Master Course
Computer Networks
IN2097

Prof. Dr.-Ing. Georg Carle
Christian Grothoff, Ph.D.

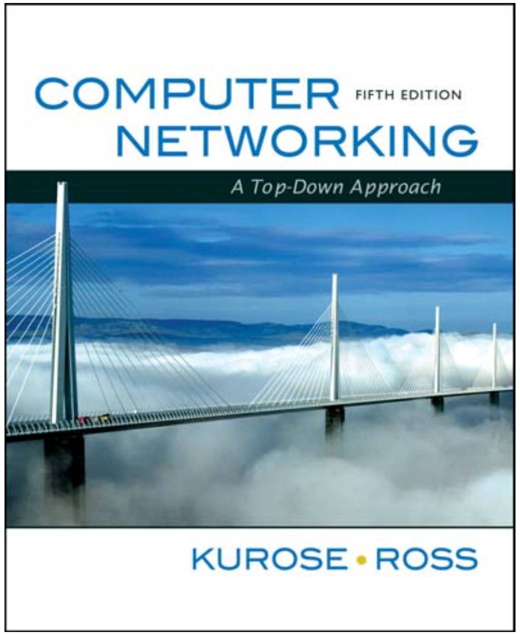
Chair for Network Architectures and Services
Department of Computer Science
Technische Universität München
http://www.net.in.tum.de

Course Outline (tentative)

- Part 1: Internet protocols
  1. Overview on Computer Networks
  2. Application Layer
  3. Transport Layer
  4. Network Layer
  5. Link Layer
  6. Wireless and Mobile Networks
  7. Interactive Communication – Voice and Video Services
- Part 2: Advanced Computer Networks Principles
  8. Network Monitoring and Measurements
  9. Network design principles
     - *common themes:* signaling, indirection, virtualization, multiplexing, randomization, scalability
     - *implementation principles:* techniques
     - *network architecture:* the big picture, synthesis
     - *Future Internet* approaches
  10. Network Simulation (if time permits)

Acknowledgements

- *Part 1 of this lecture is based on the book*
  Computer Networking: A Top Down Approach ,
  5th edition.
  Jim Kurose, Keith Ross
  Addison-Wesley, April 2009.
- The lecture is based to a significant extent on slides by Jim Kurose and Keith Ross



Jim Kurose, University of Massachusetts, Amherst

Keith Ross
Polytechnic Institute of New York University

Course organization

- Lecture
  - Monday, 16:15-17.45, MI H2 first weekly, then typically bi-weekly
  - Friday, 10:15-11.45, MI H2 weekly
- Exercises
  - After start of exercises, typically bi-weekly Monday 16:15-17.45, MI 00.08.038
- Students are requested to subscribe to lecture and exercises at
  http://www.net.in.tum.de/en/teaching/ws0910/lectures/masterkurs-rechnernetze/
- Email list, svn access
  - for subscribers of course
- Questions and Answers / Office hours
  - Prof. Dr. Georg Carle, carle@net.in.tum.de
    - After the course and upon appointment (typically Thursday 11-12)
  - Christian Grothoff, Ph.D., grothoff@net.in.tum.de
- Course Material
  - Slides are available online. Slides may be updated during the course.

Grading

- Students are requested to subscribe to lecture and exercises at http://www.net.in.tum.de/en/teaching/ws0910/lectures/masterkurs-rechnernetze/
- Exercises
  - Successfully participating at exercises gives a bonus of 0,3 for overall grade
- Practical assignments
  - Two practical assignments are planned
  - You have to succeed in at least one
  - They will be graded
- Our concept for grading
  - Final examinations will be oral and give an individual grade. You must pass the oral exam for being successful in the course.
  - For overall grade, grade of one practical assignment gives 25% of final grade
  - If your grade for a second practical assignment is better than your oral grade, it is accounted for by another 25%

Questions

- Who studies what?
  - Diploma degree?
  - Master in Informatics? Master in Information Systems [Wirtschaftsinformatik]?
  - Other Master courses?
  - Bachelor in Informatics?
- Which previous relevant courses?
  - IN0010 - Grundlagen Rechnernetze und Verteilte Systeme?
  - What else?

Overview:

- what's the Internet?
- what's a protocol?
- network edge; hosts, access net, physical media
- network core: packet/circuit switching, Internet structure
- performance: loss, delay, throughput
- protocol layers, service models

Our goal:

- get "feel" and terminology
- more depth, detail *later* in course
- approach:
  - use Internet as example

Chapter 1: roadmap

1.1 What is the Internet?

1.2 Network edge

end systems, access networks, links

1.3 Network core

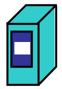
circuit switching, packet switching, network structure

1.4 Delay, loss and throughput in packet-switched networks

1.5 Protocol layers, service models

What's the Internet: "nuts and bolts" view

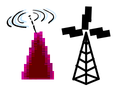

PC


server


wireless laptop


cellular handheld

- millions of connected computing devices: *hosts = end systems*
  - running *network apps*

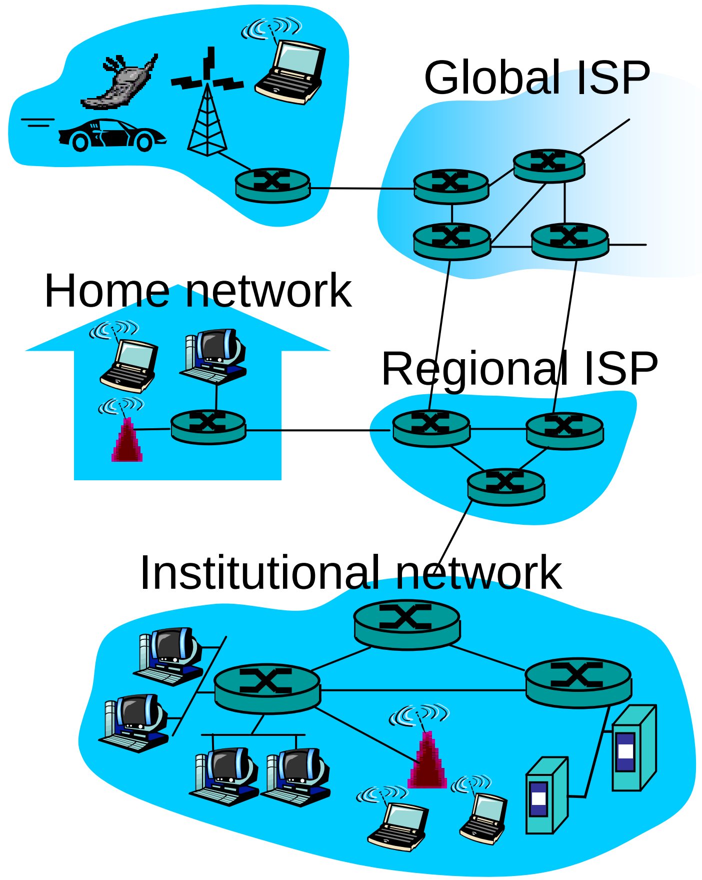

access points

wired links

- *communication links*
  - fiber, copper, radio, satellite
  - transmission rate = *bandwidth*

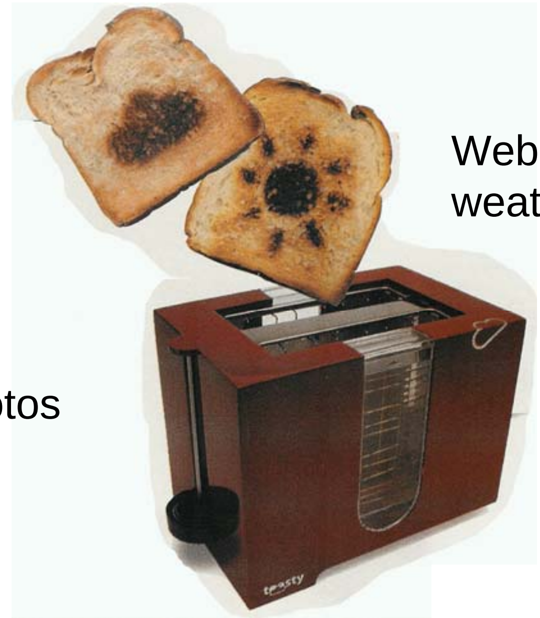

router

- *routers:* forward packets (chunks of data)

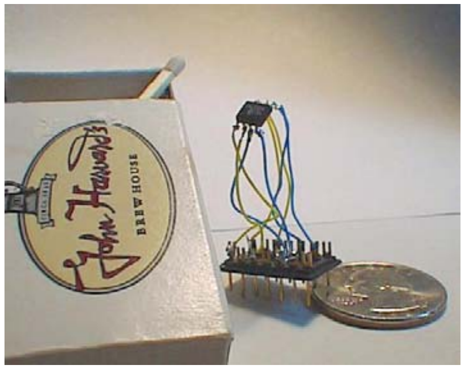


“Cool” internet appliances

IP picture frame
http://www.ceiva.com/
Free invitations for guests to send photos



Web-enabled toaster +
weather forecaster



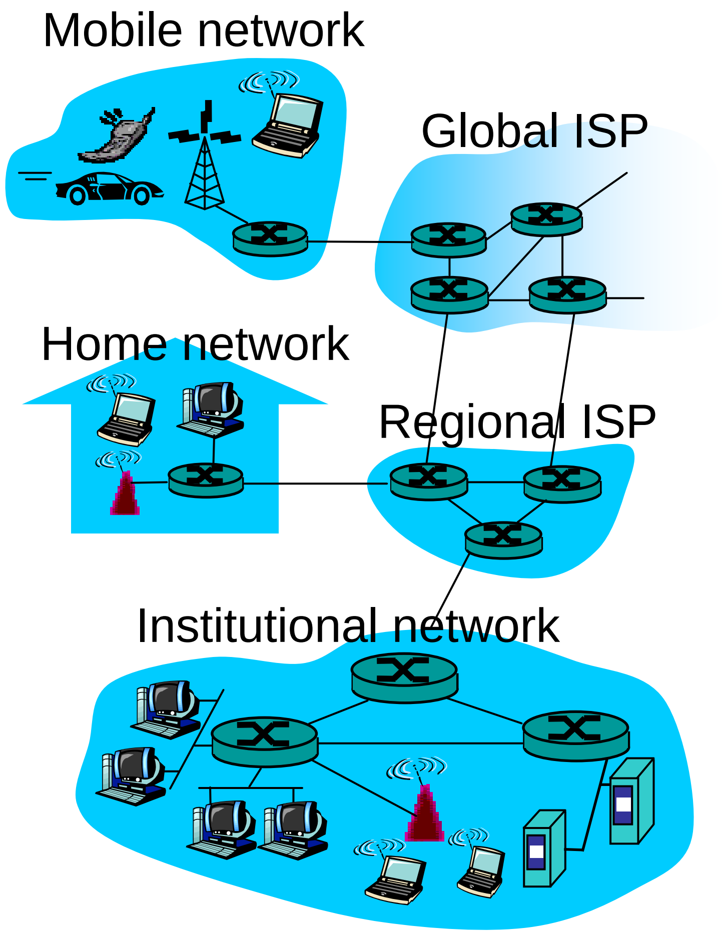
World’s smallest web server
in 1999

Internet phones

⇨ Who knows other cool internet appliances?

What's the Internet: "nuts and bolts" view

- *protocols* control 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
- communication *infrastructure* enables distributed applications:
  - Web, VoIP, email, games, e-commerce, file sharing
- communication services provided to applications:
  - reliable data delivery from source to destination
  - "best effort" (unreliable) data delivery



What's a protocol?

human protocols:

- "what's the time?"
- "I have a question"
- introductions

… specific msgs sent

… specific actions taken when messages received, or other events

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 message transmission, receipt

Chapter 1: roadmap

1.1 What *is* the Internet?

1.2 Network edge

- end systems, access networks, links

1.3 Network core

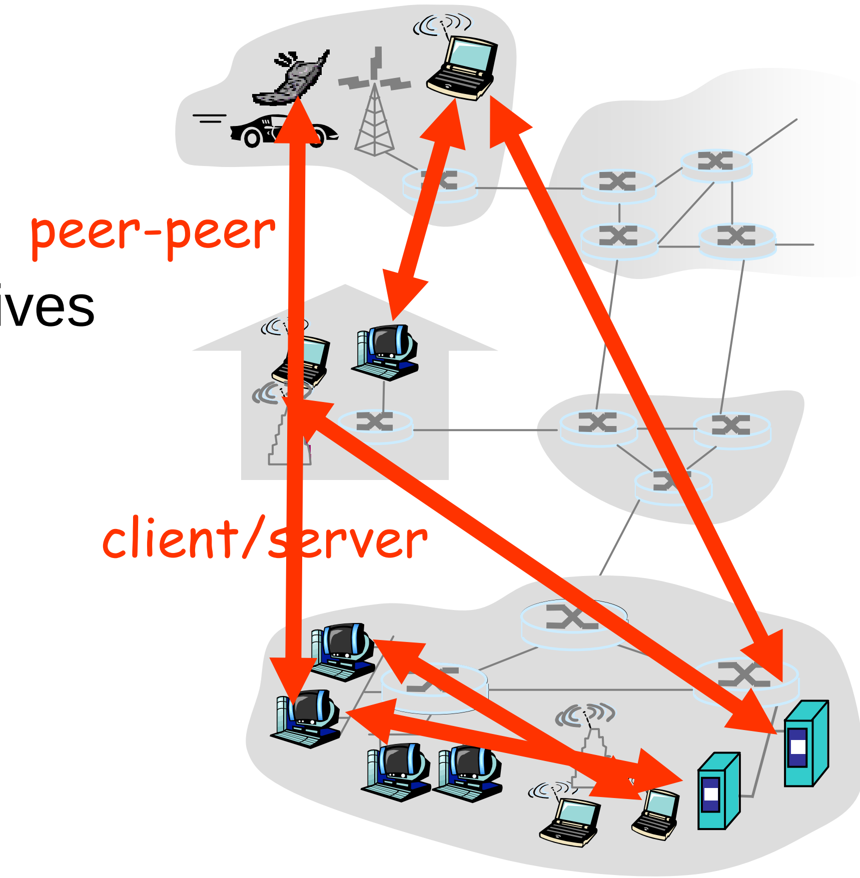
- circuit switching, packet switching, network structure

1.4 Delay, loss and throughput in packet-switched networks

1.5 Protocol layers, service models

The network edge:

- end systems (hosts):
  - run application programs
  - e.g. Web, email
  - at “edge of network”
- client/server model
  - client host requests, receives service from always-on server
  - e.g. Web browser/server; email client/server
- peer-peer model:
  - minimal (or no) use of dedicated servers
  - e.g. Skype, BitTorrent



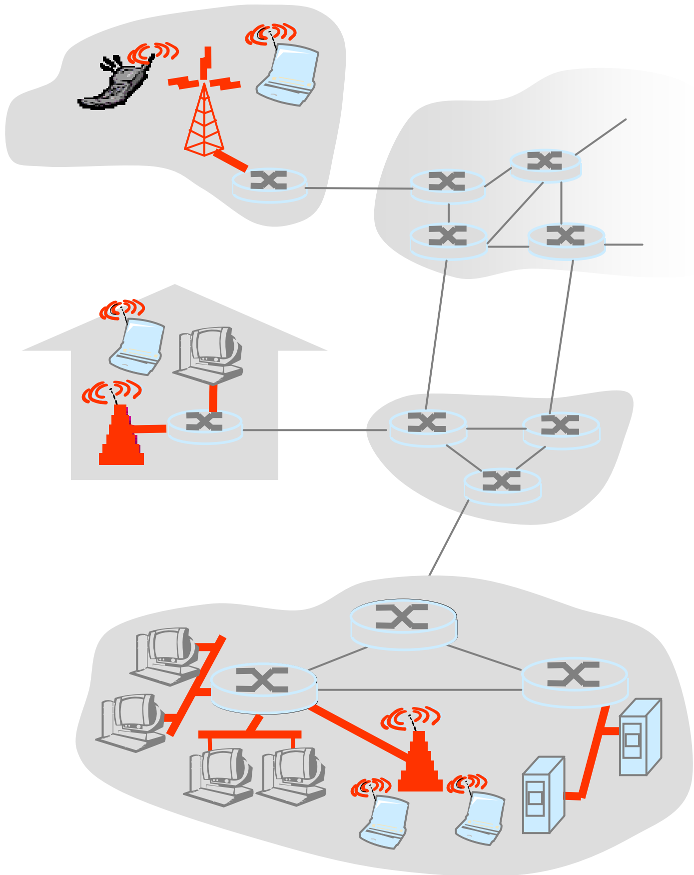
Access networks and physical media

Q: How to connect end systems to edge router?

- residential access networks
- institutional access networks (school, company)
- mobile access networks

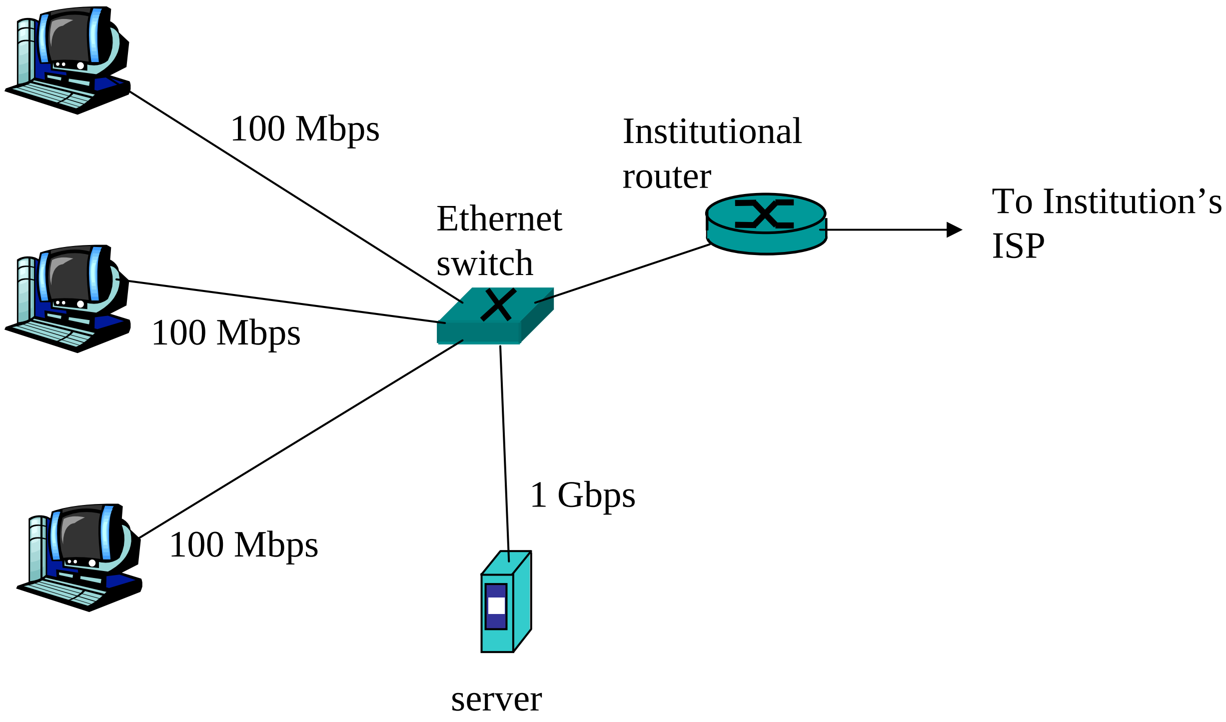
Relevant:

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



Ethernet Internet access

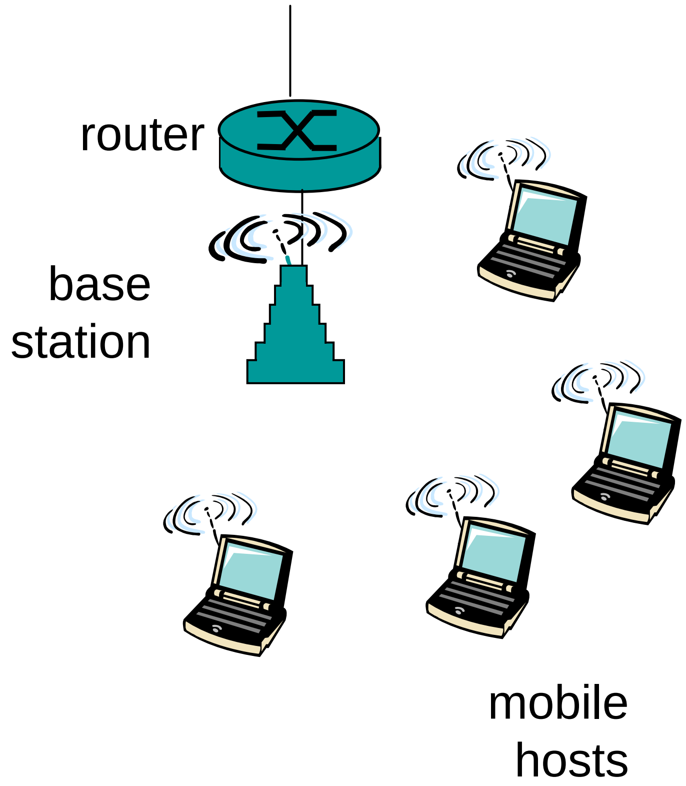
- Typically used in companies, universities, etc
  - 10 Mbs, 100Mbps, 1Gbps, 10Gbps Ethernet
  - Today, end systems typically connect into Ethernet switch



⇨ why?

Wireless access networks

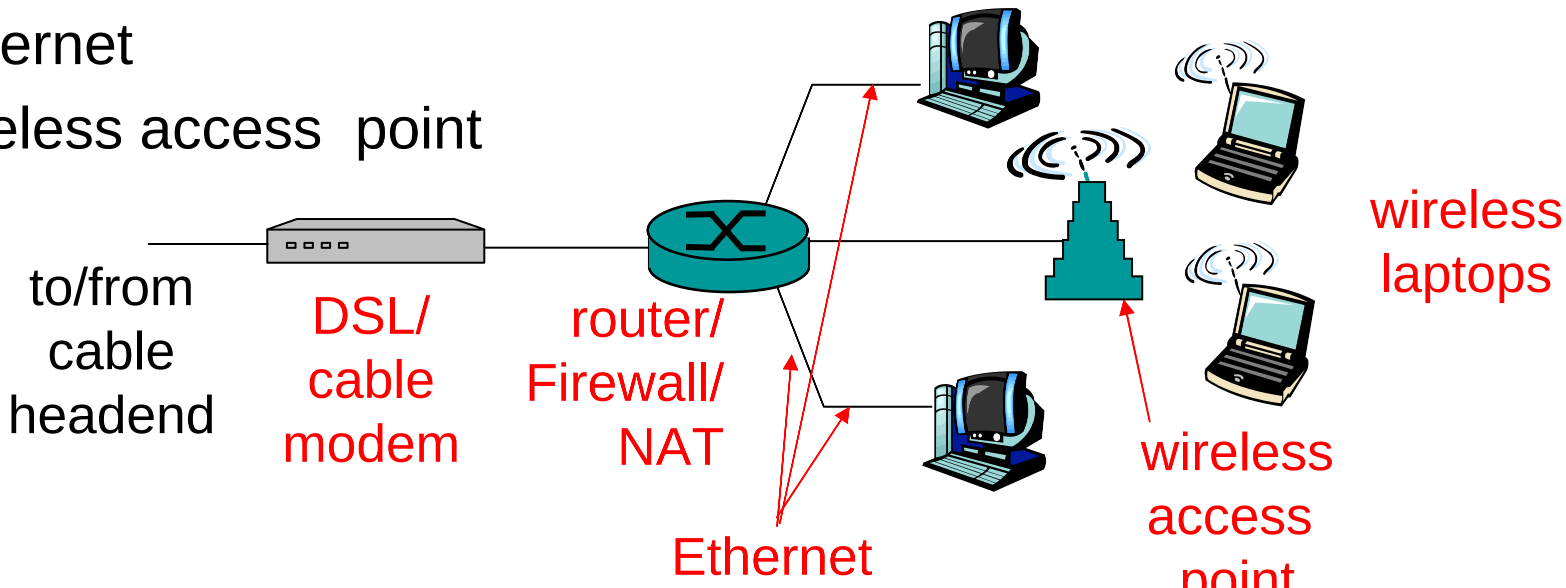
- shared *wireless* access network connects end system to router
  - via base station aka "access point"
- wireless LANs:
  - 802.11b/g (WiFi): 11 or 54 Mbps
- wider-area wireless access
  - provided by telco operator
  - ~1Mbps over cellular system (HSDPA)
  - next cellular network technology: LTE (10's Mbps) over wide area



Home networks

Typical home network components:

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



⇨ Our research project AutHoNe: targetting many innovations

Chapter 1: roadmap

1.1 What *is* the Internet?

1.2 Network edge

- end systems, access networks, links

1.3 Network core

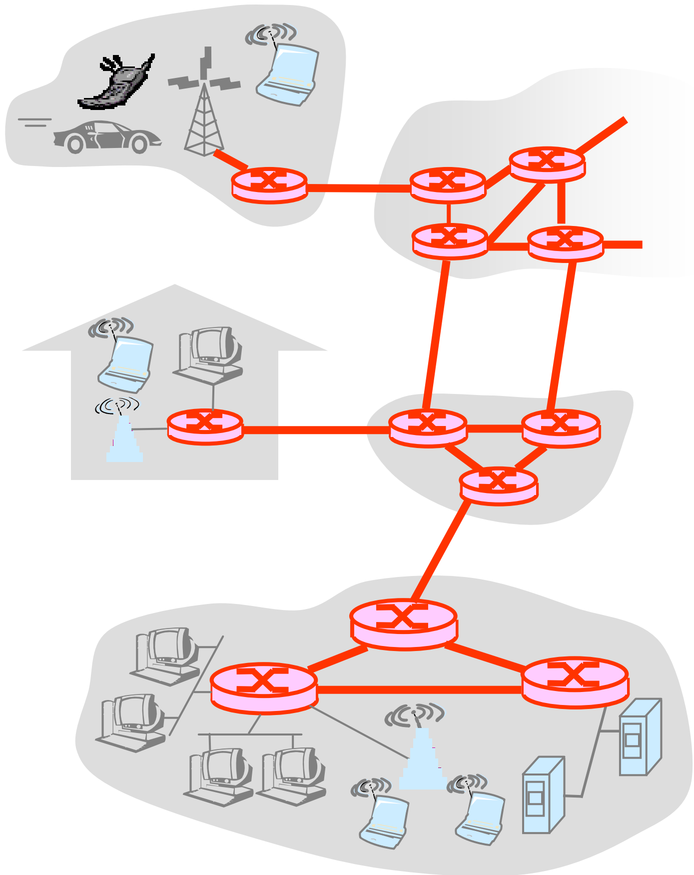
- circuit switching, packet switching, network structure

1.4 Delay, loss and throughput in packet-switched networks

1.5 Protocol layers, service models

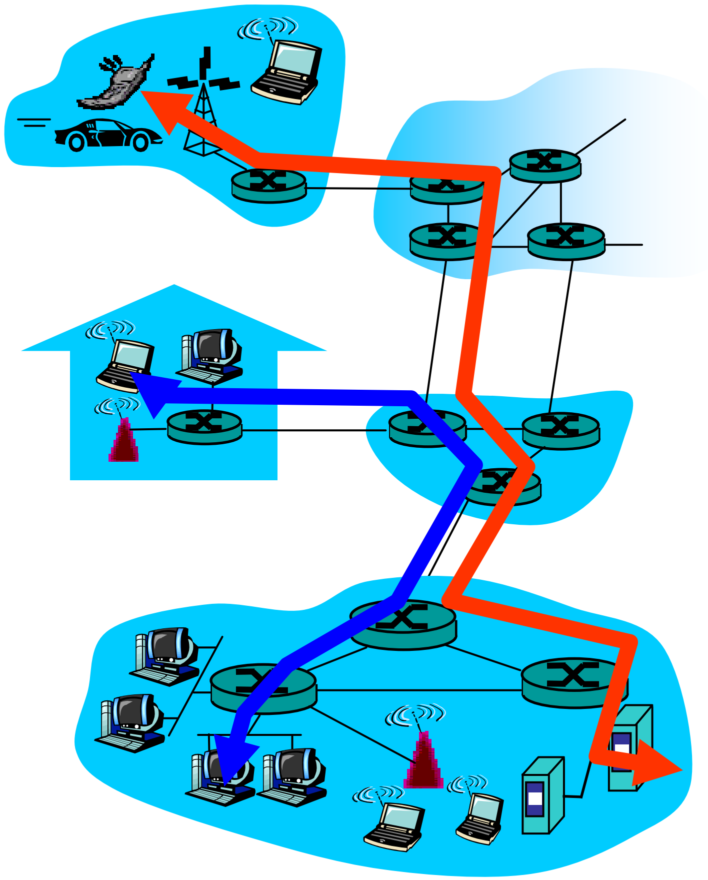
The Network Core

- mesh of interconnected routers
- *the* fundamental question: how is data transferred through net?
  - circuit switching: dedicated circuit per call: telephone net
  - packet-switching: data sent thru net in discrete “chunks”



Network Core: Circuit Switching

- End-end resources reserved for "call"
  - link bandwidth, switch capacity
  - dedicated resources: no sharing
  - circuit-like (guaranteed) performance
  - call setup required
- network resources (e.g., bandwidth) divided into "pieces"
  - pieces allocated to calls
  - resource piece *idle* if not used by owning call *(no sharing)*
- dividing link bandwidth into "pieces"
  - frequency division
  - time division
- Inefficient for bursty sources (⇨why?)
- Quality guarantee, but call blocking



Network Core: 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*

resource contention:

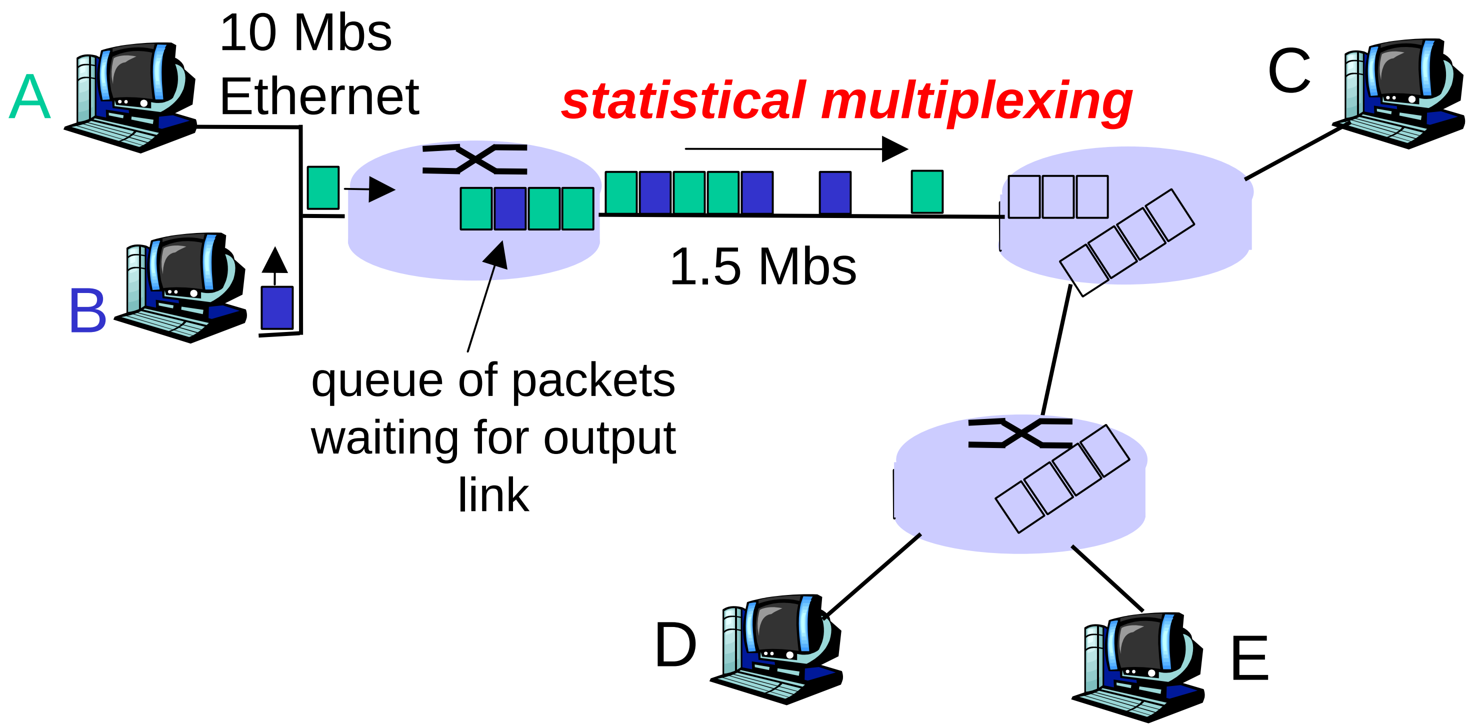
- aggregate resource demand can exceed amount available
- congestion: packets queue, wait for link use
- store and forward: packets move one hop at a time
  - Node receives complete packet before forwarding

Bandwidth division into “pieces”

Dedicated allocation

Resource reservation

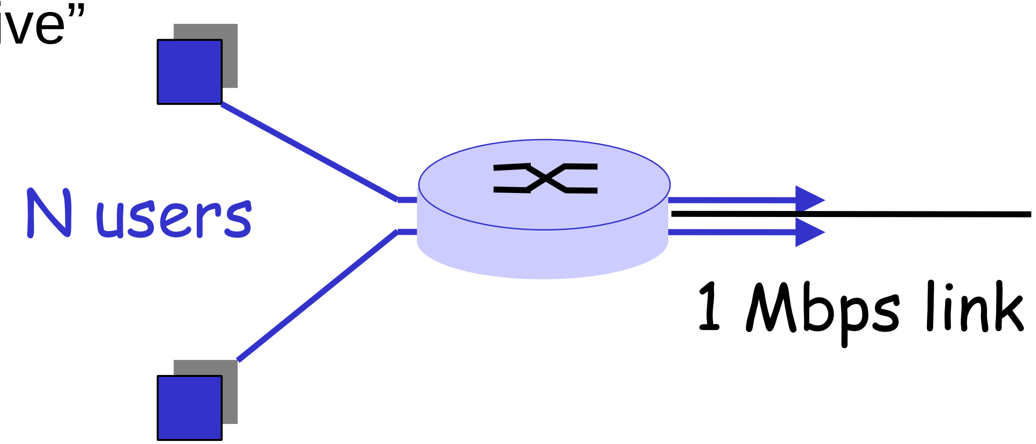
Packet Switching: Statistical Multiplexing



- Sequence of A & B packets does not have fixed pattern ➔ ***statistical multiplexing***.
- In TDM each host gets same slot in revolving TDM frame.

Packet switching versus circuit switching

- For bursty sources, Packet switching allows more users to use network! Example:
  - 1 Mbit link
  - each user:
    - 100 kbps when “active”
    - active 10% of time
  - circuit-switching:
    - 10 users
  - packet switching:
    - with 35 users, probability > 10 active less than .0004



Packet switching versus circuit switching

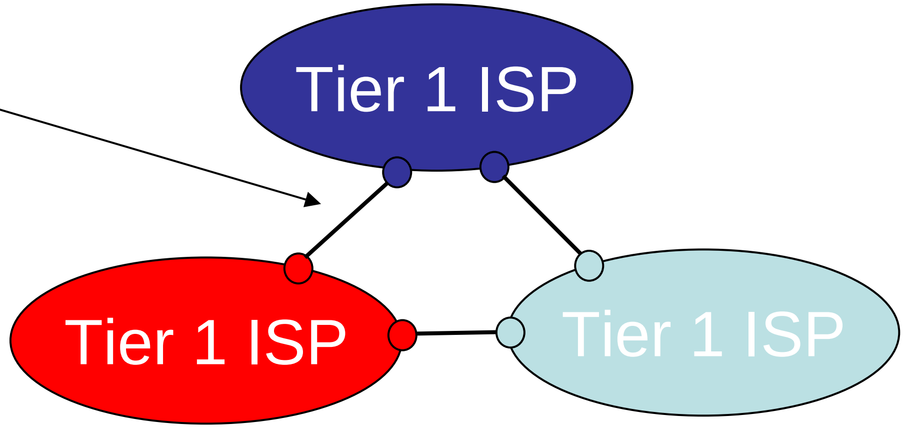
Is packet switching obviously better than circuit switching?

- packet switching is great for bursty data
  - resource sharing
  - simpler, no call setup
- possibility of excessive congestion: packet delay and loss
  - protocols needed for reliable data transfer, congestion control
- Q: How to provide circuit-like behavior?
  - bandwidth guarantees needed for audio/video apps
  - Internet-wide still an unsolved problem (⇨later)

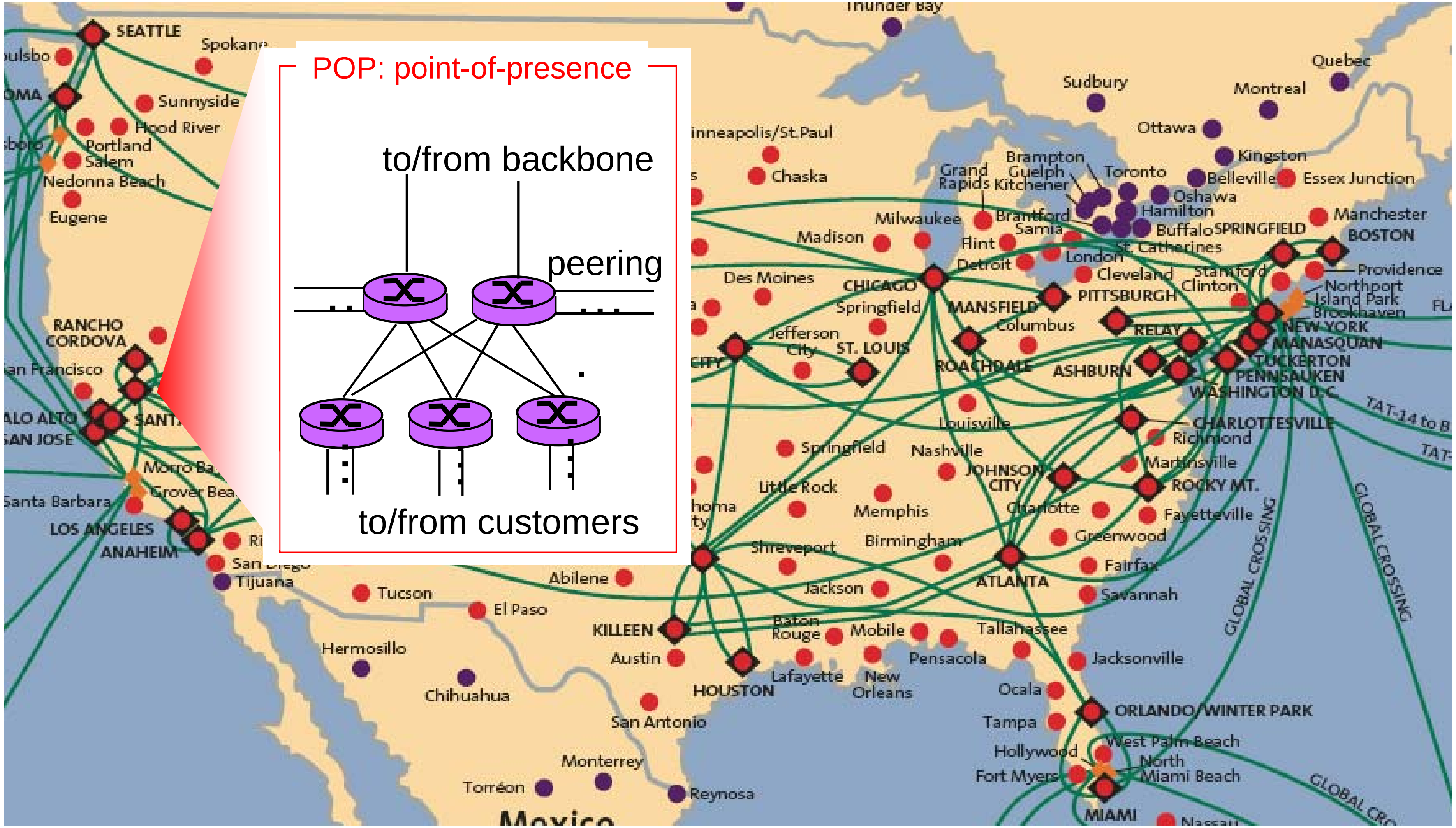
Internet structure: network of networks

- roughly hierarchical
- at center: “tier-1” ISPs (AT&T, Global Crossing, Level 3, NTT, Qwest, Sprint, Tata, Verizon (UUNET), Savvis, TeliaSonera), national/international coverage
  - treat each other as equals
  - can reach every other network on the Internet without purchasing IP transit or paying settlements.

Tier-1 providers interconnect (peer) privately



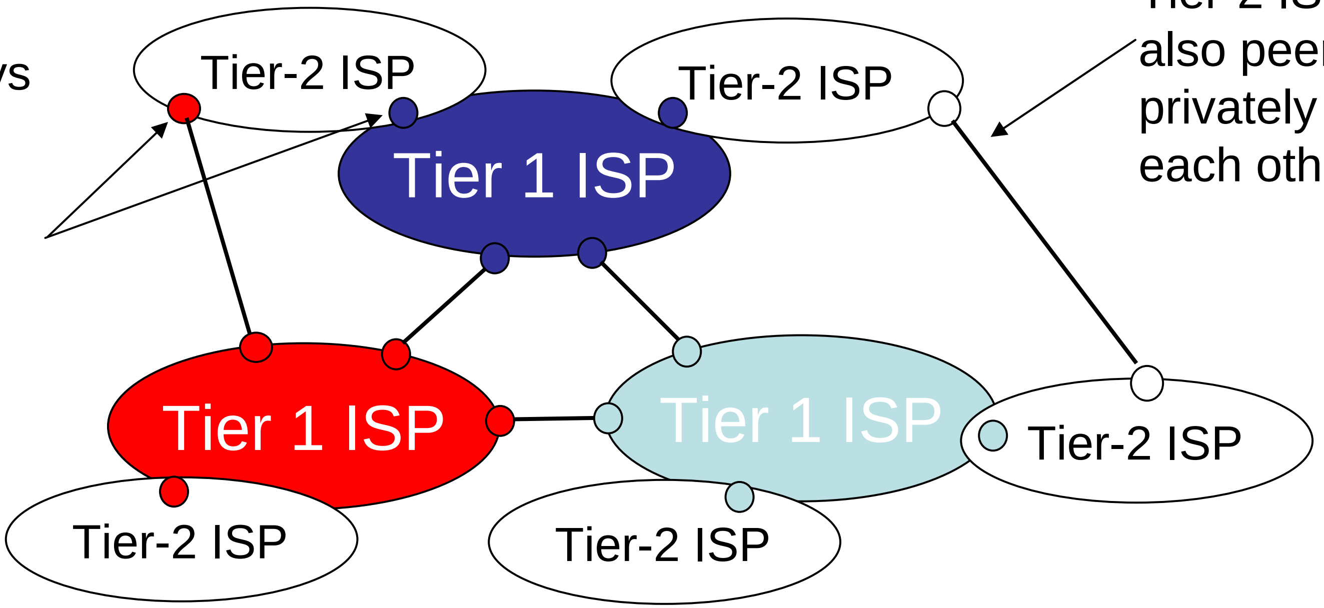
Tier-1 ISP: e.g., Sprint



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

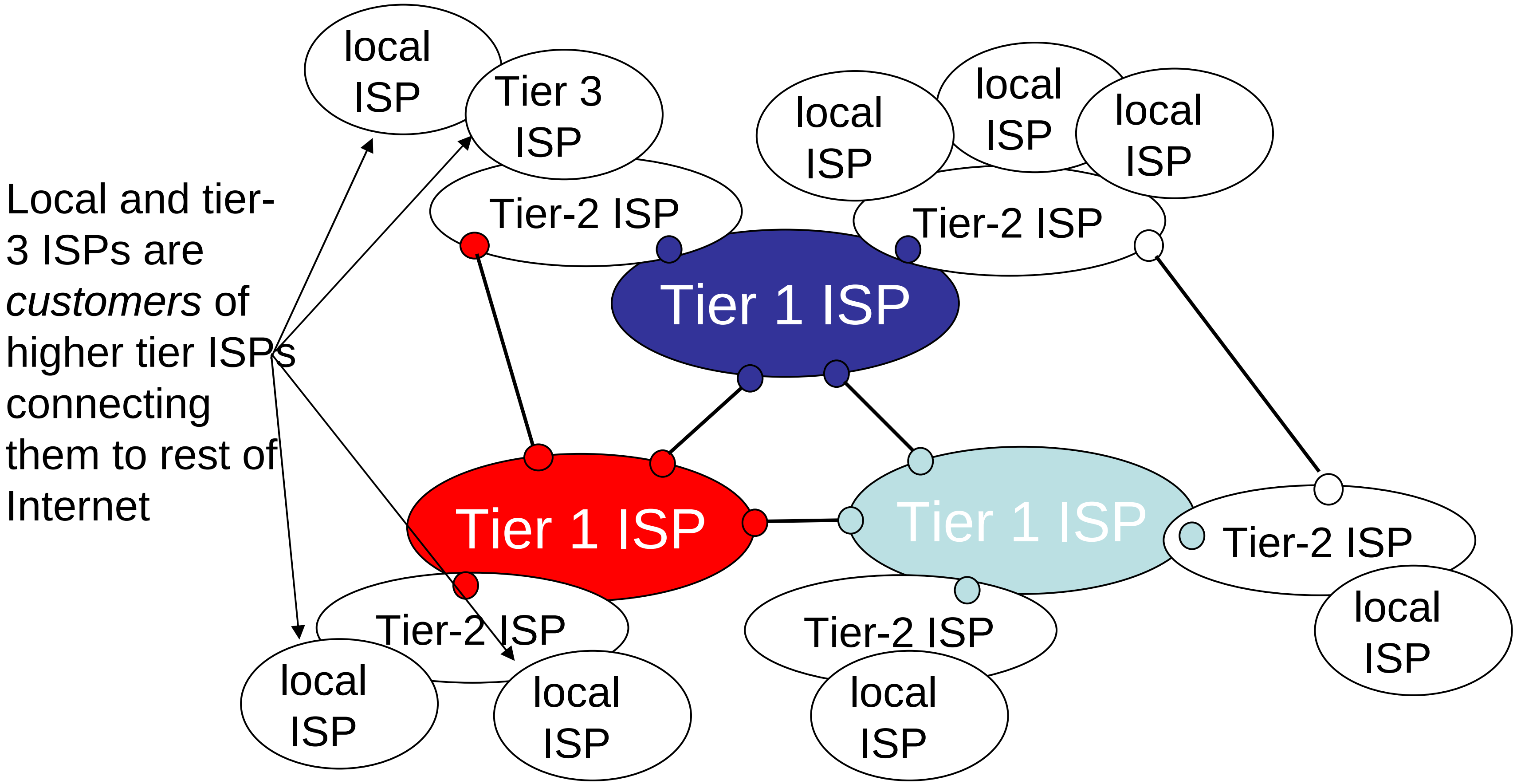
- Tier-2 ISP pays tier-1 ISP for connectivity to rest of Internet
- tier-2 ISP is *customer* of tier-1 provider



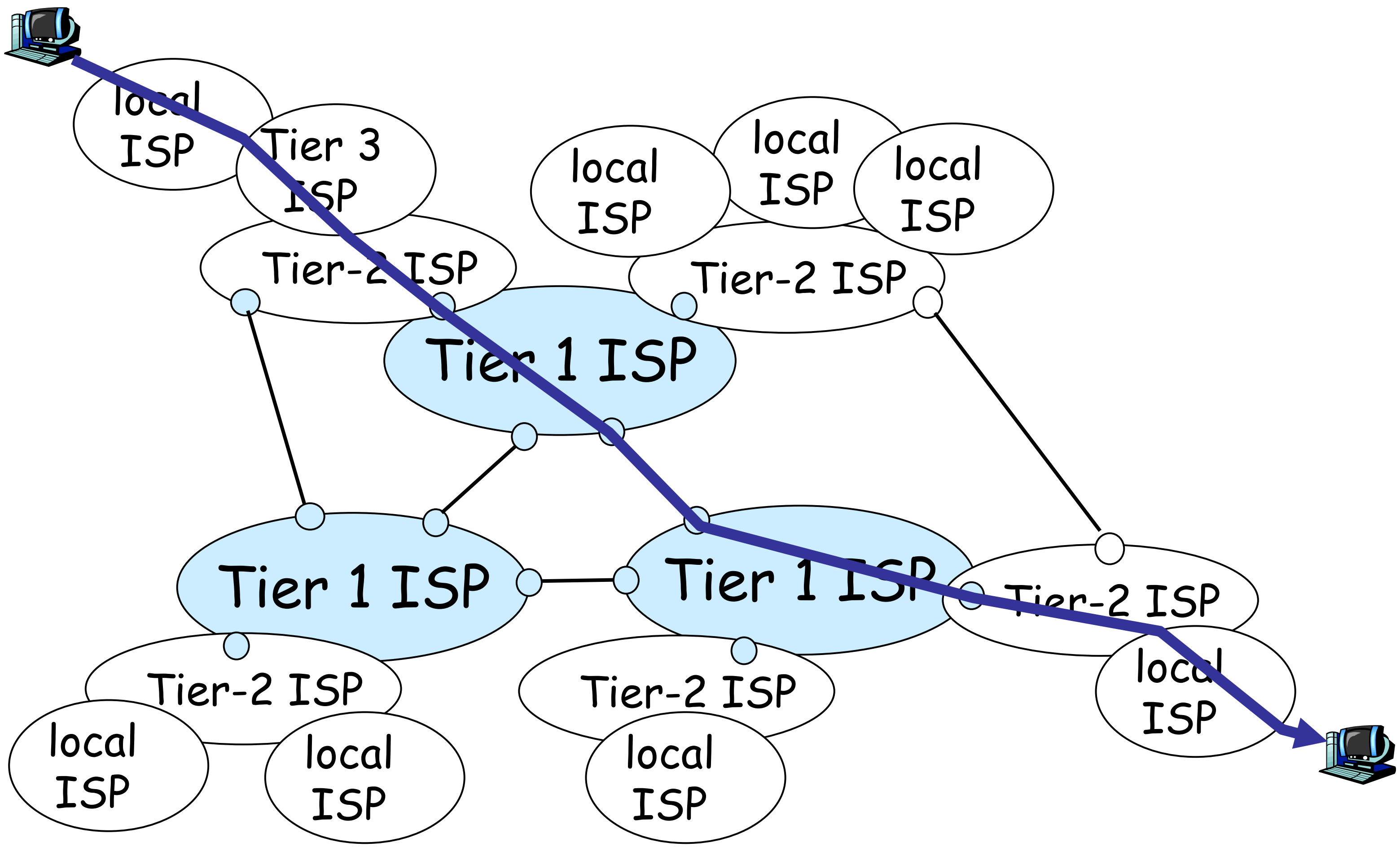
Tier-2 ISPs also peer privately with each other.

Internet structure: network of networks

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



- a packet passes through many networks!



Chapter 1: roadmap

1.1 What *is* the Internet?

1.2 Network edge

- end systems, access networks, links

1.3 Network core

- circuit switching, packet switching, network structure

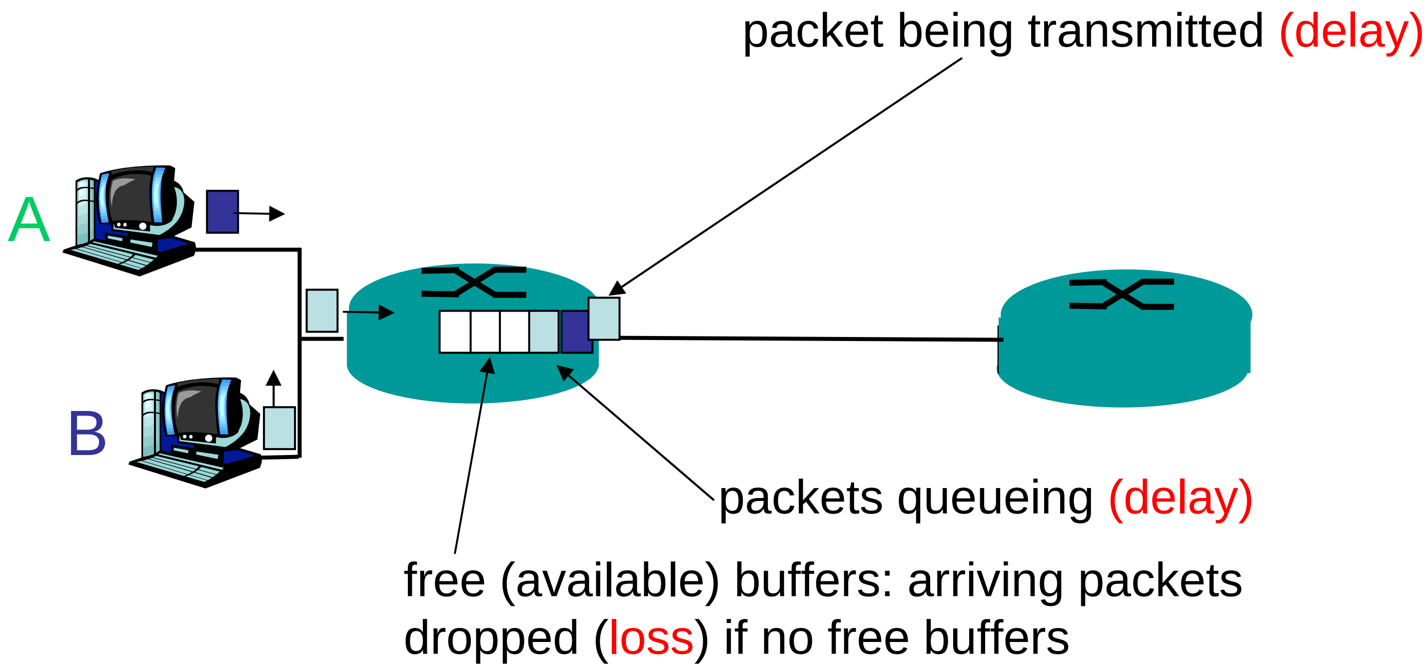
1.4 Delay, loss and throughput in packet-switched networks

1.5 Protocol layers, service models

How do loss and delay occur?

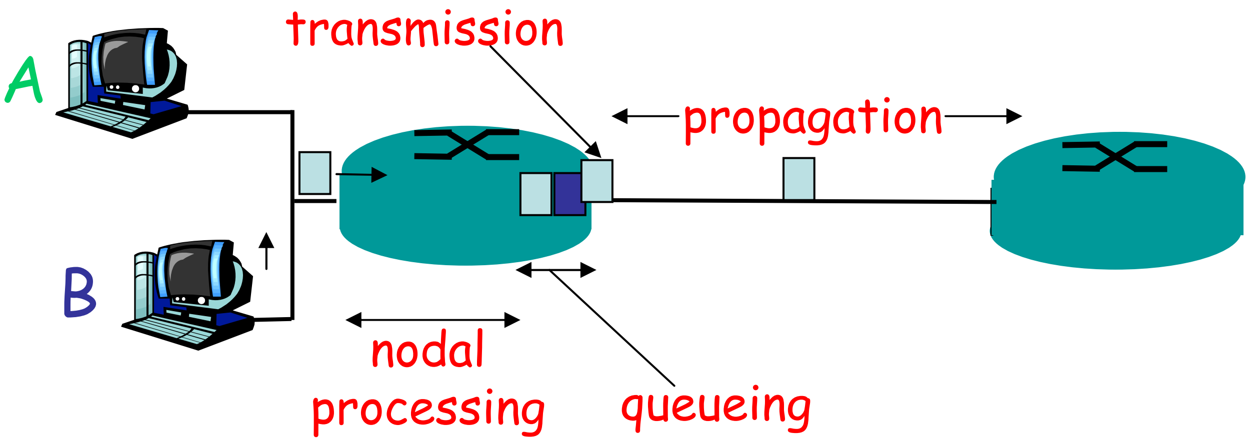
packets *queue* in router buffers

- packet arrival rate to link exceeds output link capacity
- packets queue, wait for turn



Four sources of packet delay

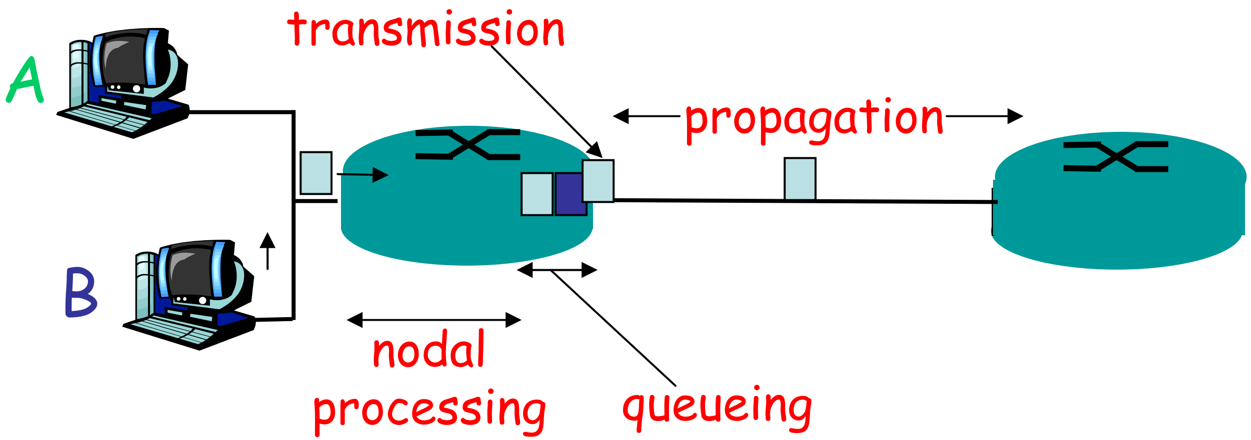
- 1. nodal processing:
  - check bit errors
  - determine output link
- 2. queueing
  - time waiting at output link for transmission
  - depends on congestion level of router



Delay in packet-switched networks

3. Transmission delay:
- R=link bandwidth (bps)
- L=packet length (bits)
- time to send bits into link = L/R

4. Propagation delay:
- d = length of physical link
- s = propagation speed in medium (~2x10^8 m/sec)
- propagation delay = d/s

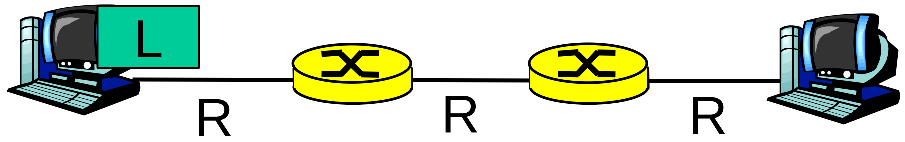


Nodal delay

- d_{proc} = processing delay
  - typically a few microsecs or less
- d_{queue} = queuing delay
  - depends on congestion
- d_{trans} = transmission delay
  - = L/R, significant for low-speed links
- d_{prop} = propagation delay
  - a few microsecs to hundreds of msecs

$$\boxed{d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}}$$

Packet-switching: store-and-forward



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

Example:

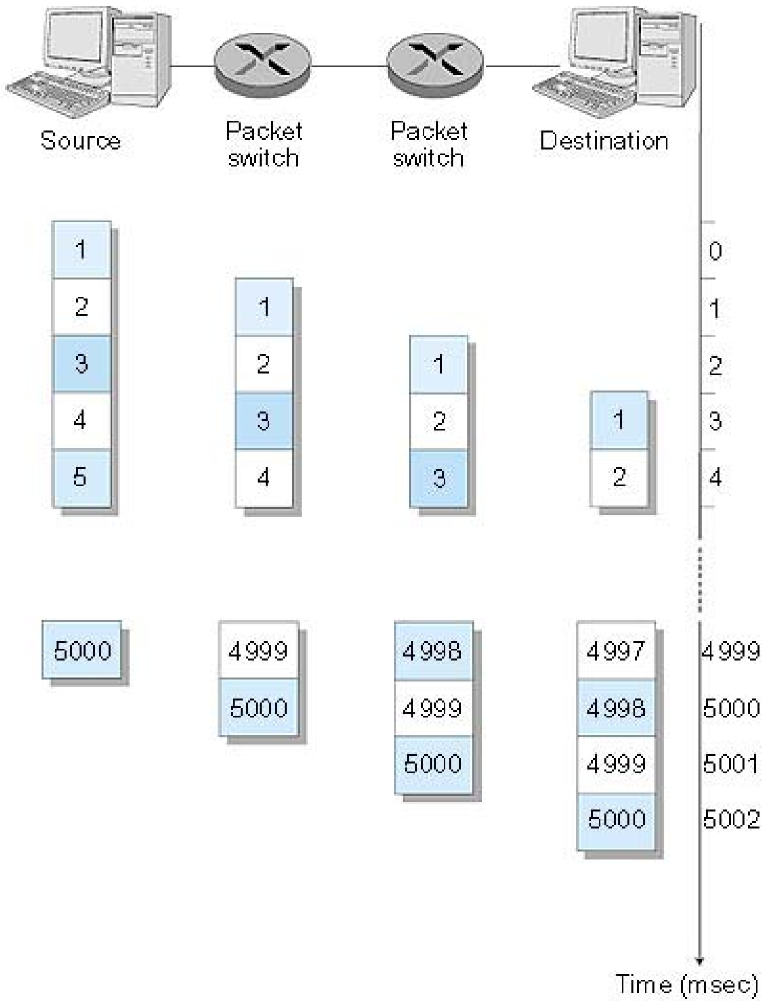
Circuit Switching:

- L = 7.5 Mbits
- R = 1.5 Mbps
- Transmission delay = 5 sec

Packet Switching:

- L = 7.5 Mbits
- R = 1.5 Mbps
- Transmission delay = 15 sec

Packet Switching: Message Segmenting



Now break up the message into 5000 packets

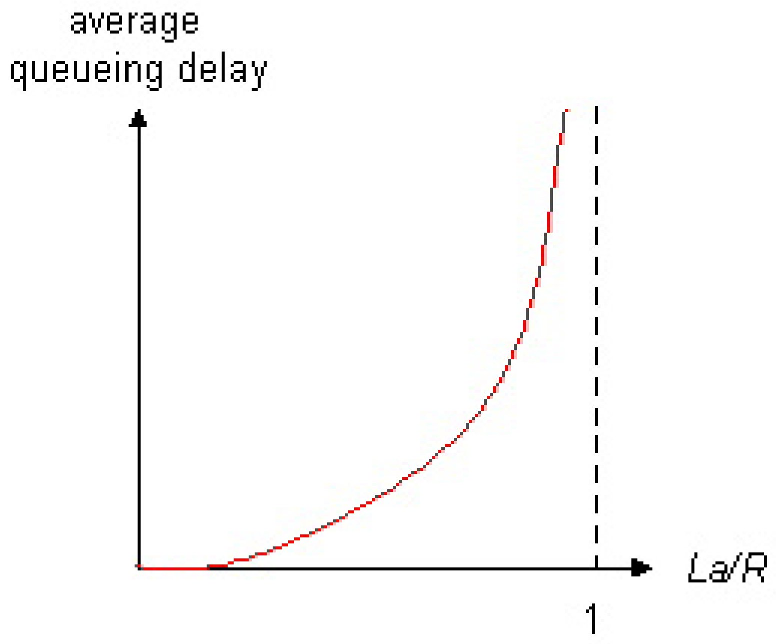
- Each packet 1,500 bits
- 1 msec to transmit packet on one link
- *pipelining:* each link works in parallel
- Delay reduced from 15 sec to 5.002 sec (as good as circuit switched)
- What did we achieve over circuit switching?
- Drawbacks (of packet vs. Message)

Queueing delay (revisited)

- R=link bandwidth (bit/s)
- L=packet length (bit)
- a=average packet arrival rate

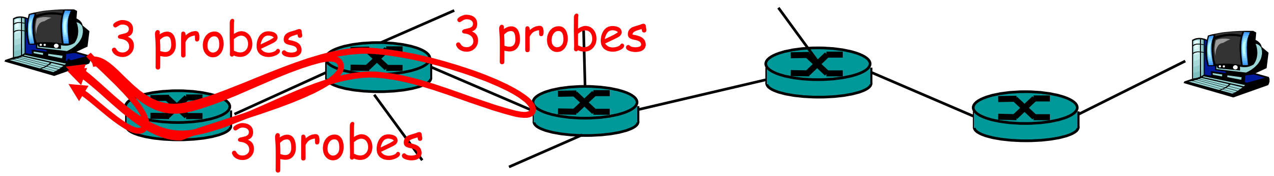
traffic intensity = L·a/R

- L·a/R ~ 0: average queueing delay small
- L·a/R -> 1: delays become large
- L·a/R > 1: more “work” arriving than can be serviced, average delay infinite!



“Real” Internet delays and routes

- What do “real” Internet delay & loss look like?
- **`Traceroute`** program: provides delay measurement from source to router along end-end Internet path towards destination. For all *i:*
  - sends three packets that will reach router *i* on path towards destination
  - router *i* will return packets to sender
  - sender times interval between transmission and reply.



“Real” Internet delays and routes

traceroute: gaia.cs.umass.edu to www.eurecom.fr

Three delay measurements from
gaia.cs.umass.edu to cs-gw.cs.umass.edu

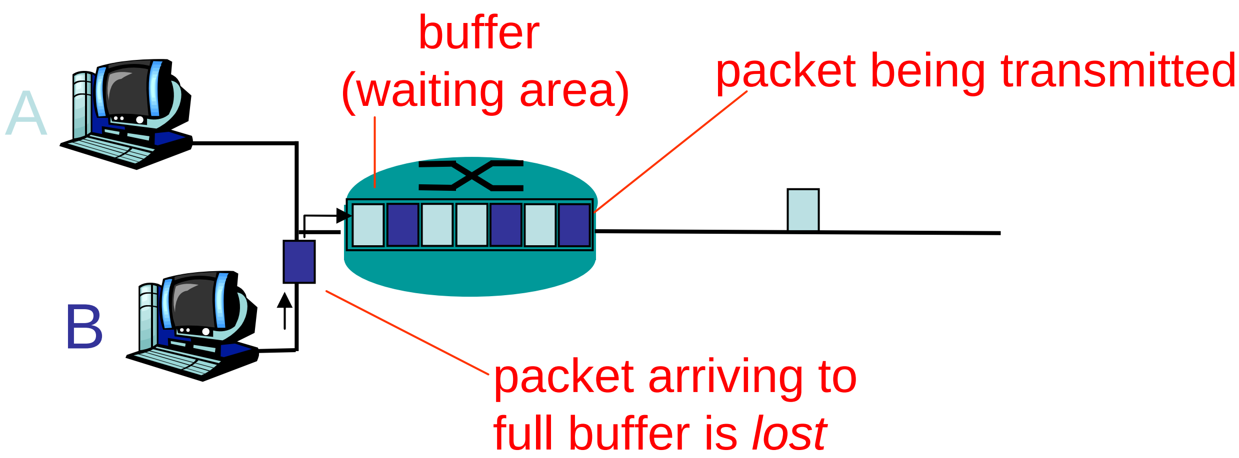
1 cs-gw (128.119.240.254) 1 ms 1 ms 2 ms
2 border1-rt-fa5-1-0.gw.umass.edu (128.119.3.145) 1 ms 1 ms 2 ms
3 cht-vbns.gw.umass.edu (128.119.3.130) 6 ms 5 ms 5 ms
4 jn1-at1-0-0-19.wor.vbns.net (204.147.132.129) 16 ms 11 ms 13 ms
5 jn1-so7-0-0-0.wae.vbns.net (204.147.136.136) 21 ms 18 ms 18 ms
6 abilene-vbns.abilene.ucaid.edu (198.32.11.9) 22 ms 18 ms 22 ms
7 nycm-wash.abilene.ucaid.edu (198.32.8.46) 22 ms 22 ms 22 ms
8 62.40.103.253 (62.40.103.253) 104 ms 109 ms 106 ms
9 de2-1.de1.de.geant.net (62.40.96.129) 109 ms 102 ms 104 ms
10 de.fr1.fr.geant.net (62.40.96.50) 113 ms 121 ms 114 ms
11 renater-gw.fr1.fr.geant.net (62.40.103.54) 112 ms 114 ms 112 ms
12 nio-n2.cssi.renater.fr (193.51.206.13) 111 ms 114 ms 116 ms
13 nice.cssi.renater.fr (195.220.98.102) 123 ms 125 ms 124 ms
14 r3t2-nice.cssi.renater.fr (195.220.98.110) 126 ms 126 ms 124 ms
15 eurecom-valbonne.r3t2.ft.net (193.48.50.54) 135 ms 128 ms 133 ms
16 194.214.211.25 (194.214.211.25) 126 ms 128 ms 126 ms
17 * * *
18 * * *
19 fantasia.eurecom.fr (193.55.113.142) 132 ms 128 ms 136 ms

trans-oceanic link

* means no response (probe lost, router not replying)

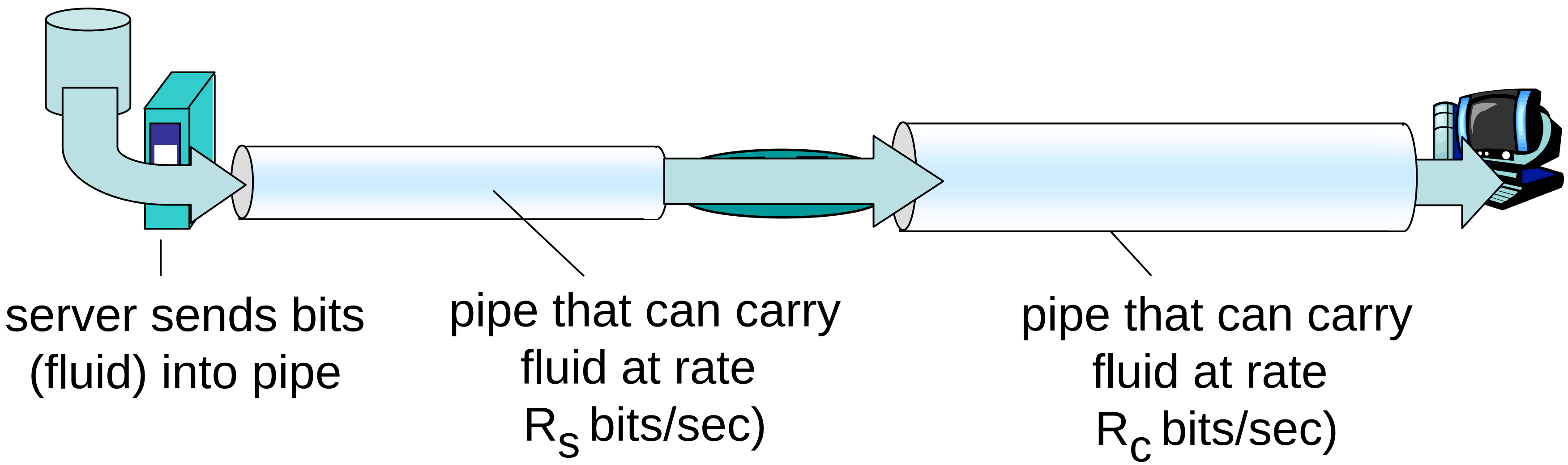
Packet loss

- queue (aka buffer) preceding link in buffer has finite capacity
- packet arriving to full queue dropped (aka lost)
- lost packet may be retransmitted by previous node, by source end system, or not at all



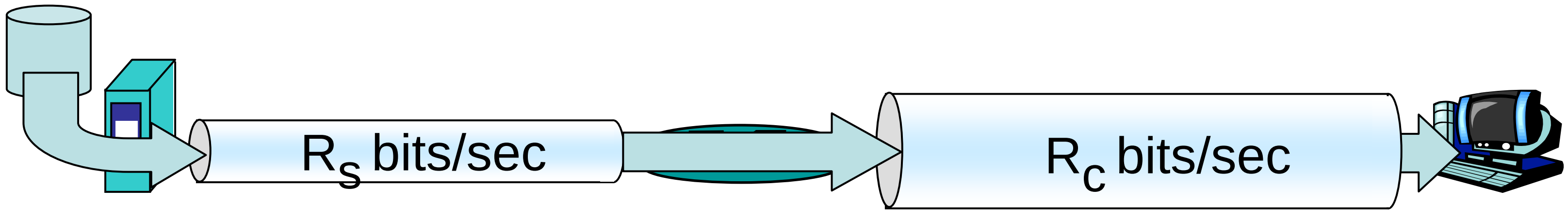
Throughput

- *throughput:* rate (bits/time unit) at which bits transferred between sender/receiver
  - *instantaneous:* rate at given point in time
  - *average:* rate over longer period of time

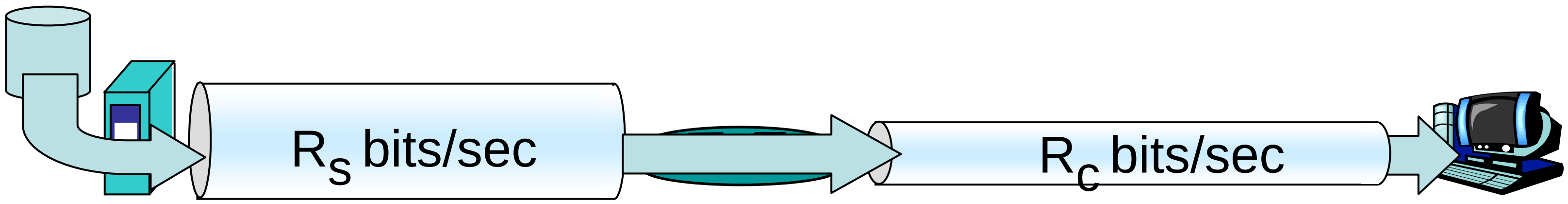


Throughput (more)

- $R_s < R_c$ What is average end-end throughput?



- $R_s > R_c$ What is average end-end throughput?



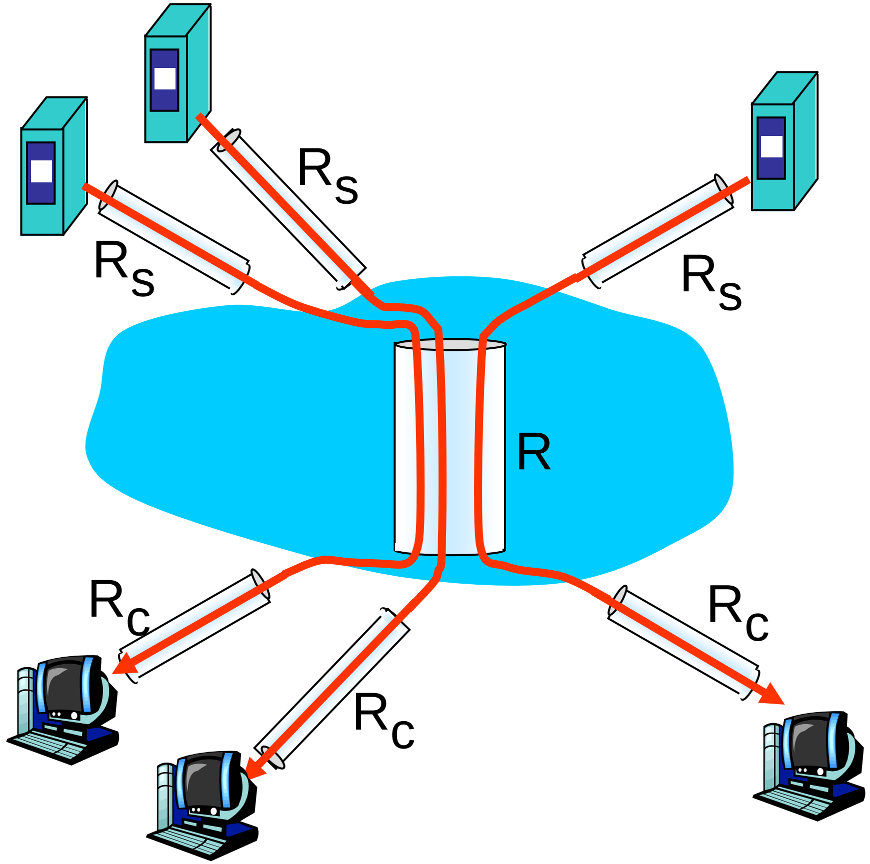
bottleneck link

link on end-end path that constrains end-end throughput

⇨ measurement challenge for networks with many nodes:
identify bottleneck interfaces, e.g. with packet-pair measurements

Throughput: Internet scenario

- Example: 10 clients / servers share a bottleneck link
  - per-connection end-end throughput: $\min(R_c, R_s, R/10)$
- in practice: R_c or R_s is often bottleneck



10 connections (fairly) share backbone bottleneck link R bits/sec

Chapter 1: roadmap

1.1 What *is* the Internet?

1.2 Network edge

- end systems, access networks, links

1.3 Network core

- circuit switching, packet switching, network structure

1.4 Delay, loss and throughput in packet-switched networks

1.5 Protocol layers, service models

1.6 Networks under attack: security

1.7 History

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?

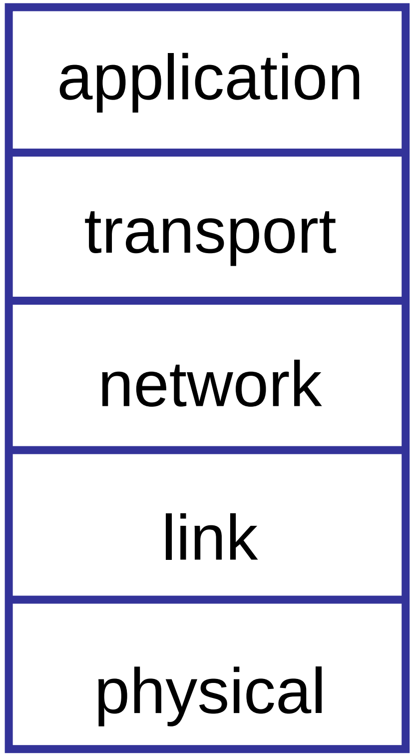
Why layering?

Dealing with complex systems:

- explicit structure allows identification, relationship of complex system's pieces
  - layered reference model for discussion
- modularization eases maintenance, updating of system
  - change of implementation of layer's service transparent to rest of system
  - e.g., change in gate procedure doesn't affect rest of system
- layering considered harmful?

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
  - PPP, Ethernet
- physical: bits “on the wire”



Introduction: Summary

Covered a lot of material!

- Internet overview
- what's a protocol?
- network edge, core, access network
  - packet-switching versus circuit-switching
  - Internet structure
- performance: loss, delay, throughput
- layering, service models

You now have:

- context, overview, "feel" of networking
- more depth, detail *to follow!*

Master Course
Computer Networks
IN2097

Prof. Dr.-Ing. Georg Carle
Christian Grothoff, Ph.D.

Chair for Network Architectures and Services
Institut für Informatik
Technische Universität München
http://www.net.in.tum.de

Chapter 2: Application layer

- Principles of network applications
- Web and HTTP
- DNS
- P2P applications
- Socket programming with TCP
- Socket programming with UDP

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

Some network applications

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

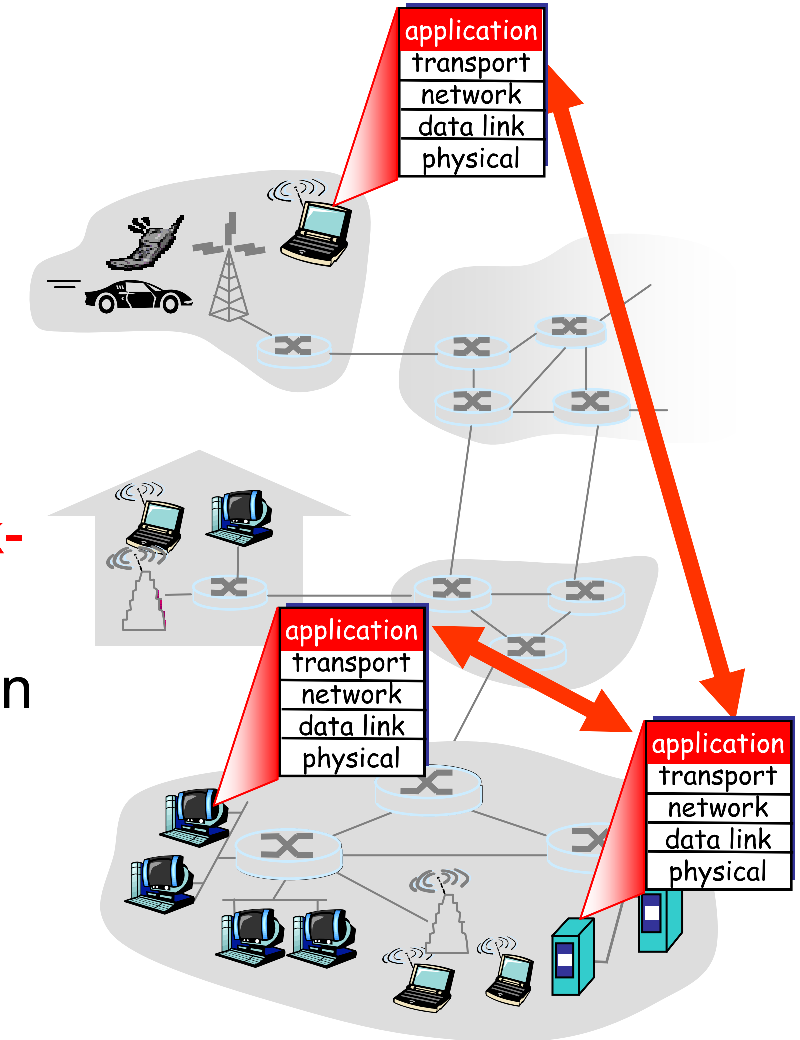
Creating a network application

write programs that

- run on (different) *end systems*
- communicate over 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 application development, propagation



⇨ think of different viewpoint:
what would be the benefits if you could program your router?

- **Principles of network applications**
- Web and HTTP
- DNS
- P2P applications
- Socket programming with TCP
- Socket programming with UDP

Application architectures

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

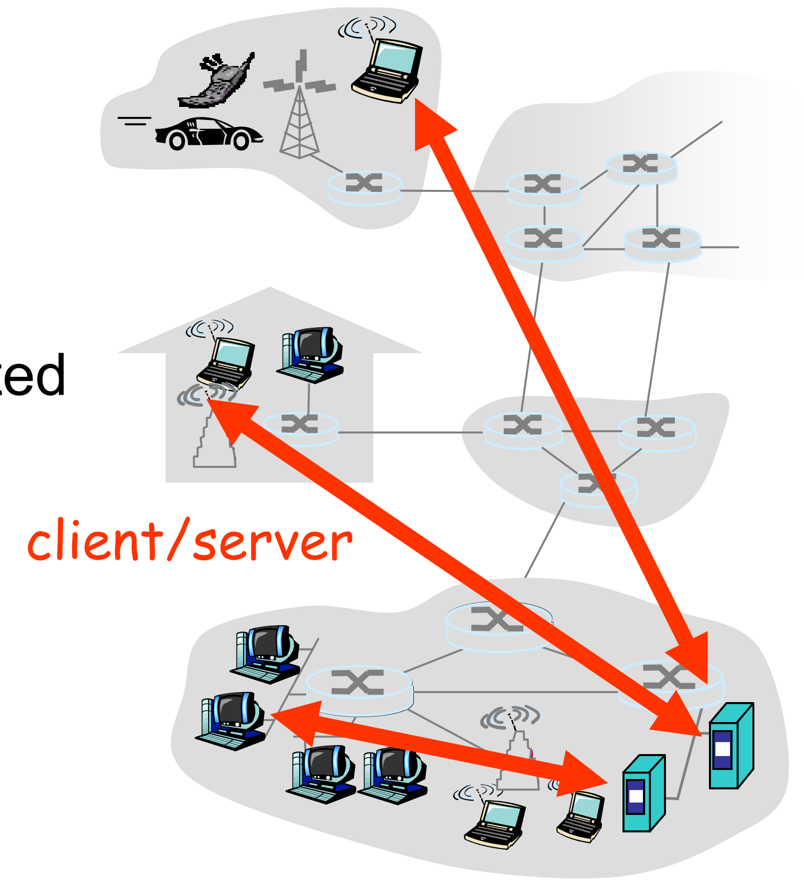
Client-server architecture

server:

- always-on host
- permanent IP address
- server farms for scaling

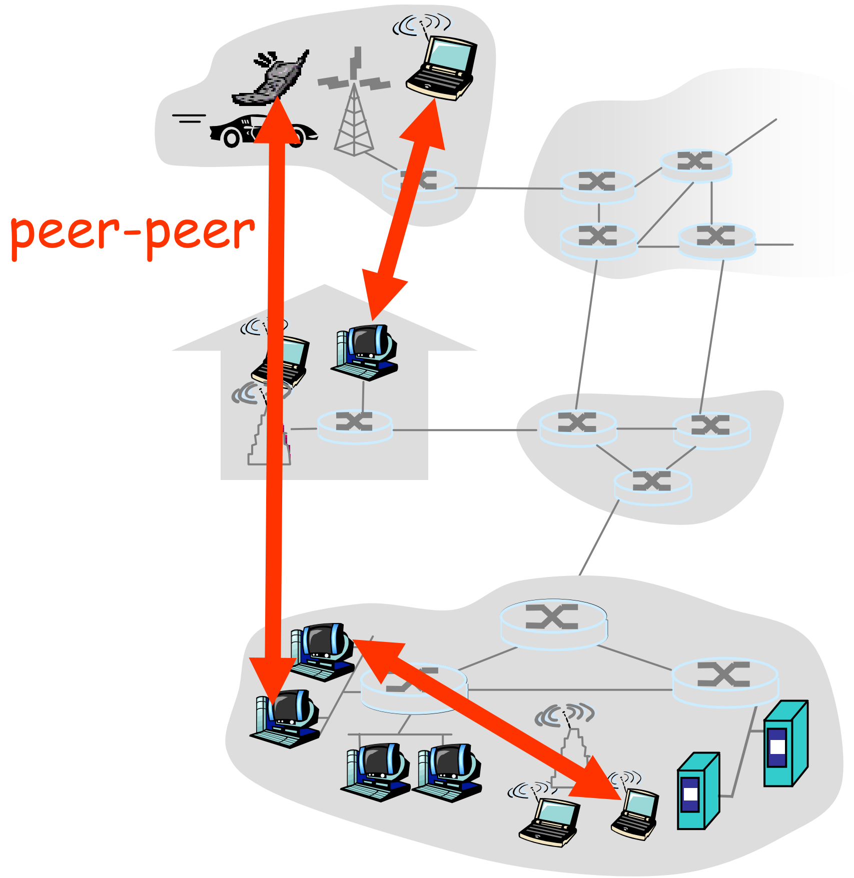
clients:

- communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other



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

Hybrid of client-server and P2P

Skype

- voice-over-IP P2P application
- centralized server: authenticates user, finds address of remote party
- client-client connection: direct (not through server)

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

Processes communicating

Process: program running within a host.

- within same host, two processes communicate using inter-process communication (defined by OS).
- processes in different hosts communicate by exchanging messages

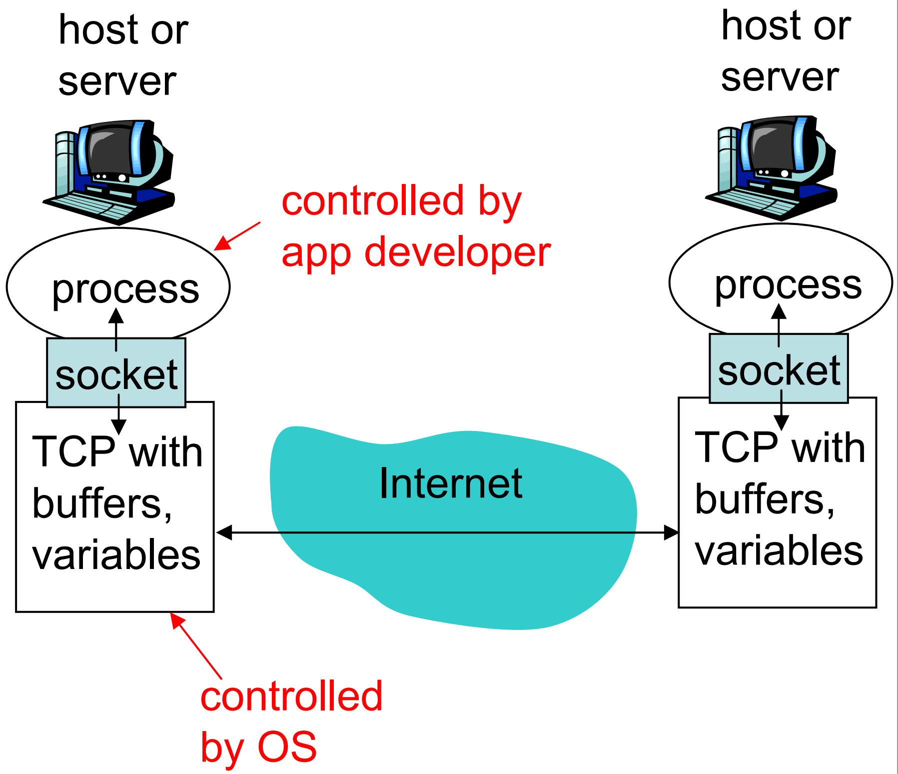
Client process: process that initiates communication

Server process: process that waits to be contacted

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

Sockets

- process sends/receives messages to/from its socket
- socket analogous to door
  - sending process shoves message out door
  - sending process relies on transport infrastructure on other side of door which brings message to socket at receiving process



- API: (1) choice of transport protocol; (2) ability to fix a few parameters (lots more on this later)

Addressing processes

- to receive messages, process must have *identifier*
- host device has unique 32-bit IP address
- *Q:* does IP address of host suffice for identifying the process?

Addressing processes

- to receive messages, process must have *identifier*
- host device has unique 32-bit IP address
- Q: does IP address of host on which process runs suffice for identifying the process?
  - A: No, *many* processes can be running on 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
- more shortly…

Application-layer protocol defines

- 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

Public-domain protocols:

- defined in RFCs
- allows for interoperability
- e.g., HTTP, SMTP

Proprietary protocols:

- e.g., Skype

What transport service does an application need?

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 "effective"
- frequently the applications also need timestamps (e.g. specifying playout time)

Throughput

- some apps (e.g., multimedia) require minimum amount of throughput to be "effective"
- other apps ("elastic apps") make use of whatever throughput they get

Security

- Some apps (e.g. Internet banking) require security services such as encryption, data integrity, …

Transport service requirements of common apps

Application	Data loss	Throughput	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	yes, few secs
interactive games	loss-tolerant	few kbps up	yes, 100's msec
instant messaging	no loss	elastic	yes and no

Internet transport protocols services

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:* sender throttled when network overloaded
- *does not provide:* timing, minimum throughput guarantees, security

UDP service:

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

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	HTTP (e.g., Youtube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	typically UDP

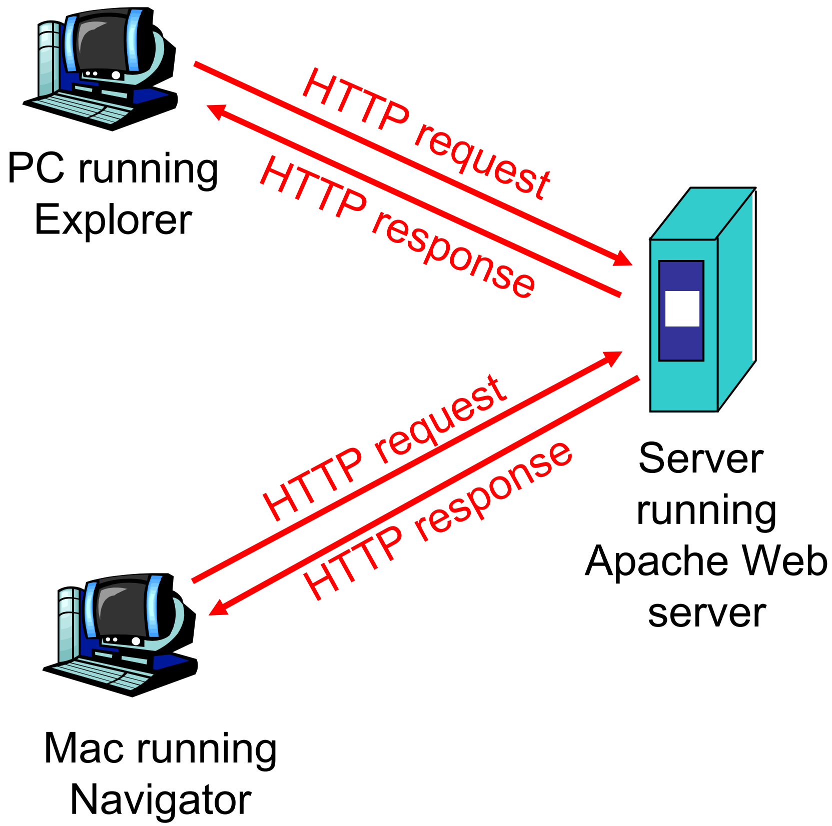
Chapter 2: Application layer

- Principles of network applications
- **Web and HTTP**
- DNS
- P2P applications
- Socket programming with TCP
- Socket programming with UDP

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 overview (continued)

HTTP uses TCP:

- 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)
- http1.0: TCP connection closed

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

⇨ research by PhD candidate Andreas Klenk: stateless negotiation protocol suitable for Web services

HTTP connections

Nonpersistent HTTP (v1.0)

- At most one object is sent over a TCP connection.

Persistent HTTP (v1.1)

- Multiple objects can be sent over single TCP connection between client and server.

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.)

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

time

6. Steps 1-5 repeated for each of 10 jpeg objects

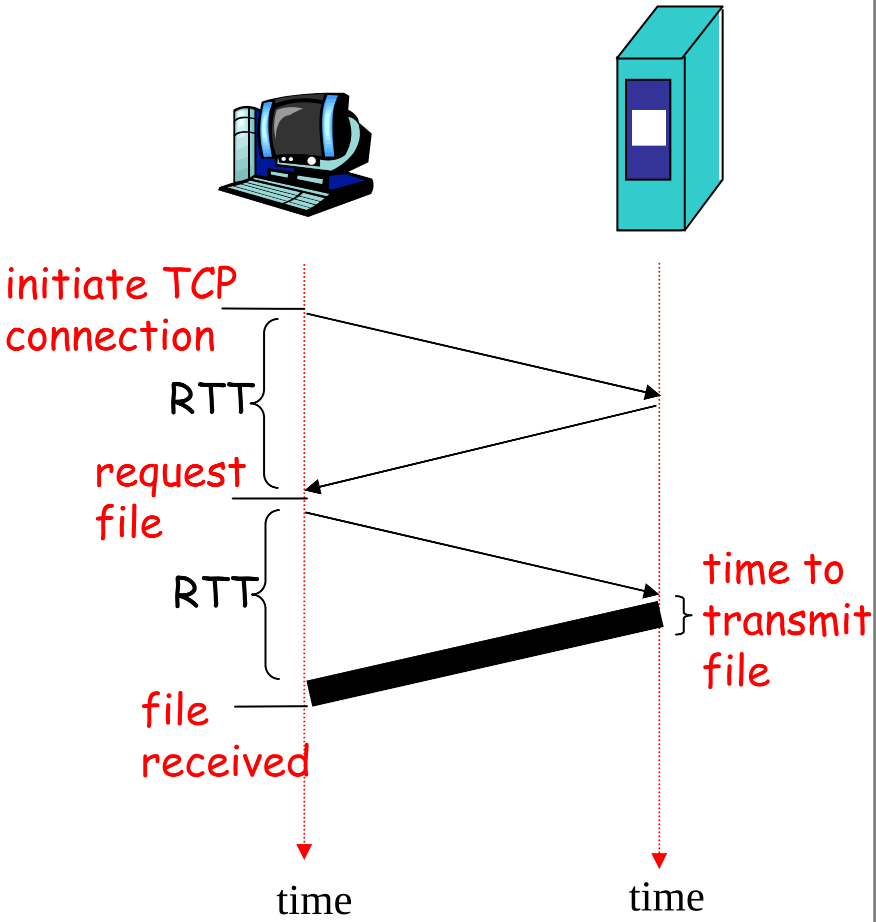
Non-Persistent HTTP: Response time

Definition of RTT: time for 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

total = 2RTT+ transmit time



Persistent HTTP

Nonpersistent HTTP issues:

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

Persistent HTTP

- server leaves connection open after sending response
- subsequent HTTP messages between same client/server sent over open connection
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects

HTTP request message

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

request line (GET, POST, HEAD commands)

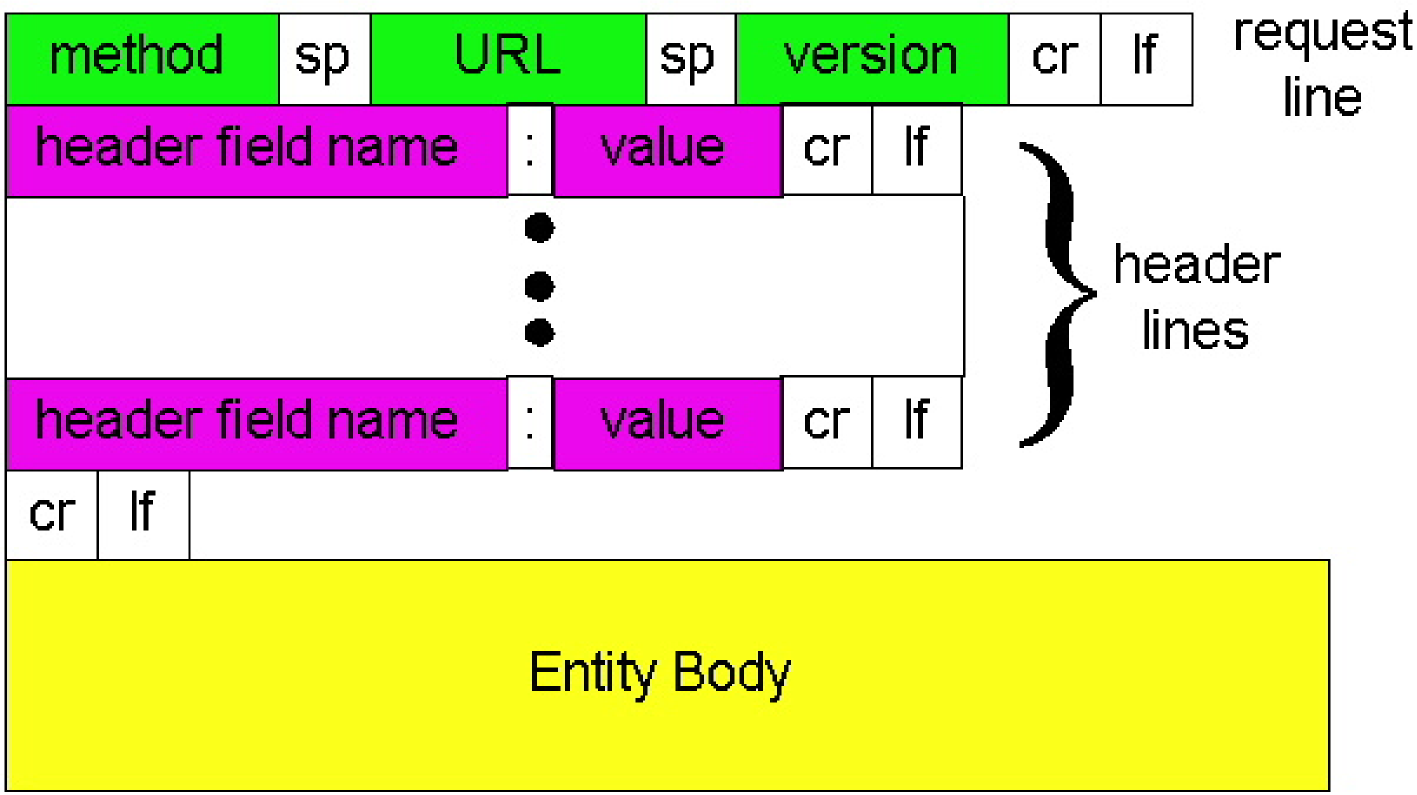
```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:fr
```

header lines

(extra carriage return, line feed)

Carriage return, line feed indicates end of message

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

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`

Trying out HTTP (client side) for yourself

1. Telnet to your favorite Web server:

telnet cis.poly.edu 80

Opens TCP connection to port 80
(default HTTP server port) at cis.poly.edu.
Anything typed in sent
to port 80 at cis.poly.edu

2. Type in a GET HTTP request:

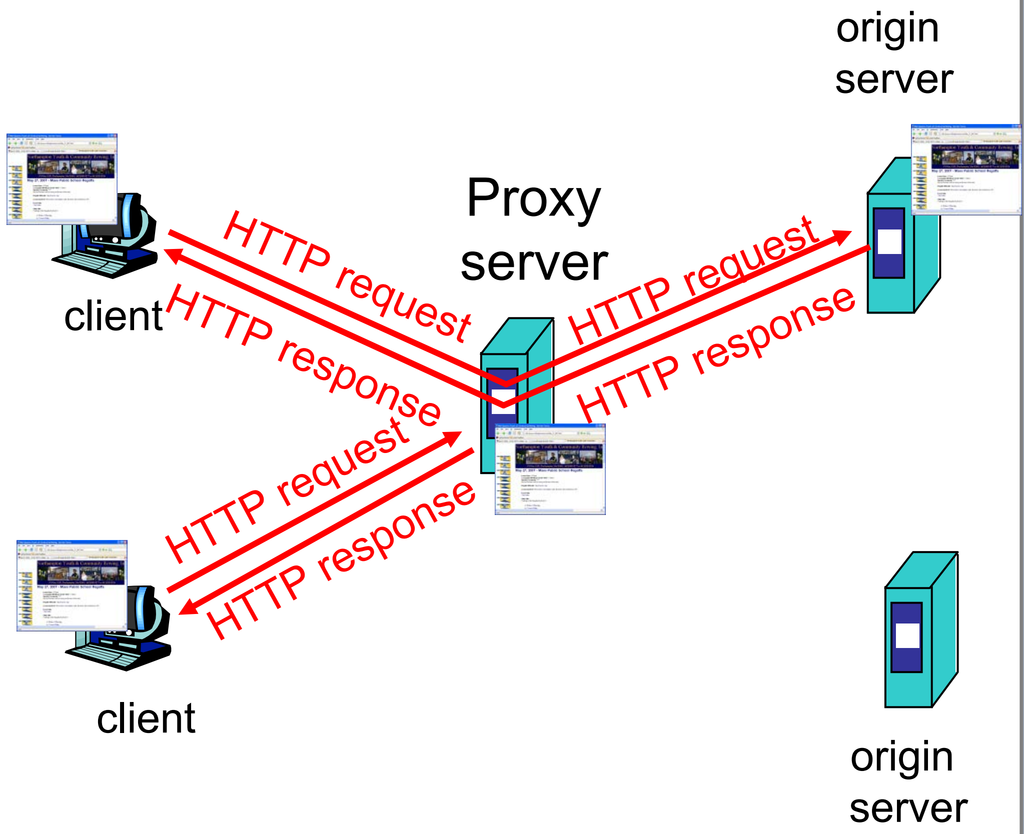
GET /~ross/ HTTP/1.1
Host: cis.poly.edu

By typing this in (hit carriage
return twice), you send
this minimal (but complete)
GET request to HTTP server

3. Look at response message sent by HTTP server!

Web caches (proxy server)

- Goal: satisfy client request without involving origin server
- non-transparent web cache:
  user sets browser: Web accesses via cache
- browser sends all HTTP requests to cache
  - object in cache: cache returns object
  - else cache requests object from origin server, then returns object to client



More about Web caching

- cache acts as both client and server
- typically cache is installed by ISP (university, company, residential ISP)

Why Web caching?

- reduce response time for client request
  (Q.: under which condition is this statement true?)
- reduce traffic on an institution's access link.
- Internet dense with caches: enables "poor" content providers to effectively deliver content (but so does P2P file sharing)

Caching example

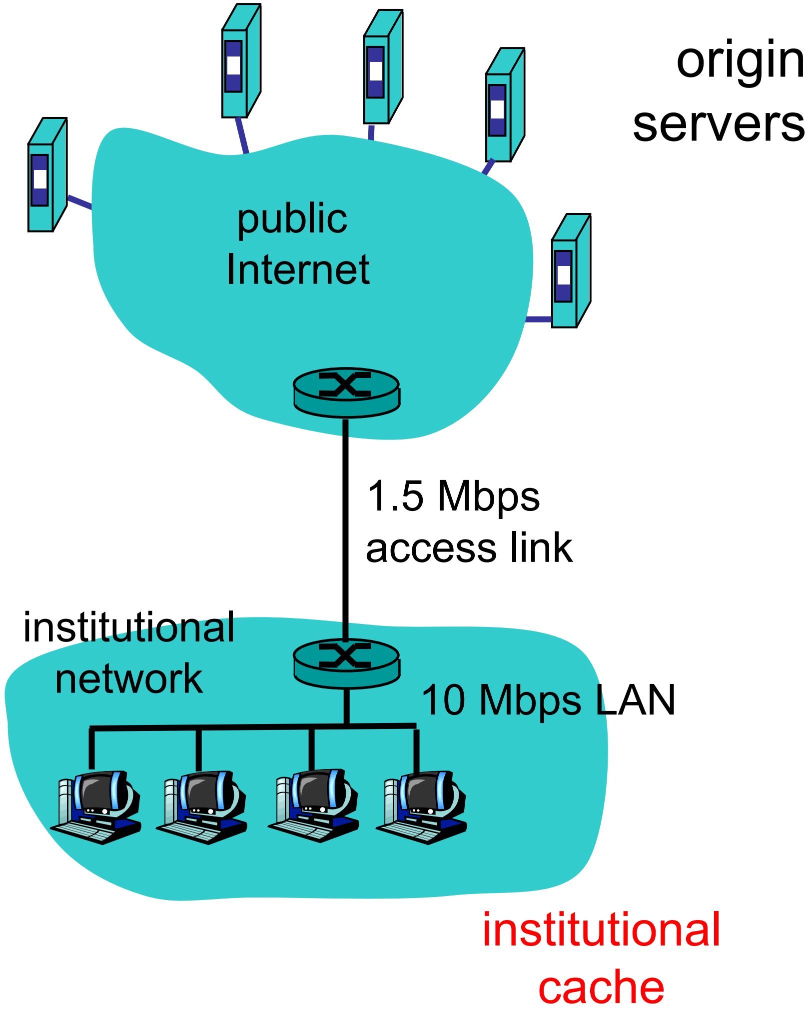
Assumptions

- average object size = 100,000 bits
- avg. request rate from institution's browsers to origin servers = 15/sec
- delay from institutional router to any origin server and back to router = 2 sec

Consequences

- utilization on LAN = 15%
- utilization on access link = 100%
- total delay = Internet delay + access delay + LAN delay

= 2 sec + minutes + milliseconds



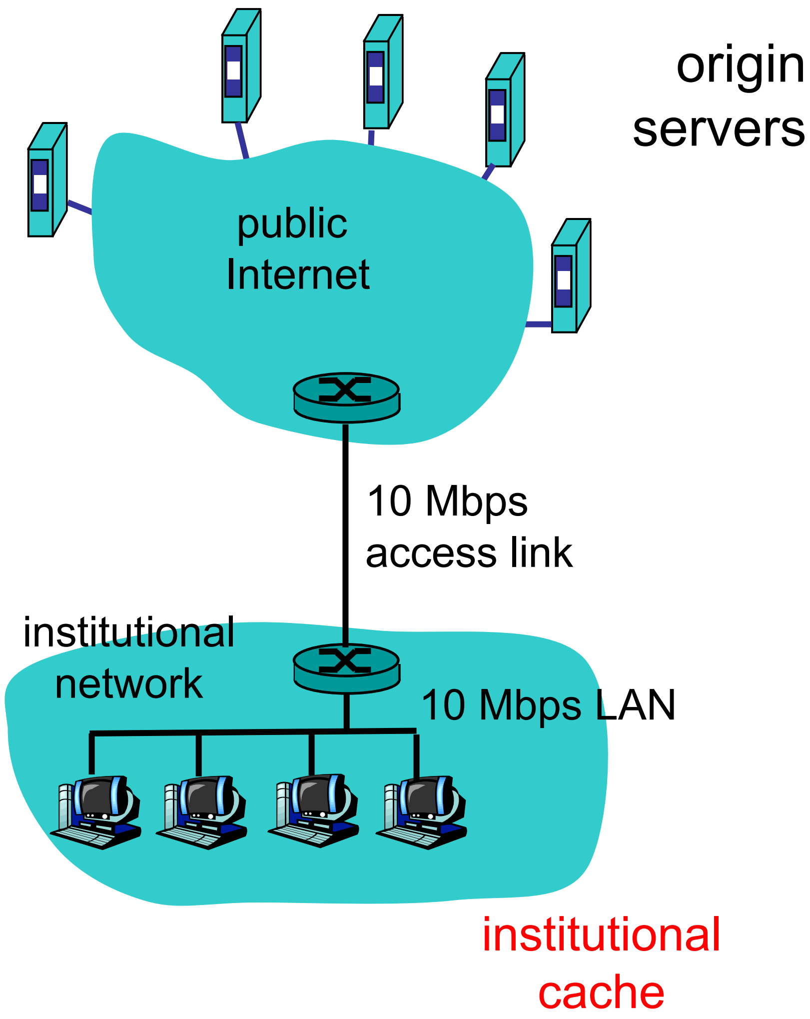
Caching example (cont)

possible solution

- increase bandwidth of access link to, say, 10 Mbps

consequence

- utilization on LAN = 15%
- utilization on access link = 15%
- Total delay = Internet delay + access delay + LAN delay
  = 2 sec + msecs + msecs
- often a costly upgrade



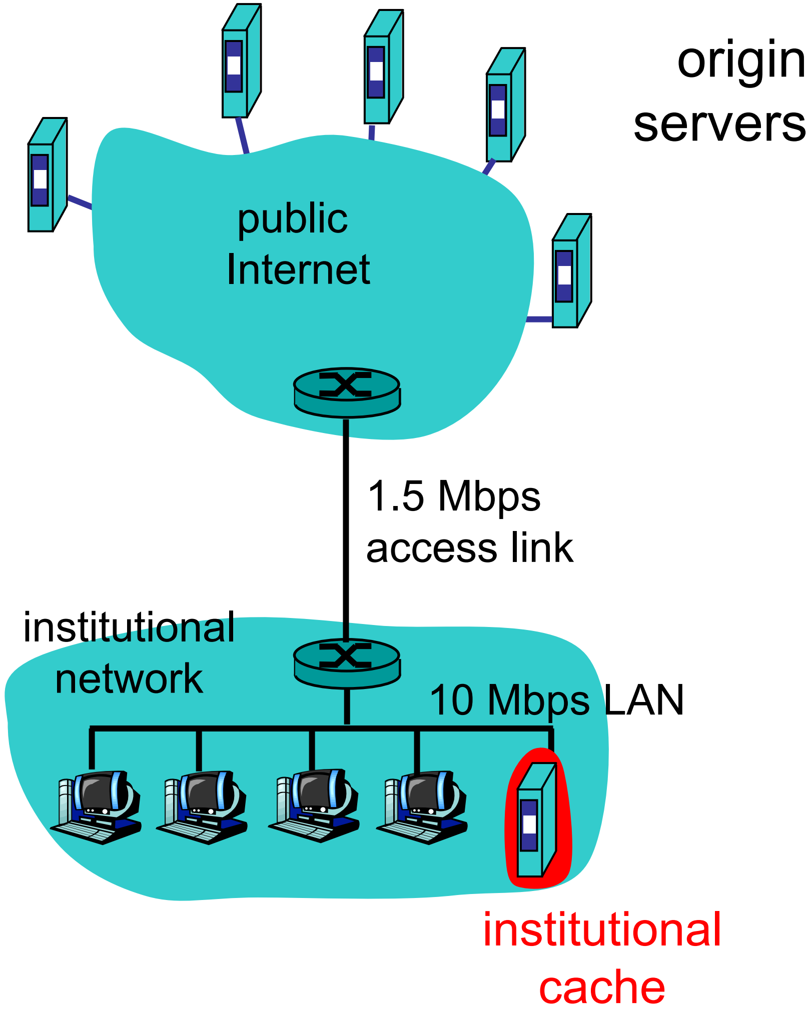
Caching example (cont)

possible solution: install cache

- suppose hit rate is 0.4

consequence

- 40% requests will be satisfied almost immediately
- 60% requests satisfied by origin server
- utilization of access link reduced to 60%, resulting in negligible delays (say 10 msec)
- total avg delay = Internet delay + access delay + LAN delay = .6*(2.01) secs + .4*milliseconds < 1.4 secs

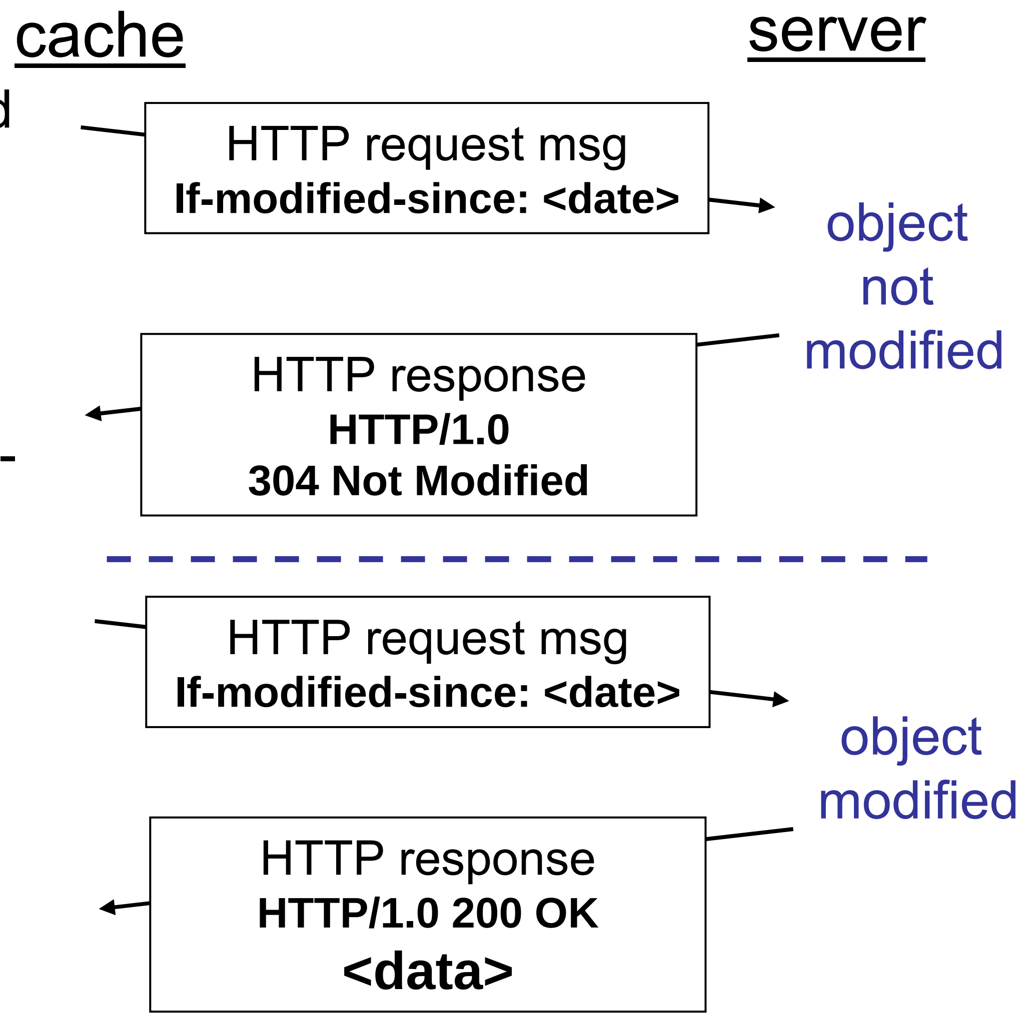


Conditional GET

- Goal: don't send object if cache has up-to-date cached version
- cache: specify date of cached copy in HTTP request

  `If-modified-since: <date>`
- server: response contains no object if cached copy is up-to-date:

  `HTTP/1.0 304 Not Modified`



Chapter 2: Application layer

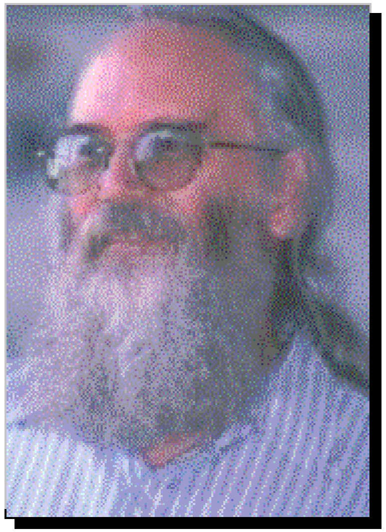
- Principles of network applications
- Web and HTTP
- **DNS**
- P2P applications
- Socket programming with TCP
- Socket programming with UDP

Paul Mockapetris

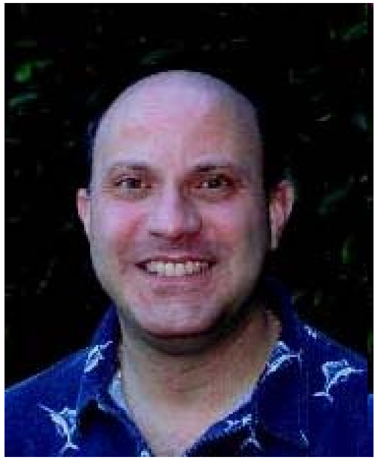
- „Father" of DNS
- Did design DNS in 1983, while working at Information Sciences Institute (ISI) of University of Southern California (USC)
- DNS Architecture: RFCs 882, 883
- Obsoleted by RFCs 1034,1035
- Company Nominum

Jon Postel

- Jon Postel (1943 – 1998)
  - Editor of RFC series
  - co-developer many Internet standards such as TCP/IP, SMTP, and DNS
  - Internet Assigned Numbers Authority (IANA)
  - "Be liberal in what you accept, and conservative in what you send."
  - obituary published in RFC 2468



- Postel Center at Information Sciences Institute, http://www.postel.org/
- Joe Touch
  Postel Center Director, USC/ISI
  Research Associate Professor, USC Dept. of Computer Science



DNS: Domain Name System

People: many identifiers:

- Social Secuity Number, name, passport #

Internet hosts, routers:

- IP address (32 bit) - used for addressing datagrams
- "name", e.g., ww.yahoo.com - used by humans

Q: map between IP addresses and name ?

Domain Name System:

- *distributed database* implemented in hierarchy of many *name servers*
- *application-layer protocol* host, routers, name servers to communicate to *resolve* names (address/name translation)
  - note: core Internet function, implemented as application-layer protocol
  - complexity at network's "edge"

DNS

Why not centralize DNS?

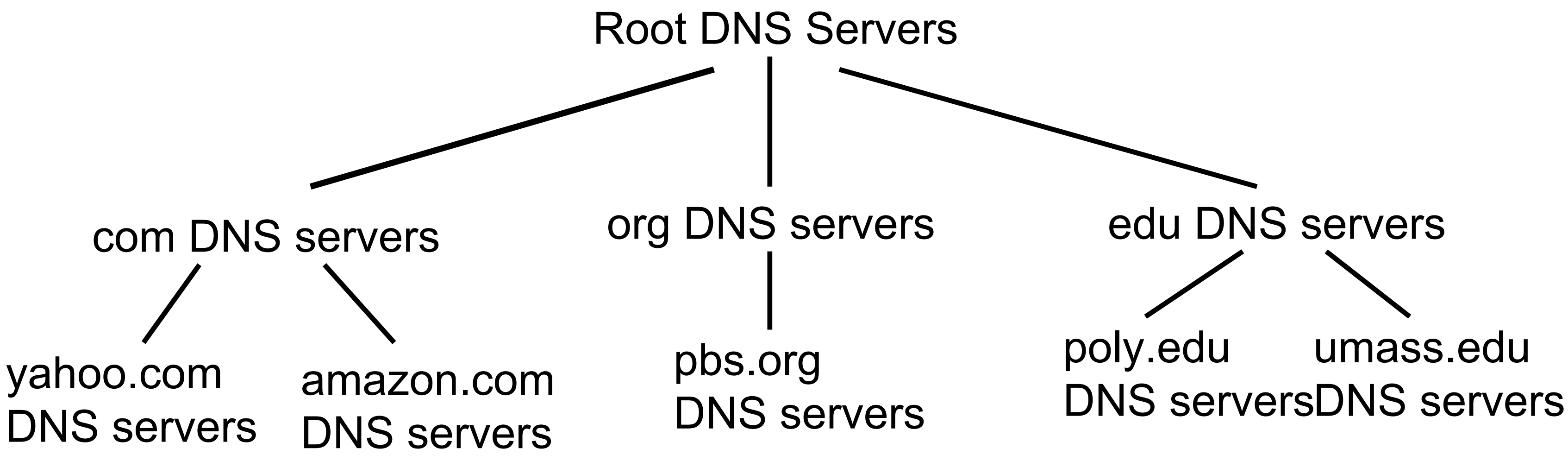
- single point of failure
- traffic volume
- distant centralized database
- maintenance

doesn't *scale!*

DNS services

- hostname to IP address translation
- host aliasing
  - Canonical, alias names
- mail server aliasing
- load distribution
  - replicated Web servers: set of IP addresses for one canonical name

Distributed, Hierarchical Database

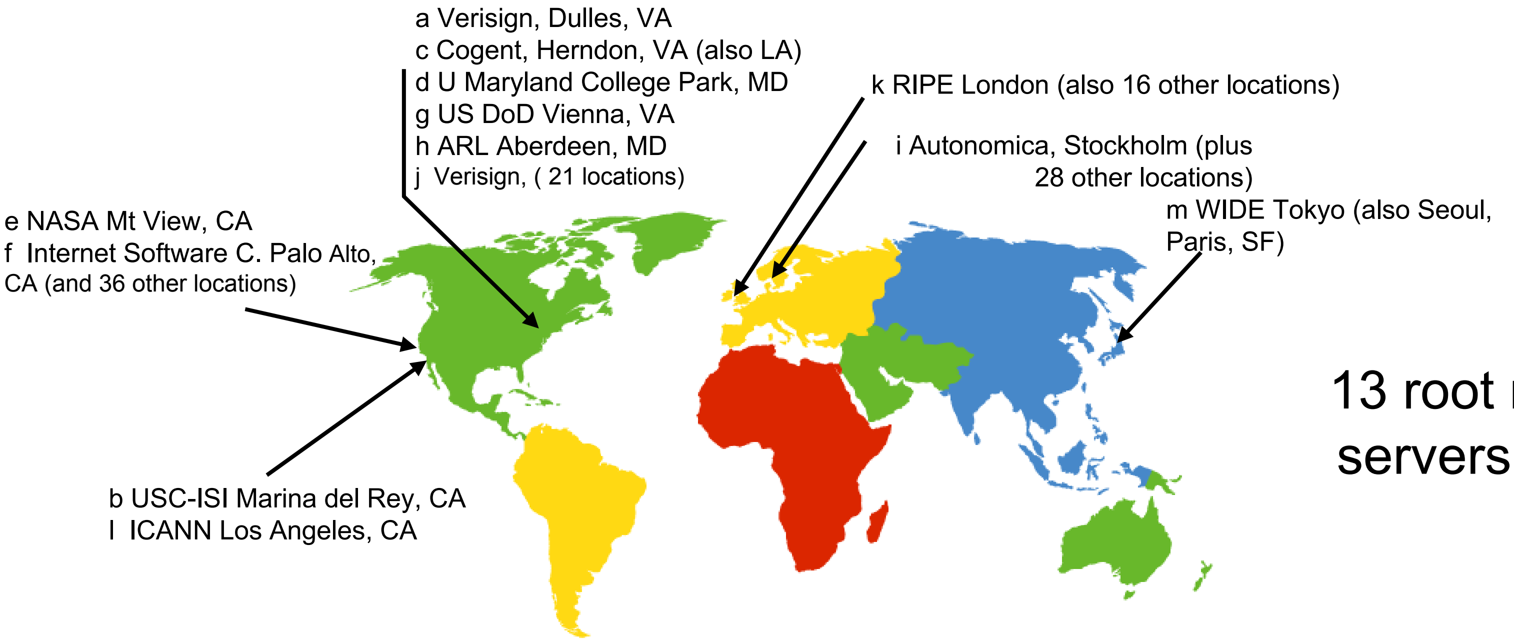


Client wants IP for www.amazon.com; 1st approx:

- 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 can not 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

DNS root servers

- Only 13 physical servers?
  - No, there are 13 operators of a redundant set of DNS root servers
  - nine of the servers operate in multiple geographical locations using anycast routing (→ later), for better performance and more fault toleranc

TLD and Authoritative Servers

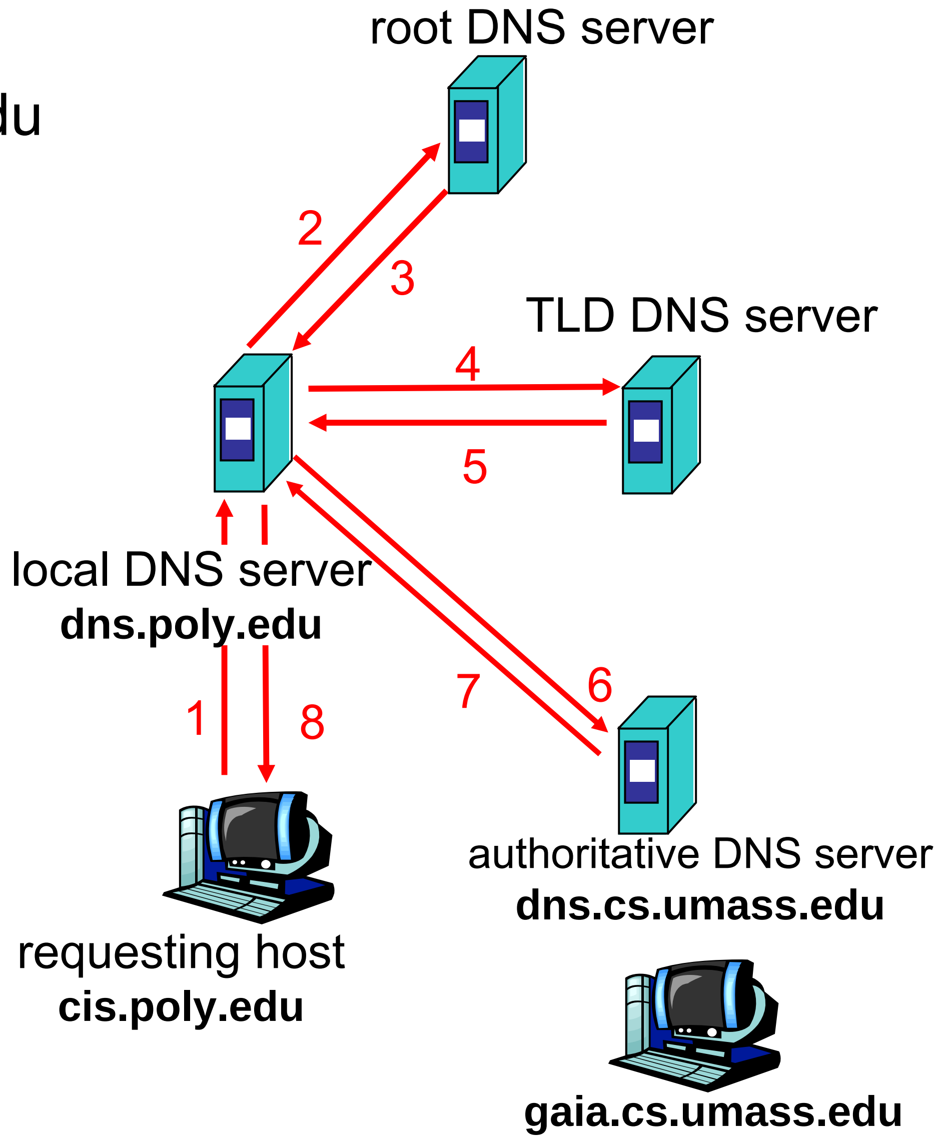
- Top-level domain (TLD) servers:
  - responsible for com, org, net, edu, etc, and all top-level country domains de, uk, fr, ca, jp…
  - the company Network Solutions maintains servers for com TLD
  - the company Educause for edu TLD
- Authoritative DNS servers:
  - organization's DNS servers, providing authoritative hostname to IP mappings for organization's servers (e.g., Web, 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 host makes DNS query, query is sent to its local DNS server
  - acts as proxy, forwards query into hierarchy

DNS name resolution example

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



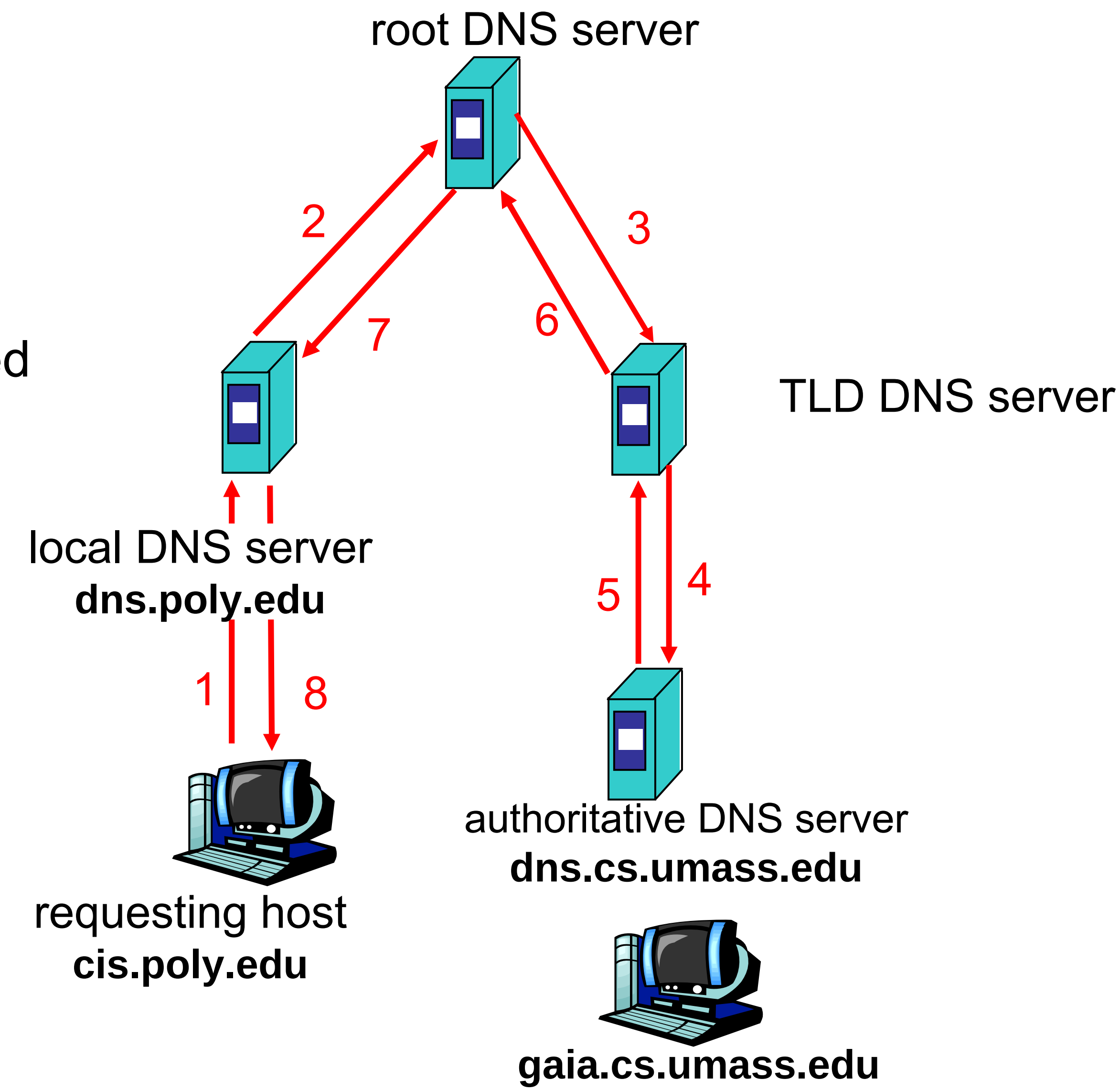
iterated query:

- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"

DNS name resolution example

recursive query:

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



DNS: caching and updating records

- once (any) name server learns mapping, it *caches* mapping
  - cache entries timeout (disappear) after some time
  - TLD servers typically cached in local name servers
    - Thus root name servers not often visited
- update/notify mechanisms designed by IETF
  - RFC 2136

DNS records

DNS: distributed database storing resource records (RR)

RR format: **(name, value, type, ttl)**

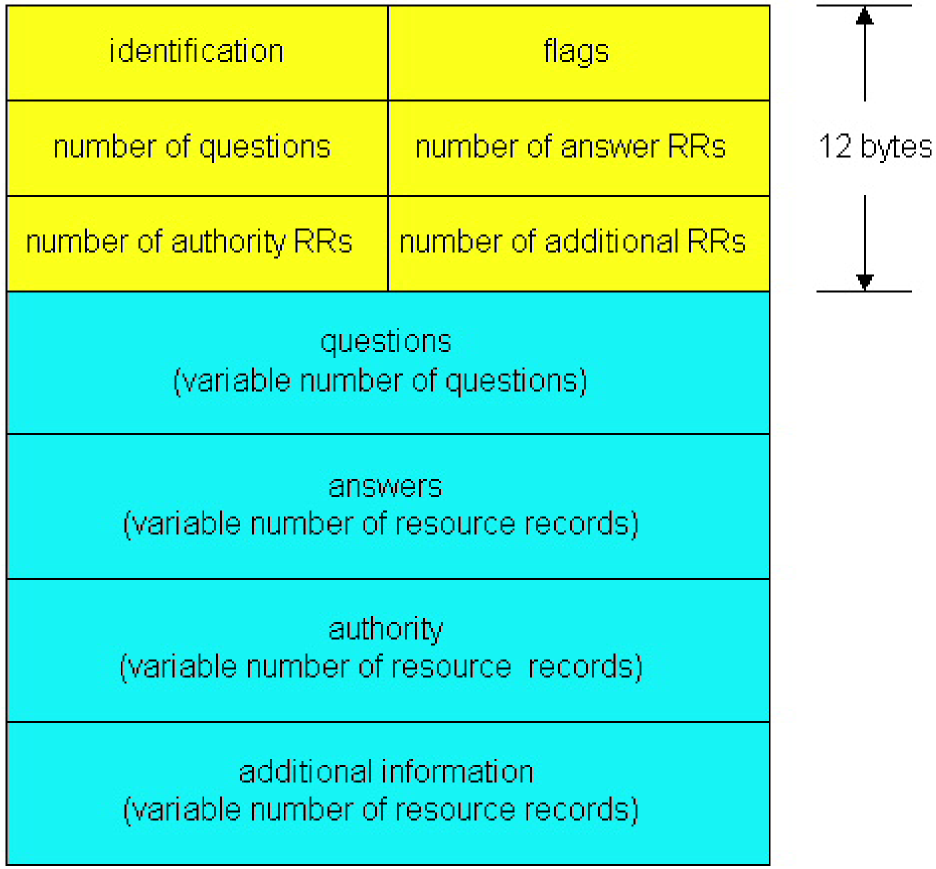
- Type=A
  - name is hostname
  - value is IP address
- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** is hostname of authoritative name server for this domain
- Type=CNAME
  - name is alias name for some "canonical" (the real) name
  - www.ibm.com is really servereast.backup2.ibm.com
  - value is canonical name
- Type=MX
  - value is name of mailserver associated with name

DNS protocol, messages

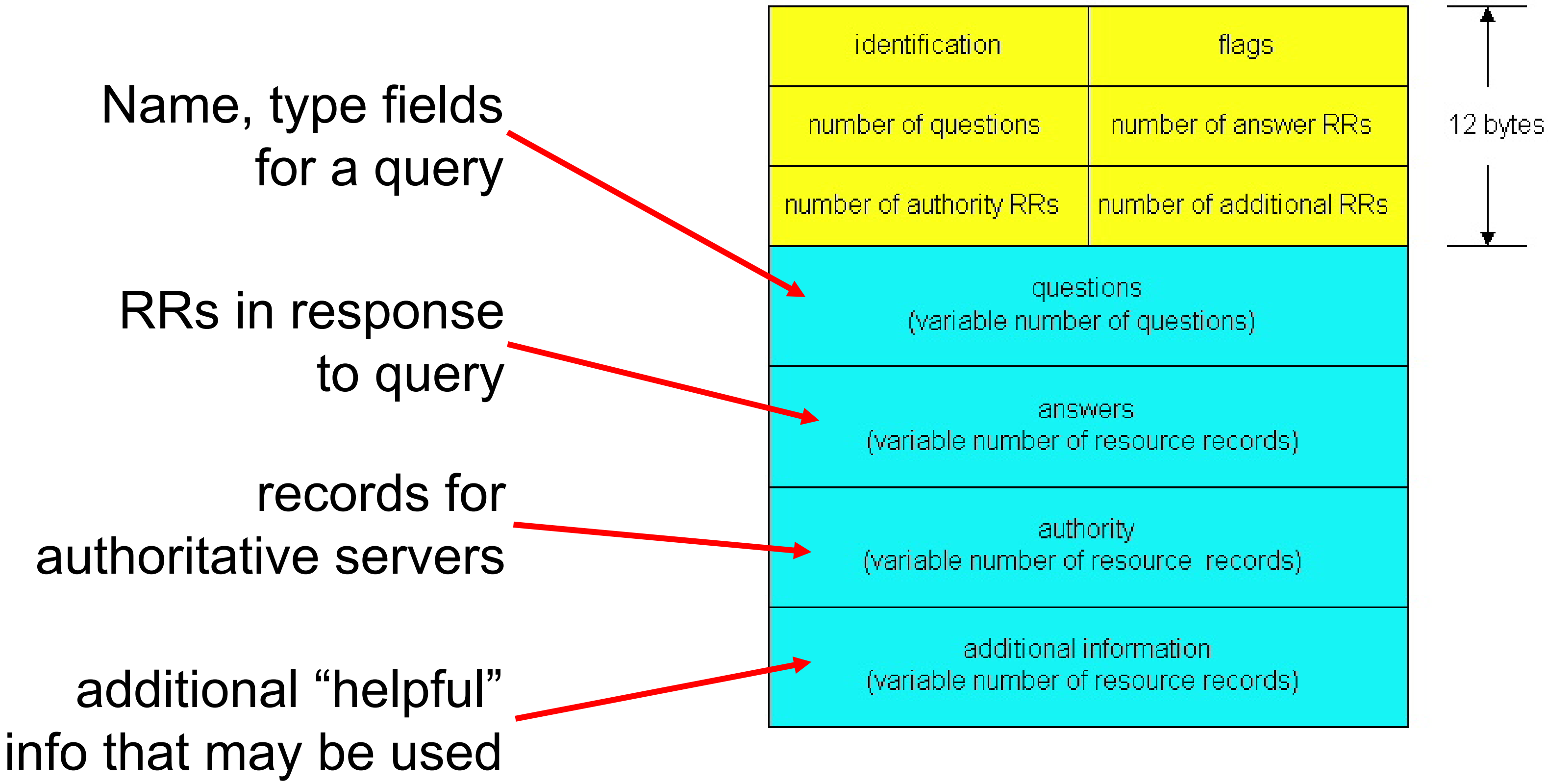
DNS protocol : *query* and *reply* messages, both with same *message format*

msg header

- identification: 16 bit # for query, reply to query uses same #
- flags:
  - query (0) or reply (1)
  - recursion desired (1)
  - recursion available (1)
  - reply is authoritative (1)



DNS protocol, messages



Inserting records into DNS

- example: new startup "Network Utopia"
- register name networkuptopia.com at *DNS* *registrar* (e.g., Network Solutions)
  - provide names, IP addresses of authoritative name server (primary and secondary)
  - registrar inserts two RRs into com TLD server:

```
(networkutopia.com, dns1.networkutopia.com,
  NS)
(dns1.networkutopia.com, 212.212.212.1, A)
```

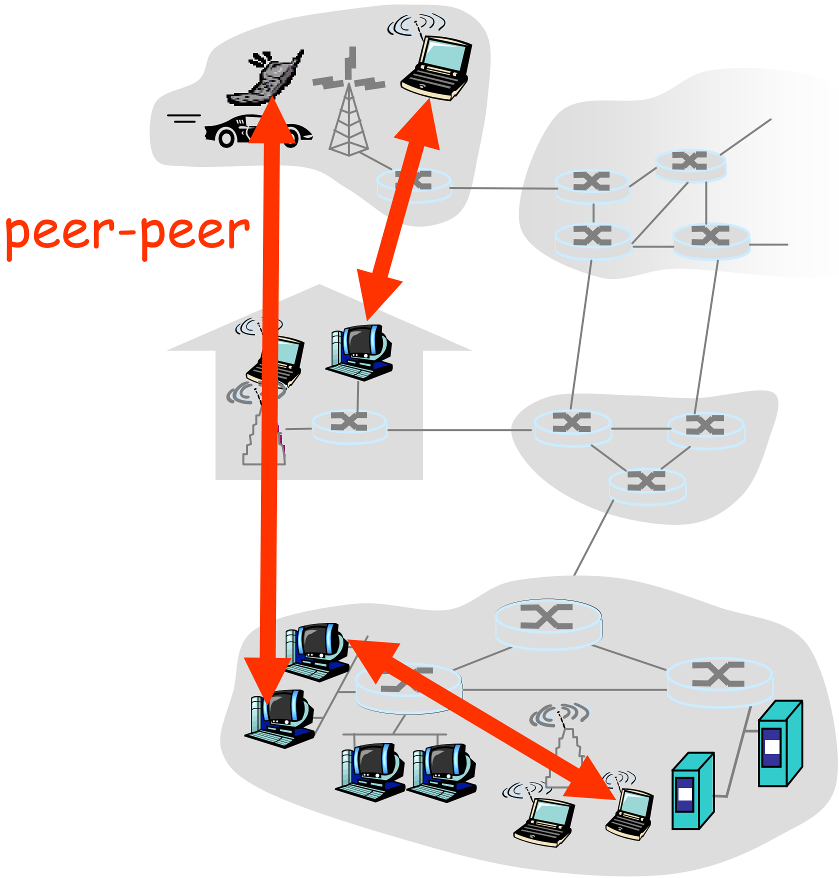
- create authoritative server Type A records for www.networkuptopia.com;
  Type MX record for networkutopia.com
- How do people get IP address of your Web site?

- Principles of network applications
- Web and HTTP
- DNS
- **P2P applications**
- Socket programming with TCP
- Socket programming with UDP

Pure P2P architecture

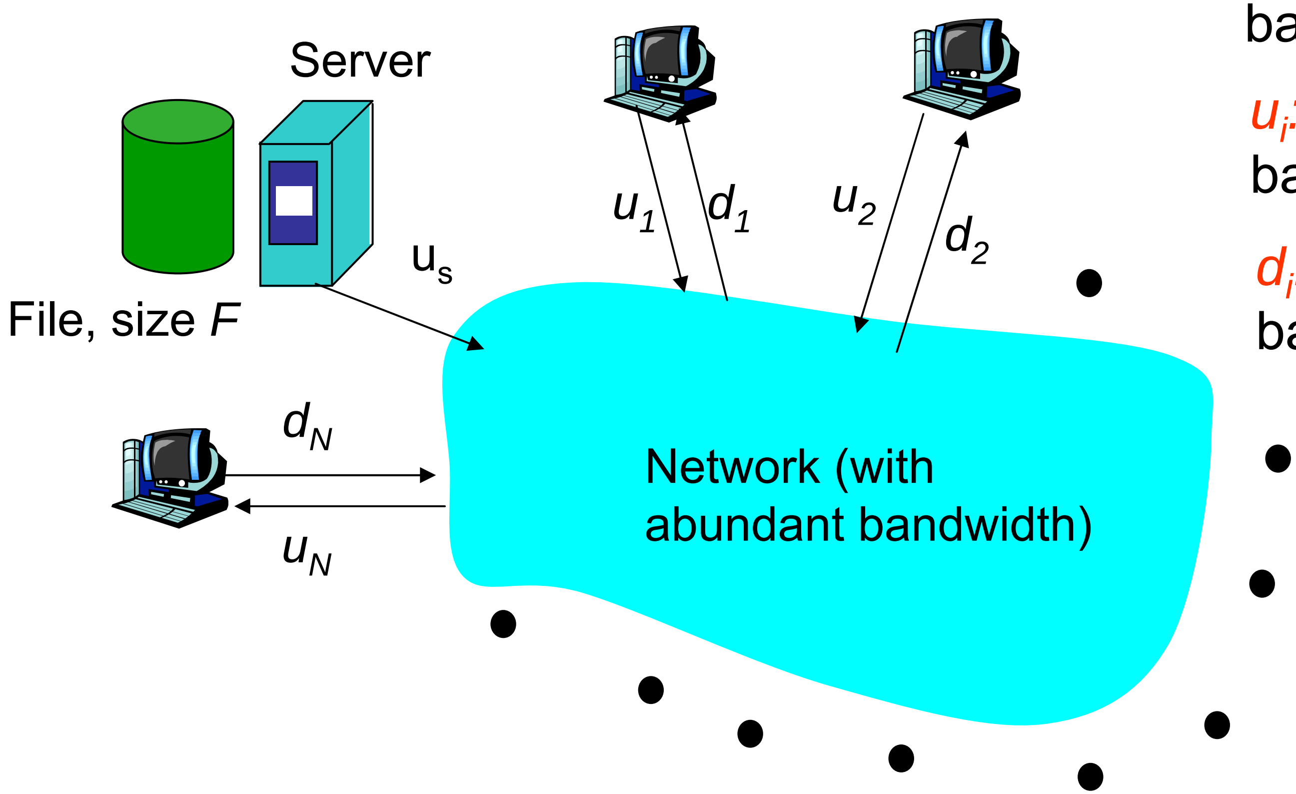
- *no* always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses

- Three topics:
  - File distribution
  - Searching for information
  - Case Study: Skype



File Distribution: Server-Client vs P2P

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



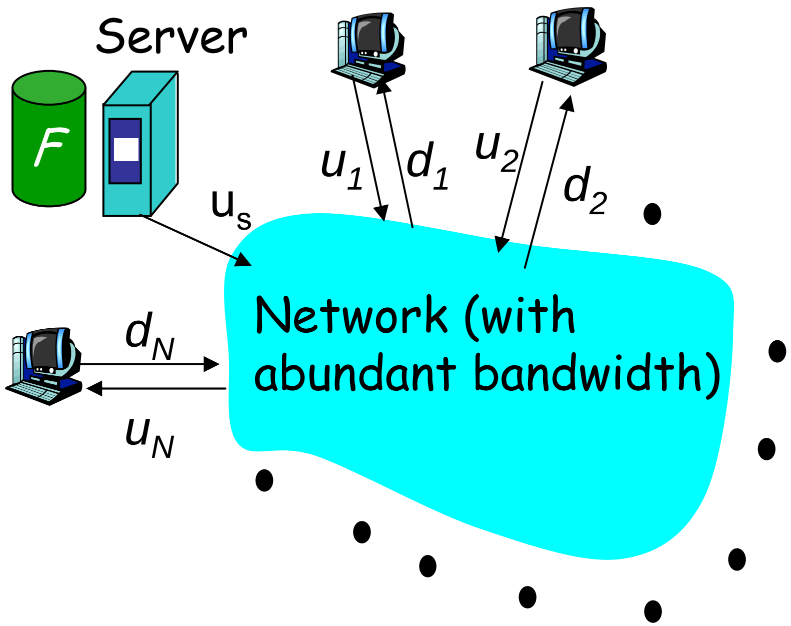
u_s: server upload bandwidth

u_i: peer i upload bandwidth

d_i: peer i download bandwidth

File distribution time: server-client

- server sequentially sends N copies:
  - NF/u_s time
- client i takes F/d_i time to download



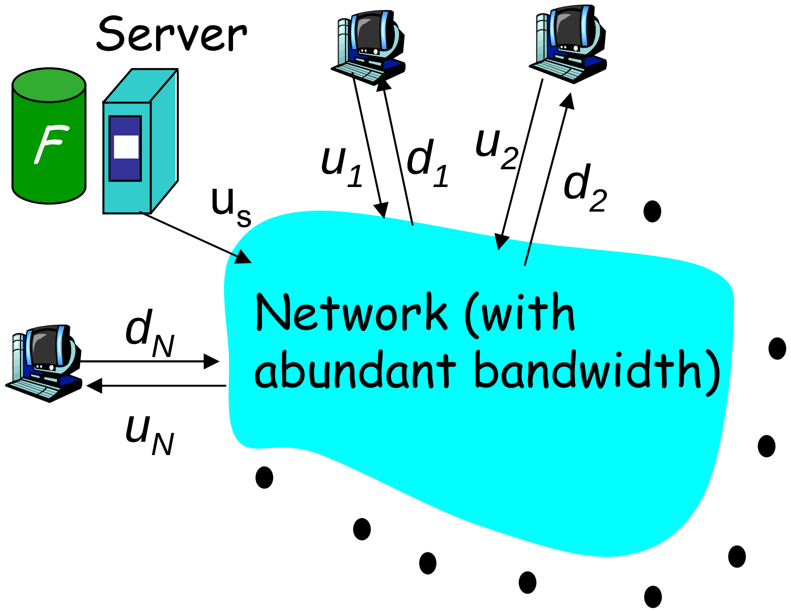
Time to distribute *F* to *N* clients using client/server approach $= d_{cs} = \max \{ NF/u_s, F/\min_i(d_i) \}$

increases linearly in N (for large N)

File distribution time: P2P

- server must send one copy: F/u_s time
- client i takes F/d_i time to download
- NF bits must be downloaded (aggregate)

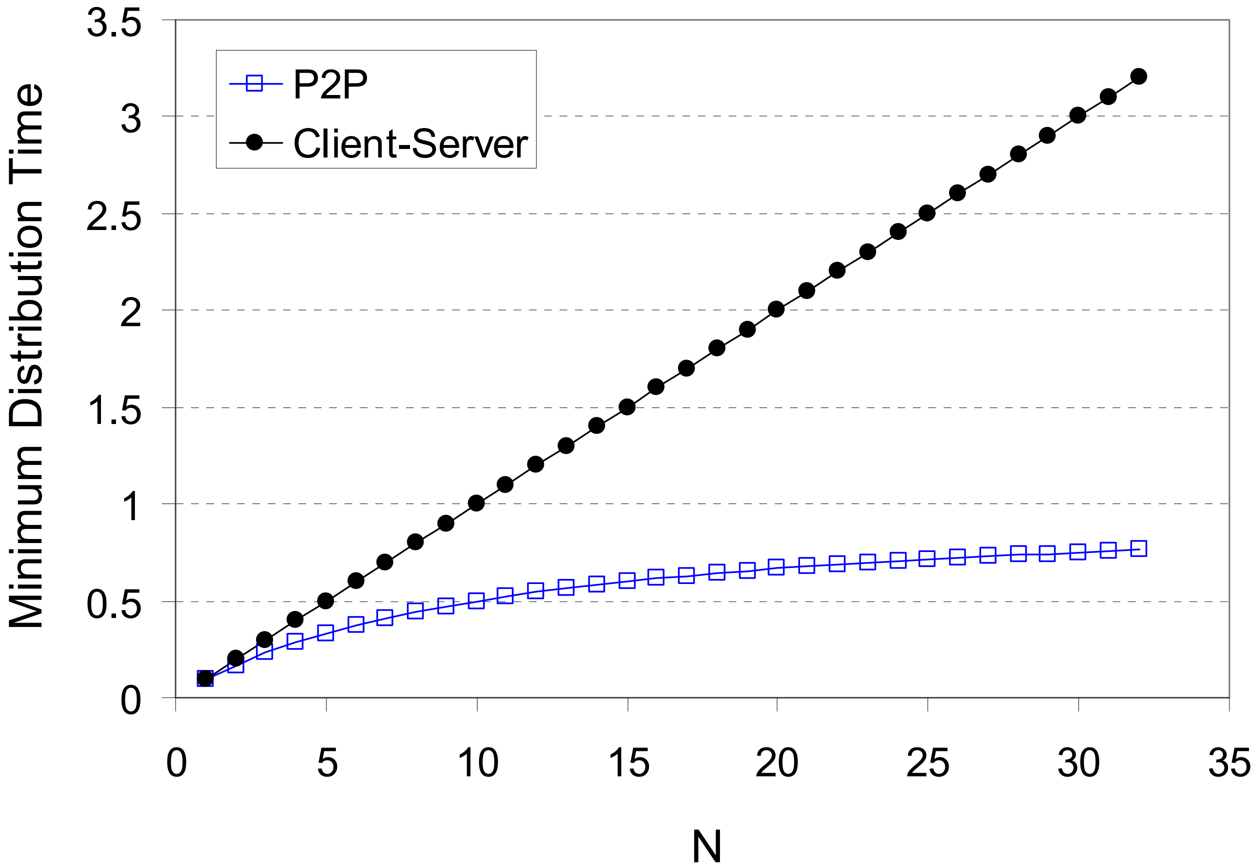
fastest possible upload rate: $u_s + \Sigma u_i$



$$d_{P2P} = \max \left\{ F/u_s,\ F/\min_i(d_i),\ NF/(u_s + \Sigma u_i) \right\}$$

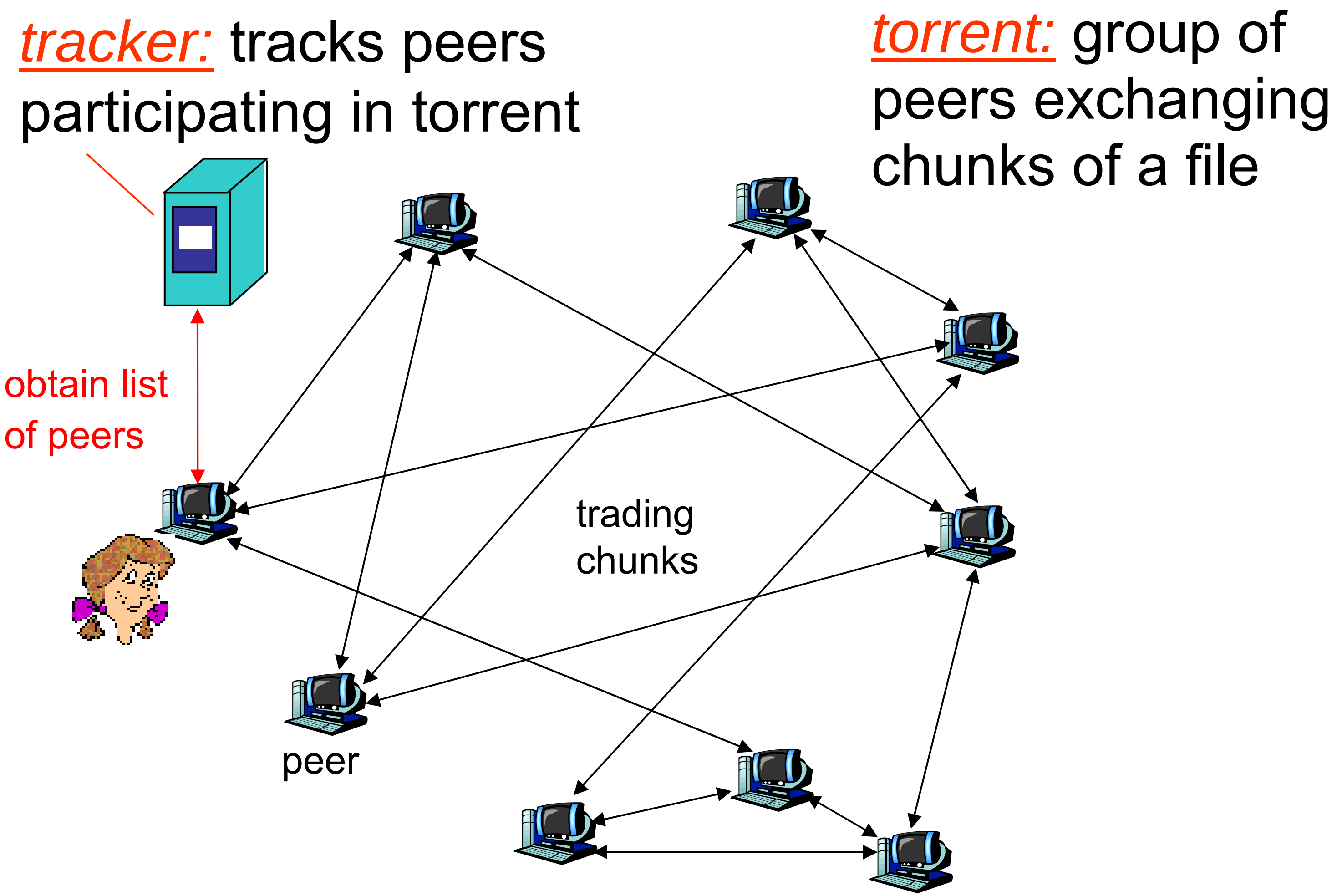
Server-client vs. P2P: example

Client upload rate = u, F/u = 1 hour, $u_s = 10u$, $d_{min} \geq u_s$



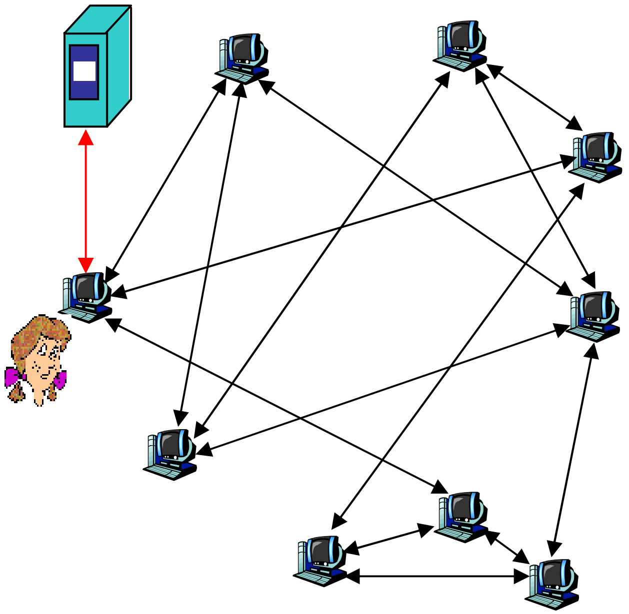
File distribution: BitTorrent

- P2P file distribution



BitTorrent (1)

- 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 (2)

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

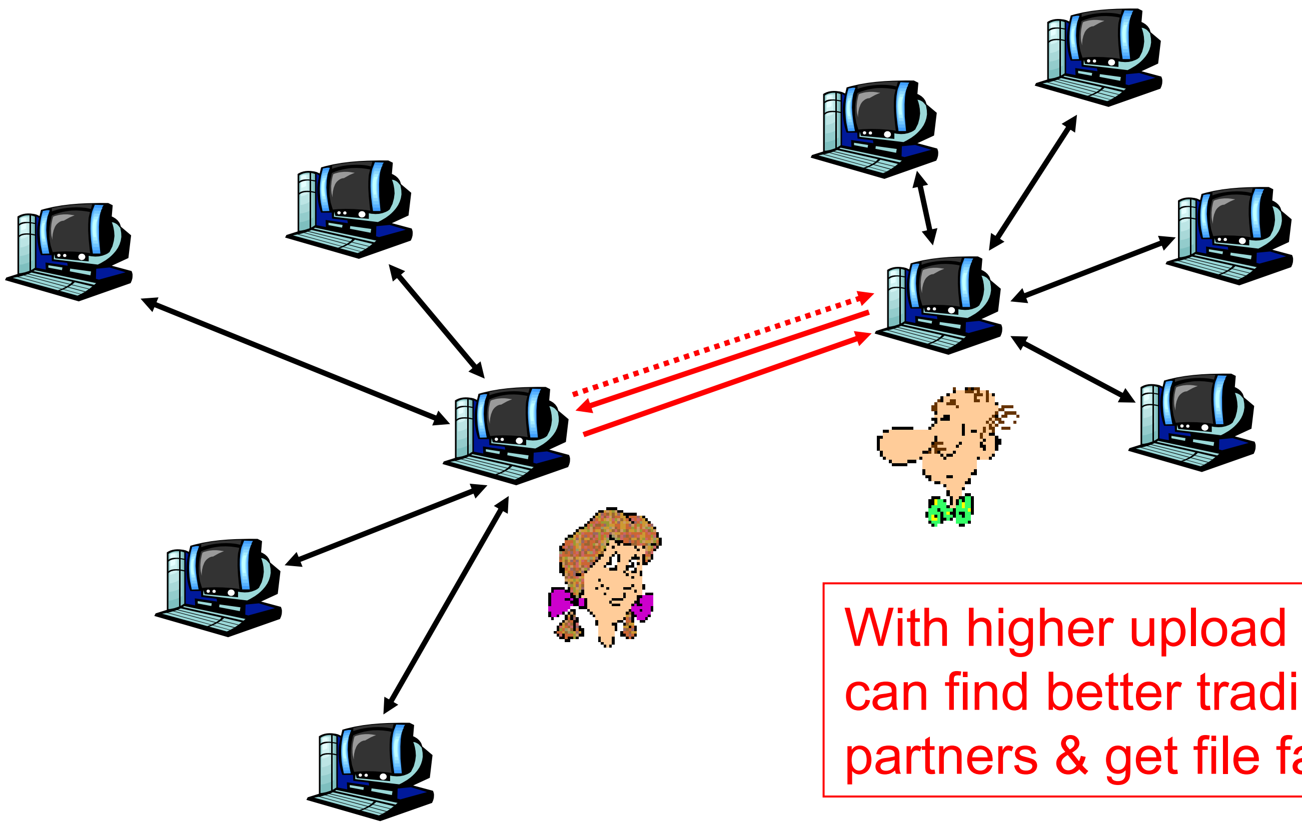
- Alice sends chunks to four neighbors currently sending her chunks *at the highest rate*
  - re-evaluate top 4 every 10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - newly chosen peer may join top 4
  - "optimistically unchoke"

BitTorrent: Tit-for-tat

(1) Alice "optimistically unchokes" Bob

(2) Alice becomes one of Bob's top-four providers; Bob reciprocates

(3) Bob becomes one of Alice's top-four providers



With higher upload rate, can find better trading partners & get file faster!

Distributed Hash Table (DHT)

- DHT = distributed P2P database
- Database has (key, value) pairs;
  - key: ss number; value: human name
  - key: content type; value: IP address
- Peers query DB with key
  - DB returns values that match the key
- Peers can also insert (key, value) peers

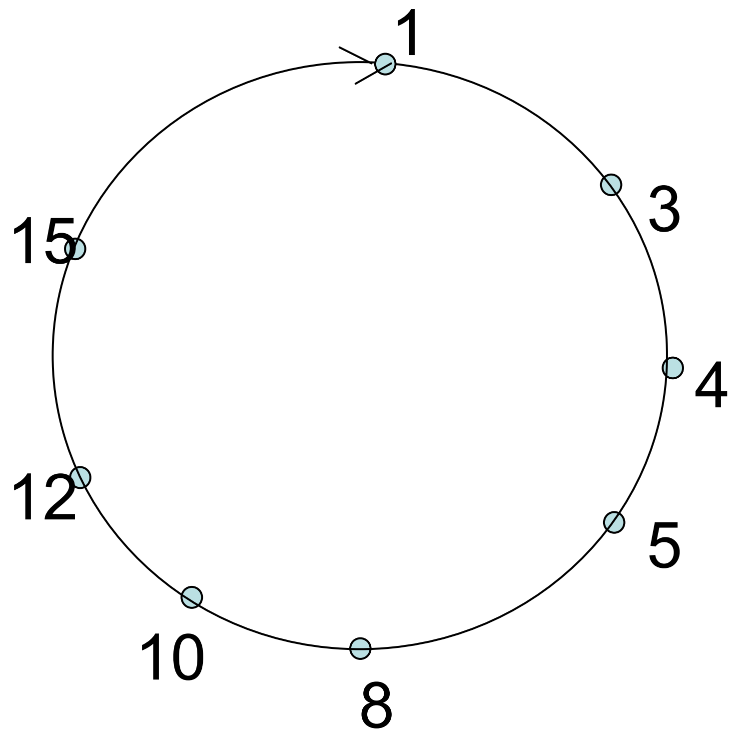
DHT Identifiers

- Assign integer identifier to each peer in range $[0,2^n-1]$.
  - Each identifier can be represented by n bits.
- Require each key to be an integer in same range.
- To get integer keys, hash original key.
  - eg, key = h(“Led Zeppelin IV”)
  - This is why they call it a distributed “hash” table

How to assign keys to peers?

- Central issue:
  - Assigning (key, value) pairs to peers.
- Rule: assign key to the peer that has the closest ID.
- Convention in lecture: closest is the immediate successor of the key.
- Ex: n=4; peers: 1,3,4,5,8,10,12,14;
  - key = 13, then successor peer = 14
  - key = 15, then successor peer = 1

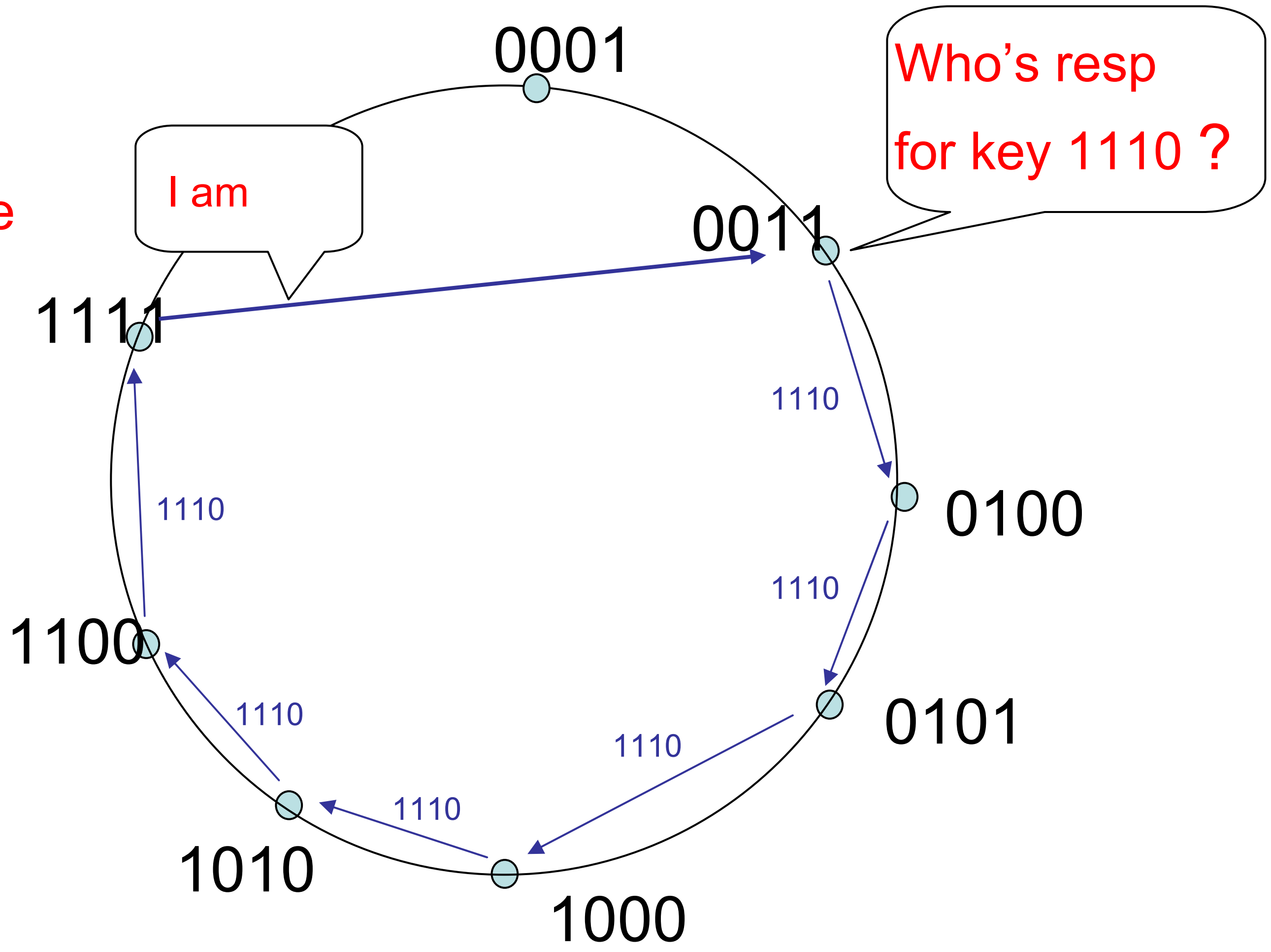
Circular DHT (1)



- Each peer *only* aware of immediate successor and predecessor.
- “Overlay network”

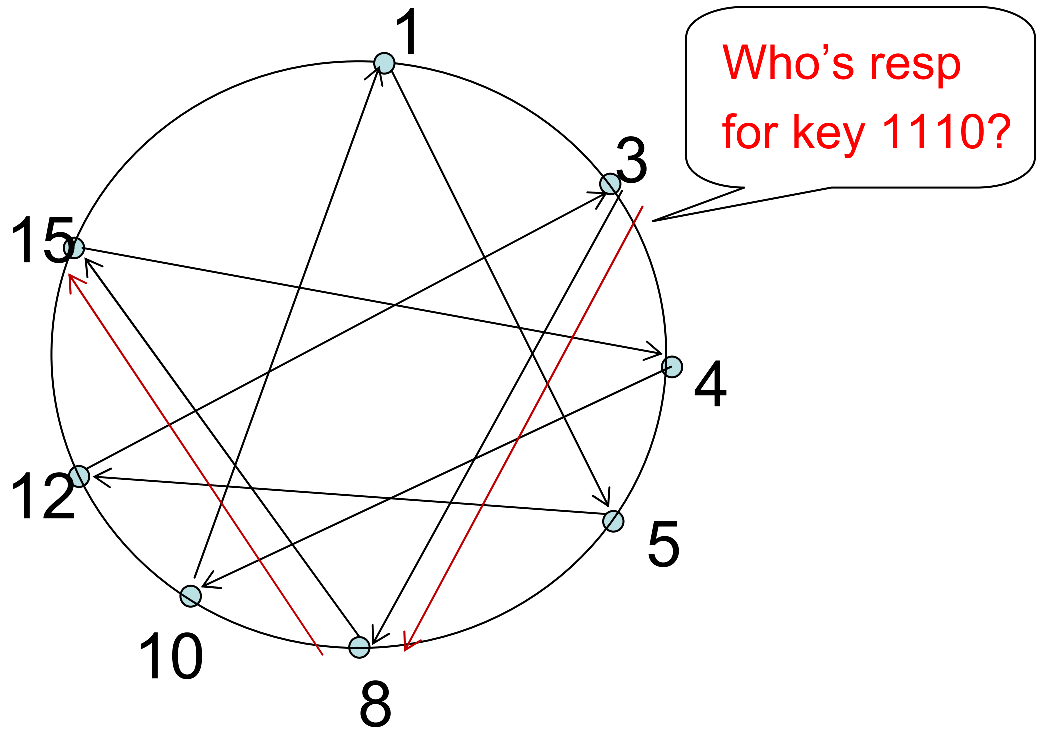
Circle DHT (2)

O(N) messages on avg to resolve query, when there are N peers



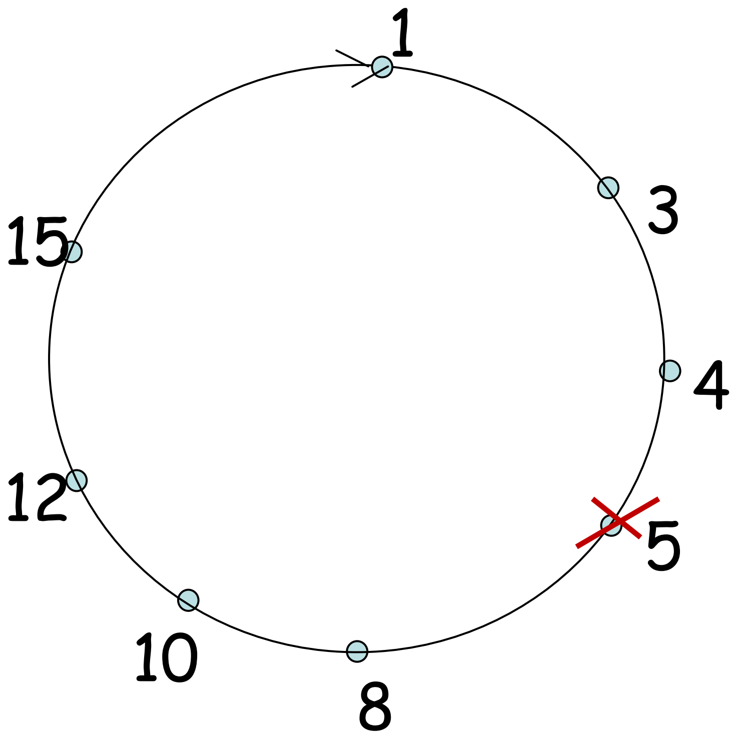
Define <u>closest</u> as closest successor

Circular DHT with Shortcuts



- Each peer keeps track of IP addresses of predecessor, successor, short cuts.
- Reduced from 6 to 2 messages.
- Possible to design shortcuts so O(log N) neighbors, O(log N) messages in query

Peer Churn

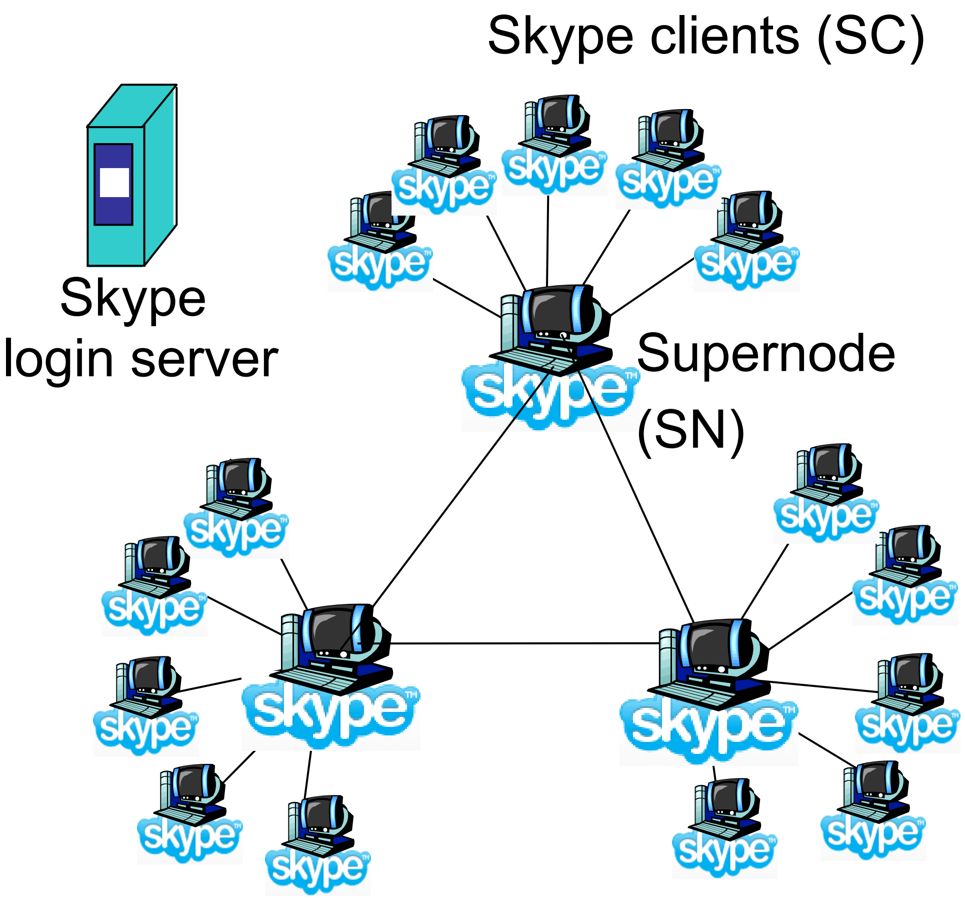


- To handle peer churn, require each peer to know the IP address of its two successors.
- Each peer periodically pings its two successors to see if they are still alive.

- Peer 5 abruptly leaves
- Peer 4 detects; makes 8 its immediate successor; asks 8 who its immediate successor is; makes 8's immediate successor its second successor.
- What if peer 13 wants to join?

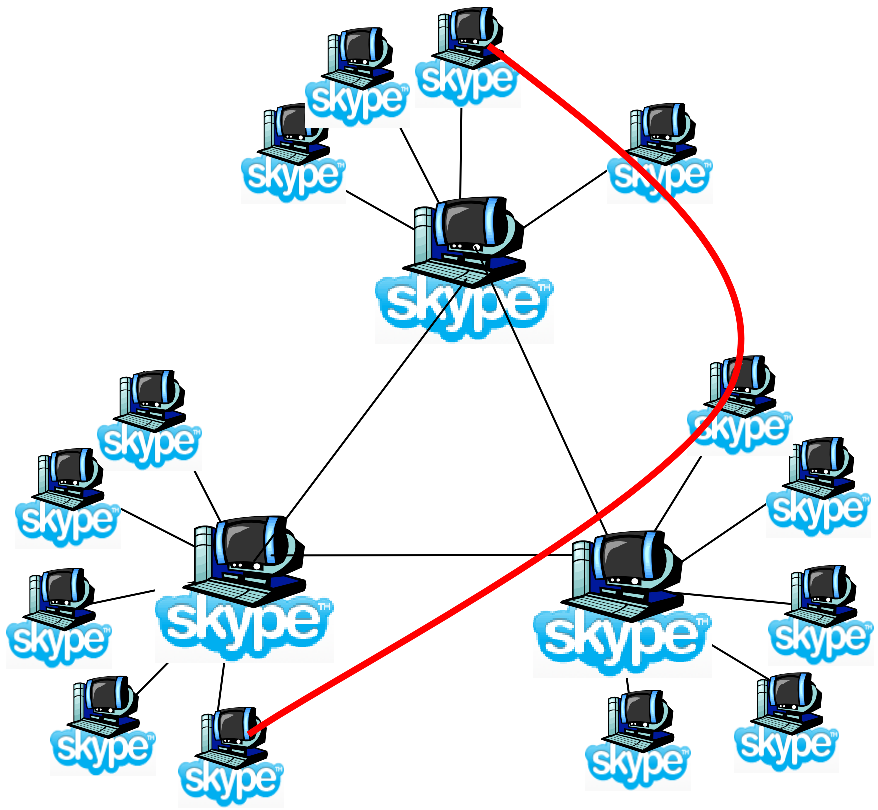
P2P Case study: Skype

- inherently P2P: pairs of users communicate.
- proprietary application-layer protocol (inferred via reverse engineering)
- hierarchical overlay with 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

- Principles of network applications
- Web and HTTP
- DNS
- P2P applications
- **Socket programming with TCP**
- Socket programming with UDP

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

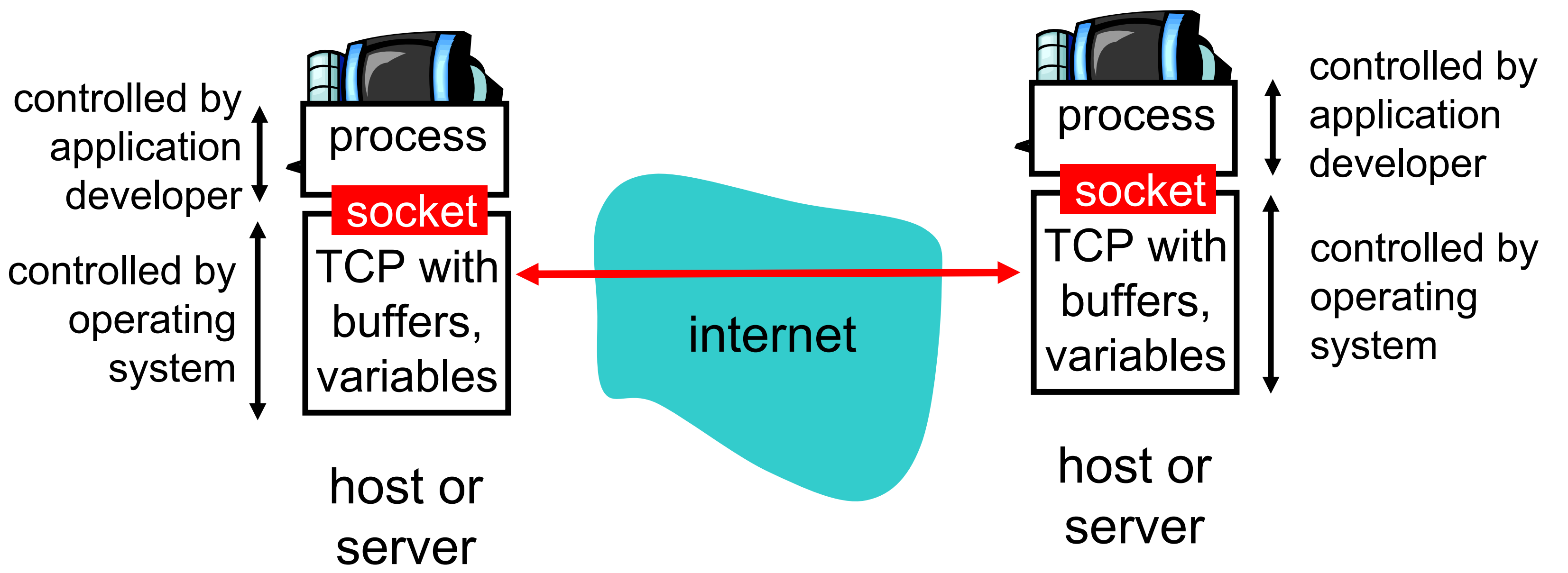
socket

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:

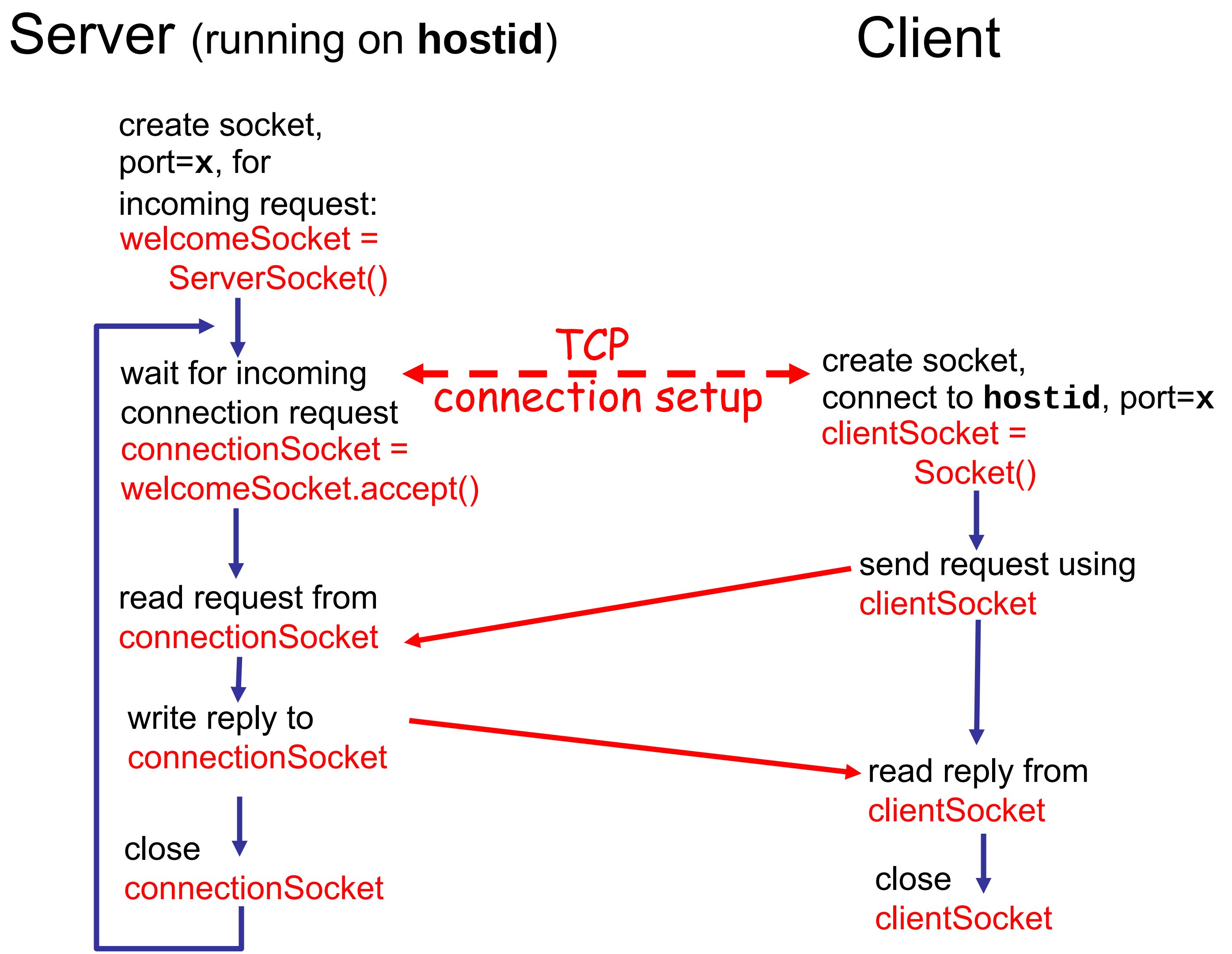
- creating client-local TCP socket
- specifying IP address, port number of server process
- When client creates socket: client TCP establishes connection to server TCP

- 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

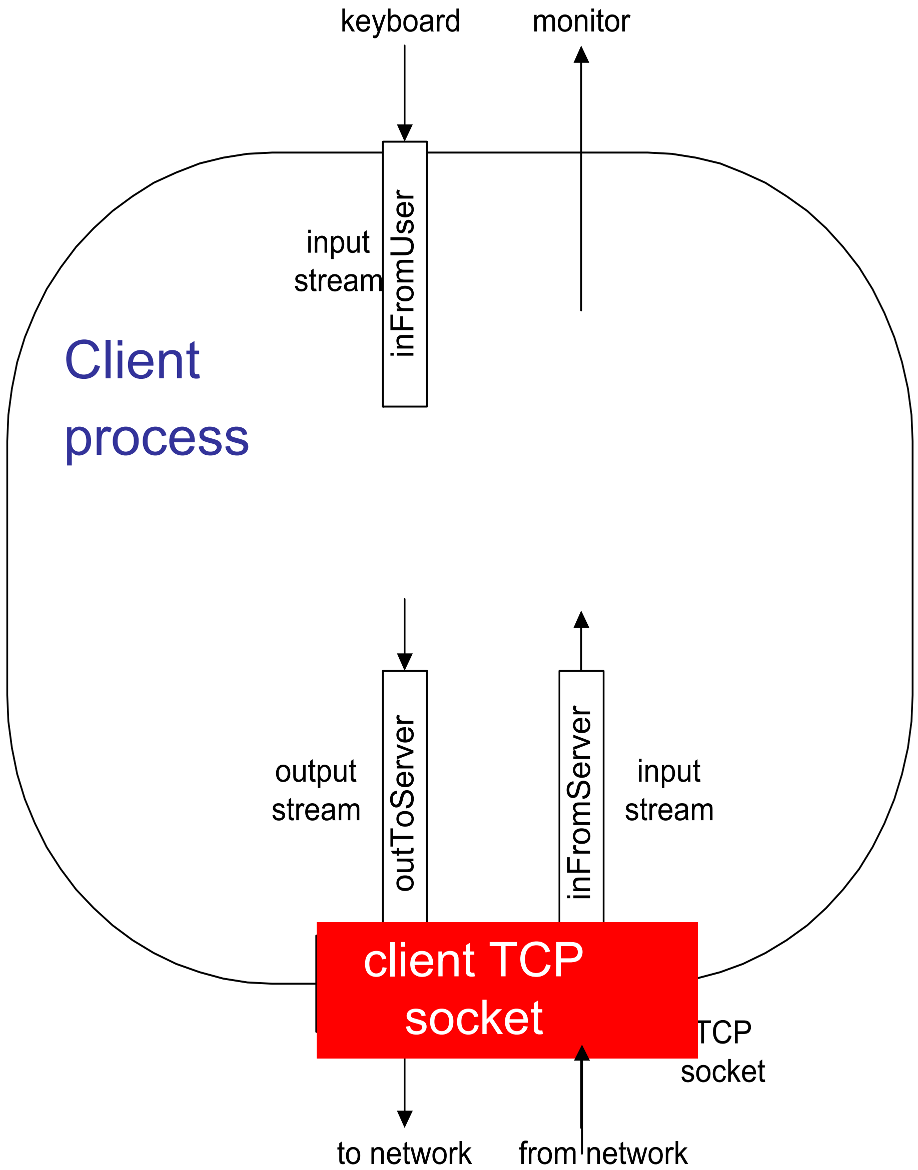
TCP provides reliable, in-order transfer of bytes ("pipe") between client and server

Client/server socket interaction: TCP



Stream Jargon

- A stream is a sequence of characters that flow into or out of a process.
- An input stream is attached to some input source for the process, e.g., keyboard or socket.
- An output stream is attached to an output source, e.g., monitor or socket.



Socket programming with TCP

Example client-server app:

1) client reads line from standard input (`inFromUser` stream) , sends to server via socket (`outToServer` stream)
2) server reads line from socket
3) server converts line to uppercase, sends back to client
4) client reads, prints modified line from socket (`inFromServer` stream)

Example: Java client (TCP)

```
import java.io.*;
import java.net.*;
class TCPClient {

    public static void main(String argv[]) throws Exception
    {
        String sentence;
        String modifiedSentence;

        BufferedReader inFromUser =
          new BufferedReader(new InputStreamReader(System.in));

        Socket clientSocket = new Socket("hostname", 6789);

        DataOutputStream outToServer =
          new DataOutputStream(clientSocket.getOutputStream());
```

Create input stream

Create client socket, connect to server

Create output stream attached to socket

Example: Java client (TCP), cont.

Create input stream attached to socket

Send line to server

Read line from server

```
BufferedReader inFromServer =
  new BufferedReader(new
  InputStreamReader(clientSocket.getInputStream()));

sentence = inFromUser.readLine();

outToServer.writeBytes(sentence + '\n');

modifiedSentence = inFromServer.readLine();

System.out.println("FROM SERVER: " + modifiedSentence);

clientSocket.close();

  }
}
```

Example: Java server (TCP)

```
import java.io.*;
import java.net.*;

class TCPServer {

  public static void main(String argv[]) throws Exception
   {
     String clientSentence;
     String capitalizedSentence;
```

Create welcoming socket at port 6789

```
     ServerSocket welcomeSocket = new ServerSocket(6789);

     while(true) {
```

Wait, on welcoming socket for contact by client

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

Create input stream, attached to socket

```
       BufferedReader inFromClient =
         new BufferedReader(new
         InputStreamReader(connectionSocket.getInputStream()));
```

Example: Java server (TCP), cont

Create output stream, attached to socket

Read in line from socket

Write out line to socket

```
DataOutputStream  outToClient =
  new DataOutputStream(connectionSocket.getOutputStream());

clientSentence = inFromClient.readLine();

capitalizedSentence = clientSentence.toUpperCase() + '\n';

outToClient.writeBytes(capitalizedSentence);
}
}
}
```

End of while loop, loop back and wait for another client connection

- Principles of network applications
- Web and HTTP
- DNS
- P2P applications
- Socket programming with TCP
- **Socket programming with UDP**

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.
serverSocket =
DatagramSocket()

read datagram from
serverSocket

write reply to
serverSocket
specifying
client address,
port number

Client

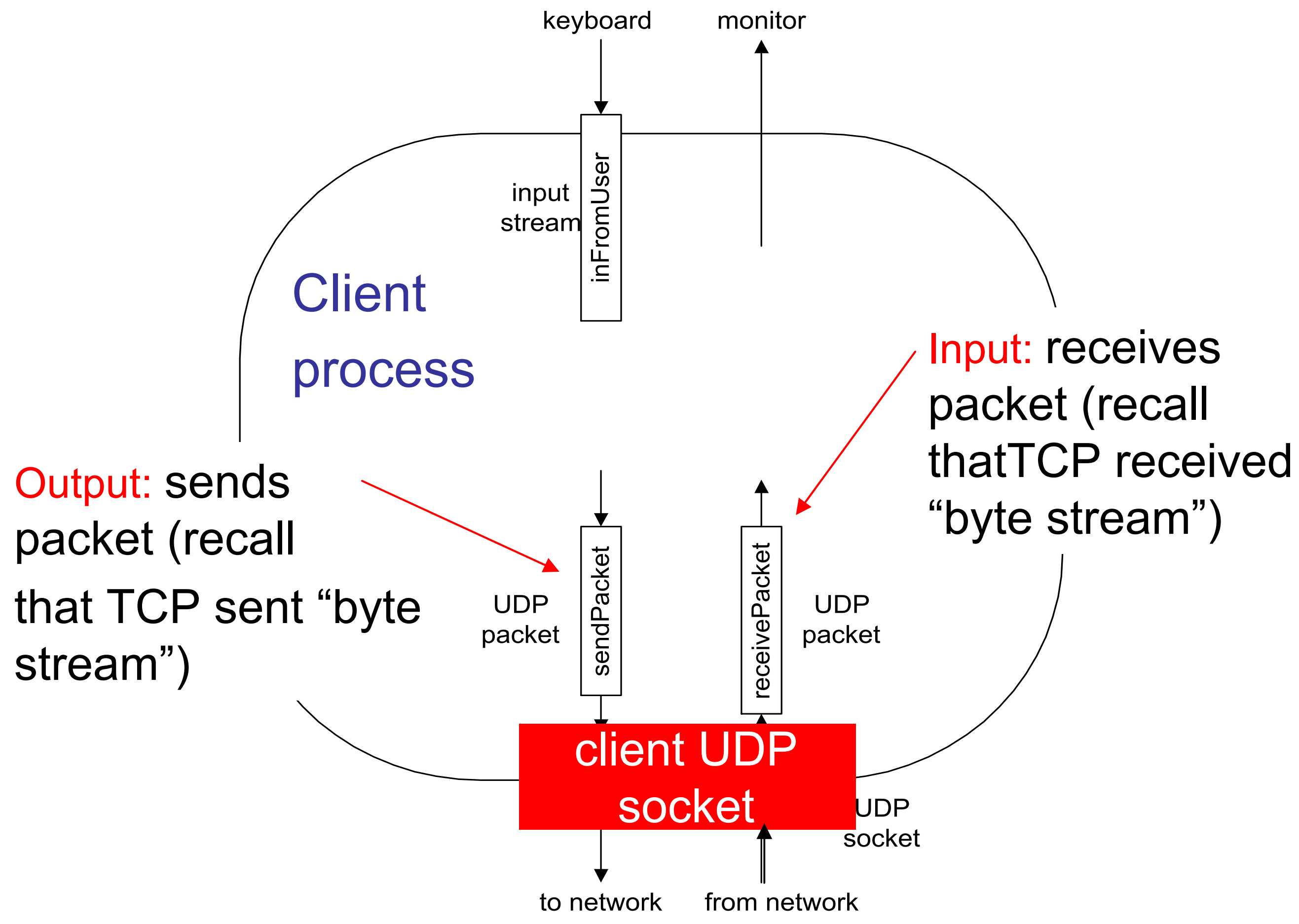
create socket,
clientSocket =
DatagramSocket()

Create datagram with server IP and
port=x; send datagram via
clientSocket

read datagram from
clientSocket

close
clientSocket

Example: Java client (UDP)



Example: Java client (UDP)

Create input stream

Create client socket

Translate hostname to IP address using DNS

```
import java.io.*;
import java.net.*;

class UDPClient {
    public static void main(String args[]) throws Exception
    {

      BufferedReader inFromUser =
        new BufferedReader(new InputStreamReader(System.in));

      DatagramSocket clientSocket = new DatagramSocket();

      InetAddress IPAddress = InetAddress.getByName("hostname");

      byte[] sendData = new byte[1024];
      byte[] receiveData = new byte[1024];

      String sentence = inFromUser.readLine();

      sendData = sentence.getBytes();
```

Example: Java client (UDP), cont.

Create datagram with data-to-send, length, IP addr, port

Send datagram to server

Read datagram from server

```
DatagramPacket sendPacket =
   new DatagramPacket(sendData, sendData.length, IPAddress, 9876);

clientSocket.send(sendPacket);

DatagramPacket receivePacket =
   new DatagramPacket(receiveData, receiveData.length);

clientSocket.receive(receivePacket);

String modifiedSentence =
    new String(receivePacket.getData());

System.out.println("FROM SERVER:" + modifiedSentence);
clientSocket.close();
}
}
```

Example: Java server (UDP)

Create datagram socket at port 9876

Create space for received datagram

Receive datagram

```
import java.io.*;
import java.net.*;

class UDPServer {
  public static void main(String args[]) throws Exception
    {

      DatagramSocket serverSocket = new DatagramSocket(9876);

      byte[] receiveData = new byte[1024];
      byte[] sendData  = new byte[1024];

      while(true)
        {

          DatagramPacket receivePacket =
             new DatagramPacket(receiveData, receiveData.length);
           serverSocket.receive(receivePacket);
```

Example: Java server (UDP), cont

Get IP addr port #, of sender

Create datagram to send to client

Write out datagram to socket

End of while loop, loop back and wait for another datagram

```
        String sentence = new String(receivePacket.getData());

        InetAddress IPAddress = receivePacket.getAddress();

        int port = receivePacket.getPort();

                String capitalizedSentence = sentence.toUpperCase();

        sendData = capitalizedSentence.getBytes();

        DatagramPacket sendPacket =
          new DatagramPacket(sendData, sendData.length, IPAddress,
                             port);

        serverSocket.send(sendPacket);
      }
    }
}
```

Chapter 2: Summary

our study of network apps now finished!

- 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
  - P2P: BitTorrent, Skype
- 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
- *Important themes:*
- control vs. data msgs
  - in-band, out-of-band
- centralized vs. decentralized
- stateless vs. stateful
- reliable vs. unreliable msg transfer
- "complexity at network edge"

Master Course
Computer Networks
IN2097

Prof. Dr.-Ing. Georg Carle
Christian Grothoff, Ph.D.
Lecturer today: Dr. Nils Kammenhuber

Chair for Network Architectures and Services
Institut für Informatik
Technische Universität München
http://www.net.in.tum.de

Chapter 3: Transport Layer

Our goals:

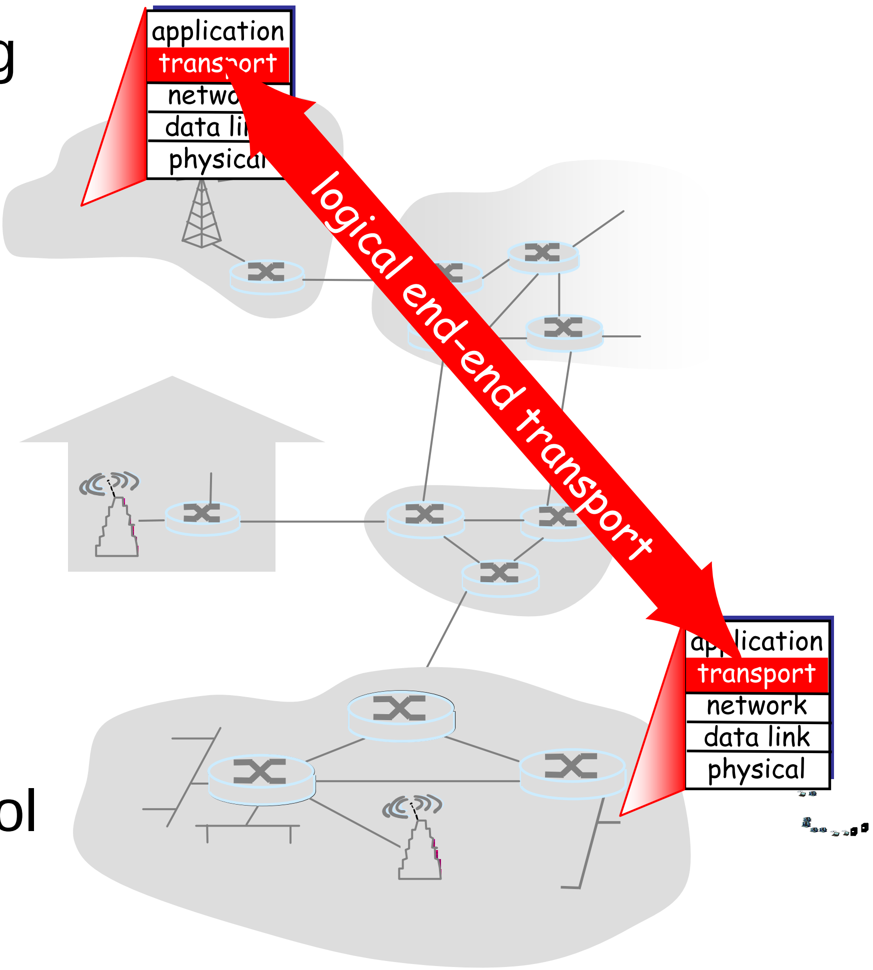
- understand principles behind transport layer services:
  - multiplexing/demultiplexing
  - reliable data transfer
  - flow control
  - congestion control
- learn about transport layer protocols in the Internet:
  - UDP: connectionless transport
  - TCP: connection-oriented transport
  - TCP congestion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

Chapter 3 outline

- **3.1 Transport-layer services**
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 largely omitted
- 3.5 Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- 3.6 largely omitted
- 3.7 TCP congestion control

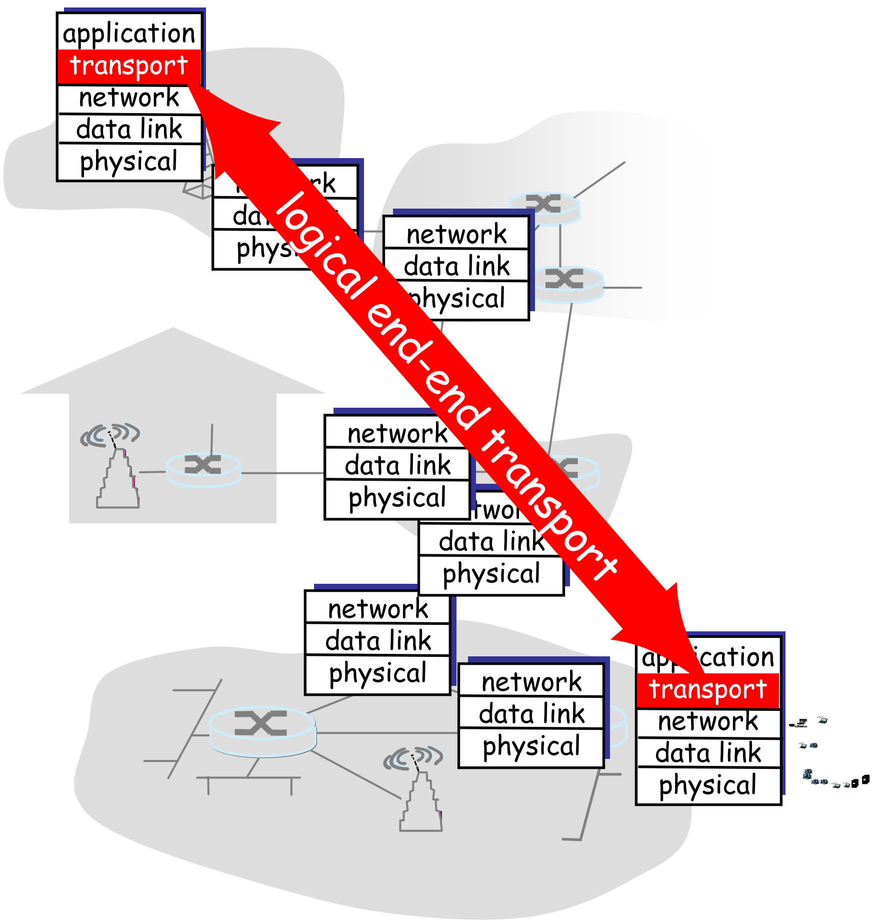
Transport services and protocols

- 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
  - rcv side: reassembles segments into messages, passes to app layer
- more than one transport protocol available to apps
  - Internet: TCP and UDP



Internet transport-layer protocols

- reliable, in-order delivery (TCP)
  - congestion control
  - flow control
  - connection setup
- unreliable, unordered delivery: UDP
  - no-frills extension of "best-effort" IP
- services not available:
  - delay guarantees
  - bandwidth guarantees



Chapter 3 outline

- 3.1 Transport-layer services
- **3.2 Multiplexing and demultiplexing**
- 3.3 Connectionless transport: UDP
- -
- 3.5 Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- -
- 3.7 TCP congestion control

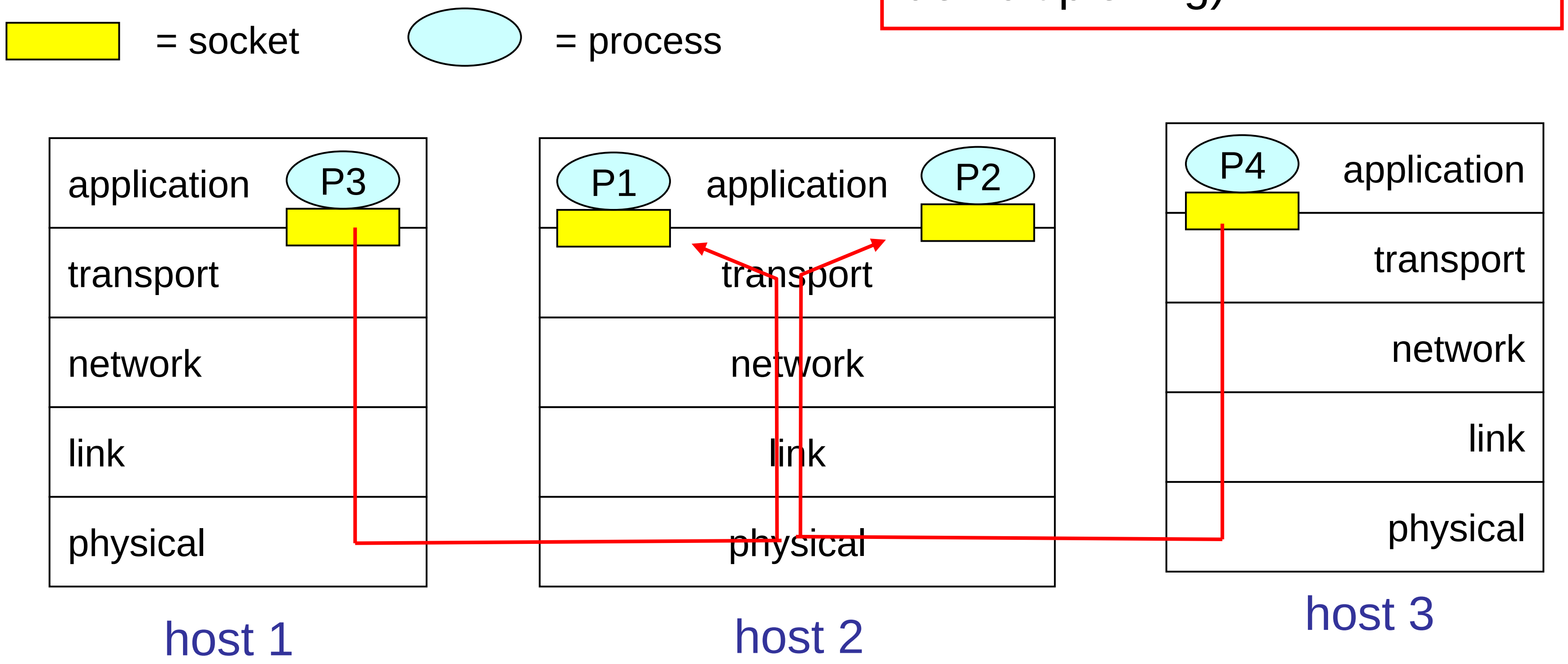
Multiplexing/demultiplexing

Demultiplexing at rcv host:

delivering received segments to correct socket

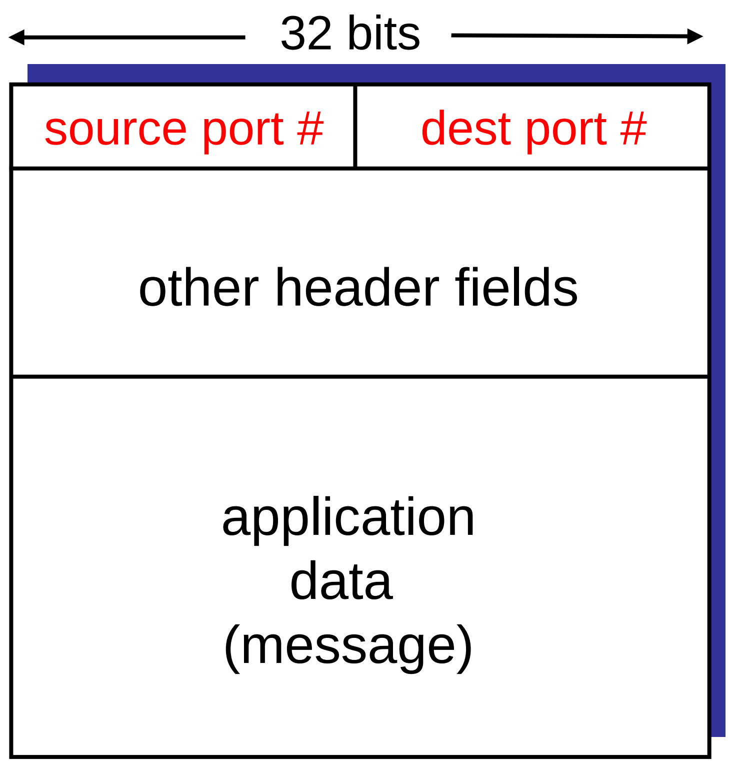
Multiplexing at send host:

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



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

Connectionless demultiplexing

- Create sockets with port numbers:

```
DatagramSocket mySocket1 = new DatagramSocket(12534);
DatagramSocket mySocket2 = new DatagramSocket(12535);
```

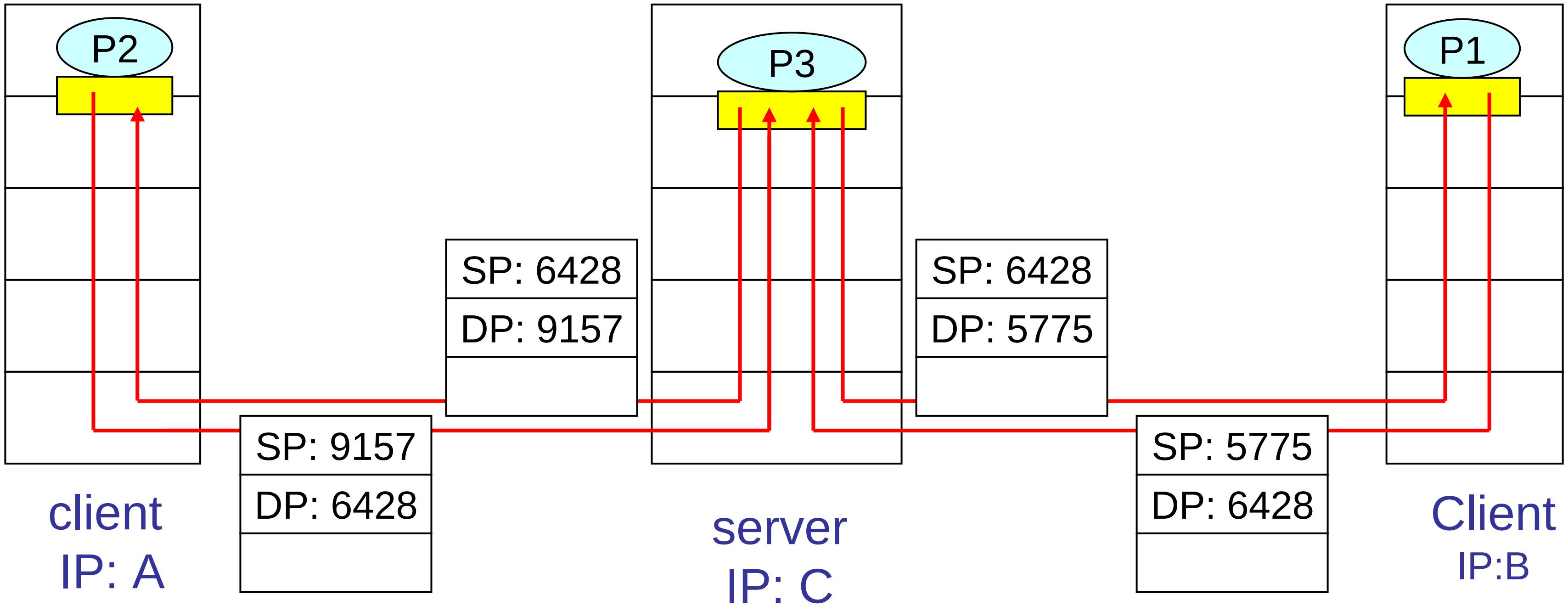
- UDP socket identified by two-tuple:

  (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 directed to same socket

Connectionless demux (cont)

```
DatagramSocket serverSocket = new DatagramSocket(6428);
```

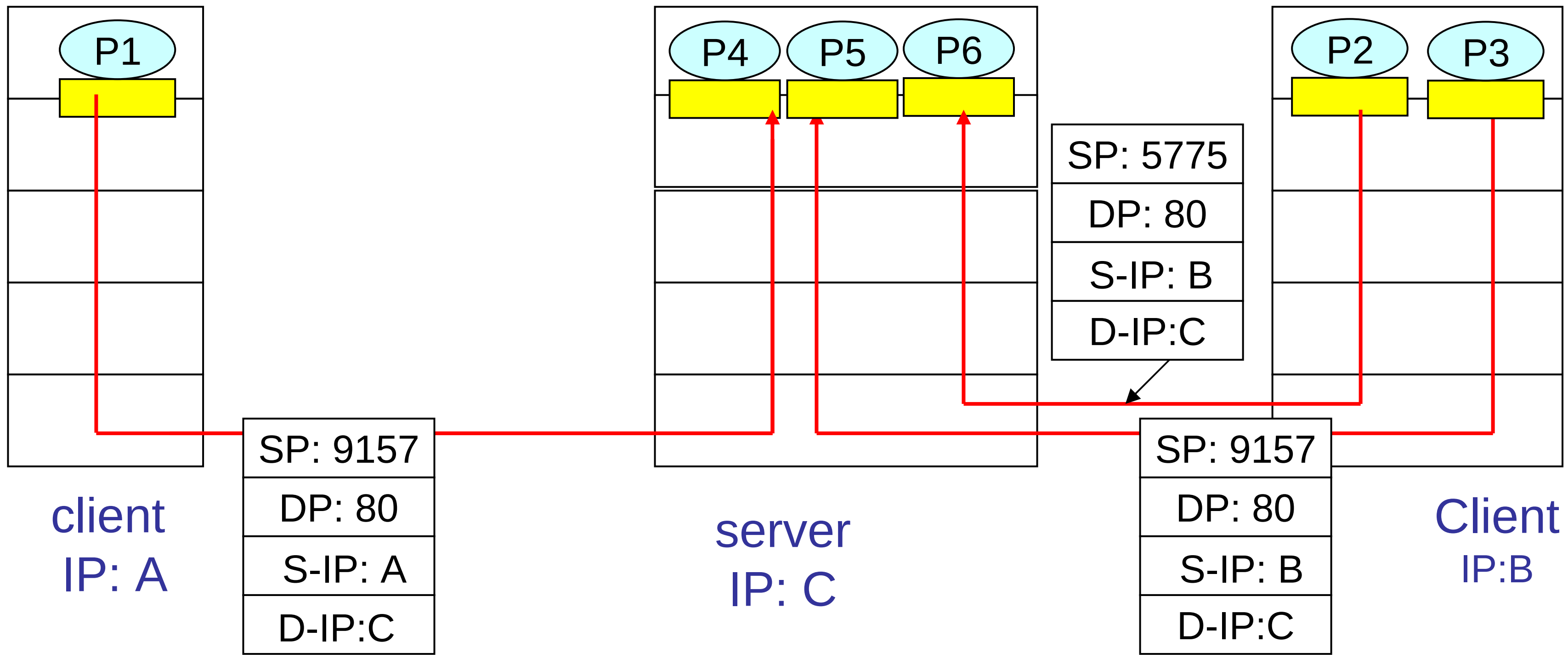


Source Port (SP) provides “return address”

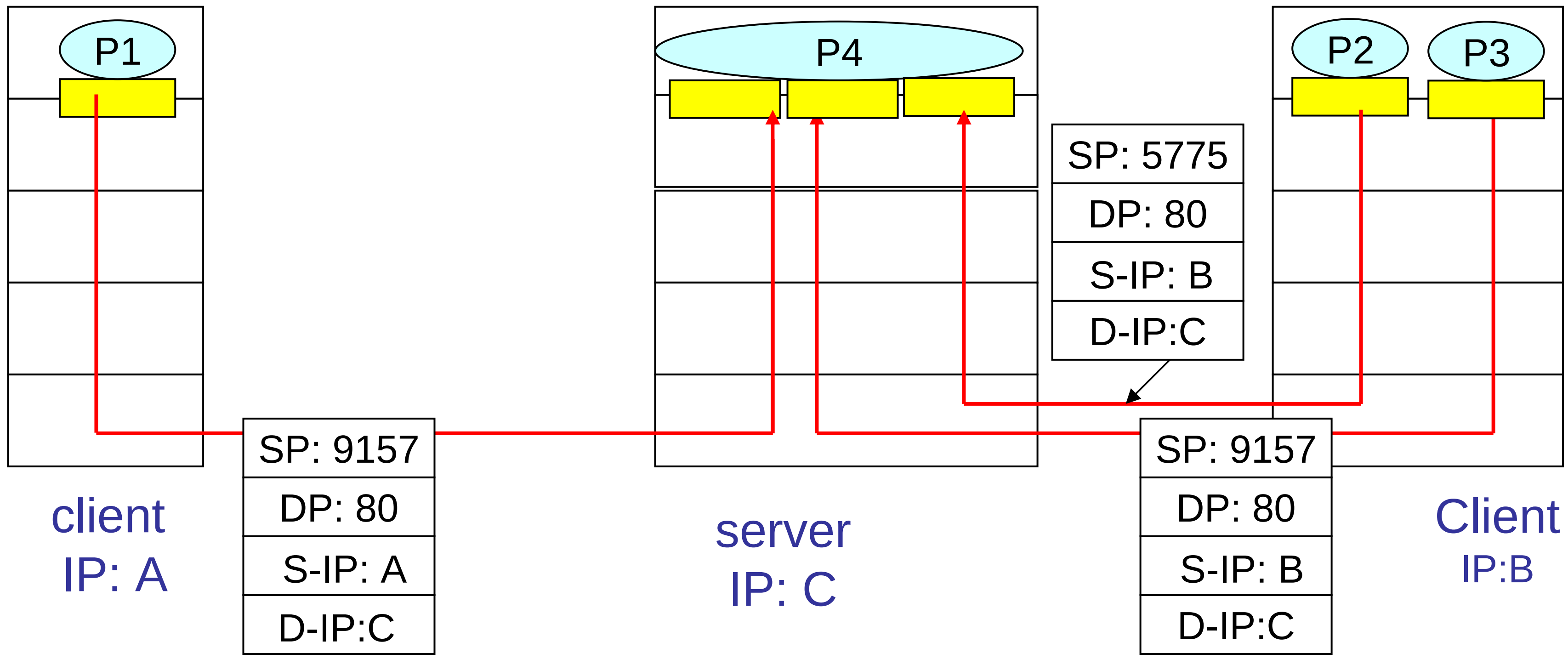
Connection-oriented demux

- 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
- 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 (cont)



Connection-oriented demux: Threaded Web Server



Chapter 3 outline

- 3.1 Transport-layer services
- 3.2 Multiplexing and demultiplexing
- **3.3 Connectionless transport: UDP**
- -
- 3.5 Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- -
- 3.7 TCP congestion control

UDP: User Datagram Protocol [RFC 768]

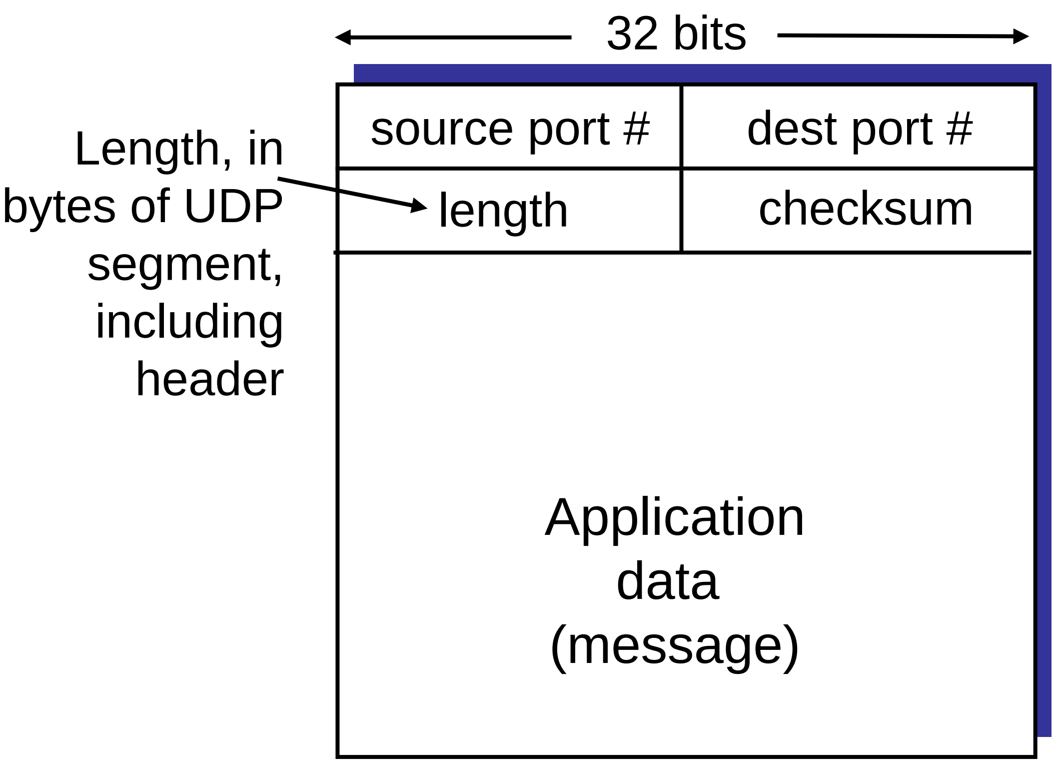
- "no frills," "bare bones" Internet transport protocol
- "best effort" service, UDP segments may be:
  - lost
  - delivered out of order to app
- *connectionless:*
  - no handshaking between UDP sender, receiver
  - each UDP segment handled independently of others

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: more

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



UDP segment format

UDP checksum

Goal: detect "errors" (e.g., flipped bits) in transmitted segment

Sender:

- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

Receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - NO - error detected
  - YES - no error detected. *But maybe errors nonetheless?* More later ….

Internet Checksum Example

- Note
  - When adding numbers, a carryout from the most significant bit needs to be added to the result
- Example: add two 16-bit integers

		1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
		1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
wraparound	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
sum		1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum		0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

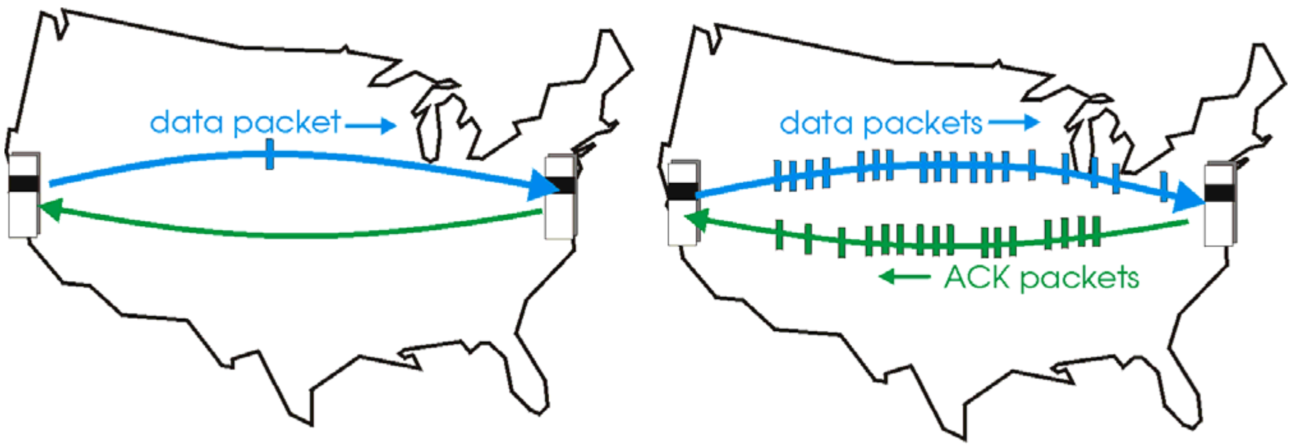
Chapter 3 outline

- 3.1 Transport-layer services
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- **3.4 Principles of reliable data transfer**
- 3.5 Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- 3.6 Principles of congestion control
- 3.7 TCP congestion control

Pipelined protocols

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

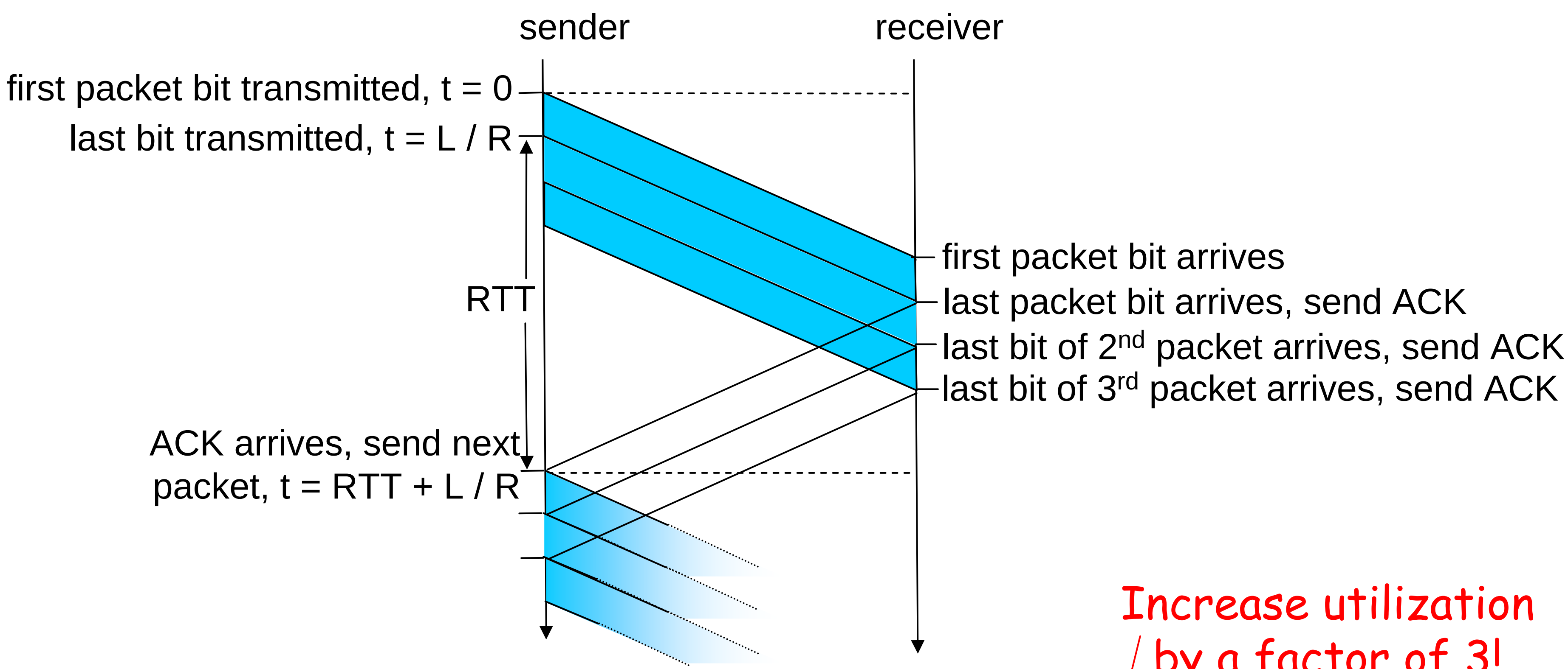
- range of sequence numbers must be increased
- buffering at sender and/or receiver



(a) a stop-and-wait protocol in operation (b) a pipelined protocol in operation

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

Pipelining: increased utilization



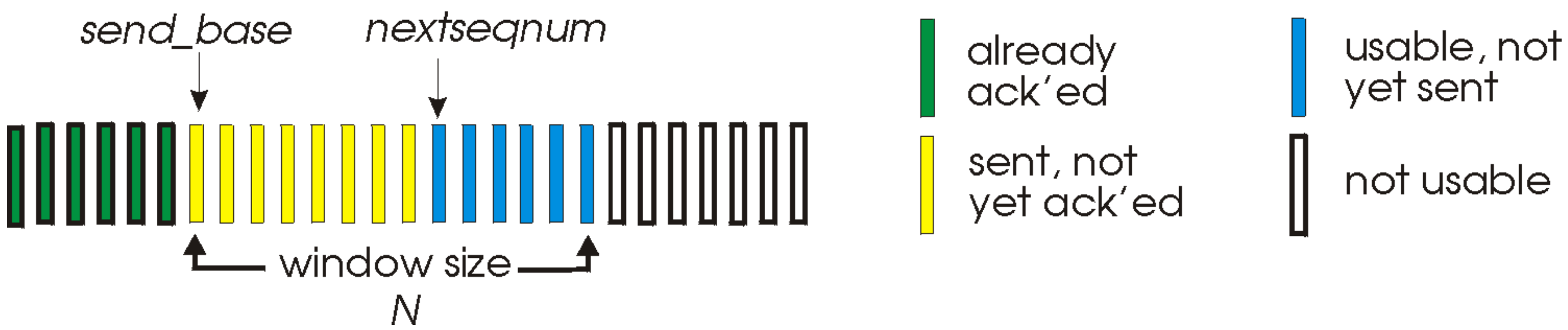
Increase utilization by a factor of 3!

$$U_{sender} = \frac{3 * L / R}{RTT + L / R} = \frac{.024}{30.008} = 0.0008$$

Go-Back-N

Sender:

- k-bit seq # in pkt header
- "window" of up to N, consecutive unack'ed pkts allowed



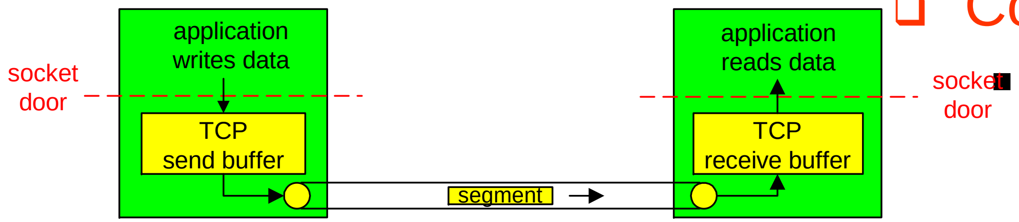
- ACK(n): ACKs all pkts up to, including seq # n - "cumulative ACK"
  - may receive duplicate ACKs (see receiver)
- timer for each in-flight pkt
- *timeout(n):* retransmit pkt n and all higher seq # pkts in window

Chapter 3 outline

- 3.1 Transport-layer services
- **3.2 Multiplexing and demultiplexing**
- 3.3 Connectionless transport: UDP
- 3.4 Principles of reliable data transfer
- **3.5 Connection-oriented transport: TCP**
  - **segment structure**
  - **reliable data transfer**
  - **flow control**
  - **connection management**
- 3.6 Principles of congestion control
- 3.7 TCP congestion control

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
- Congestion controlled:
  - Will not overwhelm network

TCP segment structure

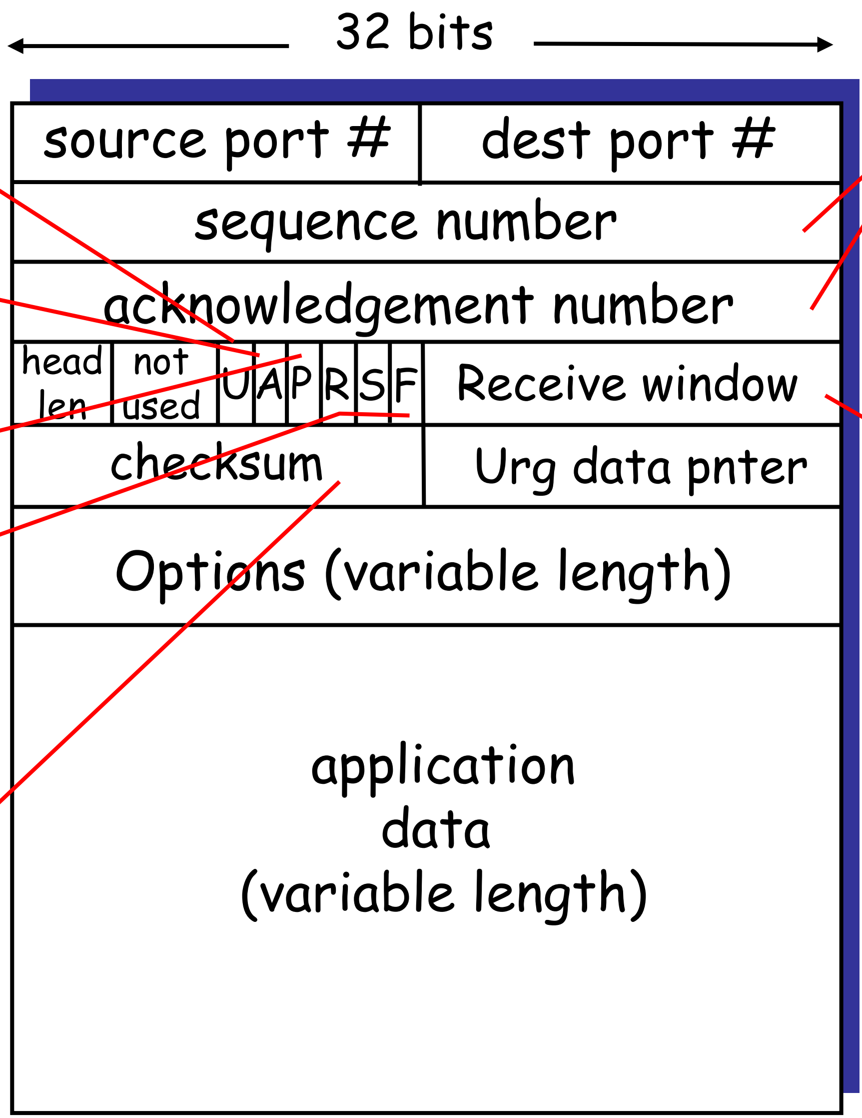
URG: urgent data
(generally not used)

ACK: ACK #
valid

PSH: push data now
(generally not used)

RST, SYN, FIN:
connection estab
(setup, teardown
commands)

Internet
checksum
(as in UDP)



counting
by bytes
of data
(not segments!)

bytes
rcvr willing
to accept

TCP seq. #'s and ACKs

Seq. #'s:

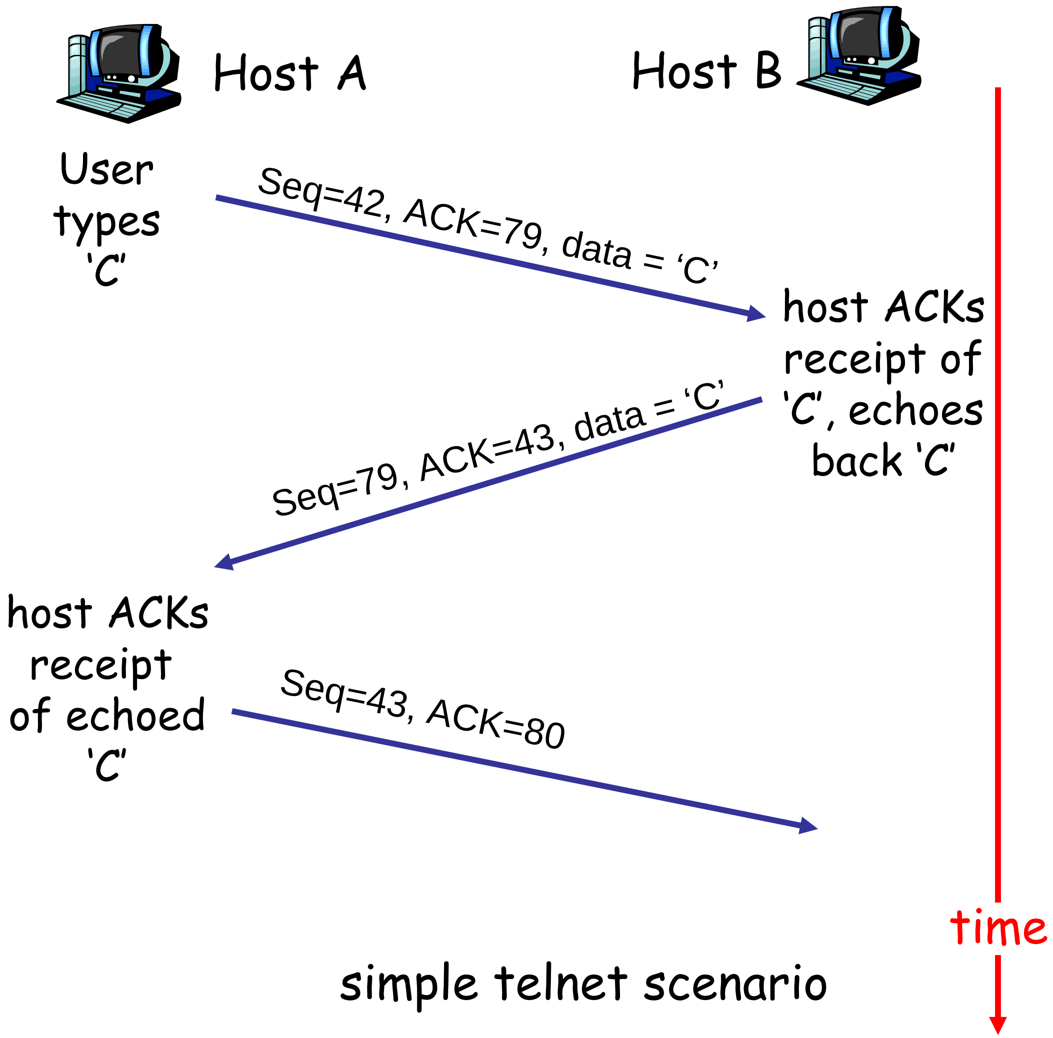
- byte stream "number" of first byte in segment's data

ACKs:

- seq # of next byte expected from other side
- cumulative ACK

Q: how receiver handles out-of-order segments

- A: TCP spec doesn't say, - up to implementor



simple telnet scenario

TCP Round Trip Time and Timeout

Q: how to set TCP timeout value?

- longer than RTT
  - but RTT varies
- too short: premature timeout
  - unnecessary retransmissions
- too long: slow reaction to segment loss

Q: how to estimate RTT?

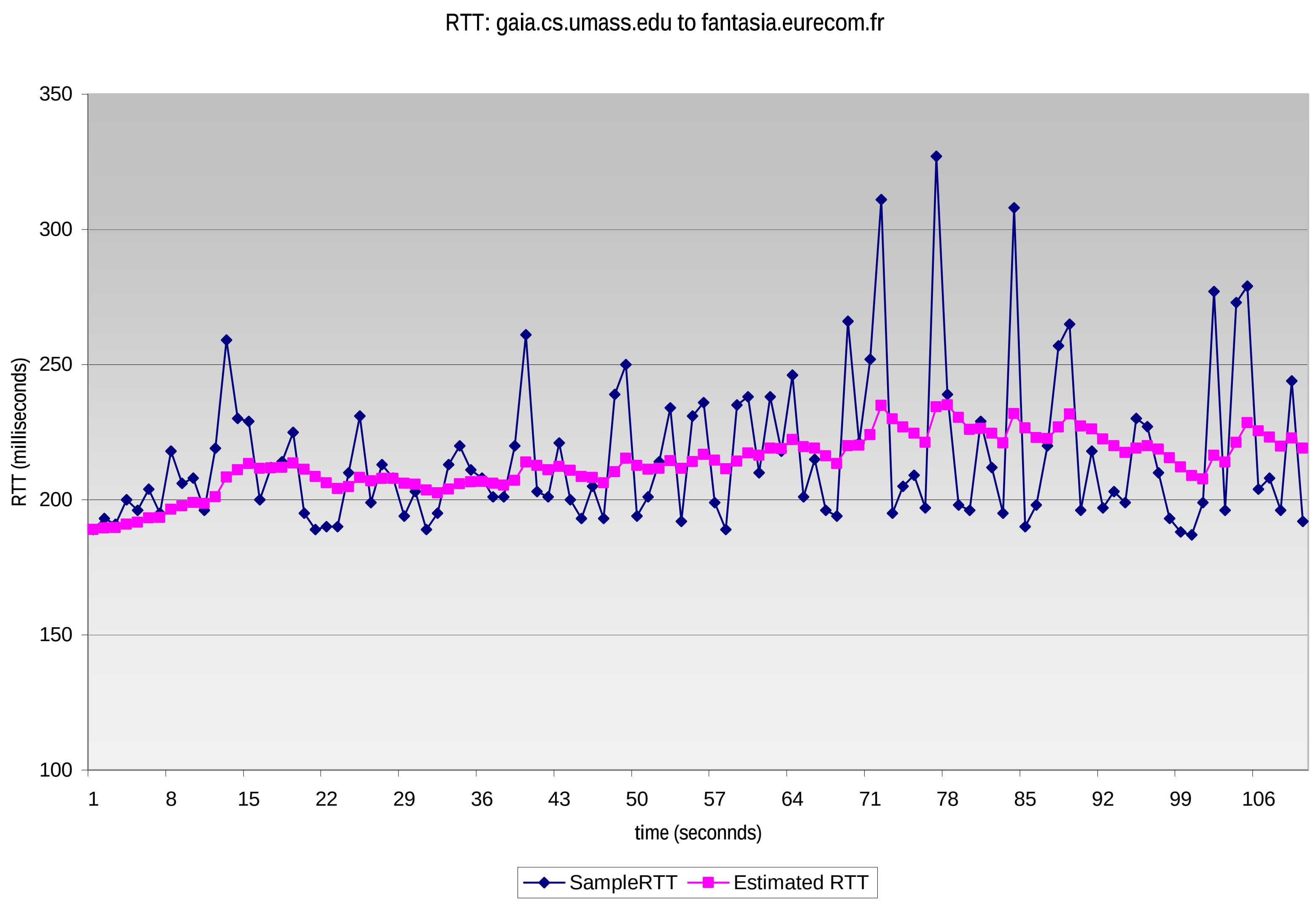
- `SampleRTT`: measured time from segment transmission until ACK receipt
  - ignore retransmissions
- **`SampleRTT`** will vary, want estimated RTT “smoother”
  - average several recent measurements, not just current **`SampleRTT`**

TCP Round Trip Time and Timeout

```
EstimatedRTT = (1- α)*EstimatedRTT + α*SampleRTT
```

- Exponential weighted moving average
- influence of past sample decreases exponentially fast
- typical value: α = 0.125

Example RTT estimation:



TCP Round Trip Time and Timeout

Setting the timeout

- **`EstimtedRTT`** plus "safety margin"
  - large variation in **`EstimatedRTT ->`** larger safety margin
- first estimate of how much SampleRTT deviates from EstimatedRTT:

$$\texttt{DevRTT} = (1-\beta)*\texttt{DevRTT} + \beta*|\texttt{SampleRTT-EstimatedRTT}|$$

(typically, $\beta = 0.25$)

Then set timeout interval:

```
TimeoutInterval = EstimatedRTT + 4*DevRTT
```

Chapter 3 outline

- 3.1 Transport-layer services
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Principles of reliable data transfer
- 3.5 Connection-oriented transport: TCP
  - segment structure
  - **reliable data transfer**
  - flow control
  - connection management
- 3.6 Principles of congestion control
- 3.7 TCP congestion control

TCP reliable data transfer

- TCP creates rdt service on top of IP's unreliable service
- Pipelined segments
- Cumulative acks
- TCP uses single retransmission timer

- Retransmissions are triggered by:
  - timeout events
  - duplicate acks
- Initially consider simplified TCP sender:
  - ignore duplicate acks
  - ignore flow control, congestion control

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 acknowledges previously unacked segments
  - update what is known to be acked
  - start timer if there are outstanding segments

TCP sender (simplified)

```
NextSeqNum = InitialSeqNum
SendBase = InitialSeqNum
 loop (forever) {
   switch(event)

   event: data received from application above
        create TCP segment with sequence number NextSeqNum
        if (timer currently not running)
             start timer
        pass segment to IP
        NextSeqNum = NextSeqNum + length(data)

    event: timer timeout
        retransmit not-yet-acknowledged segment with
               smallest sequence number
        start timer

    event: ACK received, with ACK field value of y
        if (y > SendBase) {
             SendBase = y
            if (there are currently not-yet-acknowledged segments)
                   start timer  }
  }  /* end of loop forever */
```

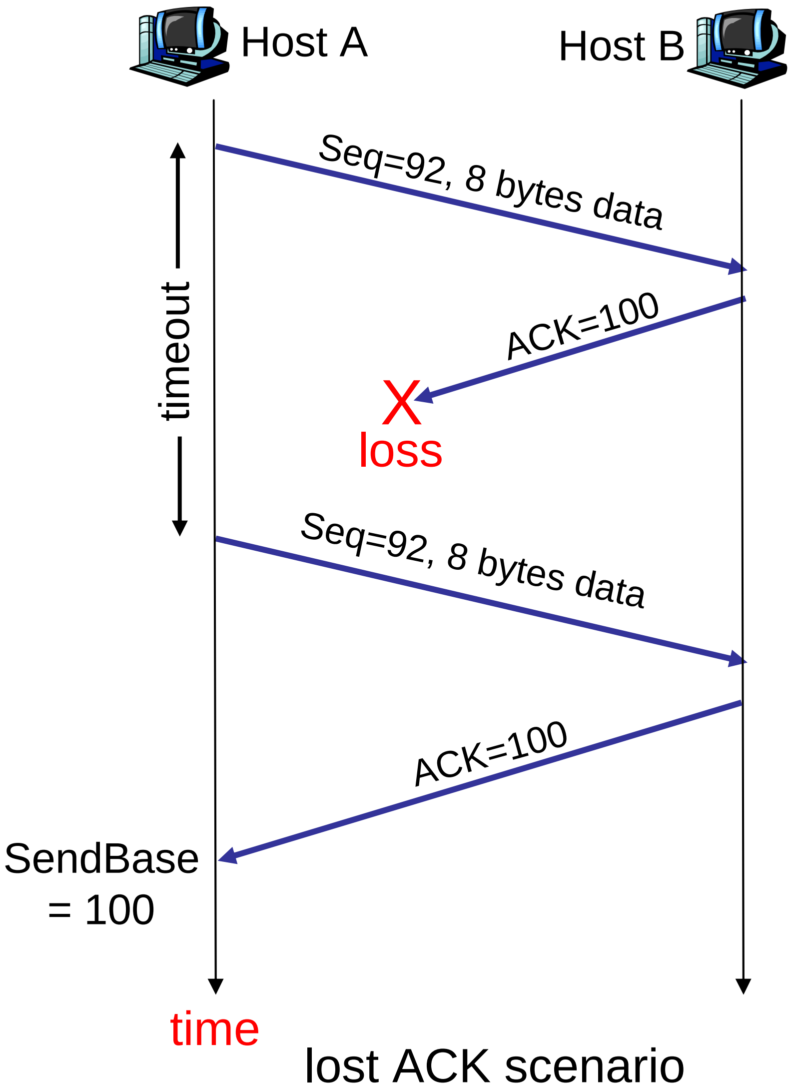
Comment:

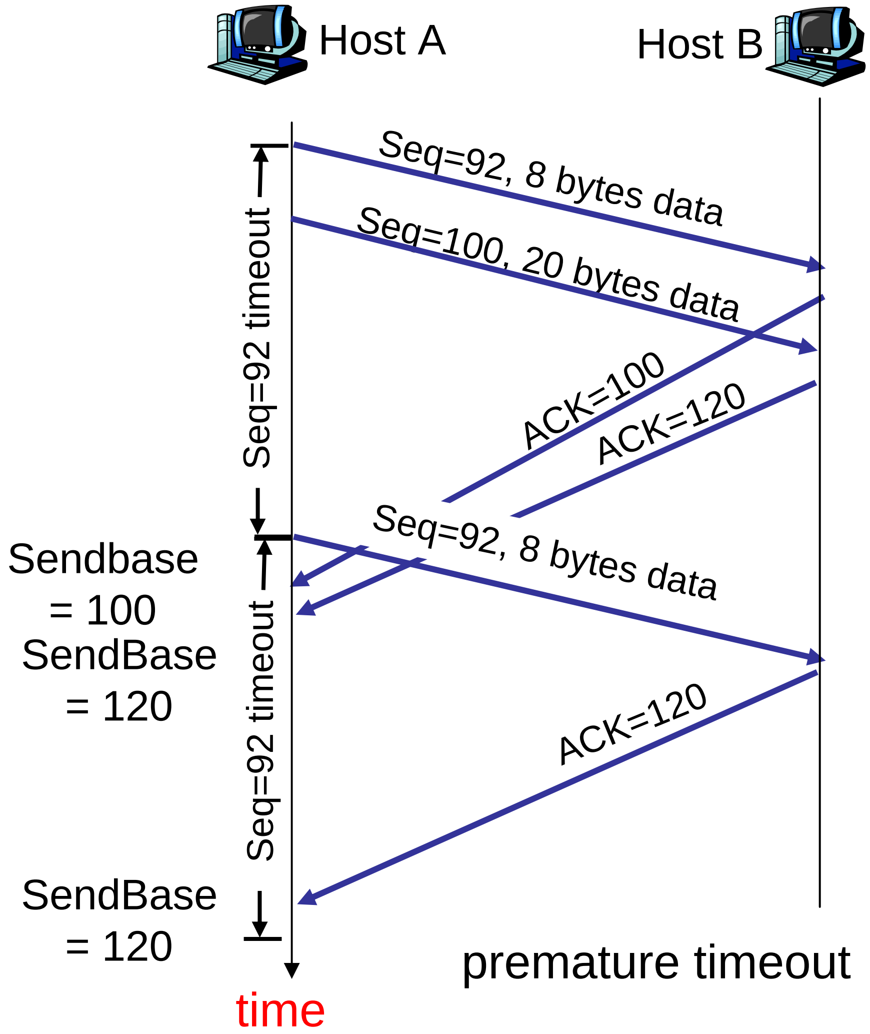
- SendBase-1: last cumulatively ack'ed byte

Example:

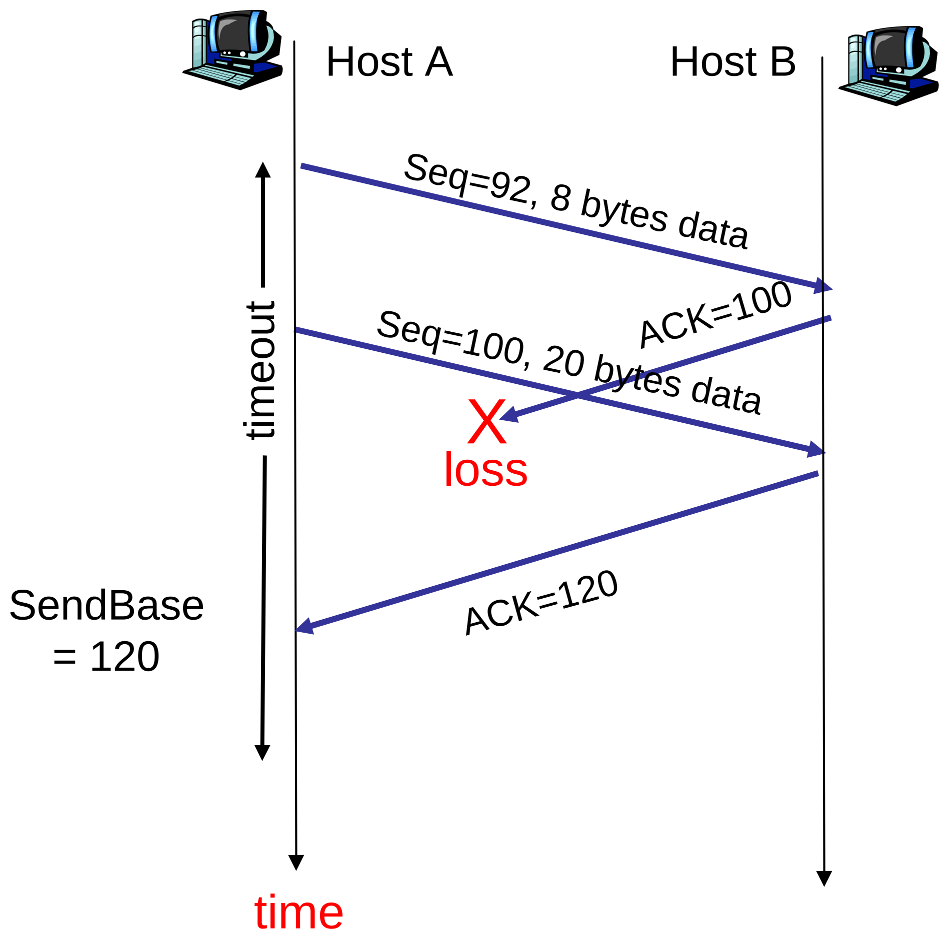
- SendBase-1 = 71; y= 73, so the rcvr wants 73+ ; y > SendBase, so that new data is acked

TCP: retransmission scenarios





TCP retransmission scenarios (more)



Cumulative ACK scenario

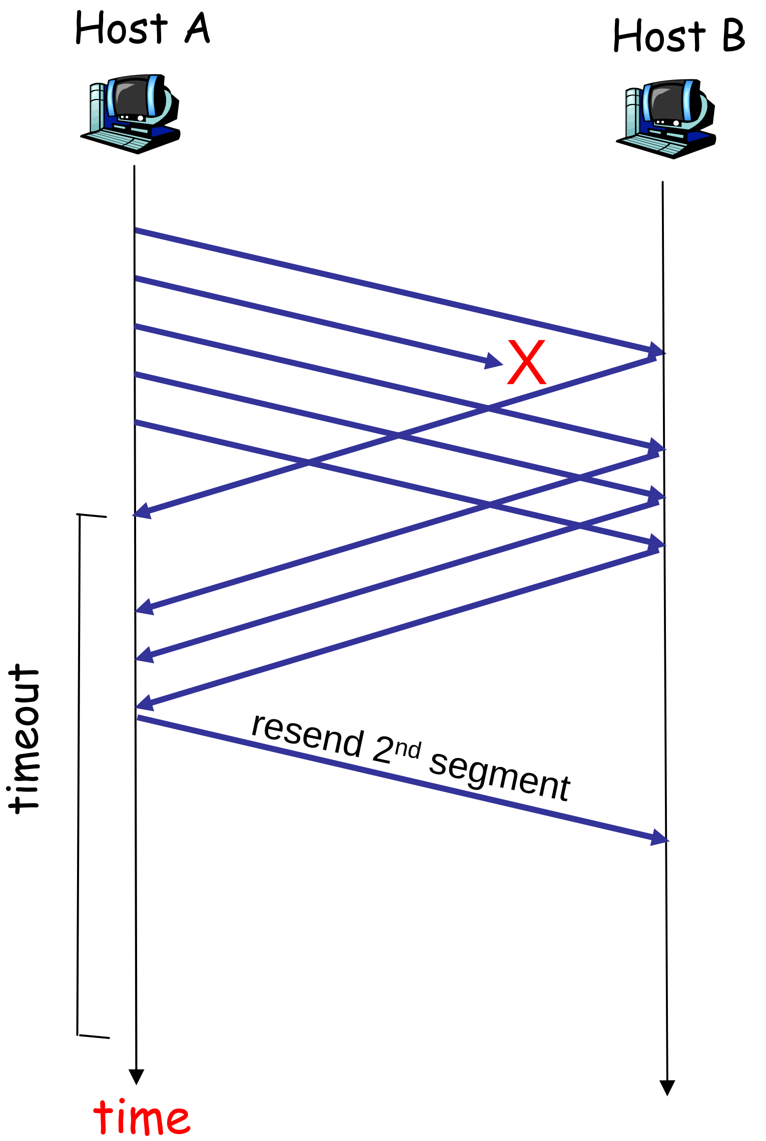
TCP ACK generation [RFC 1122, RFC 2581]

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 *duplicate ACK*, indicating seq. # of next expected byte
Arrival of segment that partially or completely fills gap	Immediate send ACK, provided that segment starts at 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

Resending a segment after triple duplicate ACK



Fast retransmit algorithm:

```
event: ACK received, with ACK field value of y
          if (y > SendBase) {
              SendBase = y
              if (there are currently not-yet-acknowledged segments)
                  start timer
             }
          else {
               increment count of dup ACKs received for y
               if (count of dup ACKs received for y = 3) {
                   resend segment with sequence number y
               }
```

a duplicate ACK for already ACKed segment

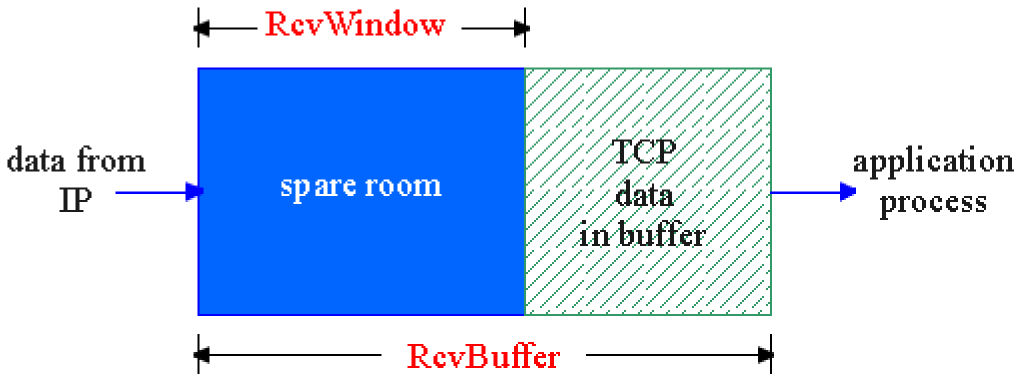
fast retransmit

Chapter 3 outline

- 3.1 Transport-layer services
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Principles of reliable data transfer
- 3.5 Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - **flow control**
  - connection management
- 3.6 Principles of congestion control
- 3.7 TCP congestion control

TCP Flow Control

- receive side of TCP connection has a receive buffer:

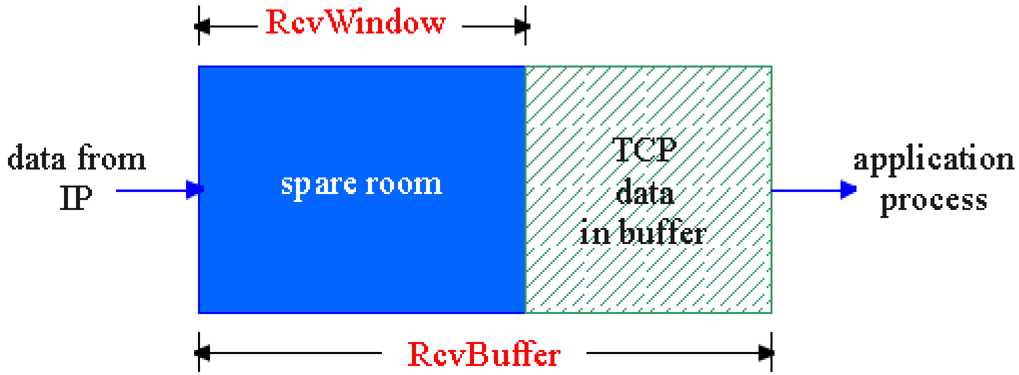


flow control

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

Chapter 3 outline

- 3.1 Transport-layer services
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Principles of reliable data transfer
- 3.5 Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - **connection management**
- 3.6 Principles of congestion control
- 3.7 TCP congestion control

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

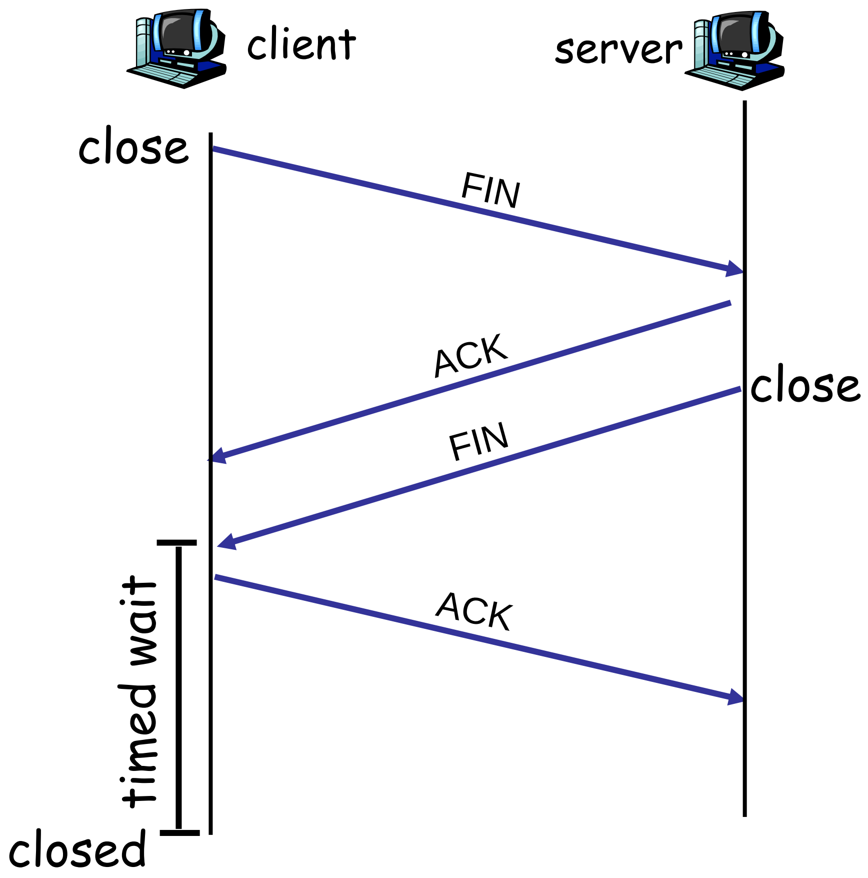
TCP Connection Management (cont.)

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.



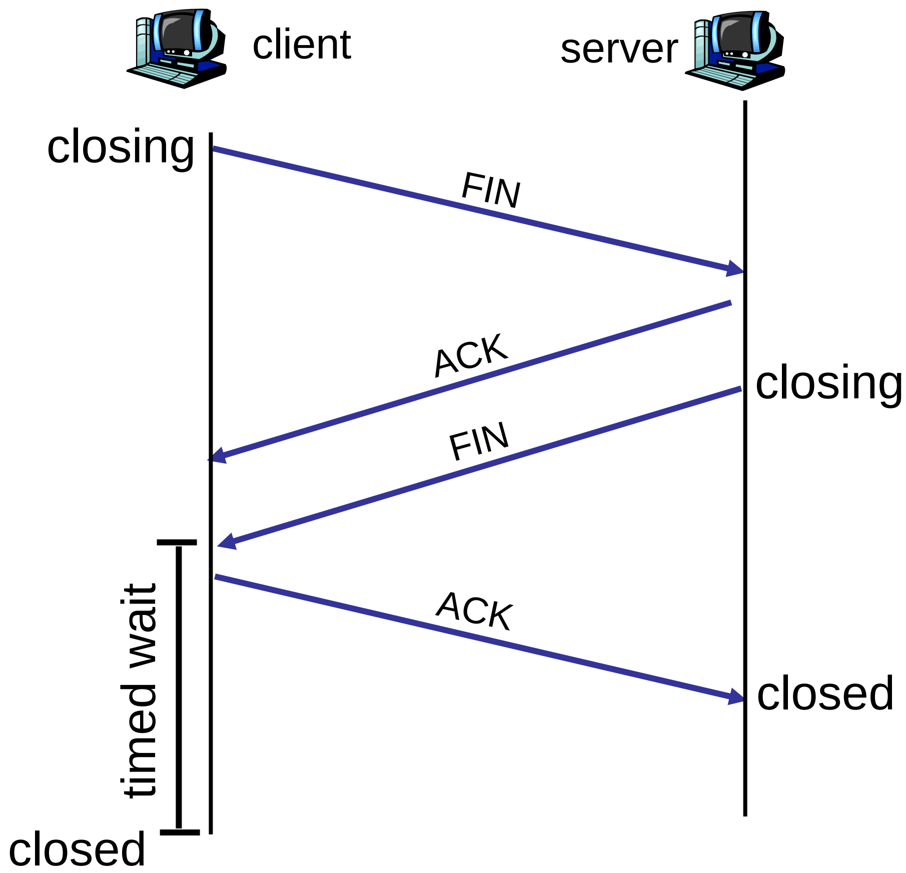
TCP Connection Management (cont.)

Step 3: client receives FIN, replies with ACK.

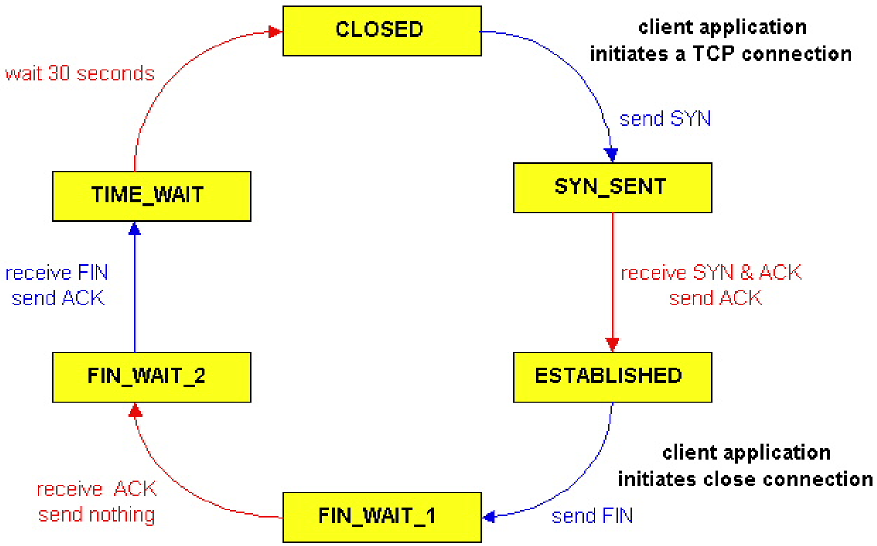
- Enters “timed wait” - will respond with ACK to received FINs

Step 4: server, receives ACK. Connection closed.

Note: with small modification, can handle simultaneous FINs.

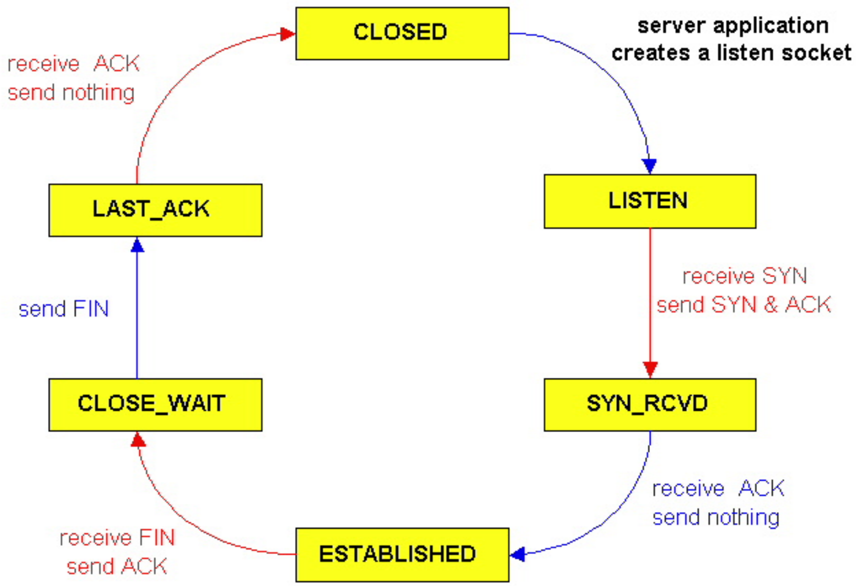


TCP Connection Management (cont)



TCP client lifecycle

TCP server lifecycle



Chapter 3 outline

- 3.1 Transport-layer services
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Principles of reliable data transfer
- 3.5 Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- **3.6 Principles of congestion control**
- 3.7 TCP congestion control

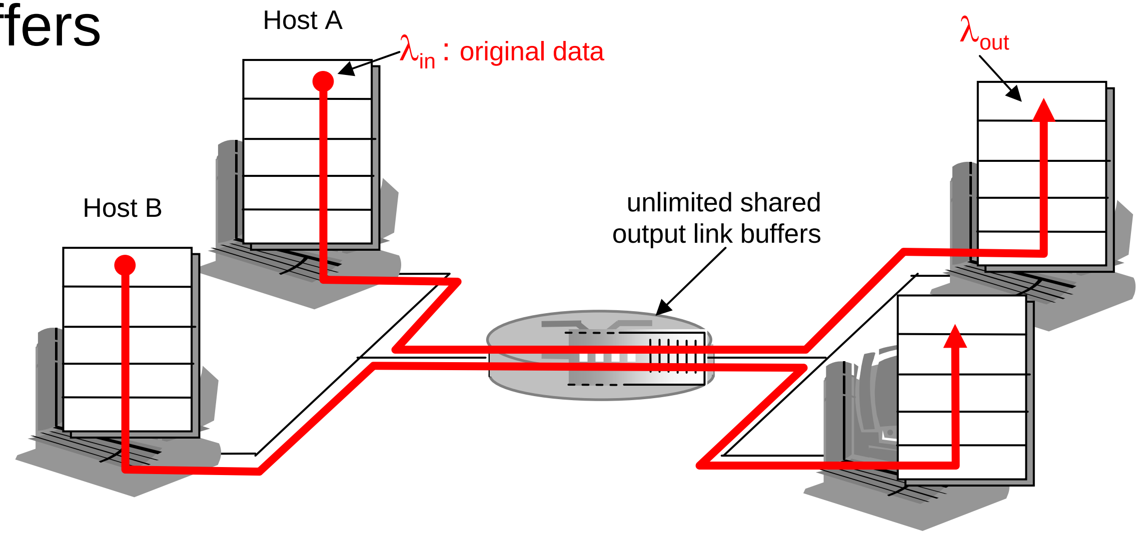
Principles of Congestion Control

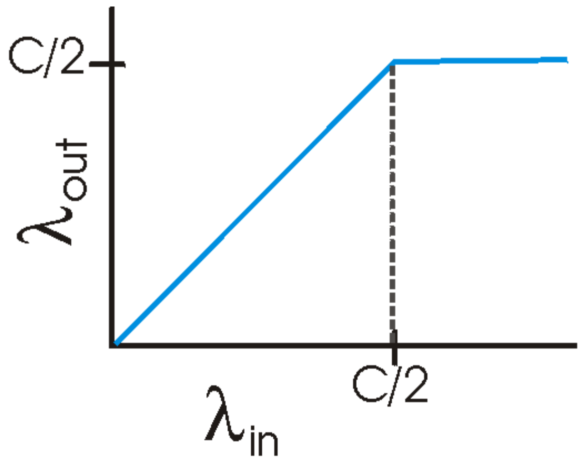
Congestion:

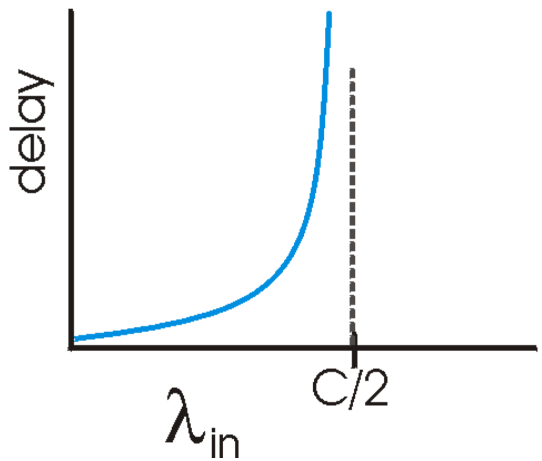
- informally: "too many sources sending too much data too fast for *network* to handle"
- different from flow control!
- manifestations:
  - lost packets (buffer overflow at routers)
  - long delays (queueing in router buffers)
- a top-10 problem!

Causes/costs of congestion: scenario 1

- two senders, two receivers
- one router, infinite buffers
- no retransmission



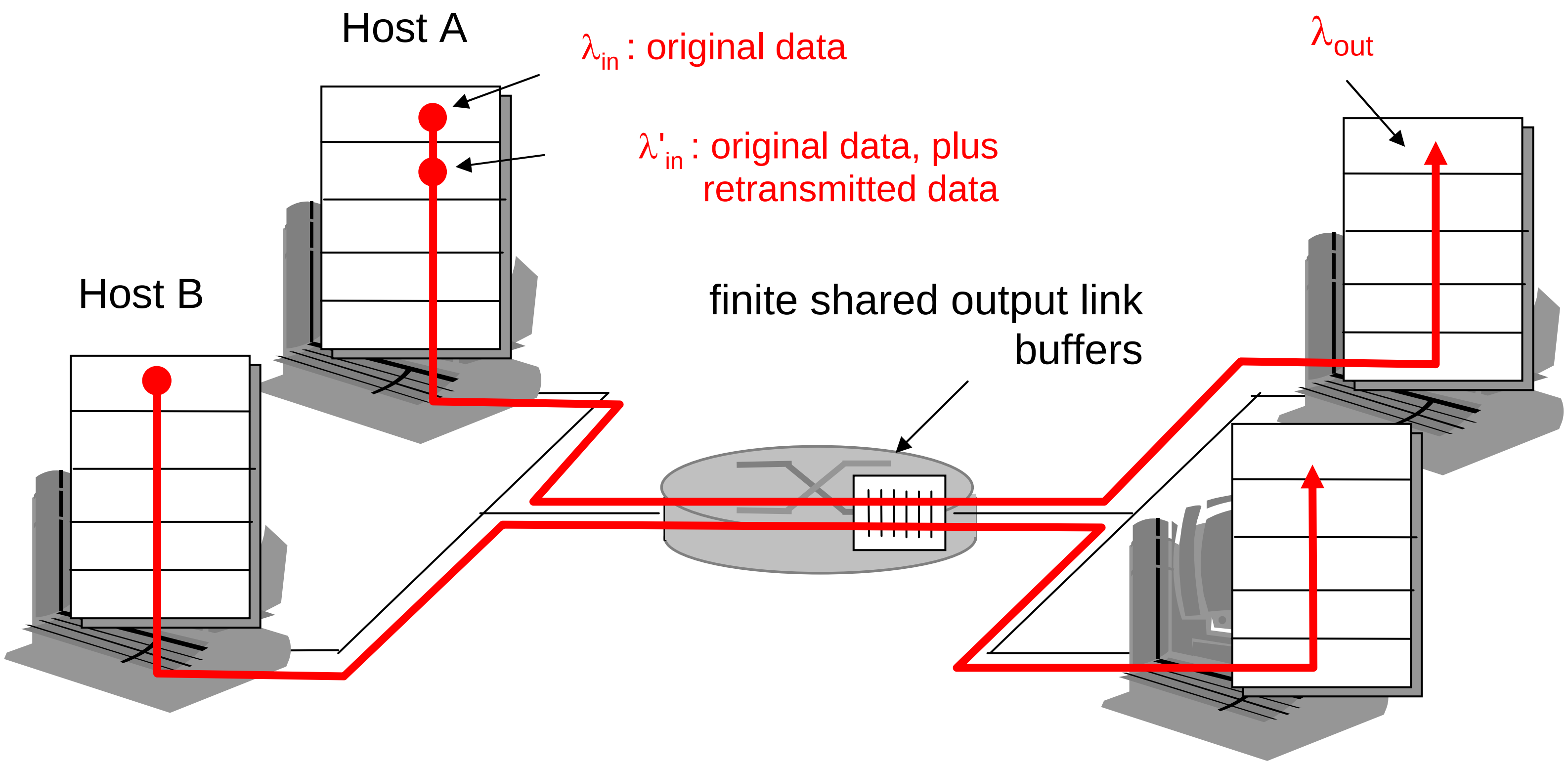




- large delays when congested
- maximum achievable throughput

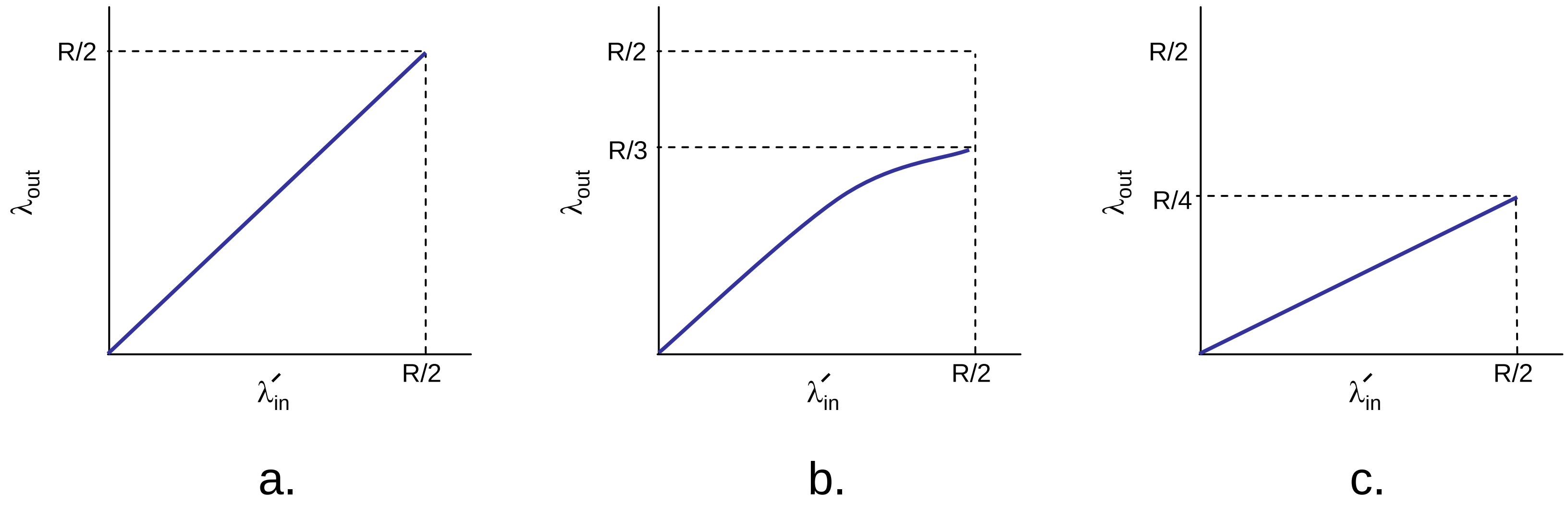
Causes/costs of congestion: scenario 2

- one router, *finite* buffers
- sender retransmission of lost packet



Causes/costs of congestion: scenario 2

- always: $\lambda_{in} = \lambda_{out}$ (goodput)
- "perfect" retransmission only when loss: $\lambda'_{in} > \lambda_{out}$
- retransmission of delayed (not lost) packet makes λ'_{in} larger (than perfect case) for same λ_{out}



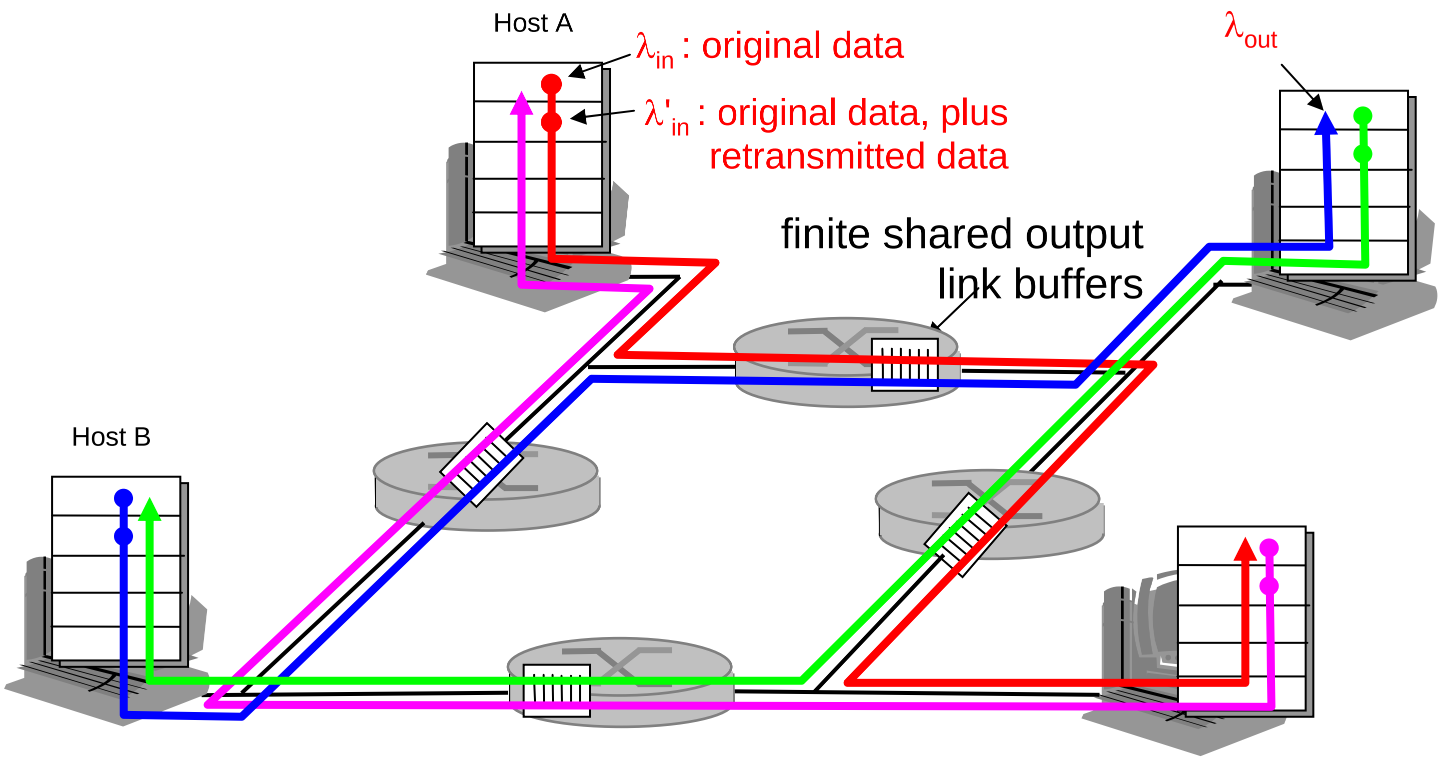
"costs" of congestion:

- more work (retrans) for given "goodput"
- unneeded retransmissions: link carries multiple copies of pkt

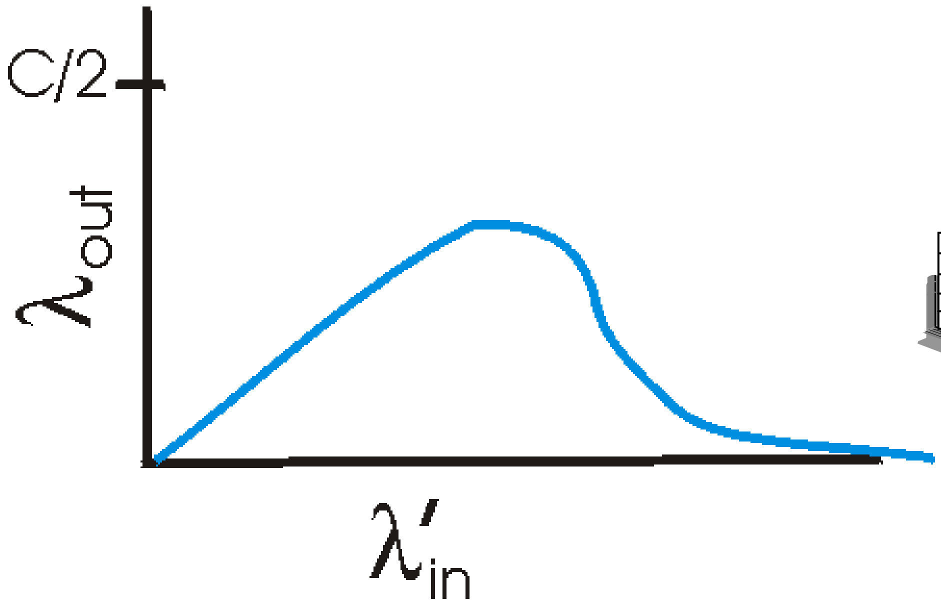
Causes/costs of congestion: scenario 3

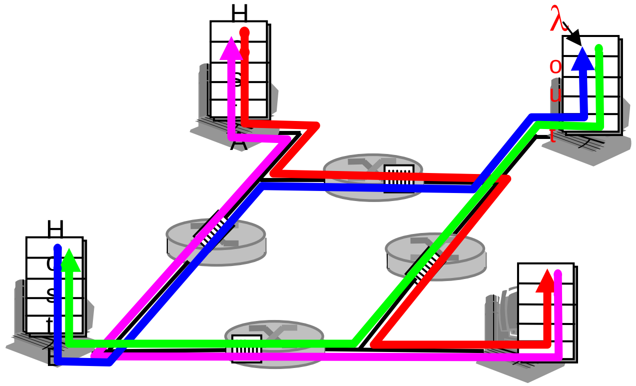
- four senders
- multihop paths
- timeout/retransmit

Q: what happens as λ_{in} and λ'_{in} increase ?



Causes/costs of congestion: scenario 3





Another "cost" of congestion:

- when packet dropped, any "upstream transmission capacity used for that packet was wasted!

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

Case study: ATM ABR congestion control

ABR: available bit rate:

- "elastic service"
- if sender's path "underloaded":
  - sender should use available bandwidth
- if sender's path congested:
  - sender throttled to minimum guaranteed rate

RM (resource management) cells:

- sent by sender, interspersed with data cells
- bits in RM cell set by switches ("*network-assisted*")
  - NI bit: no increase in rate (mild congestion)
  - CI bit: congestion indication
- RM cells returned to sender by receiver, with bits intact

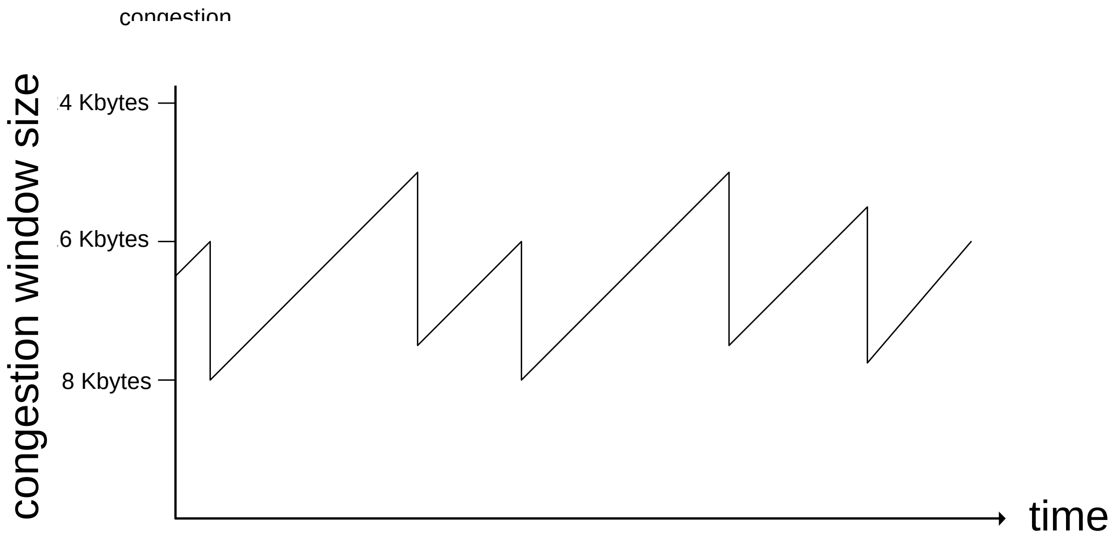
Chapter 3 outline

- 3.1 Transport-layer services
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Principles of reliable data transfer
- 3.5 Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- 3.6 Principles of congestion control
- **3.7 TCP congestion control**

TCP congestion control: 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

Saw tooth behavior: probing for bandwidth



TCP Congestion Control: details

- sender limits transmission:

  `LastByteSent-LastByteAcked` $\leq$ `CongWin`

- Roughly,

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

- **`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:

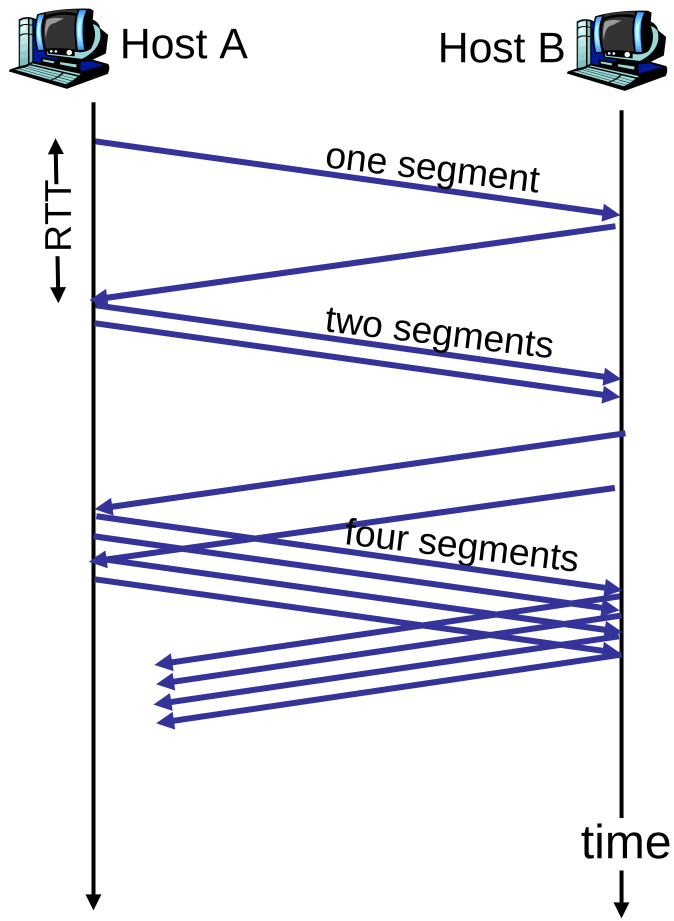
- AIMD
- slow start
- conservative after timeout events

TCP Slow Start

- When connection begins, **`CongWin`** = 1 MSS
  - Example: MSS = 500 bytes & RTT = 200 msec
  - initial rate = 20 kbps
- available bandwidth may be >> 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: 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:

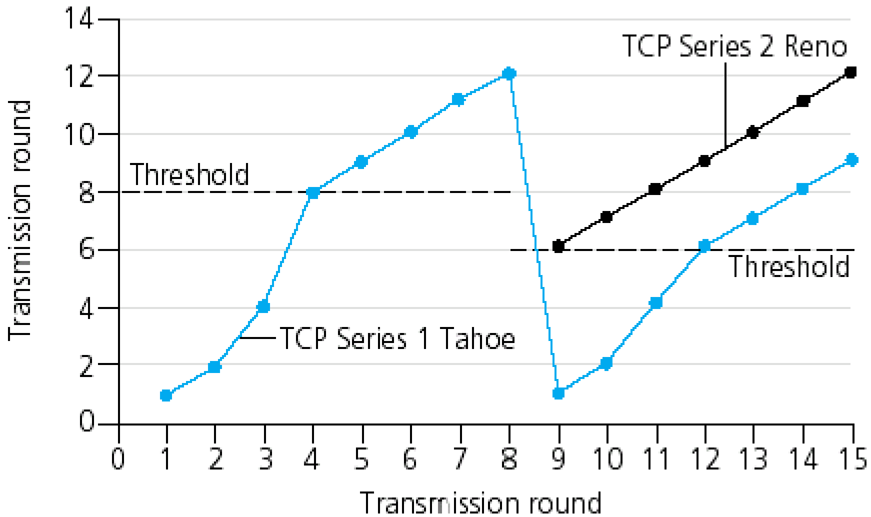
- 3 dup ACKs indicates network capable of delivering some segments
- timeout indicates a "more alarming" congestion scenario

Refinement

- Q: When should the exponential increase switch to linear?
- A: When CongWin gets to 1/2 of its value before timeout.

<u>Implementation:</u>

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



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.

TCP sender congestion control

State	Event	TCP Sender Action	Commentary
Slow Start (SS)	ACK receipt for previously unacked data	CongWin = CongWin + MSS, If (CongWin > Threshold) set state to “Congestion Avoidance”	Resulting in a doubling of CongWin every RTT
Congestion Avoidance (CA)	ACK receipt for previously unacked data	CongWin = CongWin+MSS * (MSS/CongWin)	Additive increase, resulting in increase of CongWin by 1 MSS every RTT
SS or CA	Loss event detected by triple duplicate ACK	Threshold = CongWin/2, CongWin = Threshold, Set state to “Congestion Avoidance”	Fast recovery, implementing multiplicative decrease. CongWin will not drop below 1 MSS.
SS or CA	Timeout	Threshold = CongWin/2, CongWin = 1 MSS, Set state to “Slow Start”	Enter slow start
SS or CA	Duplicate ACK	Increment duplicate ACK count for segment being acked	CongWin and Threshold not changed

TCP summary

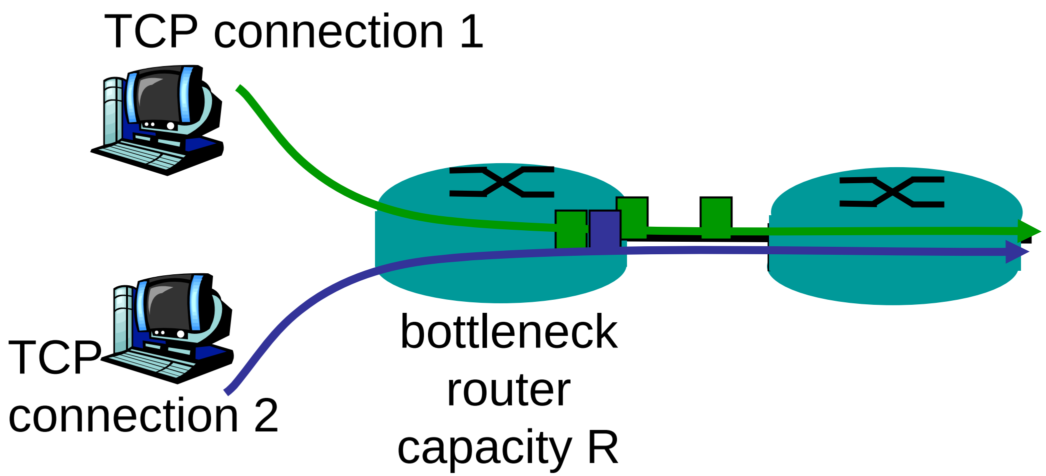
- Connection-oriented: SYN, SYNACK; FIN
- Retransmit lost packets; in-order data: sequence no., ACK no.
- ACKs: either piggybacked, or no-data pure ACK packets if no data travelling in other direction
- Don’t overload receiver: rwin
  - rwin advertised by receiver
- Don’t overload network: cwin
  - cwin affected by receiving ACKs
- Sender buffer = min { rwin, cwin }
- Congestion control:
  - Slow start: exponential growth of cwin
  - Congestion avoidance: linear groth of cwin
  - Timeout; duplicate ACK: shrink cwin
- Continuously adjust RTT estimation

TCP throughput

- What's the average throughout of TCP as a function of window size and RTT?
  - Ignore slow start
- Let W be the window size when loss occurs.
- When window is W, throughput is W/RTT
- Just after loss, window drops to W/2, throughput to W/2RTT.
- Average throughout: 0.75 W/RTT

TCP Fairness

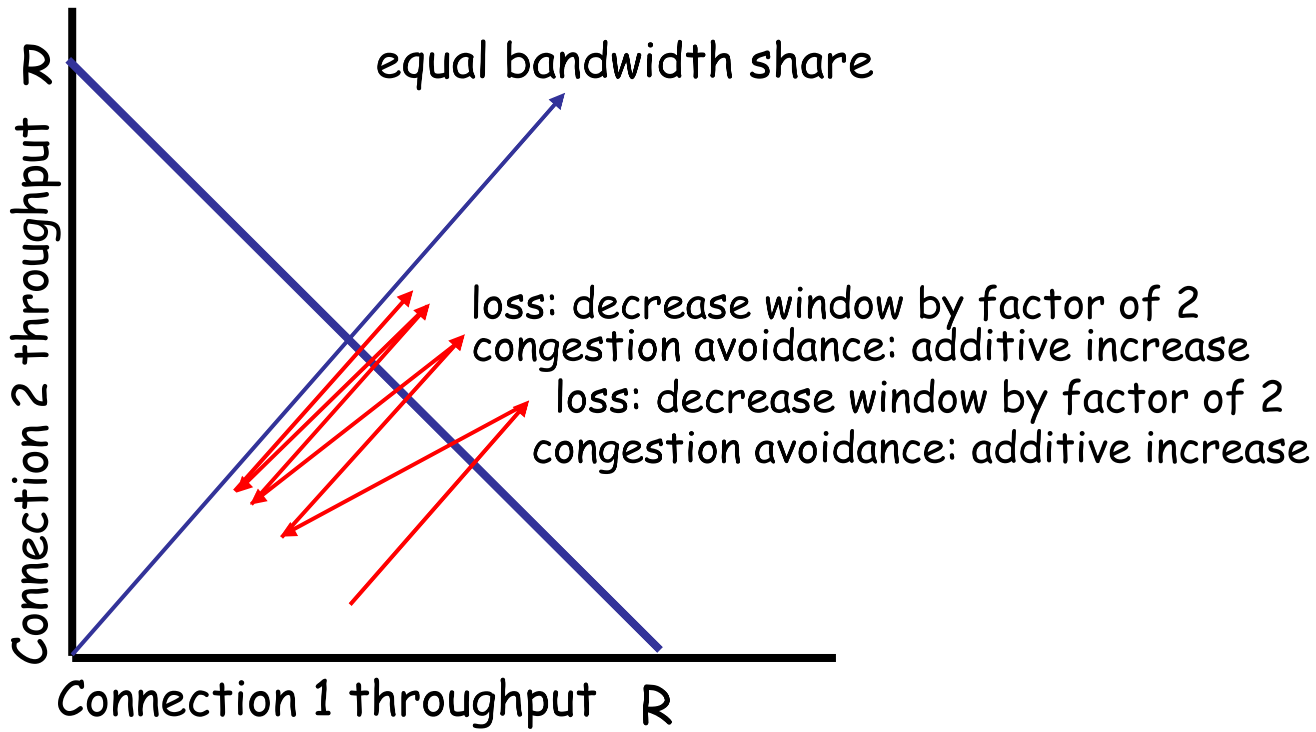
Fairness goal: if K TCP sessions share same bottleneck link of bandwidth R, each should have average rate of R/K



Why is TCP fair?

Two competing sessions:

- Additive increase gives slope of 1, as throughout 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
- Research area: TCP friendly

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: Summary

- principles behind transport layer services:
  - multiplexing, demultiplexing
  - reliable data transfer
  - flow control
  - congestion control
- instantiation and implementation in the Internet
  - UDP
  - TCP

Next:

- leaving the network “edge” (application, transport layers)
- into the network “core”

Master Course
Computer Networks
IN2097

Prof. Dr.-Ing. Georg Carle
Christian Grothoff, Ph.D.

Chair for Network Architectures and Services
Institut für Informatik
Technische Universität München
http://www.net.in.tum.de

Chapter 4: Network Layer

Chapter goals:

- understand principles behind network layer services:
  - network layer service models
  - forwarding versus routing
  - how a router works
  - routing (path selection)
  - dealing with scale
  - advanced topics: IPv6, mobility
- instantiation, implementation in the Internet

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

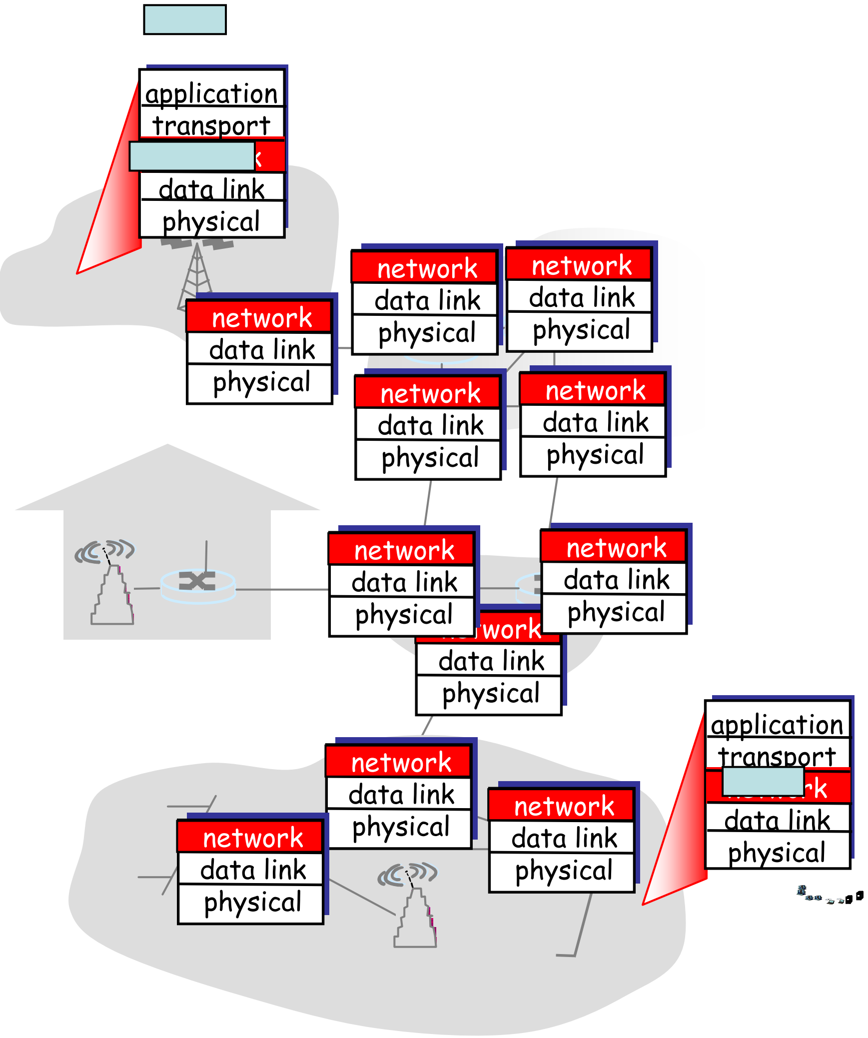
- IPv6
- NAT and NAT Traversal
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
- Broadcast and multicast routing

Network layer

- transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- on rcving side, delivers segments to transport layer
- network layer protocols in *every* host, router
- router examines header fields in all IP datagrams passing through it



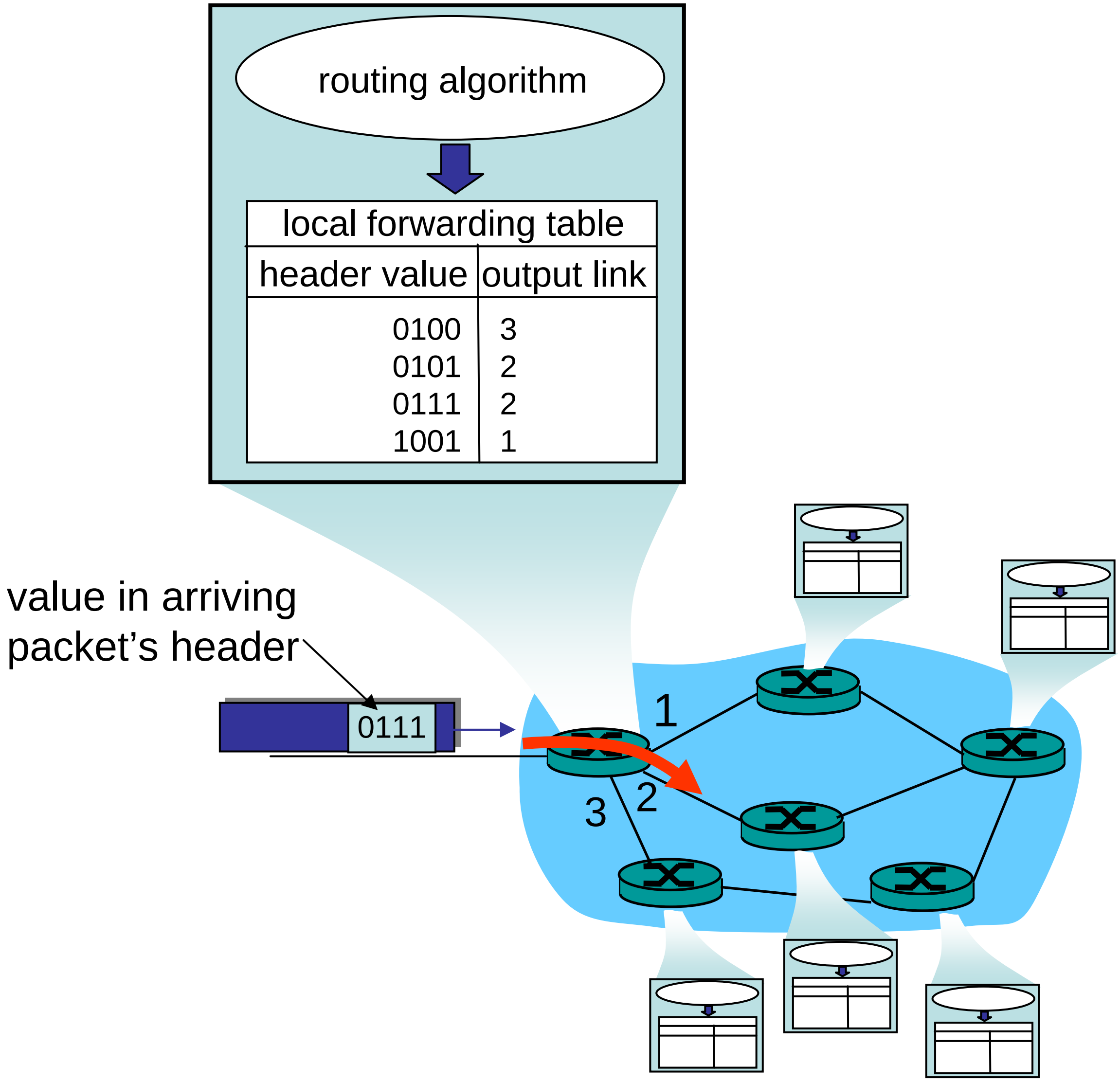
Two Key Network-Layer Functions

- *routing:* determine route taken by packets from source to dest.
  - *routing algorithms*
- *forwarding:* move packets from router's input to appropriate router output

analogy:

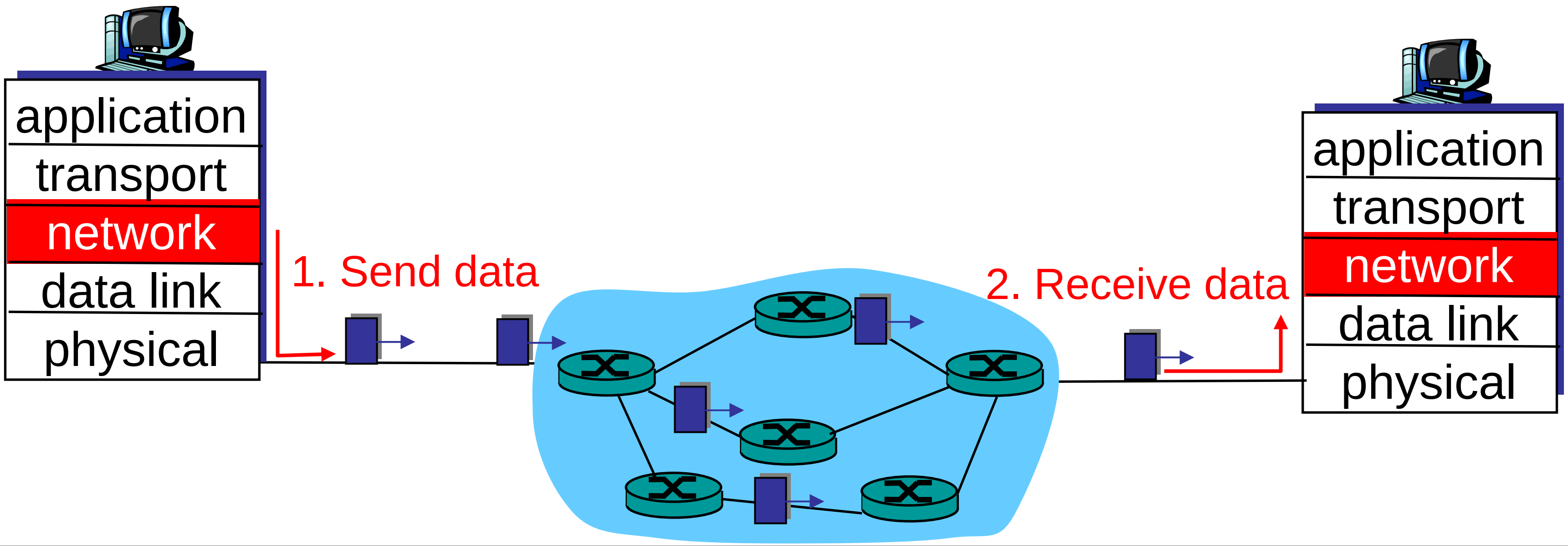
- routing: process of planning trip from source to dest
- forwarding: process of getting through single interchange

Interplay between routing and forwarding



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



Forwarding table

4 billion possible entries

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Longest prefix matching

Prefix Match	Link Interface
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
otherwise	3

Examples

DA: 11001000 00010111 00010110 10100001 Which interface?

DA: 11001000 00010111 00011000 10101010 Which interface?

Part 1

- Introduction
- **IP: Internet Protocol**
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

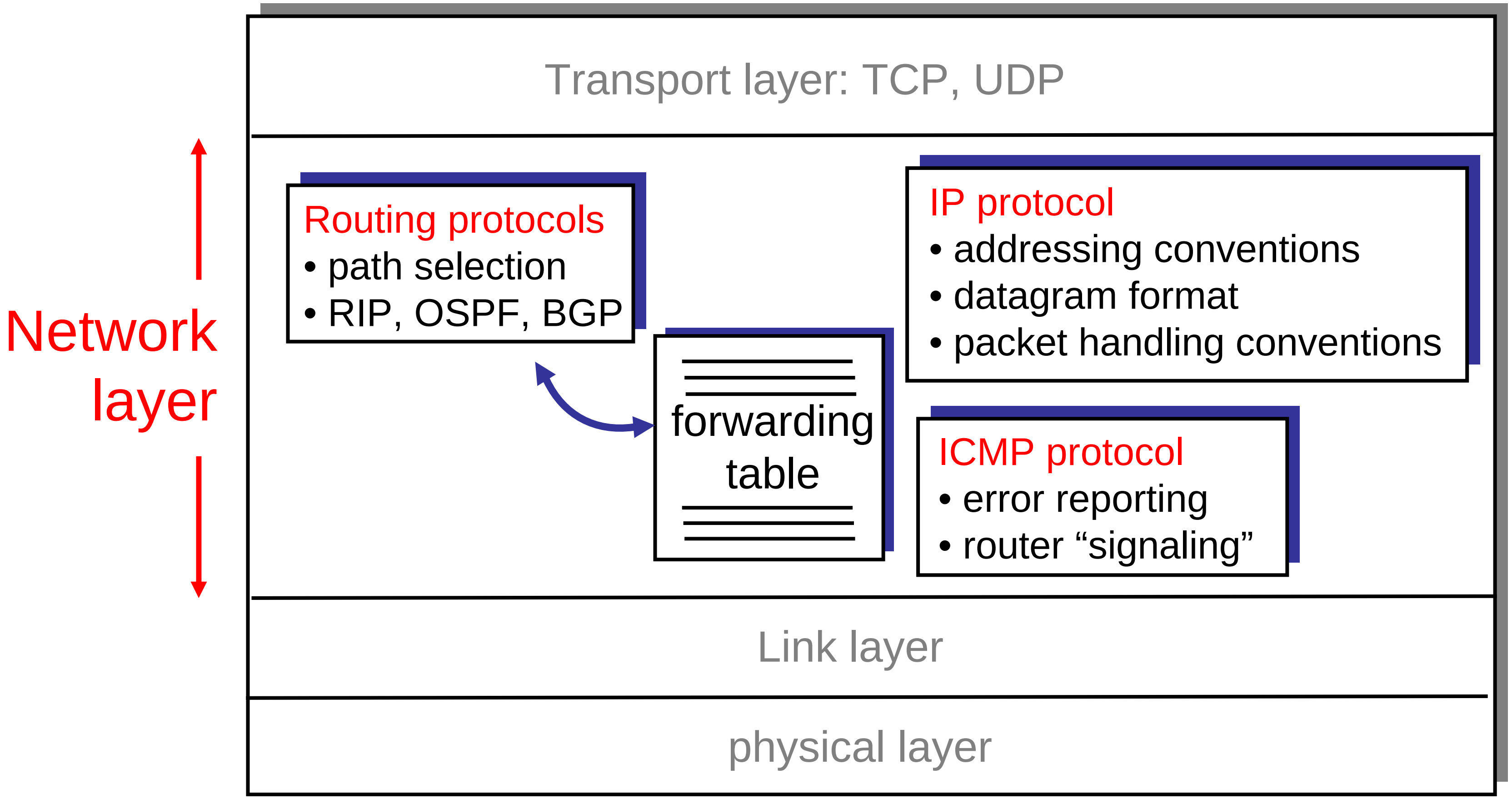
- IPv6
- NAT and NAT Traversal
- Virtual circuit and datagram networks
- What's inside a router

Part 3

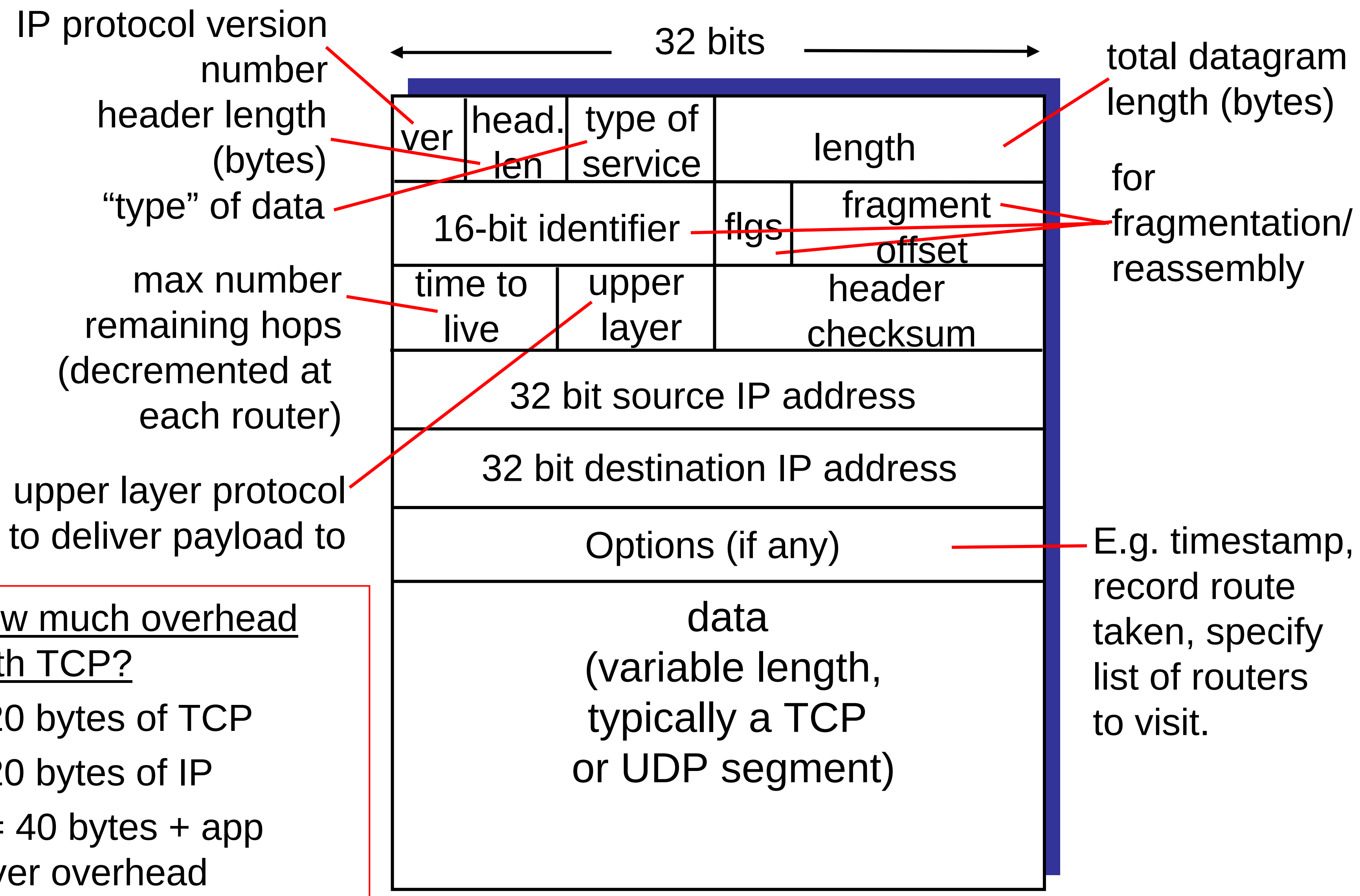
- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
- Broadcast and multicast routing

The Internet Network layer

Host, router network layer functions:



IP datagram format

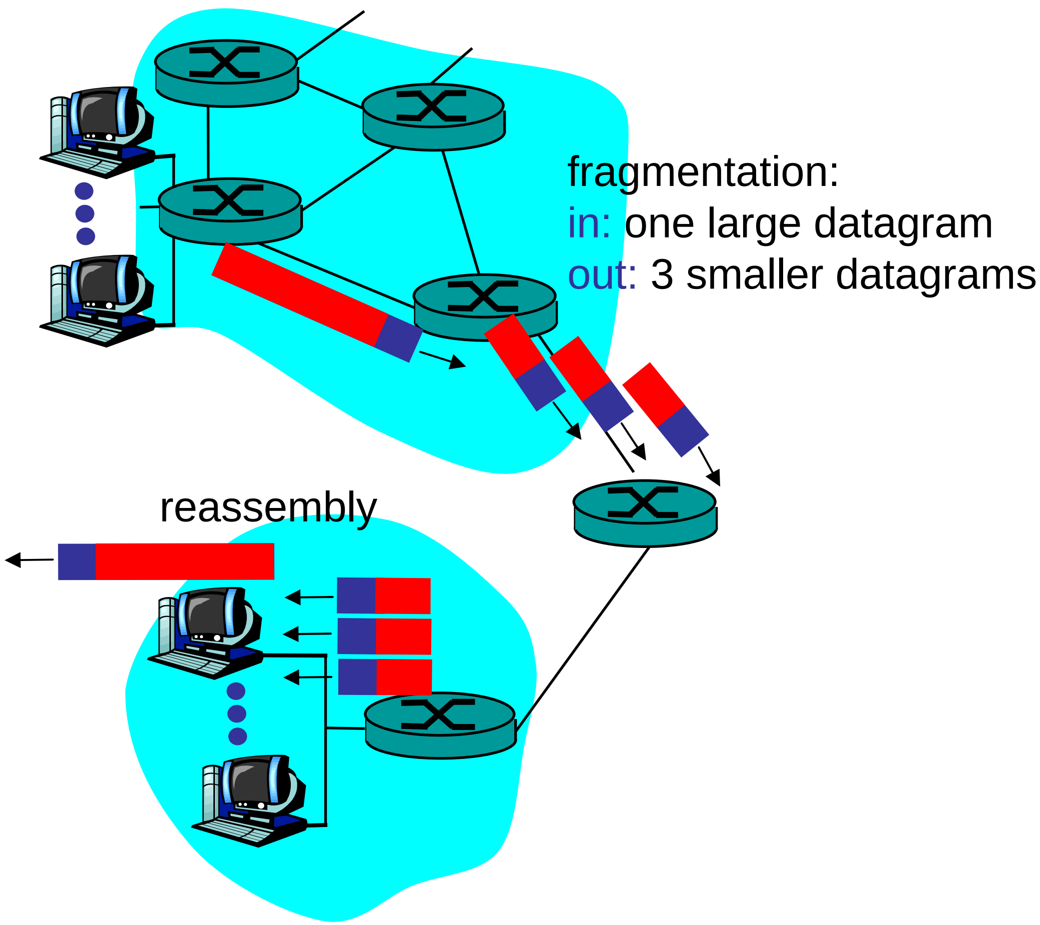


<u>how much overhead with TCP?</u>

- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead

IP Fragmentation & Reassembly

- network links have MTU (max.transfer size) - largest possible link-level frame.
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify, order related fragments

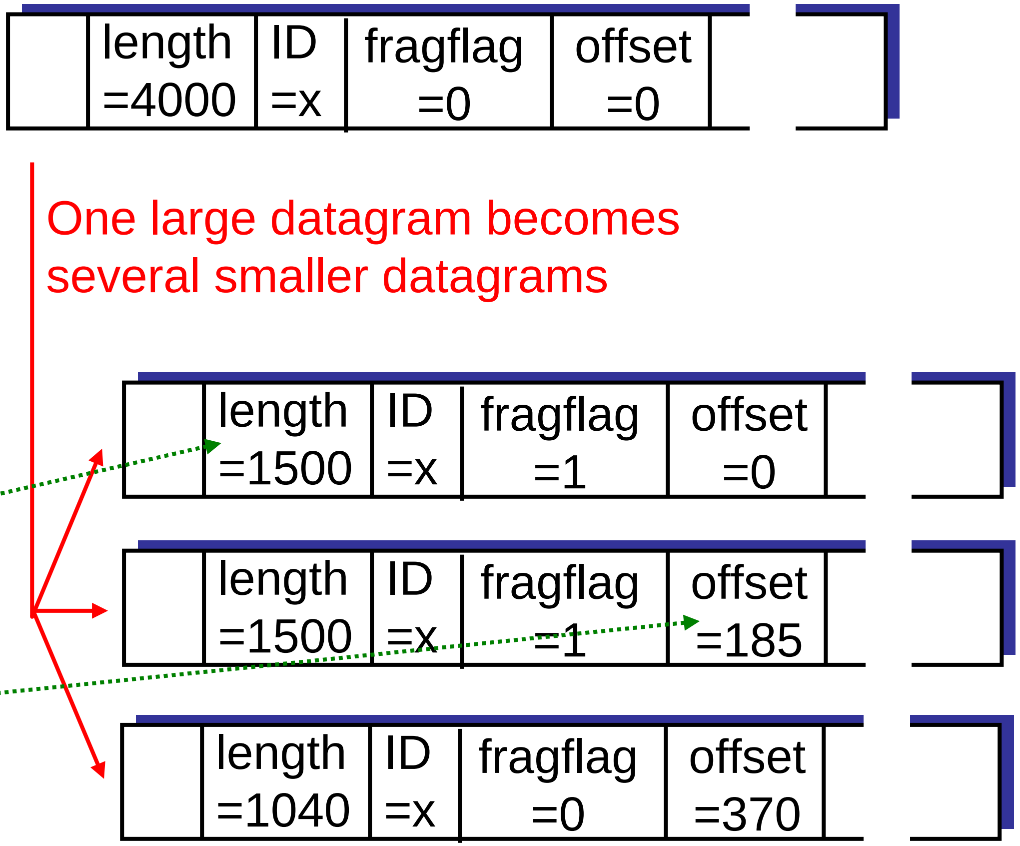


IP Fragmentation and Reassembly

<u>Example</u>

- 4000 byte datagram
- MTU = 1500 bytes

One large datagram becomes several smaller datagrams



1480 bytes in data field

offset = 1480/8

Chapter 4: Network Layer

Part 1

- Introduction
- **IP: Internet Protocol**
  - Datagram format
  - **IPv4 addressing**
  - ICMP

Part 2

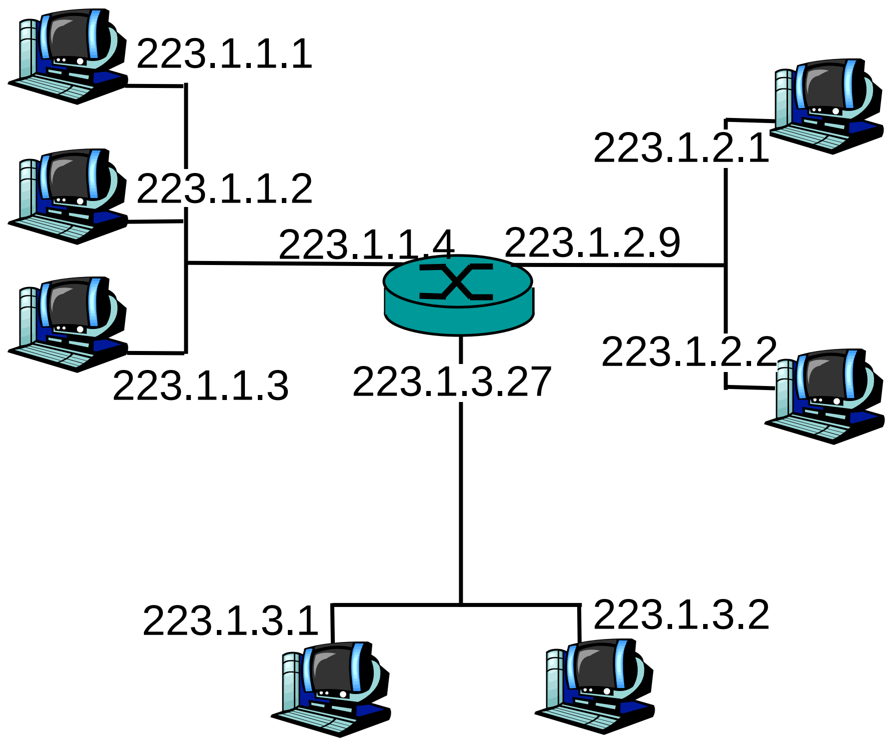
- IPv6
- NAT and NAT Traversal
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
- Broadcast and multicast routing

IP Addressing: introduction

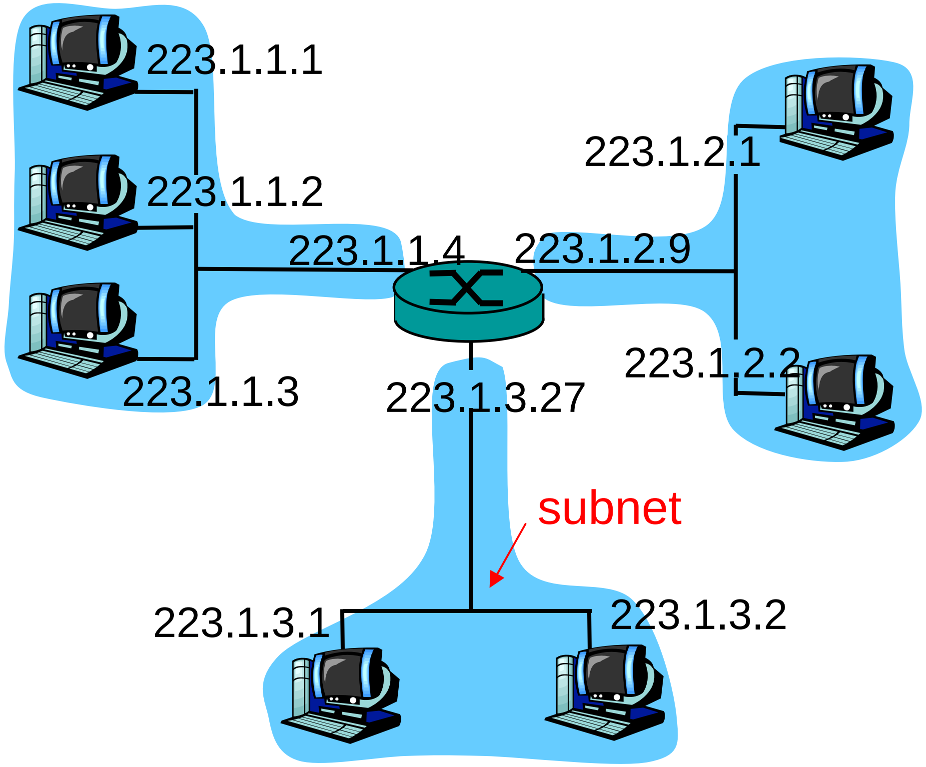
- IP address: 32-bit identifier for host, router *interface*
- *interface:* connection between host/router and physical link
  - router's typically have multiple interfaces
  - host typically has one interface
  - IP addresses associated with each interface



$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$

Subnets

- IP address:
  - subnet part (high order bits)
  - host part (low order bits)
- *What's a subnet ?*
  - device interfaces with same subnet part of IP address
  - can physically reach each other without intervening router

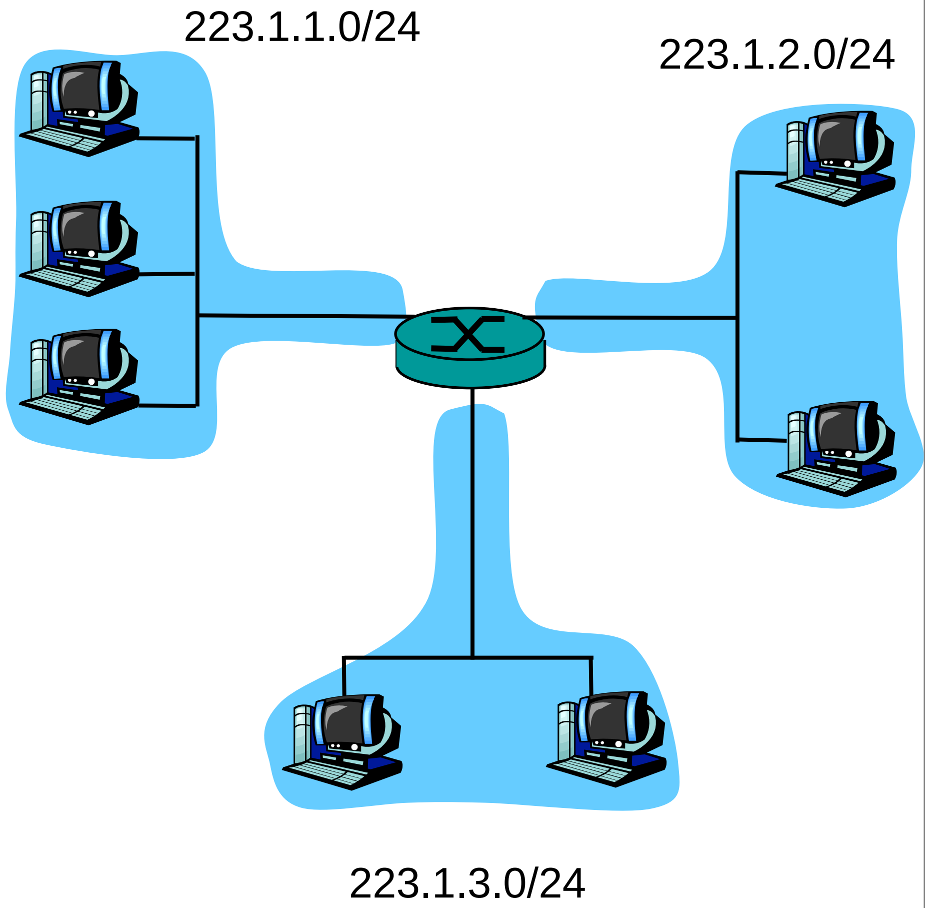


network consisting of 3 subnets

Subnets

Recipe

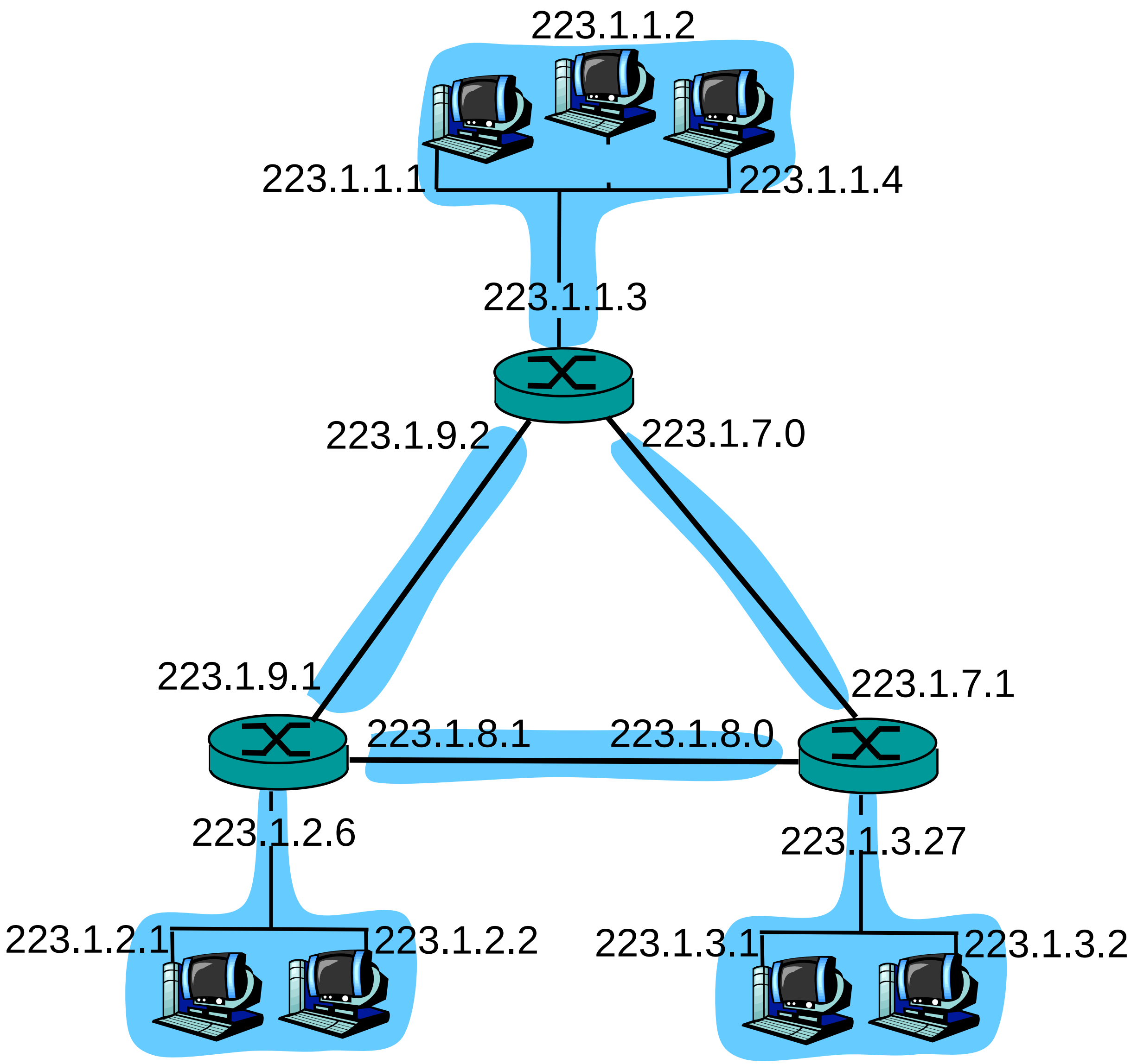
- To determine the subnets, detach each interface from its host or router, creating islands of isolated networks. Each isolated network is called a subnet.



Subnet mask: /24

Subnets

How many?



IP addressing: CIDR

CIDR: Classless InterDomain Routing

- subnet portion of address of arbitrary length
- address format: a.b.c.d/x, where x is # bits in subnet portion of address

subnet part

host part

11001000 00010111 00010000 00000000

200.23.16.0/23

IP addresses: how to get one?

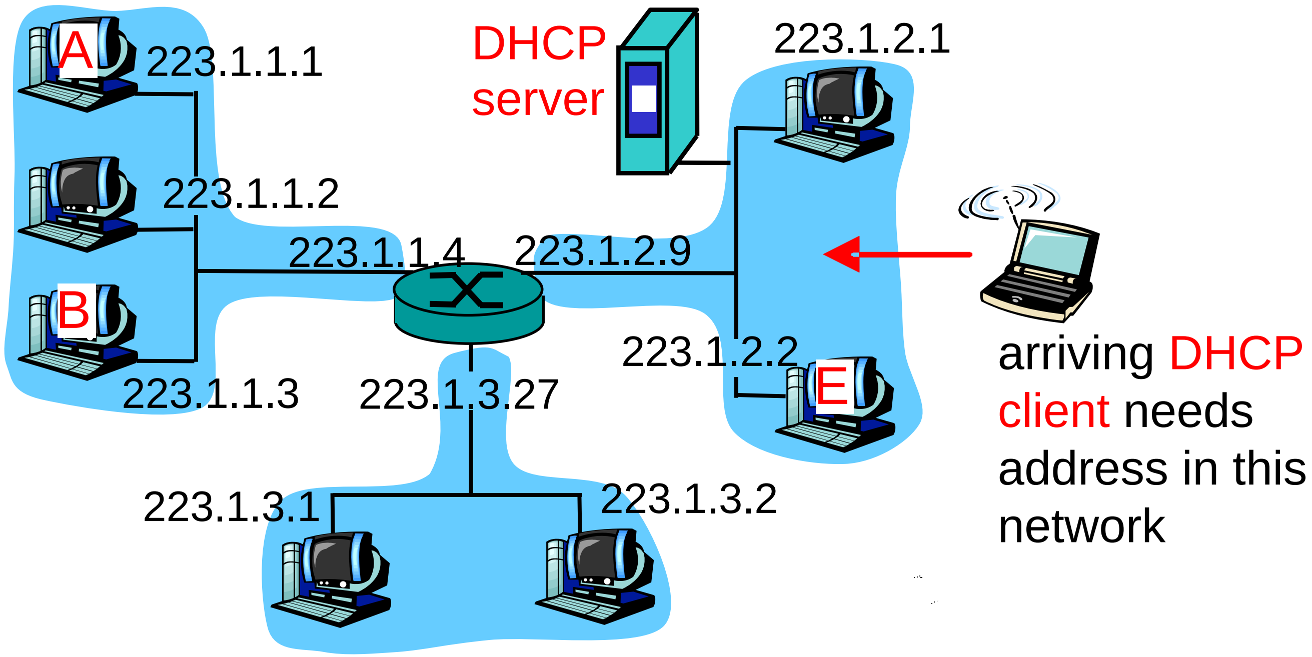
Q: How does a *host* get IP address?

- 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"

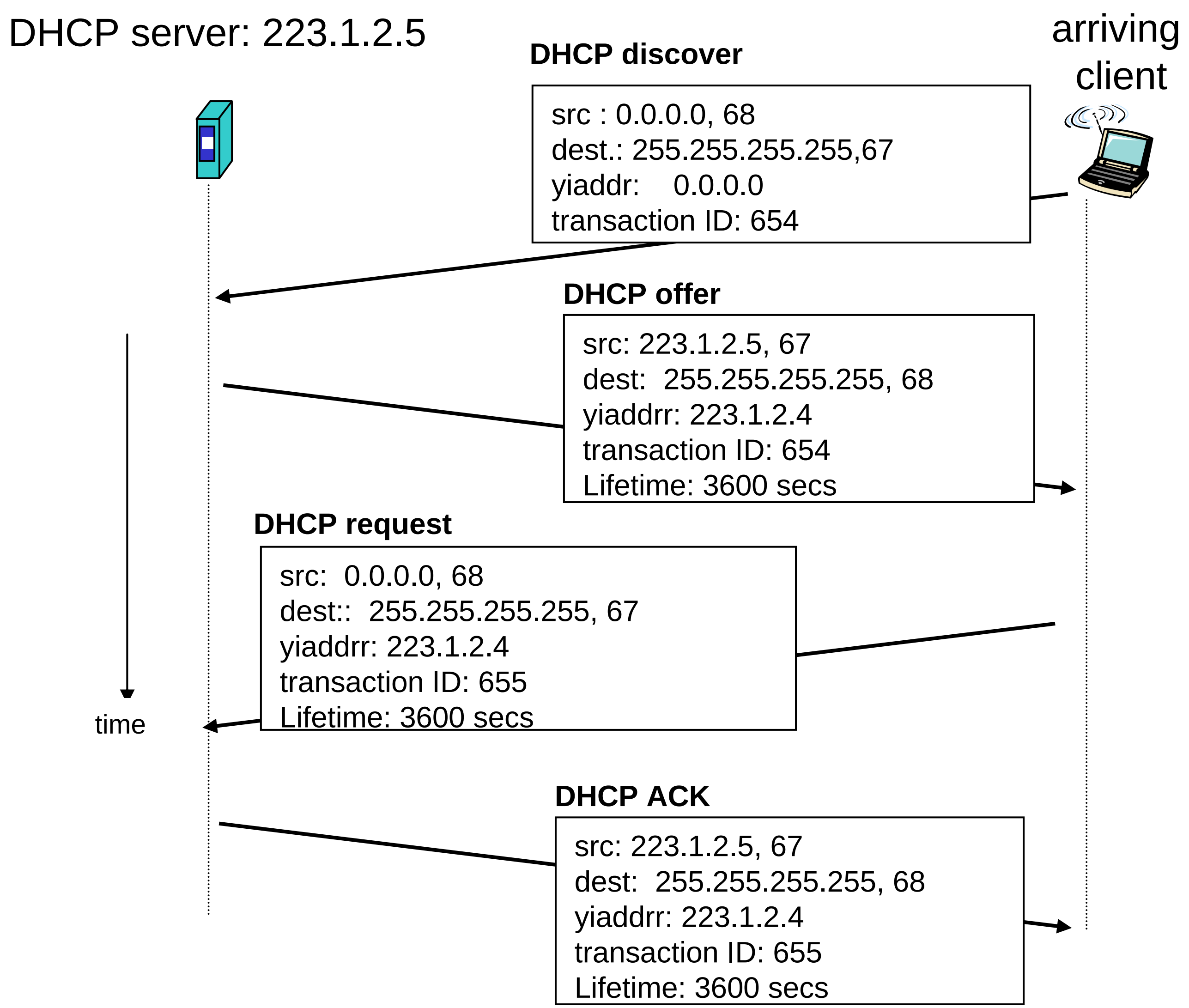
DHCP: Dynamic Host Configuration Protocol

- 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 an "on")
  - Support for mobile users who want to join network (more shortly)
- DHCP overview:
  - host broadcasts "DHCP discover" msg
  - DHCP server responds with "DHCP offer" msg
  - host requests IP address: "DHCP request" msg
  - DHCP server sends address: "DHCP ack" msg

DHCP client-server scenario



DHCP client-server scenario



IP addresses: how to get one?

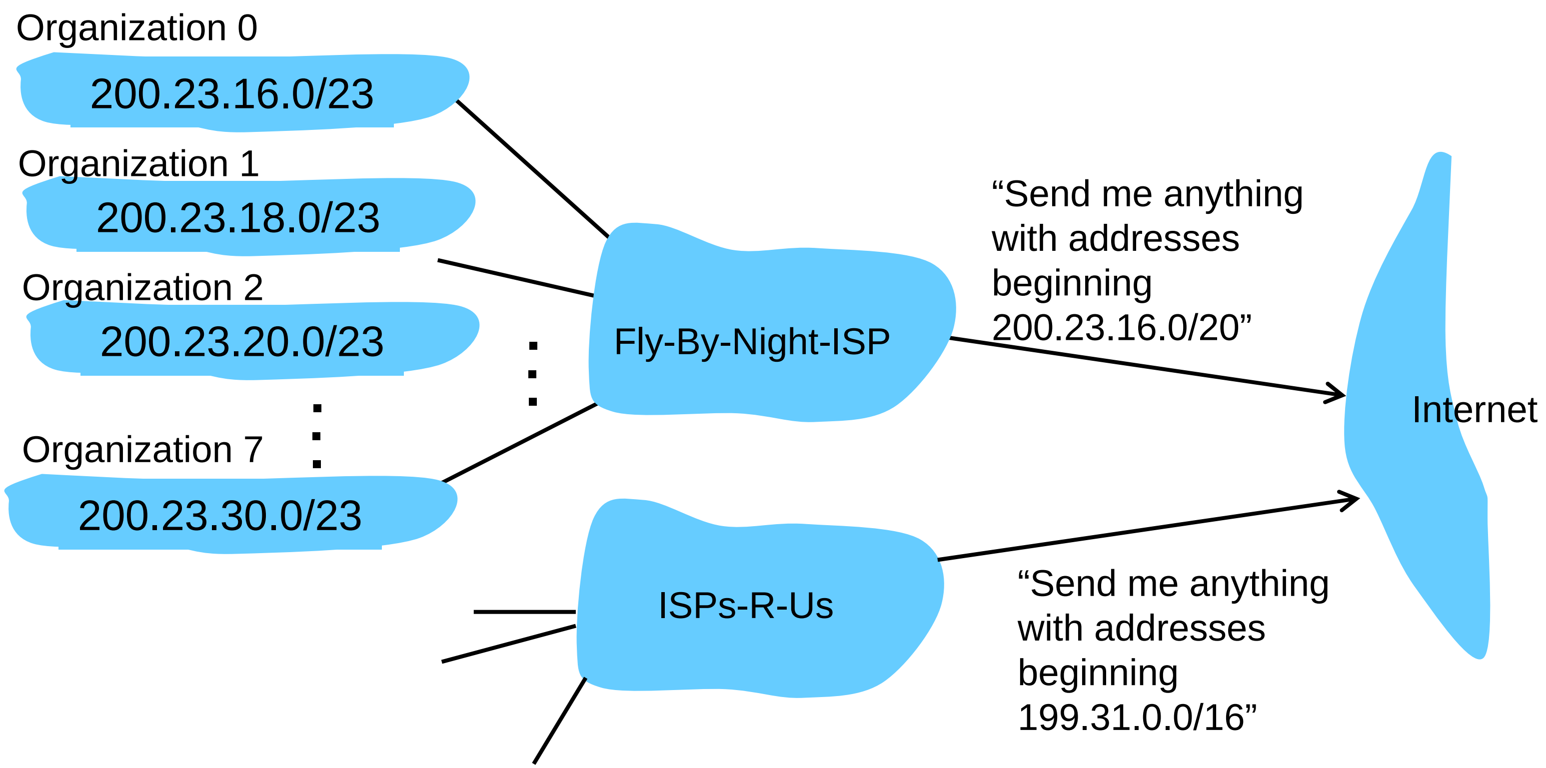
Q: How does *network* get subnet part of IP addr?

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

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

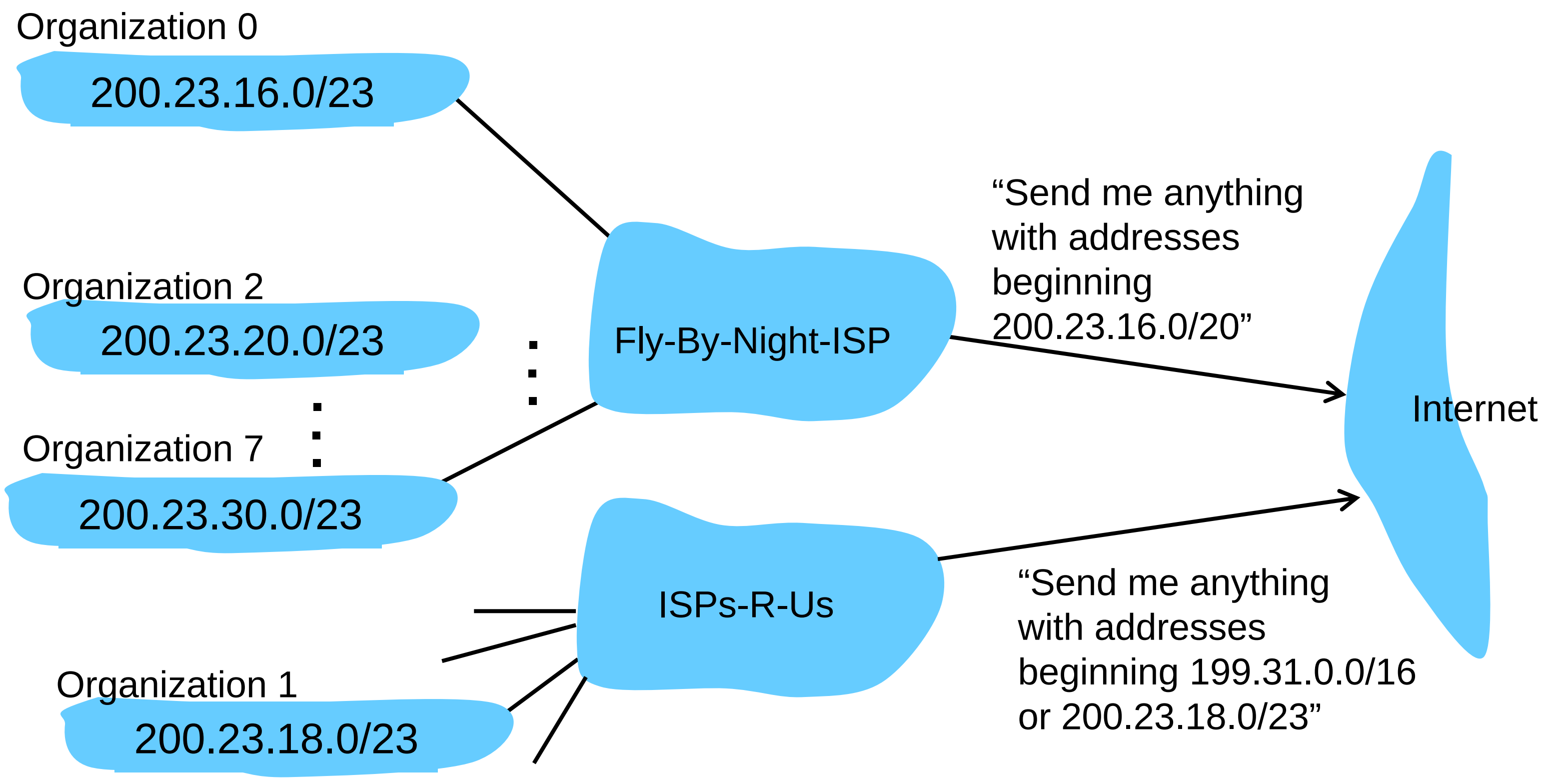
Hierarchical addressing: route aggregation

Hierarchical addressing allows efficient advertisement of routing information:



Hierarchical addressing: more specific routes

ISPs-R-Us has a more specific route to Organization 1



IP addressing: the last word...

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

Part 1

- Introduction
- **IP: Internet Protocol**
  - Datagram format
  - IPv4 addressing
  - **ICMP**

Part 2

- IPv6
- NAT and NAT Traversal
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
- Broadcast and multicast routing

ICMP: Internet Control Message Protocol

- used by hosts & routers to communicate network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer "above" IP:
  - ICMP msgs carried in IP datagrams
- ICMP message: type, code plus first 8 bytes of IP datagram causing error

Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

Traceroute and ICMP

- Source sends series of UDP segments to dest
  - First has TTL =1
  - Second has TTL=2, etc.
  - Unlikely port number
- When nth datagram arrives to nth router:
  - Router discards datagram
  - And sends to source an ICMP message (type 11, code 0)
  - Message includes name of router& IP address

- When ICMP message arrives, source calculates RTT
- Traceroute does this 3 times

Stopping criterion

- UDP segment eventually arrives at destination host
- Destination returns ICMP "host unreachable" packet (type 3, code 3)
- When source gets this ICMP, stops.

IPv6

Christian Grothoff
christian@grothoff.org
http://grothoff.org/christian/

"One of the chief factors that has prevented this transformation, though objectively it has been on the agenda for years, is the absence or the repression of the need for transformation, which has to be present as the qualitatively differentiating factor among the social groups that are to make the transformation." – Herbert Marcuse

Overview

- Motivation for IPv6
- Key Differences between IPv4 and IPv6
- Security Considerations
- Infrastructure Migration
- Migrating Code to IPv6

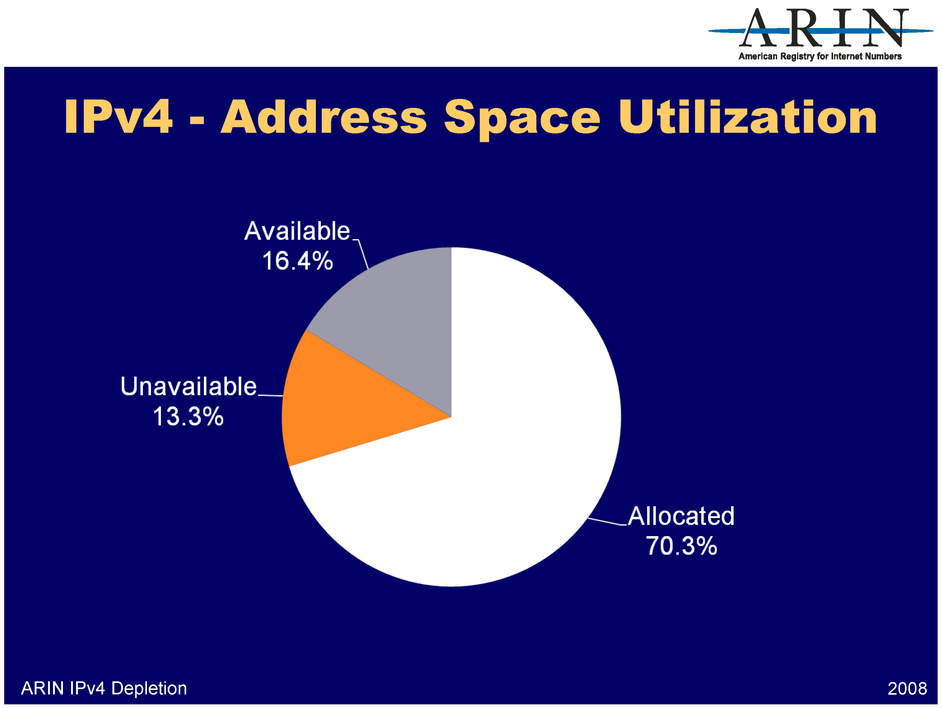
Motivation

We're running out of IPv4 addresses:

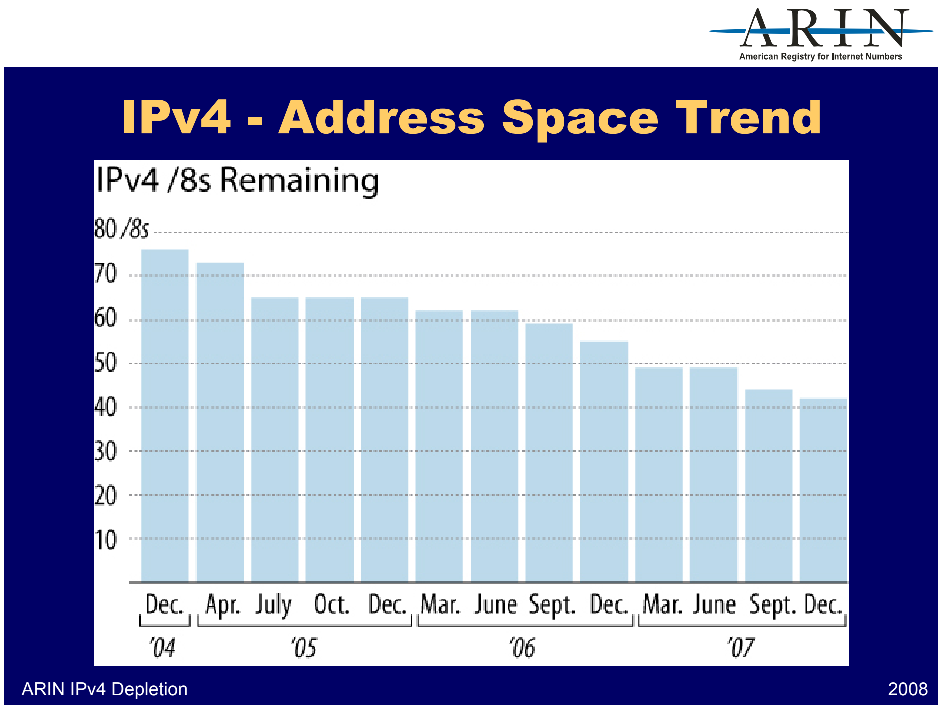
- 32-bit
- Routing considerations limit use (CIDR, renumbering costs)
- Impact differs by geography (see RIR assignments)
- New services accelerate pace of address consumption (mobiles!)

US Federal Networks must be IPv6-capable since June 2008.

IPv4 Address Space Utilization



IPv4/8s Remaining



IPv4 Address Fractal Map Jan-2000

000 Reserved	001 Reserved	014 PDN	015 HP	016 DEC	019 Ford	020 CsC	021 US DoD	234 Multicast	235 Multicast	236 Multicast	239 Multicast	240 Class E	241 Class E	254 Class E	255 Class E
003 GE	002 Reserved	013 Xerox	012 AT&T	017 Apple	018 MIT	023 Reserved	022 US DoD	233 Multicast	232 Multicast	237 Multicast	238 Multicast	243 Class E	242 Class E	253 Class E	252 Class E
004 L3	007 Reserved	008 L3	011 US DoD	030 US DoD	029 US DoD	024 Reserved	025 UK Defense	230 Multicast	231 Multicast	226 Multicast	225 Multicast	244 Class E	247 Class E	248 Class E	251 Class E
005 Reserved	006 US DoD	009 IBM	010 Private	031 Reserved	028 US DoD	027 Reserved	026 US DoD	229 Multicast	228 Multicast	227 Multicast	224 Multicast	245 Class E	246 Class E	249 Class E	250 Class E
058 Reserved	057 SITA	054 Merck	053 Cap Debis	032 AT&T	035 MERIT	036 Reserved	037 Reserved	218 Reserved	219 Reserved	220 Reserved	223 Reserved	202 APnic	201 Reserved	198 Various	197 Reserved
059 Reserved	056 US Postal	055 US DoD	052 El duPONT	033 US DoD	034 Haliburton	039 Reserved	038 PSI	217 Reserved	216 ARIN	221 Reserved	222 Reserved	203 APnic	200 Reserved	199 ARIN	196 AfrNIC
060 Reserved	061 APnic	050 Reserved	051 UK DSS	046 Reserved	045 Interop	040 Eli Lily	041 Reserved	214 US DoD	215 US DoD	210 APnic	209 ARIN	204 ARIN	205 ARIN	194 RIPE	195 RIPE
063 ARIN	062 RIPE	049 Reserved	048 Prudential	047 Bell North	044 Radio	043 Inet	042 Reserved	213 RIPE	212 RIPE	211 APnic	208 ARIN	207 ARIN	206 ARIN	192 RIPE	192 Various
064 ARIN	067 Reserved	068 Reserved	069 Reserved	122 Reserved	123 Reserved	124 Reserved	127 Loopback	128 Various	131 Various	132 Various	133 Various	186 Reserved	187 Reserved	188 Reserved	191 Various
065 Reserved	066 Reserved	071 Reserved	070 Reserved	121 Reserved	120 Reserved	125 Reserved	126 Reserved	129 Various	130 Various	135 Various	134 Various	185 Reserved	184 Reserved	189 Reserved	190 Reserved
078 Reserved	077 Reserved	072 Reserved	073 Reserved	118 Reserved	119 Reserved	114 Reserved	113 Reserved	142 Various	141 Various	136 Various	137 Various	182 Reserved	183 Reserved	178 Reserved	177 Reserved
079 Reserved	076 Reserved	075 Reserved	074 Reserved	117 Reserved	116 Reserved	115 Reserved	112 Reserved	143 Various	140 Various	139 Various	138 Various	181 Reserved	180 Reserved	179 Reserved	176 Reserved
080 Reserved	081 Reserved	094 Reserved	095 Reserved	096 Reserved	097 Reserved	110 Reserved	111 Reserved	144 Various	145 Various	158 Various	159 Various	160 Various	161 Various	174 Reserved	175 Reserved
083 Reserved	082 Reserved	093 Reserved	092 Reserved	099 Reserved	098 Reserved	109 Reserved	108 Reserved	147 Various	146 Various	157 Various	156 Various	163 Various	162 Various	173 Reserved	172 Various
084 Reserved	087 Reserved	088 Reserved	091 Reserved	100 Reserved	103 Reserved	104 Reserved	107 Reserved	148 Various	151 Various	152 Various	155 Various	164 Various	167 Various	168 Various	171 Various
085 Reserved	086 Reserved	089 Reserved	090 Reserved	101 Reserved	102 Reserved	105 Reserved	106 Reserved	149 Various	150 Various	153 Various	154 Various	165 Various	166 Various	169 Various	170 Various

Fractal map: Layout by Randall Munroe, Time Sequence by Tony Hain, Highlighted by Jeff Apcar

IPv4 Address Fractal Map Jan-2001

000 Reserved	001 Reserved	014 PDN	015 HP	016 DEC	019 Ford	020 CsC	021 US DoD	234 Multicast	235 Multicast	236 Multicast	239 Multicast	240 Class E	241 Class E	254 Class E	255 Class E
003 GE	002 Reserved	013 Xerox	012 AT&T	017 Apple	018 MIT	023 Reserved	022 US DoD	233 Multicast	232 Multicast	236 Multicast	238 Multicast	243 Class E	242 Class E	253 Class E	252 Class E
004 L3	007 Reserved	008 L3	011 US DoD	030 US DoD	029 US DoD	024 Reserved	025 UK Defense	230 Multicast	231 Multicast	226 Multicast	225 Multicast	244 Class E	247 Class E	248 Class E	251 Class E
005 Reserved	006 US DoD	009 IBM	010 Private	031 Reserved	028 US DoD	027 Reserved	026 US DoD	229 Multicast	228 Multicast	227 Multicast	224 Multicast	245 Class E	246 Class E	249 Class E	250 Class E
058 Reserved	057 SITA	054 Merck	053 Cap Debis	032 AT&T	035 MERIT	036 Reserved	037 Reserved	218 APnic	219 Reserved	220 Reserved	223 Reserved	202 APnic	201 Reserved	198 Various	197 Reserved
059 Reserved	056 US Postal	055 US DoD	052 El duPONT	033 US DoD	034 Haliburton	039 Reserved	038 PSI	217 RIPE	216 ARIN	221 Reserved	222 Reserved	203 APnic	200 Reserved	199 ARIN	196 AfrNIC
060 Reserved	061 APnic	050 Reserved	051 UK DSS	046 Reserved	045 Interop	040 Eli Lily	041 Reserved	214 US DoD	215 US DoD	210 APnic	209 ARIN	204 ARIN	205 ARIN	194 RIPE	195 RIPE
063 ARIN	062 RIPE	049 Reserved	048 Prudential	047 Bell North	044 Radio	043 Inet	042 Reserved	213 RIPE	212 RIPE	211 APnic	208 ARIN	207 ARIN	206 ARIN	192 RIPE	192 Various
064 ARIN	067 Reserved	068 Reserved	069 Reserved	122 Reserved	123 Reserved	124 Reserved	127 Loopback	128 Various	131 Various	132 Various	133 Various	186 Reserved	187 Reserved	188 Reserved	191 Various
065 ARIN	066 ARIN	071 Reserved	070 Reserved	121 Reserved	120 Reserved	125 Reserved	126 Reserved	129 Various	130 Various	135 Various	134 Various	185 Reserved	184 Reserved	189 Reserved	190 Reserved
078 Reserved	077 Reserved	072 Reserved	073 Reserved	118 Reserved	119 Reserved	114 Reserved	113 Reserved	142 Various	141 Various	136 Various	137 Various	182 Reserved	183 Reserved	178 Reserved	177 Reserved
079 Reserved	076 Reserved	075 Reserved	074 Reserved	117 Reserved	116 Reserved	115 Reserved	112 Reserved	143 Various	140 Various	139 Various	138 Various	181 Reserved	180 Reserved	179 Reserved	176 Reserved
080 Reserved	081 Reserved	094 Reserved	095 Reserved	096 Reserved	097 Reserved	110 Reserved	111 Reserved	144 Various	145 Various	158 Various	159 Various	160 Various	161 Various	174 Reserved	175 Reserved
083 Reserved	082 Reserved	093 Reserved	092 Reserved	099 Reserved	098 Reserved	109 Reserved	108 Reserved	147 Various	146 Various	157 Various	156 Various	163 Various	162 Various	173 Reserved	172 Various
084 Reserved	087 Reserved	088 Reserved	091 Reserved	100 Reserved	103 Reserved	104 Reserved	107 Reserved	148 Various	151 Various	152 Various	155 Various	164 Various	167 Various	168 Various	171 Various
085 Reserved	086 Reserved	089 Reserved	090 Reserved	101 Reserved	102 Reserved	105 Reserved	106 Reserved	149 Various	150 Various	153 Various	154 Various	165 Various	166 Various	169 Various	170 Various

Fractal map: Layout by Randall Munroe, Time Sequence by Tony Hain, Highlighted by Jeff Apcar

IPv4 Address Fractal Map Jan-2002

Fractal map: Layout by Randall Munroe, Time Sequence by Tony Hain, Highlighted by Jeff Apcar

000 Reserved	001 Reserved	014 PDN	015 HP	016 DEC	019 Ford	020 CsC	021 US DoD	234 Multicast	235 Multicast	236 Multicast	239 Multicast	240 Class E	241 Class E	254 Class E	255 Class E
003 GE	002 Reserved	013 Xerox	012 AT&T	017 Apple	018 MIT	023 Reserved	022 US DoD	233 Multicast	232 Multicast	237 Multicast	238 Multicast	243 Class E	242 Class E	253 Class E	252 Class E
004 L3	007 Reserved	008 L3	011 US DoD	030 US DoD	029 US DoD	024 Cable	025 UK Defense	230 Multicast	231 Multicast	226 Multicast	225 Multicast	244 Class E	247 Class E	248 Class E	251 Class E
005 Reserved	006 US DoD	009 IBM	010 Private	031 Reserved	028 US DoD	027 Reserved	026 US DoD	229 Multicast	228 Multicast	227 Multicast	224 Multicast	245 Class E	246 Class E	249 Class E	250 Class E
058 Reserved	057 SITA	054 Merck	053 Cap Debis	032 AT&T	035 MERIT	036 Reserved	037 Reserved	218 APnic	219 APnic	220 APnic	223 Reserved	202 APnic	201 Reserved	198 Various	197 Reserved
059 Reserved	056 US Postal	055 US DoD	052 El duPONT	033 US DoD	034 Haliburton	039 Reserved	038 PSI	217 RIPE	216 ARIN	221 Reserved	222 Reserved	203 APnic	200 Reserved	199 ARIN	196 AfrNIC
060 Reserved	061 APnic	050 Reserved	051 UK DSS	046 Reserved	045 Interop	040 Eli Lily	041 Reserved	214 US DoD	215 US DoD	210 APnic	209 ARIN	204 ARIN	205 ARIN	194 RIPE	195 RIPE
063 ARIN	062 RIPE	049 Reserved	048 Prudential	047 Bell North	044 Radio	043 Inet	042 Reserved	213 RIPE	212 RIPE	211 APnic	208 ARIN	207 ARIN	206 ARIN	192 RIPE	192 Various
064 ARIN	067 Reserved	068 Reserved	069 Reserved	122 Reserved	123 Reserved	124 Reserved	127 Loopback	128 Various	131 Various	132 Various	133 Various	186 Reserved	187 Reserved	188 Reserved	191 Various
065 ARIN	066 ARIN	071 Reserved	070 Reserved	121 Reserved	120 Reserved	125 Reserved	126 Reserved	129 Various	130 Various	135 Various	134 Various	185 Reserved	184 Reserved	189 Reserved	190 Reserved
078 Reserved	077 Reserved	072 Reserved	073 Reserved	118 Reserved	119 Reserved	114 Reserved	113 Reserved	142 Various	141 Various	136 Various	137 Various	182 Reserved	183 Reserved	178 Reserved	177 Reserved
079 Reserved	076 Reserved	075 Reserved	074 Reserved	117 Reserved	116 Reserved	115 Reserved	112 Reserved	143 Various	140 Various	139 Various	138 Various	181 Reserved	180 Reserved	179 Reserved	176 Reserved
080 RIPE	081 RIPE	094 Reserved	095 Reserved	096 Reserved	097 Reserved	110 Reserved	111 Reserved	144 Various	145 Various	158 Various	159 Various	160 Various	161 Various	174 Reserved	175 Reserved
083 Reserved	082 Reserved	093 Reserved	092 Reserved	099 Reserved	098 Reserved	109 Reserved	108 Reserved	147 Various	146 Various	157 Various	156 Various	163 Various	162 Various	173 Reserved	172 Various
084 Reserved	087 Reserved	088 Reserved	091 Reserved	100 Reserved	103 Reserved	104 Reserved	107 Reserved	148 Various	151 Various	152 Various	155 Various	164 Various	167 Various	168 Various	171 Various
085 Reserved	086 Reserved	089 Reserved	090 Reserved	101 Reserved	102 Reserved	105 Reserved	106 Reserved	149 Various	150 Various	153 Various	154 Various	165 Various	166 Various	169 Various	170 Various

IPv4 Address Fractal Map Jan-2003

000 Reserved	001 Reserved	014 PDN	015 HP	016 DEC	019 Ford	020 CsC	021 US DoD	234 Multicast	235 Multicast	236 Multicast	239 Multicast	240 Class E	241 Class E	254 Class E	255 Class E
003 GE	002 Reserved	013 Xerox	012 AT&T	017 Apple	018 MIT	023 Reserved	022 US DoD	233 Multicast	232 Multicast	237 Multicast	238 Multicast	243 Class E	242 Class E	253 Class E	252 Class E
004 L3	007 Reserved	008 L3	011 US DoD	030 US DoD	029 US DoD	024 Cable	025 UK Defense	230 Multicast	231 Multicast	226 Multicast	225 Multicast	244 Class E	247 Class E	248 Class E	251 Class E
005 Reserved	006 US DoD	009 IBM	010 Private	031 Reserved	028 US DoD	027 Reserved	026 US DoD	229 Multicast	228 Multicast	227 Multicast	224 Multicast	245 Class E	246 Class E	249 Class E	250 Class E
058 Reserved	057 SITA	054 Merck	053 Cap Debis	032 AT&T	035 MERIT	036 Reserved	037 Reserved	218 APnic	219 APnic	220 APnic	223 Reserved	202 APnic	201 Reserved	198 Various	197 Reserved
059 Reserved	056 US Postal	055 US DoD	052 El duPONT	033 US DoD	034 Haliburton	039 Reserved	038 PSI	217 RIPE	216 ARIN	221 APnic	222 Reserved	203 APnic	200 Reserved	199 ARIN	196 AfrNIC
060 Reserved	061 APnic	050 Reserved	051 UK DSS	046 Reserved	045 Interop	040 Eli Lily	041 Reserved	214 US DoD	215 US DoD	210 APnic	209 ARIN	204 ARIN	205 ARIN	194 RIPE	195 RIPE
063 ARIN	062 RIPE	049 Reserved	048 Prudential	047 Bell North	044 Radio	043 Inet	042 Reserved	213 RIPE	212 RIPE	211 APnic	208 ARIN	207 ARIN	206 ARIN	192 RIPE	192 Various
064 ARIN	067 ARIN	068 ARIN	069 ARIN	122 Reserved	123 Reserved	124 Reserved	127 Loopback	128 Various	131 Various	132 Various	133 Various	186 Reserved	187 Reserved	188 Reserved	191 Various
065 ARIN	066 ARIN	071 Reserved	070 Reserved	121 Reserved	120 Reserved	125 Reserved	126 Reserved	129 Various	130 Various	135 Various	134 Various	185 Reserved	184 Reserved	189 Reserved	190 Reserved
078 Reserved	077 Reserved	072 Reserved	073 Reserved	118 Reserved	119 Reserved	114 Reserved	113 Reserved	142 Various	141 Various	136 Various	137 Various	182 Reserved	183 Reserved	178 Reserved	177 Reserved
079 Reserved	076 Reserved	075 Reserved	074 Reserved	117 Reserved	116 Reserved	115 Reserved	112 Reserved	143 Various	140 Various	139 Various	138 Various	181 Reserved	180 Reserved	179 Reserved	176 Reserved
080 RIPE	081 RIPE	094 Reserved	095 Reserved	096 Reserved	097 Reserved	110 Reserved	111 Reserved	144 Various	145 Various	158 Various	159 Various	160 Various	161 Various	174 Reserved	175 Reserved
083 Reserved	082 RIPE	093 Reserved	092 Reserved	099 Reserved	098 Reserved	109 Reserved	108 Reserved	147 Various	146 Various	157 Various	156 Various	163 Various	162 Various	173 Reserved	172 Various
084 Reserved	087 Reserved	088 Reserved	091 Reserved	100 Reserved	103 Reserved	104 Reserved	107 Reserved	148 Various	151 Various	152 Various	155 Various	164 Various	167 Various	168 Various	171 Various
085 Reserved	086 Reserved	089 Reserved	090 Reserved	101 Reserved	102 Reserved	105 Reserved	106 Reserved	149 Various	150 Various	153 Various	154 Various	165 Various	166 Various	169 Various	170 Various

Fractal map: Layout by Randall Munroe, Time Sequence by Tony Hain, Highlighted by Jeff Apcar

IPv4 Address Fractal Map Jan-2004

000 Reserved	001 Reserved	014 PDN	015 HP	016 DEC	019 Ford	020 CsC	021 US DoD	234 Multicast	235 Multicast	236 Multicast	239 Multicast	240 Class E	241 Class E	254 Class E	255 Class E
003 GE	002 Reserved	013 Xerox	012 AT&T	017 Apple	018 MIT	023 Reserved	022 US DoD	233 Multicast	232 Multicast	237 Multicast	238 Multicast	243 Class E	242 Class E	253 Class E	252 Class E
004 L3	007 Reserved	008 L3	011 US DoD	030 US DoD	029 US DoD	024 Cable	025 UK Defense	230 Multicast	231 Multicast	226 Multicast	225 Multicast	244 Class E	247 Class E	248 Class E	251 Class E
005 Reserved	006 US DoD	009 IBM	010 Private	031 Reserved	028 US DoD	027 Reserved	026 US DoD	229 Multicast	228 Multicast	227 Multicast	224 Multicast	245 Class E	246 Class E	249 Class E	250 Class E
058 Reserved	057 SITA	054 Merck	053 Cap Debis	032 AT&T	035 MERIT	036 Reserved	037 Reserved	218 APnic	219 APnic	220 APnic	223 Reserved	202 APnic	201 LACnic	198 Various	197 Reserved
059 Reserved	056 US Postal	055 US DoD	052 El duPONT	033 US DoD	034 Haliburton	039 Reserved	038 PSI	217 RIPE	216 ARIN	221 APnic	222 APnic	203 APnic	200 LACnic	199 ARIN	196 AfrNIC
060 Reserved	061 APnic	050 Reserved	051 UK DSS	046 Reserved	045 Interop	040 Eli Lily	041 Reserved	214 US DoD	215 US DoD	210 APnic	209 ARIN	204 ARIN	205 ARIN	194 RIPE	195 RIPE
063 ARIN	062 RIPE	049 Reserved	048 Prudential	047 Bell North	044 Radio	043 Inet	042 Reserved	213 RIPE	212 RIPE	211 APnic	208 ARIN	207 ARIN	206 ARIN	192 RIPE	192 Various
064 ARIN	067 ARIN	068 ARIN	069 ARIN	122 Reserved	123 Reserved	124 Reserved	127 Loopback	128 Various	131 Various	132 Various	133 Various	186 Reserved	187 Reserved	188 Various	191 Various
065 ARIN	066 ARIN	071 Reserved	070 Reserved	121 Reserved	120 Reserved	125 Reserved	126 Reserved	129 Various	130 Various	135 Various	134 Various	185 Reserved	184 Reserved	189 Reserved	190 Reserved
078 Reserved	077 Reserved	072 Reserved	073 Reserved	118 Reserved	119 Reserved	114 Reserved	113 Reserved	142 Various	141 Various	136 Various	137 Various	182 Reserved	183 Reserved	178 Reserved	177 Reserved
079 Reserved	076 Reserved	075 Reserved	074 Reserved	117 Reserved	116 Reserved	115 Reserved	112 Reserved	143 Various	140 Various	139 Various	138 Various	181 Reserved	180 Reserved	179 Reserved	176 Reserved
080 RIPE	081 RIPE	094 Reserved	095 Reserved	096 Reserved	097 Reserved	110 Reserved	111 Reserved	144 Various	145 Various	158 Various	159 Various	160 Various	161 Various	174 Reserved	175 Reserved
083 Reserved	082 RIPE	093 Reserved	092 Reserved	099 Reserved	098 Reserved	109 Reserved	108 Reserved	147 Various	146 Various	157 Various	156 Various	163 Various	162 Various	173 Reserved	172 Various
084 Reserved	087 Reserved	088 Reserved	091 Reserved	100 Reserved	103 Reserved	104 Reserved	107 Reserved	148 Various	151 Various	152 Various	155 Various	164 Various	167 Various	168 Various	171 Various
085 Reserved	086 Reserved	089 Reserved	090 Reserved	101 Reserved	102 Reserved	105 Reserved	106 Reserved	149 Various	150 Various	153 Various	154 Various	165 Various	166 Various	169 Various	170 Various

Fractal map: Layout by Randall Munroe, Time Sequence by Tony Hain, Highlighted by Jeff Apcar

IPv4 Address Fractal Map Jan-2005

000 Reserved	001 Reserved	014 PDN	015 HP	016 DEC	019 Ford	020 CsC	021 US DoD	234 Multicast	235 Multicast	236 Multicast	239 Multicast	240 Class E	241 Class E	254 Class E	255 Class E
003 GE	002 Reserved	013 Xerox	012 AT&T	017 Apple	018 MIT	023 Reserved	022 US DoD	233 Multicast	232 Multicast	237 Multicast	238 Multicast	243 Class E	242 Class E	253 Class E	252 Class E
004 L3	007 Reserved	008 L3	011 US DoD	030 US DoD	029 US DoD	024 Cable	025 UK Defense	230 Multicast	231 Multicast	226 Multicast	225 Multicast	244 Class E	247 Class E	248 Class E	251 Class E
005 Reserved	006 US DoD	009 IBM	010 Private	031 Reserved	028 US DoD	027 Reserved	026 US DoD	229 Multicast	228 Multicast	227 Multicast	224 Multicast	245 Class E	246 Class E	249 Class E	250 Class E
058 APnic	057 SITA	054 Merck	053 Cap Debis	032 AT&T	035 MERIT	036 Reserved	037 Reserved	218 APnic	219 APnic	220 APnic	223 Reserved	202 APnic	201 LACnic	198 Various	197 Reserved
059 APnic	056 US Postal	055 US DoD	052 El duPONT	033 US DoD	034 Haliburton	039 Reserved	038 PSI	217 RIPE	216 ARIN	221 APnic	222 APnic	203 APnic	200 LACnic	199 ARIN	196 AfrNIC
060 APnic	061 APnic	050 Reserved	051 UK DSS	046 Reserved	045 Interop	040 Eli Lily	041 Reserved	214 US DoD	215 US DoD	210 APnic	209 ARIN	204 ARIN	205 ARIN	194 RIPE	195 RIPE
063 ARIN	062 RIPE	049 Reserved	048 Prudential	047 Bell North	044 Radio	043 Inet	042 Reserved	213 RIPE	212 RIPE	211 APnic	208 ARIN	207 ARIN	206 ARIN	192 RIPE	192 Various
064 ARIN	067 ARIN	068 ARIN	069 ARIN	122 Reserved	123 Reserved	124 Reserved	127 Loopback	128 Various	131 Various	132 Various	133 Various	186 Reserved	187 Reserved	188 Various	191 Various
065 ARIN	066 ARIN	071 ARIN	070 Reserved	121 Reserved	120 Reserved	125 Reserved	126 Reserved	129 Various	130 Various	135 Various	134 Various	185 Reserved	184 Reserved	189 Reserved	190 Reserved
078 Reserved	077 Reserved	072 ARIN	073 Reserved	118 Reserved	119 Reserved	114 Reserved	113 Reserved	142 Various	141 Various	136 Various	137 Various	182 Reserved	183 Reserved	178 Reserved	177 Reserved
079 Reserved	076 Reserved	075 Reserved	074 Reserved	117 Reserved	116 Reserved	115 Reserved	112 Reserved	143 Various	140 Various	139 Various	138 Various	181 Reserved	180 Reserved	179 Reserved	176 Reserved
080 RIPE	081 RIPE	094 Reserved	095 Reserved	096 Reserved	097 Reserved	110 Reserved	111 Reserved	144 Various	145 Various	158 Various	159 Various	160 Various	161 Various	174 Reserved	175 Reserved
083 Reserved	082 RIPE	093 Reserved	092 Reserved	099 Reserved	098 Reserved	109 Reserved	108 Reserved	147 Various	146 Various	157 Various	156 Various	163 Various	162 Various	173 Reserved	172 Various
084 Reserved	087 RIPE	088 RIPE	091 Reserved	100 Reserved	103 Reserved	104 Reserved	107 Reserved	148 Various	151 Various	152 Various	155 Various	164 Various	167 Various	168 Various	171 Various
085 Reserved	086 Reserved	089 Reserved	090 Reserved	101 Reserved	102 Reserved	105 Reserved	106 Reserved	149 Various	150 Various	153 Various	154 Various	165 Various	166 Various	169 Various	170 Various

Fractal map: Layout by Randall Munroe, Time Sequence by Tony Hain, Highlighted by Jeff Apcar

IPv4 Address Fractal Map Jan-2006

000 Reserved	001 Reserved	014 PDN	015 HP	016 DEC	019 Ford	020 CsC	021 US DoD	234 Multicast	235 Multicast	236 Multicast	239 Multicast	240 Class E	241 Class E	254 Class E	255 Class E
003 GE	002 Reserved	013 Xerox	012 AT&T	017 Apple	018 MIT	023 Reserved	022 US DoD	233 Multicast	232 Multicast	237 Multicast	238 Multicast	243 Class E	242 Class E	253 Class E	252 Class E
004 L3	007 Reserved	008 L3	011 US DoD	030 US DoD	029 US DoD	024 Cable	025 UK Defense	230 Multicast	231 Multicast	226 Multicast	225 Multicast	244 Class E	247 Class E	248 Class E	251 Class E
005 Reserved	006 US DoD	009 IBM	010 Private	031 Reserved	028 US DoD	027 Reserved	026 US DoD	229 Multicast	228 Multicast	227 Multicast	224 Multicast	245 Class E	246 Class E	249 Class E	250 Class E
058 APnic	057 SITA	054 Merck	053 Cap Debis	032 AT&T	035 MERIT	036 Reserved	037 Reserved	218 APnic	219 APnic	220 APnic	223 Reserved	202 APnic	201 LACnic	198 Various	197 Reserved
059 APnic	056 US Postal	055 US DoD	052 El duPONT	033 US DoD	034 Haliburton	039 Reserved	038 PSI	217 RIPE	216 ARIN	221 APnic	222 APnic	203 APnic	200 LACnic	199 ARIN	196 AfrNIC
060 APnic	061 APnic	050 Reserved	051 UK DSS	046 Reserved	045 Interop	040 Eli Lily	041 Reserved	214 US DoD	215 US DoD	210 APnic	209 ARIN	204 ARIN	205 ARIN	194 RIPE	195 RIPE
063 ARIN	062 RIPE	049 Reserved	048 Prudential	047 Bell North	044 Radio	043 Inet	042 Reserved	213 RIPE	212 RIPE	211 APnic	208 ARIN	207 ARIN	206 ARIN	192 RIPE	192 Various
064 ARIN	067 ARIN	068 ARIN	069 ARIN	122 Reserved	123 Reserved	124 APnic	127 Loopback	128 Various	131 Various	132 Various	133 Various	186 Reserved	187 Reserved	188 Various	191 Various
065 ARIN	066 ARIN	071 ARIN	070 ARIN	121 Reserved	120 Reserved	125 APnic	126 APnic	129 Various	130 Various	135 Various	134 Various	185 Reserved	184 Reserved	189 LACnic	190 LACnic
078 Reserved	077 Reserved	072 ARIN	073 ARIN	118 Reserved	119 Reserved	114 Reserved	113 Reserved	142 Various	141 Various	136 Various	137 Various	182 Reserved	183 Reserved	178 Reserved	177 Reserved
079 Reserved	076 ARIN	075 ARIN	074 ARIN	117 Reserved	116 Reserved	115 Reserved	112 Reserved	143 Various	140 Various	139 Various	138 Various	181 Reserved	180 Reserved	179 Reserved	176 Reserved
080 RIPE	081 RIPE	094 Reserved	095 Reserved	096 Reserved	097 Reserved	110 Reserved	111 Reserved	144 Various	145 Various	158 Various	159 Various	160 Various	161 Various	174 Reserved	175 Reserved
083 RIPE	082 RIPE	093 Reserved	092 Reserved	099 Reserved	098 Reserved	109 Reserved	108 Reserved	147 Various	146 Various	157 Various	156 Various	163 Various	162 Various	173 Reserved	172 Various
084 RIPE	087 RIPE	088 RIPE	091 RIPE	100 Reserved	103 Reserved	104 Reserved	107 Reserved	148 Various	151 Various	152 Various	155 Various	164 Various	167 Various	168 Various	171 Various
085 RIPE	086 RIPE	089 RIPE	090 RIPE	101 Reserved	102 Reserved	105 Reserved	106 Reserved	149 Various	150 Various	153 Various	154 Various	165 Various	166 Various	169 Various	170 Various

Fractal map: Layout by Randall Munroe, Time Sequence by Tony Hain, Highlighted by Jeff Apcar

IPv4 Address Fractal Map Jan-2007

Fractal map: Layout by Randall Munroe, Time Sequence by Tony Hain, Highlighted by Jeff Apcar

000 Reserved	001 Reserved	014 PDN	015 HP	016 DEC	019 Ford	020 CsC	021 US DoD	234 Multicast	235 Multicast	236 Multicast	239 Multicast	240 Class E	241 Class E	254 Class E	255 Class E
003 GE	002 Reserved	013 Xerox	012 AT&T	017 Apple	018 MIT	023 Reserved	022 US DoD	233 Multicast	232 Multicast	237 Multicast	238 Multicast	243 Class E	242 Class E	253 Class E	252 Class E
004 L3	007 Reserved	008 L3	011 US DoD	030 US DoD	029 US DoD	024 Cable	025 UK Defense	230 Multicast	231 Multicast	226 Multicast	225 Multicast	244 Class E	247 Class E	248 Class E	251 Class E
005 Reserved	006 US DoD	009 IBM	010 Private	031 Reserved	028 US DoD	027 Reserved	026 US DoD	229 Multicast	228 Multicast	227 Multicast	224 Multicast	245 Class E	246 Class E	249 Class E	250 Class E
058 APnic	057 SITA	054 Merck	053 Cap Debis	032 AT&T	035 MERIT	036 Reserved	037 Reserved	218 APnic	219 APnic	220 APnic	223 Reserved	202 APnic	201 LACnic	198 Various	197 Reserved
059 APnic	056 US Postal	055 US DoD	052 El duPONT	033 US DoD	034 Haliburton	039 Reserved	038 PSI	217 RIPE	216 ARIN	221 APnic	222 APnic	203 APnic	200 LACnic	199 ARIN	196 AFRnic
060 APnic	061 APnic	050 Reserved	051 UK DSS	046 Reserved	045 Interop	040 Eli Lily	041 AFRNic	214 US DoD	215 US DoD	210 APnic	209 ARIN	204 ARIN	205 ARIN	194 RIPE	195 RIPE
063 ARIN	062 RIPE	049 Reserved	048 Prudential	047 Bell North	044 Radio	043 Inet	042 Reserved	213 RIPE	212 RIPE	211 APnic	208 ARIN	207 ARIN	206 ARIN	192 RIPE	192 Various
064 ARIN	067 ARIN	068 ARIN	069 ARIN	122 APnic	123 APnic	124 APnic	127 Loopback	128 Various	131 Various	132 Various	133 Various	186 Reserved	187 Reserved	188 Various	191 Various
065 ARIN	066 ARIN	071 ARIN	070 ARIN	121 APnic	120 Reserved	125 APnic	126 APnic	129 Various	130 Various	135 Various	134 Various	185 Reserved	184 Reserved	189 LACnic	190 LACnic
078 RIPE	077 RIPE	072 ARIN	073 ARIN	118 Reserved	119 Reserved	114 Reserved	113 Reserved	142 Various	141 Various	136 Various	137 Various	182 Reserved	183 Reserved	178 Reserved	177 Reserved
079 RIPE	076 ARIN	075 ARIN	074 ARIN	117 Reserved	116 Reserved	115 Reserved	112 Reserved	143 Various	140 Various	139 Various	138 Various	181 Reserved	180 Reserved	179 Reserved	176 Reserved
080 RIPE	081 RIPE	094 Reserved	095 Reserved	096 ARIN	097 ARIN	110 Reserved	111 Reserved	144 Various	145 Various	158 Various	159 Various	160 Various	161 Various	174 Reserved	175 Reserved
083 RIPE	082 RIPE	093 Reserved	092 Reserved	099 ARIN	098 ARIN	109 Reserved	108 Reserved	147 Various	146 Various	157 Various	156 Various	163 Various	162 Various	173 Reserved	172 Various
084 RIPE	087 RIPE	088 RIPE	091 RIPE	100 Reserved	103 Reserved	104 Reserved	107 Reserved	148 Various	151 Various	152 Various	155 Various	164 Various	167 Various	168 Various	171 Various
085 RIPE	086 RIPE	089 RIPE	090 RIPE	101 Reserved	102 Reserved	105 Reserved	106 Reserved	149 Various	150 Various	153 Various	154 Various	165 Various	166 Various	169 Various	170 Various

IPv4 Address Fractal Map Jan-2008

000 Reserved	001 Reserved	014 PDN	015 HP	016 DEC	019 Ford	020 CsC	021 US DoD	234 Multicast	235 Multicast	236 Multicast	239 Multicast	240 Class E	241 Class E	254 Class E	255 Class E
003 GE	002 Reserved	013 Xerox	012 AT&T	017 Apple	018 MIT	023 Reserved	022 US DoD	233 Multicast	232 Multicast	237 Multicast	238 Multicast	243 Class E	242 Class E	253 Class E	252 Class E
004 L3	007 Reserved	008 L3	011 US DoD	030 US DoD	029 US DoD	024 Cable	025 UK Defense	230 Multicast	231 Multicast	226 Multicast	225 Multicast	244 Class E	247 Class E	248 Class E	251 Class E
005 Reserved	006 US DoD	009 IBM	010 Private	031 Reserved	028 US DoD	027 Reserved	026 US DoD	229 Multicast	228 Multicast	227 Multicast	224 Multicast	245 Class E	246 Class E	249 Class E	250 Class E
058 APnic	057 SITA	054 Merck	053 Cap Debis	032 AT&T	035 MERIT	036 Reserved	037 Reserved	218 APnic	219 APnic	220 APnic	223 Reserved	202 APnic	201 LACnic	198 Various	197 Reserved
059 APnic	056 US Postal	055 US DoD	052 El duPONT	033 US DoD	034 Haliburton	039 Reserved	038 PSI	217 RIPE	216 ARIN	221 APnic	222 APnic	203 APnic	200 LACnic	199 ARIN	196 AFRnic
060 APnic	061 APnic	050 Reserved	051 UK DSS	046 Reserved	045 Interop	040 Eli Lily	041 AFRNic	214 US DoD	215 US DoD	210 APnic	209 ARIN	204 ARIN	205 ARIN	194 RIPE	195 RIPE
063 ARIN	062 RIPE	049 Reserved	048 Prudential	047 Bell North	044 Radio	043 Inet	042 Reserved	213 RIPE	212 RIPE	211 APnic	208 ARIN	207 ARIN	206 ARIN	192 RIPE	192 Various
064 ARIN	067 ARIN	068 ARIN	069 ARIN	122 APnic	123 APnic	124 APnic	127 Loopback	128 Various	131 Various	132 Various	133 Various	186 LACnic	187 LACnic	188 Various	191 Various
065 ARIN	066 ARIN	071 ARIN	070 ARIN	121 APnic	120 APnic	125 APnic	126 APnic	129 Various	130 Various	135 Various	134 Various	185 Reserved	184 Reserved	189 LACnic	190 LACnic
078 RIPE	077 RIPE	072 ARIN	073 ARIN	118 APnic	119 APnic	114 APnic	113 Reserved	142 Various	141 Various	136 Various	137 Various	182 Reserved	183 Reserved	178 Reserved	177 Reserved
079 RIPE	076 ARIN	075 ARIN	074 ARIN	117 APnic	116 APnic	115 APnic	112 Reserved	143 Various	140 Various	139 Various	138 Various	181 Reserved	180 Reserved	179 Reserved	176 Reserved
080 RIPE	081 RIPE	094 RIPE	095 RIPE	096 ARIN	097 ARIN	110 Reserved	111 Reserved	144 Various	145 Various	158 Various	159 Various	160 Various	161 Various	174 Reserved	175 Reserved
083 RIPE	082 RIPE	093 RIPE	092 RIPE	099 ARIN	098 ARIN	109 Reserved	108 Reserved	147 Various	146 Various	157 Various	156 Various	163 Various	162 Various	173 Reserved	172 Various
084 RIPE	087 RIPE	088 RIPE	091 RIPE	100 Reserved	103 Reserved	104 Reserved	107 Reserved	148 Various	151 Various	152 Various	155 Various	164 Various	167 Various	168 Various	171 Various
085 RIPE	086 RIPE	089 RIPE	090 RIPE	101 Reserved	102 Reserved	105 Reserved	106 Reserved	149 Various	150 Various	153 Various	154 Various	165 Various	166 Various	169 Various	170 Various

Fractal map: Layout by Randall Munroe, Time Sequence by Tony Hain, Highlighted by Jeff Apcar

IPv4 Address Fractal Map Jan-2009

Fractal map: Layout by Randall Munroe, Time Sequence by Tony Hain, Highlighted by Jeff Apcar

000 Reserved	001 Reserved	014 Reserved	015 HP	016 DEC	019 Ford	020 CsC	021 US DoD	234 Multicast	235 Multicast	236 Multicast	239 Multicast	240 Class E	241 Class E	254 Class E	255 Class E
003 GE	002 Reserved	013 Xerox	012 AT&T	017 Apple	018 MIT	023 Reserved	022 US DoD	233 Multicast	232 Multicast	237 Multicast	238 Multicast	243 Class E	242 Class E	253 Class E	252 Class E
004 L3	007 Reserved	008 L3	011 US DoD	030 US DoD	029 US DoD	024 Cable	025 UK Defense	230 Multicast	231 Multicast	226 Multicast	225 Multicast	244 Class E	247 Class E	248 Class E	251 Class E
005 Reserved	006 US DoD	009 IBM	010 Private	031 Reserved	028 US DoD	027 Reserved	026 US DoD	229 Multicast	228 Multicast	227 Multicast	224 Multicast	245 Class E	246 Class E	249 Class E	250 Class E
058 APnic	057 SITA	054 Merck	053 Cap Debis	032 AT&T	035 MERIT	036 Reserved	037 Reserved	218 APnic	219 APnic	220 APnic	223 Reserved	202 APnic	201 LACnic	198 Various	197 Reserved
059 APnic	056 US Postal	055 US DoD	052 EI duPONT	033 US DoD	034 Haliburton	039 Reserved	038 PSI	217 RIPE	216 ARIN	221 APnic	222 APnic	203 APnic	200 LACnic	199 ARIN	196 AFRnic
060 APnic	061 APnic	050 Reserved	051 UK DSS	046 Reserved	045 Interop	040 Eli Lily	041 AFRNic	214 US DoD	215 US DoD	210 APnic	209 ARIN	204 ARIN	205 ARIN	194 RIPE	195 RIPE
063 ARIN	062 RIPE	049 Reserved	048 Prudential	047 Bell North	044 Radio	043 Inet	042 Reserved	213 RIPE	212 RIPE	211 APnic	208 ARIN	207 ARIN	206 ARIN	192 RIPE	192 Various
064 ARIN	067 ARIN	068 ARIN	069 ARIN	122 APnic	123 APnic	124 APnic	127 Loopback	128 Various	131 Various	132 Various	133 Various	186 LACnic	187 LAcnic	188 Various	191 Various
065 ARIN	066 ARIN	071 ARIN	070 ARIN	121 APnic	120 APnic	125 APnic	126 APnic	129 Various	130 Various	135 Various	134 Various	185 Reserved	184 Reserved	189 LACnic	190 LACnic
078 RIPE	077 RIPE	072 ARIN	073 ARIN	118 APnic	119 APnic	114 APnic	113 Next	142 Various	141 Various	136 Various	137 Various	182 Reserved	183 Reserved	178 Next	177 Next
079 RIPE	076 ARIN	075 ARIN	074 ARIN	117 APnic	116 APnic	115 APnic	112 Next	143 Various	140 Various	139 Various	138 Various	181 Reserved	180 Reserved	179 Next	176 Next
080 RIPE	081 RIPE	094 RIPE	095 RIPE	096 ARIN	097 ARIN	110 Next	111 Next	144 Various	145 Various	158 Various	159 Various	160 Various	161 Various	174 Next	175 Next
083 RIPE	082 RIPE	093 RIPE	092 RIPE	099 ARIN	098 ARIN	109 Reserved	108 Reserved	147 Various	146 Various	157 Various	156 Various	163 Various	162 Various	173 Next	172 Various
084 RIPE	087 RIPE	088 RIPE	091 RIPE	100 Next	103 Reserved	104 Reserved	107 Reserved	148 Various	151 Various	152 Various	155 Various	164 Various	167 Various	168 Various	171 Various
085 RIPE	086 RIPE	089 RIPE	090 RIPE	101 Next	102 Reserved	105 Reserved	106 Reserved	149 Various	150 Various	153 Various	154 Various	165 Various	166 Various	169 Various	170 Various

IPv4 Address Fractal Map Jan-2010

000 Reserved	001 Reserved	014 Reserved	015 HP	016 DEC	019 Ford	020 CsC	021 US DoD	234 Multicast	235 Multicast	236 Multicast	239 Multicast	240 Class E	241 Class E	254 Class E	255 Class E
003 GE	002 Reserved	013 Xerox	012 AT&T	017 Apple	018 MIT	023 Reserved	022 US DoD	233 Multicast	232 Multicast	237 Multicast	238 Multicast	243 Class E	242 Class E	253 Class E	252 Class E
004 L3	007 Reserved	008 L3	011 US DoD	030 US DoD	029 US DoD	024 Cable	025 UK Defense	230 Multicast	231 Multicast	226 Multicast	225 Multicast	244 Class E	247 Class E	248 Class E	251 Class E
005 Reserved	006 US DoD	009 IBM	010 Private	031 Reserved	028 US DoD	027 Reserved	026 US DoD	229 Multicast	228 Multicast	227 Multicast	224 Multicast	245 Class E	246 Class E	249 Class E	250 Class E
058 APnic	057 SITA	054 Merck	053 Cap Debis	032 AT&T	035 MERIT	036 Reserved	037 Reserved	218 APnic	219 APnic	220 APnic	223 Reserved	202 APnic	201 LACnic	198 Various	197 Reserved
059 APnic	056 US Postal	055 US DoD	052 El duPONT	033 US DoD	034 Haliburton	039 Reserved	038 PSI	217 RIPE	216 ARIN	221 APnic	222 APnic	203 APnic	200 LACnic	199 ARIN	196 AFRnic
060 APnic	061 APnic	050 Reserved	051 UK DSS	046 Reserved	045 Interop	040 Eli Lily	041 AFRNic	214 US DoD	215 US DoD	210 APnic	209 ARIN	204 ARIN	205 ARIN	194 RIPE	195 RIPE
063 ARIN	062 RIPE	049 Reserved	048 Prudential	047 Bell North	044 Radio	043 Inet	042 Reserved	213 RIPE	212 RIPE	211 APnic	208 ARIN	207 ARIN	206 ARIN	192 RIPE	192 Various
064 ARIN	067 ARIN	068 ARIN	069 ARIN	122 APnic	123 APnic	124 APnic	127 Loopback	128 Various	131 Various	132 Various	133 Various	186 LACnic	187 LAcnic	188 Various	191 Various
065 ARIN	066 ARIN	071 ARIN	070 ARIN	121 APnic	120 APnic	125 APnic	126 APnic	129 Various	130 Various	135 Various	134 Various	185 Next	184 Next	189 LACnic	190 LACnic
078 RIPE	077 RIPE	072 ARIN	073 ARIN	118 APnic	119 APnic	114 APnic	113 Next	142 Various	141 Various	136 Various	137 Various	182 Next	183 Next	178 Next	177 Next
079 RIPE	076 ARIN	075 ARIN	074 ARIN	117 APnic	116 APnic	115 APnic	112 Next	143 Various	140 Various	139 Various	138 Various	181 Next	180 Next	179 Next	176 Next
080 RIPE	081 RIPE	094 RIPE	095 RIPE	096 ARIN	097 ARIN	110 Next	111 Next	144 Various	145 Various	158 Various	159 Various	160 Various	161 Various	174 Next	175 Next
083 RIPE	082 RIPE	093 RIPE	092 RIPE	099 ARIN	098 ARIN	109 Next	108 Next	147 Various	146 Various	157 Various	156 Various	163 Various	162 Various	173 Next	172 Various
084 RIPE	087 RIPE	088 RIPE	091 RIPE	100 Next	103 Next	104 Next	107 Next	148 Various	151 Various	152 Various	155 Various	164 Various	167 Various	168 Various	171 Various
085 RIPE	086 RIPE	089 RIPE	090 RIPE	101 Next	102 Next	105 Next	106 Next	149 Various	150 Various	153 Various	154 Various	165 Various	166 Various	169 Various	170 Various

Fractal map: Layout by Randall Munroe, Time Sequence by Tony Hain, Highlighted by Jeff Apcar

IPv4 Address Fractal Map Jan-2011

000 Reserved	001 Reserved	014 Next	015 HP	016 DEC	019 Ford	020 CsC	021 US DoD	234 Multicast	235 Multicast	236 Multicast	239 Multicast	240 Class E	241 Class E	254 Class E	255 Class E
003 GE	002 Next	013 Xerox	012 AT&T	017 Apple	018 MIT	023 Next	022 US DoD	233 Multicast	232 Multicast	237 Multicast	238 Multicast	243 Class E	242 Class E	253 Class E	252 Class E
004 L3	007 Next	008 L3	011 US DoD	030 US DoD	029 US DoD	024 Cable	025 UK Defense	230 Multicast	231 Multicast	226 Multicast	225 Multicast	244 Class E	247 Class E	248 Class E	251 Class E
005 Next	006 US DoD	009 IBM	010 Private	031 Next	028 US DoD	027 Next	026 US DoD	229 Multicast	228 Multicast	227 Multicast	224 Multicast	245 Class E	246 Class E	249 Class E	250 Class E
058 APnic	057 SITA	054 Merck	053 Cap Debis	032 AT&T	035 MERIT	036 Next	037 Next	218 APnic	219 APnic	220 APnic	223 Next	202 APnic	201 LACnic	198 Various	197 Next
059 APnic	056 US Postal	055 US DoD	052 El duPONT	033 US DoD	034 Haliburton	039 Next	038 PSI	217 RIPE	216 ARIN	221 APnic	222 APnic	203 APnic	200 LACnic	199 ARIN	196 AFRnic
060 APnic	061 APnic	050 Next	051 UK DSS	046 Next	045 Interop	040 Eli Lily	041 AFRNic	214 US DoD	215 US DoD	210 APnic	209 ARIN	204 ARIN	205 ARIN	194 RIPE	195 RIPE
063 ARIN	062 RIPE	049 Next	048 Prudential	047 Bell North	044 Radio	043 Inet	042 Next	213 RIPE	212 RIPE	211 APnic	208 ARIN	207 ARIN	206 ARIN	192 RIPE	192 Various
064 ARIN	067 ARIN	068 ARIN	069 ARIN	122 APnic	123 APnic	124 APnic	127 Loopback	128 Various	131 Various	132 Various	133 Various	186 LANnic	187 LANnic	188 Various	191 Various
065 ARIN	066 ARIN	071 ARIN	070 ARIN	121 APnic	120 APnic	125 APnic	126 APnic	129 Various	130 Various	135 Various	134 Various	185 Next	184 Next	189 LACnic	190 LACnic
078 RIPE	077 RIPE	072 ARIN	073 ARIN	118 APnic	119 APnic	114 APnic	113 Next	142 Various	141 Various	136 Various	137 Various	182 Next	183 Next	178 Next	177 Next
079 RIPE	076 ARIN	075 ARIN	074 ARIN	117 APnic	116 APnic	115 APnic	112 Next	143 Various	140 Various	139 Various	138 Various	181 Next	180 Next	179 Next	176 Next
080 RIPE	081 RIPE	094 RIPE	095 RIPE	096 ARIN	097 ARIN	110 Next	111 Next	144 Various	145 Various	158 Various	159 Various	160 Various	161 Various	174 Next	175 Next
083 RIPE	082 RIPE	093 RIPE	092 RIPE	099 ARIN	098 ARIN	109 Next	108 Next	147 Various	146 Various	157 Various	156 Various	163 Various	162 Various	173 Next	172 Various
084 RIPE	087 RIPE	088 RIPE	091 RIPE	100 Next	103 Next	104 Next	107 Next	148 Various	151 Various	152 Various	155 Various	164 Various	167 Various	168 Various	171 Various
085 RIPE	086 RIPE	089 RIPE	090 RIPE	101 Next	102 Next	105 Next	106 Next	149 Various	150 Various	153 Various	154 Various	165 Various	166 Various	169 Various	170 Various

Fractal map: Layout by Randall Munroe, Time Sequence by Tony Hain, Highlighted by Jeff Apcar

IPv4 Depletion

Stephan Lagerholm predicts[1] IANA IPv4 depletion at:

2011-02-10

and full central IPv4 pool depletion (all RIRs) at

2012-04-15

We're currently allocating a full /8 about every 8 weeks.

[1] http://ipv4depletion.com/dashboard/

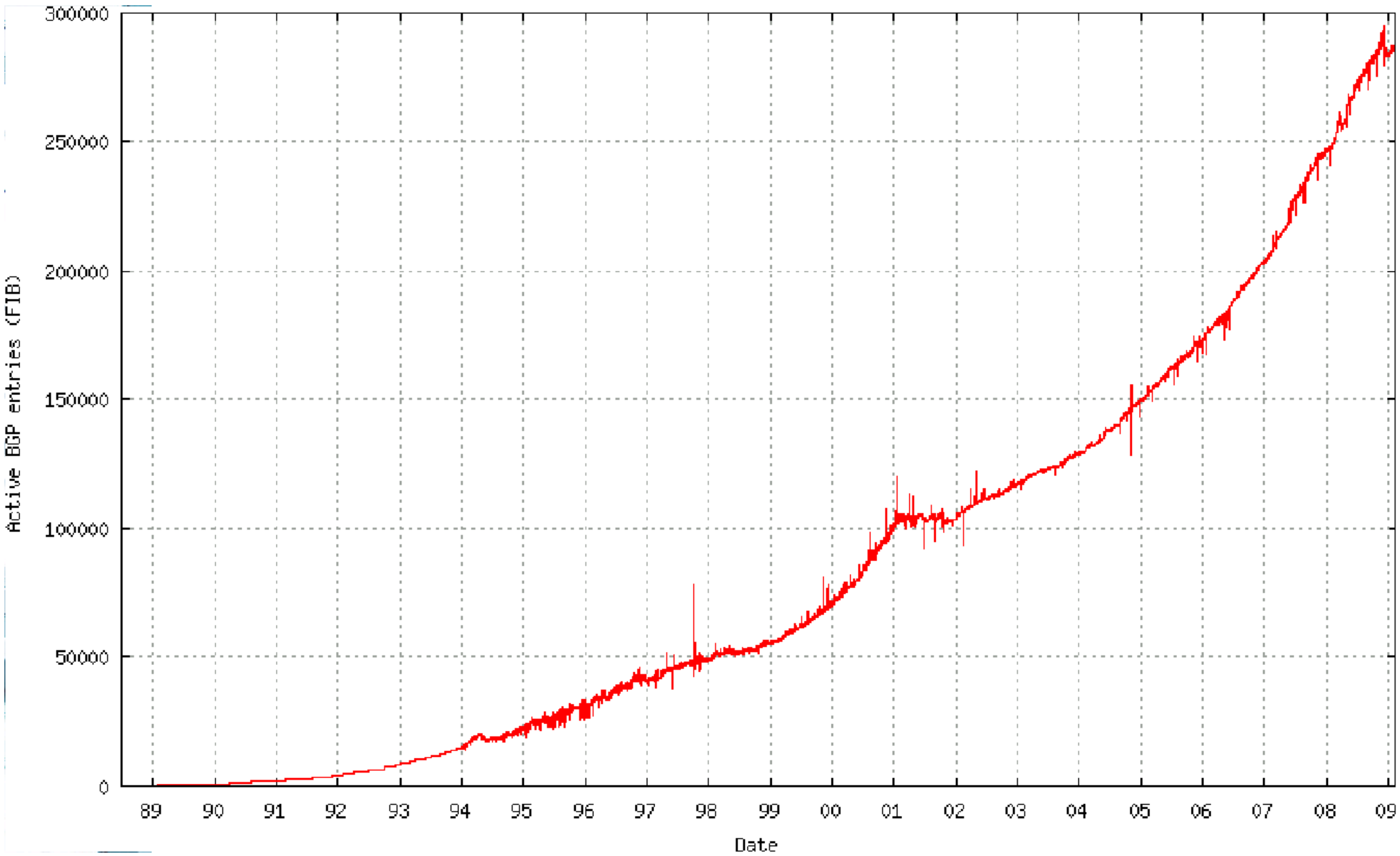
Other Reasons

- Many changes in details for IPv6
- Research $\neq$ KISS
- Can have advantages, but without address space issues nobody would think twice of adopting any of them

Mitigation Risks

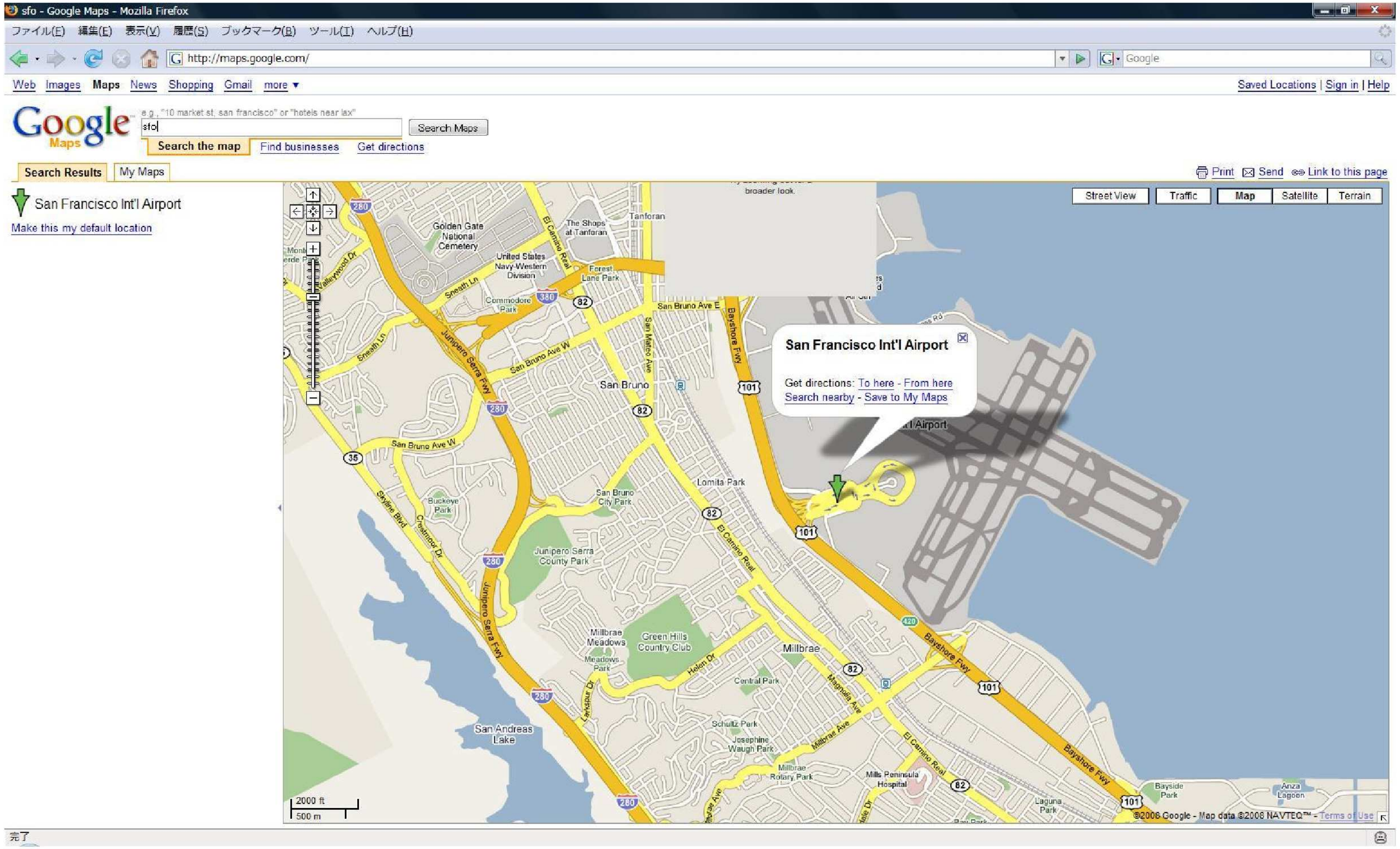
- Using smaller address blocks will cause the global IPv4 routing table to grow in size
- Using NAT limits the number of parallel connections

Current IPv4 BGP Database



Multiple connection applications

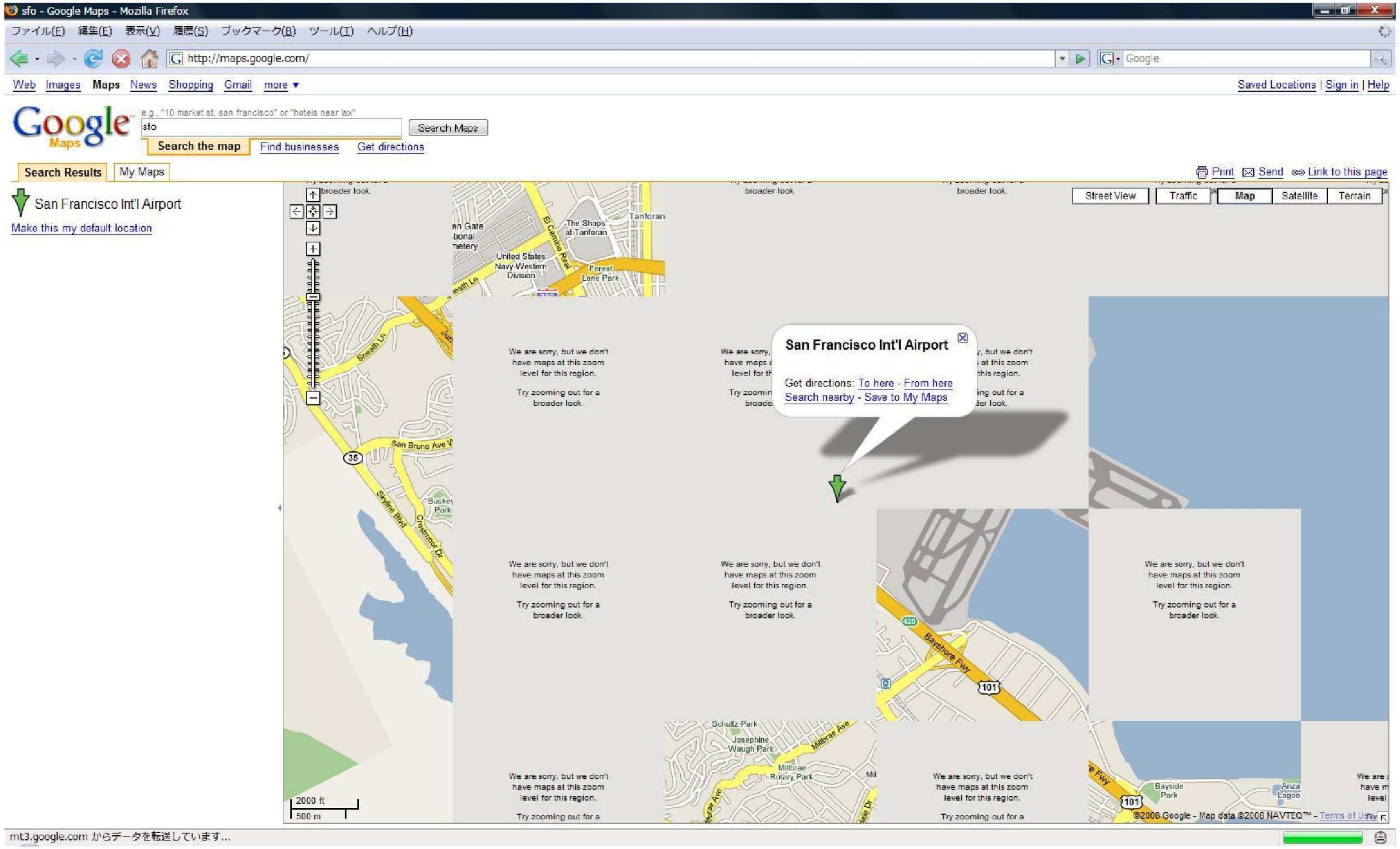
Max 20 Connections



Source: Shin Miyakawa , Ph.D. NTT Communications Corporation

Multiple connection applications

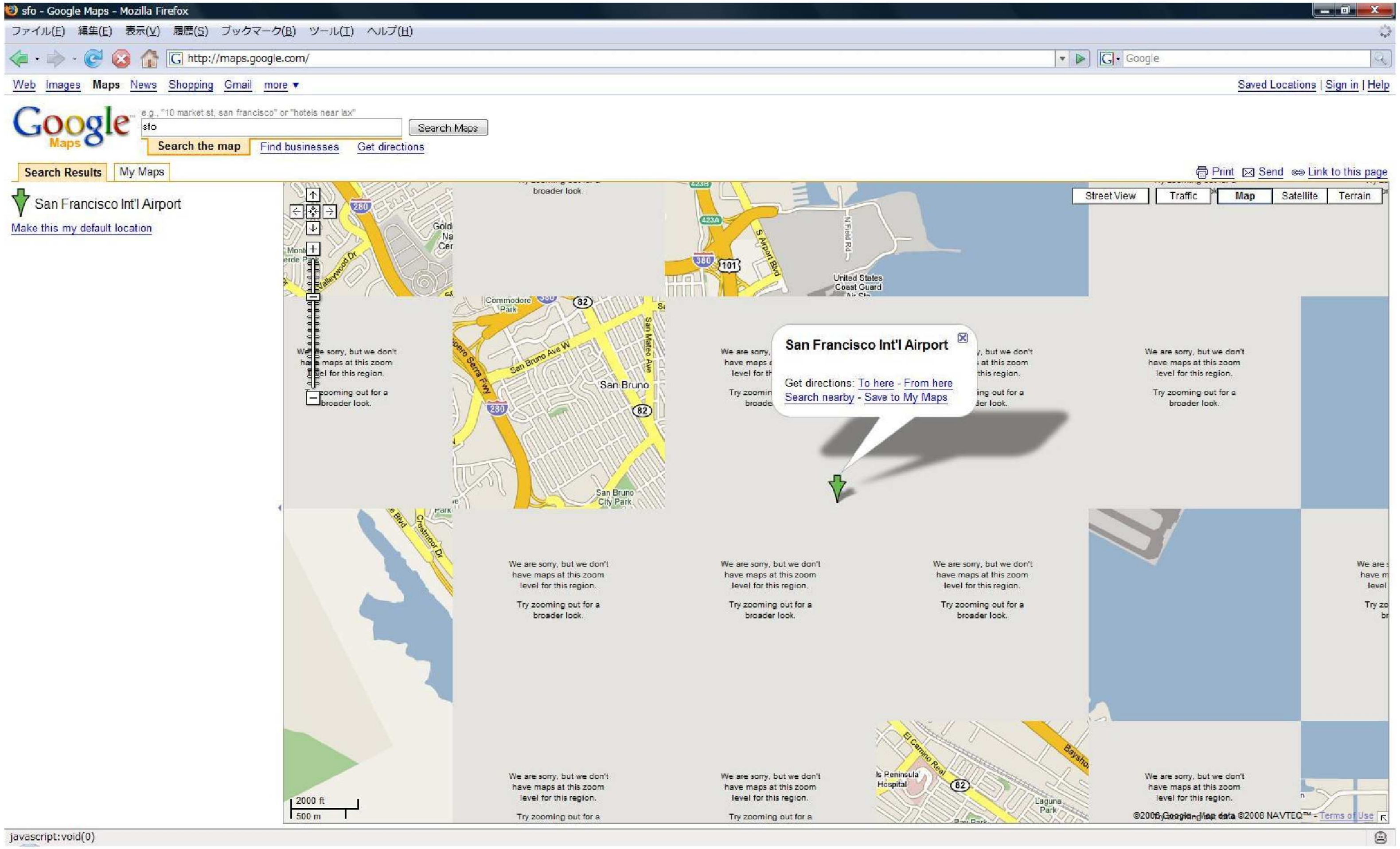
Max 15 Connections



Source: Shin Miyakawa , Ph.D. NTT Communications Corporation

Multiple connection applications

Max 10 Connections



Source: Shin Miyakawa , Ph.D. NTT Communications Corporation

Multiple connection applications

Max 5 Connections

Source: Shin Miyakawa , Ph.D. NTT Communications Corporation

Multiple connection applications

- Google maps opens about 70 parallel connections, iTunes as many as 300

- IPv4/NAT multiplexes users through the port range

$$\frac{64k \quad \texttt{ports}}{300 \quad \texttt{connections}} \approx 200 \quad \texttt{customers per ISP based NAT}$$

The Business Case

- Access providers need more addresses than content providers, so they want to switch once their customers are willing
- Content providers need to deploy IPv6 before access providers can switch, but they don't need that many addresses (thanks to virtual hosting)

The Research Case

- Autoconfiguration, large sensor networks, 6LoWPAN (RFC 4919 & 4944): Routing Over Low power and Lossy networks
- Migration strategies (6over4, 6to4, Teredo, ISATAP, etc.)
- IPv6 multicast `http://www.videolan.org/`, `http://www-mice.cs.ucl.ac.uk/multimedia/software/`
- Moblile IPv6 (RFC 3775 & 4584)

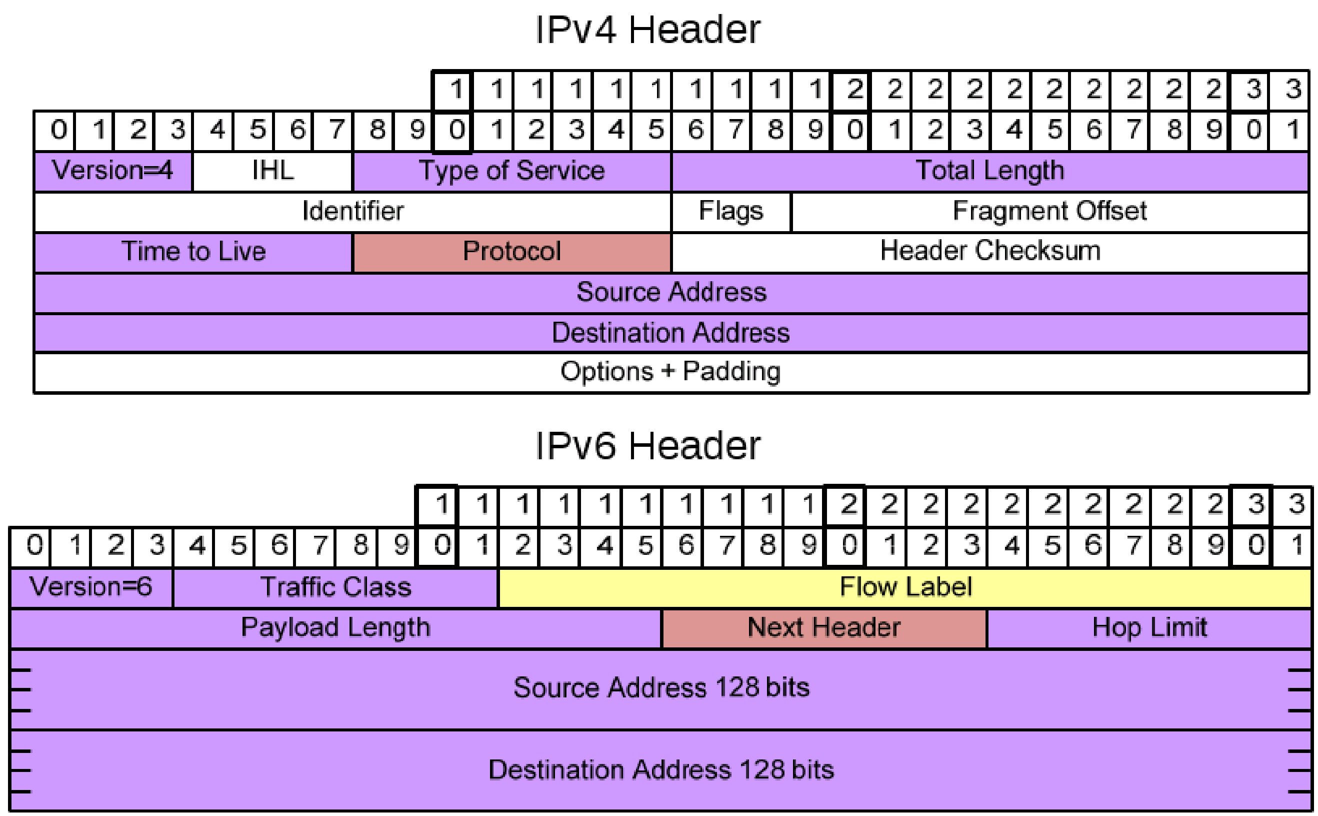
Key Differences between IPv4 and IPv6

- Header
- Fragmentation
- Address Space
- QoS (not discussed today)

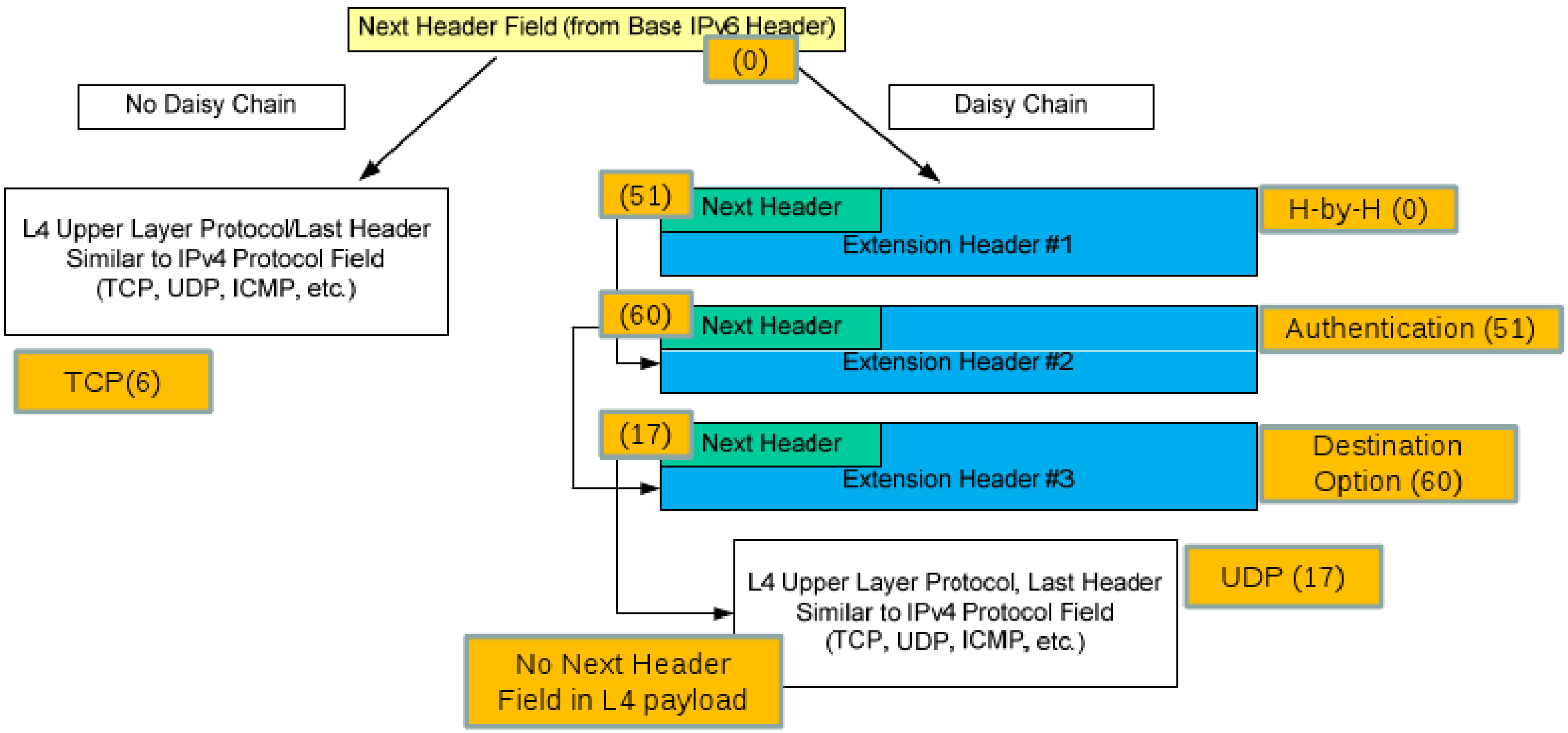
IPv6 Header

- Fixed length (40 bytes) $\Rightarrow$ more efficient
- Fewer fields $\Rightarrow$ more efficient
- No header error checking $\Rightarrow$ more efficient
- Fragmentation fields removed $\Rightarrow$ more efficient
- Aligned on 64-bit boundaries $\Rightarrow$ more efficient
- Extensible via extension header

IPv6 Header


IPv4 Header
Version=4
IHL
Type of Service
Total Length
Identifier
Flags
Fragment Offset
Time to Live
Protocol
Header Checksum
Source Address
Destination Address
Options + Padding
IPv6 Header
Version=6
Traffic Class
Flow Label
Payload Length
Next Header
Hop Limit
Source Address 128 bits
Destination Address 128 bits

IPv6 Extension Headers



Fragmentation

- IPv6 routers do not fragment packets
- IPv6 MTU must be at least 1280 bytes, recommended 1500
- Nodes should implement MTU PD or not exceed 1280 bytes
- MTU path discovery uses ICMPv6 “packet to big” messages

⇒ Do not filter those!

IPv6 Addresses

IPv6 address is 128 bits long:

- First 32 bits typically ISP (::/32)
- First 48 bits typically Enterprise (::/48)
- First 64 bits typically subnet (::/64)
- Low 64 bits often include interface MAC address

Written in Hex, colon breaks into 16-bit "chunks"

Writing IPv6 Addresses

The written format is "<address>/<prefix-length>". Example:

`2001:ABAD:9252:0000:0032:0000:0000:0102/64`

The "/64" in the above example is the number of leftmost bits that constitutes the prefix.

Zeros in IPv6 Addresses

Addresses often contain many 0 (zero) bits. **One** such group can be abbreviated, and leading zeros in each chunk can be dropped:

`2001:ABAD:9252:0:32::0102/64`

IPv6 Address Types

- Unicast
- Multicast
- Anycast

IPv6 Addresses

Address Type	Binary Prefix	IPv6 Notation
Unspecified	$0 \dots 0$	::/128
Loopback	$0 \dots 01$	::1/128
Link-local unicast	1111111010	FE80::/10
Unique Local unicast	1111110	FC00::/7
Site-local unicast	1111111011	FEC0::/10
Multicast	11111111	FF00::/8
Global unicast	(everything else)	

Table 1: Address types and binary representations.

Link-local Addresses

- Only valid on a single link or subnet
- Begin with prefix “FE80::/10”, then contain 54 bits of zeros, followed by the 64-bit interface ID
- Can be automatically generated or manually configured

Unique Local Addresses (RFC 4193)

- Replace site-local unicast which replaced "10.x.x.x" private addresses
- Not routable on Internet; routable within organization
- Site-scoped prefix based on 40 bit hash + 16 bit subnet + interface ID

$\Rightarrow$ Likely globally unique

$\Rightarrow$ Organizations can likely merge without problems

64-bit Interface Identifiers

- Must be unique on the link
- Need not be unique across multiple links
- May be globally unique (i.e., based on MAC address, google EUI-64 construction rules)
- Some IIDs are reserved for subnet-router anycast (all-zeros) and subnet anycast (certain high IIDs)

IPv6 Privacy/Temporary Addresses

- IPv6 autoconfigured addresses can be tracked over time
- IPv6 autoconfigured addresses relate to MAC address

$\Rightarrow$ Location tracking possibility, privacy issues!

Privacy addresses randomize IPv6 address IID so that there is no fixed EIU-64 identifier enabling tracking despite the (possibly) changing /64 prefix.

Multicast Address Format

8 bits FF – Multicast!

4 bits flags, for example:

- 0000 = permanent (IANA)
- 0001 = temporary (local/random)

4 bits scope, for example:

- 0x2 = link-local
- 0x5 = site-local
- 0x8 = organization-local
- 0xE = global

112 bits multicast group ID

Anycast Addresses

- Used to reach a “nearest” instance of a given address
- Drawn from the unicast address space — no special format!
- Should be used for DNS servers

Required Addresses

- Link-local: Required for each interface
- Loopback: Required
- All-Nodes Multicast: Required
- Solicited-Node Multicast: Required for each unicast and anycast address
- Additional unicast, multicast and anycast are optional for hosts
- Router has more, such as “all routers multicast”

ICMPv6

- Router redirect
- Destination unreachable
- Packet too big
- Time exceeded
- Parameter problem
- Echo request/reply
- **Neighbour Discovery** — replace ARP!

Neighbor Discovery Messages

- Neighbour Solicitation uses multicast, not broadcast:

$$2001{:}DB8{::}1234{:}5678{:}9ABC \Rightarrow FF02{::}1{:}FF\mathbf{78{:}9ABC}$$
$$\Rightarrow 33\text{-}33\text{-}FF\text{-}78\text{-}9A\text{-}BC$$

- Router Solicitation uses multicast, can replace DHCP!

Neighbour Solicitation is also used to detect duplicate addresses.

Router Solicitation

When an interface is initialized, it can send a router solicitation instead of waiting for a router advertisement:

33-33-00-00-00-02

Router Advertisement

RAs are sent periodically and on-demand. Include:

- Router lifetime
- Lifetime values for prefixes
- Possibly a hop limit
- Possibly default router preference and specific routes
- Possibly recursive DNS server addresses
- ... or information telling node to use DHCP

DHCPv6

- Similar to DHCP for v4
- “stateless” configuration does not provide addresses (only “other” configuration parameters)
- Can be used to delegate entire prefix (not just single address)
- Currently **no** option to set a host’s default route in the standard! This must be done using RA!

Security Considerations

- Hosts reachable over **two** protocols
- Hosts reachable under **many** addresses
- IPv4 hosts reachable via IPv6 **tunnels** (6over4)

$\Rightarrow$ Traditional Layer-2 firewall rules for IPv4 don't work!

New Attacks

- Abuse of IPv4 compatible addresses
- Abuse of 6to4 addresses
- Abuse of IPv4 mapped addresses
- Attacks by combining different address formats
- Attacks that deplete NAT-PT address pools

Reconnaissance

- Address space is larger, no more ping sweeps
- Ping FF02::1 and neighbor cache will give results for insider!
- Node Information Queries (RFC 4620)
- Stateless auto-configuration makes MITM attack easy by spoofing RAs or DHCPv6
- ICMP redirects (still) exist

Transition Mechanism Threats

- Dual Stack: only as secure as the weaker stack
- Tunnels: 6to4 relay routers are "open relays"

What to do?

"Be liberal in what you accept, and conservative in what you send."
— John Postel, RFC 760.

- Today, organizations are attempting to reach mail and webservers via IPv6
- In the near future, there will be organization that have no choice but to reach you via IPv6

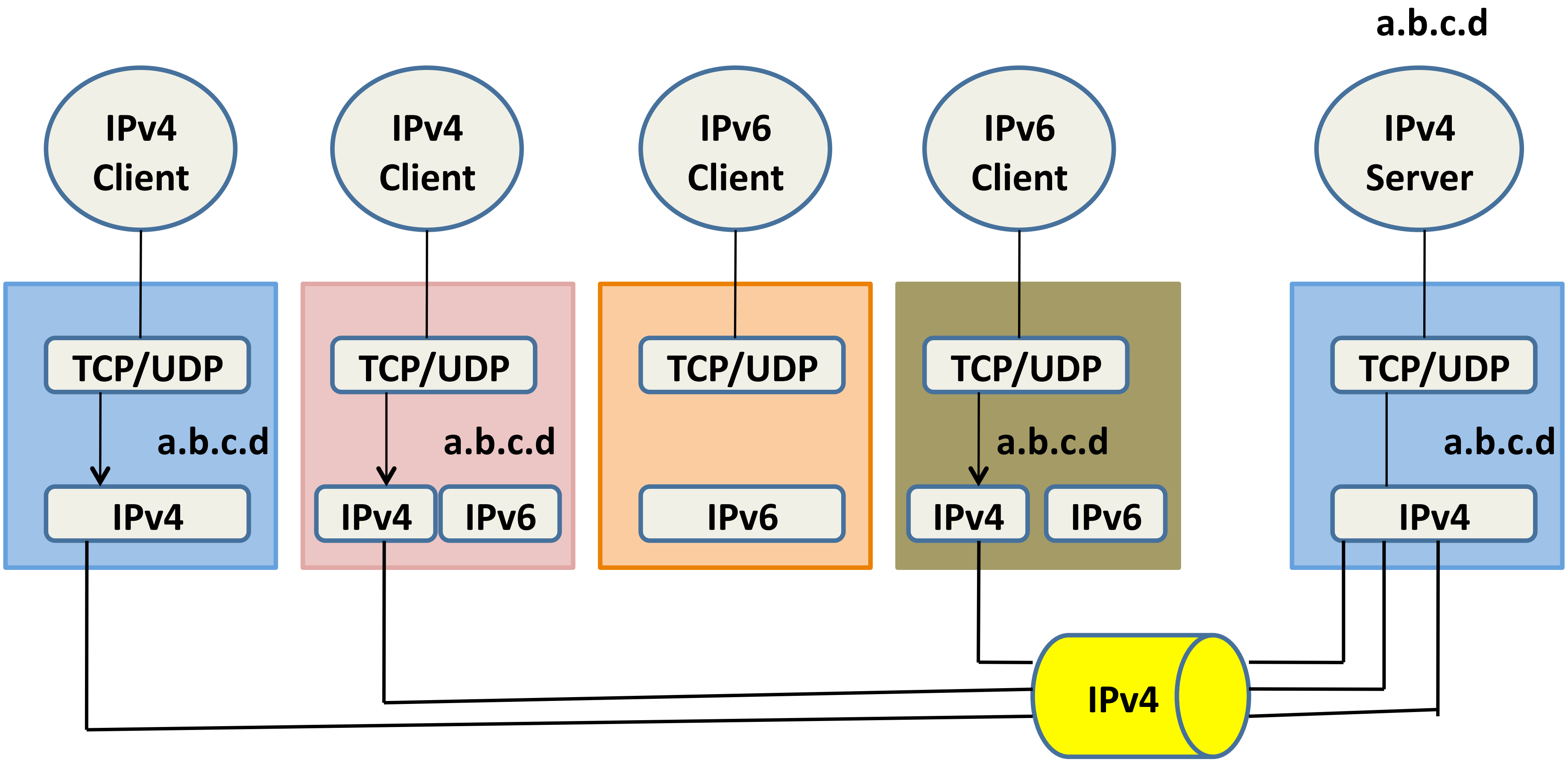
$\Rightarrow$ Dual stack where you can, tunnel where you must

Dual Stack

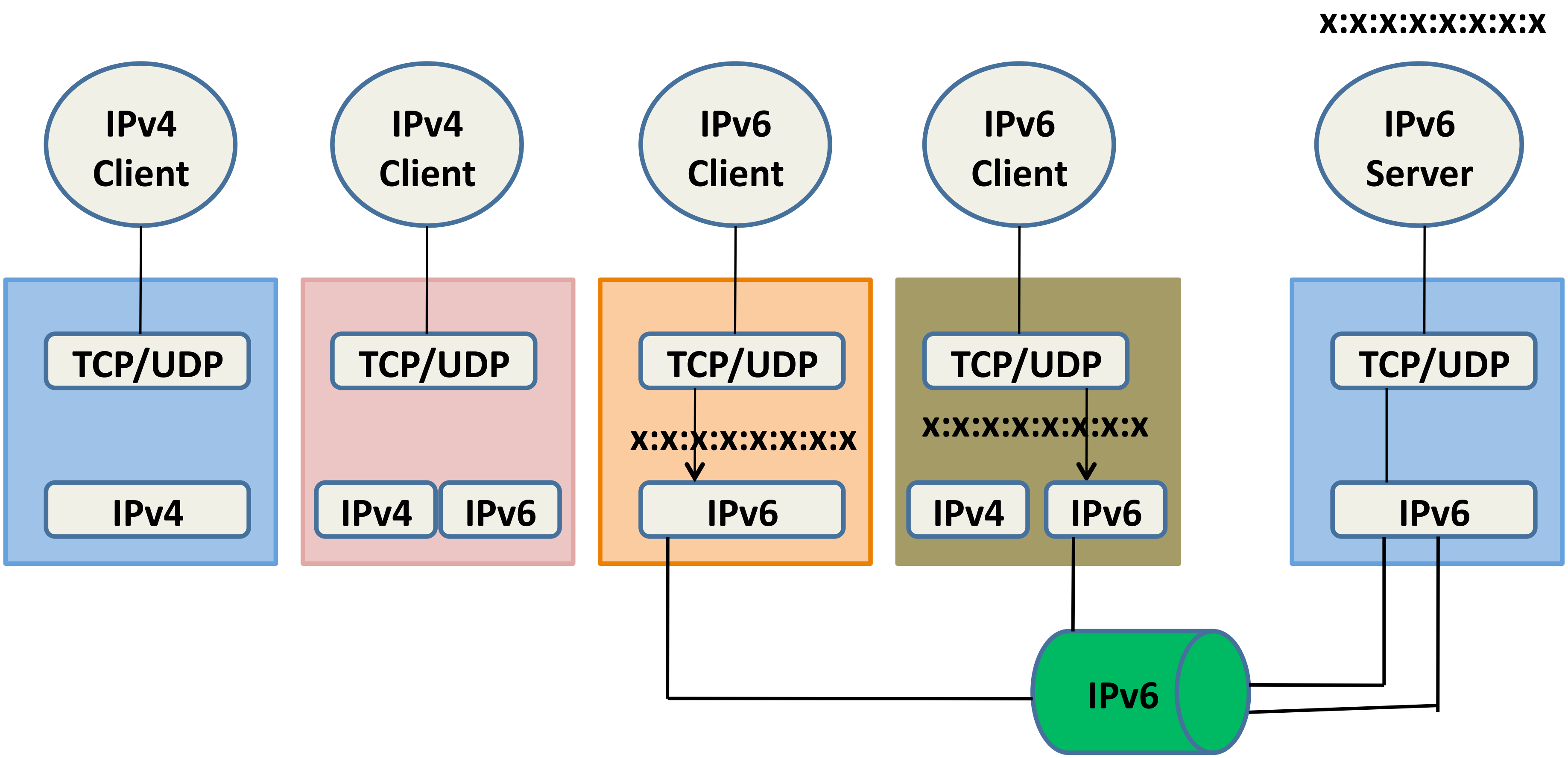
- Evolve the Internet to have two IP versions at the same time
- Interoperate (possibly with limitations) for a while
- Use IPv6 alone in the future

Dual-stacking increases CPU and memory utilization by 15-25% (for routers).

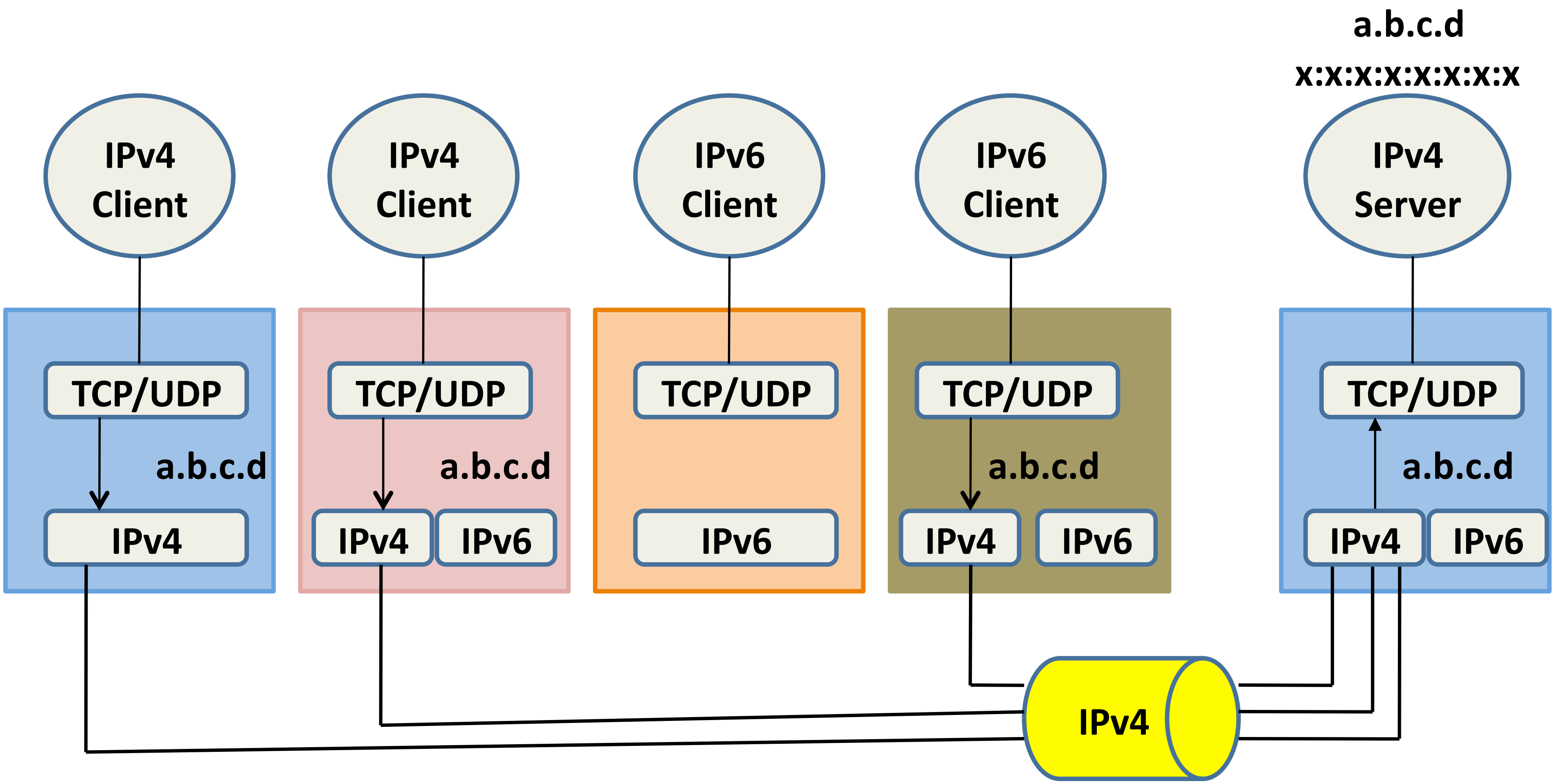
IPv6/IPv4 clients connecting to an IPv4 server at IPv4-only node



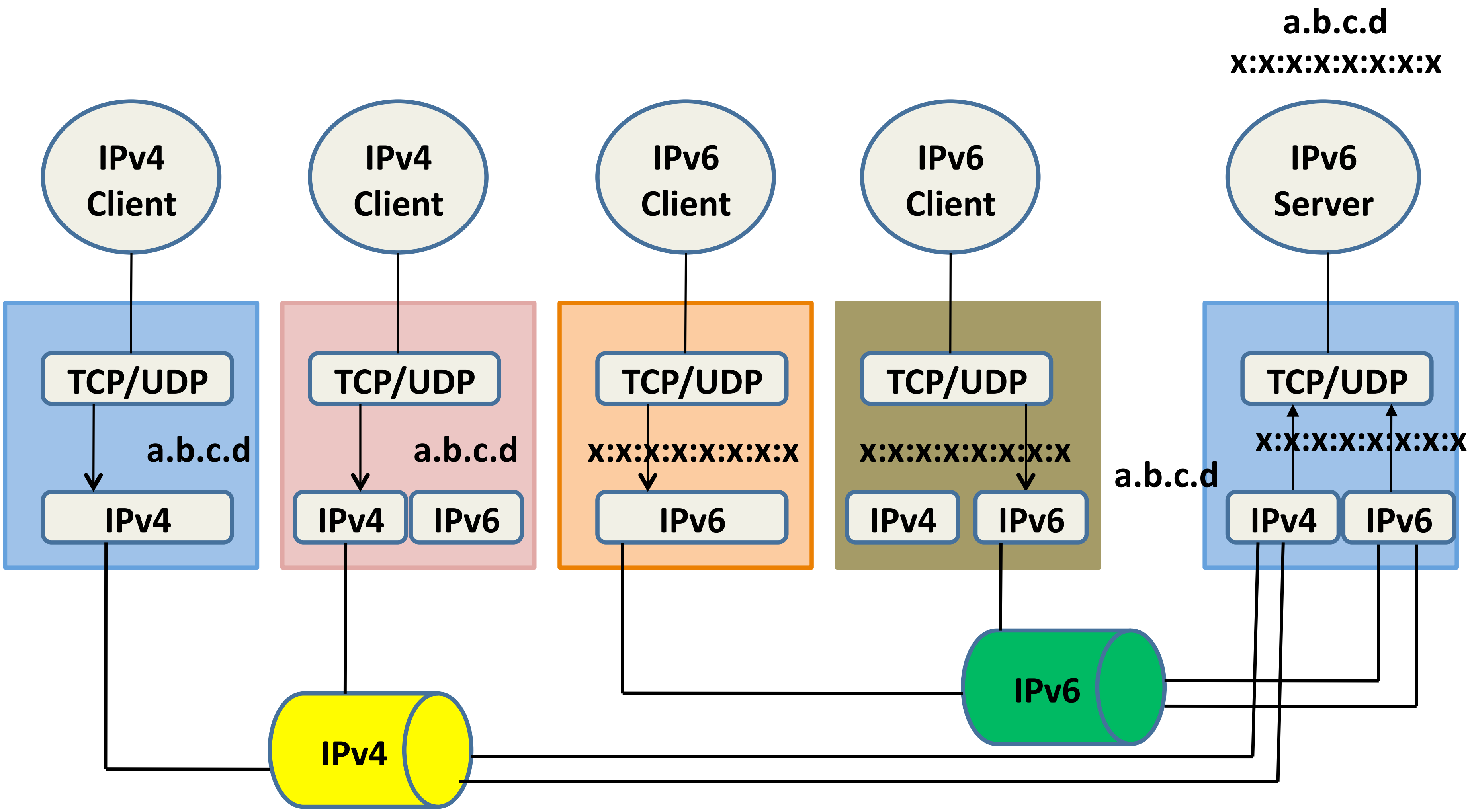
2 IPv6/IPv4 clients connecting to an IPv6 server at IPv6-only node



3 IPv6/IPv4 clients connecting to an IPv4 server at dual stack node



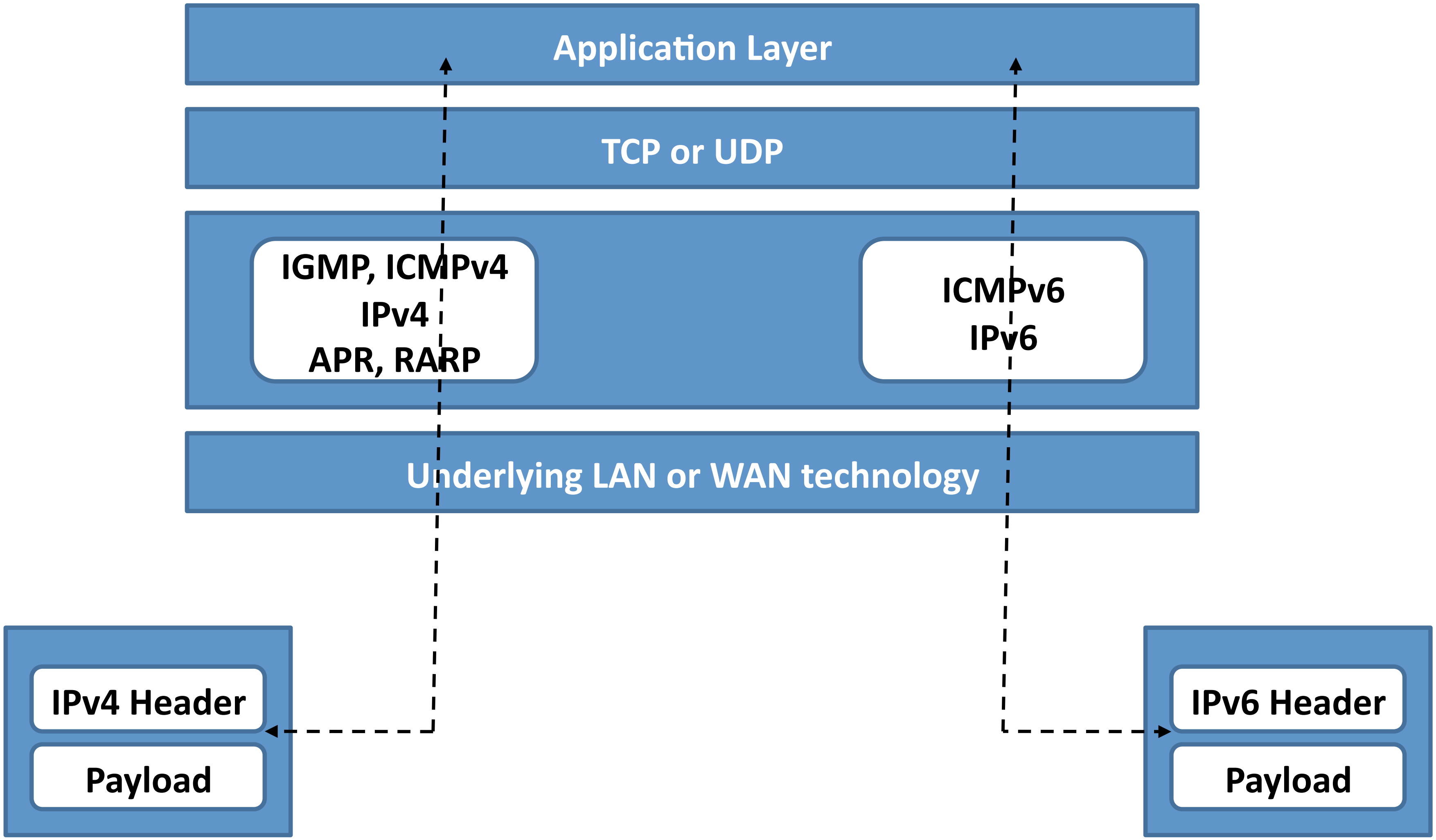
4 IPv6/IPv4 clients connecting to an IPv6 server at dual stack node



Client server & network type combinations

		IPv4 Server Application		IPv6 Server Application	
		IPv4 Node	Dual-Stack	IPv6 node	Dual-Stack
IPv4 client	IPv4 node	IPv4	IPv4	X	IPv4
	Dual-stack	IPv4	IPv4	X	IPv4
IPv6 client	IPv6 node	X	X	IPv6	IPv6
	Dual-stack	IPv4	IPv4/X	IPv6	IPv6

Application Perspective within a Dual Stack

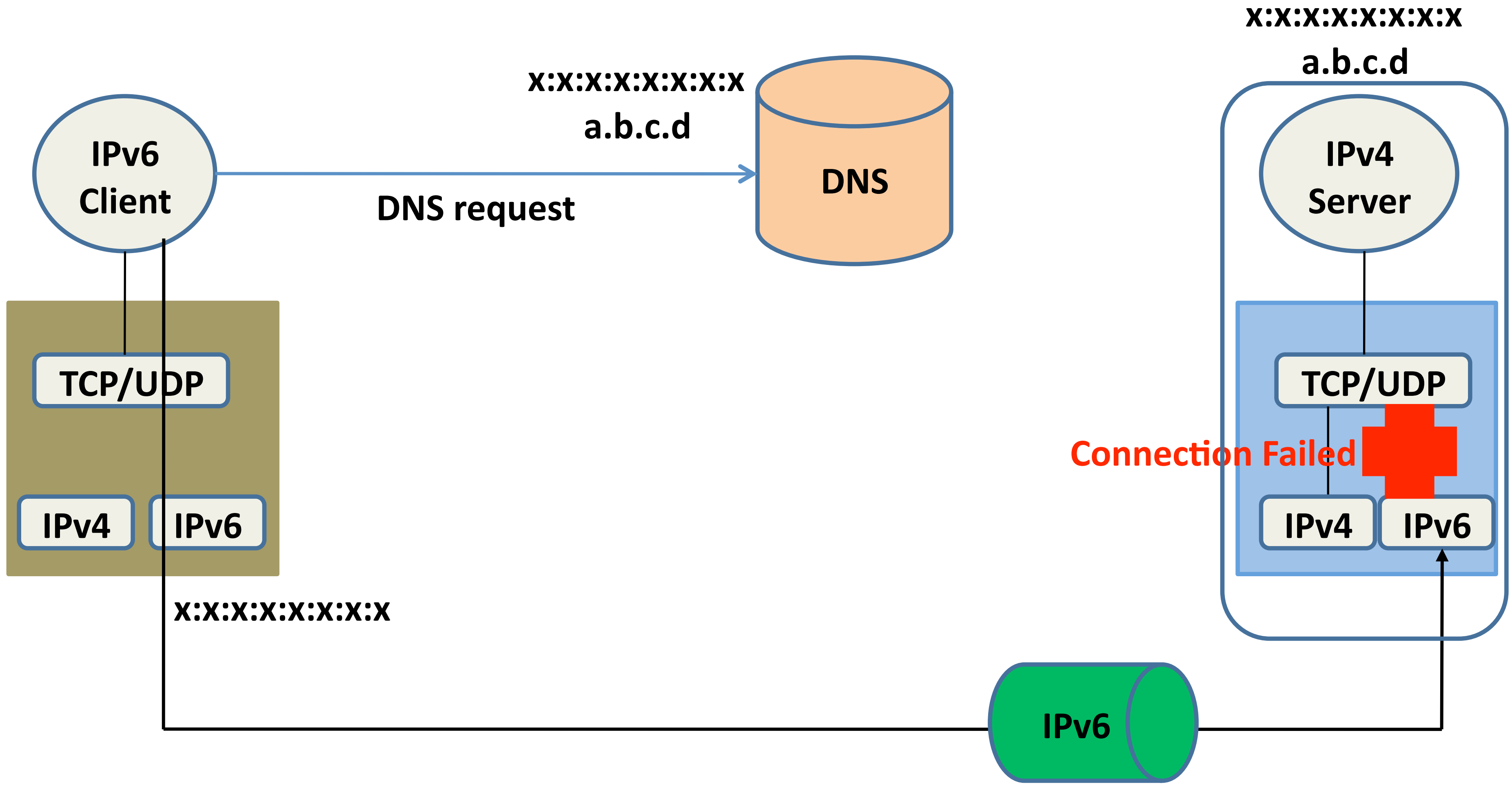


Impact of IPv6 stack on Applications

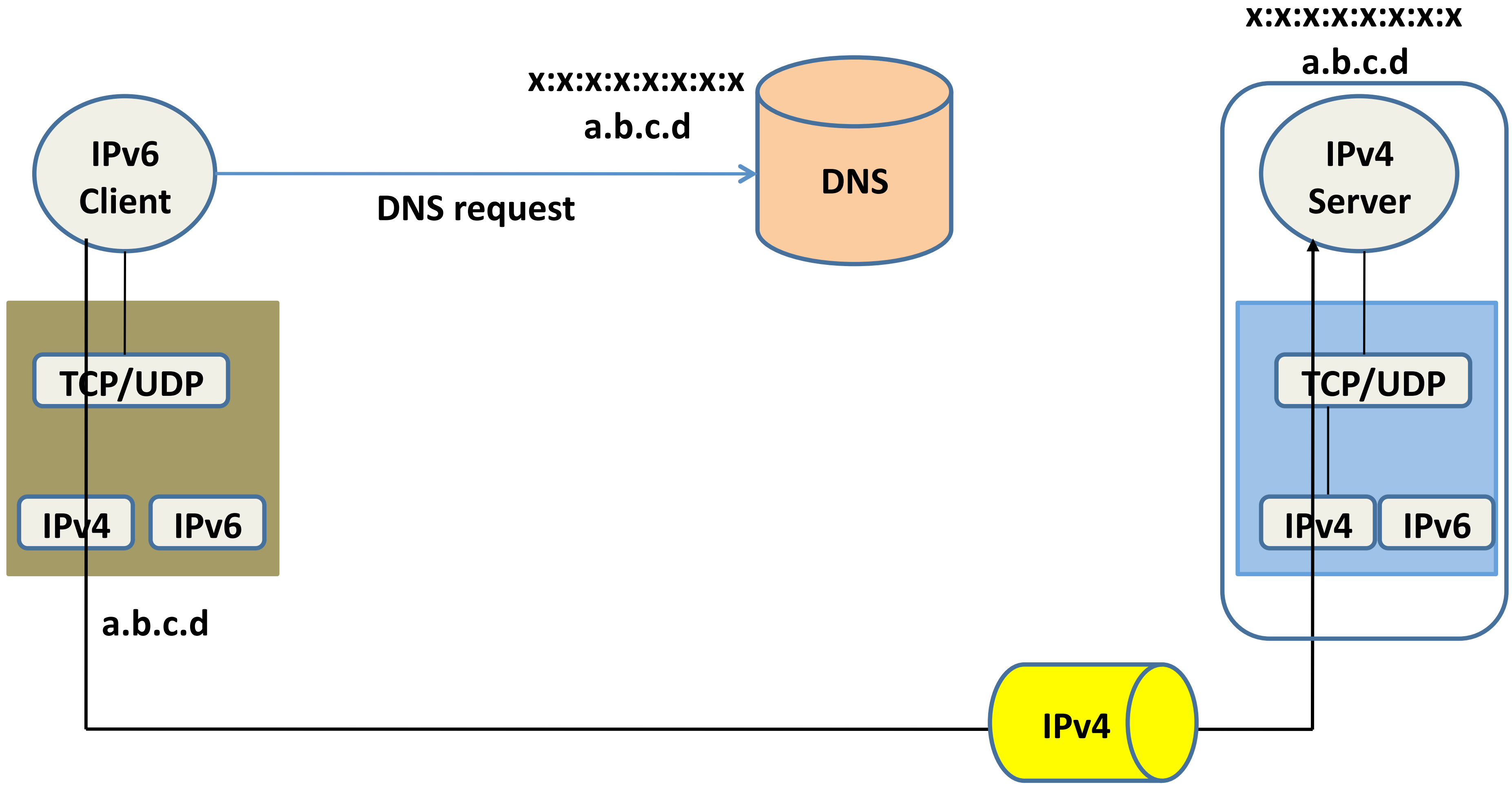
Applications should support dual stack:

- Applications in a dual stack host prever to use IPv6
- In IPv6, it is **normal** to have multiple addresses associated with an interface.
- A configurable default address selection algorithm decides which sender address use (if the application does not specify)
- Applications should try all addresses (both v4 and v6) they get from DNS if necessary

IPv6 enabled client connecting to an IPv4 server at dual stack node



IPv6 enabled client connecting to an IPv4 server at dual stack node



Migrating Code to IPv6

- A minimal example: TCP server and client
- Migration of the minimal example
- DNS, URLs and other migration concerns
- Hard problems
- Checking application IPv6 readiness

Example: minimal IPv4 TCP server

Functionality (as before):

- Listen to port 5002
- Write incoming TCP stream to disk
- Support multiple clients in parallel using `pthreads`

Use of `select` or `epoll` instead of `pthreads` to handle multiple clients never changes anything for IPv6.

Keeping it short...

- No declarations of variables unrelated to IPv4/6
- No error handling code
- Minor details ignored

$\Rightarrow$ Read man-pages to easily fill the gaps

Server Example: processing

```
static void * process (struct T * t) {
  int n;
  char buf[4092];

  int f = creat (filename, S_IRUSR | S_IWUSR);
  while ( (-1 != (n=read (t->a, buf, sizeof (buf)))) &&
          (n != 0) )
    write (f, buf, n);
  close (f);
  close (t->a);
  return NULL;
}
```

IPv4 Server Example: accepting

```
struct sockaddr_in addr;
int s = socket (PF_INET,  SOCK_STREAM, 0);
memset (&addr, 0, sizeof (addr));
struct sockaddr * ia = (struct sockaddr*) &addr;
addr.sin_family = AF_INET; addr.sin_port = htons (5002);
bind (s, ia, sizeof (addr));
listen (s, 5);
while (1) {
  memset (&addr, 0, sizeof (addr));
  socklen_t alen = sizeof (struct sockaddr_in);
  t->a = accept (s, &addr, &alen);
  pthread_create (&pt, NULL, &process, t);           }
```

IPv6 Server Example: accepting

```
struct sockaddr_in6 addr;
int s = socket (PF_INET6, SOCK_STREAM, 0);
memset (&addr, 0, sizeof (addr));
struct sockaddr* ia = (struct sockaddr*) &addr;
addr.sin6_family=AF_INET6; addr.sin6_port= htons (5002);
bind (s, ia, sizeof (addr));
listen (s, 5);
while (1) {
  memset (&addr, 0, sizeof (addr));
  socklen_t alen = sizeof (struct sockaddr_in6);
  t->a = accept (s, &addr, &alen);
  pthread_create (&pt, NULL, &process, t);            }
```

Client Example: processing

```
static void process (int s) {
  char buf[4092];
  int f = open (FILENAME, O_RDONLY);
  while ( (-1 != (n = read (f, buf, sizeof (buf)))) &&
          (n != 0) ) {
    pos = 0;
    while (pos < n) {
      ssize_t got = write (s, &buf[pos], n - pos);
      if (got <= 0) goto END;
      pos += got;                                     } }
END:
  close (f);                                          }
```

IPv4 Client Example

```
struct sockaddr_in addr;
struct sockaddr *ia;

int s = socket (PF_INET,  SOCK_STREAM, 0);
memset (&addr, 0, sizeof (addr));
addr.sin_family = AF_INET;
addr.sin_port = htons (5002);
addr.sin_addr.s_addr = htonl (INADDR_LOOPBACK);
ia = (struct sockaddr *) &addr;
connect (s, ia, sizeof (addr));
process(s);
close (s);
```

IPv6 Client Example

```
struct sockaddr_in6 addr;
struct sockaddr *ia;

int s = socket (PF_INET6, SOCK_STREAM, 0);
memset (&addr, 0, sizeof (addr));
addr.sin6_family= AF_INET6;
addr.sin6_port= htons (5002);
addr.sin6_addr = in6addr_loopback;
ia = (struct sockaddr*) &addr;
connect (s, ia, sizeof (addr));
process(s);
close (s);
```

What are we missing?

What about...

- ... running on an OS that does not support IPv6?
- ... parsing user-specified addresses?
- ... IP-based access control?
- ... DNS resolution?
- ... URL support?

Levels of OS support

The OS could:

- Lack basic IPv6 definitions in the C libraries (i.e., no `PF_INET6` constant defined)
- Have support in the C libraries but lack kernel support (IPv6 operations fail)
- Have kernel support enabled but only use IPv4 addresses for networking (some IPv6 operations succeed)
- Use IPv4 and IPv6 for networking, possibly depending on the interface
- Only use IPv6

Handling lack of IPv6 OS support (1/2)

```
  int v6 = 0;
  int s = -1;
#if HAVE_INET6_DEFINES
  s = socket (PF_INET6, SOCK_STREAM, 0);
  if (s != -1)
    v6 = 1;
  else
#endif
    s = socket (PF_INET4, SOCK_STREAM, 0);
  memset (&addr, 0, sizeof (addr));
```

Handling lack of IPv6 OS support (2/2)

```
#if HAVE_INET6_DEFINES
  if (v6 == 1) {
    ia6.sin_family = AF_INET6;
    socklen = sizeof(struct sockaddr_in6);
    addr = (struct sockaddr_in*) &ia6;
  } else
#endif
  { ia4.sin_family = AF_INET;
    socklen = sizeof(struct sockaddr_in);
    addr = (struct sockaddr_in*) &ia4;
  }
  connect (s, &addr, socklen);
```

IP-based access control

- Bind socket to limited IP addresses
- Check that connection is from trusted network
- Check that IP matches certain DNS names

IPv4 Server Example: loopback only

```
struct sockaddr_in ia;
int s = socket (PF_INET,  SOCK_STREAM, 0);
memset (&ia, 0, sizeof (ia));
ia.sin_family = AF_INET;
ia.sin_addr.s_addr = htonl (INADDR_LOOPBACK);
ia.sin_port = htons (5002);
struct sockaddr *addr = (struct sockaddr *)&ia;
bind (s, addr, sizeof (ia));
// ...
```

IPv6 Server Example: loopback only

```
struct sockaddr_in6 ia;
int s = socket (PF_INET6, SOCK_STREAM, 0);
memset (&ia, 0, sizeof (ia));
ia.sin6_family= AF_INET6;
ia.sin6_addr = inaddr6_loopback;
ia.sin6_port= htons (5002);
struct sockaddr* addr = (struct sockaddr*)&ia;
bind (s, addr, sizeof (ia));
// ...
```

Parsing IPv4 addresses

```
int parse_v4(const char * in,
             struct in_addr * out) {
  int ret = inet_pton(AF_INET,  in, out);
  if (ret < 0)
    fprintf(stderr, "AF_INET not supported!\n");
  else if (ret == 0)
    fprintf(stderr, "Syntax error!\n");
  else
    return 0;
  return -1;
}
```

Parsing IPv6 addresses

```
int parse_v6(const char * in,
             struct in6_addr * out) {
  struct in_addr v4;
  int ret = inet_pton(AF_INET6, in, out);
  if (ret > 0) return 0;
  ret = inet_pton(AF_INET, in, &v4);
  if (ret < 0) return -1; /* error */
  memset(out, 0, sizeof(struct in6_addr));
  ((unsigned int *) out)[2] = htonl (0xffff);
  memcpy (&((char *) out)[sizeof (struct in6_addr) -
                          sizeof (struct in_addr)],
          &v4, sizeof (struct in_addr)); return 0; }
```

IPv4 network check

```
int
test_in_network_v4 (const struct in_addr * network,
                    const struct in_addr * mask,
                    const struct in_addr * addr) {
  return ( (addr->s_addr & mask.s_addr)
           == network.s_addr & mask.s_addr)
}
```

IPv6 network check

```
int test_in_network_v6 (const struct in6_addr * network,
                        const struct in6_addr * mask,
                        const struct in6_addr * addr) {
  unsigned int i;
  for (i=0; i<sizeof(struct in6_addr)/sizeof (int); i++)
    if ( ((((int *) ip     )[i] & ((int *) mask)[i])) !=
         (((int *) network)[i] & ((int *) mask)[i]))
      return 0;
  return 1;
}
```

Generic network check

```
int test (struct in_addr * n4, struct in_addr * m4,
          struct in6_addr* n6, struct in6_addr* m6,
          struct in6_addr * addr) {
  struct in_addr ip4;
  if (test_in_network_v6(n6, m6, addr)) return 1;
  memcpy (&ip4, &((char *) &ip6)
     [sizeof(struct in6_addr)-sizeof(struct in_addr)],
     sizeof (struct in_addr));
  if (IN6_IS_ADDR_V4MAPPED (&a6->sin6_addr))
    return test_in_network_v4(n4, m4, addr);
  return 0; }
```

IPv4 DNS request

```
int
resolve_v4 (const char * hostname,
            struct in_addr * addr) {
  struct hostent * he;
  struct sockaddr_in *addr;
  he = gethostbyname(hostname);
  assert (he->h_addrtype == AF_INET);
  assert (hp->h_length == sizeof (struct in_addr));
  memcpy (addr, hp->h_addr_list[0], hp->h_length);
  return OK;
}
```

`gethostbyname` issues

- Synchronous
- IPv4 only

⇒ `gethostbyname2`

`gethostbyname` issues

- Synchronous
- IPv4 only

⇒ `gethostbyname2`

- Not reentrant

⇒ both are obsolete!

DNS request with getaddrinfo

```
void resolve_v6 (const char * hostname) {
  struct addrinfo hints;
  struct addrinfo *result;
  memset (&hints, 0, sizeof (struct addrinfo));
  hints.ai_family = AF_INET6;
  getaddrinfo (hostname, NULL, &hints, &result);
  process_result (result);
  freeaddrinfo (result);
}
```

Processing DNS reply from `getaddrinfo`

```
void process_result (const struct addrinfo *pos) {
 for (;NULL != pos;pos = pos->ai_next) {
  switch (pos->ai_family) {
   case AF_INET : if (OK == tryv4
     ((struct sockaddr_in  *) pos->ai_addr)) return;
     break;
   case AF_INET6: if (OK == tryv6
     ((struct sockaddr_in6 *) pos->ai_addr)) return;
     break;
 }  }
 fail(); }
```

Generic Client Example

```
struct sockaddr * addr;
resolve(HOSTNAME, &addr, &alen, &af);
s = socket (af == AF_INET ? PF_INET : PF_INET6,
            SOCK_STREAM, 0);
if (af == AF_INET)
  ((struct sockaddr_in*)addr)->sin_port=htons (5002);
else
  ((struct sockaddr_in6*)addr)->sin6_port=htons (5002);
connect (s, addr, alen);
process(s);
free(addr); close (s);
```

URL support

- IPv4: `http://127.0.0.1:8080/`

URL support

- IPv4: `http://127.0.0.1:8080/`
- IPv6: `http://::1:8080/` – does not work!

URL support

- IPv4: `http://127.0.0.1:8080/`
- IPv6: `http://::1:8080/` – does not work!
- Solution: `http://[::1]:8080/`

Other considerations

- Use `getnameinfo` instead of `gethostbyaddr` for reverse lookup
- Check if your system uses IPv4 binary addresses embedded in network protocols
- You must specify the interface if you use IPv6 link local addresses (or do not use them!)
- Check IPv6 support in libraries (GNU ADNS does not support IPv6!)

IPv6 and Infrastructure

- IPv6 clients talking to IPv4-only server
- IPv4 clients talking to IPv6-only server
- Improved security / new IPv6 options:
  - Some new options require using raw sockets
  - Compatibility and migration nightmare
  - Applications already use SSL/IPsec

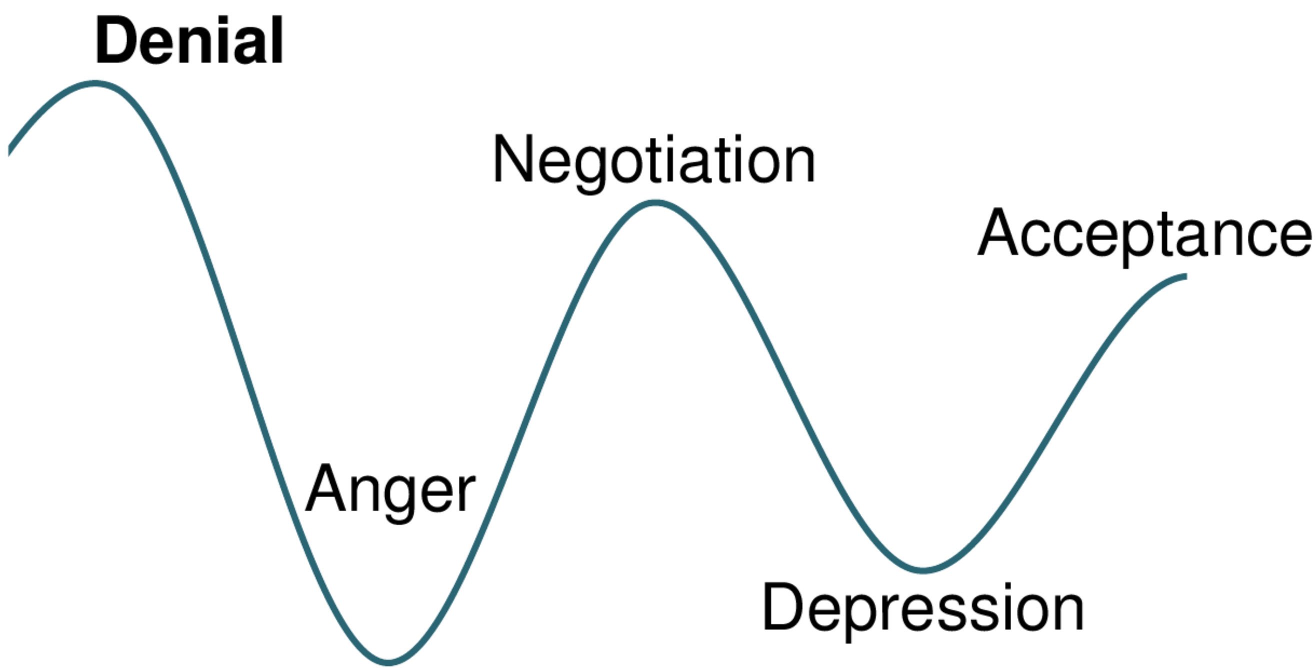
$\Rightarrow$ Rarely supported (nicely) by OS

Are we done yet?

On a GNU/Linux system, run:

- `$ netstat -nl`

The Stages of Grief



"Misery motivates, not utopia." – Karl Marx

Questions

?

Acknowledgements

Thanks to John Curran, Tony Hain, Carl Williams, John Spence and Scott Hogg for ideas and slides.

Copyright

Master Course
Computer Networks
IN2097

Prof. Dr.-Ing. Georg Carle
Christian Grothoff, Ph.D.

Chair for Network Architectures and Services
Institut für Informatik
Technische Universität München
http://www.net.in.tum.de

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- **IPv6**
- Virtual circuit and datagram networks
- What's inside a router
- NAT

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
- Broadcast and multicast routing

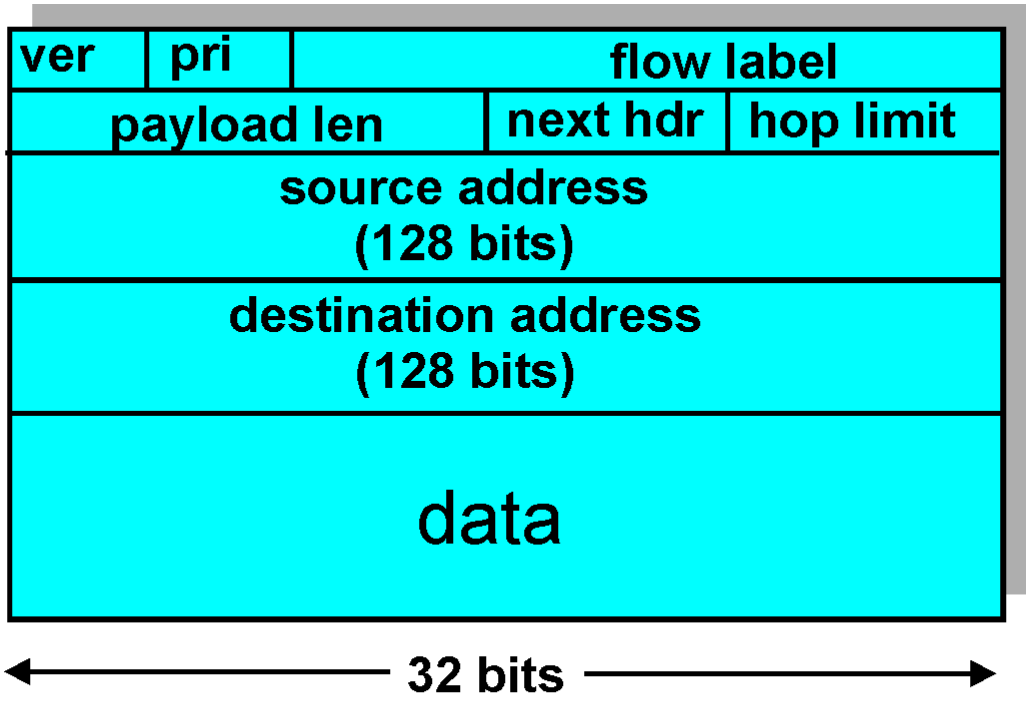
IPv6

- Initial motivation: 32-bit address space soon to be completely allocated.
- Additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

  IPv6 datagram format:
  - fixed-length 40 byte header
  - no fragmentation allowed

IPv6 Header (Cont)

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



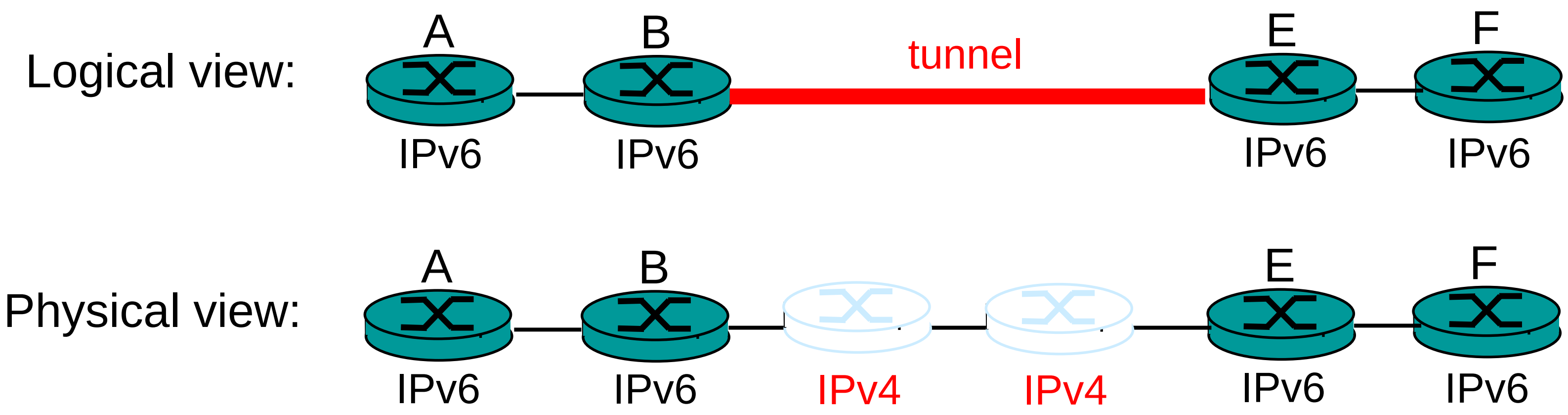
Other Changes from IPv4

- *Checksum*: removed entirely to reduce processing time at each hop
- *Options:* allowed, but outside of header, indicated by “Next Header” field
- *ICMPv6:* new version of ICMP
  - additional message types, e.g. “Packet Too Big”
  - multicast group management functions

Transition From IPv4 To IPv6

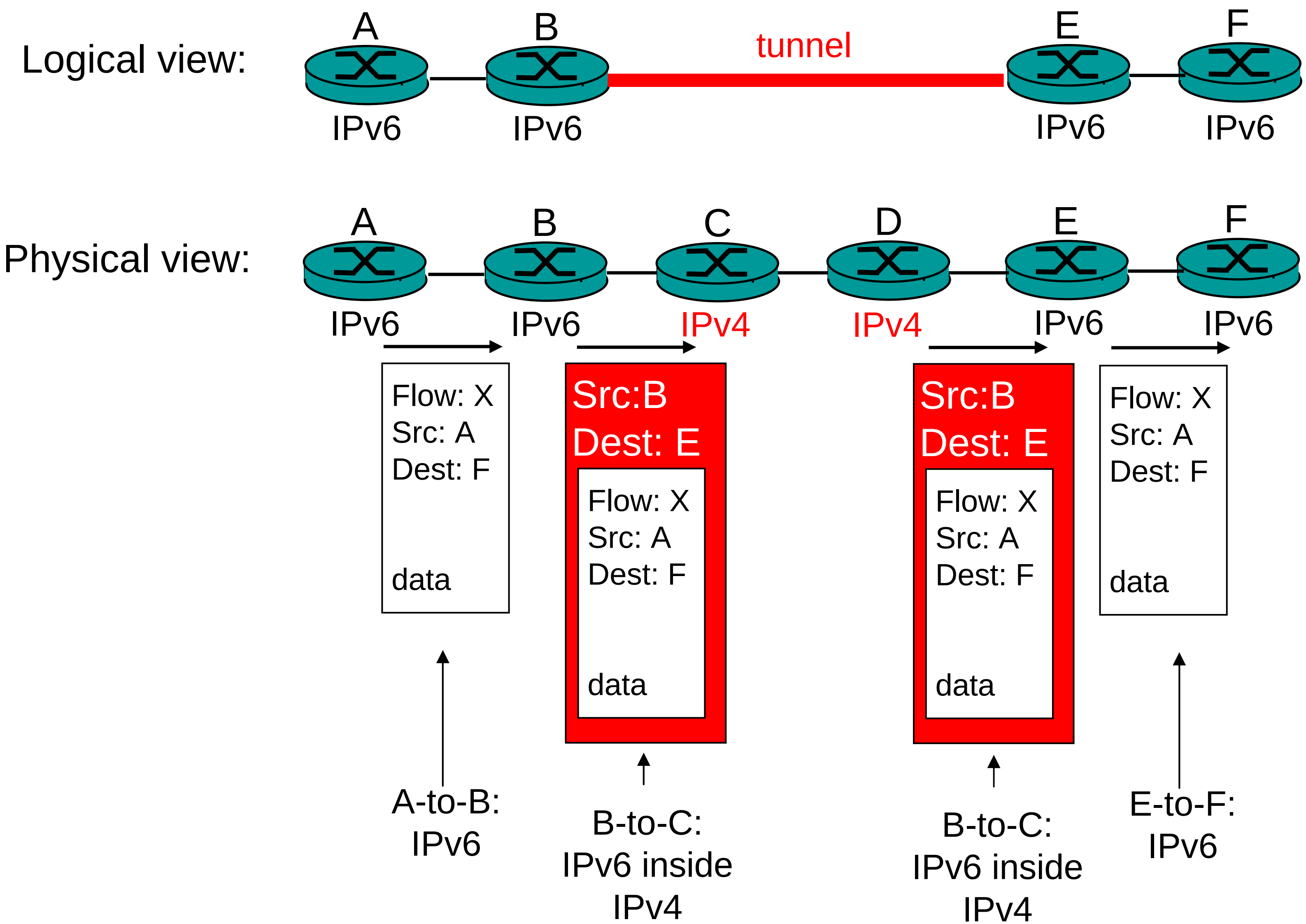
- Not all routers can be upgraded simultaneous
  - no “flag days”
  - How will the network operate with mixed IPv4 and IPv6 routers?
- *Tunneling:* IPv6 carried as payload in IPv4 datagram among IPv4 routers

Tunneling



Logical view:
A
B
tunnel
E
F
IPv6
IPv6
IPv6
IPv6
Physical view:
A
B
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6

Tunneling



Logical view:
A
B
tunnel
E
F
IPv6
IPv6
IPv6
IPv6
Physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
Flow: X
Src: A
Dest: F
data
Src:B
Dest: E
Flow: X
Src: A
Dest: F
data
Src:B
Dest: E
Flow: X
Src: A
Dest: F
data
Flow: X
Src: A
Dest: F
data
A-to-B:
IPv6
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
E-to-F:
IPv6

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- **Virtual circuit and datagram networks**
- What's inside a router
- NAT

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
- Broadcast and multicast routing

Connection setup

- In addition to routing and forwarding, 3rd important function in some network architectures:
  - ATM, frame relay, X.25
- before datagrams flow, two end hosts and intervening switches/routers establish virtual connection
  - switches/routers get involved
- network vs transport layer connection service:
  - network: between two hosts (may also involve intervening switches/routers in case of VCs)
  - transport: between two processes

Network service model

Q: What *service model* for "channel" transporting datagrams from sender to receiver?

<u>Example services for individual datagrams:</u>

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

<u>Example services for a flow of datagrams:</u>

- 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 ? Bandwidth	Loss	Order	Timing	Congestion feedback
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

Network layer connection and connection-less service

- datagram network provides network-layer connectionless service
- VC network provides network-layer connection service
- analogous to the transport-layer services, but:
  - service: host-to-host
  - no choice: network provides one or the other
  - implementation: in network core

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)

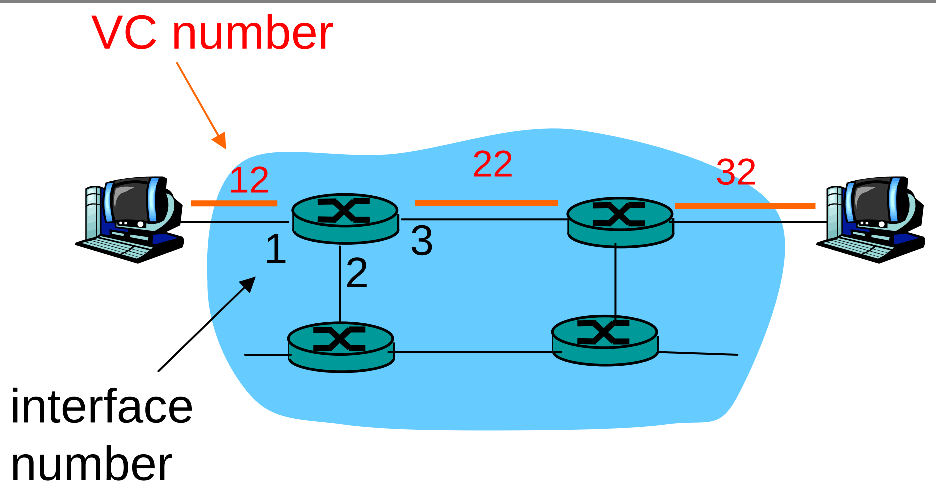
VC implementation

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

Forwarding table



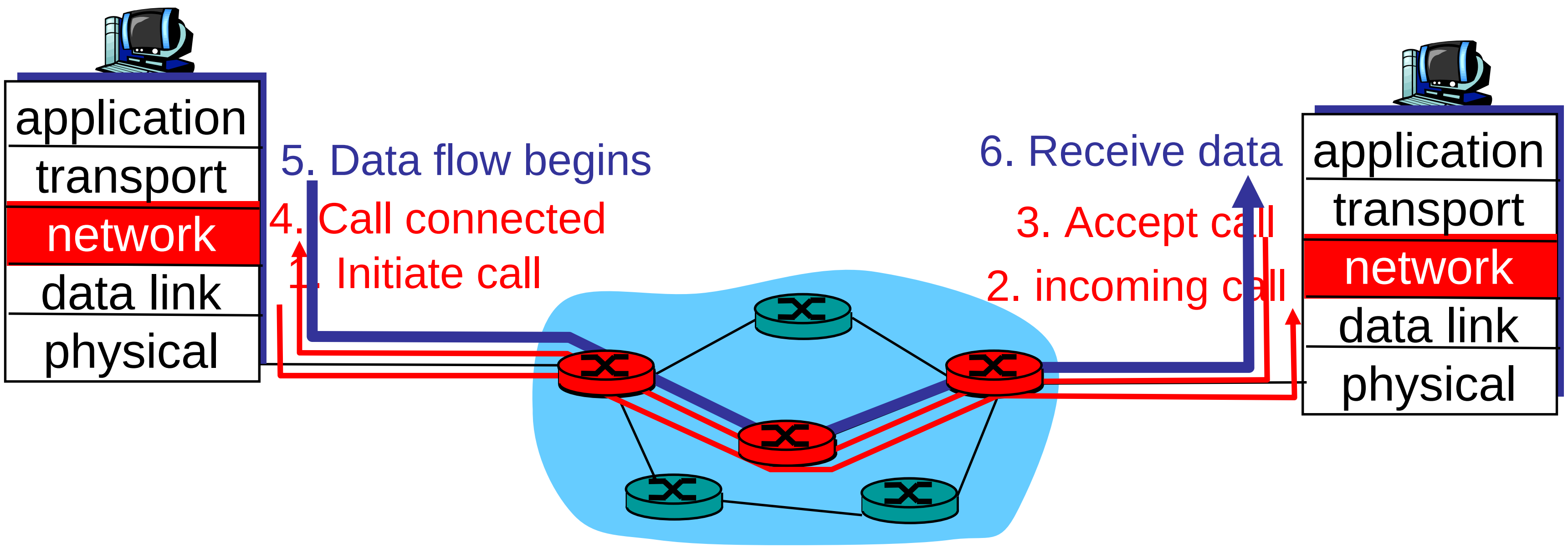
Forwarding table in northwest router:

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

Routers maintain connection state information!

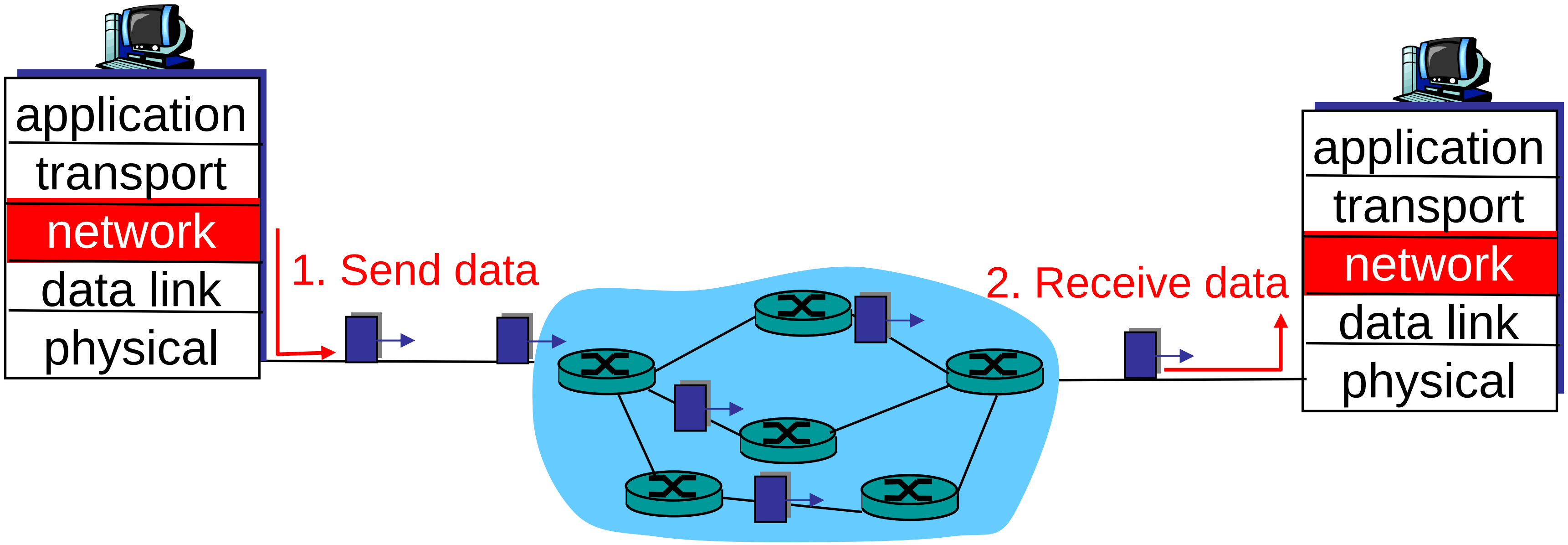
Virtual circuits: signaling protocols

- used to setup, maintain teardown VC
- used in ATM, frame-relay, X.25
- not used in today's Internet



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 or VC network: why?

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”
- 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

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

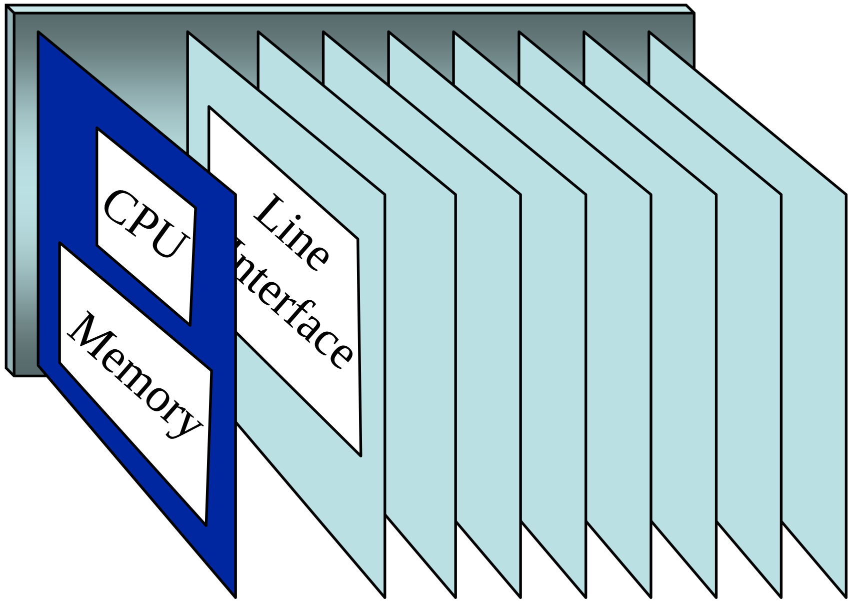
Part 2

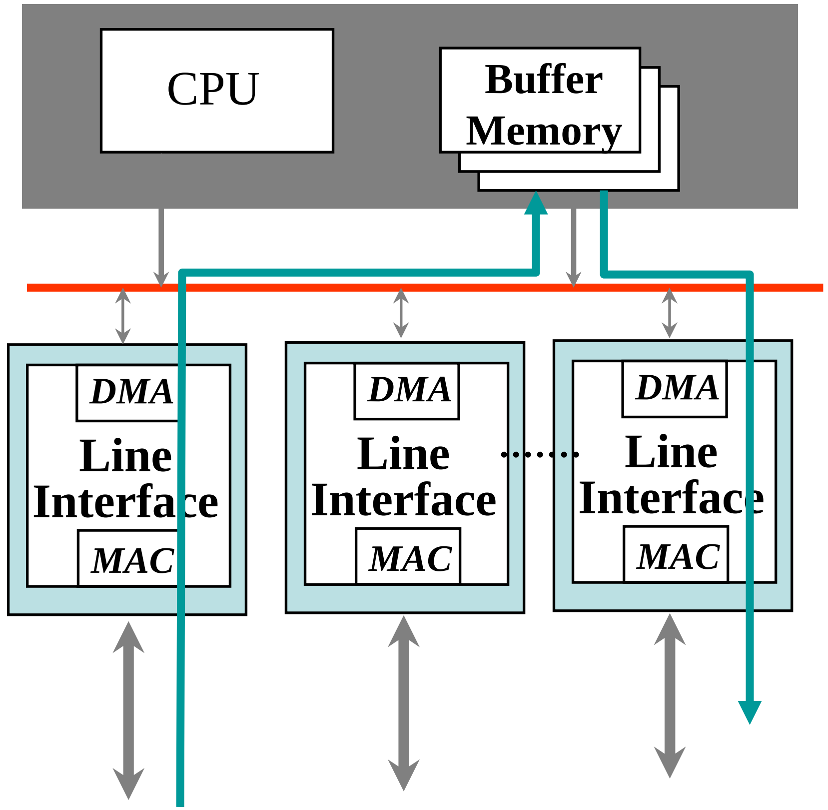
- IPv6
- Virtual circuit and datagram networks
- **What's inside a router**

Part 3

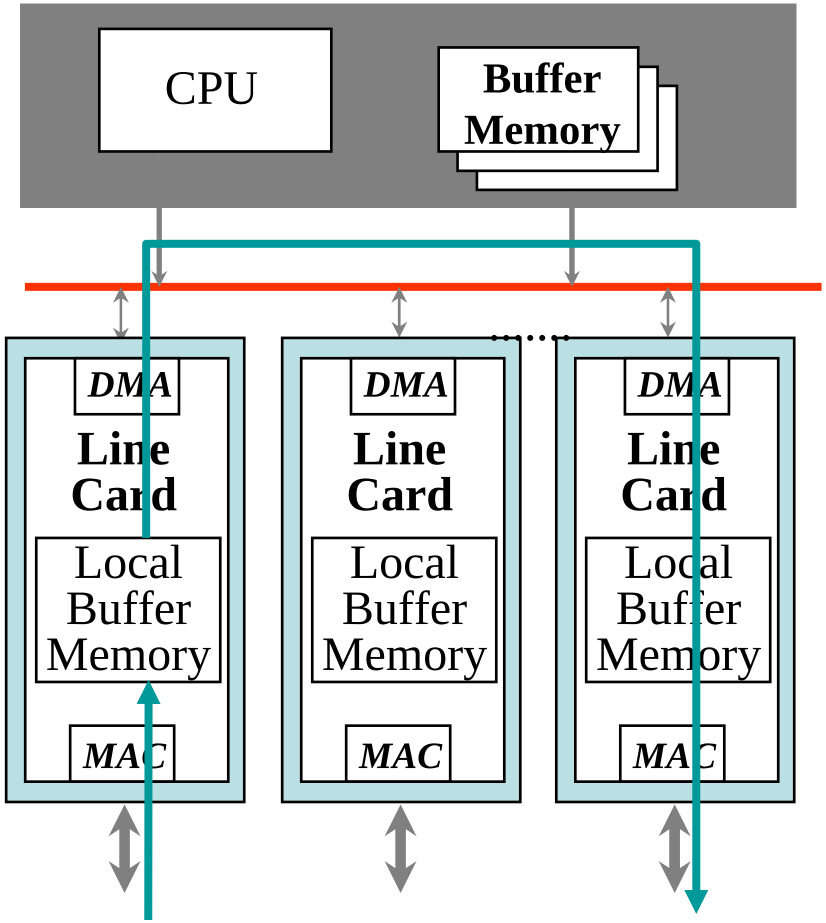
- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
- Broadcast and multicast routing

First-Generation IP Routers

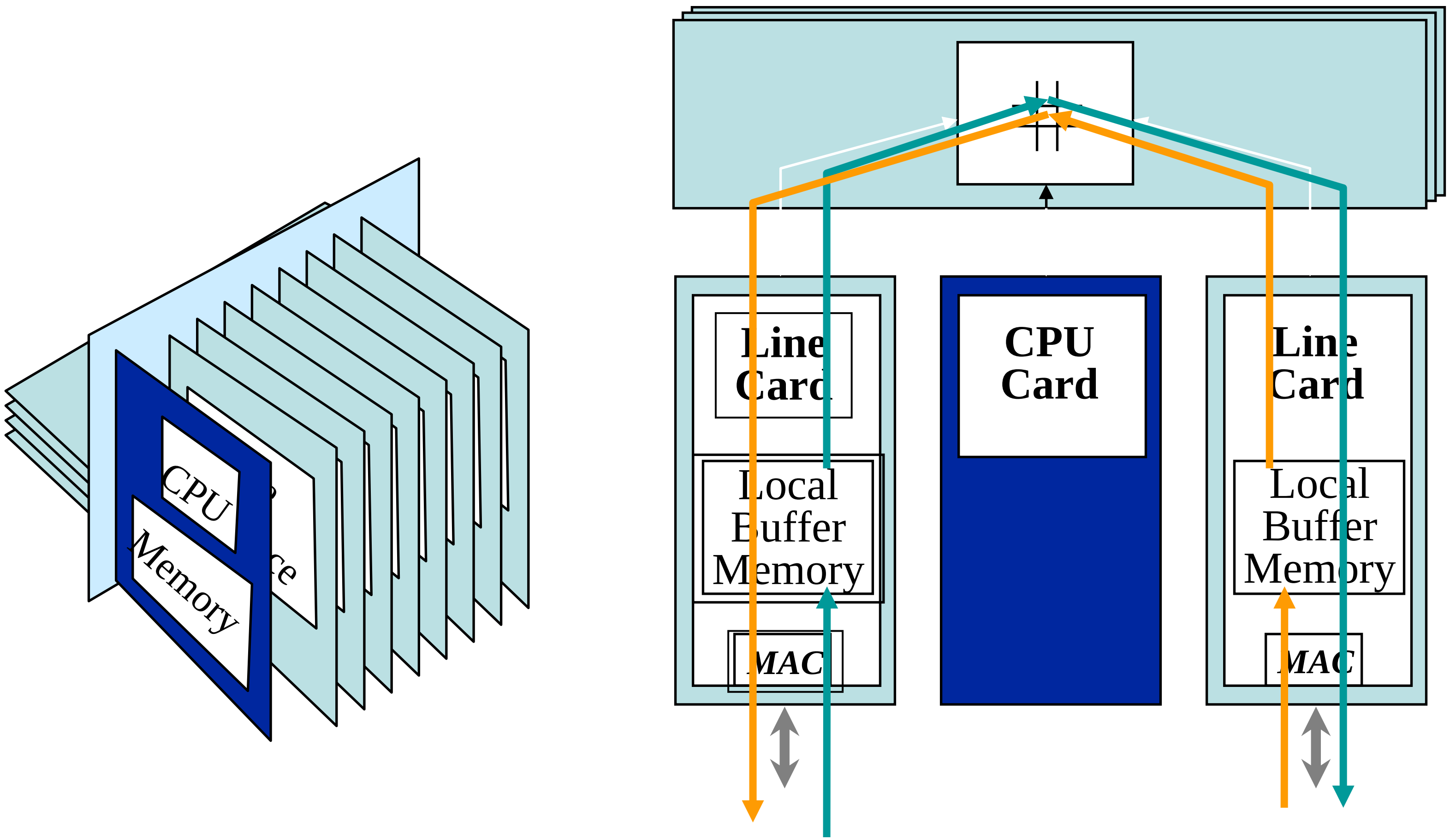




Second-Generation IP Routers

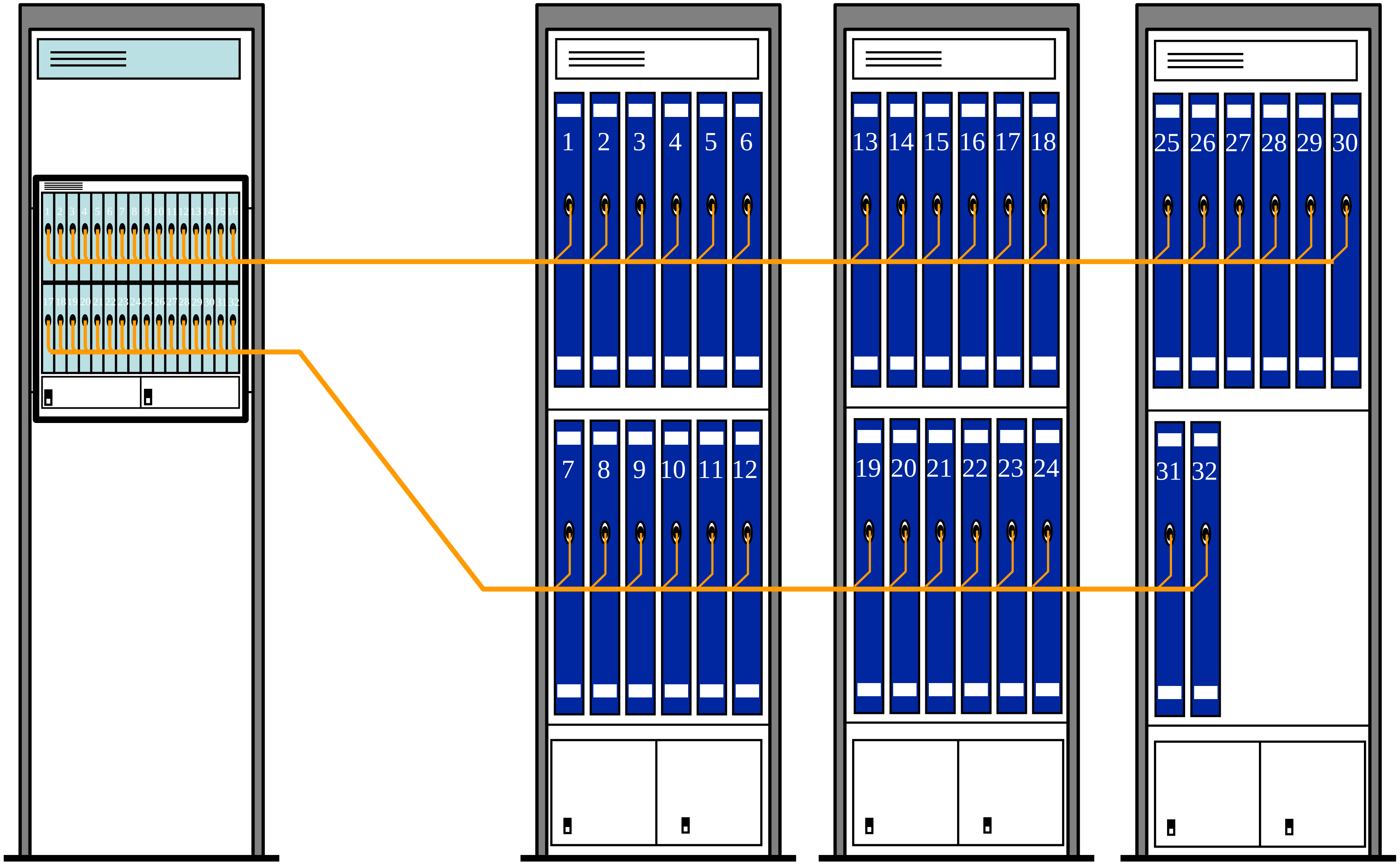


Third-Generation Switches/Routers



Fourth-Generation Switches/Routers

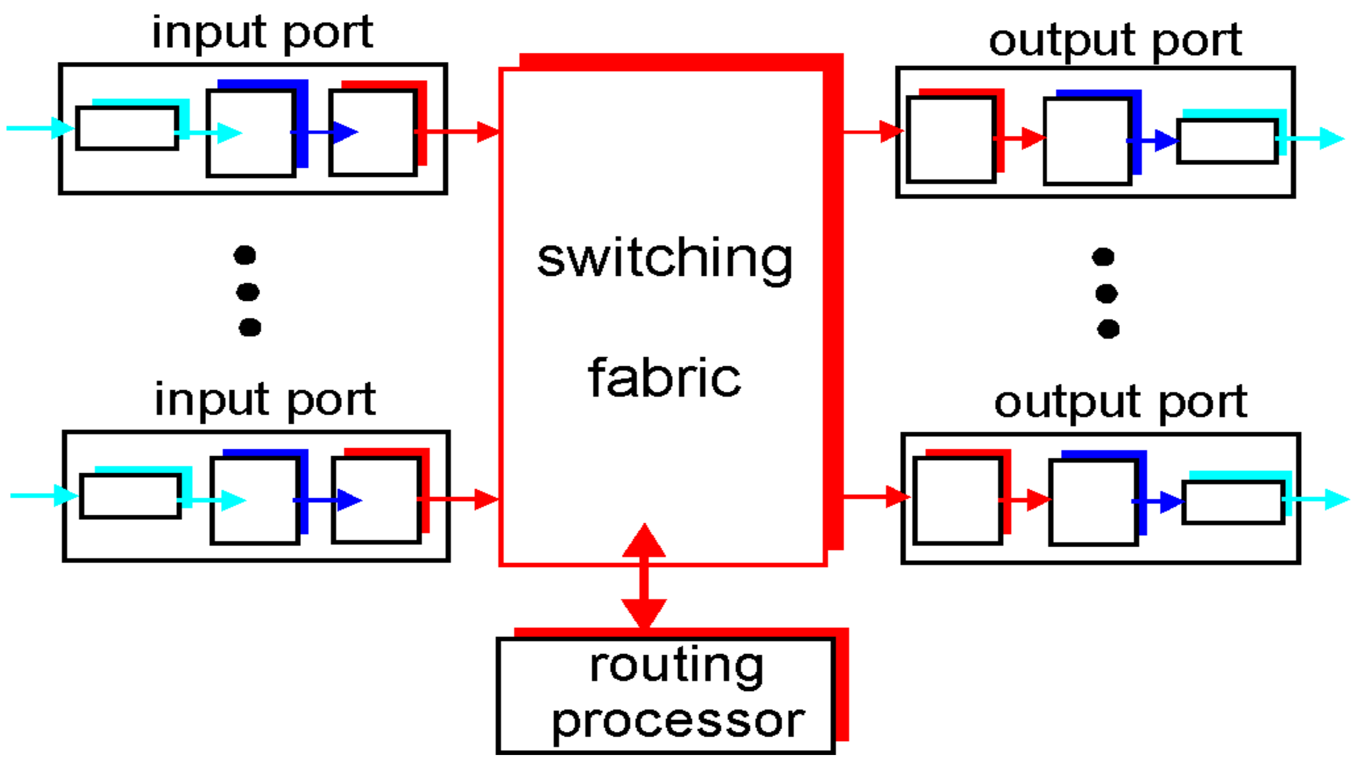
Clustering and Multistage



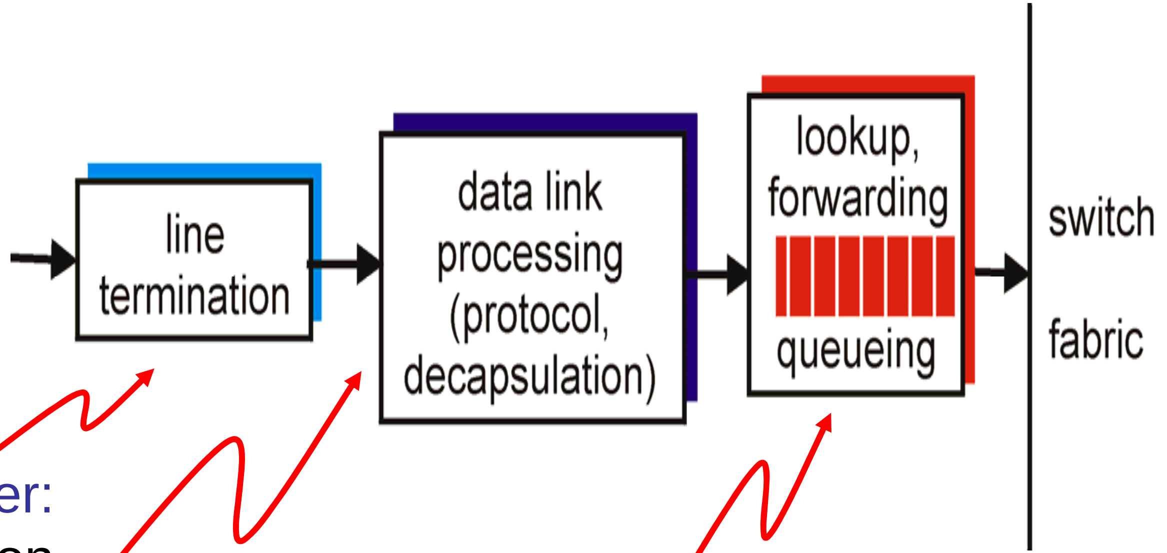
Router Architecture Overview

Two key router functions:

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



Input Port Functions



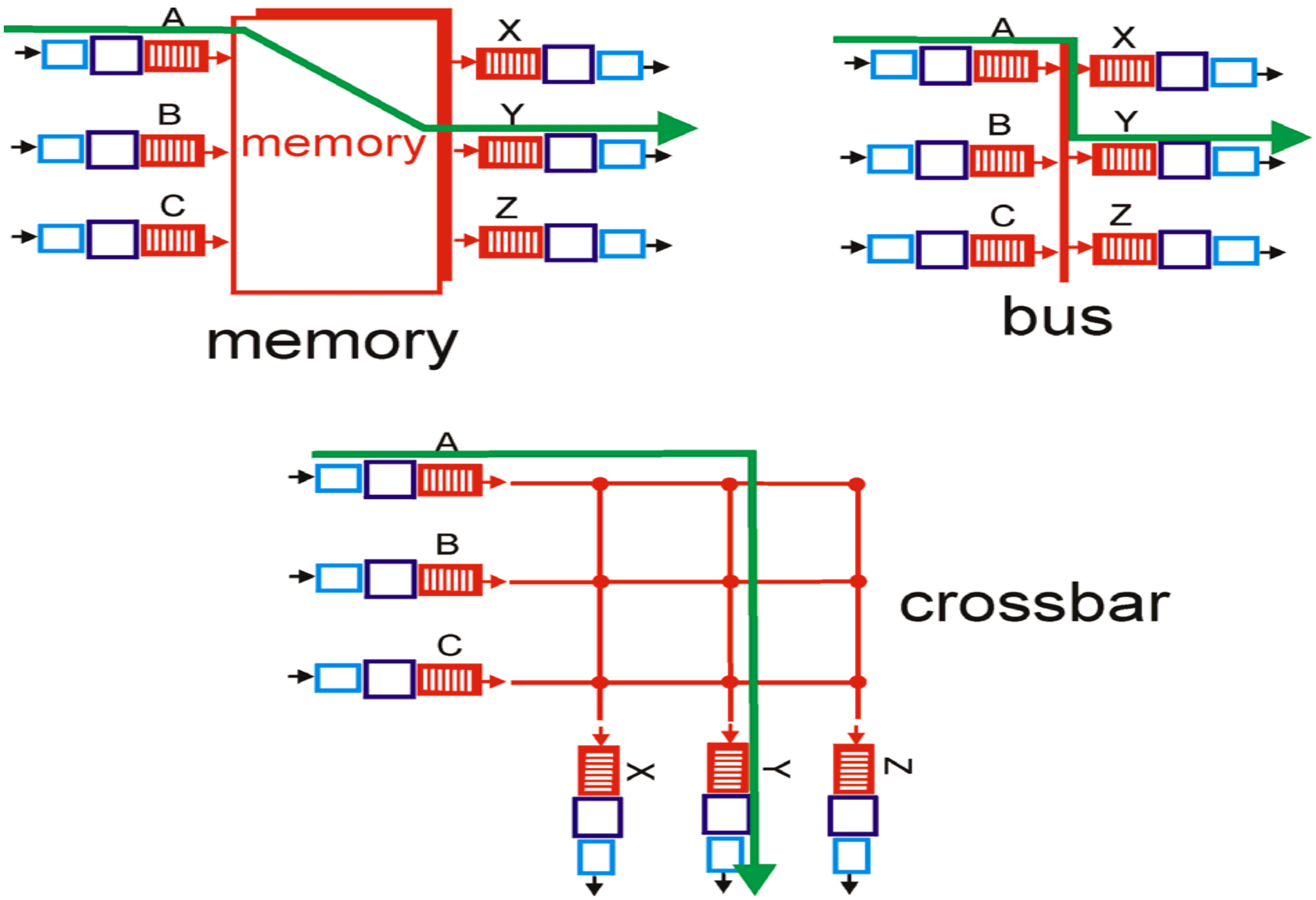
Physical layer:
bit-level reception

Data link layer:
e.g., Ethernet
see chapter 5

Decentralized switching:

- given datagram dest., lookup output port using forwarding table in input port memory
- goal: complete input port processing at 'line speed'
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

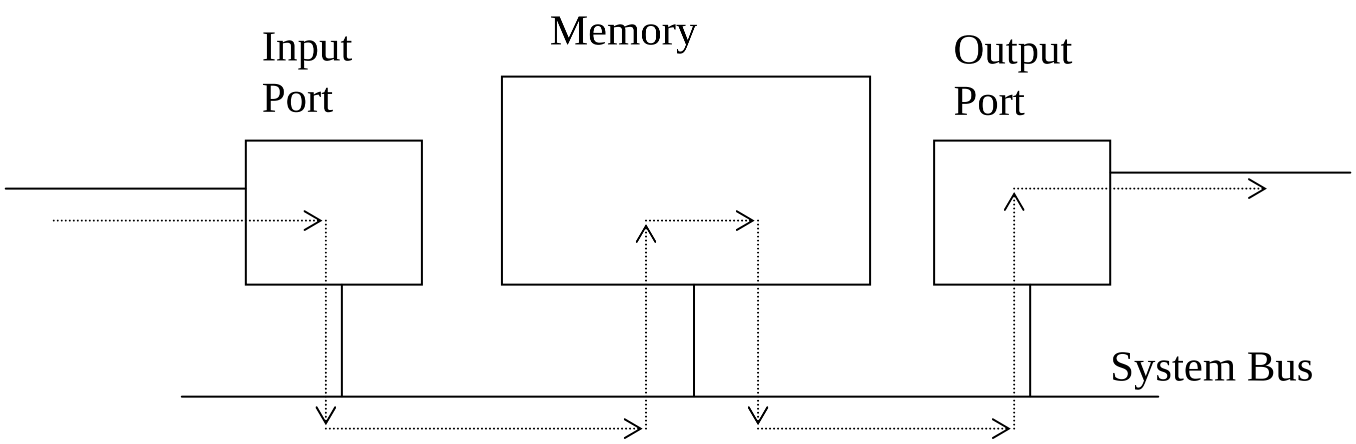
Three types of switching fabrics



Switching Via Memory

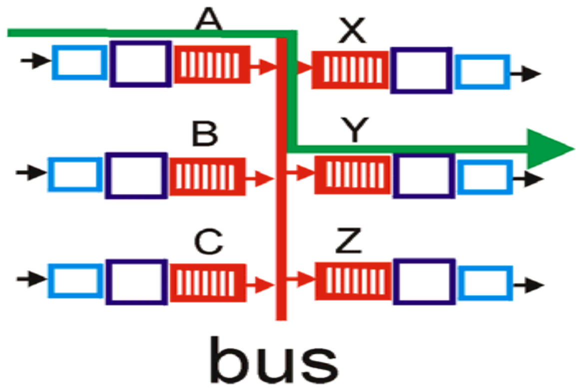
First generation routers:

- traditional computers with switching under direct control of CPU
- packet copied to system's memory
- speed limited by memory bandwidth (2 bus crossings per datagram)



Switching Via a Bus

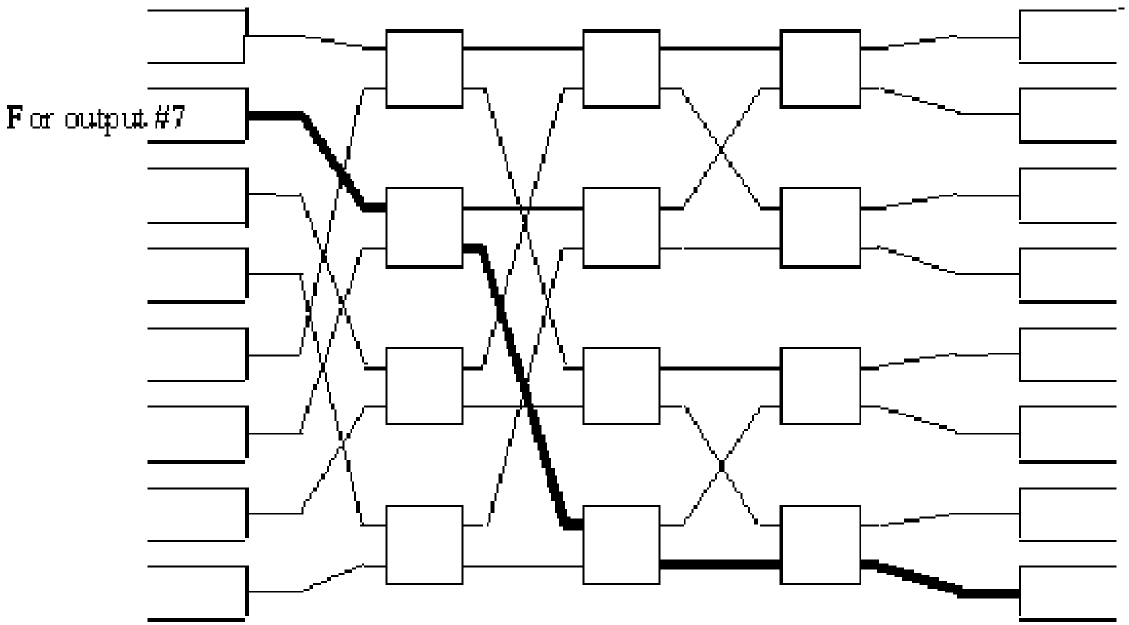
- datagram from input port memory
  to output port memory via a shared bus
- bus contention: switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers



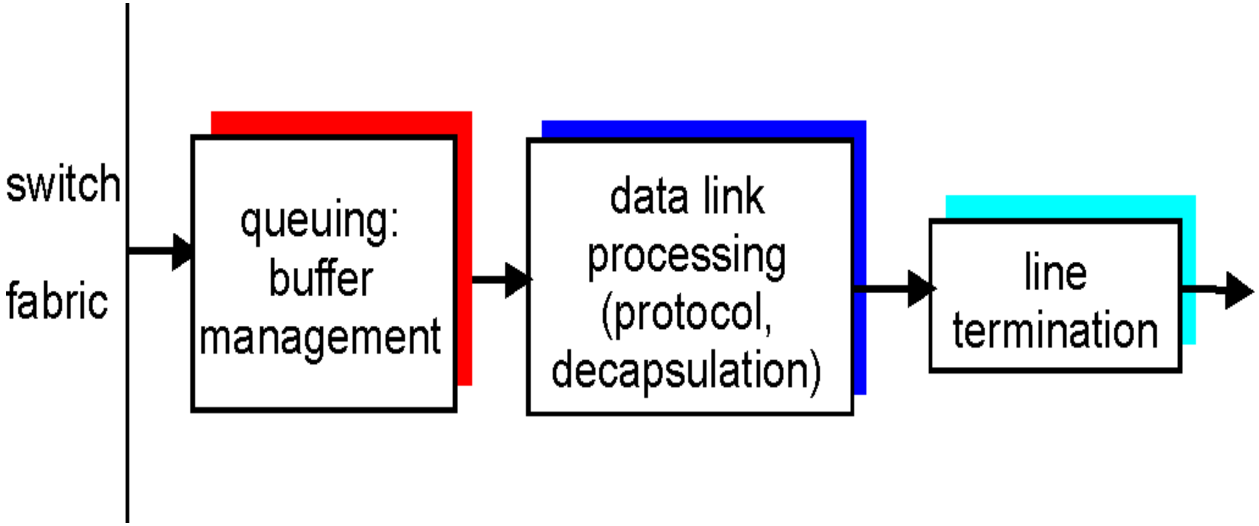
Switching Via An Interconnection Network

- overcome bus bandwidth limitations
- Banyan networks, other interconnection nets initially developed to connect processors in multiprocessor
- advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches 60 Gbps through the interconnection network

Banyan network:

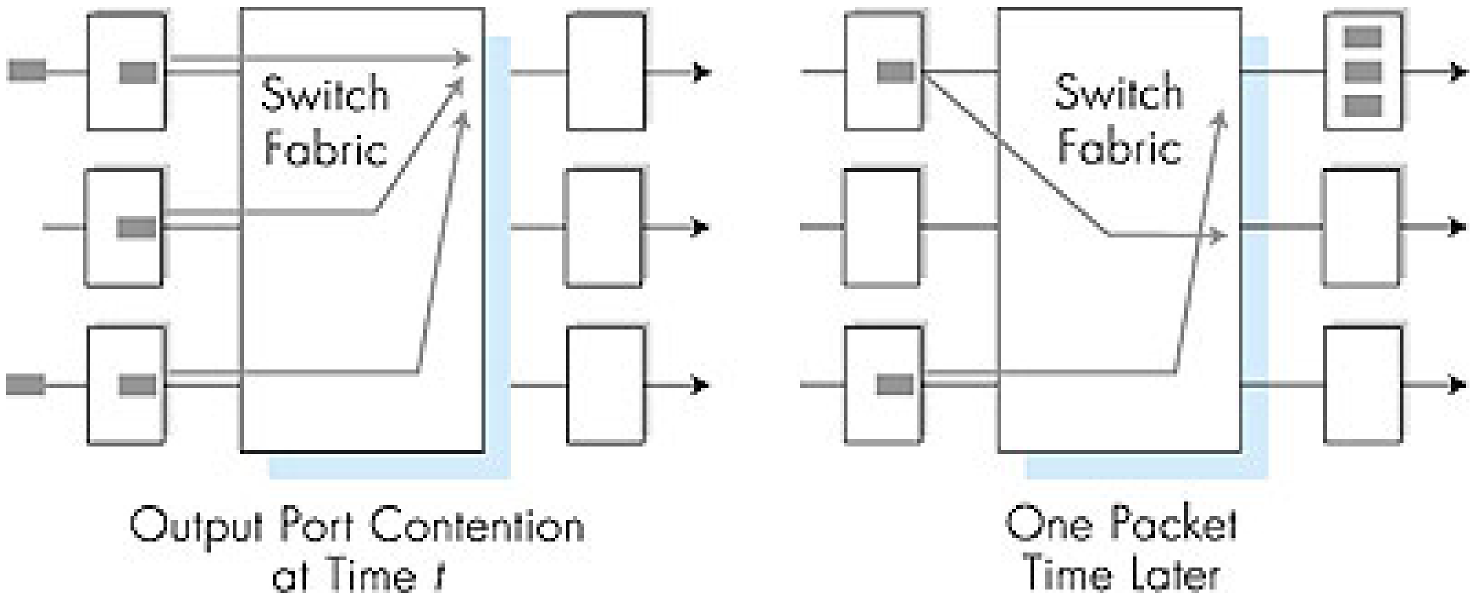


Output Ports



- *Buffering* required when datagrams arrive from fabric faster than the transmission rate
- *Scheduling discipline* chooses among queued datagrams for transmission

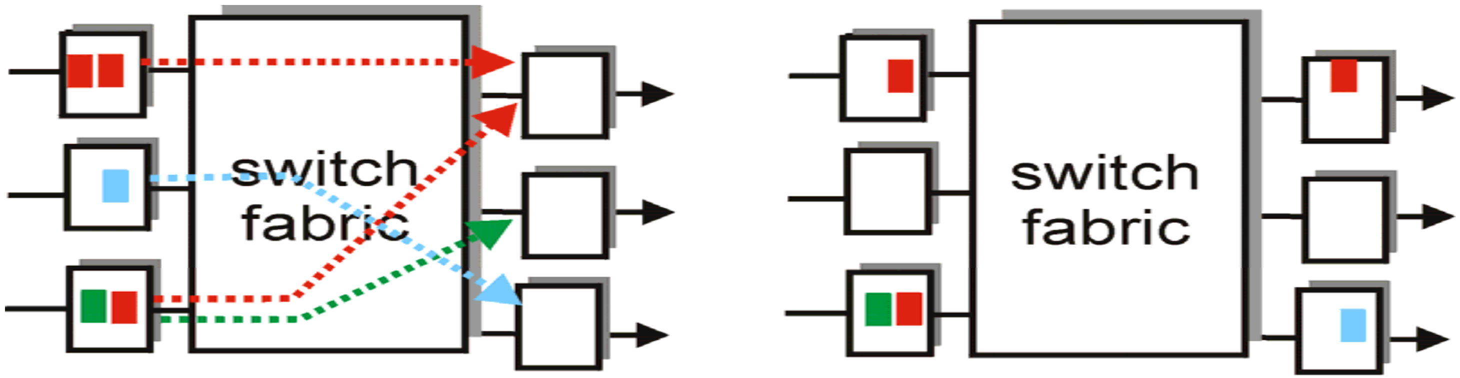
Output port queueing



- buffering when arrival rate via switch exceeds output line speed
- *queueing (delay) and loss due to output port buffer overflow!*

Input Port Queuing

- Fabric slower than input ports combined -> queueing may occur at input queues
- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward
- *queueing delay and loss due to input buffer overflow!*



output port contention at time t - only one red packet can be transferred

green packet experiences HOL blocking

Master Course
Computer Networks
IN2097

Prof. Dr.-Ing. Georg Carle
Christian Grothoff, Ph.D.
Dr. Nils Kammenhuber

Chair for Network Architectures and Services
Institut für Informatik
Technische Universität München
http://www.net.in.tum.de

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

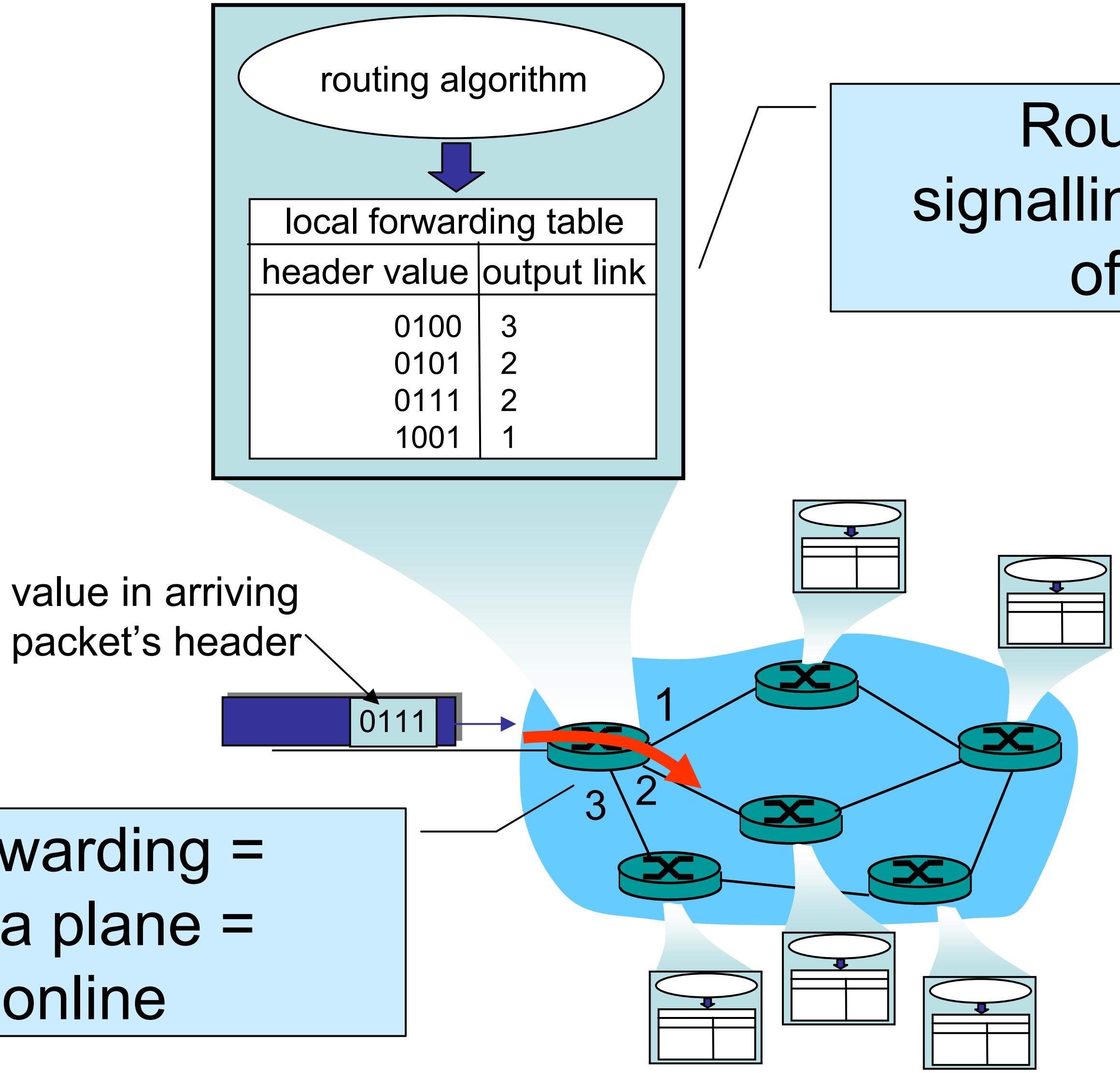
Part 2

- IPv6
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- **Routing algorithms**
  - Link state
  - Distance Vector
  - Path Vector
  - Hierarchical routing
- Internet routing protocols
  - RIP
  - OSPF
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering

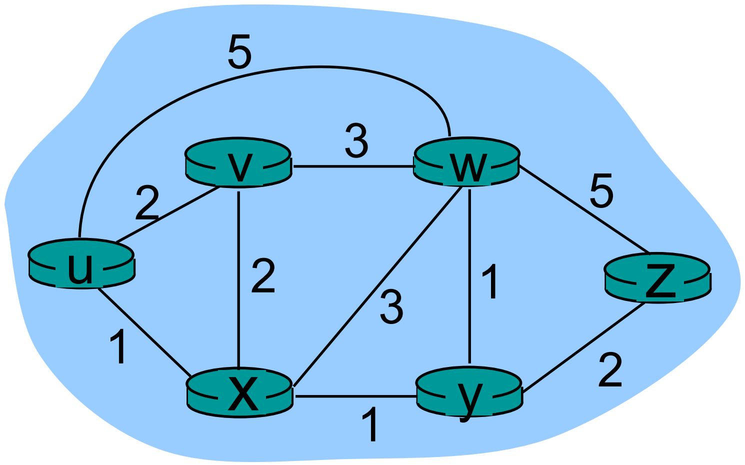
Recall: Interplay between routing and forwarding



Routing =
signalling plane =
offline

Forwarding =
data plane =
online

Graph abstraction: costs



- c(x,x') =: cost of link (x,x')
  e.g.: c(w,z) = 5

- cost could always be 1,
- or inversely related to bandwidth,
- or inversely related to congestion

Cost of path $(x_1, x_2, x_3,\dots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \dots + c(x_{p-1},x_p)$

Question: What's the least-cost path between u and z ?

Routing algorithm: algorithm that finds least-cost path

Routing Algorithm classification

Global or decentralized information?

Global:

- All routers have complete topology and link cost info
- *link state* algorithms (L-S)

Decentralized:

- Router only knows physically-connected neighbors and link costs to neighbors
- Iterative process of computation = exchange of info with neighbors
- *distance vector* algorithms (D-V)
- Variant: *path vector* algorithms

Static or dynamic?

Static:

- Routes change slowly over time

Dynamic:

- Routes change more quickly
  - periodic update
  - in response to link cost changes

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- **Routing algorithms**
  - **Link state**
  - Distance Vector
  - Path Vector
  - Hierarchical routing
- Internet routing protocols
  - RIP
  - OSPF
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering

A Link-State Routing Algorithm

- Net topology and link costs made known to each node
  - Accomplished via *link state broadcasts*
  - All nodes have same info
- Each node independently computes least-cost paths from one node (“source”) to all other nodes
  - Usually done using Dijkstra’s shortest-path algorithm
    - refer to any algorithms & data structures lecture/textbook
    - n nodes in network $\Rightarrow$ $O(n^2)$ or $O(n \log n)$
  - Gives forwarding table for that node
- Result:
  - All nodes have the same information,
  - … thus calculate the same shortest paths,
  - … hence obtain consistent forwarding tables

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- **Routing algorithms**
  - Link state
  - **Distance Vector**
  - Path Vector
  - Hierarchical routing
- Internet routing protocols
  - RIP
  - OSPF
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering

Distance Vector Algorithm

- No node knows entire topology
- Nodes only communicate with neighbours (i.e., no broadcasts)
- Nodes *jointly* calculate shortest paths
  - Iterative process
  - Algorithm == protocol
- Distributed application of Bellman-Ford algorithm
  - refer to any algorithms&data structures lecture/textbook

Distance Vector Algorithm

Bellman-Ford Equation (dynamic programming)

Let

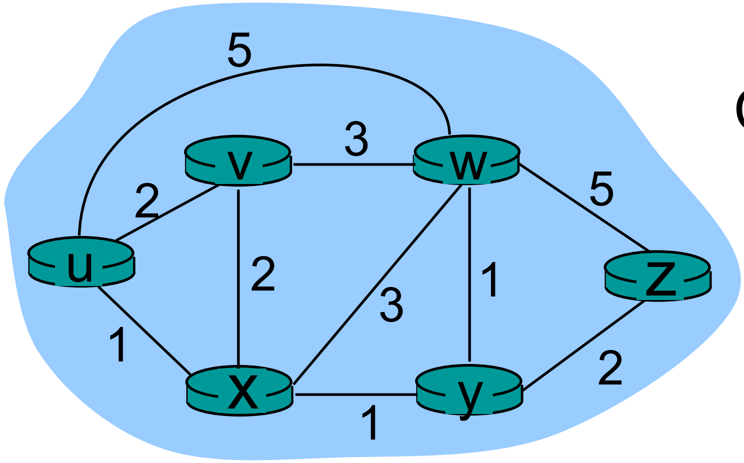
- $c(x,y)$:= cost of edge from x to y
- $d_x(y)$:= cost of least-cost path from x to y

Then

$d_x(y) = \min \{c(x,v) + d_v(y) \}$

where min is taken over all neighbours v of x

Bellman-Ford example



Clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$\begin{aligned} d_u(z) &= \min \{ c(u,v) + d_v(z), \\ &\qquad c(u,x) + d_x(z), \\ &\qquad c(u,w) + d_w(z) \} \\ &= \min \{2 + 5, \\ &\qquad 1 + 3, \\ &\qquad 5 + 3\} = 4 \end{aligned}$$

Node that achieves minimum is next hop in shortest path
→ forwarding table

Distance Vector Algorithm

- Define $D_x(y)$:= estimate of least cost from *x* to *y*
- Node *x* knows cost to each neighbour *v*: $c(x,v)$
- Node *x* maintains distance vector $\mathbf{D}_x = [\ D_x(y): y \in N\ ]$
  (N := set of nodes)
- Node *x* also maintains its neighbours' distance vectors:
  - For each neighbour *v*,
    x maintains $\mathbf{D}_v = [\ D_v(y): y \in N\ ]$

Distance vector algorithm (4)

Basic idea:

- From time-to-time, each node sends its own distance vector estimate D to neighbors
  - Asynchronously
- When a node x receives new DV estimate from neighbour, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

- Under minor, natural conditions, these estimates $D_x(y)$ *converge to the actual least cost* $d_x(y)$

Distance Vector Algorithm (5)

Iterative, asynchronous:

each local iteration caused by:

- local link cost change
- DV update message from neighbour

Distributed:

- Each node notifies neighbors *only* when its DV changes
  - neighbours then notify their neighbours if this caused *their* DV to change
  - etc.

Each node:

Forever:

wait for (change in local link cost *or* message arriving from neighbour}

recompute estimates

if (DV to any destination has changed) { *notify* neighbours }

Distance Vector Algorithm (6)

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

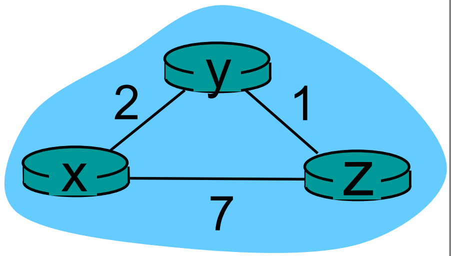
$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0



time

$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$
$= \min\{2+0\ ,\ 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\}$
$= \min\{2+1\ ,\ 7+0\} = 3$

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

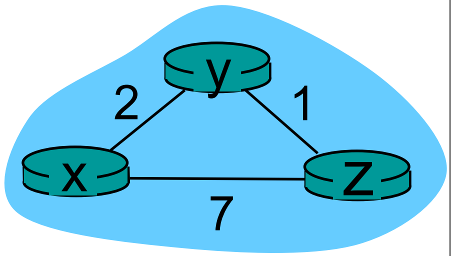
node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

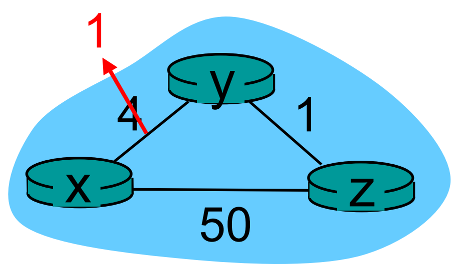
time



Distance Vector: link cost changes (1)

Link cost changes:

- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notify neighbors



“good news travels fast”

At time t_0, *y* detects the link-cost change, updates its DV, and informs its neighbors.

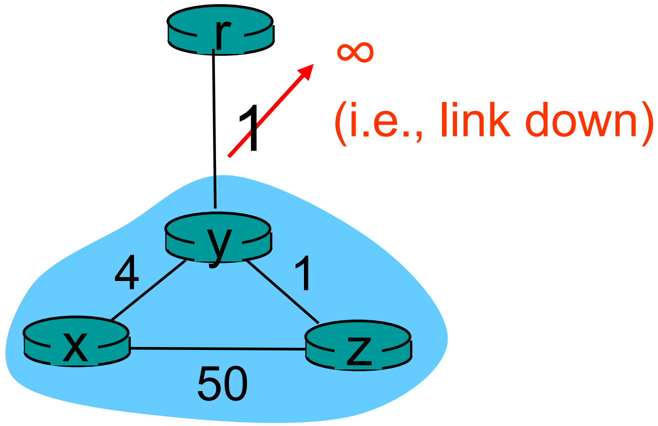
At time t_1, *z* receives the update from *y* and updates its table. It computes a new least cost to *x* and sends its neighbors its DV.

At time t_2, *y* receives *z*’s update and updates its distance table. *y*’s least costs do not change and hence *y* does *not* send any message to *z*.

Distance Vector: link cost changes (2)

- But: bad news travels slow — "count to infinity" problem!
- In example: Many iterations before algorithm stabilizes!

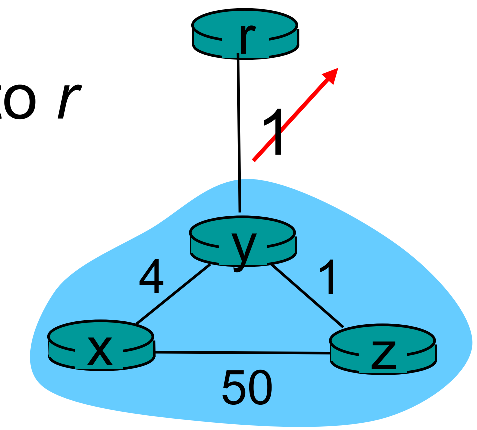
1. Cost increase for $y \rightarrow r$:
   - *y* consults DV,
   - *y* selects "cheaper" route via *z* (cost 2+1 = 3),
   - Sends update to *z* and *x* (cost to *r* now 3 instead of 1)



2. *z* detects cost increase for path to *r*:
   - was 1+1, is now 3+1
   - Sends update to *y* and *x* (cost to *r* now 4 instead of 2)
3. *y* detects cost increase, sends update to *z*
4. *z* detects cost increase, sends update to *y*
5. ….

Distance Vector: Solutions that only half work

- Finite infinity: Define some number to be ∞ (in RIP: 16 := ∞)
- Split Horizon:
  - Tell to a neighbour that is part of a best path to a destination that the destination cannot be reached
  - If *z* routes through *y* to get to *r*
    z tells *y* that its own (i.e., *y*'s) distance to *r* is infinite (so *y* won't route to *r* via *z*)



- Poisoned Reverse:
  - In addition, *actively* advertise a route as unreachable to the neighbour from which the route was learned
- (Warning: Terms often used interchangeably!)
- Often help, but cannot solve all problem instances
- Can significantly increase number of routing messages

Comparison of LS and DV algorithms

Message complexity

- LS: with *n* nodes, *E* links, O(*nE*) msgs sent
- DV: exchange between neighbors only
  - convergence time varies

Speed of Convergence

- LS: $O(n^2)$ algorithm requires O(nE) msgs
  - may have oscillations
- DV: convergence time varies
  - may be routing loops
  - count-to-infinity problem

Robustness: what happens if router malfunctions?

LS:

- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagate thru network

Part 1
- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2
- IPv6
- Virtual circuit and datagram networks
- What's inside a router

Part 3
- **Routing algorithms**
  - Link state
  - Distance Vector
  - **Path Vector**
  - Hierarchical routing
- Internet routing protocols
  - RIP
  - OSPF
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering

Path Vector protocols

- Problem with D-V protocol:
  Path cost is "anonymous" single number
- Path Vector protocol:
  - For each destination, advertise entire path (=sequence of node identifiers) to neighbours
  - Cost calculation can be done by looking at path
  - Easy loop detection: Does my node ID already appear in the path?
- Not used very often
  - only in BGP …
  - … and BGP is much more complex than just paths!

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- **Routing algorithms**
  - Link state
  - Distance Vector
  - Path Vector
  - **Hierarchical routing**
- Internet routing protocols
  - RIP
  - OSPF
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering

Hierarchical Routing

Our routing study thus far = idealisation

- all routers identical
- network "flat"

… *not* true in practice!

Scale = billions of destinations:

- Can't store all destinations in routing tables!
- Routing table exchange would swamp links!

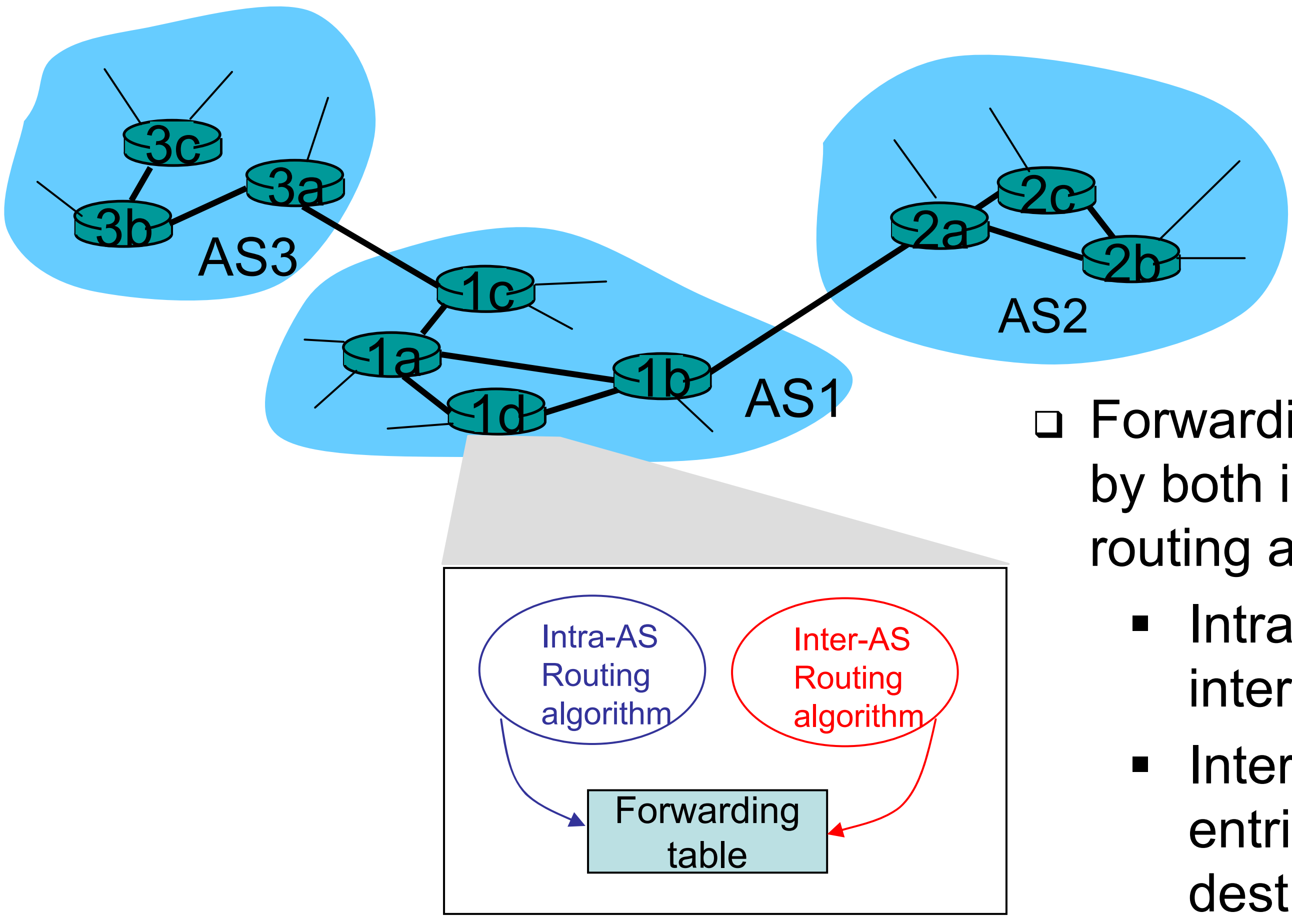
Administrative autonomy

- Internet = network of networks
- Each network admin may want to control routing in its own network — no central administration!

Hierarchical Routing

- Aggregate routers into regions called
  “autonomous systems” (short: AS; plural: ASes)
- Routers in same AS run same routing protocol
  - = “intra-AS” routing protocol (also called “intradomain”)
  - Routers in different ASes can run different intra-AS routing protocols
- ASes are connected: via gateway routers
  - Direct link to [gateway] router in another AS
    = “inter-AS” routing protocol (also called “interdomain”)
  - Warning: Non-gateway routers need to know about inter-AS routing as well!

Interconnected ASes



- Forwarding table configured by both intra- *and* inter-AS routing algorithm:
  - Intra-AS sets entries for internal destinations
  - Inter-AS *and* intra-AS set entries for external destinations

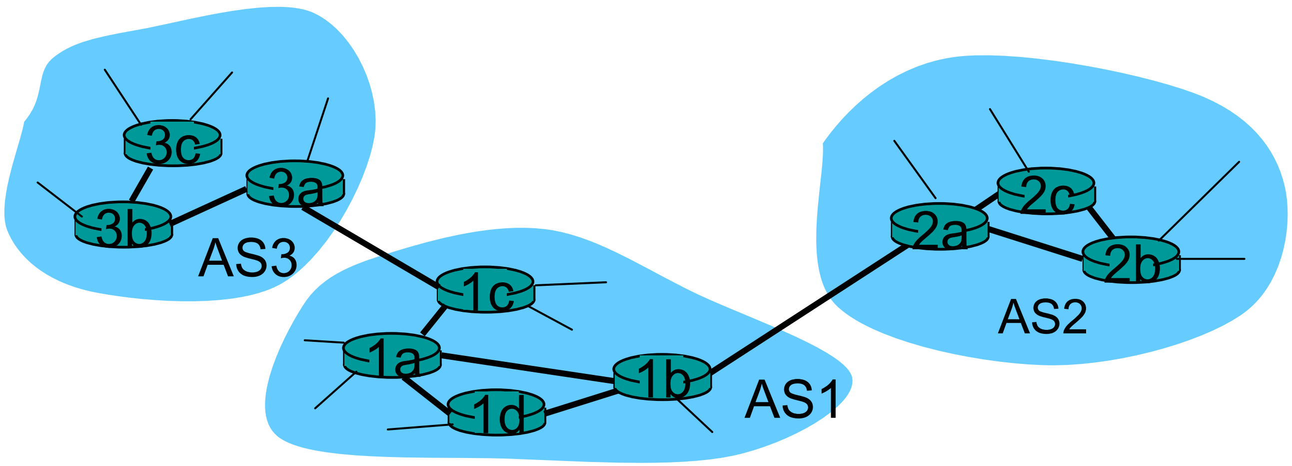
Inter-AS tasks

- Suppose router in AS1 receives datagram destined outside of AS1:
  - router should forward packet to gateway router, but which one?

AS1 must:

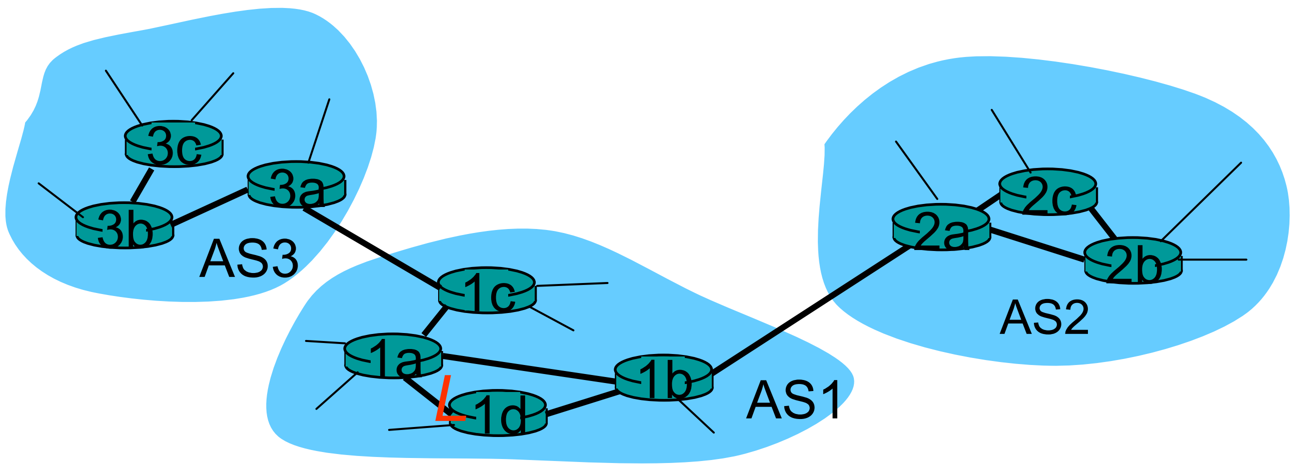
1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info *to all* routers in AS1 (i.e., not just the gateway routers)

Job of inter-AS routing!



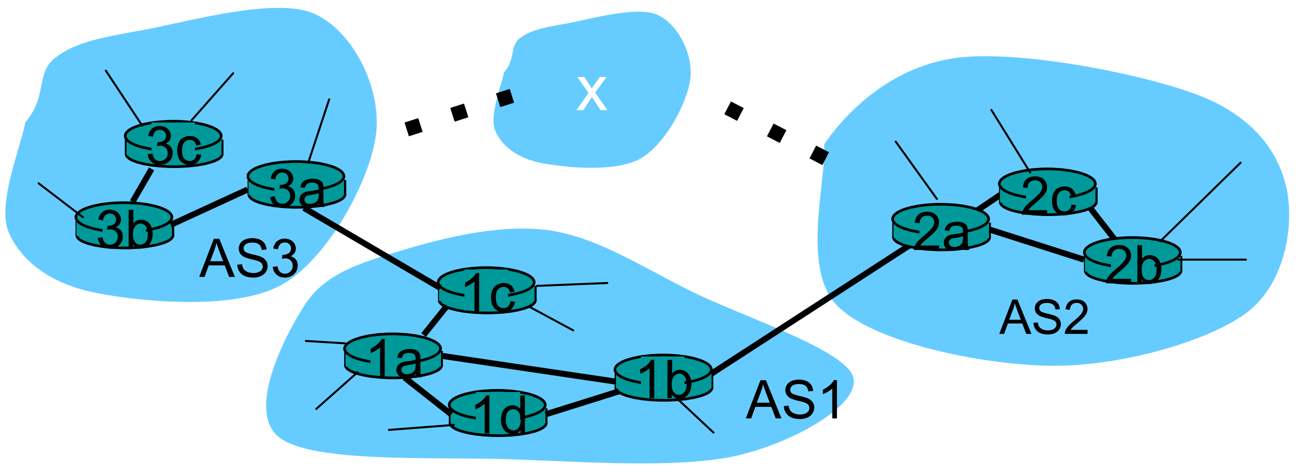
Example: Setting forwarding table in router 1d

- Suppose AS1 learns (via inter-AS protocol) that subnet *x* is reachable via AS3 (gateway 1c) but not via AS2.
- Inter-AS protocol propagates reachability info to all internal routers.
- Router 1d determines from intra-AS routing info that its interface *I* (i.e., interface to 1a) is on the least cost path to 1c.
  - installs forwarding table entry *(x,I)*



Example: Choosing among multiple ASes

- Now suppose AS1 learns from inter-AS protocol that subnet x is reachable from AS3 *and* from AS2.
- To configure forwarding table, router 1d must determine towards which gateway it should forward packets for destination x.
  - This is also job of inter-AS routing protocol!



Interplay of inter-AS and intra-AS routing

- Inter-AS routing
  - Only for destinations outside of own AS
  - Used to determine gateway router
  - Also: Steers transit traffic
    (from AS *x* to AS *y* via our own AS)
- Intra-AS routing
  - Used for destinations within own AS
  - Used to reach gateway router for outside destinations

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Path Vector
  - Hierarchical routing
- **Internet routing protocols**
  - (RIP)
  - OSPF
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering

Intra-AS Routing

- Also known as Interior Gateway Protocols (IGP)
- Most common Intra-AS routing protocols:
  - RIP: Routing Information Protocol — DV (typically small systems)
  - OSPF: Open Shortest Path First — hierarchical LS (typically medium to large systems)
  - IS-IS: Intermediate System to Intermediate System — hierarchical LS (typically medium-sized ASes)
  - (E)IGRP: (Enhanced) Interior Gateway Routing Protocol (Cisco proprietary) — hybrid of LS and DV

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Path Vector
  - Hierarchical routing
- **Internet routing protocols**
  - (RIP)
  - **OSPF**
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering

OSPF (Open Shortest Path First)

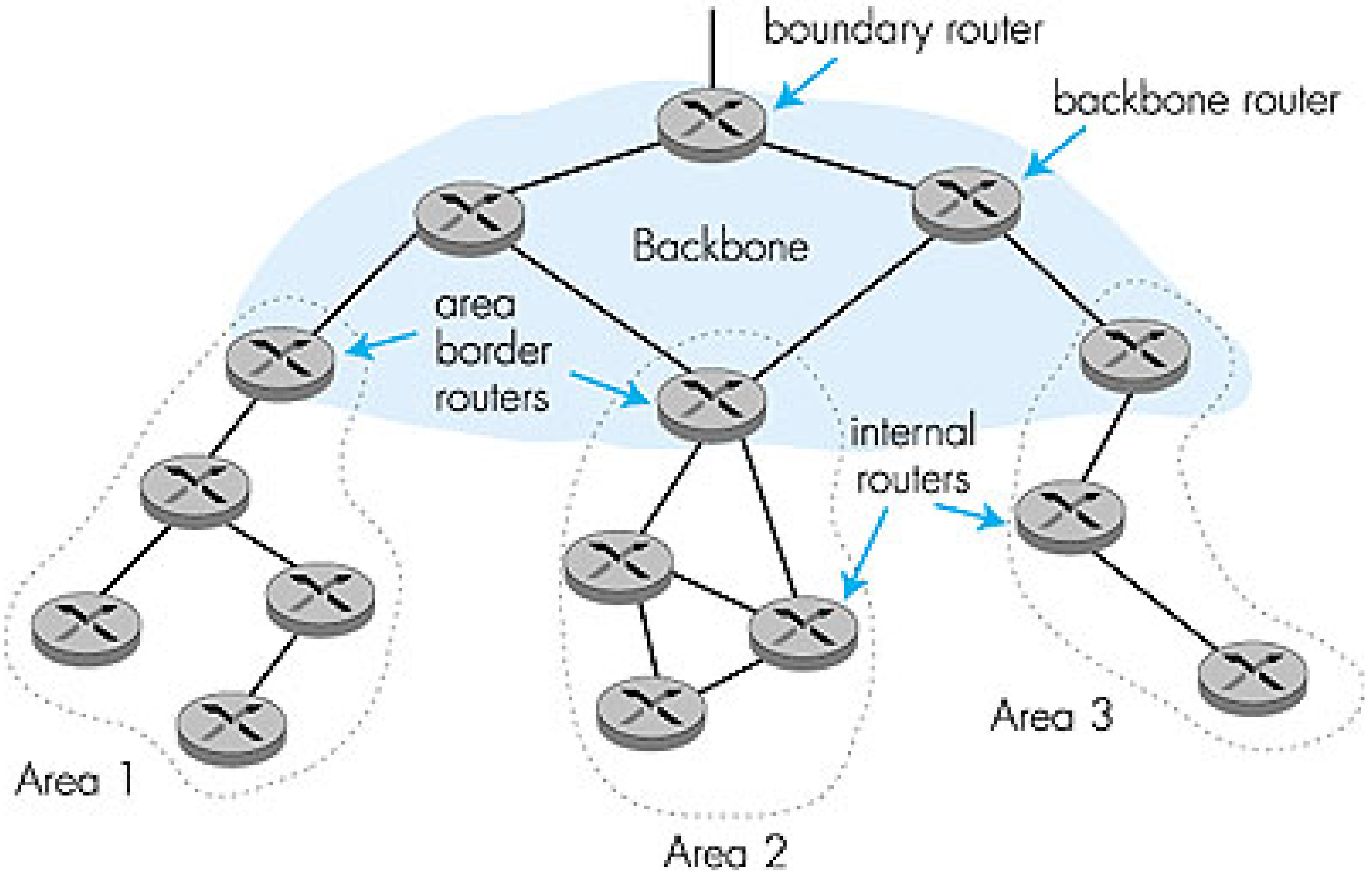
- “open”: publicly available
- uses Link State algorithm
  - LS packet dissemination
  - topology map at each node
  - route computation using Dijkstra’s algorithm

- OSPF advertisement carries one entry per neighbor router
- advertisements disseminated to entire AS (via flooding)
  - carried in OSPF messages directly over IP (rather than TCP or UDP

OSPF “advanced” features (not in RIP)

- security: all OSPF messages authenticated (to prevent malicious intrusion)
- multiple same-cost paths allowed (only one path in RIP)
- For each link, multiple cost metrics for different TOS (e.g., satellite link cost set “low” for best effort; high for real time)
- integrated uni- and multicast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- hierarchical OSPF in large domains.

Hierarchical OSPF



Hierarchical OSPF

- OSPF *can* create a two-level hierarchy similar to inter-AS and intra-AS routing within an AS
  - Two levels: local *areas* and the *backbone*
  - Link-state advertisements only within local area
  - Each node has detailed area topology; but only knows direction (shortest path) to networks in other areas
- *Area border routers:* "summarize" distances to networks in own area; advertise distances to other Area Border routers
- *Backbone routers:* run OSPF routing limited to backbone
- *Boundary routers:* connect to other ASes

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Path Vector
  - Hierarchical routing
- **Internet routing protocols**
  - RIP
  - OSPF
  - **BGP**
- Business considerations
  - Policy routing
  - Traffic engineering

Internet inter-AS routing: BGP

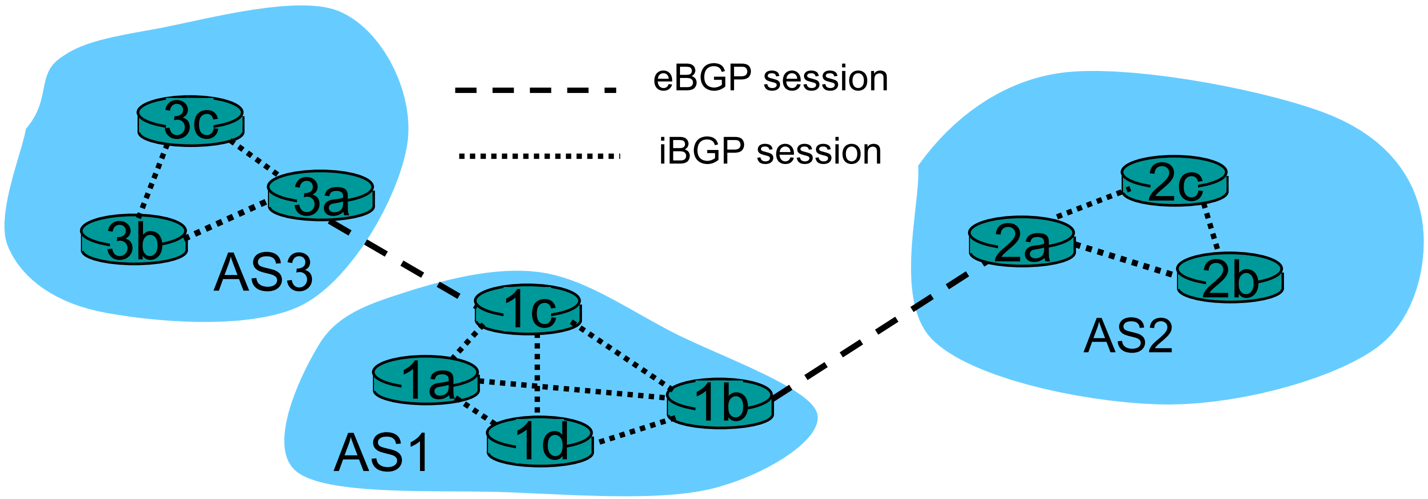
- BGP (Border Gateway Protocol):
  The de facto standard for inter-AS routing
- BGP provides each AS a means to:
  1. Obtain subnet reachability information from neighbouring ASes.
  2. Propagate reachability information to all AS-internal routers.
  3. Determine "good" routes to subnets based on reachability information and policy.
- Allows an AS to advertise the existence of an IP prefix to rest of Internet: *"This subnet is here"*

BGP basics

- Pairs of routers (BGP peers) exchange routing info over semi-permanent TCP connections: BGP sessions
  - BGP sessions need not correspond to physical links!
- When AS2 advertises an IP prefix to AS1:
  - AS2 *promises* it will forward IP packets towards that prefix
  - AS2 can aggregate prefixes in its advertisement

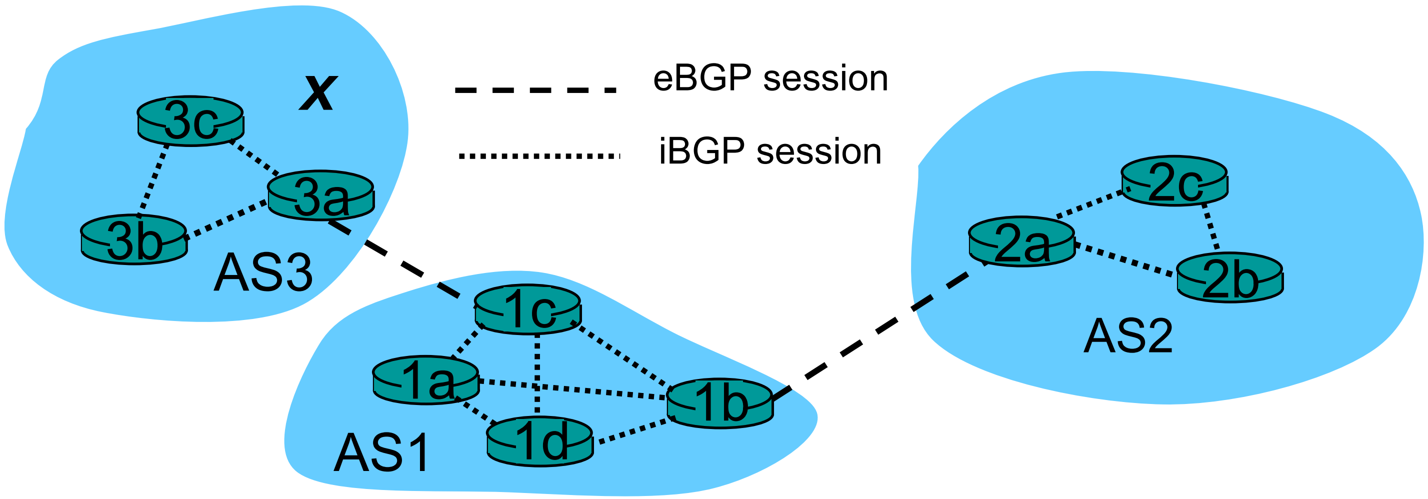
eBGP and iBGP

- External BGP: between routers in *different* ASes
- Internal BGP: between routers in *same* AS
  - Remember: In spite of intra-AS routing protocol, *all* routers need to know about external destinations (not only border routers)
- No different protocols — just slightly different configurations!



Distributing reachability info

- Using eBGP session between 3a and 1c, AS3 sends reachability info about prefix *x* to AS1.
  - 1c can then use iBGP to distribute new prefix info to all routers in AS1
  - 1b can then re-advertise new reachability info to AS2 over 1b-to-2a eBGP session
- When router learns of new prefix *x*, it creates entry for prefix in its forwarding table.



Slides subject to change after this point until Monday!

Path attributes & BGP routes

- advertised prefix includes BGP attributes.
  - prefix + attributes = "route"
- two important attributes:
  - AS-PATH: contains ASs through which prefix advertisement has passed: e.g, AS 67, AS 17
  - NEXT-HOP: indicates specific internal-AS router to next-hop AS. (may be multiple links from current AS to next-hop-AS)
- when gateway router receives route advertisement, uses import policy to accept/decline.

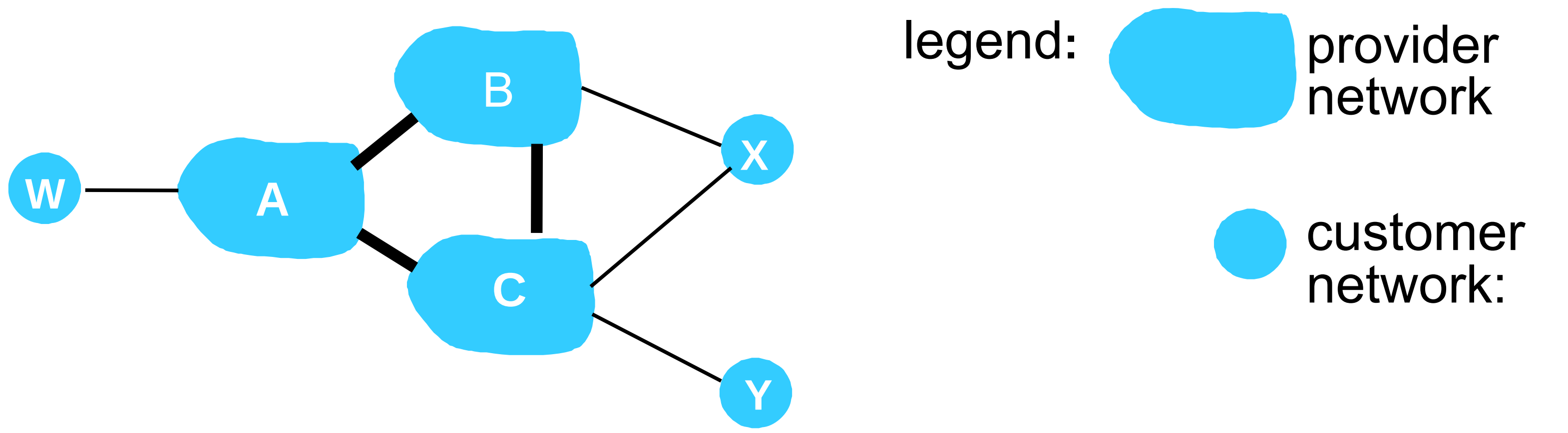
BGP route selection

- router may learn about more than 1 route to some prefix. Router must select route.
- elimination rules:
  1. local preference value attribute: policy decision
  2. shortest AS-PATH
  3. closest NEXT-HOP router: hot potato routing
  4. additional criteria

BGP messages

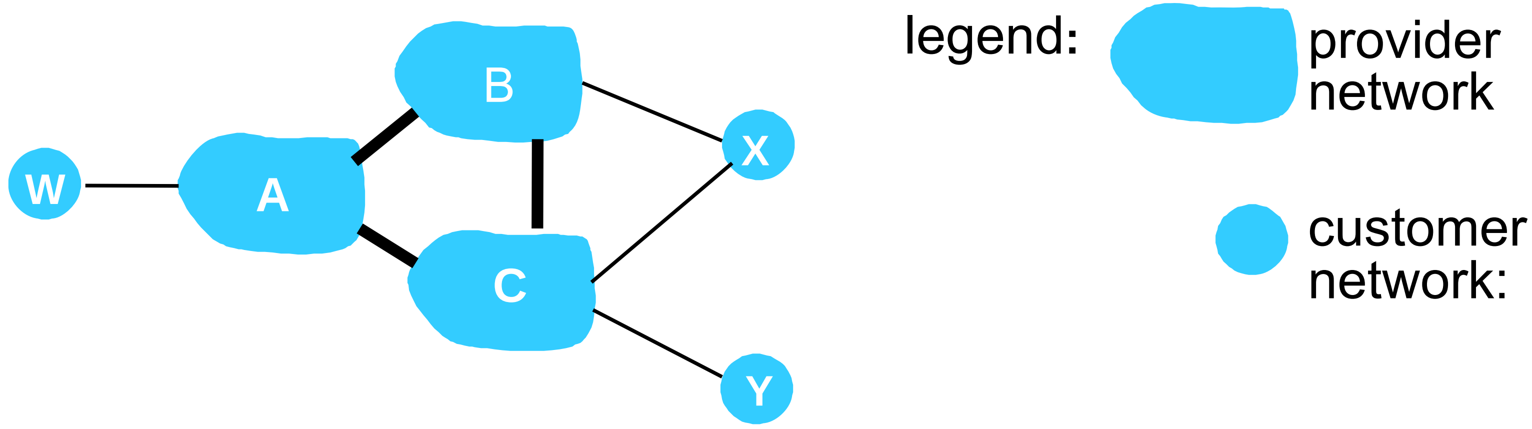
- BGP messages exchanged using TCP.
- BGP messages:
  - OPEN: opens TCP connection to peer and authenticates sender
  - UPDATE: advertises new path (or withdraws old)
  - KEEPALIVE keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION: reports errors in previous msg; also used to close connection

BGP routing policy



- A,B,C are provider networks
- X,W,Y are customer (of provider networks)
- X is dual-homed: attached to two networks
  - X does not want to route from B via X to C
  - .. so X will not advertise to B a route to C

BGP routing policy (2)



- A advertises path AW to B
- B advertises path BAW to X
- Should B advertise path BAW to C?
  - No way! B gets no "revenue" for routing CBAW since neither W nor C are B's customers
  - B wants to force C to route to w via A
  - B wants to route *only* to/from its customers!

Why different Intra- and Inter-AS routing?

Policy:

- Inter-AS: admin wants control over how its traffic routed, who routes through its net.
- Intra-AS: single admin, so no policy decisions needed

Scale:

- hierarchical routing saves table size, reduced update traffic

Performance:

- Intra-AS: can focus on performance
- Inter-AS: policy may dominate over performance

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

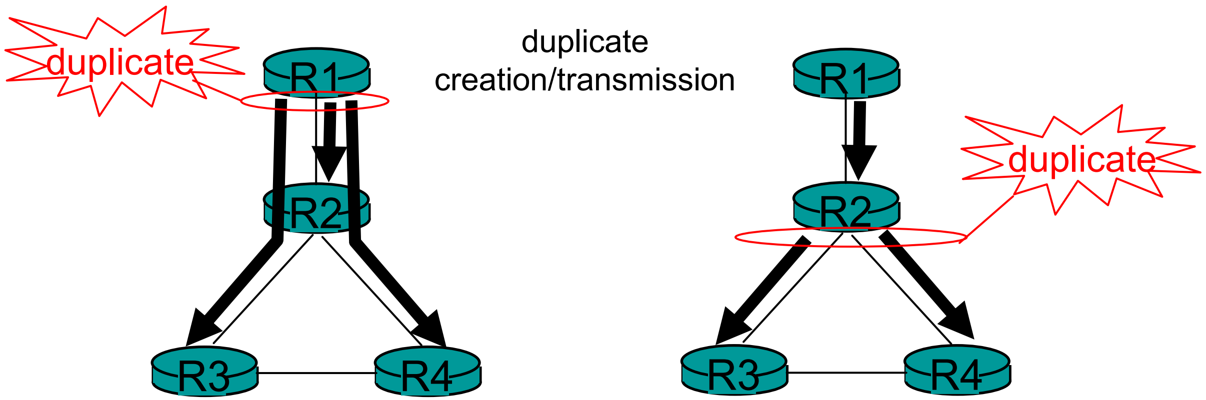
- IPv6
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
- **Broadcast and multicast routing**

Broadcast Routing

- deliver packets from source to all other nodes
- source duplication is inefficient:



source duplication

in-network duplication

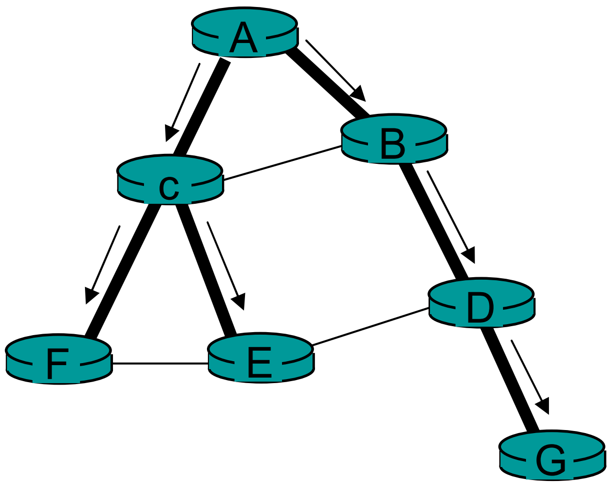
- source duplication: how does source determine recipient addresses?

In-network duplication

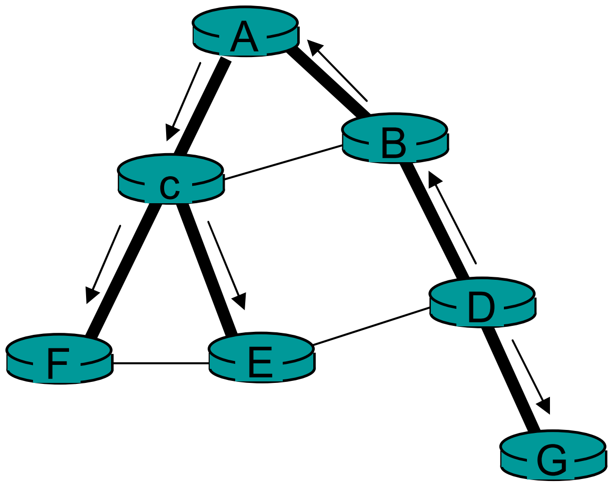
- flooding: when node receives brdcst pckt, sends copy to all neighbors
  - Problems: cycles & broadcast storm
- controlled flooding: node only brdcsts pkt if it hasn’t brdcst same packet before
  - Node keeps track of pckt ids already brdcsted
  - Or reverse path forwarding (RPF): only forward pckt if it arrived on shortest path between node and source
- spanning tree
  - No redundant packets received by any node

Spanning Tree

- First construct a spanning tree
- Nodes forward copies only along spanning tree



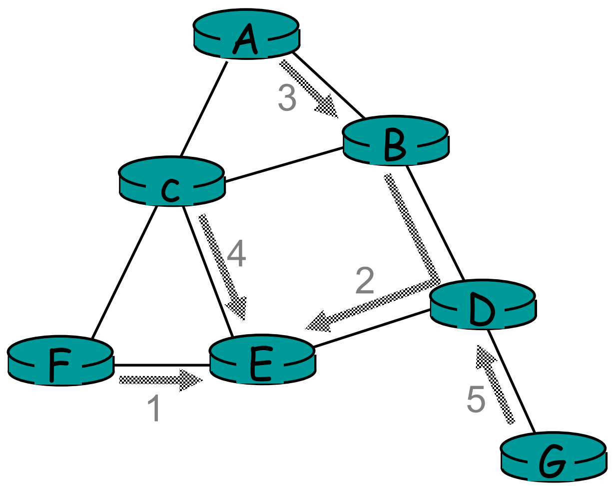
(a) Broadcast initiated at A



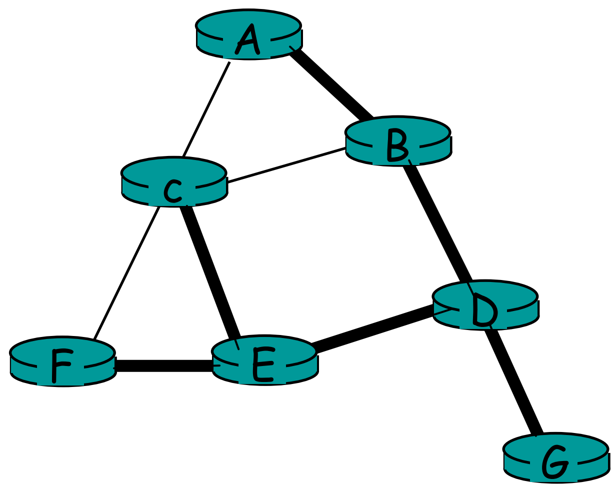
(b) Broadcast initiated at D

Spanning Tree: Creation

- Center node
- Each node sends unicast join message to center node
  - Message forwarded until it arrives at a node already belonging to spanning tree



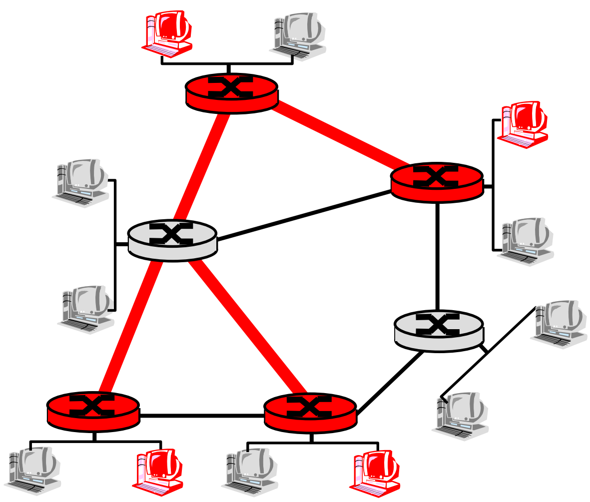
(a) Stepwise construction of spanning tree



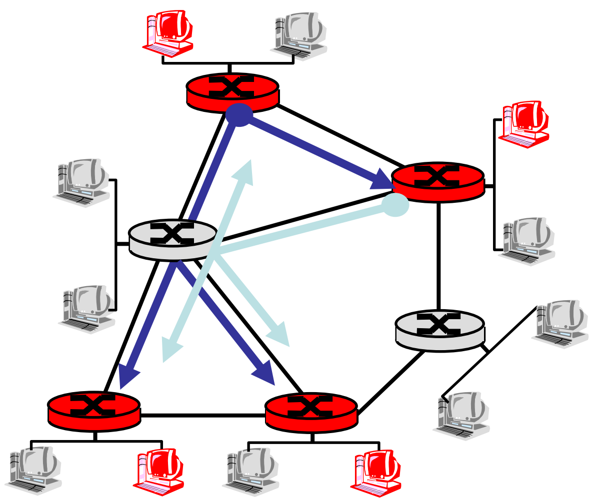
(b) Constructed spanning tree

Multicast Routing: Problem Statement

- ***Goal:*** find a tree (or trees) connecting routers having local mcast group members
  - *tree:* not all paths between routers used
  - *source-based:* different tree from each sender to rcvrs
  - *shared-tree:* same tree used by all group members



Shared tree



Source-based trees

Approaches for building mcast trees

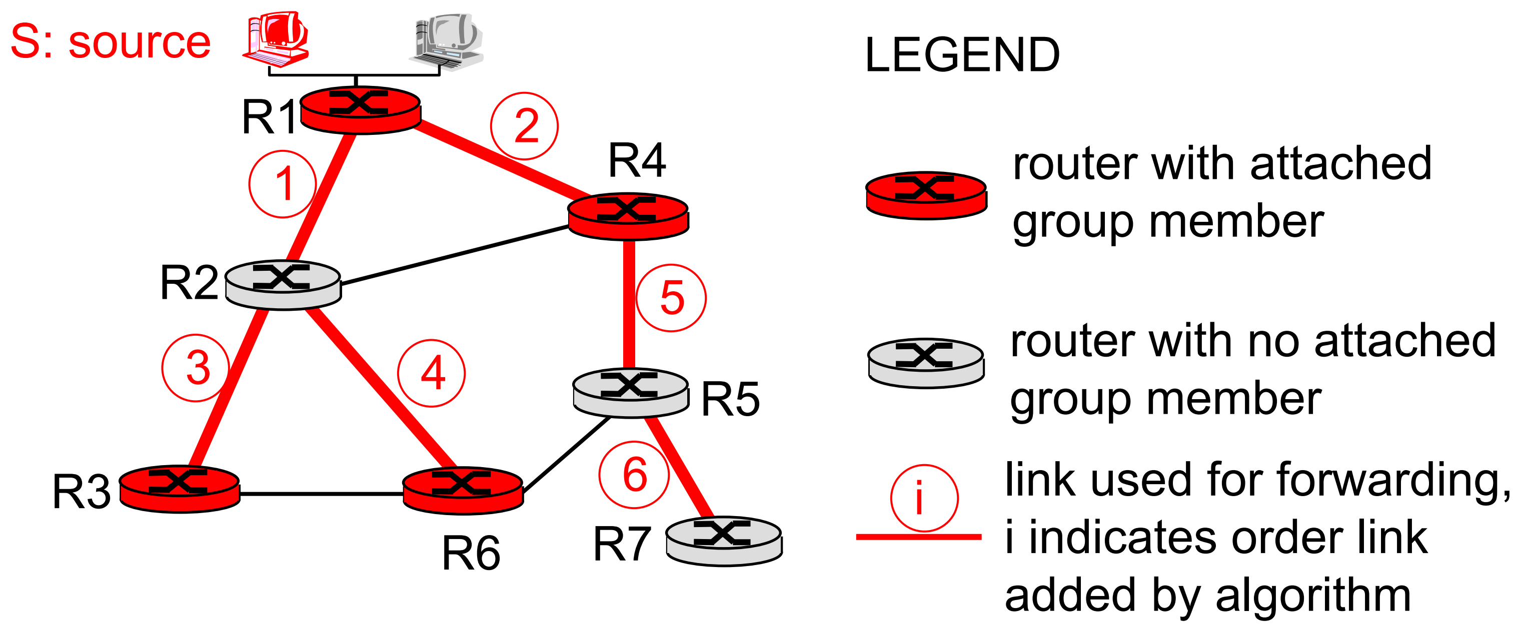
Approaches:

- source-based tree: one tree per source
  - shortest path trees
  - reverse path forwarding
- group-shared tree: group uses one tree
  - minimal spanning (Steiner)
  - center-based trees

…we first look at basic approaches, then specific protocols adopting these approaches

Shortest Path Tree

- mcast forwarding tree: tree of shortest path routes from source to all receivers
  - Dijkstra's algorithm

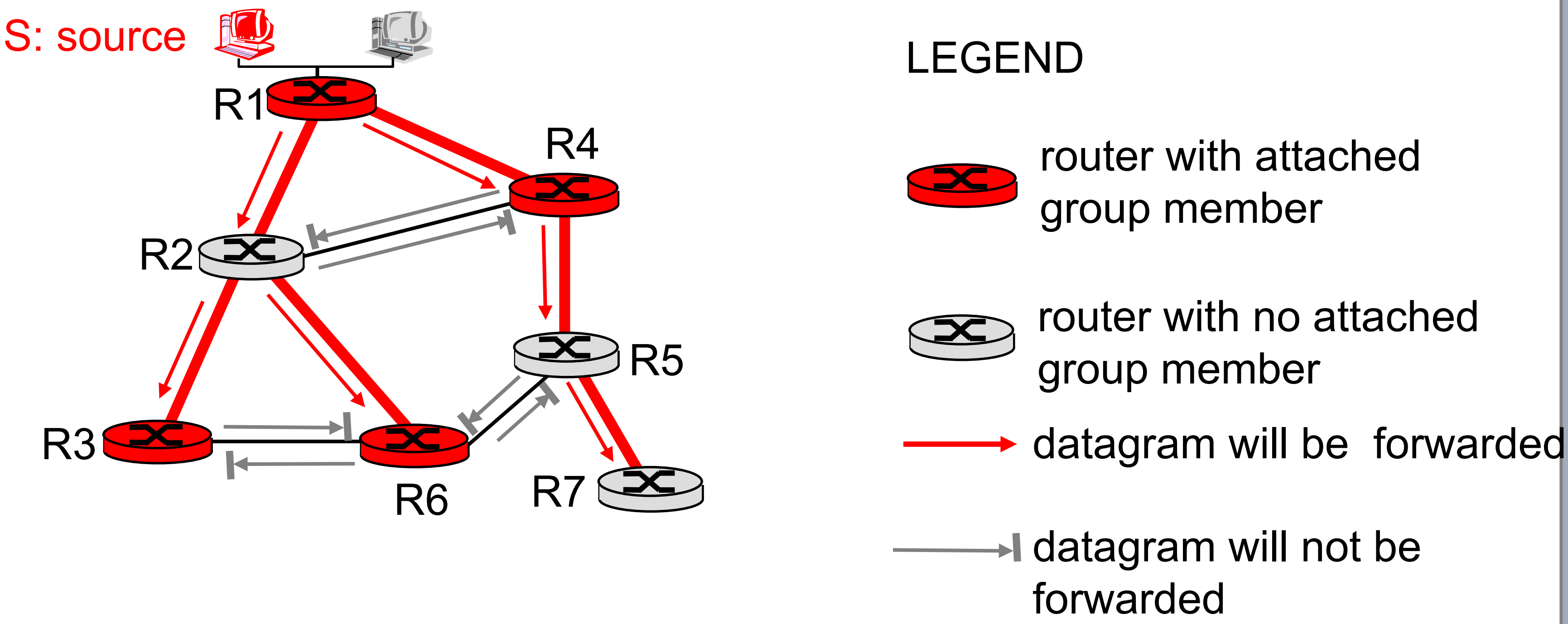


Reverse Path Forwarding

- rely on router's knowledge of unicast shortest path from it to sender
- each router has simple forwarding behavior:

if (mcast datagram received on incoming link on shortest path back to center)
  then flood datagram onto all outgoing links
  else ignore datagram

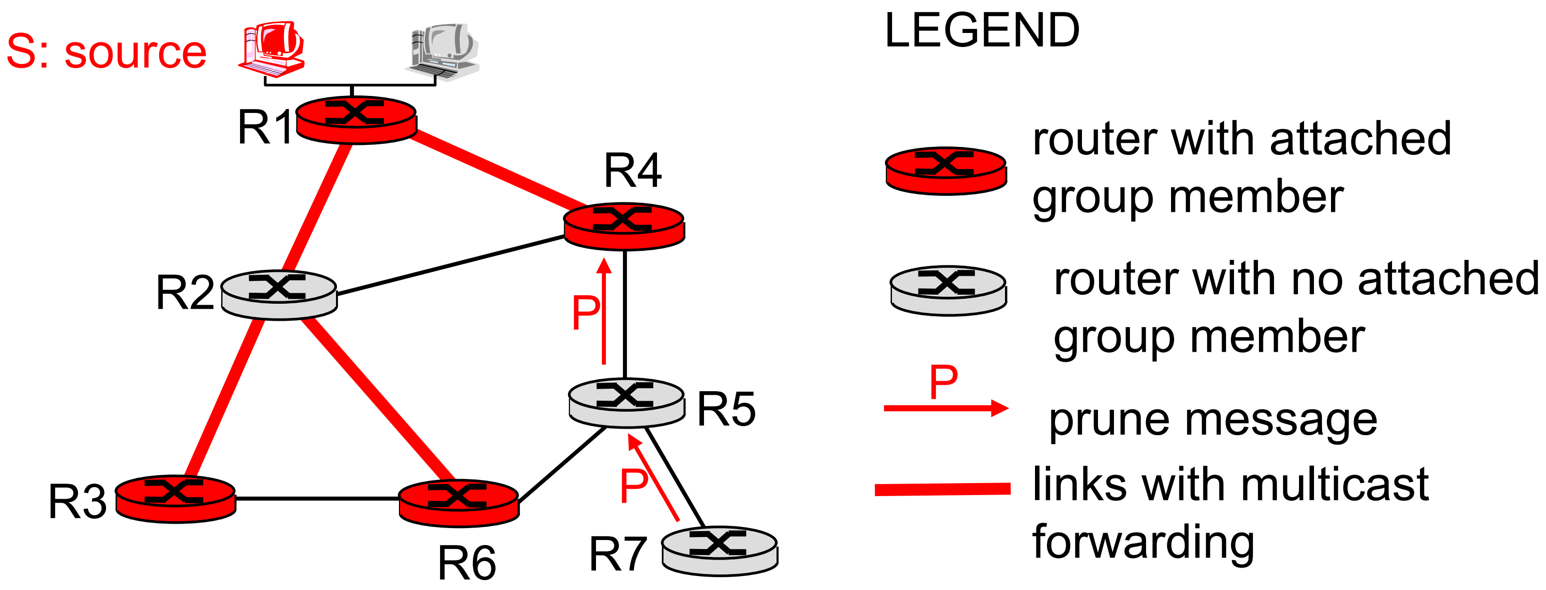
Reverse Path Forwarding: example



- result is a source-specific *reverse* SPT
  - may be a bad choice with asymmetric links

Reverse Path Forwarding: pruning

- forwarding tree contains subtrees with no mcast group members
  - no need to forward datagrams down subtree
  - "prune" msgs sent upstream by router with no downstream group members



Shared-Tree: Steiner Tree

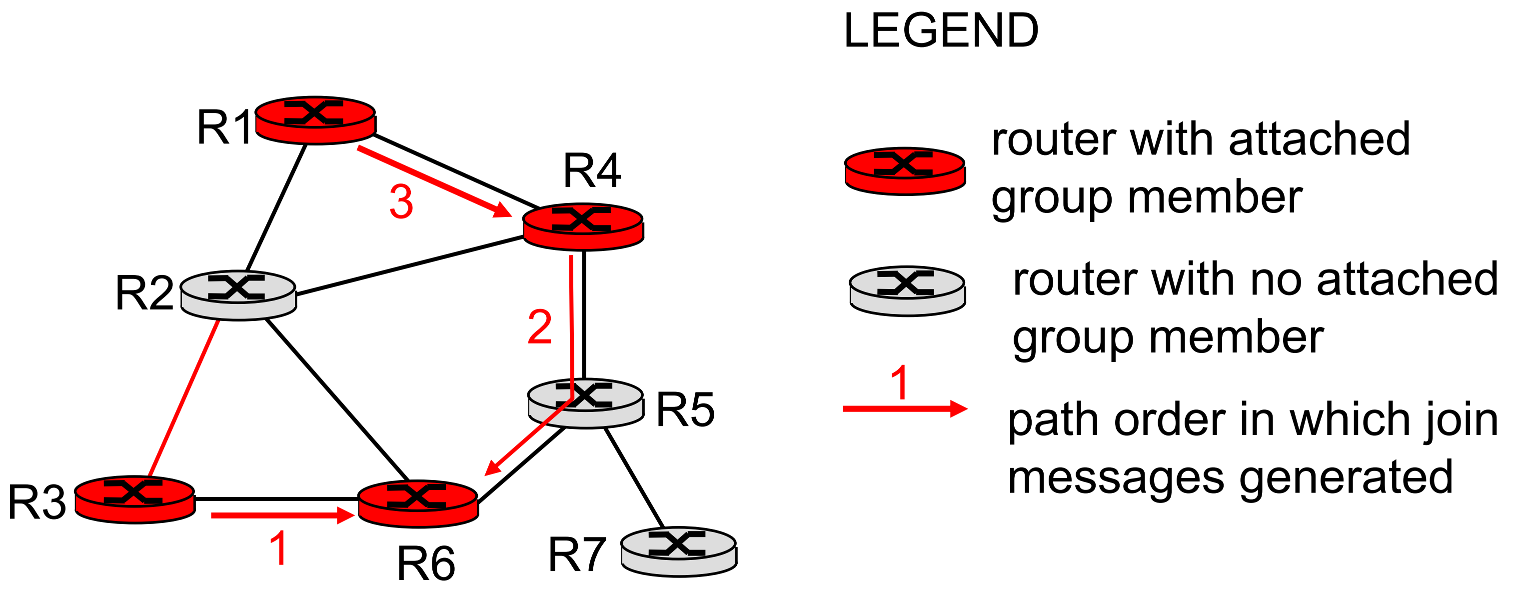
- Steiner Tree: minimum cost tree connecting all routers with attached group members
- problem is NP-complete
- excellent heuristics exists
- not used in practice:
  - computational complexity
  - information about entire network needed
  - monolithic: rerun whenever a router needs to join/leave

Center-based trees

- single delivery tree shared by all
- one router identified as *“center”* of tree
- to join:
  - edge router sends unicast *join-msg* addressed to center router
  - *join-msg* “processed” by intermediate routers and forwarded towards center
  - *join-msg* either hits existing tree branch for this center, or arrives at center
  - path taken by *join-msg* becomes new branch of tree for this router

Center-based trees: an example

Suppose R6 chosen as center:



Internet Multicasting Routing: DVMRP

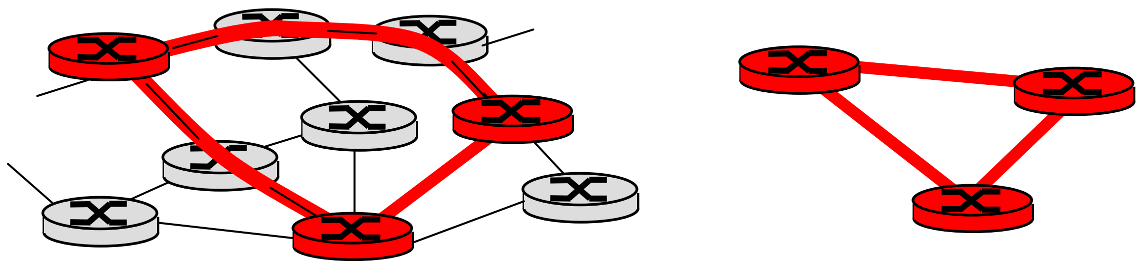
- DVMRP: distance vector multicast routing protocol, RFC1075
- *flood and prune:* reverse path forwarding, source-based tree
  - RPF tree based on DVMRP's own routing tables constructed by communicating DVMRP routers
  - no assumptions about underlying unicast
  - initial datagram to mcast group flooded everywhere via RPF
  - routers not wanting group: send upstream prune msgs

DVMRP: continued…

- *soft state:* DVMRP router periodically (1 min.) “forgets” branches are pruned:
  - mcast data again flows down unpruned branch
  - downstream router: reprune or else continue to receive data
- routers can quickly regraft to tree
  - following IGMP join at leaf
- odds and ends
  - commonly implemented in commercial routers
  - Mbone routing done using DVMRP

Tunneling

Q: How to connect "islands" of multicast routers in a "sea" of unicast routers?



physical topology

logical topology

- mcast datagram encapsulated inside "normal" (non-multicast-addressed) datagram
- normal IP datagram sent thru "tunnel" via regular IP unicast to receiving mcast router
- receiving mcast router unencapsulates to get mcast datagram

PIM: Protocol Independent Multicast

- not dependent on any specific underlying unicast routing algorithm (works with all)
- two different multicast distribution scenarios :

Dense:

- group members densely packed, in "close" proximity.
- bandwidth more plentiful

Sparse:

- # networks with group members small wrt # interconnected networks
- group members "widely dispersed"
- bandwidth not plentiful

Consequences of Sparse-Dense Dichotomy:

Dense

- group membership by routers *assumed* until routers explicitly prune
- *data-driven* construction on mcast tree (e.g., RPF)
- bandwidth and non-group-router processing *profligate*

Sparse:

- no membership until routers explicitly join
- *receiver- driven* construction of mcast tree (e.g., center-based)
- bandwidth and non-group-router processing *conservative*

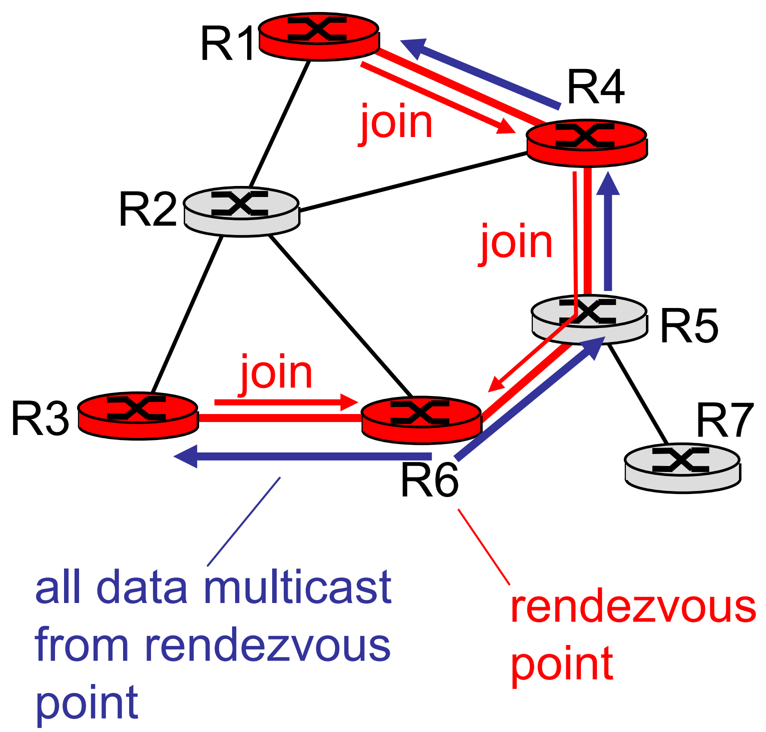
PIM- Dense Mode

flood-and-prune RPF, similar to DVMRP but

- underlying unicast protocol provides RPF info for incoming datagram
- less complicated (less efficient) downstream flood than DVMRP reduces reliance on underlying routing algorithm
- has protocol mechanism for router to detect it is a leaf-node router

PIM - Sparse Mode

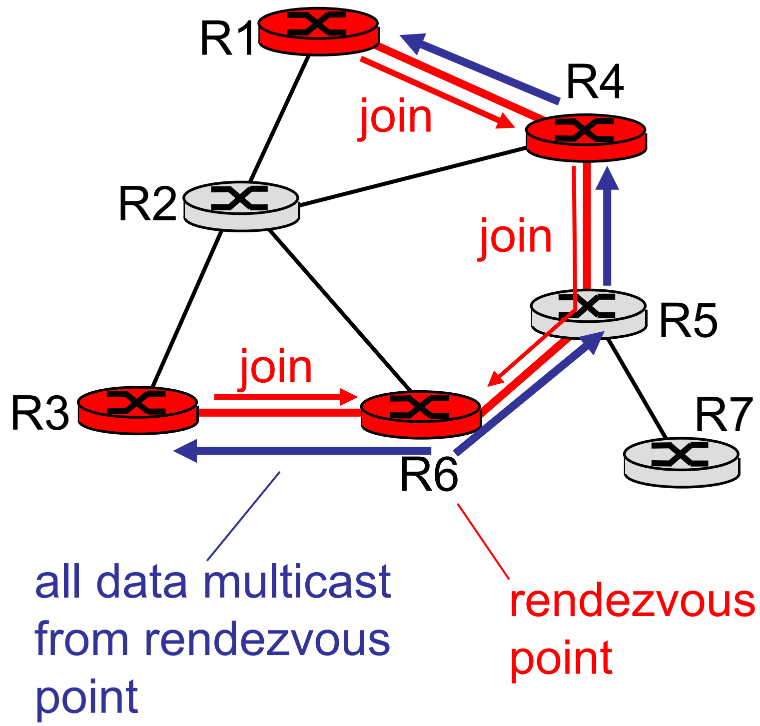
- center-based approach
- router sends *join* msg to rendezvous point (RP)
  - intermediate routers update state and forward *join*
- after joining via RP, router can switch to source-specific tree
  - increased performance: less concentration, shorter paths



PIM - Sparse Mode

sender(s):

- unicast data to RP, which distributes down RP-rooted tree
- RP can extend mcast tree upstream to source
- RP can send *stop* msg if no attached receivers
  - "no one is listening!"



Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
- Broadcast and multicast routing

Master Course
Computer Networks
IN2097

Prof. Dr.-Ing. Georg Carle
Christian Grothoff, Ph.D.
Dr. Nils Kammenhuber

Chair for Network Architectures and Services
Institut für Informatik
Technische Universität München
http://www.net.in.tum.de

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- NAT
- Virtual circuit and datagram networks
- What’s inside a router

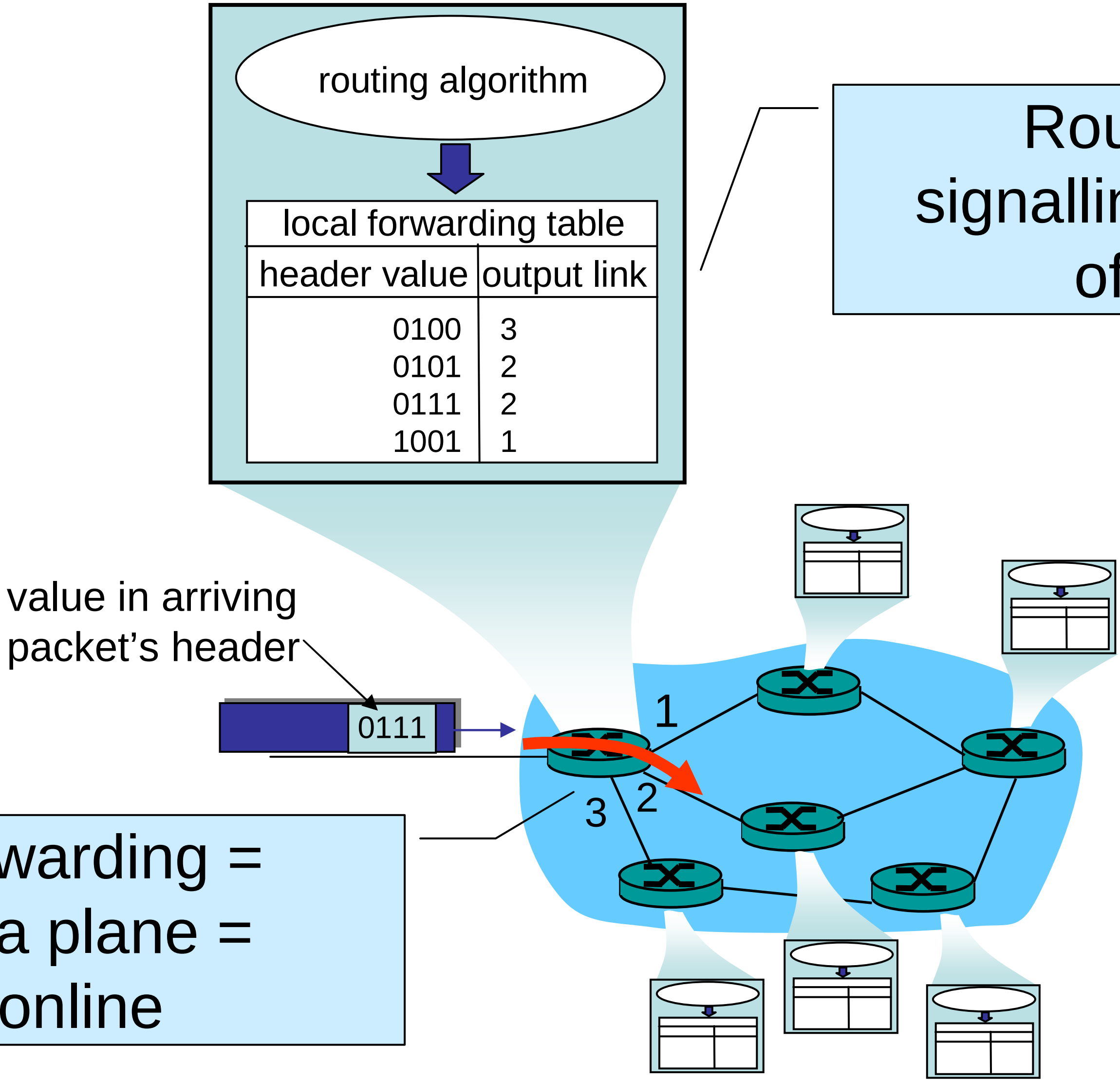
Part 3

- **Routing algorithms**
  - Link state
  - Distance Vector
  - Path Vector
  - Hierarchical routing
- Internet routing protocols
  - RIP
  - OSPF
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering

Short note on pronunciation of the word “routing”

- [‘ru:tɪŋ] r-oo-ting = British English
- [‘raʊdɪŋ] r-ow-ding = American English
- Both are correct!

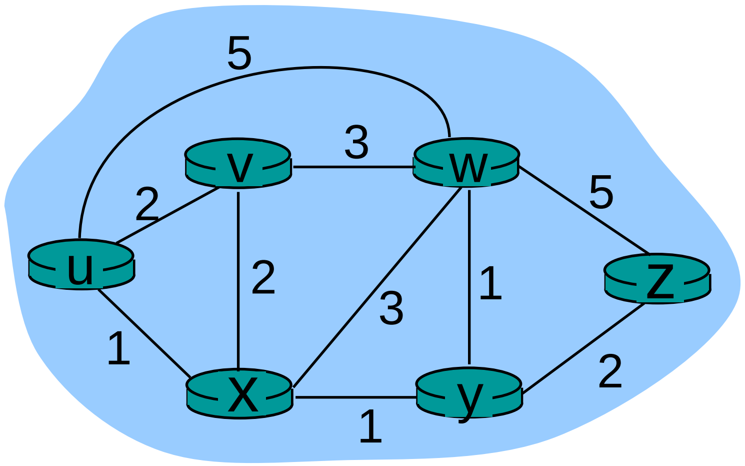
Recall: Interplay between routing and forwarding



Routing =
signalling plane =
offline

Forwarding =
data plane =
online

Graph abstraction: costs



- $c(x,x')$ =: cost of link (x,x')
  e.g.: $c(w,z) = 5$

- cost could always be 1,
- or inversely related to bandwidth,
- or inversely related to congestion

Cost of path $(x_1, x_2, x_3,\ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

Question: What's the least-cost path between u and z ?

Routing algorithm: algorithm that finds least-cost path

Routing Algorithm classification

Global or decentralized information?

Global:

- All routers have complete topology and link cost info
- *link state* algorithms (L-S)

Decentralized:

- Router only knows physically-connected neighbors and link costs to neighbors
- Iterative process of computation = exchange of info with neighbors
- *distance vector* algorithms (D-V)
- Variant: *path vector* algorithms

Static or dynamic?

Static:

- Routes change slowly over time

Dynamic:

- Routes change more quickly
  - periodic update
  - in response to link cost changes

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- NAT
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- **Routing algorithms**
  - **Link state**
  - Distance Vector
  - Path Vector
  - Hierarchical routing
- Internet routing protocols
  - RIP
  - OSPF
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering

A Link-State Routing Algorithm

- Net topology and link costs made known to each node
  - Accomplished via *link state broadcasts*
  - All nodes have same info
- Each node independently computes least-cost paths from one node ("source") to all other nodes
  - Usually done using Dijkstra's shortest-path algorithm
    - refer to any algorithms & data structures lecture/textbook
    - n nodes in network $\Rightarrow$ $O(n^2)$ or $O(n \log n)$
  - Gives forwarding table for that node
- Result:
  - All nodes have the same information,
  - … thus calculate the same shortest paths,
  - … hence obtain consistent forwarding tables

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- NAT
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- **Routing algorithms**
  - Link state
  - **Distance Vector**
  - Path Vector
  - Hierarchical routing
- Internet routing protocols
  - RIP
  - OSPF
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering

Distance Vector Algorithm

- No node knows entire topology
- Nodes only communicate with neighbours (i.e., no broadcasts)
- Nodes *jointly* calculate shortest paths
  - Iterative process
  - Algorithm == protocol
- Distributed application of Bellman-Ford algorithm
  - refer to any algorithms&data structures lecture/textbook

Distance Vector Algorithm

<u>Bellman-Ford Equation (dynamic programming)</u>

Let

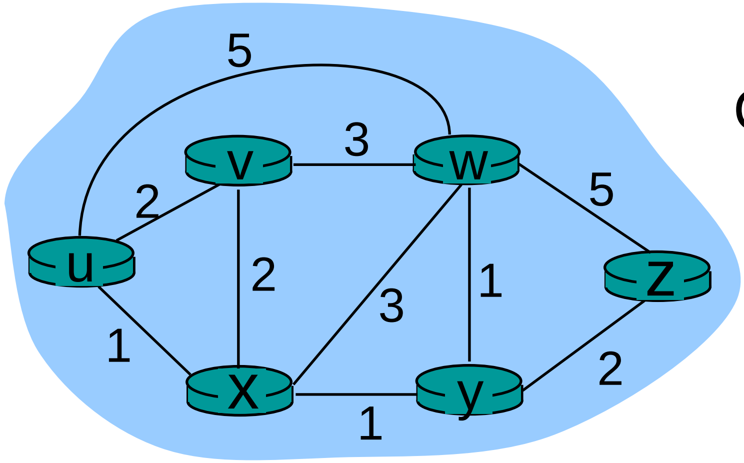
- $c(x,y)$:= cost of edge from x to y
- $d_x(y)$:= cost of least-cost path from x to y

Then

$d_x(y) = \min \{c(x,v) + d_v(y) \}$

where min is taken over all neighbours v of x

Bellman-Ford example



Clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$\begin{aligned} d_u(z) &= \min \{ c(u,v) + d_v(z), \\ &\qquad c(u,x) + d_x(z), \\ &\qquad c(u,w) + d_w(z) \} \\ &= \min \{2 + 5, \\ &\qquad 1 + 3, \\ &\qquad 5 + 3\} = 4 \end{aligned}$$

Node that achieves minimum is next hop in shortest path
→ forwarding table

Distance Vector Algorithm

- Define $D_x(y)$:= estimate of least cost from x to y
- Node x knows cost to each neighbour v: $c(x,v)$
- Node x maintains distance vector $\mathbf{D}_x = [\, D_x(y){:}\ y \in N \,]$
  (N := set of nodes)
- Node x also maintains its neighbours' distance vectors:
  - For each neighbour v,
    x maintains $\mathbf{D}_v = [\, D_v(y){:}\ y \in N \,]$

Distance vector algorithm (4)

Basic idea:

- From time-to-time, each node sends its own distance vector estimate D to neighbors
  - Asynchronously
- When a node x receives new DV estimate from neighbour, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

- Under minor, natural conditions, these estimates $D_x(y)$ *converge to the actual least cost* $d_x(y)$

Distance Vector Algorithm (5)

Iterative, asynchronous:
each local iteration caused by:

- local link cost change
- DV update message from neighbour

Distributed:

- Each node notifies neighbors *only* when its DV changes
    - neighbours then notify their neighbours if this caused *their* DV to change
    - etc.

Each node:

Forever:

wait for (change in local link cost *or* message arriving from neighbour}

recompute estimates

if (DV to any destination has changed) { *notify* neighbours }

Distance Vector Algorithm (6)

node x table

cost to

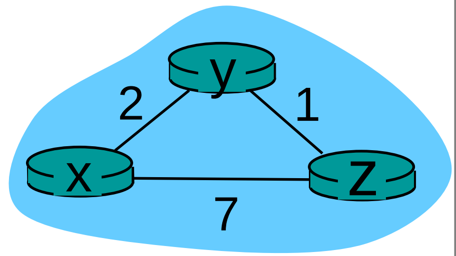
from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$
$= \min\{2+0\ ,\ 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\}$
$= \min\{2+1\ ,\ 7+0\} = 3$



node y table

cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

cost to

from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

time

$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$
$= \min\{2+0\ ,\ 7+1\} = 2$

$D_x(z) = \min\{c(x,y) +$
$D_y(z),\ c(x,z) + D_z(z)\}$
$= \min\{2+1\ ,\ 7+0\} = 3$

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

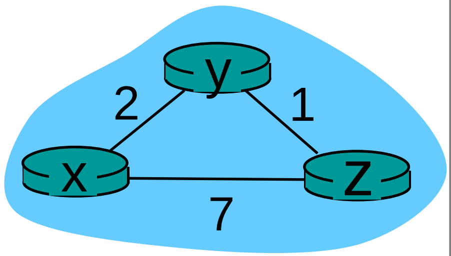
node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

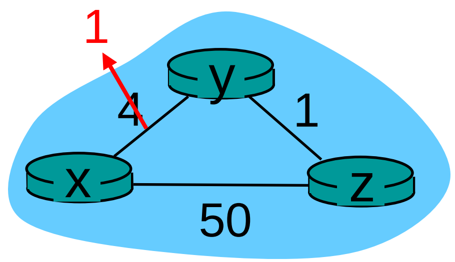
time



Distance Vector: link cost changes (1)

Link cost changes:

- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notify neighbors



"good news travels fast"

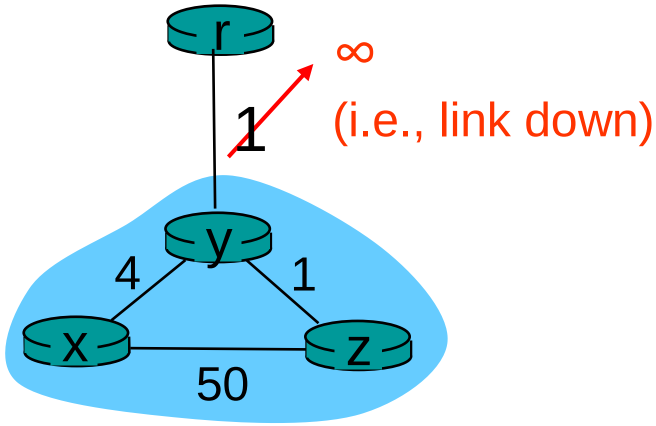
At time t_0, y detects the link-cost change, updates its DV, and informs its neighbors.

At time t_1, z receives the update from y and updates its table. It computes a new least cost to x and sends its neighbors its DV.

At time t_2, y receives z's update and updates its distance table. y's least costs do not change and hence y does *not* send any message to z.

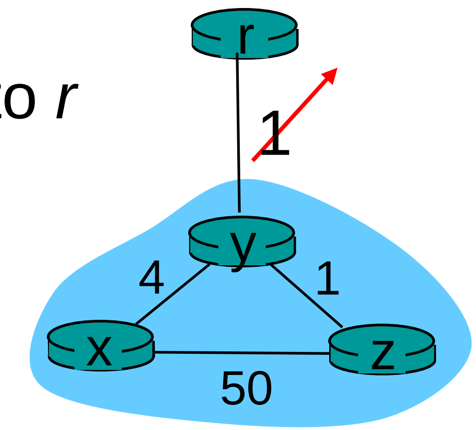
Distance Vector: link cost changes (2)

- But: bad news travels slow — “count to infinity” problem!
- In example: Many iterations before algorithm stabilizes!
  1. Cost increase for *y*→*r*:
     - *y* consults DV,
     - *y* selects “cheaper” route via *z* (cost 2+1 = 3),
     - Sends update to *z* and *x* (cost to *r* now 3 instead of 1)
  2. *z* detects cost increase for path to *r*:
     - was 1+1, is now 3+1
     - Sends update to *y* and *x* (cost to *r* now 4 instead of 2)
  3. *y* detects cost increase, sends update to *z*
  4. *z* detects cost increase, sends update to *y*
  5. ….



Distance Vector: Solutions that only half work

- Finite infinity: Define some number to be ∞ (in RIP: 16 := ∞)
- Split Horizon:
  - Tell to a neighbour that is part of a best path to a destination that the destination cannot be reached
  - If *z* routes through *y* to get to *r*
    z tells *y* that its own (i.e., *y*'s) distance to *r* is infinite (so *y* won't route to *r* via *z*)
- Poisoned Reverse:
  - In addition, *actively* advertise a route as unreachable to the neighbour from which the route was learned



- (Warning: Terms often used interchangeably!)
- Often help, but cannot solve all problem instances
- Can significantly increase number of routing messages

Comparison of LS and DV algorithms

Message complexity

- LS: with *n* nodes, *E* links, O(*nE*) msgs sent
- DV: exchange between neighbors only
  - convergence time varies

Speed of Convergence

- LS: O(n²) algorithm requires O(nE) msgs
  - may have oscillations
- DV: convergence time varies
  - may be routing loops
  - count-to-infinity problem

Robustness: what happens if router malfunctions?

LS:

- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagate thru network

Part 1
- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2
- IPv6
- NAT
- Virtual circuit and datagram networks
- What's inside a router

Part 3
- **Routing algorithms**
  - Link state
  - Distance Vector
  - **Path Vector**
  - Hierarchical routing
- Internet routing protocols
  - RIP
  - OSPF
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering

Path Vector protocols

- Problem with D-V protocol:
  Path cost is “anonymous” single number
- Path Vector protocol:
  - For each destination, advertise entire path (=sequence of node identifiers) to neighbours
  - Cost calculation can be done by looking at path
  - Easy loop detection: Does my node ID already appear in the path?
- Not used very often
  - only in BGP …
  - … and BGP is much more complex than just paths!

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- NAT
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- **Routing algorithms**
  - Link state
  - Distance Vector
  - Path Vector
  - **Hierarchical routing**
- Internet routing protocols
  - RIP
  - OSPF
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering

Hierarchical Routing

Our routing study thus far = idealisation

- all routers identical
- network “flat”

… *not* true in practice!

Scale = billions of destinations:

- Can’t store all destinations in routing tables!
- Routing table exchange would swamp links!

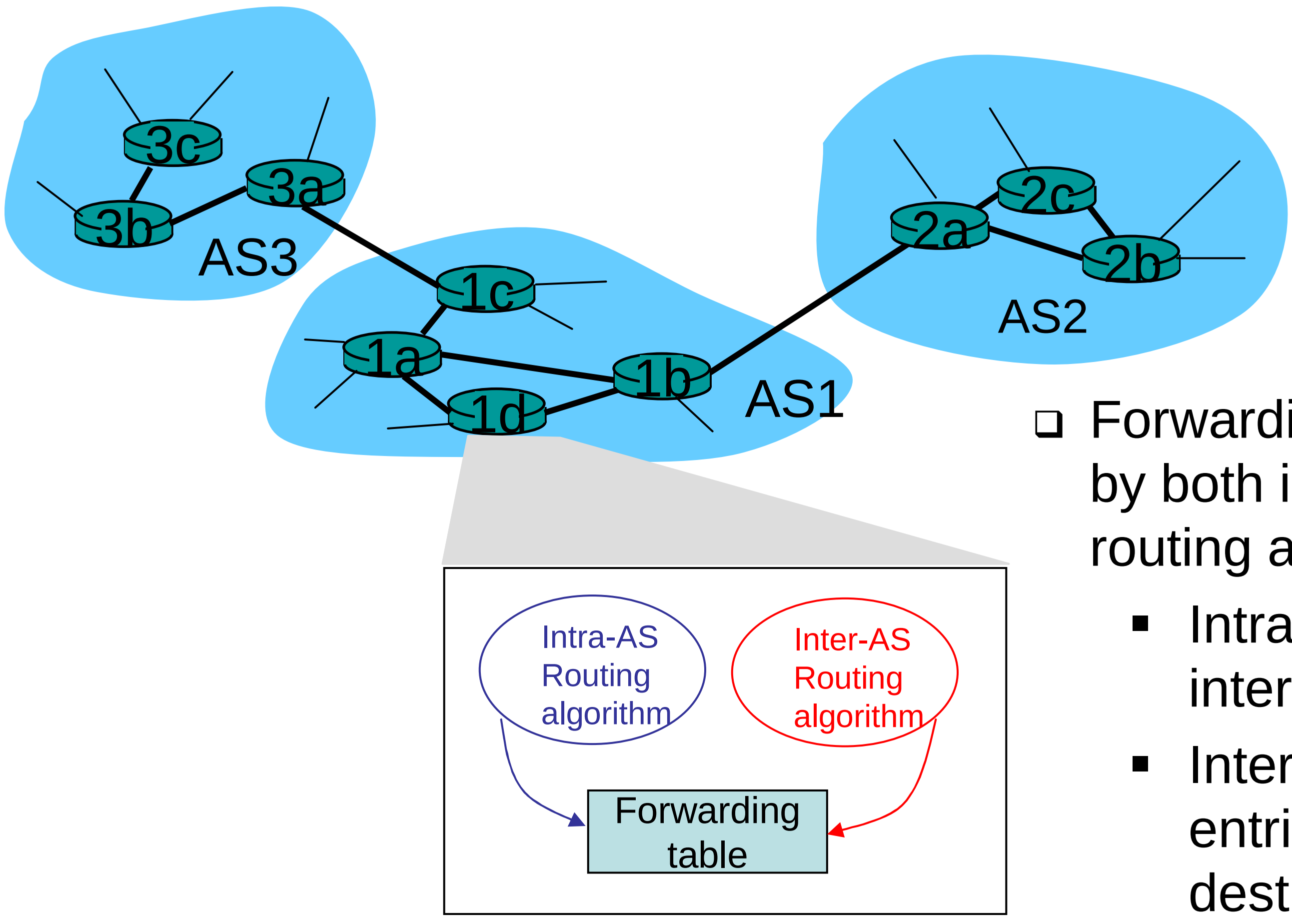
Administrative autonomy

- Internet = network of networks
- Each network admin may want to control routing in its own network — no central administration!

Hierarchical Routing

- Aggregate routers into regions called
  "autonomous systems" (short: AS; plural: ASes)
- Routers in same AS run same routing protocol
  - = "intr**a**-AS" routing protocol (also called "intradomain")
  - Routers in different ASes can run different intra-AS routing protocols
- ASes are connected: via gateway routers
  - Direct link to [gateway] router in another AS
    = "int**er**-AS" routing protocol (also called "interdomain")
  - Warning: Non-gateway routers need to know about inter-AS routing as well!

Interconnected ASes



- Forwarding table configured by both intra- *and* inter-AS routing algorithm:
  - Intra-AS sets entries for internal destinations
  - Inter-AS *and* intra-AS set entries for external destinations

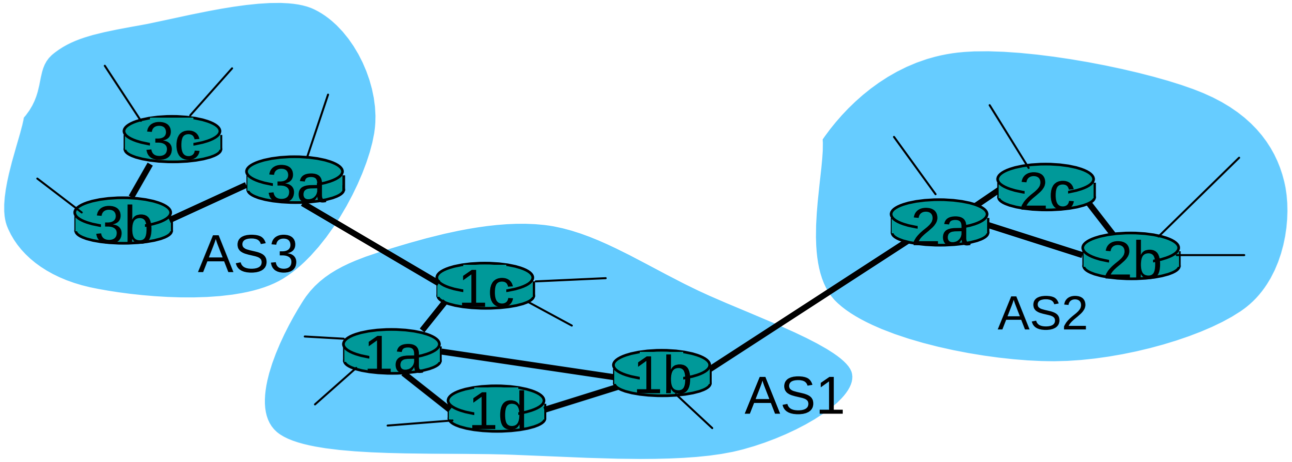
Inter-AS tasks

- Suppose router in AS1 receives datagram destined outside of AS1:
  - router should forward packet to gateway router, but which one?

AS1 must:

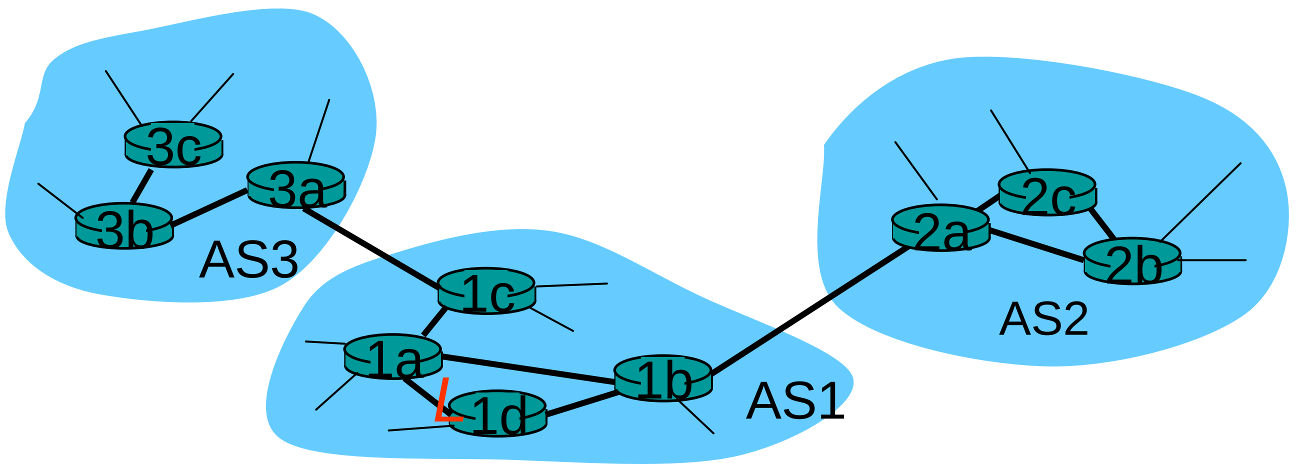
1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info *to all* routers in AS1 (i.e., not just the gateway routers)

Job of inter-AS routing!



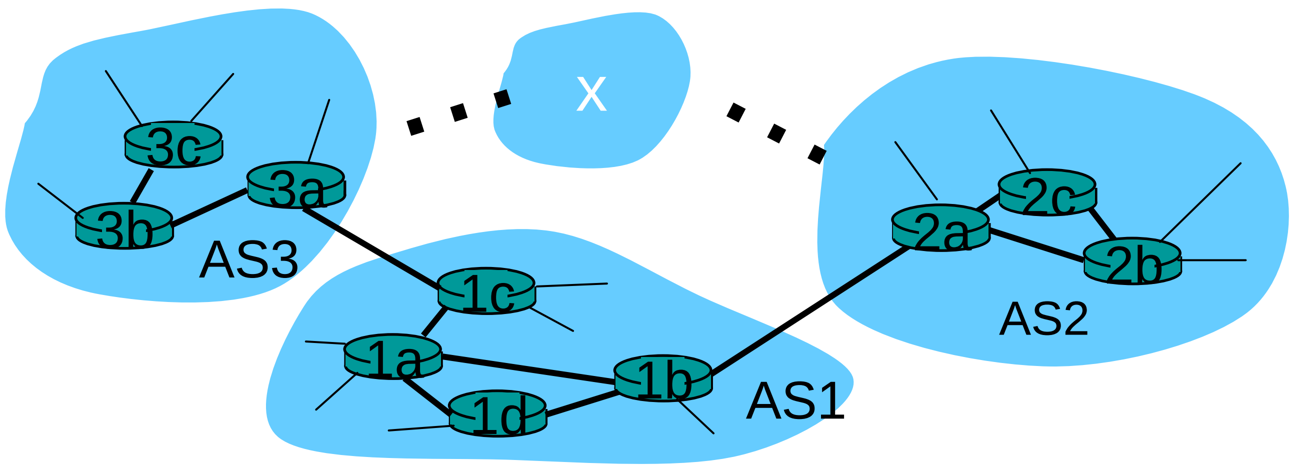
Example: Setting forwarding table in router 1d

- Suppose AS1 learns (via inter-AS protocol) that subnet *x* is reachable via AS3 (gateway 1c) but not via AS2.
- Inter-AS protocol propagates reachability info to all internal routers.
- Router 1d determines from intra-AS routing info that its interface *I* (i.e., interface to 1a) is on the least cost path to 1c.
  - installs forwarding table entry *(x,I)*



Example: Choosing among multiple ASes

- Now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- To configure forwarding table, router 1d must determine towards which gateway it should forward packets for destination x.
  - This is also job of inter-AS routing protocol!



Interplay of inter-AS and intra-AS routing

- Inter-AS routing
  - Only for destinations outside of own AS
  - Used to determine gateway router
  - Also: Steers transit traffic
    (from AS *x* to AS *y* via our own AS)
- Intra-AS routing
  - Used for destinations within own AS
  - Used to reach gateway router for outside destinations

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- NAT
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Path Vector
  - Hierarchical routing
- **Internet routing protocols**
  - (RIP)
  - OSPF
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering

Recall: Inter-AS and intra-AS routing

- Inter-AS routing
  - Only for destinations outside of own AS
  - Used to determine gateway router
  - Also: Steers transit traffic (from AS *x* to AS *y* via our own AS)
- Intra-AS routing
  - Used for destinations within own AS
  - Used to reach gateway router for outside destinations

⇒ Routers need to run *both* types of routing protocols

Intra-AS Routing

- Also known as Interior Gateway Protocols (IGP)
- Most common Intra-AS routing protocols:
  - RIP: Routing Information Protocol — DV (typically small systems)
  - OSPF: Open Shortest Path First — hierarchical LS (typically medium to large systems)
  - IS-IS: Intermediate System to Intermediate System — hierarchical LS (typically medium-sized ASes)
  - (E)IGRP: (Enhanced) Interior Gateway Routing Protocol (Cisco proprietary) — hybrid of LS and DV

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- NAT
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Path Vector
  - Hierarchical routing
- **Internet routing protocols**
  - (RIP)
  - **OSPF**
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering

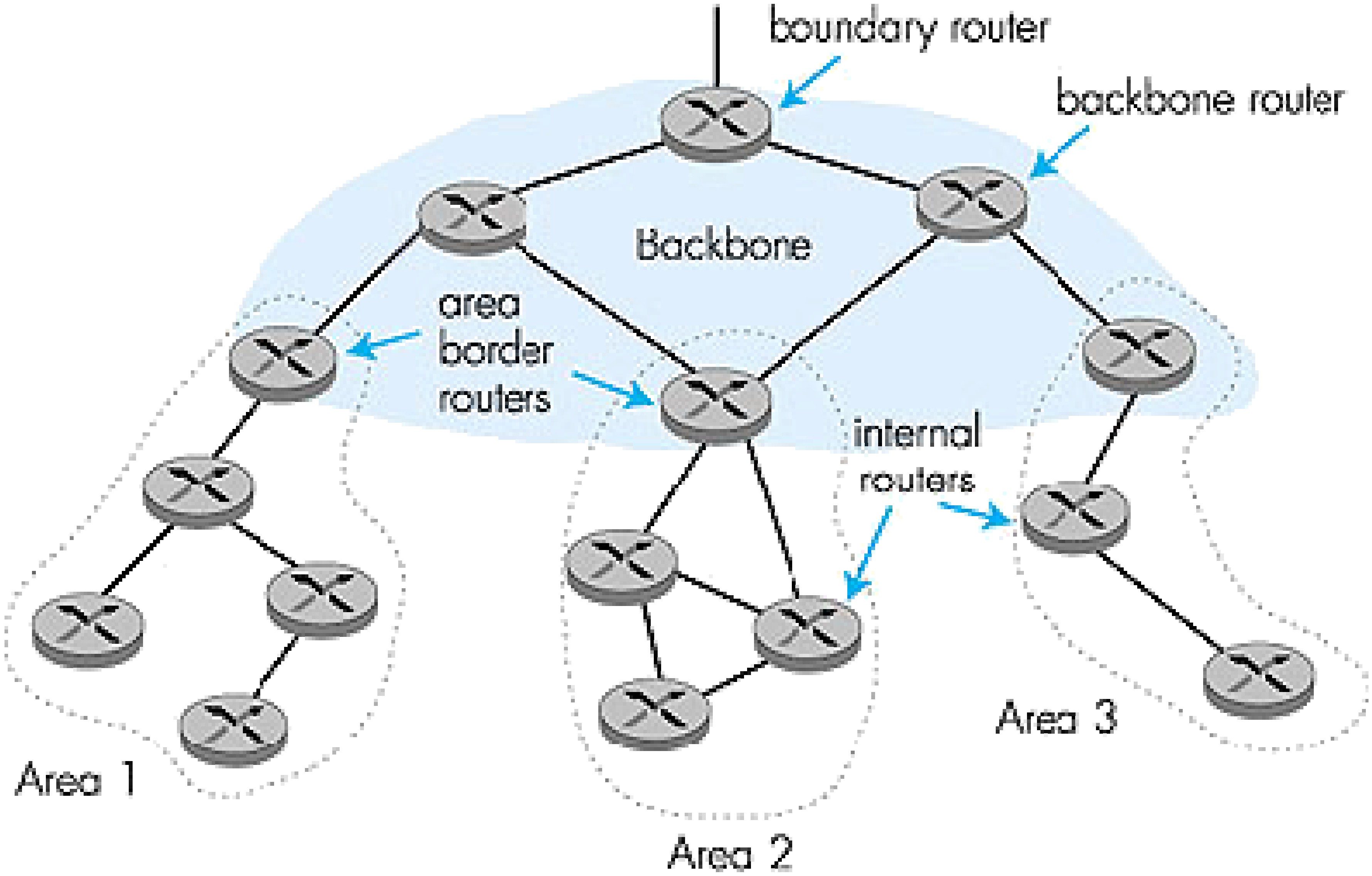
OSPF (Open Shortest Path First)

- "Open": publicly available (vs. vendor-specific, e.g., EIGRP = Cisco-proprietary)
- Uses Link State algorithm
  - LS packet dissemination (broadcasts)
  - Unidirectional edges (⇒costs may differ by direction)
  - Topology map at each node
  - Route computation using Dijkstra's algorithm
- OSPF advertisement carries one entry per neighbour router
- Advertisements disseminated to entire AS (via flooding)
  - (exception: hierarchical OSPF, see next slides)
  - carried in OSPF messages directly over IP (rather than TCP or UDP)

OSPF "advanced" features

- Security: all OSPF messages authenticated (to prevent malicious intrusion)
- Multiple same-cost paths allowed (only one path in RIP): *ECMP* (equal-cost multipath)
- For each link, multiple cost metrics for different Type of Service (TOS):
  e.g., satellite link cost set "low" for best effort, but high for real time
- Integrated uni- and multicast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- Hierarchical OSPF in large domains

Hierarchical OSPF



Hierarchical OSPF

- OSPF *can* create a two-level hierarchy *within* an AS
  - Similar to inter-AS and intra-AS routing in Internet
- Two levels: local *areas* and the *backbone*
  - Link-state advertisements only within local area
  - Each node has detailed area topology; but only knows direction (shortest path) to networks in other areas
- *Area border routers:* "summarize" distances to networks in own area; advertise distances to other Area Border routers
- *Backbone routers:* run OSPF routing limited to backbone
- *Boundary routers:* connect to other ASes

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- NAT
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Path Vector
  - Hierarchical routing
- **Internet routing protocols**
  - RIP
  - OSPF
  - **BGP**
- Business considerations
  - Policy routing
  - Traffic engineering

Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol):
  The de facto standard for inter-AS routing
- BGP provides each AS a means to:
  1. Obtain subnet reachability information from neighbouring ASes.
  2. Propagate reachability information to all AS-internal routers.
  3. Determine “good” routes to subnets based on reachability information and policy.
- Allows an AS to advertise the existence of an IP prefix to rest of Internet: *“This subnet is here”*

BGP basics

- Pairs of routers (BGP peers) exchange routing info over semi-permanent TCP connections: BGP sessions
  - BGP sessions need not correspond to physical links!
- When AS2 advertises an IP prefix to AS1:
  - AS2 *promises* it will forward IP packets towards that prefix
  - AS2 can aggregate prefixes in its advertisement

How does BGP work?

- BGP = “path++” vector protocol
- BGP messages exchanged using TCP
  - Possible to run eBGP sessions not on border routers
- BGP Message types:
  - OPEN: set up new BGP session, after TCP handshake
  - NOTIFICATION: an error occurred in previous message
    → tear down BGP session, close TCP connection
  - KEEPALIVE: “null” data to prevent TCP timeout/auto-close; also used to acknowledge OPEN message
  - UPDATE:
    - Announcement: inform peer about new / changed route to some target
    - Withdrawal: (inform peer about non-reachability of a target)

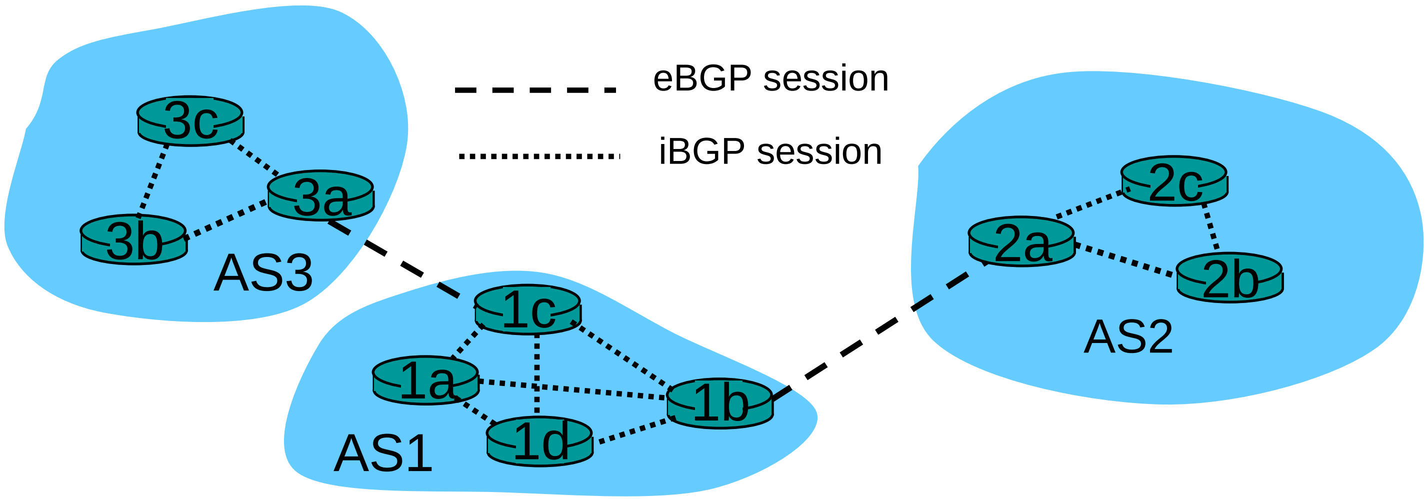
BGP updates

- Update (Announcement) message consists of
  - Destination (IP prefix)
  - AS Path (=Path vector)
  - Next hop (=IP address of our router connecting to other AS)
- …but update messages also contain a lot of further attributes:
  - Local Preference: used to prefer one gateway over another
  - Origin: route learned via { intra-AS | inter-AS | unknown }
  - MED, Community, …

⇒ Not a pure path vector protocol: More than just the path vector

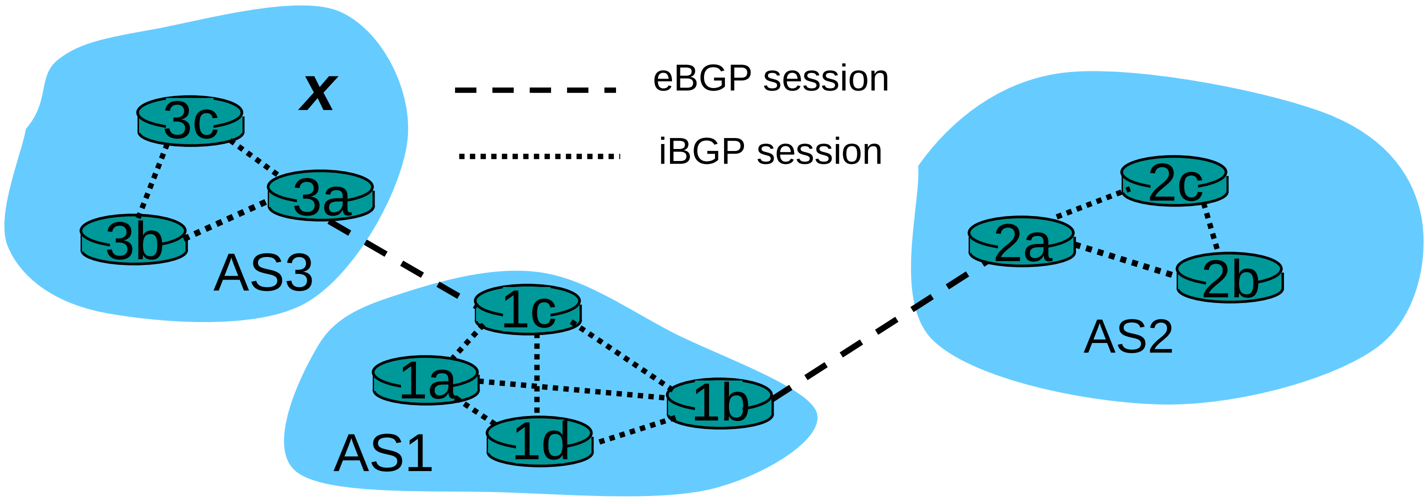
eBGP and iBGP

- External BGP: between routers in *different* ASes
- Internal BGP: between routers in *same* AS
  - Remember: In spite of intra-AS routing protocol, *all* routers need to know about external destinations (not only border routers)
- No different protocols — just slightly different configurations!



Distributing reachability info

- Using eBGP session between 3a and 1c, AS3 sends reachability info about prefix *x* to AS1.
  - 1c can then use iBGP to distribute new prefix info to all routers in AS1
  - 1b can then re-advertise new reachability info to AS2 over 1b-to-2a eBGP session
- When router learns of new prefix *x*, it creates entry for prefix in its forwarding table.



Path attributes & BGP routes

- Advertised prefix includes BGP attributes
  - prefix + attributes = “route”
- Most important attributes:
  - AS-PATH: contains ASs through which prefix advertisement has passed: e.g, AS 67, AS 17
    - ASes identified by *AS numbers*, e.g.,: lrz.de=AS12816
  - NEXT-HOP: indicates specific internal-AS router to next-hop AS. (may be multiple links from current AS to next-hop-AS)
- When gateway router receives route advertisement, it uses an import policy to accept/decline the route
  - More on this later

BGP route selection

- Router may learn about more than 1 route to some prefix
  ⇒ Router must select route.
- Elimination rules:
  1. Local preference value attribute: policy decision
  2. Shortest AS-PATH
  3. Closest NEXT-HOP router: hot potato routing
  4. Additional criteria

BGP Route Reflectors (RR)

- Every router in AS should know external routes
  - Not only local neighbours, but also neighbours connected at other routers
  - ⇒ Many/all routers in AS have to run BGP sessions
- Need to select best inter-AS routes
  - ⇒ Routers need to exchange routing information via iBGP
- $O(n)$ BGP routers ⇒ $O(n^2)$ iBGP sessions ϟ ϟ ϟ
- **Idea:**
  - One special router = Route Reflector (RR)
  - Every **e**BGP router sends routes learned from **e**BGP via iBGP to RR
  - RR collects routes, may do policing
  - RR distributes routes to all other BGP routers in AS via iBGP
  - $O(n)$ BGP routers, $O(n)$ BGP sessions ☺

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- NAT
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Path Vector
  - Hierarchical routing
- Internet routing protocols
  - (RIP)
  - OSPF
  - BGP
- **Business considerations**
  - **Policy routing**
  - Traffic engineering

Business relationships

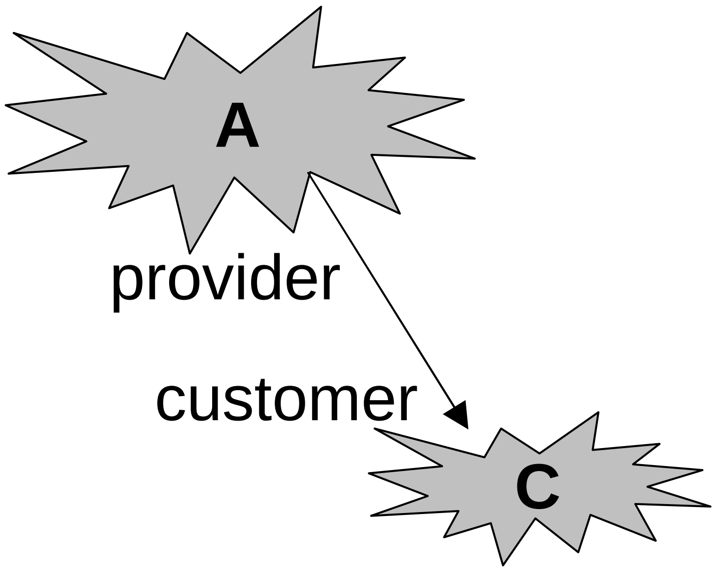
- Internet = network of networks (ASes)
  - Many thousands of ASes
  - Not every network connected to every other network
  - BGP used for routing between ASes
- Differences in economical power/importance
  - Some ASes huge, intercontinental (AT&T, Cable&Wireless)
  - Some ASes small, local (e.g., München: M”Net, SpaceNet)
- Small ASes customers of larger ASes: Transit traffic
  - Smaller AS pays for connecting link + for data = buys transit
  - Business relationship = customer—provider
- Equal-size/-importance ASes
  - Usually share cost for connecting link[s]
  - Business relationship = peering (no transit traffic)
- Warning: peering (“equal-size” AS) ≠ peers of a BGP connection (also may be customer or provider) ≠ peer-to-peer network

Business and policy routing (1)

- Basic principle #1
  - Prefer routes that incur financial gain
- Basic principle #2
  - Announce routes that incur financial gain if others use them
    - Others = customers
  - Announce routes that reduce costs if others use them
    - Others = peers
  - Do not announce routes that incur financial loss
    (…as long as alternative paths exist)

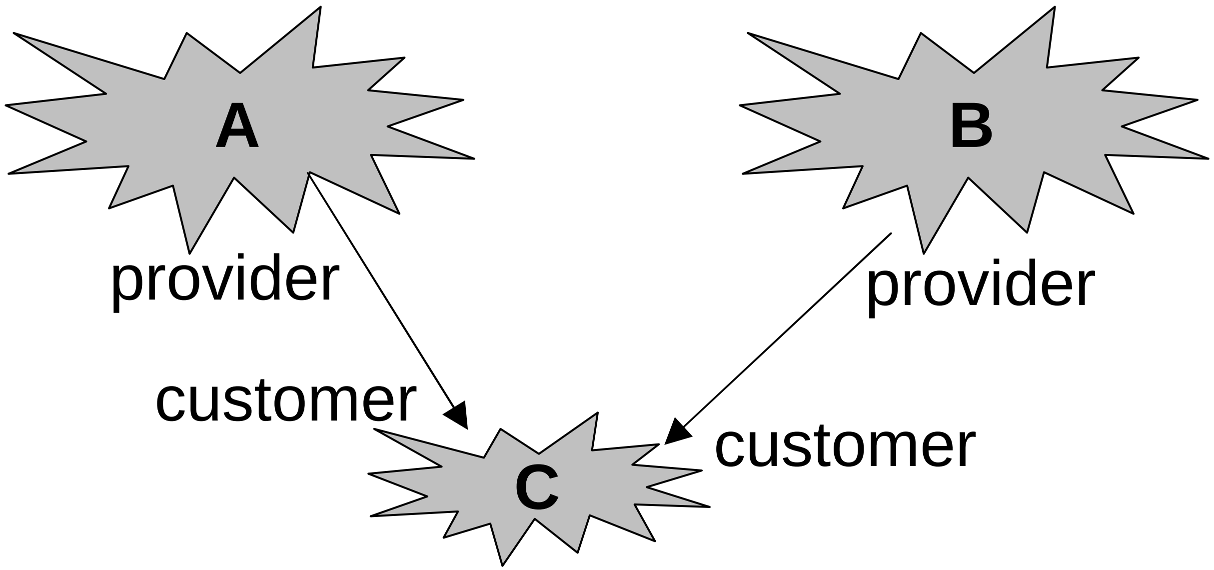
Business and policy routing (2)

- A tells C all routes it uses to reach other ASes
  - The more traffic comes from C, the more money A makes



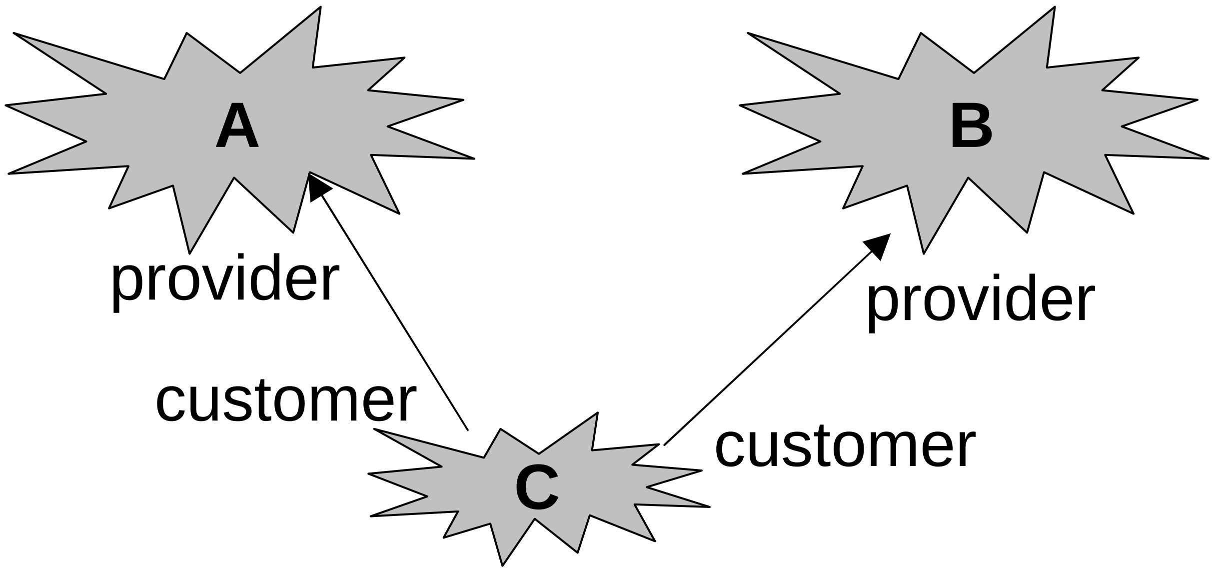
Business and policy routing (3)

- A and B tell C all routes they use to reach other ASes
  - The more traffic flows from C to A, the more money A makes
  - The more traffic flows from C to B, the more money B makes



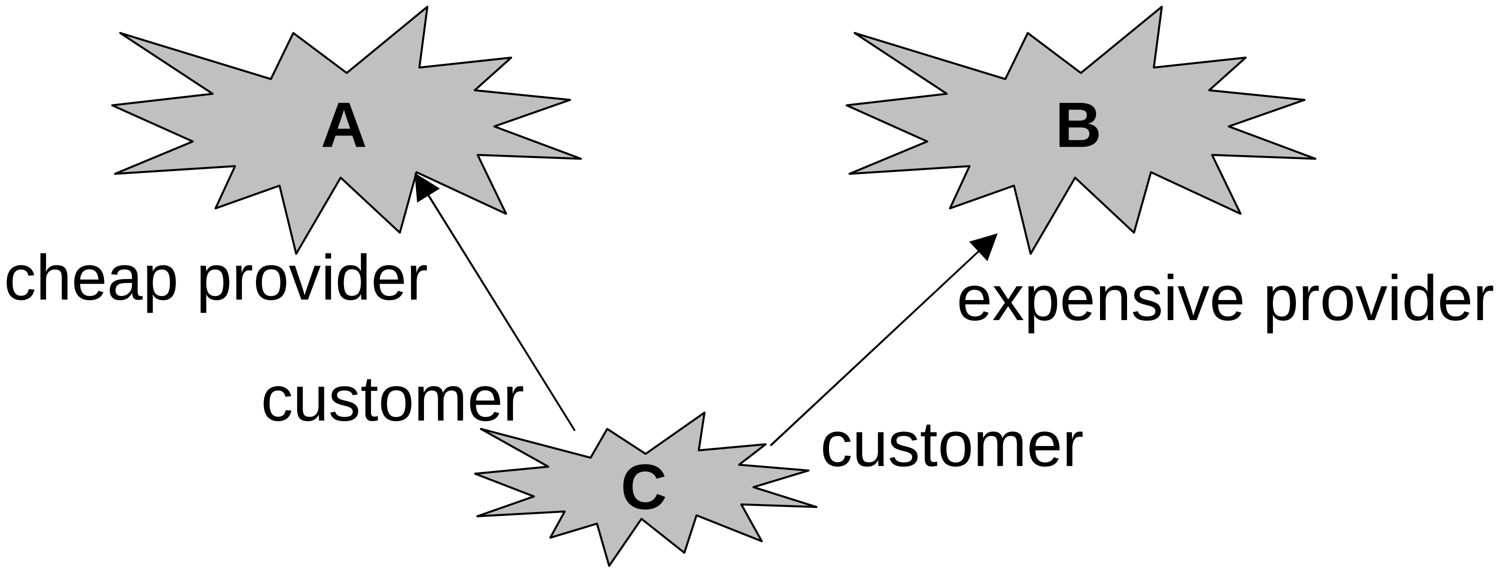
Business and policy routing (4)

- C tells A its own prefixes; C tells B its own prefixes
  - C wants to be reachable from outside
- C does not tell A routes learned from/via B
  C does not tell B routes learned from/via A
  - C does not want to pay money for traffic …↔A ↔C ↔B ↔…



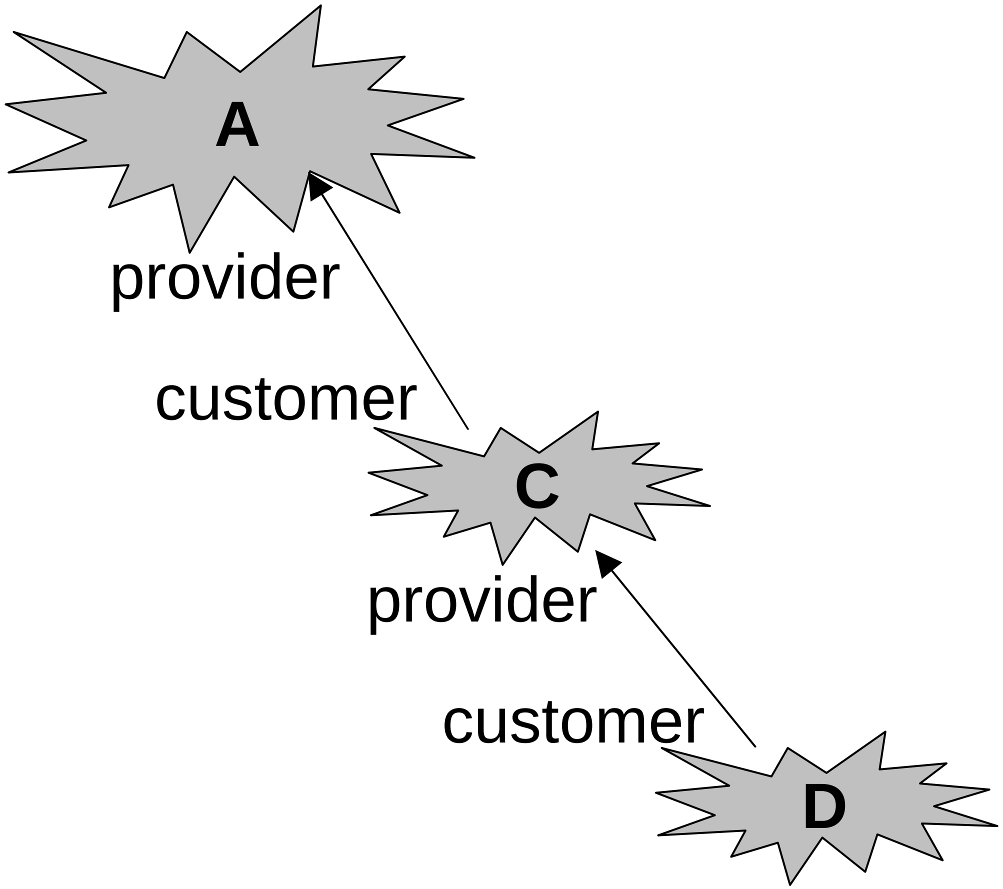
Business and policy routing (5): AS path prepending

- C tells A its own prefixes
- C may tell B its own prefixes
  - …but inserts “C” multiple times into AS path
  - Result: Route available, but longer path = less attractive
  - Technique is called *AS path prepending*



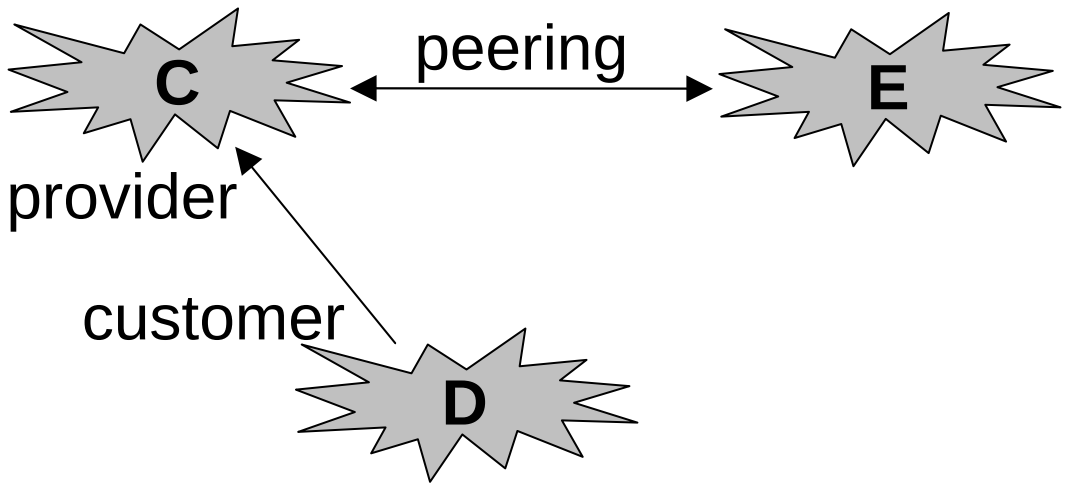
Business and policy routing (6)

- C tells A about its own prefixes
- C tells A about its route to D’s prefixes:
  loses money to A, but gains money from D



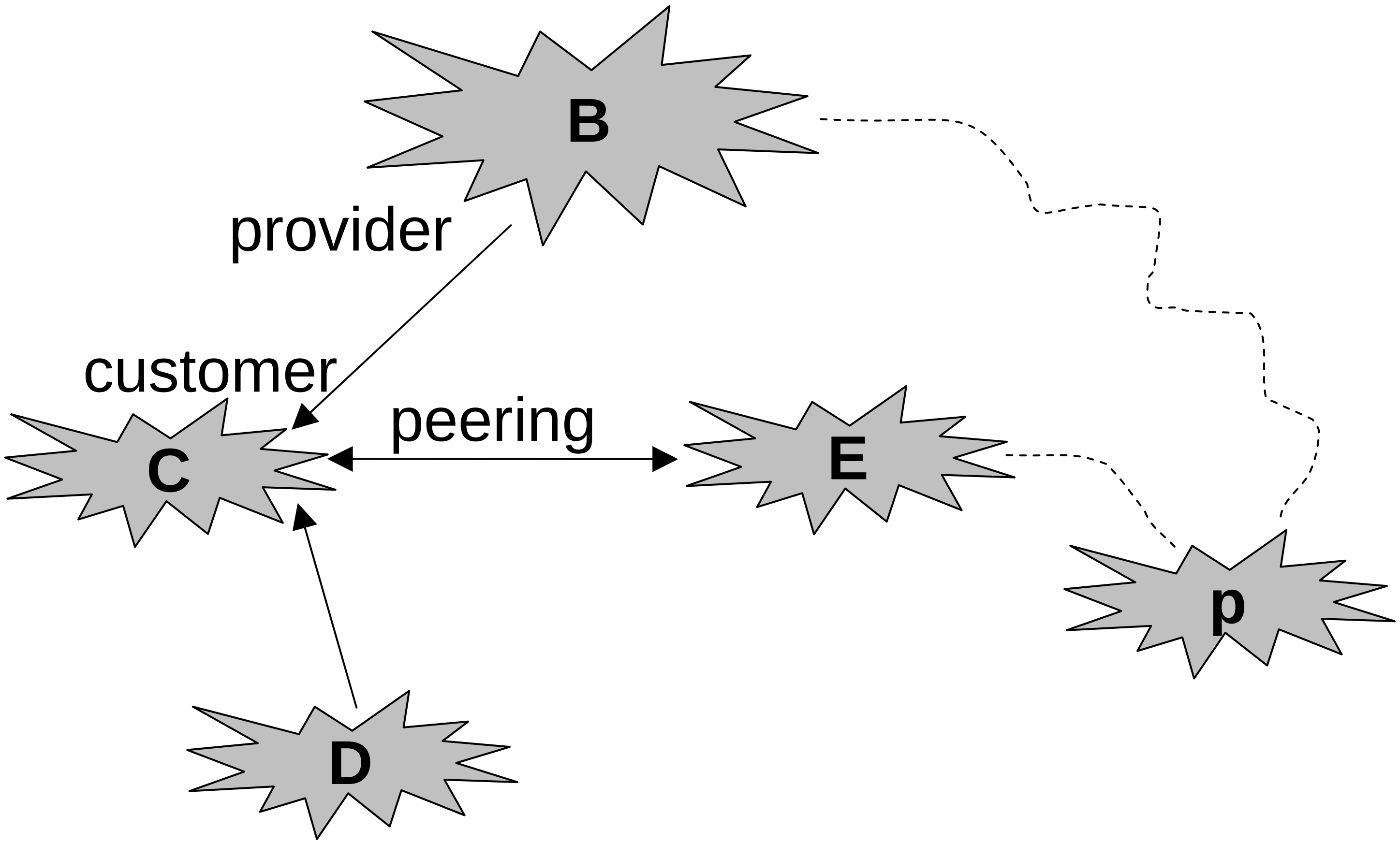
Business and policy routing (7)

- C tells peering partner E about its own prefixes and route to D: no cost on link to E, but gains money from D



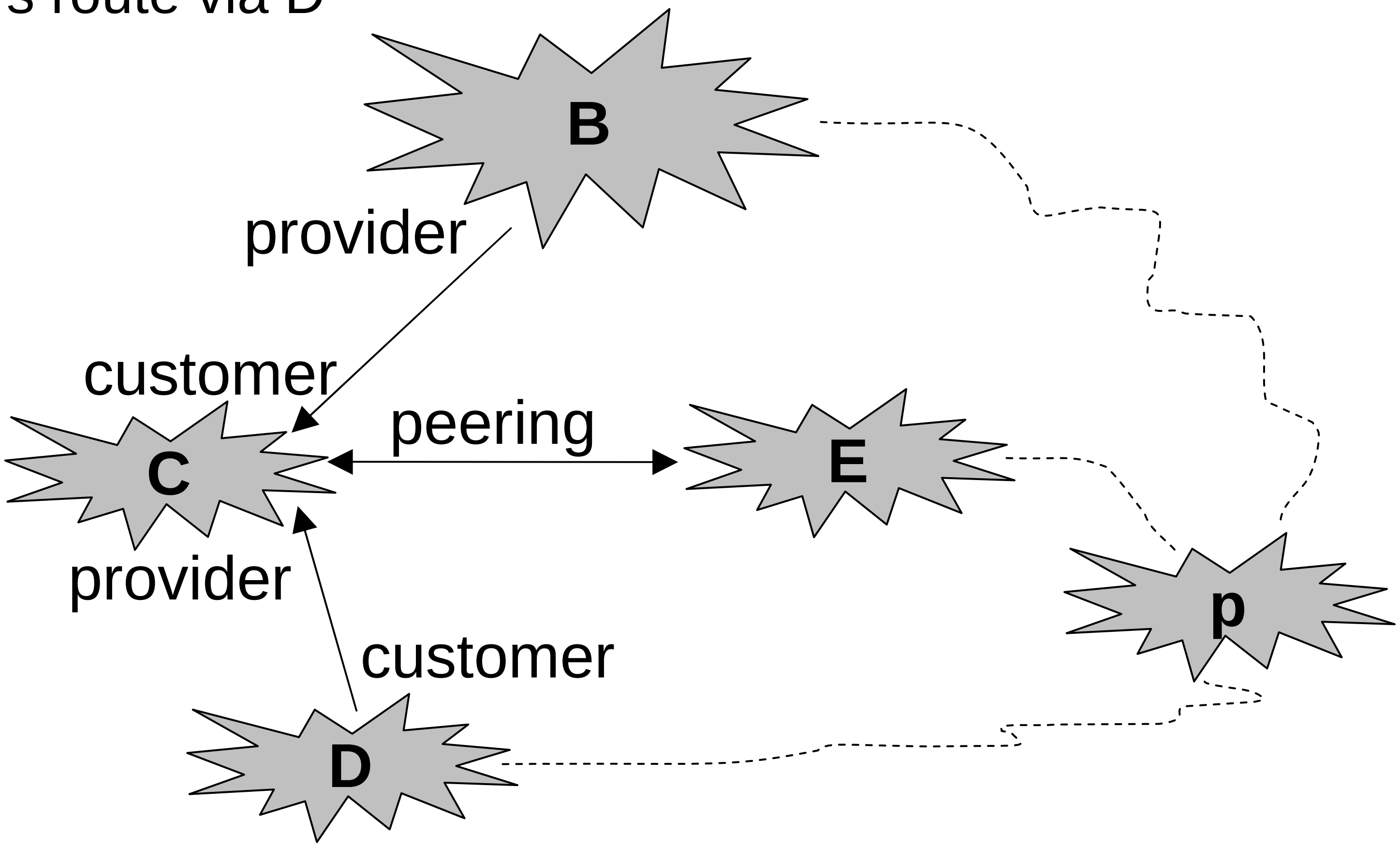
Business and policy routing (8)

- B tells C about route to prefix p (lose money)
- E tells C about route to prefix p (± 0)
- C prefers route via E



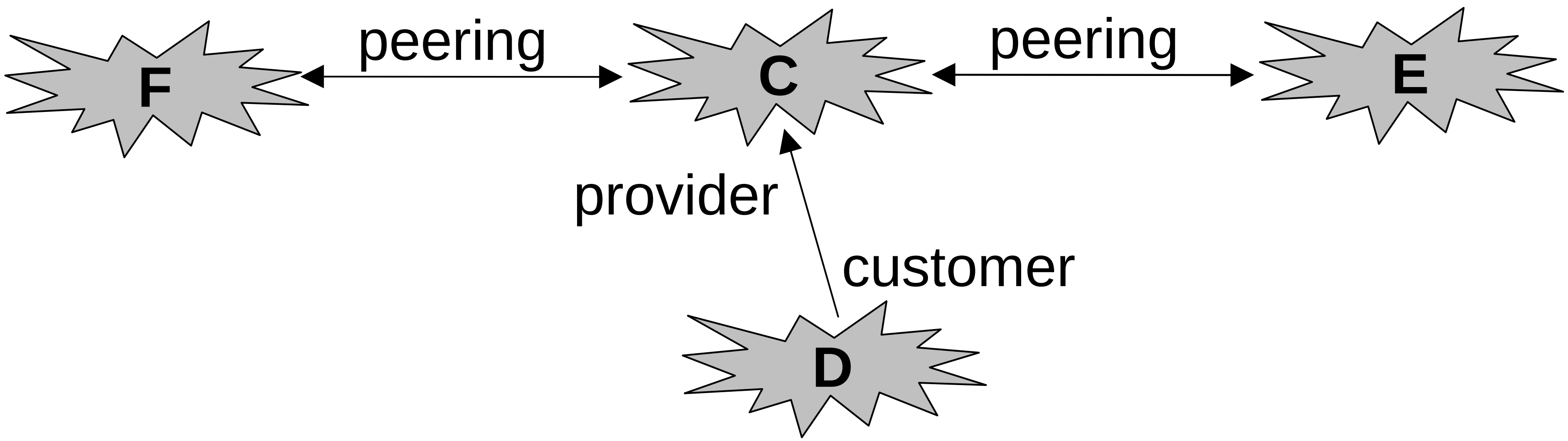
Business and policy routing (8)

- B tells C about route to prefix p (lose money)
- E tells C about route to prefix p (± 0)
- D tells C about route to prefix p (gain money)
- C prefers route via D



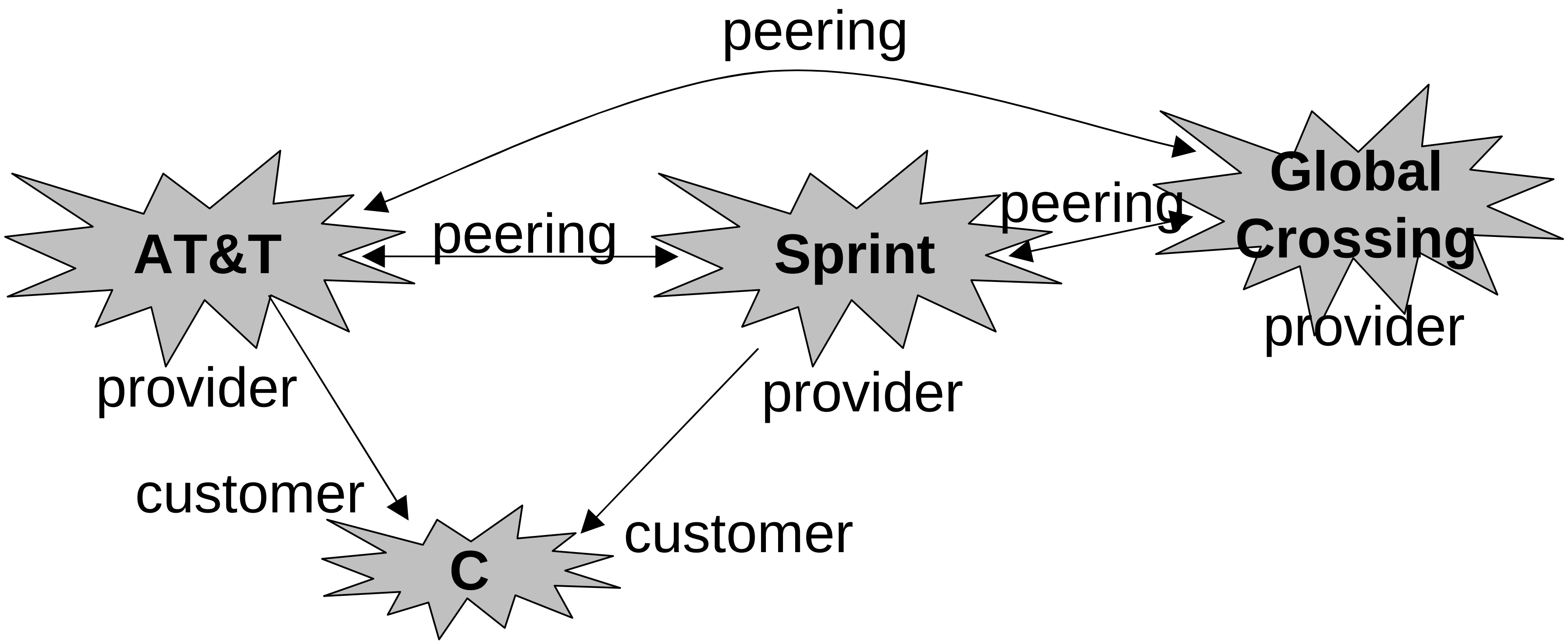
Business and policy routing (9)

- C announces to F and E: its own prefixes and D's routes
- C does *not* announce to E: routes going via F
  - Otherwise: E could send traffic towards F but wouldn't pay anything, F wouldn't pay either, and C's network gets loaded with additional traffic
- C does *not* announce to F: routes going via E
  - Same reason



Business and policy routing (10): "Tiers" / "DFZ"

- Big players have no providers, only customers and peers
  - "Tier-1" providers
  - or "Default-Free Zone" (have no default route to "provider")
- Each Tier-1 peers with each other



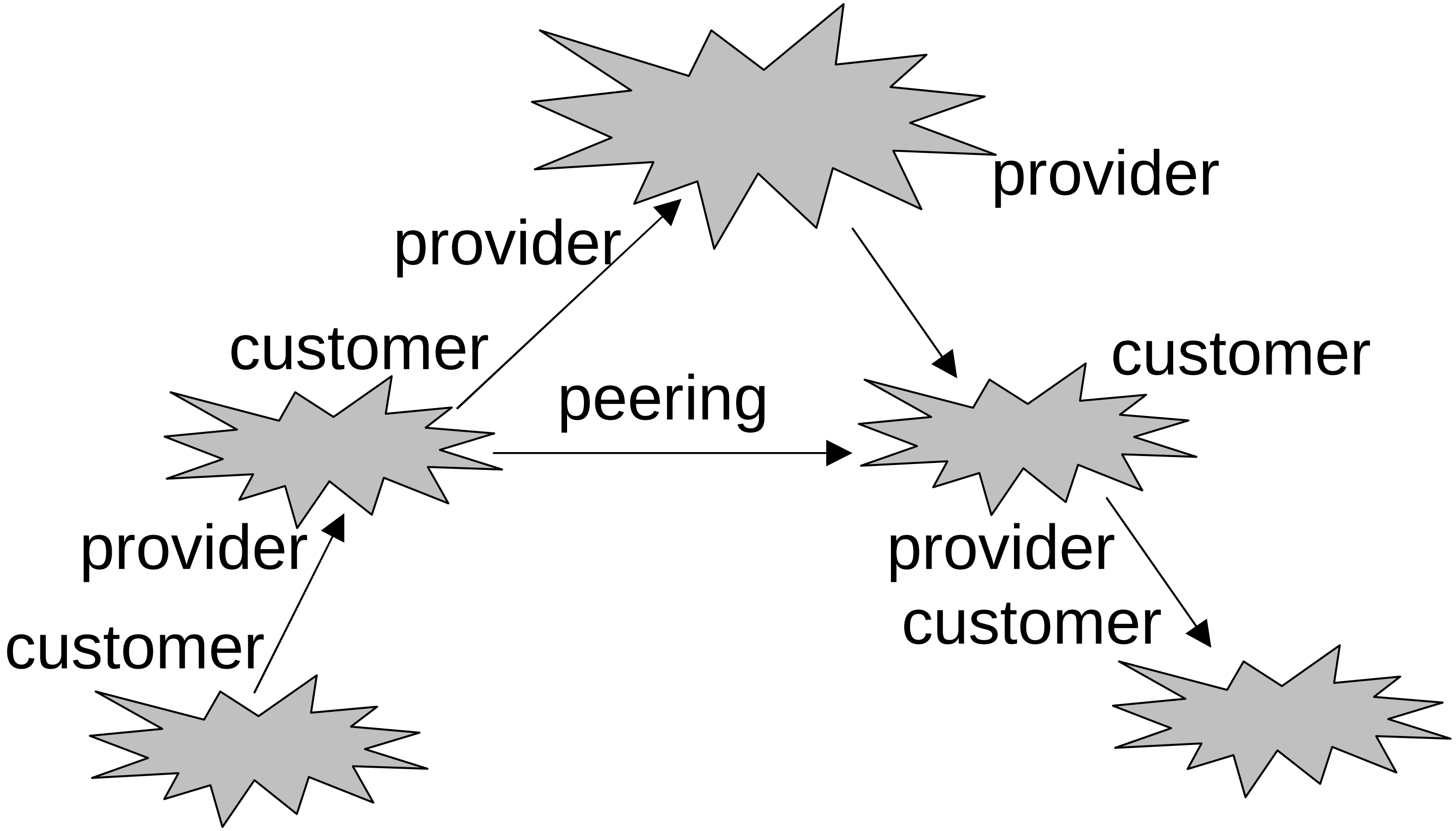
Tier-1, Tier-2, Tier-3 etc.

- Tier-1/DFZ = only peerings, no providers
- Tier-2 = only peerings and Tier-1 providers
- Tier-3 = at least one Tier-2 as a provider
- Tier-*n*: defined recursively
  - $n \geq 4$: Rare in Western Europe, North America, East Asia
- "Tier-1.5" = almost a Tier-1 but pays money for some links
  - Example: Deutsche Telekom pays money to Sprint, but peers with other Tier-1 providers
  - Marketing purposes: Tier-1 sounds better

Valley-free routing

Results: Packets always travel…

1. upstream: sequence of C→P links (possibly length = 0)
2. then possibly across *one* peering link
3. then downstream: sequence of P→C links (possibly length = 0)



Siblings

- Not everything is provider/customer or peering
- Sibling = mutual transit agreement
  - Provide connectivity to the rest of the Internet for each other
  - ≈ very extensive peering
- Examples
  - Two small ASes close to each other that cannot afford additional Internet services
  - Merging two companies
    - Merging two ASes into one = difficult,
    - Keeping two ASes and exchaning everything for free = easier

To peer or not to peer, this is the question

Peer:

- Reduce upstream costs
- Possibly increases performance
- Perhaps only way to connect your customers (Tier-1)

Don't peer

- You don't gain any money
- Peers are usually your competitors
- What if it turns out the peering is more beneficial to you peer than to you? ⇒ Require periodic regenotiation

Where to peer

- Private peering
- At public peering locations (IX, Internet Exchange Point)
  - “A house full of routers that many providers connect to”
  - E.g., DE-CIX, AMS-IX, LINX

BGP/Policy routing Summary

- Import Policy = Which routes to use
  - Select path that incurs most money
  - Special/political considerations (e.g., Iranian AS does not want traffic to pass Israeli AS; other kinds of censorship)
- Export Policy = Which routes to propagate to other ASes
  - Not all possible routes propagate:
Export only…
    - If it incurs revenue
    - If it reduces cost
    - If it is inevitable
  - Propagation driven by business considerations
  - Propagation not driven by technical considerations!
Example: Slower route via peer may be preferred over faster route via provider

BGP policy routing: Technical summary

1. Receive BGP update
2. Apply import policies
   - Filter routes
   - Tweak attributes (advanced topic…)
3. Best route selection based on attribute values
   - Install forwarding tables entries for best routes
   - Possibly transfer to Route Reflector
4. Apply export policies
   - Filter routes
   - Tweak attributes
5. Transmit BGP updates

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

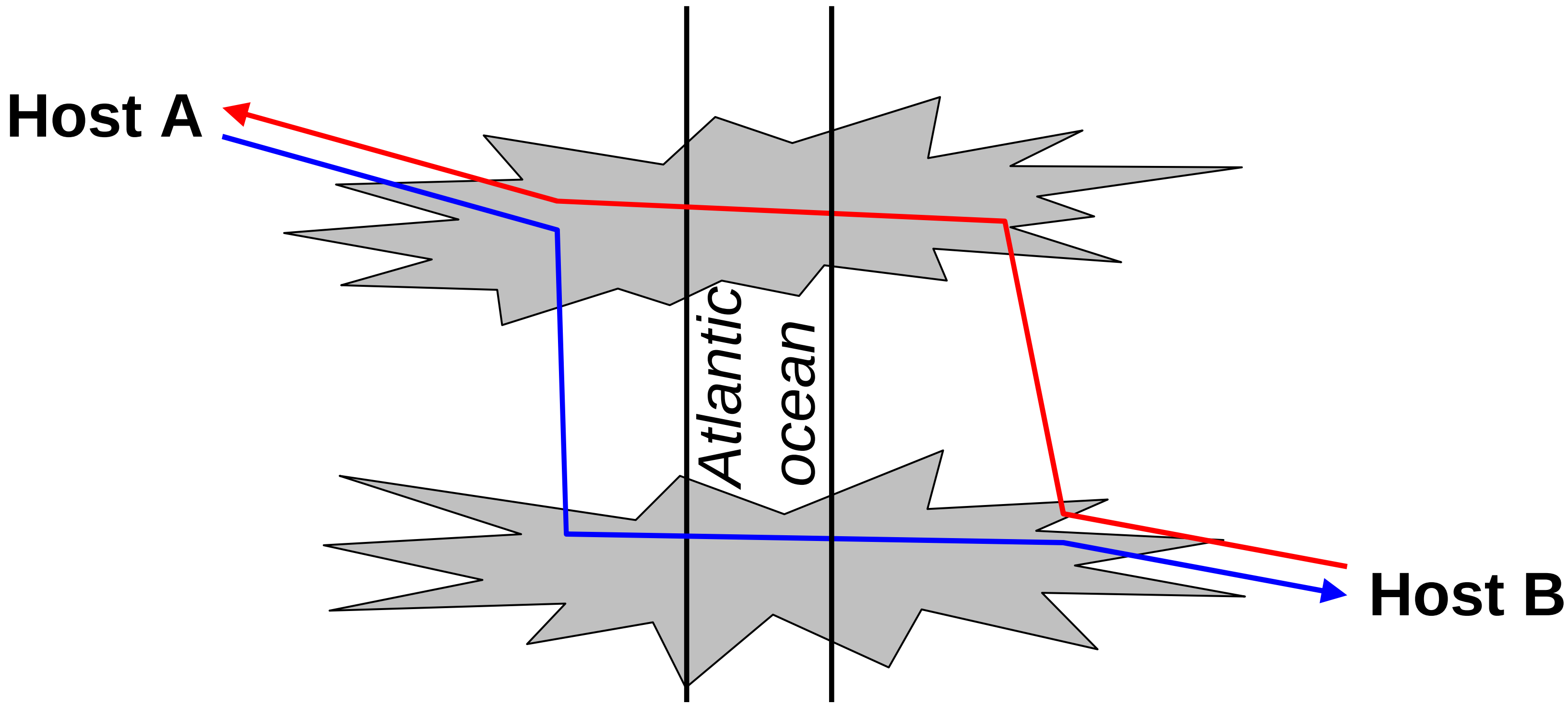
- IPv6
- NAT
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Path Vector
  - Hierarchical routing
- Internet routing protocols
  - (RIP)
  - OSPF
  - BGP
- **Business considerations**
  - Policy routing
  - **Hot-potato routing**
  - Traffic engineering

Hot-potato routing

- Interaction between Inter-AS and Intra-AS routing
  - Business: If traffic is destined for other AS, get rid of it ASAP
  - Technical: Intra-AS routing finds shortest path to gateway
- Multiple transit points ⇒asymmetrical routing



Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- NAT
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Path Vector
  - Hierarchical routing
- Internet routing protocols
  - (RIP)
  - OSPF
  - BGP
- **Business considerations**
  - Policy routing
  - **Traffic engineering**

Routing: Optimization purposes

- Inter-AS routing
  - Optimality = select route with highest revenue/least loss
- Intra-AS routing
  - Optimality = configure routing such that network can host as much traffic as possible

Traffic Engineering

1. Collect traffic statistics: Traffic Matrix
   - How much traffic flowing from A to B?
   - Difficult to measure! (drains router performance); thus often estimated: research area
2. Optimize routing
   - E.g., calculate good choice of OSPF weights
   - Goal: minimize maximum link load in entire network; keep average link load below 50%
     - why? Fractal TCP traffic leads to spikes!
3. Deploy new routing
   - Performance may deteriorate during update
   - E.g., routing loops during OSPF convergence

Dynamic traffic engineering

Why not dynamic?

- Routing loops during convergence
- Packet reordering:
  - Packet P1 arrives later than Packet P2
  - TCP will think that P1 got lost! ⇒ congestion control!
- Thus: Congestion control in end hosts, not in network

Note on exercise sheet

- Measurement exercise sheet ≠ 2nd project (on measurement)
- BUT:
  It gives some theoretical foundations for measurement project

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- NAT
- Virtual circuit and datagram networks
- What's inside a router

Part 3

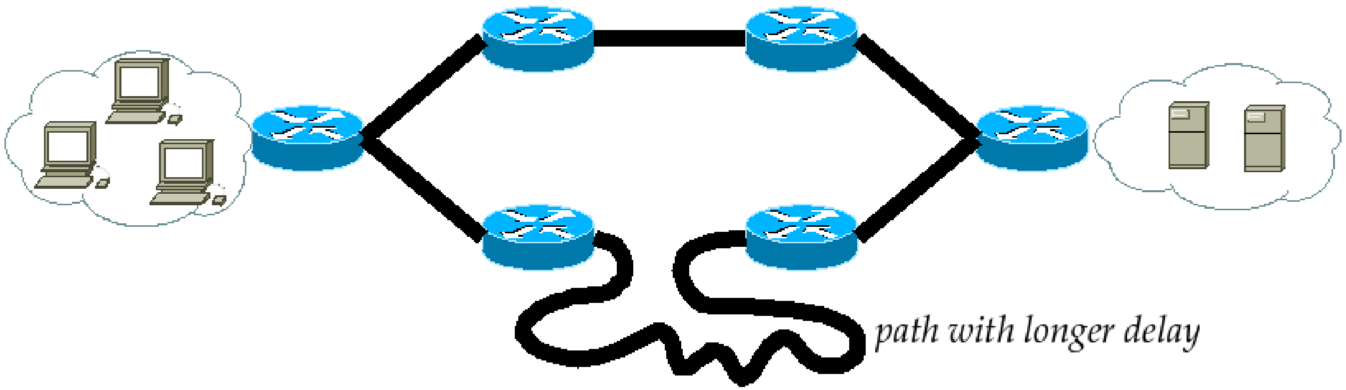
- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - OSPF
  - BGP
  - Business considerations
  - **Multipath and TCP**
- Multicast routing
- *NAT (different slide set)*
- Weaknesses and shortcomings

Multipath routing

- Routing = finding best-cost route
- What if more than one exists?
- Some routing protocols allow Equal-Cost Multipath (ECMP) routing, e.g., OSPF
  - ≥ 2 routes of same cost exist to destination prefix?
    → Evenly distribute traffic across these routes

Multipath routing: TCP problem

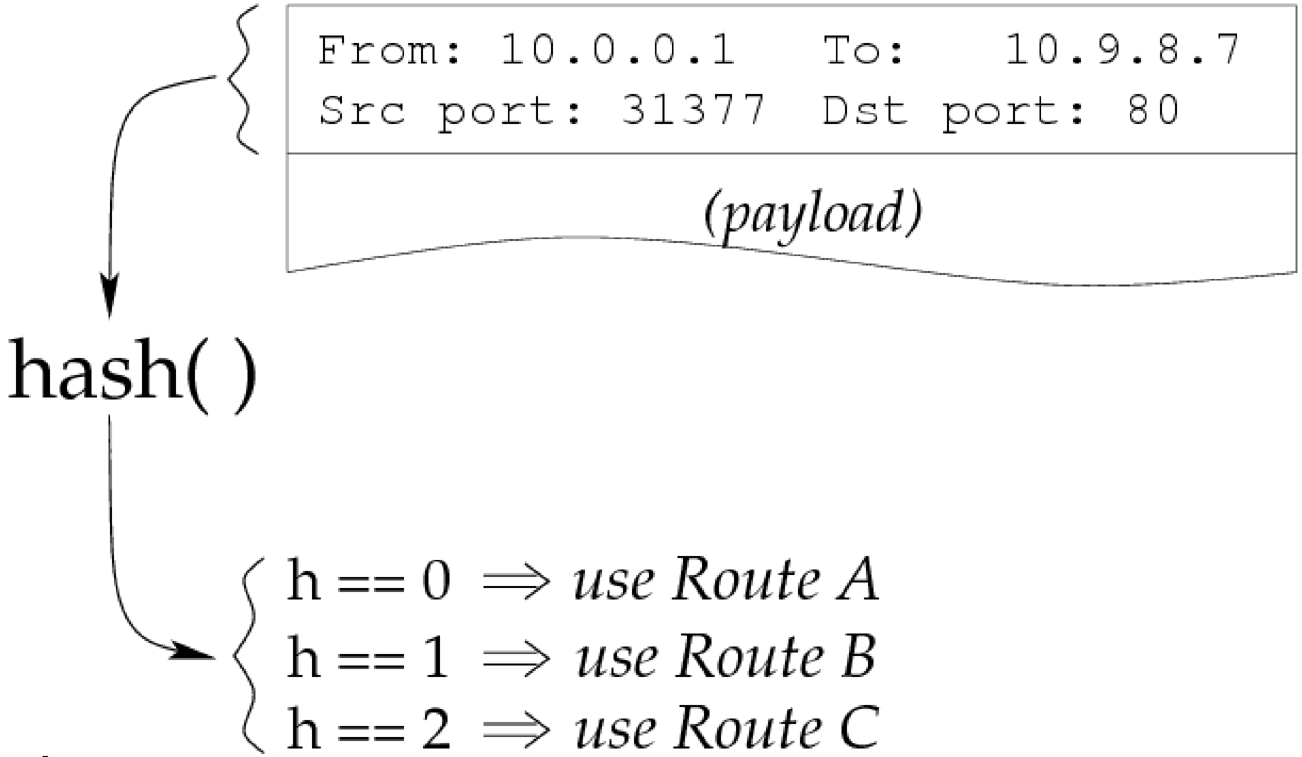
- How to distribute traffic? Naïve approaches:
  - Round-robin
  - Distribute randomly
- Equal cost does not mean equal latency:



- Again: Problem with TCP = Packet reordering!
  - Packets sent: P1, P2
  - Packets received: P2, P1
  - Receiver receives P2 → believes P1 to be lost → triggers congestion control mechanisms → performance degrades

Multipath routing: Solution

- Hash “randomly”…
- …but use packet headers as “random” values:



- Result:
  - Packets from same TCP connection yield same hash value
  - No reordering possible

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

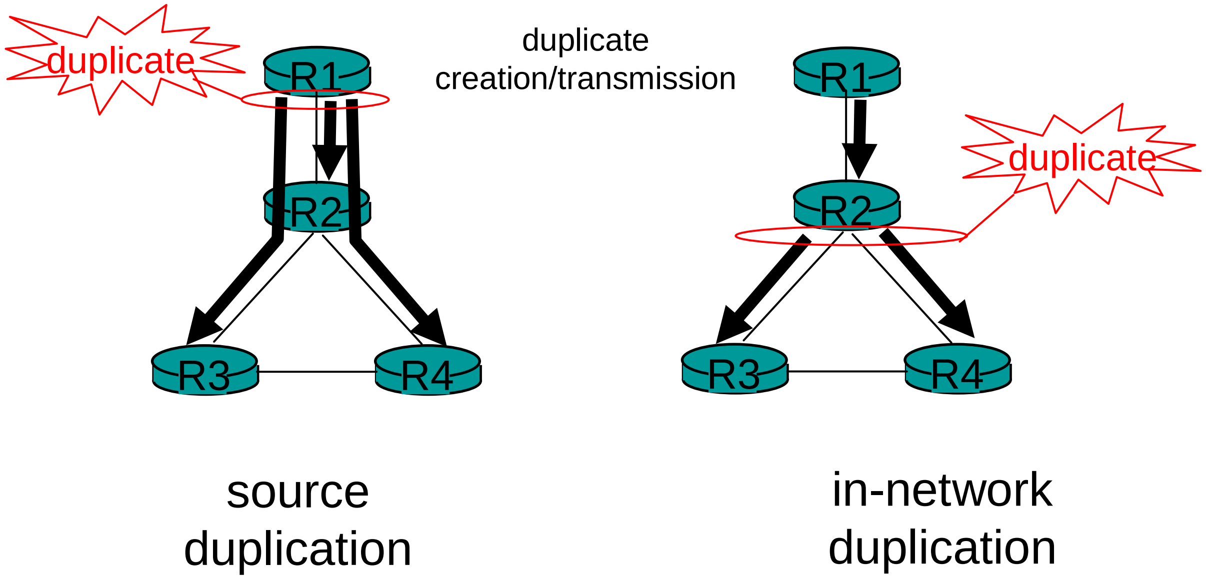
- IPv6
- NAT
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - OSPF
  - BGP
  - Business considerations
  - Multipath routing and TCP
- **Multicast routing**
- *NAT (different slide set)*
- Weaknesses and shortcomings

Broadcast Routing

- Deliver packets from source to all other nodes
- Source duplication is inefficient:



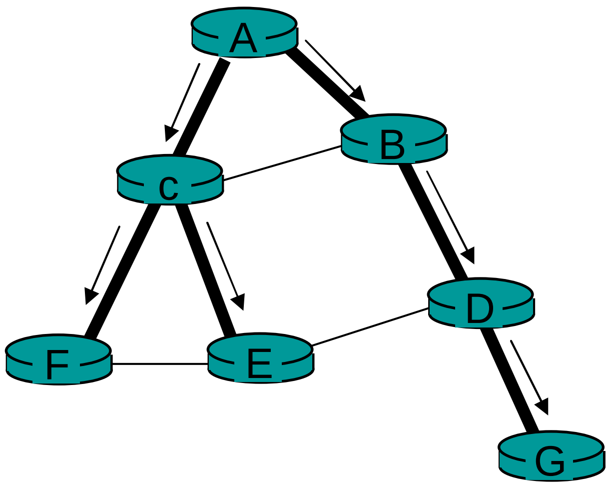
- Source duplication: how does source determine recipient addresses?

In-network duplication

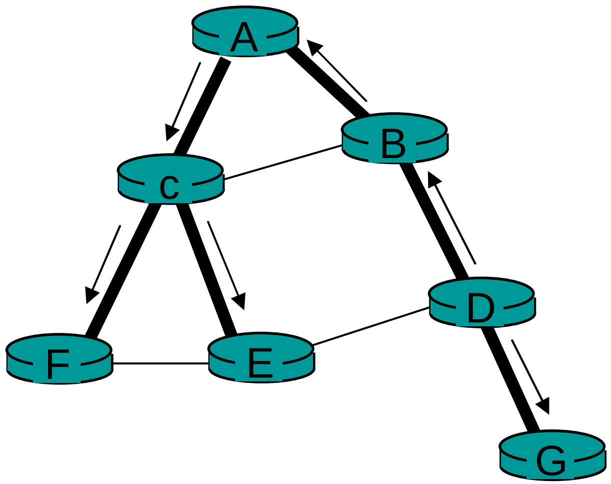
- Flooding: when node receives broadcast packet, sends copy to all neighbours
  - Problems: cycles & broadcast storm
- Controlled flooding: node only broadcasts packet if it hasn’t broadcast same packet before
  - Node keeps track of packet IDs already broadcasted
    (need memory! expensive!)
  - Or reverse path forwarding (RPF): only forward packet if it arrived on shortest path between node and source
- Spanning tree
  - No redundant packets received by any node

Spanning Tree

- First construct a spanning tree
  - cf. algorithms + data structures lecture / textbook
- Nodes forward copies only along spanning tree



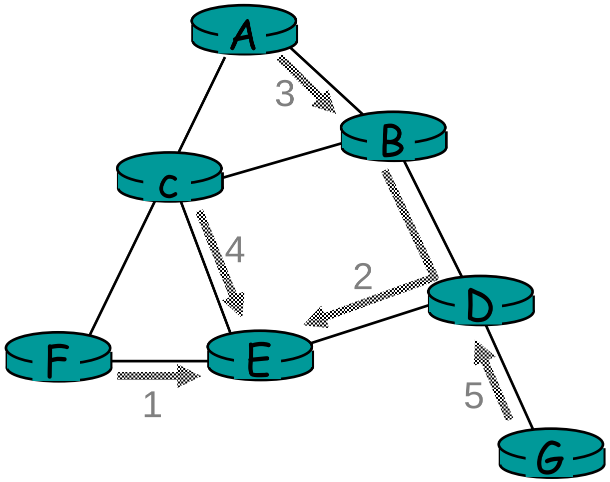
(a) Broadcast initiated at A



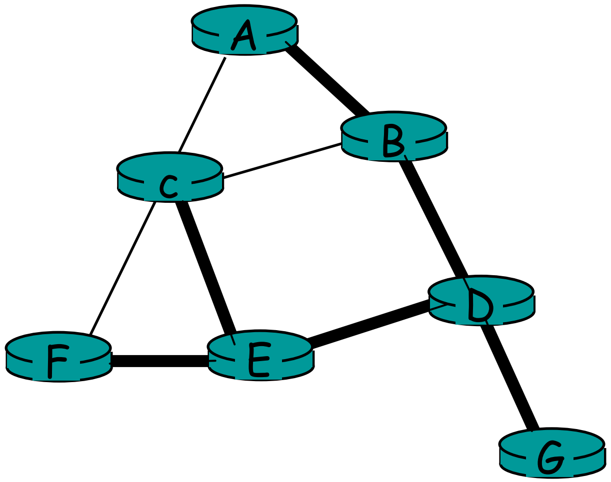
(b) Broadcast initiated at D

Spanning Tree: Creation

- Center node
- Each node sends unicast join message to center node
  - Message forwarded until it arrives at a node already belonging to spanning tree



(a) Stepwise construction of spanning tree



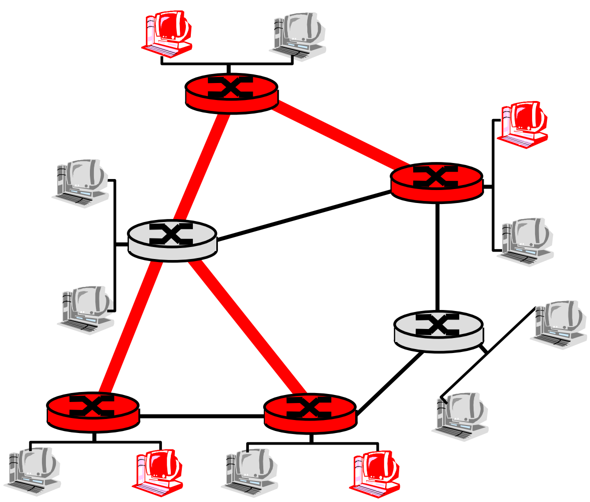
(b) Constructed spanning tree

Multicast groups

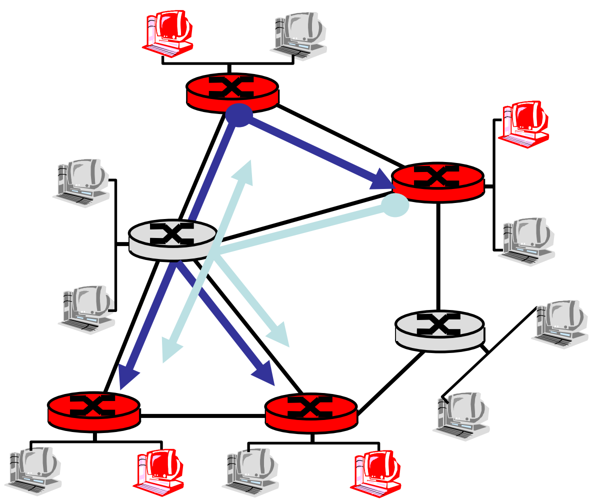
- One single shortest-path graph for all kind of unicast traffic
  - (let's not consider QoS routing here…)
- One single tree for multicast traffic?
  - Consider TV:
    - German TV in D-A-CH area
    - Korean TV in Korea
  - Why multicast German TV in entire Korean IP network and vice versa?
- Multicast group:
  - Different multicast routing trees for different resources
  - Internet: 1 Multicast group = 1 "special" IP address
    - Special = from specific Multicast prefix
    - IP address does not specify one host, but entire group: multiple receivers; multiple senders

Multicast Routing: Problem Statement

- ***Goal:*** find a tree (or trees) connecting routers having local multicast group members
  - *tree:* not all paths between routers used
  - *source-based:* different tree from each sender to rcvrs
  - *shared-tree:* same tree used by all group members



Shared tree



Source-based trees

Approaches for building multicast trees

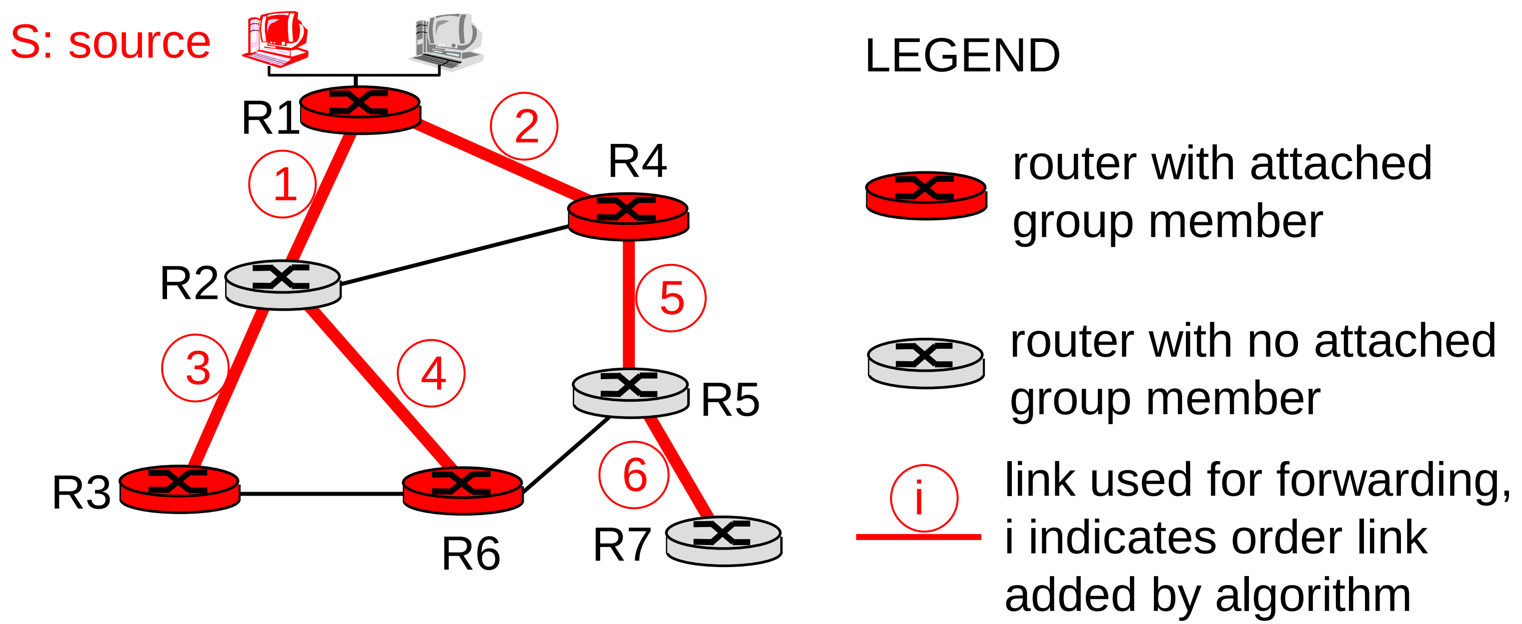
Approaches:

- source-based tree: one tree per source
  - shortest path trees
  - reverse path forwarding
- group-shared tree: group uses one tree
  - minimal spanning (Steiner)
  - center-based trees

…we first look at basic approaches, then specific protocols adopting these approaches

Shortest Path Tree

- Multicast forwarding tree: tree of shortest path routes from source to all receivers
  - Dijkstra's algorithm



Reverse Path Forwarding

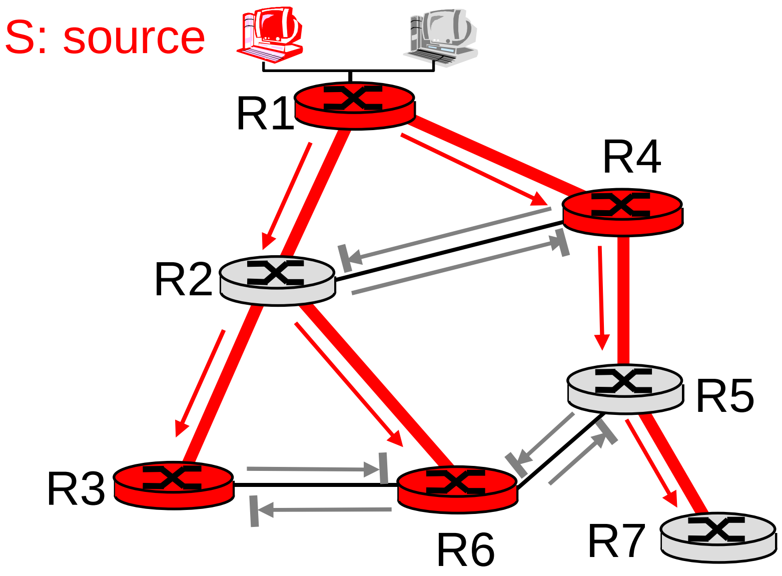
- Rely on router's knowledge of unicast shortest path from it to sender
- Each router has simple forwarding behavior:

if (mcast datagram received on incoming link on shortest path back to center)

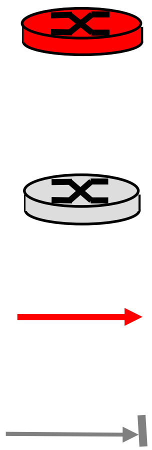
then flood datagram onto all outgoing links

else ignore datagram

Reverse Path Forwarding: example



LEGEND



router with attached group member

router with no attached group member

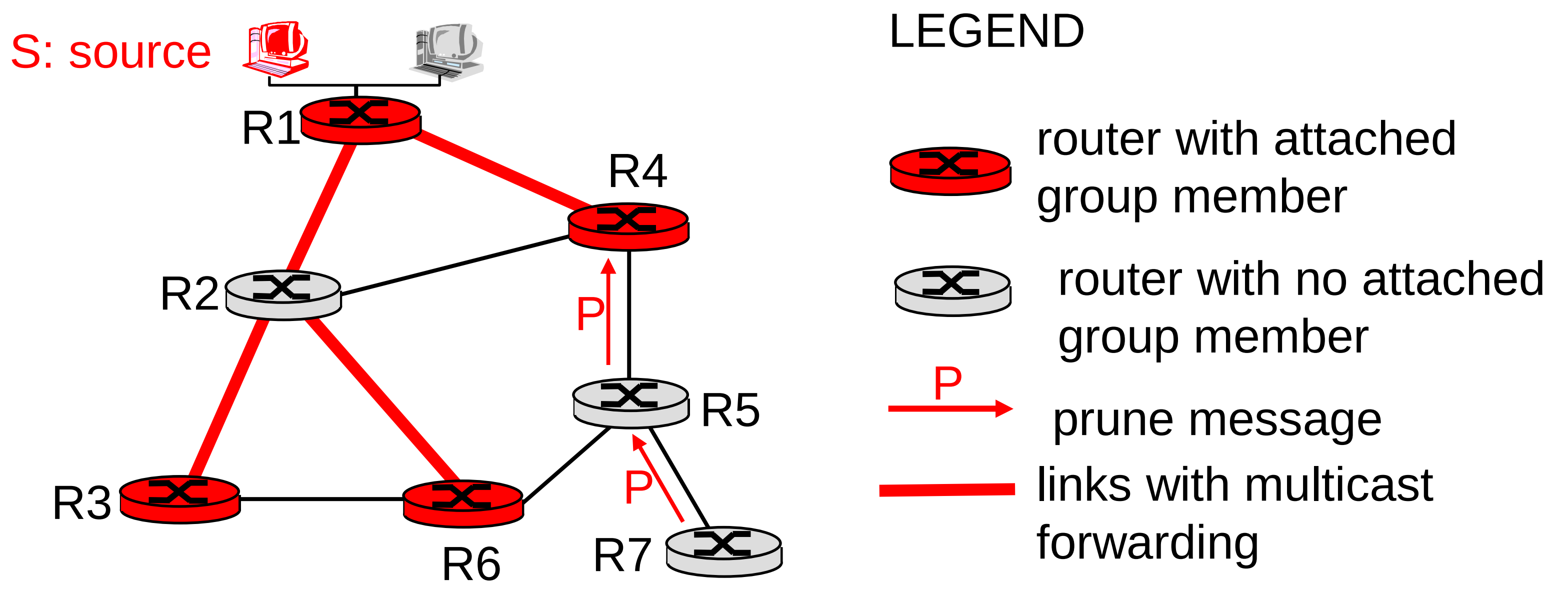
datagram will be forwarded

datagram will not be forwarded

- result is a source-specific *reverse* SPT
  - may be a bad choice with asymmetric links

Reverse Path Forwarding: pruning

- Forwarding tree contains subtrees with no multicast group members
  - no need to forward datagrams down subtree
  - "prune" msgs sent upstream by router with no downstream group members



Shared-Tree: Steiner Tree

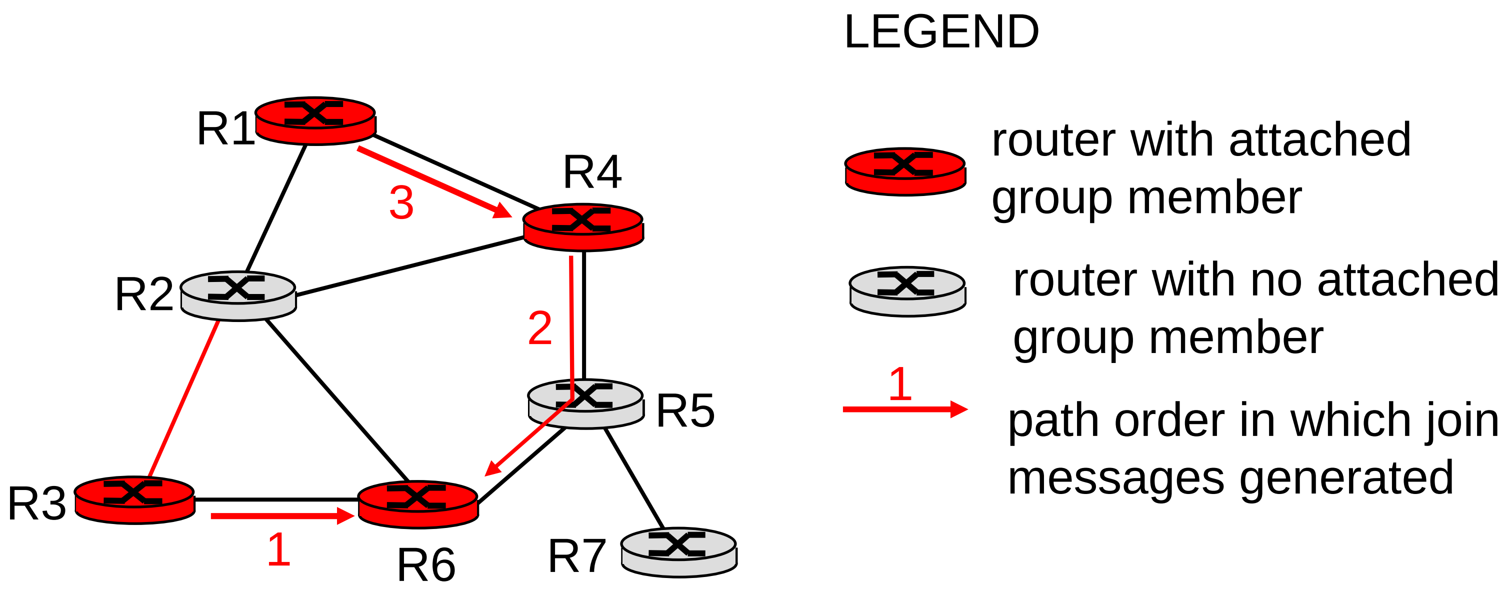
- Steiner Tree: minimum cost tree connecting all routers with attached group members
- Problem is NP-complete
- Excellent heuristics exists; active research area in theoretical computer science
- But not used in practice:
  - Computational complexity
  - Information about entire network needed
  - Monolithic: rerun whenever a router needs to join/leave

Center-based trees

- Single delivery tree shared by all members
- One router identified as *"center"* of tree
- To join:
  - edge router sends unicast *join-msg* addressed to center router
  - *join-msg* "processed" by intermediate routers and forwarded towards center
  - *join-msg* either hits existing tree branch for this center, or arrives at center
  - path taken by *join-msg* becomes new branch of tree for this router

Center-based trees: an example

Suppose R6 chosen as center:



Multicast routing in the Internet (1)

- Multicast routing protocols
  - DVMRP: distance-vector multicast routing protocol
  - MOSPF: Multipath OSPF
  - PIM: Protocol Independent Multicast
- But the end hosts!?
  - End hosts send/receive Multicast traffic,…
  - …but do not run routing protocols!
- IGMP (Internet Group Management Protocol): IPv4
  - Host can join/leave multicast group
  - Sits on top of IPv4
- MLD (Multicast Listener Discovery): IPv6
  - Router discovers multicast listeners
  - Embedded into IPv6

Multicast routing in the Internet (2)

- Who uses it? Almost nobody!

No application makes use of it

Not available to "normal" end users

Nobody asks for it

Providers normally don't offer it

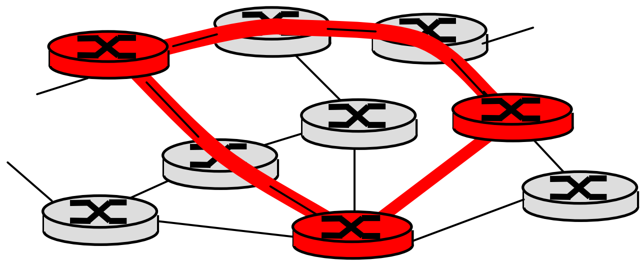
Accounting quarrels: who pays for what?

Technical issues: need rapid+efficient join/leave across AS boundaries…

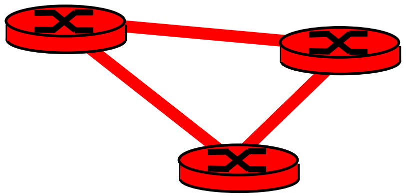
- Available as special solution in some isolated contexts
  - Triple Play: Internet + telephone (VoIP) + TV over IP
  - Software updates in large companies

Tunneling

Q: How to connect "islands" of multicast routers in a "sea" of unicast routers?



physical topology



logical topology

- Multicast datagram encapsulated inside "normal" (non-multicast-addressed) datagram
- Normal IP datagram sent through "tunnel" via regular IP unicast to receiving multicast router
- Receiving mcast router unencapsulates to get multicast datagram

PIM: Protocol Independent Multicast

- Not dependent on any specific underlying unicast routing algorithm (works with all)
- Two different multicast distribution scenarios :

Dense:

- group members densely packed, in "close" proximity
- bandwidth more plentiful

Sparse:

- #of networks with group members small with respect to # of interconnected networks
- group members "widely dispersed"
- bandwidth not plentiful

Consequences of Sparse/Dense Dichotomy:

Dense

- Group membership by routers *assumed* until routers explicitly prune
- *Data-driven* construction on multicast tree
- Bandwidth and non-group-router processing may *waste* resources

Sparse:

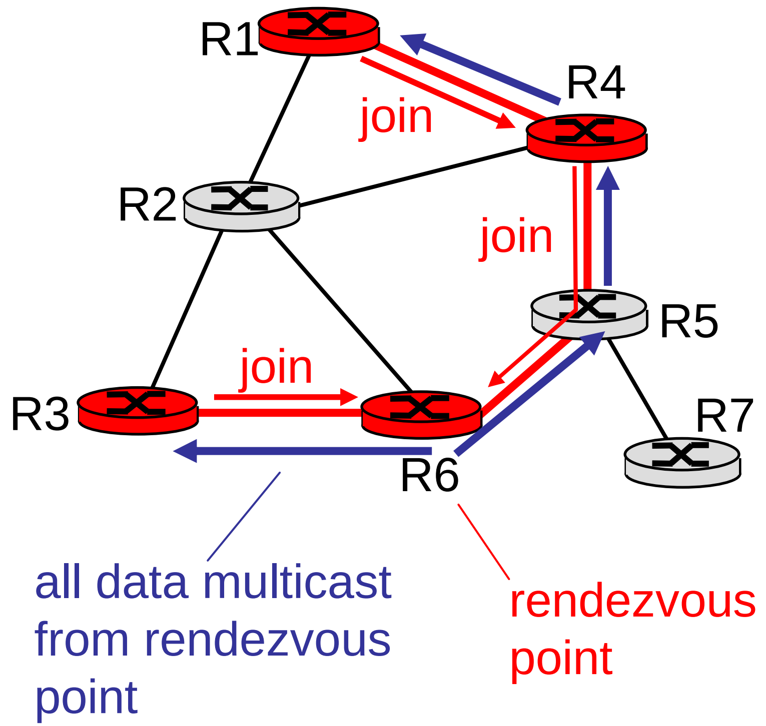
- No membership until routers explicitly join
- *Receiver- driven* construction of multicast tree (e.g., center-based)
- Bandwidth and non-group-router processing *conservative*

PIM: Dense mode

- "Flood and prune":
  Use Reverse Path Forwarding to flood information across network
- Uninterested leaf routers *may* prune network
  - Mechanism to detect if leaf router

PIM: Sparse Mode

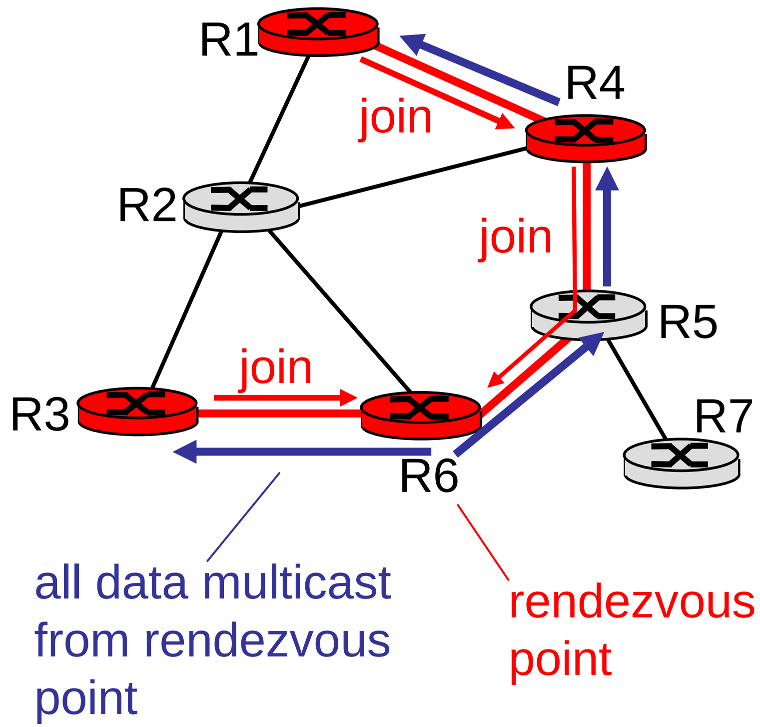
- Center-based approach
- Router sends *join* msg to rendezvous point (RP)
  - intermediate routers update state and forward *join*
- After joining via RP, router can switch to source-specific tree
  - increased performance: less concentration, shorter paths



PIM: Sparse Mode

sender(s):

- unicast data to Rendezvous Point (!)
- RP then distributes down RP-rooted tree
- RP can extend multicast tree upstream to source
- RP can send *stop* msg if no attached receivers
  - "no one is listening!"



Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- NAT
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - OSPF
  - BGP
  - Business considerations
  - Multipath routing and TCP
- Multicast routing
- ***NAT (different slide set)***
- Weaknesses and shortcomings

Chapter 4: Network Layer

Part 1

- Introduction
- IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP

Part 2

- IPv6
- NAT
- Virtual circuit and datagram networks
- What's inside a router

Part 3

- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - OSPF
  - BGP
  - Business considerations
  - Multipath routing and TCP
- Multicast routing
- *NAT (different slide set)*
- **Weaknesses and shortcomings**

Network Layer: Weaknesses and shortcomings (1)

- Violation of layering principles
  - NAT: share IP address (L3) based on TCP/UDP ports (L4)
  - Firewalls: router (L3) blocks traffic based on TCP/UDP (L4)
  - Intelligent firewalls, transparent proxies: IP traffic (L3) intercepted and mangled depending on application! (L5–7)
  - Multi-path routing (L3): take care of TCP (L4)
  - Dynamic routing (L3): complicated due to TCP (L4)
  - Layer 1, Layer 2 must not drop too many packets (e.g., wireless, satellite, etc.):
    TCP (L4) believes losses to be caused by congestion

Network layer: Weaknesses and Shortcomings (2)

- Security
  - Denial of service attacks:
    Undesired traffic dropped at receiver, not in network
  - Other attacks: hard to trace, no sender signature
  - BGP misconfiguration can create havoc
    - Example: Pakistan created YouTube black hole
  - BGP implementation errors can create havoc
    - Example: Czech provider creates huge AS path
      => Many routers crash world-wide
      => Wildly oscillates
  - Question: What about concerted attack on BGP…? ☹ ☹ ☹
- Routing = destination-based
  - No complete choice of paths
  - Restricts solutions for traffic engineering

Network layer: Weaknesses and shortcomings (3)

- No network congestion control:
  Dynamic routing / dynamic traffic engineering = difficult!
  - Tried out in ARPANET: Oscillations everywhere
  - Today: Interaction with TCP congestion control feedback loop → even worse!
- Convergence speed (link/router failures)
  - OSPF: 200ms … several seconds
    - Routing loops may occur during convergence = black holes
  - BGP: seconds to several minutes!
    - Never really converges: there's always something going on
- More and more prefixes in routing tables
  - 300,000 and growing
  - IPv6 does *not* help! (in contrast…)

Network Layer: Weaknesses and shortcomings (4)

- Manageability
  - Routing = complex to set up
  - Even more complex to manage/debug
    - What/who caused the error? – Difficult to answer!
- End hosts: increasingly mobile
  - WLAN → UMTS? = IP address changes!
- Multicast is not deployed
- Quality of service
  - Different applications have different service demands
    - File transfer: max bandwidth
    - Chat, VoIP, games: min delay
    - E-Mail: min cost
  - QoS = different *classes of service*
  - Works in theory and lab — but is not deployed!
    (same reasons as with multicast)

Future Internet

- Obviously, the Internet as we know it needs to change
- Research term: “Future Internet”
  - Lots of €€$$¥¥££ spent on this
  - Everyone is doing Future Internet research nowadays…
- Revolutionary approach (“clean slate”)
  - Throw everything old away, make it new from scratch
  - Tackle multiple problems at once
- Evolutionary approach
  - Change disturbing aspects separately,
  - one at a time
  - In coexistence with today’s network landscape

THANK YOU

NAT and NAT Traversal

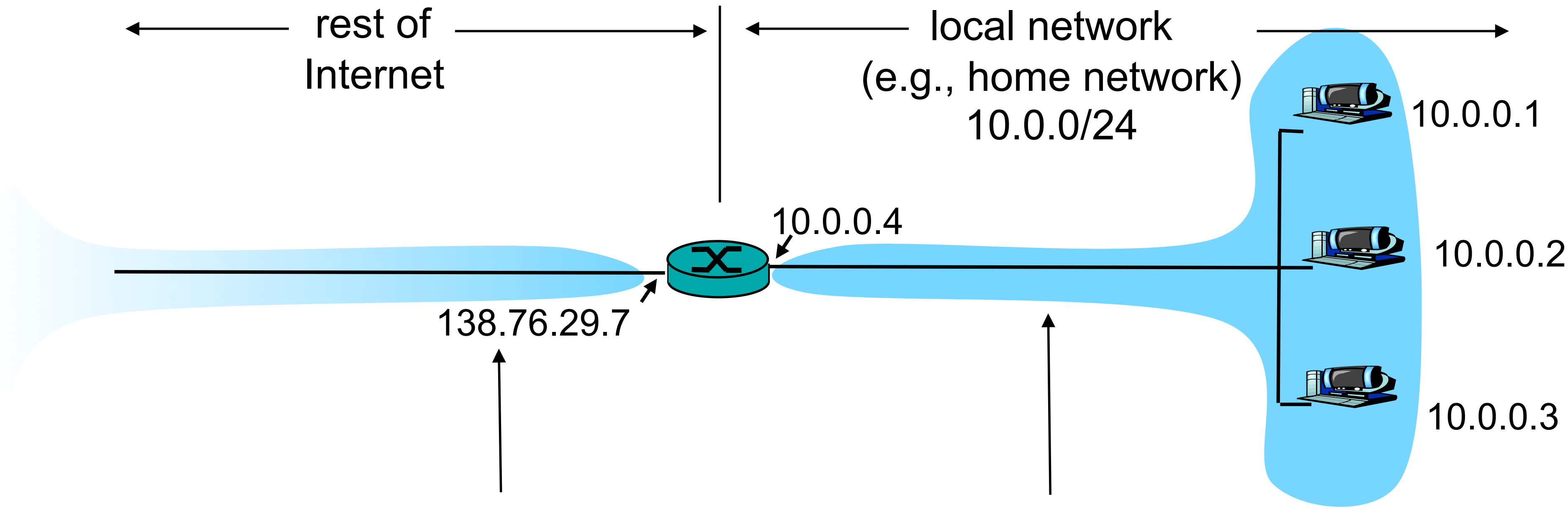
lecturer: Andreas Müller
mueller@net.in.tum.de

NAT: Network Address Translation

- Problem: shortage of IPv4 addresses
  - more and more devices
  - only 32bit address field

- Idea: 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, visible by outside world (a security plus).

NAT: Network Address (and Port) Translation



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

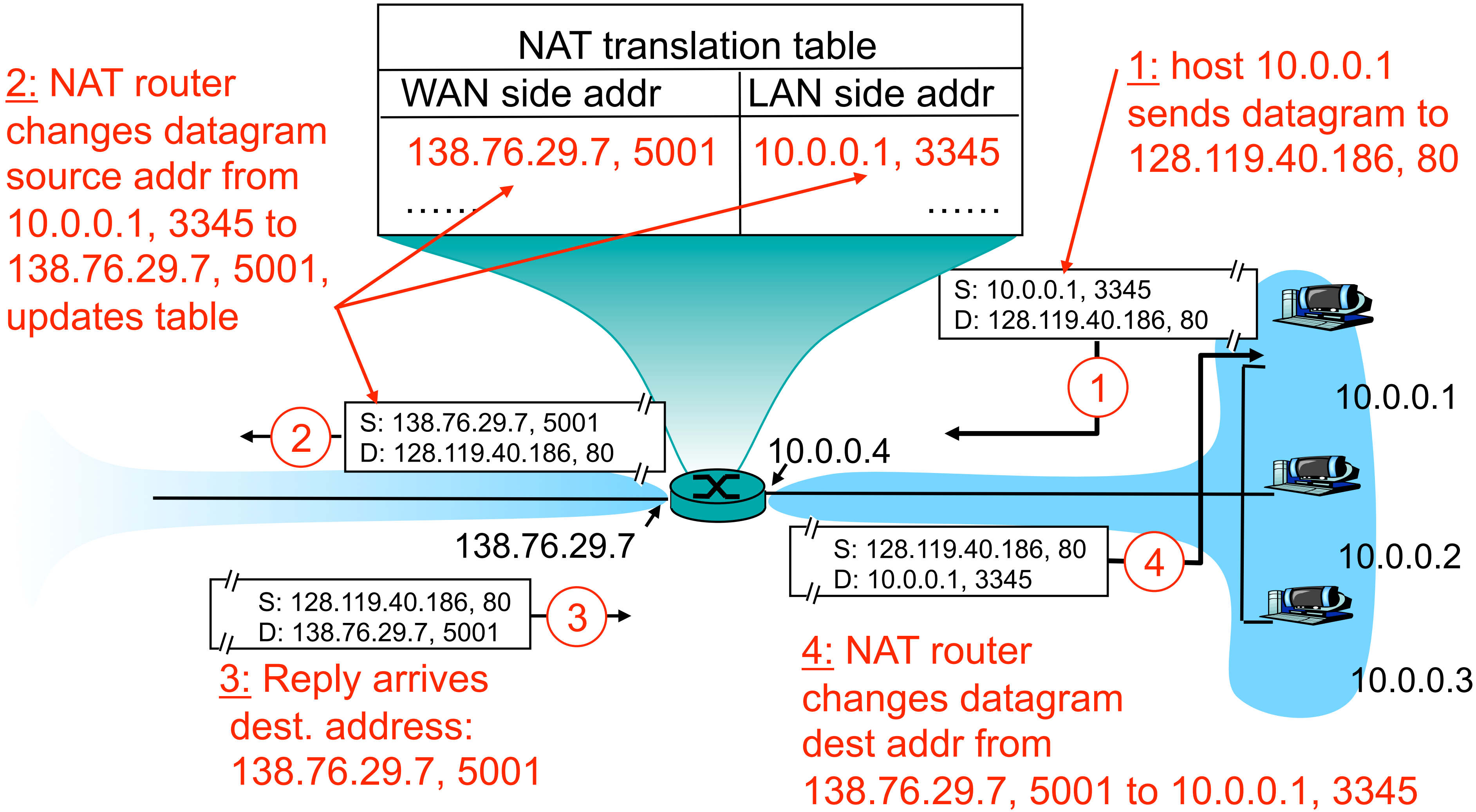
Datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

NAT: Network Address Translation

Implementation: NAT router must:

- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
  . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr.
- *remember (in NAT translation table)* every (source IP address, port #) to (NAT IP address, new port #) translation pair
  -> we have to maintain a state in the NAT
- *incoming datagrams: replace* (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: Network Address Translation



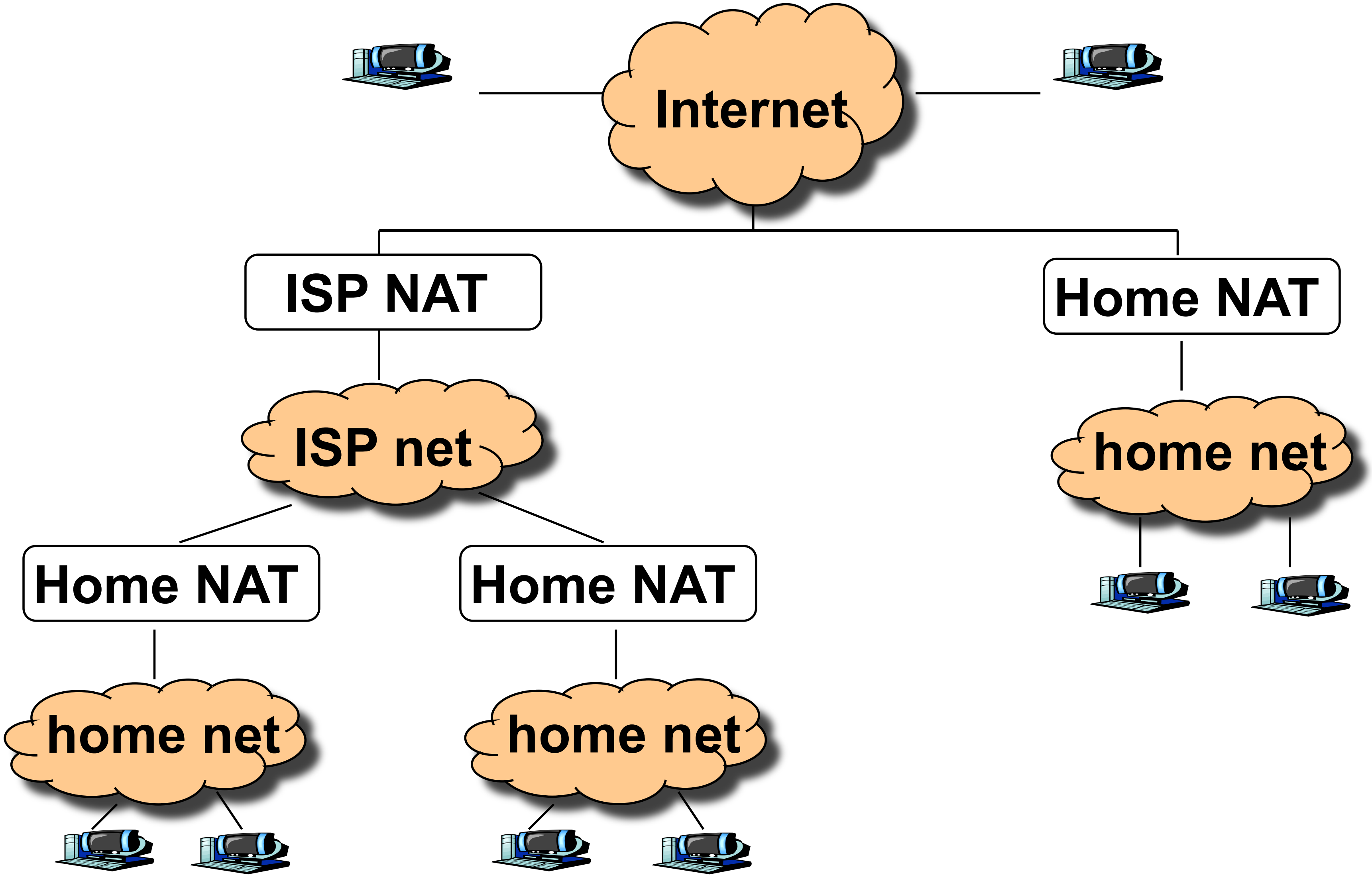
NAT: Network Address Translation

- 16-bit port-number field:
  - ~65000 simultaneous connections with a single LAN-side address!
  - helps against the IP shortage

- 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

Deployment of NAT

- Multiple levels of NAT possible



NAT Implementation

- Implementation not standardized
  - thought as a temporary solution
- implementation differs from model to model
  - if an application works with one NAT does not imply that is always works in a NATed environment
- NAT behavior
  - Binding
    - NAT binding
    - Port binding
  - Endpoint filtering

Binding

- Binding covers context based packet translation
- When creating a new state, the NAT has to assign a new source port and IP address to the connection
- Port binding describes the strategy a NAT uses for the assignment of a new port
- NAT binding describes the behavior of the NAT regarding the reuse of an existing binding
  - 2 consecutive connections from the same source
  - 2 different bindings?

Port binding

- Port-Preservation:
  - the local source port is preserved
- Port-Overloading:
  - port preservation is always used
  - existing state is dropped
- Port-Multiplexing:
  - ports are preserved and multiplexing is done using the destination transport address
  - more flexible
  - additional entry in the NAT table
- No Port-Preservation:
  - the NAT changes the source port for every mapping

NAT binding

- Reuse of existing bindings
  - two consecutive connections from the same transport address
  - NAT binding: assignment strategy for the connections

- Endpoint-Independent
  - the external port is only dependent on the source transport address
  - both connections have the same IP address and port

- Address (Port)-Dependent
  - dependent on the internal and external transport address
  - 2 different destinations result in two different bindings
  - 2 connections to the same destination: same binding

- Connection-Dependent
  - a new port is assigned for every connection
  - strategy could be random, but also something more predictable
  - Port prediction is hard

Endpoint filtering

- Filtering describes
  - how existing mappings can be used by external hosts
  - How a NAT handles incoming connections

- Independent-Filtering:
  - All inbound connections are allowed
  - Independent on source address
  - As long as a packet matches a state it is forwarded
  - No security

- Address Restricted Filtering:
  - packets coming from the same host (matching IP-Address) the initial packet was sent to are forwarded

- Address and Port Restricted Filtering:
  - IP address and port must match

NAT Types

- With Binding and Filtering 4 NAT types can be defined

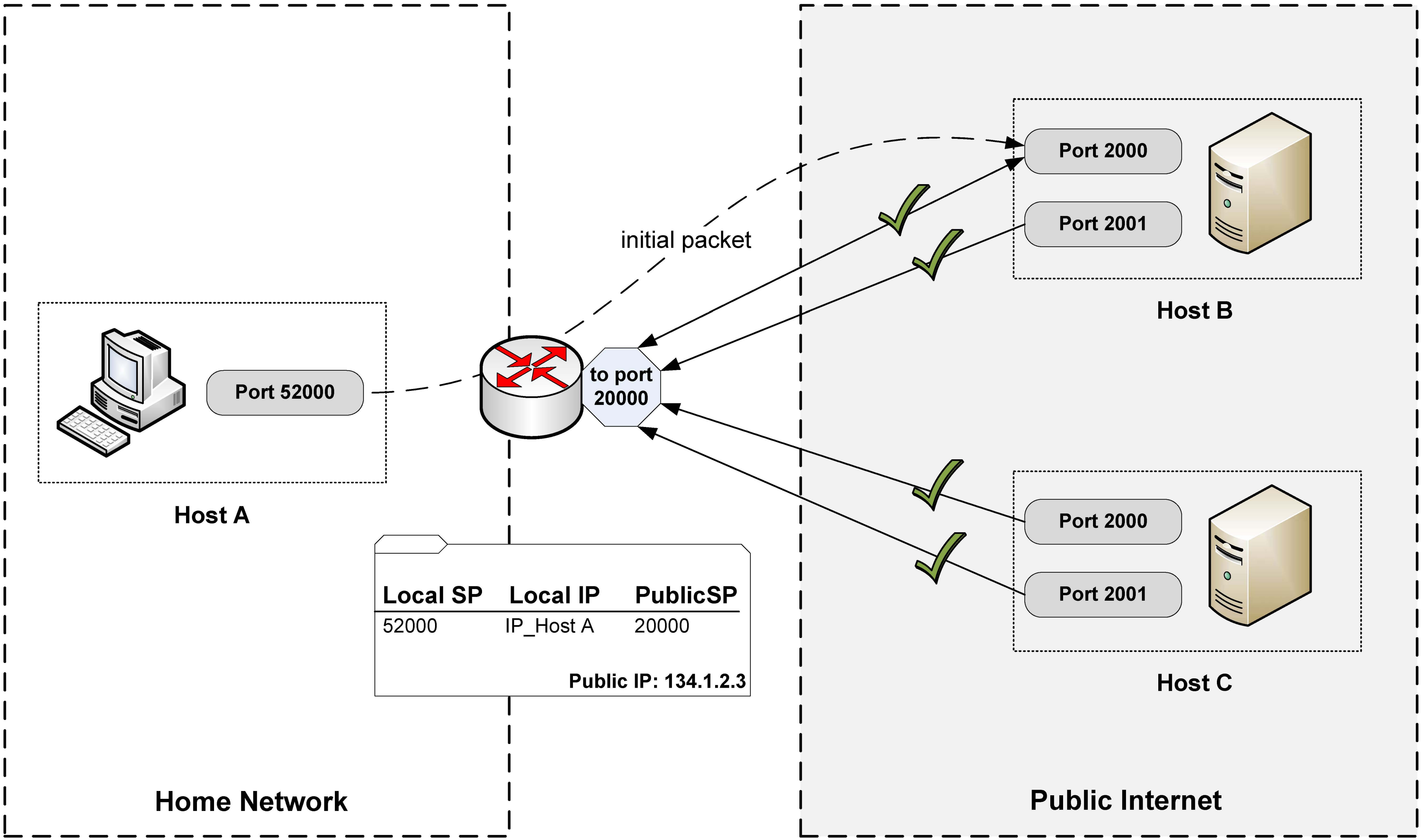
- Full Cone NAT
  - Endpoint independent
  - Independent filtering

- Address Restricted NAT
  - Endpoint independent binding
  - Address restricted filtering

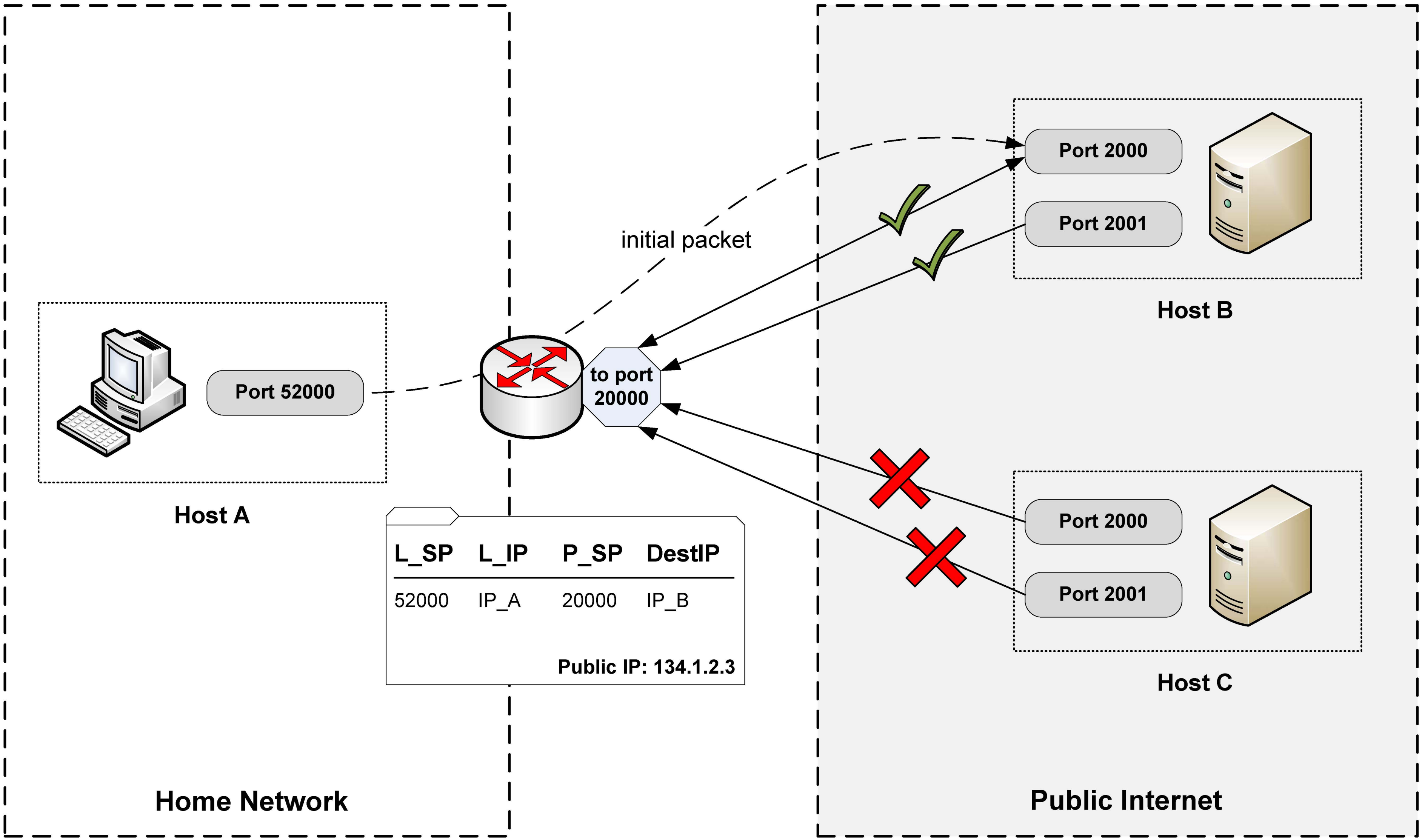
- Port Address Restricted NAT
  - Endpoint independent binding
  - Port address restricted filtering

- Symmetric NAT
  - Endpoint dependent binding
  - Port address restricted filtering

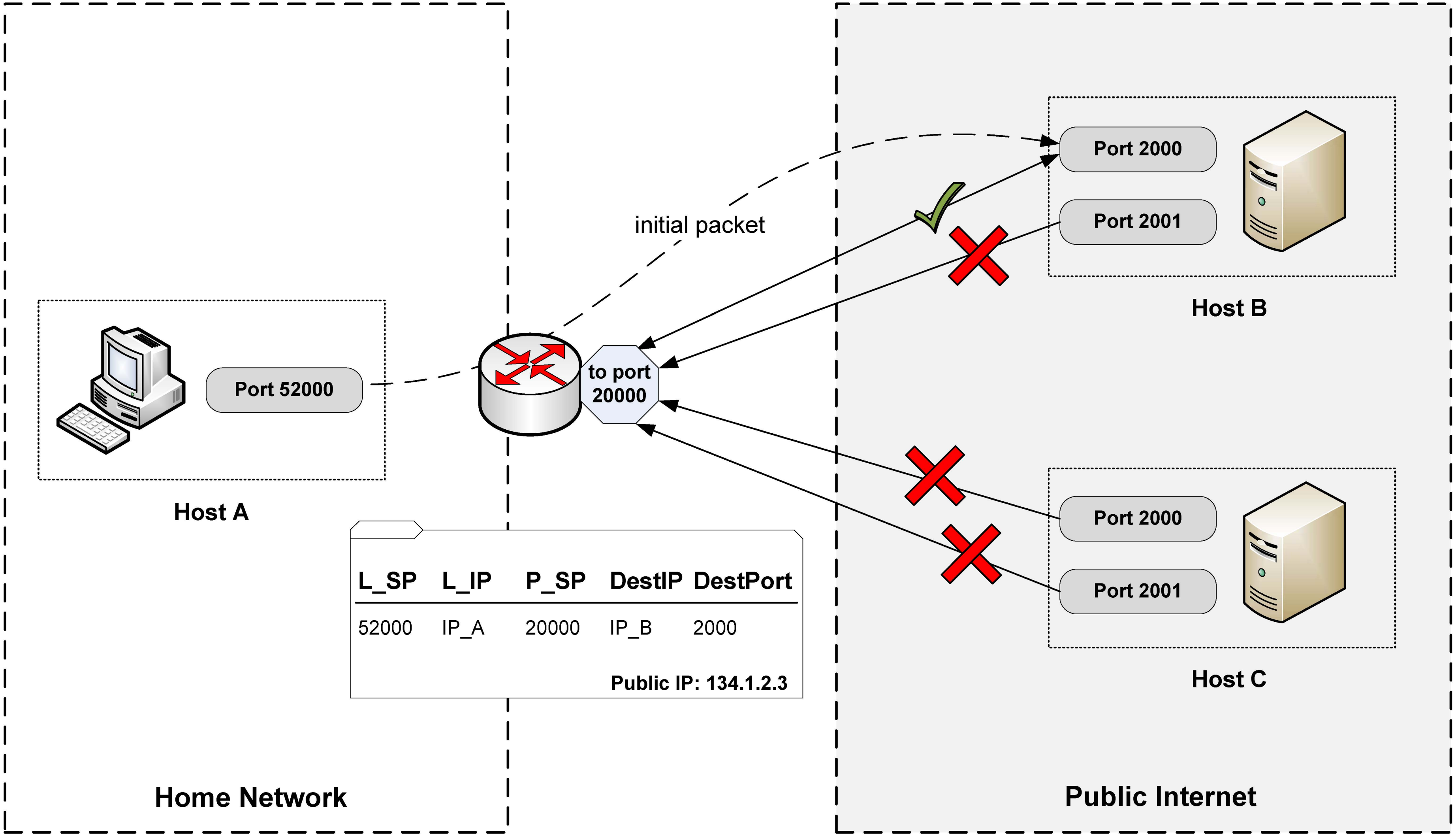
Full Cone NAT



Address Restricted Cone NAT



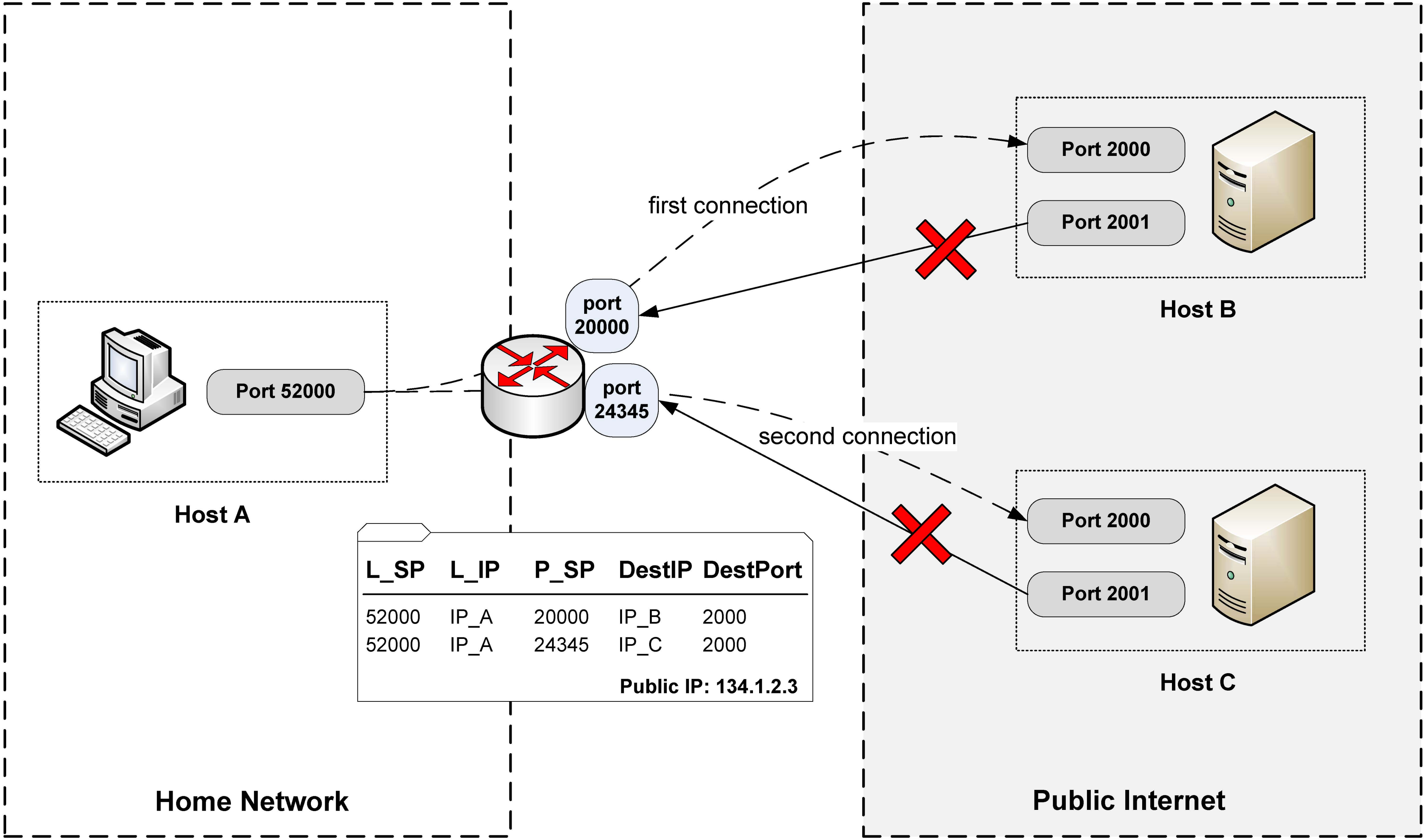
Port Address Restricted Cone NAT



L_SP	L_IP	P_SP	DestIP	DestPort
52000	IP_A	20000	IP_B	2000

Public IP: 134.1.2.3

Symmetric NAT

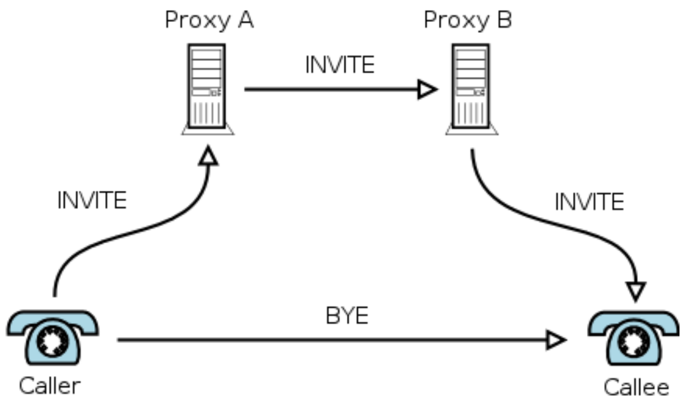


L_SP	L_IP	P_SP	DestIP	DestPort
52000	IP_A	20000	IP_B	2000
52000	IP_A	24345	IP_C	2000

NAT-Traversal Problem

- Divided into four categories: (derived from IETF-RFC 3027)
  - **Realm-Specific IP-Addresses in the Payload**
    - *SIP*
  - **Peer-to-Peer Applications**
    - *Any service behind a NAT*
  - **Bundled Session Applications (Inband Signaling)**
    - *FTP*
    - *RTSP*
    - *SIP together with SDP*
  - **Unsupported Protocols**
    - *SCTP*
    - *IPSec*

Example: Session Initiation Protocol (SIP)



Request/Respone Line

INVITE sip:Callee@200.3.4.5 SIP/2.0

Message-Header

Via: SIP/2.0/UDP **192.168.1.5:5060**
From: < sip:Caller@**192.168.1.5** >
To: <sip:Callee@200.3.4.5>
CSeq: 1 INVITE
Contact: <sip:Caller@192.168.1.5:5060>
Content-Type: application/sdp

Message-Body (optional)

v=0
o=Alice 214365879 214365879 IN IP4 **192.168.1. 5**
c=IN IP4 **192.168.1.5**
t= 0 0
m=audio 5200 RTP/AVP 0 9 7 3
a=rtpmap:8 PCMU/
8000
a=rtpmap:3 GSM/8000

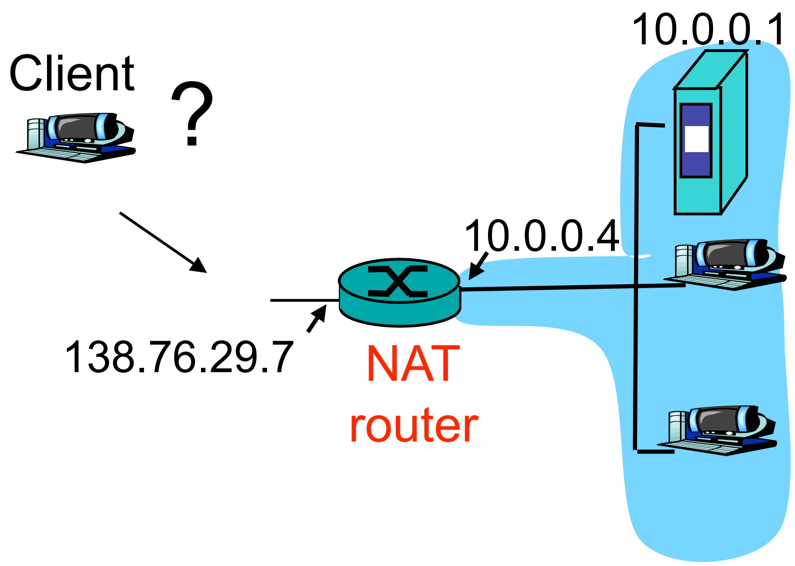
RTP-Session Specification (for 2nd channel)

Media description for 2nd channel

SDP

example: p2p applications

- 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



Existing Solutions to the NAT-Traversal Problem

- Individual solutions
  - Explicit support by the NAT
    - static port forwarding, UPnP, NAT-PMP
  - NAT-behavior based approaches
    - dependent on knowledge about the NAT
    - hole punching using STUN (IETF - RFC 3489)
  - External Data-Relay
    - TURN (IETF - Draft)
    - routing overhead
    - Single Point of Failure

- Frameworks integrating several techniques
  - framework selects a working technique
  - ICE as the most promising for VoIP (IETF - Draft)

Explicit support by the NAT (1)

- Application Layer Gateway (ALG)
  - implemented on the NAT device and operates on layer 7
  - supports Layer 7 protocols that carry realm specific addresses in their payload
    - SIP, FTP

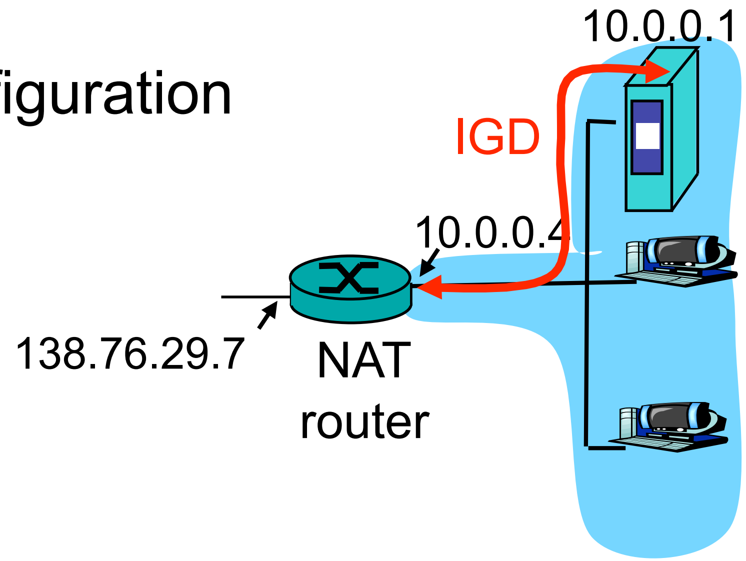
- Advantages
  - transparent for the application
  - no configuration necessary

- Drawbacks
  - protocol dependent (e.g. ALG for SIP, ALG for FTP...)
  - may or may not be available on the NAT device

Explicit support by the NAT (2)

- Universal Plug and Play (UPnP)
  - Automatic discovery of services (via Multicast)
  - Internet Gateway Device (IGD) for NAT-Traversal

- IGD allows NATed host to
  - automate static NAT port map configuration
  - learn public IP address (138.76.29.7)
  - add/remove port mappings (with lease times)

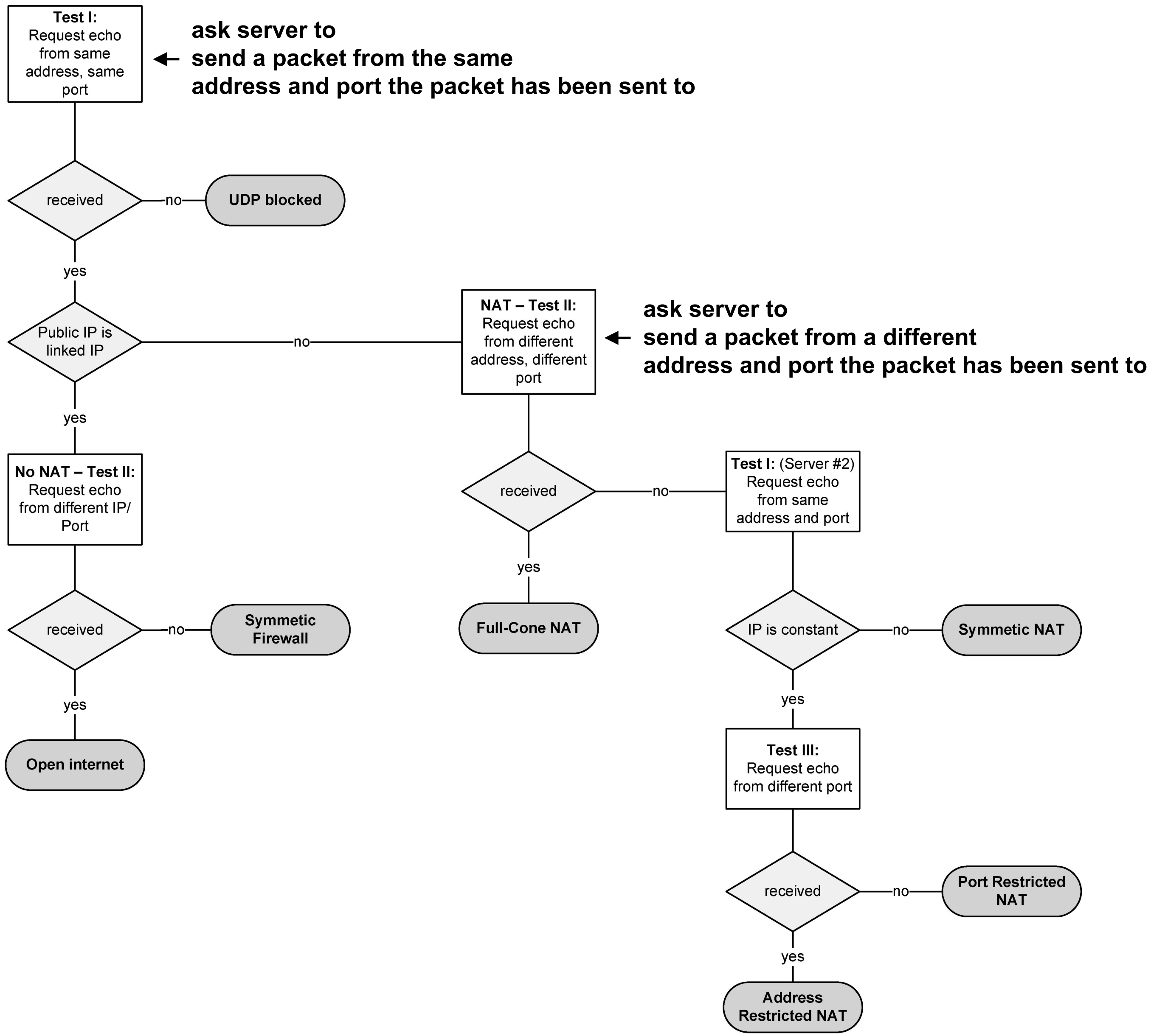


- Drawbacks
  - no security, evil applications can establish port forwarding entries
  - doesn‘t work with cascaded NATs

Behavior based (1): STUN

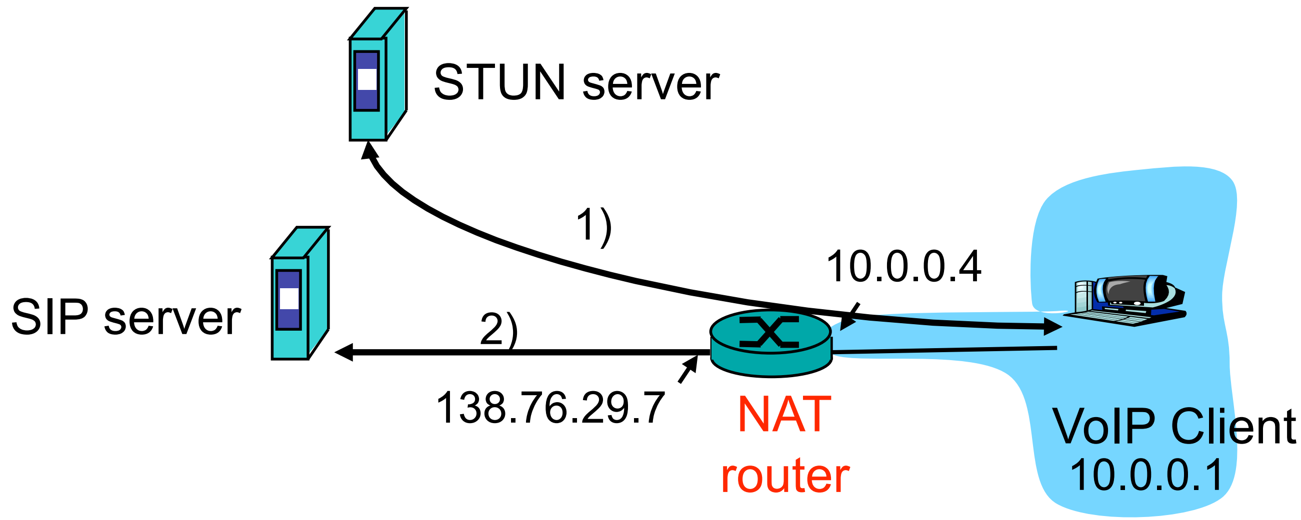
- Simple traversal of UDP through NAT (old)
  - Session Traversal Utilities for NAT (new)
- Lightweight client-server protocol
  - queries and responses via UDP (optional TCP or TCP/TLS)
- Helps to determine the external transport address (IP address and port) of a client.
  - e.g. query from 192.168.1.1:5060 results in 131.1.2.3:20000
- Algorithm to discover NAT type
  - server needs 2 public IP addresses

STUN Algorithm



Example: STUN and SIP

- VoIP client queries STUN server
  - learns its public transport address
  - can be used in SIP packets



Request/Respone Line	INVITE sip:Callee@200.3.4.5 SIP/2.0
Message-Header	Via: SIP/2.0/UDP **138.76.29.7:5060** From: < sip:Caller@**138.76.29.7** > To: <sip:Callee@200.3.4.5> CSeq: 1 INVITE Contact: <sip:Caller@**138.76.29.7:5060**> Content-Type: application/sdp

Limitations of STUN

- STUN only works if
  - the NAT assigns the external port (and IP address) only based on the source transport address
  - Endpoint independent NAT binding
    - Full Cone NAT
    - Address Restricted Cone NAT
    - Port Address restricted cone NAT
  - Not with symmetric NAT!

- Why?
  - Since we first query the STUN server (different IP and port) and then the actual server

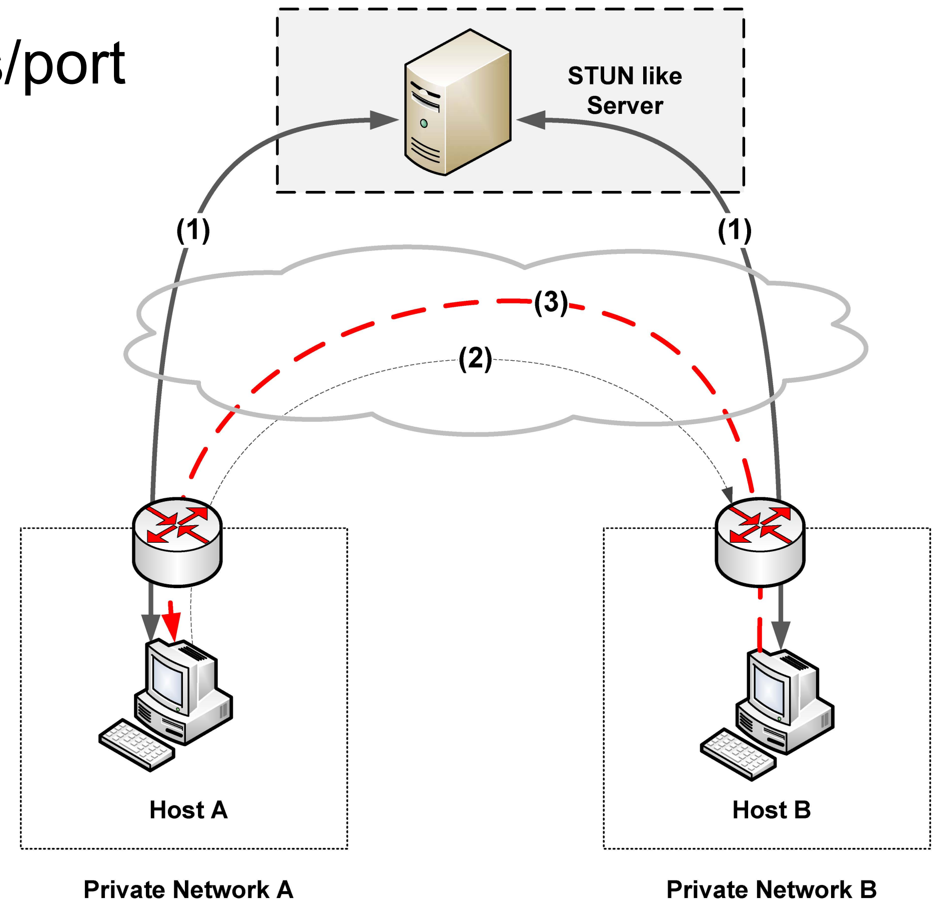
STUN and Hole Punching

- STUN not only helps if we need IP addresses in the payload
  - for establishing a direct connection between two peers

1) determine external IP address/port and exchange it through Rendezvous Point

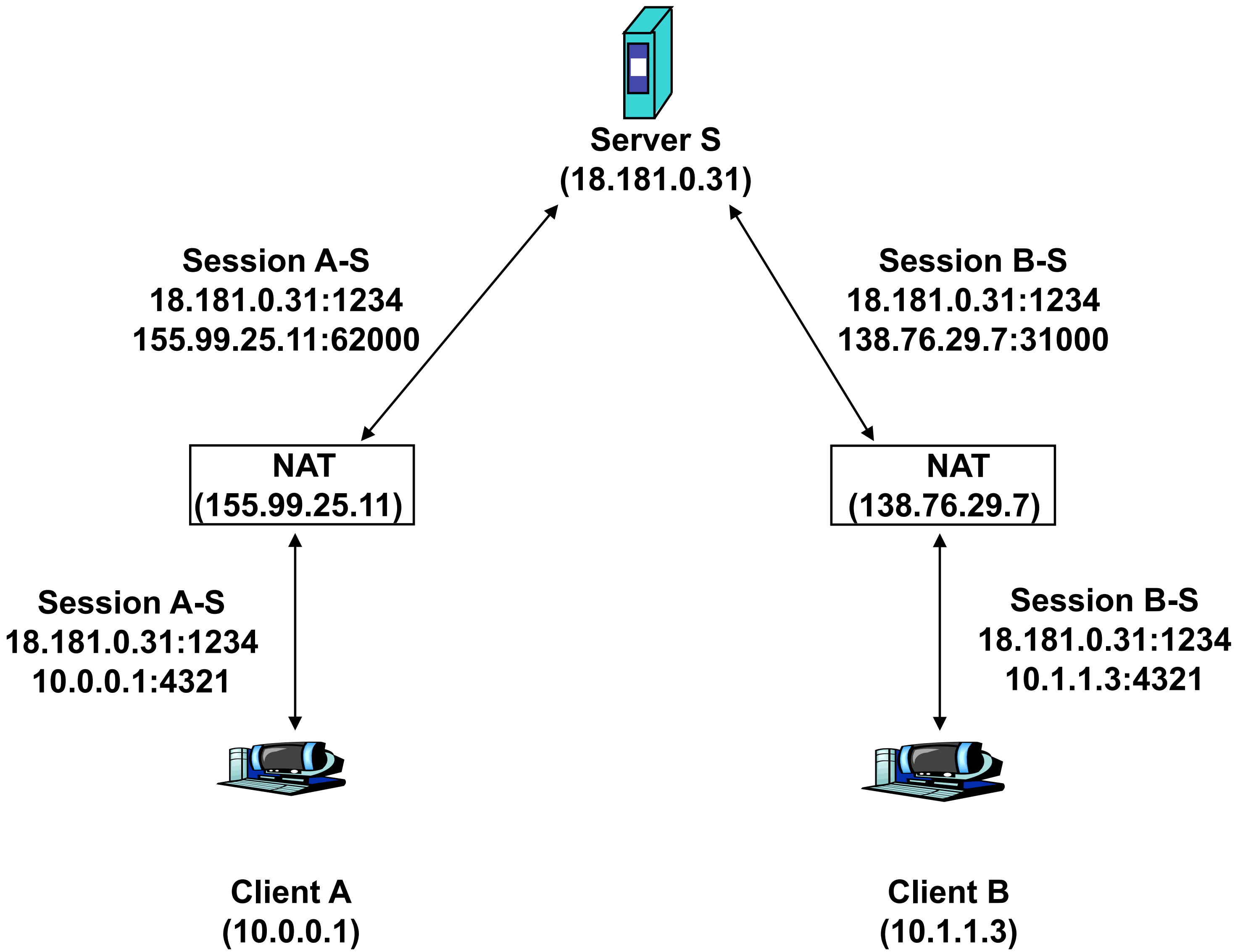
2) both hosts send packets towards the other host outgoing packet creates hole

3) establish connection hole is created by first packet



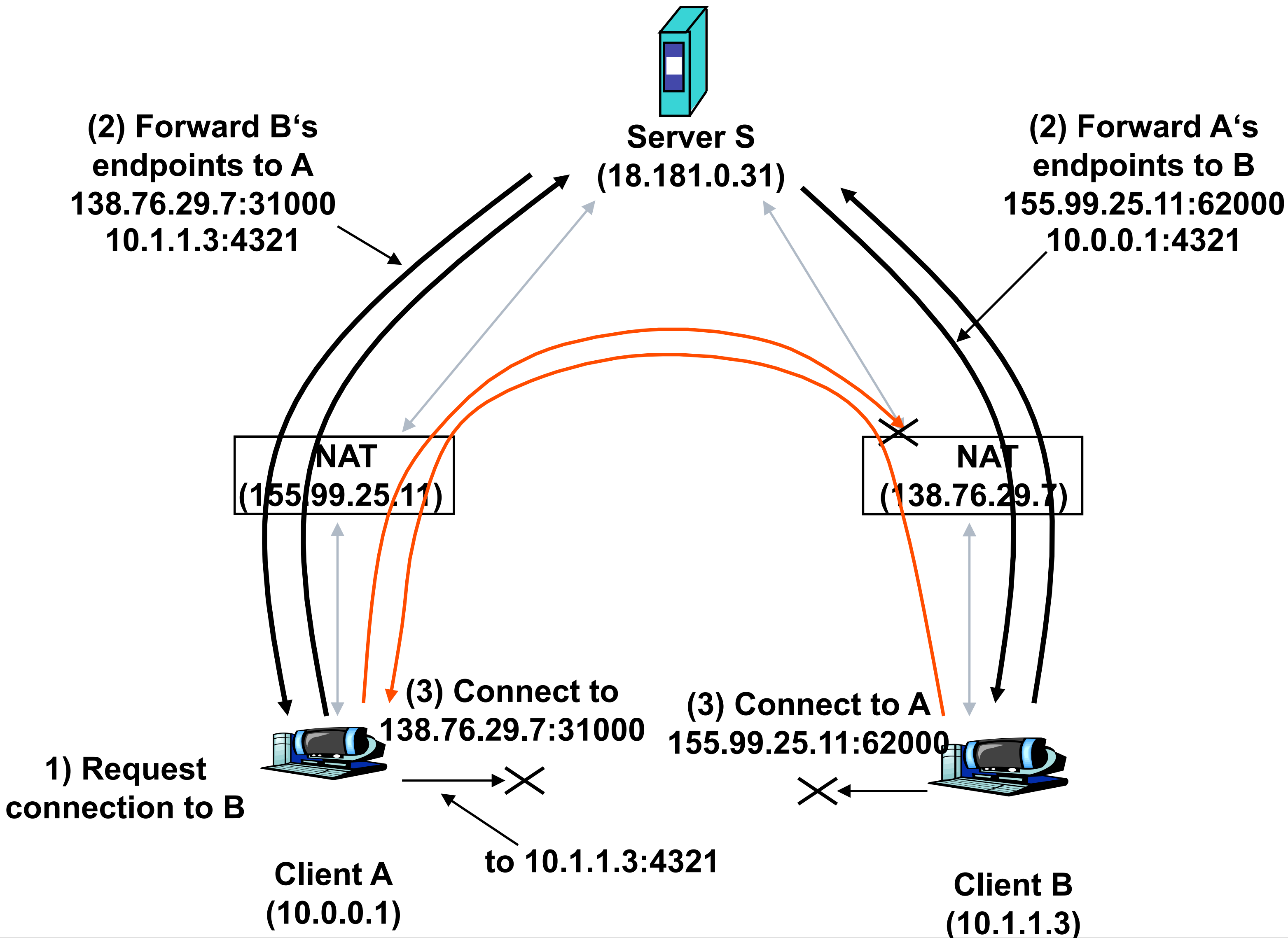
Hole Punching in detail

- Before hole punching



Hole Punching in detail

- Hole punching



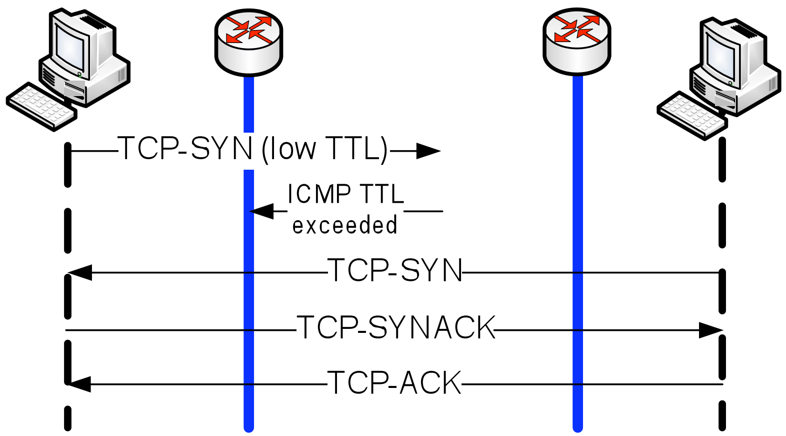
DIY Hole Punching: practical example

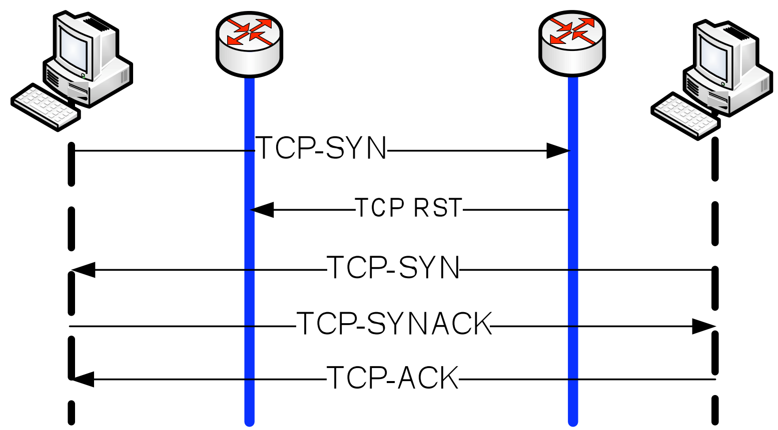
- You need 2 hosts
  - One in the public internet (client)
  - One behind a NAT (server)
- Firstly start a UDP listener on UDP port 20000 on the “server” console behind the NAT/firewall
  - server/1# nc -u -l -p 20000
- An external computer “client" then attempts to contact it
  - client# echo "hello" | nc -p 5000 -u serverIP 20000
  - Note: 5000 is the source port of the connection
- as expected nothing is received because the NAT has no state
- Now on a second console, server/2, we punch a hole
  - Server/2# hping2 -c 1 -2 -s 20000 -p 5000 clientIP
- On the second attempt we connect to the created hole
  - client# echo "hello" | nc -p 5000 -u serverIP 20000

TCP Hole Punching

- Hole Punching not straight forward due to stateful design of TCP
  - 3-way handshake
  - Sequence numbers
  - ICMP packets may trigger RST packets

- Low/high TTL(Layer 3) of Hole-Punching packet
  - As implemented in STUNT (Cornell University)

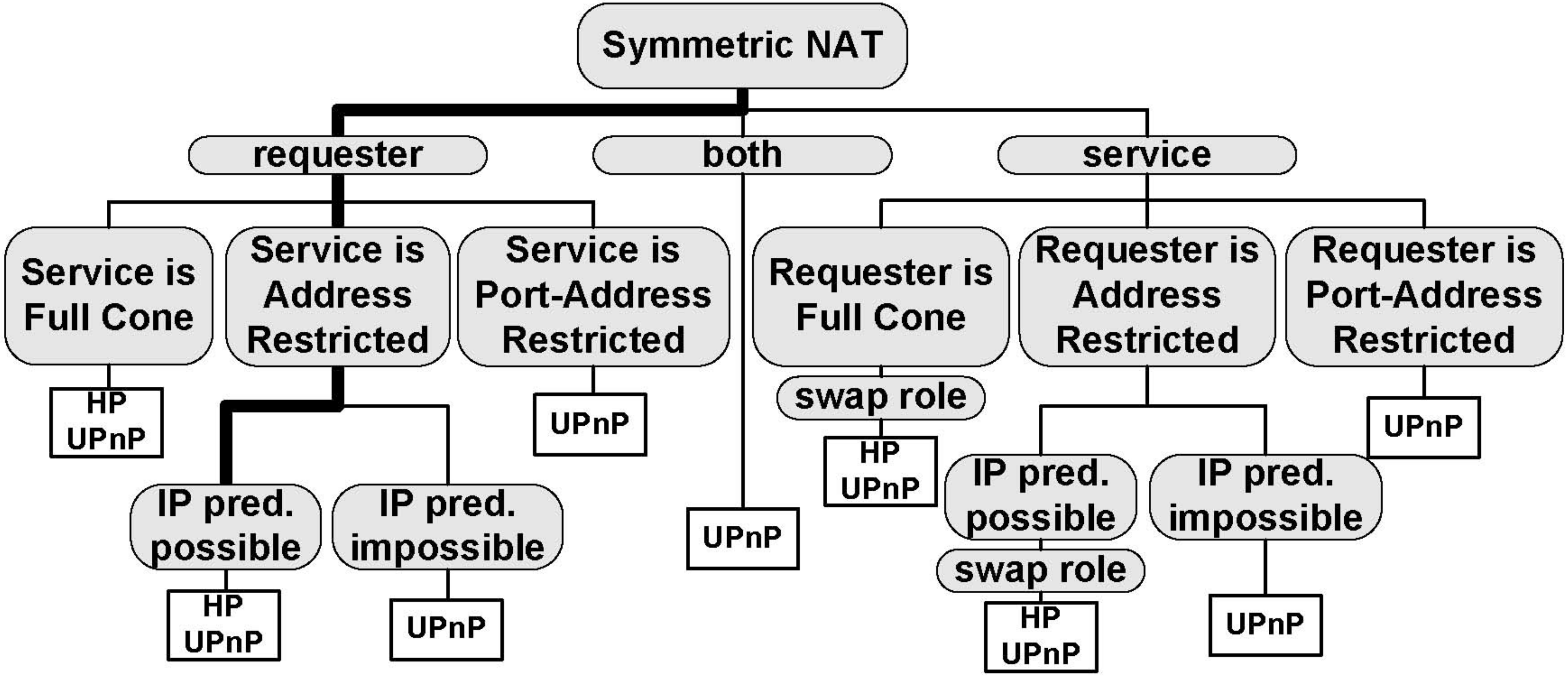




- Bottom line: NAT is not standardized

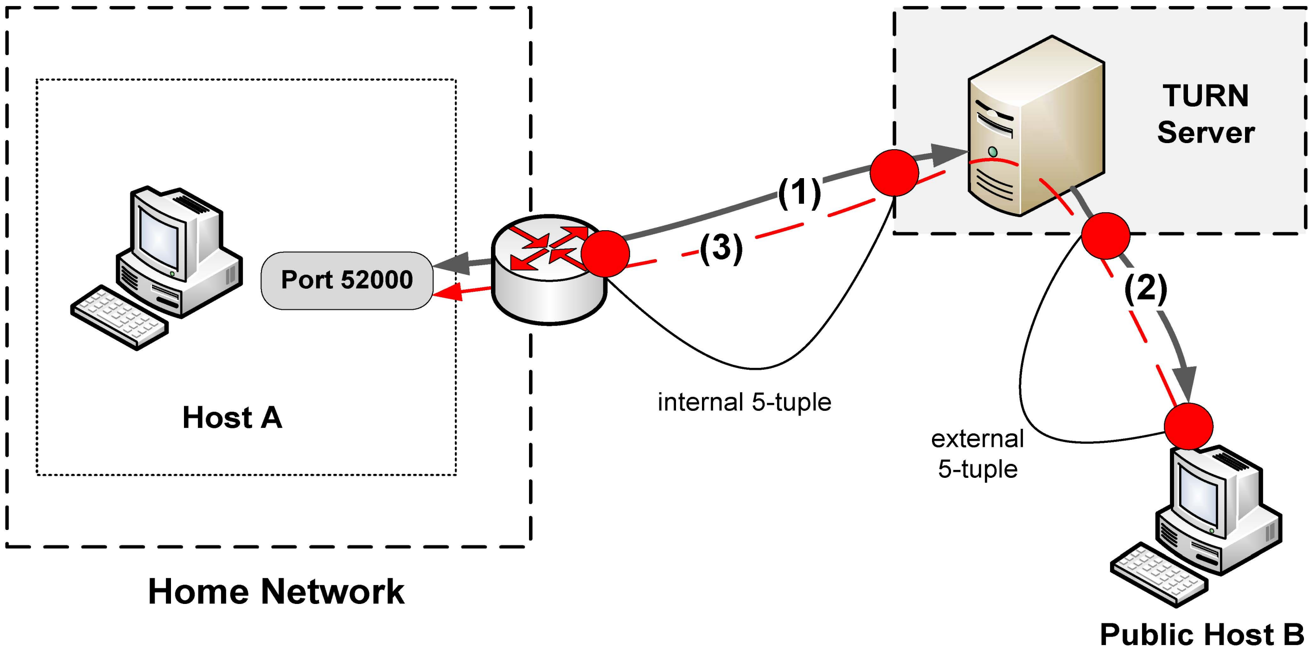
Symmetric NATs

- How can we traverse symmetric NATs
  - Endpoint dependent binding
    - hole punching in general only if port prediction is possible
  - Address and port restricted filtering



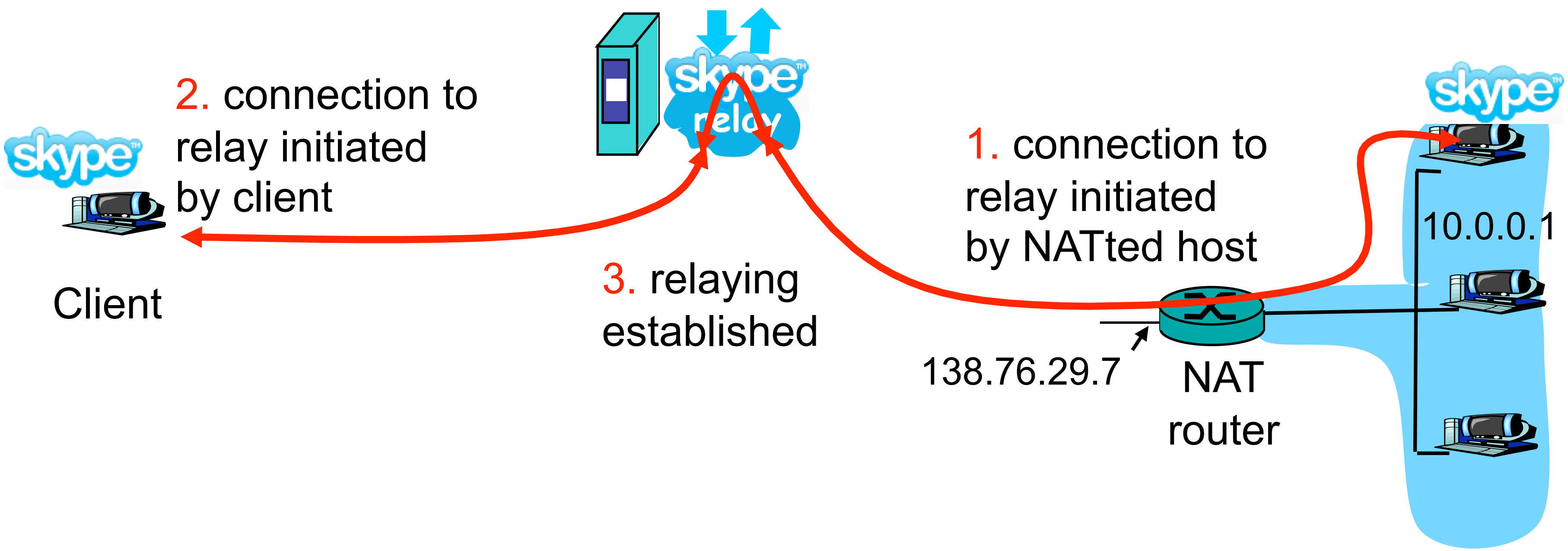
Data Relay (1)

- Idea: Outbound connections are always possible
- 3rd party (relay server) in the public internet
- Both hosts actively establish a connection to relay server
- Relay server forwards packets between these hosts
- TURN as IETF draft



Data Relay

- relaying (used in Skype)
  - NATed client establishes connection to relay
  - External client connects to relay
  - relay bridges packets between to connections
  - IETF draft: TURN



Frameworks

- Interactive Connectivity Establishment (ICE)
  - IETF draft
  - mainly developed for VoIP
  - signaling messages embedded in SIP/SDP
- All possible endpoints are collected and exchanged during call setup
  - local addresses
  - STUN determined
  - TURN determined
- All endpoints are „paired“ and tested (via STUN)
  - best one is determined and used for VoIP session
- Advantages
  - high sucess rate
  - integrated in application
- Drawbacks
  - overhead
  - latency dependent on number of endpoints (pairing)

Success Rates for existing solutions

- http://nattest.net.in.tum.de

- UPnP 31 %

- Hole Punching
  - UDP 73%
  - TCP low TTL 42%
  - TCP high TTL 35%

- Relay 100%

- Propabilities for a direct connection
  - UDP Traversal: 85 %
  - TCP Traversal: 82 %
  - TCP inclusive tunneling: 95 %

New approach

- Advanced NAT-Traversal Service (ANTS)
  - considers different service categories
    - who runs framework
    - which external entities are available?
  - pre-signaling and security
  - knowledge based
    - NAT-Traversal decision is made upon knowledge
  - performance
    - Less latency through knowledge based approach
  - success rates
    - 95% for a direct connection for TCP
  - available for new (API) and legacy applications (TUN)

- for more information
  - http://nattest.net.in.tum.de/?mod=publications

NAT Conclusion

- NAT helps against the shortage of IPv4 addresses
  - only the border gateway needs a public IP address
  - NAT maintains mapping table and translates addresses
- NAT works as long as the server part is in the public internet
- P2P communications across NATs are difficult
  - NAT breaks the end-to-end connectivity model
- NAT behavior is not standardized
  - keep that in mind when designing a protocol
- many solutions for the NAT-Traversal problem
  - none of them works with all NATs
  - framework can select the most appropriate technique

NAT and IPv6

- IPv6 provides a 128bit address field
  - do we still need NAT?
- Firewall traversal
  - realm specific IP addresses in the payload
  - bundled session applications
- Topology hiding
  - „security“
- Business models of ISPs
  - how many IP addresses do we really get (for free)?
- NAT for IPv6 (NAT66) standardization already started (IETF)
  - goal: „well behaved NAT“

Master Course
Computer Networks
IN2097

Prof. Dr.-Ing. Georg Carle
Christian Grothoff, Ph.D.
Dr. Nils Kammenhuber

Chair for Network Architectures and Services
Institut für Informatik
Technische Universität München
http://www.net.in.tum.de

Chapter 4 - Network Measurements

- Introduction
- Architecture & Mechanisms
- Protocols
  - IPFIX (Netflow Accounting)
  - PSAMP (Packet Sampling)
- Scenarios

Network Measurements

- Active measurements
  - “intrusive”
  - Measurement traffic is generated and sent via the operational network
  - Advantages
    - Straightforward
    - Does not depend on existing traffic by active applications
    - Allows measurement of specific parts of the network
  - Disadvantages
    - Additional load
    - Network traffic is affected by the measurement
    - Measurements are influenced by (possibly varying) network load

Network Measurements II

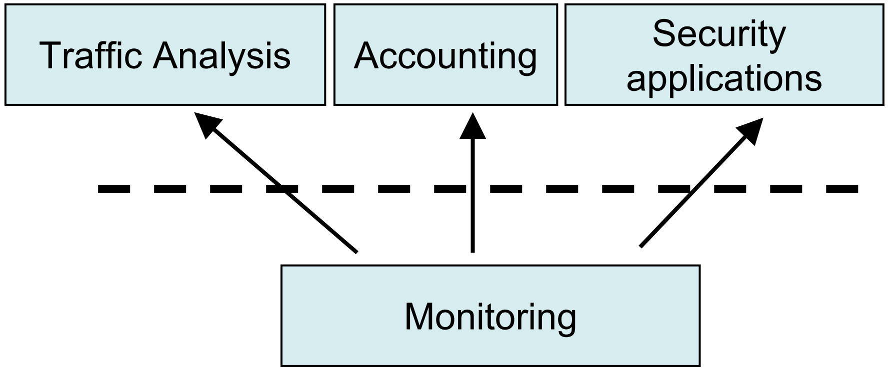
- Passive measurements (or **Network Monitoring**)
  - “non-intrusive”
  - Monitoring of existing traffic
  - Establishing of packet traces at different locations
  - Identification of packets, e.g. using hash values
  - Advantages
    - Does not affect applications
    - Does not modify the network behavior
  - Disadvantages
    - Requires active network traffic
    - Limited to analysis of existing / current network behavior, situations of high load, etc. cannot be simulated/enforced
    - Does not allow the transport of additional information (time stamps, etc.) within measured traffic

Network Measurements III

- Hybrid measurements
  - Modification of packet flows
    - Piggybacking
    - Header modification
  - Advantages
    - Same as for “passive”
    - additional information can be included (time-stamps, etc.)
  - Disadvantages
    - Modifying of data packets may cause problems if not used carefully

Network Monitoring

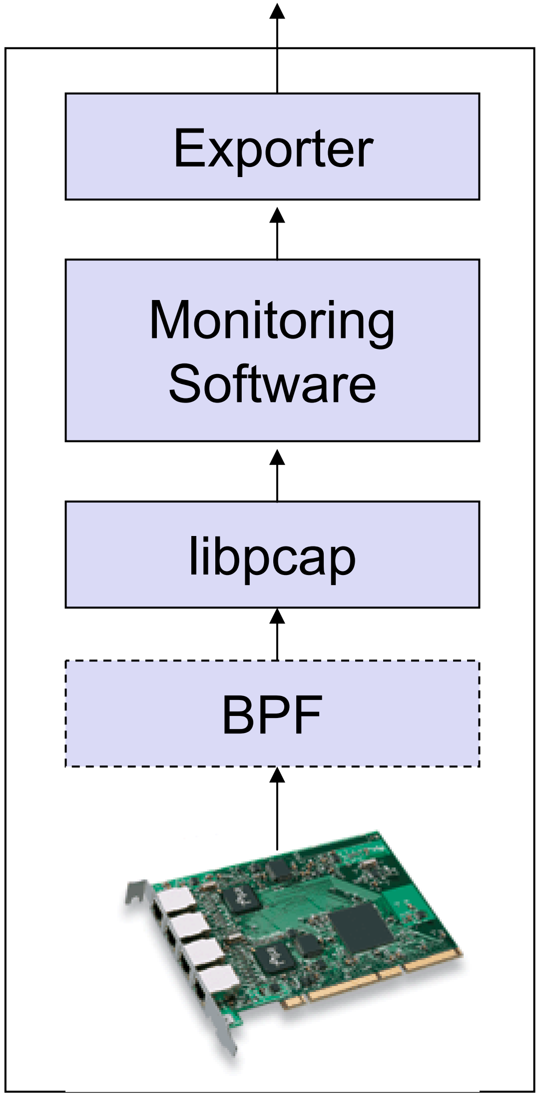
- Applications of network monitoring
  - Traffic analysis
    - Traffic engineering
    - Anomaly detection
  - Accounting
    - Resource utilization
    - Accounting and charging
  - Security
    - Intrusion detection
    - Detection of prohibited data transfers (e.g., P2P applications)



- Open issues
  - Protection of measurement data against illegitimate use (encryption, …)
  - Applicable law (“lawful interception”)

Monitoring Probe

- Standardized data export
- Monitoring Software
- HW adaptation, [filtering]
- OS dependent interface (BSD)
- Network interface



High-Speed Network Monitoring

- Requirements
  - Multi-Gigabit/s Links
  - Cheap hardware and software → standard PC
  - Simple deployment

- Problems
  - Several possible bottlenecks in the path from capturing to final analysis

Bottlenecks?

Packet capturing → Pre-processing → Statistics exporting → Statistics collecting → Post-processing

High-Speed Network Monitoring II

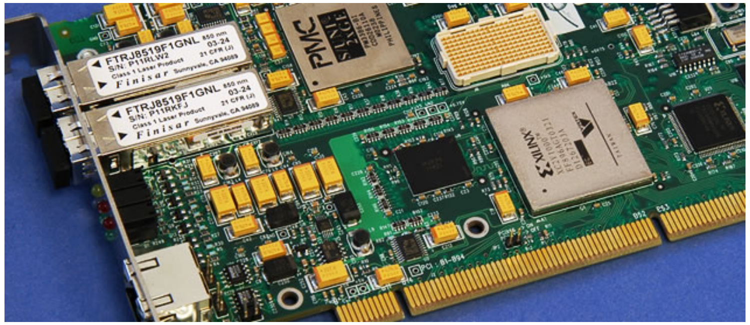
- Approaches
  - High-end (intelligent) network adapters
  - Sophisticated algorithms for
    - Maintaining packet queues
    - Elimination of packet copy operations
    - Managing hash tables describing packet flows
  - Sampling
  - Filtering
  - Aggregation

  ⇨ more on subsequent slides

Special Network Adapters

- Server NICs (Network Interface Cards)
  - Direct access to main memory (without CPU assistance)
  - Processing of multiple packets in a single block (reduction of copy operations)

    → Reduced interrupt rates



- Monitoring interface cards
  - Dedicated monitoring hardware
  - Programmable, i.e. certain processing can be performed on the network interface card

Packet capturing → Pre-processing → Statistics exporting → Statistics collecting → Post-processing

Memory Management

- Hash tables
  - Allow fast access to previously stored information
  - Depending on the requirements, different sections of a packet can be used as input to the hash function.
- Reduction of copy operations
  - Copy operations can be reduced by only transferring references pointing to memory positions holding the packet
  - Management of the memory is complex, garbage collection required
- Aggregation
  - If aggregated results are sufficient, only counters have to be maintained

Packet capturing → $\boxed{\text{Pre-processing}}$ → Statistics exporting → Statistics collecting → Post-processing

Packet Sampling

- Goals
  - Reduction of the number of packets to analyze
  - Statistically dropping packets
- Sampling algorithms
  - Systematic sampling
    - Periodic selection of every n-th element of a trace
    - Selection of all packets that arrive at pre-defined points in time
  - Random sampling
    - n-out-of-N
    - Probabilistic

Packet Filtering

- Goals
  - Reduction of the number of packets to analyze
  - Possibility to look for particular packet flows in more detail, or to completely ignore other packet flows
- Filter algorithms (explained subsequently)
  - Mask/match filtering
  - Router state filtering
  - Hash-based selection

Packet Filtering – Algorithms

- Mask/match filtering
  - Based on a given mask and value
  - In the simplest case, the selection range can be a single value in the packet header (e.g., mask out the least significant 6 bits of source IP address, match against 192.0.2.0)
  - In general, it can be a sequence of non-overlapping intervals of the packet
- Router state filtering
  - Selection based on one or more of the following conditions
    - Ingress/egress interface is of a specific value
    - Packet violated ACL on the router
    - Failed RPF (Reverse Path Forwarding)
    - Failed RSVP
    - No route found for the packet
    - Origin/destination AS equals a specific value or lies within a given range

Packet Filtering – Algorithms II

- Hash-based filtering
  - Hash function h maps the packet content c, or some portion of it, to a range R
  - The packet is selected if h(c) is an element of S, which is a subset of R called the selection range
  - Required statistical properties of the hash function h
    - h must have good mixing properties
      - Small changes in the input cause large changes in the output
      - Any local clump of values of c is spread widely over R by h
      - Distribution of h(c) is fairly uniform even if the distribution of c is not

Packet Filtering – Algorithms III

- Hash-based filtering (cont.)
  - Usage
    - Random sampling emulation
      - Hash function (normalized) is a pseudorandom variable in the interval [0,1]
    - Consistent packet selection and its application
      - Also known as trajectory sampling
      - If packets are selected quasi-randomly using identical hash function and identical selection range at different points in the network, and are exported to a collector, the latter can reconstruct the trajectories of the selected packets
      - Applications: network path matrix, detection of routing loops, passive performance measurement, network attack tracing

IPFIX: IP Flow Information Export

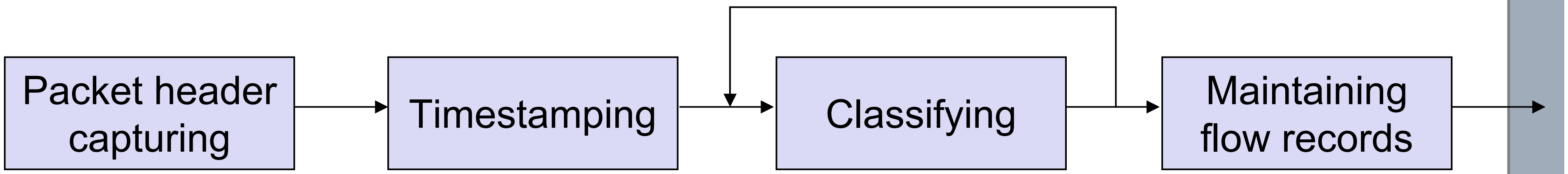
- IPFIX (IP Flow Information eXport) IETF Working Group
  - Standard track protocol based on Cisco Netflow v5…v9
- Goals
  - Collect usage information of individual data flows
  - Accumulate packet and byte counter to reduce the size of the monitored data
- Approach
  - Each flow is represented by its IP 5-tupel (prot, src-IP, dst-IP, src-Port, dst-Port)
  - For each arriving packet, the statistic counters of the appropriate flow are modified
  - If a flow is terminated (TCP-FIN, timeout), the record is exported
  - Sampling algorithms can be activated to reduce the # of flows or data to be analyzed
- Benefits
  - Allows high-speed operation (standard PC: up to 1Gbps)
  - Flow information can simply be used for accounting purposes as well as to detect attack signatures (increasing # of flows / time)

IPFIX - IP Flow Information Export Protocol

- RFCs
  - Requirements for IP Flow Information Export (RFC 3917)
  - Evaluation of Candidate Protocols for IP Flow Information Export (RFC3955)
  - Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information (RFC 5101)
  - Information Model for IP Flow Information Export (RFC 5102)
  - Bidirectional Flow Export using IP Flow Information Export (IPFIX) (RFC 5103)
  - IPFIX Implementation Guidelines (RFC 5153)
- Information records
  - **Template Record** defines structure of fields in **Flow Data Record**
  - Flow Data Record is a data record that contains values of the Flow Parameters
- Transport protocol: transport of information records
  - SCTP must be implemented, TCP and UDP may be implemented
  - SCTP should be used
  - TCP may be used
  - UDP may be used (with restrictions – congestion control!)

IPFIX – Terminology

- IP Traffic Flow
  - A flow is defined as a set of IP packets passing an observation point in the network during a certain time interval. All packets belonging to a particular flow have a set of common properties.
- Observation Point
  - The observation point is a location in the network where IP packets can be observed. One observation point can be a superset of several other observation points.
- Metering Process
  - The metering process generates flow records. It consists of a set of functions that includes packet header capturing, timestamping, sampling, classifying, and maintaining flow records.



IPFIX – Terminology II

- Flow Record
  - A flow record contains information about a specific flow that was metered at an observation point. A flow record contains measured properties of the flow (e.g. the total number of bytes of all packets of the flow) and usually also characteristic properties of the flow (e.g. the source IP address).
- Exporting Process
  - The exporting process sends flow records to one or more collecting processes. The flow records are generated by one or more metering processes.
- Collecting Process
  - The collecting process receives flow records from one or more exporting processes for further processing.

IPFIX – Devices

```
  +---+     +-----+     +---------+       +---------+
   | E-+->   |  E--+->   |    E----+->   <-+--E   E--+->
   | | |     |  |  |     |   / \   |       |  |   |  |
   | M |     |  M  |     |  M   M  |       |  M   M  |
   | | |     | /|\ |     | /|\ /|\ |       | /|\ /|\ |
   | O |     | OOO |     | OOO OOO |       | OOO OOO |
   +---+     +-----+     +---------+       +---------+
   Probe      Basic        Complex          Multiple
              Router       Router           Exporting
                                            Processes
 +---+     +---+     +---+
 | E-+->   | E-+->   | E-+------------->---+
 | | |     | | |     | | | +---+         +-+-----+
 +-+-+     | M |     | M | | E-+------->-+-C-M-E-+->
   |       | | |     | | | | | | +---+   +-+-----+
 +-+-+     +-+-+     | O | | M | | E-+->---+
 | | |       |       +---+ | | | | | |
 | M |     +-+-+           | O | | M |
 | | |     | | |           +---+ | | |           +-----+
 | O |     | O |                 | O |        ->-+-C-E-+->
 +---+     +---+                 +---+           +-----+

Protocol   Remote              Concentrator        Proxy
Converter  Observation
```

O ... Observation point
M ... Metering process
E ... Exporting process

IPFIX – Work Principles

- Identification of individual traffic flows
  - 5-tupel: Protocol, Source-IP, Destination-IP, Source-Port, Destination-Port
  - Example: TCP, 134.2.11.157, 134.2.11.159, 2711, 22
- Collection of statistics for each traffic flow
  - # bytes
  - # packets
- Periodical statistic export for further analysis

Flow	Packets	Bytes
TCP, 134.2.11.157,134.2.11.159, 4711, 22	10	5888
TCP, 134.2.11.157,134.2.11.159, 4712, 25	7899	520.202

IPFIX – Applications

- Usage based accounting
  - For non-flat-rate services
  - Accounting as input for billing
  - Time or volume based tariffs
  - For future services, accounting per class of service, per time of day, etc.
- Traffic profiling
  - Process of characterizing IP flows by using a model that represents key parameters such as flow duration, volume, time, and burstiness
  - Prerequisite for network planning, network dimensioning, etc.
  - Requires high flexibility of the measurement infrastructure
- Traffic engineering
  - Comprises methods for measurement, modeling, characterization, and control of a network
  - The goal is the optimization of network resource utilization

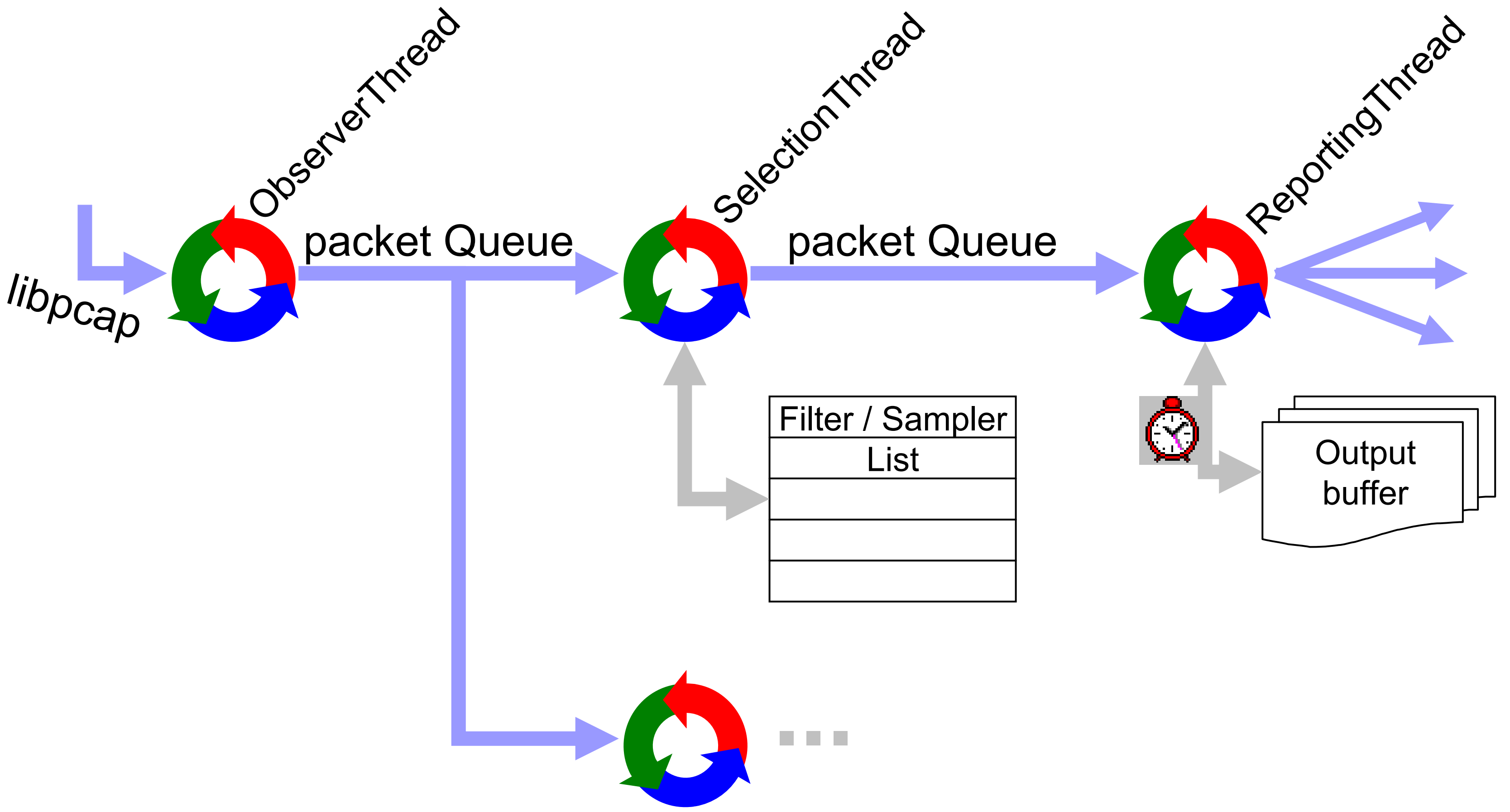
IPFIX – Applications II

- Attack/intrusion detection
  - Capturing flow information plays an important role for network security
  - Detection of security violation
    1) detection of unusual situations or suspicious flows
    2) flow analysis in order to get information about the attacking flows
- QoS monitoring
  - Useful for passive measurement of quality parameters for IP flows
  - Validation of QoS parameters negotiated in a service level specification
  - Often, correlation of data from multiple observation points is required
  - This required clock synchronization of the involved monitoring probes

Packet Sampling

- PSAMP (Packet SAMPling) WG (IETF)
- Goals
  - Network monitoring of ultra-high-speed networks
  - Sampling of single packets including the header and parts of the payload for post-analysis of the data packets
  - Allowing various sampling and filtering algorithms
    - Algorithms can be combined in any order
    - Dramatically reducing the packet rate
- Benefits
  - Allows very high-speed operation depending on the sampling algorithm and the sampling rate
  - Post-analysis for statistical accounting and intrusion detection mechanisms

Implementation of PSAMP / IPFIX



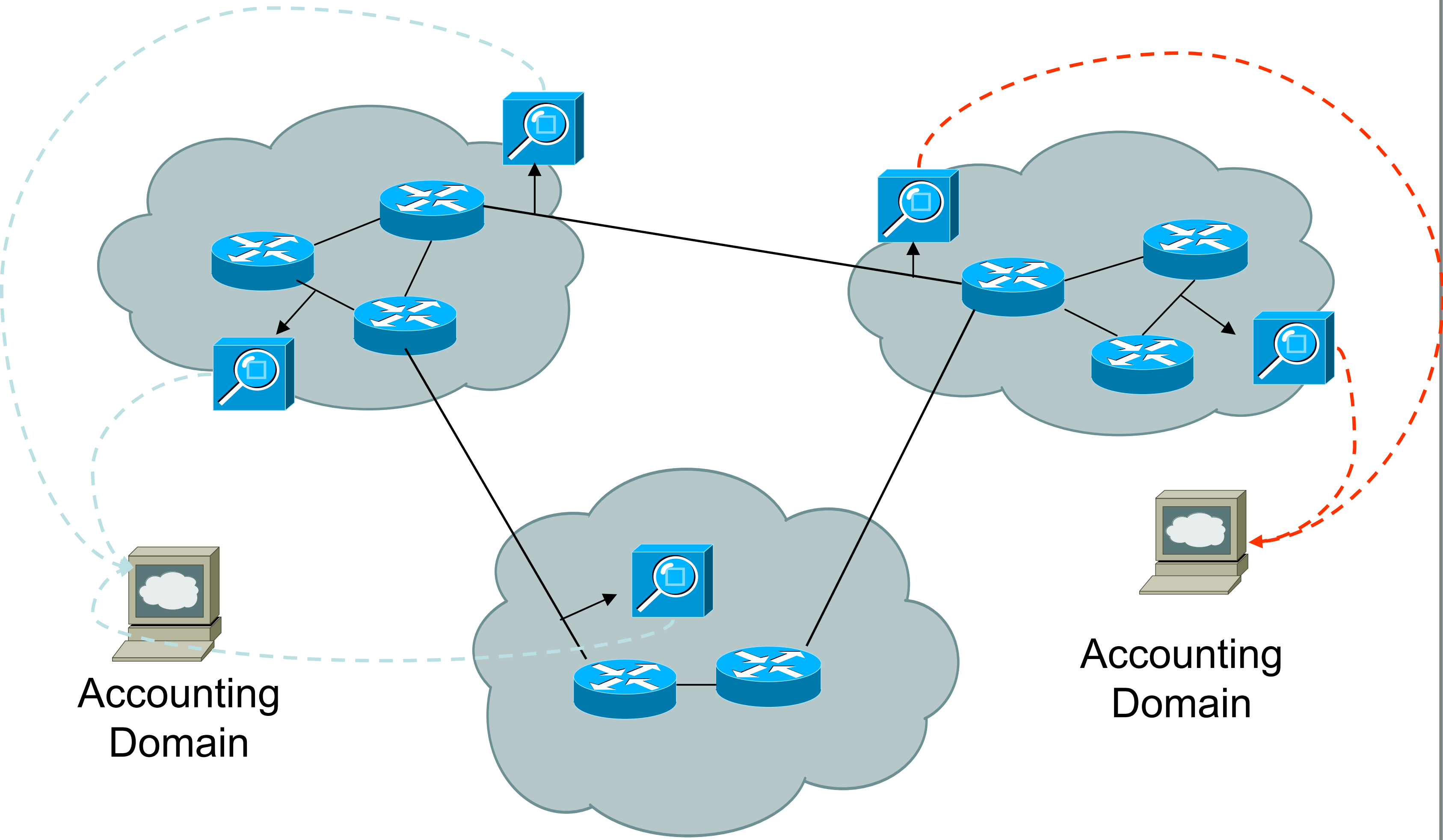
Application Scenarios

- Accounting and Charging
  - Accounting for statistical reasons
  - Accounting for charging

- Traffic engineering
  - Identification of primary traffic paths
  - Optimization of network parameters (e.g. routing parameters) for better network utilization

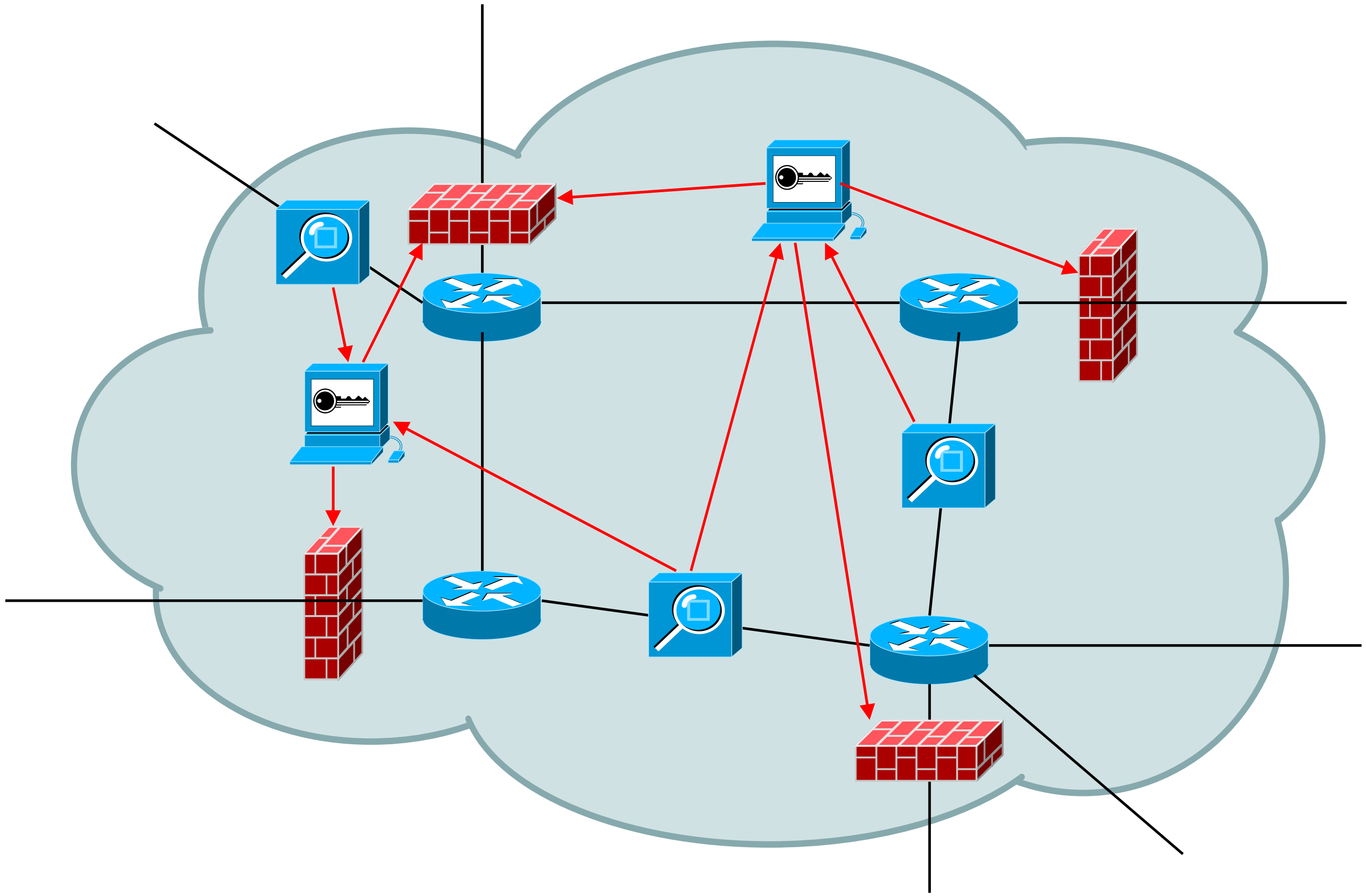
- Network Security
  - Detection of denial-of-service attacks
  - Forensic methods for post-intrusion analysis

Accounting and Charging



Network Security

- Intrusion detection with automated firewall configuration



Master Course
Computer Networks
IN2097

Prof. Dr.-Ing. Georg Carle
Christian Grothoff, Ph.D.
Dr. Nils Kammenhuber

Chair for Network Architectures and Services
Institut für Informatik
Technische Universität München
http://www.net.in.tum.de

MPLS: Multi-Protocol Label Switching

- Motivation + why to use MPLS
- How it works
  - Datagram format
  - Layer 2.5 switching
  - FECs, Labels, and LSPs
- What comes with it
  - LDP
  - CR-LDP
  - RSVP-TE
- GMPLS
- Why not to use MPLS
- Summary

MPLS

- Multi-Protocol Label Switching
- "Layer 2.5":
  - Below IP (Layer 3), but above Link Layer (Layer 2)
  - Borrows a lot of information from IP Layer
  - Borrows a lot of concepts from ATM (Layer 2)
  - "A compromise/marriage between IP world and ATM world"
- Mixture of packet switching and circuit switching
  - Establish virtual circuits (LSPs) between endpoints
  - Send labelled packets along these LSPs
- Used by many, but not all, large ISPs

Why a new protocol? — Deficiencies of IP

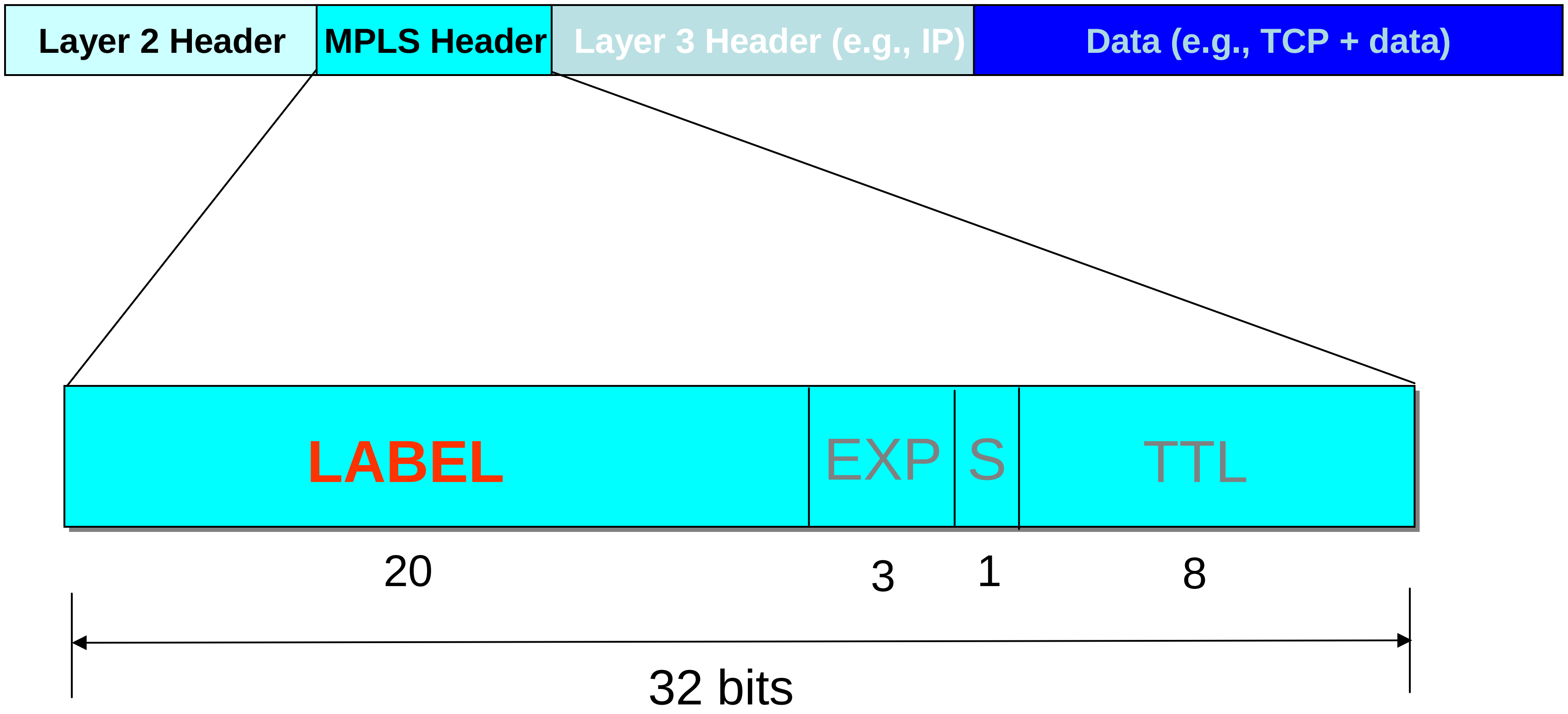
- IP forwarding = longest prefix match on address = expensive
  - < 1K gateways to other ASes; >> 100K interdomain prefixes!
  - Longest prefix match in *every* router on path through our network!
- IP forwarding = destination-based
  - Not all paths possible, cf. exercise #8
  - Would be nicer for traffic engineering
- IP header is long complex
  - Destination, TTL [, QoS bits] at different byte/bit offsets
  - Expensive to parse in hardware
- Traffic of different VPN customers may disturb each other
  - Not visible to each other, but overloads on common links
- IP routing = slow: OSPF convergence 300ms to *X* seconds
  - Routing loops etc. during convergence → packet losses
  - Think of VoIP, videoconferencing, games, telesurgery, …

Why should we use MPLS?

- Original motivation:
  - Switching is faster than routing
  - Build cheaper high-speed “routers” (which *switch* MPLS)
- Today’s motivation:
  - Separation of virtual circuits; MPLS-VPNs
  - Multiservice networks: not only IP
  - Arbitrary paths, better for traffic engineering
  - Fast reroute mechanisms (protection switching): 50ms
  - Better control over routing: More deterministic, more predictable, better for QoS service level agreements (SLAs)

MPLS ideas (1): Short and simple header

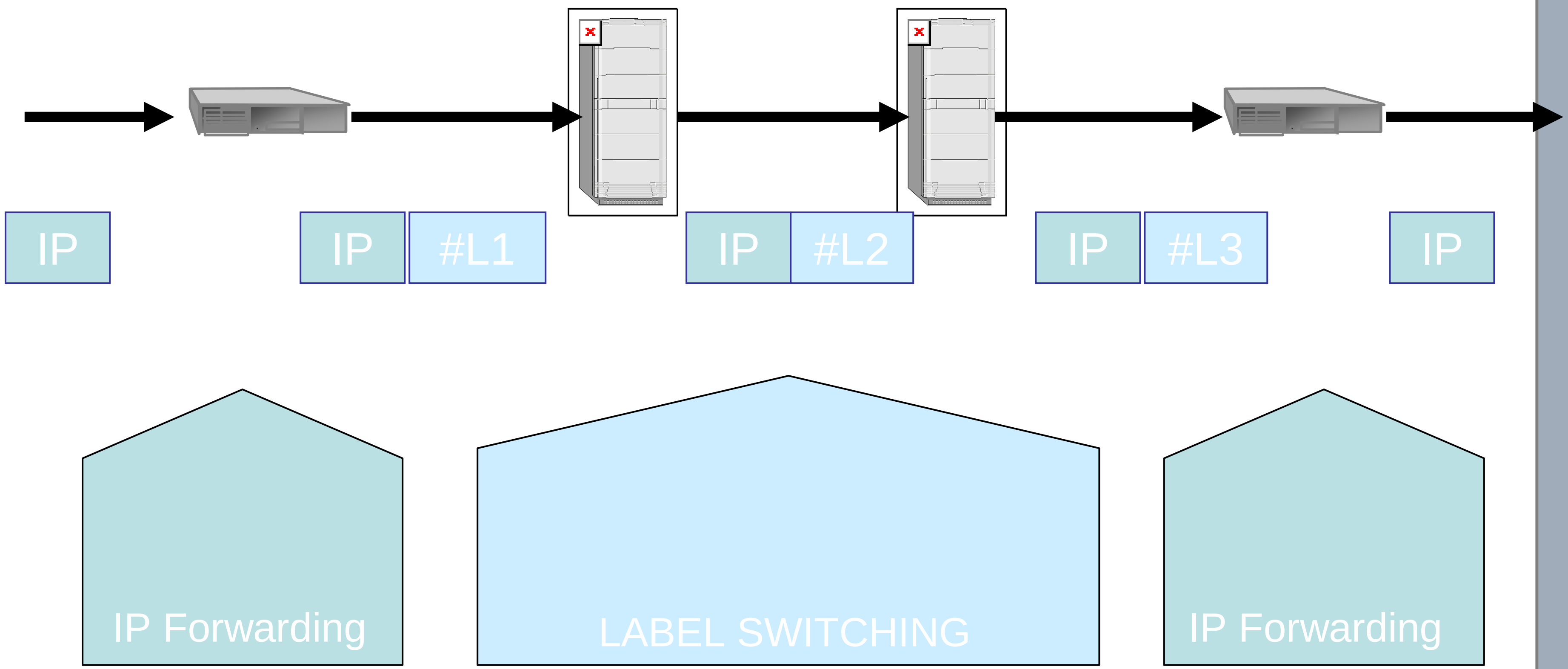
- Easy to parse in hardware



Basic Concepts and Terms

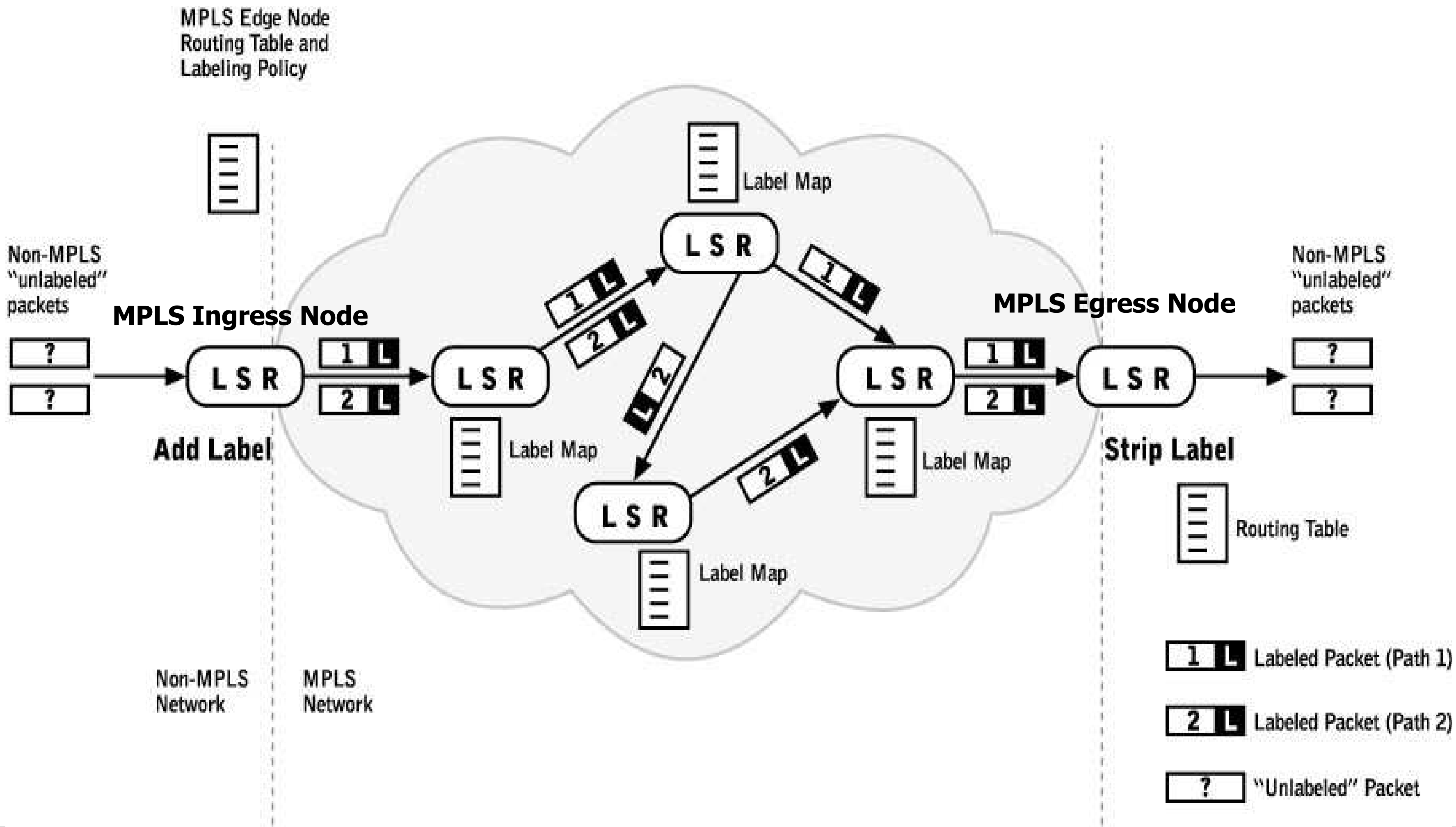
- Label Switching Routers (LSRs): Any router supporting MPLS
- Label
  - A fixed-length (20-bit) address
  - Label semantics are *local* to a router: One label ≠ one path!
  - Labels may be swapped at each router
  - Labels may be stacked: “MPLS in MPLS”
- Label Switched Paths (LSPs)
  - An MPLS virtual circuit: Like a tunnel through the network
  - LSPs are unidirectional
- Forwarding Equivalence Classes (FECs):
  All packets that are to to forwarded….:
  - To the same next hop
  - Out the same interface
  - [With the same forwarding treatment (CoS)]

MPLS ideas (2a): Route at edge, switch in core

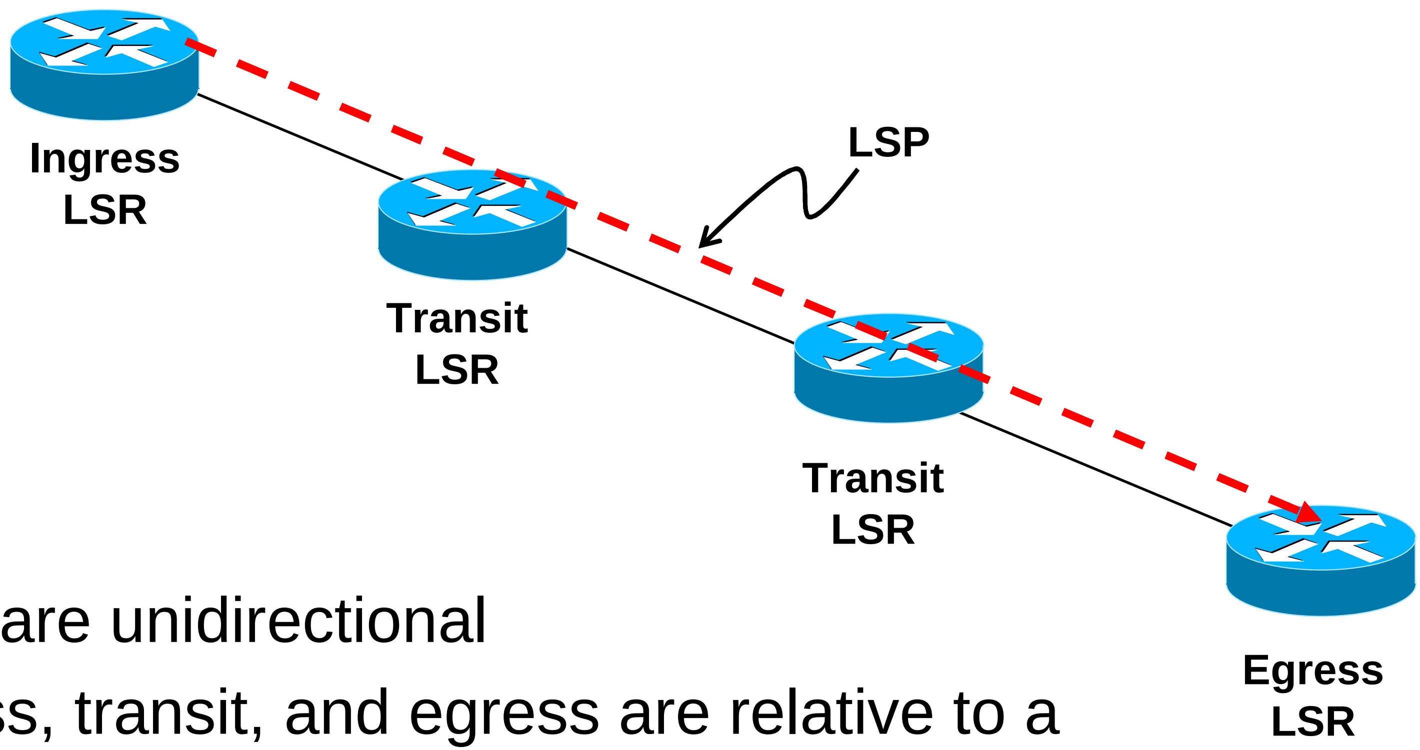


MPLS ideas (2b): Forwarding through the network

- IP packets: Labelled at ingress, label stripped at egress
- Within network: Forwarding by label, not by IP address!



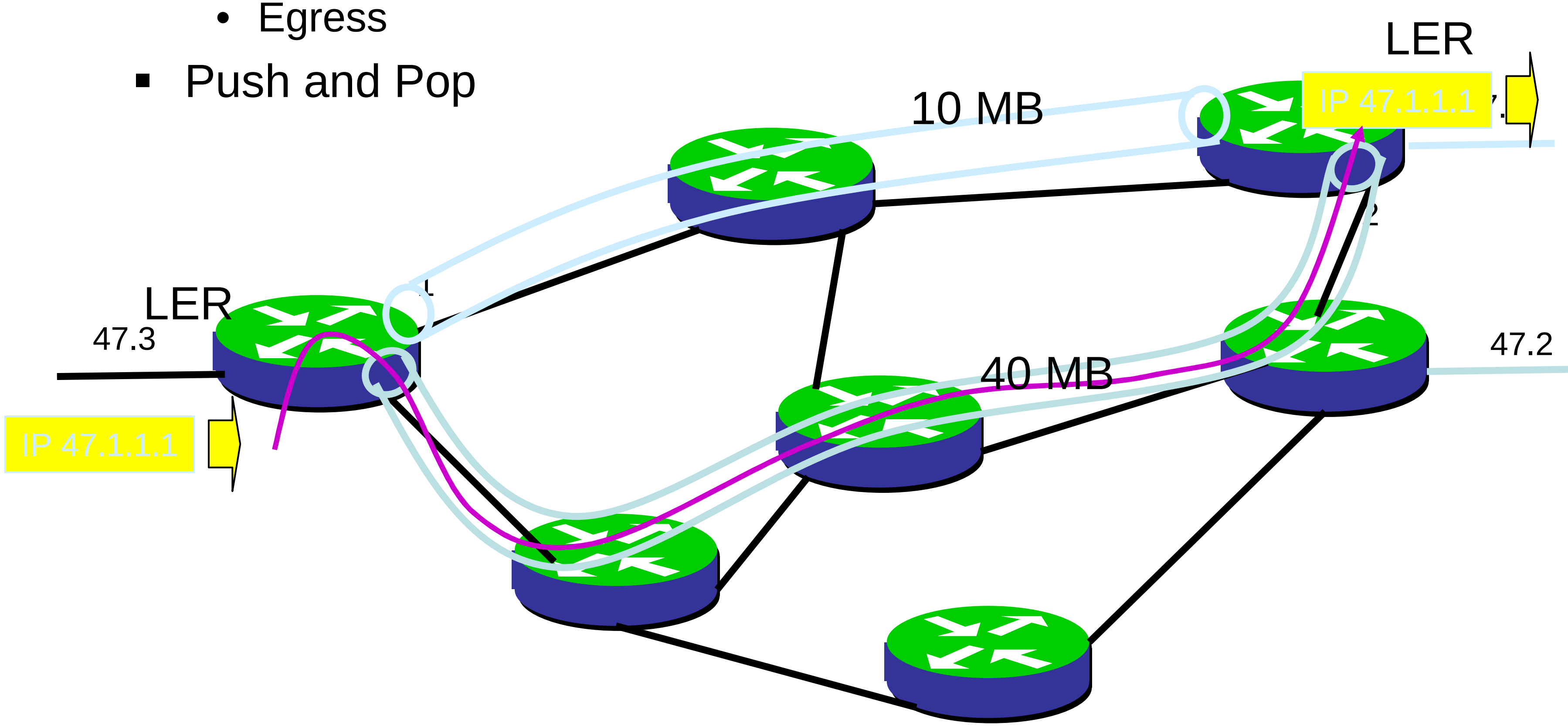
Basic Concepts and Terms



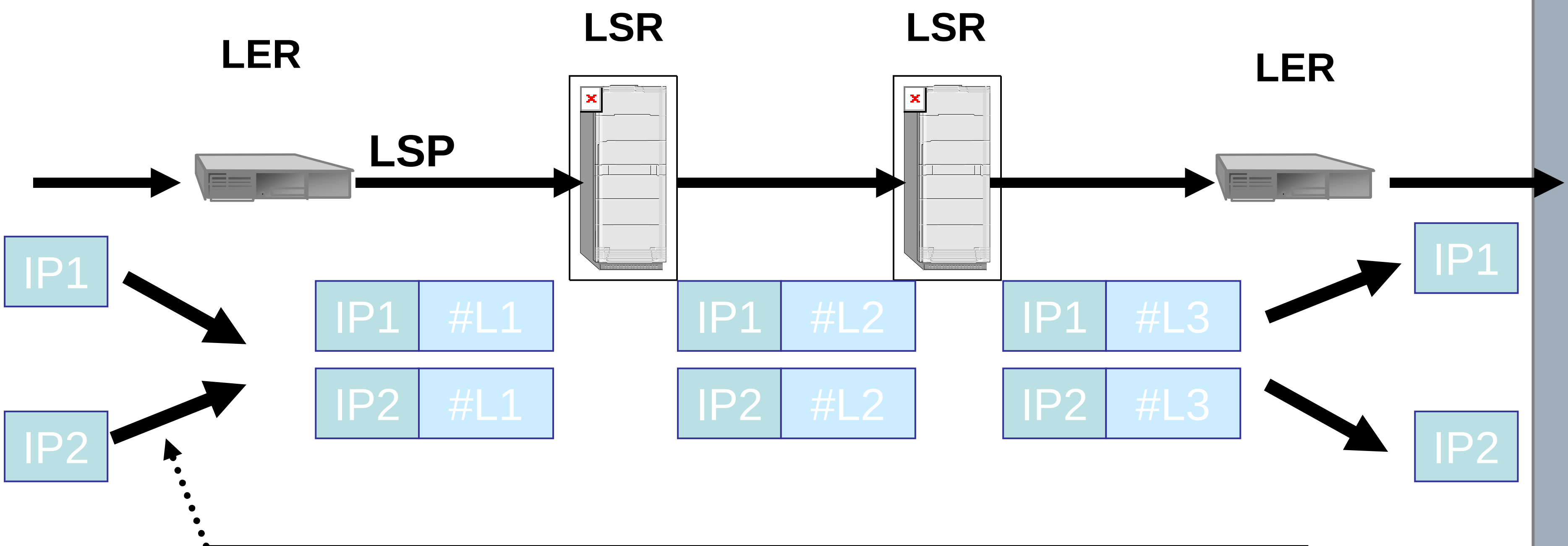
- LSPs are unidirectional
- Ingress, transit, and egress are relative to a given LSP
- A given router can be ingress, egress, and transit for different LSPs

Label Edge Routers

- Label Edge Router (LER)
  - A tunnel (LSP) endpoint
    - Ingress
    - Egress
  - Push and Pop



Forwarding Equivalence Classes

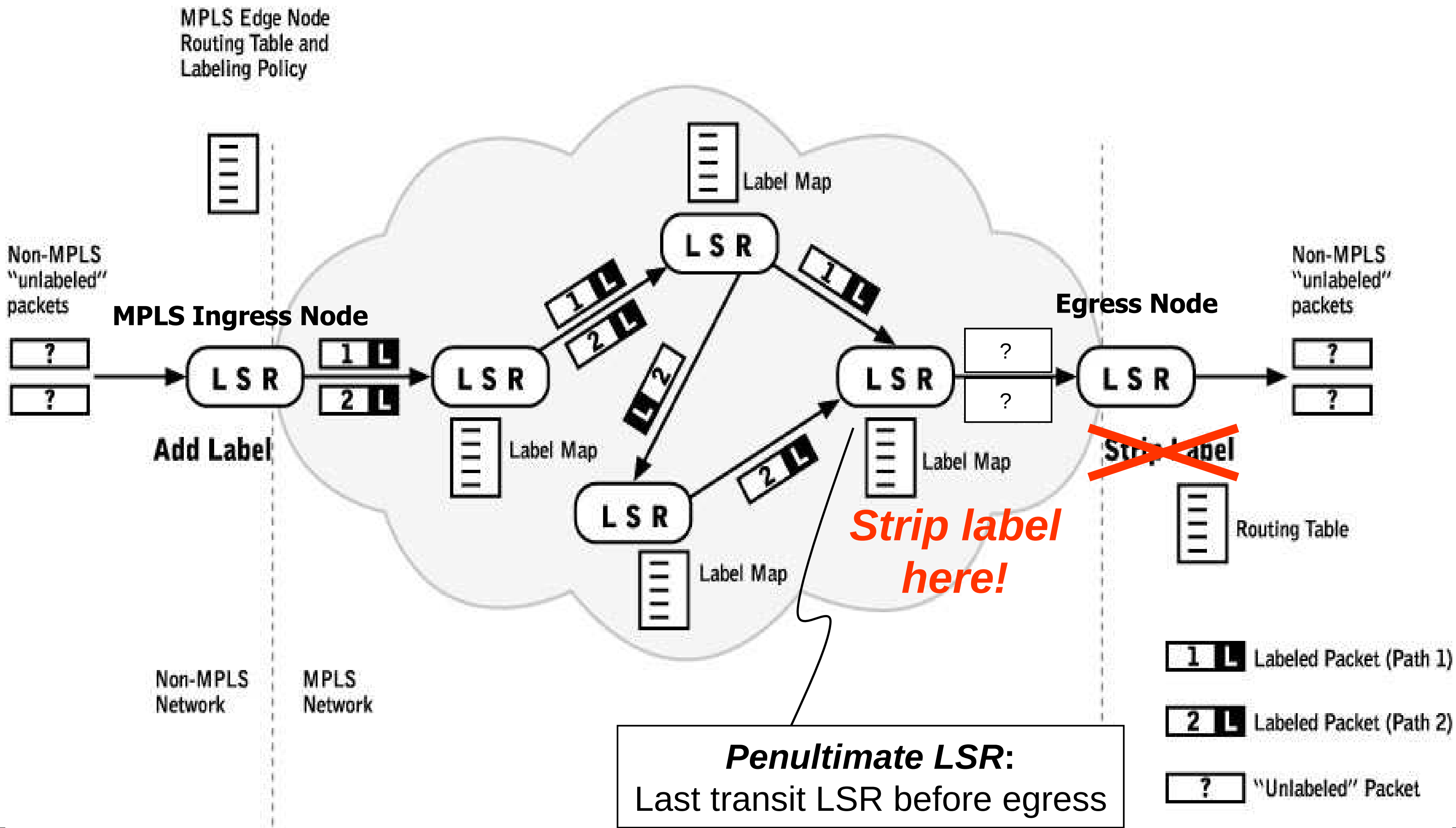


Packets are destined for different address prefixes, but can be mapped to common path

- FEC = “A subset of packets that are all treated the same way by a router”
- The concept of FECs provides for a great deal of flexibility and scalability
- In conventional routing, a packet is assigned to a FEC at each hop (i.e. L3 look-up), in MPLS it is only done once at the network ingress.

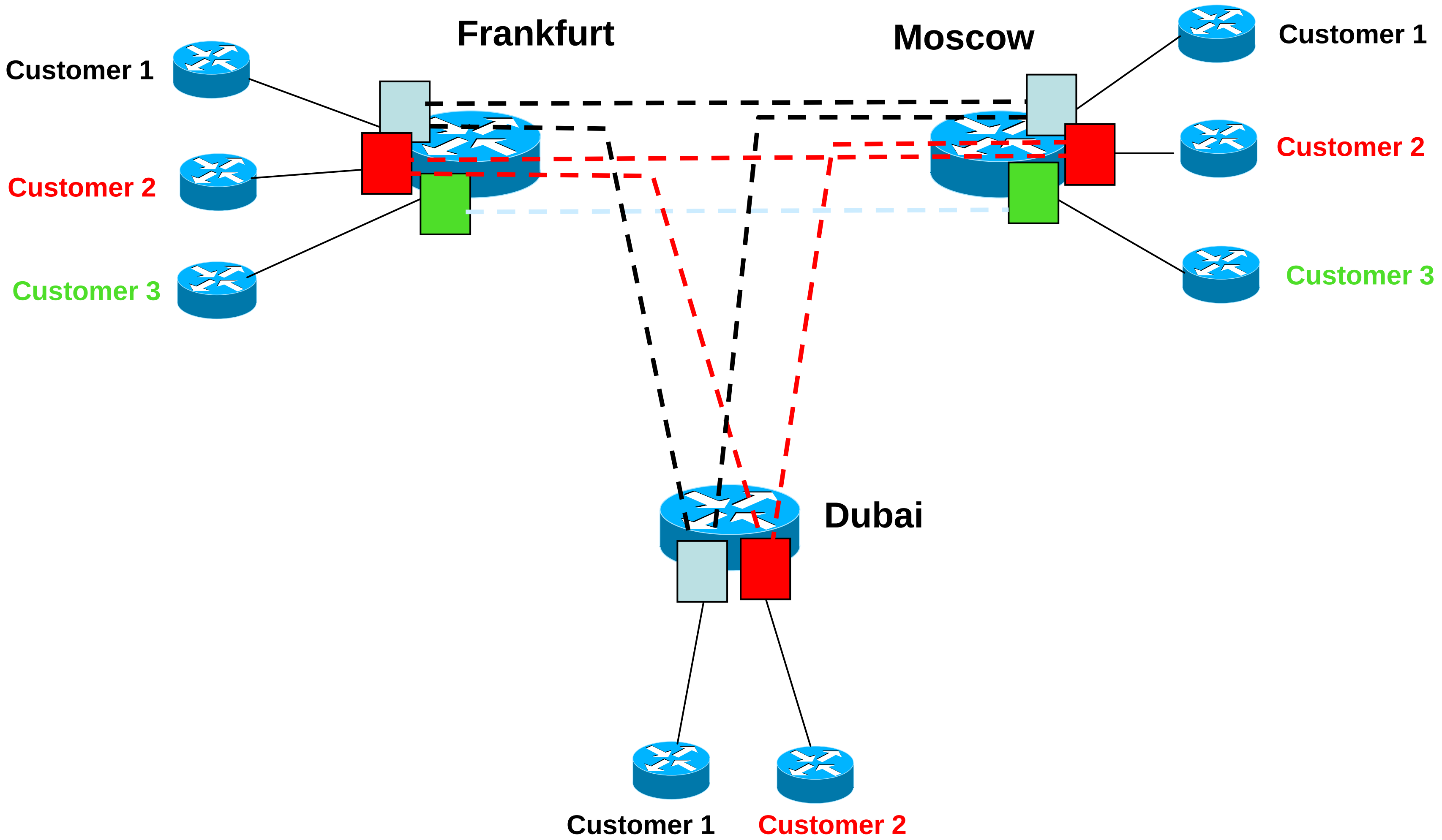
Penultimate Hop Popping

- Egress has to apply IP routing anyway
- → Can remove MPLS label one hop *before* egress



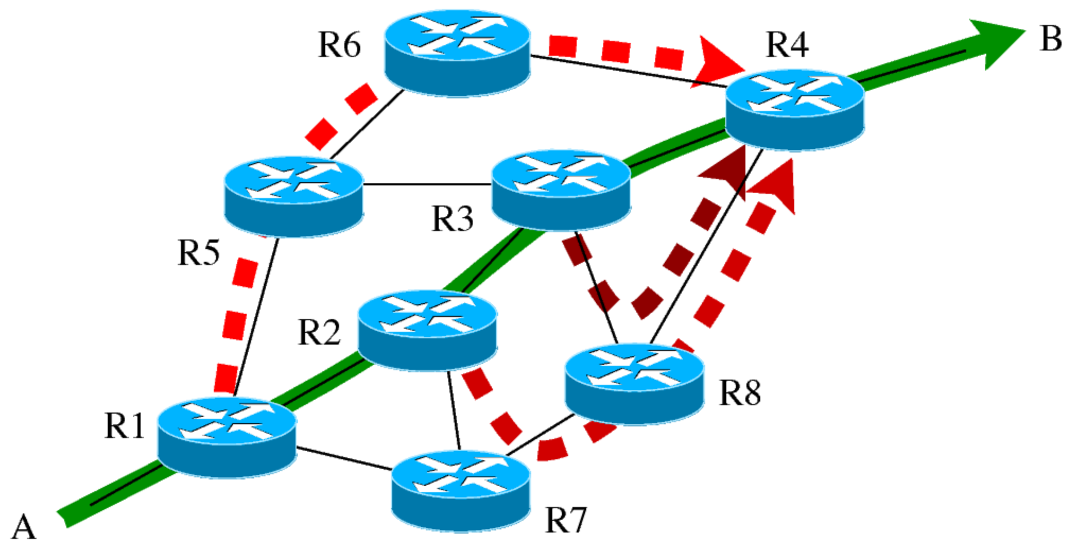
MPLS VPNs

- Virtual Private Networks: Make customers feel as if they have a direct and private connection



MPLS Fast Reroute (FRR)

- Configuration of backup paths in network
  - Many (local) backup paths for each primary LSR
- Upon detection of a failure
  - LSR immediately switches to its local backup path
  - No need to wait for signalling upstream! (in our example: R1)
- Very fast reaction speed: 50ms



Configuring labels

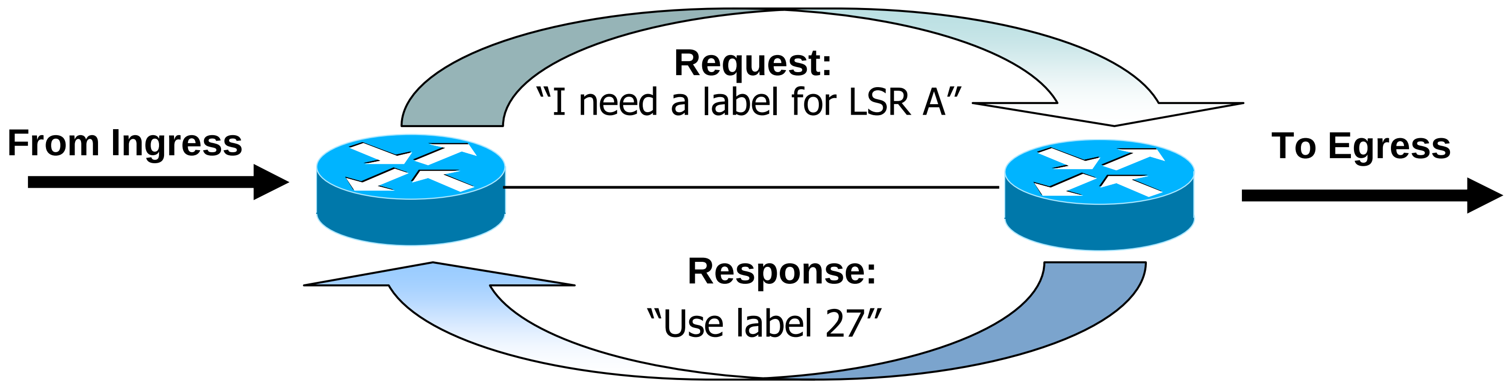
- Labels: *local* to each router
- How do routers get to know labels and their semantics?
  a) Manual configuration: does not scale
  b) Signalling: Using some label distribution protocol
     - Set of procedures by which one LSR informs another LSRs of the bindings (label/FEC) it has made

Label Assignment and Distribution

- Decision to bind a particular label L to a particular FEC F
  - made by LSR which is downstream (with respect to that binding)
- Downstream LSR informs upstream LSR of the binding
- Direction
  - Labels are ‘downstream assigned’
  - Label bindings are distributed in ‘downstream to upstream’ direction.

Label Distribution

- Requests for labels flow *downstream*
  - From Ingress to Egress (like the MPLS packets)
  - Because ingress is the LSR that establishes the LSP
- Assignment of labels (label binding) flows *upstream*
  - From Egress to Ingress
  - Because LSRs need to map *incoming* labels to some action (Push, Swap, Pop)



Label Distribution Protocols

- Label Distribution Protocol (LDP)
  - Hop-by-hop label distribution
  - Follows IGP best path: No traffic engineering capabilities
  - Highly scalable: Best suited for apps using thousands of LSPs (VPNs)
- Resource Reservation Protocol with Traffic Engineering Extensions (RSVP-TE)
  - End-to-end LSP signaling
  - Enables specification of path constraints
  - Less scalable, LSRs maintain soft state: Best suited for traffic engineering in the core

Label Distribution: RSVP

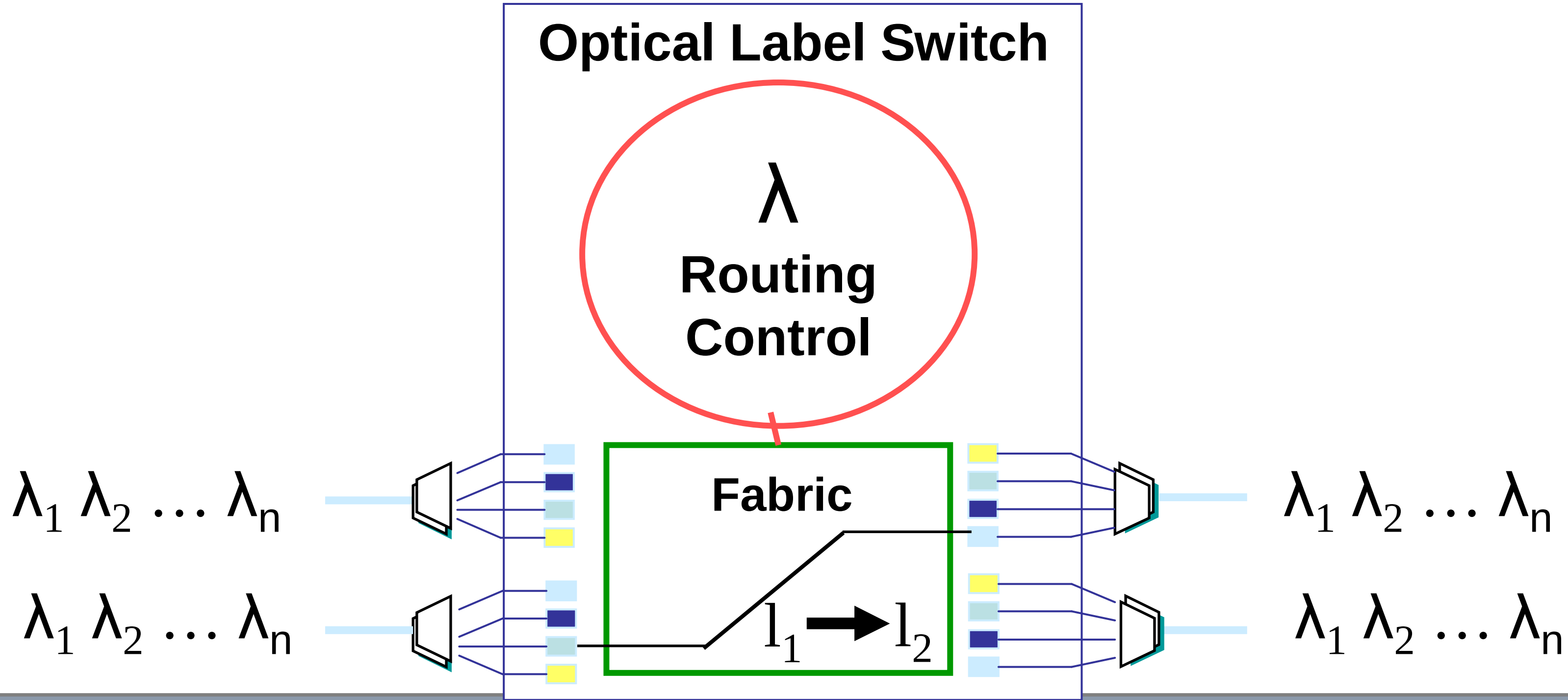
- End-to-end *constrained* path signaling
- Enabled by OSPF or IS-IS with TE extensions
  - Extended IGPs flood TE interface parameters, e.g.:
    - Maximum Reservable Bandwidth
    - Unreserved Bandwidth
    - …
- Interface parameters used to build *Traffic Engineering Database* (TED)
- *Constrained Shortest Path First* (CSPF):
  Calculates best path based on specified constraints

Label Distribution Protocols: Less used

- Constraint-Based Routed LDP (CR-LDP)
  - TE-capable LDP
  - Never widely deployed
- MP-BGP
  - Best suited for inter-AS VPNs
  - Inter-AS MPLS is a pain in the neck…

Generalized MPLS (GMPLS) for optical media

- Optical networks:
  - Switch fabric ≈ mirrors that reflect light beams
  - One glass fibre, multiple wavelenghts: λ_1 λ_2 … λ_n
  - Problem: Keep same wavelength λ_i through entire network!
- λ_i = just another label to distribute! No new protocols required.



Why not to use MPLS

- Complexity
  - MPLS + some LDP = complex
  - Intradomain IP routing + MPLS + some LDP + intelligent Link Layer = very complex
- Higher complexity means…
  - Hard to debug
  - More administration overhead, and administrators are expensive
- Inter-AS MPLS only works in theory
  - Intradomain routing + Interdomain routing + MPLS + own LDP configuration + LDP configuration of peer ASes + intelligent Link Layer + intelligent Link Layers of other ASes = unmanageable

MPLS: Summary

- Sits between IP (L3) and Link Layer (L2)
- Switching instead of routing
- Aribtrary paths in network (LSPs)
- Setup of LSPs: Label distribution protocols
- GMPLS for optical networks

THANK YOU

Master Course
Computer Networks
IN2097

Prof. Dr.-Ing. Georg Carle
Christian Grothoff, Ph.D.
Dr. Nils Kammenhuber

Chair for Network Architectures and Services
Institut für Informatik
Technische Universität München
http://www.net.in.tum.de

Chapter 5: The Data Link Layer

Goals:

- understand principles behind data link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - reliable data transfer, flow control: *c.f. transport layer*
- instantiation and implementation of various link layer technologies

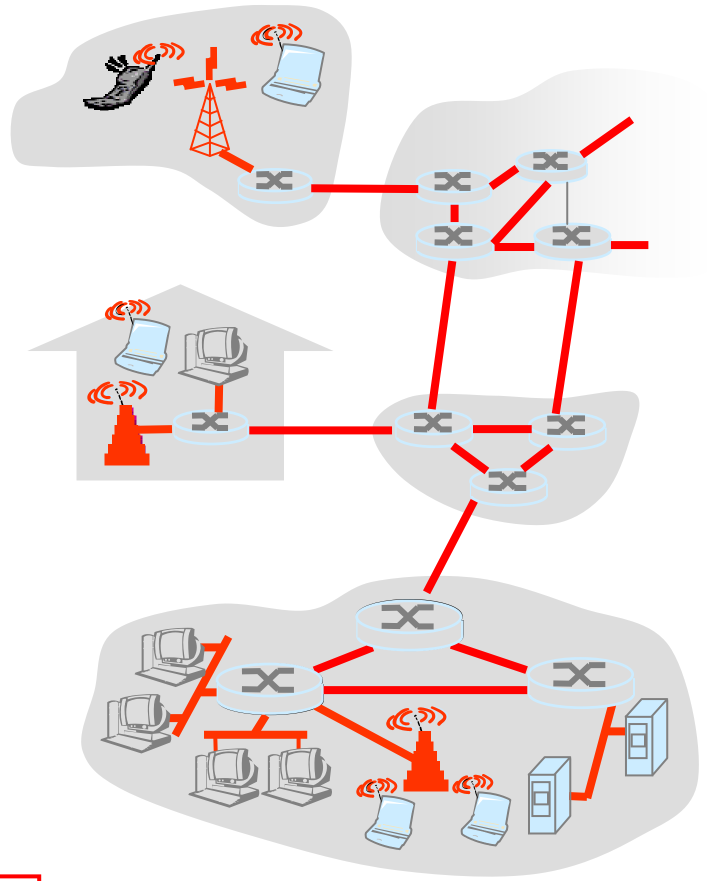
Link Layer

- 5.1 Introduction and services
- 5.2 Multiple access protocols
- 5.3 Link-layer Addressing
- 5.4 Ethernet
- 5.5 Link-layer switches

Link Layer: Introduction

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 Services

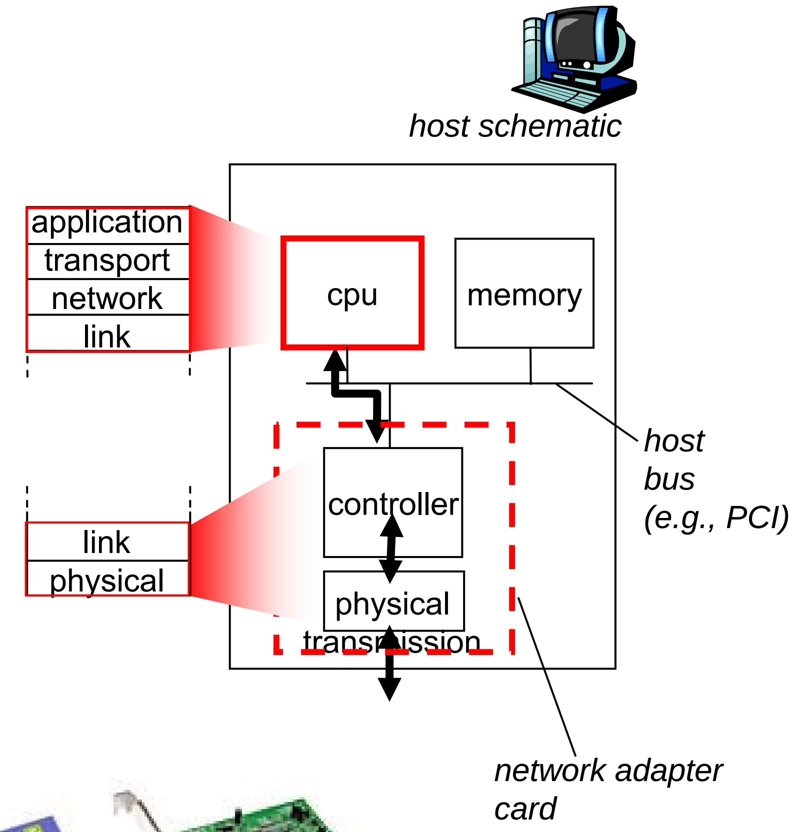
- framing, link access:
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - “MAC” addresses used in frame headers to identify source, destination
    - different from IP address!
- reliable delivery between adjacent nodes
  - stateful protocol needed to do this already (c.f. chapter 3)
  - seldom used on low bit-error link (fiber, some twisted pair)
  - wireless links: high error rates
    - Q: why both link-level and end-end reliability?

Link Layer Services (more)

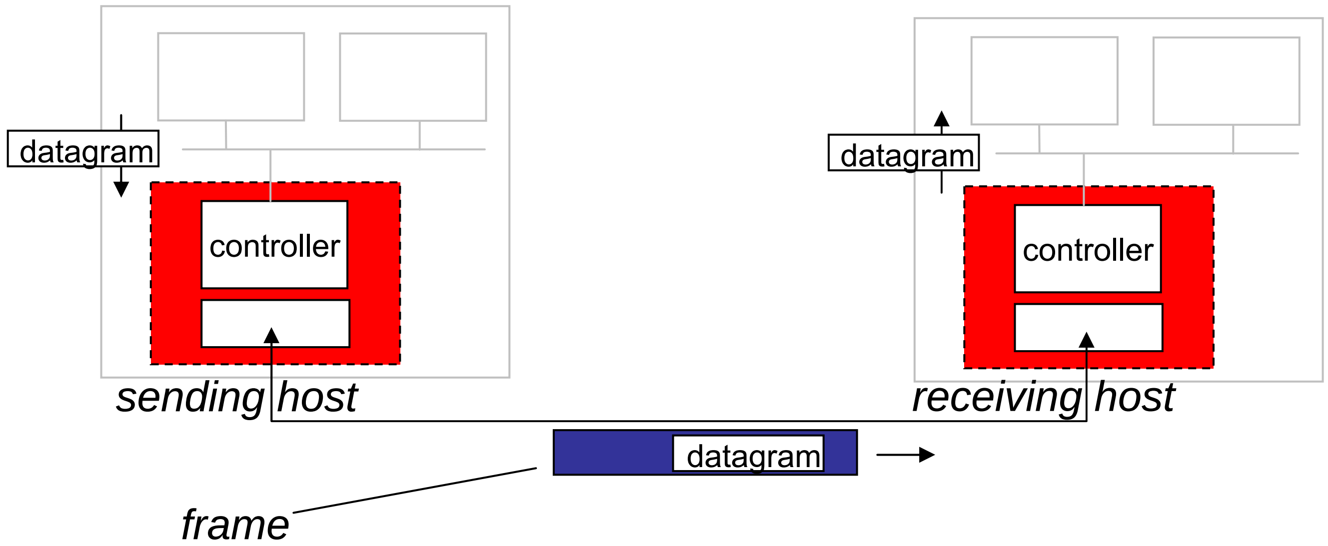
- *flow control:*
  - pacing between adjacent sending and receiving nodes
- *error detection*:
  - errors caused by signal attenuation, noise.
  - receiver detects presence of errors:
    - signals sender for retransmission or drops frame
- error correction:
  - receiver identifies *and corrects* bit 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, 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, reliable data transfer (rdt), flow control, etc.
- ❑ receiving side
  - looks for errors, rdt, flow control, etc
  - extracts datagram, passes to upper layer at receiving side

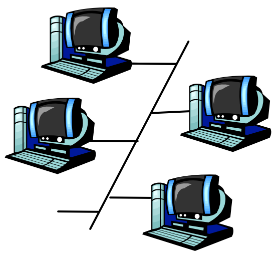
Link Layer

- 5.1 Introduction and services
- 5.2 Multiple access protocols
- 5.3 Link-layer Addressing
- 5.4 Ethernet
- 5.5 Link-layer switches

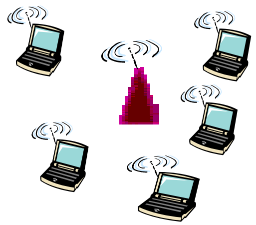
Multiple Access Links and Protocols

Two types of “links”:

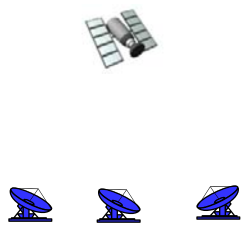
- point-to-point
  - PPP for dial-up access
  - point-to-point link between Ethernet switch and host
- broadcast (shared wire or medium)
  - old-fashioned (coax) Ethernet
  - upstream HFC (Hybrid Fiber Coax)
  - 802.11 wireless LAN



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

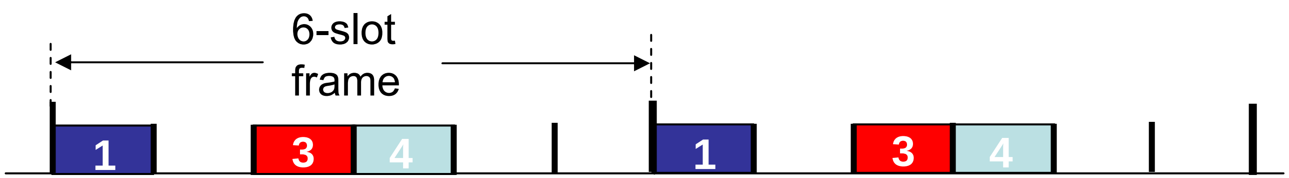
Three broad classes:

- Channel Partitioning
  - divide channel into smaller “pieces” (time slots, frequency, code)
  - allocate piece to node for exclusive use
- Random Access
  - channel not divided, allow collisions
  - “recover” from collisions
- “Taking turns”
  - nodes take turns, but nodes with more to send can take longer turns

Channel Partitioning MAC protocols: TDMA

TDMA: time division multiple access

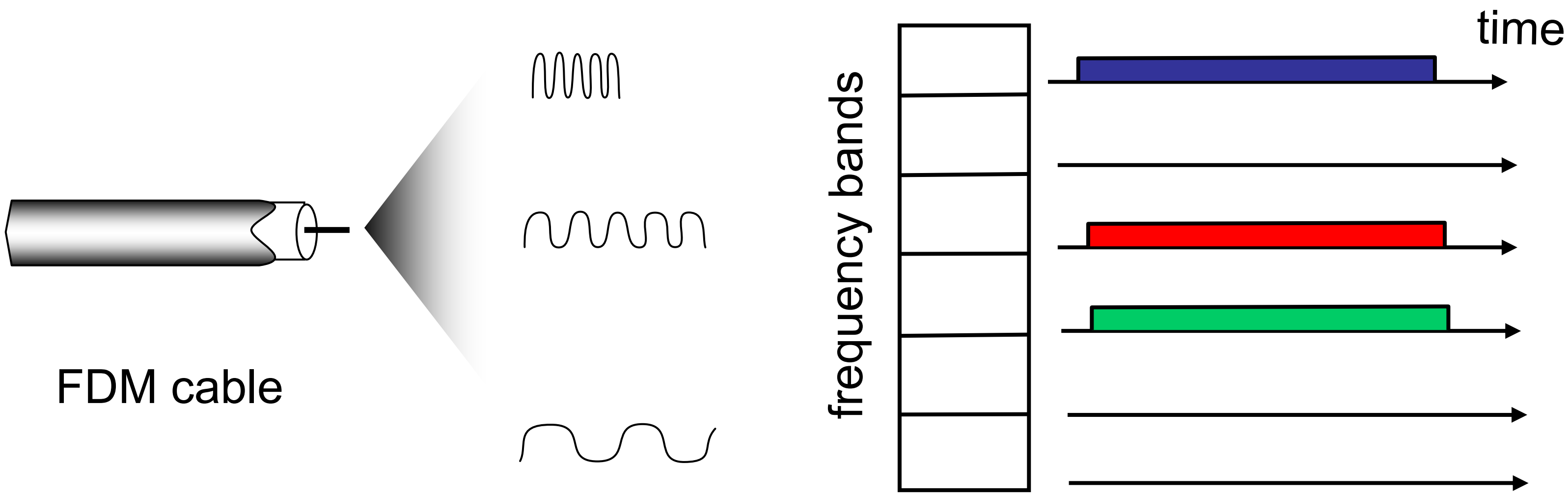
- access to channel in "rounds"
- each station gets fixed length slot (length = packet transmission time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle



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 packet, frequency bands 2,5,6 idle



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:
  - CSMA, CSMA/CD, CSMA/CA

CSMA (Carrier Sense Multiple Access)

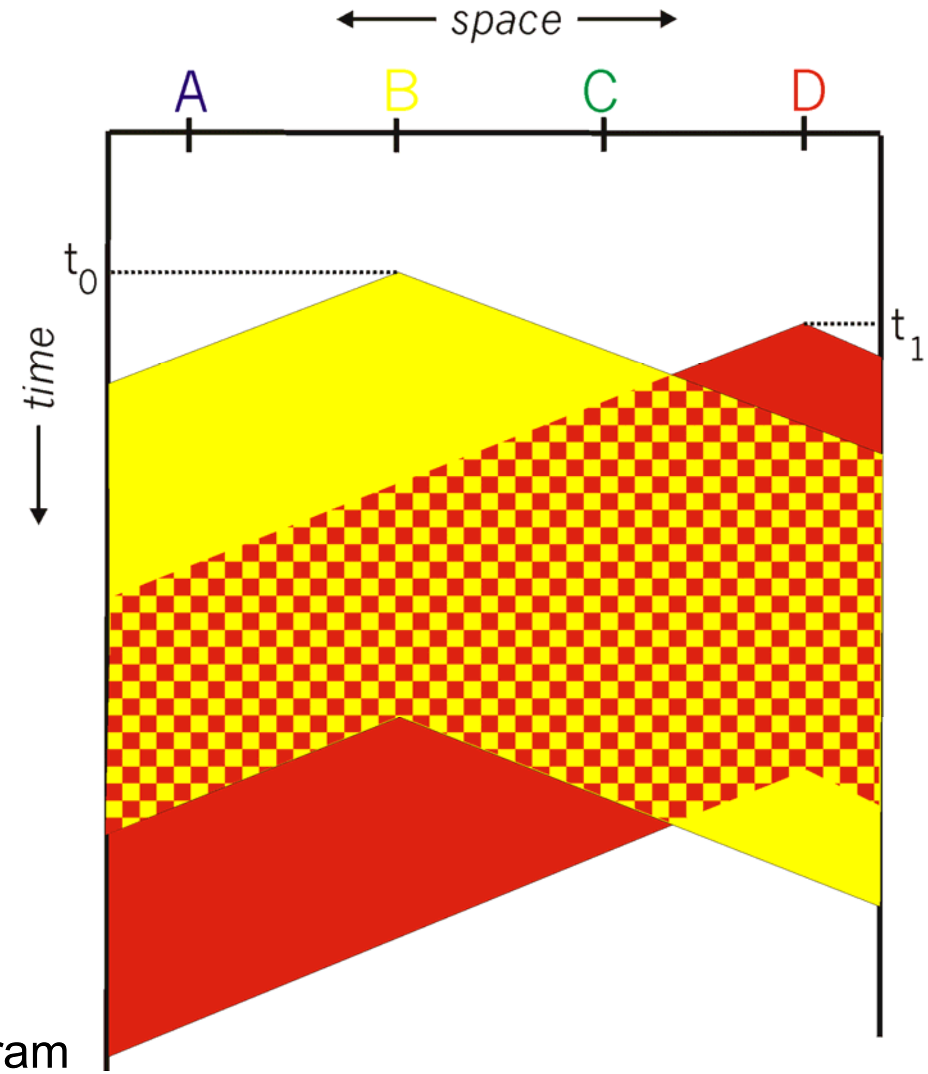
CSMA: listen before transmit:

- If channel sensed idle: transmit entire frame
- If channel sensed busy, defer transmission

- human analogy: don't interrupt others!

CSMA collisions

spatial layout of nodes



space-time-diagram

collisions *can* still occur:

propagation delay means two nodes may not hear each other's transmission

collision:

entire packet transmission time wasted

note:

role of distance & propagation delay in determining collision probability

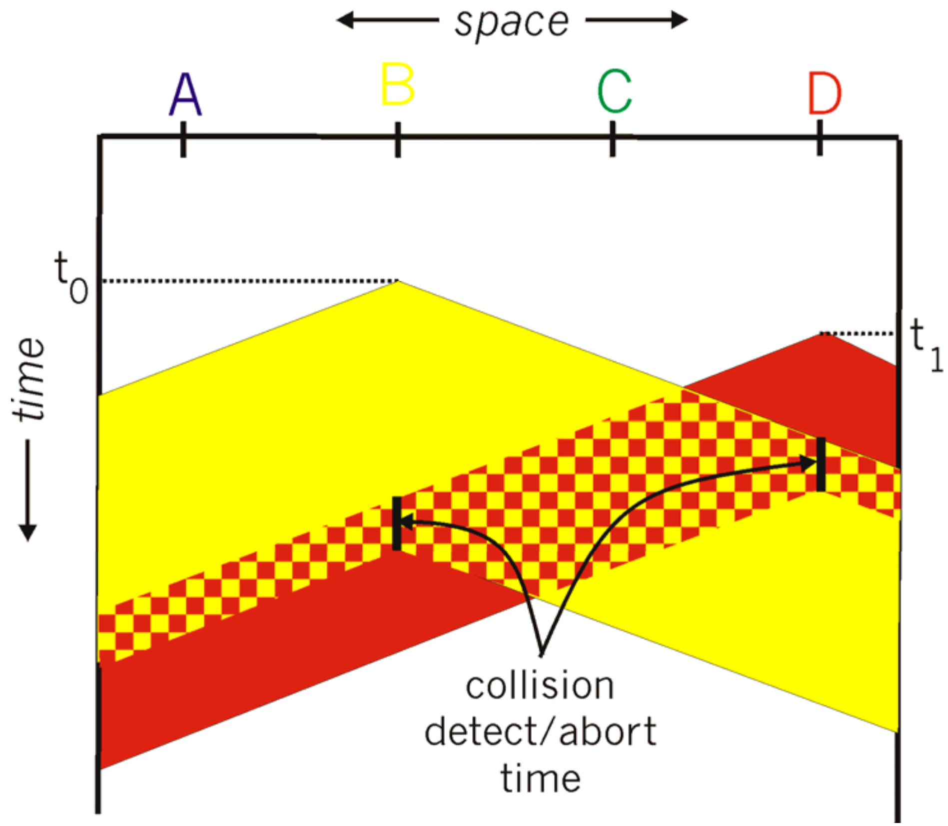
CSMA/CD (Collision Detection)

CSMA/CD: carrier sensing, deferral as in CSMA

- 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: received signal strength overwhelmed by local transmission strength
- human analogy: the polite conversationalist

CSMA/CD collision detection



“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

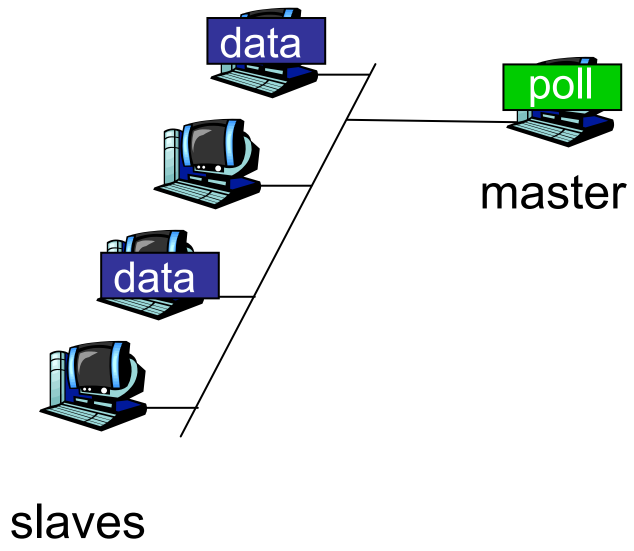
“taking turns” protocols

look for best of both worlds!

“Taking Turns” MAC protocols

Polling:

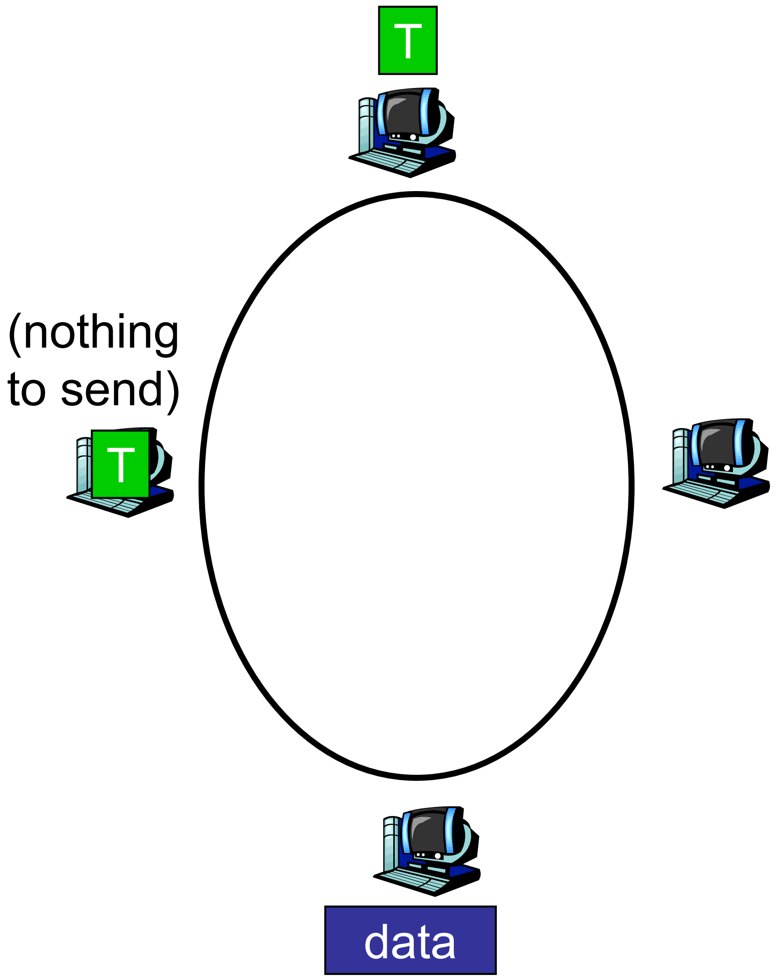
- master node “invites” slave nodes to transmit in turn
- typically used with “dumb” slave devices
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)



“Taking Turns” MAC protocols

Token passing:

- control **token** passed from one node to next sequentially.
- token message
- concerns:
  - token overhead
  - latency
  - single point of failure (token)



Summary of MAC protocols

- *channel partitioning,* by time, frequency or code
  - Time Division, Frequency Division
- *random access* (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 central site, token passing
  - FDDI, Token Ring

Link Layer

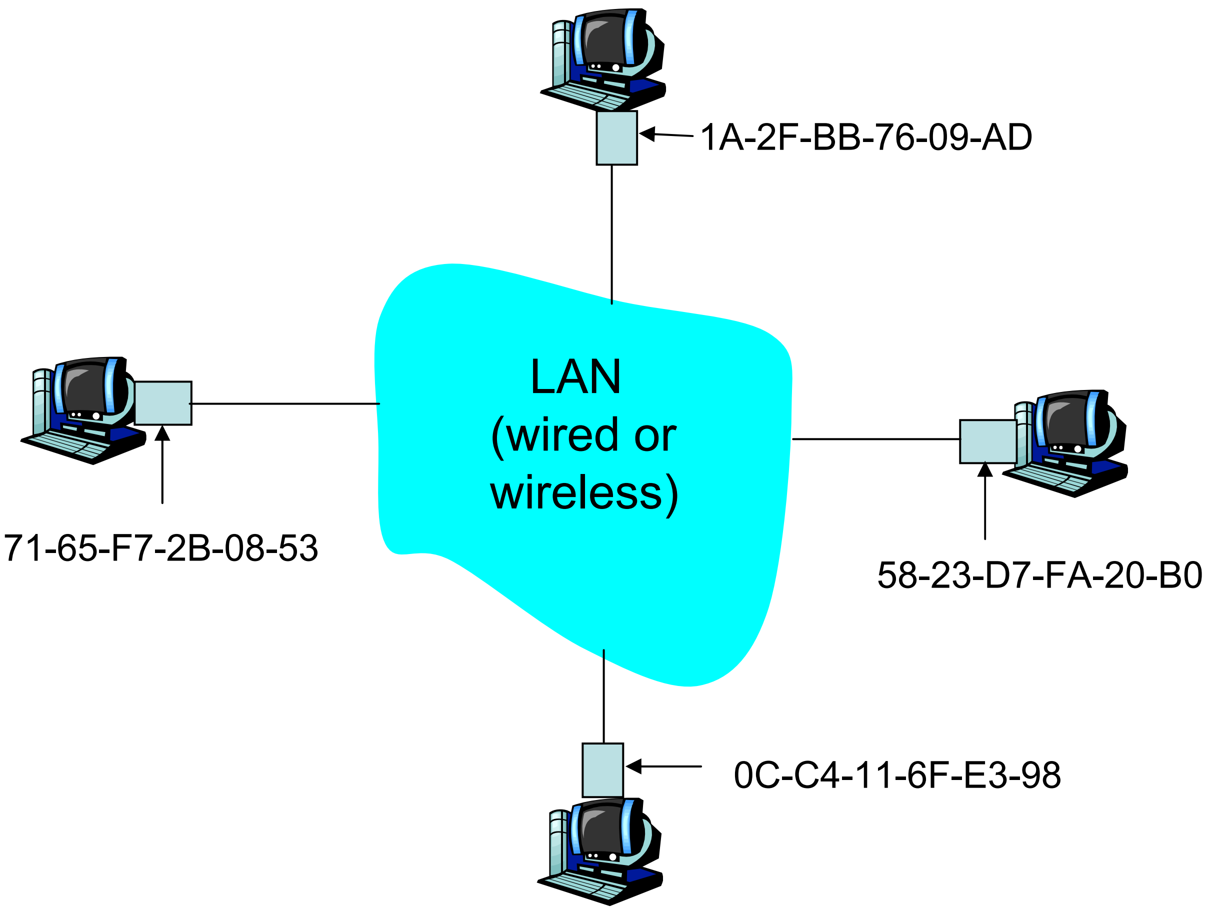
- 5.1 Introduction and services
- 5.2 Multiple access protocols
- 5.3 Link-layer Addressing
- 5.4 Ethernet
- 5.5 Link-layer switches

MAC Addresses and ARP

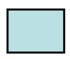
- 32-bit IP address:
  - *network-layer* address
  - used to get datagram to destination IP subnet
- MAC (or LAN or physical or Ethernet) address:
  - function: *get frame from one interface to another physically-connected interface (same network)*
  - 48 bit MAC address (for most LANs)
    - burned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

LAN Addresses and ARP

Each adapter on LAN has unique LAN address



Broadcast address = FF-FF-FF-FF-FF-FF

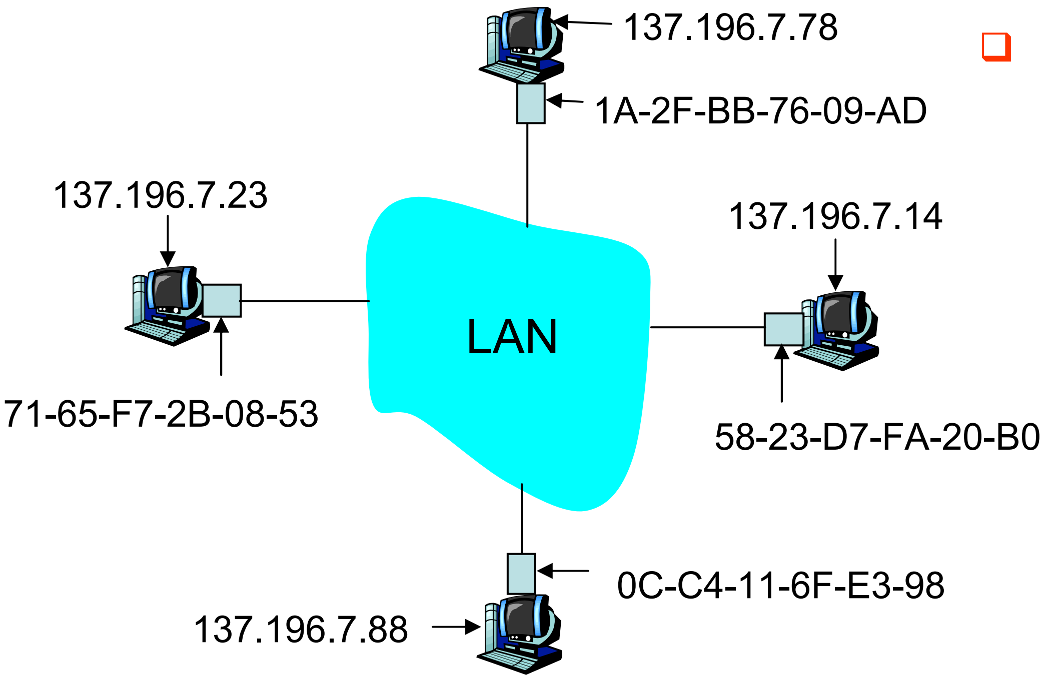


LAN Address (more)

- 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 flat address ➜ portability
  - can move LAN card from one LAN to another
- IP hierarchical address NOT portable
  - address depends on IP subnet to which node is attached

ARP: Address Resolution Protocol

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



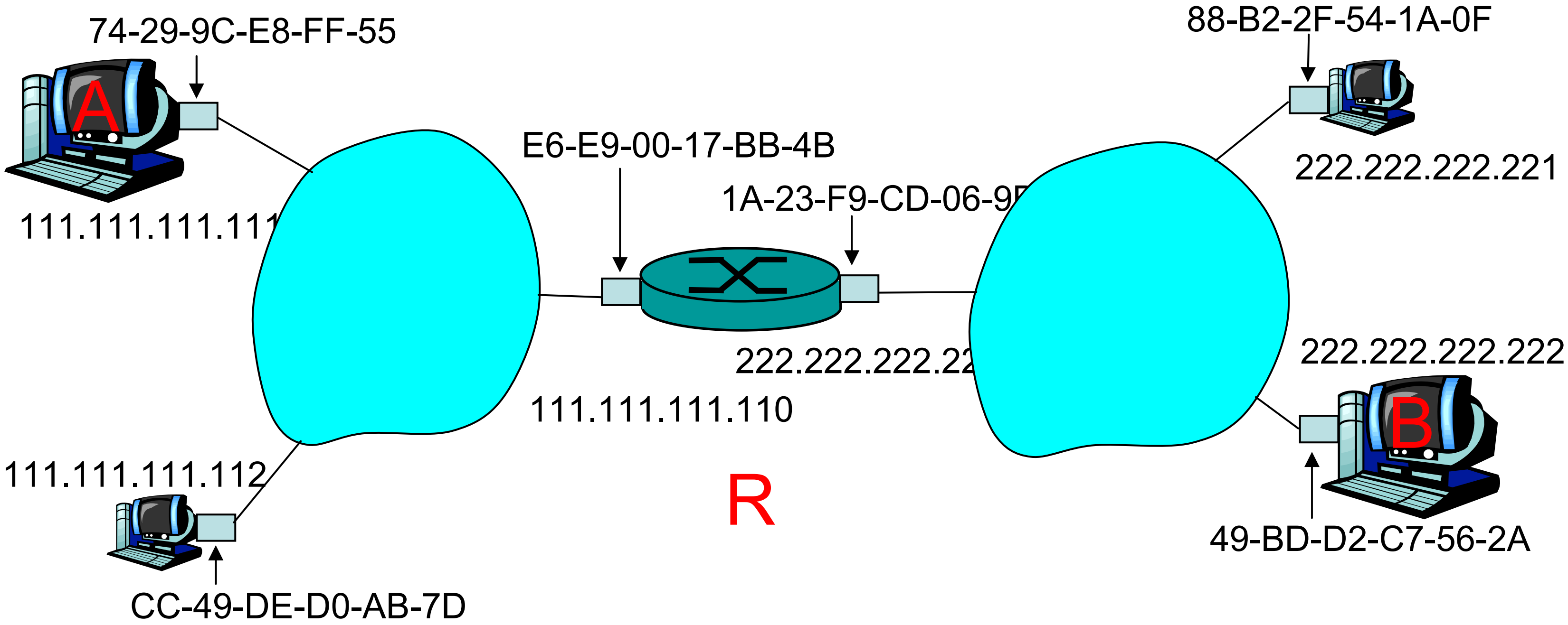
- Each IP node (host, router) on LAN has ARP (Address Resolution Protocol) table
- ARP table: IP/MAC address mappings for some LAN nodes
- <IP address; MAC address; TTL>
  - TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

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

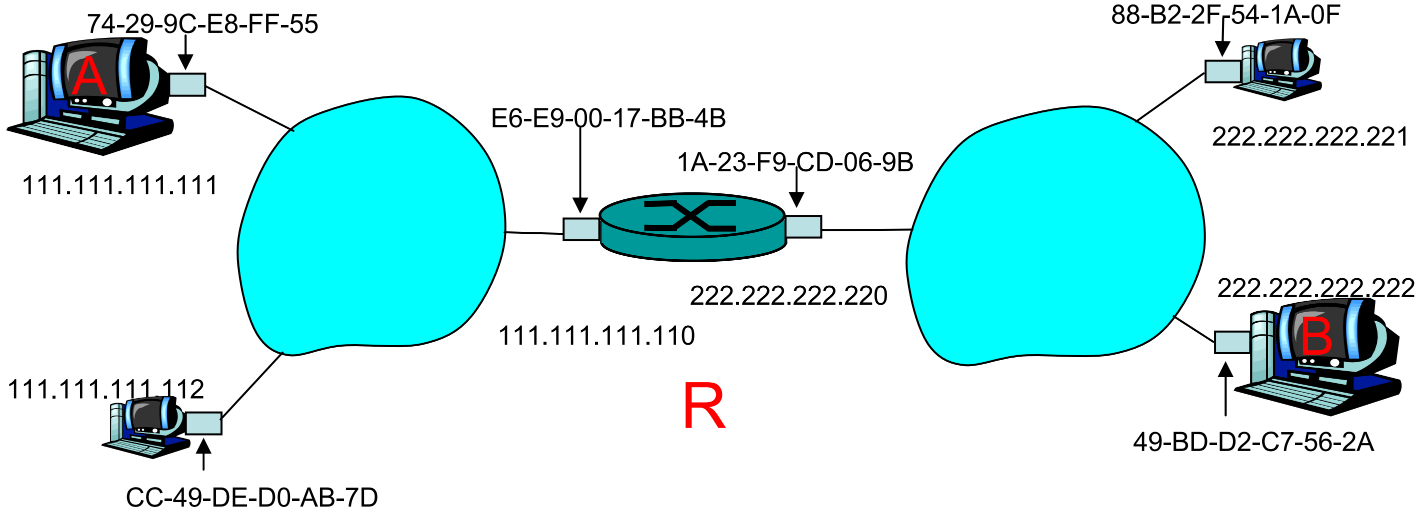
- walkthrough: send datagram from A to B via R
  assume A knows B's IP address



- two ARP tables in router R, one for each IP network (LAN)

Addressing: routing to another LAN (2)

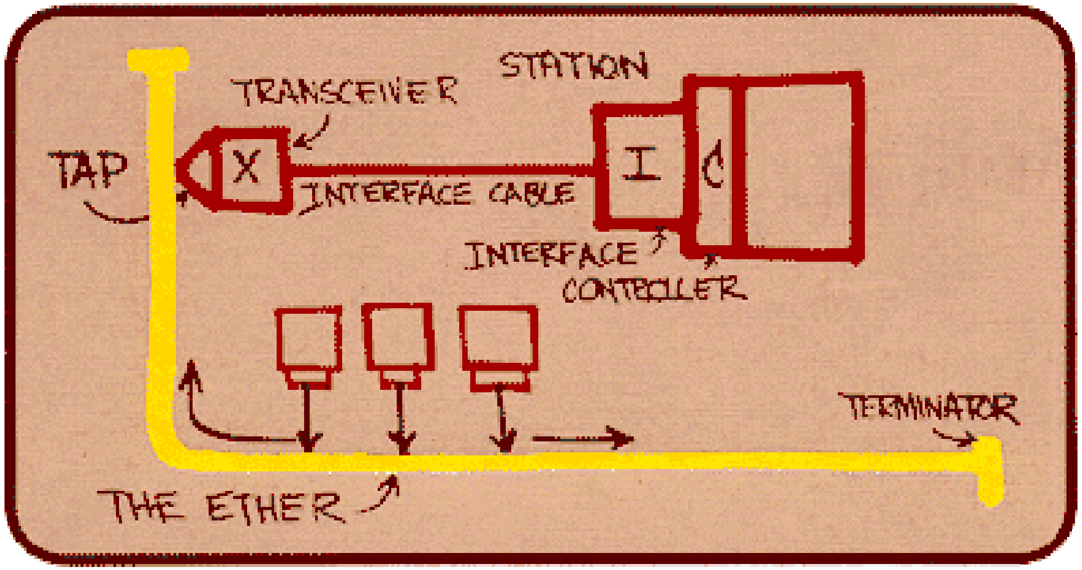
- A creates IP datagram with source A, destination B
- A checks its forwarding table on to which next hop to send datagram
- A uses ARP to get R's MAC address for 111.111.111.110
- A creates link-layer frame with dest MAC address of R, frame contains A-to-B IP datagram
- A's NIC sends frame
- R's NIC receives frame
- R extracts 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



- 5.1 Introduction and services
- 5.2 Multiple access protocols
- 5.3 Link-layer Addressing
- 5.4 Ethernet
- 5.5 Link-layer switches

Ethernet

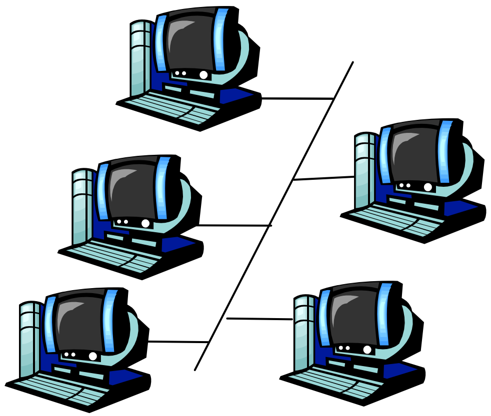
- "dominant" wired LAN technology:
- cheap: $/€ 20 for NIC
- first widely used LAN technology
- simpler, cheaper than token LANs and ATM
- kept up with speed race: 10 Mbps – 10 Gbps



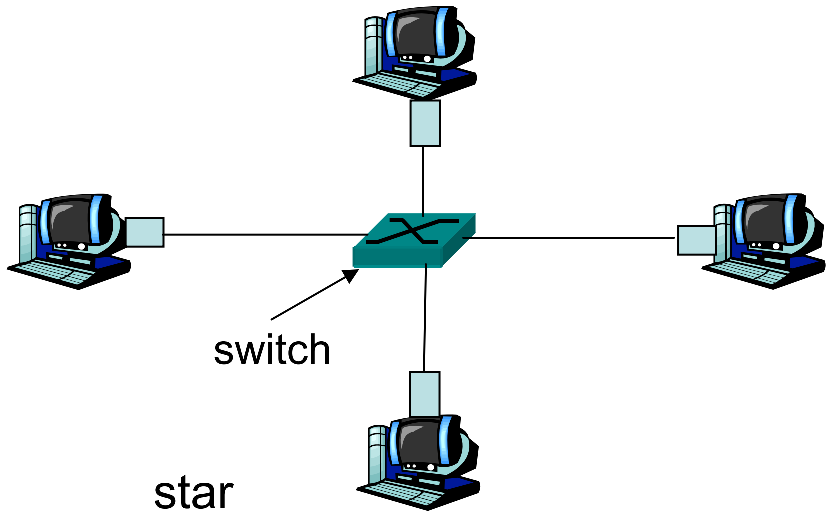
Metcalfe's Ethernet sketch

Star topology

- bus topology 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)



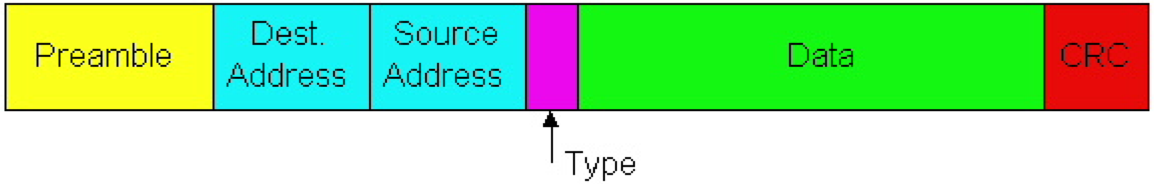
bus: coaxial cable



star

Ethernet Frame Structure

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

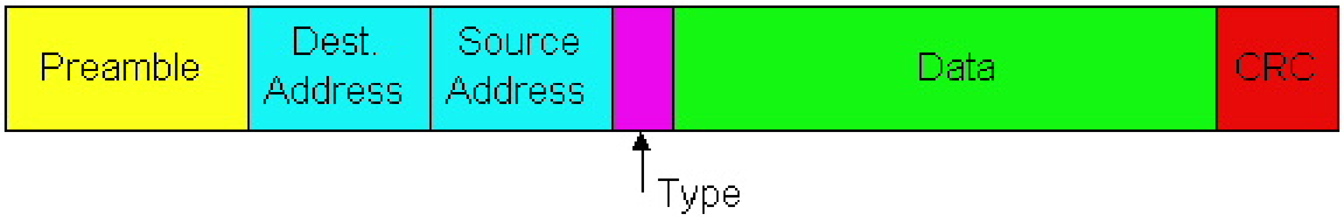


Preamble:

- 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 network layer protocol
  - otherwise, adapter discards frame
- **Type**: indicates higher layer protocol (mostly IP but others possible, e.g., Novell IPX, AppleTalk)
- **CRC**: checked at receiver, if error is detected, frame is dropped



Ethernet: Unreliable, connectionless

- connectionless:
  no handshaking between sending and receiving NICs
- unreliable:
  receiving NIC doesn't send acks or nacks to sending NIC
  - stream of datagrams passed to network layer can have gaps (missing datagrams)
  - gaps will be filled if app is using TCP
  - otherwise, app will see gaps
- Ethernet's MAC protocol: unslotted CSMA/CD

Ethernet CSMA/CD algorithm

1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel idle, starts frame transmission
   If NIC senses channel busy, waits until channel idle, then transmits
3. If NIC transmits entire frame without detecting another transmission, NIC is done with frame
4. If NIC detects another transmission while transmitting, aborts and sends jam signal
5. After aborting, NIC enters **exponential backoff**: after *m*th collision, NIC chooses *K* at random from $\{0,1,2,\ldots,2^m-1\}$. NIC waits $K\cdot512$ bit times, returns to Step 2

Ethernet's CSMA/CD (more)

Jam Signal: make sure all other transmitters are aware of collision; 48 bits

Bit time: 0.1 microsec for 10 Mbps Ethernet ;
for K=1023, wait time is about 50 msec

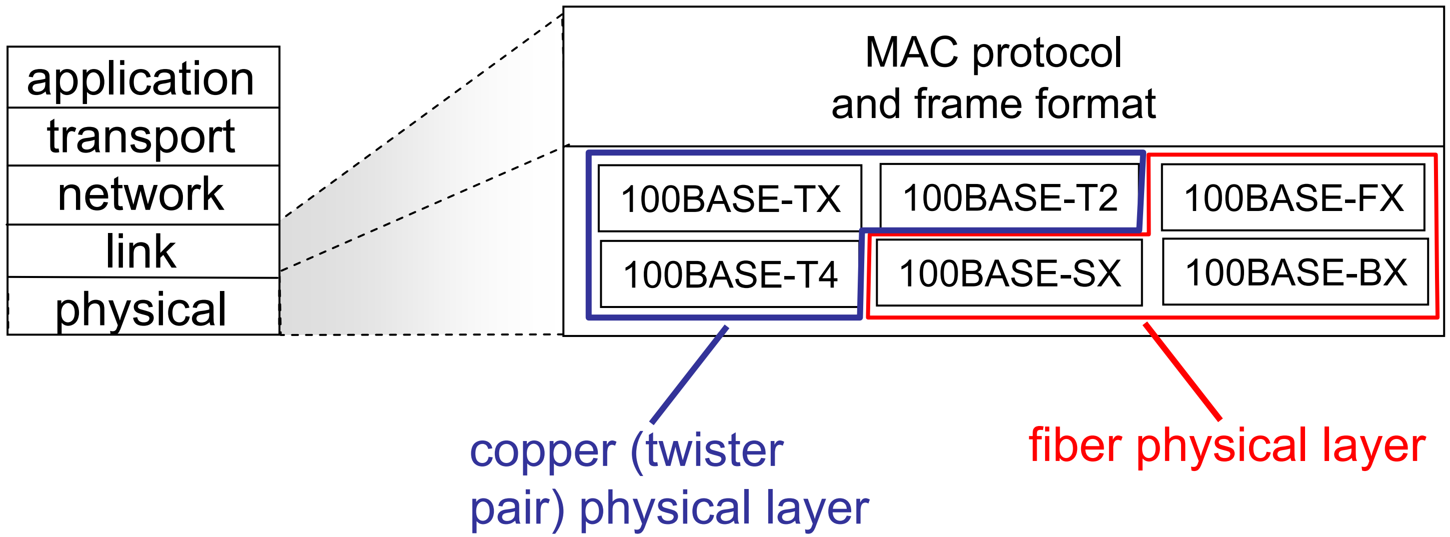
Exponential Backoff:

- *Goal*: adapt retransmission attempts to estimated current load
  - heavy load: random wait will be longer
- first collision: choose K from {0,1}; delay is K· 512 bit transmission times
- after second collision: choose K from {0,1,2,3}…
- after ten collisions, choose K from {0,1,2,3,4,…,1023}

See/interact with Java applet on AW Web site:
http://wps.aw.com/aw_kurose_network_5/
⇨student resources - highly recommended !

802.3 Ethernet Standards: Link & Physical Layers

- many different Ethernet standards
  - common MAC protocol and frame format
  - different speeds: 2 Mbps, 10 Mbps, 100 Mbps, 1Gbps, 10G bps
  - different physical layer media: fiber, cable

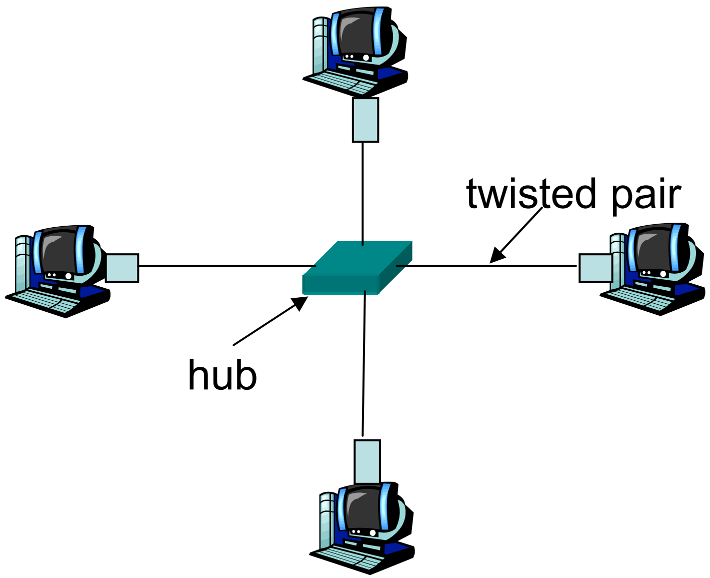


Link Layer

- 5.1 Introduction and services
- 5.2 Multiple access protocols
- 5.3 Link-layer Addressing
- 5.4 Ethernet
- 5.5 Link-layer switches

Hubs

- … physical-layer ("dumb") repeaters:
  - bits coming in one link go out all other links at same rate
  - all nodes connected to hub can collide with one another
  - no frame buffering
  - no CSMA/CD at hub: host NICs detect collisions

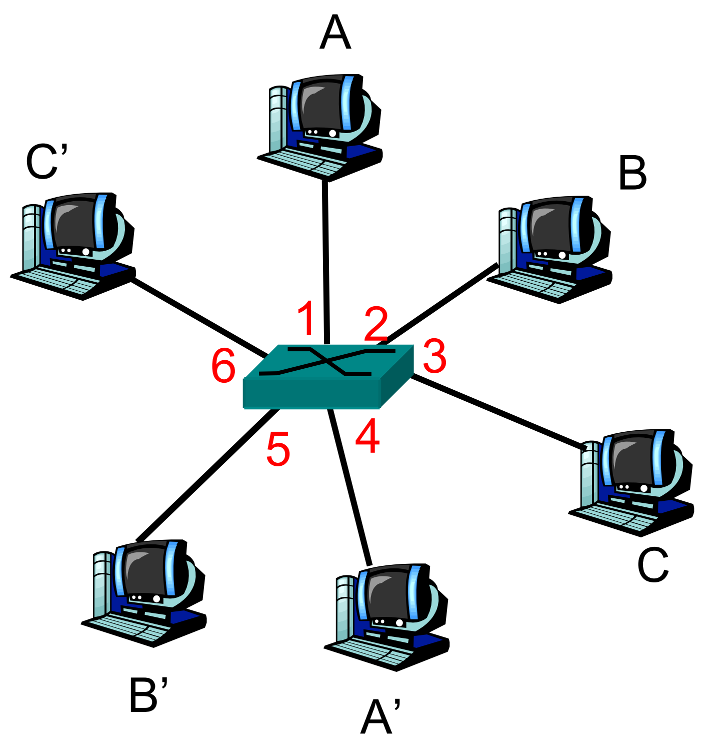


Switch

- 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

Switch: allows multiple simultaneous transmissions

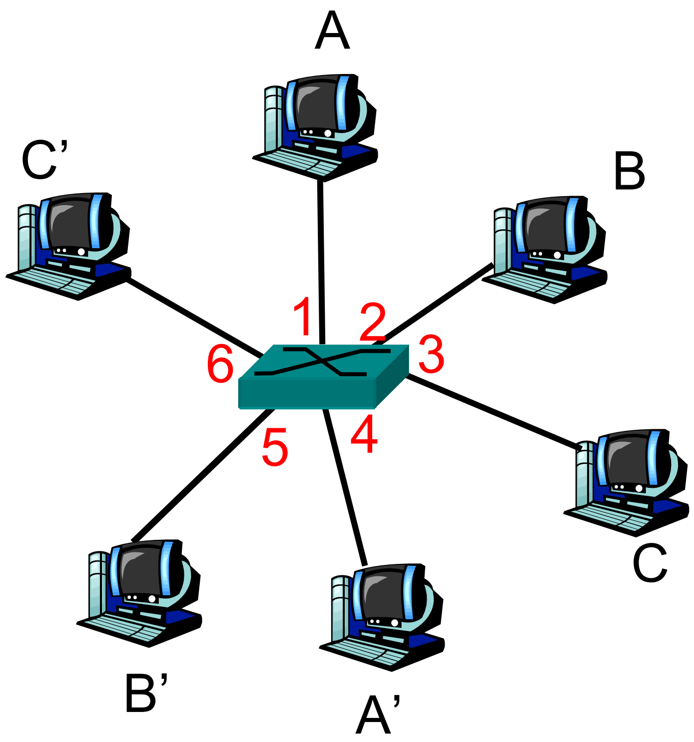
- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, but no collisions; full duplex
  - each link is its own collision domain
- *switching:* A-to-A' and B-to-B' simultaneously, without collisions
  - not possible with dumb hub



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

Switch Table

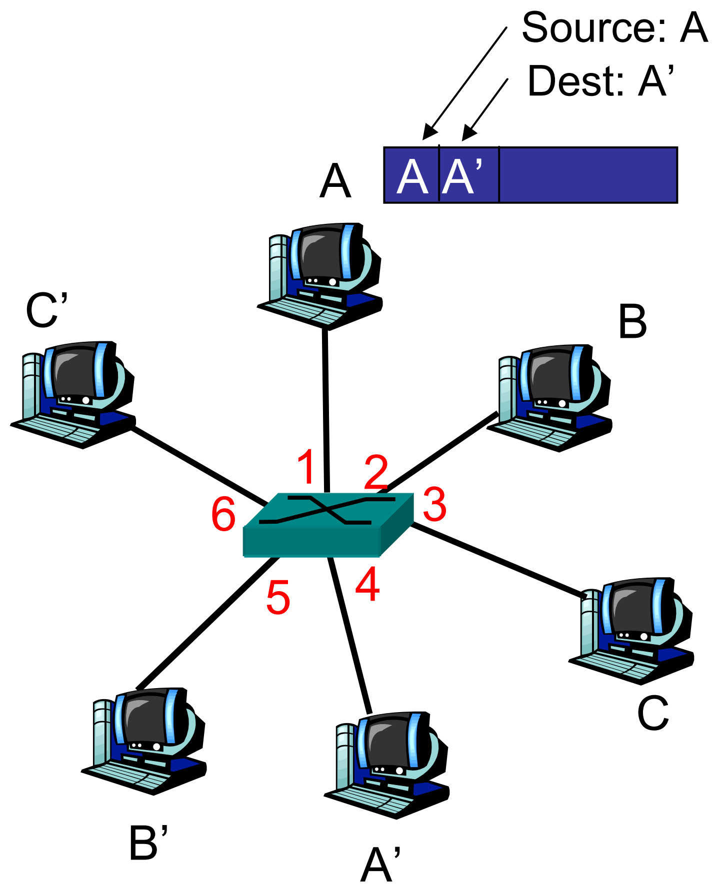
- *Q:* how does switch know that A' reachable via interface 4, B' reachable via interface 5?
- *A:* each switch has a switch table, each entry:
  - (MAC address of host, interface to reach host, time stamp)
- looks like a routing table!
- *Q:* how are entries created, maintained in switch table?
  -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



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

Switch: self-learning

- 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 destination address
3. **if** entry found for destination

    then {

    if destination on segment from which frame arrived

        then drop the frame

        else forward the frame on interface indicated

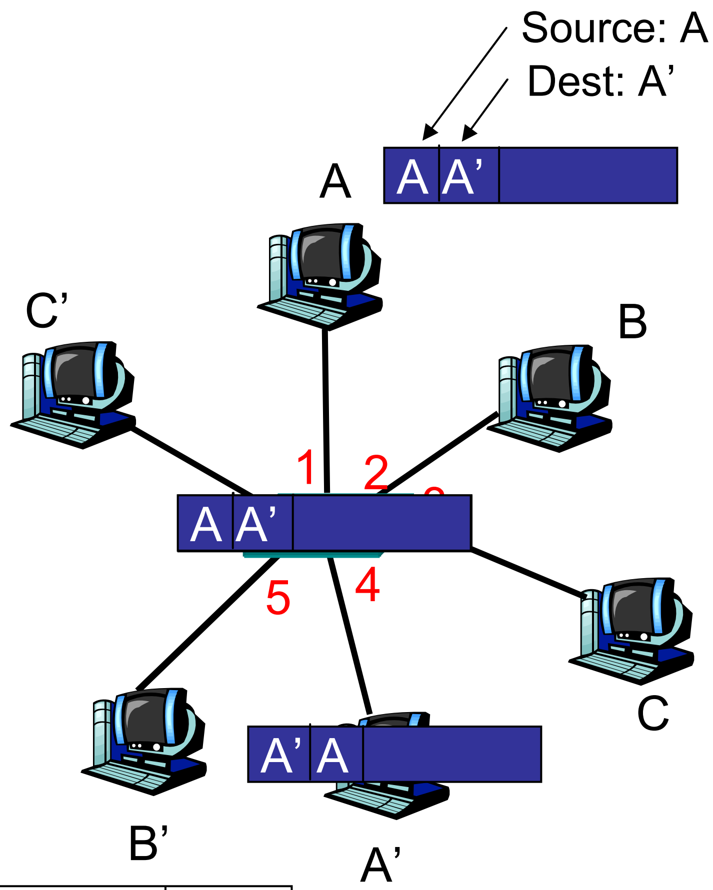
    }

    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*

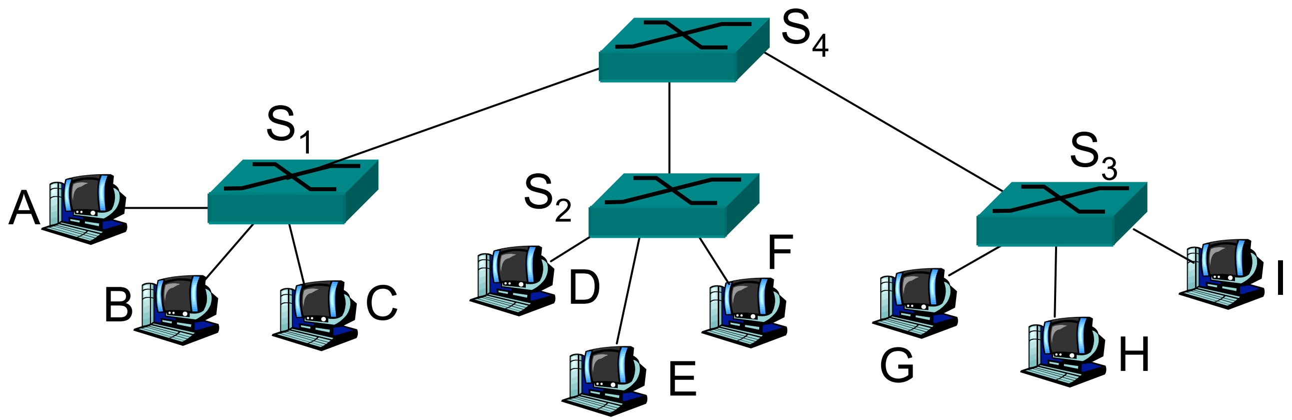


MAC addr	interface	TTL
A	*1*	*60*
A'	*4*	*60*

Switch table (initially empty)

Interconnecting switches

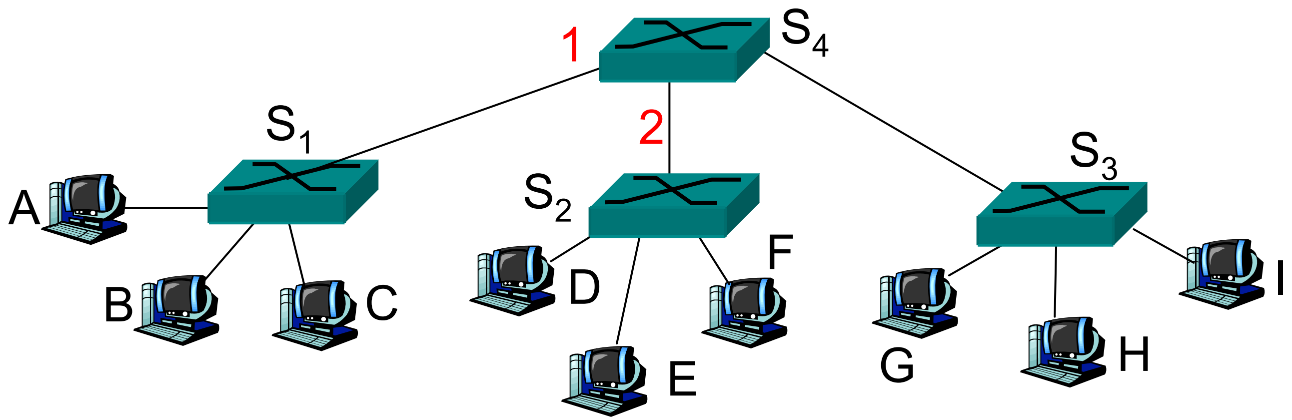
- switches can be connected together



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

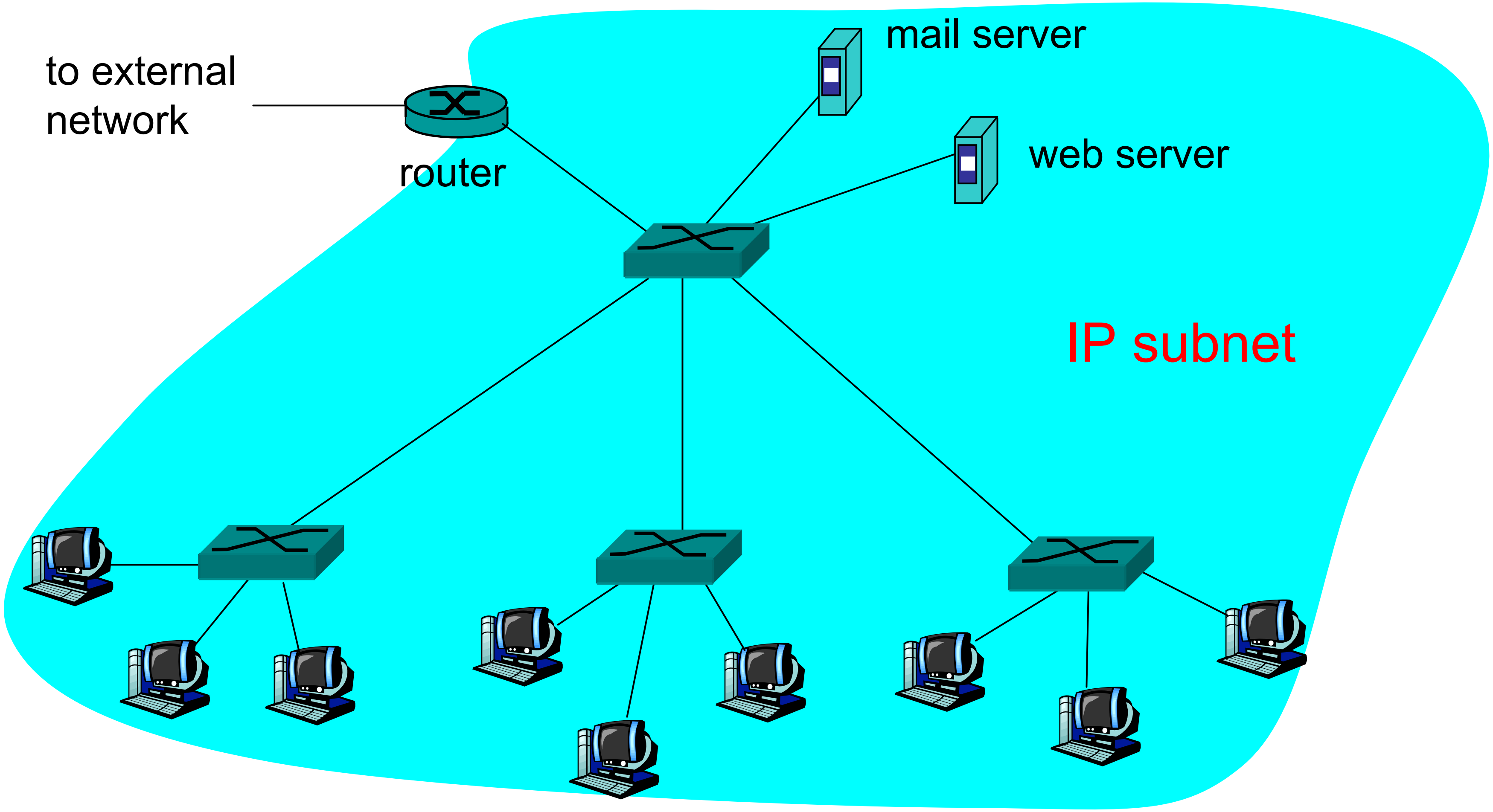
Self-learning multi-switch example

- Suppose C sends frame to I, I responds to C



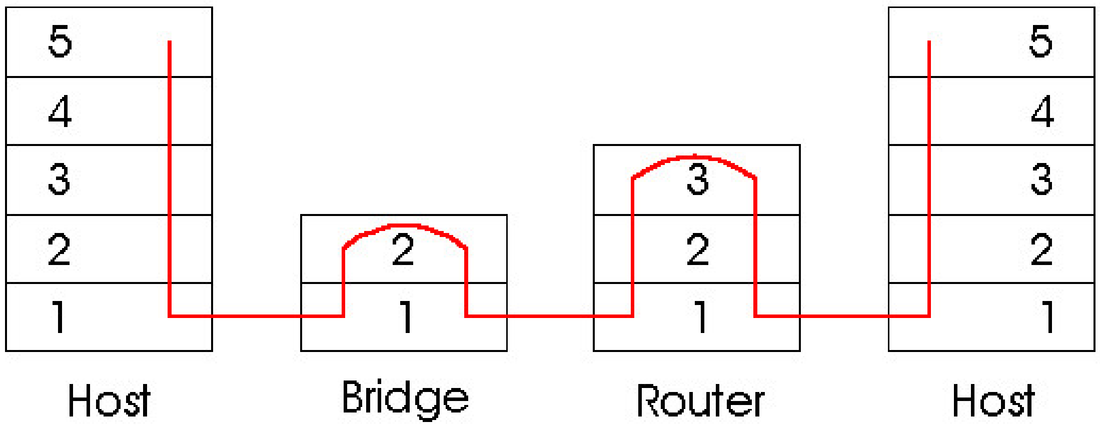
- *Q:* show switch tables and packet forwarding in S_1, S_2, S_3, S_4

Institutional network



Switches vs. Routers

- both store-and-forward devices
  - routers: network layer devices (examine network layer headers)
  - switches are link layer devices
- routers maintain routing tables, implement routing algorithms
- switches maintain switch tables, implement filtering, learning algorithms



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

Master Course
Computer Networks
IN2097

Prof. Dr.-Ing. Georg Carle
Christian Grothoff, Ph.D.
Dr. Nils Kammenhuber

Chair for Network Architectures and Services
Institut für Informatik
Technische Universität München
http://www.net.in.tum.de

Chapter 6 outline – Quality-of-Service Support

6.1 Link virtualization: ATM

6.2 Providing multiple classes of service

6.3 Providing Quality-of-Service (QoS) guarantees

6.4 Signalling for QoS

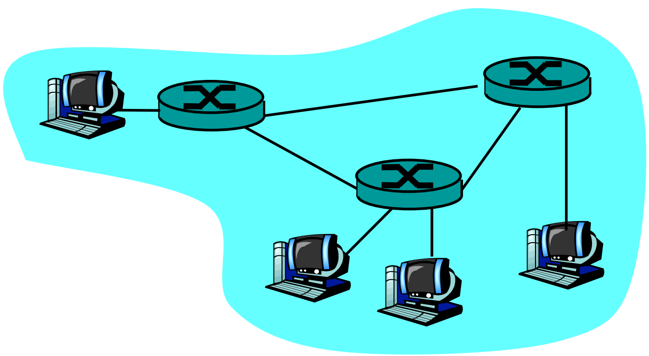
Virtualization of networks

- Virtualization of resources: powerful abstraction in systems engineering:
- Computing examples: virtual memory, virtual devices
  - Virtual machines: e.g., java
  - IBM VM os from 1960's/70's
- Layering of abstractions: don't sweat the details of the lower layer, only deal with lower layers abstractly
- Virtualization of networks: hot topic also today, e.g. applicability for cloud computing

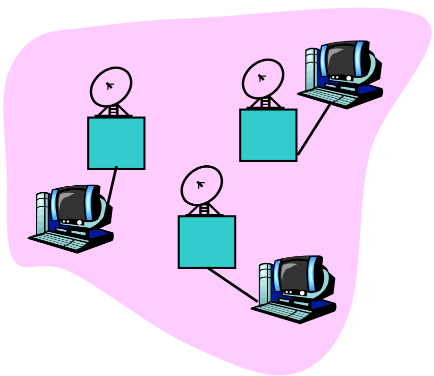
The Internet: virtualizing networks

- 1974: multiple unconnected nets
  - ARPAnet
  - data-over-cable networks
  - packet satellite network (Aloha protocol)
  - packet radio network

… differing in:
  - addressing conventions
  - packet formats
  - error recovery
  - routing



ARPAnet



satellite net

"A Protocol for Packet Network Intercommunication",
V. Cerf, R. Kahn, IEEE Transactions on Communications,
May, 1974, pp. 637-648.

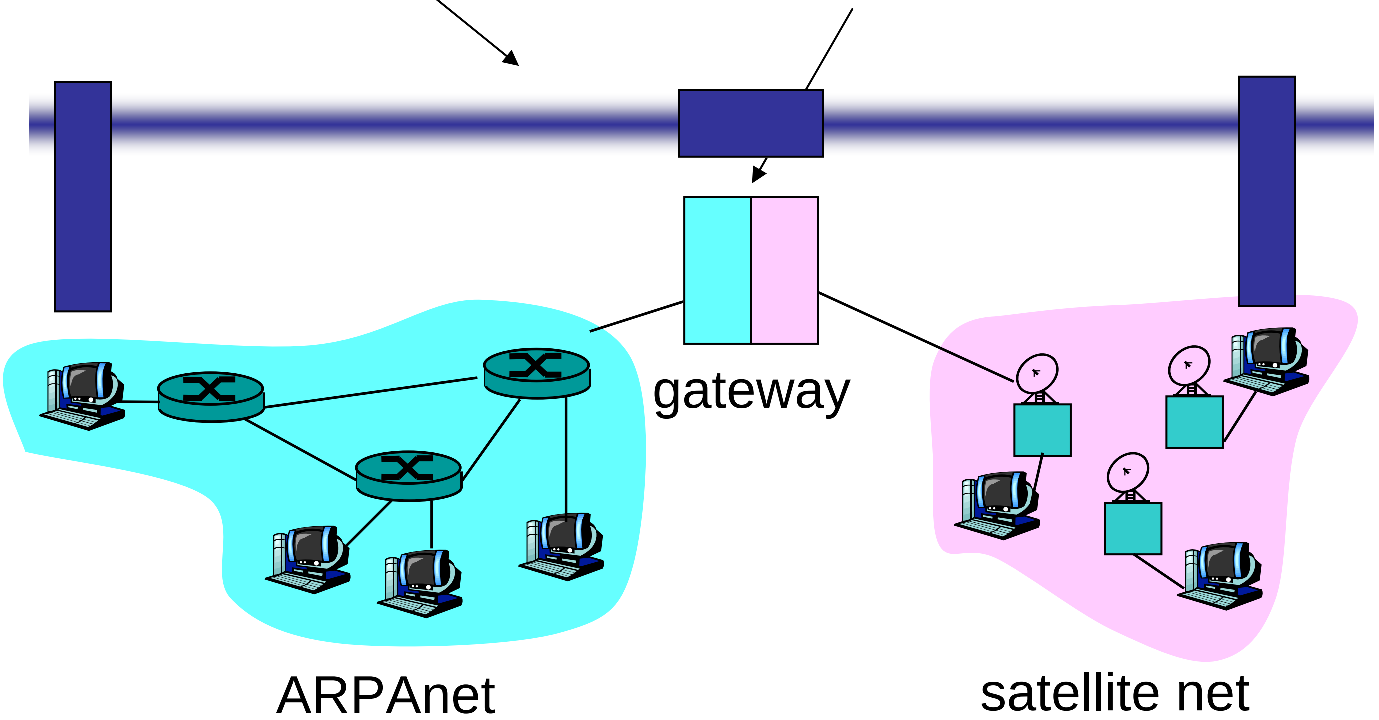
The Internet: virtualizing networks

Internetwork layer (IP):

- addressing: internetwork appears as single, uniform entity, despite underlying local network heterogeneity
- network of networks

Gateway:

- “embed internetwork packets in local packet format or extract them”
- route (at internetwork level) to next gateway



Cerf & Kahn's Internetwork Architecture

- What is virtualized?
- virtualization results in two layers of addressing: internetwork and local network
- new layer (IP) makes everything homogeneous at internetwork layer
- underlying local network technology
  - cable
  - satellite
  - 56K telephone modem
  - today: ATM, MPLS
- … "invisible" at internetwork layer.

ATM and MPLS

- ATM, MPLS separate networks in their own right
  - different
    - service models,
    - addressing,
    - routing

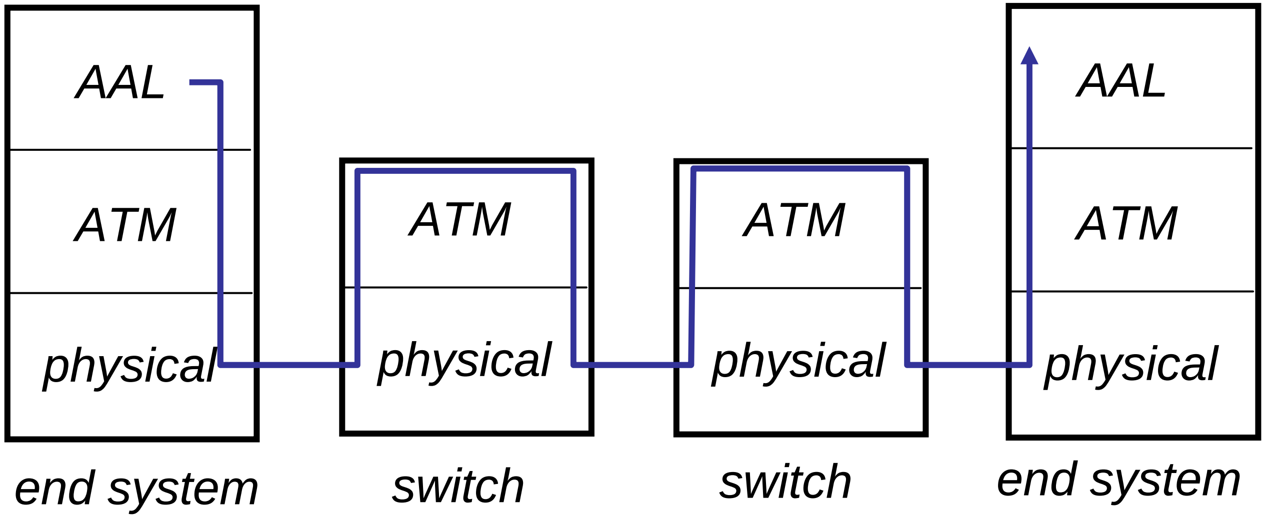
    from Internet
- viewed by Internet as logical link connecting IP routers
  - just like dialup link is really part of separate network (telephone network)
- ATM, MPLS: of technical interest in their own right

Asynchronous Transfer Mode: ATM

- ATM was 1990's/00 standard for high-speed (155Mbps to 622 Mbps and higher) *Broadband Integrated Service Digital Network* architecture

- Goal: *integrated, end-end transport to carry 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

ATM architecture



- AAL – ATM Adaptation Layer: only at edge of ATM network
  - data segmentation/reassembly
  - error detection and (optionally) error recovery
  - roughly analogous to Internet transport layer
- ATM layer: “network” layer
  - cell switching, routing
- physical layer

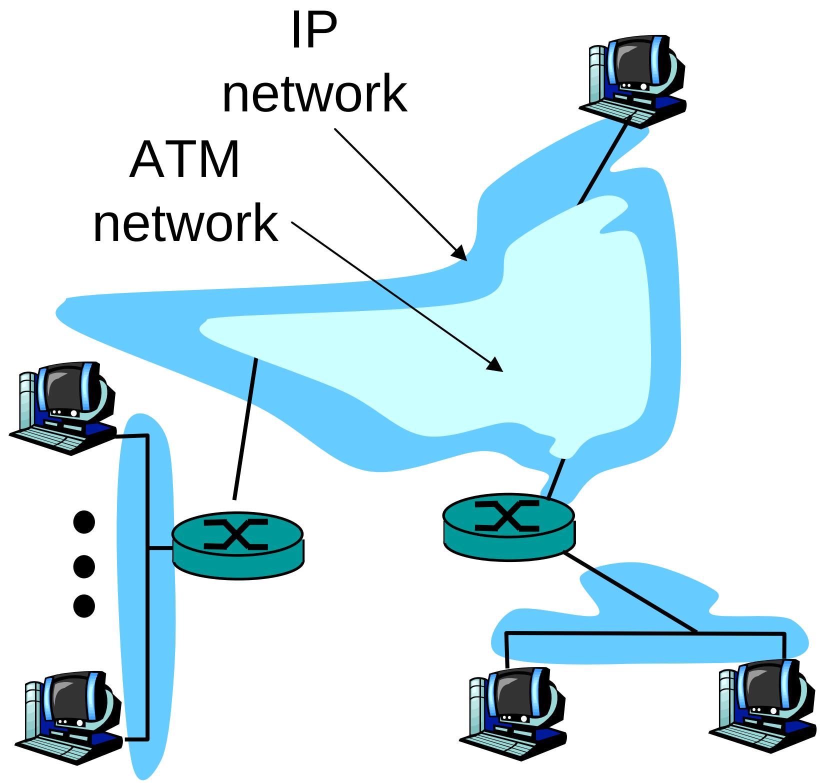
ATM: network or link layer?

Vision: end-to-end transport: "ATM from desktop to desktop"

- ATM *is* a network technology

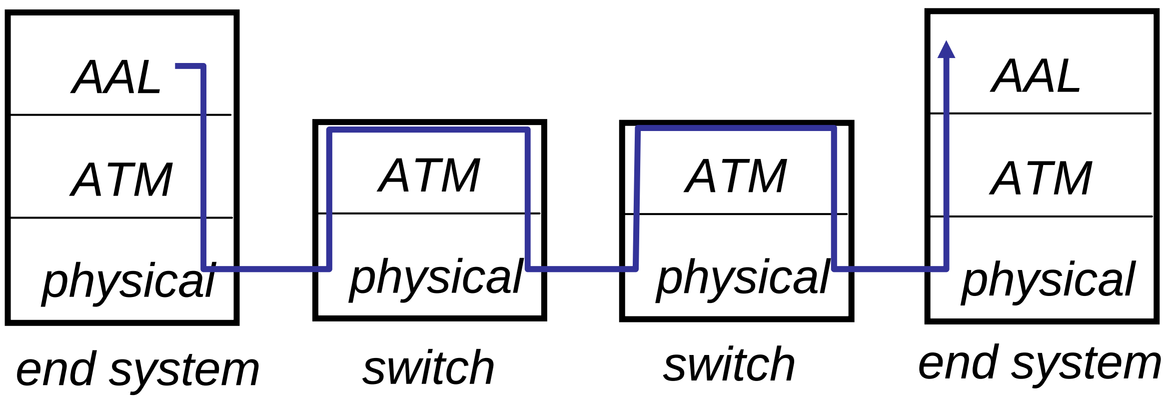
Reality: used to connect IP backbone routers

- "IP over ATM"
- ATM as switched link layer, connecting IP routers



ATM Adaptation Layer (AAL)

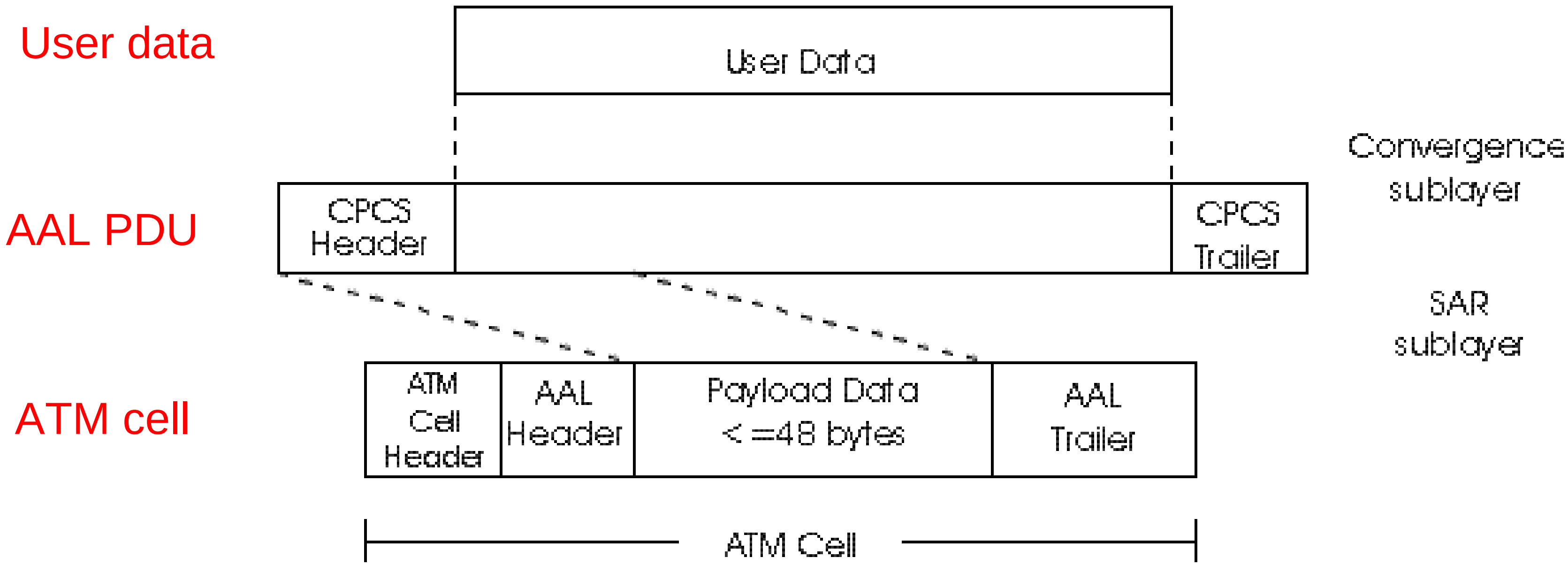
- ATM **Adaptation Layer** (AAL): “adapts” upper layers (IP or native ATM applications) to ATM layer below
- AAL present in data plane **only in ATM end systems**, not in switches (however, signalling in switches needs AAL)
- AAL layer segment (header/trailer fields, data) fragmented across multiple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



(CPCS: Common Part convergence Sublayer)

ATM Layer

ATM Service: transport cells across ATM network

- analogous to IP network layer
- very different services than IP network layer

Network Architecture	Service Model	Guarantees ? Bandwidth	Loss	Order	Timing	Congestion feedback
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

(ABR: Arbitrary Bit Rate
UBR: Unspecified Bit Rate)

ATM Layer: Virtual Circuits

- "source-to-destination path behaves much like telephone circuit"
  - performance-wise
  - network actions along source-to-destination path
- VC transport: cells carried on virtual circuit (VC) from source to destination
  - call setup, teardown for each call *before* data can flow
  - each packet carries VC identifier (not destination ID)
  - *every* switch on source-destination path maintain "state" for each passing connection
  - link, switch resources (bandwidth, buffers) may be *allocated* to VC: to get circuit-like performance
- Permanent VCs (PVCs)
  - long lasting connections
  - typically: "permanent" route between to IP routers
  - configuration by network management
- Switched VCs (SVC):
  - dynamically set up on per-call basis (signalling)

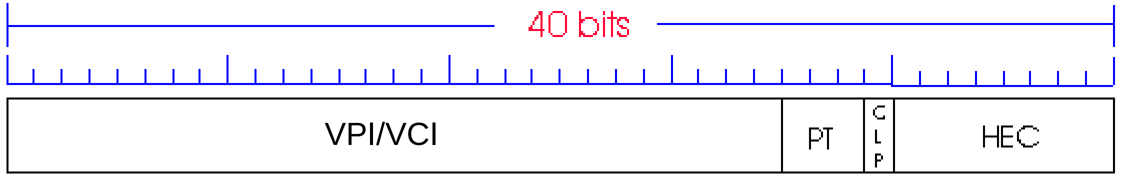
ATM VCs

- 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/destination pair) does not scale (N*2 connections needed)
  - SVC introduces call setup latency, processing overhead for short lived connections

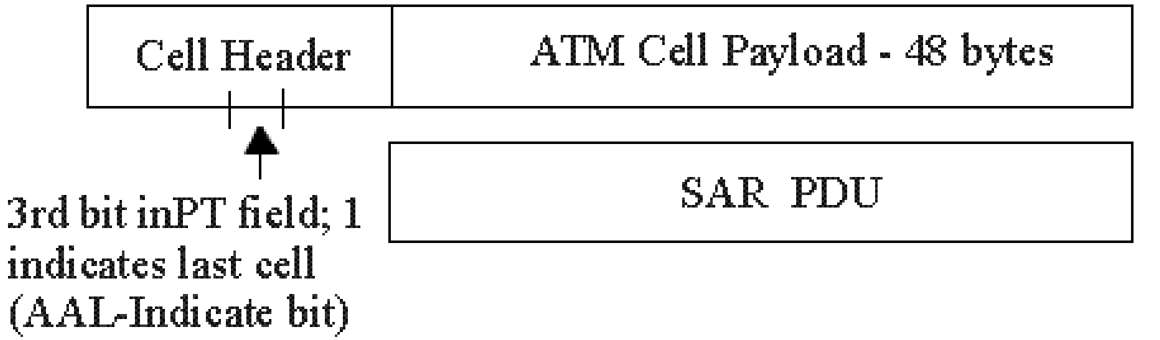
ATM Layer: ATM cell

- 5-byte ATM cell header
- 48-byte payload
  - Why?: small payload ⇨ short cell-creation/transmission delay for digitized voice
  - halfway between 32 and 64 (compromise!)
- Benefit of cells over variable-length packets:
  avoiding that some packets must wait while a packet of maximum size is transmitted. (ATM is still attractive for slow links (e.g. access technologies such as DSL.)

Cell header

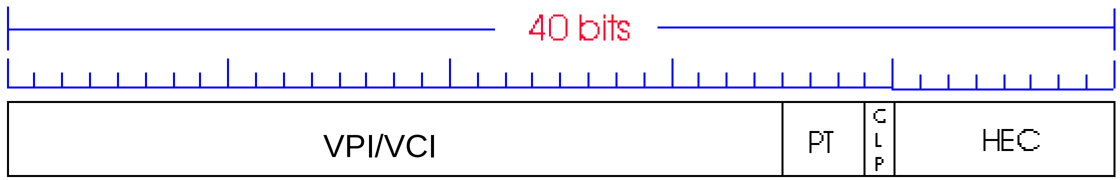


Cell format



ATM cell header

- **VPI/VCI:**
  - ID-space of virtual connections structured into Virtual Path Identifier (VPI) und Virtual Channel Identifier (VCI)
  - may *change* from link to link through network
- **PT:** Payload type
  - e.g. Resource Management (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



Datagram or VC network: why?

Internet

- data exchange among computers
  - “elastic” service, no strict timing requirements
- “smart” end systems (computers)
  - can adapt, perform control, error recovery
  - simple inside network, complexity at “edge”
- many link types
  - different characteristics
  - uniform service difficult

ATM

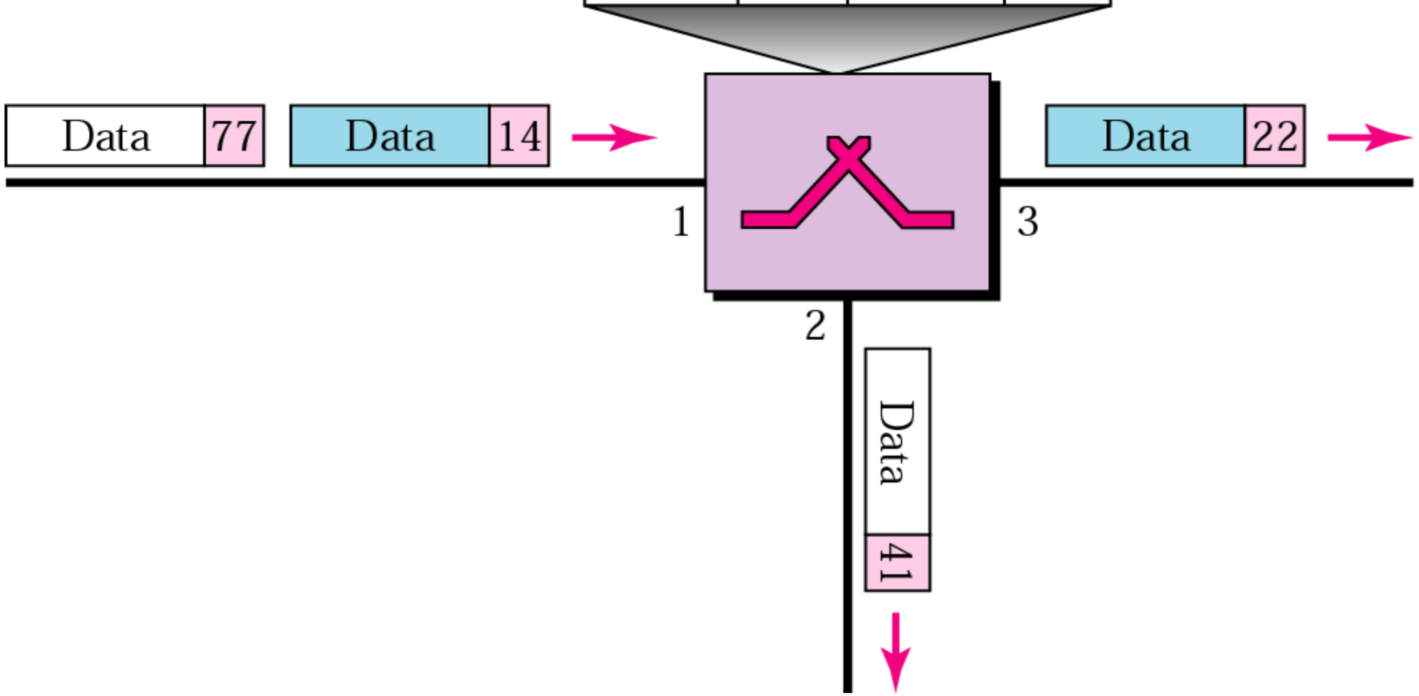
- evolved from telephony
- human conversation:
  - strict timing, reliability requirements
  - need for guaranteed service
- “dumb” end systems
  - telephones
  - complexity inside network

Virtual Circuits and Label Swapping

- Virtual Circuit Switching
- Multiplexing of Variable vs. Fixed Size Packets
- ATM Cell
- Virtual Path Identifiers and Virtual Channel Identifiers
- ATM Virtual Connections

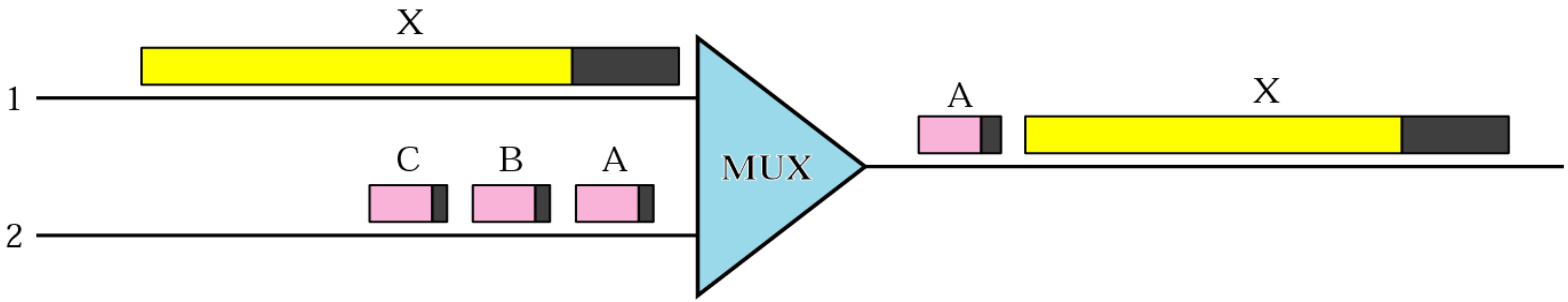
Virtual Circuit Switching

Incoming		Outgoing	
Port	VCI	Port	VCI
1	14	3	22
1	77	2	41

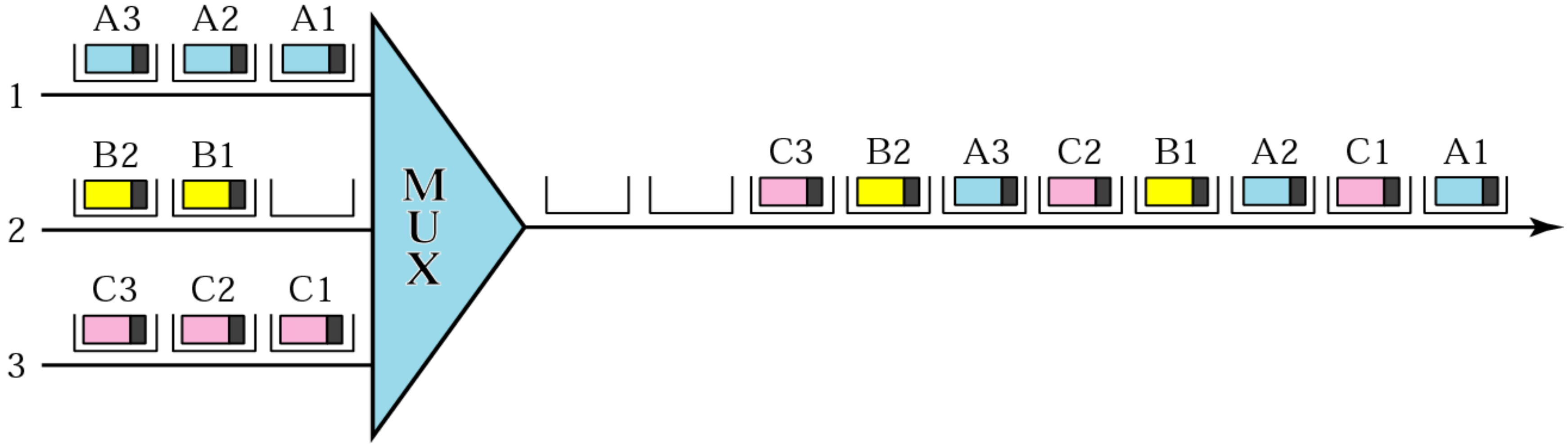


Multiplexing of Variable vs. Fixed Size Packets

- Multiplexing of variable size packets

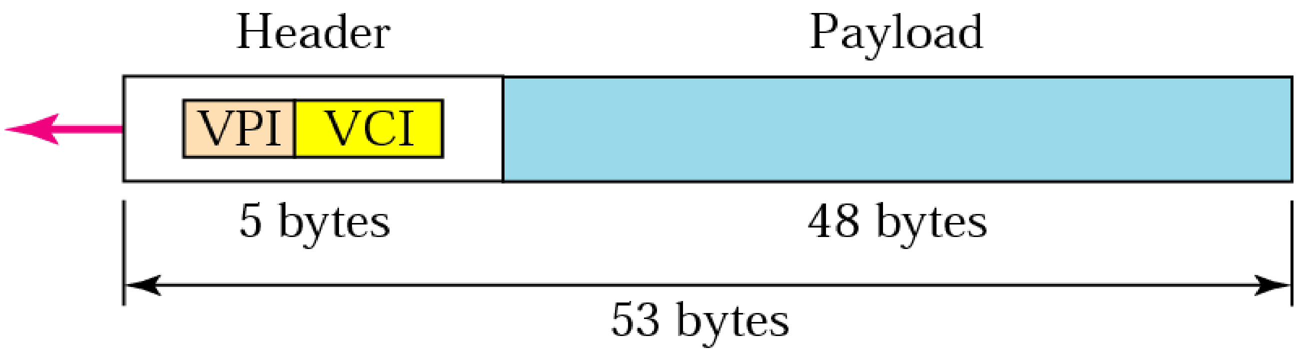


- ATM Multiplexing

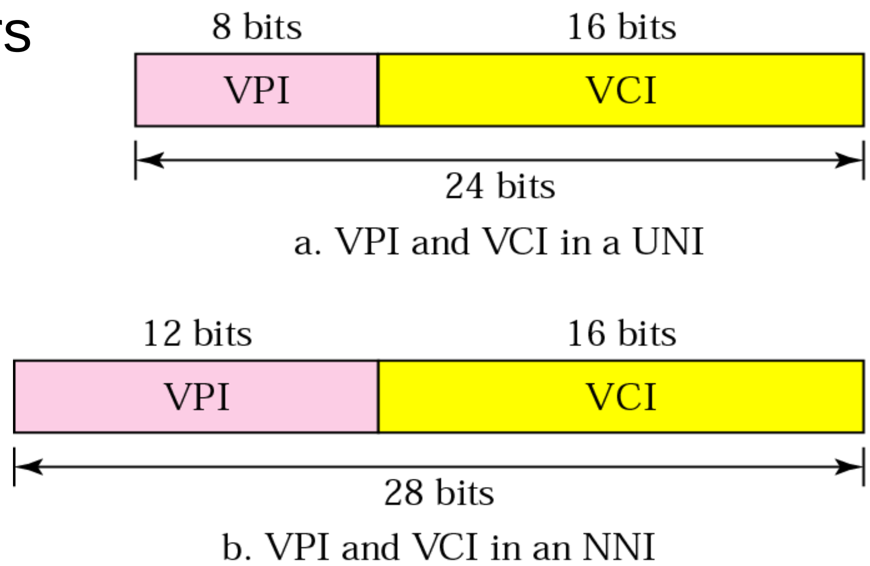


ATM Identifiers

- ATM Cell

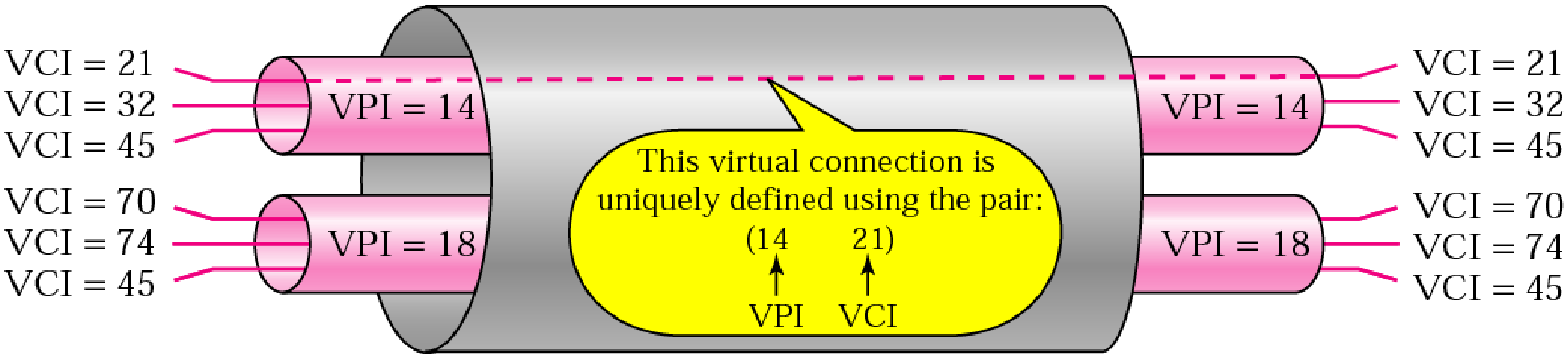


- Virtual Path Identifiers and Virtual Channel Identifiers



(UNI: User-to-Network-Interface
NNI: Network-to-Network-Interface)

ATM Virtual Connections



ATM Physical Layer (more)

Two pieces (sublayers) of physical layer:

- Transmission Convergence Sublayer (TCS): adapts ATM layer above to Physical Medium Dependent (PMD) sublayer below
- Physical Medium Dependent: depends on physical medium being used

TCS Functions:

  - Header **checksum** generation: 8 bits CRC
  - Cell **delineation**
  - With “unstructured” PMD sublayer, transmission of **idle cells** when no data cells to send

ATM Physical Layer

Physical Medium Dependent (PMD) sublayer

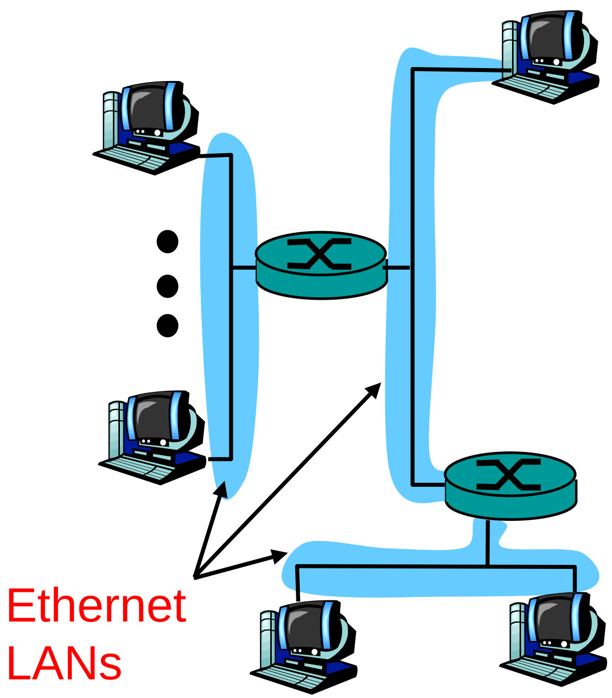
- SONET/SDH: transmission frame structure (like a container carrying bits);
  - bit synchronization;
  - bandwidth partitions (TDM);
  - several speeds:

    OC3 = 155.52 Mbps;
    OC12 = 622.08 Mbps;
    OC48 = 2.45 Gbps,
    OC192 = 9.6 Gbps
- TI/T3: transmission frame structure (old telephone hierarchy): 1.5 Mbps / 45 Mbps
- unstructured: just cells (busy/idle)

IP-Over-ATM

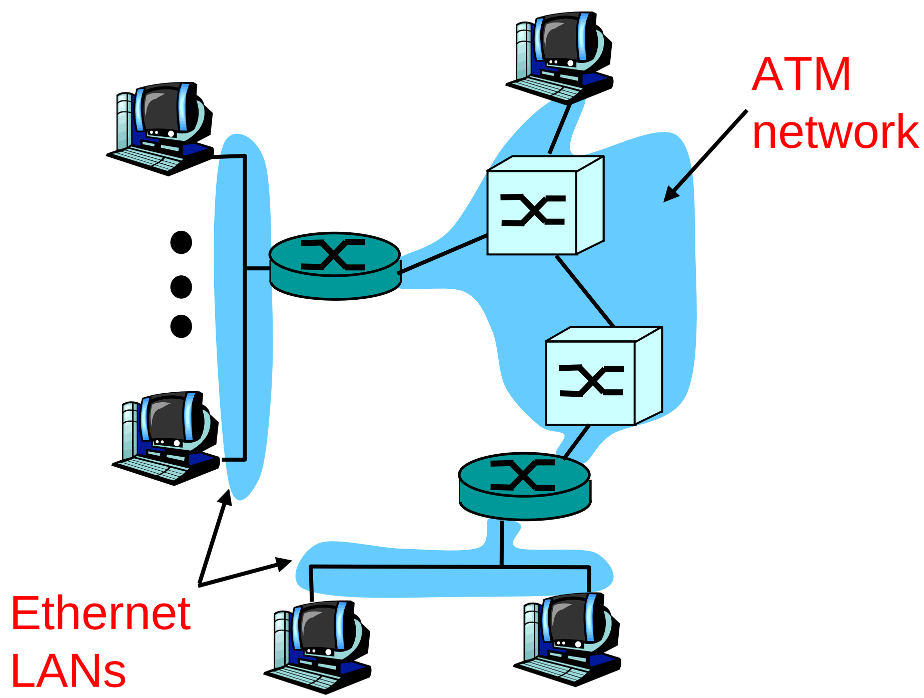
Classic IP only

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

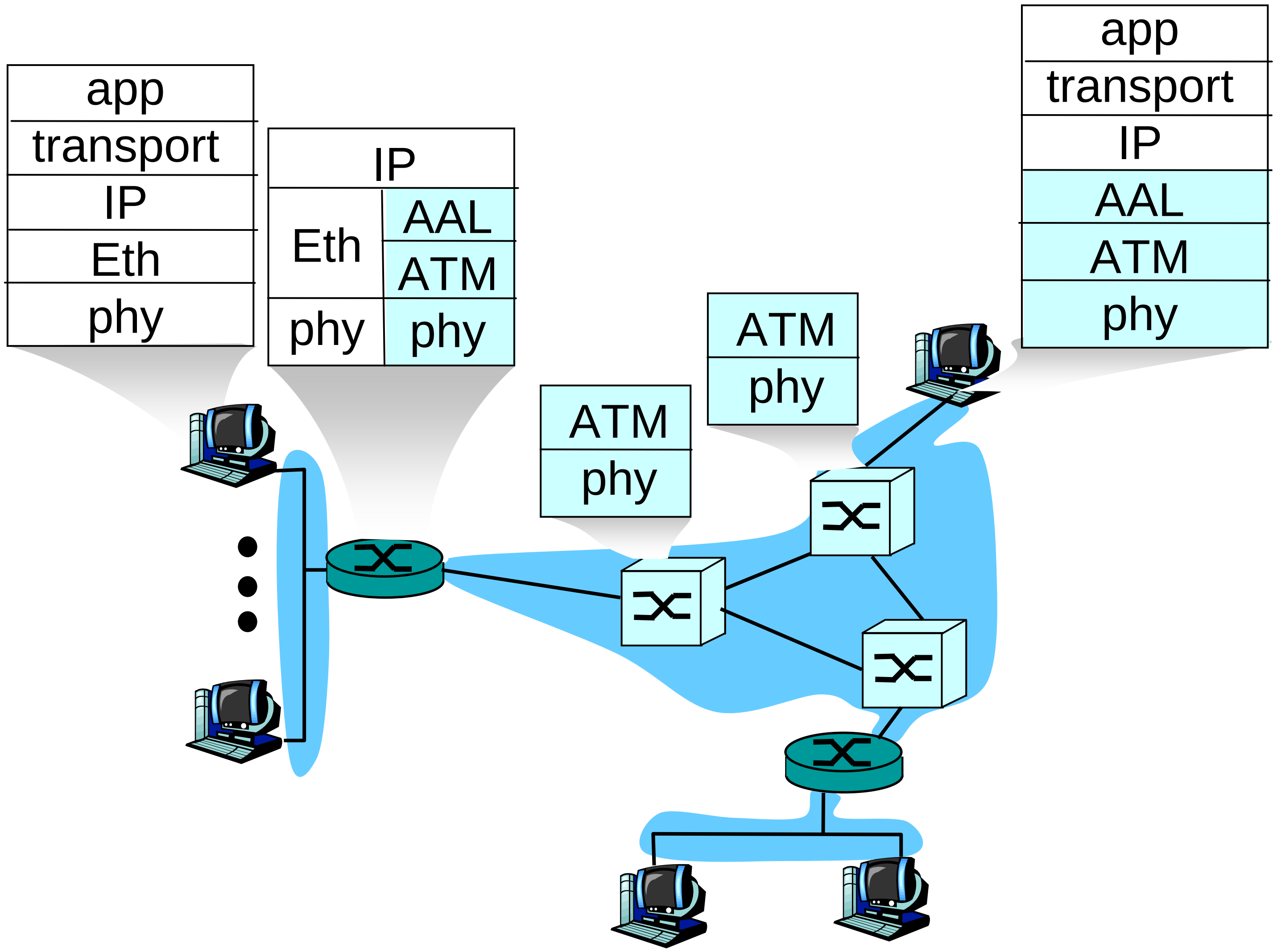


IP over ATM

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



IP-Over-ATM


app
transport
IP
Eth
phy
IP
Eth
AAL
ATM
phy
phy
ATM
phy
ATM
phy
app
transport
IP
AAL
ATM
phy

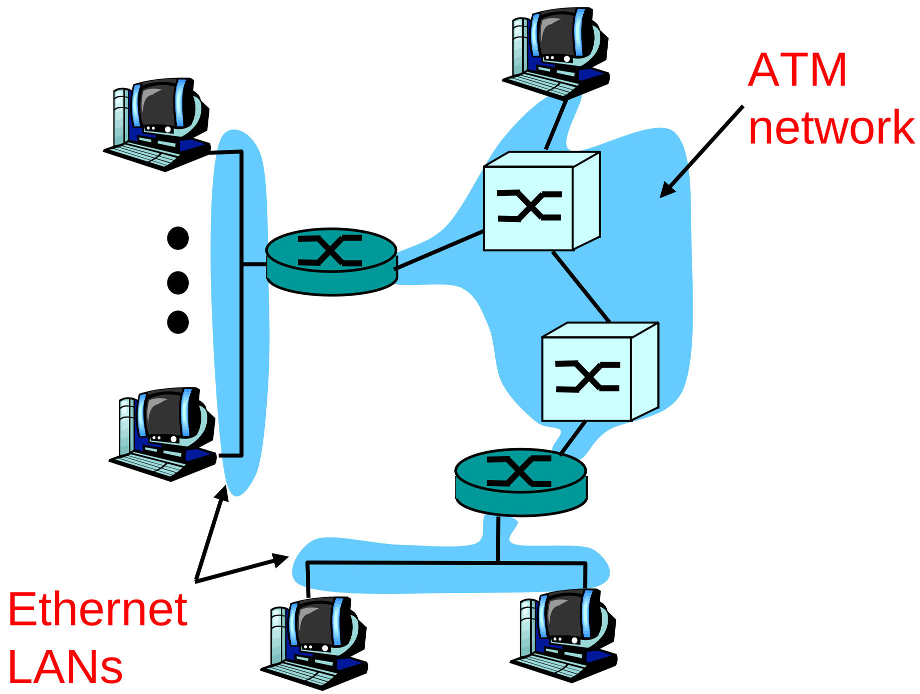
Datagram Journey in IP-over-ATM Network

- at Source Host:
  - IP layer maps between IP, ATM destination 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

IP-Over-ATM

Issues:

- IP datagrams into ATM AAL5 PDUs
- from IP addresses to ATM addresses
  - just like IP addresses to 802.3 MAC addresses!



Chapter 6 outline – Quality-of-Service Support

6.1 Link virtualization: ATM

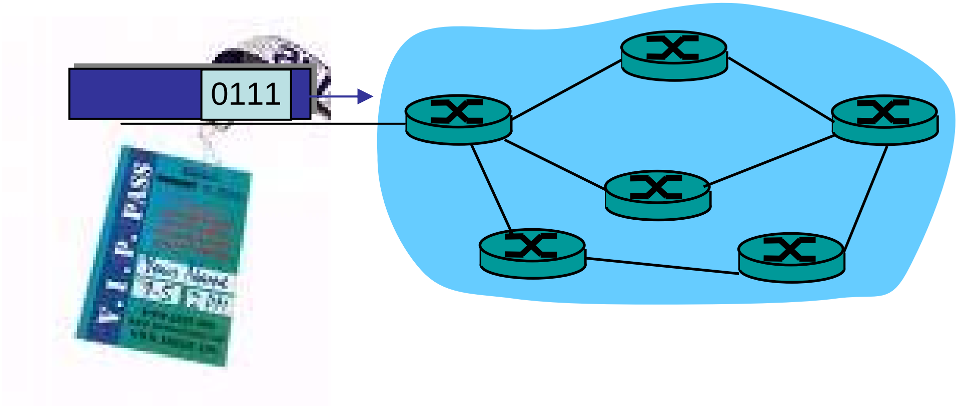
6.2 Providing multiple classes of service

6.3 Providing QoS guarantees

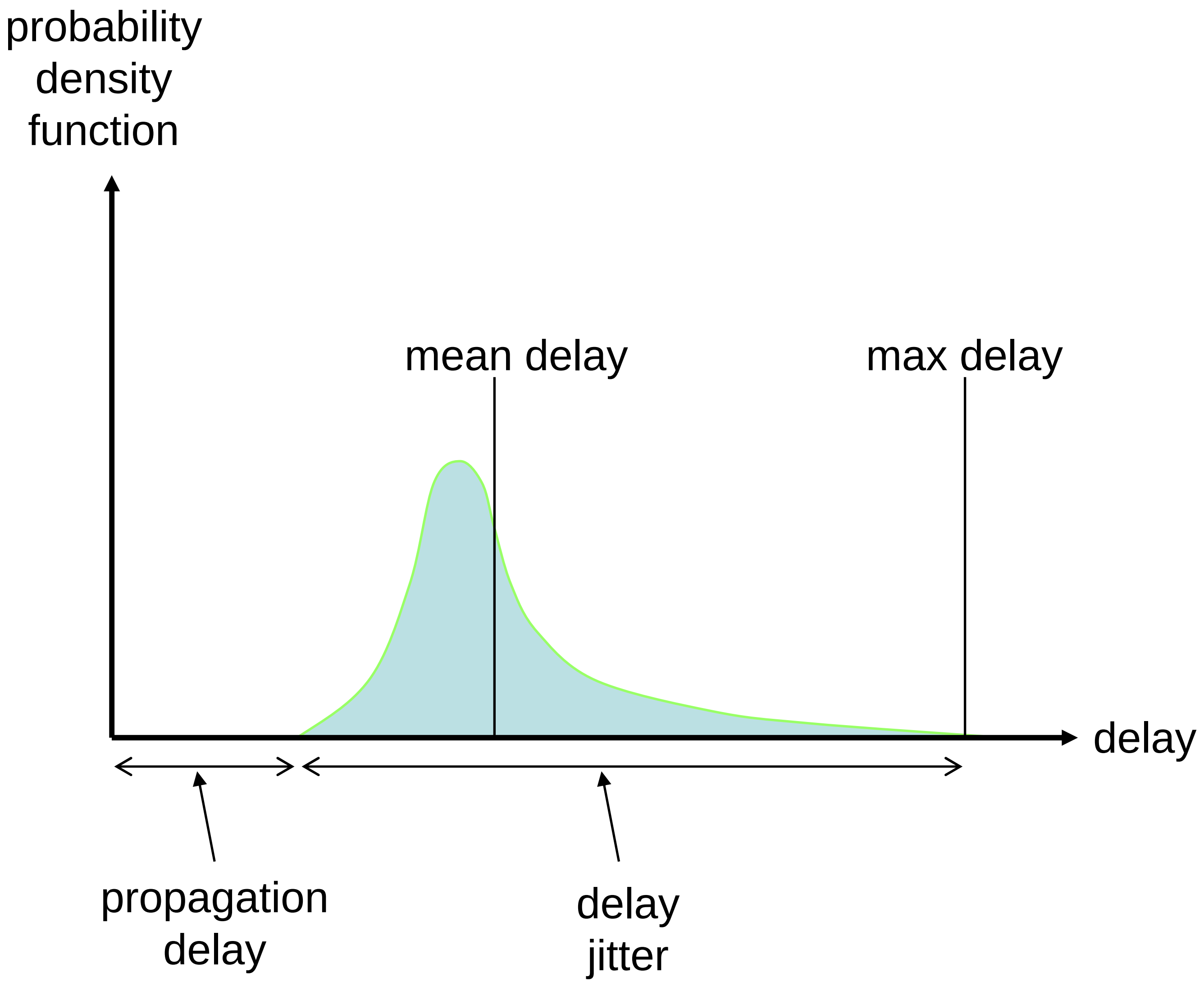
6.4 Signalling for QoS

Providing Multiple Classes of Service

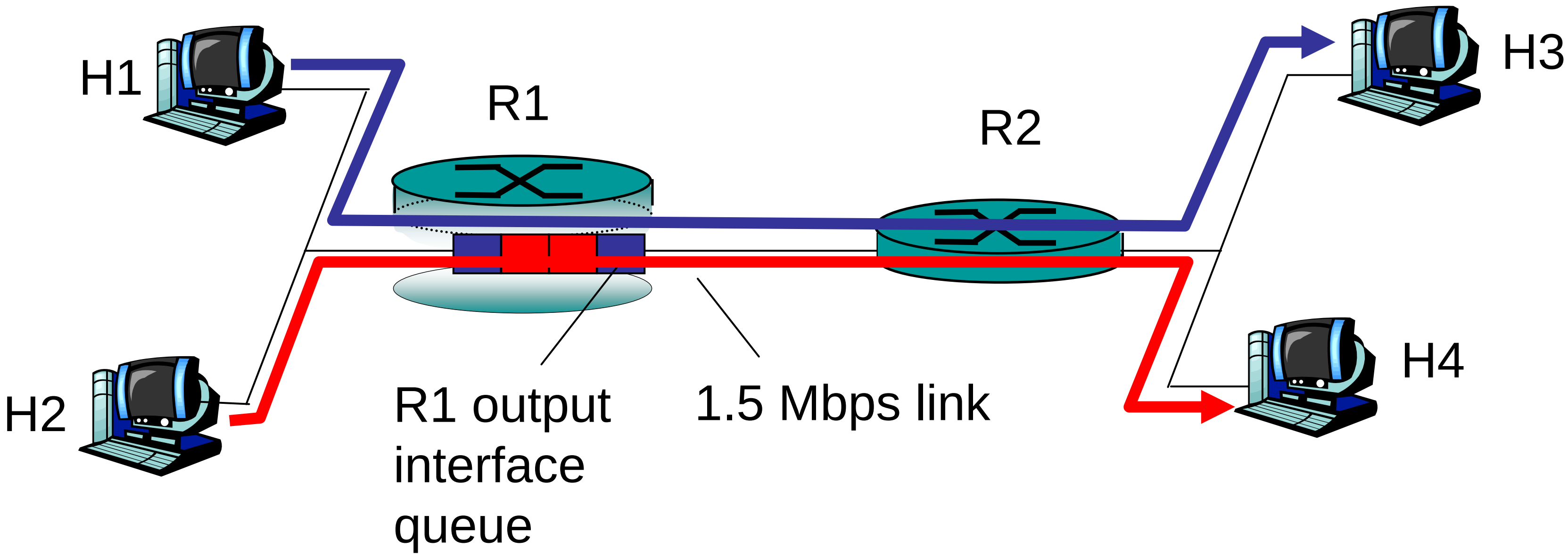
- Traditional Internet approach: making the best of best effort service
  - one-size fits all service model
- Alternative approach: multiple classes of service
  - partition traffic into classes
  - network treats different classes of traffic differently (analogy: VIP service vs regular service)
- granularity:
  differential service among multiple classes, not among individual connections
- history:
  ToS bits in IP header



Delay Distributions

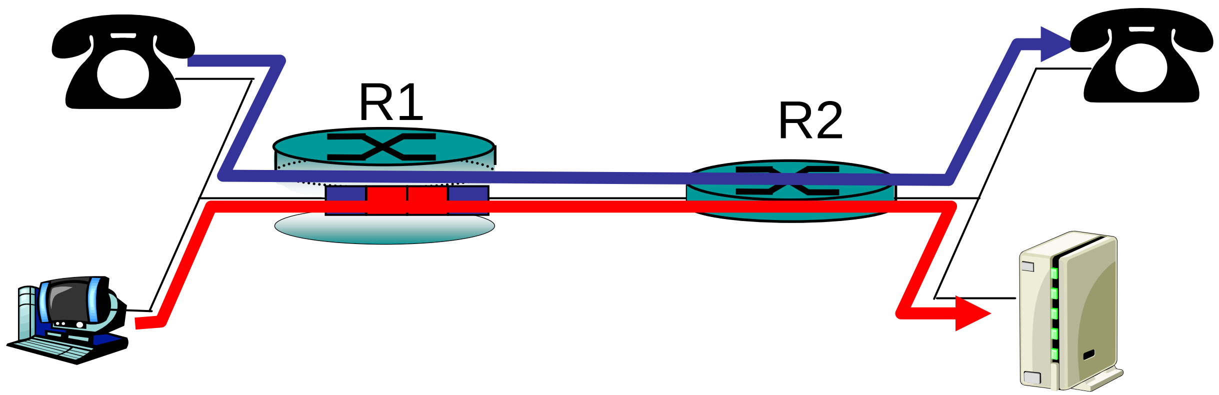


Multiple classes of service: scenario



Scenario 1: mixed FTP and audio

- Example: 1Mbps IP phone, FTP or NFS share 1.5 Mbps link.
  - bursts of FTP or NFS can congest router, cause audio loss
  - want to give priority to audio over FTP

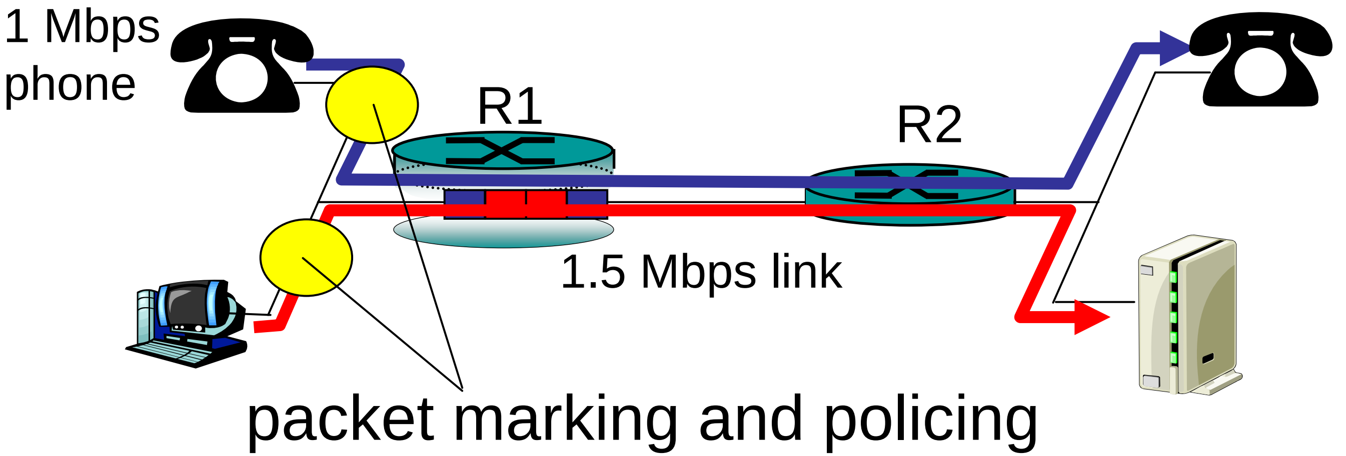


Principle 1

packet marking needed for router to distinguish between different classes; and new router policy to treat packets accordingly

Principles for QOS Guarantees (more)

- what if applications misbehave (audio sends higher than declared rate)
  - policing: force source adherence to bandwidth allocations
- marking and policing at network edge:
  - similar to ATM UNI (User Network Interface)

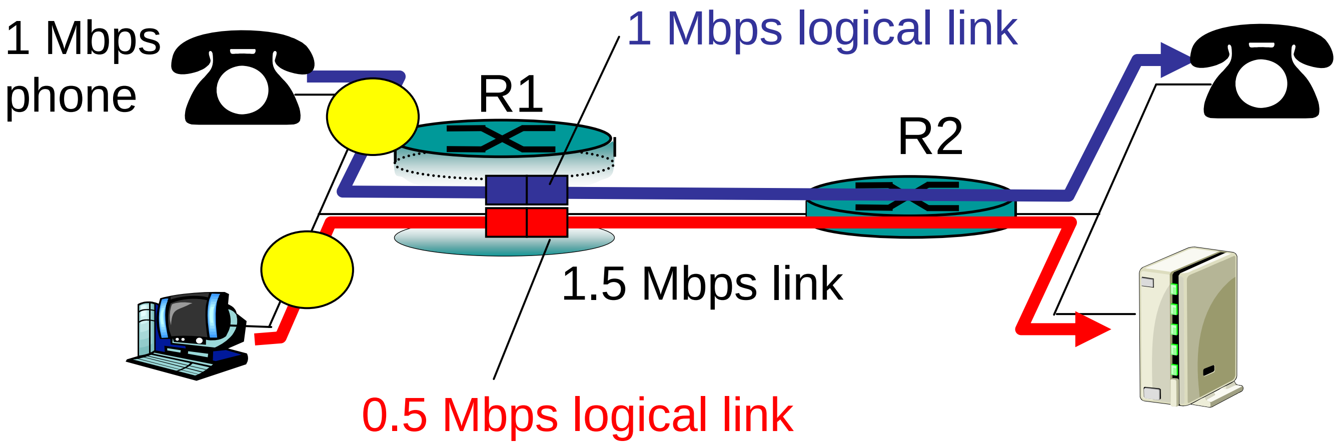


Principle 2

provide protection (*isolation*) for one class from others

Principles for QOS Guarantees (more)

- Allocating *fixed* (non-sharable) bandwidth to flow: *inefficient* use of bandwidth if flows doesn't use its allocation

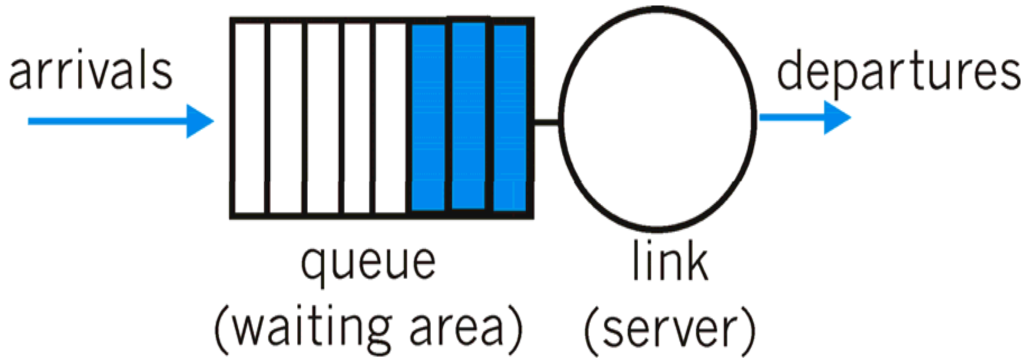


Principle 3

While providing **isolation**, it is desirable to use resources as efficiently as possible

Scheduling And Policing Mechanisms

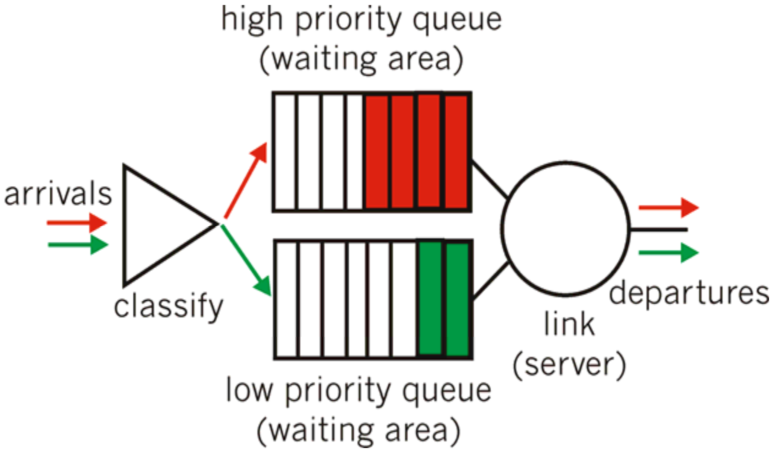
- scheduling: choose next packet to send on link
- FIFO (first in first out) scheduling: send in order of arrival to queue
  - ⇨real-world example?
  - discard policy: if packet arrives to full queue: who to discard?
    - Tail drop: drop arriving packet
    - priority: drop/remove on priority basis
    - random: drop/remove randomly

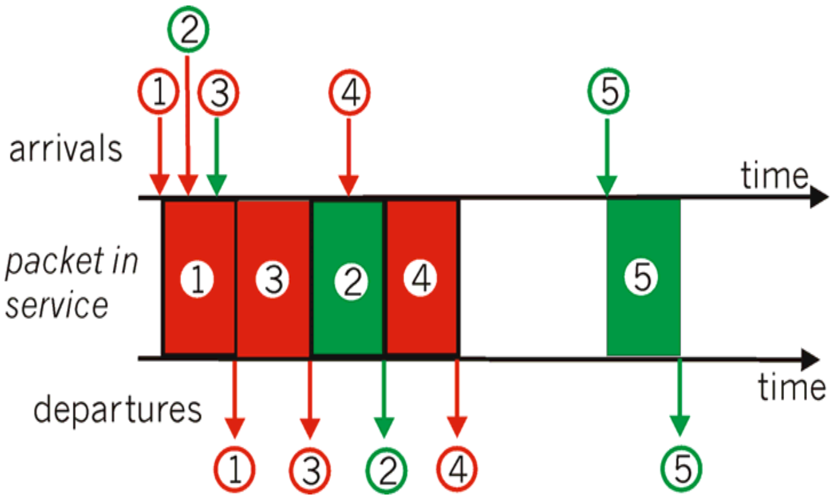


Scheduling Policies: more

Priority scheduling: transmit highest priority queued packet

- multiple *classes*, with different priorities
  - class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc..

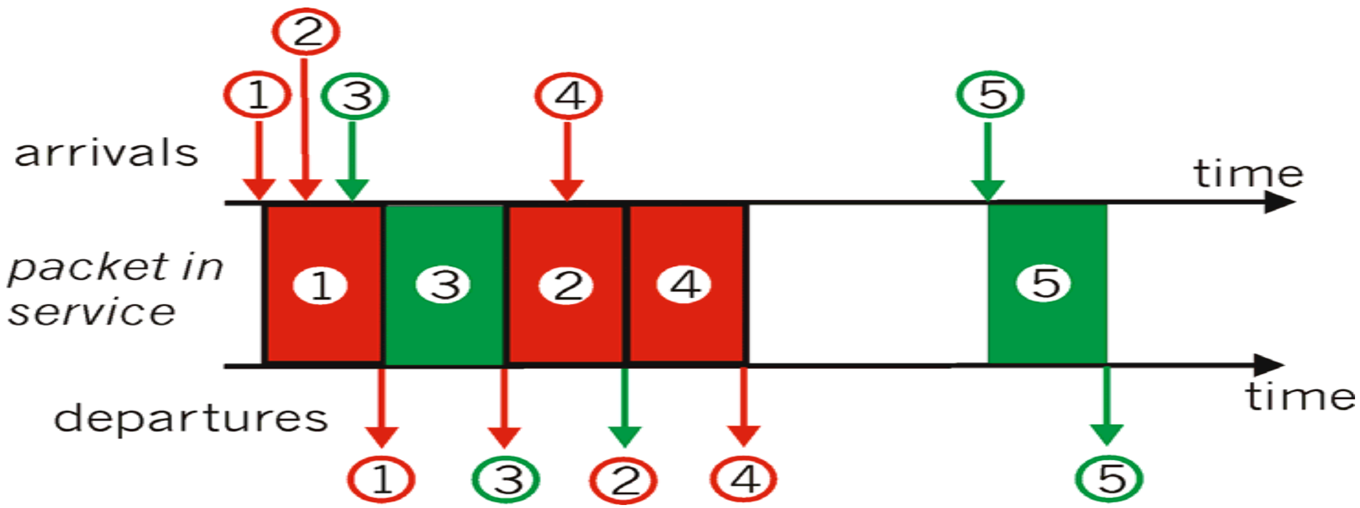




Scheduling Policies: still more

round robin scheduling:

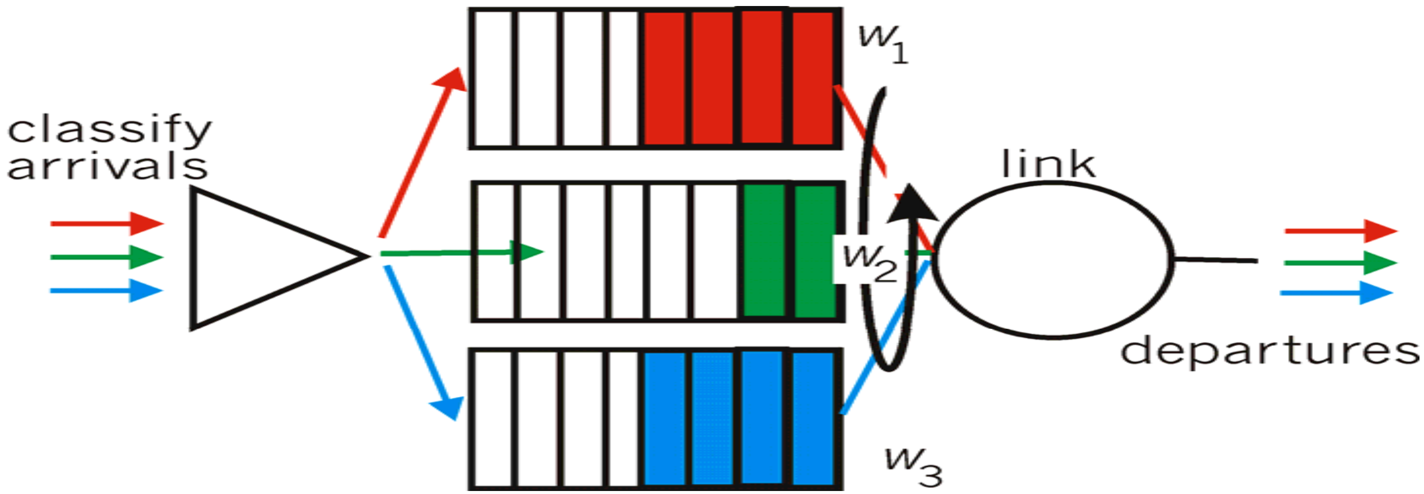
- multiple classes
- cyclically scan class queues, serving one from each class (if available)



Scheduling Policies: still more

Weighted Fair Queuing:

- generalized Round Robin
- each class gets weighted amount of service in each cycle



Policing Mechanisms

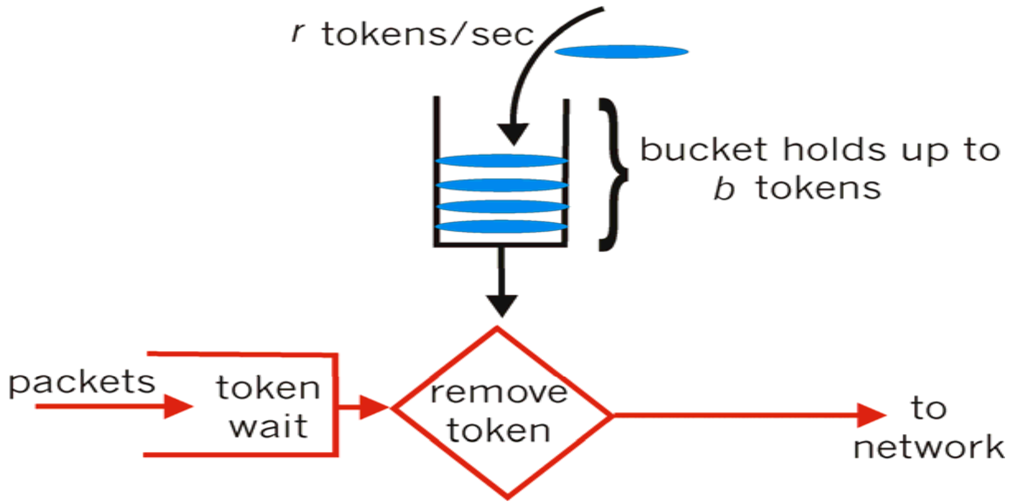
Goal: limit traffic to not exceed declared parameters

Three common-used criteria:

- *(Long term) Average Rate:* how many packets can be sent per unit time (in the long run)
  - crucial question: what is the interval length: 100 packets per sec or 6000 packets per min have same average!
- *Peak Rate:* e.g., 6000 packets per min. (ppm) avg.; 1500 ppm peak rate
- *(Max.) Burst Size:* max. number of packets sent consecutively (with no intervening idle)

Policing Mechanisms

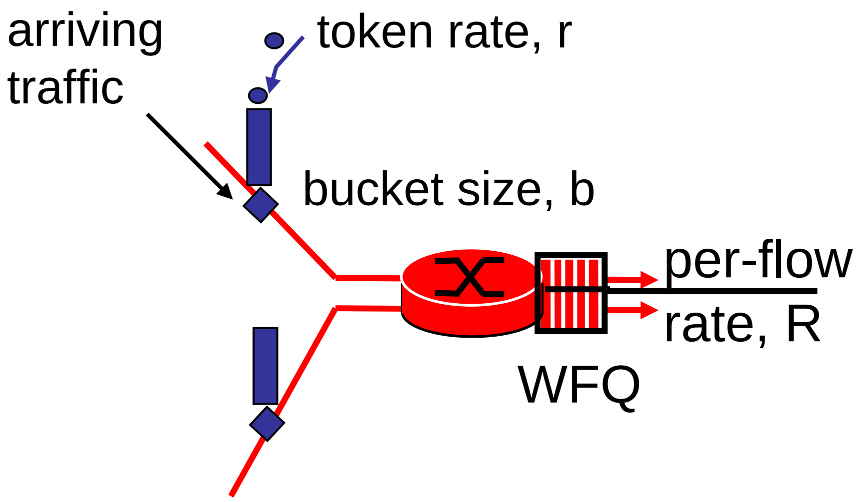
Token Bucket: limit input to specified Burst Size and Average Rate.



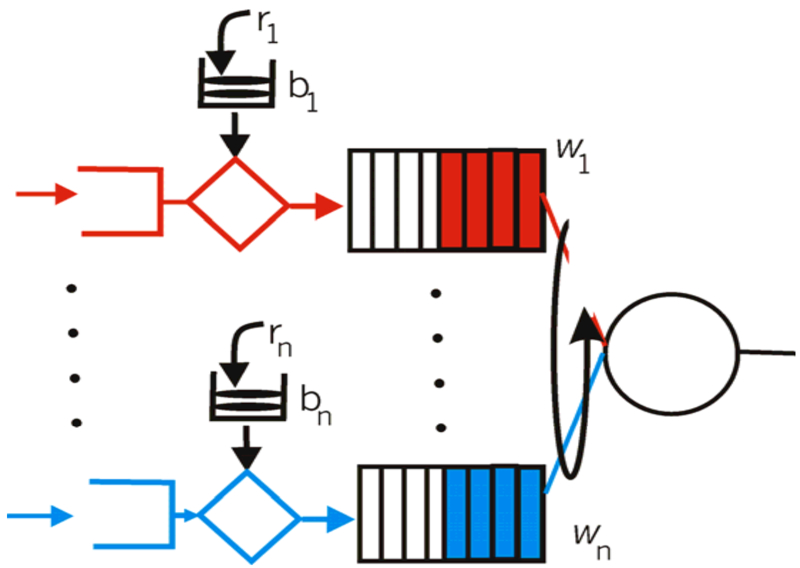
- bucket can hold b tokens
- tokens generated at rate *r token/sec* unless bucket full
- *over interval of length t: number of packets admitted less than or equal to $(r\, t + b)$.*

Policing Mechanisms (more)

- token bucket, WFQ combined provide guaranteed upper bound on delay, i.e., *QoS guarantee*!



$D_{max} = b/R$



IETF Differentiated Services

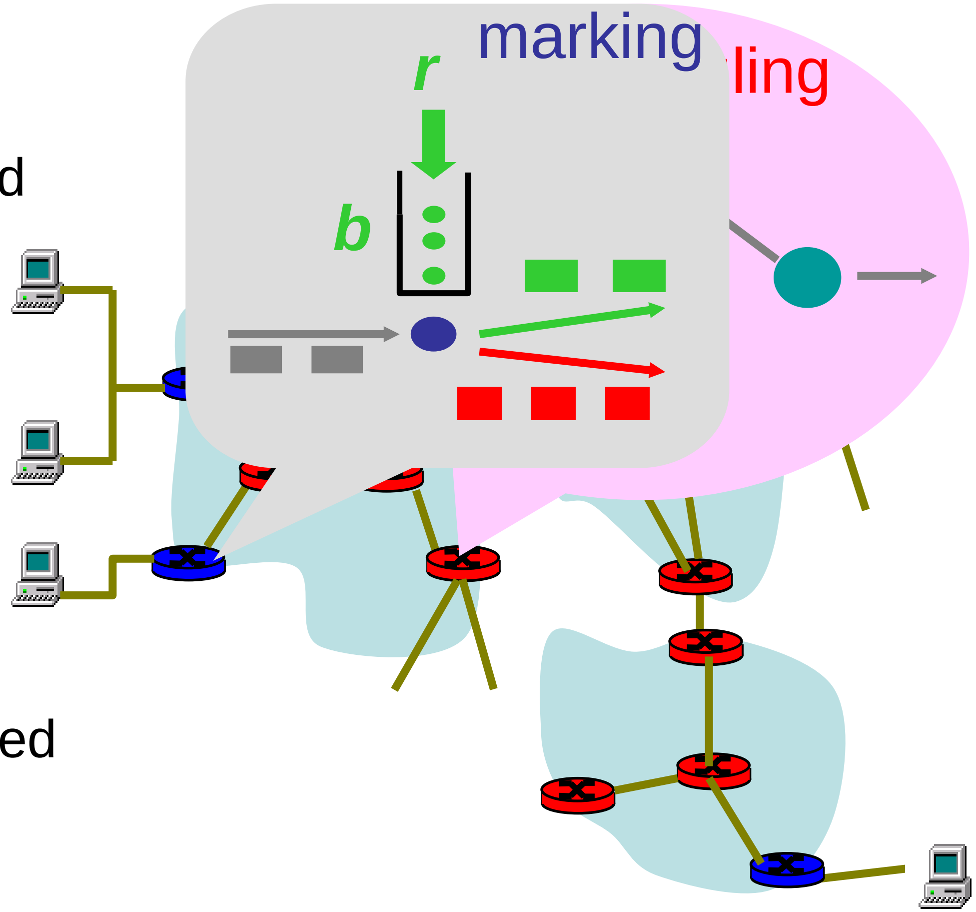
- want "qualitative" service classes
  - "behaves like a wire"
  - relative service distinction: Platinum, Gold, Silver
- *scalability:* simple functions in network core, relatively complex functions at edge routers (or hosts)
  - signaling, maintaining per-flow router state difficult with large number of flows
- don't define define service classes, provide functional components to build service classes

Edge router:

- per-flow traffic management
- marks packets as in-profile and out-profile

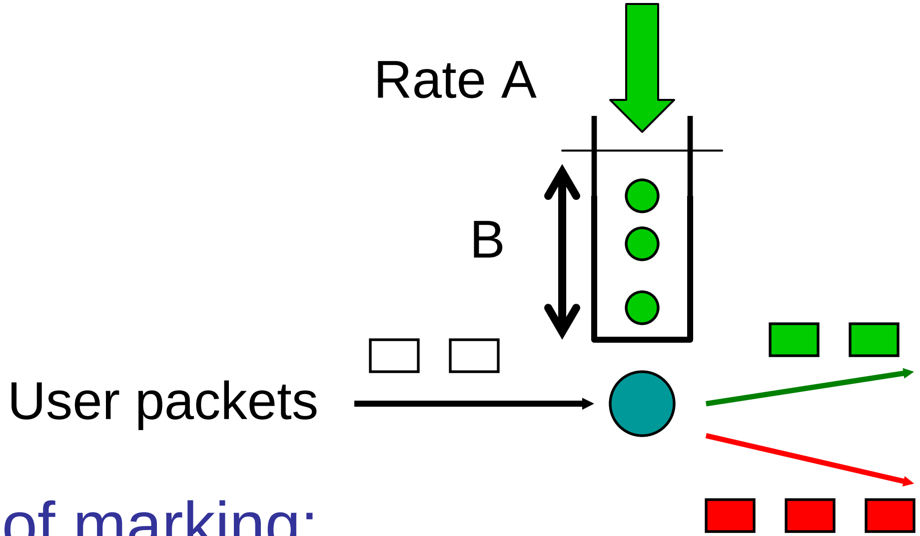
Core router:

- per class traffic management
- buffering and scheduling based on marking at edge
- preference given to in-profile packets



Edge-router Packet Marking

- profile: pre-negotiated rate A, bucket size B
- packet marking at edge based on per-flow profile

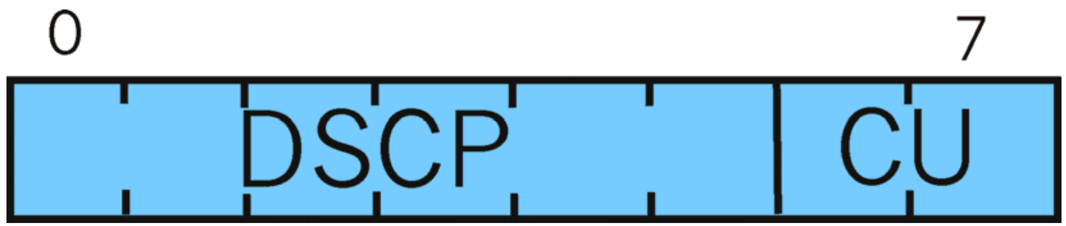


Possible usage of marking:

- class-based marking: packets of different classes marked differently
- intra-class marking: conforming portion of flow marked differently than non-conforming one

Classification and Conditioning

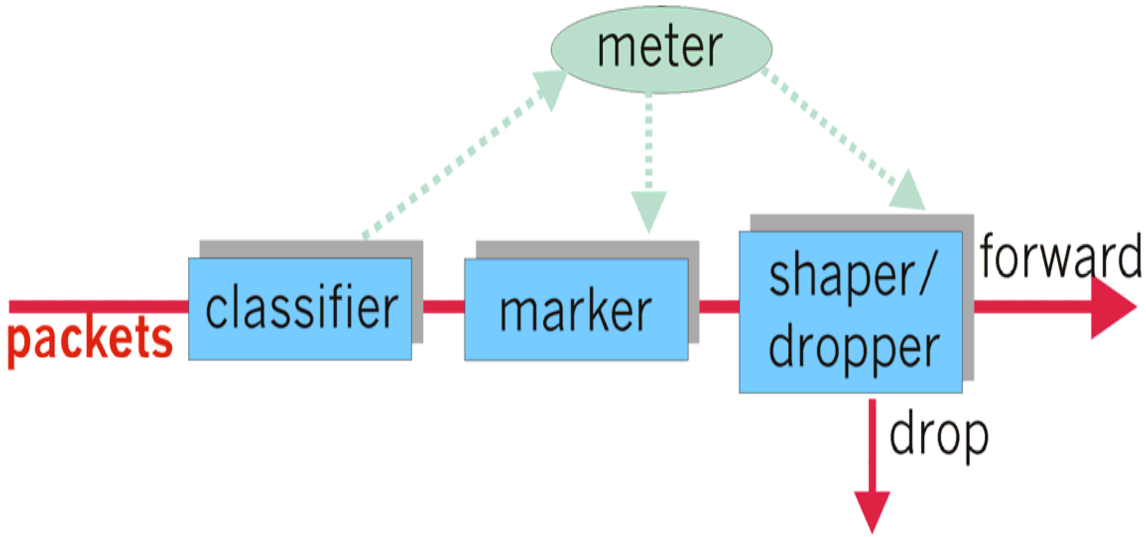
- Packet is marked in the Type of Service (TOS) in IPv4, and Traffic Class in IPv6
- 6 bits used for Differentiated Service Code Point (DSCP) and determine PHB that the packet will receive
- 2 bits can be used for congestion notification



Classification and Conditioning

May be desirable to limit traffic injection rate of some class:

- user declares traffic profile (e.g., rate, burst size)
- traffic metered, shaped if non-conforming



Forwarding (PHB)

- PHB result in a different observable (measurable) forwarding performance behavior
- PHB does not specify what mechanisms to use to ensure required PHB performance behavior
- Examples:
  - Class A gets x% of outgoing link bandwidth over time intervals of a specified length
  - Class A packets leave first before packets from class B

Forwarding (PHB)

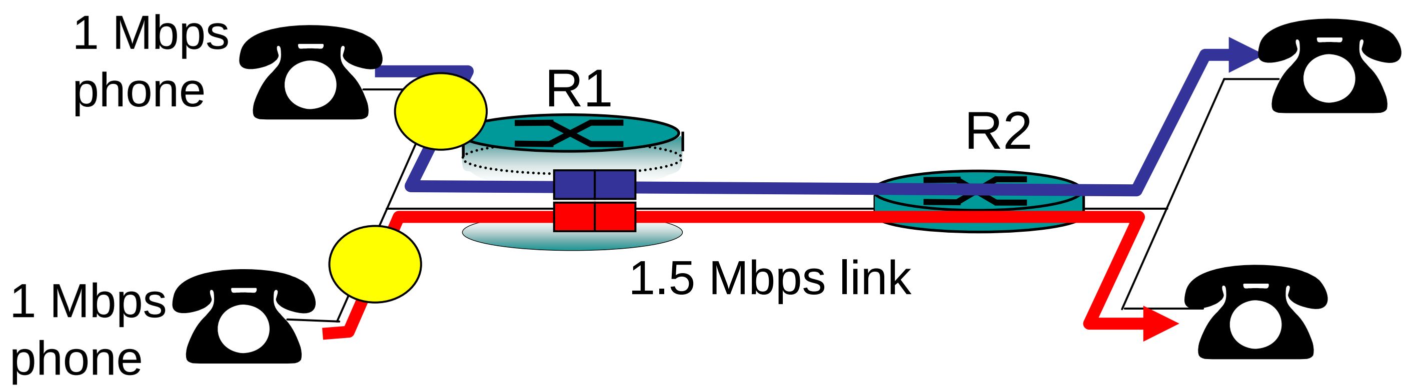
PHBs being developed:

- Expedited Forwarding: packet departure rate of a class equals or exceeds specified rate
  - logical link with a minimum guaranteed rate
- Assured Forwarding: e.g. 4 classes of traffic
  - each guaranteed minimum amount of bandwidth
  - each with three drop preference partitions

Chapter 6 outline – Quality-of-Service Support

Principles for QOS Guarantees (more)

- *Basic fact of life:* can not support traffic demands beyond link capacity

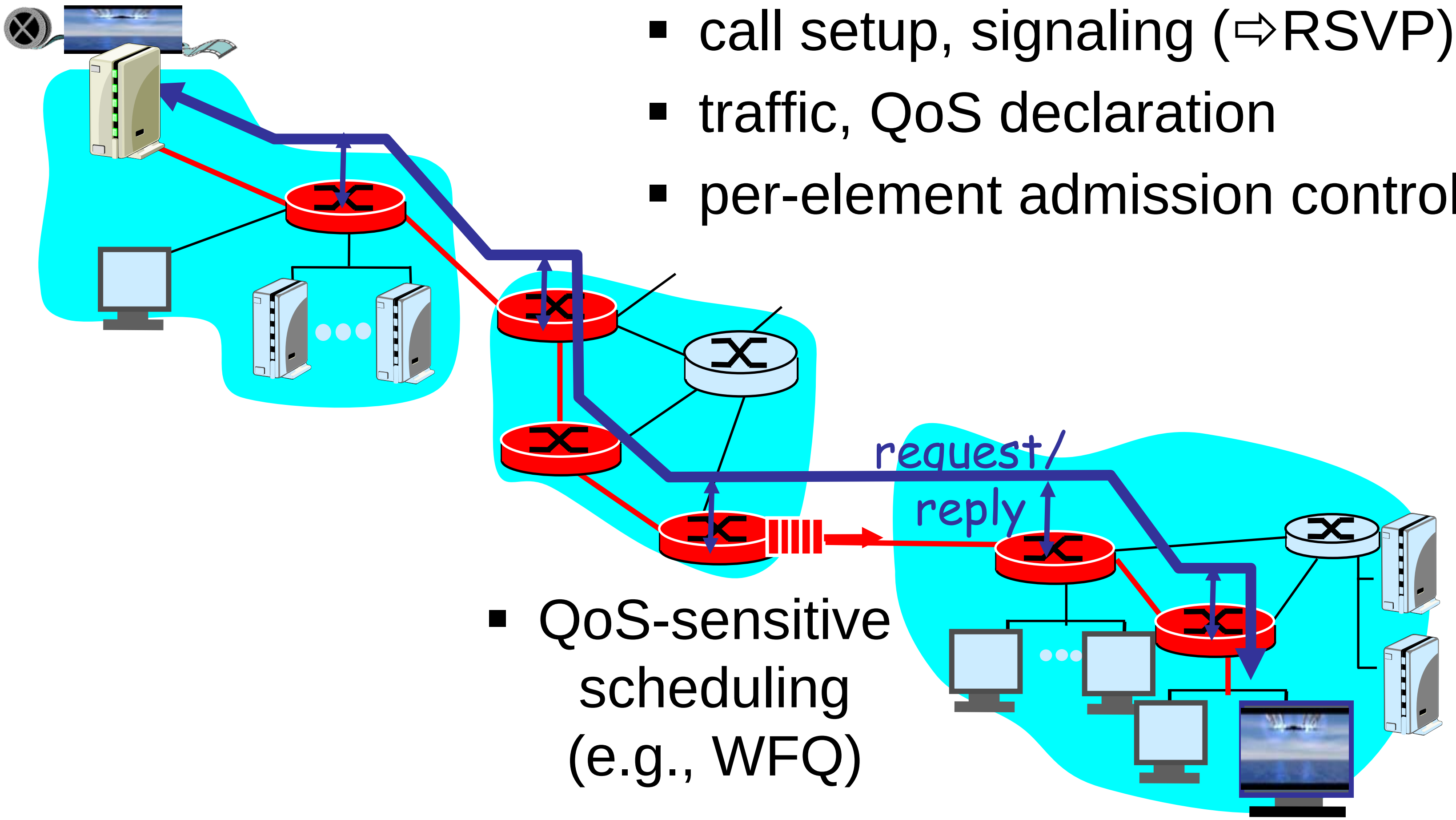


Principle 4

Call Admission: flow declares its needs, network may block call (e.g., busy signal) if it cannot meet needs

QoS Guarantee Scenario

❑ Resource reservation
  - call setup, signaling (⇨RSVP)
  - traffic, QoS declaration
  - per-element admission control



- QoS-sensitive scheduling (e.g., WFQ)

IETF Integrated Services

- architecture for providing QOS guarantees in IP networks for individual application sessions
- resource reservation: routers maintain state info (as for VCs) of allocated resources, QoS requests
- admit/deny new call setup requests:

Question: can newly arriving flow be admitted with performance guarantees while not violated QoS guarantees made to already admitted flows?

Call Admission

Arriving session must :

- declare its QOS requirement
  - R-spec: defines the QOS being requested
- characterize traffic it will send into network
  - T-spec: defines traffic characteristics
- signaling protocol: needed to carry R-spec and T-spec to routers (where reservation is required)
  - RSVP

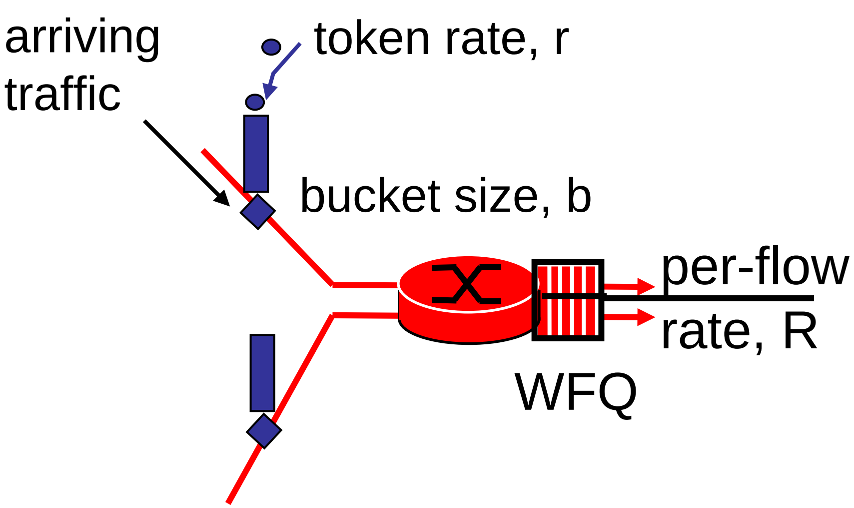
Intserv QoS: Service models [RFC 2211, RFC 2212]

Guaranteed service:

- worst case traffic arrival: leaky-bucket-policed source
- simple (mathematically provable) *bound* on delay [Parekh 1992, Cruz 1988]

Controlled load service:

- "a quality of service closely approximating the QoS that same flow would receive from an unloaded network element."



$$D_{max} = b/R$$

Chapter 6 outline – Quality-of-Service Support

6.1 Link virtualization: ATM

6.2 Providing multiple classes of service

6.3 Providing QoS guarantees

6.4 Signalling for QoS

Signaling in the Internet

connectionless (stateless) forwarding by IP routers	+	best effort service	=	no network signaling protocols in initial IP design

- New requirement: reserve resources along end-to-end path (end system, routers) for QoS for multimedia applications
- RSVP: Resource Reservation Protocol [RFC 2205]
  - " … allow users to communicate requirements to network in robust and efficient way." i.e., signaling !
- earlier Internet Signaling protocol: ST-II [RFC 1819]

RSVP Design Goals

1. accommodate heterogeneous receivers (different bandwidth along paths)
2. accommodate different applications with different resource requirements
3. make multicast a first class service, with adaptation to multicast group membership
4. leverage existing multicast/unicast routing, with adaptation to changes in underlying unicast, multicast routes
5. control protocol overhead to grow (at worst) linear in # receivers
6. modular design for heterogeneous underlying technologies

RSVP: does not…

- specify how resources are to be reserved
  - rather: a mechanism for communicating needs
- determine routes packets will take
  - that's the job of routing protocols
  - signaling decoupled from routing
- interact with forwarding of packets
  - separation of control (signaling) and data (forwarding) planes

RSVP: overview of operation

- senders, receiver join a multicast group
  - done outside of RSVP
  - senders need not join group
- sender-to-network signaling
  - *path message:* make sender presence known to routers
  - path teardown: delete sender's path state from routers
- receiver-to-network signaling
  - *reservation message:* reserve resources from sender(s) to receiver
  - reservation teardown: remove receiver reservations
- network-to-end-system signaling
  - path error
  - reservation error

Master Course Computer Networks IN2097

Prof. Dr.-Ing. Georg Carle,
Christian Grothoff, Ph.D.,
Dr. Nils Kammenhuber,
Dirk Haage

Chair for Network Architectures and Services
Department of Computer Science
Technische Universität München
http://www.net.in.tum.de

Chapter 7:

Network Measurements

Part 2

Chapter 7 Outline – Network Measurements

- Localization of nodes
  - Geoip
  - Network coordinates
- Cross-Layer considerations

Localization of nodes

Localization of nodes

- Provide location-based services
  - Local advertisements
  - Extend/reduce service for local/non-local users (e.g. IPTV often restricted to country boundaries)
- Choosing of servers
  - Load balancing between hosting location
  - Choose nearest instance of a service (anycast)
  - Locate nearest peers in P2P networks
  - Content delivery networks
  - Online games (gameserver)
  - Resource placement in distributed systems
  - TOR
- Find friends, coworkers, …
  - Google Latitude
- Optimization of application layer multicast trees
- …

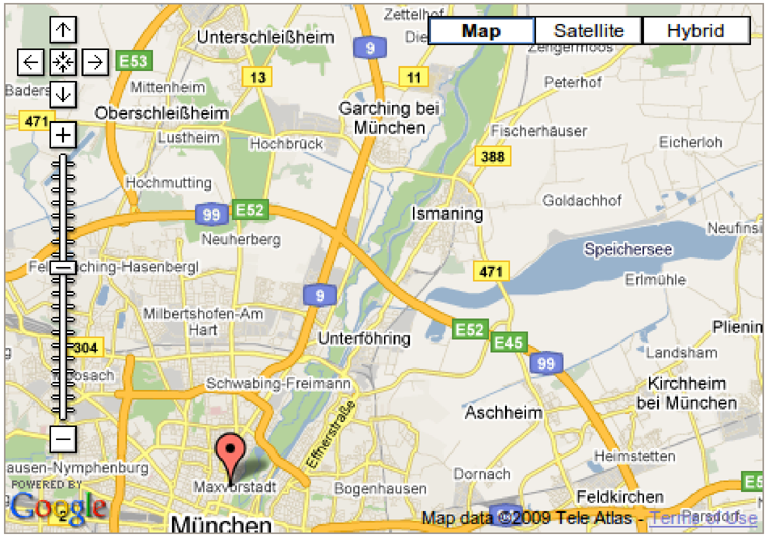
Localization of nodes (II)

- Mapping IP addresses to geo locations
- Determination of distance via latencies
- Triangulation, Trilateration (e.g. wireless networks)
- GPS, Cellular positioning/ Cell ID
- …

GeoIP

- Map IP to a location in the world
- Granularity levels
  - Country/ continent
  - City, maybe urban districts
  - Street/ exact location

Host Name:	131.159.20.45
IP Address:	131.159.20.45
Country:	Germany
Country code:	DE (DEU)
Region:	Bayern
City:	Munich
Postal code:	
Calling code:	+49
Longitude:	11.5833
Latitude:	48.15



GeoIP (II)

- Basic data sources
  - AS information
  - Whois/RIR information
  - Provider data

- Additional sources
  - User input
    - Update location manually
    - Accurate positioning devices
      - Smart phone with GPS
      - Verify/ update current position for used public IP
    - Track changes in IP of same user
      - Mitigate effect of changing IP connection

- Reduce bias by combining sources
  - Verify data, filter inaccurate data

GeoIP (III)

- Accuracy depends on location database
  - More accurate for static IPs (server, university, …)
  - Less accurate for home connections
    - Frequent changes
    - Change in IP often also changes the geo location of that IP
  - Not usable in private networks
    - E.g. cellular network (currently private networks + NAT)

- Provider cooperation is required
  - Detailed information for each and every IP
  - Disclose internal structure (subnets, connectivity)
    - Which subnet is used at which site (city, maybe even parts)
  - Update in case of changes

- Single point of failure
  - Excessive use slows down localization
  - Not usable for massive requests

- Many different implementations

Network coordinates

- Latencies between nodes as a metric for distance
  - Round trip time
    - Simplest measurement at all (ping)
    - Most accurate (only one clock involved)
    - Similar to real distance (propagation speed nearly constant)

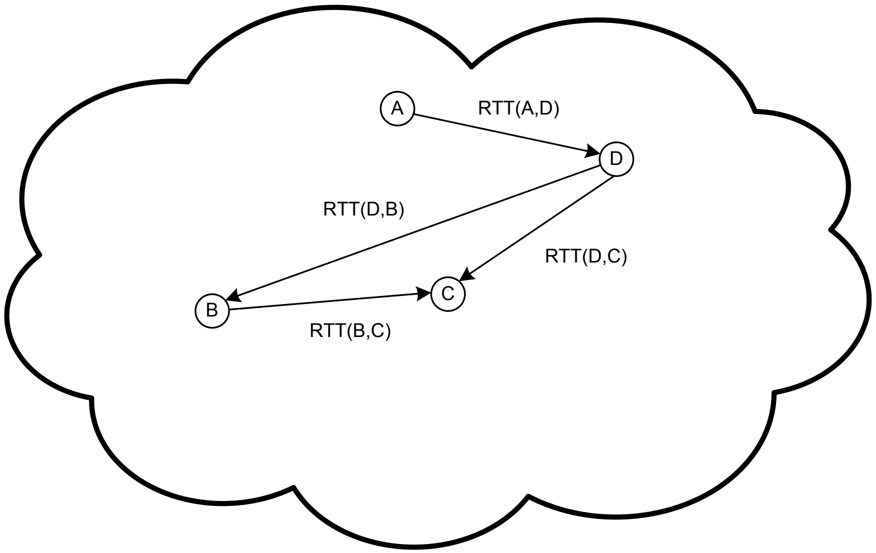
- How to get?
- Simple approach:
  Measurements between all pairs of nodes
  - $O(n^2)$
  - Does not scale (cannot be used for large networks)
  - Rely on actual traffic → hybrid measurement
  - Normally no traffic to all nodes available
    - Active measurements (even worse scaling)

Network coordinates (II)

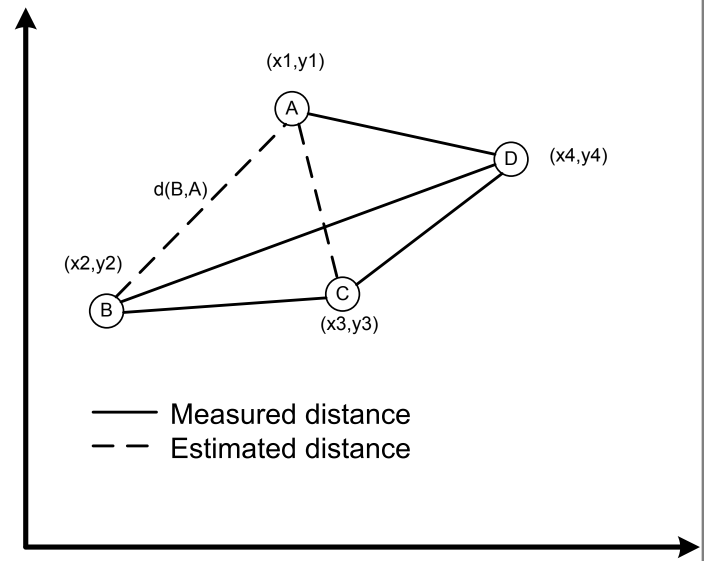
- Measure the distances to some neighbors
  - Neighbors might be known hosts, not near hosts
- Calculate a artificial coordinate in a metric space
  - Metric space = distance between nodes can be calculated
  - E.g. Euclidean n-space
- Approximate the latency
  - Distance between nodes in the coordinate system is approximation to the latency
- Abstract definition:
  - Embed network graph into a metric space
  - Metric embedding/ graph embedding

Example

Internet



Euclidean space (2D)



$$d(B,A) = \left|(x_2, y_2),(x_1, y_1)\right| = \left(\left|x_1 - x_2\right|, \left|y_1 - y_2\right|\right)$$

Network coordinates (III)

- Advantages
  - Small overhead
    - Only requires small number of measurements
    - No additional traffic
(application traffic = measurement traffic)
    - Piggy-back the coordinate information
  - Each host can calculate the distance to every other host
    - Only requires the coordinates

- Design goals
  - Accuracy: small error for RTT estimations
  - Scalability: large-scale networks, small overhead, no bottlenecks
  - Flexibility: adapt coordinates to network changes
  - Stability: no drift, oscillation of coordinates
  - Robustness: small impact of error by malicious nodes, nodes with high errors

Triangle inequality

- Intuition:
  direct latency between 2 nodes should be smaller than any indirection

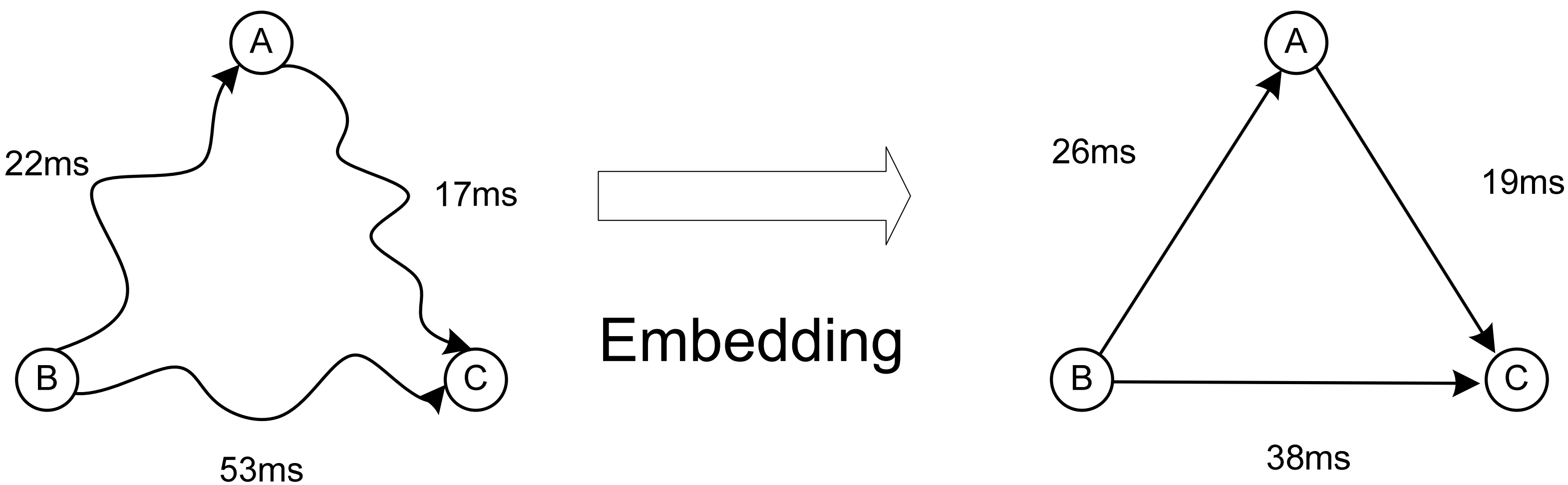
$$d(a,b)+d(b,c)\geq d(a,c)$$

- Triangle inequality violations (TIV) inherent to Internet routing structure
  - Selective/ private peering
  - Hot potato routing
  - Link metric ≠ latency
  - Asymmetric links (e.g. DSL, UMTS)
- TIVs are common
  - >85% of all host pairs part of a TIV
  - For 20-35% exists a path that is at least 20% shorter

(Traces: King, Azureus)

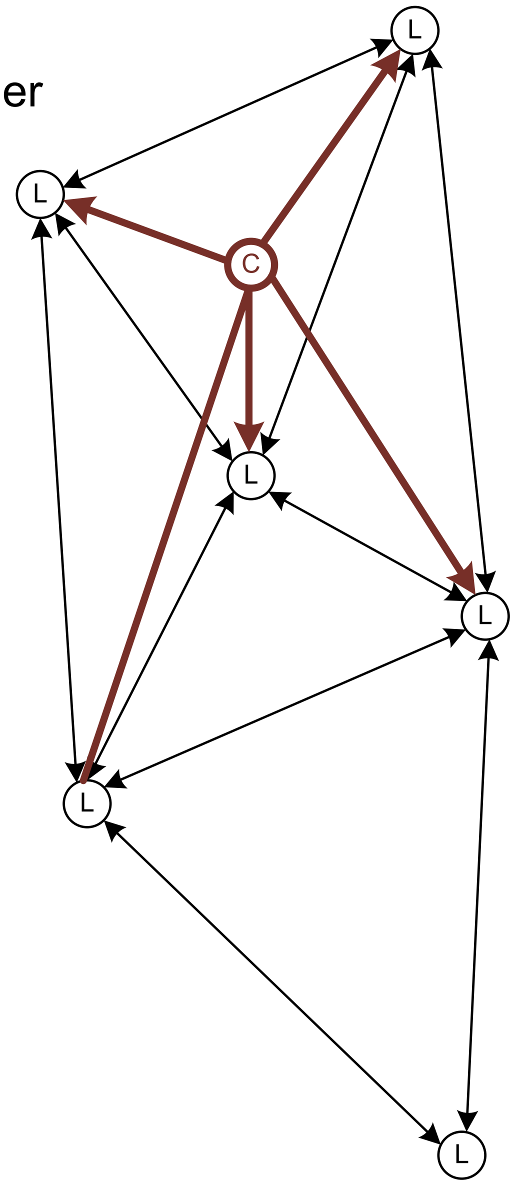
Triangle inequality (II)

- Possible spaces for embedding are metric
  - Distance function satisfies triangle inequality
- Embedding can not be exact
  - Number and weight of TIVs limits embedding quality



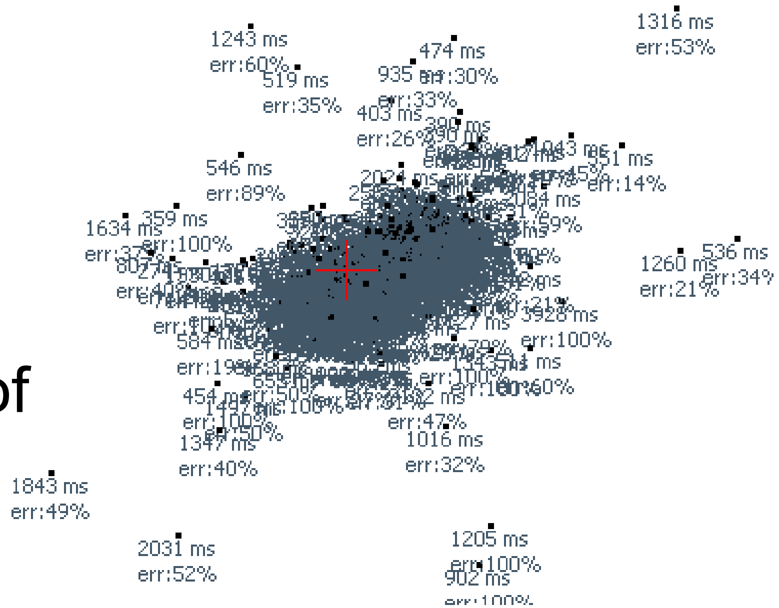
History

- Global Network Positioning (Ng, Zhang, 2002)
  - Landmark nodes measure distance between eachother
  - New nodes measure distance to landmarks
  - Coordinates relative to landmarks
  - Embedding via Downhill-Simplex in 3D space
  - Problems:
    - Scalability
    - Placement of landmarks
    - Single point of failure
- Lighthouse (Pias et al., 2003)
  - Several groups of landmarks
- PIC (Costa, Castro, Rowstron, Key, 2004)
  - Generalization of GNP
  - All nodes with known coordinates can be landmarks
- Big-Bang-Simulation (2004)
  - Analogy to physics: nodes as particles in a force field



Vivaldi (Dabek, Cox, Kaashoek, Morris, 2004)

- Fully distributed
  - No infrastructure, no specialized nodes
- Continuous upgrade of coordinates with new latency values
- Based on application traffic
- Small number of communication partners required for meaningful results
- Can be used with various types of spaces
- State of the art
- Actively used (e.g. bittorrent, azureus)



Vivaldi Algorithm

1. Choose random (obviously wrong) position
2. Initiate communication with some nodes
3. Measure latency
4. Nodes provide coordinates and error estimation
5. Revise coordinates (relative to other nodes)

Optimization

- due to TIVs and measurement errors
  - No exact embedding in low-dimensional spaces
  - Requires at most n-1 dimensions

- Optimization problem
  - Minimize error
    (= difference between real and estimated latency)

$$E = \sum_i \sum_j (L_{ij} - \|x_i - x_j\|)^2$$

$\|x_i - x_j\|$: distance between coordinates i, j

L_{ij} : measured latency

  - Distance depends on space

Optimization (II)

- Spring Embedder
  - Physical analogy: network of springs
  - Between each pair (i,j) of hosts exists a spring
    - Length in equilibrium position: L_{ij}
    - Current length: $||x_i - x_j||$
    - Potential energy proportional to expansion squared: $(L_{ij} - ||x_i - x_j||)^2$
      - Energy of the spring = error
      - Minimal energy in the system = minimal global error
  - Force between i and j (Hooks law)

$$F_{ij} = (L_{ij} - \|x_i - x_j\|) \times u(x_i - x_j)$$

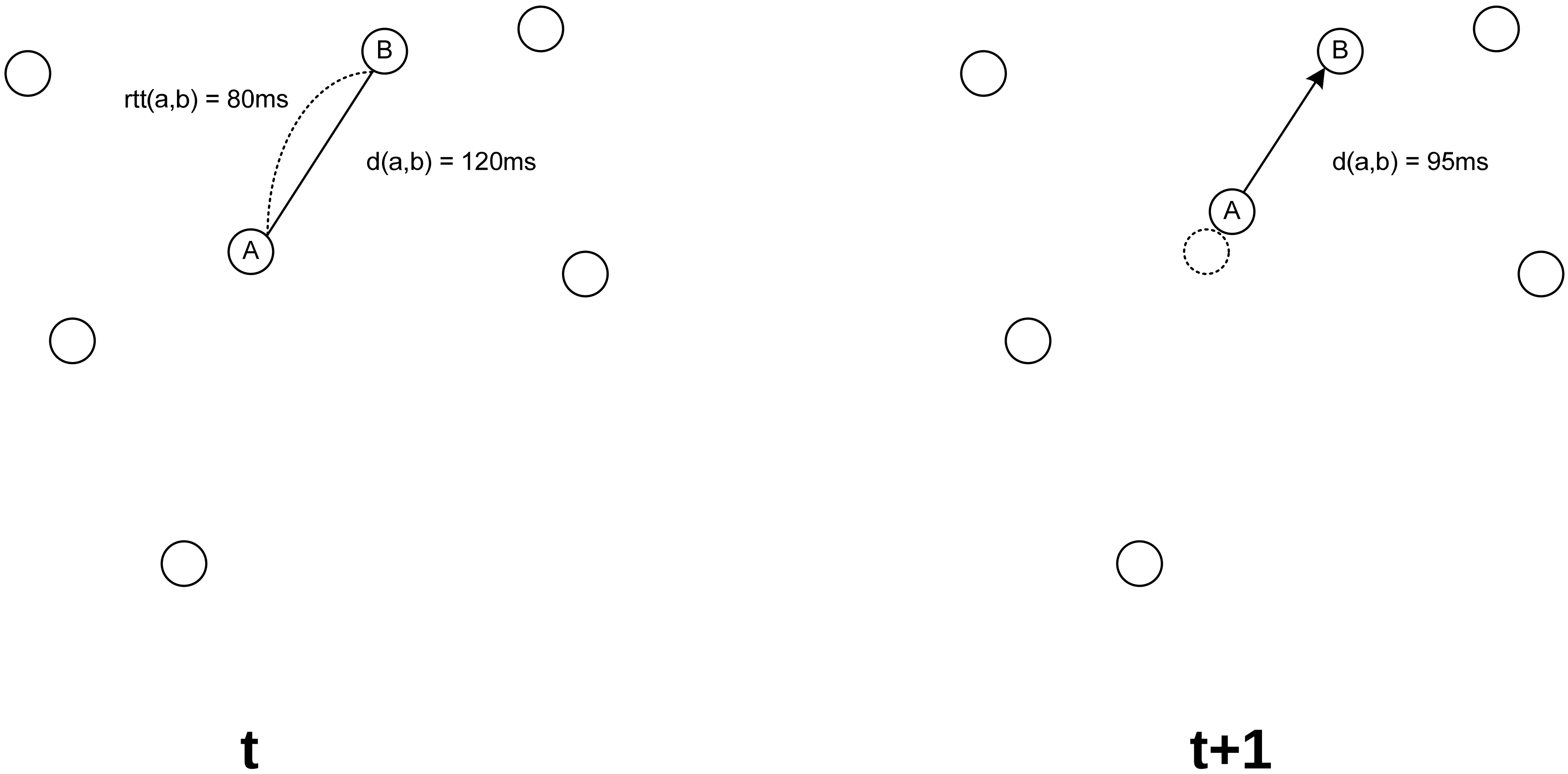
  - Move node to minimize its energy

Optimization (III)

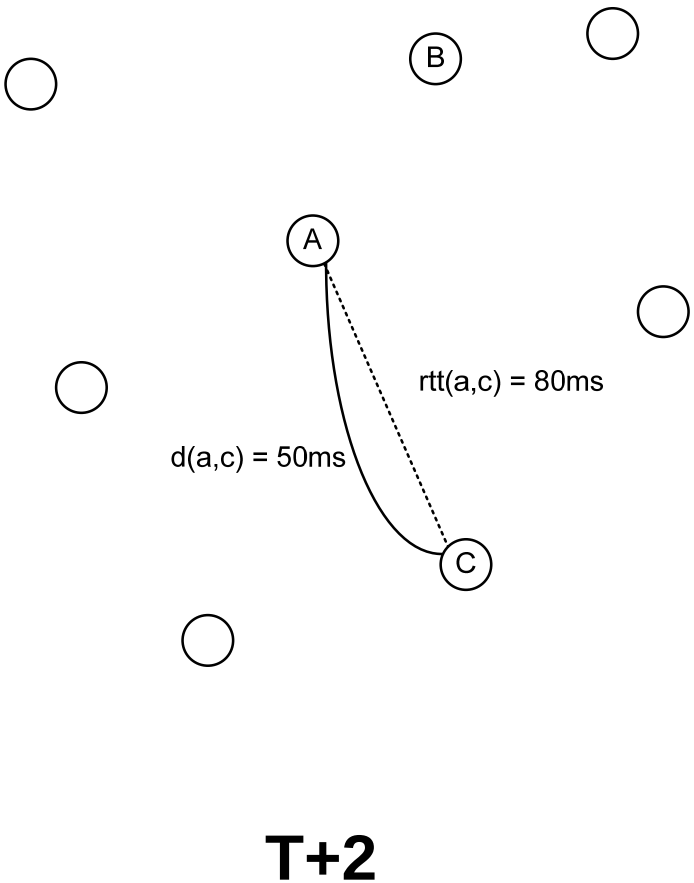
- Local
  - Iteratively move each node I by $\delta \cdot F_i$ per step
    δ = attenuation
- Global/ distributed
  - Each node calculates its coordinates
  - Large attenuation: oscillation
  - Small attenuation: slow convergence
  - Small impact of coordinates with high error
    - Adaptive attenuation

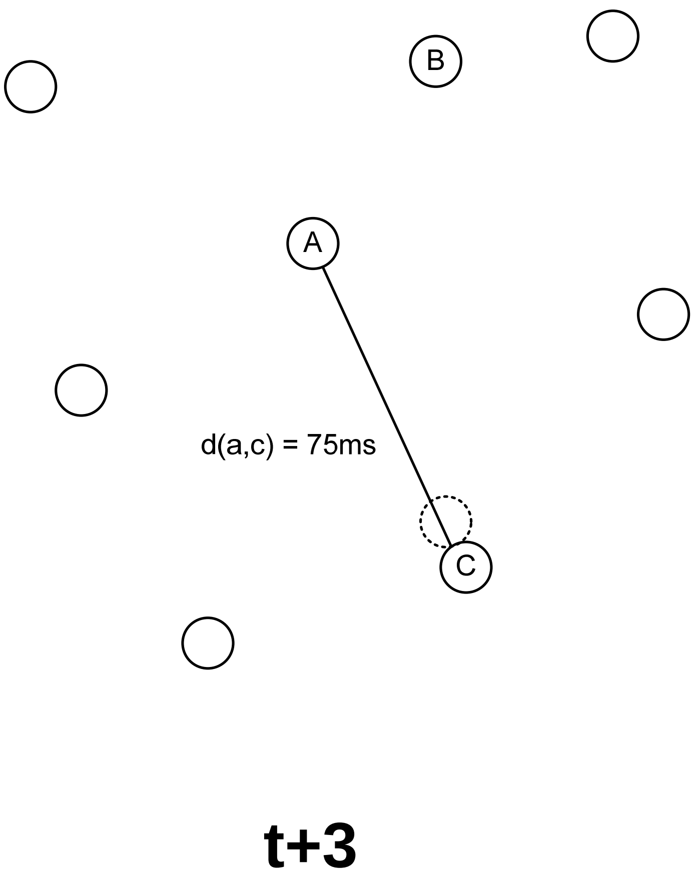
$$\delta = c_c \cdot \frac{e_i}{e_i + e_j}, c_c \text{constant}$$

Example



Example (II)





Which space to choose?

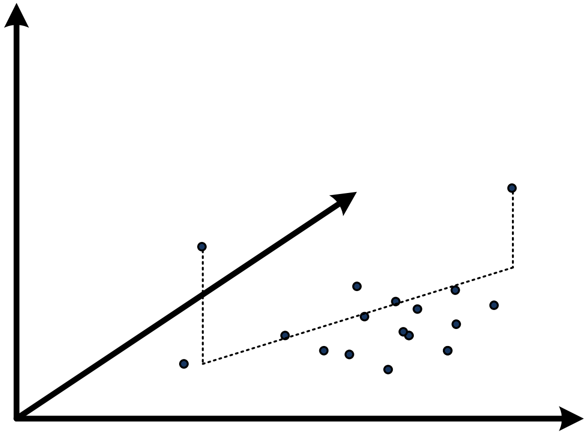
- Physics:
  - Anology uses 3D space
  - Any space with a definition of distance, difference between coordinates and scalar multiplication possible

- Question:
  Which space characterizes the Internet most?
  - 2D, 3D
  - Sphere, torus
  - Complex network → complex space?
  - From GNP: embedding in 3D, why?

- Result from tests and simulations:
  - 2-3 dimension sufficient
  - More dimensions require more computation without significant improvement

Handling TIVs

- Again:
  - TIVs occur for asymmetric routes, links, …
  - Occur quite often
  - Enlarge the error for the embedding
- Instead of using n dimensions, use n-1 + height
  - Euclidean n-space models the core network
    - High connectivity
    - Fast, symmetric links
  - Height models the slow access links
  - Packets are transmitted in the core, not above it
  - Slow hosts are pushed out of the plane



Some results

- Error below 20% for 80. percentile (2D+H)
- Spherical coordinates do not improve the result
- Adaptive attenuation improve result
- Neighbors
  - < 32: bad results
  - > 64: no improvements
  - Best results with a mixture of near and distant neighbors
- Lookup times in DHTs improved by 30% for 80% of the nodes

- Problems
  - Instability due to churn, latency fluctuation
    - Neighbor decay
    - Latency filter
    - Update filter
  - Drift

Security

- Attacks
  - Disorder
    - Maximize error in coordinates
    - Denial of service
  - Isolation/ Repulsion
    - Move target into “isolated space”
    - Convince target that another node is far away
    - Redirect target to malicious node, replica server
      - Man in the middle attack
- Mitigation based on statistics
  - Classify nodes into bad/good via their behaviour

Cross Layer Considerations

Crosslayer considerations

- Network stack
  - Encapsulation of functionality
  - No knowledge required in upper layers about how the network works

But

- Protocols and applications make assumptions on the underlying network
  - Network might change over time
  - Assumptions might not be correct for all parts of the network
- Diverse underlay

- Example: TCP
  - Loss = congestion
  - Increasing delay = upcoming congestion
  - Long delay = narrow bandwidth
  - Are these assumptions still true?
    - Wireless networks
    - Satellite links
    - …

- Include information from different layers in the network stack: Cross layer approach

Implications for measurements

- Upper layers are not fully isolated from the underlay
- Network types and condition might change the outcome

- Questions that should be answered:
  - Does the measurement change the network and how? Does the network condition changes the measurement?
    - UMTS RTT measurements
    - Number of nodes in a WLAN
  - Are the assumption made for the measurement evaluation correct for this specific network
    - Dependency between delay and bandwidth
    - The way the underlay acts on losses (WLAN vs. Ethernet)

Conclusion

- Localization of nodes
  - Required for many purposes
  - GeoIP
  - Network coordinates

- Cross layer considerations
  - Take all layers into account while measuring

Thanks for listening!

Questions?

Master Course
Computer Networks
IN2097

Prof. Dr.-Ing. Georg Carle,
Christian Grothoff, Ph.D.,
Dr. Nils Kammenhuber,
Dirk Haage

Chair for Network Architectures and Services
Department of Computer Science
Technische Universität München
http://www.net.in.tum.de

Chapter 7:

Network Measurements

Acknowledgements:

The content of this chapter
is partly based on slides from Anja Feldmann

Chapter 7 Outline – Network Measurements

- Recapitulation: Why do we measure and how?

- Network Traffic
  - Traffic patterns
  - Traffic characterization
  - Traffic models
  - Self-similar traffic

- Interpretation of measurement data
  - Before you start
  - Statistics 1-0-1
  - Dos and don´ts

Why do we measure the network?

- Network Provider View
  - Manage traffic
    - Predict future, model reality, plan network
    - Avoid bottlenecks in advance
  - Reduce cost
  - Accounting

- Client View
  - Get the best possible service
  - Check the service („Do I get what I´ve paid for?)

- Service Provider View
  - Get information about the client
  - Adjust service to demands
  - Reduce load on service
  - Accounting

But why?

- The network is well engineered
- Well documented protocols, mechanisms, …
- In theory we can know everything that is going on
- ⇨ There should be no need for measurements

- But:
  - Moving target:
    - requirements change
    - growth, usage, structure changes
  - Highly interactive system
  - Heterogeneity in all directions
  - The total is more than the sum of its pieces

- And: The network is built, driven and used by humans
  - Detection of errors, misconfigurations, flaws, failures, misuse, …

Measurement types

- Active Measurements
  - Intrusive
  - Find out what the network is capable of
  - Changes the network state
- Passive Measurements (or network monitoring)
  - Non-intrusive
  - Find out what the current situation is
  - Does not influence the network state (more or less)
- Hybrid
  - Alter actual traffic
  - Reduce the impact of active measurements
  - Might introduce new bias for applications

Network Traffic

Traffic by Port (I)

18 hours of traffic to AT&T dial clients on July 22, 1997

Name	Port	% Bytes	% Packets	Bytes/Packet
www	80	56,75	44,79	819
nntp	119	24,65	12,90	1235
pop3 email	110	1,88	3,17	384
cuseeme	7648	0,95	1,85	333
secure www	443	0,74	0,79	603
irc	6667	0,27	0,74	239
ftp	20	0,65	0,64	659
dns	53	0,19	0,58	210
…				

Traffic by Port (II)

24 hours of traffic to/from MWN clients in 2006

Name	Port	% Conns	% Succes	%Payload
www	80	70,82	68,13	72,59
cifs	445	3,53	0,01	0,00
secure www	443	2,34	2,08	1,29
ssh	22	2,12	1,75	1,71
smtp	25	1,85	1,05	1,71
	1042	1,66	0,00	0,00
	1433	1,06	0,00	0,00
	135	1,04	0,00	0,00
	< 1024	83,68	73,73	79,05
	> 1024	16,32	4,08	20,95

Traffic by Port (III)

- Port 80 dominates traffic mix
  - Still growing
    - More web applications
    - Tunnel everything over port 80
- Characterization of traffic by port is possible
  - Well-known ports
    (1-1024, take a look at /etc/services)
- Growing margin of error
  - Automatic configuration
  - * over http – VPN, P2P, VoIP, …
  - Aggressive applications (e.g. Skype):
    „just find me an open port“

Traffic Flows

18 hours of traffic to AT&T dial clients on July 22, 1997

Name	Port	% Bytes	% Pkts	Bytes/ Pkt	% Flows	Pkts/ Flow	Duration (s)
www	80	56,75	44,79	819	74,58	12	11,2
nntp	119	24,65	12,90	1235	1,20	210	132,6
pop3 email	110	1,88	3,17	384	2,80	22	10,3
cuseeme	7648	0,95	1,85	333	0,03	1375	192,0
secure www	443	0,74	0,79	603	0,99	16	14,2
irc	6667	0,27	0,74	239	0,16	89	384,6
ftp	20	0,65	0,64	659	0,26	47	30,1
dns	53	0,19	0,58	210	10,69	1	0,5
…							

Elephants and Mices

- Many very short flows (30% < 300 bytes)
- Many medium-sized flows (short web transfers)
- Most bytes belong to long flows (large images, files, flash, video)
- Same picture for other metrics
  - Bytes/flow
  - Packets/flow
  - lifetime
- Flow densities are traffic patterns and signatures

More ways to classify traffic

- Distribution of flows over time
- Distribution of packets over time
  - Globally
  - Within a flow
- Distribution of packet sizes

- Payload, Deep Packet Inspection
  - Expensive (time, processing power)
  - Does not work with encrypted traffic
  - Can also be used for intrusion detection
    - Trojans, viruses

Self-Similar Traffic

- It has shown that for some environments the traffic pattern is self-similar rather than Poisson

- Self-similarity is a concept related to two others
  - Fractals
  - Chaos theory

- Statement by Manfred-Schroeder:

  The unifying concept underlying fractals, chaos, and power laws is self-similarity. Self-similarity, or invariance against changes in scale or size, is an attribute of many laws in nature and innumerable phenomena in the world around us. Self-similarity is, in fact, one of the decisive symmetries that shape our universe and our effort to comprehend it.

Self-Similarity – An Example

- Network monitoring, analysis of the interarrival time of single frames
- Minimum transmission time for one frame: 4ms

- Recorded arrivals (ms):

  0 8 24 32 72 80 96 104 216 224 240 248 288 296 312 320

  648 656 672 680 720 728 744 752 864 872 888 896 936 944 960 968

- Clustering all samples with gaps smaller than 20ms:

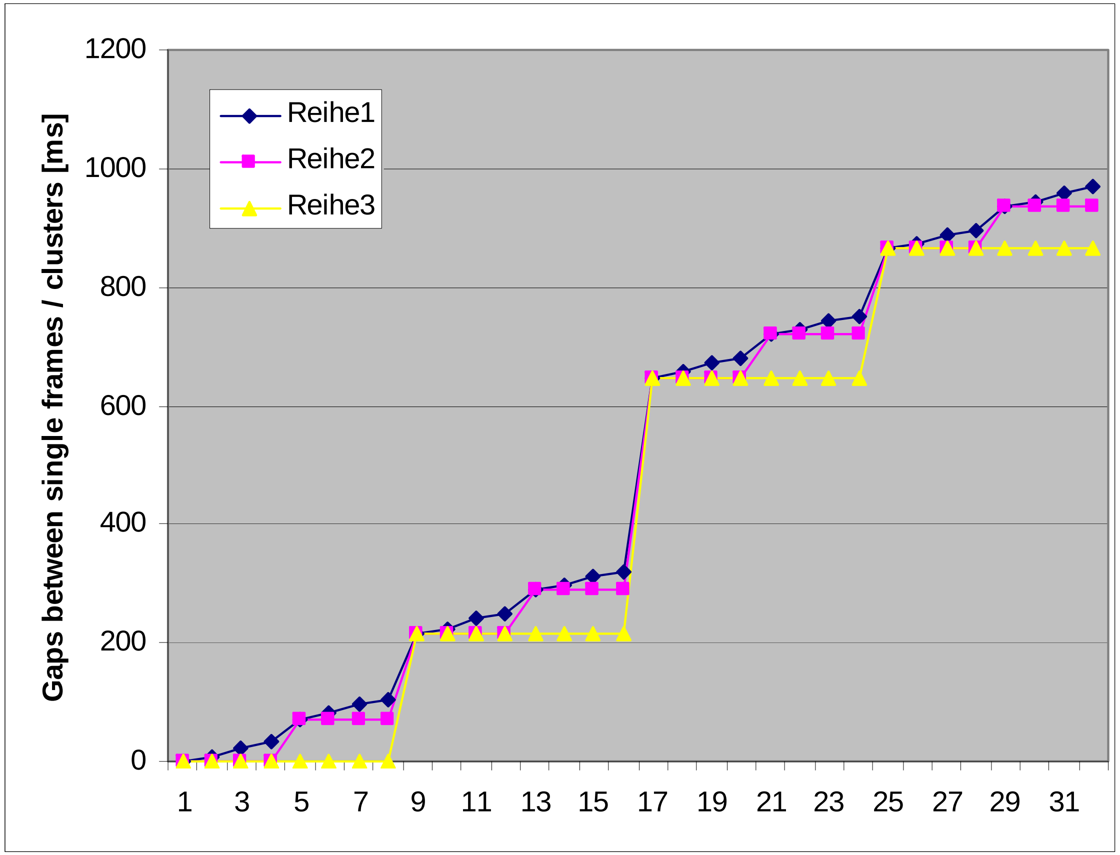
  0 72 216 288 648 720 864 936

- Clustering all samples with gaps smaller than 40ms:

  0 216 648 864

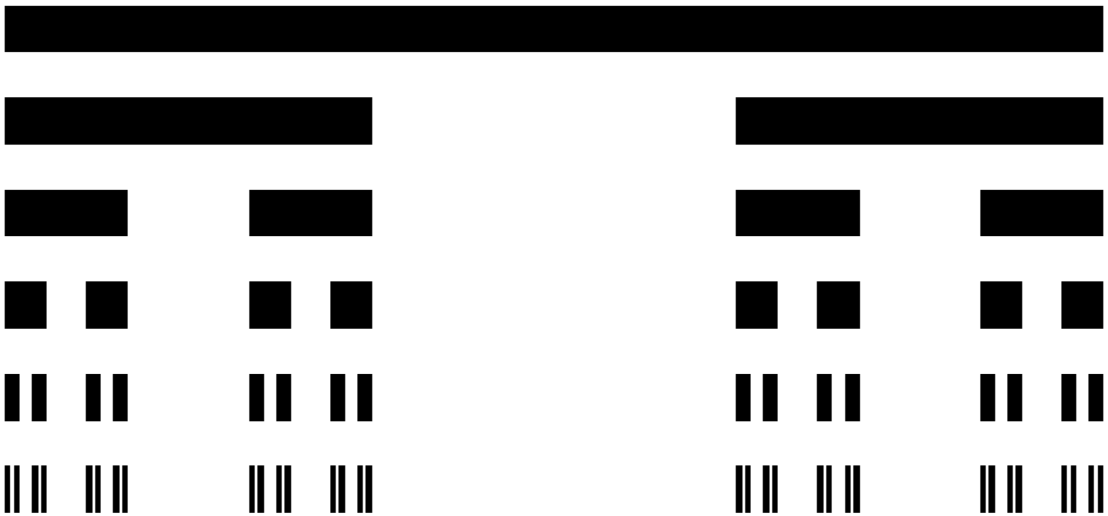
Self-Similarity – An Example II

- Repeating patterns: arrival, short gap, arrival, long gap, arrival, short gap, arrival)



Cantor Set

- Famous construct appearing in virtually every book on chaos, fractals, and nonlinear dynamics
- Construction rules:
  - Begin with the closed interval [0,1], represented by a line segment
  - Remove the open middle third of a line
  - For each succeeding step, remove the middle third of the lines left by the preceding step
- Cantor set:
  - $S_0 = [0, 1]$
  - $S_1 = [0, 1/3] \cup [2/3, 1]$
  - $S_3 = [0, 1/9] \cup [2/9, 1/3] \cup [2/3, 7/9] \cup [8/9, 1]$



Cantor Set II

- Properties of Cantor sets seen in all self-similar phenomena
  - It has a structure at arbitrarily small scales. If we magnify part of the set repeatedly, we continue to see a complex pattern of points separated by gaps of various sizes. The process seems unending. In contrast, when we look at a smooth, continuous curve under repeated magnification, it becomes more and more featureless.
  - The structure repeat. A self-similar structure contains smaller replicas of itself at all scales. For example, at every step, the left (and right) portion of the Cantor set is an exact replica of the full set in the preceding step.

- These properties do not hold indefinitely for real phenomena. At some point under magnification, the structure and the self-similarity break down. But over a large range of scales, many phenomena exhibit self-similarity.

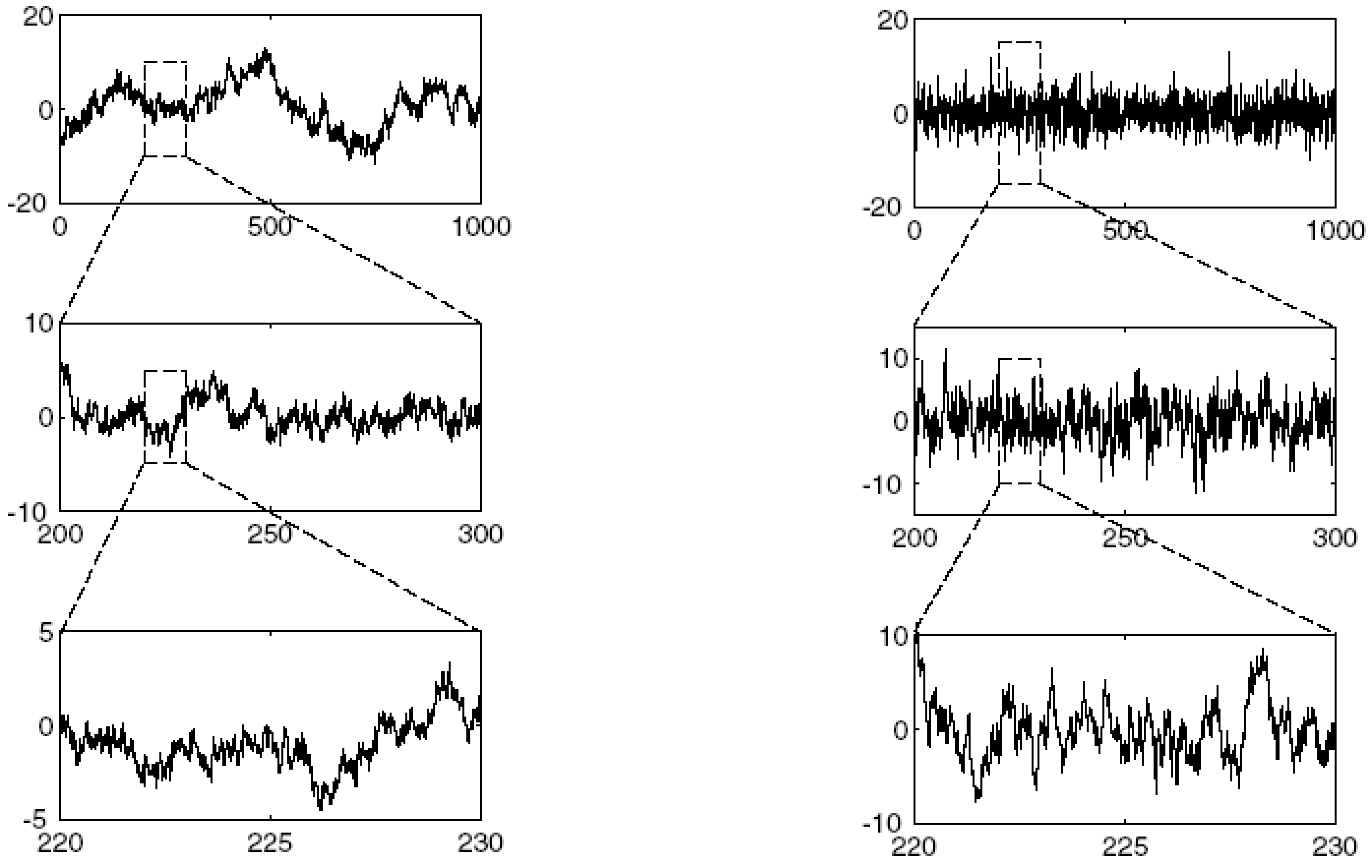
Stochastical Self-Similarity

- So far, we examined exact self-similarity:
  A pattern is reproduced exactly at different scales

- Data traffic is a stochastic process, therefore we talk about statistical self-similarity.
- For a stochastic process, we say that the statistics of the process do not change with the change in the time scale. The average behavior of the process in the short-term is the same as it is in the long term.

- Examples
  - Data traffic
  - Earthquakes
  - Ocean waves
  - Fluctuations in the stock market

Self-Similar Stochastic Process



(a) Self-Similar Process

(b) Non-Self-Similar Process

Network Traffic Characteristics

- Traffic characteristics experienced in the network
  - Changes over time
  - Varies in many dimensions
  - Each application has its characteristic traffic pattern
  - Must match the model used for planning

- Numerous ways of classification
  - Port, Flow sizes, Packet sizes, Packet count, Arrival times, …

- Packet/ Flow/… distribution
  - Poisson
    - Good for performance evaluation, network planning
  - Gauss, Pareto, …
  - Self-similarity

Interpretation of Measurement Results

Literature:

Raj Jain: The Art of Computer Systems Performance Analysis, John Wiley

D.C. Montgomery “Design and Analysis of Experiments”

- „If you require a straight curve, only measure two times“

- „If you can´t reproduce a result, only conduct the experiment once“

- „post hoc ergo propter hoc“
  „from coincidence follows correlation“

Before you start a measurement

- Wanted:
  - Answer to a question
- To be considered:
  - Correctness
  - Significance (of the measured values)
  - Relevance (in regard to the question)
  - Effort

- Modelling the reality
  - Simplify to much
  - Forget important parameters
  - Make assumptions that make life easy

- Modelling our tools: overfitting
  - Change the behaviour of our measurement tool so it works perfectly in the tests
  - What happens in other scenarios?

- Example: a new TCP flavor and we want to know how it performs
  - Cross-traffic: static/dynamic, distribution, number of flows/packets/…?
  - Underlying network: layer 2, topology, …?
  - What did we want to measure again? – ah, the performance:
    - Delay, recovery time, throughput, startup time, …?

Statistics

- Why do we need it?
  - Transform data into information
  - Get rid of noise

- Statistic:
  - Merriam-Webster:
    „A quantity that is computed from a sample [of data]“
  - A single number to summarize a larger collection of values
- Statistic***s***:
  - Merriam-Webster:
    „A branch of mathematics dealing with the collection, analysis, interpretation, and presentation of masses of numerical data.“
  - Analysis and interpretation

Sampling the measured data

- Sample = subset of whole process
  - Not possible to enumerate fully
    - too much data
    - ongoing process
- Selection types
  - Random
  - Systematic – every nth packet, flow, …

- Sample Bias
  - Selection area
    - only use a “good” part of the data
    - Partition the data based on knowledge
  - Interval – start and end at a convenient time
  - Exposure – selection is not independent from the process itself
  - Rejection of „bad“ data, outliers, …
  - Overmatching
  - Quantization error

- Examples
  - Heise Browser Statistics
  - Counting the number of cars on the street every Monday at 9:00

The simplest statistic: a mean

- Reduce sample to a single number
- But what does it mean?
  - Tries to capture the „center“ of a distribution of values
    - Mean
    - Median
    - Mode
  - Use this „center“ to summarize
  - „Sample“ implies
    - Values are measured from a discrete random variable
    - Only an approximation of the underlying process
    - True mean value cannot be known
      (requires infinite number of measurements)

- To provide „mean“ value
  - Understand how to choose the best type
  - Detect bad results

Arithmetic Mean

- Common „average“

$$\bar{x}_{arithm} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

- Potential problems
  - Equal weight to all values
  - Outliers can have a large influence
  - Distorts our intuition about central tendency

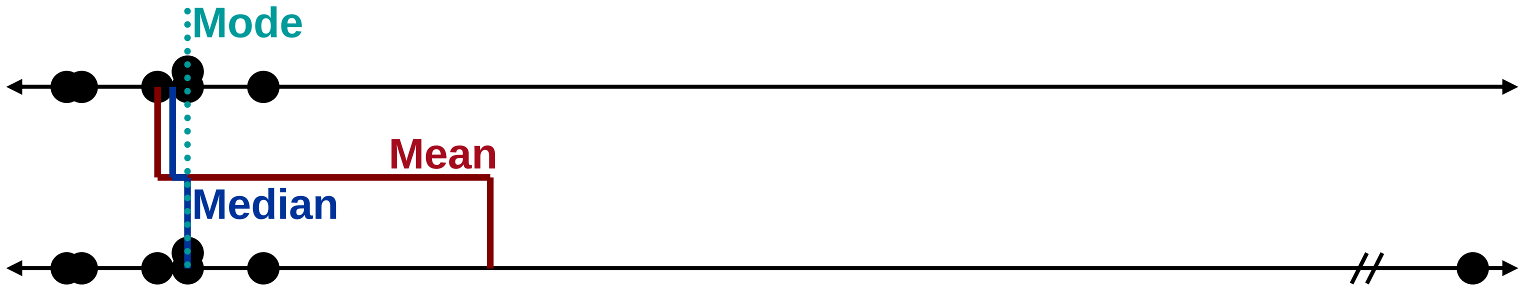
Median and Mode

- Median
  - ½ of the values larger, ½ smaller
  - Algorithm
    - Sort n measurements by value
    - If n is odd: Median = middle value
    - Else: Median = mean of two middle values
  - Reduces skewing effect of outliers

- Mode
  - Value that occurs most often
  - May not exist
  - May not be unique: multiple modes
    - e.g. „bi-modal“ distribution:
      Two values occur with same frequency

Mean vs. Median vs. Mode

- Measured Values: 10, 23, 16, 18, 18, 11
  - Mean: 16
  - Median: 17
  - Mode: 18
- Obtain one more measurement: 173
  - Mean: 38
  - Median: 18
  - Mode: 18



Mean, mode or median?

- Mean
  - If the sum of all values is meaningful
  - Incorporates all information
- Median
  - Intuitive Sense of central tendency with outliers
  - What is „typical" of a set of values?
- Mode
  - When data can be grouped into distinct types, categories

Other means

- Geometric
  - Growth rates, benchmarks
  - Example:
    The usage of a webservice doubles the first year and octuplicates the second year
    - Geometric mean: 4
    - Arithmetic mean: 5
  - Less sensible

$$\bar{x}_{geom} = \sqrt[n]{\prod_{i=1}^{n} x_i}$$

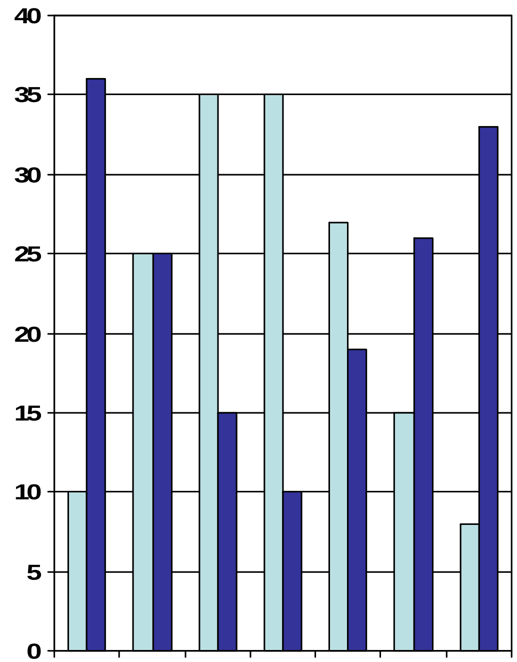
- Harmonic
  - Proportional data, ratios
  - Example:
    Download 10MB of data with 1MB/s, 5MB/s and 10MB/s (10s+2s+1s=13s)
    - Harmonic Mean: 2,33 MB/s
      and: 30 MB with 2,33MB/s: 13s
    - Arithmetic mean: 5,33 MB/s
      but: 30MB with 5,33MB/s: 5,625s
    - Download per time is again arithmetic!

$$\bar{x}_{harm} = \frac{n}{\sum_{i=1}^{n} x_i}$$

Variability

- How spread out are the values?
- How much spread relative to the mean?
- What is the shape of the distribution
- A mean hides information about variability

- Example
  - Similiar mean values
  - Widely different distribution



- How to capture this in one number?

Dispersion

- Range: max-min
- 10- and 90- percentiles
- Maximum distance from mean
  max (| x_i-mean |)
- Neither efficiently incorporates all available information

- Variance
  - Squares of the distances to mean
  - Gives „units-squared“ – hard to compare with

$$\mathrm{var} = s^2 = \frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n}$$

- Standard deviation ***s***
  - Square root of variance
  - Same unit as mean

Expectation value

- Also called mean ☹ or first moment
- Limit of sample mean for infinite number of values

- Not „the most probable value“
  - Expectation value might be unlikely or even impossible
  - rolling a dice: Expectation value: 3.5

- „Law of large numbers“
  - Information for large scales
  - No information about single events/ small samples!

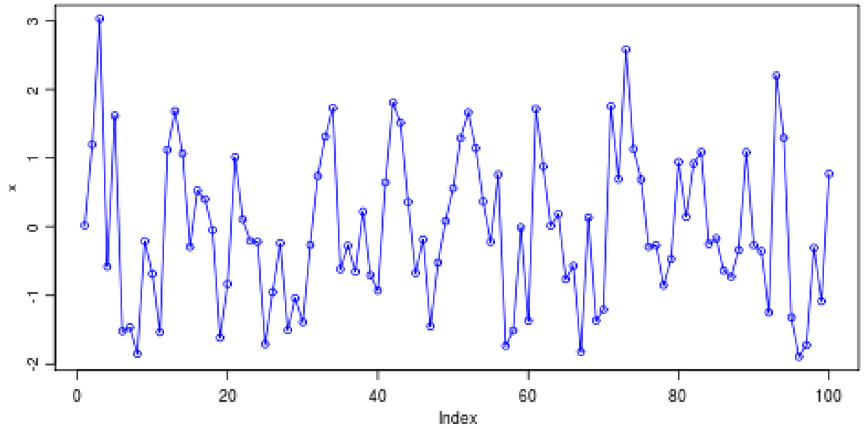
Autocorrelation

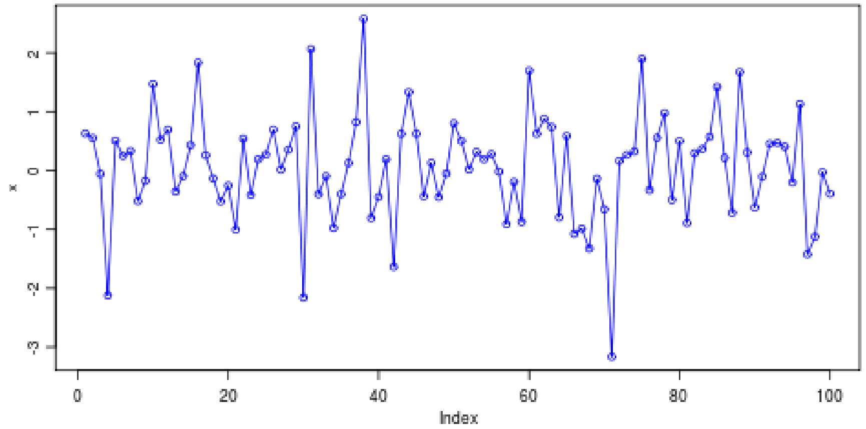
- Correlation of a signal with itself
  - Checking for randomness
    - Most standard statistical tests rely on randomness (validity of the test is directly linked to the validity of the randomness assumption)
    - In short: If you don not check for randomness, the validity of your conclusions are questionable
  - Find repeating patterns (e.g. underlying frequencies)

- Concept
  - Calculate variance C_0 for data set
  - For each lag
    - Calculate variance C_h over the data set
    - Normalize C_h/C_0

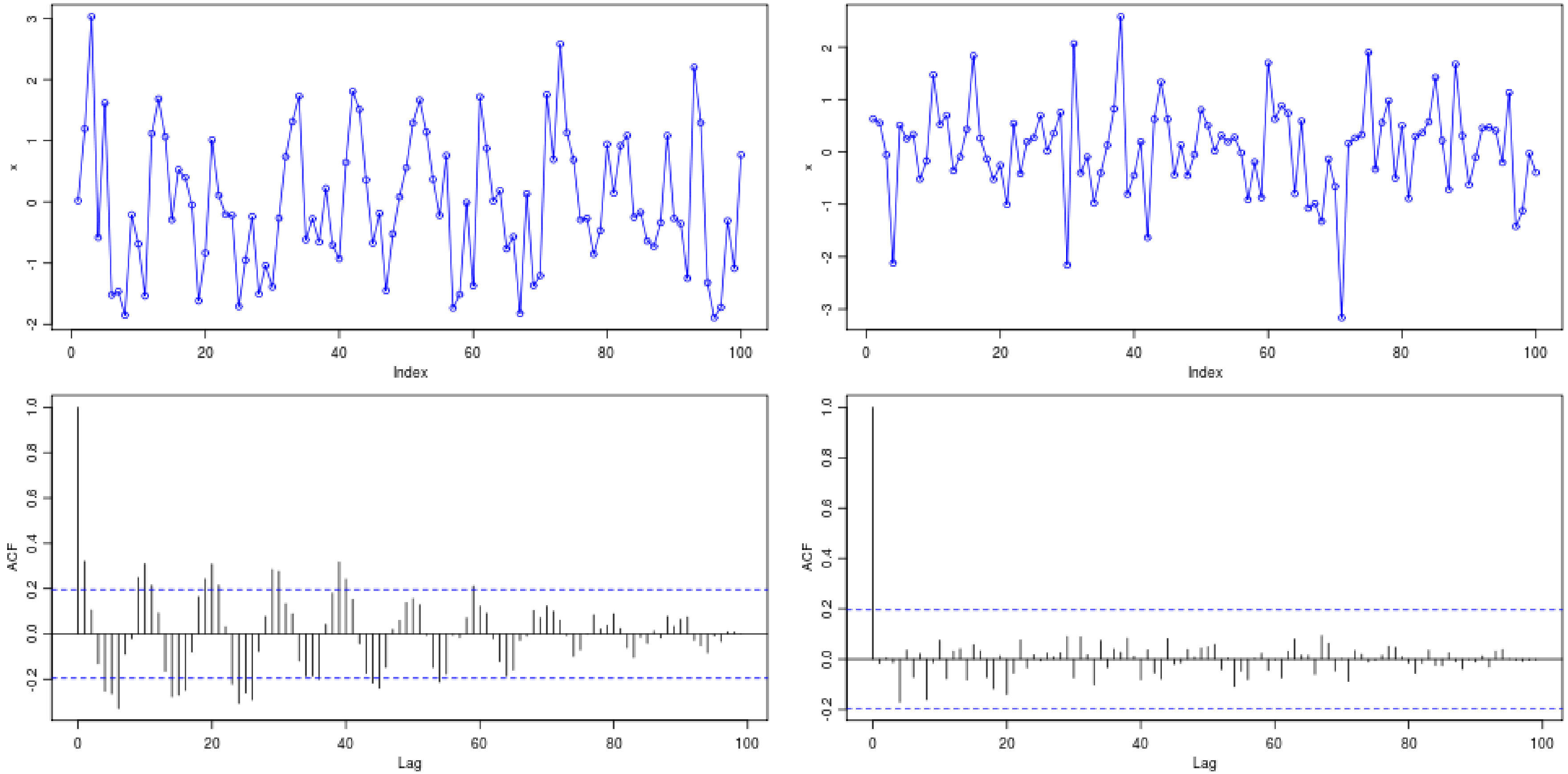
- Interpretation
  - If random: near zero for all and any lag separations
  - If non-random: one or more autocorrelations significantly non-zero
  - Lag shows the frequency for the autocorrelation

Autocorrelation Example (1)





Autocorrelation Example (2)



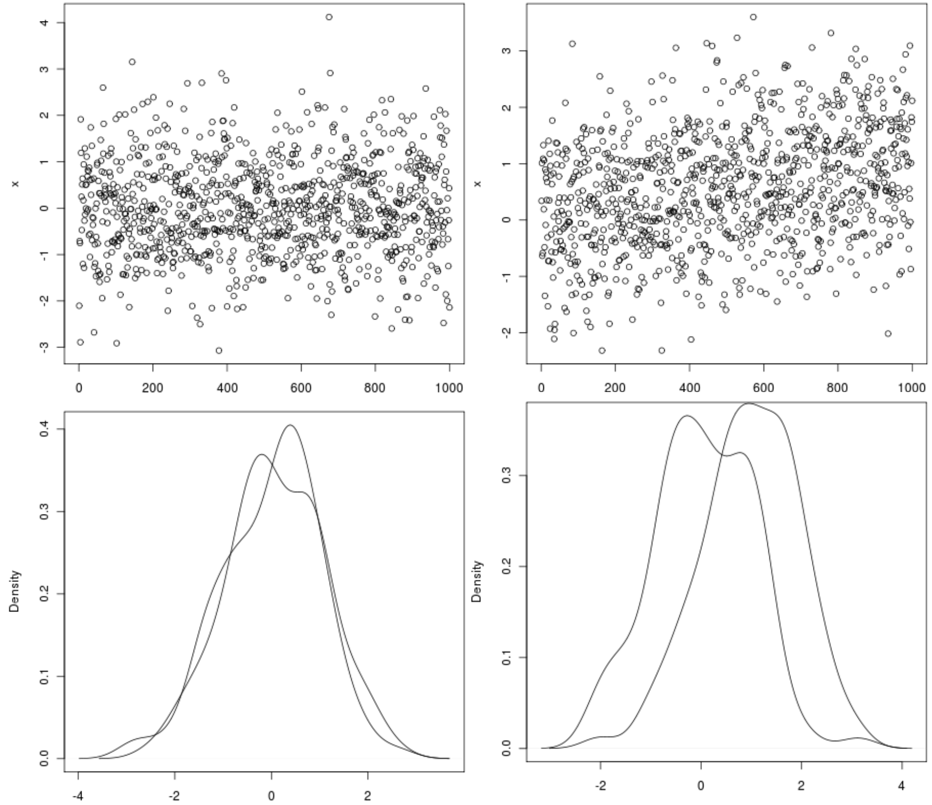
- ❑ Vertical Axis: Autocorrelation coefficient
  - Always between -1 and +1
- ❑ Horizontal Axis: Time lag
- ❑ Reference lines: 95% confidence band

- ❑ Left: hidden sinus
  - Positive autocorrelation for f
  - Negative autocorrelation for f/2
- ❑ Right: „truly“ random

Stationarity

- a random process where all of its statistical properties do not vary with time

- First order stationary process
  - Mean,
    variance,
    autocorrelation
    do not change over time
- Example:
  - Random
  - Random with trend



- Transformation to achieve stationarity
  - Take the diffs between values
  - Trend:
    fit some type of curve
    (e.g. a straight line), model
    residuals from that fit
  - Non-constant variance:
    try square root or logarithm
    to stabilize the data

Summary

- Network Measurement
  - Why, what and how?

- Network Traffic
  - Traffic Pattern
  - Traffic Models
  - Self-similar traffic

- Evaluation of measurements
  - Statistics
    - Only the tip of the iceberg
    - Common Errors!
    - Think before you start, before you calculate, before you extrapolate!
    - Be careful in every step
  - If you want to play with this
    - Octave – www.gnu.org/software/octave/
    - R – www.r-project.org

Master Course
Computer Networks
IN2097

Prof. Dr.-Ing. Georg Carle
Christian Grothoff, Ph.D.
Dr. Nils Kammenhuber

Chair for Network Architectures and Services
Institut für Informatik
Technische Universität München
http://www.net.in.tum.de

Signalling - Introduction

Design principles:

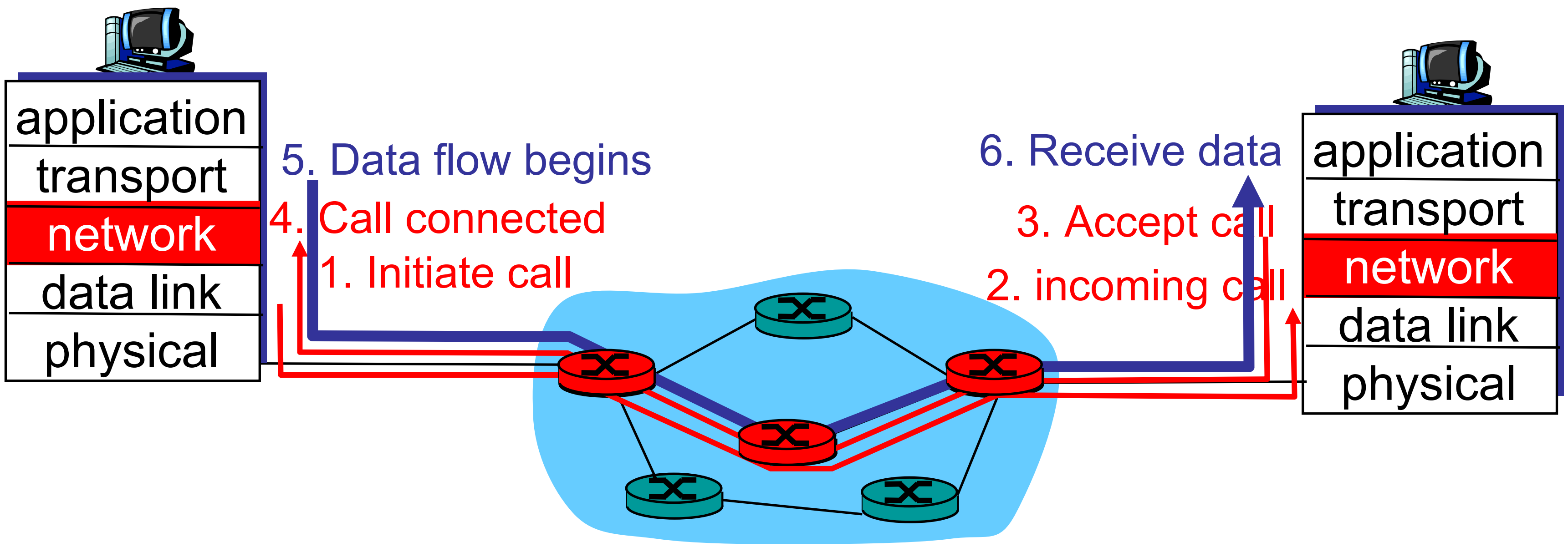
- network virtualization: overlays
- separation of data, control
- hard state versus soft state
- randomization
- indirection
- multiplexing
- design for scale

Goals:

- identify, study common architectural components, protocol mechanisms
- what approaches do we find in network architectures?
- *synthesis:* big picture

Virtual circuits: signaling protocols

- used to set up, maintain teardown VC
- used in (G)MPLS, ATM, frame-relay, X.25
- not used in today's Internet at L3 (network layer)



Signaling

signaling: exchange of messages among network entities to enable (provide service) to connection/call

- before, during, after connection/call
  - call setup and teardown (state)
  - call maintenance (state)
  - measurement, billing (state)
- between
  - end-user <-> network
  - end-user <-> end-user
  - network element <-> network element
- examples
  - Q.921, SS7 (Signaling System no. 7): telephone network
  - Q.2931: ATM
  - RSVP (Resource Reservation Protocol)
  - H.323: Internet telephony
  - **SIP** (Session Initiation Protocol): Internet telephony

SIP

Credits in addition to
Jim Kurose and Keith Ross:

Julie Chan, Vovida Networks.
Milind Nimesh, Columbia University
Christian Hoene, University of Tübingen

SIP: Session Initiation Protocol [RFC 3261]

SIP long-term vision:

all telephone calls, video conference calls take place over Internet

- people are identified by names or e-mail addresses, rather than by phone numbers
- you can reach callee, no matter where callee roams, no matter what IP device callee is currently using

SIP key person:

Henning Schulzrinne, Columbia University

- M. Handley, H. Schulzrinne, and E. Schooler, “SIP: session initiation protocol,” Internet Draft, Internet Engineering Task Force, March 1997. Work in progress.
- H. Schulzrinne, A comprehensive multimedia control architecture for the Internet, 1997

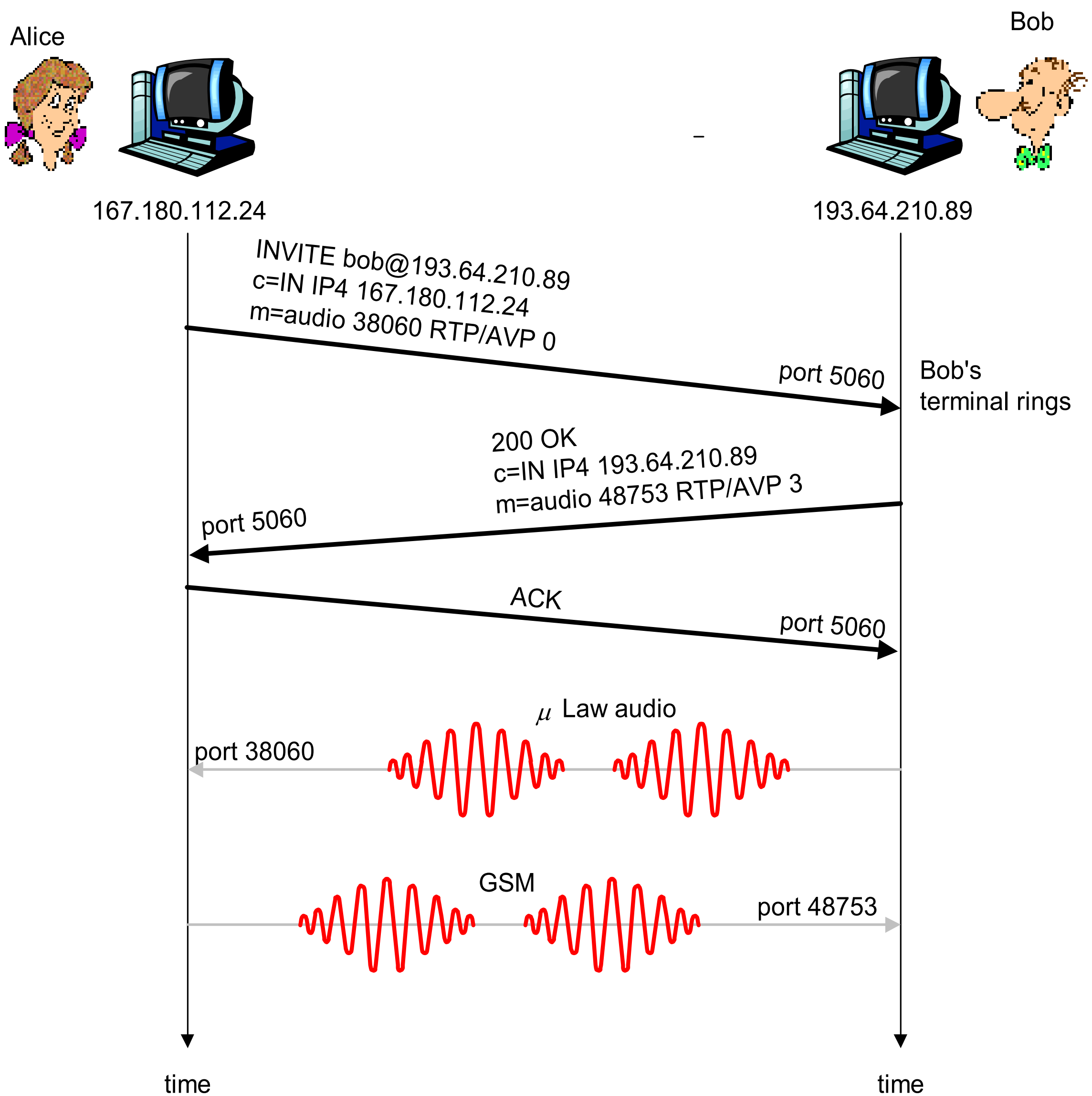
SIP

- IETF RFC 2543: Session Initiation Protocol –
  An application layer signalling protocol that defines initiation, modification and termination of interactive, multimedia communication *sessions* between users.
- Sessions include
  - voice
  - video
  - chat
  - interactive games
  - virtual reality
- SIP is a text-based protocol, similar to HTTP and SMTP.

SIP Services

- Setting up a call, SIP provides mechanism
  - for caller to let callee know she wants to establish a call
  - so caller, callee can agree on media type, encoding
  - to end call
- determine current IP address of callee:
  - maps mnemonic identifier to current IP address
- call management:
  - add new media streams during call
  - change encoding during call
  - invite others
  - transfer, hold calls

Setting up a call to known IP address



- Alice's SIP invite message indicates her port number, IP address, encoding she prefers to receive (e.g. AVP 0: PCM ulaw)
- Bob's 200 OK message indicates his port number, IP address, preferred encoding (e.g. AVP 3: GSM)
- SIP is an out-of-band signalling protocol
- SIP messages can be sent over TCP or UDP. (All messages are ack'ed)
- default SIP port number is 5060.

Setting up a call (more)

- codec negotiation:
  - suppose Bob doesn't have PCM ulaw encoder.
  - Bob will instead reply with 606 Not Acceptable Reply, listing his encoders
  - Alice can then send new INVITE message, advertising different encoder
- rejecting a call
  - Bob can reject with replies "busy," "gone," "payment required," "forbidden"
- media can be sent over RTP or some other protocol

Example of SIP message

INVITE sip:bob@domain.com SIP/2.0
Via: SIP/2.0/UDP 167.180.112.24
From: sip:alice@hereway.com
To: sip:bob@domain.com
Call-ID: a2e3a@pigeon.hereway.com
Content-Type: application/sdp
Content-Length: 885

c=IN IP4 167.180.112.24
m=audio 38060 RTP/AVP 0

- Here we don't know Bob's IP address. Intermediate SIP servers needed.
- Alice sends, receives SIP messages using SIP default port 5060
- Via: header specifies intermediate server(s)

Notes:

- HTTP message syntax
- sdp = session description protocol
- Call-ID is unique for every call.

Name translation and user location

- caller wants to call callee, but only has callee's name or e-mail address.
- need to get IP address of callee's current host:
  - user moves around
  - DHCP protocol
  - user has different IP devices (PC, PDA, car device)
- result can be based on:
  - time of day (work, home)
  - caller (e.g. don't want boss to call you at home)
  - status of callee (calls sent to voicemail when callee is already talking to someone)

Service provided by SIP servers:

- SIP registrar server
- SIP proxy server

SIP Registrar

- when Bob starts SIP client, client sends SIP REGISTER message to Bob's registrar server

  (similar function needed by Instant Messaging)
- registrar analogous to authoritative DNS server

Register Message:

```
REGISTER sip:domain.com SIP/2.0
Via: SIP/2.0/UDP 193.64.210.89
From: sip:bob@domain.com
To: sip:bob@domain.com
Expires: 3600
```

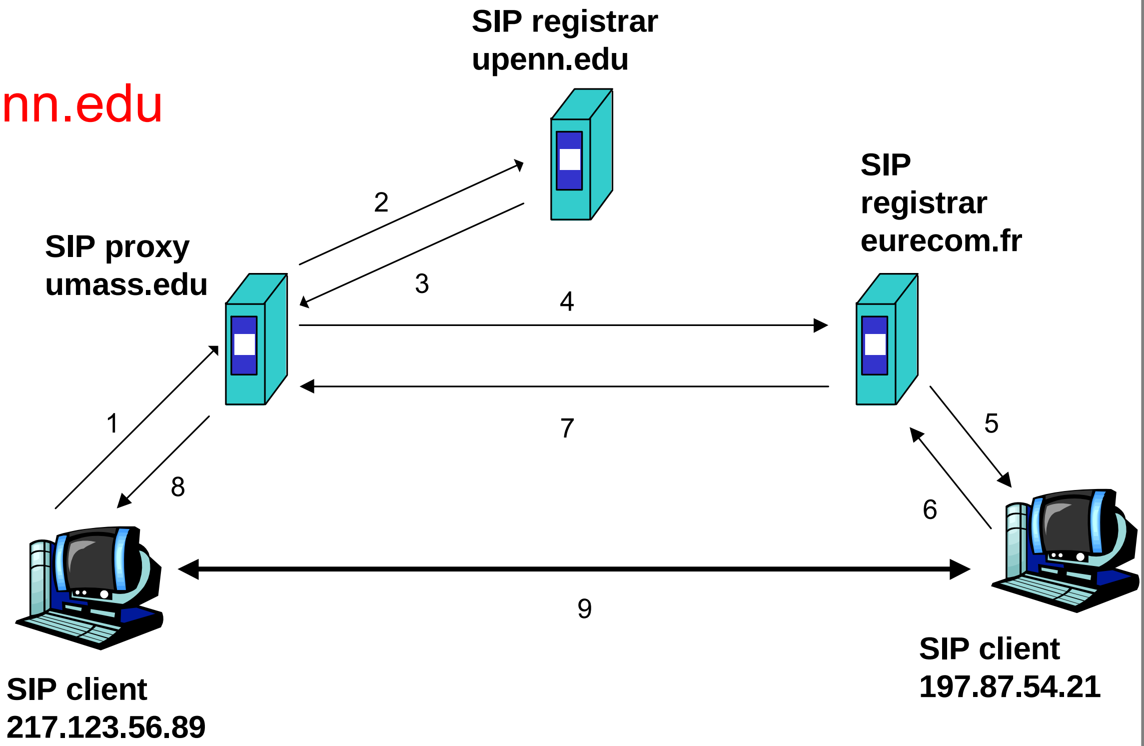
SIP Proxy

- Alice sends invite message to her proxy server
  - contains address sip:bob@domain.com
- proxy responsible for routing SIP messages to callee
  - possibly through multiple proxies.
- callee sends response back through the same set of proxies.
- proxy returns SIP response message to Alice
  - contains Bob's IP address
- proxy analogous to local DNS server

Example

Caller jim@umass.edu
places a call to keith@upenn.edu

(1) Jim sends INVITE message to umass SIP proxy.
(2) Proxy forwards request to upenn registrar server.
(3) upenn server returns redirect response, indicating that it should try keith@eurecom.fr

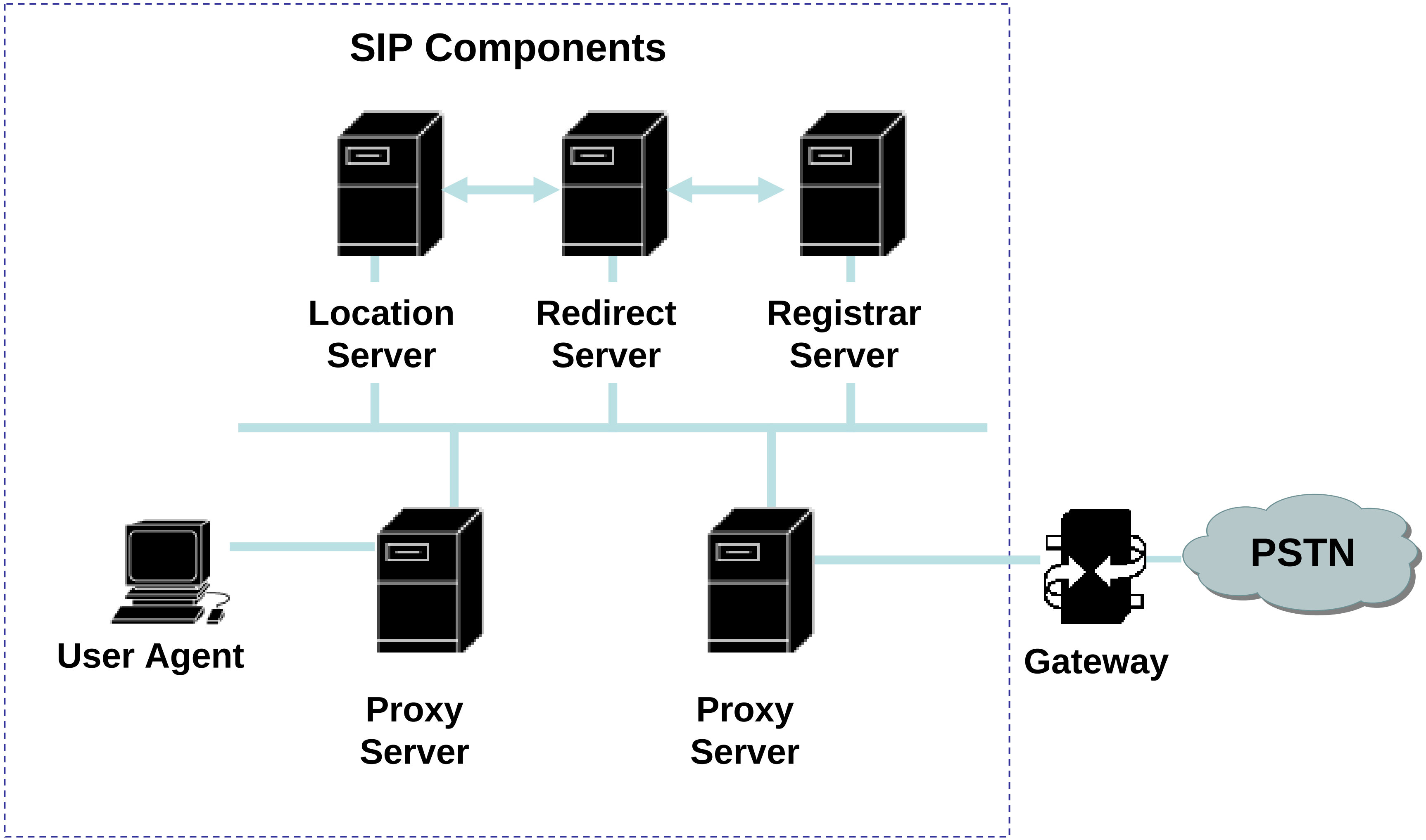


(4) umass proxy sends INVITE to eurecom registrar.
(5) eurecom registrar forwards INVITE to 197.87.54.21, which is running keith's SIP client.
(6-8) SIP response sent back
(9) media sent directly between clients.
Note: SIP ack messages not shown.

SIP consists of a few RFCs

RFC	Description
2976	The SIP INFO Method
3361	DHCP Option for SIP Servers
3310	Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA)
3311	The Session Initiation Protocol UPDATE Method
3420	Internet Media Type message/sipfrag
3325	Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks
3323	A Privacy Mechanism for the Session Initiation Protocol (SIP)
3428	Session Initiation Protocol Extension for Instant Messaging
3326	The Reason Header Field for the Session Initiation Protocol (SIP)
3327	Session Initiation Protocol Extension for Registering Non-Adjacent Contacts
3329	Security Mechanism Agreement for the Session Initiation Protocol (SIP) Sessions
3313	Private Session Initiation Protocol (SIP)Extensions for Media Authorization
3486	Compressing the Session Initiation Protocol
3515	The Session Initiation Protocol (SIP) Refer Method
3319	Dynamic Host Configuration Protocol (DHCPv6)Options for Session Initiation Protocol (SIP) Servers
3581	An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing
3608	Session Initiation Protocol Extension Header Field for Service Route Discovery During Registration
3853	S/MIME AES Requirement for SIP
3840	Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)
3841	Caller Preferences for the Session Initiation Protocol (SIP)
3891	The Session Inititation Protocol (SIP) 'Replaces' Header
3892	The SIP Referred-By Mechanism
3893	SIP Authenticated Identity Body (AIB) Format
3903	An Event State Publication Extension to the Session Initiation Protocol (SIP)
3911	The Session Inititation Protocol (SIP) 'Join' Header
3968	The Internet Assigned Number Authority (IANA) Header Field Parameter Registry for the Session Initiation Protocol (SIP)
3969	The Internet Assigned Number Authority (IANA) Universal Resource Identifier (URI) Parameter Registry for the Session Initiation Protocol (SIP)
4032	Update to the Session Initiation Protocol (SIP) Preconditions Framework
4028	Session Timers in the Session Initiation Protocol (SIP)
4092	Usage of the Session Description Protocol (SDP) Alternative Network Address Types (ANAT) Semantics in the Session Initiation Protocol (SIP)
4168	The Stream Control Transmission Protocol (SCTP) as a Transport for the Session Initiation Protocol (SIP)
4244	An Extension to the Session Initiation Protocol (SIP) for Request History Information
4320	Actions Addressing Identified Issues with the Session Initiation Protocol's (SIP) non-INVITE Transaction
4321	Problems identified associated with the Session Initiation Protocol's (SIP) non-INVITE Transaction
4412	Communications Resource Priority for the Session Initiation Protocol (SIP)
4488	Suppression of Session Initiation Protocol (SIP) REFER Method Implicit Subscription
4508	Conveying Feature Tags with Session Initiation Protocol (SIP) REFER Method
4483	A Mechanism for Content Indirection in Session Initiation Protocol (SIP) Messages
4485	Guidelines for Authors of Extensions to the Session Initiation Protocol (SIP)

SIP Architecture



User Agents

- An application that initiates, receives and terminates calls.
  - User Agent Clients (UAC) – An entity that initiates a call.
  - User Agent Server (UAS) – An entity that receives a call.

  - Both UAC and UAS can terminate a call.

Proxy Server

- An intermediary program that acts as both a server and a client to make requests on behalf of other clients.
- Requests are serviced internally or passed on, possibly after translation, to other servers.
- Interprets, rewrites or translates a request message before forwarding it.

Registrar Server

- A server that accepts REGISTER requests.
- The registrar server may support authentication.
- A registrar server is typically co-located with a proxy or redirect server and may offer location services.

Redirect Server

- A server that accepts a SIP request, maps the address into zero or more new addresses and returns these addresses to the client.
- Unlike proxy server, the redirect server does not initiate own SIP requests
- Unlike a user agent server, the redirect server does not accept or terminate calls.
- The redirect server generates 3xx responses to requests it receives, directing the client to contact an alternate set of URIs.
- In some architectures it may be desirable to reduce the processing load on proxy servers that are responsible for routing requests, and improve signaling path robustness, by relying on redirection.
- Redirection allows servers to push routing information for a request back to the client, thereby taking themselves out of the loop of further messaging while still aiding in locating the target of the request.
  - When the originator of the request receives the redirection, it will send a new request based on the URI(s) it has received.
  - By propagating URIs from the core of the network to its edges, redirection allows for considerable network scalability.
- C.f. iterative (non-recursive) DNS queries

Location Server

- A location server is used by a SIP redirect or proxy server to obtain information about a called party's possible location(s).

- A location Server is a logical IP server that transmits a Presence Information Data Format - Location Object (PIDF-LO).
- A PIDF-LO is an XML Scheme for carrying geographic location of a target.
- As stated in RFC 3693, location often must be kept private. The Location Object (PIDF-LO) contains rules which provides guidance to the Location Recipient and controls onward distribution and retention of the location.

SIP Messages – Methods and Responses

SIP components communicate by exchanging SIP messages:

SIP Methods:

- INVITE – Initiates a call by inviting user to participate in session.
- ACK - Confirms that the client has received a final response to an INVITE request.
- BYE - Indicates termination of the call.
- CANCEL - Cancels a pending request.
- REGISTER – Registers the user agent.
- OPTIONS – Used to query the capabilities of a server.
- INFO – Used to carry out-of-band information, such as DTMF (Dual-tone multi-frequency) digits.

SIP Responses:

- 1xx - Informational Messages.
- 2xx - Successful Responses.
- 3xx - Redirection Responses.
- 4xx - Request Failure Responses.
- 5xx - Server Failure Responses.
- 6xx - Global Failures Responses.

SIP Headers

- SIP borrows much of the syntax and semantics from HTTP.
- A SIP messages looks like an HTTP message:
  message formatting, header and MIME support.
- An example SIP header:

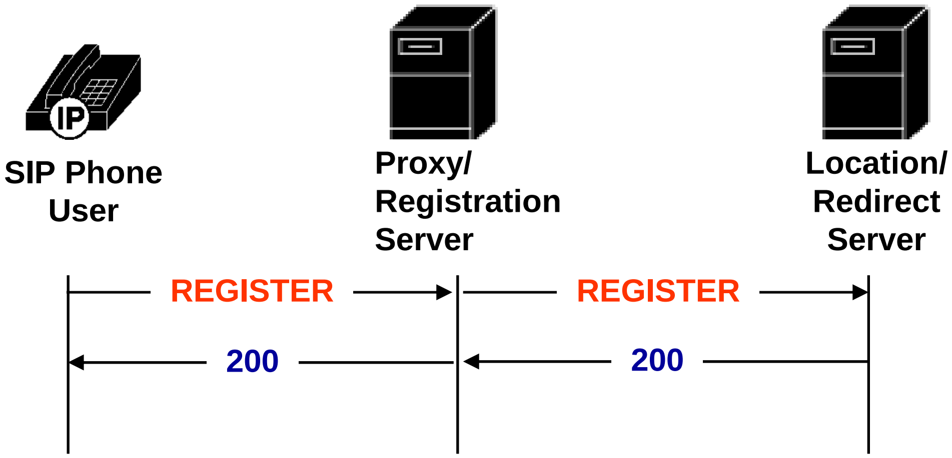
```
--------------------------------------------------------------
                       SIP Header
--------------------------------------------------------------
INVITE sip:5120@192.168.36.180 SIP/2.0
Via: SIP/2.0/UDP 192.168.6.21:5060
From: sip:5121@192.168.6.21
To: <sip:5120@192.168.36.180>
Call-ID: c2943000-e0563-2a1ce-2e323931@192.168.6.21
CSeq: 100 INVITE
Expires: 180
User-Agent: Cisco IP Phone/ Rev. 1/ SIP enabled
Accept: application/sdp
Contact: sip:5121@192.168.6.21:5060
Content-Type: application/sdp
```

SIP Addressing

- The SIP address is identified by a SIP URL, in the format: user@host.
- Examples of SIP URLs:
  - sip:user@domain.com
  - sip:user@192.168.10.1
  - sip:14083831088@domain.com

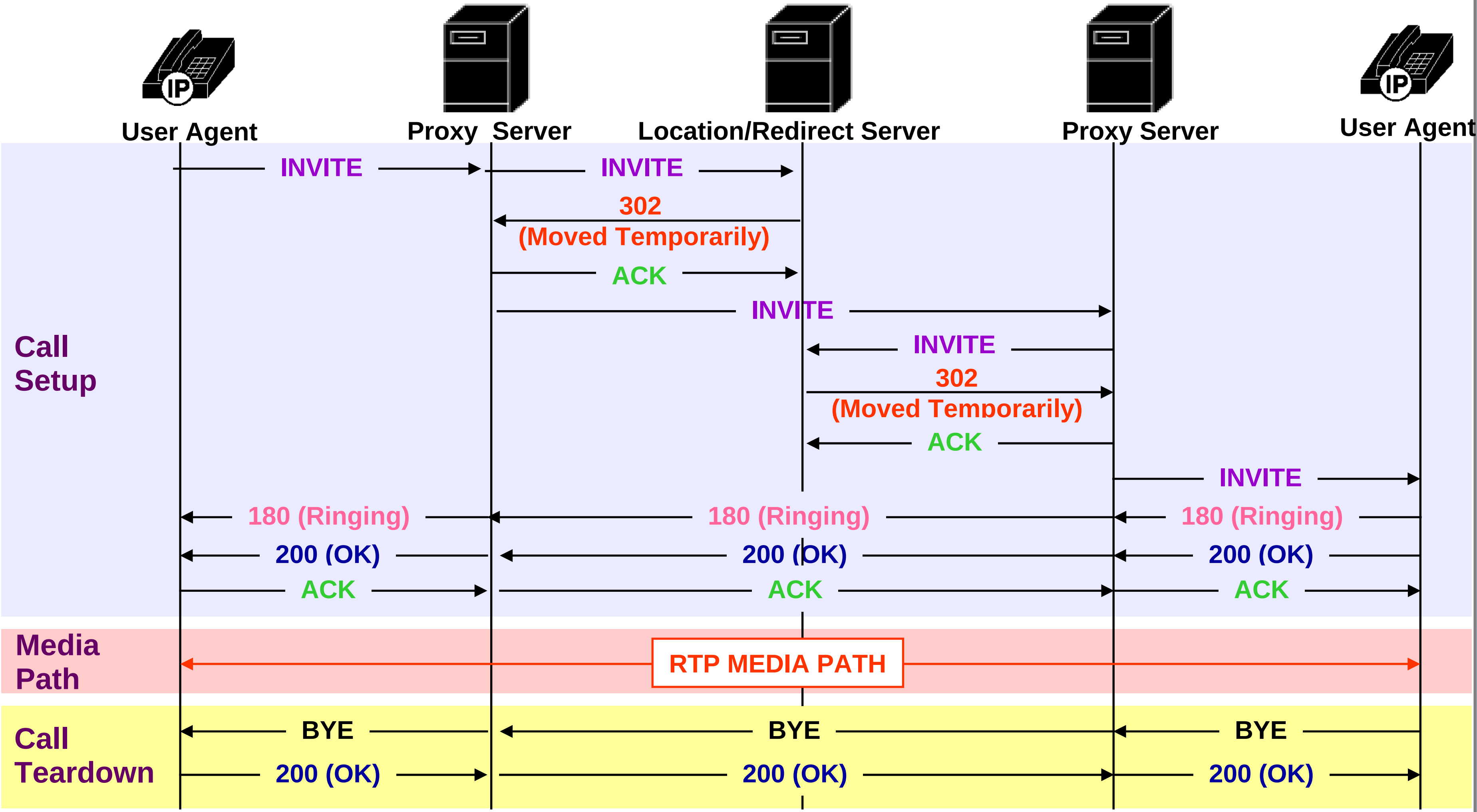
Registration

- Each time a user turns on the SIP user client (SIP IP Phone, PC, or other SIP device), the client registers with the proxy/registration server.
- Registration can also occur when the SIP user client needs to inform the proxy/registration server of its location.
- The registration information is periodically refreshed and each user client must re-register with the proxy/registration server.
- Typically the proxy/registration server will forward this information to be saved in the location/redirect server.



SIP Messages:
REGISTER – Registers the address listed in the To header field.
200 – OK.

Simplified SIP Call Setup and Teardown



SIP – Design Framework

- SIP was designed for:
  - Integration with existing IETF protocols.
  - Scalability and simplicity.
  - Mobility.
  - Easy feature and service creation.

Integration with IETF Protocols

- Other IETF protocol standards can be used to build a SIP based application. SIP works with existing IETF protocols, for example:
  - RTP Real Time Protocol - to transport real time data and provide QOS feedback.
  - SDP Session Description Protocol – for describing multimedia sessions.
  - RSVP - to reserve network resources.
  - RTSP Real Time Streaming Protocol - for controlling delivery of streaming media.
  - SAP Session Advertisement Protocol - for advertising multimedia session via multicast.
  - MIME – Multipurpose Internet Mail Extension – describing content on the Internet.
  - COPS – Common Open Policy Service.
  - OSP – Open Settlement Protocol.

Scalability and Simplicity

- Scalability:
  The SIP architecture is scalable, flexible and distributed.
  - Functionality such as proxying, redirection, location, or registration can reside in different physical servers.
  - Distributed functionality allows new processes to be added without affecting other components.

- Simplicity:
  SIP is designed to be:
  - "Fast and simple in the core."
  - "Smarter with less volume at the edge."
  - Text based for easy implementation and debugging.

Feature Creation

- SIP can support these features and applications:
  - Basic call features (call waiting, call forwarding, call blocking etc.)
  - Unified messaging (the integration of different streams of communication - e-mail, SMS, Fax, voice, video, etc. - into a single unified message store, accessible from a variety of different devices.)
  - Call forking
  - Click to talk
  - Presence
  - Instant messaging
  - Find me / Follow me

Feature Creation (2)

- A SIP based system can support rapid feature and service creation
- For example, features and services can be created using:
  - Common Gateway Interface (CGI).
    - A standard for interfacing external applications with information servers, such as Web servers (or SIP servers).
      A CGI program is executed in real-time, so that it can output dynamic information.
  - Call Processing Language (CPL).
    - Jonathan Lennox, Xiaotao Wu, Henning Schulzrinne: RFC3880
    - Designed to be implementable on either network servers or user agents. Meant to be simple, extensible, easily edited by graphical clients, and independent of operating system or signalling protocol. Suitable for running on a server where users may not be allowed to execute arbitrary programs, as it has no variables, loops, or ability to run external programs.
    - Syntactically, CPL scripts are represented by XML documents.

References

- For more information on SIP:
- IETF
  - http://www.ietf.org/html.charters/sip-charter.html
- Henning Schulzrinne's SIP page
  - http://www.cs.columbia.edu/~hgs/sip/

Location Information and IETF GeoPriv Working Group

credits:

Milind Nimesh, Columbia University

Location Information

- Describes physical position of a person or device:
  - geographical
  - civic (i.e., address)
  - descriptive (e.g. library, airport)
- Formatting and transfer of location information – relatively easy
- Privacy and security – complex
- Application:
  - emergency services
  - resource management
  - social networking
  - search
  - navigation

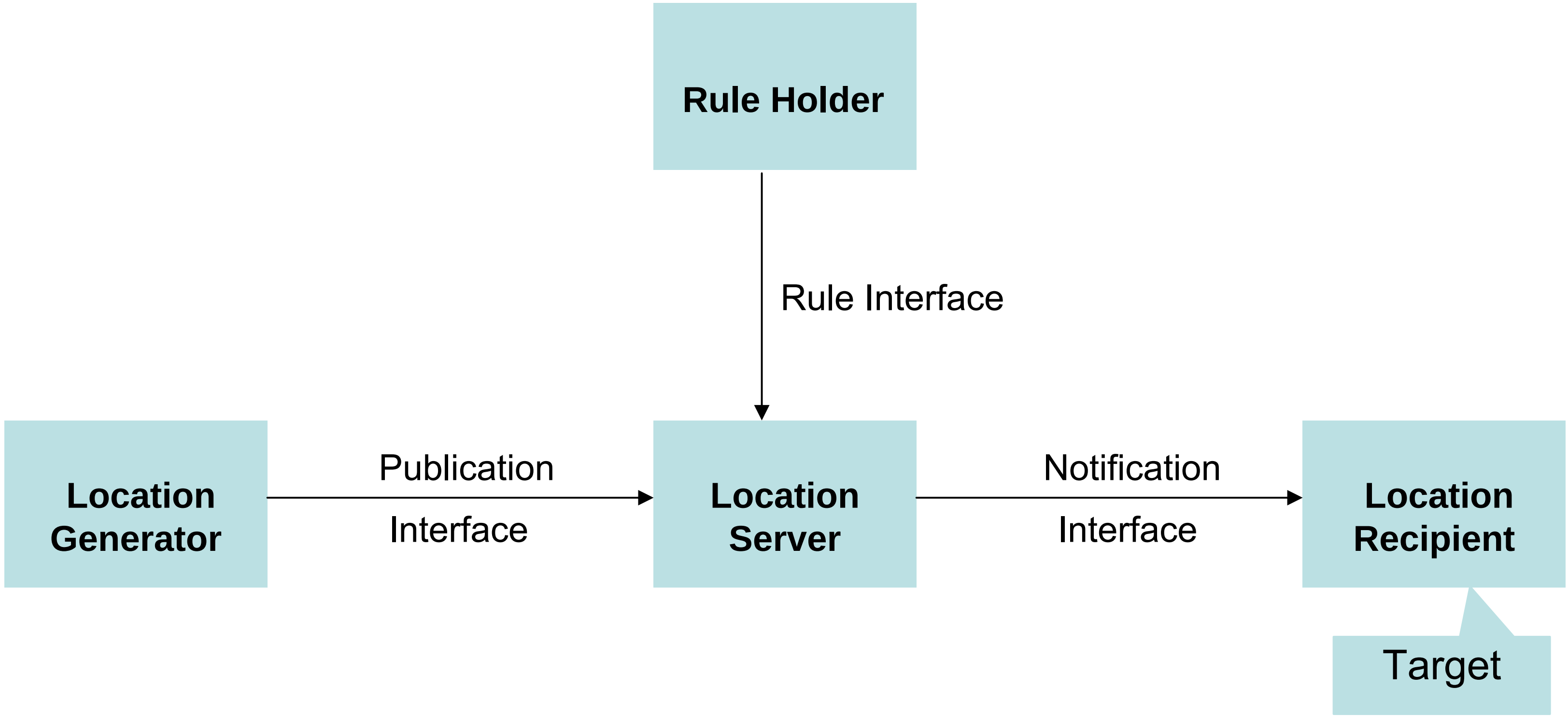
IETF Geopriv Working Group

- Geographic Location/Privacy working group

- Primary tasks for this working group
  - assess authorization, integrity and privacy requirements
  - select standardized location information format
    - enhance format → availability of security & privacy methods
  - authorization of: requester, responders, proxies

- Goal: transferring location information: private + secure

Geopriv Entities



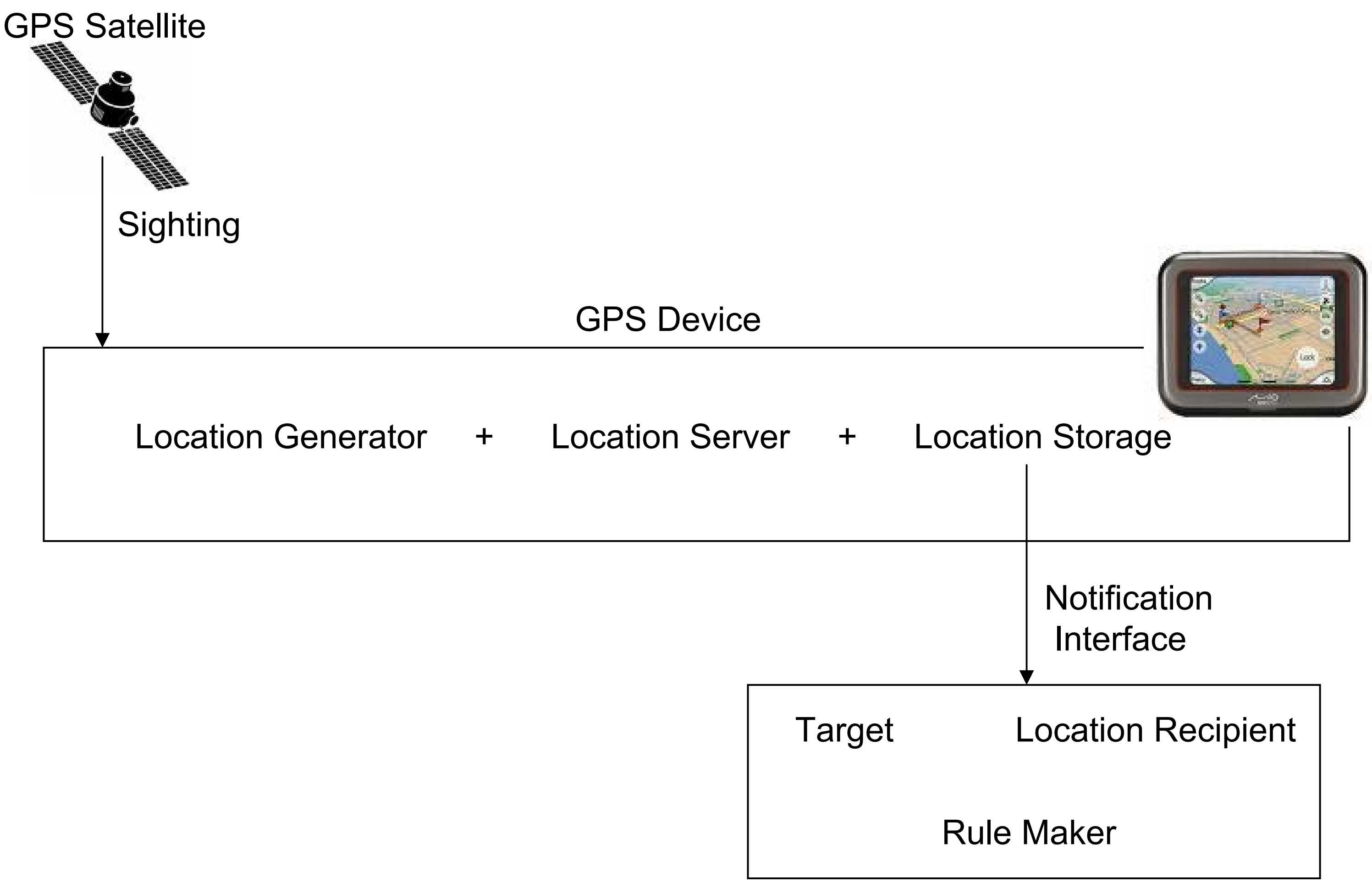
Geopriv Terminology

- Location Object: conveys location information + privacy rules
- Rule Maker: creates rules → governs access to location information
- Target: person/entity whose location communicated
- Using Protocol: protocol carrying location object
- Viewer: consumes location information but does not pass information further
- c.f. RFC 3693

Geopriv Requirements

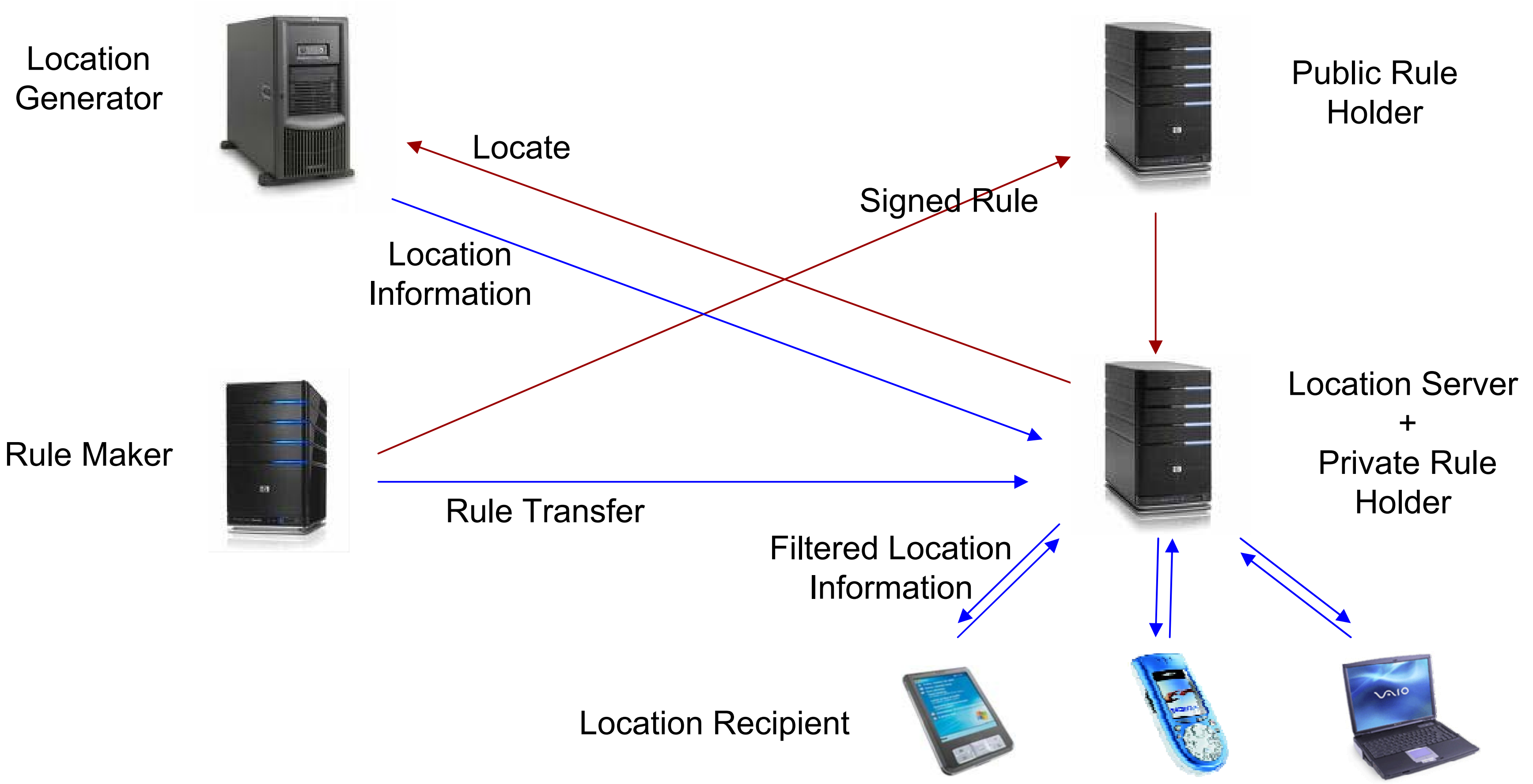
- Secure transmission of location objects
- User controlled privacy rules
- Filtering location information
- Location object carries core set of privacy rules
- Ability of user to hide real identity

Scenarios



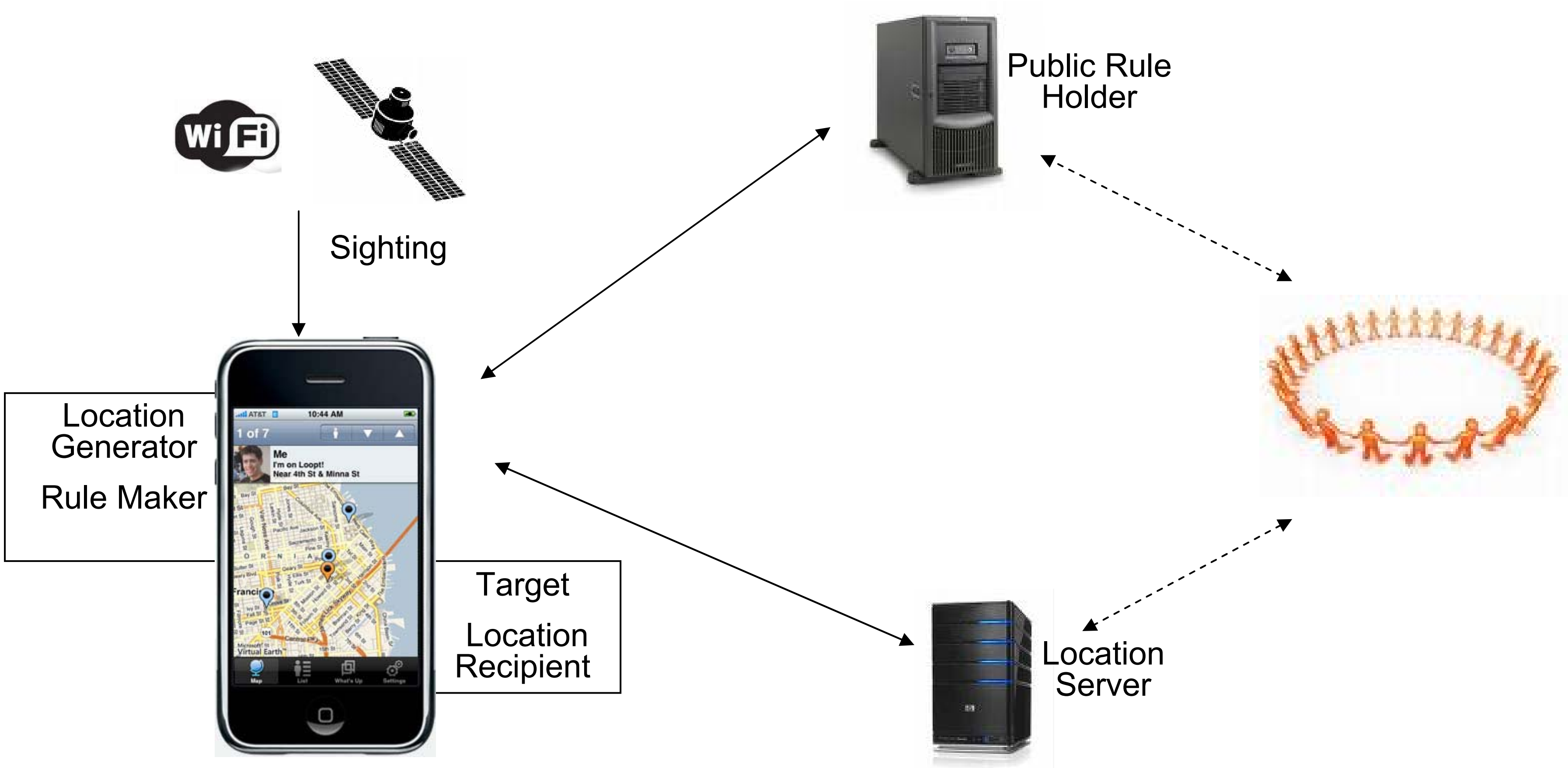
GPS Device with Internal Computing Power: Closed System

Scenarios



Mobile Communities and Location-Based Services

Applications: Social Networking



Location Configuration

Configuring the location of a device, using means such as:

- DHCP extensions
  - RFC3825 : Option 123, geo-coordinate based location
  - RFC4776 : Option 99, civic address
- Link Layer Discovery Protocol - Media Endpoint Discovery
  - LLDP - a vendor-neutral Layer 2 protocol that allows a network device to advertise its identity and capabilities on the local network.
    IEEE standard 802.1AB-2005 in May 2005.
    Supersedes proprietary protocols like Cisco Discovery Protocol,
  - auto-discovery of LAN information (system id, port id, VLAN id, DiffServ settings, …) ⇨ plug & play
  - cisco discovery protocol: switch broadcasts switch/port id
    - switch → floor, port → room ⇨ room level accuracy
- HTTP Enabled Location Delivery
  - device retrieves location from Location Information Server (LIS)
  - assumption: device & LIS present in same admin domain;
    find LIS by DHCP, IPv6 anycast, …

- Applications ⇨ emergency 911, VoiP, location based applications

Security Considerations

- Traffic Analysis
  - attacks on target and privacy violations
- Securing the Privacy Rules
  - rules accessible to LS
  - authenticated using signature
- Emergency Case
  - handling authentication failure
- Identities & Anonymity

Presence Information Data Format - PIDF

- XML based object format to communicate presence information
  ⇨ Instant Messaging (IM)

- PIDF extension to carry geographical information:

- Extended PIDF encapsulates
  - preexisting location information formats
  - security & policy control

- Protocols capable of carrying XML or MIME types suitable

- Security: MIME-level → S/MIME

PIDF Elements

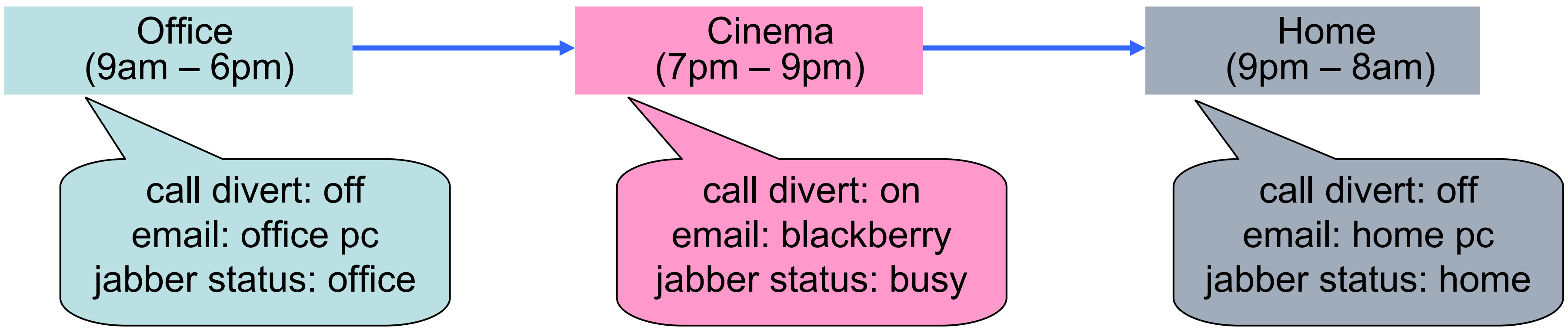
Baseline: RFC 3863

- entity
- contact (how to contact the person)
- timestamp
- status
- tuple (provide a way of segmenting presence information)

Extensions: RFC 4119

- location-info
- usage-rules
  - retransmission-allowed
  - retention-expires
  - ruleset-reference
  - note-well
- method
- provided-by

Location Type Registry



- Describes places of humans or end systems
- Application
  - define location-based actions
  - e.g. if loc = “classroom” then cell phone ringer = off
  - e.g. if loc = “cinema” then call divert = on
- Location coordinate knowledge ≠ context
- airport, arena, bank, bar, bus-station, club, hospital, library….

⇨ Prediction:
most communication will be presence-initiated or pre-scheduled

Maintaining network state

Design Principles

Goals:

- identify, study common architectural components, protocol mechanisms
- what approaches do we find in network architectures?
- *synthesis:* big picture

7 design principles:

- network virtualization: overlays
- separation of data, control
  ⇨signalling
- **hard state versus soft state**
- randomization
- indirection
- multiplexing
- design for scale

Maintaining network state

state: information *stored* in network nodes by network protocols

- updated when network “conditions” change
- stored in multiple nodes
- often associated with end-system generated call or session
- examples:
  - ATM switches maintain lists of VCs: bandwidth allocations, VCI/VPI input-output mappings
  - RSVP routers maintain lists of upstream sender IDs, downstream receiver reservations
  - TCP: Sequence numbers, timer values, RTT estimates

Hard-state

- state *installed* by receiver on receipt of *setup message* from sender
- state *removed* by receiver on receipt of *teardown message* from sender
- *default assumption:* state valid unless told otherwise
  - in practice: failsafe-mechanisms (to remove orphaned state) in case of sender failure e.g., receiver-to-sender “heartbeat”: is this state still valid?
- examples:
  - Q.2931 (ATM Signaling)
  - ST-II (Internet hard-state signaling protocol - outdated)
  - TCP

Soft-state

- state *installed* by receiver on receipt of setup (trigger) message from sender (typically, an endpoint)
  - sender also sends periodic *refresh* message: indicating receiver should continue to maintain state
- state *removed* by receiver via timeout, in absence of refresh message from sender
- default assumption: state becomes invalid unless refreshed
  - in practice: explicit state removal (*teardown*) messages also used
- examples:
  - RSVP, RTP/RTCP, IGMP

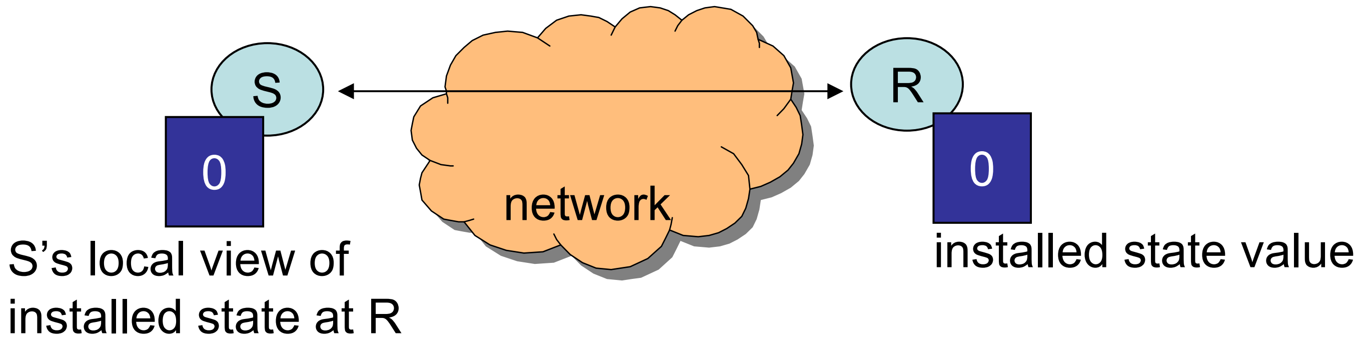
State: senders, receivers

- sender: network node that *(re)generates* signaling (control) messages to install, keep-alive, remove state from other nodes

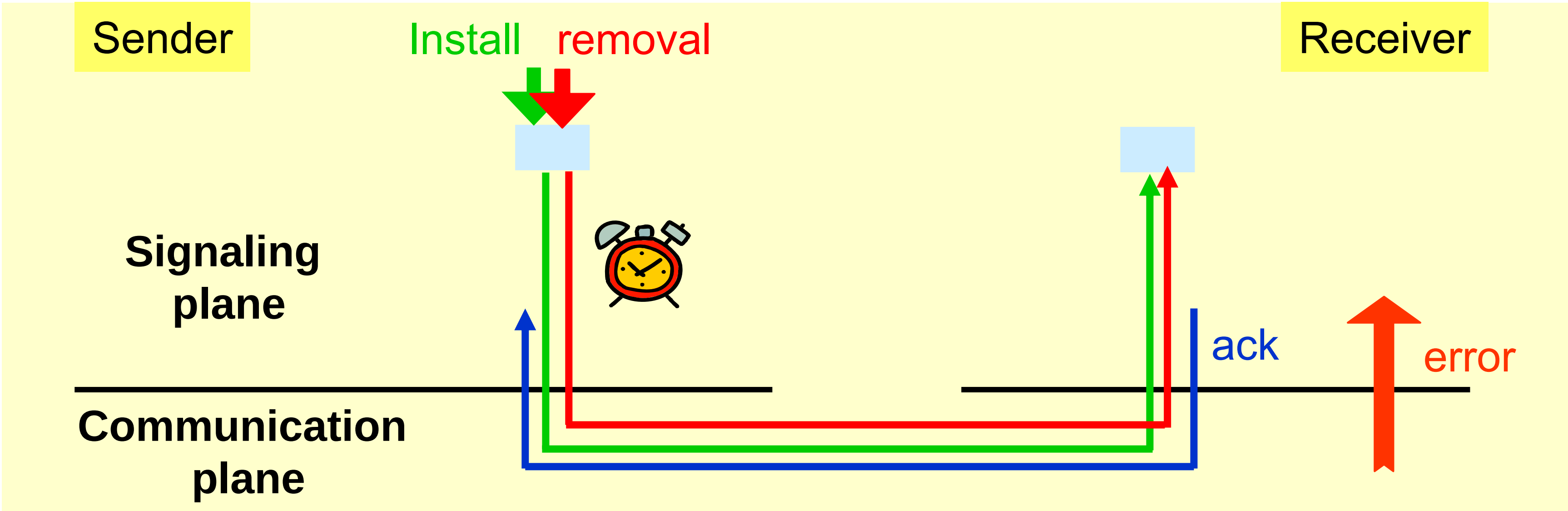
- receiver: node that creates, maintains, removes state based on signaling messages *received* from sender

Let's build a signaling protocol

- *S:* state *S*ender (state installer)
- *R:* state *R*eceiver (state holder)
- desired functionality:
  - S: set values in R to 1 when state "installed", set to 0 when state "not installed"
  - if other side is down, state is not installed (0)
  - initial condition: state not installed

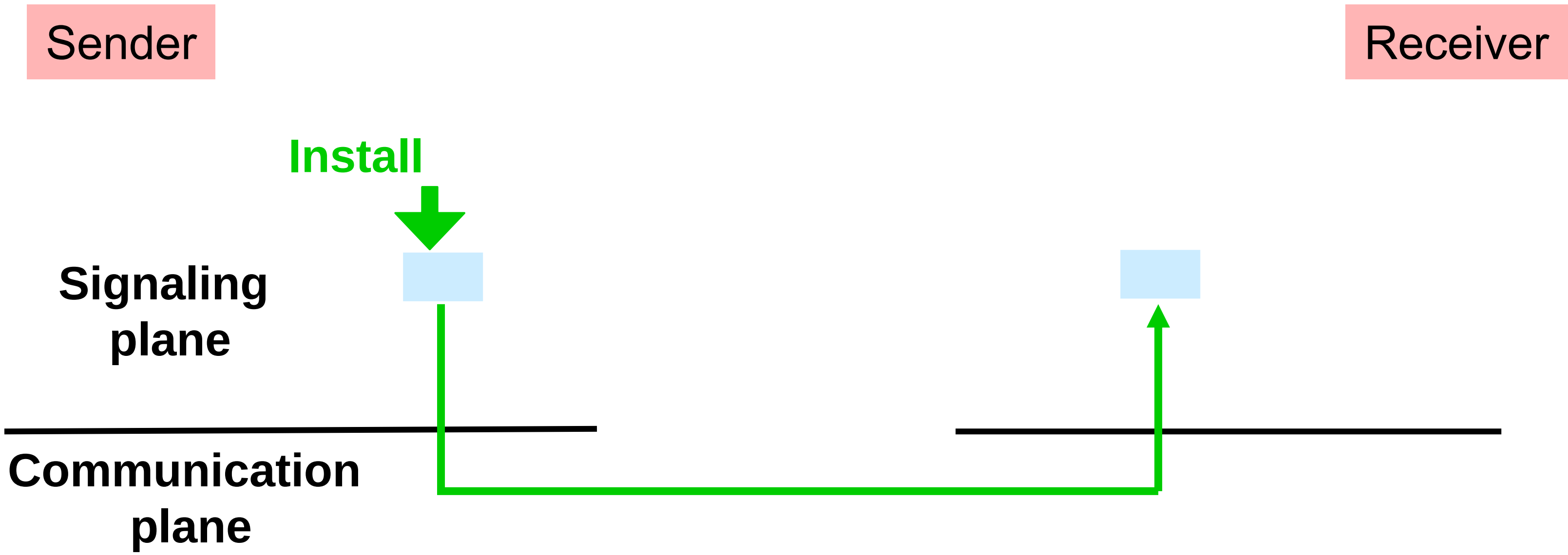


Hard-state signaling



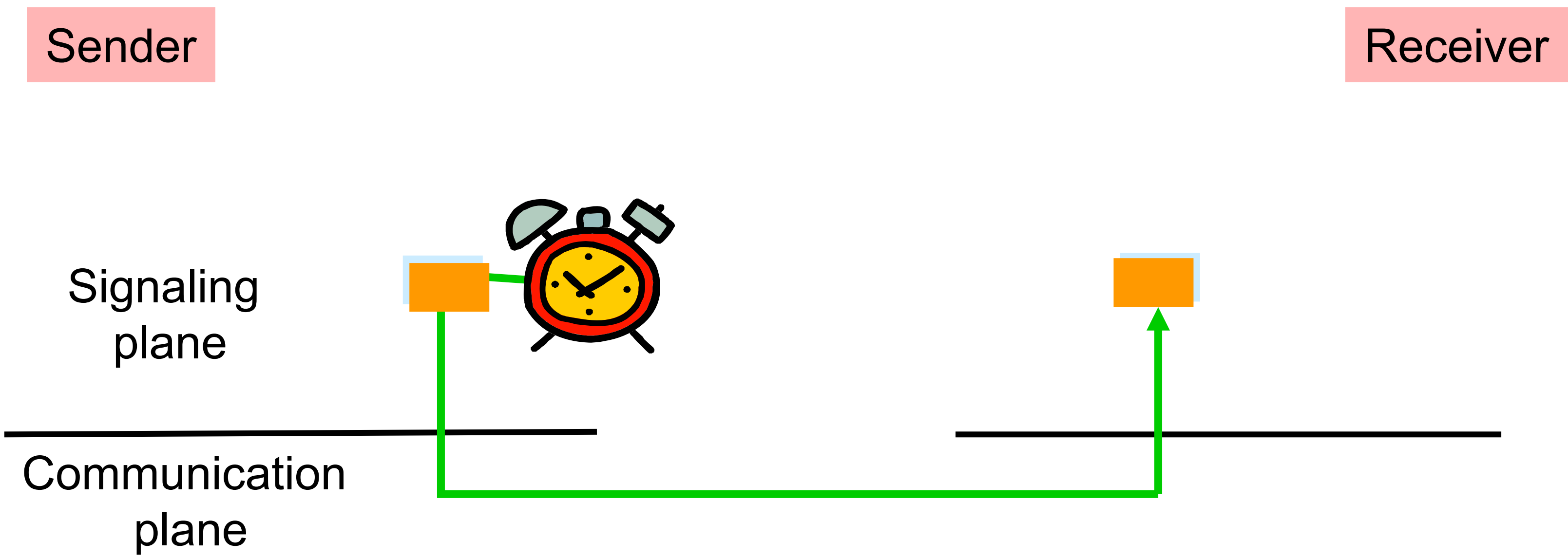
- reliable signaling
- state removal by request
- requires additional error handling
  - e.g., sender failure

Soft-state signaling



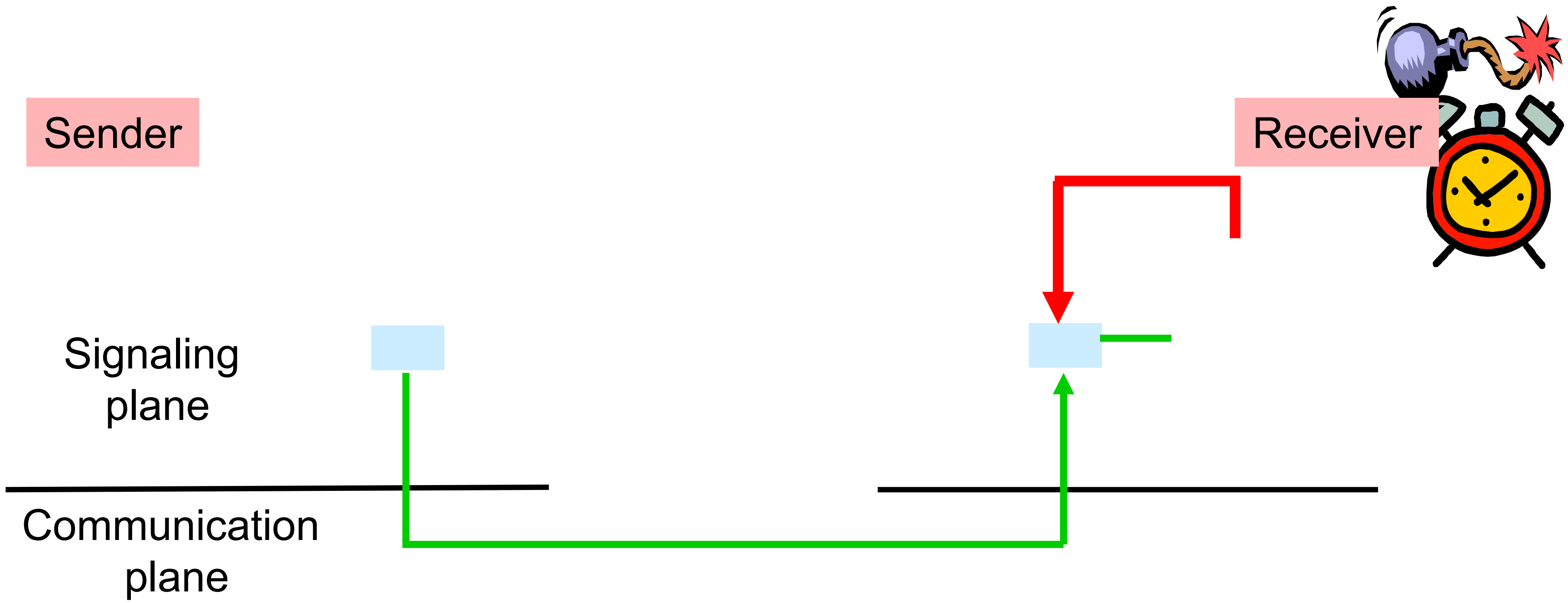
- best effort signaling

Soft-state signaling



- best effort signaling
- refresh timer, periodic refresh

Soft-state signaling



- best effort signaling
- refresh timer, periodic refresh
- state time-out timer, state removal only by time-out

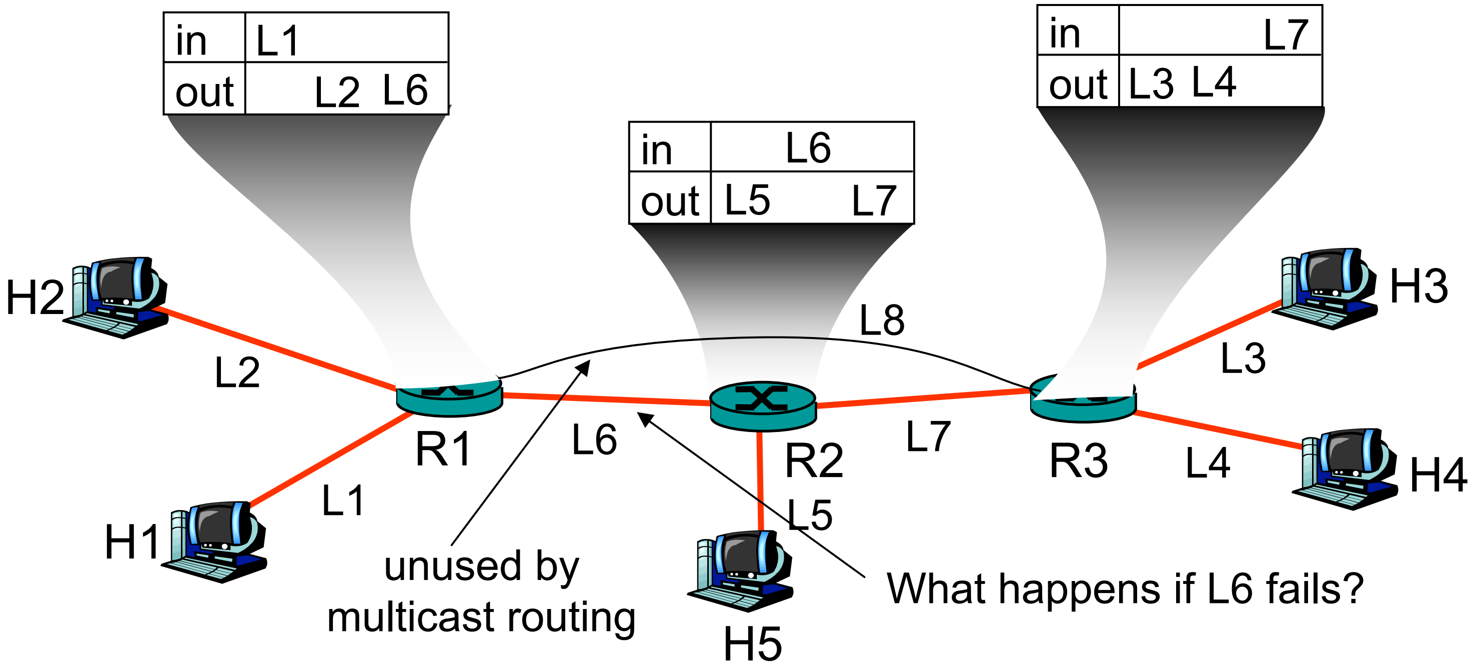
Soft-state: claims

- “Systems built on soft-state are robust” [Raman 99]
- “Soft-state protocols provide .. greater robustness to changes in the underlying network conditions…” [Sharma 97]
- “obviates the need for complex error handling software” [Balakrishnan 99]

What does this mean?

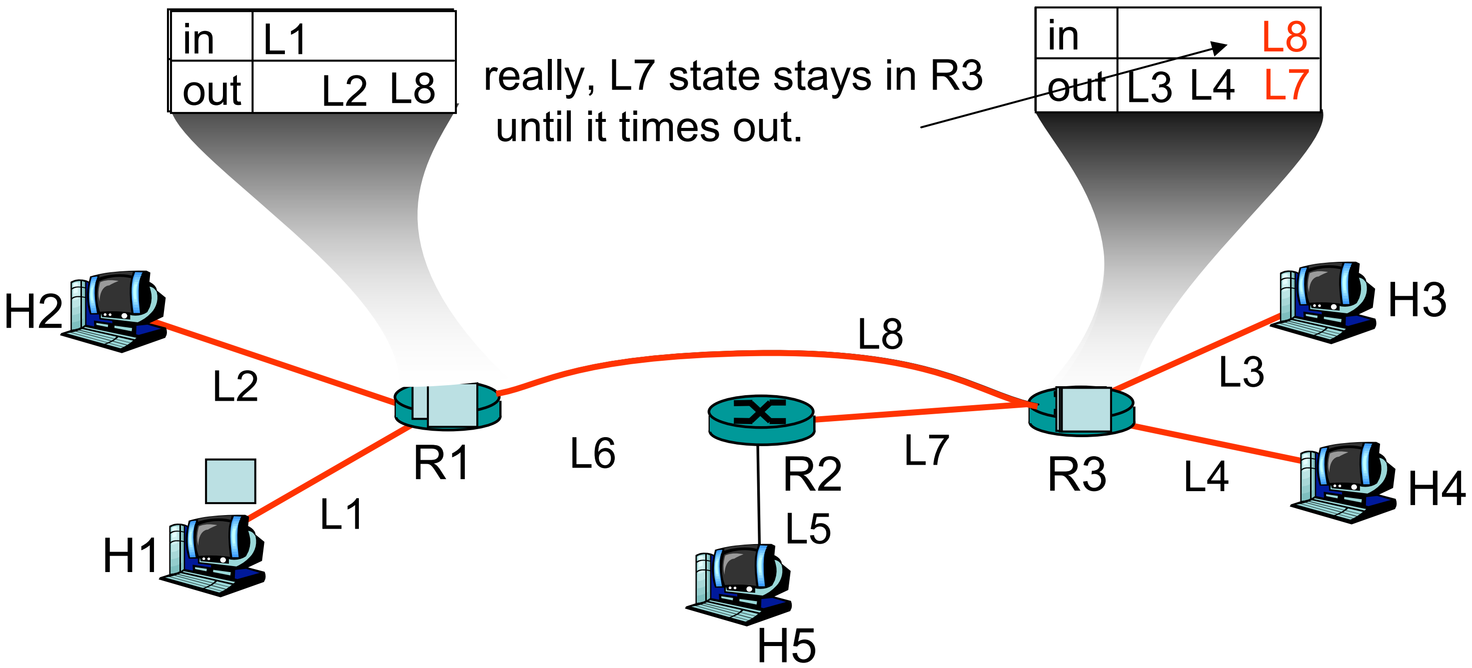
Soft-state: "easy" handling of changes

- Periodic refresh: if network "conditions" change, refresh will re-establish state under new conditions
- example: RSVP/routing interaction: if routes change (nodes fail) RSVP PATH refresh will *re-establish* state along new path



Soft-state: "easy" handling of changes

- L6 goes down, multicast routing reconfigures but…
- H1 data no longer reaches H3, H4, H5 (no sender or receiver state for L8)
- H1 refreshes PATH, establishes *new* state for L8 in R1, R3
- H4 refreshes RESV, propagates upstream to H1, establishes new receiver state for H4 in R1, R3



Soft-state: “easy” handling of changes

- “recovery” performed transparently to end-system by normal refresh procedures
- no need for network to signal failure/change to end system, or end system to respond to specific error
- less signaling (volume, types of messages) than hard-state from network to end-system but…
- more signaling (volume) than hard-state from end-system to network for refreshes

Soft-state: refreshes

- refresh messages serve many purposes:
  - trigger: first time state-installation
  - refresh: refresh state known to exist (“I am still here”)
  - <lack of refresh>: remove state (“I am gone”)
- challenge: all refresh messages unreliable
  - problem: what happens if first PATH message gets lost?
    - copy of PATH message only sent after refresh interval
  - would like triggers to result in state-installation a.s.a.p.
  - enhancement: add receiver-to-sender refresh_ACK for triggers
  - sender initiates retransmission if no refresh_ACK is received after short timeout
  - e.g., see paper “Staged Refresh Timers for RSVP” by Ping Pan and Henning Schulzrinne
  - approach also applicable to other soft-state protocols

Signaling Spectrum

periodic refresh

Soft-state (SS)

SS + explicit removal
IGMPv2/v3

SS + reliable trigger
RSVP new version

SS + reliable trigger/removal
ST-II

Hard-state

- best effort periodic state installation/refresh
- state removal by time out
- RSVP, IGMPv1

- reliable signaling
- explicit state removal
- requires additional mechanism to remove orphan state
- Q2931b

Master Course
Computer Networks
IN2097

Prof. Dr.-Ing. Georg Carle
Christian Grothoff, Ph.D.
Dr. Nils Kammenhuber

Chair for Network Architectures and Services
Institut für Informatik
Technische Universität München
http://www.net.in.tum.de

Architecture: the big picture

Architecture: the big picture

Goals:

- identify, study principles that can guide network architecture
- "bigger" issues than specific protocols or implementation wisdom,
- *synthesis:* the *really* big picture

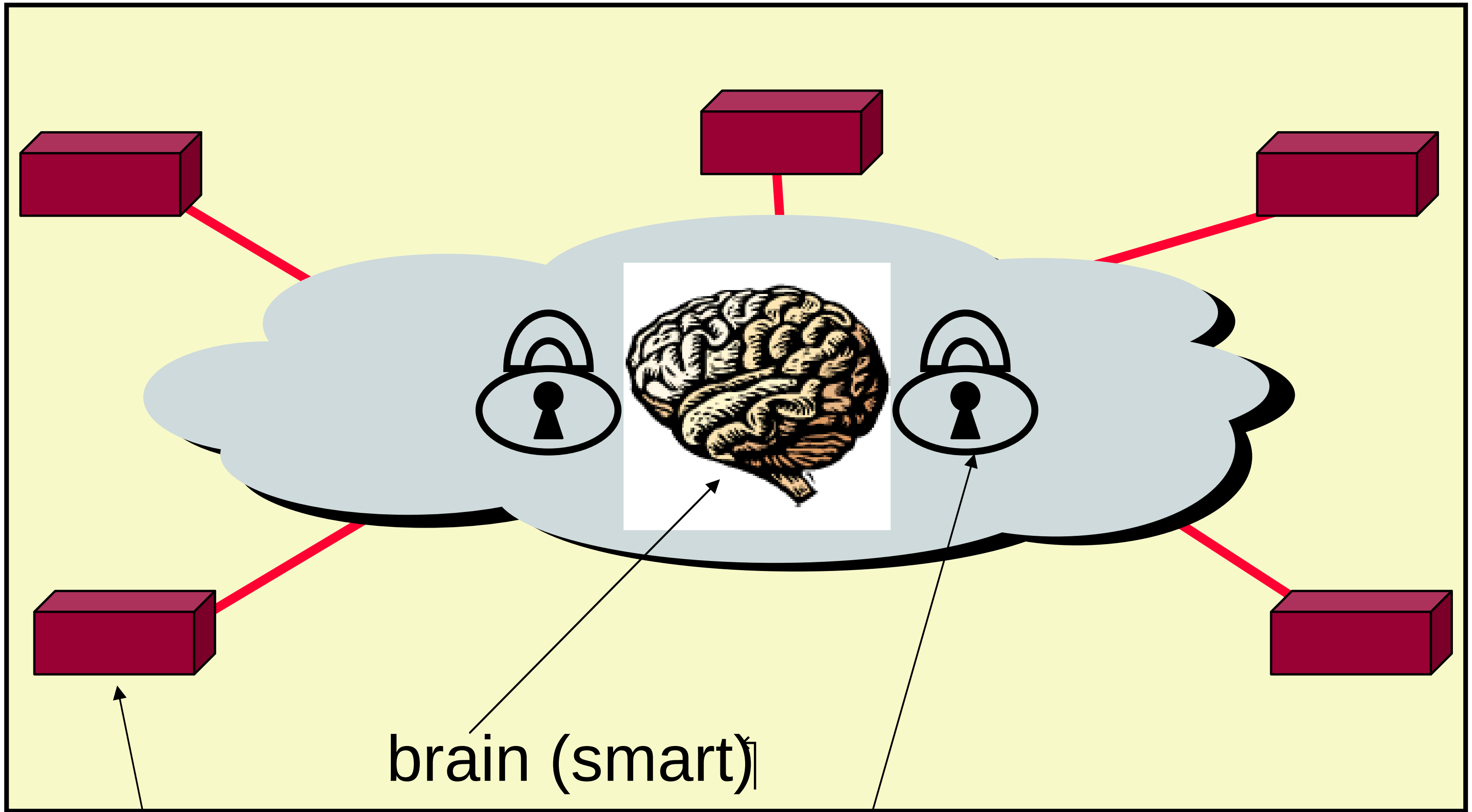
Overview:

- Internet design principles
- rethinking the Internet design principles
- packet switching versus circuit switching revisited

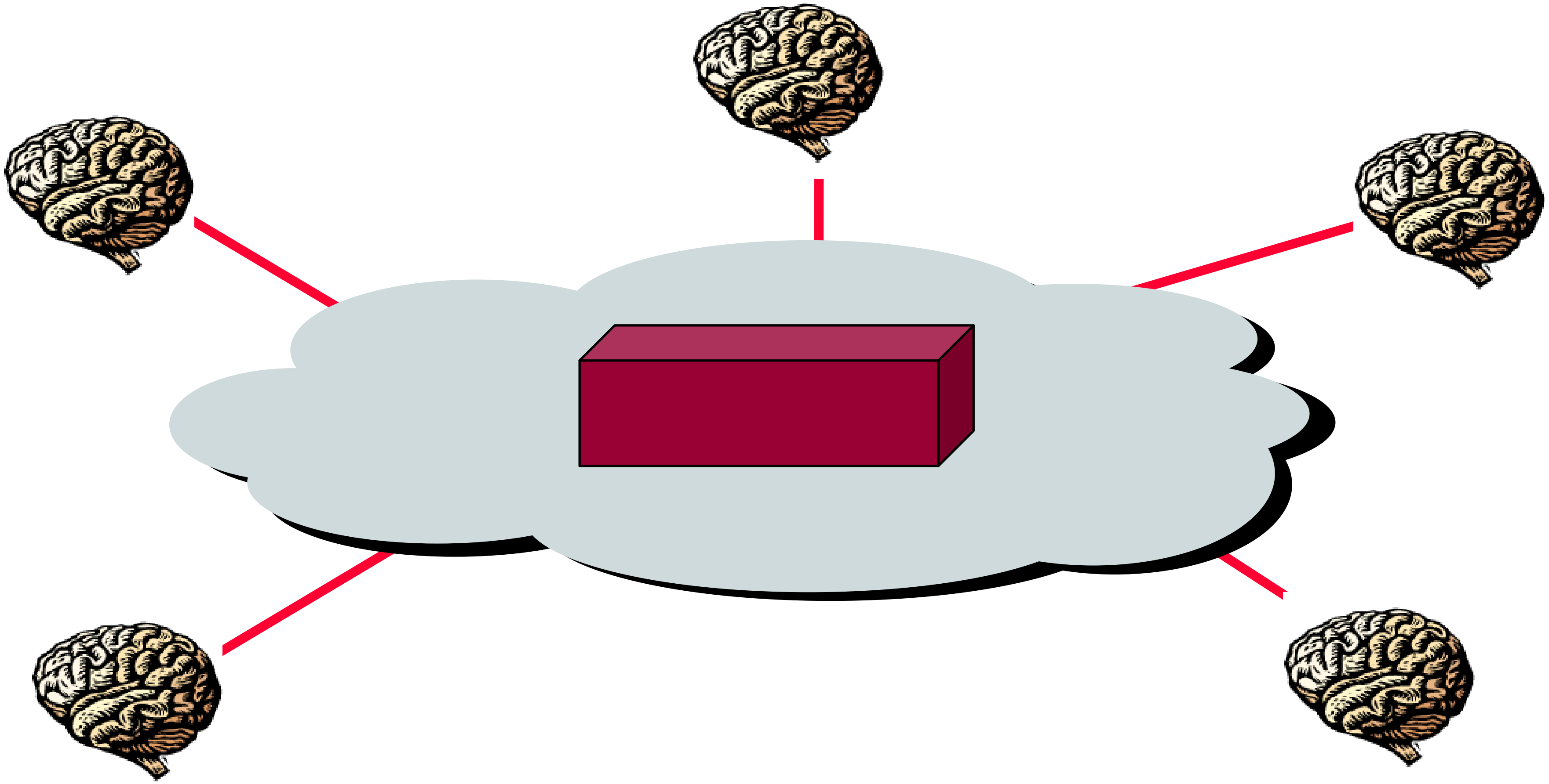
Key questions

- How to decompose the complex system functionality into protocol layers?
- Which functions placed *where* in network, at which layers?
- Can a function be placed at multiple levels?

- Answer these questions in context of
  - Internet
  - Telephone network
    (Nickname 1: Telco — telecommunications provider)
    (Nickname 2: POTS — “plain old telephone system”)


brain (smart)
brick (dumb)
lock (you can’t get in)

Common View of the IP Network: Dumb network, smart end hosts



The Internet End-to-End principle

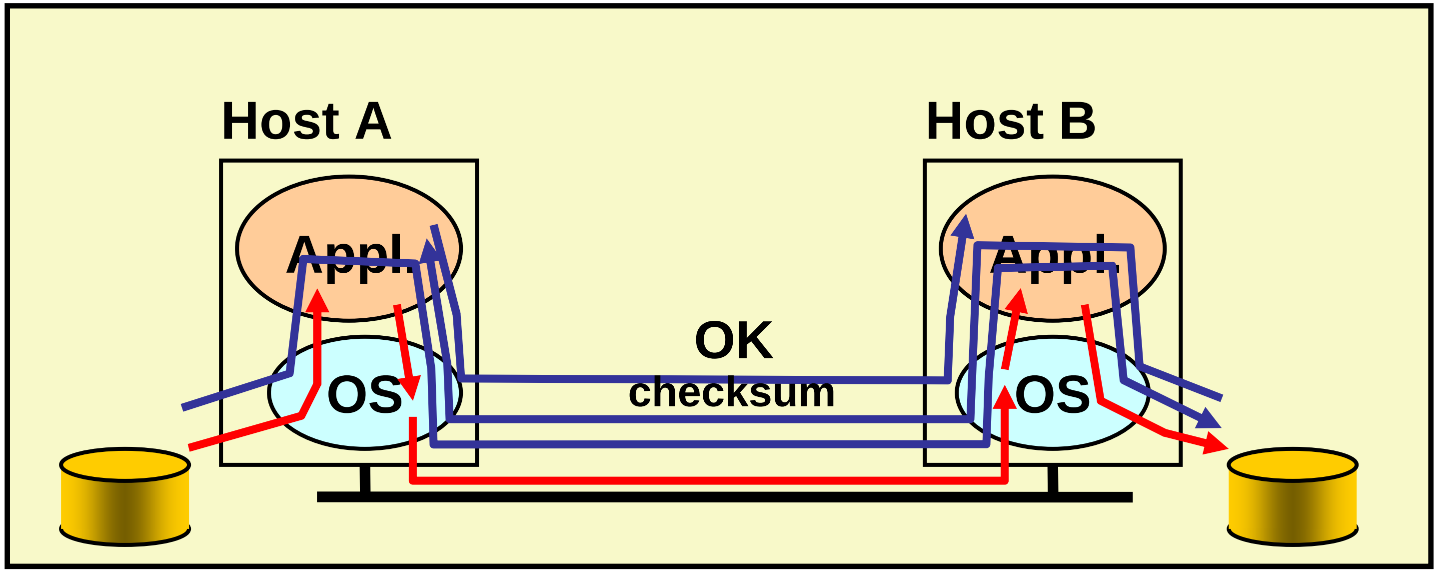
Internet End-to-End Principle

- "…functions placed at the lower levels may be *redundant* or of *little value* when compared to the cost of providing them at the higher level…"

- "…sometimes an *incomplete* version of the function provided by the communication system (lower levels) may be useful as a *performance enhancement*…"

- This leads to a philosophy diametrically opposite to the telephone world of dumb end-systems (the telephone) and intelligent networks.

Example: Reliable File Transfer



- Solution 1: make each step reliable, and then concatenate them

- Solution 2: each step unreliable: end-to-end check and retry (…the Internet way)

Discussion

- Is solution 1 good enough?
  - No — what happens if components on path fail or misbehave (bugs)?
- Is reliable communication sufficient:
  - No — what happens if disk errors?
- So need application to make final correctness check anyway!
- Thus, full functionality can be entirely implemented at application layer; *no* need for reliability from lower layers

Discussion

Q: Is there any reason to implement reliability at lower layers?

A: YES: "easier" (and more efficient) to check and recovery from errors at each intermediate hop

- e.g.: faster response to errors, localized retransmissions
- Concrete example: Error correction on wireless links (in spite of TCP packet loss detection)

Trade-offs

- application has more information about the data and semantics of required service (e.g., can check only at the end of each data unit)
- lower layer has more information about constraints in data transmission (e.g., packet size, error rate)
- *Note:* these trade-offs are a direct result of layering!

Internet & End-to-End Argument

- Network layer provides one simple service: best effort datagram (packet) delivery
- Transport layer at network edge (TCP) provides end-end error control
  - Performance enhancement used by many applications (which could provide their own error control)
- All other functionality …
  - *All* application layer functionality
  - Network services: DNS
  - ⇨ Implemented at application level

Internet & End-to-End Argument

- Discussion: congestion control, flow control: why at transport, rather than link or application layers?
- congestion control needed for many applications (assumes reliable application-to-TCP data passing)
- many applications “don’t care” about congestion control – it’s the network’s concern
- consistency across applications — you ***have*** to use it if you use TCP (social contract — everybody does)
- why do it at the application level
  - Flow control — application knows how/when it wants to consume data
  - Congestion control — application can do TCP-friedly congestion control

Internet & End-to-End Argument

- Discussion: congestion control, flow control: Why not at the link layer?
  1. Not every application needs it/wants it
  2. Lots of state at each router (each connection needs to buffer, need back pressure) — it's hard
  3. Congestion control in the entire network, e.g., load-adaptive dynamic IP routing? — multiple reasons against it:
     - hard to do
     - prone to oscillations
     - didn't work out in ARPANET → "never again" attitude

E2E Argument: Interpretations

- One interpretation:
  - A function can only be completely and correctly implemented with the knowledge and help of the applications *standing at the communication endpoints*
- Another: (more precise…)
  - A system (or subsystem level) should consider only functions that can be *completely and correctly* implemented within it.
- Alternative interpretation: (also correct …)
  - Think twice before implementing a functionality that you believe that is useful to an application at a lower layer
  - If the application can implement a functionality correctly, implement it a lower layer *only* as a performance enhancement

End-to-End Argument: Critical Issues

- End-to-end principle emphasizes:
  - *function placement*
  - *correctness, completeness*
  - *overall system costs*
- Philosophy: if application can do it, don't do it at a lower layer — application best knows what it needs
  - add functionality in lower layers iff (1) used by and improves performances of many applications, (2) does not hurt other applications
- allows *cost-performance* tradeoff

End-to-End Argument: Discussion

- End-end argument emphasizes correctness & completeness, but does not emphasize…:
  - *complexity:* Does complexity at edges result in a "simpler" architecture?
  - *evolvability:* Ease of introduction of new functionality; ability to evolve because easier/cheaper to add new edge applications than to change routers?
  - *technology penetration:* Simple network layer makes it "easier" for IP to spread everywhere

Internet Design Philosophy (Clark' 88)

In order of importance:

Different ordering of priorities would make a different architecture!

0. Connect existing networks
   - initially ARPANET, ARPA packet radio, packet satellite network
1. Survivability
   - ensure communication service even with network and router failures
2. Support multiple types of services
3. Must accommodate a variety of networks
4. Allow distributed management
5. Allow host attachment with a low level of effort
6. Be cost effective
7. **Allow resource accountability**

1. Survivability

- Continue to operate even in the presence of network failures (e.g., link and router failures)
  - as long as network is not partitioned, two endpoints should be able to communicate
  - any other failure (excepting network partition) should be transparent to endpoints
- Decision: maintain end-to-end transport state only at end-points
  - eliminate the problem of handling state inconsistency and performing state restoration when router fails
- Internet: stateless network-layer architecture
  - No notion of a session/call at network layer
  - Example: Your TCP connection shouldn't break when a router along the path fails
- Assessment: ??

2. Types of Services

- Add UDP to TCP to better support other apps
  - e.g., “real-time” applications
- arguably main reason for separating TCP, IP
- datagram abstraction: lower common denominator on which other services can be built
  - service differentiation was considered (remember ToS field in IP header?), but this has never happened on the large scale (Why?)
- Assessment: ?

3. Variety of Networks

- Very successful (why?)
  - because the minimalist service; it requires from underlying network only to deliver a packet with a “reasonable” probability of success
- …does not require:
  - reliability
  - in-order delivery
- The mantra: IP over everything
  - Then: ARPANET, X.25, DARPA satellite network..
  - Subsequently: ATM, SONET, WDM…
- Assessment: ?

Other Goals

- Allow distributed management
  - Administrative autonomy: IP interconnects networks
    - each network can be managed by a different organization
    - different organizations need to interact only at the boundaries
    - … but this model complicates routing
  - Assessment: ?

- Cost effective
  - sources of inefficiency
    - header overhead
    - retransmissions
    - routing
  - …but “optimal” performance never been top priority
  - Assessment: ?

Other Goals (Cont)

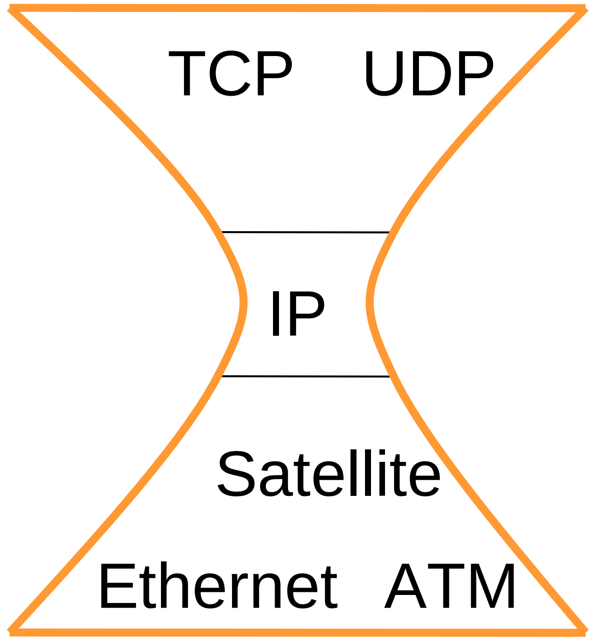
- Low cost of attaching a new host
  - not a strong point → higher than other architecture because the intelligence is in hosts (e.g., telephone vs. computer)
  - bad implementations or malicious users can produce considerably harm (remember fate-sharing?)
  - Assessment: ?
- Accountability
  - Assessment: ?

What About the Future?

- Datagram not the best abstraction for:
  - resource management, accountability, QoS
- new abstraction: flow (see IPv6)
  - Typically: (src, dst, #bytes) tuple
  - But: “flow” not precisely defined
    - when does it end? Explicit connection teardown? Timeout?
    - *src* and *dst* =...? ASes? Prefixes? Hosts? Hosts&Protocol?
  - IPv6: difficulties to make use of flow IDs
- routers require to maintain per-flow state
- state management: recovering lost state is hard
- in context of Internet (1988) we see the first proposal of “soft state”!
  - soft-state: end-hosts responsible to maintain the state

Summary: Internet Architecture

- packet-switched datagram network
- IP is the glue (network layer overlay)
- IP hourglass architecture
  - all hosts and routers run IP
- stateless architecture
  - no per flow state inside network



IP hourglass

Summary: Minimalist Approach

- Dumb network
  - IP provide minimal functionalities to support connectivity
  - addressing, forwarding, routing
- Smart end systems
  - transport layer or application performs more sophisticated functionalities
  - flow control, error control, congestion control
- Advantages
  - accommodate heterogeneous technologies (Ethernet, modem, satellite, wireless, ...)
  - support diverse applications (telnet, ftp, Web, X windows)
  - decentralized network administration

But that was yesterday

……. what about tomorrow?

Rethinking Internet Design

What's changed?

- operation in untrustworthy world
  - endpoints can be malicious: Spam, Worms, (D)DoS, ...
  - If endpoint not trustworthy, but want trustworthy network
    ⇨ more mechanisms in network core

- more demanding applications
  - end-to-end best effort service not enough
  - new service models in network (IntServ, DiffServ)?
  - new application-level service architecture built on top of network core (e.g., CDN, P2P)?

Rethinking Internet Design

What's changed (cont.)?

- ISP service differentiation
  - ISP doing more (than other ISPs) in core is competitive advantage
- Rise of third party involvement
  - interposed between endpoints (even against will)
  - e.g., Chinese government, recording industry, Vorratsdatenspeicherung
- less sophisticated users

All five changes motivate shift away from end-to-end!

What's at stake?

"At issue is the conventional understanding of the "Internet philosophy"

- freedom of action
- user empowerment
- end-user responsibility for actions taken
- lack of control *"in"* the net that limit or regulate what users can do

The end-end argument fostered that philosophy because they enable the freedom to innovate, install new software at will, and run applications of the users choice."

[Blumenthal and Clark, 2001]

Technical response to changes

- Trust: emerging distinction between what is “in” network (*us*, trusted) and what is not (*them*, untrusted).
  - ingress filtering
  - emergence of Internet UNI (user network interface, as in ATM)?
- Modify endpoints
  - harden endpoints against attack
  - endpoints/routers do content filtering: Net-nanny
  - CDN, ASPs: rise of structured, distributed applications in response to inability to send content (e.g., multimedia, high bw) at high quality

Technical response to changes

- Add functions to the network core:
  - filtering firewalls
  - application-level firewalls
  - NAT boxes
  - active networking

… All operate within network, making use of application-level information

- which addresses can do what at application level?
- If addresses have meaning to applications, NAT must “understand” that meaning

Epilogue: will IP take over the world?

- Reasons for success of IP:
  - *reachability:* reach every host; adapts topology when links fail.
  - *heterogeneity:* single service abstraction (best effort) regardless of physical link topology

- many other claimed (or commonly accepted) reasons for IP's success may not be true

  …. let's take a closer look

1. IP already dominates global communications?

- business revenues (in US$, 2007):
  - ISPs: 13B
  - Broadcast TV: 29B
  - Cable TV: 29.8B
  - Radio broadcast: 10.6B
  - Phone industry: 268B

- Router/telco switch markets:
  - Core router: 1.7B; edge routers: 2.4B
  - SONET/SDH/WDM: 28B, Telecom MSS: 4.5B

Q: IP equipment cheaper? Economies of scale? (lots of routers?)

Q: per-device, IP is cheaper (one line into house, multiple devices)

Q: # bits carried in each network?

Q: Internet, more traffic and congestion is spread among all users (bad?)

2. IP is more efficient?

- Statistical multiplexing versus circuit switching
- Link utilization:
  - Avg. link utilization in Internet core: 3% to 30% (ISPs: never run above 50%!)
  - Avg. utilization of Ethernet is currently 1%
  - Avg. link utilization of long distance phone lines: 33%
- low IP link utilization: purposeful!
  - predictability, stability, low delay, resilience to failure
  - at higher utilization: traffic spikes induce short congestion periods → deterioration of QoS

- At low utilization, we loose benefits of statistical multiplexing!

3. IP is more robust?

- "Internet was built to sustain a nuclear war" — marketing vapor!
  - Remember large-scale network outages, e.g. on Sep 11th 2001?

- Median IP network availability: downtime: 471 min/yr
- Avg. phone network downtime: 5 min/yr

- Convergence time with link failures:
  - BGP: ≈ 3–15 min,
    intra-domain: ≈ 0.1–1 s (e.g., OSPF)
  - SONET: 50 ms

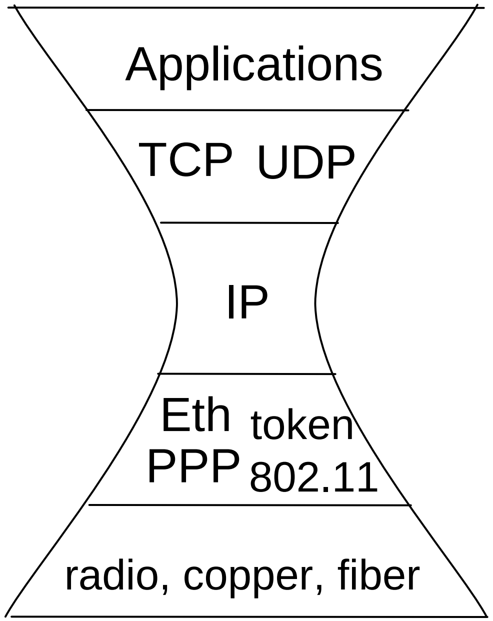
- Inconsistent routing state
  - human misconfigurations
  - in-band signaling (signaling and data share same network)
  - routing computation "complex"

4. IP is simpler?

- Intelligence at edge, simplicity in core
  - Cisco IOS: 8M lines of code
  - Telephone switch: 3M lines of code

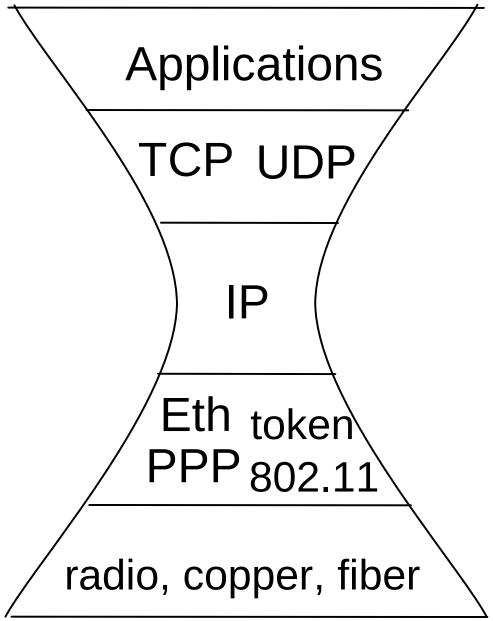
- Linecard complexity:
  - Router: 30M gates in ASICs, 1 CPU, 300M packet buffers
  - Switch: 25% of gates, no CPU, no packet buffers

Big picture: Original idea, the IP hour glass figure

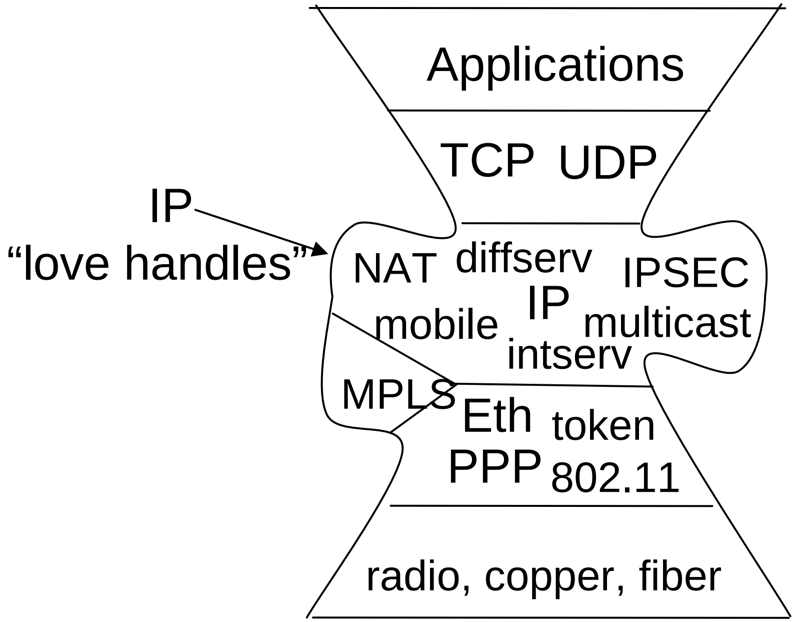


IP “hourglass”

Big picture: supporting new applications – losing the IP hour glass figure? (1)

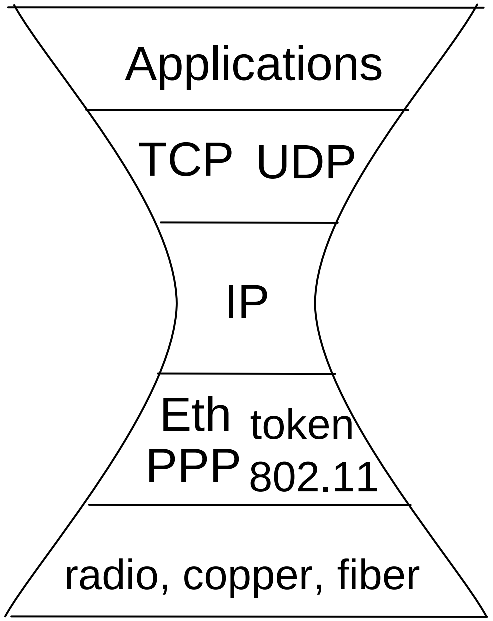


IP “hourglass”

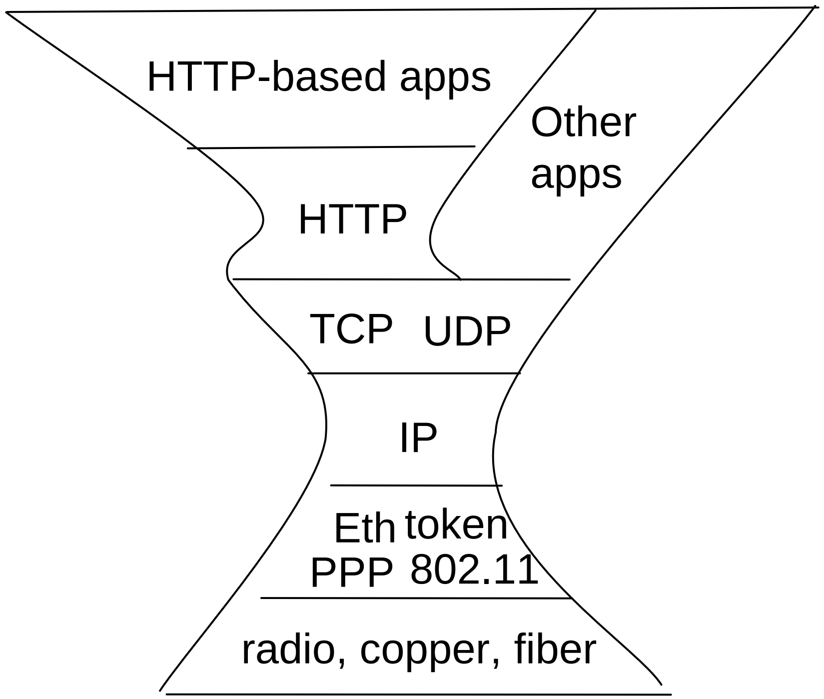


Middle-age IP = “hourglass” ?

Big picture: supporting new applications – losing the IP hour glass figure? (2)

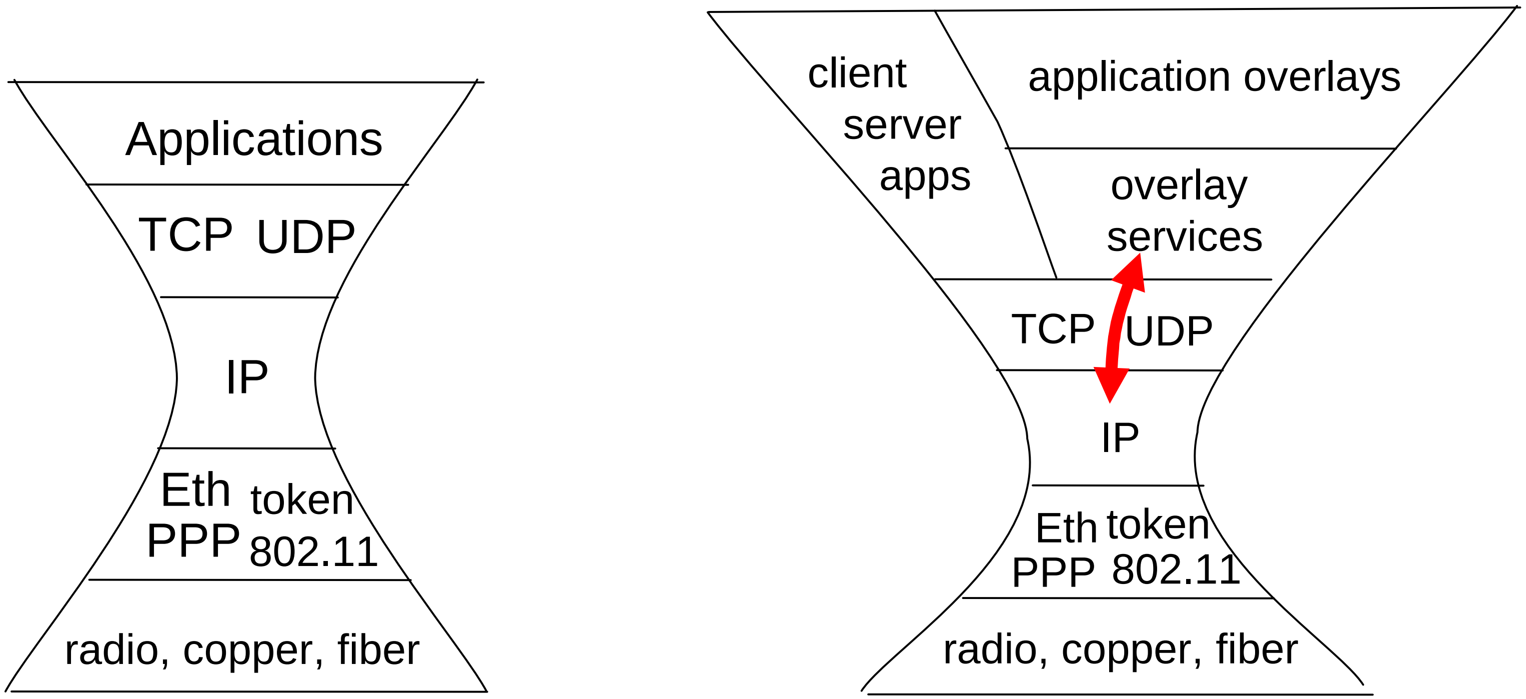


Original idea:
IP is greatest common denominator



Today:
HTTP is greatest common denominator

Big picture: supporting new applications – losing the IP hour glass figure? (3)



IP “hourglass”

Some advice on protocol design

- A loose collection of important thoughts related to protocol design
- ... actually, not only protocol design, but also
  - Programming in general
  - Systems in general (e.g., workflows in companies)
  - Life :)

Thought-triggering questions (1)

What problem am I trying to solve?

- Have at least one ***well-defined*** problem in mind
- Solve other problems without complicating the solution?

Will my solution scale?

- Think about what happens if you're successful: your protocol will be used by millions!
- Does the protocol make sense in small situations as well?

Thought-triggering questions (2)

How "robust" is my solution?

- adapt to failure/change
  - self-stabilization: eventually adapt to failure/change
  - Byzantine robustness: will work in spite of malicious users
- What are the underlying assumptions?
  - What if they are not true? catastrophe?
- maybe better to crash than degrade when problems occur: signal problem exists
- techniques for limited spread of failures
- protocol should degrade gracefully in overload, at least detect overload and complain

Further thoughts

Forward compatibility

- think about future changes, evolution
- make fields large enough
- reserve some spare bits
- specify an options field that can be used/augmented later

Parameters...

- Protocol parameters can be useful
  - designers can't determine reasonable values
  - tradeoffs exist: leave parameter choice to users
- Parameters can be bad
  - users (often not well informed) will need to choose values
  - try to make values plug-and-play

Simplicity vs Flexibility versus optimality

- Is a more complex protocol reasonable?
- Is “optimal” important?
- KISS: “The simpler the protocol, the more likely it is to be successfully implemented and deployed.”
- 80:20 rule:
  80% of gains achievable with 20% of effort

Why are protocols overly complex?

- design by committee
- backward compatibility
- flexibility: heavyweight swiss army knife
- unreasonble stiving for optimality
- underspecification
- exotic/unneeded features

Trading accuracy for time

- If computing the exact result is too slow, maybe an approximate solution will do
  - optimal solutions may be hard: heuristics will do (e.g., optimal multicast routing is a Steiner tree problem)
  - faster compression using “lossy” compression
    - lossy compression: decompression at receiver will not exactly recreate original signal

- Real-world examples?
  - games like chess: can’t compute an exact solution

Don’t confuse specification with implementation

- A general problem of computer scientists!
- Specifications indicate external effects/interaction of protocol.
- How protocol is implemented is up to designer
- Programming language specifications: in addition to specifying *what*, tend to suggest *how*.

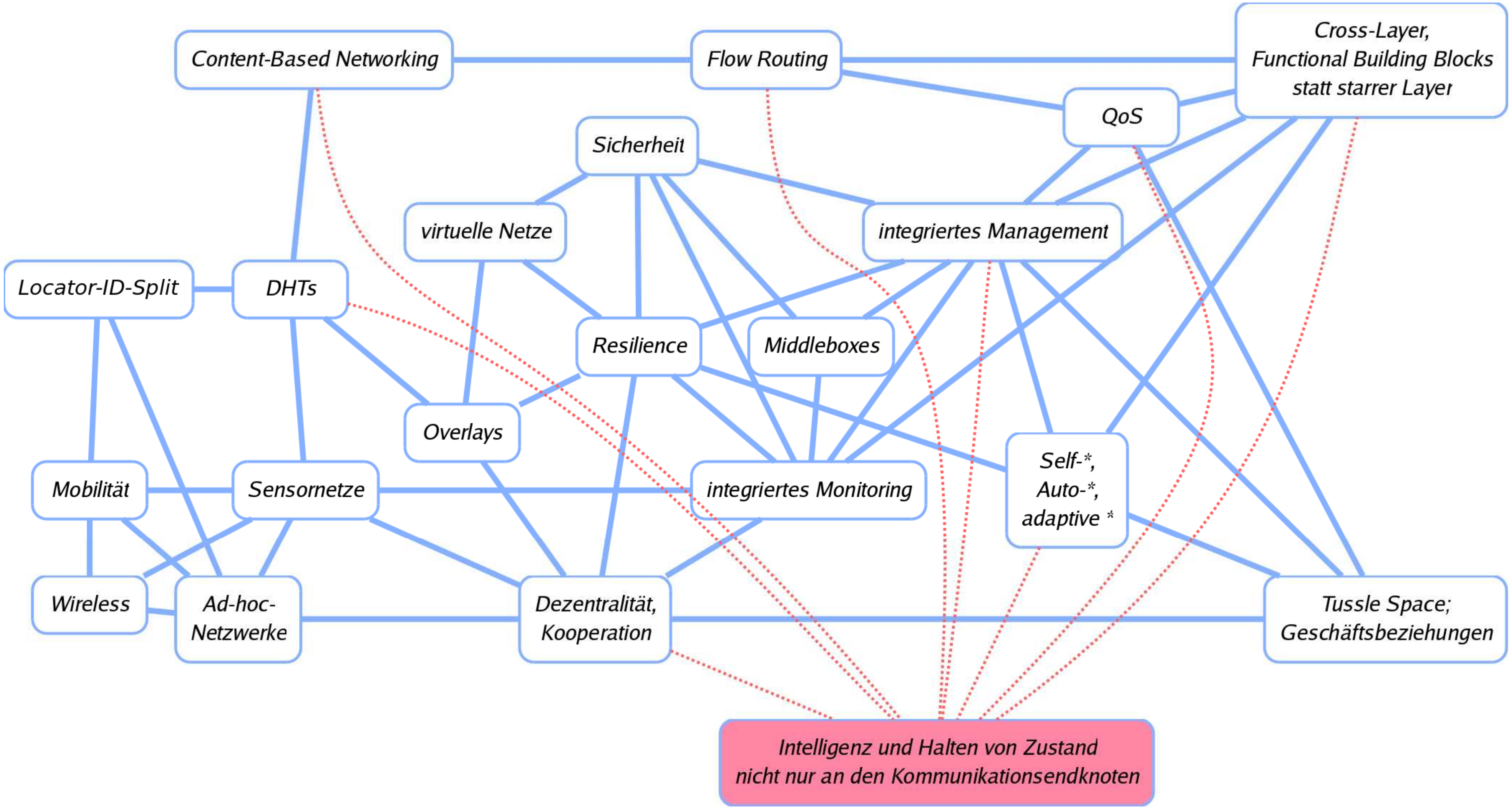
- real-world example: recipe
    1. Cut onions
    2. Cut potatoes
    3. Put onion and potatoes into pot and boil

  steps 1 and 2 can obviously be interchanged……

Where are we headed: Current/upcoming research topics

- Network management: Measurement, automation (“managemt. plane”)
- Service management:
  - Application-level networks, overlays, distributed hash tables (DHT)
  - QoS: Not a solved problem end-end
- Wireless networking, mobility
- New types of networks:
  - Sensor nets, body nets, home nets
- Security:
  - Lack of cryptographic signatures in many protocols
  - Most traffic unencrypted (…which is good for measurement…)
- Resilience: more robust networks (reacting faster / to more failures)
- “Future Internet”
  - Evolutionary approach: step-by-step introduction of new protocols
  - Revolutionary / clean-slate approach: Radical architecture change
- Ease of use, deployment (but what are the research problems here?)

(sorry for the German labels, but most notions are in English anyway…)



The *really* big picture

- Importance of user requirements

“It’s the end-user, stupid”

“It’s the application, stupid”

“It’s the network, stupid”

of course, not everyone agrees ….

Verizon product, purchased 2007

The end!