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

Introduction

Intended Learning Outcomes and Competences

- Goals of the course
  - Learn to take responsibility for yourself
  - Think about the topics
    (do not repeat content of theses slides without deeper understanding)
  - Learn to formulate and present technical problems
  - Understand the principles
    - What is the essence to be remembered in some years?
    - What would you consider suitable questions in an exam?
  - Learn from practical project performed during course

General Learning Outcomes

- Knowlege
  - Being able to reproduce facts
- Understanding
  - Being able to explain properties with own words
- Applying
  - apply known methods to solve questions
- Analyzing
  - Identifying the inherent structure of a complex system
- Synthesis
  - Creating new solutions - from known elements
- Assessment
  - Identifying suitable criteria and perform assessment

Learning Outcomes
- what students are expected to acquire from the course

- ❑ Knowledge, Understanding, Applying
  - ▪ protocols:
    application layer, transport layer, network layer, data link layer
  - ▪ concepts:
    measurements, signalling, QoS, resilience

  ⇨lectures, exercise questions
  final examination
- ❑ Applying, Analyzing, Synthesis, Assessment
  - ▪ special context: IPv6 vs. IPv4, DNS, tunneling
  - ▪ tools: svn, measurement tools, ...
  - ▪ methods: plan, configure, administer system and network, measure, program, reflect

  ⇨course project

Course Outline

- Part 1: Internet protocols

  Overview on Computer Networks
  Internet Core Technologies
  Network Layer
  Transport Layer

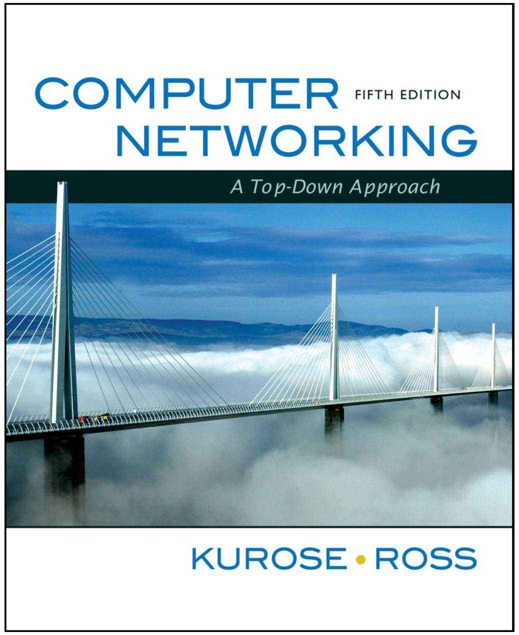
- Part 2: Advanced Concepts

  Signalling
  Resilience
  Node Architectures and Mechanisms
  Quality of Service Support
  Measurements
  Design Principles and Future Internet

Acknowledgements - Book Recommendation

- *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

- Time slots
  - Friday, 10:15-11.45, MI H2
  - Monday, 16:15-17.45, MI H2
- TUMonline: registration required (for exam registration + Email)
- Students are requested to subscribe by October 30, 2011 in groups of two for project work at
  http://www.net.in.tum.de/en/teaching/ws1011/vorlesungen/masterkurs-rechnernetze/
  ⇨ link to registration form for svn access
- 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
    - Drop in or by appointment.
- Course Material
  - Slides made available online (may be updated during the course).

Course Overview

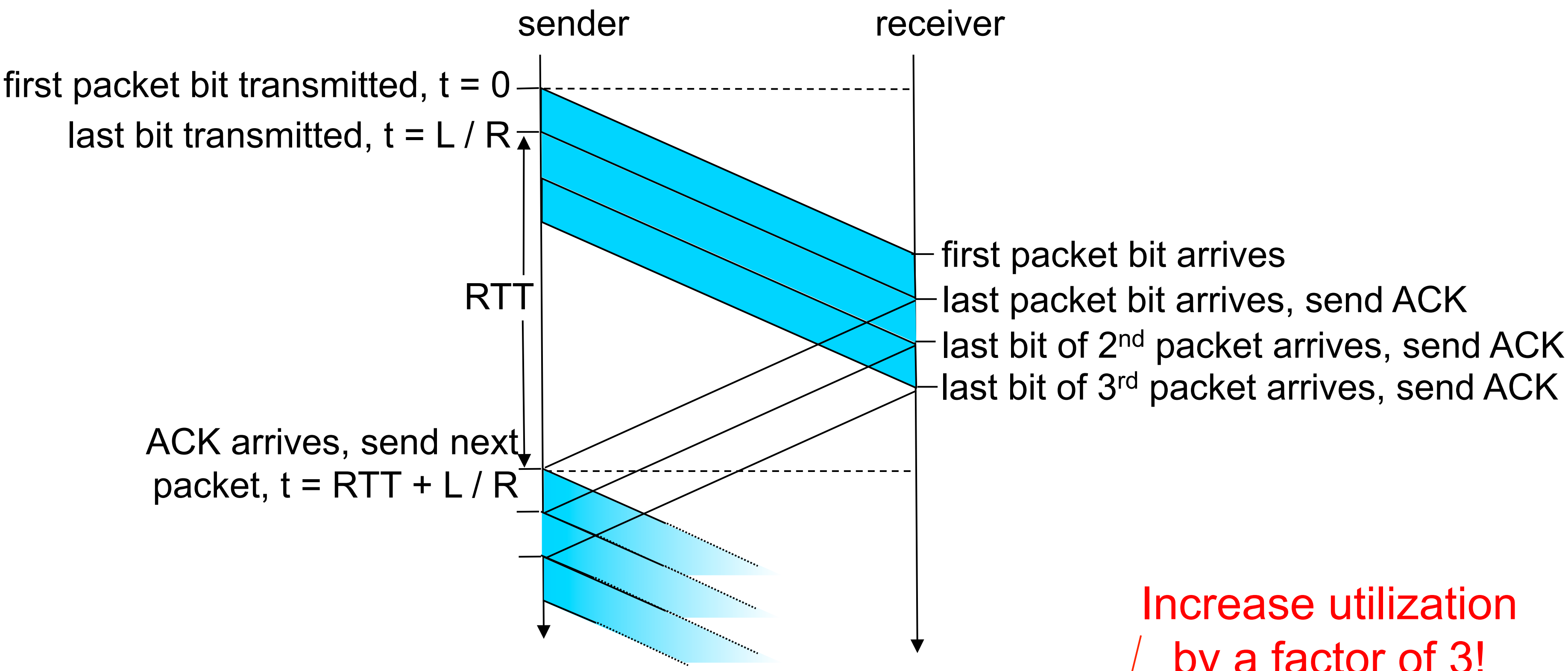
Internet Core Technologies

- DNS
- Tunneling
- IPv4
- IPv6

Chapter: Transport Layer Services

- ❑ 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

Pipelining for 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

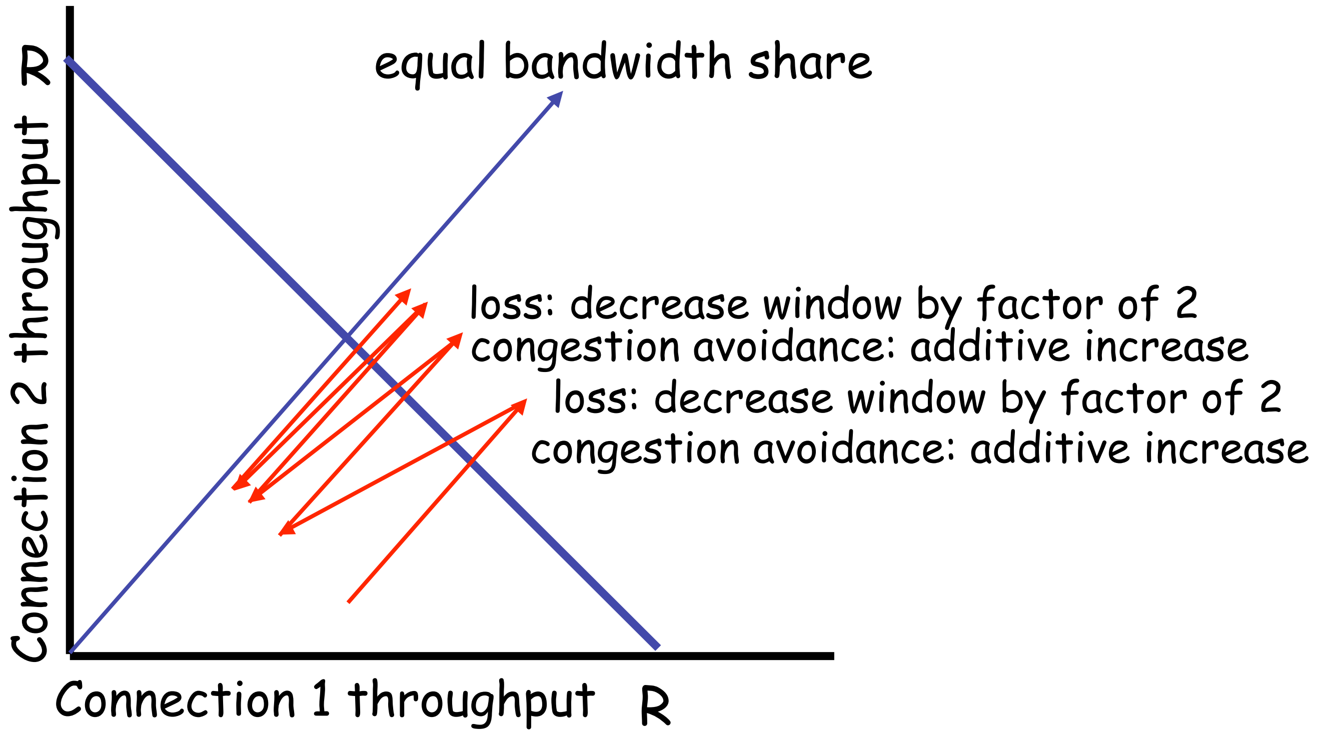
Increase utilization by a factor of 3!

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

Why is TCP fair?

Two competing sessions:

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

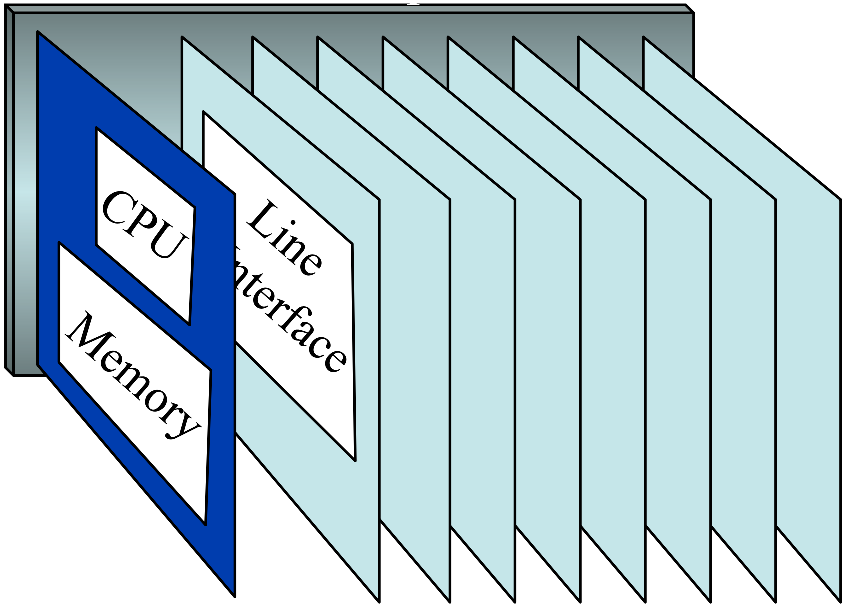


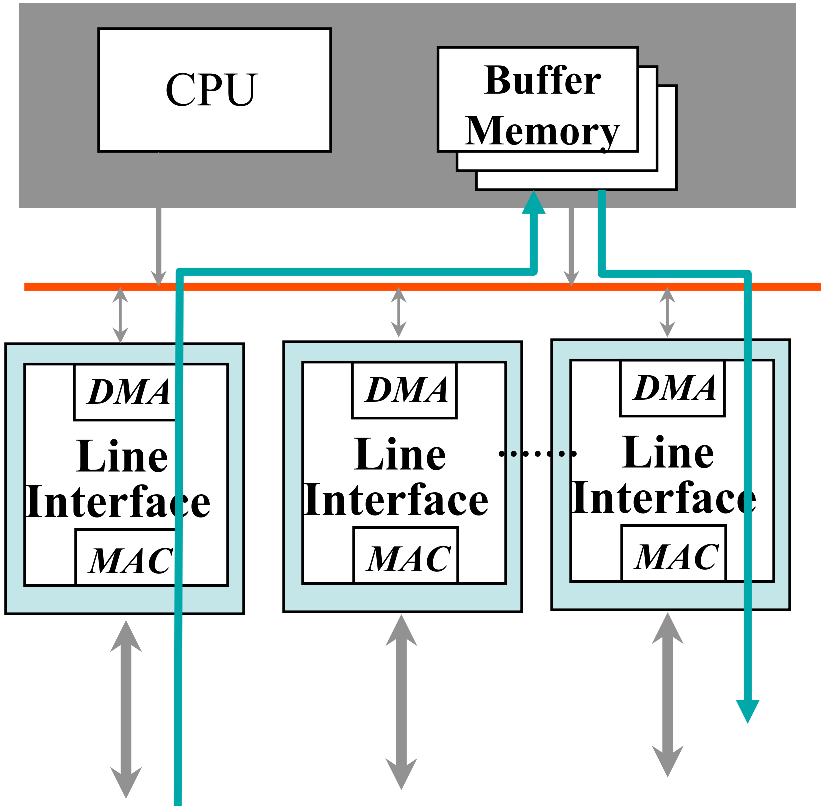
R
equal bandwidth share
loss: decrease window by factor of 2
congestion avoidance: additive increase
loss: decrease window by factor of 2
congestion avoidance: additive increase
Connection 2 throughput
Connection 1 throughput
R

Chapter: Network Layer - Routing

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

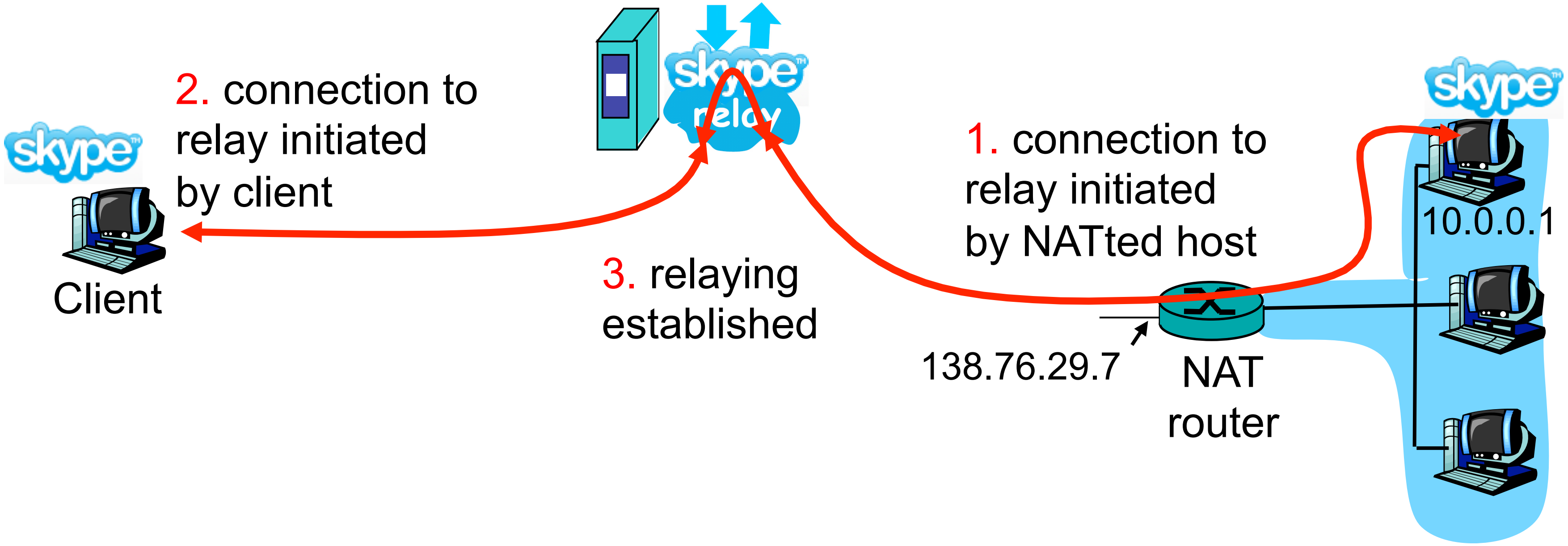
- First-Generation IP Routers





NAT Traversal

- One of several NAT traversal solutions: relaying (e.g. used in Skype)
  - NATed client establishes connection to relay node
  - External client connects to relay node
  - relay node forwards packets between two connections



2. connection to relay initiated by client

3. relaying established

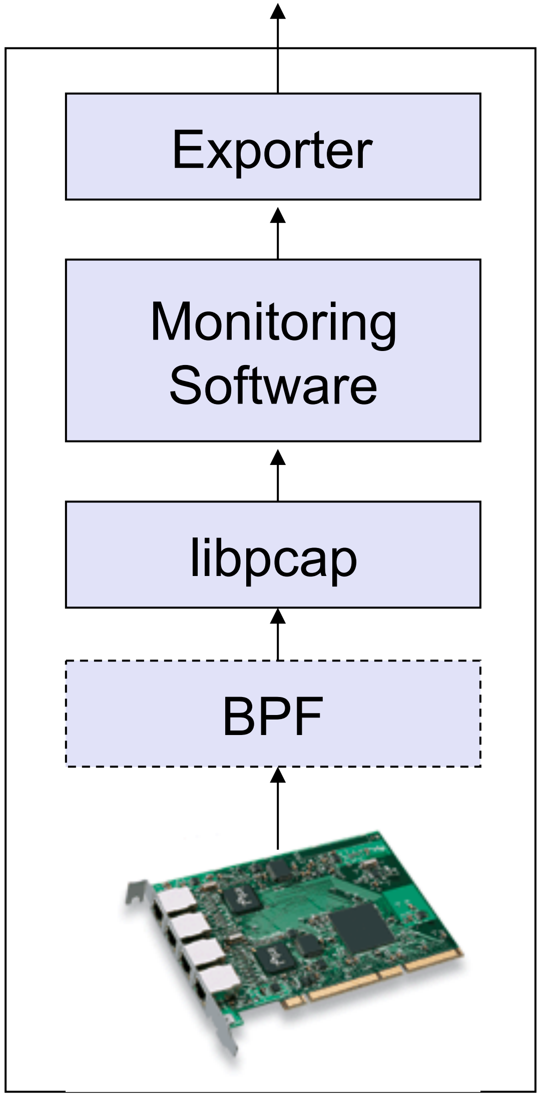
1. connection to relay initiated by NATted host

Network Measurements

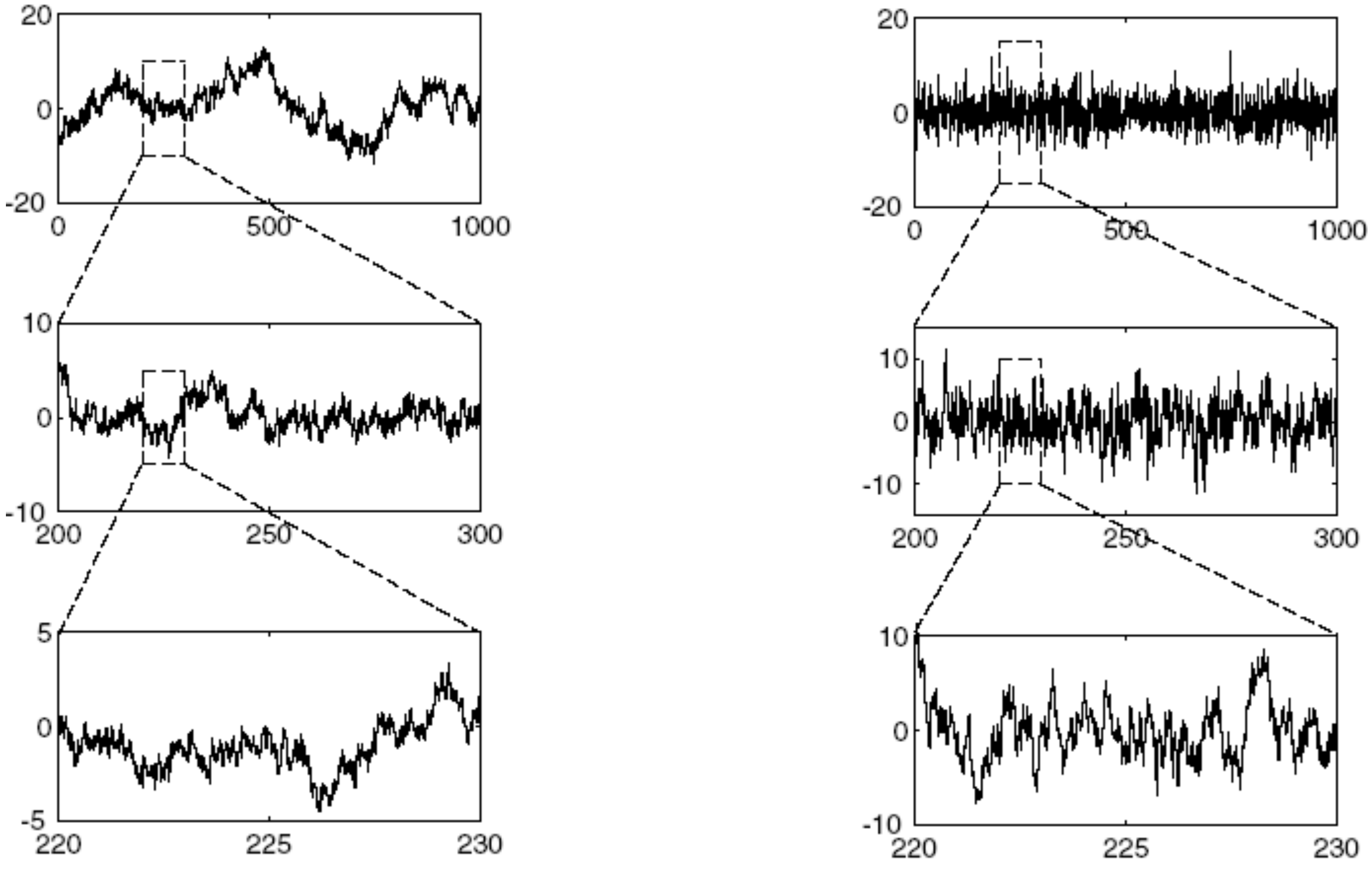
- Introduction
- Architecture & Mechanisms
- Protocols
  - IPFIX (Netflow Accounting)
  - PSAMP (Packet Sampling)
- Scenarios

Monitoring Probe

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



Self-Similar Stochastic Process



(a) Self-Similar Process

(b) Non-Self-Similar Process

Quality-of-Service Support

- Link virtualization: ATM
- Providing multiple classes of service
- Providing Quality-of-Service (QoS) guarantees
- QoS Architectures
  - Integrated Services
  - Differentiated Services

Chapter: 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 and SS7 (Signaling System no. 7): telephone network
  - Q.2931: ATM
  - RSVP (Resource Reservation Protocol)
  - H.323: Internet telephony
  - **SIP** (Session Initiation Protocol): Internet telephony

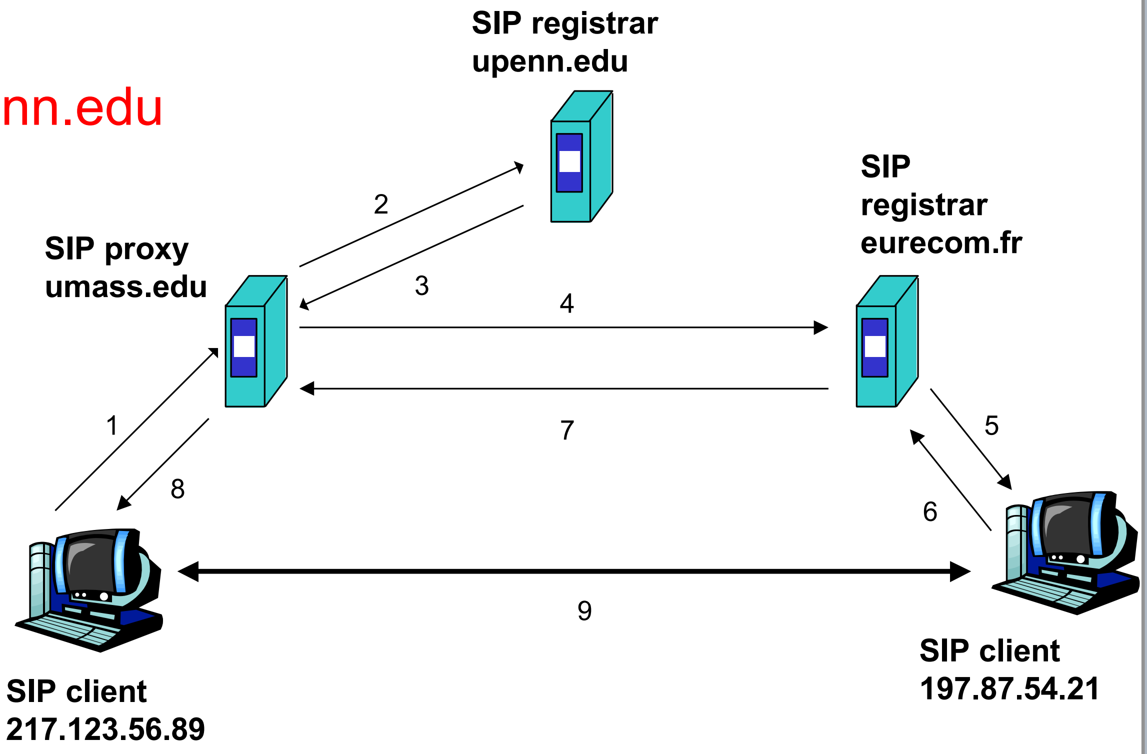
Voice over IP Example

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

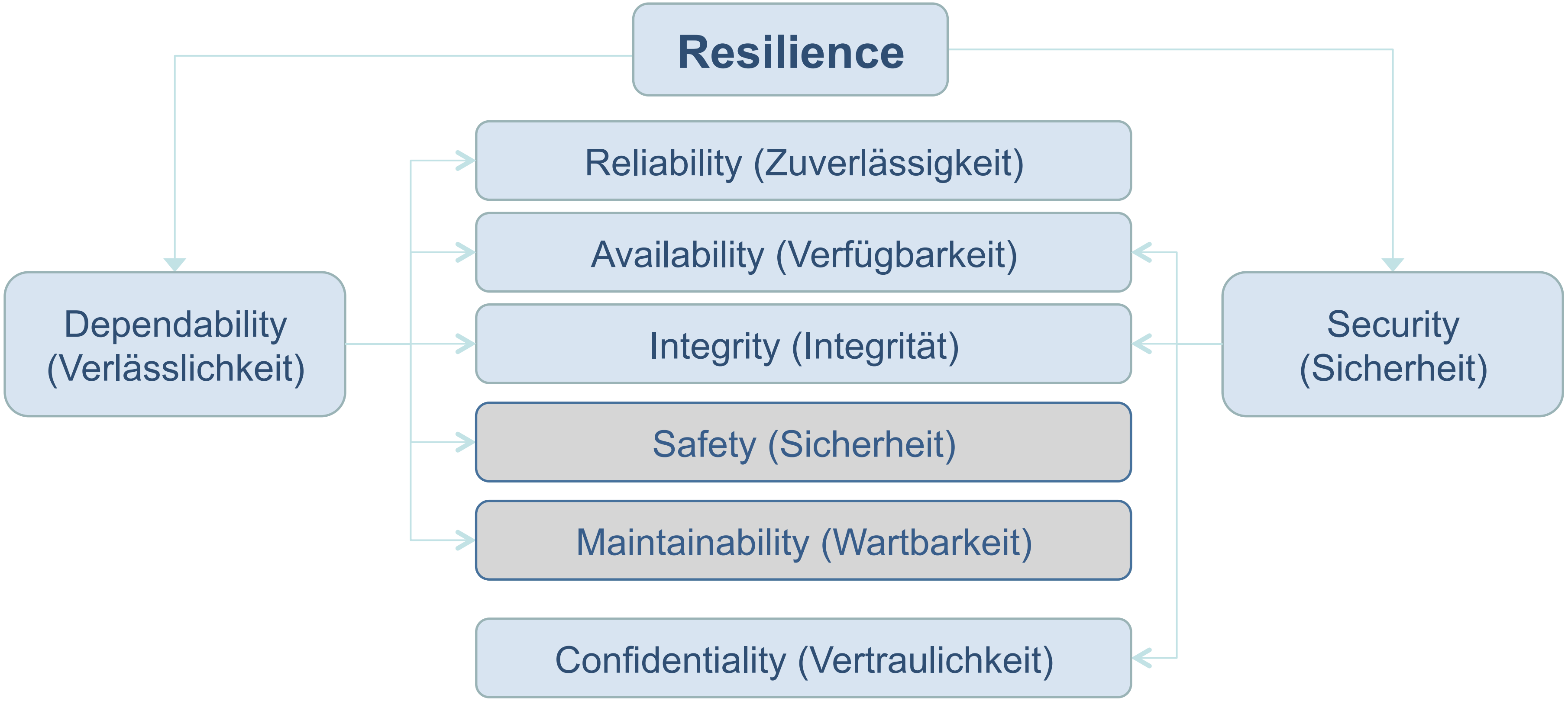
(9) media sent directly between clients.

Note: SIP ack messages not shown.

Chapter: Resilience

- <u>Definition</u>:
  - "Resilience is the persistence of *<u>dependability</u>* when facing *<u>changes</u>*."

- Changes can be particularly *attacks*



Resilience

Reliability (Zuverlässigkeit)

Availability (Verfügbarkeit)

Integrity (Integrität)

Safety (Sicherheit)

Maintainability (Wartbarkeit)

Confidentiality (Vertraulichkeit)

Dependability (Verlässlichkeit)

Security (Sicherheit)

Chapter: Design principles and Future Internet

- Network design principles
  - common themes: indirection, virtualization, multiplexing, randomization, scalability
  - implementation principles
  - network architecture: the big picture, synthesis

- Future Internet approaches

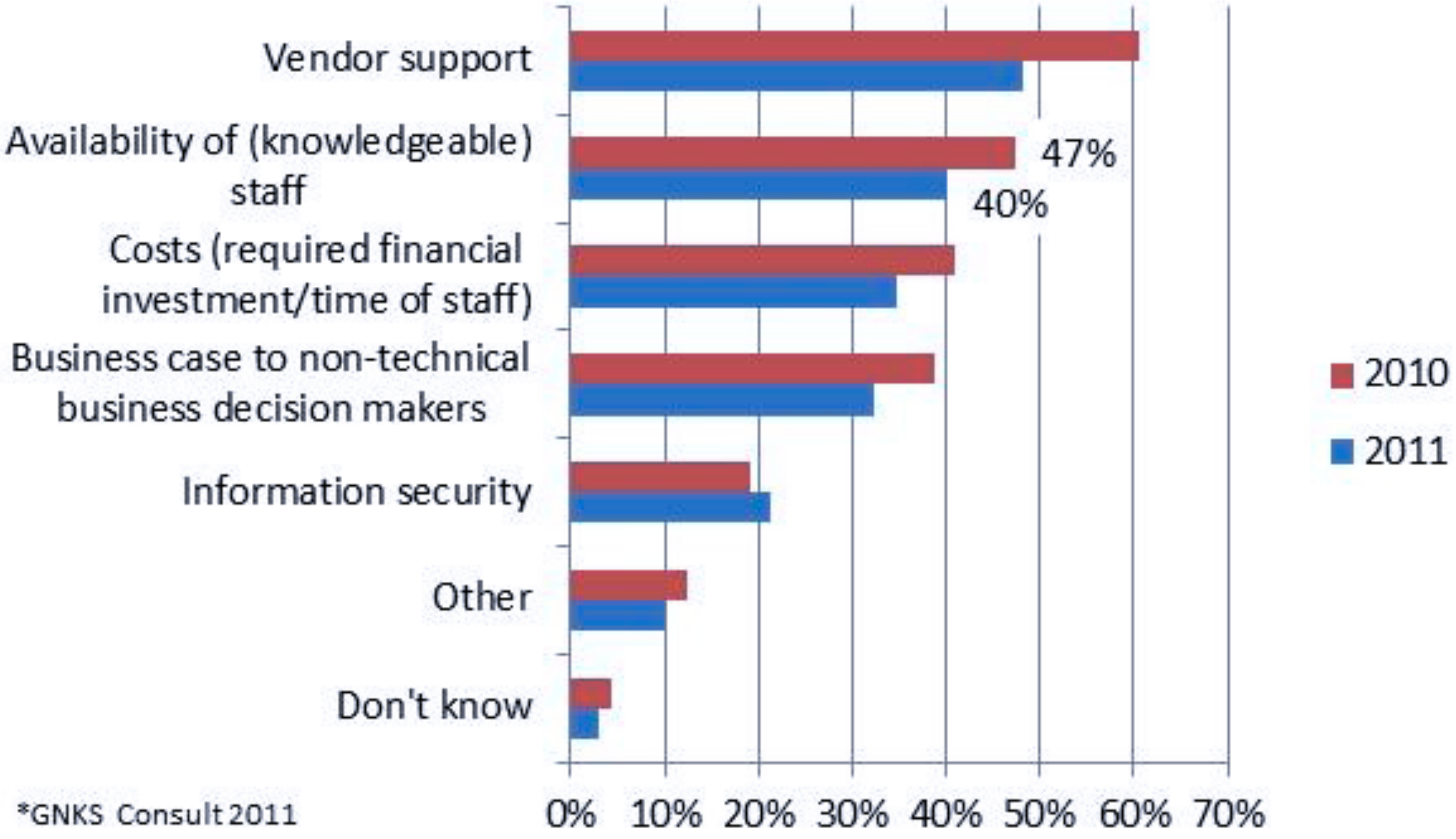
Chapter:
Internet Core Technologies

- See Slides by Christian Grothoff

IPv6 Deployment

Standardisation

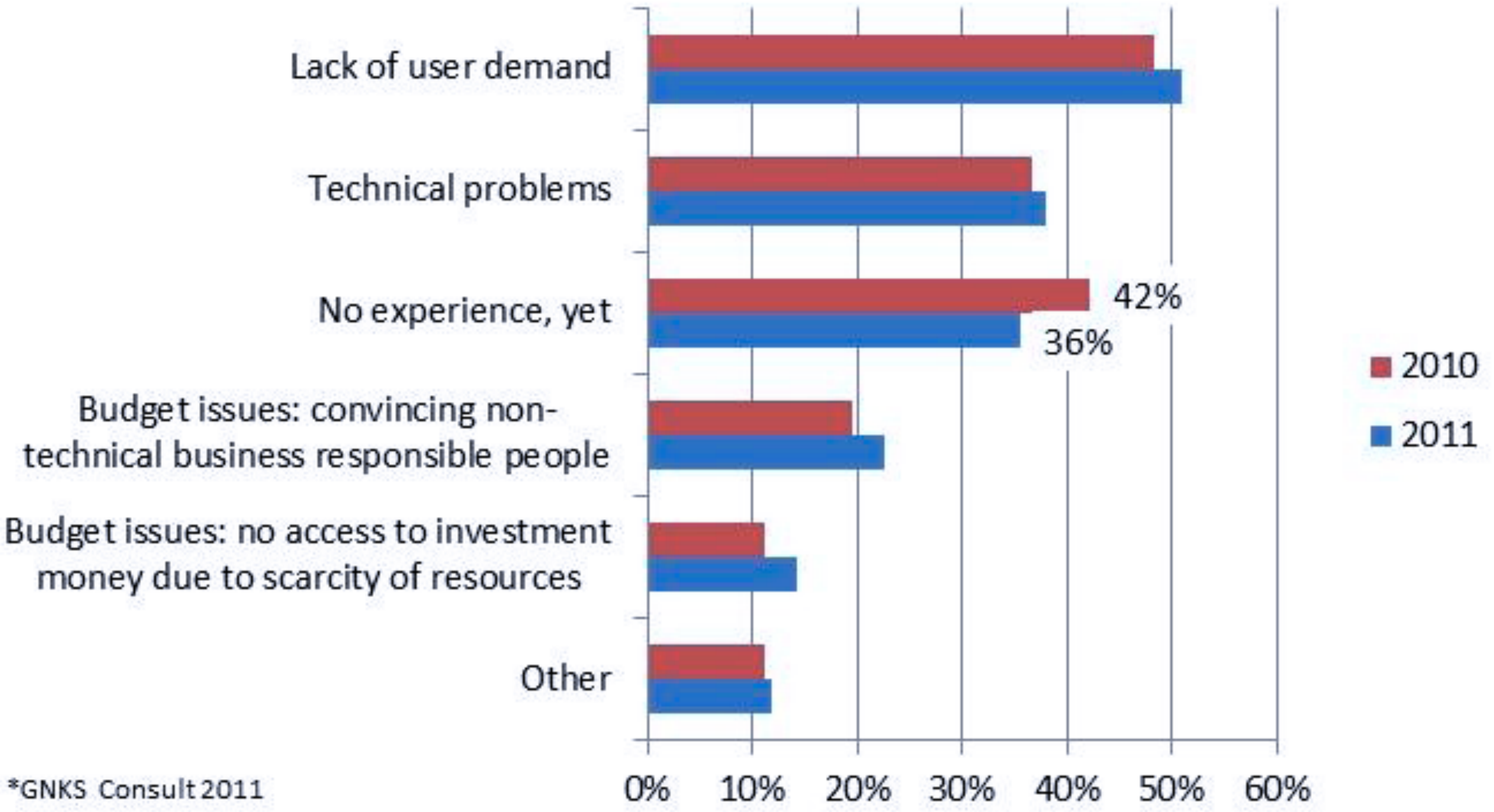
Biggest hurdles when deploying IPv6



*GNKS Consult 2011

- Maarten Botterman, GNKS Consult: Results of the 2011 Global IPv6 Deployment Monitoring Survey - Presentation at RIPE-63

Biggest problems with IPv6 in practice



*GNKS Consult 2011

RFC 2460: IPv6 Specification

- The routing header is used by an IPv6 source to list one or more intermediate nodes to be “visited” on the way to packet’s destination.
- Each extension header should occur at most once, except for the destination options header which should occur at most twice.
- IPv6 nodes must accept and attempt to process extension headers in any order and occurring any number of times in the same packet.

- c.f. Merike Kaeo, merike@doubleshotsecurity.com
  Presentation „IPv6 Routing Header Security “ - RIPE54 Meeting, Tallin, Estonia, May 2007

Router Configurations

- Cisco
  - "no ipv6 source-route„
- Linux
  - # Filter all packets that have RT0 headers
  - ip6tables -A INPUT -m rt--rt-type 0 -j DROP
  - ip6tables -A FORWARD -m rt--rt-type 0 -j DROP
  - ip6tables -A OUTPUT -m rt--rt-type 0 -j DROP
  - (of course before accepting anything else ;)
- FreeBSD
  - Upgrade the kernel with at least the following patch in place:

    http://www.freebsd.org/cgi/cvsweb.cgi/src/sys/netinet6/route6.c.diff?r1=1.12&r2=1.13

Routing Header Processing

- Disabling IPv6 type 0 routing header processing still allows other nodes to be used for attack
- Dropping is required for ISP's
- RFC 5095 - deprecate [„ablehnen“/“missbilligen“]

```
Network Working Group                                        J. Abley
Request for Comments: 5095                                    Afilias
Updates: 2460, 4294                                         P. Savola
Category: Standards Track                                   CSC/FUNET
                                                      G. Neville-Neil
                                            Neville-Neil Consulting
                                                        December 2007

Deprecation of Type 0 Routing Headers in IPv6

Abstract

   The functionality provided by IPv6's Type 0 Routing Header can be
   exploited in order to achieve traffic amplification over a remote
   path for the purposes of generating denial-of-service traffic.  This
   document updates the IPv6 specification to deprecate the use of IPv6
   Type 0 Routing Headers, in light of this security concern.
```

IETF Structure and Internet Standards Process

Scott Bradner

Harvard University
http://www.sobco.com/sob/sob.html

77th IETF - March 2010
Anaheim, California, USA

The IETF - Internet Engineering Task Force

- Formed in 1986
  - evolved out of US government activities
  - ARPA's Internet Configuration Control Board (ICCB) (1979) and Internet Activities Board (1983)
- Was not considered important for a long time - good!!
- Not government approved - great!!
  - but funding support from U.S. Government until 1997
- Specifications always available without charge (vs. ITU-T, IEEE)
- People not companies

"We reject kings, presidents and voting.

We believe in rough consensus and running code"

Dave Clark (1992)

IETF Organisation

- 1K to 2K people at 3/year meetings (many more on mail lists)
- >100 working groups with working group chairs
- 8 areas with Area Directors (ADs):
  GEN, APS, INT, O&M, RAI, RTG, SEC, TSV:
  - IETF Chair & AD for General Area (gen) - 0 WGs
  - Applications (app) - 15 WGs
  - Internet (int) - 28 WGs
  - Operations & Management (ops) - 15 WGs
  - Real-time Applications and Infrastructure (rai) - 19 WGs
  - Routing (rtg) - 16 WGs
  - Security (sec) - 17 WGs
  - Transport Services (tsv) - 14 WGs
- Internet Enginnering Steering Group (IESG): ADs + IETF Chair
- Internet Architecture Board (IAB): architectural guidance, liaisons
- IETF produces standards and other documents

Working Groups

- no defined membership
  - just participants
- "*Rough consensus* *and running code...*"
  - no formal voting - can not define constituency
    - can do show of hands or hum - but **no** count
  - does **not** require unanimity
  - chair determines if there is consensus
  - disputes resolved by discussion
  - mailing list and face-to-face meetings
  - final decisions must be verified on mailing list
    - to ensure those not present are included
      - but taking into account face-to-face discussion
- sessions are being streamed & recorded

IETF Standardisation Procedure

- Proposals published as Internet Drafts (ID)
- Worked on in a Working Group (WG)
- WG sends to IESG request to publish an ID ‘when ready’
- proposal reviewed by AD
  - can be sent back to working group for more work
- IETF Last-Call
- IESG review
  - last call comments + own technical review
  - can be sent back to Working Group for more work
- publication as RFC

RFC Repository Contains:

- ❑ standards track
  - ▪ OSPF, IPv6, IPsec ...
- ❑ obsolete Standards
  - ▪ RIPv1
- ❑ requirements
  - ▪ Host Requirements
- ❑ policies
  - ▪ Classless Inter-Domain Routing
- ❑ april fool's day jokes
  - ▪ IP on Avian Carriers ...
  - ▪ ... updated for QoS

- ❑ poetry
  - ▪ 'Twas the night before startup
- ❑ white papers
  - ▪ On packet switches with infinite storage
- ❑ corporate documentation
  - ▪ Ascend multilink protocol (mp+)
- ❑ experimental history
  - ▪ Netblt
- ❑ process documents
  - ▪ IETF Standards Process

Standards Track RFCs

- Best Current Practices (BCP)
  - policies or procedures (best way we know how)
- 3-stage standards track (not all that well followed)
  - Proposed Standard (PS)
    - good idea, no known problems
  - Draft Standard (DS)
    - PS + stable
    - multiple interoperable implementations
    - note: interoperability not conformance
  - Internet Standard (STD)
    - DS + wide use
- *"The Internet runs on proposed standards"* – perhaps first said by Fred Baker, Cisco Fellow, IETF Chair 1996-2001

Challenge Interoperability

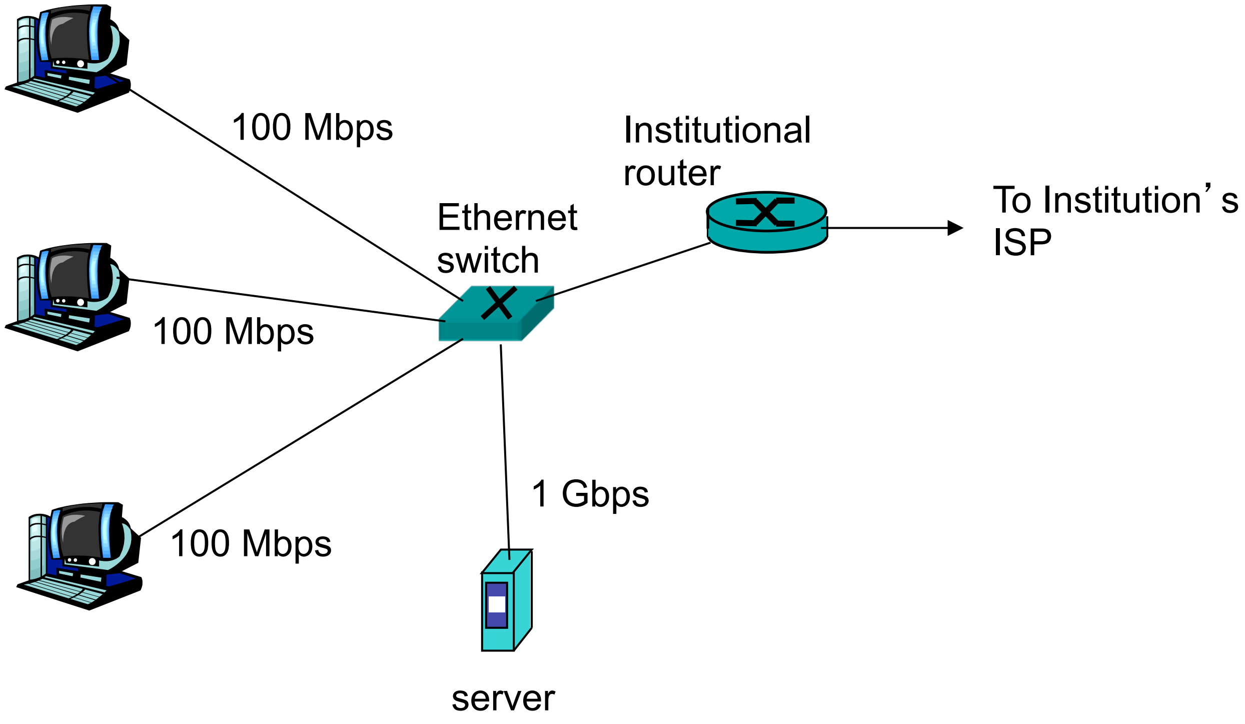
Example:
IPFIX Interoperability Test Event, 63rd IETF

- ❑ Participants
  - ▪ CISCO
  - ▪ IBM Research Zürich
  - ▪ NEC Laboratories Heidelberg
  - ▪ Fraunhofer FOKUS, Berlin
  - ▪ University team of Prof. Carle
    - c.f. RFC 3333, 5477, 5815
- ❑ Lession learned:
  Organisation of interoperability activities is useful. We do not necessarily need to organize joint meetings, but should make more of a habit of organizing joint testing, e.g. combined with chat sessions.

Delay, loss and throughput

Ethernet Internet access

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

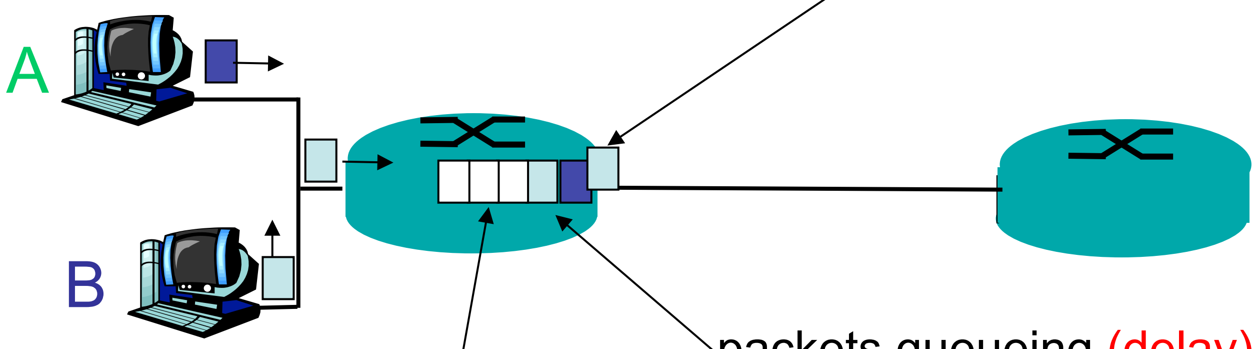


⇨ why?

Reasons for delay and loss

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

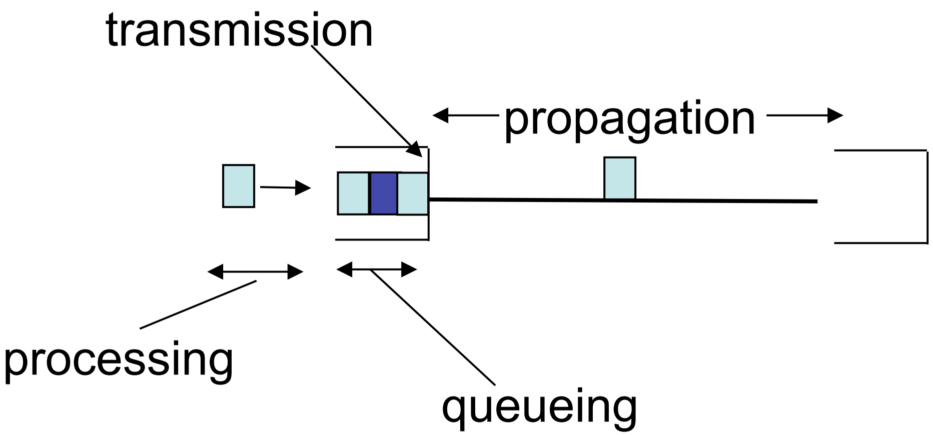
Background: Sources of packet delay

1. Processing delay:
   - Sending: prepare data for being transmitted
   - Receiving: interrupt handling

2. Queueing delay
   - time waiting at output link for transmission

3. Transmission delay:
   - L=packet length (bits)
   - R=link bandwidth (bps)
   - 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 microseconds (µs) or less
- d_{queue} = queuing delay
  - depends on congestion - may be large
- d_{trans} = transmission delay
  - = L/R, significant for low-speed links
- d_{prop} = propagation delay
  - a few microseconds to hundreds of msecs

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

Impact Analysis: Advances in Network Technology

Data rate	Delay (1bit)	Length (1bit)	Delay (1kbyte)	Length (1kbyte)
1 Mbit/s	1 us	200 m	8 ms	1600 km
10 Mbit/s	100 ns	20 m	0,8 ms	160 km
100 Mbit/s	10 ns	2 m	80 us	16 km
1 Gbit/s	1 ns	0,2 m	8 us	1600 m
10 Gbit/s	100 ps	0,02 m	0,8 us	160 m
100 Gbit/s	10 ps	0,002 m	80 ns	16 m

- Assessment
  - Transmission delay becomes less important
    ⇨ over time; in the core
  - Distance becomes more important
    ⇨matters for communication beyond data center
  - Network adapter latency less important
    ⇨ Latency of communication software becomes important

Propagation Delay

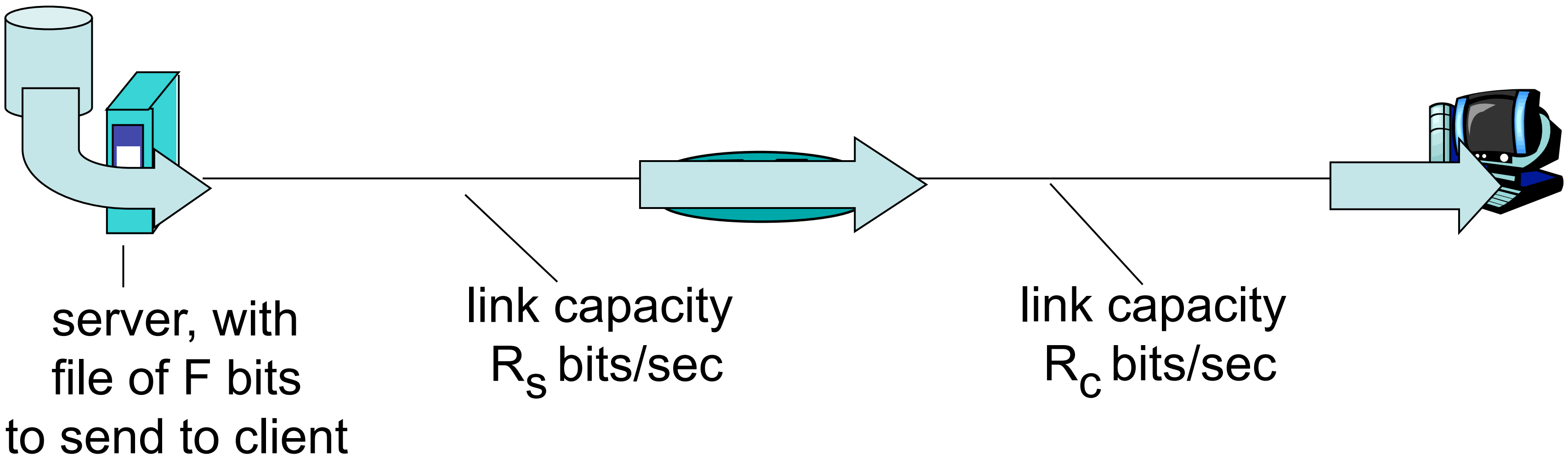
- Propagation speed: $2x10^8$ m/sec
- Transmission of 625 byte (= 5000 bit): t= L/R=5000 / 1Gbit/s = 5 us

Distance	Propagation Delay	equivalent Transmission Delay (625 byte)	CPU cycles per packet (1 GHz)	CPU cycles per byte (1 GHz)
100 m	500 ns	10 Gbit/s	500	<1
1 km	5 us	1 Gbit/s	5.000	8
10 km	50 us	100 Mbit/s	50.000	80
100 km	500 us	10 Mbit/s		800
1.000 km	5 ms	1 Mbit/s		8.000
10.000 km	50 ms	100 Kbit/s		80.000

- Suggestion for homework exercise: plot graphs

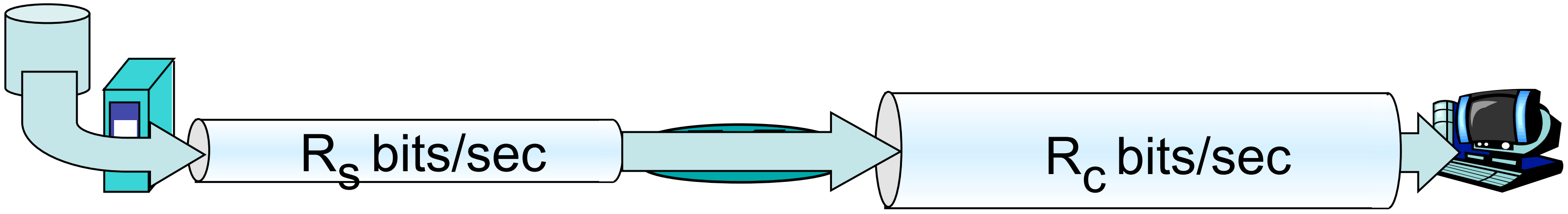
Throughput

- *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

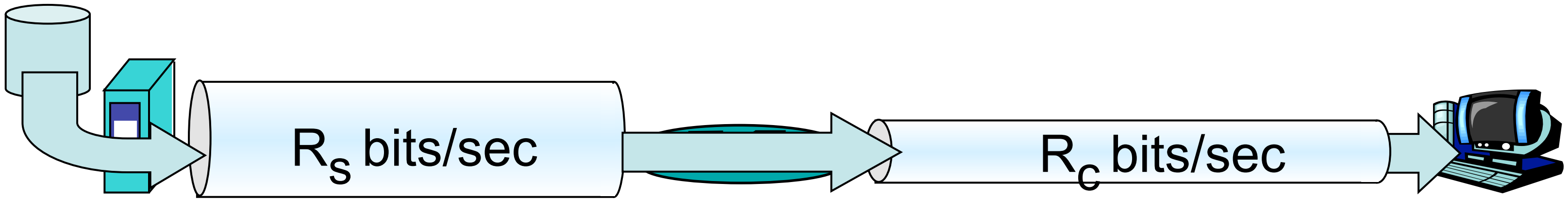


Throughput (more)

- $R_s < R_c$



- $R_s > R_c$

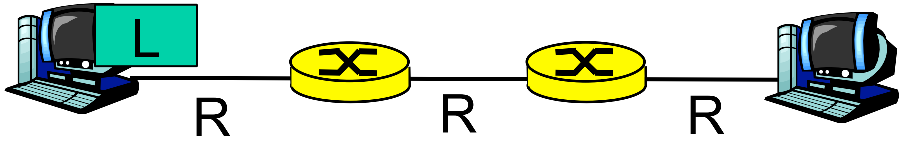


bottleneck link

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

Store-and-Forward vs. Circuit Switching



- 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: Large Message L

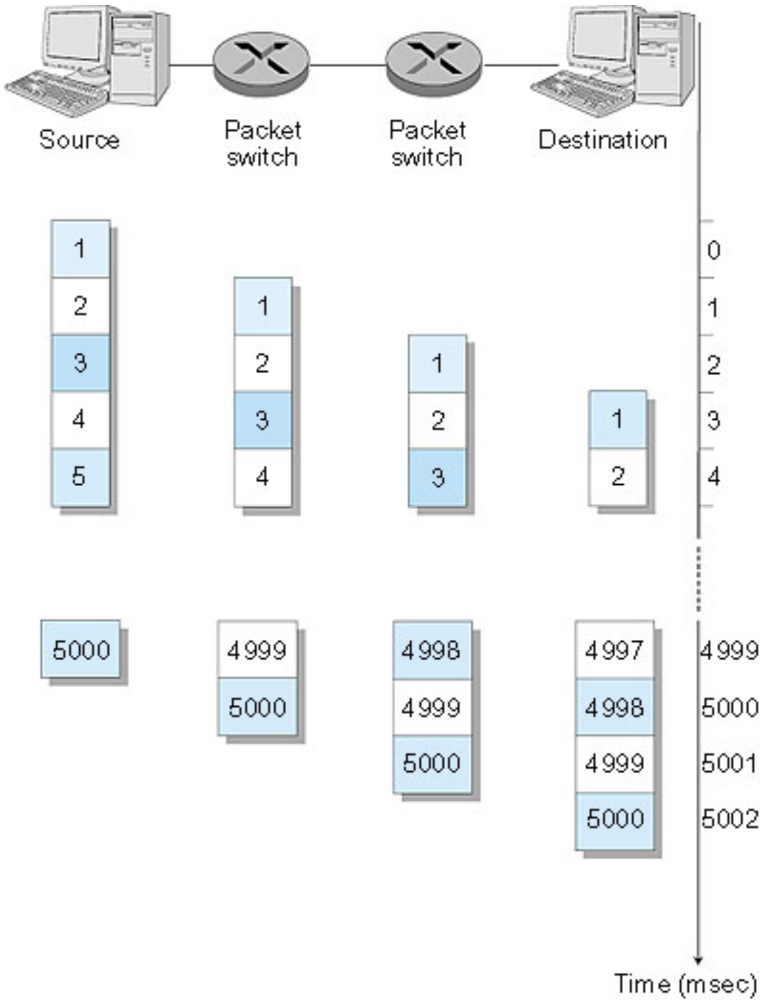
Circuit Switching:

- L = 7.5 Mbit
- R = 1.5 Mbit/s
- Transmission delay = 5 s

Store-and-Forward:

- L = 7.5 Mbit
- R = 1.5 Mbit/s
- Transmission delay = 15 s

Packet Switching: Message Segmenting



Now break up the message into 5000 packets

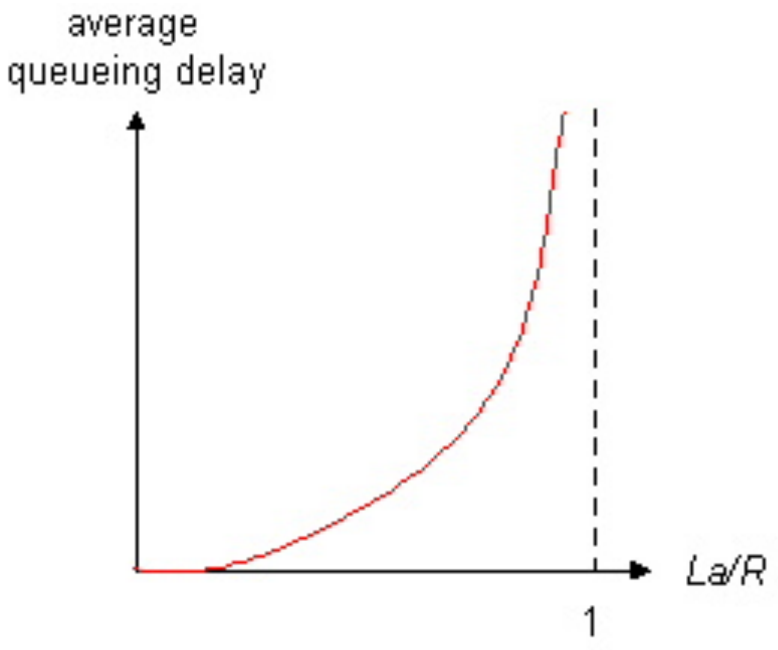
- ❑ Each packet 1,500 bits
- ❑ 1 msec to transmit packet on one link
- ❑ *pipelining:* each link works in parallel
- ❑ Delay reduced from 15 sec to 5.002 sec (as good as circuit switched)
- ❑ Advantages over circuit switching?
- ❑ Drawbacks (of packet vs. Message)

Queueing delay (revisited)

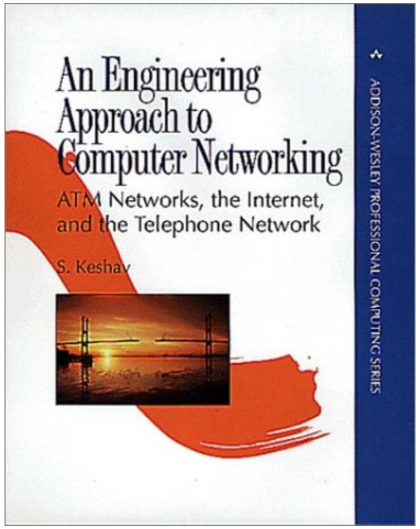
- ❑ R=link bandwidth (bit/s)
- ❑ L=packet length (bit)
- ❑ a=average packet arrival rate

traffic intensity = a· L/R

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

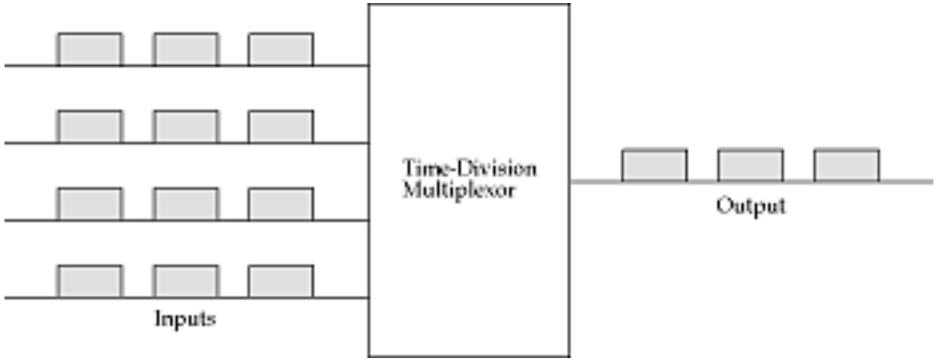


- S. Keshav: *An Engineering Approach to Computer Networking*. Addison-Wesley, 1997


An Engineering Approach to Computer Networking
ATM Networks, the Internet, and the Telephone Network
S. Keshav
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

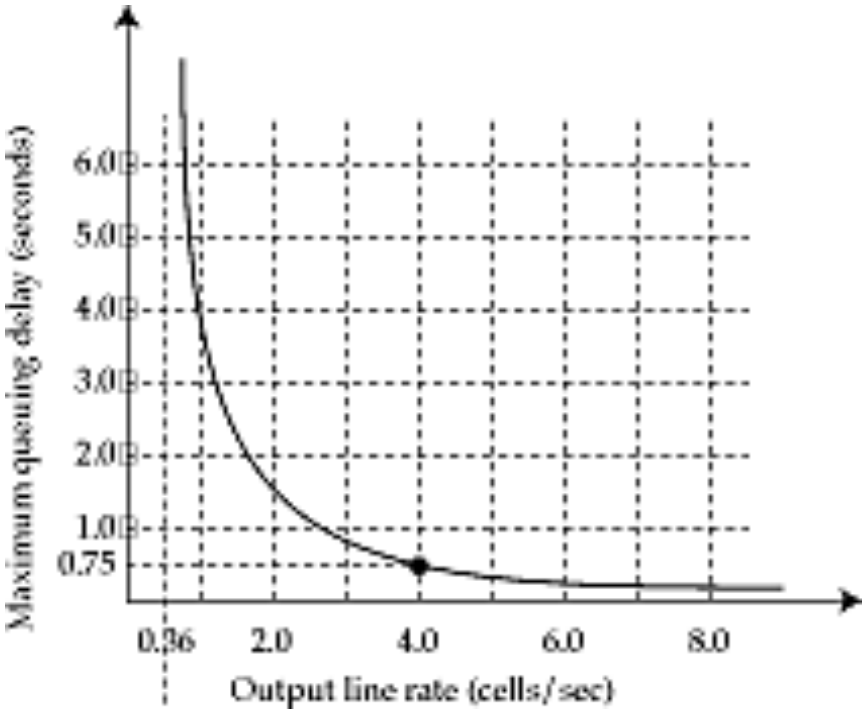
- Srinivasan Keshav - University of Waterloo

Statistical multiplexing



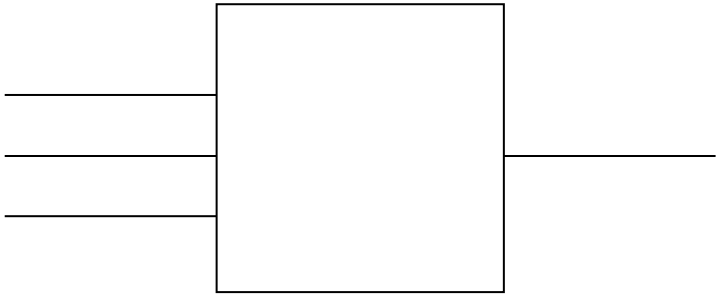
- Suppose packets/cells arrive in bursts
    - each burst has 10 packets/cells evenly spaced 1 second apart
    - gap between bursts = 100 seconds
- What should be service rate of output line?

Statistical Multiplexing



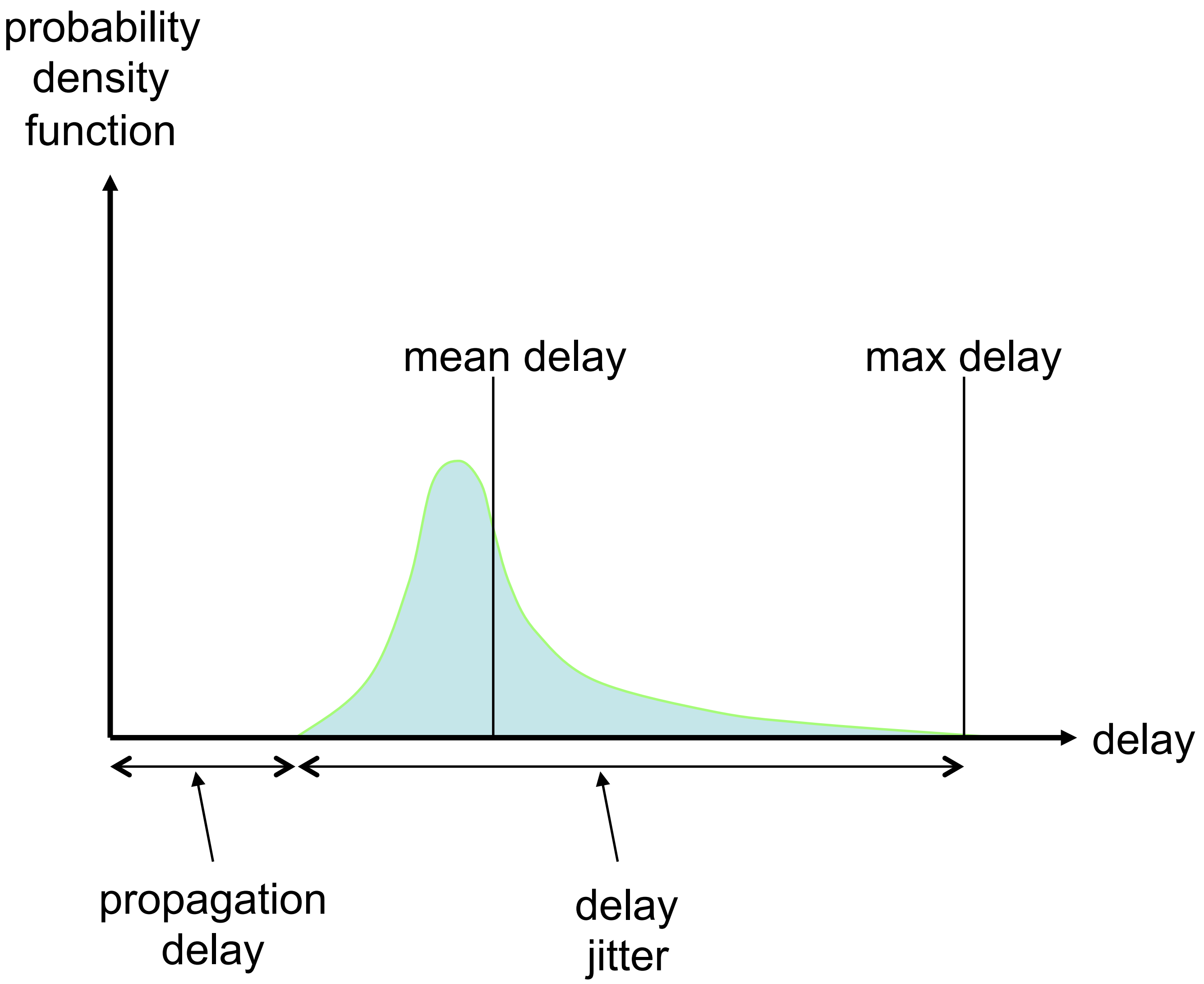
- We can trade off worst-case delay against speed of output trunk
- Statistical Multiplexing Gain
  = (sum of peak input rate)/(output rate)
- Whenever long term average rate differs from peak, we can trade off service rate for delay

Statistical Multiplexing



- ❑ Packets with L=625 byte; R=100 Mbit/s ⇒ d_{trans} = L/R = 50 us
- ❑ Input link: average load = 10%, i.e. a_{in} = 0.1
- ❑ Output link 200 Mbit/s: average load out = 15%, i.e. a_{out} = 0.15
  ⇒ d_{queue_max} = 2x 25 us = 50 us
- ❑ Output link 100 Mbit/s: average load = 30%, i.e. a_{out} = 0.3
  ⇒ d_{queue_max} = 2x 50 us = 100 us
- ❑ Output link 50 Mbit/s: average load = 60%, i.e. a_{out} = 0.6
  ⇒ d_{queue_max} = 2x 100 us = 200 us

Delay Distributions



Discussion

- ❑ Can you „imagine“ a visualisation of packets being transmitted over different types of links?

- ❑ What is the role of statistical multiplexing

- ❑ What are the benefits of overprovisioning?

- ❑ What is the cost of tunneling?

- ❑ What is the role of header lengths?

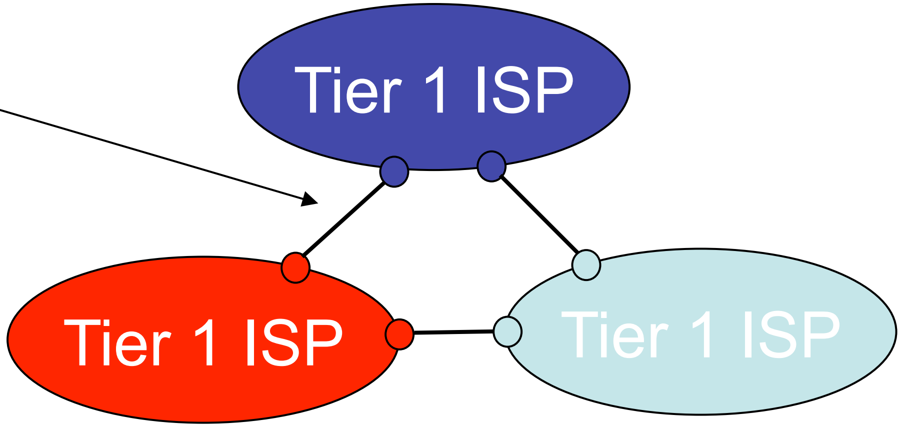
- ❑ What is the role of compact headers / header compression?

Internet Structure

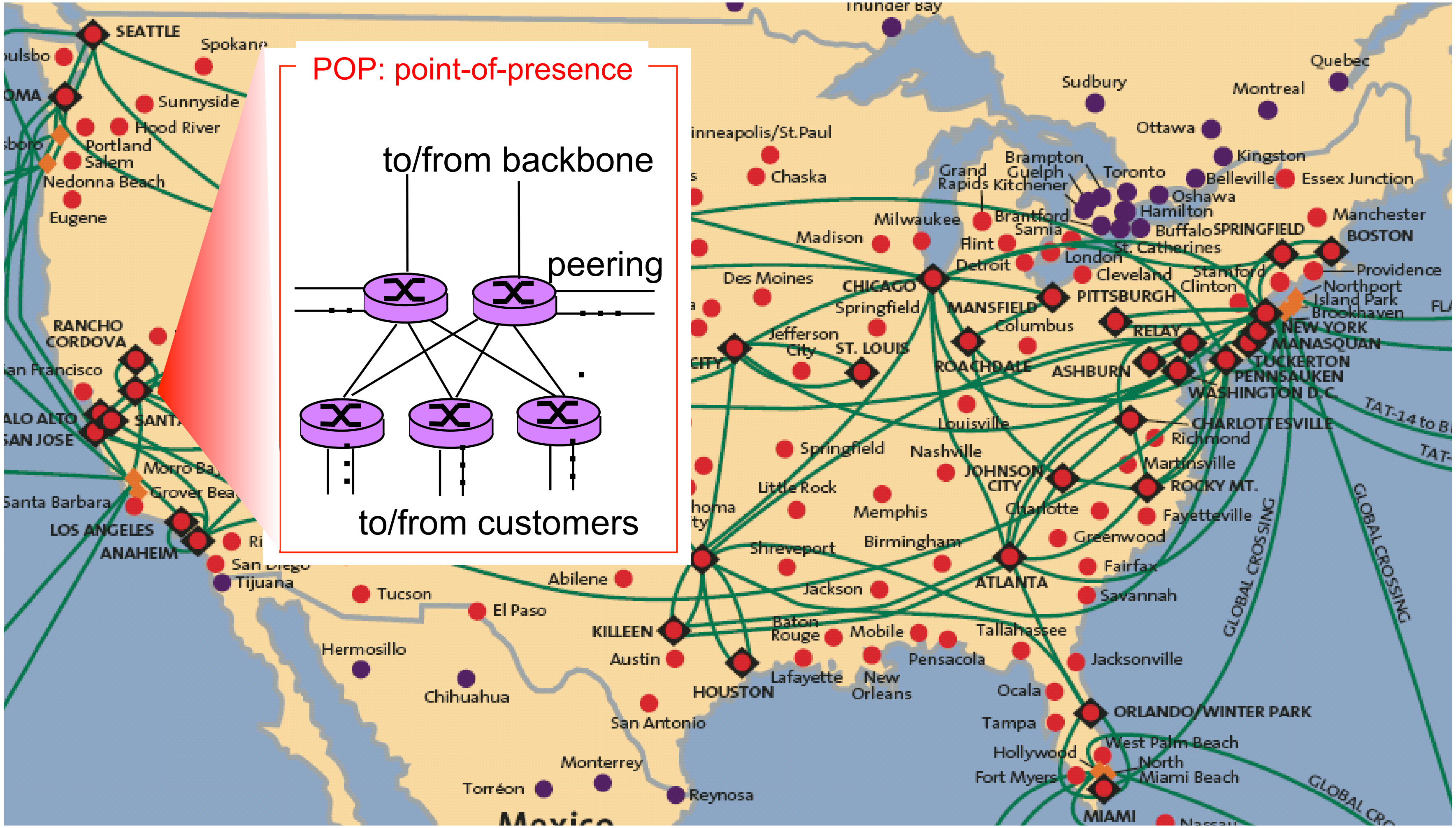
Internet structure: network of networks

- 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



POP: point-of-presence

to/from backbone

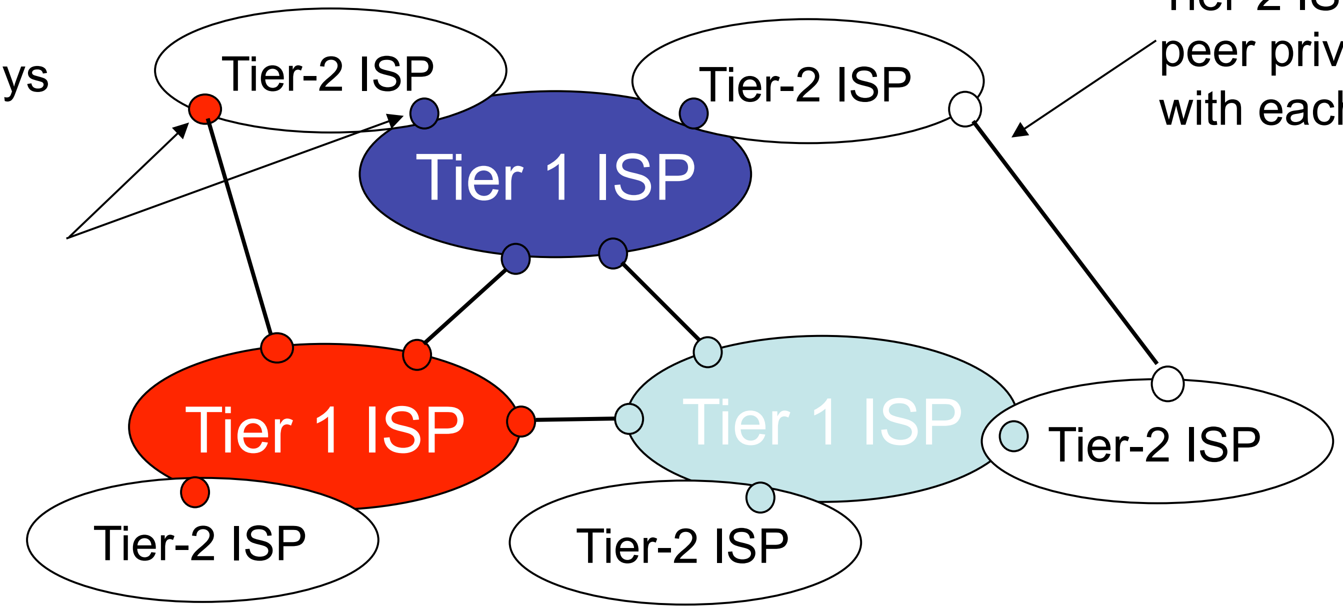
peering

to/from customers

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

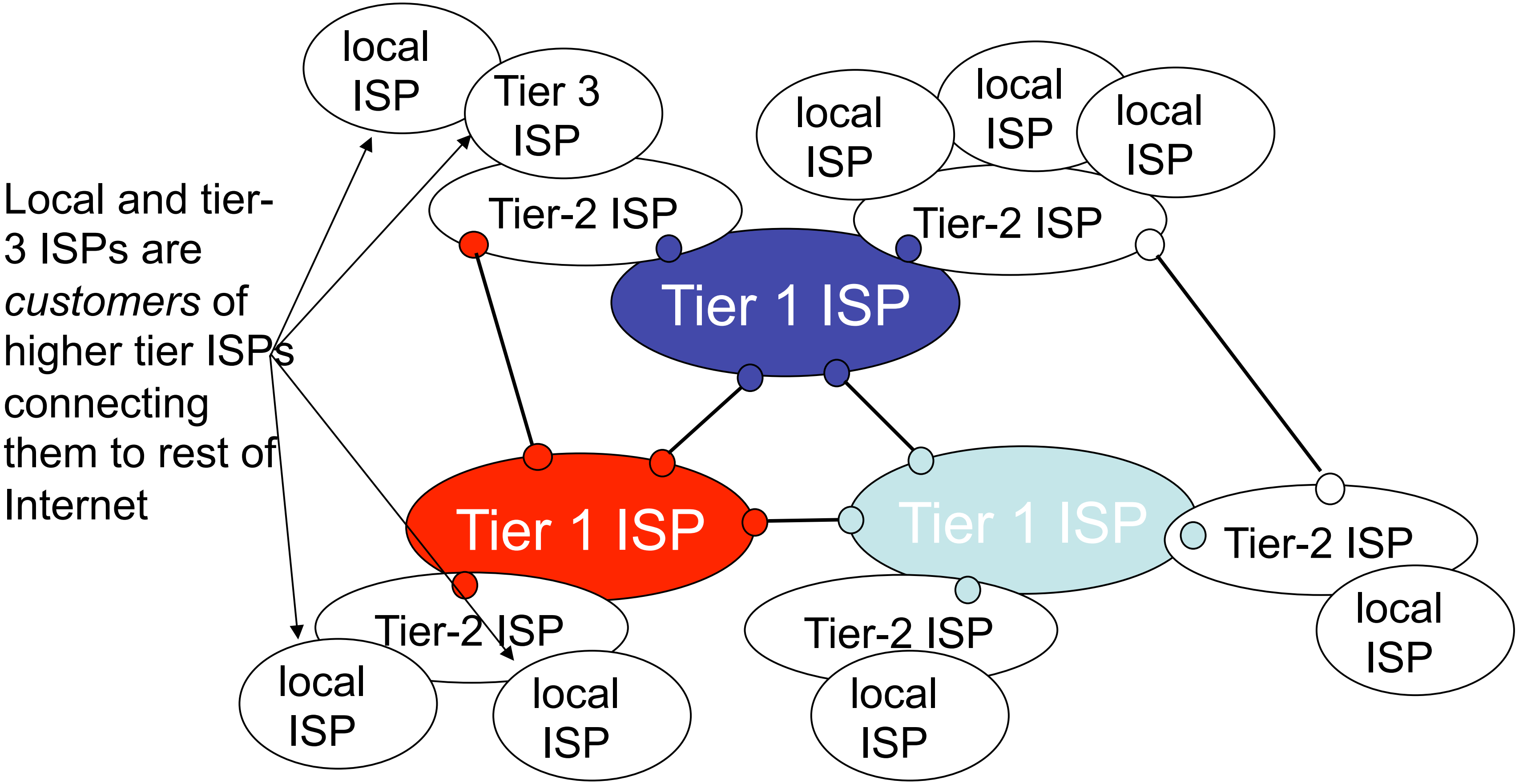
❑ Tier-2 ISP pays tier-1 ISP for connectivity to rest of Internet
❑ tier-2 ISP is *customer* of tier-1 provider



Tier-2 ISPs also peer privately with each other.

Internet structure: network of networks

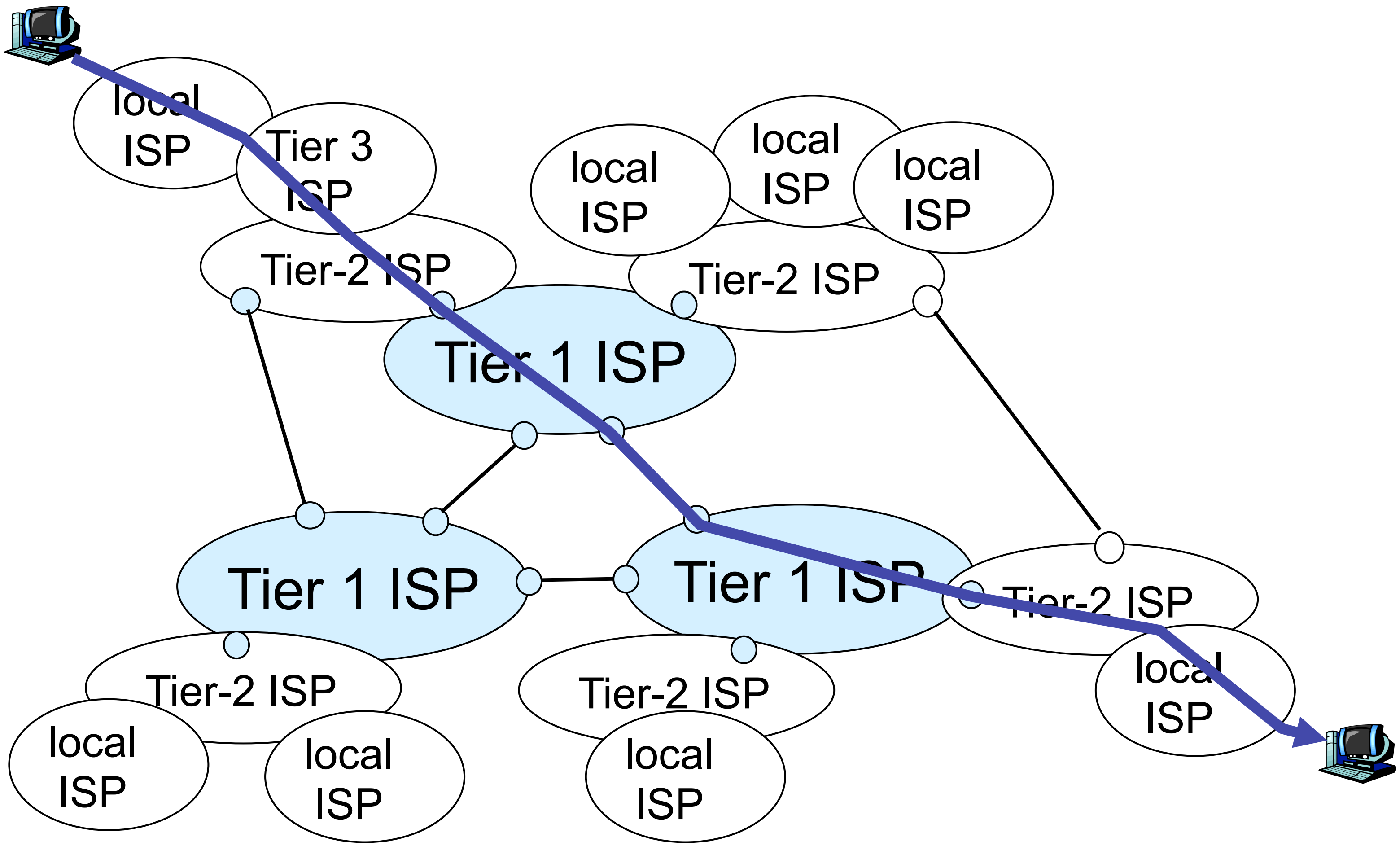
- "Tier-3" ISPs and local ISPs
  - last hop ("access") network (closest to end systems)



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

Internet structure: network of networks

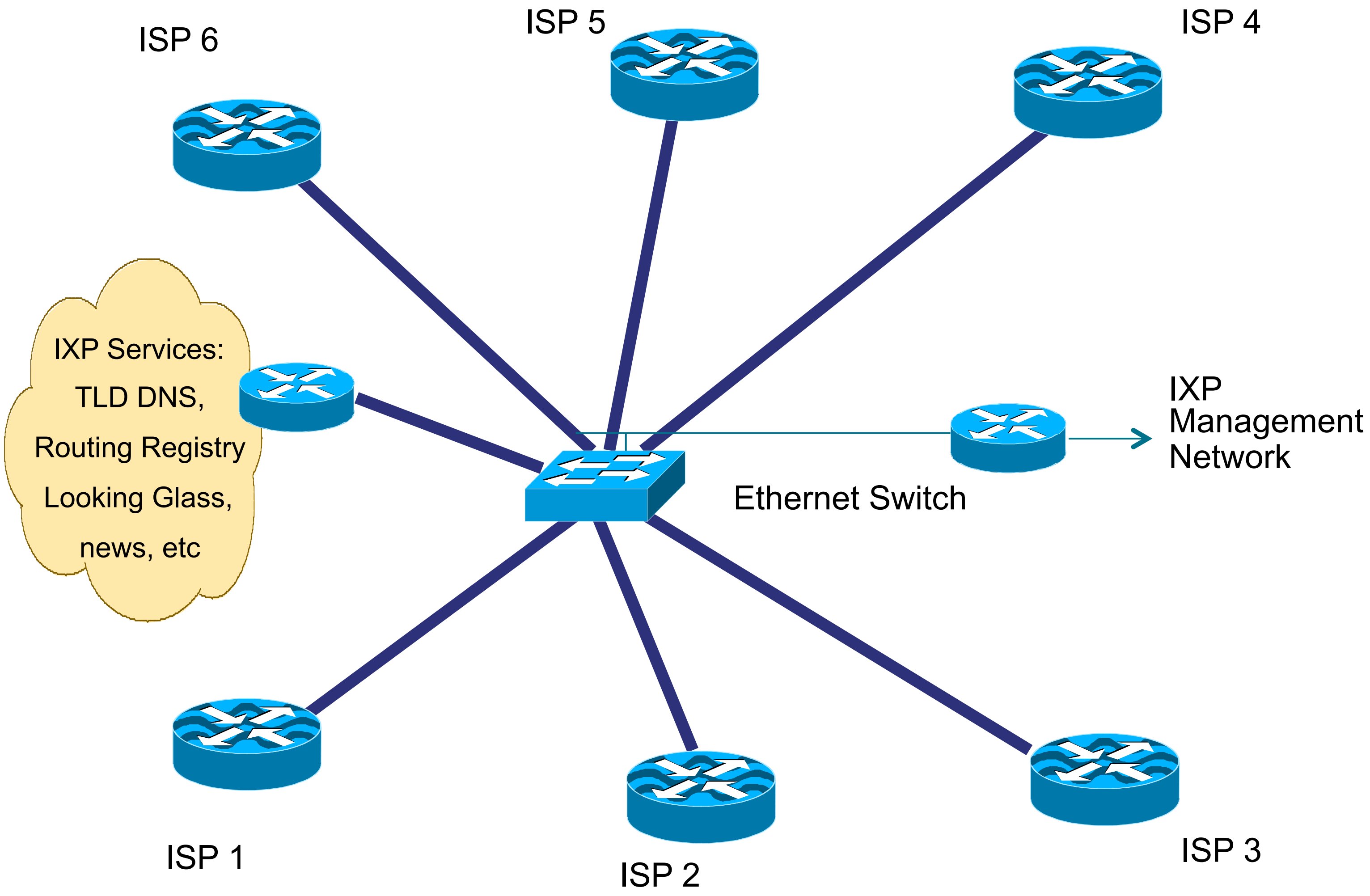
- a packet passes through many networks!



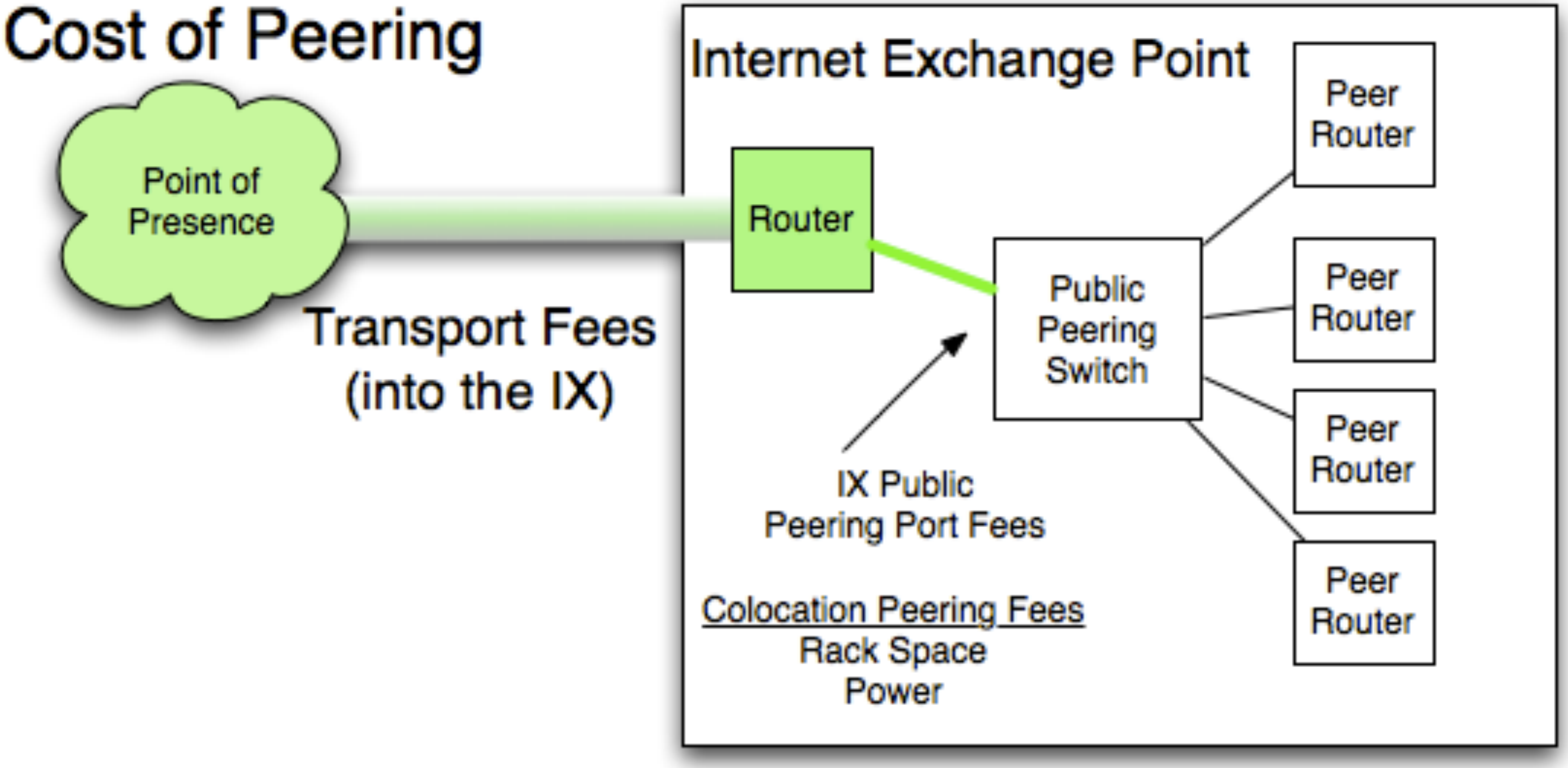
Internet Ecosystem

- >30,000 autonomous networks
- Networks with different
  - different roles and business type
    - stub networks
    - transit networks
    - content providers
  - Influenced by traffic patterns, application popularity, economics, regulation, ….
- Peering
  - bilateral contracts
  - Customer-provider, settlement-free peering, or in between
- Internet Exchange Points

Internet Exchange Point

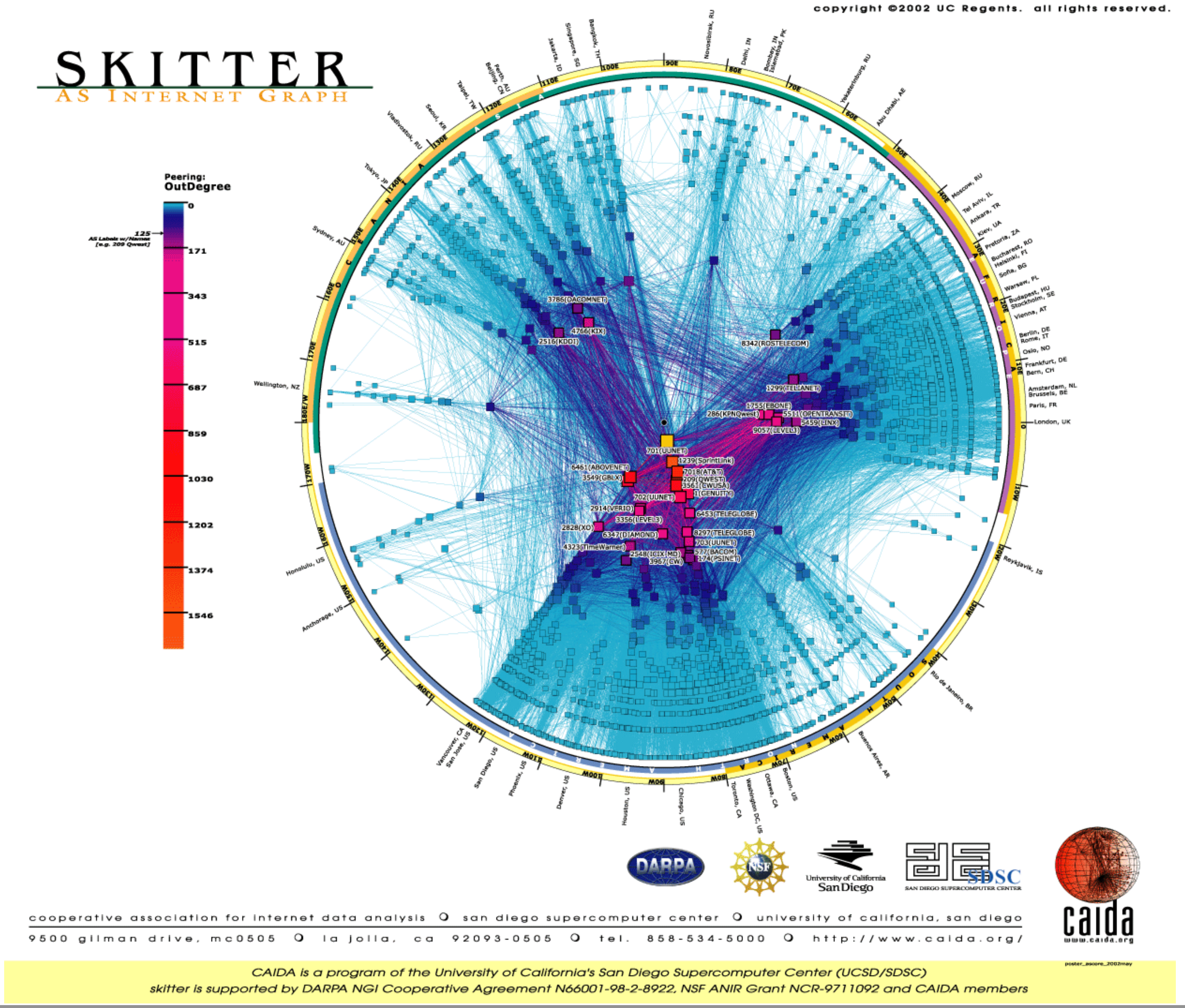


Cost of Peering at Internet Exchange Point



source: William B. Norton, „Internet Peering“, http://drpeering.net/

ISP Peering Relations



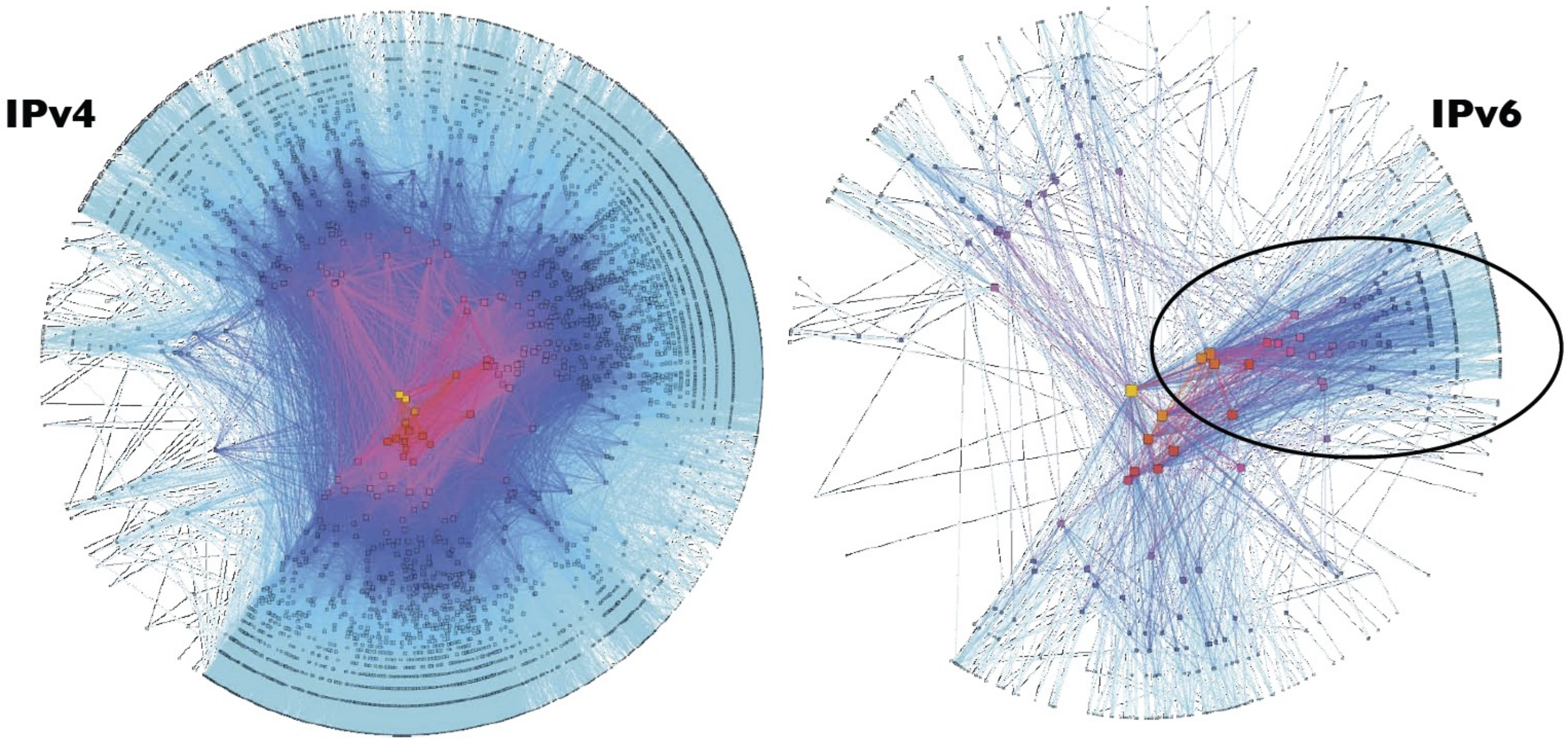
SKITTER
AS INTERNET GRAPH

cooperative association for internet data analysis ○ san diego supercomputer center ○ university of california, san diego

9500 gilman drive, mc0505 ○ la jolla, ca 92093-0505 ○ tel. 858-534-5000 ○ http://www.caida.org/

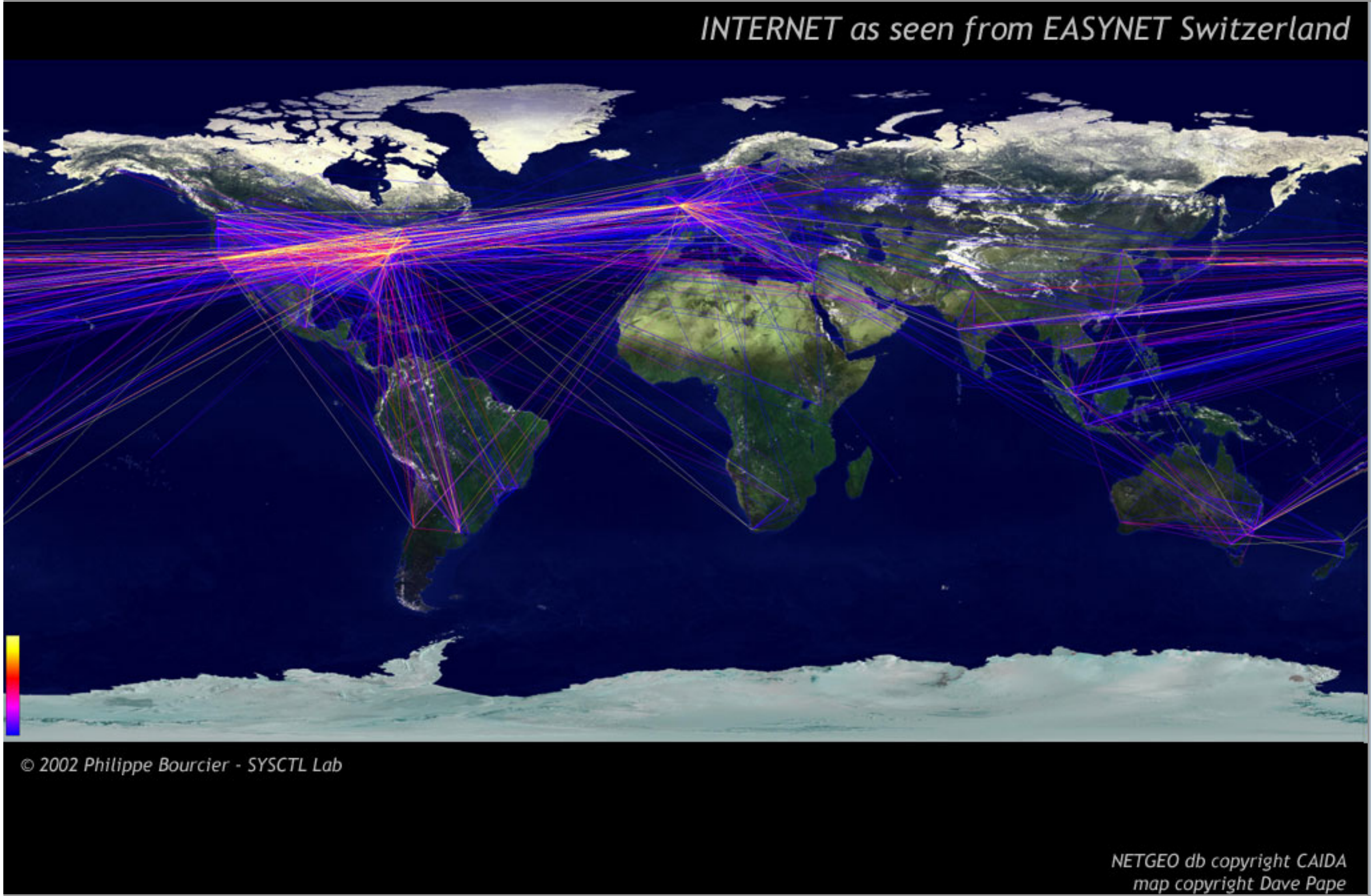
CAIDA is a program of the University of California's San Diego Supercomputer Center (UCSD/SDSC)
skitter is supported by DARPA NGI Cooperative Agreement N66001-98-2-8922, NSF ANIR Grant NCR-9711092 and CAIDA members

IPv4 vs. IPv6 Graphs



source: caida.org

AS Connectivity



INTERNET as seen from EASYNET Switzerland
© 2002 Philippe Bourcier - SYSCTL Lab
NETGEO db copyright CAIDA
map copyright Dave Pape

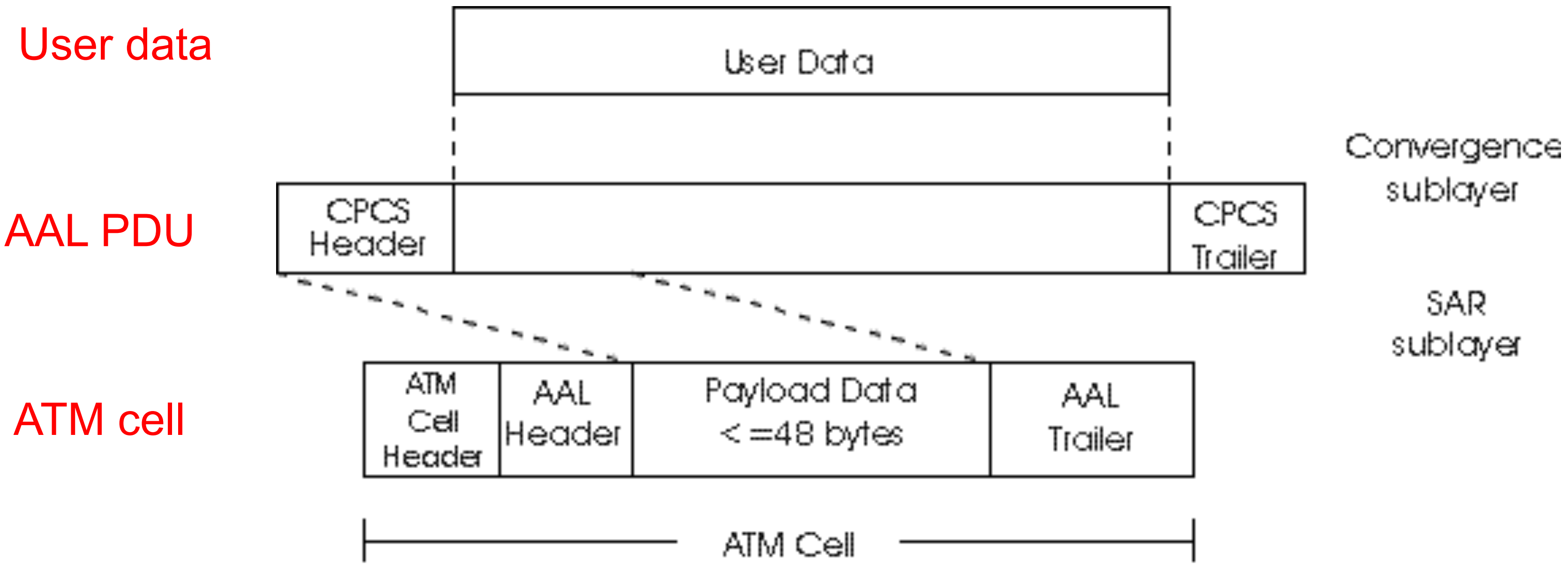
Network Architectures

Link virtualization: ATM, MPLS

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 (e.g., IP datagrams)



ATM Layer

Service: transport cells across ATM network

- analogous to IP network layer
- very different services than IP network layer
- possible Quality of Service (QoS) Guarantees

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

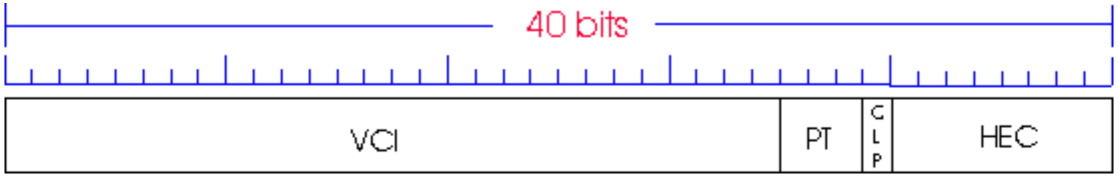
ATM VCs

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

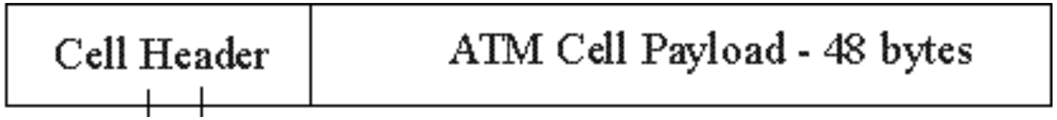
ATM Layer: ATM cell

- 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

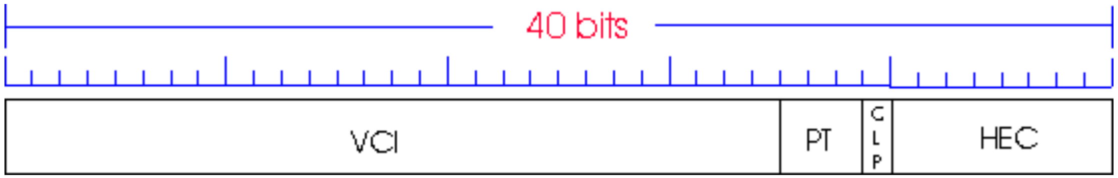


Cell format

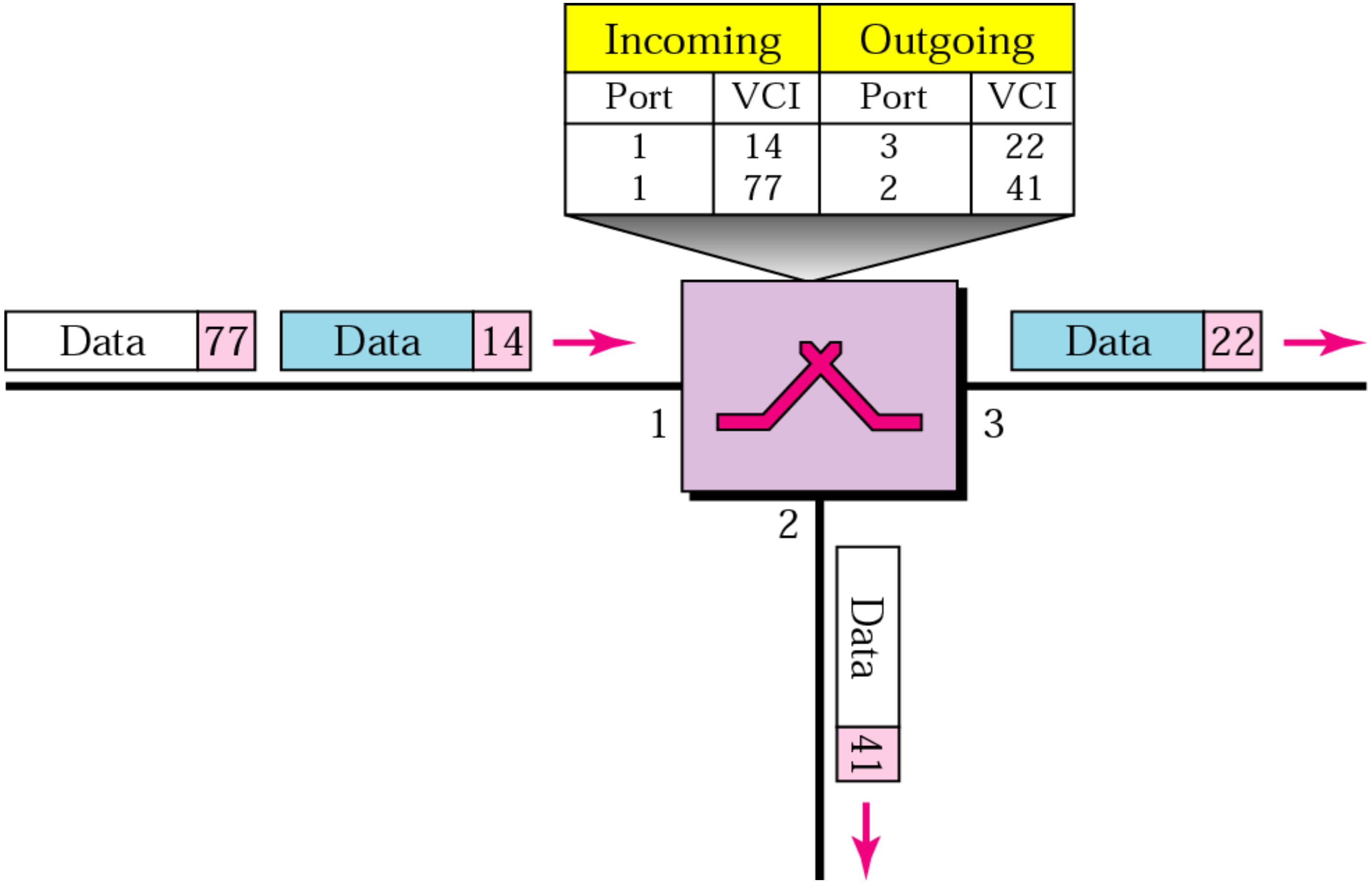


ATM cell header

- **VCI:** virtual channel ID
  - may *change* from link to link through network
- **PT:** Payload type: RM (resource management) vs. 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



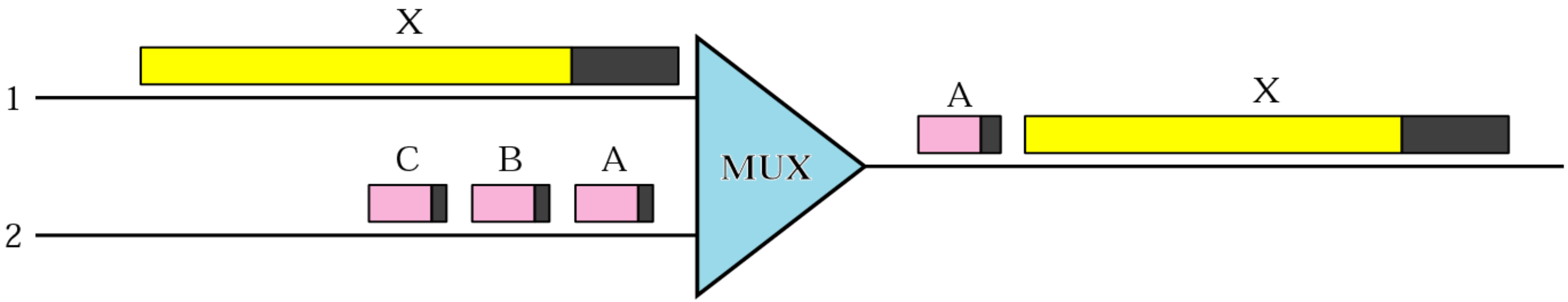
Virtual Circuit Switching



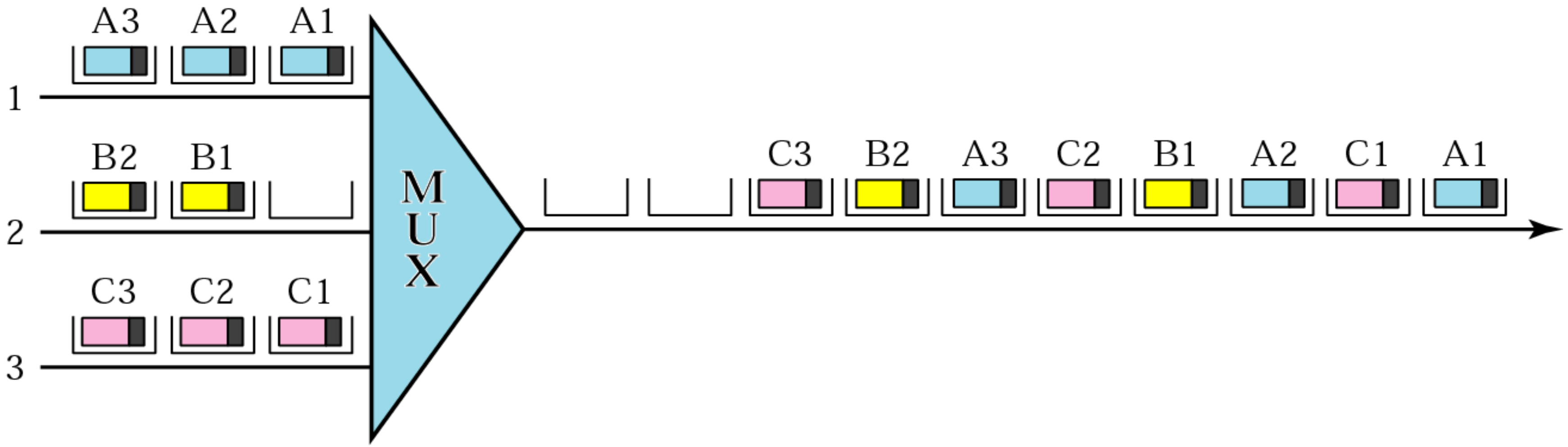
Incoming		Outgoing	
Port	VCI	Port	VCI
1	14	3	22
1	77	2	41

Multiplexing of Variable vs. Fixed Size Packets

- Multiplexing of variable size packets

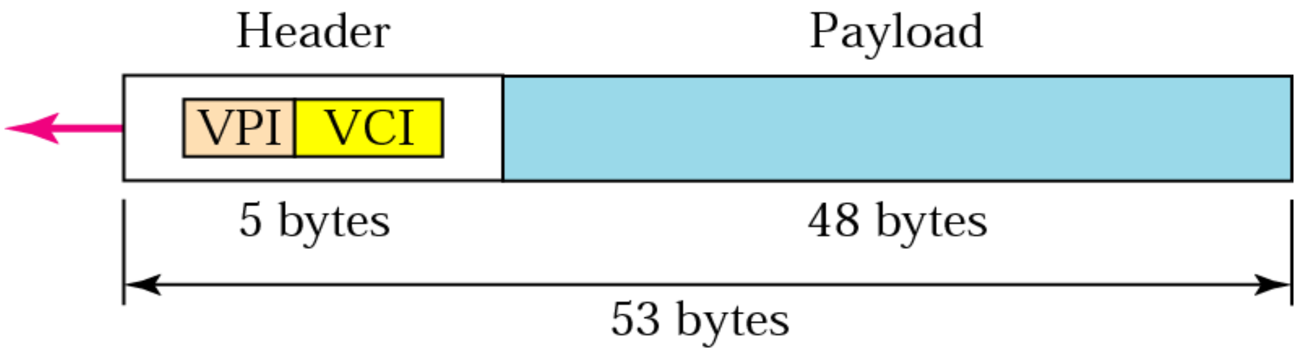


- ATM Multiplexing

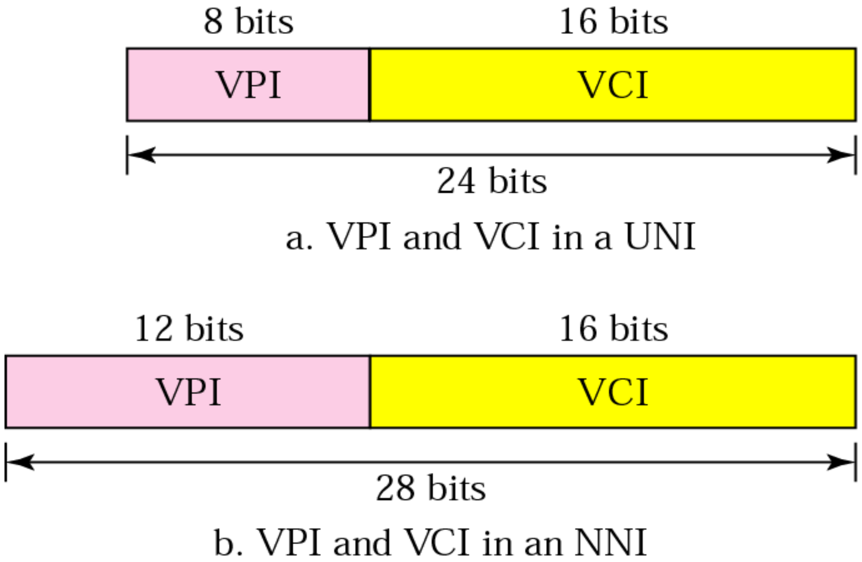


ATM Identifiers

- ATM Cell



- Virtual Path Identifiers and Virtual Channel Identifiers

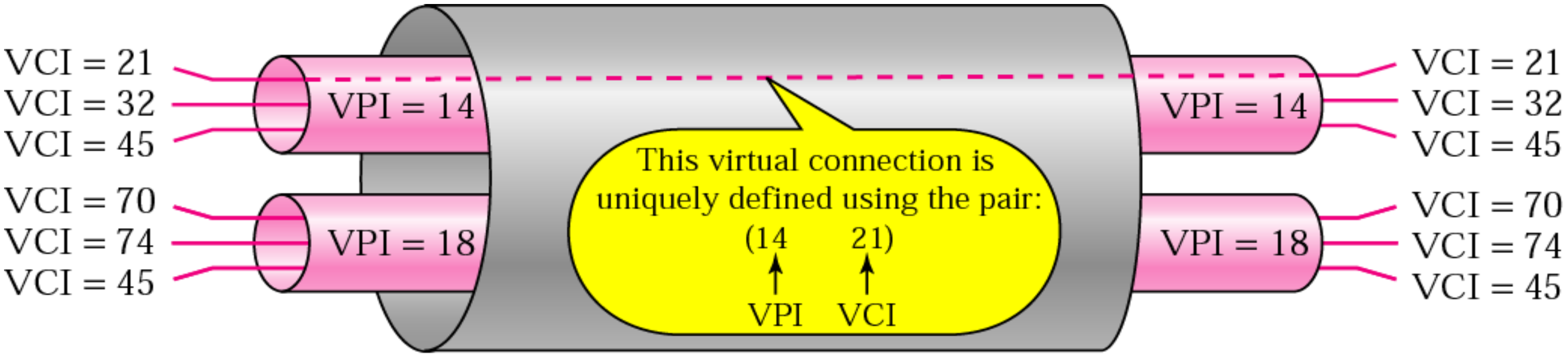


a. VPI and VCI in a UNI

b. VPI and VCI in an NNI

(UNI: User-to-Network-Interface
NNI: Network-to-Network-Interface)

ATM Virtual Connections



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)
  - transmission of **idle cells** when no data cells to send

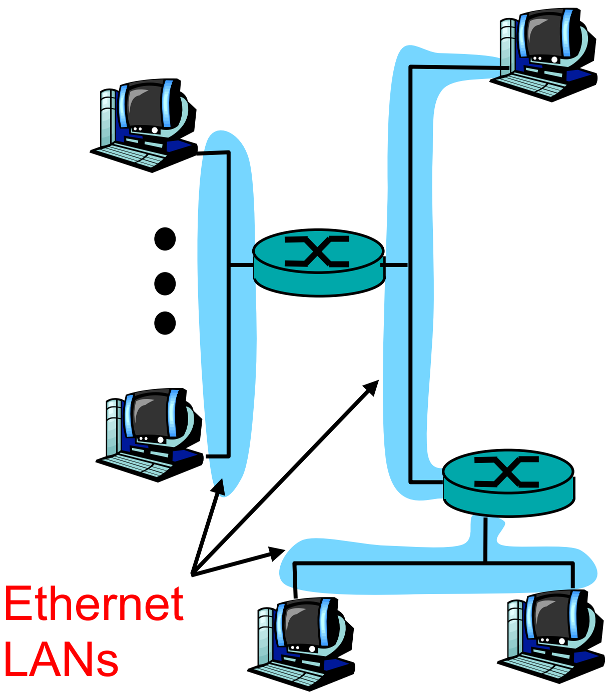
IP-Over-ATM

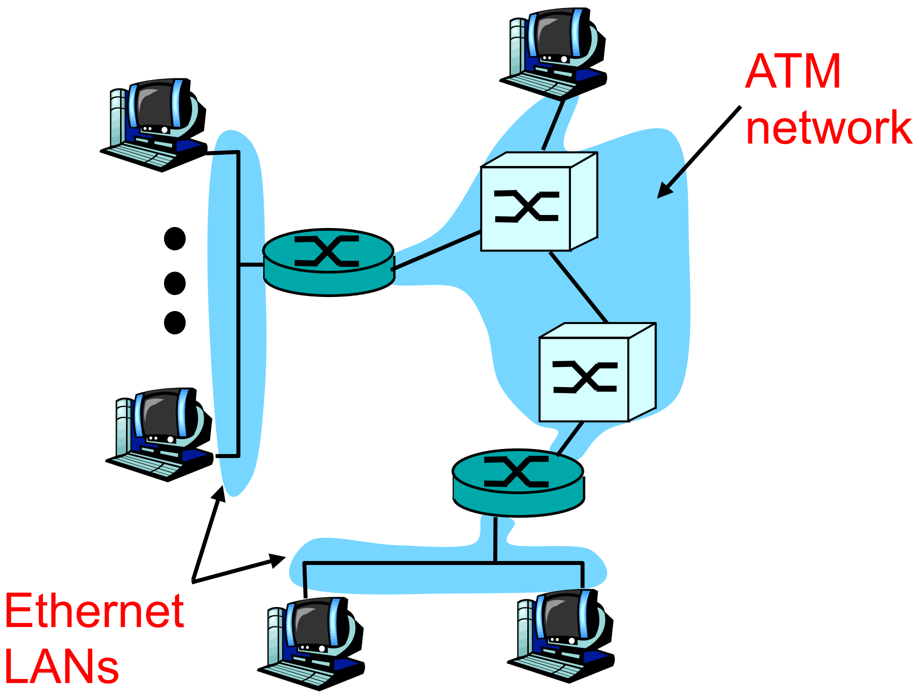
Classic IP only

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

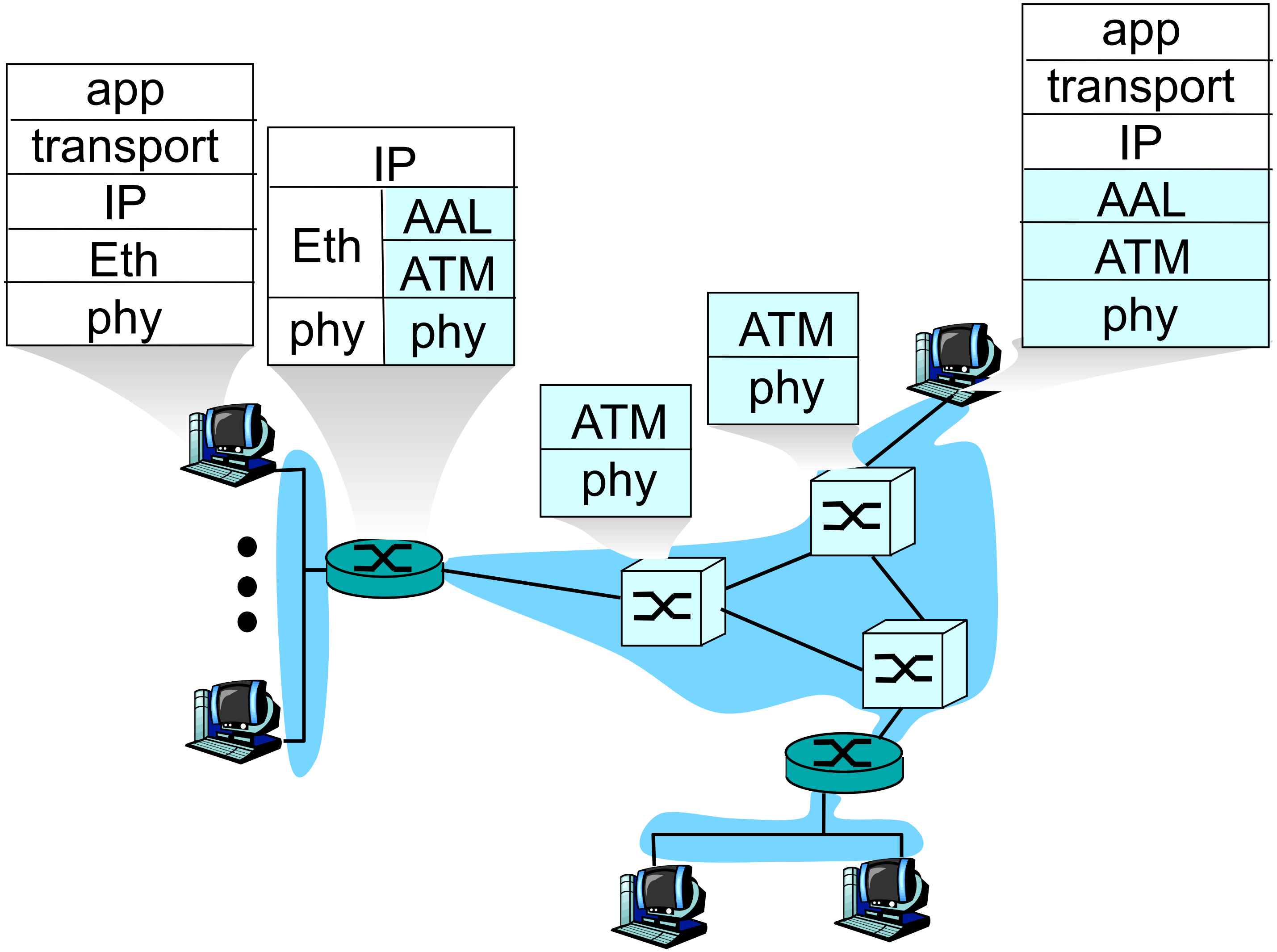
IP over ATM

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





IP-Over-ATM


app
transport
IP
Eth
phy
IP
Eth
AAL
ATM
phy
phy
ATM
phy
ATM
phy
app
transport
IP
AAL
ATM
phy

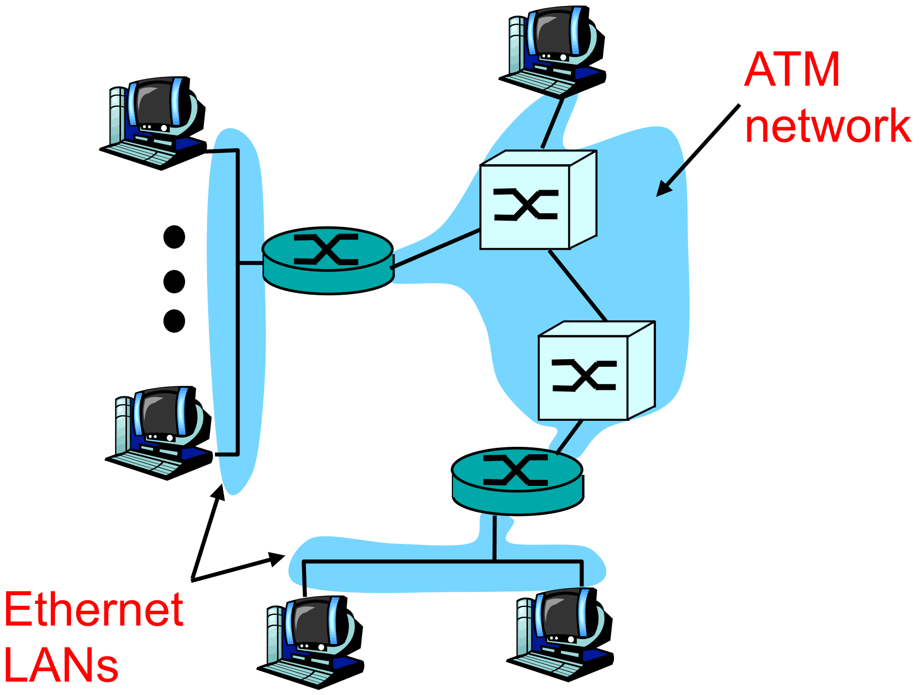
Datagram Journey in IP-over-ATM Network

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

IP-Over-ATM

Issues:

- ❑ IP datagrams into ATM AAL5 PDUs
- ❑ from IP addresses to ATM addresses
  - ▪ just like IP addresses to 802.3 MAC addresses!
  - ▪ ARP server



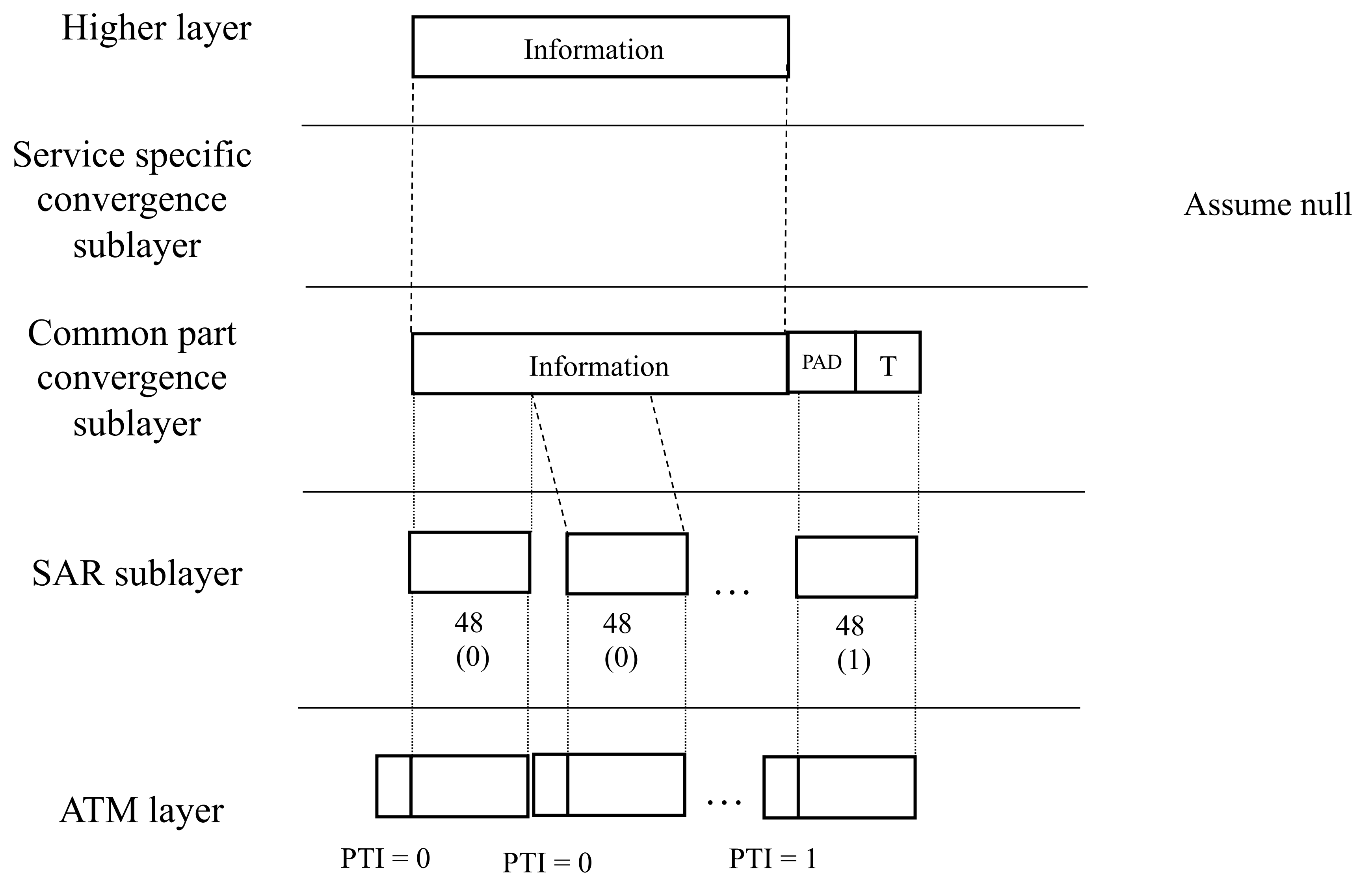
AAL 5 Protocol

- AAL5 is a simple and efficient AAL (SEAL) to perform a subset of the functions of AAL3/4
- The CPCS-PDU payload length can be up to 65,535 octets and must use PAD (0 to 47 octets) to align CPCS-PDU length to a multiple of 48 octets

PAD	Padding
CPCS-UU	CPCS User-to-User Indicator
CPI	Common Part Indicator
Length	CPCS-PDU Payload Length
CRC-32	Cyclic Redundancy Chuck

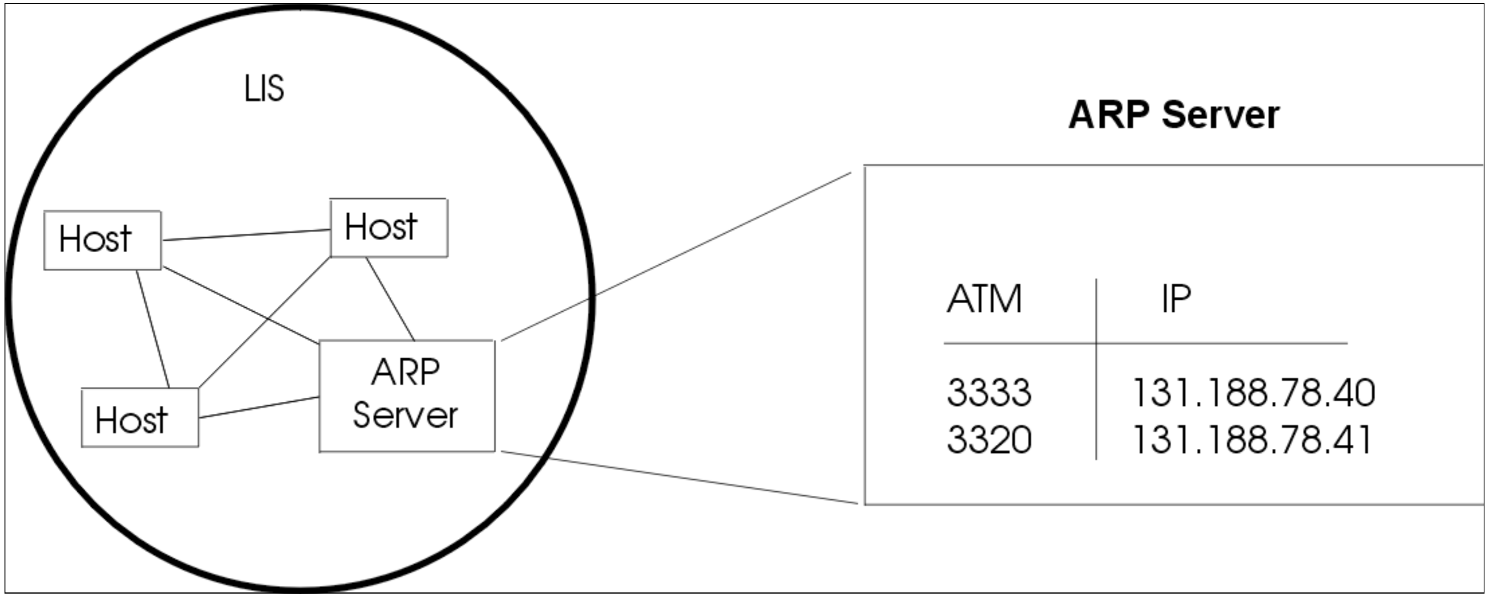
	0 - 47	1	1	2	4
CPCS-PDU Payload	PAD	CPCS UU	CPI	Length	CRC-32

AAL 5 Layering



Classical IP and ARP over ATM (CLIP)

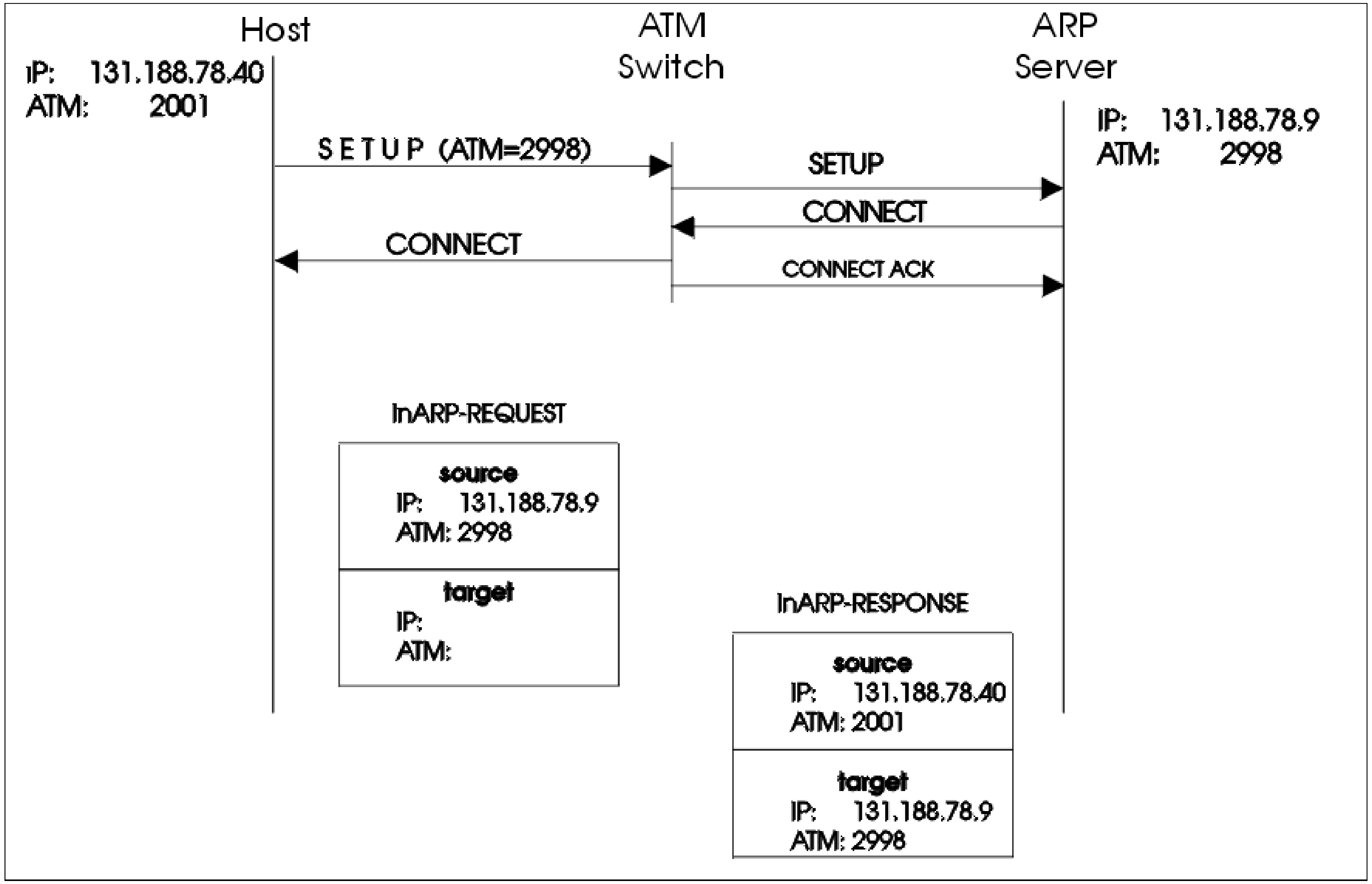
- Specification of a complete IP implementation for ATM
- Suitable for ATM unicast communication
- Encapsulation of IP packets into AAL PDUs
- Support for large MTU sizes
- There must be an ATMARP server in each LIS (Logical IP Subnet)



ATM	IP
3333	131.188.78.40
3320	131.188.78.41

Classical IP and ARP over ATM (CLIP)

- The host registers its IP/ATM address information at the ATMARP server using the InARP protocol



Classical IP and ARP over ATM (CLIP)

- RFC 1577: Classical IP and ARP over ATM
- ATMARP Server Operational Requirements
  - The ATMARP server, upon the completion of an ATM call/ connection of a new VC, will transmit an InATMARP request to determine the IP address of the client.
  - The InATMARP reply from the client contains the information necessary for the ATMARP Server to build its ATMARP table cache.
  - This information is used to generate replies to the ATMARP requests it receives.
- InATMARP is the same protocol as the original InARP protocol presented in RFC 1293 but applied to ATM networks: Discover the protocol address of a station associated with a virtual circuit.
- RFC 1293: Bradely, T., and C. Brown, "Inverse Address Resolution Protocol“, January 1992.

Classical IP and ARP over ATM (CLIP)

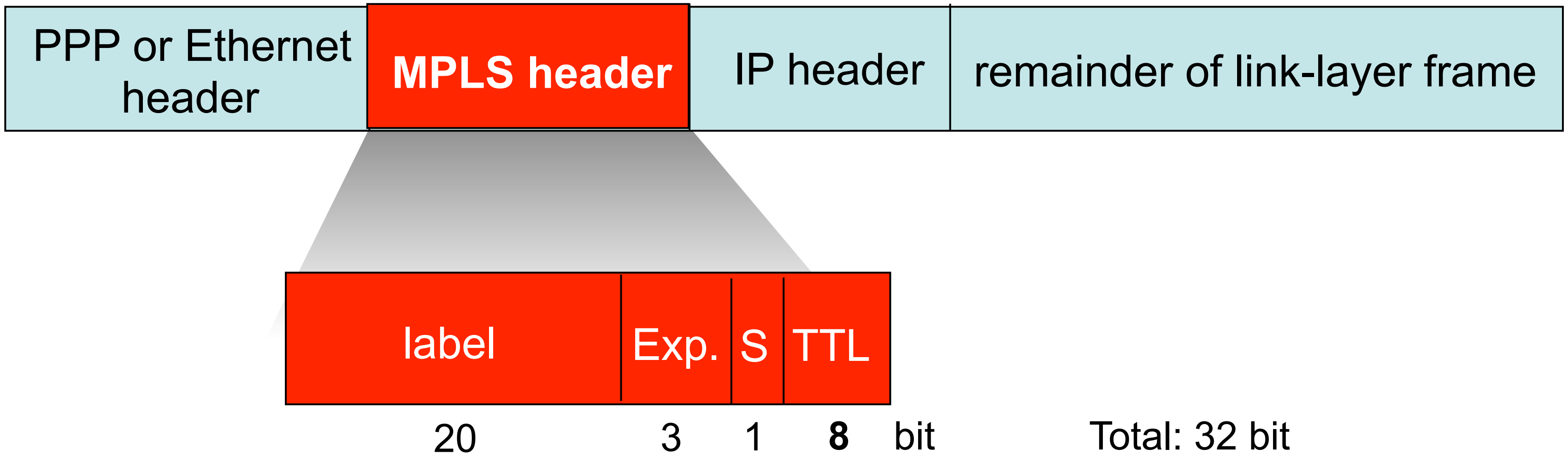
- RFC 1577: Classical IP and ARP over ATM
- ATMARP Client Operational Requirements
  1. Initiate the VC connection to the ATMARP server for transmitting and receiving ATMARP and InATMARP packets.
  2. Respond to ARP_REQUEST and InARP_REQUEST packets received on any VC appropriately.
  3. Generate and transmit ARP_REQUEST packets to the ATMARP server and to process ARP_REPLY appropriately. ARP_REPLY packets should be used to build/refresh its own client ATMARP table entries.
  4. Generate and transmit InARP_REQUEST packets as needed and to process InARP_REPLY packets appropriately. InARP_REPLY packets should be used to build/refresh its own client ATMARP table entries.
  5. Provide an ATMARP table aging function to remove own old client ATMARP tables entries after a period of time.

MPLS

Multi-Protocol Label Switching

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
  - IP datagram still keeps IP address
  - RFC 3032 defines MPLS header
    - Label: has role of Virtual Circuit Identifier
    - Exp: experimental usage, may specify Class of Service (CoS)
    - S: Bottom of Stack - end of series of stacked headers
    - TTL: time to live



Multiprotocol label switching (MPLS)

- RFC 3270: Le Faucheur, F., Wu, L., Davie, B., Davari, S., Vaananen, P., Krishnan, R., Cheval, P. and J. Heinanen, “Multi-Protocol Label Switching (MPLS) Support of Differentiated Services”, May 2002.
  - EXP: 3 bits - this field contains the value of the EXP field for the EXP<->PHB (Per-Hop-Behaviour) mapping
  - Mapping transported via signaling protocol
- RFC 3140: Black, D., Brim, S., Carpenter, B. and F. Le Faucheur, "Per Hop Behavior Identification Codes", June 2001.
  - Case 1: PHBs defined by standards action, as per [RFC 2474]. PHB is recommended 6-bit DSCP value for that PHB, left-justified in a 16 bit field, with bits 6 through 15 set to zero.
  - Case 2: PHBs not defined by standards action, i.e., experimental or local use PHBs In this case an arbitrary 12 bit PHB-ID is placed left-justified in the a bit field.
    Bit 15 is set to 1, Bits 12 and 13 are zero.

MPLS TTL Processing

c.f. RFC 3032 - MPLS Label Stack Encoding

- Protocol-independent rules
  - "outgoing TTL" of a labeled packet is either
    a) one less than the incoming TTL, or b) zero.
  - Packets with TTL=0 are discarded
- IP-dependent rules
  - When an IP packet is first labeled, the TTL field of the label stack is set to the value of the IP TTL field.
  - If the IP TTL field needs to be decremented, as part of the IP processing, it is assumed that this has already been done.
  - When a label is popped, and the resulting label stack is empty, then the value of the IP TTL field SHOULD BE replaced with the outgoing MPLS TTL value.
  - A network administration may prefer to decrement the IPv4 TTL by one as it traverses an MPLS domain.

ICMP

- When a router receives an IP datagram that it can't forward, it sends an ICMP message to the datagram's originator
- The ICMP message indicates why the datagram couldn't be delivered
  - E.g., Time Expired, Destination Unreachable
- The ICMP message also contains the IP header and at least leading 8 octets of the original datagram
  - RFC 1812 - Requirements for IP Version 4 Routers extends this to "as many bytes as possible"
  - Historically, every ICMP error message has included the Internet header and at least
  - Including only the first 8 data bytes of the datagram that triggered the error is no longer adequate, due to use e.g. of IP-in-IP tunneling

ICMP in presence of MPLS

- When an LSR receives an MPLS encapsulated datagram that it can’t deliver
  - It removes entire MPLS labels stack
  - It sends an ICMP message to datagram’s originator
- The ICMP message indicates why the datagram couldn’t be delivered (e.g., time expired, destination unreachable)
- The ICMP message also contains the IP header and leading 8 octets of the original datagram
  - RFC 1812 extends this to “as many bytes as possible”

ICMP in Presence of MPLS

Issue

- The ICMP message contains no information regarding the MPLS stack that encapsulated the datagram when it arrived at the LSR
- This is a significant omission because:
  - The LSR tried to forward the datagram based upon that label stack
  - Resulting ICMP message may be confusing

Why?

ICMP in Presence of MPLS

Issue

- ❑ ICMP Destination Unreachable
  - Message contains IP header of original datagram
  - Router sending ICMP message has an IP route to the original datagram's destination
  - Original datagram couldn't be delivered because MPLS forwarding path was broken
- ❑ ICMP Time Expired
  - Message contains IP header of original datagram
  - TTL value in IP header is greater than 1
  - TTL expired on MPLS header. ICMP Message contains IP header of original datagram

ICMP with MPLS

c.f. RFC 4950 - ICMP Extensions for Multiprotocol Label Switching

- defines an ICMP extension object that permits an LSR to append MPLS information to ICMP messages.
- ICMP messages include the MPLS label stack, as it arrived at the router that is sending the ICMP message.
- equally applicable to ICMPv4 [RFC792] and ICMPv6 [RFC4443]
- sample output from an enhanced TRACEROUTE:

```
> traceroute 192.0.2.1
traceroute to 192.0.2.1 (192.0.2.1), 30 hops max, 40 byte packets
1 192.0.2.13 (192.0.2.13) 0.661 ms 0.618 ms 0.579 ms
2 192.0.2.9 (192.0.2.9) 0.861 ms 0.718 ms 0.679 ms
   MPLS Label=100048 Exp=0 TTL=1 S=1
3 192.0.2.5 (192.0.2.5) 0.822 ms 0.731 ms 0.708 ms
   MPLS Label=100016 Exp=0 TTL=1 S=1
4 192.0.2.1 (192.0.2.1) 0.961 ms 8.676 ms 0.875 ms
```

- MPLS Label Stack Object: can be appended to ICMP Time Exceeded and Destination Unreachable messages.

```
      0             1              2            3
+-------------+-------------+-------------+-------------+
|                 Label              |EXP |S|    TTL    |
+-------------+-------------+-------------+-------------+
|                                                       |
|       // Remaining MPLS Label Stack Entries //        |
|                                                       |
+-------------+-------------+-------------+-------------+
```

- Must be preceded by an ICMP Extension Structure Header and an ICMP Object Header, defined in [RFC4884].

Multi-Part ICMP Messages - RFC 4884

- ICMP Extension Structure may be appended to ICMP v4 / v6 Destination Unreachable and Time Exceeded messages
- ICMP Extension Structure Header

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|      (Reserved)       |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

ICMP extension version number: 2

- ICMP Object Header and Object Payload

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Length            |   Class-Num   |   C-Type      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   // (Object Payload) //                      |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

# MPLS for Linux

# The work of James Leu:

https://sourceforge.net/projects/mpls-linux/

Discussions:

http://sourceforge.net/mailarchive/forum.php?forum_name=mpls-linux-devel

# Bug fixes of Jorge Boncompte:

http://mpls-linux.git.sourceforge.net/git/gitweb.cgi?p=mpls-linux/net-next;a=shortlog;h=refs/heads/net-next-mpls

# Additional bug fixes by Igor Maravić:

https://github.com/i-maravic/MPLS-Linux

https://github.com/i-maravic/iproute2

# MPLS for Linux Labs

by Irina Dumitrascu and Adrian Popa: graduation project with purpose of teaching MPLS to university students, at Limburg Catholic University College

http://ontwerpen1.khlim.be/~lrutten/cursussen/comm2/mpls-linux-docs/

inlcudes e.g. Layer 2 VPN with MPLS, Layer 3 VPN with MPLS

Virtual Private Networks

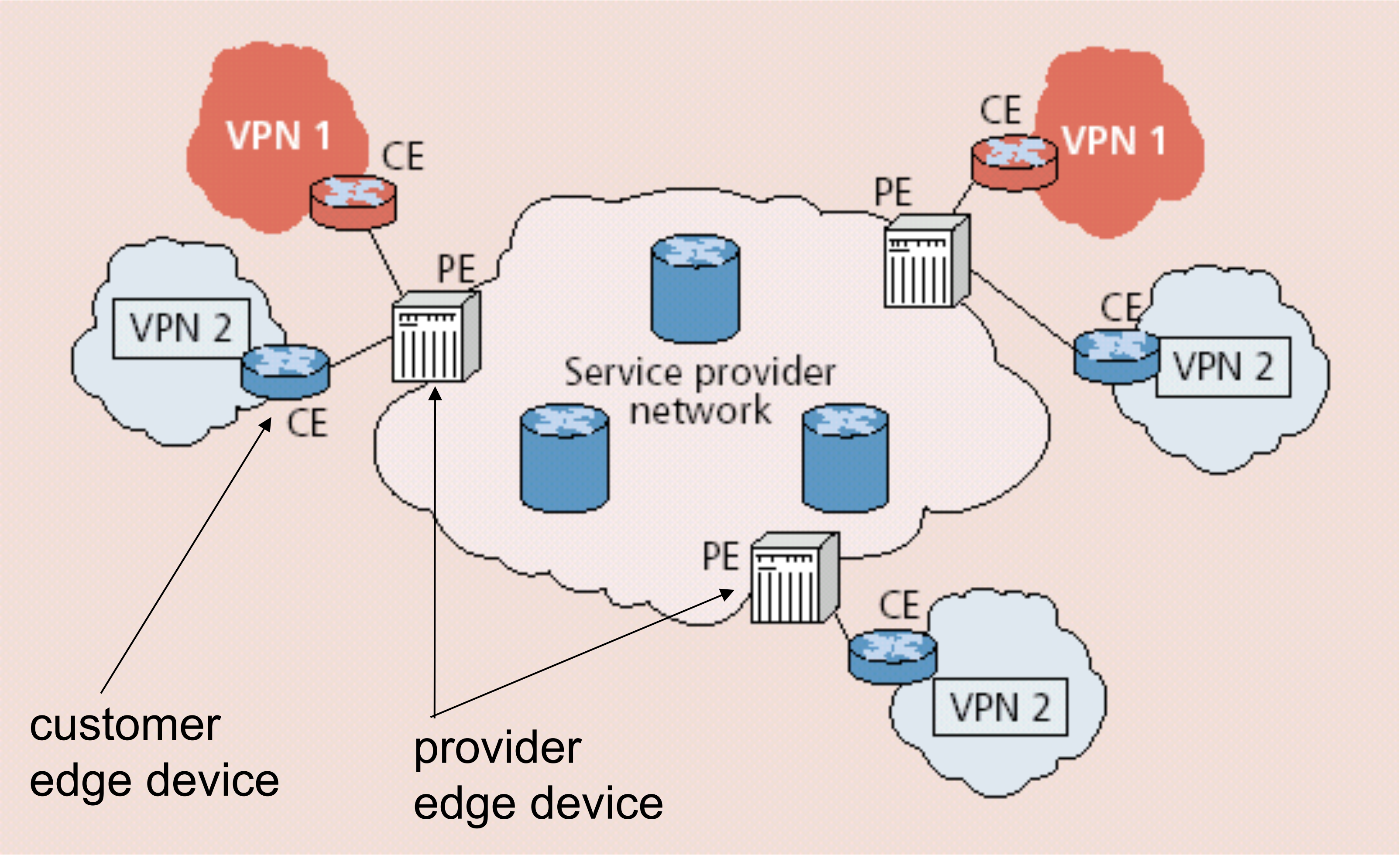
Virtual Private Networks (VPN)

VPNs

Networks perceived as being private networks by customers using them, but built over shared infrastructure owned by service provider (SP)

- ❑ Service provider infrastructure:
  - ▪ backbone
  - ▪ provider edge devices
- ❑ Customer:
  - ▪ customer edge devices
(communicating over shared backbone)

VPN Reference Architecture



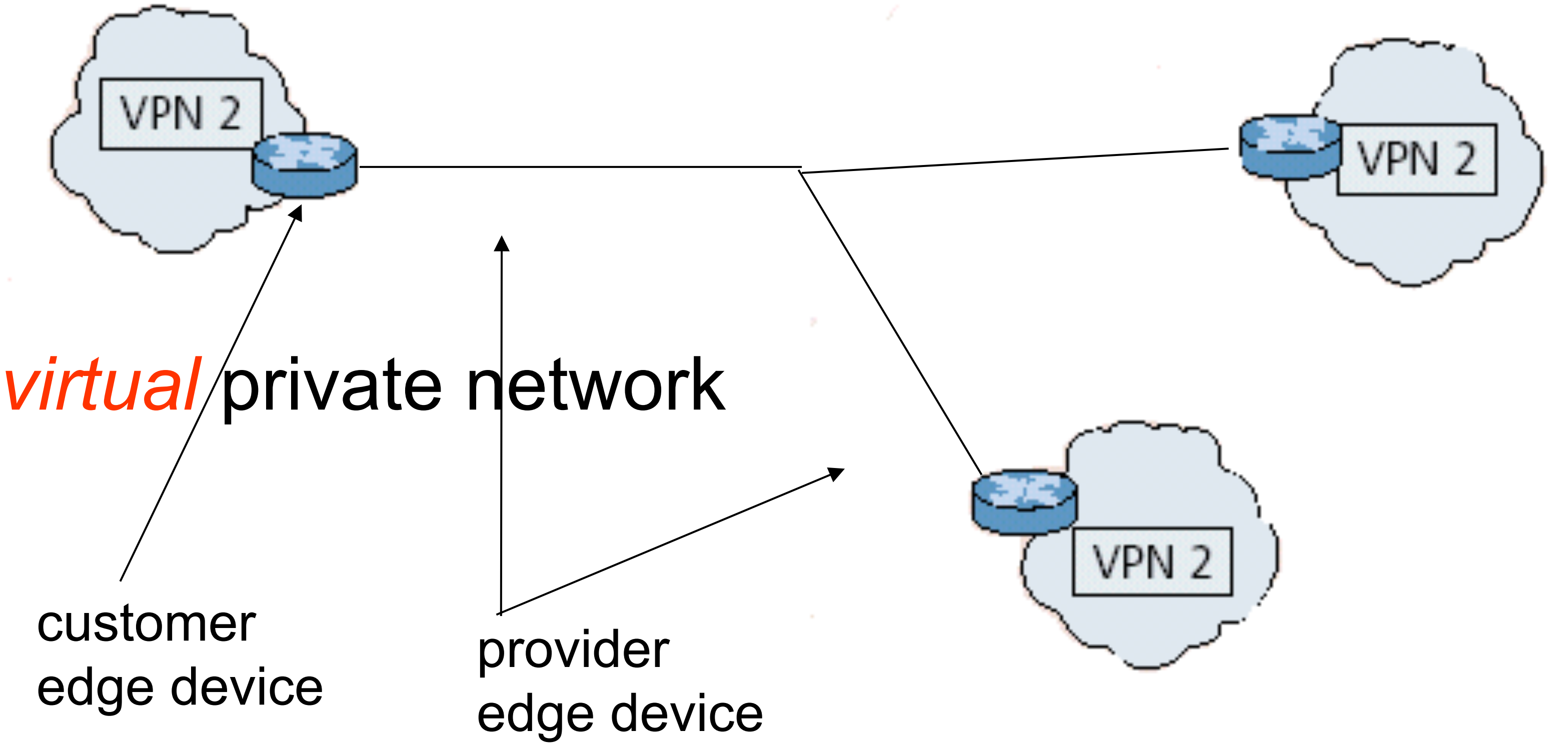
customer
edge device

provider
edge device

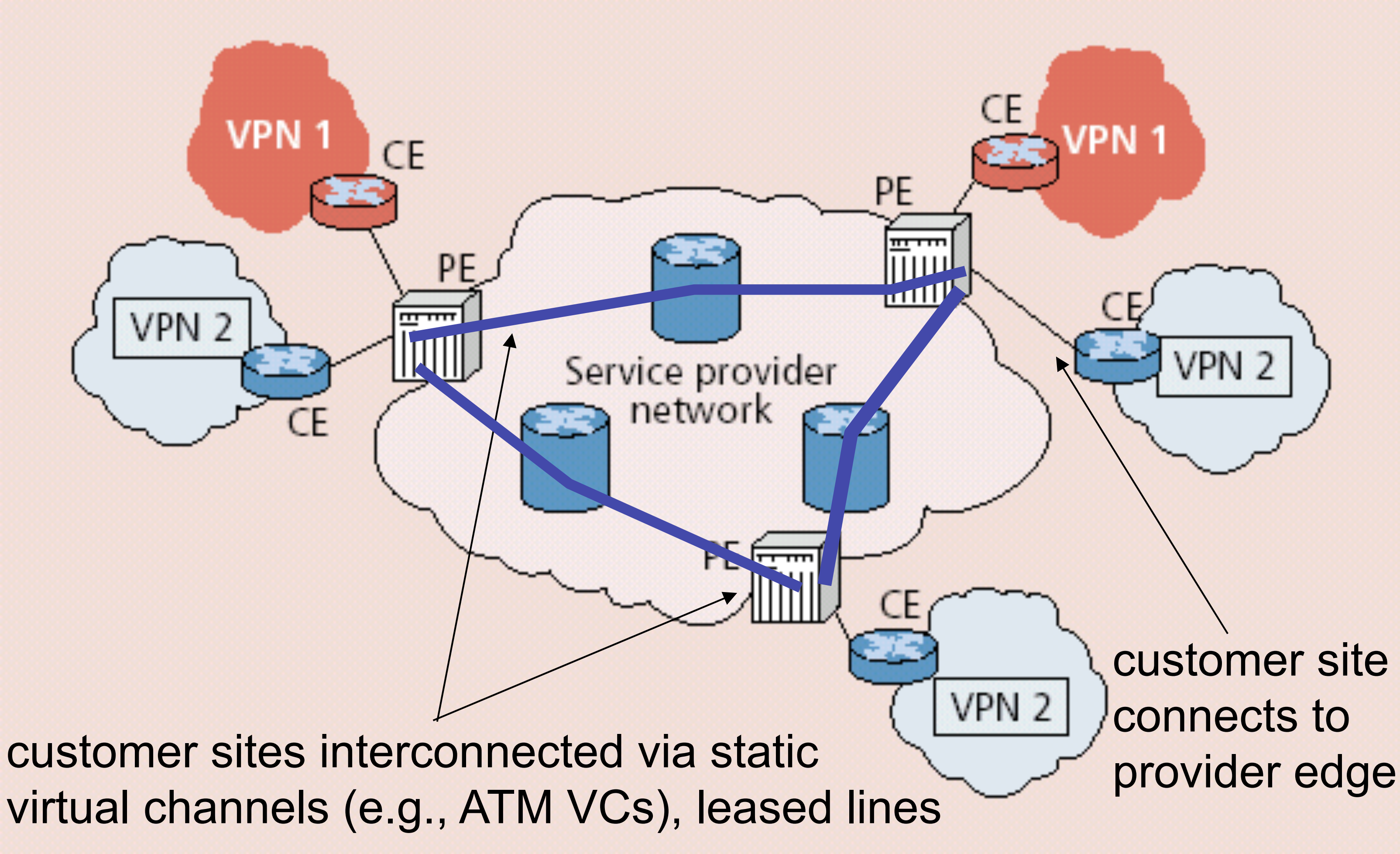
VPNs: Why?

- Privacy
- Security
- Works well with mobility (looks like you are always at home)
- Cost
  - many forms of newer VPNs are cheaper than leased line VPNs
  - ability to share at lower layers even though logically separate means lower cost
  - exploit multiple paths, redundancy, fault-recovery in lower layers
  - need isolation mechanisms to ensure resources shared appropriately
- Abstraction and manageability
  - all machines with addresses that are "in" are trusted no matter where they are

VPN: logical view



Leased-Line VPN

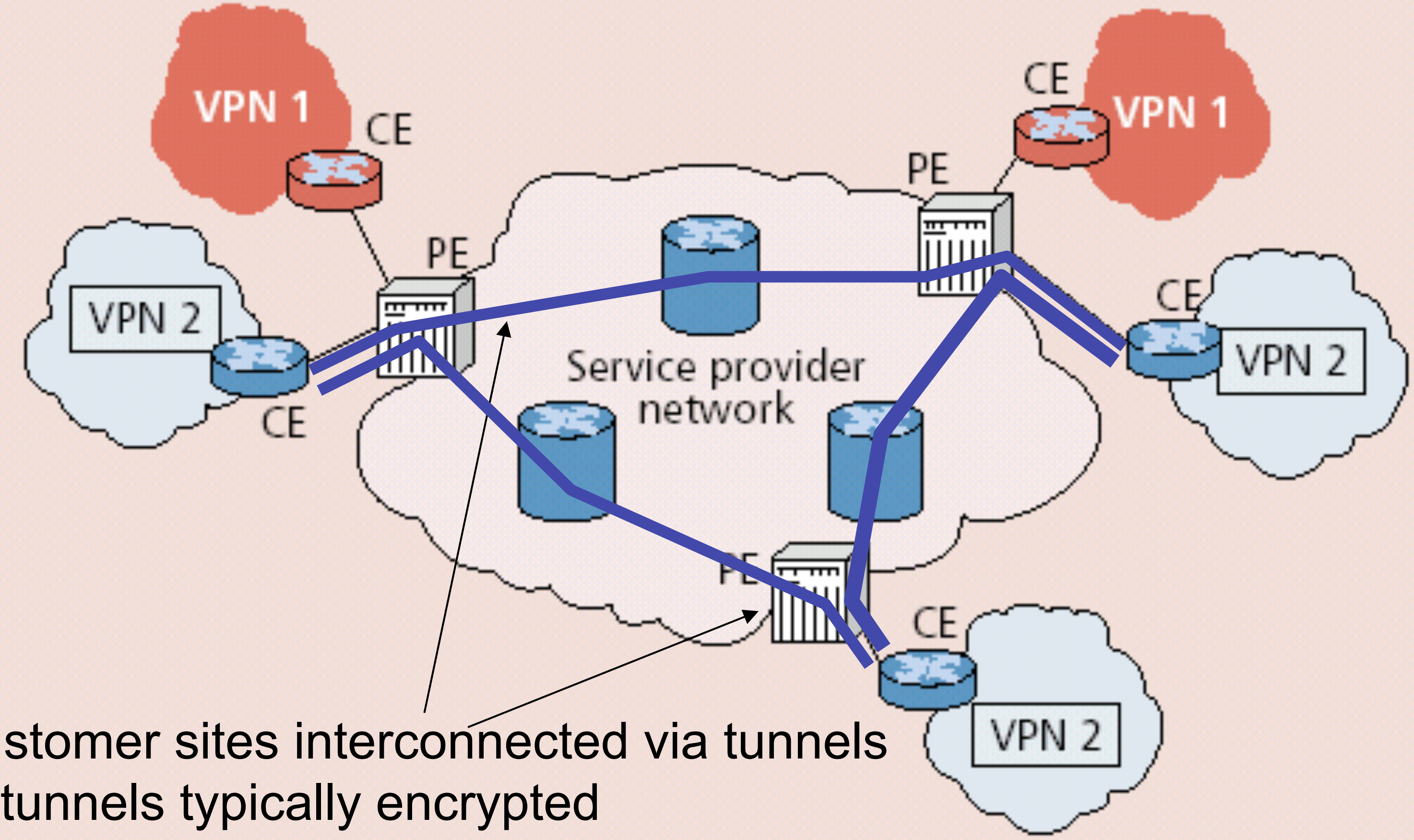


customer sites interconnected via static virtual channels (e.g., ATM VCs), leased lines

customer site connects to provider edge

Customer Premise VPN

- ❑ all VPN functions implemented by customer



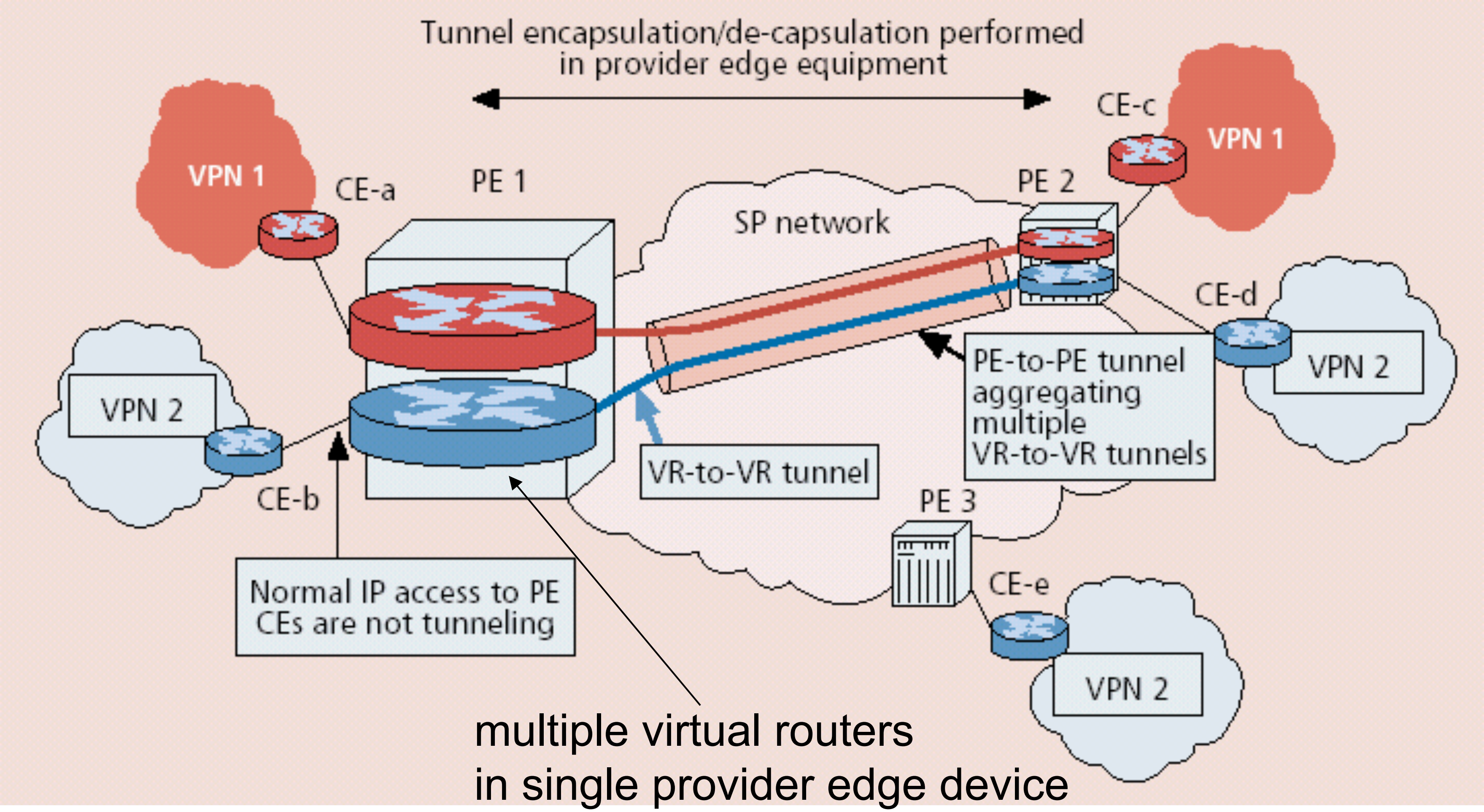
customer sites interconnected via tunnels

- ❑ tunnels typically encrypted
- ❑ Service provider treats VPN packets like all other packets

Variants of VPNs

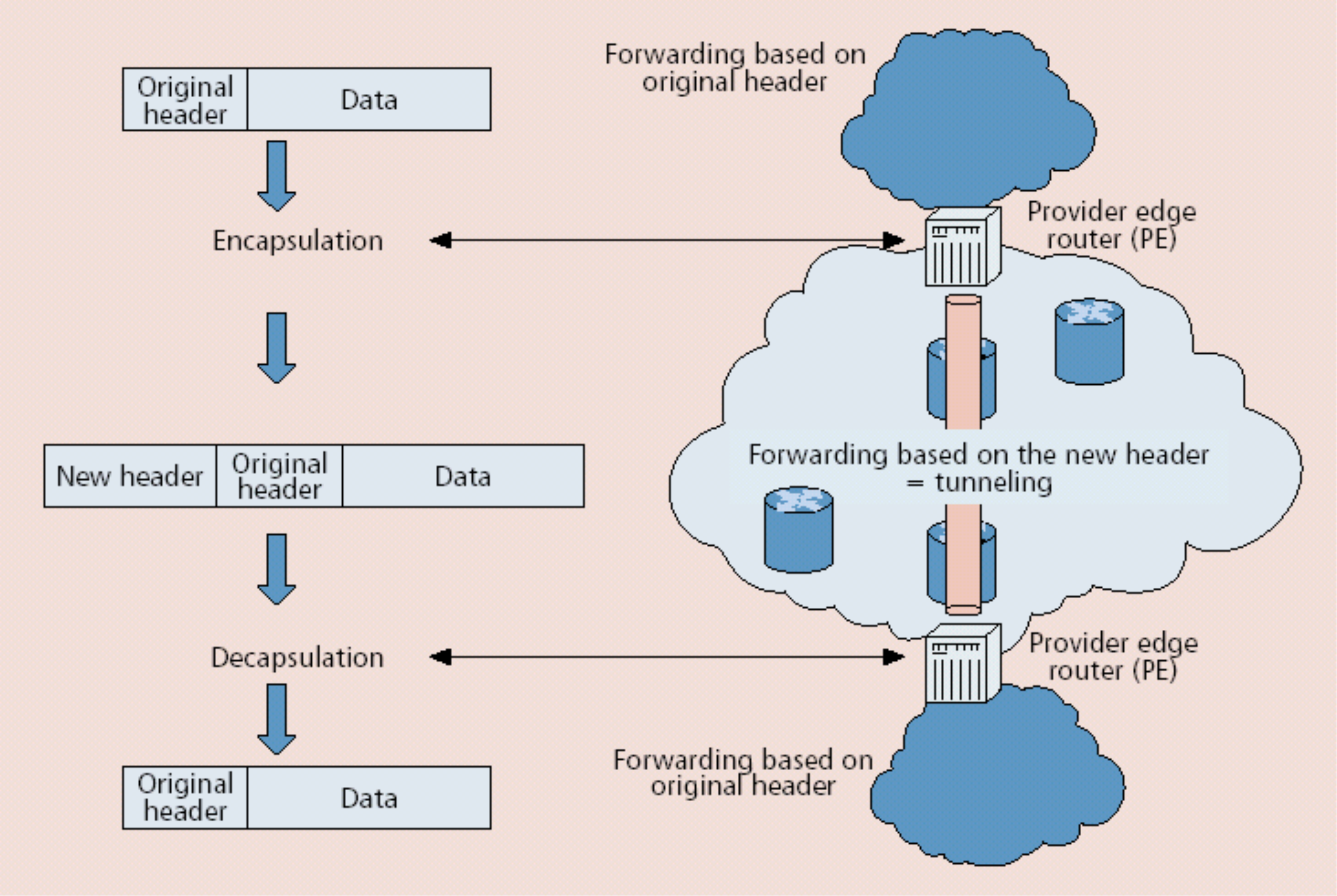
- Leased-line VPN
  - configuration costs and maintenance by service provider: long time to set up, manpower
- CPE-based VPN
  - expertise by customer to acquire, configure, manage VPN
- Network-based VPN
  - Customer routers connect to service provider routers
  - Service provider routers maintain separate (independent) IP contexts for each VPN
    - sites can use private addressing
    - traffic from one VPN cannot be injected into another

Network-based Layer 3 VPNs

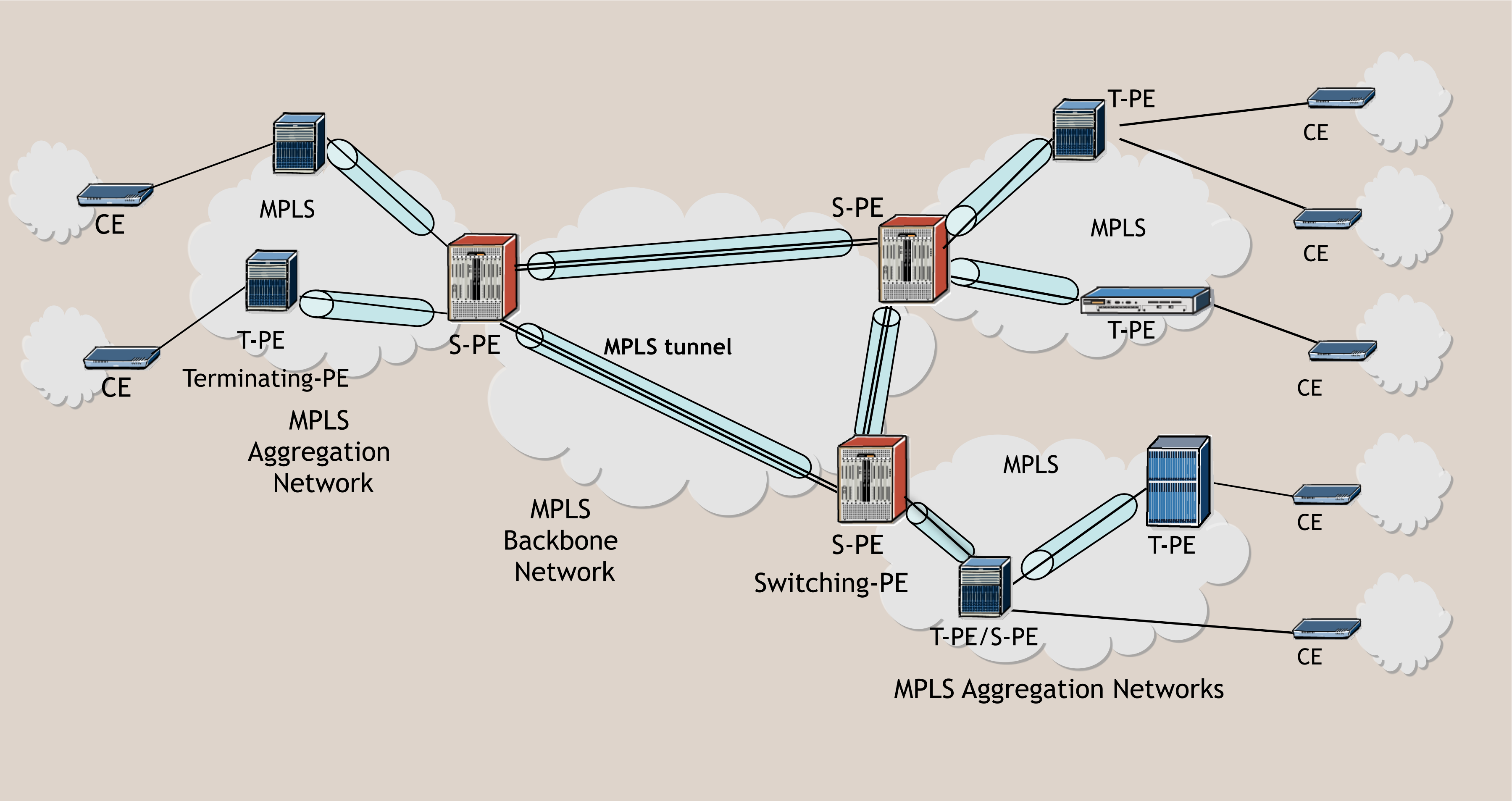

Tunnel encapsulation/de-capsulation performed in provider edge equipment
VPN 1
CE-a
PE 1
CE-c
VPN 1
PE 2
SP network
CE-d
VPN 2
VPN 2
CE-b
PE-to-PE tunnel aggregating multiple VR-to-VR tunnels
VR-to-VR tunnel
PE 3
Normal IP access to PE CEs are not tunneling
CE-e
VPN 2

multiple virtual routers
in single provider edge device

Tunneling


Original header
Data
Encapsulation
New header
Original header
Data
Decapsulation
Original header
Data
Forwarding based on original header
Provider edge router (PE)
Forwarding based on the new header = tunneling
Provider edge router (PE)
Forwarding based on original header

MPLS-based VPN



NAT and NAT Traversal

Overview

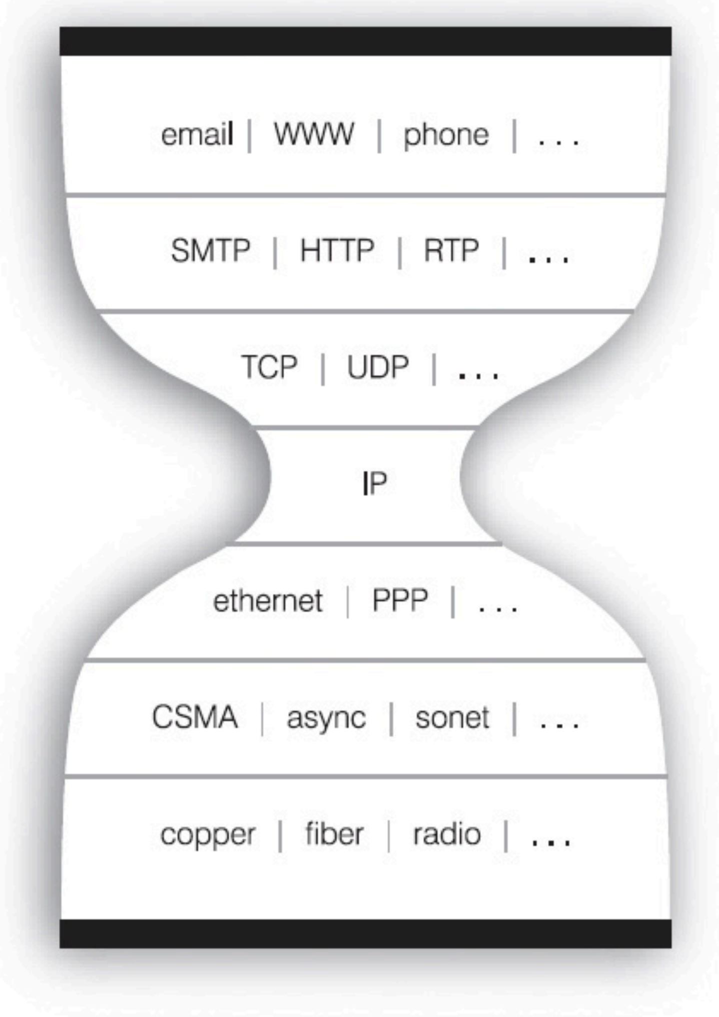
- Introduction to Network Address Translation
- Behavior of NAT
- The NAT Traversal problem
- Solutions to the problem
- Large Scale NATs

Problem

- ❑ More and more devices connect to the Internet
  - PCs
  - Cell phones
  - Internet radios
  - TVs
  - Home appliances
  - Future: sensors, cars...

- ❑ IP addresses need to be globally unique
  - IPv4 provides a 32bit field
  - Many addresses not usable because of classful allocation

→ We are running out of IP addresses


email | WWW | phone | . . .
SMTP | HTTP | RTP | . . .
TCP | UDP | . . .
IP
ethernet | PPP | . . .
CSMA | async | sonet | . . .
copper | fiber | radio | . . .

Address Space

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

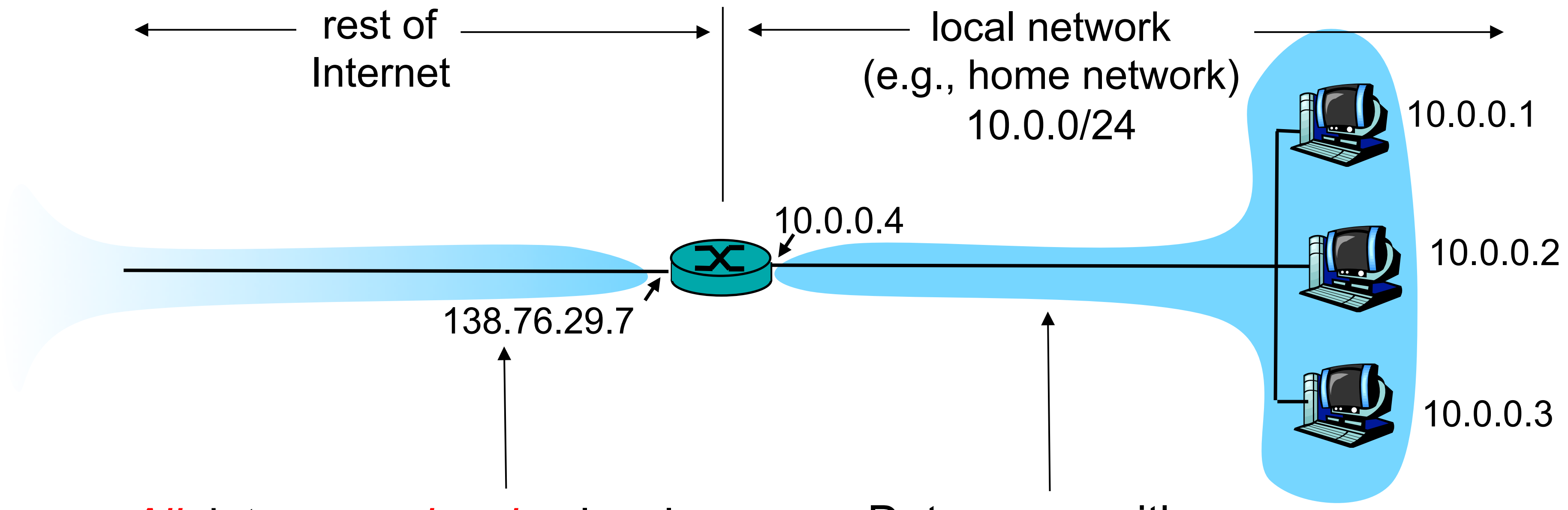
- RFC 1918 (published in 1996) directs IANA to reserve the following IPv4 address ranges for private networks
  - 10.0.0.0 – 10.255.255.255
  - 172.16.0.0 – 172.31.255.255
  - 192.168.0.0 – 192.168.255.255

- 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 hosts 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 (always on)
  - 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:

- *On outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)

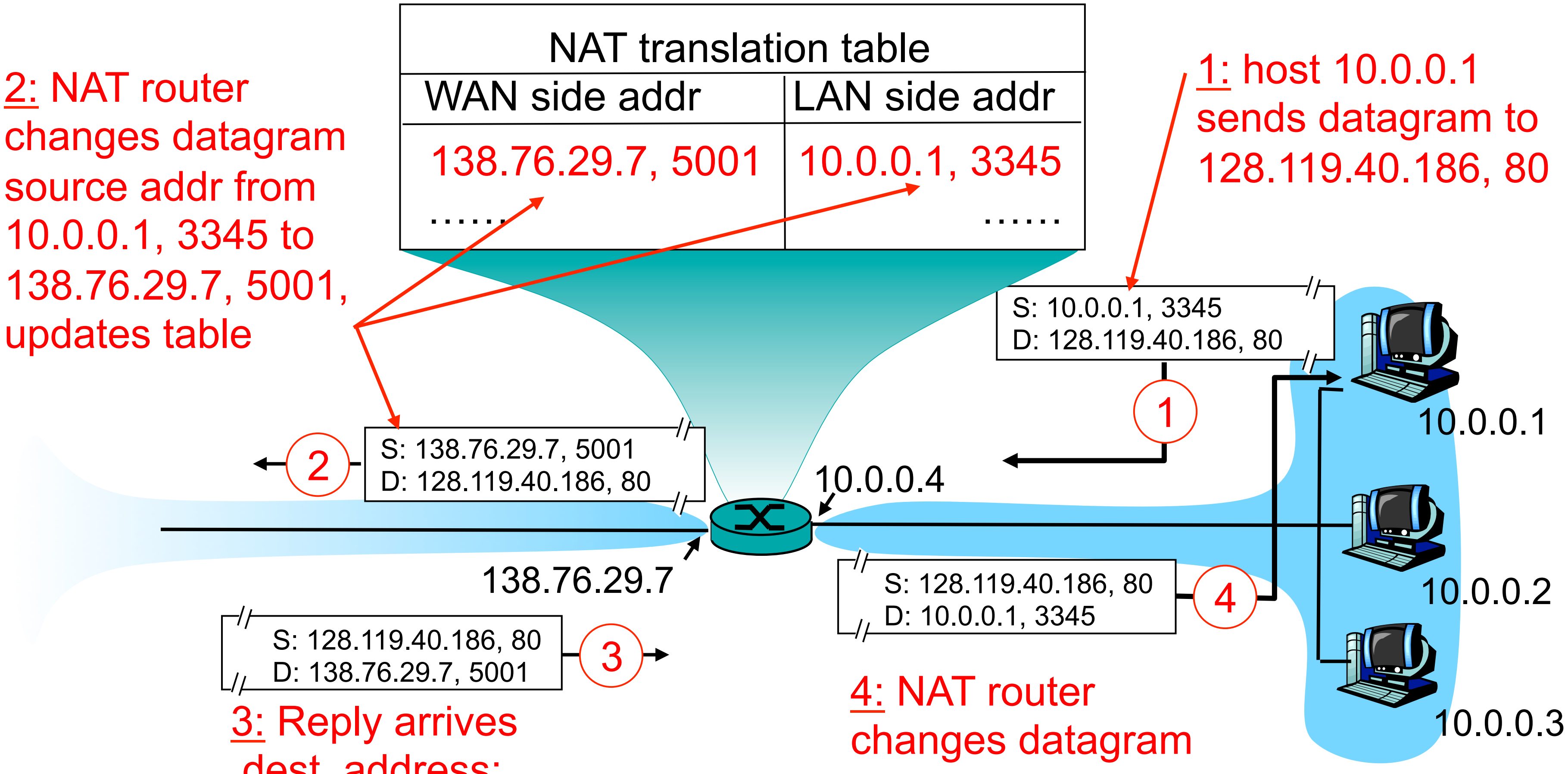
  . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr.

- *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
……	……

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

S: 10.0.0.1, 3345
D: 128.119.40.186, 80

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

S: 138.76.29.7, 5001
D: 128.119.40.186, 80

S: 128.119.40.186, 80
D: 138.76.29.7, 5001

3: Reply arrives dest. address: 138.76.29.7, 5001

S: 128.119.40.186, 80
D: 10.0.0.1, 3345

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

NAT: Network Address Translation

- NAPT:
  - ~65000 simultaneous connections with a single LAN-side address!
  - helps against the IP shortage
  - More advantages:
    - we can change addresses of devices in local network without notifying outside world
    - we can change ISP without changing local addresses
    - devices inside local net not explicitly addressable/visible by the outside world (a security plus)
- NAT is controversial:
  - routers should only process up to layer 3
  - violates end-to-end argument

NAT Behavior and 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 (which external mapping is allocated)
    - NAT binding
    - Port binding
  - Endpoint filtering (who is allowed to access the mapping)

Binding

- When creating a new state, the NAT has to assign a new source port and IP address to the connection

- **Port binding** describes the strategy a NAT uses for the assignment of a new external source port
  - Port Preservation (if possible)
  - Some algorithm (e.g. +1)
  - Random

NAT binding

- **NAT binding** describes the behavior of the NAT regarding the reuse of an existing binding
  - two consecutive connections from the same transport address (combination of IP address and port)
  - 2 different bindings?
  - If the binding is the same → Port prediction possible

- Endpoint Independent
  - the external port is only dependent on the source transport address
  - both connections have the same IP address and port

- Endpoint 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)

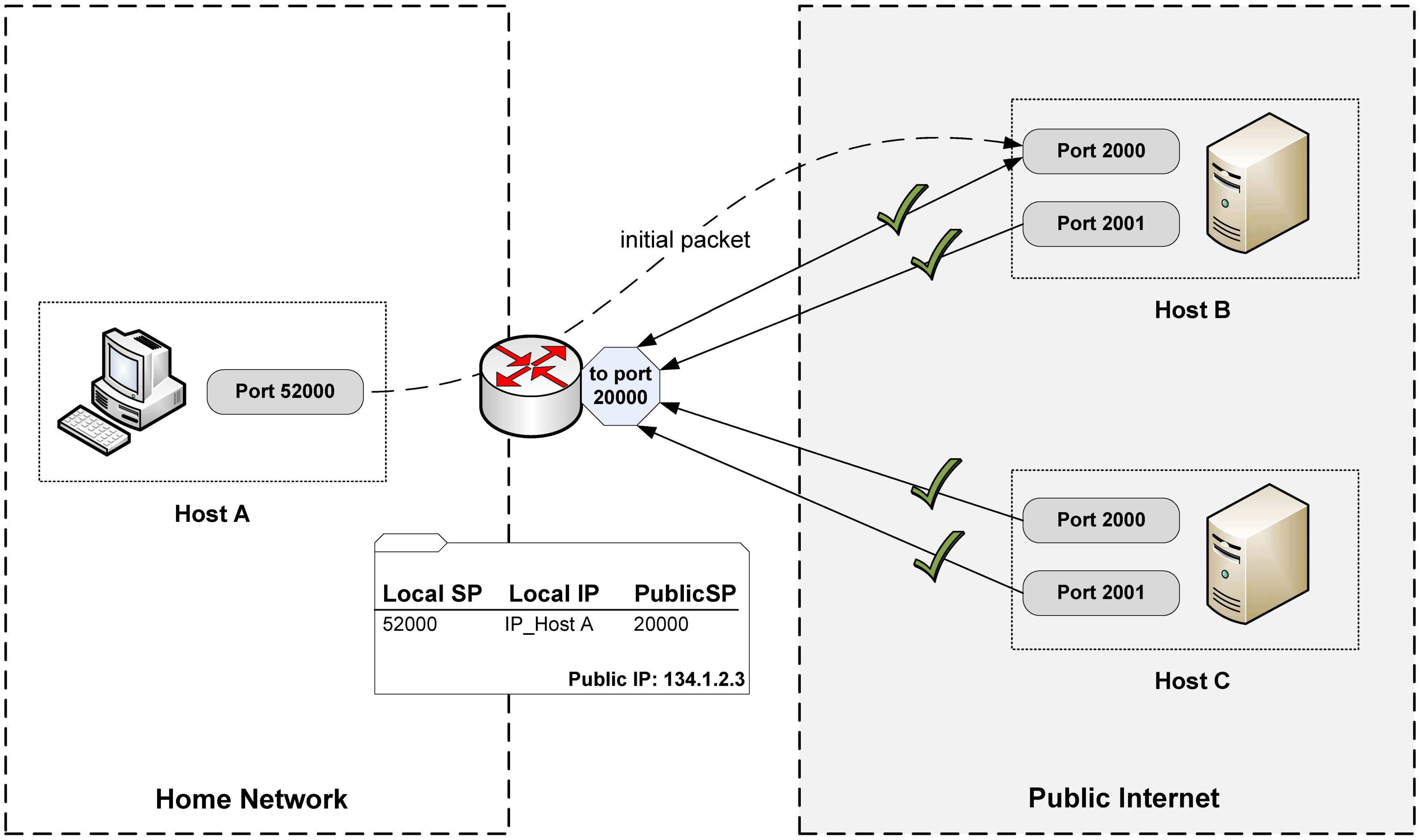
- **Full Cone NAT**
  - **Endpoint independent**
  - **Independent filtering**

- 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



Local SP	Local IP	PublicSP
52000	IP_Host A	20000

Public IP: 134.1.2.3

NAT Types

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

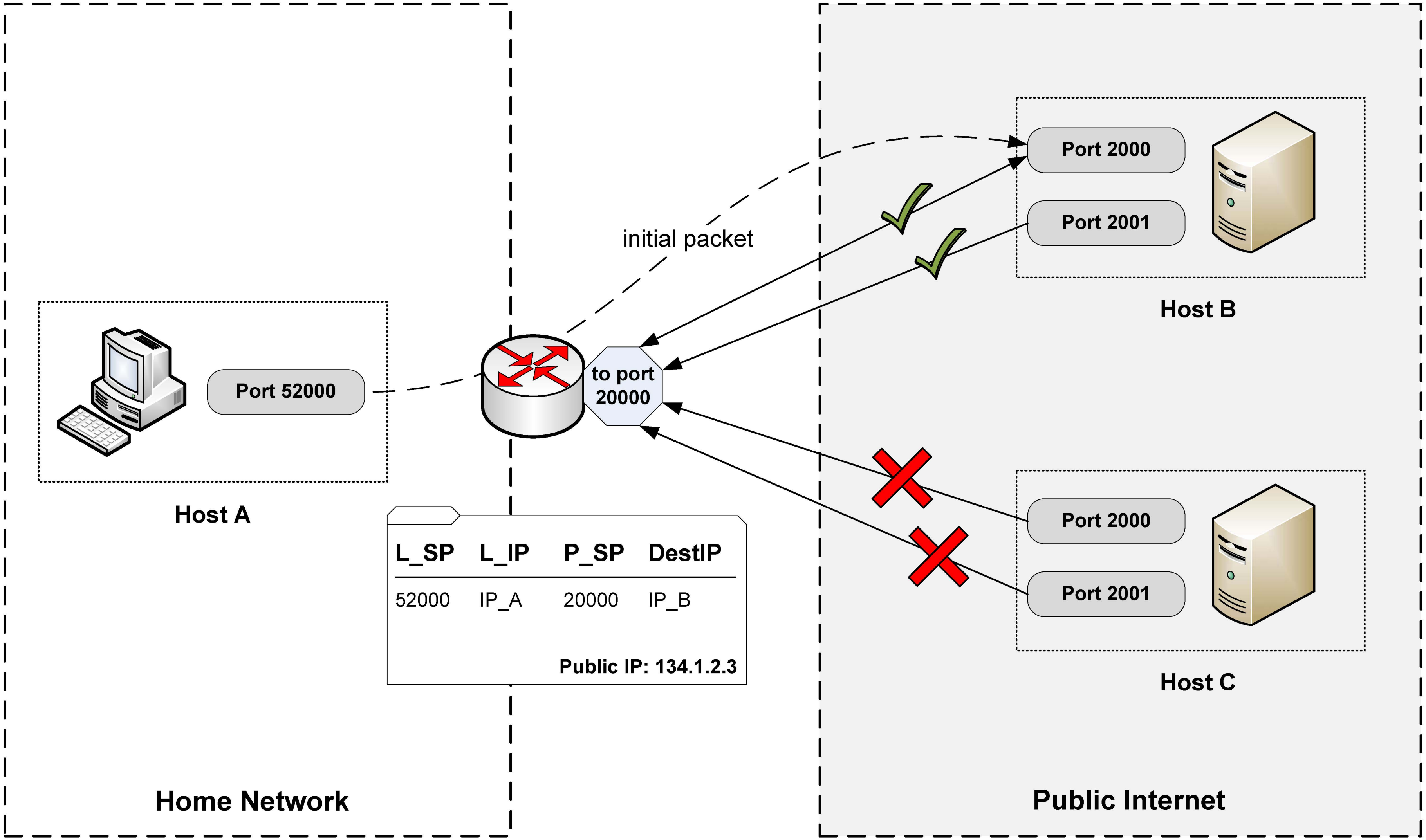
- Full Cone NAT
  - Endpoint independent
  - Independent filtering

- **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

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)

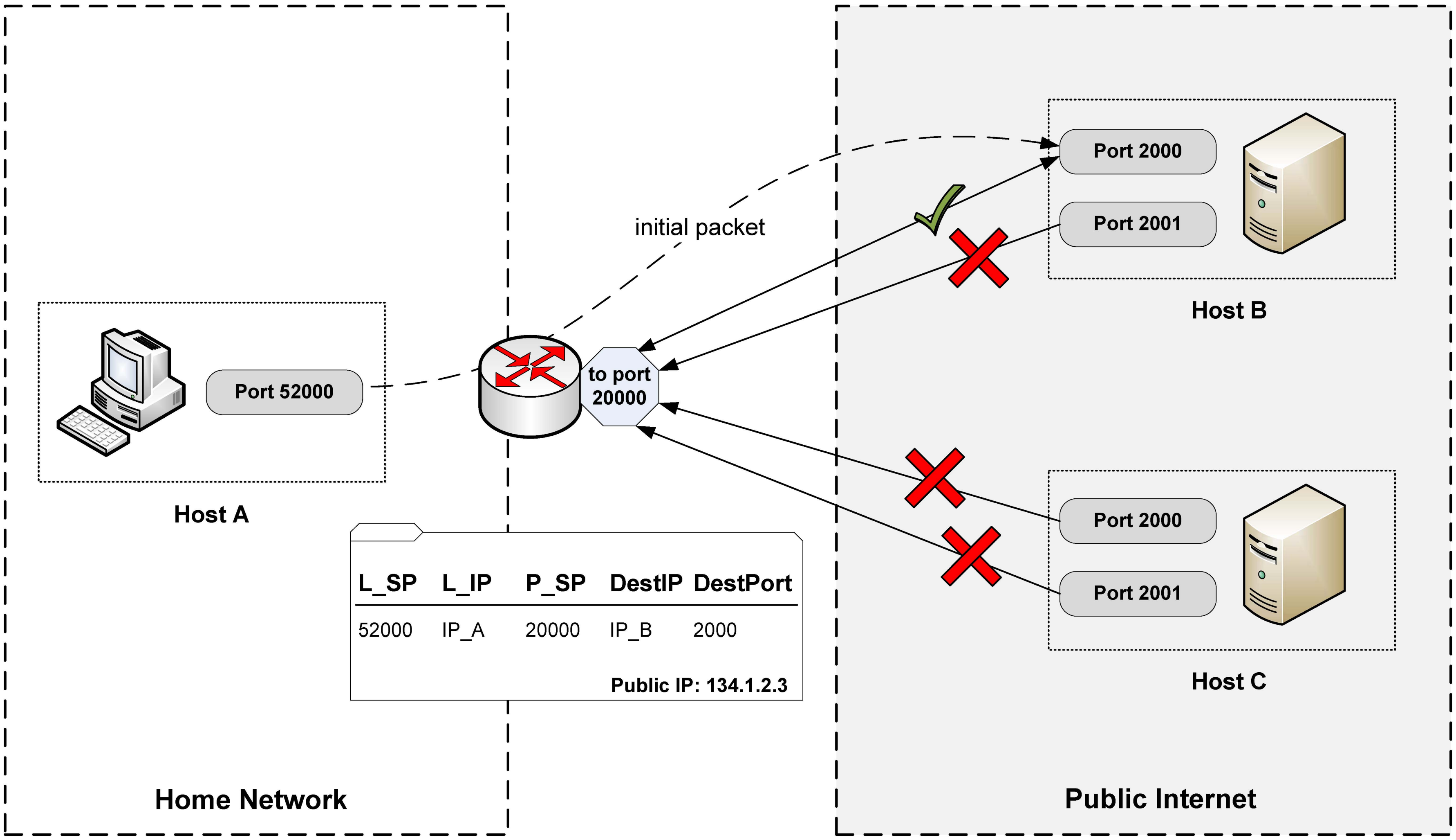
- Full Cone NAT
  - Endpoint independent
  - Independent filtering

- 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)

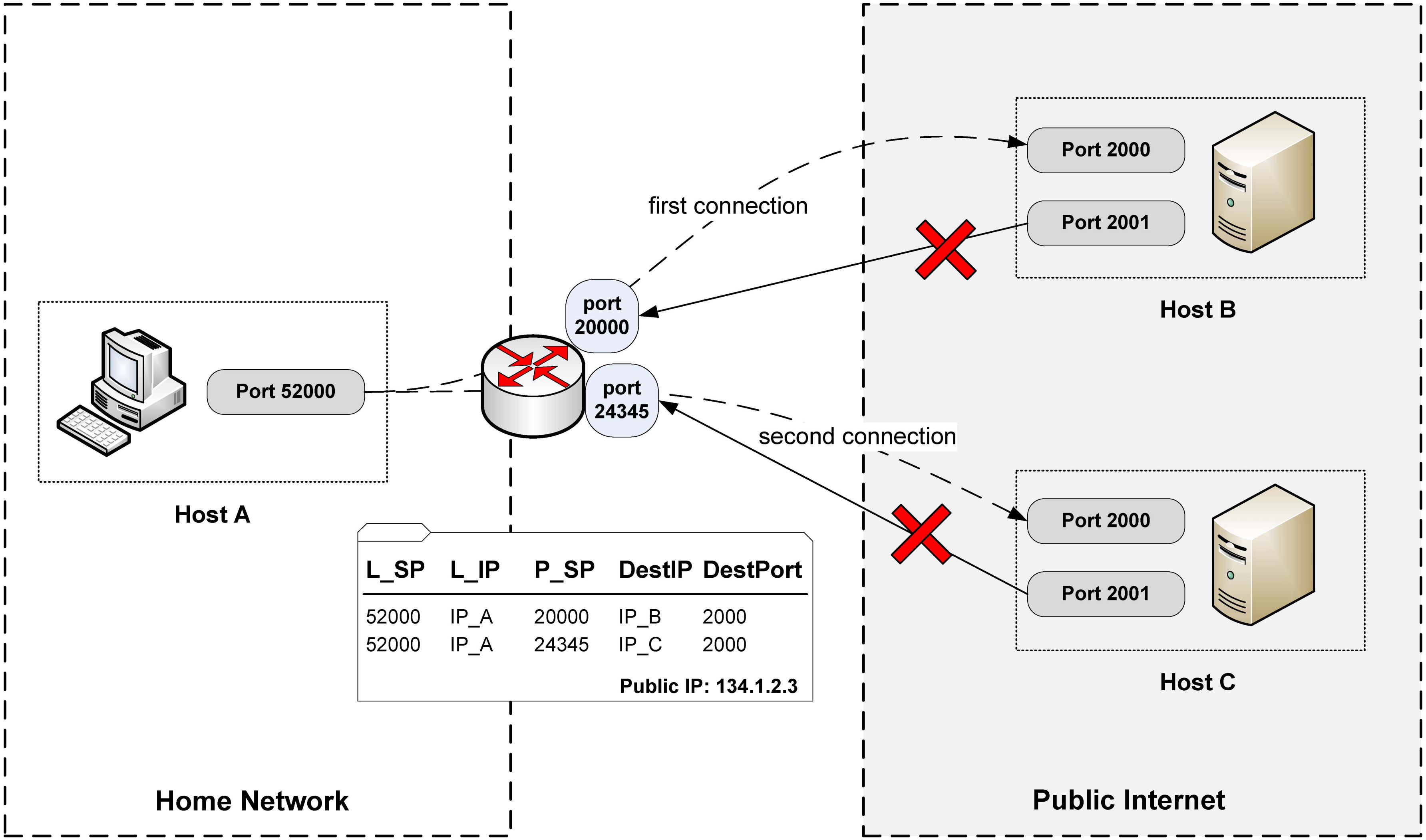
- Full Cone NAT
  - Endpoint independent
  - Independent filtering

- 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

And where is the problem?

- NAT was designed for the client-server paradigm

- Nowadays the internet consists of applications such as
  - P2P networks
  - Voice over IP
  - Multimedia Streams

- Protocols are getting more and more complex
  - Multiple layer 4 connections (data and control session)
  - Realm specific addresses in layer 7

- Connectivity requirements have changed
  - P2P is becoming more and more important
    - Especially for future home and services
  - Direct connections between hosts is necessary

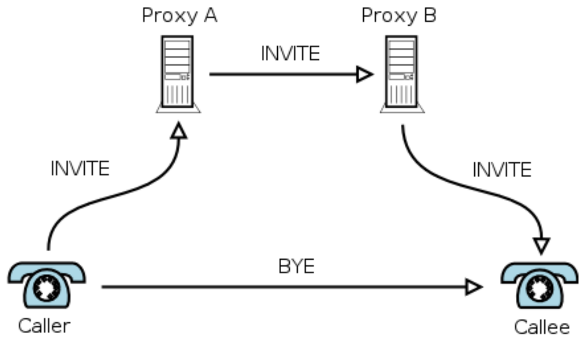
- NATs break the end-to-end connectivity model of the internet
  - Inbound packets can only be forwarded if an appropriate mapping exists
  - Mappings are only created on outbound packets

NAT-Traversal Problem

- Divided into four categories: (derived from IETF-RFC 3027)
  - **Realm-Specific IP-Addresses in the Payload**
    - *Session Initiation Protocol (SIP)*
  - **Peer-to-Peer Applications**
    - *Any service behind a NAT*
  - **Bundled Session Applications (Inband Signaling)**
    - *FTP*
    - *Real time streaming protocol (RTSP)*
    - *SIP together with SDP (Session Description Protocol)*
  - **Unsupported Protocols**
    - *SCTP (Stream Control Transmission Protocol)*
    - *IPSec*

Example: Session Initiation Protocol (SIP)

- ❑ Realm Specific IP addresses in the payload (SIP)
- ❑ Bundled Session Application (RTP)



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

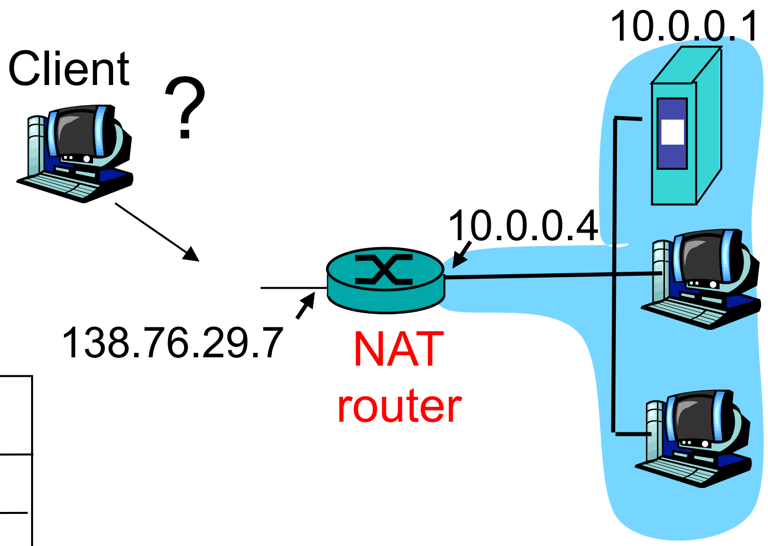
RTP-Session Specification (for 2nd channel)

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
  - NAT does not have any idea where to forward packets to



NAT translation table

WAN side addr	LAN side addr
138.76.29.7, 80	10.0.0.1, 80
……	……

Existing Solutions to the NAT-Traversal Problem

- Individual solutions
  - Explicit support by the NAT
    - Static port forwarding, ALG, 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

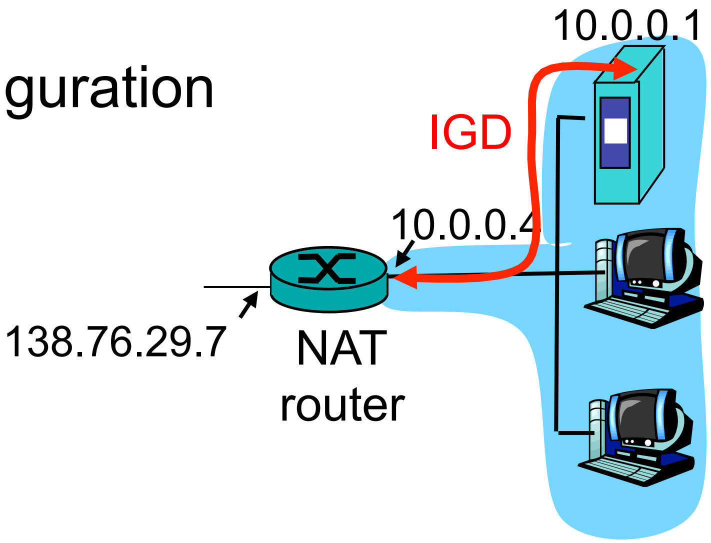
- Advantages
  - transparent for the application
  - no configuration necessary

- 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

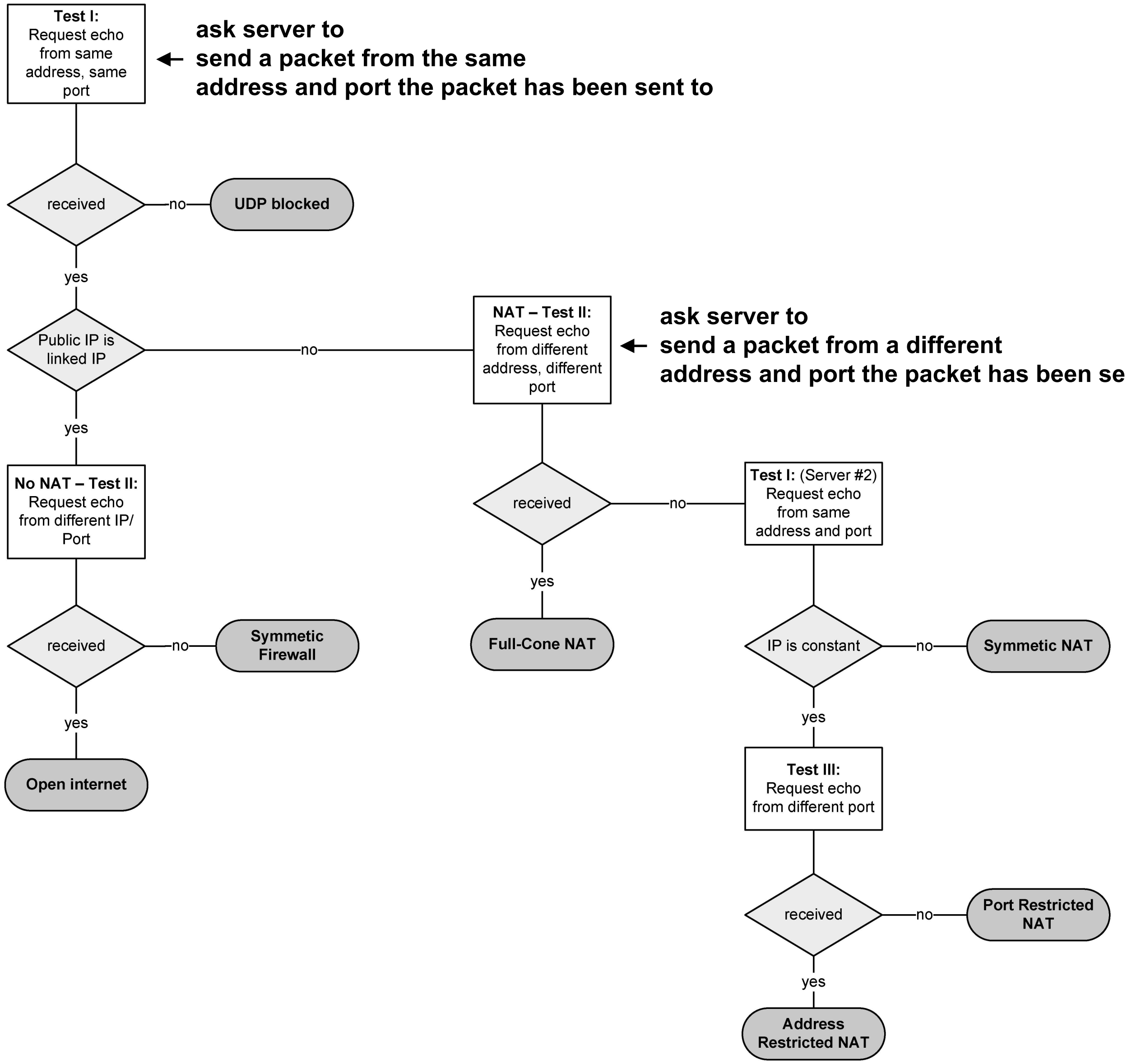
- Simple traversal of UDP through NAT (old) (RFC 3489)
  - Session Traversal Utilities for NAT (new) (RFC 5389)

- 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

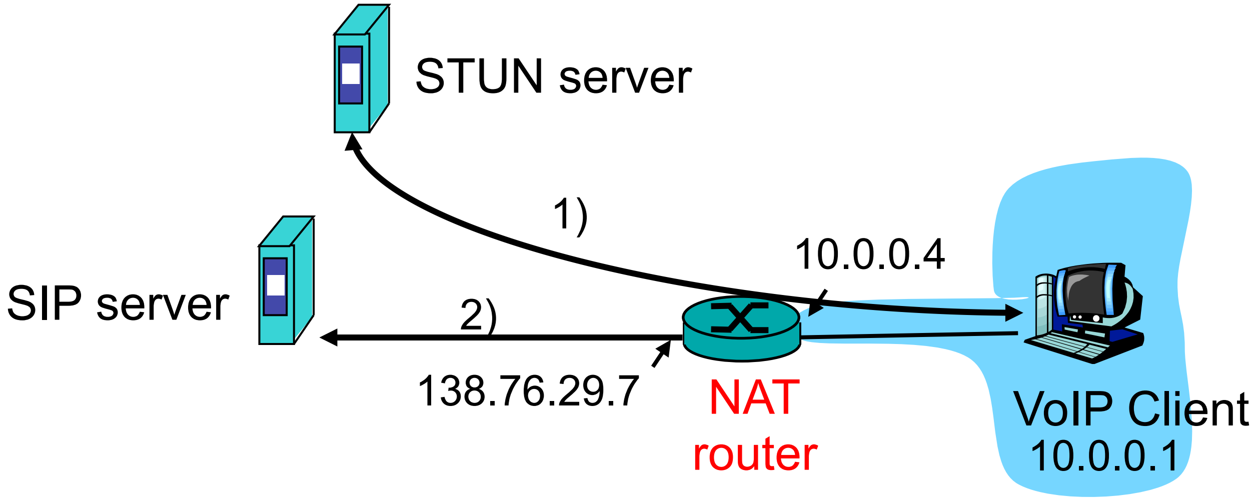


ask server to
send a packet from the same
address and port the packet has been sent to

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
  - The external endpoint must only be dependent on the source transport address

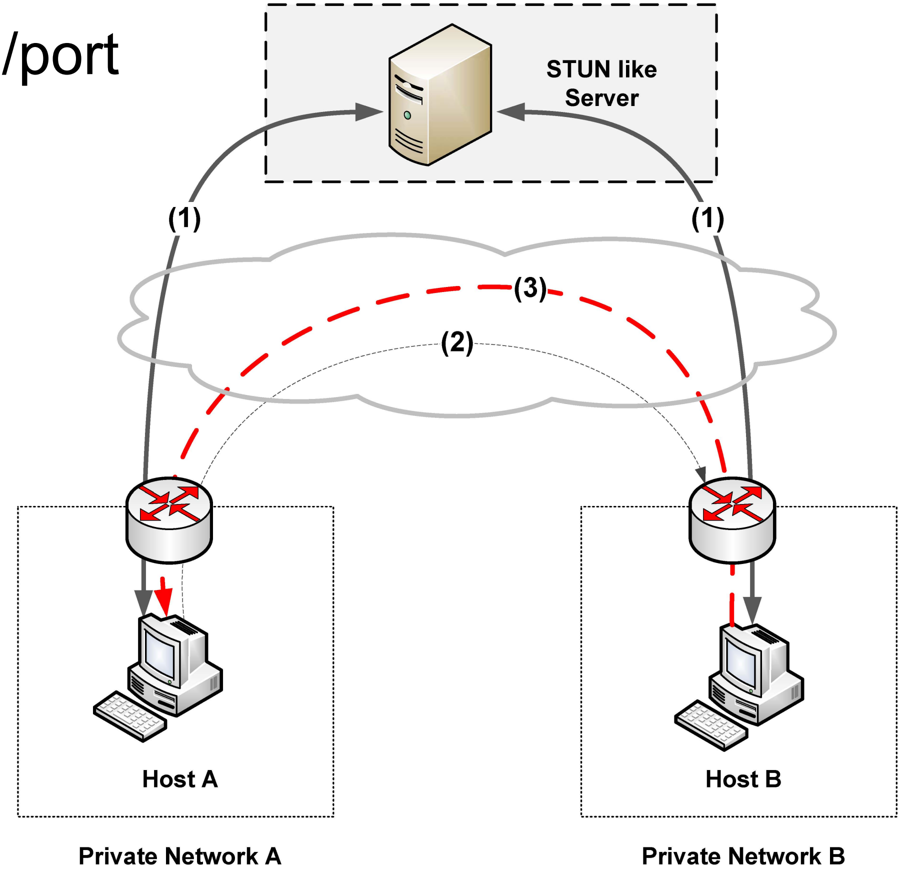
STUN and Hole Punching

- STUN not only helps if we need IP addresses in the payload
  - also for establishing a direct connection between two peers

1) determine external IP address/port and exchange it through Rendezvous Point

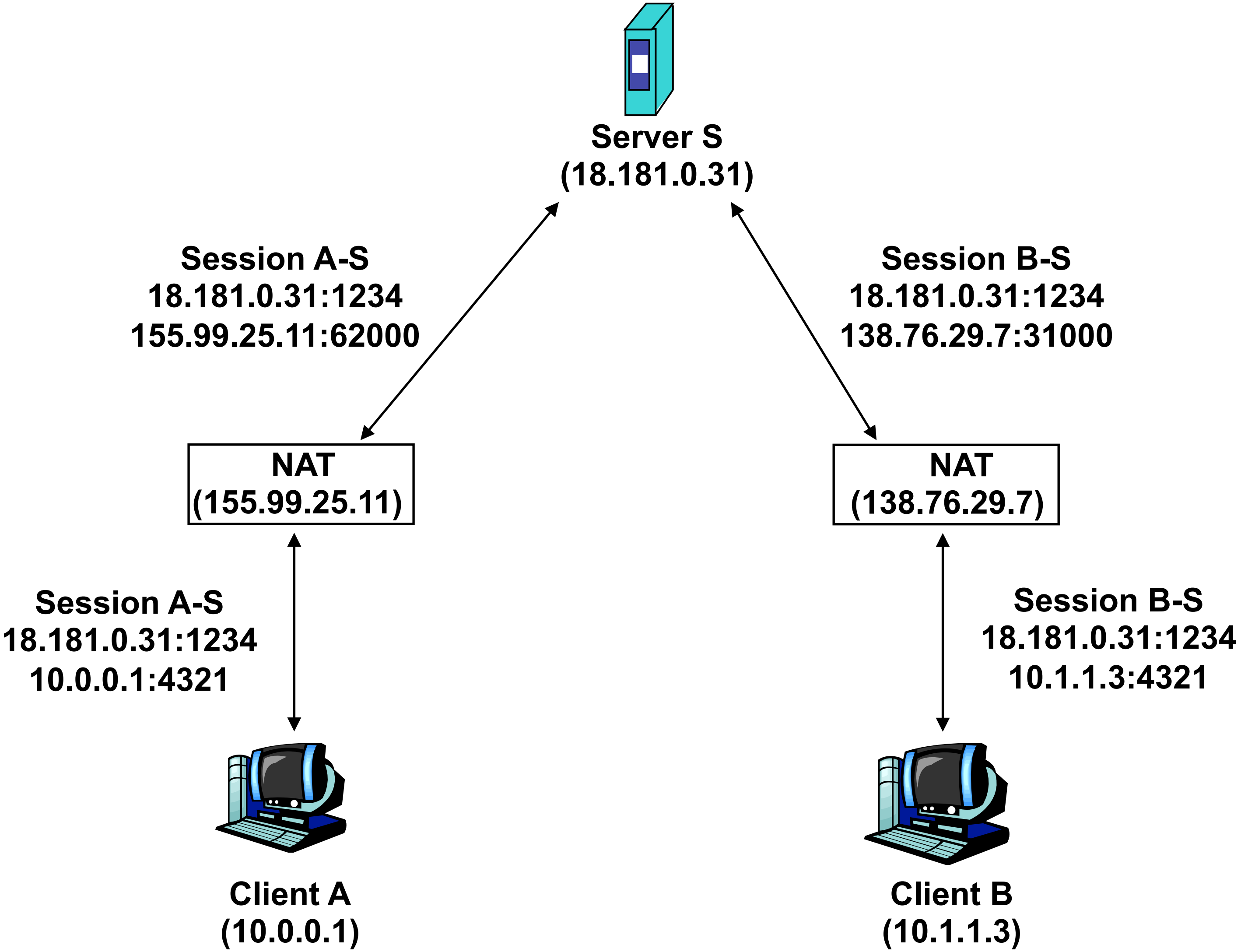
2) both hosts send packets towards the other host outgoing packet creates hole

3) establish connection. hole is created by first packet



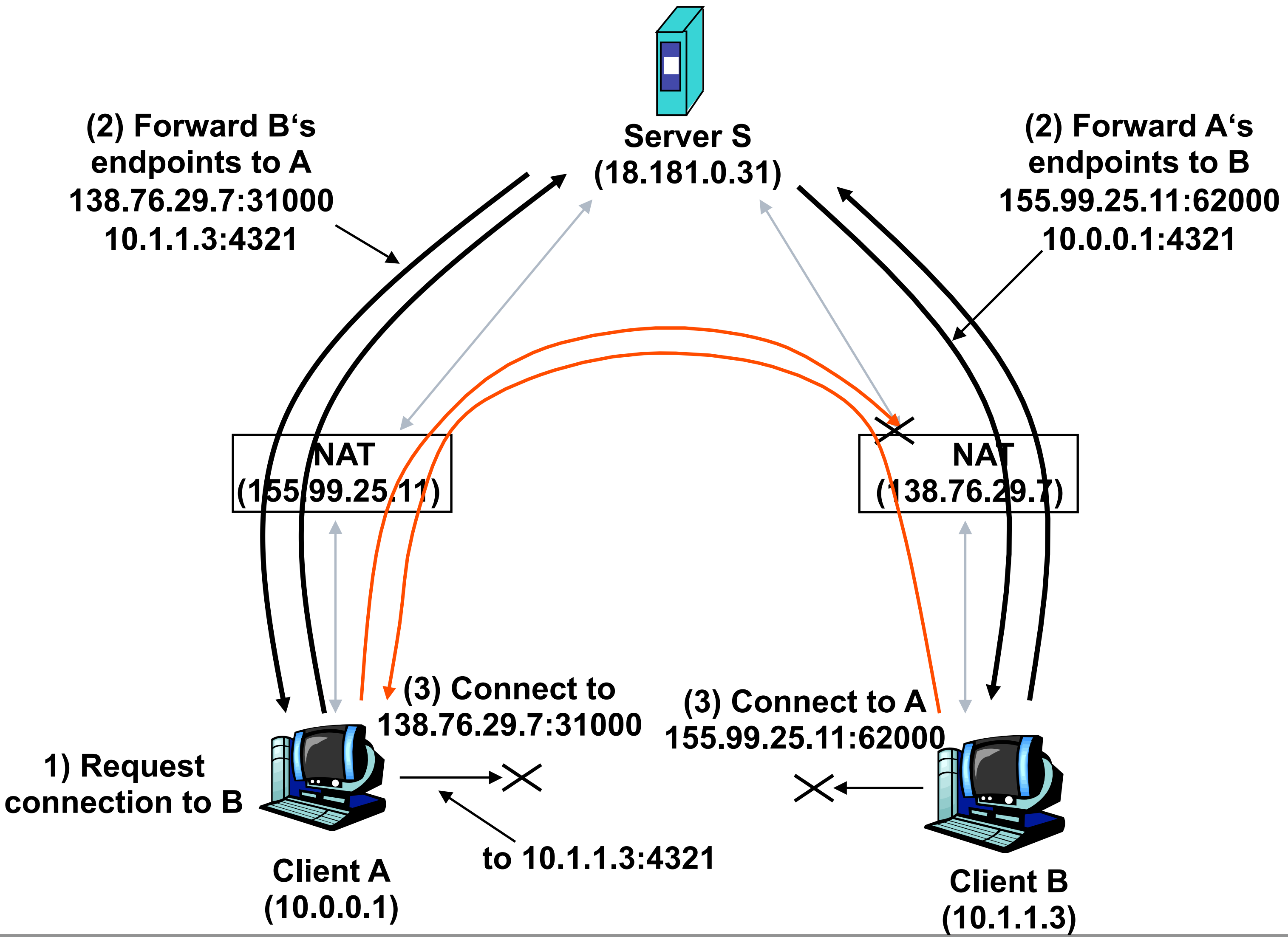
Hole Punching in detail

- 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

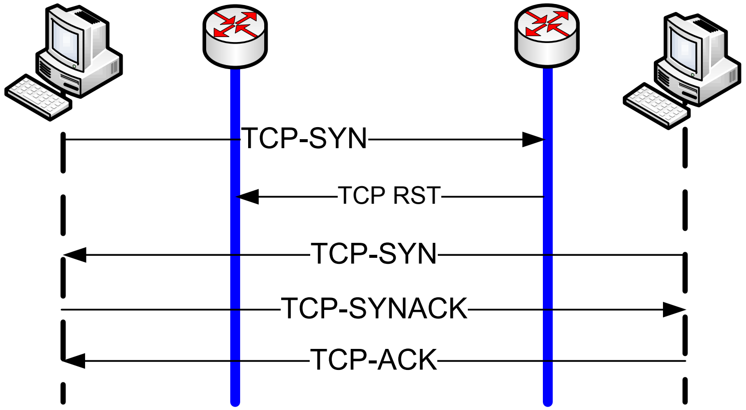
- Now on a second console, server/2, we punch a hole
  - Server/2# hping2 -c 1 -2 -s 20000 -p 5000 clientIP

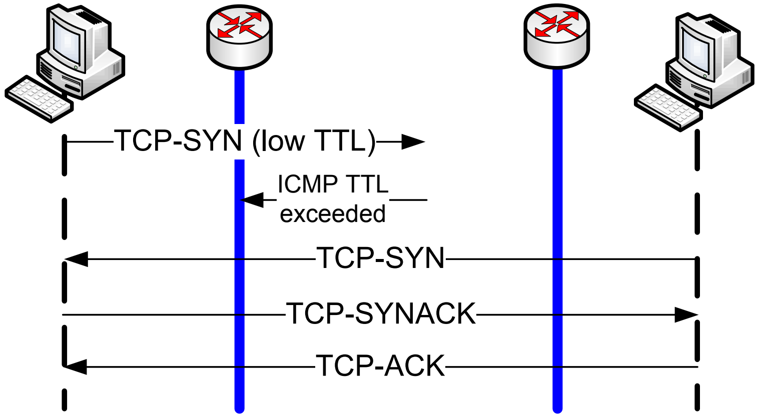
- On the second attempt we connect to the created hole
  - client# echo "hello" | nc -p 5000 -u serverIP 20000

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)

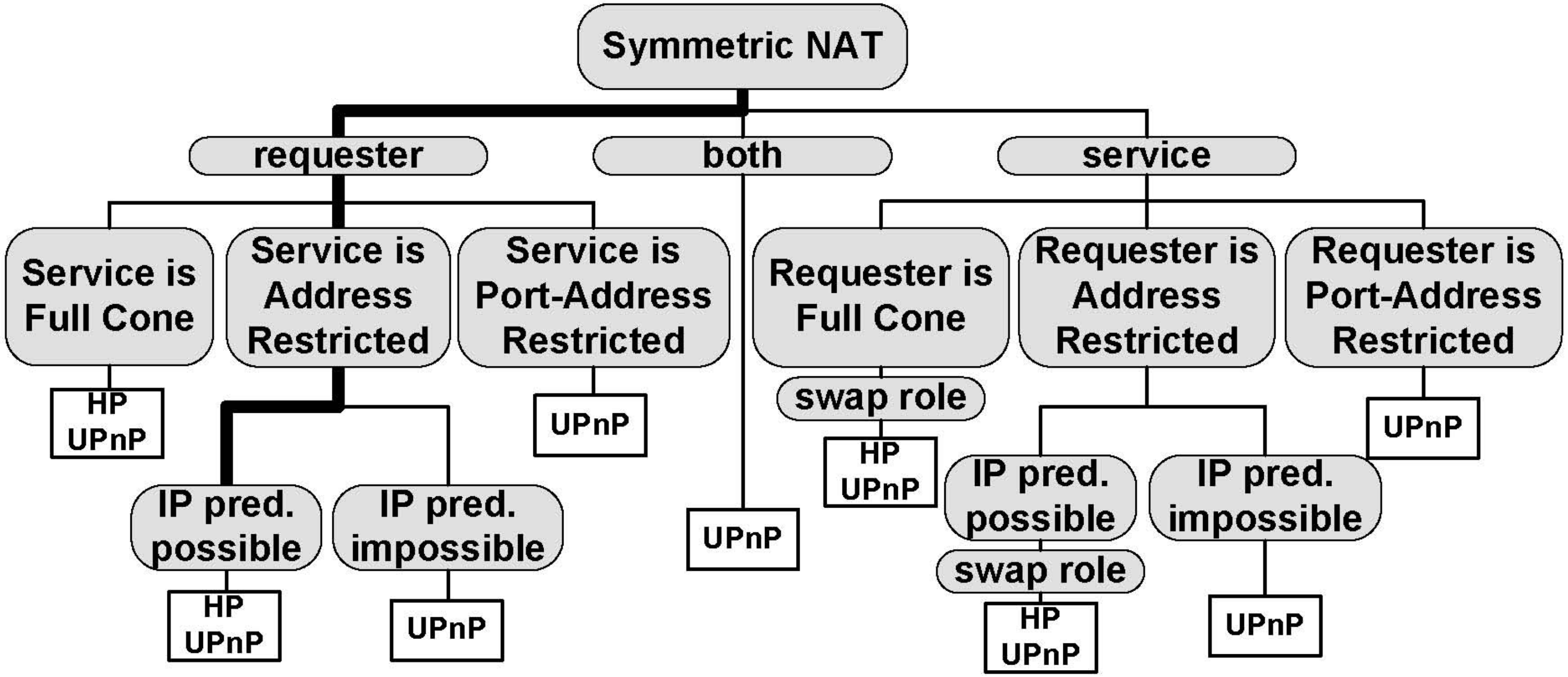




- Bottom line: NAT is not standardized

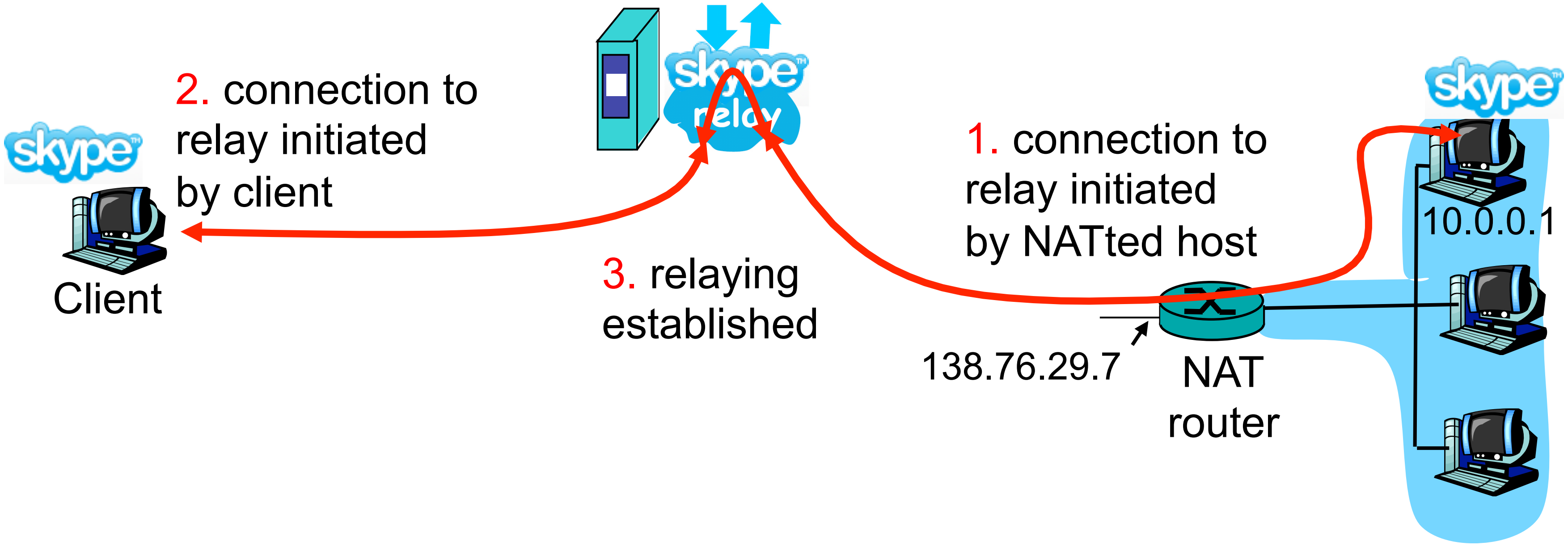
Symmetric NATs

- 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

- relaying (used in Skype)
  - NATed client establishes connection to relay
  - External client connects to relay
  - relay bridges packets between to connections
  - Traversal using Relay NAT (TURN) as IETF draft


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

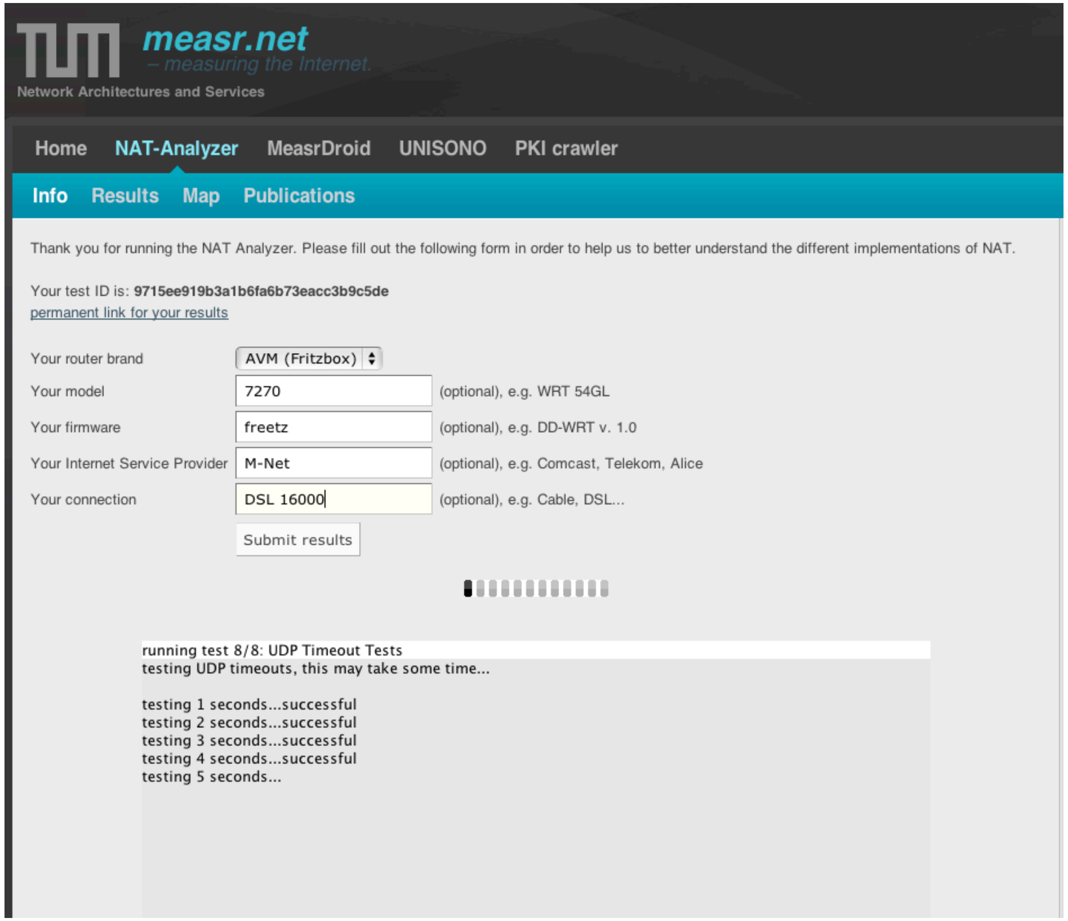
- Advantages
  - high sucess rate
  - integrated in application

- Drawbacks
  - overhead
  - latency dependent on number of endpoints (pairing)

NAT Analyzer - Overview

- Public field test with more than 1500 NATs
  - understand existing traversal techniques and NAT behavior

(http://nattest.net.in.tum.de)



Thank you for running the NAT Analyzer. Please fill out the following form in order to help us to better understand the different implementations of NAT.

Your test ID is: **9715ee919b3a1b6fa6b73eacc3b9c5de**
permanent link for your results

Your router brand	AVM (Fritzbox)	
Your model	7270	(optional), e.g. WRT 54GL
Your firmware	freetz	(optional), e.g. DD-WRT v. 1.0
Your Internet Service Provider	M-Net	(optional), e.g. Comcast, Telekom, Alice
Your connection	DSL 16000	(optional), e.g. Cable, DSL...

Submit results

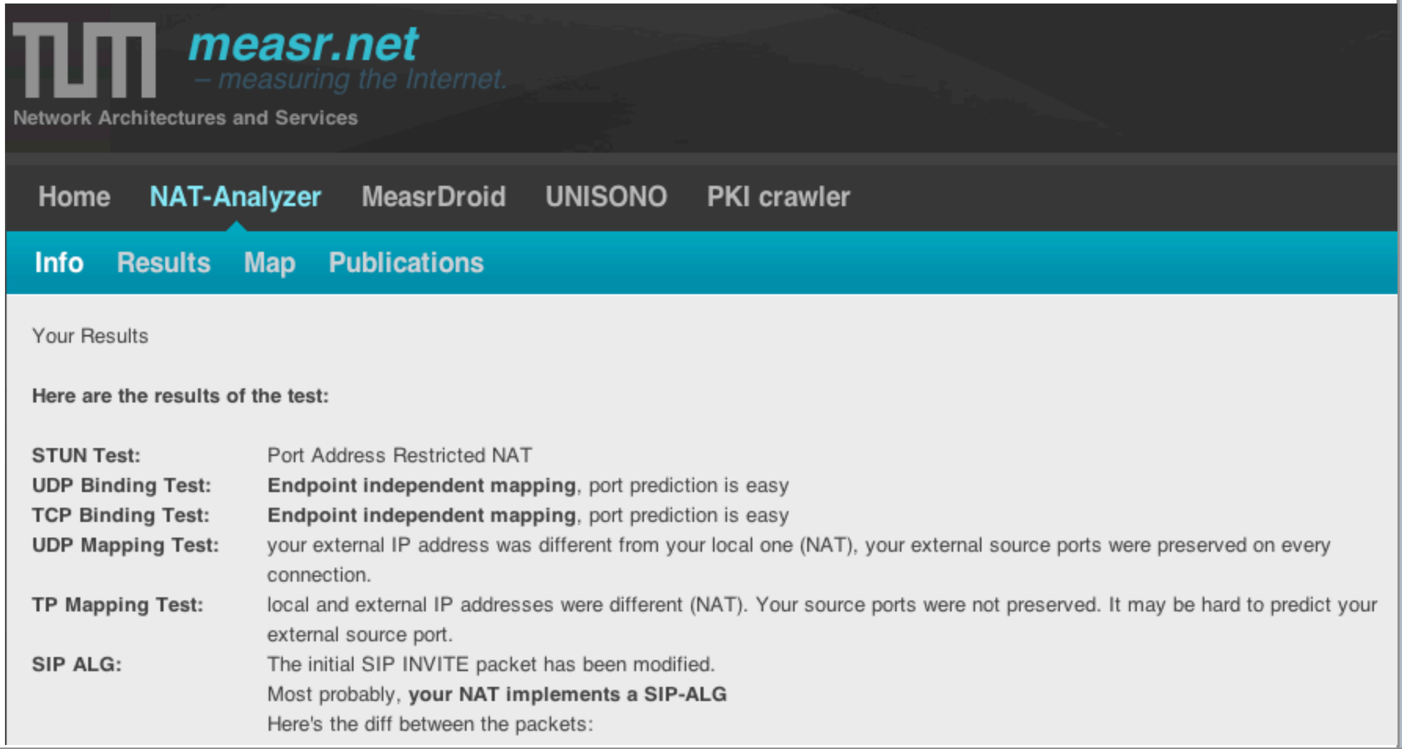
```
running test 8/8: UDP Timeout Tests
testing UDP timeouts, this may take some time...

testing 1 seconds...successful
testing 2 seconds...successful
testing 3 seconds...successful
testing 4 seconds...successful
testing 5 seconds...
```

NAT Analyzer

- Connectivity tests with a server at TUM
  - NAT Type
  - Mapping strategy
  - Binding Strategy
  - Hole Punching behavior using different techniques
  - Timeouts
  - ALGs

- Example Result



Your Results

Here are the results of the test:

STUN Test: Port Address Restricted NAT

UDP Binding Test: **Endpoint independent mapping**, port prediction is easy

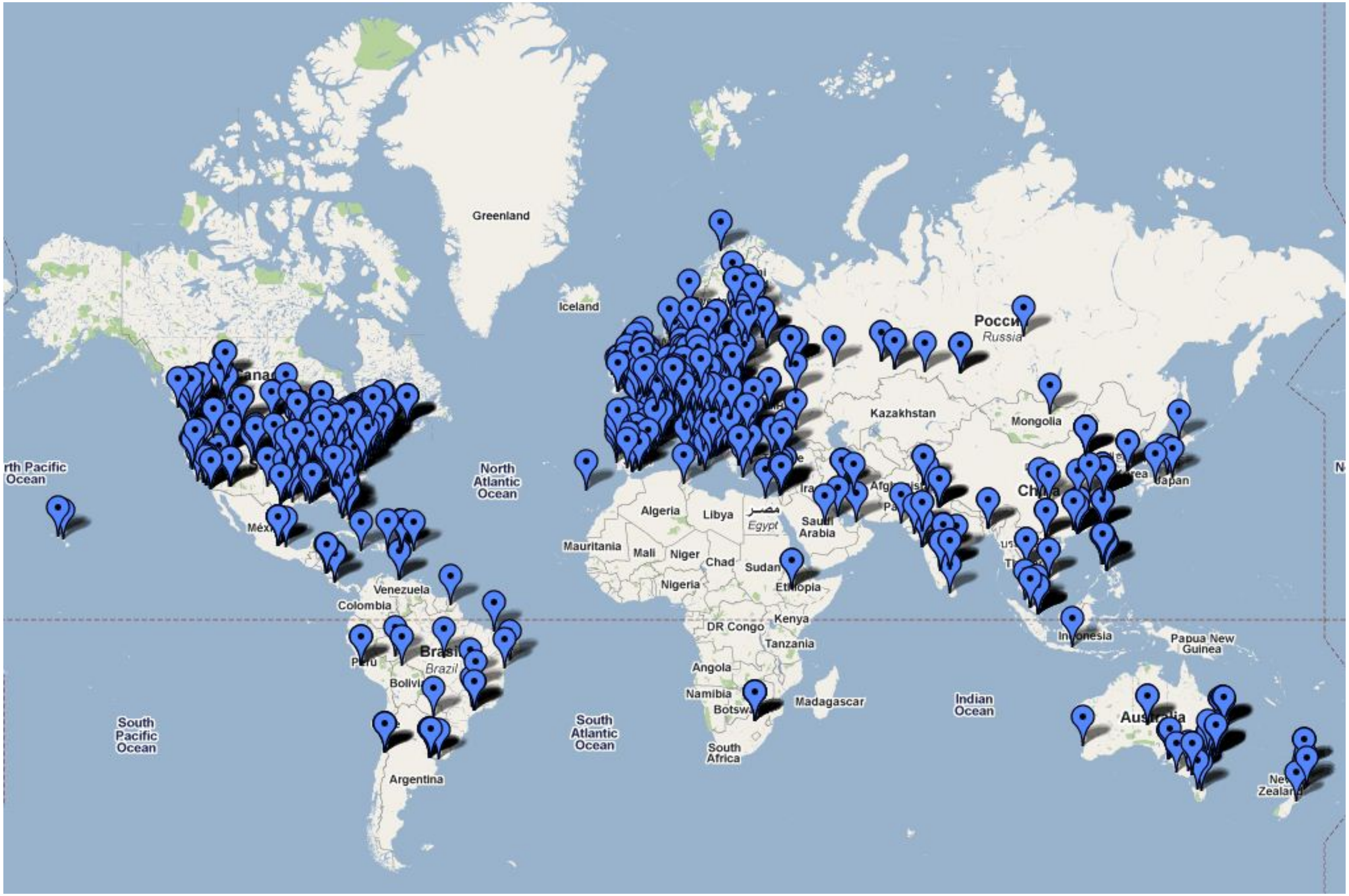
TCP Binding Test: **Endpoint independent mapping**, port prediction is easy

UDP Mapping Test: your external IP address was different from your local one (NAT), your external source ports were preserved on every connection.

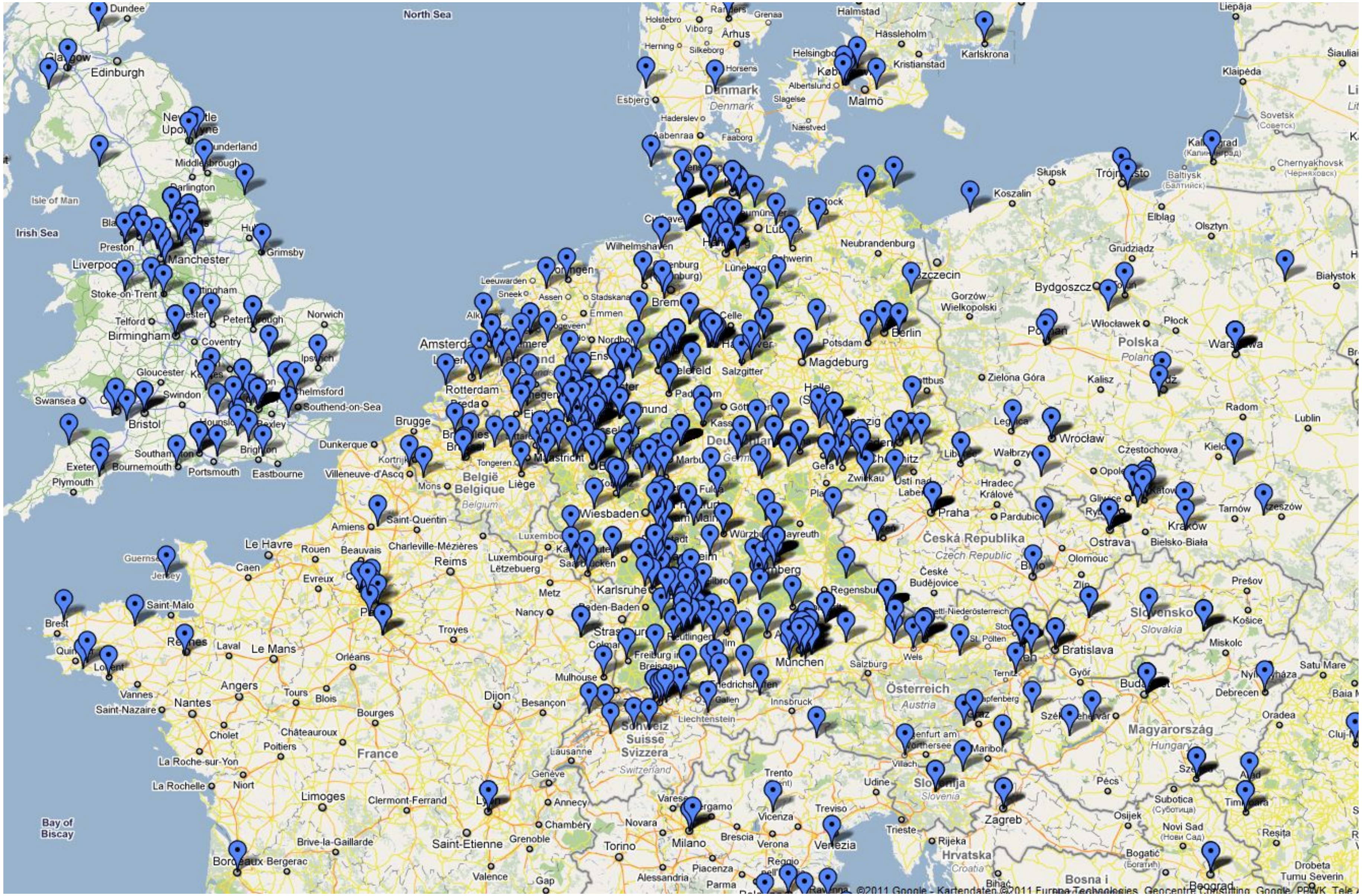
TP Mapping Test: local and external IP addresses were different (NAT). Your source ports were not preserved. It may be hard to predict your external source port.

SIP ALG: The initial SIP INVITE packet has been modified.
Most probably, **your NAT implements a SIP-ALG**
Here's the diff between the packets:

NAT Tester – Results (World)



NAT Tester – Results (Central Europe)



NAT Tester – Results (Providers)

Deutsche Telekom	186
Alice	49
Comcast (US)	47
Arcor	40
Freenet	40
SBS (US)	34
Kabel Deutschland	25
Virgin Media (GB)	23
China Telecom (CN)	20
Road Runner (CA)	18

NAT Tester – Results (Findings)

- Ranking NAT Router

Others	30%
Linksys	16%
Netgear	10%
AVM	7 %
D-Link	7%
Dt. Telekom	6%

- Symmetric „NATs“
  - China
  - Iran
  - Malaysia
  - Israel

Success Rates for existing traversal solutions

❑ UPnP	31 %
❑ Hole Punching	
▪ UDP	80%
▪ TCP low TTL	42%
▪ TCP high TTL	35%
❑ Relay	100%

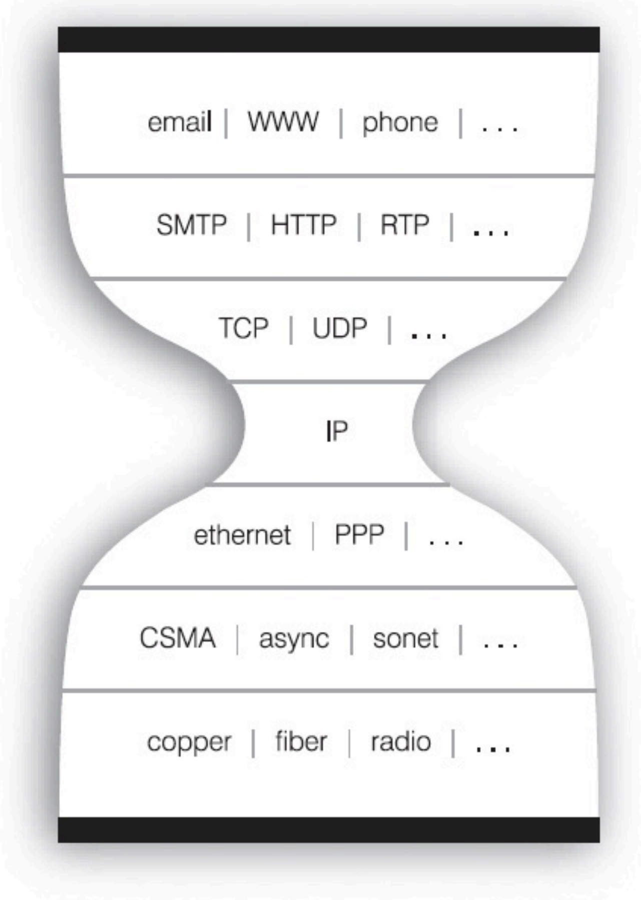
- Propabilities for a direct connection
  - UDP Traversal: 85 %
  - TCP Traversal: 82 %
  - TCP inclusive tunneling: 95 %

The problem is becoming even worse

- **More and more devices connect to the Internet**
  - PCs
  - Cell phones
  - Internet radios
  - TVs
  - Home appliances
  - Future: sensors, cars...

- With NAT, every NAT router needs an IPv4 address

- → ISPs run out of global IPv4 addresses


email | WWW | phone | . . .
SMTP | HTTP | RTP | . . .
TCP | UDP | . . .
IP
ethernet | PPP | . . .
CSMA | async | sonet | . . .
copper | fiber | radio | . . .

Large Scale NAT (LSN)

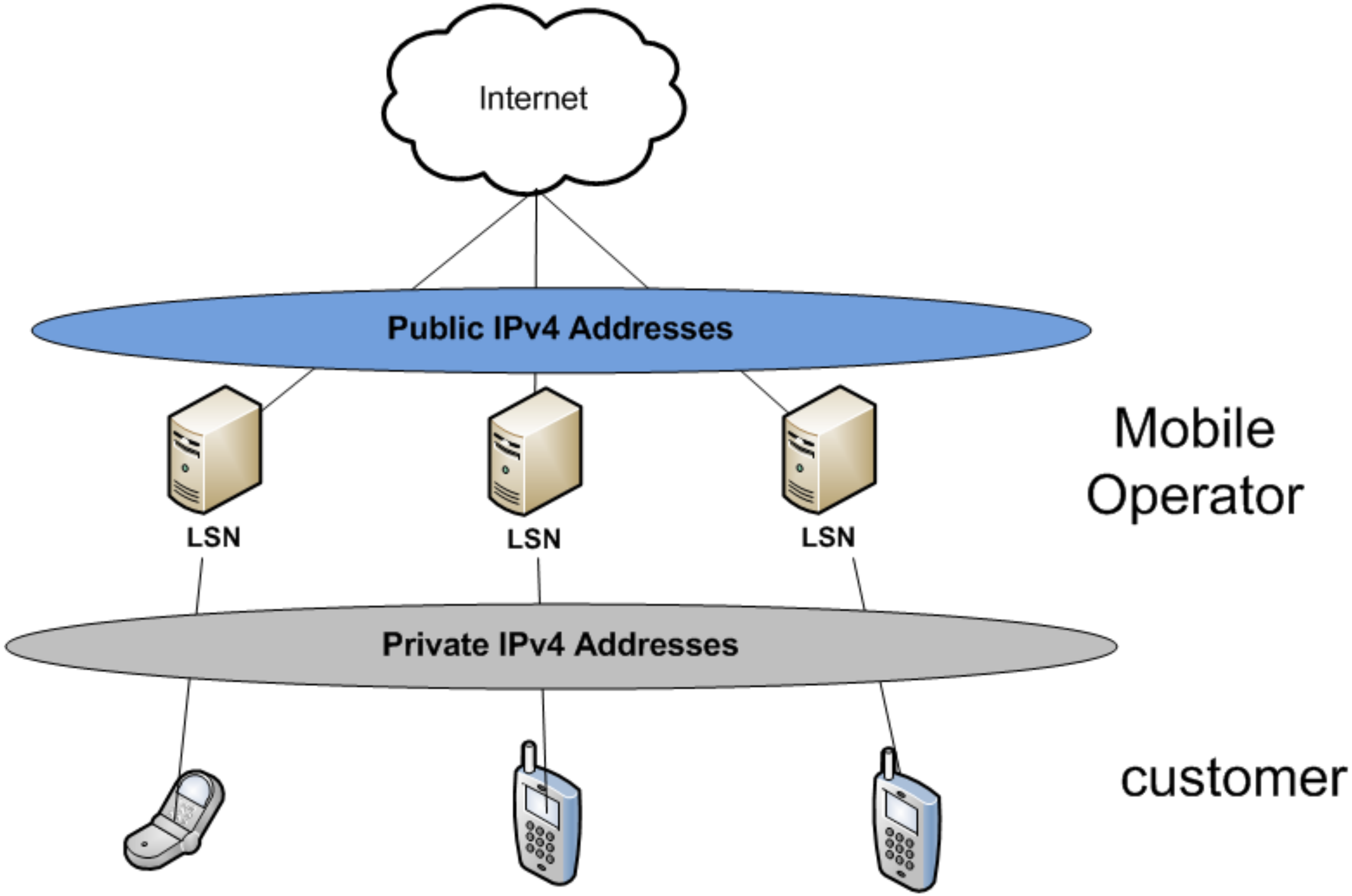
- Facts
  - ISPs run out of global IPv4 addresses
  - Many hosts are IPv4 only
  - Not all content in the web is (and will be) accessible via IPv6
    - infact: < 5% of the Top 100 Websites (09/2011)

- Challenges for ISPs
  - access provisioning for new customers
  - allow customers to use their IPv4 only devices/CPEs
  - provide access to IPv4 content

- Approach: move public IPv4 addresses from customer to provider

- Large Scale NAT (LSN) / Carrier Grade NAT (CGN)
  at provider for translating addresses

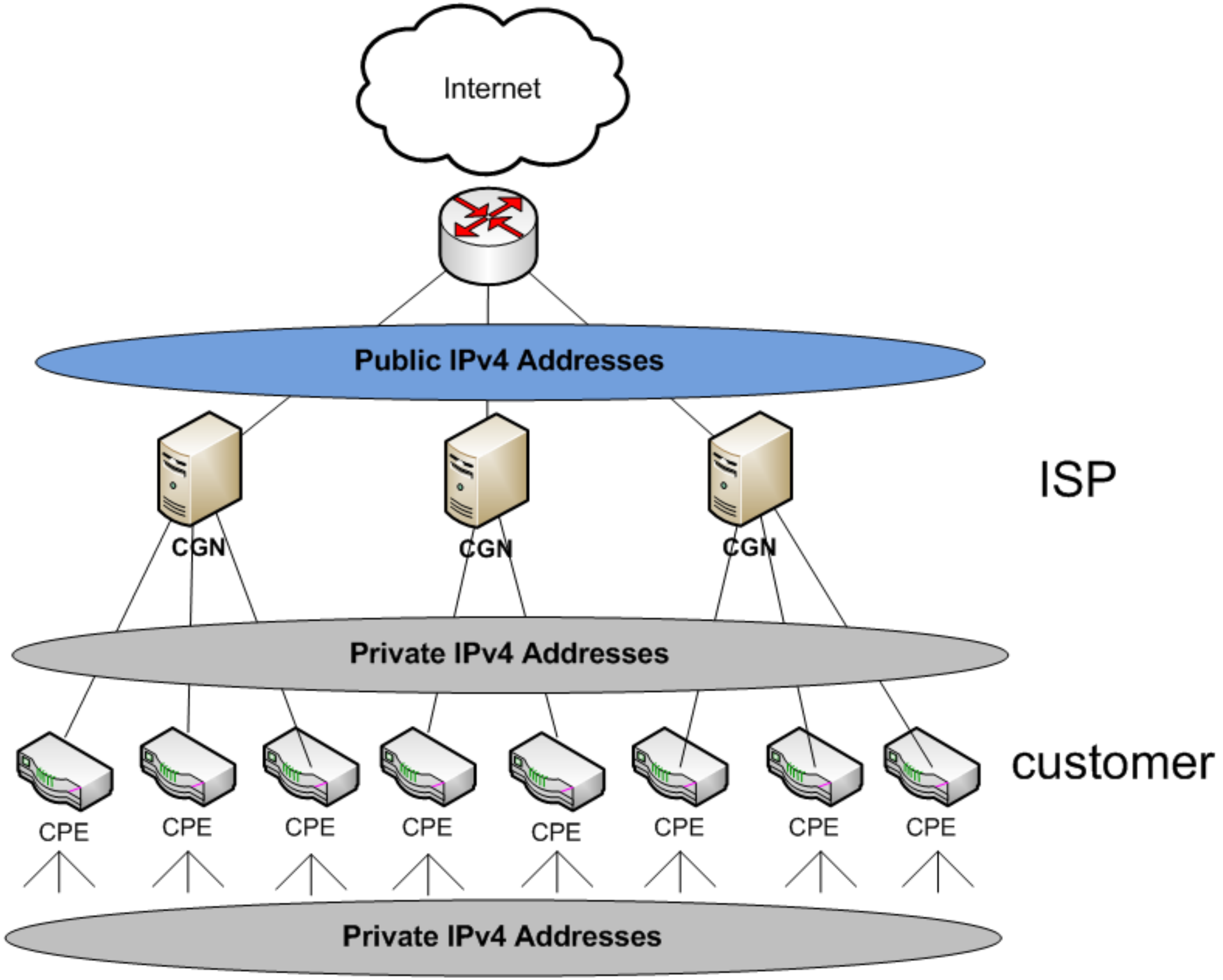
Large Scale NAT already common today



NAT Analyzer – Results (Mobile Operators)

- Germany
  - T-Mobile, Germany
  - Vodafone, Germany
  - O2 Germany
  - E-Plus, Germany
- Europe
  - Hutchison 3G, Ireland
  - Vodafone, Spain
  - Panafone (Vodafone) Greece
  - Eurotel, Czech
  - Tele2 SWIPnet, Sweden
  - Hutchison Drei, Austria
- World
  - Cingular, USA
  - Kyivstar GSM, Ukraine

NAT 444


Internet
Public IPv4 Addresses
CGN
CGN
CGN
ISP
Private IPv4 Addresses
CPE
CPE
CPE
CPE
CPE
CPE
CPE
CPE
customer
Private IPv4 Addresses

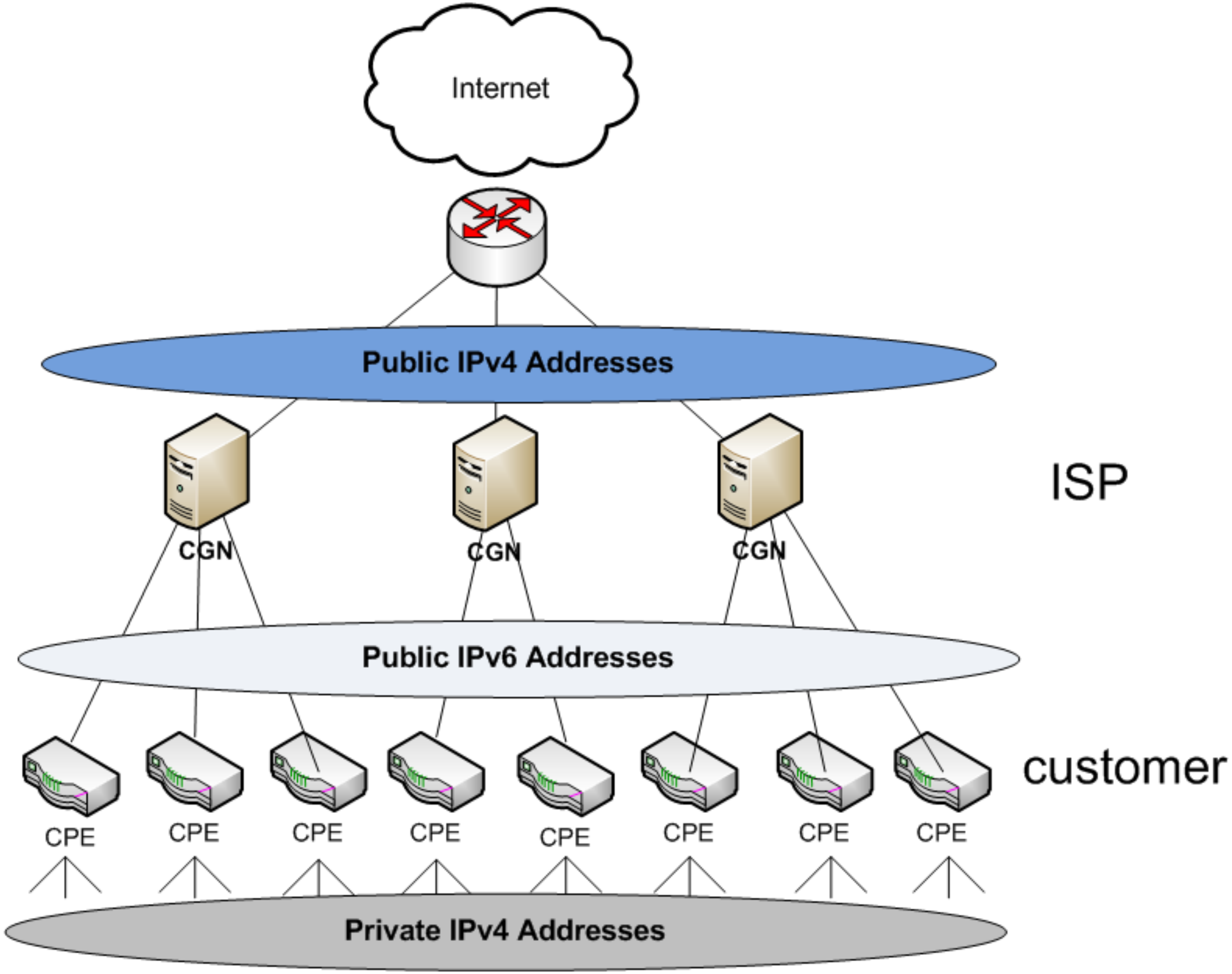
NAT 444

- Easiest way to support new customers
  - immediately available
  - no changes at CPEs (Customer Premises Equipment)

- Problems:
  - Address overlap -> same private IP address on both sides
  - Hairpinning necessary: firewalls on CPE may block incoming packets with a private source address

- Solutions
  - declare a range of public IP addresses as „ISP shared“ and reuse it as addresses between CGN and CPE
  - NAT 464: IPv6 between CPE and CGN
    - Problem: CPEs must implement NAT64

NAT 464


Internet
Public IPv4 Addresses
CGN
CGN
CGN
ISP
Public IPv6 Addresses
customer
CPE
CPE
CPE
CPE
CPE
CPE
CPE
CPE
Private IPv4 Addresses

Dual Stack lite

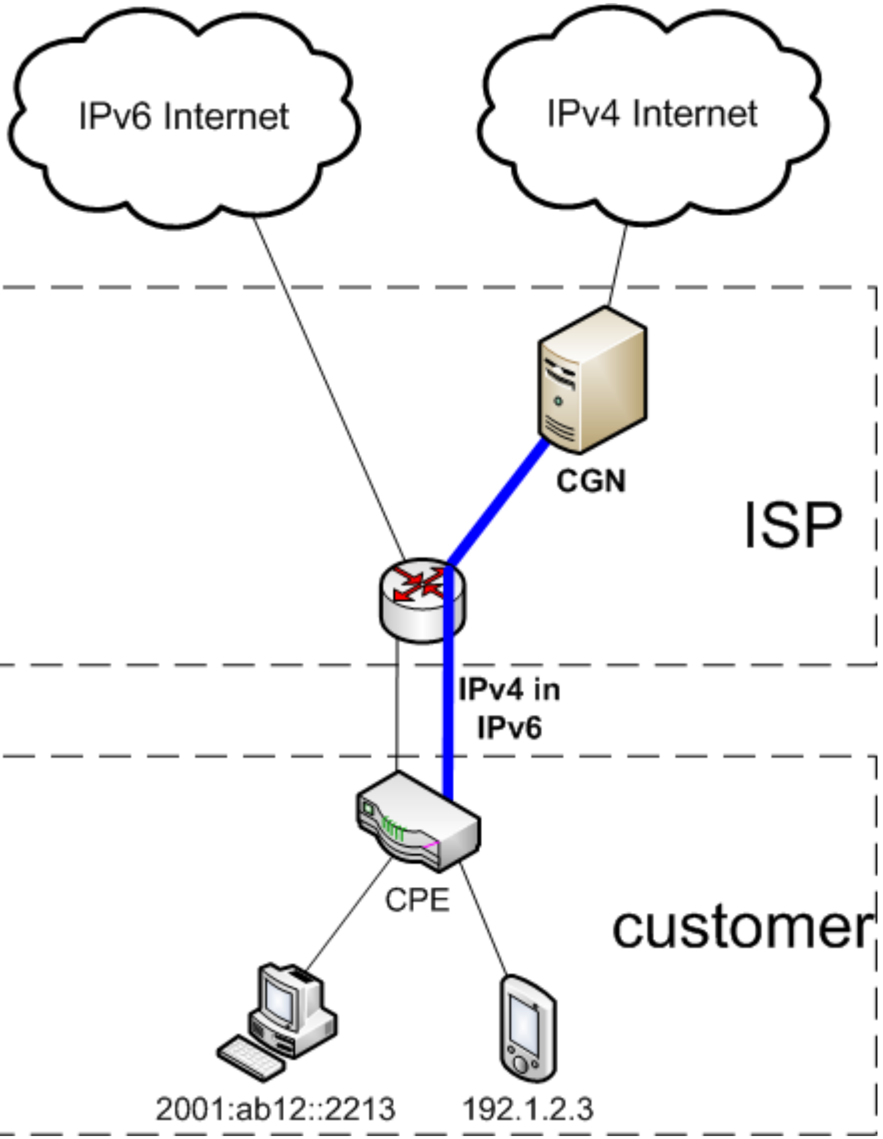
- Mixture of NAT 444 and NAT 464

- IPv4 in IPv6 tunnel between CPE and ISP
  - No need for protocol translation
  - No cascaded NATs

- Allows to deploy IPv6 in the ISP network while still supporting IPv4 content and IPv4 customers
  - As IPv6 devices become available they can be directly connected without the need for a tunnel

- Mainly pushed by Comcast (in IETF)

Dual Stack Lite



LSN - Challenges

- Mainly: how to manage resources
  - Ports (number of ports, allocation limit (time)
  - Addresses
  - Bandwidth
  - legal issues (logging)

- NAT behavior
  - desired: first packet reserves a bin for the customer -> less logging effort
  - IP address pooling: random vs. paired (same ext IP for internal host)
    - Pairing between external and internal IP address

- Impacts of double NAT for users
  - Blacklisting as done today (based on IPs) will be a problem
  - No control of ISP NATs

- Possible Approaches
  - Small static pool of ports in control of customer
  - Needs configuration/reservation/security protocols

Network Address Translation today

- Thought as a temporary solution

- Home Users
  - to share one public IP address
  - to hide the network topology and to provide some sort of security

- ISPs
  - for connecting more and more customers
  - for the planned transition to IPv6

- Mobile operators
  - to provide connectivity to a large number of customers
  - „security“

- Enterprises
  - to hide their topology
  - to be address independent

NAT Conclusion

- NAT helps against the shortage of IPv4 addresses

- NAT works as long as the server part is in the public internet

- P2P communication across NAT is difficult

- 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

- New challenges with the transition to IPv6

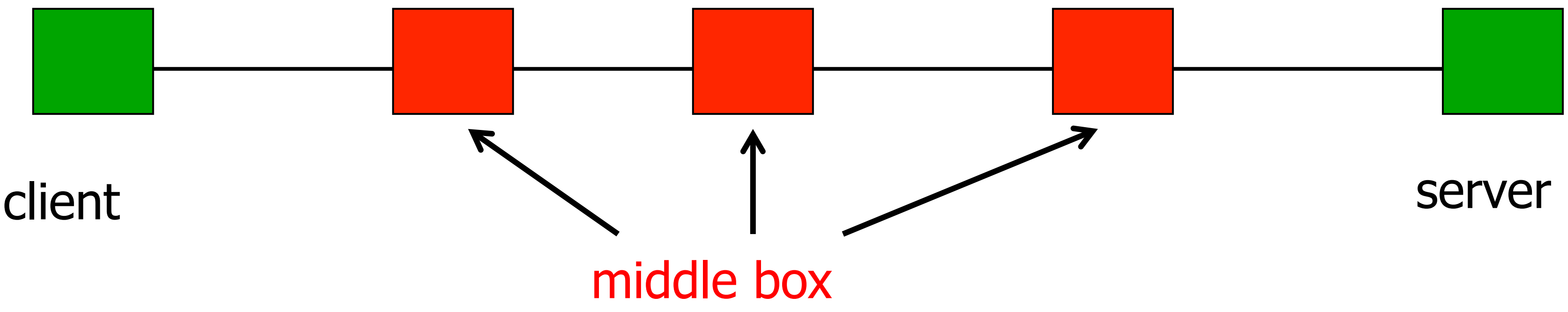
Middleboxes

RFC 3234 - Middleboxes

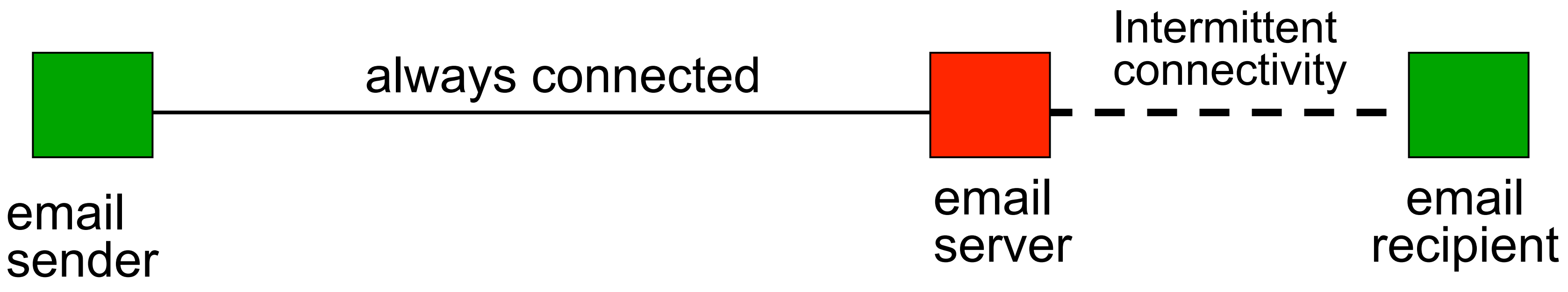
- The phrase "middlebox" was coined by Lixia Zhang as a graphic description of a recent phenomenon in the Internet.

Lixia Zhang,
UCLA

What are *middle* boxes?

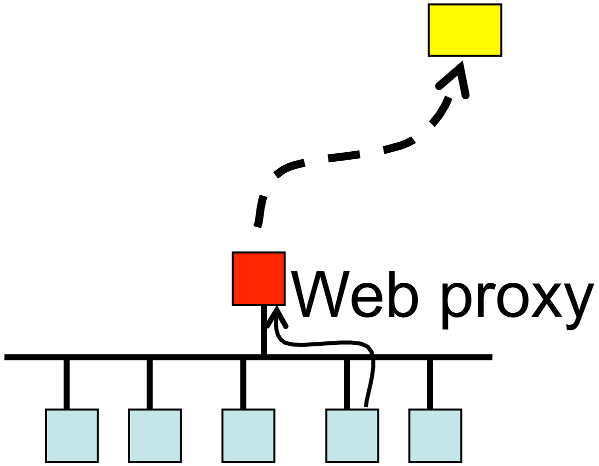


- data is no longer delivered between the two end boxes by *direct* IP path
- The first middleman: email server

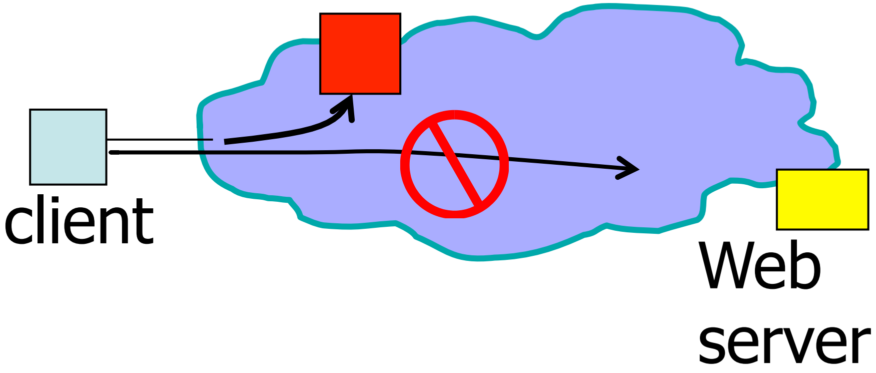


Middleboxes

- Web proxies



- "transparent" Web caches



Packet hijacking!("for your benefit")

Middleboxes Address Practical Challenges

- IP address depletion
  - Allowing multiple hosts to share a single address
- Host mobility
  - Relaying traffic to a host in motion
- Security concerns
  - Discarding suspicious or unwanted packets
  - Detecting suspicious traffic
- Performance concerns
  - Controlling how link bandwidth is allocated
  - Storing popular content near the clients

Layer Violation Boxes

- Peek into application layer headers…
- Send certain packets to a different server…
- Proxy certain request without being asked...
- Rewrite requests …

- Result: unpredictable behaviour, inexplicable failures
- c.f. RFC 3234

RFC 3234 - Middleboxes: Taxonomy and Issues

- A middlebox is **defined** as any intermediary device performing functions other than standard functions of an IP router on the datagram path between a source host and destination host.
- Standard IP router: transparent to IP packets
- End-to-end principle: asserts that some functions (such as security and reliability) can only be implemented completely and correctly end-to-end.
- Note: providing an incomplete version of such functions in the network can sometimes be a performance enhancement, but not a substitute for the end-to-end implementation of the function.

Properties

- Middleboxes may
  - Drop, insert or modify packets.
  - Terminate one IP packet flow and originate another.
  - Transform or divert an IP packet flow in some way.
- Middleboxes are never the ultimate end-system of an application session

- Examples
  - Network Address Translators
  - Firewalls
  - Traffic Shapers
  - Load Balancers

Concerns

- New middleboxes challenge **old protocols**. Protocols designed without consideration of middleboxes may fail, predictably or unpredictably, in the presence of middleboxes.
- Middleboxes introduce **new failure modes**;
  rerouting of IP packets around crashed routers is no longer the only case to consider. The fate of sessions involving
  crashed middleboxes must also be considered.
- **Configuration** is no longer limited to the two ends of a session; middleboxes may also require configuration and management.
- **Diagnosis** of failures and misconfigurations is more complex.

Middlebox Classification

1. Protocol layer (IP layer, transport layer, app layer, or mixture?)
2. Explicit (design feature of the protocol)
   or implicit (add-on not by the protocol design)
3. Single hop vs. multi-hop (can there be several middleboxes?)
4. In-line (executed on the datapath) vs. call-out (ancillary box)
5. Functional (required by application session) vs. optimising
6. Routing vs. processing (change packets or create side-effect)
7. Soft state (session may continue while middlebox rebuilds state)
   vs. hard state
8. Failover (may a session be redirected to alternative box?)
   vs. restart

Specific Middleboxes

- **Packet classifiers**
  - classify packets flowing through them according to policy
  - either select them for special treatment or mark them
  - may alter the sequence of packet flow through subsequent hops, since they control the behaviour of traffic conditioners.
  - {1 multi-layer, 2 implicit, 3 multihop, 4 in-line, 5 optimising, 6 processing, 7 soft, 8 failover or restart}
- **IP Firewalls**
  - Inspects IP and Transport headers
  - configured policies decide which packets are discarded, e.g.:
    - Disallows incoming traffic to certain port numbers
    - Disallows traffic to certain subnets
  - Does not alter forwarded packets
  - Not visible as protocol end-point

Specific Middleboxes

- **Proxies**
  - An intermediary program that acts as a client and server
  - Makes requests on behalf of a client and then serves the result

- **Application Firewalls**
  - act as a protocol end point and relay (e.g., Web proxy); may

  (1) implement a "safe" subset of the protocol,

  (2) perform extensive protocol validity checks,

  (3) use implementation methodology for preventing bugs,

  (4) run in an insulated, "safe" environment, or

  (5) use combination of above

Middlebox Types according to RFC 3234

1. **NAT,**
2. **NAT-PT**,
3. **SOCKS gateway,**
4. **IP tunnel endpoints,**
5. **packet classifiers, markers, schedulers,**
6. **transport relay,**
7. TCP performance enhancing proxies,
8. **load balancers that divert/munge packets,**
9. **IP firewalls,**
10. **application firewalls,**
11. application-level gateways

bold - act per packet
- do not modify application payload
- do not insert additional packets

12. gatekeepers / session control boxes,
13. transcoders,
14. (Web or SIP) proxies,
15. (Web) caches,
16. modified DNS servers,
17. content and applications distribution boxes,
18. load balancers that divert/munge URLs,
19. application-level interceptors,
20. application-level multicast,
21. **involuntary packet redirection,**
22. **anonymizers.**

Assessment of Middlebox Classification

1. Protocol layer (IP layer, transport layer, app layer, or mixture?)
2. Explicit (design feature of the protocol) or implicit
3. Single hop vs. multi-hop (can there be several middleboxes?)
4. In-line (executed on the datapath) vs. call-out (ancillary box)
5. Functional (required by application session) vs. optimising
6. Routing vs. processing (change packets or create side-effect)
7. Soft state (session may continue while rebuilding state) vs. hard state
8. Failover (may a session be redirected to alternative box?) vs. restart

Of 22 classes of Middleboxes:

- ❑ 17 are application or multi-layer
- ❑ 16 are implicit
- ❑ 17 are multi-hop
- ❑ 21 are in-line; call-out is rare
- ❑ 18 are functional; pure optimisation is rare
- ❑ Routing & processing evenly split
- ❑ 16 have hard state
- ❑ 21 must restart session on failure

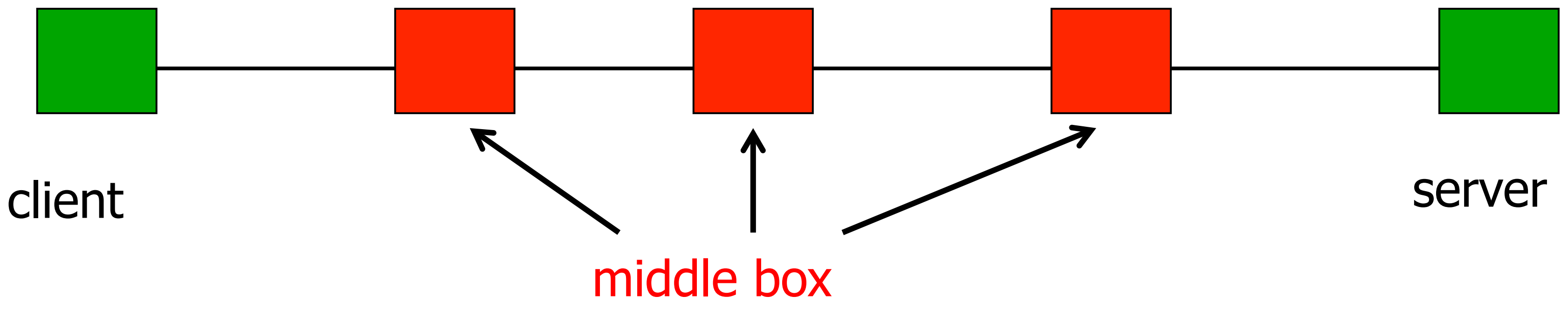
Middleboxes

RFC 3234 - Middleboxes

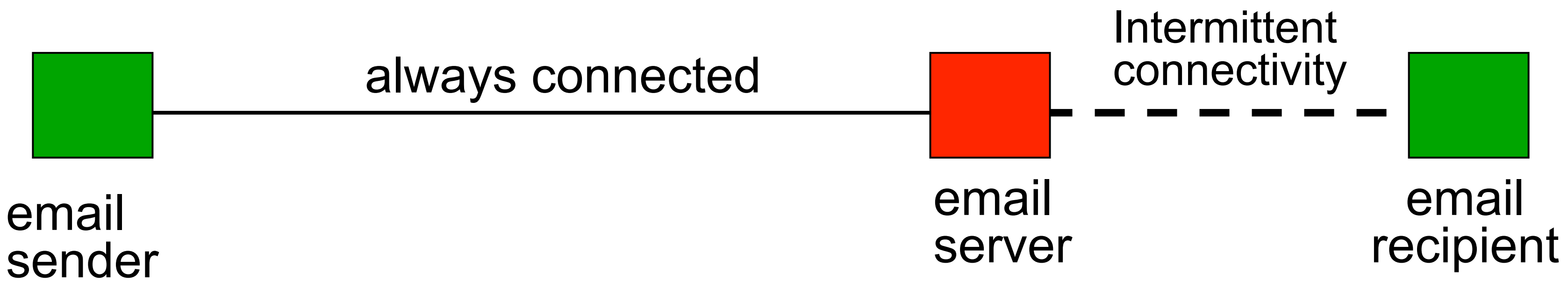
- The phrase "middlebox" was coined by Lixia Zhang as a graphic description of a recent phenomenon in the Internet.

Lixia Zhang,
UCLA

What are *middle* boxes?

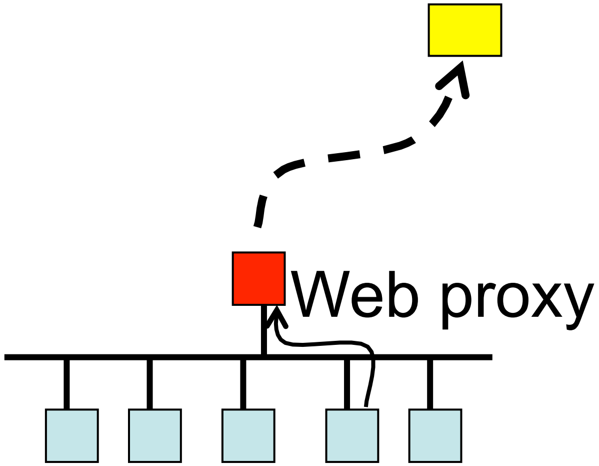


- data is no longer delivered between the two end boxes by *direct* IP path
- The first middleman: email server

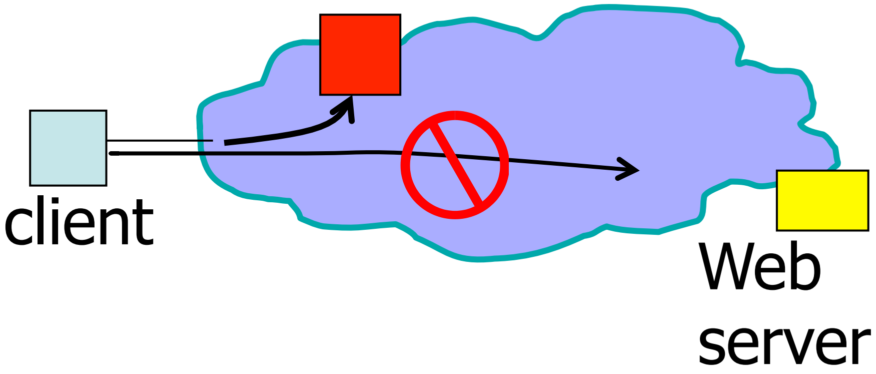


Middleboxes

- Web proxies



- "transparent" Web caches



Packet hijacking!("for your benefit")

Middleboxes Address Practical Challenges

- IP address depletion
  - Allowing multiple hosts to share a single address
- Host mobility
  - Relaying traffic to a host in motion
- Security concerns
  - Discarding suspicious or unwanted packets
  - Detecting suspicious traffic
- Performance concerns
  - Controlling how link bandwidth is allocated
  - Storing popular content near the clients

Layer Violation Boxes

- Peek into application layer headers
- Send certain packets to a different server
- Proxy certain request without being asked
- Rewrite requests

- Result: unpredictable behaviour, inexplicable failures
- c.f. RFC 3234

RFC 3234 - Middleboxes: Taxonomy and Issues

- A middlebox is **defined** as any intermediary device performing functions other than standard functions of an IP router on the datagram path between a source host and destination host.

- Standard IP router: transparent to IP packets

- End-to-end principle: asserts that some functions (such as security and reliability) can only be implemented completely and correctly end-to-end.
- Note: providing an incomplete version of such functions in the network can sometimes be a performance enhancement, but not a substitute for the end-to-end implementation of the function.

Properties

- Middleboxes may
  - Drop, insert or modify packets.
  - Terminate one IP packet flow and originate another.
  - Transform or divert an IP packet flow in some way.
- Middleboxes are never the ultimate end-system of an application session

- Examples
  - Network Address Translators
  - Firewalls
  - Traffic Shapers
  - Load Balancers

Concerns

- New middleboxes challenge **old protocols**. Protocols designed without consideration of middleboxes may fail, predictably or unpredictably, in the presence of middleboxes.

- Middleboxes introduce **new failure modes**;
  rerouting of IP packets around crashed routers is no longer the only case to consider. The fate of sessions involving *crashed middleboxes* must also be considered.

- **Configuration** is no longer limited to the two ends of a session; middleboxes may also require configuration and management.

- **Diagnosis** of failures and misconfigurations is more complex.

RFC 3234: Middlebox Classification

1. Protocol layer (IP layer, transport layer, app layer, or mixture?)
2. Explicit (design feature of the protocol)
   or implicit (add-on not by the protocol design)
3. Single hop vs. multi-hop (can there be several middleboxes?)
4. In-line (executed on the datapath) vs. call-out (ancillary box)
5. Functional (required by application session) vs. optimising
6. Routing vs. processing (change **path** or create side-effect)
7. Soft state (session may continue while middlebox rebuilds state)
   vs. hard state
8. Failover (may a session be redirected to alternative box?)
   vs. restart

Specific Middleboxes

- **Packet classifiers**
  - classify packets flowing through them according to policy
  - either select them for special treatment or mark them
  - may alter the sequence of packet flow through subsequent hops, since they control the behaviour of traffic conditioners.
  - {1 multi-layer, 2 implicit, 3 multihop, 4 in-line, 5 optimising, 6 processing, 7 soft, 8 failover or restart}
- **IP Firewalls**
  - Inspects IP and Transport headers
  - configured policies decide which packets are discarded, e.g.:
    - Disallows incoming traffic to certain port numbers
    - Disallows traffic to certain subnets
  - Does not alter forwarded packets
  - Not visible as protocol end-point

Specific Middleboxes

- **Proxies**
  - Intermediary program that acts as client and server
  - Make requests on behalf of client and then serves result

- **Application Firewalls**
  - Act as a protocol end point and relay (e.g., Web proxy)
  - May

    (1) implement a "safe" subset of the protocol,

    (2) perform extensive protocol validity checks,

    (3) use implementation methodology for preventing bugs,

    (4) run in an insulated, "safe" environment, or

    (5) use combination of above

Middlebox Types according to RFC 3234

1. **NAT**
2. **NAT-PT**
3. **SOCKS gateway**
4. **IP tunnel endpoints**
5. **packet classifiers, markers, schedulers**
6. **transport relay**
7. TCP performance enhancing proxies
8. **load balancers that divert/munge packets**
9. **IP firewalls**
10. **application firewalls**
11. application-level gateways

bold - act per packet
- do not modify application payload
- do not insert additional packets

12. gatekeepers / session control boxes
13. transcoders
14. (Web or SIP) proxies
15. (Web) caches
16. modified DNS servers
17. content and applications distribution boxes
18. load balancers that divert/munge URLs
19. application-level interceptors
20. application-level multicast
21. **involuntary packet redirection**
22. **anonymizers**

Assessment of Middlebox Classification

1. Protocol layer (IP layer, transport layer, app layer, or mixture?)
2. Explicit (design feature of the protocol) or implicit
3. Single hop vs. multi-hop (can there be several middleboxes?)
4. In-line (executed on the datapath) vs. call-out (ancillary box)
5. Functional (required by application session) vs. optimising
6. Routing vs. processing (change packets or create side-effect)
7. Soft state (session may continue while rebuilding state) vs. hard state
8. Failover (may a session be redirected to alternative box?) vs. restart

Of 22 classes of Middleboxes:

- ❑ 17 are application or multi-layer
- ❑ 16 are implicit
- ❑ 17 are multi-hop
- ❑ 21 are in-line; call-out is rare
- ❑ 18 are functional; pure optimisation is rare
- ❑ Routing & processing evenly split
- ❑ 16 have hard state
- ❑ 21 must restart session on failure

Assessment

- Although the rise of middleboxes has negative impact on the end to end principle at the packet level, it is still a desirable principle of applications protocol design.
- Future application protocols should be designed in recognition of the likely presence of middleboxes (e.g. network address translation, packet diversion, and packet level firewalls)
- Approaches for failure handling needed
  - soft state mechanisms
  - rapid failover or restart mechanisms
- Common features available to many applications needed
  - Middlebox discovery and monitoring
  - Middlebox configuration and control
  - Routing preferences
  - Failover and restart handling
  - Security

Routing

Part I

Monday, 2011-12-05

Short note on pronunciation of the word “routing”

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

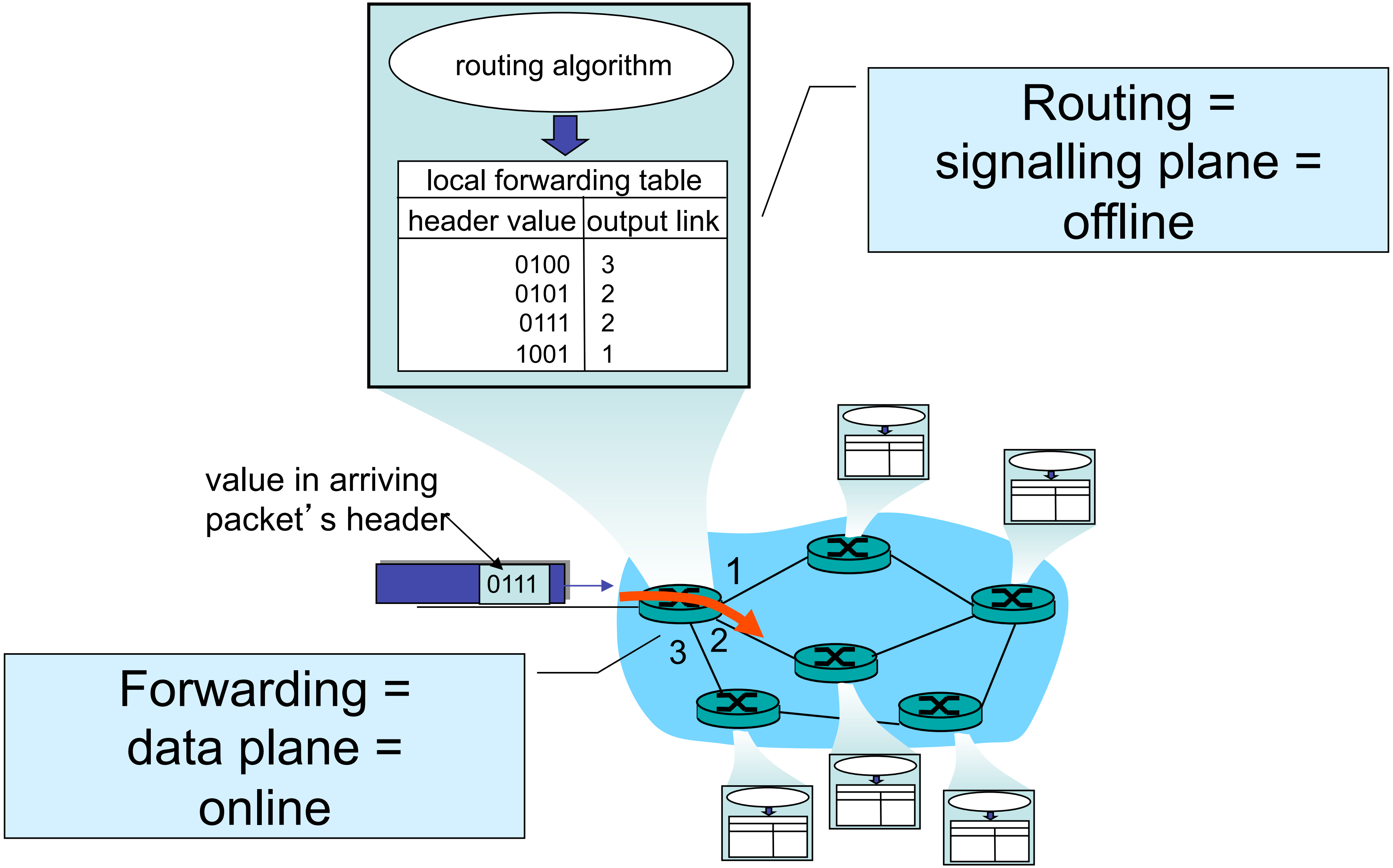
Chapter outline: Routing

- Routing and forwarding
- Routing algorithms recapitulated
  - Link state
  - Distance Vector
  - Path Vector
- Intradomain routing protocols
  - RIP
  - OSPF
- Interdomain routing
  - Hierarchical routing
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering
- Routing security
- Multicast routing

Routing ≠ Forwarding

- Routing:
  - The process of determining the best path for a specific type of packets (usually: all packets with the same destination) through the network
  - Performed jointly by the routers of a network by exchanging many messages
  - Analogy: Read street map, plan journey
- Forwarding:
  - The process where a router relays a packet to a neighbouring router. Selection of the neighbouring router depends on the previous routing protocol calculations
  - Performed by one router on one packet
  - Analogy: Read a street sign and determine if we should take the next exit
- In practice, this distinction is often ignored
  - “If router A routes packet X, then …”
  - Actually, it doesn‘t – it *forwards* X.

Signalling plane and data plane



local forwarding table

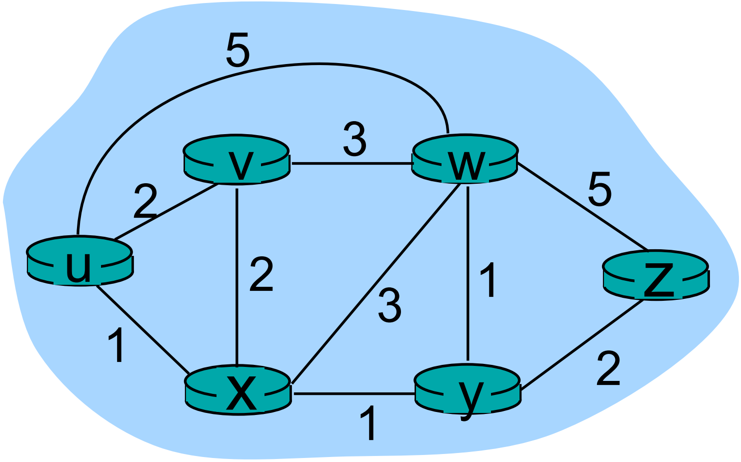
header value	output link
0100	3
0101	2
0111	2
1001	1

Routing =
signalling plane =
offline

value in arriving
packet's header

Forwarding =
data plane =
online

Graph abstraction



Graph: G = (N,E)

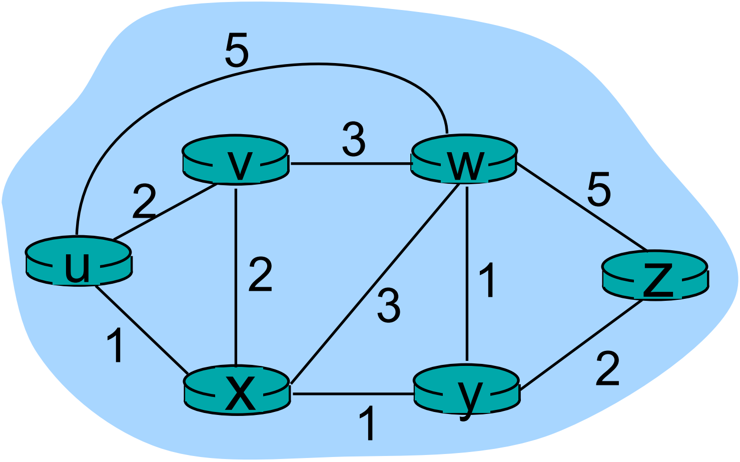
N = nodes = set of routers = { u, v, w, x, y, z }

E = edges = 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 and link cost info
- *link state* algorithms (L-S)

Decentralized:

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

Static or dynamic?

Static:

- Routes change slowly over time

Dynamic:

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

A broader routing classification

- Type of algorithm: Link State, Distance Vector, Path Vector, …
- Scope:
  - Intradomain
  - Interdomain
  - Special purpose (e.g., sensor network)
- Type of traffic: Unicast vs. multicast
- Type of reaction: “Static” vs. Dynamic/adaptive
  - Warning: “Dynamic routing” is a fuzzy term:
    a) Dynamic ≔ reacts to topology changes (state of the art)
    b) Dynamic ≔ reacts to traffic changes (even better, but most protocols don’t do that!)
- Trigger type:
  - Permanent routing (standard)
  - On-demand routing: only start routing algorithm if there is traffic to be forwarded (e.g., some wireless ad-hoc networks)

A Link-State Routing Algorithm

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

Distance Vector Algorithm

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

Distance Vector Algorithm

Bellman-Ford Equation (dynamic programming)

Let

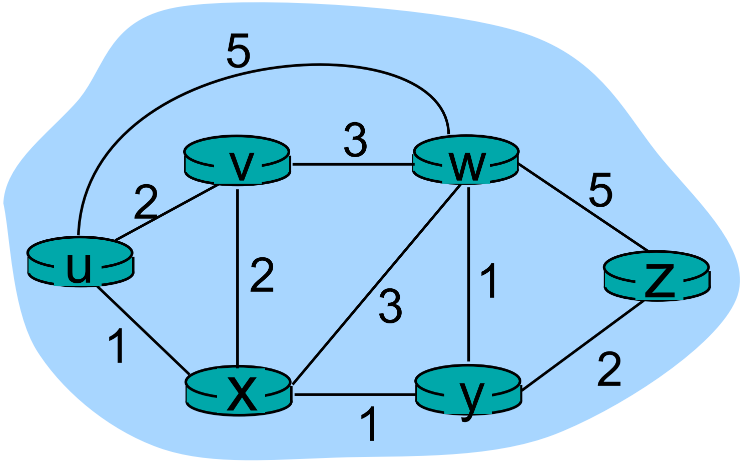
- $c(x,y) :=$ cost of edge from *x* to *y*
- $d_x(y) :=$ cost of least-cost path from *x* to *y*
- Set to ∞ if no path / no edge available

Then

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

where min is taken over all neighbours *v* of *x*

Bellman-Ford example



We can see that

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

B-F equation says:

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

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

Distance Vector Algorithm

- Define $D_x(y)$ ≔ estimate of least cost from x to y
- Node x knows cost to each neighbour v: $c(x,v)$
- Node x maintains distance vector $\overrightarrow{D_x} \coloneqq [\, D_x(y): y \in N \,]$ (N ≔ set of nodes)
- Node x also maintains copies of its neighbours' distance vectors
  - Received via update messages from neighbours
  - For each neighbour v, x knows $\overrightarrow{D_v} = [\, D_v(y): y \in N \,]$

Distance vector algorithm (4)

Basic idea:

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

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

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

Distance Vector Algorithm (5)

Iterative, asynchronous:

Each local iteration caused by:

- ❑ Local link cost change
- ❑ DV update message from neighbour

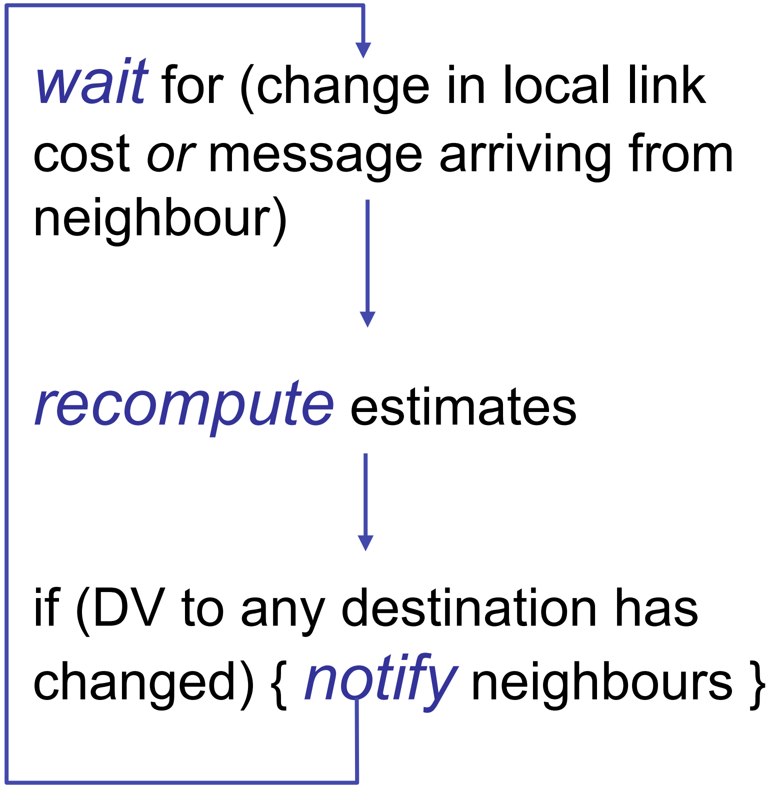
Distributed:

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

Usually some waiting delay between consecutive updates

Each node:

Forever:

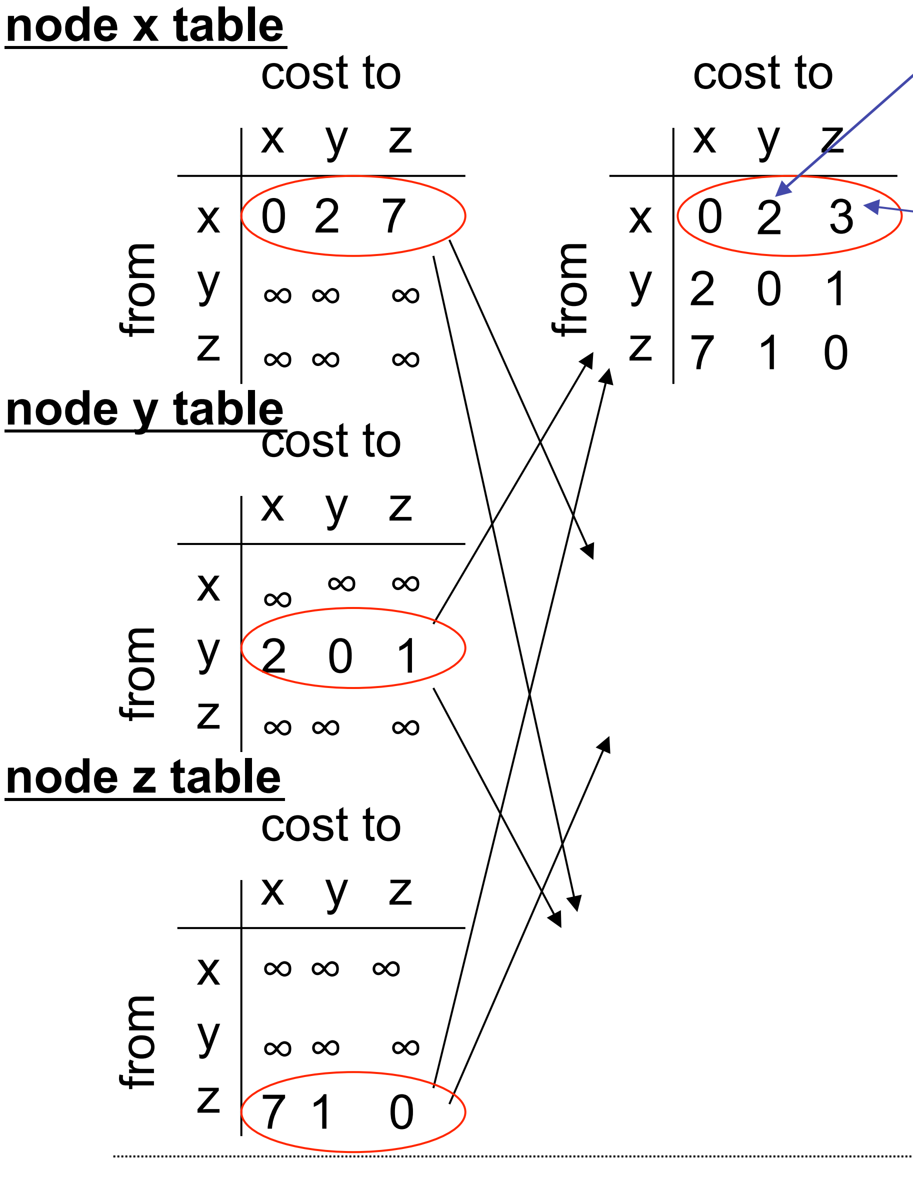


wait for (change in local link cost *or* message arriving from neighbour)

recompute estimates

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

Distance Vector Algorithm (6)



node x table

cost to

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

cost to

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

node y table

cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

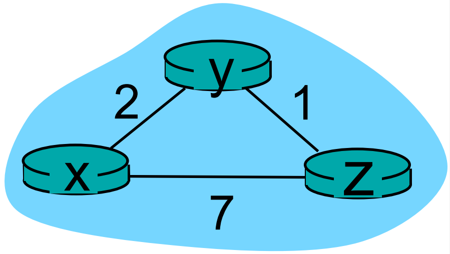
cost to

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

cost to

		x	y	z
from	x	0	2	7
	y	∞	∞	∞
	z	∞	∞	∞

cost to

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

cost to

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

node y table

cost to

		x	y	z
from	x	∞	∞	∞
	y	2	0	1
	z	∞	∞	∞

cost to

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

cost to

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

node z table

cost to

		x	y	z
from	x	∞	∞	∞
	y	∞	∞	∞
	z	7	1	0

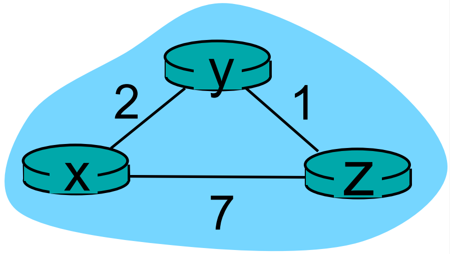
cost to

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

cost to

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

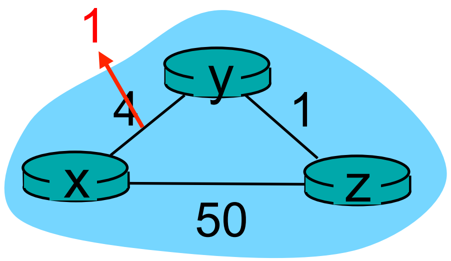
time



Distance Vector: link cost changes (1)

Link cost changes:

- ❑ Node detects local link cost change
- ❑ Updates routing info, recalculates distance vector
- ❑ If DV changes, notify neighbours



"good news travels fast"

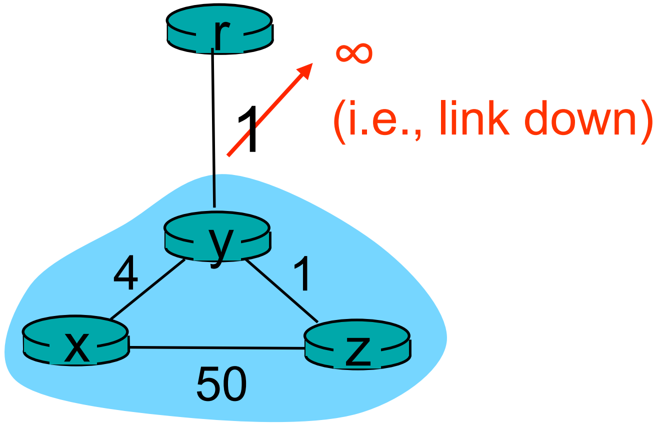
At time t_0, *y* detects the link-cost change, updates its DV, and informs its neighbours.

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 neighbours its new DV.

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

Distance Vector: link cost changes (2)

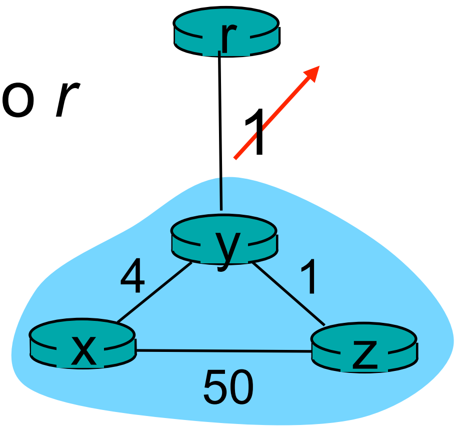
- But: bad news travels slow
- In example: Many iterations before algorithm stabilizes!
  1. Cost increase for *y*→*r*:
     - *y* consults DV,
     - *y* selects "cheaper" route via *z* (cost 2+1 = 3),
     - Sends update to *z* and *x* (cost to *r* now 3 instead of 1)
  2. *z* detects cost increase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



- Symptom: "count to infinity" problem

Distance Vector: Problem Solutions…

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



- Poisoned Reverse:
  - In addition, *actively* advertise a route as unreachable to the neighbour from which the route was learned
- (Warning: Terms often used interchangeably!)

…that only half work

- Mechanisms can be combined

- Both mechanisms can significantly increase number of routing messages

- Often help, but cannot solve all problem instances
  - Think yourselves: Come up with a topology where this does not help
  - Try it – it’s not hard and a good exercise

Comparison of LS and DV algorithms

Message complexity

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

Speed of Convergence

- LS: O(n^2) algorithm requires O(*nE*) messages
  - 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:

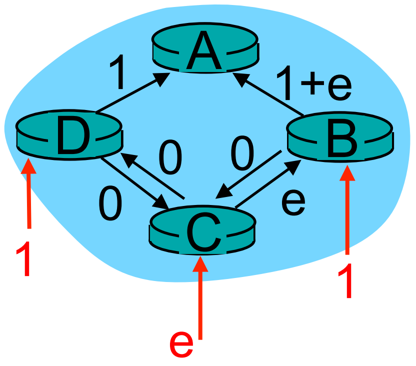
- DV node can advertise incorrect *path* cost
- each node' s table used by others
  - error propagates through network

Path Vector protocols

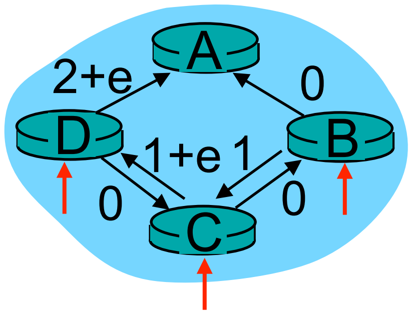
- Problem with D-V protocol:
  Path cost is “anonymous” single number; does not contain any topology information
- Path Vector protocol:
  - For each destination, advertise entire path (=sequence of node identifiers) to neighbours
  - Cost calculation can be done by looking at path
    - E.g., count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

Dynamic (i.e., traffic-adaptive) routing?

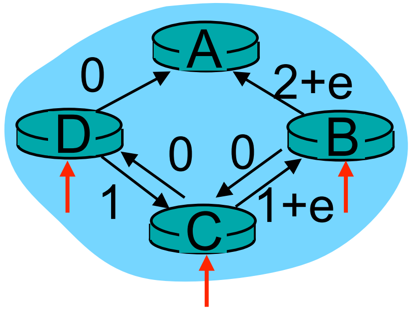
- Dangerous: Oscillations possible!
- e.g., link cost = amount of carried traffic



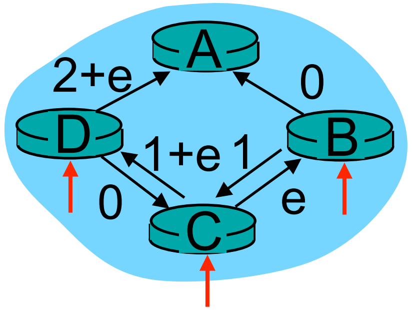
initially



… recompute routing



… recompute



… recompute

- Why is this a bad thing?
  - Possibly sub-optimal choice of paths (as in example above)
  - Additional routing protocol traffic in network
  - Increased CPU load on routers
  - Inconsistent topology information during convergence: worst! (why?)

Inconsistent topology information

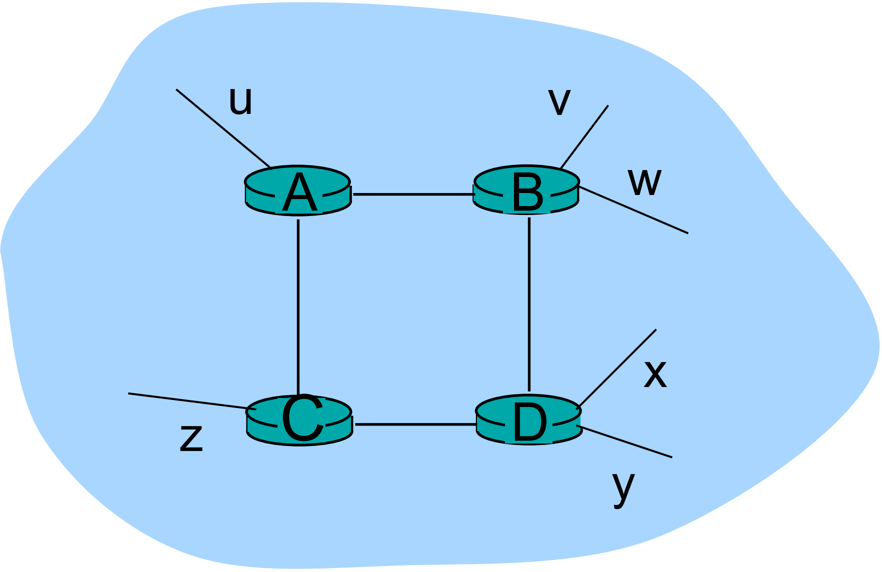
- Typical causes (not exhaustive)
  - One router finished with calculations, another one not yet
  - Relevant information has not yet reached entire network
    - LS: Broadcasts = fast
    - DV: Receive message, calculate table, inform neighbours: slow
  - DV: Count-to-infinity problem
  - LS: Different algorithm implementations!
  - LS: Problem if there is no clear rule for handling equal-cost routes
- Possible consequences?
  - Erroneously assuming some dst is not reachable
  - Routing loops
    - Think yourselves: What happens when there is a routing loop?

Intra-AS Routing

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

RIP (Routing Information Protocol)

- Distance vector algorithm
- Included in BSD-UNIX Distribution in 1982
- Distance metric: # of hops (max = 15 hops, ∞ := 16)
- Sometimes still in use by very small ISPs



From router A to subnets:

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

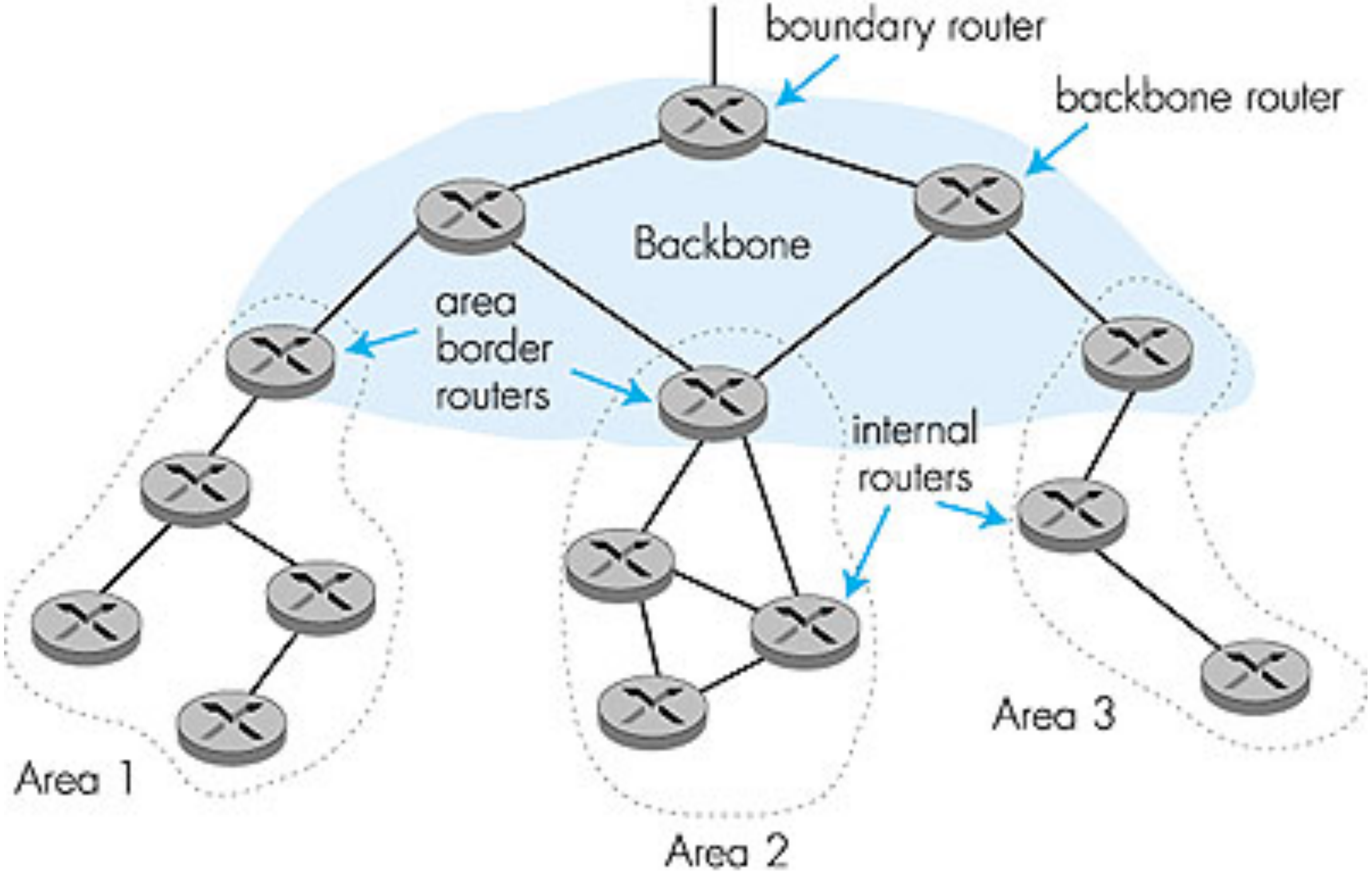
OSPF (Open Shortest Path First)

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

OSPF "advanced" features (not in, e.g., RIP)

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

Hierarchical OSPF



Hierarchical OSPF

- OSPF *can* create a two-level hierarchy
  - (similar, but not identical to to inter-AS and intra-AS routing within an AS)
- Two levels: local *areas* and the *backbone*
  - Link-state advertisements only within local area
  - Each node has detailed area topology;
    but only knows coarse direction to networks in other areas (shortest path to border router)
- *Area border routers:* "summarize" distances to networks in own area; advertise distances to other Area Border and Boundary routers
- *Backbone routers:* run OSPF routing limited to backbone
- *Boundary routers:* connect to other Ases
  - "The outside world" ≈ another area

Hierarchical Routing in the Internet

Our routing study thus far = idealisation

- all routers identical
- network "flat"

… *not* true in practice!

Scale = billions of destinations:

- Cannot store all destinations in routing tables
- Routing table exchange would swamp links
- Thousands of OSPF Areas? Would not scale!

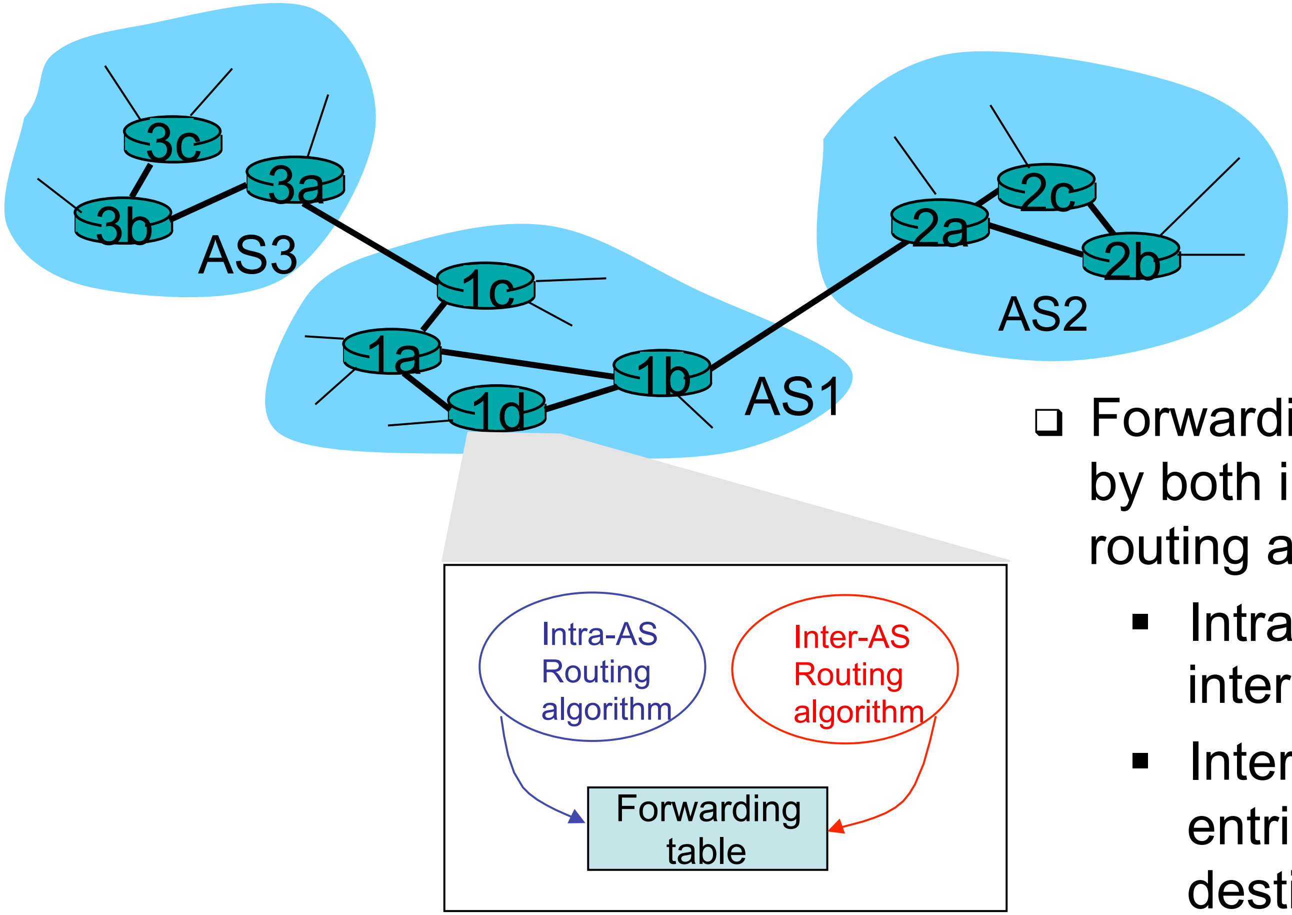
Administrative autonomy

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

Hierarchical Routing

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

Interconnected ASes



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

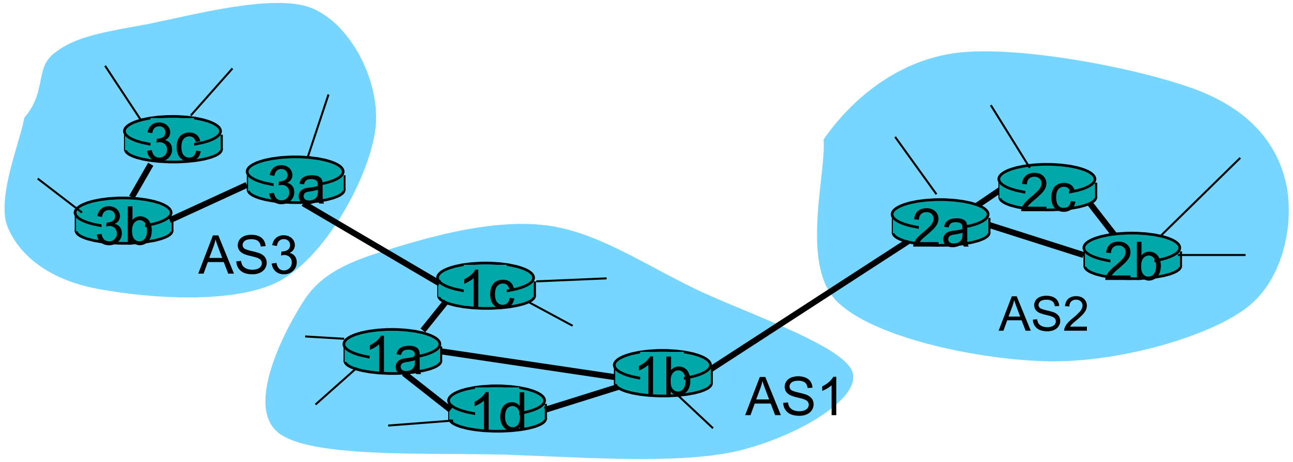
Inter-AS tasks

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

AS1 must:

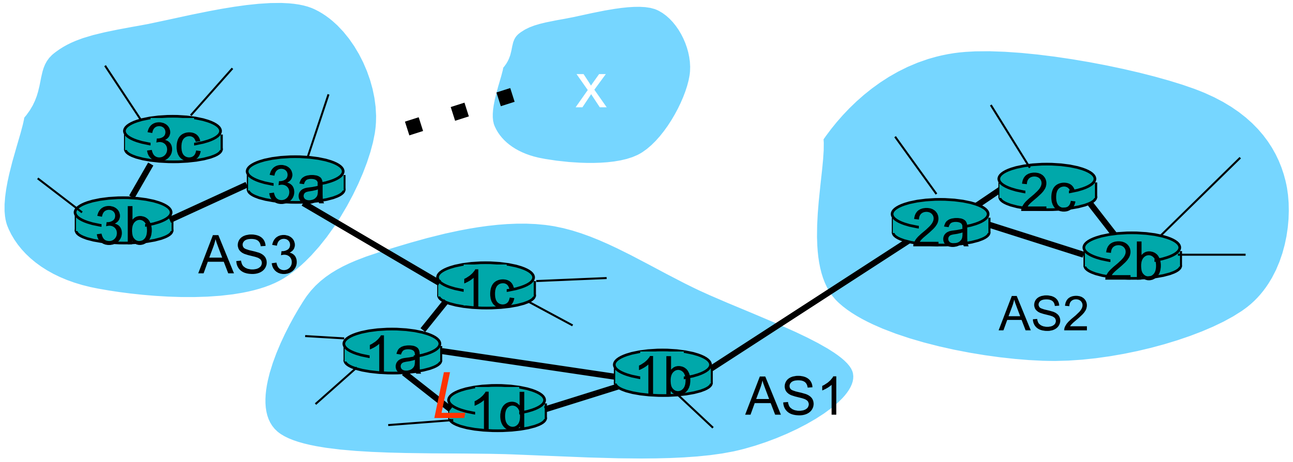
1. learn which destinations are reachable through AS2, which through AS3
2. propagate this reachability info *to all* routers in AS1 (i.e., not just the gateway routers)

Job of inter-AS routing!



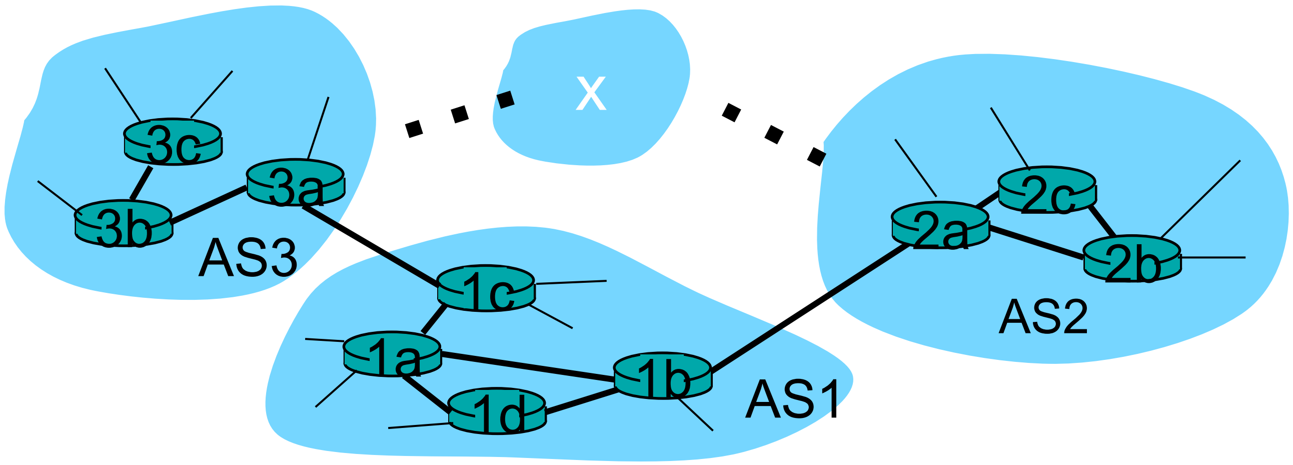
Example: Setting forwarding table in router 1d

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



Example: Choosing among multiple ASes

- Now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- To configure forwarding table, router 1d must determine towards which gateway it should forward packets for destination x.
    - "Do we like AS2 or AS3 better?"
    - Also the job of inter-AS routing protocol!



Interplay of inter-AS and intra-AS routing

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

⇒ Often, routers need to run *both* types of routing protocols… even if they are not directly connected to other ASes!

Internet inter-AS routing: BGP

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

BGP basics

- Pairs of routers (BGP peers) exchange routing info over semi-permanent TCP connections: BGP sessions
  - BGP sessions need not correspond to physical links!
- When AS2 advertises an IP prefix to AS1:
  - AS2 *promises* it will forward IP packets towards that prefix
  - AS2 can aggregate prefixes in its advertisement (e.g.: 10.11.12.0/26, 10.11.12.64/26, 10.11.12.128/25 into 10.11.12.0/24)

How does BGP work?

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

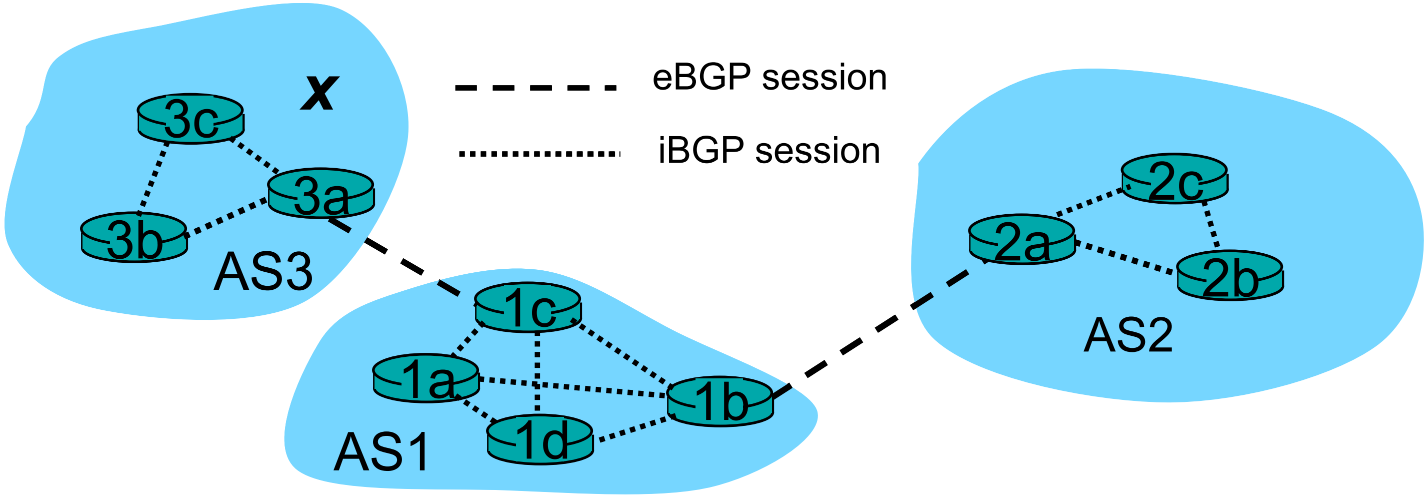
BGP updates

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

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

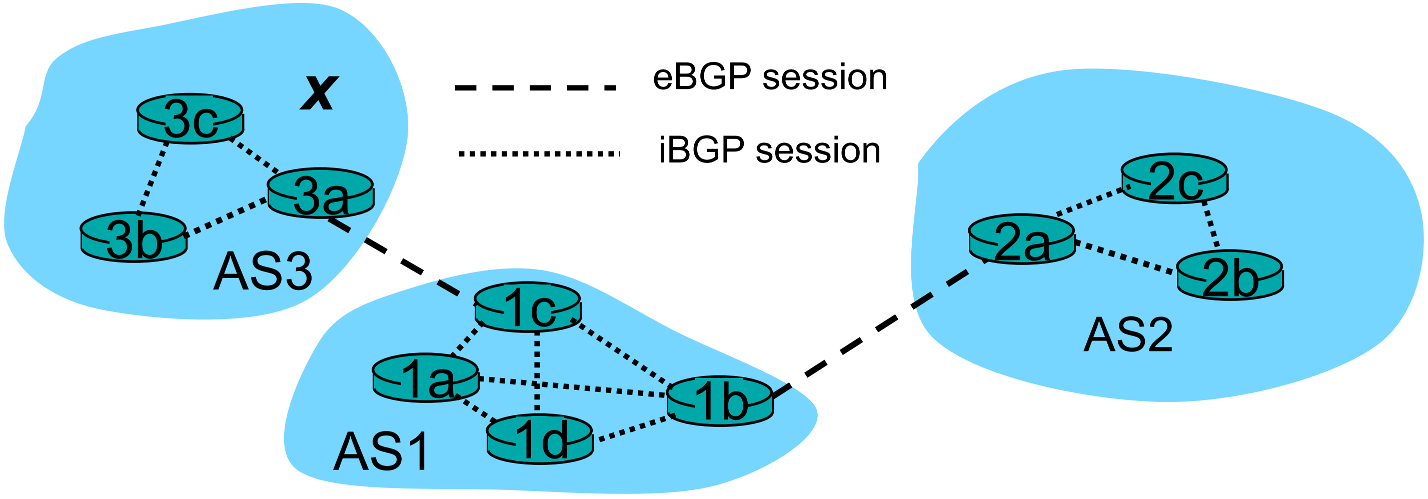
eBGP and iBGP

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



Distributing reachability info

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



Path attributes & BGP routes

- Advertised prefix includes [many] BGP attributes
  - prefix + attributes = “route”
- Most important attributes:
  - AS-PATH: contains ASes through which prefix advertisement has passed: e.g., AS 67, AS 17, AS 7018
  - NEXT-HOP: indicates specific internal-AS router to next-hop AS (may be multiple links from current AS to next-hop-AS)
- When gateway router receives route advertisement, it uses an import policy to accept/decline the route
  - More on this later

AS Numbers

- How do we express a BGP path?
- ASes identified by *AS Numbers* (short: ASN)
  Examples:
  - Leibnitz-Rechenzentrum = AS12816
  - Deutsche Telekom = AS3320
  - AT&T = AS7018, AS7132, AS2685, AS2686, AS2687
- ASNs used to be 16bit, but can be 32bit nowadays
  - May have problems with 16bit ASNs on very old routers
- ASN assignment: similar to IP address space
  - ASN space administered IANA
  - Local registrars, e.g., RIPE NCC in Europe

BGP update: Very simple example

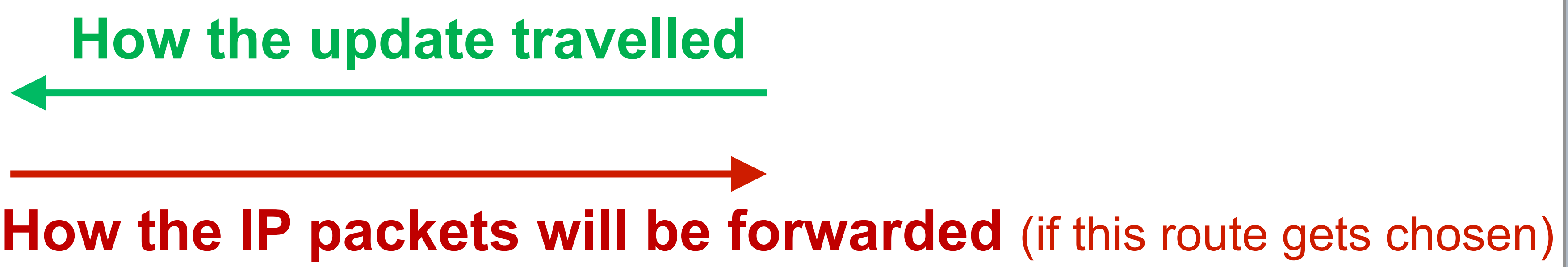
- Type: Announcement
  - Either this is a new route to the indicated destination,
  - or the existing route has been changed
- Destination prefix: 10.11.128.0/17
- AS Path:

`7018 3320 4711 815 12816`

Current AS

Originator:
The AS that “owns”
10.11.128.0/17

- Next Hop: 192.168.69.96
  - The router that connects the current AS to AS 3320



How the update travelled

How the IP packets will be forwarded (if this route gets chosen)

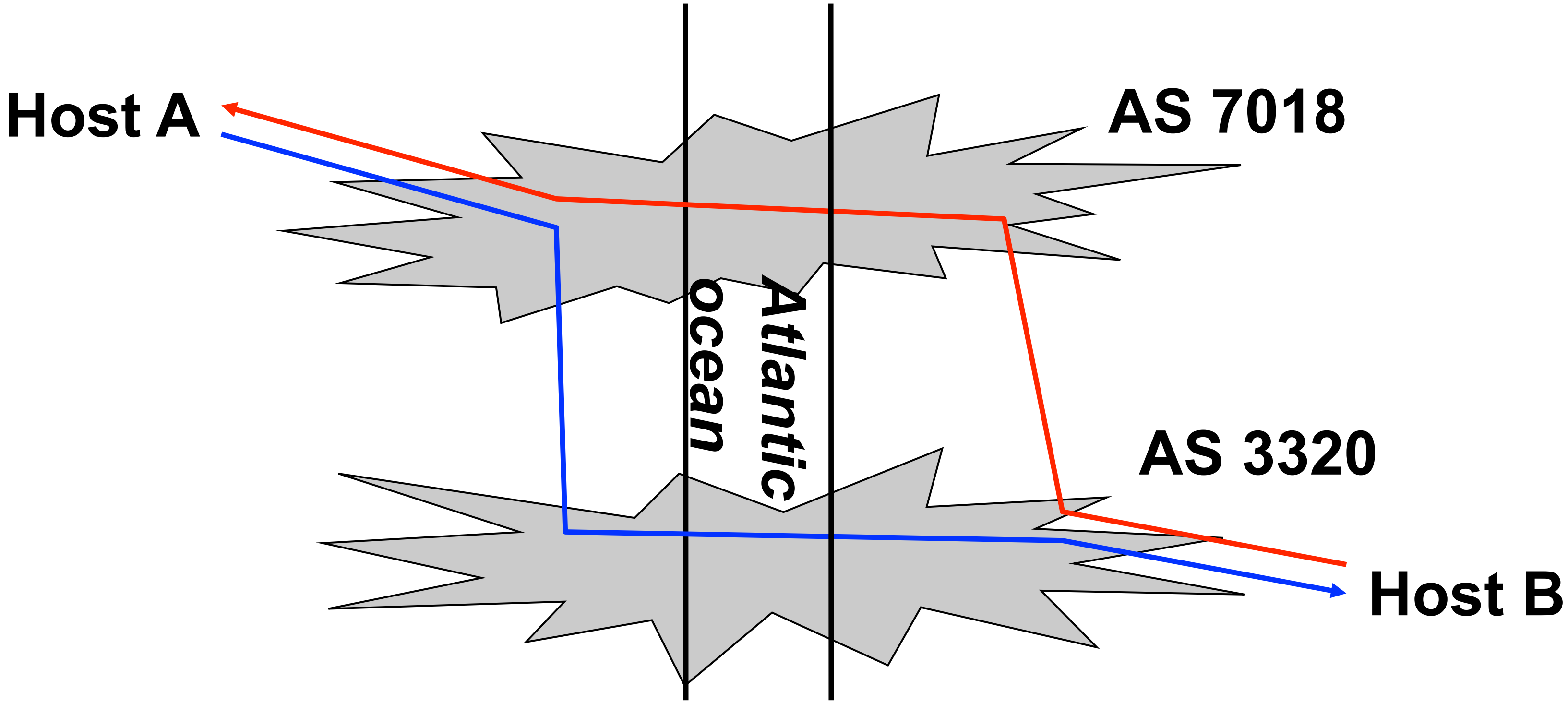
BGP route selection

- ❑ Router may learn about more than 1 route to some prefix
  ⇒ Router must select the best one among these
- ❑ Elimination rules (**simplified**):

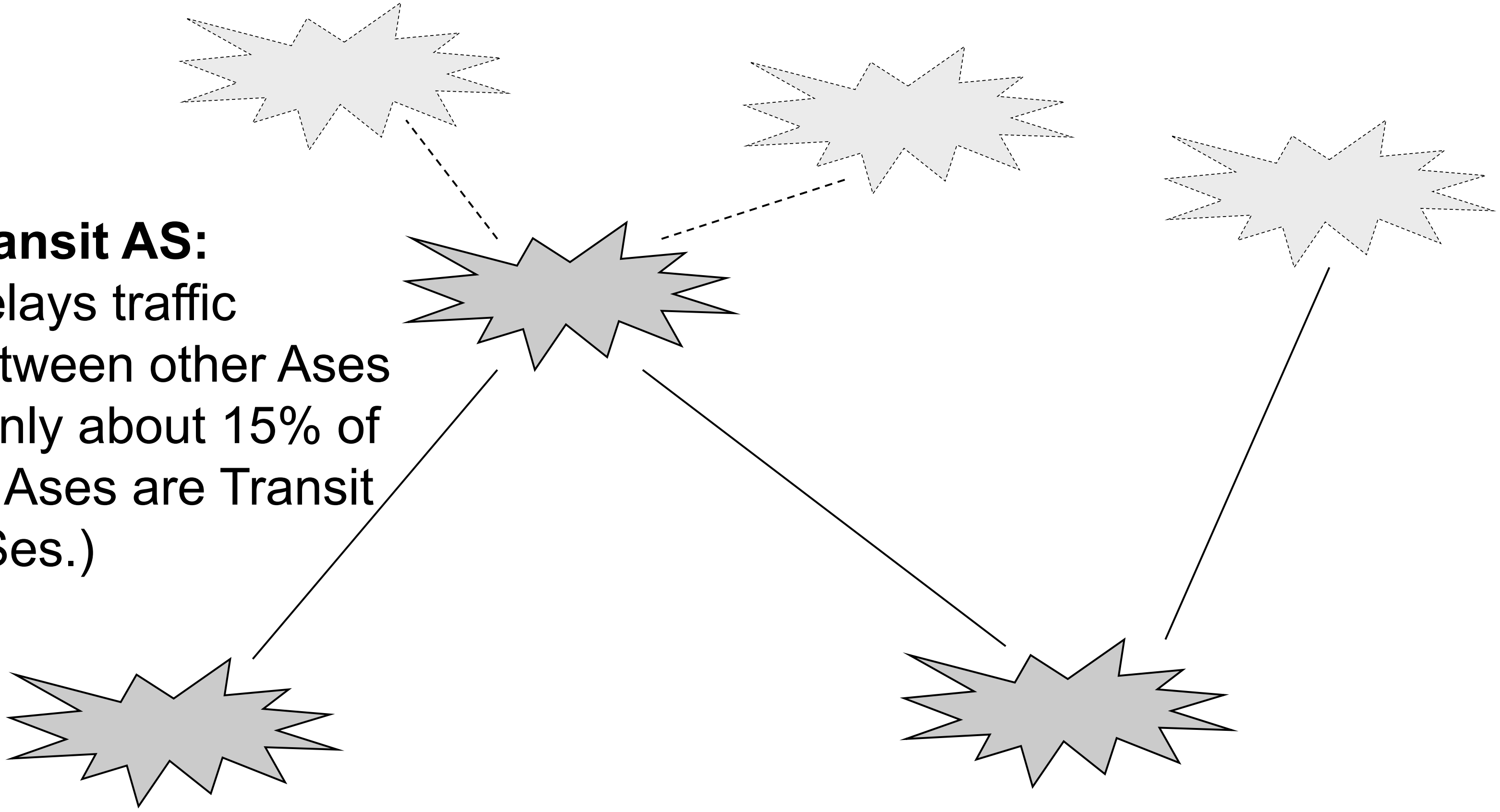
1. Local preference value attribute: policy decision
2. Shortest AS-PATH
3. Closest NEXT-HOP router: hot potato routing
4. Additional criteria

Business and Hot-potato routing

- ❑ Interaction between Inter-AS and Intra-AS routing
  - Business: If traffic is destined for other AS, get rid of it ASAP
  - Technical: Intra-AS routing finds shortest path to gateway
- ❑ Multiple transit points ⇒ asymmetrical routing
  - ❑ Asymmetrical paths are very common on the Internet



Terminology: Transit AS, stub AS, multi-homed AS



Transit AS:
Relays traffic between other Ases (Only about 15% of all Ases are Transit ASes.)

Stub AS: Buys transit from only one other AS, but does not offer transit for other ASes

Multi-homed AS: Buys transit from ≥2 other ASes, but does not offer transit for other ASes

Business relationships

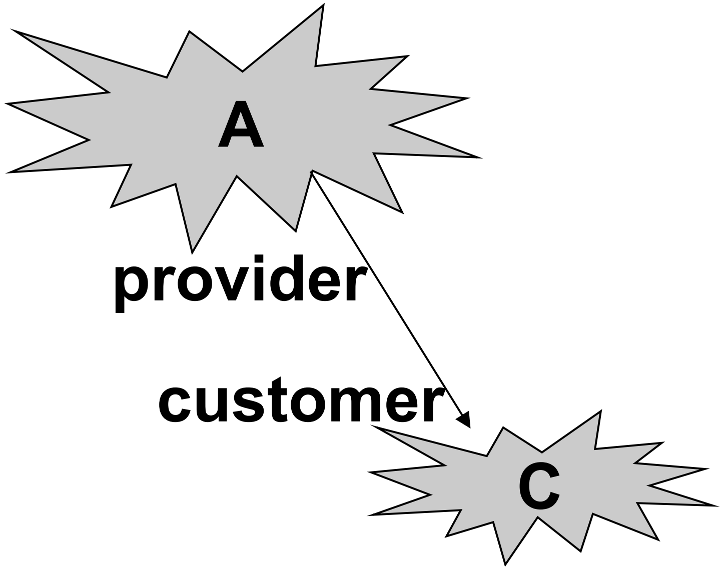
- Internet = network of networks (ASes)
  - Many thousands of ASes
  - Not every network connected to every other network
  - BGP used for routing between ASes
- Differences in economical power/importance
  - Some ASes huge, intercontinental (AT&T, Cable&Wireless)
  - Some ASes small, local (e.g., München: M”Net, SpaceNet)
- Small ASes customers of larger ASes: Transit traffic
  - Smaller AS pays for connecting link + for data = buys transit
  - Business relationship = customer—provider
- Equal-size/-importance ASes
  - Usually share cost for connecting link[s]
  - Business relationship = peering (*specific* transit traffic is for free)
- Warning: peering (“equal-size” AS)
  ≠ peers of a BGP connection (also may be customer or provider)
  ≠ peer-to-peer network

Business and policy routing (1)

- Basic principle #1 (Routing)
  - Prefer routes that incur financial gain
- Corollary: If you have the choice, then…
  - …routes via a customer…
  - …are better than routes via a peer, which…
  - …are better than routes via a provider.
- Basic principle #2 (Route announcement)
  - Announce routes that incur financial gain if others use them
    - Others = customers
  - Announce routes that reduce costs if others use them
    - Others = peers
  - Do not announce routes that incur financial loss (…as long as alternative paths exist)

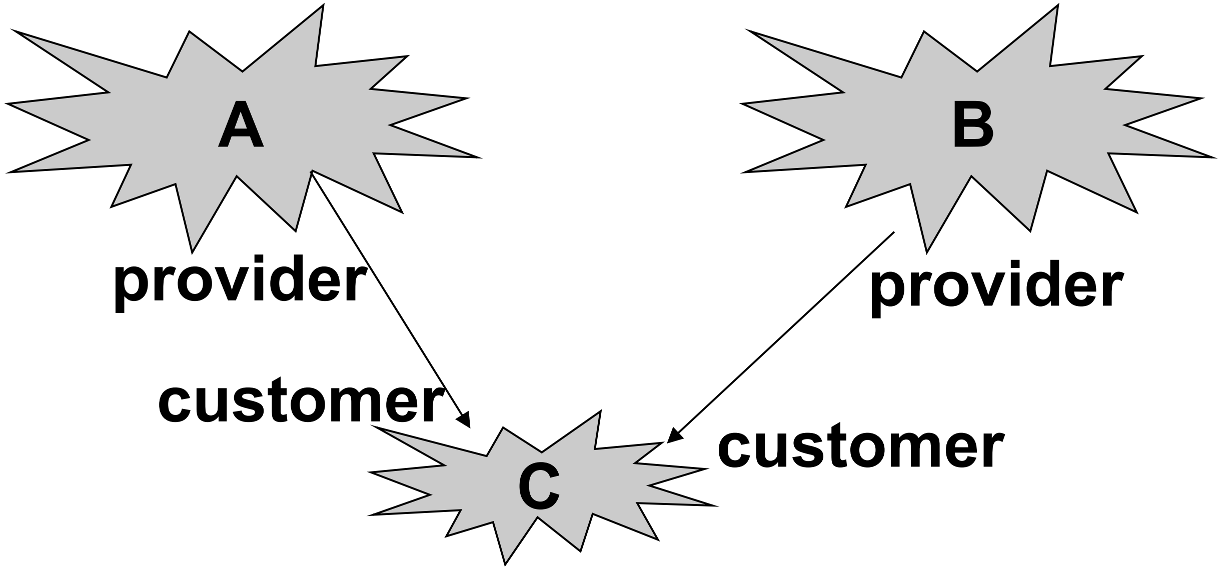
Business and policy routing (2)

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



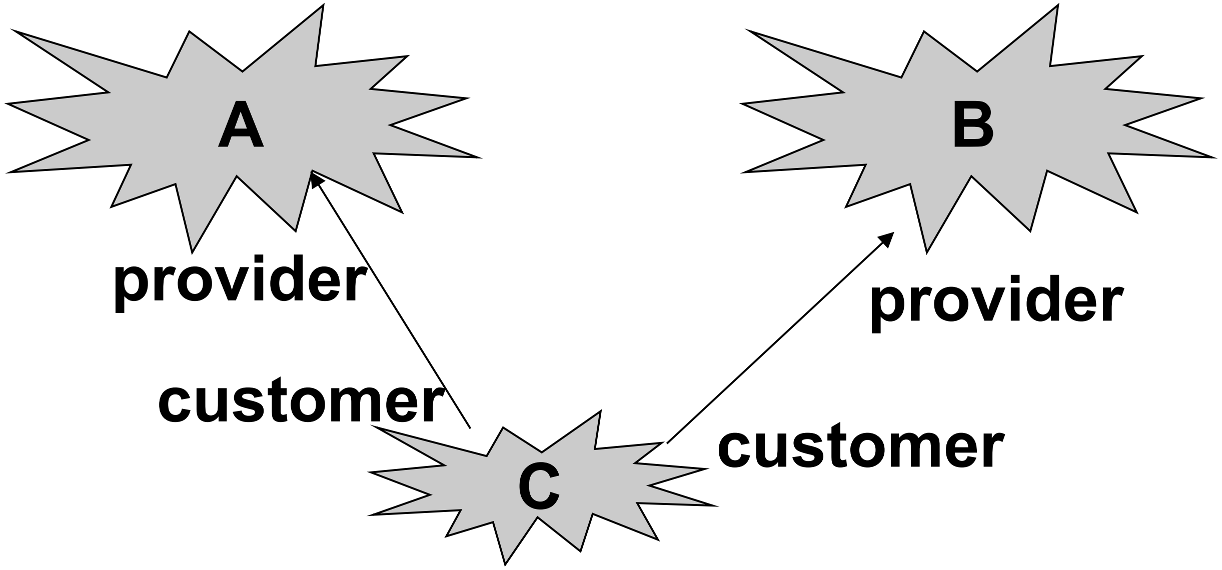
Business and policy routing (3)

- A and B tell C all routes they use to reach other ASes
  - The more traffic flows from C to A, the more money A makes
  - The more traffic flows from C to B, the more money B makes
  - C will pick the one with the cheaper offer / better quality / …



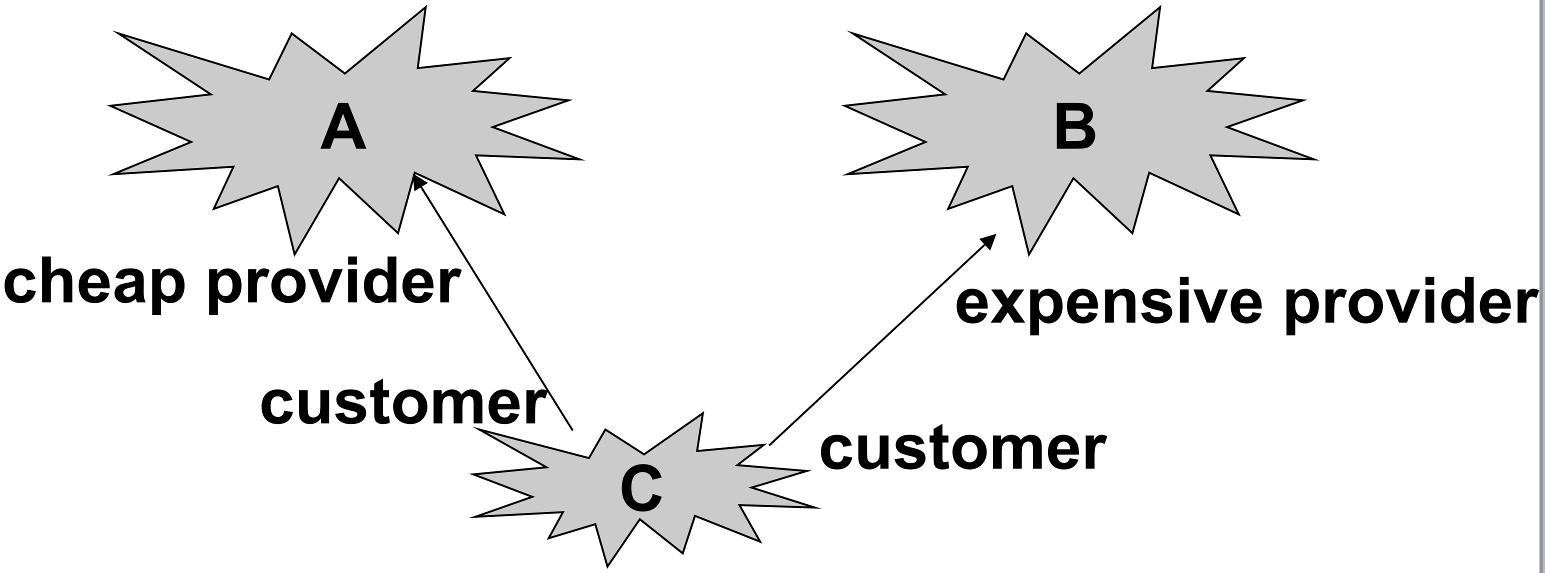
Business and policy routing (4)

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



Business and policy routing (5): AS path prepending

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

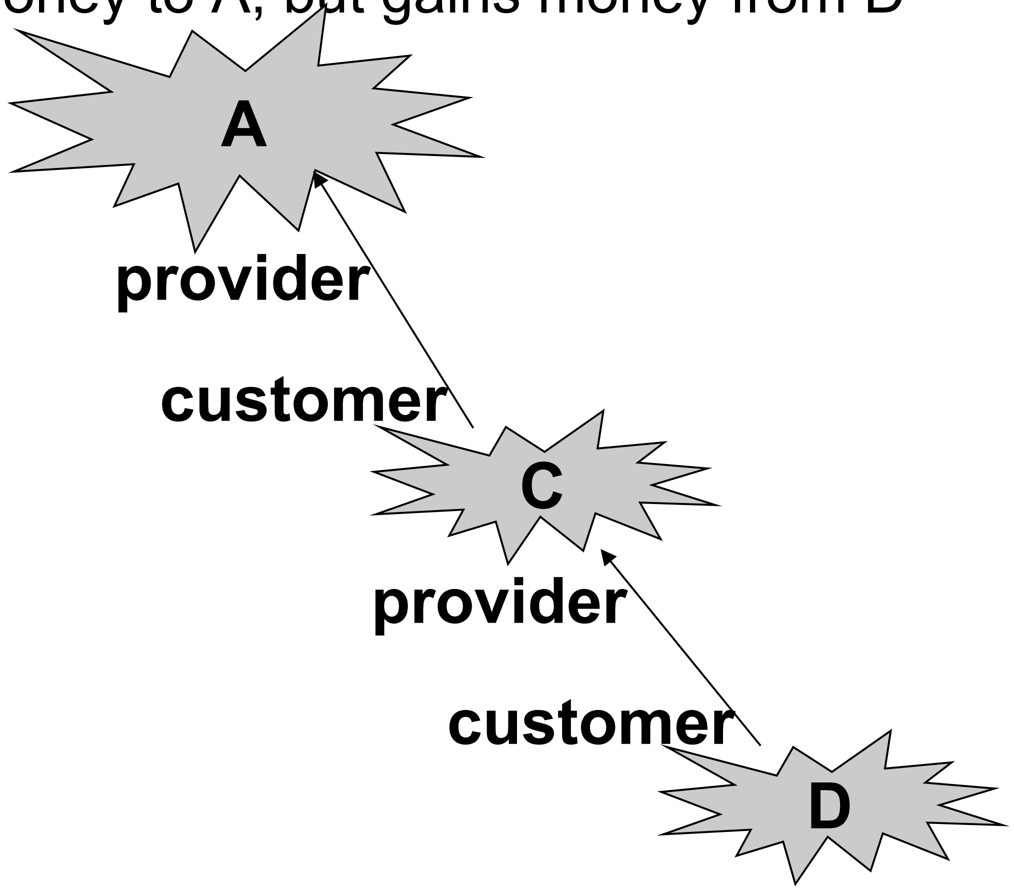


AS path prepending

- The same ASN *subsequently* within an AS path does not constitute a loop
- Recall the elimination rule for selecting from multiple path alternatives
  - "Prefer the shortest AS path" is rule 2
  - Only ignored if *Local Pref* value is set
  - AS path prepending makes a route less attractive – will then only be used when there is no alternative
- How many times to repeat the AS number?
  - Usually just 1 or 2 repetitions
  - More than ≈5 is useless

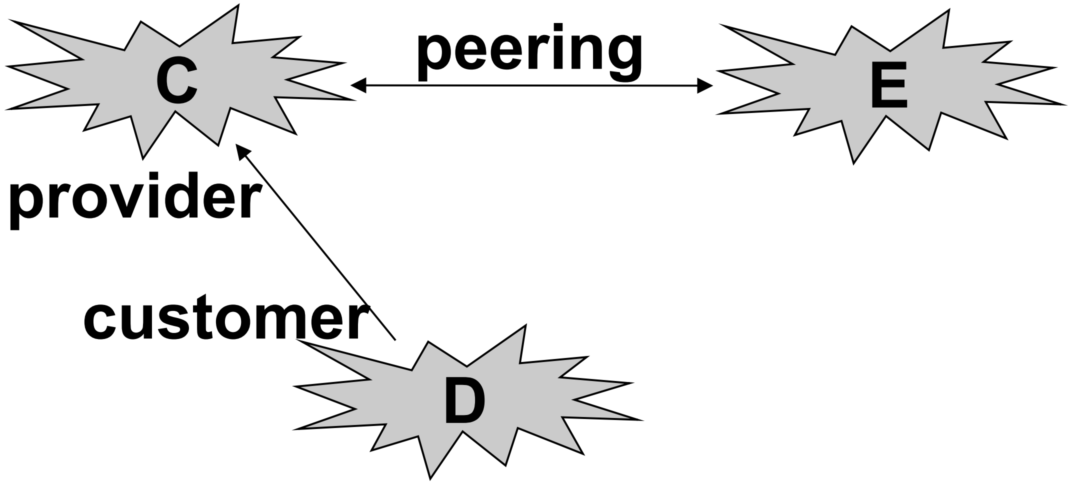
Business and policy routing (6)

- What should C announce here?
  - C tells A about its own prefixes
  - C tells A about its route to D's prefixes: loses money to A, but gains money from D



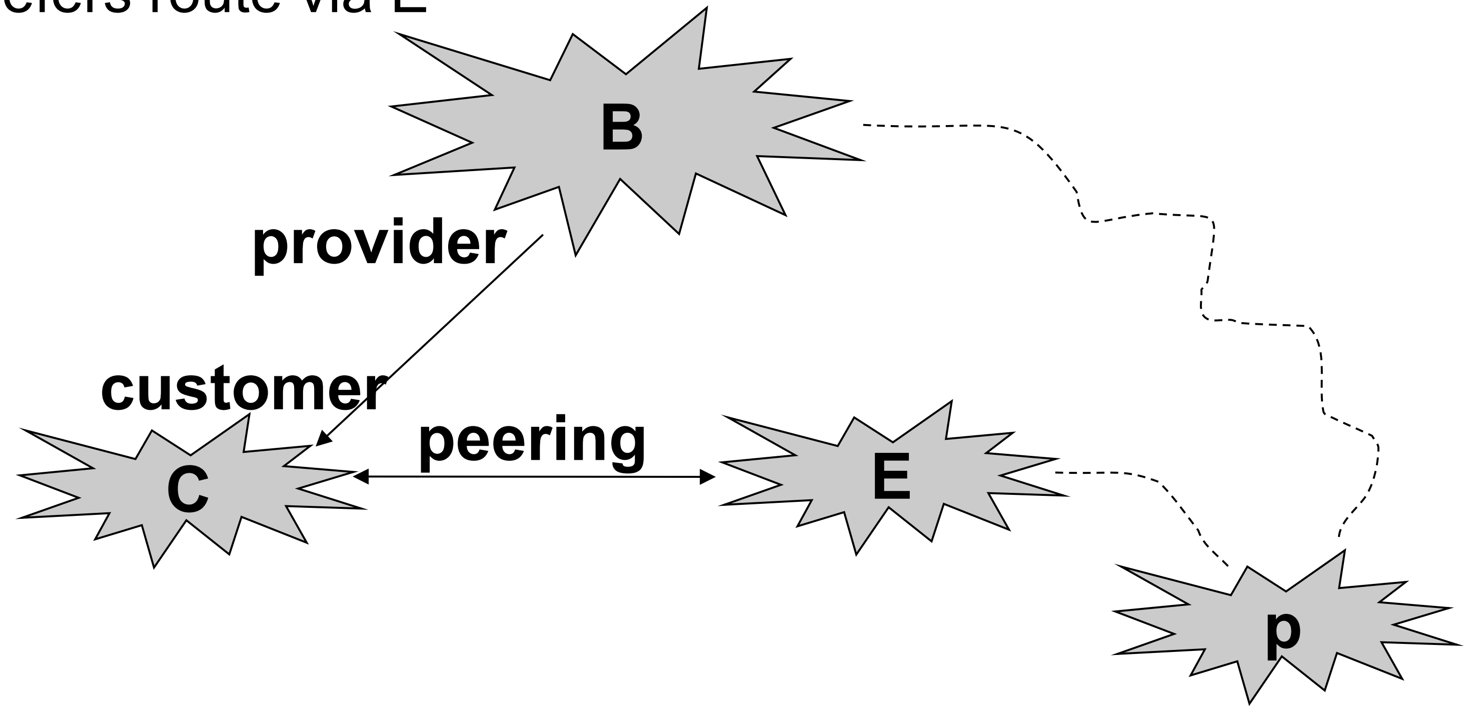
Business and policy routing (7)

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



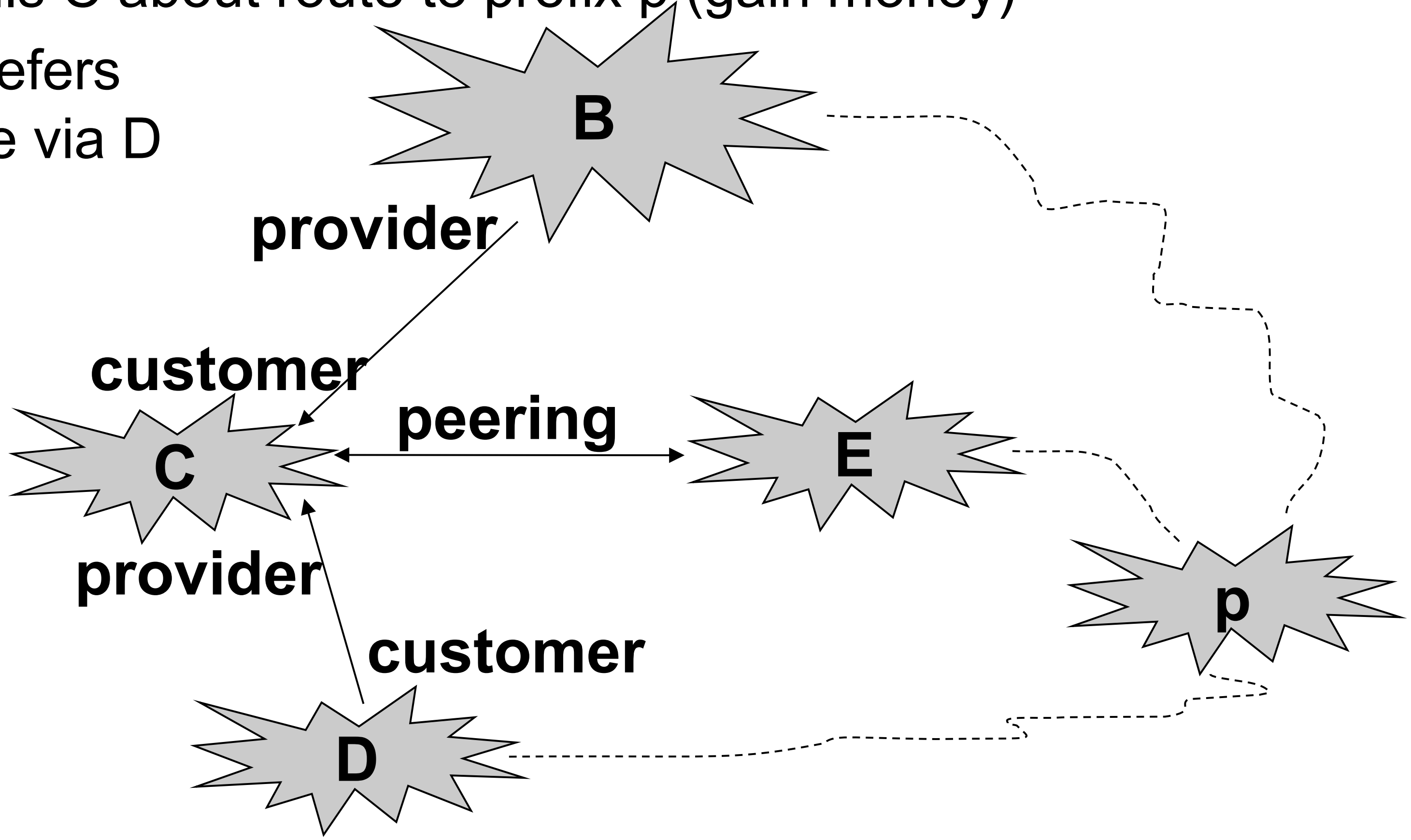
Business and policy routing (8a)

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



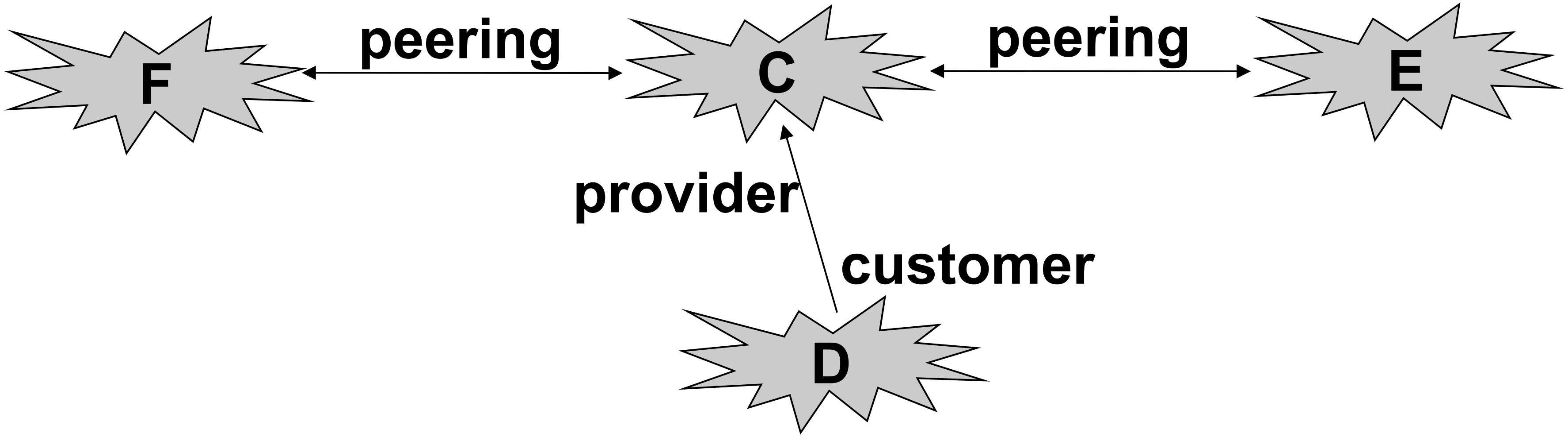
Business and policy routing (8b)

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



Business and policy routing (9)

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

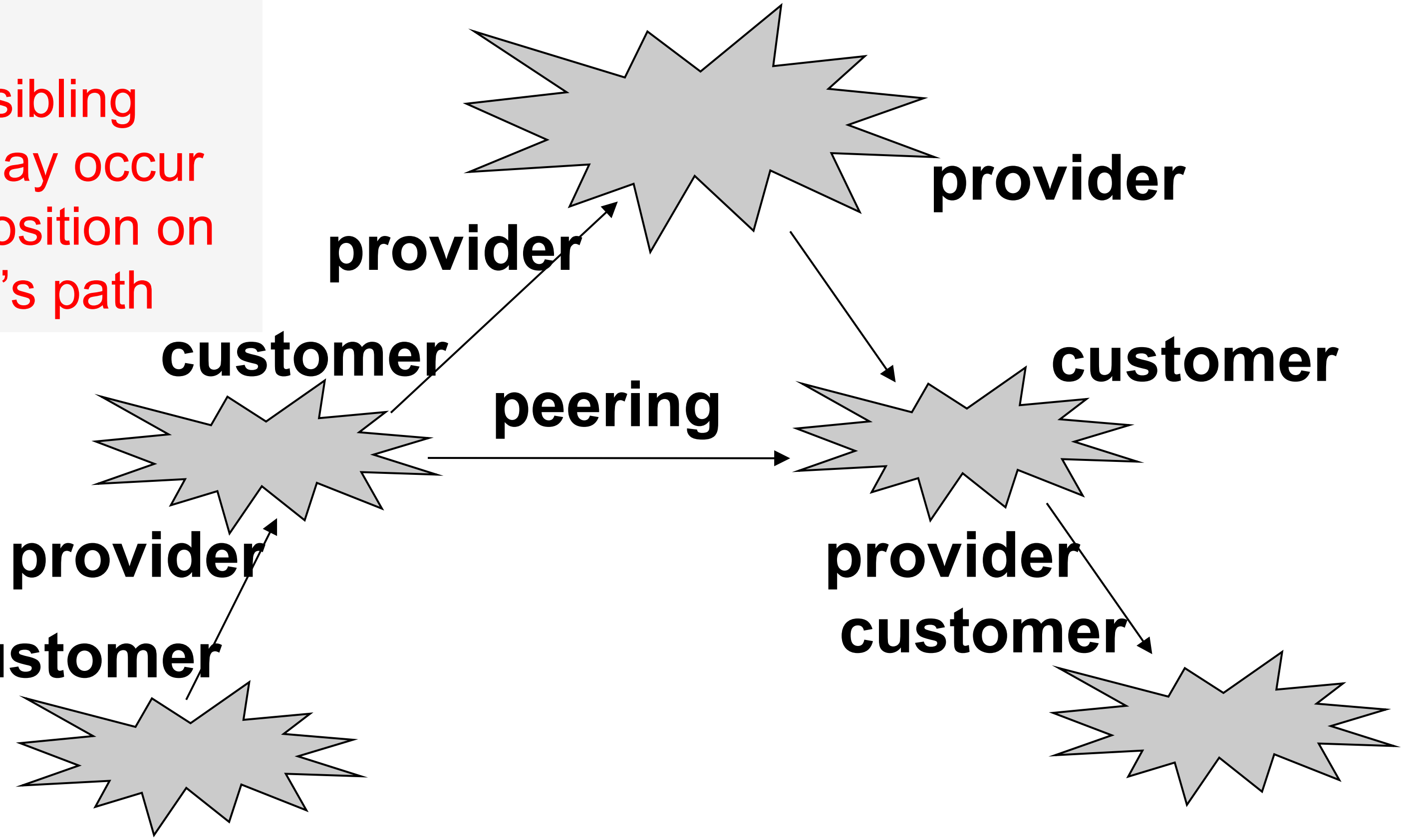


Policy routing: Valley-free routing (idealised!)

Results: Packets always travel…

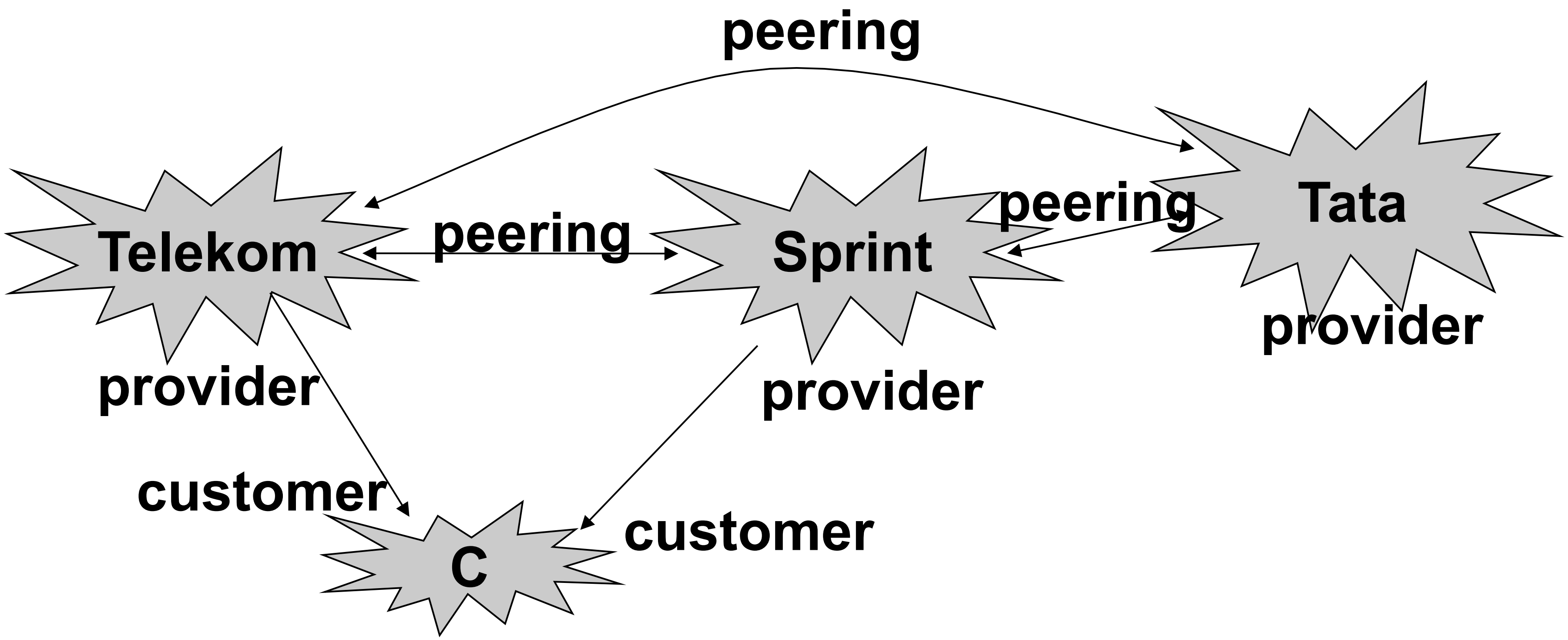
1. upstream: sequence of C→P links (possibly length = 0)
2. then possibly across *one* peering link
3. then downstream: sequence of P→C links (possibly length = 0)

But:
Sibling–sibling edges may occur at any position on a packet's path



Business and policy routing (10): “Tiers” / “DFZ”

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



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

- Tier-1/DFZ = only peerings, no providers
- Tier-2 = only peerings and one or more Tier-1 providers
- Tier-3 = at least one Tier-2 as a provider
- Tier-*n* = at least one Tier-(n-1) provider
  - defined recursively
  - *n*≥4: Rare in Western Europe, North America, East Asia

- "Tier-1.5" = almost a Tier-1 but pays money for *some* links
  - Example: Deutsche Telekom used to pay money to Sprint, but is now Tier-1
  - Marketing purposes: Tier-1 sounds better

Siblings

- Not everything is provider/customer or peering
- Sibling = mutual transit agreement
  - Provide connectivity to the rest of the Internet for each other
  - ≈ very extensive peering
- Examples
  - Two small ASes close to each other that cannot afford additional Internet services
  - Merging two companies
    - Merging two ASes into one = difficult,
    - Keeping two ASes and exchaning everything for free = easier
  - Example: AT&T has five different AS numbers (7018, 7132, 2685, 2686, 2687)

BGP policy routing: Technical summary

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

BGP policy routing: Business relationship summary

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

Where to peer

(Here: Peering = having a BGP relationship)

A) Private peering

- ❑ The obvious solution: “Let’s have a cable from your server room to our server room”

B) At public peering locations (Internet Exchange Point, IX, IXP)

- ❑ “A room full of switches that many providers connect to”
- ❑ Configure VLAN connections in switch, instead of having to put in O(n^2) separate wires
- ❑ Examples:
  - ❑ DE-CIX, Frankfurt (purportedly largest in world)
  - ❑ AMS-IX, Amsterdam
  - ❑ LINX, London
  - ❑ MSK-IX, Moscow

BGP "security" today – a sad topic…

- BGP sessions use TCP
  - No encryption – interceptors can read everything
  - "Authentication": accept or decline AS number in OPEN message
  - Further authentication (recommended, but optional): TCP-MD5, TCP-AO
    - TCP header option contains cryptographic signature of packet
    - TCP connections only accepted from peers with accepted signature
    - No protection against replay attacks, against eavesdropping, …
  - Only accept BGP sessions from specific IP addresses?
- Defensive filtering
  - Provider knows prefixes of its (stub) AS customers:
    - Don't accept updates for other prefixes from them
    - Don't accept updates with other ASNs from them

BGP Routing security case study 1: How Pakistan Telecom inadvertently hijacked Youtube

- On 2008-02-25, users worldwide could not reach YouTube…:
- Pakistan Telecom were ordered by a Pakistani court to block access to a certain YouTube video
- Only feasible choice was to block all YouTube traffic (208.65.152.0/2**2**)
- They created an internal "black hole route" for their network:
  - Manual insertion of a new route for 208.65.152.0/2**4** into IGP
  - Packets sent via that route get discarded at the endpoint
  - Longest prefix match ➔ This route absorbs ¼ of the /22 traffic (in this case: the part containing the servers)
- Unfortunately, this black hole route slipped into eBGP…
  - … so BGP routers world-wide saw the new route and used it
- Quick remedy by Google/YouTube?
  - Announcement of even longer prefixes 208.65.152.0/25 and 208.65.152.128/25

Youtube hijacking: Assessment

- Which security mechanisms could have worked here?
- Authentication?
  - No!
  - Pakistan Telecom is a legit BGP speaker
  - Not known for malicious behaviour
- Defensive filtering?
  - Probably not!
  - Pakistan Telecom ist not just some tiny stub AS with only one or two prefixes

BGP Routing security case study 2: How a small Czech provider terrorized the world’s BGP routers

- On 2009-02-16, there was a world-wide surge in BGP updates.
- Small Czech provider SuproNet (AS 47868) wanted to announce their prefix with AS path prepending
- Cisco syntax: `[…] as-path prepend 47868 47868 47868`
- …but they used MikroTik routers. Syntax: `bgp-prepend 3`
- 47868 cast into 8 bits: 47868 mod 256 = 252
- Result: AS path of length 252 (=unusually long)
- Path became longer as the announcement travelled through the world… and approached length 256 (=maximum)
- Many Cisco routers could not handle the long AS path and sent out invalid BGP messages
- Result = BGP session resets at their BGP neighbours
  - Remove all BGP routes learned from the crashed router
  - Accordingly, send BGP updates to neighbours

AS path terror: Assessment (1)

So… who is to blame?

❑ SuproNet
  - Network administrator principle:
    Thou shalt read the documentation of your router…
  - …especially if it is about BGP

❑ MikroTik
  - Number was way too large
  - UI design principle:
    Thou shalt do error checking on user input!
    (If a user can enter garbage, he will do it.)

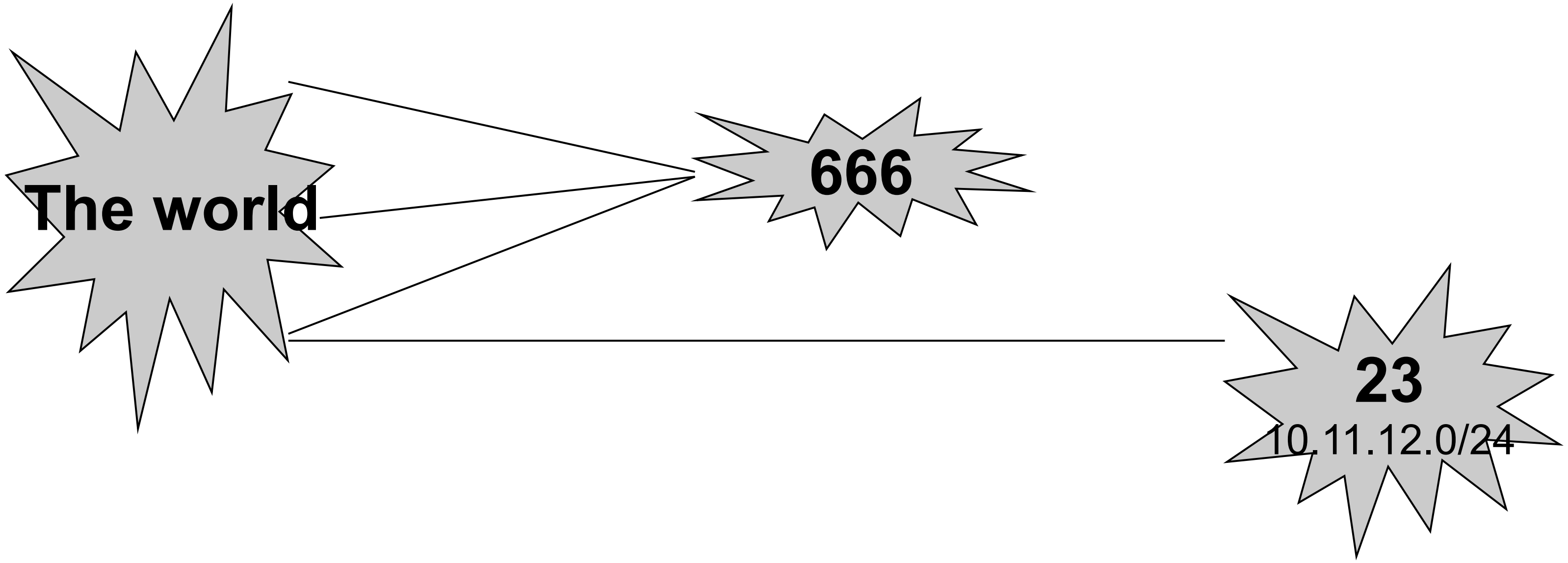
❑ Cisco
  - Strange input (long AS path) resulted in malformed output
  - Network software design principle:
    - Thou shalt do error checking on network input
    - Error checking on network output is a good idea

AS path terror: Assessment (2)

- Which security mechanisms could have worked here?
- Authentication?
  - No!
  - SuproNet is a legit BGP speaker
  - Not known for malicious behaviour
- Defensive filtering?
  - SuproNet just announced their very own prefix
- Intercepting malformed BGP updates?
  - That’s exactly what crashed those BGP sessions…

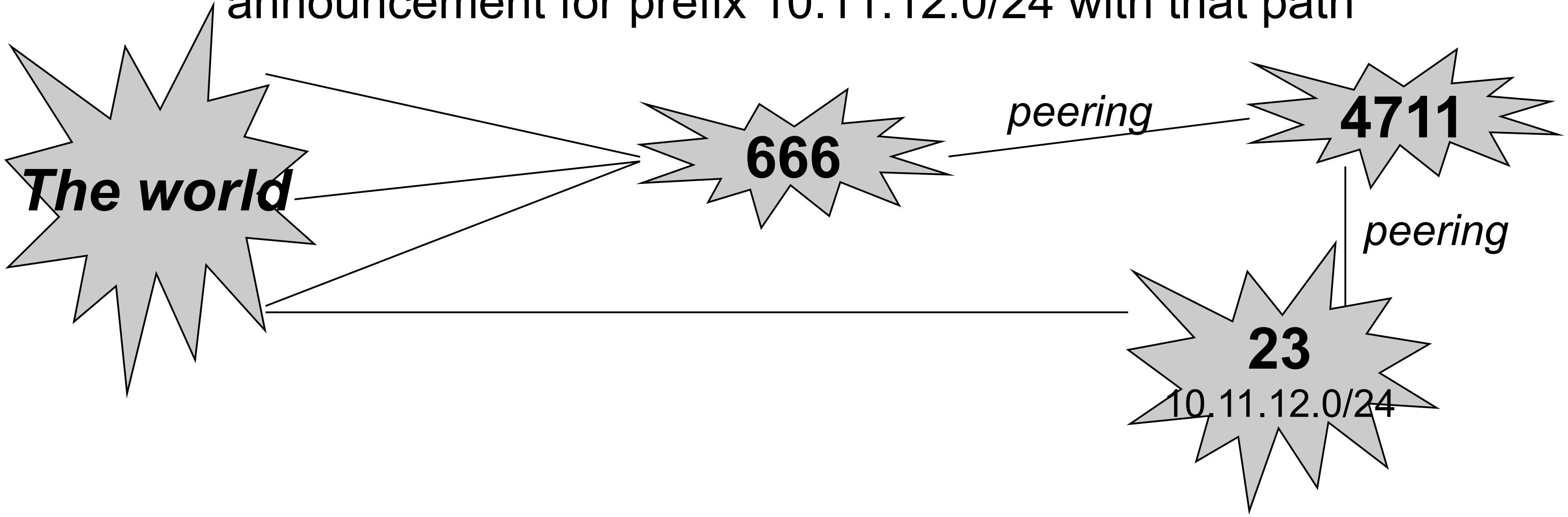
BGP security: Suggested mechanisms (1)

- **Origin authentication:** Only ASes that “own” a prefix can announce it
  - Can secure this cryptographically (PKI)
  - Can we outsmart this?
    - Let 10.11.12.0/24, owned by AS23, be the prefix to be hijacked
    - Rogue AS 666 can lie by announcing non-existent paths: Prefix: 10.11.12.0/24, AS path: 666 23



BGP security: Suggested mechanisms (2)

- **Secure origin authentication:** Only paths that physically exist can announce it
  - Cryptographically secured path database
  - Can we outsmart this?
    - Can announce paths that we should not see
    - Rogue AS666 knows paths 23–4711 and 4711–666 exist
    - Can announce 66 4711 23, even though it never received an announcement for prefix 10.11.12.0/24 with that path



S-BGP

- Secure origin authentication
- Additional attribute allows to sign a route step-by-step
- IPsec protects updates
- Can we outsmart this?
  - Rogue AS666 can still announce a “good” route but then actually use a “bad” route – or even drop the traffic

BGP security: Further reading

- Renesys blog:
  - Posts with ‘security’ tag: www.renesys.com/blog/security/
  - Entry “Reckless driving on the Internet”
  - Entry “Longer is not always better”
  - Entry “Pakistan hijacks YouTube”

- Butler, Farley, McDaniel, Rexford:
  A survey of BGP security issues and solutions
  Proceedings of the IEEE, January 2010

- Goldberg, Schapira, Hummon, Rexford:
  How secure are secure interdomain routing protocols?
  Proceedings of ACM SIGCOMM, August 2010

Routing: Optimization purposes

- Inter-AS routing
  - Optimality = select route with highest revenue/least loss
  - Mainly policy driven – we’ve seen that now
- Intra-AS routing
  - Optimality = configure routing such that network can host as much traffic as possible
  - Traffic engineering methods

Traffic Engineering

1. Collect traffic statistics: Traffic Matrix
   - ❑ How much traffic is flowing from A to B?
   - ❑ Often difficult to measure!
     - ❑ Drains router performance
     - ❑ Therefore often estimated – active research area
     - ❑ Alternative: Build lots of MPLS tunnels, measure each tunnel
2. Optimize routing
   - ❑ E.g., calculate good choice of OSPF weights
   - ❑ Typical goal: minimize maximum link load in entire network; keep average link load below 50% or 70%
     - ❑ (Why? Fractal TCP traffic leads to spikes.)
3. Deploy new routing
   - ❑ Performance may deteriorate during update
   - ❑ E.g., routing loops during OSPF convergence

Dynamic traffic engineering

Why static? Why don't we do it dynamically?

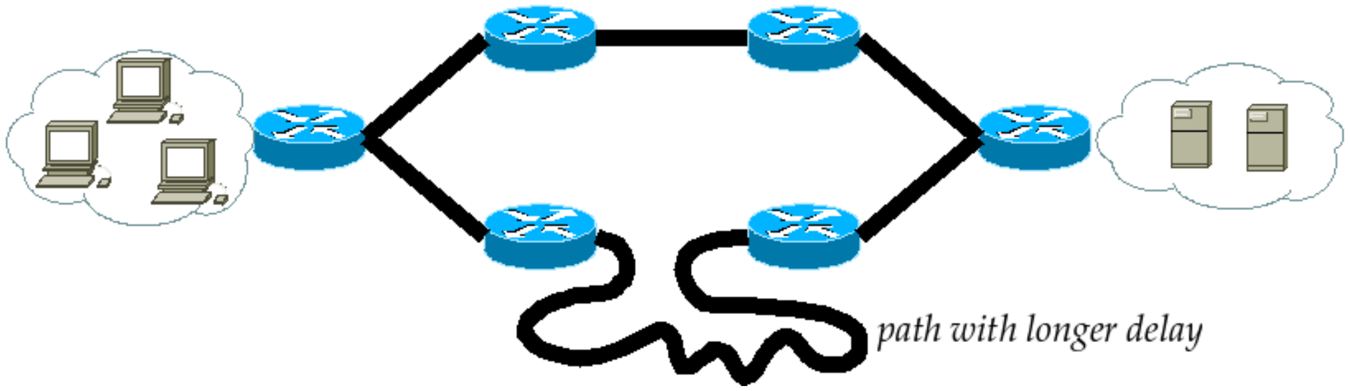
- ❑ Prone to oscillations and chaotic behaviour
  - ▪ Bad experiences in the ARPANET
  - ▪ Ex.: Route A congested, route B free
→ Everyone switches from A to B
→ Route A free, route B congested → …
- ❑ Routing loops during convergence → packet losses
- ❑ Packet reordering:
  - ▪ Packet P1 arrives later than Packet P2
  - ▪ TCP will think that P1 got lost! ⇒ congestion control!
- ❑ Actually, a difficult problem
  - ▪ Stale information
  - ▪ Interaction with TCP congestion control
  - ▪ Interaction with dynamic TE mechanisms in other ASes
- ❑ Thus: Congestion control in end hosts (TCP), usually not in network

Multipath routing

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

Multipath routing: TCP problem

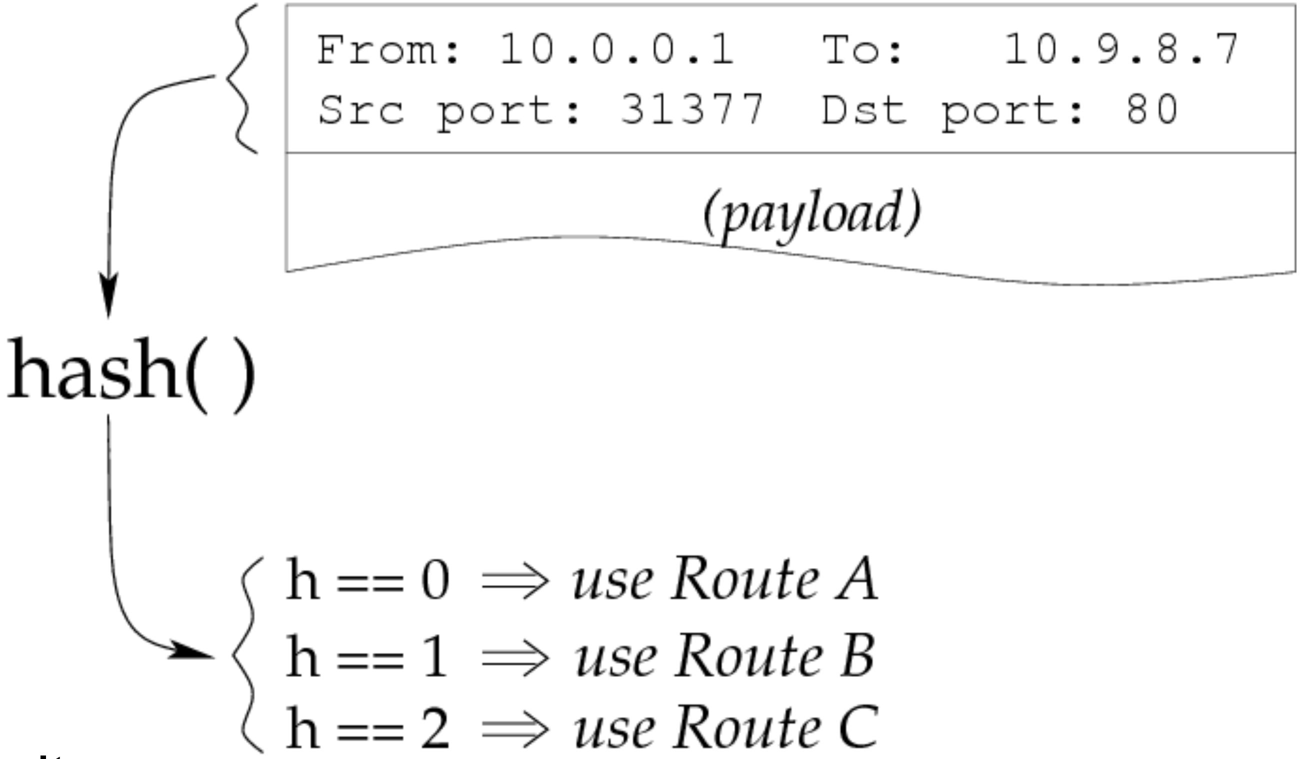
- How to distribute traffic? Naïve approaches:
  - Round-robin
  - Distribute randomly
- Equal cost does not mean equal latency:



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

Multipath routing: Solution

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



```
From: 10.0.0.1   To:    10.9.8.7
Src port: 31377  Dst port: 80
```

(payload)

hash( )

$h == 0 \Rightarrow$ *use Route A*

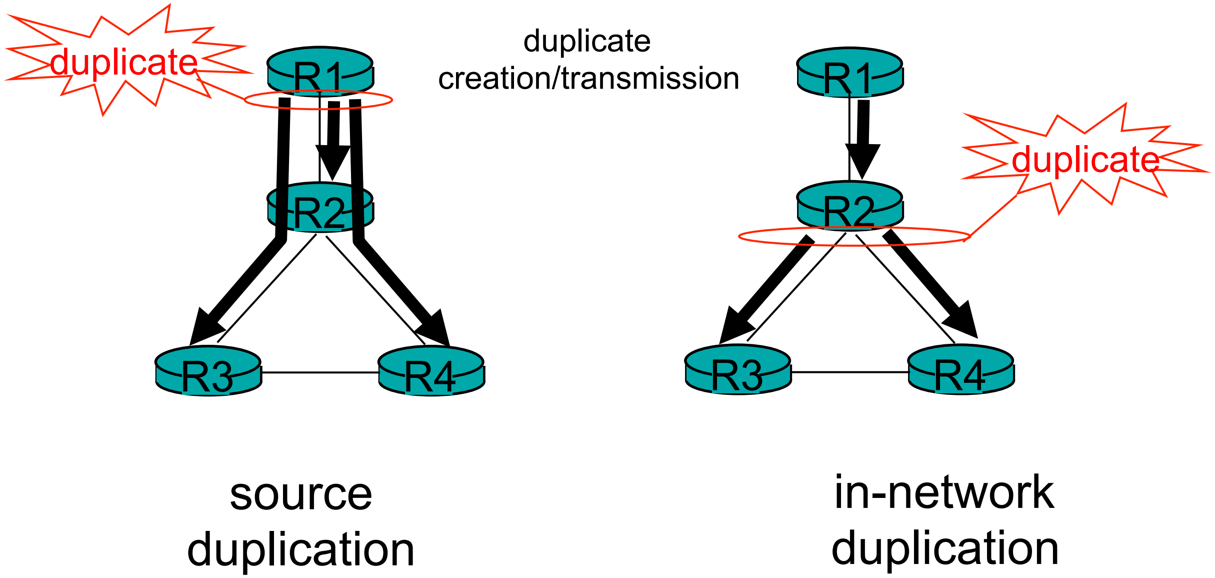
$h == 1 \Rightarrow$ *use Route B*

$h == 2 \Rightarrow$ *use Route C*

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

Broadcast Routing

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



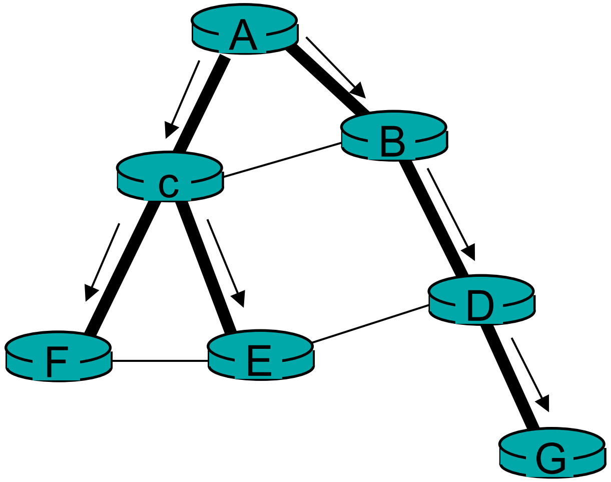
- Source duplication: how does source determine recipient addresses?

In-network duplication

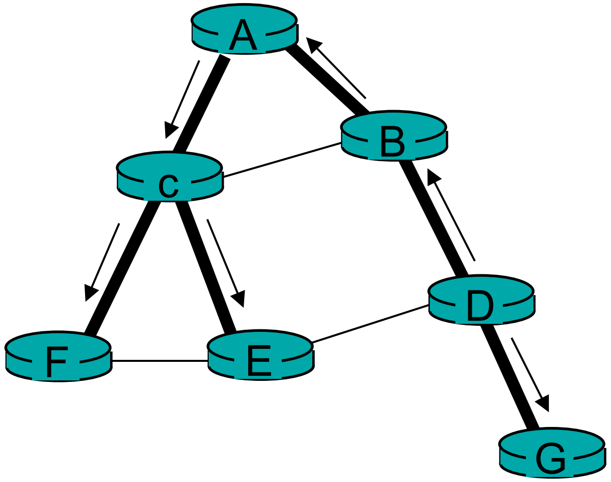
- Flooding: when node receives broadcast packet, sends copy to all neighbours
  - Problems: cycles & broadcast storm
- Controlled flooding: node only broadcasts packet if it hasn’t broadcast same packet before
  - Node keeps track of packet IDs already broadcast
  - Or reverse path forwarding (RPF):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

Spanning Tree

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



(a) Broadcast initiated at A

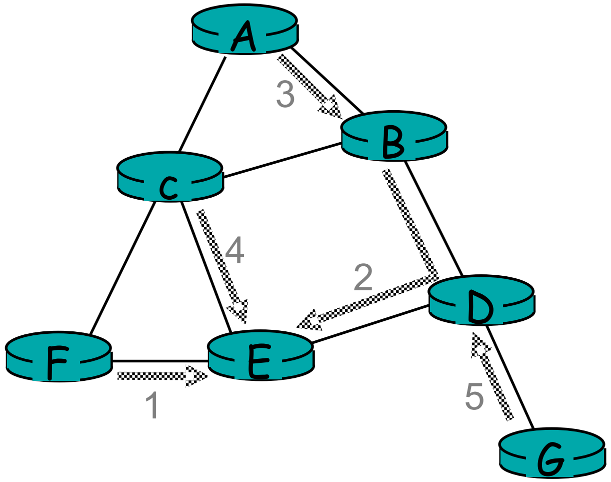


(b) Broadcast initiated at D

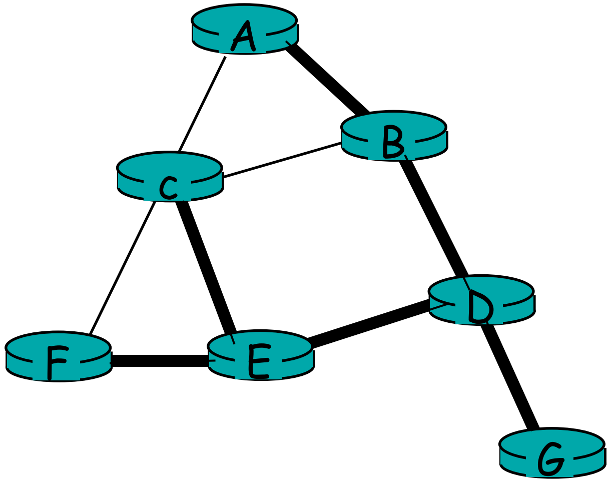
- One spanning tree is sufficient!
  - Edges of tree can be used either way
  - Choice of root is arbitrary (performance differences aside)

Spanning Tree: Creation

- Denominate center node (here: node E)
- Each node that wants to be part of the multicast network: send 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

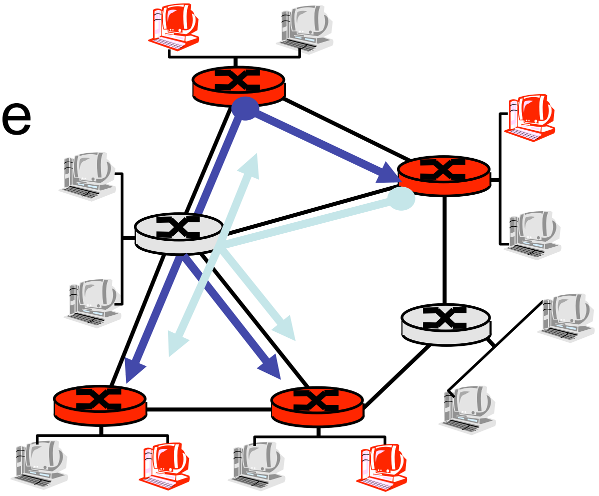
Multicast Routing: Problem Statement

- Multicast = Send one packet to a group of receivers
- ***Do not confuse this with multi-path routing!***
- **Goal:** find a tree (or trees) that connects all routers having local multicast group members
- We first look at basic approaches, then specific protocols adopting these approaches

Approaches for building multicast trees

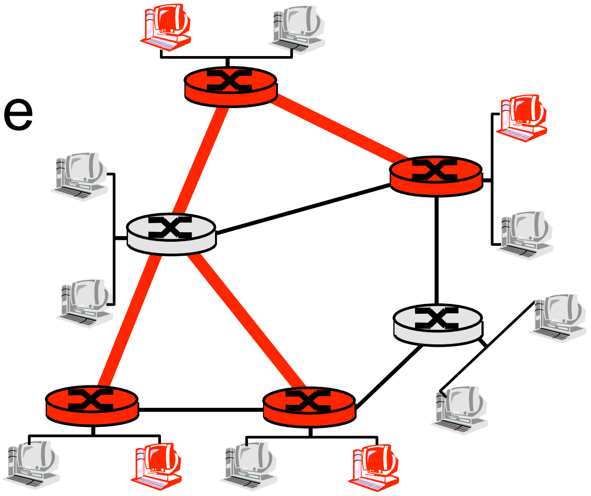
Approaches:

- Source-based tree: one tree per source
  - Shortest path trees
  - Reverse path forwarding



Source-based trees

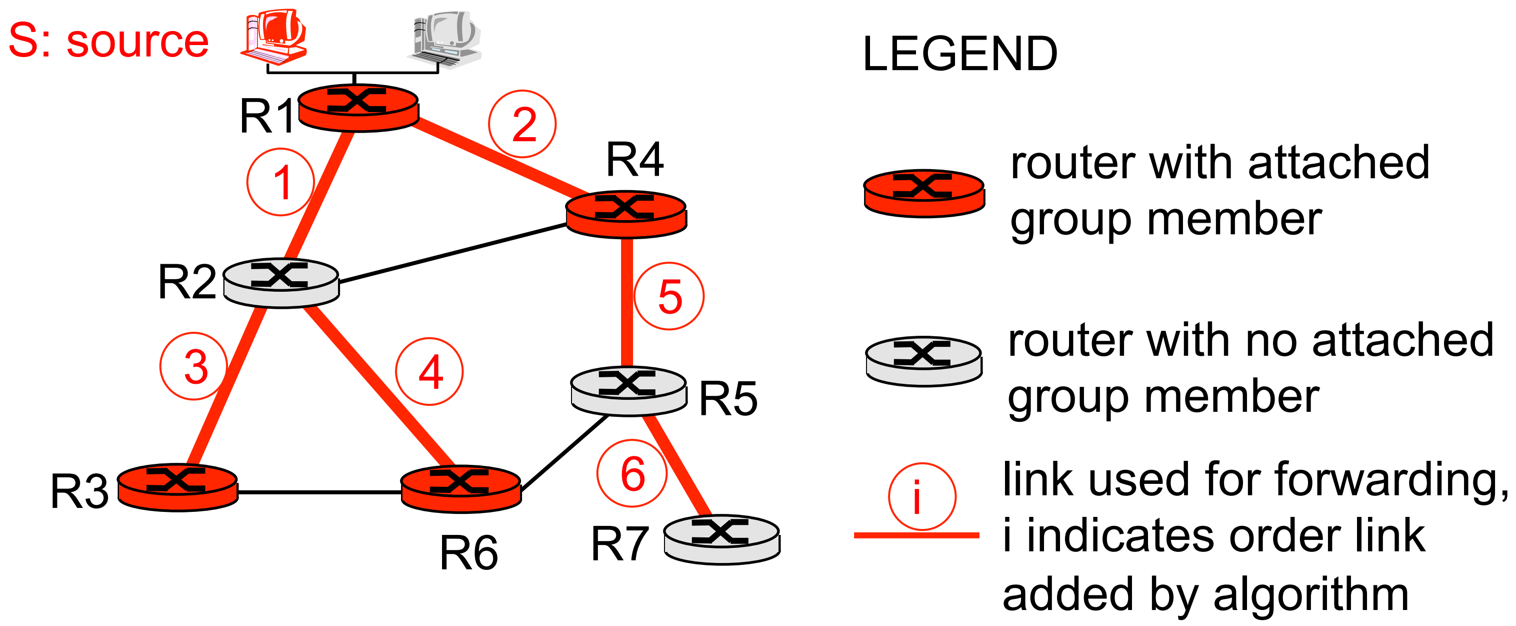
- Group-shared tree: group uses one tree
  - Minimal spanning (Steiner)
  - Center-based trees



Shared tree

Shortest Path Tree

- Multicast forwarding tree: Tree of shortest path routes from source to all receivers
  - e.g., Dijkstra's algorithm



LEGEND

router with attached group member

router with no attached group member

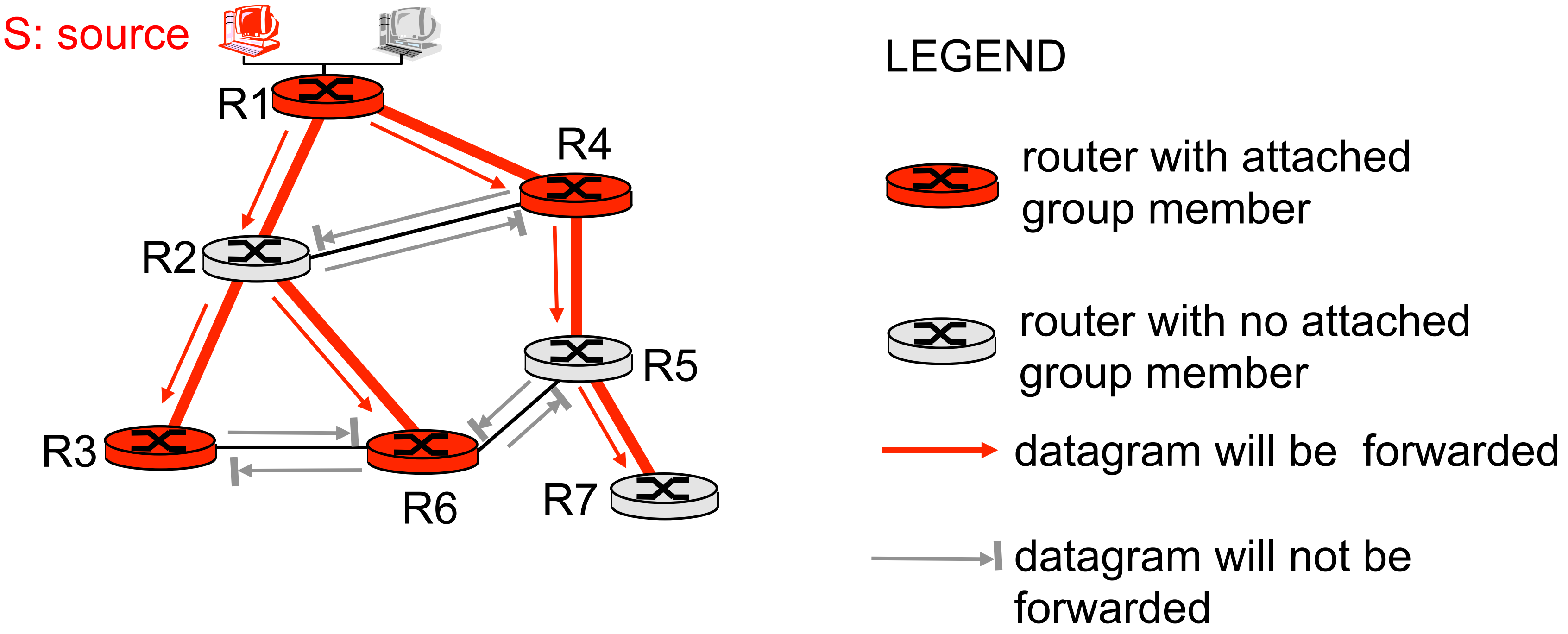
link used for forwarding, i indicates order link added by algorithm

Reverse Path Forwarding

- Observation: When taken together, all shortest paths to the same unicast address form a tree
- Rely on router’ s knowledge of unicast shortest path from itself back to the sender of the multicast packets
- Each router has simple forwarding behaviour:

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

Reverse Path Forwarding: example



LEGEND

router with attached group member

router with no attached group member

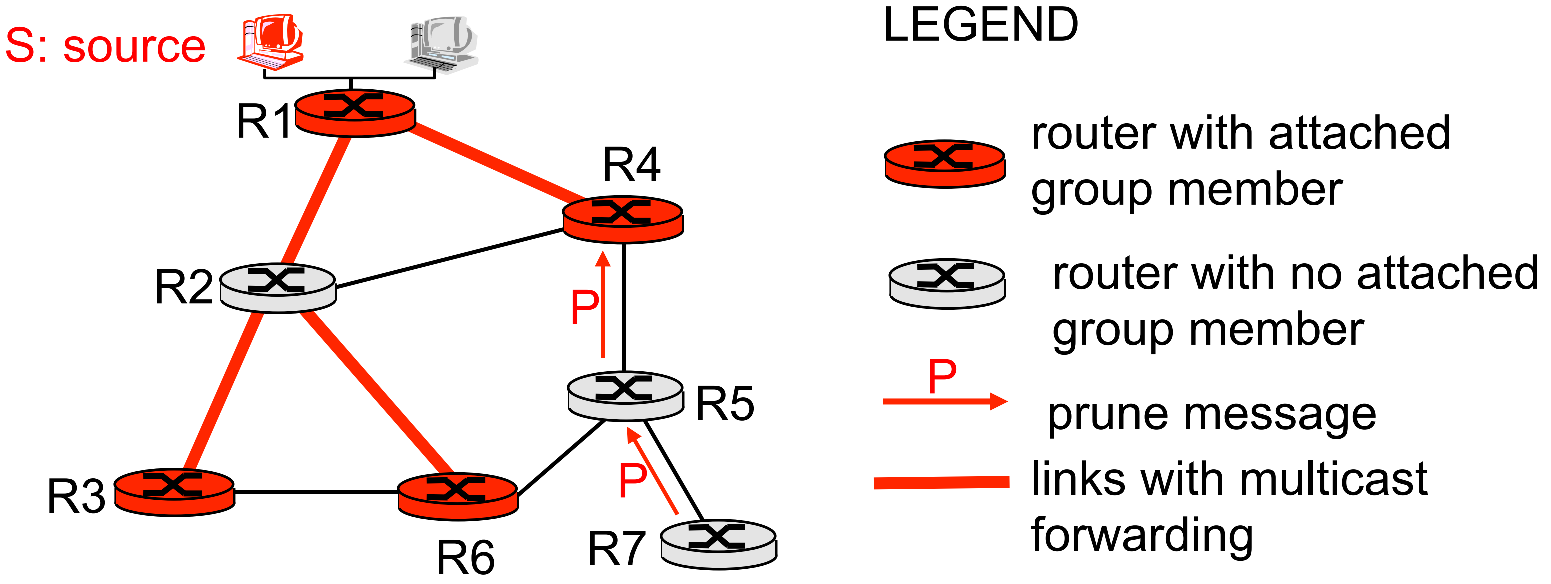
datagram will be forwarded

datagram will not be forwarded

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

Reverse Path Forwarding: pruning

- Forwarding tree contains subtrees with no multicast group members
  - No need to forward datagrams down subtree
  - Send out pruning messages ("I don' t want this traffic" msgs.)
    - Emitted by routers with no downstream group members
    - Sent upstream


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

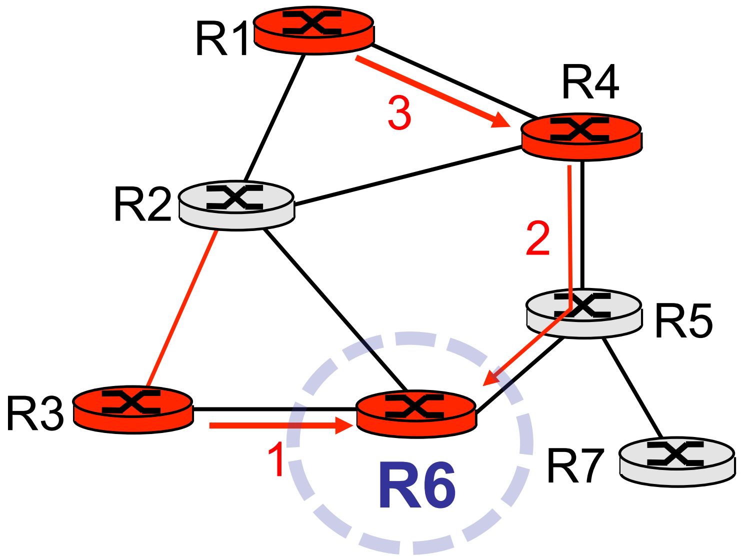
- Steiner Tree: Minimum cost tree connecting all routers with attached group members
  - Popular problem in theoretical computer science
  - Problem is NP-complete
  - Excellent heuristics exist
- But: not used in practice
  - Computational complexity
  - Information about entire network required
  - Monolithic: rerun whenever a router needs to join/leave

Shared-Tree: Center-based trees

- Single delivery tree shared by all
- One router selected as *"center"* of tree (arbitrarily)
- In order to join:
  - Edge router sends unicast *join* message addressed to center router
  - Join message processed by intermediate routers and forwarded towards center
  - Join message either hits existing tree branch for this center, or arrives at center node
  - Path taken by join message 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

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 routing
    - Initial datagram to multicast group flooded everywhere via RPF
    - Routers that receive the packet but don’t want the group: send upstream prune messages

PIM: Protocol Independent Multicast

- Not dependent on any specific underlying unicast routing algorithm (works with all)
- Four modes of operation
- Two modes cover two different multicast distribution scenarios:

Dense:

- Group members densely packed, in “close” proximity
- Bandwidth more plentiful
- Example:
  10,000 receivers within company LAN

Sparse:

- # networks with group members small in relation to
  # interconnected networks
- Group members “widely dispersed”
- Bandwidth not plentiful
- Example:
  10,000 receivers world-wide spread across various ASes

Consequences of Sparse—Dense Dichotomy:

Dense mode

- Group membership by routers *assumed* until routers explicitly prune
- *Data-driven* construction on multicast tree (e.g., RPF)
- Bandwidth and non-group-router processing *waste resources*

Sparse mode

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

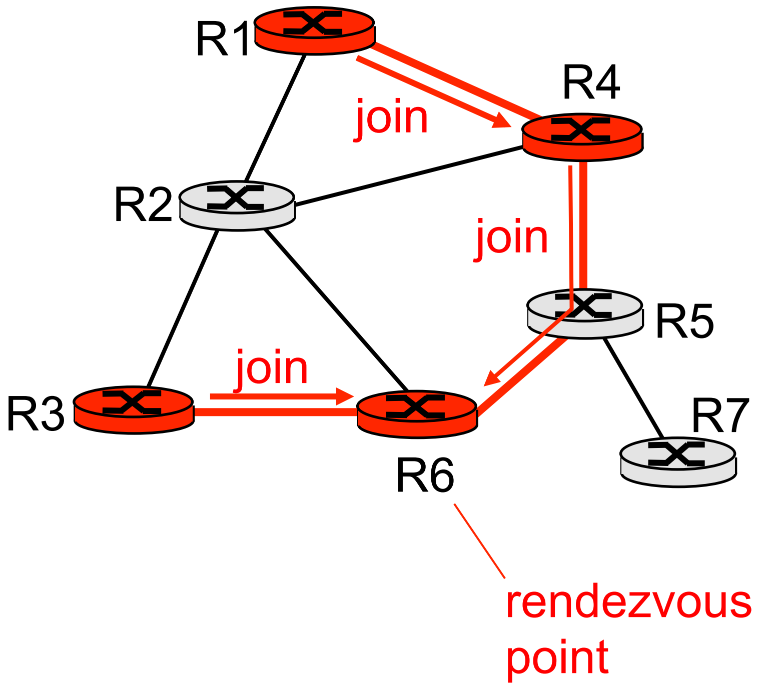
PIM – Dense Mode

Flood-and-prune RPF

- ❑ Underlying unicast protocol provides RPF info for incoming datagram
- ❑ Broadcast incoming packets along all RPF links
- ❑ Send pruning message if undesired traffic is received
- ❑ Has protocol mechanism for router to detect whether it is a leaf-node router

PIM – Sparse Mode

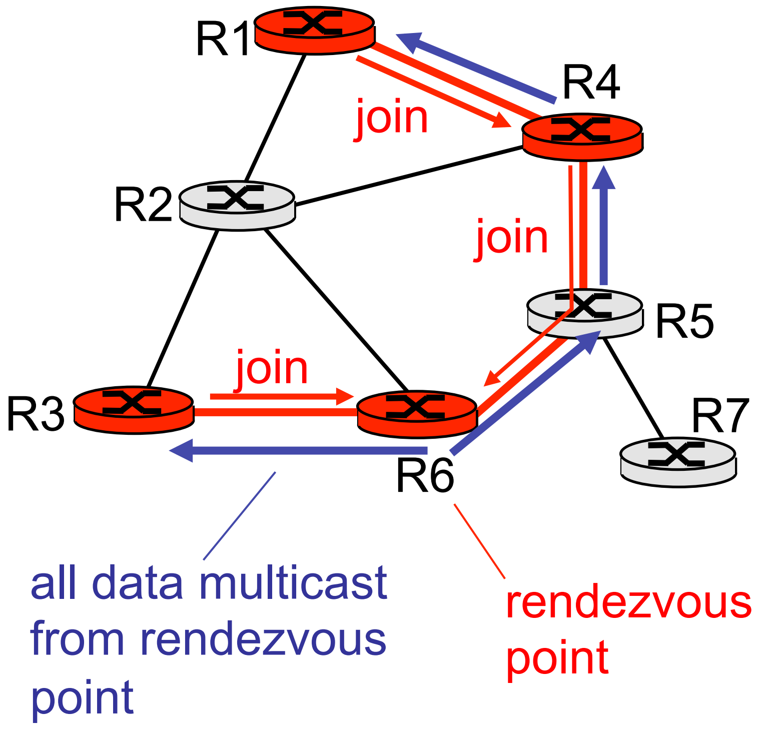
- Center-based approach
- Router sends *join* message towards rendezvous point (RP)
  - Intermediate routers update state and forward *join*



PIM – Sparse Mode

Sender(s):

- Send data to RP via unicast
- RP multicasts data down the RP-rooted tree
- RP can extend multicast tree upstream to source
- RP can send *stop* message if no attached receivers (pruning)



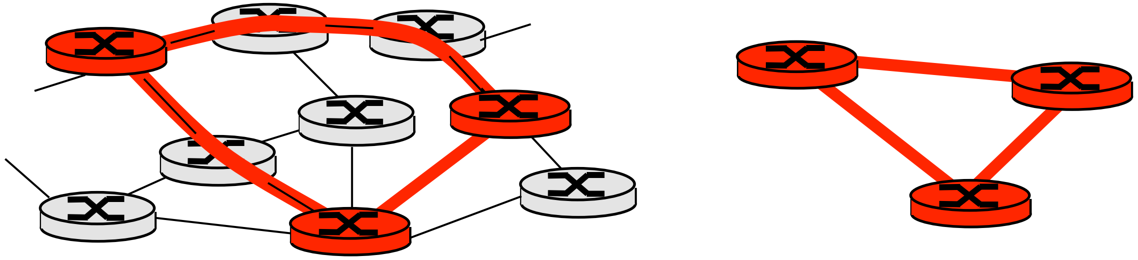
PIM – other modes

- ❑ Source-specific multicast
  - ▪ Not one shared tree for the group, but per-source trees
  - ▪ Increased performance:
    1. Uses more links ➔ less congestion
    2. Shorter paths
       – Theoretical computer science: Think about the difference between paths created by Prim's algorithm and paths created by Dijkstra's algorithm

- ❑ Bidirectional PIM
  - ▪ Group shared tree
  - ▪ Treat all links as bidirectional
  - ▪ Scales better: no source-specific state, but one state per group

Tunneling

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



physical topology

logical topology

- Multicast datagram encapsulated inside "normal" (i.e., non-multicast-addressed) datagram
- Normal IP datagram sent via regular IP unicast to receiving multicast router: "Tunnel"
- Receiving multicast router unwraps original multicast datagram

- Rather universal, versatile trick – it's called *overlay network*

IP/Routing: Weaknesses and shortcomings (1)

- No network congestion control:
  Dynamic routing / dynamic traffic engineering = difficult!
  - Tried out in ARPANET: Oscillations everywhere
  - Today: Interaction with TCP congestion control feedback loop → even worse!
- Convergence speed (link/router failures)
  - OSPF: 200ms … several seconds
    - Routing loops may occur during convergence = black holes
  - BGP: seconds to several minutes!
    - Many timers (MRAI, route flap damping,…), prefix aggregation
    - Never really converges: there’s always something going on
- More and more prefixes in routing tables of Tier-1 core routers
  - 300,000 and growing

IP/Routing: Weaknesses and Shortcomings (2)

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

Chapter:
Transport Layer

Chapter: Transport Layer

Our goals:

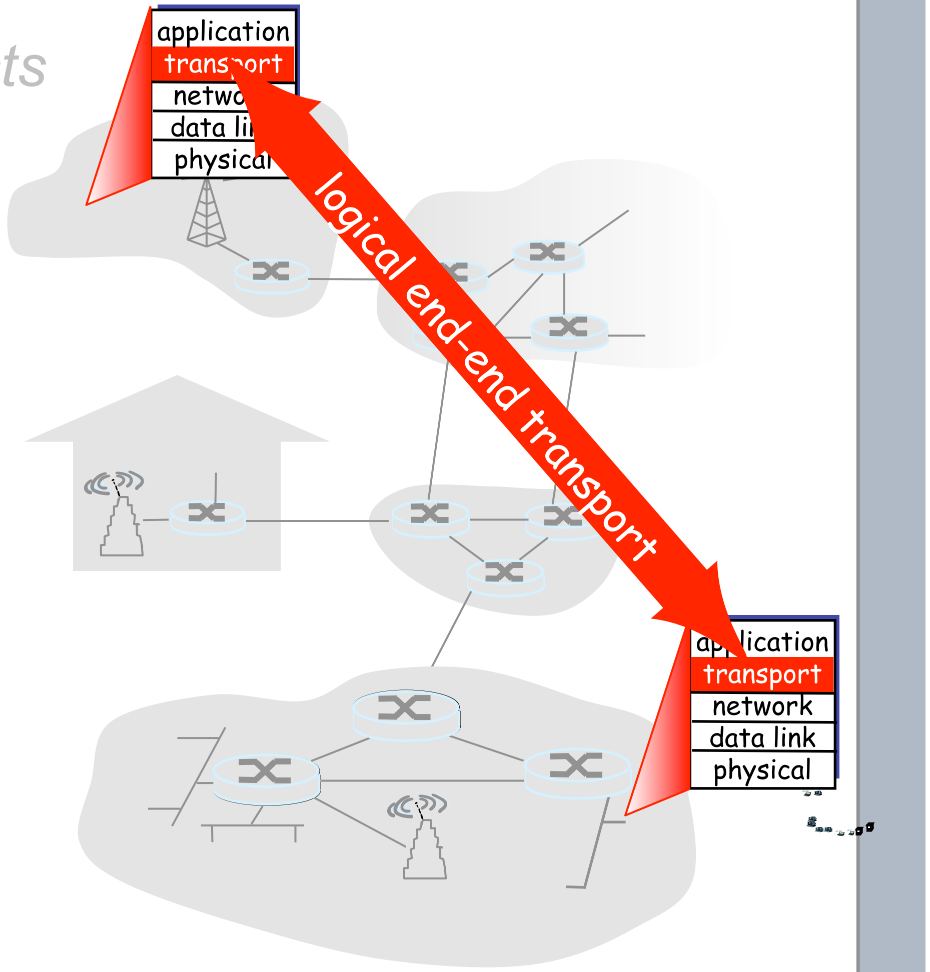
- Understand *principles* behind transport layer services:
  - multiplexing/demultiplexing
  - reliable data transfer
  - flow control
  - congestion control
- Learn about transport layer *protocols* in the Internet:
  - UDP: connectionless transport
  - TCP: connection-oriented transport
    - TCP congestion control
  - (Maybe: SCTP, if time permits)

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

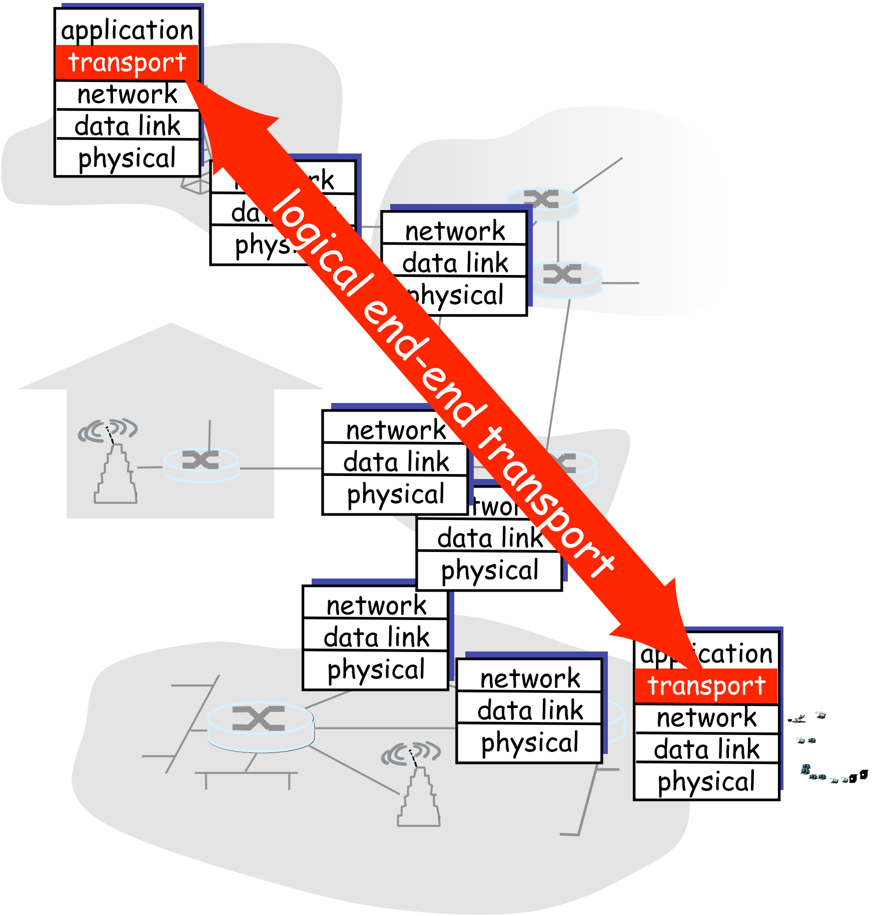
Transport services and protocols

- Provide logical communication *between application processes* running on different hosts
  - ↔Network layer: *between hosts*
- Transport protocols run in end systems
  - Sender side: breaks app messages into segments, passes to network layer
  - Rcver side: reassembles segments into messages, passes to app layer
- More than one transport protocol available to apps
  - Internet: mainly TCP, 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



Multiplexing/demultiplexing

Socket: File handle that allows to send/receive network traffic

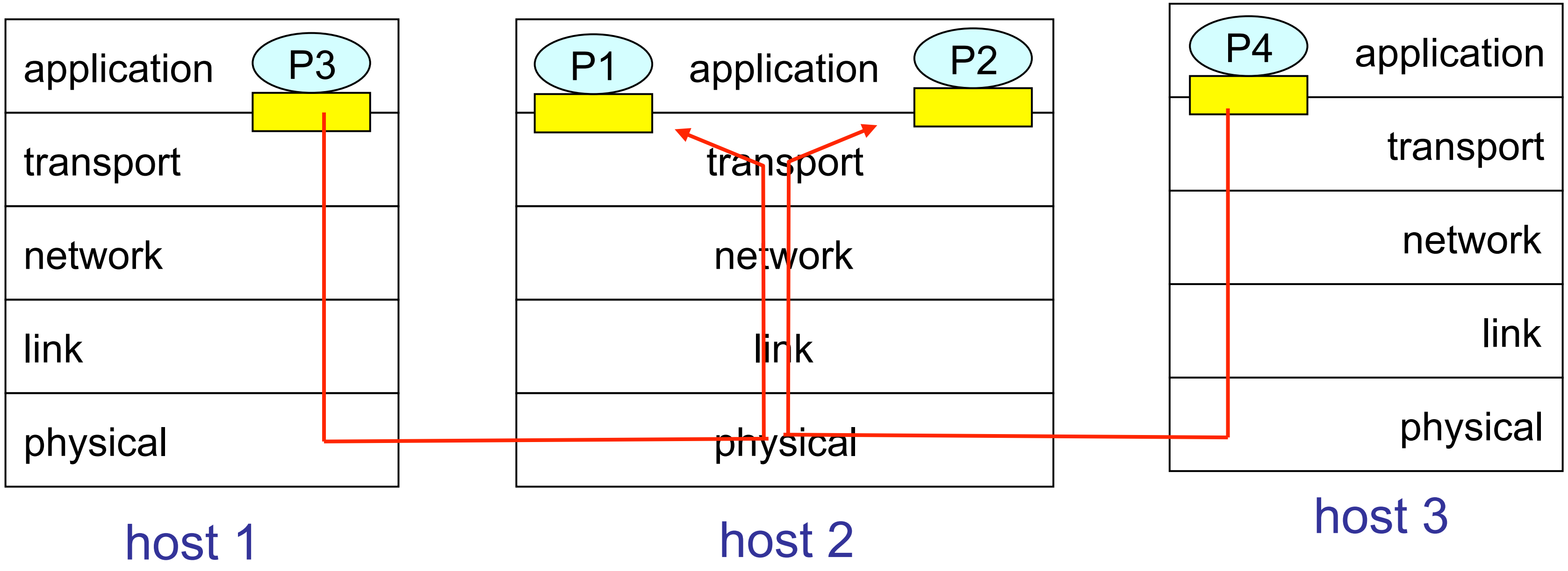
Demultiplexing at rcv host:

Delivering received segments to correct socket

Multiplexing at send host:

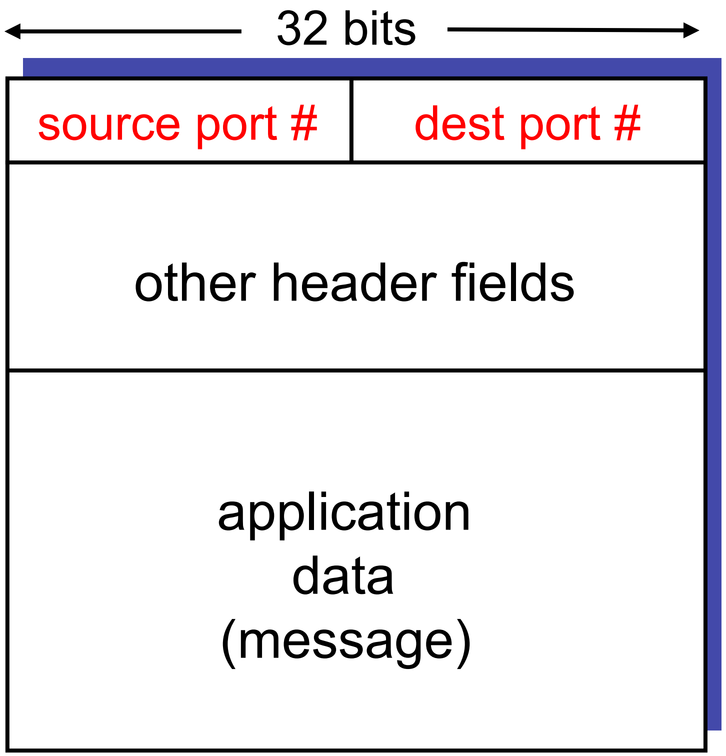
Gathering data from multiple sockets, enveloping data with header (later used for demultiplexing)

🟨 = 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 *and* port numbers to direct segment to appropriate socket



TCP/UDP segment format

Connectionless demultiplexing (UDP)

- Create sockets with port numbers (in Java):

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

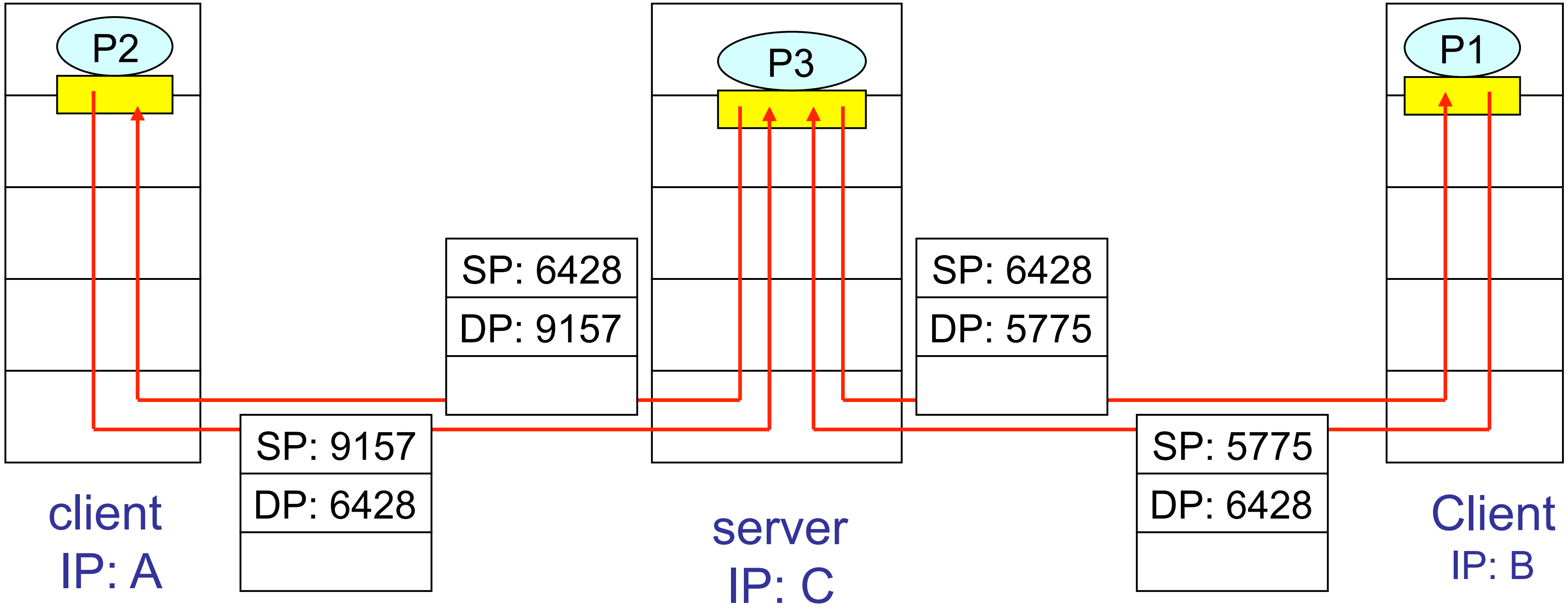
- UDP socket identified by two-tuple:

  (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
  - Receiving process cannot easily distinguish differing communication partners on same socket

Connectionless demux (cont)

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

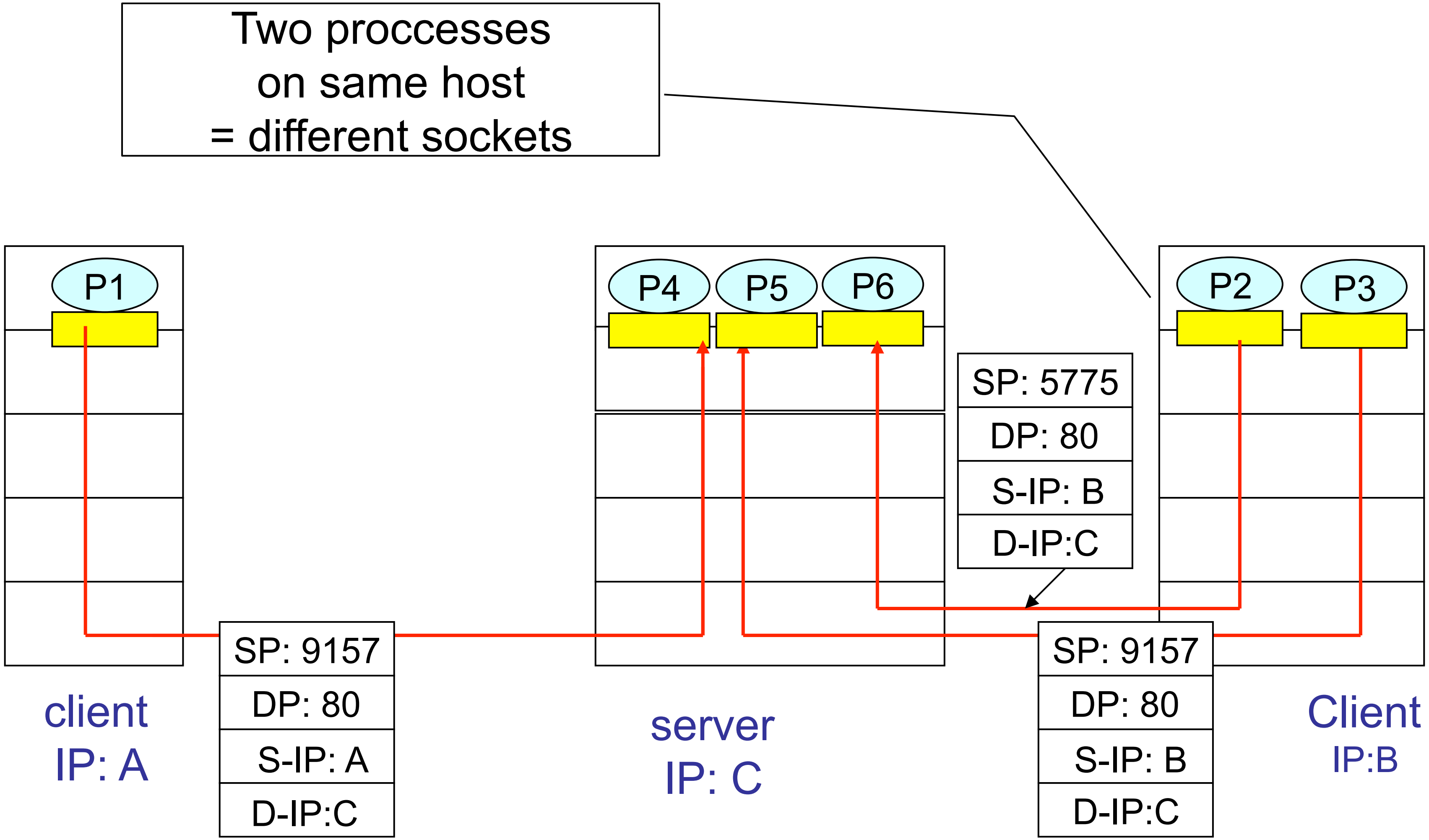


Source Port (SP) provides “return address”

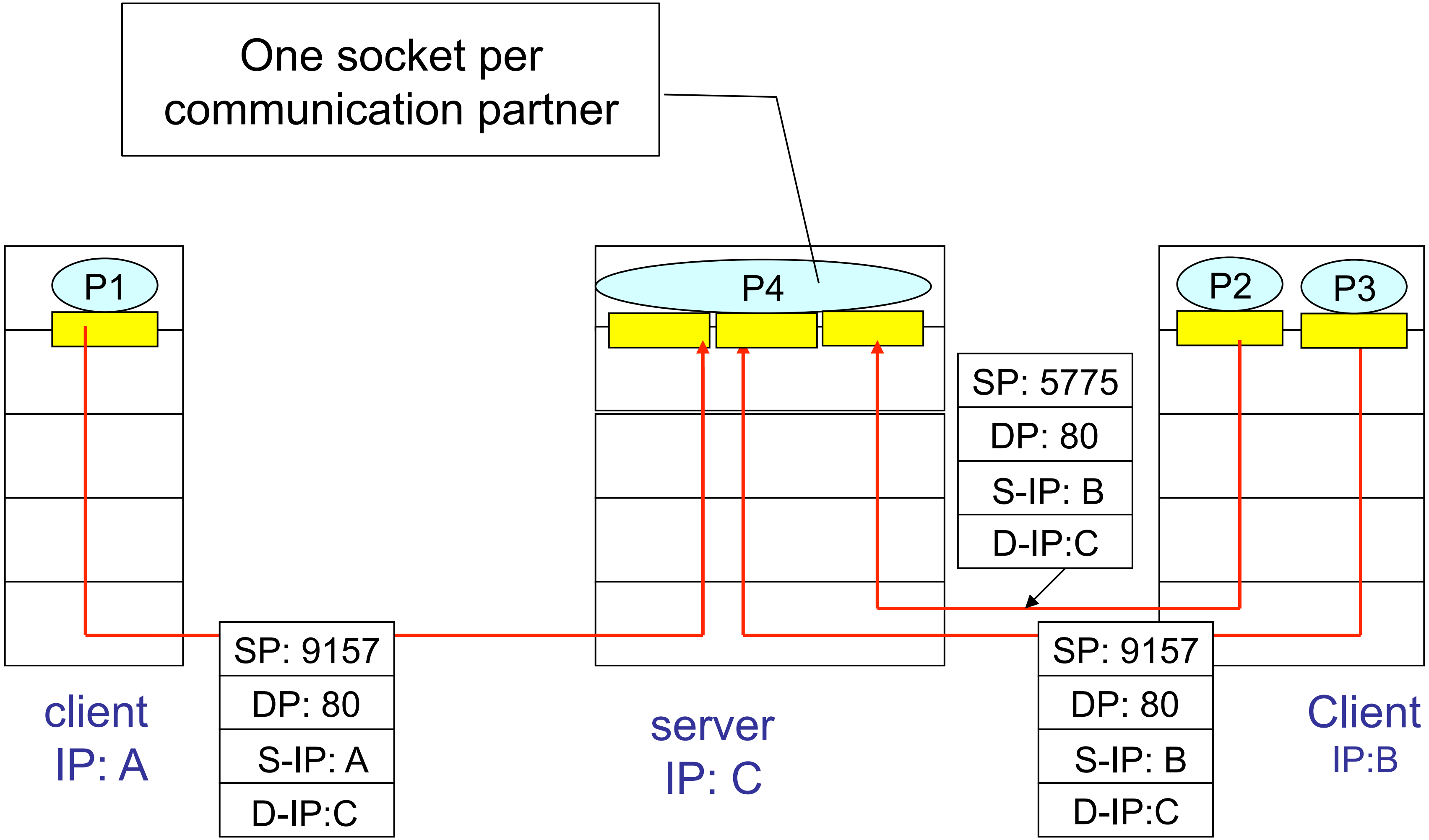
Connection-oriented demux (TCP)

- TCP socket identified by 4-tuple:
  - Source IP address
  - Source port number
  - Destination IP address
  - Destination port number
- Receiving 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
- Example:
  Web servers have different sockets for each connecting client
  - Non-persistent HTTP will even have different socket for each request

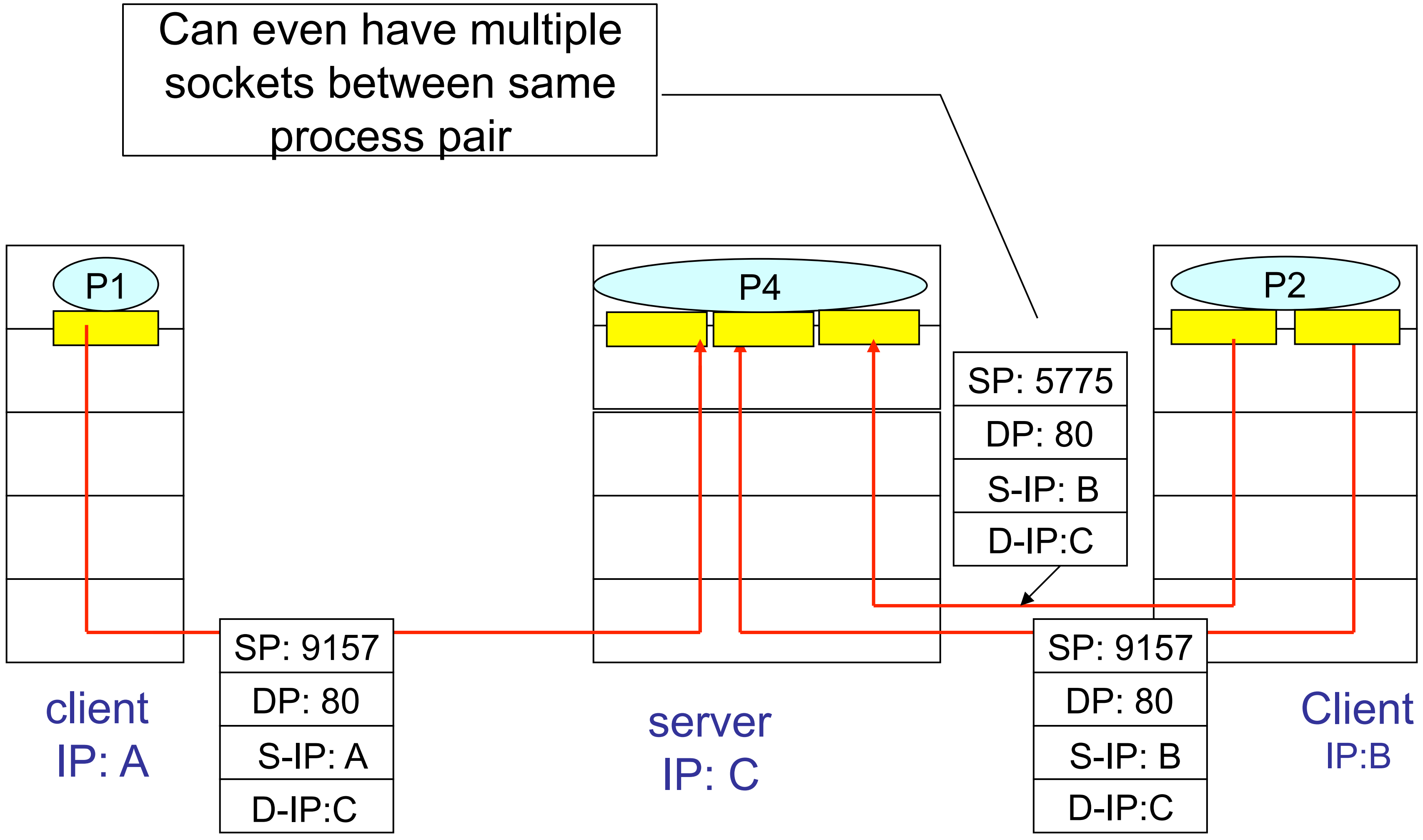
Connection-oriented demux (cont)



Connection-oriented demux: Threaded Web Server



Connection-oriented demux: Fast client



Can even have multiple sockets between same process pair

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

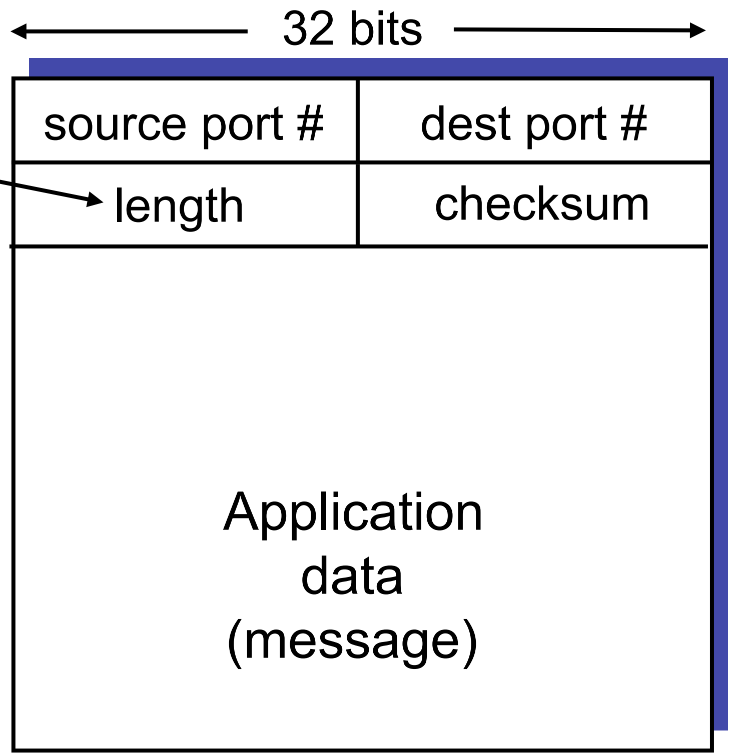
Why is there a UDP?

- No connection establishment (which can add delay)
- Simple: no connection state at sender, at 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
  - SIP
- Reliable transfer over UDP:
  - Add reliability at application layer → application-specific error recovery!

Length, in bytes of UDP segment, including header



32 bits

source port #	dest port #
length	checksum
Application data (message)	

UDP segment format

UDP checksum

Goal: Detect TX 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. Drop segment.
  - YES → no error detected. *But maybe errors nonetheless?* More later ….

Internet Checksum Example

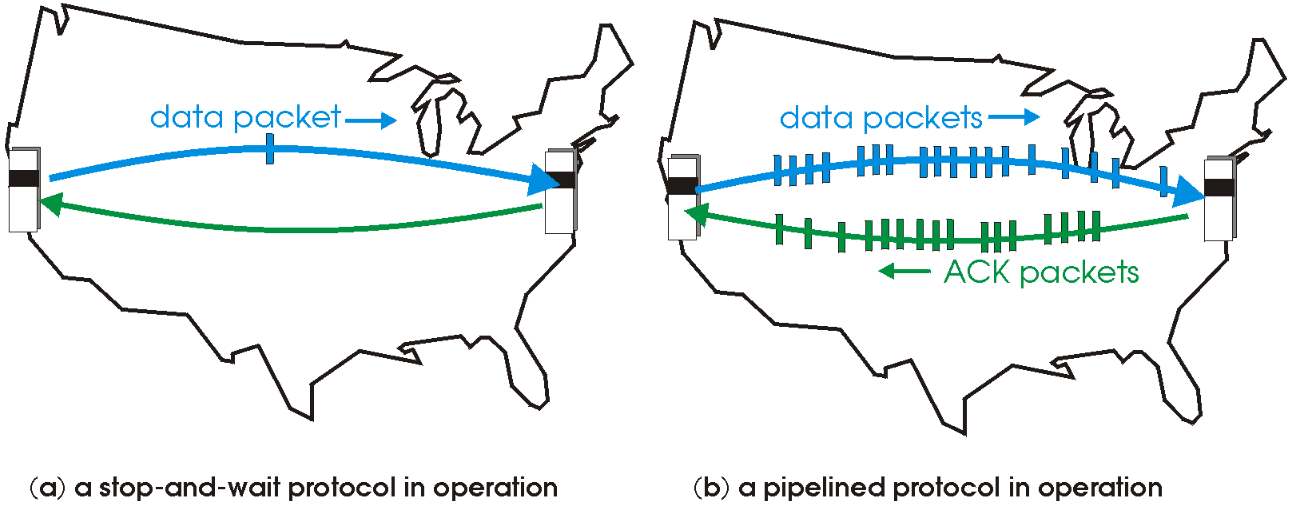
- Note
  - When adding numbers, a carryout from the most significant bit needs to be added to the result
- Example: add two 16-bit integers

		1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
		1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
wrap around	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
sum		1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum (=inverse)		0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

Pipelined protocols

Pipelining: Sender allows multiple, “in-flight”, yet-to-be-acknowledged packets

- Range of sequence numbers must be large enough
- Buffering at sender and/or receiver

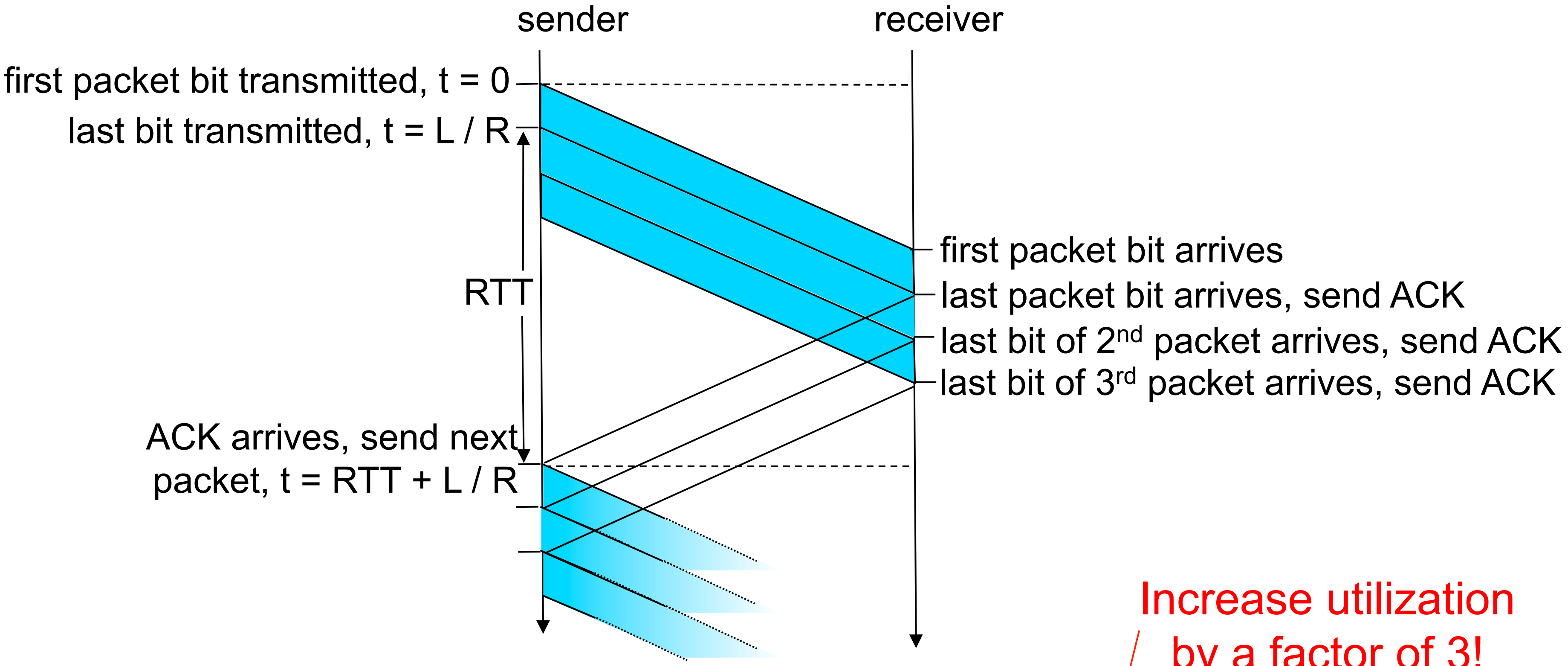


(a) a stop-and-wait protocol in operation
(b) a pipelined protocol in operation

❑ 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
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

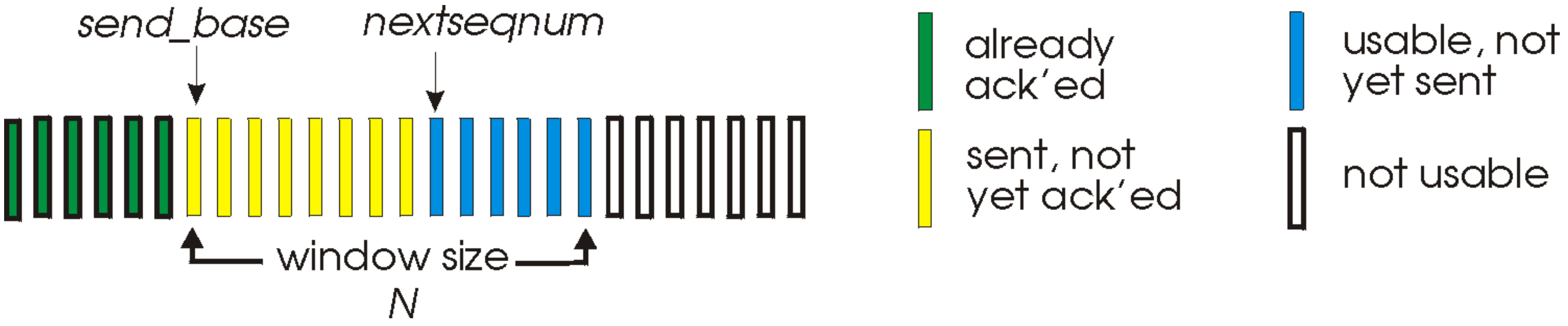
Increase utilization by a factor of 3!

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

Go-Back-N

Sender:

- k-bit sequence number in packet header
- “window” of up to N, consecutive unack’ed packets allowed

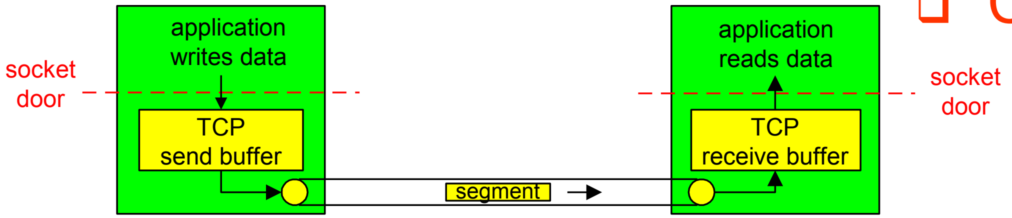


- ACK(n): acknowledges all packets up to and including packet seq# n – “cumulative ACK”
  - May receive duplicate ACKs (see receiver)
- Timer for each in-flight packet
- *Timeout(n):* retransmit pkt n and all higher seq # pkts in window

TCP: Overview

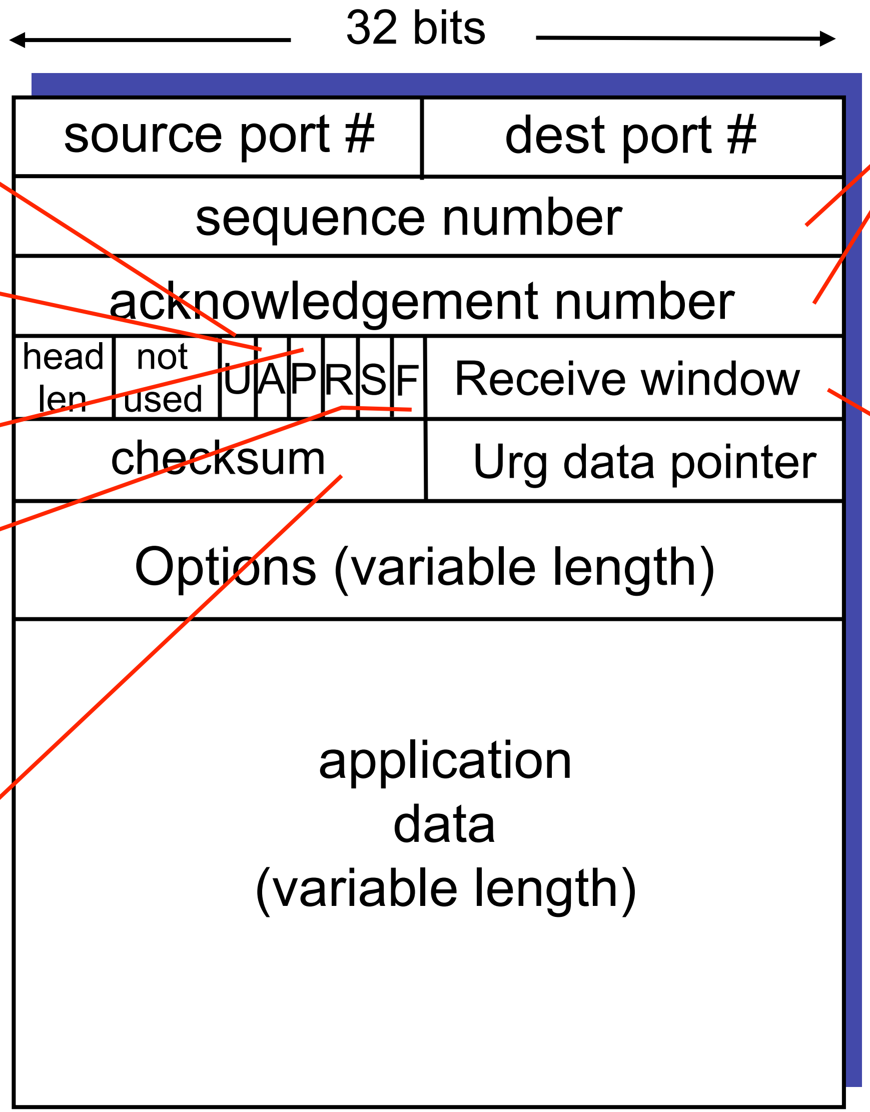
RFCs: 793, 1122, 1323, 2018, 2581

- 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) initialises sender & receiver state before data exchange
- Flow controlled:
  - Sender will not overwhelm receiver
- Congestion controlled:
  - Sender 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 pointer	
Options (variable length)								
application data (variable length)								

URG: urgent data (generally not used)

ACK: ACK # valid

PSH: push data now (used, but generally ignored)

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

Internet checksum (as in UDP)

counting *by bytes of data* (not segments!)

# bytes rcvr is willing to accept

TCP sequence numbers and ACKs

Sequence numbers:

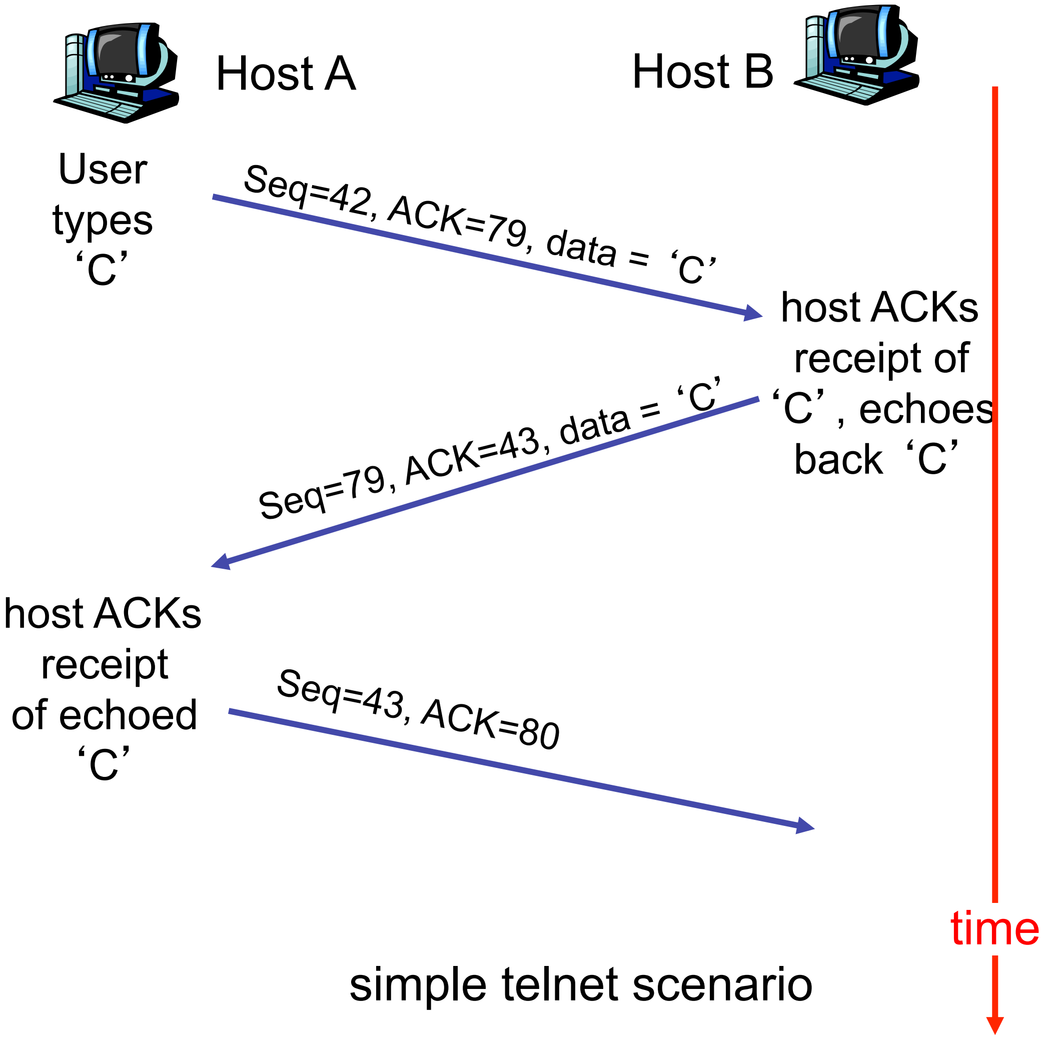
- Byte stream "number" of first byte in segment's data
- Start value not 0, but chosen arbitrarily

ACKs:

- Seq # of next byte expected from other side
- Cumulative ACK

Q: How should receiver handle out-of-order segments?

- 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

TCP Round Trip Time (RTT) and Timeout

Q: How to set TCP timeout value for detecting lost packets?

- Obviously: Longer than RTT
  - but RTT varies
- Too short:
  - premature timeout
  - unnecessary retransmissions
- Too long:
  - slow reaction to segment loss

Q: How to estimate RTT?

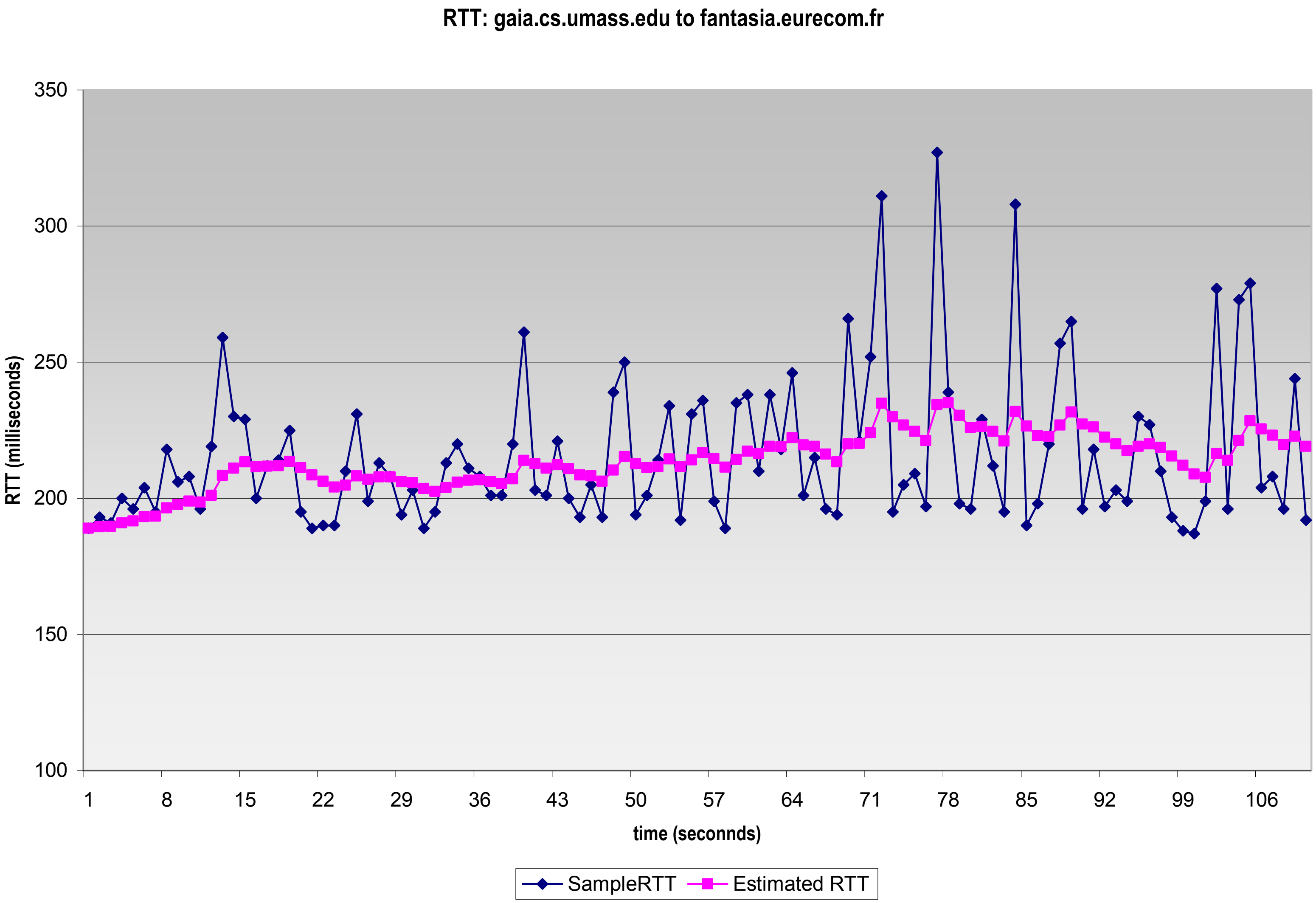
- `SampleRTT`: measured time from segment transmission until ACK receipt
  - Ignore retransmissions (why?)
- `SampleRTT` will vary, want estimated RTT "smoother"
  - Average several recent measurements, not just current `SampleRTT`
  - Exponential moving average (EMA)

TCP Round Trip Time and Timeout

$$\texttt{EstimatedRTT} = (1 - \alpha)*\texttt{EstimatedRTT} + \alpha*\texttt{SampleRTT}$$

- ❑ Exponential weighted moving average (EMA)
- ❑ Influence of past sample decreases exponentially fast
- ❑ Typical value: $\alpha = 0.125$

Example RTT estimation:



TCP Round Trip Time and Timeout

Setting the timeout

- **EstimtedRTT** plus “safety margin”
  - Small variation in **EstimatedRTT** → smaller 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 reliable data transfer

- TCP creates reliable data transfer 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, let’s consider simplified TCP sender:
  - Ignore duplicate acks
  - Ignore flow control, congestion control

TCP sender events:

Data received from application:

- 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`

When timeout occurs:

- Retransmit segment that caused timeout
- Restart timer

When ack received:

- *If* it acknowledges previously un-acked segments
  - Update what is known to be acked
  - Stop timer for this data
  - (Re)start timer if there are other outstanding segments

TCP sender (simplified)

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

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

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

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

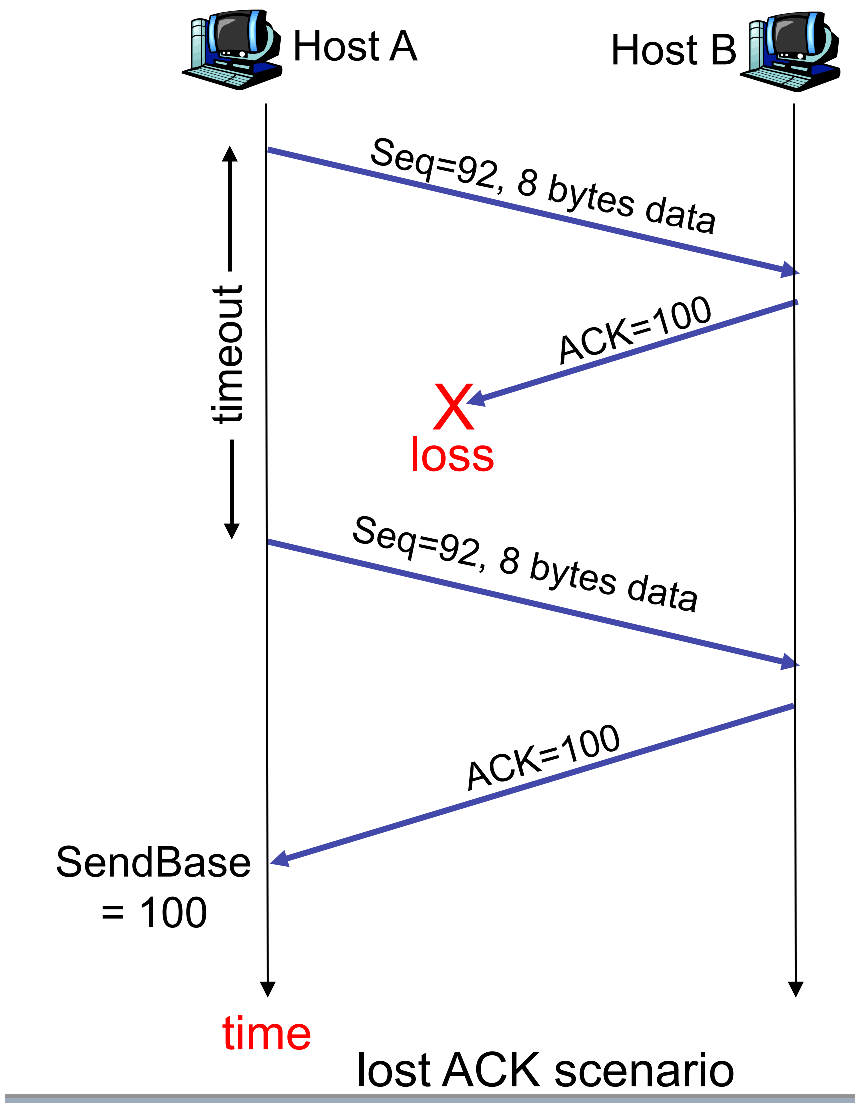
Comment:

- SendBase-1: last cumulatively ack’ed byte

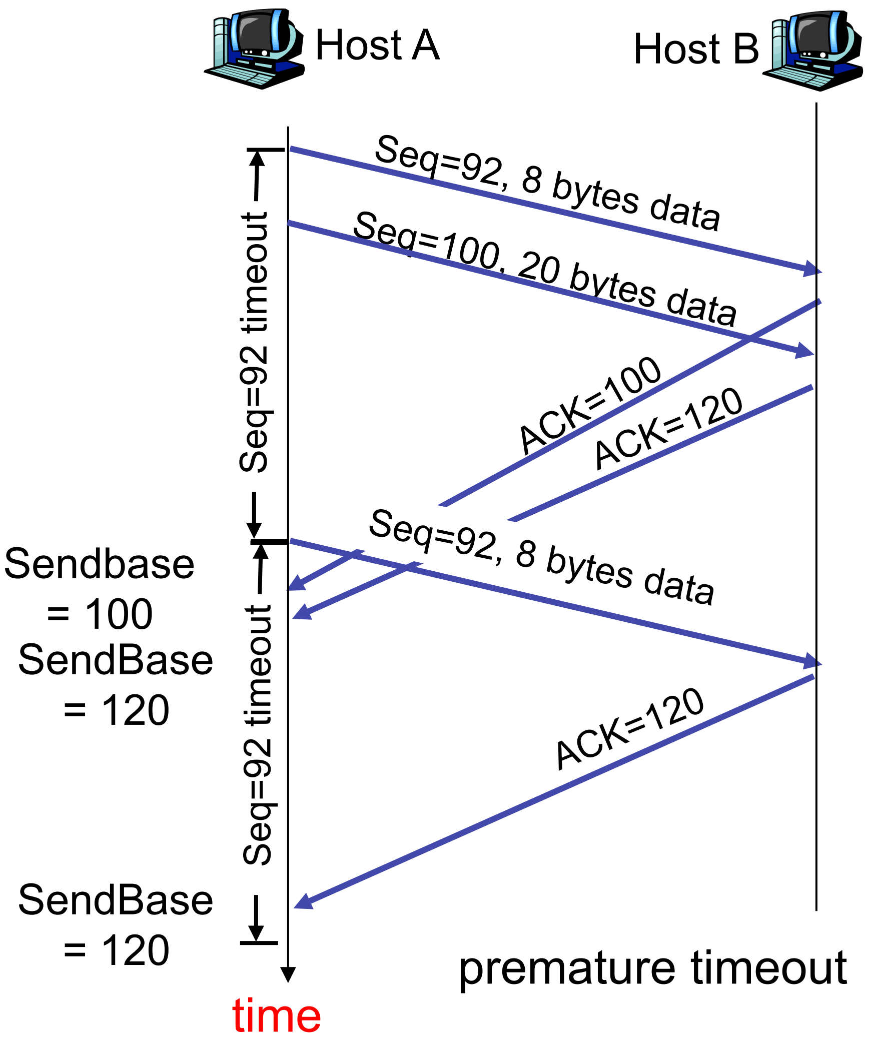
Example:

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

TCP: Retransmission scenarios

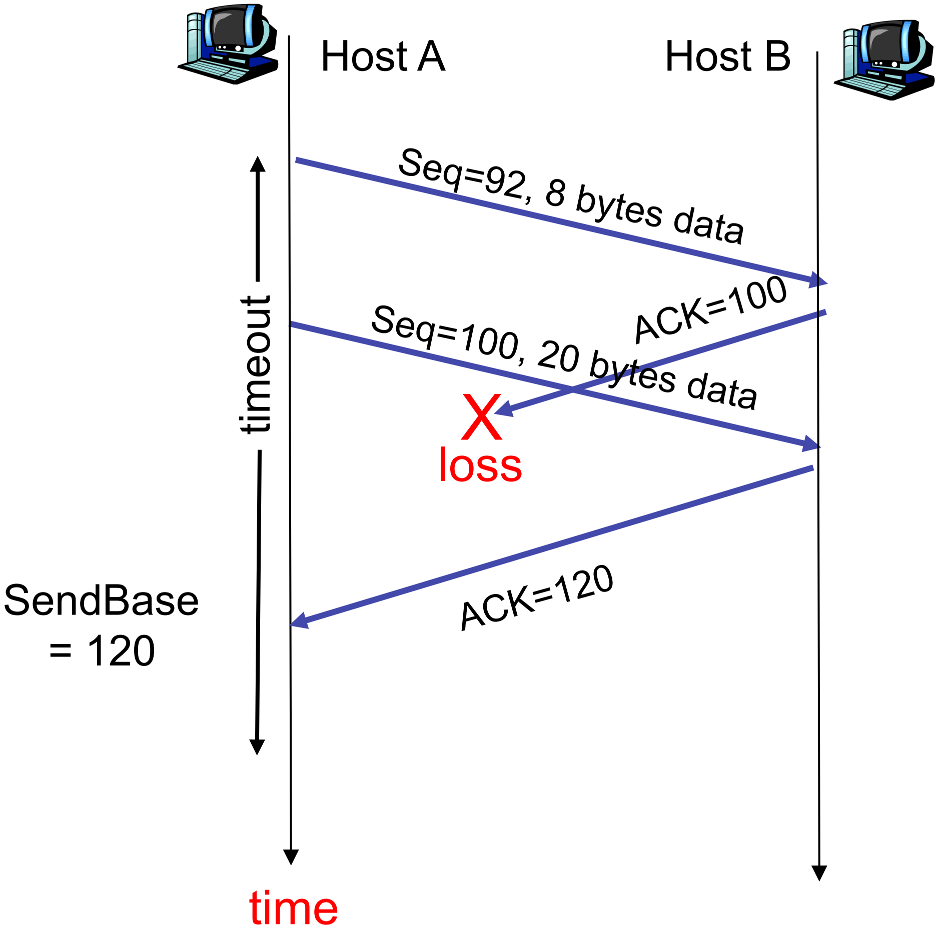


lost ACK scenario



premature timeout

TCP retransmission scenarios (more)



Cumulative ACK scenario

Retransmit of Seq# 92?
Or no retransmit?

No retransmit: We have cumulative ACKs!

TCP ACK generation [RFC 1122, RFC 2581]

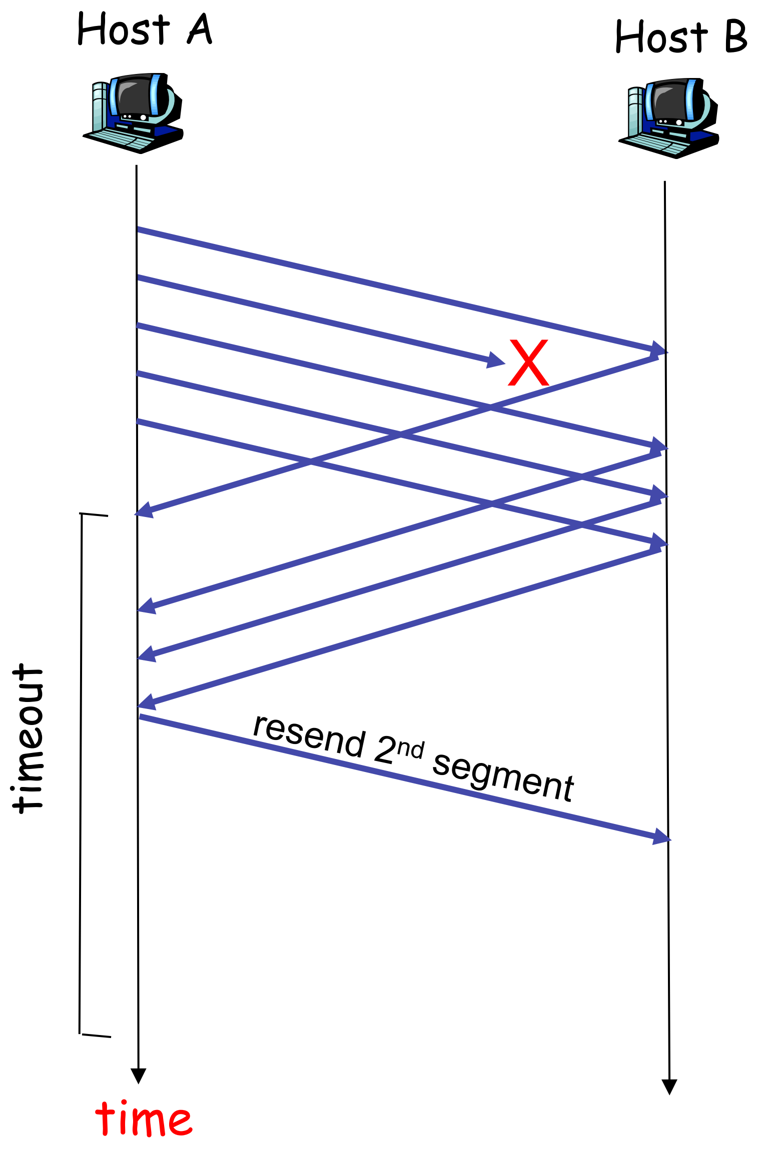
Event at Receiver	TCP Receiver action
Arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	Delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK
Arrival of in-order segment with expected seq #. One other segment has ACK pending	Immediately 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

A small TCP optimisation: Fast Retransmit

- Time-out period often relatively long:
  - Long delay before resending lost packet
- Can 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
    - Assume that only one segment was lost

Resending a segment after triple duplicate ACK



Fast retransmit algorithm:

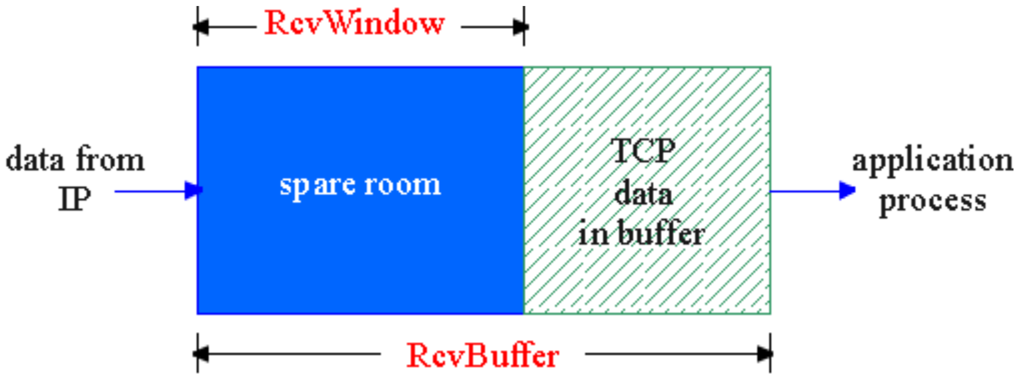
```
event: ACK received, with ACK field value of y
          if (y > SendBase) {
              SendBase = y
              if (there are currently not-yet-acknowledged segments)
                  start timer
            }
          else {
               increment count of dup ACKs received for y
    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

a duplicate ACK for already ACKed segment

fast retransmit

TCP Flow Control

- Receive side of TCP connection has a receive buffer:

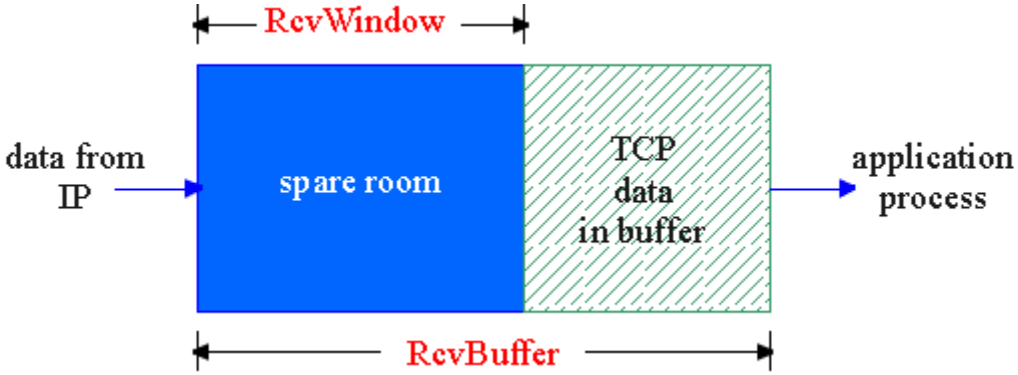


flow control

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

- Application process may be slow at reading from buffer (e.g., mobile phone)
- Speed-matching service: matching the send rate to the receiving application'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]`**

- ❑ Receiver advertises spare room by including value of **`RcvWindow`** in segments
- ❑ Sender limits unACKed data to **`RcvWindow`**
  - ▪ guarantees receive buffer doesn’t overflow

TCP Connection Management

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

- Initialize TCP variables:
  - Sequence numbers
  - 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();
```

Note: Cannot distinguish client and server after connection establishment

Three way handshake:

Step 1: client host sends TCP SYN segment to server

- i.e., SYN bit is set
- Specifies initial seq #
- No data

Step 2: server host receives SYN, replies with SYNACK segment

- i.e., SYN and ACK bits set
- 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:

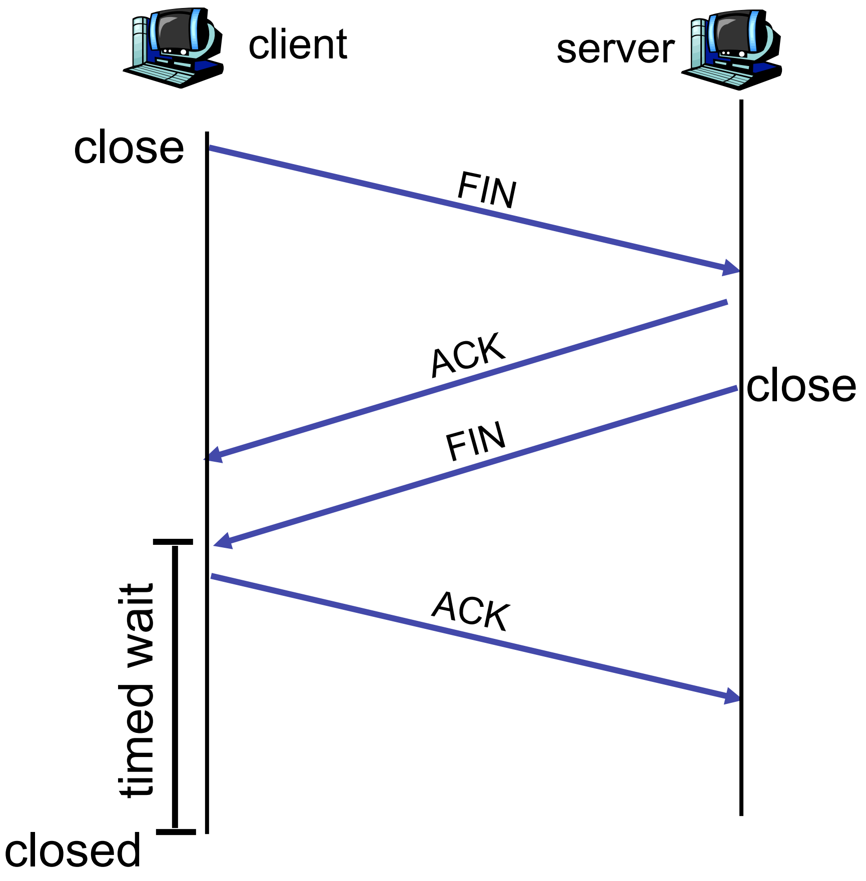
“Client” closes socket:
clientSocket.close();

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

- ❑ Promise: “I won’t transmit any further data to you”: Half-closed connection

Step 2: Server receives FIN, replies with ACK. Informs application. Application closes connection, TCP sends FIN.

Note: Server can continue sending data between step 1 and Step 2!



TCP Connection Management (cont.)

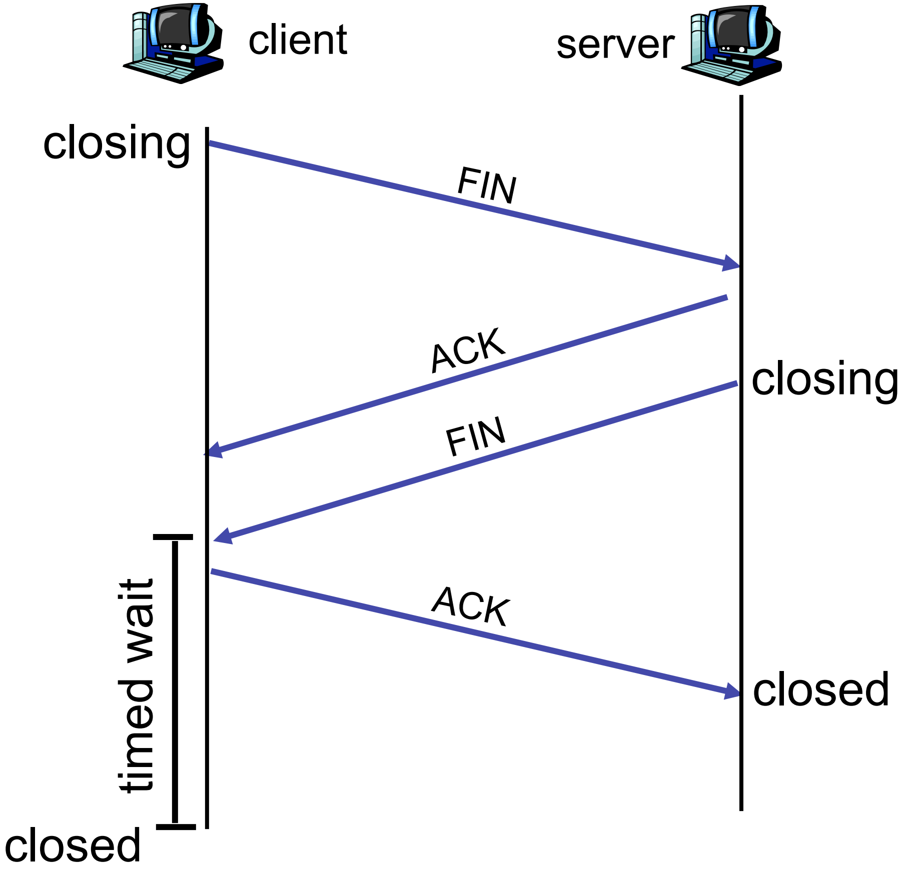
Step 3: client receives FIN, replies with ACK.

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

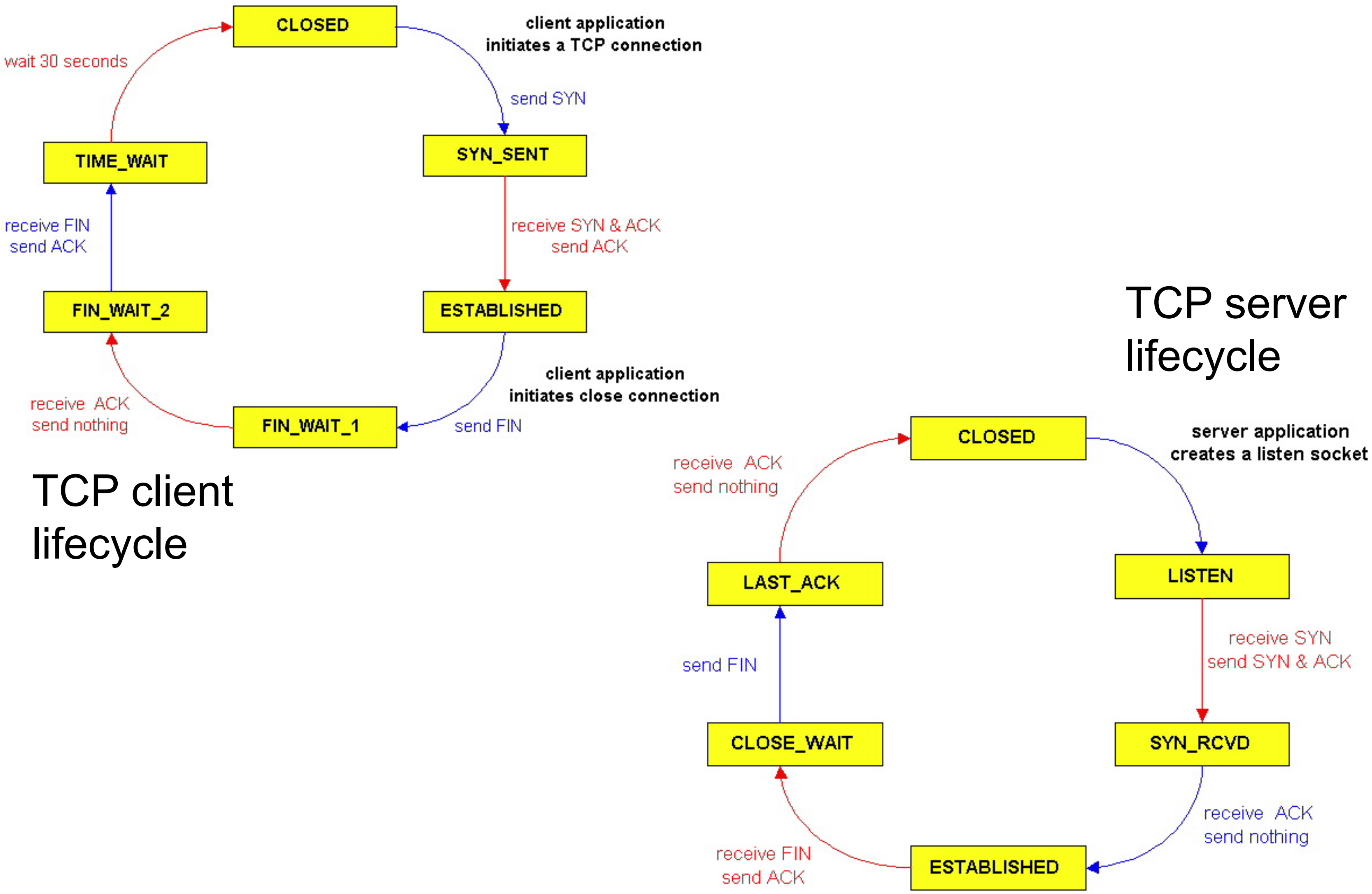
Step 4: server, receives ACK. Connection closed.

Notes:

- With small modification, can handle simultaneous FINs
- Any partner in connection can send the first FIN



TCP Connection Management (cont)



TCP client lifecycle

TCP server lifecycle

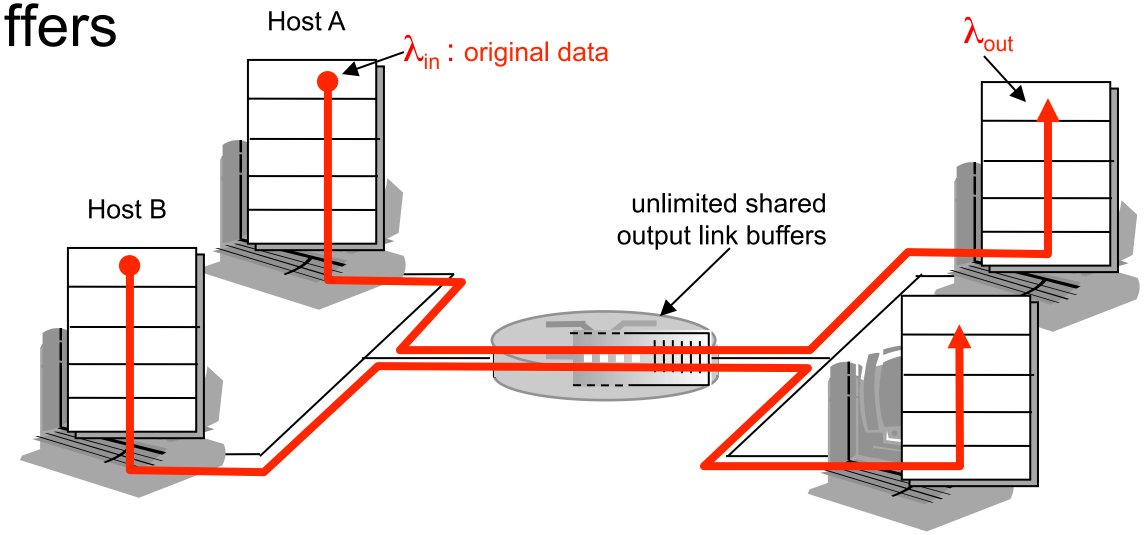
Principles of Congestion Control

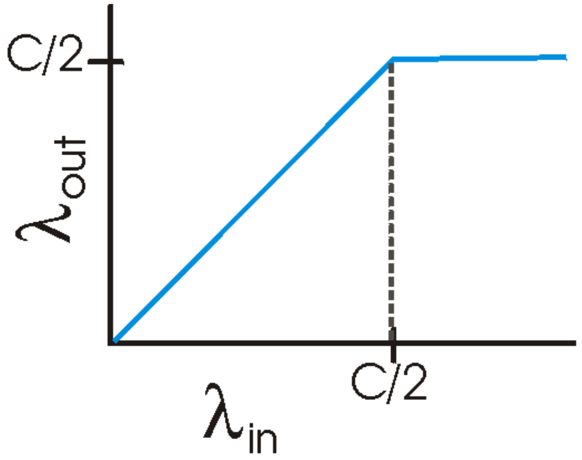
Congestion:

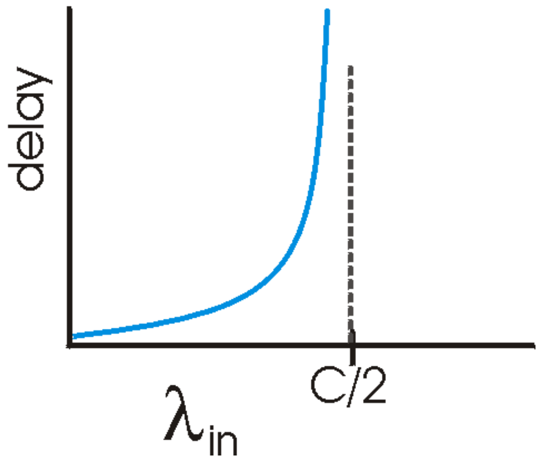
- Informally: "Too many sources sending too much data too fast for the *network* to handle"
- What' s the difference to flow control?
  - Flow control: "One source sending too much data too fast for the *other application* to handle"
- Manifestations:
  - Lost packets (buffer overflow at routers)
  - 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

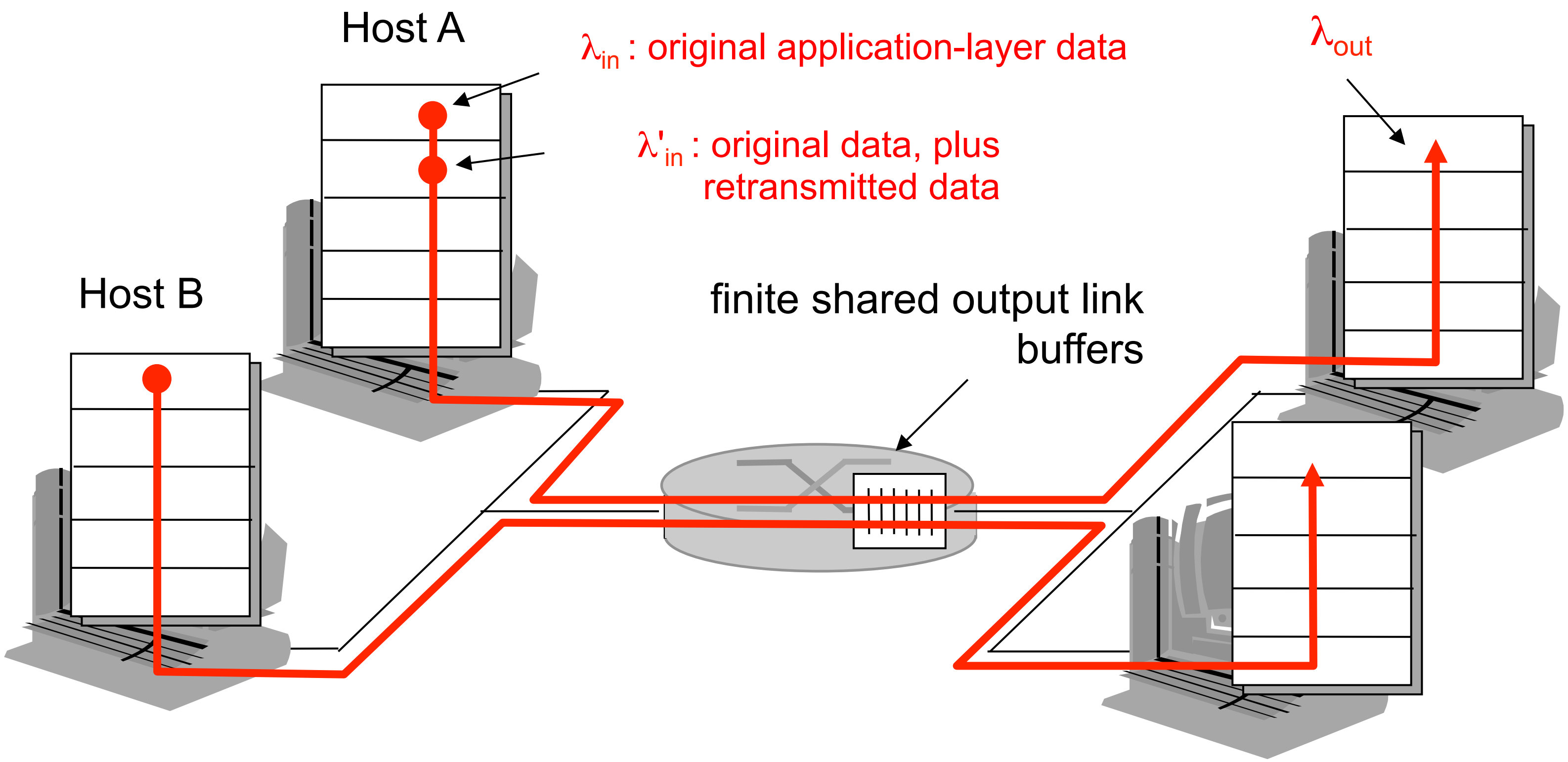




- Large delays when congested
- Maximum achievable throughput

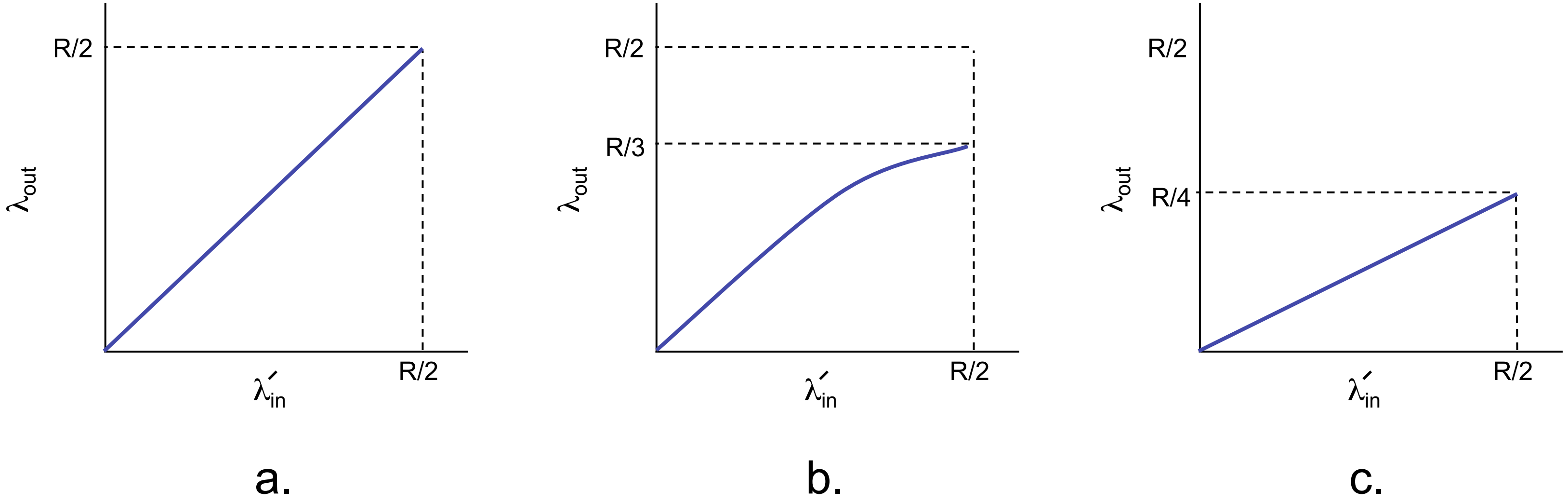
Causes/costs of congestion: scenario 2

- One router, *finite* buffers
- Sender retransmission of lost packet


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

Causes/costs of congestion: scenario 2

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



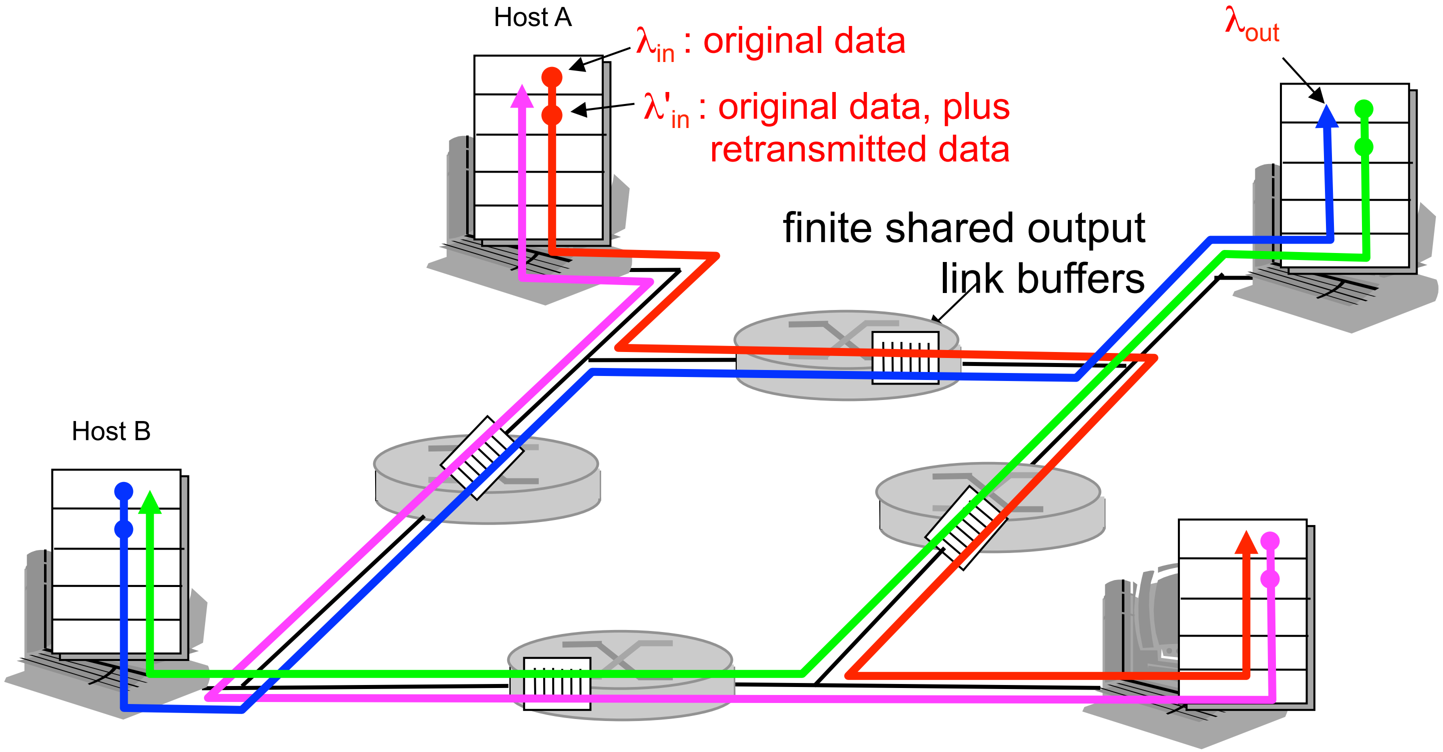
"Costs" of congestion:

- More work (retransmissions) for given "goodput"
- Unnecessary retransmissions: Link carries multiple copies of same packet

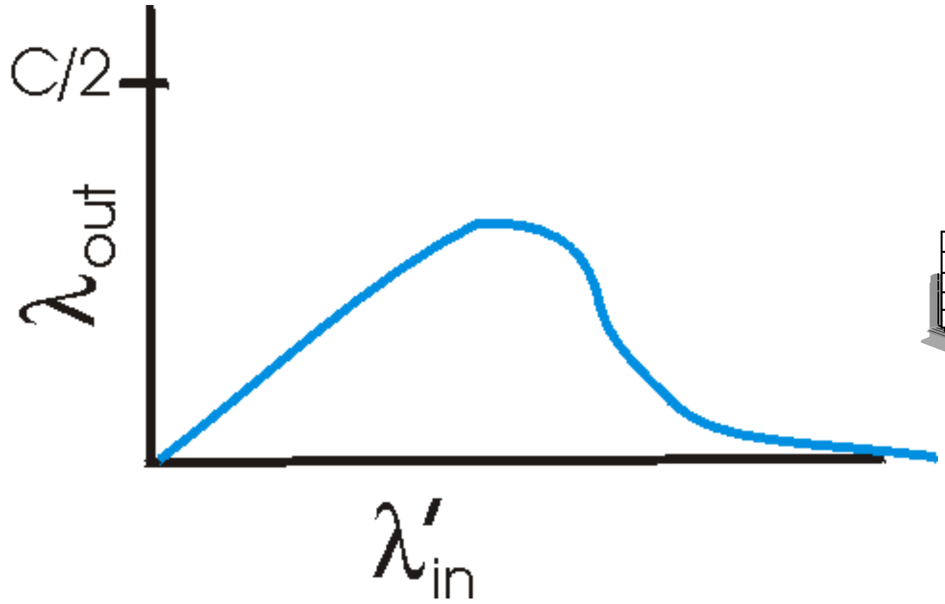
Causes/costs of congestion: scenario 3

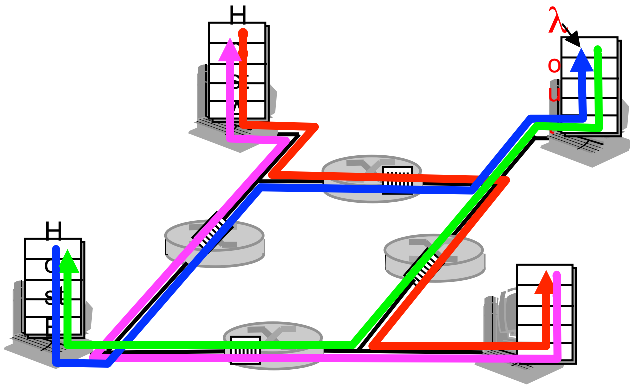
- ❑ Four senders
- ❑ Multihop paths
- ❑ Timeout/retransmit

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


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

Causes/costs of congestion: scenario 3





Another "cost" of congestion:

- When packet is 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 bit, ICMP source quench ATM)
  - Explicit rate sender should send at
- TCP/IP has support for ECN, but almost never used
- ICMP source quench: dito

Case study: ATM ABR congestion control

ABR: available bit rate:

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

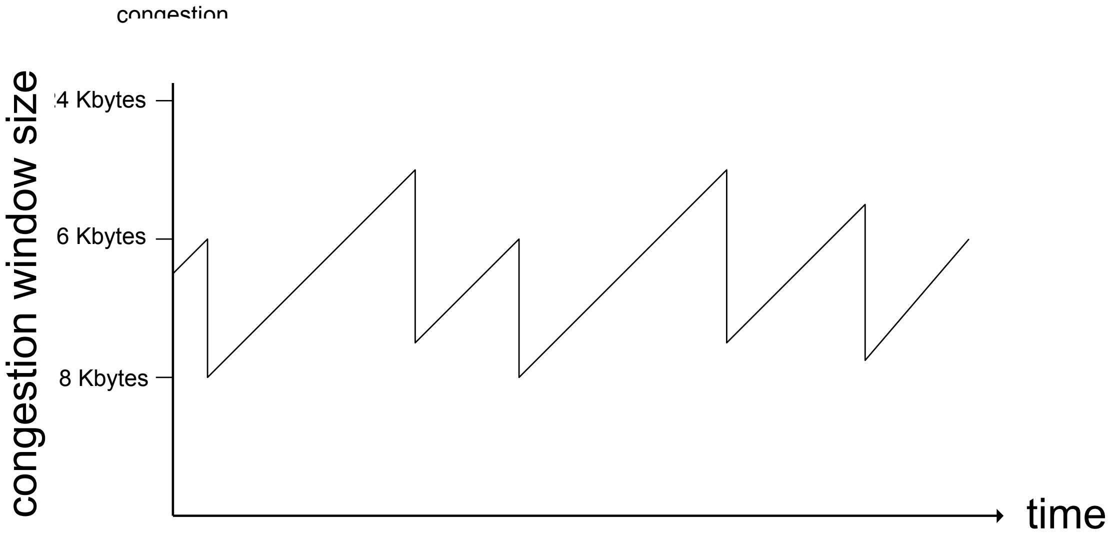
RM (resource management) cells:

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

TCP congestion control: Additive increase, Multiplicative decrease (AIMD)

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

Saw tooth behavior: probing for bandwidth



TCP Congestion Control: details

- Sender limits transmission:

LastByteSent – LastByteAcked ≤ CongWin

- Roughly,

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

- **CongWin** is dynamic: Function of perceived network congestion

How does sender perceive congestion?

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

Three mechanisms:

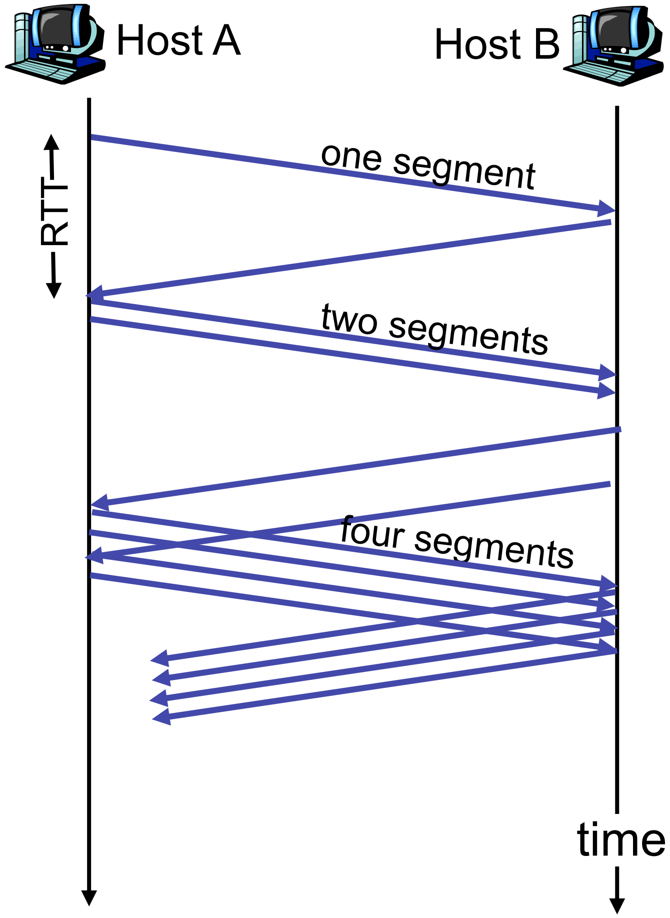
- AIMD
- Slow start
- conservative after timeout events

TCP Slow Start

- When connection begins, `CongWin` = 1 MSS
  - Example: MSS = 500 bytes; RTT = 200 msec
  - Initial rate = 20 kbps
- But: 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
  - N.B.: Exponential growth caused by additions, not multiplications or exponentiations!
- Summary: Initial rate is slow but ramps up exponentially fast



Refinement: Inferring loss

- After 3 duplicate 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:

Why this distincion?

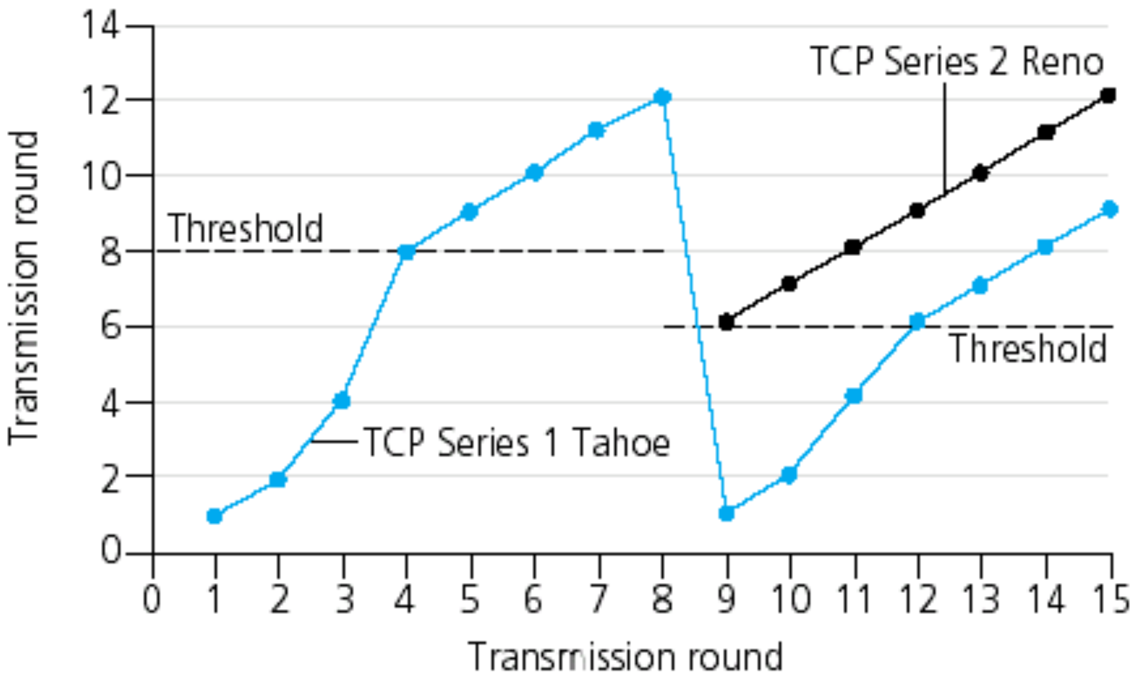
- 3 duplicate ACKs indicates: Network still capable of delivering some (actually, most) segments
- Timeout indicates a more alarming congestion scenario: (Almost) no segments got through!

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



Summary: TCP Congestion Control

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

TCP sender congestion control

State	Event	TCP Sender Action	Commentary
Slow Start (SS)	ACK receipt for previously unacked data	CongWin = CongWin + MSS, If (CongWin > Threshold) set state to “Congestion Avoidance”	Resulting in a doubling of CongWin every RTT
Congestion Avoidance (CA)	ACK receipt for previously unacked data	CongWin = CongWin+MSS * (MSS/ CongWin)	Additive increase, resulting in increase of CongWin by 1 MSS every RTT

SS or CA	Loss event detected by triple duplicate ACK	Threshold = CongWin/2,
		CongWin = Threshold,
		Set state to “Congestion Avoidance”
			Fast recovery, implementing multiplicative decrease. CongWin will not drop below 1 MSS.

SS or CA	Timeout	Threshold = CongWin/2,
		CongWin = 1 MSS,
		Set state to “Slow Start”
			Enter slow start

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

TCP summary

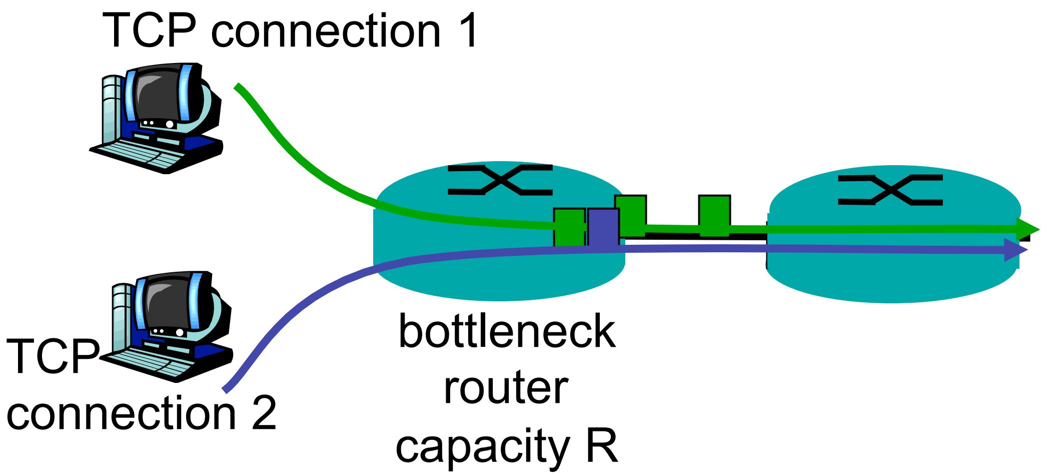
- Connection-oriented: SYN, SYNACK; FIN
- Retransmit lost packets; in-order data: sequence no., ACK no.
- ACKs: either piggybacked, or no-data pure ACK packets if no data travelling in other direction
- Don't overload receiver: rwin
  - rwin advertised by receiver
- Don't overload network: cwin
  - cwin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

TCP throughput

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

TCP Fairness

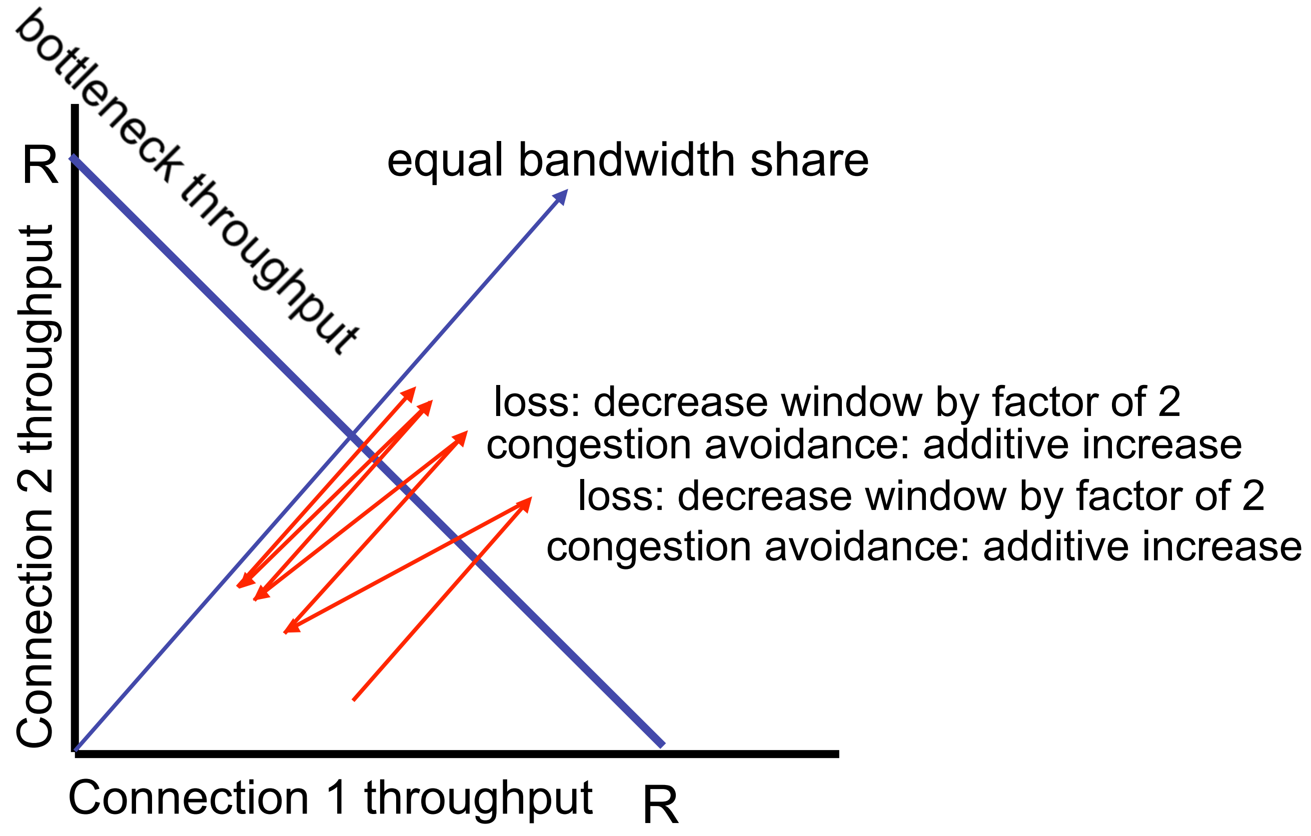
Fairness goal: If K TCP sessions share same bottleneck link of bandwidth R, each should have average rate of R/K



Why is TCP fair?

Two competing sessions:

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


bottleneck throughput
R
equal bandwidth share
Connection 2 throughput
loss: decrease window by factor of 2
congestion avoidance: additive increase
loss: decrease window by factor of 2
congestion avoidance: additive increase
Connection 1 throughput
R

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: Make these protocols TCP friendly
- ❑ One approach: DCCP (Datagram Congestion Control Protocol)
  - ▪ "UDP with congestion control"
  - ▪ Not very popular (as yet)

Fairness and parallel TCP connections

- ❑ Nothing prevents app from opening parallel connections between 2 hosts.
- ❑ Web browsers do this
- ❑ Example: Bottleneck link of rate R that is already supporting 9 connections
  - ▪ New application opens 1 TCP conn → gets rate R/10
  - ▪ New application opens 11 TCP conns → gets rate R/2 !

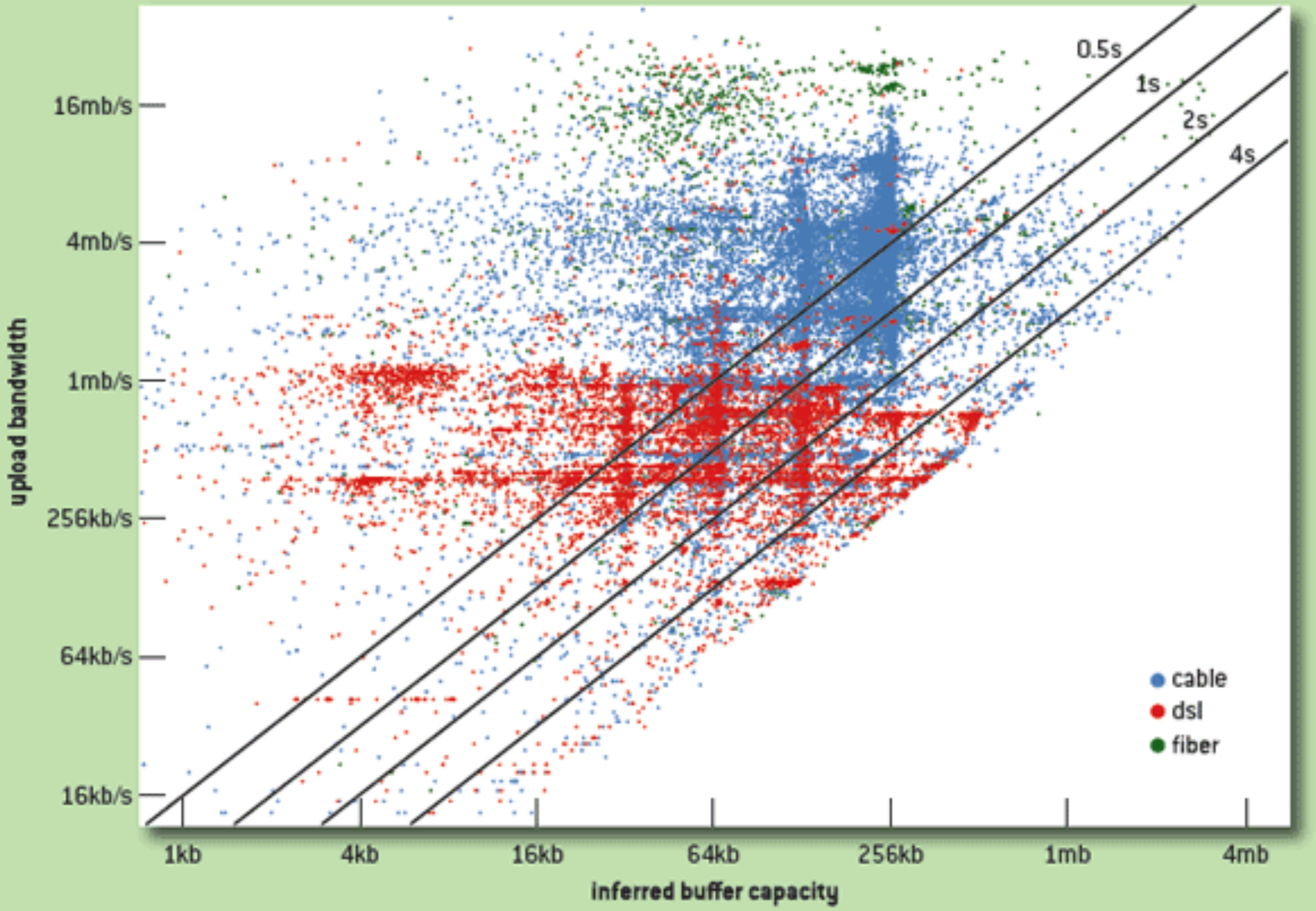
TCP and Buffer Bloat

- Capacities of router queues
  - “Large queue = good: Less packet losses at bottlenecks”
  - Do you agree? What would happen to TCP?
- Effects of large Buffers at bottleneck on TCP connections
  - Once queues are full: Queueing delays increase dramatically
  - TCP congestion control gets no early warning
    - No duplicate ACKS ⇨ no Fast Retransmit
    - Instead: Sudden timeouts
  - Congestion windows way too large
  - Many parallel TCP connections over same link get warning way too late
    - Synchronisation: Oscillation between “All send way too much” and “all get frightened by timeouts and send way too little”
    - Huge variations in queueing delays ⇨ DevRTT becomes very large ⇨ Timeout value becomes very large

Buffer bloat is a real-world problem

FIGURE 5

Plot Reproduced from ICSI's Netalyzr Studies



Chapter: Summary

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

Next:

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

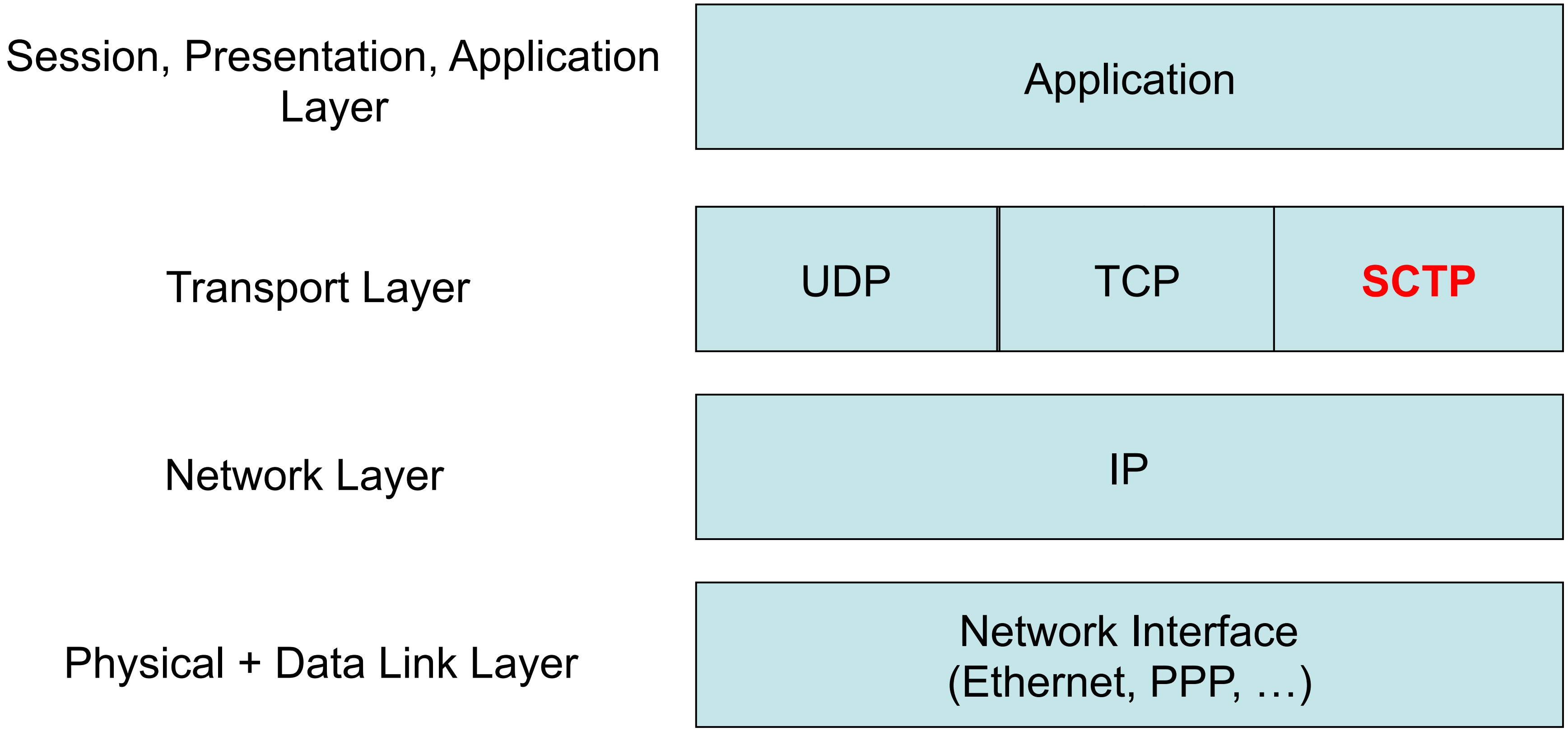
Chair for Network Architectures and Services – Prof. Carle
Department for Computer Science
TU München

Stream Control Transmission Protocol (SCTP)

Technische Universität München

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?

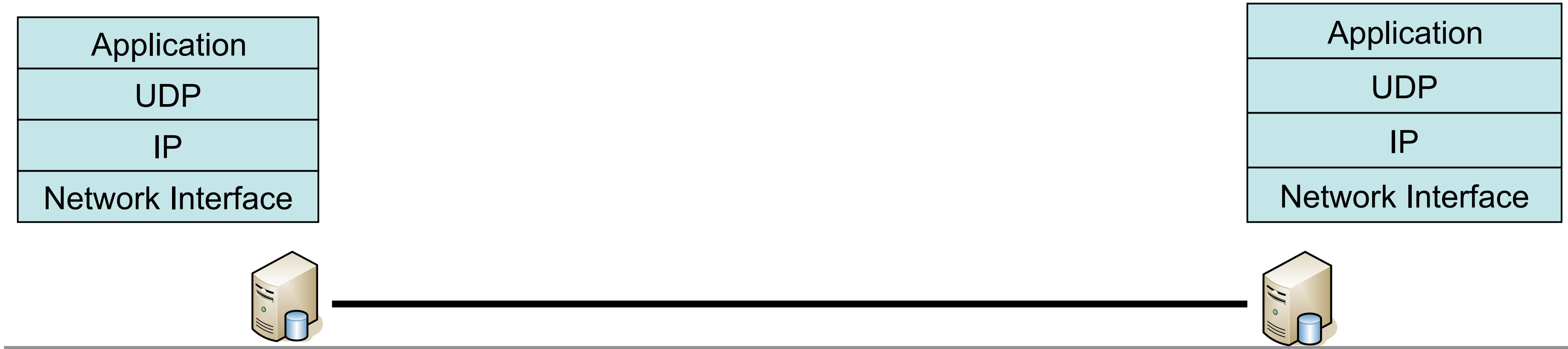
Contents

❑ Limitations of UDP and TCP

❑ The Stream Control Transmission Protocol (SCTP)
  - Association setup / stream setup
  - Message types
  - Partial Reliability
  - Multi-Homing support
  - Congestion control

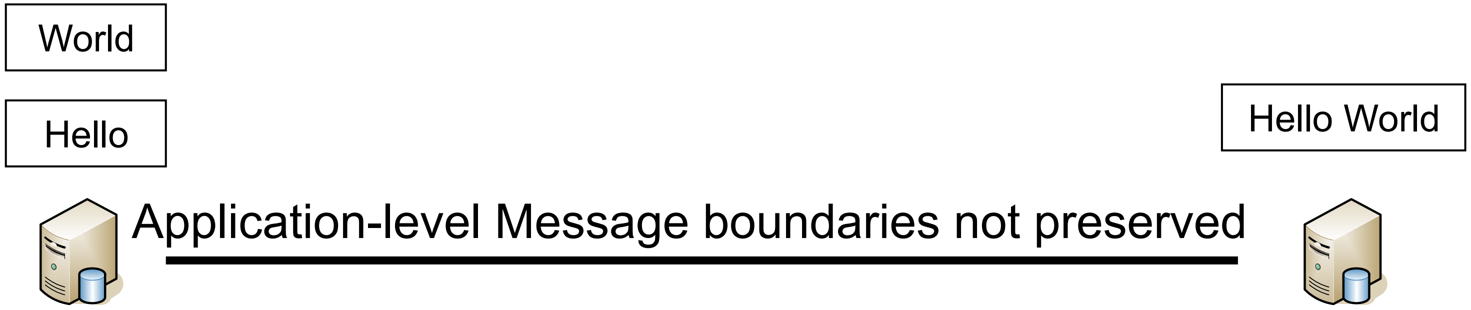
User Datagram Protocol

- 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

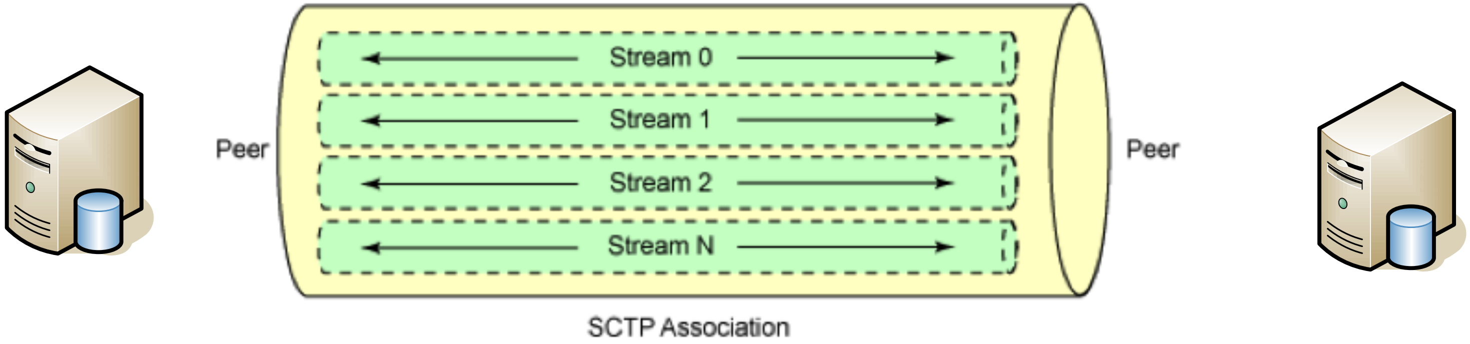
- Congestion control
  - TCP tries to share bandwidth equally between all end-points

Problems

- Certain applications have problems with 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:
  - Out-of-order delivery possible
  - Lost packets neither detected nor corrected
  - 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 delivery can be unreliable, fully reliable or partially reliable

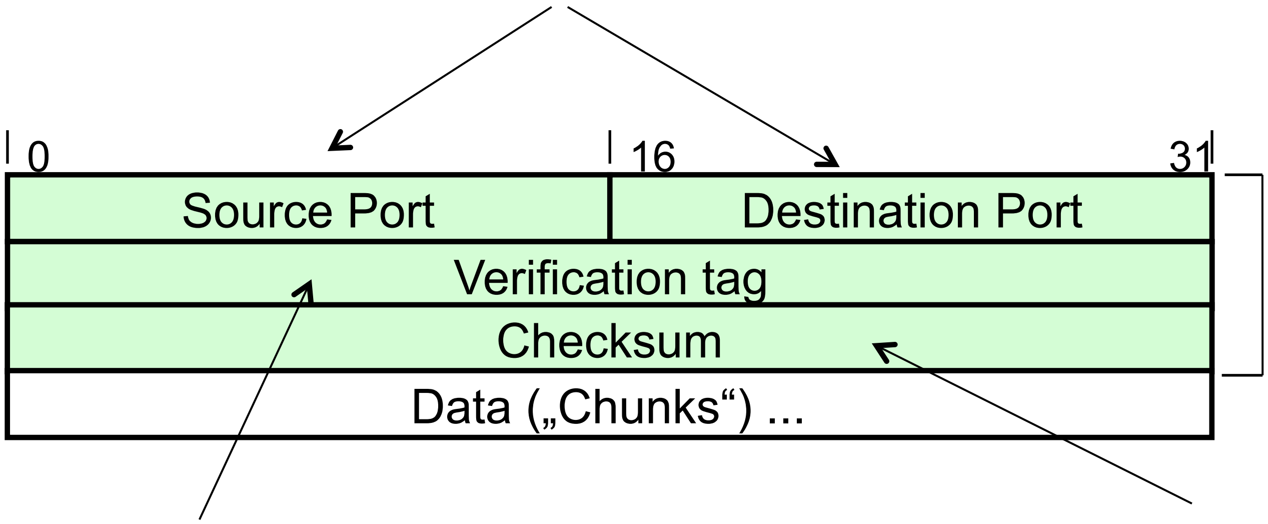
- **Multi-Homing**
  - SCTP can use multiple IP addresses

SCTP Message Format

❑ Common header format

- 12 byte header
- included in every SCTP message

Ports address the application



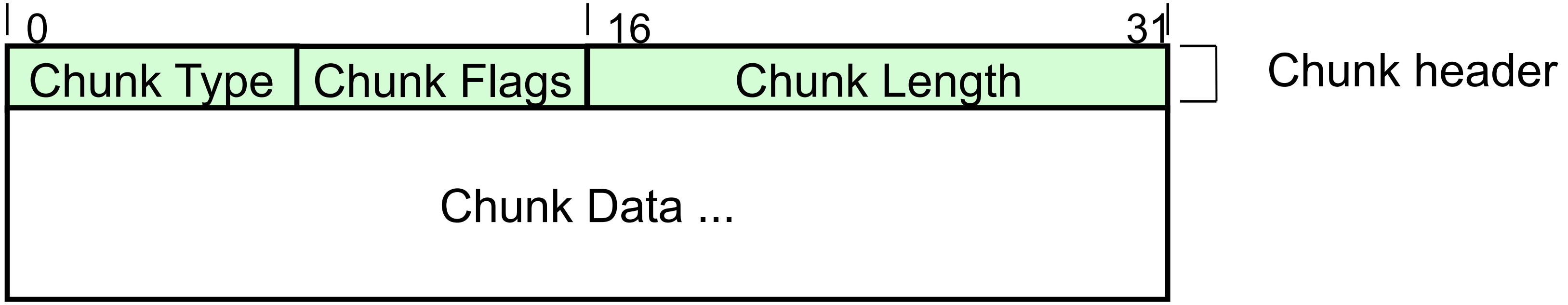
Random number which
Identifies a given association:
Used to distinguish new from old connections

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

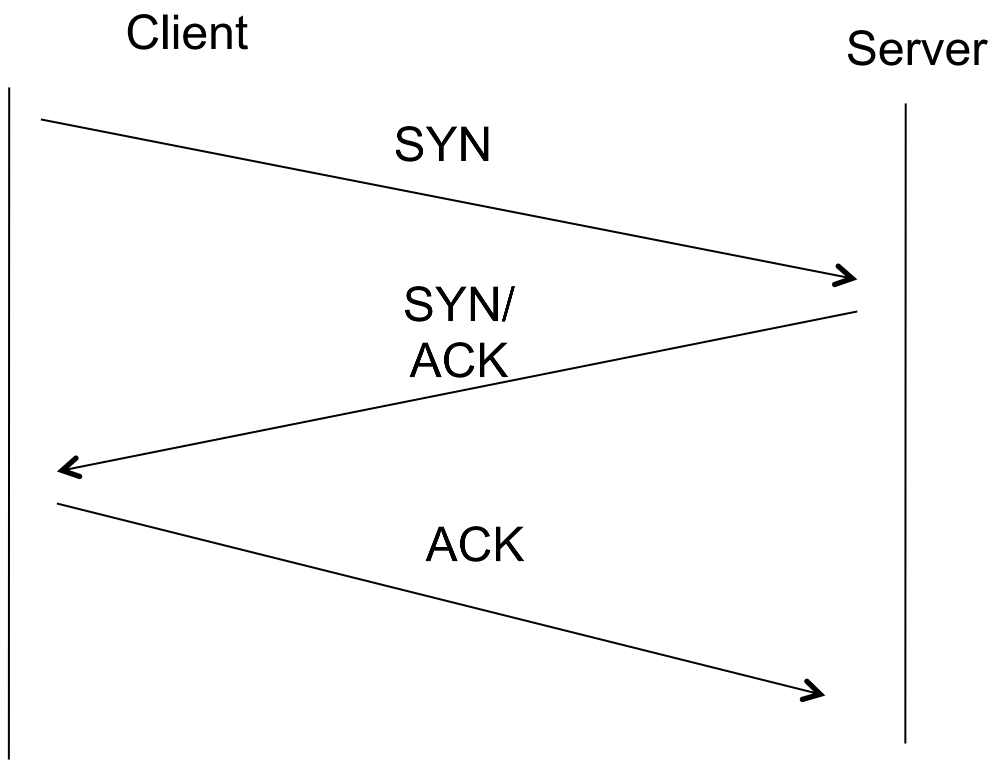
- Common chunk format



- Additional formats are defined for specific chunk types

Connection Setup

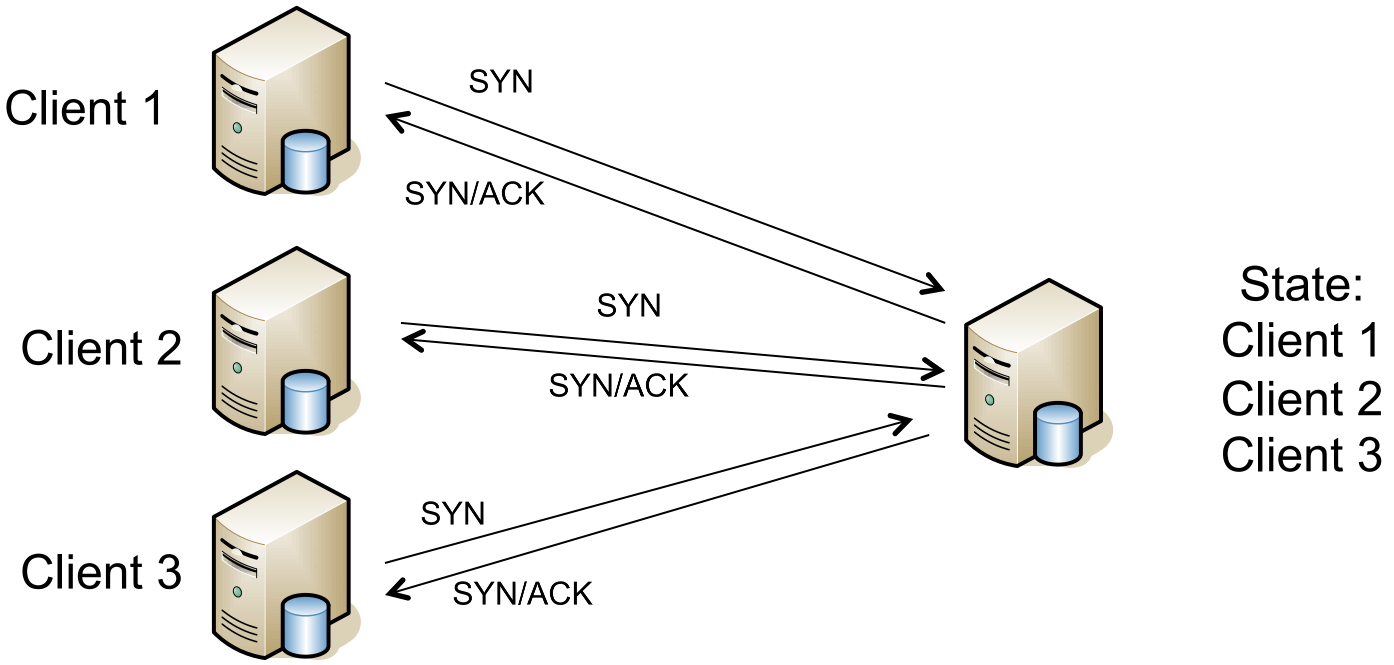
- TCP connection setup



Create State for
TCP connection: Store
client information

- Known Problem: TCP SYN-Flooding

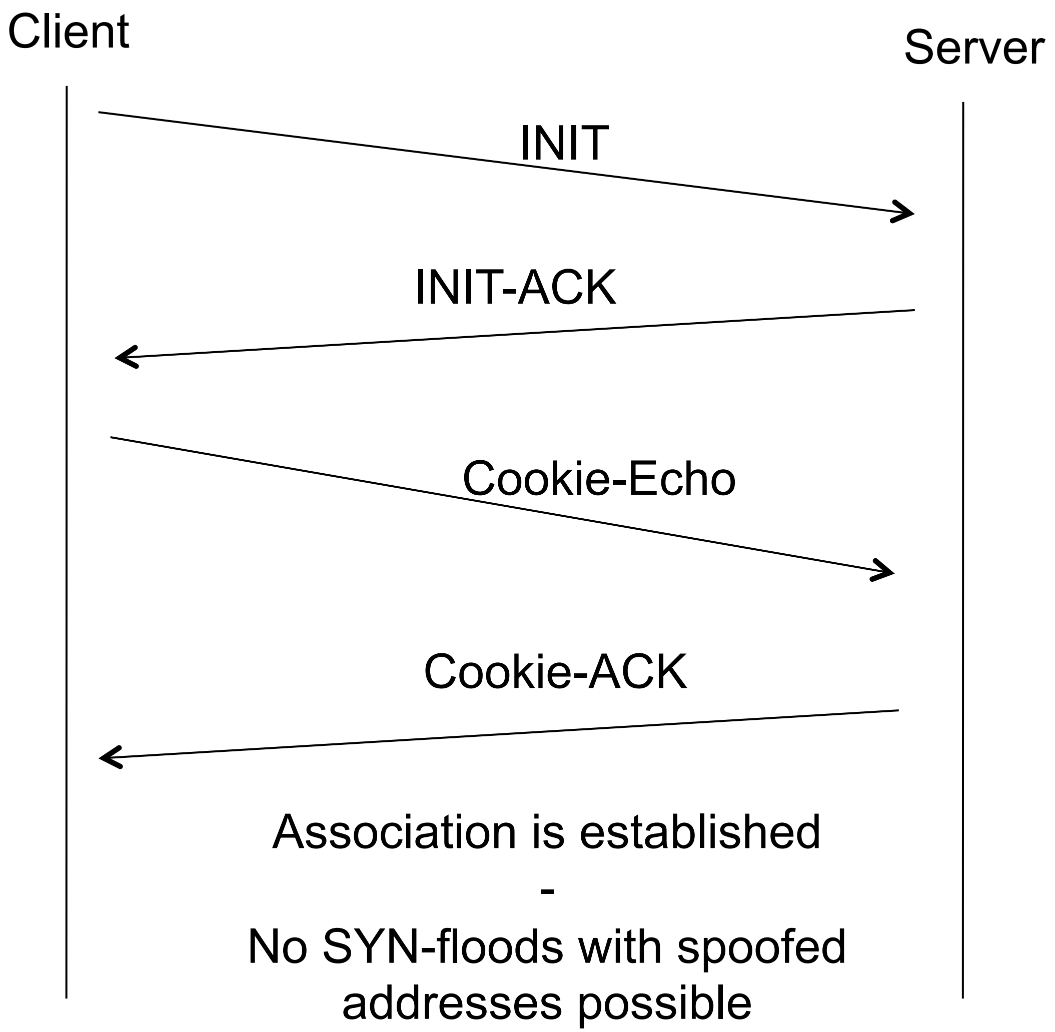
SYN Flodding



- ❑ 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
  - ▪ May lead to resource exhaustion, during which server cannot accept new connections

SCTP Association Setup

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



Client

Server

INIT

INIT-ACK

Cookie-Echo

Cookie-ACK

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

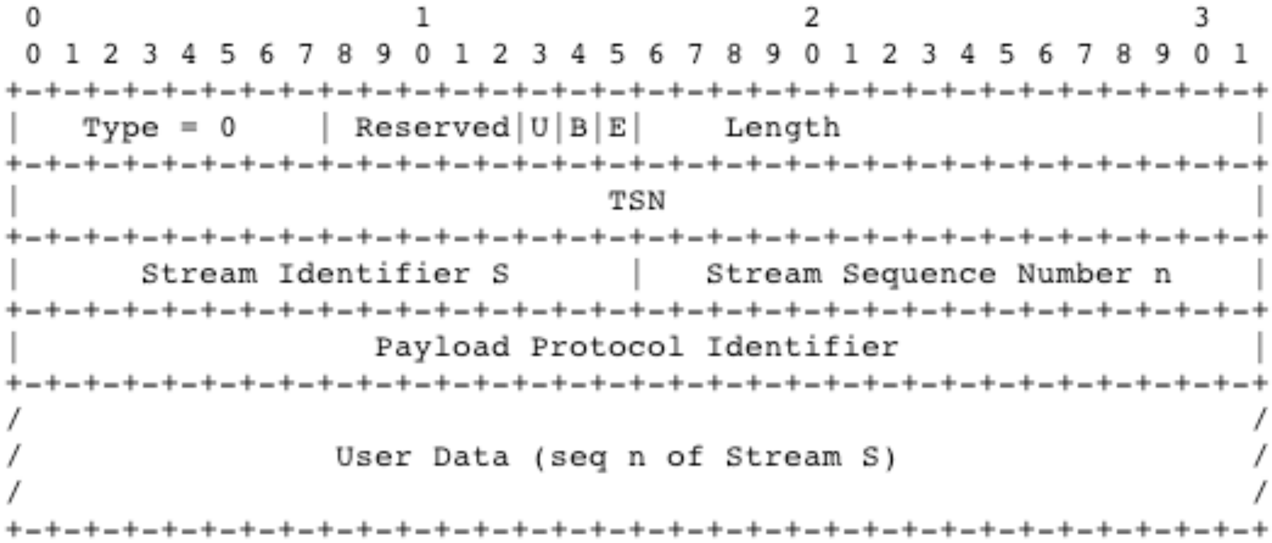
Generate client specific cookie

Send cookie ⇨ forget client

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

Data Transmission

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

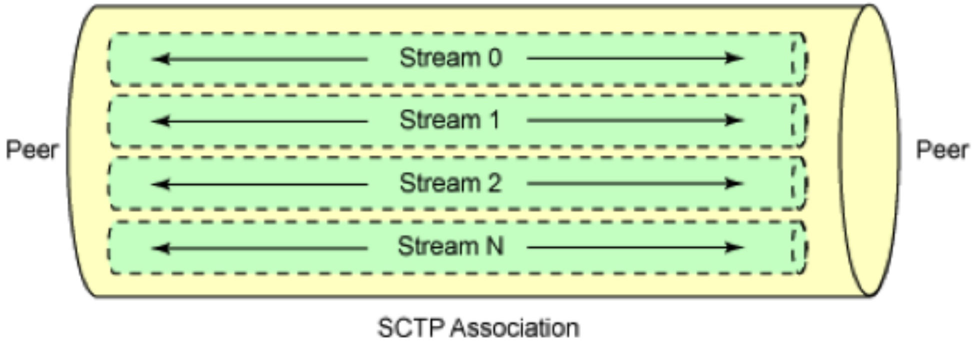


```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 0    | Reserved|U|B|E|    Length                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              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

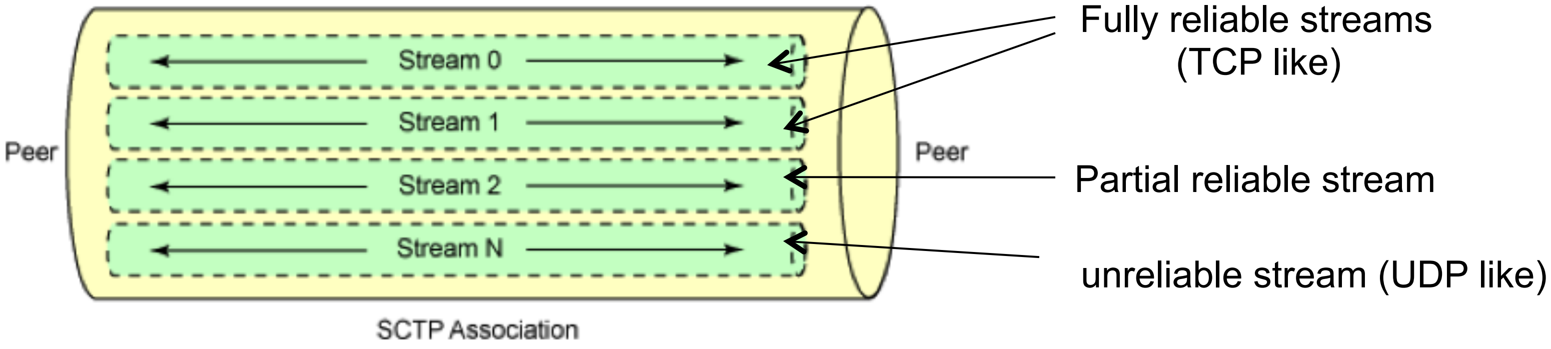
Transmission reliability (1)

- TCP
  - Segments are transmitted fully reliably
  - Segments 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
  - Simpler firewall rules (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



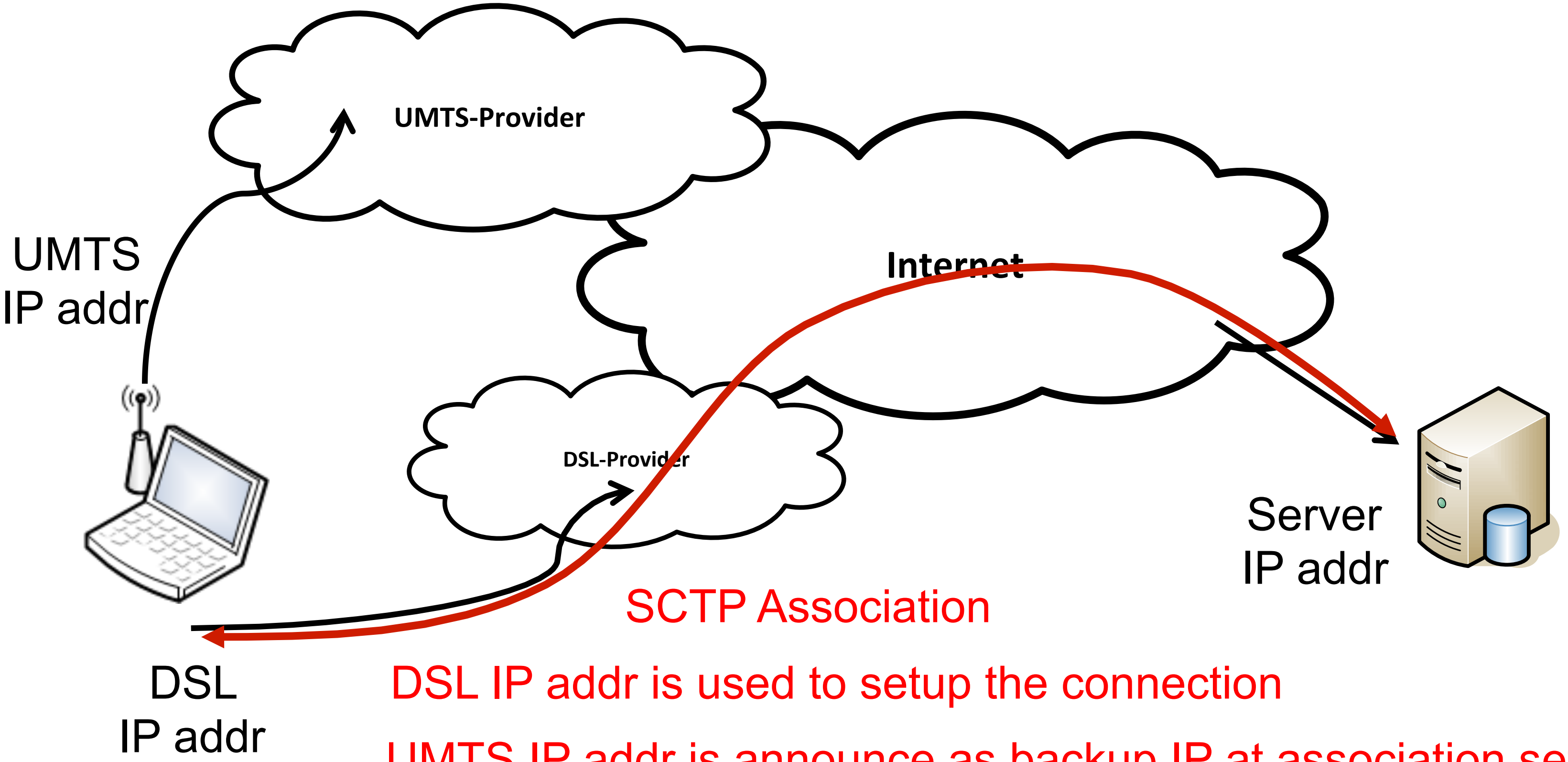
Fully reliable streams (TCP like)

Partial reliable stream

unreliable stream (UDP like)

Multi-Homing: Association setup

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

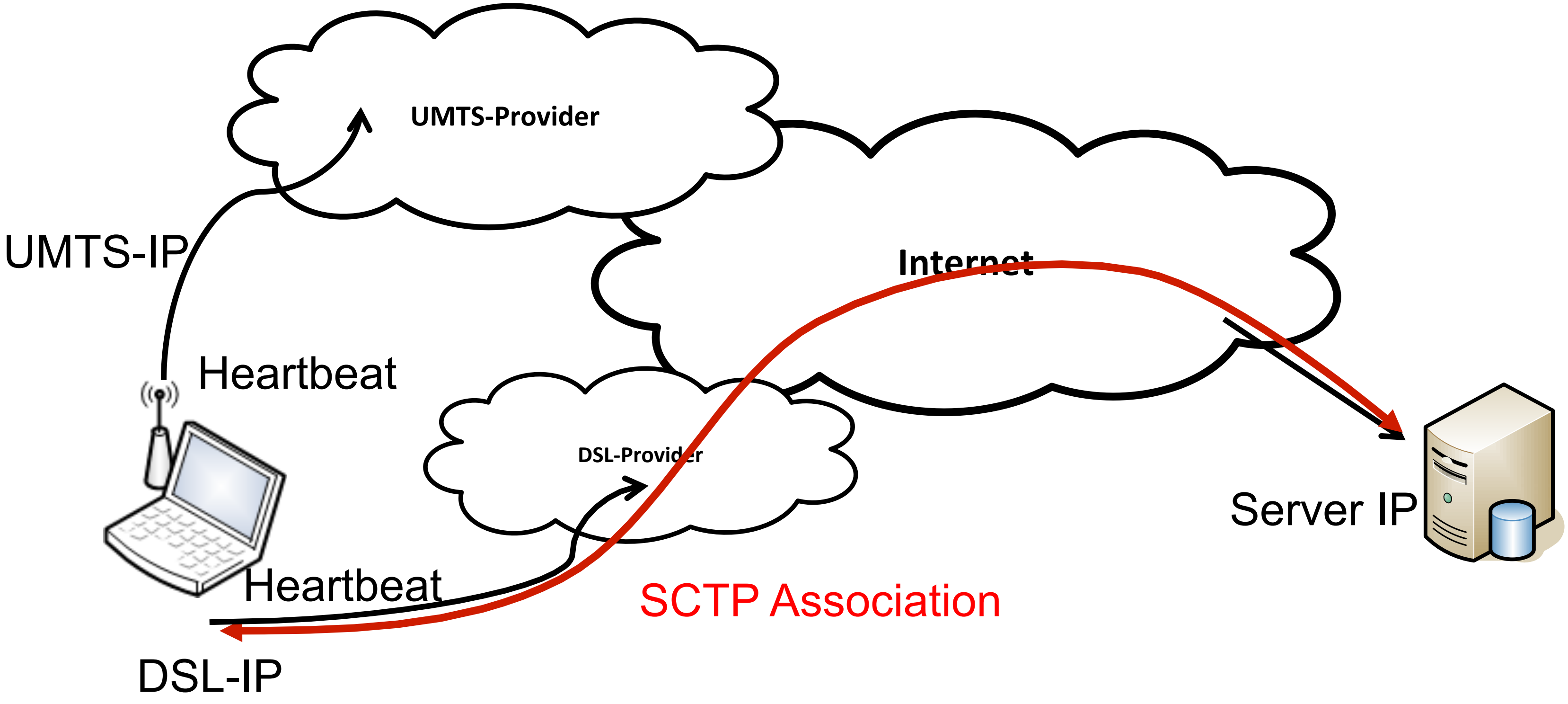


DSL IP addr is used to setup the connection

UMTS IP addr is announce as backup IP at association setup

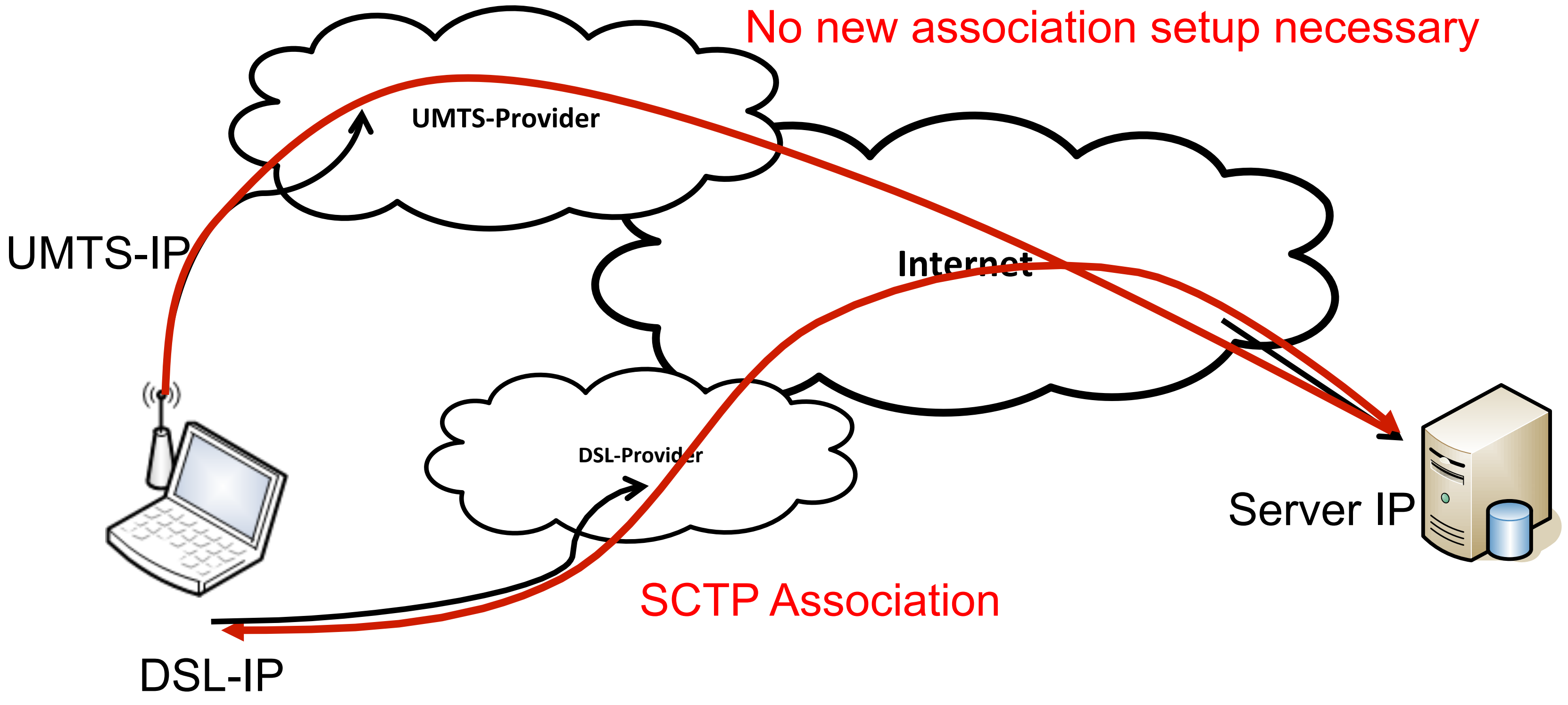
Multi-Homing

- 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 Deployment

- SCTP has attractive features
  - but to which extent is it used?

- Why do we use HTTP over TCP for Video Streaming?

- Firewall and NAT issues
  - Most home routers simply can‘t translate SCTP

- Implementations
  - not yet supported by all operating systems / hosts

- BUT: mandatory for some newly developed protocols such as IPFIX (IP Flow Information Export)

SCTP Standardisation

RFC 6458 Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)

RFC 6096 Stream Control Transmission Protocol (SCTP) Chunk Flags Registration (updates RFC 4960)

RFC 5062 Security Attacks Found Against the Stream Control Transmission Protocol (SCTP) and Current Countermeasures

RFC 5061 Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration

RFC 5043 Stream Control Transmission Protocol (SCTP) Direct Data Placement (DDP) Adaptation

RFC 4960 Stream Control Transmission Protocol

RFC 4895 Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)

RFC 4820 Padding Chunk and Parameter for the Stream Control Transmission Protocol (SCTP)

RFC 4460 Stream Control Transmission Protocol (SCTP) Specification Errata and Issues

RFC 3873 Stream Control Transmission Protocol (SCTP) Management Information Base (MIB)

RFC 3758 Stream Control Transmission Protocol (SCTP) Partial Reliability Extension

RFC 3554 On the Use of Stream Control Transmission Protocol (SCTP) with IPsec

RFC 3436 Transport Layer Security over Stream Control Transmission Protocol

RFC 3309 Stream Control Transmission Protocol (SCTP) Checksum Change (obsoleted by RFC 4960)

RFC 3286 An Introduction to the Stream Control Transmission Protocol

RFC 3257 Stream Control Transmission Protocol Applicability Statement

RFC 2960 Stream Control Transmission Protocol (updated by RFC 3309 and obsoleted by RFC 4960)

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

Reliable Multicast Transport

Technische Universität München

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

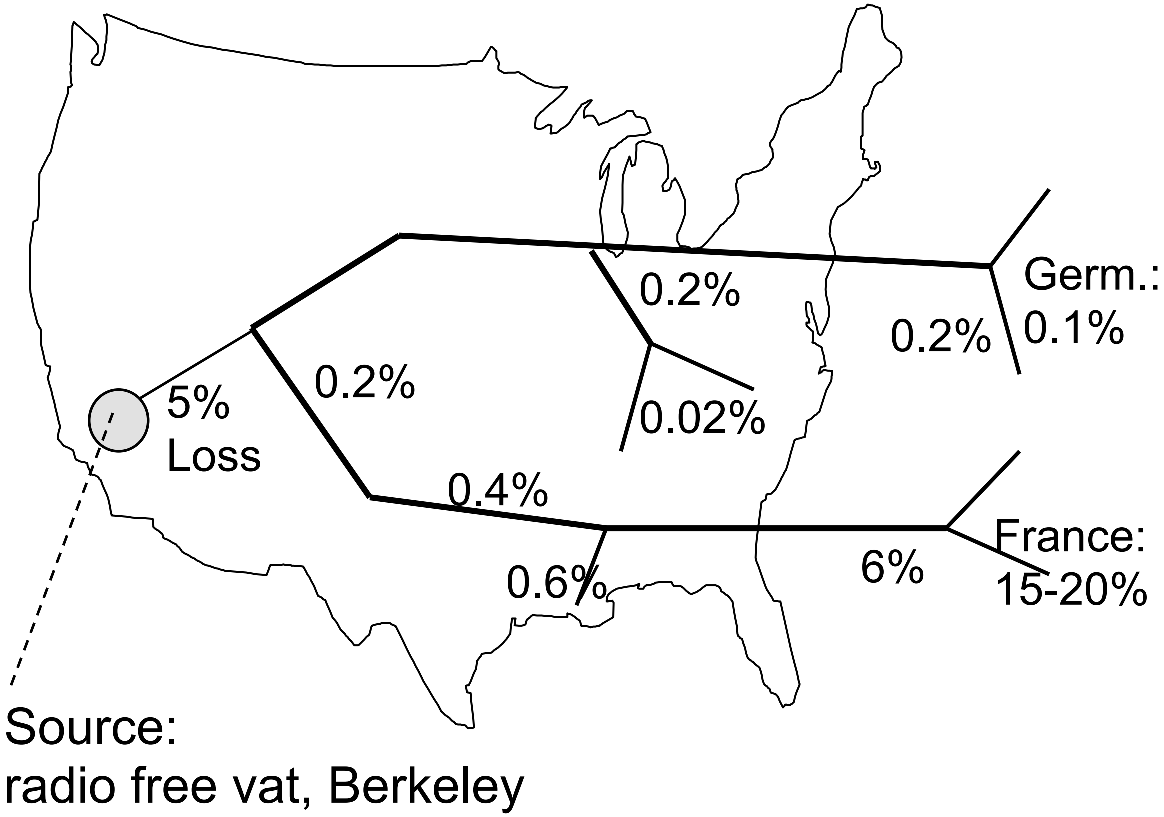
Classification of Multicast Applications

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 independently
  - full topology: measured per link loss independently
  - modified star: source-to-backbone plus star
    ⇨ good fit for example data set

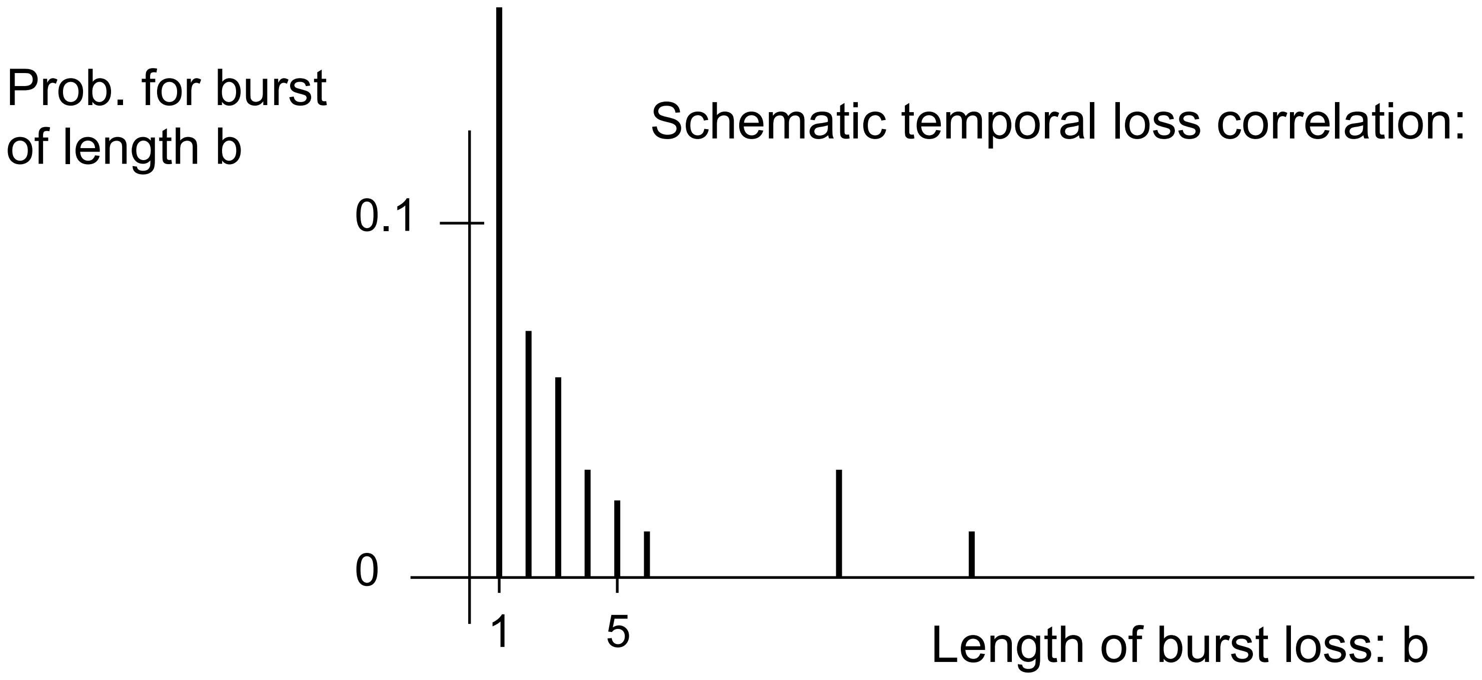
Temporal Loss Correlation

Q: do losses occur individually 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 / servers
  - 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!

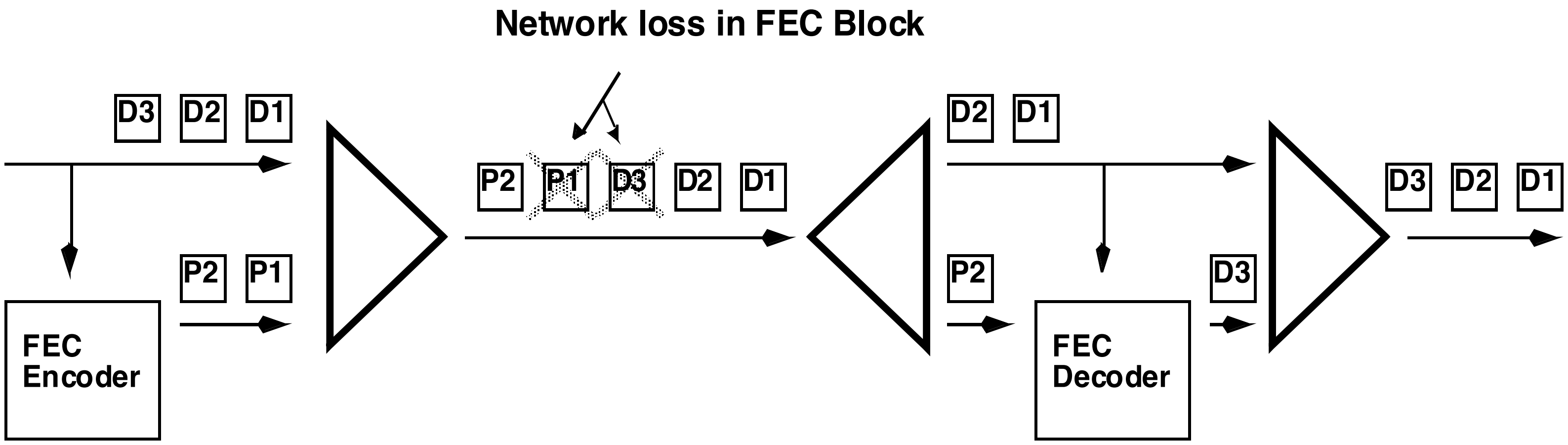
Forward Error Correction (FEC)

- 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

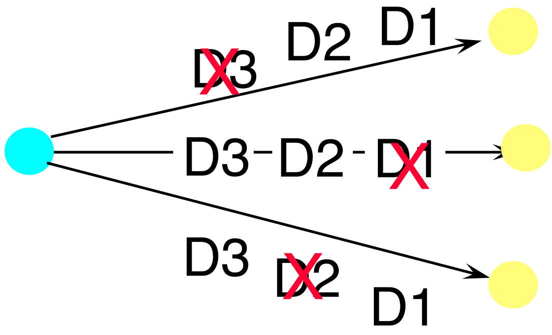
  - overhead at end-hosts

  - increased network load may increase loss probability

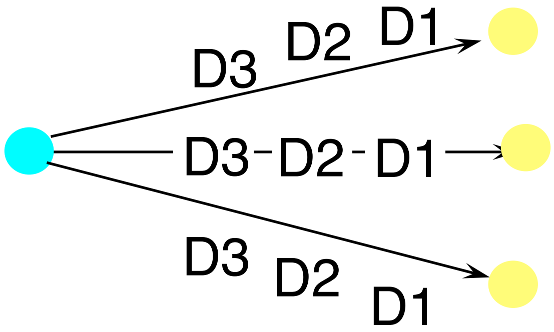


Potential Benefits of FEC

Initial Transmission

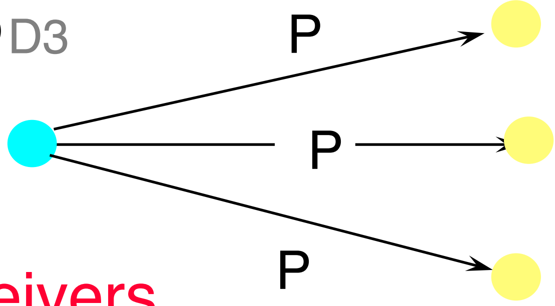


Data Retransmission



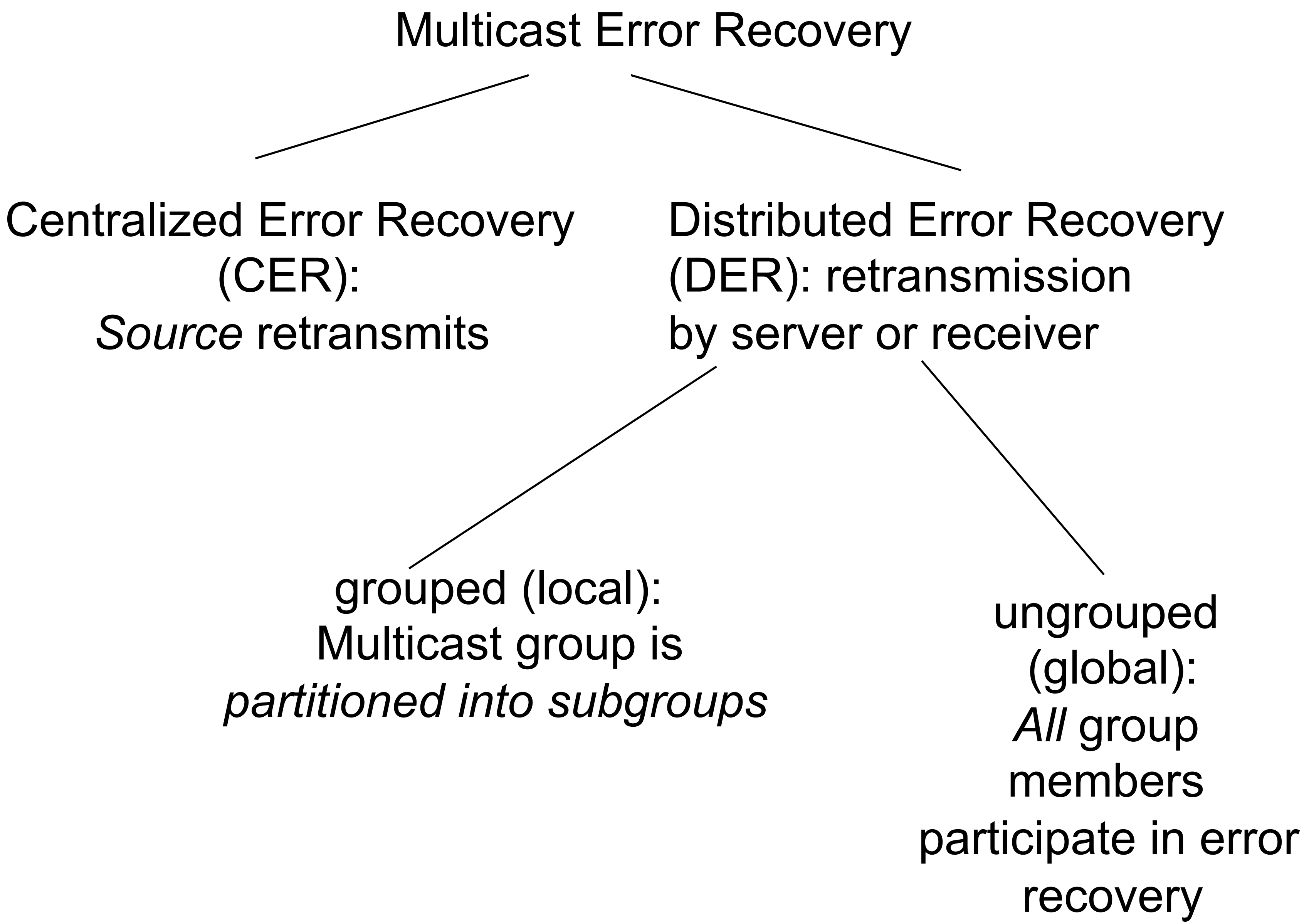
Parity Retransmission

P=D1⊗D2⊗D3



One parity packet can recover different data packets at different receivers

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

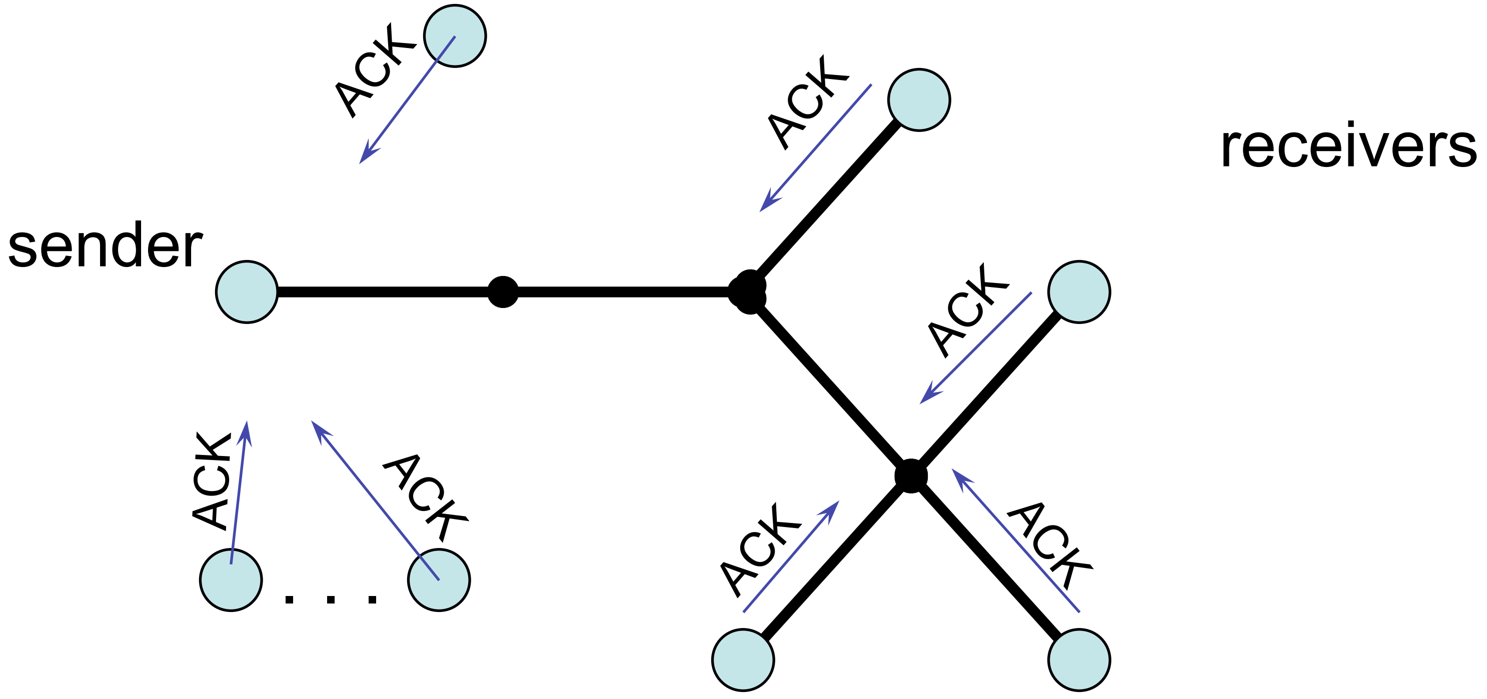
- Assume: R Receivers, independent packet loss probability p
- Calculate feedback per packet:
  - average number of ACKs: R - pR
  - average number of NAKs: pR

  ⇨ 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

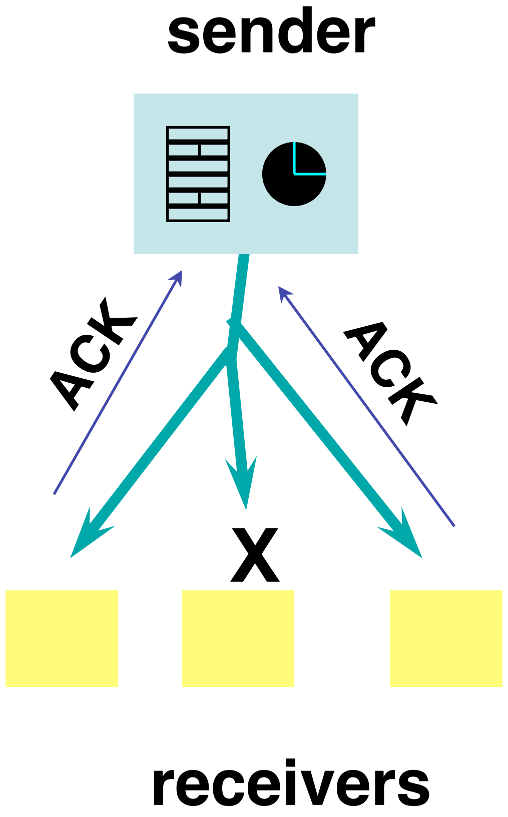
  For redundant feedback additionally:
  - Feedback suppression (e.g. multicast NAKs, receiver back off randomly)

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

  ⇨ design tradeoffs

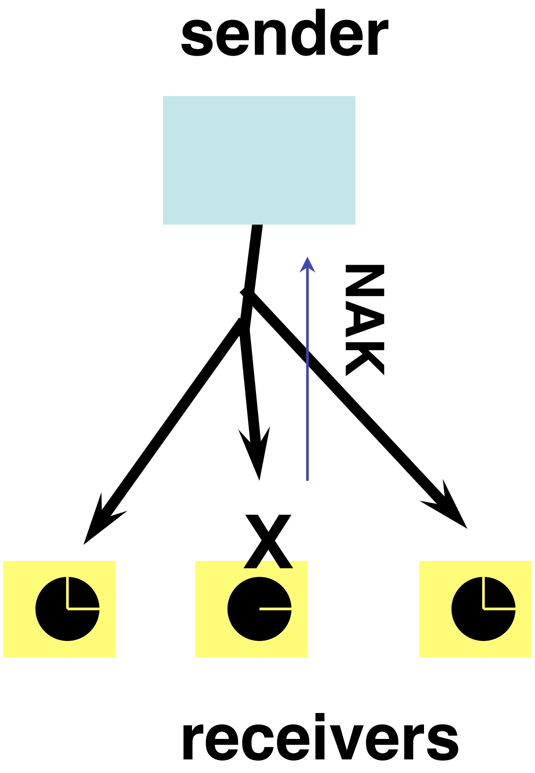
Sender Oriented Reliable Multicast

- 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

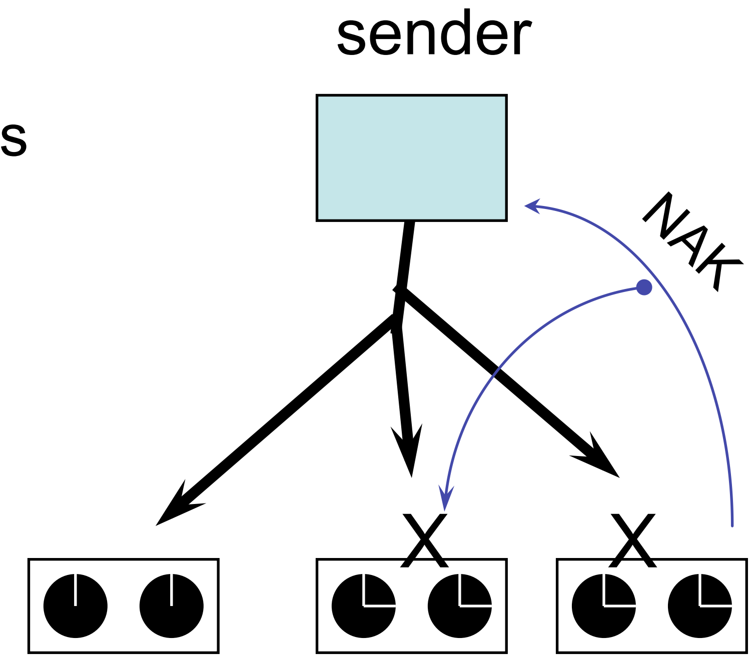
- ❑ randomly delay NAKs
- ❑ multicast to all receivers

  + reduce bandwidth

  - additional complexity at receivers (timers, etc)

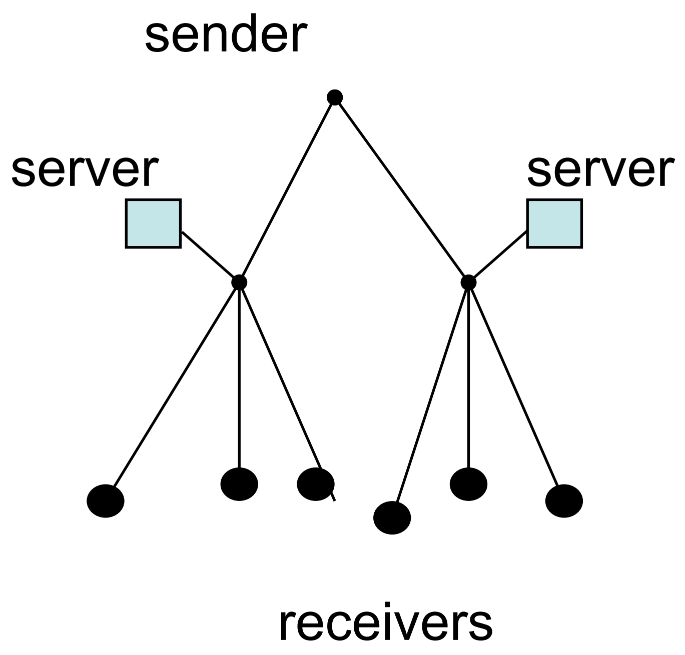
  - increase latencies (timers)

- ❑ similar to CSMA/CD



Server-based Reliable Multicast

- first transmissions: 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

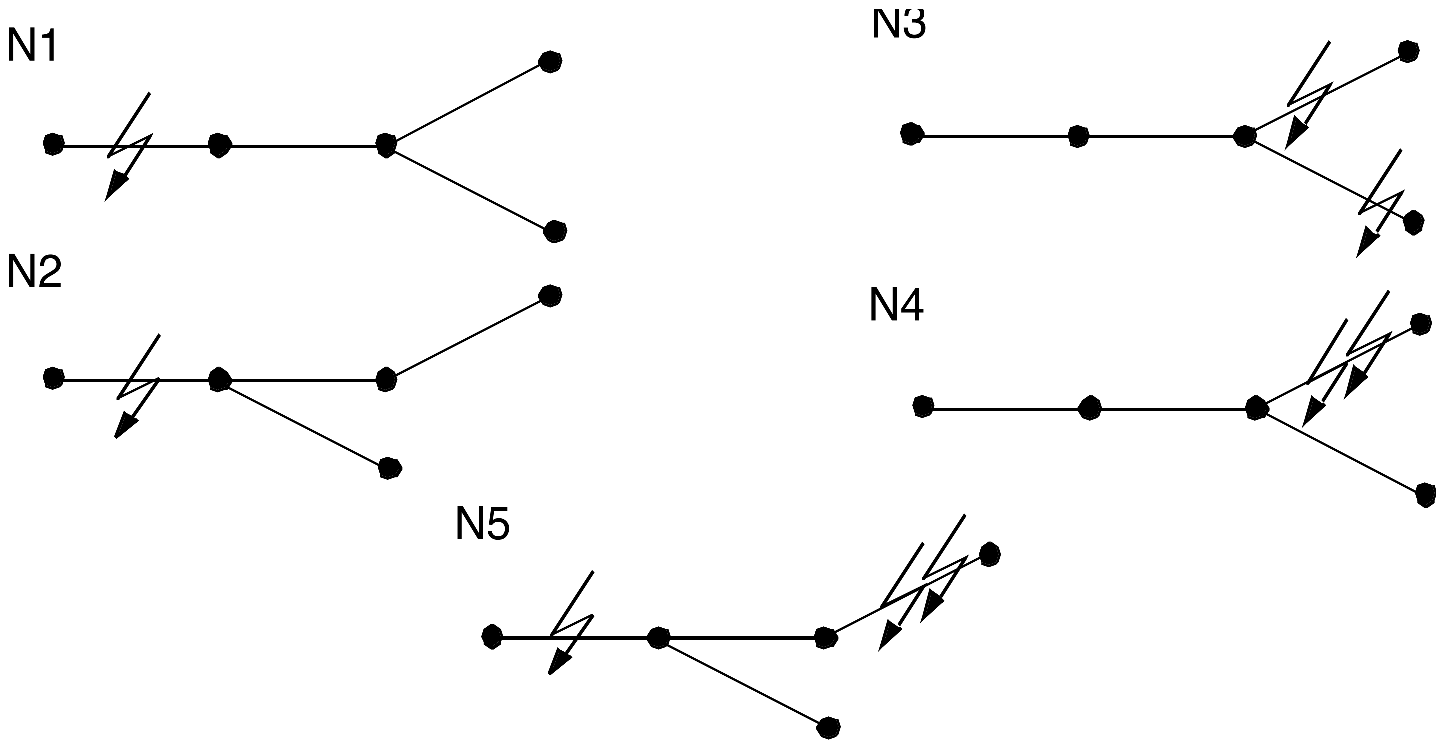
- hybrid methods

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

Influence of topology: Selected Scenarios for Modeling Heterogeneity

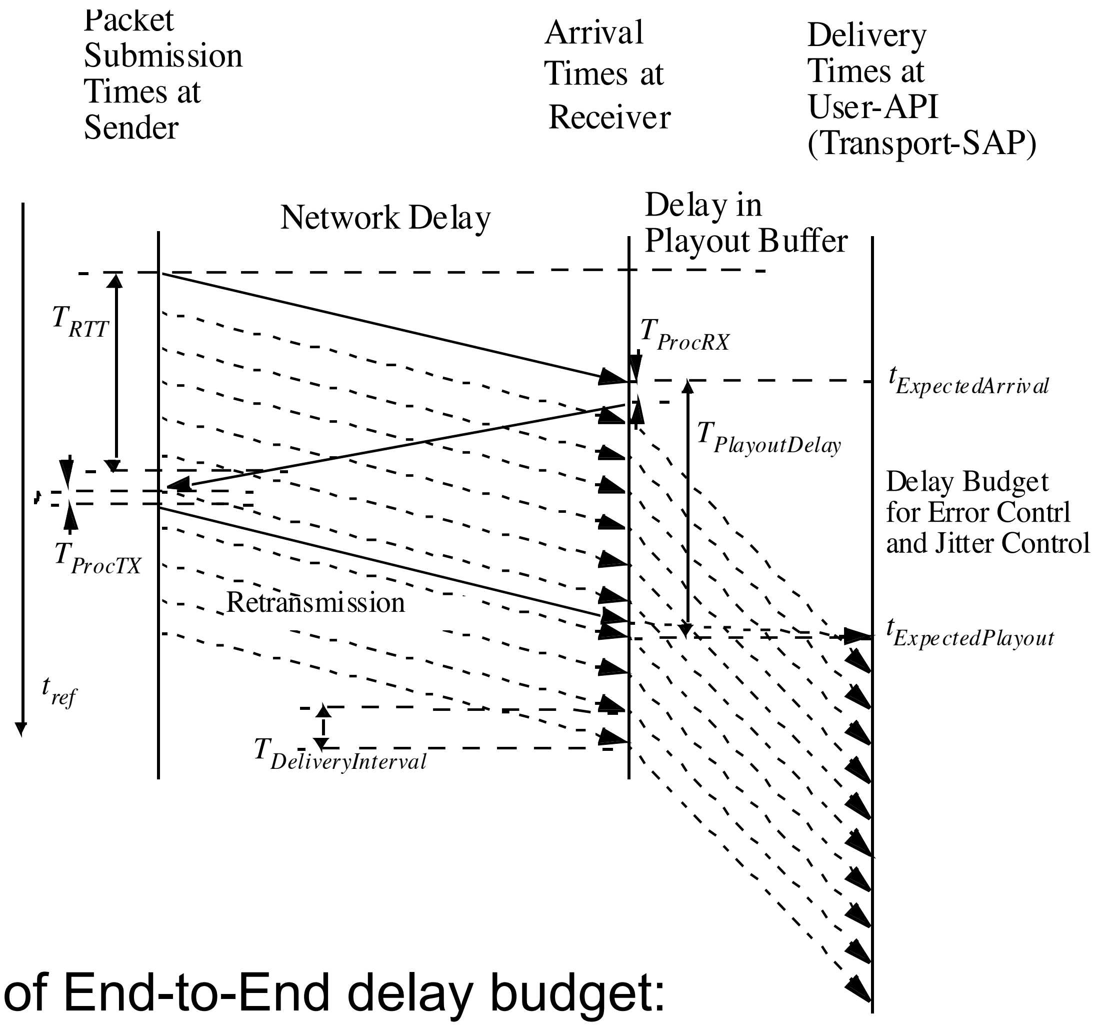
- Loss: on shared links / on individual links
- Loss: homogeneous / heterogeneous probability
- RTT: homogeneous / heterogeneous



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)

Reliable Multicast for Audio-Visual Applications



- ❑ Exploitation of End-to-End delay budget:
  we can always trade-off reliability for delay!
  (e.g. use 10 s delay budget to get 20% loss probability down to 2%)

Chapter:
Signalling

SIP
Session Initiation Protocol

Credits

Jim Kurose and Keith Ross

Julie Chan, Vovida Networks.

Milind Nimesh, Columbia University

Christian Hoene, University of Tübingen

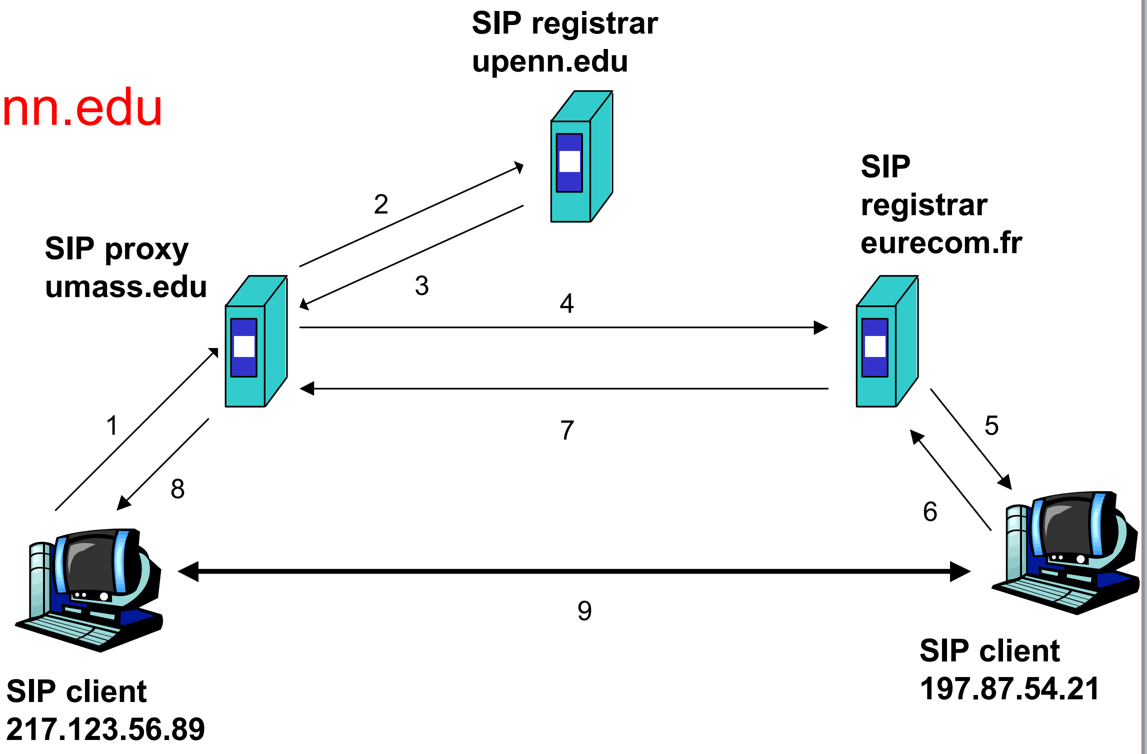
Example

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 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

SIP Messages – Methods and Responses

SIP components communicate by exchanging SIP messages:

SIP Methods:

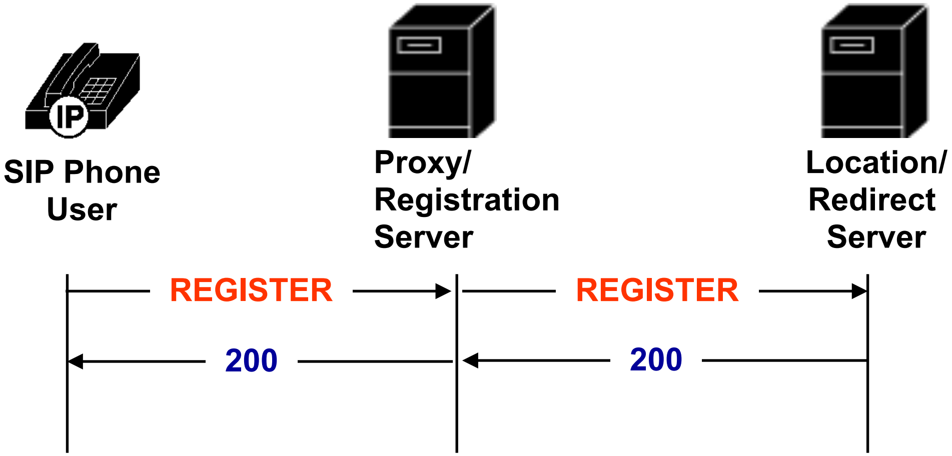
- INVITE – Initiates a call by inviting user to participate in session.
- ACK - Confirms that the client has received a final response to an INVITE request.
- BYE - Indicates termination of the call.
- CANCEL - Cancels a pending request.
- REGISTER – Registers the user agent.
- OPTIONS – Used to query 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.

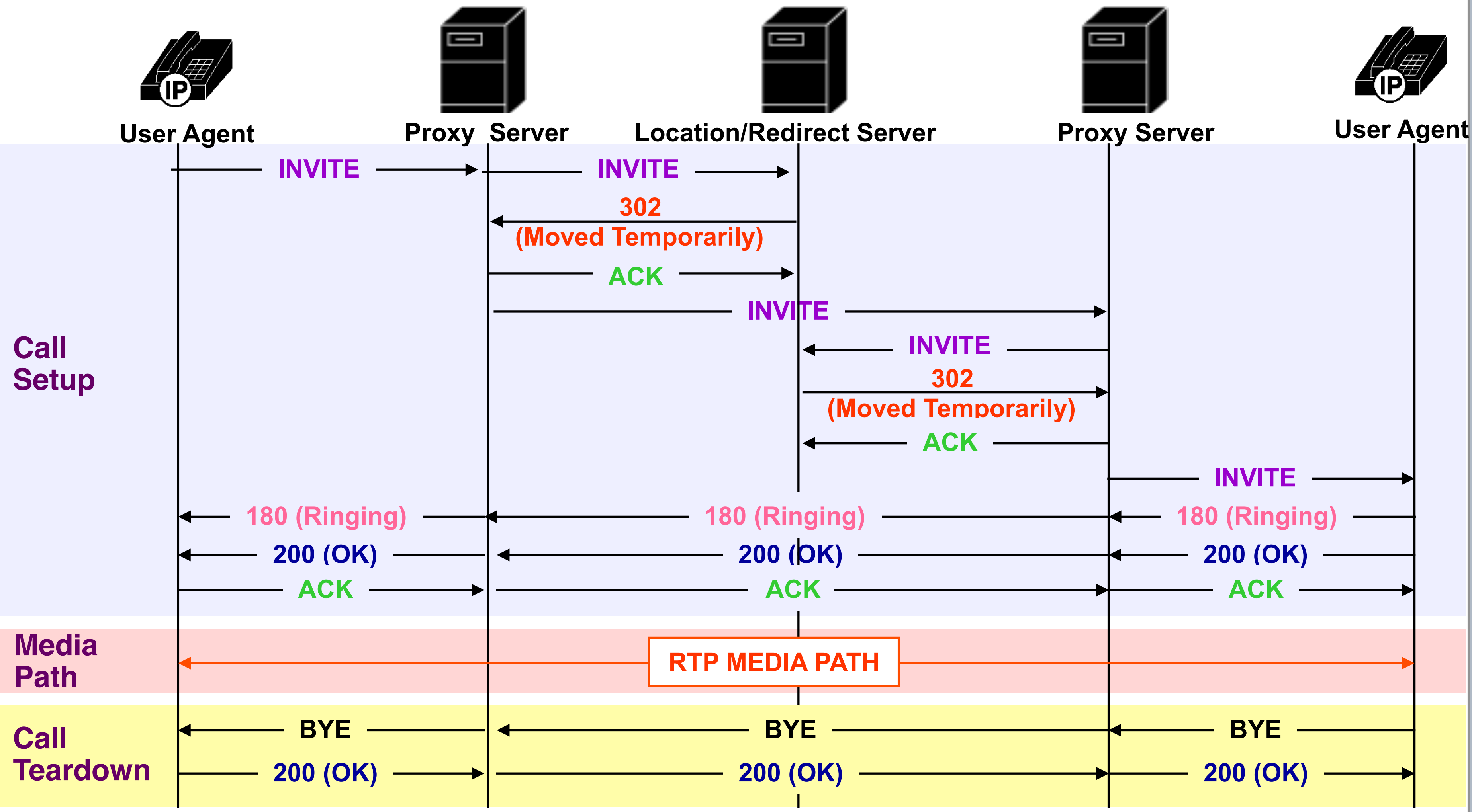
Registration

- ❑ Each time a user turns on the SIP user client (SIP IP Phone, PC, or other SIP device), the client registers with the proxy/ registration server.
- ❑ Registration can also occur when the SIP user client needs to inform the proxy/registration server of its location.
- ❑ The registration information is periodically refreshed and each user client must re-register with the proxy/registration server.
- ❑ Typically the proxy/registration server will forward this information to be saved in the location/redirect server.

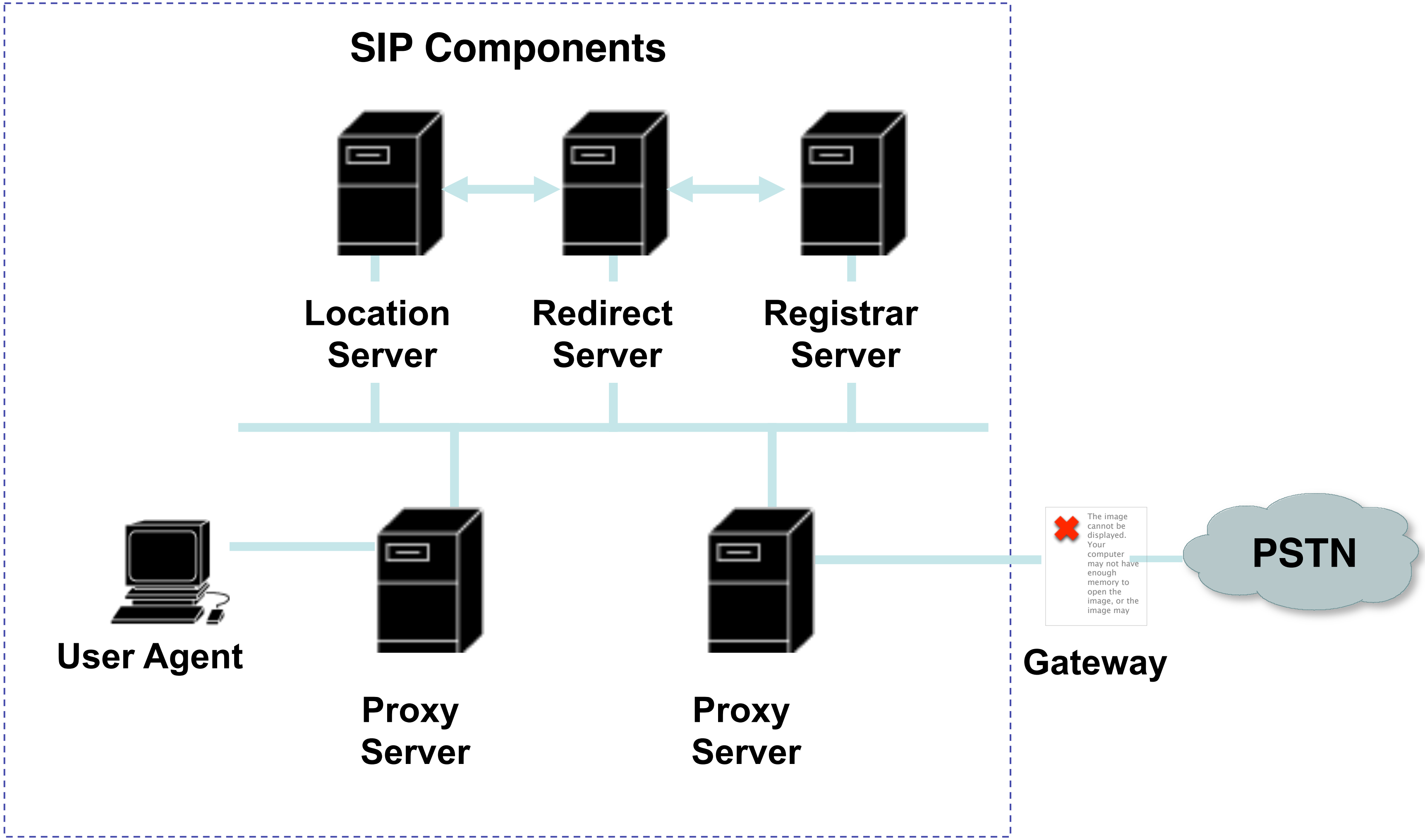


SIP Messages:
REGISTER – Registers the address listed in the To header field.
200 – OK.

Simplified SIP Call Setup and Teardown



SIP Architecture


SIP Components
Location Server
Redirect Server
Registrar Server
User Agent
Proxy Server
Proxy Server
The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may
Gateway
PSTN

User Agents, Proxy Server, Registrar Server

- **User Agent**: 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).
- Location can be transmitted by-value or by-reference.
- Location by reference is done by a URI that refers to a UA or proxy server
- A location Server transmits location by value in form of 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 – 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: RFC 3880
        - Designed to be implementable on either network servers or user agents. Meant to be simple, extensible, easily edited by graphical clients, and independent of operating system or signalling protocol. Suitable for running on a server where users may not be allowed to execute arbitrary programs, as it has no variables, loops, or ability to run external programs.
        - Syntactically, CPL scripts are represented by XML documents.

References

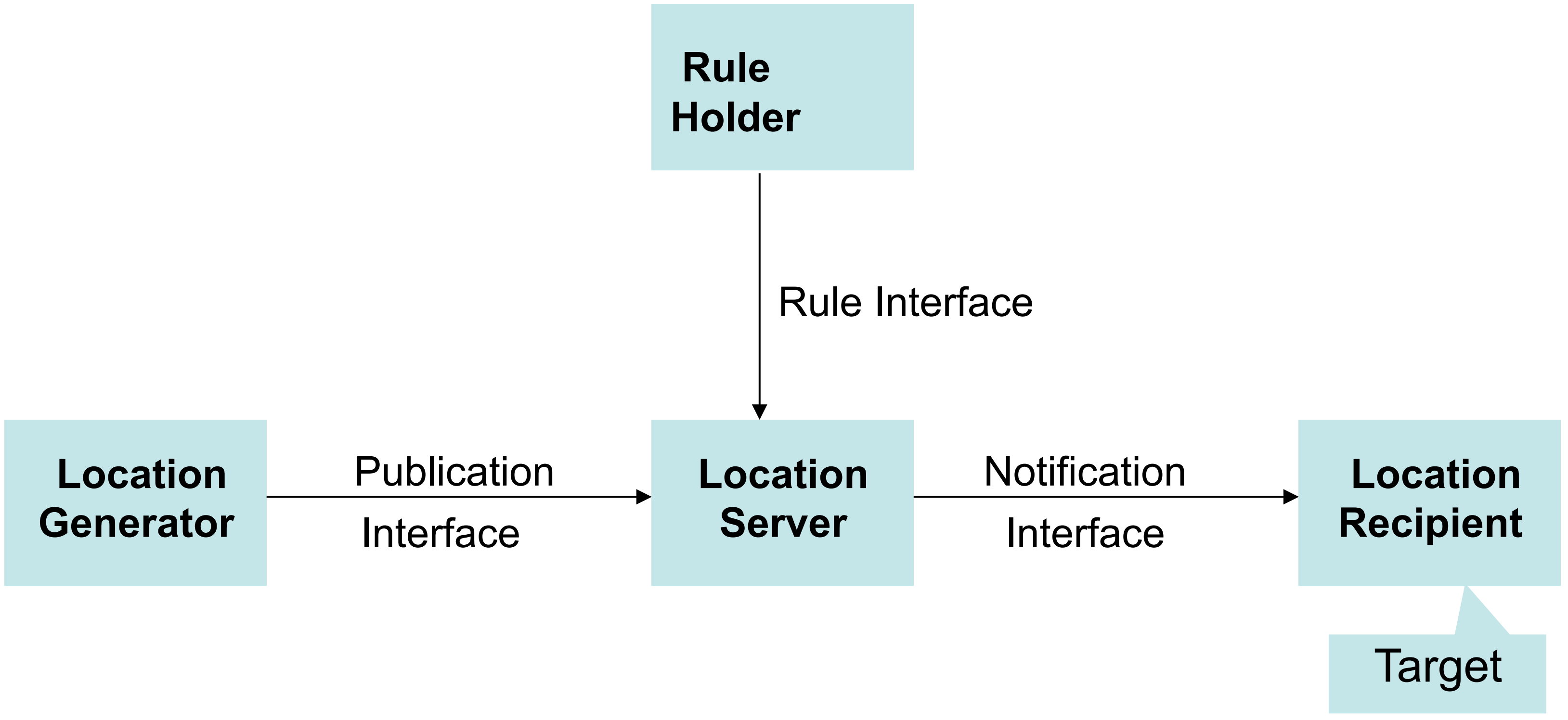
- For more information on SIP:
  - IETF: http://www.ietf.org/html.charters/sip-charter.html
- Henning Schulzrinne's SIP page
  - http://www.cs.columbia.edu/~hgs/sip/

Location Information and IETF GeoPriv Working Group

credits:

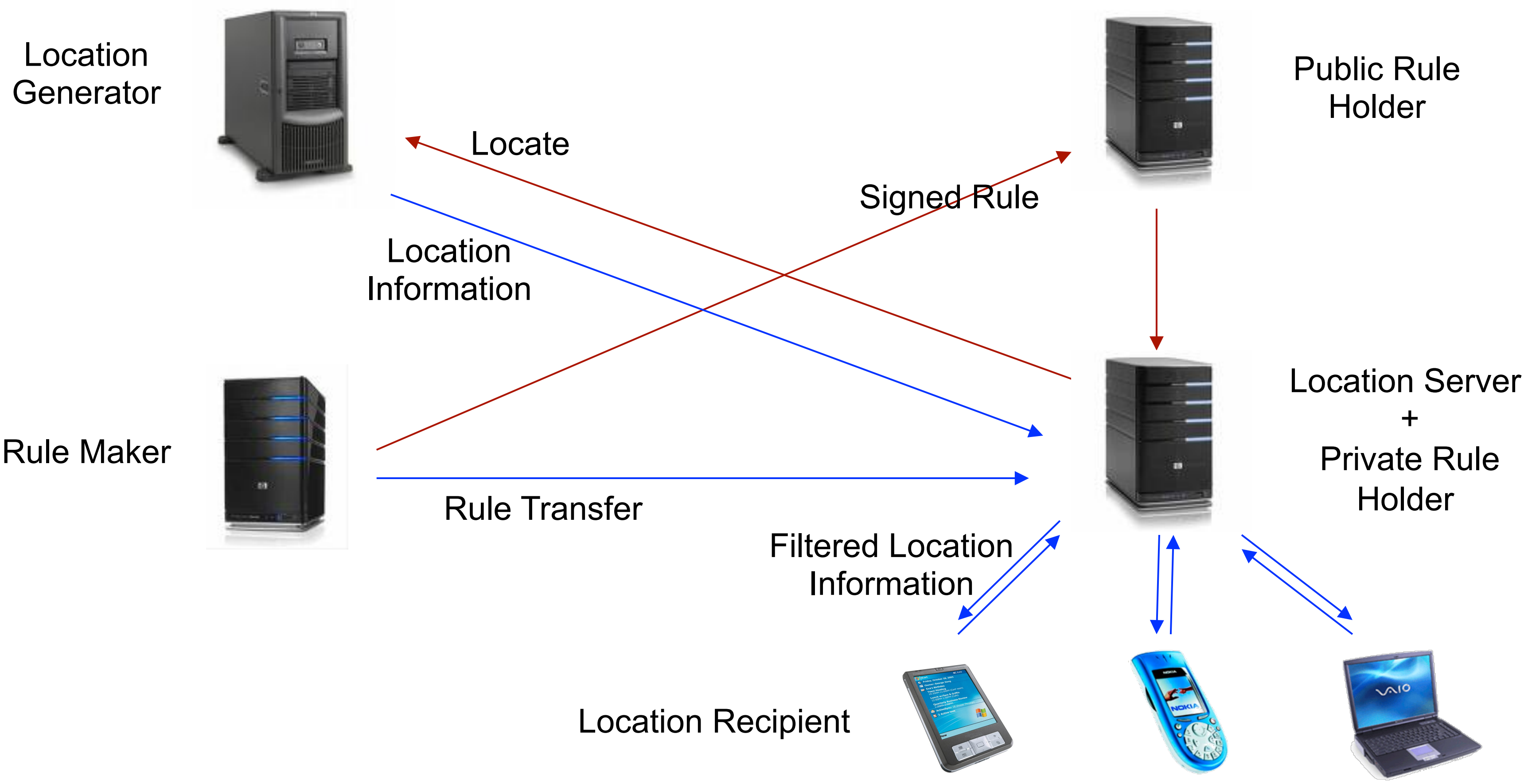
Milind Nimesh, Columbia University

Geopriv Entities



Location Recipients may request that a Location Server provide them with GEOPRIV location information concerning a particular Target.

Scenarios



Mobile Communities and Location-Based Services

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

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

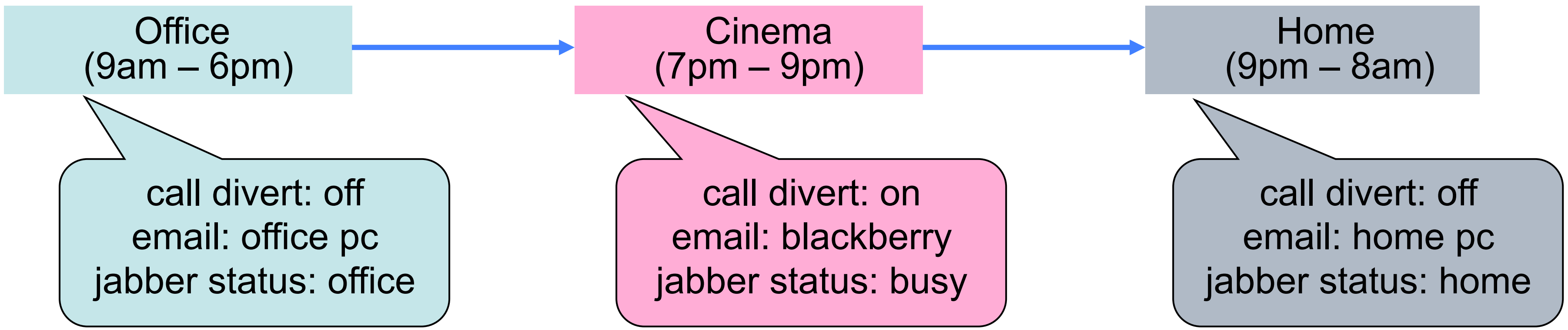
- location-info
- usage-rules
  - retransmission-allowed
  - retention-expires
  - ruleset-reference
  - note-well
- method
- provided-by

PIDF-LO Example

- ❑ PIDF-LO: RFC 4119 (RFC 5139, RFC 5491)
- ❑ c.f. http://www.voip-sos.net/tools/pidflo/

```
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
     xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
     entity="pres:sample@example.com">
<tuple id="0815">
<status>
 <gp:geopriv>
  <gp:location-info><!--  location information is inserted here --></gp:location-info>
  <gp:usage-rules>
     <gp:retransmission-allowed>no</gp:retransmission-allowed>
     <gp:retention-expiry>2010-08-10T09:00:10+02:00</gp:retention-expiry>
  </gp:usage-rules>
 </gp:geopriv>
</status>
<timestamp>2010-08-10T08:31:00+02:00</timestamp>
</tuple>
</presence>
```

Location Type Registry



Office
(9am – 6pm)

call divert: off
email: office pc
jabber status: office

Cinema
(7pm – 9pm)

call divert: on
email: blackberry
jabber status: busy

Home
(9pm – 8am)

call divert: off
email: home pc
jabber status: home

- ❑ 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

GeoPriv RFCs

- RFC 3693: Geopriv Requirements, 2004 (Informational), Updated by RFC 6280
- RFC 3694: Threat Analysis of the Geopriv Protocol, 2004 (Informational), Updated by RFC 6280
- RFC 3825: Dynamic Host Configuration Protocol Option for Coordinate-based Location Configuration Information, 2004 (Proposed Standard), Obsoleted by RFC 6225
- RFC 4079: A Presence Architecture for the Distribution of GEOPRIV Location Objects, 2005 (Informational)
- RFC 4119: A Presence-based GEOPRIV Location Object Format, 2005 (Proposed Standard), Updated by RFC 5139, RFC 5491
- RFC 4589: Location Types Registry, 2006 (Proposed Standard)
- RFC 4676: Dynamic Host Configuration Protocol (DHCPv4 and DHCPv6) Option for Civic Addresses Configuration Information, 2006 (Proposed Standard), Obsoleted by RFC 4776
- RFC 4745, Common Policy: A Document Format for Expressing Privacy Preferences, 2007 (Proposed Standard)
- RFC 4776: Dynamic Host Configuration Protocol (DHCPv4 and DHCPv6) Option for Civic Addresses Configuration Information, 2006 (Proposed Standard), Updated by RFC 5774

GeoPriv RFCs

- RFC 5139: Revised Civic Location Format for Presence Information Data Format Location Object (PIDF-LO), 2008 (Proposed Standard)
- RFC 5491: GEOPRIV Presence Information Data Format Location Object (PIDF-LO) Usage Clarification, Considerations, and Recommendations 2009 (Proposed Standard)
- RFC 5580: Carrying Location Objects in RADIUS and Diameter, 2009 (Proposed Standard)
- RFC 5606: Implications of 'retransmission-allowed' for SIP Location Conveyance, 2009 (Informational)
- RFC 5687: GEOPRIV Layer 7 Location Configuration Protocol: Problem Statement and Requirements, 2010 (Informational)
- RFC 5774: Considerations for Civic Addresses in the Presence Information Data Format Location Object (PIDF-LO): Guidelines and IANA Registry Definition, 2010 (Best Current Practice)
- RFC 5808: Requirements for a Location-by-Reference Mechanism, 2010 (Informational)

GeoPriv RFCs

- RFC 5870: A Uniform Resource Identifier for Geographic Locations ('geo' URI), 2010 (Proposed Standard)
- RFC 5985: HTTP-Enabled Location Delivery (HELD), 2010 (Proposed Standard)
- RFC 5986: Discovering the Local Location Information Server (LIS), 2010 (Proposed Standard)
- RFC 6155: Use of Device Identity in HTTP-Enabled Location Delivery (HELD), 2011 (Proposed Standard)
- RFC 6225: Dynamic Host Configuration Protocol Options for Coordinate-Based Location Configuration Information, 2011 (Proposed Standard)
- RFC 6280: An Architecture for Location and Location Privacy in Internet Applications, 2011 (Best Current Practice)

GeoPriv Tools

c.f. http://trac.tools.ietf.org/wg/geopriv/trac/wiki/GeoprivTools

- Open Source LIS: A PHP-based HELD server with a Java-based client, http://held-location.sourceforge.net/
- The Internet Geolocation Toolkit: A multi-platform, multi-protocol C++ library for geolocation access, http://igtk.sourceforge.net/
- ECRITdroid: An emergency calling client for Android. Doesn't do GEOPRIV now (just LoST/ECRIT), but should soon, in order to be fully ECRIT-compliant, http://ecritdroid.googlecode.com/
- Online DHCP encoders: An AJAX tool for encoding location values for use in the DHCP location options; http://geopriv.dreamhosters.com/dhcloc/
- Firefox implementation of W3C Geolocation API: supports a limited profile of HELD. To enable: Go to "about:config"; set "geo.wifi.protocol" to "1"; set "geo.wifi.uri" to URL of HELD server, https://bugzilla.mozilla.org/show_bug.cgi?id=545001
- CommScope LIS: commercial LIS, http://www.commscope.com

Maintaining network state

Design Principles

Goals:

- identify, study common architectural components, protocol mechanisms
- what approaches do we find in network architectures?
- *synthesis:* big picture

7 design principles:

- network virtualization: overlays
- separation of data, control ⇨signalling
- **hard state versus soft state**
- randomization
- indirection
- multiplexing
- design for scale

Maintaining network state

state: information *stored* in network nodes by network protocols

- updated when network “conditions” change
- stored in multiple nodes
- often associated with end-system generated call or session
- examples:
  - ATM switches maintain lists of VCs: bandwidth allocations, VCI/VPI input-output mappings
  - RSVP routers maintain lists of upstream sender IDs, downstream receiver reservations
  - TCP: Sequence numbers, timer values, RTT estimates

Hard-state

- state *installed* by receiver on receipt of *setup message* from sender
- state *removed* by receiver on receipt of *teardown message* from sender
- *default assumption:* state valid unless told otherwise
  - in practice: failsafe-mechanisms (to remove orphaned state) in case of sender failure e.g., receiver-to-sender "heartbeat": is this state still valid?
- examples:
  - Q.2931 (ATM Signaling)
  - ST-II (Internet hard-state signaling protocol - outdated)
  - TCP

Soft-state

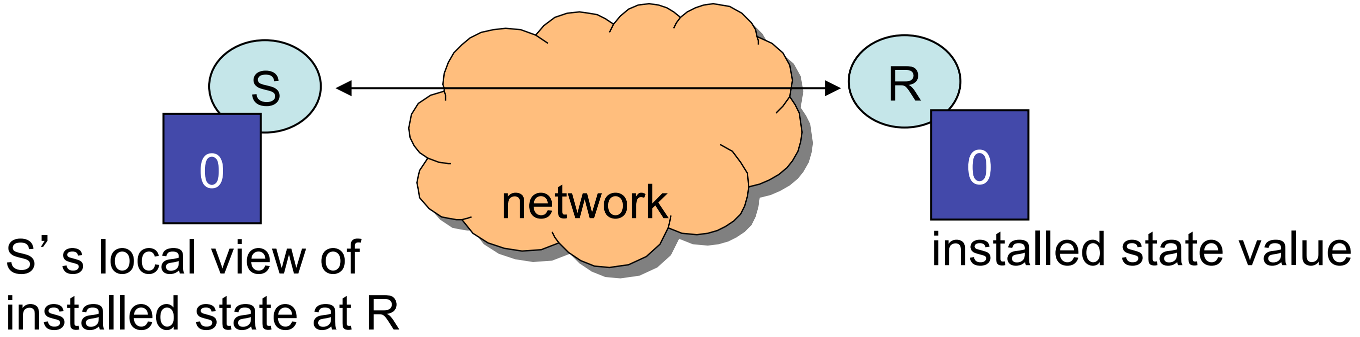
- state *installed* by receiver on receipt of setup (trigger) message from sender (typically, an endpoint)
  - sender also sends periodic *refresh* message: indicating receiver should continue to maintain state
- state *removed* by receiver via timeout, in absence of refresh message from sender
- default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

State: senders, receivers

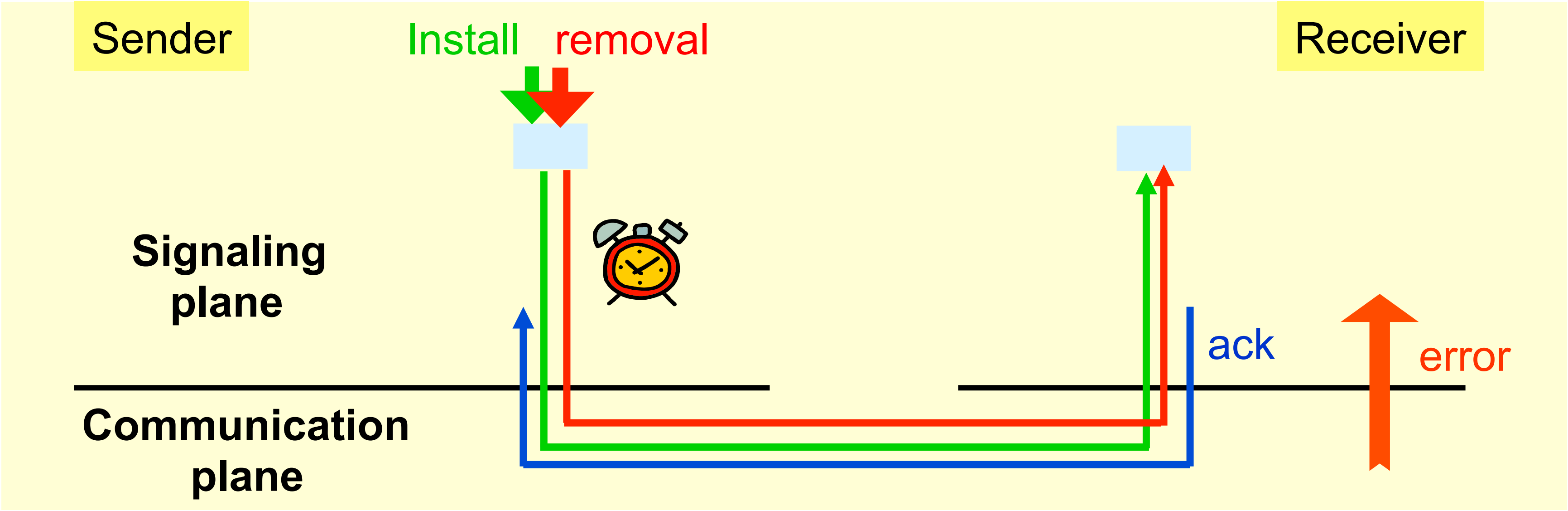
- sender: network node that *(re)generates* signaling (control) messages to install, keep-alive, remove state from other nodes
- receiver: node that creates, maintains, removes state based on signaling messages *received* from sender

Let's build a signaling protocol

- *S:* state *S*ender (state installer)
- *R:* state *R*eceiver (state holder)
- desired functionality:
  - S: set values in R to 1 when state "installed", set to 0 when state "not installed"
  - if other side is down, state is not installed (0)
  - initial condition: state not installed

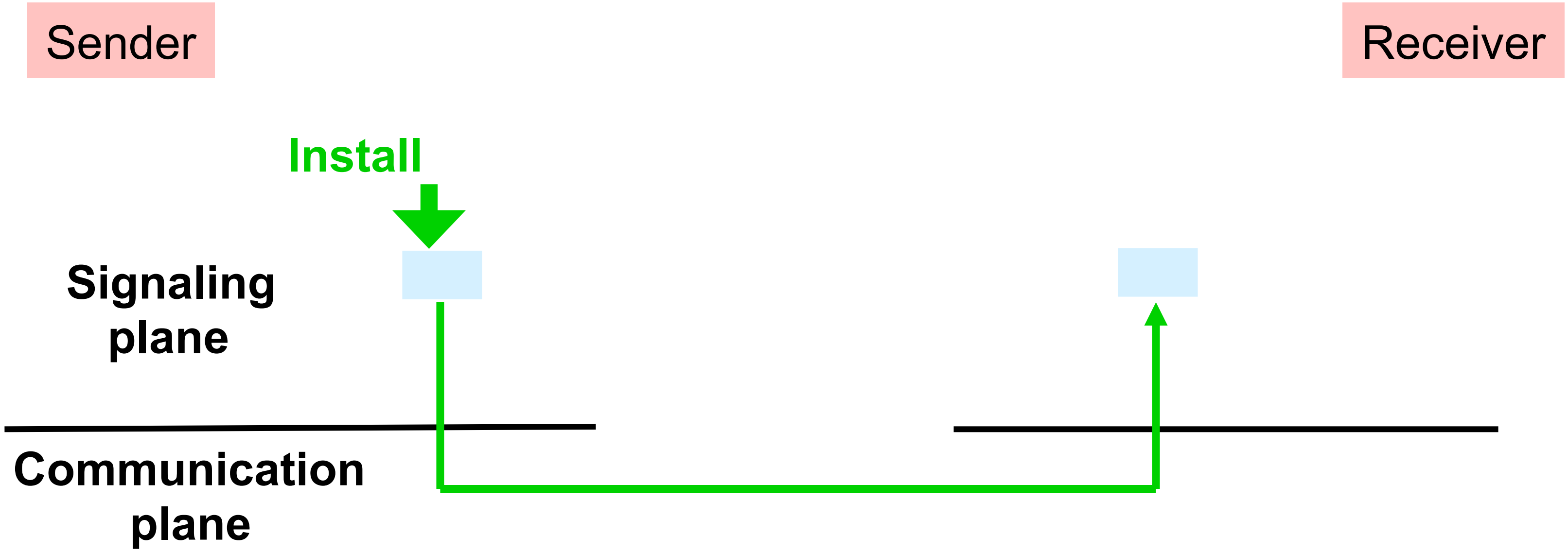


Hard-state signaling



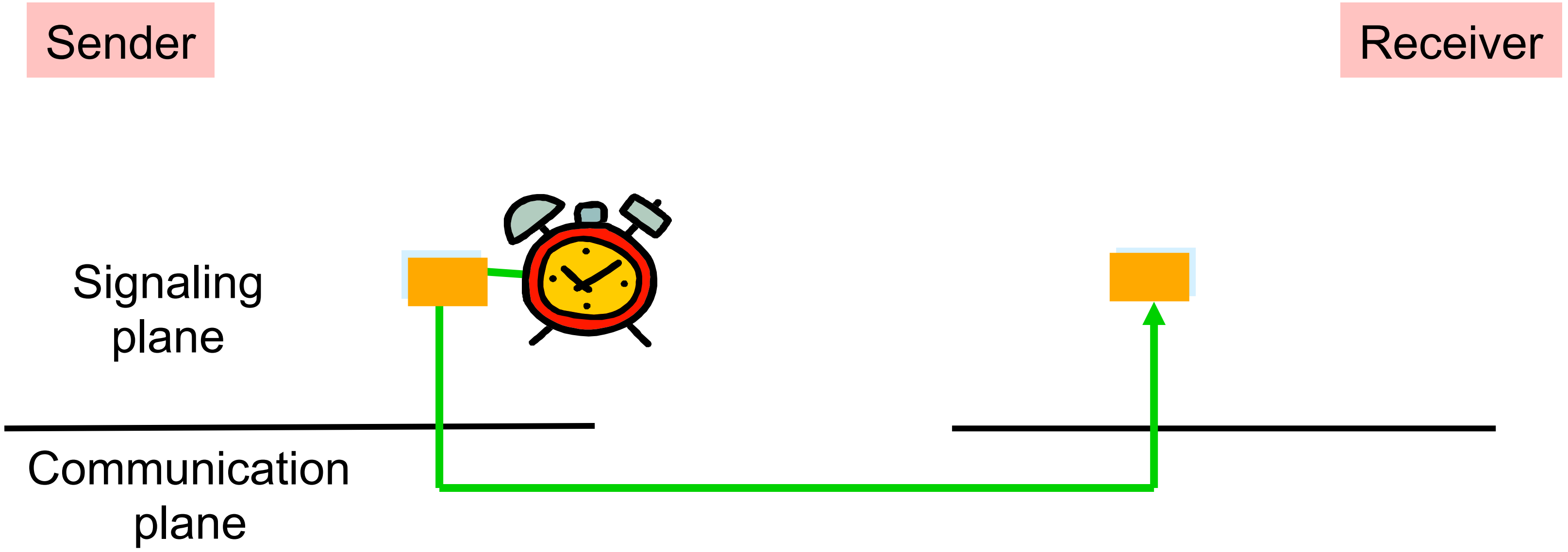
- reliable signaling
- state removal by request
- requires additional error handling
  - e.g., sender failure

Soft-state signaling



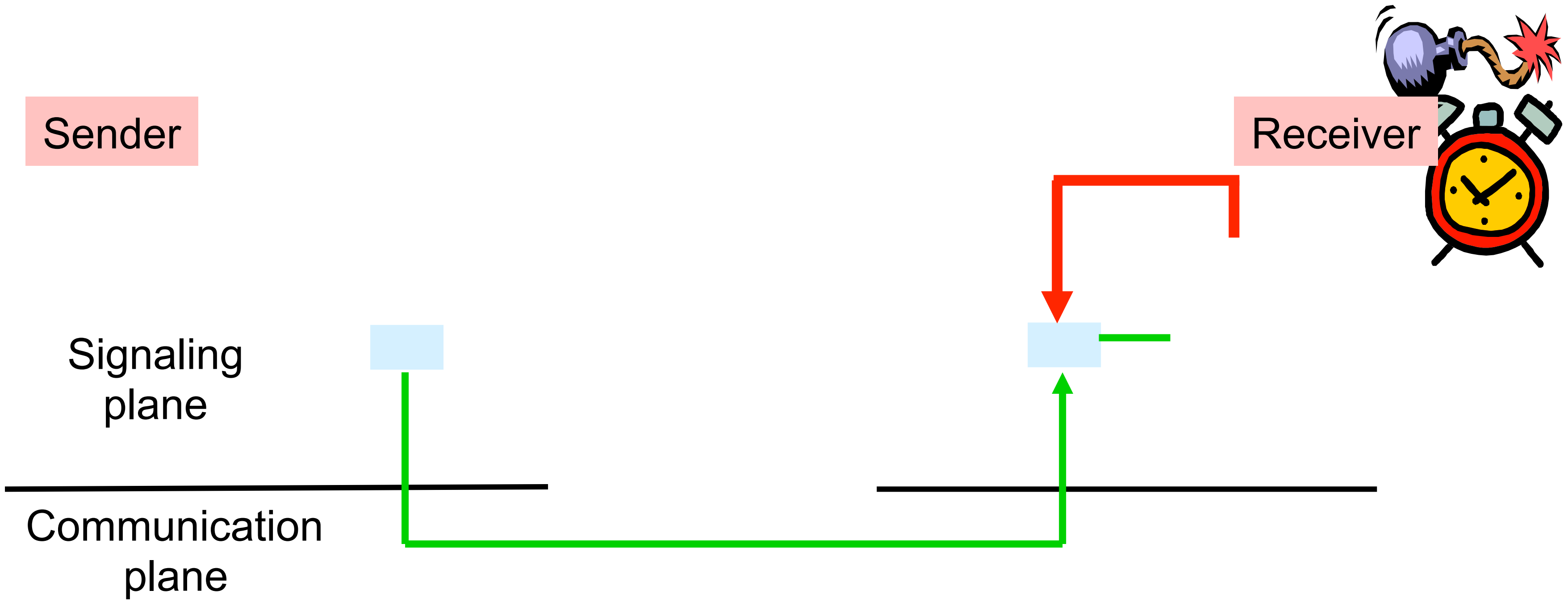
- best effort signaling

Soft-state signaling



- best effort signaling
- refresh timer, periodic refresh

Soft-state signaling



- best effort signaling
- refresh timer, periodic refresh
- state time-out timer, state removal only by time-out

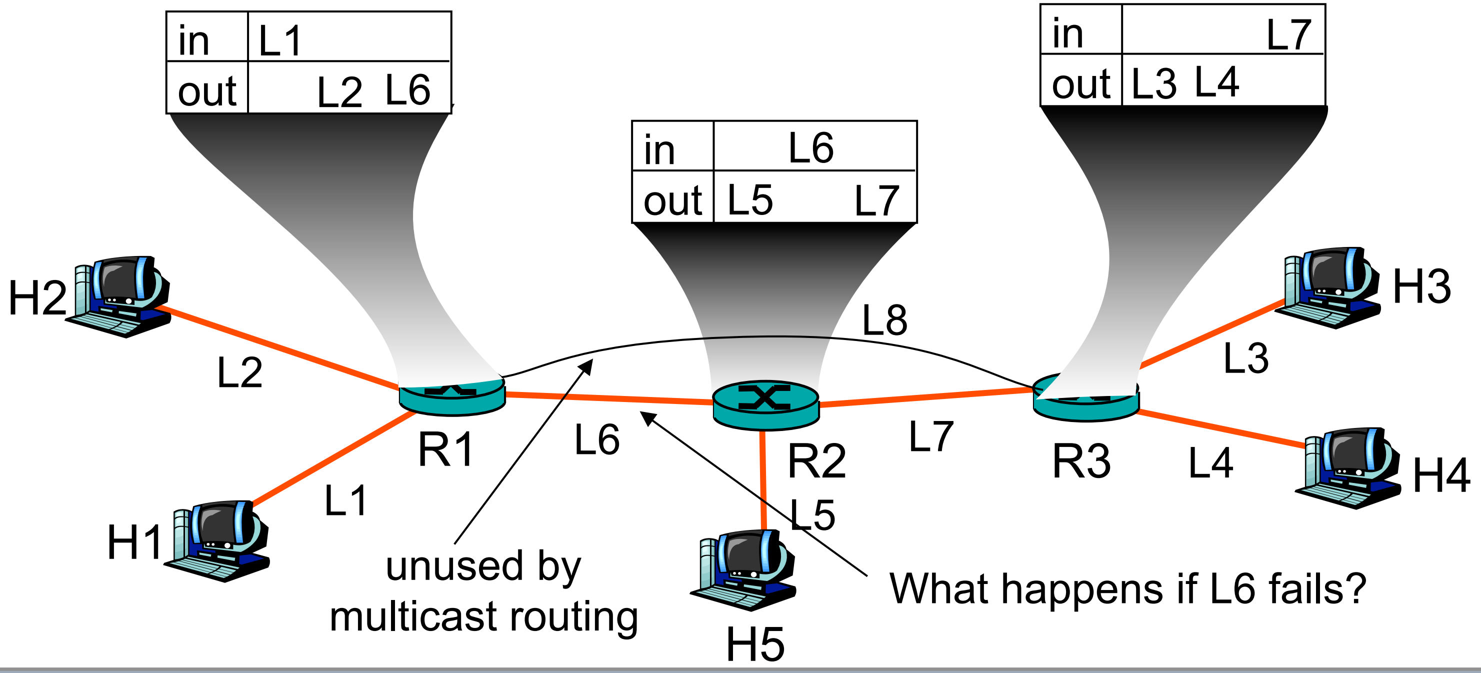
Soft-state: claims

- ❑ "Systems built on soft-state are robust" [Raman 99]
- ❑ "Soft-state protocols provide .. greater robustness to changes in the underlying network conditions…" [Sharma 97]
- ❑ "obviates the need for complex error handling software" [Balakrishnan 99]

What does this mean?

Soft-state: “easy” handling of changes

- Periodic refresh: if network “conditions” change, refresh will re-establish state under new conditions
- example: RSVP/routing interaction: if routes change (nodes fail) RSVP PATH refresh will *re-establish* state along new path



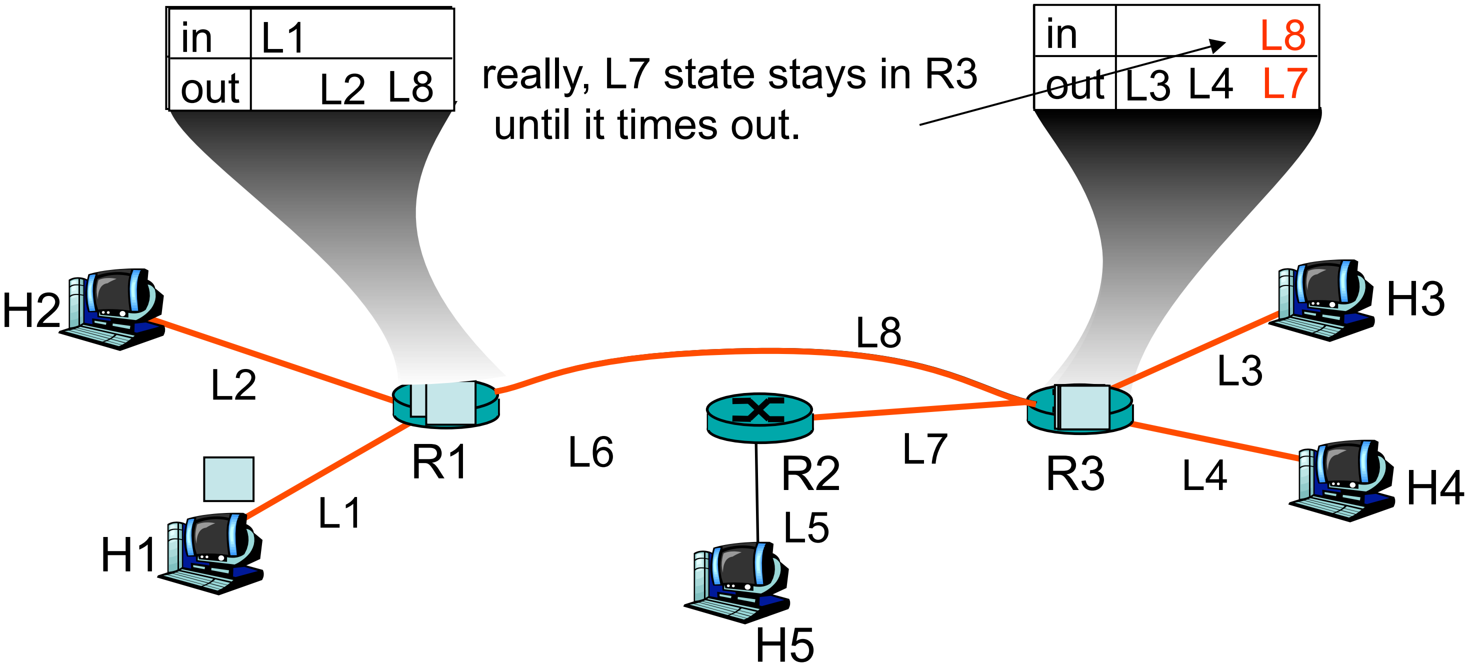
in	L1
out	L2 L6

in	L6
out	L5 L7

in	L7
out	L3 L4

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



in	L1
out	L2 L8

really, L7 state stays in R3 until it times out.

in	L8
out	L3 L4 L7

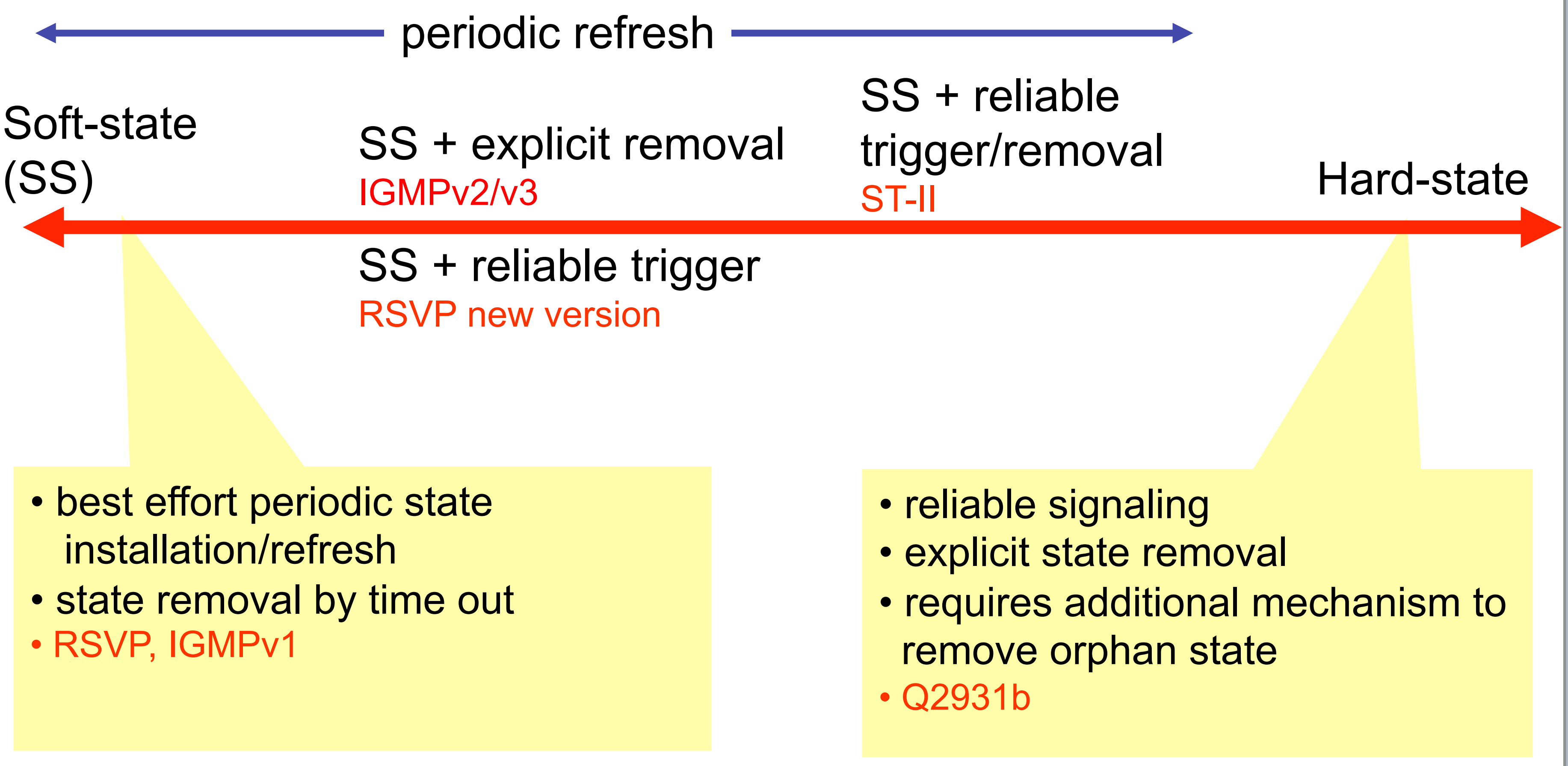
Soft-state: “easy” handling of changes

- “recovery” performed transparently to end-system by normal refresh procedures
- no need for network to signal failure/change to end system, or end system to respond to specific error
- less signaling (volume, types of messages) than hard-state from network to end-system but…
- more signaling (volume) than hard-state from end-system to network for refreshes

Soft-state: refreshes

- refresh messages serve many purposes:
  - trigger: first time state-installation
  - refresh: refresh state known to exist (“I am still here”)
  - <lack of refresh>: remove state (“I am gone”)
- challenge: all refresh messages unreliable
  - problem: what happens if first PATH message gets lost?
    - copy of PATH message only sent after refresh interval
  - would like triggers to result in state-installation a.s.a.p.
  - enhancement: add receiver-to-sender refresh_ACK for triggers
  - sender initiates retransmission if no refresh_ACK is received after short timeout
  - e.g., see paper “Staged Refresh Timers for RSVP” by Ping Pan and Henning Schulzrinne
  - approach also applicable to other soft-state protocols

Signaling Spectrum



periodic refresh

Soft-state (SS)

SS + explicit removal
IGMPv2/v3

SS + reliable trigger/removal
ST-II

Hard-state

SS + reliable trigger
RSVP new version

- 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

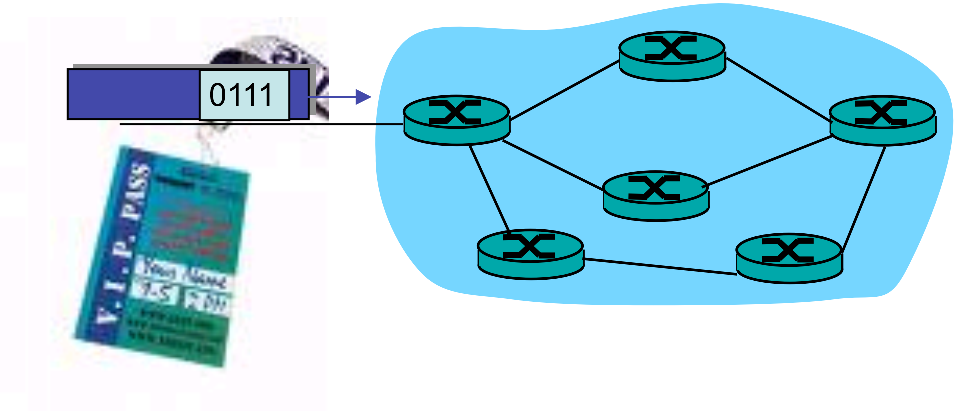
Chapter:
Quality of Service Support

Chapter outline – Quality-of-Service Support

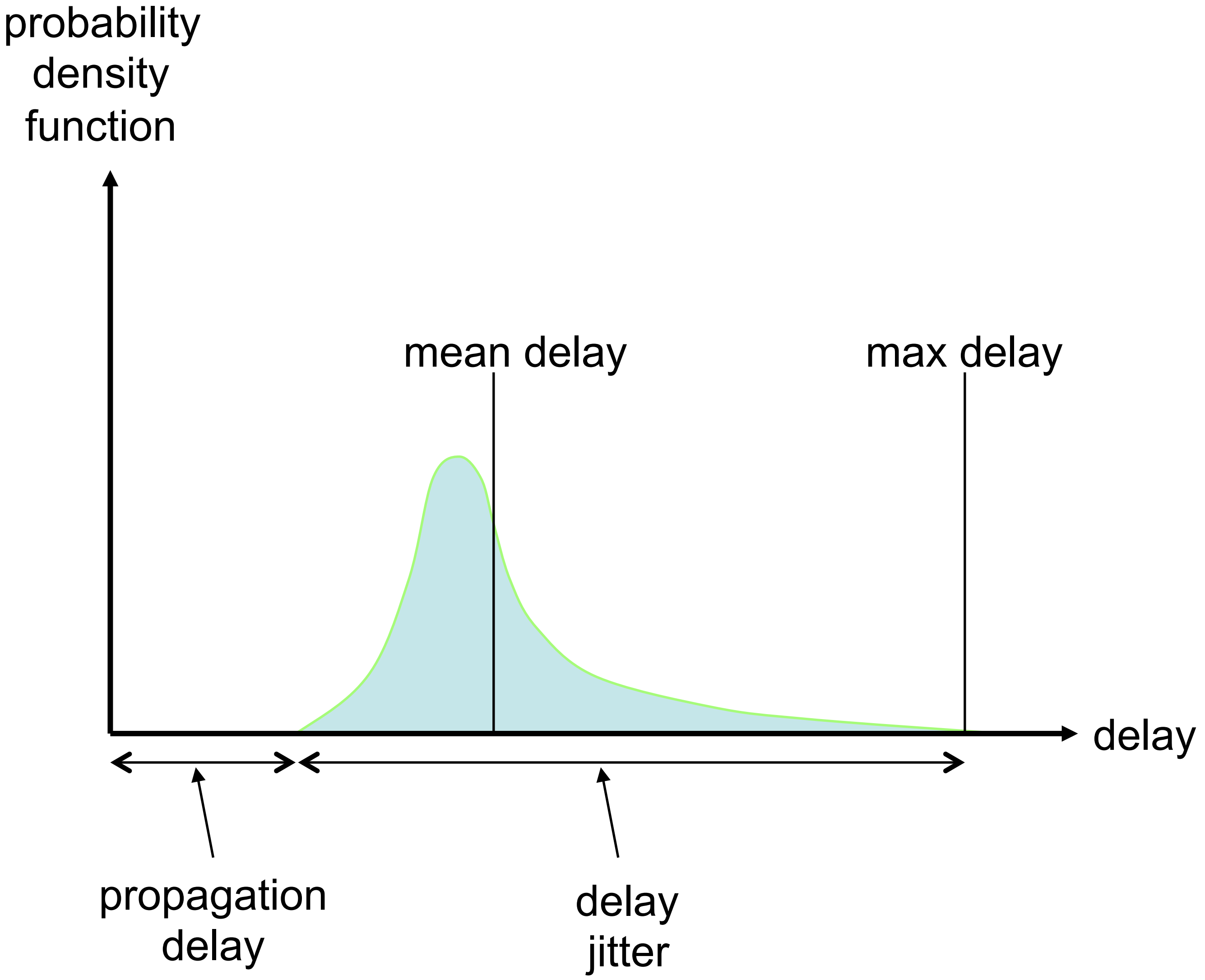
- ❑ Providing multiple classes of service
- ❑ Providing QoS guarantees
- ❑ **Signalling for QoS**

Providing Multiple Classes of Service

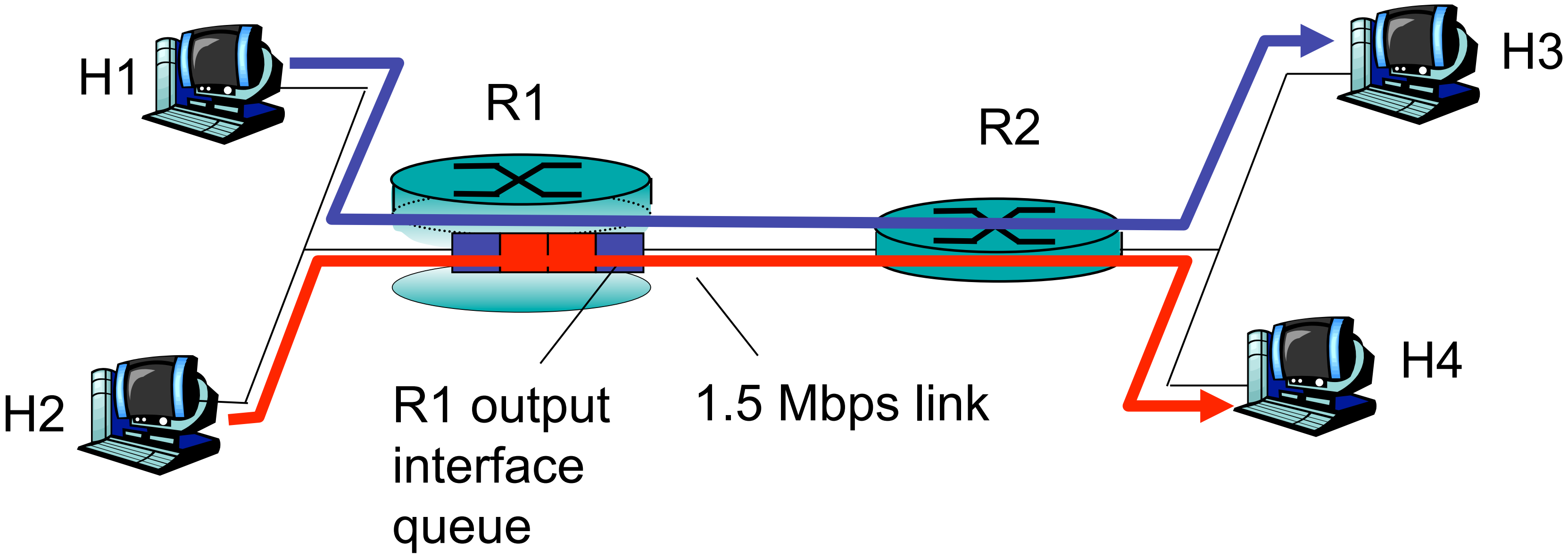
- Traditional Internet approach: making the best of best effort service
  - one-size fits all service model
- Alternative approach: multiple classes of service
  - partition traffic into classes
  - network treats different classes of traffic differently (analogy: VIP service vs regular service)
- granularity: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



Delay Distributions

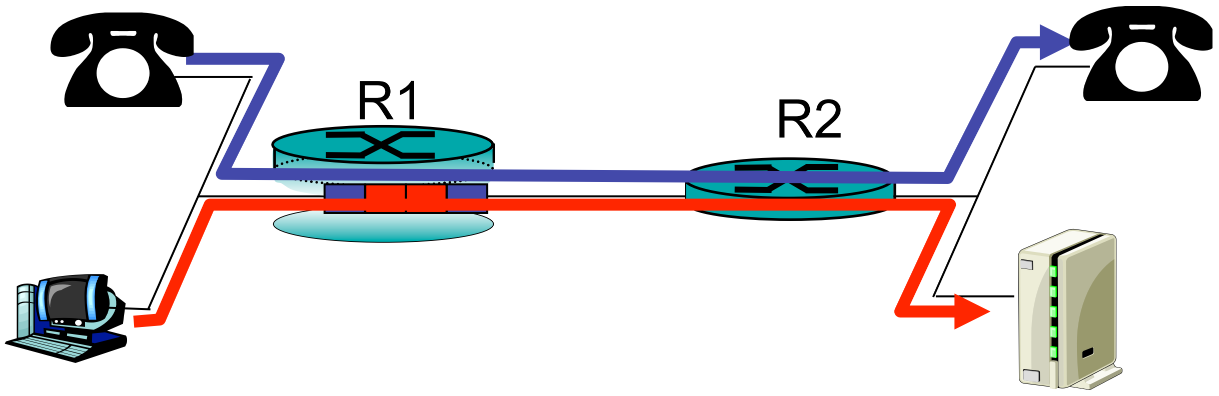


Multiple classes of service: scenario



Scenario 1: mixed FTP and audio

- Example: 1Mbps IP phone, FTP or NFS share 1.5 Mbps link.
  - bursts of FTP or NFS can congest router, cause audio loss
  - want to give priority to audio over FTP

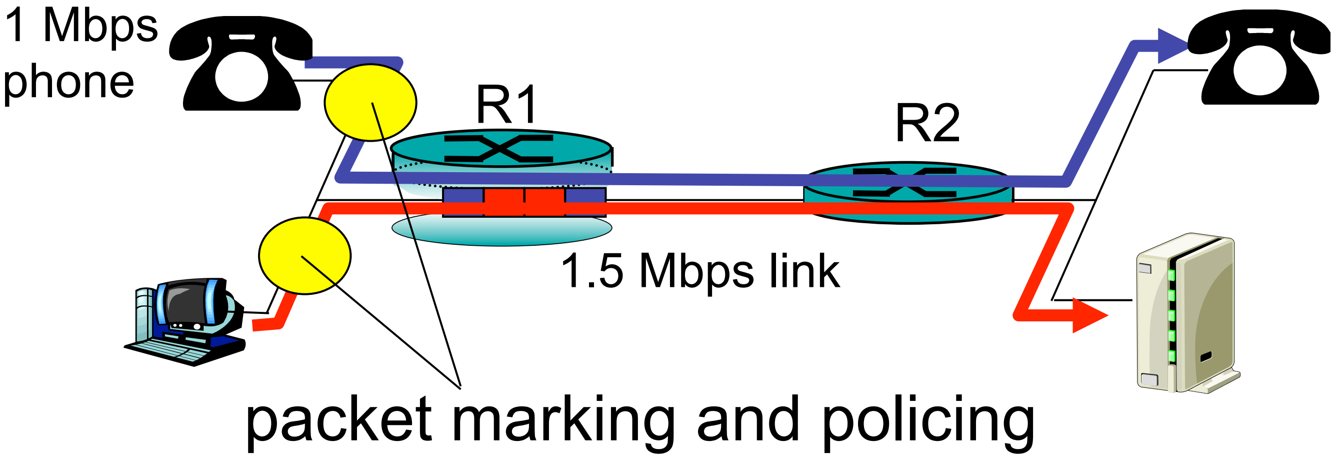


Principle 1

packet marking needed for router to distinguish between different classes; and new router policy to treat packets accordingly

Principles for QOS Guarantees (more)

- what if applications misbehave (audio sends higher than declared rate)
  - policing: force source adherence to bandwidth allocations
- marking and policing at network edge:
  - similar to ATM UNI (User Network Interface)

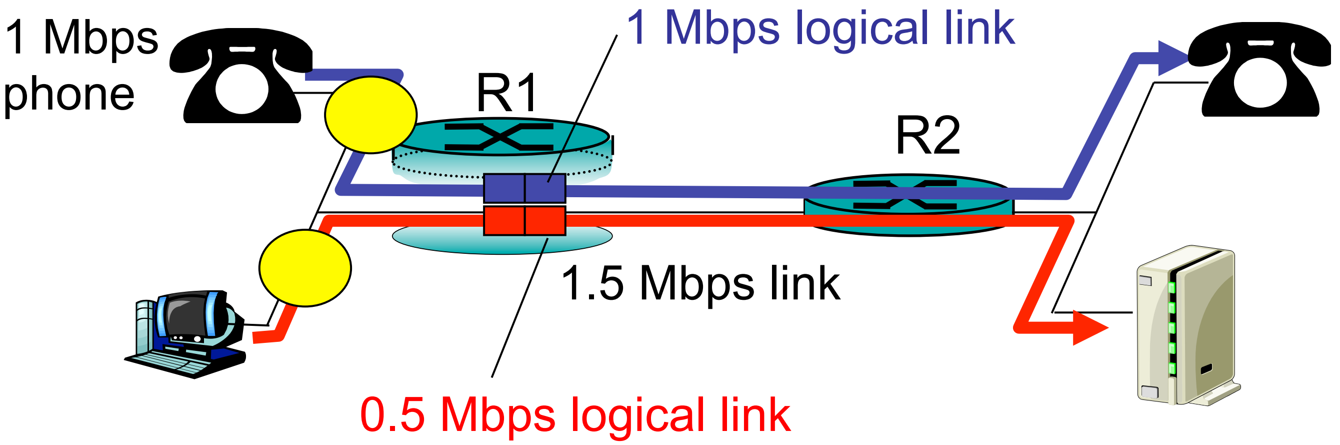


Principle 2

provide protection (*isolation*) for one class from others

Principles for QOS Guarantees (more)

- Allocating *fixed* (non-sharable) bandwidth to flow: *inefficient* use of bandwidth if flows doesn't use its allocation

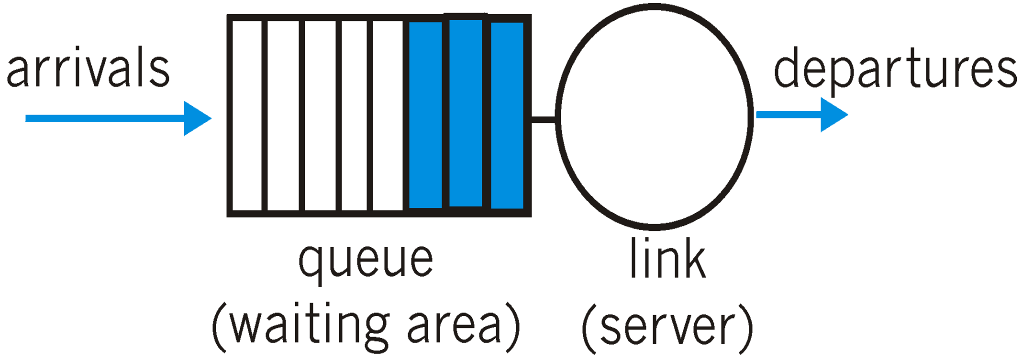


Principle 3

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

Scheduling And Policing Mechanisms

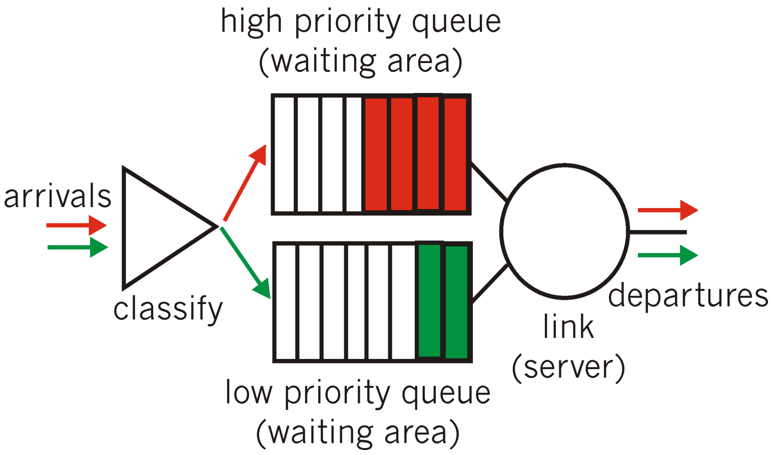
- scheduling: choose next packet to send on link
- FIFO (first in first out) scheduling: send in order of arrival to queue
  - ⇨real-world example?
  - discard policy: if packet arrives to full queue: who to discard?
    - Tail drop: drop arriving packet
    - priority: drop/remove on priority basis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

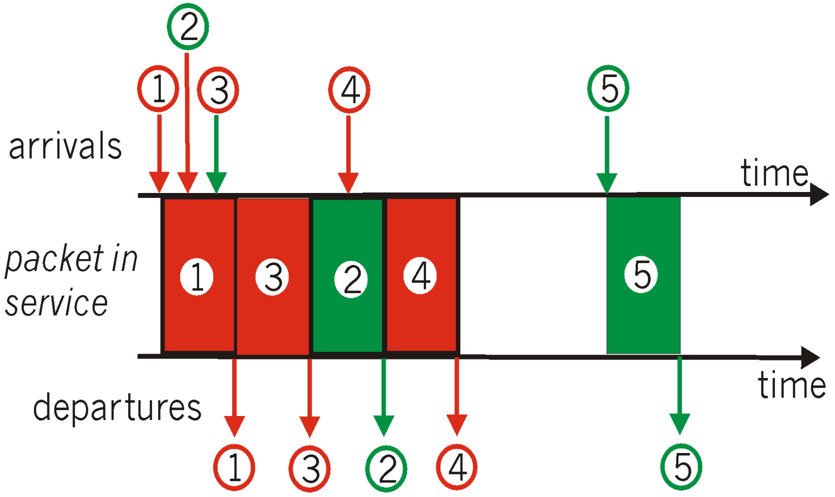


Scheduling Policies: more

Priority scheduling: transmit highest priority queued packet

- multiple *classes*, with different priorities
  - class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc..

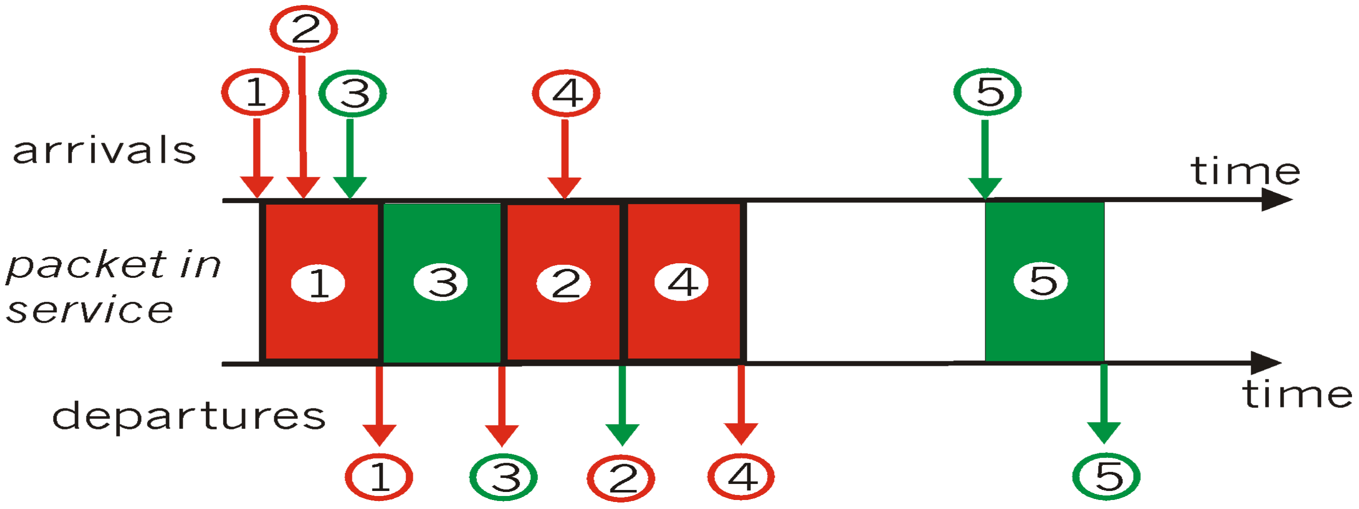




Scheduling Policies: still more

round robin scheduling:

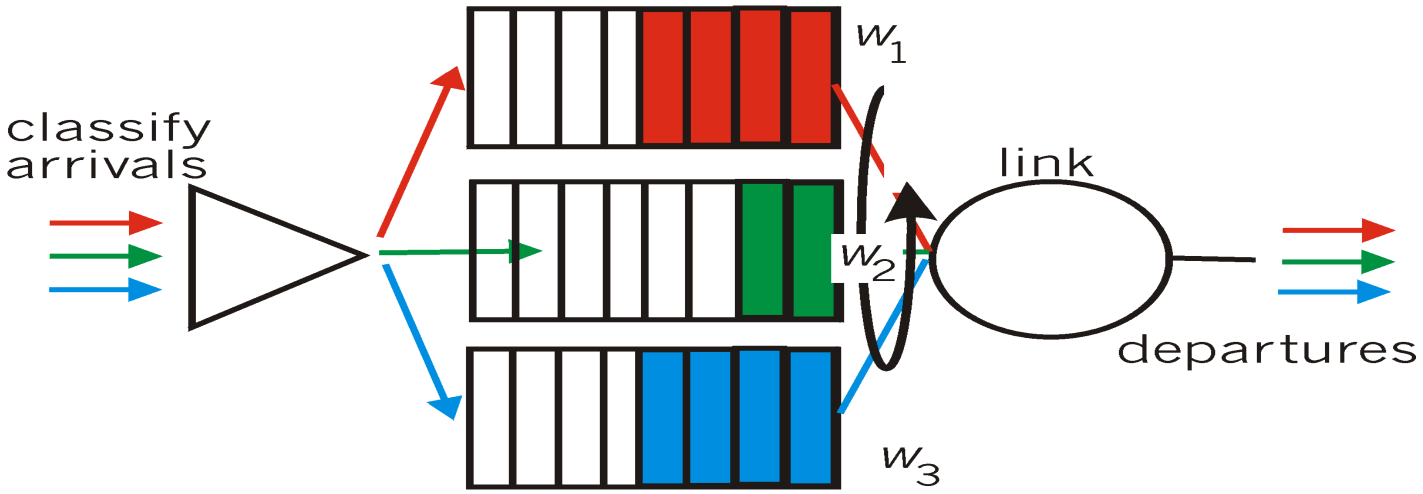
- multiple classes
- cyclically scan class queues, serving one from each class (if available)



Scheduling Policies: still more

Weighted Fair Queuing:

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



Policing Mechanisms

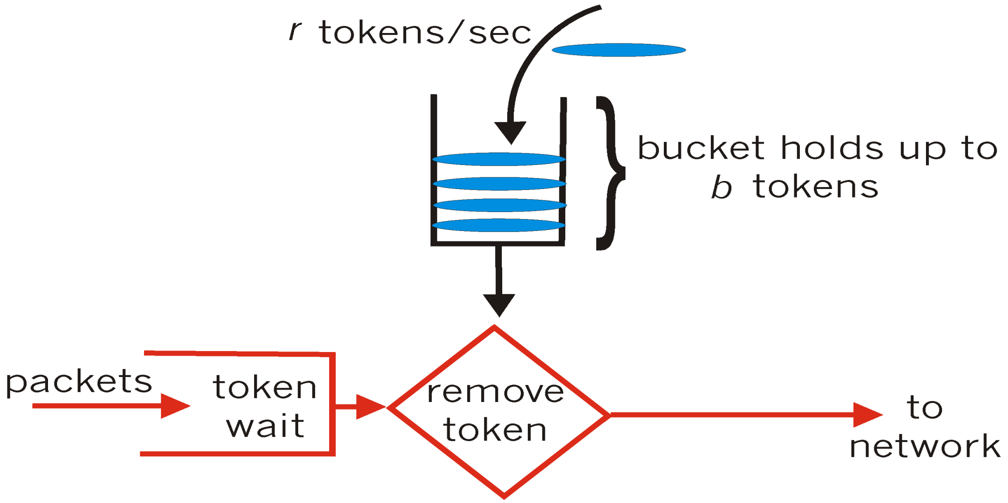
Goal: limit traffic to not exceed declared parameters

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

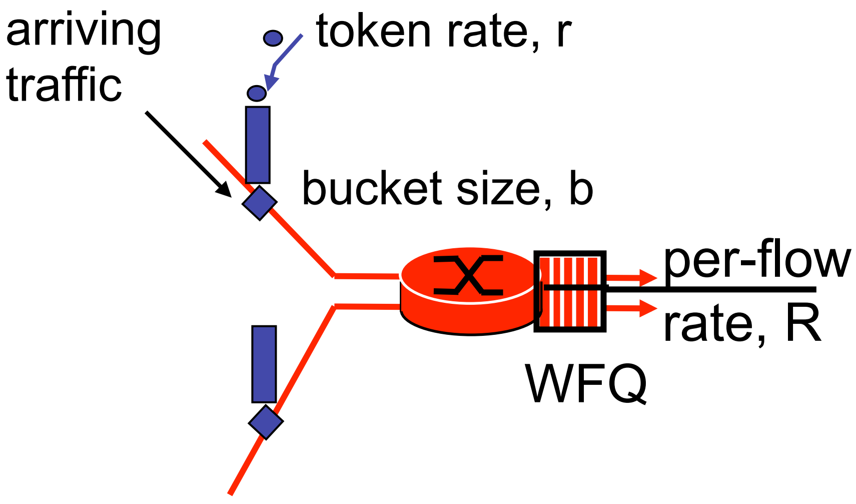
Token Bucket: limit input to specified Burst Size and Average Rate.



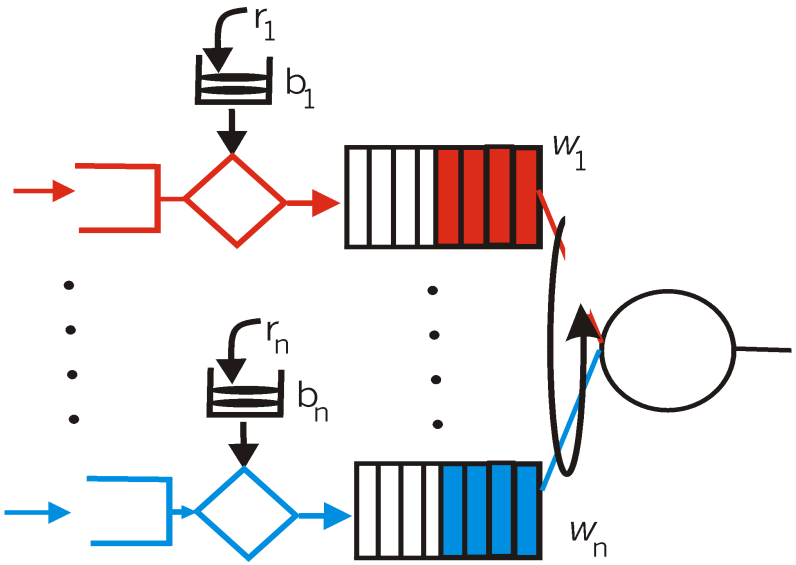
- bucket can hold b tokens ⇒ limits maximum burst size
- tokens generated at rate *r token/sec* unless bucket full
- *over interval of length t: number of packets admitted less than or equal to (r t + b).*

Policing Mechanisms (more)

- token bucket, WFQ combined provide guaranteed upper bound on delay, i.e., *QoS guarantee*



$D_{max} = b/R$



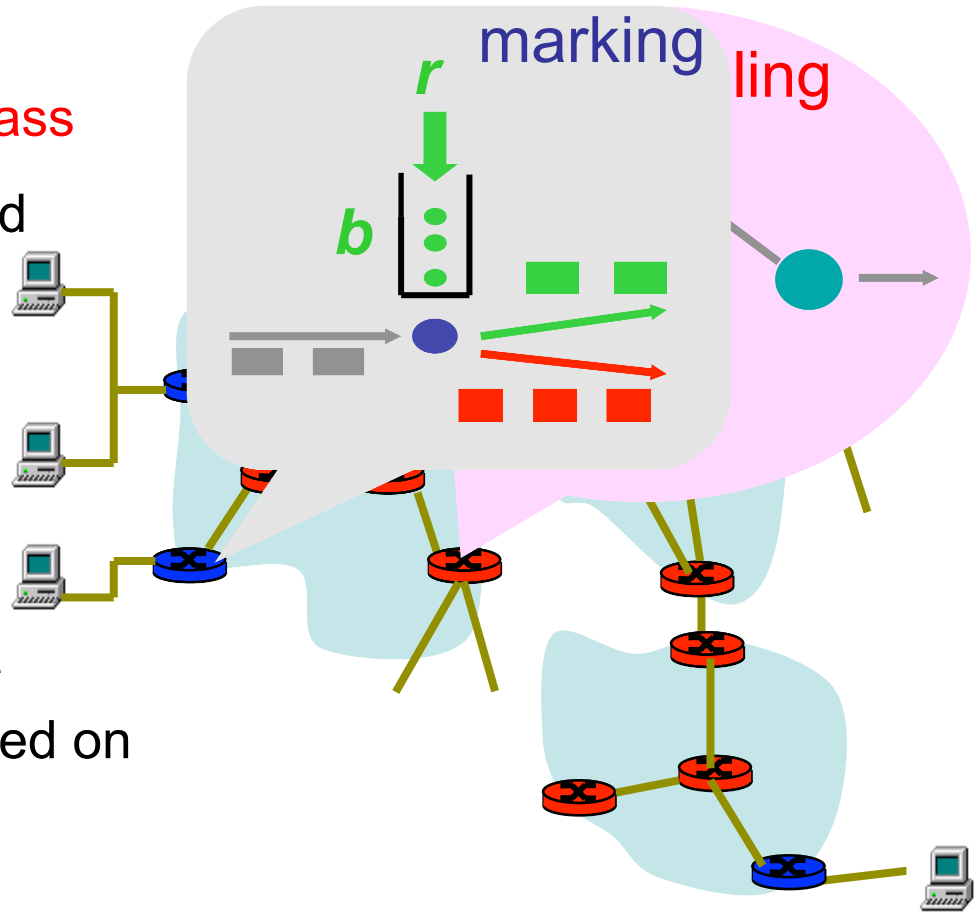
IETF Differentiated Services

- want “qualitative” service classes
  - “behaves like a wire”
  - relative service distinction: Platinum, Gold, Silver
- *scalability:* simple functions in network core, relatively complex functions at edge routers (or hosts)
  - 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

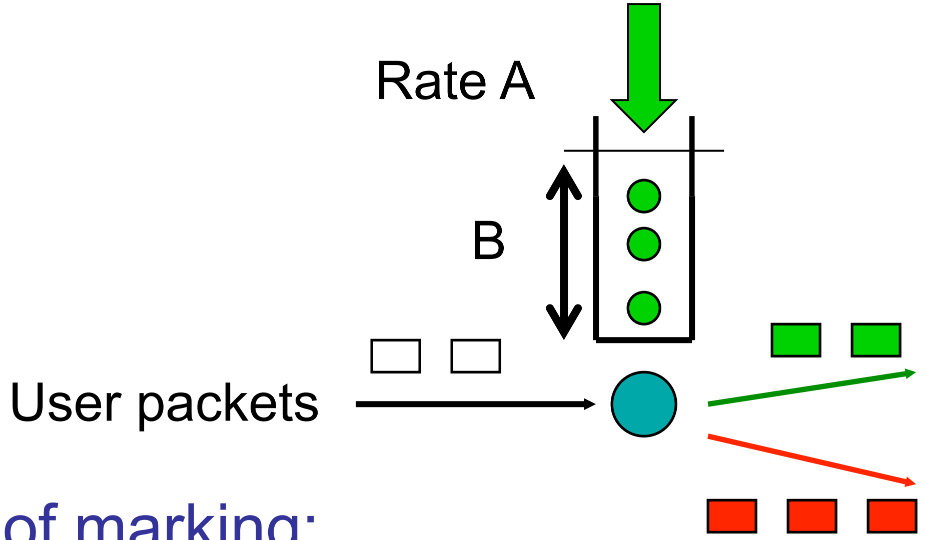


Core router:

- ❑ 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

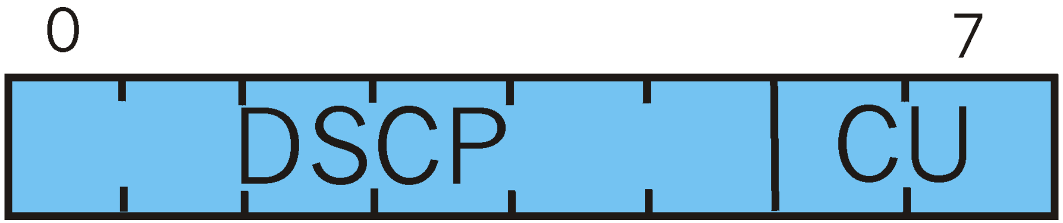


Possible usage of marking:

- 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

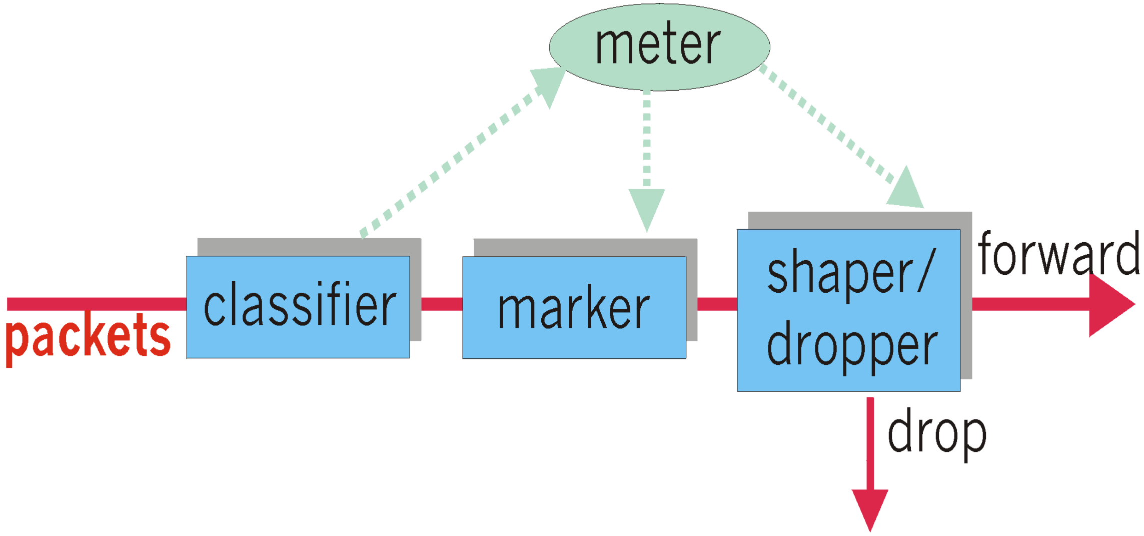
- Packet is marked in the Type of Service (TOS) in IPv4, and Traffic Class in IPv6
- 6 bits used for Differentiated Service Code Point (DSCP) and determine PHB that the packet will receive
- 2 bits can be used for congestion notification:
  Explicit Congestion Notification (ECN), RFC 3168



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:

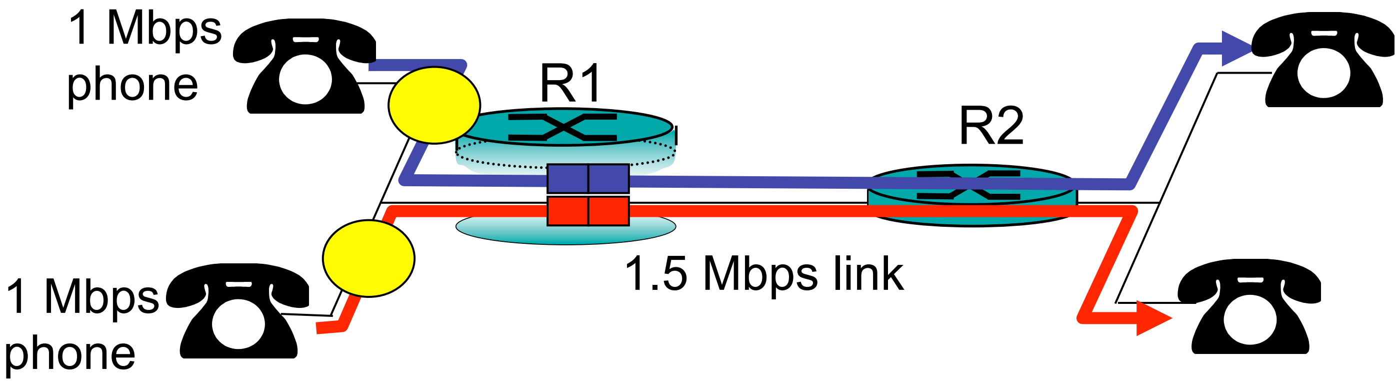
- Expedited Forwarding: packet departure rate of a class equals or exceeds specified rate
  - logical link with a minimum guaranteed rate
- Assured Forwarding: e.g. 4 classes of traffic
  - each class guaranteed minimum amount of bandwidth and a minimum of buffering
  - packets each class have one of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

Chapter outline – Quality-of-Service Support

- ❑ Providing multiple classes of service
- ❑ Providing QoS guarantees
- ❑ **Signalling for QoS**

Principles for QOS Guarantees (more)

- *Basic fact of life:* can not support traffic demands beyond link capacity

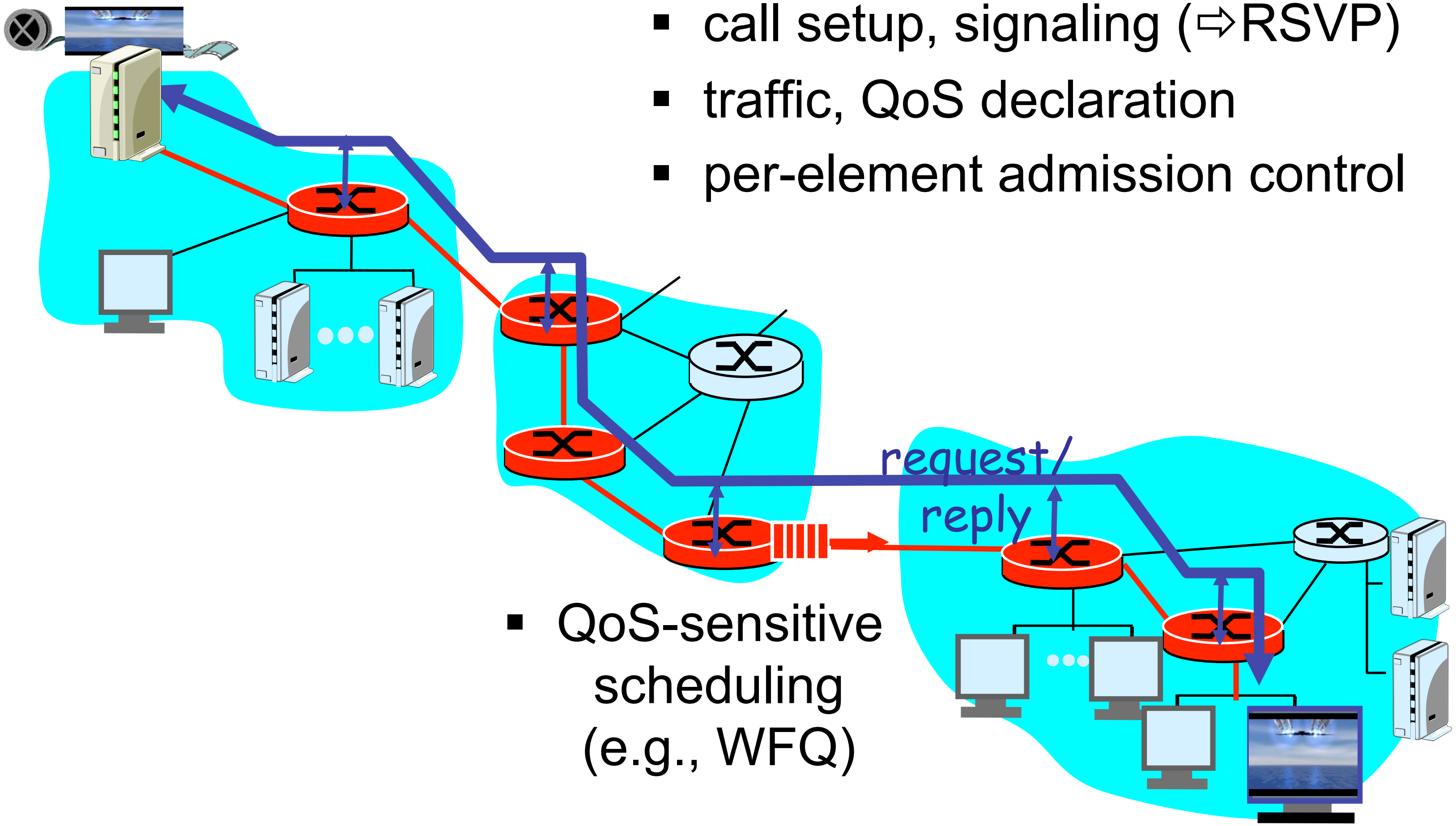


Principle

Call Admission: flow declares its needs, network may block call (e.g., busy signal) if it cannot meet needs

QoS Guarantee Scenario

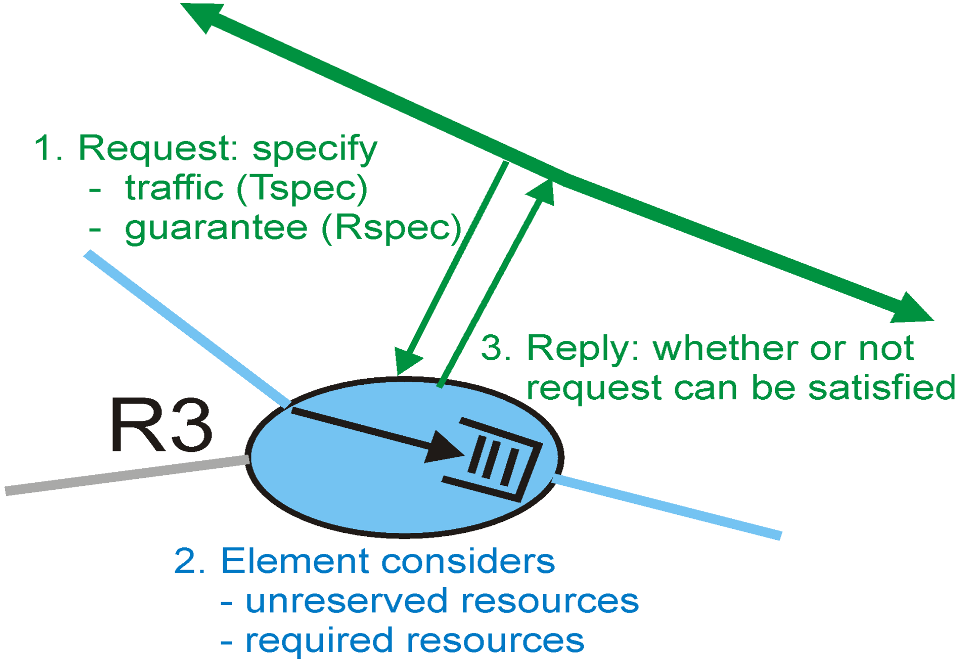
❑ Resource reservation
  - call setup, signaling (⇨RSVP)
  - traffic, QoS declaration
  - per-element admission control



- QoS-sensitive scheduling (e.g., WFQ)

Call Admission

- Routers will admit calls based on:
- Flow behavior:
  - R-spec and T-spec
- the current resource allocated at the router to other calls.



IETF Integrated Services

- architecture for providing QOS guarantees in IP networks for individual application sessions
- resource reservation: routers maintain state info (as for VCs) of allocated resources, QoS requests
- admit/deny new call setup requests:

Question: can newly arriving flow be admitted with performance guarantees while not violated QoS guarantees made to already admitted flows?

Call Admission

Arriving session must :

- declare its QoS requirement
  - R-spec: defines the QoS being requested
- characterize traffic it will send into network
  - T-spec: defines traffic characteristics
- signaling protocol: needed to carry R-spec and T-spec to routers (where reservation is required)
  - RSVP

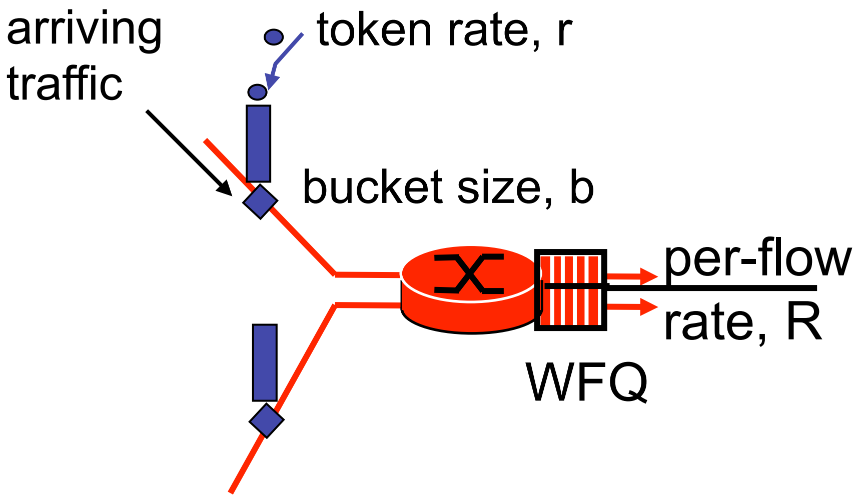
Intserv QoS: Service models [RFC 2211, RFC 2212]

Guaranteed service:

- ❑ worst case traffic arrival: leaky-bucket-policed source
- ❑ simple (mathematically provable) *bound* on delay [Parekh 1992, Cruz 1988]

Controlled load service:

- ❑ "a quality of service closely approximating the QoS that same flow would receive from an unloaded network element."



$$D_{max} = b/R$$

Chapter outline – Quality-of-Service Support

- Providing multiple classes of service
- Providing QoS guarantees
- **Signalling for QoS**

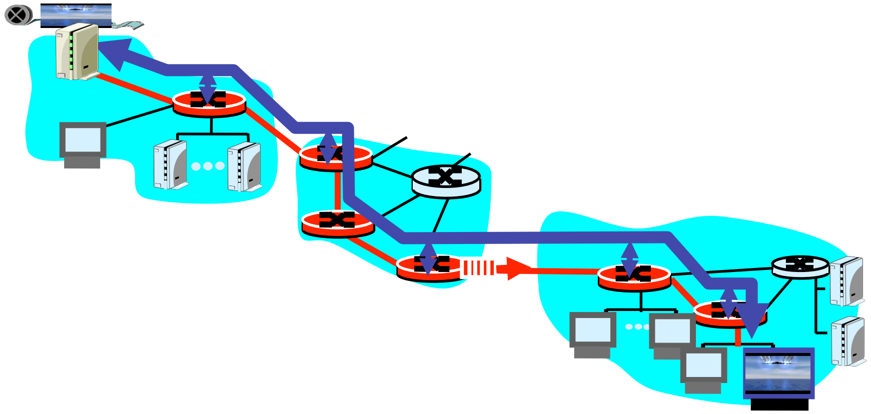
Signaling in the Internet

connectionless (stateless) forwarding by IP routers + best effort service = no network signaling protocols in initial IP design

- ❑ New requirement: reserve resources along end-to-end path (end system, routers) for QoS for multimedia applications
- ❑ RSVP: Resource Reservation Protocol [RFC 2205]
  - ▪ " … allow users to communicate requirements to network in robust and efficient way." i.e., signaling !
- ❑ earlier Internet Signaling protocol: ST-II [RFC 1819]

RSVP Design Goals

1. accommodate heterogeneous receivers (different bandwidth along paths)
2. accommodate different applications with different resource requirements
3. make multicast a first class service, with adaptation to multicast group membership
4. leverage existing multicast/unicast routing, with adaptation to changes in underlying unicast, multicast routes
5. control protocol overhead to grow (at worst) linear in # receivers
6. modular design for heterogeneous underlying technologies



RSVP: does not…

- specify *how* resources are to be reserved
  - rather: a mechanism for *communicating needs*
- determine routes packets will take
  - that' s the job of routing protocols
  - signaling decoupled from routing
- interact with forwarding of packets
  - separation of control (signaling) and data (forwarding) planes

RSVP: overview of operation

- senders, receiver join a multicast group
  - done outside of RSVP
  - senders need not join group
- sender-to-network signaling
  - *path message:* make sender presence known to routers
  - path teardown: delete sender's path state from routers
- receiver-to-network signaling
  - *reservation message:* reserve resources from sender(s) to receiver
  - reservation teardown: remove receiver reservations
- network-to-end-system signaling
  - path error
  - reservation error

RSVP Messages

- There are two primary types of messages:
- Path messages (*path*)
  - The *path* message is sent from the sender host along the data path and stores the *path state* in each node along the path.
  - The *path state* includes the IP address of the previous node, and some data objects:
    - *sender template* to describe the format of the sender data
    - *sender tspec* to describe the traffic characteristics of the data flow
    - *adspec* that carries advertising data (see RFC 2210 for more details).
- Reservation messages (*resv*)
  - The *resv* message is sent from the receiver to the sender host along the reverse data path. At each node the IP destination address of the *resv* message will change to the address of the next node on the reverse path and the IP source address to the address of the previous node address on the reverse path.
  - The *resv* message includes the *flowspec* data object that identifies the resources that the flow needs.

Chapter:

Network Resilience

Overview

- Terminology
- Challenges in the current Internet
- Resilience Mechanisms

Overview

- **Terminology**
- Challenges in the current Internet
- Resilience Mechanisms

Terminolgy - Overview

- ❑ The "*fault* ➔ *error* ➔ *failure*" chain
- ❑ Fault tolerance
- ❑ Resilience
- ❑ Dependability
- ❑ Security
- ❑ 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

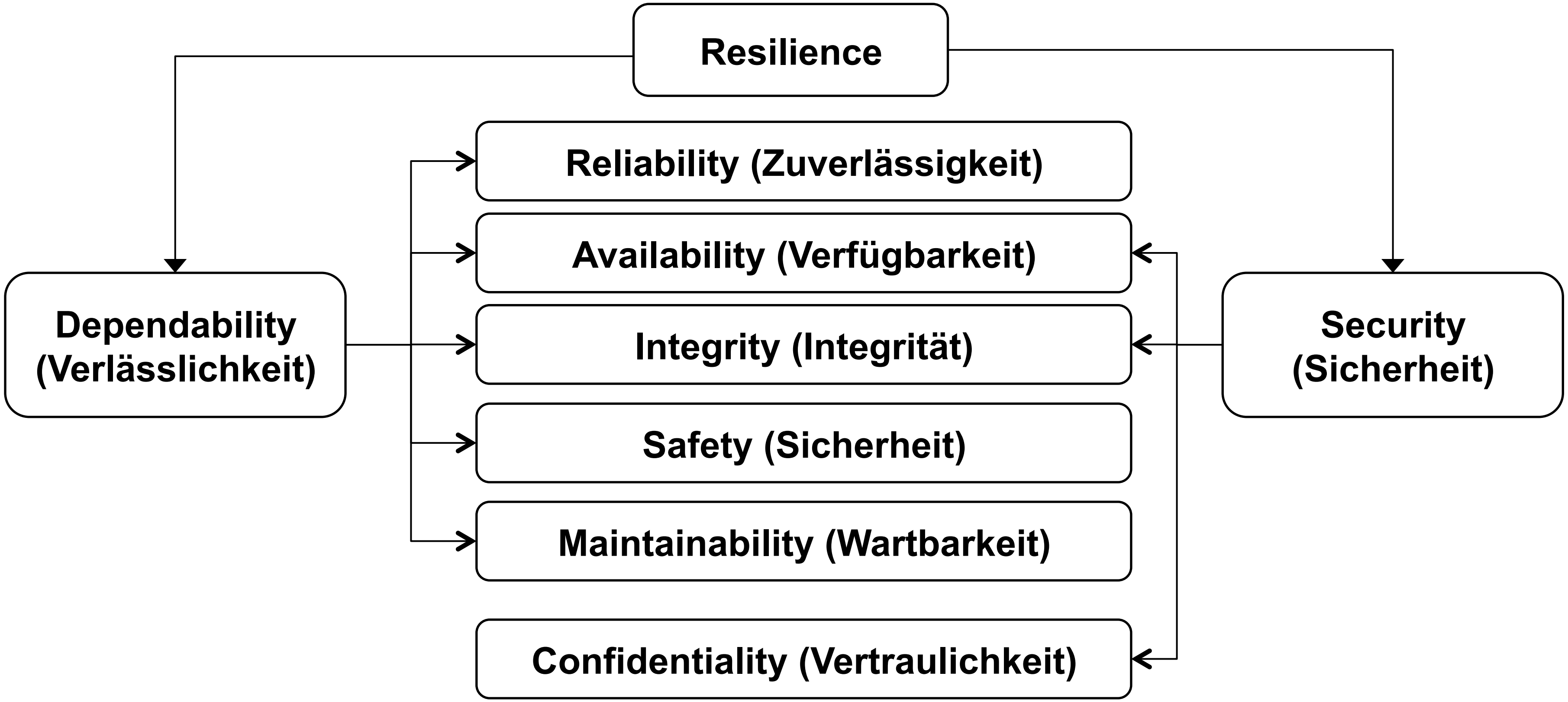
- Origin
  - Latin verb: “resilire” ~ jump back
- Resilience definition in different fields
  - Physics
    - A material’ s property of being able to recover to a normal state after a deformation resulting from external forces;
  - Ecology
    - Moving from a stability domain to another under the influence 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

  $\mathrm{A}(t) = Pr$ [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

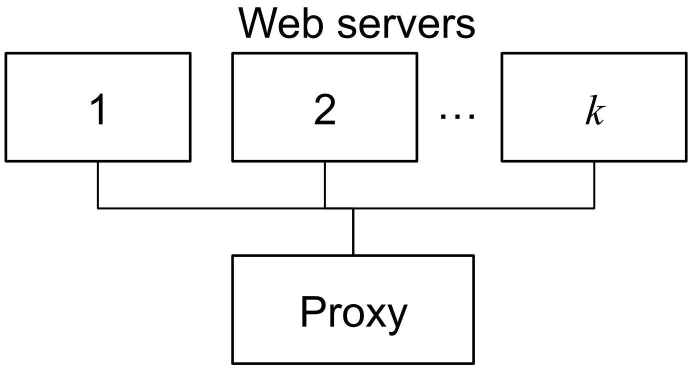
- ❑ DNS lookup (stateless service)
  - MTTF: 30 min
  - MTTR: 1 ms
  - A_{avg} = 0.998

☞ 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



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

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

- ❑ Same holds for the availability

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

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

Overview

- ❑ Terminology
- ❑ **Challenges in the current Internet**
- ❑ Resilience Mechanisms

Challenges in the current Internet

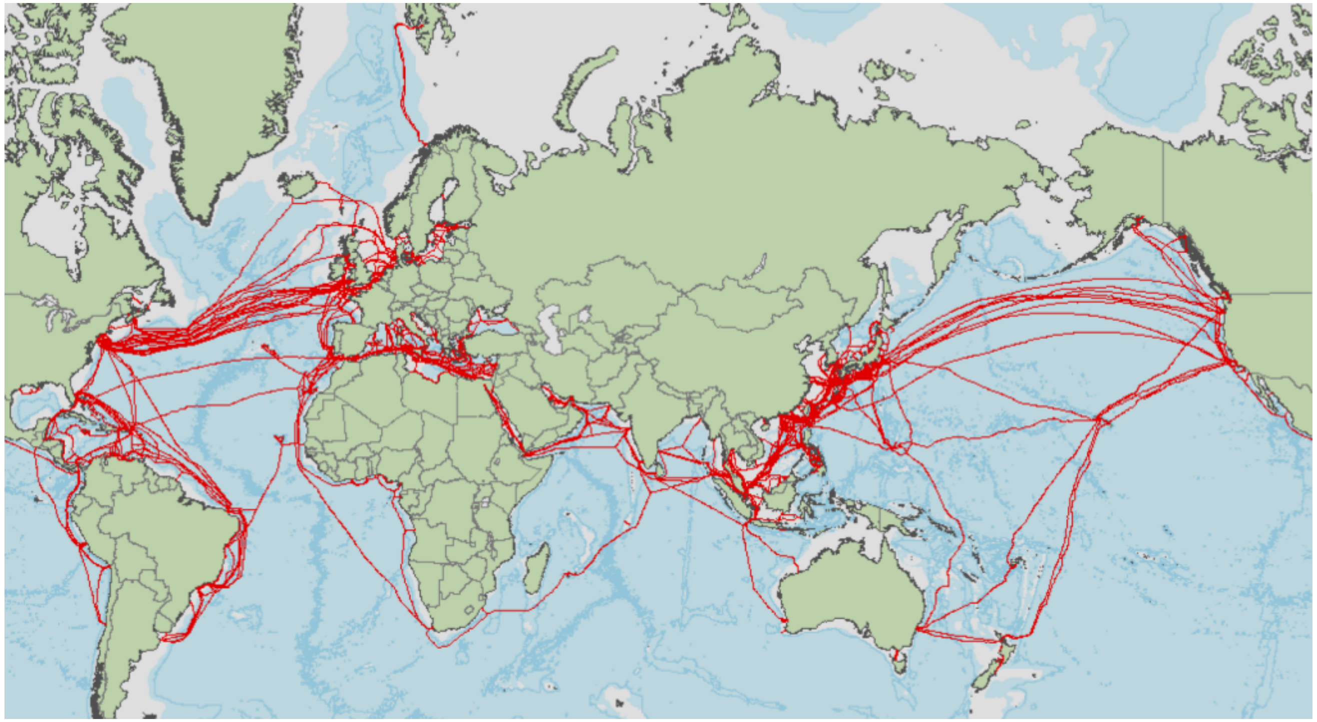
- Topology Failures
- Overload
- Lack of Integrity
- Software Faults
- Domino Effects

Topology Failures

- ❑ Failures in the “network graph”
- ❑ Network graph
  - ▪ Physical topology
  - ▪ Logical topology including service dependencies, e.g., DNS

    ☞ *Dependency graphs*

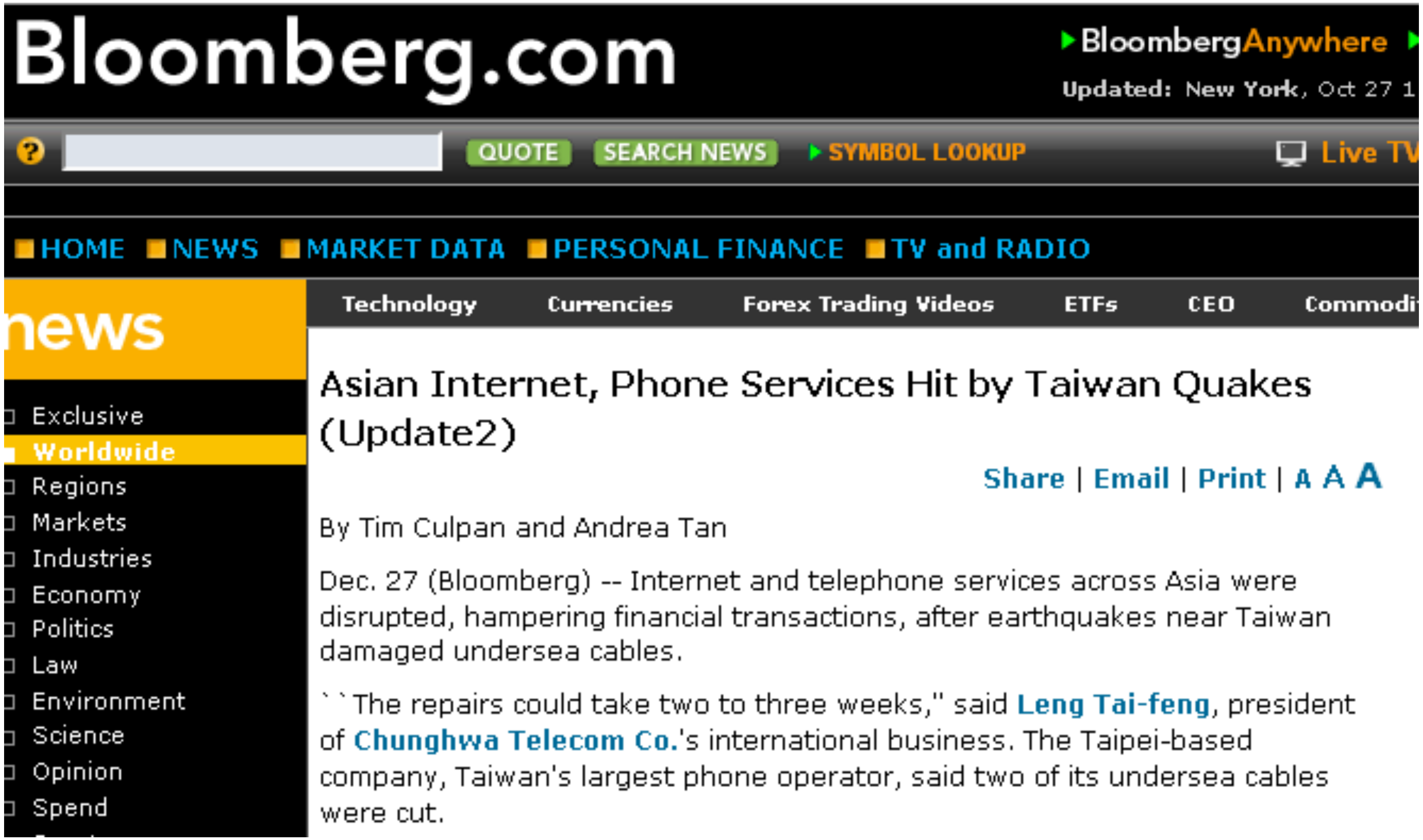
Sub-Marine Cables



- ❑ ~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.

Topological Failures; Routing

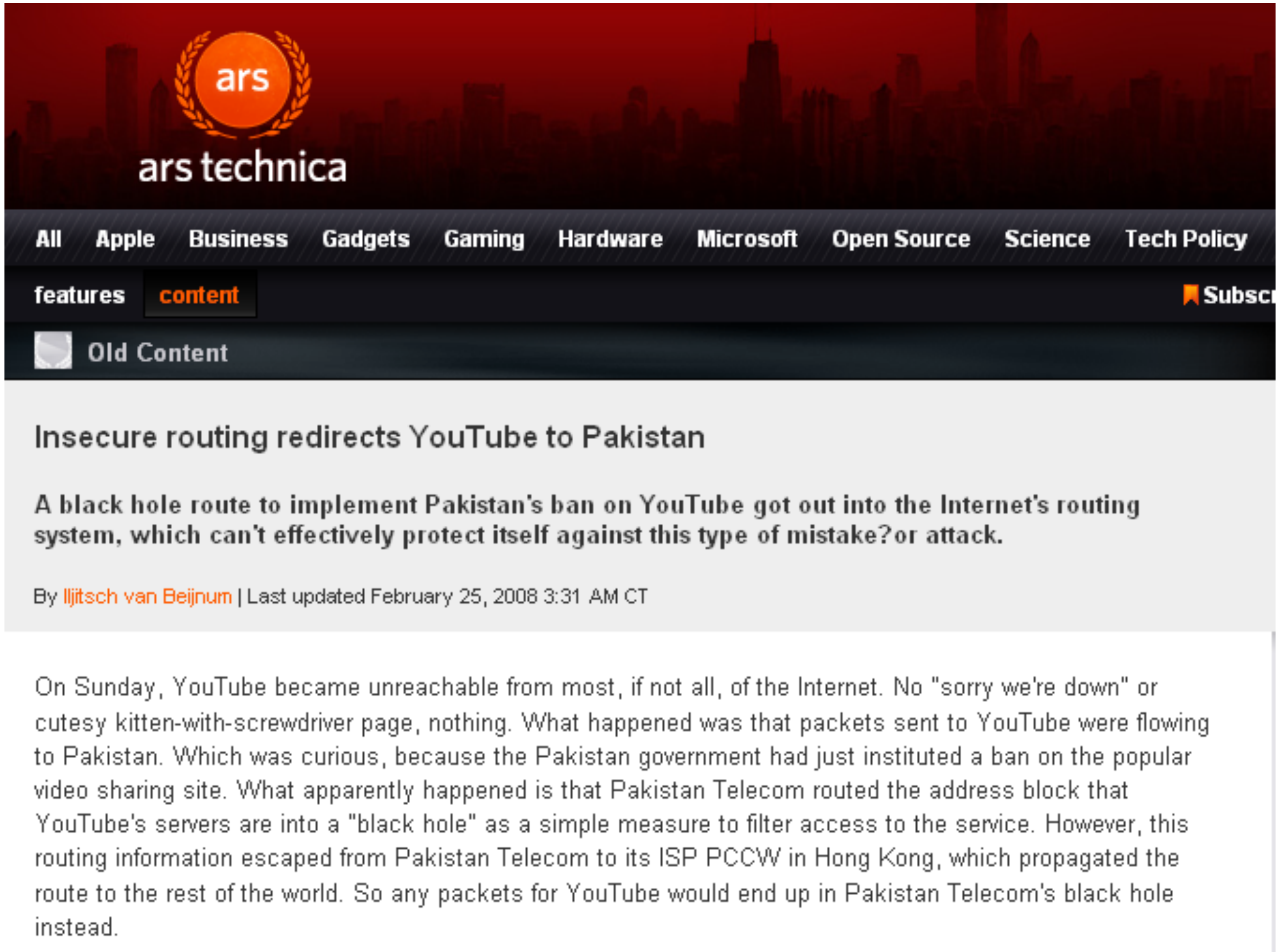
- ❑ Failures in the IP topology graph
  - ▪ Failures of routers (nodes)
  - ▪ Failure of links between routers
- ❑ Failure of links between routers generally caused by disconnection at lower layers
- ❑ Failure of routers
  - ▪ DoS attacks
  - ▪ Failures due to software bugs
  - ▪ Examples of reported 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


ars
ars technica
All Apple Business Gadgets Gaming Hardware Microsoft Open Source Science Tech Policy
features content
Subsc
Old Content

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.

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

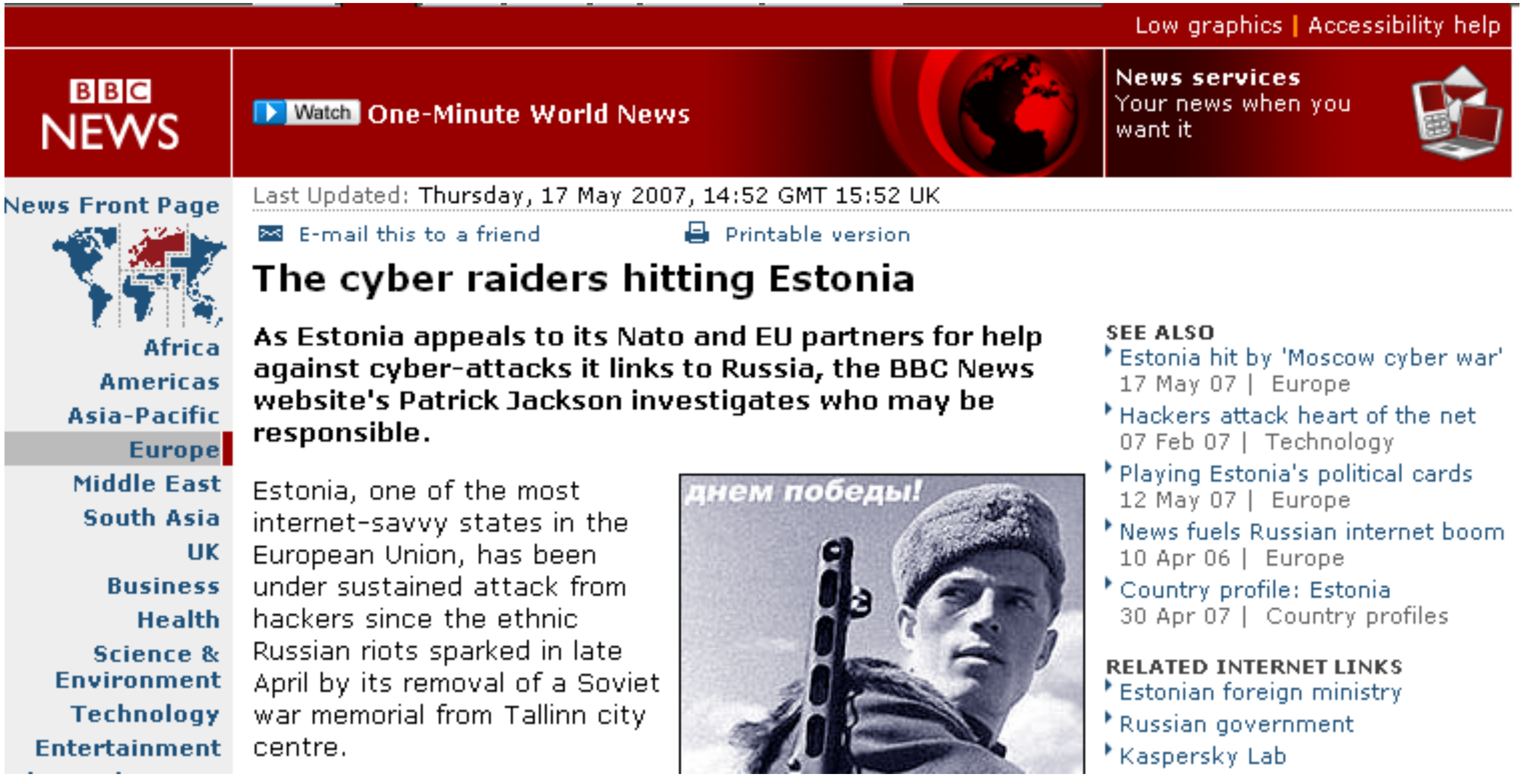
- ❑ Routing protocols react to the failure of a link or a router.
- ❑ But not to network congestions
- ❑ ARPANET had some mechanisms to react to congestions
- ❑ But they resulted into oscillations
- ❑ Congestion control was introduced in the Internet as enhancement of TCP
- ❑ But TCP has
  - no knowledge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



Low graphics | Accessibility help

BBC NEWS

Watch One-Minute World News

News services
Your news when you want it

News Front Page

Africa
Americas
Asia-Pacific
Europe
Middle East
South Asia
UK
Business
Health
Science & Environment
Technology
Entertainment

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

E-mail this to a friend   Printable version

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

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.

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";

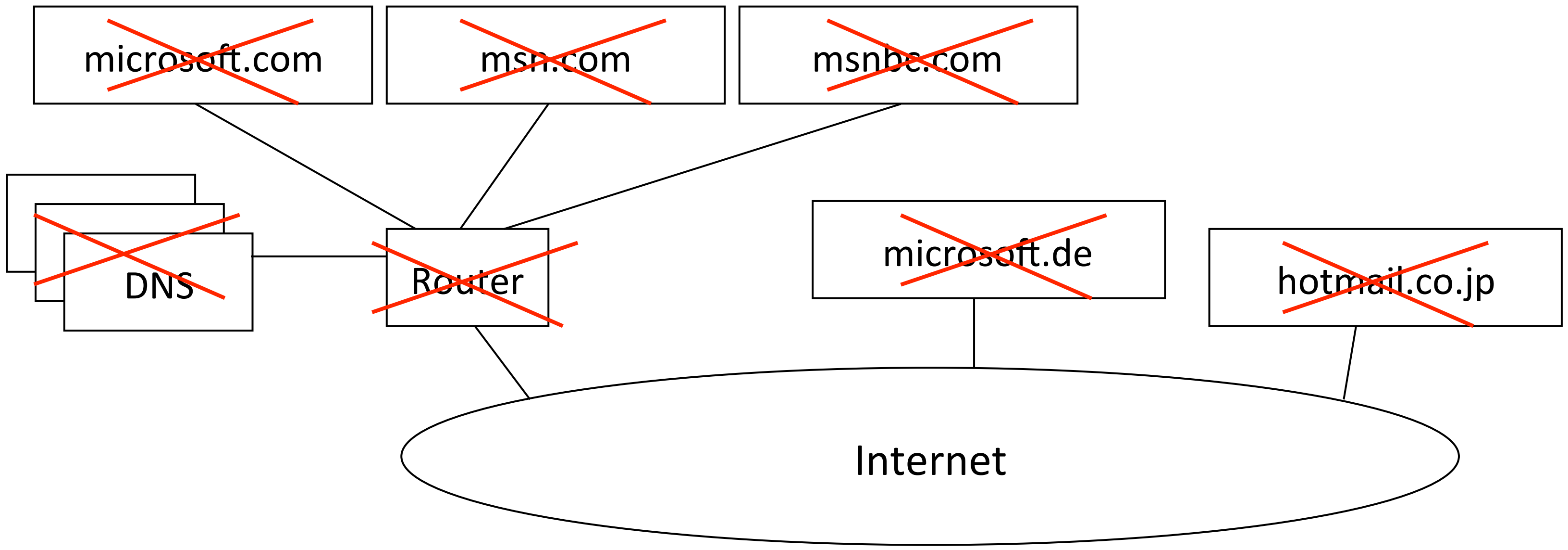
  Root cause: Database of suspected sites was mistakenly extended by ‚/ '
- Software update of the Authentication Server (Home Location Register HLR) of T-Mobile on April 21st 2009
  - Impact: phone calls and text messaging were not possible for 4 hours

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

- ❑ Terminology
- ❑ Challenges in the current Internet
- ❑ **Resilience Mechanisms**

Resilience Mechanisms

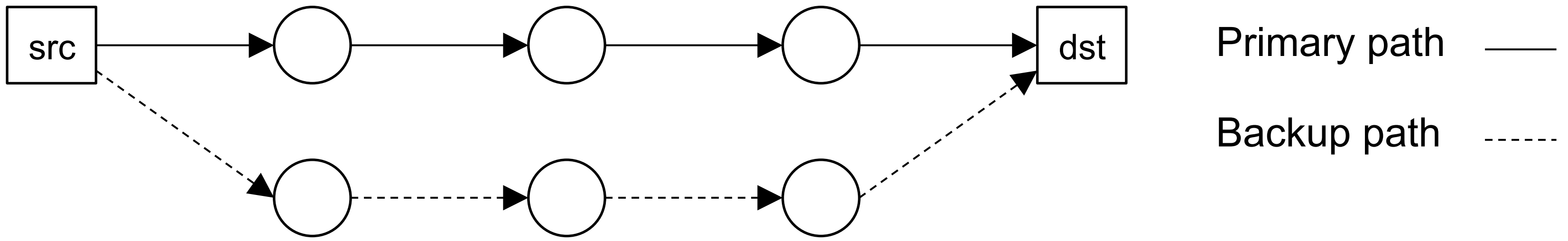
- ❑ **Topology Protection**
- ❑ Congestion Control
- ❑ Signaling Integrity
- ❑ Server Redundancy
- ❑ Virtualization
- ❑ Overlay and P2P Networks

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

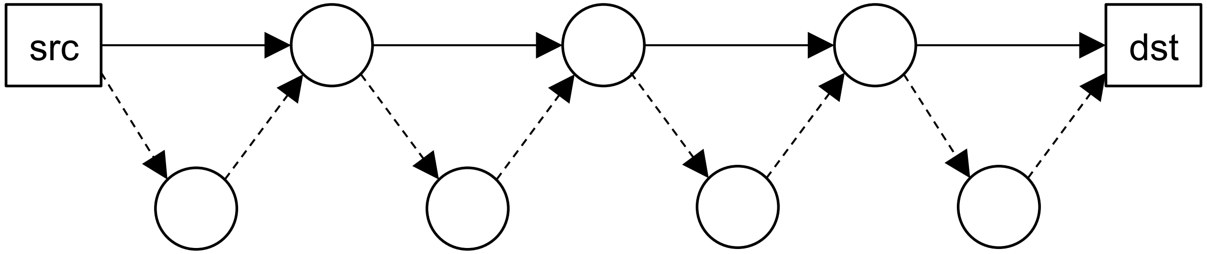


- ❑ Traffic is forwarded using backup path in case of failure
- ❑ Source needs to monitor the operation of primary path
  - ☞ Info about node or link failure needs to be propagated back to src

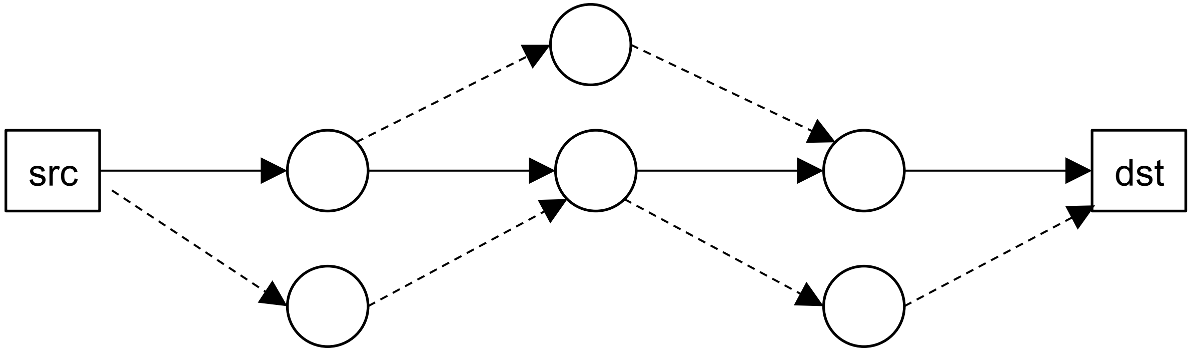
Local Protection

- ❑ 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

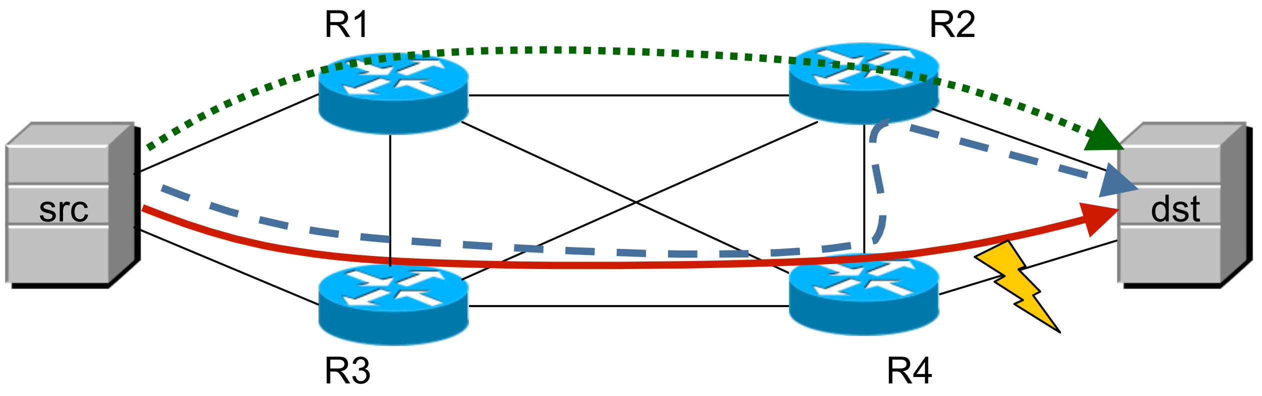
Link protection



Node protection



Example

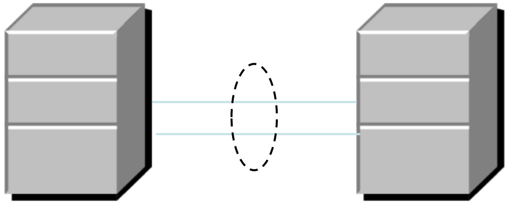


- ❑ Location protection at IP layer
- ❑ Routing protocol: OSPF
- ❑ Local protection according to IP Fast Reroute (IPFRR) (RFC 5714)

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

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

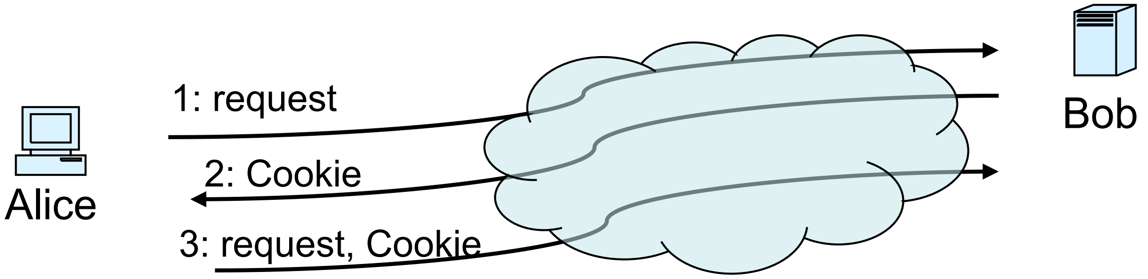
Congestion Control

- ❑ 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

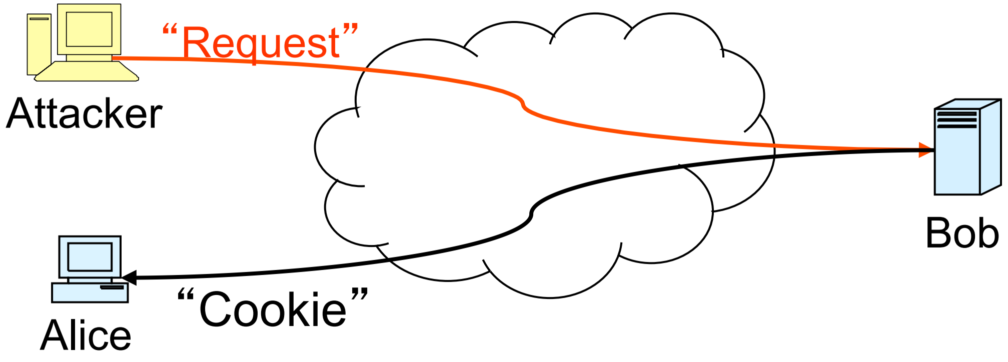
Traffic Engineering

- 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 (4)

- 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

- Topology Protection
- Congestion Control
- **Signaling Integrity**
- Server Redundancy
- Virtualization
- 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

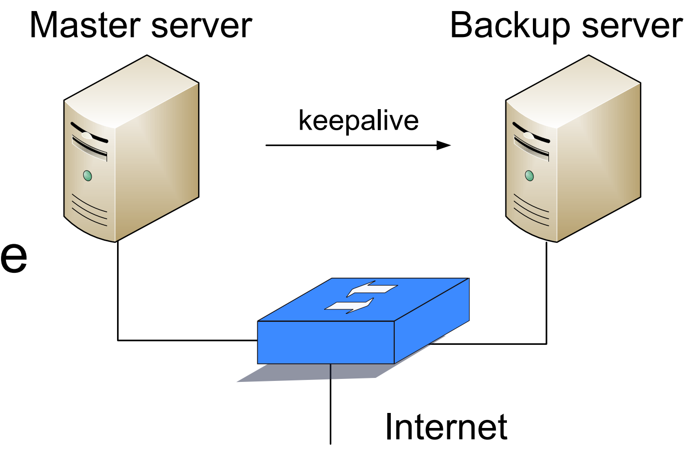
- ❑ Topology Protection
- ❑ Congestion Control
- ❑ Signaling Integrity
- ❑ **Server Redundancy**
- ❑ Virtualization
- ❑ Overlay and P2P Networks

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

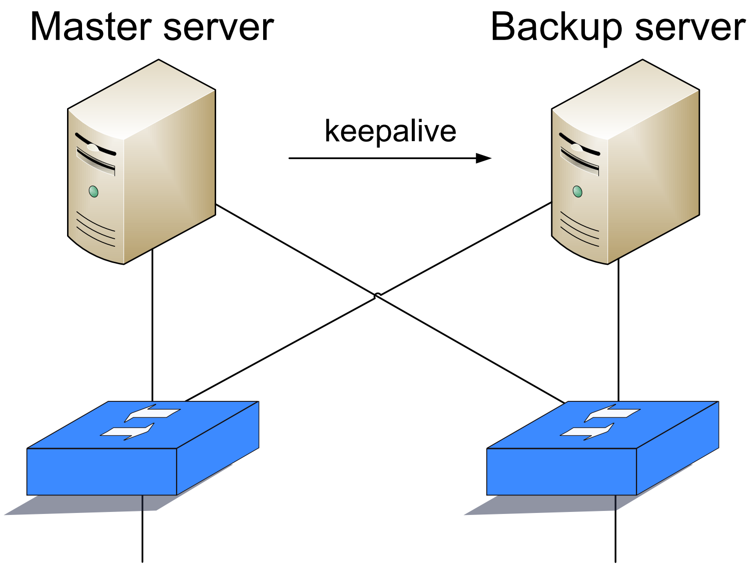
- ❑ Simple redundancy mechanism
- ❑ Backup server receives periodic “keep alive” messages from master server, e.g., every 10ms
- ❑ In case of no response
  - ▪ Backup server broadcasts an ARP message in the LAN
  - ▪ From now on, all IP traffic is forwarded to the backup server
- ❑ Drawbacks
  - ▪ Existing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



Server Redundancy; IP Takeover with 2 Switches

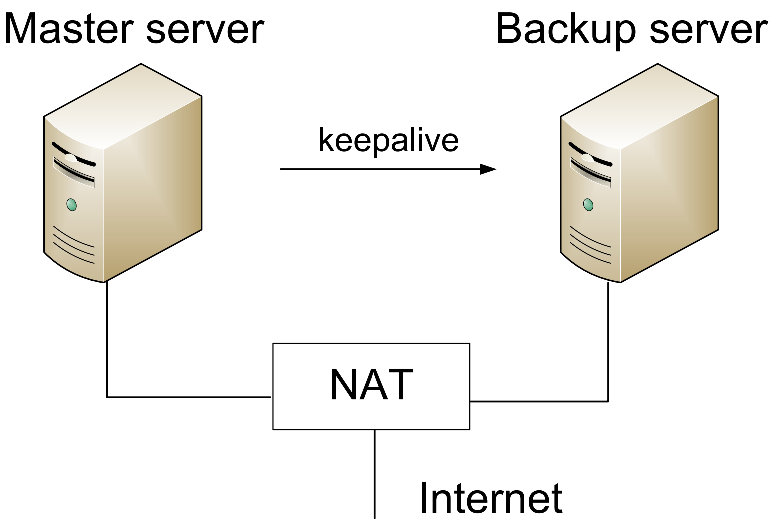
- ❑ Both master and backup servers are connected to 2 switches
- ❑ Same procedure with ARP

  ☞ 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

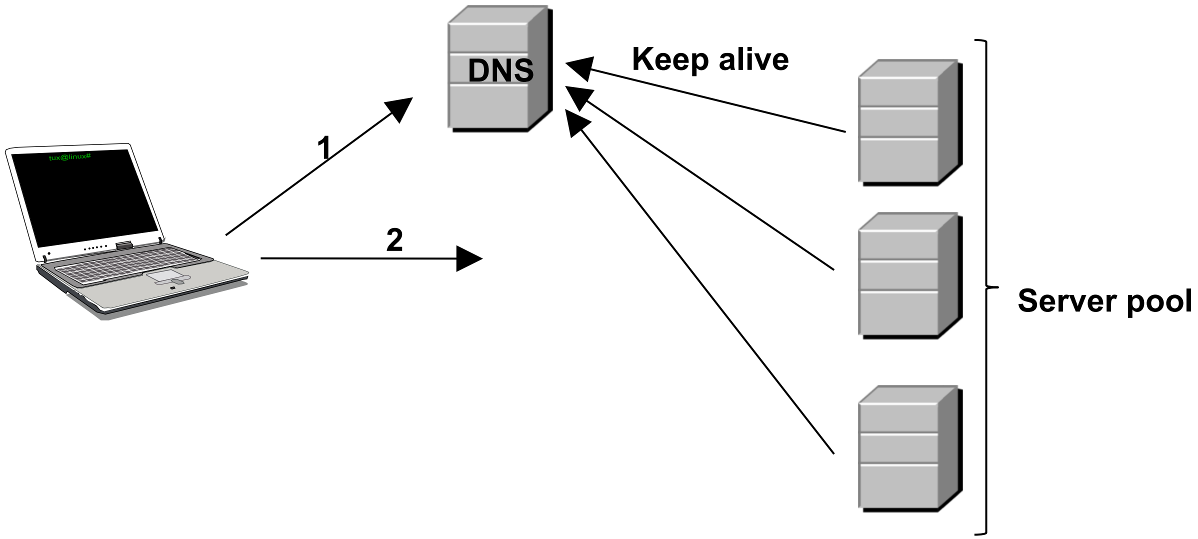
- ❑ Similar to IP Takeover
- ❑ "Keep alive" messages from backup to master server
- ❑ Change NAT binding upon lack of response from master server
  - ☞ Incoming requests are forwarded to the backup server



- ❑ 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

- ❑ Topology Protection
- ❑ Congestion Control
- ❑ Signaling Integrity
- ❑ Server Redundancy
- ❑ **Virtualization**
- ❑ 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

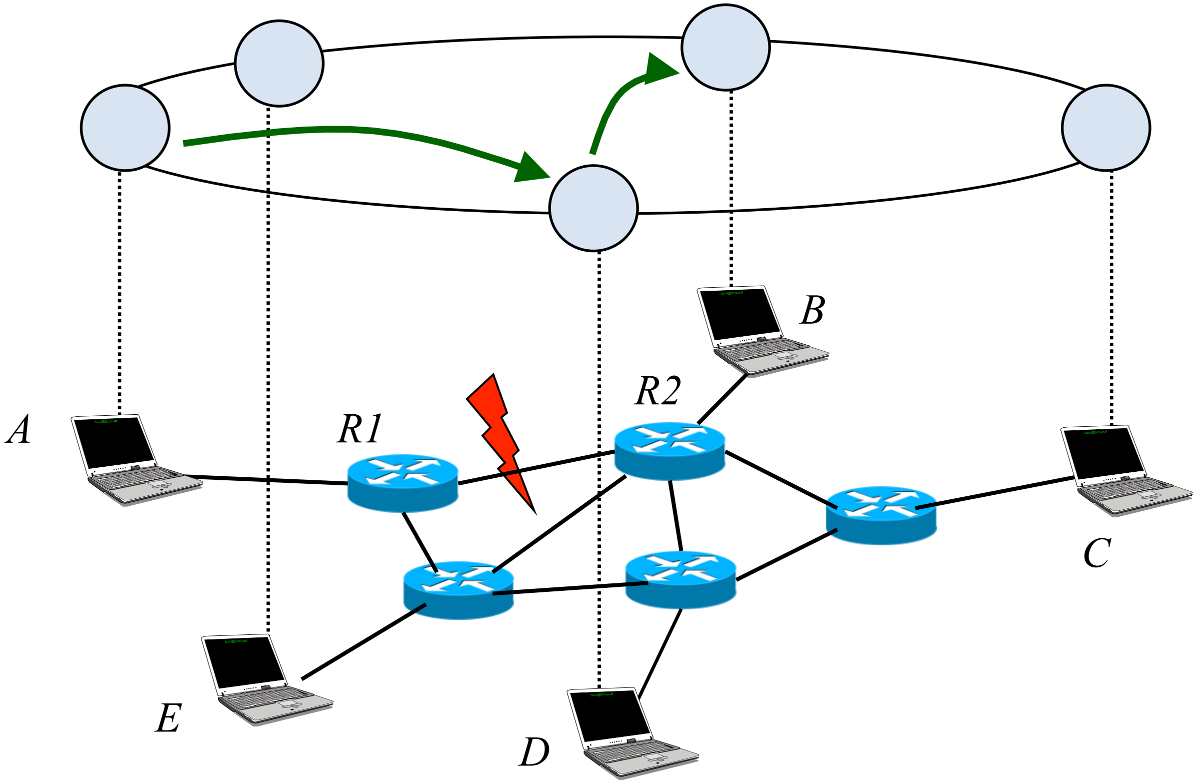
- ❑ Topology Protection
- ❑ Congestion Control
- ❑ Signaling Integrity
- ❑ Server Redundancy
- ❑ Virtualization
- ❑ **Overlay and P2P Networks**

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

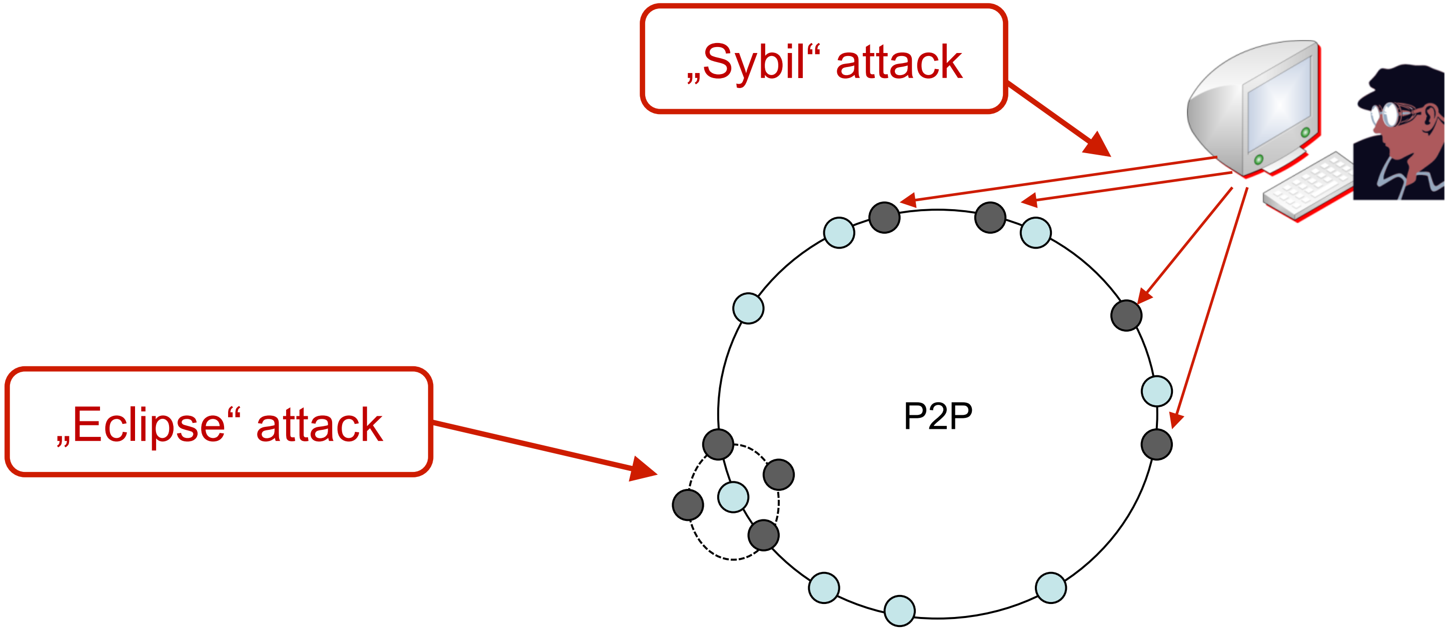
- Tor
  - while it is actually designed with anonymization in mind, it provides overlay routing and can be useful in case of network partial failures
- Skype
  - Skype supernodes typically provide connectivity for Skype clients behind firewalls or NATs

P2P Networks

- ❑ Resilience properties
  - ▪ Decentralization
  - ▪ Geographic diversity
  - ▪ Ability to cope with “churn”
    - • “Churn” means that peers join and leave at any time
    - ☞ Data replication
    - ☞ Autonomic recovery from stale 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

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

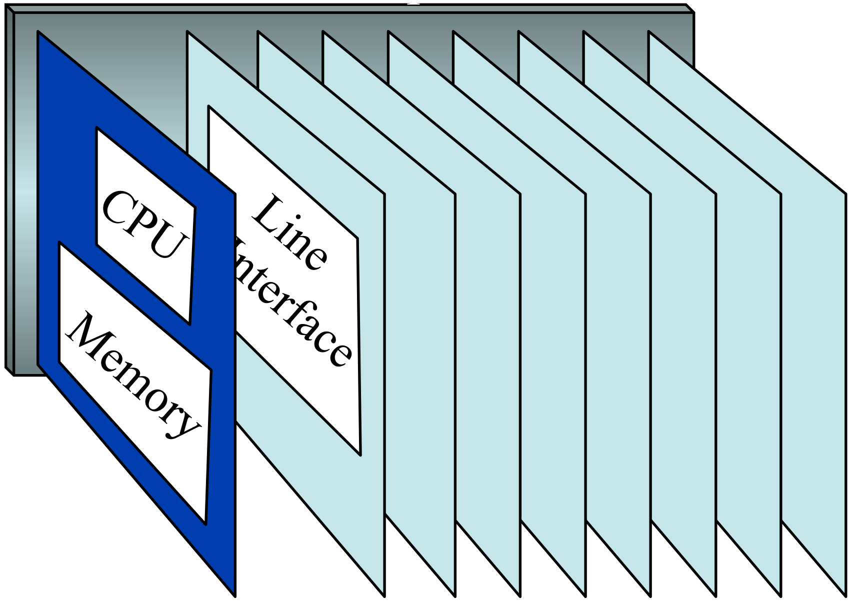
Summary

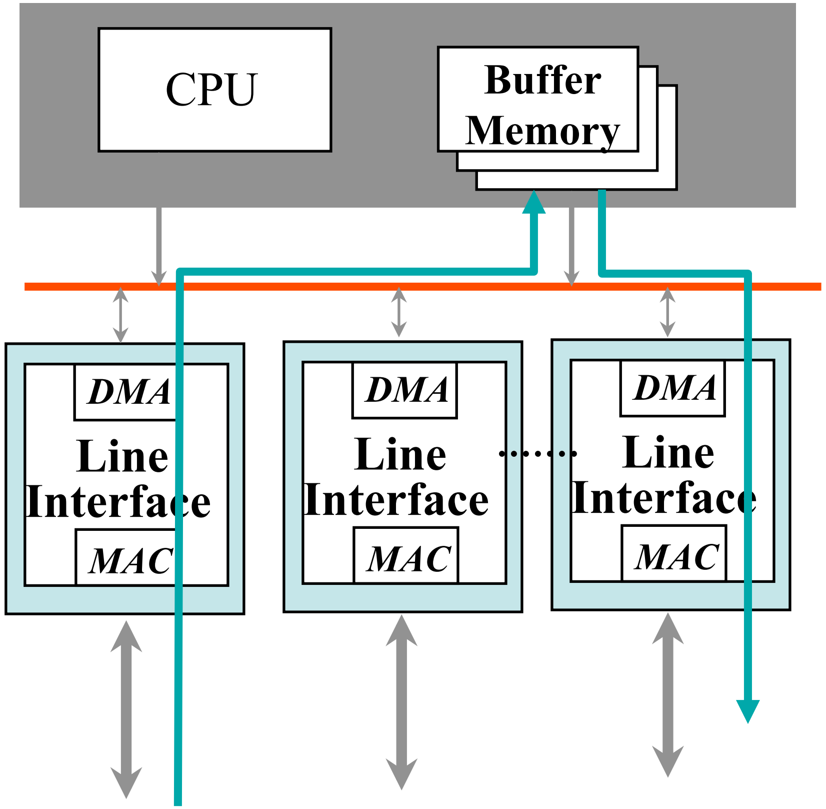
- ❑ Terminology
  - ❑ The "*fault* ➔ *error* ➔ *failure*" chain
  - ❑ Fault tolerance, Resilience, Dependability, Security
  - ❑ Availability vs. Reliability
- ❑ Challenges in the current Internet
  - ❑ Topological Failures, Overload, Lack of Integrity
  - ❑ Software Faults, Domino Effects
- ❑ Resilience Mechanisms
  - ❑ Topology Protection, Congestion Control, Signaling Integrity
  - ❑ Server Redundancy, Virtualization, Overlay and P2P Networks

Chapter:

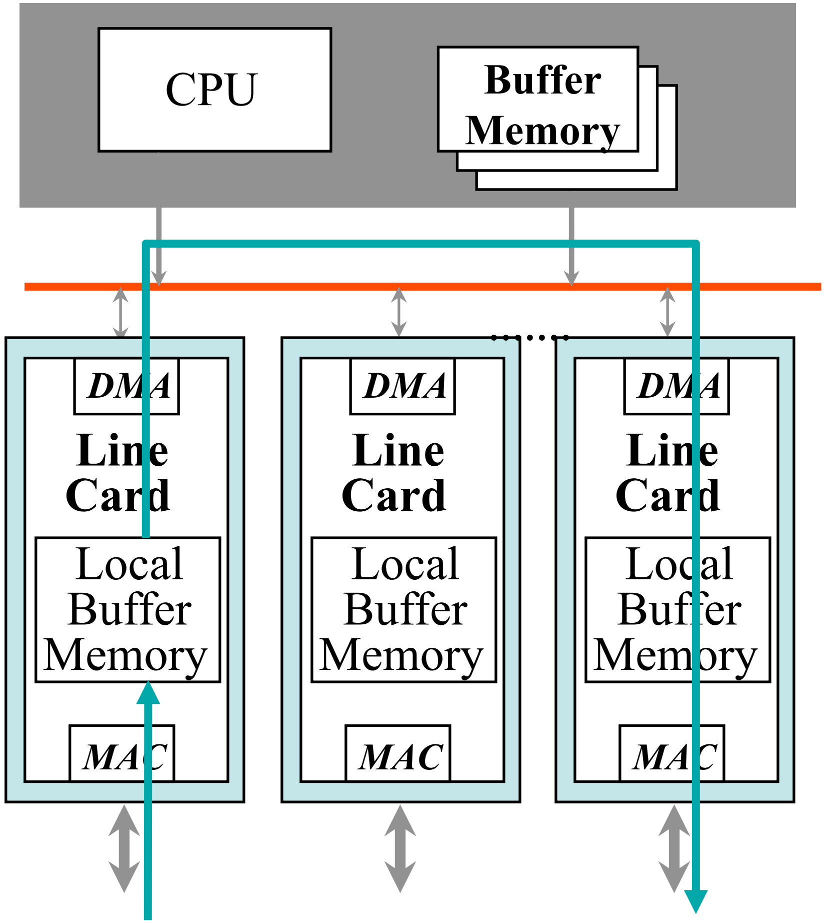
Node Architectures

First-Generation IP Routers

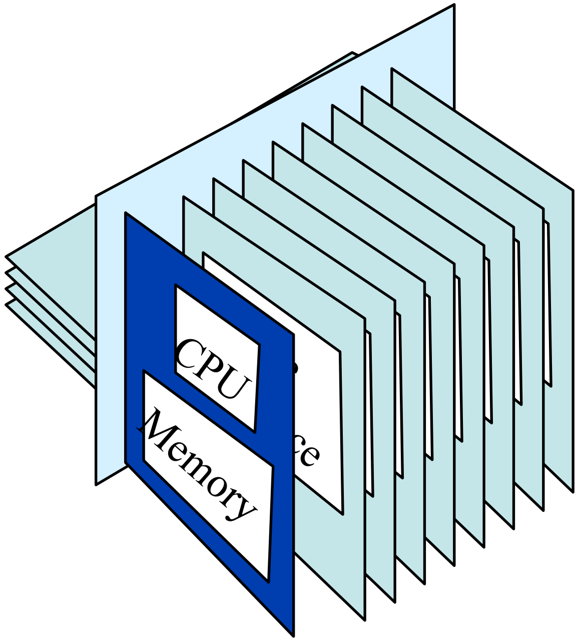


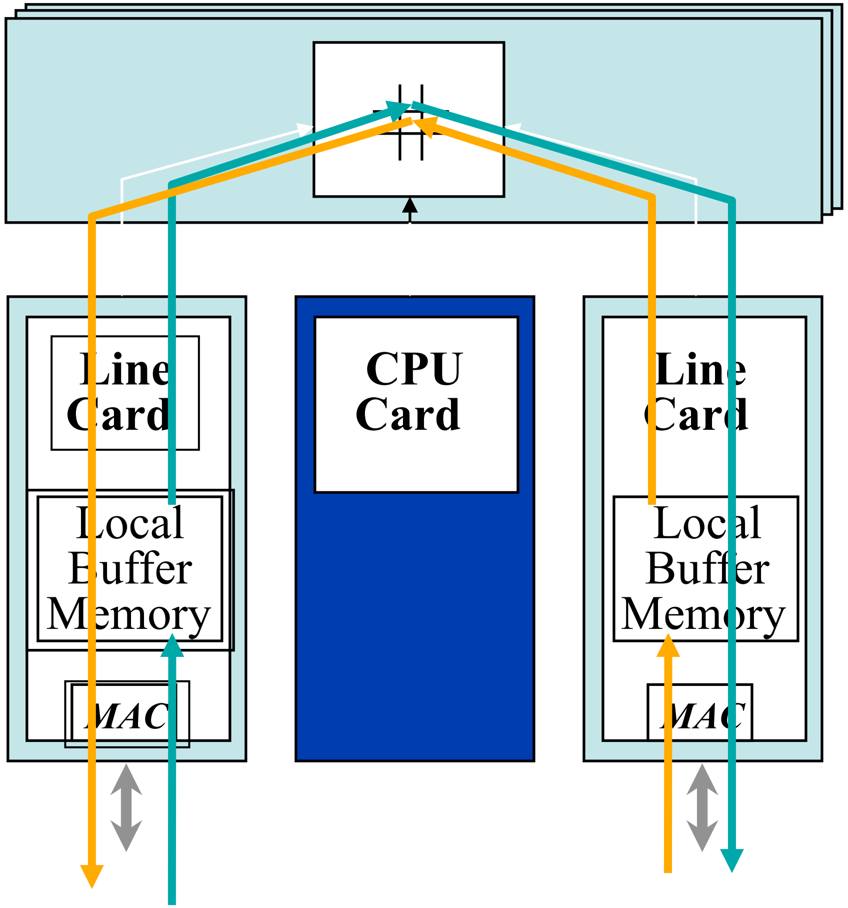


Second-Generation IP Routers



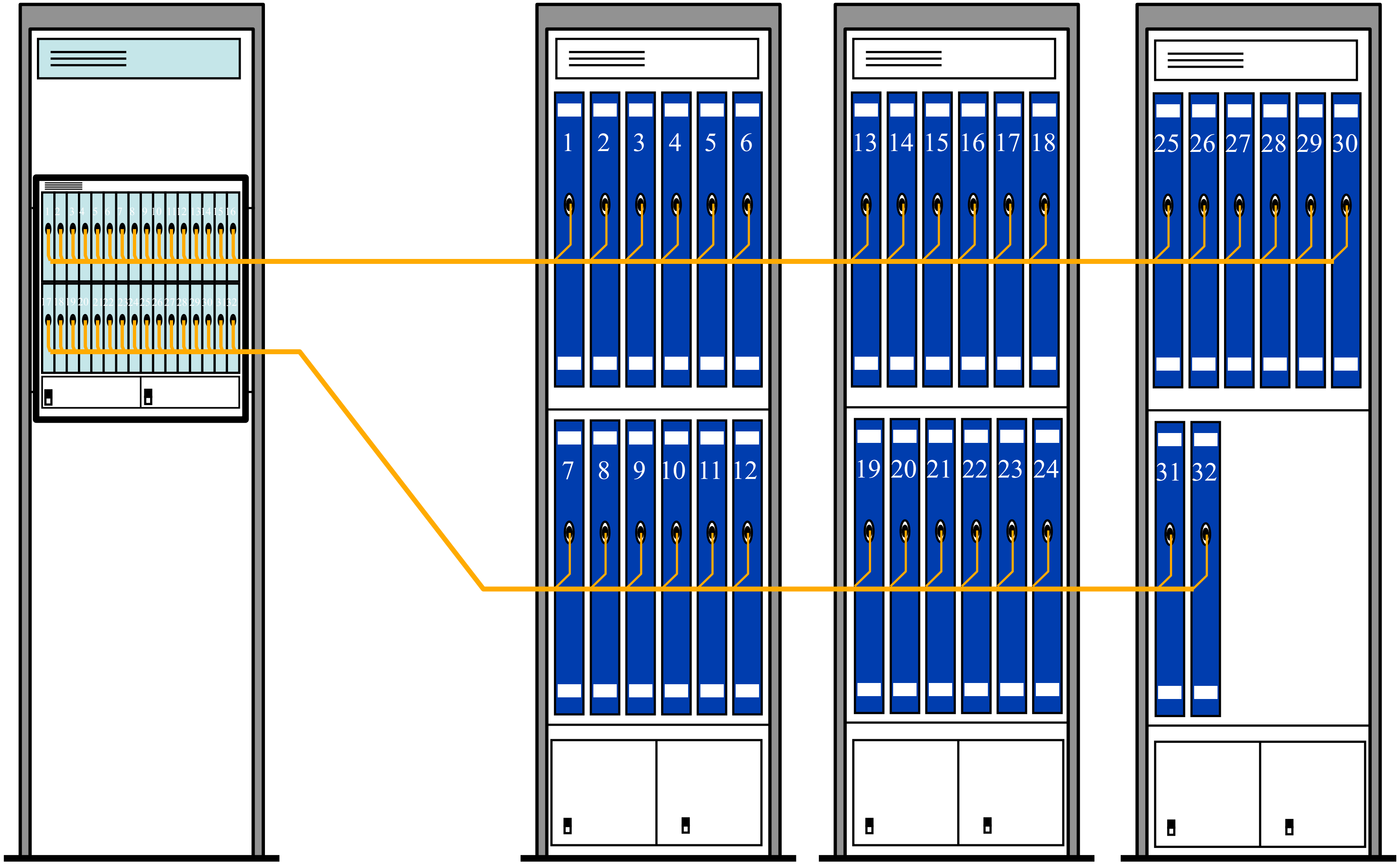
Third-Generation Switches/Routers





Fourth-Generation Switches/Routers

Clustering and Multistage



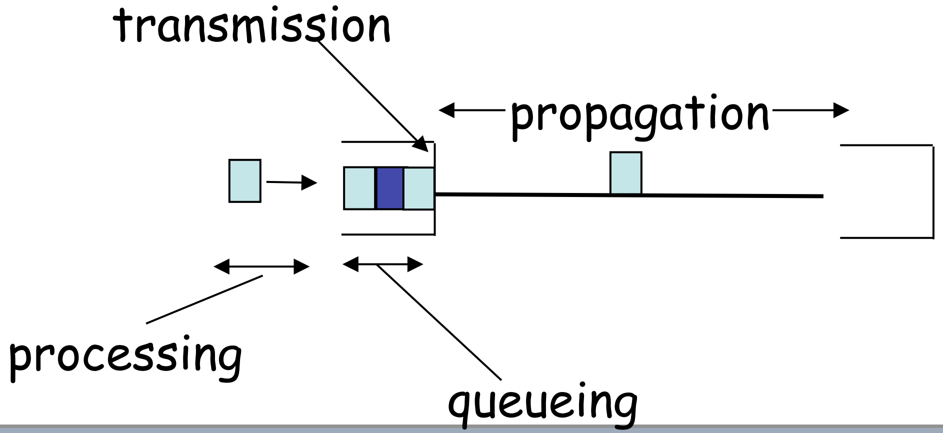
Background: Sources of packet delay

1. Processing delay:
   - Sending: prepare data for being transmitted
   - Receiving: interrupt handling

2. Queueing delay
   - Time waiting at output link for transmission
   - More congestion ➔ More queueing delay

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 (~$2 \cdot 10^8$ m/s)
   - Propagation delay = d/s



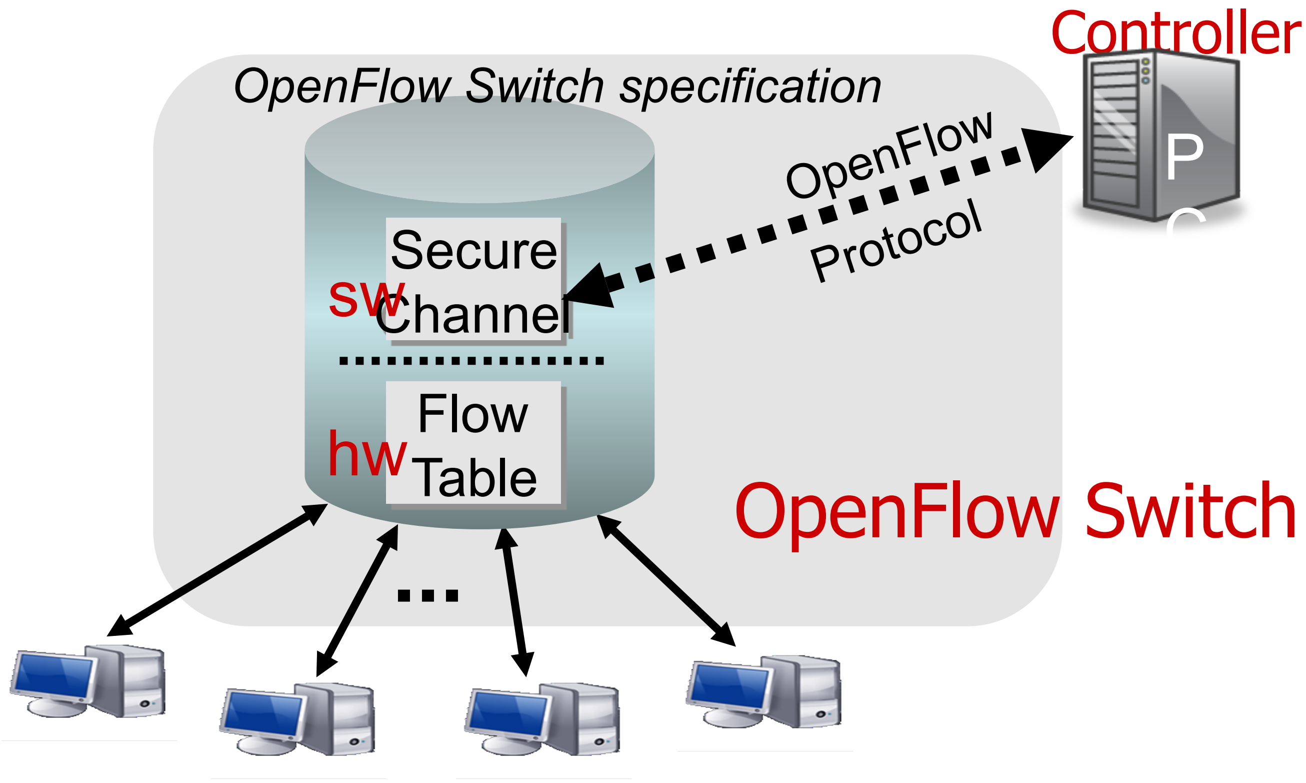
Impact Analysis: Advances in Network Technology

Data rate	Delay (1bit)	Length (1bit)	Delay (1kbyte)	Length (1kbyte)
10 Mbit/s	100 ns	20 m	0,8 ms	160 km
100 Mbit/s	10 ns	2 m	80 us	16 km
1 Gbit/s	1 ns	0,2 m	8 us	1600 m
10 Gbit/s	100 ps	0,02 m	0,8 us	160 m
40 Gbit/s	25 ps	0,005 m	0,2 us	40 m
100 Gbit/s	10 ps	0,002 m	80 ns	16 m

- ❑ Assessment
  - Transmission delay becomes less important
  - Distance (and # of RTTs!) becomes more important
    ⇨ Matters for communication beyond data center
  - Network adapter latency less important
    ⇨ Low-latency communication software becomes important

Advances in Switching

- ❑ Example: OpenFlow Switch architecture, Stanford University
- ❑ Concept: separation of switch fabric and switch control
- ❑ Allows for cheap switches, centrally controlled by switch manager
- ⇨ Assessment: suitable for low-latency data center communication



The Stanford Clean Slate Program

http://cleanslate.stanford.edu

Advances by Virtualisation and Parallelization

- Disruptive Technologies:
  Virtualisation & Multicore Architectures **in the Network**
- Drivers for **virtualisation**

  1) Complexity
     - Virtualization allows to hide complexity if it is done right (Problem: right level of abstraction)

  2) New management principles
     - network management is a driver for virtualization
- What about **multicore** and **networking**?
  - In future there may be dozens, hundreds of cores
  - Need to **expose parallel processing**
  - Affects the way how to design protocols (?)
  - Need to provide ways to access flow state

Advances in Communication Software

- Example: Wire-speed packet capture and transmission
    - Important for…
        - Network research (traffic analysis)
        - Network security (intrusion detection systems)
    - Linux APIs:
        - PF_Ring: network socket for high-speed packet capturing
        - NAPI: New API – interrupt mitigation techniques for networking devices in the Linux kernel
        - TNAPI – Multithreaded NAPI
- Issues
    - How to make advances in packet capturing available for general purpose applications?
    - How to assess advances in communication software for other OSes?

Chapter:
Network Measurements

Acknowledgements:

The content of this chapter
is partly based on slides from:
Anja Feldmann, Constantine Dovrolis

Why do we measure the network?

- Network Provider View
  - Manage traffic
    - Predict future, model reality, plan network
    - Avoid bottlenecks in advance
  - Reduce cost
  - Accounting
- Client View
  - Get the best possible service
  - Check the service („Do I get what I' ve paid for?)
- Service Provider View
  - Get information about the client
  - Adjust service to demands
  - Reduce load on service
  - Accounting
- Researcher View
  - Performance evaluation (e.g., "could our new routing algorithm handle all this real-world traffic?")
- Security view
  - Detect malicious traffic, malicious hosts, malicious networks, …

But why should we do it at all?

- Do we really have to?
  - The network is well engineered
  - Well documented protocols, mechanisms, …
  - Everything built by humans ➔ no unknowns (compare this to, e.g., physics: String theory valid? Cosmic inflation phase sound? G.U.T.? etc.)
  - In theory, we can know everything that is going on
  - ⇨ There should be no need for measurements

- But:
  - Moving target:
    - Requirements change
    - Growth, usage, structure changes
  - Highly interactive system
  - Heterogeneity in all directions
  - The total is more than the sum of its pieces

- And: The network is built, driven and used by humans
  - Detection of errors, misconfigurations, flaws, failures, misuse, …

Chapter: Network Measurements

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

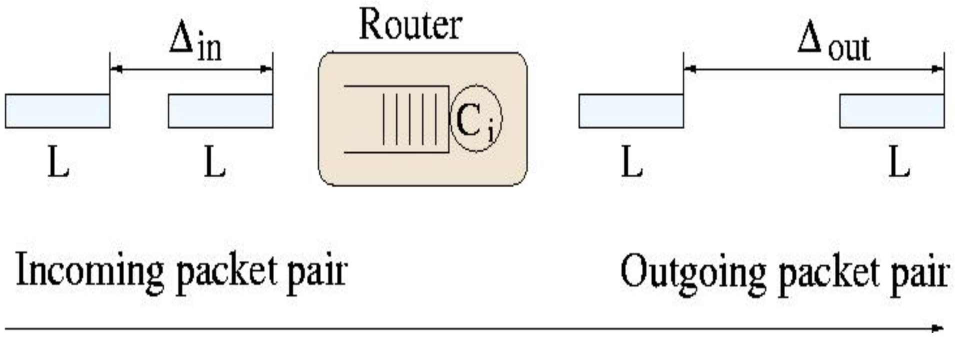
Network Measurements

- Active measurements
  - “intrusive”
  - Measurement traffic is generated and sent via the operational network.
    (Examples: ping, traceroute)
  - 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

Example: Packet pair probing

- Packet Pair (P-P) technique
  - Originally due to Jacobson & Keshav
- Send two equal-sized packets back-to-back
  - Packet size: L
  - Packet TX time at link i: L/C_i
- P-P dispersion = time interval between last bit of two packets
- Without any cross traffic, the dispersion at receiver is determined by bottleneck links (i.e., slowest link):

$$\Delta_{out} = \max\left(\Delta_{in}, \frac{L}{C_i}\right)$$



$$\Delta_R = \max_{i=1,\dots,H}\left(\frac{L}{C_i}\right) = \frac{L}{C}$$

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 suitable 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

Measurement types (summary)

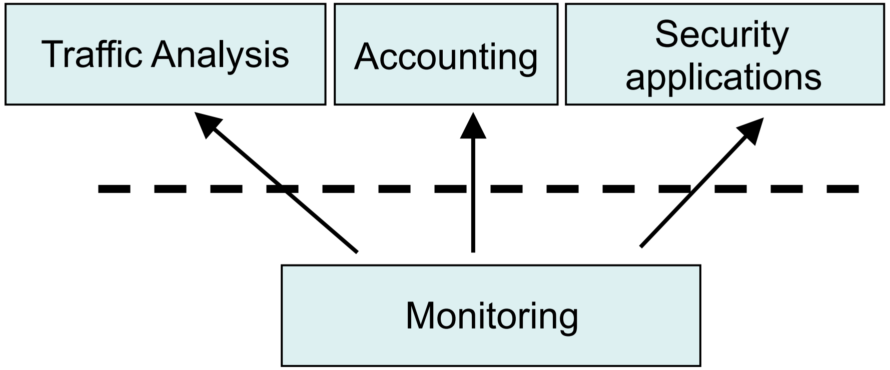
- Active Measurements
  - Intrusive
  - Find out what the network is capable of
  - Changes the network state

- Passive Measurements (or network monitoring)
  - Non-intrusive
  - Find out what the current situation is
  - Does not influence the network state (more or less)

- Hybrid
  - Alter actual traffic
  - Reduce the impact of active measurements
  - Might introduce new bias for applications

Network Monitoring

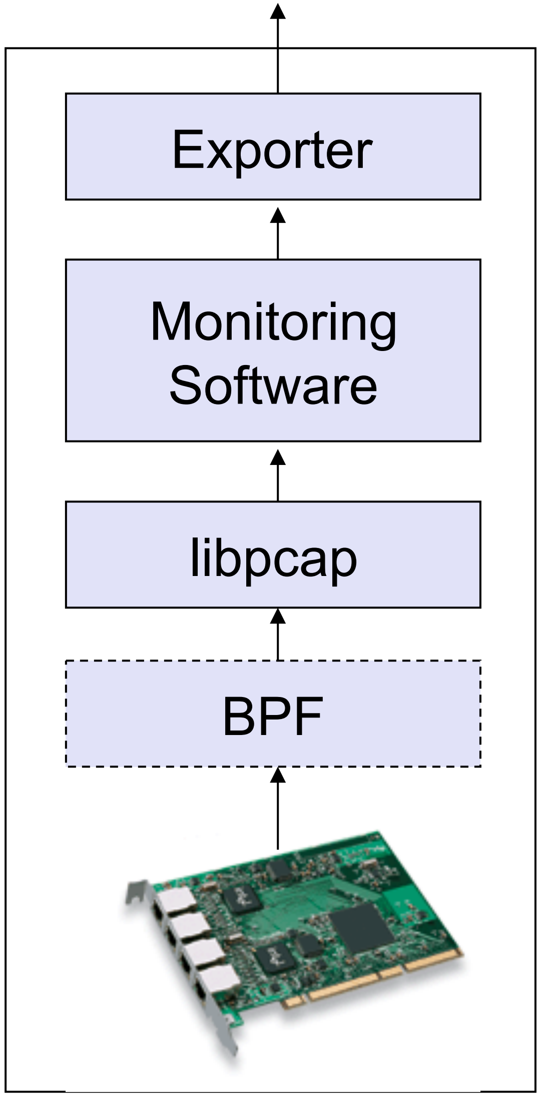
- Applications of network monitoring
  - Traffic analysis
    - Traffic engineering
    - Anomaly detection
  - Accounting
    - Resource utilization
    - Accounting and charging
  - Security
    - Intrusion detection
    - Detection of prohibited data transfers (e.g., P2P applications)
  - Research



- Open issues
  - Protection of measurement data against illegitimate use (encryption, …)
  - Applicable law ("lawful interception", privacy laws, …)

Monitoring Probe

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



Exporter

Monitoring Software

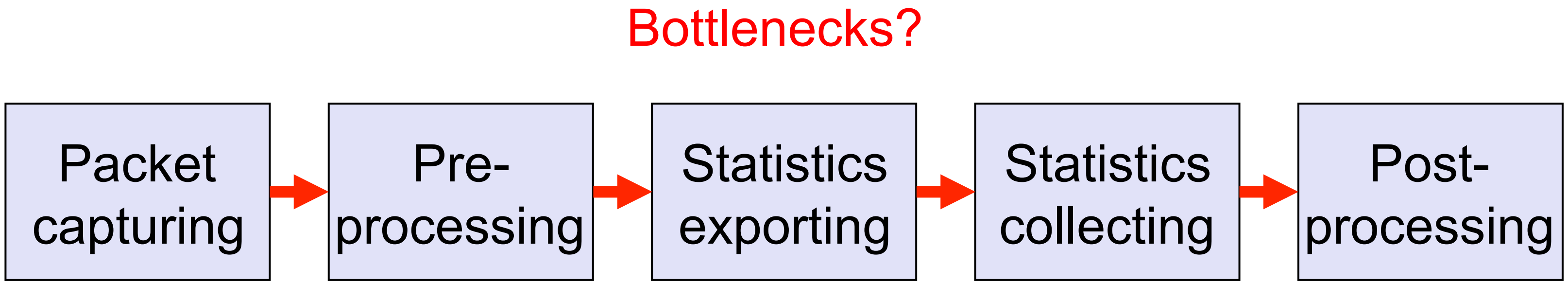
libpcap

BPF

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

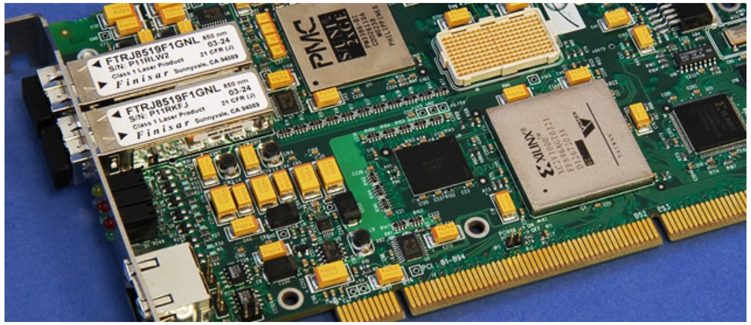
- Approaches
  - High-end (intelligent) network adapters
    - Large amounts of memory
    - Can do filtering, timestamping etc. on their own
  - Sophisticated algorithms/techniques in OS stack for
    - Maintaining packet queues
    - Elimination of packet copy operations
    - Maintaining state (e.g., managing hash tables describing packet flows; sophisticated packet classification algorithms)
  - Sampling
  - Filtering
  - Aggregation

⇨ more on subsequent slides

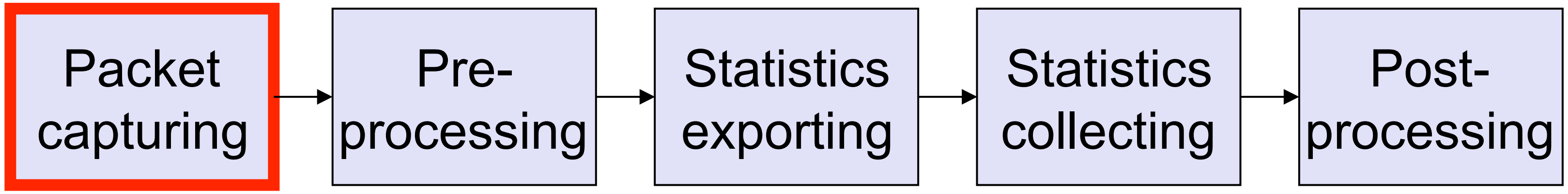
Special Network Adapters

- 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

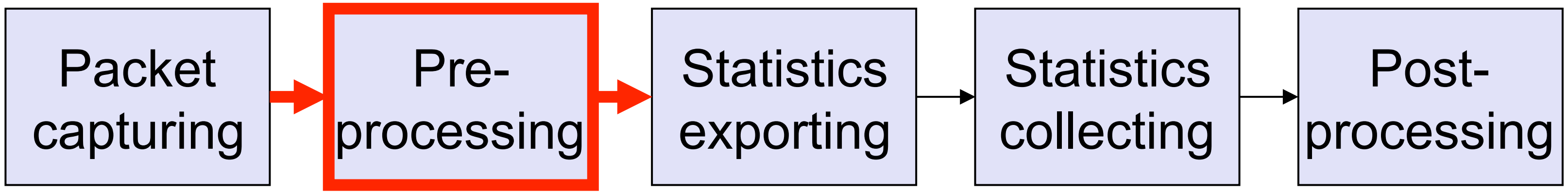


- Monitoring interface cards
  - Dedicated monitoring hardware (usually only RX, no TX)
  - Programmable, i.e. certain processing (filtering, high-precision timestamps, ...) can be performed on the network interface card



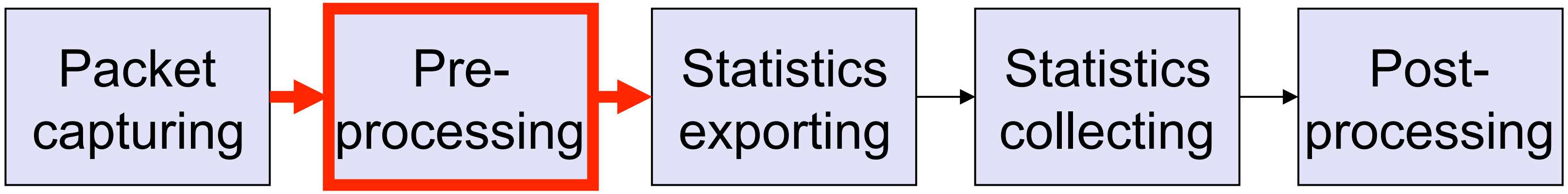
Memory Management I

- 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



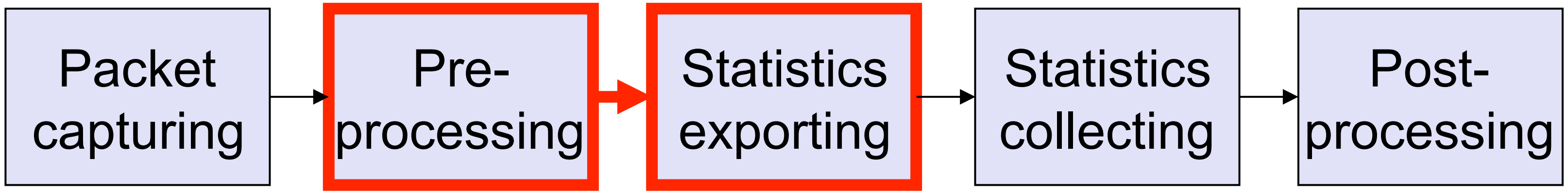
Memory Management II

- 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
- Multi-dimensional packet classification algorithms (e.g., HiPac)
  - Allow to test for 1,000s of complex filtering rules within one lookup operation (e.g., "all TCP packets from network 131.159.14.0/24, but not 131.159.14.0/27, and with source port 80, 443 or 6666–6670, but not with destination address 192.168.69.96–192.168.69.99 ➔ Apply rule 34")
  - Mostly tree-based ➔ Lookups fast, but tree alterations costly.



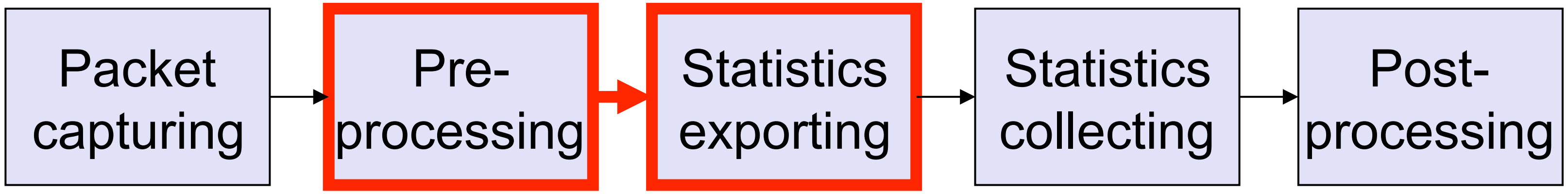
Packet Sampling

- 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
  - “Time machine” sampling: Sample first N bytes of every flow



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
      - 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
      - ➔ Technique also known as *trajectory sampling*
      - 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-tuple (protocol, srcIP, dstIP, srcPort, dstPort)
  - For each arriving packet, the statistic counters of the appropriate flow are modified
  - Whenever a flow is terminated (TCP FIN, TCP RST, timeout), its record is exported
  - Sampling algorithms can reduce the # of flows 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 (e.g. increasing # of flows / time)

IPFIX – Work Principles

- Identification of individual traffic flows
  - 5-tuple: 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 – IP Flow Information Export Protocol

- Quite a number of 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)

- Transport protocol: Transport of exported IPFIX 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 – 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

Network Traffic

Traffic by Port (I)

18 hours of traffic to AT&T dial clients on July 22, 1997

Name	Port	% Bytes	% Packets	Bytes/Packet
www	80	56,75	44,79	819
nntp	119	24,65	12,90	1235
pop3 email	110	1,88	3,17	384
cuseeme	7648	0,95	1,85	333
secure www	443	0,74	0,79	603
irc	6667	0,27	0,74	239
ftp	20	0,65	0,64	659
dns	53	0,19	0,58	210
…				

Traffic by Port (II)

24 hours of traffic to/from MWN clients in 2006

Name	Port	% Conns	% Succes	%Payload
www	80	70,82	68,13	72,59
cifs	445	3,53	0,01	0,00
secure www	443	2,34	2,08	1,29
ssh	22	2,12	1,75	1,71
smtp	25	1,85	1,05	1,71
	1042	1,66	0,00	0,00
	1433	1,06	0,00	0,00
	135	1,04	0,00	0,00
	< 1024	83,68	73,73	79,05
	> 1024	16,32	4,08	20,95

Traffic by Port (III)

- Port 80 dominates traffic mix
  - Still growing
    - More web applications
    - Tunnel everything over port 80
- Characterization of traffic by port is possible
  - Well-known ports
    (1–1024; take a look at `/etc/services`)
- Growing margin of error
  - Automatic configuration
  - * over http: VPN, P2P, Skype, AJAX-SSH, …
  - Aggressive applications (e.g. Skype):
    „just find me an open port“

Traffic Flows

18 hours of traffic to AT&T dial clients on July 22, 1997

Name	Port	% Bytes	% Pkts	Bytes/ Pkt	% Flows	Pkts/ Flow	Duration (s)
www	80	56,75	44,79	819	74,58	12	11,2
nntp	119	24,65	12,90	1235	1,20	210	132,6
pop3 email	110	1,88	3,17	384	2,80	22	10,3
cuseeme	7648	0,95	1,85	333	0,03	1375	192,0
secure www	443	0,74	0,79	603	0,99	16	14,2
irc	6667	0,27	0,74	239	0,16	89	384,6
ftp	20	0,65	0,64	659	0,26	47	30,1
dns	53	0,19	0,58	210	10,69	1	0,5
…							

Elephants and Mice

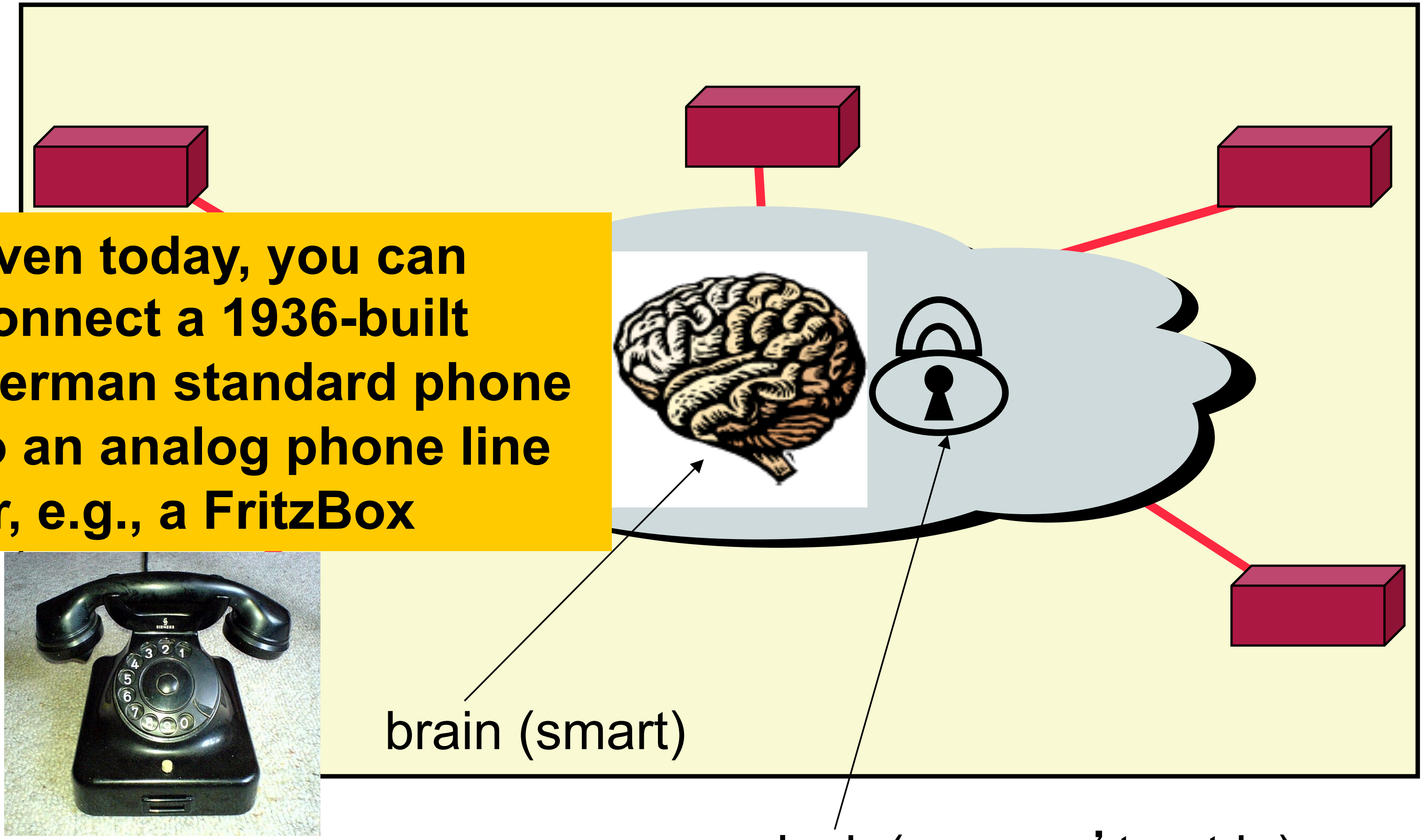
- Many very short flows (30% < 300 bytes)
- Many medium-sized flows (short web transfers)
- Few long flows
- But:
  Most bytes belong to these long flows (large images, files, flash, video)
- Same picture for other metrics
  - Bytes/flow
  - Packets/flow
  - Lifetime
- Flow densities are traffic patterns and signatures

Chapter:
Internet Architecture

Structure

- Internet architecture
  - Requirements, assumptions
  - Design decisions
- Shortcomings and „Future Internet“ concepts
  - „Legacy Future Internet“: IPv6, SCTP, …
  - Security
  - QoS, multicast
  - Economic implications, „tussle space“
  - Mobility and Locator–ID split
  - In-network congestion control
  - Modules instead of layers
  - Delay-tolerant/disruption-tolerant networking
  - Content-based networking/Publish–subscribe architectures
  - Evolutionary vs. Revolutionary/Clean-slate

Common View of the Phone Network



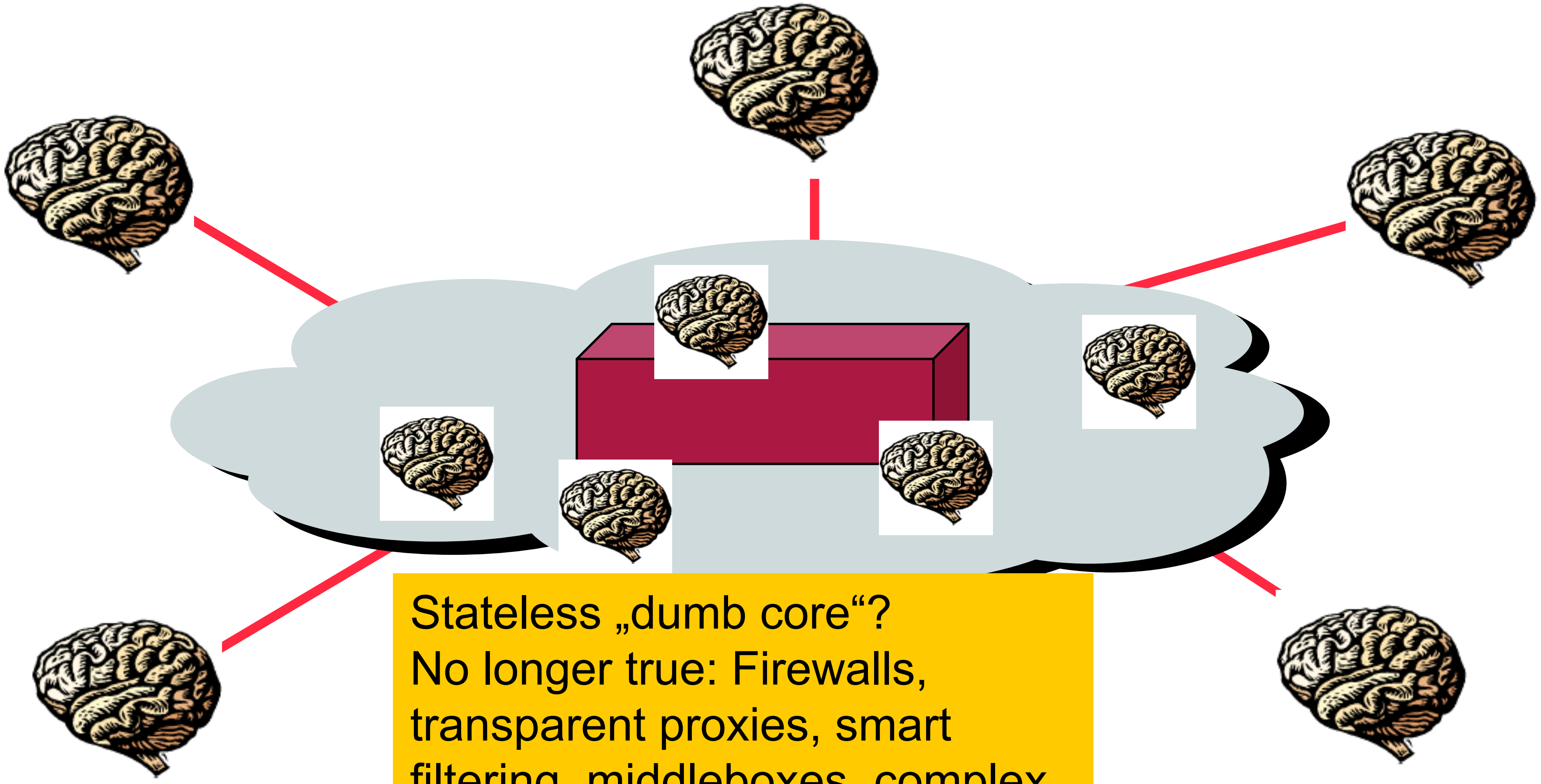
Even today, you can connect a 1936-built German standard phone to an analog phone line or, e.g., a FritzBox

brain (smart)

lock (you can’t get in)

brick (dumb)

Common View of the IP Network



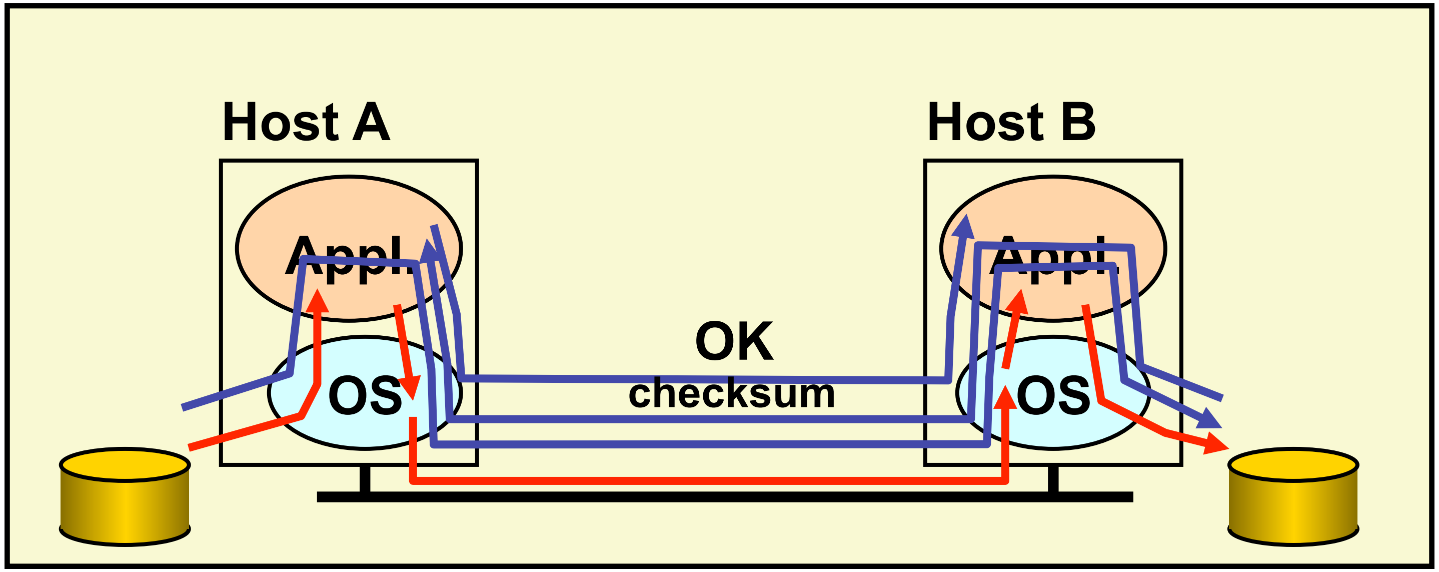
Stateless „dumb core“?
No longer true: Firewalls, transparent proxies, smart filtering, middleboxes, complex routing protocols like BGP, …

The Internet End-to-End principle

Internet End-to-End Principle

- "…functions placed at the lower levels may be *redundant* or of *little value* when compared to the cost of providing them at the higher level…"
- "…sometimes an *incomplete* version of the function provided by the communication system (lower levels) may be useful as a *performance enhancement*…"
- This leads to a philosophy diametrically opposite to the telephone world of dumb end-systems (the telephone) and intelligent networks.

Example: Reliable File Transfer



- ❑ Solution 1: make each step reliable, and then concatenate them
- ❑ Solution 2: each step unreliable – end-to-end check and retry

Discussion

- Is solution 1 good enough?
  - No – what happens if components fail or misbehave (bugs)?
- Is reliable communication sufficient?
  - No – what happens in case of, e.g., 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 at 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

Internet Design Philosophy (Clark' 88)

In order of importance:

Different ordering of priorities would make a different architecture!

0 Connect existing networks
   - initially ARPANET, ARPA packet radio, packet satellite network
1. Survivability
   - ensure communication service even with network and router failures
2. Support multiple types of services
3. Must accommodate a variety of networks
4. Allow distributed management
5. Allow host attachment with a low level of effort
6. Be cost effective
7. Allow resource accountability

1. Survivability

- ❑ Continue to operate even in the presence of network failures (e.g., link and router failures)
  - ▪ As long as network is not partitioned, two endpoints should be able to communicate
  - ▪ Any other failure (excepting network partition) should be transparent to endpoints
- ❑ Decision: maintain end-to-end transport state only at end-points
  - ▪ eliminate the problem of handling state inconsistency and performing state restoration when router fails
- ❑ Internet: stateless network-layer architecture
  - ▪ No notion of a session/call at network layer
- ❑ Remark: “Internet was built to survive global thermonuclear war” = urban legend; untrue

2. Types of Services

- Add UDP to TCP to better support other apps
  - e.g., “real-time” applications
- Arguably main reason for separating TCP, IP
- Datagram abstraction: lower common denominator on which other services can be built
  - Service differentiation was considered (ToS bits in IP header), but this has never happened on the large scale (Why?)

3. Variety of Networks

- Very successful (why?)
  - Because of minimalism
  - Only requirement from underlying network: to deliver a packet with a "reasonable" probability of success
- …but does *not* require:
  - Reliability
  - In-order delivery
  - Bandwidth, delay, other QoS guarantees
- The mantra: IP over everything
  - Then: ARPANET, X.25, DARPA satellite network, phone lines, …
  - Today: Ethernet, DSL, 802.11, GSM/UMTS, …
  - Soon: LTE, WIMAX, …

Other Goals

- Allow distributed management
  - Administrative autonomy: IP interconnects networks
    - each network can be managed by a different organization
    - different organizations need to interact only at the boundaries
    - … but this model complicates routing
- Cost effective
  - sources of inefficiency
    - header overhead
    - retransmissions
    - routing
  - …but "optimal" performance never been top priority

Other Goals (Cont)

- Low cost of attaching a new host
  - not a strong point → higher than other architecture because the intelligence is in hosts (e.g., telephone vs. computer)
  - bad implementations or malicious users can produce considerably harm (remember fate-sharing?)

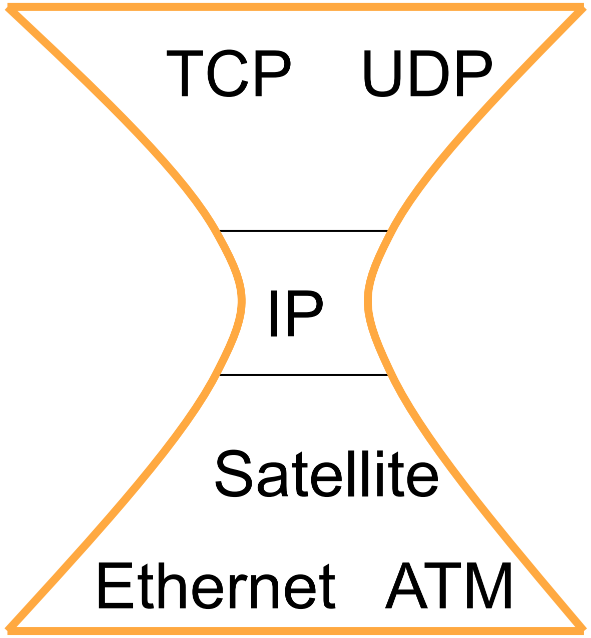
- Accountability
  - Not a strong point: no financial interests (research network!)

What About the Future

- Datagram not the best abstraction for:
  - resource management,accountability, QoS
- new abstraction: flow (see IPv6)
  - flow not precisely defined (when does it end?)
  - IPv6: difficulties to make use of flowids
- 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
  - IP hides transport/application details from network
  - IP hides network details from transport/application
- Stateless architecture
  - No per-flow state inside network
  - Intelligence (i.e., state keeping) in end hosts, but not in core



IP hourglass

Summary: Minimalist Approach

- Dumb network
  - IP provides minimal functionalities to support connectivity
  - Addressing, forwarding, routing
- Smart end system
  - 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, SMTP, FTP, X11, Web, ssh, SSL/TLS, POP, IMAP, Peer-to-Peer, …)
  - Decentralized network administration

The KISS principle

- KISS = “Keep it simple, stupid!”
- Success of…
  - IP
  - Ethernet
  - RISC processors
  - SIP vs. H.323
- “Building complex functions into network optimizes network for small number of services, while substantially increasing cost for uses unknown at design time”

Internet architecture: Some explicit or implicit assumptions

- A research network
  - No economic/business/judicial aspects, no competition
  - Cooperative, perhaps even altruistic participants
- Knowledgeable and responsible end users; administrators even more so
- Almost no malicious participants
  - Perhaps some malicious users? (➔ password protection),
  - …but no malicious systems administrators,
  - …and certainly no malicious network operators
- A couple of thousand nodes, perhaps a million users
- No mobility: End hosts will not shift their position within network
- Most links are wired; packet loss indicates network congestion
- Just a temporary solution

- **…and yet it still works!? Amazing!**

But that was yesterday

… what about tomorrow? Or even: today?

Rethinking Internet Design

What’ s changed?

- ❑ Operation in untrustworthy world
  - ▪ Endpoints can be malicious
  - ▪ If endpoint not trustworthy, but want trustworthy network ⇨ more mechanism in network core

- ❑ More demanding applications
  - ▪ End-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 of users)
  - ▪ e.g., Chinese government, US recording industry

- ❑ less sophisticated users

All five changes motivate shift away from end-to-end!

What's at stake?

"At issue is the conventional understanding of the "Internet philosophy"

- freedom of action
- user empowerment
- end-user responsibility for actions taken
- lack of control "in" the net that limit or regulate what users can do

The end-to-end argument fostered that philosophy because they enable the freedom to innovate, install new software at will, and run applications of the users' choice"

[Blumenthal and Clark, 2001]

Technical response to changes

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

- 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

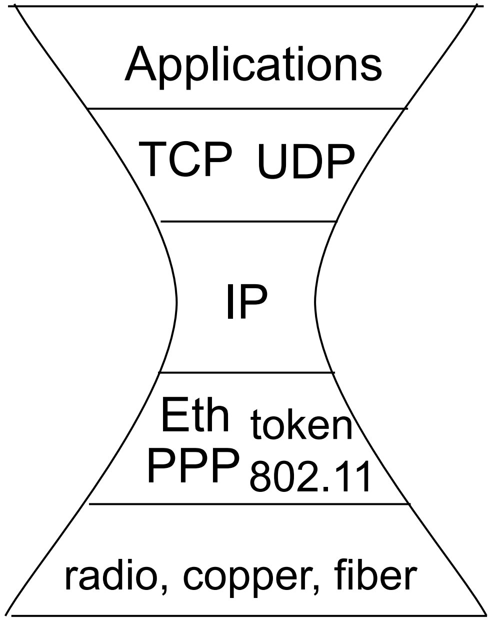
Technical response to changes

- Add functions to the network core:
  - Filtering firewalls
  - Application-level firewalls
  - NAT boxes
  - Transparent Web proxies

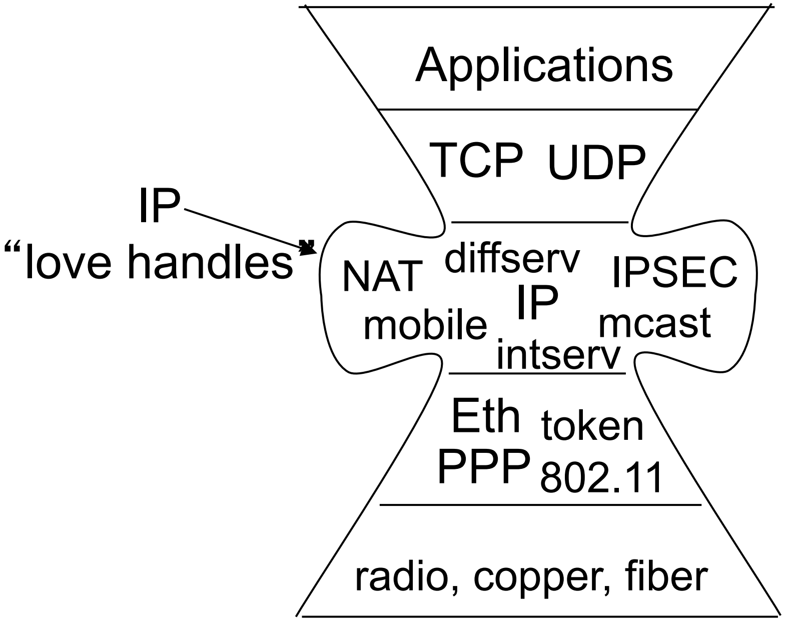
All operate *within* network, making use of application-level information

- Which addresses can do what at application level?
- If addresses have meaning to applications, NAT must "understand" that meaning. Difficult!

Big picture: supporting new applications – losing the IP hour glass figure?

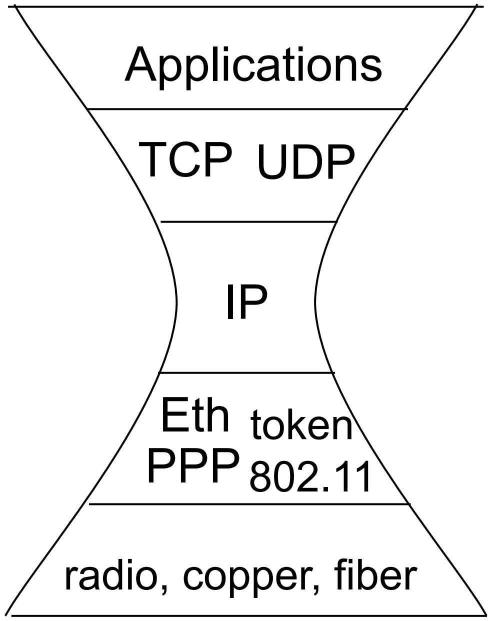


IP “hourglass”

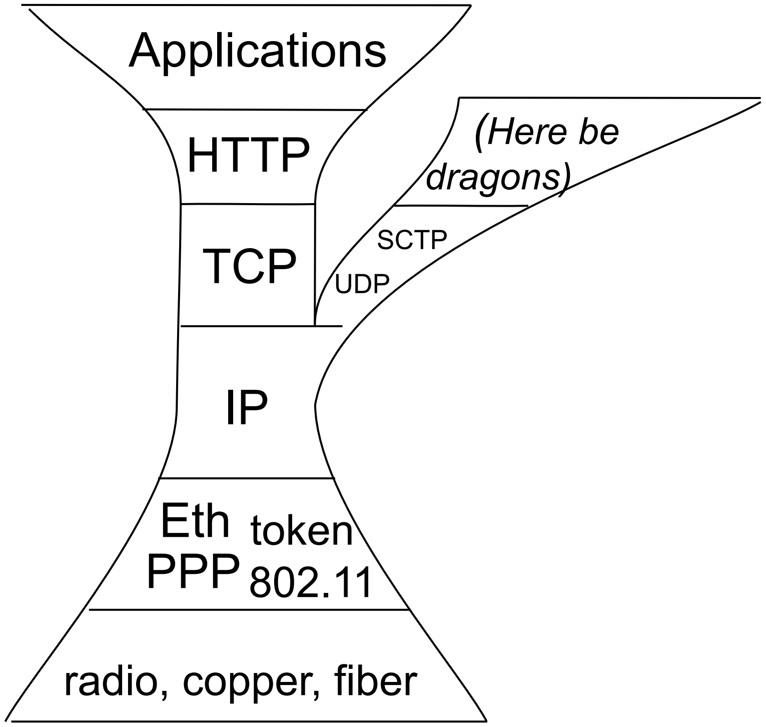


Middle-age IP “hourglass”?

Big picture: supporting new applications – losing the IP hour glass figure?

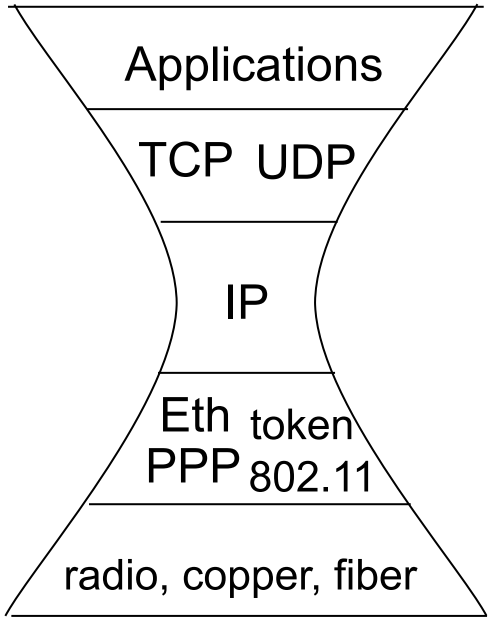


IP “hourglass”

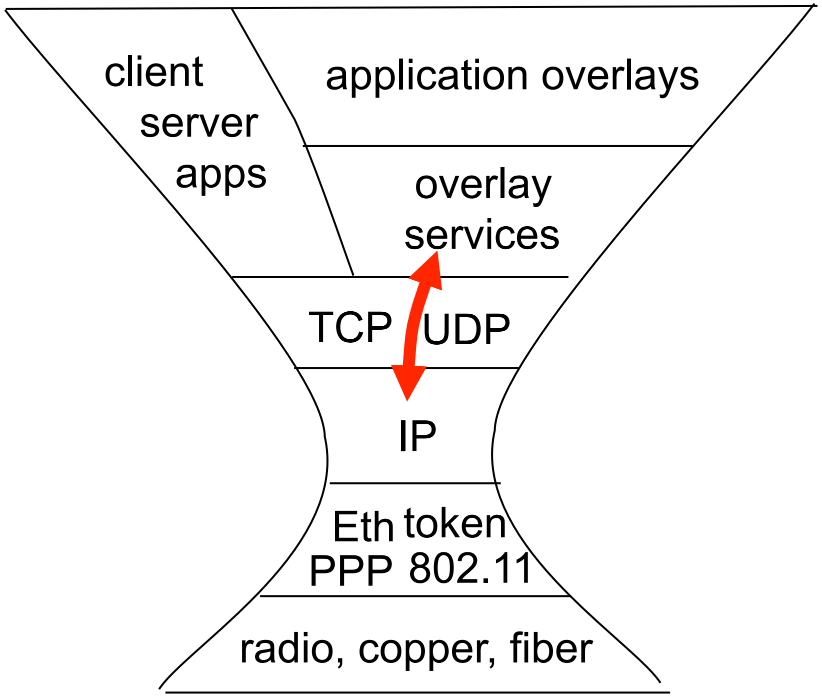


HTTP/IP “long-neck hourglass”

Big picture: supporting new applications – losing the IP hour glass figure?



IP “hourglass”



Future Internet concepts

- Shortcomings and „Future Internet“ concepts
  - Security
  - QoS, multicast
  - Economic implications, „tussle space“
  - Mobility and Locator–ID split
  - In-network congestion control
  - Modules instead of layers
  - Delay-tolerant/disruption-tolerant networking
  - Content-based networking/Publish–subscribe architectures
  - Evolutionary vs. Revolutionary/Clean-slate

FIND: Future Internet Network Design

- New long-term US NSF initiative
- Questions:
  - Requirements: for the global network of 15 years from now - what should that network look like and do?
  - How would we re-conceive tomorrow's global network today, if we could design it from scratch?
- Major thrusts:
  - Security, manageability, mobility (DTN, naming, wireless)
  - I.e.: what the original Internet didn’t get right

The Internet has no built-in security (I)

- Problem #1: Cannot protect from unwanted traffic
  - Spam
  - DoS attacks
  - Wustrow, Karir, Bailey, Jahanian, Huston: *Internet background radiation revisited.* Proceedings of ACM/USENIX Internet Measurement Conference, 2010
- Solutions
  - Protocols
    - DKIM
    - Cookies (e.g., TCP SYN cookies)
  - Treating the symptoms
    - Spam filters
    - Rate limiting at firewall
    - Tar pits, honey pots
    - Network intrusion detection systems (NIDS)
    - …

The Internet has no built-in security (II)

- Problem #2: Traffic not encrypted by default
  - E-Mail, Web: readable by attackers
- Problem #3: Traffic not authenticated by default
  - E-Mail, Web: can be manipulated/forged

- Solutions
  - IPSec
  - SSL/TLS
  - ssh
  - …but do they work?

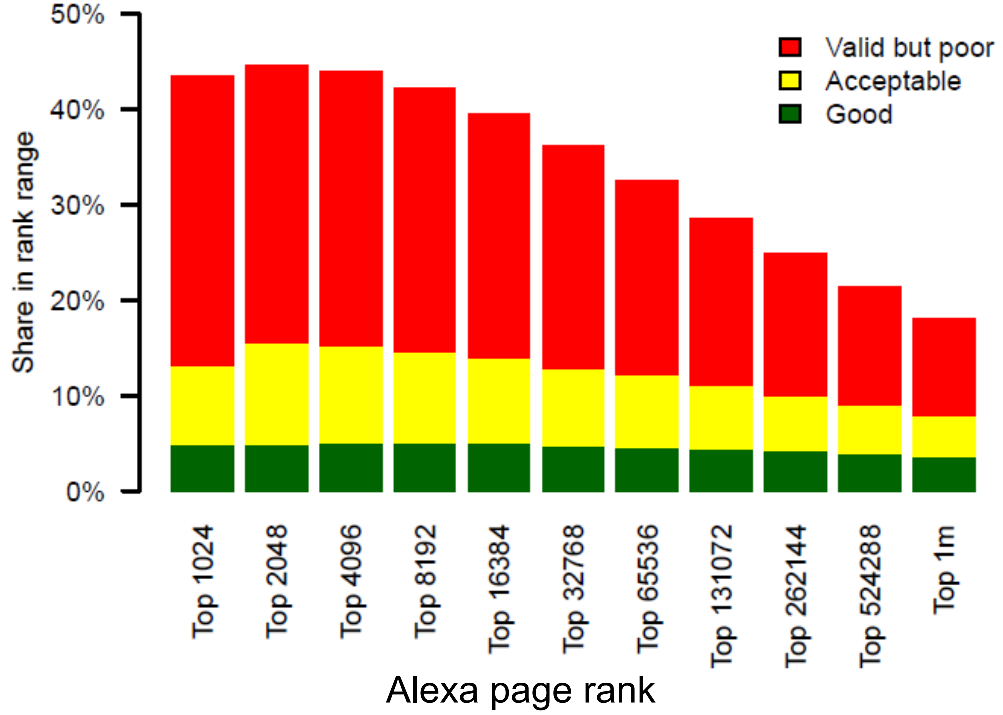
Problems with X.509 certificates (I)

- X.509 certificates: Used for, e.g., SSL/TLS

- Every root CA, every intermediate CA can issue certificates for *any* domain. Example:
  - Authoritarian regime installs transparent HTTPS proxy…
  - ...and gains access to some intermediate CA
  - Proxy intercepts all HTTPS connections, answers with *valid(!)* certificate to client (MITM attack)
  - Client thinks it talks to HTTPS server – in fact proxy can read everything in plaintext
  - You can buy such boxes for a couple of 1,000$
  - (Firefox plugins for detection: CrossBear, CertificatePatrol, CertificateWatch,…)

Problems with X.509 certificates (II)

- Poor administrative knowledge
- Example: Certificate quality in top 1 million Web sites



- Taken from:
  Holz, Braun, Kammenhuber, Carle: *The SSL landscape – a thorough analysis of the X.509 PKI using active and passive measurements.* Proceedings of ACM/USENIX Internet Measurement Conference (IMC), 2011

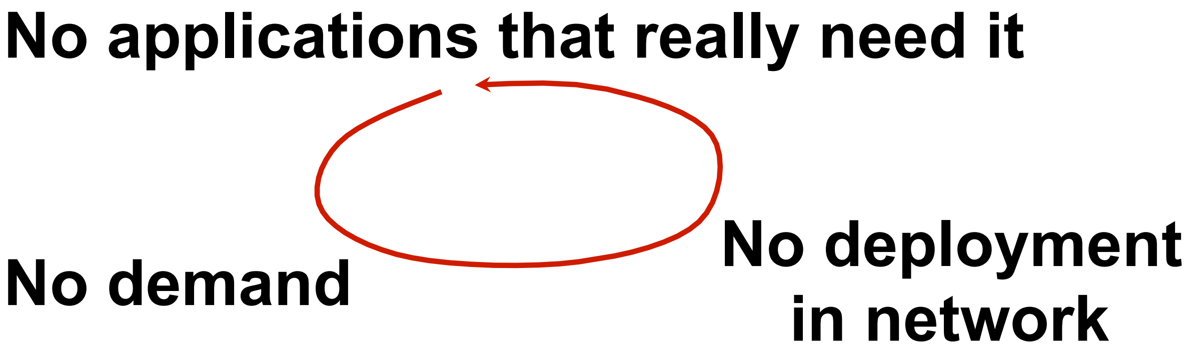
Multicast and QoS

- Multicast routing protocols (MOSPF, PIM, …) exist and work
- QoS protocols (IntServ, DiffServ, …) exist and work
- IP header *and* Ethernet header (802.1p) contain ToS bits

- …but no end user application is using it!
  - Multicast: Would be nice for online TV
  - QoS: Would be nice for throttling P2P and ftp downloads while increasing responsiveness of ssh and games and stability of VoIP calls and video streaming
- At least some „invisible“ usage
  - Prioritization of specific traffic within company networks
  - ISPs may give QoS guarantees for VPNs
  - TV over IP („Triple play“) uses multicast, but application not directly accessible by user

Why don’t ISPs offer multicast or QoS to end users?

1. Same chicken–egg problem / vicious circle as with IPv6:



No applications that really need it

No demand

No deployment in network

2. Who should pay once traffic crosses AS boundaries?
   - Who pays „expedited forwarding“?
     Sender AS, receiver AS, both?
   - Who pays in-network duplication for multicast?
     Sender AS, receiver ASes, or entire network?
   - How can sender/receiver be charged?
   - How can multicast sender know how much it will be charged?

Economic aspects, conflicting interests: „Tussle“

- Internet participants
  - Different stakeholders
  - Competition
  - Conflicting interests
- Examples
  - Users want to share music and videos – GEMA/RIAA don‘t
  - Users want secret communication – governments don‘t
  - ISPs need to cooperate – but are fierce competitors
- Call this aspect „tussle“
  - Internet architecture only partially reflects this (BGP policy routing)
  - Tussle Space: Future Internet architecture should anticipate various kinds of tussle and integrate defined mechanisms

Clark, Sollins, Wroclawski, Braden: Tussle in Cyberspace: Defining Tomorrow‘s Internet. Proceedings of ACM SIGCOMM, 2002

Mobility, Locator–ID split: Problem

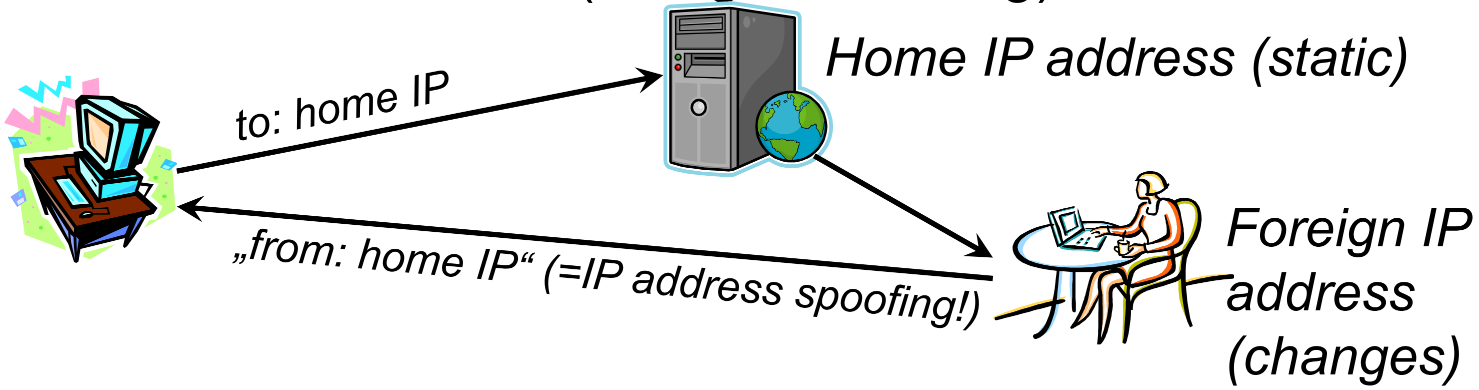
- **Identifier:** IP address identifies communication endpoint
  - Keywords: TCP 4-tuple (srcIP, dstIP, srcP, dstP), DNS entry, …
- **Locator:** IP address specifies how to reach destination
  - Keywords: Netmask, longest prefix match, CIDR, …
- Problem: What if IP addresses change?
  - Scenario 1: User mobility
    Example: Lose WLAN connection, switch to UMTS/LTE
    ➔ IP address changes
    ➔ All active TCP, UDP connections break: ssh, Jabber,…
  - Scenario 2: Network mobility
    Example: Middle-sized company switches to a different ISP
    ➔ *All* IP addresses of *all* their hosts need to be changed
    ➔ High maintenance effort; cannot switch instantaneously
  - Scenario 3: IP anycast = one IP address, but multiple hosts.
    Example: Some DNS root servers use one IP address for multiple servers at entirely different locations

Mobility, Locator–ID split: Solutions (1)

- Dynamic DNS
  - Assumptions:
    - Mostly use short-lived connections
    - Mostly connect to host names, not IP addresses
  - Idea:
    - Keep short-lived DNS entries
    - If IP address changes, immediately update DNS entry
  - Drawbacks:
    - Service unavailable for several minutes (until new old entry has expired, new entry has propagated)
    - Some faulty DNS servers ignore short-lived timeout value
    - Does not help active connections
    - Does not help connections that do not use DNS

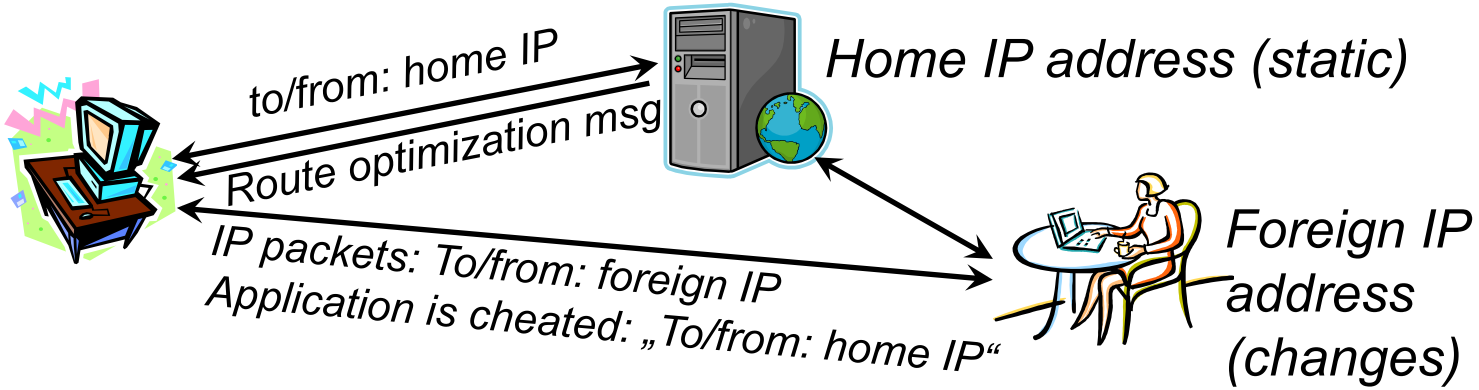
Mobility, Locator–ID split: Solutions (2)

- Mobile IP
  - Old standard: Mobile IPv4 (triangular routing)



to: home IP
Home IP address (static)
„from: home IP“ (=IP address spoofing!)
Foreign IP address (changes)

    - Incompatible with firewalls, ingress filtering, …
  - New standard: Mobile IPv6



to/from: home IP
Route optimization msg
Home IP address (static)
IP packets: To/from: foreign IP
Application is cheated: „To/from: home IP“
Foreign IP address (changes)

  - **Drawbacks: Both require a Home Agent**

Mobility, Locator–ID split: Solutions (3)

- Host Identity Protocol (HIP)
  - Additional HIP layer between IP and transport (e.g., TCP)
  - Every host has *static* 128-bit Host Identifier
    - Identifier „looks“ like an IPv6 address to transport protocol
  - Two hosts that want to communicate initiate a HIP session
    - Exchange of Host Identifiers
    - Exchange of crypto keys
  - If IP address of one host changes:
    - Send information address change to other HIP partner
    - Will send future HIP traffic to new IP address
    - Information cryptographically signed ➔ no connection hijacking
  - Drawbacks
    - One additional RTT for HIP handshake at start of connection
    - Not transparent – need changes in operating system
    - Both communication partners need to support HIP

Mobility, Locator–ID split: Solutions (4)

- Locator–ID Separation Protocol (LISP)
  - Use some IP addresses as locators, use others as identifiers
  - End hosts / end networks only see identifier IPs
  - Network core only sees locator IPs
  - Addresses become dynamically re-written (similar to NAT) upon arrival at / departure from LISP-enabled network
  - Moving host, moving network: Update address rewrite tables
  - Good: Incrementally deployable; transparent
  - Bad: Not really for end hosts (scalability); not yet supported
- SCTP
  - SCTP association knows all IP addresses of both endpoints
  - If primary connection fails: transparent switch-over
  - Drawback: Only works with SCTP… but nobody uses SCTP!

Shortcoming: No in-network congestion control

- Congestion control today
  - End hosts: Short timescales
    - TCP
    - Others (e.g., DCCP): Should be TCP-friendly
    - Disadvantages:
      - No enforcement (e.g., UDP)
      - Can only adjust speed; cannot select better path
  - Network: Long timescales
    - Traffic engineering: Measure traffic, reconfigure routing
    - EIGRP
    - No cooperation across AS boundaries
  - Why not at shorter timescales?
    - Bad experience in ARPANET
    - Highly nonlinear system: prone to oscillation
    - Interaction with TCP congestion control → even worse

Layers vs. Modules („Functionality Lego“)

- Observation: Many functionalities implemented multiple times at multiple layers. Examples:
  - Encryption and authentication:
    ssh (Application), SSL (Session), IPSec (Network), GSM/UMTS/LTE (Data Link)
  - Flow control:
    TCP (Transport), Ethernet and WLAN (Data Link)
  - Guaranteed delivery through ACKs/resends:
    Custom protocols (Application), TCP (Transport), high-loss satellite links (Data Link)
- Idea:
  - Encapsulate specific functionality within modules
  - Ensure that modules can be plugged together in (more or less) arbitrary combination and sequence
  - Application/communication endpoints (and network?) specify „building plan“ during initial handshake

DTN: Delay-tolerant networking / Disruption-tolerant networking (I)

- ❑ Andrew Tanenbaum: „Never underestimate the bandwidth of a station wagon full of tapes hurtling down the highway.“
  - Send small packet with SD cards or hard disk (1 TByte)
  - Let journey time be 1 week (➔ RTT = 2 weeks!)
  - Bandwidth = around 13 Mbit/s!
- ❑ Underdeveloped regions: Send data via, e.g.,
  - Letters/packets containing storage media
  - Messengers carrying storage media
  - Homing pigeons ☺ („IP over avian carriers“, RFC1149 et al.)
  - WLAN-/Bluetooth-equipped phones/laptops/… that can exchange data in passing and cache it during transit
- ❑ Also could be used during emergency with large-scale infrastructure failures (e.g., Hurricane Katrina)
- ❑ Similar characteristic: Space travel! (Very long delays; long connection breaks, e.g., when spacecraft behind a planet)

DTN: Delay-tolerant networking / Disruption-tolerant networking (II)

- Protocols: No „gold standard“ yet
  - Vastly different scenarios (e.g., underdeveloped regions vs. space travel)
- Protocol/application selection
  - Bundle Protocol
  - Lidlicker
  - Saratoga
  - Offline browsing proxy (WWWoffle)
- Experiments/prototype deployments
  - Some in Lapland, some in South Africa
  - EU project to connect remote villages in Slovenia
- Future research includes:
  - Routing algorithms
  - Gateways and interfaces to existing services (Mail, Web, …)

Content-based networking and publish–subscribe architectures (I)

- Observation:
  - IP addresses *hosts*
  - Browsers, P2P clients etc. address *content objects:* Specific Web pages, MP3 files with specific music, …
- Idea:
  - Address content chunks instead of hosts
  - Routers can replicate and/or cache popular chunks
- Requesting chunks:
  - Send interest/subscription request into network
  - Request will be forwarded from router to router
  - If matching content chunk(s) found, send them to requester

Content-based networking and publish–subscribe architectures (II)

- A lot of features automatically built in:
  - Multicast (even asynchronously!)
  - In-network caching
  - Resilience: If one router with content fails, it still will be available on other routers
  - Delay-tolerant networking: Routers cache contents anyway, so why not have the caching routers roam around as well?
  - Some protection from DoS attacks: I only get traffic that I requested

Content-based networking and publish–subscribe architectures (III)

- ❑ Some issues to be addressed
  - Authenticity: How to make sure that malicious users cannot inject a fake version of, e.g., an online banking service?
  - Routing: How do routers know which interest packets should be forwarded to which neighbour(s)?
  - Versioning: How to make sure that old versions of a content object are quickly replaced in router caches (e.g., content object „current DAX level“ or „Mensa food plan“)
  - Protocol logic:
    - Subscription („send me all matching chunks“) vs. requests („send me one matching chunk“)
    - Timeouts
  - Protection from flooding induced by excessive subscription
  - Addressing scheme

Content-based networking and publish–subscribe architectures (IV)

- OK, sounds good for things like YouTube, heise.de, etc.
- But what about obvious peer–peer sessions? (ssh, VoIP, etc.)

- Solution:
  - Subscribe to contact requests
  - If contact request is received, subscribe to answer packets of contact request originator
  - Start sending out own data (e.g., own voice)
  - Receive answers from peer (e.g., acknowledgement packets; other‘s voice)

Content-based networking and publish–subscribe architectures (V)

- Some thoughts on the address length: How much do we need?
- Current Internet
  - IPv4: 32 bits = 4 billion addresses (about 30% used)
  - IPv6: 128 bits
- Consider something like a worst-case scenario:
  - Assume *every atom* is used to store one information chunk!
    - About 10^{80} particles in the visible universe
  - Every chunk changes its state every 10^{-44}s! (Planck Time)
  - For 1 million years!
  - We waste 99% of the address space! (IPv4: only 60% wasted)
  - How many bits do we need?
    - $\log_2\left(10^{80} \cdot (10^{44} \cdot 60) \cdot (60 \cdot 24 \cdot 365 \cdot 10^6) \cdot 100\right)$

      = 463 bits = 58 Bytes. (N.B.: IPv6 header+TCP header = 56 Bytes)
    - One of the rare cases where exponential growth is in our favour!

Future Internet approaches

Revolutionary (clean slate)

- ❑ Today's Internet is broken by design
- ❑ Trying to fix it leaves us with *-over-HTTP-over-TCP-over-IP, i.e., with something like the memory model of Intel x86, the A20 gate, 110V vs. 230V and 50Hz vs. 60Hz power, …
- ❑ New architecture will be radically different
- ❑ ➔ Let's throw everything away and start completely anew to ~~get it right from the beginning~~ introduce new design mistakes

Evolutionary

- ❑ The Internet has been amended many times in the past:
  - Adding congestion control to TCP
  - Introduction of DNS instead of distribution of `/etc/hosts` text files
  - Introduction of classless interdomain routing instead of Class-A, Class-B, Class-C networks
  - Introduction of SSL, IPSec, ssh, …
  - Introduction of Multicast, ToS bits
  - Introduction of IPv6
- ❑ ➔ Let's fix the shortcomings incrementally by introducing new protocols: ~~Never change a running system~~ Create a truly unmanageable behemoth of conflicting protocols

Future Internet: Some readings

- Mark Handley: *Why the Internet only just works.* BT Technology journal, 2006
- Anja Feldmann: *Internet Clean-Slate Design: What and Why?* Editorial note, ACM CCR, 2007
- Akhshabi, Dovrolis: *The evolution of layered protocol stacks leads to an hourglass-shaped architecture.* Proceedings of ACM SIGCOMM, 2011

- N.B. With a TUM or LMU IP address, you can download most scientific articles for free if you enable the LRZ proxy: http://www.lrz.de/services/netzdienste/proxy/journals-access/

The end!