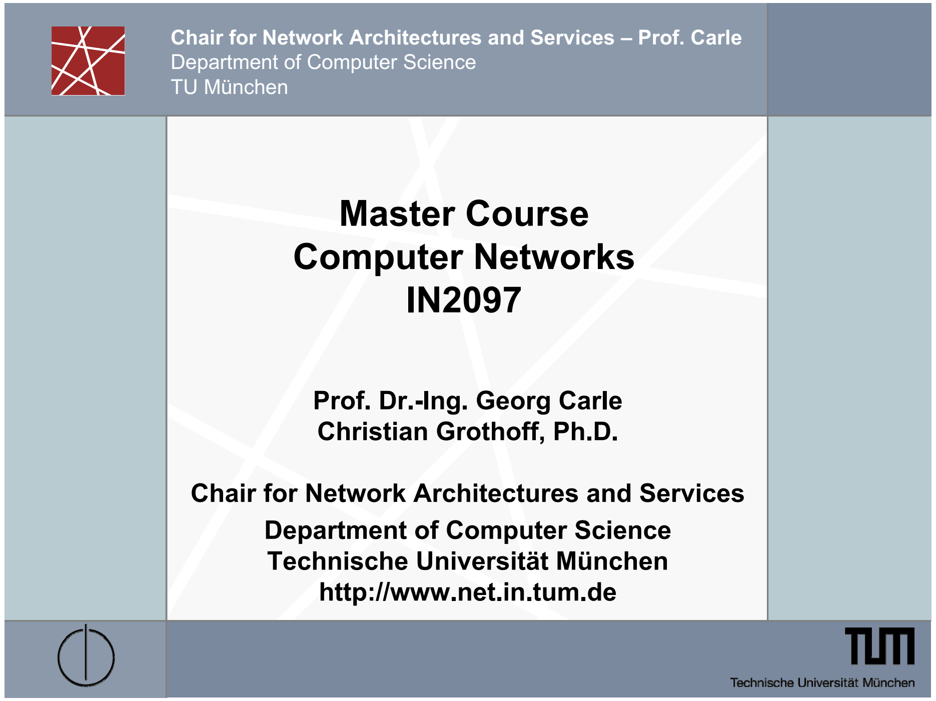


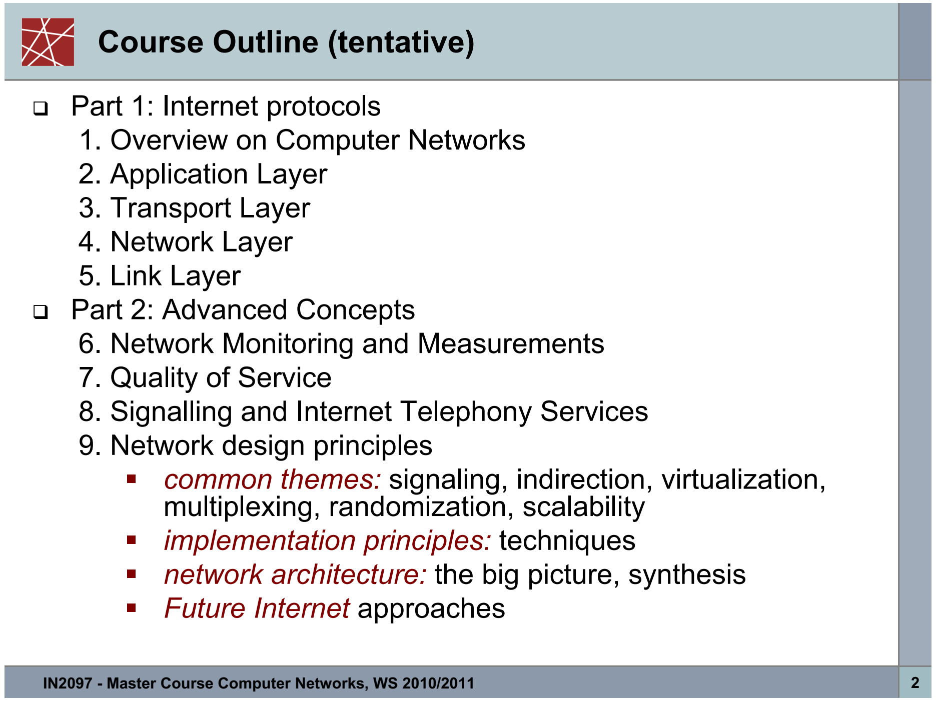
Chair for Network Architectures and Services – Prof. Carle
Department of Computer Science
TU München

Master Course Computer Networks IN2097

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

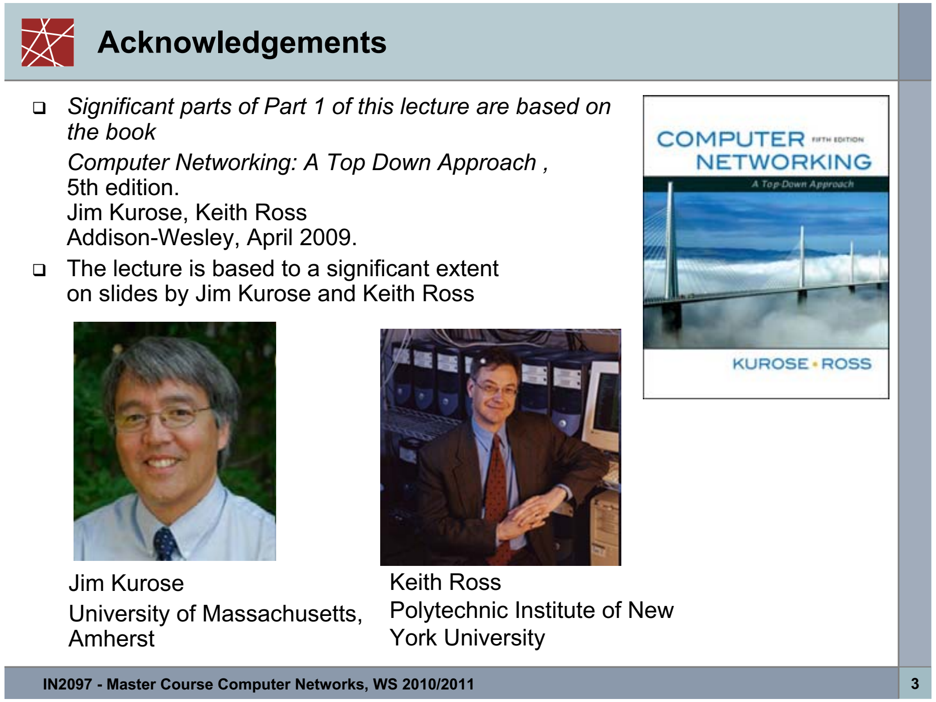
Chair for Network Architectures and Services
Department of Computer Science
Technische Universität München
http://www.net.in.tum.de

Technische Universität München



Course Outline (tentative)

- Part 1: Internet protocols
  1. Overview on Computer Networks
  2. Application Layer
  3. Transport Layer
  4. Network Layer
  5. Link Layer
- Part 2: Advanced Concepts
  6. Network Monitoring and Measurements
  7. Quality of Service
  8. Signalling and Internet Telephony Services
  9. Network design principles
     - *common themes:* signaling, indirection, virtualization, multiplexing, randomization, scalability
     - *implementation principles:* techniques
     - *network architecture:* the big picture, synthesis
     - *Future Internet* approaches

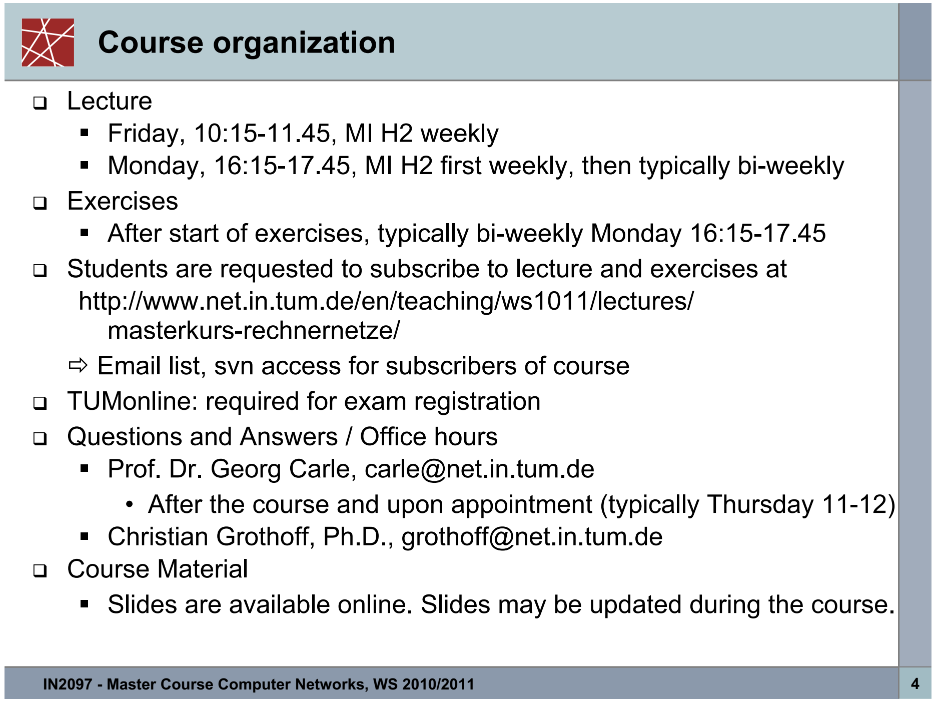


Acknowledgements

- *Significant parts of Part 1 of this lecture are 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
  - Friday, 10:15-11.45, MI H2 weekly
  - Monday, 16:15-17.45, MI H2 first weekly, then typically bi-weekly
- Exercises
  - After start of exercises, typically bi-weekly Monday 16:15-17.45
- Students are requested to subscribe to lecture and exercises at
  http://www.net.in.tum.de/en/teaching/ws1011/lectures/masterkurs-rechnernetze/
  ⇨ Email list, svn access for subscribers of course
- TUMonline: required for exam registration
- 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

- Exercises
  - prepare for the examination (but do not give a bonus)
- Practical assignments
  - Two practical assignments are planned
  - You have to succeed in at least one
  - They will be graded
- Our concept for grading
  (may be changed – rules will be fixed before registration for the exam)
  - 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 examination grade, it is accounted for by another 25%

Questions

- Who studies what?
  - Diploma degree?
  - Master in Informatics?
  - Master in Informatics – English Track?
  - Master in Information Systems [Wirtschaftsinformatik]?
  - Other Master courses?
  - Bachelor in Informatics?
  - Other courses
- Who comes from where?

- Which previous relevant courses?
  - IN0010 - Grundlagen Rechnernetze und Verteilte Systeme?
  - What else?

Chair for Network Architectures and Services – Prof. Carle
Department of Computer Science
TU München

Overview

Technische Universität München

Chapter 1: Introduction

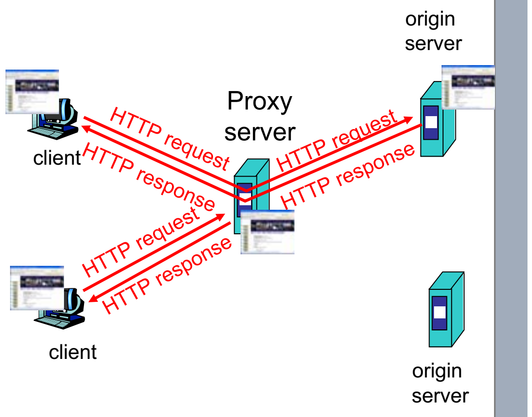
- what's the Internet?
- what's a protocol?
- network edge; hosts, access net, physical media
- network core: packet/circuit switching, Internet structure
- performance: loss, delay, throughput
- protocol layers, service models

Chapter 2: Application layer

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

Chapter 2: 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



origin server

Proxy server

client

HTTP request

HTTP response

HTTP request

HTTP response

HTTP request

HTTP response

client

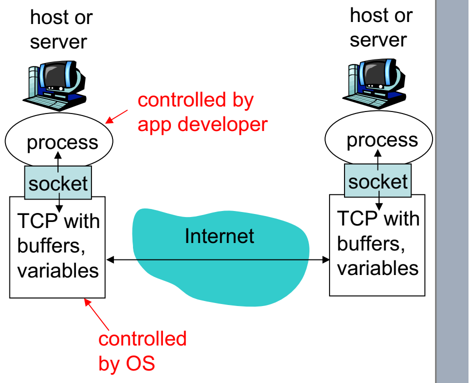
origin server

Chapter 3 Outline

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

Chapter 3: 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



host or server

host or server

controlled by app developer

process

process

socket

socket

TCP with buffers, variables

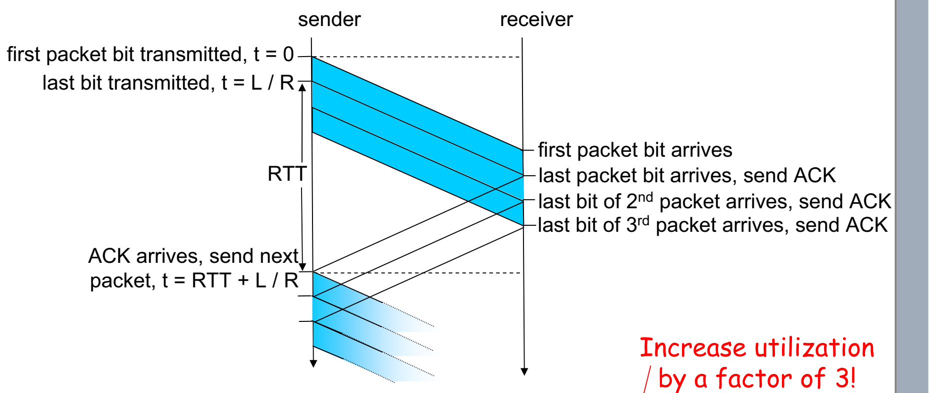
Internet

TCP with buffers, variables

controlled by OS

- API: (1) choice of transport protocol; (2) ability to fix a few parameters

Chapter 3: Pipelining: increased utilization


sender
receiver
first packet bit transmitted, t = 0
last bit transmitted, t = L / R
RTT
first packet bit arrives
last packet bit arrives, send ACK
last bit of 2nd packet arrives, send ACK
last bit of 3rd packet arrives, send ACK
ACK arrives, send next packet, t = RTT + L / R

Increase utilization by a factor of 3!

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

Chapter 3: 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:

```
DevRTT = (1-β)*DevRTT +
             β*|SampleRTT-EstimatedRTT|

(typically, β = 0.25)
```

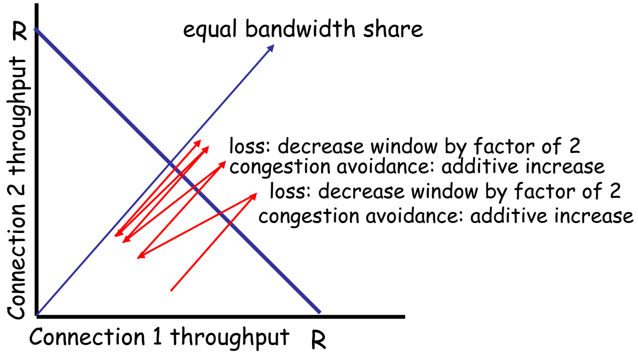
Then set timeout interval:

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

Chapter 3: Why is TCP fair?

Two competing sessions:

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



Chapter 4: Network Layer

Part 1

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

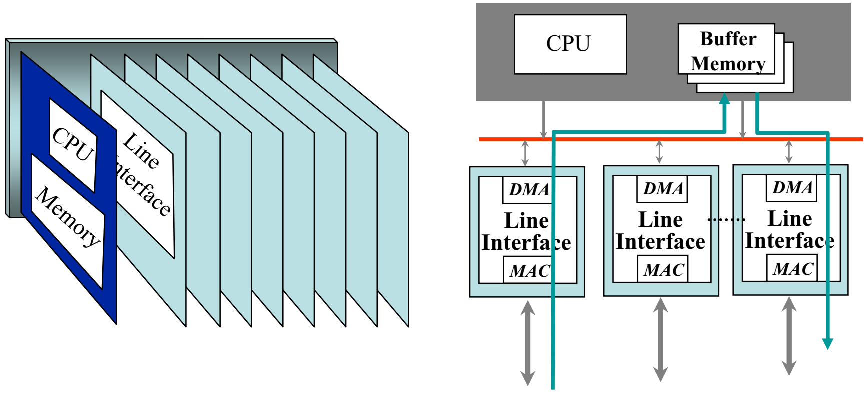
Part 2

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

Part 3

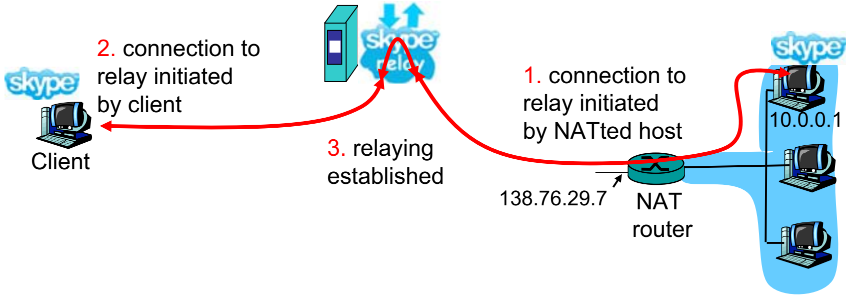
- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
- Broadcast and multicast routing

Chapter 4: First-Generation IP Routers



Chapter 4: NAT traversal

- One of several NAT traversal solutions: relaying (used in Skype)
  - NATed client establishes connection to relay
  - External client connects to relay
  - relay bridges packets between to connections



2. connection to relay initiated by client

Client

skype relay

3. relaying established

1. connection to relay initiated by NATted host

10.0.0.1

138.76.29.7

NAT router

Chapter 5: Link Layer

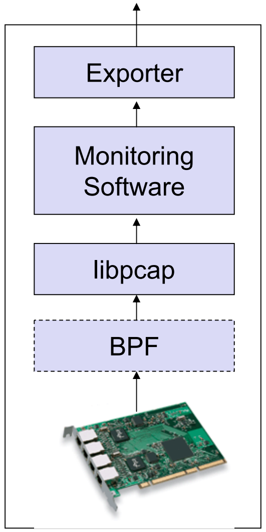
- Introduction and services
- Multiple access protocols
- Link-layer Addressing
- Ethernet
- Link-layer switches
- Link virtualization: ATM, MPLS

Chapter 6: Network Measurements

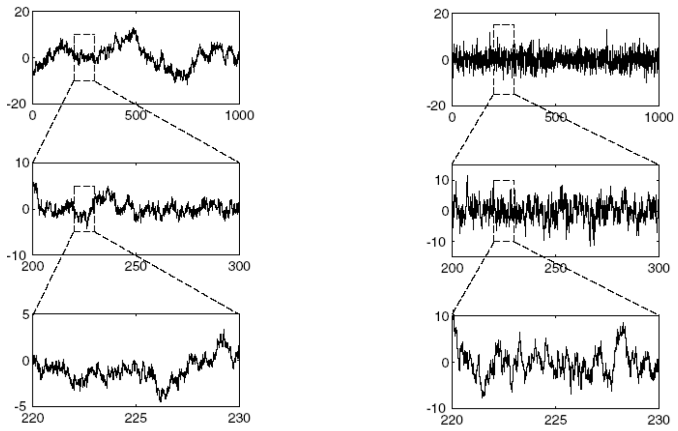
- Introduction
- Architecture & Mechanisms
- Protocols
  - IPFIX (Netflow Accounting)
  - PSAMP (Packet Sampling)
- Scenarios

Chapter 6: Monitoring Probe

- Standardized data export
- Monitoring Software
- HW adaptation, [filtering]
- OS dependent interface (BSD)
- Network interface



Chapter 6: Self-Similar Stochastic Process



(a) Self-Similar Process

(b) Non-Self-Similar Process

Chapter 7 outline – Quality-of-Service Support

- Link virtualization: ATM
- Providing multiple classes of service
- Providing Quality-of-Service (QoS) guarantees
- Signalling for QoS

Chapter 8: Signaling

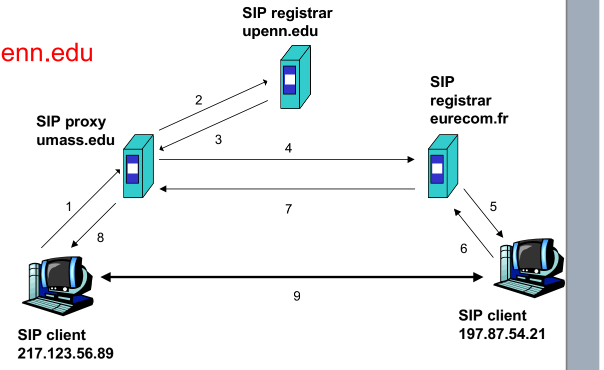
signaling: exchange of messages among network entities to enable (provide service) to connection/call

- 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

Chapter 9: Voice over IP 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.

Chapter 9: Network design principles

Network design principles

- *common themes:* signaling, indirection, virtualization, multiplexing, randomization, scalability
- *implementation principles:* techniques
- *network architecture:* the big picture, synthesis
- *Future Internet* approaches

Chapter 9: Design Principles and Future Internet

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

Chair for Network Architectures and Services – Prof. Carle
Department of Computer Science
TU München

Introduction

Technische Universität München

Chapter 1: Introduction

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*

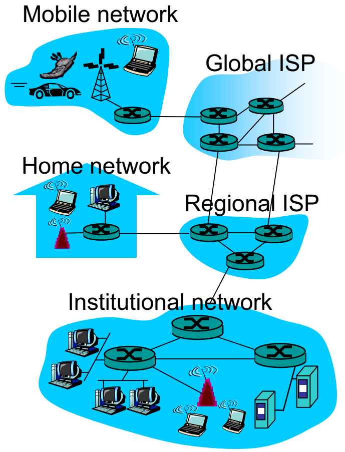
access points

wired links

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

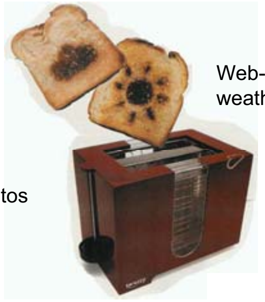
router

- ❑ *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

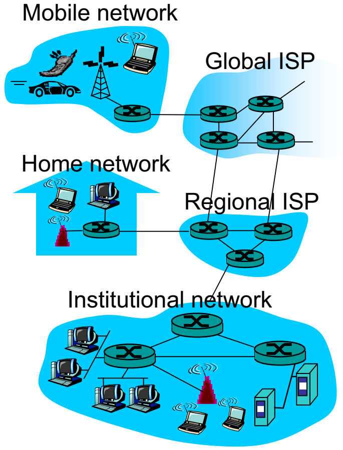
World's smallest web server in 1999

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

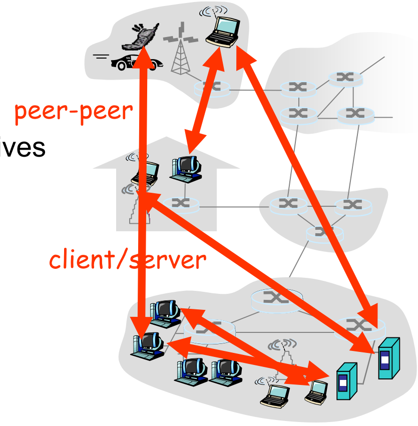
- ❑ circuit switching, packet switching, network structure

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



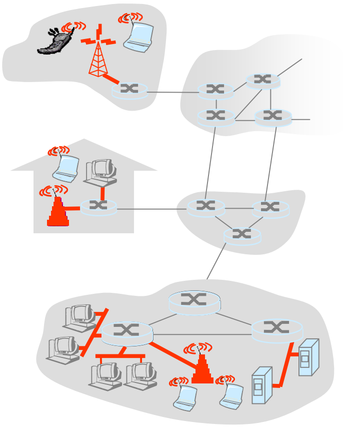
Access networks and physical media

Q: How to connect end systems to edge router?

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

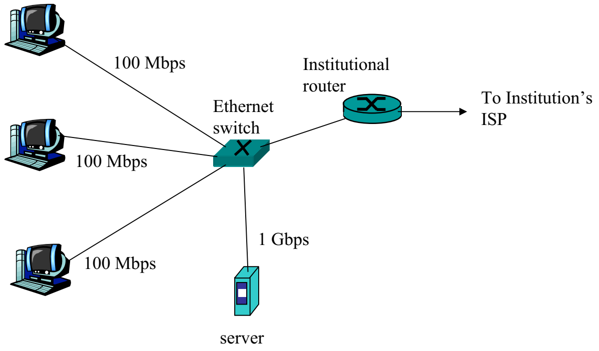
Relevant:

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



Ethernet Internet access

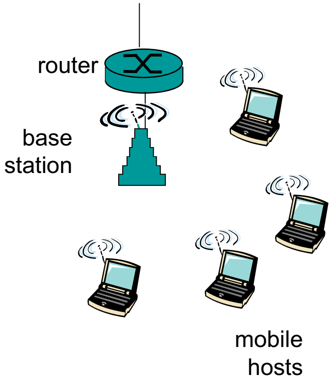
- Typically used in companies, universities, etc
  - 10 Mbs, 100Mbps, 1Gbps, 10Gbps Ethernet
  - Today, end systems typically connect into Ethernet switch



⇨ why?

Wireless access networks

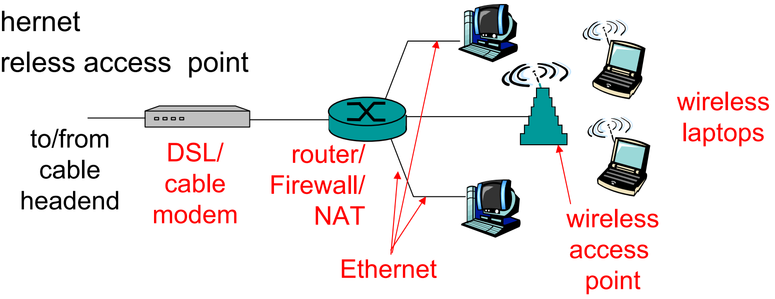
- shared *wireless* access network connects end system to router
  - via base station aka “access point”
- wireless LANs:
  - 802.11b/g (WiFi): 11 or 54 Mbps
- wider-area wireless access
  - provided by telco operator
  - ~1Mbps over cellular system (HSDPA)
  - next cellular network technology: LTE (10's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

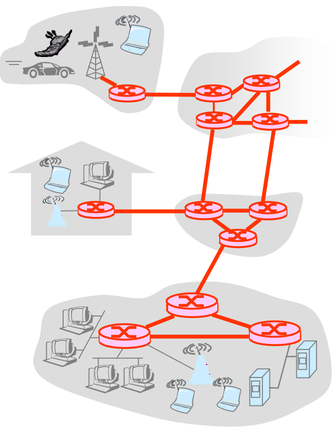
- circuit switching, packet switching, network structure

1.4 Delay, loss and throughput in packet-switched networks

1.5 Protocol layers, service models

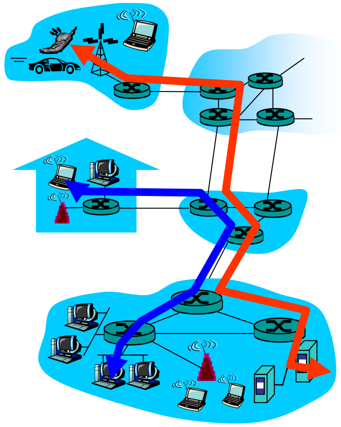
The Network Core

- 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



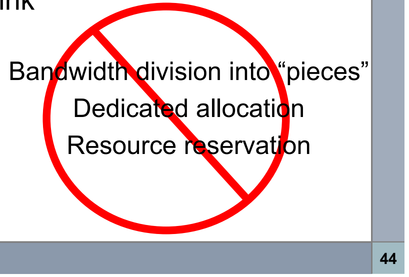
Network Core: Packet Switching

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

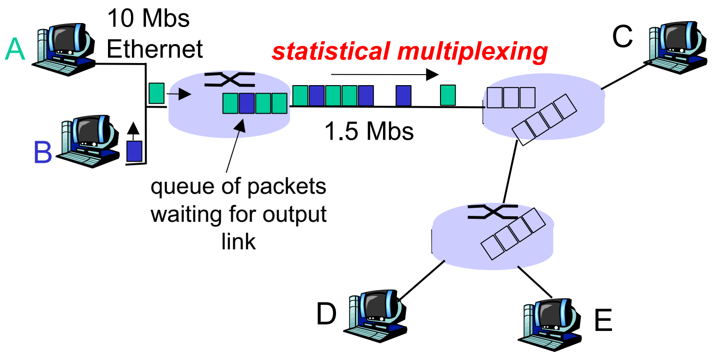


Bandwidth division into “pieces”

Dedicated allocation

Resource reservation

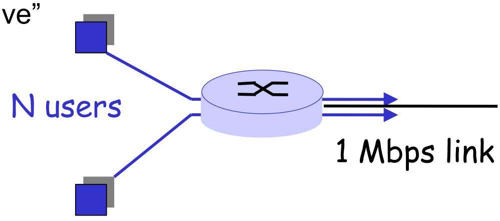
Packet Switching: Statistical Multiplexing



- 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

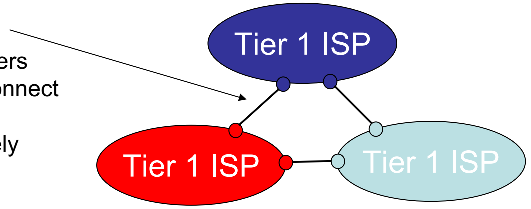
Is packet switching obviously better than circuit switching?

- 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)

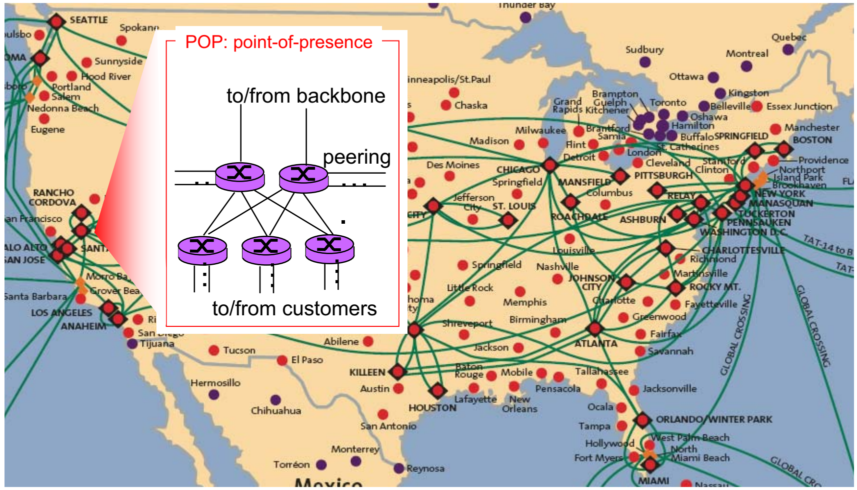
Internet structure: network of networks

- 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



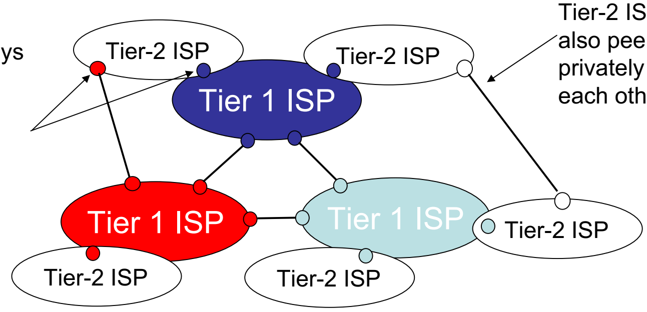
Tier-1 ISP: e.g., Sprint



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

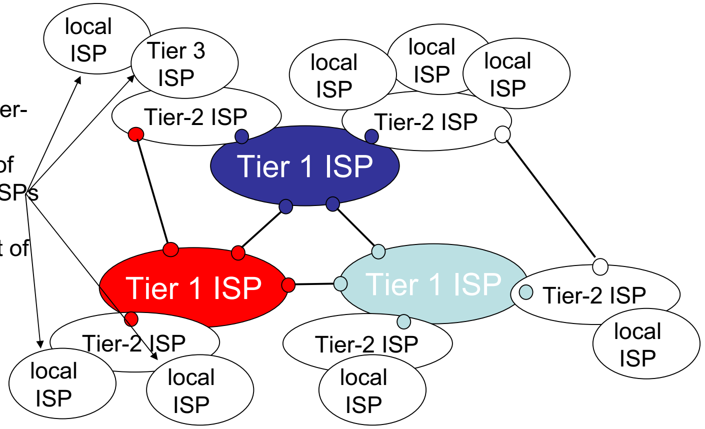


Tier-2 ISPs also peer privately with each other.

Internet structure: network of networks

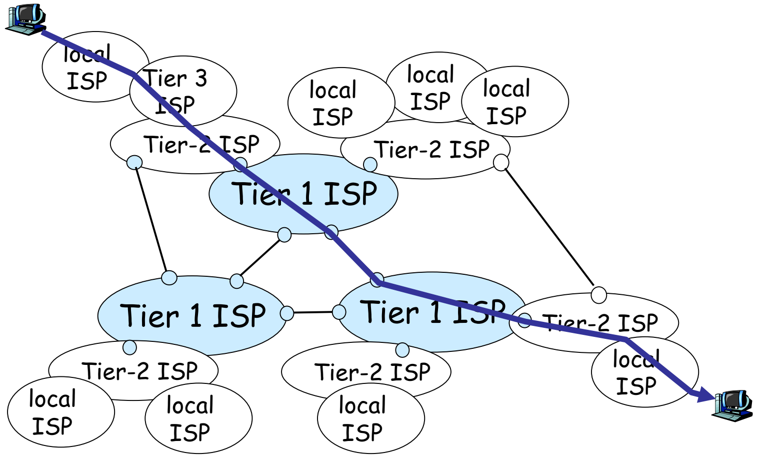
- “Tier-3” ISPs and local ISPs
  - last hop (“access”) network (closest to end systems)

Local and tier-3 ISPs are *customers* of higher tier ISPs connecting them to rest of Internet



Internet structure: network of networks

- 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

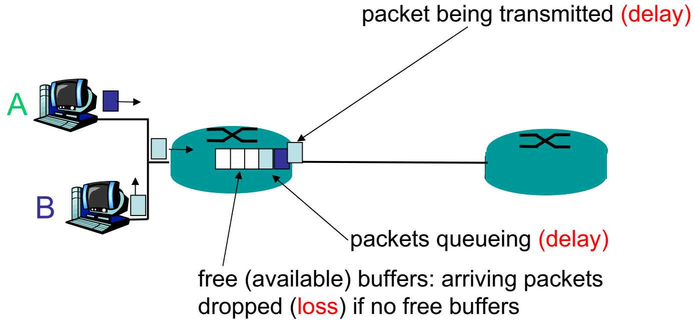
1.4 Delay, loss and throughput in packet-switched networks

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



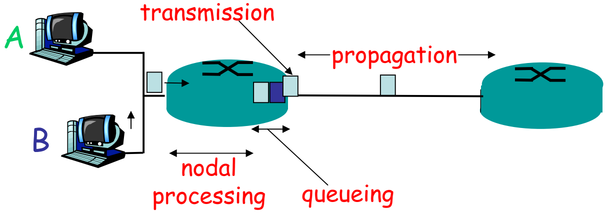
packet being transmitted (delay)

packets queueing (delay)

free (available) buffers: arriving packets dropped (loss) if no free buffers

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



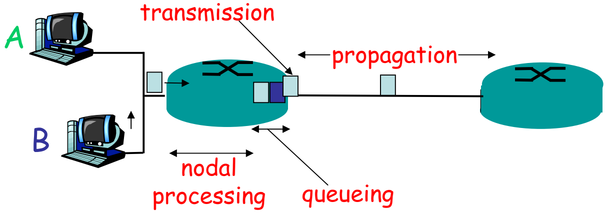
Delay in packet-switched networks

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

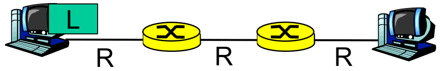


Nodal delay

- 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:

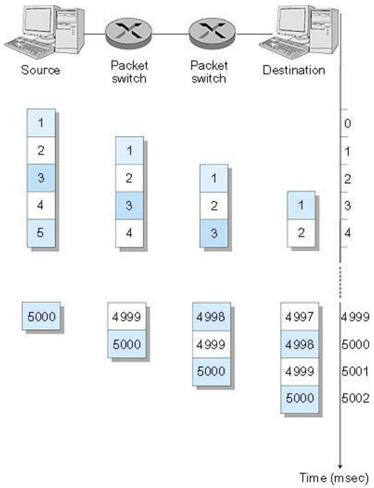
Circuit Switching:

- 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

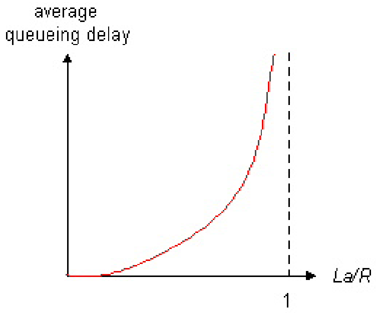
- Each packet 1,500 bits
- 1 msec to transmit packet on one link
- *pipelining:* each link works in parallel
- Delay reduced from 15 sec to 5.002 sec (as good as circuit switched)
- What did we achieve over circuit switching?
- Drawbacks (of packet vs. Message)

Queueing delay (revisited)

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

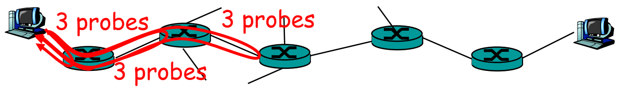
traffic intensity = L·a/R

- 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

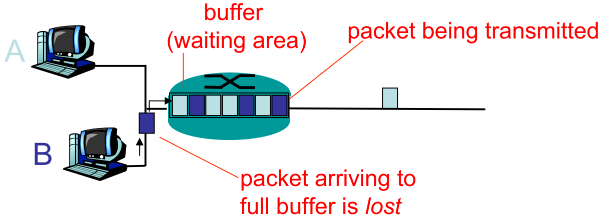
```
1  cs-gw (128.119.240.254)  1 ms  1 ms  2 ms
2  border1-rt-fa5-1-0.gw.umass.edu (128.119.3.145)  1 ms  1 ms  2 ms
3  cht-vbns.gw.umass.edu (128.119.3.130)  6 ms 5 ms 5 ms
4  jn1-at1-0-0-19.wor.vbns.net (204.147.132.129)  16 ms 11 ms 13 ms
5  jn1-so7-0-0-0.wae.vbns.net (204.147.136.136)  21 ms 18 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)

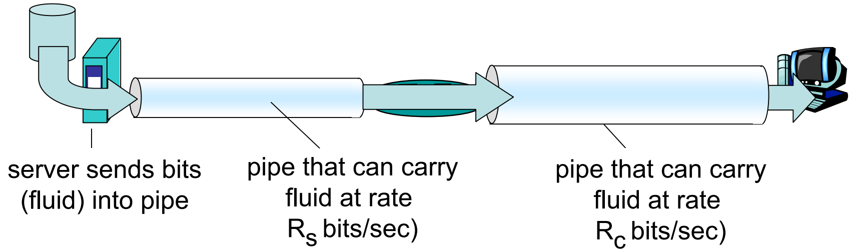
Packet loss

- 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



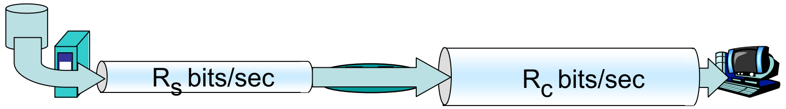
Throughput

- *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

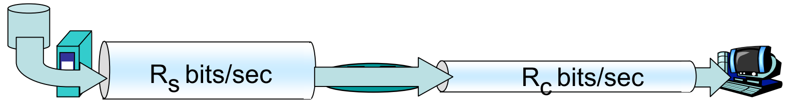


Throughput (more)

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



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



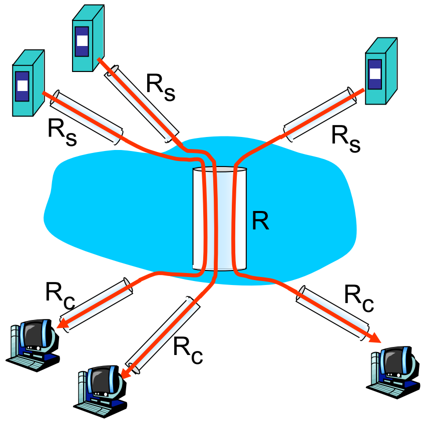
bottleneck link

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?

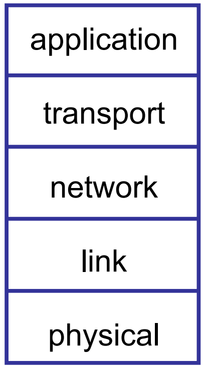
Why layering?

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
- ❑ Summary

Chapter 2: Application Layer

Our goals:

- ❑ conceptual, implementation aspects of network application protocols
  - transport-layer service models
  - client-server paradigm
  - peer-to-peer paradigm
- ❑ learn about protocols by examining popular application-level protocols
  - HTTP
  - DNS
- ❑ programming network applications
  - socket API

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

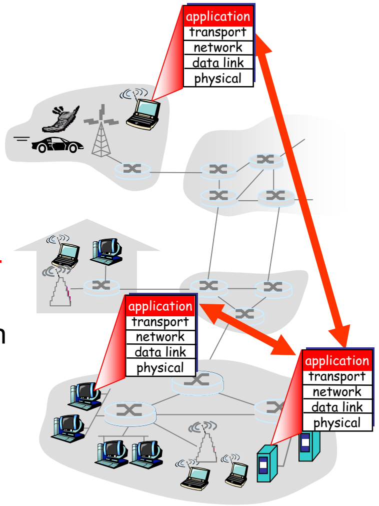
Creating a network application

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 whether a counter-example exists:
what would be benefits if you could program your router?

Chapter 2: Application layer

- ❑ **Principles of network applications**
- ❑ Web and HTTP
- ❑ DNS
- ❑ P2P applications
- ❑ Summary

Application architectures

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

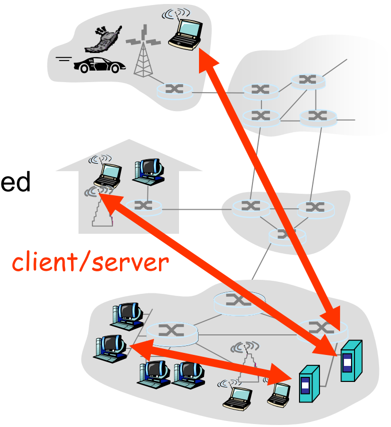
Client-server architecture

server:

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

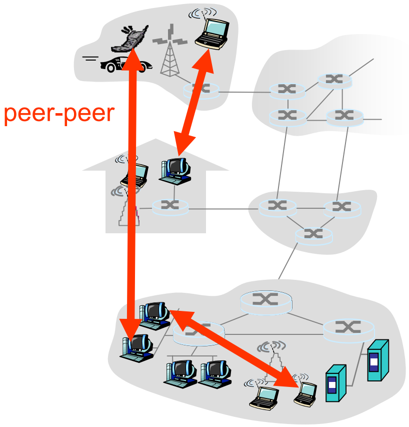
clients:

- 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

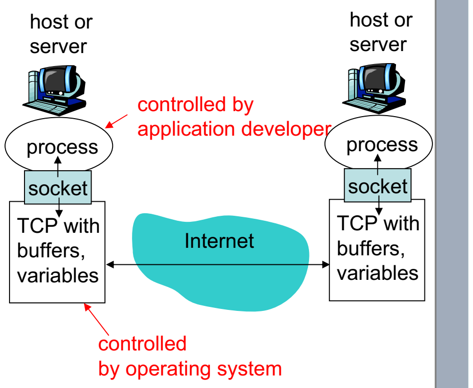
Client process: process that initiates communication

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



host or server

process

socket

TCP with buffers, variables

controlled by application developer

Internet

host or server

process

socket

TCP with buffers, variables

controlled by operating system

- API: (1) choice of transport protocol; (2) ability to fix a few parameters

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

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

Timing

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

Security

- 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:* throttle sender 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

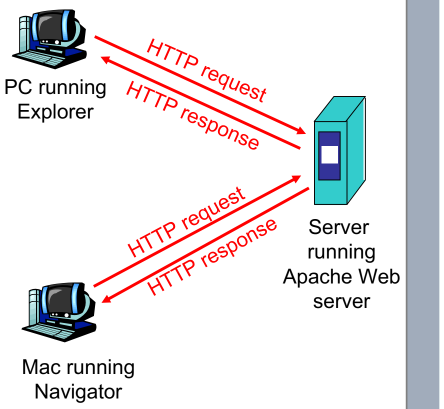
Chapter 2: Application layer

- Principles of network applications
- **Web and HTTP**
- DNS
- P2P applications
- Summary

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 at 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 after HTTP response

HTTP is "stateless"

- server maintains no information about past client requests

aside

Protocols that maintain "state" are complex!

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

HTTP connections

Nonpersistent HTTP (v1.0)

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

Persistent HTTP (v1.1)

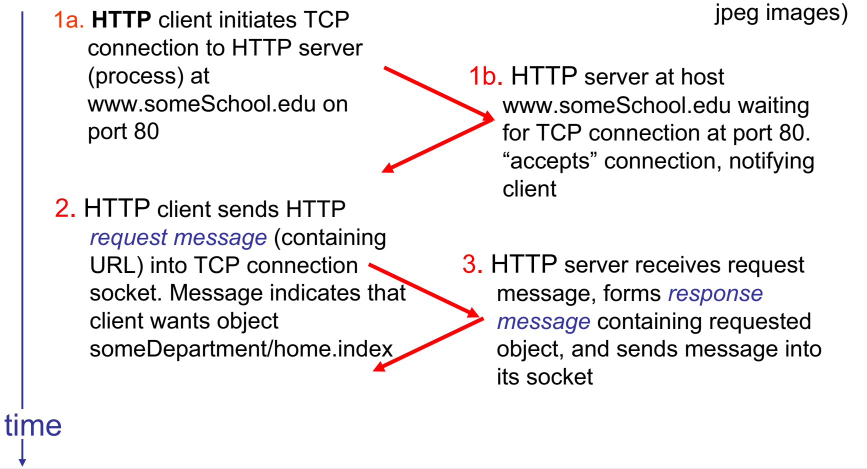
- Multiple objects can be sent over single TCP connection between client and server.

Nonpersistent HTTP

Suppose user enters URL

`www.someSchool.edu/someDepartment/home.index`

(contains text, references to 10 jpeg images)



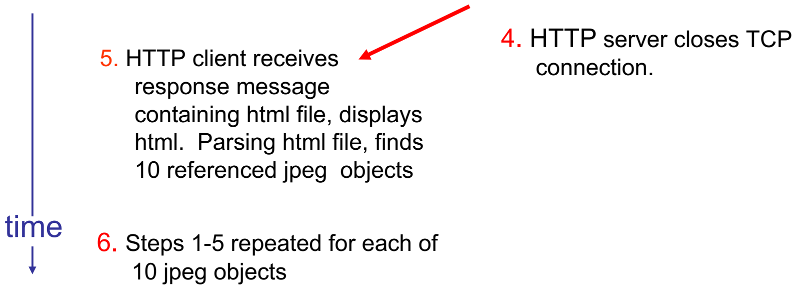
1a. **HTTP** client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80

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

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

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

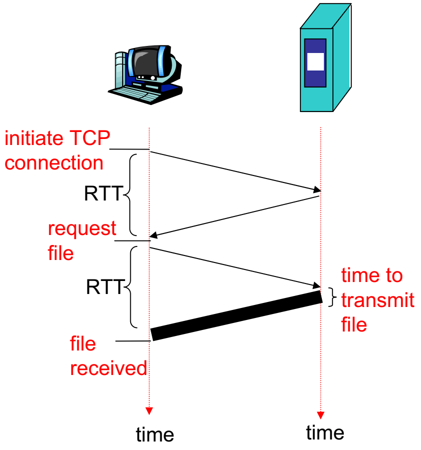
Non-Persistent HTTP: Response time

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:

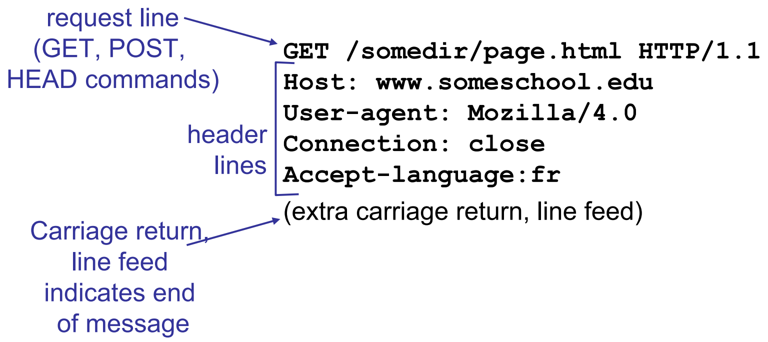
- requires 2 RTTs per object
- OS overhead for *each* TCP connection
- browsers often open parallel TCP connections to fetch referenced objects

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)



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

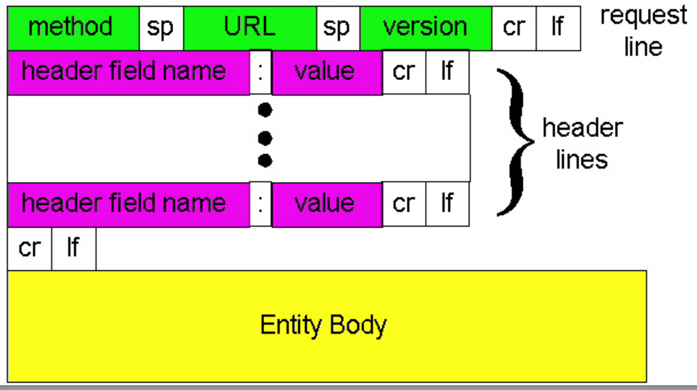
(extra carriage return, line feed)

request line (GET, POST, HEAD commands)
header lines
Carriage return, line feed indicates end of message

HTTP request message: general format

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

(extra carriage return, line feed)



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

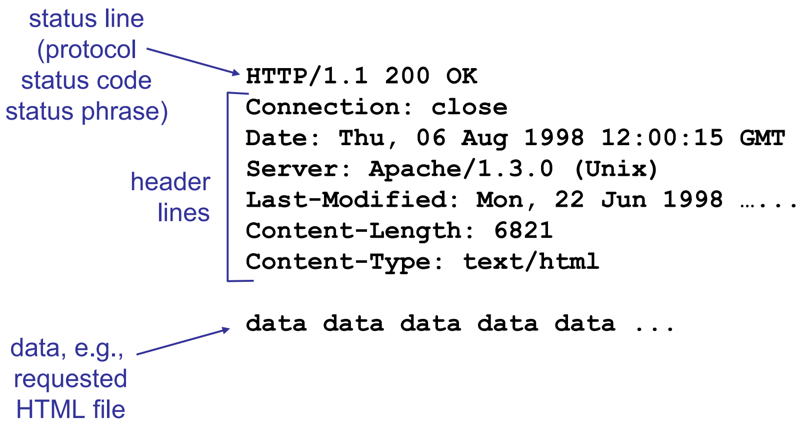
HTTP/1.0

- 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:

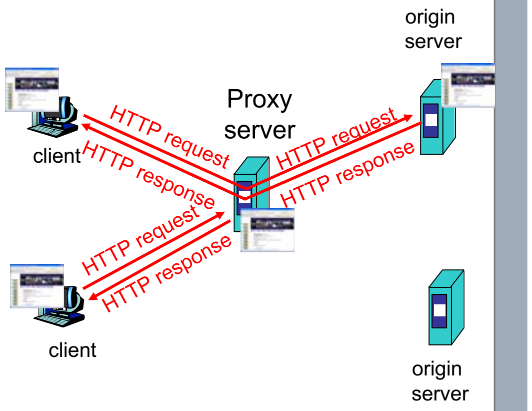
GET /~ross/ HTTP/1.1
Host: cis.poly.edu

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
- 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
- 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)

Example

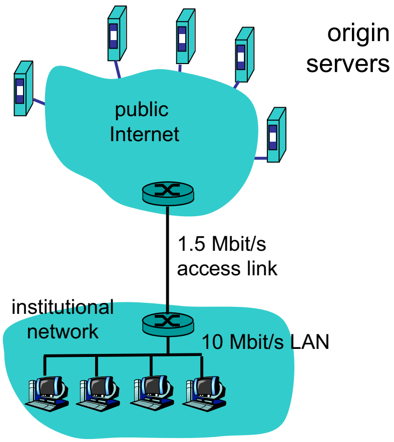
Assumptions

- 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

- traffic intensity (utilization) on LAN = 15%
- traffic intensity (utilization) on access link = 100%
- total delay = Internet delay + access delay + LAN delay

  = 2 sec + minutes + milliseconds



Example (cont)

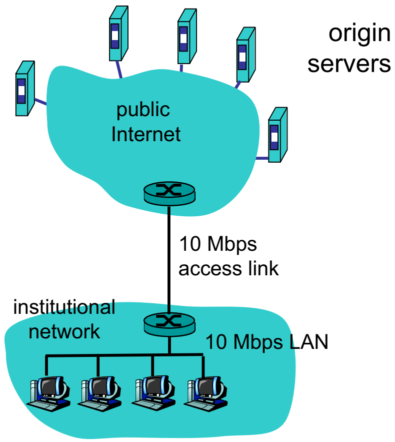
possible solution

- 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



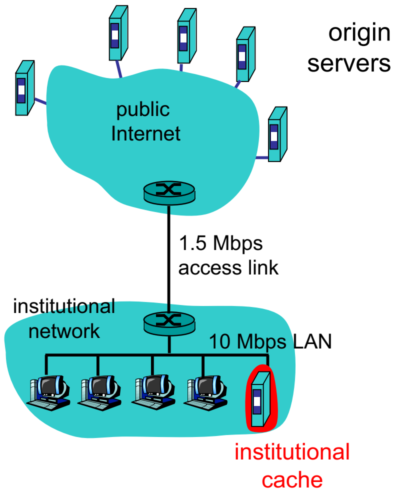
Example (cont)

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 average delay
  = 60%*{ Internet delay
  + access delay
  + LAN delay}
  + 40% * milliseconds
  = 0.6*(2.01) sec
  + 0.4*milliseconds
  ≈ 1.2 secs

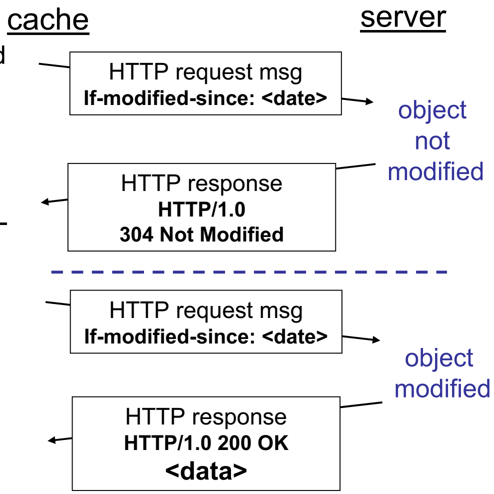


Conditional GET

- 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`



cache

server

HTTP request msg
If-modified-since: <date>

object not modified

HTTP response
HTTP/1.0
304 Not Modified

HTTP request msg
If-modified-since: <date>

object modified

HTTP response
HTTP/1.0 200 OK
<data>

Chapter 2: Application layer

- Principles of network applications
- Web and HTTP
- **DNS**
- P2P applications
- Summary

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?

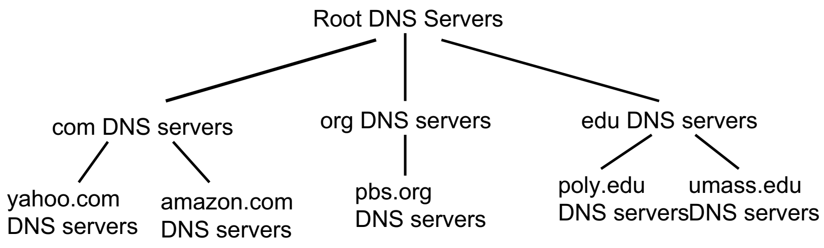
- single point of failure
- traffic volume
- distant centralized database
- maintenance

doesn't *scale!*

DNS services

- hostname to IP address translation
- host aliasing
  - canonical name
  - alias names
- mail server aliasing
  - mnemonic host name desired
  - MX record allows mnemonic host name reused for mail server
- load distribution
  - replicated Web servers: set of IP addresses for one canonical name

Distributed, Hierarchical Database



Root DNS Servers

com DNS servers

org DNS servers

edu DNS servers

yahoo.com DNS servers

amazon.com DNS servers

pbs.org DNS servers

poly.edu DNS servers

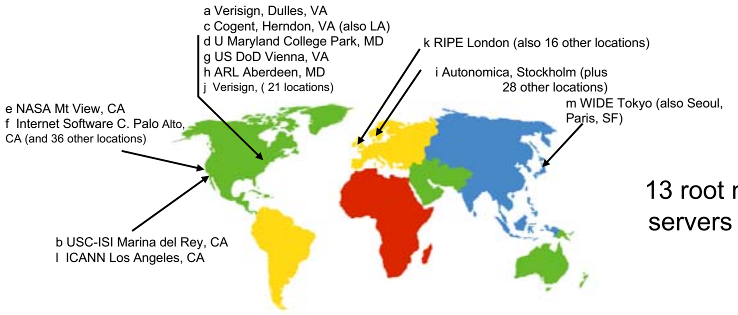
umass.edu DNS servers

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 (authorative DNS server – configured by original source) 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



a Verisign, Dulles, VA
c Cogent, Herndon, VA (also LA)
d U Maryland College Park, MD
g US DoD Vienna, VA
h ARL Aberdeen, MD
j Verisign, (21 locations)
k RIPE London (also 16 other locations)
i Autonomica, Stockholm (plus 28 other locations)
m WIDE Tokyo (also Seoul, Paris, SF)
e NASA Mt View, CA
f Internet Software C. Palo Alto, CA (and 36 other locations)
b USC-ISI Marina del Rey, CA
l ICANN Los Angeles, CA

13 root name servers worldwide

TLD and Authoritative Servers

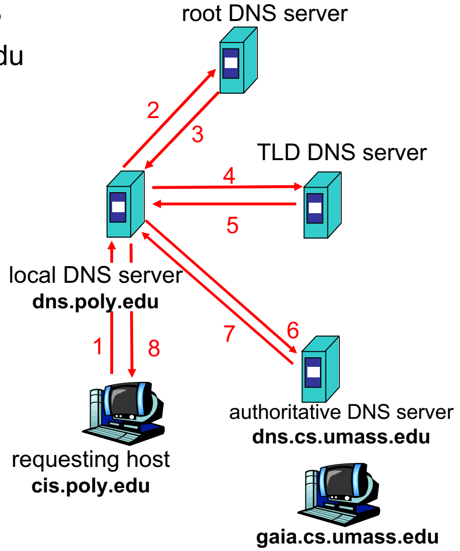
- **Top-level domain (TLD) servers:**
  - responsible for com, org, net, edu, etc, and all top-level country domains uk, fr, ca, jp.
  - organisations hosting TLD servers:
    - Network Solutions maintains servers for com TLD
    - Educause for edu TLD
- **Authoritative DNS servers:**
  - organization's DNS servers, providing authoritative hostname to IP mappings for organization's servers (e.g., Web, 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



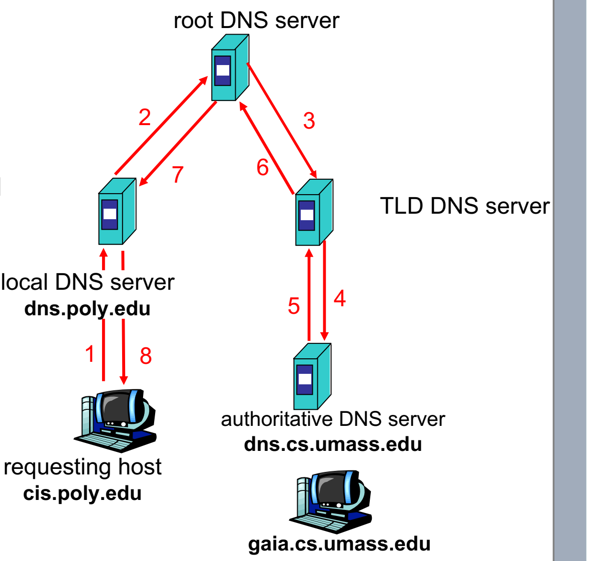
iterated query:

- 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
  - RFC 2136
  - http://www.ietf.org/html.charters/dnsind-charter.html
  - „notify“ mechanism: primary sends a message to known secondaries. for fast convergence of servers

DNS records

DNS: distributed db storing resource records (RR)

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

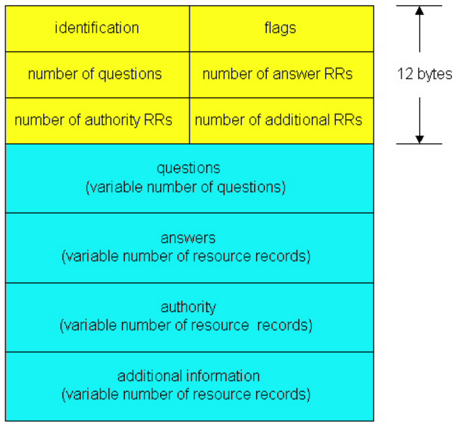
- Type=A
  - name is hostname
  - value is IP address
- Type=NS
  - **`name`** is domain (e.g. foo.com)
  - **`value`** is hostname of authoritative name server for this domain
- Type=CNAME
  - name is alias name for some "canonical" (the real) name
  - e.g.: www.ibm.com is really servereast.backup2.ibm.com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

DNS protocol, messages

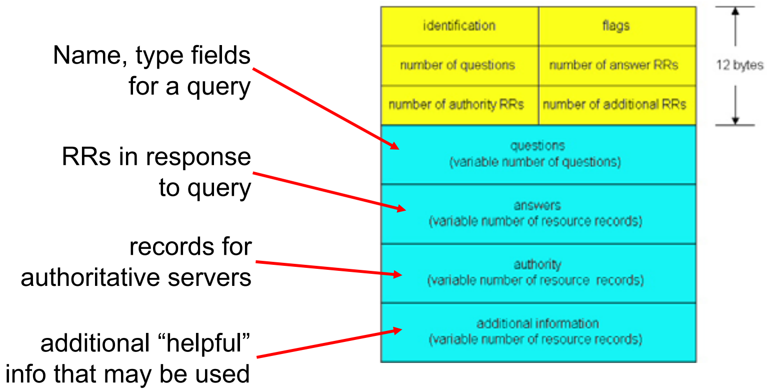
DNS protocol : *query* and *reply* messages, both with same *message format*

message header

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



DNS protocol, messages


Name, type fields for a query
RRs in response to query
records for authoritative servers
additional "helpful" info that may be used
identification
flags
number of questions
number of answer RRs
number of authority RRs
number of additional RRs
12 bytes
questions
(variable number of questions)
answers
(variable number of resource records)
authority
(variable number of resource records)
additional information
(variable number of resource records)

Inserting records into DNS

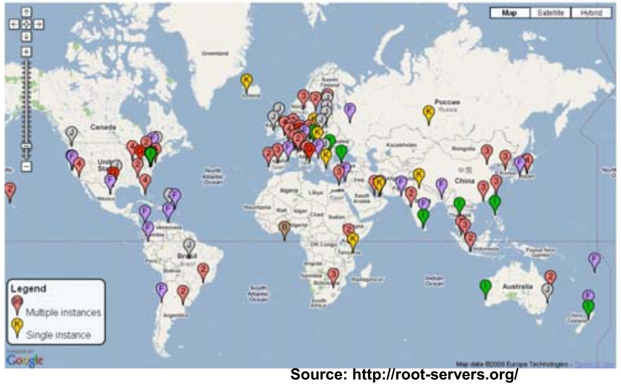
- example: new startup "Network Utopia"
- register name networkuptopia.com at *DNS registrar* (e.g., Network Solutions)
  - provide names, IP addresses of authoritative name server (primary and secondary)
  - registrar inserts two RRs into com TLD server:

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

- create authoritative server
  Type A record for www.networkuptopia.com
  Type MX record for networkutopia.com

DNS Root Servers

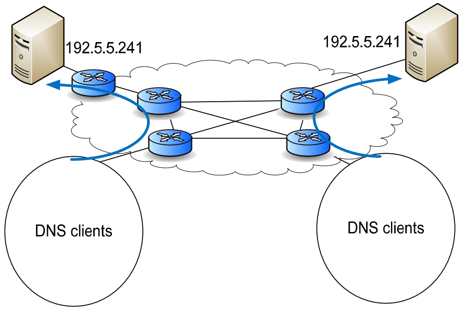
- 13 root servers (A to M)
- But number of physical servers in total is higher
- and increasing:
  - 191 by Oct. 2009
  - 229 by Oct. 2010



Source: http://root-servers.org/

DNS and IP Anycast

- Multiple servers can be made reachable under the same IP address
- Via *IP anycast*
- E.g. F-root server (IPv4: 192.5.5.241; IPv6: 2001:500:2f::f)



- IP anycast used for DNS since 2002 for root servers and many TLDs

➔ High robustness

➔ New servers can be easily added without updating the DNS clients.

DNS Caching

- TTL not specified in the standard (RFC 1034-1035)
- But in practice TTLs often up to 24 hours
- Records for TLDs are provided by root servers and typically stored even for 48 hours

- Caching typically improves lookup performance
- Caching relieves upper nodes in the hierarchy (root + TLDs)

- Massive caching makes it difficult to:
  - Dynamically react to current load
  - Migrate services

  ➔ TTLs of 60 s are typical today (e.g. amazon.com)

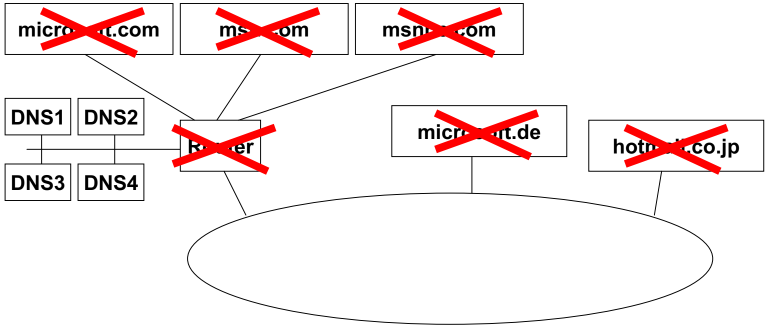
Example: DNS with Low TTLs

- e.g. amazon.com

```
user@host:~$ dig amazon.com
; <<>> DiG 9.6.1-P2 <<>> amazon.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42197
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 7,
   ADDITIONAL: 9

;; QUESTION SECTION:
;amazon.com.                       IN      A

;; ANSWER SECTION:
amazon.com.          60       IN      A       72.21.210.250
amazon.com.          60       IN      A       207.171.166.252
amazon.com.          60       IN      A       72.21.207.65
```

Dependency on DNS

- DoS-Attack targeting Microsoft in January 2001
  - First: router problem ➔ Microsoft's websites and services were down on January 23rd 2001
  - The damage was surprisingly large



Dependency on DNS

- Web servers are be running
- But DNS failure leads to service failure

➔ Need to deploy multiple DNS authorative servers

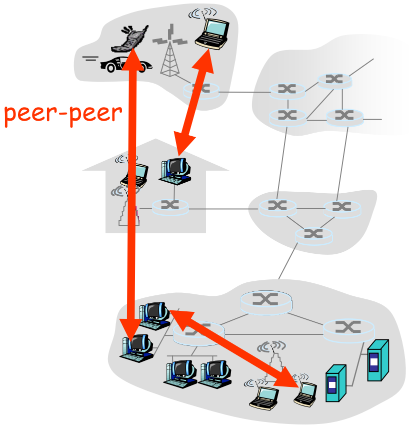
➔ In different networks

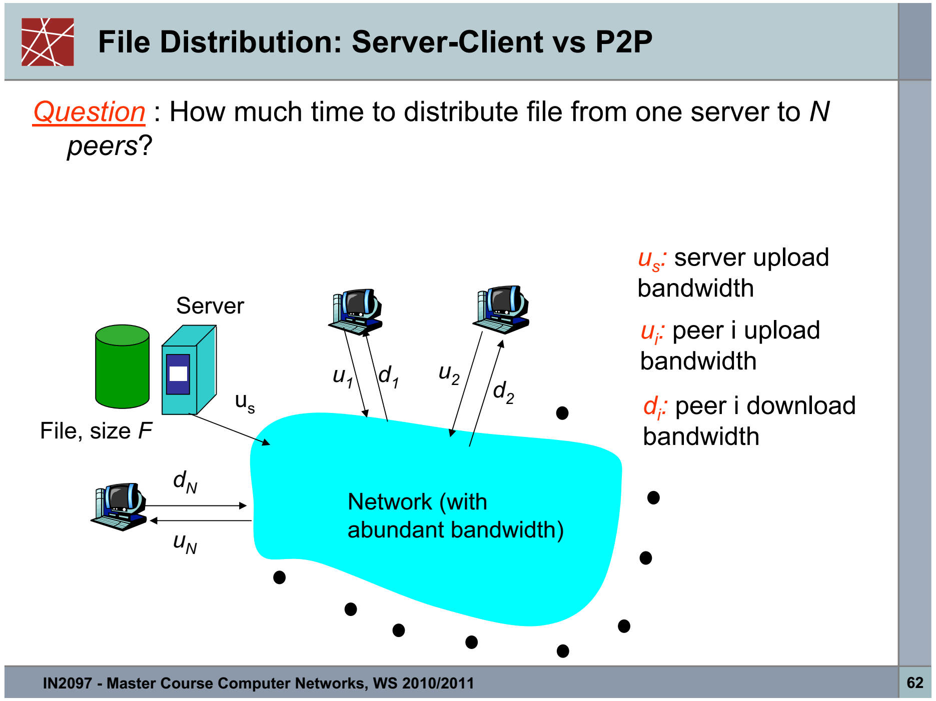
Chapter 2: Application layer

- Principles of network applications
- Web and HTTP
- DNS
- **P2P applications**
- Summary

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





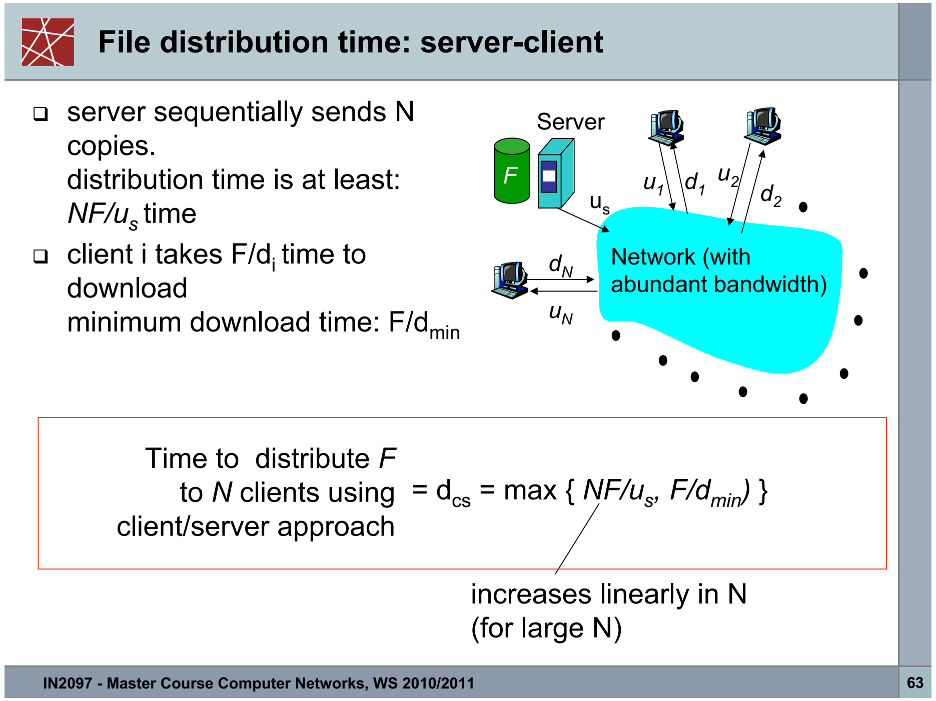
File Distribution: Server-Client vs P2P

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

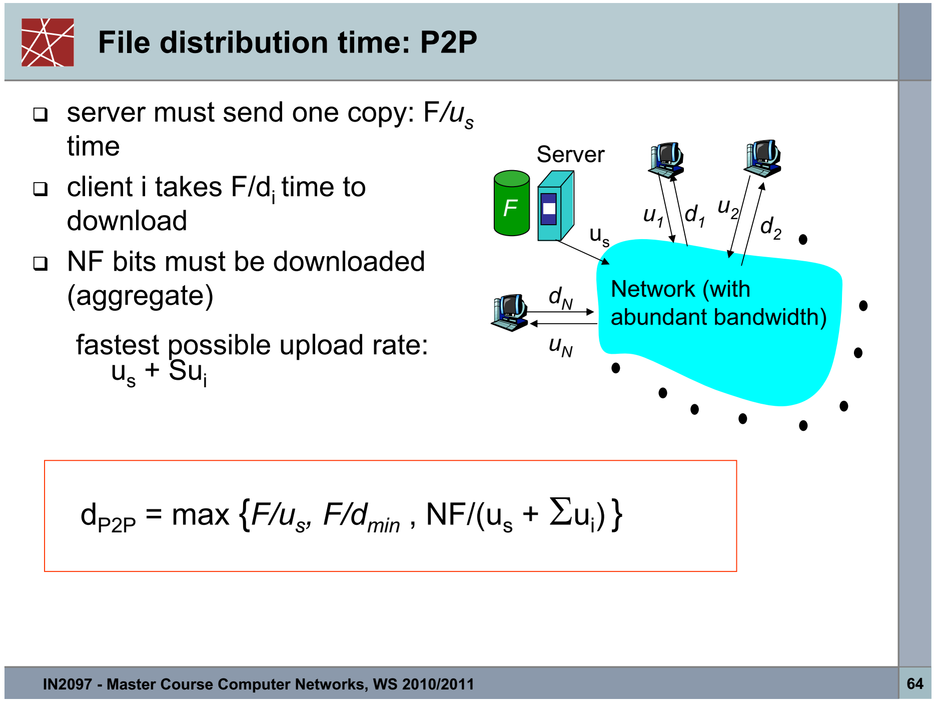


File distribution time: server-client

- server sequentially sends N copies.
  distribution time is at least: NF/u_s time
- client i takes F/d_i time to download
  minimum download time: F/d_{min}

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

increases linearly in N (for large N)

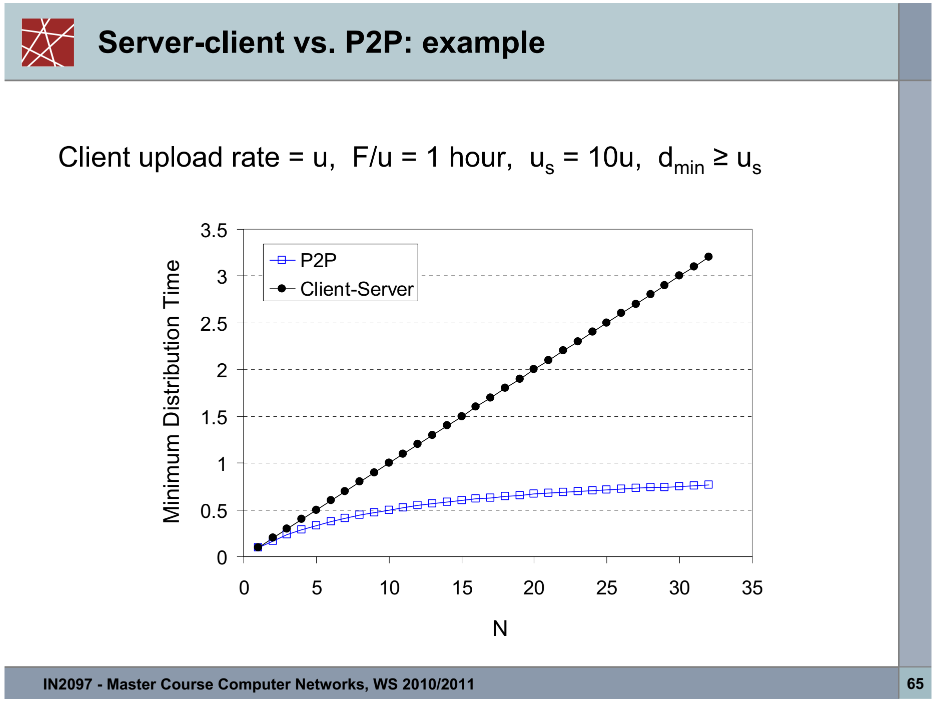


File distribution time: P2P

- 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 + Su_i

$$d_{P2P} = \max \{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$



Server-client vs. P2P: example

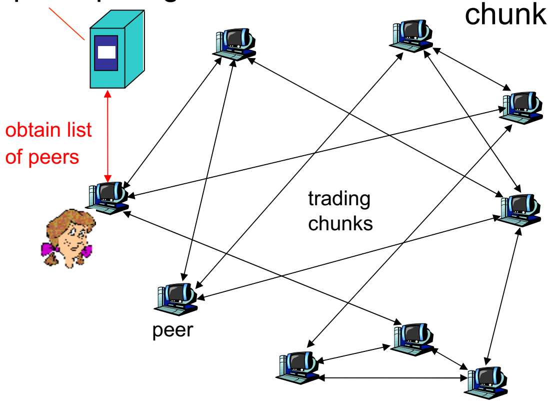
Client upload rate = u, F/u = 1 hour, $u_s = 10u$, $d_{min} \geq u_s$

File distribution: BitTorrent

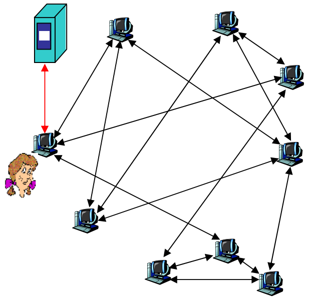
- P2P file distribution

tracker: tracks peers participating in torrent

torrent: group of peers exchanging chunks of a file



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

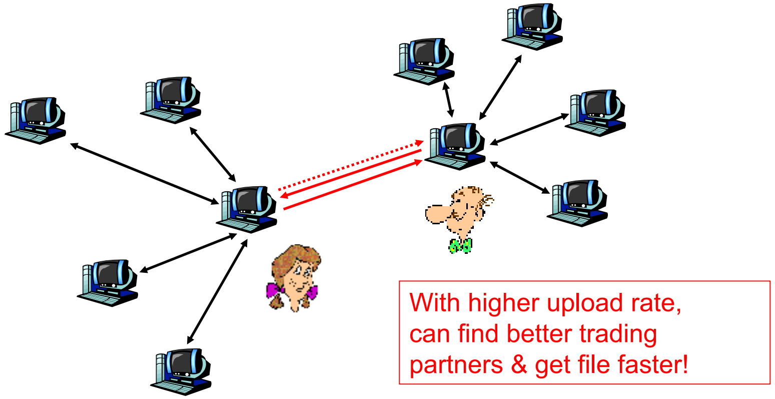
- Alice sends chunks to four neighbors currently sending her chunks *at the highest rate*
  - re-evaluate top 4 every 10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - newly chosen peer may join top 4
  - “optimistically unchoke”

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: social security number; value: human name
  - key: content identifier; value: IP address
- Peers query DB with key
  - DB returns values that match the key
- Peers can also insert (key, value) peers

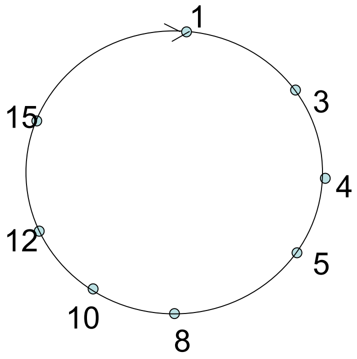
DHT Identifiers

- 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.
- Example: n=4; peers: 1,3,4,5,8,10,12,14;
  - key = 13, then successor peer = 14
  - key = 15, then successor peer = 1

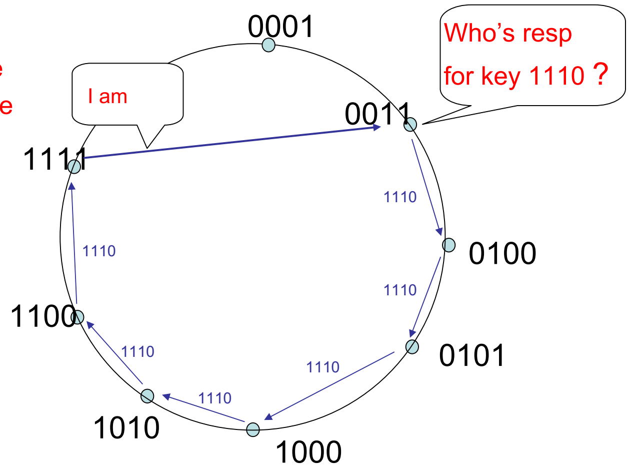
Circular DHT (1)



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

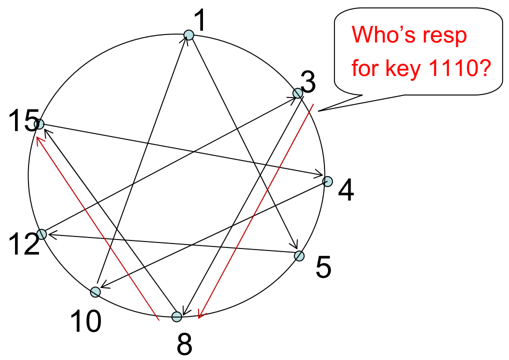
Circle DHT (2)

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



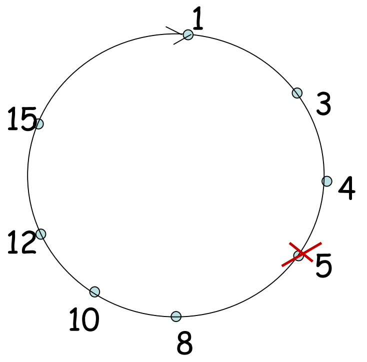
Define closest as closest successor

Circular DHT with Shortcuts



- ❑ Each peer keeps track of IP addresses of predecessor, successor, short cuts.
- ❑ Shortcuts reduce required number of query messages (e.g. from 6 to 2).
- ❑ Possible to design shortcuts so O(log N) neighbors, O(log N) messages in query

Peer Churn

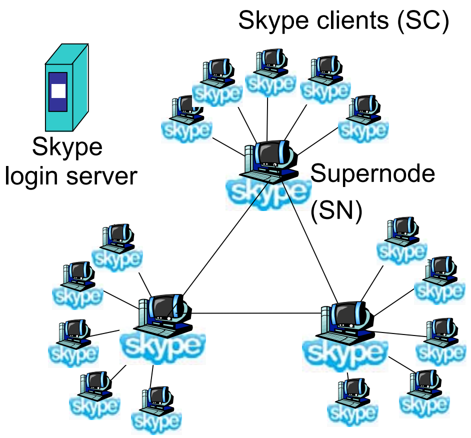


- To handle peer churn, require each peer to know the IP address of its two successors.
- Each peer periodically pings its two successors to see if they are still alive.

- ❑ 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?

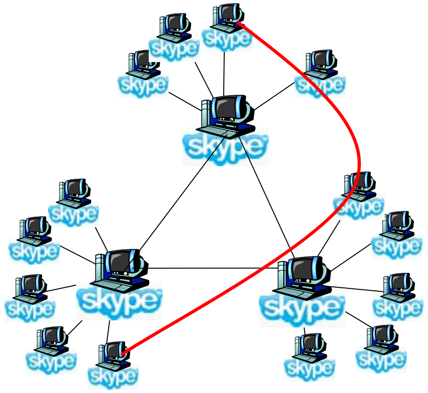
P2P Case study: Skype

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



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 Supernodes, 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
- **Summary**

Chapter 2: Summary

network application level issues

- 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 messages
  - in-band, out-of-band
- centralized vs. decentralized
- stateless vs. stateful
- reliable vs. unreliable message transfer
- "complexity at network edge"

Chair for Network Architectures and Services – Prof. Carle
Department of Computer Science
TU München

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

Technische Universität München

Chapter 3: Transport Layer

Our goals:

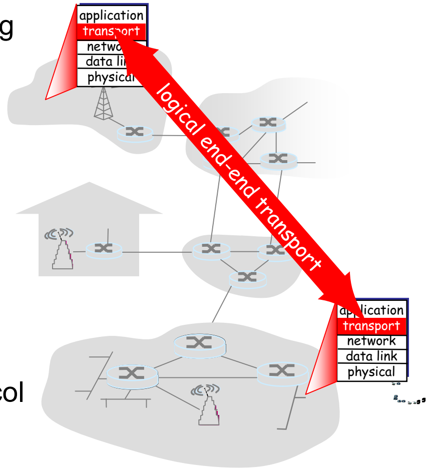
- understand principles behind transport layer services:
  - multiplexing/demultiplexing
  - reliable data transfer
  - flow control
  - congestion control
- learn about transport layer protocols in the Internet:
  - UDP: connectionless transport
  - TCP: connection-oriented transport
  - TCP congestion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

Chapter 3 outline

- **Transport-layer services**
- Multiplexing and demultiplexing
- Connectionless transport: UDP
- Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- TCP congestion control
- SCTP
- Reliable Multicast

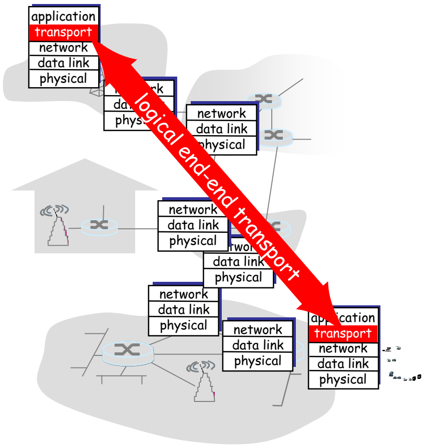
Transport services and protocols

- 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

- Transport-layer services
- **Multiplexing and demultiplexing**
- Connectionless transport: UDP
- Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- TCP congestion control
- SCTP
- Reliable Multicast

Multiplexing/demultiplexing

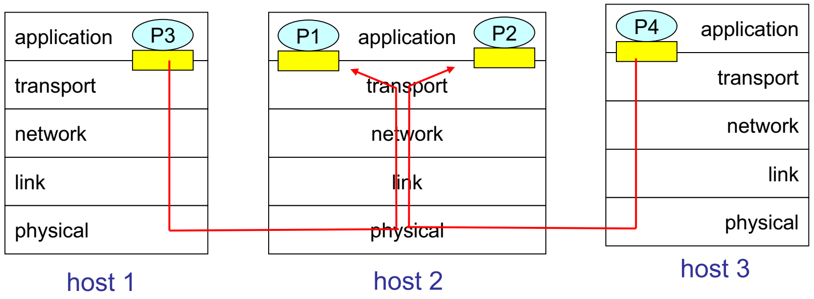
Demultiplexing at rcv host:

delivering received segments to correct socket

Multiplexing at send host:

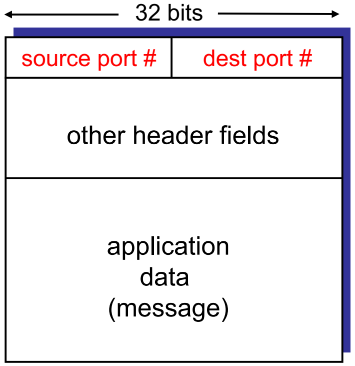
gathering data from multiple sockets, enveloping data with header (later used for demultiplexing)

🟨 = socket ○ = process



How demultiplexing works

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



TCP/UDP segment format

Connectionless demultiplexing

- Create sockets with port numbers:

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

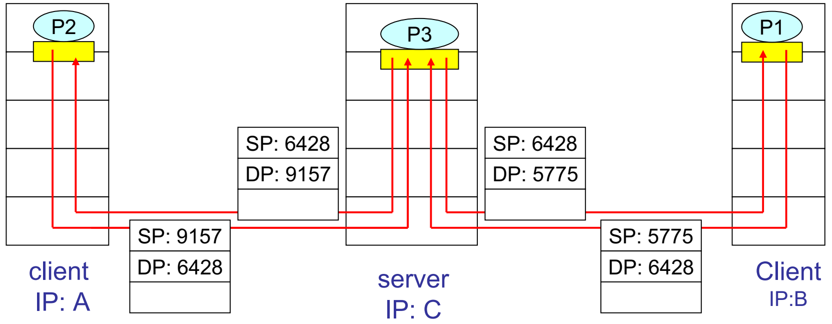
- UDP socket identified by two-tuple:

  (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);
```

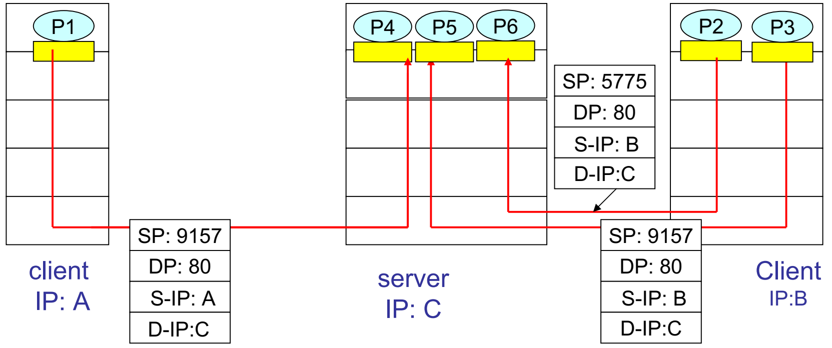


Source Port (SP) provides “return address”

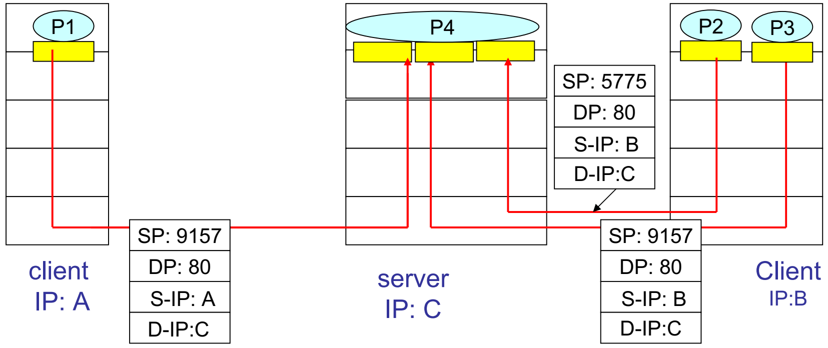
Connection-oriented demux

- 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

- Transport-layer services
- Multiplexing and demultiplexing
- **Connectionless transport: UDP**
- Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- TCP congestion control
- SCTP
- Reliable Multicast

UDP: User Datagram Protocol [RFC 768]

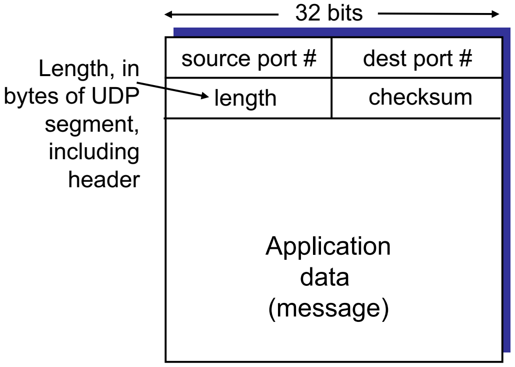
- “no frills,” “bare bones” Internet transport protocol
- “best effort” service, UDP segments may be:
  - lost
  - delivered out of order to app
- *connectionless:*
  - no handshaking between UDP sender, receiver
  - each UDP segment handled independently of others

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!



32 bits
source port #
dest port #
length
checksum
Length, in bytes of UDP segment, including header
Application data (message)

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 ….

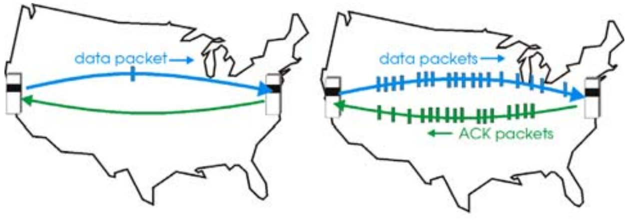
Chapter 3 outline

- Transport-layer services
- Multiplexing and demultiplexing
- Connectionless transport: UDP
- **Principles of reliable data transfer: Pipelining**
- Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- Principles of congestion control
- TCP congestion control
- SCTP
- Reliable Multicast

Pipelined protocols

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

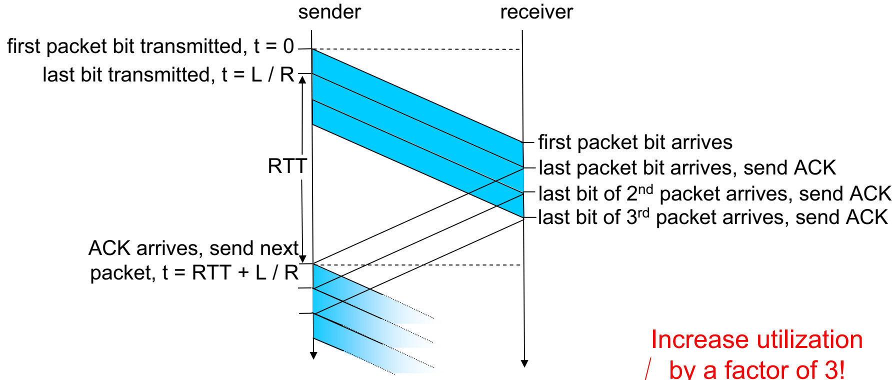
- range of sequence numbers must be increased
- buffering at sender and/or receiver



(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



sender

receiver

first packet bit transmitted, t = 0

last bit transmitted, t = L / R

RTT

ACK arrives, send next packet, t = RTT + L / R

first packet bit arrives

last packet bit arrives, send ACK

last bit of 2nd packet arrives, send ACK

last bit of 3rd packet arrives, send ACK

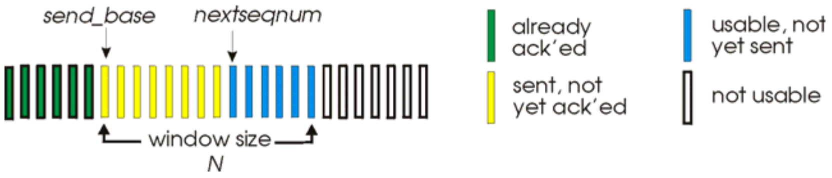
Increase utilization by a factor of 3!

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

Go-Back-N

Sender:

- k-bit seq # in pkt header
- “window” of up to N, consecutive unack’ed pkts allowed



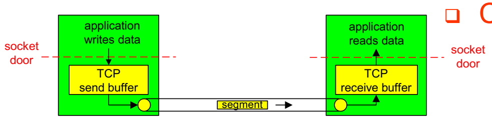
- ACK(n): ACKs all pkts up to, including seq # n - “cumulative ACK”
  - may receive duplicate ACKs (see receiver)
- timer for each in-flight pkt
- *timeout(n):* retransmit pkt n and all higher seq # pkts in window

Chapter 3 outline

- Transport-layer services
- Multiplexing and demultiplexing
- Connectionless transport: UDP
- Principles of reliable data transfer
- **Connection-oriented transport: TCP**
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- Principles of congestion control
- TCP congestion control
- SCTP
- Reliable Multicast

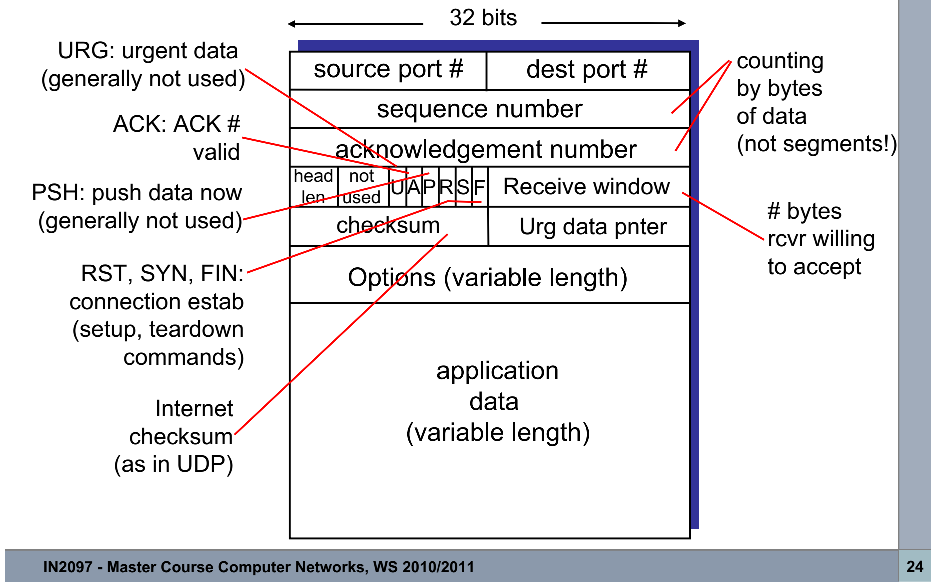
TCP: Overview RFCs: 793, 1122, 1323, 2018, 2581

- 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



32 bits
source port #
dest port #
sequence number
acknowledgement number
head len
not used
U
A
P
R
S
F
Receive window
checksum
Urg data pnter
Options (variable length)
application data (variable length)
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:

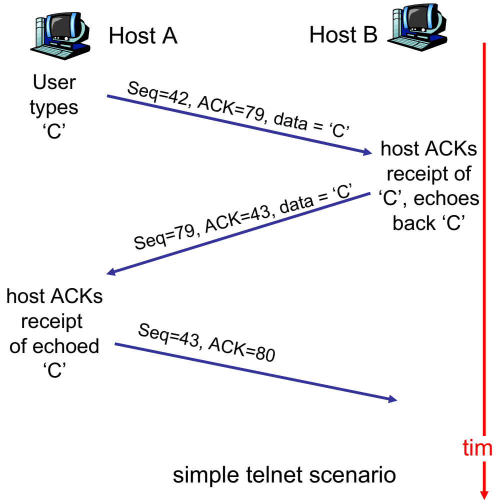
- byte stream "number" of first byte in segment's data

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


Host A
Host B
User types 'C'
Seq=42, ACK=79, data = 'C'
host ACKs receipt of 'C', echoes back 'C'
Seq=79, ACK=43, data = 'C'
host ACKs receipt of echoed 'C'
Seq=43, ACK=80
time

simple telnet scenario

Chapter 3 outline

- Transport-layer services
- Multiplexing and demultiplexing
- Connectionless transport: UDP
- Principles of reliable data transfer
- Connection-oriented transport: TCP
  - segment structure
  - **reliable data transfer**
  - flow control
  - connection management
- Principles of congestion control
- TCP congestion control
- SCTP
- Reliable Multicast

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 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 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?

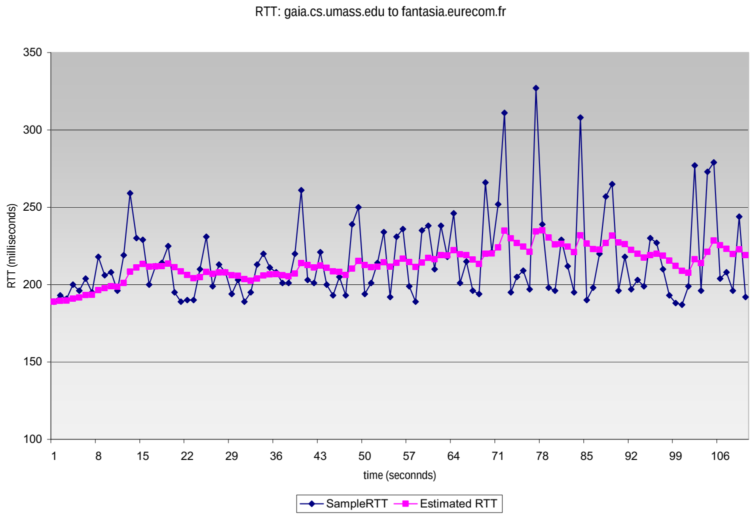
- `SampleRTT`: measured time from segment transmission until ACK receipt
  - ignore retransmissions
- `SampleRTT` will vary, want estimated RTT “smoother”
  - average several recent measurements, not just current `SampleRTT`

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:

```
DevRTT = (1-β)*DevRTT +
             β*|SampleRTT-EstimatedRTT|

(typically, β = 0.25)
```

Then set timeout interval:

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

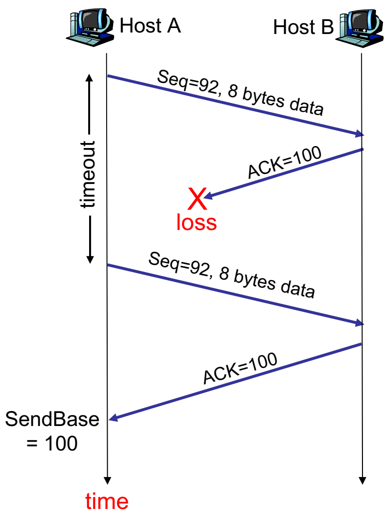
TCP sender (simplified)

```
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 */
```

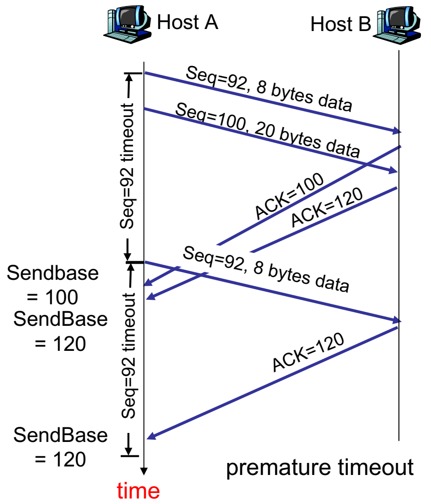
Comment:
- SendBase-1: last cumulatively ack'ed byte

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

TCP: retransmission scenarios

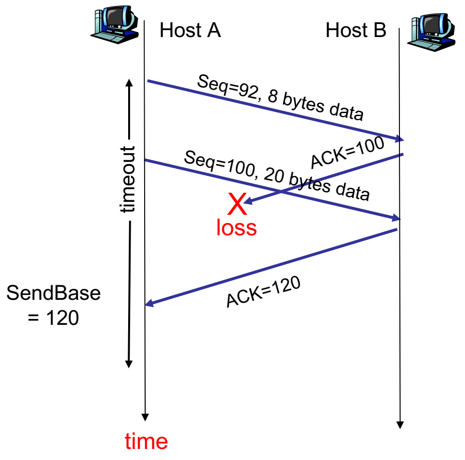


lost ACK scenario



premature timeout

TCP retransmission scenarios (more)



Cumulative ACK scenario

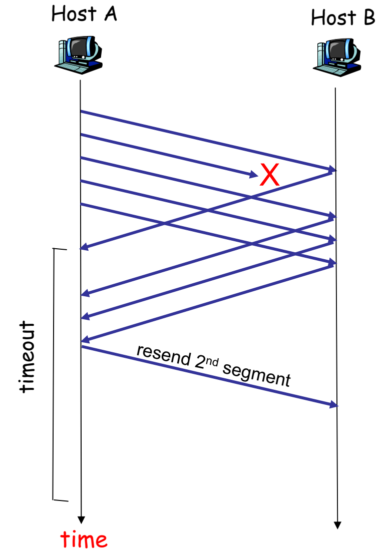
TCP ACK generation [RFC 1122, RFC 2581]

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

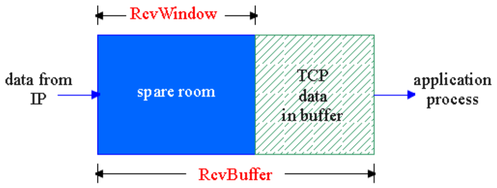
fast retransmit

Chapter 3 outline

- Transport-layer services
- Multiplexing and demultiplexing
- Connectionless transport: UDP
- Principles of reliable data transfer
- Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - **flow control**
  - connection management
- Principles of congestion control
- TCP congestion control
- SCTP
- Reliable Multicast

TCP Flow Control

- receive side of TCP connection has a receive buffer:

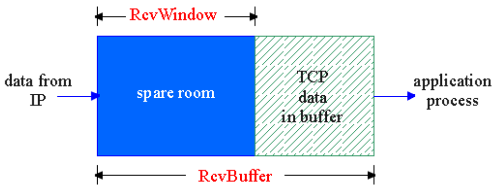


flow control

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

- Transport-layer services
- Multiplexing and demultiplexing
- Connectionless transport: UDP
- Principles of reliable data transfer
- Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - **connection management**
- Principles of congestion control
- TCP congestion control
- SCTP
- Reliable Multicast

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

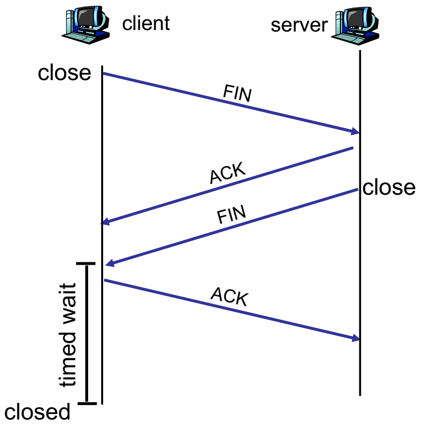
TCP Connection Management (cont.)

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.



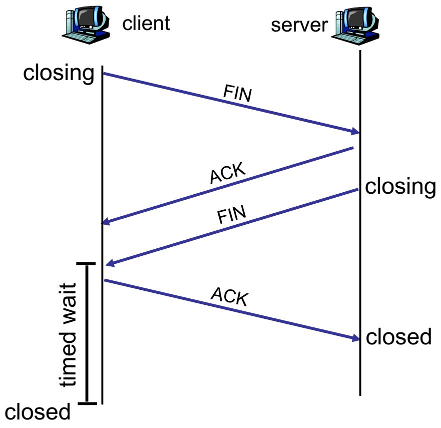
TCP Connection Management (cont.)

Step 3: client receives FIN, replies with ACK.

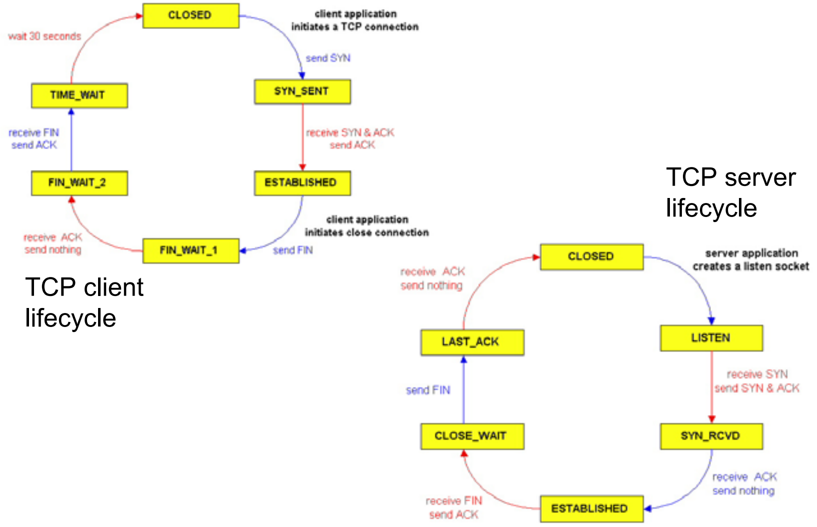
- Enters "timed wait" - will respond with ACK to received FINs

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

- Transport-layer services
- Multiplexing and demultiplexing
- Connectionless transport: UDP
- Principles of reliable data transfer
- Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- **Principles of congestion control**
- TCP congestion control
- SCTP
- Reliable Multicast

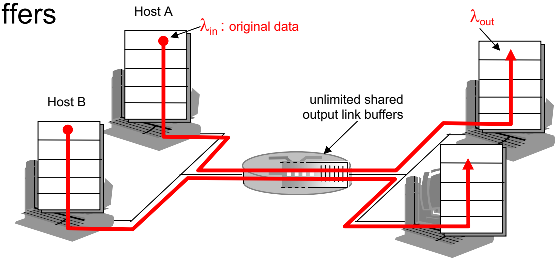
Principles of Congestion Control

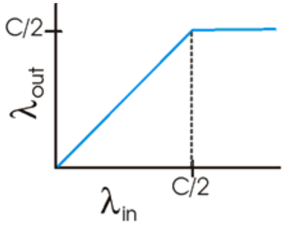
Congestion:

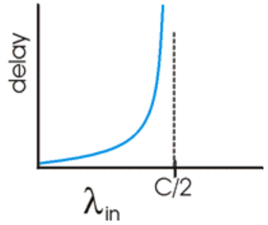
- informally: “too many sources sending too much data too fast for *network* to handle”
- different from flow control!
- manifestations:
  - lost packets (buffer overflow at routers)
  - long delays (queueing in router buffers)
- a top-10 problem!

Causes/costs of congestion: scenario 1

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


Host A
λ_{in} : original data
λ_{out}
Host B
unlimited shared output link buffers

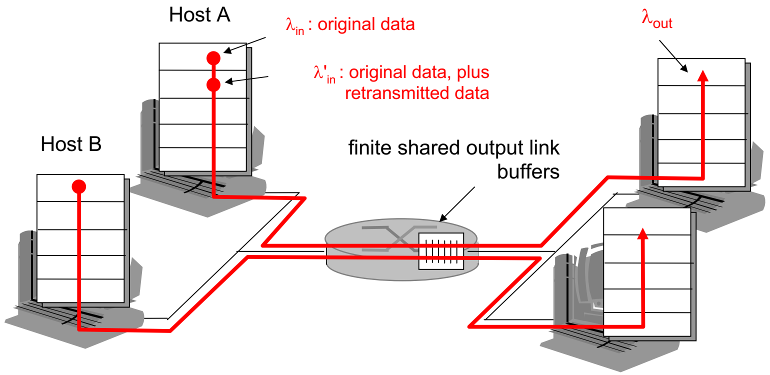




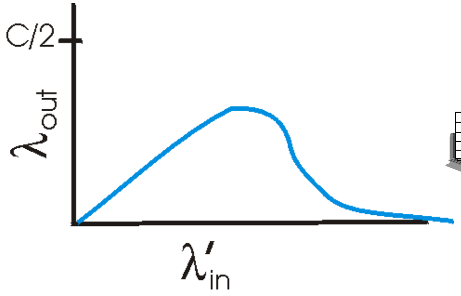
- large delays when congested
- maximum achievable throughput

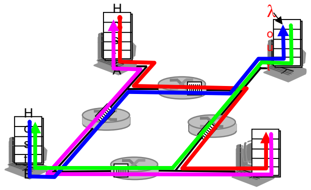
Causes/costs of congestion: scenario 2

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


Host A
λ_{in} : original data
λ'_{in} : original data, plus retransmitted data
λ_{out}
Host B
finite shared output link buffers

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

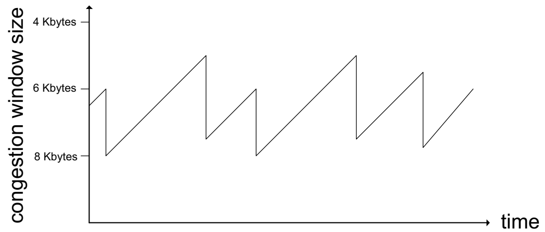
Chapter 3 outline

- ❑ Transport-layer services
- ❑ Multiplexing and demultiplexing
- ❑ Connectionless transport: UDP
- ❑ Principles of reliable data transfer
- ❑ Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- ❑ Principles of congestion control
- ❑ **TCP congestion control**
- ❑ SCTP
- ❑ Reliable Multicast

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:

  the amount of unacked data at sender is limited by CongWin:

  LastByteSent-LastByteAcked ≤ CongWin
- ❑ Roughly:

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

- ❑ **CongWin** is dynamic, function of perceived network congestion
- ❑ **Self-clocking**: sender may send additional segments when acks arrive

How does sender perceive congestion?

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

three mechanisms:

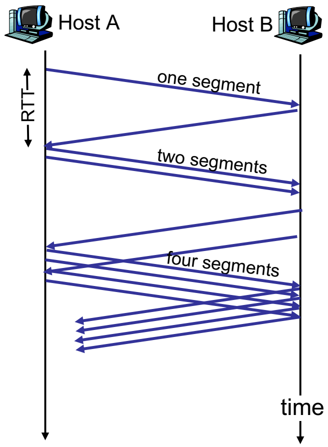
- AIMD
- slow start
- conservative after timeout events

TCP Slow Start

- When connection begins, **`CongWin`** = 1 MSS
  - Example: MSS = 1000 bytes, RTT = 100 msec
  - initial rate ≈ CongWin/RTT = 80 kbit/s
- 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:

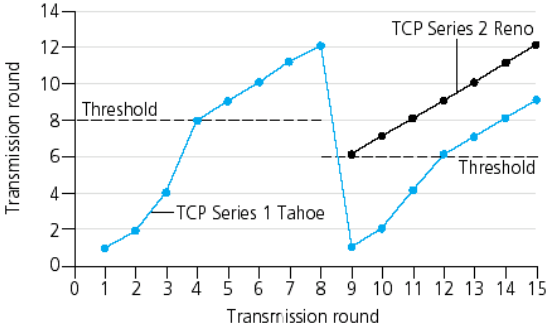
- ❑ 3 dup ACKs indicates network capable of delivering some segments
- ❑ timeout indicates a “more alarming” congestion scenario

Refinement

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

Implementation:

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



- TCP Reno: Fast Recovery after 3 dup Acks

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** (Fast Recovery, TCP Reno)
- When timeout occurs, **Threshold** set to **CongWin/2** and **CongWin** is set to 1 MSS.

TCP sender congestion control

State	Event	TCP Sender Action	Comment
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

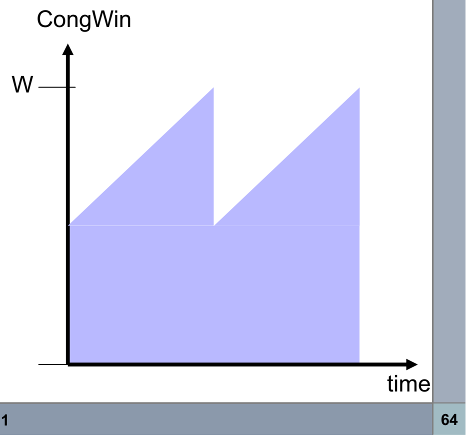
SS or CA	Duplicate ACK	Increment duplicate ACK count for segment being acked	CongWin and Threshold not changed

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: **RcvWin**
  - **RcvWin** advertised by receiver
- Don’t overload network: **CongWin**
  - **CongWin** affected by receiving ACKs
- Sender buffer = min {**RcvWin**, **CongWin** }
- Congestion control:
  - Slow start: exponential growth of **CongWin**
  - Congestion avoidance: linear growth of **CongWin**
  - Timeout; duplicate ACK: shrink **CongWin**
- Continuously adjust RTT estimation

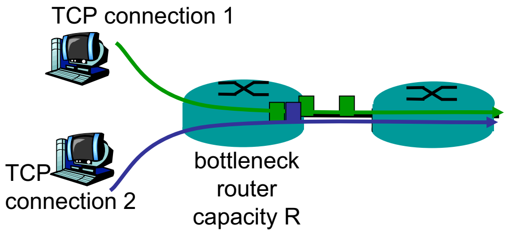
TCP throughput

- What’s the average throughput 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 throughput: 0.75 W/RTT



TCP Fairness

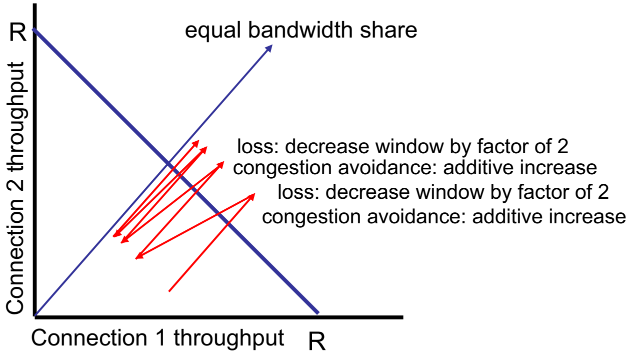
Fairness goal: if K TCP sessions share same bottleneck link of bandwidth R, each should have average rate of R/K



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

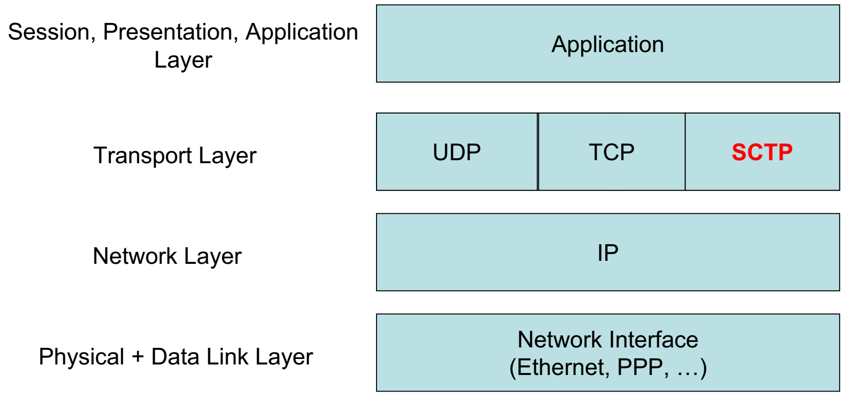
- nothing prevents app from opening parallel connections between 2 hosts.
- Web browsers do this
- Example: link of rate R supporting 9 connections;
  - new app asks for 1 TCP, gets rate R/10
  - new app asks for 11 TCPs, gets R/2 !

Chapter 3 outline

- Transport-layer services
- Multiplexing and demultiplexing
- Connectionless transport: UDP
- Principles of reliable data transfer
- Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- Principles of congestion control
- TCP congestion control
- **SCTP**
- Reliable Multicast

Internet Protocol Stack

- The Internet Protocol Stack



Session, Presentation, Application Layer	Application		
Transport Layer	UDP	TCP	SCTP
Network Layer	IP		
Physical + Data Link Layer	Network Interface (Ethernet, PPP, …)		

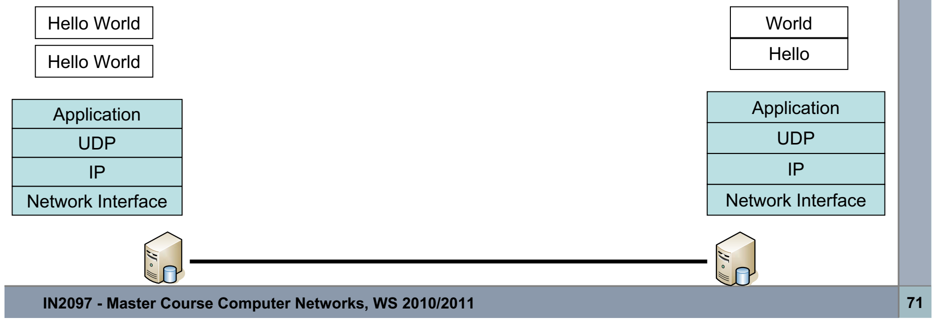
- Why another transport layer protocol?

Contents

- Limitations of UDP and TCP
- The Stream Control Transmission Protocol (SCTP)
  - Association setup / stream setup
  - Message types
  - Partial Reliability
  - Multi-Homing support
  - Congestion control

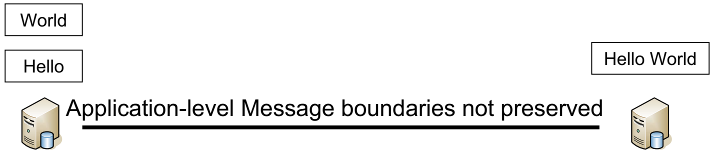
User Datagram Protocol

- Message oriented
  - Sending application writes a N byte message
  - Receiving application reads a N byte message
- Unreliable
  - Lost packets will not be retransmitted
- Unordered delivery
  - Packets may be re-ordered in the network



Transmission Control Protocol

- Connection/Stream oriented. Not message oriented



Application-level Message boundaries not preserved

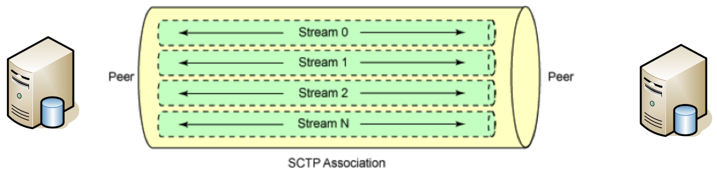
- Reliable transmission
  - Lost packets are retransmitted
  - Retransmission will be repeated until acknowledgment is received
- In-order delivery
  - Segments n + 1, n + 2, n + 3, will be delivered after segment n
- Congestion control
  - TCP tries to share bandwidth equally between all end-points

Problems

- Certain applications pose problems to UDP and TCP
- TCP: Head-of-line blocking with video streaming
  - Frames 2,3,4 arrived but cannot be shown because frame 1 is missing
  - ⇨ Video will stop until frame 1 is delivered
- UDP:
  - Unordered delivery: Second image is delivered after first image
  - Packet loss: Certain frames get lost ⇨ low video/audio quality
  - No congestion control
- Example: Internet-Telephony
  - Two types of traffic:
    - Signalling traffic: should be delivered reliable + in-order (TCP)
    - Voice traffic: should not suffer from head-of-line blocking (UDP)
  - Need to manage two sockets
- SCTP can deal with these problems

SCTP Features at a glance

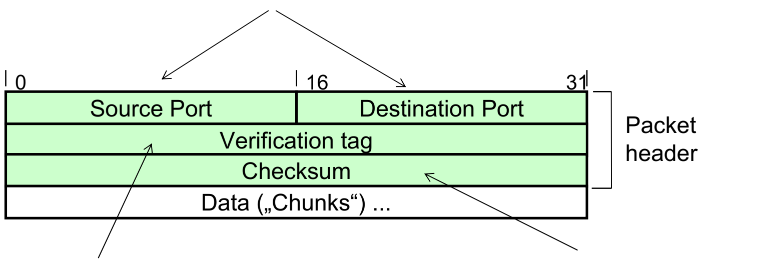
- **Connection and message oriented**
  - SCTP builds an "association" between two peers
  - Association can contain multiple "streams"
  - Messages are sent over one of the streams



- **Partial reliability**
  - "Lifetime" defined for each message
    - Retransmission of a message is performed during its lifetime
  - Messages can be delivered unreliable, full reliable or partial reliable
- **Multi-Homing**
  - SCTP can use multiple IP addresses

SCTP Message Format

- **Common header format**
  - 12 byte header
  - included in every SCTP message



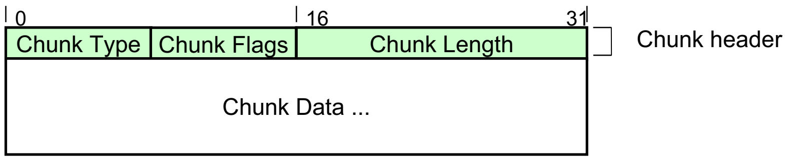
Ports address the application

Random number which
Identifies a given association:
Used to distinguish new from old connections

Checksum on the complete
SCTP message: Common
header and "chunks"

SCTP Chunk Format

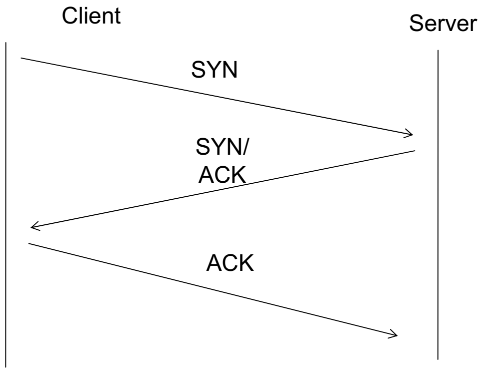
- Data and signaling information is transported in chunks
  - One or more chunks in a SCTP message
  - Each chunk type has a special meaning:
    - INIT, INIT-ACK, COOKIE, COOKIE-ACK ⇨ Connection setup
    - DATA ⇨ Transports user data
    - SACK ⇨ Acknowledge Data
- Common chunk format



- Additional formats are defined for specific chunk types

Connection Setup

- □ **TCP connection setup**



Client

Server

SYN

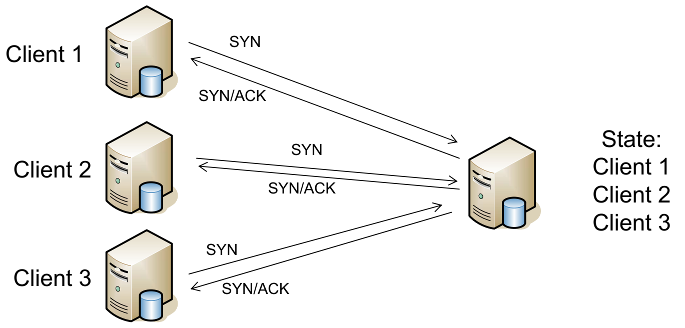
SYN/
ACK

ACK

Create State for
TCP connection: Store
client information

- □ **Known Problem: TCP SYN-Flooding**

SYN Flodding



Client 1

Client 2

Client 3

SYN

SYN/ACK

SYN

SYN/ACK

SYN

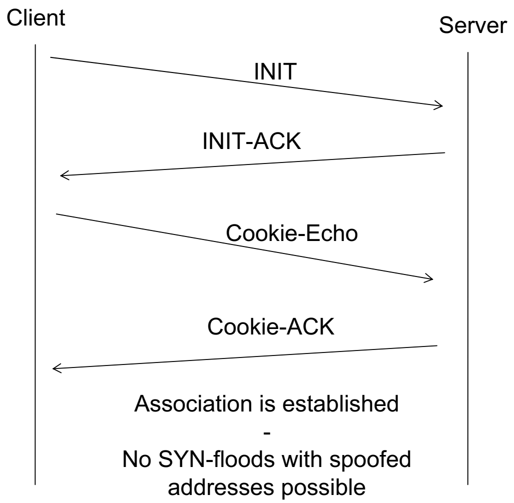
SYN/ACK

State:
Client 1
Client 2
Client 3

- □ Clients send SYN-Packets but do not respond to SYN-ACK
  - ▪ Usually done by a single client that performs IP address spoofing
  - ▪ Works because only a single forged packet is necessary
- ⇨ Server has to store state until a TCP timeout occurs
  - ▪ Leads to resource exhaustion
    - ⇨ Server cannot accept any more connections

SCTP Association Setup

- □ **Solution to SYN-Flood problem: Cookies**



Client

Server

INIT

Generate client specific cookie

INIT-ACK

Send cookie ⇨ forget client

Cookie-Echo

Check if cookie is valid ⇨
Create state only on valid
cookie

Cookie-ACK

Association is established
-
No SYN-floods with spoofed
addresses possible

Data Transmission

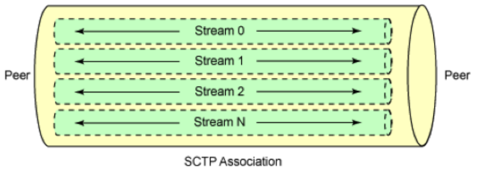
- □ Application data is transmitted in Data Chunks
  - ▪ A data chunk is associated to a stream (Stream Identifier S)

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 0    | Reserved|U|B|E|    Length                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              TSN                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Stream Identifier S      |   Stream Sequence Number n    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Payload Protocol Identifier                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                                                               /
/                 User Data (seq n of Stream S)                 /
/                                                               /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- □ TSN (Transport Sequence Number)
  - ▪ Global Sequence Number
  - ▪ Similar to TCP sequence number, used for retransmissions
- □ Stream sequence number
  - ▪ Necessary for per-stream transmission reliability

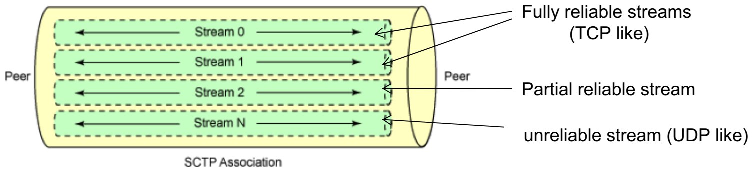
Transmission reliability (1)

- TCP
  - Packets are transmitted fully reliable ⇨ retransmitted until received
  - Packets are delivered in-order to the application
  - Slow start and congestion avoidance for congestion control
- UDP
  - Packets are transmitted fully unreliable ⇨ never retransmitted
  - No re-ordering ⇨ packet order may be changed at the receiver
  - No congestion control
- SCTP can do both and more, in a stream-specific way



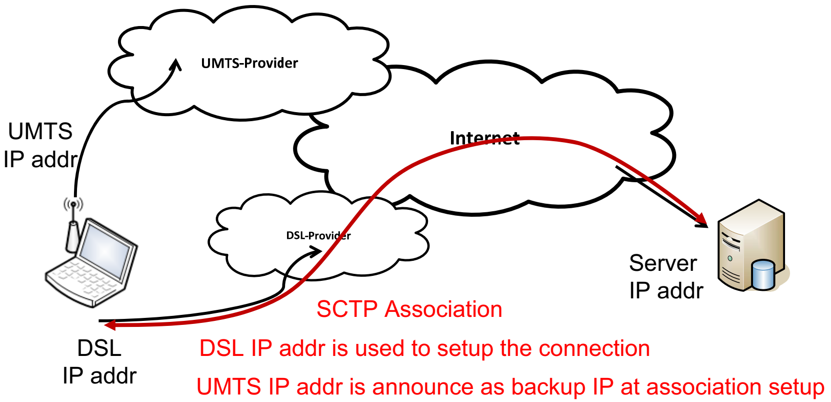
Transmission reliability (2)

- Why multiple streams?
  - Solves head of line blocking
  - No firewall issues (only one port for several streams)
  - Partial Reliability Extension (PR-SCTP) for different reliability levels
- PR-SCTP
  - Allows to set a lifetime parameter for each stream
  - Lifetime specifies how long the sender should try to retransmit a packet
  - Allows to mix reliable and unreliable streams



Multi-Homing: Association setup

- SCTP chooses one IP address at association setup
  - IP address can be specified by user

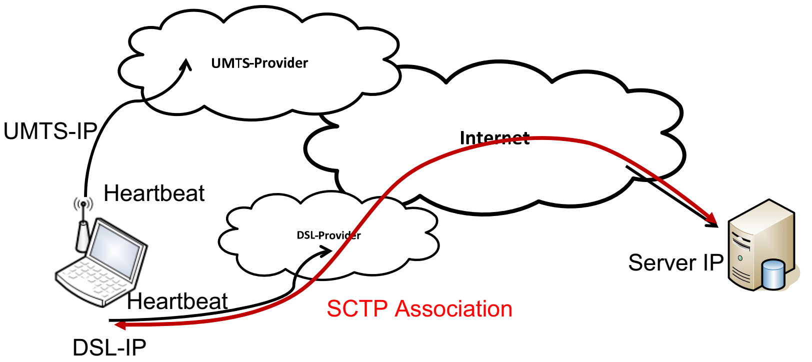


DSL IP addr is used to setup the connection

UMTS IP addr is announce as backup IP at association setup

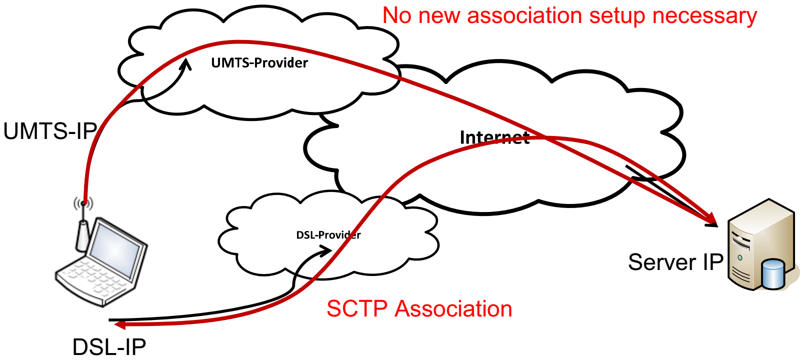
Multi-Homing

- Heartbeat messages are periodically sent to check link availability



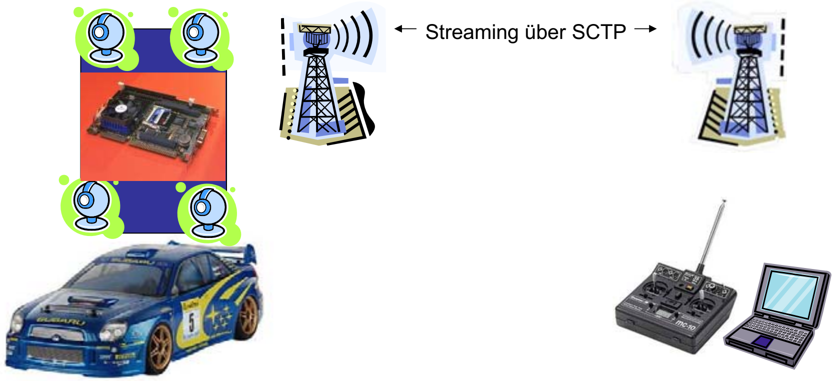
Multi-Homing

- Changes occur when the default link is found to be broken
  - Is identified because of packet loss (data or heartbeat)
  - Consequence: SCTP will resume on the backup link



SCTP Example Scenario

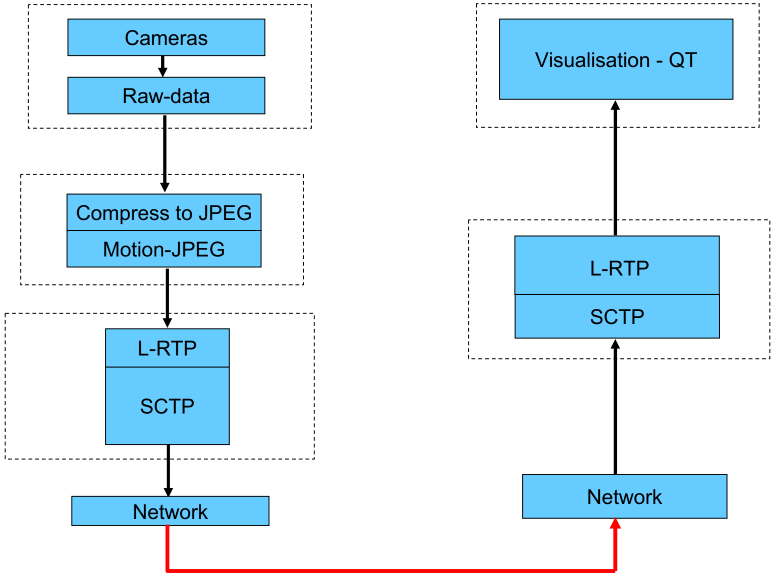
- Real-time transmission of video streams and control data in vehicular scenario



Server: vehicle with embedded PC (Linux)

Client: Unix/Windows

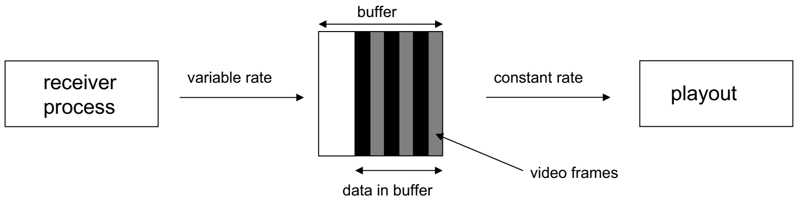
Protocol Architecture



Leightweight - RTP

L-RTP implementation:

- Timestamping for synchronisation
- Packet loss detection
- Buffering



SCTP Deployment

- SCTP has attractive features
  - but to which extent is it used?
- Why do we use HTTP over TCP for Video Streaming?
- Why is IP Multicast not generally deployed?
  - Because HTTP over TCP streaming just works „good enough“
- Firewall and NAT issues
  - Most home routers simply can‘t translate SCTP
- Implementations
  - Currently no native Windows support (only userspace lib)
- BUT: mandatory for some newly developed protocols such as IPFIX (IP Flow Information Export)

Chapter 3 outline

- Transport-layer services
- Multiplexing and demultiplexing
- Connectionless transport: UDP
- Principles of reliable data transfer
- Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- Principles of congestion control
- TCP congestion control
- SCTP
- **Reliable Multicast**

Many Uses of Multicasting

- Teleconferencing
- Distributed Games
- Software/File Distribution
- Video Distribution
- Replicated Database Updates

⇨ multicast transport is done differently for each application

Multicast Application Modes

- Point-to-Multipoint:
  Single Source, Multiple Receivers
- Multipoint-to-Multipoint:
  Multiple Sources, Multiple Receivers
- Sources are receivers
- Sources are not receivers

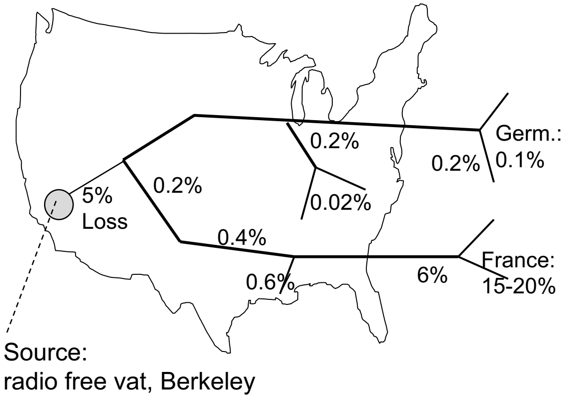
Classification of Multicast Applications

Transport service type	***Fully reliable multicast***	***Real-time multicast***
Single source: ***1:N***	Multicast-FTP; Software update	Audio-visual conference; Continuous Media Dissemination
Multiple Sources ***M:N***	CSCW; Distributed computing	DIS; VR

- CSCW: Computer Supported Cooperative Work
- DIS: Distributed Interactive Simulation
- VR: Virtual Reality

Where Does Multicast Loss Occur

❑ Example measurements
(April 96, Yajnik, Kurose, Towsely, Univ. Mass., Amherst)



Simultaneous Packet Loss

❑ Q: distribution of number of receivers losing packet?

❑ Example dataset:
47% packets lost somewhere
5% shared loss

❑ Similar results across different datasets

❑ Models of packet loss (for protocol design, simulation, analysis):
- star: end-end loss independent
- full topology: measured per link loss independent
- modified star: source-to-backbone plus star
⇨ good fit for example data set

Temporal Loss Correlation

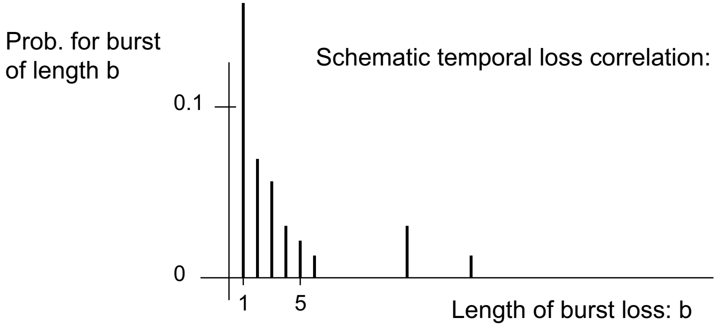
Q: do losses occur singly or in “bursts”?

❑ occasional long periods of 100% loss

❑ generally isolated losses

❑ occasional longer bursts

Schematic temporal loss correlation:



Reliable Multicast Challenge

- How to transfer data reliably from source to R receivers
- scalability: 10s - 100s - 1000s - 10000s - 100000s of receivers
- heterogeneity
  - different capabilities of receivers (processing power, buffer, protocol capabilities)
  - different network conditions for receivers (bottleneck bandwidths, loss rates, delay)
- feedback implosion problem

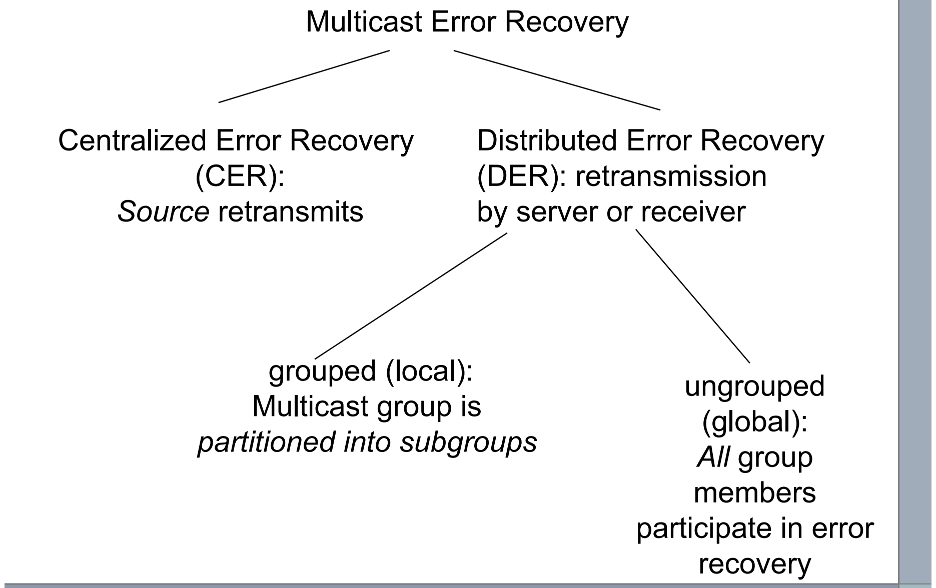
ARQ: Alternatives for Basic Mechanisms

- Who retransmits
  - source
  - network
  - other group member.
- Who detects loss
  - sender based: waiting for all ACKs
  - receiver based: NACK, more receivers - faster loss detection.
- How to retransmit
  - Unicast
  - Multicast
  - Subgroup-multicast

Approaches

- shift responsibilities to receivers (in contrast to TCP: sender is responsible for large share of functionality)
- feedback suppression (some feedback is usually required)
- multiple multicast groups (e.g. for heterogeneity problems; can be used statically or dynamically)
- local recovery (can be used to reduce resource cost and latency)
- server-based recovery
- forward error correction (FEC)
  - FEC for unicast: frequently no particular gain
  - FEC for multicast: gain may be tremendous!

Classification of Multicast Error Control



Multicast Error Recovery

Centralized Error Recovery (CER): *Source* retransmits

Distributed Error Recovery (DER): retransmission by server or receiver

grouped (local): Multicast group is *partitioned into subgroups*

ungrouped (global): *All* group members participate in error recovery

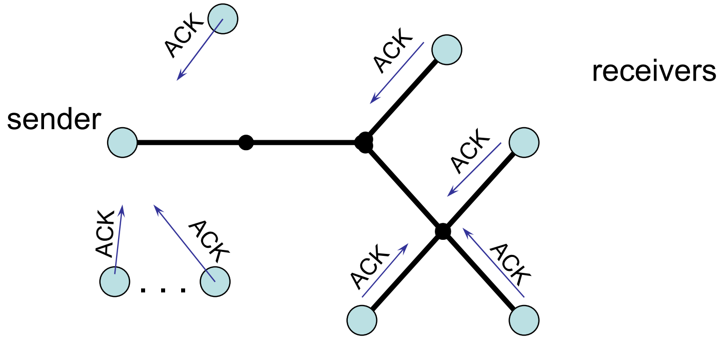
Reliable Multicast: Building Blocks

- Elements from Unicast:
  - Loss detection
    - Sender-based (ACK): 1 ACK per receiver and per packet; Sender needs a table of per-receiver ACK
    - Receiver-based (NAK): distributed over receivers; potentially only 1 NAK per lost packet
  - Loss recovery: ARQ vs. FEC
- Additional new Elements for Multicast:
  - Mechanisms for control message **Implosion Avoidance**
  - Mechanisms to deal with *heterogeneous receivers*

Feedback Processing

- Assume: R Receivers, independent packet loss probability p
- Calculate feedback per packet:
  - average number of ACKs: R - pR
  - average number of NAKs: pR
  - ⇨ more ACKs than NAKs
- Processing: higher throughput for receiver-based loss detection
- Reliability needs ACKs
  (No NAK does not mean successful reception)
  - ⇨ use NAK for loss signalling
  - ⇨ use ACKs at low frequency to ensure reliability

Multicast Challenge: Feedback Implosion Problem



NAK Implosion

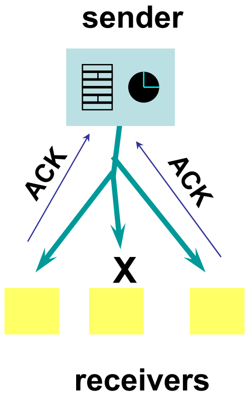
- Shared loss: All receivers loose same packet: All send NAK
  ⇨ NAK implosion
- Implosion avoidance techniques
  - Cluster/Hierarchy
  - Token
  - Timers

  For redundant feedback additionally:
  - Feedback suppression (e.g. multicast NAKs, receiver back off randomly)

  Drawback of implosion avoidance techniques : delay
- Fast NAKs (risk of NAK implosion):
  - Fast retransmission
  - Smaller sender/receiver buffer

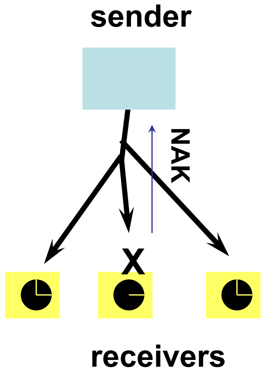
Sender Oriented Reliable Multicast

- Sender:
  multicasts all (re)transmissions
  - selective repeat
  - use of timeouts for loss detection
  - ACK table
- receiver: ACKs received packets
- Note: group membership important
- Example (historic):
  Xpress Transport Protocol (XTP)
  - extension of unicast protocol



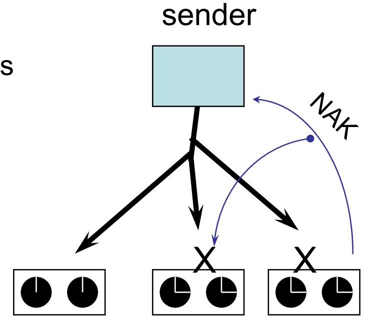
Receiver Oriented Reliable Multicast

- Sender: multicasts (re)transmissions
  - selective repeat
  - responds to NAKs
- Receiver: upon detecting packet loss
  - sends pt-pt NAK
  - timers to detect lost retransmission
- Note: easy to allow joins/leaves



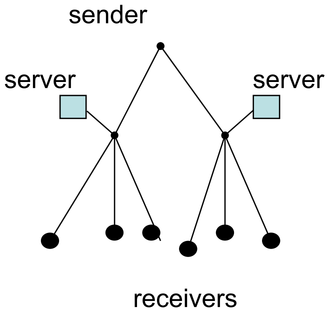
Feedback Suppression

- randomly delay NAKs
- multicast to all receivers
  + reduce bandwidth
  - additional complexity at receivers (timers, etc)
  - increase latencies (timers)
- similar to CSMA/CD (⇨ later)



Server-based Reliable Multicast

- first transmisions: multicast to all receivers and servers
- each receiver assigned to server
- servers perform loss recovery
- servers can be subset of receivers or provided by network
- can have more than 2 levels



Assessment:

- clear performance benefits
- how to configure
  - static/dynamic
  - many-many

Local Recovery

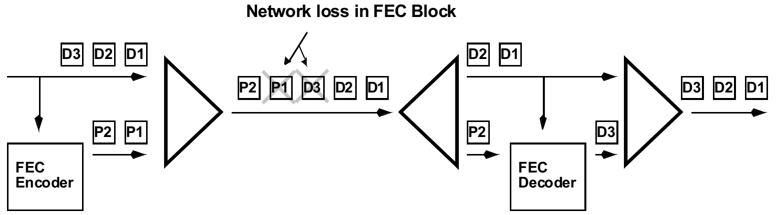
- lost packets recovered from nearby receivers
- deterministic methods
  - impose tree structure on receivers with sender as root
  - receiver goes to upstream node on tree
- self-organizing methods
  - receivers elect nearby receiver to act as retransmitter
- hybrid methods

Issues with Server- and Local Based Recovery

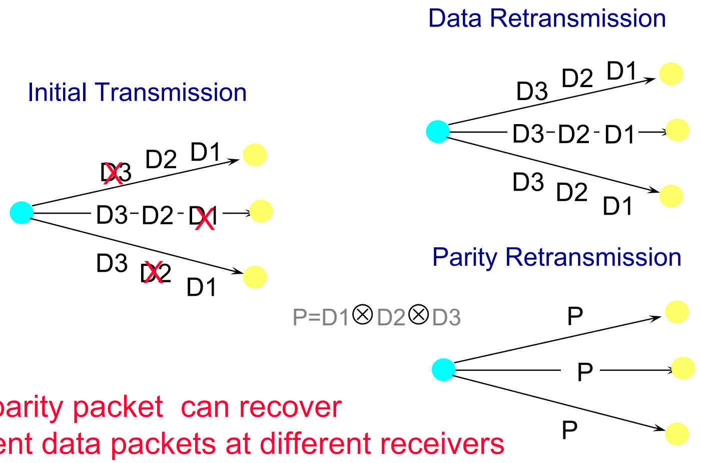
- how to configure tree
- what constitutes a local group
- how to permit joins/leaves
- how to adapt to time-varying network conditions

Forward Error Correction (FEC)

- k original data packets form a **Transmission Group (TG)**
- h parity packets derived from the k data packets
- any k received out of k+h are sufficient
- Assessment
  - + allows to recover lost packets
  - - overhead at end-hosts
  - - increased network load may increase loss probability



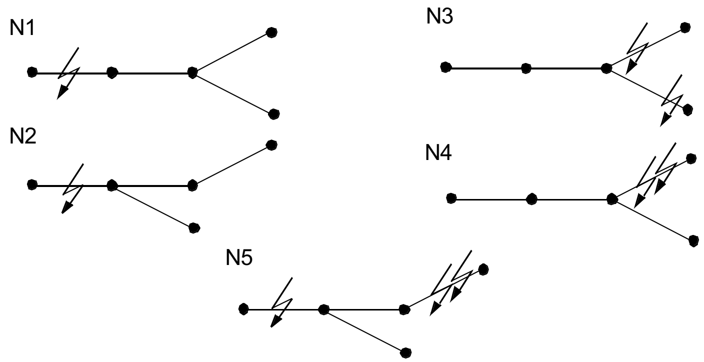
Potential Benefits of FEC



One parity packet can recover different data packets at different receivers

Influence of topology: Selected Scenarios for Modeling Heterogeneity

- Loss: on shared links / on individual links
- Loss: homogeneous/heterogeneous probability
- RTT: homogeneous/heterogeneous.



Scenario-specific Selection of Mechanisms

- FEC is of particular benefit in the following scenarios:
  - Large groups
  - No feedback
  - Heterogeneous RTTs
  - Limited buffer.
- ARQ is of particular benefit in the following scenarios:
  - Herterogeneous loss
  - Loss in shared links of multicast tree dominates
  - Small groups (Statistic by AT&T: on average < 7 participants in conference)
  - Non-interactive applications.
- ARQ by local recovery:
  - large groups (good for individual losses, heterogeneous RTT).

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
  - SCTP
  - Reliable multicast protocols

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

Technische Universität München

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

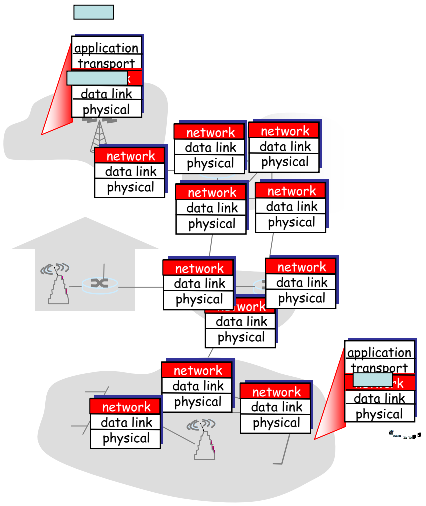
- IPv6
- Virtual circuit and datagram networks
- What's inside a router
- NAT

Part 3

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

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



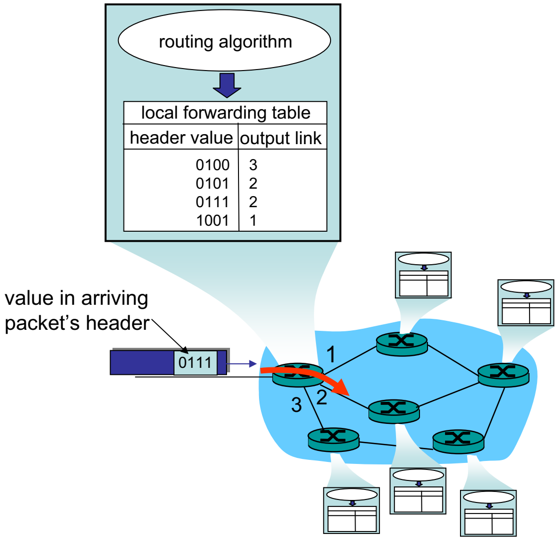
Two Key Network-Layer Functions

- *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

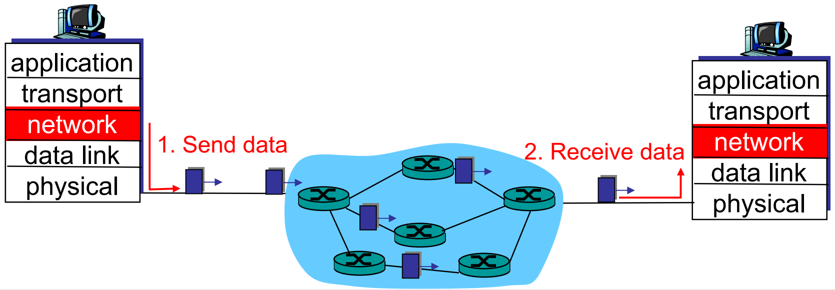
Interplay between routing and forwarding



local forwarding table	
header value	output link
0100	3
0101	2
0111	2
1001	1

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?

Chapter 4: Network Layer

Part 1

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

Part 2

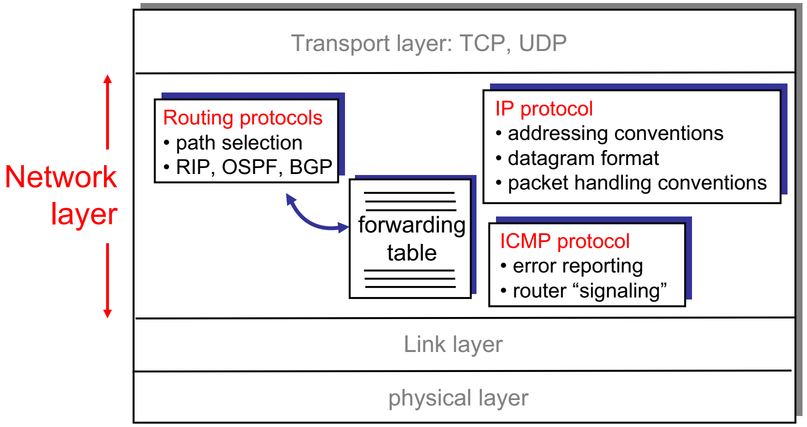
- ❑ IPv6
- ❑ Virtual circuit and datagram networks
- ❑ What's inside a router
- ❑ NAT

Part 3

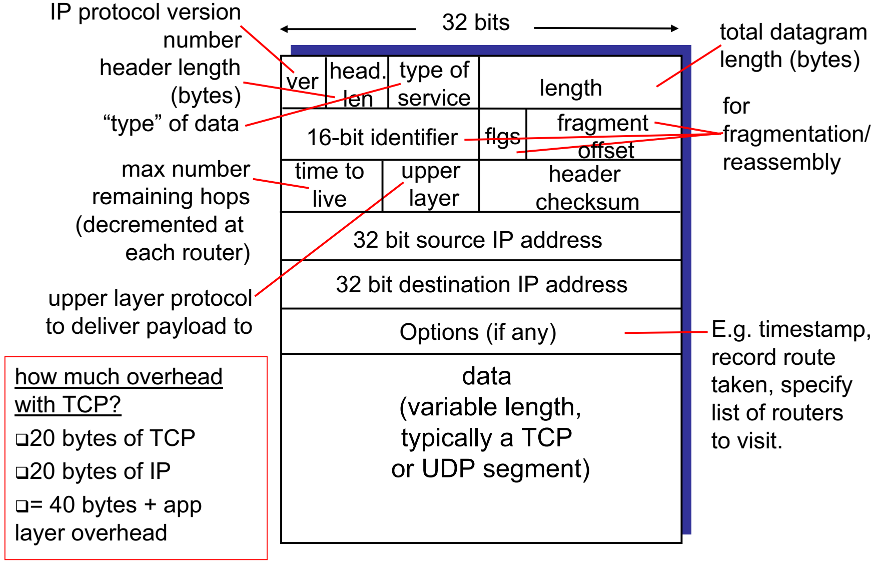
- ❑ Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- ❑ Routing in the Internet
  - RIP
  - OSPF
  - BGP
- ❑ Broadcast and multicast routing

The Internet Network layer

Host, router network layer functions:



IP datagram format

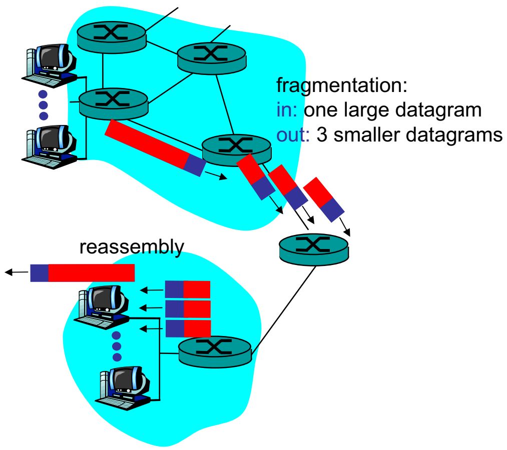

IP protocol version number
header length (bytes)
"type" of data
max number remaining hops (decremented at each router)
upper layer protocol to deliver payload to
32 bits
ver
head. len
type of service
length
16-bit identifier
flgs
fragment offset
time to live
upper layer
header checksum
32 bit source IP address
32 bit destination IP address
Options (if any)
data (variable length, typically a TCP or UDP segment)
total datagram length (bytes)
for fragmentation/ reassembly
E.g. timestamp, record route taken, specify list of routers to visit.

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

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

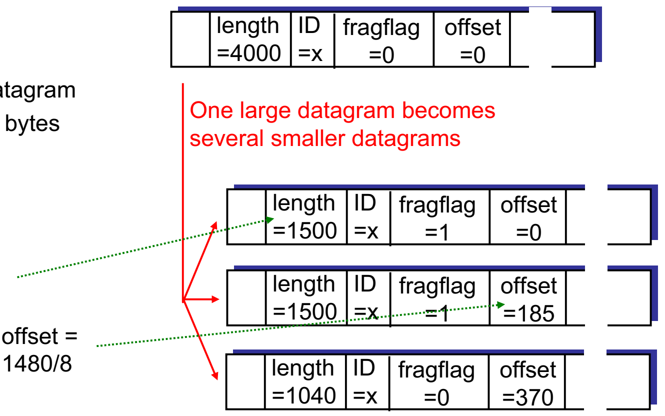
IP Fragmentation & Reassembly

- ❑ 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

Example
- ❑ 4000 byte datagram
- ❑ MTU = 1500 bytes



length =4000 | ID =x | fragflag =0 | offset =0

One large datagram becomes several smaller datagrams

length =1500 | ID =x | fragflag =1 | offset =0

length =1500 | ID =x | fragflag =1 | offset =185

length =1040 | ID =x | fragflag =0 | offset =370

1480 bytes in data field

offset = 1480/8

Chapter 4: Network Layer

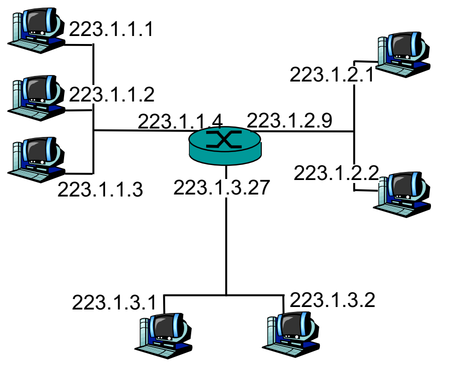
Part 1
- ❑ Introduction
- ❑ **IP: Internet Protocol**
  - Datagram format
  - **IPv4 addressing**
  - ICMP

Part 2
- ❑ IPv6
- ❑ 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

IP Addressing: introduction

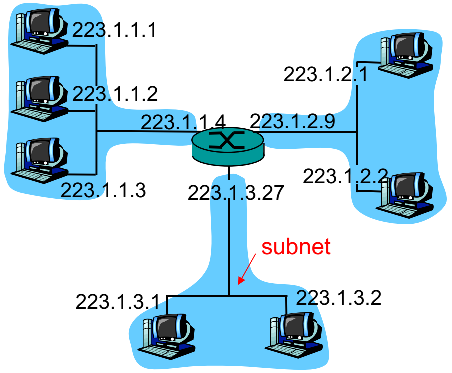
- ❑ IP address: 32-bit identifier for host, router *interface*
- ❑ *interface:* connection between host/router and physical link
  - router's typically have multiple interfaces
  - host typically has one interface
  - IP addresses associated with each interface



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

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

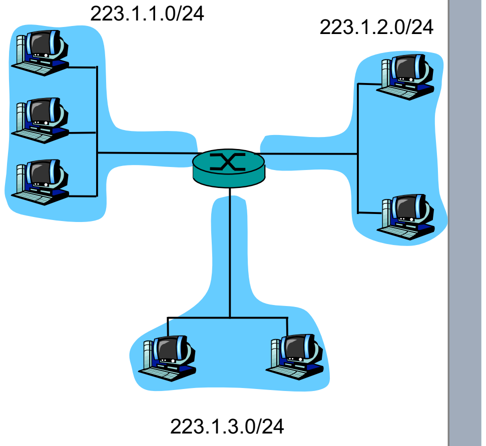


network consisting of 3 subnets

Subnets

Recipe

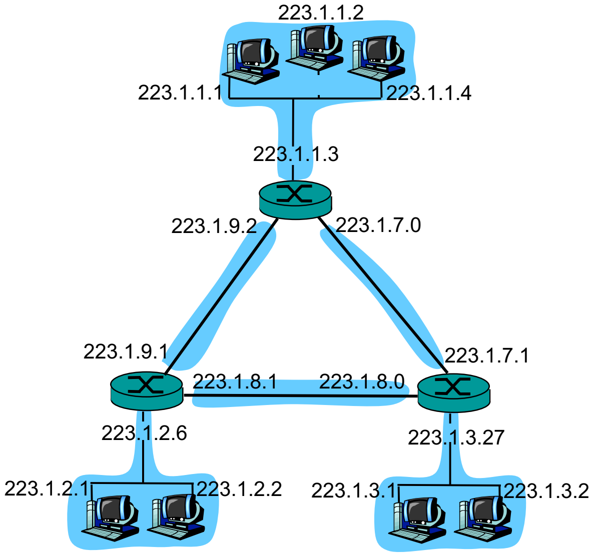
- To determine the subnets, detach each interface from its host or router, creating islands of isolated networks. Each isolated network is called a subnet.



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?

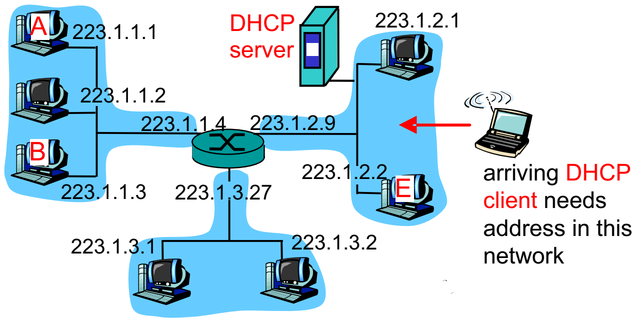
Q: How does a *host* get IP address?

- 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”

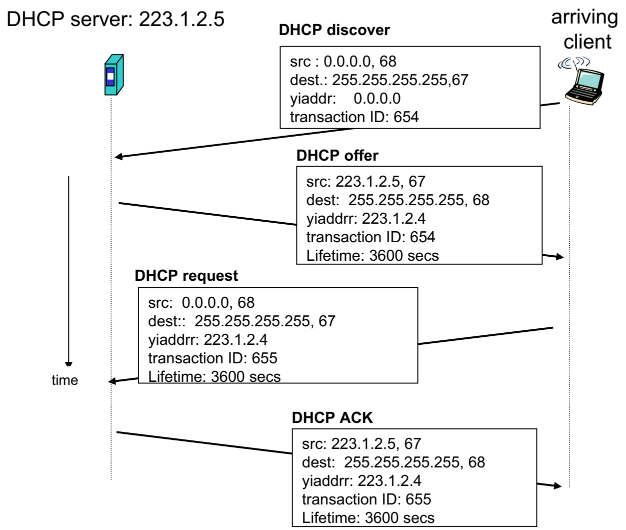
DHCP: Dynamic Host Configuration Protocol

- 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



DHCP discover

src : 0.0.0.0, 68
dest.: 255.255.255.255,67
yiaddr: 0.0.0.0
transaction ID: 654

DHCP offer

src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddrr: 223.1.2.4
transaction ID: 654
Lifetime: 3600 secs

DHCP request

src: 0.0.0.0, 68
dest:: 255.255.255.255, 67
yiaddrr: 223.1.2.4
transaction ID: 655
Lifetime: 3600 secs

DHCP ACK

src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddrr: 223.1.2.4
transaction ID: 655
Lifetime: 3600 secs

IP addresses: how to get one?

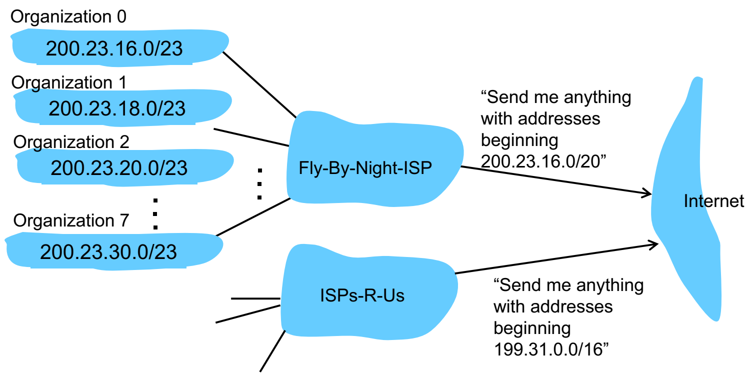
Q: How does *network* get subnet part of IP addr?

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:

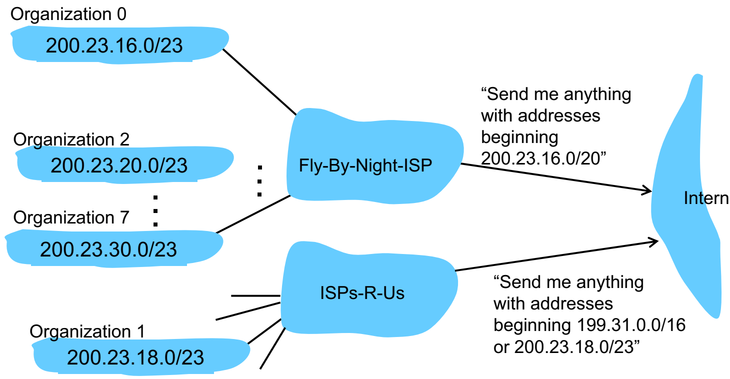


“Send me anything with addresses beginning 200.23.16.0/20”

“Send me anything with addresses beginning 199.31.0.0/16”

Hierarchical addressing: more specific routes

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



“Send me anything with addresses beginning 200.23.16.0/20”

“Send me anything with addresses beginning 199.31.0.0/16 or 200.23.18.0/23”

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

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

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 "dest port unreachable" packet (type 3, code 3)
- When source gets this ICMP, stops.

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

IPv6

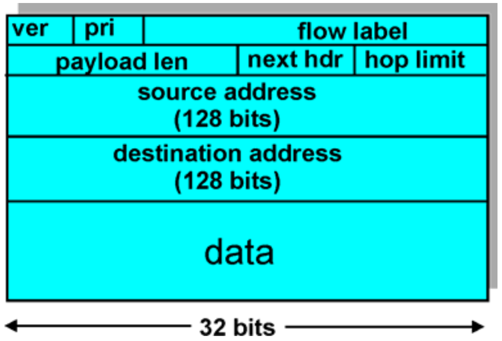
- Initial motivation: 32-bit address space soon to be completely allocated.
- Additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

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



ver | pri | flow label

payload len | next hdr | hop limit

source address (128 bits)

destination address (128 bits)

data

32 bits

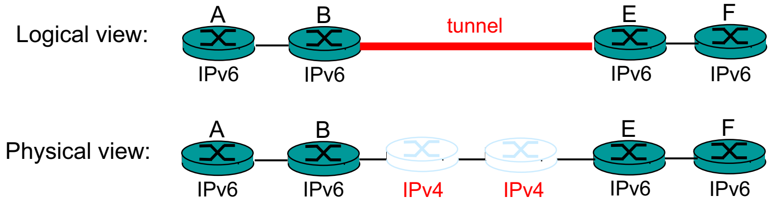
Other Changes from IPv4

- *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

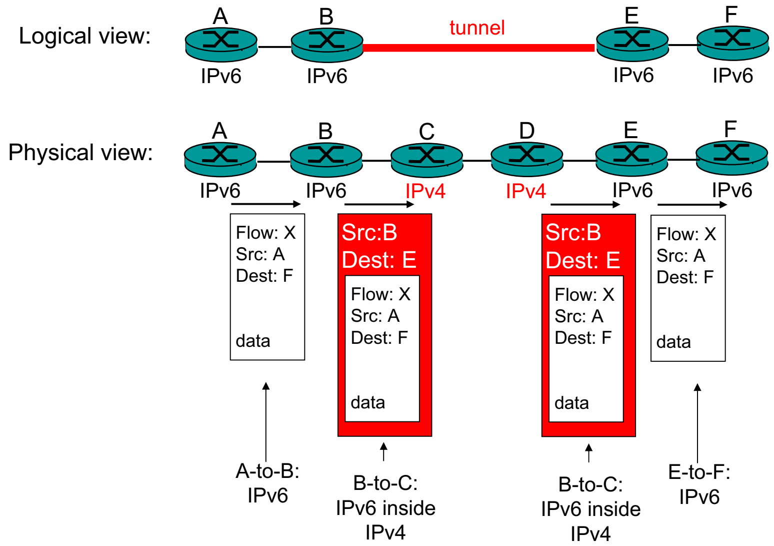
Transition From IPv4 To IPv6

- 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



Tunneling



Logical view:

A B E F

IPv6 IPv6 tunnel IPv6 IPv6

Physical view:

A B C D E F

IPv6 IPv6 IPv4 IPv4 IPv6 IPv6

Flow: X
Src: A
Dest: F

data

A-to-B: IPv6

Src:B
Dest: E

Flow: X
Src: A
Dest: F

data

B-to-C: IPv6 inside IPv4

Src:B
Dest: E

Flow: X
Src: A
Dest: F

data

B-to-C: IPv6 inside IPv4

Flow: X
Src: A
Dest: F

data

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?

Example services for individual datagrams:

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

Example services for a flow of datagrams:

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

Network layer service models

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

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)

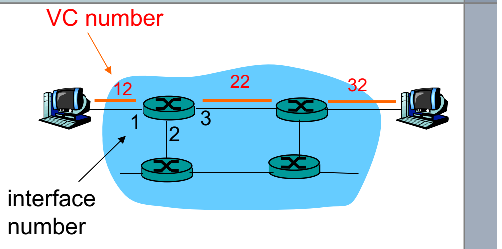
VC implementation

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



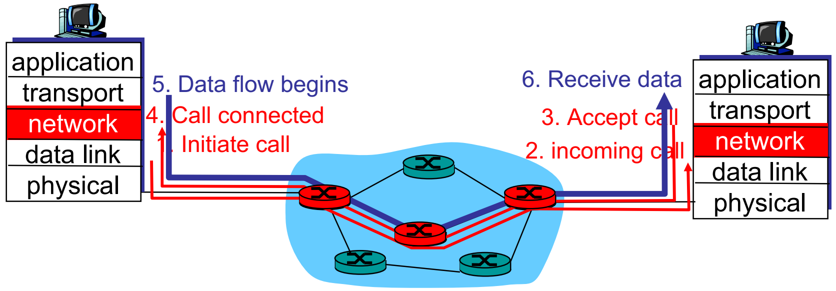
Forwarding table in northwest router:

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!

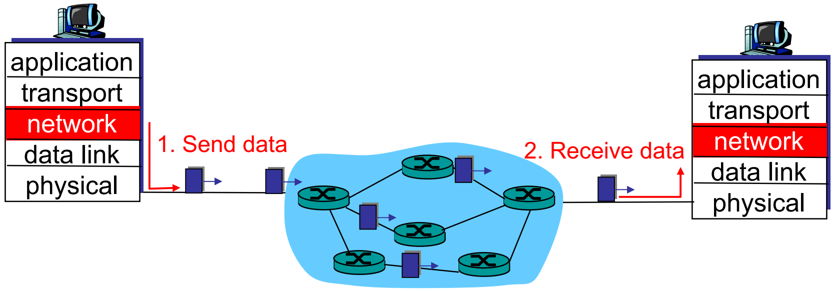
Virtual circuits: signaling protocols

- 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

Chapter 4: Network Layer

Part 1

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

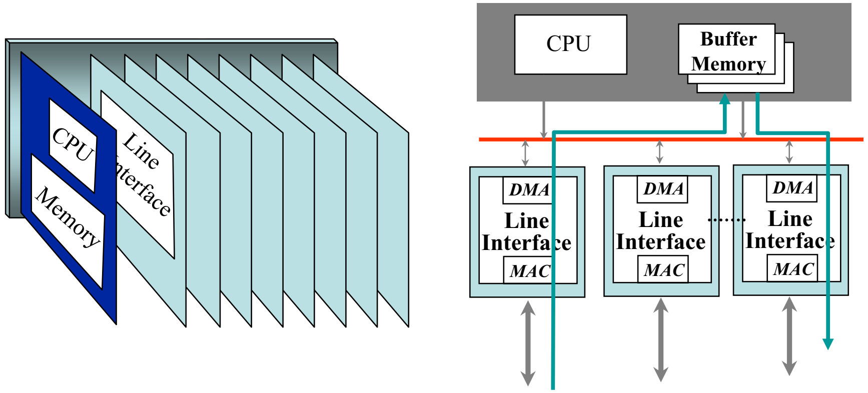
Part 2

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

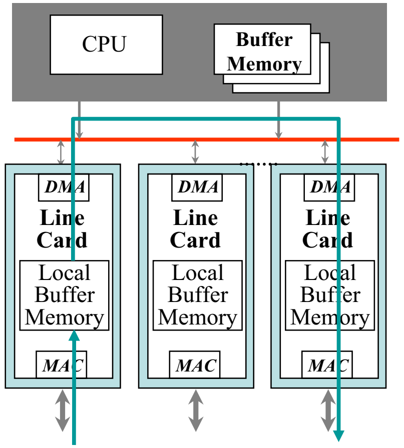
Part 3

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

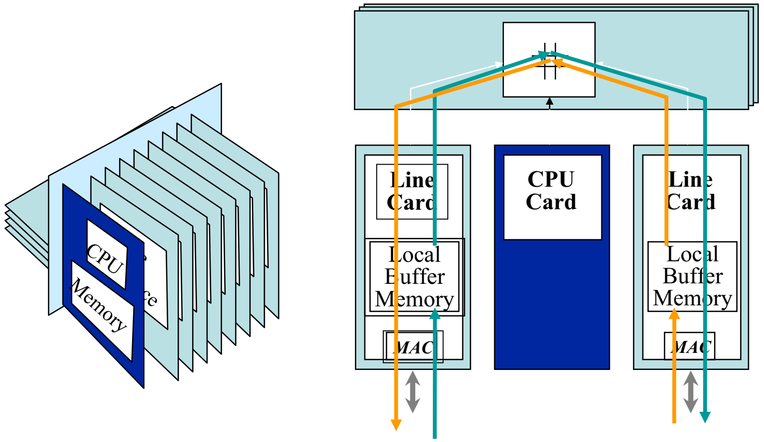
First-Generation IP Routers



Second-Generation IP Routers

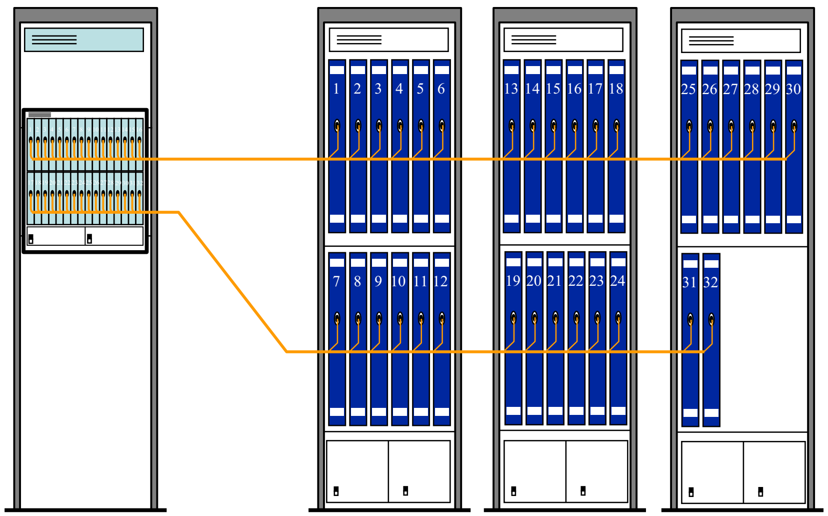


Third-Generation Switches/Routers



Fourth-Generation Switches/Routers

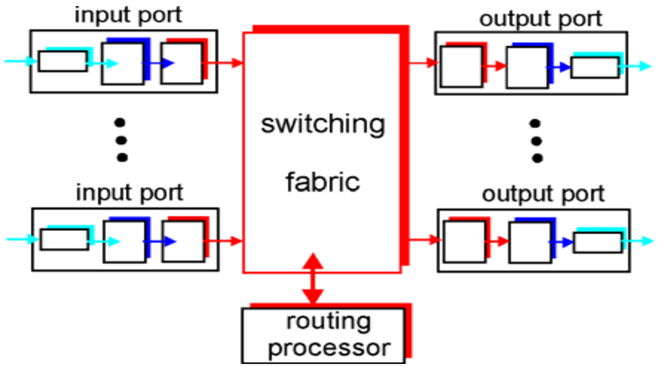
Clustering and Multistage



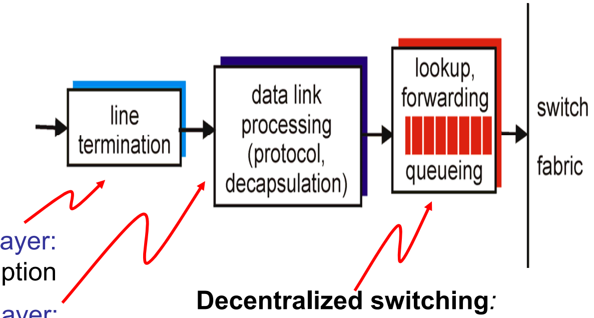
Router Architecture Overview

Two key router functions:

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



Input Port Functions



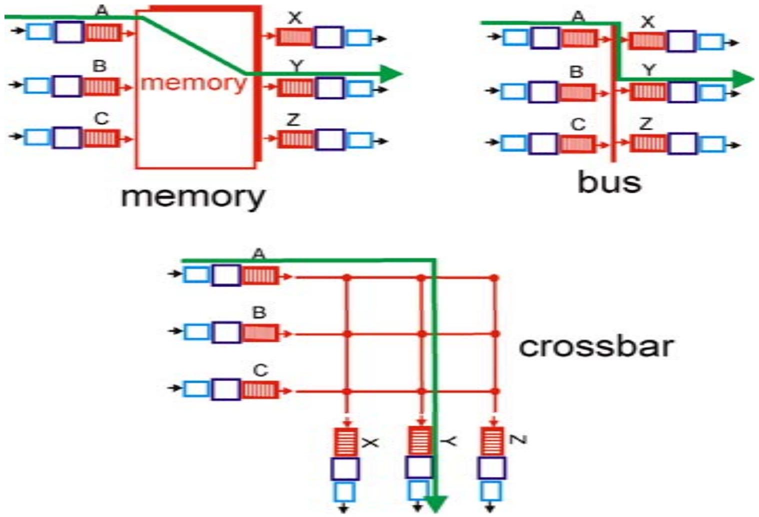
Physical layer:
bit-level reception

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

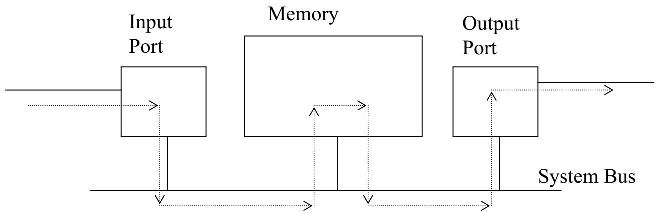
Three types of switching fabrics



Switching Via Memory

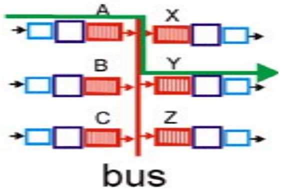
First generation routers:

- 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

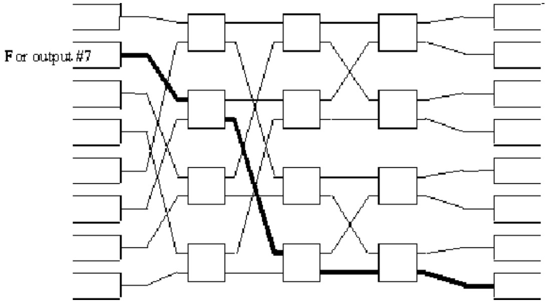
- datagram from input port memory to output port memory via a shared bus
- bus contention: switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers



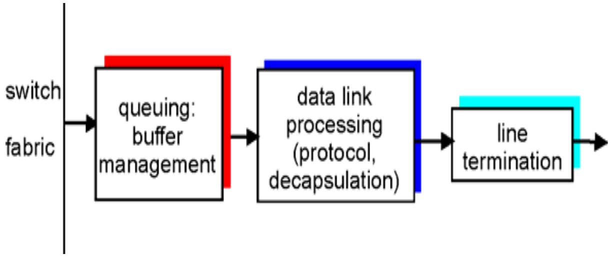
Switching Via An Interconnection Network

- 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:

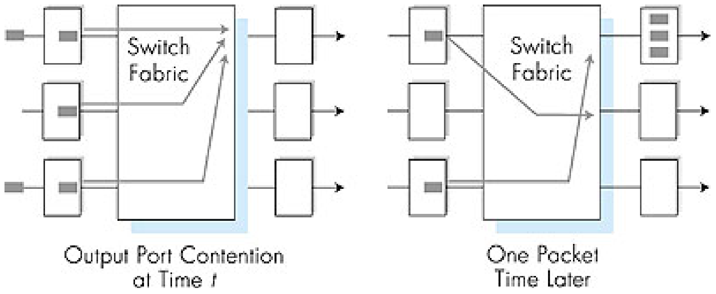


Output Ports



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

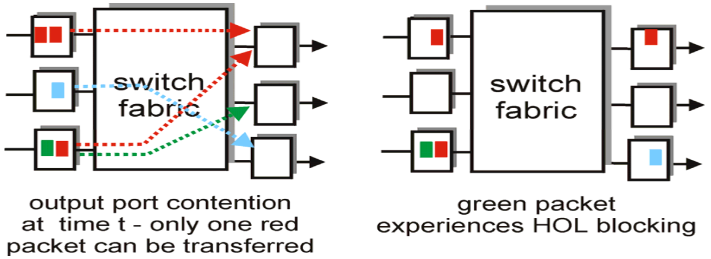
Output port queueing



- 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!*


switch fabric
output port contention at time t - only one red packet can be transferred
switch fabric
green packet experiences HOL blocking

Chapter 4: Network Layer

Part 1

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

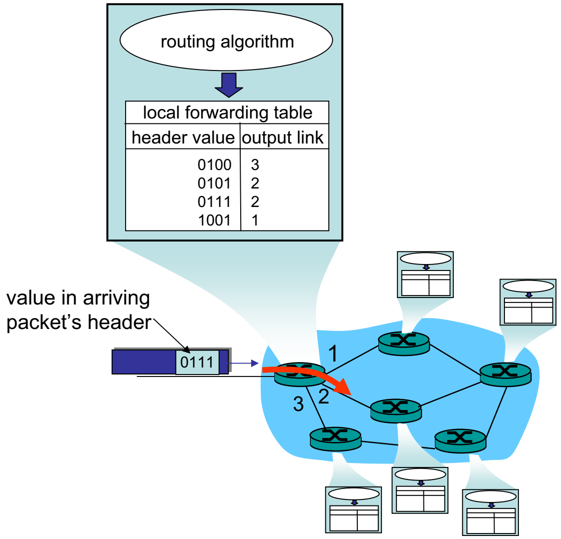
Part 2

- 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

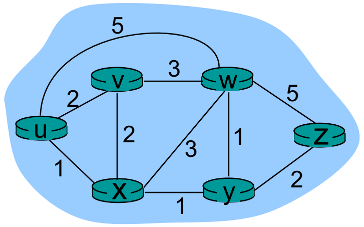
Interplay between routing, forwarding



local forwarding table

header value	output link
0100	3
0101	2
0111	2
1001	1

Graph abstraction



Graph: G = (N,E)

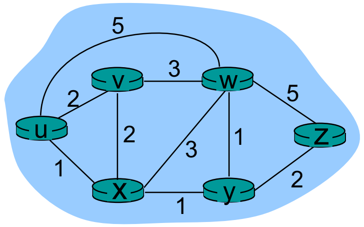
N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

Remark: Graph abstraction is useful in other network contexts

Example: P2P, where N is set of peers and E is set of TCP connections

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, link cost info
- "link state" algorithms

Decentralized:

- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- "distance 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
- NAT

Part 3

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

A Link-State Routing Algorithm

Dijkstra's algorithm

- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ('source") to all other nodes
  - gives forwarding table for that node
- iterative: after k iterations, know least cost path to k dest.'s

Notation:

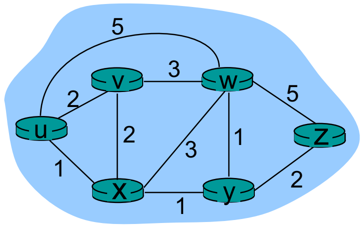
- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current value of cost of path from source to dest. v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

Dijsktra's Algorithm

```
Initialization:
 N' = {u}
 for all nodes v
   if v adjacent to u
       then D(v) = c(u,v)
   else D(v) = ∞

Loop
  find w not in N' such that D(w) is a minimum
  add w to N'
  update D(v) for all v adjacent to w and not in N' :
     D(v) = min( D(v), D(w) + c(w,v) )
  /* new cost to v is either old cost to v or known
   shortest path cost to w plus cost from w to v */
until all nodes in N'
```

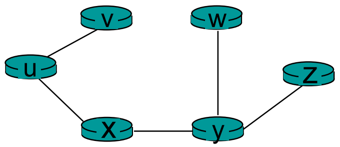
Dijkstra's algorithm: example

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Dijkstra's algorithm: example (2)

Resulting shortest-path tree from u:



Resulting forwarding table in u:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

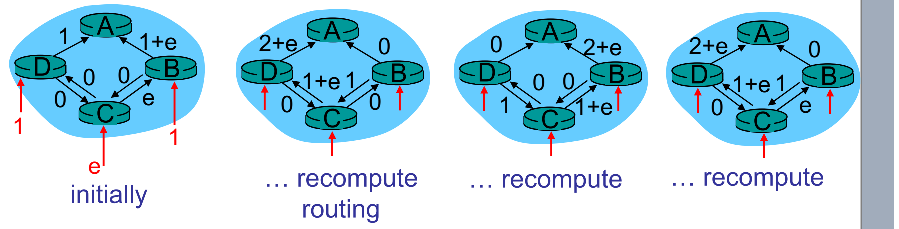
Dijkstra's algorithm, discussion

Algorithm complexity: n nodes

- each iteration: need to check all nodes, w, not in N
- n(n+1)/2 comparisons: $O(n^2)$
- more efficient implementations possible: O(nlogn)

Oscillations possible:

- e.g., link cost = amount of carried traffic



initially

… recompute routing

… recompute

… recompute

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

Distance Vector Algorithm

Bellman-Ford Equation (dynamic programming)

Define

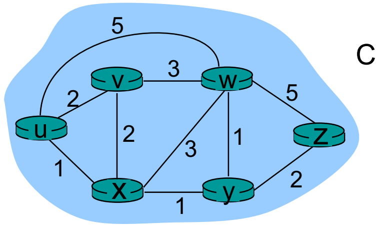
$d_x(y)$:= cost of least-cost path from x to y

Then

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

where min is taken over all neighbors 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

- $D_x(y)$ = estimate of least cost from x to y
- Node x knows cost to each neighbor v: c(x,v)
- Node x maintains distance vector $\mathbf{D}_x = [D_x(y): y \in N]$
- Node x also maintains its neighbors' distance vectors
  - For each neighbor 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 to neighbors
- Asynchronous
- When a node x receives new DV estimate from neighbor, 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, the estimate $D_x(y)$ *converge to the actual least cost* $d_x(y)$

Distance Vector Algorithm (5)

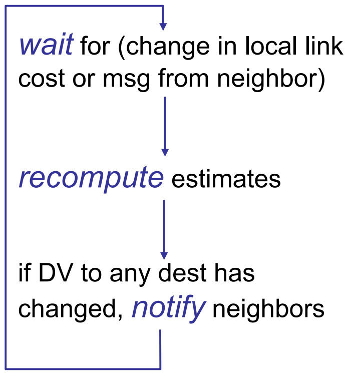
Iterative, asynchronous: each local iteration caused by:

- local link cost change
- DV update message from neighbor

Distributed:

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

Each node:

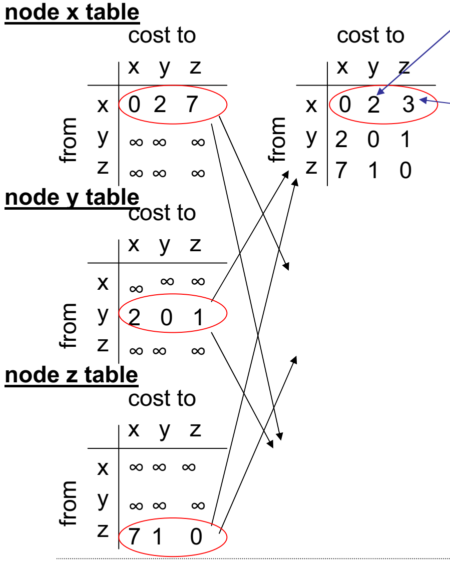


wait for (change in local link cost or msg from neighbor)

recompute estimates

if DV to any dest has changed, *notify* neighbors

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

node y table

cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

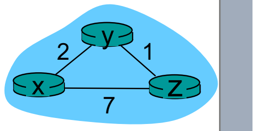
cost to

from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

time

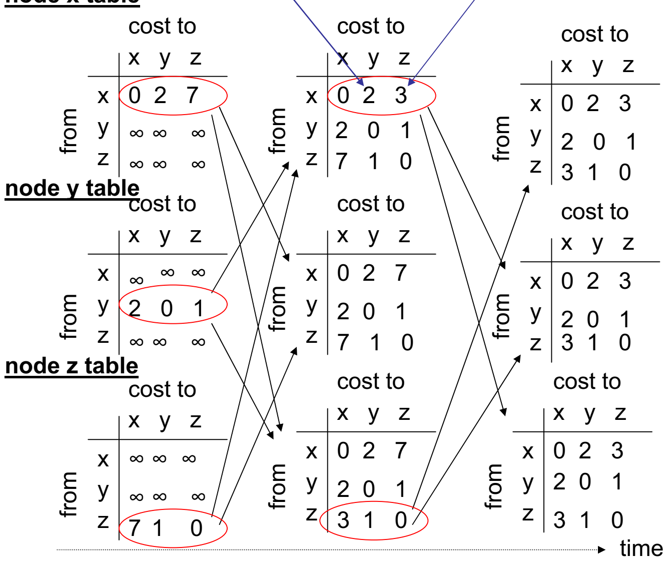
$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

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



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

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



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

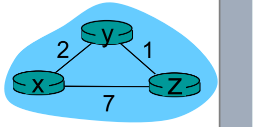
node z table

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

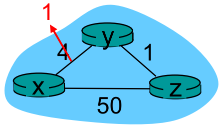
time



Distance Vector: link cost changes

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.

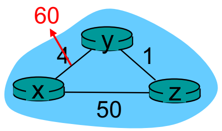
At time t_1, *z* receives the update from *y* and updates its table. It computes a new least cost to *x* and sends its neighbors its DV.

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

Link cost changes:

- ❑ good news travels fast
- ❑ bad news travels slow - "count to infinity" problem!
- ❑ 44 iterations before algorithm stabilizes: see text



Poisoned reverse:

- ❑ If Z routes through Y to get to X :
  - ▪ Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- ❑ will this completely solve count to infinity problem?

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

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

Hierarchical Routing

Our routing study thus far - idealization

- all routers identical
- network "flat"

… *not* true in practice

scale: with 200 million destinations:

- can't store all dest's 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

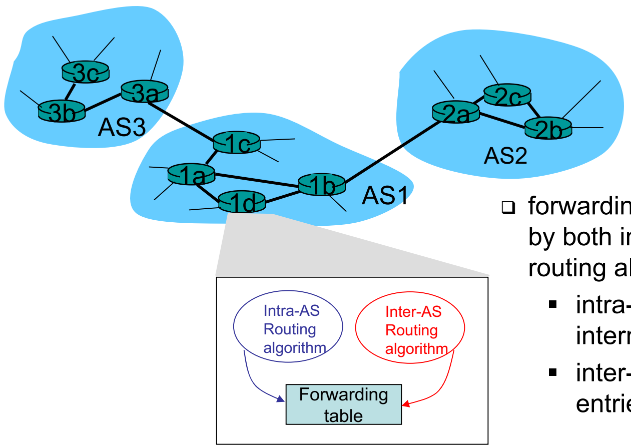
Hierarchical Routing

- aggregate routers into regions, "autonomous systems" (AS)
- routers in same AS run same routing protocol
  - "intra-AS" routing protocol
  - routers in different AS can run different intra-AS routing protocol

Gateway router

- Direct link to router in another AS

Interconnected ASes



- forwarding table configured by both intra- and inter-AS routing algorithm
  - intra-AS sets entries for internal dests
  - inter-AS & intra-As sets entries for external dests

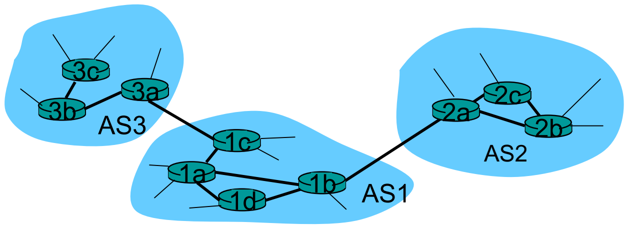
Inter-AS tasks

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

AS1 must:

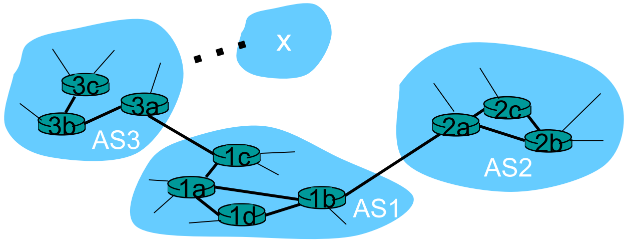
1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

Job of inter-AS routing!



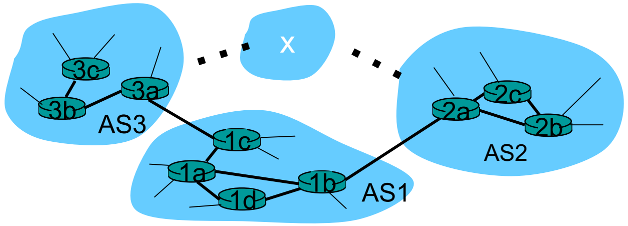
Example: Setting forwarding table in router 1d

- suppose AS1 learns (via inter-AS protocol) that subnet *x* 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* is on the least cost path to 1c.
  - installs forwarding table entry *(x,I)*



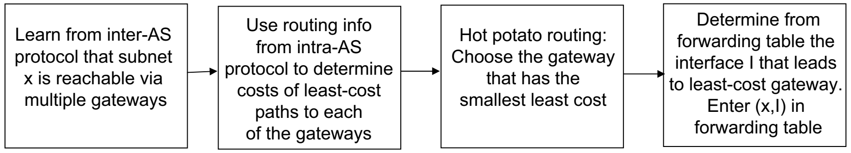
Example: Choosing among multiple ASes

- 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 dest x.
  - this is also job of inter-AS routing protocol!



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 dest x.
  - this is also job of inter-AS routing protocol!
- hot potato routing: send packet towards closest of two routers.



Learn from inter-AS protocol that subnet x is reachable via multiple gateways

Use routing info from intra-AS protocol to determine costs of least-cost paths to each of the gateways

Hot potato routing: Choose the gateway that has the smallest least cost

Determine from forwarding table the interface I that leads to least-cost gateway. Enter (x,I) in forwarding table

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

Intra-AS Routing

- also known as Interior Gateway Protocols (IGP)
- most common Intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

Chapter 4: Network Layer

Part 1

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

Part 2

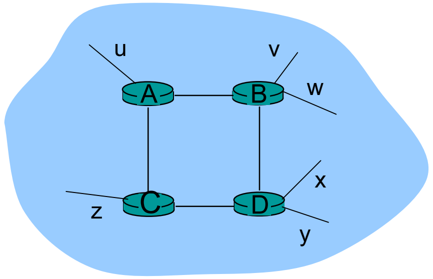
- IPv6
- Virtual circuit and datagram networks
- What's inside a router
- NAT

Part 3

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

RIP (Routing Information Protocol)

- distance vector algorithm
- included in BSD-UNIX Distribution in 1982
- distance metric: # of hops (max = 15 hops)



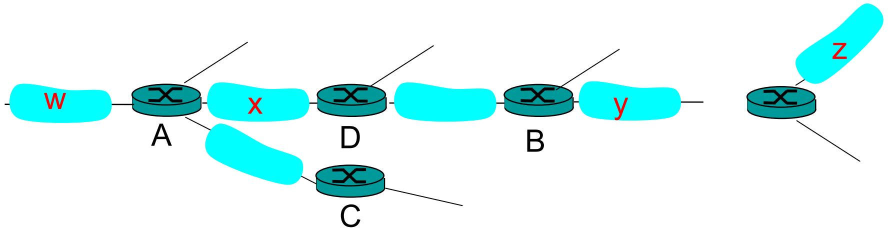
From router A to subnets:

destination	hops
u	1
v	2
w	2
x	3
y	3
z	2

RIP advertisements

- *distance vectors:* exchanged among neighbors every 30 sec via Response Message (also called advertisement)
- each advertisement: list of up to 25 destination subnets within AS

RIP: Example



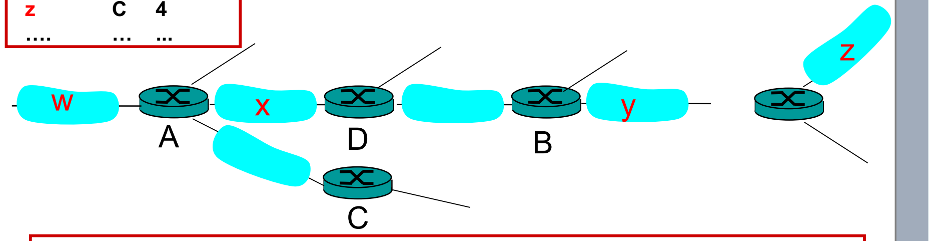
Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	B	7
x	--	1
….	….	….

Routing/Forwarding table in D

RIP: Example

Dest	Next	hops
w	-	1
x	-	1
z	C	4
….	…	…

Advertisement from A to D



Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	~~B~~ A	~~7~~ 5
x	--	1
….	….	….

Routing/Forwarding table in D

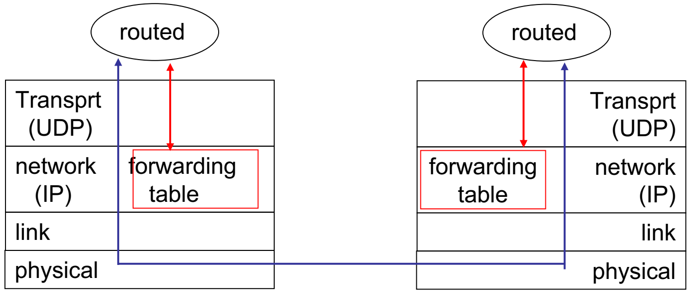
RIP: Link Failure and Recovery

If no advertisement heard after 180 sec --> neighbor/link declared dead

- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly (?) propagates to entire net
- *poison reverse* used to prevent ping-pong loops (infinite distance = 16 hops)

RIP Table processing

- RIP routing tables managed by **application-level** process called route-d (daemon)
- advertisements sent in UDP packets, periodically repeated



routed

Transprt (UDP)

network (IP)

forwarding table

link

physical

routed

Transprt (UDP)

forwarding table

network (IP)

link

physical

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

OSPF (Open Shortest Path First)

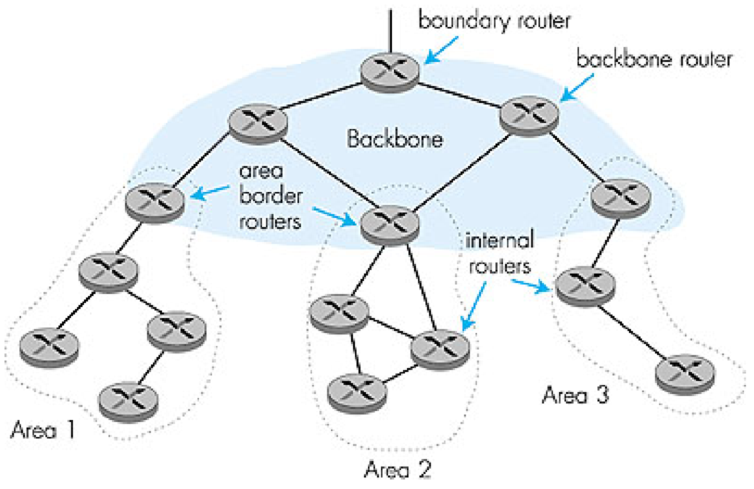
- "open": publicly available
- uses Link State algorithm
  - LS packet dissemination
  - topology map at each node
  - route computation using Dijkstra's algorithm

- 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

- two-level hierarchy: local area, backbone.
  - Link-state advertisements only in area
  - each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.
- *area border routers:* "summarize" distances to nets in own area, advertise to other Area Border routers.
- *backbone routers:* run OSPF routing limited to backbone.
- *boundary routers:* connect to other AS's.

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

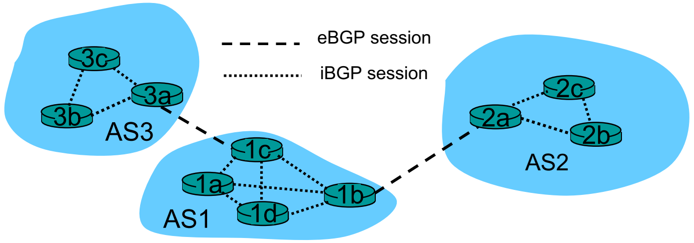
Part 3
- Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- **Routing in the Internet**
  - RIP
  - OSPF
  - **BGP**
- Broadcast and multicast routing

Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): *the* de facto standard
- BGP provides each AS a means to:
  1. Obtain subnet reachability information from neighboring ASs.
  2. Propagate reachability information to all AS-internal routers.
  3. Determine "good" routes to subnets based on reachability information and policy.
- allows subnet to advertise its existence to rest of Internet: *"I am here"*

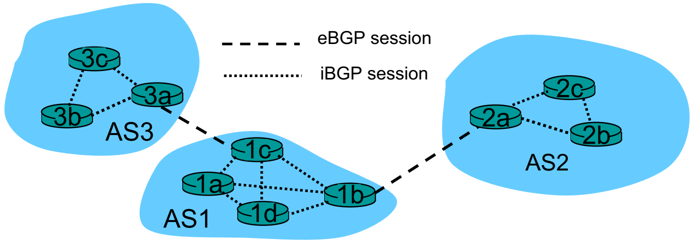
BGP basics

- 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 a prefix to AS1:
  - AS2 *promises* it will forward datagrams towards that prefix.
  - AS2 can aggregate prefixes in its advertisement



Distributing reachability info

- using eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1.
  - 1c can then use iBGP do 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, it creates entry for prefix in its forwarding table.



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.

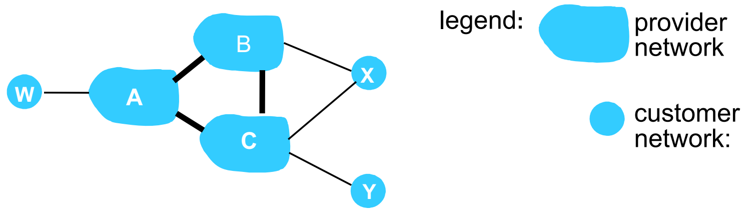
BGP route selection

- 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

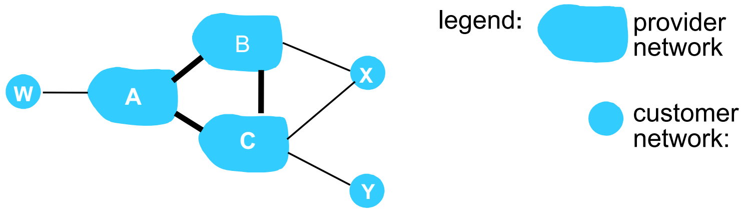
- BGP messages exchanged using TCP.
- BGP messages:
  - OPEN: opens TCP connection to peer and authenticates sender
  - UPDATE: advertises new path (or withdraws old)
  - KEEPALIVE keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION: reports errors in previous msg; also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

Chapter 4: Network Layer

Part 1

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

Part 2

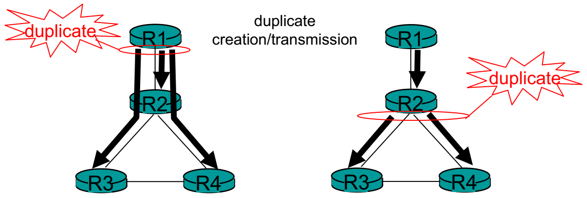
- IPv6
- Virtual circuit and datagram networks
- What's inside a router
- NAT

Part 3

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

Broadcast Routing

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



source duplication

in-network duplication

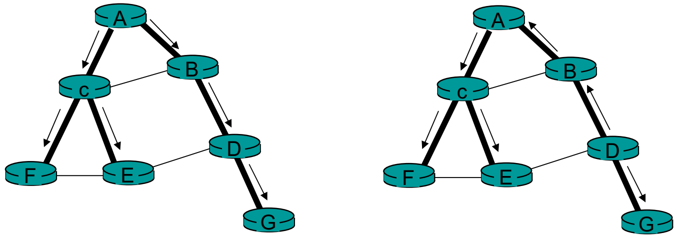
- source duplication: how does source determine recipient addresses?

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

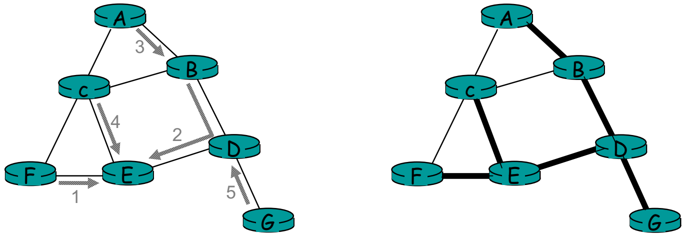


(a) Broadcast initiated at A

(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

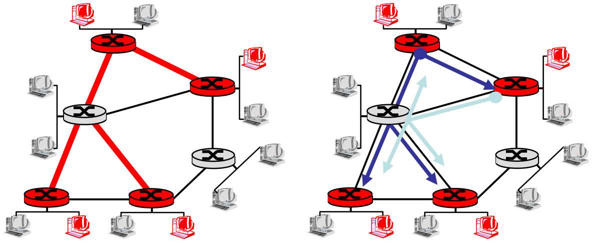


(a) Stepwise construction of spanning tree

(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

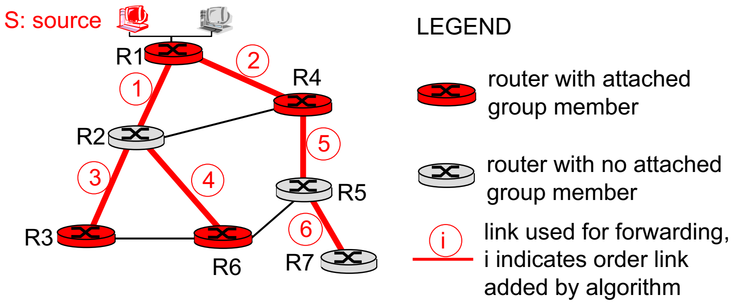
Approaches:

- 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



LEGEND

router with attached group member

router with no attached group member

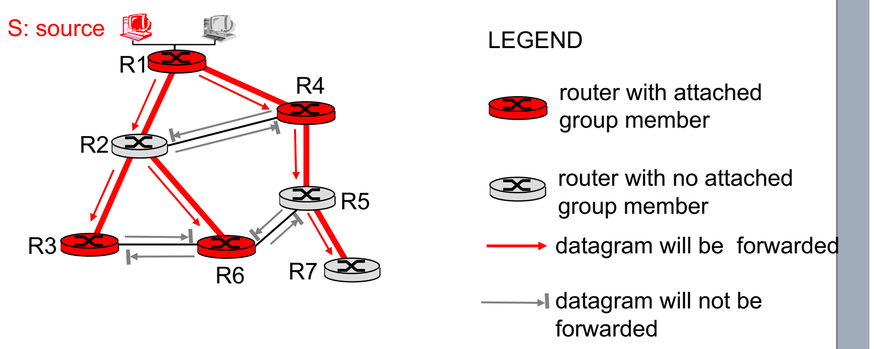
link used for forwarding, i indicates order link added by algorithm

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
```

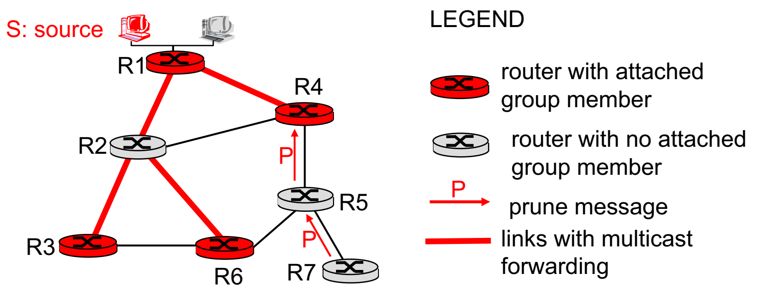
Reverse Path Forwarding: example


S: source
R1
R2
R3
R4
R5
R6
R7
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 mcast group members
  - ▪ no need to forward datagrams down subtree
  - ▪ "prune" msgs sent upstream by router with no downstream group members


S: source
R1
R2
R3
R4
R5
R6
R7
P
P
LEGEND
router with attached group member
router with no attached group member
P
prune message
links with multicast forwarding

Shared-Tree: Steiner Tree

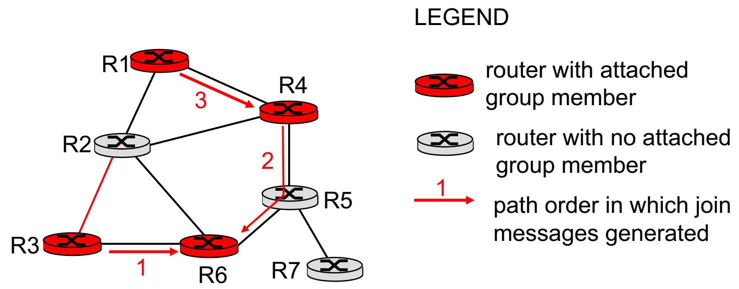
- ❑ Steiner Tree: minimum cost tree connecting all routers with attached group members
- ❑ problem is NP-complete
- ❑ excellent heuristics exists
- ❑ not used in practice:
  - ▪ computational complexity
  - ▪ information about entire network needed
  - ▪ monolithic: rerun whenever a router needs to join/leave

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:



LEGEND

- 🟥 router with attached group member
- ⬜ router with no attached group member
- 1 → path order in which join messages generated

Internet Multicasting Routing: DVMRP

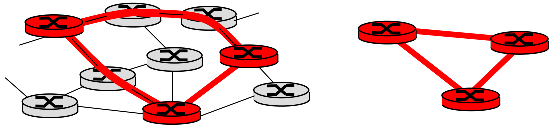
- DVMRP: distance vector multicast routing protocol, RFC1075
- *flood and prune:* reverse path forwarding, source-based tree
  - RPF tree based on DVMRP's own routing tables constructed by communicating DVMRP routers
  - no assumptions about underlying unicast
  - initial datagram to mcast group flooded 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*

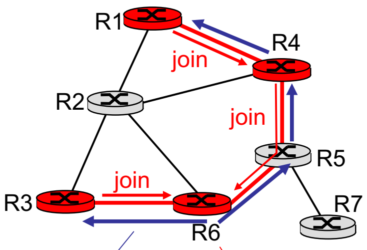
PIM- Dense Mode

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



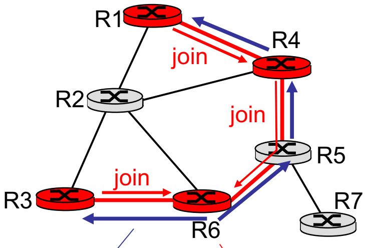
all data multicast from rendezvous point

rendezvous point

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!"



all data multicast from rendezvous point

rendezvous point

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

Chair for Network Architectures and Services – Prof. Carle
Department for Computer Science
TU München

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

Technische Universität München

Chapter 5: The Data Link Layer

Our 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

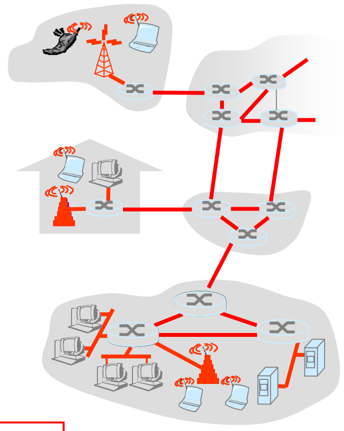
Link Layer

- 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 transmission over link

transportation analogy

- trip from Princeton to Lausanne
  - limo: Princeton to JFK
  - plane: JFK to Geneva
  - train: Geneva to Lausanne
- tourist = datagram
- transport segment = communication link
- transportation mode = link layer protocol
- travel agent = routing algorithm

Link Layer Services

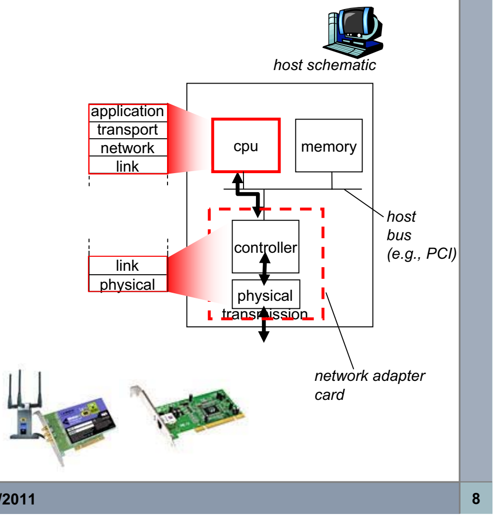
- framing, link access:
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - “MAC” addresses used in frame headers to identify source, dest
    - different from IP address!
- reliable delivery between adjacent nodes
  - we learned how to do this already (transport layer, 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)

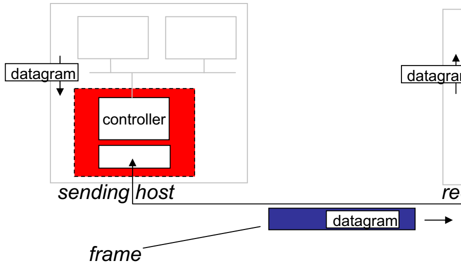
- *flow control:*
  - pacing between adjacent sending and receiving nodes
- *error detection*:
  - errors caused by signal attenuation, noise.
  - receiver detects presence of errors:
    - signals sender for retransmission or drops frame
- error correction:
  - receiver identifies *and corrects* bit 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, PCMCI 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 transmission, flow control, etc.
- receiving side
  - looks for errors, rdt, flow control, etc
  - extracts datagram, passes to upper layer at receiving side

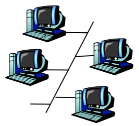
Link Layer

- 5.1 Introduction and services
- 5.2 Multiple access protocols
- 5.3 Link-layer Addressing
- 5.4 Ethernet
- 5.5 Link-layer switches

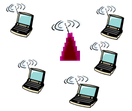
Multiple Access Links and Protocols

Two types of "links":

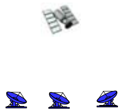
- point-to-point
  - PPP for dial-up access
  - point-to-point link between Ethernet switch and host
- broadcast (shared wire or medium)
  - old-fashioned Ethernet
  - upstream HFC
  - 802.11 wireless LAN



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

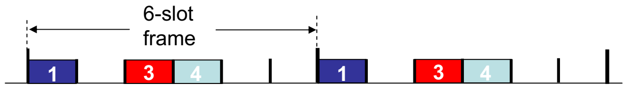
Three broad classes:

- 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

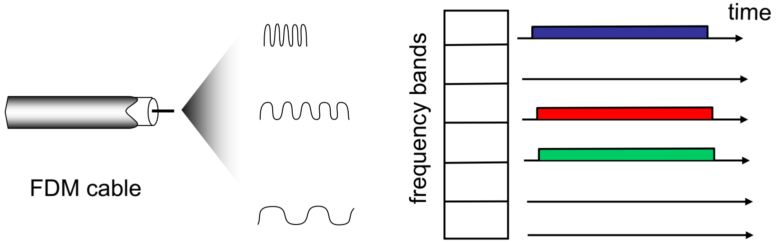
- access to channel in "rounds"
- each station gets fixed length slot (length = pkt trans time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have packet; slots 2,5,6 are idle



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 are 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:
  - slotted ALOHA
  - ALOHA
  - CSMA, CSMA/CD, CSMA/CA

Slotted ALOHA

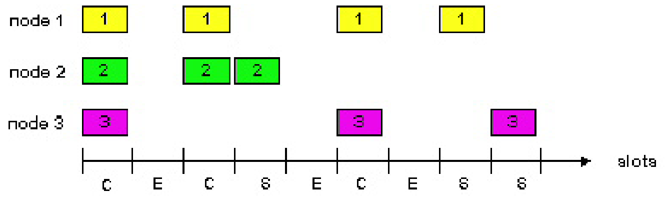
Assumptions:

- all frames same size
- time divided into equal size slots (time to transmit 1 frame)
- nodes start to transmit only at slot beginning
- nodes are synchronized
- if 2 or more nodes transmit in slot, all nodes detect collision

Operation:

- when node obtains fresh frame, transmits in next slot
  - *if no collision:* node can send new frame in next slot
  - *if collision:* node retransmits frame in each subsequent slot with prob. p until success

Slotted ALOHA



Pros

- single active node can continuously transmit at full rate of channel
- highly decentralized: only slots in nodes need to be in sync
- simple

Cons

- collisions, wasting slots
- idle slots
- nodes may be able to detect collision in less time than time to transmit packet
- clock synchronization

Slotted Aloha efficiency

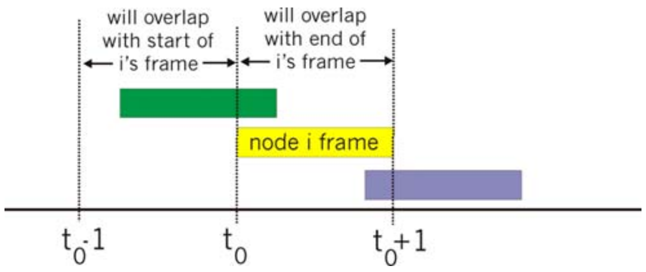
Efficiency: long-run fraction of successful slots (many nodes, all with many frames to send)

- *suppose:* N nodes with many frames to send, each transmits in slot with probability *p*
- probability that given node has success in a slot = one node transmits ^ N nodes do not transmit = $p(1-p)^{N-1}$
- probability that *any* node has a success = $Np(1-p)^{N-1}$
- max efficiency: find p* that maximizes $Np(1-p)^{N-1}$
- for many nodes, take limit of $Np^*(1-p^*)^{N-1}$ as N goes to infinity, gives: Max efficiency = 1/e = .37

At best: channel used for useful transmissions 37% of time! !

Pure (unslotted) ALOHA

- unslotted Aloha: simpler, no synchronization
- when frame first arrives
  - transmit immediately
- collision probability increases:
  - frame sent at t0 collides with other frames sent in [t0-1,t0+1]


will overlap with start of i's frame
will overlap with end of i's frame
node i frame
t_0-1
t_0
t_0+1

Pure Aloha efficiency

P(success by given node) = P(node transmits) ·

P(no other node transmits in $[t_0-1,t_0]$ ·

P(no other node transmits in $[t_0,t_0+1]$

$= p \cdot (1-p)^{N-1} \cdot (1-p)^{N-1}$

$= p \cdot (1-p)^{2(N-1)}$

P(success by *any* node) = $Np(1-p)^{2(N-1)}$

… choosing optimum p and letting n → infinity ...

⇨ Max efficiency = 1/(2e) = .18

⇨ only 50% of slotted Aloha!

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

collisions *can* still occur:

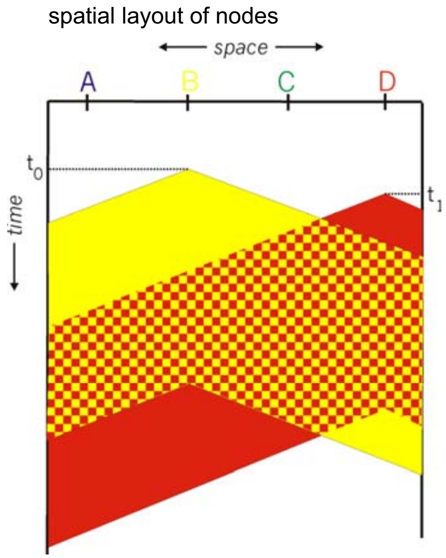
propagation delay means two nodes may not hear each other's transmission

collision:

entire packet transmission time wasted

note:

role of distance & propagation delay in determining collision probability



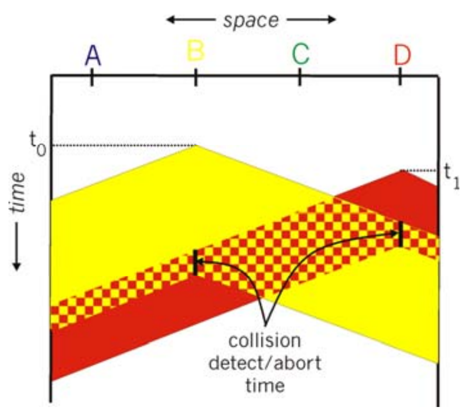
CSMA/CD (Collision Detection)

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

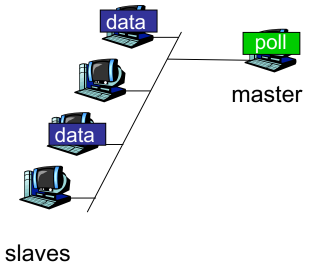
"taking turns" protocols

look for best of both worlds!

"Taking Turns" MAC protocols

Polling:

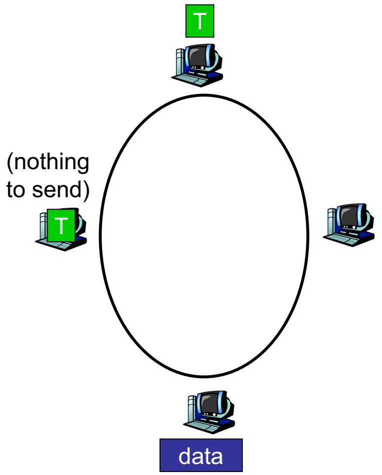
- ❑ master node "invites" slave nodes to transmit in turn
- ❑ typically used with "dumb" slave devices
- ❑ concerns:
  - polling overhead
  - latency
  - single point of failure (master)



"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
  - Bluetooth, FDDI, IBM Token Ring

Link Layer

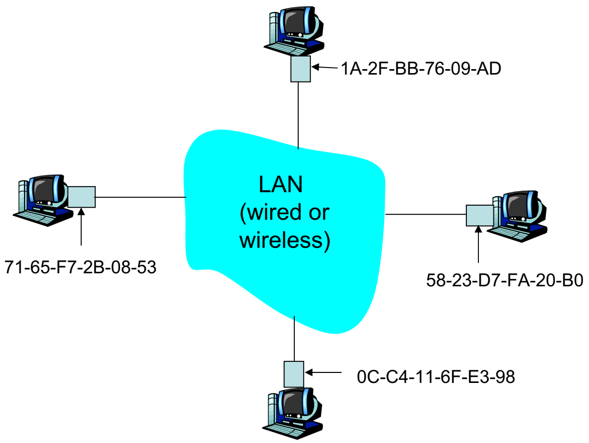
- 5.1 Introduction and services
- 5.2 Multiple access protocols
- 5.3 Link-layer Addressing
- 5.4 Ethernet
- 5.5 Link-layer switches

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

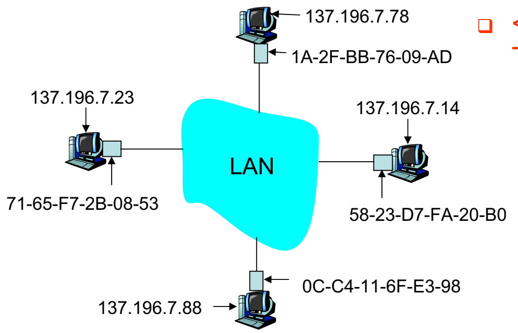
▭ = adapter

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)

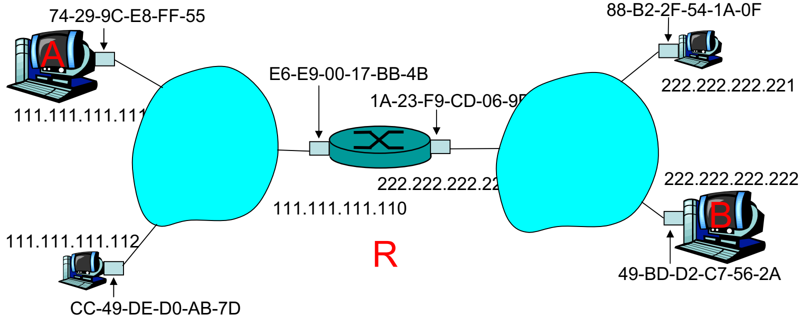
ARP protocol: Same LAN (network)

- 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

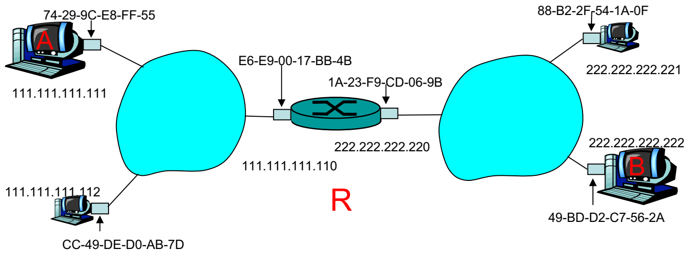
  assume A knows B's IP address



- 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 uses ARP to get R’s MAC address for 111.111.111.110
- A creates link-layer frame with R's MAC address as destination, 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 and sends it to B

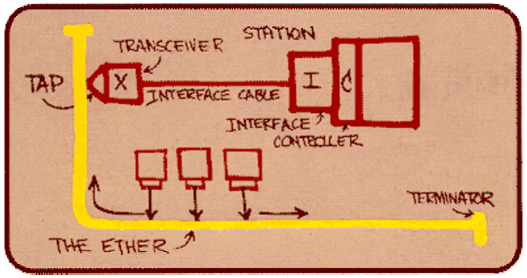


Link Layer

- 5.1 Introduction and services
- 5.2 Multiple access protocols
- 5.3 Link-layer Addressing
- 5.4 Ethernet
- 5.5 Link-layer switches

Ethernet

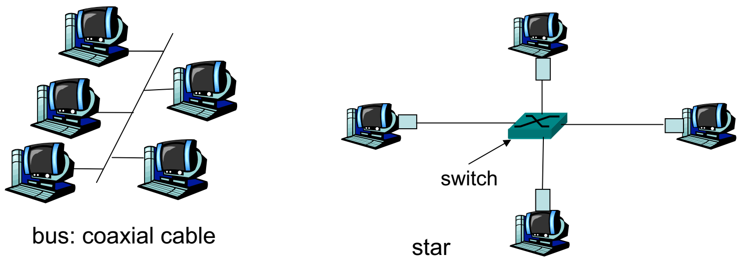
- “dominant” wired LAN technology:
- cheap $20 for NIC
- first widely used LAN technology
- simpler, cheaper than token LANs and ATM
- kept up with speed race: 10 Mbps – 10 Gbps



Metcalfe’s Ethernet sketch

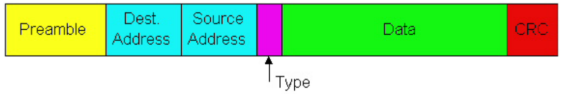
Star topology

- 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)



Ethernet Frame Structure

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

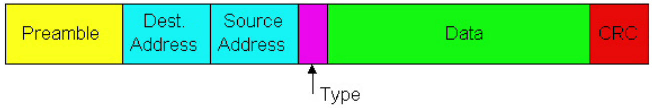


Preamble:

- 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 NAKs 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, application 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·512 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 - recommended!

CSMA/CD Efficiency

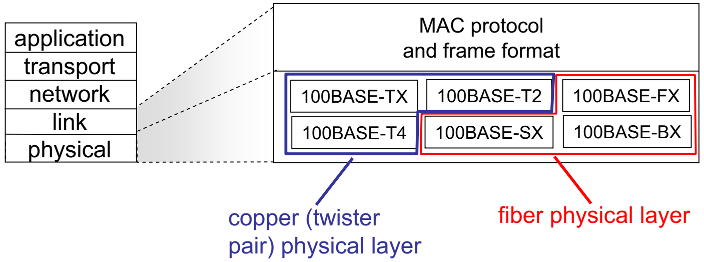
- T_{prop} = max propagation delay between 2 nodes in LAN
- t_{trans} = time to transmit max-size frame
- Approximation:

$$efficiency = \frac{1}{1+5t_{prop}/t_{trans}}$$

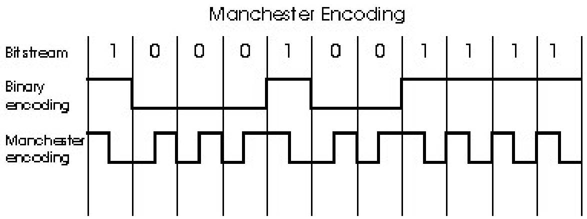
- Efficiency goes to 1
  - as t_{prop} goes to 0
  - as t_{trans} goes to infinity
- Better performance than ALOHA: and simple, cheap, decentralized!

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



Manchester encoding



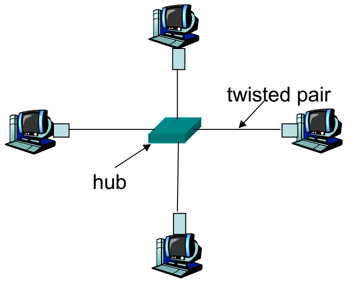
- used in 10BaseT
- each bit has a transition
- allows clocks in sending and receiving nodes to synchronize to each other
  - no need for a centralized, global clock among nodes!
- This is physical-layer stuff

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

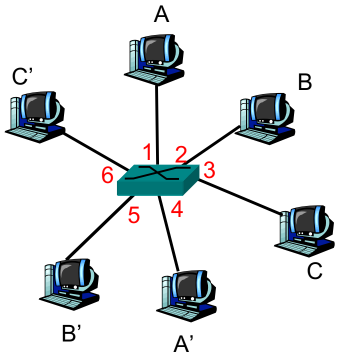


Switch

- 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

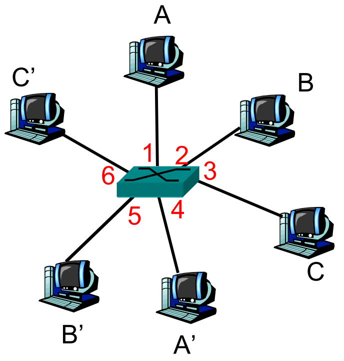
- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, but no collisions; full duplex
  - each link is its own collision domain
- *switching:* A-to-A’ and B-to-B’ simultaneously, without collisions
  - not possible with dumb hub



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

Switch Table

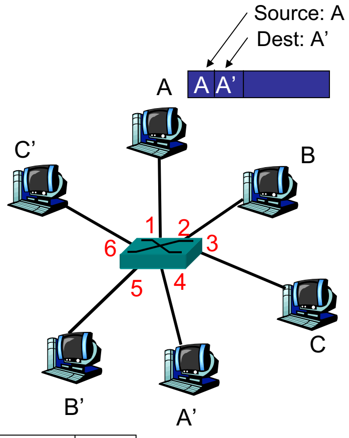
- *Q:* how does switch know that A' reachable via interface 4, B' reachable via interface 5?
- *A:* each switch has a switch table, each entry:
  - (MAC address of host, interface to reach host, time stamp)
- looks like a routing table!
- *Q:* how are entries created, maintained in switch table?
  -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

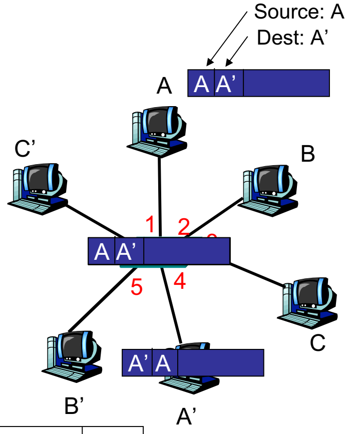
When frame received:

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

forward on all but the interface on which the frame arrived

Self-learning, forwarding: example

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

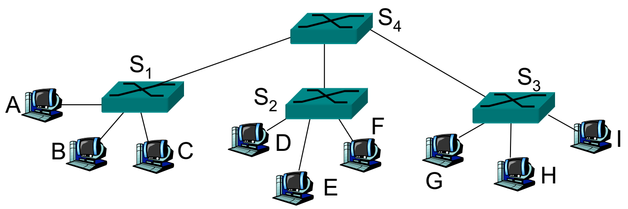


MAC addr	interface	TTL
A *A'*	*1* *4*	*60* *60*

Switch table (initially empty)

Interconnecting switches

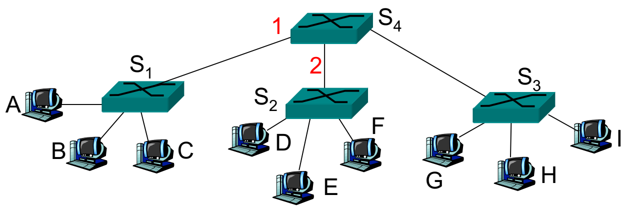
- switches can be connected together



- *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!)

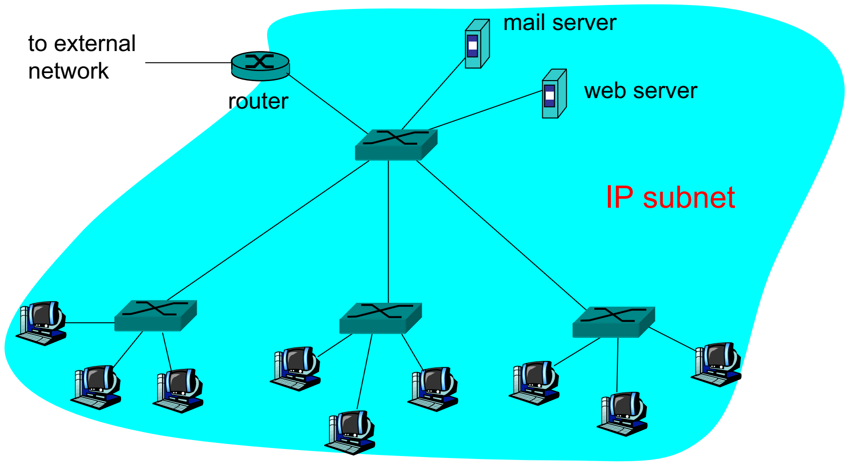
Self-learning multi-switch example

- Suppose C sends frame to I, I responds to C



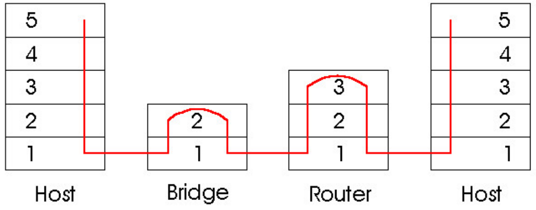
- *Q:* show switch tables and packet forwarding in S_1, S_2, S_3, S_4

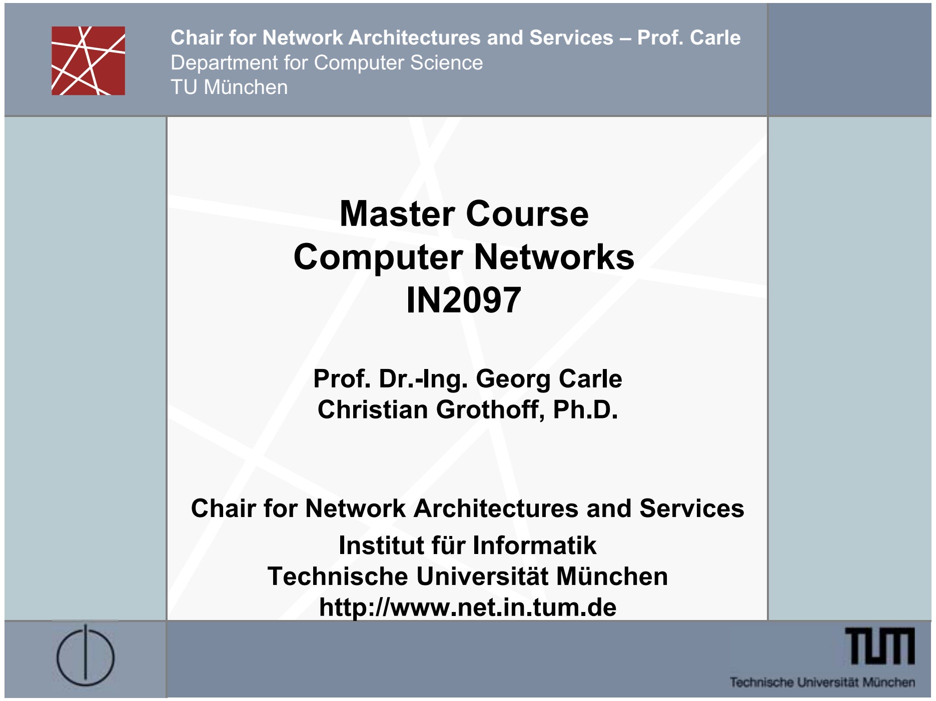
Institutional network



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





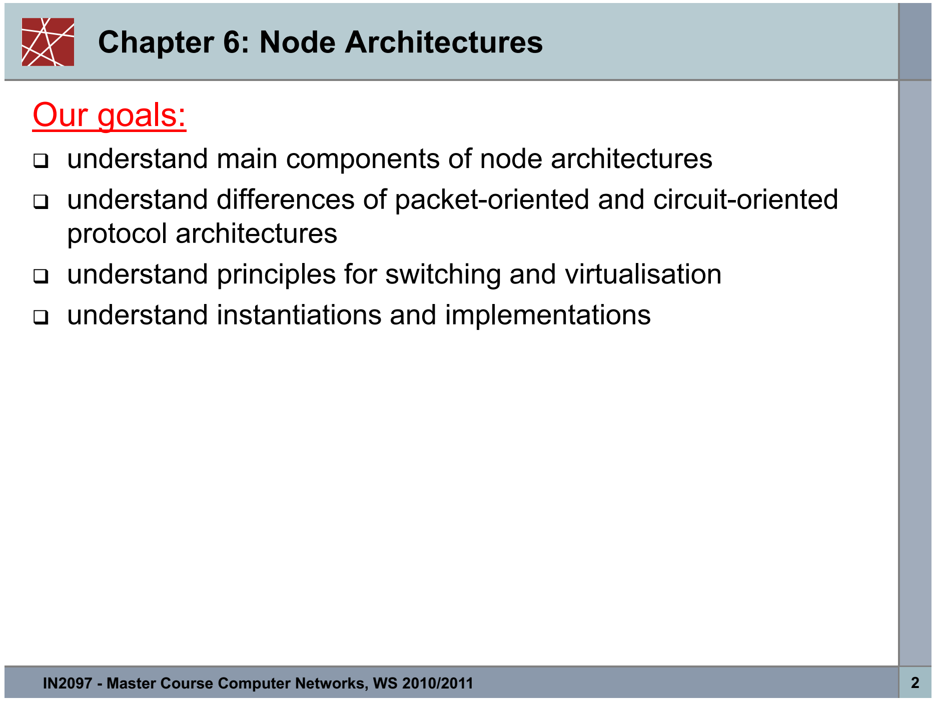
Chair for Network Architectures and Services – Prof. Carle
Department for Computer Science
TU München

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

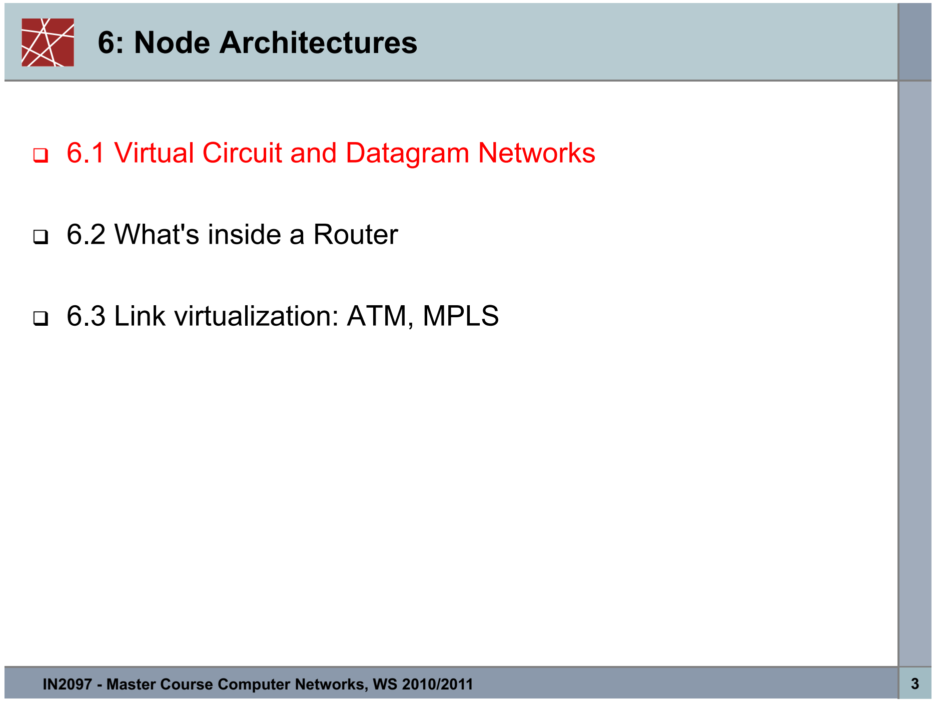
Technische Universität München



Chapter 6: Node Architectures

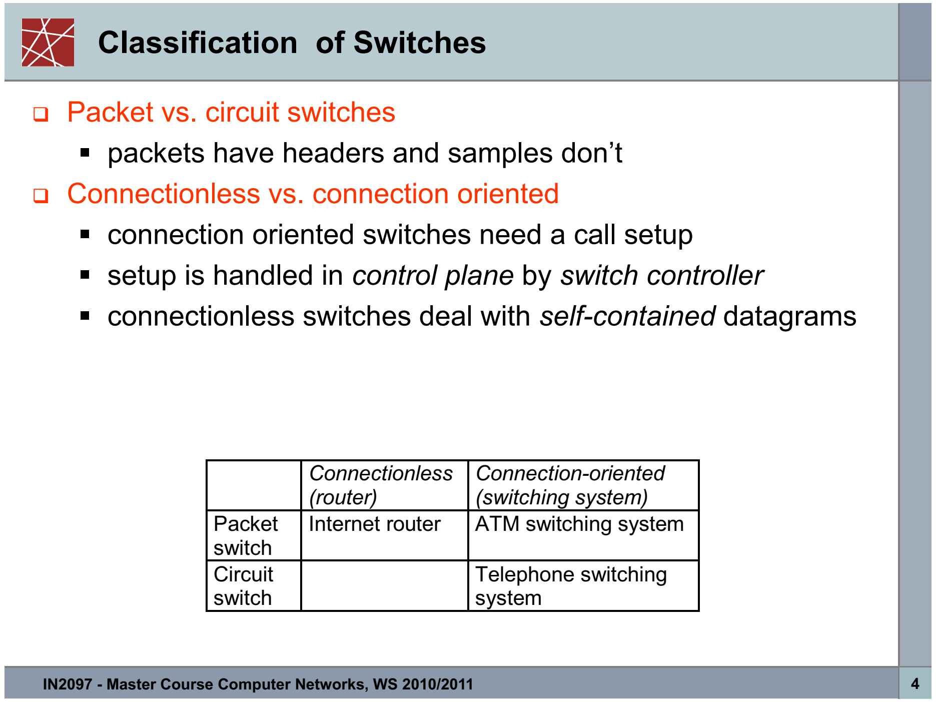
Our goals:

- understand main components of node architectures
- understand differences of packet-oriented and circuit-oriented protocol architectures
- understand principles for switching and virtualisation
- understand instantiations and implementations



6: Node Architectures

- 6.1 Virtual Circuit and Datagram Networks
- 6.2 What's inside a Router
- 6.3 Link virtualization: ATM, MPLS



Classification of Switches

- Packet vs. circuit switches
  - packets have headers and samples don’t
- Connectionless vs. connection oriented
  - connection oriented switches need a call setup
  - setup is handled in *control plane* by *switch controller*
  - connectionless switches deal with *self-contained* datagrams

	Connectionless (router)	*Connection-oriented (switching system)*
Packet switch	Internet router	ATM switching system
Circuit switch		Telephone switching system

Requirements

- Capacity of a switch is the maximum rate at which it can move information, assuming all data paths are simultaneously active
- Primary goal: maximize capacity
  - subject to cost and reliability constraints
- Circuit switch must reject call if can't find a path for samples from input to output
  - goal: minimize call blocking
- Packet switch must reject a packet if it can't find a buffer to store it awaiting access to output trunk
  - goal: minimize packet loss
- Don't reorder packets

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?

Example services for individual datagrams:

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

Example services for a flow of datagrams:

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

Network layer service models

Network Architecture	Service Model	Guarantees ? 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-destination path behaves much like telephone circuit"
> - performance-wise
> - network actions along source-to-destination path

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

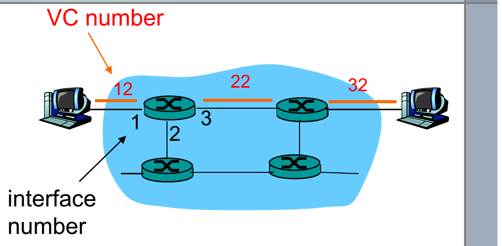
VC implementation

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 destination address)
- VC number can be changed on each link.
  - New VC number comes from forwarding table

Forwarding table



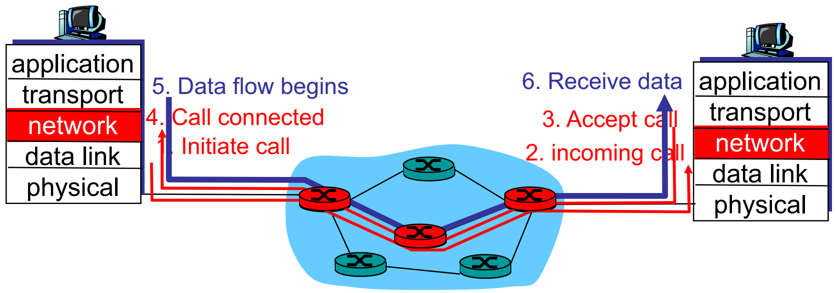
Forwarding table in northwest router:

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!

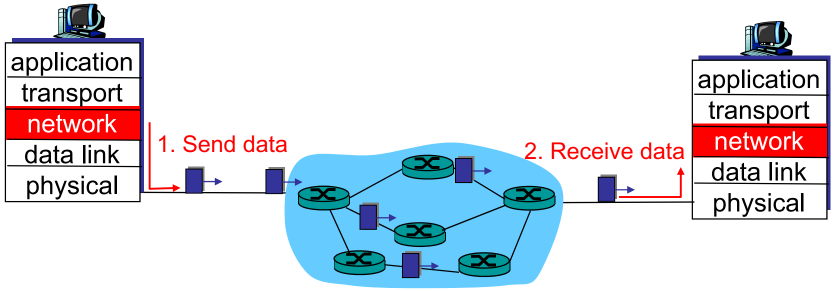
Virtual circuits: signaling protocols

- ❑ 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

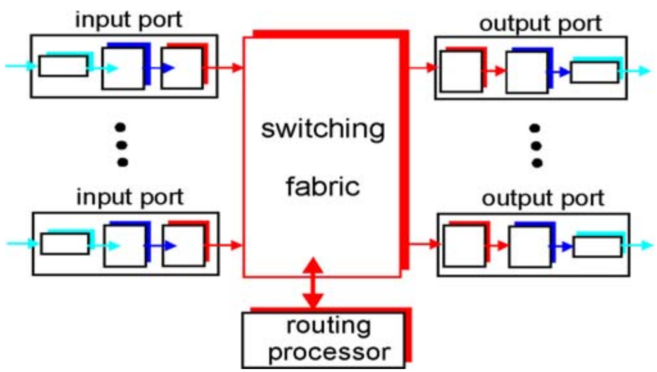
6: Node Architectures

- ❑ 6.1 Virtual Circuit and Datagram Networks
- ❑ 6.2 What's inside a Router
- ❑ 6.3 Link virtualization: ATM, MPLS

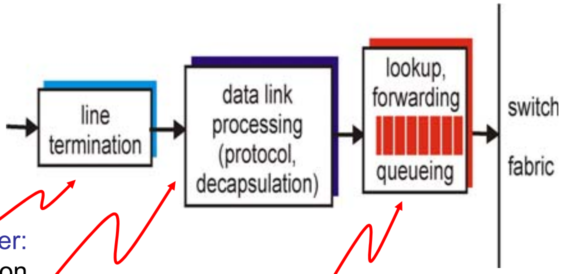
Router Architecture Overview

Two key router functions:

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



Input Port Functions



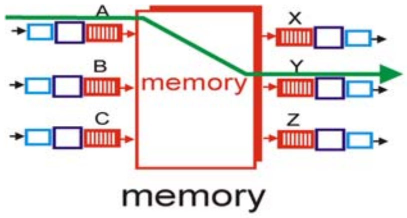
Physical layer:
bit-level reception

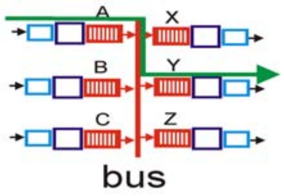
Data link layer:
e.g., Ethernet

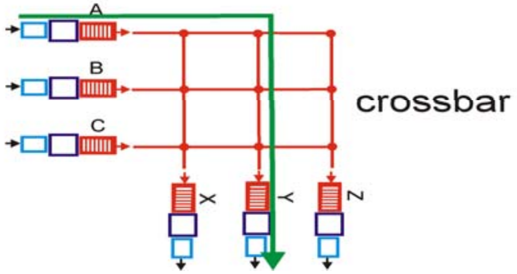
Decentralized switching:

- given datagram destination, 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



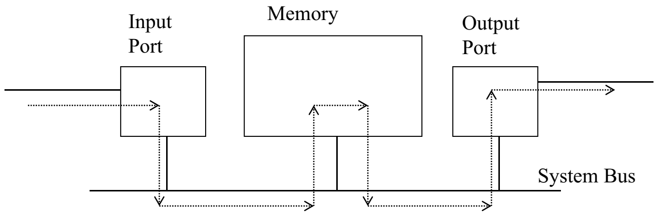




Switching Via Memory

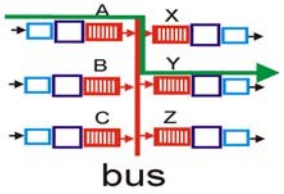
First generation routers:

- 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

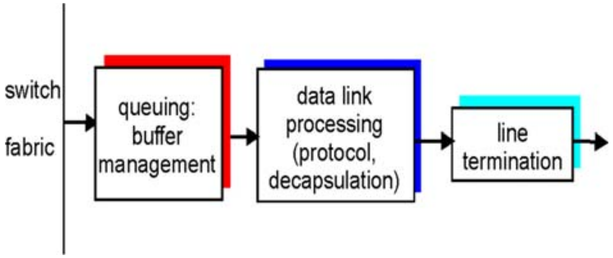
- datagram from input port memory to output port memory via a shared bus
- bus contention: switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers



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

Output Ports



switch fabric

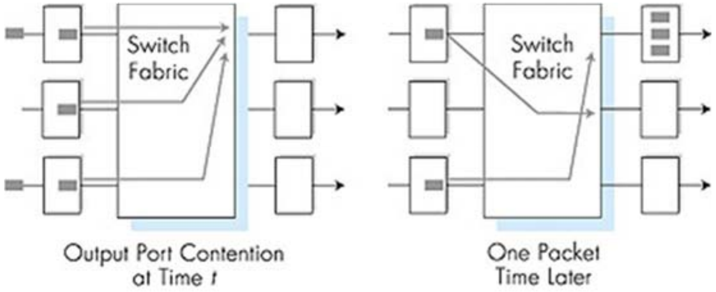
queuing: buffer management

data link processing (protocol, decapsulation)

line termination

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

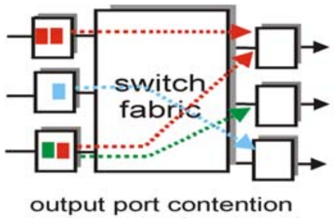
Output port queueing



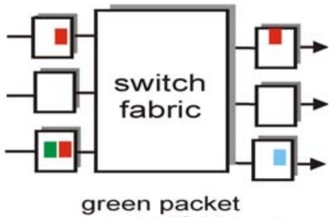
- buffering when arrival rate via switch exceeds output line speed
- *queueing (delay) and loss due to output port buffer overflow!*

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

6: Node Architectures

- 6.1 Virtual Circuit and Datagram Networks
- 6.2 What's inside a Router
- 6.3 Link virtualization: ATM, MPLS

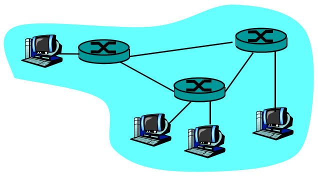
Virtualization of networks

- Virtualization of resources: powerful abstraction in systems engineering:
- computing examples: virtual memory, virtual devices
  - Virtual machines: e.g., java
  - IBM VM operation system from 1960's/70's
- layering of abstractions: don't sweat the details of the lower layer, only deal with lower layers abstractly

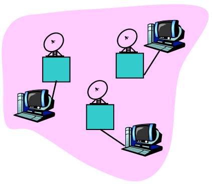
The Internet: virtualizing networks

- 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.

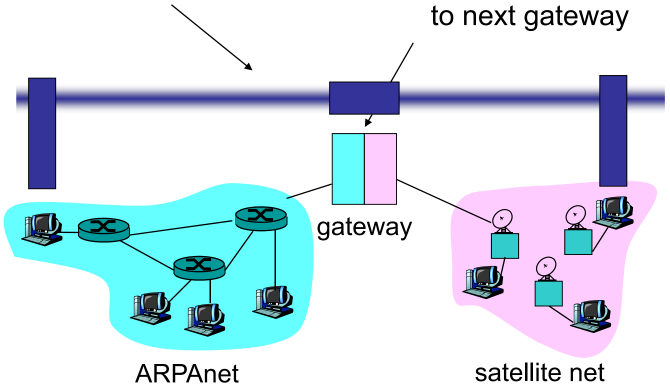
The Internet: virtualizing networks

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?
- 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. Looks like a link layer technology to IP!

ATM and MPLS

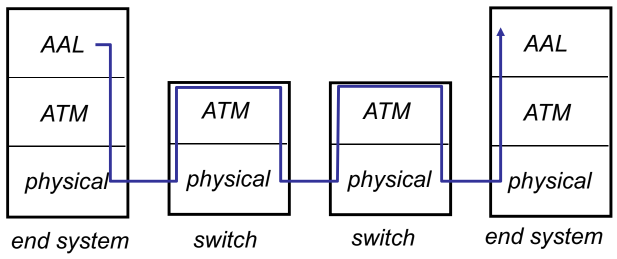
- ATM, MPLS separate networks in their own right
  - different service models, addressing, routing from Internet
- viewed by Internet as logical link connecting IP routers
  - just like dialup link is really part of separate network (telephone network)
- ATM, MPLS: of technical interest in their own right

Asynchronous Transfer Mode: ATM

- **1990’s/00 standard for high-speed** (155Mbps to 622 Mbps and higher) *Broadband Integrated Service Digital Network* architecture

- Goal: *integrated, end-end transport of 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



- ❑ adaptation layer: only at edge of ATM network
  - data segmentation/reassembly
  - roughly analagous to Internet transport layer
- ❑ ATM layer: “network” layer
  - cell switching, routing
- ❑ physical layer

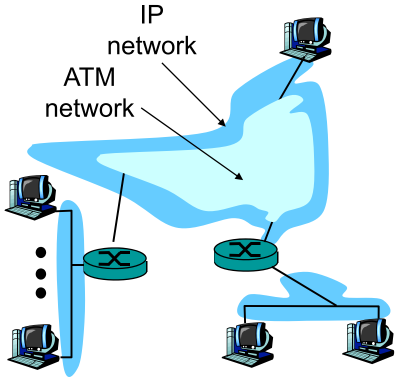
ATM: network or link layer?

Vision: end-to-end transport: “ATM from desktop to desktop”

- ATM *is* a network technology

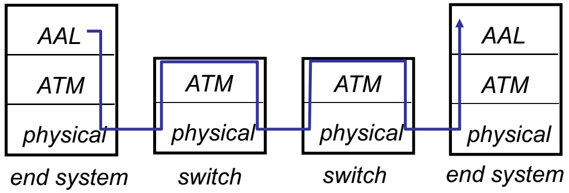
Reality: used to connect IP backbone routers

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



ATM Adaptation Layer (AAL)

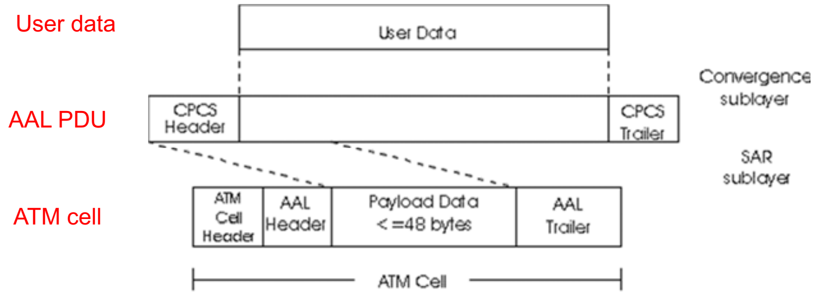
- ❑ ATM **Adaptation Layer** (AAL): “adapts” upper layers (IP or native ATM applications) to ATM layer below
- ❑ AAL present **only in end systems**, not in switches
- ❑ AAL layer segment (header/trailer fields, data) fragmented across multiple ATM cells
  - analogy: TCP segment in many IP packets



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)



ATM Layer

Service: transport cells across ATM network

- analogous to IP network layer
- very different services than IP network layer

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

ATM Layer: Virtual Circuits

- VC transport: cells carried on VC from source to dest
  - call setup, teardown for each call *before* data can flow
  - each packet carries VC identifier (not destination ID)
  - *every* switch on source-dest path maintain “state” for each passing connection
  - link,switch resources (bandwidth, buffers) may be *allocated* to VC: to get circuit-like perf.
- Permanent VCs (PVCs)
  - long lasting connections
  - typically: “permanent” route between to IP routers
- Switched VCs (SVC):
  - dynamically set up on per-call basis

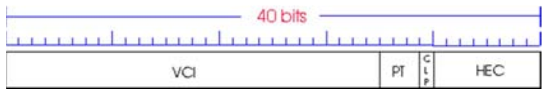
ATM VCs

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

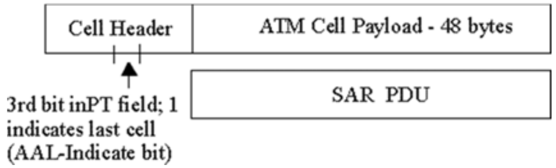
ATM Layer: ATM cell

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

Cell header

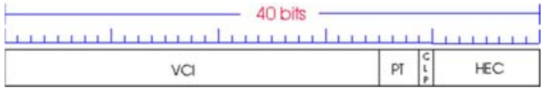


Cell format


Cell Header
ATM Cell Payload - 48 bytes
SAR PDU
3rd bit inPT field; 1 indicates last cell (AAL-Indicate bit)

ATM cell header

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



ATM Physical Layer (more)

Two pieces (sublayers) of physical layer:

- Transmission Convergence Sublayer (TCS): adapts ATM layer above to 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)

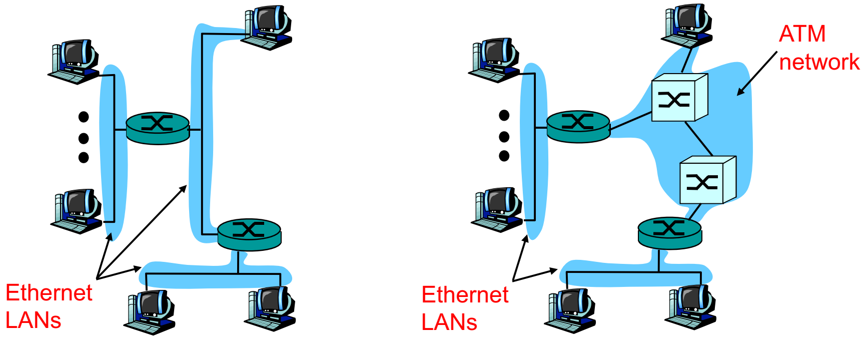
IP-Over-ATM

Classic IP only

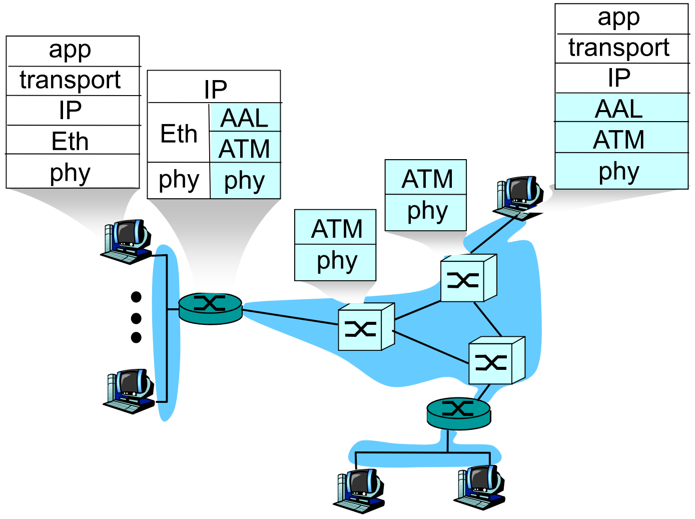
- 3 “networks” (e.g., LAN segments)
- MAC (802.3) and IP addresses

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

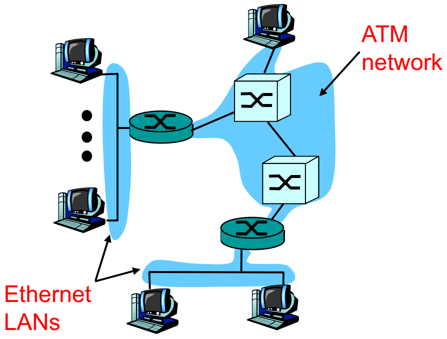
Datagram Journey in IP-over-ATM Network

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

IP-Over-ATM

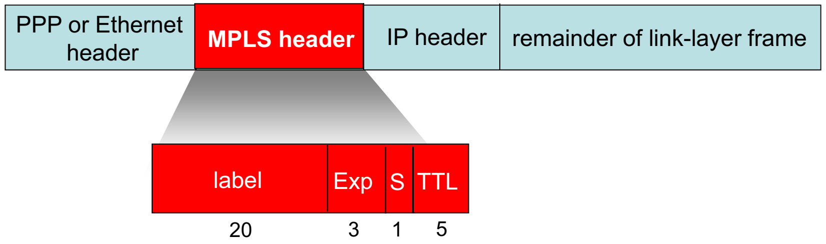
Issues:

- IP datagrams into ATM AAL5 PDUs
- from IP addresses to ATM addresses
  - just like IP addresses to 802.3 MAC addresses!



Multiprotocol label switching (MPLS)

- initial goal: speed up IP forwarding by using fixed length label (instead of IP address) to do forwarding
  - borrowing ideas from Virtual Circuit (VC) approach
  - but IP datagram still keeps IP address!


PPP or Ethernet header
MPLS header
IP header
remainder of link-layer frame

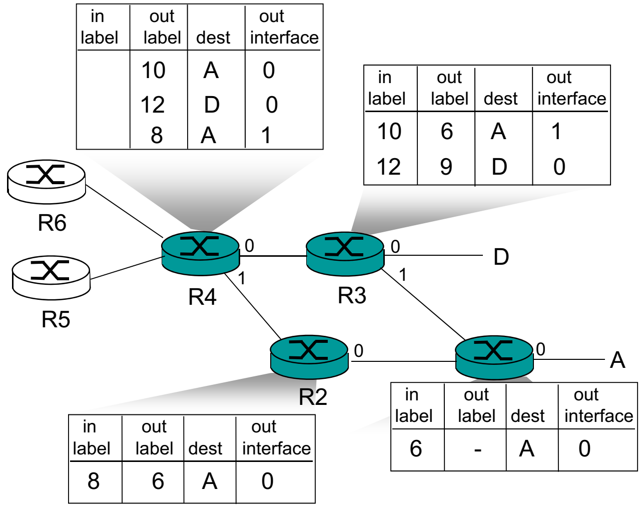
label
Exp
S
TTL

20
3
1
5

MPLS capable routers

- a.k.a. label-switched router
- forwards packets to outgoing interface based only on label value (don't inspect IP address)
  - MPLS forwarding table distinct from IP forwarding tables
- signaling protocol needed to set up forwarding
  - RSVP-TE
  - forwarding possible along paths that IP alone would not allow (e.g., source-specific routing) !!
  - use MPLS for traffic engineering
- must co-exist with IP-only routers

MPLS forwarding tables



in label	out label	dest	out interface
	10	A	0
	12	D	0
	8	A	1

in label	out label	dest	out interface
10	6	A	1
12	9	D	0

in label	out label	dest	out interface
8	6	A	0

in label	out label	dest	out interface
6	-	A	0

Chair for Network Architectures and Services – Prof. Carle
Department for Computer Science
TU München

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

Technische Universität München

Chapter 7 - 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

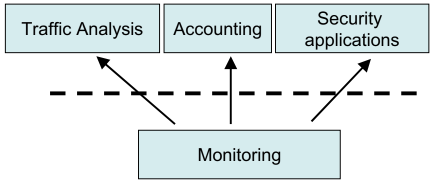
- Passive measurements (or **Network Monitoring**)
  - “non-intrusive”
  - Monitoring of existing traffic
  - Establishing of packet traces at different locations
  - Identification of packets, e.g. using hash values
  - Advantages
    - Does not affect applications
    - Does not modify the network 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

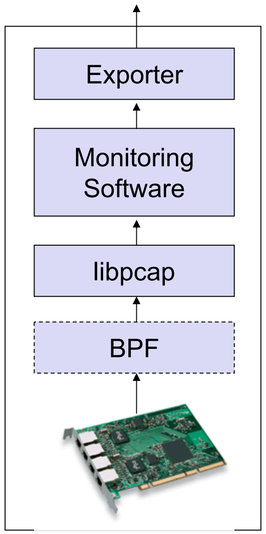
- Applications of network monitoring
  - Traffic analysis
    - Traffic engineering
    - Anomaly detection
  - Accounting
    - Resource utilization
    - Accounting and charging
  - Security
    - Intrusion detection
    - Detection of prohibited data transfers (e.g., P2P applications)



- Open issues
  - Protection of measurement data against illegitimate use (encryption, …)
  - Applicable law (“lawful interception”)

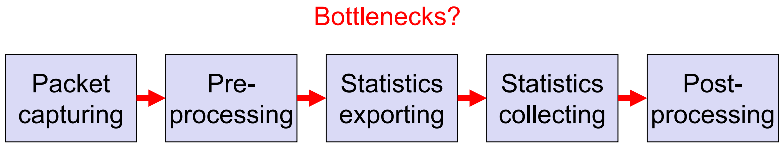
Monitoring Probe

- 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



High-Speed Network Monitoring II

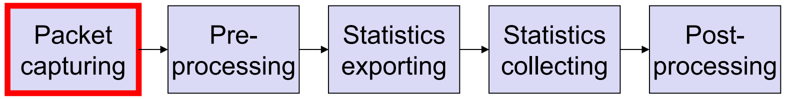
- Approaches
  - High-end (intelligent) network adapters
  - Sophisticated algorithms for
    - Maintaining packet queues
    - Elimination of packet copy operations
    - Managing hash tables describing packet flows
  - Sampling
  - Filtering
  - Aggregation

  ⇨ more on subsequent slides

Special Network Adapters

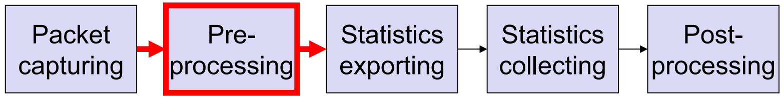
- Server NICs (Network Interface Cards)
  - Direct access to main memory (without CPU assistance)
  - Processing of multiple packets in a single block (reduction of copy operations)
    → Reduced interrupt rates

- Monitoring interface cards
  - Dedicated monitoring hardware
  - Programmable, i.e. certain processing can be performed on the network interface card



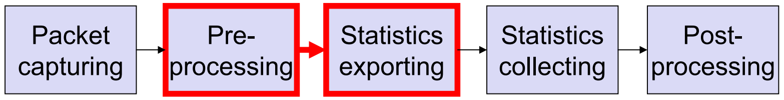
Memory Management

- 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



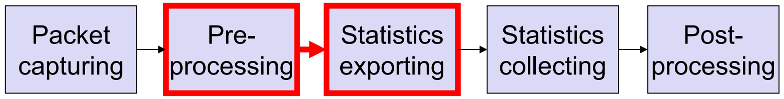
Packet Sampling

- 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

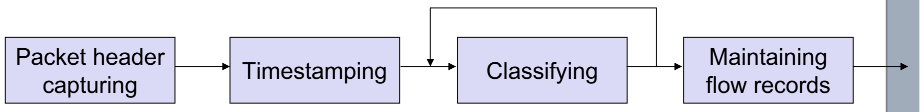
- IPFIX (IP Flow Information eXport) IETF Working Group
  - Standard track protocol based on Cisco Netflow v5…v9
- Goals
  - Collect usage information of individual data flows
  - Accumulate packet and byte counter to reduce the size of the monitored data
- Approach
  - Each flow is represented by its 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

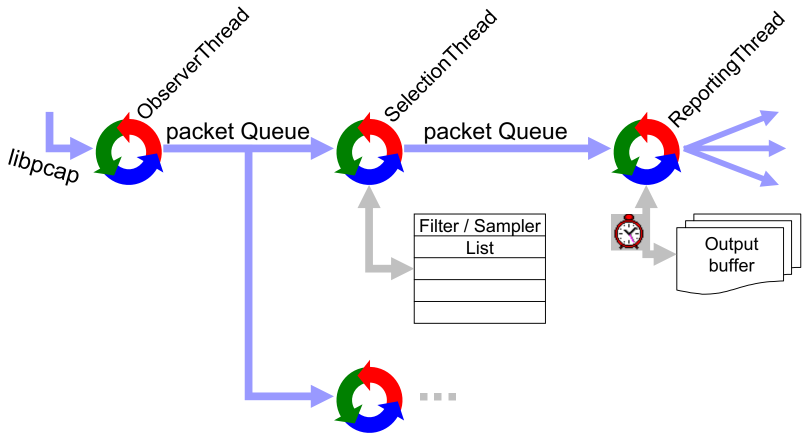
IPFIX – Applications II

- 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

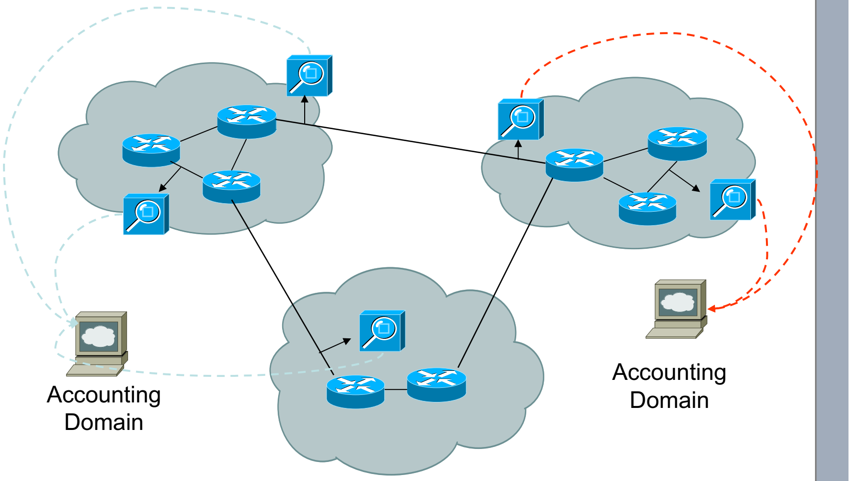
Implementation of PSAMP / IPFIX



Application Scenarios

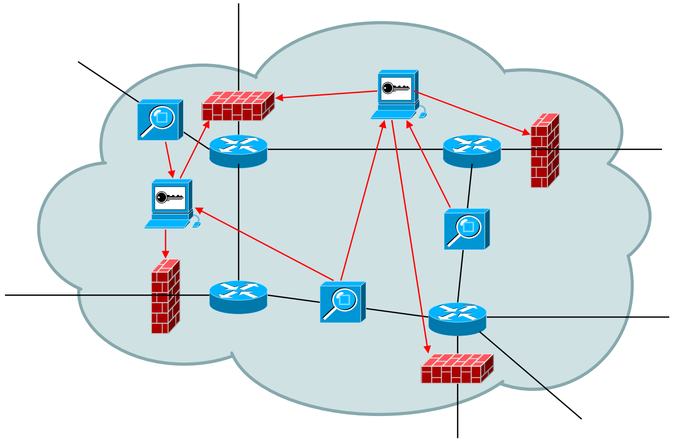
- Accounting and Charging
  - Accounting for statistical reasons
  - Accounting for charging
- Traffic engineering
  - Identification of primary traffic paths
  - Optimization of network parameters (e.g. routing parameters) for better network utilization
- Network Security
  - Detection of denial-of-service attacks
  - Forensic methods for post-intrusion analysis

Accounting and Charging



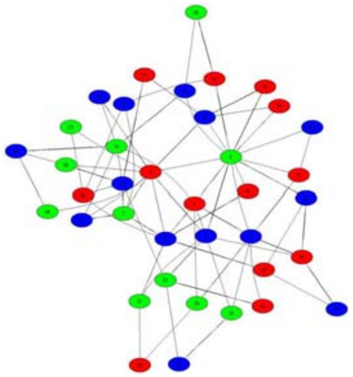
Network Security

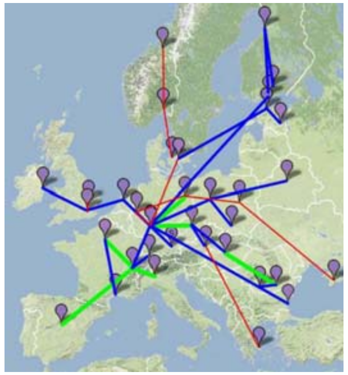
- Intrusion detection with automated firewall configuration



Project 2: Preview

- *Goal:* Generating Internet maps on
  - logical and
  - geographical levels





Project 2: Preview

- *Approach:* active measurements based on various
  - traceroute and
  - geolocation tools
- *Outline:*
  - Understanding and comparing different measurement approaches and their limitations
  - Investigating daily-life routing anomalies
  - Drawing graphs and maps based on measurement data
- *Benefits:*
  - Gaining knowledge on your personal access to the Internet as well as Internet routing in general
  - Getting in touch with load balancing, MPLS and IP-Anycast
  - Understanding and questioning current geolocation techniques
- **To be handed in by Mon, 17th Jan 2010, 10h**

Chair for Network Architectures and Services – Prof. Carle
Department for Computer Science
TU München

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

Technische Universität München

Chair for Network Architectures and Services – Prof. Carle
Department for Computer Science
TU München

Network Address Translation (NAT)

Technische Universität München

NAT: Network Address Translation

- Problem: shortage of IPv4 addresses
  - only 32bit address field
  - more and more devices
- Idea: local network uses just one IP address as far as outside world is concerned:
  - only one IP address for all devices needed from ISP
  - can change addresses of devices in local network without notifying outside world
  - can change ISP without changing local addresses
  - devices inside local net not explicitly addressable/visible by outside world (a security plus).

Address Space

- IP addresses are assigned by the Internet Assigned Numbers Authority (IANA)

- RFC 1918 directs IANA to reserve the following IPv4 address ranges for private networks
- 10.0.0.0 – 10.255.255.255
- 172.16.0.0 – 172.31.255.255
- 192.168.0.0 – 192.168.255.255

- The addresses may be used and reused by everyone
- Not routed in the public Internet
- Therefore a mechanism for translating addresses is needed

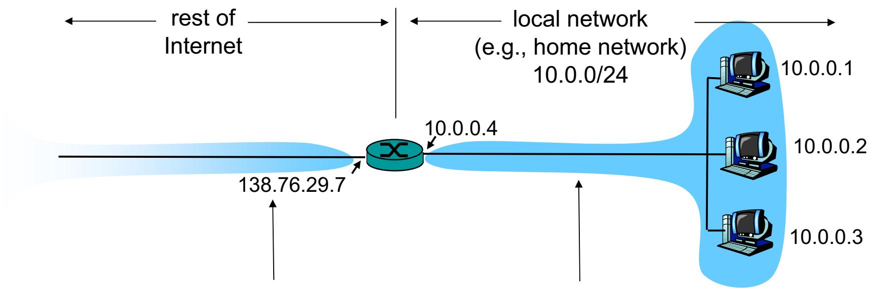
First approach – Network Address Translation

- Idea: only host communicating with the public Internet need a public address
  - Once a host connects to the Internet we need to allocate one
  - Communication inside the local network is not affected

- A small number of public addresses may be enough for a large number of private clients

- Only a subset of the private hosts can connect at the same time
  - not realistic anymore
  - We still need more than one public IP address

NAPT: 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

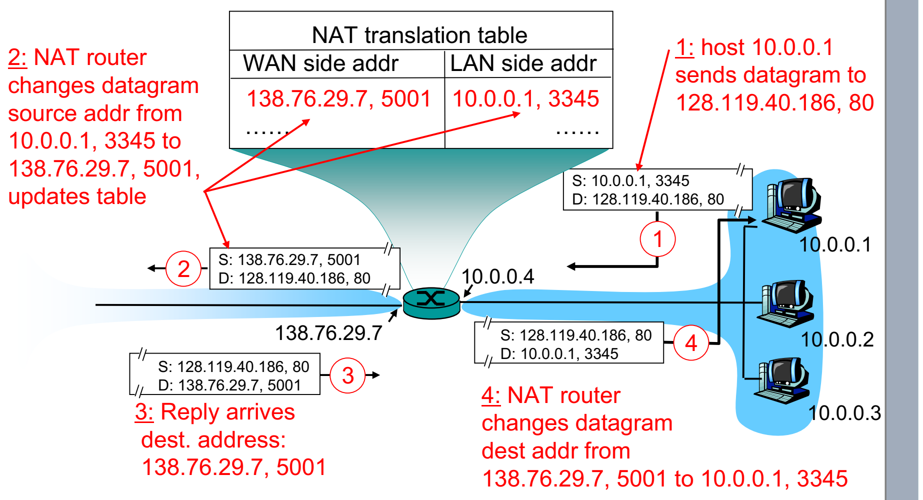
Implementation: NAT router must:

- *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 translation table	
WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
……	……

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

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

3: Reply arrives dest. address: 138.76.29.7, 5001

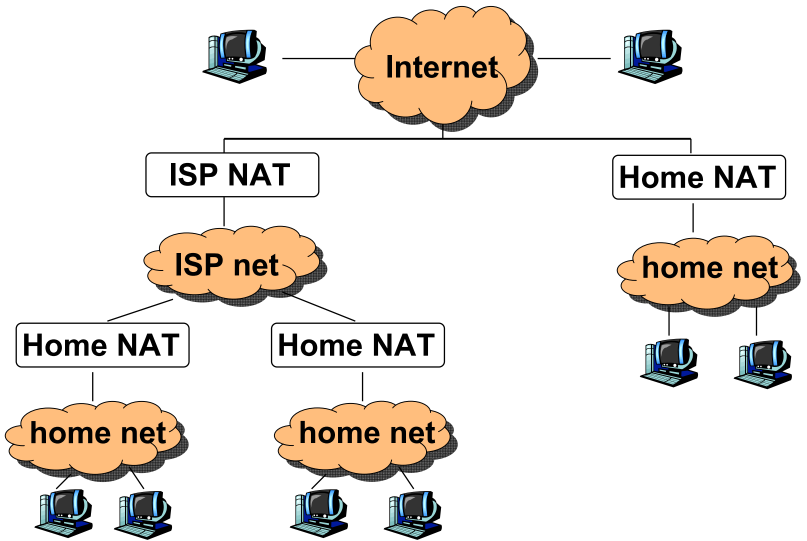
4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345

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 controversal:
  - 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

- When creating a new state, the NAT has to assign a new source port and IP address to the connection
- Binding covers context based packet translation
- **Port binding** describes the strategy a NAT uses for the assignment of a new external source port
  - source port can only be preserved if not already taken
- **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 (combination of IP address and port)
  - 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 source and destination 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

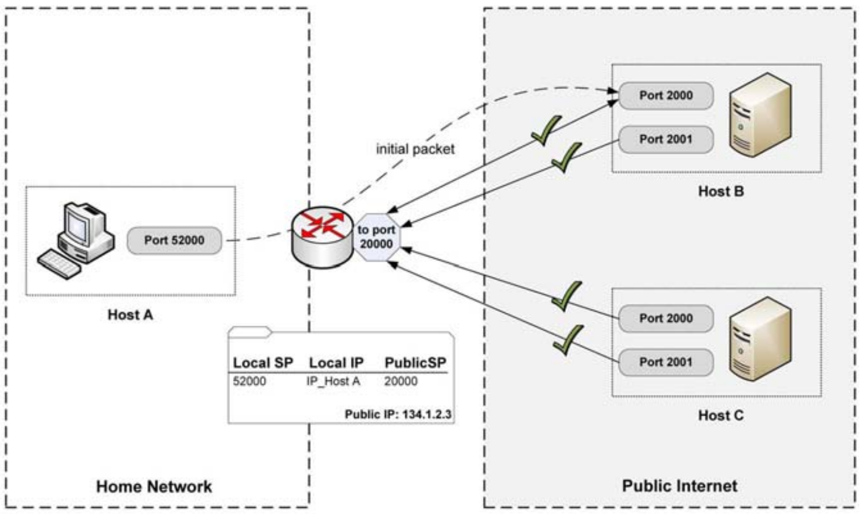
NAT Types

- With Binding and Filtering 4 NAT types can be defined (RFC 3489)
- 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

NAT Types

- With Binding and Filtering 4 NAT types can be defined (RFC 3489)
- **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

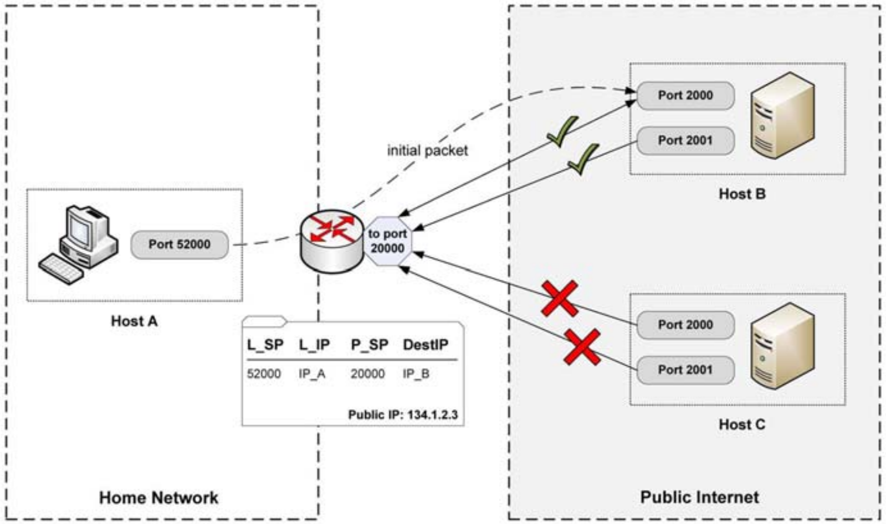
Full Cone NAT



NAT Types

- With Binding and Filtering 4 NAT types can be defined (RFC 3489)
- 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

Address Restricted Cone NAT



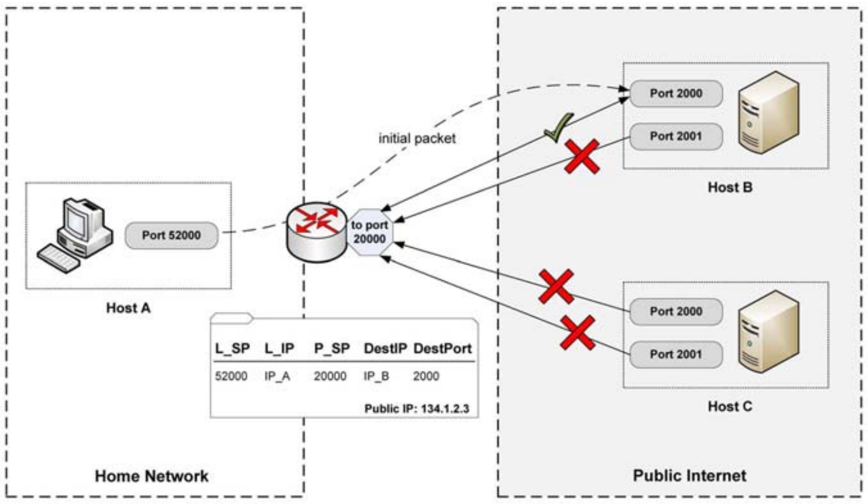
L_SP	L_IP	P_SP	DestIP
52000	IP_A	20000	IP_B

Public IP: 134.1.2.3

NAT Types

- With Binding and Filtering 4 NAT types can be defined (RFC 3489)
- 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

Port Address Restricted Cone NAT



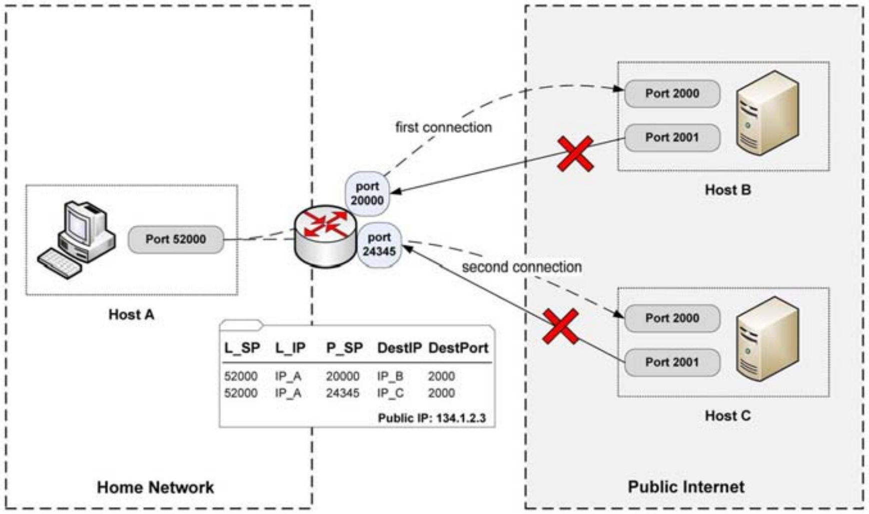
L_SP	L_IP	P_SP	DestIP	DestPort
52000	IP_A	20000	IP_B	2000

Public IP: 134.1.2.3

NAT Types

- With Binding and Filtering 4 NAT types can be defined (RFC 3489)
- 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**

Symmetric NAT



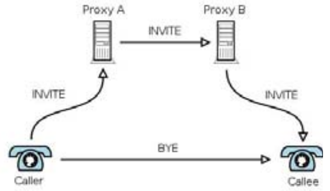
L_SP	L_IP	P_SP	DestIP	DestPort
52000	IP_A	20000	IP_B	2000
52000	IP_A	24345	IP_C	2000

Public IP: 134.1.2.3

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
```

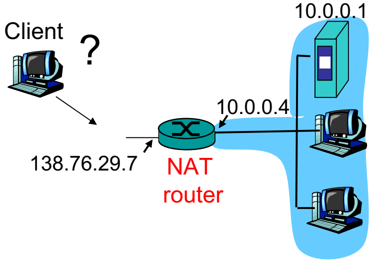
RTP-Session Specification (for 2nd channel)

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

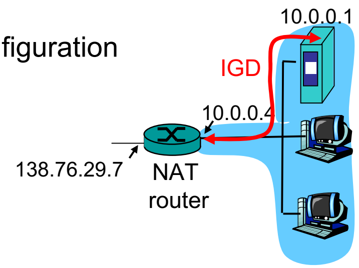
- Individual solutions
  - Explicit support by the NAT
    - static port forwarding, UPnP, NAT-PMP
  - NAT-behavior based approaches
    - dependent on knowledge about the NAT
    - hole punching using STUN (IETF - RFC 3489)
  - External Data-Relay
    - TURN (IETF - Draft)
- Frameworks integrating several 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)

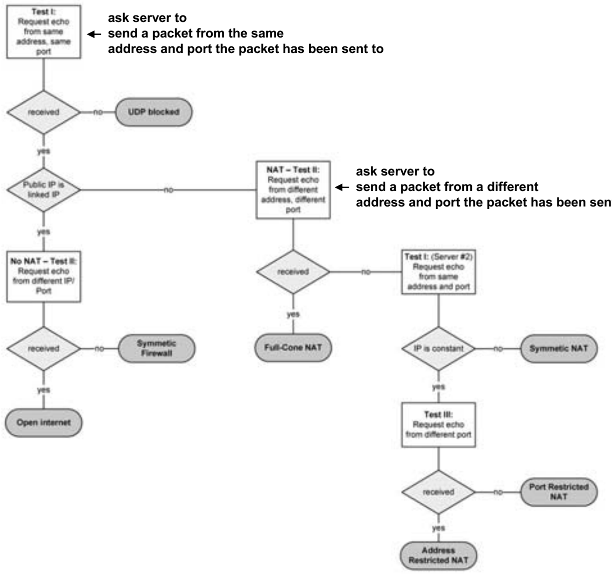


- Drawbacks
  - no security, evil applications can establish port forwarding entries
  - doesn‘t work with cascaded NATs

Behavior based (1): STUN

- Simple traversal of UDP through NAT (old) (RFC 3489)
  - Session Traversal Utilities for NAT (new) (RFC 5389)
- 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

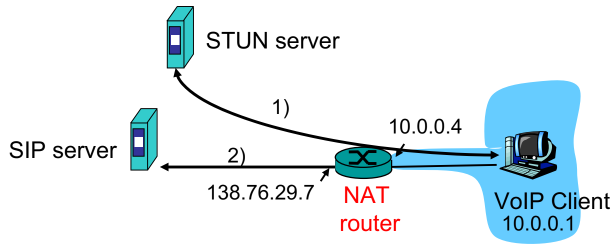


ask server to
send a packet from the same
address and port the packet has been sent to

ask server to
send a packet from a different
address and port the packet has been sent to

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

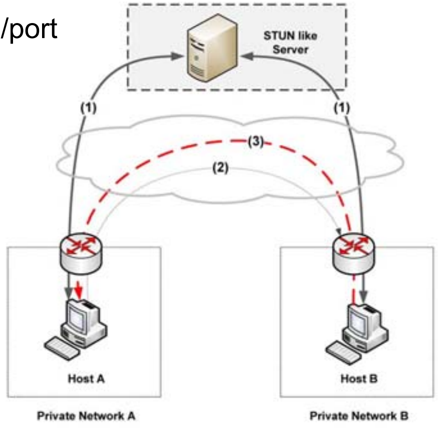
STUN and Hole Punching

- 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

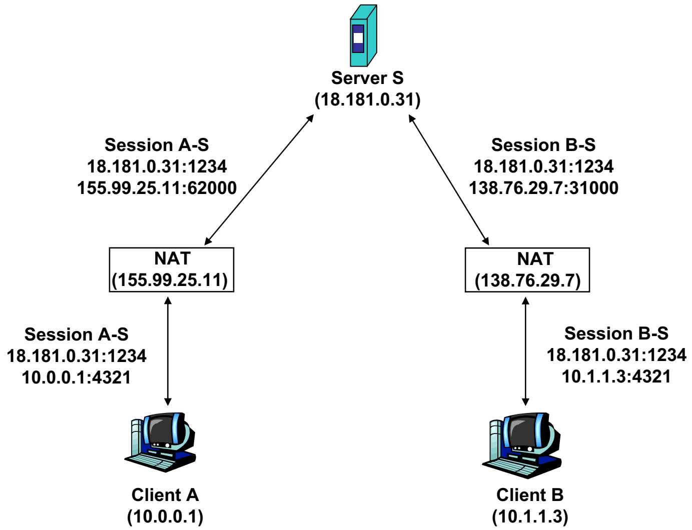
2) both hosts send packets towards the other host outgoing packet creates hole

3) establish connection. hole is created by first packet



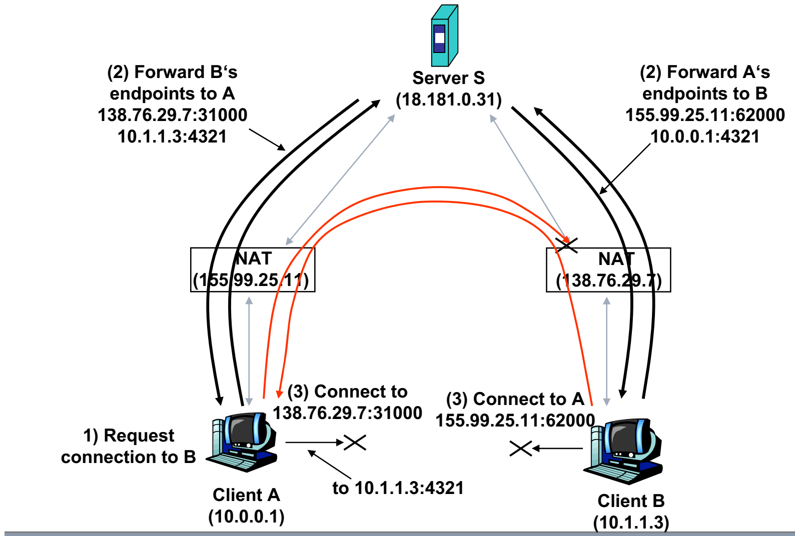
Hole Punching in detail

- ❑ Before hole punching



Hole Punching in detail

- ❑ Hole punching

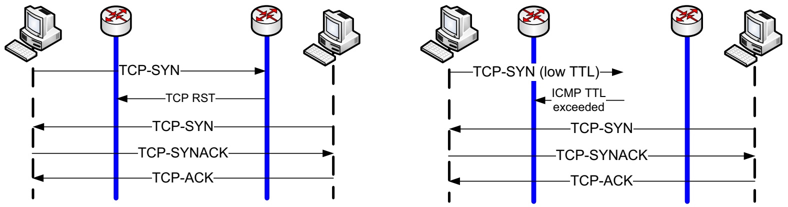


DIY Hole Punching: practical example

- ❑ 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

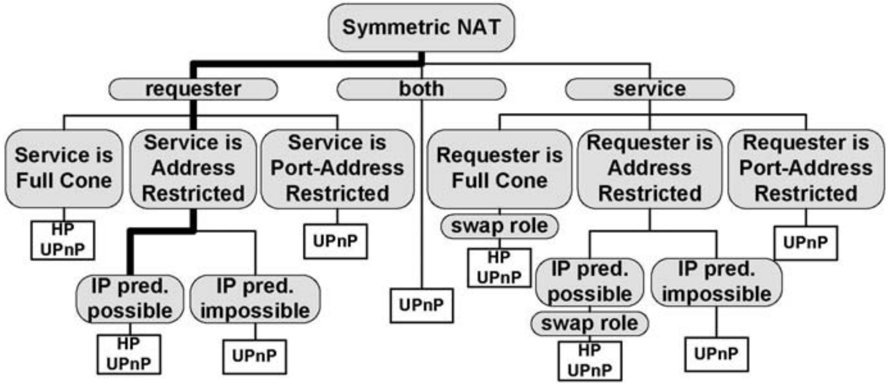
- ❑ Hole Punching not straight forward due to stateful design of TCP
  - ▪ 3-way handshake
  - ▪ Sequence numbers
  - ▪ ICMP packets may trigger RST packets
- ❑ Low/high TTL(Layer 3) of Hole-Punching packet
  - ▪ As implemented in STUNT (Cornell University)



- ❑ Bottom line: NAT is not standardized

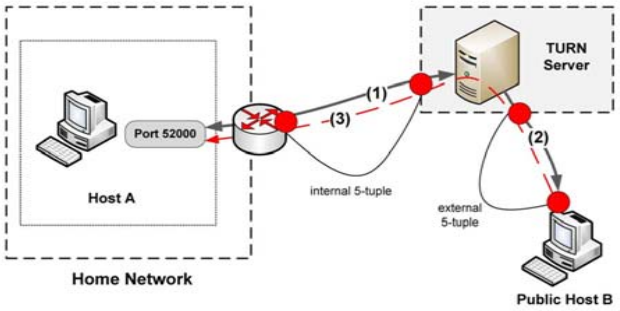
Symmetric NATs

- How can we traverse symmetric NATs
  - Endpoint dependent binding
    - hole punching in general only if port prediction is possible
  - Address and port restricted filtering



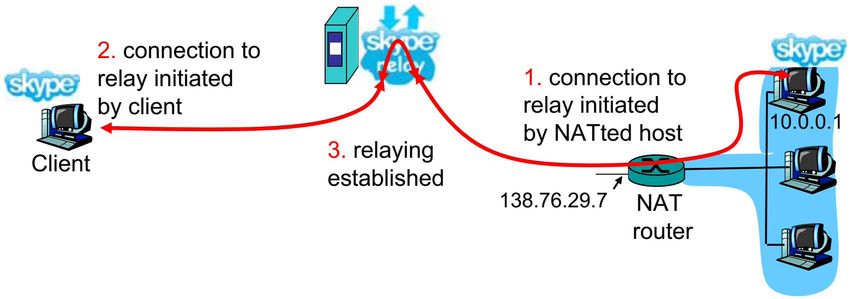
Data Relay (1)

- 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


2. connection to relay initiated by client
skype
Client
skype relay
3. relaying established
1. connection to relay initiated by NATted host
skype
10.0.0.1
138.76.29.7
NAT router

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

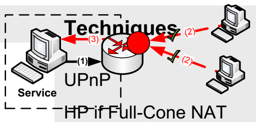
- http://nattest.net.in.tum.de

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 %

Service Categories for NAT-Traversal (TUM)

- Global Service-Provisioning (GSP)
  - Globally accessible public endpoint
  - Only the service host needs software support



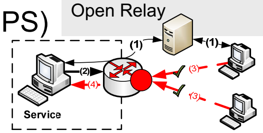
Techniques

UPnP

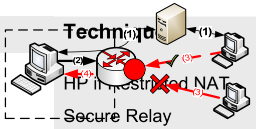
HP if Full-Cone NAT

Open Relay

- Service-Provisioning using Pre-Signaling (SPPS)
  - Pre-Signaling through Rendezvous-Point
  - No assumptions about NAT-Traversal techniques
  - Both hosts need software support



- Secure Service-Provisioning (SSP)
  - Extension for SPPS
  - Only authorized users can allocate mappings
  - Created mapping can only be accessed by the creator



Techniques

HP if Restricted NAT

Secure Relay

- ALG Service-Provisioning (ALG-SP)
  - Explicit support for Layer 7 protocols (SIP-VoIP)

New approach (TUM)

- 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 communication across NAT is 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
  - 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“

Chair for Network Architectures and Services – Prof. Carle
Department for Computer Science
TU München

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

Signalling - Introduction

Design principles:

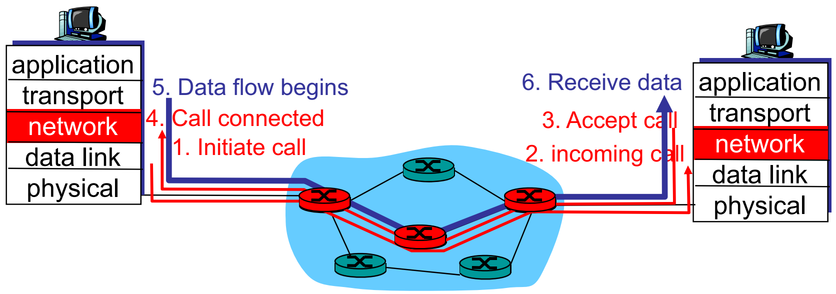
- network virtualization: overlays
- separation of data, control
- hard state versus soft state

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

Chair for Network Architectures and Services – Prof. Carle
Department for Computer Science
TU München

H.323

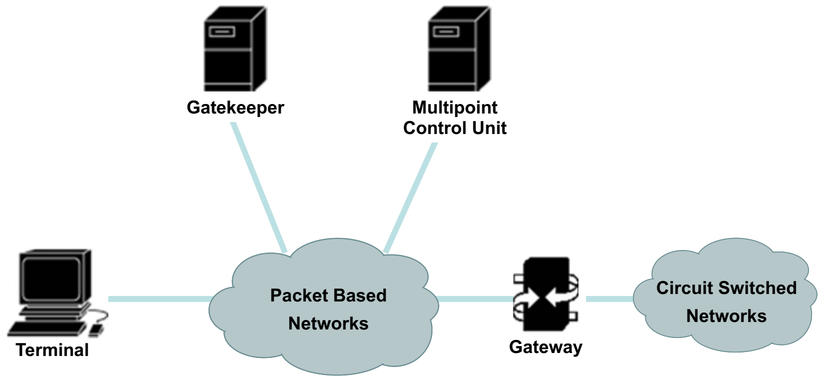
Technische Universität München

What is H.323?

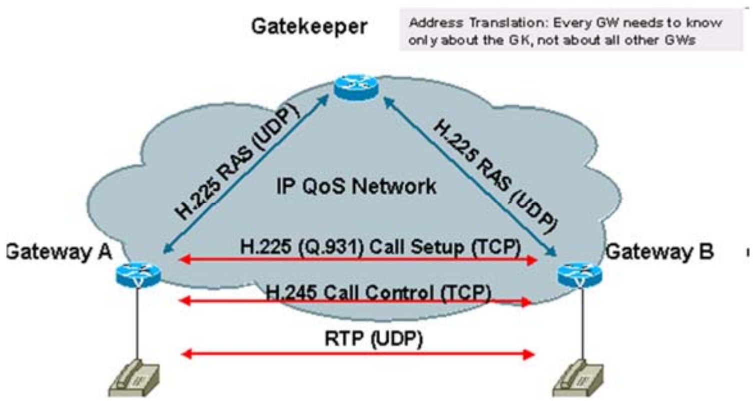
- ITU-T Recommendation H.323 Version 4

  Describes terminals and other entities that provide multimedia communications services over Packet Based Networks (PBN) which may not provide a guaranteed Quality of Service. H.323 entities may provide real-time audio, video and/or data communications.
- H.323 framework defines:
  - Call establishment and teardown.
  - Audio visual or multimedia conferencing.

H.323 Components



H.225 and H.245


Gatekeeper
Address Translation: Every GW needs to know only about the GK, not about all other GWs
IP QoS Network
H.225 RAS (UDP)
H.225 RAS (UDP)
Gateway A
Gateway B
H.225 (Q.931) Call Setup (TCP)
H.245 Call Control (TCP)
RTP (UDP)

H.323 Terminals

- H.323 terminals are client endpoints that must support:
  - H.225 call control signaling.
    - Call-control messages to gatekeeper (or directly to terminal), transport via TCP, port 1720
    - RAS: Registration, Admission, Status messages for address resolution and admission control
  - H.245 control channel signaling.
    - End-to-end messages between H.323 terminals.
    - Initiating and closing logical channels
    - Exchange of information on transmission capacity
  - RTP/RTCP protocols for media packets.
  - Audio codecs.
  - Video codecs support is optional.

H.323 is an "Umbrella" Specification

Media

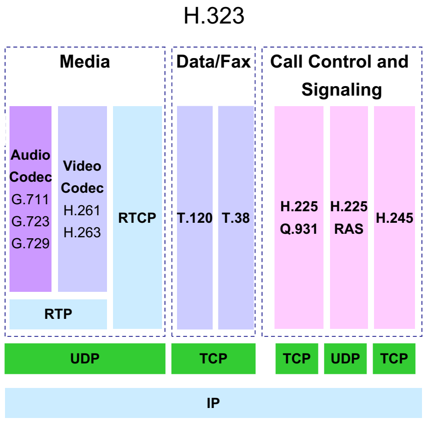
- H.261 and H.263 – Video codecs.
- G.711, G.723, G.729 – Audio codecs.
- RTP/RTCP – Media transport and control protocol.

Data/Fax

- T.120 – Data conferencing.
- T.38 – Fax.

Call Control and Signaling

- H.245 - Capabilities advertisement, media channel establishment, and conference control.
- H.225
- Q.931 - call signaling and call setup.
- RAS – registration, admission control, status messages with a gatekeeper.



Chair for Network Architectures and Services – Prof. Carle
Department for Computer Science
TU München

SIP

Credits in addition to
Jim Kurose and Keith Ross:

Julie Chan, Vovida Networks.
Milind Nimesh, Columbia University
Christian Hoene, University of Tübingen

Technische Universität München

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

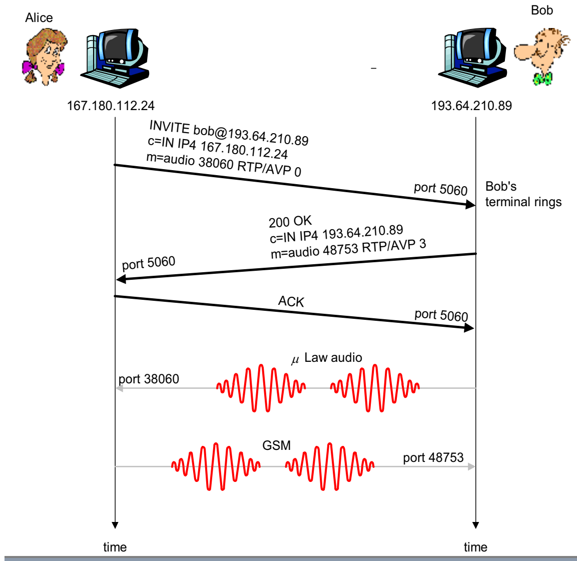
SIP

- 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 for TCP and UDP, and 5061 for TSL-over-TCP.

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
```

Notes:

- HTTP message syntax
- sdp = session description protocol
- Call-ID is unique for every call.

- 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)
- Session Description Protocol (SDP), RFC 4566 is a format for describing streaming media initialization parameters.

c=connection information
m= (media name and transport address)

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 server

- 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:

- Used by a UA to indicate its current IP address and the URLs for which it would like to receive calls.

```
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.
  - each proxy includes its address as VIA address
- 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

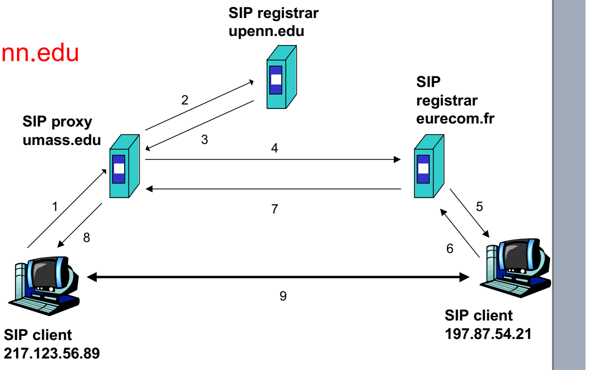
Example

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

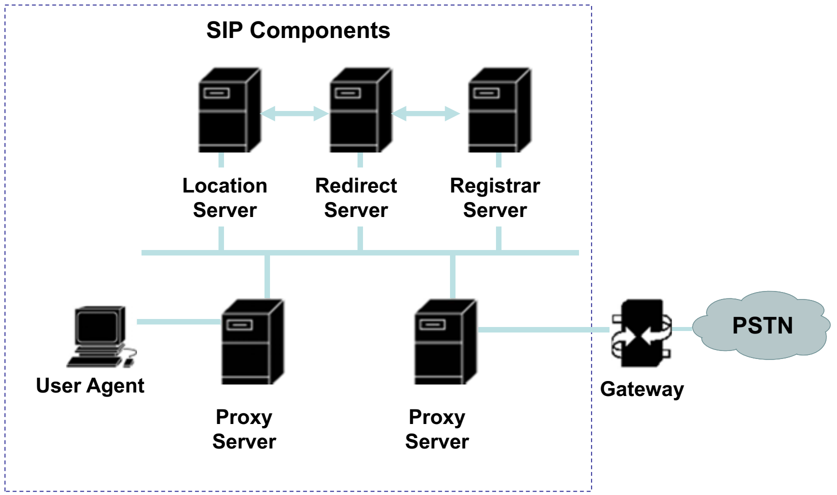
(9) media sent directly between clients.

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:

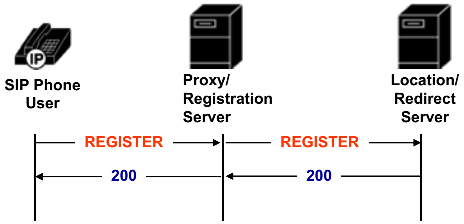
```
----------------------------------------------------------------
                        SIP Header
----------------------------------------------------------------
INVITE sip:5120@192.168.36.180 SIP/2.0
Via: SIP/2.0/UDP 192.168.6.21:5060
From: sip:5121@192.168.6.21
To: <sip:5120@192.168.36.180>
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 Phone User

Proxy/ Registration Server

Location/ Redirect Server

REGISTER

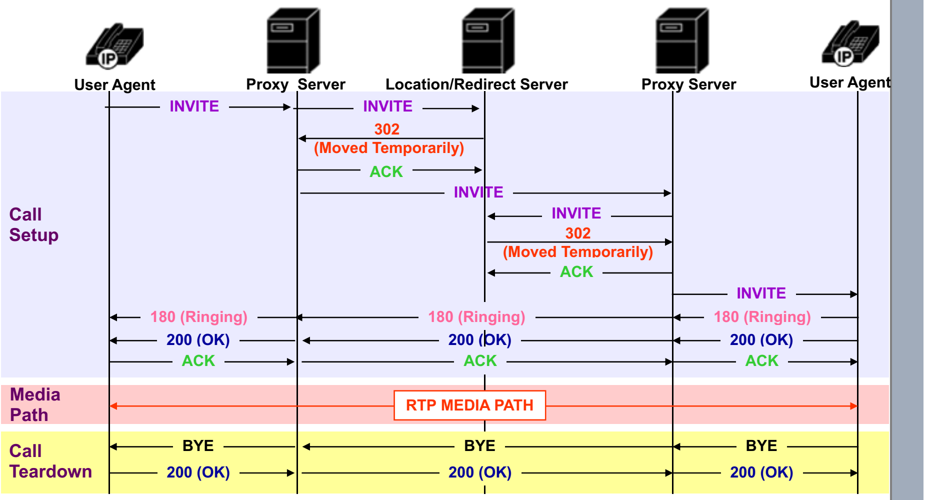
REGISTER

200

200

SIP Messages:
REGISTER – Registers the address listed in the To header field.
200 – OK.

Simplified SIP Call Setup and Teardown



User Agent

Proxy Server

Location/Redirect Server

Proxy Server

User Agent

Call Setup

INVITE

INVITE

302 (Moved Temporarily)

ACK

INVITE

INVITE

302 (Moved Temporarily)

ACK

INVITE

180 (Ringing)

180 (Ringing)

180 (Ringing)

200 (OK)

200 (OK)

200 (OK)

ACK

ACK

ACK

Media Path

RTP MEDIA PATH

Call Teardown

BYE

BYE

BYE

200 (OK)

200 (OK)

200 (OK)

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/

Chair for Network Architectures and Services – Prof. Carle
Department for Computer Science
TU München

Location Information and IETF GeoPriv Working Group

credits:
Milind Nimesh, Columbia University

Technische Universität München

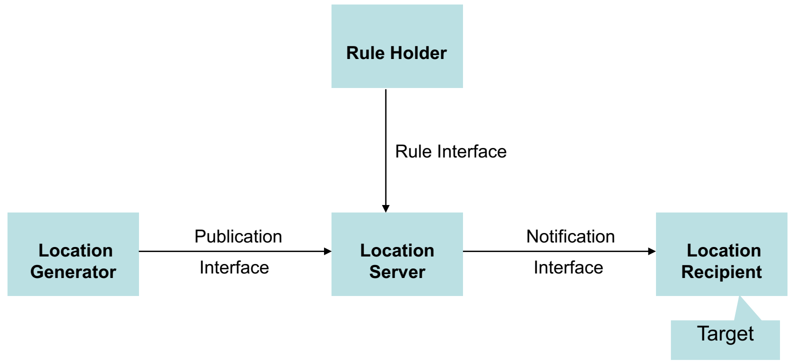
Location Information

- 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



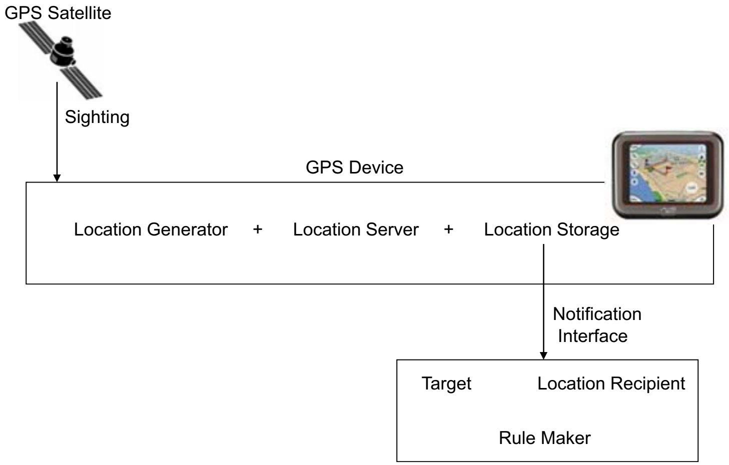
Geopriv Terminology

- 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 Satellite

Sighting

GPS Device

Location Generator + Location Server + Location Storage

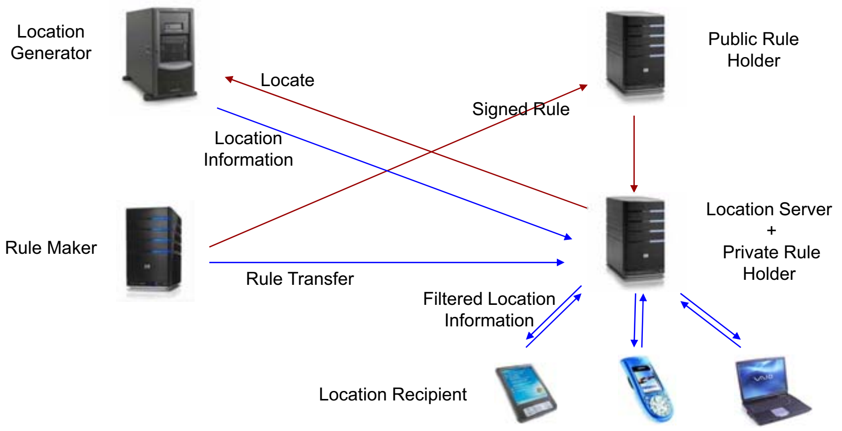
Notification Interface

Target Location Recipient

Rule Maker

GPS Device with Internal Computing Power: Closed System

Scenarios



Location Generator

Public Rule Holder

Locate

Signed Rule

Location Information

Location Server + Private Rule Holder

Rule Maker

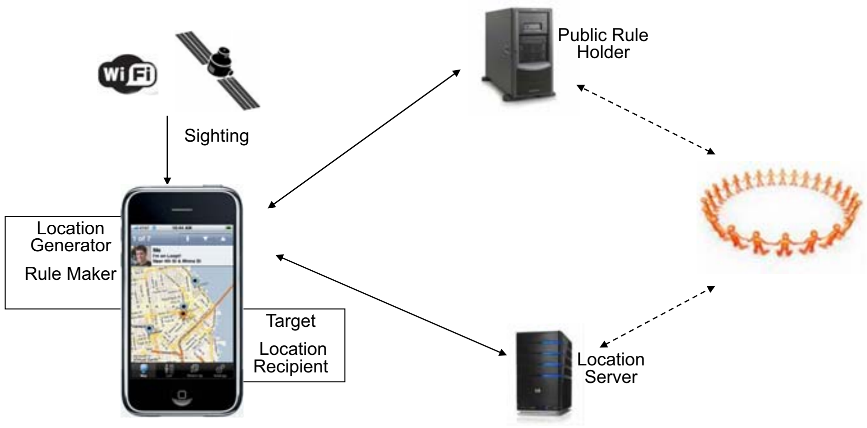
Rule Transfer

Filtered Location Information

Location Recipient

Mobile Communities and Location-Based Services

Applications: Social Networking



Location Configuration

Configuring the location of a device, using means such as:

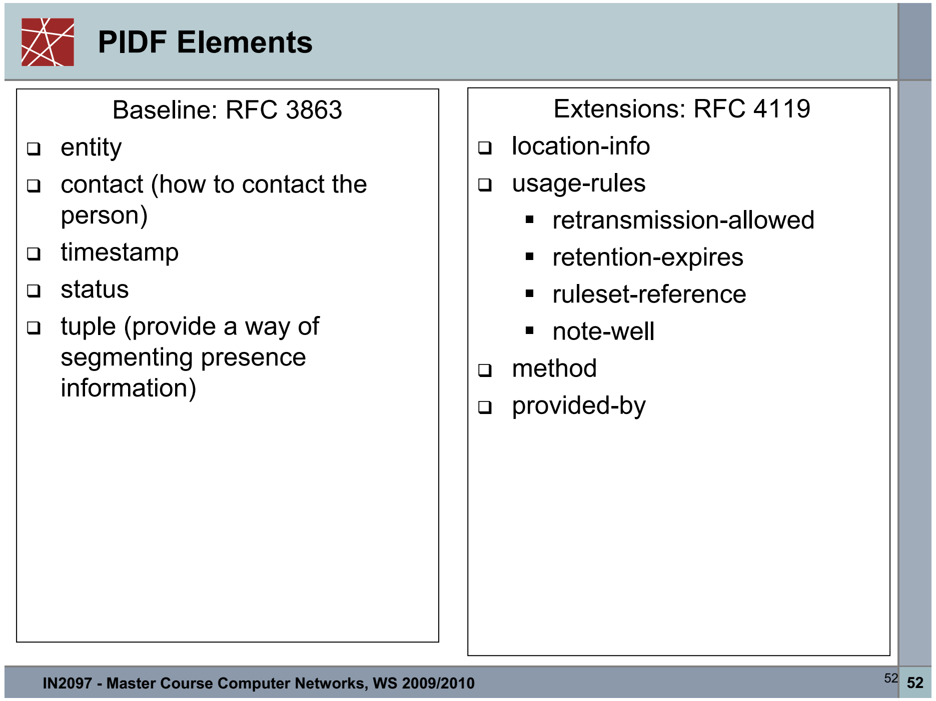
- ❑ DHCP extensions
  - RFC3825 : Option 123, geo-coordinate based location
  - RFC4776 : Option 99, civic address
- ❑ Link Layer Discovery Protocol - Media Endpoint Discovery
  - LLDP - a vendor-neutral Layer 2 protocol that allows a network device to advertise its identity and capabilities on the local 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



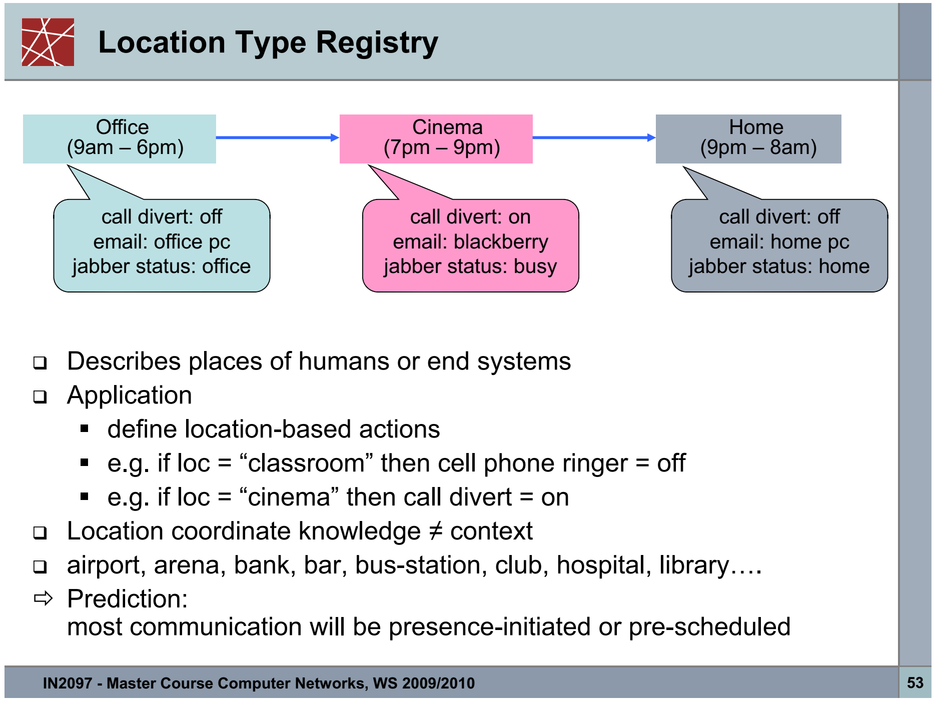
PIDF Elements

Baseline: RFC 3863

- ❑ entity
- ❑ contact (how to contact the person)
- ❑ timestamp
- ❑ status
- ❑ tuple (provide a way of segmenting presence information)

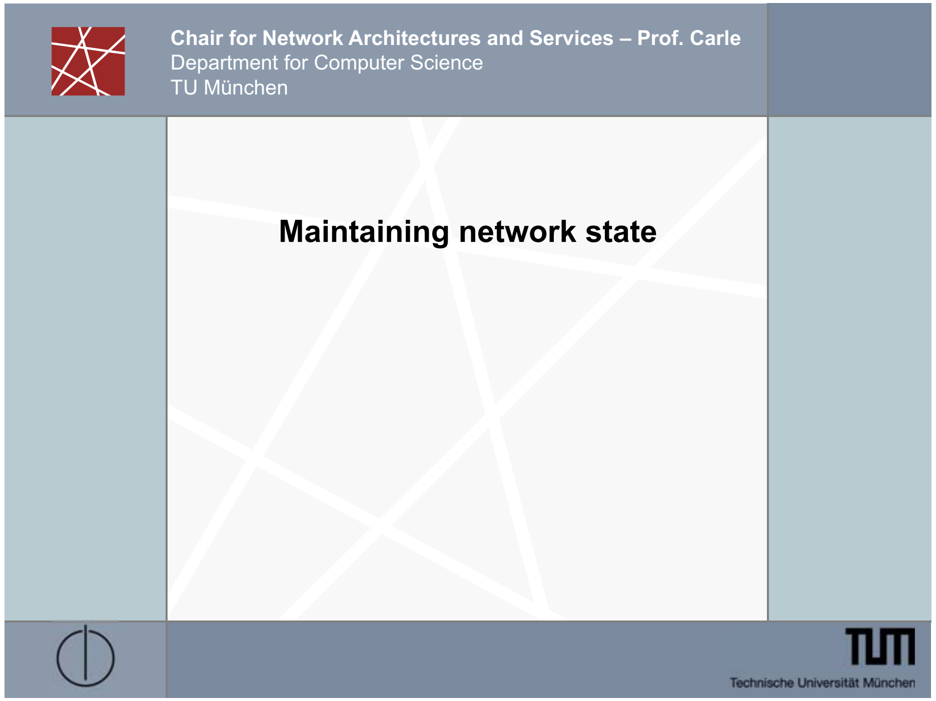
Extensions: RFC 4119

- ❑ location-info
- ❑ usage-rules
  - ▪ retransmission-allowed
  - ▪ retention-expires
  - ▪ ruleset-reference
  - ▪ note-well
- ❑ method
- ❑ provided-by



Location Type Registry

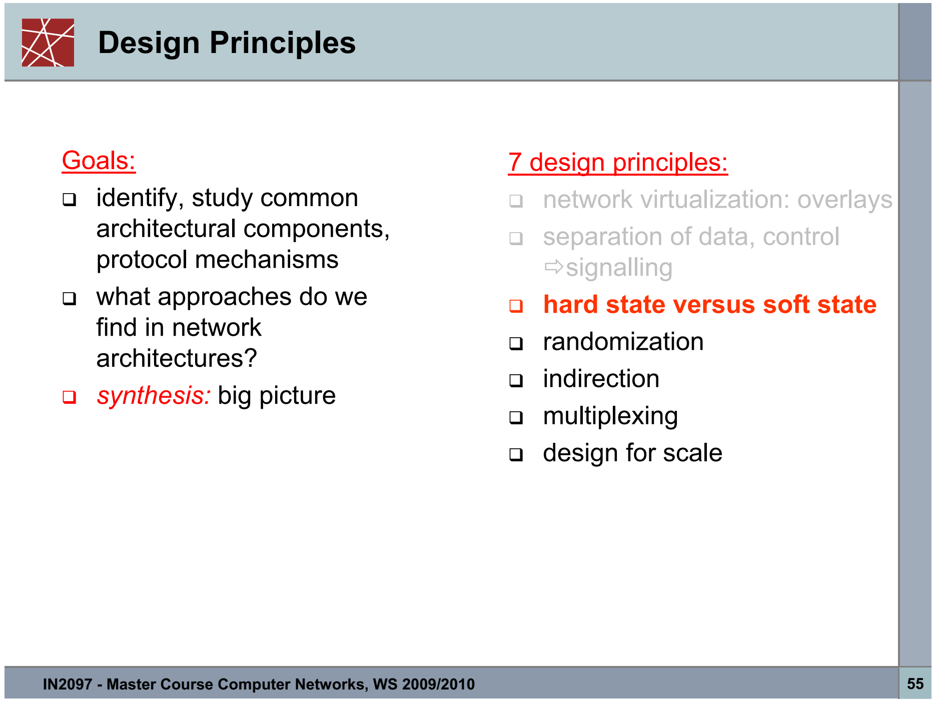
- ❑ Describes places of humans or end systems
- ❑ Application
  - ▪ define location-based actions
  - ▪ e.g. if loc = “classroom” then cell phone ringer = off
  - ▪ e.g. if loc = “cinema” then call divert = on
- ❑ Location coordinate knowledge ≠ context
- ❑ airport, arena, bank, bar, bus-station, club, hospital, library….
- ⇨ Prediction:
  most communication will be presence-initiated or pre-scheduled



Chair for Network Architectures and Services – Prof. Carle
Department for Computer Science
TU München

Maintaining network state

Technische Universität München



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

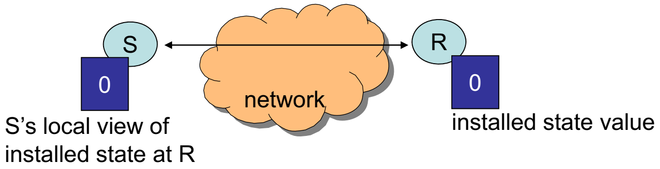
- state *installed* by receiver on receipt of setup (trigger) message from sender (typically, an endpoint)
  - sender also sends periodic *refresh* message: indicating receiver should continue to maintain state
- state *removed* by receiver via timeout, in absence of refresh message from sender
- default assumption: state becomes invalid unless refreshed
  - in practice: explicit state removal (*teardown*) messages also used
- examples:
  - RSVP, RTP/RTCP, IGMP

State: senders, receivers

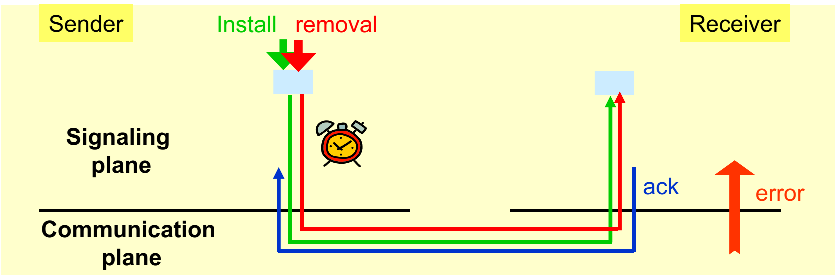
- sender: network node that *(re)generates* signaling (control) messages to install, keep-alive, remove state from other nodes
- receiver: node that creates, maintains, removes state based on signaling messages *received* from sender

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

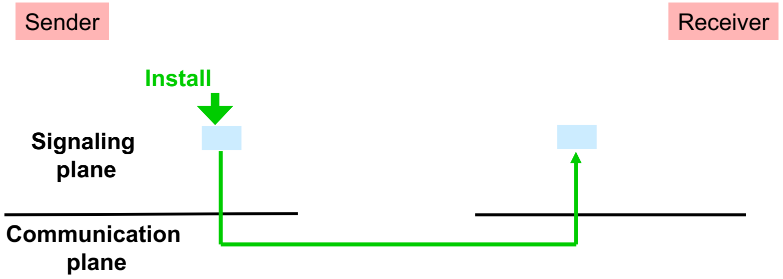

S
0
S’s local view of installed state at R
network
R
0
installed state value

Hard-state signaling


Sender
Install
removal
Receiver
Signaling plane
ack
error
Communication plane

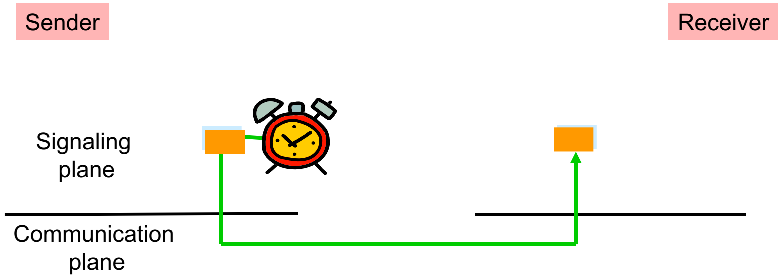
- reliable signaling
- state removal by request
- requires additional error handling
  - e.g., sender failure

Soft-state signaling


Sender
Receiver
Install
Signaling plane
Communication plane

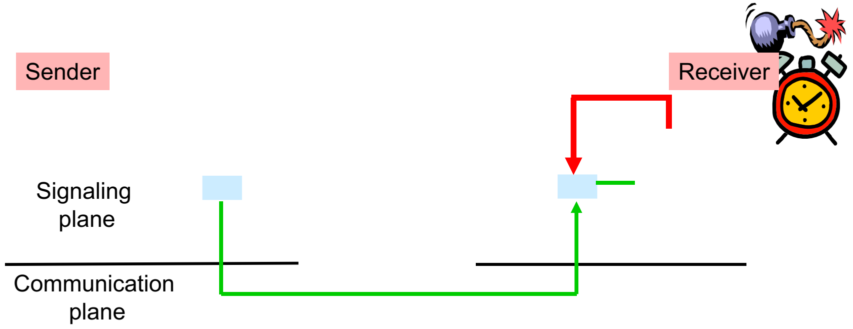
- best effort signaling

Soft-state signaling


Sender
Receiver
Signaling plane
Communication plane

- 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

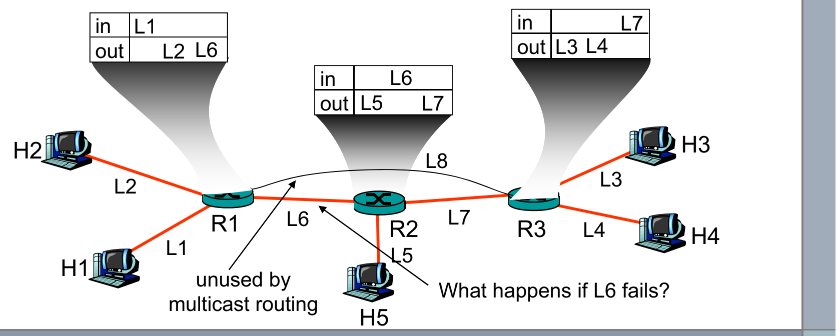
Soft-state: claims

- “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?

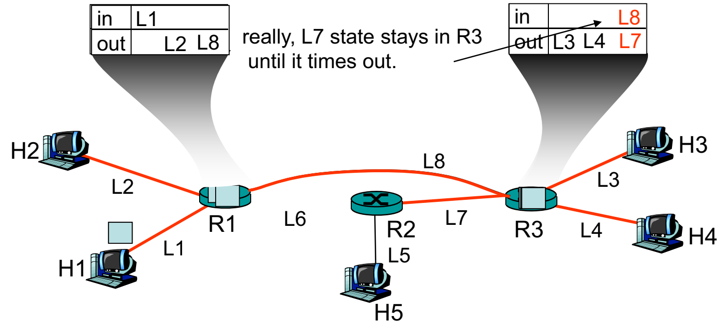
Soft-state: “easy” handling of changes

- 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



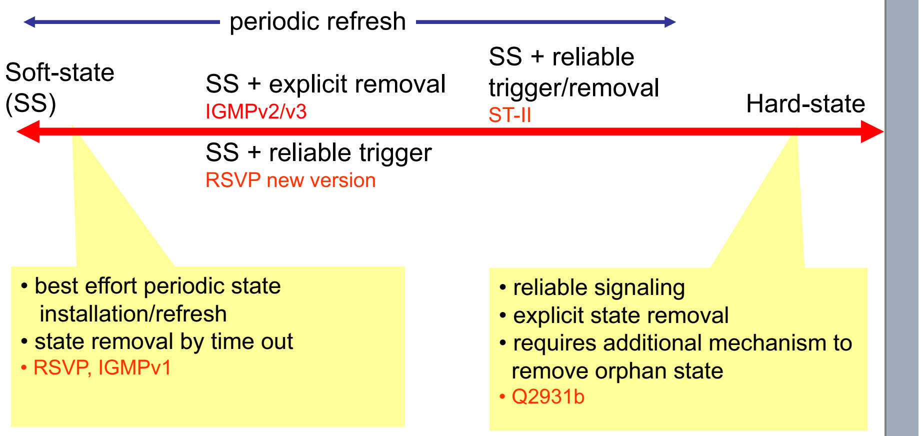
Soft-state: “easy” handling of changes

- “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



- 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

Chair for Network Architectures and Services – Prof. Carle
Department for Computer Science
TU München

Master Course Computer Networks (IN2097)

Introduction to

Network Resilience

Dr. Ali Fessi
Dr. Nils Kammenhuber
Prof. Dr.-Ing. Georg Carle

Technische Universität München

Overview

I. Terminology

II. Challenges in the current Internet

III. Resilience Mechanisms

Overview

I. **Terminology**

II. Challenges in the current Internet

III. Resilience Mechanisms

Terminolgy - Overview

1. The “*fault* ➔ *error* ➔ *failure*” chain
2. Fault tolerance
3. Resilience
4. Dependability
5. Security
6. Availability vs. Reliability

The “*fault* ➔ *error* ➔ *failure*” chain

- *Service*:
  - Sequence of the system‘s external state
- *Correct service* is delivered when the service implements the system function
- Definition
  - A *service failure*, or simply *failure,* is an event that occurs when the delivered service deviates from *correct service*
  - i.e., at least one external state of the system deviates from the correct service state
  - (de: Ausfall)

The “*fault* ➔ *error* ➔ *failure*” chain

- Definition
  - The deviation of an external state of the system from the correct service state is called an *error*
  - Thus, an error is the part of the total state of the system that may lead to its subsequent failure
  - (de: Defekt)
- Definition
  - The cause of an error (adjuged or hypothesized) is called a *fault*
  - (de: Fehler)

☞ “*fault* ➔ *error* ➔ *failure*”

Fault Tolerance

- Definition
  - A system is fault-tolerant if it can mask the presence of *faults* in the system by using *redundancy*
- Redundancy means
  1. *Replication* of the same object (software or hardware) or
  2. *Diversity*
     - Design or implementation
     - Hardware or software

Resilience

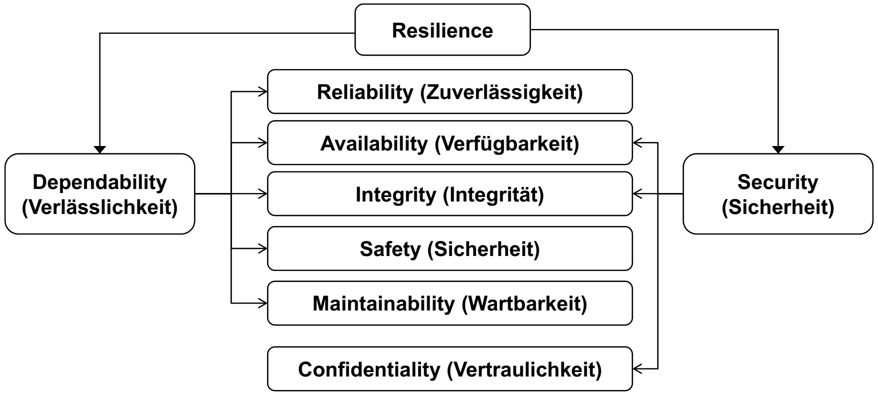
- Origin
  - Latin verb: “resilire” ~ jump back
- Resilience definition in different fields
  - Physics
    - A material’s property of being able to recover to a normal state after a deformation resulting from external forces;
  - Ecology
    - Moving from a stability domain to another under the influence of disturbance;
  - Psychology and psychiatry
    - Living and developing successfully when facing adversity;
  - Business
    - the capacity to reinvent a business model before circumstances force to;

Resilience

- Definition:
  - “Resilience is the persistence of *dependability* when facing *changes*.”

J.-C. Laprie. “From Dependability to Resilience”. In 38th International Conference On Dependable Systems and Networks. IEEE/IFIP, 2008.

- Changes can be particularly *attacks*



Resilience

Dependability (Verlässlichkeit)

Security (Sicherheit)

Reliability (Zuverlässigkeit)

Availability (Verfügbarkeit)

Integrity (Integrität)

Safety (Sicherheit)

Maintainability (Wartbarkeit)

Confidentiality (Vertraulichkeit)

Dependability Attributes

- Availability
  - Readiness for correct service
- Reliability
  - Continuity of correct service
- Safety
  - Absence of catastrophic consequences on the user(s) and the environment
- Integrity
  - Absence of improper system alterations
- Maintainability
  - Ability to undergo repair and modification

Security Attributes

- "CIA" model
  - Confidentiality, Integrity, Availability
- Confidentiality
  - Absence of unauthorized disclosure of information
- Availability
  - Readiness for correct service
- Integrity
  - Absence of improper system alterations
- Notes:
  - CIA model actually not sufficient to describe "security"
  - "Security" addresses all kind of possible attacks which may lead to the deviation from correct service

Reliability vs. Availability

- The reliability of a unit at a point of time t is the probability that the unit is operational until t

  $R(t) = Pr$ [unit is operating until t]

- The availability of a unit at a point of time t is the probability that the unit is operational at t

  $A(t) = Pr$ [unit is operating at t]

MTTF & MTTR

- Mean Time To Failure (MTTF)
  - Mean time between
    - Point of time when a unit is put into operation
    - Point of time when the unit fails for the next time
- Mean Time To Repair (MTTR)
  - Mean time between
    - Point of time when a unit fails
    - Point of time when the unit is put into operation again
- This results into an average availability

$$A_{avg} = \frac{MTTF}{MTTF + MTTR}$$

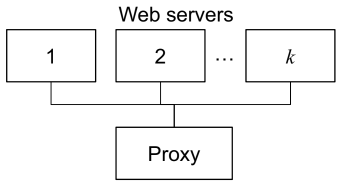
Examples

- DNS lookup (stateless service)
  - MTTF: 30 min
  - MTTR: 1 ms
  - A_{avg} = 0.998

☞ One can achieve
  - high availability
  - with low reliability (low MTTF)
  - if MTTR is sufficiently low

- Conference bridge (statefull service)
  - Each time, the bridge fails, participants need to re-dial
  - Even if MTTR is sufficiently low, it has to be guaranteed that the MTTF is sufficiently high to assure service quality

Examples



$$R_{system}(t) = R_{proxy}(t) \cdot R_{webserver\ pool}(t)$$

$$R_{webserver\ pool}(t) = 1-(1-R_{webserver}(t))^k$$

❑ Same holds for the availability

$$A_{system}(t) = A_{proxy}(t) \cdot A_{webserver\ pool}(t)$$

$$A_{webserver\ pool}(t) = 1-(1-A_{webserver}(t))^k$$

Overview

I. Terminology

II. Challenges in the current Internet

III. Resilience Mechanisms

Challenges in the current Internet

1. Topology Failures
2. Overload
3. Lack of Integrity
4. Software Faults
5. Domino Effects

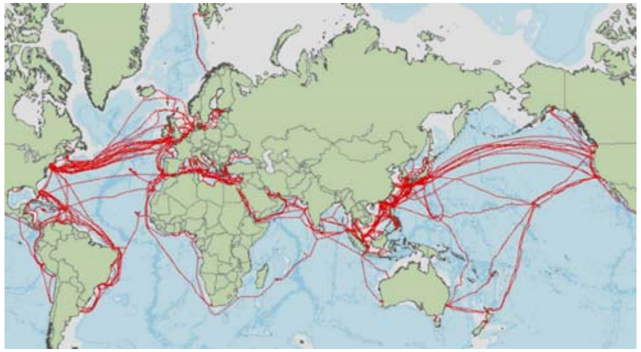
Challenges in the current Internet

1. **Topology Failures**
2. Overload
3. Lack of Integrity
4. Software Faults
5. Domino Effects

Topology Failures

- ❑ Failures in the "network graph"
- ❑ Network graph
  - Physical topology
  - Logical topology including service dependencies, e.g., DNS
    - ☞ *Dependency graphs*

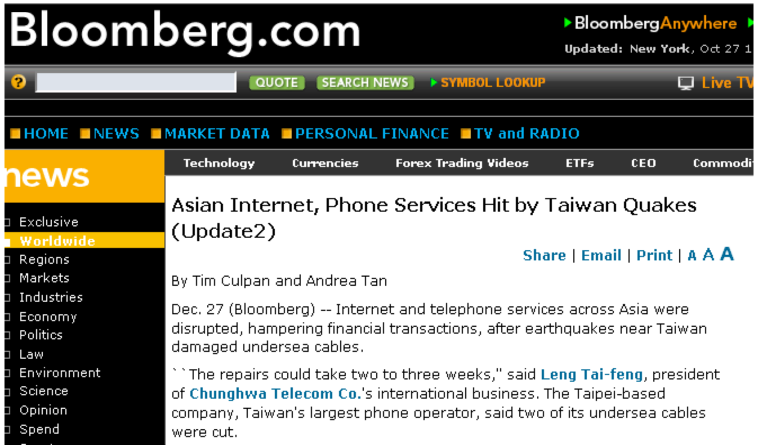
Topology Failures; Sub-Marine Cables



- ❑ ~99% of inter-continental Internet traffic (less than 1% using satellites)
- ❑ High redundant
- ❑ But vulnerable to
  - Fishing and anchoring (70% of sub-marine cable failures)
  - Natural disasters (12%)
  - Cable theft

Submarine Cables; Natural Disasters

- ❑ Hengchun earthquake (December 2006)



Asian Internet, Phone Services Hit by Taiwan Quakes (Update2)

By Tim Culpan and Andrea Tan

Dec. 27 (Bloomberg) -- Internet and telephone services across Asia were disrupted, hampering financial transactions, after earthquakes near Taiwan damaged undersea cables.

``The repairs could take two to three weeks,'' said Leng Tai-feng, president of Chunghwa Telecom Co.'s international business. The Taipei-based company, Taiwan's largest phone operator, said two of its undersea cables were cut.

Submarine Cables; Natural Disasters

❑ Hengchun earthquake (December 2006)

❑ Impact

- Affected countries: China, Taiwan, Hong Kong, Philippines
- China's Internet connectivity reduced by 70%
- Hong Kong's Internet access completely disabled

❑ Recovery

- BGP automatic re-routing helped to reduce disconnectivity
- But resulted into congested links
- Manual BGP policy changes + switch port re-configuration were necessary
- Hong Kong's Internet users were still experiencing slow Internet connections 5 days after the earthquake

Submarine Cables; Failures in the Mediterranean Sea

❑ In Jan. + Feb. 2008, 3 successive events

❑ Impact

- Affected countries: Egypt, Iran, India and a number of other middle east countries
- Disruption of
  - 70% in Egypt
  - 60% in India

Submarine Cables; Cable Theft

❑ In March 2007, pirates stole an 11 kilometers section of the submarine cable connecting Thailand, Vietnam and Hong Kong,

❑ Impact: significant downgrade in Internet speed in Vietnam.

❑ Intention: The thieves wanted to sell 100 tons of cable as scrap.

Topology Failures; Routing

❑ Failures in the IP topology graph

- Failures of routers (nodes)
- Failure of links between routers

❑ Failure of links between routers generally caused by disconnection at lower layers

❑ Failure of routers

- DoS attacks
- Failures due to software bugs
- Examples of reported bugs
  - Vulnerability to too long AS (BGP Autonomous Systems) paths
  - Long passwords to login to the router
  - Overflow of connection tables in some commercial firewalls

Topological Failures; Routing

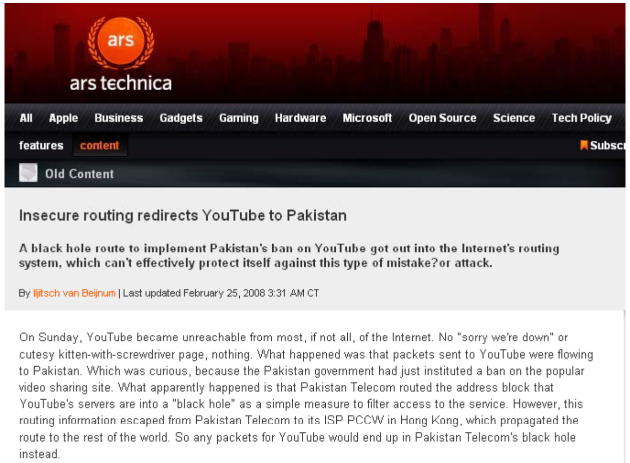
❑ Time to Recovery

- Intra-domain routing (OSPF, RIP, IS-IS, EIGRP): up to several 100ms
- Inter-domain routing (BGP): up to several minutes

Topological Failures; Routing

❑ Other reasons

- Misconfiguration which leads to false modification of the Internet topology



Insecure routing redirects YouTube to Pakistan

A black hole route to implement Pakistan's ban on YouTube got out into the Internet's routing system, which can't effectively protect itself against this type of mistake?or attack.

By Iljitsch van Beijnum | Last updated February 25, 2008 3:31 AM CT

On Sunday, YouTube became unreachable from most, if not all, of the Internet. No "sorry we're down" or cutesy kitten-with-screwdriver page, nothing. What happened was that packets sent to YouTube were flowing to Pakistan. Which was curious, because the Pakistan government had just instituted a ban on the popular video sharing site. What apparently happened is that Pakistan Telecom routed the address block that YouTube's servers are into a "black hole" as a simple measure to filter access to the service. However, this routing information escaped from Pakistan Telecom to its ISP PCCW in Hong Kong, which propagated the route to the rest of the world. So any packets for YouTube would end up in Pakistan Telecom's black hole instead.

Challenges in the current Internet

1. Topology Failures
2. **Overload**
3. Lack of Integrity
4. Software Faults
5. Domino Effects

Overload

❑ Topology failures are binary (link or node is up or down)

❑ But equipment in the network (routers, servers, etc.) have limited capacity

- Queue length
- CPU power
- etc.

☞ Overload (congestion) is not rare

Lack of Congestion at the Network Layer

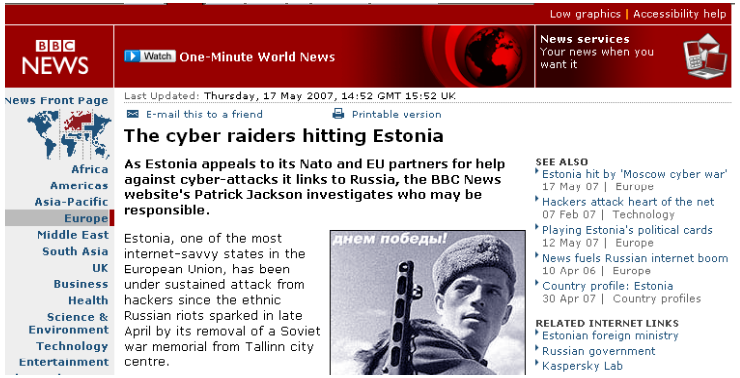
- ❑ Routing protocols react to the failure of a link or a router.
- ❑ But not to network congestions
- ❑ ARPANET had some mechanisms to react to congestions
- ❑ But they resulted into oscillations
- ❑ Congestion control was introduced in the Internet as enhancement of TCP
- ❑ But TCP has
  - no knowledge about the network topology
  - no way of re-wiring the traffic path in case of congestion

DoS Attack vs. Flash Crowds

- ❑ Big challenge
  - Ambiguous differentiation between DoS attacks and *flash crowds*
  - *Flash crowds*: unusual but legitimate traffic
  - Even if attacks are identified as such, it remains difficult to separate between malicious and legitimate traffic and to eliminate the malicious traffic

DoS Attacks

- ❑ Some DoS attacks have a political or ethnical reasons



Last Updated: Thursday, 17 May 2007, 14:52 GMT 15:52 UK

The cyber raiders hitting Estonia

As Estonia appeals to its Nato and EU partners for help against cyber-attacks it links to Russia, the BBC News website's Patrick Jackson investigates who may be responsible.

Estonia, one of the most internet-savvy states in the European Union, has been under sustained attack from hackers since the ethnic Russian riots sparked in late April by its removal of a Soviet war memorial from Tallinn city centre.

SEE ALSO
- Estonia hit by 'Moscow cyber war' 17 May 07 | Europe
- Hackers attack heart of the net 07 Feb 07 | Technology
- Playing Estonia's political cards 12 May 07 | Europe
- News fuels Russian internet boom 10 Apr 06 | Europe
- Country profile: Estonia 30 Apr 07 | Country profiles

RELATED INTERNET LINKS
- Estonian foreign ministry
- Russian government
- Kaspersky Lab

Challenges in the current Internet

1. Topology Failures
2. Overload
3. **Lack of Integrity**
4. Software Faults
5. Domino Effects

Lack of Integrity

- ❑ Majority of Internet traffic (signaling and data) is not integrity-protected
- ❑ This leads to several security vulnerabilities
  - ARP poisoning
  - Forged BGP announcements
  - Forged DNS responses
  - SPAM SPAM SPAM SPAM SPAM SPAM SPAM SPAM SPAM SPAM
  - etc.

Challenges in the current Internet

1. Topology Failures
2. Overload
3. Lack of Integrity
4. **Software Faults**
5. Domino Effects

Software Faults

- ❑ Developments faults
  - Introduced during the development phase
- ❑ Configuration faults
  - Introduced during the deployment phase

Software Faults

- ❑ Examples
  - Buffer overflows in server or router implementation
  - BGP Youtube misconfiguration
  - On Jan. 31st 2009, Google search engine marked every search result with "This site may harm your computer";

    Root cause: Database of suspected sites was mistakenly extended by ‚/‘
  - Software update of the Authentication Server (Home Location Register HLR) of T-Mobile on April 21st 2009
    - Impact: phone calls and text messaging were not possible for 4 hours

Challenges in the current Internet

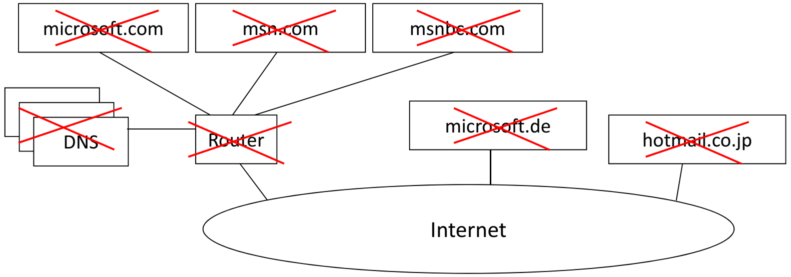
1. Topology Failures
2. Overload
3. Lack of Integrity
4. Software Faults
5. **Domino Effects**

Domino Effects

❑ Any kind of challenges mentioned above may lead to other challenges
- E.g., failure of a server in a server pool may lead to overload of neighboring servers
- Router failures may lead to congestion of neighboring links and routers
- DNS failure may lead to unavailability of other services,

Domino Effects

❑ E.g., DoS attack on Microsoft router on 24th + 25th Jan. 2001 lead to unavailability of DNS and thus of services located in other MS sites



Overview

I. Terminology
II. Challenges in the current Internet
III. **Resilience Mechanisms**

Resilience Mechanisms

1. Topology Protection
2. Congestion Control
3. Signaling Integrity
4. Server Redundancy
5. Virtualization
6. Overlay and P2P Networks

Resilience Mechanisms

1. **Topology Protection**
2. Congestion Control
3. Signaling Integrity
4. Server Redundancy
5. Virtualization
6. Overlay and P2P Networks

Topology-based Resilience Metrics

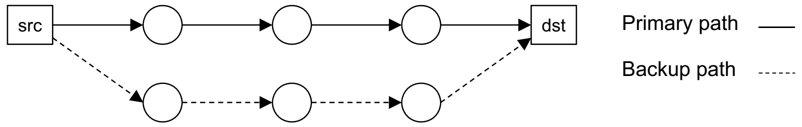
❑ Several metrics exist

❑ But not all are useful

❑ Definitions

- *k*-link (edge) connectivity is the minimal number of links whose removal would disconnect the graph
- *k*-node (vertex) connectivity is the minimal number of nodes whose removal (including removal of adjacent links) would disconnect the graph
- A *k-regular graph* is k-node-connected if there are k node-disjoint paths between any pair of nodes.

Path Protection



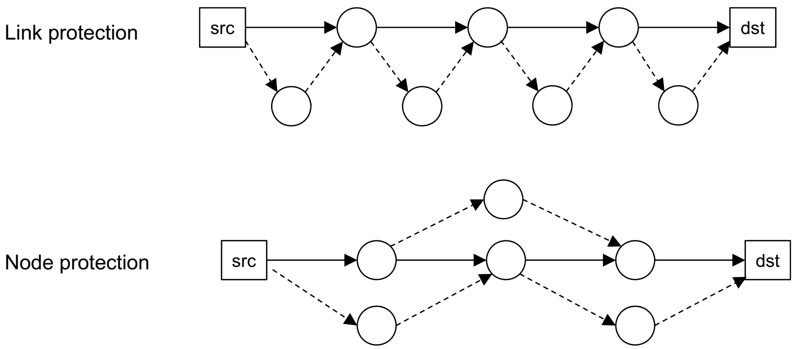
❑ Traffic is forwarded using backup path in case of failure

❑ Source needs to monitor the operation of primary path

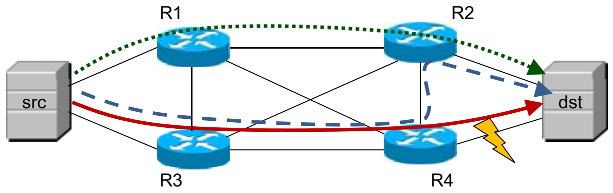
☞ Info about node or link failure needs to be propagated back to src

Local Protection

- ❑ Node or link failures are detected locally and backup paths are used until routing re-converges
  - ☞ This can reduces the MTTR by the order of a magnitude compared to *path protection*
  - ☞ Contra: higher signaling and equipment overhead



Example

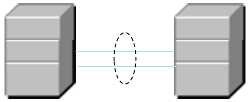


- ❑ Location protection at IP layer
- ❑ Routing protocol: OSPF
- ❑ Local protection according to IP Fast Reroute (IPFRR) (RFC 5714)

1. Normal operation: Routing from src to dst via R3 and R4
2. After failure of link between R4 and dst: Rerouting from R4 to dst via R2
3. Then, info is propagated in the network, OSPF routing converges and a new path is used from src to dst via R1 and R2.

IEEE 802.3ad: Link Aggregation

- ❑ IEEE Link Aggregation allows for bundling
  - ▪ several physical Ethernet connections
  - ▪ into a logical one
- ❑ Connection between
  - ▪ Two hosts
  - ▪ Two Ethernet switches
  - ▪ Host and switch



- ❑ IEEE Link Aggregation allows for increasing bandwidth
- ❑ But is also a fault tolerance mechanism
  - ▪ If a cable is plugged out,
    - • e.g., for maintenance reasons,
  - ▪ the two layer-2 devices remain connected.

Multihoming

- ❑ *Multihoming* refers to a network setup where a host or a network is connected to the Internet via more than 1 connection
- ❑ It can be applied in various contexts
  - ▪ Host Multihoming
    - • An IP host connected via multiple network interfaces
    - • Each network interface might be connected to a different access network
  - ▪ Multihoming at the transition point between networks
    - • An enterprise network connected to the Internet via multiple ISPs
    - • BGP peering with multiple providers

Resilience Mechanisms

1. Topology Protection
2. **Congestion Control**
3. Signaling Integrity
4. Server Redundancy
5. Virtualization
6. Overlay and P2P Networks

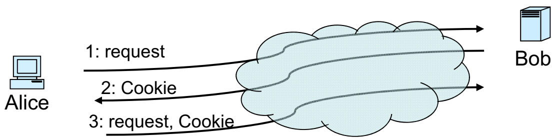
Congestion Control

- ❑ TCP congestion control
- ❑ Traffic Engineering
- ❑ Protection again DoS attacks
  - ▪ Rate limiting: vulnerable to
    - “false positives”, i.e., legitimate traffic is classified as malicious
    - “false negatives”, i.e., malicious traffic is classified as legitimate
  - ▪ Cookies

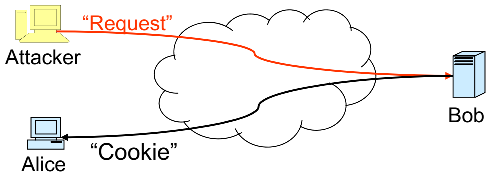
Traffic Engineering

- ❑ Addresses network congestion at the network layer
- ❑ Goals
  - ▪ Optimize network throughput, packet loss, delay
- ❑ Input
  - ▪ Network topology
  - ▪ Traffic matrix (may change over time, e.g., daily patterns)
- ❑ Output
  - ▪ (Eventually modified) link weights used to compute routing tables

Denial-of-Service Protection with Cookies (1)



- ❑ Upon receiving a request from Alice, Bob calculates a Cookie and sends it to Bob.
- ❑ Alice will receive the Cookie and resend the request with the Cookie together.
- ❑ Bob verifies that the Cookie is correct and then starts to process Alice‘s request.
- ❑ An attacker that is sending requests with a spoofed (i.e. forged) source address will not be able to send the Cookie.



Denial-of-Service Protection with Cookies (2)

- Cookies discussion:
  - Advantage: allows to counter simple address spoofing attacks
  - Drawbacks
    - Requires CPU resources
    - In some applications, e.g., DNS, it might be easier to respond to the request than generating the cookie
    - Requires one additional message roundtrip.
    - Network may remain congested

Resilience Mechanisms

1. Topology Protection
2. Congestion Control
3. **Signaling Integrity**
4. Server Redundancy
5. Virtualization
6. Overlay and P2P Networks

Signaling Integrity; “ARP” protection

- ❑ Manual configuration, e.g., ARP messages with wrong matching (IP to MAC) are discarded

  ☞ Too costly
- ❑ IPv6 SEcure Neighbor Discovery (SEND) (RFC 2461 and 2462)
  - Uses a Cryptographically Generated Address (CGA)

Routing prefix	Hash62(Host public key)

Signaling Integrity; DNSSEC

- ❑ Protects DNS responses with cryptographic signatures
- ❑ In a dedicated DNS record: the RRSIG record (RFC4034)
- ❑ DNS Records can be verified with a “chain of trust”
  - Public key of the DNS root zone must be known by clients
- ❑ Authority delegation is restricted to sub-domains
  - e.g., system administrator of “net.in.tum.de” can not sign records for “lrz.de”
  - Note: this is not the case for PKIs currently used in the web

Signaling Integrity; BGP Security

- ❑ Not trivial
- ❑ Can not be solved by simply adding message integration protection of BGP announcements
  - ▪ E.g., what is if “Pakistan Telecom” signs BGP announcements for a Youtube prefix?
- ☞ Integrity of BGP announcements needs to be validated by a combination of
  - ☞ topology authentication,
  - ☞ BGP path authentication and
  - ☞ announcement's origin authentication

Signaling Integrity

- ❑ Domain Keys Identified Mail (DKIM)
  - ▪ Allows for validation of a domain name associated with an email address
  - ▪ An organization takes responsibility for a message in a way that can be validated by a recipient
  - ▪ Prominent email service providers implementing DKIM
    - Yahoo, Gmail, and FastMail.
    - Any mail from these organizations should carry a DKIM signature

Signaling Integrity

- ❑ Spammers can still sign their outgoing messages
  - ☞ DKIM should be used with reputation:
    - Email messages sent by a domain that is known for signing good messages can be accepted
    - while others may require further examination.

Resilience Mechanisms

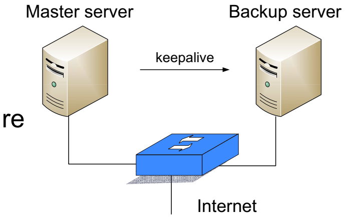
1. Topology Protection
2. Congestion Control
3. Signaling Integrity
4. **Server Redundancy**
5. Virtualization
6. Overlay and P2P Networks

Server Redundancy

- ❑ Server redundancy as a *fault tolerance* mechanism
- ❑ Servers instances may be
  - in the same LAN or
  - different sub-networks ☞ *Geographic diversity*
- ❑ Supporting mechanisms
  - IP Takeover
  - NAT Takeover
  - DNS

Server Redundancy; IP Takeover

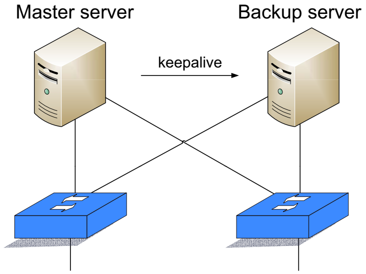
- ❑ Simple redundancy mechanism
- ❑ Backup server receives periodic “keep alive” messages from master server, e.g., every 10ms
- ❑ In case of no response
  - Backup server broadcasts an ARP message in the LAN
  - From now on, all IP traffic is forwarded to the backup server
- ❑ Drawbacks
  - Existing session state gets lost
  - Ethernet switch is a single point of failure



Server Redundancy; IP Takeover with 2 Switches

- ❑ Both master and backup servers are connected to 2 switches
- ❑ Same procedure with ARP

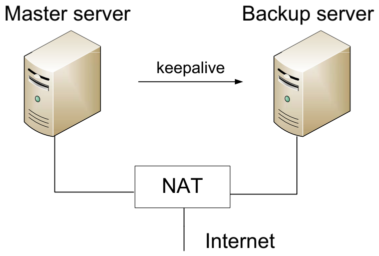
  ☞ Incoming requests from both switches is forwarded to the backup server
- ❑ Any component (server or switch or cable) can be removed, e.g., for maintenance reasons, while the service keeps on being available



Server Redundancy; NAT Takeover

- ❑ Similar to IP Takeover
- ❑ “Keep alive” messages from backup to master server
- ❑ Change NAT binding upon lack of response from master server

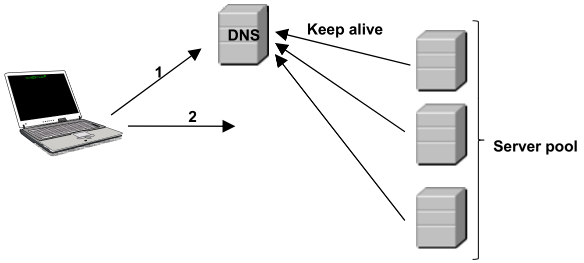
  ☞ Incoming requests are forwarded to the backup server



- ❑ Note: Master and backup server do not have to be in the same LAN

Server Redundancy; DNS

- ❑ DNS can provide several IP addresses for the same name
- ❑ By monitoring the availability of servers from a server pool, unavailable servers can be removed from DNS responses



- ❑ Moreover, DNS responses can be adjusted according to the current load
  - ☞ See, e.g., Content Distribution Networks (CDN)

Resilience Mechanisms

1. Topology Protection
2. Congestion Control
3. Signaling Integrity
4. Server Redundancy
5. **Virtualization**
6. Overlay and P2P Networks

Virtualization

- ❑ Different virtualization techniques, e.g., KVM, Xen, etc.
- ❑ Can be used to enhance resilience of network services
  - Start new servers from existing images *on demand*, e.g.,
    - To address overload situations
    - In case servers in other locations crash

Resilience Mechanisms

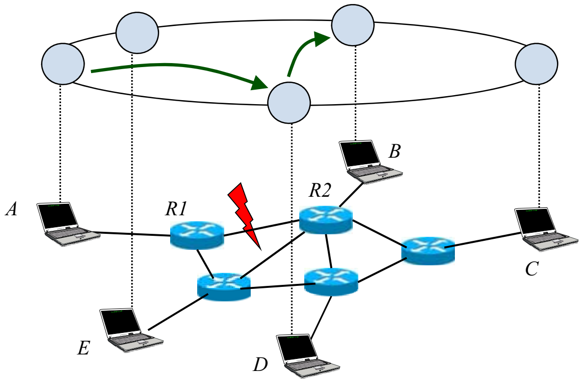
1. Topology Protection
2. Congestion Control
3. Signaling Integrity
4. Server Redundancy
5. Virtualization
6. **Overlay and P2P Networks**

Overlay Routing

- Overlay networks
  - Are networks built on top of existing networks
  - They typically provide additional functionality not provided at the „underlay“ network
- Overlay routing
  - End hosts can organize themselves in a P2P network
  - and provide routing using the overlay in case the underlay routing fails

Overlay Routing

- Example
  - Upon link failure between R1 and R2
  - A can reach B via D or C



Overlay Routing

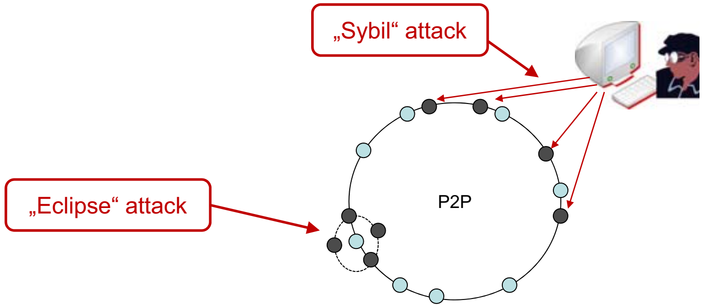
- Typical reasons for lack of connectivity in the underlay
  - Misconfigured middleboxes (firewalls, NATs)
  - Slow BGP convergence
- Systems supporting overlay routing
  - Tor
    - while it is actually designed with anonymization in mind, it provides overlay routing and can be useful in case of network partial failures
  - Skype
    - Skype supernodes typically provide connectivity for Skype clients behind firewalls or NATs

P2P Networks

- Resilience properties
  - Decentralization
  - Geographic diversity
  - Ability to cope with “churn”
    - “Churn” means that peers join and leave at any time
    - ☞ Replication of each data item on several peers
    - ☞ Autonomic recovery from stale P2P routing tables

P2P Networks

- ❑ Drawback: several attacks are possible
  - Sybil attacks:
    - Attacker participate with several fake identities
    - In order to control a portion of the network
  - Eclipse attacks,
    - Attacker control the neighborhood of a peer or content
    - In order to make unavailable for other participants in the P2P networks
  - etc.



„Sybil“ attack

„Eclipse“ attack

P2P Networks

- ❑ Common approaches
  - ☞ Managed P2P networks (or supervised P2P networks)
  - ☞ E.g., Google File System (GFS), Skype

Summary

I. **Terminology**

- ❑ The “*fault* ➔ *error* ➔ *failure*” chain
- ❑ Fault tolerance, Resilience, Dependability, Security
- ❑ Availability vs. Reliability

II. **Challenges in the current Internet**

- ❑ Topological Failures, Overload, Lack of Integrity
- ❑ Software Faults, Domino Effects

III. **Resilience Mechanisms**

- ❑ Topology Protection, Congestion Control, Signaling Integrity
- ❑ Server Redundancy**,** Virtualization, Overlay and P2P Networks

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

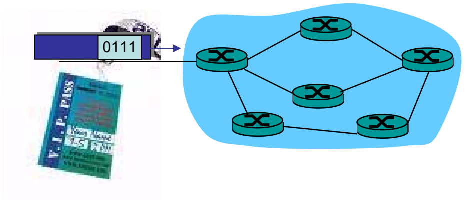
Chair for Network Architectures and Services – Prof. Carle
Department for Computer Science
TU München

Providing multiple Classes of Service

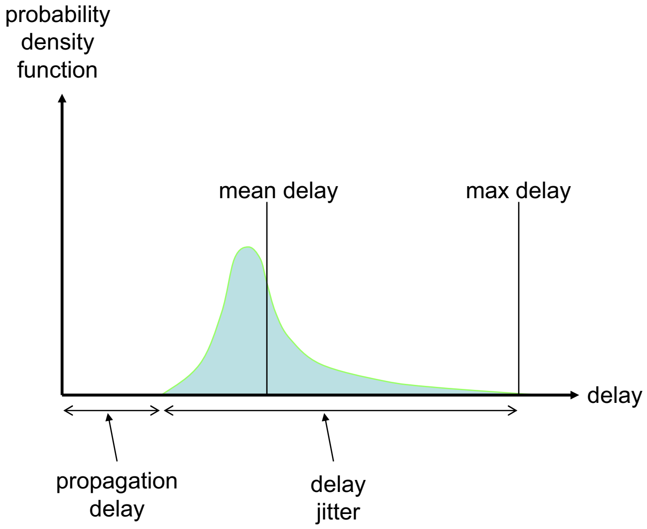
Technische Universität München

Providing Multiple Classes of Service

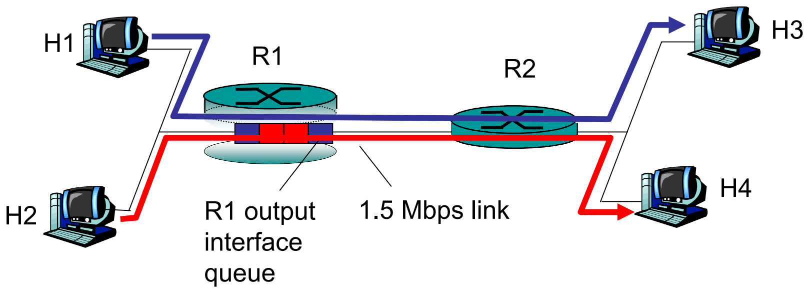
- Traditional Internet approach: making the best of best effort service
  - one-size fits all service model
- Alternative approach: multiple classes of service
  - partition traffic into classes
  - network treats different classes of traffic differently (analogy: VIP service vs regular service)
- granularity: differential service among multiple classes, not among individual connections
- history: ToS bits in IP header



Delay Distributions

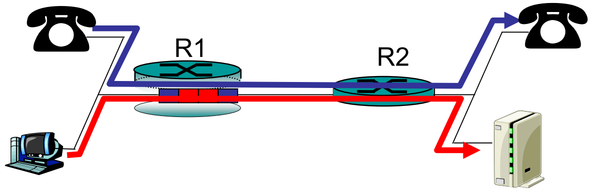


Multiple classes of service: scenario



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

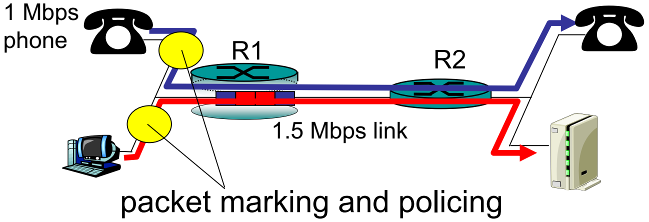


Principle 1

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)

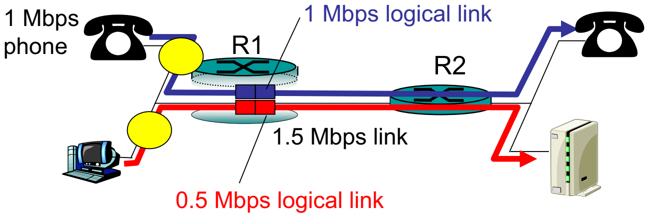


Principle 2

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

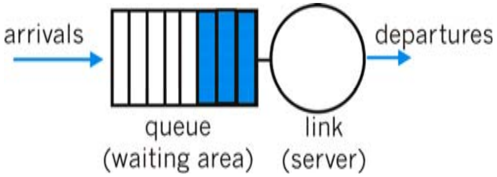


Principle 3

While providing **isolation**, it is desirable to use resources as efficiently as possible

Scheduling And Policing Mechanisms

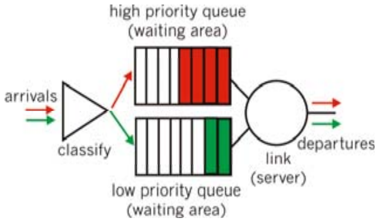
- scheduling: choose next packet to send on link
- FIFO (first in first out) scheduling: send in order of arrival to queue
  - ⇨real-world example?
  - discard policy: if packet arrives to full queue: who to discard?
    - Tail drop: drop arriving packet
    - priority: drop/remove on priority basis
    - random: drop/remove randomly

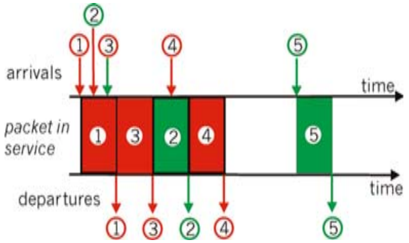


Scheduling Policies: more

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..

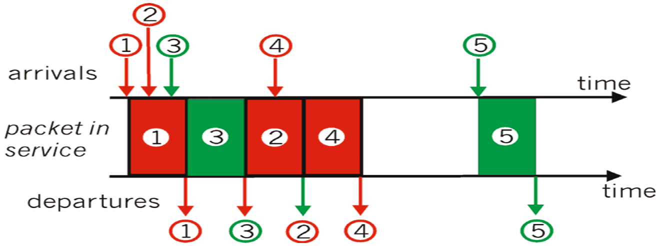




Scheduling Policies: still more

round robin scheduling:

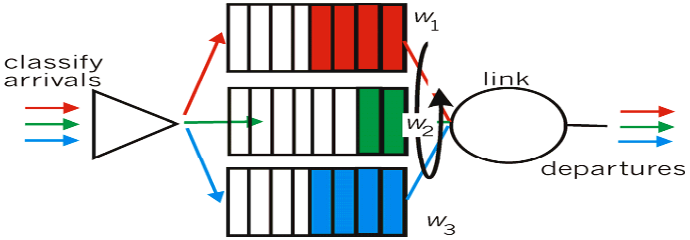
- multiple classes
- cyclically scan class queues, serving one from each class (if available)



Scheduling Policies: still more

Weighted Fair Queuing:

- generalized Round Robin
- each class gets weighted amount of service in each cycle
- when all classes have queued packets, class i will receive a bandwidht ratio of $w_i/\Sigma w_j$



Policing Mechanisms

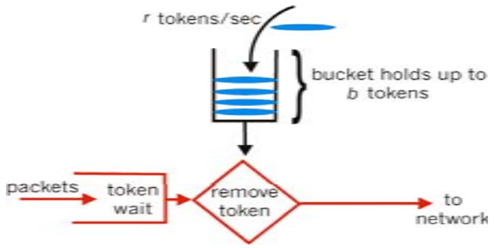
Goal: limit traffic to not exceed declared parameters

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 pps peak rate
- *(Max.) Burst Size:* max. number of packets sent consecutively

Policing Mechanisms

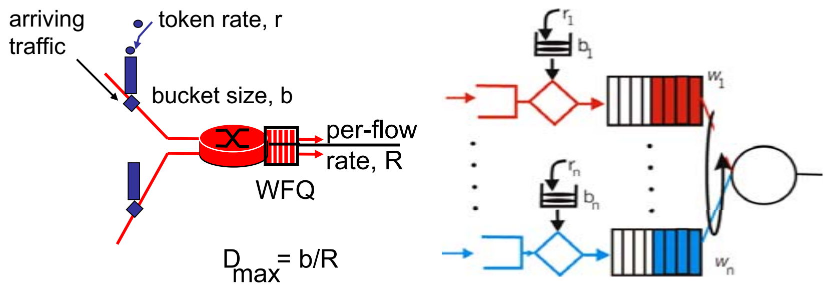
Token Bucket: limit input to specified Burst Size and Average Rate.



- bucket can hold b tokens ⇒ limits maximum burst size
- 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*



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)
  - in contrast to IETF Integrated Services: 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

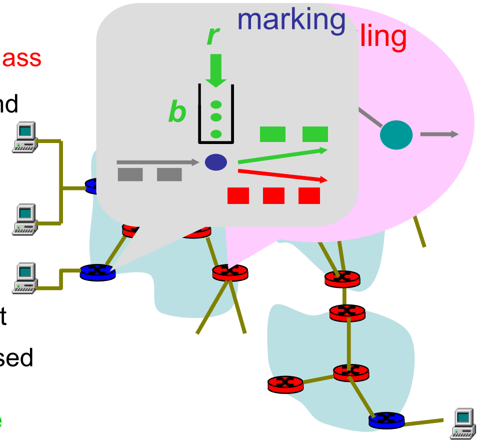
Diffserv Architecture

Edge router:

- per-flow traffic management
- marks packets according to class
- marks packets as in-profile and out-profile

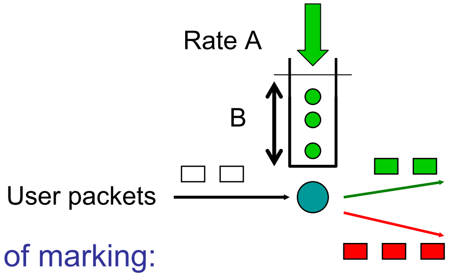
Core router:

- 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

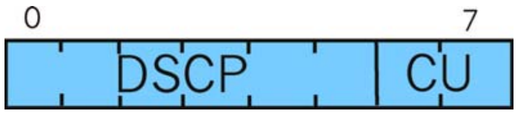


Possible usage of marking:

- 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

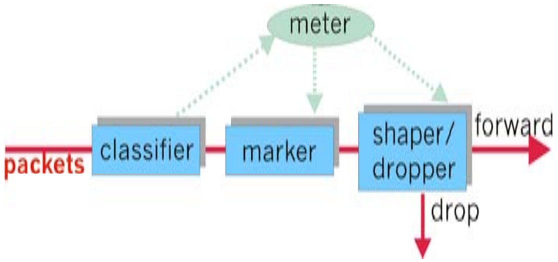
- Packet is marked in the Type of Service (TOS) in IPv4, and Traffic Class in IPv6
- 6 bits used for Differentiated Service Code Point (DSCP) and determine PHB that the packet will receive
- 2 bits can be used for congestion notification: Explicit Congestion Notification (ECN), RFC 3168



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 or dropped 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:

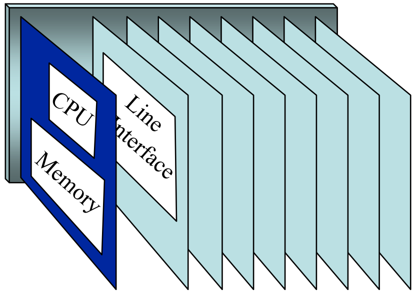
- Expedited Forwarding: packet departure rate of a class equals or exceeds specified rate
  - logical link with a minimum guaranteed rate
- Assured Forwarding: e.g. 4 classes of traffic
  - each class guaranteed minimum amount of bandwidth and a minimum of buffering
  - packets each class have one of three possible drop preferences; in case of congestion routers discard packets based on drop preference values

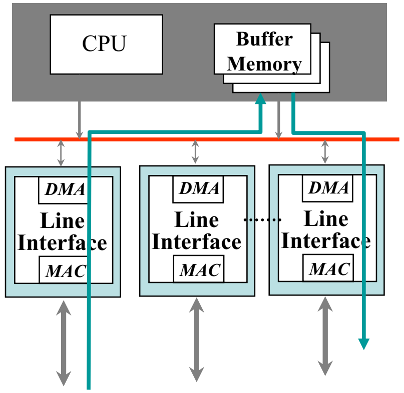
Chair for Network Architectures and Services – Prof. Carle
Department for Computer Science
TU München

The Evolution of IP Routers

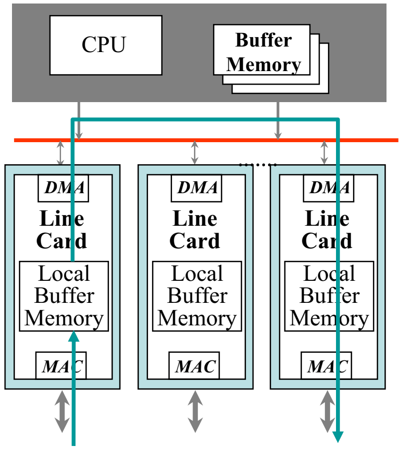
Technische Universität München

First-Generation IP Routers

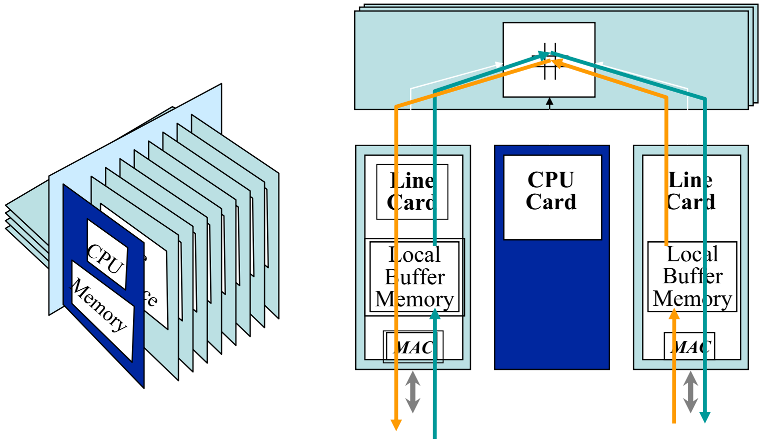




Second-Generation IP Routers

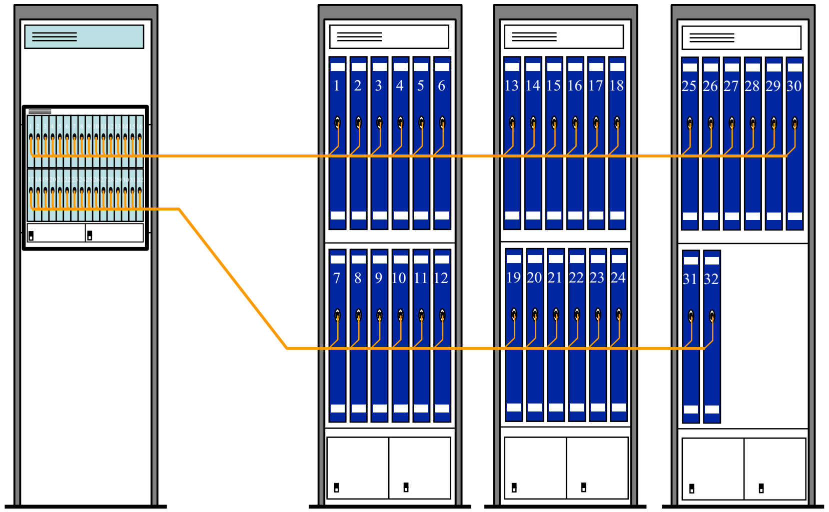


Third-Generation Switches/Routers



Fourth-Generation Switches/Routers

Clustering and Multistage



Chair for Network Architectures and Services – Prof. Carle
Department for Computer Science
TU München

Forwarding Implementations

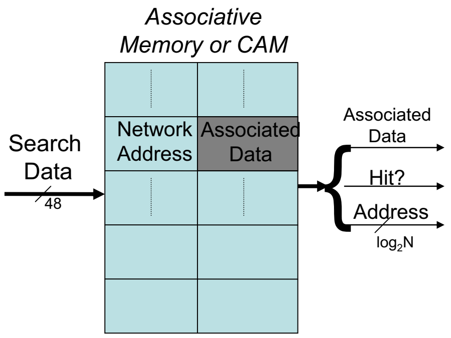
Principles

Credits:
Nick McKeown, Stanford University

Technische Universität München

Forwarding Implementation (1)

Associative Lookups



Advantages:

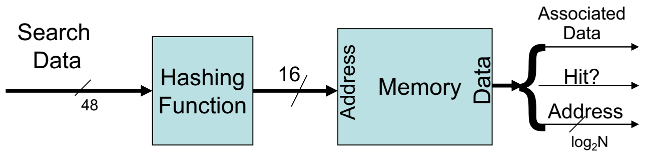
- Simple

Disadvantages

- Relatively slow
- High power consumption
- Small
- Expensive

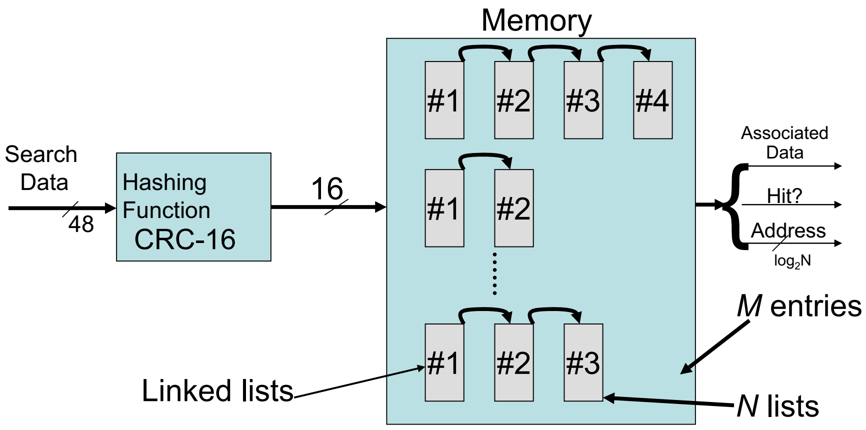
Forwarding Implementation (2)

Hashing



Lookups Using Hashing

An example



Lookups Using Hashing

Advantages:

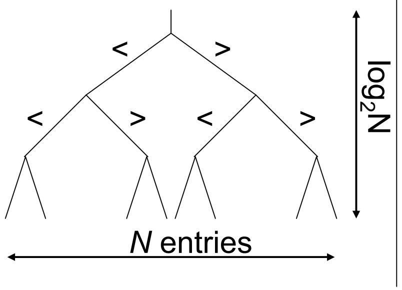
- Simple
- Expected lookup time can be small

Disadvantages

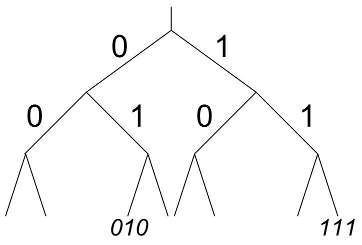
- Non-deterministic lookup time
- Inefficient use of memory

Forwarding Implementation: Trees and Tries

Binary Search Tree



Binary Search Trie

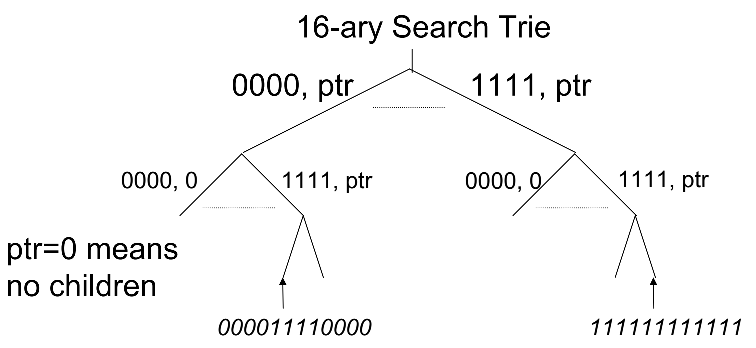


A *trie* (from re**trie**val), is a multi-way tree structure useful for storing strings over an alphabet.

Simple Tries and Exact Matching in Ethernet

- Each address in the forwarding table is of fixed length
  - E.g. using a simple binary search trie it takes 48 steps to resolve an MAC address lookup
  - When the table is much smaller than 2^48 (!), this seems like a lot of steps
  - Instead of matching one bit per level, why not m bits per level?

Trees and Tries: *Multiway tries*

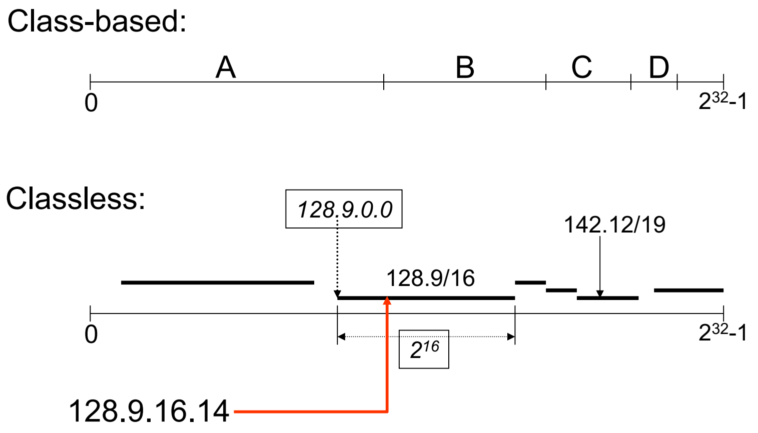


Chair for Network Architectures and Services – Prof. Carle
Department for Computer Science
TU München

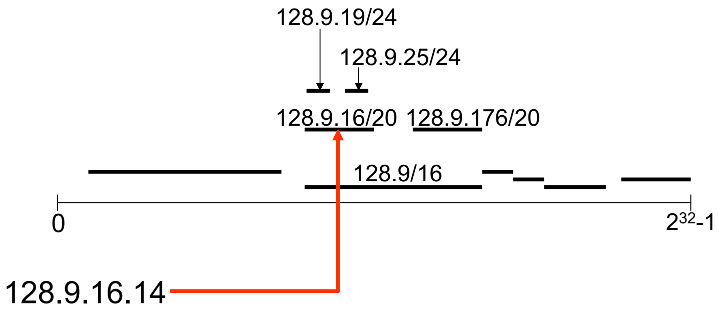
IP Forwarding

Technische Universität München

IP Routers: *CIDR*



IP Routers: *CIDR*



128.9.19/24

128.9.25/24

128.9.16/20 128.9.176/20

128.9/16

0

2^{32}-1

128.9.16.14

Most specific route = “longest matching prefix”

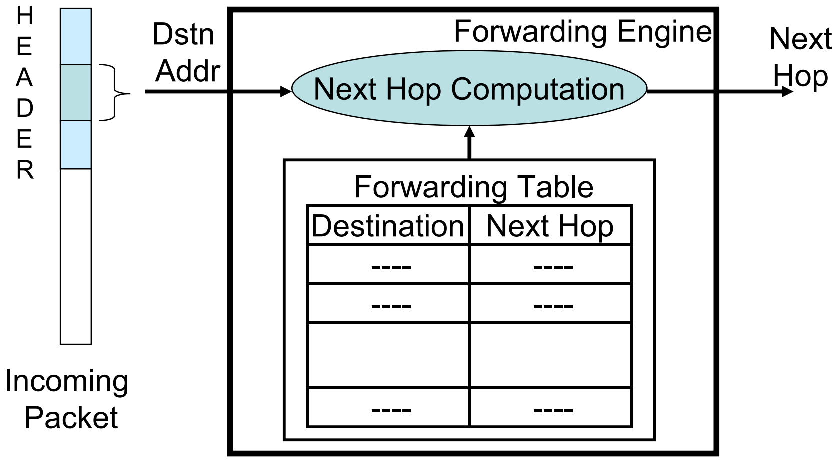
IP Routers: *Metrics for Lookups*

128.9.16.14

Prefix	Port
65/24	3
128.9/16	5
128.9.16/20	2
128.9.19/24	7
128.9.25/24	10
128.9.176/20	1
142.12/19	3

- Lookup time
- Storage space
- Update time

IP Router: *Lookup*



HEADER

Dstn Addr

Forwarding Engine

Next Hop Computation

Next Hop

Forwarding Table

Destination	Next Hop
----	----
----	----
----	----

Incoming Packet

IPv4 unicast destination address based lookup

Need more than IPv4 unicast lookups

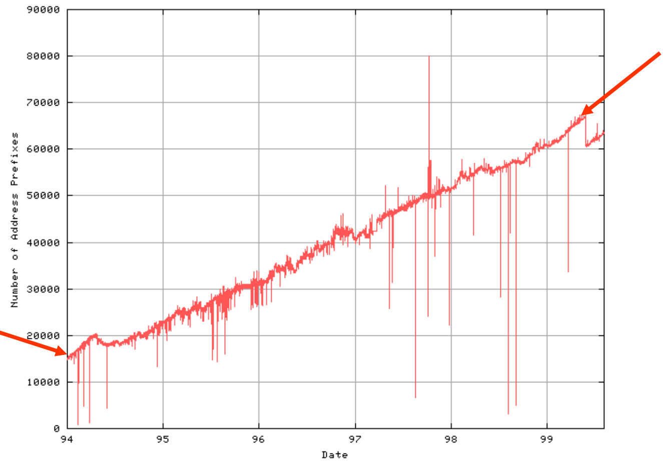
- ❑ Multicast
  - PIM-SM (Protocol-Independent Multicast, Sparse Mode)
    - Longest Prefix Matching on the source (S) and group (G) address
    - Start specific, subsequently apply wildcards: try (S,G) followed by (*,G) followed by (*,*,RP)
    - Check Incoming Interface
  - DVMRP:
    - Incoming Interface Check followed by (S,G) lookup
- ❑ IPv6
  - 128-bit destination address field

Required Lookup Performance

Line	Line Rate	Pkt-size=40B	Pkt-size=240B
T1	1.5Mbps	4.68 Kpps	0.78 Kpps
OC3	155Mbps	480 Kpps	80 Kpps
OC12	622Mbps	1.94 Mpps	323 Kpps
OC48	2.5Gbps	7.81 Mpps	1.3 Mpps
OC192	10 Gbps	31.25 Mpps	5.21 Mpps

Gigabit Ethernet (84B packets): 1.49 Mpps

Size of the Routing Table

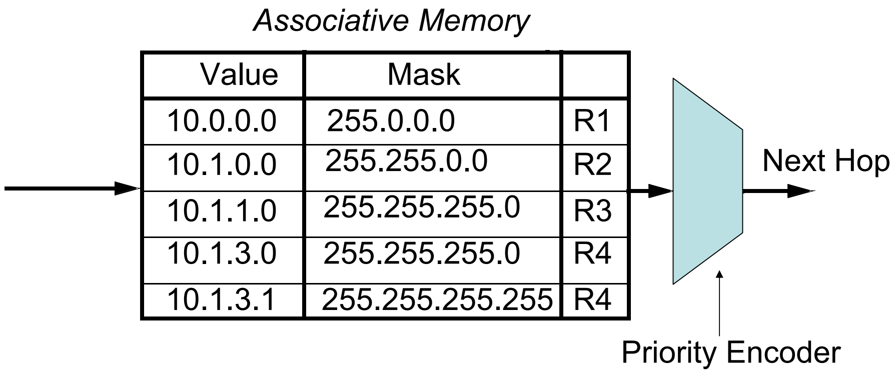


About 10k new prefixes per year

Exponential growth before CIDR

Source: http://www.telstra.net/ops/bgptable.html

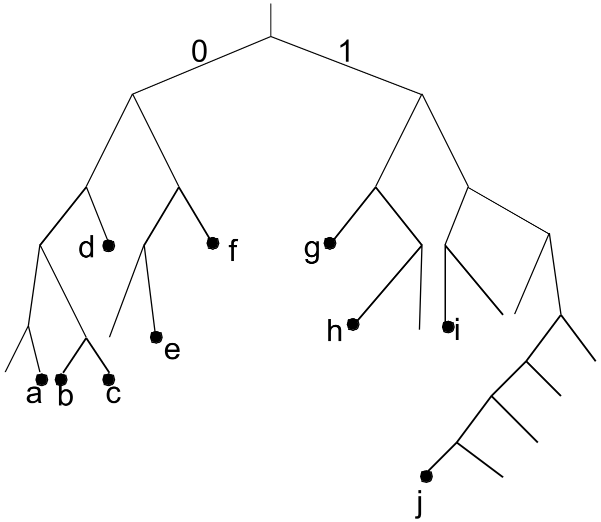
Method: CAMs (Content-Addressable Memories)



Associative Memory

Value	Mask	
10.0.0.0	255.0.0.0	R1
10.1.0.0	255.255.0.0	R2
10.1.1.0	255.255.255.0	R3
10.1.3.0	255.255.255.0	R4
10.1.3.1	255.255.255.255	R4

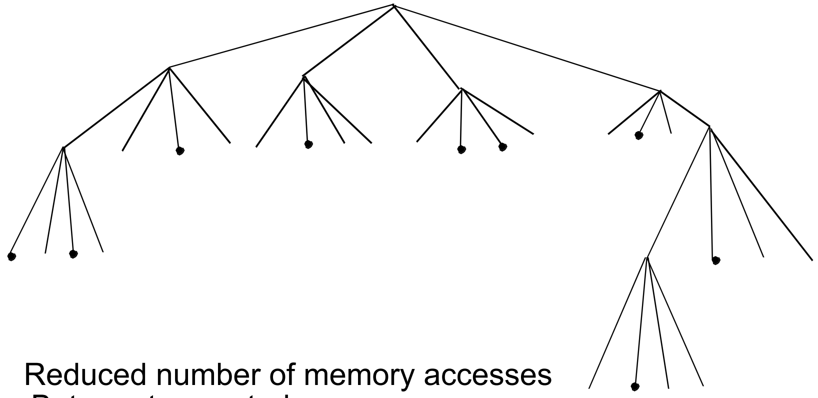
Method: Binary Tries



Example Prefixes

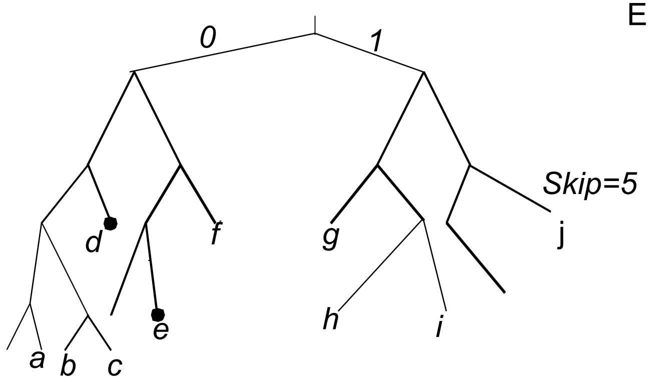
a) 00001
b) 00010
c) 00011
d) 001
e) 0101
f) 011
g) 100
h) 1010
i) 1100
j) 11110000

Four-way tries



Reduced number of memory accesses
But greater wasted space...

Method: Patricia Tree



Example Prefixes

a) 00001
b) 00010
c) 00011
d) 001
e) 0101
f) 011
g) 100
h) 1010
i) 1100
j) 11110000

Disadvantages

- Many memory accesses
- Pointers take a lot of space
  (Total storage for 40K entries is 2MB)

Advantages

- General solution
- Extensible to wider fields

Method: Compacting Forwarding Tables

- Optimize the data structure to store 40,000 routing table entries in about 150-160kBytes.
- Rely on the compacted data structure to be residing in the primary or secondary cache of a fast processor.
- Achieves e.g. 2Mpps on a Pentium.

Disadvantages

- Only 60% actually cached
- Scalability to larger tables
- Handling updates is complex

Advantages

- Good software solution for low speeds and small routing tables.

Chair for Network Architectures and Services – Prof. Carle
Department for Computer Science
TU München

Forwarding Decisions

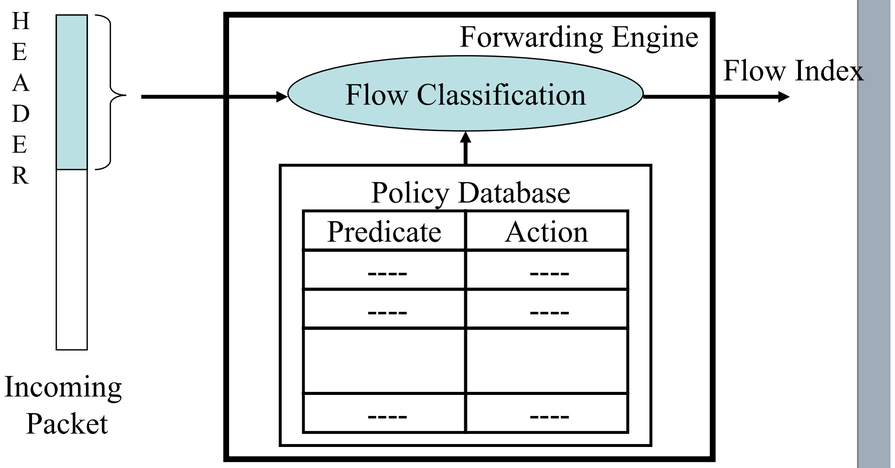
Packet Classification

Technische Universität München

Providing Value-Added Services: Some examples

- Differentiated services
  - Regard traffic from AS#33 as `platinum-grade'
- Access Control Lists
  - Deny udp host 194.72.72.33 194.72.6.64 0.0.0.15 eq snmp
- Committed Access Rate
  - Rate limit WWW traffic from sub-interface#739 to 10Mbps
- Policy-based Routing
  - Route all voice traffic through the ATM network
- Peering Arrangements
  - Restrict the total amount of traffic of precedence 7 from
  - MAC address N to 20 Mbps between 10 am and 5pm
- Accounting and Billing
  - Generate hourly reports of traffic from MAC address M

Flow Classification



Predicate	Action
----	----
----	----
----	----

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

Internet Architecture
and Future Internet

Outline

- What's wrong with today's Internet?
- Some general architecture principles
- Concepts for a Future Internet

What's wrong with today's Internet?

- Spam, Spit (=VoIP spam), worms, viruses, DDoS attacks
- Highly insecure routing protocols (e.g., BGP)
- No QoS mechanisms for end users
  (although protocols exist: DiffServ, RSVP, ToS headers in IP, …)
- Multihoming/IP roaming: difficult! IP address changes → connections break; TCP only can use one IP interface (although solutions exist: Mobile IP, SCTP, …)
- Number of prefixes in BGP routing tables grows dramatically.
  Reasons: growing number of providers; companies doing multi-homing or changing providers while keeping their address space; IPv6
- Almost no IP addresses left. (IPv6 exists, but nobody is using it.)
- Routing not very well understood. Routing and connection problems hard to analyse: what's the reason, who is to blame?
- TCP not well adapted to wireless networks: high delay variation, packet losses not induced by congestion, …

Problem: Routing security

- Recall that BGP manages traffic between providers
- By attacking BGP (e.g., injecting false information), IP packets headed for specific prefixes can be "hijacked" (e.g., diverted to the attacker)
- Problem: BGP is dangerously insecure! See the following two examples…

Prominent incident #1: Pakistan blocking YouTube

- YouTube was banned in Pakistan by law
- Pakistan Telecom thus was required to block access to YouTube
  - Pakistan Telecom announced YouTube IP prefix via iBGP, so that IP packets directed to YouTube traffic got routed into a black hole
  - Accidentally, they announced it via eBGP, too
- Consequences:
  - ASes in topological vicinity of Pakistan Telecom saw the (black hole) route in addition to the YouTube route
  - Due to the shorter AS path, they preferred it over the original
  - Result: YouTube not available in large part of the world

More details: Renesys blog, *Pakistan hijacks YouTube,* Feb 24th, 2008

Prominent incident #2: China hijacks the Internet

- Very similar to the Pakistan incident
- On Apr 8th, 2010, China telecom incorrectly announced 50,000 prefixes it did not own
  - N.B. 50,000 prefixes out of globally 350,000 prefixes = 15%; ***but not*** 15% of all traffic!
  - Prefixes included US government, US military, etc.
    In total, prefixes from 170 countries (including China!)
- Consequences
  - Many ASes in topological vicinity of China Telecom thus shifted the routes for the affected prefixes to China Telecom
  - A small ISP would have crumbled under the tsunami of traffic, but China Telecom is a big player, so they could handle it.
    In other words: Traffic now got routed via China Telecom
  - The event lasted about 20 minutes
- Deliberately?
  - Most certainly not!

More details: Renesys blog, *China's 18-Minute Mystery,* Nov 18th, 2010

What's wrong with BGP?

- Every AS can announce announce an IP prefix – even if it's not its own!
  - Inter-AS routing works on the honour system (like, e.g., Hawala)
  - Some plausibility filtering of BGP updates is done, but it's optional
- BGP sessions (=TCP connections) are not cryptographically encrypted.
- Very weak authentication
- Conjecture: Presumably a lot of bugs in router OSes that would allow buffer overflow exploits etc.

- Could become a great cyberwar battlefield!

Butler, Farley, McDaniel, Rexford: *A Survey of BGP Security Issues and Solutions*
Proceedings of IEEE, 2009

Problems with routing (2)

- Dynamics of BGP are hard to understand
  - Many different vendor implementations
  - Complex configuration
  - A lot of potential error sources
  - Many effects still not understood
- Routing issues are hard to debug
  - No global view!
  - N.B.: BGP peers are competitors at the same time, so hide as much information as possible
  - "Where does the error come from? Who is to blame?" – Often, these questions are very difficult to answer.

- Solution: More research!?

Problems with routing (3)

- How long does it take to react to a hardware failure? (e.g., cable cut)
  - MPLS and layer-2 protocols: a few milliseconds
  - OSPF and intradomain routing: 200ms – 2s
  - BGP: seconds to minutes. Sometimes even hours!
- Reasons:
  - Unrestricted BGP would send out a lot of update messages
  - Thus, many mechanisms built in to reduce # of messages:
    Route flap damping, MRAI timer, …
  - Delaying messages means delaying information about topology changes!

- Solutions:
  - Layer-2 switching, MPLS Fast ReRoute (MPLS FRR) within a provider's network (~10ms)
  - IP Fast ReRoute (IPFRR) being standardised by IETF; scope: mainly within a provider's network
  - No widely accepted solutions for interdomain FRR yet

Problems with routing (4): End host mobility, end host multi-homing

- When a laptop changes its network connection from WLAN to 3G/UMTS, it gets a new IP address
- Existing TCP and UDP connections break, e.g.:
  - Persistent HTTP connections (usually transparent)
  - Instant messenger chats (úsually short interruption)
  - VoIP conversations
  - Ongoing HTTP or FTP requests like file downloads
  - VPN tunnels, ssh sessions, …
- And why can’t I simply bundle multiple interfaces to increase bandwidth?
  - Each interface has its individual IP address

- Structural problem:
  - IP address = ***identifier*** for TCP endpoint
  - IP address = ***locator*** for IP routing

Problems with routing (5): Network mobility, network multihoming

- Network mobility:
  - Suppose company C is customer of provider P
  - Provider P owns prefix 10.0.0.0/8
  - C is assigned network 10.11.0.0/16
  - Now C wants to change to another provider X
  - Solution 1: C is assigned completely new IP addresses from X.
    A lot of administrative overhead!
  - Solution 2: The new prefix 10.11.0.0/16 is announced by X.
    (N.B.: No conflict with 10.0.0.0/8 due to longest prefix matching)
- Network multihoming:
  - Same as above, but C wants to use a link to X as a backup
  - By accident, the link C—P is cut. C now sends out packets via X.
  - But how do the reply packets come to X, not to P?
  - Solution: The prefix 10.11.0.0/16 was previously announced by P *and* X (X probably announces a worse path by employing AS path prepending).
    After the cable cut, P withdraws the prefix: The world switches to X.
- OK, a bit cumbersome… but where’s the problem here?

Problems with routing (6): IP address space fragmentation and number of routing table entries

- Number of IP prefixes in globally operating routers: ≥350,000 and rapidly growing
- Reasons:
  - Many new providers, many new users, especially in emerging markets
  - Companies that want to keep their IP addresses while
    - changing providers
    - or doing multi-homing (i.e., be connected to Internet via two different providers)
  - Plus: In the future, we’ll see more and more IPv6 prefixes
- Is there a problem with that?
  - Routers need more memory, faster CPUs
  - Linecards become more expensive (more silicon needed for hardware-based longest prefix match with more entries)
  - Increasing number of BGP updates, ASes, AS paths
    - More BGP traffic
    - …which means: more BGP instability!

Mobility, multi-homing: Solutions (1)

- Mobile IP
  - Keep a permanent IP address when roaming at a relay
  - Mobile IPv4: Relay (“home agent”) introduces delays; issues with firewalls. Mobile IPv4 is dead.
  - Mobile IPv6 Route Optimisation: Tell shortcut to “real” IP address
  - Complex; largely unknown. (And: Who uses IPv6 anyway?)
  - Purpose: A solution for end hosts or *small* mobile networks (e.g., all end nodes within a train)
- HIP (Host Identity Protocol)
  - A host maintains a permanent unique identifier
  - Identifiers have 128 bit (=length of an IPv6 address):
    HIP ID can be inserted as “layer 3.5” between IP and TCP or UDP
  - When IP address changes, a host can inform peers about the new IP address for the identifier using HIP
  - Cryptographically protected from hijacking à la BGP
  - Purpose: A solution for end hosts

[Mobility,] multi-homing: Solutions (2)

- SCTP (Stream Control Transmission Protocol)
  - ~successor to TCP (and UDP)
  - One SCTP association (connection) can use multiple interfaces
  - Problems:
    - Firewalls and NATs reject traffic that is not TCP, UDP, ICMP
    - Slightly more difficult to use than TCP or UDP
    - Nobody knows it ⇄ nobody uses it
  - Purpose: A solution for end hosts. Mainly addresses multi-homing; mobility is difficult.

- http://tdrwww.exp-math.uni-essen.de/inhalt/forschung/sctp_fb/sctp_intro.html
- Violin Yanev, SCTP, Ausarbeitung im Blockseminar Future Internet, SS2010

Mobility, multi-homing: Solutions (3)

- LISP (Locator–ID separation protocol)
  - Two types of IP addresses (i.e., disjoint address spaces):
    - End hosts / networks at edge use EIDs (end point IDs)
    - Routers in network core use RLOCs (routing locators)
  - IP Packets with EIDs are encapsulated into IP packets with RLOCs at *ITRs* (ingress tunnel routers), unwrapped at *ETRs* (egress TRs)
  - Idea:
    - Packets with EIDs are tunneled through opaque RLOC net
    - Many different EIDs, whereas #RLOCs ~ network size
  - Purpose:
    - Facilitate *network* mobility (changing providers)
    - Facilitate *network* multihoming
    - Allow these without further inflating BGP routing tables
    - *Not in focus:* host mobility…
  - Being standardised by IETF and Cisco

http://lisp4.cisco.com/lisp_over.html

Problem: Energy efficiency

- Context 1: Green IT
  - Communication infrastructure uses more and more energy
  - Bad for the environment (CO_2)
  - Increases costs
- Context 2: Mobile nodes
  - How long did a fully charged mobile phone last 10 years ago?
    And how long does a fully charged iPhone, Android, Symbian phone last?
- Solution A: Develop better hardware. (Not a problem of network research…)
- Solution B: Develop protocol improvements or new protocols that can help saving energy
  - Examples
    - 802.11b (11 Mbit/s, 100mW, range <500m): battery hog
    - UMTS HSPA (7 Mbit/s, 250mW, range ~2km): battery friendly
    - GSM (384 kbit/s, up to 2W, range ~5km): even friendlier to battery
  - Some principles (only a small selection; there's much more to it!):
    - Energy-efficient routing (wireless mesh networks)
    - Reduce broadcasts (wakes up receivers)
    - Try to send packets in one burst (wake up less often)

Problem: Malicious traffic, malicious users

- Unwanted traffic: Spam, Spit (=VoIP spam), DDoS attacks
- Worms, viruses, break-ins

- Basic problems:
  - Buggy implementations (perhaps not a problem of the network)
  - Most protocols do not use reliable authentication (e.g., SMTP)
  - The network cannot filter out undesired traffic on-demand
    - Static filters are widely used, though (rate limits, IP blocks, …)
  - "Bullet-proof hosting": ISPs that do not react to abuse reports
  - Users who are agnostic about security issues leave the door wide open for attackers ("I don't need a virus scanner, because I already have a firewall")

- Solution: ??

Problems with congestion control (1)

- Congestion control is done in end hosts (TCP)
  - Detects and minimizes congestion along the path
  - Path is determined by IP routers, not by end hosts
- What if alternate path exists without congestion?
  - Only changes in routing tables can shift traffic to uncongested paths
- Problem #1: Routing protocols normally do not react to routing!
  - Historic reason: Arpanet used traffic-adaptive routing, which led to oscillations → bad experiences
  - Technical reason: (Nonlinear) feedback loops with delay, plus coupled feedback loops → difficult
  - Today, providers use Traffic Engineering: Analyze link usage in network, change OSPF weights (etc.) to improve performance
    - Time scale: hours, not seconds

Problems with congestion control (2)

- Problem #2: What if I need TCP-like congestion control, but not the other TCP features? (e.g., for video streams)
  - Solution: Use DCCP instead of TCP or UDP.
  - But nobody uses it; Firewall issues (cf. SCTP); …
- Problem #3: TCP-friendliness
  - New congestion control schemes must be fair to existing TCP implementations: They must not take away all bottleneck bandwidth, but they must leave standard TCP its fair share
  - Restricts solution space for new congestion control schemes
- Problem #4: Radio networks
  - Hidden terminal effect, fading → packet losses.
    But TCP treats them as sign of congestion.
  - High variability in link delays. Also bad for TCP.
  - Solution: Introduce additional retransmission features on layer 2…

Problems with delay (1)

- Long network delays. Example:
  - 1,100km from here to Salerno, including detours of cable.
    Speed of light in fibre (not vacuum!) is about 200,000km/s.
    Expected propagation delay: about 5.5ms, i.e., RTT=11ms.
  - Real value: 38ms!
- Even longer protocol delays.
  Example for accessing a Web site:
  - [ARP request, response: within LAN → negligible]
  - DNS lookup, possibly across several hierarchy levels: ≥ 1 RTT
  - [In future: HIP negotiation? +1RTT]
  - Sending TCP SYN, waiting for SYNACK: +1 RTT
  - [In case of HTTPS: SSL/TLS negotiation: +1 RTT]
  - Sending HTTP request, receiving HTTP response header: +1 RTT
  - In total: 3…5 RTT *plus* transmission delays until we receive the first byte of the content
  - Try it with www.cnu.ac.kr (Chongdu National University, Daejeon, Korea). The delay is certainly *not* due to a small bandwidth!

Delay: solutions

- Solution 1: Caching (e.g., HTTP proxies)
  - Persistent HTTP connection to proxy
  - Hopefully, proxy has cached object: Saves DNS lookup and TCP handshake; and presumably RTT to proxy is small
  - But if not, add processing delay of proxy plus RTT to proxy… not good.
- Solution 2: Distribute access points across the network
  - DNS-based (used frequently for Content Delivery Networks, e.g., Akamai, Google/YouTube):
    - If you enter www.youtube.com, you get a different IP address depending on your location in the network.
    - Principle: One DNS name, many hosts
  - Anycast-based (used for some root nameservers and 6to4 gateways):
    - You use the same IP address (e.g., 199.7.83.42, the L root name server run by ICANN), but this IP prefix is announced by multiple ASes across the globe
    - Principle: One IP address (!), many hosts
  - Both solutions are rather expensive

Problems with delay (2): Delay-tolerant networks

- Case 1: Underdeveloped regions
  - No permanent connections (at least not fast ones)
  - But: could transport USB sticks/hard disks back and forth “Never underestimate the bandwidth of a station wagon”:
    Delivering 1 TByte every 10 days is about 10Mbit/s!
- Case 2: Sparse mobile wireless networks (e.g., desaster areas)
  - Transmission often interrupted
  - People with Bluetooth/ad-hoc WLAN devices wander around and meet
- Case 3: Space travel
  - You can’t beat the speed of light
  - Moon: 2s RTT, sun: 17min RTT, outer planets: several hours
  - Pre-scheduled times without connectivity (e.g., while behind a planet)
- Obviously, we can’t do things like, e.g.:
  - TCP: handshake = 1 RTT; furthermore timeout = 120s
  - DNS or other lookups prior to sending request
- Solutions:
  - Very, very old: uucp
  - Bundle Protocol

Problems with quality of service (QoS)

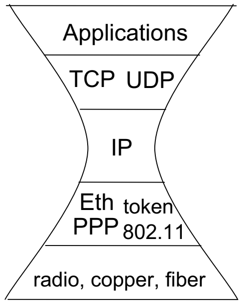
- Quality of service:
  - Interactive traffic (e.g., online games) more important than bulk traffic (e.g., e-mails, file transfers)
  - Therefore, give bandwidth guarantees and/or prefer it in queueing: Smaller queueing delays and/or smaller delay variation and/or reduced packet loss, etc.
- Solution:
  - IP TOS field (type of service) has been existing for years
  - Signalling protocols for establishing QoS connections (IntServ, DiffServ, RSVP,…) have been existing for years
- Status quo:
  - Being used within provider networks (e.g., for customer VPNs)
  - But: No end user software uses it!
- Problems:
  - Who pays when priority traffic goes across provider boundaries?
  - How to identify who has to pay?
  - What about DDoS attacks? (technically *and* financially)

Problems with multicast: Same as with QoS!

- Multicast:
  - Like broadcast, but to a specific group of recipients
  - For example, for streaming a TV program via a network
- Solution:
  - IP multicast addresses have been existing for years
  - IP multicast routing protocols have been existing for years (e.g., M-OSPF)
- Status quo:
  - Being used within provider networks (e.g., for triple-play with IPTV)
  - But: No end user software uses it!
- Problems:
  - Who pays when a multicast packet enters a provider via one link, and copies leave the network via 100 links?
  - How to identify who has to pay?
  - What about DDoS attacks? (technically *and* financially)

Problems with layers (1)

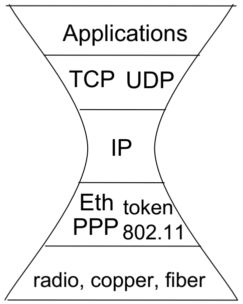
Original idea: The IP hour glass figure



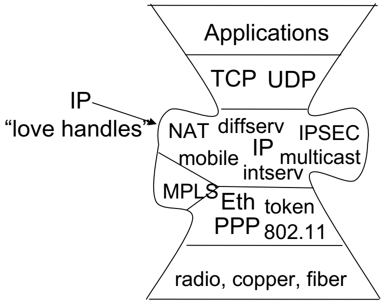
IP “hourglass”

Problems with layers (2)

Supporting new applications and services → losing the IP hour glass figure



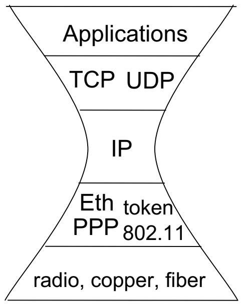
IP “hourglass”



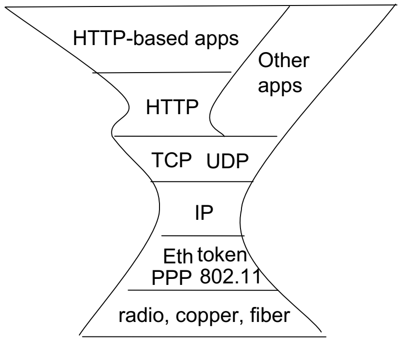
Middle-age IP = “hourglass” ?

Problems with layers (3)

- We used to have: * over IP
- We have today: * over HTTP over TCP over IP (e.g., Skype phone calls, YouTube video streams)



Original idea:
IP is greatest common denominator



Today:
HTTP is greatest common denominator

Fundamental Problem: The Internet only just works

Many more cases of “a solution exists in theory, but not in practice”:

- Example 1: IPv6
  - We’re out of IPv4 addresses.
  - IPv6 has been there for 15 years, but it’s still not being used
- Example 2: DNSsec
  - By injecting false information into the DNS base, you could conduct attacks similar to BGP (e.g., Pakistan–Youtube or China Telecom)
  - DNSSEC: cryptographically signed DNS entries
  - Recently installed for some TLDs, but no browser/resolver uses it

Fundamental reason: *Never touch a running system*

- Corollary:
  Only do something new when the old system really, really starts to hurt
  - = the “ossification” of the Internet architecture
- Examples in the past:
  - TCP/IP was born when NCP (Arpanet) went out of control
  - TCP congestion control was deployed only when a congestion collapse was imminent
  - DNS was deployed only when the centrally managed HOSTS.TXT file grew too large and got unmanageable
  - CIDR IP prefixes (instead of the old class A, B, C networks) and NAT were deployed when IP addresses got too scarce
  - Used cookies, hidden forms, GET-IDs for tracking sessions in HTTP (=sessionless)
  - PPP, DHCP, NAT, POP3 when more users connected from home
  - ssh instead of Telnet/rlogin (encryption; automated X11 forwarding)

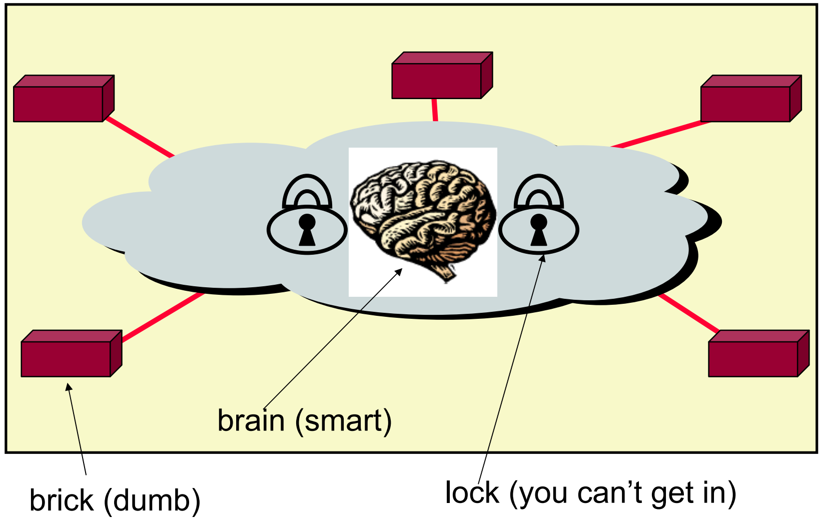
Mark Handley: *Why the Internet only just works.* BT Technology Journal, 2006

Future Internet

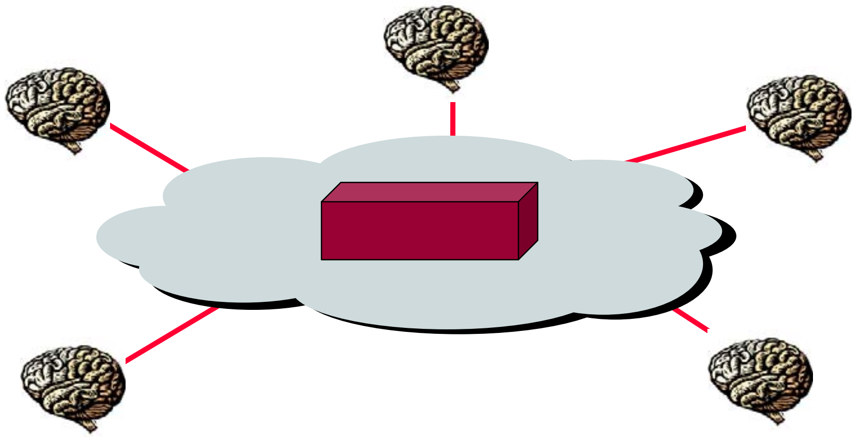
- There seem to be some fundamental flaws in the architecture of the Internet, so let's fix them
- Important research direction: A lot of money and effort going into this
- Sarcastic view:
  - Traditional network research was about developing new protocols to improve services and performance of the Internet
  - But Future Internet research is about developing new protocols to improve services and performance of the Internet
- But there is more to it – we have started asking (and answering) fundamental questions about the network architecture

- So… what are the basic concepts behind the architecture of today's Internet?

Common View of the Telco Network: Smart network, dumb endpoints

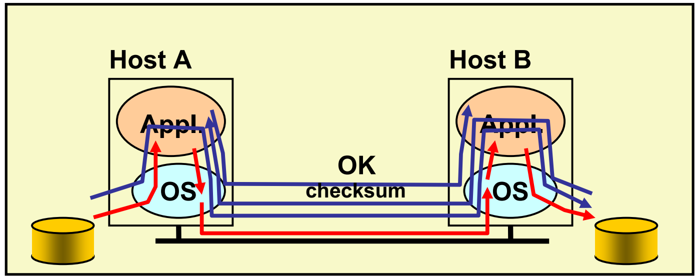


Common View of the IP Network: Dumb network, smart end hosts



The Internet End-to-End principle

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)

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 (except 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 from 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: Ethernet, Frame Relay, ISDN, FDDI, ATM
  - Today: SONET/SDH, WDM, WLAN, DSL, GSM
- Assessment: ?

Other Goals

- Allow distributed management
  - Administrative autonomy: IP interconnects networks
    - Each network can be managed by a different organisation
    - Different organisations 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 considerable harm (e.g., DHCP server running on laptop in LAN; ARP spoofing)
  - Assessment: ?
- Accountability
  - Works well if you only consider data volumes: just count bytes
  - Hard to do if you want to differentiate different kinds of traffic
    - Network neutrality: Pay extra money if you want to access Facebook, Youtube etc. in good quality
    - QoS: Cannot establish QoS connections across providers, because: who pays?
  - Very hard to pin down who did what (e.g., who is responsible for that strange BGP behaviour?)
  - Assessment: ?

Many implicit assumptions from the old days do not hold any longer

1970s, 1980s:

- Network is used by scientists/government.
  Very few malicious users, if any.
- Tens of networks, hundreds of hosts, thousands of users
- Network is jointly operated by public institutions without financial/economic interests.
- Host and network administrators are benevolent and not malicious. And they know their job. Normal (potentially unknowing, malicious) users do not have administrator privileges.

Today:

- Network is used by all kinds of people. Many malicious users (who even have financial incentives, e.g., phishing).
- Thousands to millions of networks, billions of hosts and users
- Network operated by competing companies.
- Unknowing users (“I don’t need a virus scanner, since I have a firewall”) and even malicious users (crackers, script kiddies) administrate their own hosts. Or even entire networks (e.g., bullet-proof hosting).

Technical response to changes

- Trust: emerging distinction between what is “in” network (*us*, trusted) and what is not (*them*, untrusted).
  - Firewalls, NATs
  - Ingress filtering
- 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

What’s missing?

Missing:	But be careful not to lose:
• No built-in security features (e.g., Spam, DDoS, worms)	• Possibility to communicate anonymously (e.g., Tor, Gnunet)
• Architecture does not reflect economic relations (“tussle”) (e.g., QoS, multicast)	• Network neutrality
• A routing system that is understandable, efficient, fast, and easy to debug	• Network neutrality
• Host and network mobility	• Possibility to communicate anonymously
• …many more features, these are just the most important ones	• …many more features, these are just the most important ones

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.”

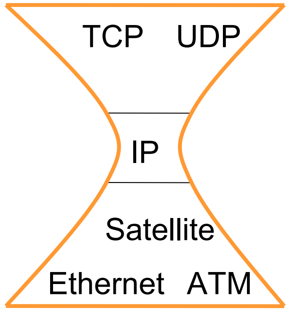
[Blumenthal and Clark, 2001]

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!

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

Before we go on:
Architecture components, concepts, principles

- Protocol machines
- Packets, continuous data stream
- PDUs
- Connection-oriented, connectionless; circuit-switched, packet-switched
- Layer abstraction
- Routing, forwarding
- Data plane, signalling plane
- In-band vs. out-of-band; separation of control and data
- Addressing, naming
- Lookups, indirection
- Virtualisation
- Flow control, congestion control
- Error correction, error recovery
- Randomisation
- Multiplexing
- Unicast, multicast, broadcast; point-to-point, point-to-multipoint

Where have we been?

Design Principles

- separation of control/data (signaling, ftp, http)
- randomization (CSMA-CD, router synch, routing)
- indirection (multicast, mobile IP, i**3)
- multiplexing: packet level (WFQ, priority), burst level, call level (routing in telephone net)
- virtualization (Internet, IP-over-ATM, MPLS, VLAN, VPN)

What's inside a protocol?

- An application that wants to communicate (or: an upper layer)
- An interface that allows to communicate with another protocol instance (or: a lower layer)
- PDUs (protocol data units) that can be sent and received via the interface
- The protocol state machine: Tells what we sent and what we expect to receive next

- Basically, a protocol can be viewed as an IPC facility! (inter-process communication)

PDUs

- Format:
  - Header (usually)
  - Data
  - Trailer (not very often. Example: Ethernet CRC)
- Size:
  - Fixed? (Easier to parse) Variable? (Less waste of resources)
  - Large? Small? – It depends!
    - Make suitable for needs of application
    - Larger PDUs usually are more efficient in the network

Layers

- cf. ISO/OSI model, Internet model

Connections and circuits

Connection-oriented

- TCP
- phone network

Connectionless

- UDP
- SMS (from user perspective)

Circuit-switched

- Phone network

Packet-switched

- IP
- …thus, also TCP!

Setting up a connection

- Enrolment: Reserve memory, create data structures
- Establishment: Tell the other end that you want to communicate
- Synchronization: Negotiate parameters
- Data transfer

- Establishment and synchronisation usually joined together
  - No synchronisation: connectionless (e.g., UDP)
  - Two-way handshake
  - Three-way handshake (e.g., TCP)
  - Multi-round negotiations

Routing and forwarding

Forwarding

- Many nodes are not directly connected to each other
- Intermediate nodes have to *forward* (to relay, to switch,…) PDUs

- [N.B.: This is often referred to as “routing”, but it’s actually wrong…]

Routing

- Intermediate nodes have to know where they have to forward the PDUs to
- *Routing:* the process of (jointly!) determining the paths through the network and setting up the forwarding

Planes

- **Warning: plane ≠ layer!**

- Data plane
  - The part of the router (or network architecture) where the packets are *forwarded* to other nodes
  - High data volume
  - Processing done in hardware
- Signalling plane
  - The part of the router (or network architecture) where the routes are set up and topology / other network information is exchanged with other nodes (*routing*)
  - Low data volume (…if not, then there’s something wrong)
  - Processing done in software
- Proposed concept: Management plane
  - A new, integrated part of the network architecture that allows to consistently manage the policies of ~all nodes in the network (i.e., their signalling planes)

In-band vs. out-of-band signalling

In-band signalling:
Protocol-related information in same channel as payload data

- Examples
  - HTTP: First headers, then the data
  - TCP: Headers control state machine, flow control, congestion control; data follows
  - IP: Routing protocols (e.g., OSPF, BGP) use IP packets to exchange information
- Assessment
  - Keeps things simpler
  - Processing can be less efficient
  - Don’t burn your bridges (e.g., router configuration via ssh…)

Out-of-band signalling:
Protocol-related information in channel separate from payload data

- Examples
  - FTP (traditional “active”): Commands on port 21, actual data on port 21
  - MPLS: only used for forwarding; but the tunnels are set up using an LDP (e.g., RSVP)
- Assessment
  - More channels: more complexity

Addressing and naming (1)

Address (Locator)

- Where is the destination of the PDU within the network's topology?
- Examples:
  - IP addresses (more precisely: IP prefix)
  - Phone numbers

Name (Identifier)

- Identify…
  - Hosts within same "area" of network (e.g., broadcast segment)
  - Processes within one host
- Examples:
  - IP addresses ☹
  - Names in phone book
  - DNS entries
  - Search keywords in filesharing networks, Web page search (Google), …

Addressing and naming (2)

Hierarchically

- Addressing
  - IP addresses (network prefixes)
  - Phone numbers (country code + area code [+ in old analogue times: initial digits within number] + MSN [+extension])
  - Helps structureing the network
- Naming
  - DNS (. .de .tum.de .in.tum.de .net.in.tum.de)
  - People (Family name, First name, perhaps middle names depending on culture)
  - Facilitates distributed administration

Flat

- Addressing
  - MAC addresses (N.B. vendor prefixes do not serve any naming purpose
  - IP addresses within one broadcast domain
  - Nowadays: Mobile phone numbers (international roaming; keeping the number after provider change)
  - AS numbers
- Naming
  - GPG/PGP keys
  - Search keywords for Google
  - People's first names
  - Hosts within same network

Lookups, Indirection

- Lookup services
  - Convert names to addresses
    - DNS
    - Phone book
    - Google! (Search keywords to URLs)
  - Convert addresses to other addresses
    - ARP (IP addresses to MAC addresses)
- Indirection
  - N.B.: The URL example showed that the same thing can be an address as well as a name. Other examples:
  - Mobile IP (home agent points to actual location)
  - HTTP Redirects (status codes 301, 302, 303, 307)
  - Multicast! (One address represents an entire group of hosts)

Virtualisation, overlay networks

- Create new functionality on top of existing functionality
  - "*on top*": layer-wise perspective
  - Compare terms to host virtualisation: "Windows VM running *inside* a Linux machine"
- Examples
  - Skype, P2P file sharing networks, Tor build a peer-to-peer overlay consisting of TCP and UDP connections on top of the existing Internet
  - PlanetLab builds a world-wide experimentation test bed on top of the existing Internet
  - MPLS builds a virtual network on top of layer 2
  - The Internet
    - was originally an overlay on top of the phone network (WAN connections = modem lines)
    - is still is an overlay on top of various different network technologies (Ethernet, WLAN, GSM/3G/UMTS/LTE, SONET/SDH, …)
      - Layer 2 topology may look vastly different from what we can see when we do traceroute (layer 3)
  - A rather **daring** assertion: TCP builds a lossless in-order virtual byte stream service on top of the lossy no-order datagram-oriented IP service. (Many wouldn't consider this to be virtualisation, though.)

Error detection, error recovery

- Error types
  - Corruption (bit flips; e.g., due to radiation)
  - Loss (entire PDUs missing; e.g., dropped during congestion)
  - Number of occurences: Just one flip / drop, or a burst of flips / drops
- Error detection
  - Checksums (e.g., CRC)
    - IP header
    - Ethernet
  - Byte/PDU counters
    - TCP (segment#, ACK#)
  - Timeouts
    - TCP
    - DHCP
- Error recovery
  - Just ignore it and try your best (GSM voice codec)
  - Retransmission (TCP)
    - Needs retransmission control
  - Forward error correction: Transmit slightly redundant signal that allows to restore full information if only a small bit of information is lost.
    (Think of it as doing RAID-5 on packets.)

Flow control, congestion control

- Flow control: Don’t overwhelm the receiver with too much data
  - E.g., fast Web server sending data to a small phone with slow CPU
- Congestion control: Don’t overwhelm the network with too much data
  - E.g., fast Web server connected to Internet via 2 Mbit/s DSL line

- Cf. TCP lecture

Randomisation

- Sometimes, determinism would just be too expensive
- In that case, use clever (!) randomisation
- Examples:
  - Backoff timer in Ethernet, DHCP
  - Key generation in cryptography

- Think about this:
  - 10 Mbit/s TokenBus:
    - Deterministic bus access, no collisions
    - Can use full hardware speed
    - Expensive, has been dead for decades
  - 10 Mbit/s Ethernet with CSMA/CD:
    - Randomised bus access tries to minimise collisions
    - The more collisions, the more bandwidth is wasted
    - Cheap, widely used, offspring protocols live on in 100 Gbit/s Ethernet (although without CSMA/CD, but switched!)

Multiplexing

- Combine multiple different signals together and send them across the same media
- At the other hand, we have to employ demultiplexing
- Examples:
  - Many different TCP connections across one Internet link (e.g., your computers at home are connected via one DSL line to the I’net)
  - Multiple wavelengths in one optical fiber can be used for different circuits
  - Two separate voice circuits across one single ISDN line
  - Multiple TV programs within one DVB signal

Fragmentation, reassembly

- Usually a consequence of
  - Multiplexing/demultiplexing
  - Layering
- Examples:
  - IP packets being fragmented, reassembled at receiver
  - IP packets being fragmented transparently into 48 byte ATM cells, reassmebled when exiting the ATM network

Ordering

No ordering

- Easier for network
- Perhaps harder for receiver
- Example: IP packets

Ordering

- Easier for receiver
- Example: TCP bytes

When to communicate

- Synchronous operation
  - Data transferred at fixed points in time
  - Usually on lower layers
  - Example: mobile phone networks (TDMA)
- Push
  - Sender just pushes data to receiver, whether wanted or unwanted
  - Example: IP (usually wanted, but DoS traffic is not wanted…)
- Pull (request/response)
  - Receiver requests data, sender sends desired data
  - Example: HTTP
- Publish/subscribe
  - Receiver describes the data he's interested in
  - Every time the sender comes across data matching the description, it is forwarded to the receiver
  - ~"Asynchronous request/response"
  - Example: RSS

*-cast, <X>point-to-<Y>point

- Unicast
  - "Normal" case: one node sends data to one other
- Multicast
  - One node sends data to specific group of other nodes
- Broadcast
  - One node sends data to all other nodes

- Point-to-point
  - Channel between two nodes
  - Usually, unicast
- Point-to-multipoint
  - One node communicates with many others
  - Usually, multicast/broadcast and replies via unicast
- Multipoint-to-multipoint
  - Nodes within a group communicate with each other
  - Replys also via multicast/broadcast

Security (1): Authentication, authorisation

Authentication

- Prove your identity
- Examples:
  - GPG signature under your e-mail
  - Entering correct login and password

Authorisation (access control)

- Depending on who you are, obtain access (or not)
- Examples:
  - File access rights in network file system
  - Access / no access to shared IMAP folders

Security (2)

- ❑ Integrity
  - ▪ Protection against unauthorised insertion or deletion of PDUs
  - ▪ Example: Transfer ***20,000€*** from account A to B
- ❑ Confidentiality
  - ▪ Ensure that contents of PDUs cannot be read by unauthorised parties
- ❑ Nonrepudiation (non-deniability)
  - ▪ Ensure that a communication party cannot deny that it has participated in a conversation
  - ▪ Examples:
    - Signed E-Mail (key identity-checked): non-deniable
    - OTRS encrypted Jabber conversation: deniable (design goal!)

Resilience

- ❑ The ability of a system to withstand failures, disruptions, and other challenges
- ❑ Acceptable network and service quality even under severe disruptions
- ❑ “Acceptable”: relative; depends on application and users
- ❑ Example #1: IP routing *within* a provider’s network
  - ▪ …is resilient: 500ms – 1s convergence time is acceptable (home end users reading e-mail, surfing the Web, downloading files,…)
  - ▪ …is not resilient: 500ms – 1s convergence time is utterly inacceptable (professional end users: online trading, telemedicine, video conferences)
- ❑ Example #2: 99.99% guaranteed availability for a provider’s network
  - ▪ Let’s see… 99.99% · 365 days = at most 1 hour downtime / year
  - ▪ Resilient for nor mal home end users
  - ▪ Not resilient for business users
- ❑ Example #3: VDSL line (60Mbit/s) with GSM backup line (384kbit/s)
  - ▪ Resilient for home end users (YouTube is slow, e-mail still works)
  - ▪ Not resilient for a small office with 30 employees

Back to the future Internet!

- ❑ Re-arrange some of these fundamental building blocks?

or

- ❑ Integrate them […anew, in abridged shape…] into the existing architecture?

Future Internet: how?

- Evolutionary approach
  - Tackle one problem at a time
  - Incrementally introduce new protocols to overcome weaknesses
    - IPv6
    - SCTP, DCCP
    - LISP, HIP, Mobile IPv6
  - Advantage: backwards compatibility
  - Disadvantage: Legacy burden; sometimes a radical cut is needed
- Revolutionary approach (“clean slate”)
  - Throw away the old architecture, and build a radically new one
  - Advantage: no legacy burden; more freedom to create sth. new
  - Disadvantage: it just won’t happen!

- Perhaps the two are more or less the same… (cf. next slide)

Where are we headed: Current/upcoming research topics

- Network management:
  - Measurement, automation (“management plane”)
  - Reflecting the fact that ISPs business entities (“tussle space”)
- 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:
  - Today: Lack of cryptographic signatures in many protocols
  - Today: Most traffic unencrypted (…good for measurements…)
  - Difficult: Accountability, non-repudiability, traceability vs. anonymity
- Resilience: more robust networks and services (reacting faster, keep up acceptable service quality under even more disruptive failures)
- Ease of use, deployment (but what are the research problems here?)

Future Internet: Some radical concepts

- Source routing in the core
- DHT-based routing and lookups
- Freely pluggable building blocks instead of fixed layers
- Content-centric networking

Radical concept: Source routing

- Current Internet: Routing purely destination-based
  - Hand your packet to the next hop and trust it will make a good decision
- Proposal: Source routing
  - Sending AS (*not:* sending host!) prescribes the exact route to receiving AS in packet header
  - If an intermediate AS doesn’t like the route (←policy), it can drop the packet
  - Advantage: Do not rely on (unreliable, misconfigured, buggy) ASes along the path
  - Challenges: Security issues, accounting, …

Radical concept: Using DHTs for basic networking functions

- DHT (Distributed hash table):
  - Imagine a hash table. It has two operations:
    - put(key, object)
    - object = get(key)
  - Now imagine the data structure to be distributed among thousands of nodes. That’s a DHT.
- Remember the architecture slide on *lookup* functionality? A DHT can be used for any of these, e.g.,
  - Mapping names to addresses
  - Higher-level indirection
  - Mapping addresses to routes
  - Storing network topology information
  - Storing routing policy information
  - …

Radical concept: Make layers more flexible

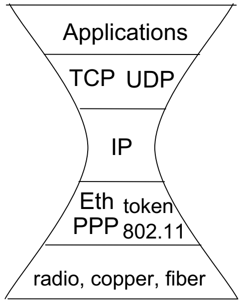
- Each application has different requirements concerning, e.g.,
  - Reliable ↔ non-reliable delivery
    - Retransmissions, FEC, timeouts, when to ignore errors
  - In-order ↔ unordered delivery
  - Congestion control ↔ predetermined bit rate
  - Flow control ↔ no flow control
  - Datagram delivery ↔ byte stream delivery
  - Connection-oriented ↔ connectionless
  - Integrity, confidentiality, authenticity, nonrepudiability, …
- Each media offers its own service
- Some ideas:
  - Application specifies requirements, network automatically transmits data using appropriate protocols
  - No fixed layers, but flexible building blocks (e.g., reliable data transfer module, flow control module, …). Application specifies how they should interact.

Radical concept: Content-centric networking

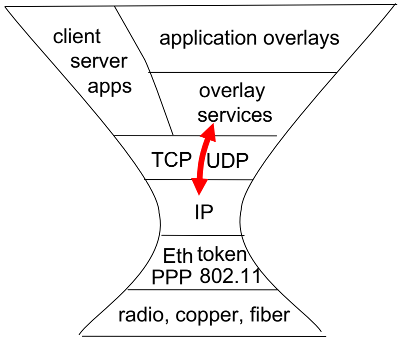
- Today’s Internet:
  - Mainly request/response (e.g., HTTP)
  - Addressing: host-based / process-based (IP addresses, ports). Receiver has to know where it should send the request to.
- Content-centric networking:
  - Publish/subscribe: The receiver tells the network what kind of data it is interested in.
  - Addressing: content-based.
    - Simple example: specify the DOI
    - More elaborate: specify keywords associated with desired content (think of adding a “Google layer” on top of the network…)
    - Futuristic: semantic description of content
  - Advantage: Replicating/caching is easy, since we address data, not hosts.
    - Replication fosters load balancing
    - Replication increases resilience (no single source of failure)
    - Replication can reduce delays
  - Interesting questions: How to do routing, how to invalidate / withdraw / update cached data, what about dynamic data (e.g., dynamic Web pages), confidentiality / authenticity / integrity, business/economic aspects and policies, legal issues, how to emulate sessions like ssh or telephony (it’s possible: subscribe to ACKs of the other end), …

Revolutionary Future Internet in an evolutionary development

- Let’s just build the Future Internet as an overlay on top of the old one!
- After all, the Internet started as an overlay of the phone network…



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……

Seven Cautionary Questions

Q2: Is this really a bottleneck?

- ❑ 80% of gains achievable by focusing on 20% of system
- ❑ use profiling tools to see where time is spent

Seven Cautionary Questions

Q3: Effect of change on rest of system?

- does change increase performance in one place but slow down in other places?

Q4: Does an initial analysis indicate potential significant improvement is possible?

- is there room for improvement?
- how close to best possible performance ? Think about *bounds,* solutions (e.g., oracle) with unachievable performance

Seven Cautionary Questions

Q5: Is it worth adding custom hardware?

- ride Moore's curve (doubling of processing speed every 18 months) or use specialized hardware?

Q6: Can protocol changes be avoided?

- Rather than scrap existing protocol, tweak/rethink it to solve problem?
- Example: TCP's imminent demise predicted many times (e.g., TCP too slow for high-speed implementation)

Seven Cautionary Questions

Q7: Does prototype confirm initial promise?

- initial high-level analysis will miss details that could be important
- some people will never be convinced without an implementation

Q8: Will performance gains be lost if environment changes?

- think about if improvements limited to small number of environments
- example: same-connection, in-order packet assumptions won't hold in busy server.

More cautionary questions…:

What problem am I trying to solve?

- have at least one well-defined problem in mind
- solve other problems without complicating solution?

Will my solution scale?

- Think about what happens if you're successful: protocol is used by millions
- Does the protocol make sense in small situations as well?

More folklore/advice

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

More folklore/advice

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

Properly parameterized?

- 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 (good-natured initial values)

Challenge: on beyond the data plane

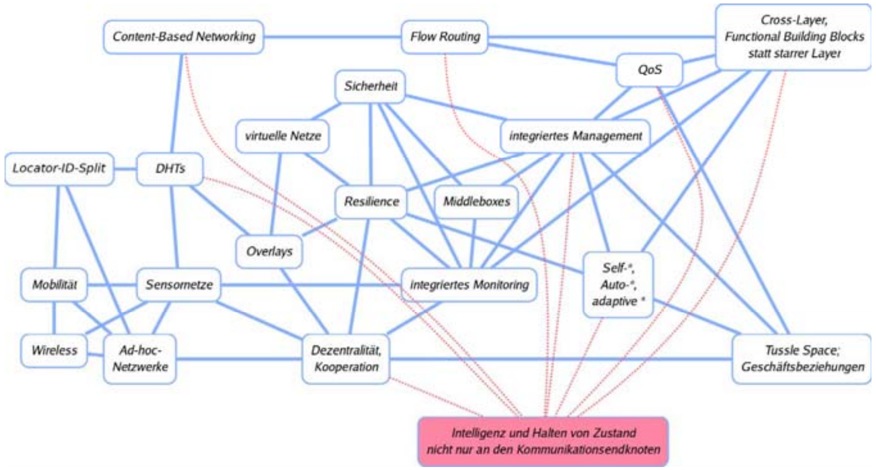
- Q: data plane performance really *the* major roadblock?
  - "robustness"
  - "complexity of control"
  - maintainability
  - evolvability
  - adaptability
  - reconfigurability
  - security
  - manageability

the "X-ities"

- Fundamental advances here are hard!
  - "efficiency" not always the most important measure
  - little/no past work on the "X-ities"
  - metrics and models still to be defined

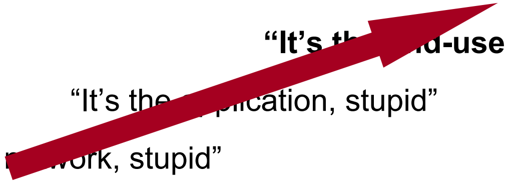
Future Internet

(sorry for the German labels, but most notions are in English anyway…)


Content-Based Networking
Flow Routing
Cross-Layer, Functional Building Blocks statt starrer Layer
QoS
Sicherheit
virtuelle Netze
integriertes Management
Locator-ID-Split
DHTs
Resilience
Middleboxes
Overlays
Mobilität
Sensornetze
integriertes Monitoring
Self-*, Auto-*, adaptive *
Wireless
Ad-hoc-Netzwerke
Dezentralität, Kooperation
Tussle Space; Geschäftsbeziehungen
Intelligenz und Halten von Zustand nicht nur an den Kommunikationsendknoten

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 wireless
Mobile Office Kit
Use this kit to link your laptop to the Internet through your BroadbandAccess Connect capable phone
It's the Network.

Verizon product, purchased 2007

The end!