Introduction to Internet Protocols

Henning Schulzrinne
Dept. of Computer Science
Columbia University
schulzrinne@cs.columbia.edu

DIMACS Tutorial
Performance of Real-time Applications on the Internet

November 5, 1996

Goals and overview

- understanding basic Internet architecture
- common Internet protocols
- real-time services

Internet Architecture

The Internet is ...

- global interconnection of networks
- technology-independent ("IP over everything")
- run by competing/cooperating for-profit/not-for-profit companies/organizations/universities
- no longer subsidized (almost...)

Intranet $\hat{=}$ internal (corporate) network using Internet protocols and applications

A Short History of the Internet: 1960's

1830: telegraph

1876: circuit-switching (telephone)

early 1960's: concept of packet switching (Paul Baran)

1965: MIT's Lincoln Laboratory commissions Thomas Marill to study computer networking

1968: ARPAnet contract awarded to Bolt Beranek and Newman (BBN)

1969: ARPAnet has 4 nodes (UCLA, SRI, UCSB, U. Utah), connected by IMPs (interface message processors); connected by 50 kb/s lines

A Short History of the Internet: 1970's

- multiple access networks: ALOHA, Ethernet (10 Mb/s)
- companies: DECnet (1975), IBM System Network Architecture (1974)

1971: 15 nodes and 23 hosts: UCLA, SRI, UCSB, U of Utah, BBN, MIT, RAND, SDC, Harvard, Lincoln Lab, Stanford, UIU(C), CWRU, CMU, NASA/Ames

1972: First public demonstration at ICCC

1973: TCP/IP design

1973: first satellite link from California to Hawaii

1973: First international connections to the ARPANET: England and Norway

1979: ARPAnet $\approx$ 100 nodes

A Short History of the Internet: 1980's

- proliferation of local area networks: Ethernet and token rings
- late 1980's: fiber optic networks; fiber distributed data interface (FDDI) at 100 Mb/s

1980's: DARPA funded Berkeley Unix, with TCP/IP

1981: Minitel deployed in France

1980-81: BITNET (IBM protocols) and CSNET (NSF-funded $\rightarrow$ 200 sites)

Jan. 1, 1983: flag day: NCP $\rightarrow$ TCP

early 1980's: split ARPANET (research), MILNET (military)

1984: Domain Name Service replaces hosts.txt file

1986: NSFNET created (56 kb/s backbone)

Nov. 1, 1988: Internet worm

1989: Internet passes 100,000 nodes

1989: first proposal for World-Wide Web

1989: NSFNET backbone upgraded to T1 (1.544Mbps)

A Short History of the Internet: 1990's

- high-speed networks: Asynchronous Transfer Mode (ATM) at 150 Mb/s and higher
- focus on new applications
- wireless local area networks
- commercialization
- National Information Infrastructure (NII) (Al Gore, U.S. VP)

1990: Original ARPANET disbanded

Fall 1991: CSNET discontinued

1991: Gopher released by University of Minnesota

1992: NSFNET backbone upgraded to T3 (44.736Mbps)

March 1992: First MBONE audio multicast

November 1992: First MBONE video multicast

February 1993: NCSA Mosaic

June 1993: 1,776,000 hosts

April 30, 1995: NSFNET backbone disbanded

July 1996: 12,881,000 hosts, 2,569,000 direct ping'able

Basic Internet Components

- rules and regulations
- infrastructure
- addresses: MAC (Ethernet), ATM $\leftrightarrow$ IP addresses $\leftrightarrow$ domain names, URLs, …
- protocols: network, transport,
- applications

Who makes the rules?

ITU.T: telecom standards: G.??, X.??, V.??

IANA: (Internet Assigned Numbers Authority) assigns numbers,
top-level domains

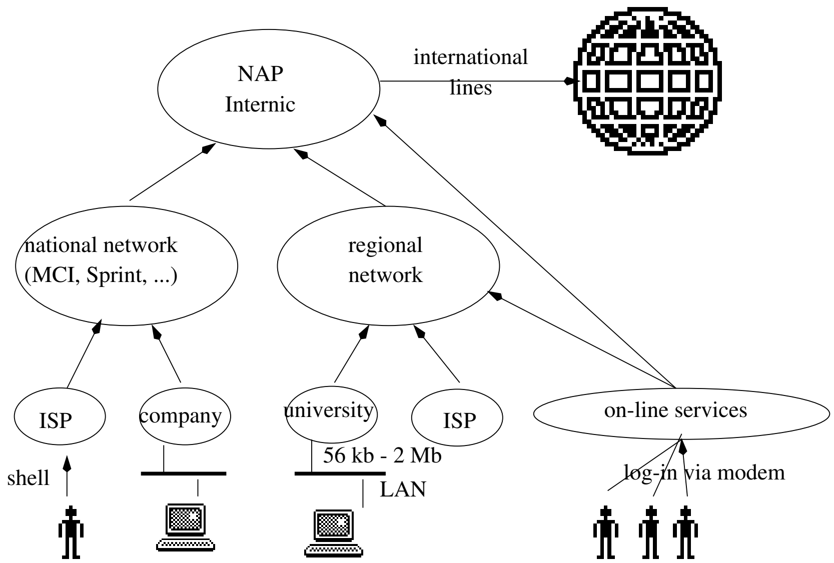
IETF and IESG: (Internet Engineering Task Force, … Steering Group)
develop/bless protocols ("RFCs")
open admission, but not quite egalitarian
most RFCs are not standards!

NANOG: North American Network Operators Group

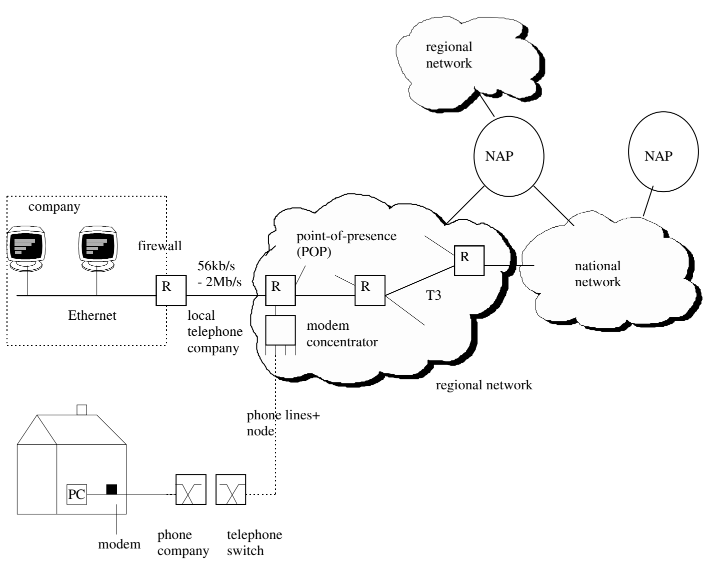
Internic, …: hands out names, blocks of addresses

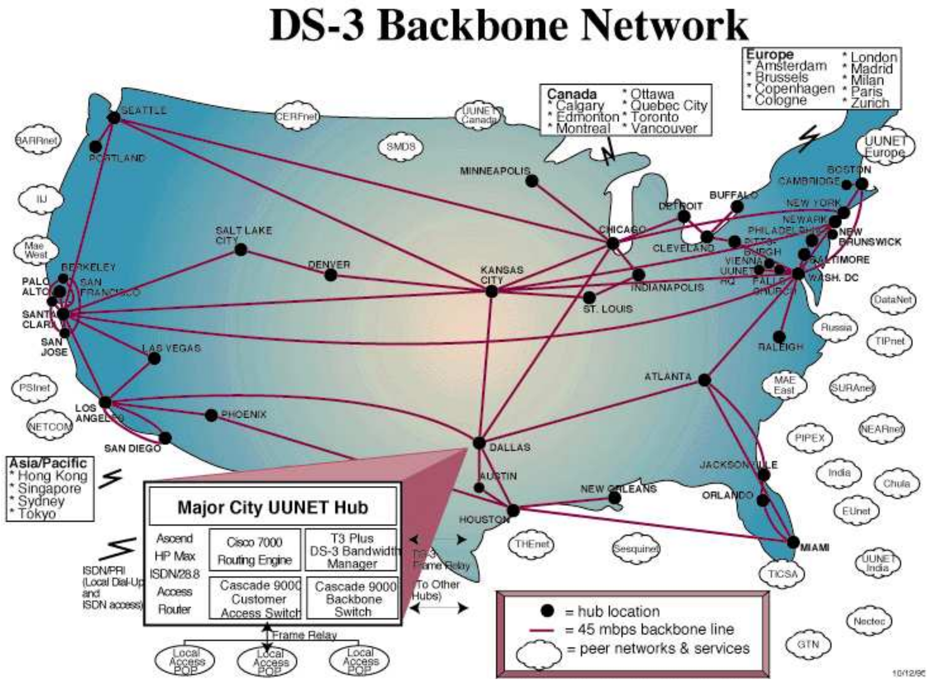
Lots of others *want* to get involved…

Who pays for the Internet?



Network Access and Interconnection





Internet Protocols

Internet characteristics

- connectionless datagrams within the network ➠
  - information carried in self-contained *packets* of $\approx 80 \ldots 1000$ bytes
  - each packet independent of others ➠ no “call setup”
  - routes periodically determined by distributed routing algorithm
- layered protocols:

  data link layer: aka *subnetwork*, MAC layer

  point-to-point: ISDN, SONET, ATM, frame relay, PPP (packetization for bit-serial services) . . .

  broadcast LANs: Ethernet, token ring, FDDI

  network layer: routing across subnetworks in IP packet ➠ routers

  transport layer: end-to-end, *may* add reliability

  application protocols: remote login, web browsing, file transfer, . . .

Architectural principles of the Internet

RFC 1958:

- large existing infrastructure ➠ backward compatibility
- “The goal is connectivity, the tool is the Internet Protocol, and the intelligence is end to end rather than hidden in the network”
- End-to-end functions can best be realised by end-to-end protocols.
- Nobody owns the Internet, there is no centralized control, and nobody can turn it off.

Principles: Design

- Heterogeneity is inevitable and must be supported by design.
- If there are several ways of doing the same thing, choose one.
- All designs must scale readily to very many nodes per site and to many millions of sites.
- Performance and cost must be considered as well as functionality.
- Keep it simple.
- In many cases it is better to adopt an almost complete solution now, rather than to wait until a perfect solution can be found.
- Avoid options and parameters whenever possible. Configure them automatically.
- Be strict when sending and tolerant when receiving.
- Be parsimonious with unsolicited packets.

- Circular dependencies must be avoided.
- Objects should be self decribing (include type and size).
- All specifications should use the same terminology and notation, and the same bit- and byte-order convention.
- Nothing gets standardised until there are multiple instances of running code.

Principles: names and addresses

- No hardcoded addresses.
- Single naming structure.
- Names should be case-insensitive ASCII.
- Addresses must be unambiguous.
- Upper-layer protocols must be able to identify end points unambiguously.

Principles: Soft state vs. hard state

Need to coordinate views of the world in a distributed system ➠

hard state: traditional telecom ➠ state is precious:

- handshake, with retransmission on timeout
- explicit teardown of state

soft state: “optimistic” approach ➠ state is restorable

- send periodic messages to set up or refresh state
- no explicit teardown (except for efficiency) ➠ time-out state
- works even if routers, hosts fail or suffer amnesia
- sometimes allows selective state maintenance
- simpler? higher state overhead? recovery time?

Principles: protocols

- small, single-function protocols
- lower-layer protocols: attention to processing efficiency ➠ word alignment, fixed-length fields: CLNP vs. IPv4/IPv6
- incremental: small applications easy ➠ embed email, HTTP, …
- upper-layer protocols are text-based ➠
  - "the universal telnet protocol simulator"
  - lots of tools ($\approx$ Unix)
  - space + time overhead
  - difficult to represent nested data structures
  - typically ASCII ➠ internationalization required
- most upper layer protocols (including NFS, ftp, HTTP, SMTP, …) are basically RPC

Internet names and addresses

	example	organization
MAC address	8:0:20:72:93:18	flat, permanent
IP address	132.151.1.35	topological (mostly)
Host name	www.ietf.org	hierarchical

host name $\stackrel{\mathrm{DNS,many-to-many}}{\rightarrow}$ IP address $\stackrel{\mathrm{ARP,1-to-1}}{\rightarrow}$ MAC address

Internet addresses

- 32-bit addresses
- (almost) every *interface* has one
- but may
  - change (dial-in)
  - have lots (WWW servers)
  - have none (some routers)
- old: class-$\{$A,B,C$\}$ ➠ 2-level addressing: network,host
- new: classless interdomain routing (CIDR) ➠ aggregation, route on prefix and mask

IP address classes

class	first octet	hosts per network	nets	used (7.96)
Class A	< 128	16 mio.	128	95
Class B	$128 \ldots 191$	65534	16384	5892
Class C	$192 \ldots 223$	254	2 mio.	128378
Class D	$224 \ldots 239$		268 mio.	dynamic
Class E	$240 \ldots 255$		134 mio.	reserved

- distinguished by prefix
- similar to ISBN, European telephone numbers

The Internet domain name system

- hierarchical, dot-separated names
- ➠ multi-level delegation
- by country and by type of organization
- needs to be overhauled (63% of all domains = .com!)

Top-level domains:

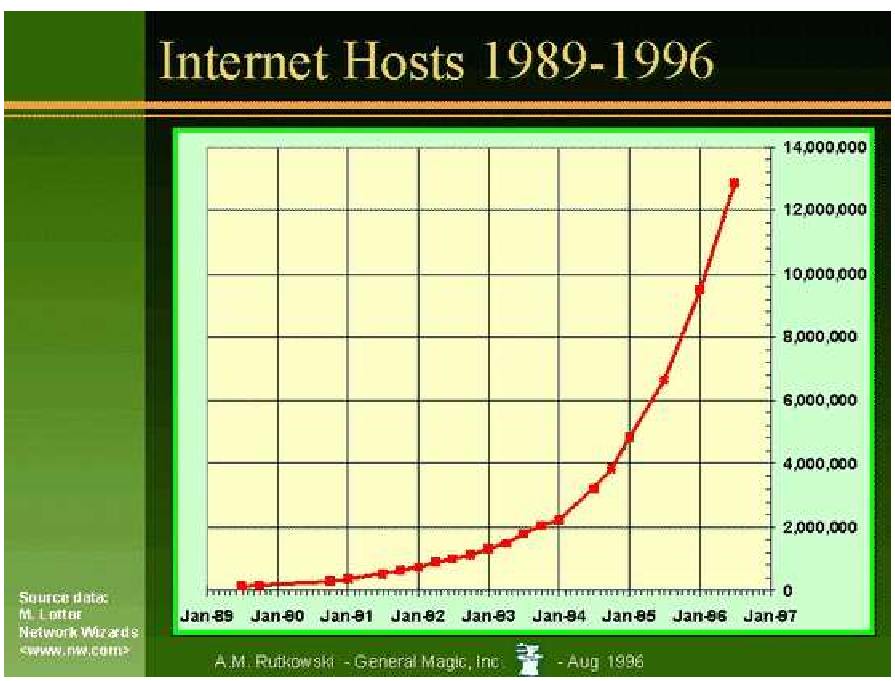
2 letters:	countries
3 letters:	independent of geography (except edu, gov, mil)

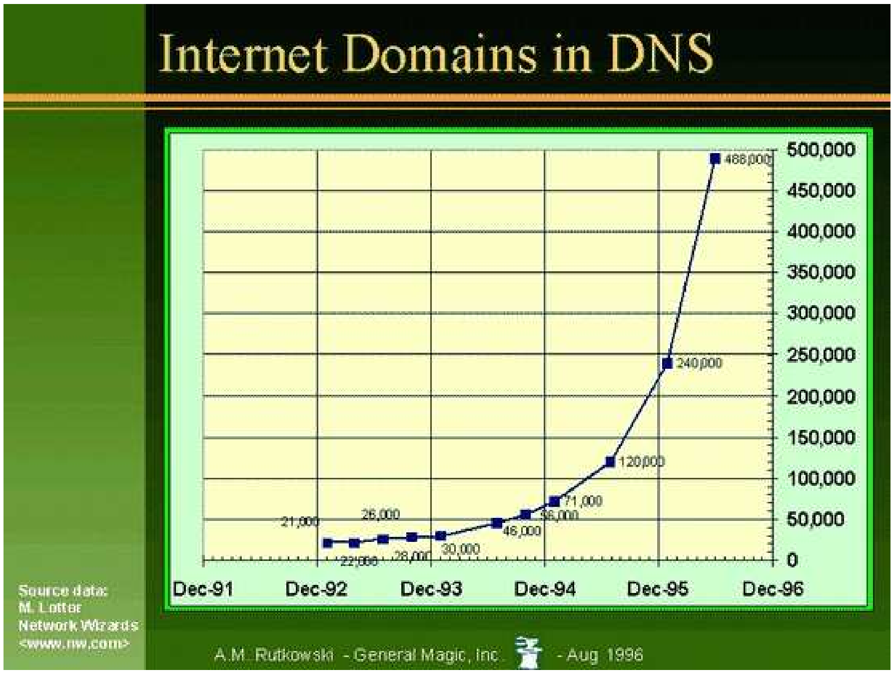
Internet domains

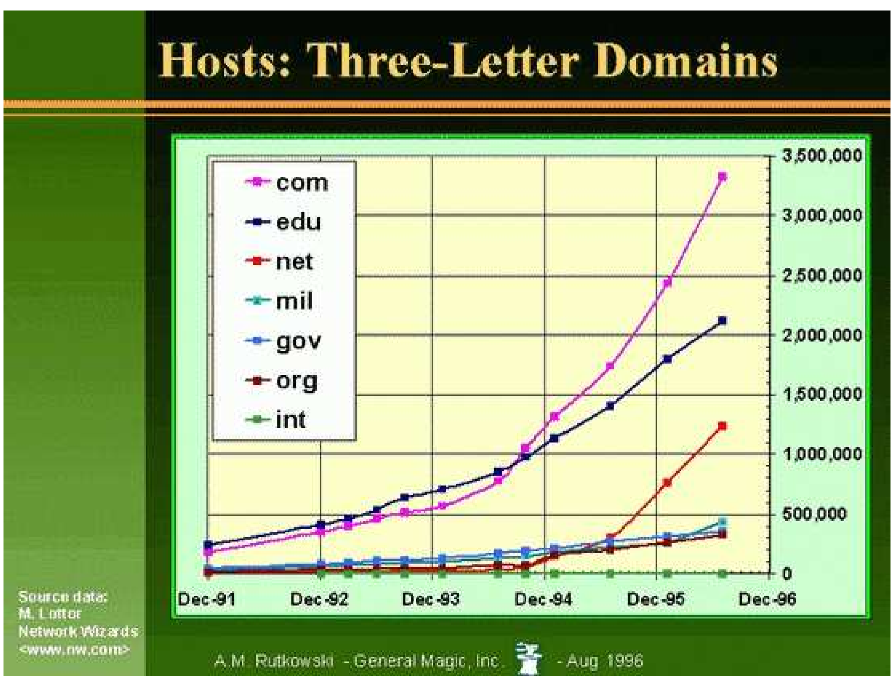
domain	usage	example	hosts (7.96)
com	business (global)	research.att.com	3323647
edu	U.S. 4 yr colleges	cs.columbia.edu	2114851
gov	U.S. non-military gov't	whitehouse.gov	361065
mil	U.S. military	arpa.mil	431939
org	non-profit orgs (global)	www.ietf.org	327148
net	network provider	nis.nsf.net	1232902
us	U.S. geographical	ietf.cnri.reston.va.us	432727
uk	United Kingdom	cs.ucl.ac.uk	579492
de	Germany	fokus.gmd.de	548168

Domain name resolution

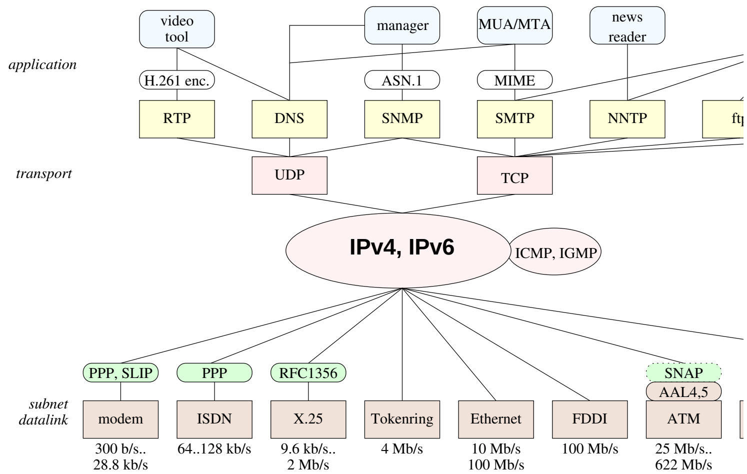
- hierarchy of redundant servers with time-limited cache
- each server knows the (9) root servers
- each root server knows
- each domain has ≥ 2 servers, often widely distributed
- also: mailbox translation
- *almost* a distributed database




Internet Domains in DNS
500,000
450,000
400,000
350,000
300,000
250,000
200,000
150,000
100,000
50,000
0
488,000
240,000
120,000
71,000
56,000
46,000
30,000
28,000
26,000
22,000
21,000
Dec-91
Dec-92
Dec-93
Dec-94
Dec-95
Dec-96
Source data:
M. Lottor
Network Wizards
<www.nw.com>
A.M. Rutkowski - General Magic, Inc. - Aug 1996


Hosts: Three-Letter Domains
com
edu
net
mil
gov
org
int
3,500,000
3,000,000
2,500,000
2,000,000
1,500,000
1,000,000
500,000
0
Dec-91
Dec-92
Dec-93
Dec-94
Dec-95
Dec-96
Source data:
M. Lottor
Network Wizards
<www.nw.com>
A.M. Rutkowski - General Magic, Inc. - Aug 1996

The Internet protocol zoo



Protocol taxonomy

Data link layer shim: SLIP, PPP

network layer + support: IPv4, IPv6; ARP, DHCP, BOOTP

transport layer: UDP, TCP

directory: LDAP

information access: HTTP, ftp, telnet

file sharing: NFS, WebNFS

electronic mail: SMTP, POP, IMAP

Many upper-layer protocols are basically RPC. . .

IPv4

- independent packets
- unreliable
- might be reordered, delayed, ...
- but: minimal service on top of *anything* (see RFC 1149)
- only header checksum

IPv6

- 32 bits ➠ $4 \cdot 10^9$ hosts, but can't number sequentially
- aggregation for routing $\leftrightarrow$ efficiency
- every $\{$light bulb, toaster, fire alarm, ...$\}$ with IP address ➠ address shortage
- 128 bits instead of 32 bits ➠ 340,282,366,920,938,463,463,374,607,431,768,211,456 ($3 \cdot 10^{38}$) hosts or 665,570,793,348,866,943,898,599 hosts/m^2 of earth surface
- hierarchical assignment ➠ inefficiency ➠ only $8 \cdot 10^{17}$ to $2 \cdot 10^{33}$ addresses if similar to telephones, etc.

IPv6: While we're at it…

- *flows*: short handle for group of packets (but $\neq$ connection)
- destination address names *intermediate* destination
- well-defined priority field
- no header checksum
- unlimited option space
- jumbograms, TTL changed, e-e fragmentation, …
- ➠ 40-byte header (instead of 20)

Transport I: Universal Datagram Protocol (UDP)

- same service as IP
- adds *ports*: differentiation of services within host
- adds payload checksum
- overhead: 8 bytes

Transport II: Transmission Control Protocol (TCP)

- reliable, sequenced byte stream on top of datagram
- connections with graceful close or hard reset
- windowed congestion control: not part of the protocol spec…
- windows:
  - flow control: avoid overrunning slow receiver ➠ advertised by receiver
  - congestion control: avoid network overload ➠ maintained by sender – *cwnd*
- probe for available bandwidth:
  - *slow start*: add one segment size for each ACK ➠ exponential
  - *congestion avoidance: cwnd* += 1/*cwnd* for each ACK
  - loss ➠ timeout, retransmit and reduce *cwnd*

- bandwidth-delay product: capacity of "pipe" ➠ continental T3 = 45 Mb/s $\times$ 5000 km $\times$ 5 μs / km $\approx$ 140 kB
- extensions: large (> 64 kB) windows for fat/long pipes, SACK
- overhead: 20 bytes

Internet Real-Time Services

Overview

- multicast
- resource reservation protocols
- transport services for real-time data

Broadcast and multicast

broadcast: all hosts on (small, local) network

directed broadcast: all hosts on remote network

multicast: multiple recipients (group)

Applications for multicast

- audio-video distribution (1-to-many) and symmetric (all-to-all)
- distributed simulation (war gaming)
- resource discovery
- file distribution (stock market quotes, new software, . . .)

Connection-oriented multicast

- enumerate sources explicitly
- examples:
  - ATM (explicit connection to each end point)
  - ST-II (several end points in setup message)
  - enumeration of end points in packet
- only connection-oriented (packet header size!)
- source needs to know destinations $\leftrightarrow$ resource discovery, dynamic groups

Host group model

Deering, 1991:

- senders need not be members;
- groups may have any number of members;
- there are no topological restrictions on group membership;
- membership is dynamic and autonomous;
- host groups may be transient or permanent.

Local multicast

Some local networks are by nature multi/broadcast: Ethernet, Token Ring, FDDI, . . .

Ethernet, Tokenring:

- broadcast: all ones
- multicast: 01.xx.xx.xx.xx.xx
- adapter hardware can filter dynamic list of addresses

ATM: point-to-point links ➠ need ATM multicast server

IP multicast

- host-group model
- network-level; data packets same, only address changes
- need help of routers
- special IP addresses (class D): 224.0.0.0 through 239.255.255.255
- 28 bits ➠ 268 million groups (plus scope)
- 224.0.0.1: all hosts
- some pre-assigned (224.0.1.2: SGI Dogfight)
- others dynamic (224.2.x.x for multimedia conferencing)
- map into Ethernet: 01.00.5E.00.00.00 + lower 23 bits

Internet Group Management Protocol (IGMP)

Multicast for local (broadcast) networks, between router and hosts

- router listens to all multicast packets on all interfaces
- hosts sends report for first process to join group to multicast group
- host *does not* send report when processes all have left
- router multicasts query to all hosts $\approx$ once a minute
- host waits and listens for others; if nobody else, send response

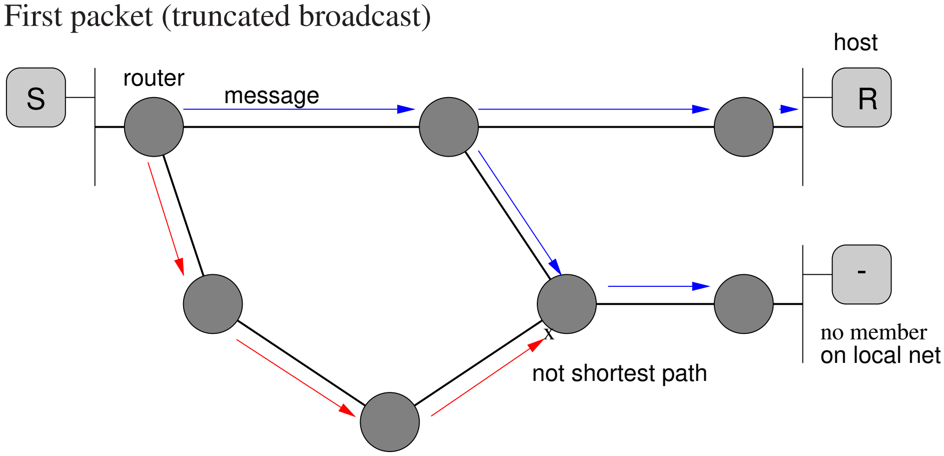
Multicast Forwarding

1. check incoming interface: discard if not on shortest path to source
2. forward to all outgoing interface except incoming
3. don't forward if interface has been *pruned*
4. prunes time out every minute
5. routers may send *grafts* upstream

routing information: DVMRP

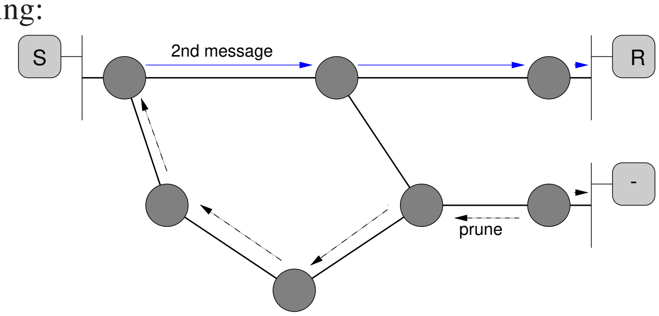
Multicast forwarding

First packet (truncated broadcast)



Multicast forwarding

With pruning:



➠ router needs to keep “negative” list for groups

Multicast programming

UDP, not TCP (obviously…)

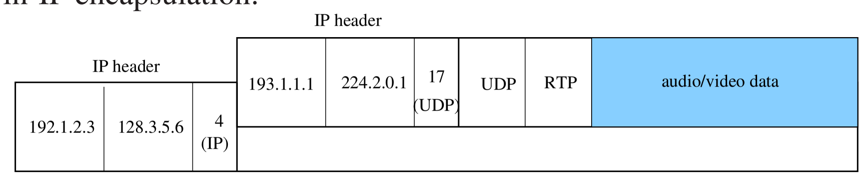
```
struct sockaddr_in name;
struct ip_mreq imr;

sock = socket(AF_INET, SOCK_DGRAM, 0);
imr.imr_multiaddr.s_addr = htonl(groupaddr);
imr.imr_interface.s_addr = htonl(INADDR_ANY);
setsockopt(sock, IPPROTO_IP, IP_ADD_MEMBERSHIP,
  &imr, sizeof(struct ip_mreq));
name.sin_addr.s_addr = htonl(groupaddr);
name.sin_port = htons(groupport);
bind(sock, &name, sizeof(name));
recv(sock, (char *)buf, sizeof(buf), 0);
```

Problems

- “multicast storms”
- state in routers for sparse groups vs. optimal trees
- multicast routing vs. unicast routing (reverse path)
- hierarchical routing

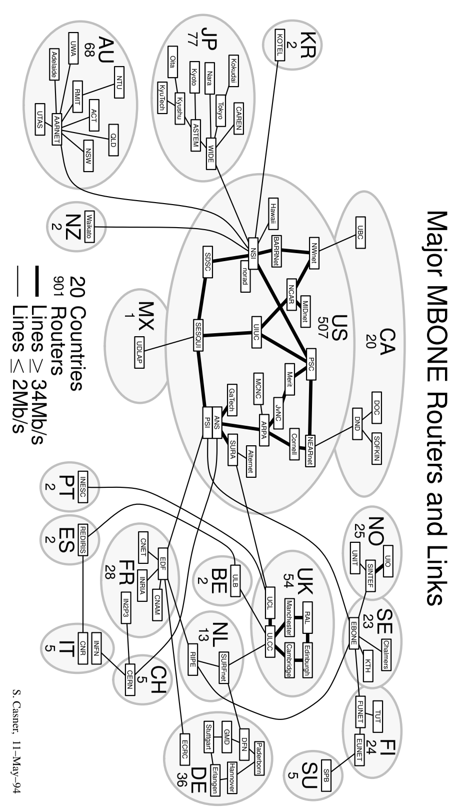
MBONE

- MBONE ≡ multicast backbone
- overlay network over Internet
- needed until deployment of multicast-capable backbone routers
- IP-in-IP encapsulation:



source: 193.1.1.1; group: 224.2.0.1; MBONE tunnel: 192.1.2.3 to 128.3.5.6

- limited capacity, resilience



Multicast: further information

- S. Deering, RFC 1112
- D. Comer, *Internetworking with TCP/IP*
- R. Perlman, *Interconnections - Bridges and Routers* (“IP multicast is bad”)
- C. Huitema, *Routing in the Internet*

Resource Reservation

Resource reservation issues

sender-oriented: sender specifies bandwidth, receivers ➠ ATM, ST-II

receiver-oriented: receivers notify sender ➠ RSVP

hard state: reliable connection set-up (handshake), explicit tear-down ➠ Q.2931, ST-II

soft state: probabilistic set up, time-outs $\leftrightarrow$ periodic state refresh

Any protocol should almost never say "no" (unless it is run by a monopoly...)

RSVP

receiver-oriented reservation protocol being standardized by IETF:

- not a routing protocol, but interacts with routing
- transports *opaque* QOS and policy parameters for sessions
- simplex ➠ setup for unidirectional data flows
- does not prescribe admission or policy control
- sets up packet classifier, but does not handle packets
- independent sessions (can't tie video and audio session)
- multicast (and unicast)
- either own protocol type or UDP encapsulated

RSVP objects

Flow descriptor =

Flowspec: • service class

- Rspec ➠ desired QoS
- Tspec ➠ describes traffic characteristics

Filterspec: which packets get this treatment ➠ sender IP address/port, protocol, other fields ➠ complex (regular expressions? IP options!) ➠ currently, sender IP address and UDP/TCP port ➠ no fragmentation

Reservation styles

sender selection	reservations distinct for each sender	shared
explicit	fixed filter	shared-explicit
wildcard (all)	–	wildcard filter

➠ mutually incompatible

shared: only one active data source ➠ e.g., reserve for twice needed for audio

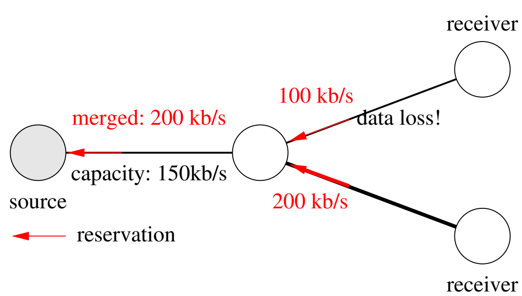
distinct: video

RSVP: basic operation

- source sends PATH messages to receivers: previous hop to source, Tspec, OPWA
- receivers send RESV messages back to senders
- reservations are merged at each node for same sender (max. flowspec)
- merge point or data sender may send confirmation (if requested)
- reservations *may* get merged between senders (audio!)
- one-pass ➠ receiver doesn't know final QoS ➠ One Pass With Advertising
- application *should* explicitly tear down reservations

Killer reservations

1. small reservation in place; another receiver larger reservation ➠ failure? ➠ keep old
2. large reservation fails again and again ➠ blocks new, smaller one



Mapping RSVP onto ATM

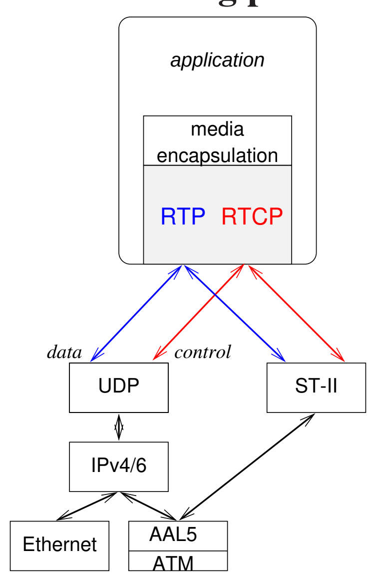
	IP, RSVP	ATM
multicast tree, reservation	sequential	same time
origin	receiver	sender (root) ➠ UNI4.0
change reservations	yes	no
routing changes	time-out	re-establish VC
flow merging (audio)	yes	no (separate VCs)
receiver diversity	not yet	no

Real-Time Transport Protocol

RTP

- protocol goals
- mixers and translators
- control: awareness, QOS feedback
- media adaptation

RTP – the big picture



RTP = Real-time transport protocol

- only part of puzzle: reservations, OS, ...
- product of Internet Engineering Task Force, AVT WG
- RFC 1889, 1890
- ITU H.323, RTSP, ...
- support for functions, but does not restrict implementation
- compression for low-bandwidth networks under study

RTP goals

lightweight: specification and implementation

flexible: provide mechanism, don't dictate algorithms

protocol-neutral: UDP/IP, ST-II, IPX, ATM-AALx, ...

scalable: unicast, multicast from 2 to $O(1000)$

separate control/data: some functions may be taken over by conference control protocol

secure: support for encryption, possibly authentication

RTP functions

- segmentation/reassembly done by UDP (or similar)
- resequencing (if needed)
- loss detection for quality estimation, recovery
- intra-media synchronization: remove delay jitter through playout buffer
- intra-media synchronization: drifting sampling clocks
- inter-media synchronization (lip sync between audio and video)
- quality-of-service feedback and rate adaptation
- source identification

RTP mixers, translators, ...

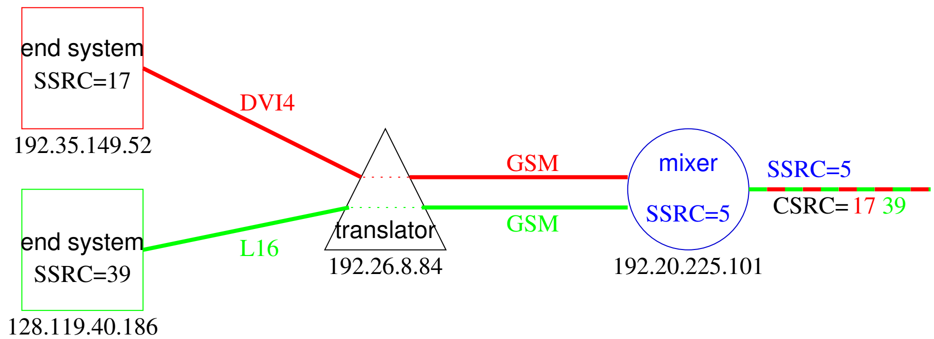
mixer:

- several media stream ➠ one new stream (new encoding)
- mixer: reduced bandwidth networks (dial-up)
- appears as new source, with own identifier

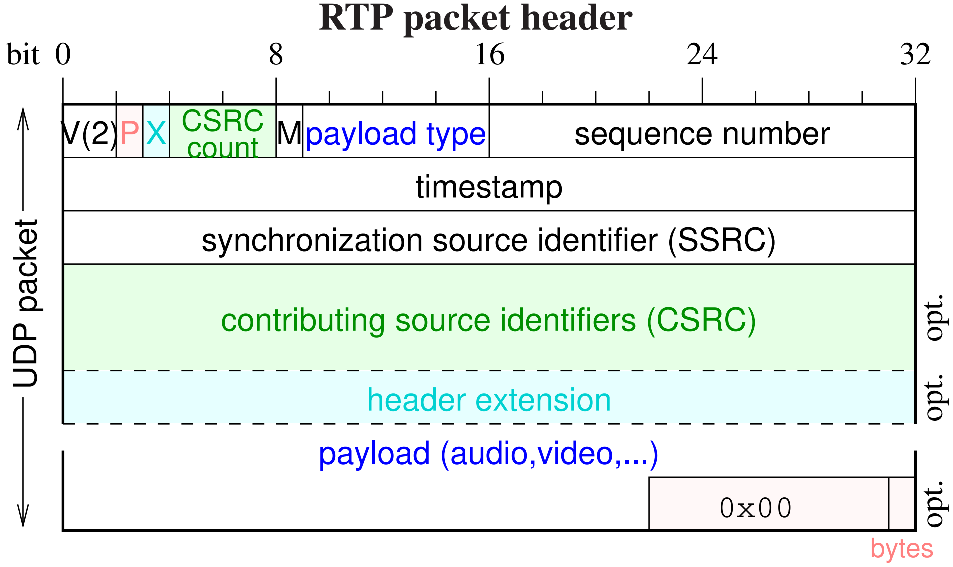
translator:

- single media stream
- *may* convert encoding
- protocol translation (native ATM $\leftrightarrow$ IP), firewall
- all packets: source address = translator address

RTP mixers, translators, . . .



RTP packet header



RTP packet header

Payload type: audio, video encoding method; may change during session

SSRC: sychronization source ➠ each source picks at random ➠ may change after *collision*!

sequence number: incremented by 1 for each packet ➠ gaps ≡ loss

P: padding (for encryption) ➠ last byte contains padding count

M: marker bit; indicates frame, beginning of talkspurt ➠ allow delay adjustment

CC: content source count (for mixers)

CSRC: list of identifiers of those contributing to (mixed into) packet

RTP timestamp

- incremented by 1 for each sample (e.g., 160 for 20 ms packets @ 8000 Hz)
- random starting value
- constant rate for each audio payload type (e.g., 8000 Hz for PCM μ-law, 44100 Hz for linear, 16-bit)
- 90 kHz for video
- several video frames may have same timestamp
- ➠ gaps ≡ silence
- time per packet may vary
- video frame maybe split (carefully…) over several packets
- typical: 20 to 100 ms of audio

RTP in a network

- typical: UDP, no fixed port; RTCP port = RTP port (even) + 1
- typical UDP size limited to few hundred bytes (OS, network, fragmentation)
- native ATM: directly into AAL5 frame
- encapsulation (length field) for others
- typically: one media (audio, video, ...) per port pair
- exception: bundled MPEG

RTP control protocol – types

stackable packets, similar to data packets

sender report (SR): bytes send ➠ estimate rate;
timestamp ➠ synchronization

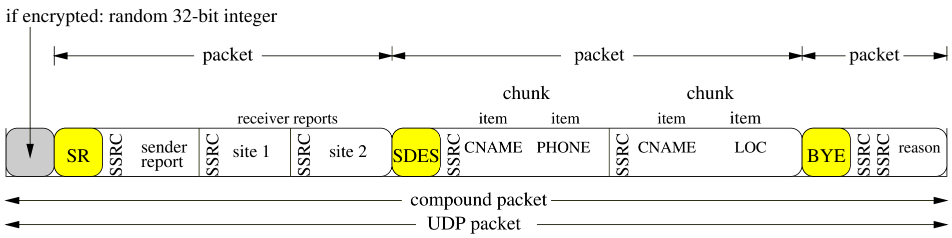
reception reports (RR): number of packets sent and expected ➠ loss, interarrival jitter, round-trip delay

source description (SDES): name, email, location, ...
CNAME (canonical name = user@host) identifies user across media

explicit leave (BYE): in addition to time-out

extensions (APP): application-specific (none yet)

RTCP packet structure



RTCP announcement interval computation

Goals:

- estimate current number & identities of participants – dynamic
- source description ("SDES") ➠ who's talking?
- quality-of-service feedback ➠ adjust sender rate
- scale to $O(1000)$ participants, small fraction of data bandwidth

➠ randomized response with rate $\downarrow$ as members $\uparrow$

- group size limited by tolerable age of status
- gives active senders more bandwidth
- soft state: if not heard from for multiple of announcement interval, delete

RTCP bandwidth scaling

- every participant: periodically multicast RTCP packet to same group as data
- ➠ everybody knows (eventually) who's out there
- session bandwidth:
  - single audio stream
  - $\sum$ of concurrently active video streams
- sender period T:

$$T = \frac{\text{\# of senders}}{0.25 \cdot 0.05 \cdot \text{session bw}} \cdot \text{avg. RTCP packet size}$$

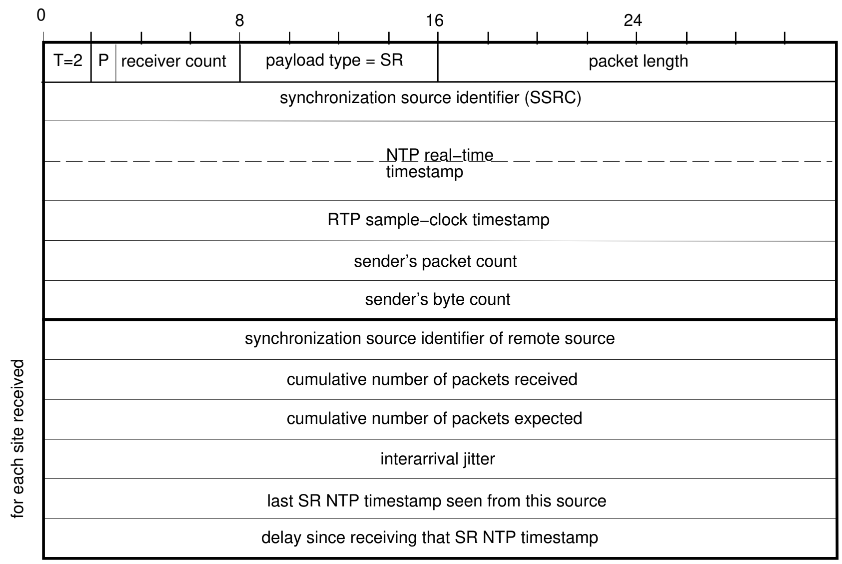
- receivers:

$$T = \frac{\text{\# of receivers}}{0.75 \cdot 0.05 \cdot \text{session bw}} \cdot \text{avg. RTCP packet size}$$

RTCP bandwidth scaling

- next packet = last packet + min(5 s, T) $\cdot$ random(0.5…1.5)
- randomization prevents "bunching"
- to reduce RTCP bandwidth, alternate between SDES components

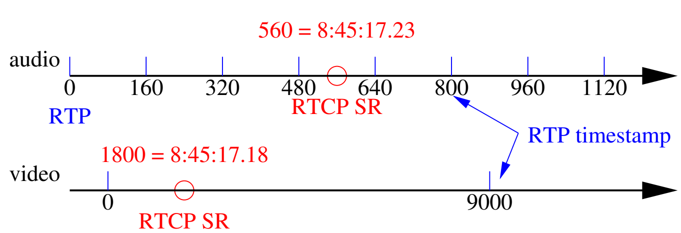
RTCP sender reports (SR)



Intermedia synchronization

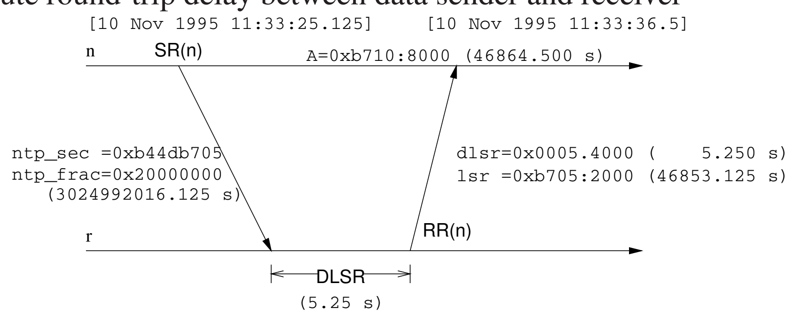
= synchronization between different streams (audio, video, slides, . . .)

- timestamps are offset with random intervals
- may not tick at nominal rate
- every SR correlates "real" time (wallclock time) with RTP timestamp
- ➠ compute when sample was generated



Round-trip delay estimation

compute round-trip delay between data sender and receiver




```
A      0xb710:8000 (46864.500 s)
DLSR  -0x0005:4000 (    5.250 s)
LSR   -0xb705:2000 (46853.125 s)
-------------------------------
delay 0x   6:2000 (    6.125 s)
```

Collision detection and resolution

Collision:

- two sources may pick the same SSRC (“birthday problem”)
- probability: about 10^{-4} if 1000 session members join more or less simultaneously
- but: don’t pick one you know about already ➠ probability much lower unless everyone joins at the same time
- send BYE for old, pick a new identifier

Loops:

- forward packet to same multicast group (directly or through translators)
- looks similar to collision, but changing SSRC doesn’t help

RTP for the masses

Problems using RTP for radio, TV:

- for 14.4 RealAudio: 90 bytes/second $\approx$ one new site per second
- takes $\approx$ 3 hours to get to know 10,000 people ➠
  - who cares? (Nielsen!)
  - useless for QOS feedback
  - control rate too high
- ➠ faster convergence: everybody reports estimate, compute max()
- ➠ statistical sample (sender determines rate): send value $[0, 1]$; pick random value; if $<$, lucky winner ➠ needs to be adaptive
- ➠ report just to sender, instead of multicast

Further reading

- D. E. Comer, *Internetworking with TCP/IP*, vol. 1. Englewood Cliffs, New Jersey: Prentice Hall, 3rd ed., 1995.
- D. E. Comer and D. L. Stevens, *Internetworking with TCP/IP*, vol. 2. Englewood Cliffs, New Jersey: Prentice Hall, 1991.
- D. C. Lynch and M. T. Rose, *Internet system handbook*. Reading, Massachusetts: Addison-Wesley, 1993.
- C. Partridge, *Gigabit networking*. Reading, Massachusetts: Addison-Wesley, 1993.
- W. R. Stevens, *TCP/IP Illustrated*, vol. 1. Reading, Massachusetts: Addison-Wesley, 1994.
- A. S. Tanenbaum, *Computer Networks*. Upper Saddle River, New Jersey: Prentice-Hall, 3rd ed., 1996.
- S. A. Thomas, *IPng and the TCP/IP Protocols*. New York: Wiley, 1996.