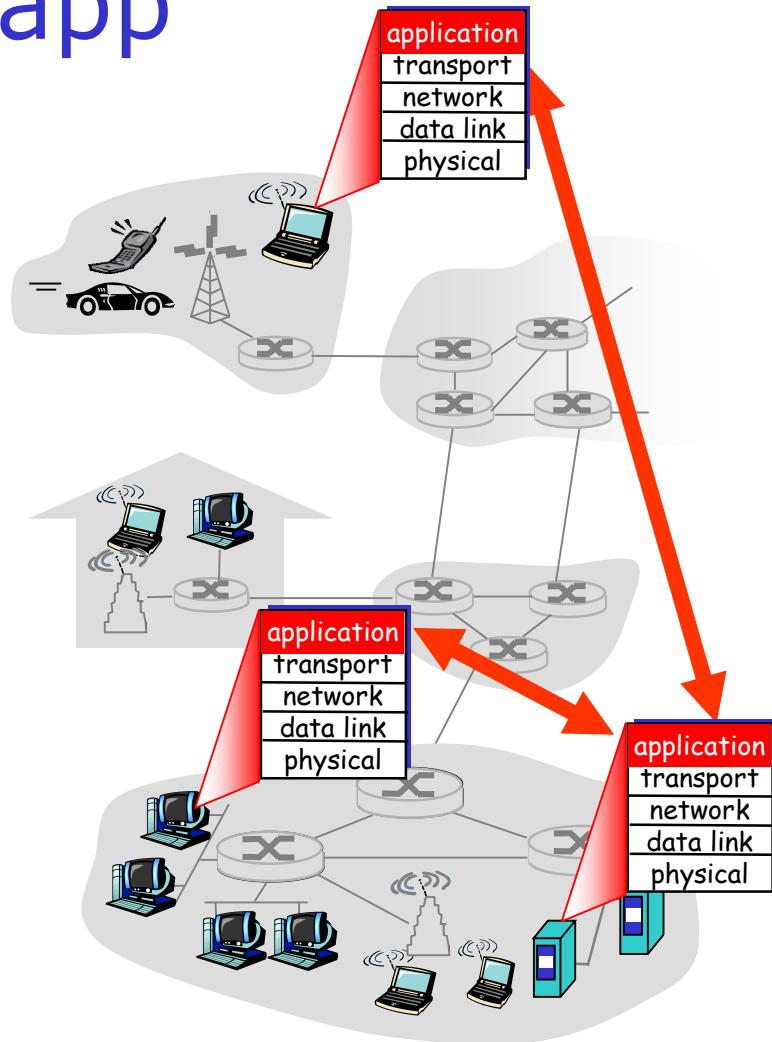
Creating a network app

Write programs that

- ❖ run on (different) *end systems*
- ❖ communicate over network – **any network**.
 - e.g., web server software communicates with browser software

No need to write software for network-core devices

- ❖ Network-core devices do not run user applications
- ❖ applications on end systems allows for rapid app development, propagation



Chapter 2: Application layer

2.1 Principles of network applications

2.2 Web and HTTP

2.3 DNS

2.4 P2P file sharing

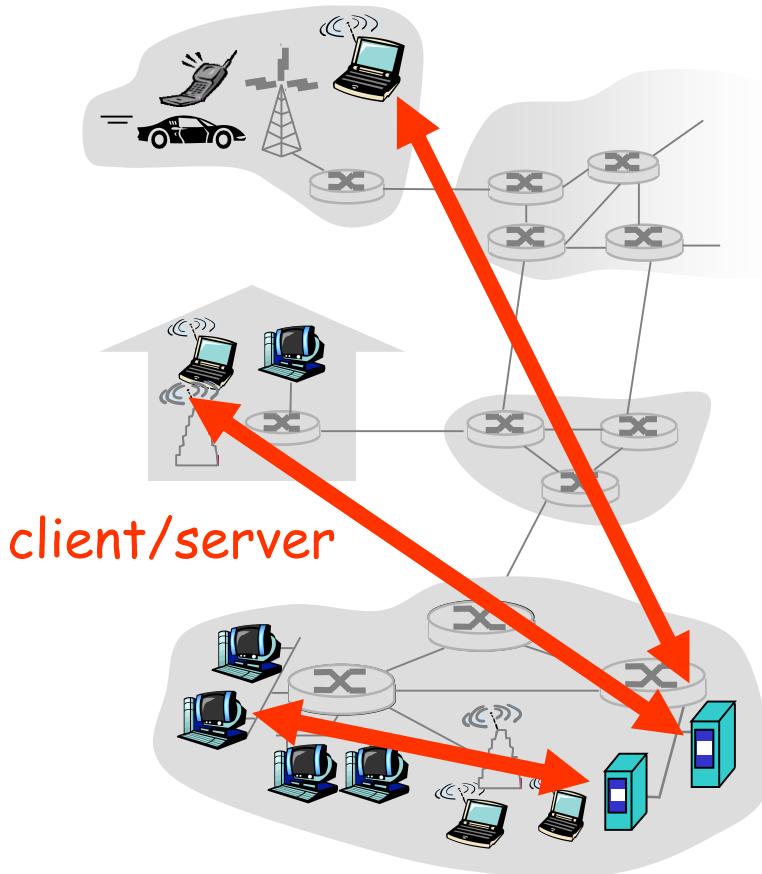
2.5 Socket programming with TCP

2.6 Socket programming with UDP

2.1.1 Application architectures

- ❑ Client-server
- ❑ Peer-to-peer (P2P)
- ❑ Hybrid of client-server and P2P

Client-server architecture



server:

- ❖ Is an always-on host
- ❖ permanent IP address
- ❖ server farms for scaling

clients:

- ❖ communicate with server
- ❖ may be intermittently connected
- ❖ may have dynamic (different each time) IP addresses
- ❖ do not communicate directly with each other

Google Data Centers

- ❑ Estimated cost of data center: \$600M
- ❑ Each data center uses 50-100 megawatts of power
- ❑ <http://www.youtube.com/watch?v=zRwPSFpLX8I>

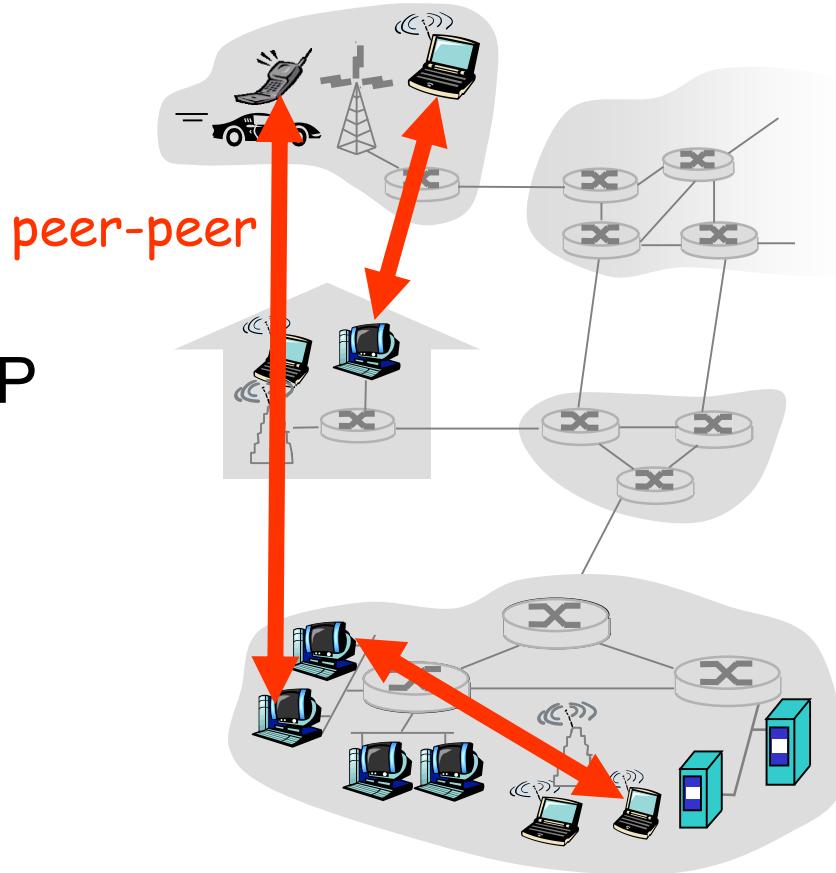


Pure P2P architecture

- ❑ No always-on server
- ❑ Arbitrary end systems directly communicate
- ❑ Peers are intermittently connected and change IP addresses

Highly scalable but difficult to manage

Q- How do you know the address of other device?



Hybrid of client-server and P2P

Skype

- ❖ Voice-over-IP P2P application
- ❖ Centralized server: finding address of remote party:
- ❖ Client-client connection: direct (not through server as total bandwidth would be too much)

Instant messaging

- ❖ Chatting between two users is P2P
- ❖ Centralized service: client presence detection/location
 - user registers its IP address with central server when it comes online
 - user contacts central server to find IP addresses of buddies

2.1.2 Process Communication

Process: a program running within a host.

- ❑ Within same host

- ❖ two processes communicate using **inter-process communication** (defined by OS).

- ❑ In different hosts

- ❖ communicate by exchanging **messages over network**.

Note: applications with P2P architectures have client processes & server processes

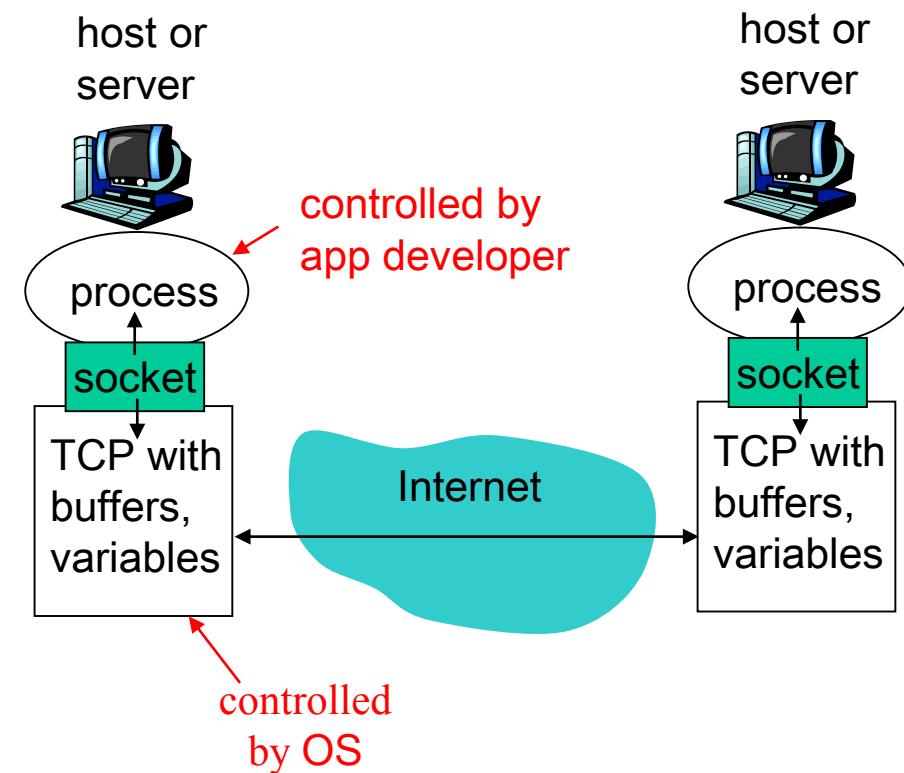
Client process: process that initiates communication

Server process: process that waits to be contacted.

But... not always, consider email clients that allow “push” notifications.

Sockets

- ❑ The “socket API” provides functions for communicating between processes
- ❑ Process sends/receives messages to/from its **socket**
- ❑ Socket analogous to door
 - ❖ sending process pushes message out of the door
 - ❖ sending process relies on transport infrastructure on other side of door which sends message to socket at receiving process



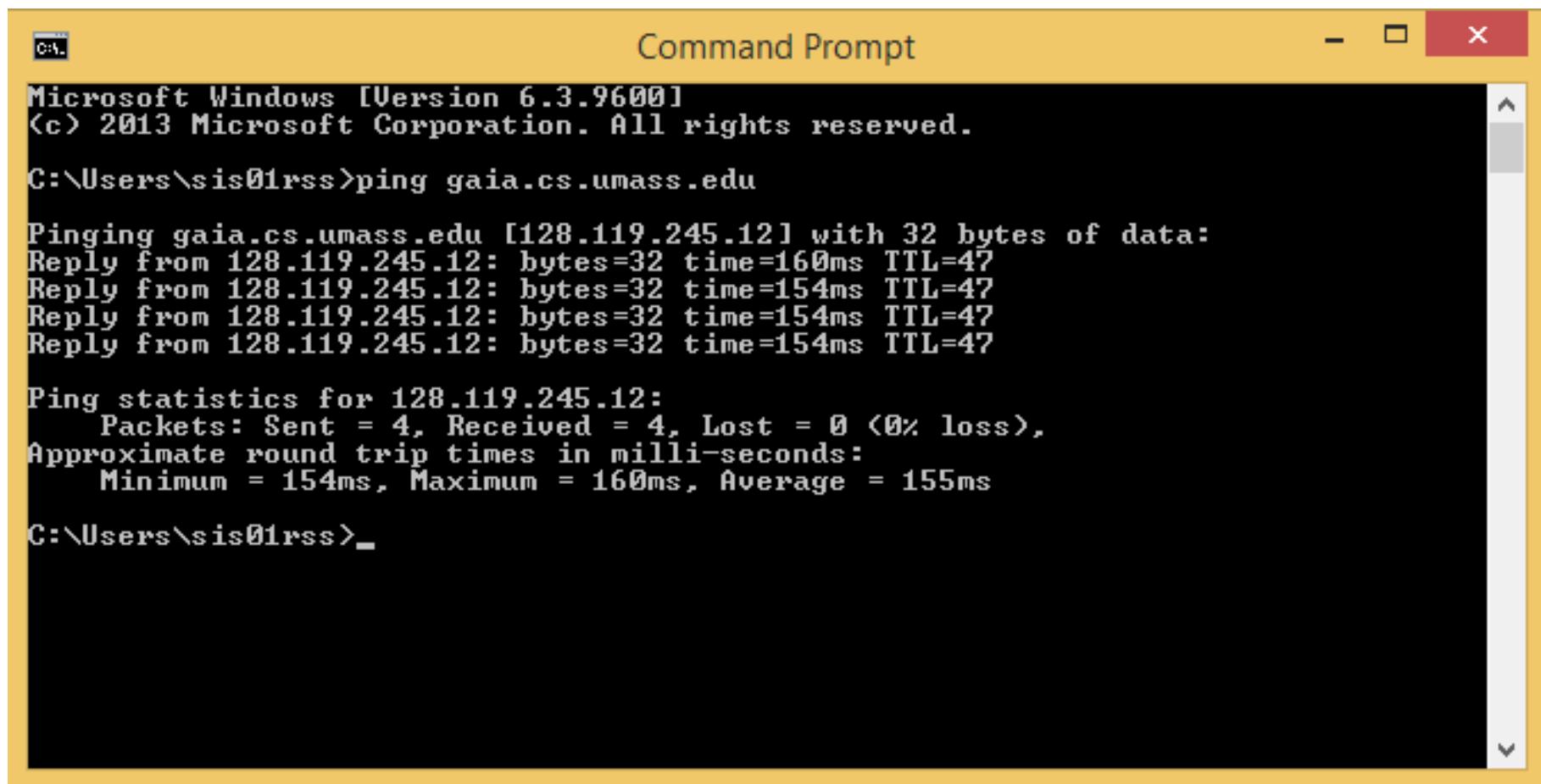
Addressing processes

- ❑ Just like a postal address, devices also need a unique *identifier*
- ❑ “TCP/IP” uses IPv4 giving a unique 32-bit IP address
- ❑ IPv6 being rolled out, with 2^{128} addresses - 3.4×10^{38} addresses.
- ❑ Vint Cerf is a father of the Internet, Chief Internet evangelist for Google (and a Reading honorary graduate) credited with creating Internet routing. See his video in IPv6 at
 - ❖ <http://www.google.com/intl/en/ipv6>

Addressing processes

- ❑ **Q:** does an IP address of a host on which process runs suffice for identifying each the process?
 - ❖ **Answer:** NO, many processes can be running on the same host
- ❑ *identifier* includes both **IP address** and **port numbers** associated with process on host.
- ❑ Example port numbers:
 - ❖ HTTP server: 80
 - ❖ Mail server: 25
- ❑ to send HTTP message to **gaia.cs.umass.edu** web server:
 - ❖ IP address: 128.119.245.12
 - ❖ Port number: 80

- ❖ Lets “Ping” that server and see:



Command Prompt

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\sis01rss>ping gaia.cs.umass.edu

Pinging gaia.cs.umass.edu [128.119.245.12] with 32 bytes of data:
Reply from 128.119.245.12: bytes=32 time=160ms TTL=47
Reply from 128.119.245.12: bytes=32 time=154ms TTL=47
Reply from 128.119.245.12: bytes=32 time=154ms TTL=47
Reply from 128.119.245.12: bytes=32 time=154ms TTL=47

Ping statistics for 128.119.245.12:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 154ms, Maximum = 160ms, Average = 155ms

C:\Users\sis01rss>
```

2.1.3 Application layer protocols

- Application layer protocol written by developer (i.e. you)
- Application layer protocol must define:
 - ❖ Types of messages exchanged,
 - e.g., request, response
 - ❖ Message syntax:
 - what fields in messages & how fields are delineated
 - ❖ Message semantics
 - meaning of information in fields
 - ❖ Rules for when and how processes send & respond to messages

Example Application protocols

Public-domain protocols:

- ❑ Defined in RFCs
- ❑ Allows for interoperability
- ❑ Examples:
 - ❖ HTTP
 - ❖ FTP
 - ❖ Telnet
 - ❖ SMTP, POP, IMAP
 - ❖ SNMP

Proprietary protocols:

- ❑ Used by specific applications
- ❑ Protocols can be open or closed
- ❑ Examples:
 - ❑ Skype
 - ❑ Instant Messaging protocols from Google, Microsoft, Yahoo! And AOL

2.1.4 Application Requirements

When defining applications, designers need to consider the following when determining the type of transport service it needs:

Data loss

- ❑ some apps (e.g., audio) can tolerate some loss
- ❑ other apps (e.g., file transfer, telnet) require 100% reliable data transfer

Timing

- ❑ some apps (e.g., Internet telephony, interactive games) require low delay to be usable

Bandwidth

- ❑ some apps (e.g., multimedia) require minimum amount of bandwidth to be “effective”
 - ❑ *Isochronous*
- ❑ other apps (“elastic apps”) make use of whatever bandwidth they get

Transport service requirements of common apps

Application	Data loss	Bandwidth	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100's msec
stored audio/video	loss-tolerant	same as above	
interactive games	loss-tolerant	few kbps up	yes, few secs
instant messaging	no loss	elastic	yes, 100's msec yes and no

Internet transport protocols services

Transmission Control

Protocol (TCP) service:

- ❑ *connection-oriented*: setup required between client and server processes
- ❑ *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 bandwidth guarantees

User Datagram Protocol (UDP) service:

- ❑ unreliable data transfer between sending and receiving process
- ❑ does not provide: connection setup, reliability, flow control, congestion control, timing, or bandwidth guarantee

Q: why bother? Why is there a UDP?

Internet apps: application, transport protocols

Application	Application layer protocol	Underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	proprietary (e.g. RealNetworks)	TCP or UDP
Internet telephony	proprietary (e.g., Vonage, Dialpad)	typically UDP

Chapter 2: Application layer

2.1 Principles of network applications

2.2 Web and HTTP

2.3 DNS

2.4 P2P file sharing

2.5 Socket programming with TCP

2.6 Socket programming with UDP

2.2 Web and HTTP

First some jargon

- Web page consists of **objects** (or **resources**)
- Object can be HTML file, JPEG image, Java applet, audio file,...
- Web page consists of **base HTML-file** which includes several referenced objects
- Each object is addressable by a **URL**
- Example URL:

www.someschool.edu/someDept/pic.gif

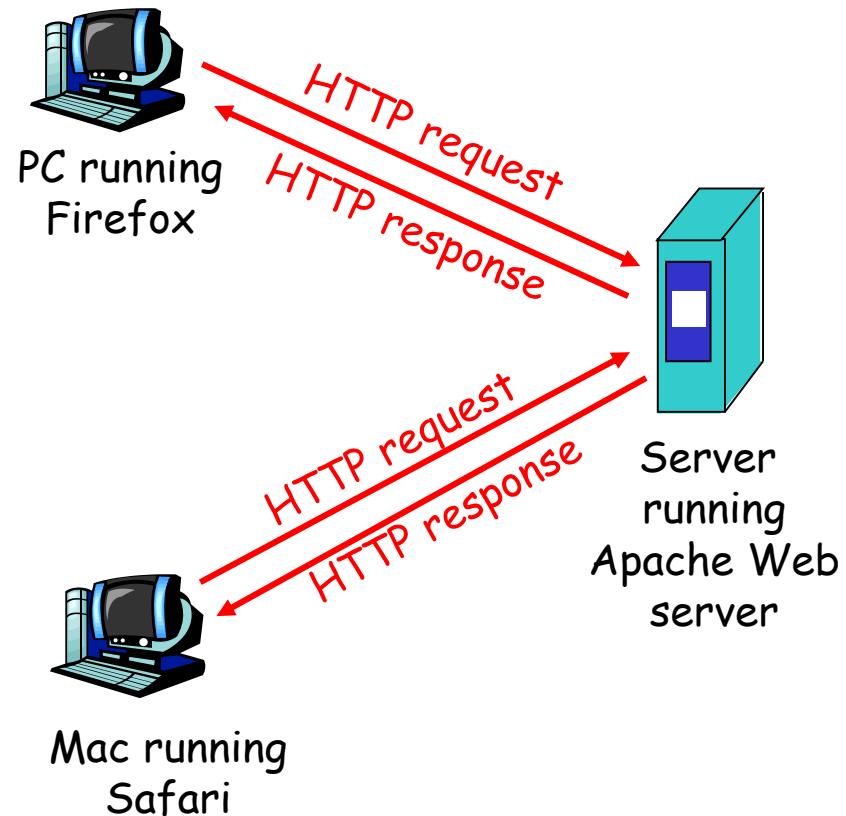
host name

path name

HTTP overview

HTTP: HyperText Transfer Protocol

- ❑ Web's application layer protocol
- ❑ client/server model
 - ❖ *client*: browser that requests, receives, "displays" Web objects
 - ❖ *server*: Web server sends objects in response to requests
- ❑ HTTP 1.0: RFC 1945
- ❑ HTTP 1.1: RFC 2068



HTTP overview (continued)

Uses TCP so connection
orientated:

- ❑ client initiates TCP connection (creates socket) to server, port 80
- ❑ server accepts TCP connection from client
- ❑ HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- ❑ TCP connection closed or times out.

HTTP is “stateless”

- ❑ server maintains no information about past client requests

aside
Protocols that maintain “state”
are complex!

- ❑ past history (state) must be maintained
- ❑ if server/client crashes, their views of “state” may be inconsistent, must be reconciled

HTTP connections

Nonpersistent HTTP

- ❑ At most one object is sent over a TCP connection.
- ❑ HTTP/1.0 uses nonpersistent HTTP

Persistent HTTP

- ❑ Multiple objects can be sent over single TCP connection between client and server.
- ❑ HTTP/1.1 uses persistent connections in default mode

2.2.1 Nonpersistent HTTP

Suppose user enters URL

www.someSchool.edu/someDepartment/home.index

(contains text,
references to 10
jpeg images)

- 1a. HTTP client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80
- 1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80. “accepts” connection, notifying client
2. HTTP client sends HTTP *request message* (containing URL) into TCP connection socket. Message indicates that client wants object someDepartment/home.index
3. HTTP server receives request message, forms *response message* containing requested object, and sends message into its socket

time
↓

Nonpersistent HTTP (cont.)

time
↓

4. HTTP server closes TCP connection.
5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects
6. Steps 1-5 repeated for each of 10 jpeg objects



Non-Persistent HTTP: Response time

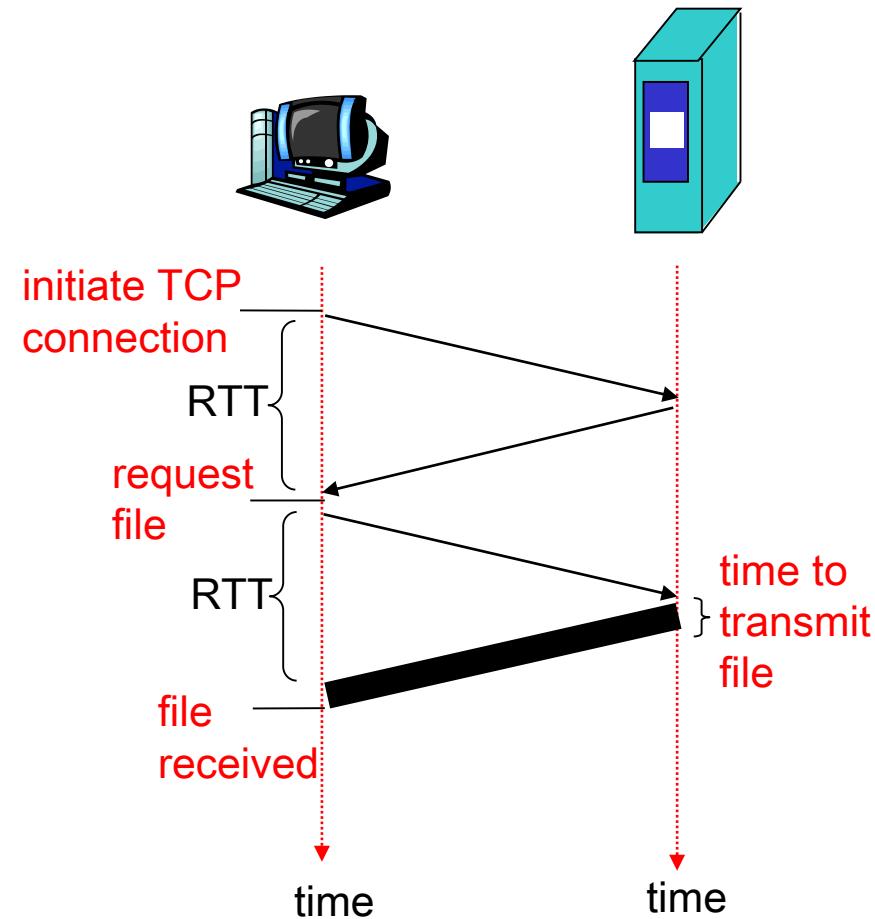
Definition of Round Trip time

(RTT): time to send a small packet to travel from client to server and back.

Response time:

- ❑ one RTT to initiate TCP connection
- ❑ one RTT for HTTP request and first few bytes of HTTP response to return
- ❑ file transmission time

$$\text{total} = 2\text{RTT} + \text{transmit time}$$

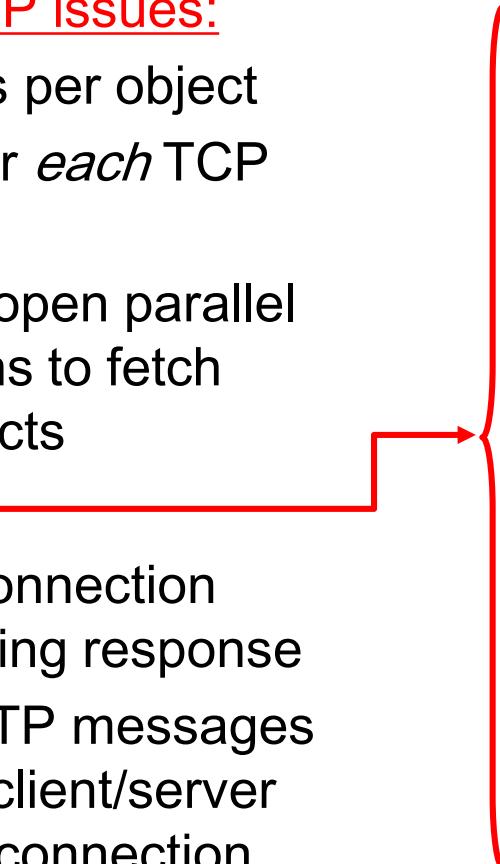


2.2.2 Persistent HTTP

Nonpersistent HTTP issues:

- ❑ requires 2 RTTs per object
- ❑ OS overhead for *each* TCP connection
- ❑ browsers often open parallel TCP connections to fetch referenced objects

Persistent HTTP



Persistent *without* pipelining:

- ❑ client issues new request only when previous response has been received
- ❑ one RTT for each referenced object

Persistent *with* pipelining:

- ❑ default in HTTP/1.1
- ❑ client sends requests as soon as it encounters a referenced object
- ❑ as little as one RTT for all the referenced objects

2.2.3 HTTP request message

- Two types of HTTP messages: *request, response*
- **HTTP request message:**
 - ❖ ASCII (human-readable format)

Request line
(GET, POST,
HEAD commands)

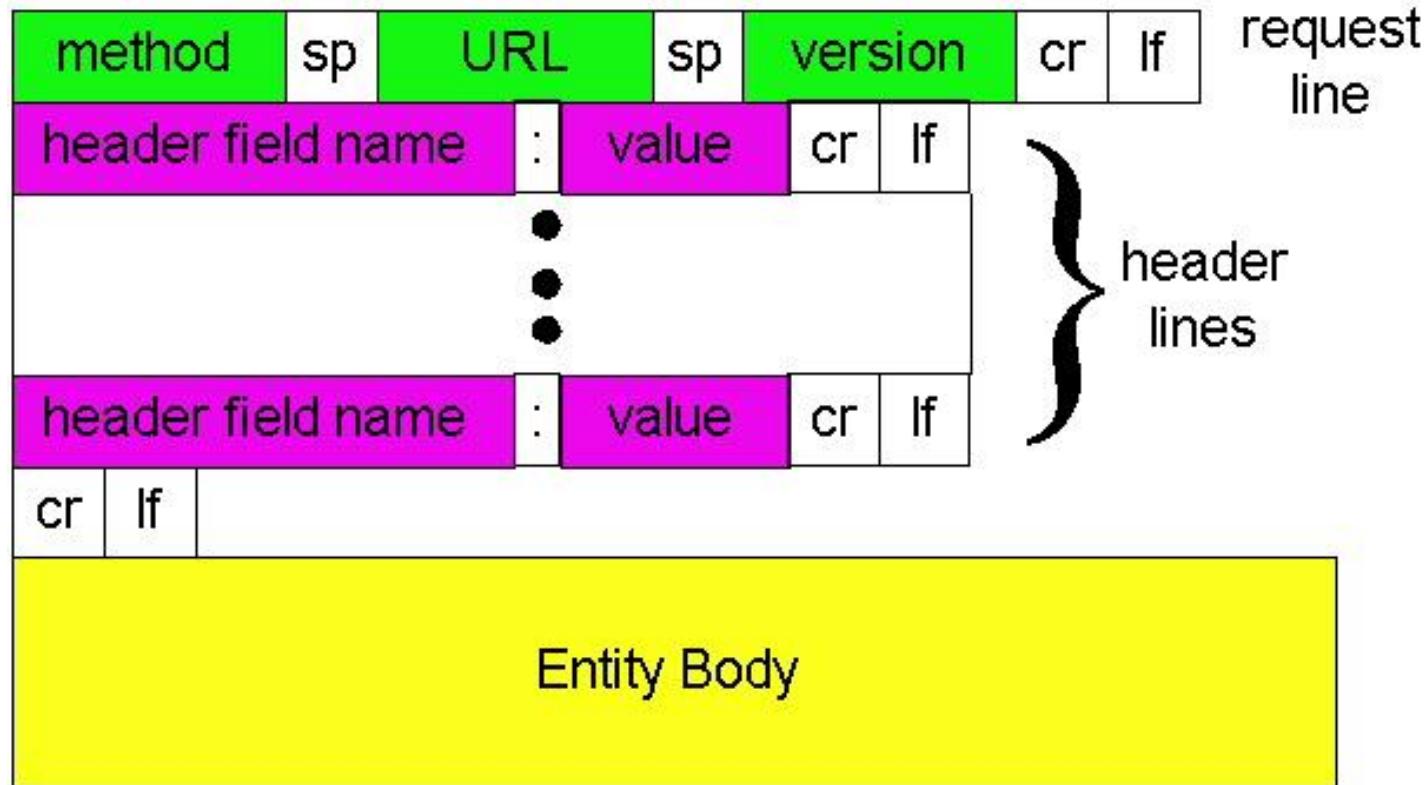
header
lines

Carriage return,
line feed
indicates end
of message

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language: fr
```

(extra carriage return, line feed)

HTTP request message: general format



Uploading form input

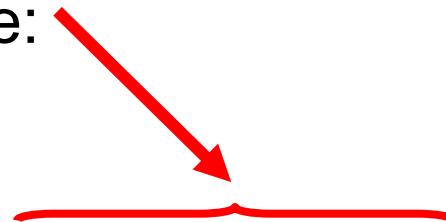
Post method:

- ❑ Web page often includes form input
- ❑ Input is uploaded to server in entity body

URL method:

- ❑ Uses GET method
- ❑ Input is uploaded in URL field of request line:

www.somesite.com/animalsearch?monkeys&banana



Method types

HTTP/1.0

- GET
- POST
- HEAD
 - ❖ asks server to leave requested object out of response

HTTP/1.1

- GET, POST, HEAD
- PUT
 - ❖ uploads file in entity body to path specified in URL field
- DELETE
 - ❖ deletes file specified in the URL field

2.2.4 HTTP response message

Status line
(protocol
status code
status phrase)

header
lines

data, e.g.,
requested
HTML file

HTTP/1.1 200 OK
Connection close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998
Content-Length: 6821
Content-Type: text/html

data data data data data ...

HTTP response status codes

In first line in server->client response message.

A few sample codes:

200 OK

- ❖ request succeeded, requested object later in this message

301 Moved Permanently

- ❖ requested object moved, new location specified later in this message (Location:)

400 Bad Request

- ❖ request message not understood by server

404 Not Found

- ❖ requested document not found on this server

505 HTTP Version Not Supported

2.2.5 User-server state: cookies

Many major Web sites use cookies

Four components:

- 1) cookie header line of HTTP *response* message
- 2) cookie header line in HTTP *request* message
- 3) cookie file kept on user's host, managed by user's browser
- 4) back-end database at Web site

Example:

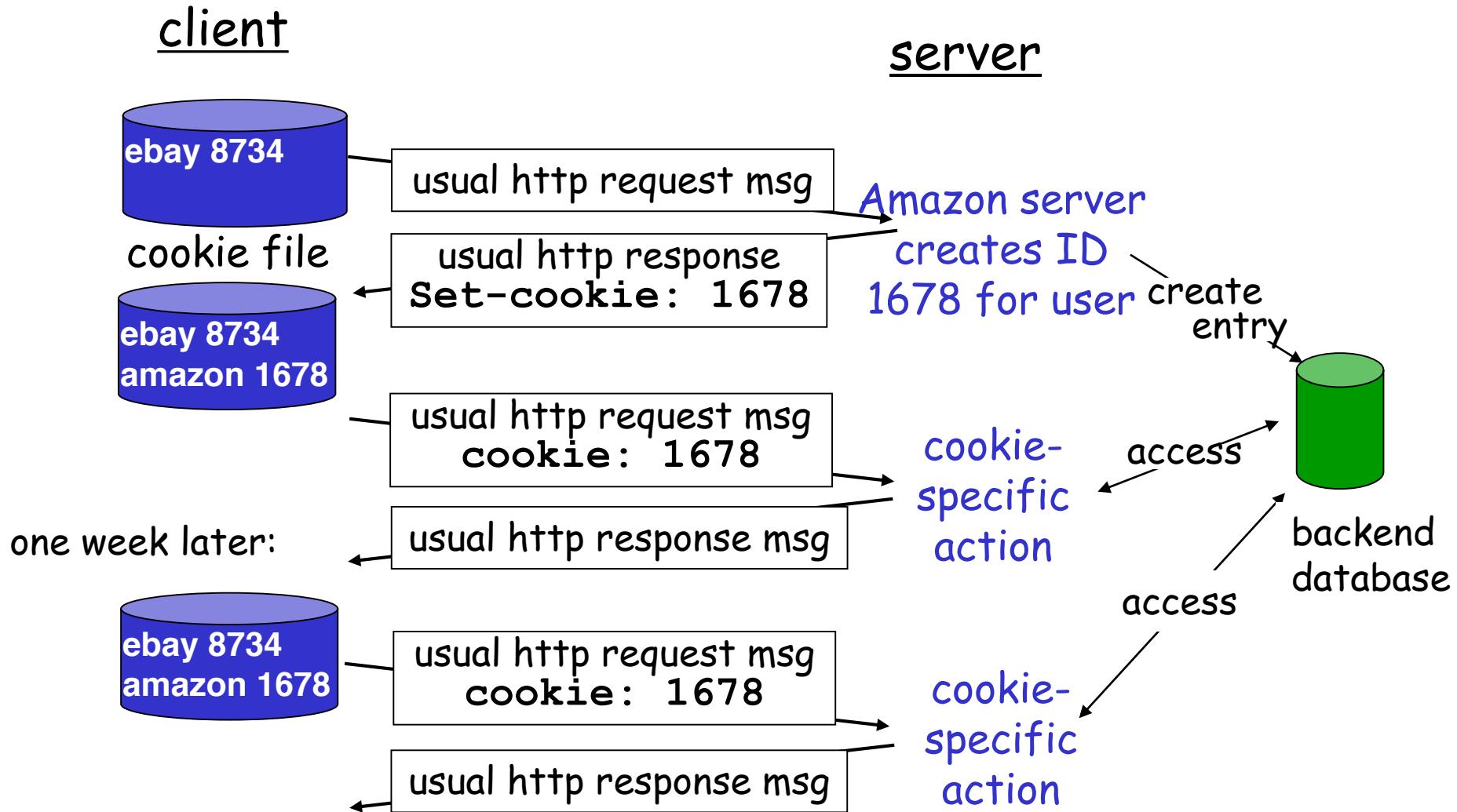
- ❖ Susan accesses Internet always from same PC
- ❖ She visits a specific e-commerce site for first time
- ❖ When initial HTTP requests arrives at site, site creates a unique ID and creates an entry in backend database for ID

The cookie law:

It started as an **EU Directive** that was adopted by all EU countries in May 2011. The Directive gave individuals rights to refuse the use of cookies that reduce their online privacy. Each country then updated its own laws to comply. In the UK this meant an update to the **Privacy and Electronic Communications Regulations**.

<http://www.cookie-law.org/the-cookie-law>

Cookies: keeping “state”



Cookies (continued)

What cookies can bring:

- ❑ authorization
- ❑ shopping carts
- ❑ recommendations
- ❑ user session state (Web e-mail)

aside

Cookies and privacy:

- ❑ cookies permit sites to learn a lot about you
- ❑ you may supply name and e-mail to sites

How to keep “state”:

- ❑ Protocol endpoints: maintain state at sender/receiver over multiple transactions
- ❑ cookies: http messages carry state

Chapter 2: Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 DNS
- 2.4 P2P file sharing
- 2.5 Socket programming with TCP
- 2.6 Socket programming with UDP

2.4 DNS: Domain Name System

People: many identifiers:

- ❖ National Insurance #, name, passport #

Internet hosts, routers:

- ❖ IP address
- ❖ “name”, e.g., www.yahoo.com - used by humans

Q: How do you map between IP addresses and name ?

Domain Name System (DNS):

- *Distributed database* implemented in hierarchy of many *name servers*
- *Application-layer protocol*
- Hosts, routers, name servers all communicate to *resolve* names (address/name translation)
 - ❖ Note: core Internet function, implemented as application-layer protocol
 - ❖ Complexity at network’s “edge”

DNS services

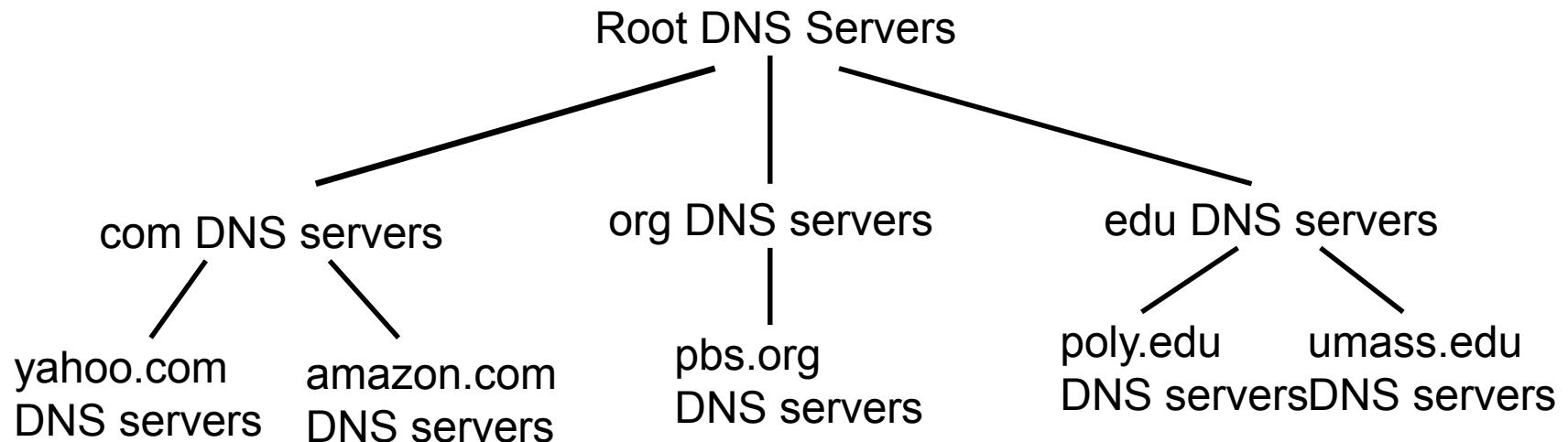
- ❑ Hostname to IP address translation
- ❑ Host aliasing
 - ❖ Canonical and alias names
- ❑ Mail server aliasing
- ❑ Load distribution
 - ❖ Replicated Web servers: set of IP addresses for one canonical name

Why not centralize DNS?

- ❑ single point of failure
- ❑ traffic volume
- ❑ distant centralized database
- ❑ maintenance

doesn't scale!

Distributed, Hierarchical Database

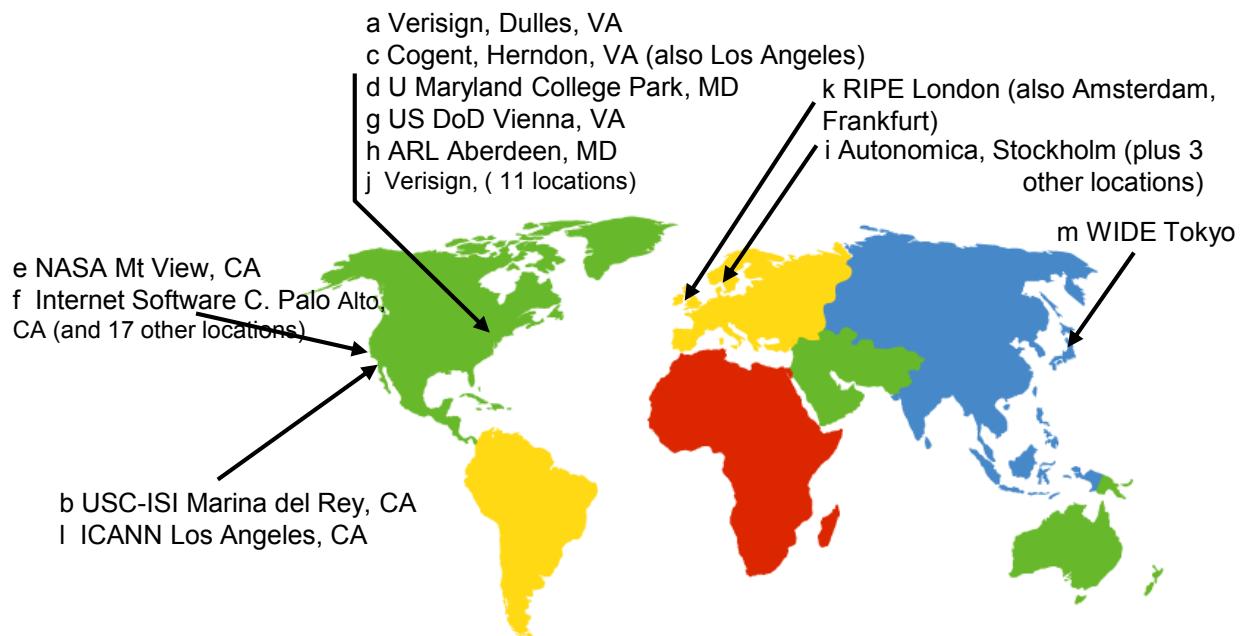


Client wants IP for www.amazon.com; 1st approx:

- Client queries a root server to find .com DNS server
- Client queries .com DNS server to get amazon.com DNS server
- Client queries amazon.com DNS server to get IP address for www.amazon.com

DNS: Root name servers

- Contacted by local name server that cannot resolve name
- Root name server:
 - ❖ contacts authoritative name server if name mapping not known
 - ❖ gets mapping
 - ❖ returns mapping to local name server



13 root name
servers worldwide

TLD and Authoritative Servers

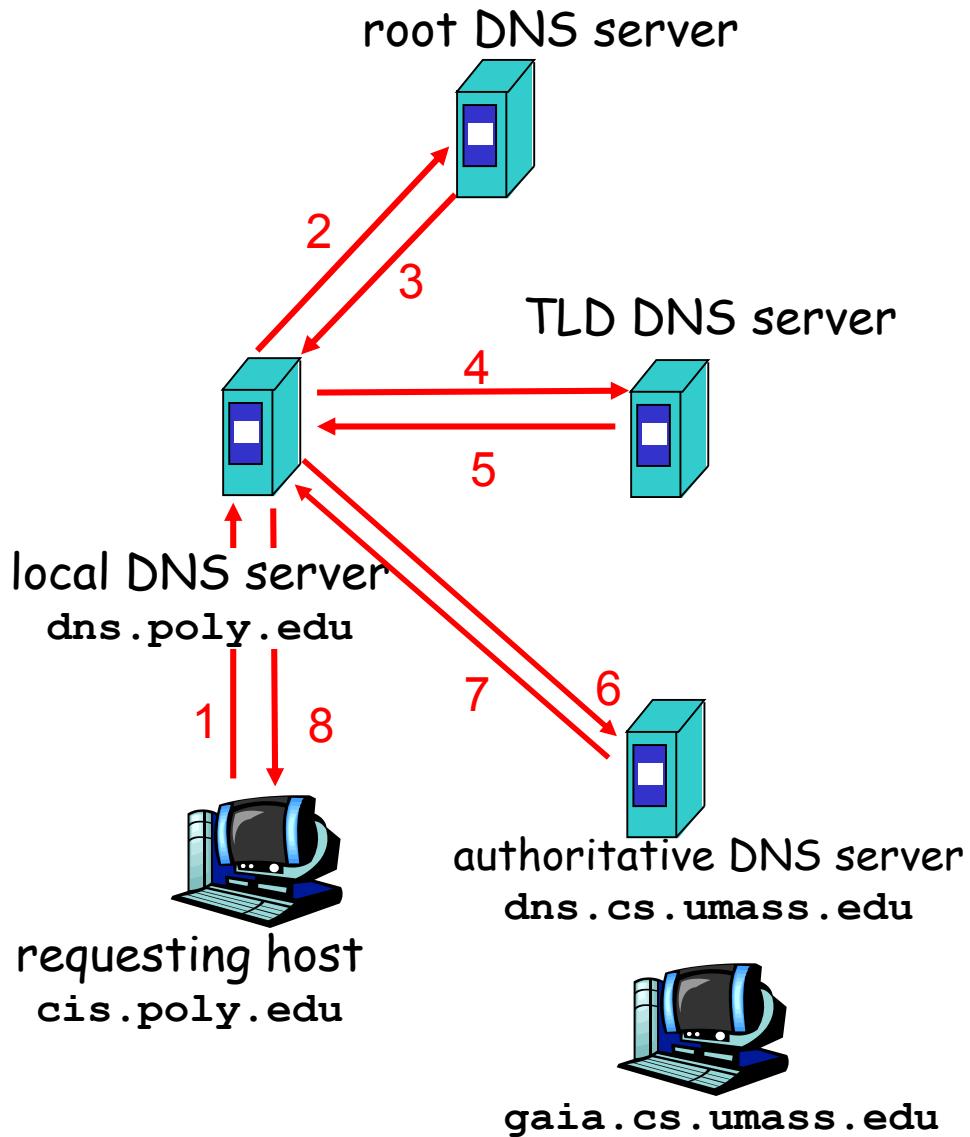
- ❑ **Top-level domain (TLD) servers:** responsible for com, org, net, edu, etc, and all top-level country domains uk, fr, ca, jp., etc.
 - ❖ Network Solutions maintains servers for com TLD
 - ❖ Educause for edu TLD
- ❑ **Authoritative DNS servers:** organization's DNS servers, providing authoritative hostname to IP mappings for organization's servers (e.g., Web and mail).
 - ❖ Can be maintained by organization or service provider

Local Name Server

- ❑ Does not strictly belong to hierarchy
- ❑ Each ISP (residential ISP, company, university) has one.
 - ❖ Also called “default name server”
- ❑ When a host makes a DNS query, query is sent to its local DNS server
 - ❖ Acts as a proxy, forwards query into hierarchy.

Example

- Host at `cis.poly.edu` wants IP address for `gaia.cs.umass.edu`



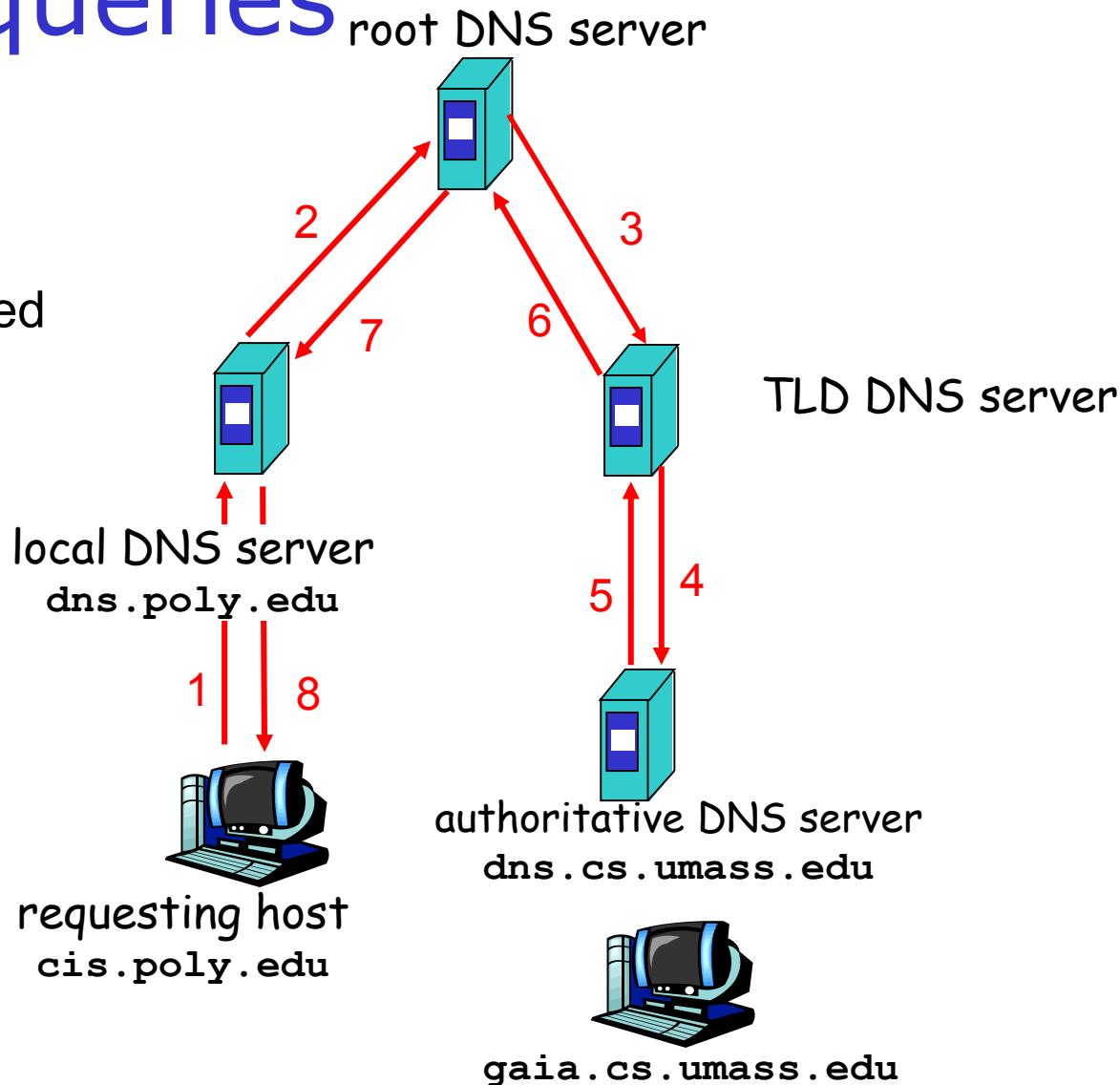
Recursive queries

recursive query:

- ❑ puts burden of name resolution on contacted name server
- ❑ heavy load?

iterated query:

- ❑ contacted server replies with name of server to contact
- ❑ “I don’t know this name, but ask this server”



DNS: caching and updating records

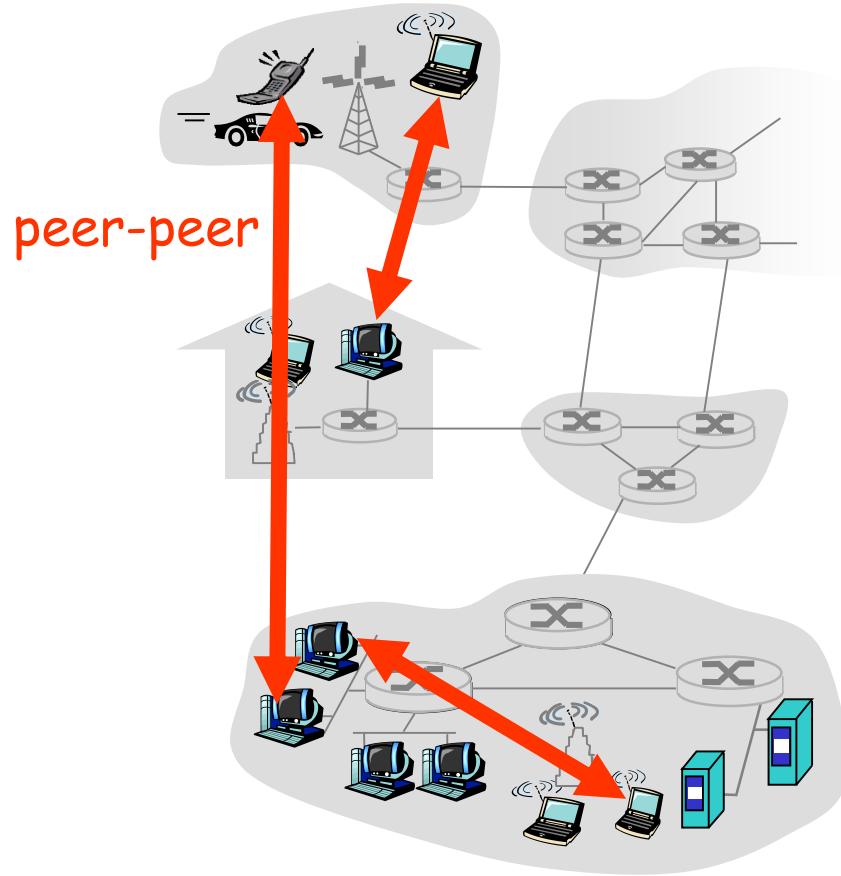
- ❑ once (any) name server learns mapping, it *caches* mapping
 - ❖ cache entries timeout (disappear) after given time
 - ❖ TLD servers typically cached in local name servers
 - Thus root name servers not often visited
- ❑ update/notify mechanisms under design by IETF
 - ❖ RFC 2136
 - ❖ <http://www.ietf.org/html.charters/dnsind-charter.html>

Chapter 2: Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 DNS
- 2.4 P2P file sharing
- 2.5 Socket programming with TCP
- 2.6 Socket programming with UDP

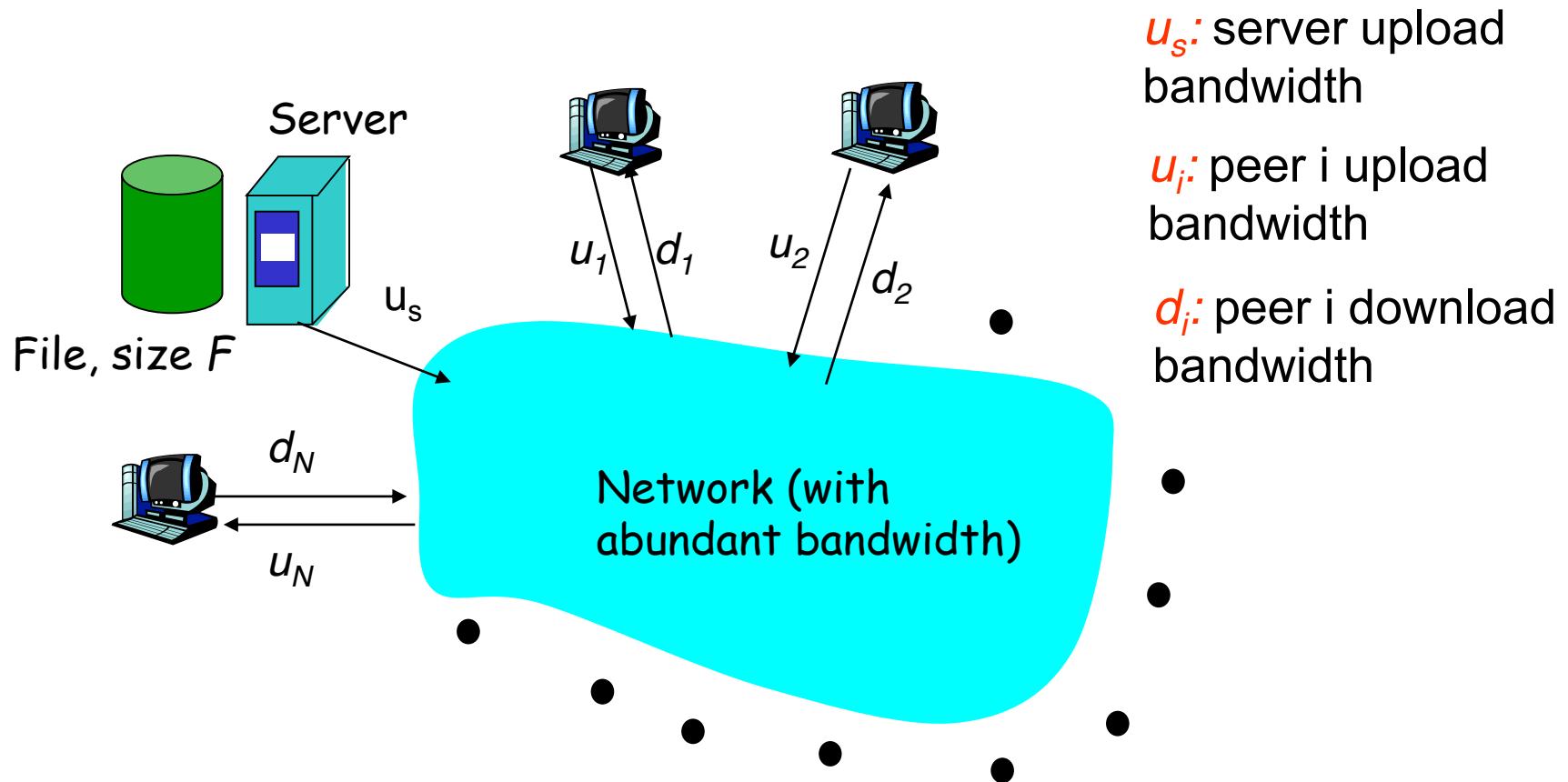
2.5 P2P architecture

- Key Characteristics
 - ❖ *no* always-on server
 - ❖ arbitrary end systems directly communicate
 - ❖ peers are intermittently connected and change IP addresses
- Key Applications
 - ❖ File Sharing
 - ❖ Messaging



2.5.1 File Sharing: Server-Client vs P2P

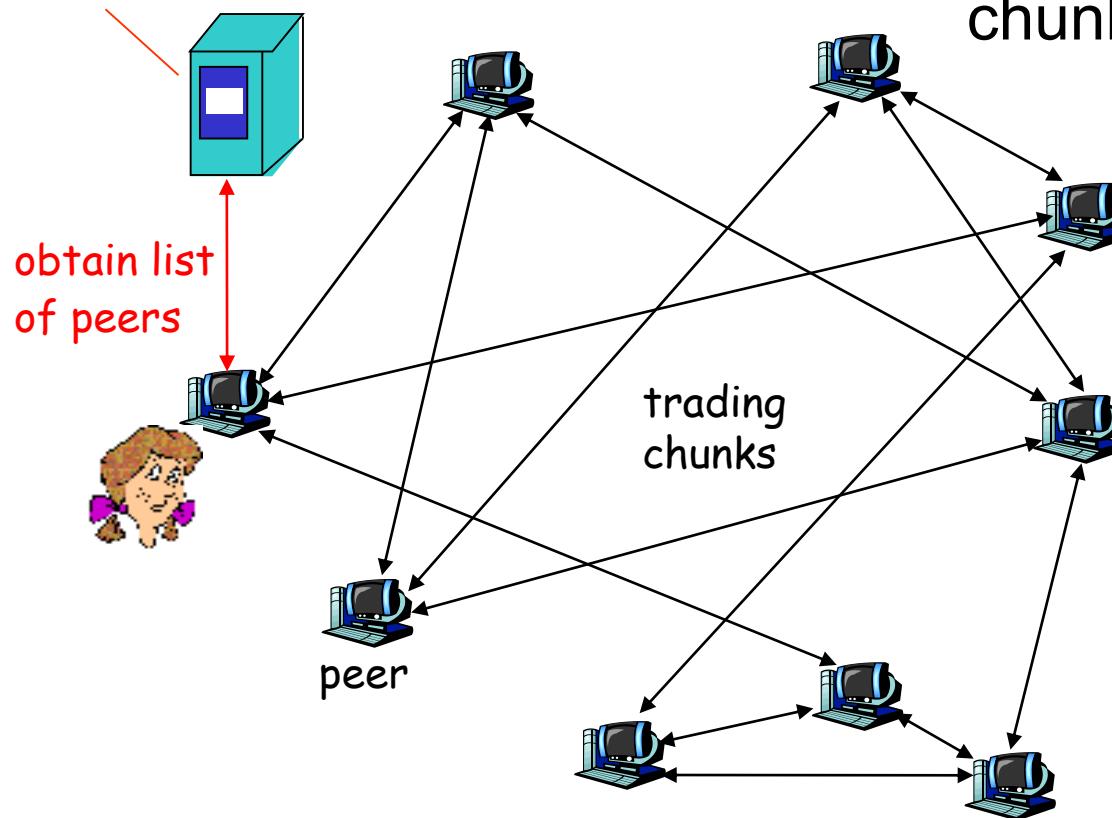
Question: How much time to distribute file from one server to N peers?



File Sharing: BitTorrent

- P2P file distribution

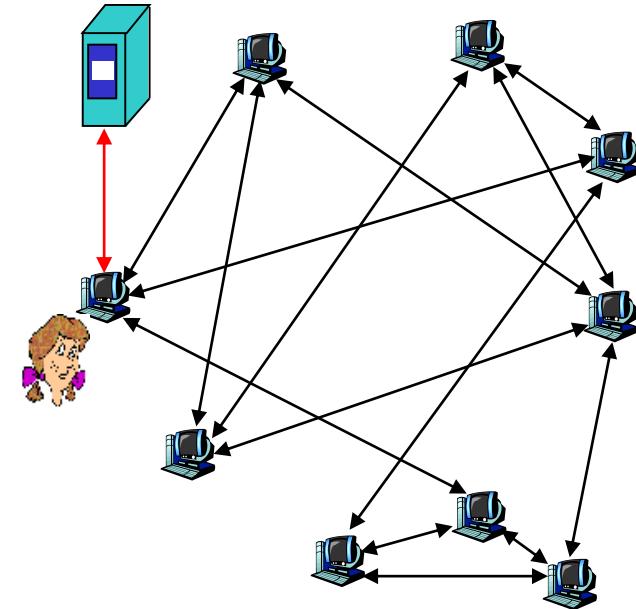
tracker: tracks peers participating in torrent



torrent: group of peers exchanging chunks of a file

BitTorrent

- ❑ file divided into 256KB *chunks*.
- ❑ peer joining torrent:
 - ❖ has no chunks, but will accumulate them over time
 - ❖ registers with tracker to get list of peers, connects to subset of peers (“neighbors”)
- ❑ while downloading, peer uploads chunks to other peers.
- ❑ peers may come and go
- ❑ once peer has entire file, it may (selfishly) leave or (altruistically) remain



BitTorrent

Pulling Chunks

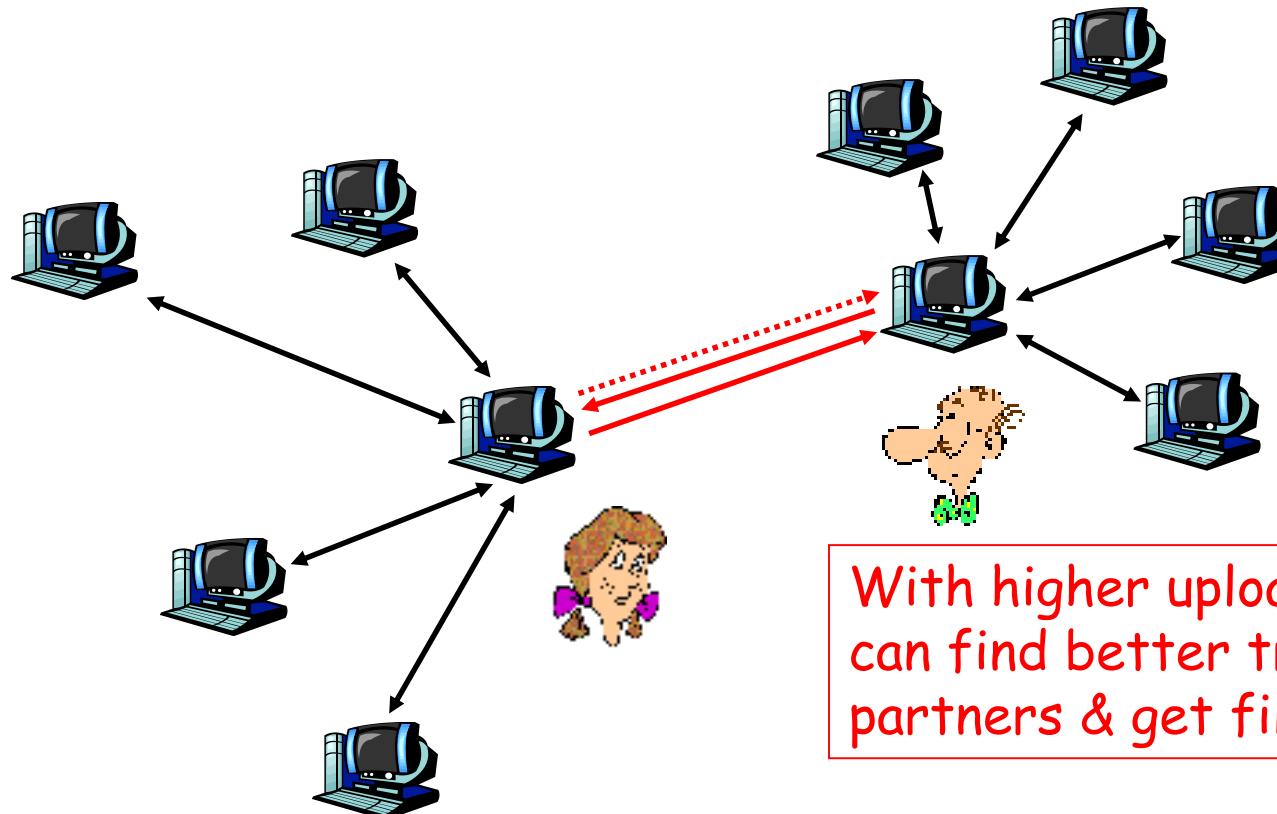
- ❑ at any given time, different peers have different subsets of file chunks
- ❑ periodically, a peer (Alice) asks each neighbor for list of chunks that they have.
- ❑ Alice sends requests for her missing chunks
 - ❖ rarest first

Sending Chunks: tit-for-tat

- ❑ Alice sends chunks to four neighbors currently sending her chunks *at the highest rate*
 - ❖ re-evaluate top 4 every 10 secs
- ❑ every 30 secs: randomly select another peer, starts sending chunks
 - ❖ newly chosen peer may join top 4
 - ❖ “optimistically unchoke”

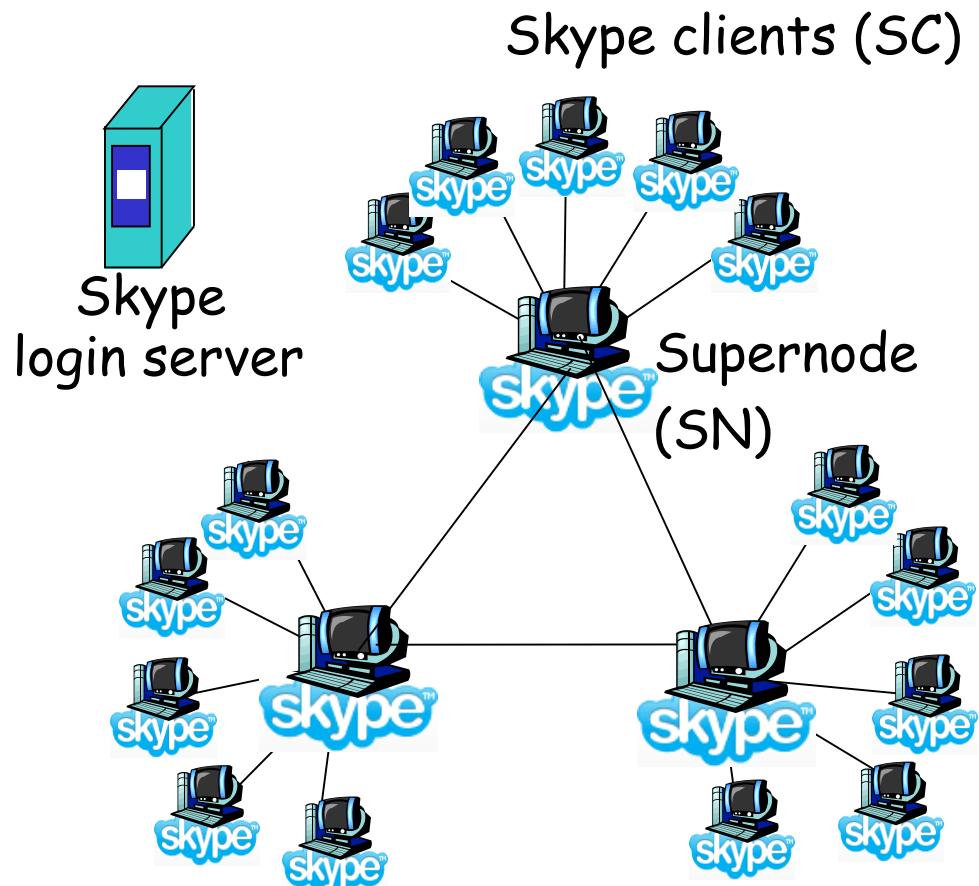
BitTorrent: Tit-for-tat

- (1) Alice “optimistically unchoke” Bob
- (2) Alice becomes one of Bob’s top-four providers; Bob reciprocates
- (3) Bob becomes one of Alice’s top-four providers



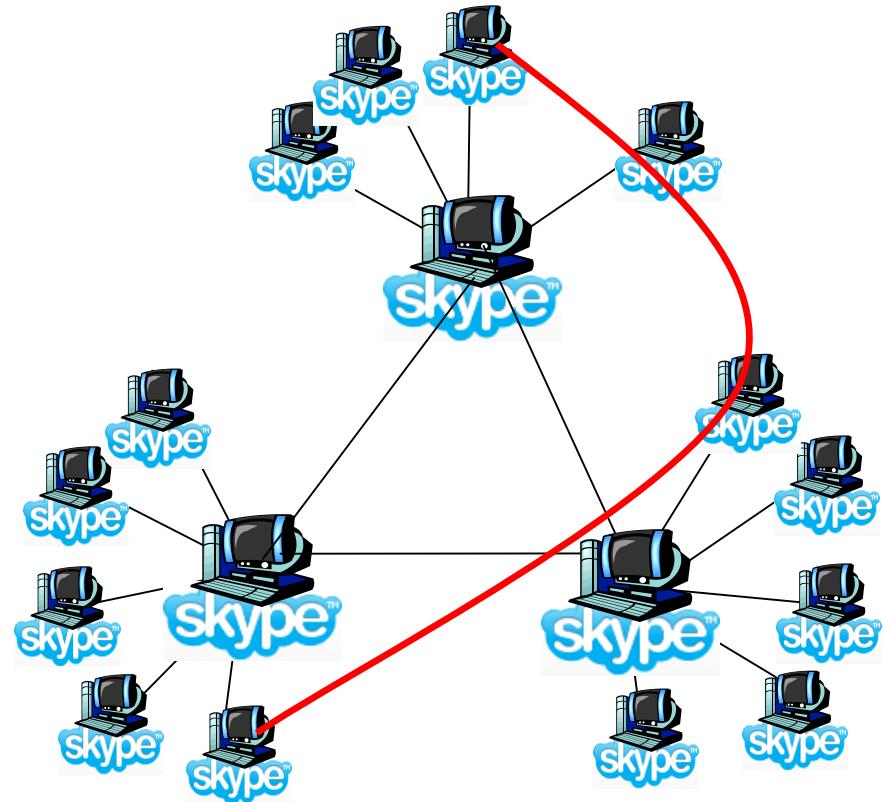
2.5.3 P2P Messaging: Skype

- ❑ inherently P2P: pairs of users communicate.
- ❑ proprietary application-layer protocol (inferred via reverse engineering)
- ❑ hierarchical overlay with SuperNodes (SNs)
- ❑ Index maps usernames to IP addresses; distributed over SNs



Peers as relays

- Problem when both Alice and Bob are behind “NATs”.
 - ❖ NAT prevents an outside peer from initiating a call to insider peer
- Solution:
 - ❖ Using Alice’s and Bob’s SNs, Relay is chosen
 - ❖ Each peer initiates session with relay.
 - ❖ Peers can now communicate through NATs via relay



Chapter 2: Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 DNS
- 2.4 P2P file sharing
- **2.5 Socket programming with TCP**
- 2.6 Socket programming with UDP

2.6 Socket programming

Goal: learn how to build client/server application that communicate using sockets

Socket API

- ❑ introduced in BSD4.1 UNIX, 1981
- ❑ explicitly created, used, released by apps
- ❑ client/server paradigm
- ❑ two types of transport service via socket API:
 - ❖ unreliable datagram
 - ❖ reliable, byte stream-oriented

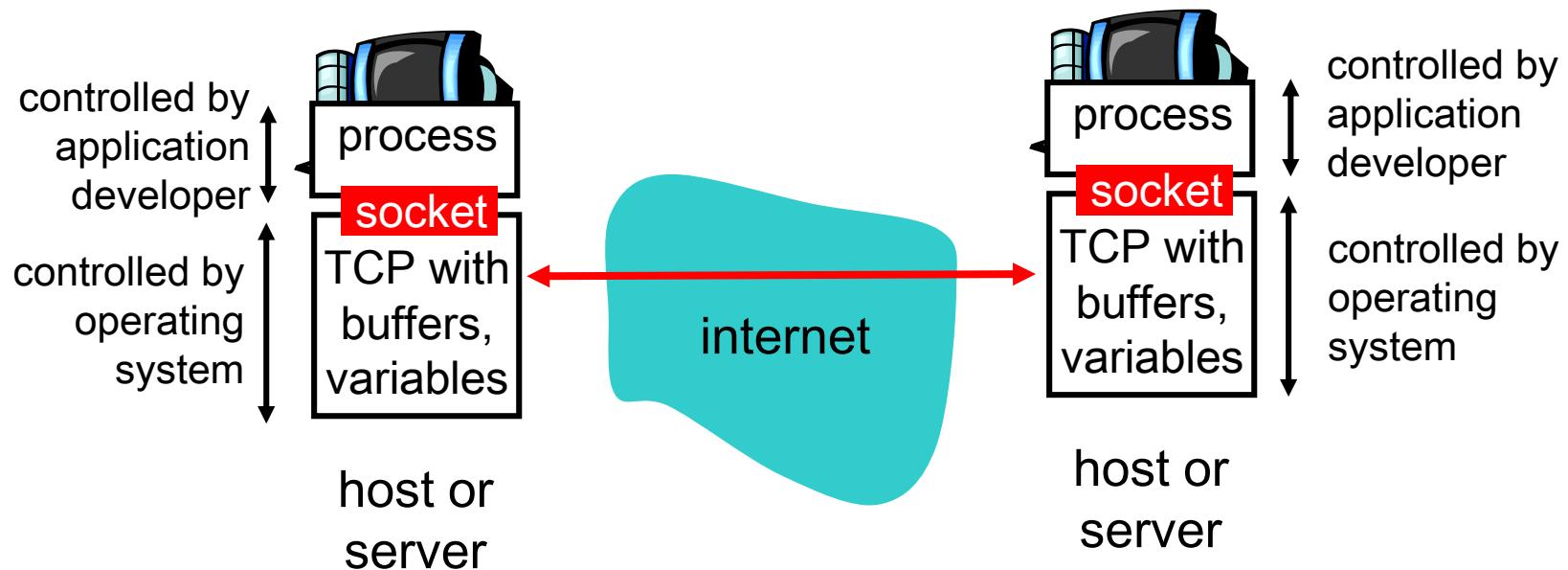
socket

a *host-local, application-created, OS-controlled* interface (a “door”) into which application process can **both send and receive** messages to/from another application process

Socket-programming using TCP

Socket: a door between application process and end-end-transport protocol (UCP or TCP)

TCP service: reliable transfer of **bytes** from one process to another



Socket programming *with TCP*

Client must contact server

- ❑ server process must first be running
- ❑ server must have created socket (door) that welcomes client's contact

Client contacts server by:

- ❑ creating client-local TCP socket
- ❑ specifying IP address, port number of server process
- ❑ When **client creates socket**: client TCP establishes connection to server TCP

- ❑ When contacted by client, **server TCP creates new socket** for server process to communicate with client
 - ❖ allows server to talk with multiple clients
 - ❖ source port numbers used to distinguish clients (**more in Chap 3**)

application viewpoint

TCP provides reliable, in-order transfer of bytes (“pipe”) between client and server

Client/server socket interaction: TCP

Server (running on hostid)

```
create socket,  
port=x, for  
incoming request:  
welcomeSocket =  
    ServerSocket()
```

```
wait for incoming  
connection request  
connectionSocket =  
    welcomeSocket.accept()
```

```
read request from  
connectionSocket
```

```
write reply to  
connectionSocket
```

```
close  
connectionSocket
```

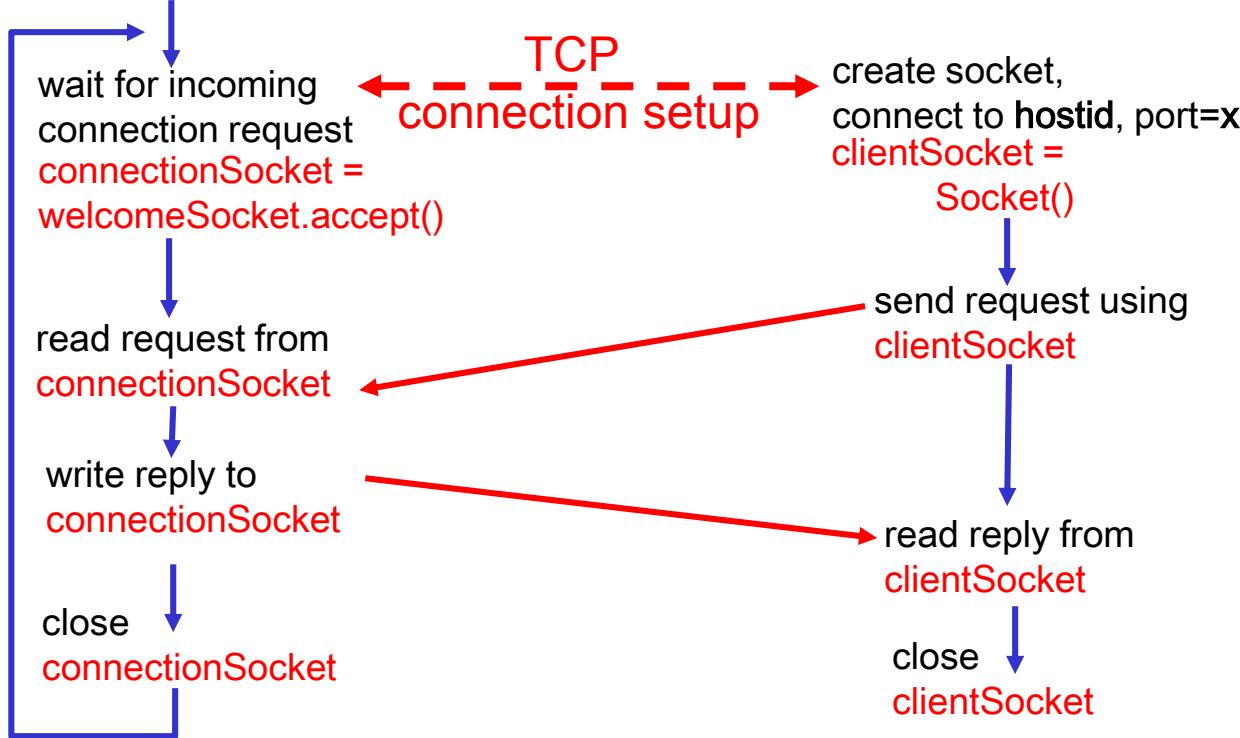
Client

```
create socket,  
connect to hostid, port=x  
clientSocket =  
    Socket()
```

```
send request using  
clientSocket
```

```
read reply from  
clientSocket  
close  
clientSocket
```

TCP
connection setup



Chapter 2: Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 DNS
- 2.4 P2P file sharing
- 2.5 Socket programming with TCP
- 2.6 Socket programming with UDP

2.7 Socket programming *with UDP*

UDP: no “connection” between client and server

- ❑ no handshaking
- ❑ sender explicitly attaches IP address and port of destination to each packet
- ❑ server must extract IP address, port of sender from received packet

UDP: transmitted data may be received out of order, or lost

application viewpoint

UDP provides unreliable transfer of groups of bytes (“datagrams”) between client and server

Client/server socket interaction: UDP

Server (running on `hostid`)

```
create socket,  
port=x, for  
incoming request:  
serverSocket =  
DatagramSocket()
```

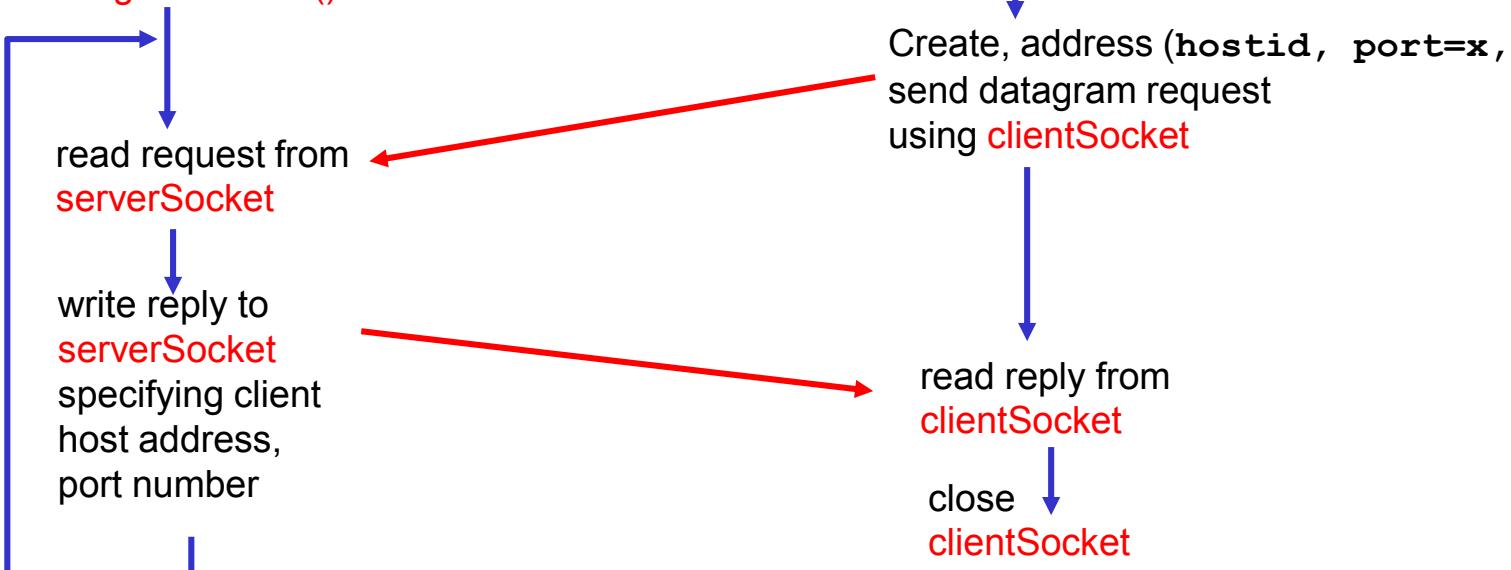
```
read request from  
serverSocket  
  
write reply to  
serverSocket  
specifying client  
host address,  
port number
```

Client

```
create socket,  
clientSocket =  
DatagramSocket()
```

```
Create, address (hostid, port=x,  
send datagram request  
using clientSocket
```

```
read reply from  
clientSocket  
  
close  
clientSocket
```



Example: UDP on Python

Lets look at a simple example you can use on your own to try out UDP. TCP is very similar, just a bit more complicated.

Python is becoming a very popular object orientated interpreted programming language, built on C++ but with modern features (e.g. lists, tuples, queues, threads, etc).

Python (v2) comes as standard on Linux and can be obtained for free for Windows.

Consider two connected devices, running Linux. A sender device sends a UDP packet every second, the responder device responds a suitable message to the original sent message.

See *sender.py* on Blackboard

See *responder.py* on Blackboard

Try it for yourself:

1. Run Linux on 2 lab PC's
2. Load a terminal on each PC
3. To see your IP address type "ip addr show"
4. Edit the Python scripts adding your IP address.
5. Run the *sender.py* script by typing "python *sender.py*"
6. On another PC run the *responder.py* script

Chapter 2: Summary

Our study of network apps now complete!

- ❑ Application architectures
 - ❖ client-server
 - ❖ P2P
 - ❖ hybrid
- ❑ application service requirements:
 - ❖ reliability, bandwidth, delay
- ❑ Internet transport service model
 - ❖ connection-oriented, reliable: TCP
 - ❖ unreliable, datagrams: UDP
- ❑ specific protocols:
 - ❖ HTTP
 - ❖ DNS
- ❑ socket programming

Chapter 2: Summary

Most importantly: learned about *protocols*

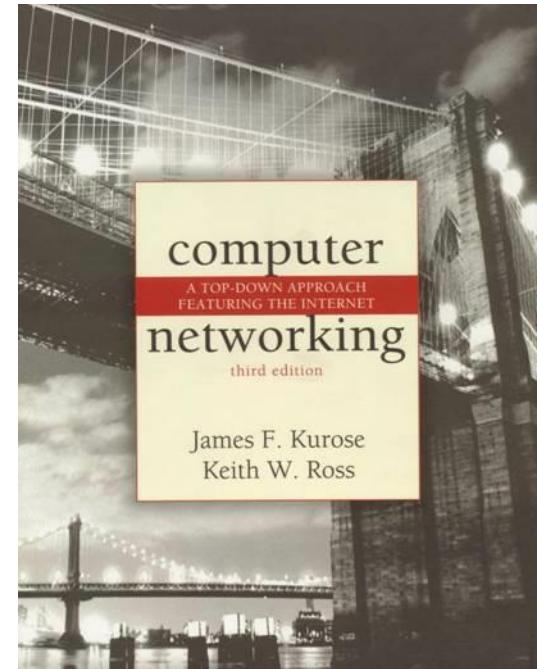
- ❑ **typical request/reply message exchange:**
 - ❖ client requests info or service
 - ❖ server responds with data, status code
- ❑ **message formats:**
 - ❖ headers: fields giving info about data
 - ❖ data: info being communicated
- ❑ control vs. data msgs
 - ❑ in-band, out-of-band
- ❑ centralized vs. decentralized
- ❑ stateless vs. stateful
- ❑ reliable vs. unreliable msg transfer
- ❑ “complexity at network edge”

Computer Networks

Chapter 3 Transport Layer

Professor R Simon Sherratt

Adapted and updated from the original
work of Dr. Michael Evans



*Computer
Networking: A Top
Down Approach
Featuring the
Internet,
Jim Kurose, Keith
Ross
Addison-Wesley*

Chapter 3: Transport Layer

Our goals:

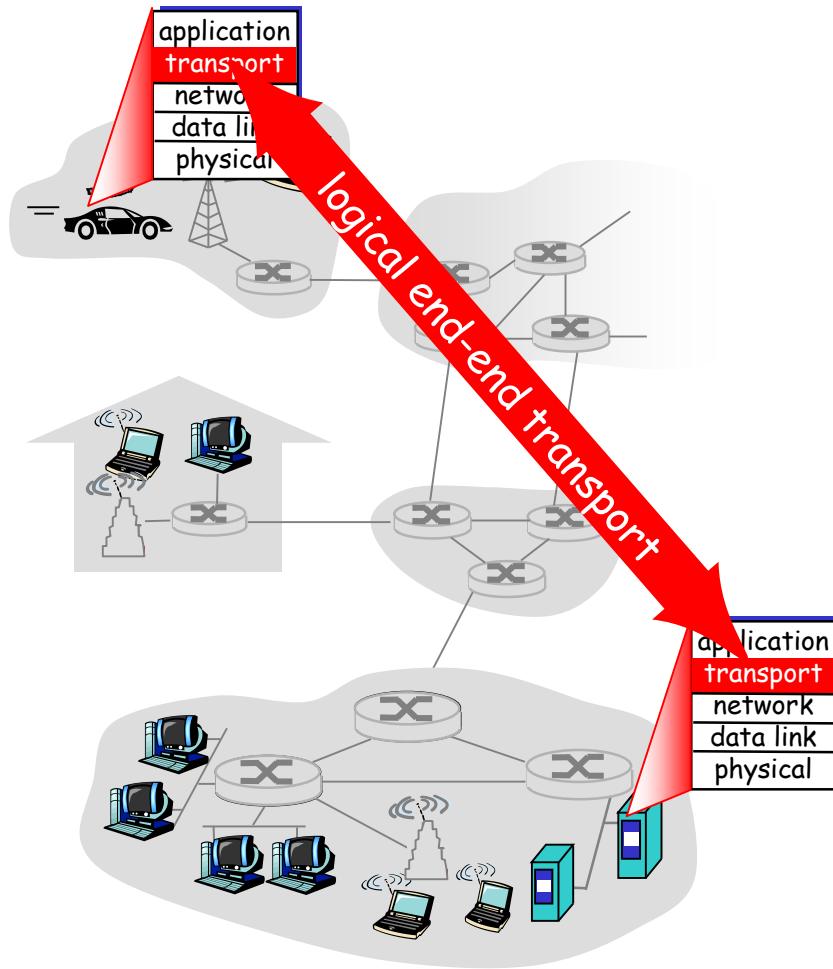
- Understand principles behind transport layer services:
 - multiplexing/demultiplexing
 - reliable data transfer
 - flow control
 - congestion control
- Learn about transport layer protocols in the Internet:
 - UDP: connectionless transport
 - TCP: connection-oriented transport
 - TCP congestion control

Chapter 3 outline

- 3.1 Transport-layer services and Protocols
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Connection-oriented transport: TCP
 - segment structure
 - reliable data transfer
 - flow control
 - connection management
 - TCP congestion control
- 3.5 Multimedia over TCP and UDP
 - Multimedia Networking Applications
 - Streaming stored multimedia over TCP and UDP
 - UDP or TCP
 - HTTP Streaming
 - Content Distribution Networks

3.1 Transport services and protocols

- ❑ provide *logical communication* between app processes running on different hosts
- ❑ transport protocols run in end systems
 - send side: breaks app messages into **segments**, passes to network layer
 - receive side: reassembles segments into messages, passes to app layer
- ❑ more than one transport protocol available to apps
 - Internet: TCP and UDP



Transport vs. Network layer

- *Network layer:*

- Logical point-to-point communication between hosts

- *Transport layer:*

- Logical end-to-end communication between application processes
 - Relies on and enhances network layer services

Internet transport-layer protocols

□ TCP

- Reliable, in-order delivery
- Congestion control
- Flow control
- Connection setup

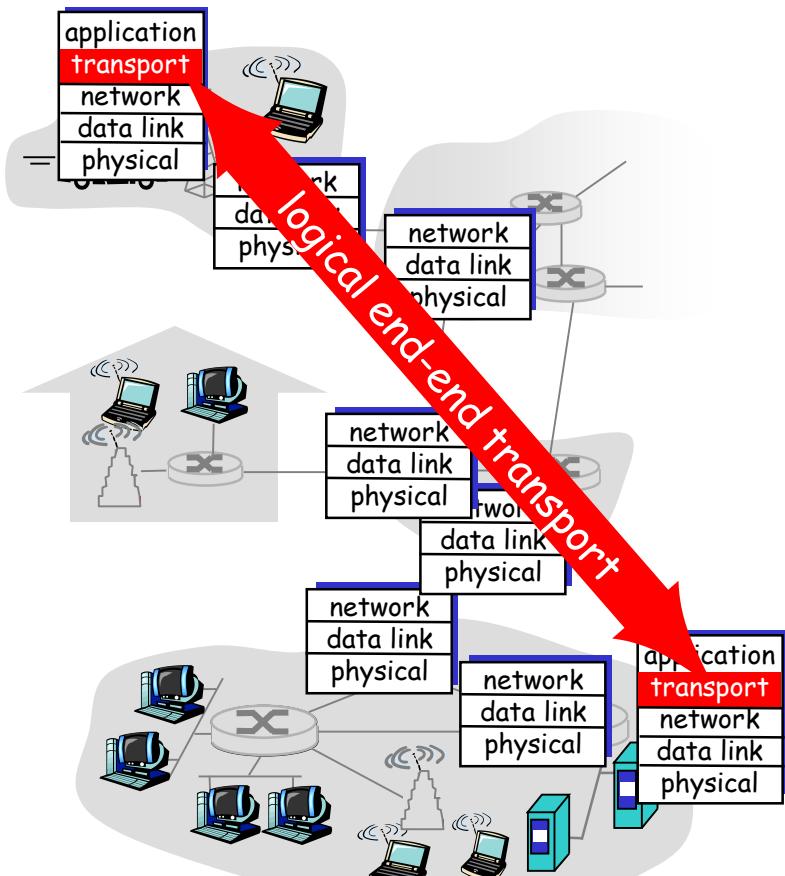
□ UDP

- Unreliable, unordered delivery:
- No-frills extension of “best-effort” IP

□ Services NOT available:

- Delay guarantees
- Bandwidth guarantees

□ **Typical mistake – TCP is still best effort as Internet is best effort**



Chapter 3 outline

- 3.1 Transport-layer services and Protocols
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Connection-oriented transport: TCP
 - segment structure
 - reliable data transfer
 - flow control
 - connection management
 - TCP congestion control
- 3.5 Multimedia over TCP and UDP
 - Multimedia Networking Applications
 - Streaming stored multimedia over TCP and UDP
 - UDP or TCP
 - HTTP Streaming
 - Content Distribution Networks

3.2 Multiplexing/demultiplexing

- Provides process-to-process delivery service for apps running on hosts
- Solves problem of **different apps** communicating on **same host** through **one network connection**
- Example
 - Web client, FTP client, Email client all communicate simultaneously
 - When new packet is received, which application should it go to?
 - Needs to be demultiplexed
- Demultiplexing service needed for all computer networks
 - Not just the Internet

Multiplexing/demultiplexing

Demultiplexing at rcv host:

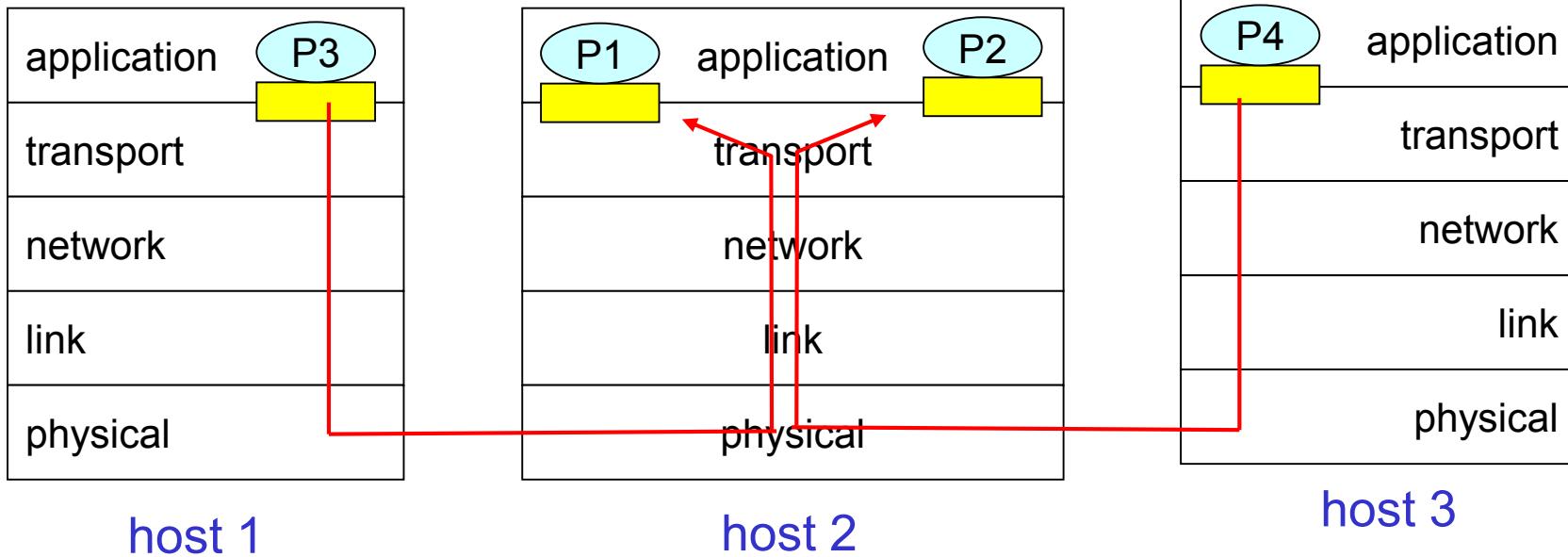
Delivering received segments
to correct socket

Multiplexing at send host:

Gathering data from multiple
sockets, enveloping data with
header (later used for
demultiplexing)

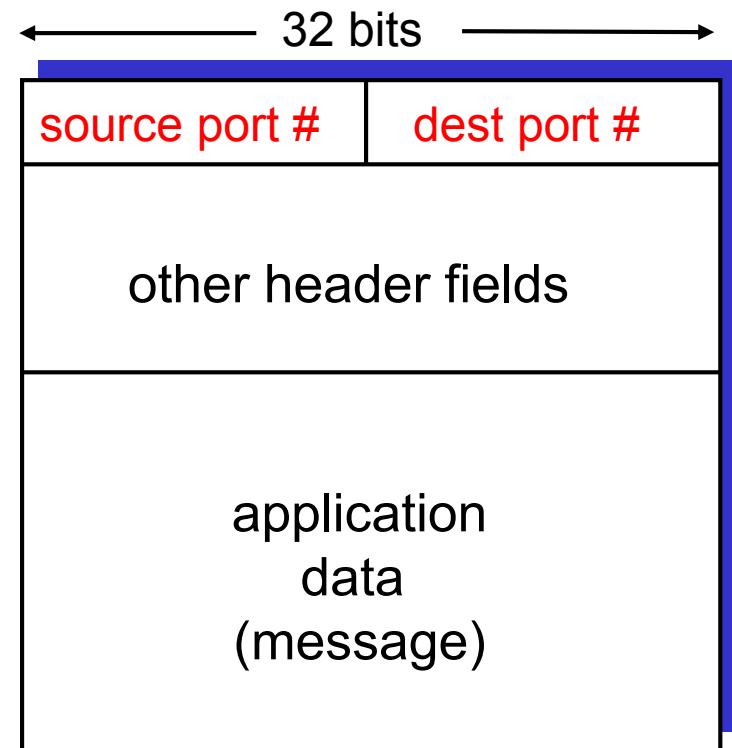
= socket

= process



How Demultiplexing works

- Host receives IP datagrams
 - Each datagram has source IP address, destination IP address
 - Each datagram carries 1 transport-layer segment
 - Each segment has source, destination port number
- Host uses IP addresses & port numbers to direct segment to appropriate socket

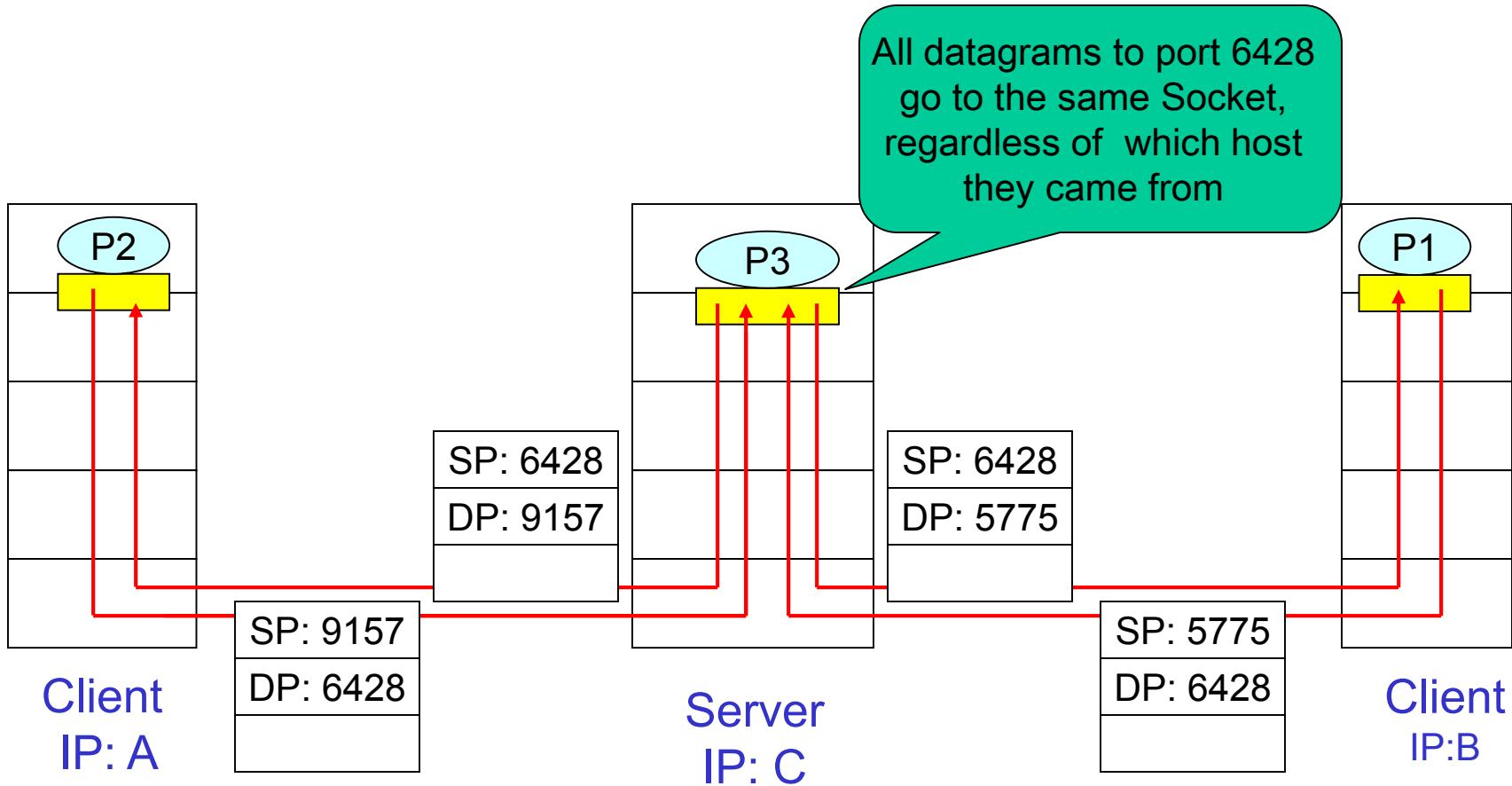


TCP/UDP segment format

3.2.1 Connectionless Demultiplexing

- Create sockets with port numbers
- UDP socket identified by two-tuple:
(dest IP address, dest port number)
- When host receives UDP segment:
 - checks destination port number in segment
 - directs UDP segment to socket with that port number
- IP datagrams with different source IP addresses and/or source port numbers are all directed to **same** socket on receiver
- Different applications can listen to the same port

Connectionless demux (cont)



 Socket

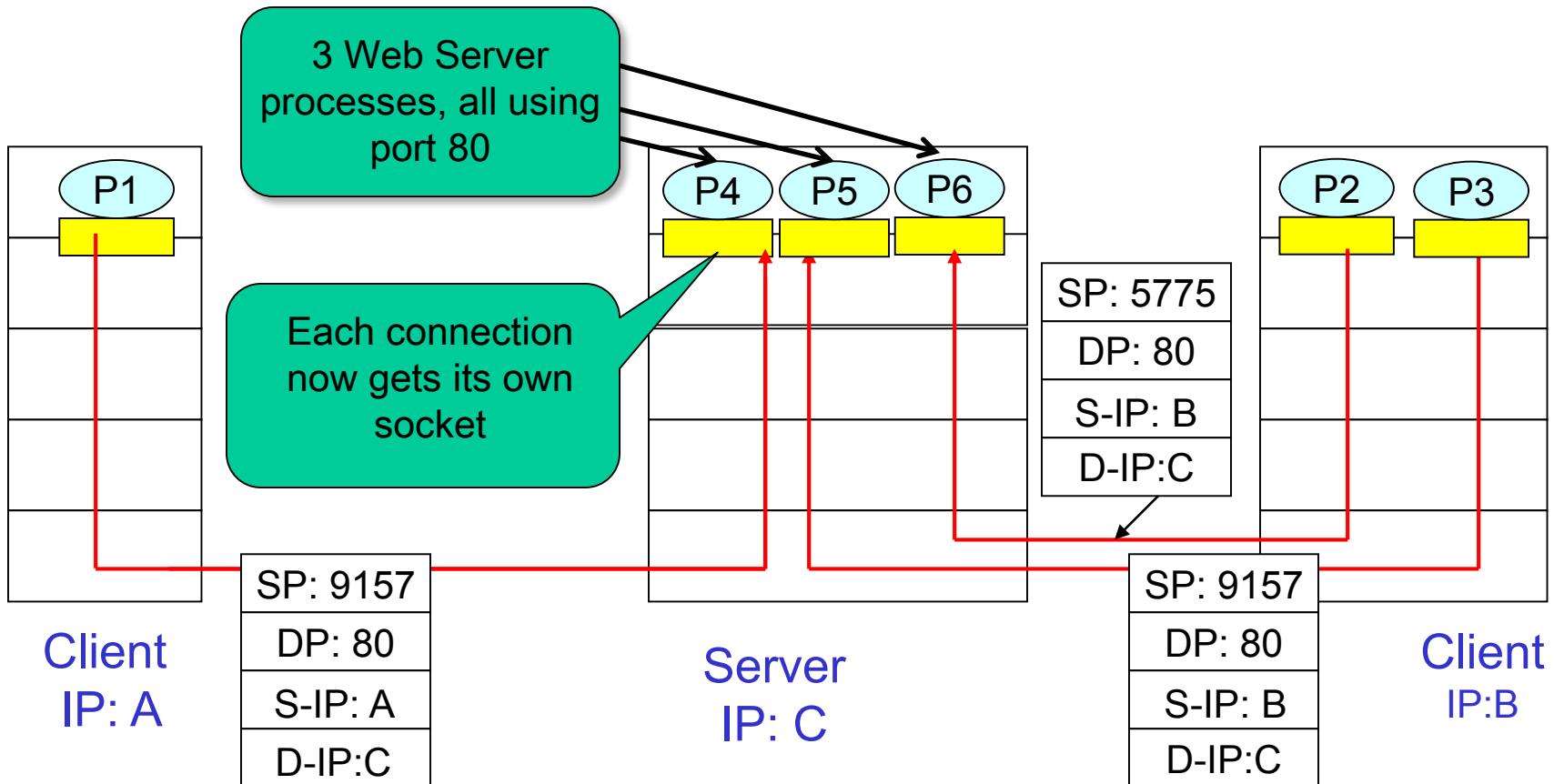
SP = Source Port
DP = Destination Port

SP provides “return address”

3.2.2 Connection-oriented demux

- TCP socket identified by 4-tuple:
 - source IP address
 - source port number
 - dest IP address
 - dest port number
- Recv host uses all four values to direct segment to appropriate socket
- Only one application can listen to a single port at a time
- Server host may support many simultaneous TCP 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 – Web Server example



Chapter 3 outline

- 3.1 Transport-layer services and Protocols
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Connection-oriented transport: TCP
 - segment structure
 - reliable data transfer
 - flow control
 - connection management
 - TCP congestion control
- 3.5 Multimedia over TCP and UDP
 - Multimedia Networking Applications
 - Streaming stored multimedia over TCP and UDP
 - UDP or TCP
 - HTTP Streaming
 - Content Distribution Networks

3.3 UDP: User Datagram Protocol

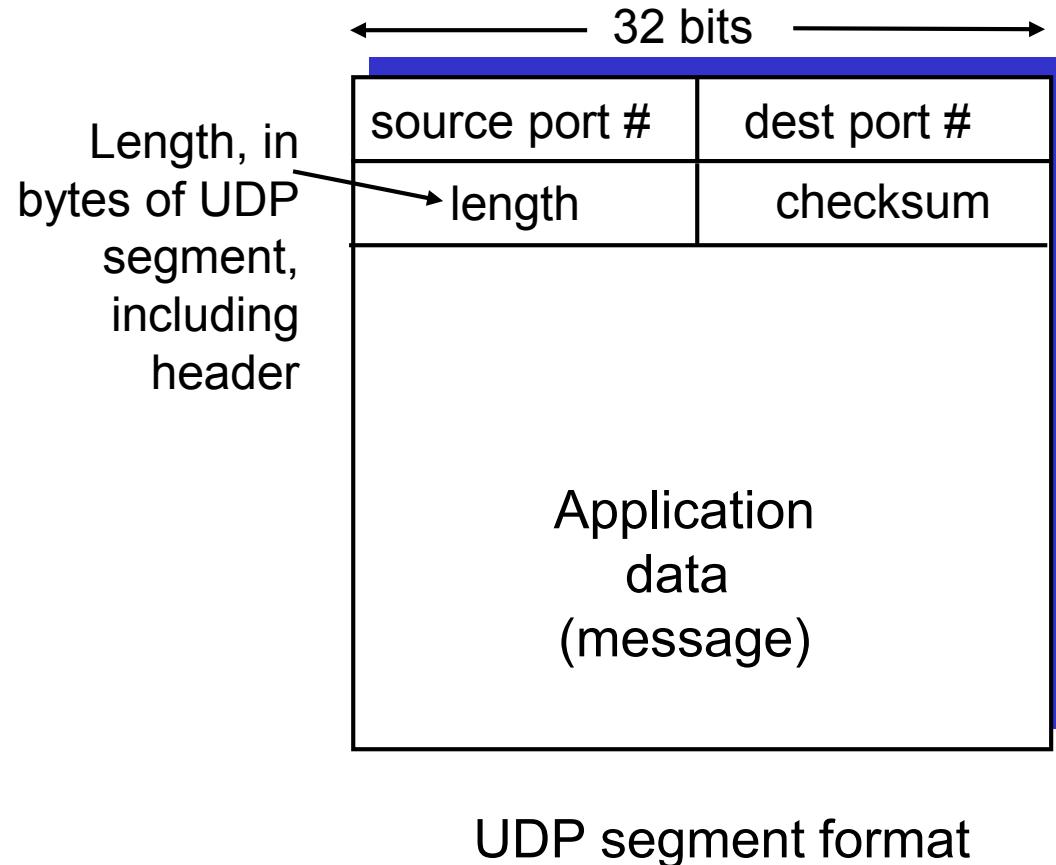
RFC 768

- ❑ “No frills,” “bare bones” Internet transport protocol
 - ❑ “Best effort” service, UDP segments may be:
 - lost
 - delivered out of order to app
- ❑ *Connectionless:***
- No handshaking between UDP sender, receiver
 - Each UDP segment handled independently of others

Why is there a UDP?

- ❑ No connection establishment (which can add delay)
- ❑ Simple:
 - no connection state at sender, receiver
 - small segment header
- ❑ No congestion control: UDP can blast away as fast as desired

UDP Segment Structure



- UDP often used for streaming multimedia apps
 - loss tolerant
 - rate sensitive
- Other UDP uses
 - DNS
 - SNMP
- For reliable transfer over UDP: add reliability at application layer
 - application-specific error recovery!

UDP of no use for most apps

- ❑ **UDP is connectionless and unreliable**
- ❑ IP, the network protocol beneath UDP is also unreliable
- ❑ Will UDP messages actually reach receiver?
 - They might, they might not!
- ❑ Most apps need reliability
 - Can't change IP, so reliable data transfer needed at transport layer
- ❑ **Step forward TCP!**

Chapter 3 outline

- 3.1 Transport-layer services and Protocols
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Connection-oriented transport: TCP
 - segment structure
 - reliable data transfer
 - flow control
 - connection management
 - TCP congestion control
- 3.5 Multimedia over TCP and UDP
 - Multimedia Networking Applications
 - Streaming stored multimedia over TCP and UDP
 - UDP or TCP
 - HTTP Streaming
 - Content Distribution Networks

3.4 TCP: Overview

RFCs: 793, 1122, 1323, 2018, 2581

- **point-to-point:**

- one sender, one receiver

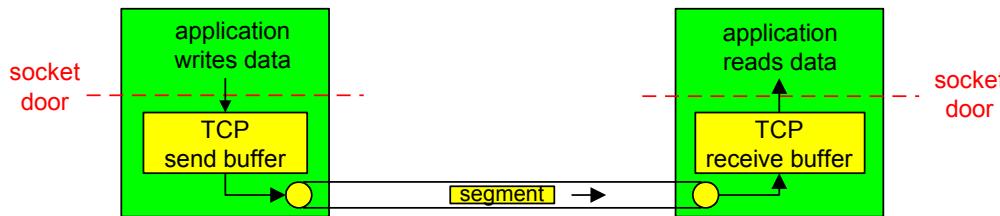
- **reliable, in-order *byte steam*:**

- no “message boundaries”

- **pipelined:**

- TCP congestion and flow control set window size

- ***send & receive buffers***



- **full duplex data:**

- bi-directional data flow in same connection
 - MSS: maximum segment size

- **connection-oriented:**

- handshaking (exchange of control msgs) init's sender, receiver state before data exchange

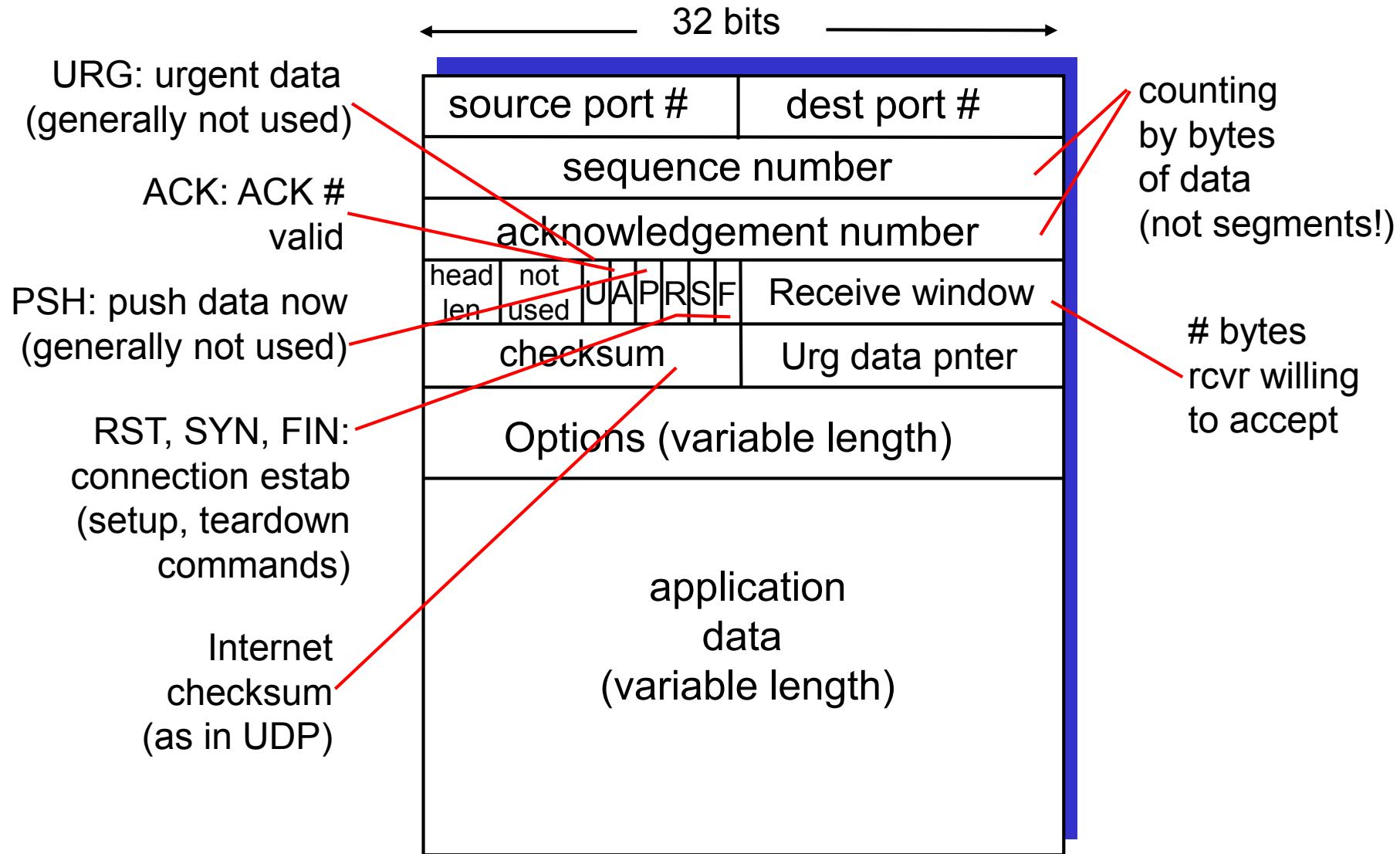
- **flow controlled:**

- sender will not overwhelm receiver

Chapter 3 outline

- 3.1 Transport-layer services and Protocols
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Connection-oriented transport: TCP
 - segment structure
 - reliable data transfer
 - flow control
 - connection management
 - TCP congestion control
- 3.5 Multimedia over TCP and UDP
 - Multimedia Networking Applications
 - Streaming stored multimedia over TCP and UDP
 - UDP or TCP
 - HTTP Streaming
 - Content Distribution Networks

3.4.1 TCP segment structure

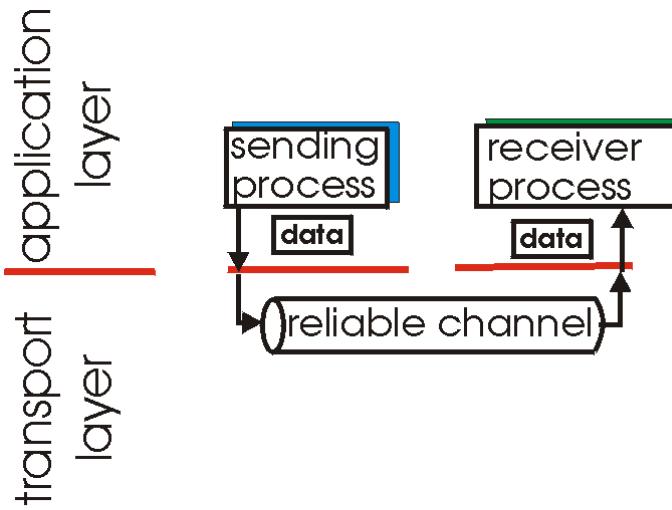


Chapter 3 outline

- 3.1 Transport-layer services and Protocols
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Connection-oriented transport: TCP
 - segment structure
 - **reliable data transfer**
 - flow control
 - connection management
 - TCP congestion control
- 3.5 Multimedia over TCP and UDP
 - Multimedia Networking Applications
 - Streaming stored multimedia over TCP and UDP
 - UDP or TCP
 - HTTP Streaming
 - Content Distribution Networks

3.4.2 Reliable data transfer

- ❑ Most applications require a reliable transport, link layer
- ❑ top-10 list of important networking topics!

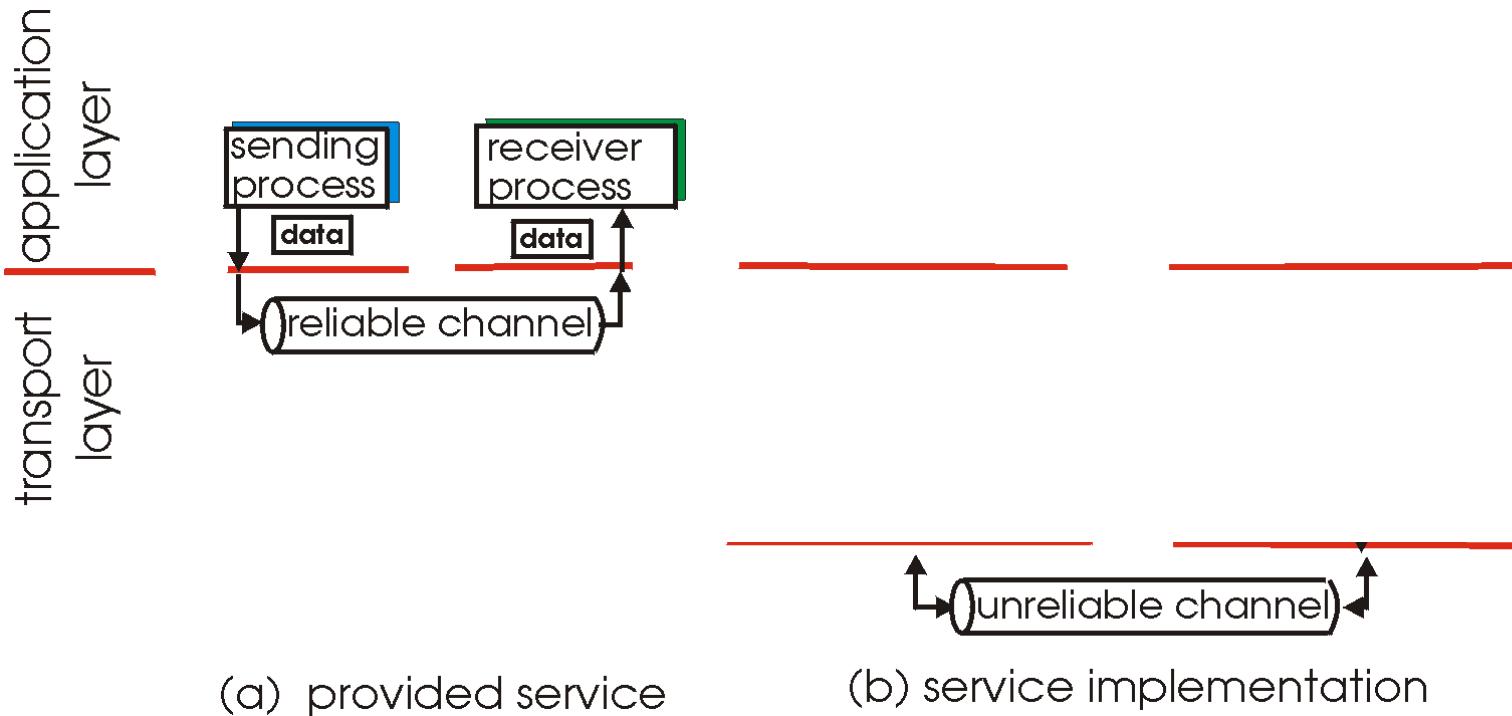


(a) provided service

- ❑ characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

3.4.2 Reliable data transfer

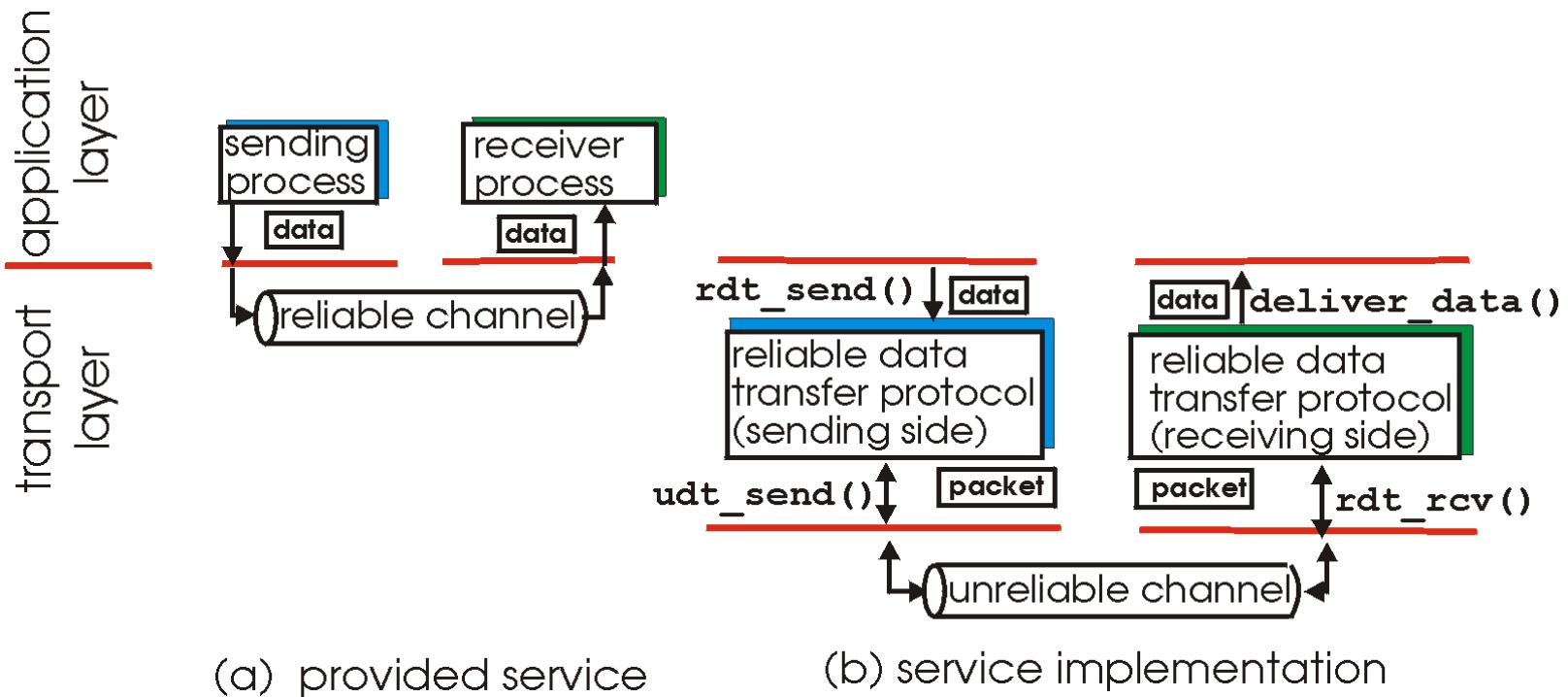
- ❑ Most applications require a reliable transport, link layer
- ❑ top-10 list of important networking topics!



- ❑ characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

3.4.2 Reliable data transfer

- ❑ Most applications require a reliable transport, link layer
- ❑ top-10 list of important networking topics!

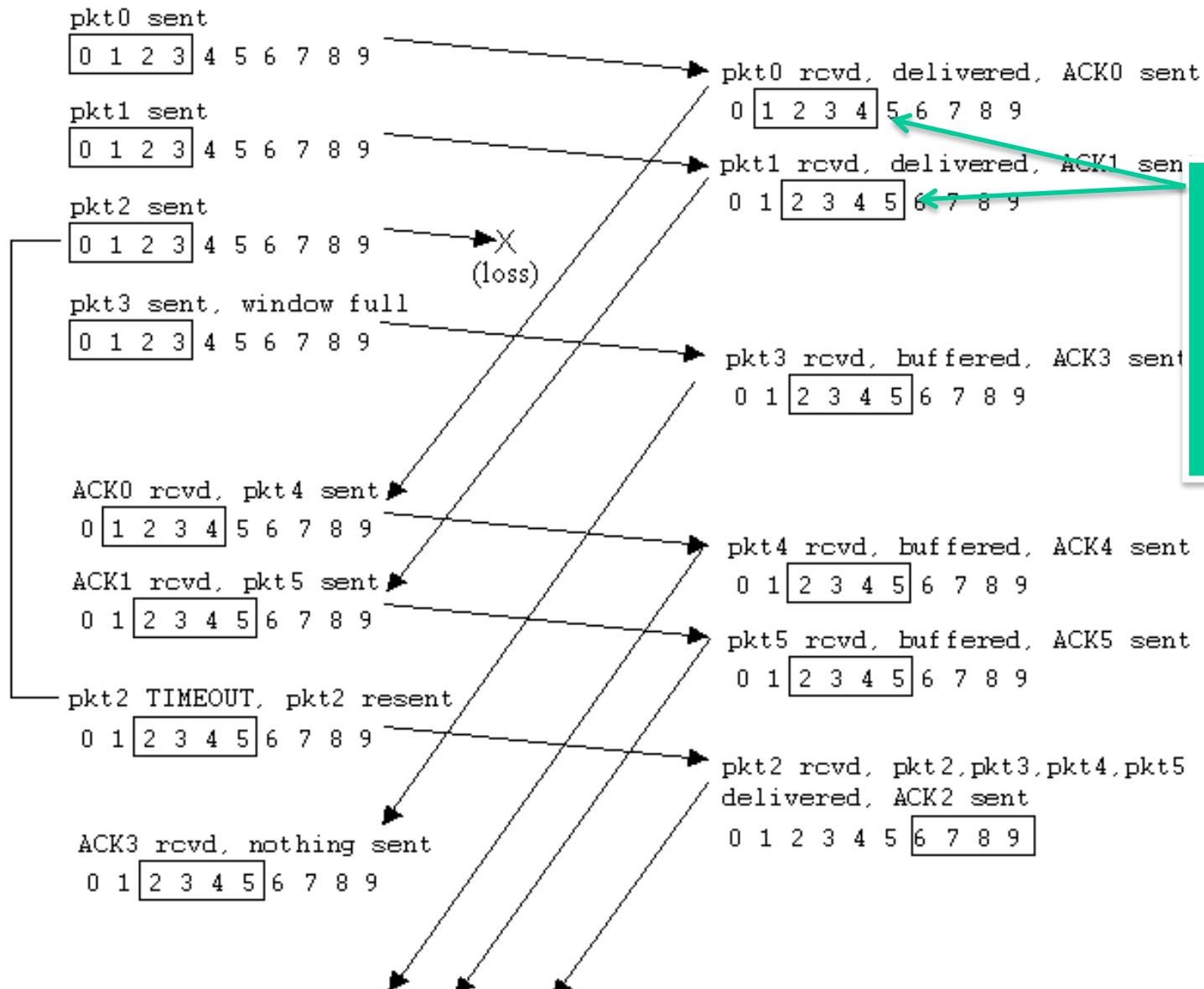


- ❑ characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

3.4.2.1 Automatic Repeat Request (ARQ)

- Any reliable protocol must handle messages that are:
 - Lost
 - Delayed (and thus duplicated if retransmitted)
 - Corrupted
- Reliability handled through Sliding Windows and ARQ mechanisms
- Types of ARQ:
 - *Error Detection*
 - *Positive ACK*
 - ACK successfully received frames
 - *Retransmission after timeout*
 - *Negative Acknowledgement (NAK)*
 - Destination sends explicit message saying “didn’t receive frame number n ”

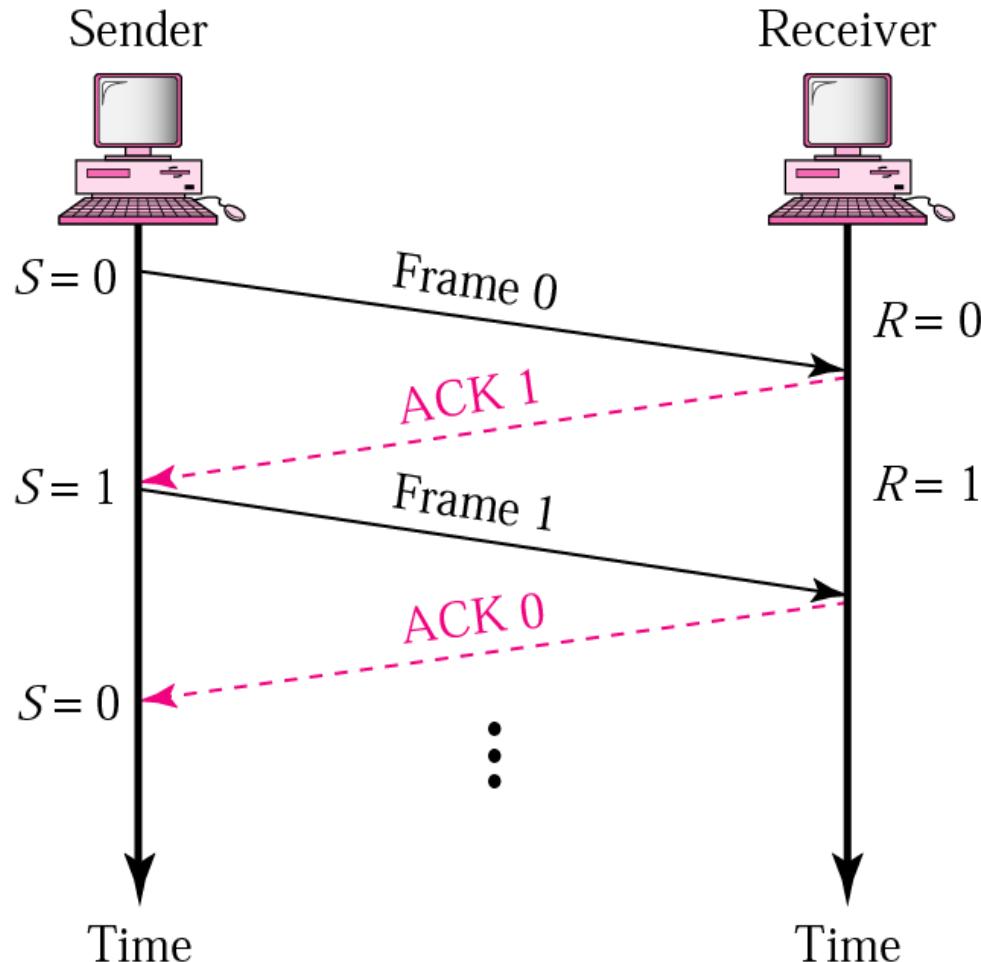
Example of Sliding Windows



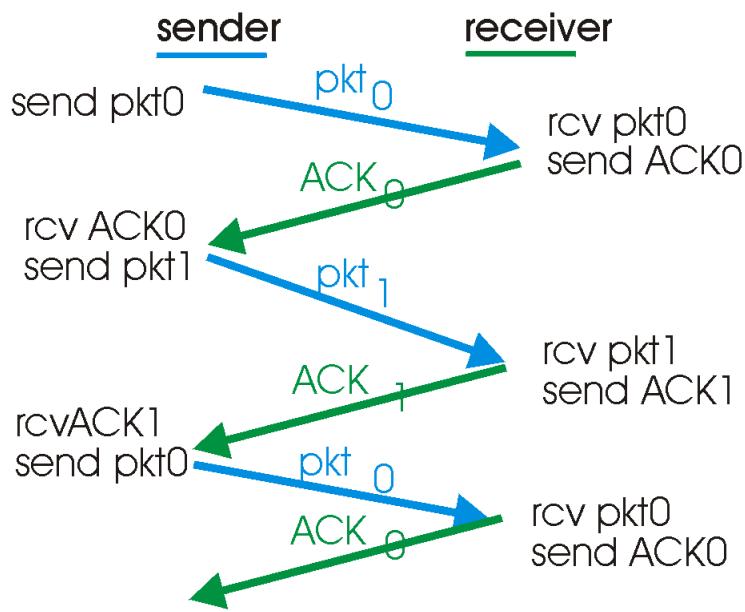
3.4.2.2 Stop-and-Wait

- Simplest form of flow control and error control
 - Sender, sends a message
 - Receiver, sends an ACKnowledgement upon successful receipt
 - Sender, must wait for ACK before transmitting next message
 - ACK contains sequence number of **next** message expected

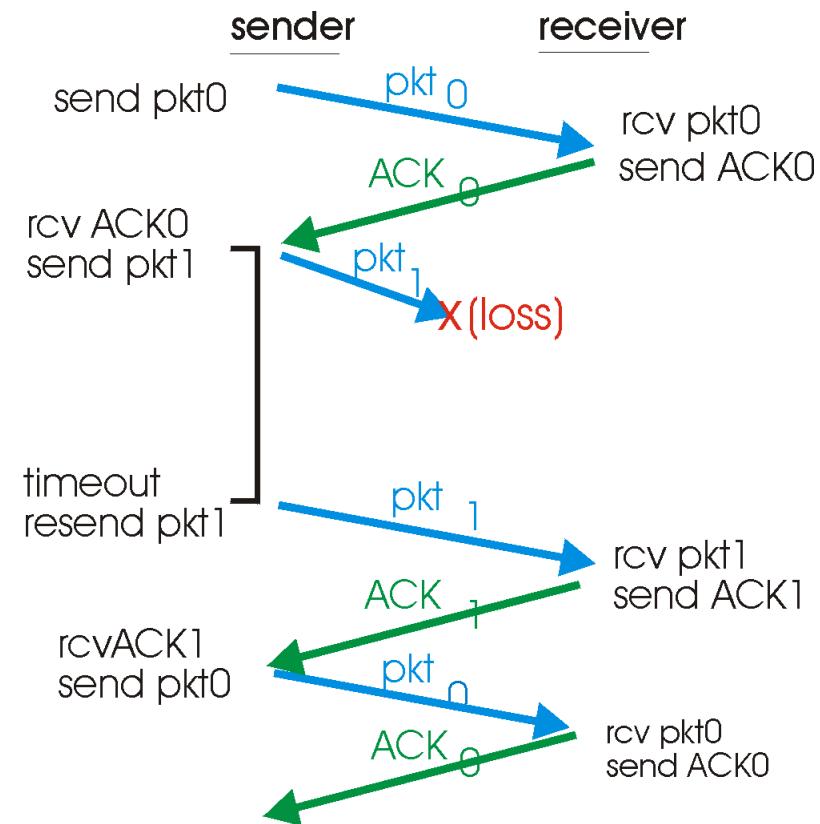
Stop-and-Wait example



Stop-and-Wait: Error Management

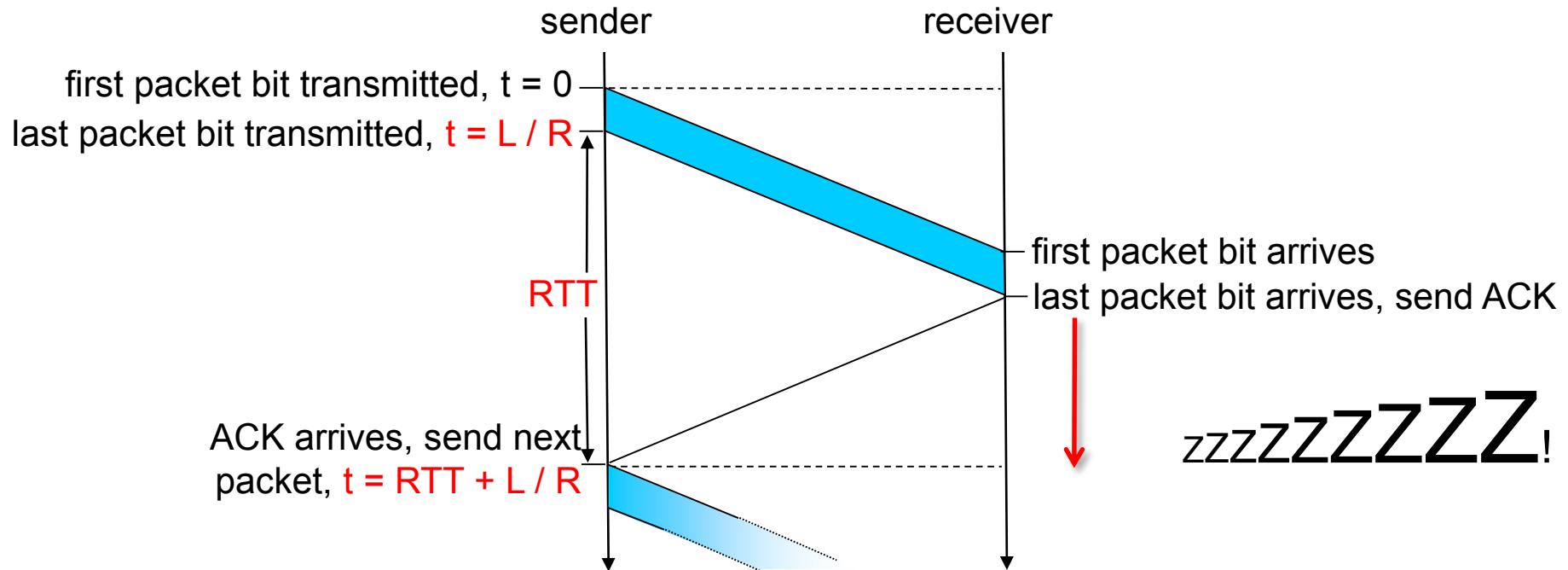


(a) operation with no loss



(b) lost packet

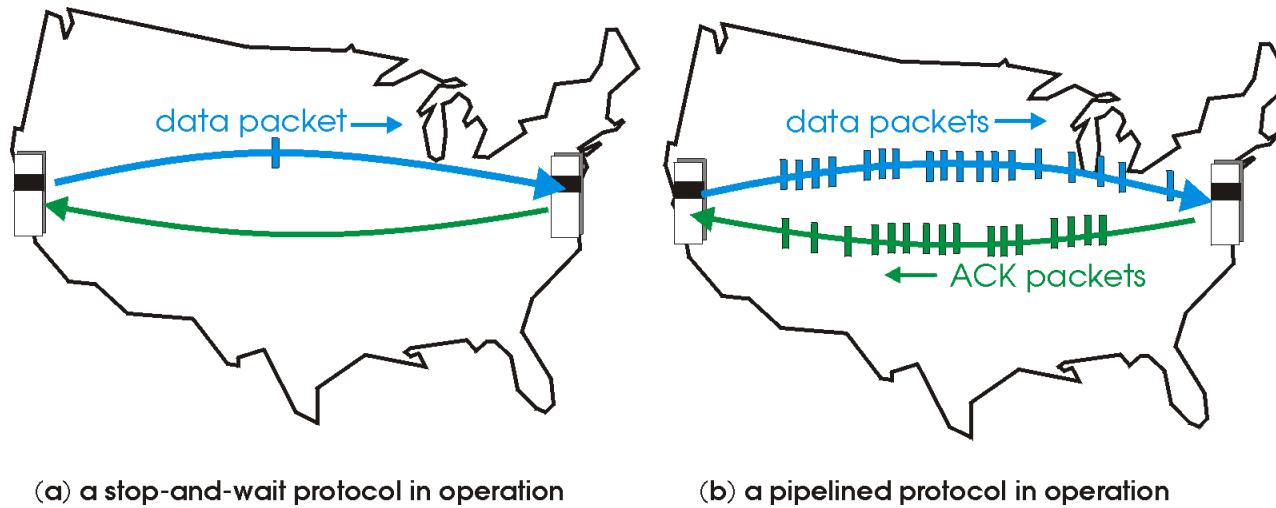
Stop-and-Wait Utilization Issue



3.4.2.3 Improving Utilization with Pipelined protocols

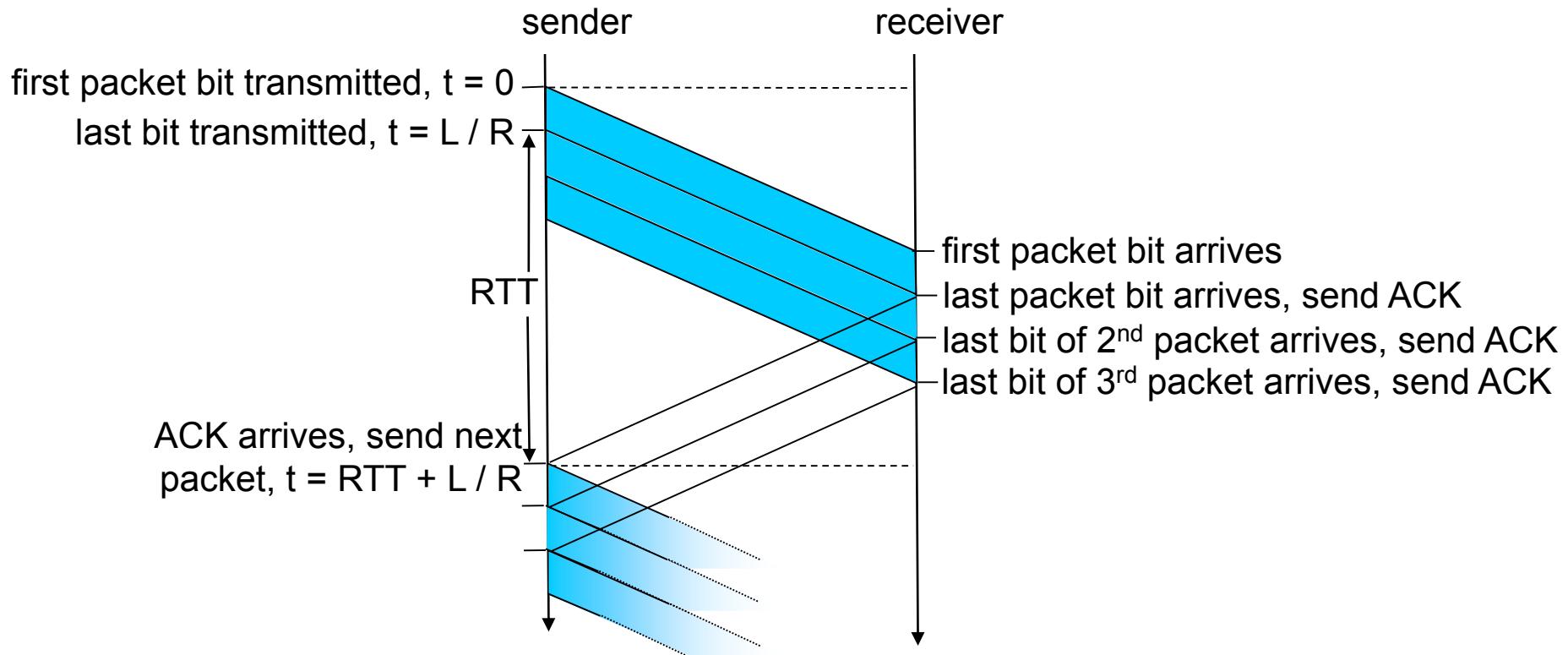
Pipelining: sender allows multiple, “in-flight”, yet-to-be-acknowledged pkts

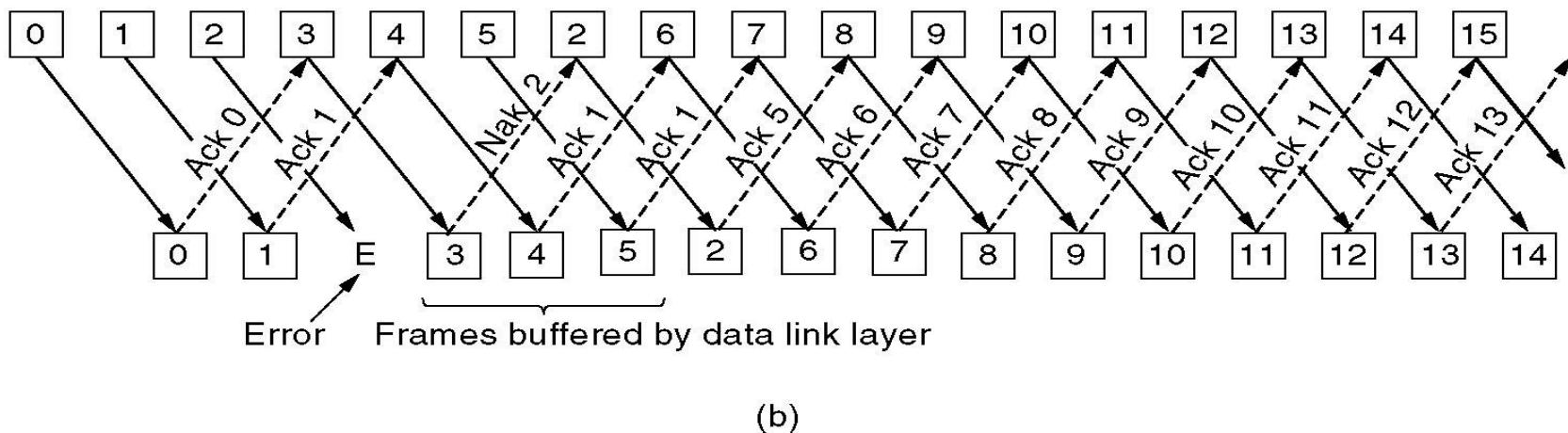
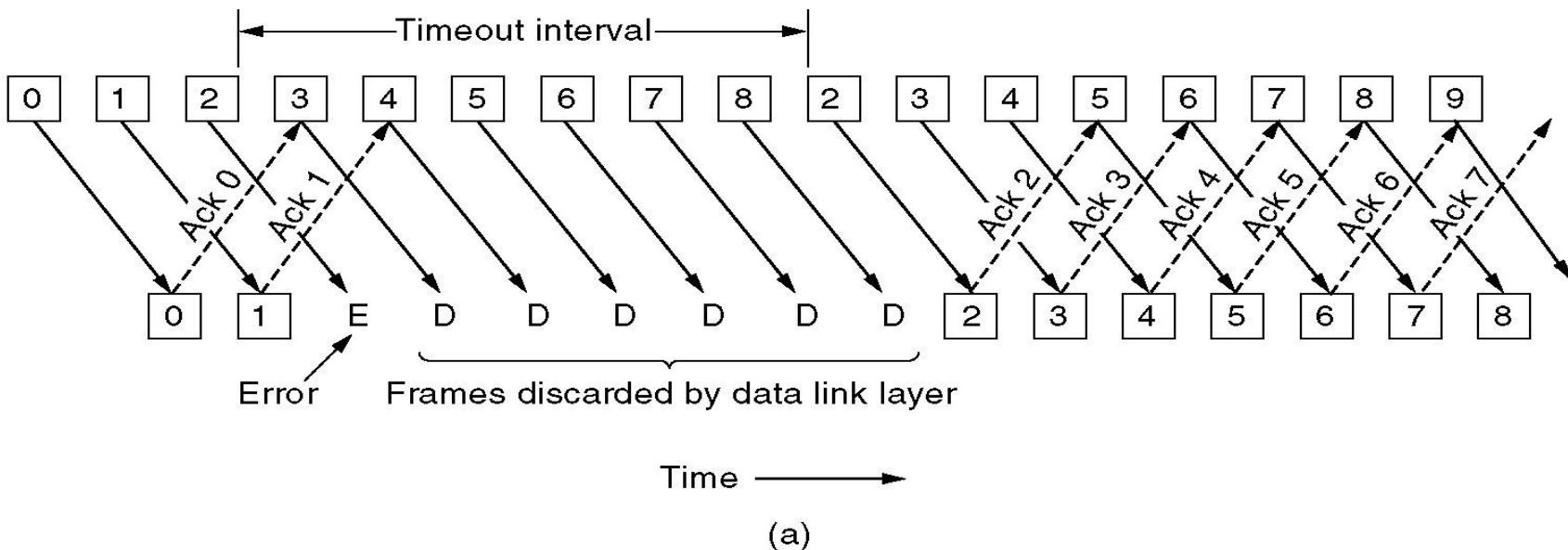
- range of sequence numbers must be increased
- buffering at sender and/or receiver



- Two generic forms of pipelined protocols: *go-Back-N*, *selective repeat*

Pipelining: increased utilization





Examples of Pipelined ARQs

a). Go Back n Sliding Window Protocol

b). Selective Repeat

3.4.2.4 TCP's reliable data transfer

- TCP creates reliable service on top of IP's unreliable service
 - Pipelined segments
 - Cumulative acks
 - Single retransmission timer
- Retransmissions are triggered by:
 - timeout events
 - duplicate acks
- Initially consider simplified TCP sender:
 - ignore duplicate acks
 - ignore flow control, congestion control
- Be careful – **reliable** cannot recover from broken links or server brownouts

TCP's Sliding Window Protocol

Hybrid approach:

- *Sender*
 - Transmits segment
 - Starts timer
- *Receiver*
 - Sends back ACK segment
 - ACK number = **next sequence number** it expects to receive
- If sender's timeout goes off before ACK is received, sender retransmits
- Receiver ACKs bytes received and size of window

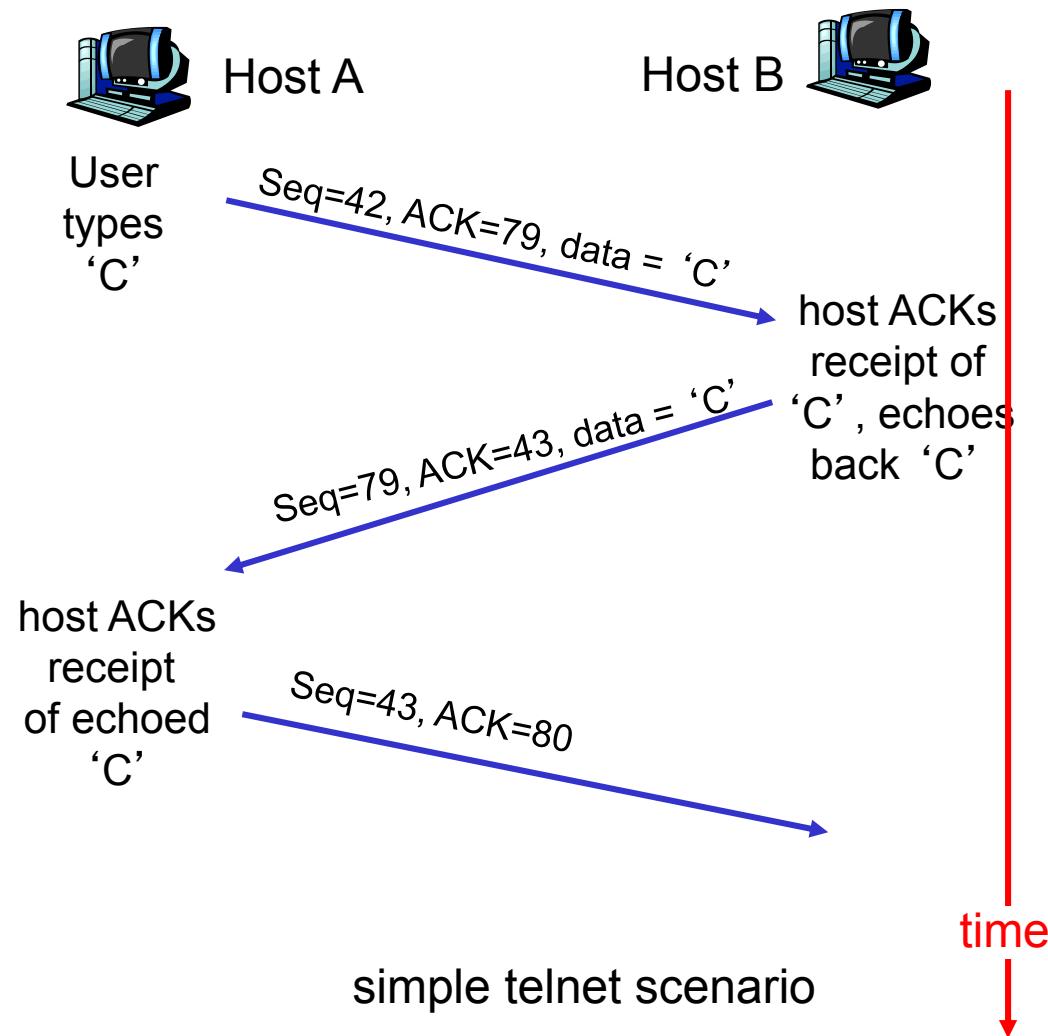
TCP Sequence Numbers and ACKs

Sequence Numbers:

- byte stream
“number” of first
byte in segment’s
data

ACKs:

- seq # of next byte
expected from other
side
- **Cumulative ACK**



TCP sender events:

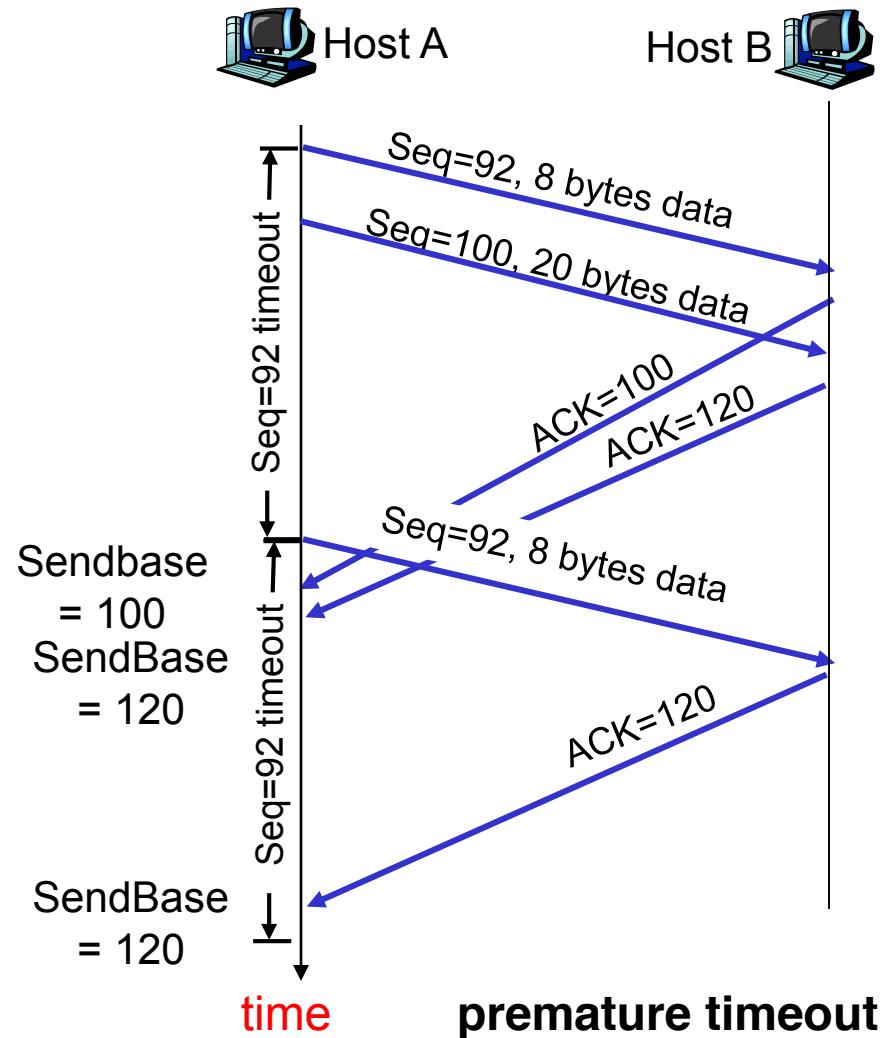
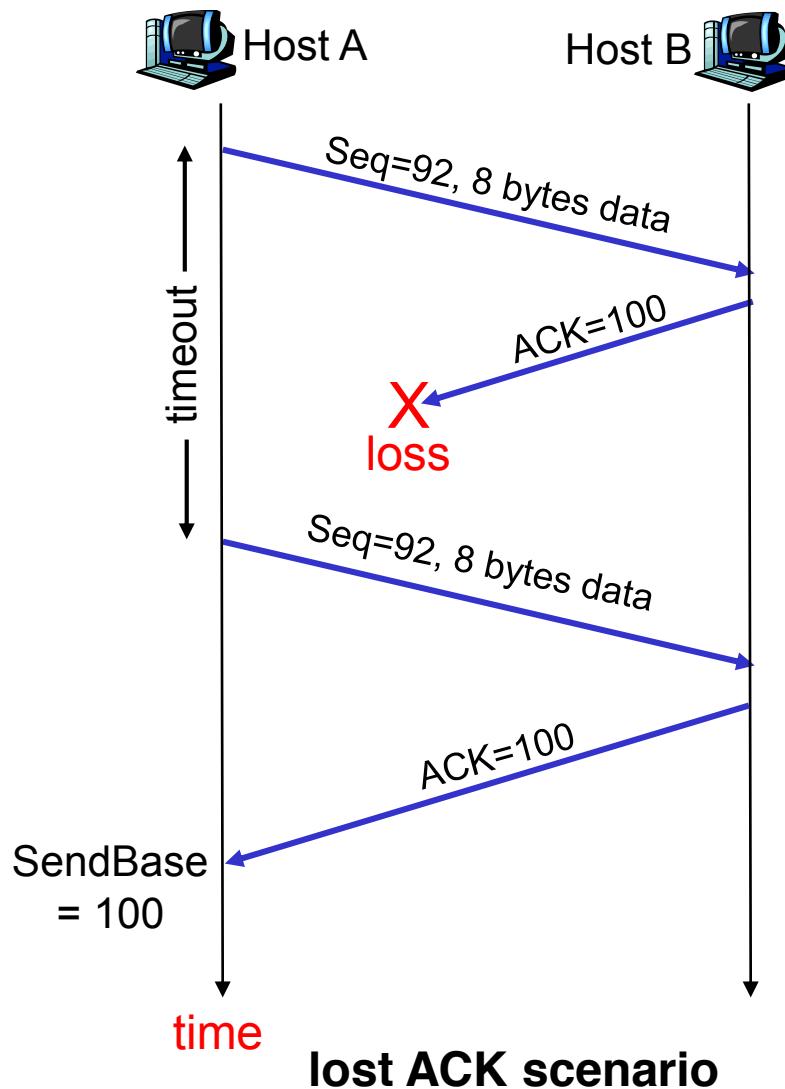
Data rcvd from app:

- Create segment with seq #
- seq # is byte-stream number of first data byte in segment**
- start timer if not already running (think of timer as for oldest unacked segment)
- expiration interval:
TimeOutInterval

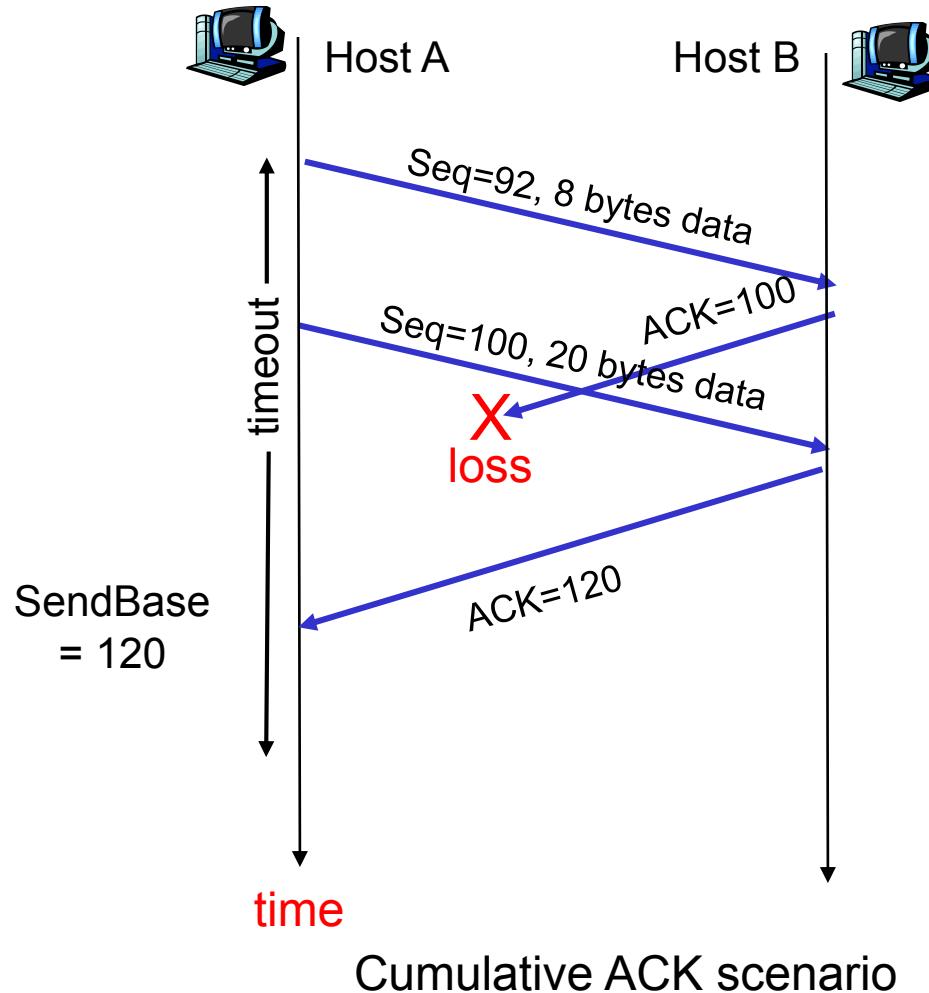
Timeout:

- retransmit segment that caused timeout
 - restart timer
- ## Ack rcvd:
- If Ack acknowledges previously unacked segments
 - update what is known to be acked
 - start timer if there are outstanding segments

TCP: Retransmission scenarios



TCP retransmission scenarios (more)



TCP ACK generation [RFC 1122, RFC 2581]

Event at Receiver	TCP Receiver action
Arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	Delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK
Arrival of in-order segment with expected seq #. One other segment has ACK pending	Immediately send single cumulative ACK, ACKing both in-order segments
Arrival of out-of-order segment higher-than-expect seq. # . Gap detected	Immediately send <i>duplicate ACK</i> , indicating seq. # of next expected byte
Arrival of segment that partially or completely fills gap	Immediate send ACK, provided that segment startsat lower end of gap

Fast Retransmit

- Time-out period often relatively long:
 - long delay before resending lost packet
- Detect lost segments via duplicate ACKs.
 - Sender often sends many segments back-to-back
 - If segment is lost, there will likely be many duplicate ACKs.
- If sender receives 3 ACKs for the same data, it supposes that segment after ACKed data was lost:
 - fast retransmit: resend segment before timer expires

Go-Back-N or Selective Repeat?

- **TCP ACKS are cumulative**
 - Out-of-order segments NOT individually ACKed
 - TCP receiver need only maintain:
 - Smallest seq# of a transmitted but unACKed byte (SendBase)
 - Seq# of next byte to send (NextSeqNum)
 - **Similar to Go-Back-N**
- But, TCP also uses selective-ACK
 - ACK out-of-order segments selectively
 - Selective retransmission - don't resend segments that have been selectively ACKed
 - **Similar to Selective Repeat**
- **Therefore TCP = Hybrid Go-Back-N and Selective Repeat**

Chapter 3 outline

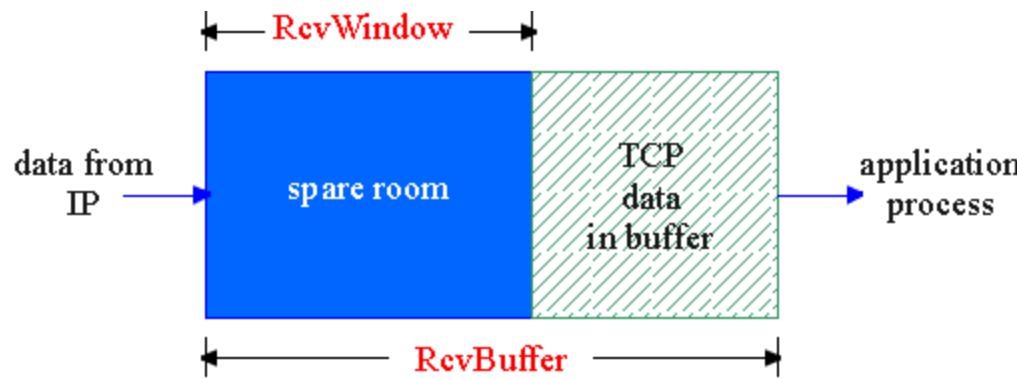
- 3.1 Transport-layer services and Protocols
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Connection-oriented transport: TCP
 - segment structure
 - reliable data transfer
 - **flow control**
 - connection management
 - TCP congestion control
- 3.5 Multimedia over TCP and UDP
 - Multimedia Networking Applications
 - Streaming stored multimedia over TCP and UDP
 - UDP or TCP
 - HTTP Streaming
 - Content Distribution Networks

3.4.3 TCP Flow Control

- ❑ receive side of TCP connection has a receive buffer:

flow control

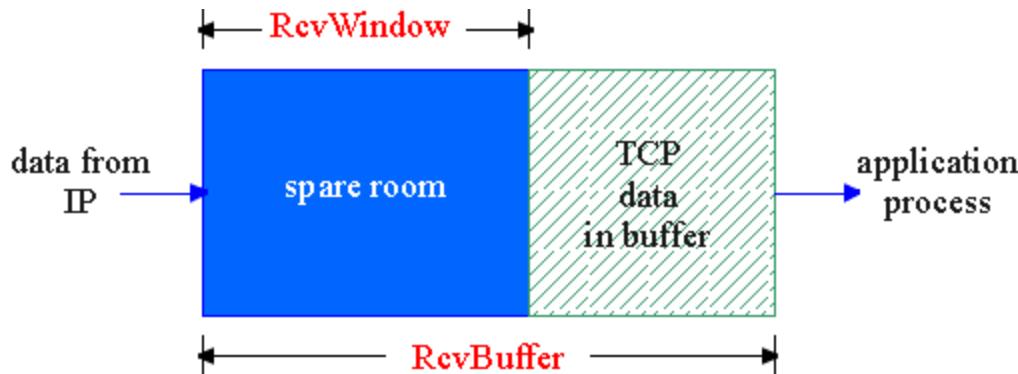
sender won't overflow receiver's buffer by transmitting too much, too fast



- ❑ app process may be slow at reading from buffer

- ❑ speed-matching service: matching the send rate to the receiving app's drain rate

TCP Flow control: how it works

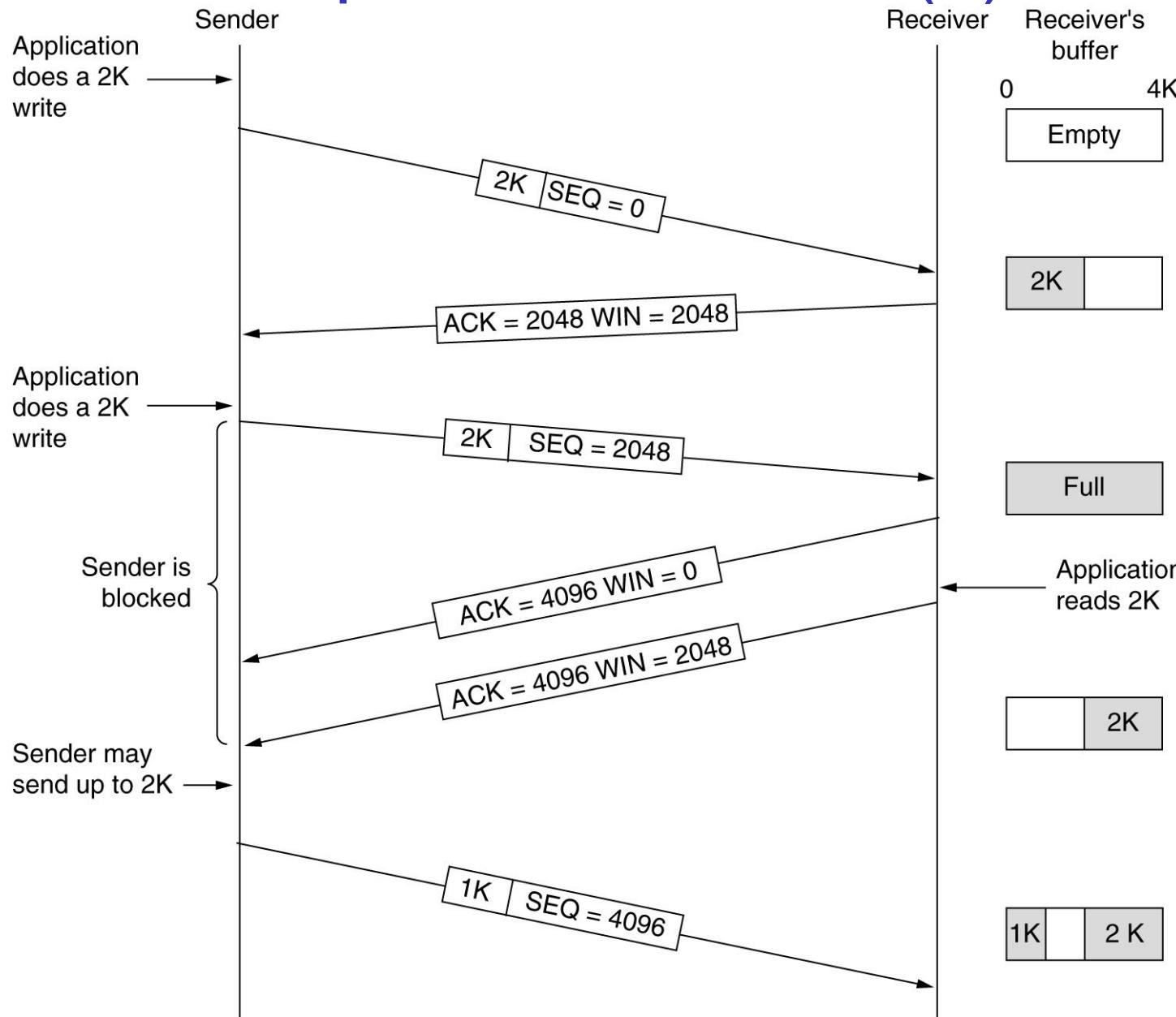


(Suppose TCP receiver discards out-of-order segments)

- spare room in buffer
 - = **RcvWindow**
 - = **RcvBuffer - [LastByteRcvd - LastByteRead]**

- Rcvr advertises spare room by including value of **RcvWindow** in segments
- Sender limits unACKed data to **RcvWindow**
 - guarantees receive buffer doesn't overflow

TCP seq. #'s and ACKs (3)



Chapter 3 outline

- 3.1 Transport-layer services and Protocols
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Connection-oriented transport: TCP
 - segment structure
 - reliable data transfer
 - flow control
 - **connection management**
 - TCP congestion control
- 3.5 Multimedia over TCP and UDP
 - Multimedia Networking Applications
 - Streaming stored multimedia over TCP and UDP
 - UDP or TCP
 - HTTP Streaming
 - Content Distribution Networks

3.4.4 TCP Connection Management

Recall: TCP sender, receiver establish “connection” before exchanging data segments

- initialize TCP variables:
 - seq. #s
 - buffers, flow control info (e.g. **RcvWindow**)
- *client*: connection initiator

```
Socket clientSocket = new
Socket("hostname", "port
number");
```
- *server*: contacted by client

```
Socket connectionSocket =
welcomeSocket.accept();
```

Three way handshake:

- Step 1: client host sends TCP SYN segment to server
 - specifies initial seq #
 - no data
- Step 2: server host receives SYN, replies with SYNACK segment
 - server allocates buffers
 - specifies server initial seq. #
- Step 3: client receives SYNACK, replies with ACK segment, which may contain data

TCP Connection Management (cont.)

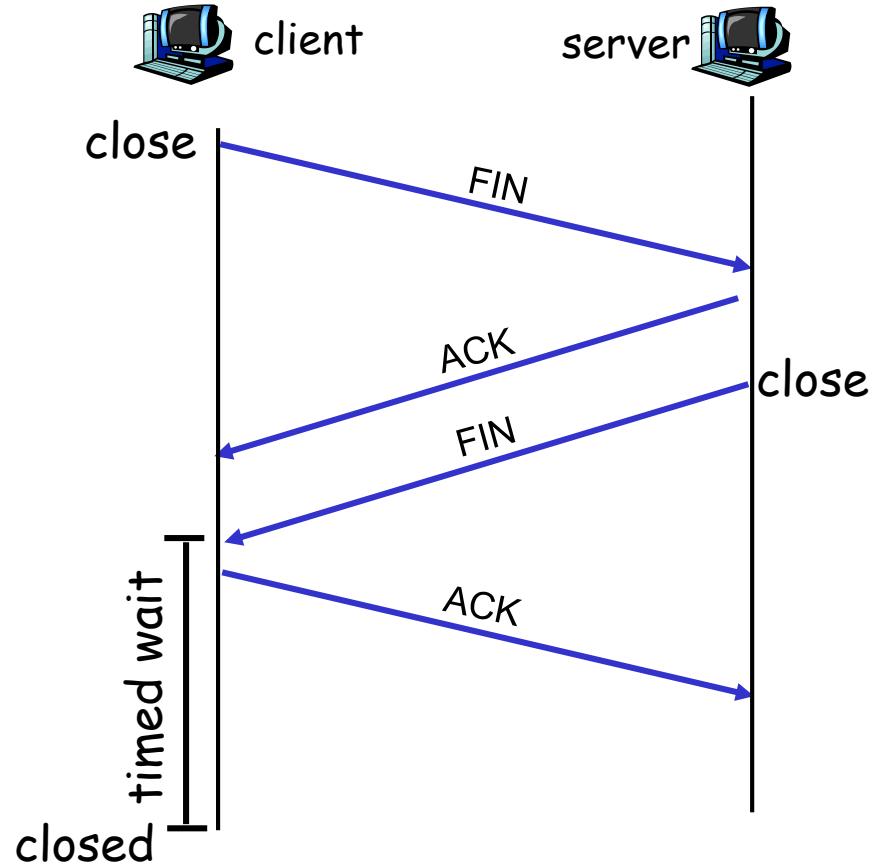
Closing a connection:

client closes socket:

```
clientSocket.close();
```

Step 1: client end system
sends TCP FIN control
segment to server

Step 2: server receives FIN,
replies with ACK. Closes
connection, sends FIN.



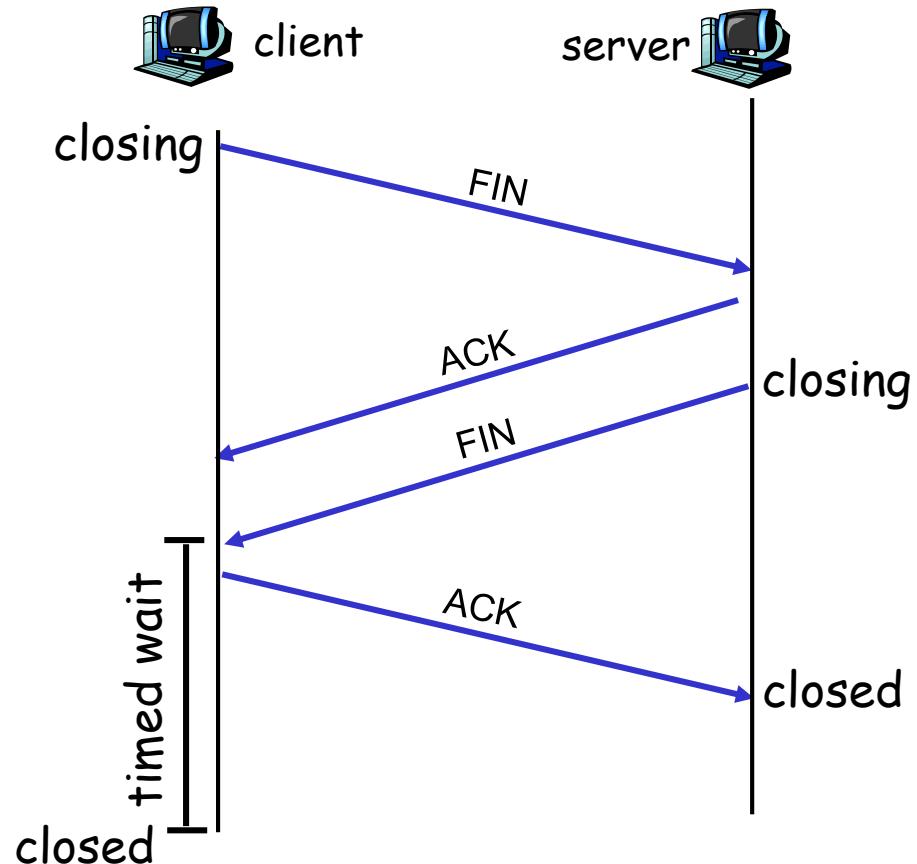
TCP Connection Management (cont.)

Step 3: client receives FIN, replies with ACK.

- Enters “timed wait” - will respond with ACK to received FINs

Step 4: server, receives ACK. Connection closed.

Note: with small modification, can handle simultaneous FINs.



Chapter 3 outline

- 3.1 Transport-layer services and Protocols
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Connection-oriented transport: TCP
 - segment structure
 - reliable data transfer
 - flow control
 - connection management
 - **TCP congestion control**
- 3.5 Multimedia over TCP and UDP
 - Multimedia Networking Applications
 - Streaming stored multimedia over TCP and UDP
 - UDP or TCP
 - HTTP Streaming
 - Content Distribution Networks

3.4.5 Congestion Control

Congestion:

- informally: “too many sources sending too much data too fast for *network* to handle”
- different from flow control!
- manifestations:
 - lost packets (buffer overflow at routers)
 - long delays (queueing in router buffers)
- a top-10 problem!

What Is Congestion?

❑ **Congestion**

- Occurs when the number of packets being transmitted through the network approaches the packet handling capacity of the network

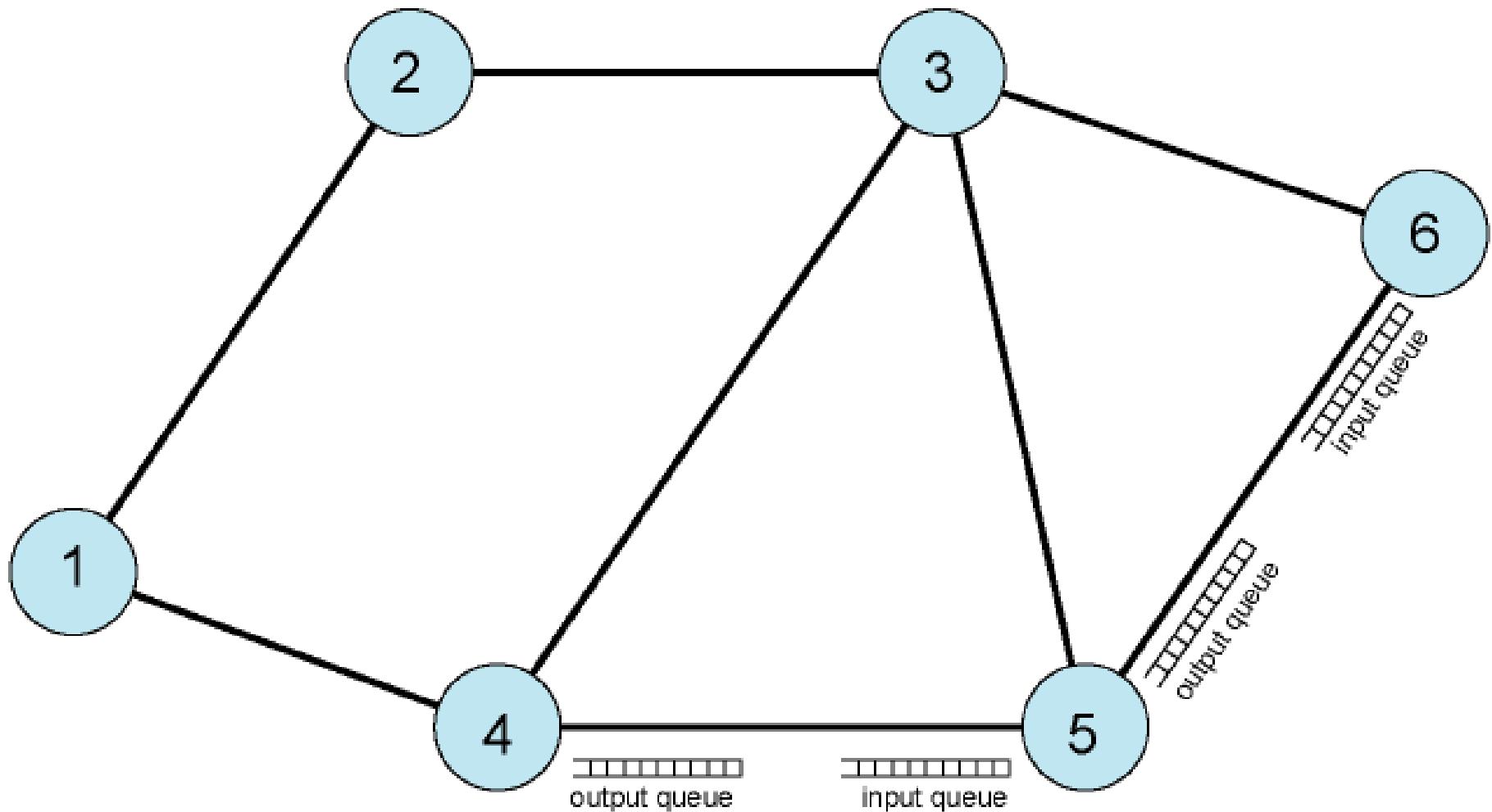
❑ **Congestion control**

- keeps number of packets below level at which performance falls off dramatically

❑ **Data network is a network of queues**

- Generally 80% utilization is critical
- Finite queues mean data may be lost

Interaction of Queues in a Data Network

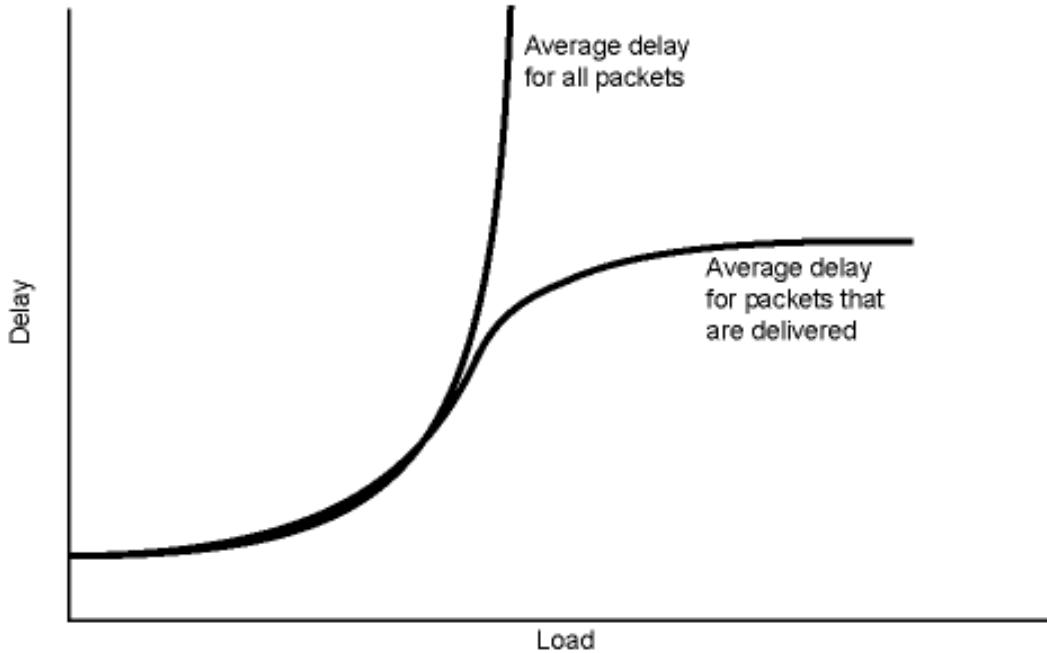
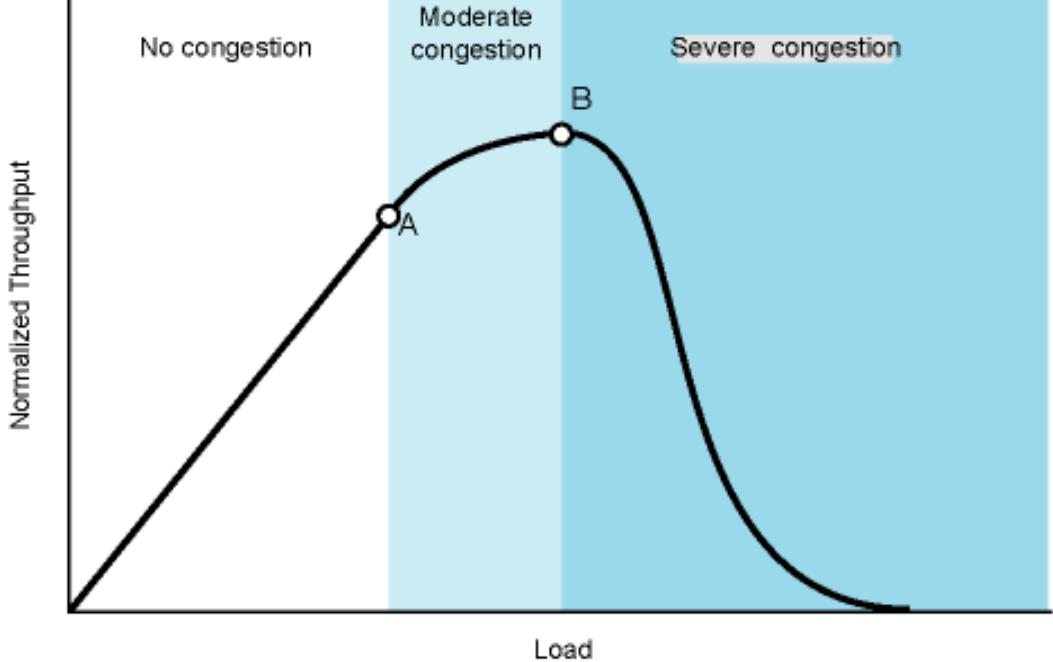


Practical Performance

- ❑ Ideal assumes infinite buffers and no overhead
- ❑ But Buffers are finite
- ❑ Overheads occur in exchanging congestion control messages

The Effects of Congestion

- As load increases, Congestion hotspots occur
 - Routing protocols attempt to route round hotspots
 - Increase in control signals increases network overhead, leading to even more congestion
- At B, queue saturation is reached
 - New packets are discarded
 - Causes more retransmits
 - Yet more network overhead
 - Throughput therefore drops



3.4.5.1 Approaches towards congestion control

Two broad approaches towards congestion control:

End-end congestion control:

- no explicit feedback from network
- congestion inferred from end-system observed loss, delay
- approach taken by TCP

Network-assisted congestion control:

- routers provide feedback to end systems
 - single bit indicating congestion (SNA, DECbit, TCP/IP ECN, ATM)
 - explicit rate sender should send at

3.4.5.2 TCP Congestion Control

- **Goal:** TCP sender should transmit as fast as possible, but without congesting network
 - Q: how to find rate *just* below congestion level
- Decentralized: each TCP sender sets its own rate, based on *implicit* feedback:
 - **ACK**: segment received (a good thing!), network not congested, so increase sending rate
 - **Lost segment**: assume loss due to congested network, so decrease sending rate

TCP Congestion Control: details

- sender limits transmission:

$$\frac{\text{LastByteSent} - \text{LastByteAcked}}{\text{CongWin}} \leq \text{CongWin}$$

- Roughly,

$$\text{rate} = \frac{\text{CongWin}}{\text{RTT}} \text{ Bytes/sec}$$

- **CongWin** is dynamic, function of perceived network congestion

How does sender perceive congestion?

- loss event = timeout or 3 duplicate acks
- TCP sender reduces rate (**CongWin**) after loss event

Three mechanisms:

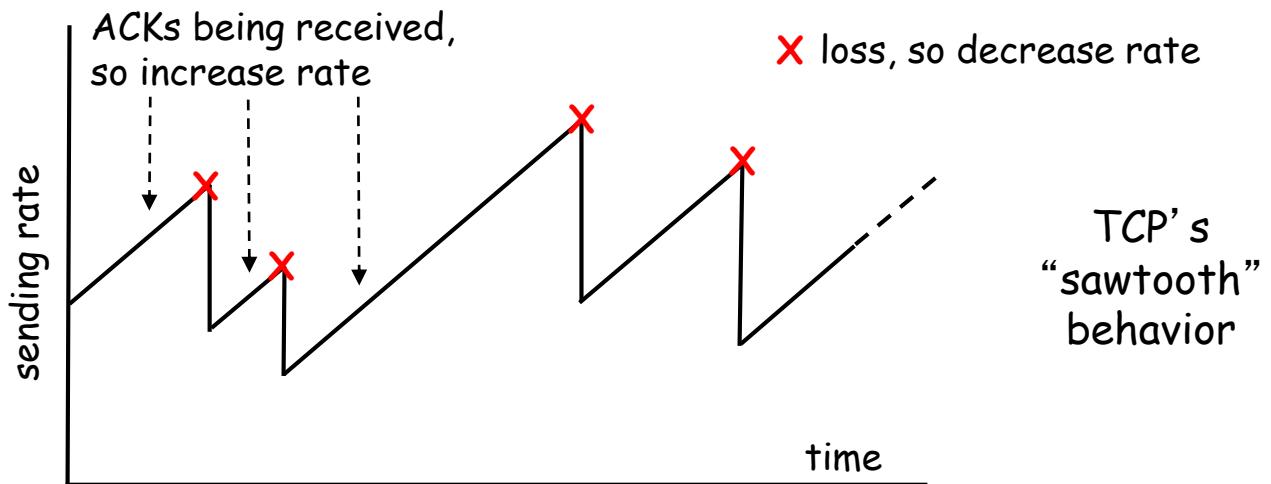
- AIMD
- slow start
- conservative after timeout events

Additive increase, Multiplicative decrease

- *Approach:* Increase transmission rate (window size), probing for usable bandwidth, until loss occurs
 - *Additive increase:* increase **CongWin** by 1 MSS every RTT until loss detected
 - *Multiplicative decrease:* cut **CongWin** in half after loss

TCP congestion control: bandwidth probing

- “**Probing for bandwidth**”: increase transmission rate on receipt of ACK, until eventually loss occurs, then decrease transmission rate
 - continue to increase on ACK, decrease on loss (since available bandwidth is changing, depending on other connections in network)

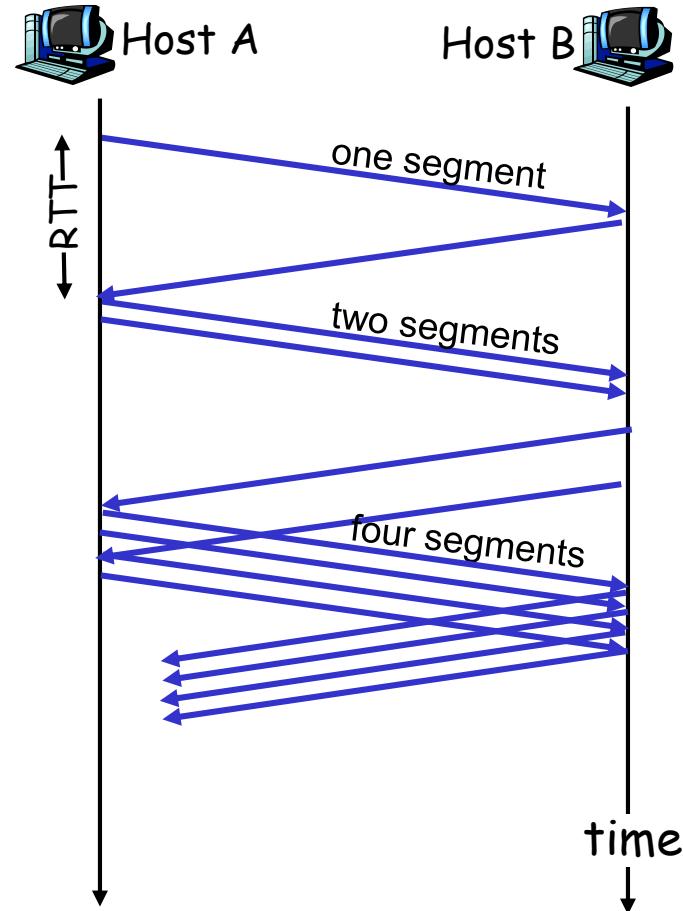


3.4.5.3 TCP Slow Start

- ❑ When connection begins, **CongWin** = 1 MSS
 - Example: MSS = 500 bytes & RTT = 200 msec
 - initial rate = 20 kbps
- ❑ available bandwidth may be \gg MSS/RTT
 - desirable to quickly ramp up to respectable rate
- ❑ When connection begins, increase rate exponentially fast until first loss event

TCP Slow Start (more)

- When connection begins, increase rate exponentially until first loss event:
 - double **CongWin** every RTT
 - done by incrementing **CongWin** for every ACK received
- Summary: initial rate is slow but ramps up exponentially fast



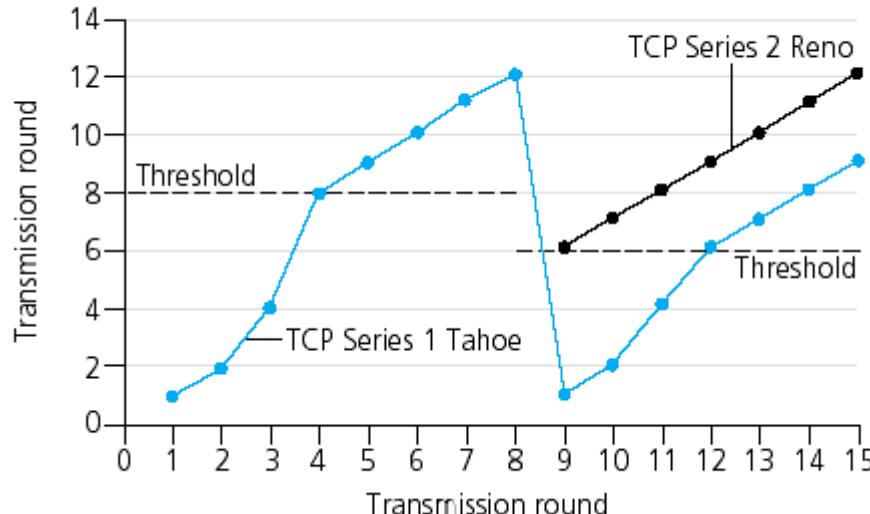
Refinement

Q: When should the exponential increase switch to linear?

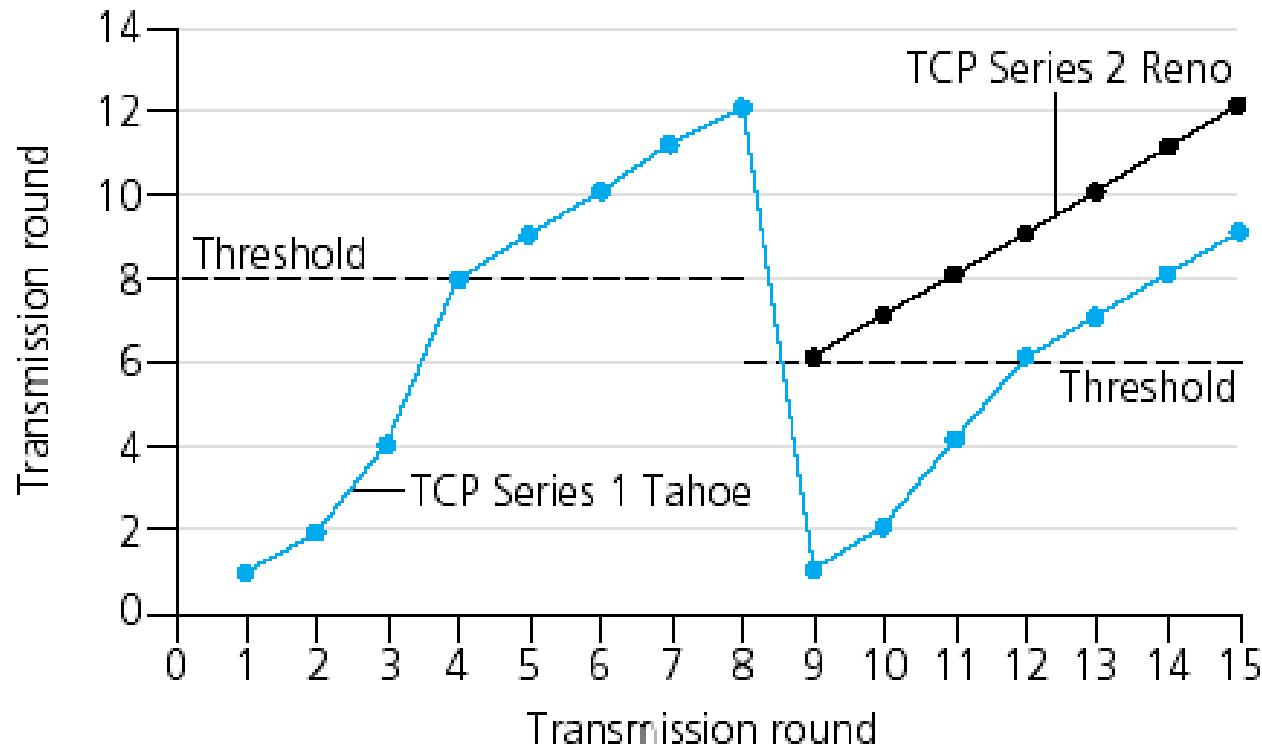
A: When **CongWin** gets to 1/2 of its value before timeout.

Implementation:

- Variable Threshold
- At loss event, Threshold is set to 1/2 of CongWin just before loss event



TCP Slow Start Congestion Control in action



Refinement: inferring loss

- After 3 dup ACKs:
 - **CongWin** is cut in half
 - window then grows linearly
- But after timeout event:
 - **CongWin** instead set to 1 MSS;
 - window then grows exponentially
 - to a threshold, then grows linearly

Philosophy: _____

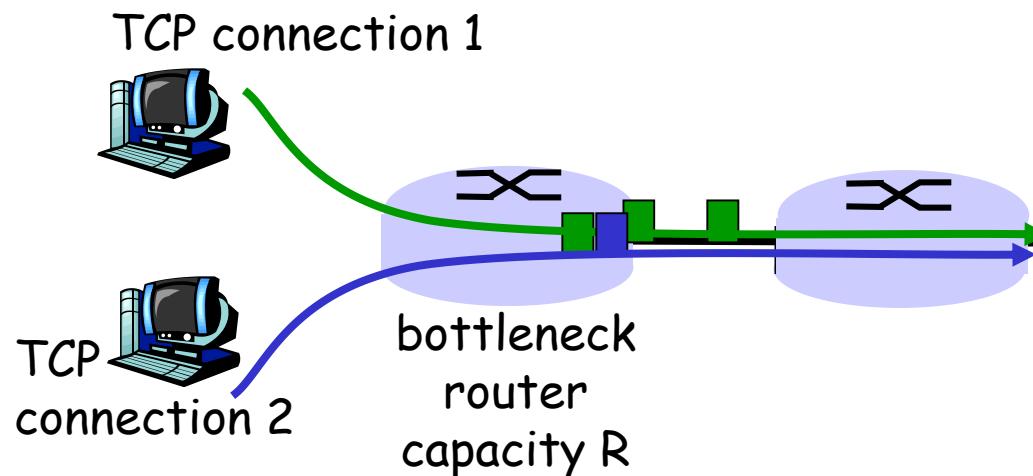
- 3 dup ACKs indicates network capable of delivering some segments
- timeout indicates a “more alarming” congestion scenario

Summary: TCP Congestion Control

- ❑ When **CongWin** is below **Threshold**, sender in **slow-start** phase, window grows exponentially.
- ❑ When **CongWin** is above **Threshold**, sender is in **congestion-avoidance** phase, window grows linearly.
- ❑ When a **triple duplicate ACK** occurs, **Threshold** set to **CongWin/2** and **CongWin** set to **Threshold**.
- ❑ When **timeout** occurs, **Threshold** set to **CongWin/2** and **CongWin** is set to 1 MSS.

3.4.5.4 TCP Fairness

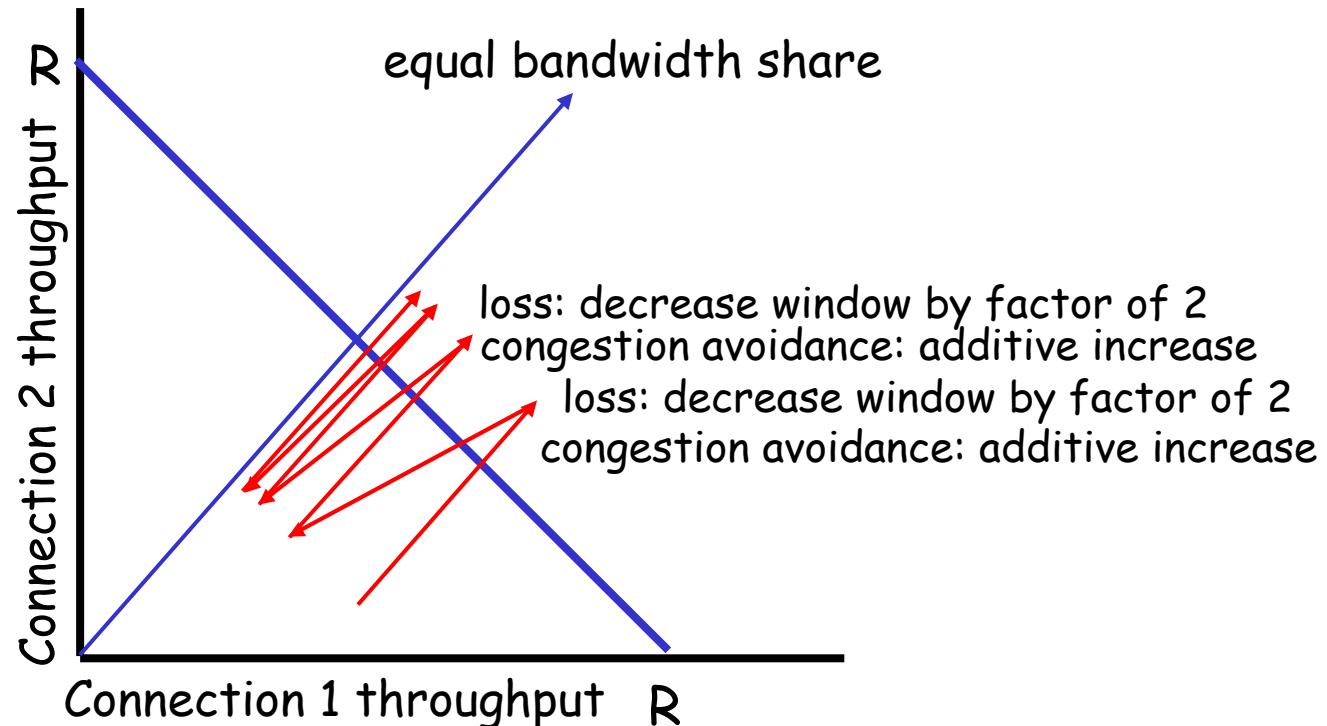
Fairness goal: if K TCP sessions share same bottleneck link of bandwidth R , each should have average rate of R/K



Why is TCP fair?

Two competing sessions:

- Additive increase gives slope of 1, as throughput increases
- multiplicative decrease decreases throughput proportionally



Fairness (more)

Fairness and UDP

- multimedia apps often do not use TCP
 - do not want rate throttled by congestion control
- instead use UDP:
 - pump audio/video at constant rate, tolerate packet loss

Fairness and parallel TCP connections

- nothing prevents app from opening parallel connections between 2 hosts.
- web browsers do this
- example: link of rate R supporting 9 connections;
 - new app asks for 1 TCP, gets rate $R/10$
 - new app asks for 11 TCPs, gets $R/2$!

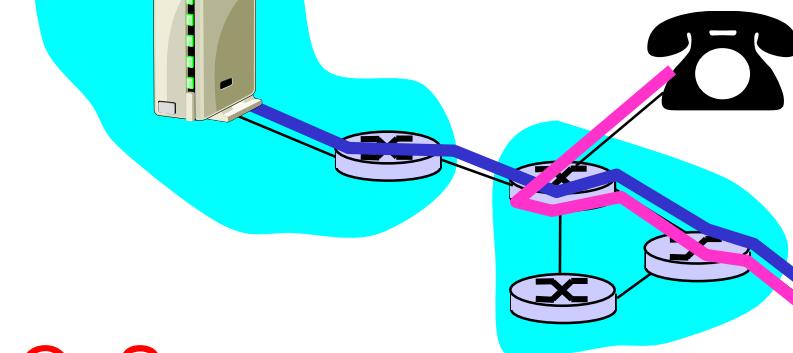
Chapter 3 outline

- 3.1 Transport-layer services and Protocols
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Connection-oriented transport: TCP
 - segment structure
 - reliable data transfer
 - flow control
 - connection management
 - TCP congestion control
- 3.5 Multimedia over TCP and UDP
 - Multimedia Networking Applications
 - Streaming stored multimedia over TCP and UDP
 - UDP or TCP
 - HTTP Streaming
 - Content Distribution Networks

3.5 Multimedia over TCP and UDP

- **TCP designed for data**
 - Slow to establish connections
 - Robust error control adds processing overhead, delay
 - Congestion control constantly adjusts rate at which packets are sent
- **UDP is faster, but still has issues**
 - No Quality of Service
 - Data either arrives or it doesn't – UDP doesn't care
 - No timing regularity between packets
- **How, then, do you stream multimedia over TCP/IP?**
 - i.e. How does YouTube work?!

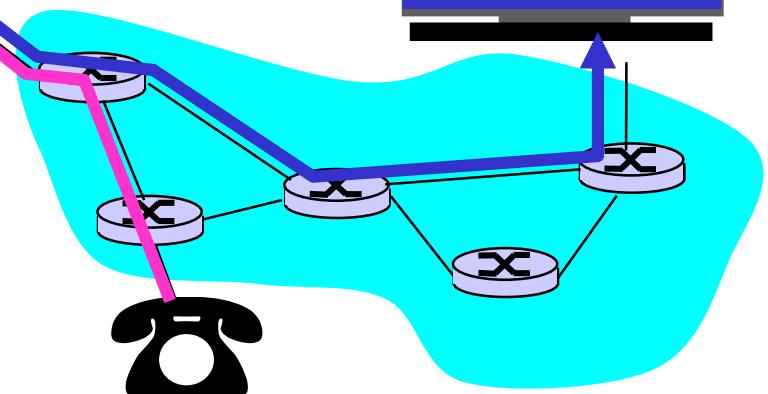
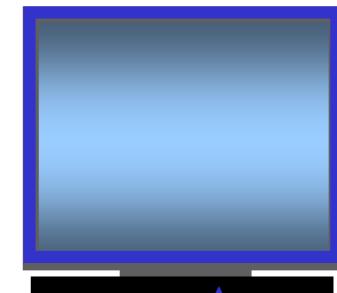
Multimedia and Quality of Service: What is it?



QoS

network provides application with *level of performance needed for application to function.*

multimedia applications:
network audio and video
("continuous media")



Chapter 3 outline

- 3.1 Transport-layer services and Protocols
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Connection-oriented transport: TCP
 - segment structure
 - reliable data transfer
 - flow control
 - connection management
 - TCP congestion control
- 3.5 Multimedia over TCP and UDP
 - **Multimedia Networking Applications**
 - Streaming stored multimedia over TCP and UDP
 - UDP or TCP
 - HTTP Streaming
 - Content Distribution Networks

3.5.1 MM Networking Applications

Classes of MM applications:

- 1) stored streaming
- 2) live streaming
- 3) interactive, real-time

Jitter is the variability of packet delays within the same packet stream

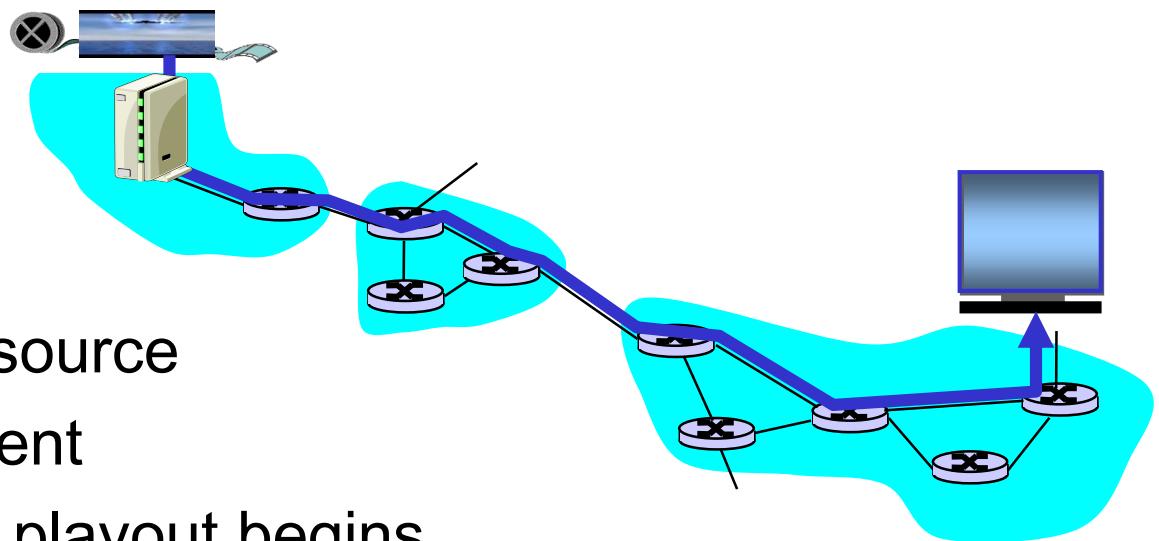
Fundamental characteristics:

- typically **delay sensitive**
 - end-to-end delay
 - delay jitter
- **loss tolerant**: infrequent losses cause minor glitches
- antithesis of data, which are loss *intolerant* but delay *tolerant*.

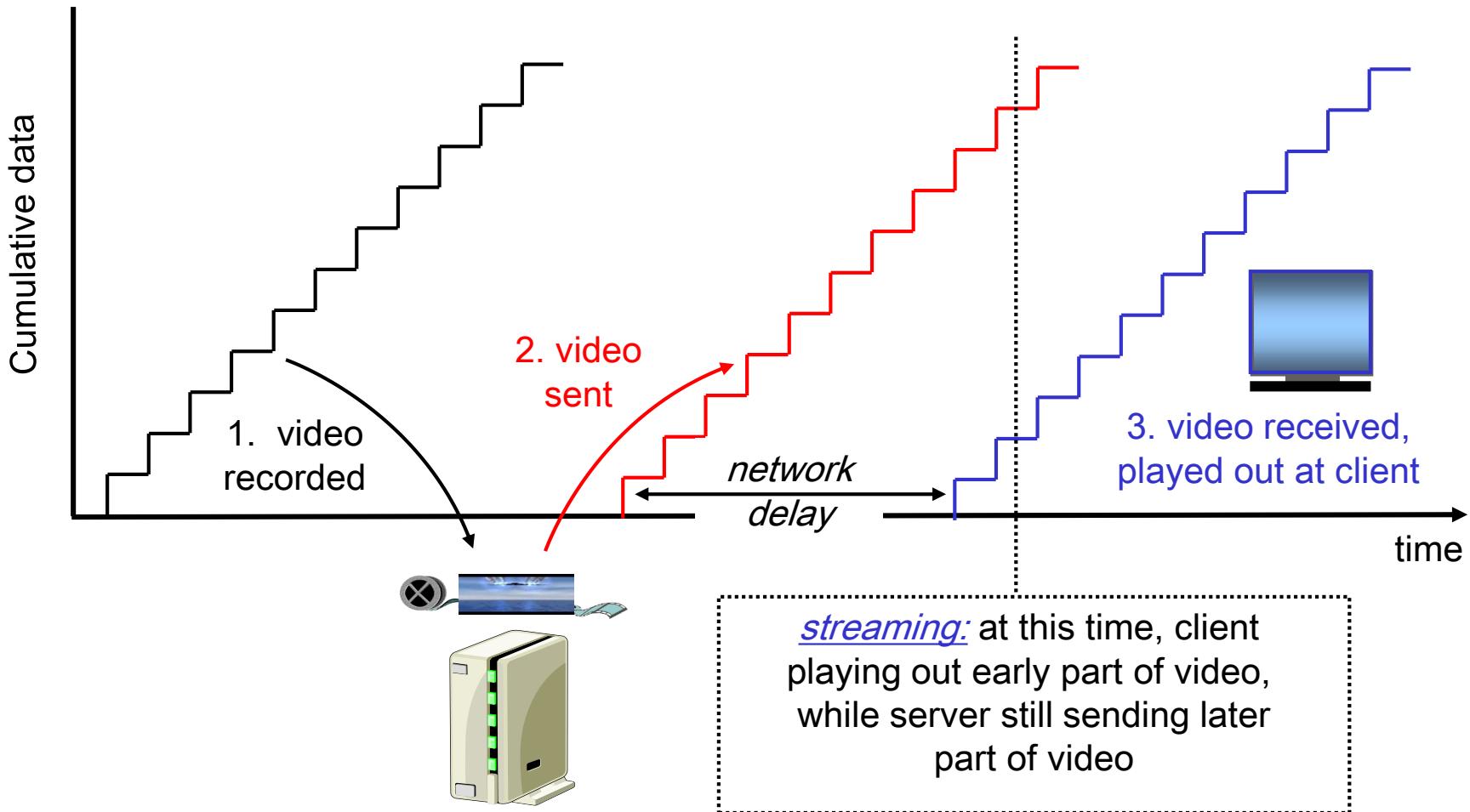
3.5.1.1 Streaming Stored Multimedia

Stored streaming:

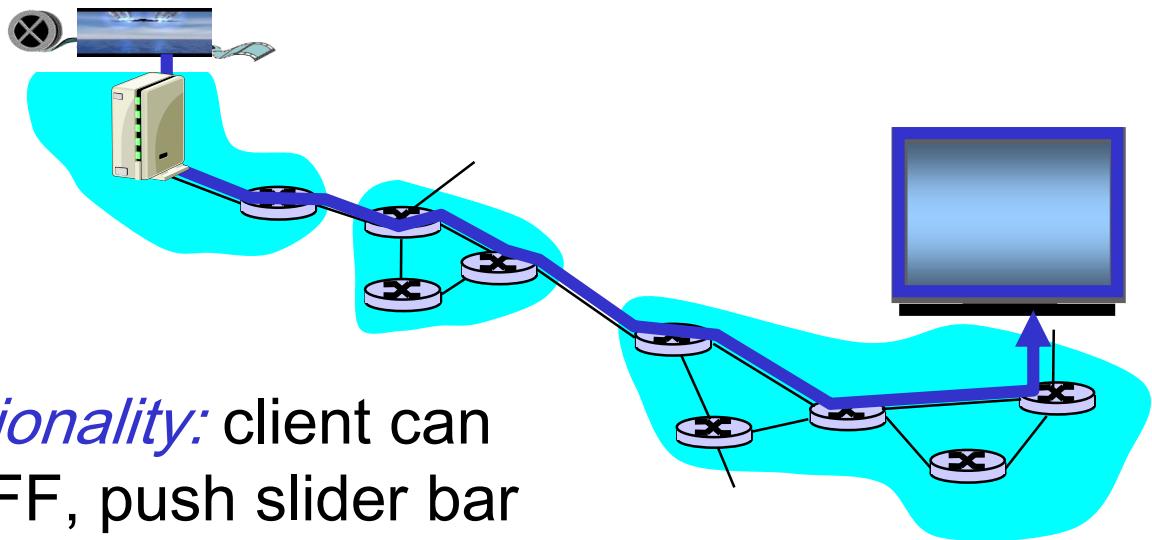
- media stored at source
- transmitted to client
- streaming: client playout begins *before* all data has arrived
- timing constraint for still-to-be transmitted data: in time for playout



Streaming Stored Multimedia: What is it?



Streaming *Stored* Multimedia: Interactivity



- *VCR-like functionality*: client can pause, rewind, FF, push slider bar
 - 10 sec initial delay OK
 - 1-2 sec until command effect OK
- timing constraint for still-to-be transmitted data: in time for playout

3.5.1.2 Streaming *Live* Multimedia

Examples:

- Internet radio talk show
- live sporting event

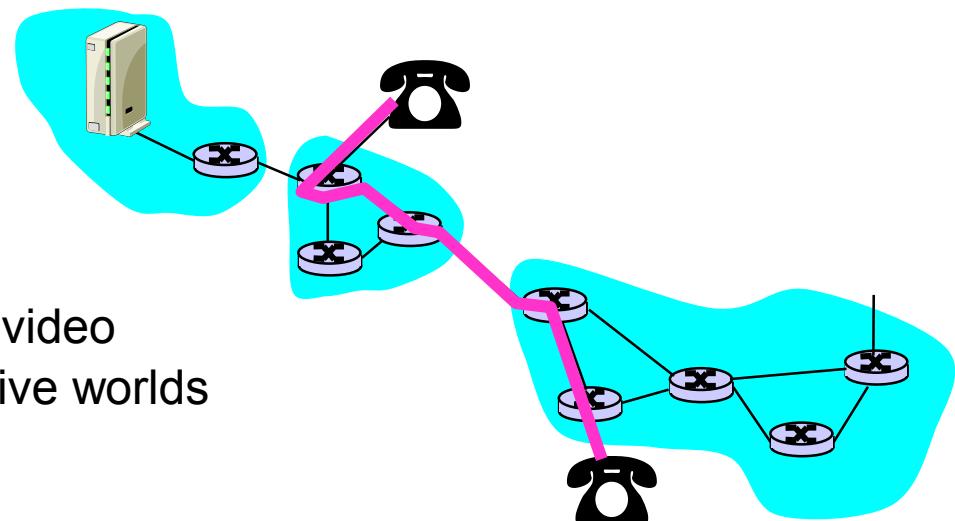
Streaming (as with streaming *stored* multimedia)

- playback buffer
- playback can lag tens of seconds after transmission
- still have timing constraint

Interactivity

- fast forward impossible
- rewind, pause possible!

3.5.1.3 Real-Time Interactive Multimedia

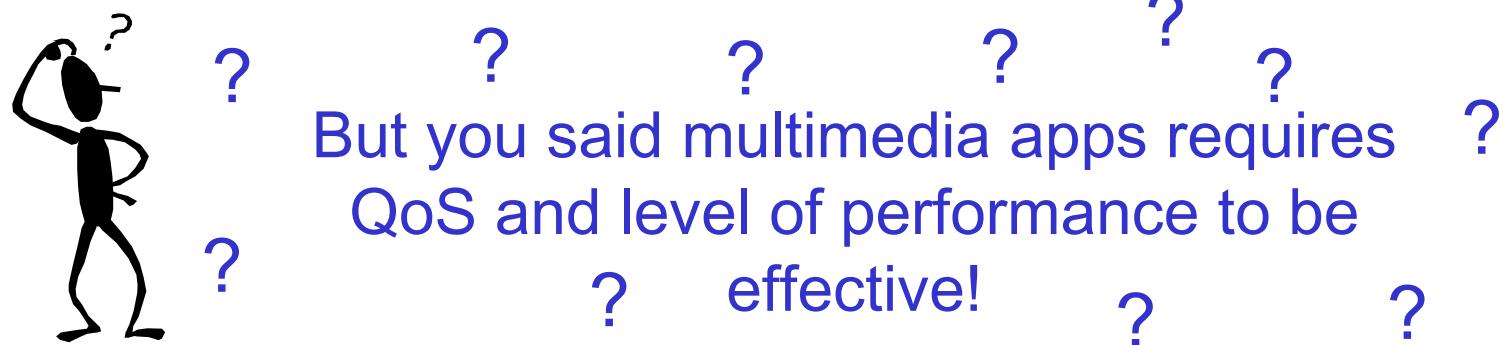


- ❑ **applications:** IP telephony, video conference, distributed interactive worlds
- ❑ **end-end delay requirements:**
 - audio: < 150 msec good, < 400 msec OK
 - includes application-level (packetization) and network delays
 - higher delays noticeable, impair interactivity
- ❑ **session initialization**
 - callee must advertise its IP address, port number, encoding algorithms

3.5.2 Multimedia Over Today's Internet

TCP/UDP/IP: “best-effort service”

- ❑ *no* guarantees on delay, loss



Today's Internet multimedia applications use application-level techniques to mitigate (as best possible) effects of delay, loss

How should the Internet evolve to better support multimedia?

Integrated services philosophy:

- fundamental changes in Internet so that apps can reserve end-to-end bandwidth
- requires new, complex software in hosts & routers

Laissez-faire

- no major changes
- more bandwidth when needed
- content distribution, application-layer multicast
 - application layer

Differentiated services philosophy:

- fewer changes to Internet infrastructure, yet provide 1st and 2nd class service



What's your opinion?

Chapter 3 outline

- 3.1 Transport-layer services and Protocols
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Connection-oriented transport: TCP
 - segment structure
 - reliable data transfer
 - flow control
 - connection management
 - TCP congestion control
- 3.5 Multimedia over TCP and UDP
 - Multimedia Networking Applications
 - **Streaming stored multimedia over TCP and UDP**
 - UDP or TCP
 - HTTP Streaming
 - Content Distribution Networks

3.5.3 Approaches to Streaming Stored Multimedia over TCP and UDP

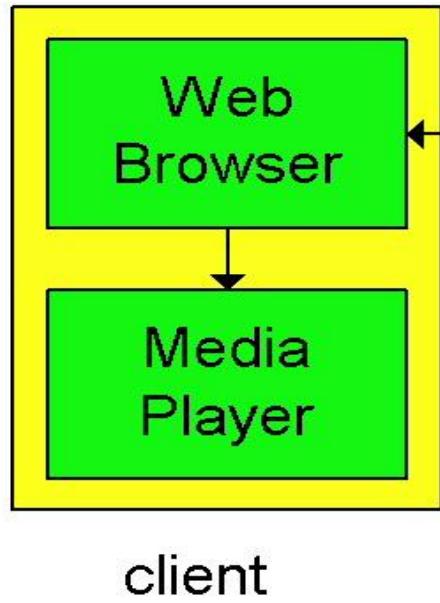
Relies on application-level streaming techniques for making the best out of best effort service:

- Use of UDP versus TCP
- Client-side buffering
- Multiple encodings of multimedia

Media Player

- jitter removal
- decompression
- error concealment
- graphical user interface w/ controls for interactivity

Internet multimedia: simplest approach

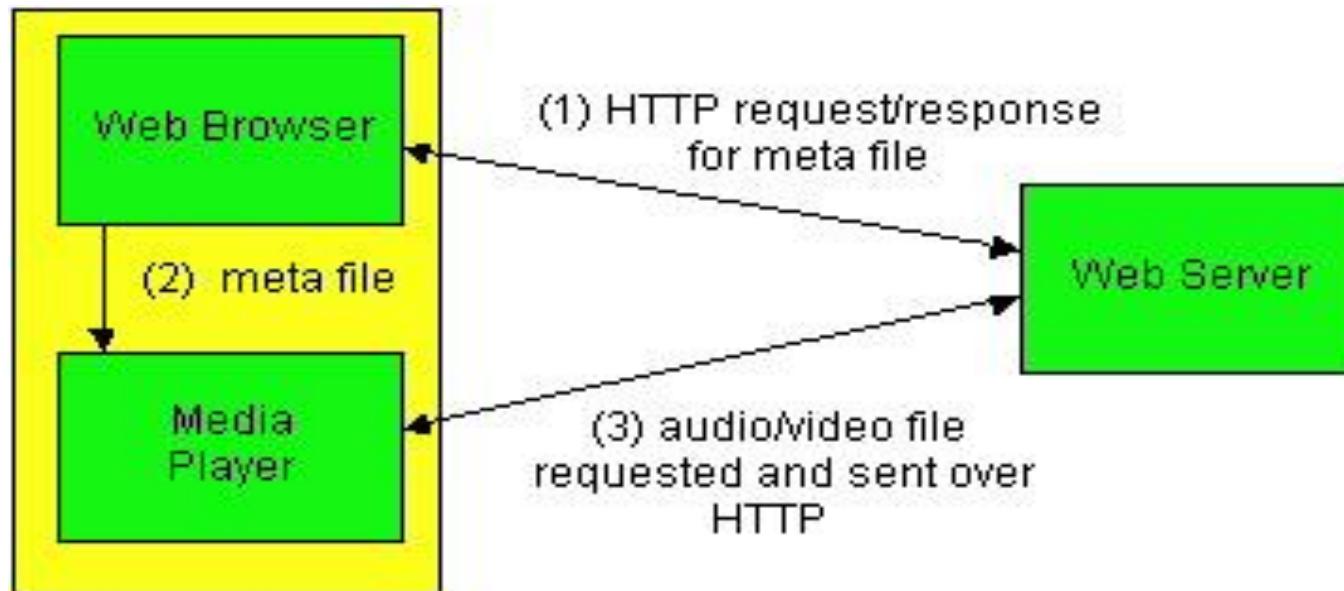


- ❑ audio or video stored in file
- ❑ files transferred as HTTP object
 - received in entirety at client
 - then passed to player

audio, video not streamed:

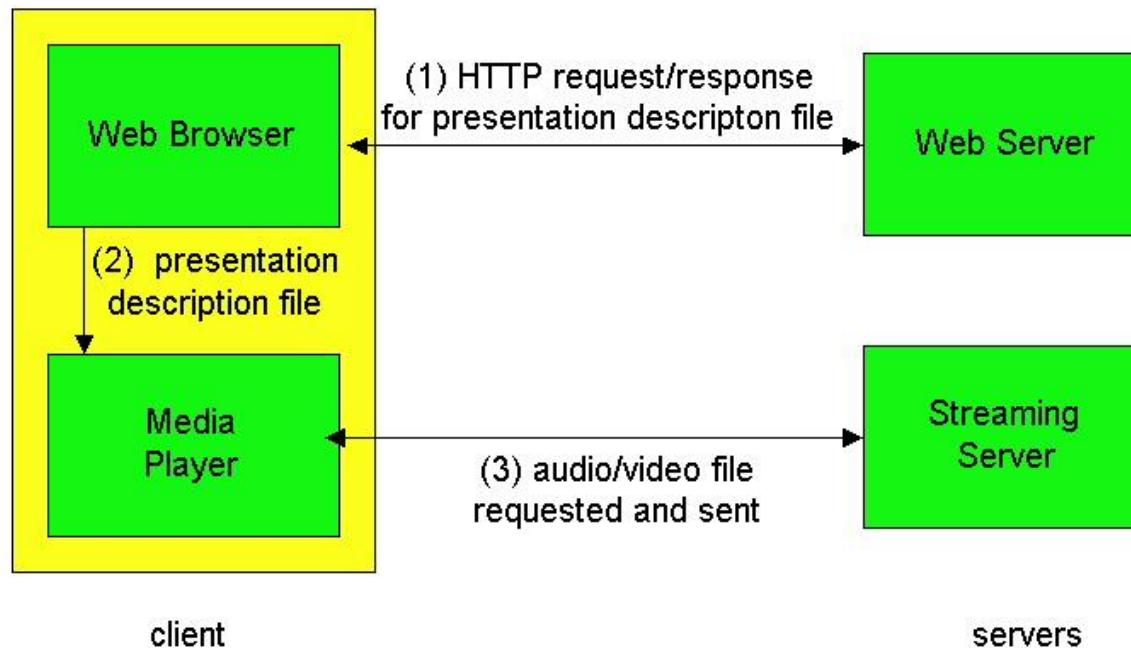
- ❑ no, “pipelining,” long delays until playout!

Internet multimedia: streaming approach



- browser GETs **metafile**
- browser launches player, passing metafile
- player contacts server
- server **streams** audio/video to player

Streaming from a streaming server



- Allows for non-HTTP protocol between server, media player
- UDP or TCP for step (3)
- Media servers are extremely expensive though

Chapter 3 outline

- 3.1 Transport-layer services and Protocols
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Connection-oriented transport: TCP
 - segment structure
 - reliable data transfer
 - flow control
 - connection management
 - TCP congestion control
- 3.5 Multimedia over TCP and UDP
 - Multimedia Networking Applications
 - Streaming stored multimedia over TCP and UDP
 - UDP or TCP
 - HTTP Streaming
 - Content Distribution Networks

3.5.4 UDP or TCP?

UDP

- server sends at rate appropriate for client (oblivious to network congestion !)
 - often send rate = encoding rate = constant rate
 - then, fill rate = constant rate - packet loss
- short playout delay (2-5 seconds) to remove network jitter
- error recover: time permitting

TCP

- send at maximum possible rate under TCP
- fill rate fluctuates due to TCP congestion control
- larger playout delay: smooth TCP delivery rate
- HTTP/TCP passes more easily through firewalls
- What would YouTube do?

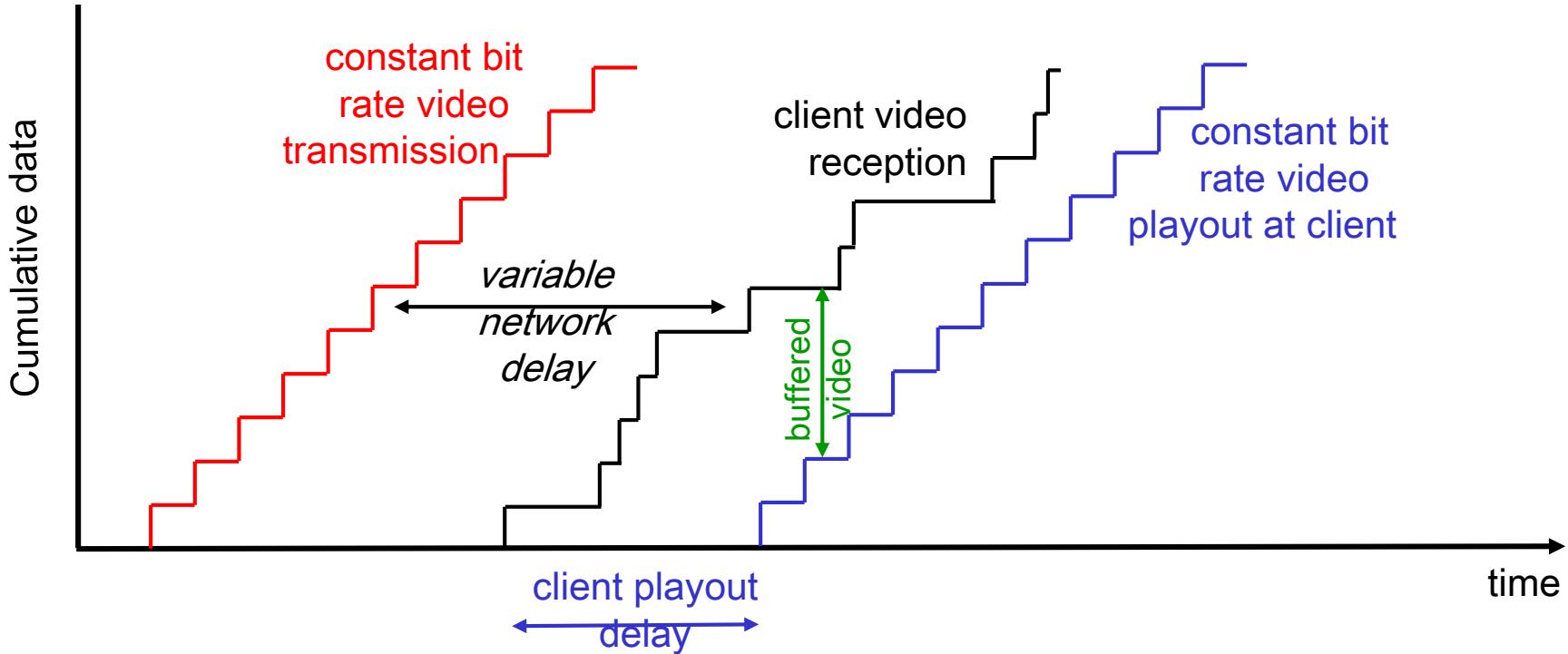
Chapter 3 outline

- 3.1 Transport-layer services and Protocols
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Connection-oriented transport: TCP
 - segment structure
 - reliable data transfer
 - flow control
 - connection management
 - TCP congestion control
- 3.5 Multimedia over TCP and UDP
 - Multimedia Networking Applications
 - Streaming stored multimedia over TCP and UDP
 - UDP or TCP
 - **HTTP Streaming**
 - Content Distribution Networks

3.5.5 HTTP Streaming over TCP

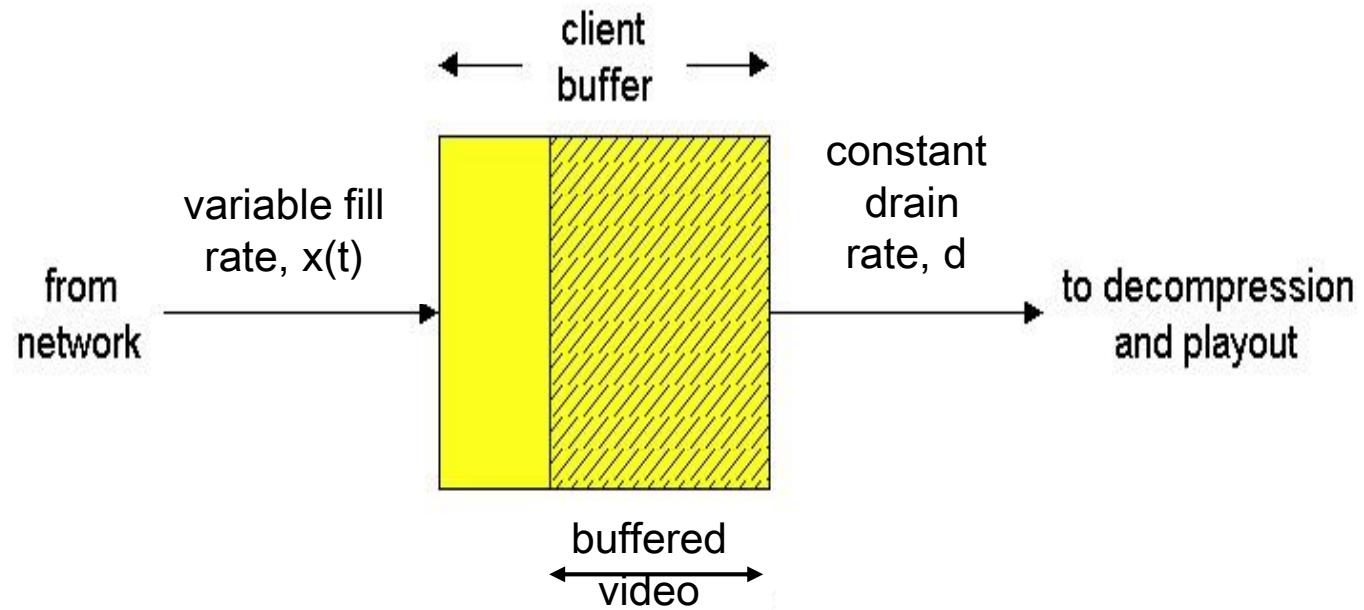
- HTTP Streaming – the YouTube approach
 - Video stored on an HTTP server as a file
 - Server sends video file within HTTP response message
 - Sent as quickly as congestion control and flow control algos will allow
- But what about “sawtooth” effect of congestion control algo?
 - Prefetching overcomes it
 - Download video file at a rate higher than the consumption rate
 - For best results, $TCP\ throughput = 2x\ Media\ bit\ rate$

3.5.5.1 Client Buffering



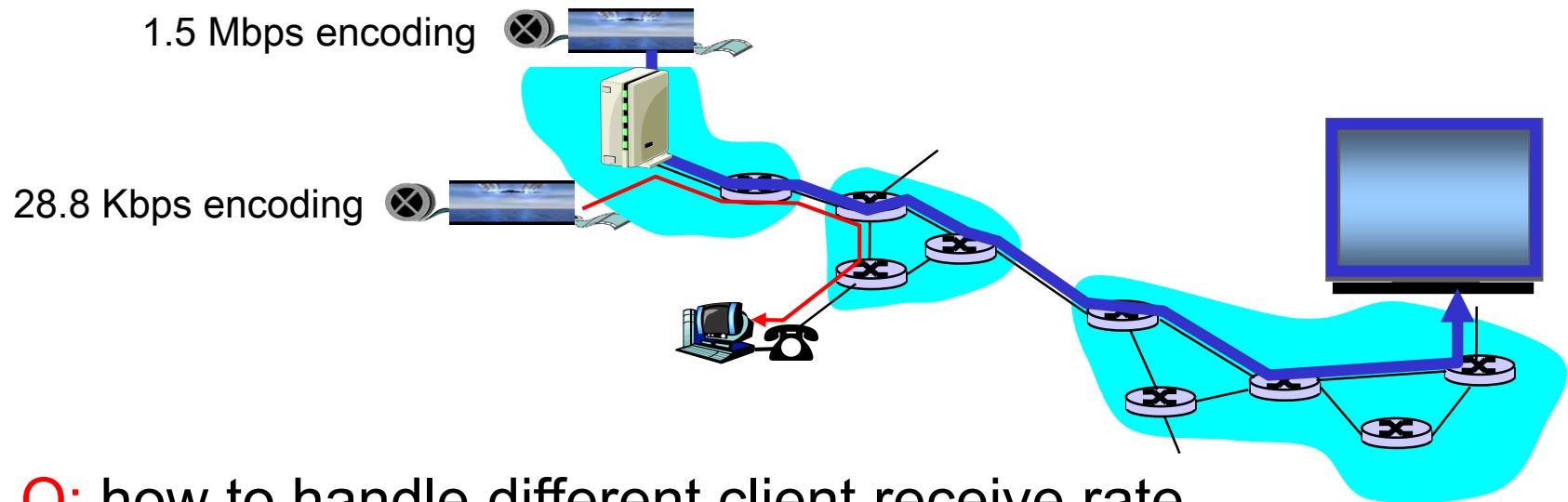
- client-side buffering, playout delay compensate for network-added delay, delay jitter

Client Buffering (cont)



- client-side buffering, playout delay compensate for network-added delay, delay jitter

3.5.5.2 Handling different client rate(s)



Q: how to handle different client receive rate capabilities?

- 28.8 Kbps dialup
- 100 Mbps Ethernet

A: server stores, transmits multiple copies of video, encoded at different rates

3.5.5.3 User Control of Streaming Media: RTSP

HTTP

- does not target multimedia content
- no commands for fast forward, etc.

RTSP: (RFC 2326)

- client-server application layer protocol
- user control: rewind, fast forward, pause, resume, repositioning, etc...

What it doesn't do:

- doesn't define how audio/video is encapsulated for streaming over network
- doesn't restrict how streamed media is transported (UDP or TCP possible)
- doesn't specify how media player buffers audio/video

RTSP: out of band control

FTP uses an “out-of-band” control channel:

- file transferred over one TCP connection.
- control info (directory changes, file deletion, rename) sent over separate TCP connection
- “out-of-band”, “in-band” channels use different port numbers

RTSP messages also sent out-of-band:

- RTSP control messages use different port numbers than media stream: out-of-band.
 - port 554
- media stream is considered “in-band”.

RTSP Example

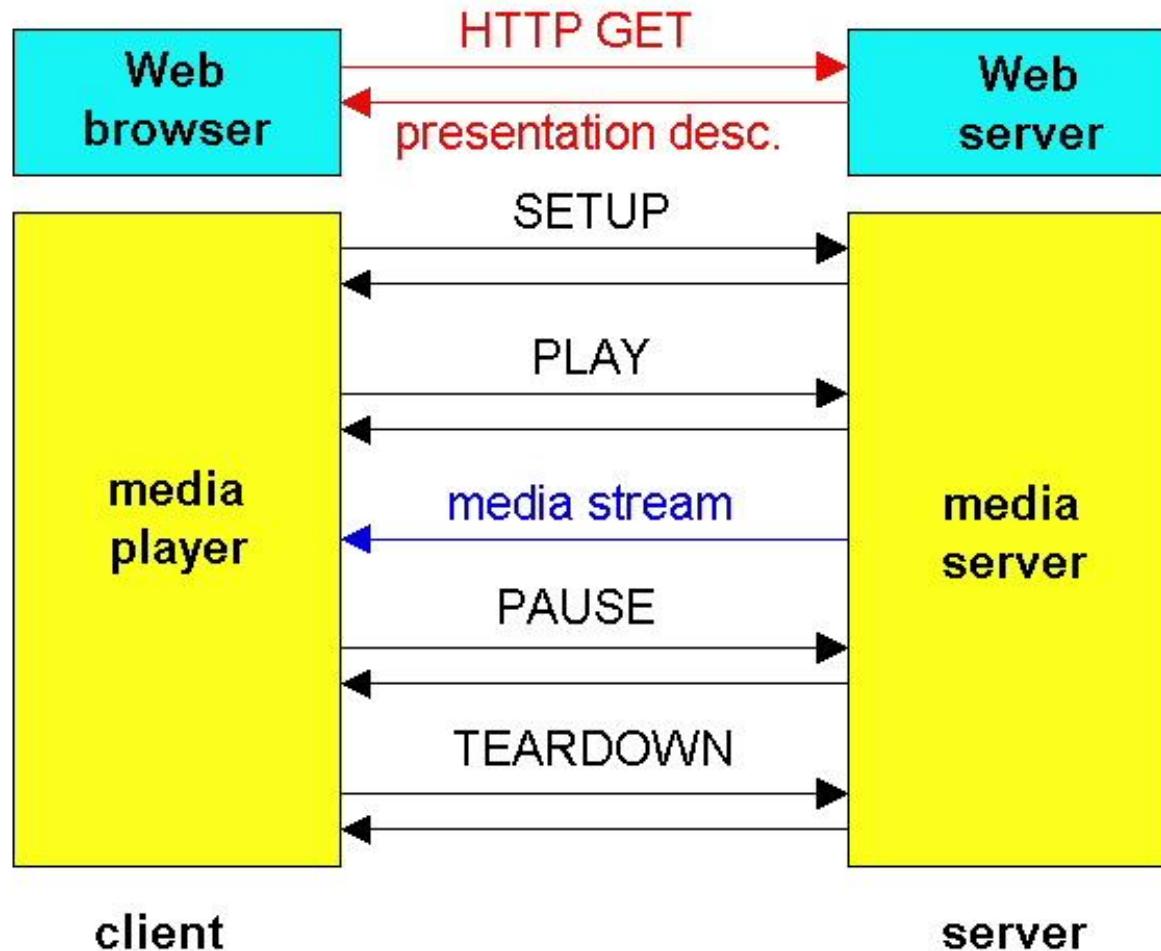
Scenario:

- ❑ metafile communicated to web browser
- ❑ browser launches player
- ❑ player sets up an RTSP control connection, data connection to streaming server

Metafile Example

```
<title>Twister</title>
<session>
  <group language=en lipsync>
    <switch>
      <track type=audio
            e="PCMU/8000/1"
            src = "rtsp://audio.example.com/twister/audio.en/lofi">
      <track type=audio
            e="DVI4/16000/2" pt="90 DVI4/8000/1"
            src="rtsp://audio.example.com/twister/audio.en/hifi">
    </switch>
    <track type="video/jpeg"
          src="rtsp://video.example.com/twister/video">
  </group>
</session>
```

RTSP Operation



RTSP Exchange Example

C: SETUP rtsp://audio.example.com/twister/audio RTSP/1.0

 Transport: rtp/udp; compression; port=3056; mode=PLAY

S: RTSP/1.0 200 1 OK

 Session 4231

C: PLAY rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0

 Session: 4231

 Range: npt=0-

C: PAUSE rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0

 Session: 4231

 Range: npt=37

C: TEARDOWN rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0

 Session: 4231

S: 200 3 OK

Advantages of using TCP for streaming

□ **Advantage of TCP over UDP**

- Can use of HTTP over TCP
- Traverses most firewalls and NATs
 - Many firewalls setup to block UDP
- Companies using TCP to stream video:
 - YouTube, NetFlix, Vimeo

□ **Advantage of using TCP over RSVP media server (e.g. RealMedia, Apple's QuickTime)**

- Less complex to setup
- Uses existing Web client and server software
- Cost of RSVP media server
 - “call representative for a quote” (never a good sign!)
 - (<http://www.realnetworks.com/helix/helix-server-editions.aspx>)

Disadvantages of using TCP

- ❑ Horrendously inefficient
 - HTTP overhead
 - HTTP is text based
 - TCP overhead
- ❑ OK for stored streaming, but not so good for live streaming
 - Prefetching always adds delay
 - For effective live streaming, need dedicated protocol over UDP

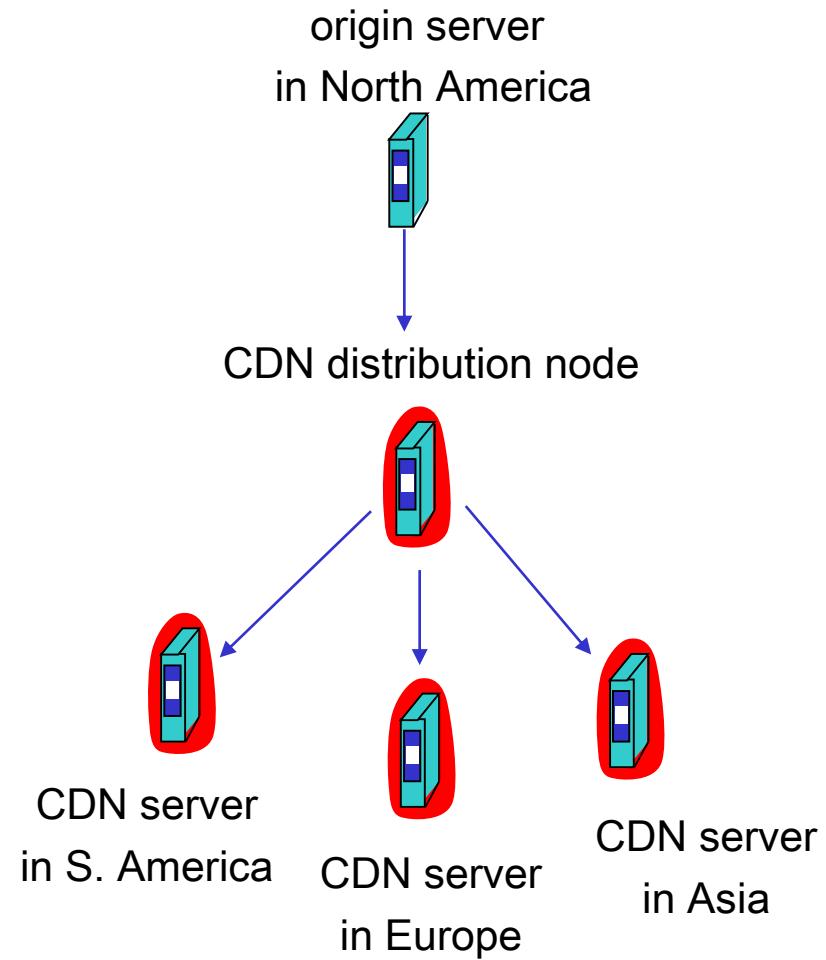
Chapter 3 outline

- 3.1 Transport-layer services and Protocols
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Connection-oriented transport: TCP
 - segment structure
 - reliable data transfer
 - flow control
 - connection management
 - TCP congestion control
- 3.5 Multimedia over TCP and UDP
 - Multimedia Networking Applications
 - Streaming stored multimedia over TCP and UDP
 - UDP or TCP
 - HTTP Streaming
 - **Content Distribution Networks**

3.5.6 Content Distribution Networks (CDNs)

Content replication

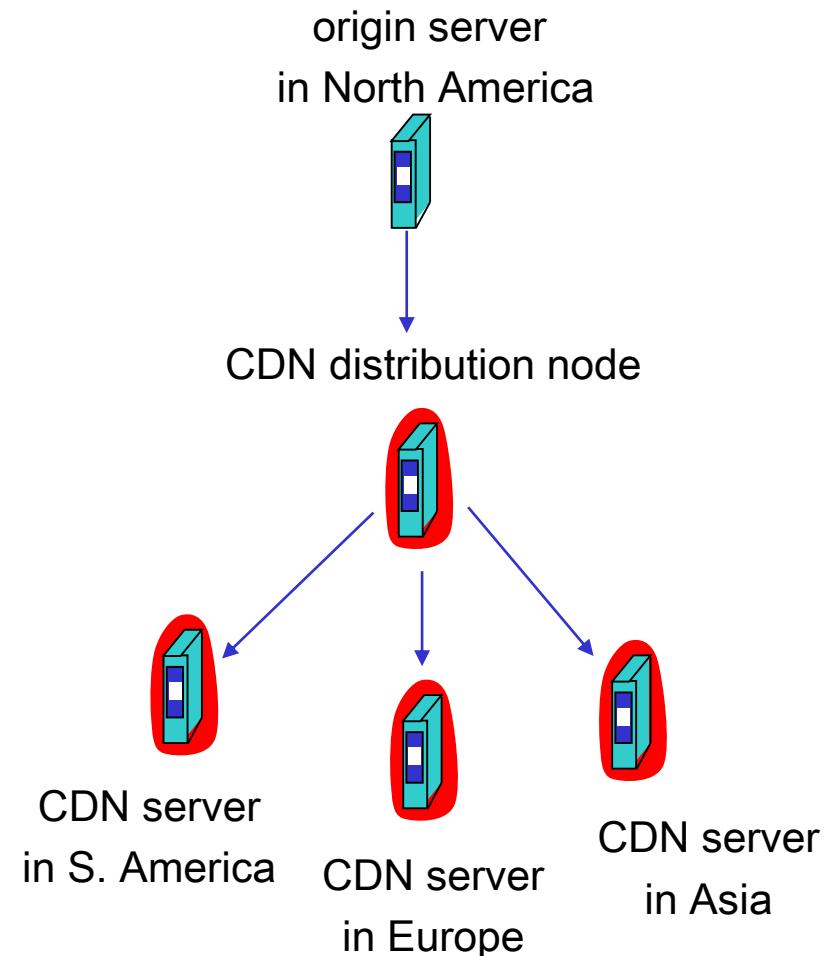
- ❑ challenging to stream large files (e.g., video) from single origin server in real time
- ❑ *solution:* replicate content at hundreds of servers throughout Internet
 - content downloaded to CDN servers ahead of time
 - *placing content “close” to user avoids impairments (loss, delay) of sending content over long paths*
 - CDN server typically in edge/access network



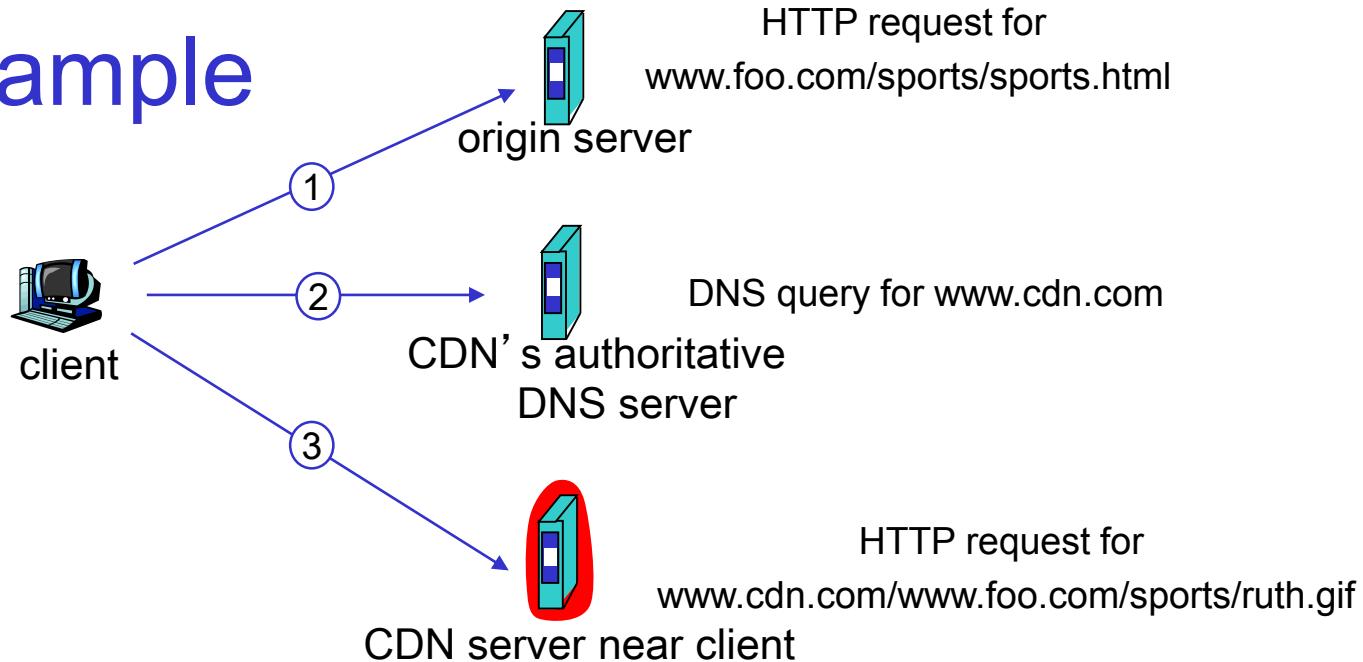
Content Distribution Networks (cont)

Content replication

- ❑ Content provider (e.g., CNN) is the CDN's (e.g., Akamai) customer
- ❑ CDN replicates customers' content in CDN servers.
- ❑ When provider updates content, CDN updates servers



CDN example



origin server (www.foo.com)

- distributes HTML
- replaces:
`http://www.foo.com/sports.ruth.gif`
with
`http://www.cdn.com/www.foo.com/sports/ruth.gif`

CDN company (cdn.com)

- distributes gif files
- uses its authoritative DNS server to route redirect requests

More about CDNs

routing requests

- CDN creates a “map”, indicating distances from leaf ISPs and CDN nodes
- when query arrives at authoritative DNS server:
 - server determines ISP from which query originates
 - uses “map” to determine best CDN server
- CDN nodes create application-layer overlay network

Chapter 3: Summary

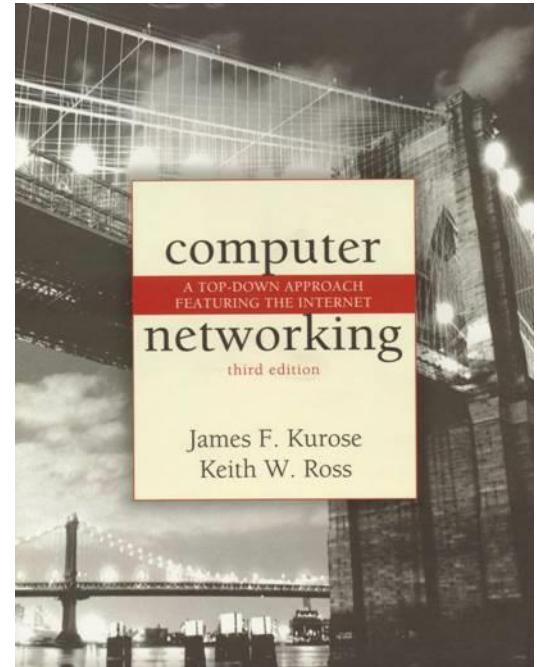
- Principles behind transport layer services:
 - multiplexing, demultiplexing
 - reliable data transfer
 - flow control
 - congestion control
 - Instantiation and implementation in the Internet
 - UDP
 - TCP
 - Multimedia at the Transport Layer
 - TCP vs UDP
 - HTTP Streaming
 - Content Distribution Networks
- Next:**
- Leaving the network “edge” (application, transport layers)
 - Deep into the network “core”

Computer Networks

Chapter 4 Network Layer

Professor R Simon Sherratt

Adapted and updated from the original work
of Dr. Michael Evans



*Computer
Networking: A Top
Down Approach
Featuring the
Internet,
Jim Kurose, Keith
Ross
Addison-Wesley*

Chapter 4: Network Layer

Chapter goals:

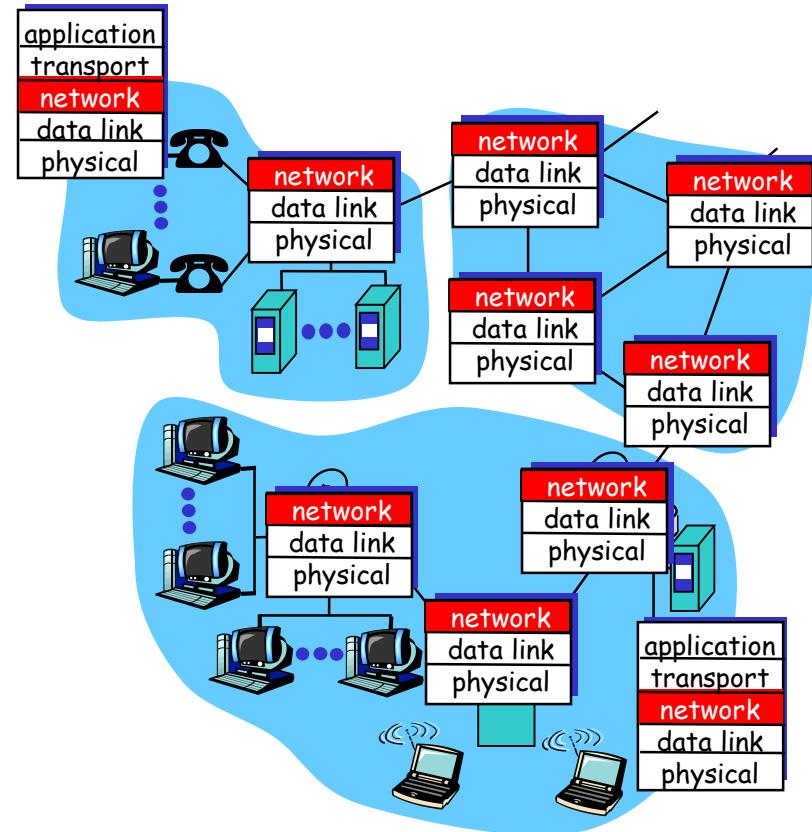
- understand principles behind network layer services:
 - network layer service models
 - forwarding versus routing
 - routing (path selection)
 - dealing with scale
 - IPv4 and IPv6
- instantiation, implementation in the Internet

Chapter 4: Network Layer

- 4.1 Introduction
- 4.2 Virtual Circuit vs Datagram networks
- 4.3 IPv4: Internet Protocol
 - Datagram format
 - IPv4 Fragmentation
 - IPv4 Addressing
 - Classful Addressing
 - Subnetting
 - Extended Addressing Schemes
 - CIDR
 - NAT
 - DHCP
- 4.4 IPv6
 - Enhancements
 - IPv6 Header
 - Flow
 - IPv6 Extension Headers
 - Hop by Hop Options
 - Routing Options
 - Fragmentation
 - IPv6 Addressing
 - Subnetting
 - Mobility
 - Security

4.1 Introduction to the Network layer

- Purpose: to transport TCP segment from sending to receiving host
- **on sending side**, encapsulate segments into datagrams
- **on receiving side**, deliver segments up to the transport layer
- Network layer protocols in every host and router
- Router is key
 - examines header fields in all IP datagrams passing through it



4.1.1 Three Key Network-Layer Functions

Forwarding

- Move packets from router's input to appropriate router output

Routing

- Determine route taken by packets from source to dest.
- Uses *routing algorithms*

Connection Setup

- Only in some network architectures

Analogy:

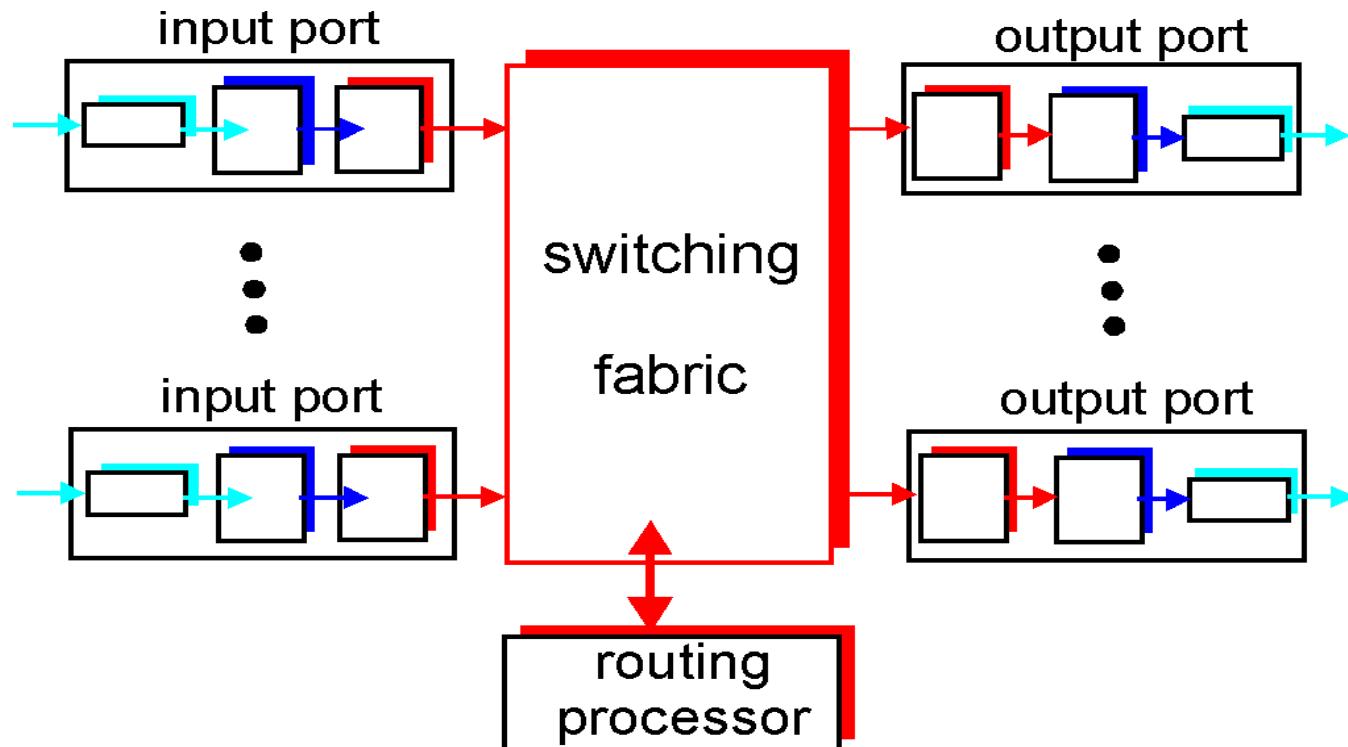
Routing: process of planning trip from source to dest

Forwarding: process of getting through single interchange

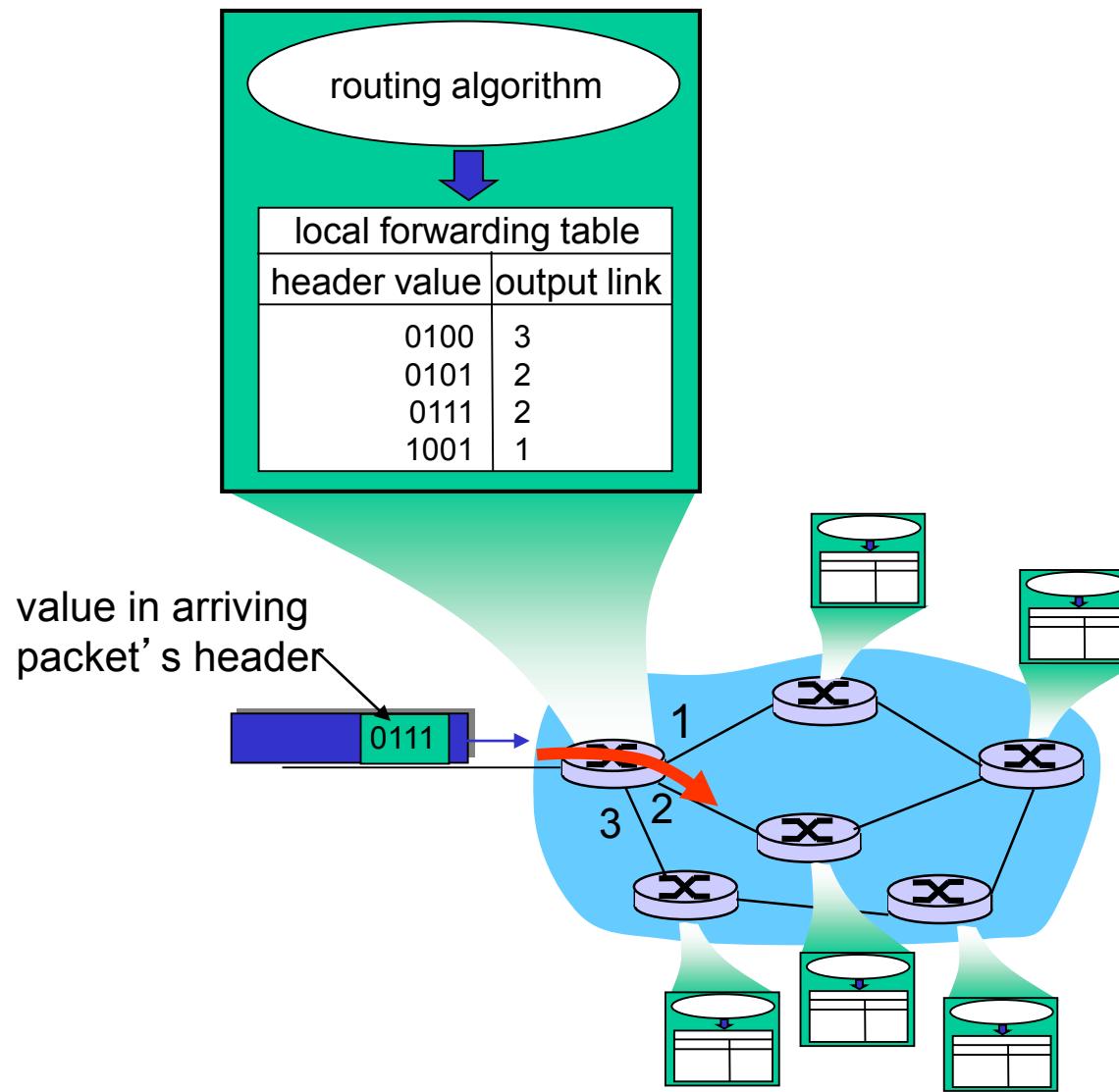
Router Architecture Overview

Two key router functions:

- ❑ run **routing algorithms/protocol** (RIP, OSPF, BGP)
- ❑ **forwarding** datagrams from incoming to outgoing link



Interplay between routing and forwarding



Connection setup

- 3rd important function in *some* network architectures:
 - e.g. ATM, frame relay, X.25
- Before datagrams flow, two end hosts *and* intervening routers establish **virtual connection**
 - routers get involved
- Network vs transport layer connection service:
 - **Network**: between two hosts (may also involve intervening routers in case of VCs)
 - **Transport**: between two processes
- *IP does NOT establish virtual connections*

4.1.2 Network Service Model

Q: What *service model* should be used for the “channel” transporting datagrams from sender to receiver?

Example services for individual datagrams:

- guaranteed delivery
- guaranteed delivery with less than 40 msec delay

Example services for a flow of datagrams:

- in-order datagram delivery
- guaranteed minimum bandwidth to flow
- restrictions on changes in inter-packet spacing

Network layer service models:

Network Architecture	Service Model	Guarantees ?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

Chapter 4: Network Layer

- 4.1 Introduction
- 4.2 Virtual Circuit vs Datagram networks
- 4.3 IPv4: Internet Protocol
 - Datagram format
 - IPv4 Fragmentation
 - IPv4 Addressing
 - Classful Addressing
 - Subnetting
 - Extended Addressing Schemes
 - CIDR
 - NAT
 - DHCP
- 4.4 IPv6
 - Enhancements
 - IPv6 Header
 - Flow
 - IPv6 Extension Headers
 - Hop by Hop Options
 - Routing Options
 - Fragmentation
 - IPv6 Addressing
 - Subnetting
 - Mobility
 - Security

4.2 Virtual Circuit vs Datagram Networks

- **Datagram** network provides network-layer connectionless service
- **Virtual Circuit (VC)** network provides network-layer connection service
- Both analogous to the transport-layer services, but:
 - **service**: host-to-host
 - **no choice**: network provides one or the other
 - **implementation**: in network core

4.2.1 Virtual Circuits

“Source-to-dest path behaves much like telephone circuit”

- performance-wise
- network actions along source-to-dest path

- Call setup, teardown for each call *before* data can flow
- Each packet carries VC identifier (not destination host address)
- *Every* router on source-dest path maintains “state” for each passing connection
- Link, router resources (bandwidth, buffers) may be *allocated* to VC (dedicated resources = predictable service)

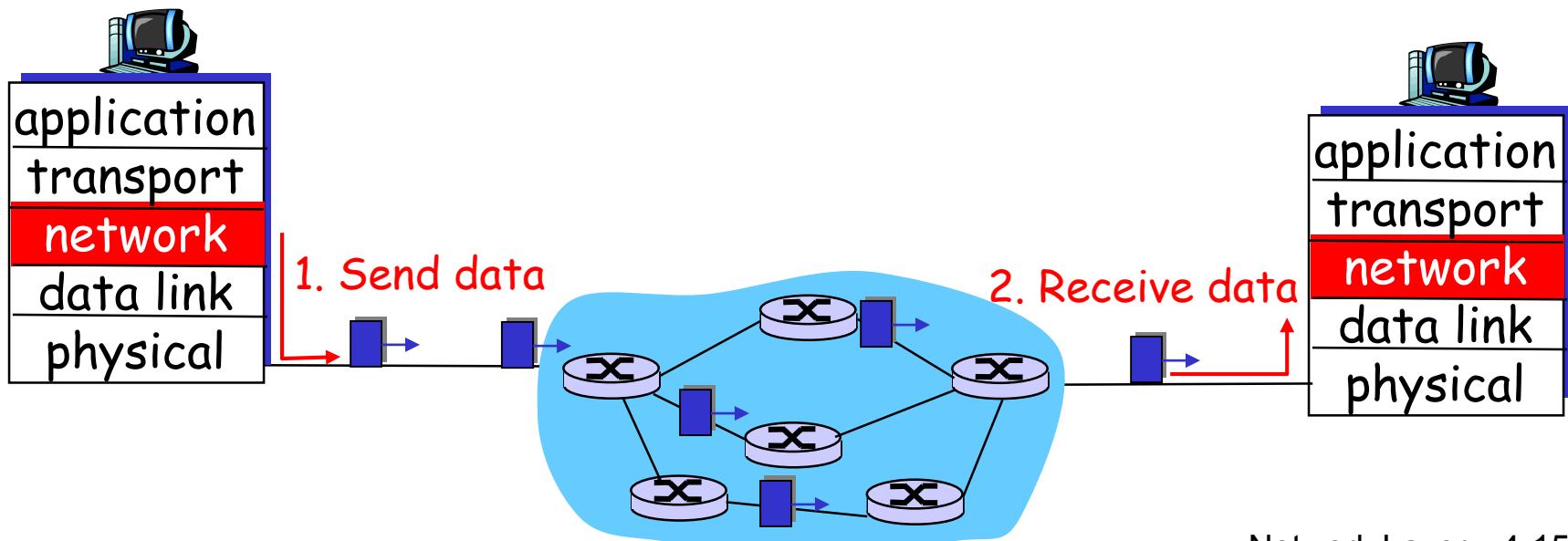
VC implementation

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 VC number (rather than dest address)
- VC number can be changed on each link.
 - New VC number comes from forwarding table

4.2.2 Datagram Networks

- ❑ No call setup at network layer
- ❑ Routers: no state about end-to-end connections
 - No network-level concept of “connection”
- ❑ Packets forwarded using destination host address
 - Packets between same source-dest pair may take different paths



Datagram vs VC Network

Internet (Datagram)

- Data exchange among computers
 - “Elastic” service, no strict timing req.
- “Smart” end systems (computers)
 - Can adapt, perform control, error recovery
 - Simple inside network, complexity at “edge”
- Must cope with many link types
 - Different characteristics
 - Uniform service difficult

ATM (VC)

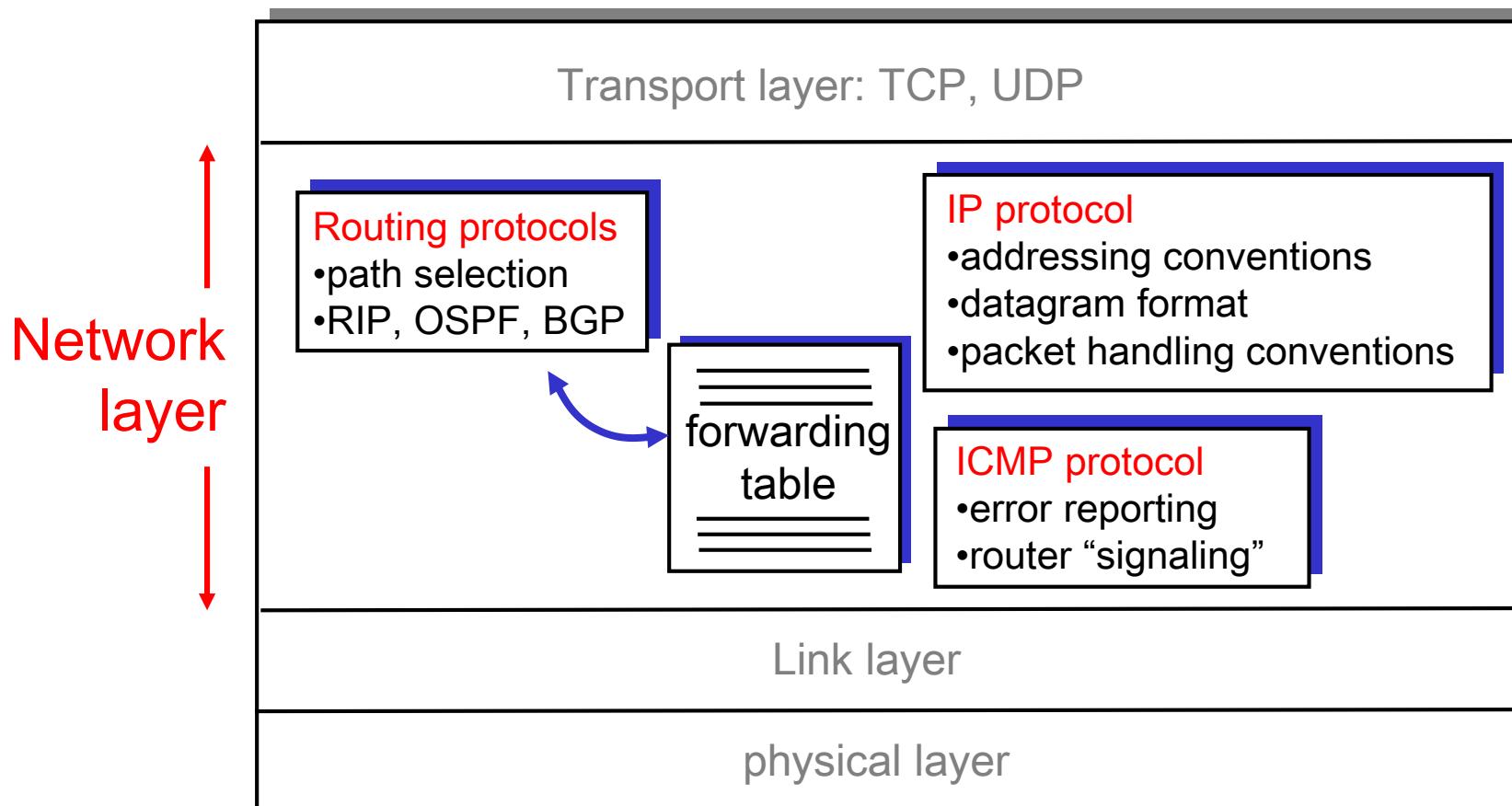
- Evolved from telephony
- Human conversation:
 - Strict timing, reliability requirements
 - Need for guaranteed service
- “Dumb” end systems
 - Telephones
 - Complexity inside network

Chapter 4: Network Layer

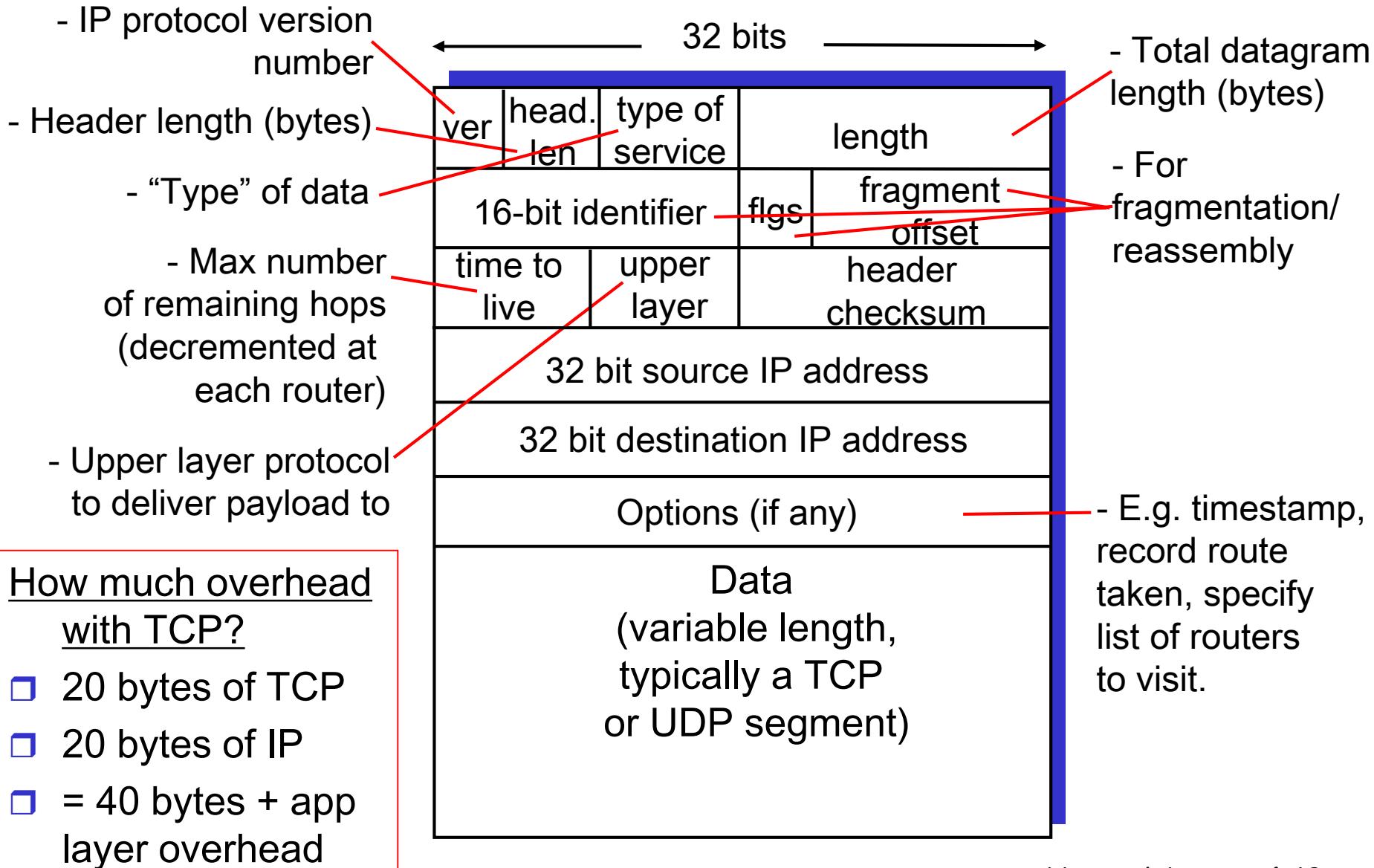
- 4.1 Introduction
- 4.2 Virtual Circuit vs Datagram networks
- **4.3 IPv4: Internet Protocol**
 - Datagram format
 - IPv4 Fragmentation
 - IPv4 Addressing
 - Classful Addressing
 - Subnetting
 - Extended Addressing Schemes
 - CIDR
 - NAT
 - DHCP
- 4.4 IPv6
 - Enhancements
 - IPv6 Header
 - Flow
 - IPv6 Extension Headers
 - Hop by Hop Options
 - Routing Options
 - Fragmentation
 - IPv6 Addressing
 - Subnetting
 - Mobility
 - Security

4.3 The Internet Network layer

Host, router network layer functions:



IPv4 Datagram Format



Chapter 4: Network Layer

- 4.1 Introduction
- 4.2 Virtual Circuit vs Datagram networks
- 4.3 IPv4: Internet Protocol
 - Datagram format
 - IPv4 Fragmentation
 - IPv4 Addressing
 - Classful Addressing
 - Subnetting
 - Extended Addressing Schemes
 - CIDR
 - NAT
 - DHCP
- 4.4 IPv6
 - Enhancements
 - IPv6 Header
 - Flow
 - IPv6 Extension Headers
 - Hop by Hop Options
 - Routing Options
 - Fragmentation
 - IPv6 Addressing
 - Subnetting
 - Mobility
 - Security

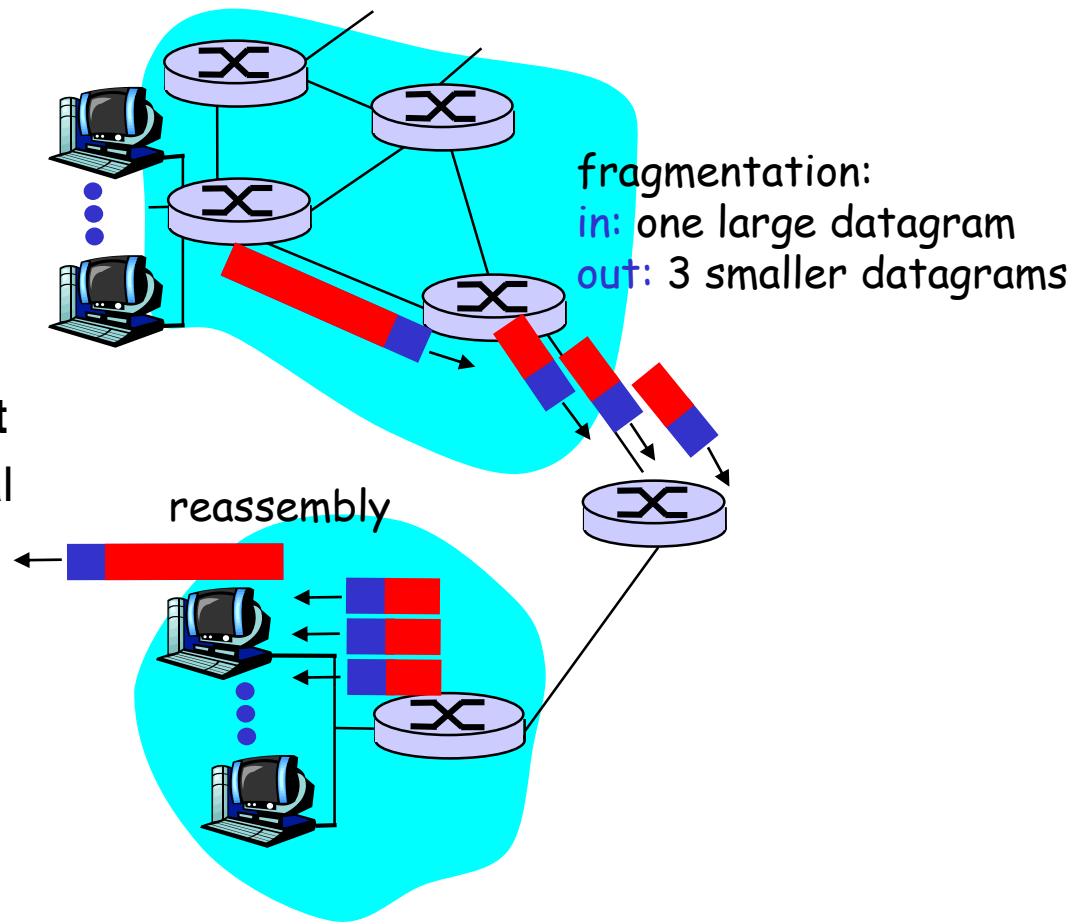
4.3.1 IP Fragmentation & Reassembly

□ At Data Link Layer:

- Network links have MTU (max.transfer size) - largest possible link-level frame.
- Different link types, different MTUs

□ Large IP datagram must be divided (“fragmented”) within net

- One datagram becomes several datagrams
- *Reassembled* only at final destination
- IP header bits used to identify and order related fragments



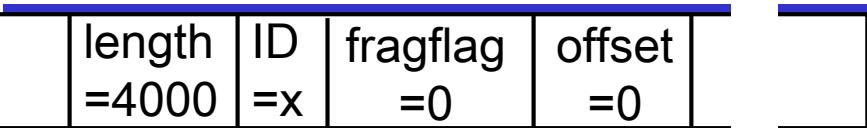
IP Fragmentation and Reassembly

Example

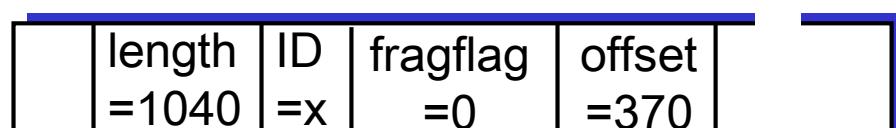
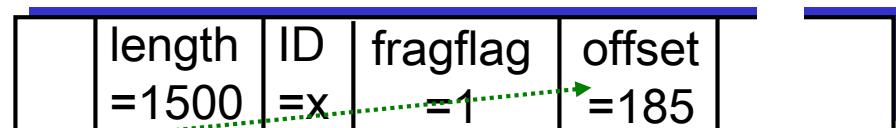
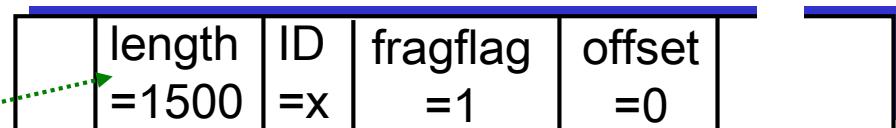
- ❑ 4000 byte datagram
- ❑ MTU = 1500 bytes

1480 bytes in data field

offset =
 $1480/8$



One large datagram becomes several smaller datagrams



Chapter 4: Network Layer

- 4.1 Introduction
- 4.2 Virtual Circuit vs Datagram networks
- 4.3 IPv4: Internet Protocol
 - Datagram format
 - IPv4 Fragmentation
 - **IPv4 Addressing**
 - Classful Addressing
 - Subnetting
 - Extended Addressing Schemes
 - CIDR
 - NAT
 - DHCP
- 4.4 IPv6
 - Enhancements
 - IPv6 Header
 - Flow
 - IPv6 Extension Headers
 - Hop by Hop Options
 - Routing Options
 - Fragmentation
 - IPv6 Addressing
 - Subnetting
 - Mobility
 - Security

4.3.2 IPv4 Addressing

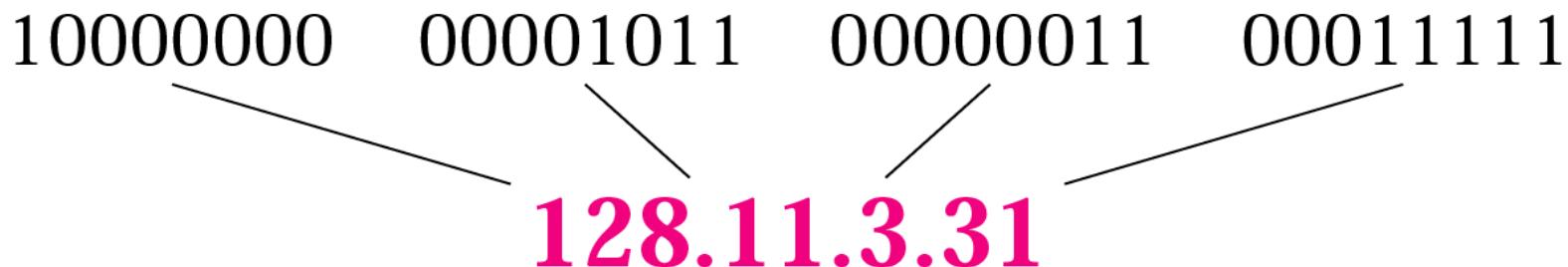
- ❑ **IPv4 Address**

- Encodes **network number and host number**
- 32 bits long
- Used in source and destination address fields

- ❑ **IP address refers to network interface, not host**

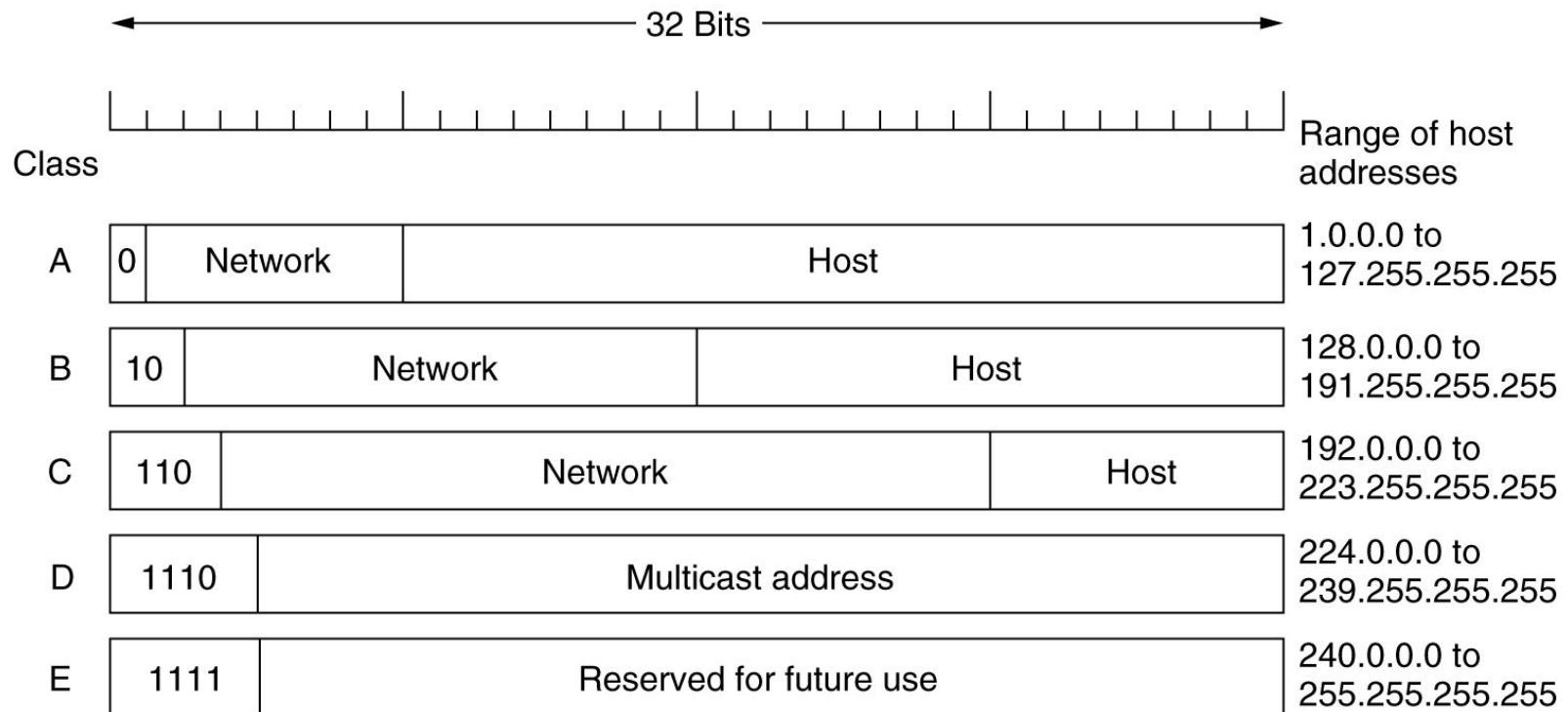
- If host is on two networks, it has two IP addresses

Example IP address



- Written in **Dotted Decimal**
 - Each byte represented as a decimal
 - 127.0.01 = loopback address (localhost)
- Original IP address employed **Classful Addressing** :
 - Class A: 128 networks each with 16 million hosts
 - Class B: 16,384 networks, 64,000 hosts
 - Class C: 2 million networks, 256 hosts

4.3.2.1 IPv4 Classful Addressing



- Network id (*Netid*) identifies network – given to you by ICANN (Internet Corp. for Assigned Names and Numbers)
- Host id (*Hostid*) identifies host within that network (assigned by you)
- All hosts on same network must have same network number

IP Addresses - Class A

- 32 bit global internet address
- Network part and host part
- Class A
 - Start with binary 0
 - All 0 mean this host
 - 01111111 (127) reserved for loopback
 - Range 1.x.x.x to 126.x.x.x
 - All allocated**

IP Addresses - Class B

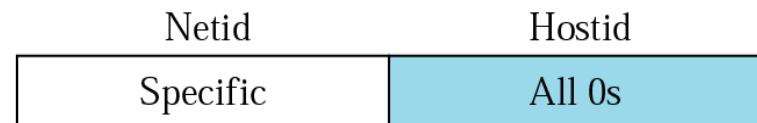
- ❑ Start 10
- ❑ Range 128.x.x.x to 191.x.x.x
- ❑ Second Octet also included in network address
- ❑ $2^{14} = 16,384$ class B addresses
- ❑ **All allocated**

IP Addresses - Class C

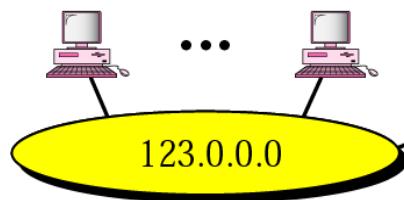
- ❑ Start 110
- ❑ Range 192.x.x.x to 223.x.x.x
- ❑ Second and third octet also part of network address
- ❑ $2^{21} = 2,097,152$ addresses
- ❑ **Nearly all allocated**
 - See IPv6

Network address, netid and hostid

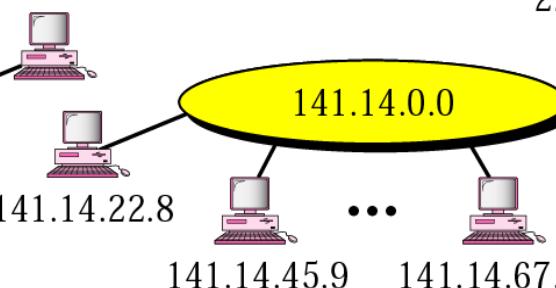
- Network address (netid) identifies network
- Host address (hostid) identifies host
- Netid specifies network; hostid set to 0



123.50.16.90 123.65.7.34

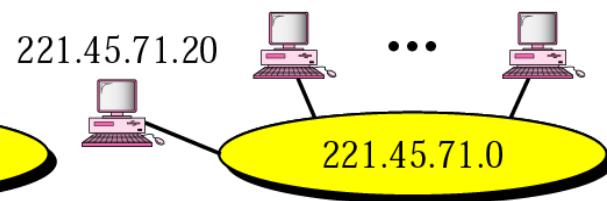


123.90.123.4



a. Class A

221.45.71.64 221.45.71.126



c. Class C

b. Class B

4.3.2.2 Subnets and Subnet Masks

□ The Subnet

- A network can be split into many parts for internal use
- Still seen as one network to rest of Internet
- Each internal network is called a *Subnet*

□ The need for Subnets

- Subnets allow arbitrary complexity of internetworked LANs within an organization
- Each LAN becomes a subnet
- Set of all LANs seen as one network
- Insulates overall Internet from internal complexity of an organization's network

Subnetting a network

- Each LAN assigned a **subnet number**
- Host portion of address (i.e. *hostid*) partitioned into **subnet number** and **host number**
- **Subnet mask** indicates which bits are subnet number and which are host number
- Local routers route within subnetted network

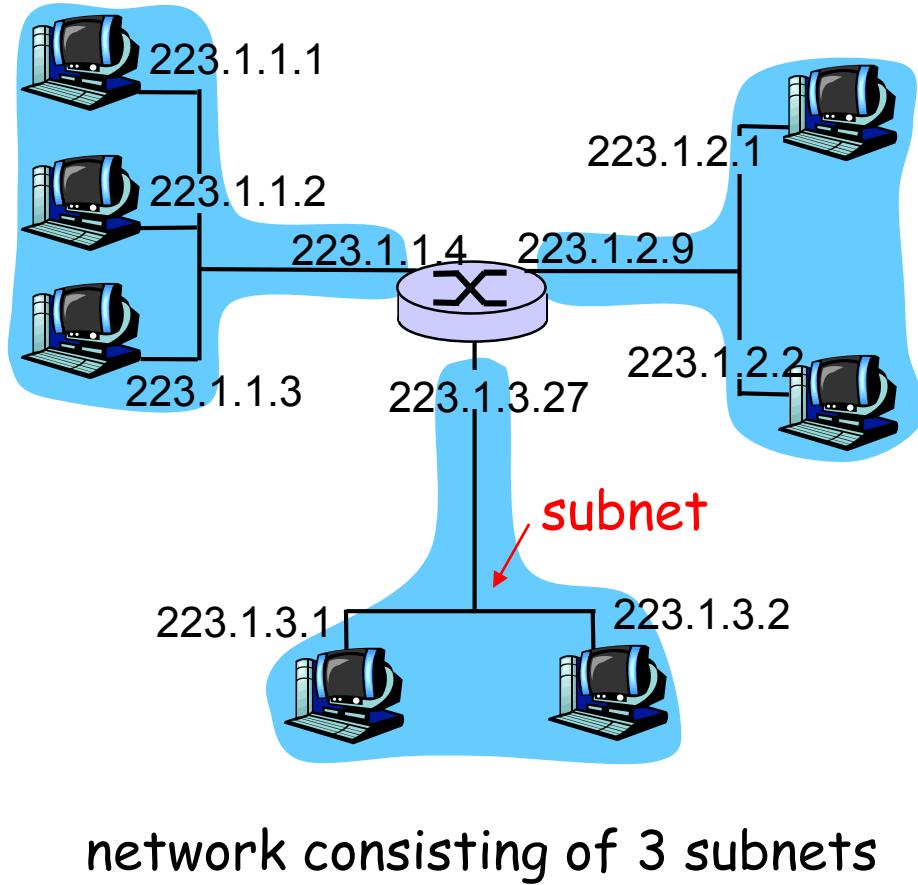
Subnet Example

□ IP address:

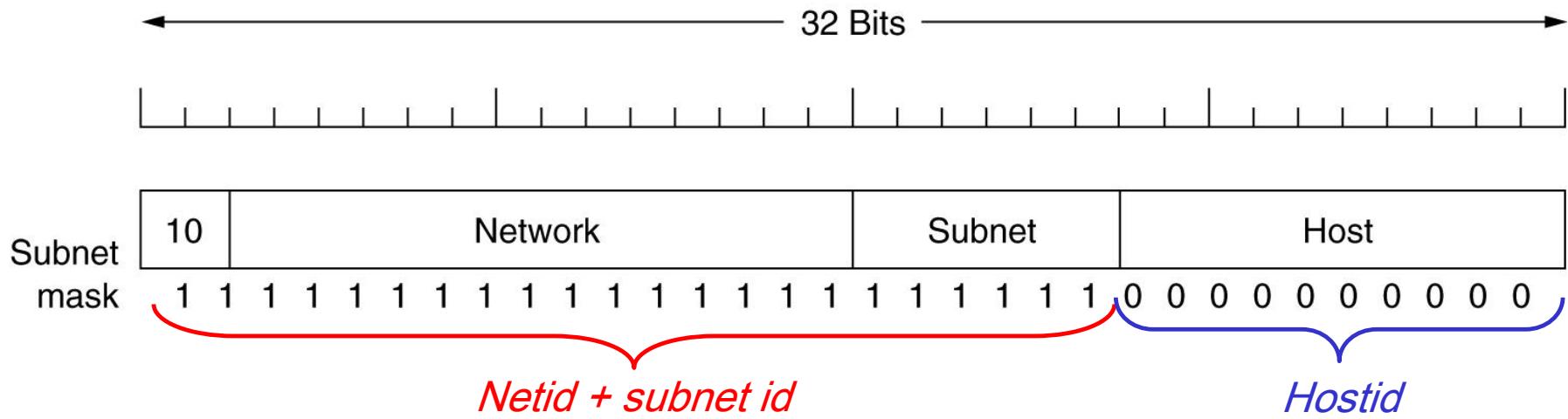
- subnet part (high order bits)
- host part (low order bits)

□ Subnet

- device interfaces with same subnet part of IP address
- devices on same subnet can physically reach each other without intervening router



A class B network subnetted into 64 subnets.



- ❑ Bits are taken away from the host number to identify the subnet number
- ❑ Network number always remains the same
- ❑ **SUBNET MASK** indicates split between (netid + subnet id) and hostid

The Subnet Mask

- ❑ Indicates split between (network + subnet id) and host
- ❑ Subnet mask is ANDed with IP address to remove host number
- ❑ Result retains *netid*, but sets *hostid* to 0

Subnet Example

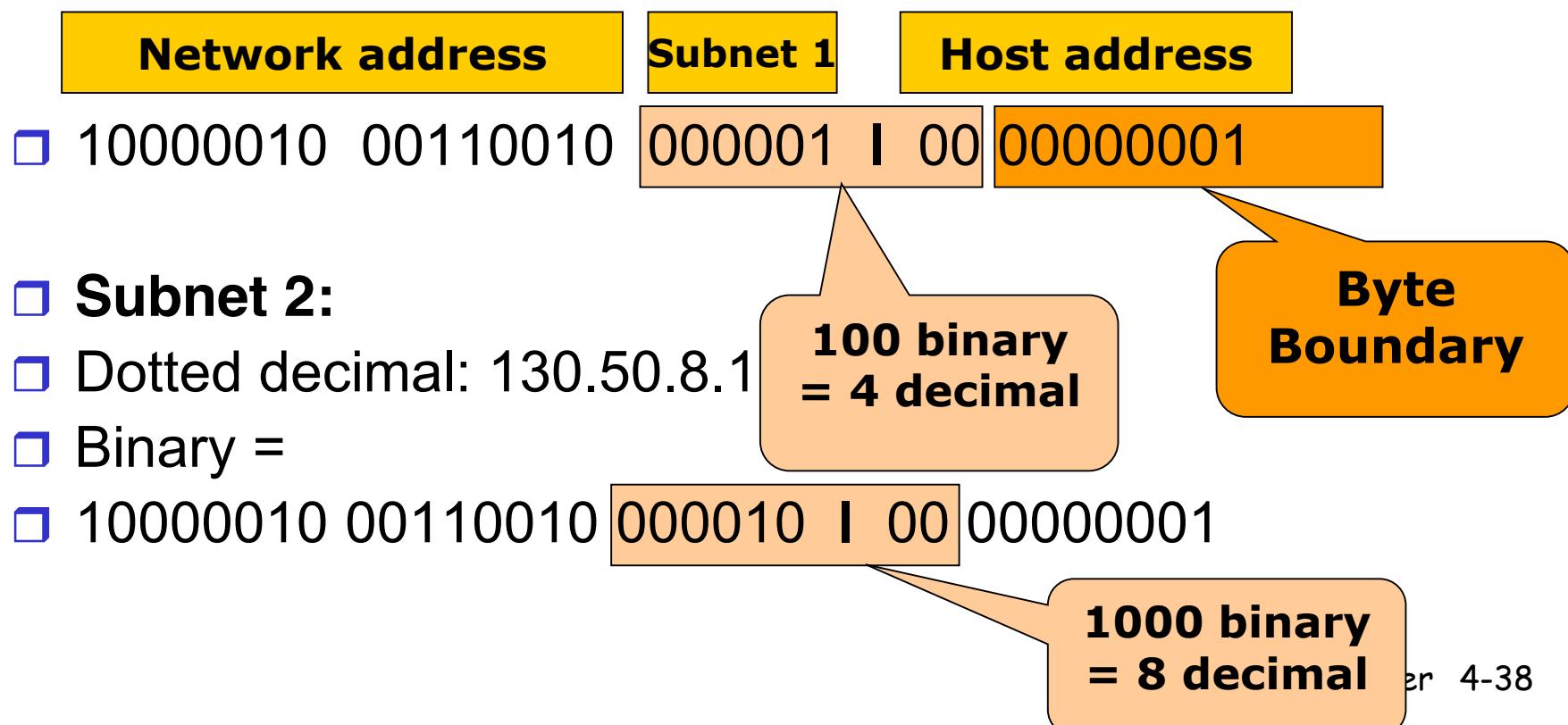
- *Calculate Subnet number when 10 bit host number is used*
 - IP address = 32 bits
 - For our example, use 130.50.4.1
 - 10 bit host number leaves first 22 bits for network and subnet
 - Address class identifies how many bits must be used for network
 - 130.50.4.1 = Class B
 - Class B = 16 bit network number
 - Therefore Subnet number = $22 - 16 = 6$ bits
 - 6 bits provides for 64 subnets (i.e. 2^6)

Subnet Example (cont.)

- Now Suppose we have 3 subnets: 1, 2, and 3
 - Subnets use IP addresses:
 - Subnet1: 130.50.4.1
 - Subnet2: 130.50.8.1
 - Subnet3: 130.50.12.1
 - Why not:
 - 130.50.4.1
 - 130.50.4.2
 - 130.50.4.3?

Subnets and IP addresses

- **Subnet 1:**
- Dotted decimal: 130.50.4.1
- Binary = 10000010 00110010 00000100 00000001



Extracting Subnet id from IP address

130.50.15.6

Subnet mask:
255.255.252.0

- 10000010 00110010 00001111 000000110
- 11111111 11111111 11111100 00000000
- 10000010 00110010 00001100 00000000

**IP Address
AND
Subnet
mask**

=

130 . 50 . 12 . 0

Subnet 3

Routing without Subnets

- Routers do not know the exact location of a host for which a packet of information is destined
 - Routers only know what network the host is a member of
 - Information in routing table determines how to get the packet to the destination host's network
 - After the packet is delivered to the destination's network, the packet is delivered to the appropriate host

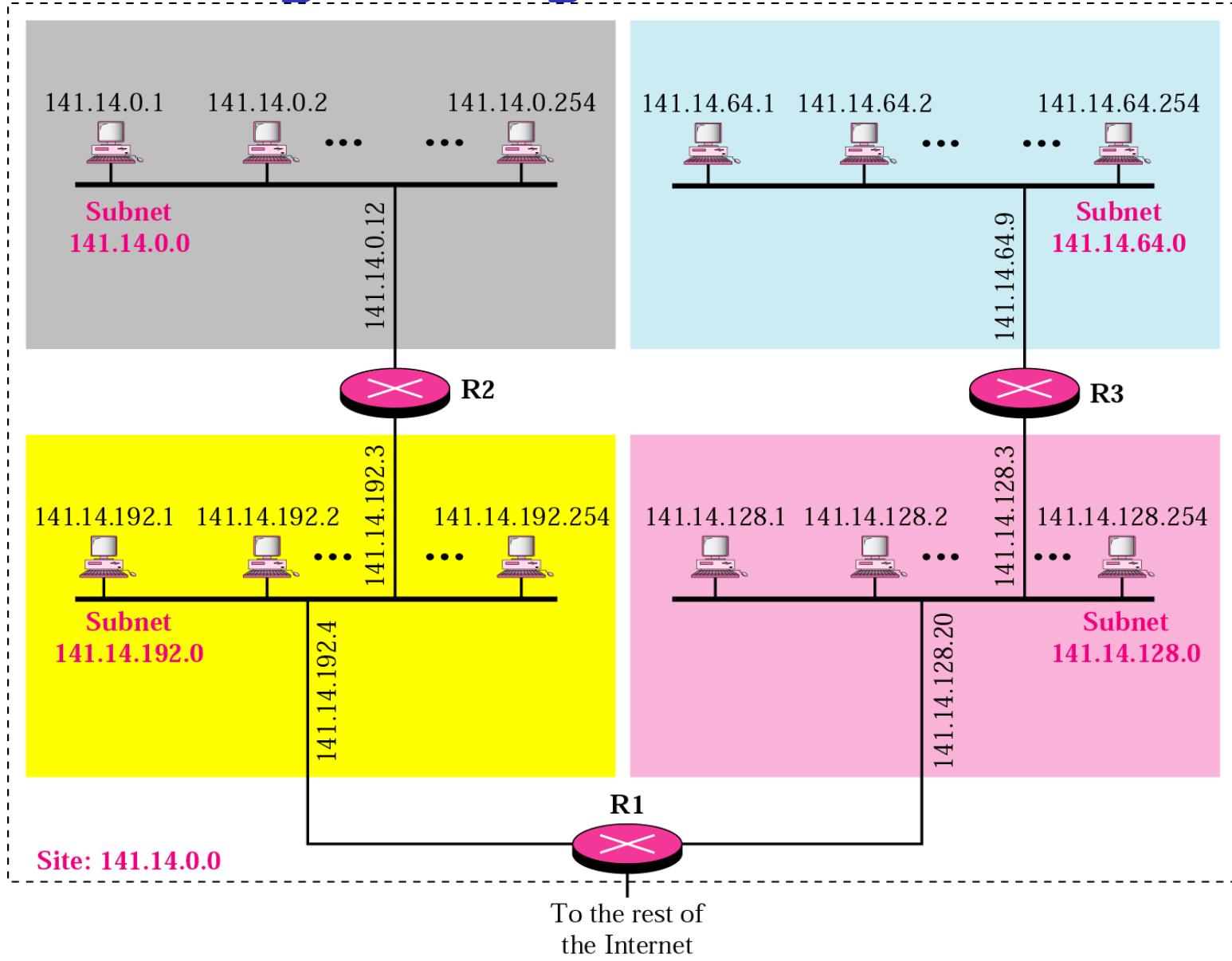
Routing without Subnets (cont)

- Each router has a table containing IP addresses
- Some IP addresses refer to distant networks
- Some refer to hosts on router's own network (e.g. a LAN)
- When packet arrives:
 - Destination address is looked up
 - Netid is extracted based on class of address
 - If Netid is same as router's netid, packet is destined for a host on this router's LAN
 - Router forwards packet to that host
 - If Netid is different, it's for a distant network, so it's forwarded to appropriate router
 - Router table instructs router which other router to use

Routing with subnets

- Subnet mask tells the router whether a host is on the local subnet or on a remote network:
 - Router takes the destination IP address
 - ANDs it with the subnet mask
 - Result: subnet id: hostid set to 0
 - If it's the subnet id for the router's network, hostid is determined, and router forwards packet to host
 - Otherwise, routing tables tell router where to forward packet
 - to another subnet
 - i.e. router within same network managing different subnet
 - Or to distant network (i.e. some other router)

Routing Using Subnets



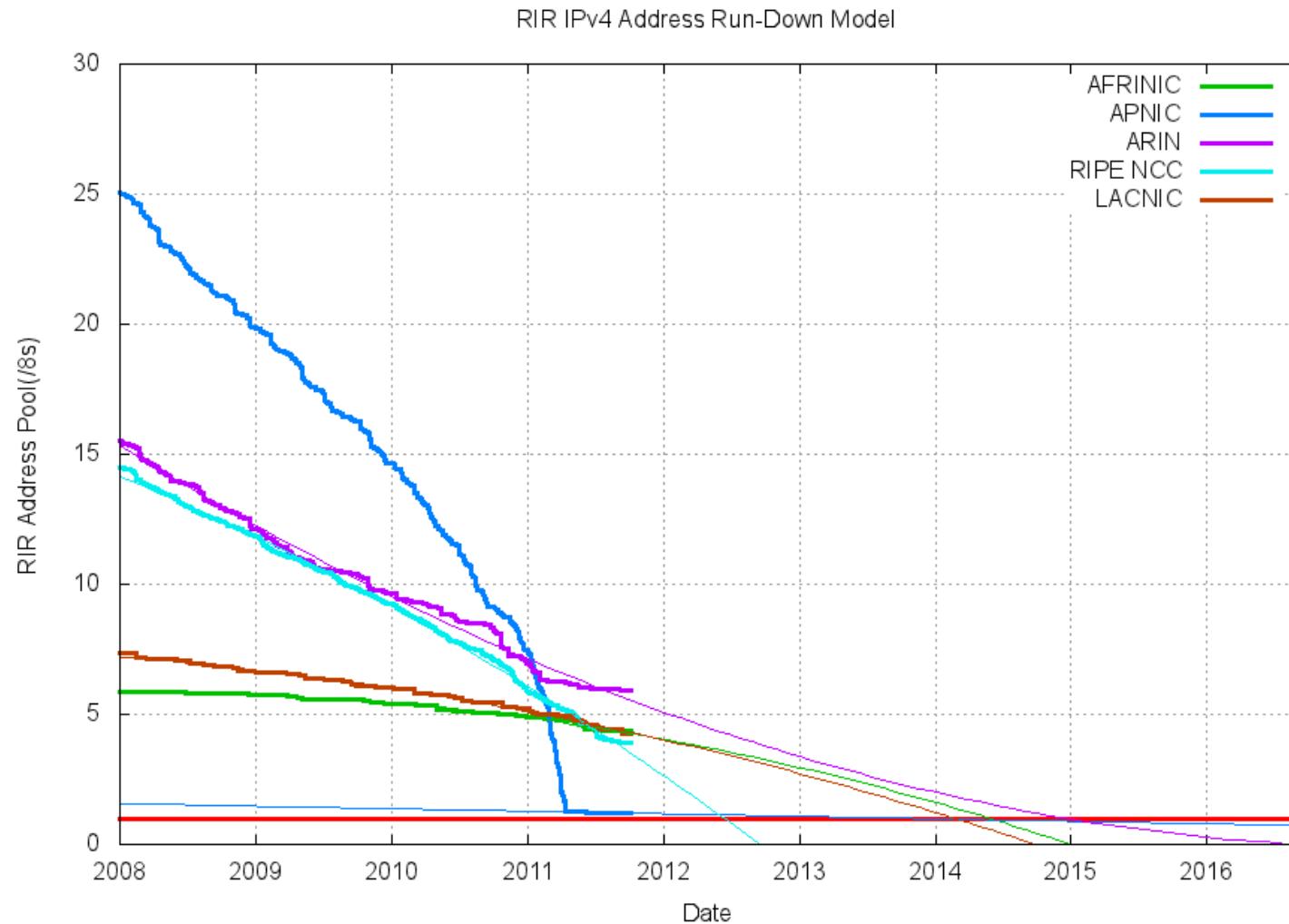
Chapter 4: Network Layer

- 4.1 Introduction
- 4.2 Virtual Circuit vs Datagram networks
- 4.3 IPv4: Internet Protocol
 - Datagram format
 - IPv4 Fragmentation
 - IPv4 Addressing
 - Classful Addressing
 - Subnetting
 - **Extended Addressing Schemes**
 - CIDR
 - NAT
 - DHCP
- 4.4 IPv6
 - Enhancements
 - IPv6 Header
 - Flow
 - IPv6 Extension Headers
 - Hop by Hop Options
 - Routing Options
 - Fragmentation
 - IPv6 Addressing
 - Subnetting
 - Mobility
 - Security

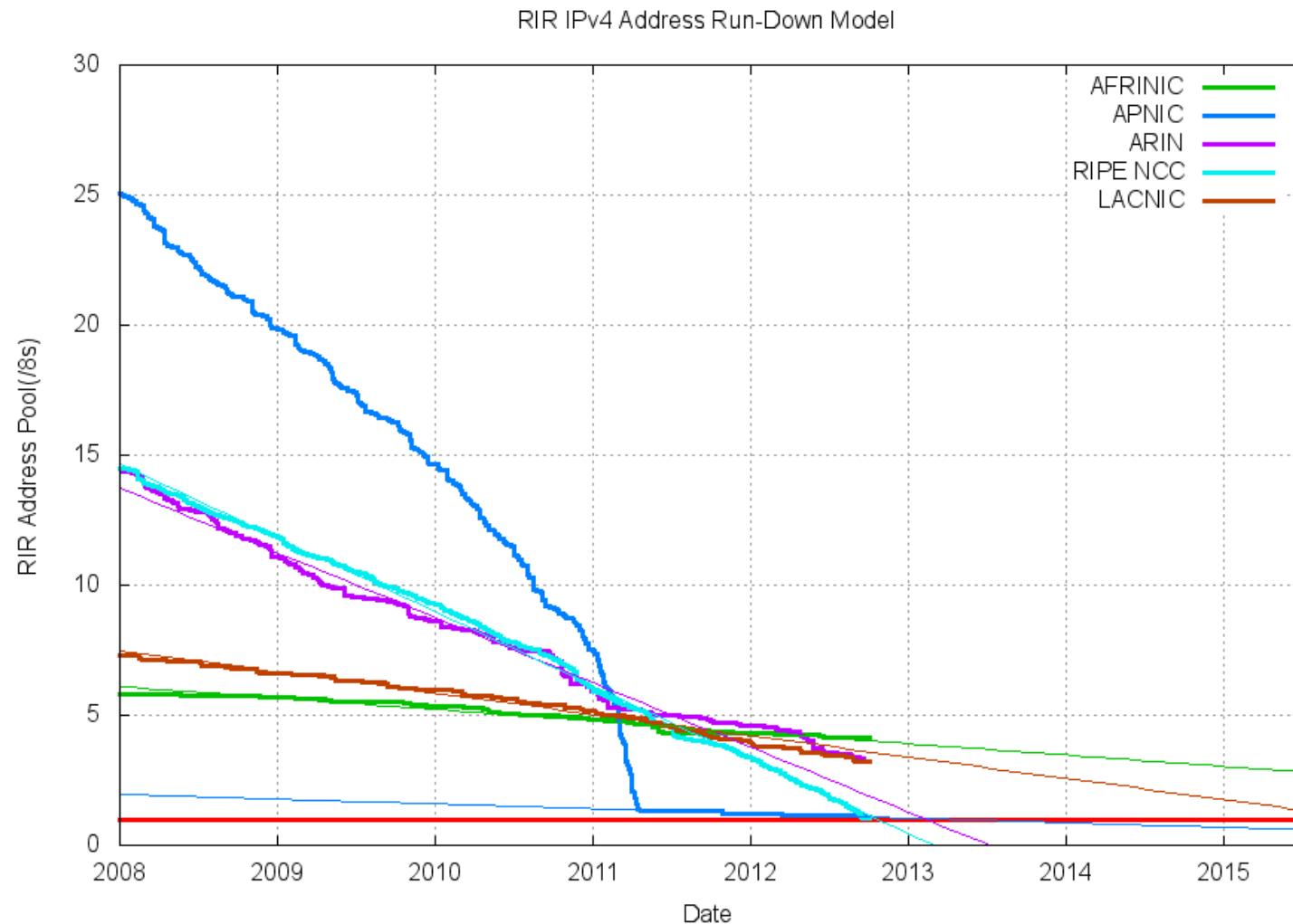
4.3.3 Alternative Addressing Schemes

- Problem: IPv4 Addresses running out
 - In 1987, 100,000th IP network predicted
 - Widely ridiculed as never happening
 - Happened in 1996
 - Now have 500,000!
 - IP has over 2 billion addresses
 - But address organization wastes millions of them
- As of 17/9/2009...
 - Just 719 days until all IP addresses are exhausted
<http://www.potaroo.net/tools/ipv4/index.html>

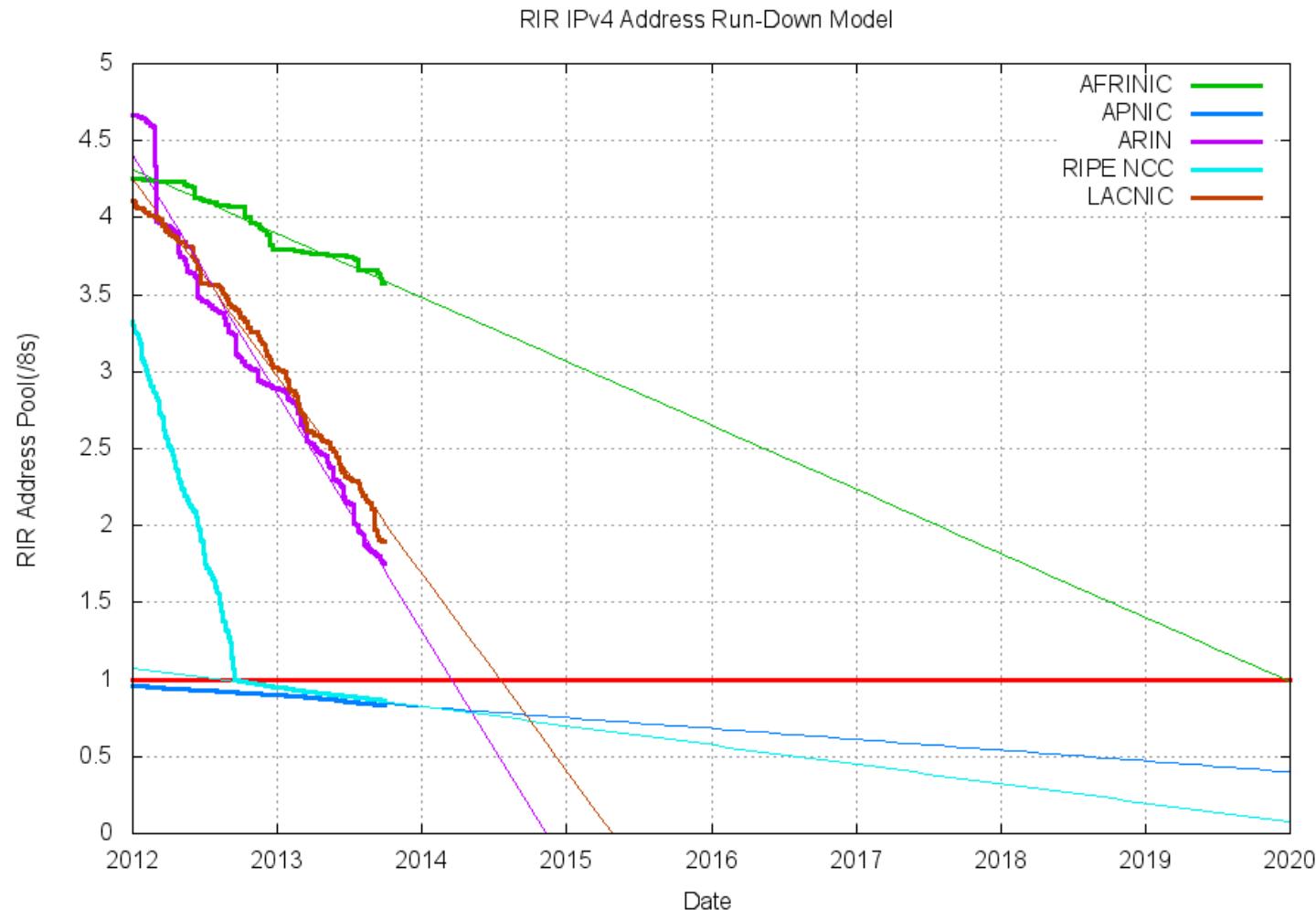
...and as of 2011:



...2012...



...and as of today



The Class B Problem

- ❑ A Class A network can address over 16,000,000 hosts
 - Too big for most
- ❑ Class C can address only 256
 - Too small
- ❑ Class B can address 65,536
 - Just right for future expansion!
- ❑ Problem:
 - Only 16,384 class B addresses exist
 - More than half of all class B networks only have 50 hosts!
 - Changing number of bits to Class B address no good either:
 - Routers need to know netid of all networks in the Internet
 - More netids, more space needed for router tables

Solution - Extended Addressing Schemes and IPv6

❑ **Extended Addressing Schemes**

- CIDR – Classless InterDomain Routing
- Network Address Translation

❑ **Internet Protocol v6**

- Enhancements
- IPv6 Addresses
- Extension Headers

4.3.3.1 CIDR – Classless InterDomain Routing

- CIDR drops classes
- Pre-CIDR: Network ID ended on 8-, 16, 24- bit boundary
- CIDR: Network ID can end at any bit boundary
- E.g. if a site needs 2000 addresses, it's given 2048 (2048 byte boundary)
- Requires 32 bit mask
- Single routing table used for all networks
- For each network, routing table stores:
 - Network's IP address
 - subnet mask for that network
 - outgoing line (which router to forward packet to)

CIDR (cont.)

- Generalizes concept of subnet addressing
- Now the Internet's address assignment strategy
 - i.e. CIDR replaced classful addressing
- New subnet notation
 - Subnet mask is written as */n* where n = number of 1s in mask
 - Example
 - Subnet mask = 255.255.252.0
 - Binary: 11111111 11111111 11111100 00000000
 - 22 x 1s, so subnet mask written as /22
 - Any IP address using this subnet mask written as:
 - a.b.c.d/22
 - e.g. **130.50.15.6/22**

CIDR Routing

- When a packet arrives:
 - Destination IP address extracted
 - Table is scanned line by line
 - For each line
 - Mask destination IP with that line's mask
 - Compare masked destination address with table entry
 - End for
- Multiple entries can exist with different subnet mask lengths
 - Longest length is used

CIDR Example

A set of IP address assignments.

University	First address	Last address	How many	Written as
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Available)	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

Routing tables

IP Addresses

1. 11000010 00011000 00000000 00000000 (Cm)
2. 11000010 00011000 00001000 00000000 (Ed)
3. 11000010 00011000 00010000 00000000 (Ox)

Subnet Masks

- 11111111 11111111 11111000 00000000 (Cm)
- 11111111 11111111 11111100 00000000 (Ed)
- 11111111 11111111 11110000 00000000 (Ox)

Destination
IP address

11000010 00011000 00010001 00000100

4.3.3.2 Network Address Translation (NAT)

- ❑ ISP might have class B (or /16) address
- ❑ Lets it address 65,534 hosts
- ❑ What if it has more customers?
 - ISP can dynamically assign IP address to dial-up customers
 - Then take it back once session finished
- ❑ But businesses prefer to stay online continuously
- ❑ Solution: **NAT**

How NAT works

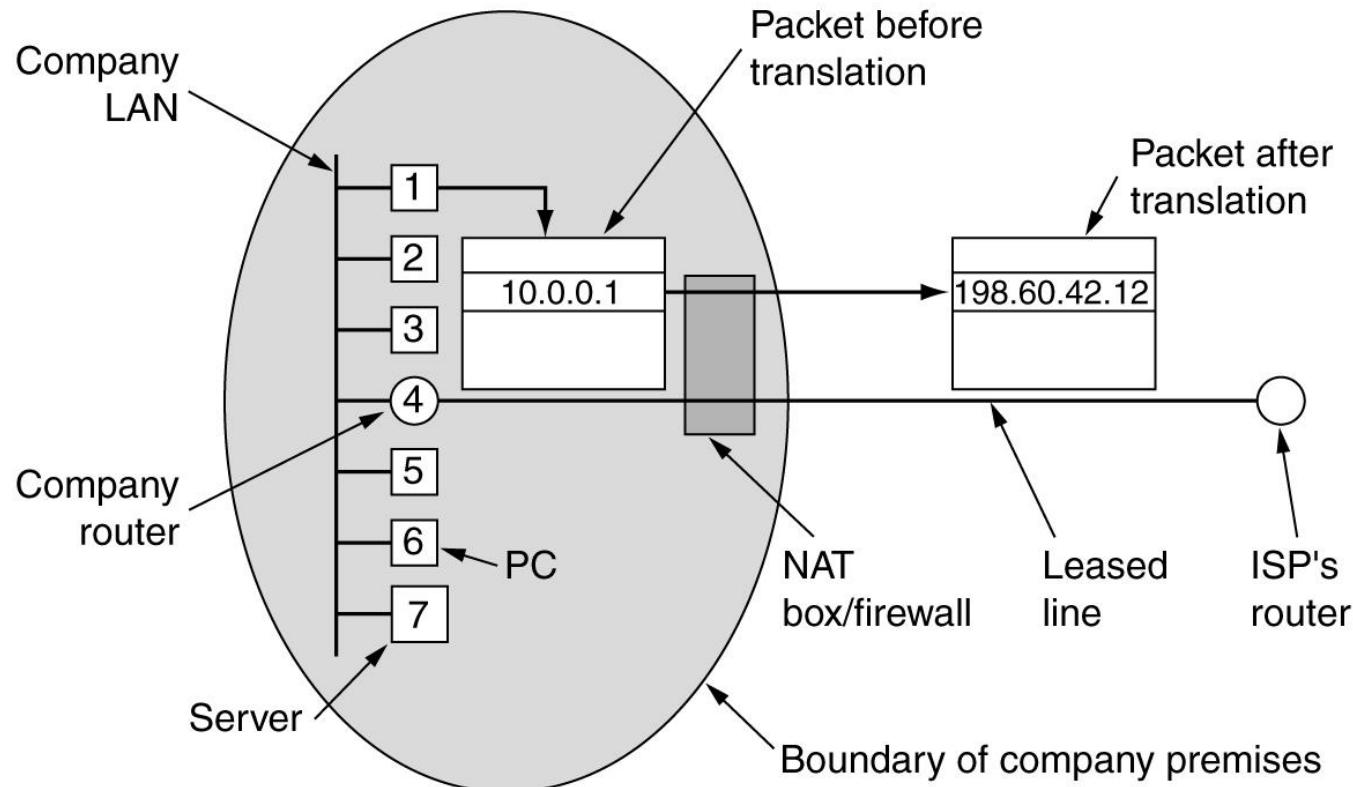
- ❑ Assign one company/customer one IP address for Internet traffic
- ❑ Within the company, every computer gets a unique IP address, used for routing **within the company only**
- ❑ When packet exits company, address is translated to single IP address again
- ❑ Implies some mapping function needed between internal network address and Internet address

NAT IP addresses

- $10.0.0.0 = 10.255.255.255/8$
 - 17,777,216 hosts
- 172.16.0.0 – 172.31.255.255/12
 - 1,048,576 hosts
- 192.168.0.0 -192.168.255.255/16
 - 65,536 hosts
- No packet containing these addresses may appear on the Internet
- Defined as **Private addresses** – internal use only

NAT – Network Address Translation

Placement and operation of a NAT box.



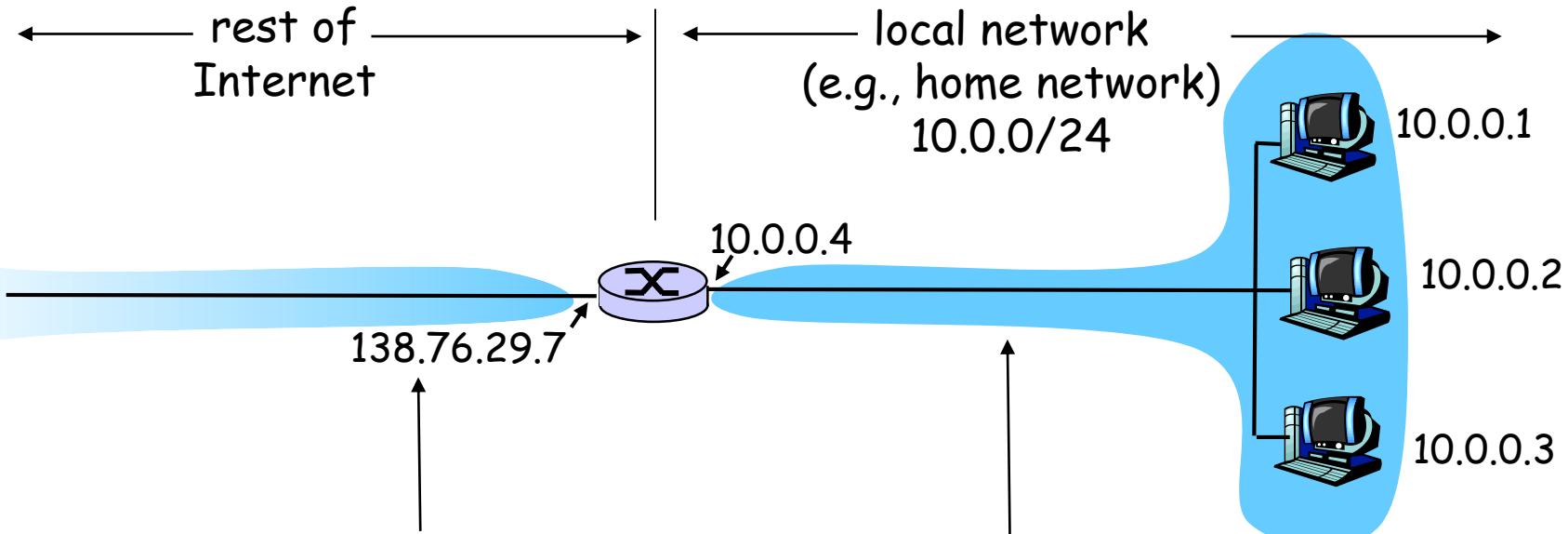
How NAT works – outgoing packets

- NAT box contains its own translation table
- Each entry contains:
 - Internal IP address of host
 - Source port of host
 - Table index number
- When packet leaves network, the NAT box:
 - Extracts source IP address of sending host
 - Extracts source port from TCP header
 - Updates table for that IP address
 - Enters new source port
 - Replaces TCP's source port value with appropriate table index number

How NAT works – incoming packets

- When packet arrives at network, the NAT box:
 - Extracts TCP's destination port of sending host
 - Uses this value as index into translation table
 - Retrieves proper source port from table
 - Updates TCP header
 - Retrieves internal IP address of receiving machine
 - Forwards packet onto correct machine

NAT: Network Address Translation



All datagrams **leaving** local network have **same** single source NAT IP address: 138.76.29.7, different source port numbers

Datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

NAT: Network Address Translation

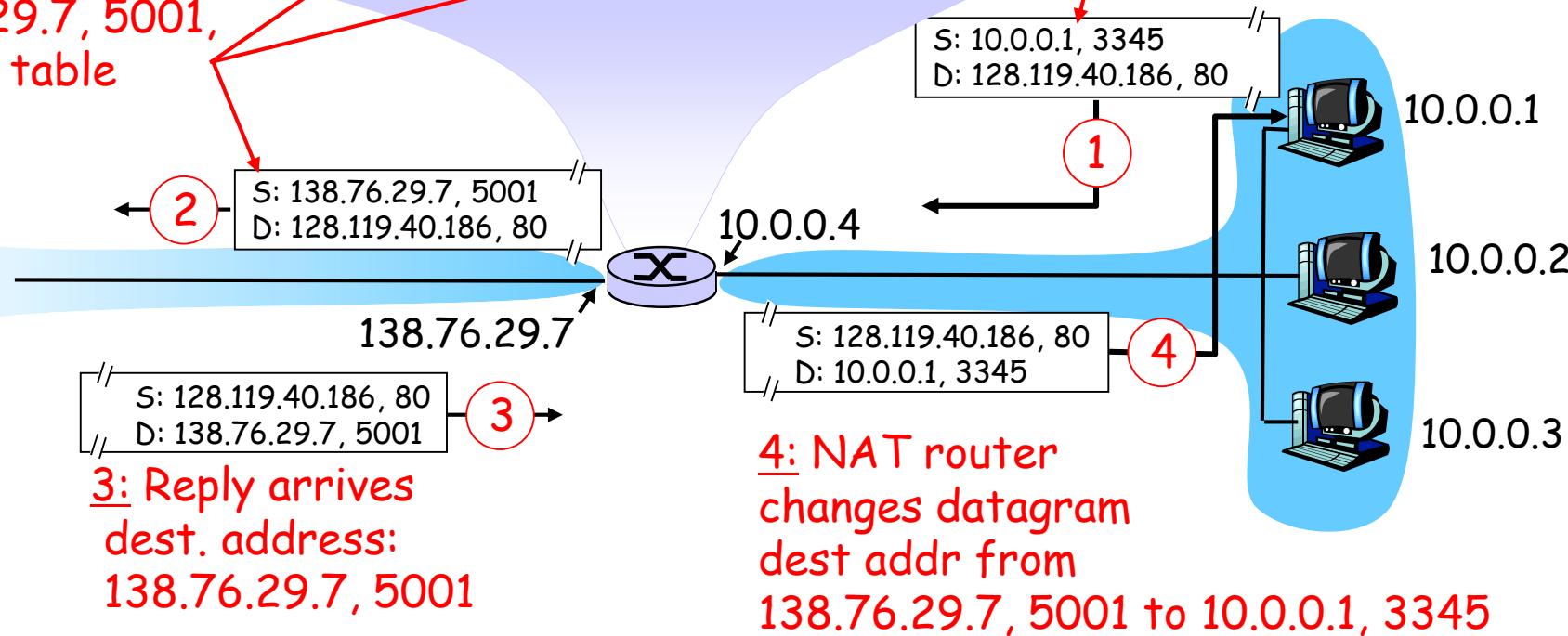
- **Motivation:** local network uses just one IP address as far as outside world is concerned:
 - range of addresses not needed from ISP: just one IP address for all devices
 - can change addresses of devices in local network without notifying outside world
 - can change ISP without changing addresses of devices in local network
 - devices inside local net not explicitly addressable or visible by outside world (a security plus).

NAT: Network Address Translation

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

NAT translation table	
WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
.....

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

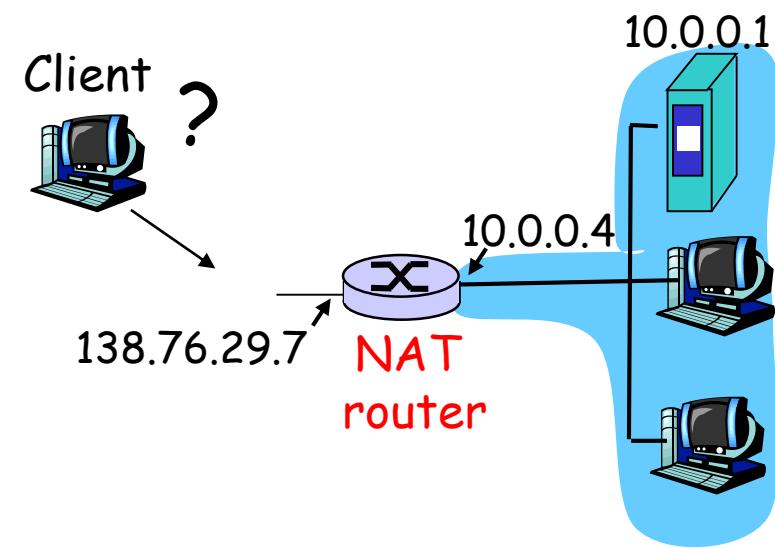


NAT: Network Address Translation

- 16-bit port-number field:
 - 60,000 simultaneous connections with a single LAN-side address!
- NAT is controversial:
 - routers should only process up to layer 3
 - violates end-to-end argument
 - NAT possibility must be taken into account by app designers, eg, P2P applications
 - address shortage should instead be solved by IPv6

NAT traversal problem

- Client wants to connect to server with address 10.0.0.1
 - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
 - only one externally visible NATted address: 138.76.29.7



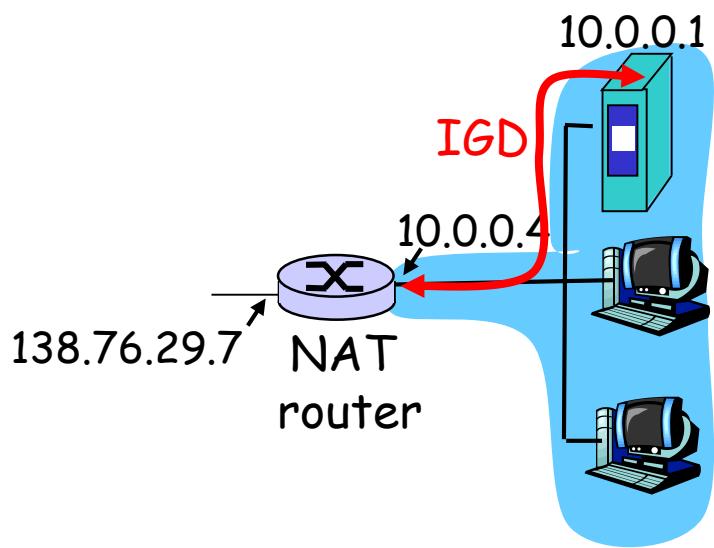
- Solution 1: statically configure NAT to forward incoming connection requests at given port to server

- e.g., (123.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000

NAT traversal problem

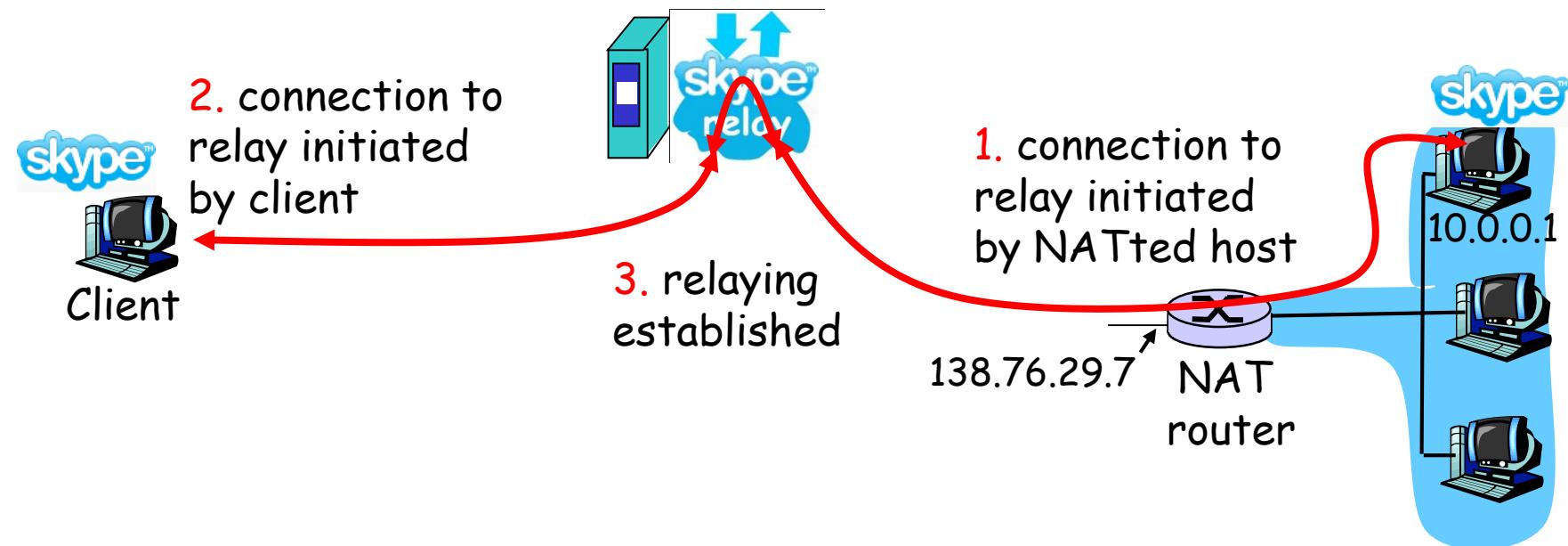
- Solution 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATted host to:
 - ❖ learn public IP address (138.76.29.7)
 - ❖ add/remove port mappings (with lease times)

i.e., automate static NAT port map configuration



NAT traversal problem

- Solution 3: relaying (used in Skype)
 - NATed client establishes connection to relay
 - External client connects to relay
 - relay bridges packets between two connections



Chapter 4: Network Layer

- 4.1 Introduction
- 4.2 Virtual Circuit vs Datagram networks
- 4.3 IPv4: Internet Protocol
 - Datagram format
 - IPv4 Fragmentation
 - IPv4 Addressing
 - Classful Addressing
 - Subnetting
 - Extended Addressing Schemes
 - CIDR
 - NAT
 - **DHCP**
- 4.4 IPv6
 - Enhancements
 - IPv6 Header
 - Flow
 - IPv6 Extension Headers
 - Hop by Hop Options
 - Routing Options
 - Fragmentation
 - IPv6 Addressing
 - Subnetting
 - Mobility
 - Security

4.3.4 IP addresses: how to get one?

Q: How does *host* get IP address?

- Hard-coded by system admin in a file
 - Windows: control-panel->network->configuration->tcp/ip->properties
 - UNIX: /etc/rc.config
- **DHCP: Dynamic Host Configuration Protocol:** dynamically get address from as server
 - “plug-and-play”

DHCP: Dynamic Host Configuration Protocol

Goal: allow host to *dynamically* obtain its IP address from network server when it joins network

Can renew its lease on address in use

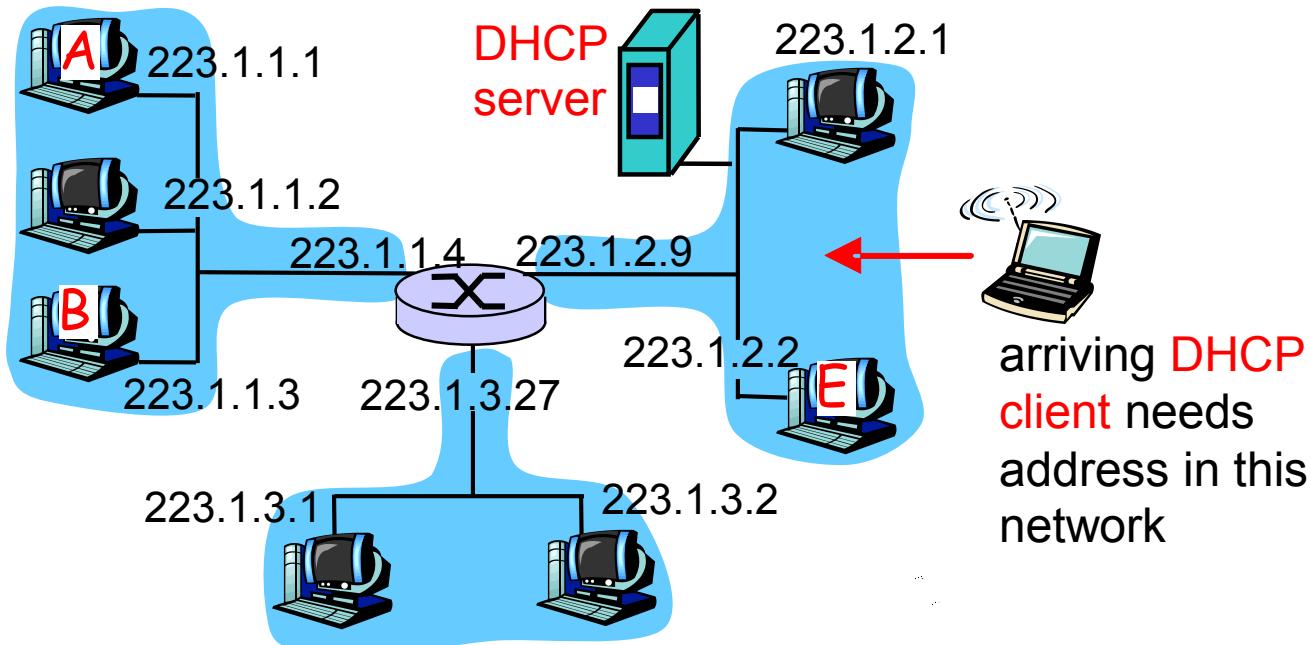
Allows reuse of addresses (only hold address while connected and “on”)

Support for mobile users who want to join network (more shortly)

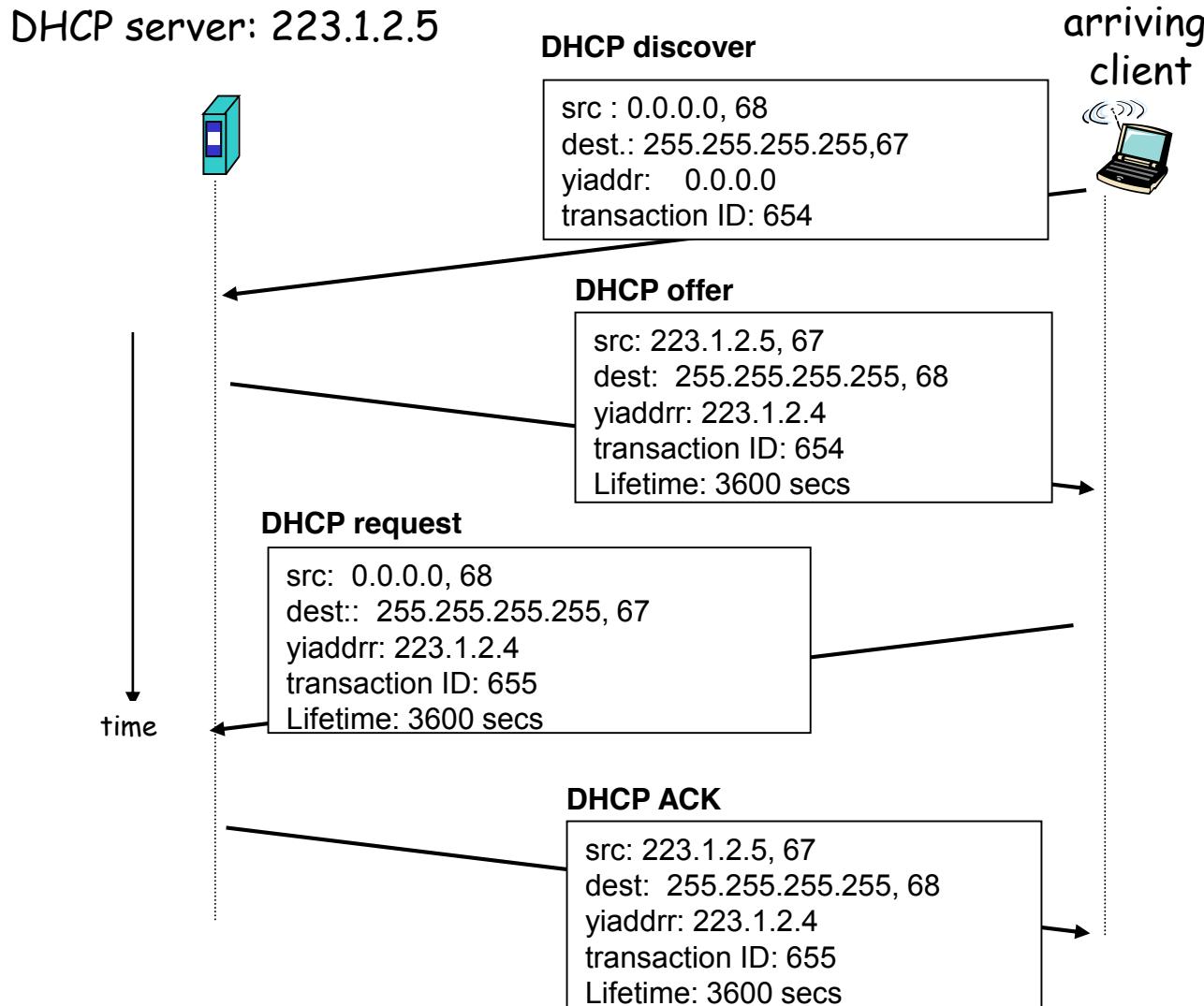
DHCP overview:

- host broadcasts “DHCP discover” msg [optional]
- DHCP server responds with “DHCP offer” msg [optional]
- host requests IP address: “DHCP request” msg
- DHCP server sends address: “DHCP ack” msg

DHCP client-server scenario



DHCP client-server scenario

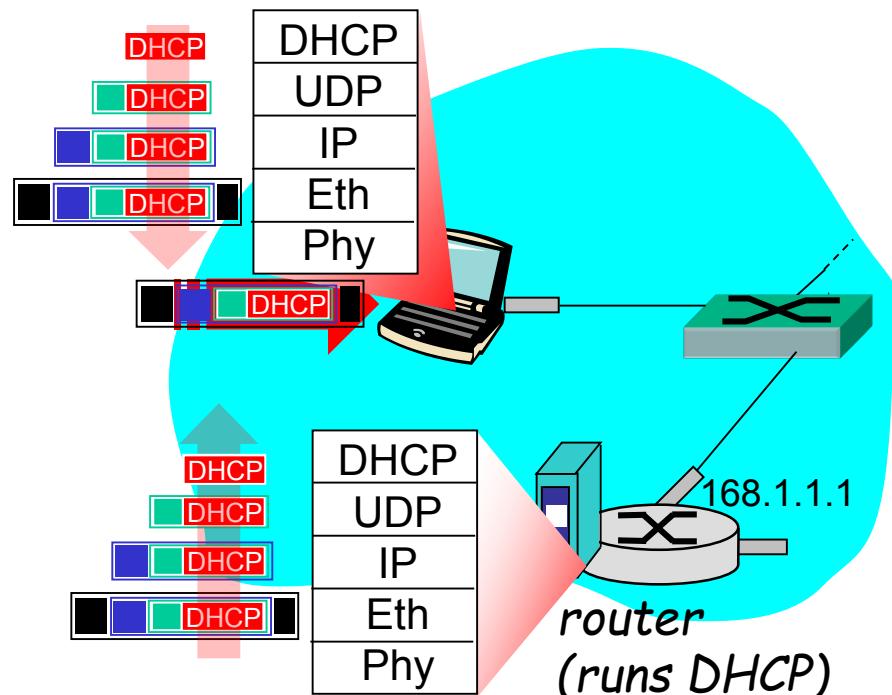


DHCP: more than IP address

DHCP can return more than just allocated IP address on subnet:

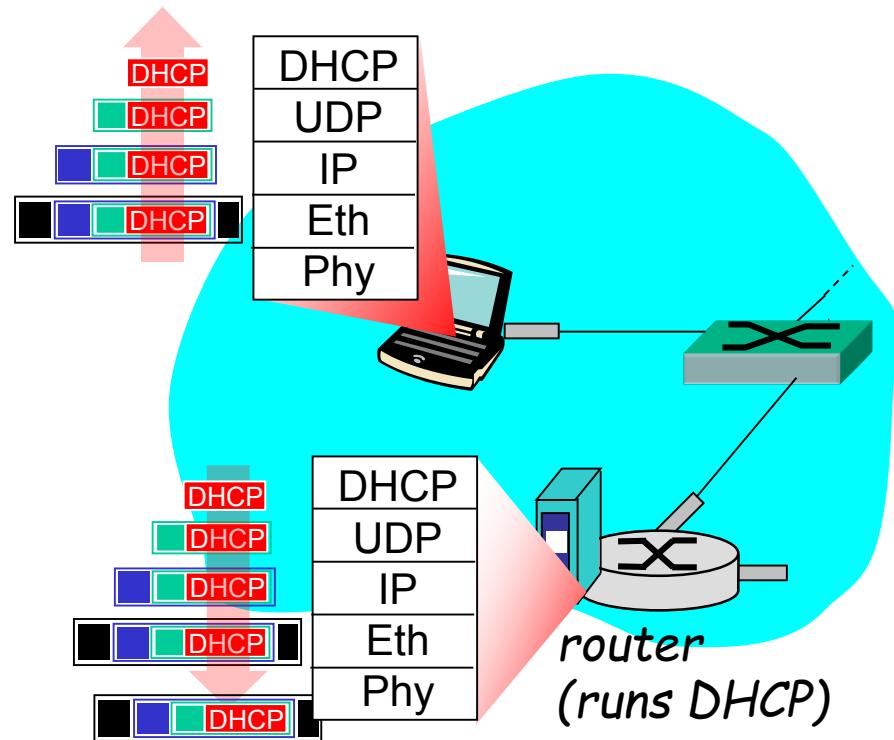
- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)

DHCP: example



- Connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP
- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demux'ed to IP demux'ed, UDP demux'ed to DHCP

DHCP: example



- DCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- Encapsulation of DHCP server, frame forwarded to client, demux' ing up to DHCP at client
- Client now knows its IP address, name and IP address of DNS server, IP address of its first-hop router

How does *network* get subnet part of IP addr?

Q: A: Gets allocated portion of its provider ISP's address space

ISP's block	<u>11001000</u> <u>00010111</u> <u>00010000</u> <u>00000000</u>	200.23.16.0/20
Organization 0	<u>11001000</u> <u>00010111</u> <u>00010000</u> <u>00000000</u>	200.23.16.0/23
Organization 1	<u>11001000</u> <u>00010111</u> <u>00010010</u> <u>00000000</u>	200.23.18.0/23
Organization 2	<u>11001000</u> <u>00010111</u> <u>00010100</u> <u>00000000</u>	200.23.20.0/23
...
Organization 7	<u>11001000</u> <u>00010111</u> <u>00011110</u> <u>00000000</u>	200.23.30.0/23

Q: How does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers

- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

Chapter 4: Network Layer

- 4.1 Introduction
- 4.2 Virtual Circuit vs Datagram networks
- 4.3 IPv4: Internet Protocol
 - Datagram format
 - IPv4 Fragmentation
 - IPv4 Addressing
 - Classful Addressing
 - Subnetting
 - Extended Addressing Schemes
 - CIDR
 - NAT
 - DHCP
- 4.4 IPv6
 - Enhancements
 - IPv6 Header
 - Flow
 - IPv6 Extension Headers
 - Hop by Hop Options
 - Routing Options
 - Fragmentation
 - IPv6 Addressing
 - Subnetting
 - Mobility
 - Security

4.4 IPv6

- **Initial motivation:** 32-bit address space soon to be completely allocated.
- Additional motivation:
 - Header format helps speed processing/forwarding
 - Header changes to facilitate QoS
 - Removes the need for NAT
- **IPv6 datagram format:**
 - Fixed-length 40 byte header
 - No fragmentation allowed

IP Version Numbering

- ❑ IP v 1-3 defined and replaced (never used)
- ❑ IP v4 - current version
- ❑ IP v5 - streams protocol
 - Connection oriented internet layer protocol
 - Developed in late 1970s (ST protocol) and later in 1990s (ST2)
 - Designed to transmit voice, video and offer QoS
 - Never officially adopted
- ❑ IP v6 - replacement for IP v4
 - During development it was called IPng
 - Next Generation

Chapter 4: Network Layer

- 4.1 Introduction
- 4.2 Virtual Circuit vs Datagram networks
- 4.3 IPv4: Internet Protocol
 - Datagram format
 - IPv4 Fragmentation
 - IPv4 Addressing
 - Classful Addressing
 - Subnetting
 - Extended Addressing Schemes
 - CIDR
 - NAT
 - DHCP
- 4.4 IPv6
 - Enhancements
 - IPv6 Header
 - Flow
 - IPv6 Extension Headers
 - Hop by Hop Options
 - Routing Options
 - Fragmentation
 - IPv6 Addressing
 - Subnetting
 - Mobility
 - Security

4.4.1 IPv6 Enhancements

- Expanded address space
 - 128 bit
- Improved option mechanism
 - Separate optional headers between IPv6 header and transport layer header
 - Most are not examined by intermediate routes
 - Improved speed and simplified router processing
 - Easier to extend options
- Address autoconfiguration
 - Dynamic assignment of addresses

IPv6 Enhancements (cont)

- Increased addressing flexibility
 - Anycast - delivered to one of a set of nodes
 - Improved scalability of multicast addresses
- Support for resource allocation
 - Replaces type of service
 - Labeling of packets to particular traffic flow
 - Allows special handling
 - e.g. real time video
- Enhanced Features
 - Security
 - Mobile

...and no checksums!

- Link-layer technologies already perform checksum and error control
- IP checksum is therefore redundant
 - Link layer technologies already reliable enough
- BUT
 - With no IP header checksum, upper-layer optional checksums are now mandatory
 - E.g. UDP

Chapter 4: Network Layer

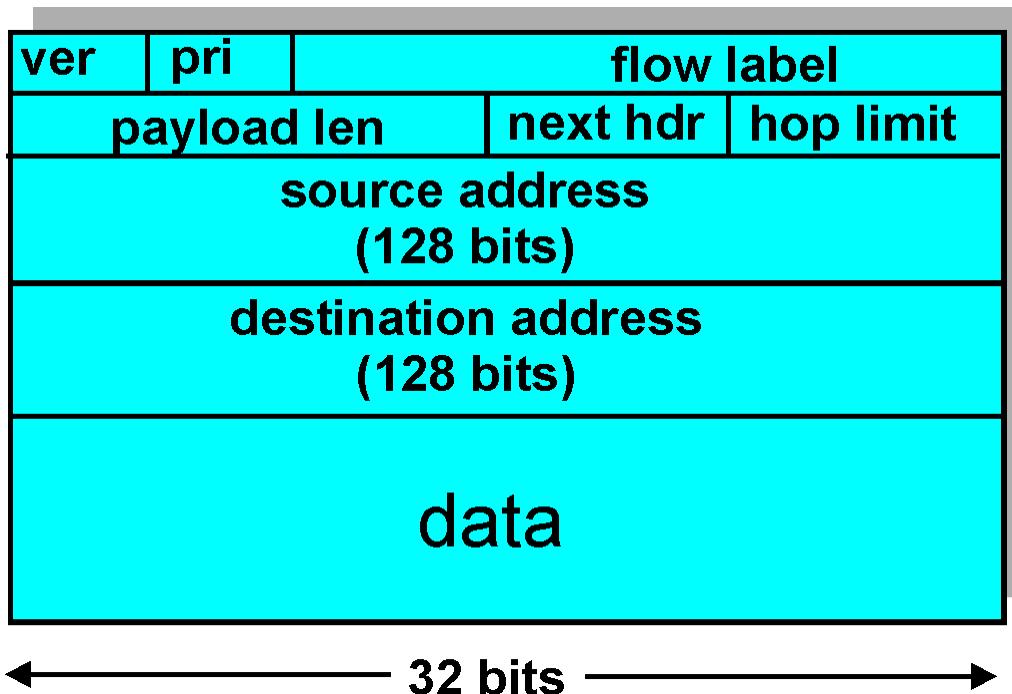
- 4.1 Introduction
- 4.2 Virtual Circuit vs Datagram networks
- 4.3 IPv4: Internet Protocol
 - Datagram format
 - IPv4 Fragmentation
 - IPv4 Addressing
 - Classful Addressing
 - Subnetting
 - Extended Addressing Schemes
 - CIDR
 - NAT
 - DHCP
- 4.4 IPv6
 - Enhancements
 - **IPv6 Header**
 - Flow
 - IPv6 Extension Headers
 - Hop by Hop Options
 - Routing Options
 - Fragmentation
 - IPv6 Addressing
 - Subnetting
 - Mobility
 - Security

4.4.2 IPv6 Header

Priority: identify priority among datagrams in flow

Flow Label: identify datagrams in same “flow.”
(concept of “flow” not well defined).

Next header: identify upper layer protocol for data



Much simpler Header

- ❑ Provides better routing efficiency
- ❑ No broadcasts and thus no potential threat of broadcast storms
- ❑ No requirement for processing checksums
- ❑ Simpler and more efficient extension header mechanisms
- ❑ Flow labels for per-flow processing with no need to open the transport inner packet to identify the various traffic flows.

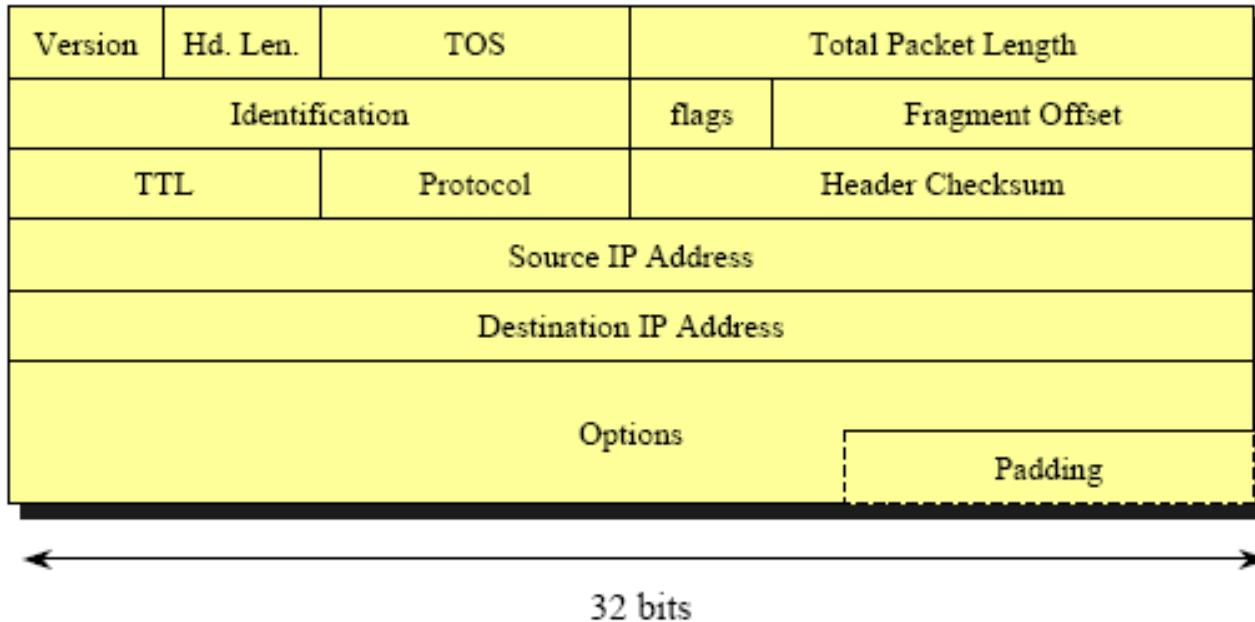
Header Differences between IPv4 and IPv6

□ IPv4:

- 12 basic header fields
- Options field
- Data portion (usually the transport layer segment).
- Total size: fixed, at 20 octets (although the variable-length options field can increase total size)

□ IPv6:

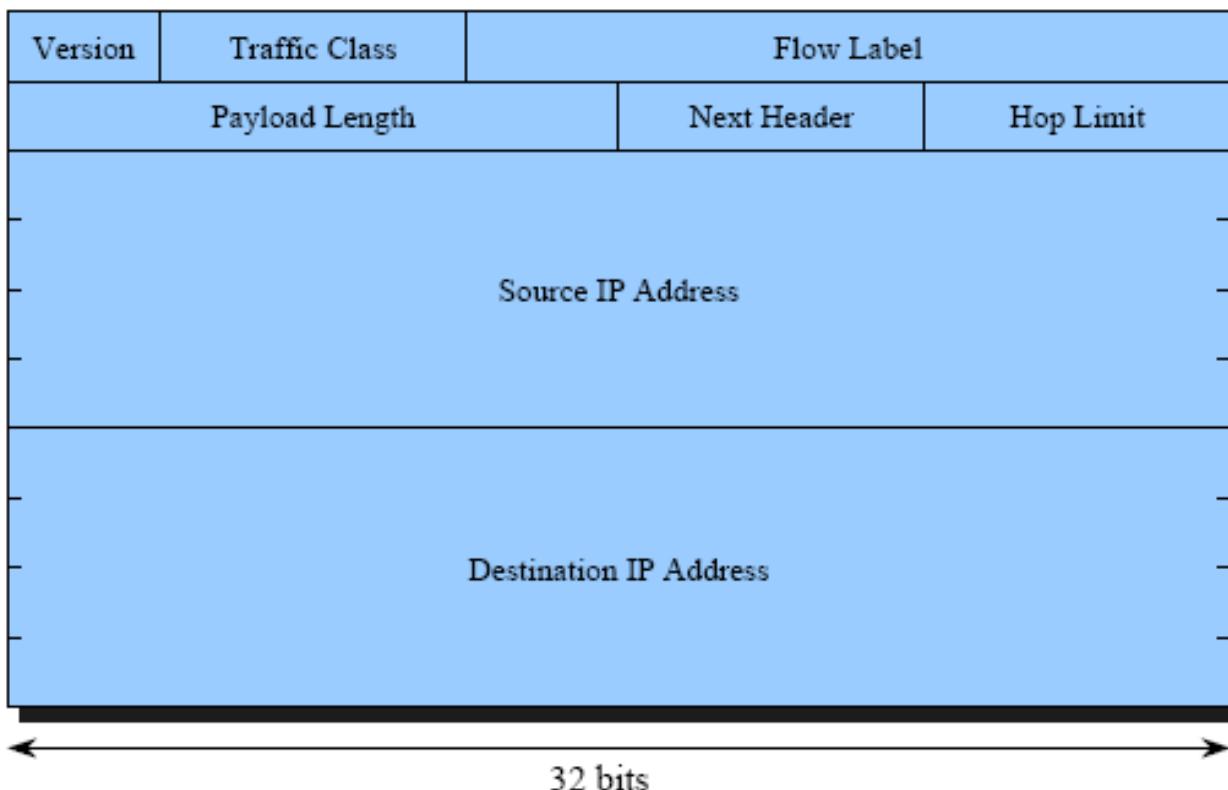
- Contains five of the 12 IPv4 basic header fields.
- Does not require the other 7 fields.
- Total size: Variable



20 bytes

IPv4 Header

n bytes



40 bytes

IPv6 Header

IP v6 Header Fields

- Version
 - 6
- Traffic Class
 - Classes or priorities of packet
 - Still under development
 - See RFC 2460
- Flow Label
 - Used by hosts requesting special handling
- Payload length
 - Includes all extension headers plus user data

IP v6 Header Fields (cont)

- Next Header
 - Identifies type of header
 - Extension or next layer up
- Source Address
- Destination address

Chapter 4: Network Layer

- 4.1 Introduction
- 4.2 Virtual Circuit vs Datagram networks
- 4.3 IPv4: Internet Protocol
 - Datagram format
 - IPv4 Fragmentation
 - IPv4 Addressing
 - Classful Addressing
 - Subnetting
 - Extended Addressing Schemes
 - CIDR
 - NAT
 - DHCP
- 4.4 IPv6
 - Enhancements
 - IPv6 Header
 - **Flow**
 - IPv6 Extension Headers
 - Hop by Hop Options
 - Routing Options
 - Fragmentation
 - IPv6 Addressing
 - Subnetting
 - Mobility
 - Security

4.4.3 Flow

□ Flow

- Sequence of packets from particular source to particular (unicast or multicast) destination
- Source desires special handling by routers
- Uniquely identified by source address, destination address, and 20-bit flow label

□ Router's view

- Sequence of packets sharing attributes affecting how packets handled
 - Path, resource allocation, discard needs, accounting, security
- Handling must be declared
 - Negotiate handling ahead of time using control protocol
 - At transmission time using extension headers
 - E.g. Hop-by-Hop Options header

Flow Label Rules

- Flow Label set to zero if not supported by host or router when originating
 - Pass unchanged when forwarding
 - Ignore when receiving
- Packets from given source with same nonzero Flow Label must have same Destination Address, Source Address, Hop-by-Hop Options header contents (if present), and Routing header contents (if present)
 - Router can make decisions by looking up flow label in table
- Source assigns flow label
 - New flow labels be chosen (pseudo-) randomly and uniformly
 - Range 1 to $2^{20} - 1$
 - Not reuse label within lifetime of existing flow
 - Zero flow label indicates no flow label

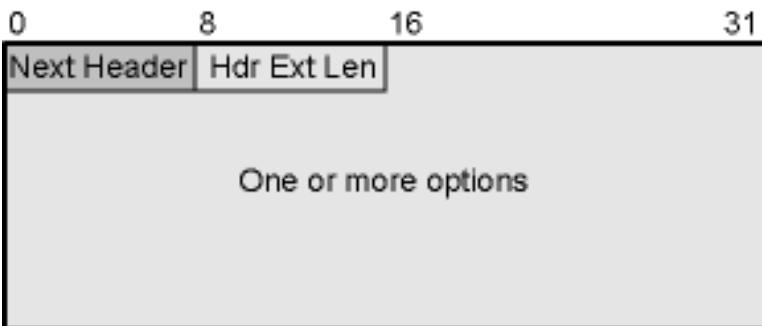
Selection of Flow Label

- Router maintains information on characteristics of active flows
- Table lookup must be efficient
- Could have 2^{20} (about one million) entries
 - Memory burden
- One entry per active flow
 - Router searches table for each packet
 - Processing burden
- Hash table
 - Hashing function using low-order few bits (say 8 or 10) of label or calculation on label
 - Efficiency depends on labels uniformly distributed over possible range
 - Hence pseudo-random, uniform selection requirement

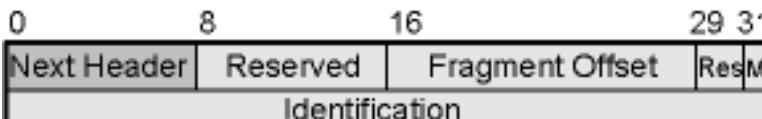
Chapter 4: Network Layer

- 4.1 Introduction
- 4.2 Virtual Circuit vs Datagram networks
- 4.3 IPv4: Internet Protocol
 - Datagram format
 - IPv4 Fragmentation
 - IPv4 Addressing
 - Classful Addressing
 - Subnetting
 - Extended Addressing Schemes
 - CIDR
 - NAT
 - DHCP
- 4.4 IPv6
 - Enhancements
 - IPv6 Header
 - Flow
 - **IPv6 Extension Headers**
 - Hop by Hop Options
 - Routing Options
 - Fragmentation
 - IPv6 Addressing
 - Subnetting
 - Mobility
 - Security

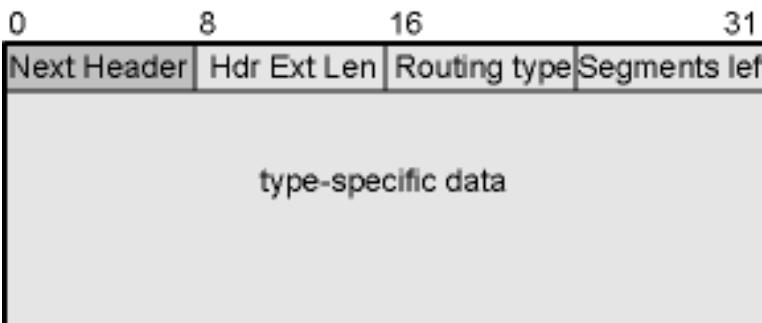
4.4.4 IPv6 Extension Headers



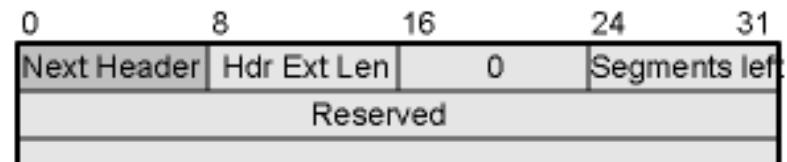
(a) Hop-by-hop options header;
destination options header



(b) Fragment header



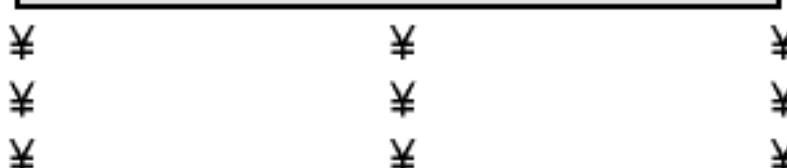
(c) Generic routing header



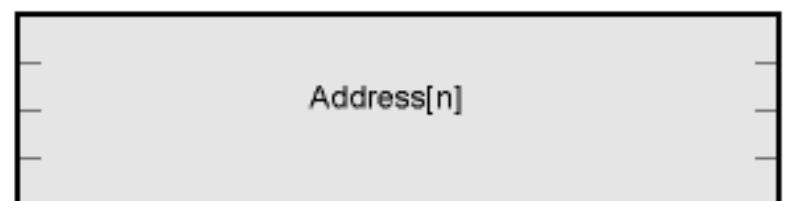
Address[1]



Address[2]

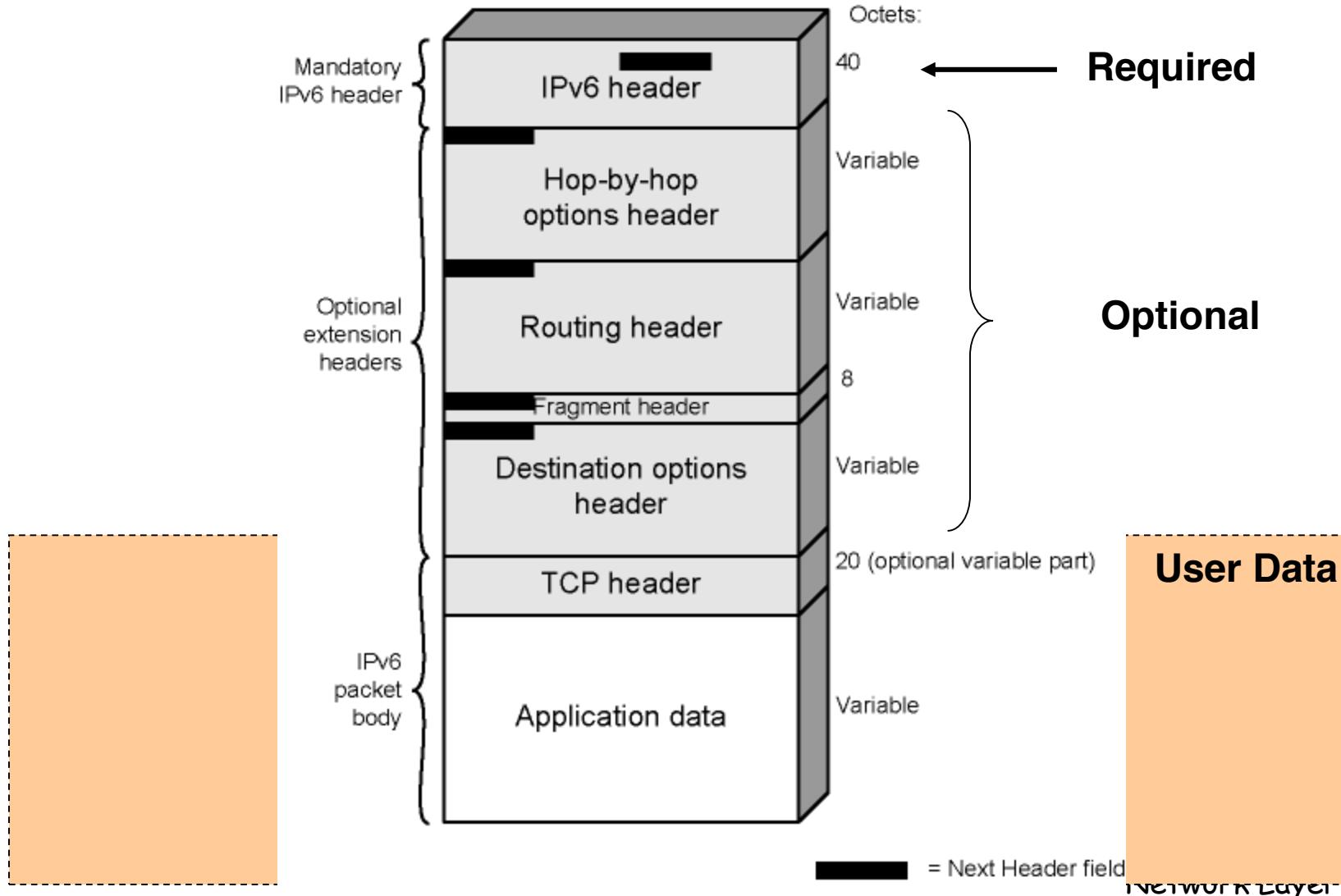


Address[n]



(d) Type 0 routing header

IPv6 Packet with Extension Headers



= Next Header field

Extension Headers In Detail

- Hop-by-Hop Options
 - Require processing at each router
- Routing
 - Similar to v4 source routing
- Fragment
- Authentication
- Encapsulating security payload
- Destination options
 - For destination node

4.4.4.1 Hop-by-Hop Options

- Next header
- Header extension length
- Options
 - Pad1
 - Insert one byte of padding into Options area of header
 - PadN
 - Insert N (≥ 2) bytes of padding into Options area of header
 - Ensure header is multiple of 8 bytes
 - Jumbo payload
 - Over $2^{16} = 65,535$ octets
 - Router alert
 - Tells router that contents of packet is of interest to router
 - Provides support for RSVP

4.4.4.2 Routing Header

- List of one or more intermediate nodes to be visited
- Next Header
- Header extension length
- Routing type
- Segments left
 - i.e. number of nodes still to be visited

4.4.4.3 Fragmentation in IPv6

- ❑ Routers handle fragmentation in IPv4
- ❑ IPv6 routers do **not** perform fragmentation.
- ❑ IPv6 routers use a discovery process
 - determines the optimum maximum transmission unit (MTU)

IPv6 Discovery Process

- Source IPv6 device sends packet with size specified by upper layers
 - e.g. transport or application layer
- If device receives “*ICMP packet too big*” message
 - retransmits the MTU discover packet with smaller MTU
 - repeats process until packet arrives successfully.
 - Then sets the MTU for the session.
- “*ICMP packet too big*” message contains the proper MTU size for the pathway.
- Each source device needs to track the MTU size for each session.

Benefits of the Discovery Process

- Routing pathways change
 - a new MTU might become more appropriate.
- When “ICMP packet too big” message is received:
 - MTU size is reduced if ICMP message contains a recommended MTU less than the current MTU of the device.
- A device performs an MTU discovery every 5 minutes
 - Checks whether the MTU has increased along the pathway.
- Application and transport layers for IPv6 accept MTU reduction notifications from the IPv6 layer.
 - IPv6 can fragment packets that are too large if application or transport layers won't listen
 - But upper layers SHOULD avoid sending messages that require fragmentation.

Fragmentation Header Fields

- ❑ Next Header
- ❑ Reserved
- ❑ Fragmentation offset
- ❑ Reserved
- ❑ More flag
- ❑ Identification

4.4.4.4 Destination Options

- ❑ Same format as Hop-by-Hop options header

4.4.5 IPv6 Addressing

- ❑ Vastly increased address space
- ❑ Improved global reachability and flexibility
- ❑ Multihoming to several Internet service providers (ISPs)
- ❑ Autoconfiguration that can include link-layer addresses in the address space
- ❑ Simplified mechanisms for address renumbering and modification

Chapter 4: Network Layer

- 4.1 Introduction
- 4.2 Virtual Circuit vs Datagram networks
- 4.3 IPv4: Internet Protocol
 - Datagram format
 - IPv4 Fragmentation
 - IPv4 Addressing
 - Classful Addressing
 - Subnetting
 - Extended Addressing Schemes
 - CIDR
 - NAT
 - DHCP
- 4.4 IPv6
 - Enhancements
 - IPv6 Header
 - Flow
 - IPv6 Extension Headers
 - Hop by Hop Options
 - Routing Options
 - Fragmentation
 - **IPv6 Addressing**
 - Subnetting
 - Mobility
 - Security

4.4.5.1 IPv6 Addresses

- 128 bits long
- Assigned to interface
- Single interface may have multiple unicast addresses
- Example IPv6 address
 - A node's address
 - 2001:0DB8:0:CD30:123:4567:89AB:CDEF
 - its subnet mask
 - 2001:0DB8:0:CD30::/60
 - abbreviated as
 - 2001:0DB8:0:CD30:123:4567:89AB:CDEF/60
 - (<http://tools.ietf.org/html/rfc4291>)

Types of address

- Unicast
 - Single interface
- Anycast
 - Set of interfaces (typically different nodes)
 - Delivered to any one interface
 - the “nearest”
- Multicast
 - Set of interfaces
 - Delivered to all interfaces identified

4.4.5.2 Subnetting with IPv6

- Subnetting in IPv4
 - Designed to improve efficiency in address utilization
- Subnetting in IPv6
 - No such restrictions
 - Used by organization to manage their network
- Subnet Masks
 - The recommended allocation is an 80-bit (/48) prefix
 - This provides 65536 subnets for a site
 - /64 also common (needed for stateless address autoconfiguration)
- Subnetting in IPv6 based on CIDR and Variable-Length Subnet Masking (VLSM)

Chapter 4: Network Layer

- 4.1 Introduction
- 4.2 Virtual Circuit vs Datagram networks
- 4.3 IPv4: Internet Protocol
 - Datagram format
 - IPv4 Fragmentation
 - IPv4 Addressing
 - Classful Addressing
 - Subnetting
 - Extended Addressing Schemes
 - CIDR
 - NAT
 - DHCP
- 4.4 IPv6
 - Enhancements
 - IPv6 Header
 - Flow
 - IPv6 Extension Headers
 - Hop by Hop Options
 - Routing Options
 - Fragmentation
 - IPv6 Addressing
 - Subnetting
 - Mobility
 - Security

4.4.6 Support for Mobility

- ❑ Mobility is built in, so any IPv6 node can use it when necessary
 - Enables people to move around in networks with mobile network devices—with many having wireless connectivity.
- ❑ IPv6 fully compliant with Mobile IP
 - Mobile IP is IETF standard available for both IPv4 and IPv6.
 - Enables mobile devices to move without breaks in established network connections.
- ❑ IPv4 does not automatically provide this kind of mobility
 - must be added with additional configurations.

Chapter 4: Network Layer

- 4.1 Introduction
- 4.2 Virtual Circuit vs Datagram networks
- 4.3 IPv4: Internet Protocol
 - Datagram format
 - IPv4 Fragmentation
 - IPv4 Addressing
 - Classful Addressing
 - Subnetting
 - Extended Addressing Schemes
 - CIDR
 - NAT
 - DHCP
- 4.4 IPv6
 - Enhancements
 - IPv6 Header
 - Flow
 - IPv6 Extension Headers
 - Hop by Hop Options
 - Routing Options
 - Fragmentation
 - IPv6 Addressing
 - Subnetting
 - Mobility
 - Security

4.4.7 Support for Security

- IPsec is the IETF standard for IP network security
 - available for both IPv4 and IPv6.
- IPsec is mandatory in IPv6.
 - IPsec is enabled on every IPv6 node and is available for use.
 - If a router supports IPv6, it supports IPsec
- The availability of IPsec on all nodes makes the IPv6 Internet more secure.
 - IPsec also requires keys for each party, which implies global key deployment and distribution.

Summary

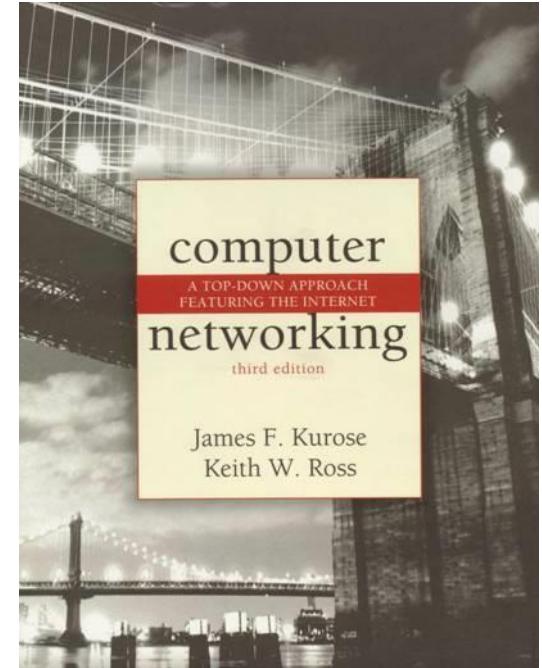
- 4.1 Introduction
- 4.2 Virtual Circuit vs Datagram networks
- 4.3 IPv4: Internet Protocol
 - Datagram format
 - IPv4 Fragmentation
 - IPv4 Addressing
 - Classful Addressing
 - Subnetting
 - Extended Addressing Schemes
 - CIDR
 - NAT
 - DHCP
- 4.4 IPv6
 - Enhancements
 - IPv6 Header
 - Flow
 - IPv6 Extension Headers
 - Hop by Hop Options
 - Routing Options
 - Fragmentation
 - IPv6 Addressing
 - Subnetting
 - Mobility
 - Security

Computer Networks

Chapter 5 Data Link Layer and LANs

Professor R Simon Sherratt

Adapted and updated from the original work
of Dr. Michael Evans



*Computer Networking: A
Top Down Approach
Featuring the Internet,*

Jim Kurose, Keith Ross
Addison-Wesley

Chapter 5: The Data Link Layer

Our goals:

- understand principles behind data link layer services:
 - sharing a broadcast channel: multiple access
 - reliable data transfer, flow control: *done!*
 - addressing
- instantiation and implementation of various link layer technologies
 - Ethernet and LAN technologies

Plus...One More Thing

Tying everything together

- How the Web and Internet works from start to finish
- ATM: A very different way of doing things
- How the Internet and ATM co-exist in today's networks

Link Layer

- 5.1 Introduction
 - 5.2 Link Layer Services
 - 5.3 Multiple access protocols
- LAN Technology
 - 5.4 LAN Addresses and ARP
 - 5.5 Ethernet
 - 5.6 Interconnecting LANs
 - 5.7 Examples of LAN Configurations
- The Whole Enchilada
 - 5.7 How the Web and Internet work
 - 5.8 ATM
 - 5.9 How IP and ATM co-exist

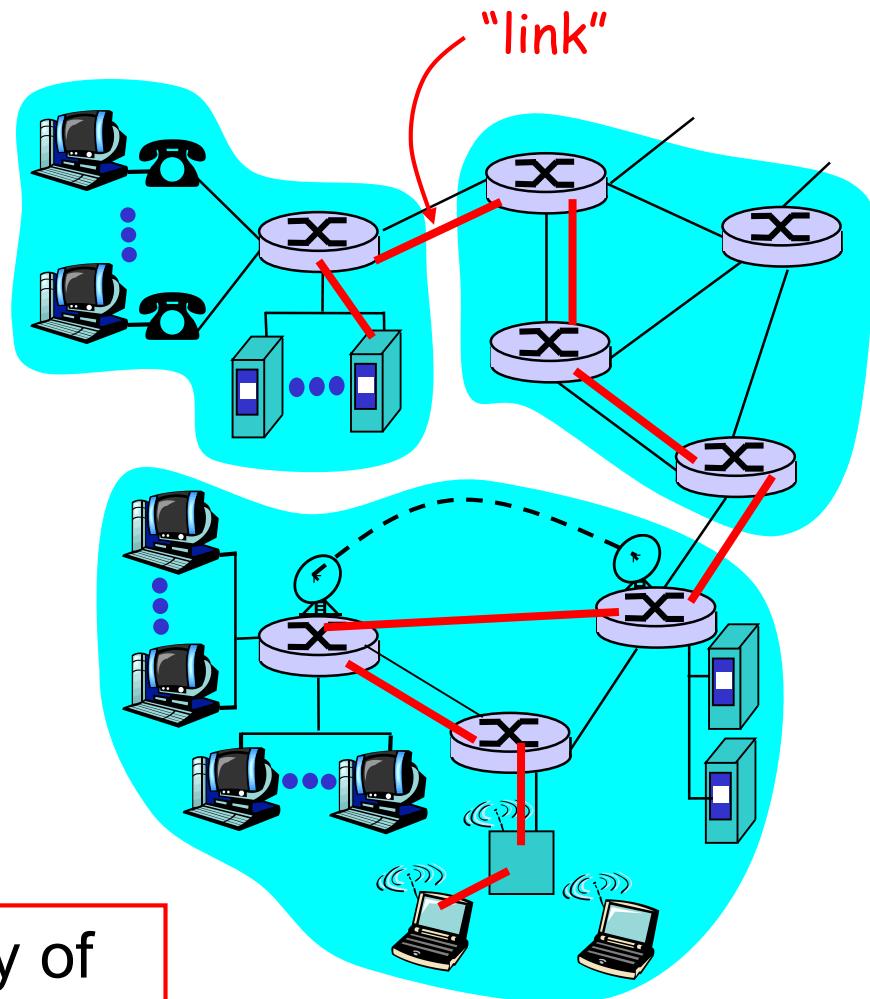
Link Layer

- **5.1 Introduction**
 - 5.2 Link Layer Services
 - 5.3 Multiple access protocols
- **LAN Technology**
 - 5.4 LAN Addresses and ARP
 - 5.5 Ethernet
 - 5.6 Interconnecting LANs
 - 5.7 Examples of LAN Configurations
- **The Whole Enchilada**
 - 5.7 How the Web and Internet work
 - 5.8 ATM
 - 5.9 How IP and ATM co-exist

5.1 Introduction to the Link Layer

Some terminology:

- ❑ Hosts and routers are **nodes**
- ❑ Communication channels that connect adjacent nodes along communication path are **links**
 - wired links
 - wireless links
 - LANs
- ❑ Layer-2 packet is a **frame**, encapsulates datagram



Data-link layer has responsibility of transferring datagram from one node to adjacent node over a link

Link layer: context

- Datagram transferred by different link protocols over different links:
 - e.g. Ethernet on first link, frame relay on intermediate links, 802.11 on last link
- Each link protocol provides different services
 - e.g. may or may not provide reliable data transfer over link

Link Layer

- 5.1 Introduction
 - 5.2 Link Layer Services
 - 5.3 Multiple access protocols
- LAN Technology
 - 5.4 LAN Addresses and ARP
 - 5.5 Ethernet
 - 5.6 Interconnecting LANs
 - 5.7 Examples of LAN Configurations
- The Whole Enchilada
 - 5.7 How the Web and Internet work
 - 5.8 ATM
 - 5.9 How IP and ATM co-exist

5.2 Link Layer Services

❑ Framing, link access:

- Encapsulate datagram into frame, adding header, trailer
- Channel access if shared medium
- “MAC” addresses used in frame headers to identify source, dest
 - different from IP address!

❑ Reliable delivery between adjacent nodes

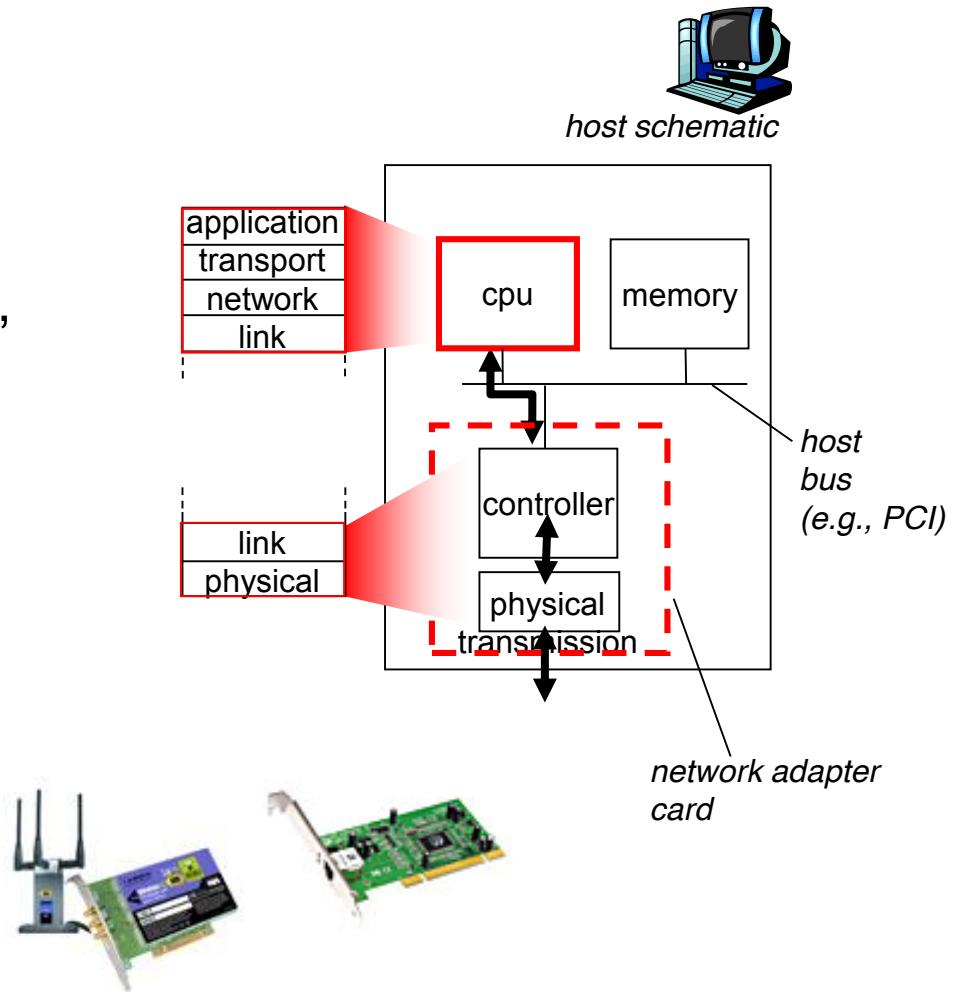
- We learned how to do this already (chapter 3)
- Seldom used on low bit error link (fiber, some twisted pair)
- Wireless links: high error rates

Link Layer Services (more)

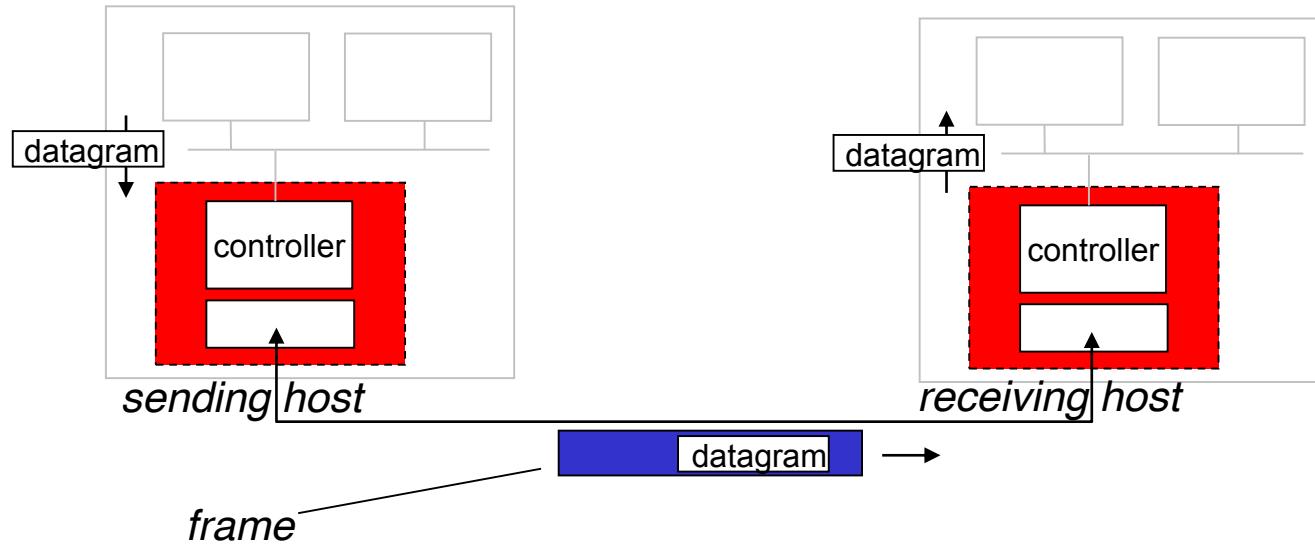
- *Flow Control:*
 - Pacing between adjacent sending and receiving nodes
- *Error Detection:*
 - Errors caused by signal attenuation, noise.
 - Receiver detects presence of errors:
 - signals sender for retransmission or drops frame
- *Error Correction:*
 - Receiver identifies *and corrects* bit error(s) without resorting to retransmission
- *Half-duplex and full-duplex*
 - With half duplex, nodes at both ends of link can transmit, but not at same time

Where is the link layer implemented?

- in each and every host
- link layer implemented in “adaptor” (aka *network interface card* NIC)
 - Ethernet card, PCMCIA card, 802.11 card
 - implements link, physical layer
- attaches into host’s system buses
- combination of hardware, software, firmware



Adaptors Communicating



- **sending side:**
 - encapsulates datagram in frame
 - adds error checking bits, rdt, flow control, etc.
- **receiving side**
 - looks for errors, rdt, flow control, etc
 - extracts datagram, passes to upper layer at receiving side

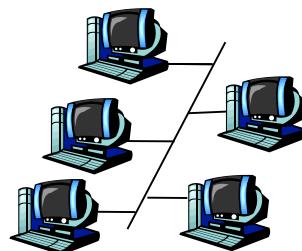
Link Layer

- 5.1 Introduction
 - 5.2 Link Layer Services
 - **5.3 Multiple access protocols**
- LAN Technology
 - 5.4 LAN Addresses and ARP
 - 5.5 Ethernet
 - 5.6 Interconnecting LANs
 - 5.7 Examples of LAN Configurations
- The Whole Enchilada
 - 5.7 How the Web and Internet work
 - 5.8 ATM
 - 5.9 How IP and ATM co-exist

5.3 Multiple Access Links and Protocols

Two types of “links”:

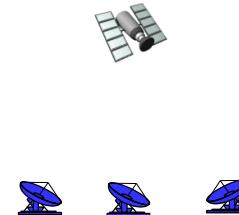
- point-to-point
 - PPP for dial-up access
 - point-to-point link between Ethernet switch and host
- broadcast (shared wire or medium)
 - old-fashioned Ethernet
 - 802.11 wireless LAN



shared wire (e.g.,
cabled Ethernet)



shared RF
(e.g., 802.11 WiFi)



shared RF
(satellite)



humans at a
cocktail party
(shared air, acoustical)

Multiple Access protocols

- ❑ Single shared broadcast channel
- ❑ Two or more simultaneous transmissions by nodes:
interference
 - Collision if node receives two or more signals at the same time

Multiple access protocol

- ❑ **Distributed algorithm that determines how nodes share channel, i.e., determine when node can transmit**
- ❑ Communication about channel sharing must use channel itself!
 - No out-of-band channel for coordination

Ideal Multiple Access Protocol

Broadcast channel of rate R bps

1. when one node wants to transmit, it can send at rate R .
2. when M nodes want to transmit, each can send at average rate R/M
3. fully decentralized:
 - no special node to coordinate transmissions
 - no synchronization of clocks, slots
4. simple

MAC Protocols: a taxonomy

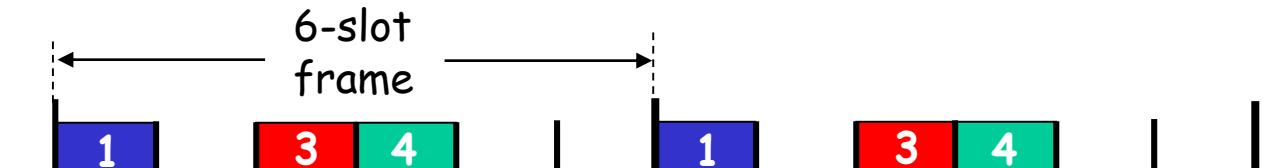
Three broad classes:

- 1. Channel Partitioning
 - Divide channel into smaller “pieces” (time slots, frequency, code)
 - Allocate piece to node for exclusive use
- 2. Random Access
 - Channel not divided, allow collisions
 - “Recover” from collisions
- 3. “Taking turns”
 - Nodes take turns, but nodes with more to send can take longer turns

5.3.1 Channel Partitioning MAC protocols: TDMA & FDMA

TDMA: time division multiple access

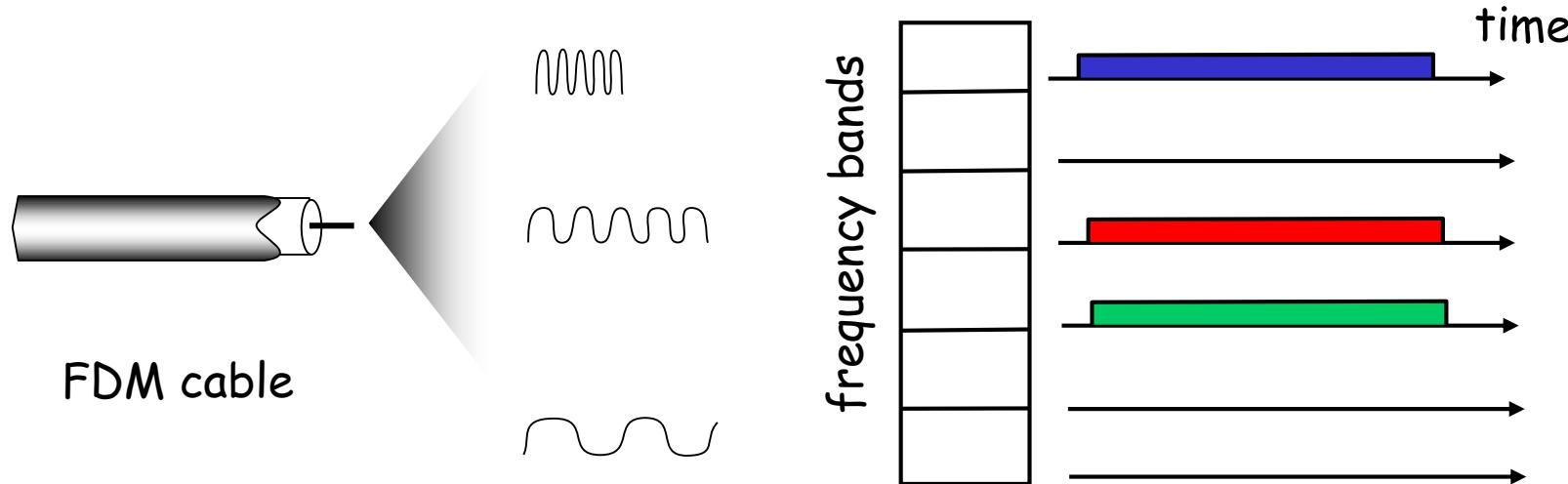
- ❑ access to channel in "rounds"
- ❑ each station gets fixed length slot (length = pkt trans time) in each round
- ❑ unused slots go idle
- ❑ example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle



Channel Partitioning MAC protocols: FDMA

FDMA: frequency division multiple access

- ❑ channel spectrum divided into frequency bands
- ❑ each station assigned fixed frequency band
- ❑ unused transmission time in frequency bands go idle
- ❑ example: 6-station LAN, 1,3,4 have pkt, frequency bands 2,5,6 idle



5.3.2 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 transmitting nodes → “collision”
- Random access MAC protocol specifies:
 - How to detect collisions
 - How to recover from collisions (e.g., via delayed retransmissions)
- Examples of random access MAC protocols:
 - Slotted ALOHA
 - ALOHA
 - CSMA, CSMA/CD, CSMA/CA

CSMA/CD (Collision Detection)

CSMA/CD: carrier sensing, deferral

- Collisions *detected* within short time
- Colliding transmissions aborted, reducing channel 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

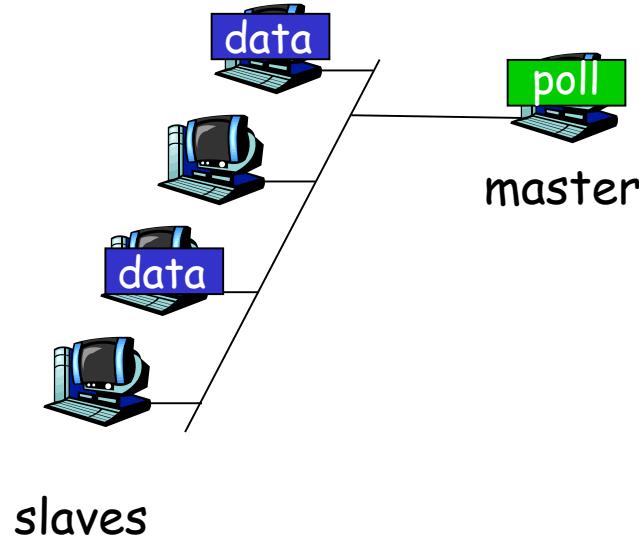
5.3.3 “Taking Turns” MAC protocols

- Channel partitioning MAC protocols:
 - Share channel efficiently and fairly at high load
 - Inefficient at low load: delay in channel access, $1/N$ bandwidth allocated even if only 1 active node!
- Random access MAC protocols
 - Efficient at low load: single node can fully utilize channel
 - High load: collision overhead
- “Taking turns” protocols
 - Look for best of both worlds!

“Taking Turns” MAC protocols

Polling:

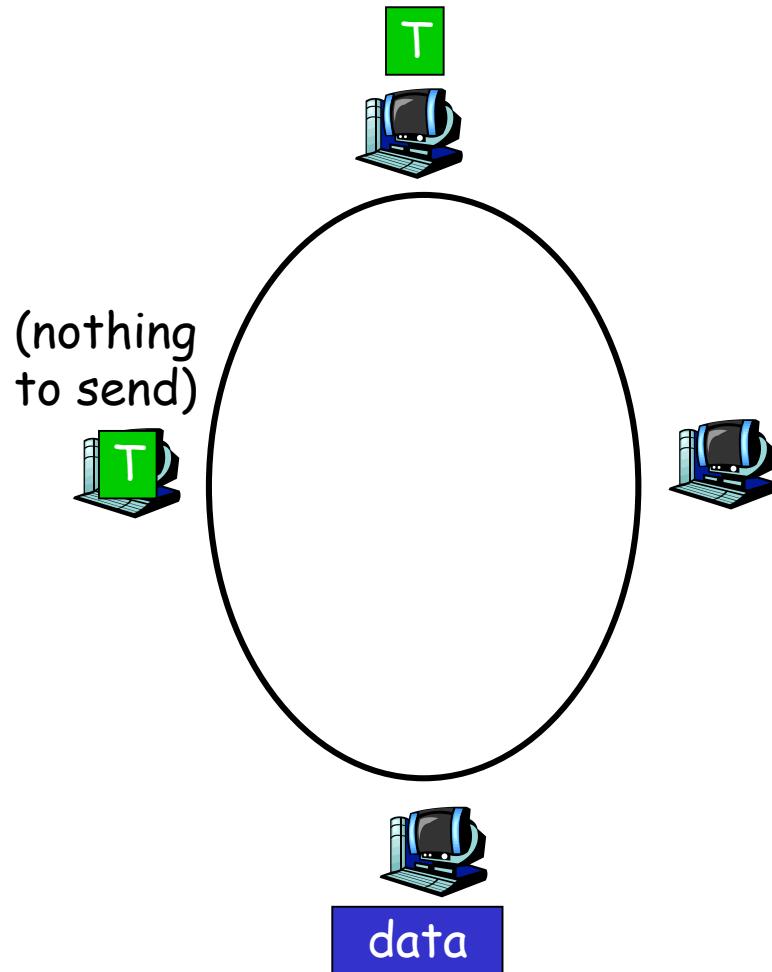
- ❑ master node “invites” slave nodes to transmit in turn
- ❑ typically used with “dumb” slave devices
- ❑ concerns:
 - polling overhead
 - latency
 - single point of failure (master)



“Taking Turns” MAC protocols

Token passing:

- Control **token** passed from one node to next sequentially.
- Only transmit on possession of token
- E.g. Token Ring
- Concerns:
 - token overhead
 - latency
 - single point of failure (token)



Summary of MAC protocols

- What do you do with a shared media?
 - **Channel Partitioning, by time, frequency or code**
 - Time Division, Frequency Division
 - **Random partitioning (dynamic),**
 - ALOHA, S-ALOHA, CSMA, CSMA/CD
 - Carrier sensing: easy in some technologies (wire), hard in others (wireless)
 - CSMA/CD used in Ethernet
 - CSMA/CA used in 802.11
 - **Taking Turns**
 - Polling from a central site, token passing

Next: LAN Technologies

- ❑ Addressing
- ❑ Ethernet
- ❑ Hubs, Switches

Link Layer

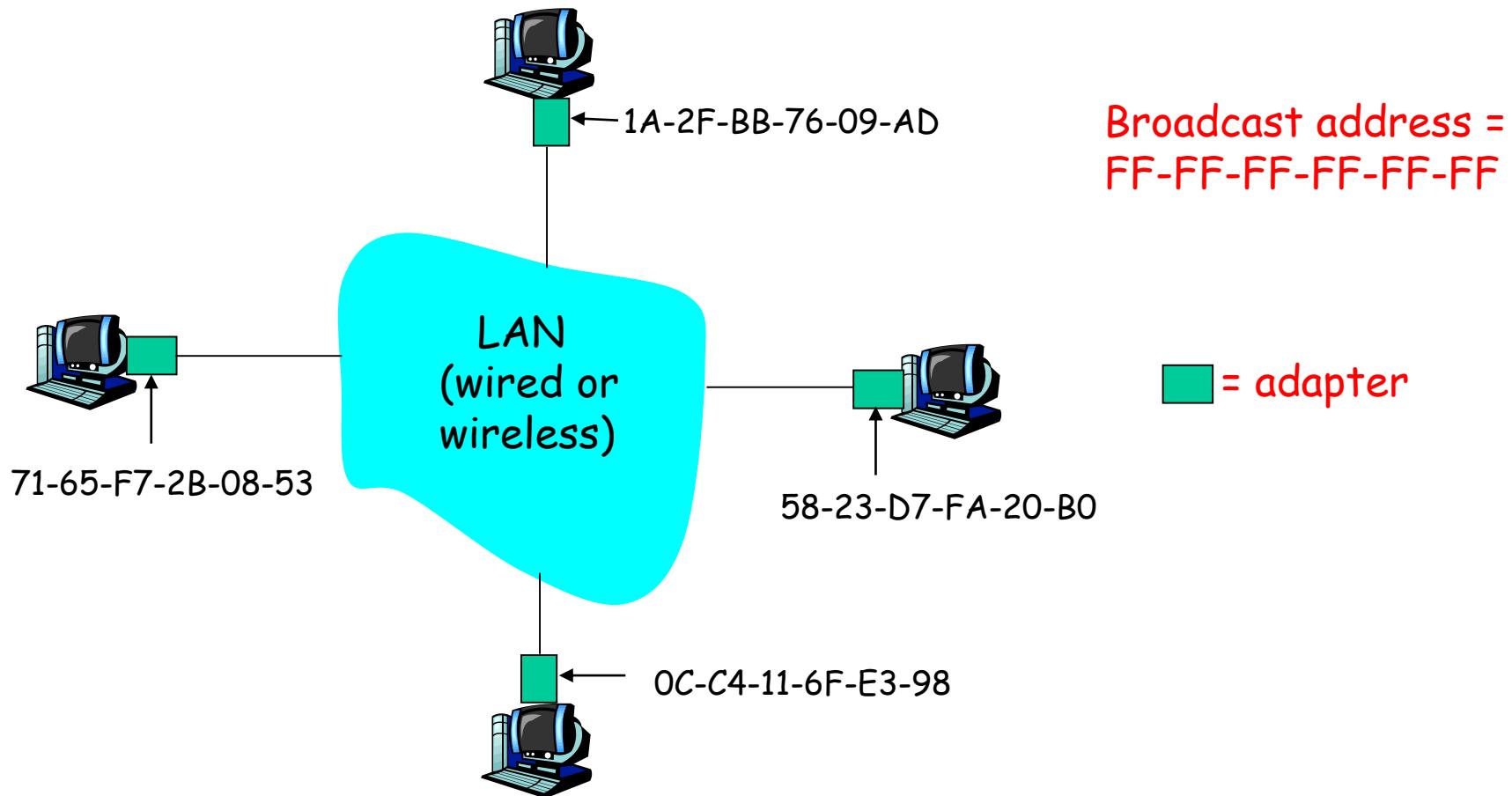
- 5.1 Introduction
 - 5.2 Link Layer Services
 - 5.3 Multiple access protocols
- LAN Technology
 - 5.4 LAN Addresses and ARP
 - 5.5 Ethernet
 - 5.6 Interconnecting LANs
 - 5.7 Examples of LAN Configurations
- The Whole Enchilada
 - 5.7 How the Web and Internet work
 - 5.8 ATM
 - 5.9 How IP and ATM co-exist

5.4 LAN Addresses and ARP

- Network Layer: 32-bit IP address:
 - *network-layer* address
 - used to get datagram to destination IP subnet
- Link Layer: MAC (aka LAN or physical or Ethernet) address:
 - function: *get frame from one interface to another physically-connected interface (same network)*
 - 48 bit MAC address (for most LANs)
 - burned in NIC ROM, also sometimes software settable

LAN Address identifies NIC

Each NIC on LAN has unique LAN address

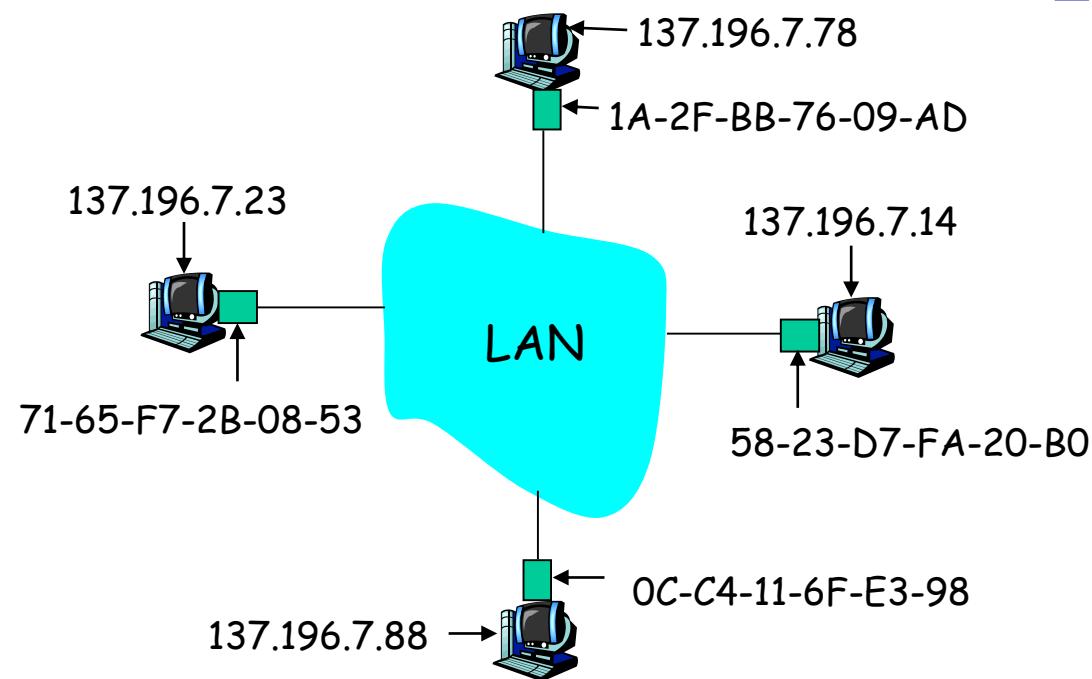


LAN Address (more)

- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- analogy:
 - (a) MAC address: like Social Security Number
 - (b) IP address: like postal address
- MAC address-space is flat → portability
 - can move LAN card from one LAN to another
- IP address-space is hierarchical → NOT portable
 - address depends on IP subnet to which node is attached

5.4.1 ARP: Address Resolution Protocol

Question: how to determine MAC address of B knowing B's IP address?



- Each IP node (host, router) on LAN has **ARP** table
- ARP table: IP/MAC address mappings for some LAN nodes
 - < IP address; MAC address; TTL >
 - TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

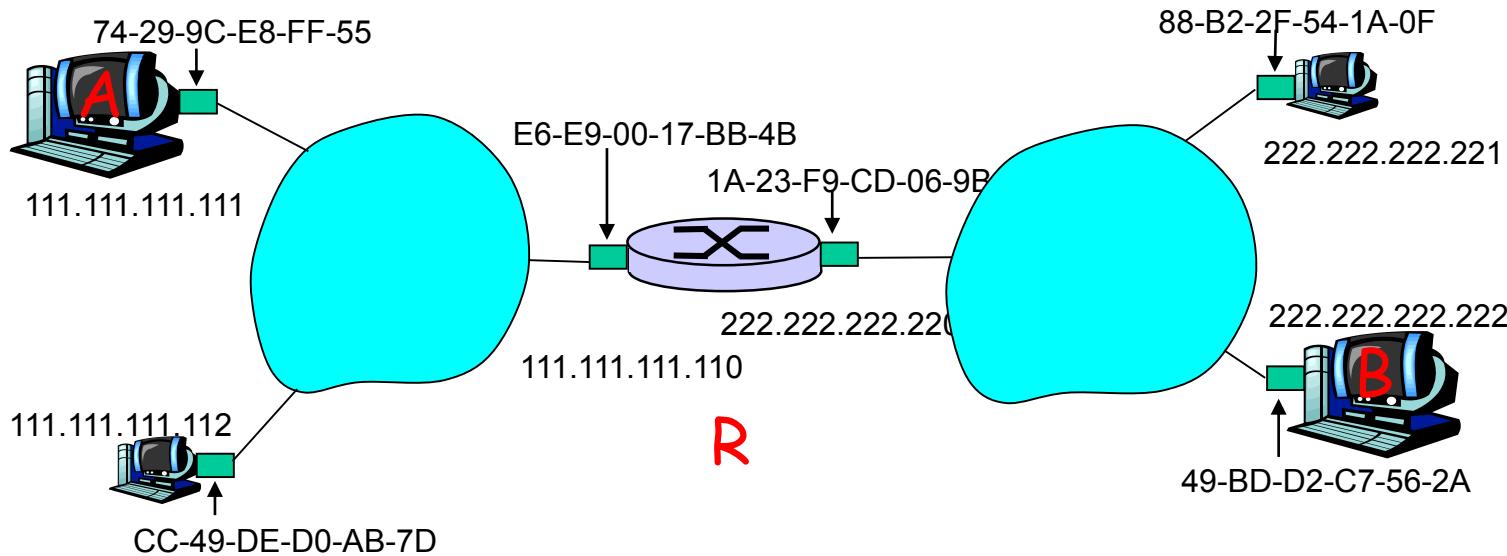
ARP protocol: Same LAN (network)

- A wants to send datagram to B, and B's MAC address not in A's ARP table.
- A **broadcasts** ARP query packet, containing B's IP address
 - dest MAC address = FF-FF-FF-FF-FF-FF
 - all machines on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) MAC address
 - frame sent to A's MAC address (unicast)
- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
 - soft state: information that times out (goes away) unless refreshed
- ARP is “plug-and-play”:
 - nodes create their ARP tables *without intervention from net administrator*

Addressing: Routing to another LAN

walkthrough: **send datagram from A to B via R**

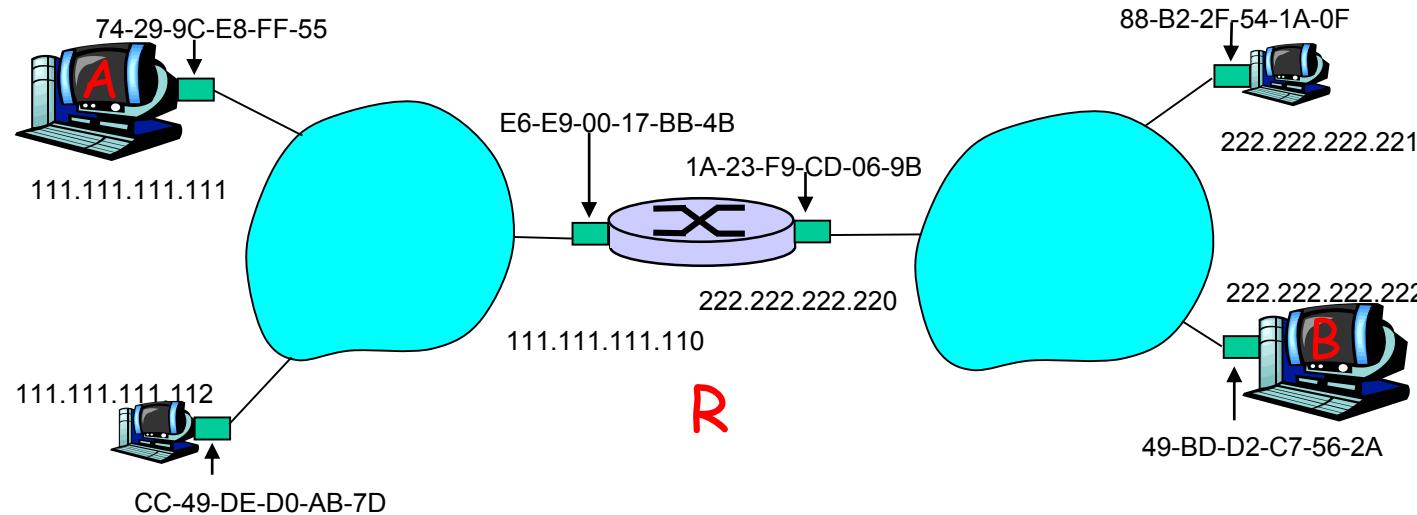
assume A knows B's IP address



- Two ARP tables in router R, one for each IP network (LAN)
- In routing table at source Host, find router 111.111.111.110
- In ARP table at source, find MAC address E6-E9-00-17-BB-4B, etc

- A creates IP datagram with source A, destination B
- A uses ARP to get R's MAC address for 111.111.111.110
- A creates link-layer frame with R's MAC address as dest, frame contains A-to-B IP datagram
- A's NIC sends frame
- R's NIC receives frame
- R removes IP datagram from Ethernet frame, sees its destined to B
- R uses ARP to get B's MAC address
- R creates frame containing A-to-B IP datagram sends to B

This is a **really** important example - make sure you understand!



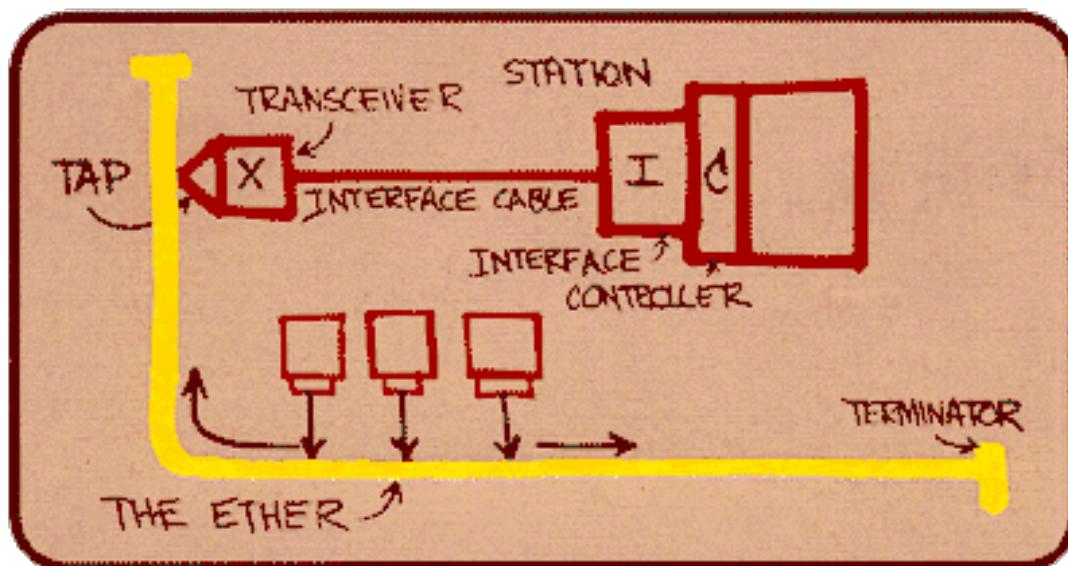
Link Layer

- 5.1 Introduction
 - 5.2 Link Layer Services
 - 5.3 Multiple access protocols
- LAN Technology
 - 5.4 LAN Addresses and ARP
 - **5.5 Ethernet**
 - 5.6 Interconnecting LANs
 - 5.7 Examples of LAN Configurations
- The Whole Enchilada
 - 5.7 How the Web and Internet work
 - 5.8 ATM
 - 5.9 How IP and ATM co-exist

5.5 Introduction to Ethernet

“Dominant” wired LAN technology:

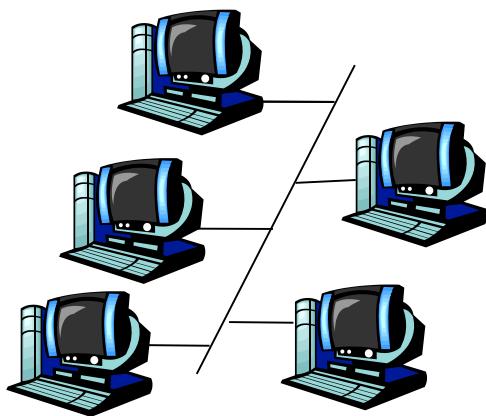
- Cheap, £5 for 1Gbps NIC
- First widely used LAN technology
- Simpler, cheaper than token LANs and ATM
- Kept up with speed race: 10 Mbps – 100 Gbps
 - IEEE now working on 1Tbps Ethernet
 - (http://news.cnet.com/8301-1023_3-57495639-93/ethernets-future-how-fast-is-fast-enough/)



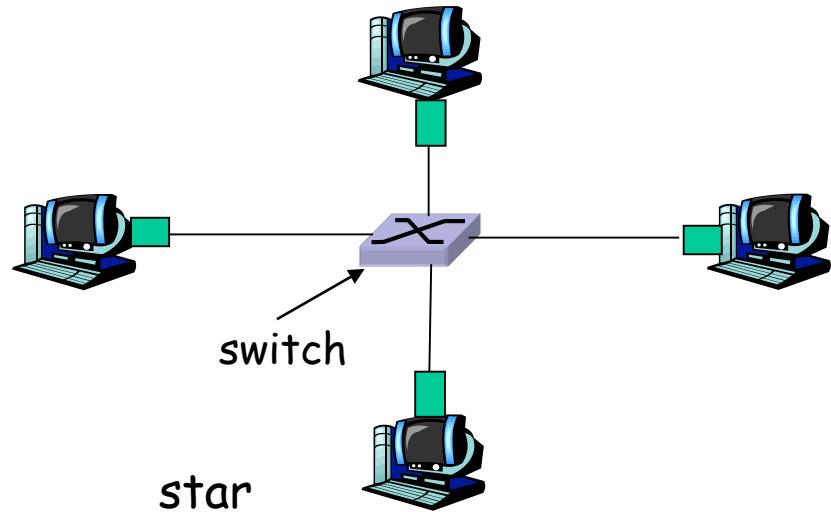
Metcalfe's Ethernet sketch

Popular Ethernet LAN Topologies

- ❑ Bus topology was popular through mid 90s
 - all nodes in same collision domain (can collide with each other)
- ❑ Today: star topology prevails
 - active *switch* in center
 - each “spoke” runs a (separate) Ethernet protocol (nodes do not collide with each other)



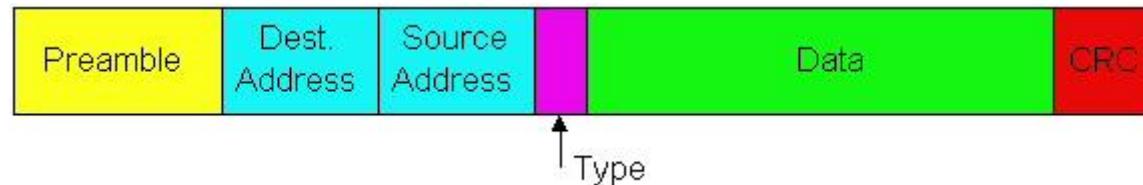
bus: coaxial cable



star

Ethernet Frame Structure

Sending adapter encapsulates IP datagram (or other network layer protocol packet) in **Ethernet frame**

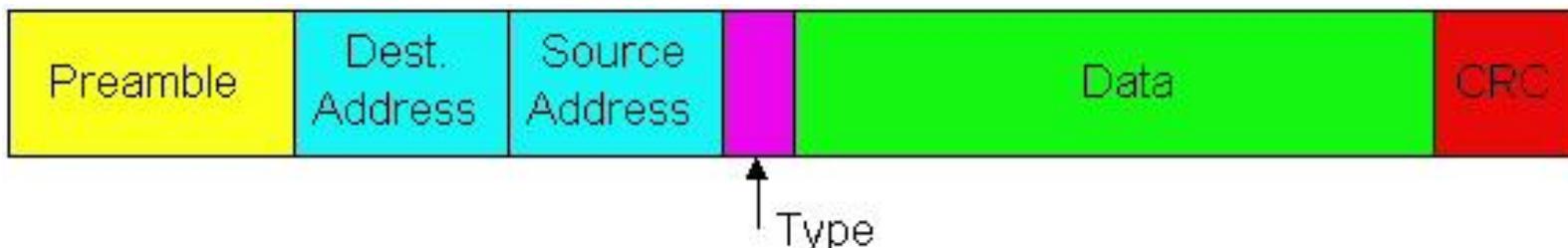


Preamble:

- 7 bytes with pattern 10101010 followed by one byte with pattern 10101011
- used to synchronize receiver, sender clock rates

Ethernet Frame Structure (more)

- **Addresses:** 6 bytes
 - If adapter receives frame with matching destination address, or with broadcast address (eg ARP packet), it passes data in frame to net-layer protocol
 - Otherwise, adapter discards frame
- **Type:** indicates the higher layer protocol (mostly IP but others may be supported such as Novell IPX and AppleTalk)
- **CRC:** checked at receiver, if error is detected, the frame is simply dropped



Unreliable, connectionless service

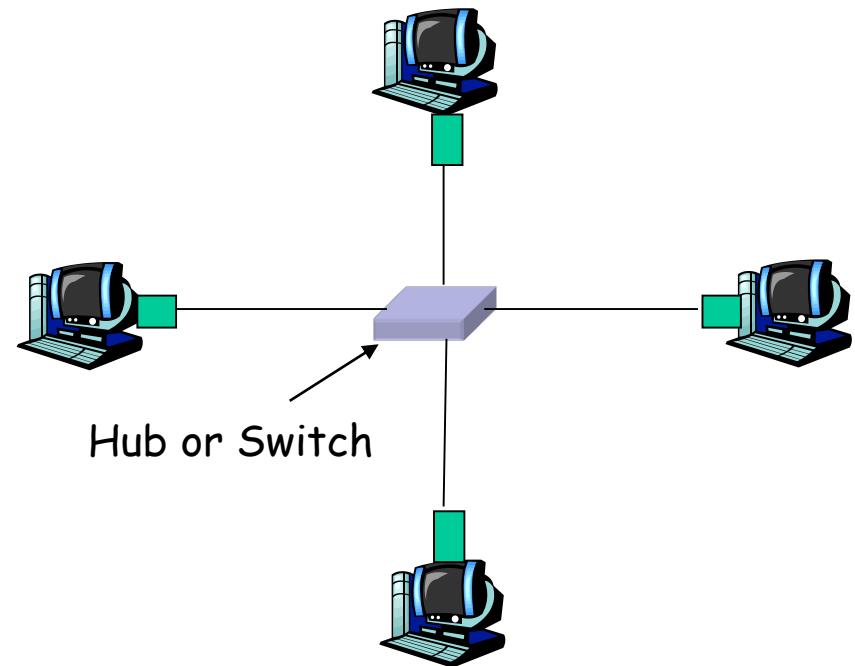
- ❑ **Connectionless:** No handshaking between sending and receiving adapter.
- ❑ **Unreliable:** receiving adapter doesn't send acks or nacks to sending adapter
 - Stream of datagrams passed to network layer can have gaps
 - Gaps will be filled if app is using TCP
 - **Otherwise, app will see the gaps**
- ❑ Ethernet MAC protocol: CSMA/CD

Link Layer

- 5.1 Introduction
 - 5.2 Link Layer Services
 - 5.3 Multiple access protocols
- LAN Technology
 - 5.4 LAN Addresses and ARP
 - 5.5 Ethernet
 - 5.6 Interconnecting LANs
 - 5.7 Examples of LAN Configurations
- The Whole Enchilada
 - 5.7 How the Web and Internet work
 - 5.8 ATM
 - 5.9 How IP and ATM co-exist

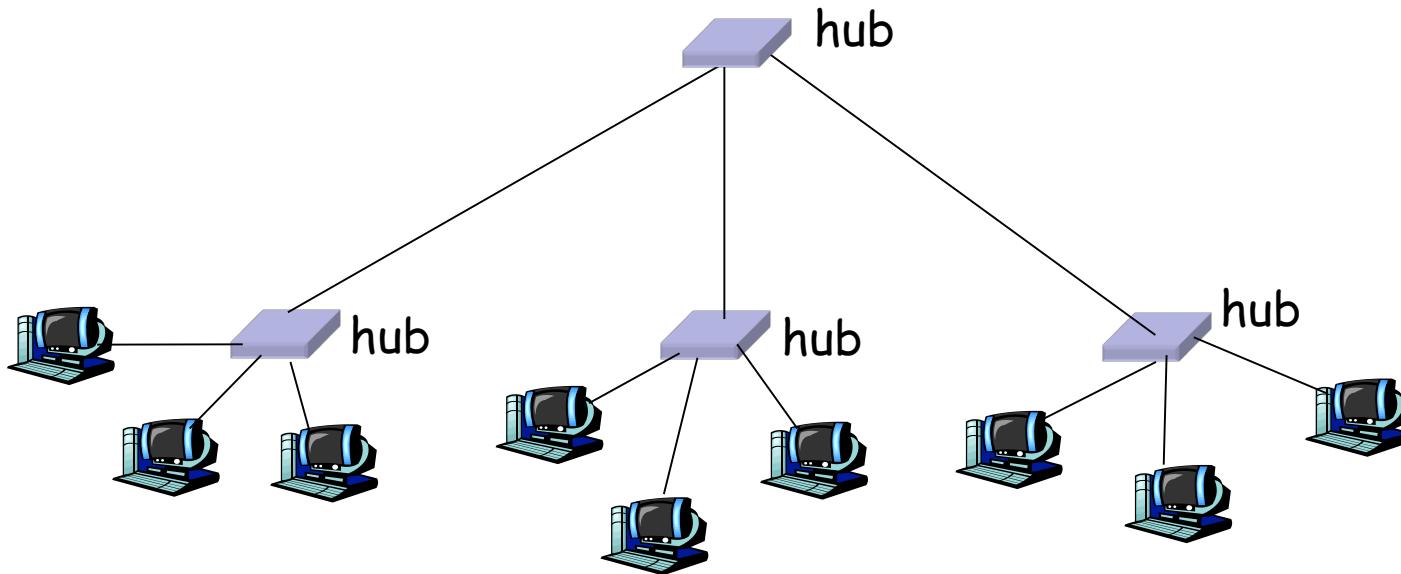
5.6 Interconnecting LANs

- Larger companies need two or more LANs
 - Geography
 - Security
 - Performance
- A device is needed to connect the two (or more) LANs
- Solution:
 - Hubs
 - Bridges
 - Switches
 - Routers?



5.6.1 Interconnecting with Hubs

- ❑ Backbone hub interconnects LAN segments
- ❑ Extends max distance between nodes
- ❑ But individual segment **collision domains** become one large collision domain
- ❑ A hub acts like a blob of solder



5.6.2 Interconnecting with Bridges

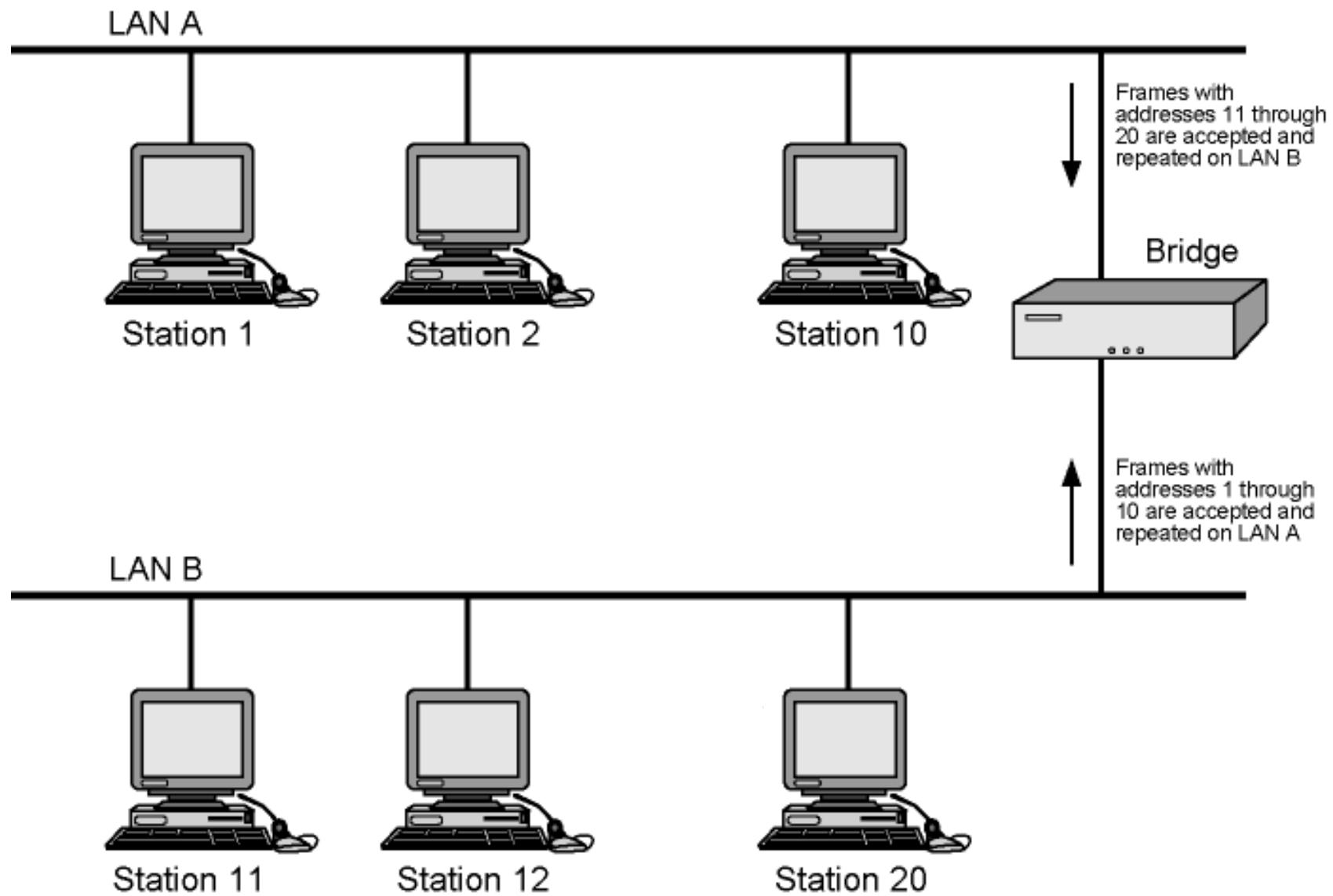
□ Key Features

- Provide interconnection to other LANs/WANs
- Connects similar LANs
- Identical protocols for physical and link layers
- Minimal processing

□ Functions

- Read all frames transmitted on one LAN and accept those address to any station on the other LAN
- Using MAC protocol for second LAN, retransmit each frame
- Do the same the other way round

Bridge Operation



Bridge Design Aspects

- No modification to content or format of frame
- No encapsulation
- Exact bitwise copy of frame
- Minimal buffering to meet peak demand
- Contains routing and address intelligence
 - Must be able to tell which frames to pass
 - May be more than one bridge to cross
- May connect more than two LANs
- Bridging is transparent to stations
 - Appears to all stations on multiple LANs as if they are on one single LAN

5.6.3 Interconnecting with Layer 2 Switches

- ❑ Central hub acts as switch
- ❑ Incoming frame from particular station switched to appropriate output line
- ❑ Unused lines can switch other traffic
- ❑ More than one station transmitting at a time
- ❑ Multiplying capacity of LAN

The Layer 2 Switch Key Features

- link-layer device: smarter than hubs, take *active* role
 - store, forward Ethernet frames
 - examine incoming frame's MAC address, selectively forward frame to one-or-more outgoing links 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

Layer 2 Switch Benefits

- No change to attached devices to convert bus LAN or hub LAN to switched LAN
- For Ethernet LAN, each device uses Ethernet MAC protocol
- Each device has dedicated capacity equal to original LAN
 - Assuming switch has sufficient capacity to keep up with all devices
 - For example, if switch can sustain throughput of 20 Mbps, each device appears to have dedicated capacity for either input or output of 10 Mbps
- Layer 2 switch scales easily
 - Additional devices attached to switch by increasing capacity of layer 2

Types of Layer 2 Switch

□ **Store-and-forward switch**

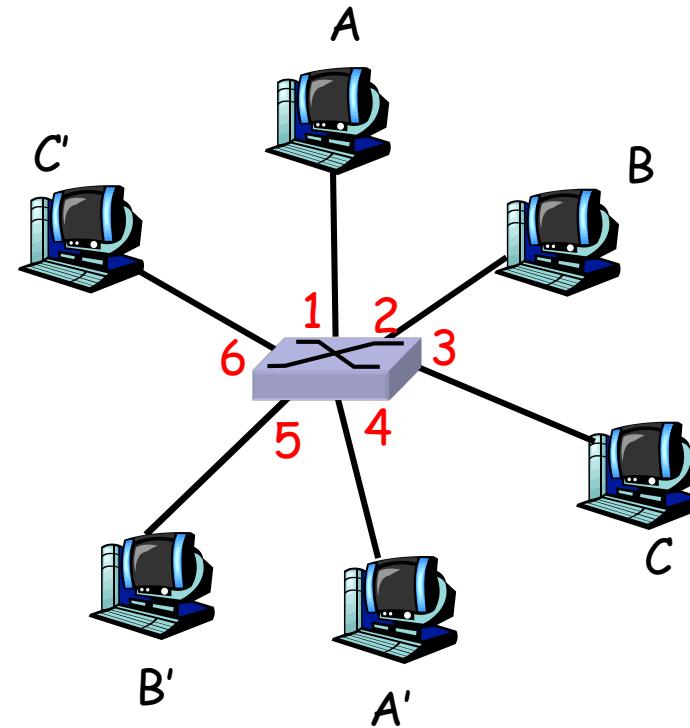
- Accepts frame on input line
- Buffers it briefly
- Then routes it to appropriate output line
- Delay between sender and receiver
- Boosts integrity of network

□ **Cut-through switch**

- Takes advantage of destination address appearing at beginning of frame
- Switch begins repeating frame onto output line as soon as it recognizes destination address
- Highest possible throughput
- Risk of propagating bad frames
 - Switch unable to check CRC prior to retransmission

Switch allows *multiple* simultaneous transmissions

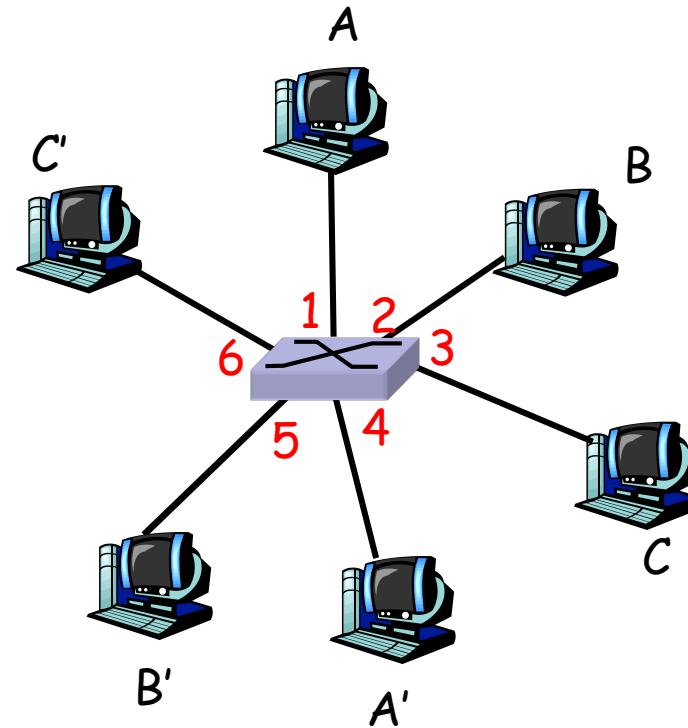
- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, but no collisions
 - full duplex
 - **each link is its own collision domain**
- **switching**: A-to-A' and B-to-B' simultaneously, without collisions
 - not possible with dumb hub or bridge



*switch with six interfaces
(1,2,3,4,5,6)*

5.6.3.1 Switch Table

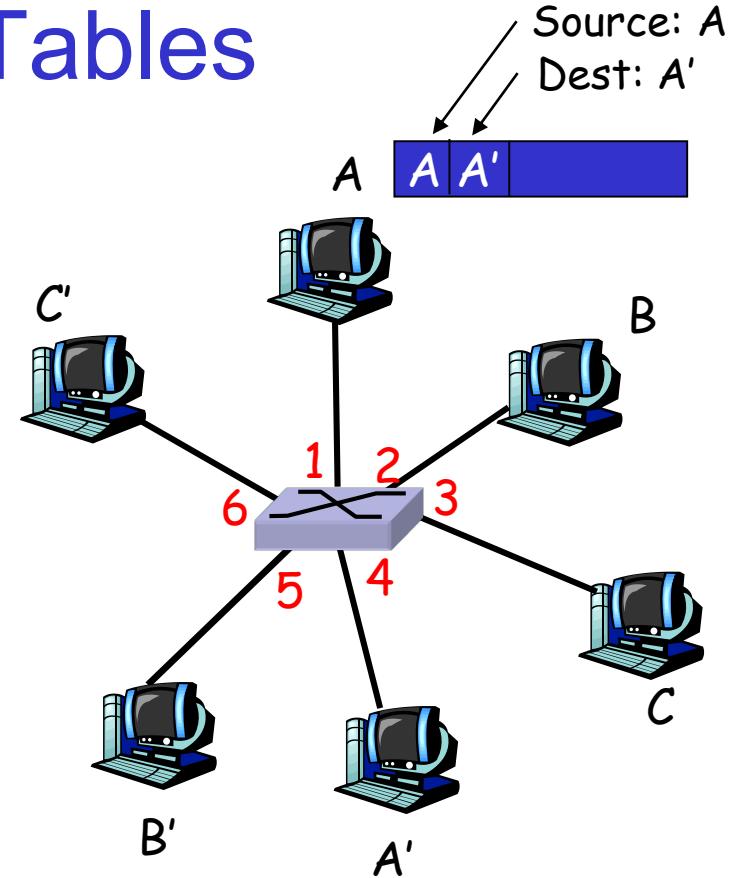
- Q: how does switch know that A' reachable via interface 4, B' reachable via interface 5?
- A: each switch has a **switch table**, each entry:
 - (MAC address of host, interface to reach host, time stamp)
- looks like a routing table!
- Q: how are entries created and maintained in switch table?
 - something like a routing protocol?...



*switch with six interfaces
(1,2,3,4,5,6)*

Self-learning Switch Tables

- Switch *learns* which hosts can be reached through which interfaces
 - when frame received, switch “learns” location of sender: incoming LAN segment
 - records sender/location pair in switch table



MAC addr	interface	TTL
A	1	60

Switch table
(initially empty)

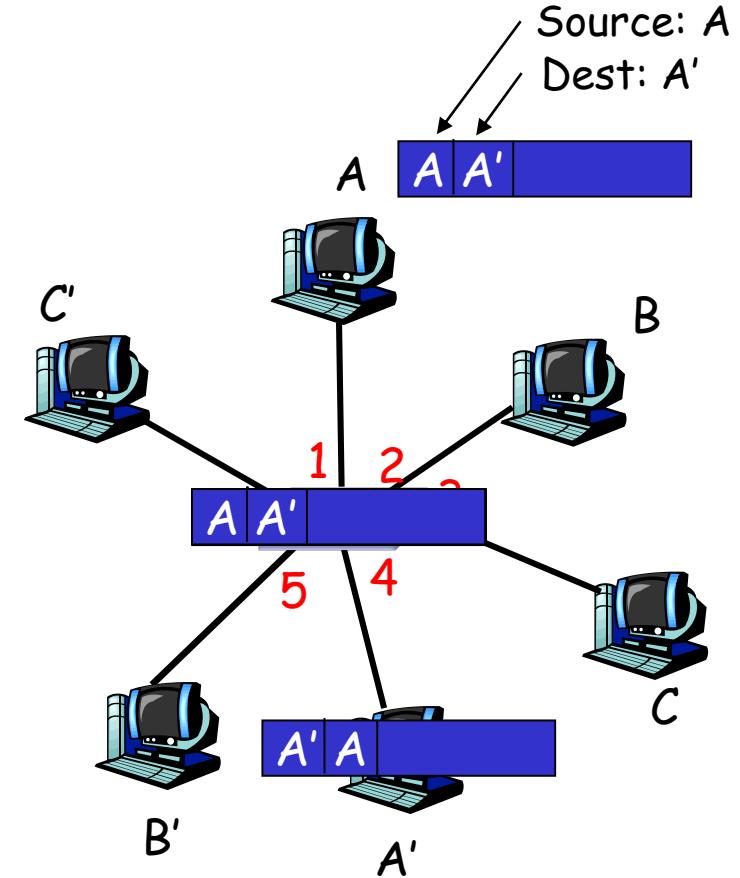
Switch: frame filtering/forwarding

When frame received:

1. record link associated with sending host
 2. index switch table using MAC dest address
 3. **if** entry found for destination
then {
 - if** dest on segment from which frame arrived
then drop the frame
 - else** forward the frame on interface indicated**}**
 - else** flood
- forward on all but the interface
on which the frame arrived*

Self-learning, forwarding: example

- frame destination unknown: *flood*
- destination A location known: *selective send*

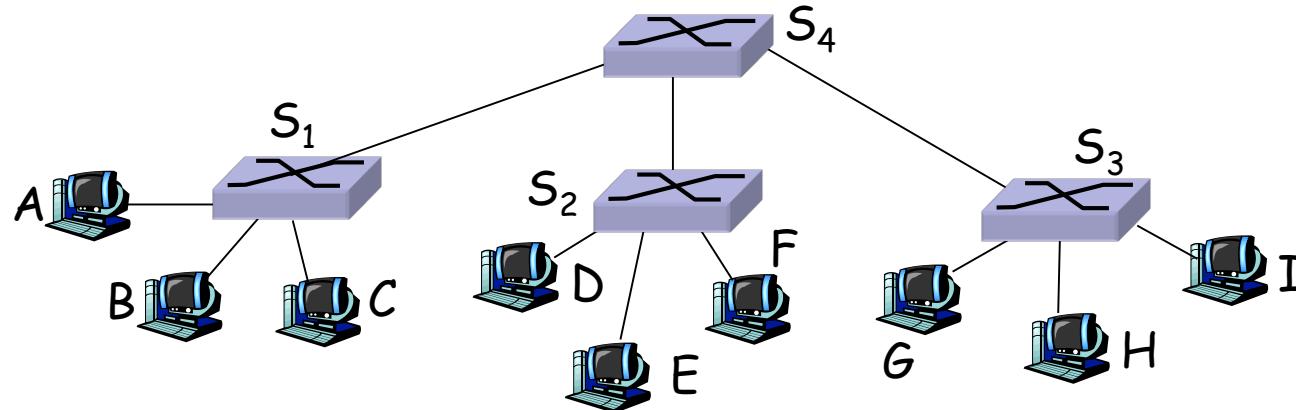


MAC addr	interface	TTL
A	1	60
A'	4	60

Switch table
(initially empty)

5.6.3.2 Interconnecting switches

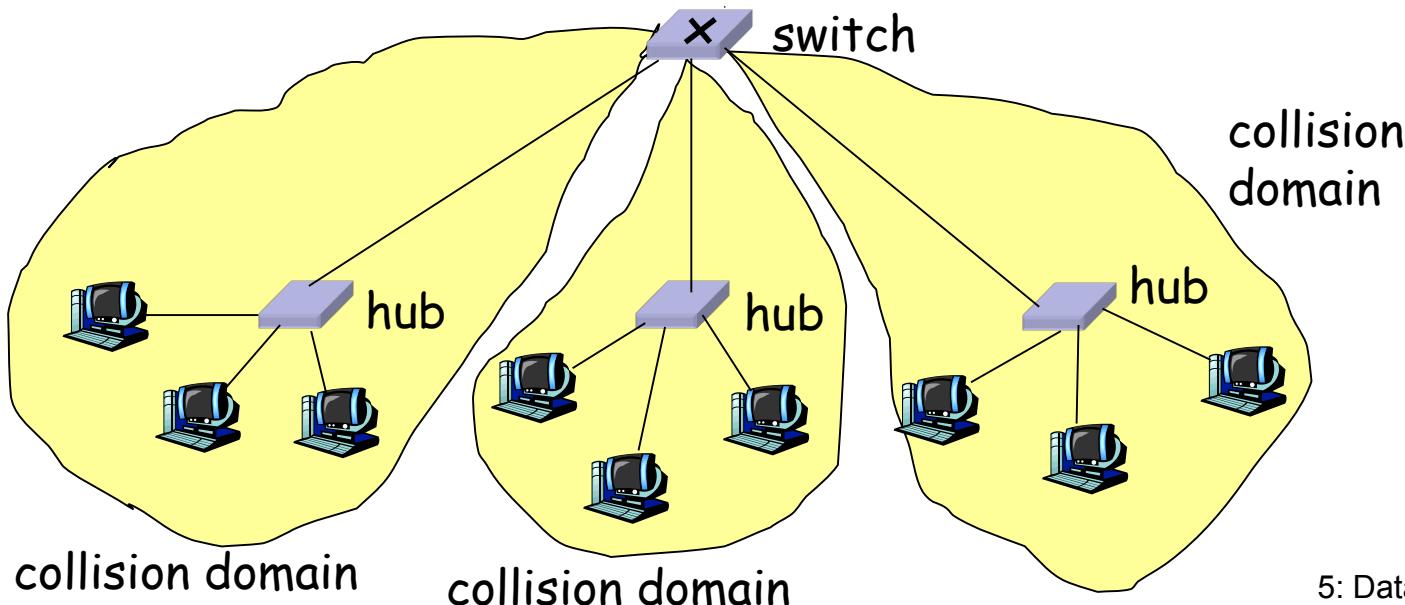
- Switches can be connected together



- ***Q:*** sending from A to G - how does S_1 know to forward frame destined to F via S_4 and S_3 ?
- ***A:*** self learning! (works exactly the same as in single-switch case!)

Switch: traffic isolation

- switch installation breaks subnet into LAN segments
- switch **filters** packets:
 - same-LAN-segment frames not usually forwarded onto other LAN segments
 - segments become separate **collision domains**



5.6.4 Comparing Switches, Bridges and Hubs

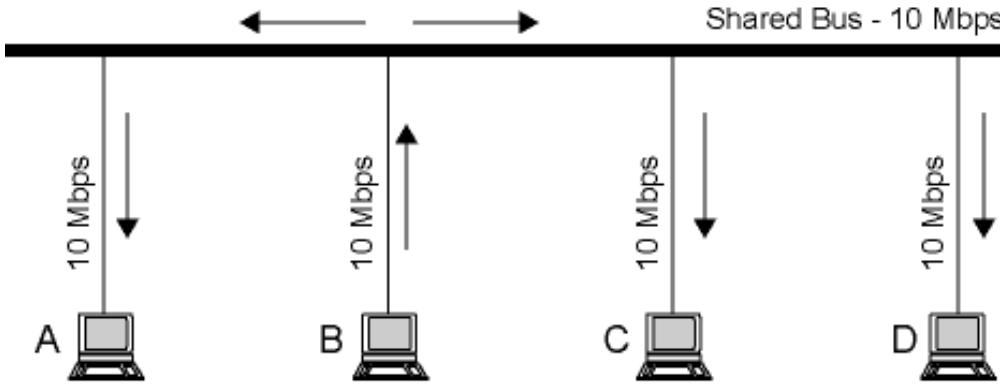
- All 3 perform similar role
 - Transparently Interconnect different LANs
- Each has different characteristics and price
- Which you choose depends on network requirement
- Consider:**
 - Performance (throughput)
 - Scalability (number of hosts supported)
 - Reliability (redundant routes)
 - Cost
 - Dedicated access vs collisions

Switch v Bridge

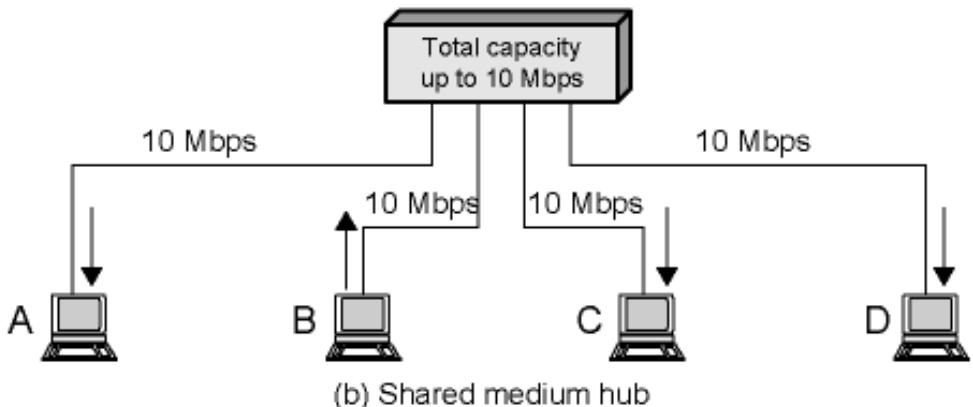
- Layer 2 switch can be viewed as full-duplex hub
 - Can incorporate logic to function as multiport bridge
- Bridge frame handling done in software
- Switch performs address recognition and frame forwarding in hardware
- Bridge only analyzes and forwards one frame at a time
- Switch has multiple parallel data paths
 - Can handle multiple frames at a time
- Bridge uses store-and-forward operation
- Switch can have cut-through operation
- Bridge suffered commercially
 - New installations typically include layer 2 switches with bridge functionality rather than bridges

Switch vs Hub

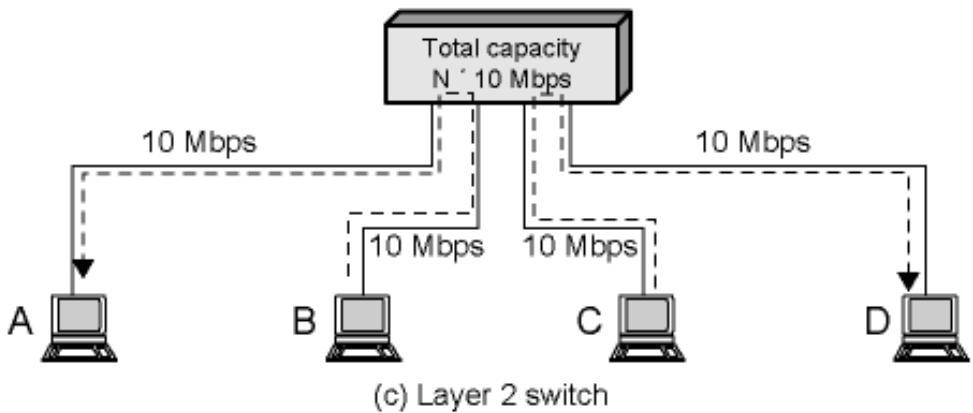
- ❑ Hub has no intelligence
- ❑ Extends collision domain
- ❑ Switch has no collisions
- ❑ Only advantage of hub is cost and reliability



(a) Shared medium bus



(b) Shared medium hub



(c) Layer 2 switch

Switch vs. Router

- Both store-and-forward devices
 - routers: network layer devices (examine network layer headers)
 - Switches: link layer devices (examine link layer headers)
- Routers maintain routing tables, implement routing algorithms
- Switches maintain switch tables, implement filtering, learning algorithms
- Routers divide LANs into separate networks (subnets)
- Switches merge separate networks into one network
- So which do you choose?...

5.6.5 Problems with Layer 2 Switches

- Layer 2 switches are limited when LAN grows
 - Broadcast storms
 - Lack of multiple paths

5.6.5.1 Broadcast Storms

- ❑ The set of devices and LANs connected by layer 2 switches have flat address space
- ❑ All users share common MAC broadcast address
- ❑ If any device issues broadcast frame, that frame is delivered to all devices attached to network connected by layer 2 switches and/or bridges
- ❑ In large network, broadcast frames can create big overhead
- ❑ Malfunctioning device can create **broadcast storm**
 - Numerous broadcast frames clog network

5.6.5.2 Lack of Multiple Paths

- Current standards for bridge protocols dictate no closed loops
 - Only one path between any two devices
 - Impossible in standards-based implementation to provide multiple paths through multiple switches between devices
 - **This limits both performance and reliability**
- *Solution:* break up network into subnetworks connected by routers
 - MAC broadcast frame limited to devices and switches contained in single subnetwork
 - IP-based routers employ sophisticated routing algorithms
 - Allow use of multiple paths between subnetworks going through different routers

Problems with Routers

- ❑ Routers do all IP-level processing in software
 - High-speed LANs and high-performance layer 2 switches pump millions of packets per second
 - Software-based router only able to handle well under a million packets per second
- ❑ Is there a solution?...
- ❑ ...Yes! (obviously!)

5.6.6 The Layer 3 Switch

- ❑ Layer 3 Switch implements packet-forwarding logic of router in hardware
- ❑ Hybrid approach – part switch, part router
- ❑ Two categories
 - Packet by packet
 - Flow based
- ❑ Packet by Packet
 - Operates in same way as traditional router
 - Order of magnitude increase in performance compared to software-based router

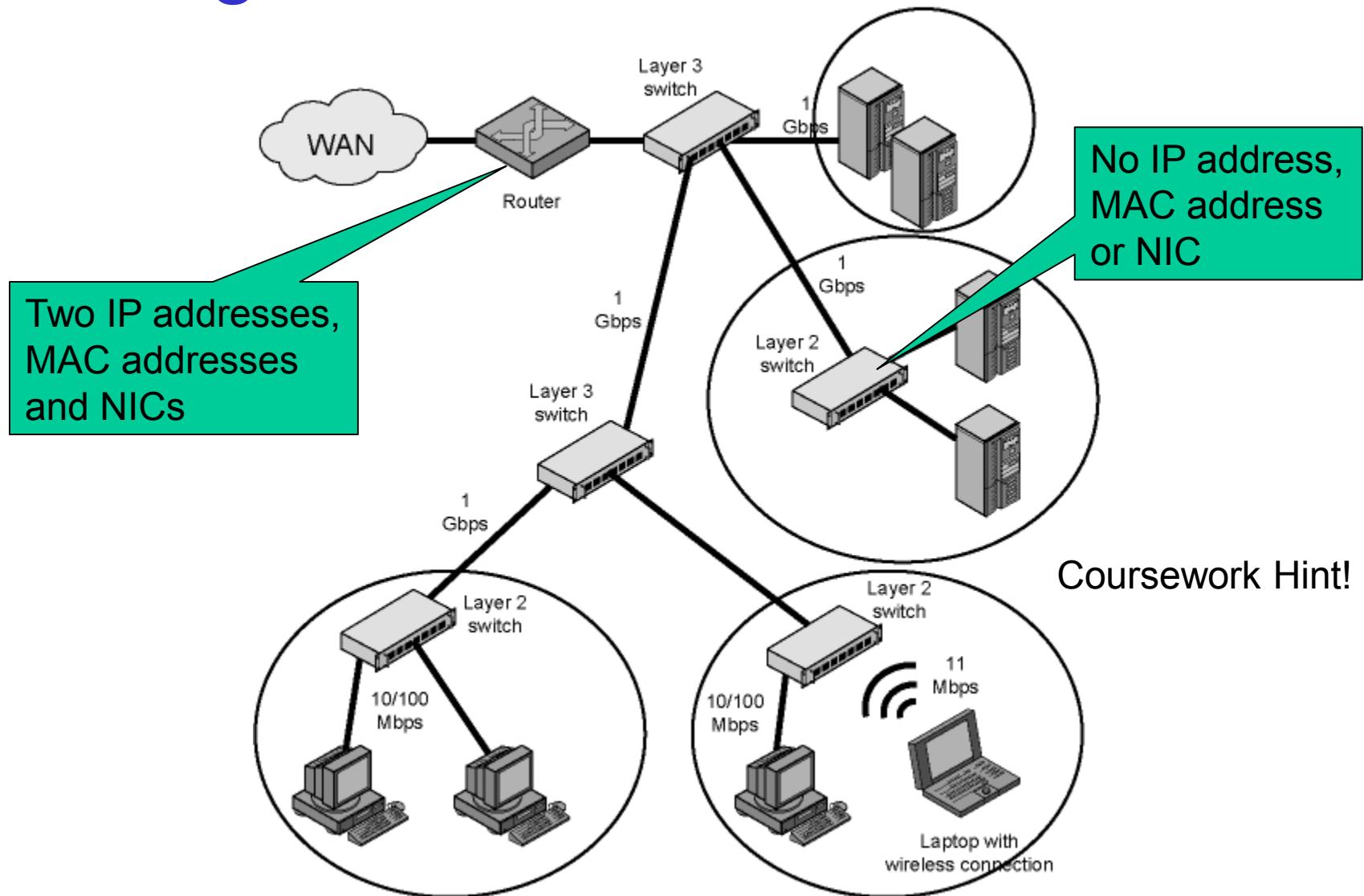
Flow Based Layer 3 Switch

- Operates in same way as traditional router
- Order of magnitude increase in performance compared to software-based router
- Flow-based switch tries to enhance performance by identifying flows of IP packets
 - Same source and destination
 - Done by observing ongoing traffic or using a special flow label in packet header (IPv6)
 - Once flow is identified, predefined route can be established

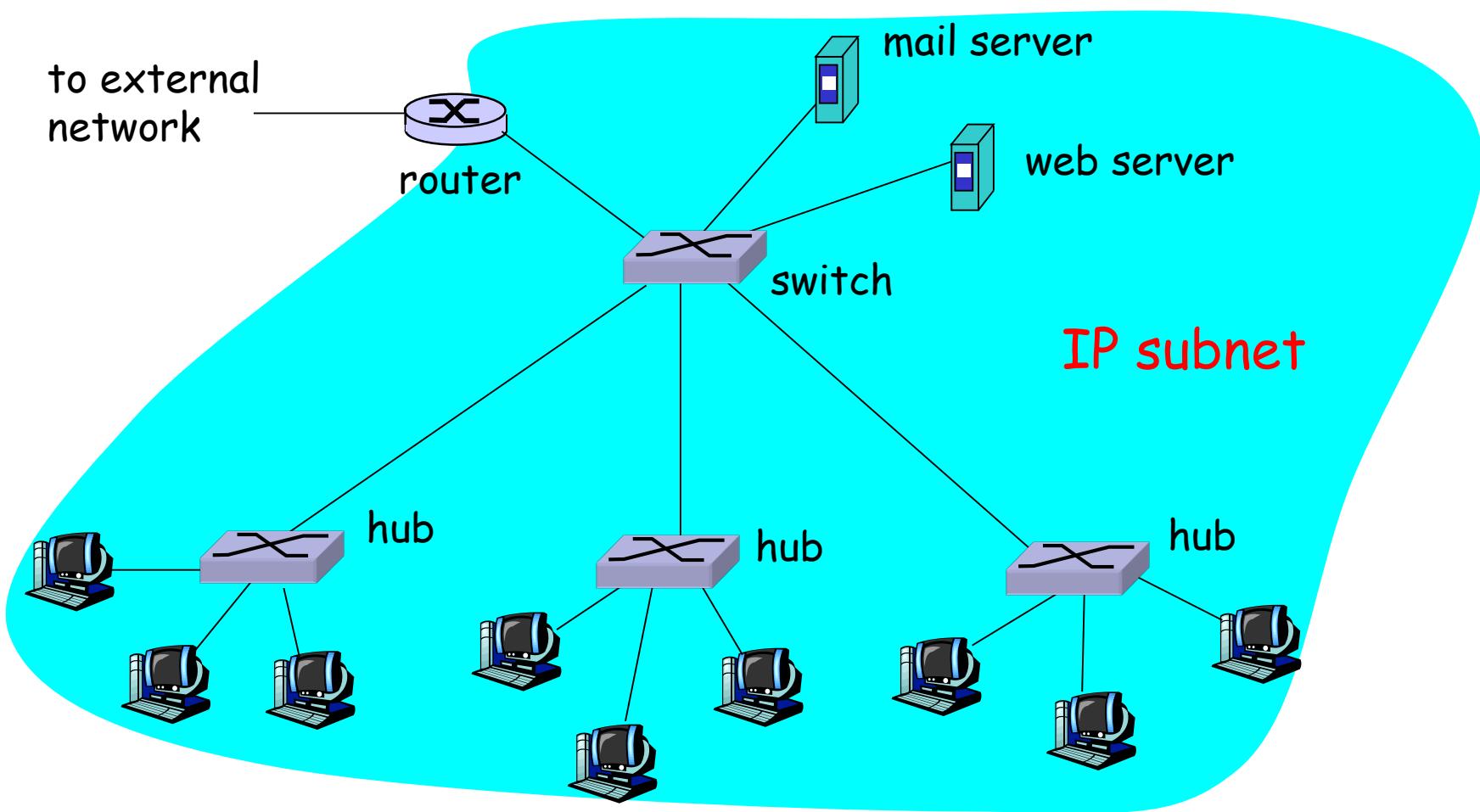
5.6.7 Examples of LAN Configuration

- ❑ Thousands to tens of thousands of devices
- ❑ Desktop systems links 10 Mbps to 100 Mbps
 - Into layer 2 switch
- ❑ Wireless LAN connectivity available for mobile users
- ❑ Layer 3 switches at local network's core
 - Form local backbone
 - Interconnected at 1 Gbps
 - Connect to layer 2 switches at 100 Mbps to 1 Gbps
- ❑ Servers connect directly to layer 2 or layer 3 switches at 1 Gbps
- ❑ Lower-cost software-based router provides WAN connection
- ❑ Circles in diagram identify separate LAN subnetworks
- ❑ MAC broadcast frame limited to own subnetwork

Example 1: Typical Network Configuration

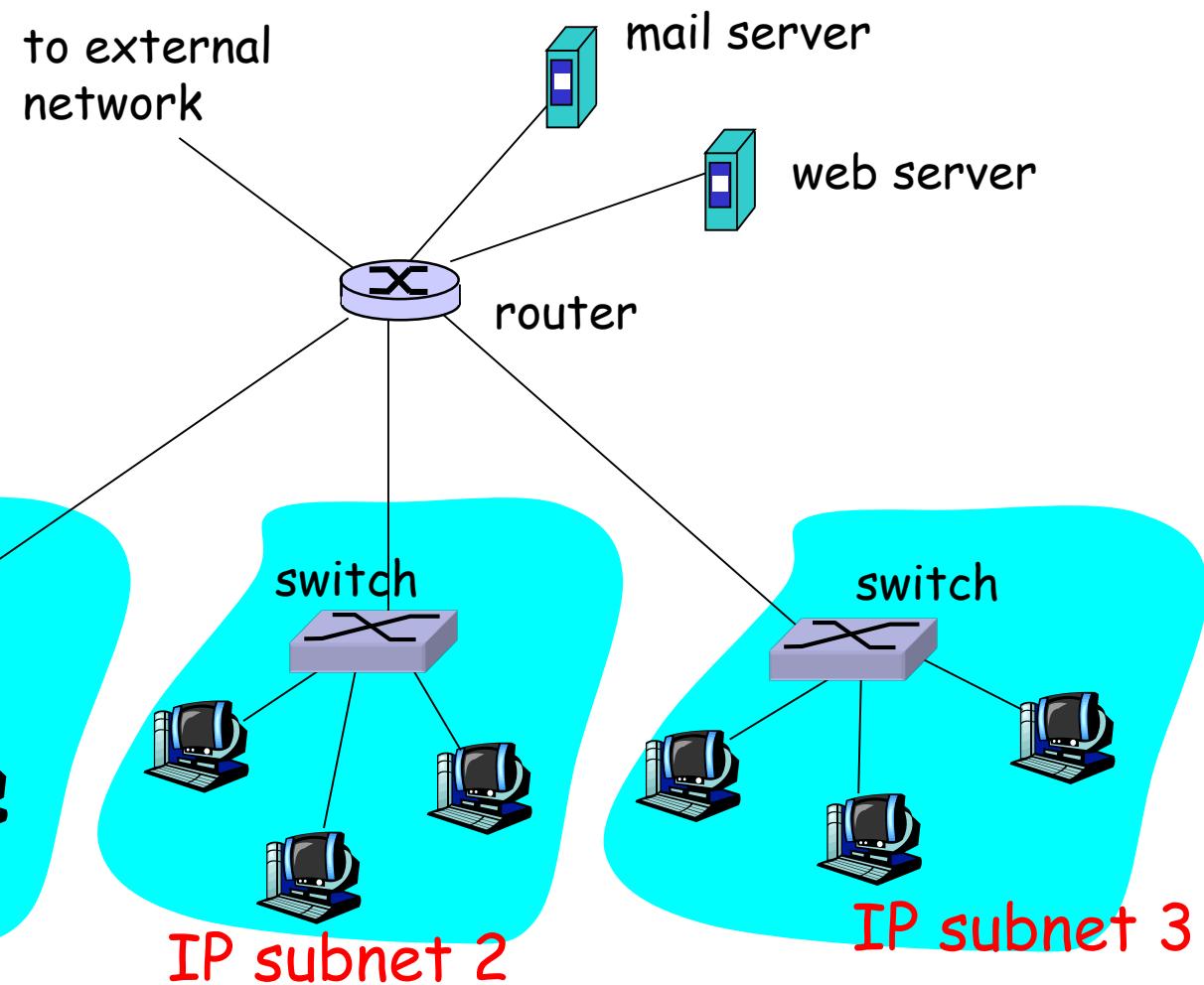


Example 2: Institutional network



Example 3: Subnetted network

Q: Do hosts connected to the same switch need to talk to the router to communicate?



Link Layer

- 5.1 Introduction
 - 5.2 Link Layer Services
 - 5.3 Multiple access protocols
- LAN Technology
 - 5.4 LAN Addresses and ARP
 - 5.5 Ethernet
 - 5.6 Interconnecting LANs
 - 5.7 Examples of LAN Configurations
- The Whole Enchilada
 - 5.7 How the Web and Internet work
 - 5.8 ATM
 - 5.9 How IP and ATM co-exist

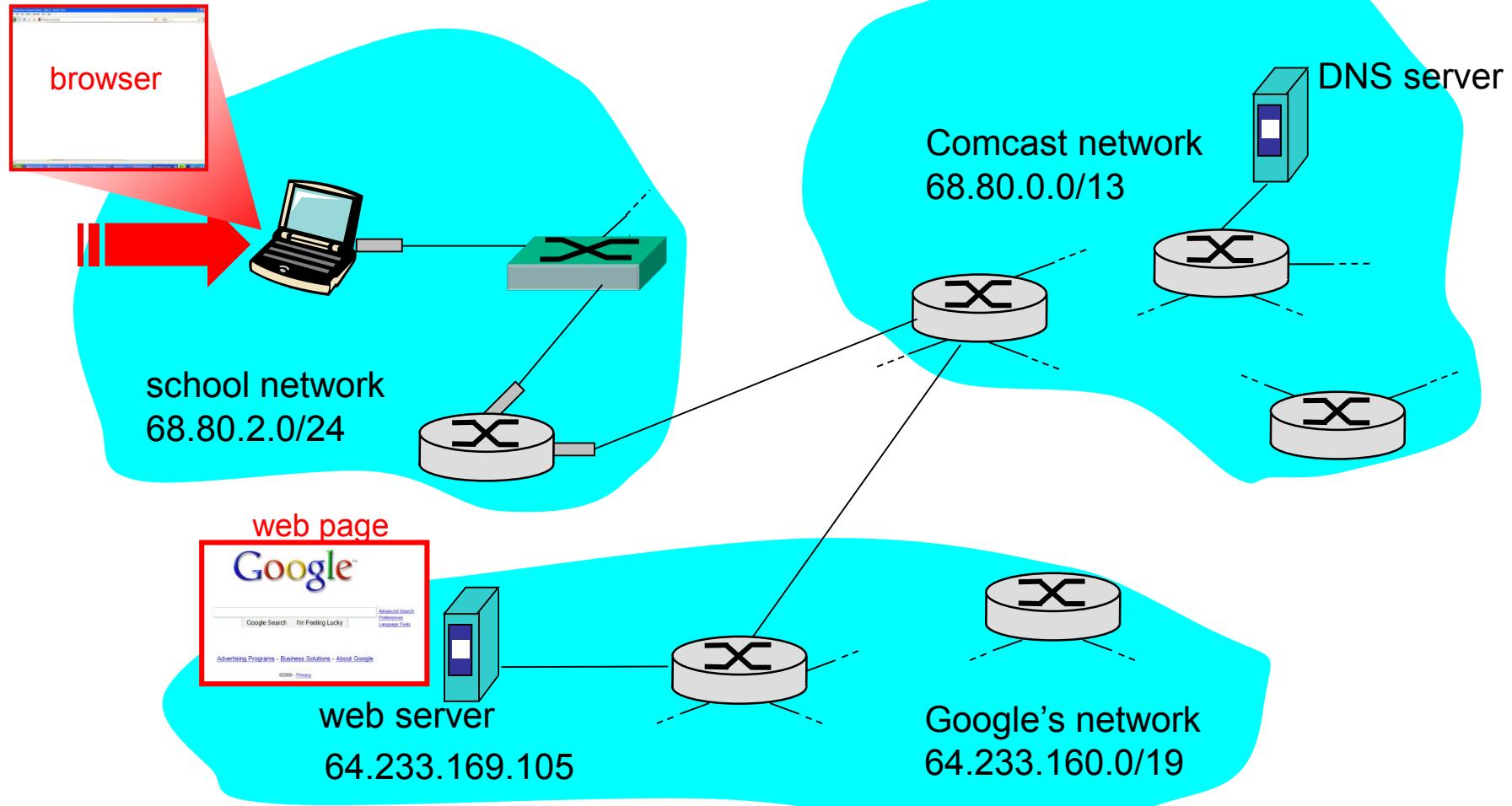
The Whole Enchilada

- **Synthesis: A day in the life of a Web request**
 - Or How the Web and the Internet actually work
- **ATM: The telecoms approach**
 - Or A Very Different Enchilada

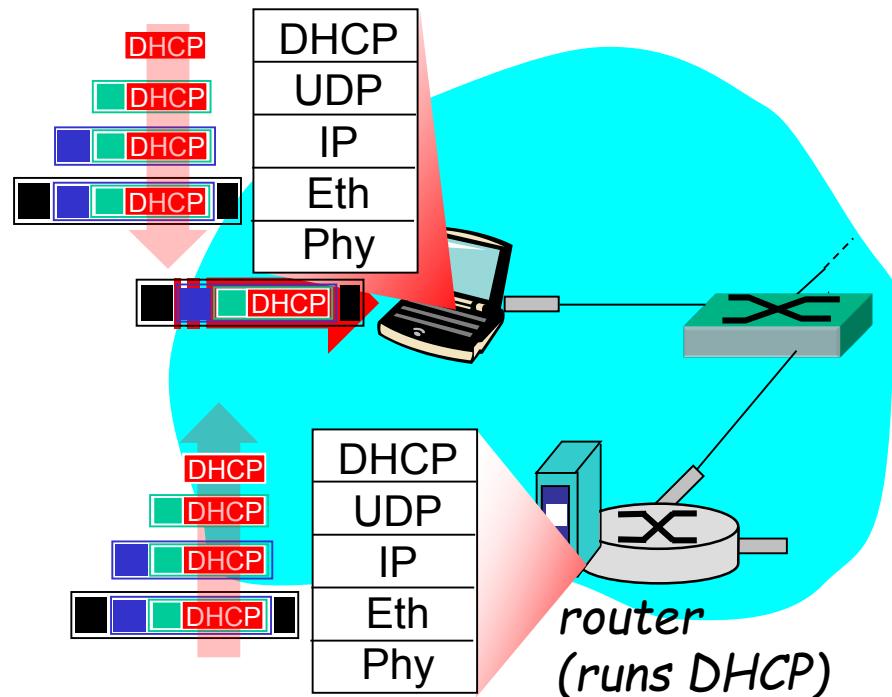
5.7 Synthesis: a day in the life of a web request

- journey down protocol stack complete!
 - application, transport, network, link
- putting-it-all-together: synthesis!
 - *goal*: identify, review, understand protocols (at all layers) involved in seemingly simple scenario: requesting www page
 - *scenario*: student attaches laptop to campus network, requests/receives www.google.com

A day in the life: scenario

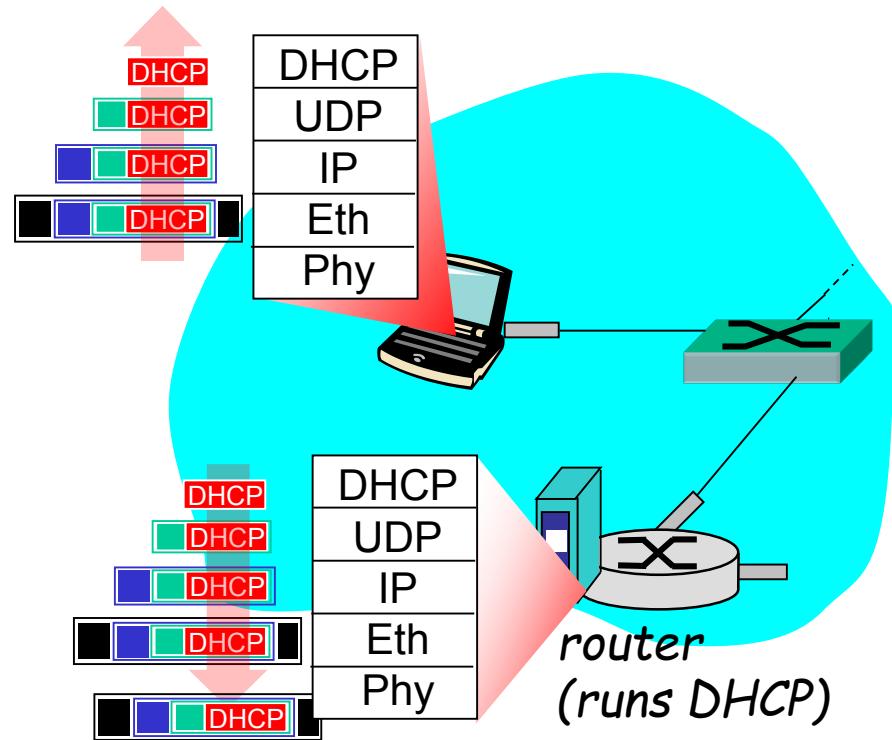


A day in the life... connecting to the Internet



- connecting laptop needs to get its own IP address, addr of first-hop router, addr of DNS server: use **DHCP**
- DHCP request *encapsulated* in **UDP**, encapsulated in **IP**, encapsulated in **802.1** Ethernet
- Ethernet frame *broadcast* (dest: FFFFFFFFFFFF) on LAN, received at router running **DHCP** server
- Ethernet *demux'ed* to IP demux'ed, UDP demux'ed to DHCP

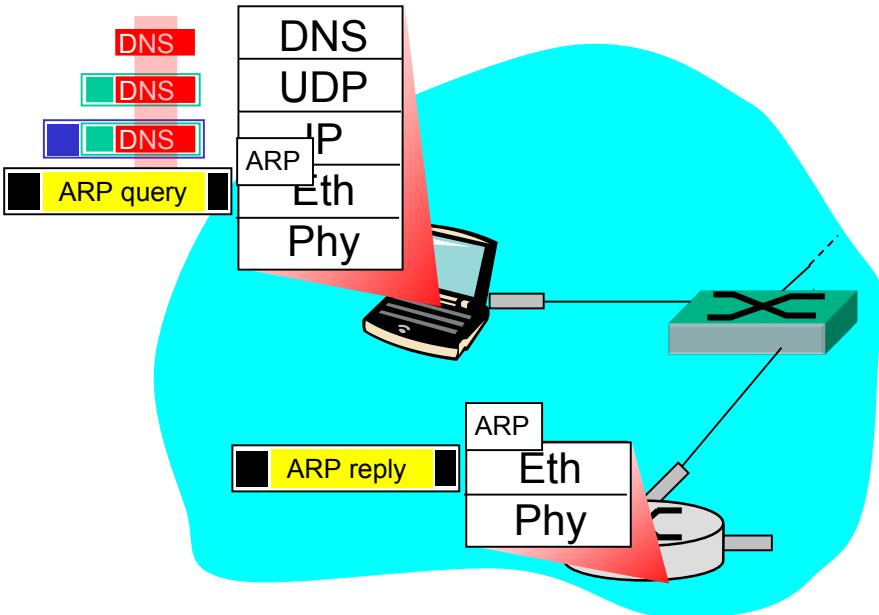
A day in the life... connecting to the Internet



- DHCP server formulates **DHCP ACK** containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- Encapsulation at DHCP server, frame forwarded (**switch learning**) through LAN, demultiplexing at client
- DHCP client receives DHCP ACK reply

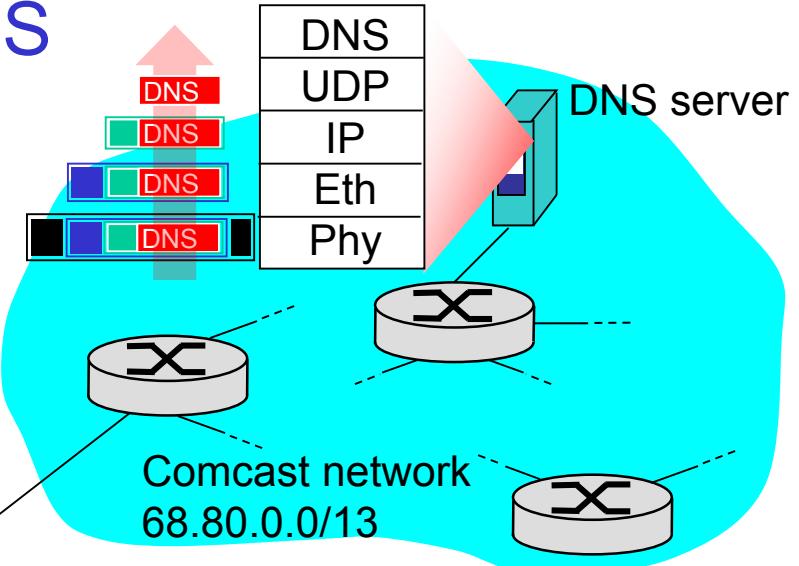
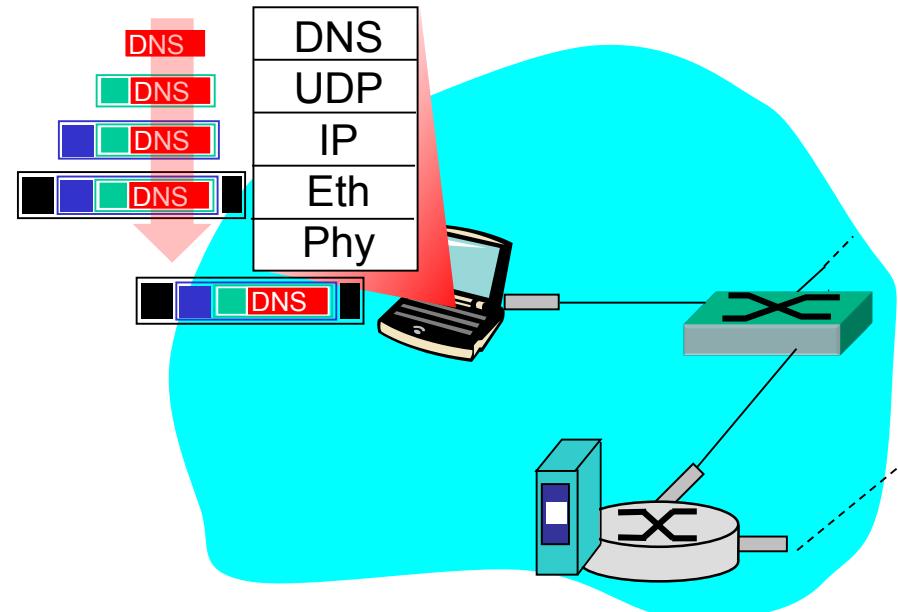
Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

A day in the life... ARP (before DNS, before HTTP)



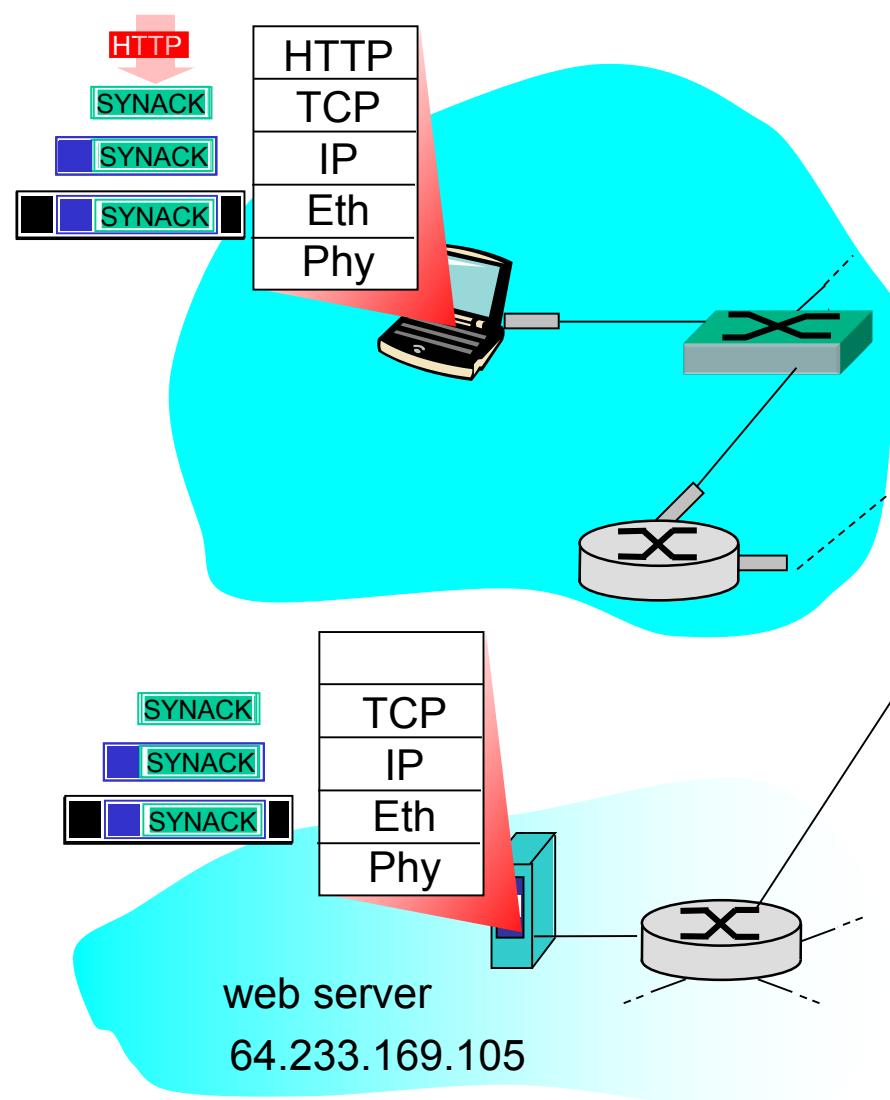
- ❑ before sending **HTTP** request, need IP address of www.google.com: **DNS**
- ❑ DNS query created, encapsulated in UDP, encapsulated in IP, encapsulated in Eth. In order to send frame to router, need MAC address of router interface: **ARP**
- ❑ **ARP query** broadcast, received by router, which replies with **ARP reply** giving MAC address of router interface
- ❑ client now knows MAC address of first hop router, so can now send frame containing DNS query

A day in the life... using DNS



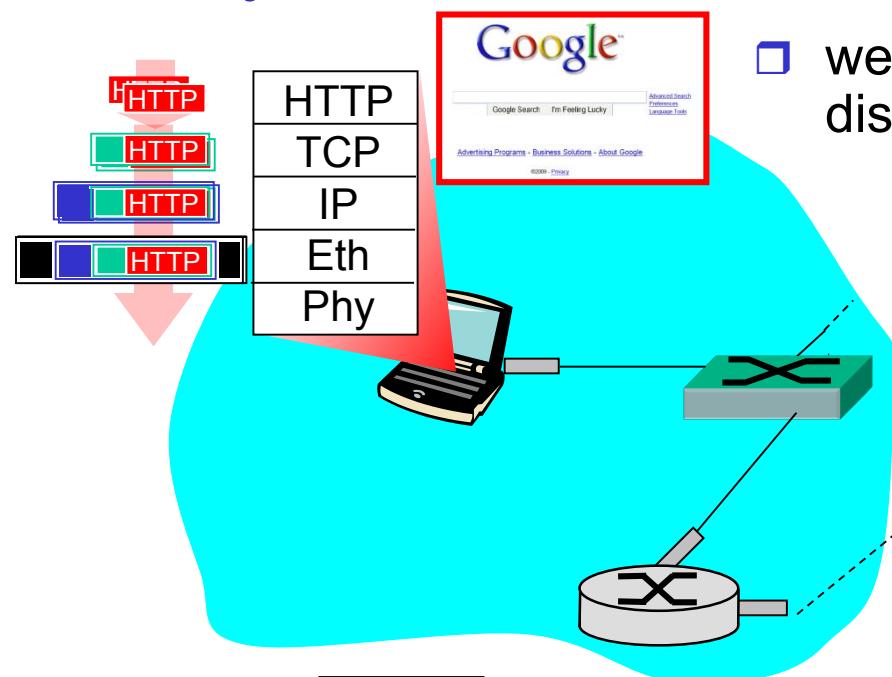
- IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router
- IP datagram forwarded from campus network into comcast network, routed (tables created by **RIP, OSPF, IS-IS** and/or **BGP** routing protocols) to DNS server
- demux'ed to DNS server
- DNS server replies to client with IP address of www.google.com

A day in the life... TCP connection carrying HTTP

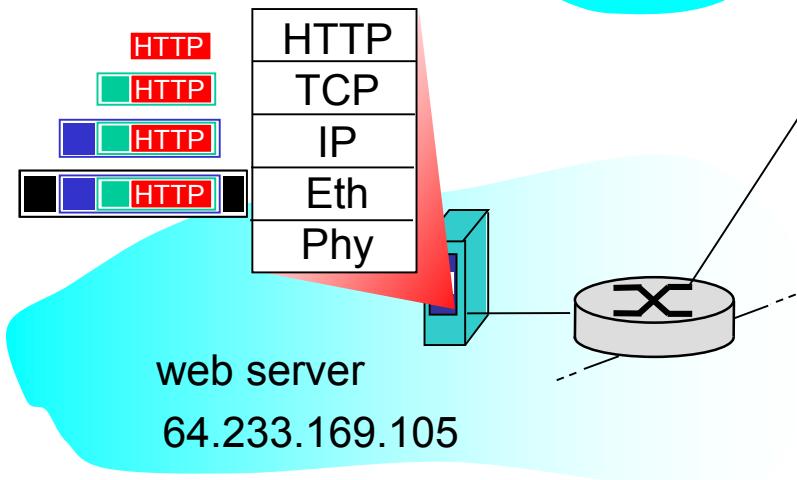


- to send HTTP request, client first opens **TCP socket** to web server
- TCP **SYN segment** (step 1 in 3-way handshake) **inter-domain routed** to web server
- web server responds with **TCP SYNACK** (step 2 in 3-way handshake)
- TCP **connection established!**

A day in the life... HTTP request/reply



- web page ***finally (!!!)*** displayed



- ***HTTP request*** sent into TCP socket
- IP datagram containing HTTP request routed to www.google.com
- web server responds with ***HTTP reply*** (containing web page)
- IP datagram containing HTTP reply routed back to client

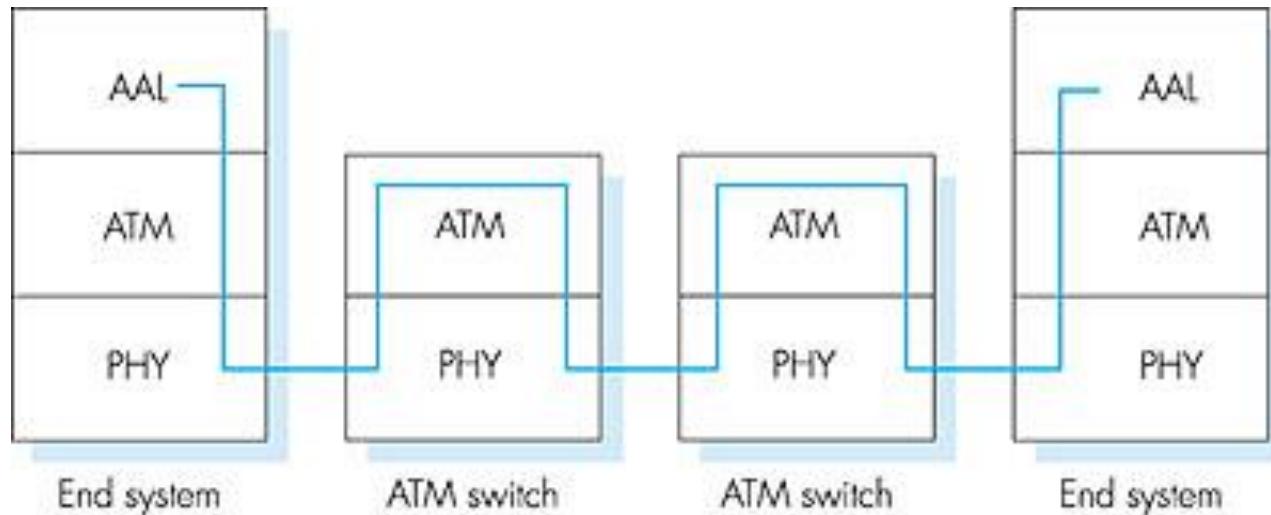
Link Layer

- 5.1 Introduction
 - 5.2 Link Layer Services
 - 5.3 Multiple access protocols
- LAN Technology
 - 5.4 LAN Addresses and ARP
 - 5.5 Ethernet
 - 5.6 Interconnecting LANs
 - 5.7 Examples of LAN Configurations
- The Whole Enchilada
 - 5.7 How the Web and Internet work
 - **5.8 ATM**
 - 5.9 How IP and ATM co-exist

5.8 Asynchronous Transfer Mode: ATM

- **1990's/00 standard for high-speed** (155Mbps to 622 Mbps and higher) *Broadband Integrated Service Digital Network* architecture
- Goal: integrated, end-end transport of voice, video, data
 - Meeting timing/QoS requirements of voice, video (versus Internet best-effort model)
 - “Next generation” telephony: technical roots in telephone world
 - Packet-switching (fixed length packets, called “cells”) using virtual circuits

5.8.1 ATM architecture



- **Adaptation layer:** only at edge of ATM network
 - Data segmentation/reassembly
 - Roughly analogous to Internet transport layer
- **ATM layer:** “network” layer
 - Cell switching, routing
- **Physical layer**

ATM: Network or Link layer?

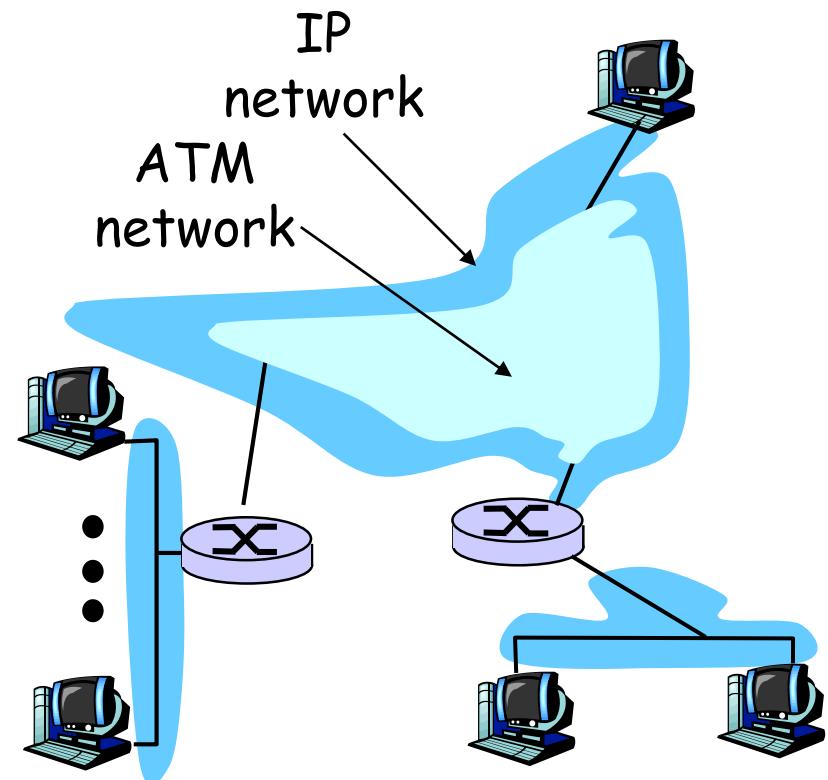
Vision: end-to-end

transport: “ATM from desktop to desktop”

- ATM *is* a network technology

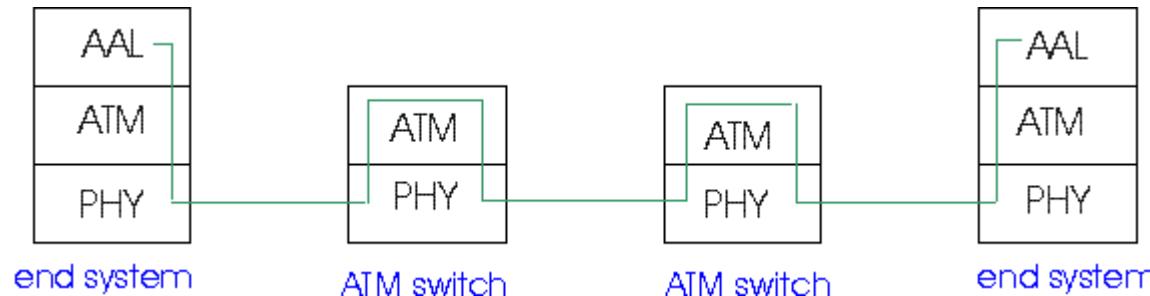
Reality: used to connect IP backbone routers

- “IP over ATM”
- ATM as switched link layer, connecting IP routers



ATM Adaptation Layer (AAL)

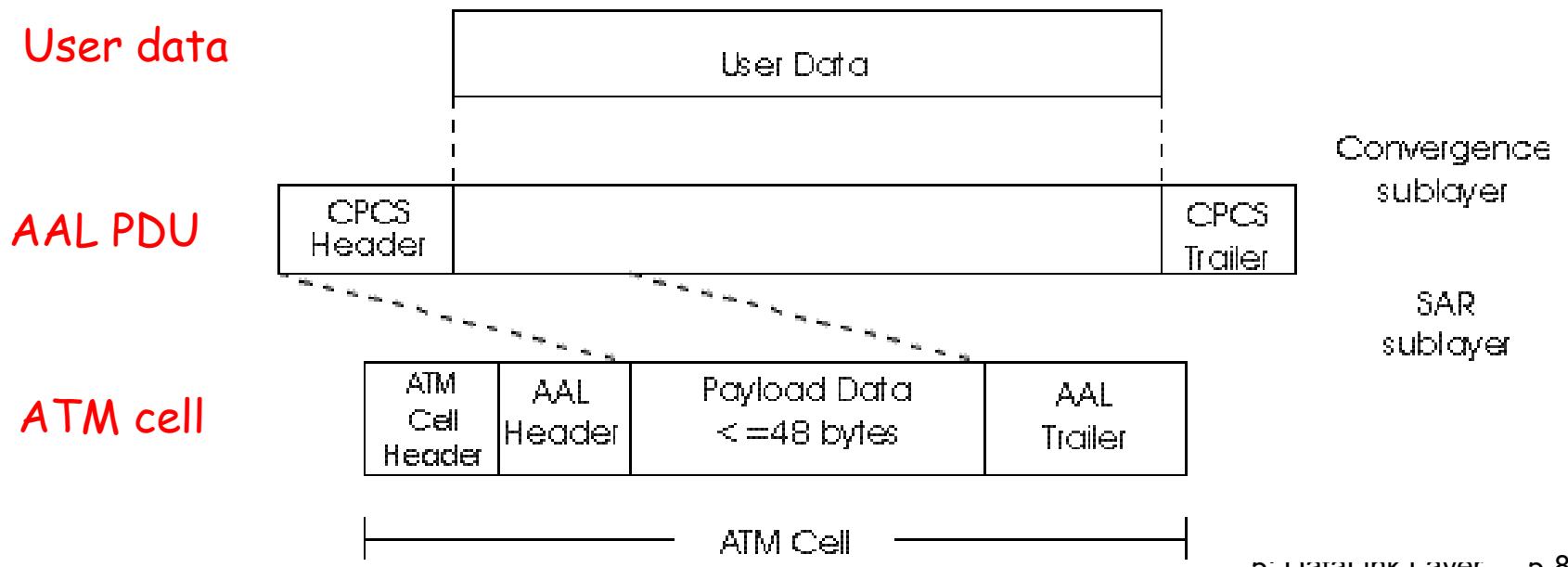
- ATM **Adaptation Layer** (AAL): “adapts” upper layers (IP or native ATM applications) to ATM layer below
- AAL present **only in end systems**, not in switches
- AAL segment (header/trailer fields, data) fragmented across multiple ATM cells
 - Analogy: TCP segment in many IP packets



ATM Adaptation Layer (AAL) [more]

Different versions of AAL layers, depending on ATM service class:

- **AAL1**: for CBR (Constant Bit Rate) services, e.g. circuit emulation
- **AAL2**: for VBR (Variable Bit Rate) services, e.g., MPEG video
- **AAL5**: for data (eg, IP datagrams)



5.8.2 ATM Layer

Service: transport cells across ATM network

- Analogous to IP network layer
- Very different services than IP network layer

Network Architecture	Service Model	Guarantees ?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

5.8.2.1 ATM Service Categories

- ATM network designed to transfer many different types of traffic simultaneously
 - Voice
 - Video
 - Bursty TCP flows
- Each traffic flow is handled the same
 - Stream of **53 byte cells** through VCC
- The way in which each flow is handled in the network differs
 - According to traffic flow
 - And application requirements

ATM Service Categories Summarized

Real time

- Constant bit rate (CBR)
- Real time variable bit rate (rt-VBR)

Non-real time

- Non-real time variable bit rate (nrt-VBR)
- Available bit rate (ABR)
- Unspecified bit rate (UBR)
- Guaranteed frame rate (GFR)

a). Real Time Services

- Real Time and Non-Real Time differentiated by tolerance to:
 - Amount of delay
 - Variation of delay (jitter)
- E.g.
 - File transfer
 - Email
 - Web browsing
 - Streaming media
 - (Videoconferencing, audio streaming, etc.)
- Apps involving interaction between people have tight constraints on delay
- > 300 milliseconds is annoying
- Demands for switching and delivery of Real Time data is high
- ATM has two Real-Time Services

b). Constant Bit Rate

- Fixed data rate continuously available
- Tight upper bound on delay
- Used for uncompressed audio and video
 - Video conferencing
 - Interactive audio
 - A/V distribution and retrieval

c). real time-Variable Bit Rate

- Used for Time sensitive applications
 - Requiring tightly constrained delay and delay variation
- rt-VBR applications transmit at a rate that varies with time
- e.g. compressed video
 - Produces varying sized image frames
 - Original (uncompressed) frame rate is constant
 - So compressed data rate varies
- Transmission varies
- Delay cannot

d). Non-Real Time Services

- ❑ Used for apps with bursty traffic characteristics
- ❑ Do not have tight constraints on delay and delay variation
- ❑ Network thus has greater flexibility in handling such traffic flows

e). Non Real Time-Variable Bit Rate

- nrt-VBR
- For some apps, it may be able to characterize expected traffic flow
- Improve QoS in loss and delay
- End system specifies:
 - Peak cell rate
 - Sustainable or average rate
 - Measure of how bursty or clumped the traffic is
 - Network can use this info to provide low delay and minimal cell loss
- e.g. Airline reservations, banking transactions

f). Unspecified Bit Rate

- ❑ There may be additional capacity over and above that used by CBR and VBR traffic
 - Not all resources dedicated to CBR and VBR
 - Bursty nature of VBR means sometimes there's spare capacity
- ❑ Unused capacity can be made available for the UBR service
- ❑ UBR suitable for applications that can tolerate some cell loss or variable delays
 - e.g. TCP based traffic
- ❑ Cells forwarded on FIFO basis using capacity not consumed by other services
- ❑ Best efforts service – other services get priority

g). Available Bit Rate

- Improved version of UBR
- Application specifies peak cell rate (PCR) and minimum cell rate (MCR)
- Resources allocated to give at least MCR
- Spare capacity shared among all ABR sources
- Uses explicit feedback to sources to assure that capacity is shared fairly
 - i.e. if one station tries to hog the spare capacity, ATM detects it and orders it to slow down
- e.g. LAN interconnection
 - End systems attached to network are routers
 - ATM (in this e.g.) does not reach the desktop

5.8.2.2 ATM Layer: Virtual Circuits

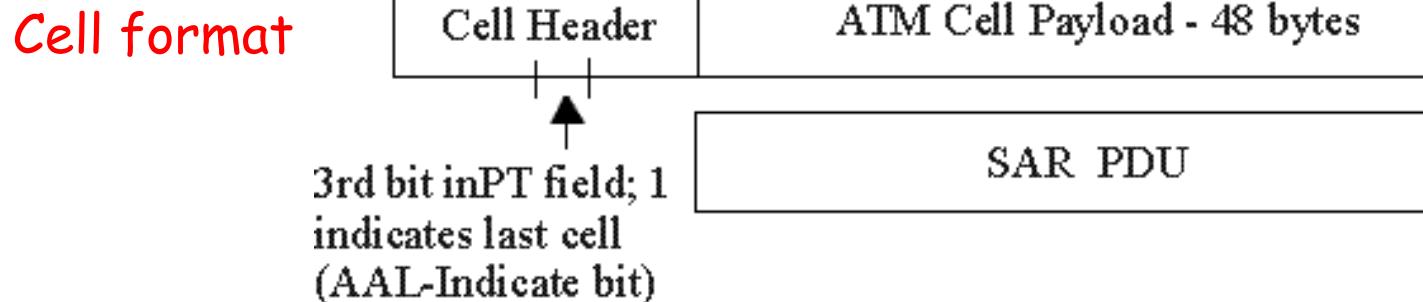
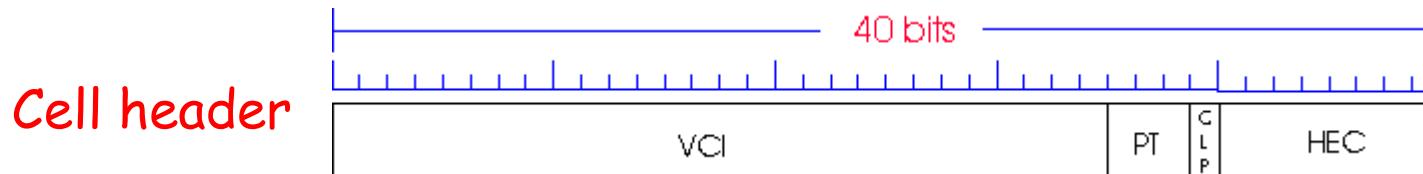
- **VC transport:** cells carried on VC from source to dest
 - Call setup, teardown for each call *before* data can flow
 - Each packet carries VC identifier (not destination ID)
 - *Every* switch on source-dest path maintains “state” for each passing connection
 - Link and switch resources (bandwidth, buffers) may be *allocated* to VC: to get circuit-like performance.
- **Permanent VCs (PVCs)**
 - Long lasting connections
 - Typically: “permanent” route between two IP routers
- **Switched VCs (SVC):**
 - Dynamically set up on per-call basis

ATM VCs

- Advantages of ATM VC approach:
 - QoS performance guarantee for connection mapped to VC (bandwidth, delay, delay jitter)
- Drawbacks of ATM VC approach:
 - Inefficient support of datagram traffic
 - One PVC between each source/dest pair does not scale (N^2 connections needed)
 - SVC introduces call setup latency, processing overhead for short lived connections

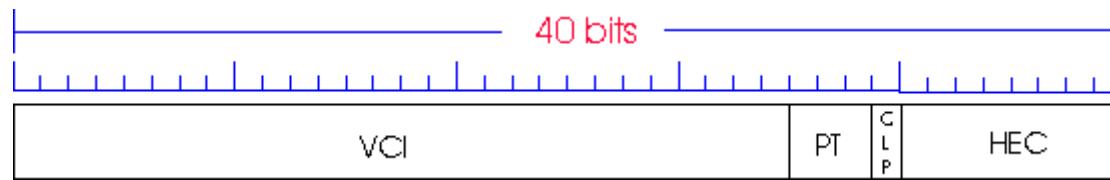
5.8.2.3 ATM Layer: ATM cell

- 5-byte ATM cell header
- 48-byte payload
 - Why?: small payload -> short cell-creation delay for digitized voice
 - halfway between 32 and 64 (compromise!)



ATM cell header

- **VCI:** virtual channel ID
 - will *change* from link to link thru net
- **PT:** Payload type (e.g. RM cell versus data cell)
- **CLP:** Cell Loss Priority bit
 - CLP = 1 implies low priority cell, can be discarded if congestion
- **HEC:** Header Error Checksum
 - Cyclic redundancy check



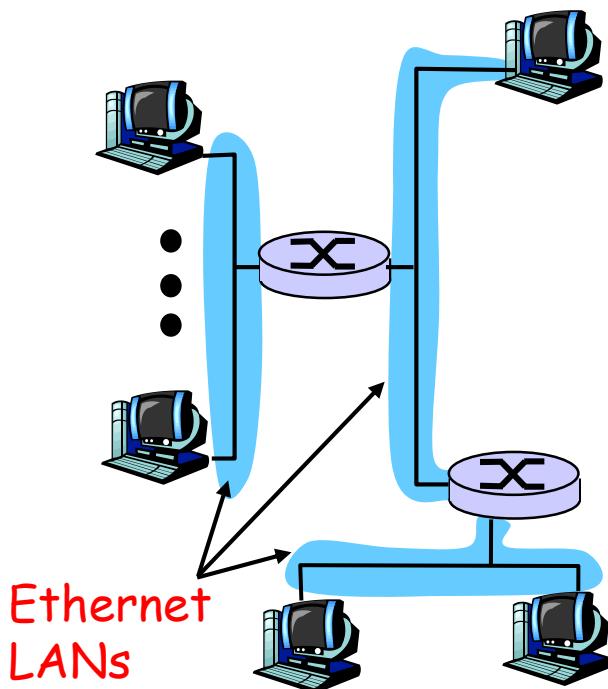
Link Layer

- 5.1 Introduction
 - 5.2 Link Layer Services
 - 5.3 Multiple access protocols
- LAN Technology
 - 5.4 LAN Addresses and ARP
 - 5.5 Ethernet
 - 5.6 Interconnecting LANs
 - 5.7 Examples of LAN Configurations
- The Whole Enchilada
 - 5.7 How the Web and Internet work
 - 5.8 ATM
 - 5.9 How IP and ATM co-exist

5.9 Tying it all up: IP-Over-ATM

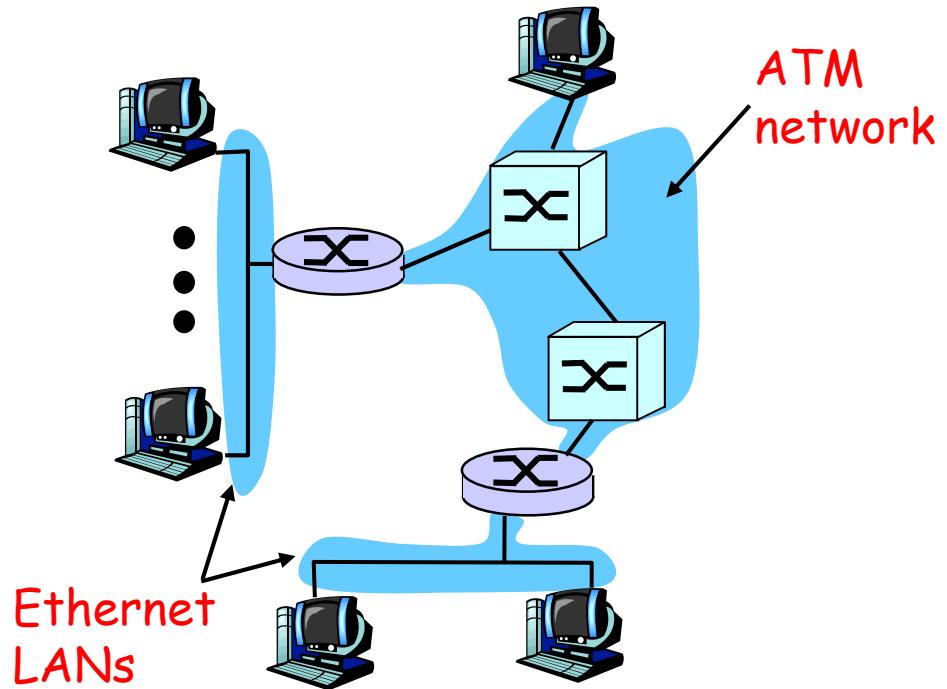
Classic IP only

- ❑ 3 “networks” (e.g., LAN segments)
- ❑ MAC (802.3) and IP addresses

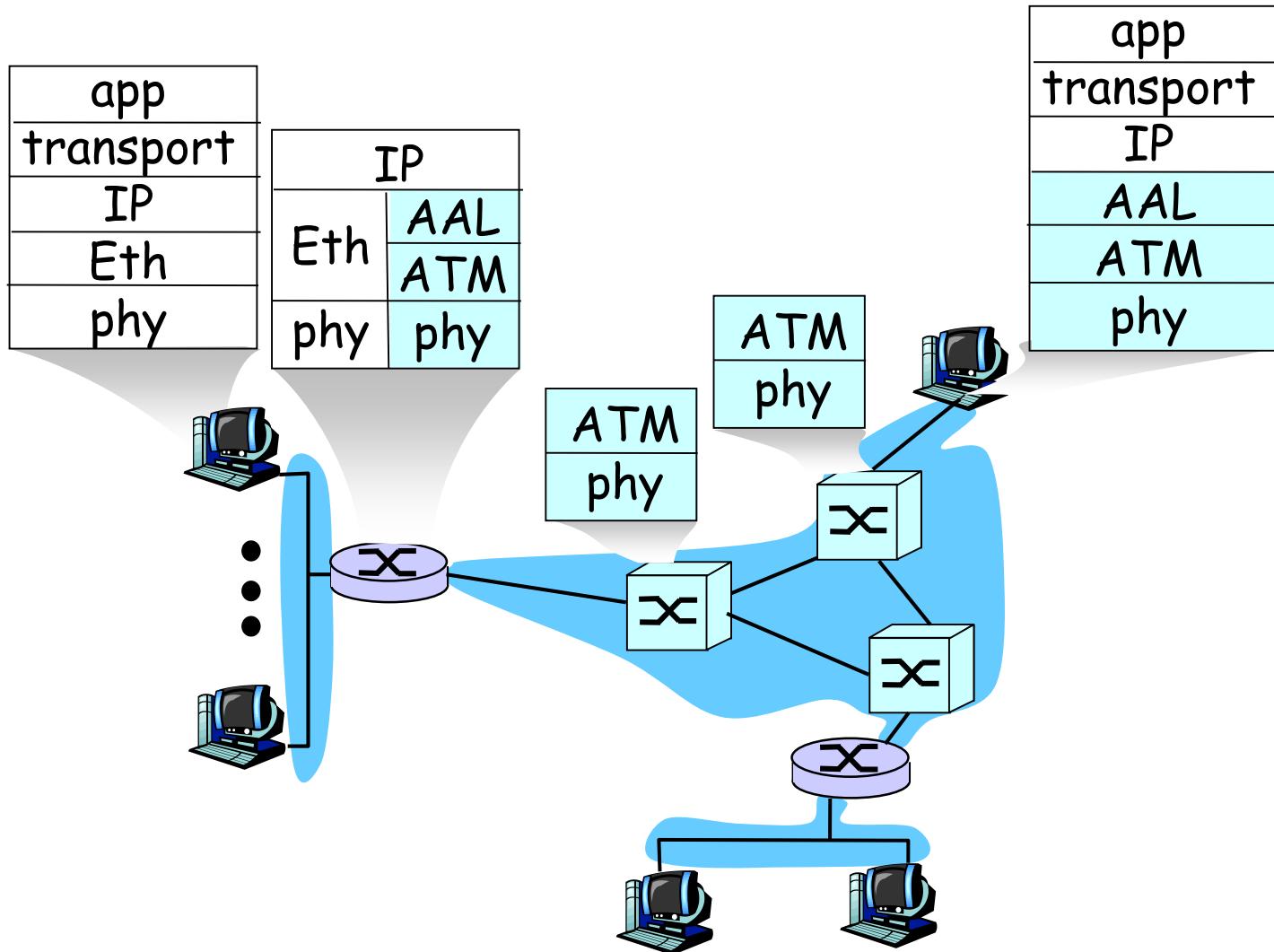


IP over ATM

- ❑ replace “network” (e.g., LAN segment) with ATM network
- ❑ ATM addresses, IP addresses



IP-Over-ATM



Datagram Journey in IP-over-ATM Network

- at Source Host:
 - IP layer maps between IP, ATM dest address (using ARP)
 - passes datagram to AAL5
 - AAL5 encapsulates data, segments cells, passes to ATM layer
- ATM network: moves cell along VC to destination
- at Destination Host:
 - AAL5 reassembles cells into original datagram
 - if CRC OK, datagram is passed to IP

Chapter 5: Summary

- Principles behind data link layer services:
 - Error detection, correction
 - Sharing a broadcast channel: multiple access
 - Link layer addressing
- Instantiation and implementation of various link layer technologies
 - Ethernet
 - Switched LANS
- The Whole Enchilada
 - A day in the life of a Web request
 - ATM – a very different network
 - IP over ATM