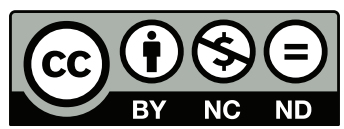
Advanced Networking '2012

Jürgen Schönwälder

JACOBS
UNIVERSITY

November 15, 2012



http://cnds.eecs.jacobs-university.de/courses/an-2012/

Part: Preface

Course Overview

Course Content

- Internet Multicasting (IGMP, DVMRP, MOSPF, PIM)
- New Internet Transport Protocols (SCTP, DCCP)
- Internet Quality of Service (IntServ, RSVP, DiffServ)
- Multimedia Transport and Signaling (RTP, RTCP, SIP)
- Voice over IP (codecs, metrics)
- Mobility (MIPv4, MIPv6, Link-Layer Handovers)
- Highspeed TCP Extensions (ECN, TCP improvements)
- Domain Name System Extensions (SRV, ENUM, DDDS)
- Label Switching (MPLS, LDP, RSVP-TE, GMPLS)
- Measurement, Modeling, Simulation (NETFLOW, NS-2)

Course Objective

- Introduce advanced networking concepts
- Combine theory with practical experiences
  - $\Rightarrow$ Advanced Networking Lab
- Prepare students for research in the networking field
  - $\Rightarrow$ Learn to read and review papers
  - $\Rightarrow$ Learn to present and evaluate ideas

Prerequisites

- Protocol Layering (ISO/OSI, Internet)
- Names, Addresses, Services, Protocols
- Transmission Media and their Properties
- Data Encoding (NRZ, RZ, Manchester)
- Media Access (TDM, FDM, CSMA-CD, MACA)
- Error Detection (Checksums, CRC)
- Sequence Numbers, Acknowledgements, Timer
- Flow Control and Congestion
- Local Area Networks (IEEE 802.3, 802.11, 802.1D)

Prerequisites (cont.)

- Internet Network Protocols (IPv4, IPv6)
- Internet Forwarding and Routing (RIP, OSPF, BGP)
- Internet Transports (UDP, TCP)
- Internet Application Protocols
  - Notations (ASN.1, ABNF)
  - Domain Name System (DNS)
  - Email (SMTP, IMAP)
  - Document Transfer (HTTP, FTP)
  - Network Management/Monitoring (SNMP)
- Remote Procedure Calls (Stubs, Semantics)
- Socket Programming in C

Reasons for not taking this course

- You do not have the time required for this course
- You do not have the required background
- You expected an introductionary course
- You find the topics covered by this course boring
- You are unable to do some programming in C/Unix
- You are not ready to take initiative
  - Reading research papers and specifications
  - Programming tasks
  - Software setup and installation

Reading Material

Background Reading Material

- A.S. Tanenbaum, "Computer Networks", 4th Edition, Prentice Hall, 2002
- W. Stallings, "Data and Computer Communications", 6th Edition, Prentice Hall, 2000
- C. Huitema, "Routing in the Internet", 2nd Edition, Prentice Hall, 1999
- D. Comer, "Internetworking with TCP/IP Volume 1: Principles Protocols, and Architecture", 4th Edition, Prentice Hall, 2000
- J.F. Kurose, K.W. Ross, "Computer Networking: A Top-Down Approach Featuring the Internet", 3rd Edition, Addison-Wesley 2004.

Important Journals

- ACM/IEEE Transactions on Networking
- ACM Computer Communications Review (SIGCOMM)
- ACM Mobile Communications Review (MOBICOM)
- IEEE Journal on Selected Areas of Communications
- IEEE Transactions on Communications
- IEEE Transactions on Wireless Communications
- IEEE Transactions on Mobile Computing
- IEEE Transactions on Network and Service Management
- IEEE Communications Magazine
- IEEE Surveys and Tutorials
- Elsevier Computer Networks
- Elsevier Computer Communications
- Elsevier Ad Hoc Networks
- ...

Important Conferences and Workshops

- ACM SIGCOMM
- IEEE INFOCOM
- IEEE GLOBECOM
- IEEE/IFIP IM / NOMS / DSOM ...
- IEEE ICC
- ...

- It is essential to know where to submit a paper; not all events have the same quality.
- Kevin C. Almeroth maintains a nice web page with conference statistics:
  `http://www.cs.ucsb.edu/~almeroth/conf/stats/`

Network Research Challenges

- Some network research areas:
  - Routing and convergence
  - Security, trust and key management
  - Network management and network operations
  - Ad-hoc networks and self-organizing networks
  - Scalable inter-domain quality of service (QoS)
  - Measurement and modeling
  - Killer applications
  - Disappearing (invisible) networks
  - Interplanetary Internet

$\Longrightarrow$ See RFC 3869 for further information on Internet Research.

Grading Scheme

Grading Scheme

- Homeworks (30%)
  - Individual submission of solutions
  - Learning by solving assignments
- Quizzes (30%)
  - Control your continued learning success
- Final examination (40%)
  - Oral exam (maybe written if too many students)
  - Covers the whole lecture

References I

A. S. Tanenbaum.

Computer Networks.
Prentice Hall, 4 edition, 2002.

W. Stallings.

Data and Computer Communications.
Prentice Hall, 7 edition, 2004.

C. Huitema.

Routing in the Internet.
Prentice Hall, 2 edition, 1999.

D. E. Comer.

Internetworking with TCP/IP: Principles, Protocols, and Architectures.
Prentice Hall, 4 edition, 2000.

J. F. Kurose and K. W. Ross.

Computer Networking: A Top-Down Approach Featuring the Internet.
Addison-Wesley, 3 edition, 2004.

R. Atkinson and S. Floyd.

IAB Concerns and Recommendations Regarding Internet Research and Evolution.
RFC 3869, Internet Architecture Board, August 2004.

Part: Internet Multicasting

Multicast Terminology

Terminology

- *Unicast*: Communication between a single sender and a single receiver (1:1).
- *Multicast*: Communication between a single sender and multiple receivers (1:n).
- *Concast*: Communication between multiple senders and a single receiver (m:1).
- *Multipeer*: Communication between multiple senders and multiple receivers (m:n).

- *Anycast*: Communication between a single sender and one selected receiver out of a group of receivers.
- *Broadcast*: Communication between a single sender and all receivers attached to a network segment.

Multicast Groups

- *Open* vs. *closed* multicast groups:
  Member of open groups accept messages from arbitrary senders while member of closed groups only accept messages from the group.
- *Dynamic* vs. *static*:
  Membership of static groups does not change during communication while membership of dynamic groups may change over time.
- *Transient* vs. *permanent*:
  Permanent groups exist permanently, even if the group has no members while transient groups only exist for a limited period of time.

Reliable Multicasts

- *Unreliable*:
  No guarantee that data is delivered to multicast group members.
- *Reliable*:
  Guarantee that data is delivered to all multicast group members.
- *k-Reliable*:
  Data is reliably delivered to at least k group members.
- *p-Reliable*:
  Data is reliably delivered to a certain percentage p of the group members.

MBONE: Multicast Backbone

- International project to establish a global multicast backbone
- Multicast is natively supported in the German research network (but currently not at Jacobs)
- Early experiements with audio and video conferencing (vat, rat, vic), shared whiteboards (wbd), shared editors, session directory (sdr), ...
- Selected IETF meetings were broadcasted over the MBONE for several years
- Not widely used outside of (multicast) research environments

Multicast Challenges

- Multicast flow and congestion control
  - Positive (ACK) vs. negative (NACK) acknowledgements
  - Maintenance of the sender's buffer
  - Rate-based instead of window-based flow control
  - Feedback implosion problems
- Multicast routing
- Multicast reliability
- Multicast security
- ...

Multicast Addresses

Group Addresses

- Group members can be identified by explicit member lists or group addresses.
- Group addresses may be assigned from a central authority or dynamically in a decentralized fashion.
- Group address assignments may be permanent or transient.
- Decentralized approaches usually require the introduction of multicast group address management server for coordination.
- Transient group addresses may be announced in a shared directory.

Internet Multicast Addresses

- IPv4 (RFC 1112):

```
| 4  |          28 bits             |
+---------------------------------+
|1110|         group ID             |
+----+------------------------------+
```

  - 224.0.0.1 (ip4-allnodes), 224.0.0.2 (ip4-allrouters)

- IPv6 (RFC 2375, RFC 3307):

```
|   8    |  4 |  4 |                    112 bits                    |
+------ -+----+----+------------------------------------------------+
|11111111|flgs|scop|                    group ID                    |
+--------+----+----+------------------------------------------------+
```

  - Permanent link-local multicast addresses:
    ff02::1 (ip6-allnodes), ff02::2 (ip6-allrouters)

Ethernet Mapping (IPv4)

- IEEE 802 MAC addresses support multicasts (lower two bits in the first octet indicate multicast)
- IANA owns a block of Ethernet MAC addresses that start with the 23-bit prefix 01:00:5E.
- Half of this address block are multicast addresses (which means there is a fixed 25-bit prefix).
- The remaining 23 bits are filled by mapping the lower 23 bits of the IP multicast address into the MAC multicast address.
- As a consequence, $2^5 = 32$ IPv4 group addresses map to the same MAC address.

Ethernet Mapping (IPv6)

- IANA owns a block of Ethernet MAC addresses that start with the 16-bit prefix 33:33.
- The lower 32 bits of the MAC address are filled by copying the lower 4 bytes of the IPv6 address into the MAC address.
- As a consequence, 2^{80} IPv6 group addresses map to the same MAC address.
- The question, of course, is how frequent collisions are in practice under normal conditions.
- To quote RFC 2464:

> *"There is no protection from duplication through accident or forgery."*

Internet Multicast Services (RFC 3569)

- *Any-Source Multicast* (ASM):
  - IP datagrams are transmitted to a group G of nodes identified by a single IP multicast address.
  - Nodes may join and leave the group G any time, and there is no restriction on their location or number.
- *Source-Specific Multicast* (SSM):
  - IP datagrams are transmitted by a source S to an SSM destination address G, and receivers can receive this datagram by subscribing to channel (S, G).
- *Source-Filtered Multicast* (SFM):
  - ASM variant with filtered source addresses.
  - Supports whitelists (only a specific set of sources) and blacklists (all except a specific set of sources).

Multicast Socket API Extensions

IPv4 Multicast Socket Extensions

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define IP_MULTICAST_IF     ...
#define IP_MULTICAST_TTL    ...
#define IP_MULTICAST_LOOP   ...

#define IP_ADD_MEMBERSHIP   ...
#define IP_DROP_MEMBERSHIP  ...

struct ip_mreq {
    struct in_addr imr_multiaddr;  /* IP address of group */
    struct in_addr imr_interface;  /* IP address of interface */
};
```

IPv6 Multicast Socket Extensions

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define IPV6_MULTICAST_IF   ...
#define IPV6_MULTICAST_HOPS ...
#define IPV6_MULTICAST_LOOP ...

#define IPV6_JOIN_GROUP     ...
#define IPV6_LEAVE_GROUP    ...

struct ipv6_mreq {
    struct in6_addr ipv6mr_multiaddr; /* IP address of group */
    unsigned int ipv6mr_interface;    /* interface number */
};
```

IP Any-Source Multicast API (RFC3678)

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define IP_BLOCK_SOURCE     ...
#define IP_UNBLOCK_SOURCE   ...

struct ip_mreq_source {
    struct in_addr imr_multiaddr;  /* IP address of group */
    struct in_addr imr_sourceaddr; /* IP address of source */
    struct in_addr imr_interface;  /* IP address of interface */
};
```

IP Source-Specific Multicast API (RFC3678)

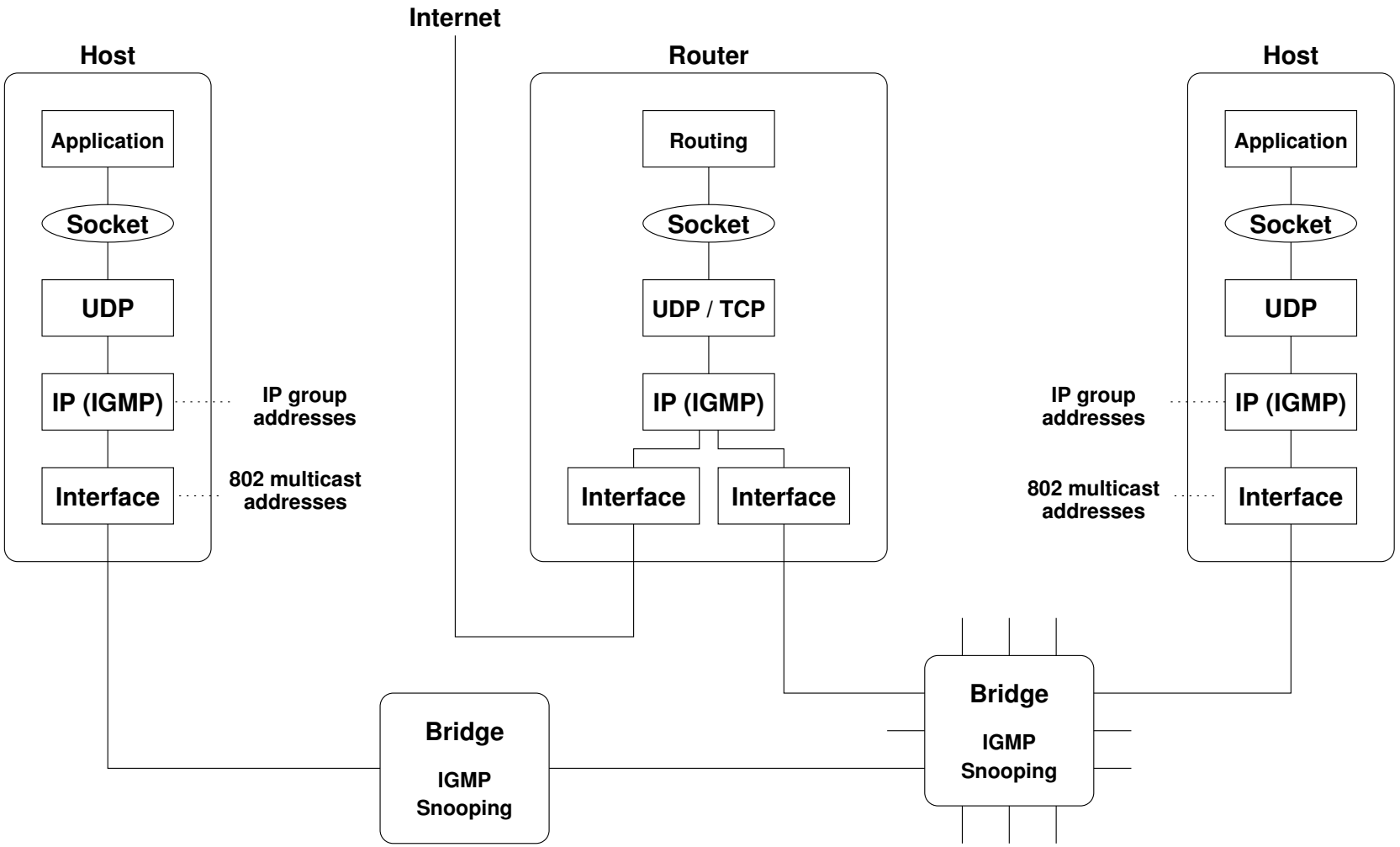
```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define IP_ADD_SOURCE_MEMBERSHIP    ...
#define IP_DROP_SOURCE_MEMBERSHIP   ...

struct ip_mreq_source {
    struct in_addr imr_multiaddr;  /* IP address of group */
    struct in_addr imr_sourceaddr; /* IP address of source */
    struct in_addr imr_interface;  /* IP address of interface */
};
```

Internet Group Management Protocol (IPV4)

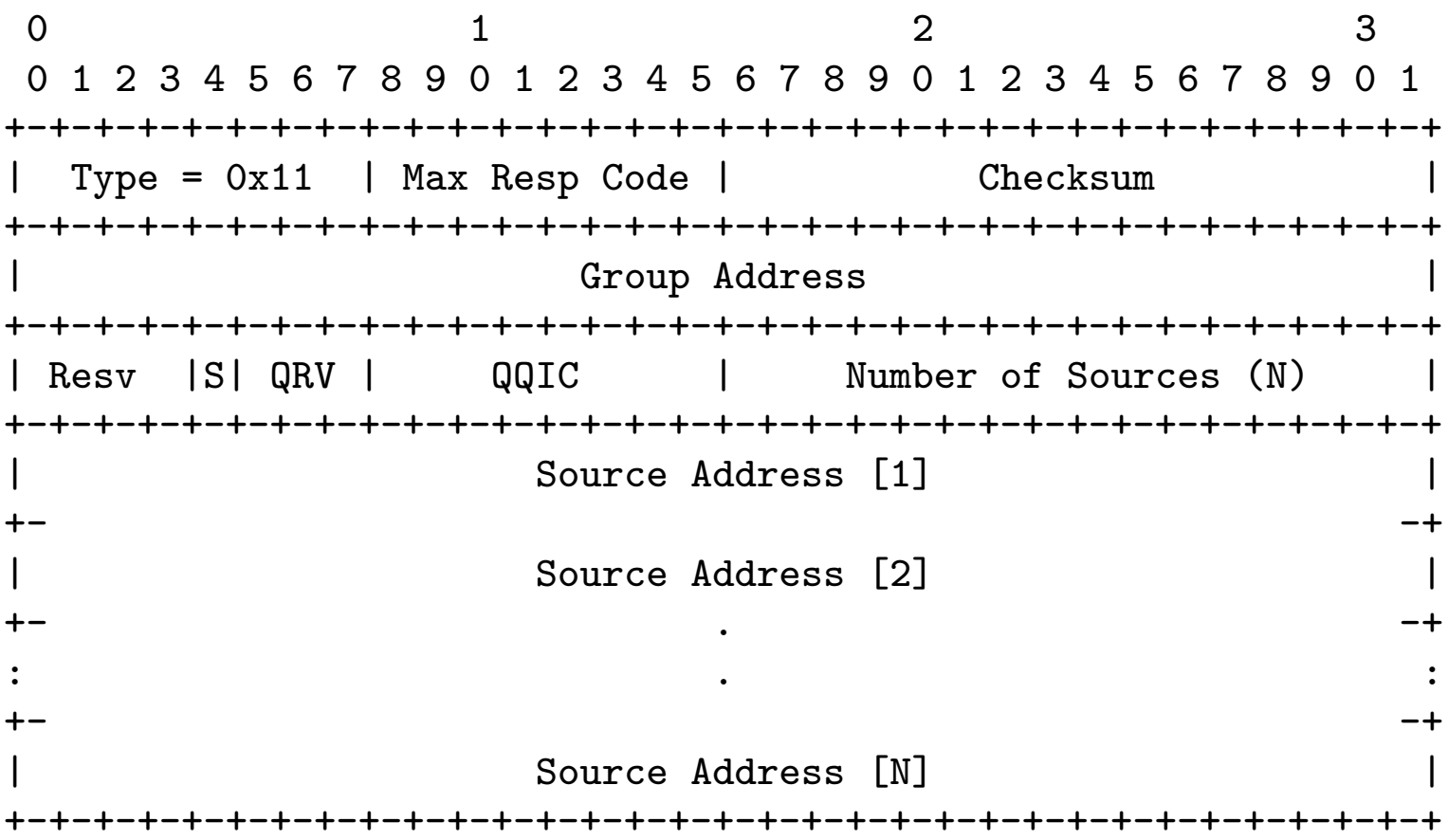
IP Multicast Scenario (LAN)



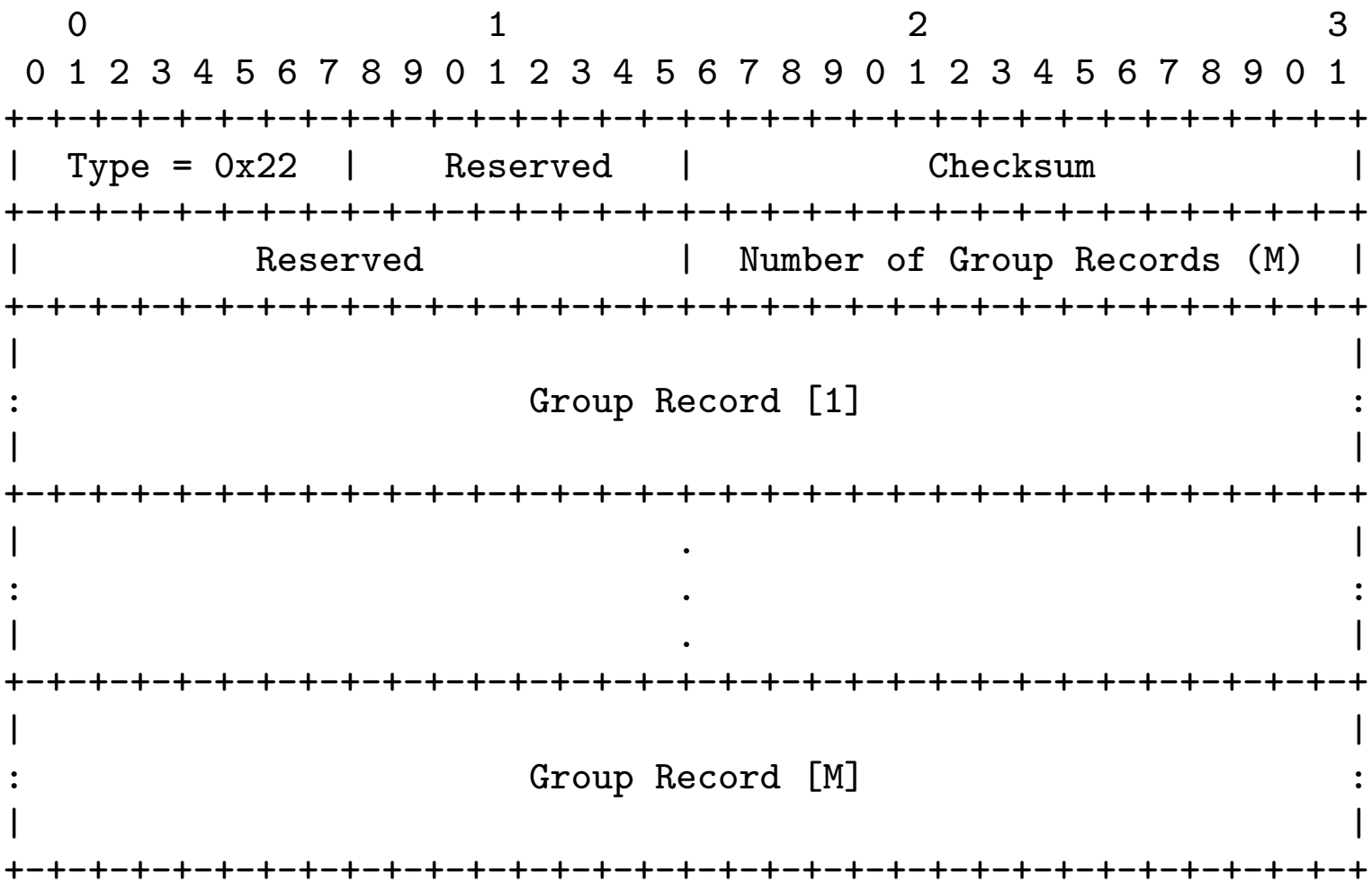
Internet Group Management Protocol

- IGMP version 3 (IGMPv3) is published in RFC 3376 and RFC4604.
- IPv4 nodes use IGMP to report their group membership to the neighboring multicast router.
- A multicast router periodically sends a query message to 224.0.0.1 to check the group membership state.
- Group members respond with a membership report message.
- Multicast router maintain a per interface multicast group list.
- Group members can send unsolicited membership reports when they join/leave a group.

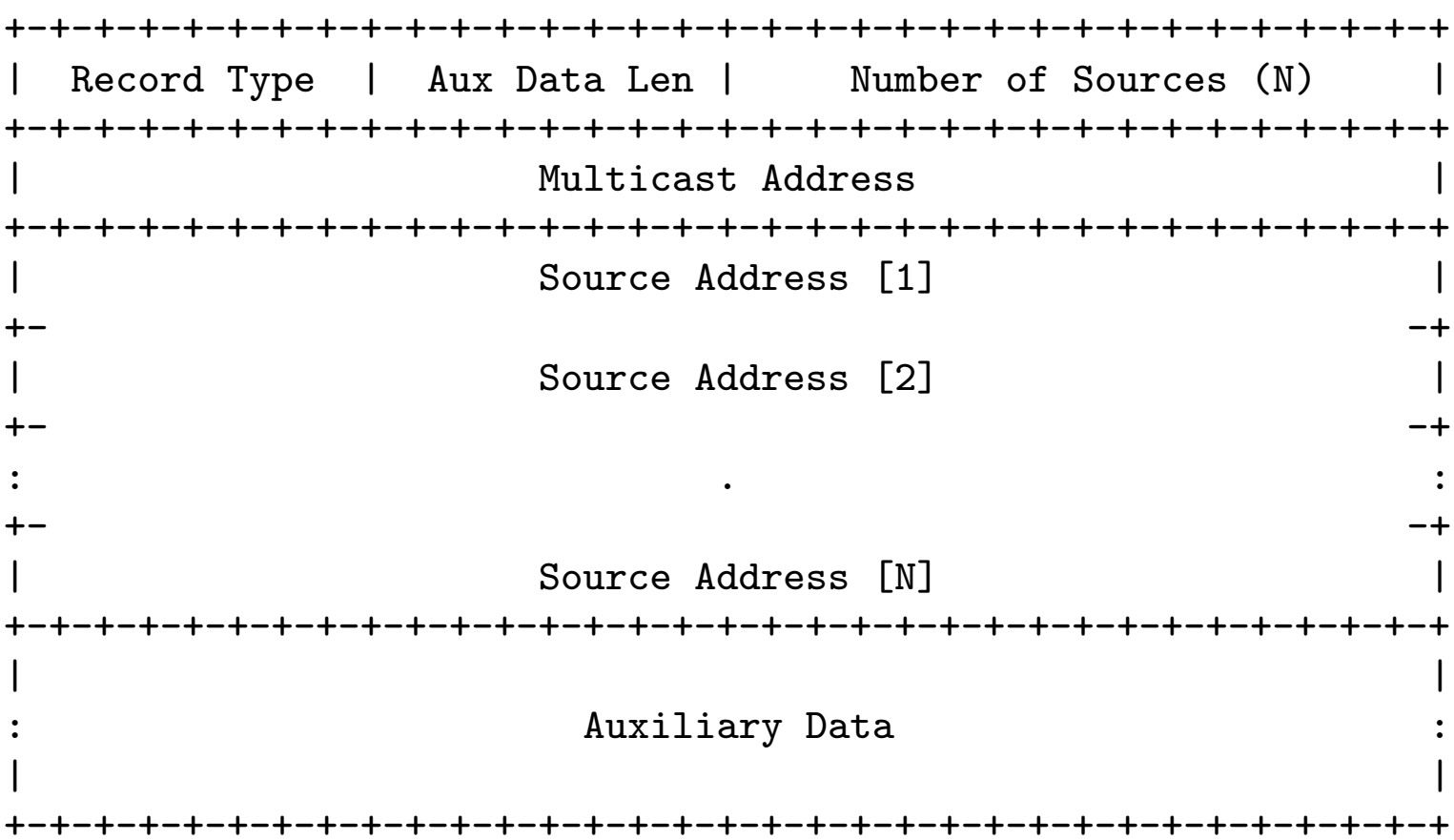
IGMPv3 Query Message



IGMPv3 Report Message



IGMPv3 Group Record



IGMPv3 Query Variants

- A “General Query” is sent by a multicast router to learn the complete multicast reception state of the neighboring interfaces.
- A “Group-Specific Query” is sent by a multicast router to learn the reception state, with respect to a *single* multicast address, of the neighboring interfaces.
- A “Group-and-Source-Specific Query” is sent by a multicast router to learn if any neighboring interface desires reception of packets sent to a specified multicast address, from any of a specified list of sources.

IGMP Snooping

- IEEE 802 bridges snoop IGMP packets to learn which port belongs to which IPv4 multicast group.
- Allows to suppress multicast traffic on segments / ports without group members.
- Widely supported by IEEE 802 bridges today.

Multicast Listener Discovery Protocol (IPV6)

Multicast Listener Discovery (IPv6)

- MLD version 2 (MLDv2) is published in RFC 3810 and RFC 4604.
- Translation of the IGMPv3 protocol for IPv6 semantics.
- Not further discussed here.

Multicast Routing Algorithms

Multicast Routing

- Challenges:
  - Route data only to group members
  - Optimize routes from the source to the receivers
  - Maintain loop-free routes
  - Distribute multicast traffic over multiple links
  - Signalling (group membership) must scale well
- Several solutions have been tried...

Flooding and Spanning Trees

- Flooding
  - + Conceptually very simple
  - + Robust (all possible paths are explored)
  - - Requires to maintain history of last seen packets which can be memory intensive on high speed networks
- Spanning Trees:
  - + Robust and well understood technique
  - + Does not require much memory
  - - Group membership is not taken into account
  - - Concentrates multicast traffic on a subset of the links

Reverse Path Forwarding (RPF)

- Principle:
  1. When a multicast packet is received, note the source S and the interface I
  2. If I belongs to the shortest path to S, forward to all interfaces except I
  3. Otherwise, discard the packet

\+ Uses unicast routing tables to derive distribution trees

\+ Shortest path from source to destination

\+ Packets from different sources may use different links

\- Group membership is not taken into account

RPF Optimization

- Problem:
  - With plain RPF as described above, routers may receive packets via multiple paths
- Optimization:
  - Router determine whether they are on the shortest path between a neighbor and the multicast source before forwarding packets to a neighbor
  - The necessary information can be obtained from the link state database in OSPF

Flooding and Pruning

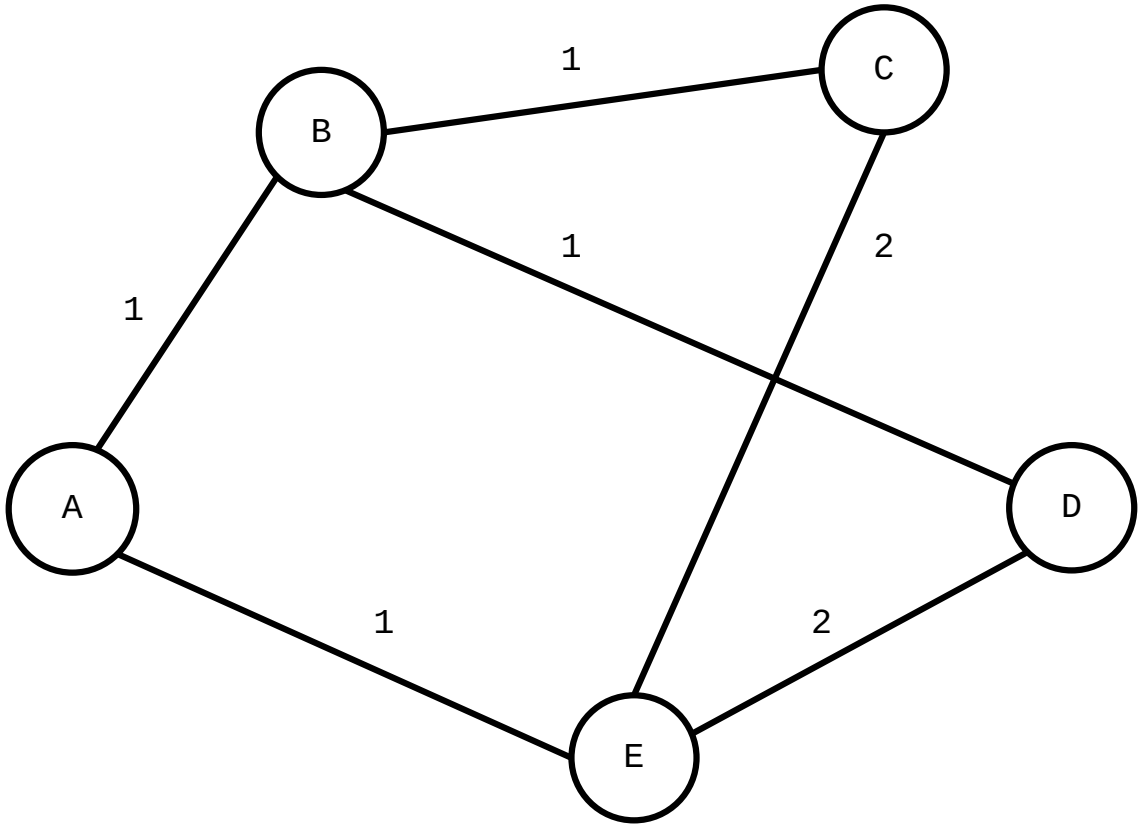
- Principle:
  1. Periodically, multicast packets are flooded
  2. Leaf routers who have no customers react by sending prune messages back towards the source
  3. Intermediate routers which do not have members on any interface will also send prune messages towards the source

\+ Computes minimal distribution trees

\- Requires periodic (global) flooding

\- Routers must keep state on a per-group and per-source basis

RPF Example



- Compute the reverse path forwarding (RPF) multicast trees for the sources A, B and D.

RPF Example: Unicast Routes

A	Dest.	Next
	B	B
	C	B
	D	B
	E	E

B	Dest.	Next
	A	A
	C	C
	D	D
	E	A

C	Dest.	Next
	A	B
	B	B
	D	B
	E	E

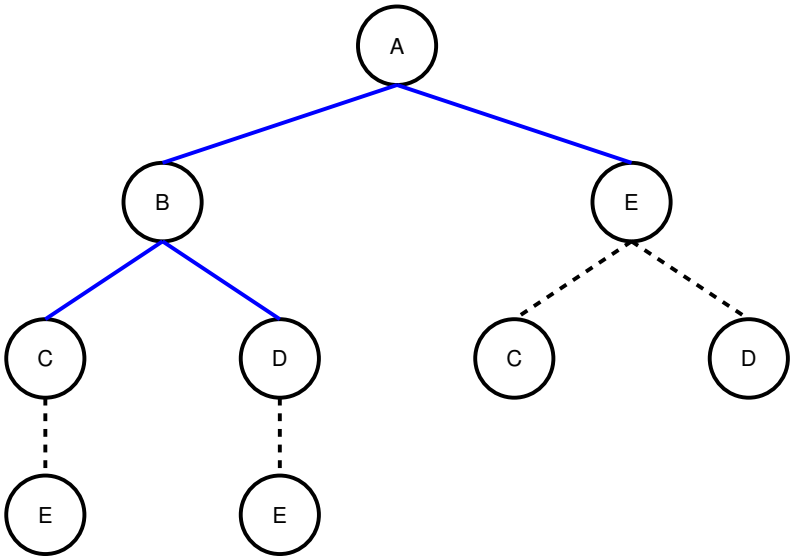
D	Dest.	Next
	A	B
	B	B
	C	B
	E	E

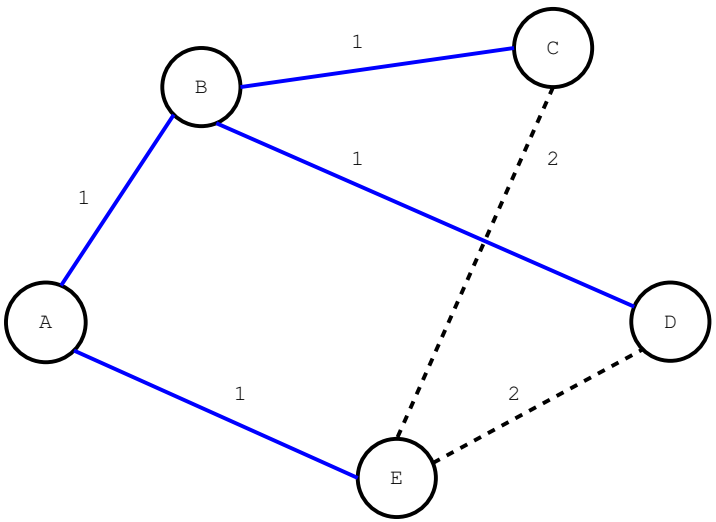
E	Dest.	Next
	A	A
	B	A
	C	C
	D	D

- Calculate unicast routes using Dijkstra or Bellman-Ford.

RPF Example: RPF Tree for A

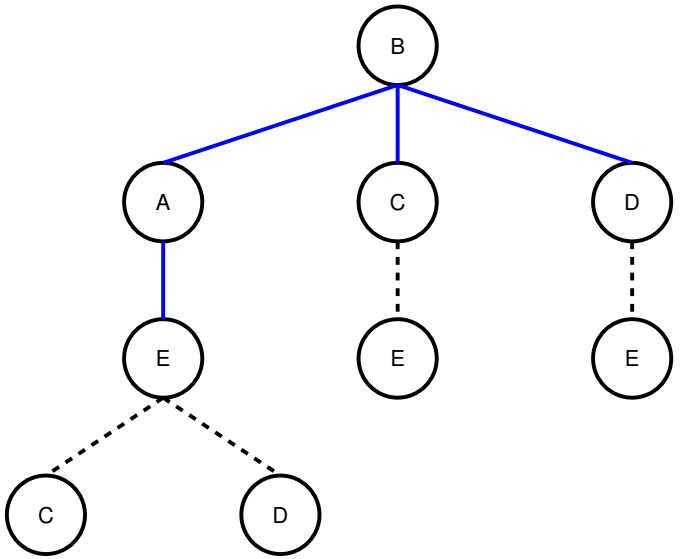
RPF tree for source A (dotted links may be pruned):

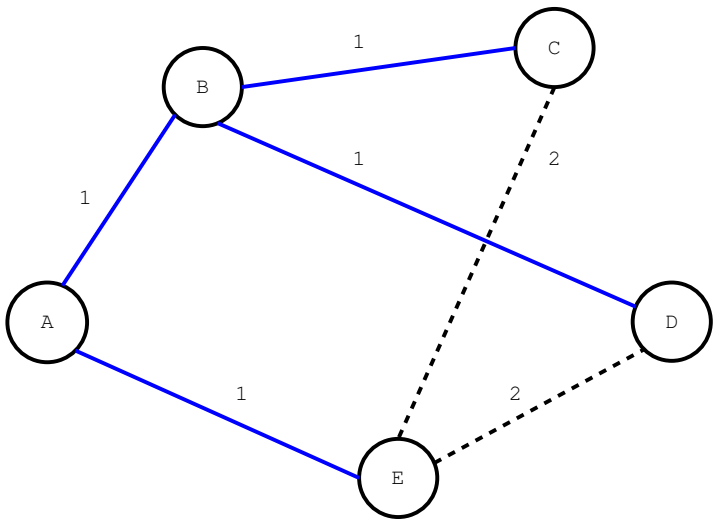




RPF Example: RPF Tree for B

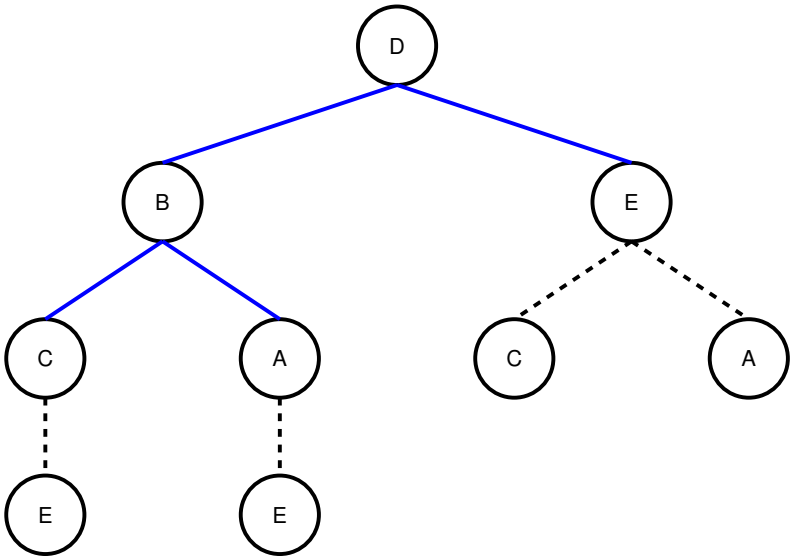
RPF tree for source B (dotted links may be pruned):

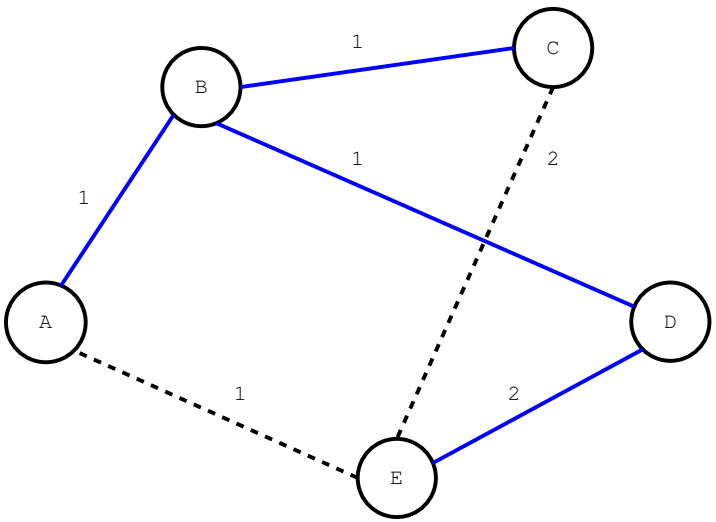




RPF Example: RPF Tree for D

RPF tree for source D (dotted links may be pruned):





Steiner Trees

- Steiner Trees:
  1. Given is a graph $G = (V, E)$ with a set of vertices V, a set of edges E and a cost function $c : E \mapsto \mathbb{N}$
  2. Let S be a subset of V. A steiner tree is a tree of G that spans S with a minimal total distance on its edges

+ Steiner trees minimize the number of links needed to connect the group members

- Steiner tree problem is known to be NP-hard
- Computation requires knowledge of the full topology
- Concentrates multicast traffic on a subset of the links
- Group membership changes lead to a recomputation and potentially massive changes of the distribution tree

Center-Based Trees

- Principle:
  1. Establish a well-known center for a multicast group
  2. Multicast receivers send join messages towards the center
  3. Intermediate routers keep state about the interfaces registered for a multicast group
- Center-based trees build a spanning tree for each group

+ Limits the expansion of multicast traffic to the set of all group members

o Routers must keep state on a per-group basis

- Choosing a center is difficult (optimal centers lead again to NP complete problems and instability)

Rendevous Points

- Instead of establishing a fixed center or core, it is possible to use rendevous points.
- The traffic originating from a new sender is encapsulated and unicast routed to a rendezvous point.
- Receivers send join messages towards the rendevous point, thereby establishing a path in the distribution tree.
- Distribution trees can be dynamically optimized by sending a source-specific join message towards the source (and subsequent pruning of the shared tree after establishing a shorter path).

Internet Multicast Routing Protocols

Internet Multicast Routing

- Available Multicast Routing Protocols:
  - Distance Vector Multicast Routing Protocol (DVMRP)
  - Multicast Extensions to OSPF (MOSPF)
  - Protocol-Independent Multicast, Dense Mode (dense PIM)
  - Protocol-Independent Multicast, Sparse Mode (sparse PIM)
  - Border Gateway Multicast Protocol (BGMP)
- Today's protocol of choice is usually PIM in sparse mode.

DVMRP (RFC 1075)

- Oldest deployed multicast routing protocol based on RPF
- DVMRP routers exchange distance vectors that contain lists of potential sources and distances
- Popular `mrouted` implementation supports tunnels and was used to establish the multicast backbone (mbone).
- DVMRP has scaling problems because of the necessity to flood frequently.
- Early implementations did not implement pruning which made periodic flooding even more expensive

MOSPF (RFC 1584)

- Multicast OSPF (MOSPF) is a multicast extension of the OSPF routing protocol.
- MOSPF essentially computes multicast distribution trees from an augmented link state database.
- Augmented link state advertisements propagate information about the multicast groups active on each network.
- MOSPF computes distribution trees for each (source,group) pair.
- Distribution trees are cached, but they must be recomputed whenever a link state changes.

PIM Dense Mode (RFC 3973)

- Dense mode PIM uses RPF and is similar to DVMRP (but can use any unicast routing protocol).
- Dense mode works well in the following situations:
  - Senders and receivers are in close proximity to one another.
  - There are few senders and many receivers.
  - The volume of multicast traffic is high.
  - The stream of multicast traffic is constant.
- Dense mode PIM maintains distribution trees for (sender,group) pairs.
- Graft messages can be used to turn a pruned branch back into a forwarding branch quickly.

PIM Sparse Mode (RFC 4601)

- Sparse mode PIM uses rendezvous points:
  - Senders initially send traffic to a rendevous point.
  - Receivers initially register with a rendevous point.
  - The initial forwarding path via a rendevous point can be optimized by the routers in the path.
- Sparse mode works well in the following situations:
  - There are few receivers in a group.
  - Senders and receivers are separated by WAN links.
  - The type of traffic is intermittent.

The bigger picture...

- Is the IP layer the right layer to provide multicast service?
- Application Layer Multicast (ALM)
  Group membership, multicast delivery structure construction, and data forwarding are solely controlled by participating end hosts (no support of intermediate nodes such as routers or proxies).
- Overlay Multicast (OM)
  Multicast service is supported by specially deployed intermediate proxies which cooperatively construct a "backbone overlay" infrastructure and establish multicast trees among themselves for data delivery.
- See the references for further discussion and comparison.

References I

L. Lao, J.-H. Cui, M. Gerla, and D. Maggiorini.

A Comparative Study of Multicast Protocols: Top, Bottom, or In the Middle?
In *Proc. 8th IEEE Global Internet Symposium (GI 2005)*, Miami, March 2005.

M. Ramalho.

Intra- and Inter-Domain Multicast Routing Protocols: A Survey and Taxonomy.
IEEE Communications Surveys and Tutorials, 3(1), 2000.

C. K. Miller.

Reliable Multicast Protocols and Applications.
The Internet Protocol Journal, 1(2):19–37, September 1998.

S. Bhattacharyya.

An Overview of Source-Specific Multicast (SSM).
RFC 3569, Spring, July 2003.

B. Cain, S. Deering, I Kouvelas, B. Fenner, and A Thyagarajan.

Internet Group Management Protocol, Version 3.
RFC 3376, Cereva Networks, Cisco Systems, AT&T Labs, Ericsson, October 2002.

R. Vida and L. Costa.

Multicast Listener Discovery Version 2 (MLDv2) for IPv6.
RFC 3810, LIP6, June 2004.

H. Holbrook, B. Cain, and B. Haberman.

Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast.
RFC 4604, Arastra, Acopia Networks, JHU APL, August 2006.

References II

D. Waitzman, C. Partridge, and S. Deering.

Distance Vector Multicast Routing Protocol.
RFC 1075, BBN STC, Stanford University, November 1988.

J. Moy.

Multicast Extensions to OSPF.
RFC 1584, Proteon, March 1994.

A. Adams, J. Nicholas, and W. Siadak.

Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised).
RFC 3973, NextHop Technologies, ITT A/CD, January 2005.

B. Fenner, M. Handley, H. Holbrook, and I Kouvelas.

Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised).
RFC 4601, AT&T Labs, UCL, Arastra, Cisco, August 2006.

D. Thaler.

Border Gateway Multicast Protocol (BGMP): Protocol Specification.
RFC 3913, Microsoft, September 2004.

D. Thaler, B. Fenner, and B. Quinn.

Socket Interface Extensions for Multicast Source Filters.
RFC 3678, Microsoft, AT&T, Research, Stardust.com, January 2004.

D. Eastlake.

IANA Considerations and IETF Protocol Usage for IEEE 802 Parameters.
RFC 5342, Eastlake Enterprises, September 2008.

Part: High-Speed TCP

Motivation

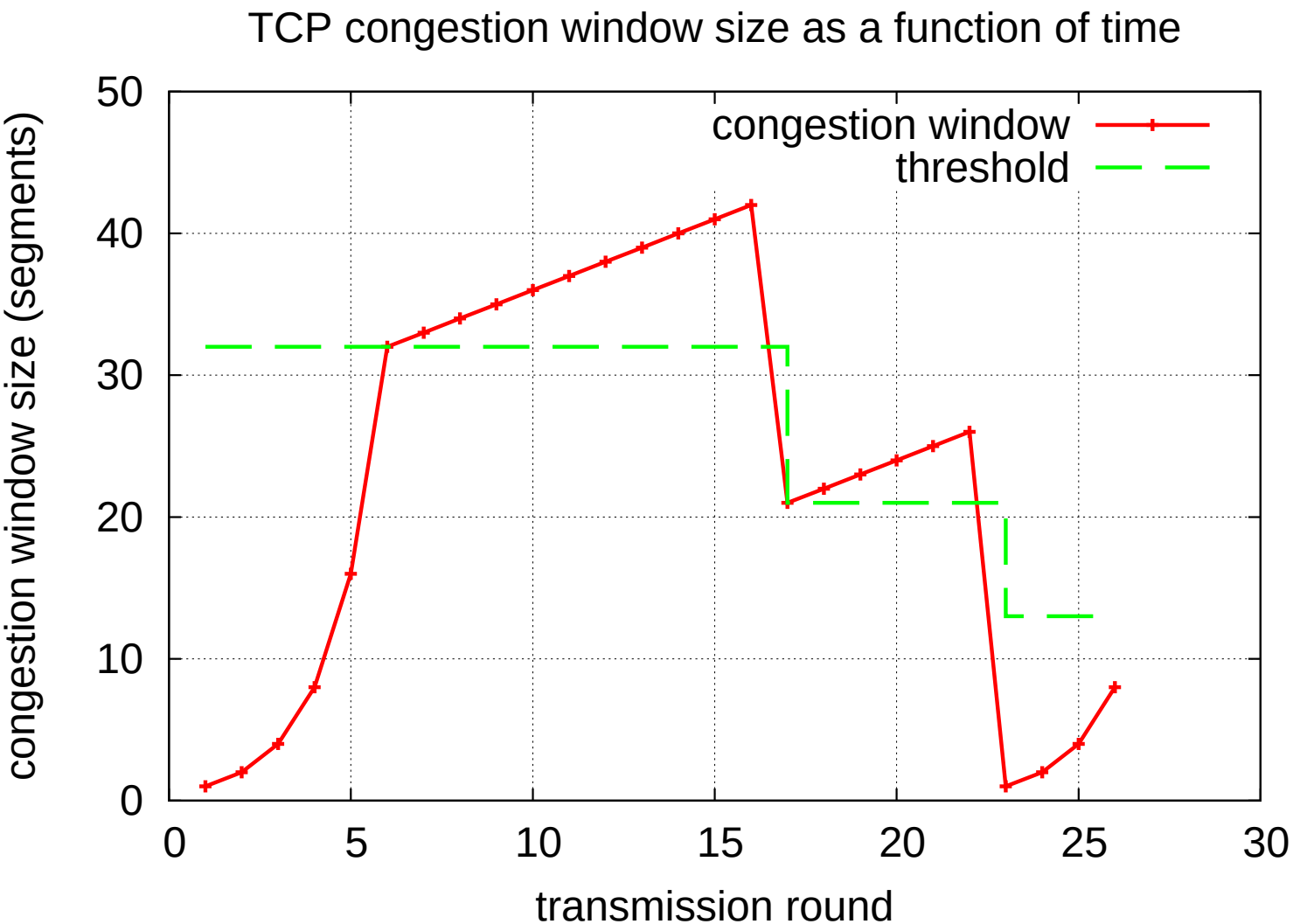
TCP on High-Speed Networks

- How well does TCP operate in gigabit / terrabit per second networks?
- What is the speed that can be realized today in real high speed networks spanning large distances?
- What changes are needed to work well over networks with large bandwidth-delay products?

Review of Traditional TCP Congestion Control

- Slow start mode:
  - Send two TCP segments in response to each ACK that advances the sender's window.
  - Exponential increase of the sending rate
- Congestion avoidance mode:
  - Send an additional segment of data for each loss-free round-trip time interval
  - Linear increase of the sending rate
- Threshold controls transition from slow start mode to congestion avoidance mode
- Timeout causes transition from congestion avoidance mode to slow start mode
- Multiple duplicate ACKs cause TCP to halve the sending rate and to enter congestion avoidance

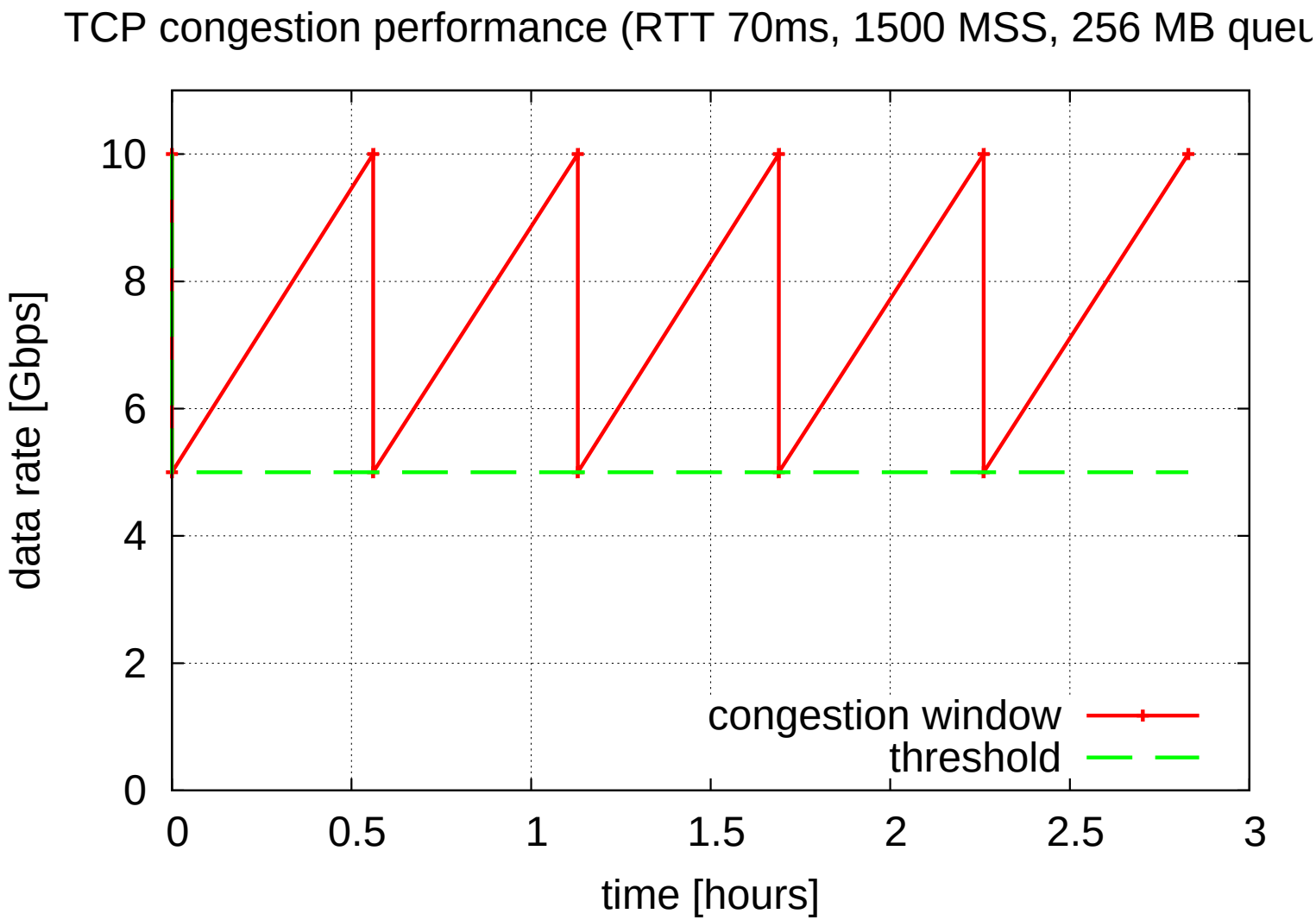
TCP Performance over Time



Example: TCP at High Speed

- 10 Gbps path, 1500 byte segments, 70ms delay
- In slow start, the speed doubles every 70ms, so after 17 round trip times, the speed exceeds 10 Gbps and this takes 1.2 seconds (assuming a very large initial threshold)
- Duplicate ACKs force congestion avoidance, which halves the congestion window and afterwards it increases linearly
- TCP congestion avoidance mode causes a sawtooth oscillation of this ideal TCP session between 5 Gbps and 10 Gbps; a single iteration of this cycle takes 34 minutes and 22 seconds
- This model assumes no packet loss due to bit errors and it implies massive data sets to be transferred

TCP Behavior at High Speed



TCP Land Speed Records

- http://lsr.internet2.edu/
- Ingredients:
  - It is essential to have the network path all to yourself
  - It is essential to have a fixed latency
  - It is essential to have an extremly low bit error rate
  - It is essential to know in advance the round-trip latency and the available bandwidth
- What can we do if we want TCP to work well without these ingredients?

High Speed Congestion Control Algorithms

Parallel TCP Streams

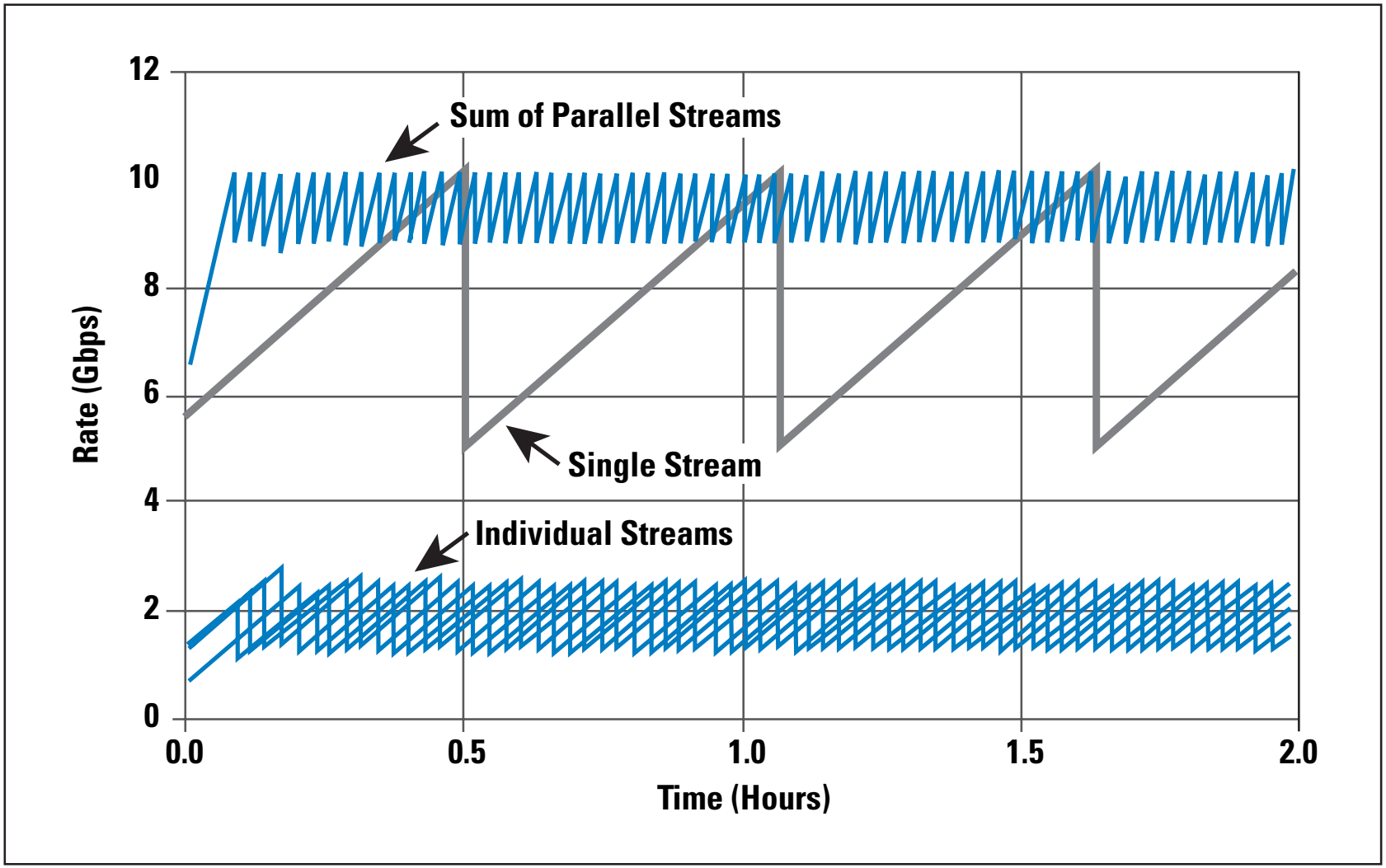
Idea

- Use multiple TCP streams in parallel
- Packet loss ideally only affects one or a few streams while the others continue probing

Properties

- Requires that data streams can be partitioned
- Packet loss can lead to synchonization of the streams
- No change of the kernel's TCP implementation required
- Examples: GridFTP, Bittorrent

Parallel TCP Streams Performance



MulTCP

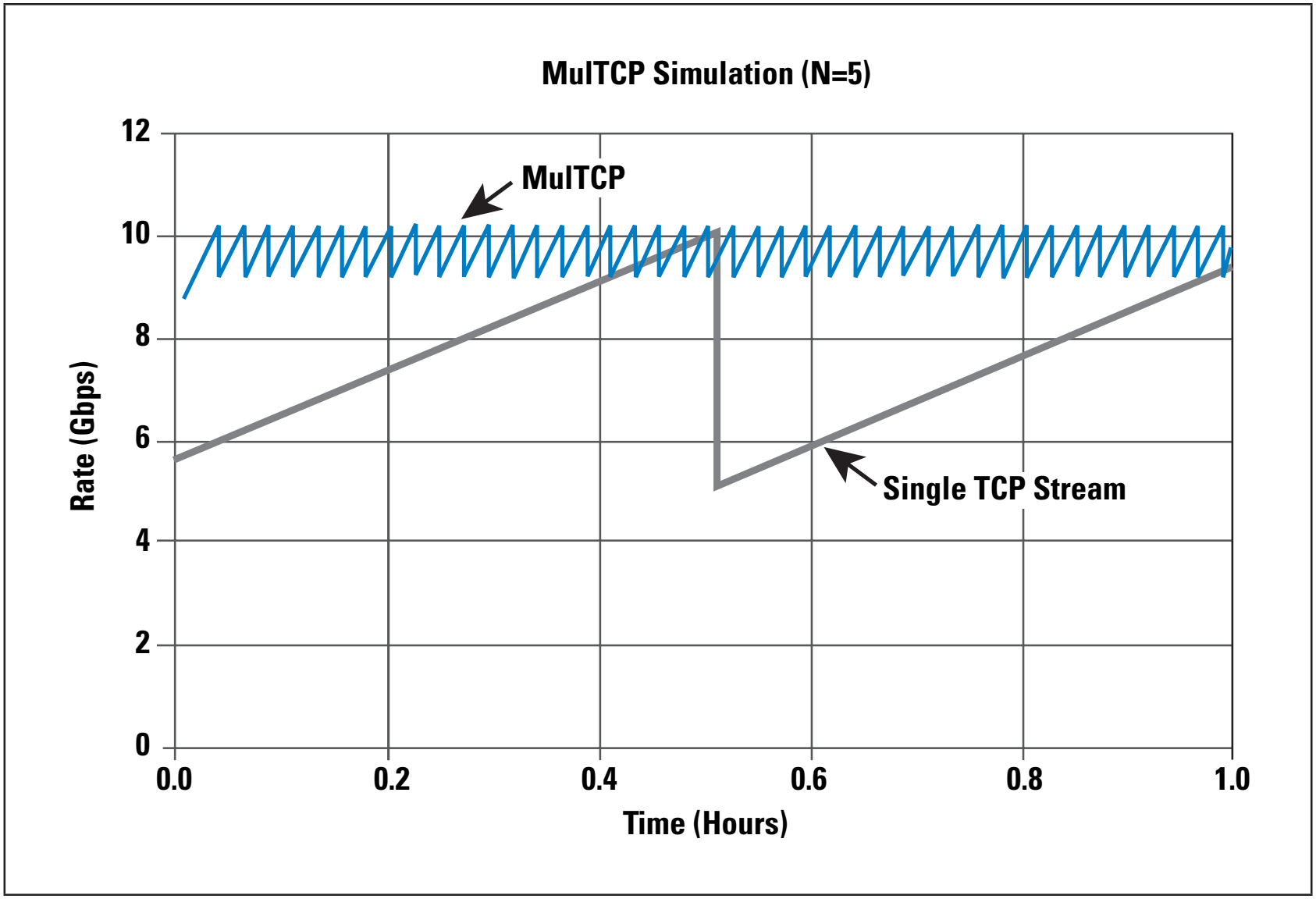
Idea

- Increase by N segments per RTT
- Reduce window W by $W/(2N)$ upon packet loss
- Emulates multiple parallel TCP streams

Properties

- Choosing N too high causes unfairness
- Choosing N too low means to not use the full network bandwidth

MulTCP Performance



HighSpeed TCP

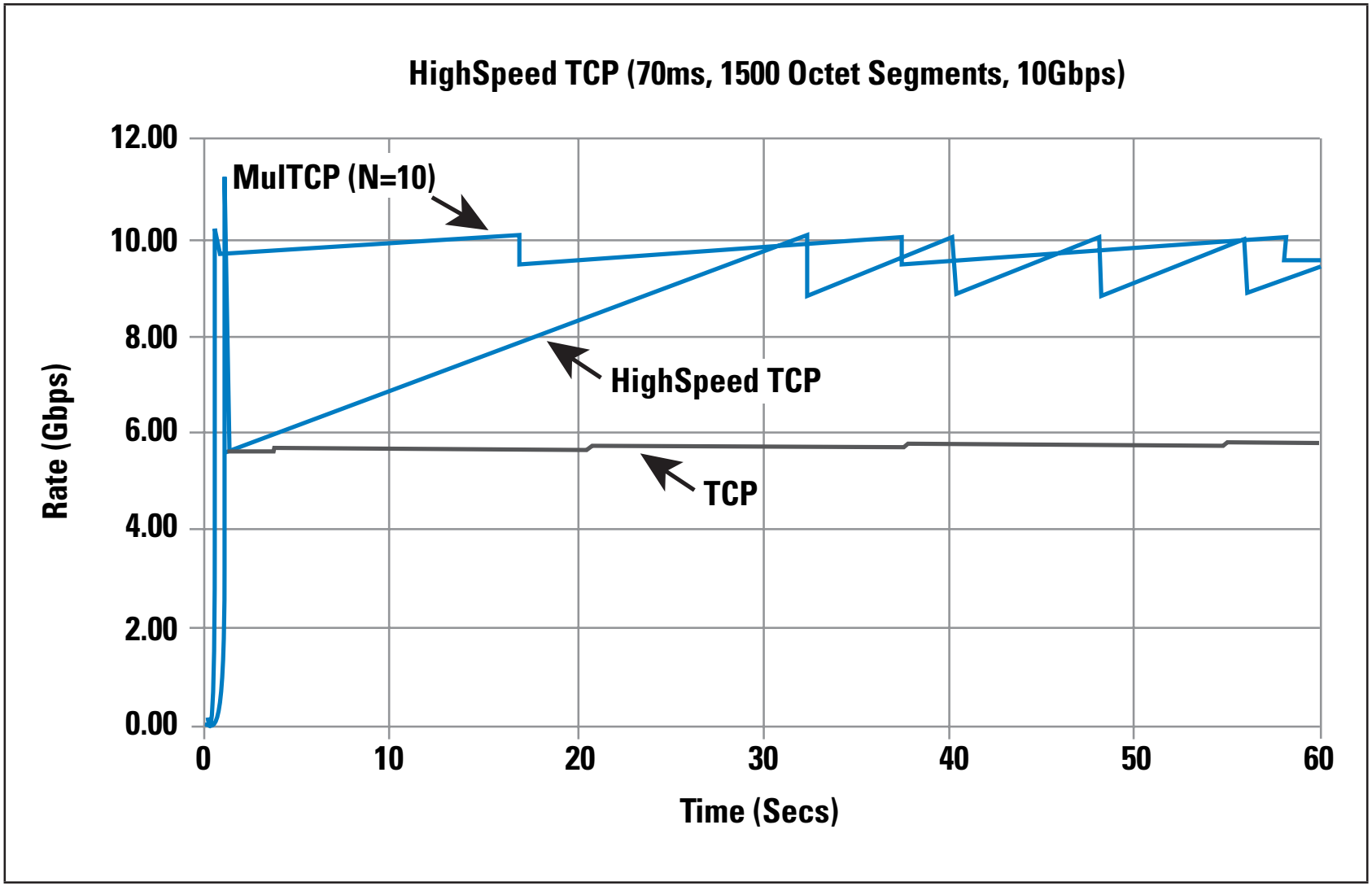
Idea

- Increase 1 segment up to 10 Mbps, 6 segments up to 100 Mbps, 26 segments at 1 Gbps, 70 segments at 10 Gbps
- Reduce the window W by $W/2$ at 10 Mbps, $W/3$ at 100 Mbps, $W/5$ at 1 Gbps, $W/10$ at 10 Gbps
- Limit slow start to not overload the network

Properties

- Recovers faster from the initial rate halving
- Short probing cycles compared to traditional TCP
- Additional slow start improvement to prevent extreme traffic bursts during slow start

HighSpeed TCP Performance



Scalable TCP

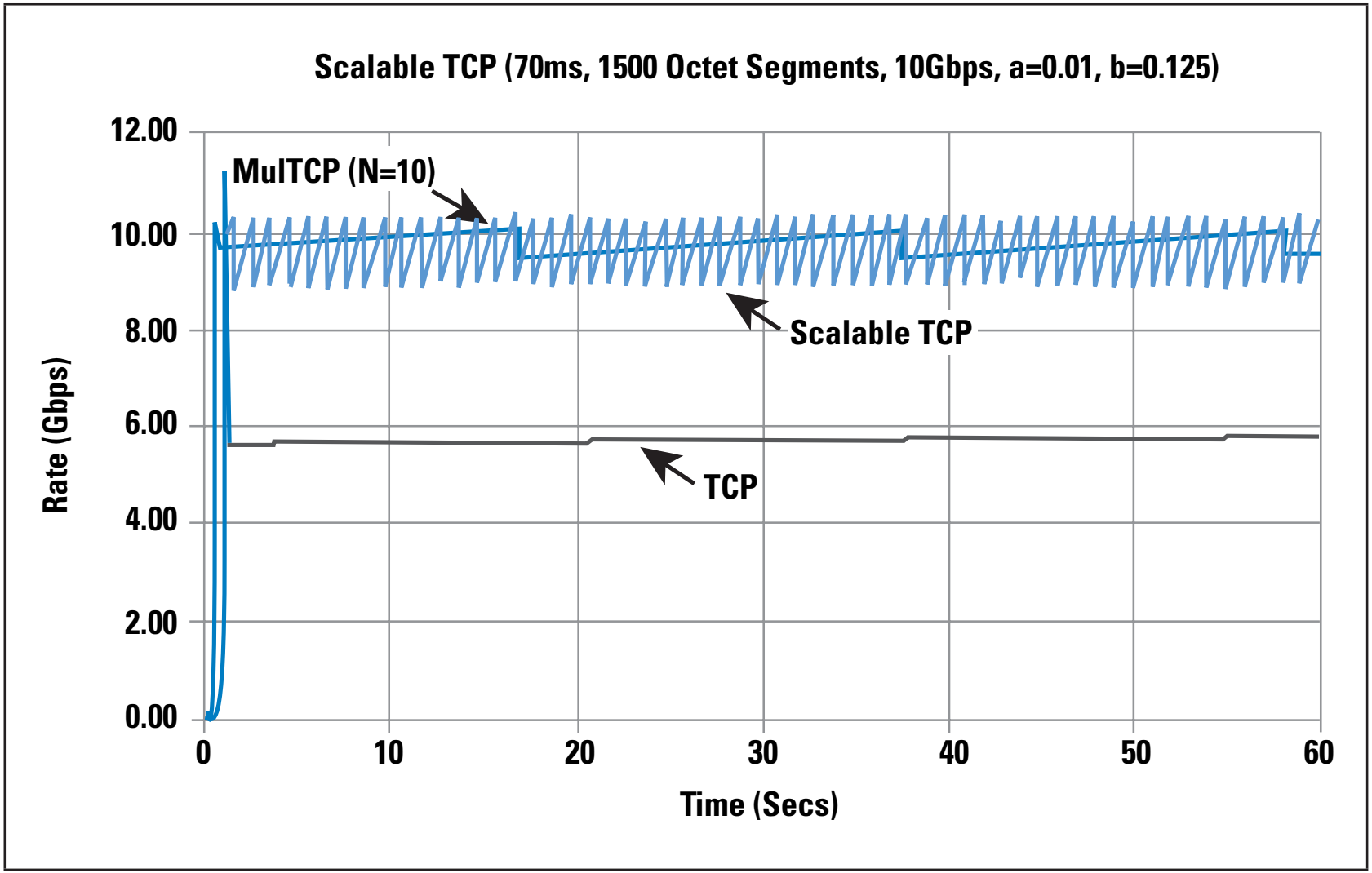
Idea

- Use multiplicative increase instead linear increase
- Parameter a controls the multiplicative increase, parameter b controls the fraction of the multiplicative decrease
- Make probing times proportional only to the round trip time, ignoring the current sending rate

Properties

- Higher frequency $f = log(1 - b)/log(1 + a)$ of oscillation
- Enable Scalable TCP only for windows above a certain size to allow smooth coexistance

Scalable TCP Performance



CUBIC

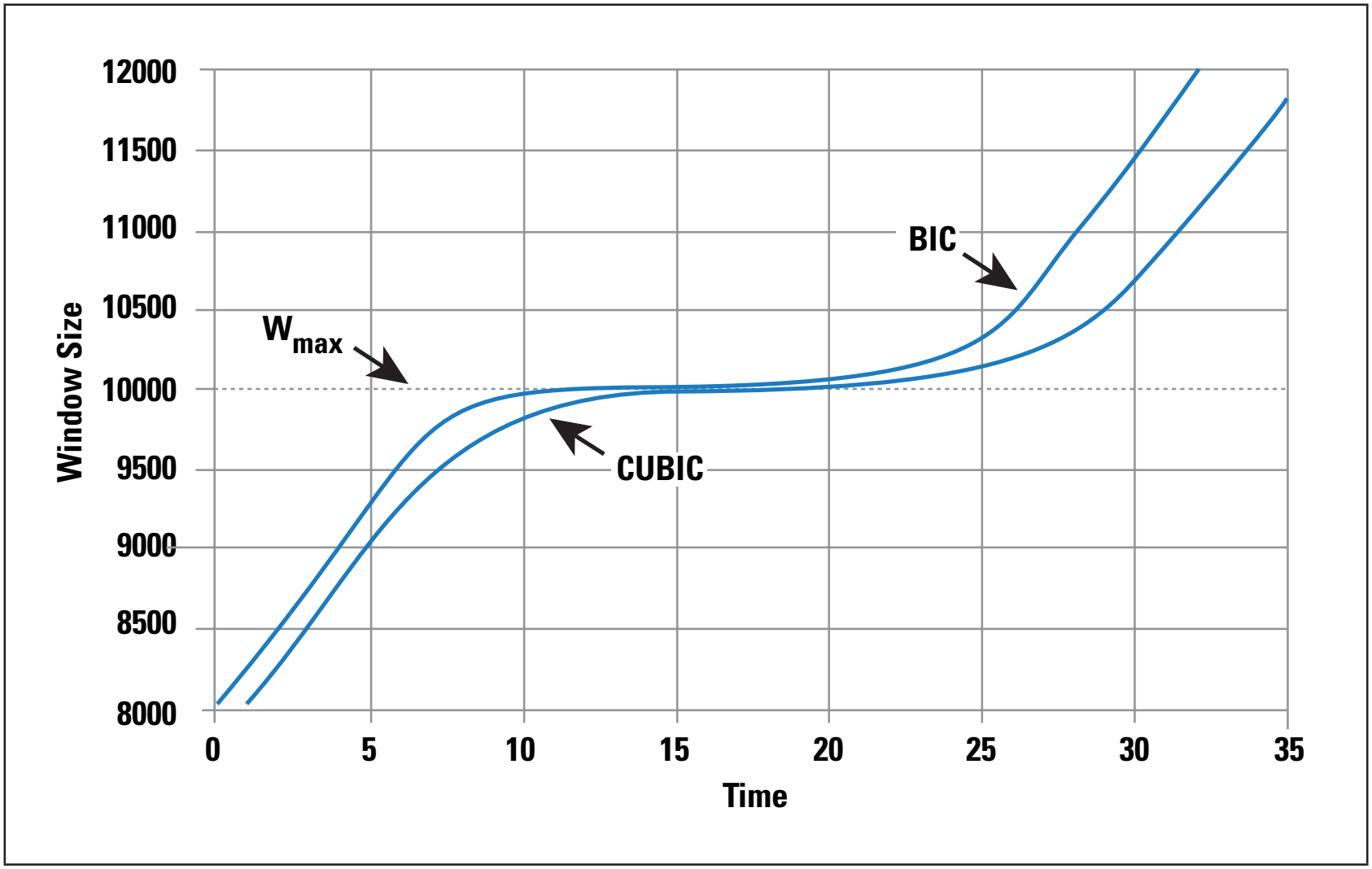
Idea

- After packet loss and window reduction, inflate the window quickly, gradually slowing down the window increase while getting closer to the window size where packet loss occurred last time
- $W = C(T - K)^3 + W_{max}$ where C is a constant scaling factor, T is the elapsed time since the last window reduction event, W_{max} is the size of the window prior to the most recent reduction, and $K = \sqrt[3]{\frac{W_{max}\beta}{C}}$

Properties

- CUBIC is the default congestion control algorithm of the Linux kernel since kernel version 2.6.18 (2006)
- TCP friendly parameters are $\beta = 0,2$ and $C = 0.4$

CUBIC Performance

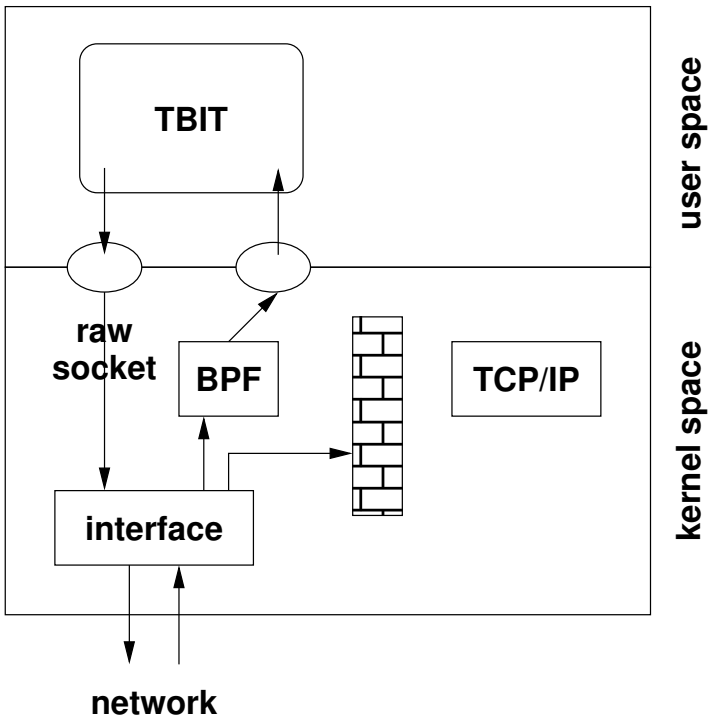


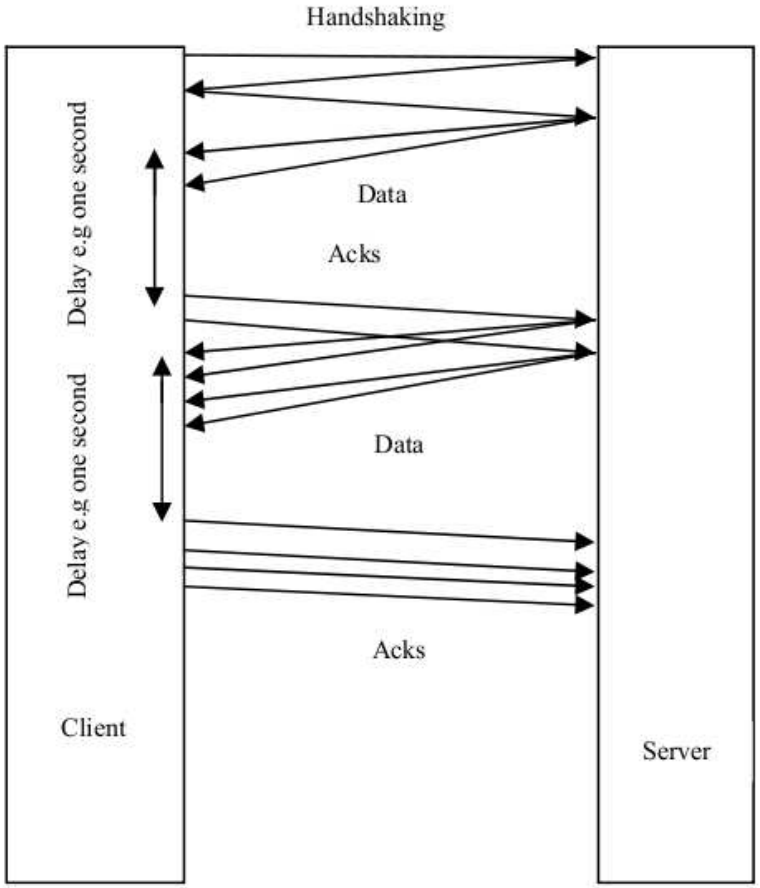
Fairness

- A crucial factor for the design of congestion control algorithms is *mutual fairness* and *overall network efficiency*.
- Does the protocol negotiate a fair share of the underlying network resource in the face of competing resource claims from concurrent transport flows?
- How do you define fairness given very different TCP connections (short vs. long, small RTT vs. long RTT, small flows vs. elephant flows, . . .)?

Probing TCP Congestion Control Algorithms

Probing TCP from User Space





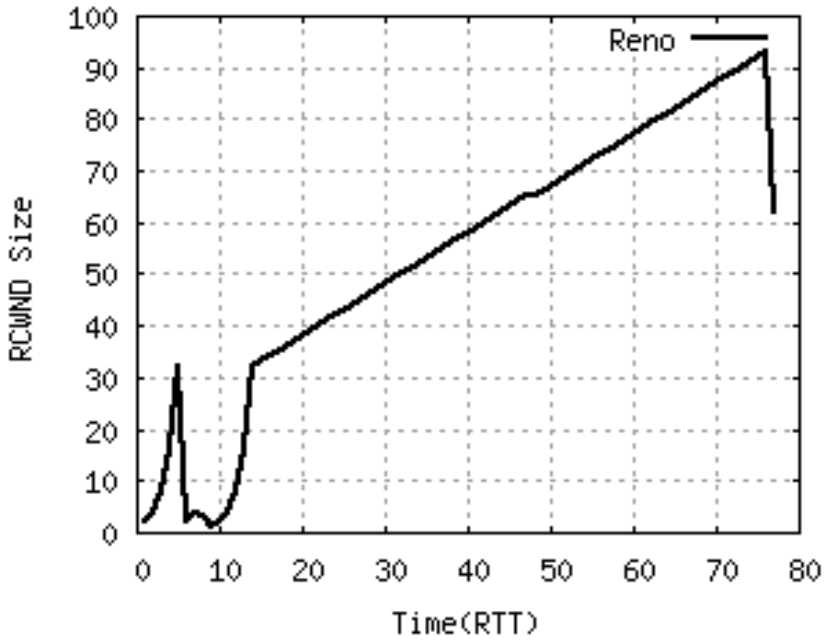
Congestion Control Probing Algorithm I

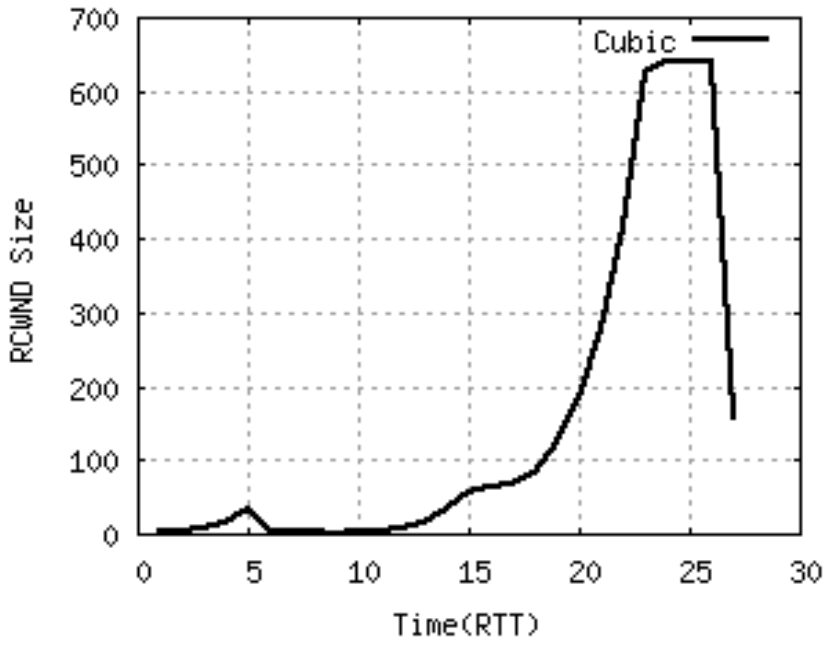
1. Our software sets up the local firewall and BPF as TBIT does
2. C sends a TCP SYN packet with the destination address of S and the destination port number 80; the maximum segment size (MSS) is set to 100 bytes
3. S sends a TCP SYN/ACK segment
4. C requests the base web page from S by sending an HTTP GET request
5. S starts sending the base page with 100-byte packets
6. C stores all received packets but does not immediately send ACKs

Congestion Control Probing Algorithm II

7. After capturing packets for a specific time, C sends ACKs for all captured packets; C counts the number of packets in its buffer and sets RCWND as the difference between the last buffer size and the newer buffer size
8. After sending the ACK for packet 62, C does not send any more ACKs and ignores all the packets in its buffer; at this point RCWND is 32 and CWND is 64 on the client and the server respectively
9. C sends two more ACKs for packet 62 to make the remote TCP go to the Fast Retransmission phase
10. C continues sending ACKs for received packets without any additional packet drops, measures the RCWND size and stores this measured values in a file
11. After receiving a FIN segment from S, C closes the connection

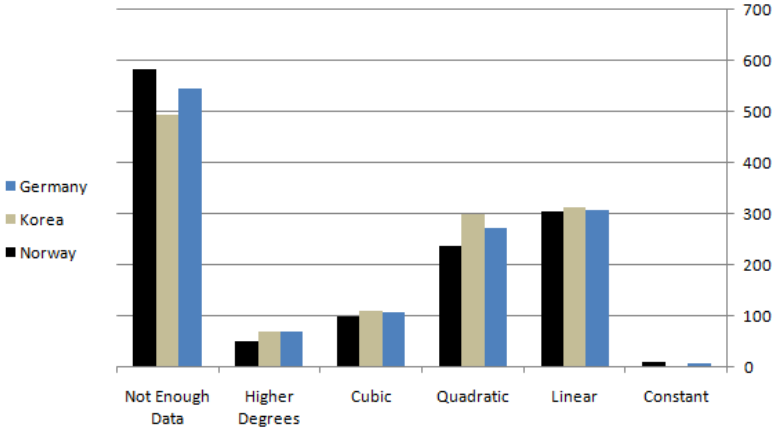
Validation of the Approach

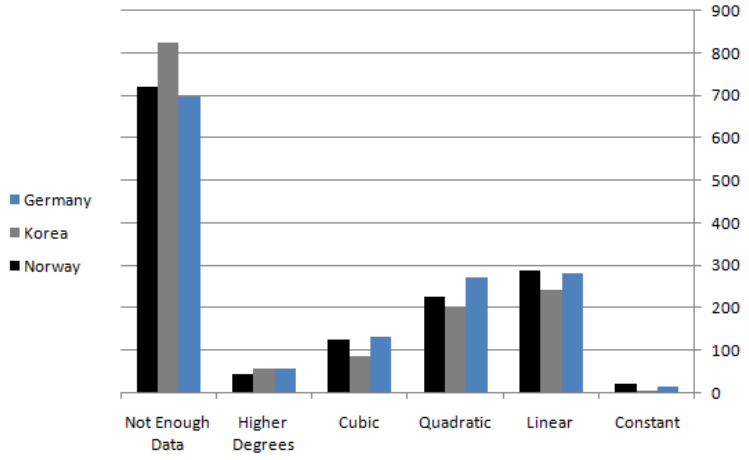




- Testing of the algorithms in a controlled lab environment
- Using MATLAB functions for fitting a higher degree polynomial against the dataset

Probing the Wild Internet





- Probed a collection of $\approx$ 2100 web servers
- There is a significant number of web servers not using a linear inflation of the congestion window
- The study should be repeated to reduce the number of probing failures
- The location of the probe has no significant impact on the results

References

G. Huston.

Gigabit TCP.
The Internet Protocol Journal, 9(2), June 2006.

A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock.

Host-to-Host Congestion Control for TCP.
IEEE Communications Surveys and Tutorials, 12(3):304–342, 2010.

Y.-T. Li, D. Leith, and R. N. Shorten.

Experimental Evaluation of TCP Protocols for High-Speed Networks.
IEEE/ACM Transactions on Networking, 15(5):1109–1122, October 2007.

J. Padhye and S. Floyd.

On Inferring TCP Behavior.
In *Proc. SIGCOMM 2001*, pages 287–298, San Diego, March 2001. ACM.

S. Feyzabadi and J. Schönwälder.

Identifying TCP Congestion Control Algorithms Using Active Probing.
In *(under review)*, 2010.

F. Qian, A. Gerber, Z. M. Mao, S. Sen, O. Spatscheck, and W. Willinger.

TCP Revisited: A Fresh Look at TCP in the Wild.
In *Proceeding of the Internet Measurement Conference (IMC 2009)*, New York, NY, USA, November 2009. ACM.

Part: New Internet Transport Protocols

Motivation for new Transport Protocols

Classic Internet Transports

- Transmission Control Protocol (TCP):
  - stream oriented
  - reliable ordered delivery of data
  - connection oriented (establishment, delivery, teardown)
  - congestion aware (AIMD congestion control)
  - head of line blocking
- User Datagram Protocol (UDP):
  - packet oriented
  - unreliable, unordered delivery of data
  - connectionless (delivery)
  - congestion unaware
  - no blocking

Why Additional Transports?

- Would it not be nice to have ...
  - a congestion aware datagram transport?
  - a reliable connection oriented transport which preserves message boundaries?
  - a reliable protocol which does not suffer from head of line blocking?
  - a transport protocol which can recover from failures in the underlying layers?
  - a transport which can bundle multiple independent streams?
  - a transport which starts quickly and recovers fast from packet loss?

Example #1: Loading a Web Page

- A web page usually consists of an HTML page which includes several embedded elements (e.g., graphics).
- Various ways to load a web page:
  - Load elements sequentially using separate TCP connections
    $\Rightarrow$ slow, high overhead for small elements (e.g., icons)
  - Load elements sequentially using a single TCP connection
    $\Rightarrow$ better, head-of-line blocking, network / server friendly
  - Load elements in parallel using multiple TCP connections
    $\Rightarrow$ faster, high overhead for small elements, no shared congestion state
- Ideally, one would like to use a single connection and multiple independent streams within this connection...

Example #2: Realtime Streams

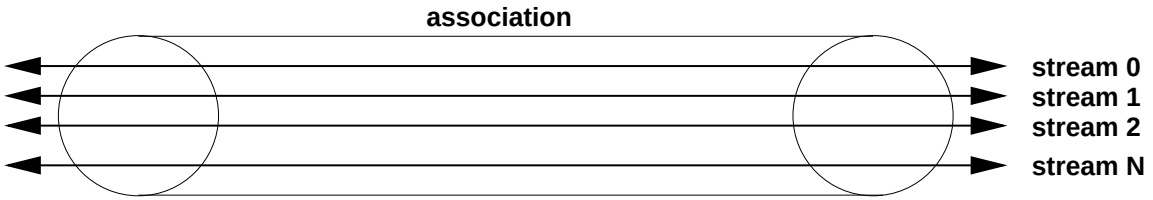
- TCP's reliable ordered delivery is unsuited for real-time applications such as voice and video streaming.
- UDP is typically used, but ...
  - UDP is not congestion aware
  - TCP streams may suffer from bad behaving UDP streams
  - voice and video may be treated differently as separate streams
- Ideally, one would like to use an unreliable congestion aware datagram transport.

Stream Control Transmission Protocol (SCTP)

Stream Control Transmission Protocol (SCTP)

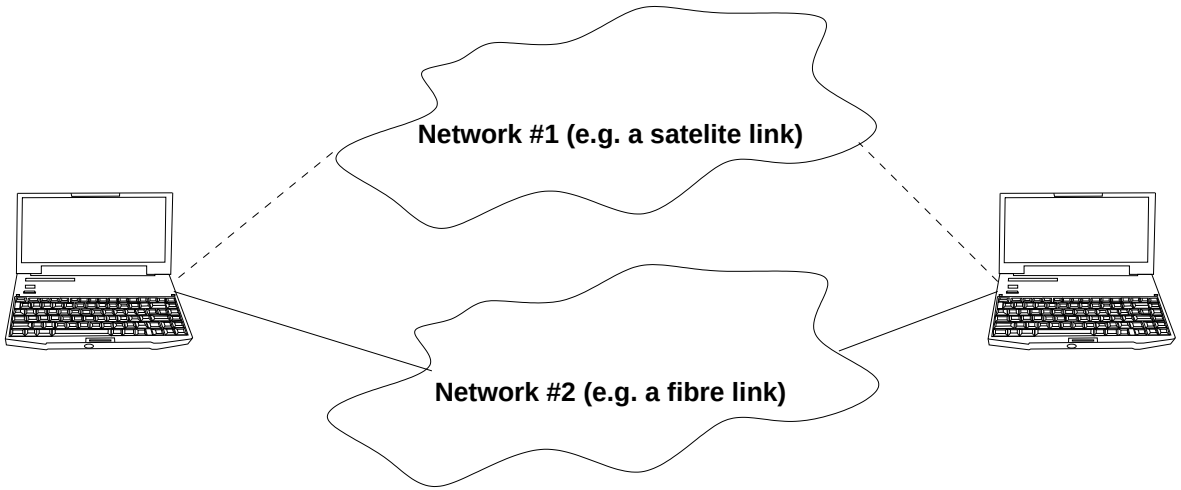
- The Stream Control Transmission Protocol (SCTP) provides the following services [**?**]:
  - Reliable, ordered and unordered delivery of data
  - Message oriented (preserves application layer framing)
  - Multiple independent streams bundled in an association
  - Multi-homing of association endpoints for fast failover
  - Initiation protection and graceful shutdown
- Initial version developed 1998-2000 in the IETF mainly as a transport for signaling protocols.
- SCTP has much wider applicability and is continuously extended.

SCTP Streams and Associations



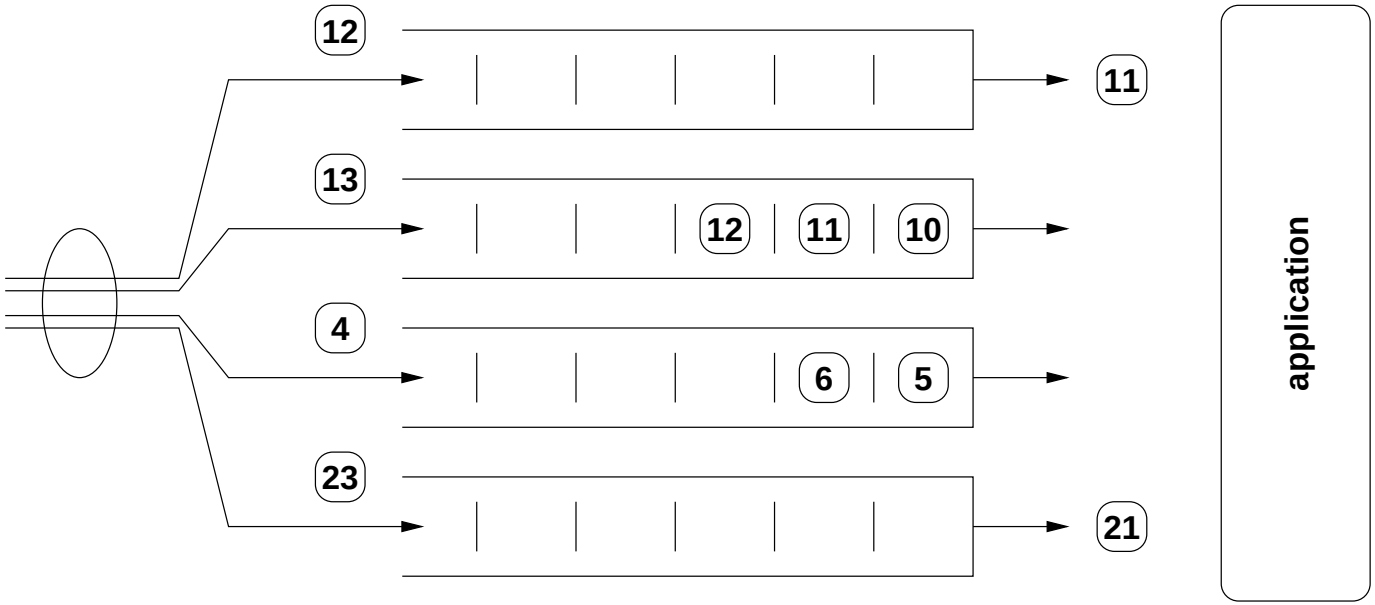
- Data transfer between two SCTP hosts takes place in the context of an association.
- An association may contain multiple data streams and each stream has the property of independently sequenced delivery.
- A message lost in one stream thus does not affect the delivery in other streams.
- SCTP accomplishes multi-streaming by creating independence between data transmission and data delivery.

SCTP Multi-homing



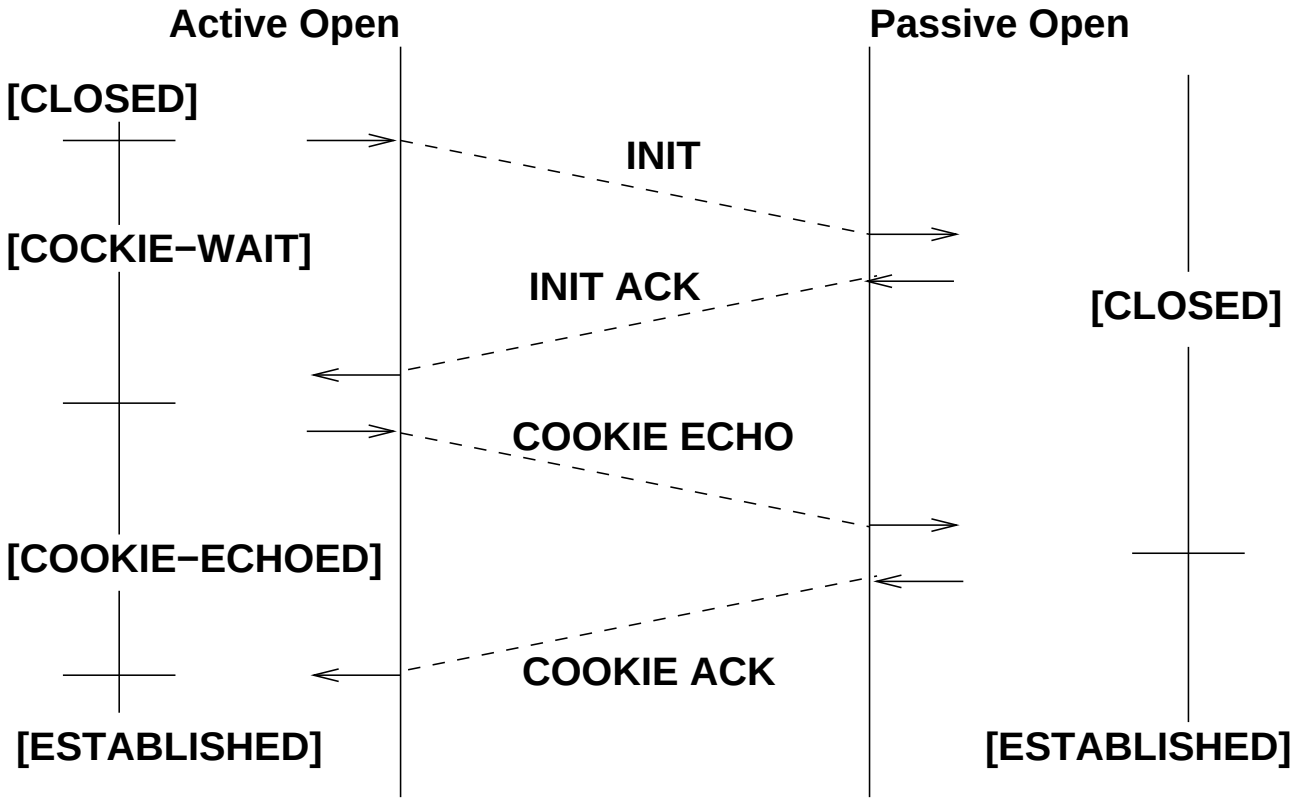
- SCTP allows SCTP endpoints to have multiple IP addresses.
- This multi-homing feature provides the benefit of potentially greater survivability of an SCTP association in the presence of network failures.
- Multi-homing is not designed as a load balancing mechanism.

SCTP Sequencing



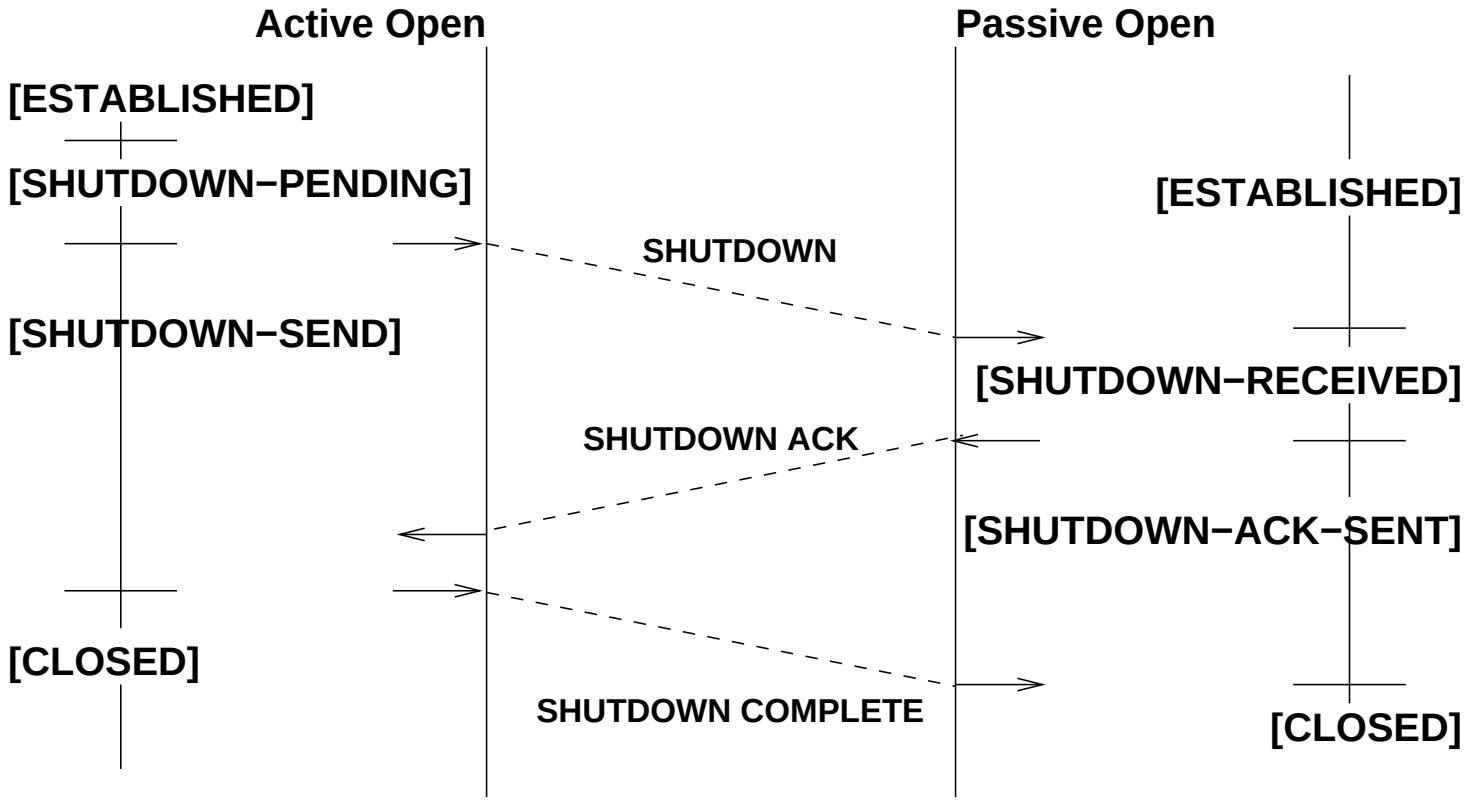
- Every stream has its own independent sequence number space
- Streams progress independently (no head-of-line blocking)

SCTP Association Establishment



- A TCB is not allocated on the server side before the `COOKIE-ECHO` has been received to avoid TCP's `SYN`-flooding problem

SCTP Association Teardown



- SCTP supports only a full teardown procedure (TCP's half closed connections do not exist in SCTP)

SCTP State Machine

State	Description
`CLOSED`	Initial and final state
`COOKIE-WAIT`	Waiting for a cookie
`COOKIE-ECHOED`	Cookied echoed to the server
`ESTABLISHED`	Association established
`SHUTDOWN-PENDING`	Shutdown requested by application
`SHUTDOWN-SENT`	Shutdown initiated
`SHUTDOWN-RECEIVED`	Shutdown request received
`SHUTDOWN-ACK-SENT`	Shutdown request acknowledged

- The SCTP state machine has the states shown above and is slightly simpler than TCP's state machine.
- For a description of the transitions, see RFC 4960.

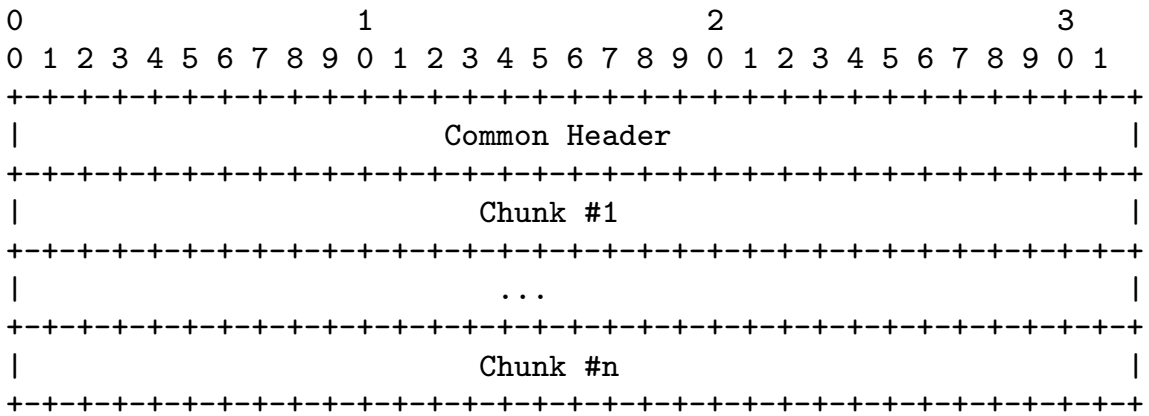
SCTP Fragmentation / Reassembly / Bundling

- In order to preserve application layer message boundaries, SCTP has to perform fragmentation and reassembly, since SCTP messages should not exceed the path MTU.
- Fragmentation / reassembly happens on the data chunk level and not on a message level.
- If a receiver runs out of buffer space while waiting for more fragments to arrive, it may pass the incomplete message to the application via a special API.
- Applications can also request the bundling of chunks so that they are shipped in a single SCTP message.

SCTP Congestion Control

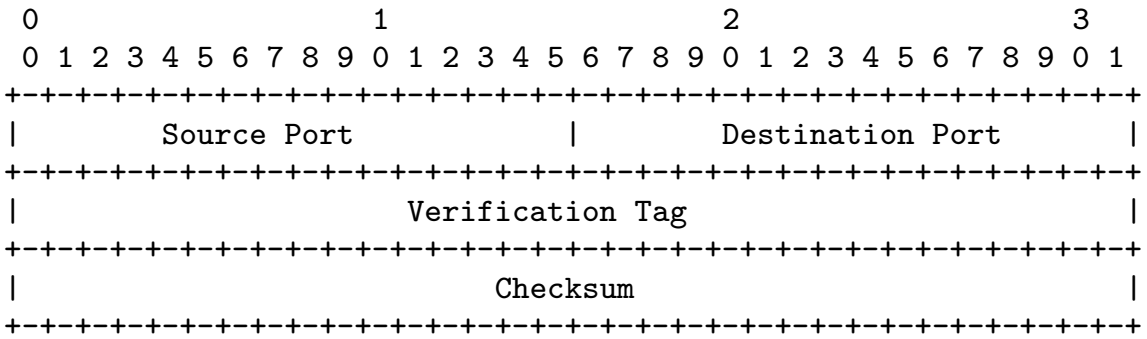
- SCTP uses selective acknowledgements (SACKs) which speeds up the detection of loss and increases bandwidth utilization.
- During congestion avoidance, `cwnd` is increased by the number of acknowledged bytes and not the number of segments.
- During congestion avoidance, `cwnd` can only be increased when the full `cwnd` is utilized.
- SCTP begins fast retransmission after receipt of four duplicate acknowledgedments (TCP after three).

SCTP Packet Format



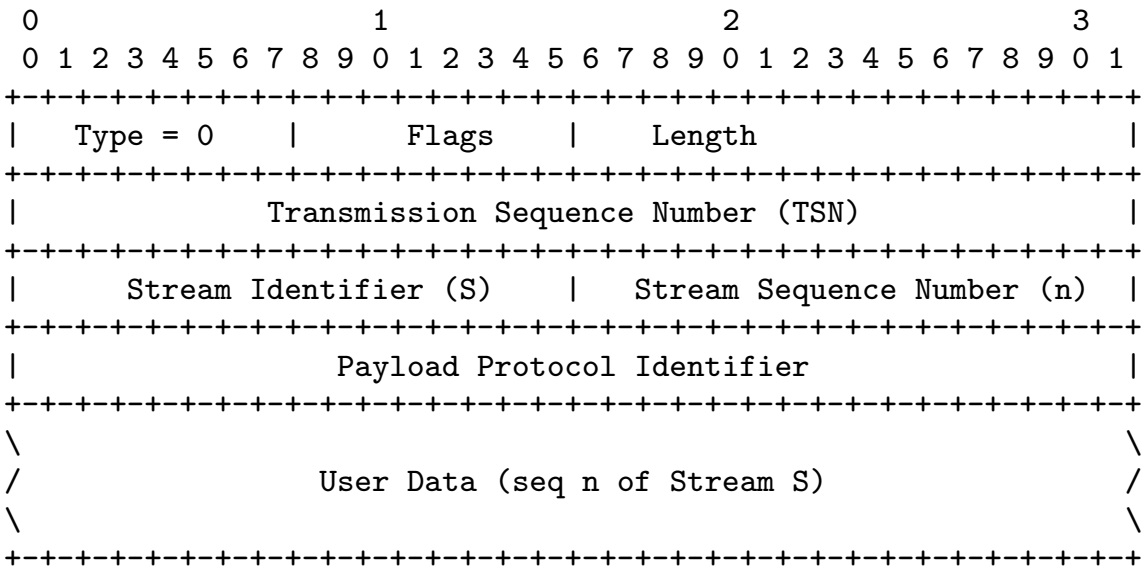
- An SCTP packet is composed of a common header and chunks.
- A chunk contains either control information or user data.
- Multiple chunks can be bundled into one SCTP packet up to the MTU size.

SCTP Common Header Format



- The `Source Port` and `Destination Port` fields contains the port number used by the sending / receiving application layer process.
- The `Verification Tag` field is used by the receiver to verify that the SCTP packet belongs to the current association and is not an old or stale packet from a previous association.
- The `Checksum` field contains a 32-bit CRC checksum as specified in RFC 4960.

SCTP Data Chunk Format I



- The `Type` field indicates the chunk type. Data chunks use the type number 0.
- The `Flags` field contains a set of binary flags:
  - `U`: The data chunk is unordered and there is no `Stream Sequence Number` assigned to the data chunk.

SCTP Data Chunk Format II

  - `B`: Beginning of a fragment of a user message.
  - `E`: Ending of a fragment (last fragment) of a user message.
- The `Length` field indicates the size of the chunk in bytes including the chunk header fields.
- The `Transmission Sequence Number` field contains the transmission sequence number which is also used by the receiver to reassemble messages.
- The `Stream Identifier` identifies the stream to which the following data belongs.
- The `Stream Sequence Number` identifies the stream sequence number of the following user data within the stream identified by the `Stream Identifier`.

SCTP Data Chunk Format III

- The `Payload Protocol Identifier` identifies the upper layer application protocol and is opaque from the viewpoint of an SCTP protocol engine.
- The `User Data` is of variable length and contains the actual payload.

Datagram Congestion Control Protocol (DCCP)

DCCP Motivation

- Multimedia streaming applications and online games often prefer timeliness over reliability.
- There is a risk that growing non-congestion-controlled UDP traffic may lead to congestion collapse.
- Implementation of effective congestion control in application protocols is difficult.
- UDP flows are hard for firewalls to handle due to a lack of a setup and teardown exchange.
- See RFC 3714 for a discussion why DCCP is not just TCP with relaxed reliability.

DCCP Features I

- Unreliable flows of datagrams.
- Reliable handshakes for connection setup and teardown.
- Reliable negotiation of options, including negotiation of a suitable congestion control mechanism.
- Mechanisms allowing servers to avoid holding state for unacknowledged connection attempts and already-finished connections.
- Support for Early Congestion Notification (ECN)
- Acknowledgement mechanisms communicating packet loss and ECN information. Acks are transmitted as reliably as the relevant congestion control mechanism requires, possibly completely reliable.

DCCP Features II

- Optional mechanisms that tell the sending application, with high reliability, which data packets reached the receiver, and whether those packets were ECN marked, corrupted, or dropped in the receive buffer.
- Suport for Path Maximum Transmission Unit (PMTU) discovery.

DCCP Congestion Control

- DCCP supports multiple congestion control mechanisms that are identified by so called congestion control identifiers (CCIDs).
- The following congestion control mechanism have been defined so far:
  - TCP-like Congestion Control (CCID-2) [RFC 4341]
  - TCP-Friendly Rate Control (TFRC) (CCID-3) [RFC 4342, RFC 5348]
  - TCP-Friendly Rate Control for Small Packets (TFRC-SP) [RFC 5622]
- The congestion control mechanism can be negotiated.
- Additional congestion control mechanism can be added in the future.

DCCP Messages

- DCCP-Request
- DCCP-Response
- DCCP-Data
- DCCP-Ack
- DCCP-DataAck
- DCCP-CloseReq
- DCCP-Close
- DCCP-Reset
- DCCP-Sync
- DCCP-SyncAck

DCCP Connection Establishment

```
  Client State                                        Server State
     CLOSED                                              LISTEN
1.   REQUEST    -->        Request          -->
2.              <--        Response         <--        RESPOND
3.   PARTOPEN   -->     Ack, DataAck        -->
4.              <--  Data, Ack, DataAck     <--        OPEN
5.   OPEN       <->  Data, Ack, DataAck     <->        OPEN
```

- Handshake allows middleboxes to track DCCP connections

DCCP Connection Teardown (#1)

```
  Client State                                    Server State
     OPEN                                            OPEN
1.              <--          CloseReq          <--   CLOSEREQ
2.   CLOSING    -->           Close            -->
3.              <--           Reset            <--   CLOSED (LISTEN)
4.   TIMEWAIT
5.   CLOSED
```

- Server initiates teardown procedure.
- Client takes the `TIMEWAIT` burden.

DCCP Connection Teardown (#2)

```
  Client State                                    Server State
     OPEN                                            OPEN
1.   CLOSING    -->          Close          -->
2.              <--          Reset          <--    CLOSED (LISTEN)
3.   TIMEWAIT
4.   CLOSED
```

- Client initiates teardown procedure.
- Client takes the `TIMEWAIT` burden.

DCCP Connection Teardown (#3)

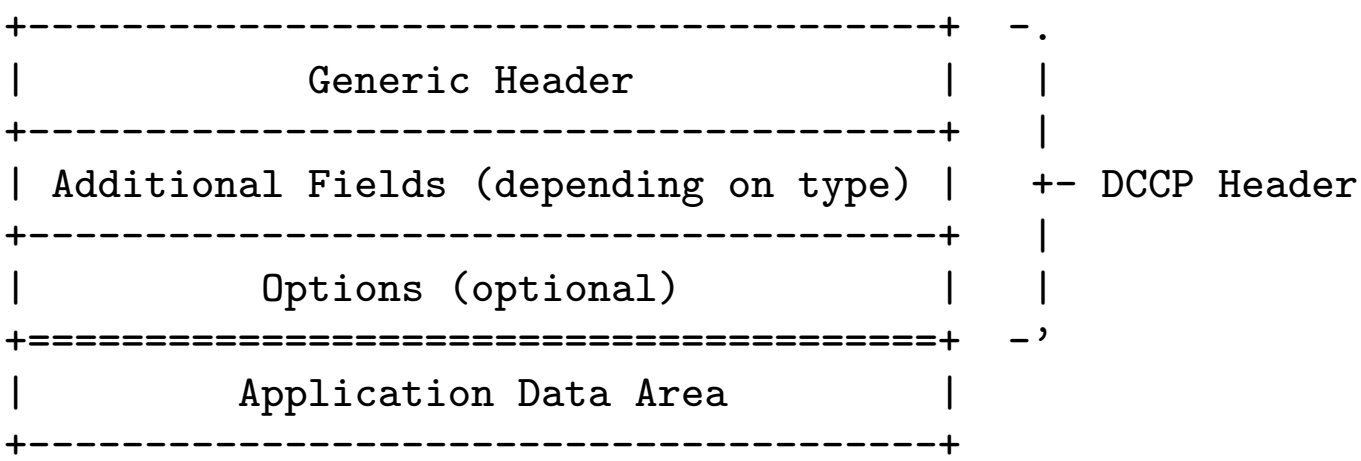
```
  Client State                                    Server State
     OPEN                                            OPEN
1.              <--          Close          <--    CLOSING
2.   CLOSED     -->          Reset          -->
3.                                                   TIMEWAIT
4.                                                   CLOSED (LISTEN)
```

- Server initiates teardown procedure.
- Server takes the `TIMEWAIT` burden.

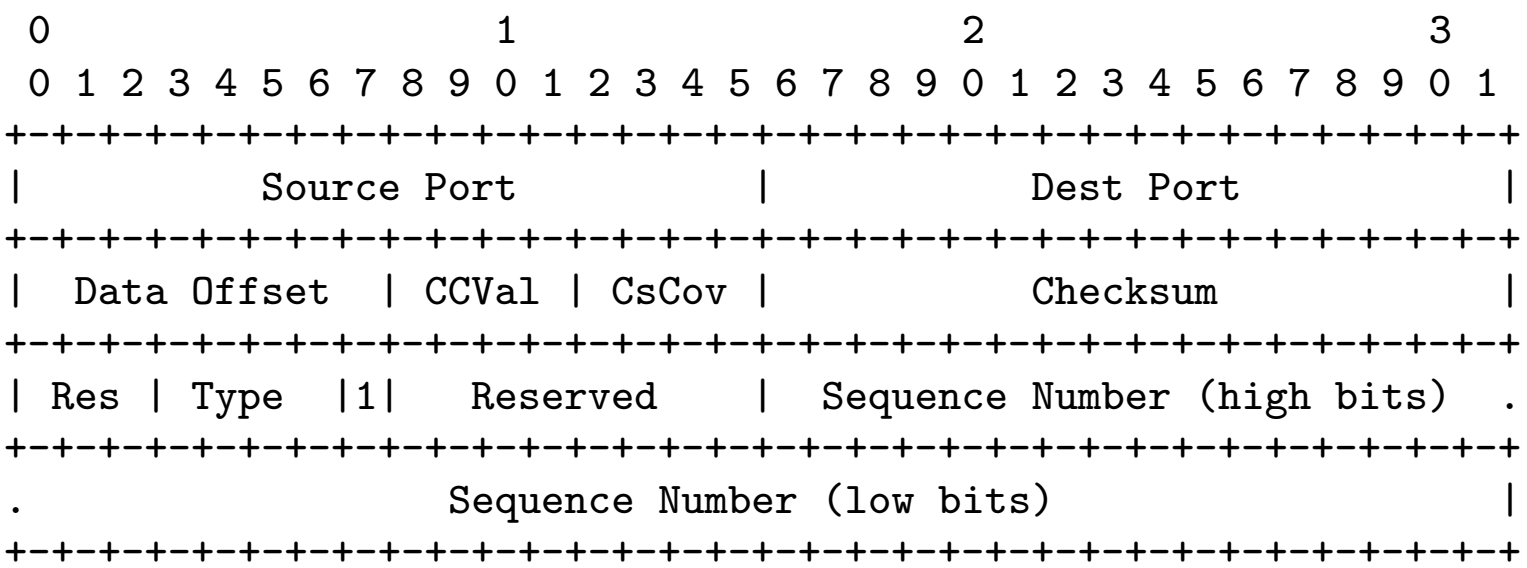
DCCP State Machine

- See RFC 4340 section 8.4...

DCCP Packet Formats

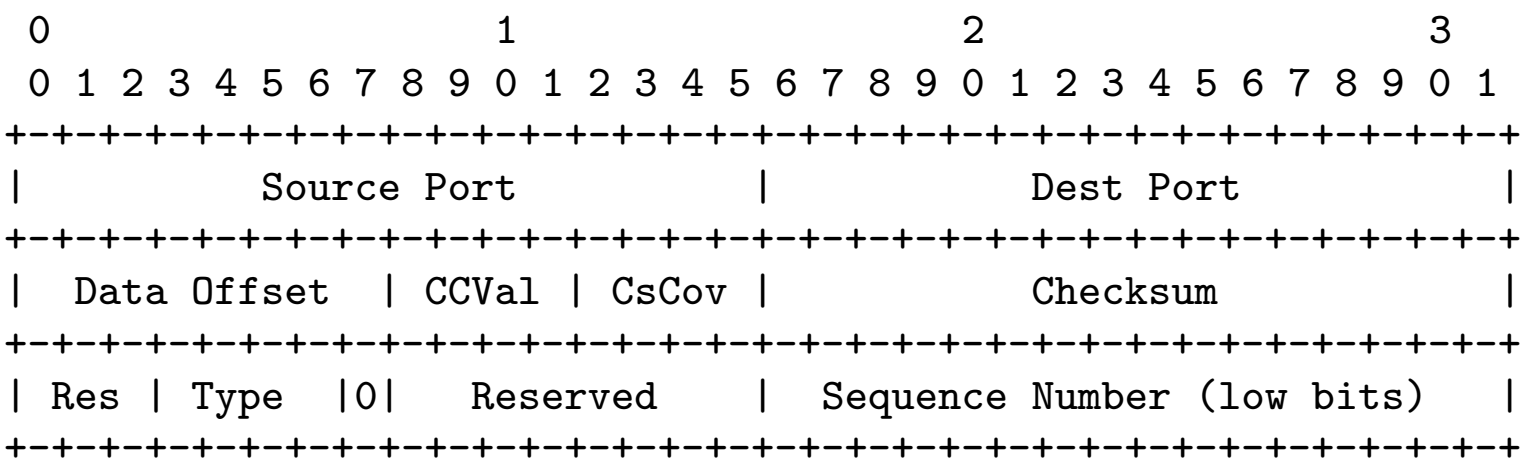


DCCP Generic Header



- The generic header with the "Extended Sequence Number" bit set to 1.

DCCP Generic Header



- The generic header with the "Extended Sequence Number" bit set to 0.

References I

R. Stewart.

Stream Control Transmission Protocol.
RFC 4960, September 2007.

L. Ong and J. Yoakum.

An Introduction to the Stream Control Transmission Protocol (SCTP).
RFC 3286, Ciena Corporation, Nortel Networks, May 2002.

R. Stewart, K. Poon, M. Tuexen, V. Yasevich, and P. Lei.

Sockets API Extensions for Stream Control Transmission Protocol (SCTP).
Internet Draft (work in progress) <draft-ietf-tsvwg-sctpsocket-17.txt>, The Resource Group, Sun Microsystems, Univ. of Applied Sciences Muenster, HP, Cisco Systems, July 2008.

R. Stewart and C. Metz.

SCTP: New Transport Protocol for TCP/IP.
IEEE Internet Computing, November 2001.

A. L. Caro, J. R. Iyengar, P. D. Amer, S. Ladha, G. J. Heinz, and K. C. Shah.

SCTP: A Proposed Standard for Robust Internet Data Transport.
IEEE Computer, 36(11), November 2003.

S. Fu and M. Atiquzzaman.

SCTP: State of the Art in Research, Products, and Technical Challenges.
IEEE Communications Magazine, 42(4), April 2004.

E. Kohler, M. Handley, and S. Floyd.

Designing DCCP: Congestion Control Without Reliability.
In *Proc. SIGCOMM 2006*, pages 27–38, Pisa, September 2006. ACM.

References II

E. Kohler, M. Handley, and S. Floyd.

Datagram Congestion Control Protocol (DCCP).
RFC 4340, UCLA, UCL, ICIR, March 2006.

Part: Internet Quality of Service

Basic Quality of Service Concepts

Elastic vs. Inelastic Traffic

- Elastic Traffic:
  - Can adjust to changes in delay and throughput
  - Traditional type of traffic in the Internet
  - Examples: E-mail, file transfers
- Inelastic Traffic:
  - Does not easily, if at all, adapt to changes in delay and throughput
  - Examples: Video and audio streams, real-time stock trading

Quality of Service (QoS) Parameters

- Throughput:
  - Some applications require a minimum throughput.
- Delay:
  - Some applications require a minimum delay.
- Jitter:
  - Some applications do not tolerate arbitrary delay variations (jitter).
- Packet loss:
  - Real-time applications vary in the amount of packet loss, if any, they can sustain.

Flows

- Quality of Service (QoS) support requires to treat aggregations of packets that belong together.
- RFC 1633 introduces the concept of a flow as follows:
  - A flow is a distinguishable stream of related datagrams that results from a single user activity and requires the same QoS.
  - A flow has a single source but may have N destinations.
  - An N-way teleconference will generally require N flows, one originating at each site.
- A transport connection carrying a video stream is an example for a single flow.

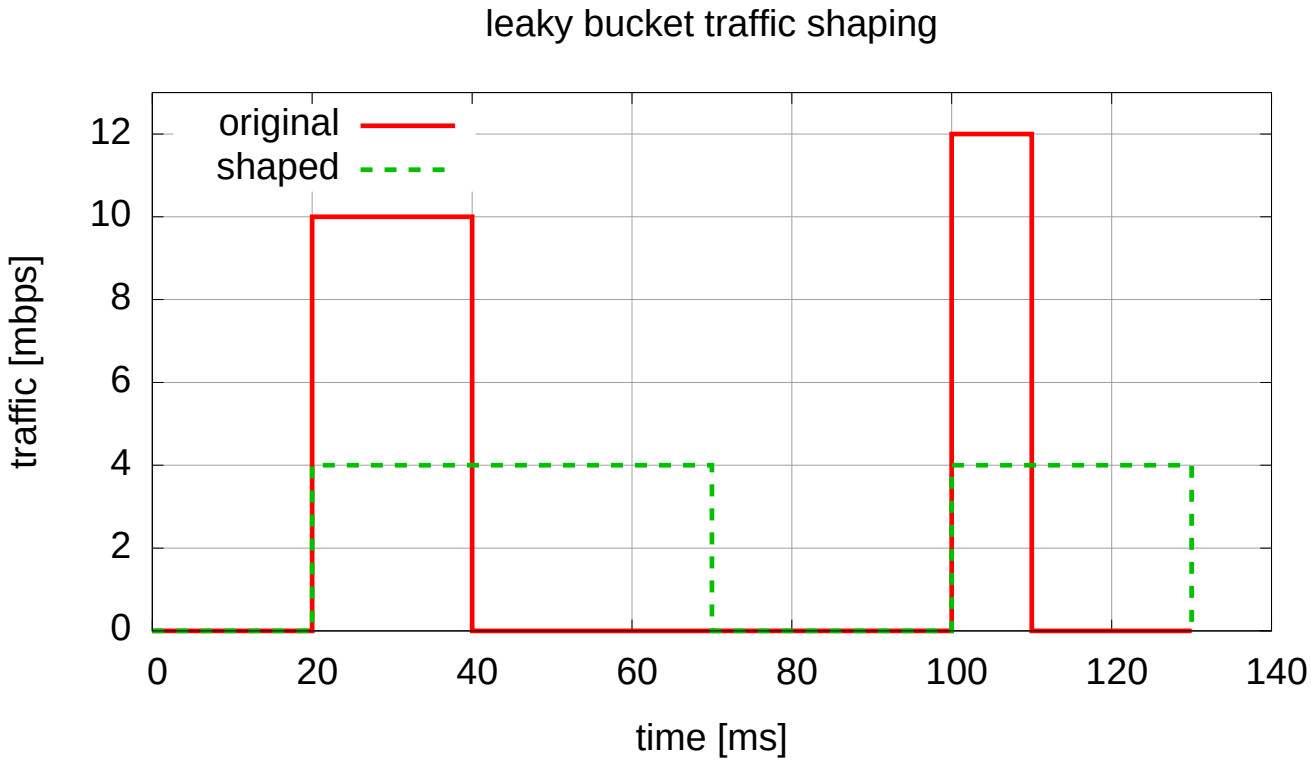
Controlling Quality of Service (QoS)

- Admission control:
  - New traffic flows are only admitted if there are enough resources to handle the flows.
  - Requires signaling phase before the data transfer.
- Routing algorithm:
  - Routing decisions may be based on a variety of QoS parameters, not just minimum delay.
- Queueing discipline:
  - Queue packets to meet QoS constraints (where necessary).
- Discard policy:
  - Discard packets (manage congestion) to meet QoS constraints.

Traffic Shaping: Leaky Bucket

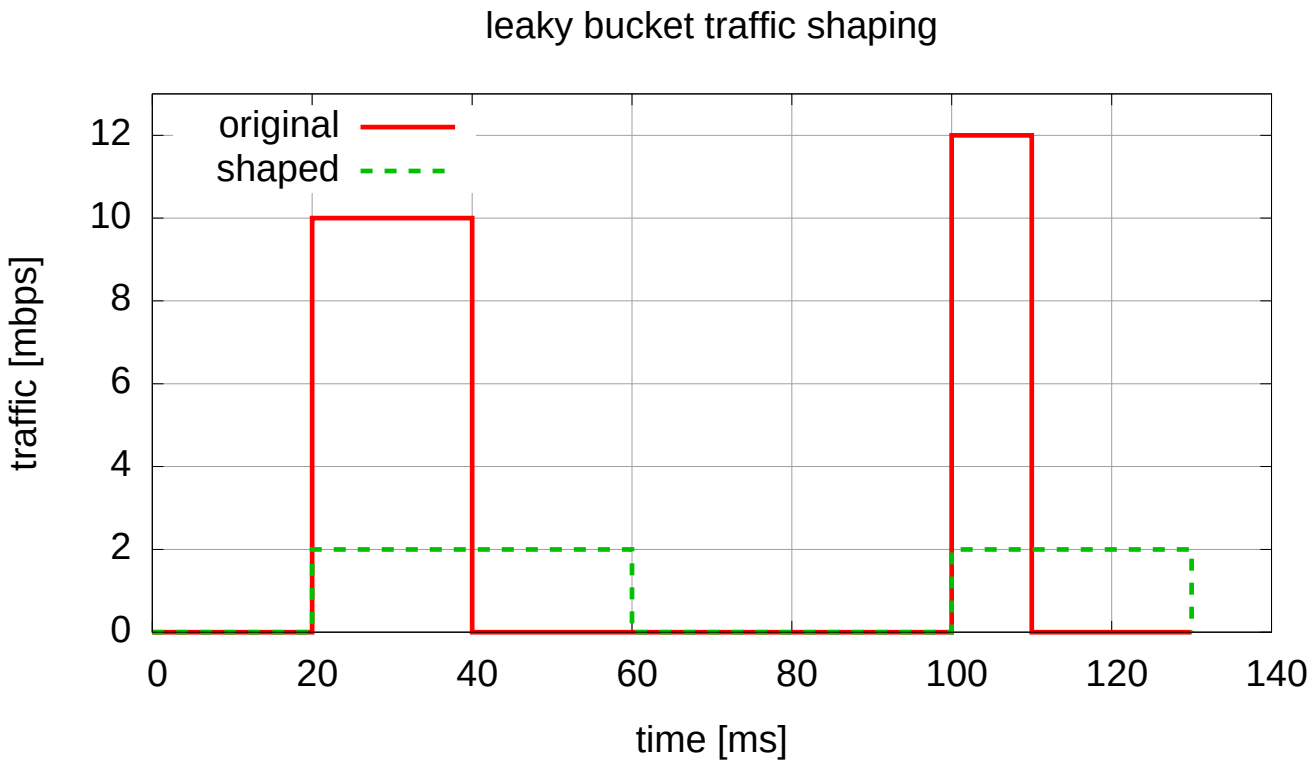
- Leaky buckets shape bursty traffic into a smooth traffic stream.
- The leaky bucket uses a conceptual bucket that can be filled with packets and which has a leak through which packets leave the bucket with a fixed rate.
- Arriving packets will be discarded if the bucket is full.
- Leaky bucket parameters:
  - bucket capacity C
  - departure rate R
- A leaky bucket is a single server queue with constant service time R and limited queue size C.
- The bucket capacity can be counted in packets or bytes.

Leaky Bucket #1: $C = 20kB$, $R = 4mbps$



- At $t = 40$, the bucket has received over the last $20ms$ the volume of $(10mbps - R) \cdot 20ms = 15kB$ of data
- Sending the buffered data at data rate R takes $30ms$

Leaky Bucket #2: $C = 5kB$, $R = 2mbps$



- At $t = 40$, the bucket has received over the last $20ms$ the volume of $(10mbps - R) \cdot 20ms = 20kB$ of data
- Since $C = 5kB$, $15kB$ of data will be lost
- Sending the buffered data at data rate R takes $20ms$

Traffic Shaping: Token Bucket

- Token buckets allow some burstiness if some capacity has not been used recently.
- The token bucket uses a conceptual bucket that is filled with tokens with a constant arrival time. Packets are allowed to leave the token bucket system once there are tokens available.
- Token bucket parameters:
  - token bucket capacity C
  - token arrival rate R
  - maximum output rate M
- Each token may represent a packet or a fixed number of bytes.

Token Bucket Burst Length

- Calculation of the token bucket burst length S must take into acount that token continue to arrive while the burst is being transmitted:
  - The output burst contains a maximum of $C + RS$ bytes.
  - The maximum number of bytes transmitted can also be written as MS.
  - Thus, we have $C + RS = MS$. Solving this equation to get S leads to:

$$S = \frac{C}{(M - R)}$$

- To fully understand how a token bucket behaves, it is also necessary to know how many tokens are initially in the token bucket.

Token Bucket Example

- Consider a token bucket with a token capacity of $C = 3$ tokens and a token arrival rate of $R = 1$ token per ms.
- The buffer used to hold packets can accommodate up to $B = 4$ packets.
- Assume that the system has been idle for some time before the following sequence of packets arrives.
- Compute for the packet arrival times $\{1, 1.1, 1.5, 2, 2.7, 2.9, 3, 3.1, 3.2, 6\}$
  - the departure times;
  - the number of tokens in the bucket;
  - the list of queued packets.

Token Bucket Example (cont.)

Packet	Arrival Time	Departure Time	Token Count	Queued Packets
1	1	1	2	$\{\}$
2	1.1	1.1	1	$\{\}$
3	1.5	1.5	0	$\{\}$
4	2	2	0	$\{\}$
5	2.7	3	0	$\{5\}$
6	2.9	4	0	$\{5, 6\}$
7	3	5	0	$\{6, 7\}$
8	3.1	6	0	$\{6, 7, 8\}$
9	3.2	7	0	$\{6, 7, 8, 9\}$
10	6	8	0	$\{9, 10\}$

Token and Leaky Bucket Combination

- Token buckets still allow some bursts, even though the maximum burst interval can be regulated by careful selection of the parameters.
- It is often desirable to further control the traffic peaks.
- Token buckets can be combined with leaky buckets to address this problem:
  - The token bucket does the primary traffic shaping.
  - The leaky bucket takes care of any remaining peaks.
- Policing such combined mechanisms can be tricky.
- Requires a good understanding of the actual traffic mix to be effective (requires ongoing measurements).
- Often considered to be part of *traffic engineering*.

Fair Queueing (FQ)

- Fair Queuing Idea:
  - Introduce a separate queue for each flow and each output interface.
  - Process queues in a round-robin fashion.
- Problem:
  - Packets have different sizes which can lead to unfairness.
- Solution:
  - Compute the time when a packet will be finished using byte-by-byte round robin.
  - Transmit packets in the order of their finishing times.

Fair Queuing Example (A. Tanenbaum)

- Assume the packets A (6 bytes), B (4 bytes), C (2 bytes), D (4 bytes), E (4 bytes) arrive simultaneously at a fair queuing interface.
- Compute the finish times for these five packets.
- Solution:

```
A -> 01 06 11 15 19 20
B -> 02 07 12 16
C -> 03 08
D -> 04 09 13 17
E -> 05 10 14 18
```

- Finish times: C (8), B (16), D(17), E(18), A (20)

Weighted Fair Queueing (WFQ)

- Weighted Fair Queueing Idea:
  - Introduce priorities so that some queues are processed more often.
  - Give some queues more than one byte per tick.
- Imlementation:
  - Compute the finish times and insert packets into a priority queue sorted by finish times.
  - Take the relative weight of the queues into account when computing the finish times.

WFQ Example (H. Schulzrinne)

- A router with $N = 3$ queues uses weighted fair queueing (WFQ). The weights of the queues are $w_1 = 0.5$, $w_2 = 0.25$, $w_3 = 0.25$.
- Packets in the first queue are 100 byte long while packets in the second and third queue are 300 byte long.
- Assume that the buffer for each queue is full and the first packet in the second queue arrived shortly after the first packet of the first queue and the first packet of the third queue arrived shortly after the first packet of the second queue.
- In which order will the WFQ scheduler serve the packets?

WFQ Example (cont.)

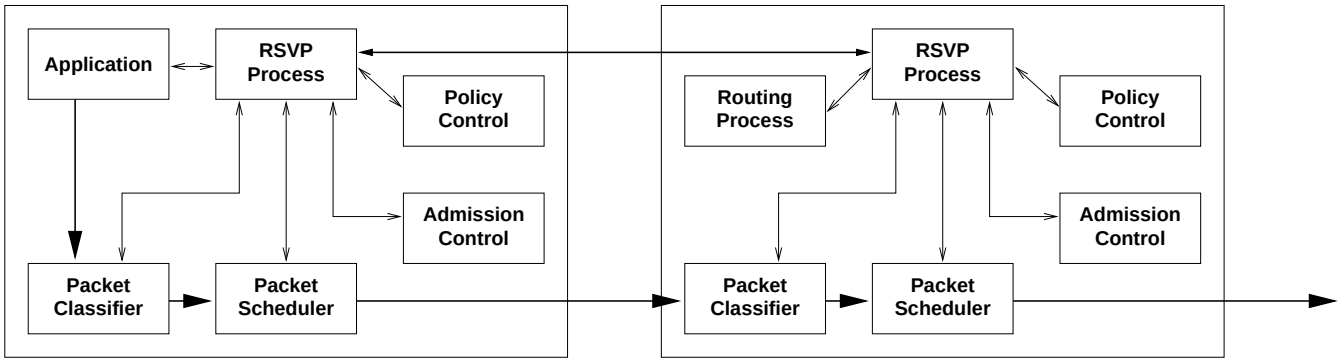
- The smallest common time slot is the time needed to transmit 100 bytes.
- The first stream needs to get half of the slots while the other two streams need to get 3 out of 12 slots each (note that packets in these two classes have a length of 3 slots).

	1	2	3	4	5	6	7	8	9	10	11	12
Q1	50	**100**	150	**200**	250	**300**	350	**400**	450	**500**	550	**600**
Q2	25	50	75	100	125	150	175	200	225	250	275	**300**
Q3	25	50	75	100	125	150	175	200	225	250	275	**300**

- The resulting class sequence is $\{1, 1, 1, 1, 1, 1, 2, 3, \ldots\}$.

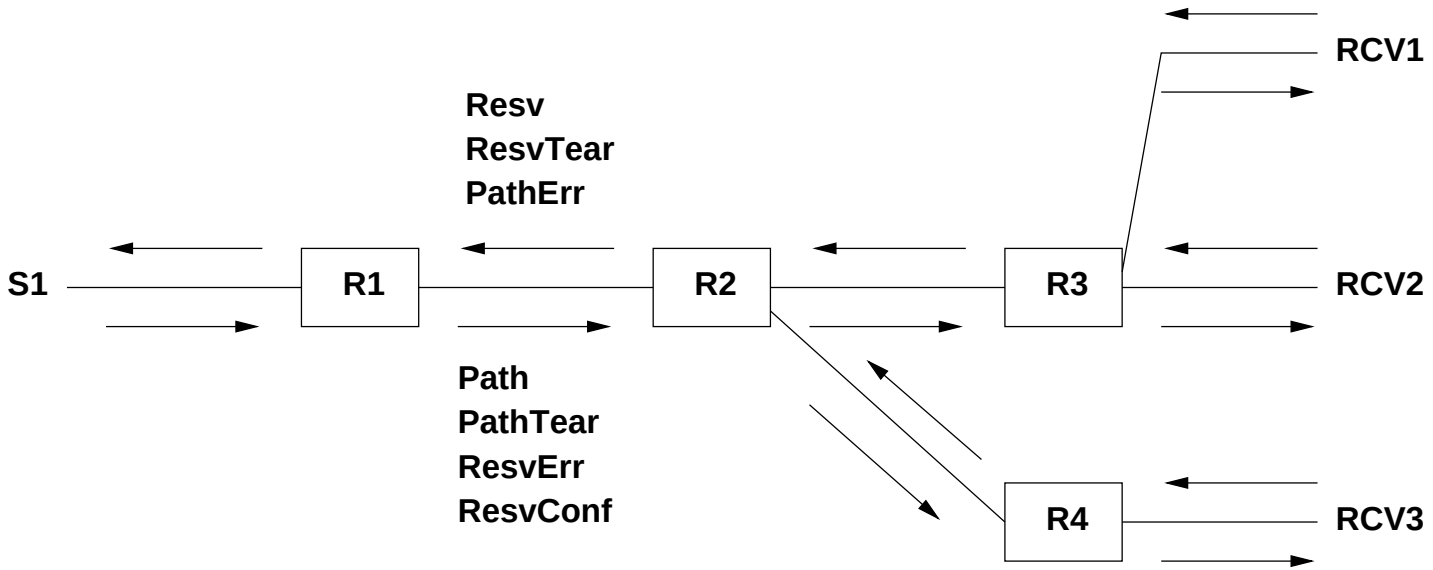
Integrated Services

Integrated Services (RFC 1633)



- Motivation: Provide a framework for service guarantees to support applications which do not function with the Internet's best effort service model.
- Requires to introduce signalling and state in the core routing infrastructure.
- Policies are required to control the reservation and admission decisions.

Ressource Reservation Protocol (RSVP)



- QoS signalling protocol defined in RFC 2205.
- Flow-based Quality of Service (QoS).
- Routers may implement QoS by mapping to lower-layer QoS mechanisms.

RSVP Characteristics

- *Unicast and multicast*:
  RSVP is simplex, i.e., it makes reservations for unidirectional unicast or multicast data flows.
- *Soft state*:
  Reservation state is created and must be periodically refreshed. If routing changes, the RSVP state will timeout and new RSVP state will be installed on the new path.
- *Receiver initiated reservations*:
  The receiver of a data flow initiates and maintains the resource reservation used for that flow.
- *Policy control*:
  RSVP transports and maintains traffic control and policy control parameters that are opaque to RSVP.

Guaranteed Service (RFC 2212)

- The end-to-end delay bound is given by:

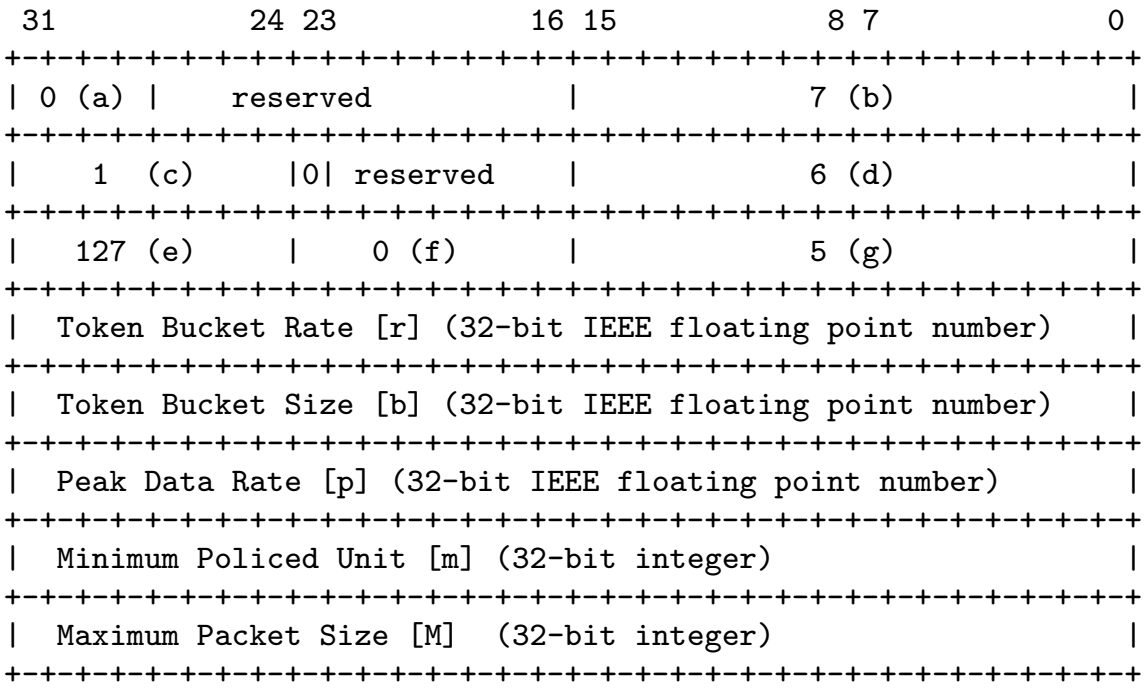
$$Q_{e2ed} = \begin{cases} \frac{b-M}{R}\frac{p-R}{p-r} + \frac{M+C_{tot}}{R+D_{tot}} & \text{for } p > R \geq r \\ \frac{M+C_{tot}}{R+D_{tot}} & \text{for } r \leq p \leq R \end{cases}$$

p	peak rate of flow (bytes/s)
b	bucket depth (bytes)
r	token bucket rate (bytes/s)
m	minimum policed unit (bytes)
M	maximum datagram size (bytes)
R	bandwidth (bytes/s)
S	slack term (s)
C_{tot}	cumulative sum of per hop error terms C
D_{tot}	cumulative sum of per hop error terms D

Controlled Load Service (RFC 2211)

- Controlled-load service provides the client data flow with a quality of service closely approximating the QoS that same flow would receive from an unloaded network element.
- Uses admission control to assure that this service is received even when the network element is overloaded.
- The important difference relative to best effort service is that controlled load service does not noticeable deteriorate as the network load increases.

Sender Traffic Specification (RFC 2210)

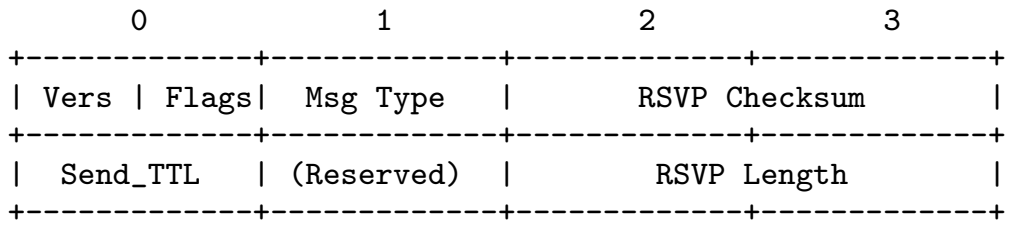


(a) - Message format version number (0)
(b) - Overall length (7 words not including header)
(c) - Service header, service number 1 (default/global information)
(d) - Length of service 1 data, 6 words not including header
(e) - Parameter ID, parameter 127 (Token_Bucket_TSpec)
(f) - Parameter 127 flags (none set)
(g) - Parameter 127 length, 5 words not including header

RSVP Message Formats

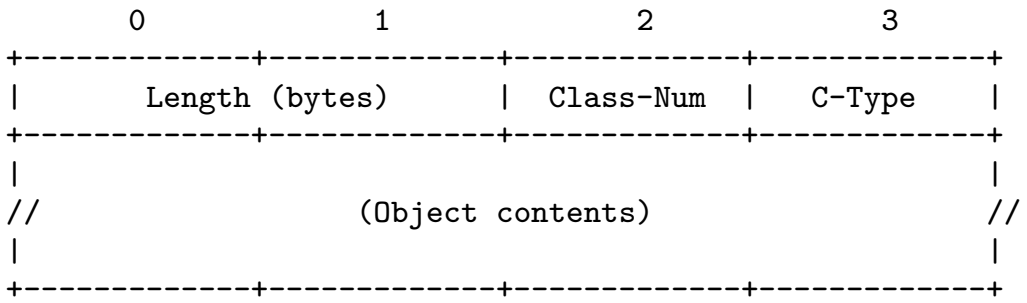
- An RSVP message consists of a common header.
- The body following the header consists of a variable number of variable-length, typed "objects".
- The permissible choice of object types is defined using an augmented Backus-Naur Form (BNF).

RSVP Common Header



- The `Vers` field contains the RSVP version (currently 1), the `Flags` field is currently unused, and the `Checksum` field contains the Internet checksum.
- The `Type` field identifies the RSVP message type and the `Length` field the overall length of the RSVP message.
- The `TTL` field contains the original TTL value.

RSVP Object Format



- Every object is encoded using one or more 32-bit words.
- The `Length` field contains the total length of an object.
- The `Class` field identifies the object's class while the `Type` field identifies the object's type within its class.

RSVP Messages

- Path message (periodically sent by sender):

```
<Path> ::= <Common Header> [ <INTEGRITY> ]
    <SESSION> <RSVP_HOP> <TIME_VALUES>
    [ <POLICY_DATA> ... ] [ <sender descriptor> ]

<sender descriptor> ::= <SENDER_TEMPLATE>
    <SENDER_TSPEC> [ <ADSPEC> ]
```

- PathTear message (sent by sender or router):

```
<PathTear> ::= <Common Header> [ <INTEGRITY> ]
    <SESSION> <RSVP_HOP> [ <sender descriptor> ]

<sender descriptor> ::= (see earlier definition)
```

RSVP Messages

- Resv message (sent by receivers)

```
<Resv> ::= <Common Header> [ <INTEGRITY> ]
    <SESSION>  <RSVP_HOP> <TIME_VALUES>
    [ <RESV_CONFIRM> ]  [ <SCOPE> ]
    [ <POLICY_DATA> ... ] <STYLE>
    <flow descriptor list>

<flow descriptor list> ::= <empty> |
    <flow descriptor list> <flow descriptor>
```

  - The content of the `flow descriptor` depends on the reservation style.

RSVP Messages

- ResvTear message (sent by receiver or router):

```
<ResvTear> ::= <Common Header> [<INTEGRITY>]
    <SESSION> <RSVP_HOP> [ <SCOPE> ] <STYLE>
    <flow descriptor list>

<flow descriptor list> ::= (see earlier definition)
```

- ResvConf message (sent by sender)

```
<ResvConf> ::= <Common Header> [ <INTEGRITY> ]
    <SESSION> <ERROR_SPEC> <RESV_CONFIRM>
    <STYLE> <flow descriptor list>

<flow descriptor list> ::= (see earlier definition)
```

RSVP Messages

- PathErr (sent by router):

```
<PathErr> ::= <Common Header> [ <INTEGRITY> ]
    <SESSION> <ERROR_SPEC>
    [ <POLICY_DATA> ...] [ <sender descriptor> ]

<sender descriptor> ::= (see earlier definition)
```

- ResvErr (sent by router):

```
<ResvErr> ::= <Common Header> [ <INTEGRITY> ]
    <SESSION>  <RSVP_HOP> <ERROR_SPEC>  [ <SCOPE> ]
    [ <POLICY_DATA> ...] <STYLE> [ <error flow descriptor> ]
```

  - The content of the `error flow descriptor` depends on the reservation style.

Integrated Services over IEEE 802

- Explains how RSVP reservations can be mapped to 802 link layer technologies.
- Defined in RFC 2815 and RFC 2816.

RSVP Critique

- RSVP requires that routers maintain state for every flow.
- Hence, RSVP does not scale with the number of flows.
- Are per-flow reservations practical in the Internet?
- RSVP only supports QoS signalling.
- Additional signalling needed (drilling holes into firewalls).
- IETF later defined the "General Internet Signaling Transport" (GIST) in RFC 5971, which is the core protocol of the "Next Steps in Signalling" (NSIS) framework defined in RFC 4080.
- For an introduction to NSIS and a comparison to RSVP, see [**?**].

Differentiated Services

Differentiated Services (RFC 2475)

- Goal: Scalability by aggregating traffic classification state which is conveyed by means of IP-layer packet marking.
- Packets are classified and marked to receive a particular per-hop forwarding behavior on nodes along their path.
- Sophisticated classification, marking, policing, and shaping operations need only be implemented at network boundaries or hosts.
- Network resources are allocated to traffic streams by service provisioning policies which govern
  - how traffic is marked and conditioned upon entry to a differentiated services-capable network, and
  - how that traffic is forwarded within that network.

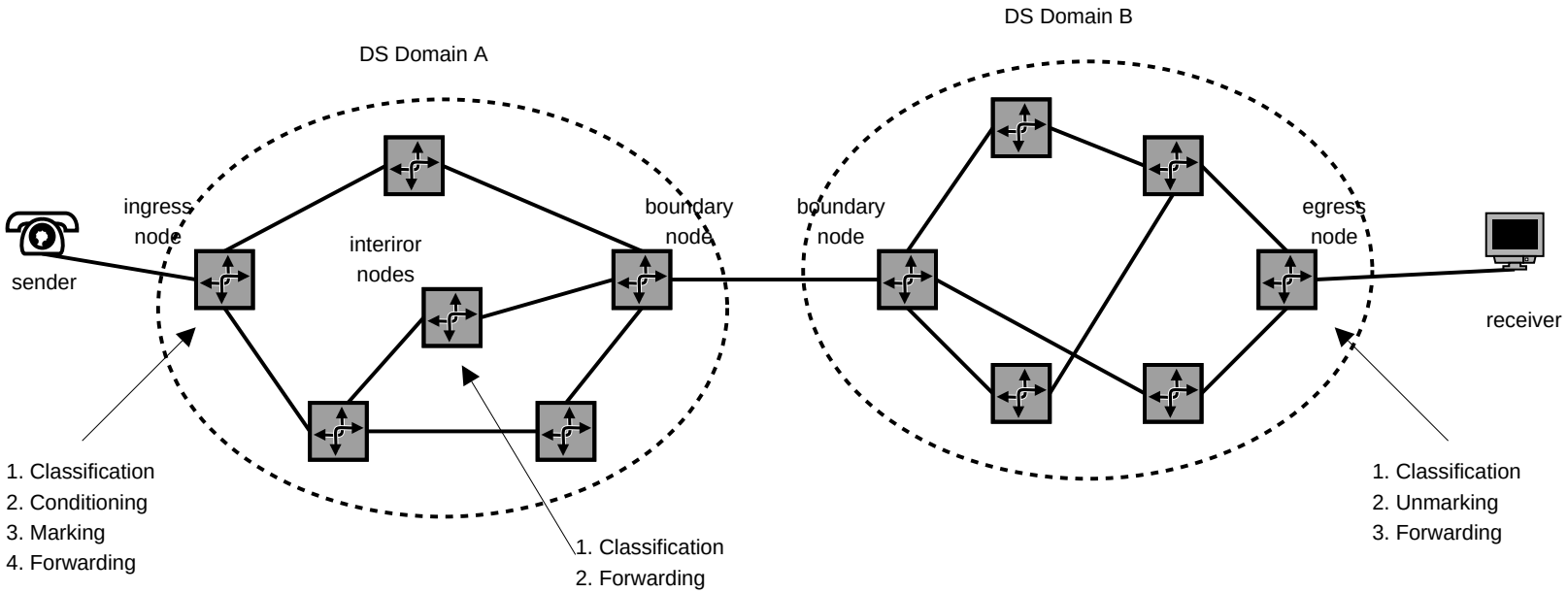
DS Code Point (RFC 2474)

- Two mechanisms are used to achieve scalability:
  1. Aggregation of related flows into service classes
  2. Reservations are provisioned for a longer period
- Packets are tagged when they enter a DiffServ domain using a 6-bit Differentiated Services Code Point (DSCP).
- The DSCP is a value carried in a DS field, which is either in
  - the Type of Service field of an IPv4 packet, or
  - the Traffic Class field of an IPv6 packet.
- Some of the 2^6 possible DSCP values have special usages (see RFC 2474).

Terminology

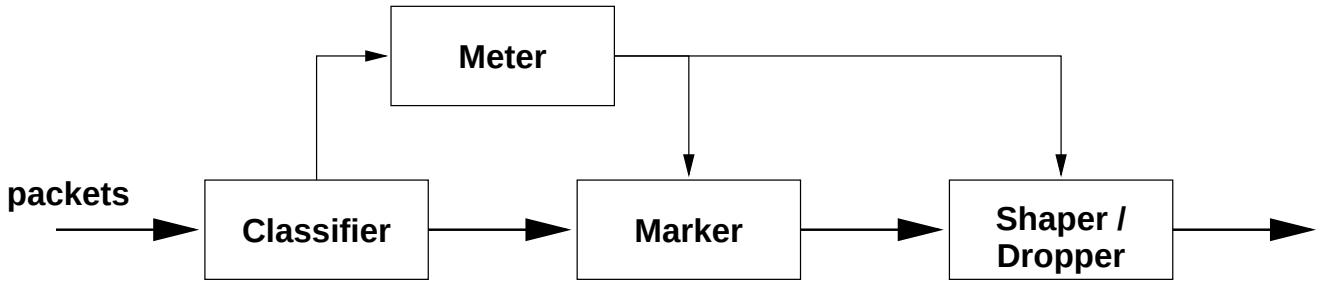
- *DS Domain*: a contiguous set of nodes which operate with a common set of service provisioning policies and PHB definitions
- *DS Ingress Node*: a node handling traffic as it enters a DS domain
- *DS Egress Node*: a node handling traffic as it leaves a DS domain
- *DS Behavior Aggregate*: a collection of packets with the same DS codepoint crossing a link in a particular direction.
- *Per-Hop-Behavior (PHB)*: the externally observable forwarding behavior applied at a DS-compliant node to a DS behavior aggregate.

Differentiated Services Domains



- Most of the complexity is moved to ingress nodes.
- Cooperating DS domains can form a DS region by establishing Traffic Conditioning Agreements (TCAs).

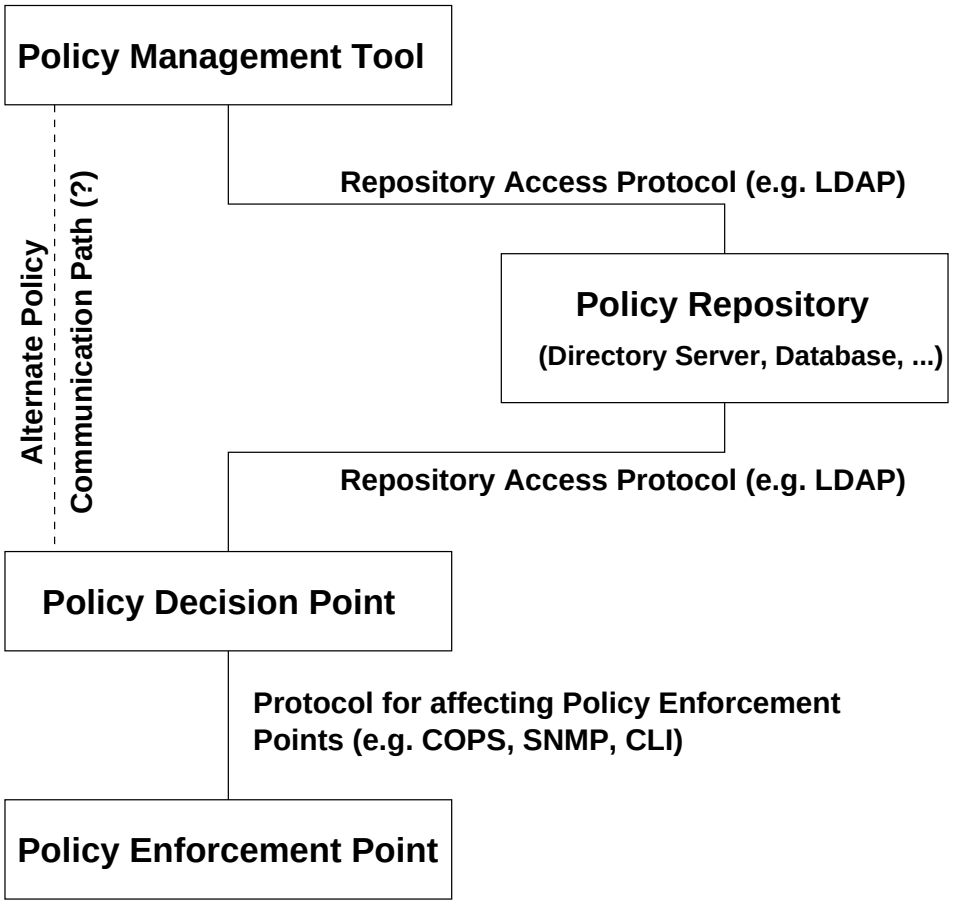
Traffic Classifier and Conditioner



- *Classifier*: selects packets based on the content of packet headers according to defined rules.
- *Marker*: a device that sets DS codepoints in packets
- *Meter*: a device that performs metering
- *Shaper/Dropper*: a device that shapes a packet stream and/or drops packets

Policy Management

Policy Management Framework

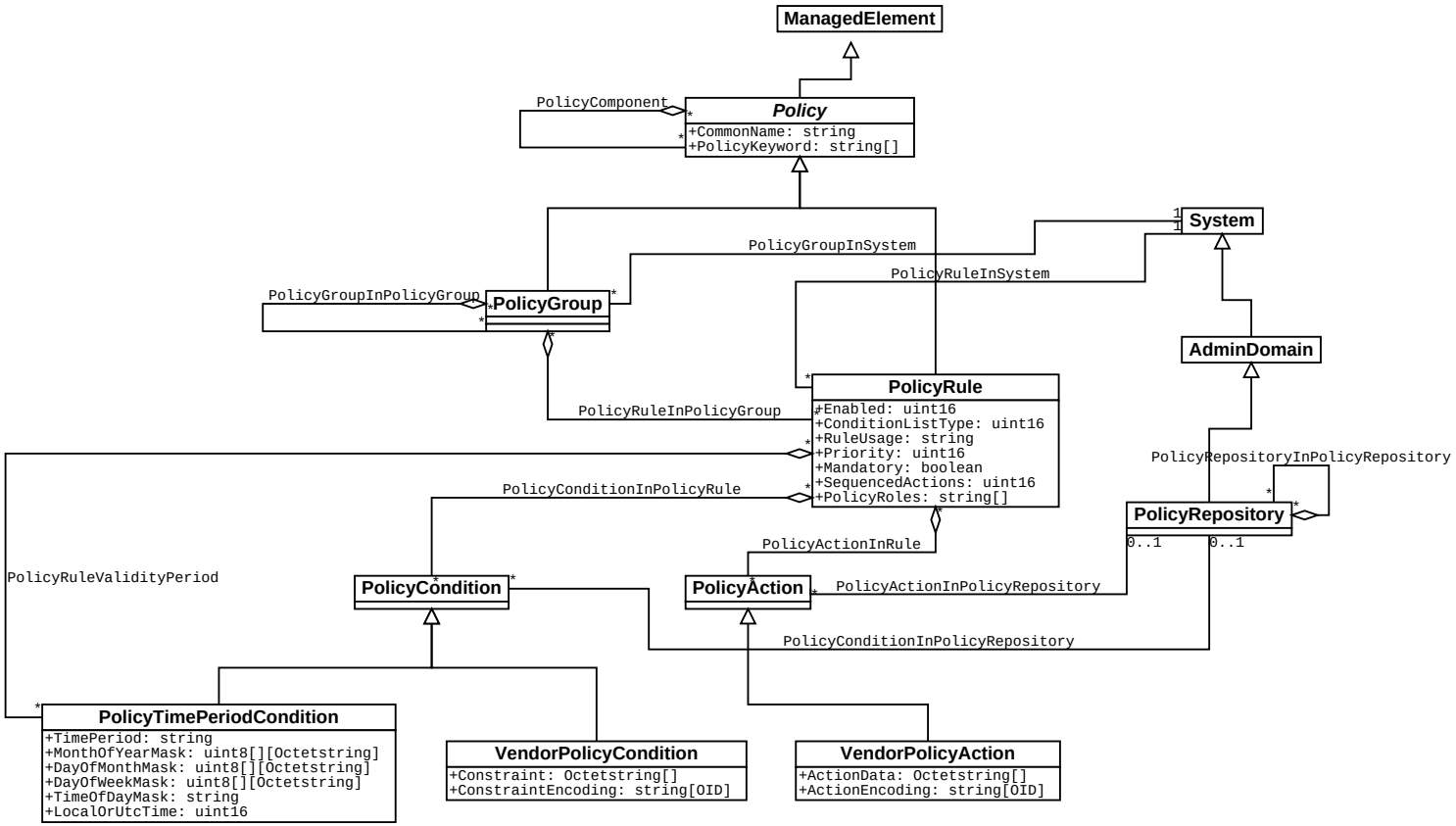


- Policy terminology is defined in RFC 3198.

Policy Core Information Model (PCIM)

- Policy Core Information Model (RFC 3460) is intended to serve as an extensible class hierarchy for defining policy objects representing policies of different types.
- Design of the Policy Core Information Model is influenced by a declarative, not procedural approach.
- Each policy rule consists of a set of conditions and a set of actions.
- The set of conditions associated with a policy rule can be in Disjunctive Normal Form (DNF) or Conjunctive Normal Form (CNF).
- For the set of actions associated with a policy rule, it is possible to specify an order of execution.
- Policy rules can be prioritized and aggregated into policy groups. Policy groups may be nested to represent a hierarchy of policies.

UML Diagram of the PCIM



PCIM Mapping to LDAP

- Structural classes:
  - Represent policy information and control of policies.
- Association classes:
  - Indicate how instances of the structural classes are related to each other.
- Mapping of structural classes:
  - PCIM classes are mapped to LDAP classes.
  - PCIM properties map to LDAP attributes.
- Mapping of association classes:
  - Partly mapped to LDAP
    - auxiliary classes,
    - attributes representing DN pointers,
    - containment in the DIT.

QoS PCIM Extensions

- Work continues on generic QoS extensions of the PCIM.
- Work continues on network device specific extensions of the QoS extension of PCIM.
- All the PCIM extensions are mapped (similar to PCIM itself) to LDAP schemas.

$\Rightarrow$ Strict top-down approach to define a policy class hierarchy.

$\Rightarrow$ Sometimes conflicts with the usual IETF way of working bottom-up.

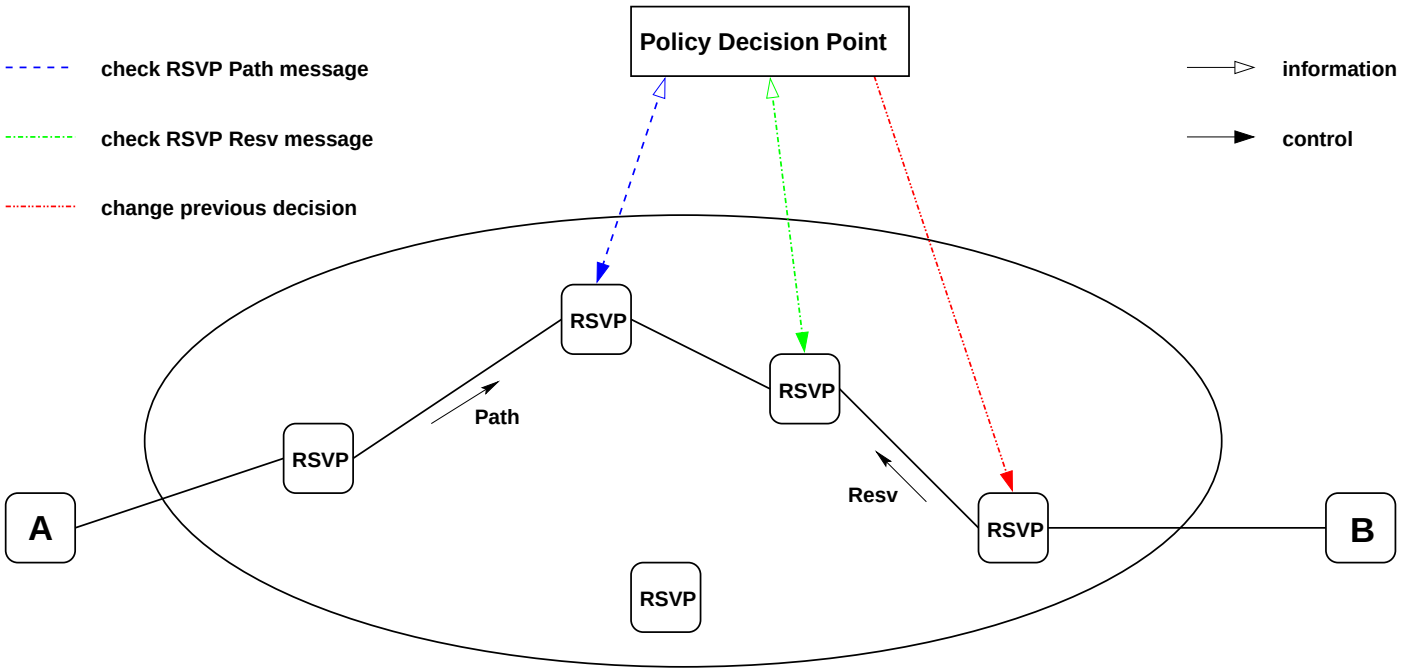
Common Open Policy Service (COPS)

- COPS (RFC 2748) is a client/server protocol between a policy decision point (PDP) and policy enforcement points (PEPs).
- Persistent TCP connections (well known port 3288).
- COPS messages contain sequences of COPS objects.
- Extensibility through self-identifying COPS objects.
- PDPs and PEPs can share state as long as the underlying TCP connection exists.
- PEP is responsible to establish a connection to its PDP.
- Optional message level security for authentication, replay protection, and message integrity through integrity objects.
- IPsec or TLS may be used for encryption.

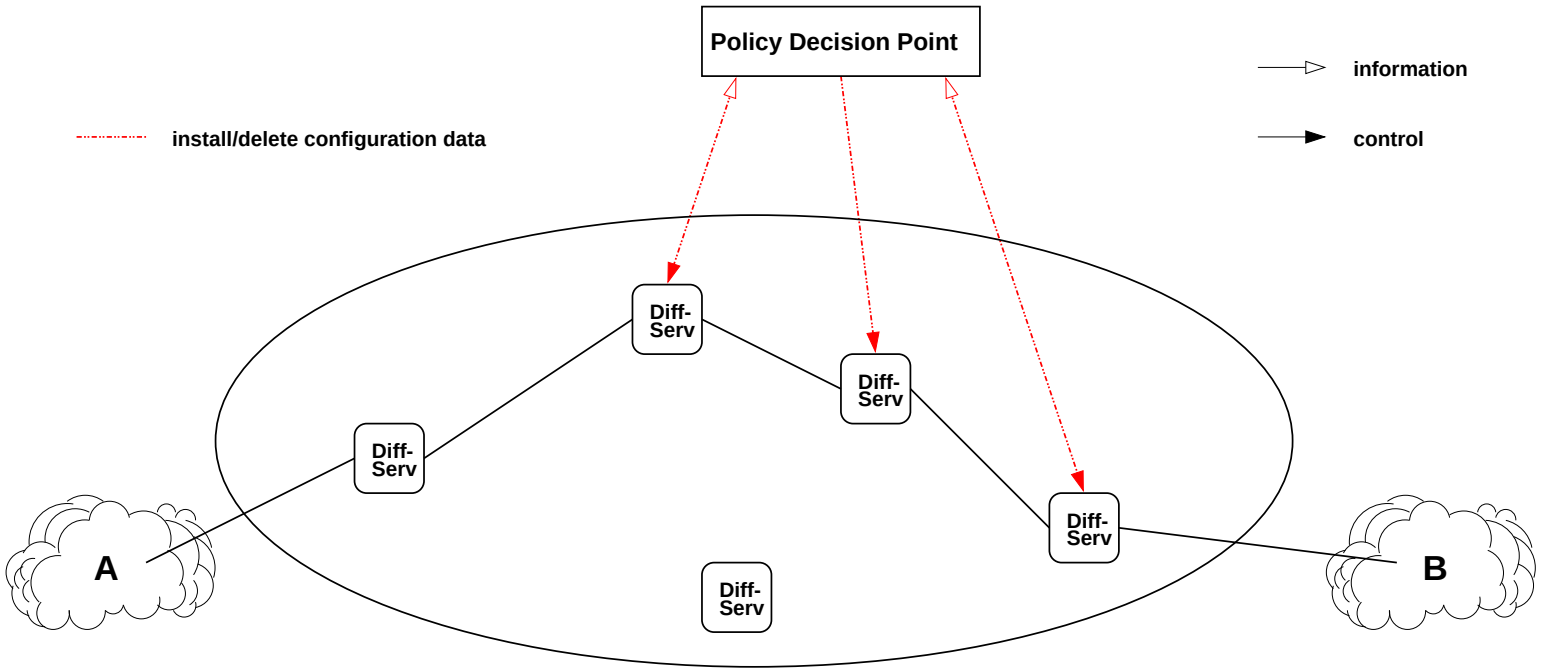
Outsourcing versus Provisioning

- Outsourcing Model:
  - Decisions are made event-driven during the signaling phase.
  - Additional asynchronous decisions from the policy-based management system.
  - Applicable where scalability is not a big issue.
- Provisioning Model:
  - Provisioning of all necessary configuration information to enforce policies locally.
  - Provisioning information is defined in a Policy Information Base (PIB).
  - PIBs are defined using the Structure of Policy Provisioning Information (SPPI).
  - No real-time policy interactions, highly scalable.

Outsourcing Model (RSVP)



Provisioning Model (DiffServ)



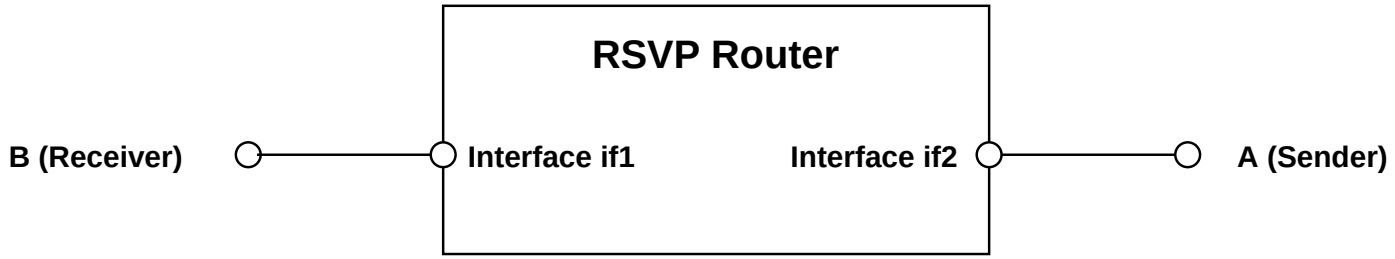
COPS Operation Types (RFC 2748)

Operation	Description	Direction
REQ	Request	PEP $\rightarrow$ PDP
DEC	Decision	PDP $\rightarrow$ PEP
RPT	Report State	PEP $\rightarrow$ PDP
DRQ	Delete Request State	PEP $\rightarrow$ PDP
SSQ	Synchronize State Request	PDP $\rightarrow$ PEP
SSC	Synchronize State Complete	PEP $\rightarrow$ PDP
OPN	Client-Open	PEP $\rightarrow$ PDP
CAT	Client-Accept	PDP $\rightarrow$ PEP
CC	Client-Close	PEP $\rightarrow$ PDP, PDP $\rightarrow$ PEP
KA	Keep-Alive	PEP $\rightarrow$ PDP, PDP $\rightarrow$ PEP

COPS for RSVP (RFC 2749)

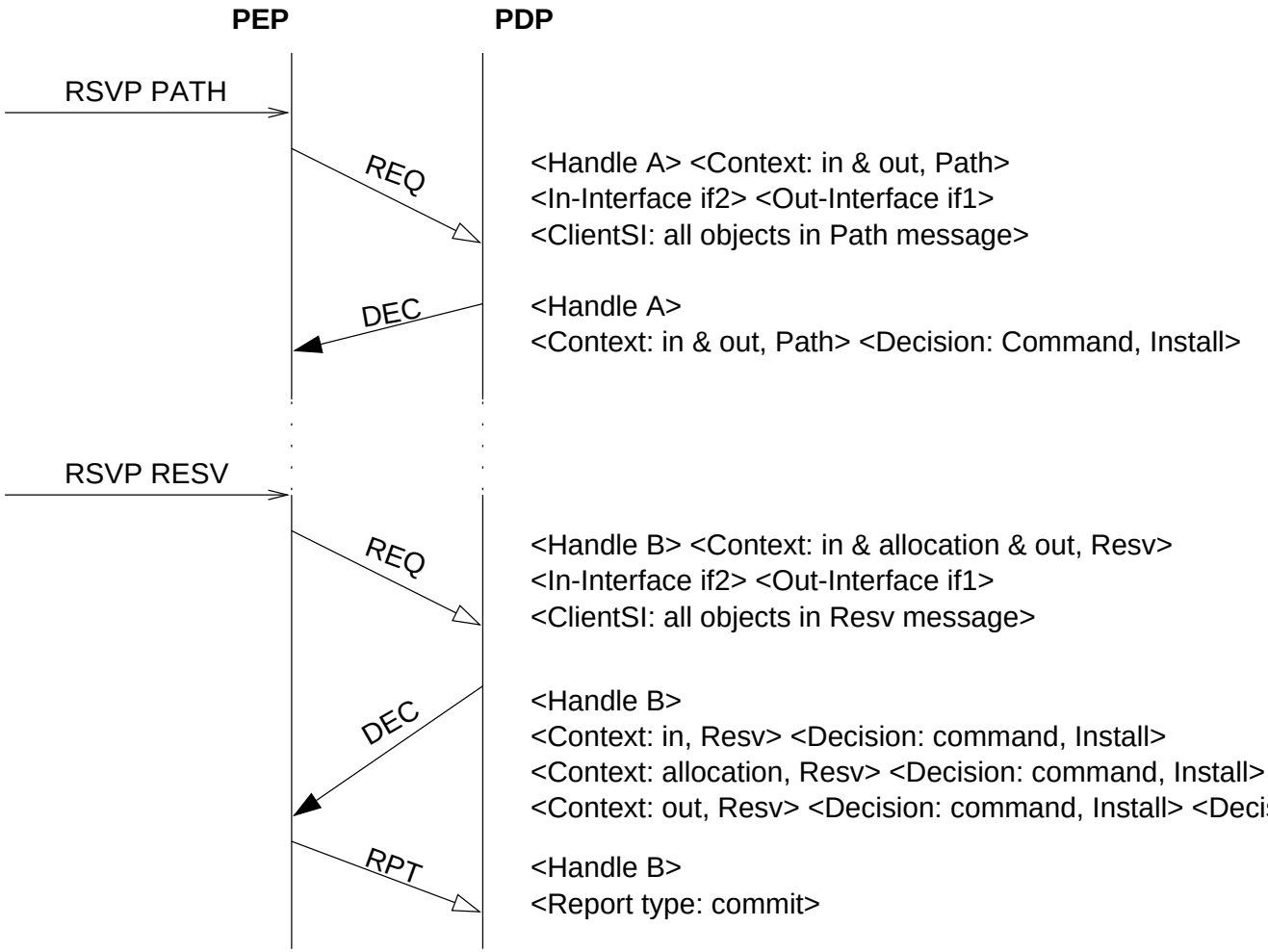
- All objects received in an RSVP message are encapsulated inside the COPS Client Specific Information Object (ClientSI) send from the PEP to the PDP.
- The PEP and PDP share RSVP state.
- Install decision command:
  - Accept/Allow/Admit an RSVP message or local resource allocation.
- Remove decision command:
  - Deny/Reject/Remove an RSVP message or local resource allocation.
- PEP may cache decisions in order to use them for a given time interval while the connection between the PEP and its PDP is lost.

COPS-RSVP Protocol Example

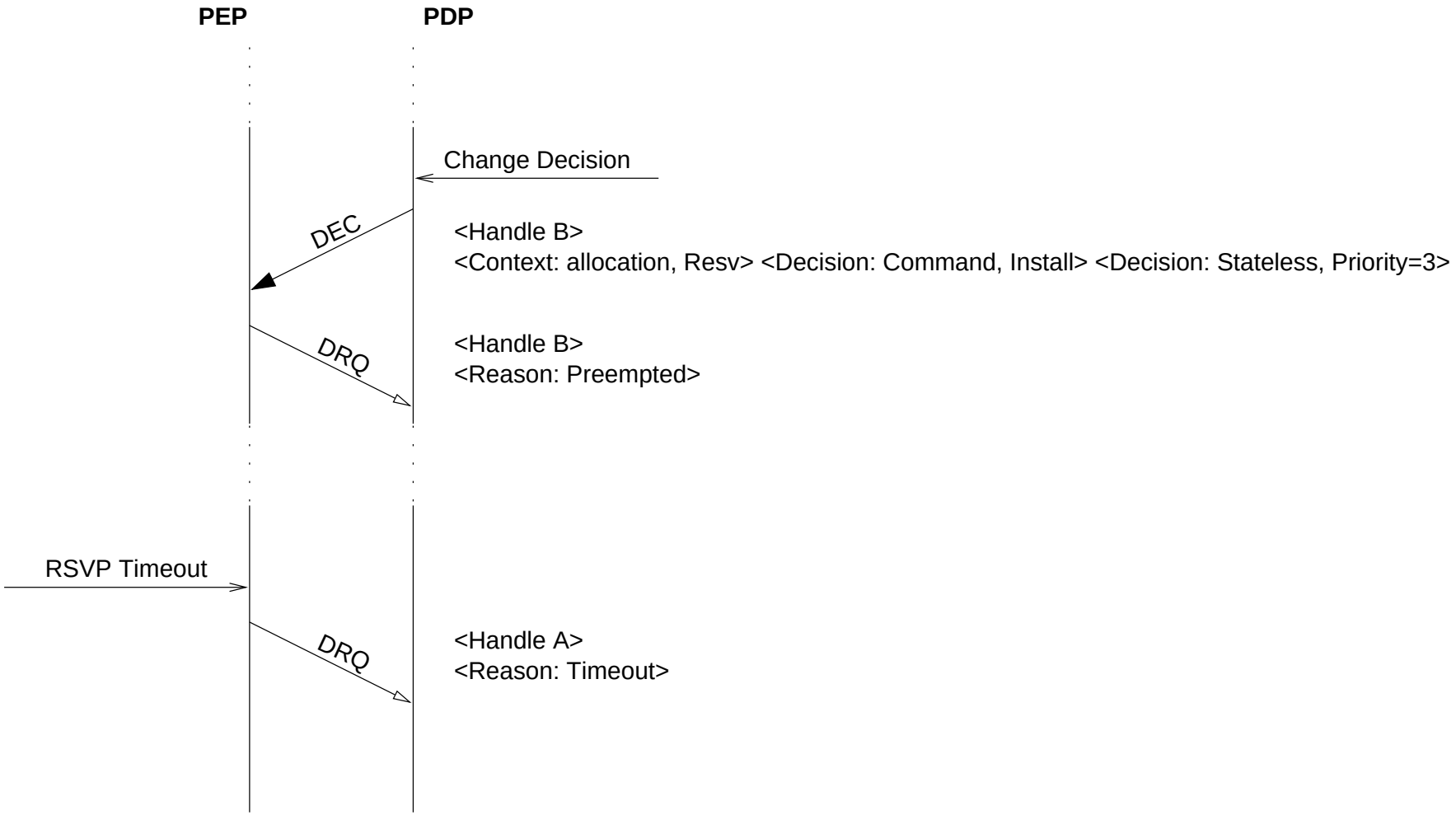


- The PEP router has two network interfaces (if1, if2).
- Sender A sends to receiver B.
- COPS RSVP is used to control the unicast RSVP flow between A and B.

COPS-RSVP Protocol Example



COPS-RSVP Protocol Example



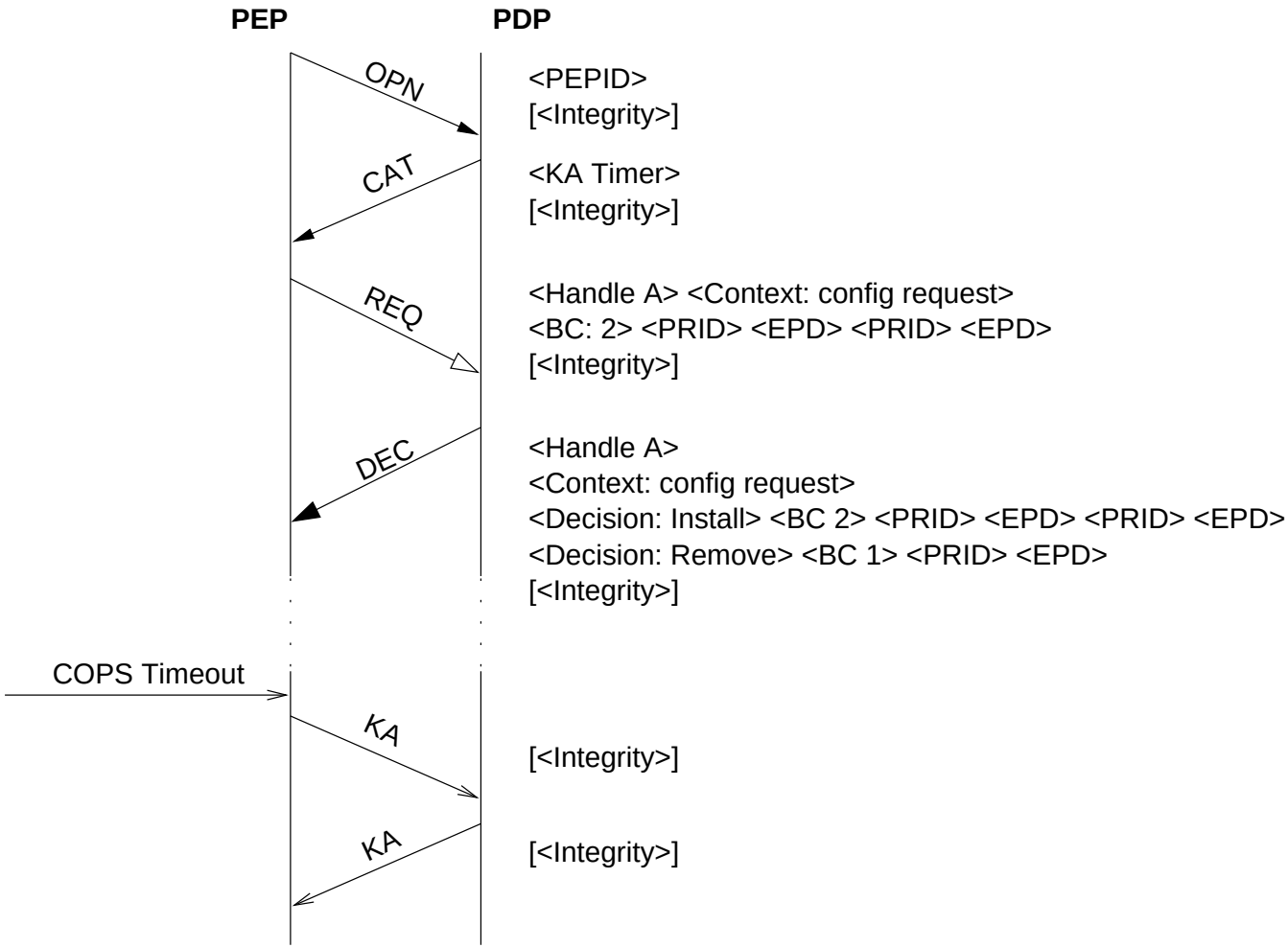
COPS for Provisioning

- Provisioning of policy configurations (a set of policy class instances) at PEPs.
- COPS-PR specific terminology:
  - Policy Rule Class (PRC):
    An ordered set of attributes. PRCs are defined in PIB modules and registered in the Object Identifier tree.
  - Policy Rule Instance (PRI):
    An instantiation of a PRC.
  - Policy Rule Instance Identifier (PRID):
    A positive integer which identifies a PRI of a given PRC.
  - Encoded Policy Instance Data (EPD):
    BER encoded representation of a PRI.

Policy Information Base (PIB)

- A Policy Information Base (PIB) defines a set PRCs for use with COPS-PR.
- PIB modules are written using the Structure of Policy Provisioning Information (SPPI).
- The SPPI is an adapted superset of the SNMP's SMIv2.
- Hooks in the SPPI can be used to generate MIBs from PIBs for usage with SNMP.

COPS-PR Protocol Example



References I

X. Fu, H. Schulzrinne, A. Bader, D. Hogrefe, C. Kappler, G. Karagiannis, H. Tschofenig, and S. van den Bosch.
NSIS: A New Extensible IP Signaling Protocol Suite.
IEEE Communications Magazine, 43(10):133–141, October 2005.

R. Braden, D. Clark, and S. Shenker.
Integrated Services in the Internet Architecture: an Overview.
RFC 1633, ISI, MIT, Xerox PARC, June 1994.

R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin.
Resource ReSerVation Protocol (RSVP) Version 1 Functional Specification.
RFC 2205, ISI, UCLA, IBM Research, Univ. of Michigan, September 1997.

J. Wroclawski.
The Use of RSVP with IETF Integrated Services.
RFC 2210, MIT LCS, September 1997.

J. Wroclawski.
Specification of the Controlled-Load Network Element Service.
RFC 2211, MIT LCS, September 1997.

S. Shenker, C. Partridge, and R. Guerin.
Specification of Guaranteed Quality of Service.
RFC 2212, Xerox, BBN, IBM, September 1997.

S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss.
An Architecture for Differentiated Services.
RFC 2475, Torrent Networking Technologies, EMC Corporation, Sun Microsystems, Nortel UK, Bell Labs Lucent Technologies, Lucent Technologies, December 1998.

References II

K. Nichols, S. Blake, F. Baker, and D. Black.

Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers.
RFC 2474, Cisco Systems, Torrent Networking Technologies, EMC Corporation, December 1998.

Y. Bernet, S. Blake, D. Grossman, and A. Smith.

An Informal Management Model for Diffserv Routers.
RFC 3290, Microsoft, Ericsson, Motorola, Harbour Networks, May 2002.

B. Moore.

Policy Core Information Model (PCIM) Extensions.
RFC 3460, IBM, January 2003.

Y. Snir, Y. Ramberg, J. Strassner, R. Cohen, and B. Moore.

Policy Quality of Service (QoS) Information Model.
RFC 3644, Cisco Systems, Intelliden, Ntear LLC, IBM, November 2003.

Part: Multimedia Transport and Signaling

Real-time Transport Protocol (RTP)

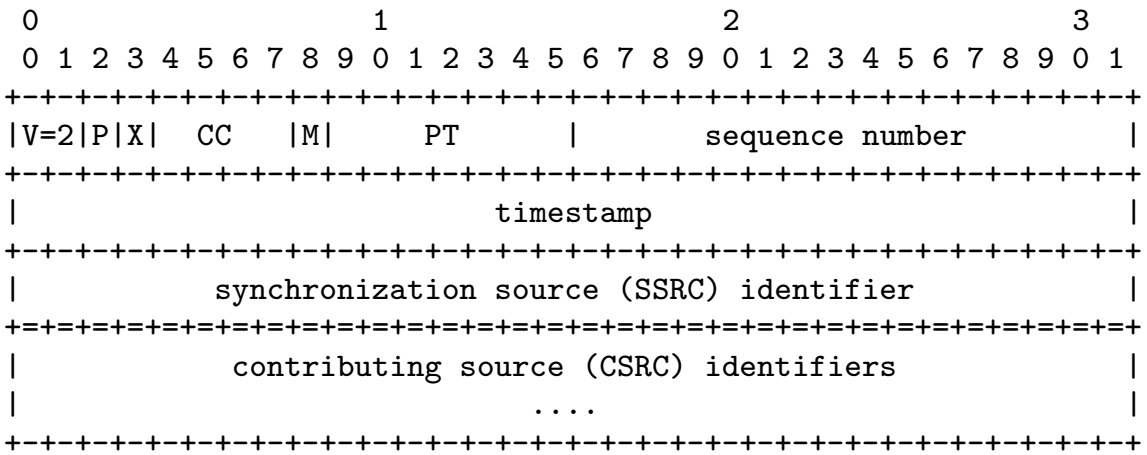
Real-time Transport Protocol (RTP)

- RTP provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services.
- RTP and RTCP are designed to be independent of the underlying transport and network layers (but commonly used over UDP).
- The protocol supports the use of RTP-level translators and mixers.

RTP Elements

- *Synchronisation Source*: A source identified by a 32-bit number which is originating data streams.
- *Mixer*: A component capable of resynchronizing streams, mixing reconstructed streams into a single stream, or translating the encoding of a stream (e.g., from a high-quality encoding to a low-bandwidth encoding).
- *Translator*: A component capable to translate streams in order to cross firewalls or different transports.
- *Receiver*: A component resynchronizing received streams, usually using playout buffers.

RTP Message Format



- RTP is defined in RFC 3550.
- RTP supports multicasting of multimedia streams since it is running over UDP

RTP Message Header

- The `V` fied contains the RTP version number (current version is 2).
- The `X` bit indicates whether there are any extension headers
- The `P` bit indicates that there are padding bytes at the end of the packet. (The last padding byte contains the number of padding bytes.)
- The `CC` field contains the number of CSRC identifiers.
- The `M` bit may be used by profiles that mark certain bytes in the packets.
- The `PT` identifies the format of the RTP payload.

RTP Message Header

- The `sequence number` field contains a sequence number for each packet.
- The `timestamp` field indicates the relative time of the packet in the overall media stream (media timestamp).
- The `SSRC` field identifies the synchronization source.
- The `CSRC` identifiers (if present) identify the additional synchronization sources in cases where multiple media streams have been mixed into a single stream.
- RTP profiles define how RTP is used to transport specific codecs.

RTP Control Protocol (RTCP)

- RTCP allows senders and receivers to transmit a series of reports to one another that contain additional information about
  - the data being transmitted and
  - the performance of the network.
- RTCP packet types:
  - `SR`: Sender report, for transmission and reception statistics from participants that are active senders
  - `RR`: Receiver report, for reception statistics from participants that are not active senders
  - `SDES`: Source description items, including CNAME
  - `BYE`: Indicates end of participation
  - `APP`: Application-specific functions

RTCP Extended Reports (XR)

- XR packets convey information beyond that already contained in the reception report blocks of RTCP's sender report (SR) or Receiver Report (RR) packets.
- Packet-by-packet report blocks:
  - *Loss RLE Report Block*: Run length encoding of reports concerning the losses and receipts of RTP packets.
  - *Duplicate RLE Report Block*: Run length encoding of reports concerning duplicates of received RTP packets.
  - *Packet Receipt Times Report Block*: A list of reception timestamps of RTP packets.

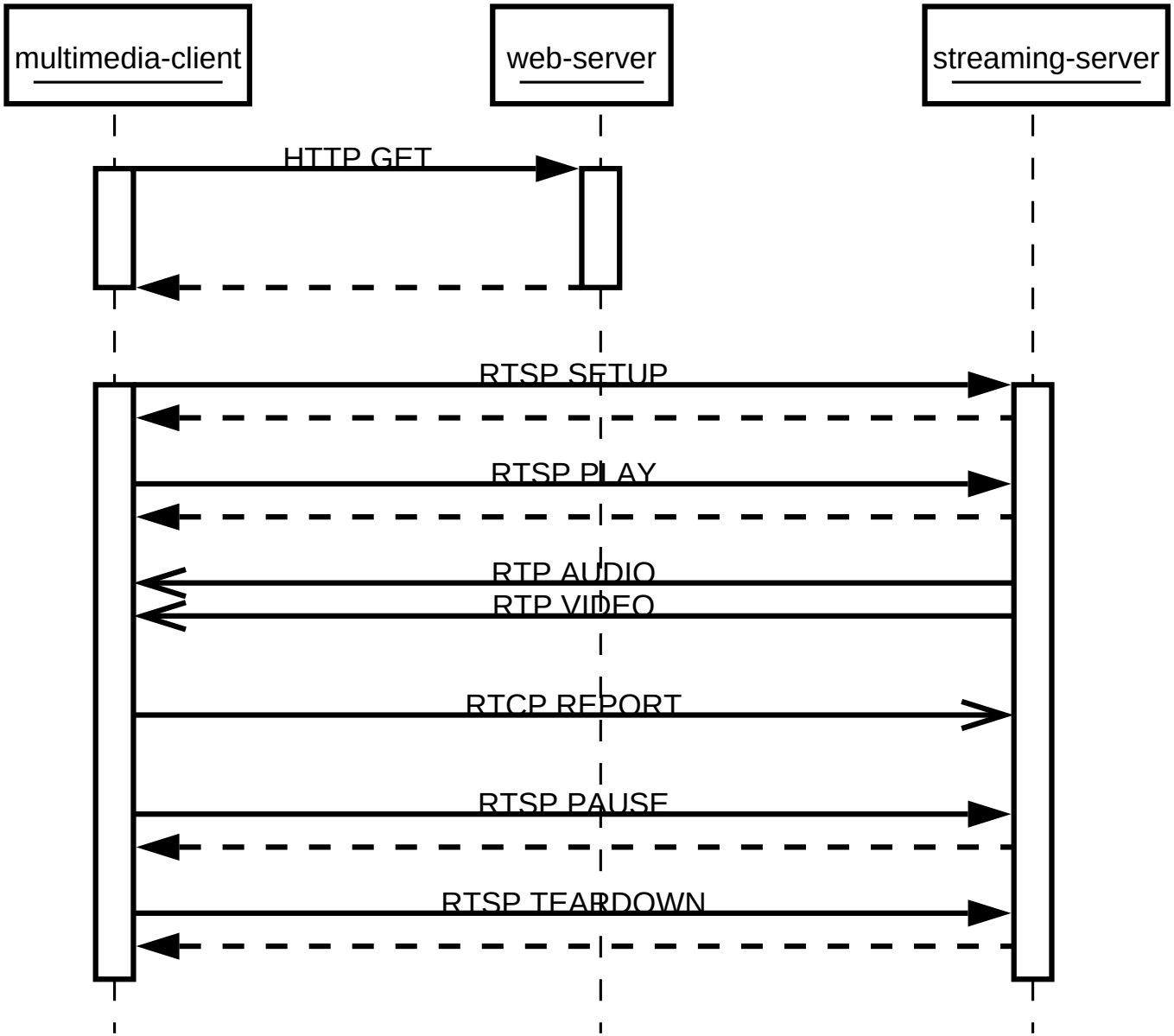
RTCP Extended Reports (XR)

- Reference time report blocks:
  - *Receiver Reference Time Report Block*: Receiver-end wallclock timestamps. Together with the DLRR Report Block mentioned next, these allow non-senders to calculate round-trip times.
  - *DLRR Report Block*: The delay since the last Receiver Reference Time Report Block was received.
- Metric report blocks:
  - *Statistics Summary Report Block*: Statistics on RTP packet sequence numbers, losses, duplicates, jitter, and TTL or Hop Limit values.
  - *VoIP Metrics Report Block*: Metrics for monitoring Voice over IP (VoIP) calls.

Real-Time Streaming Protocol (RTSP)

- The Real-Time Streaming Protocol (RTSP) defined in RFC 2326 establishes and controls either a single or multiple time-synchronized streams of continuous media.
- RTSP acts as a "network remote control" for multimedia servers.
- The RTSP protocol is similar in syntax and operation to HTTP version 1.1.
- RTSP, however, differs fundamentally from HTTP in that data delivery takes place out-of-band in a different protocol.
- While RTSP was written to support RTP, it is not tied to RTP as the real-time media transport protocol.

RTP + RTCP + RTSP + HTTP



RSTP Options

- `OPTIONS` get available methods
- `SETUP` establish transport
- `ANNOUNCE` change description of media object
- `DESCRIBE` get (low-level) description of media object
- `PLAY` start playback, reposition
- `RECORD` start recording
- `REDIRECT` redirect client to new server
- `PAUSE` halt delivery, but keep state
- `SET PARAMETER` device or encoding control
- `GET PARAMETER` device or encoding control
- `TEARDOWN` remove state

Session Description Protocol (SDP)

Session Description Protocol (SDP)

- SDP as defined in RFC 4566 is intended for describing multimedia sessions for the purposes of session announcement, session invitation, and other forms of multimedia session initiation.
- The SDP description format is used by other protocols (such as RSTP).
- A session description includes:
  - Session name and purpose
  - Time(s) the session is active
  - The media comprising the session
  - Information needed to receive the media streams (addresses, ports, formats and so on)

Sample Session Description

```
v=0
o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
e=mjh@isi.edu (Mark Handley)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 31
m=application 32416 udp wb
a=orient:portrait
```

- Description of a session called "SDP Seminar" which is sent to the multicast group 224.2.17.12 and contains three channels (audio, video, whiteboard).
- Start and stop times are indicated in the t= field.

Session Initiation Protocol (SIP)

Session Initiation Protocol (SIP)

- An application layer control (signaling) protocol for creating, modifying, and terminating sessions with one or more participants.
- Sessions include Internet telephone calls, multimedia distribution, and multimedia conferences.
- SIP makes use of elements called proxy servers to help route requests to the user's current location, authenticate and authorize users for services, implement provider call-routing policies, and provide features to users.
- SIP runs on top of several different transport protocols.
- SIP is defined in RFC 3261.

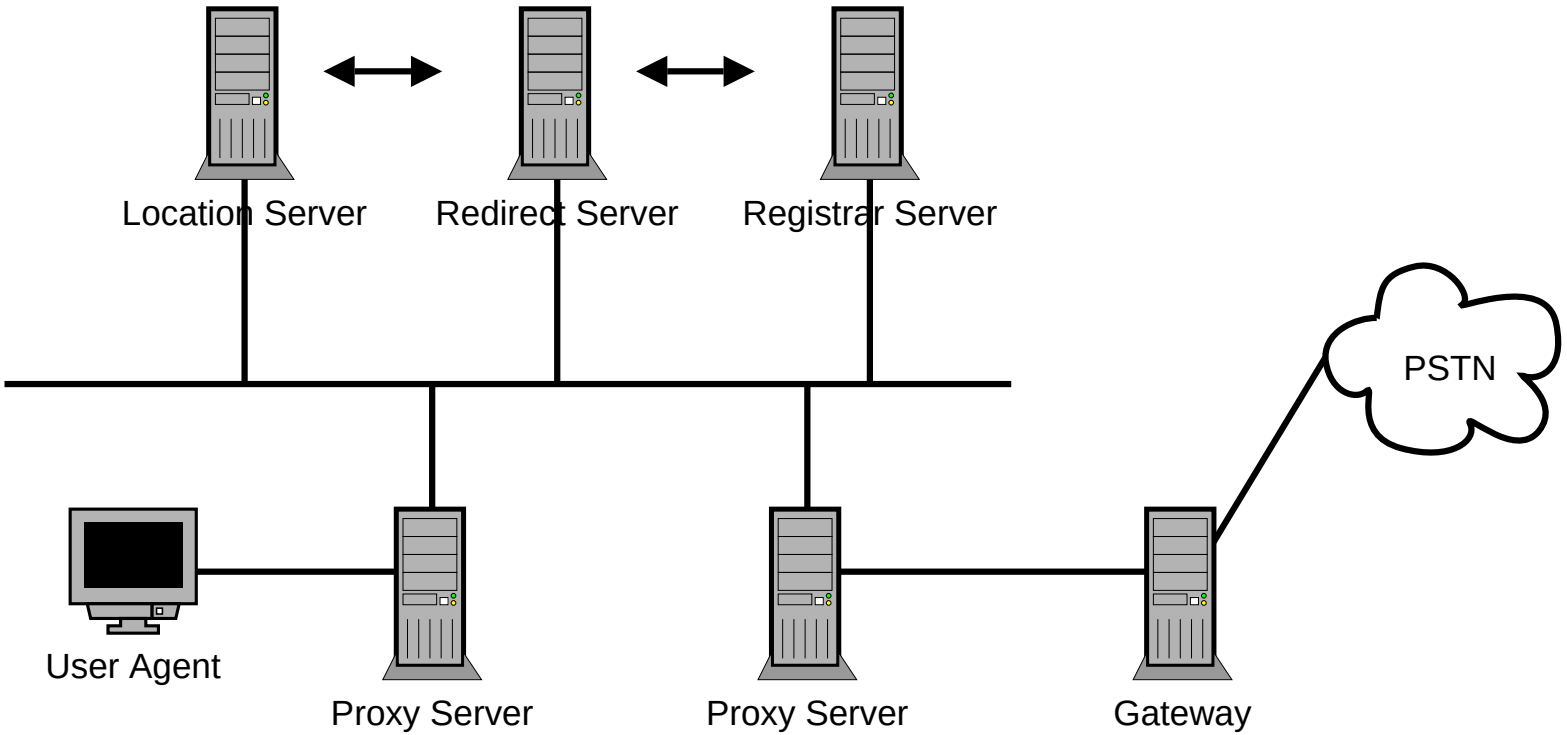
SIP Facets

- *User location*: determination of the end system to be used for communication
- *User availability*: determination of the willingness of the called party to engage in communications
- *User capabilities*: determination of the media and media parameters to be used
- *Session setup*: "ringing", establishment of session parameters at both called and calling party
- *Session management*: including transfer and termination of sessions, modifying session parameters, and invoking services

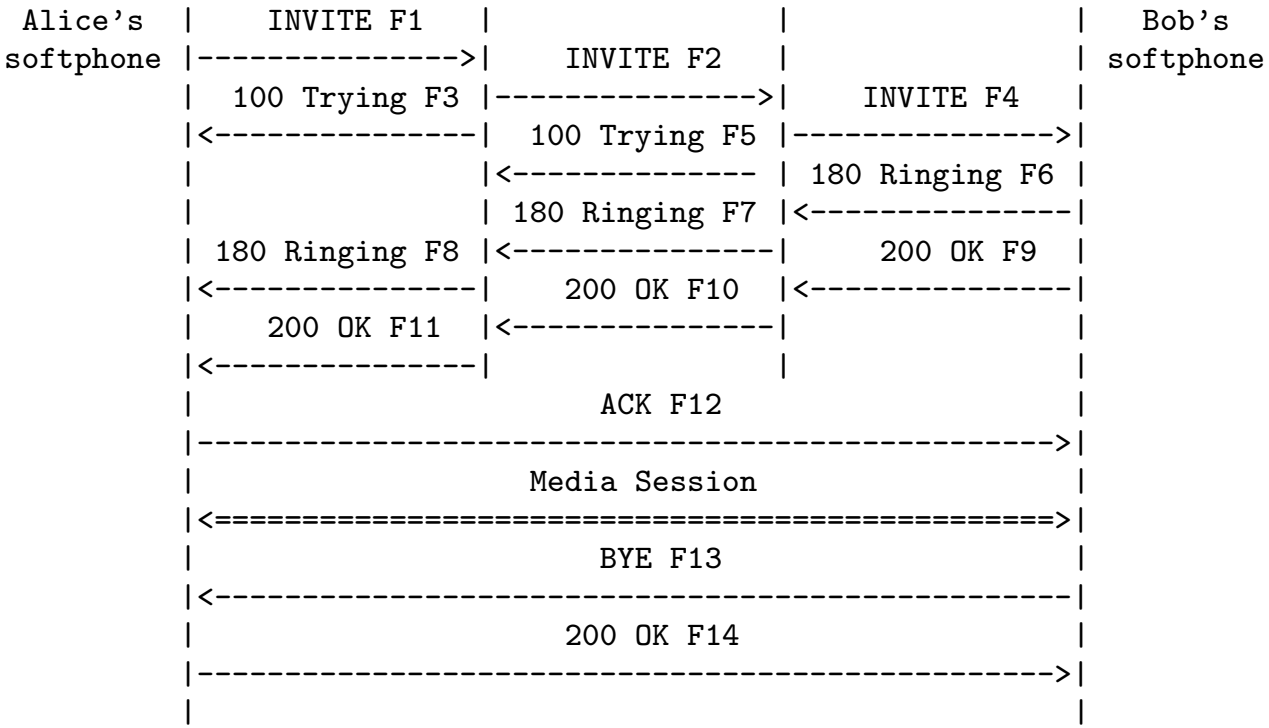
SIP Properties

- HTTP-like textual message format
- HTTP-like method calls and status responses
- Utilizes the Session Description Protocol (SDP) for the description of multimedia sessions
- User agents can initiate and receive calls (session endpoints)
- Proxy server provide an infrastructure to help user agents to establish sessions
- ...

SIP Interworking



SIP Session Setup Example



Alice's INVITE Message

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142

(Alice's SDP not shown)
```

Bob's 200 Response

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP server10.biloxi.com
   ;branch=z9hG4bKnashds8;received=192.0.2.3
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com
   ;branch=z9hG4bK77ef4c2312983.1;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com
   ;branch=z9hG4bK776asdhds ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:bob@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131

(Bob's SDP not shown)
```

SIP URIs

`sip:user:password@host:port;uri-parameters?headers`

- The `user` token identifies a particular resource at the host being addressed.
- The `password` token is a password associated with `user` (usage not recommended).
- The `host` token identifies the host providing SIP resources.
- The `port` number is the port to which a request is to be sent.
- The `uri-parameters` affect the request constructed from the URI.
- The `headers` token specifies the header fields to be included in a request constructed from a URI.

SIP URI Examples

- Typical SIP URI for user alice:

  `sip:alice@atlanta.com`

- The same with an explicit IP address:

  `sip:alice@192.0.2.4`

- The same with a password and an explicti transport:

  `sip:alice:secretword@atlanta.com;transport=tcp`

- SIP URI with an embedded PSTN phone number:

  `sip:+1-212-555-1212:1234@gateway.com;user=phone`

- SIP URI with an explicit method call:

  `sip:atlanta.com;method=REGISTER?to=alice%40atlanta.com`

Locating SIP Servers (RFC 3263)

- Given a SIP URI, how do you find the responsible SIP server?
- Need to determine
  - the transport protocol and
  - the IP address and port number

  of the SIP server or proxy.
- In the general case, it is preferrable to have SIP URIs that belong to a domain rather than a specific host:

  `sip:j.schoenwaelder@jacobs-university.de`
- A mechanism similar to `MX` records for email is needed.
- Could there be a generalized solution?

Transport Selection

```
if <SIP URI specifies transport> {
    <use specified transport>
} else {
    if <host part of the URI is numeric> {
        <use udp for sip: and tcp for sips:>
    } else {
        <lookup a NAPTR DNS record using transport
         selection fields SIP+D2U, SIP+D2T, SIP+D2S>
        if <no NAPTR records available> {
            <construct and perform SRV queries>
        }
        if <still not successfull> {
            <use udp for sip: and tcp for sips:>
        }
    }
}
```

Port and IP Address Selection

```
if <host part of the URI is numeric> {
    <use IP address>
    if <port part of the URI is not empty> {
        <use the port number contained in the URI>
    } else {
        <use the default port number>
    }
} else {
    if <port part of the URI is not empty> {
        <lookup IP addresses using A or AAAA queries>
        <try the addresses with the port number until success>
    } else {
        <lookup a SRV record for the determined transport>
        if <no SRV record available> {
            <lookup IP addresses using A or AAAA queries>
            <try the addresses with the port number until success>
        } else {
            <try the locations specified by the SRV record until success>
        }
    }
}
```

DNS NAPTR and SRV Resource Records

```
;          order pref flags service      regexp  replacement
   IN NAPTR 50   50  "s"  "SIPS+D2T"     ""  _sips._tcp.example.com.
   IN NAPTR 90   50  "s"  "SIP+D2T"      ""  _sip._tcp.example.com.
   IN NAPTR 100  50  "s"  "SIP+D2U"      ""  _sip._udp.example.com.
;           Priority Weight Port   Target
    IN SRV  0        1      5060   server1.example.com
    IN SRV  0        2      5060   server2.example.com
```

- NAPTR records provide a mapping from a domain to the SRV record for contacting a server with the specific transport protocol in the NAPTR services field (RFC 2915).
- SRV records specify the location of server(s) for a specific protocol and domain (RFC 2782).

SIP Methods

- `INVITE`
  - Invites a user to participate in a session (call).
- `ACK`
  - Confirms final response to an `INVITE` request.
- `OPTIONS`
  - Used to query the capabilities of a server.
- `BYE`
  - Indicates termination of a call.
- `CANCEL`
  - Cancels a pending request.
- `REGISTER`
  - Registers a user agent at a proxy.

SIP Status Codes

- 1xx Provisional – request received, continuing to process the request
- 2xx Success – the action was successfully received, understood, and accepted
- 3xx Redirection – further action needs to be taken in order to complete the request
- 4xx Client Error – the request contains bad syntax or cannot be fulfilled at this server
- 5xx Server Error – the server failed to fulfill an apparently valid request
- 6xx Global Failure – the request cannot be fulfilled at any server

SIP Communication Establishment

- Communication establishment is done in six steps:
  1. Registering, initiating and locating the user.
  2. Determine the media to use – involves delivering a description of the session that the user is invited to.
  3. Determine the willingness of the called party to communicate.
  4. Call setup.
  5. Call modification of handling – example, call transfer (opional)
  6. Call termination

SIP Registration

- During startup, a SIP user agent registers with its proxy/registration server.
- Registration can also occur when the SIP user agent moves and the new location needs to be communicated.
- The registration information is periodically refreshed (each SIP user agent has to re-register).
- Typically, the proxy/registration server will forward the location information to the location/redirect server.
- In many cases, the different servers might be co-located.

ENUM and DDDS (RFC 3761)

- ENUM defined in RFC 3761 provides a mechanism to lookup information associated with telephone numbers in the DNS.
- Number conversion:
  1. Remove all characters with the exception of the digits. Example: "+442079460148" $\longrightarrow$ "442079460148"
  2. Put dots (".") between each digit. Example: 4.4.2.0.7.9.4.6.0.1.4.8
  3. Reverse the order of the digits. Example: 8.4.1.0.6.4.9.7.0.2.4.4
  4. Append the string ".e164.arpa" to the end. Example: 8.4.1.0.6.4.9.7.0.2.4.4.e164.arpa
- Lookup a NAPTR record for the resulting DNS name.

DDDS

- The Dynamic Delegation Discovery System (DDDS) defined in RFC 3403 is used to implement lazy binding of strings to data, in order to support dynamically configured delegation systems.
- The DDDS functions by mapping some unique string to data stored within a DDDS Database by iteratively applying string transformation rules until a terminal condition is reached.
- The core of DDDS are NAPTR records.

References

H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson.

RTP: A Transport Protocol for Real-Time Applications.
RFC 3550, Columbia University, Packet Design, Blue Coat Systems Inc., Packet Design, July 2003.

A. Clark T. Friedman, R. Caceres.

RTP Control Protocol Extended Reports (RTCP XR).
RFC 3611, Paris 6, IBM Research, A. Clark, November 2003.

M. Handley, V. Jacobson, and C. Perkins.

SDP: Session Description Protocol.
RFC 4566, UCL, Packet Design, University of Glasgow, July 2006.

J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler.

SIP: Session Initiation Protocol.
RFC 3261, dynamicsoft, Columbia University, Ericsson, WorldCom, Neustar, ICIR, ATT, June 2002.

J. Rosenberg and H. Schulzrinne.

Session Initiation Protocol (SIP): Locating SIP Servers.
RFC 3263, dynamicsoft, Columbia University, June 2002.

H. Schulzrinne and J. Rosenberg.

The Session Initiation Protocol: Internet-Centric Signaling.
IEEE Communications Magazine, 38(10), October 2000.

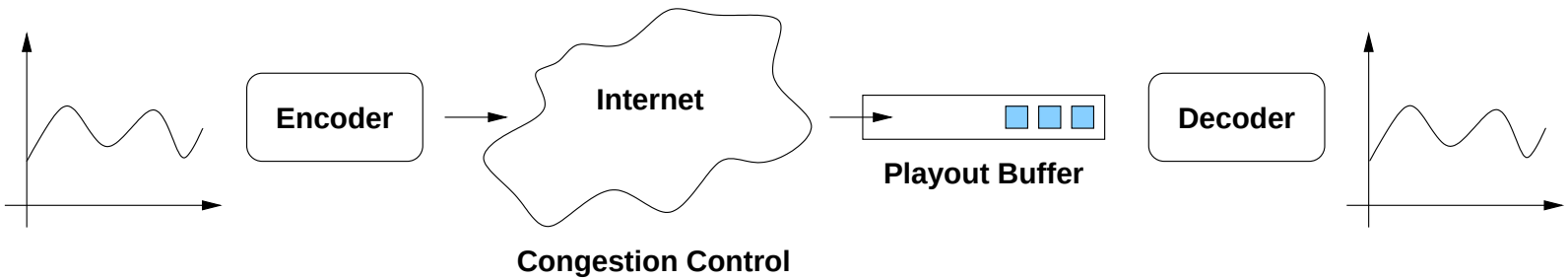
Part: Voice over IP

Background and Codecs

Voice over IP

- Idea: Send voice over the packet switched Internet
  - Digitizing and encoding voice signals
  - Transmission over the Internet
  - Decoding and generating the analog voice signal
- RTP/RTCP (UDP/IP) can be used for the transmission
- Need a common signalling protocol (ringing the phone)
- Need an infrastructure to locate users and phones

Voice over IP



- Voice quality is impacted by
  - encoding of the digitized analog signal
  - transmission impairments (delay, jitter, loss)
- Playout buffers can mitigate some of the effects
  - Playout buffers should be adaptive
  - For bidirectional voice conversations, there is an upper limit of delay

Pulse Code Modulation (PCM)

- Voice bandwidth is 4 kHz, so sampling bandwidth has to be 8 kHz (for Nyquist).
- Represent each sample with 8 bit (having 256 possible values).
- Throughput is 8000 Hz * 8 bit = 64 kbit/s, as a typical digital phone line.
- In real applications mu-law (North America) and a-law (Europe) variants are used, which code the analog signal on a logarithmic scale using 12 or 13 bits instead of 8 bits (see Standard ITU-T G.711).

Other Codecs

- Adaptive differential PCM (ADPCM), ITU-T G.726
  - Encode the difference between the actual and the previous voice packet, requiring 32 kbps.
- LD-CELP, Standard ITU-T G.728
- CS-ACELP, Standard ITU-T G.729 and G.729a
- MP-MLQ, Standard ITU-T G.723.1, 6.3kbps, Truespeech
- ACELP, Standard ITU-T G.723.1, 5.3kbps, Truespeech
- LPC-10, able to reach 2.5 kbps
- iLBC, low bit-rate, able to deal with packet loss
- speex, free codec, 8/16/32 kHZ sampling

VoIP and PSTN

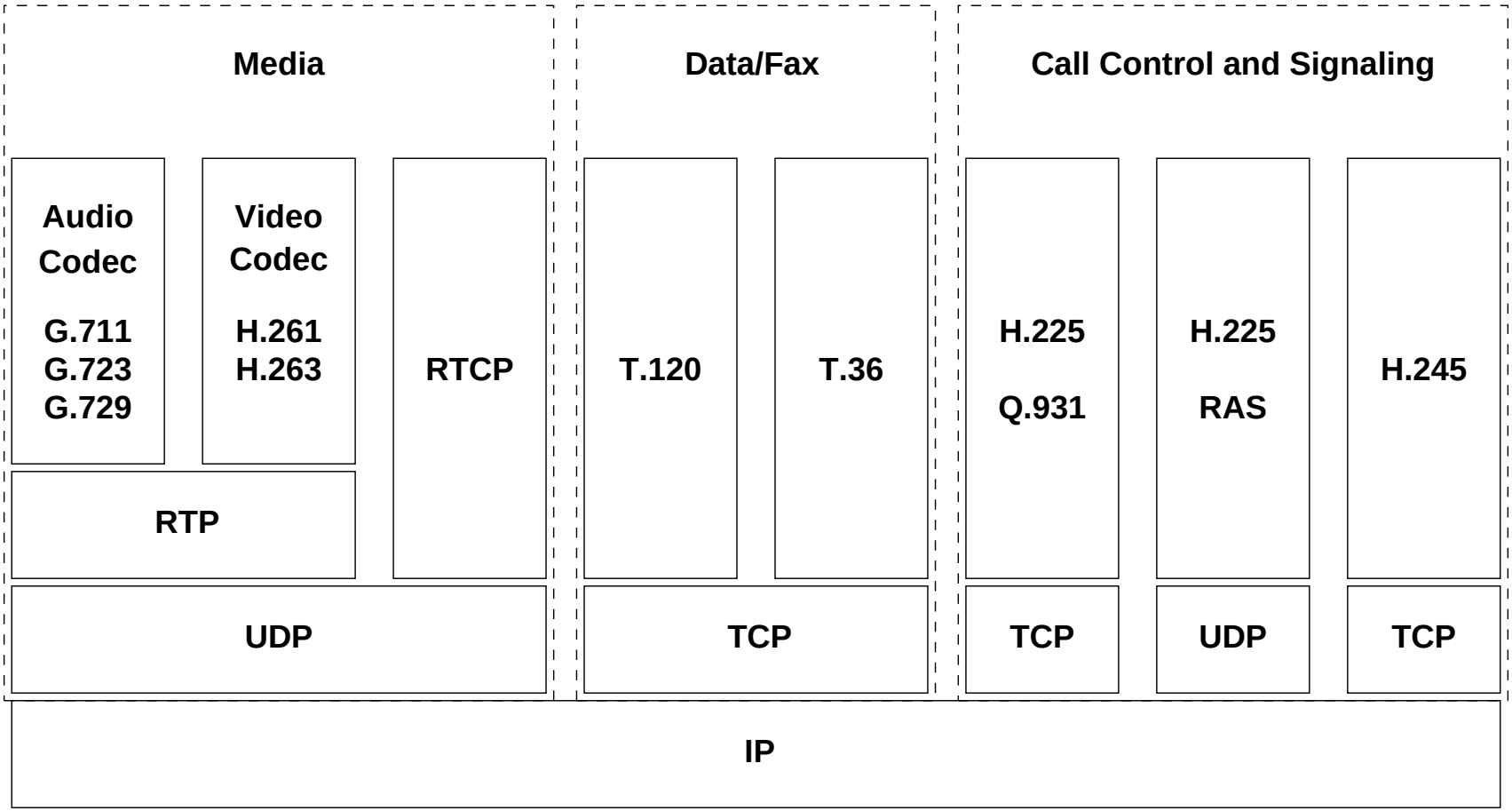
Interworking with the PSTN

- PSTN: Public Switched Telephone Network
- Users like to call from a VoIP phone a PSTN phone
- Users like to call from a PSTN phone a VoIP phone
- Users like to be reachable via the same "call number" regardless where I attach to the network
- Users like to be mobile while placing voice calls
- Two big standardization bodies involved:
  - ITU-T (International Telecommunication Union)
  - IETF (Internet Engineering Task Force)

H.323 (ITU-T)

- The H.323 standard provides a foundation for audio, video, and data communications across IP-based networks, including the Internet.
- H.323 is an umbrella recommendation setting standards for multimedia communications over packet switched networks.
- H.323 includes parts of H.225.0 - RAS, Q.931, H.245 RTP/RTCP and audio/video codecs, such as the audio codecs (G.711, G.723.1, G.728, etc.) and video codecs (H.261, H.263) that compress and decompress media streams.
- Media streams are transported on RTP/RTCP.
- The signaling is transported reliably over TCP.

H.323 Protocols



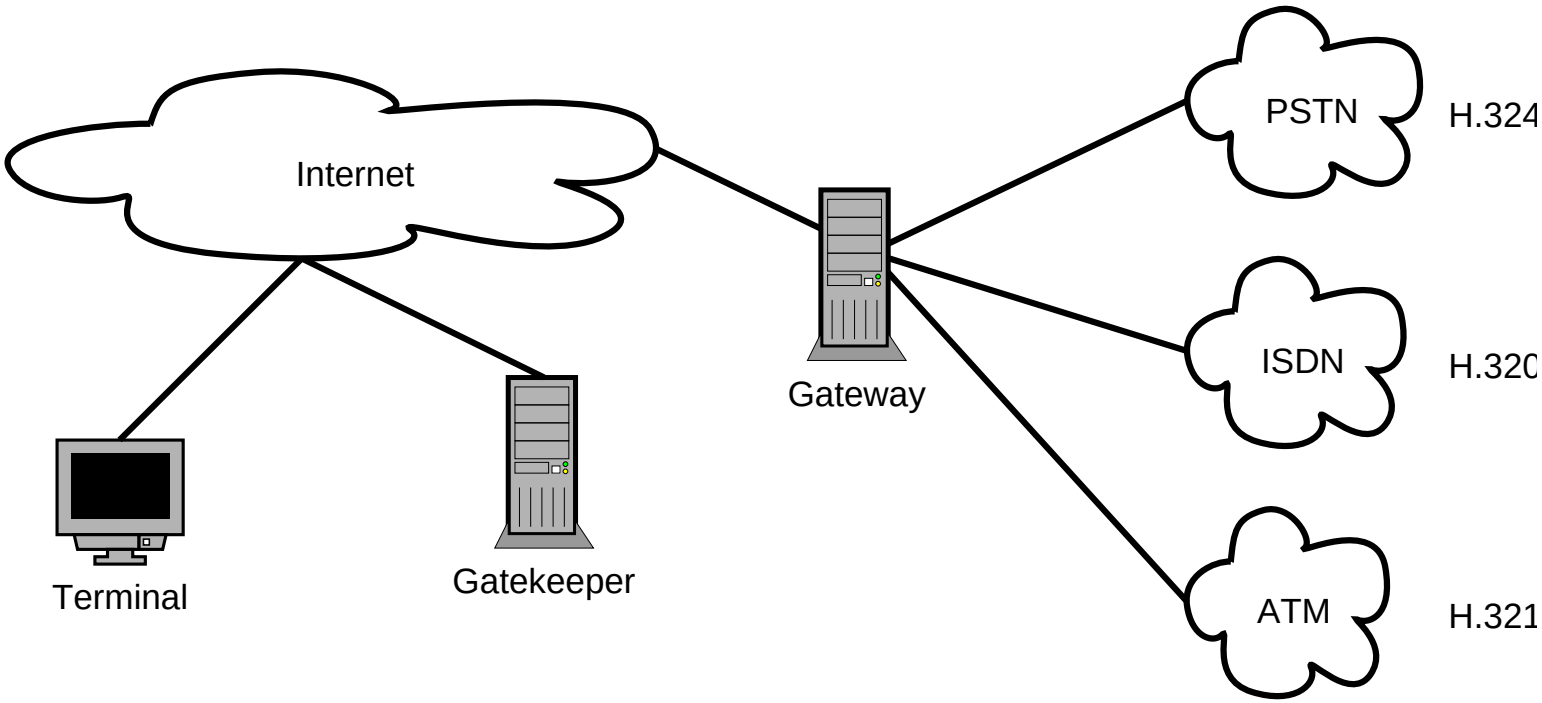
H.323 Terminology

- *Terminal*
  - End system (a phone or multimedia PC) supporting
  - H.225 call control signaling
  - H.245 control channel signaling
  - RTP/RTCP media transport
  - Audio (video) codecs
- *Gateway*
  - Interface to non-H.323 terminals
    - Translation between entities in packet switched networks and circuit switched networks
    - Tramission format and communication procedure translation

H.323 Terminology (cont.)

- *Gatekeeper*
  - Maintains information about terminals (optional).
  - Can perform address translation, admission control, bandwidth control, zone management, call control signaling, call authorization, bandwidth management, call management.
- *Multipoint Control Units*
  - Support for multi-party communication (conferences).
  - The Multipoint Controller (MC) provides control functions.
  - The Multipoint Processor (MP) receives and processes audio, video and/or data streams.

H.323 Interworking



H.323 Communication Establishment

- Communication establishment is done in five steps:
  1. Call setup
  2. Initial communication and capabilities exchange
  3. Audio/video communication establishment
  4. Call services
  5. Call termination

Why SIP and MEGACO

- H.323
  - Too heavy for devices with limited processing power
  - Does not specifically address mobility / roaming
- Session Initiation Protocol (SIP)
  - Designed for lightweight signalling
  - Addresses any media, not voice only
  - Suitable for Internet telephony
- Media Gateway Control Protocols (MGCPs)
  - Protocols for Media Gateways
  - Focuses on PSTN-PSTN via IP

Voice Quality Metrics

Mean Opinion Scores (MOS) [ITU P.800]

MOS	Quality	Impairment
1	Bad	Very annoying
2	Poor	Annoying
3	Fair	Slightly annoying
4	Good	Perceptible but not annoying
5	Excellent	Imperceptible

- Subjective tests are done by a group of testers. The MOS scores of the testers are averaged to obtain the overall MOS.
- Objective tests are done by using a model to compute MOS scores.

MOS Scores for Several Codecs

Codec	Data Rate	Mean Opinion Score (MOS)
G.711	64 kbit/s	4.1
G.729	8 kbit/s	3.92
G.723.1	6.3 kbit/s	3.9
G.729a	8 kbit/s	3.7
G.723.1	5.3 kbit/s	3.65

- Different codecs achieve different MOS values.
- Trade-off between saving bandwith and quality loss due to the codec itself.

E Model [ITU G.107]

- Idea: Calculate a factor R representing a transmission quality rating
- Definition:

$$R = R_0 - I_s - I_e - I_d + A$$

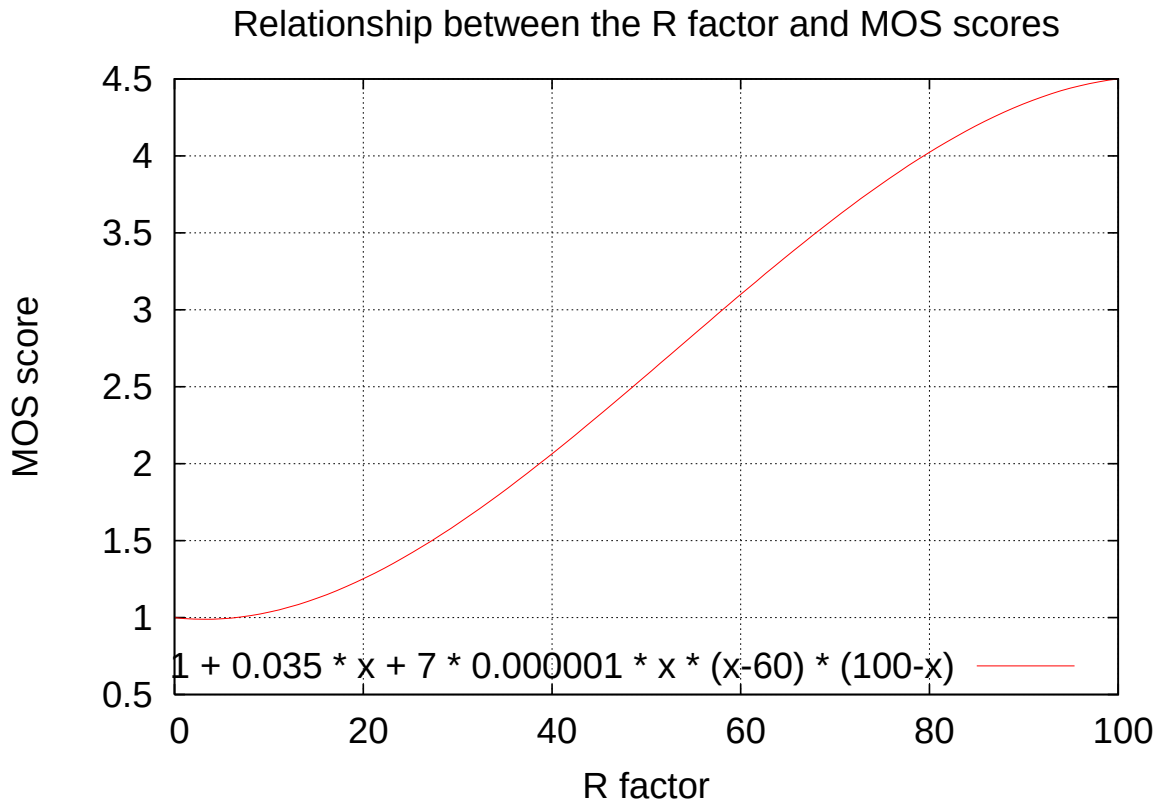
- Parameters:
  - R_0 - noise (S/N at 0 dB)
  - I_s - impairments simultaneous to voice signal
  - I_d - impairments delayed after voice signal
  - I_e - effects of special equipment (e.g., codecs)
  - A - advantage factor
- Assumption: "Psychological factors on the psychological scale are additiv"

Interpretation of the R factor

R-factor	Quality	MOS
$90 < R < 100$	Best	4.34 - 4.50
$80 < R < 90$	High	4.03 - 4.34
$70 < R < 80$	Medium	3.60 - 4.03
$60 < R < 70$	Low	3.10 - 3.60
$50 < R < 60$	Poor	2.58 - 3.10

- The R factor is in the range $[0 \ldots 100]$.
- The R factor can be used directly as a quality metric.
- A translation to MOS scores is possible as well.

R factor versus MOS



$$MOS = 1 + 0.035 \cdot R + 7 \cdot 10^{-6} \cdot R \cdot (R - 60) \cdot (100 - R)$$

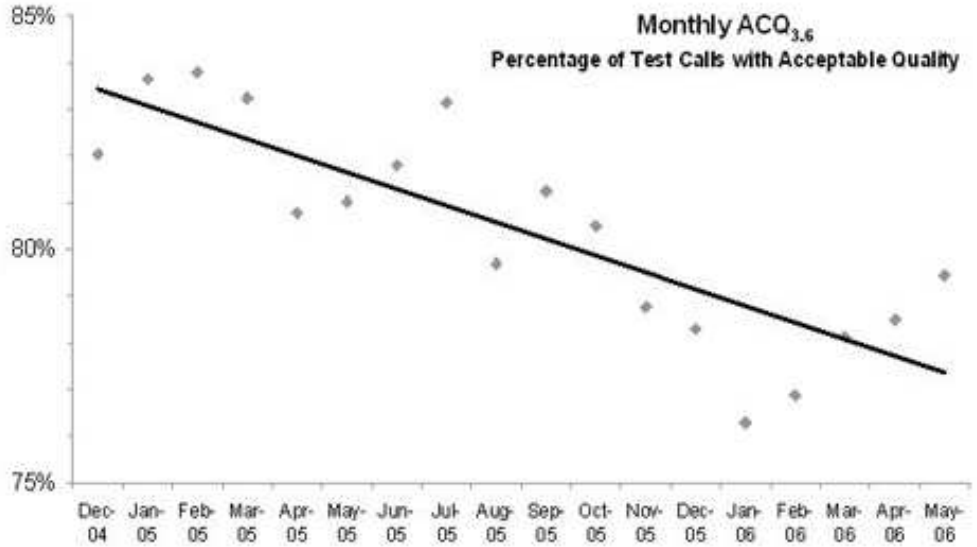
Simplified E Model

- Cole and Rosenbluth [**?**] have further simplified the model for a VoIP system:

$$R = 94.2 - I_e - I_d$$

- Parameters:
  - $I_e = \lambda_1 + \lambda_2 \cdot ln(1 + \lambda_3 e)$
  - $I_d = 0.024 \cdot d + 0.11 \cdot (d - 177.3) \cdot I(d - 177.3)$
  - e - represents the overall packet loss
  - d - represents the total end-to-end delay
  - $I(x)$ - unity step function
  - λ_i - codec parameters (reaction to loss)
- All parameters are easy to measure once the λ_i are known.
- Note that jitter does not at all impact this model.

Large Scale VoIP Measurements



- Brix Networks is running www.TestYourVoIP.com, a test site collecting (simulated) call statistics [**?**].
- The metric $ACQ_{3.6}$ is the number of calls with $MOS \geq 3.6$ recorded on the test systems.

References

ITU.

Recommendation P.800: Methods for subjective determination of transmission quality.
Recommendation ITU-T P.800, International Telecommunication Union, August 1996.

ITU.

Recommendation G.107: A computational model for use in transmission quality.
Recommendation ITU-T G.107, International Telecommunication Union, December 1998.

R. G. Cole and J. H. Rosenbluth.

Voice over IP performance monitoring.
SIGCOMM Comput. Commun. Rev., 31(2):9–24, 2001.

M. Sylor, N. Venna, and H. Ripps.

Voice Quality on the Internet in 2005 as Measured by www.TestYourVoIP.com.
In *Proc. DSOM 2006*, pages 112–123, Dublin, October 2006. Springer LNCS 3775.

Part: Internet Mobility

Mobile IP Terminology

Motivation

- With the advent of portable devices (PDAs, cell phones, music players, . . .), there is an increasing need to communicate while moving between networks.
- Mobility should ideally not impact any applications running on portable devices.
- Mobility might lead to periods of intermitted network access.
- The Internet was not designed with mobility in mind . . .

Terminology (RFC 3753)

- Fixed Node (FN):
  - A node, either a host or a router, unable to change its point of attachment to the network and its IP address without breaking open sessions.
- Mobile Node (MN):
  - An IP node capable of changing its point of attachment to the network.
  - A Mobile Node may either be a Mobile Host or a Mobile Router.

Terminology (RFC 3753)

- Mobile Host (MH):
  - A Mobile Node that is an end host and not a router.
  - A Mobile Host is capable of sending and receiving packets, that is, being a source or destination of traffic, but not a forwarder of it.
- Mobile Router (MR):
  - A router capable of changing its point of attachment to the network, moving from one link to another link.
  - The MR is capable of forwarding packets between two or more interfaces, and possibly running a dynamic routing protocol modifying the state by which it does packet forwarding.

Terminology (RFC 3753)

- Mobile Network (MN):
  - An entire network, moving as a unit, which dynamically changes its point of attachment to the Internet and thus its reachability in the topology.
  - The mobile network is composed of one or more IP-subnets and is connected to the global Internet via one or more Mobile Routers (MR).
  - The internal configuration of the mobile network is assumed to be relatively stable with respect to the MR.

Handover Terminology (RFC 3753)

- Roaming
  - An operator-based term involving formal agreements between operators that allows a mobile to get connectivity from a foreign network.
- Handover
  - The process by which an active MN changes its point of attachment to the network, or when such a change is attempted.
  - The access network may provide features to minimize the interruption to sessions in progress.
- Seamless Handover
  - A handover in which there is no change in service capability, security, or quality.

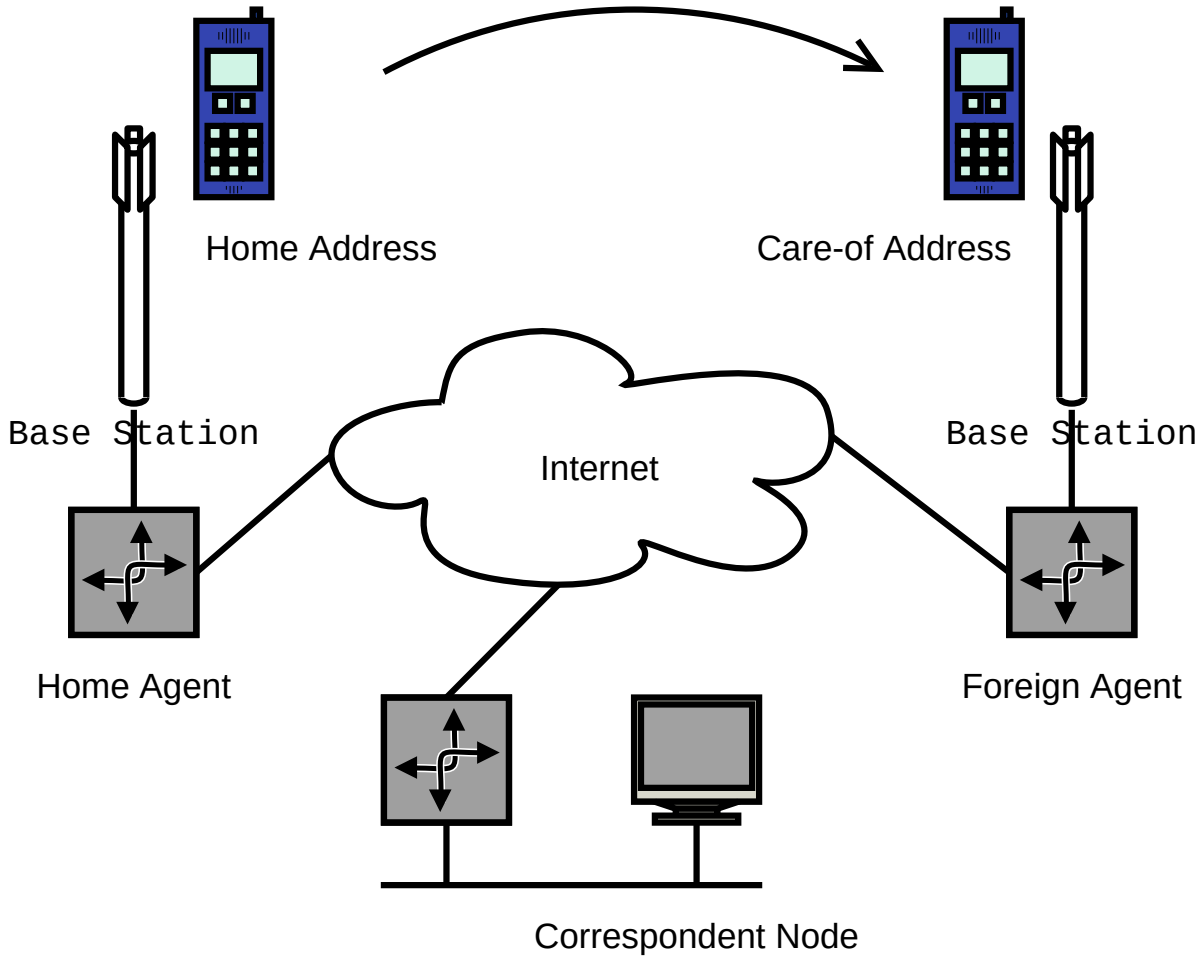
Mobility and Layering

- Link layer
  - xxx
- Network layer
  - xxx
- Transport layer
  - xxx
- Session layer
  - xxx

Mobile IP (MIP) Requirements

- Transparency
  - Mobile nodes keep their IP address
  - Transport and application protocols do not change
- Compatibility
  - Support for the existing layer two protocols
  - No changes needed on existing deployed nodes
- Security
  - Security should not be sacrificed
  - Privacy must be maintained where necessary
- Efficiency and Scalability
  - Minimize overhead for mobility support
  - Must scale to a huge number of mobile nodes (mobile phones)

Mobile IP Scenario



Mobile IPv4 (MIPv4)

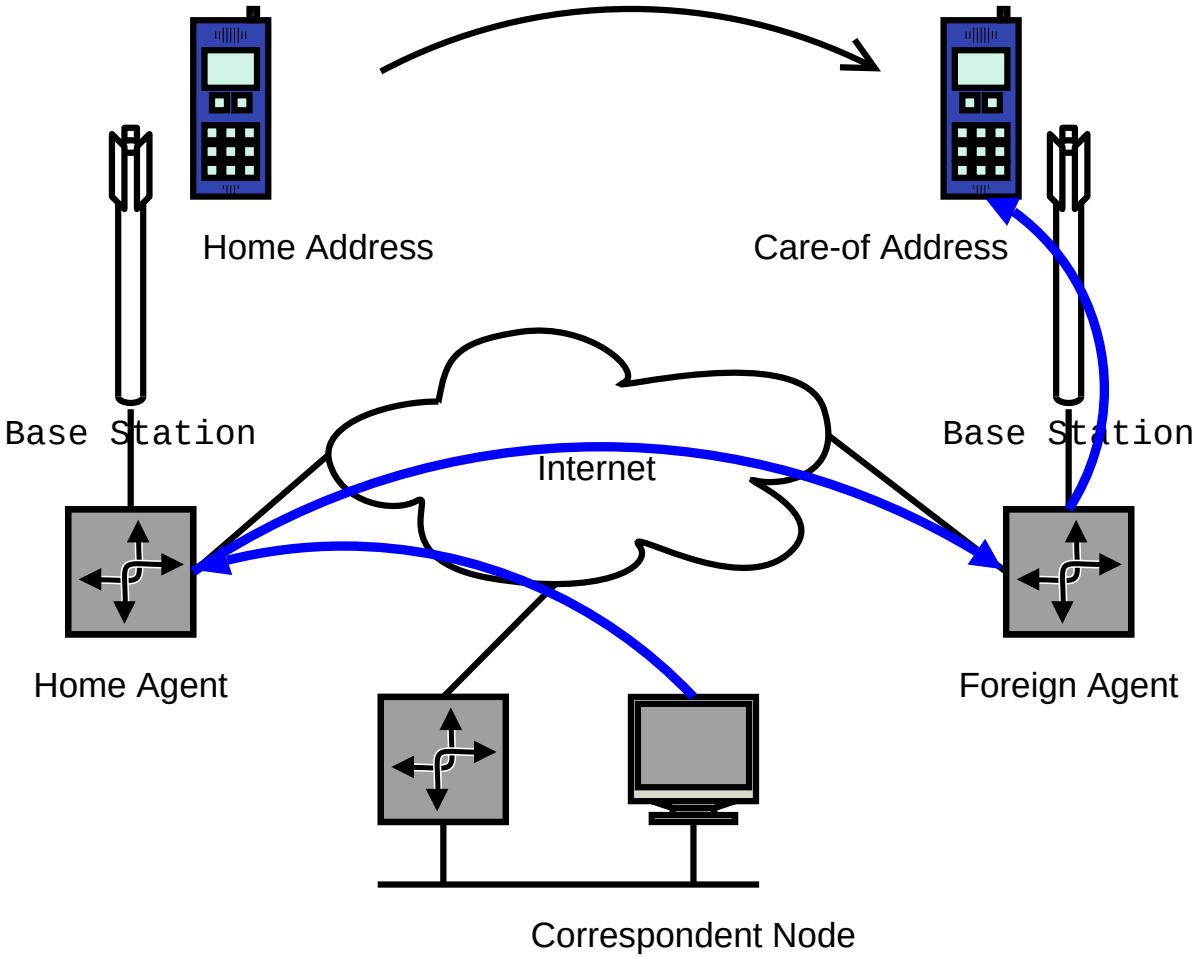
Mobile IPv4 Terminology

- Home Agent
  - A router on a mobile node's home network which tunnels datagrams for delivery to the mobile node when it is away from home, and maintains current location information for the mobile node.
- Foreign Agent
  - A router on a mobile node's visited network which provides routing services to the mobile node while registered.
  - The foreign agent detunnels and delivers datagrams to the mobile node that were tunneled by the mobile node's home agent.
  - For datagrams sent by a mobile node, the foreign agent may serve as a default router for registered mobile nodes.

Mobile IPv4 Terminology

- Home Address
  - An IP address that is assigned for an extended period of time to a mobile node. It remains unchanged regardless of where the node is attached to the Internet
- Care-of Address
  - The termination point of a tunnel toward a mobile node, for datagrams forwarded to the mobile node while it is away from home.
- Correspondent Node
  - A peer with which a mobile node is communicating. A correspondent node may be either mobile or stationary.

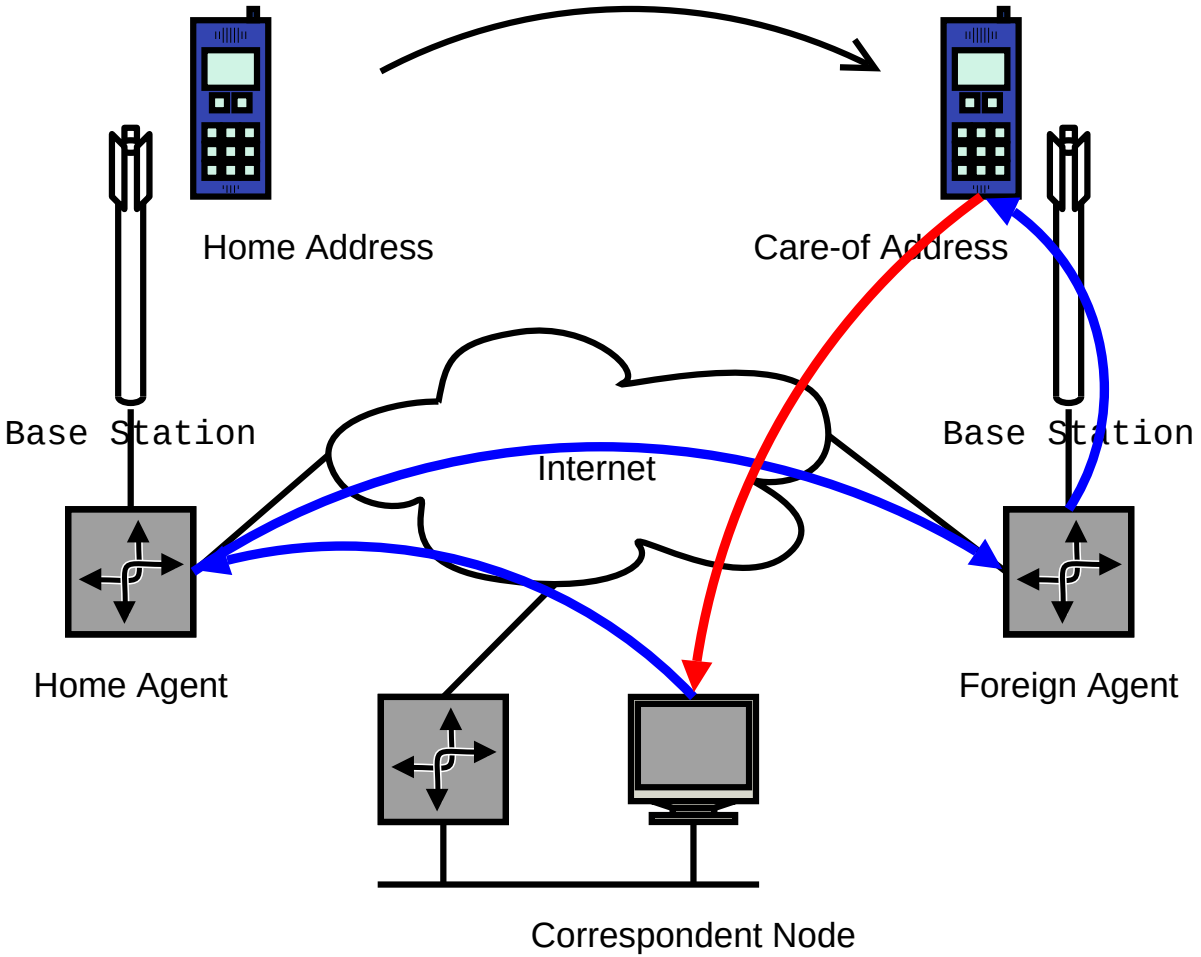
MIPv4 (RFC 3344)



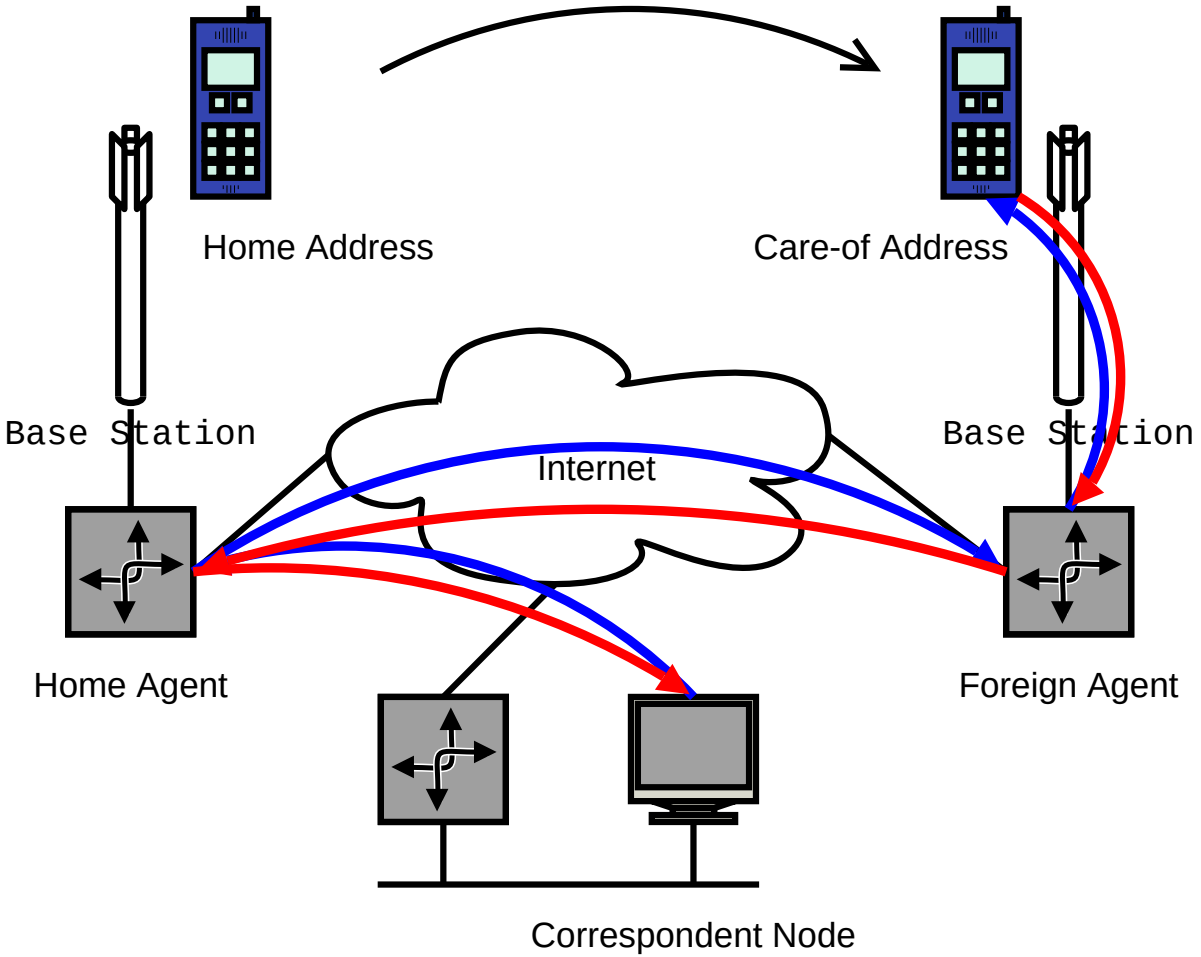
MIPv4 (RFC 3344)

- Correspondent node $\rightarrow$ mobile node:
  - Correspondent node sends IP packets to the home address of the mobile node.
  - Home agent intercepts packets and tunnels them to the current care-of address via the foreign agent.
  - Foreign agent forwards packet to the mobile node.
- Mobile node $\rightarrow$ correspondent node:
  - Direct forwarding (RFC 3344)
  - Reverse tunneling (RFC 3024)

MIPv4 (RFC 3344)



MIPv4 (RFC 3344, RFC 3024)



Direct Forwarding vs. Reverse Tunneling

- Problems with direct forwarding:
  - Asymmetric routing may cause routing problems
  - Firewall problems (topological incorrect source addresses)
  - Potential problems with multicast support
  - Loss of transparency
- Problems with reverse tunneling:
  - Double triangular routing costs
  - Reverse tunnel may be misused to bypass firewalls

Other MIPv4 Issues

- Congestion Control
  - Transport protocols may get confused if the properties of the communication path change (PMTU, congestion state, ...).
- Convergence
  - Convergence time may be significant.
  - Link layer notifications may be used to trigger the handover early.
  - It might be necessary to forward packets via multiple paths during handover.
- Security and Privacy
  - Mobile IP should not introduce any new security and privacy issues (but it does in reality).

MIPv4 Agent Discovery

- Mobile hosts discover agents (home or foreign) by listening to agent advertisements which are multicasted to the "all systems on this link" multicast address (224.0.0.1).
- The Agent Advertisement is send together with a router advertisement and contains parameters (encapsulation mechanisms) and care-of addresses.
- A mobile host can ask for agent advertisements by sending a router solicitation message.

MIPv4 Registration

- Registration messages are exchanged between a mobile node, (optionally) a foreign agent, and the home agent.
- Registration creates or modifies a *mobility binding* at the home agent, associating the mobile node's home address with its care-of address for the specified lifetime.
- Registration messages must be authenticated using keyed message digests and nonces or timestamps for replay protection.
- The home agent uses proxy ARP and gratuitous ARP while the mobile host is registered on a foreign network to intercept and redirect traffic.

Mobile IPv4 (MIPv6)

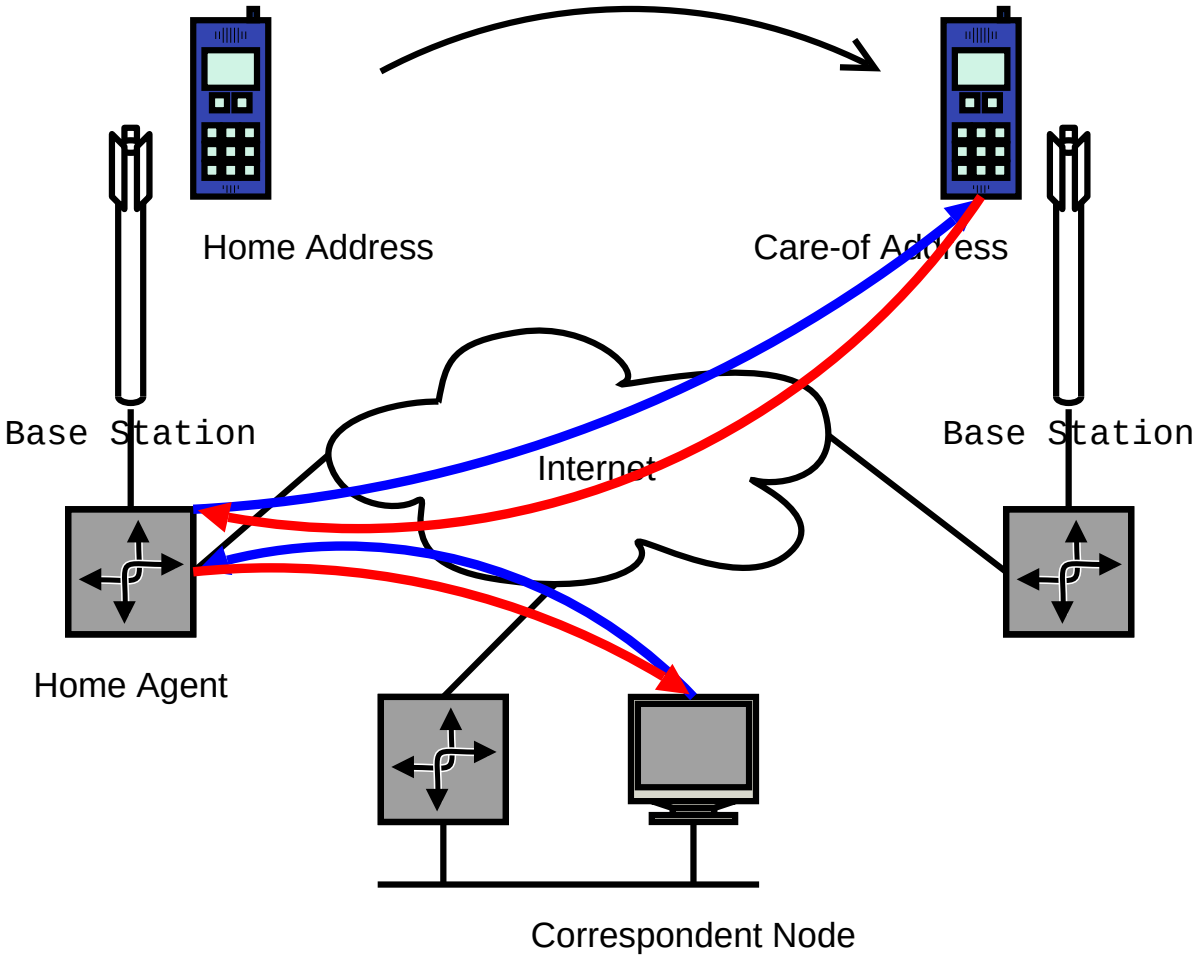
MIPv6 vs. MIPv4 (RFC 3775)

- There is no need to deploy foreign agents in MIPv6.
- Support for route optimization is a fundamental part of MIPv6.
- Mobile IPv6 route optimization can operate securely even without pre-arranged security associations.
- Most packets sent to a mobile node while away from home in Mobile IPv6 are sent using an IPv6 routing header rather than IP encapsulation, reducing the amount of resulting overhead compared to Mobile IPv4.
- Mobile IPv6 is decoupled from any particular link layer, as it uses IPv6 Neighbor Discovery.
- ...

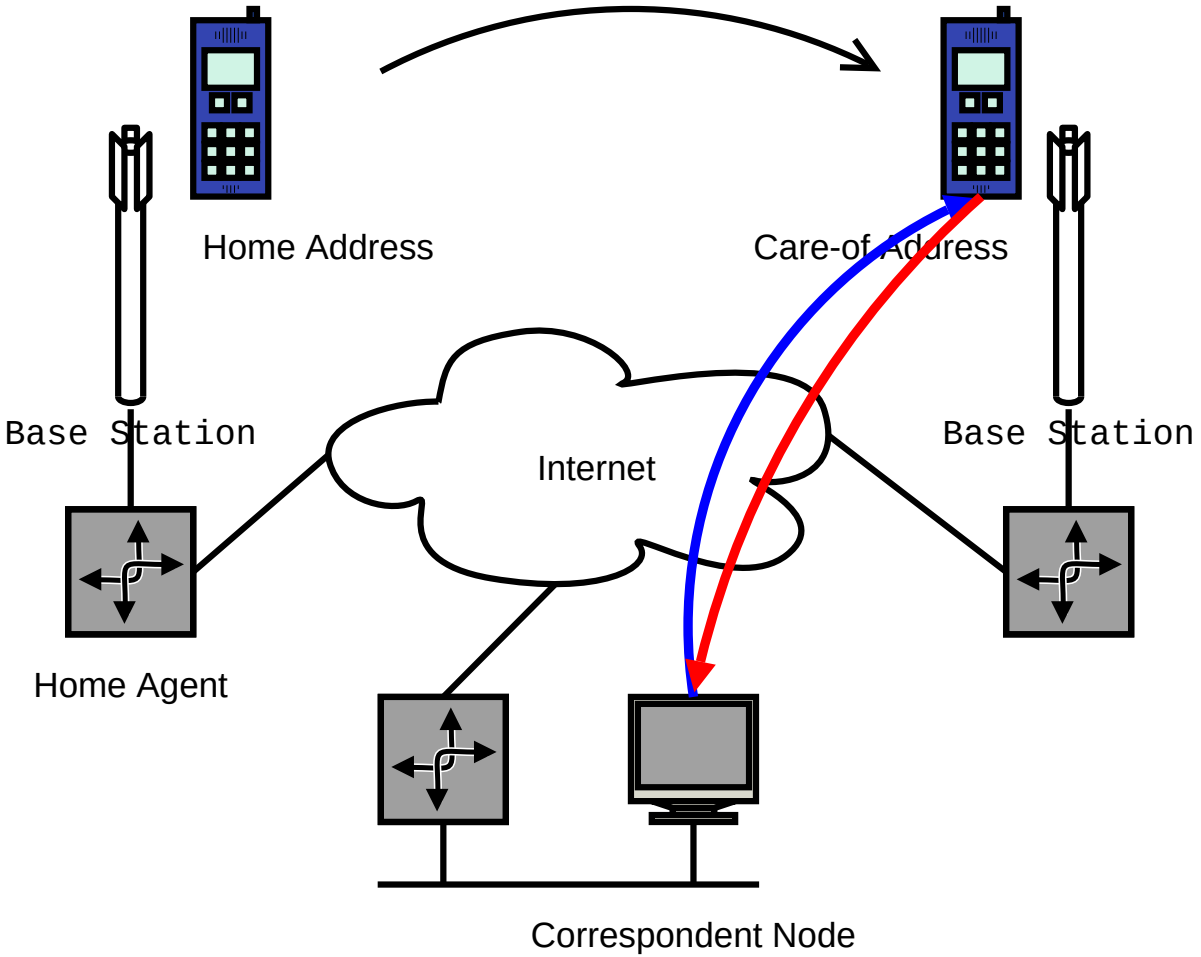
MIPv6 Principles (RFC 3775)

- Mobile nodes acquire care-of addresses via auto-configuration. (Mobile nodes may have multiple care-of addresses.)
- Mobile nodes register their primary care-of address with a router on their home links (binding).
- Mobile nodes can register with correspondent nodes to establish a direct binding.
- Mobile nodes can dynamically discover their home agent, even when the mobile node is away from home.

MIPv6 Tunnel Mode (RFC 3775)



MIPv6 Route Optimization (RFC 3775)



MIPv6 Principles

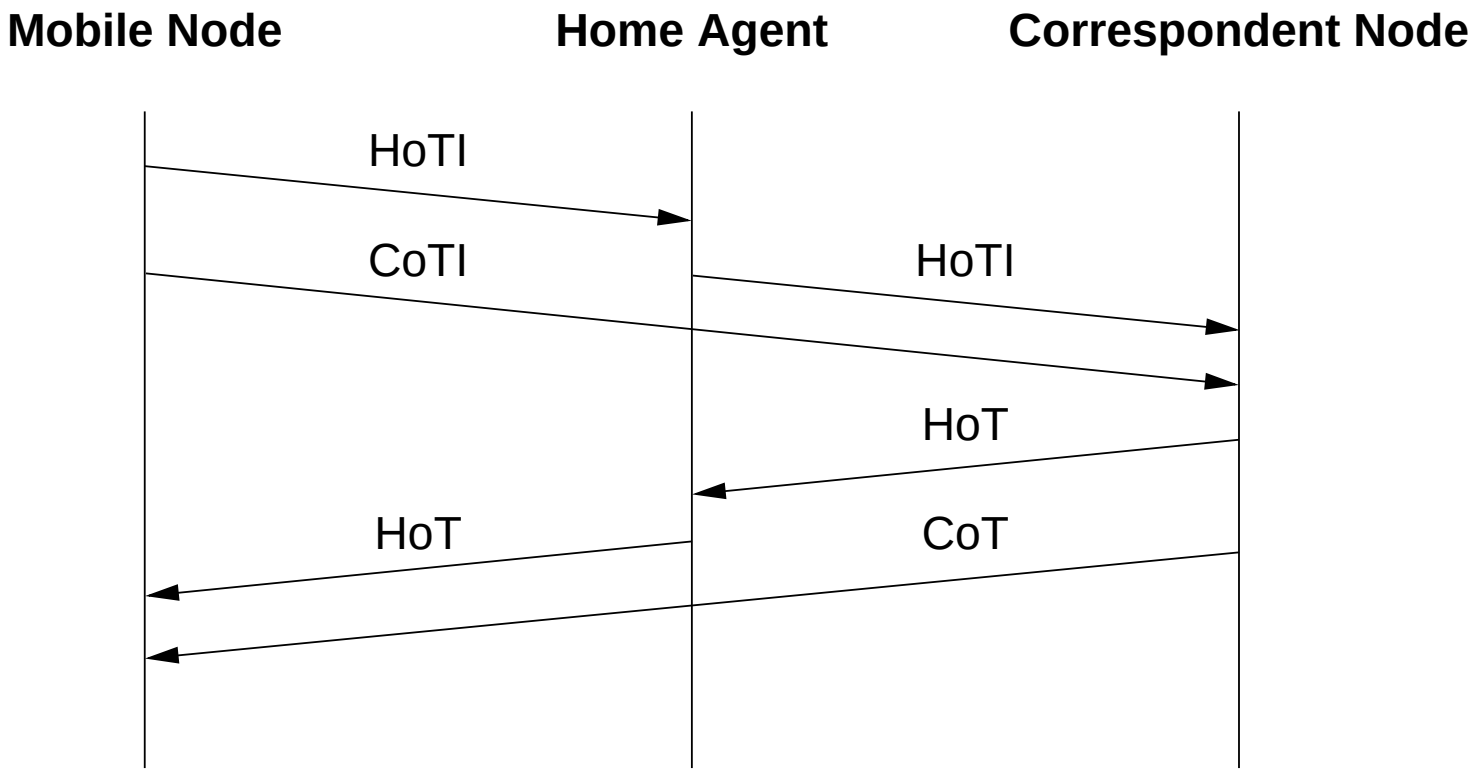
- Bidirectional tunneling mode:
  - The home agent intercepts traffic addressed to the mobile node by neighbor discovery intercept and tunnels the traffic to the mobile node's primary care-of address using IPv6 encapsulation.
- Route optimization mode:
  - After registration at the correspondent node, the correspondent node sends traffic directly to the mobile node using a new type of IPv6 routing header.
  - The mobile node directly sends packets to the correspondent node, storing its home address in a new home address destination option.

MIPv6 Security

- Binding updates between the mobile node and the home agent are protected by IPsec, which requires pre-installed security associations.
- Binding updates between the mobile node and the correspondent nodes use a procedure called "return routability procedure", which does not require an authentication infrastructure.
- The "return routability procedure" enables the correspondent node to obtain some *reasonable assurance* that the mobile node is in fact addressable at its claimed care-of address as well as at its home address.

$\Rightarrow$ Does not protect against attackers who are on the path between the home network and the correspondent node.

Return Routability Procedure

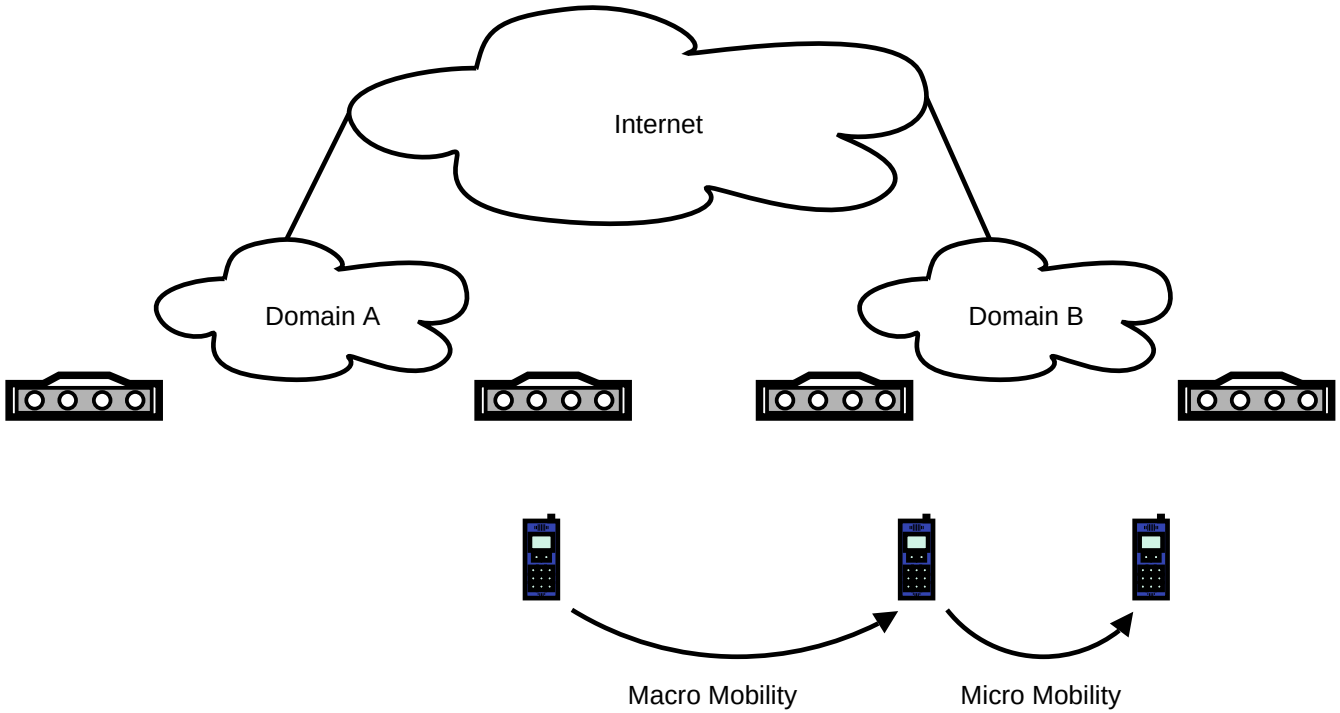


- Home Test Init (HoTI), Home Test (HoT)
- Care-of Test Init (CoTI), Care-of Test (CoT)

Return Routability Procedure

- Testing whether packets addressed to the two claimed addresses are routed to the mobile node.
- A mobile node can pass the test only if it is able to supply proof that it received certain data (the "keygen tokens") which the correspondent node sends to those addresses.
- These data are combined by the mobile node into a binding management key, denoted K_{bm}.
- $K_{bm} = SHA1(home\_keygen\_token | care\_of\_keygen\_token)$
- The binding management key K_{bm} is checked by the correspondent node during the binding update.
- Nonces are used to guarantee the freshness of the messages.

Micro Mobility vs. Macro Mobility



- Mobile IP works reasonably well for macro mobility
- Micro mobility requires additional work for seamless handover

Fast Handover for MIPv6 (RFC 4260)

- Goals:
  - Reduce the time needed to restore IP connectivity.
  - Support real-time services in a mobile network.
- Mechanisms:
  - Leverage information from the link-layer to predict and rapidly respond to handover events.
  - Tunnel data between the old and new access routers.
  - Provide IP connectivity in advance of actual mobile IP registration.
- Generic fast handover extension plus link-layer specific adaptations.

802.11 Handover Procedure

0. A station (S) realizes that a handoff is necessary due to degrading radio transmission environment for the current access point (AP).
1. S performs a scan to see what APs are available. The result of the scan is a list of APs together with physical layer information, such as signal strength.
2. S chooses one of the APs and performs a join to synchronize its physical and MAC layer timing parameters with the selected AP.
3. S requests authentication with the new AP. For an "Open System", such authentication is a single round-trip message exchange with null authentication.

802.11 Handover Procedure (cont.)

4. S requests association or re-association with the new AP. A re-association request contains the MAC-layer address of the old AP, while a plain association request does not.
5. If operating in accordance with 802.11i, S and AP would execute 802.1X EAP-on-LAN procedures to authenticate the association.
6. If operating in accordance with the IAPP, AP may contact the old AP to transfer some information about the session and clean up the state at the old AP.
7. The new AP sends a Layer 2 Update frame on the local LAN segment to update the learning tables of any connected bridges.

802.11 Implementation Issues

- Some NICs scan the network periodically in the background while others do it only when needed.
- Scanning may take several hundred milliseconds to complete.
- Some implementations do the first steps in firmware, making it impossible for the host to get involved.
- Some implementations decide in firmware which AP is being selected, leaving the host without control over this decision.
- The coverage area of an AP is called as its Basic Service Set (BSS). Several APs with a common ESSID can form an Extended Service Set (ESS). Handover between ESSs may require a fast MIP handover.

Fast Handover for MIPv6 and 802.11

a. The MN sends a Router Solicitation for Proxy (RtSolPr) to find out about neighboring Access Routers (ARs).
b. The MN receives a Proxy Router Advertisement (PrRtAdv) containing [AP-ID, AR-Info] tuples.
c. The MN sends a Fast Binding Update (FBU) to the Previous Access Router (PAR).
d. The PAR sends a Handover Initiate (HI) message to the New Access Router (NAR).
e. The NAR sends a Handover Acknowledge (HAck) message to the PAR.
f. The PAR sends a Fast Binding Acknowledgement (FBack) message to the MS the new link.
g. The MN sends Fast Neighbor Advertisement (FNA) to the NAR after attaching to it.

FMIPv6 802.11 Scenarios

- 1abcdef234567g (predictive mode)
  - Requires that the scan and join operations (steps 1 and 2) can be performed separately and under host control.
  - The scan data must be recent once the FMIP handover procedure is triggered (otherwise it might choose the wrong AP).
- ab1234567cdefg (reactive mode)
  - Does not require host intervention of the link-layer handover.
  - Requires that the mobile node obtains the link-layer address of the NAR prior to handover.

FMIPv6 802.11 Scenarios (cont.)

- 1234567abcdefg (reactive mode)
  - Completely reactive, consists of soliciting a router advertisement after handover.
  - Does not require support from the firmware.

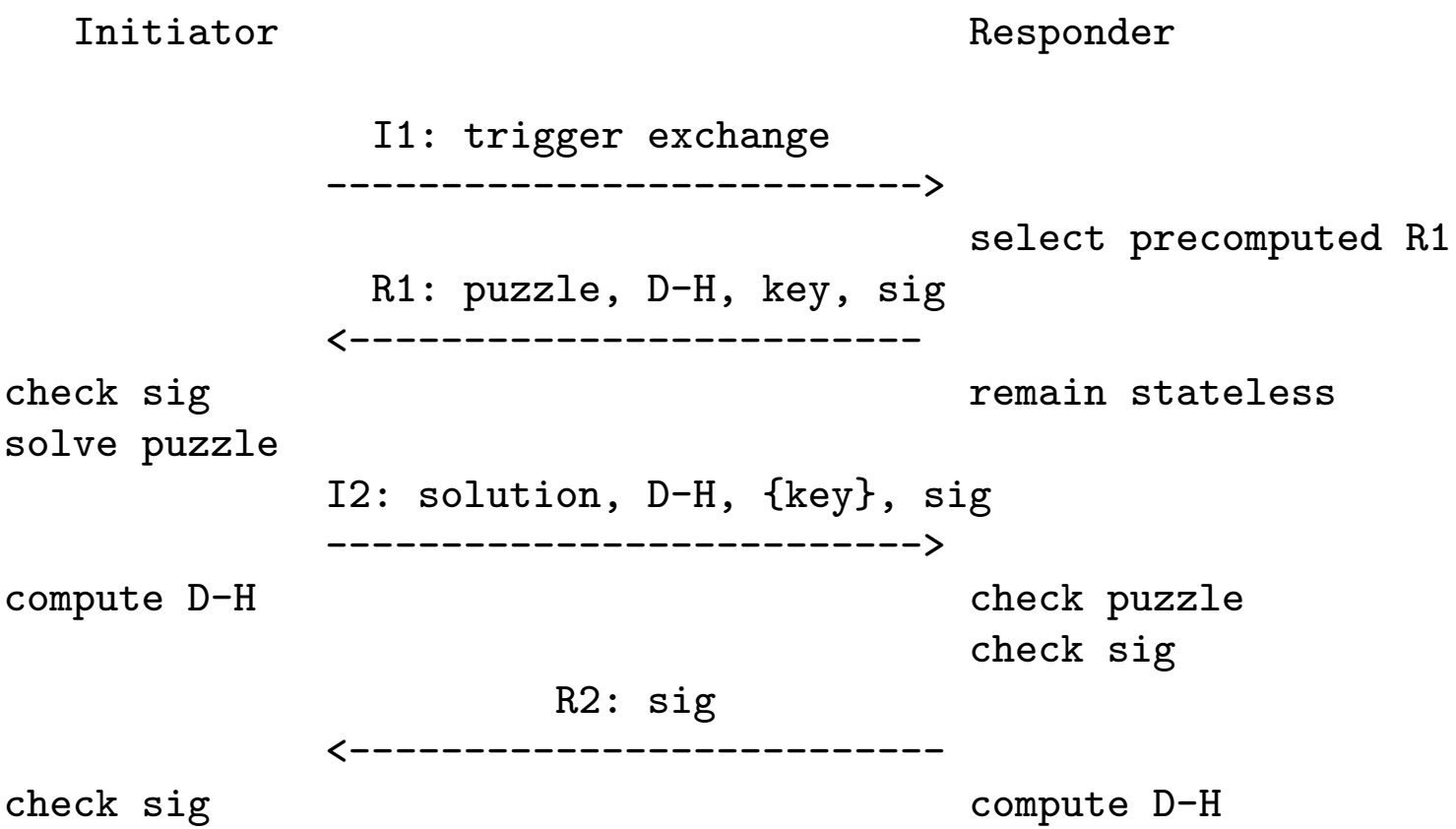
Host Identity Protocol (HIP) (RFC 4423)

Host Identity Protocol (HIP)

Basic Concepts

- Introduction of a new name space, separating the host identifier from the locator
- The locator may change while the host identifier used by transport endpoints remains stable.
- Every host has a public/private key pair; the public key is the Host Identifier (HI).
- A Host Identity Tag (HIP) is a 128-bit representation of a Host Identity consisting of an IPv6 /28 prefix followed by a cryptographic hash of the HI (100 bits).
- Host keys can be used to establish security associations for the IPsec protocol.

HIP Base Exchange (RFC 5201)



HIP Puzzle (RFC 5201)

- The Responder starts the puzzle exchange when it receives an I1.
- The Responder supplies a random number I, and requires the Initiator to find a number J.
- To select a proper J, the Initiator must create the concatenation of I, the HITs of the parties, and J, and take a hash over this concatenation using the HIP hash algorithm.
- The lowest order K bits of the result MUST be zeros. The value K sets the difficulty of the puzzle.

HIP DNS Records and Rendezvous Server

HIP DNS Records (RFC 5205)

- A DNS resource record allowing a HIP node to store its Host Identity (HI), Host Identity Tag (HIT), and the domain names of its rendezvous servers (RVSs).
- Simplifies discovery (if the DNS can be trusted).

HIP Rendezvous Server (RFC 5204)

- Mobile nodes can register their HIT $\rightarrow$ IP address mappings with a rendezvous server (RVS).
- Peers can initiate a HIP base exchange with the IP address of the RVS, which will relay this initial communication such that the base exchange may successfully complete.

References I

J. Manner and M. Kojo.

Mobility Related Terminology.
RFC 3753, June 2004.

C. Perkins.

IP Mobility Support for IPv4.
RFC 3344, Nokia Research Center, August 2002.

G. Montenegro.

Reverse Tunneling for Mobile IP, revised.
RFC 3024, Sun Microsystems, Inc., January 2001.

D. Johnson, C. Perkins, and J. Arkko.

Mobility Support in IPv6.
RFC 3775, Rice University, Nokia Research Center, Ericsson, June 2004.

P. McCann.

Mobile IPv6 Fast Handovers for 802.11 Networks.
RFC 4260, Lucent Technologies, November 2005.

W. M. Eddy.

At what layer does mobility belong?
IEEE Communications Magazine, 42(10):155–159, October 2004.

T. Koponen, P. Eronen, and Mikko Saärelä.

Resilient connections for SSH and TLS.
In *Proc. of USENIX Annual Technical Conference 2006*, Boston, May 2006.

References II

P. Nikander, A. Gurtov, and T. R. Henderson.

Host Identity Protocol (HIP): Connectivity, Mobility, Multi-Homing, Security, and Privacy over IPv4 and IPv6.

IEEE Communications Surveys and Tutorials, 12(2):186–204, 2010.

Part: Asynchronous Transfer Mode (ATM)

Packet vs. Circuit Switching, Virtual Circuits

Packet Switching

Characteristics

- Source splits messages or a data stream into (variable length) packets
- Every packet is self-contained and includes the destination address
- Every packet travels independently to the destination
- Destination recreates the messages or data stream
- Routers need to maintain information about how to forward packets to arbitrary destinations

Circuit Switching

Characteristics

- Source first establishes a connection (circuit) to the destination
- Each switch on the path might reserve some resources
- Source sends data without addressing information since the switches know the path
- Finally, the connection (circuit) is torn down
- Switches need to maintain information about all circuits running through them

Traditional Circuit Switching

Circuit of Dedicated Wires

The circuit is built out of dedicated wires, like the traditional phone network.

+ Every wire has predictable performance (bandwidth, delay, ...)
+ Simple switch design: no need for complex destination address lookups
- Costly and inefficient if there is no traffic: no statistical multiplexing
- Efforts involved in establishing circuits and tearing them down

Virtual Circuits

Virtualization of Wires

- Each wire carries many "virtual" circuits
- A virtual circuit identifier is used to distinguish virtual circuits running over a physical wire
- Data carries the virtual circuit identifier in a small header
- Switching based on the virtual circuit identifier
- Statistical multiplexing enables more efficient use of wires
- A path is determined when a virtual circuit is established
- Can support a wide range of quality of service

Packet Switching vs. Virtual Circuits

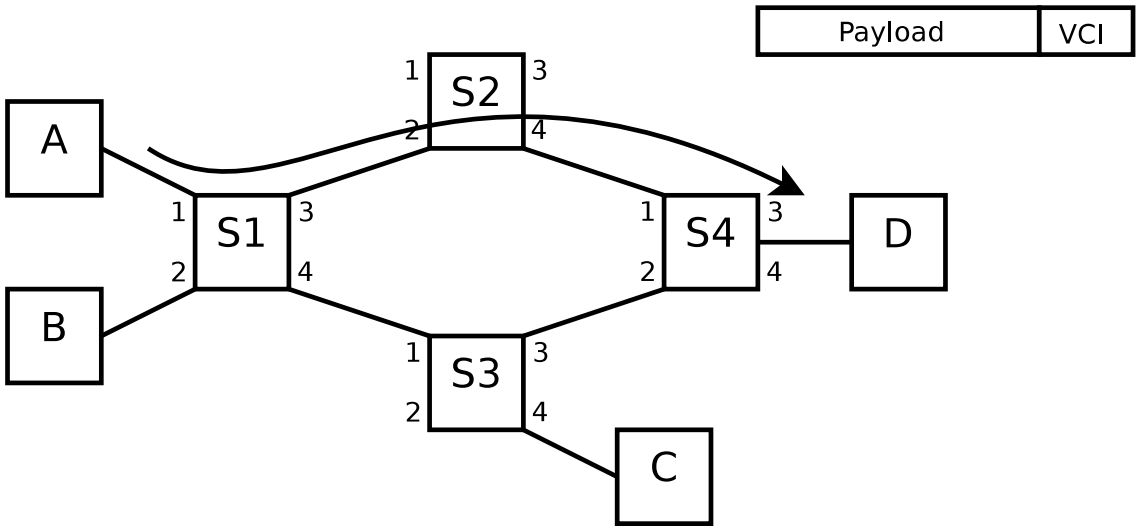
Similarities

- Store and forward communication based on an address
- Data is send in "units" with a small header
- Multiplexing without reservation: statistical multiplexing
- Multiplexing with reservation: some "flows" are guaranteed to get a certain number of "slots"

Differences

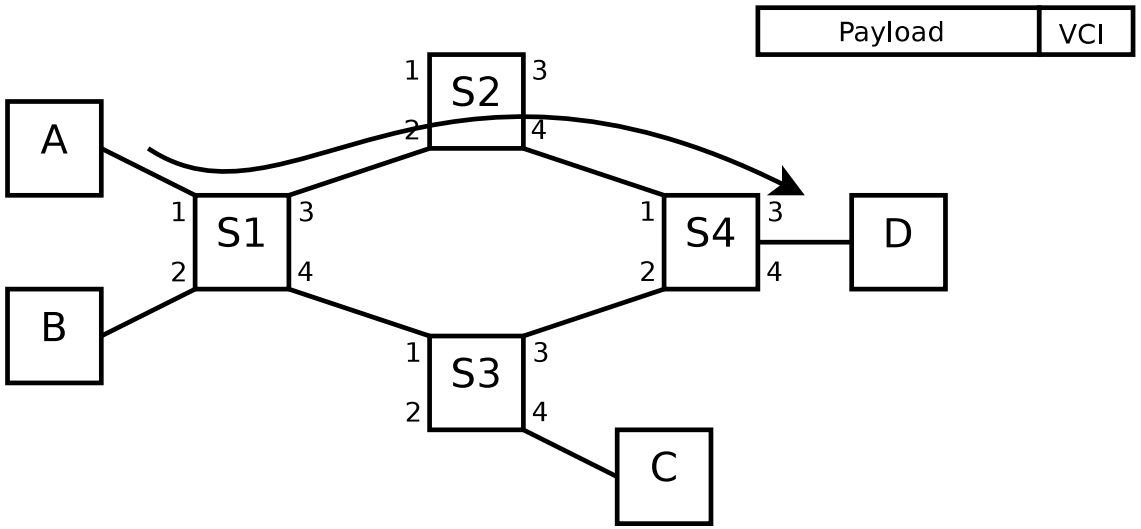
- Switches know the connections and can easily implement quality of service features
- Fast forwarding based on simple virtual circuit identifiers
- Packet switching does not require to setup a connection
- Routers are stateless: easier to recover from failures

Virtual Circuit Identifier



- How to allocate virtual circuit identifier (VCIs)?
- Globally unique? Per-link unique? . . . ?

VCI Swapping



Switch	In Port	In VCI	Out Port	Out VCI
S1	1	5	3	8
S2	2	8	4	7
S4	1	7	3	6

Signalling Protocol

Requirements

- Must find a path in a given network topology
- Must allocate per-link unused VCI
- May allocate resources for QoS guarantees
- May exercise admission control
- May provide support for traffic engineering

Realizations

- Dedicated signalling protocols (ATM)
- Piggybacked on routing protocols (MPLS)

Virtual Circuits in Practice

ATM

- Kitchen sink
- Support for voice and data communication
- Intended as IP replacement ...
- Widely deployed in Telco networks (e.g., DSL)

MPLS

- Good ideas stolen from ATM
- Integrates well with IP (designed by "IP heads")
- Widely deployed in backbone networks
- Supports traffic engineering, virtual private networks, ...

Asynchronous Transfer Mode

Asynchronous Transfer Mode (ATM)

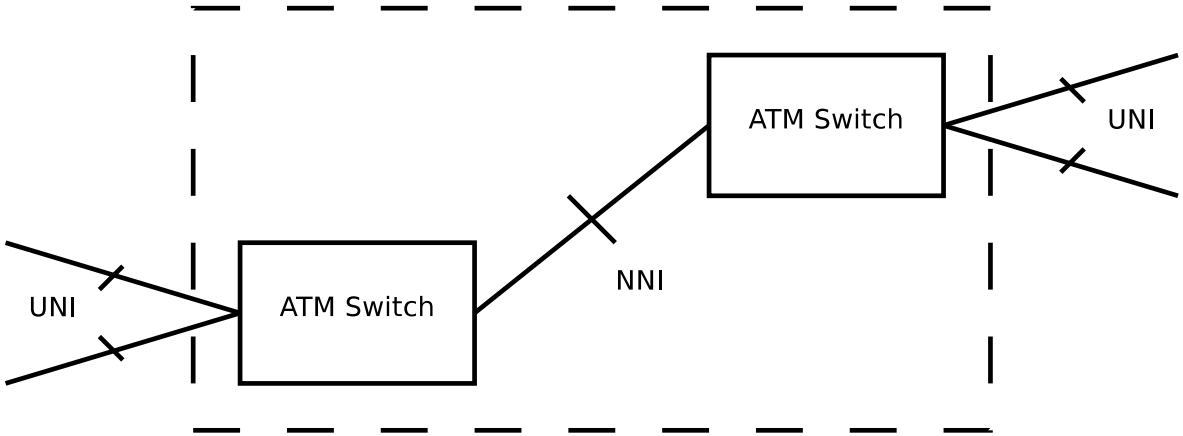
Characteristics

- Designed to carry voice and data traffic
- Data carried in cells (48 byte payload, 5 byte header)
- Pre-configured Permanent Virtual Circuits (PVCs)
- Dynamically created Switched Virtual Circuits (SVCs)

ATM service qualities

- Constant Bit Rate (CBR) - maximum peak cell rate
- Variable Bit Rate (VBR) - token bucket controlled
- Available Bit Rate (ABR) - minimum guaranteed cell rate
- Unspecified Bit Rate (UBR) - left over capacity

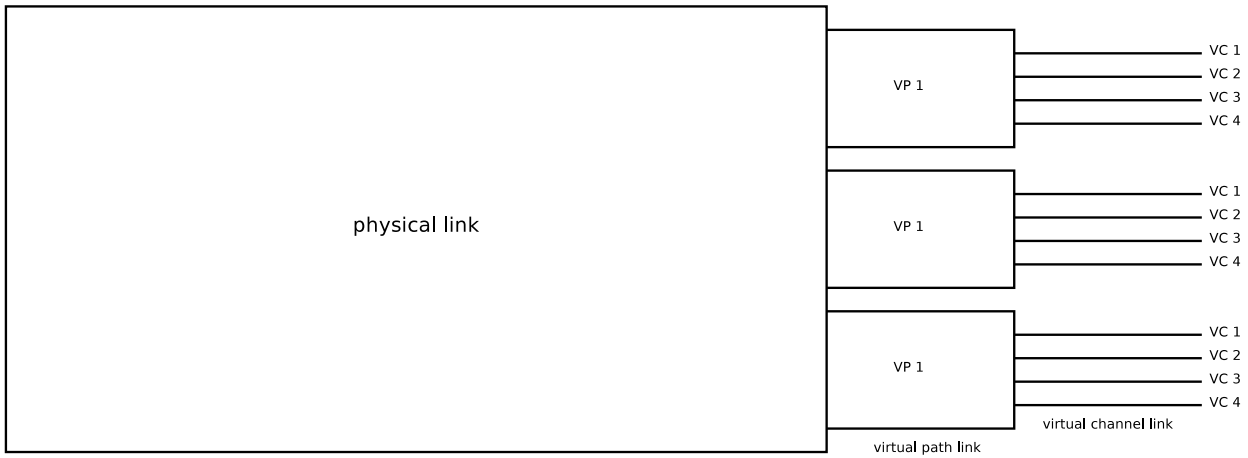
ATM and UNI vs. NNI



- End systems are connected to an ATM switch via the User Network Interface (UNI)
- ATM switches within an ATM network are connected via the Network Node Interface (NNI)

ATM Layering Model

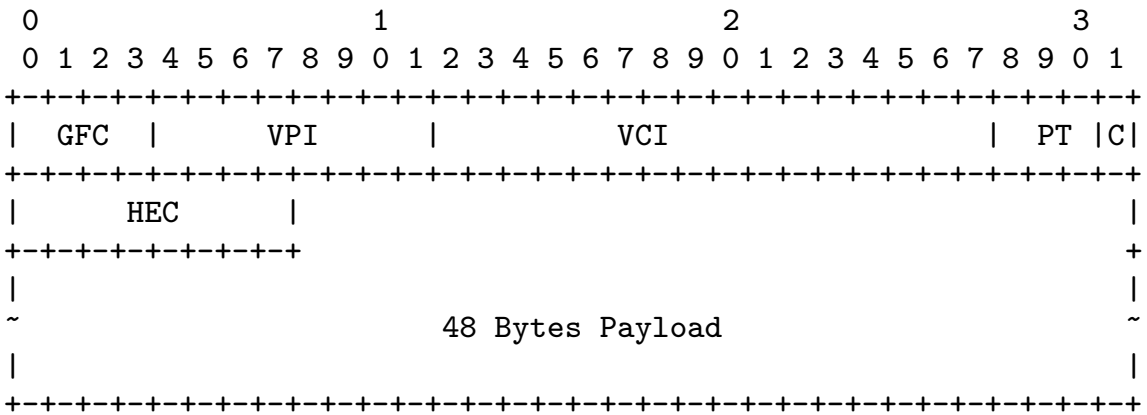
Virtual Paths and Virtual Channels



- A virtual channel connects two end systems.
- Several channels can be aggregated to virtual paths.
- A physical link carries several virtual path links and each virtual path link carries several virtual channel links.

Cells and Cell Switching

ATM Cell Format



- GFC = Generic Flow Control (or VPI in NNI)
- VPI = Virtual Path Identifier
- VCI = Virtual Channel Identifier
- PT = Payload Type
- C = Cell Loss Priority
- HEC = Header Error Control (8-bit CRC $x^8 + x^2 + x + 1$)

ATM Cell Switching

Cells Switching Properties

- Fixed-size cells simplify fast hardware implementations
- Small size reduces blocking delays
- Hierarchical virtual circuits (VPI/VCI)
- Fast table lookups through VPI/VCI switching

Why 53?

- France wanted 32 byte cells: 32 bytes = 4 ms packetization delay, France is 3 ms wide $\Rightarrow$ no echo cancellation needed
- USA wanted 64 byte cells: USA is 16 ms wide, echo cancellation needed anyways, 64 byte is more efficient
- Outcome: 48 byte payload — pain for everybody

Adaptation Layers

ATM Adaptation Layers

- AAL1: audio and video with constant bitrate
- AAL2: audio and video with variable bitrate
- AAL3: connectionless data traffic
- AAL4: data traffic
- AAL5: data traffic

AAL5 Adaptation Layer

Principle

- A data frame is split into many cells
- The last cell includes an End of Frame (EOF) flag
- A frame is reassembled when it leaves the ATM network

Example

A 1500 Ethernet frame is split into 32 ATM cells. The 32 cells introduce $32 \cdot 5 = 160$ bytes cell header overhead. Loss of a single cell causes in the worst case (when the cell carrying the EOF flag is lost) two packets to get lost.

AAL5 Adaptation Layer

Problem

- Low cell loss rate can still result in a high frame loss rate
- Retransmissions of large frames is expensive

Solution: Partial Packet Discard

- If a cell must be dropped, drop all cells that belong to the same frame as well, i.e., drop all cells up to and including the next EOF marked cell.

Solution: Early Packet Discard

- If buffers fill up, prefer to drop complete frames instead of waiting until partial frames are discarded

Practical Usage

LAN Emulation

LAN Emulation

- Emulation of IEEE 802 local area networks
- Problem: ATM has no broadcast/multicast capability
- Solution: Special server emulate broadcasts/multicasts by sending (potentially many) unicast frames
- Complex technology, error prone, expensive

IP over ATM

Option #1: Static Virtual Circuits

- Static virtual circuits are treated as links
- IP is not aware of the underlying ATM circuit

Option #2: Dynamic Virtual Circuits

- First packet in a flow dynamically creates a virtual circuit to be used by the flow
- Virtual circuit times out when it is not used
- IP flows can take advantage of ATM QoS capabilities
- Problem: First packet is slow due to ATM signalling
- Problem: Costs not justified for all IP flows

IP over ATM (cont.)

Option #3: IP Switching

- Monitor flows passing through the network
- Create specific virtual circuits for high-volume flows
- Bypass slow IP forwarding by using ATM switching
- Fall back to IP forwarding in case of an error
- Problem: How to identify "elephant" flows?

Discussion

- Complex technology, replication of functionality
- ATM switching core is simple and efficient
- ATM signalling is complicated

References

Le Boudec.

The Asynchronous Transfer Mode: A Tutorial.
Computer Networks and ISDN Systems, 24(4), May 1992.

Part: Multiprotocol Label Switching (MPLS)

Multiprotocol Label Switching

MPLS Overview

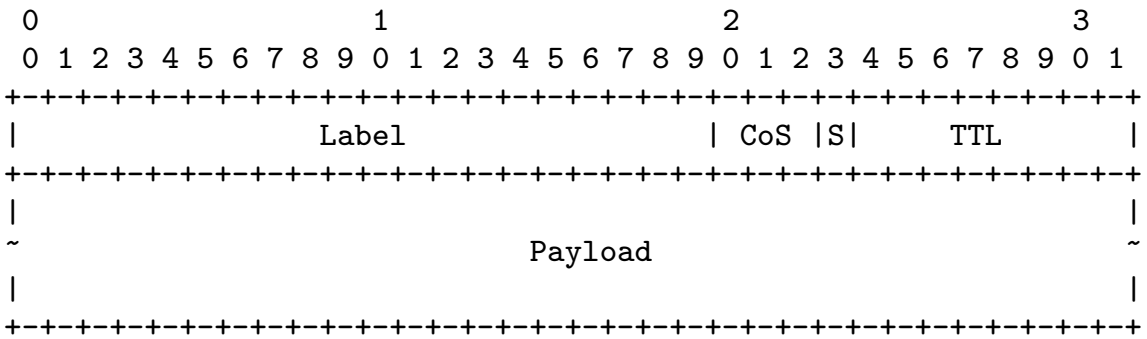
Basic Idea

- Classify IP packets with similar characteristics and which may be forwarded the same way into Forwarding Equivalence Classes (FECs)
- Attach a label to packets when they enter the network
- Forwarding is based on the label in the network core

Benefits

- Packet forwarding can be faster
- Routing can be based on ingress router and interface
- Can easily support more complex routing decision than longest prefix matches
- Can force packets to follow a pinned route (even without a routable prefix)

MPLS Label



- Label = 20-bit label
- CoS = 3-bit class of service
- S = bottom of stack flag
- TTL = 8-bit time to live

Label Features

Label Stacking

- Packets may carry multiple labels (stack of labels)
- Forwarding decision based on the label on the top of the stack
- The bottom of stack bit indicates the bottom of the stack

Time to Live

- Labels carry a TTL field to protect against loops
- TTL can be copied back into the IP header on the egress MPLS switch

MPLS Terminology

- LSR = Label Switch Router
  A router/switch supporting MPLS label switching.
- LSP = Label Switched Path
  A label switched path of LSRs from an ingress edge LSR to an egress edge LSR.
- LDP = Label Distribution Protocol
  A signalling protocol to setup MPLS forwarding tables by negotiating labels between LSRs.

Generalized Multiprotocol Label Switching

Generalized MPLS (GMPLS)

- Generalized MPLS (GMPLS) extends MPLS functionality by establishing and provisioning paths for:
  - Time Division Multiplexing (TDM) paths, where time slots are the labels (SONET).
  - Frequency Division Multiplexing (FDM) paths, where electromagnetic frequency is the label (light waves).
  - Space division multiplexed paths, where the label indicates the physical position of data (Photonic Cross-connect).
- GMPLS is particulary important for mapping traffic to different wavelengths on optical networks.

References I

W. Stallings.

MPLS.

The Internet Protocol Journal, 4(3):2–14, September 2001.

E. Rosen, A. Viswanathan, and R. Callon.

Multiprotocol Label Switching Architecture.

RFC 3031, Cisco Systems, Force10 Networks, Juniper Networks, January 2001.

E. Mannie.

Generalized Multi-Protocol Label Switching (GMPLS) Architecture.

RFC 3945, October 2004.

L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas.

LDP Specification.

RFC 3036, Nortel Networks, Ennovate Networks, IBM, PhotonEx, Cisco Systems, January 2001.

A. Viswanathan, N. Feldman, Z. Wang, and R. Callon.

Evolution of Multiprotocol Label Switching.

IEEE Communications Magazine, 36(5):165–173, May 1998.

A. Banerjee, J. Drake, J. Land, B. Turner, D. Awduche, L. Berger, K. Kompella, and Y. Rekther.

Generalized Multiprotocol Label Switching: An Overview of Signaling Enhancements and Recovery Techniques.

IEEE Communications Magazine, 39(1):144–151, January 2001.

Part: Security Protocols

Internet Protocol Security (IPsec)

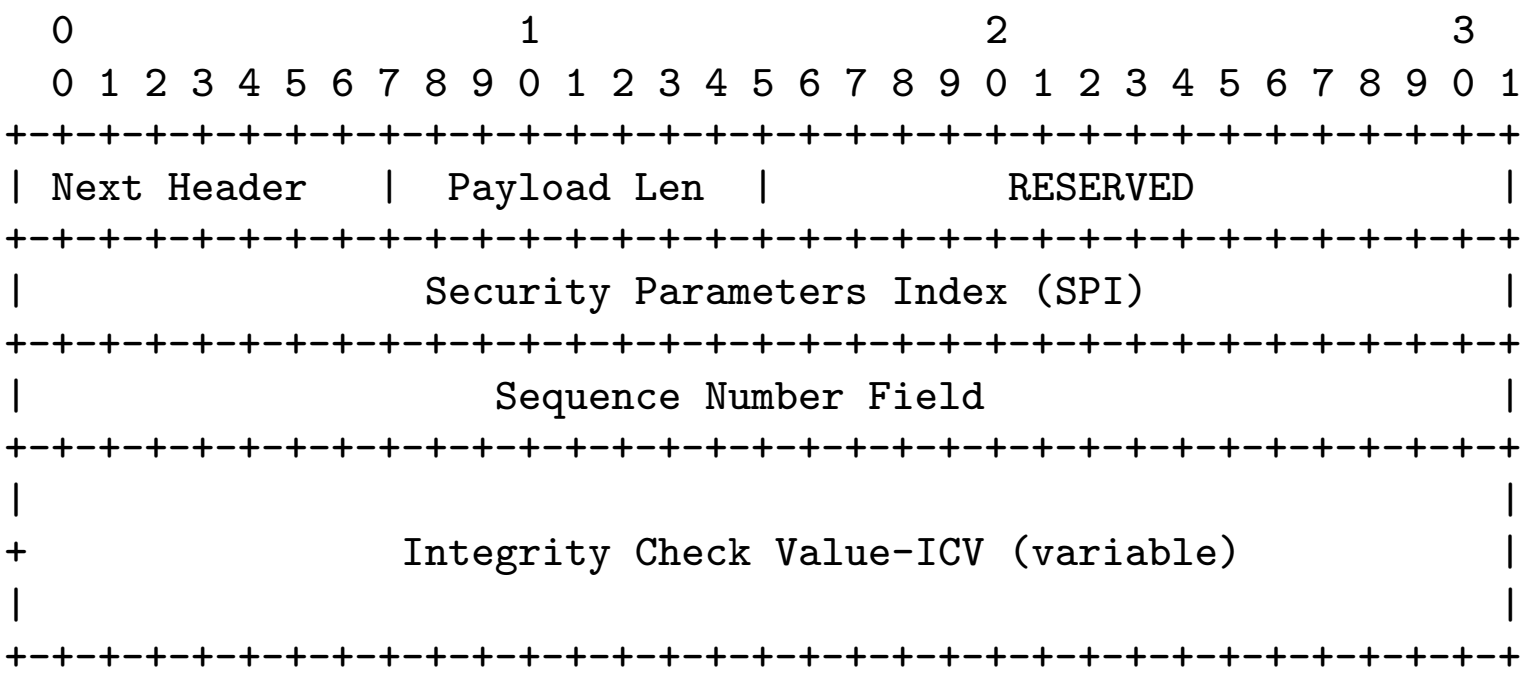
IPsec in a Nutshell

- Goal: Secure packets at the IP network layer.
- Modes:
  - Transport mode:
    The IPsec header is inserted just after the IP header.
  - Tunnel mode:
    A complete IP packet is encapsulated in a new IP packet.
- A Security Association (SA) is a simplex association between two IPsec endpoints.
- For duplex unicast communication, two SAs must be established.

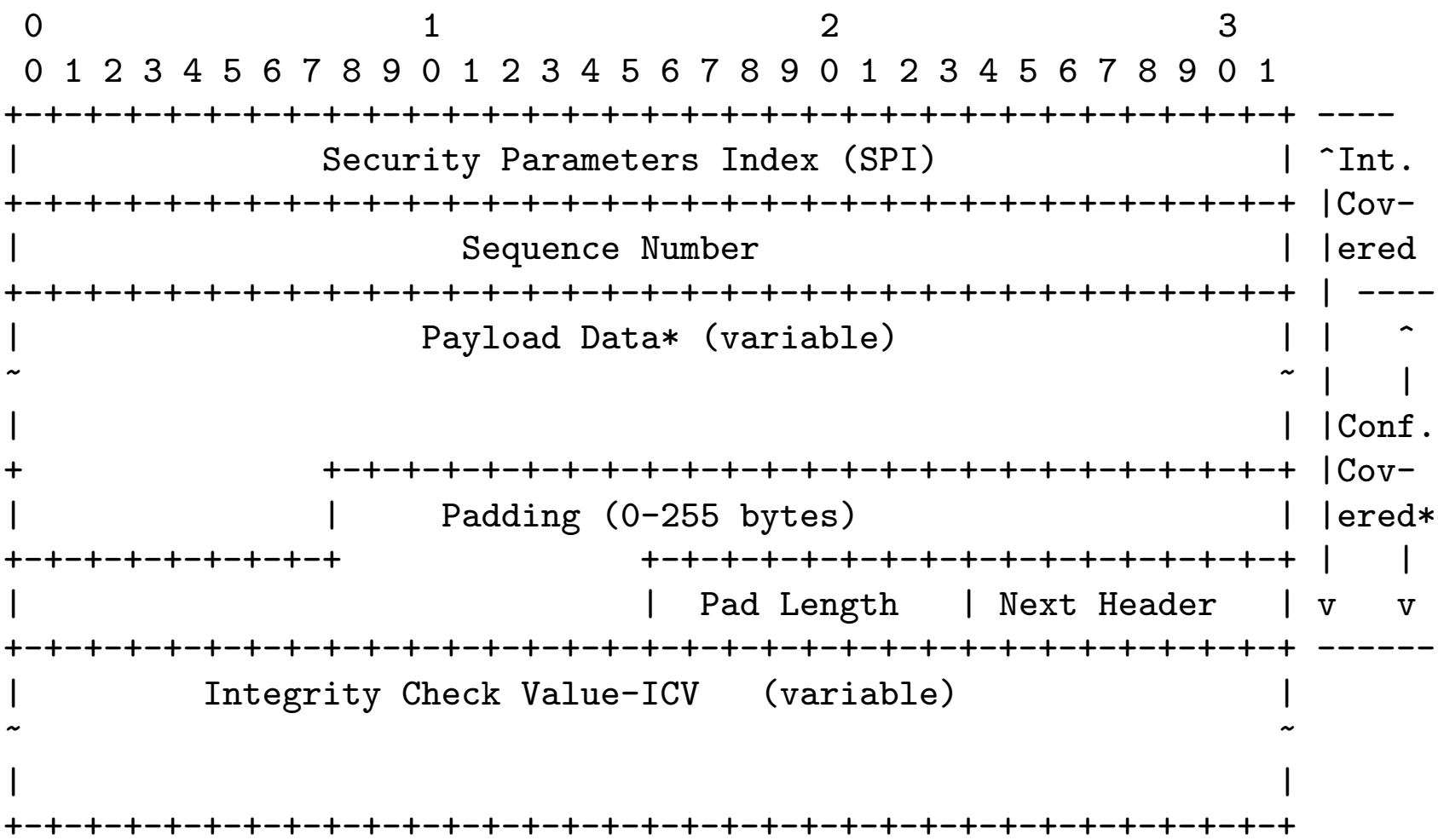
IPsec Headers

- The IP Authentication Header (AH) offers integrity and data origin authentication, with optional (at the discretion of the receiver) anti-replay features.
- The Encapsulating Security Payload (ESP) protocol offers the same set of services, and also offers confidentiality.
- Both AH and ESP offer access control, enforced through the distribution of cryptographic keys and the management of traffic flows as dictated by the Security Policy Database (SPD).
- The index into the SPD is called the SPI and typically used to identify SAs.

IPsec AH (RFC 4302)



IPsec ESP (RFC 4303)



IPsec Keying

- Manual keying (not really recommended)
- Automatic keying with the Internet Key Exchange (IKEv2) protocol (version 2).
  - IKEv2 itself uses the Diffie-Hellman algorithm plus certificates.
  - IKEv2 is a simplified version IKEv1 (which happened to be too complicated).
- No widespread deployment of an automatic keying infrastructure and as a consequence transport-mode IPsec is not widely used on the open Internet.

Manual Keying Example

```
#!/usr/sbin/setkey -f

# Clear the sa database and the spd database
flush;
spdflush;

# Traffic going from 212.201.49.188 to 10.70.17.11 needs an AH signed
# using HMAC-SHA1 using secret 12345678901234567890
add 212.201.49.188 10.70.17.11 ah 15700 -A hmac-sha1 "12345678901234567890";

# Traffic going from 212.201.49.188 to 10.70.17.11 needs encryption
# using 3des-cbc with key 123456789012123456789012'
add 212.201.49.188 10.70.17.11 esp 15701 -E 3des-cbc "123456789012123456789012"

# Traffic going out to 10.70.17.11 must be encrypted and be wrapped
# in an AH authentication header.
spdadd 212.201.49.188 10.70.17.11 any -P out ipsec
    esp/transport//require
    ah/transport//require;
```

Manual Keying Example

```
#!/usr/sbin/setkey -f

# Clear the sa database and the spd database
flush;
spdflush;

# Traffic going from 212.201.49.188 to 10.70.17.11 needs an AH signed
# using HMAC-SHA1 using secret 12345678901234567890
add 212.201.49.188 10.70.17.11 ah 15700 -A hmac-sha1 "12345678901234567890";

# Traffic going from 212.201.49.188 to 10.70.17.11 needs encryption
# using 3des-cbc with key 123456789012123456789012'
add 212.201.49.188 10.70.17.11 esp 15701 -E 3des-cbc "123456789012123456789012"

# Traffic coming in from 212.201.49.188 must be encrypted and wrapped
# in an AH authentication header.
spdadd 212.201.49.188 10.70.17.11 any -P in ipsec
    esp/transport//require
    ah/transport//require;
```

Transport Layer Security (TLS)

Transport Layer Security

- Transport Layer Security (TLS), formerly known as Secure Socket Layer (SSL), was created by Netscape in order to secure data transfers on the Web.
- As a user-space implementation, TLS can be shipped with applications and does not require operating system support.
- TLS uses X.509 certificates to authenticate servers and clients (although client authentication is often not used at the TLS layer).
- X.509 certificates more or less require a public key infrastructure including revocation server.
- TLS is widely used to secure application protocols running over TCP (e.g., http, smtp, ftp, telnet, imap, IIOP, ...)

X.509 Certificate ASN.1 Definition

```
Certificate  ::=  SEQUENCE  {
     tbsCertificate       TBSCertificate,
     signatureAlgorithm   AlgorithmIdentifier,
     signatureValue       BIT STRING  }

TBSCertificate  ::=  SEQUENCE  {
     version         [0]  EXPLICIT Version DEFAULT v1,
     serialNumber         CertificateSerialNumber,
     signature            AlgorithmIdentifier,
     issuer               Name,
     validity             Validity,
     subject              Name,
     subjectPublicKeyInfo SubjectPublicKeyInfo,
     issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version MUST be v2 or v3
     subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version MUST be v2 or v3
     extensions      [3]  EXPLICIT Extensions OPTIONAL
                          -- If present, version MUST be v3
     }
```

X.509 Certificate ASN.1 Definition

```
Version  ::=  INTEGER  {  v1(0), v2(1), v3(2)  }

CertificateSerialNumber  ::=  INTEGER

Validity ::= SEQUENCE {
     notBefore      Time,
     notAfter       Time }

Time ::= CHOICE {
     utcTime        UTCTime,
     generalTime    GeneralizedTime }

UniqueIdentifier  ::=  BIT STRING

SubjectPublicKeyInfo  ::=  SEQUENCE  {
     algorithm            AlgorithmIdentifier,
     subjectPublicKey     BIT STRING  }

Extensions  ::=  SEQUENCE SIZE (1..MAX) OF Extension

Extension  ::=  SEQUENCE  {
     extnID      OBJECT IDENTIFIER,
     critical    BOOLEAN DEFAULT FALSE,
     extnValue   OCTET STRING
                 -- contains the DER encoding of an ASN.1 value
                 -- corresponding to the extension type identified
                 -- by extnID
     }
```

X.509 Subject Alternative Name Extension

```
id-ce-subjectAltName OBJECT IDENTIFIER ::=  { id-ce 17 }

SubjectAltName ::= GeneralNames

GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName

GeneralName ::= CHOICE {
     otherName                       [0]     OtherName,
     rfc822Name                      [1]     IA5String,
     dNSName                         [2]     IA5String,
     x400Address                     [3]     ORAddress,
     directoryName                   [4]     Name,
     ediPartyName                    [5]     EDIPartyName,
     uniformResourceIdentifier       [6]     IA5String,
     iPAddress                       [7]     OCTET STRING,
     registeredID                    [8]     OBJECT IDENTIFIER }

OtherName ::= SEQUENCE {
     type-id    OBJECT IDENTIFIER,
     value      [0] EXPLICIT ANY DEFINED BY type-id }

EDIPartyName ::= SEQUENCE {
     nameAssigner            [0]     DirectoryString OPTIONAL,
     partyName               [1]     DirectoryString }
```

TLS Record Protocol

Record Protocol

The record protocol takes messages to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, and transmits the result. Received data is decrypted, verified, decompressed, reassembled, and then delivered to higher-level clients.

- The record layer is used by the handshake protocol, the change cipher spec protocol, the alert protocol, and the application data protocol.
- The fragmentation and reassembly provided does not preserve application message boundaries.

TLS Handshake Protocol

Handshake Protocol

The handshake protocol establishes cryptographic parameters:

- Exchange messages to agree on algorithms, exchange random values, and check for session resumption.
- Exchange the necessary cryptographic parameters to allow the client and server to agree on a premaster secret.
- Exchange certificates and cryptographic information to allow the client and server to authenticate themselves.
- Generate a master secret from the premaster secret and exchanged random values.
- Provide security parameters to the record layer.
- Allow the client and server to verify that their peer has calculated the same security parameters and that the handshake occurred without tampering by an attacker.

TLS Change Cipher Spec Protocol

Change Cipher Spec Protocol

The change cipher spec protocol is used to signal transitions in ciphering strategies.

- The protocol consists of a single ChangeCipherSpec message.
- This message is sent by both the client and the server to notify the receiving party that subsequent records will be protected under the newly negotiated CipherSpec and keys.

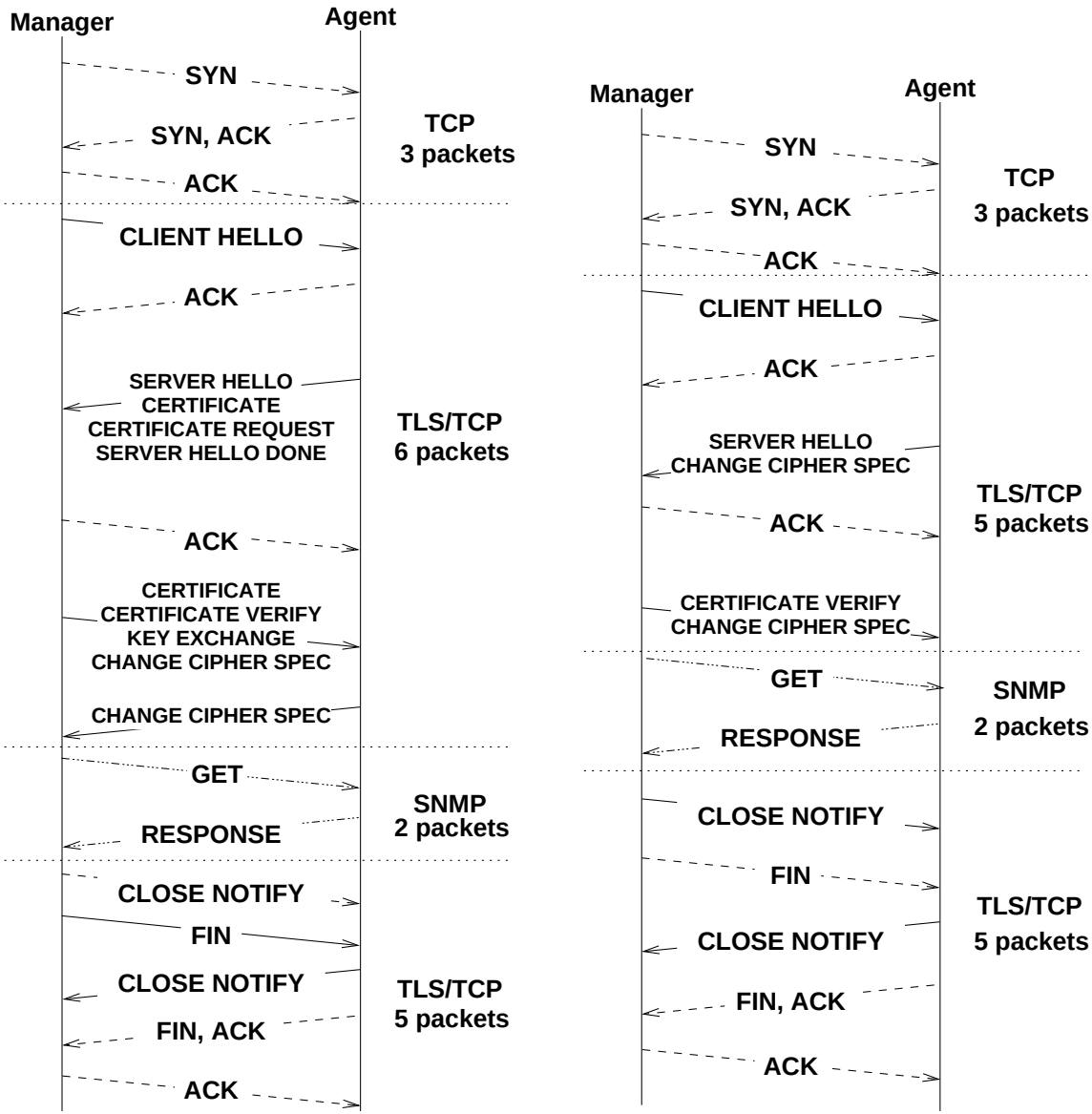
TLS Alert Protocol

Alert Protocol

The alert protocol is used to signal exceptions (warnings, errors) that occured during the processing of TLS protocol messages. An alert has an alert level and an alert description (enumeration).

- The alert protocol is used to properly close a TLS connection by exchanging `close_notify` alert messages.
- The closure exchange allows to detect truncation attacks.

Details of an SNMP GET Operation over TLS



Secure Shell (SSH)

Secure Shell (SSH)

- SSH provides a secure connection through which user authentication and several inner protocols can be run.
- The general architecture of SSH is defined in RFC 4251.
- SSH was initially developed by Tatu Ylonen at the Helsinki University of Technology in 1995, who later founded SSH Communications Security.
- SSH was quickly adopted as a replacement for insecure remote login protocols such as telnet or rlogin/rsh.
- Several commercial and open source implementations are available running on almost all platforms.
- SSH is a Proposed Standard protocol of the IETF since 2006.

SSH Protocol Overview

SSH Protocol Layers

1. The **Transport Layer Protocol** provides server authentication, confidentiality, and integrity with perfect forward secrecy
2. The **User Authentication Protocol** authenticates the client-side user to the server
3. The **Connection Protocol** multiplexes the encrypted data stream into several logical channels

$\Rightarrow$ SSH authentication is not symmetric!

$\Rightarrow$ The SSH protocol is designed for clarity, not necessarily for efficiency (shows its academic roots)

SSH Terminology

Host Key

Every machine must have a public/private host key pair. Host Keys are often identified by their fingerprint.

User Key

Users may have their own public/private key pairs.

User Password

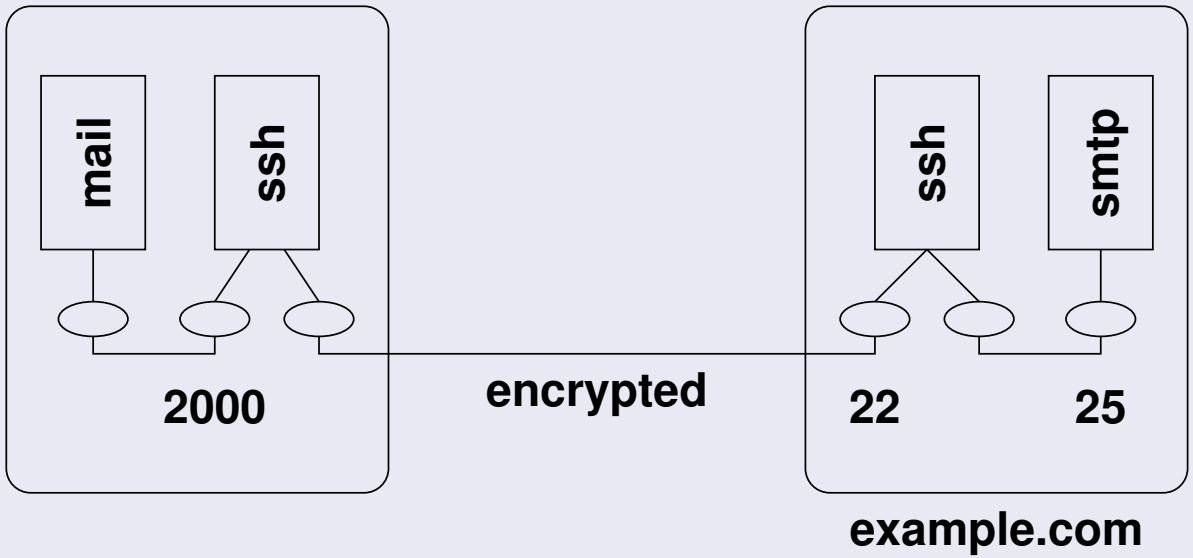
Accounts may have passwords to authenticate users.

Passphrase

The storage of a user's private key may be protected by a passphrase.

SSH Features: TCP Forwarding

```
ssh -f joe@example.com -L 2000:example.com:25 -N
```

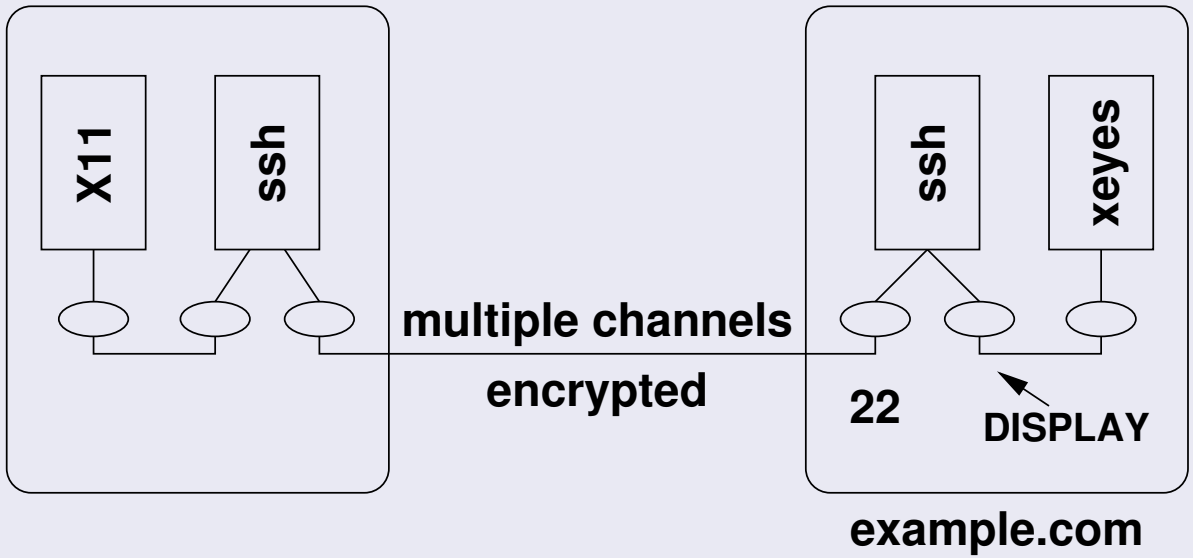


TCP Forwarding

TCP forwarding allows users to tunnel unencrypted traffic through an encrypted SSH connection.

SSH Features: X11 Forwarding

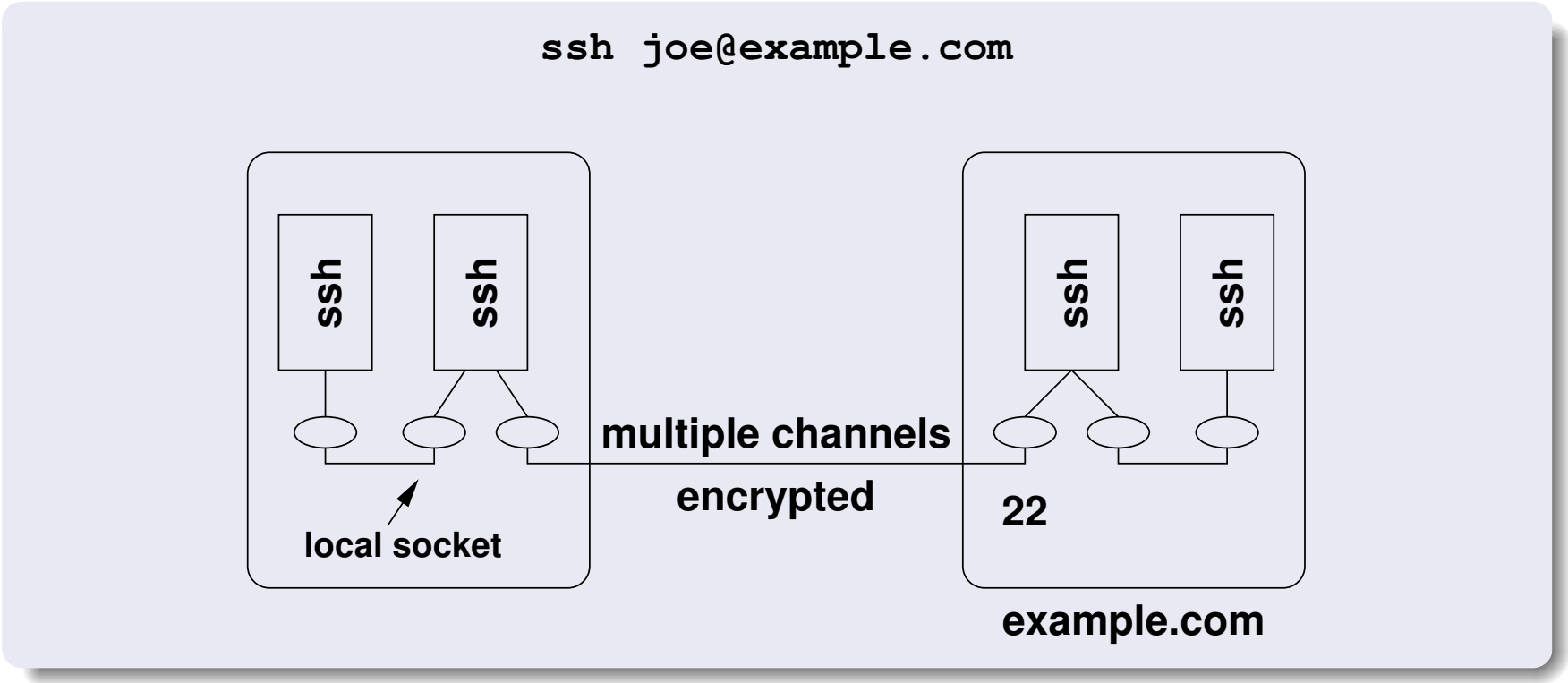
```
ssh -X joe@example.com
```



X11 Forwarding

X11 forwarding is a special application of TCP forwarding allowing X11 clients on remote machines to access the local X11 server (managing the display and the keyboard/mouse).

SSH Features: Connection Sharing

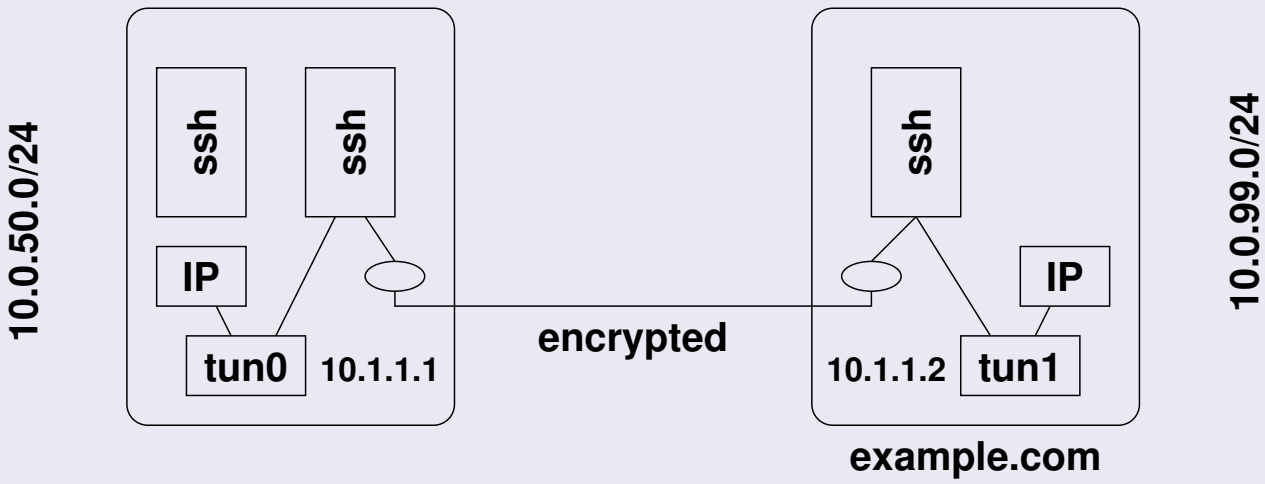


Connection Sharing

New SSH connections hook as a new channel into an existing SSH connection, reducing session startup times (speeding up shell features such as tab expansion).

SSH Features: IP Tunneling

```
ssh -f -w 0:1 example.com
```



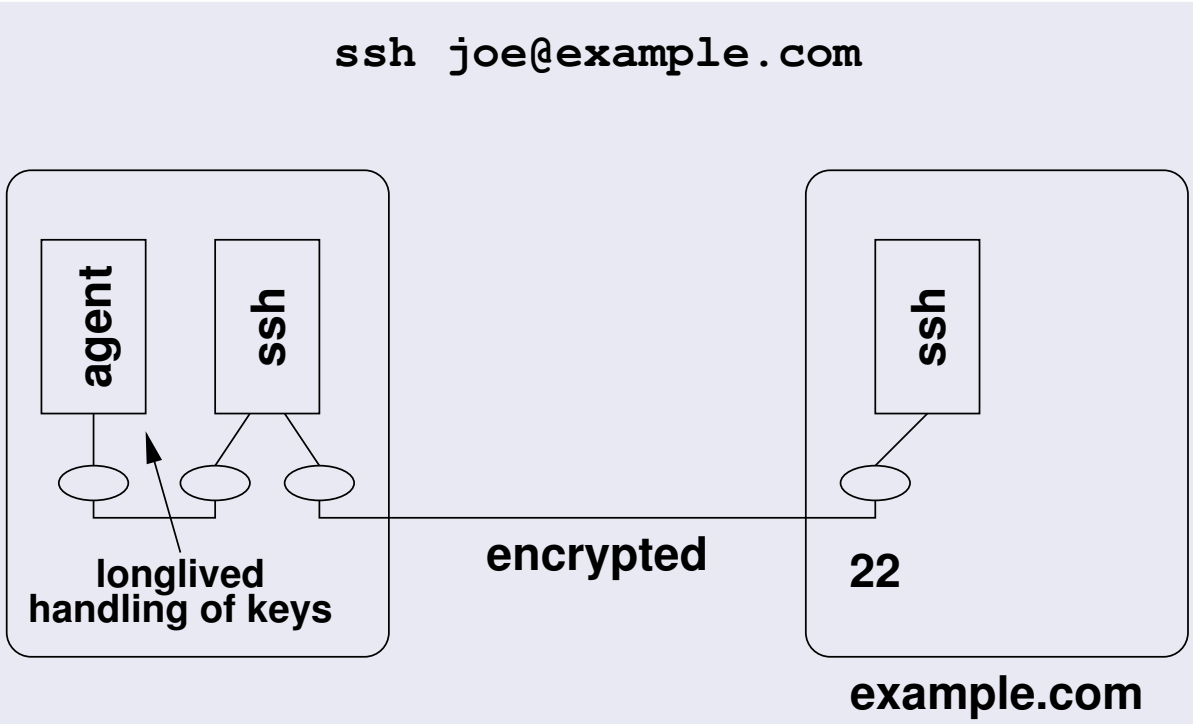

```
ifconfig tun0 10.1.1.1 10.1.1.2 \
        netmask 255.255.255.255
route add 10.0.99.0/24 10.1.1.2
```

```
ifconfig tun0 10.1.1.2 10.1.1.1 \
        netmask 255.255.255.255
route add 10.0.50.0/24 10.1.1.1
```

IP Tunneling

Tunnel IP packets over an SSH connection by inserting tunnel interfaces into the kernels and by configuring IP forwarding.

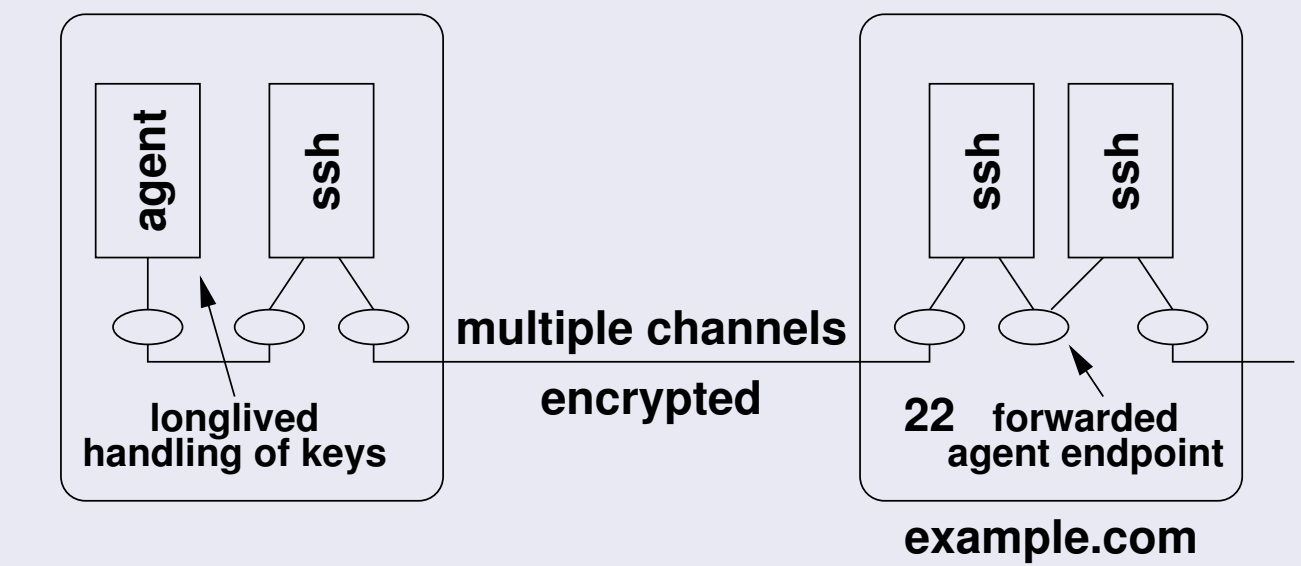
SSH Features: SSH Agent



SSH Agent

Maintains client credentials during a login session so that credentials can be reused by different SSH invocations without further user interaction.

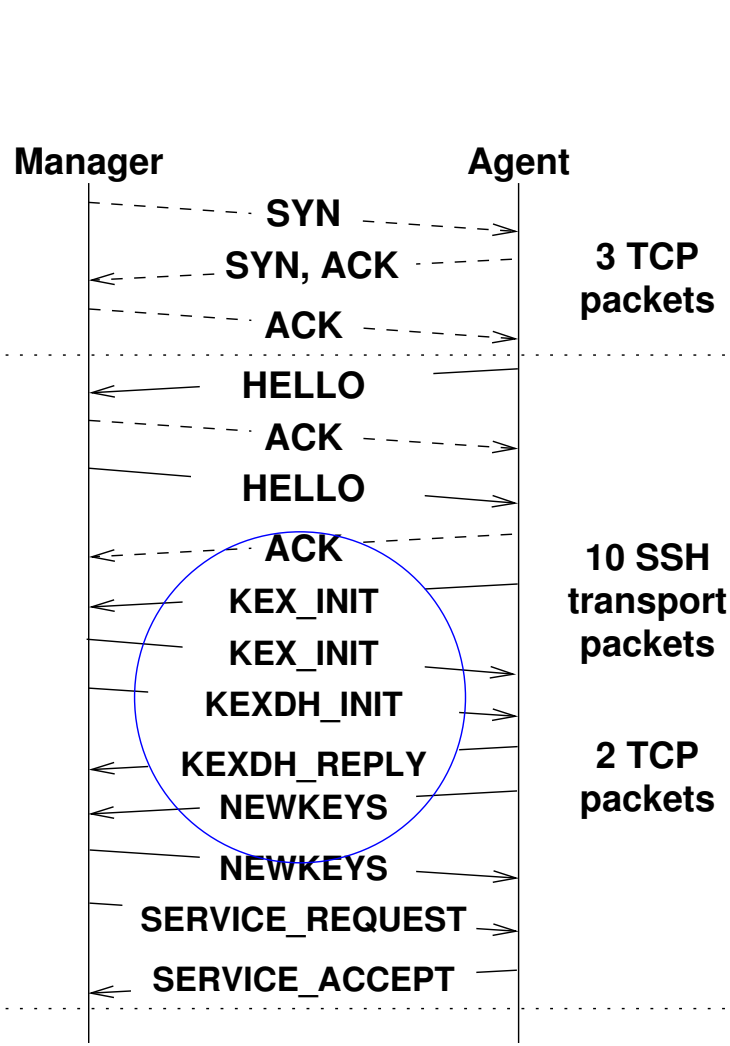
SSH Features: SSH Agent Forwarding

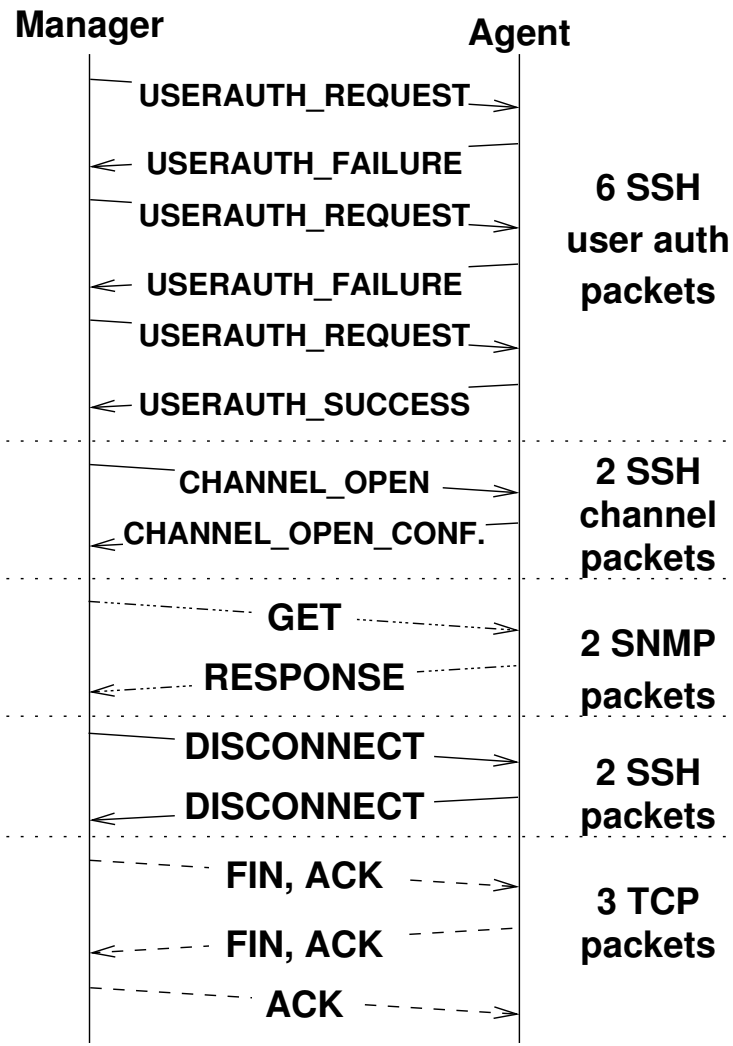


SSH Agent Forwarding

An SSH server emulates an SSH Agent and forwards requests to the SSH Agent of its client, creating a chain of SSH Agent delegations.

Details of an SNMP GET Operation over SSH





SSH Transport Protocol

- Transport Protocol (RFC 4253) provides
  - strong encryption,
  - server authentication,
  - integrity protection, and
  - optionally compression.
- SSH transport protocol typically runs over TCP
- 3DES (required), AES128 (recommended)
- HMAC-SHA1 (recommended)
- Automatic key re-exchange, usually after 1 GB of data have been transferred or after 1 hour has passed, whichever is sooner.

SSH Key Exchange

- The SSH host key keyex identifies a server by its hostname or IP address and possibly port number.
- Other keyex mechanisms use different naming schemes for a host.
- Different key exchange algorithms
  - Diffie-Hellman style key exchange
  - GSS-API style key exchange
- Different Host key algorithms
  - Host key used to authenticate key exchange
  - SSH RSA and DSA keys
  - X.509 (under development)

SSH User Authentication

- Executes after transport protocol initialization (key exchange) to authenticate client.
- Authentication methods:
  - Password (classic password authentication)
  - Interactive (challenge response authentication)
  - Host-based (uses host key for user authentication)
  - Public key (usually DSA or RSA keypairs)
  - GSS-API (Kerberos / NETLM authentication)
  - X.509 (under development)
- Authentication is client-driven.

SSH Connection Protocol

- Allows opening of multiple independent channels.
- Channels may be multiplexed in a single SSH connection.
- Channel requests are used to relay out-of-band channel specific data (e.g., window resizing information).
- Channels commonly used for TCP forwarding.

OpenSSH Implementation

Privilege Separation

Privilege separation is a technique in which a program is divided into parts which are limited to the specific privileges they require in order to perform a specific task.

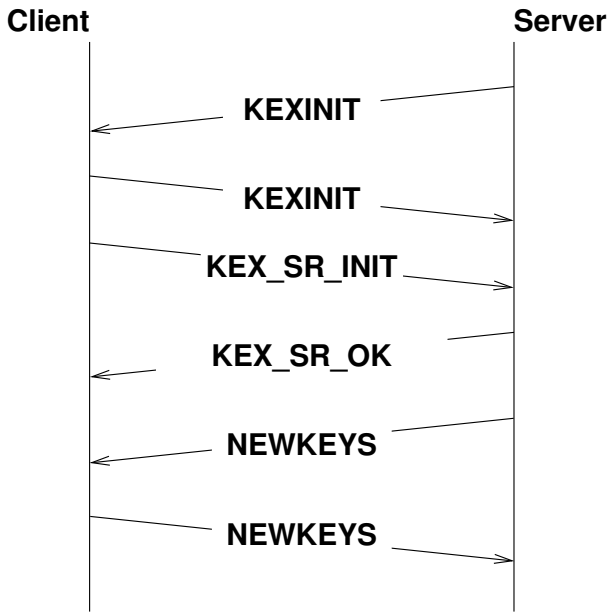
- OpenSSH is using two processes: one running with special privileges and one running under normal user privileges
- The process with special privileges carries out all operations requiring special permissions.
- The process with normal user privileges performs the bulk of the computation not requiring special rights.
- Bugs in the code running with normal user privileges do not give special access rights to an attacker.

Performance of Short Lived Sessions

Protocol	Time (meat) [ms]			Time (turtle) [ms]			Data [bytes]	Packets
	min	avg	max	min	avg	max		
v1/CSM/UDP/nn	0.24	0.25	0.29	0.85	0.95	1.43	292	2
v1/CSM/TCP/nn	0.39	0.40	0.43	1.27	1.38	1.72	1012	10
v2/CSM/UDP/nn	0.24	0.25	0.30	0.85	0.96	1.50	292	2
v2/CSM/TCP/nn	0.46	0.48	0.58	1.28	1.46	2.40	1012	10
v3/USM/UDP/nn	0.48	0.48	0.54	1.75	1.84	1.95	718	4
v3/USM/TCP/nn	0.63	0.64	0.69	2.22	2.46	9.59	1490	12
v3/USM/UDP/an	0.50	0.63	0.87	1.79	1.89	2.34	742	4
v3/USM/TCP/an	0.65	0.66	0.70	2.21	2.31	2.48	1514	12
v3/USM/UDP/ap	0.51	0.52	0.59	1.88	2.05	4.17	763	4
v3/USM/TCP/ap	0.66	0.68	0.71	2.31	2.42	2.60	1535	12
v3/TSM/SSH/ap	**13.49**	**13.73**	**14.20**	**107.35**	**110.45**	**144.33**	**5310**	**31**
v3/TSM/TLS/ap	**11.01**	**11.15**	**12.57**	**67.44**	**68.70**	**86.59**	**4107**	**16**
v3/TSM/DTLS/ap	10.89	11.05	12.00	67.68	69.96	155.10	3457	8
v3/TSM/TLSsr/ap	**2.23**	**2.27**	**2.45**	**5.47**	**5.72**	**6.28**	**1457**	**15**

- SSH (TLS/DTLS) transports behave like a DoS attack for short-lived SNMP sessions (e.g., shell scripts)
- TLS's session resumption mechanism cures the problem
- How can we do session resumption with SSH?

Session Resumption Key Exchange



- Server maintains session state for recently closed sessions
- Client and server perform session resumption by using of a session resumption key exchange algorithm
- SSH's algorithm negotiation feature handles this nicely

Session Resumption with Server Side State

Algorithm (Server Side State)

C: Client sends the session identifier and a MAC computed over the session keys to the server in a `SSH2_MSG_KEXSR_INIT` message

S: Server looks up the cached session and verifies the MAC

- If successful, it returns an `SSH2_MSG_KEX_SR_OK` message, followed by a standard `SSH2_MSG_NEWKEYS` exchange
- On failure, `SSH2_MSG_KEX_SR_ERROR` is sent and key exchange proceeds with another key exchange algorithm, or fails

\+ Simple design and easy to implement

– Server has to maintain session state (scalability)

Session Resumption with Client Side State

Algorithm (Client Side State)

S: After key (re)negotiation, the server sends an encrypted ticket in a `SSH2_MSG_KEX_SR_TICKET` message

C: The client sends the encrypted ticket and a MAC computed over the session identifier to the server in a `SSH2_MSG_KEXSR_INIT` message

S: The server decrypts the ticket and verifies the MAC

- If successful, it returns an `SSH2_MSG_KEX_SR_OK` message, followed by a standard `SSH2_MSG_NEWKEYS` exchange.
- On failure, `SSH2_MSG_KEX_SR_ERROR` is sent and key exchange proceeds with another key exchange algorithm, or fails.

\+ Server side state reduced to a key for encrypting tickets

TicketContent Data Structure

```
struct TicketEnc {
    char* name;
    u_char* key;
    u_char* iv;
};

struct TicketMac {
    char* name;
    u_char* key;
};
```

```
struct TicketContent {
    u_char* session_id;
    u_int session_id_len;
    TicketEnc tenc_ctos;
    TicketEnc tenc_stoc;
    TicketMac tmac_ctos;
    TicketMac tmac_stoc;
    char* tcomp_ctos;
    char* tcomp_stoc;
    int hostkey_type;
    char* client_version_string;
    char* server_version_string;
};
```

- SSH allows to use different algorithms in each direction!

Ticket Data Structure

```
struct Ticket {
    u_int seq_nr;
    u_char* id;
    u_char* enc_ticket;
    u_int enc_ticket_len;
    int64_t time_stamp;
};
```

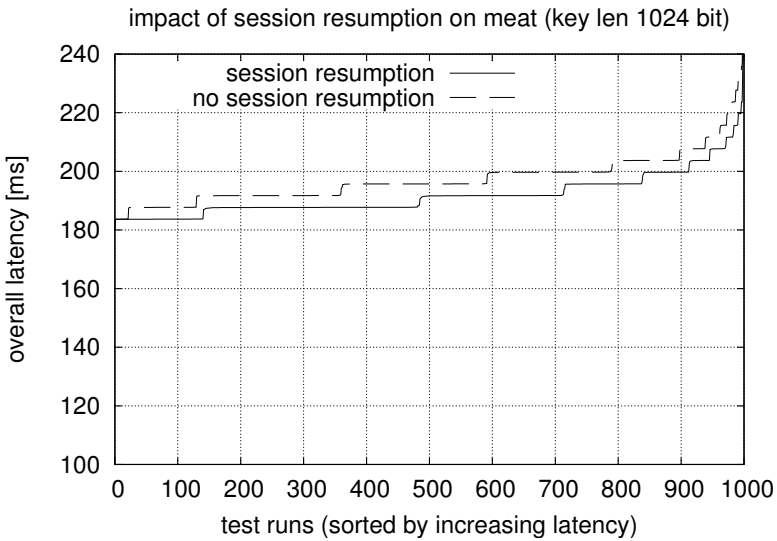
- Contains the encrypted `TicketContent` data structure in `enc_ticket`
- The `id` uniquely identifiers a ticket
- The `seq_nr` and `time_stamp` fields can be used to quickly discard outdated tickets
- Encryption key and its IV are generated at server start-up

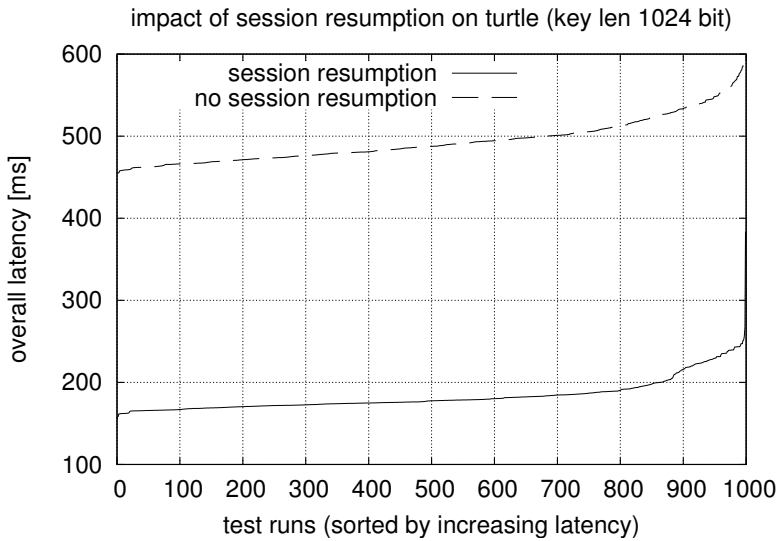
Performance Evaluation

Name	CPUs	RAM	Ethernet	Kernel
`meat`	2 Xeon 3 GHz	2 GB	1 Gbps	2.6.16.29
`veggie`	2 Xeon 3 GHz	1 GB	1 Gbps	2.6.16.29
`turtle`	1 Ultra Sparc IIi	128 MB	100 Mbps	2.6.20

- SSH client: `veggie` / SSH server: `meat` and `turtle`
- Measuring overall execution time of "`ssh $host exit`"
- Used HMAC-MD5 hash function and AES-128 encryption
- Hosts and the network were idle during the experiments
- 1000 experiments, results sorted by the measured latency
- Absolute numbers irrelevant, look at relative numbers

Session Resumption Performance (key length 1024)

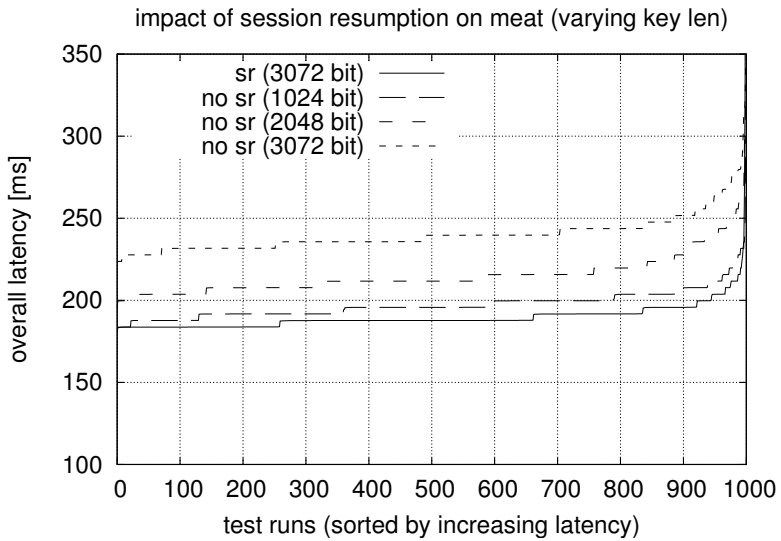




- With a key length of 1024 bits, the performance gain on an idle fast machine is observable but small
- With the same key length, the performance gain on a small idle machine is significant (factor 4)

$\Rightarrow$ Session resumption is particularly useful for processing power constrained low-end consumer /enterprise products

Impact of the Key Length on the Performance



- Session resumption performance is largely independent of the key length
- With increasing key length, the performance gain increases also on fast idle machines

$\Rightarrow$ Even on a fast processors, the performance gain is significant if you need long keys to achieve strong security

References I

S. Kent and K. Seo.

Security Architecture for the Internet Protocol.
RFC 4301, BBN Technologies, December 2005.

S. Kent.

IP Authentication Header.
RFC 4302, BBN Technologies, December 2005.

S. Kent.

IP Encapsulating Security Payload (ESP).
RFC 4303, BBN Technologies, December 2005.

T. Ylonen and C. Lonvick.

The Secure Shell (SSH) Protocol Architecture.
RFC 4251, SSH Communications Security Corp, Cisco Systems, January 2006.

T. Ylonen and C. Lonvick.

The Secure Shell (SSH) Authentication Protocol.
RFC 4252, SSH Communications Security Corp, Cisco Systems, January 2006.

T. Ylonen and C. Lonvick.

The Secure Shell (SSH) Transport Layer Protocol.
RFC 4253, SSH Communications Security Corp, Cisco Systems, January 2006.

T. Ylonen and C. Lonvick.

The Secure Shell (SSH) Connection Protocol.
RFC 4254, SSH Communications Security Corp, Cisco Systems, January 2006.

References II

J. Schönwälder, G. Chulkov, E. Asgarov, and M. Cretu.

Session Resumption for the Secure Shell Protocol.

In *Proc. 11th IFIP/IEEE International Symposium on Integrated Network Management (IM 2009)*, pages 157–163, May 2009.

T. Dierks and E. Rescorla.

The Transport Layer Security (TLS) Protocol Version 1.2.

RFC 5246, Independent, RTFM, August 2008.

E. Rescorla and N. Modadugu.

Datagram Transport Layer Security.

RFC 4347, RTFM, Stanford University, April 2006.

D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk.

Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.

RFC 5280, NIST, Microsoft, Trinity College Dublin, Entrust, Vigil Security, May 2008.

Part: Network Management and Measurement

41 Network Management Overview

42 Network Monitoring using SNMP

43 Network Configuration using NETCONF / YANG
- Configuration Management Approaches
- NETCONF Protocol Overview
- YANG Data Modeling Overview

44 System Logging Protocol (SYSLOG)

45 Traffic Analysis using NETFLOW / IPFIX

Network Management Overview

Why Network Management?

- Networks of non-trivial size need management:
  - Fault detection and isolation
  - Configuration generation and installation
  - Accounting data gathering
  - Performance monitoring and tuning
  - Security management (keys, access control)

$\Rightarrow$ FCAPS functional areas (very broad but widely accepted functional categorization)

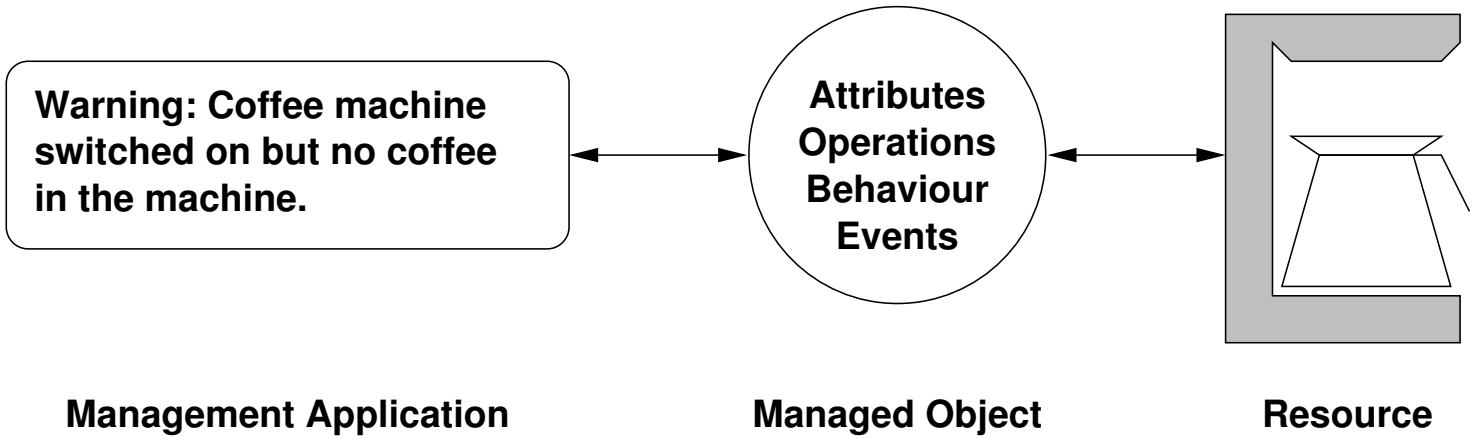
Why is Network Management Hard?

- Scalability is a key concern (millions of devices/users)
- Short technology life times (what happened to ATM?)
- Heterogenity requires standards-based solutions
- Lack of skilled persons

$\Longrightarrow$ But network management is not really fundamentally different from other complex control systems (e.g., systems that control robots in a vehicle fabric).

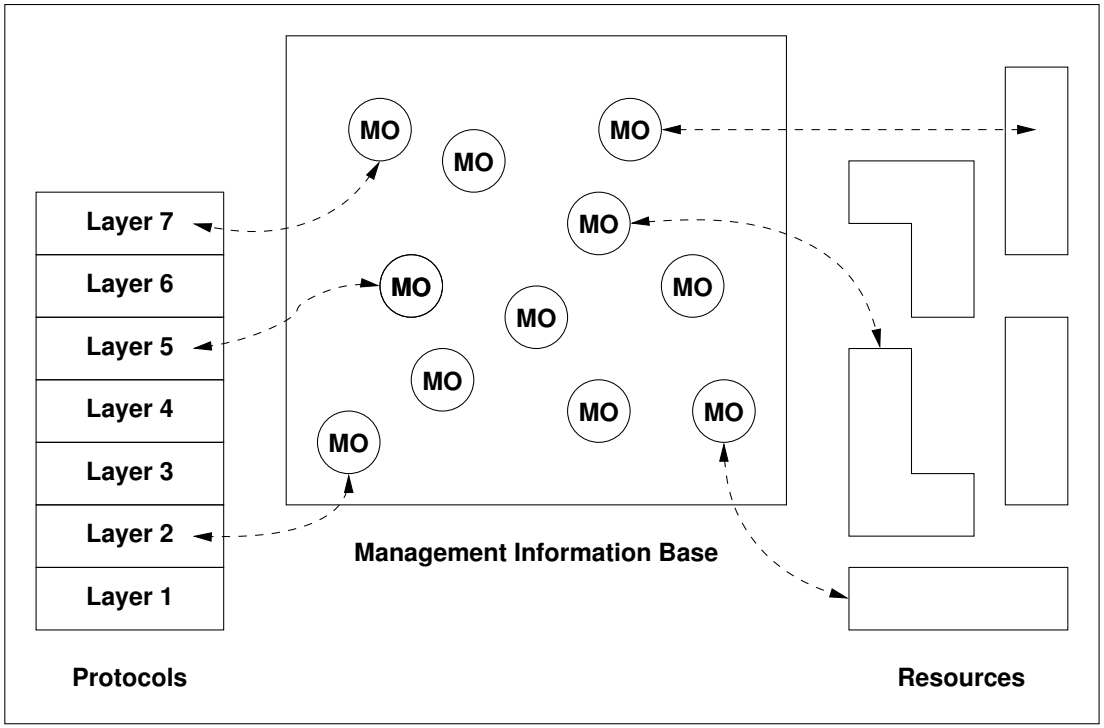
$\Longrightarrow$ However, network management terminology is often very different and sometimes somewhat confusing (especially for people with computer science background).

Abstraction of Managed Objects (MOs)



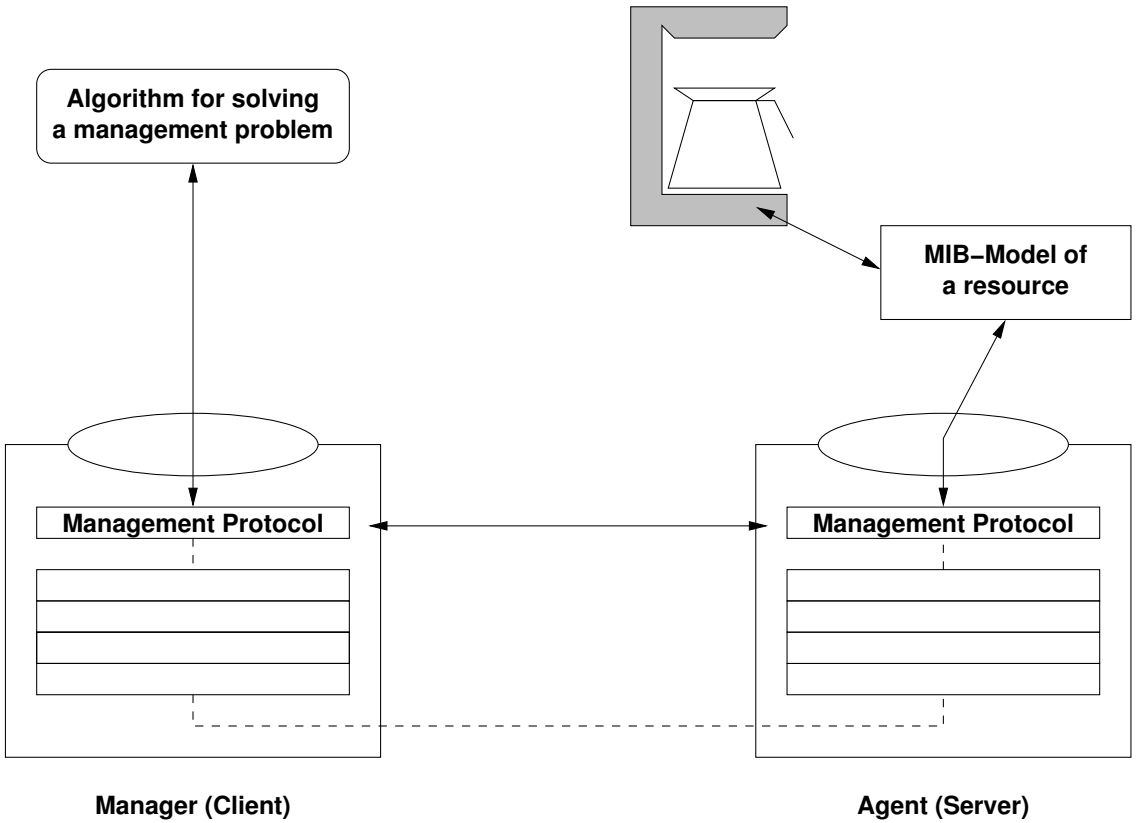
- A managed object is the abstracted view of a resource that presents its properties as seen by (and for the purpose of) management (ISO 7498-4).
- The boundary of a managed object defines the level of details which are accessible for management systems.

Management Information Base (MIB)



- The set of managed objects within a system, together with their attributes, constitutes that system's management information base (ISO 7498-4).

Management Protocols



- Management protocols realize the access to MOs contained in a MIB.

Data-centric Approach

- The device is represented as a collection of data objects representing all the properties and capabilities of a device.
- The management protocol manipulates the data objects representing a device and its state.
- Manipulation of data objects might cause side effects.

$\Rightarrow$ Example: Internet management (SNMP)

Command-centric Approach

- The device is considered to be a stateful black box.
- A sequence of commands can be send to the device to
  a) change the state of the device or to
  b) retrieve data about the current state of the device (or portions thereof).
- Determining the right sequence of commands to bring a device into a certain state might not be trivial.

$\Rightarrow$ Example: Command line interfaces of routers or switches

Object-centric Approach

- The device is represented as a collection of data objects with associated methods.
- This can be seen as a combination of the data- and the command-centric approaches.
- Usually leads to object-oriented modeling and thus object-oriented approaches.
- A critical design decision is the *granularity* of the objects and the level of *interdependencies* between objects

$\Rightarrow$ Example: OSI management (CMIP), DMTF information models

Document-centric Approach

- The configuration and state of a device is represented as a structured document.
- Management operations are realized by manipulating the structured document.
- Allows to use general document processors for management purposes.
- Closely related to data-centric approaches.

$\Rightarrow$ Example: Most XML-based management approaches follow this model.

Essential Management Protocol Primitives

- From a very abstract viewpoint, the following set of management protocol primitives is essential for data-centric or object-centric management protocols:
  - `GET`, `SET`
  - `CREATE`, `DELETE`
  - `SEARCH` (or at the very least `ITERATE`)
  - `LOCK`, `UNLOCK`, `COMMIT`, `ROLLBACK`
  - `NOTIFY`
  - `EXECUTE` an operation or `INVOKE` a method
- Command-centric protocols usually have a very rich set of primitives (which are often hierarchically structured).

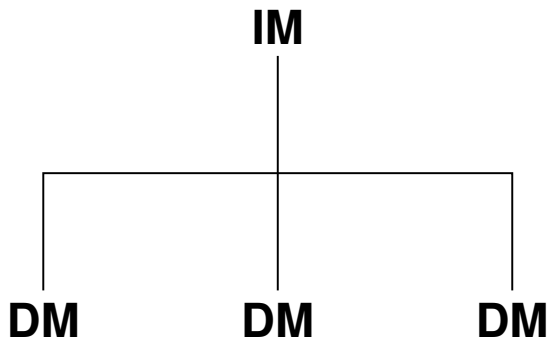
Information Models (RFC 3444)

- Information Models (IMs) are used to model managed objects at a conceptual level, independent of any specific protocols used to transport the data.
- The degree of specificity (or detail) of the abstractions defined in the IM depends on the modeling needs of its designers.
- In order to make the overall design as clear as possible, IMs should hide all protocol and implementation details.
- IMs focus on relationships between managed objects.
- IMs are often represented in Unified Modeling Language (UML) diagrams, but there are also informal IMs written in plain English language.

Data Models (RFC 3444)

- Data Models (DMs) are defined at a lower level of abstraction and include many details (compared to IMs).
- They are intended for implementors and include implementation- and protocol-specific constructs.
- DMs are often represented in formal data definition languages that are specific to the management protocol being used.

Information Models vs. Data Models

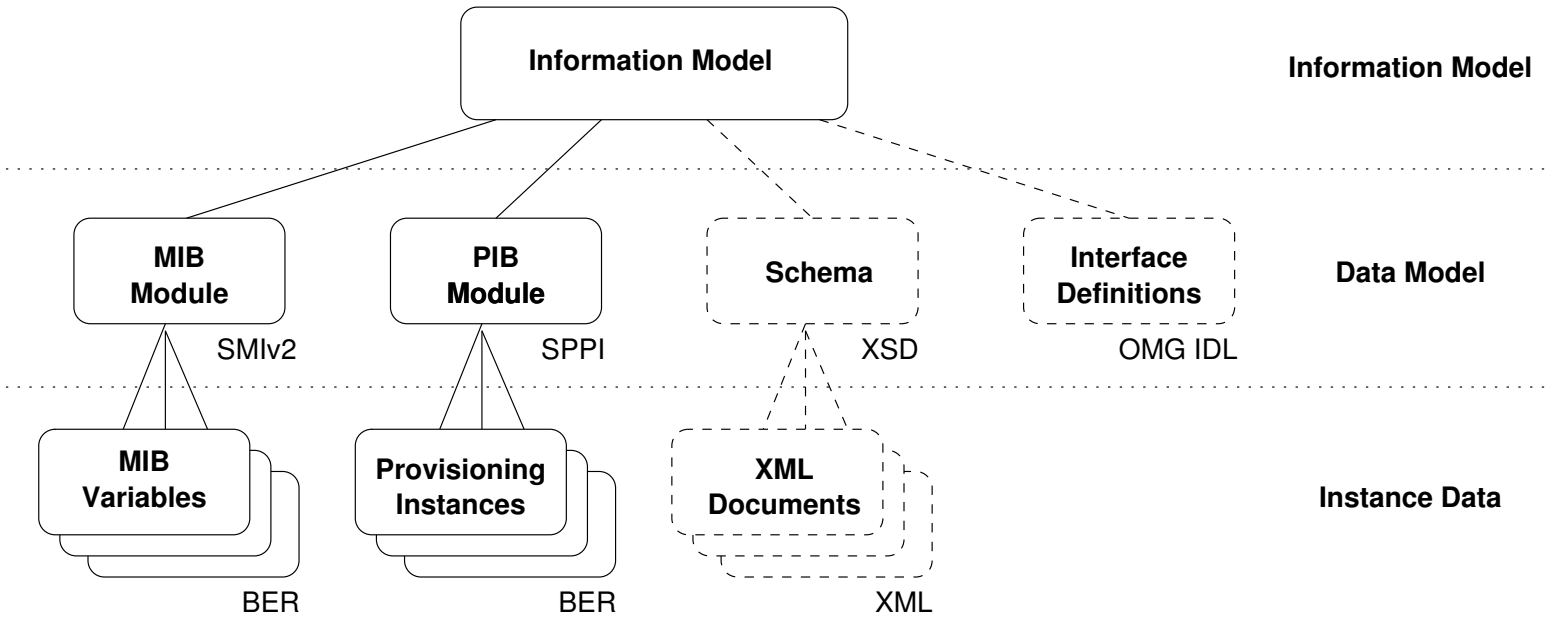


conceptual/abstract model for designers and operators

concrete/detailed model for implementors

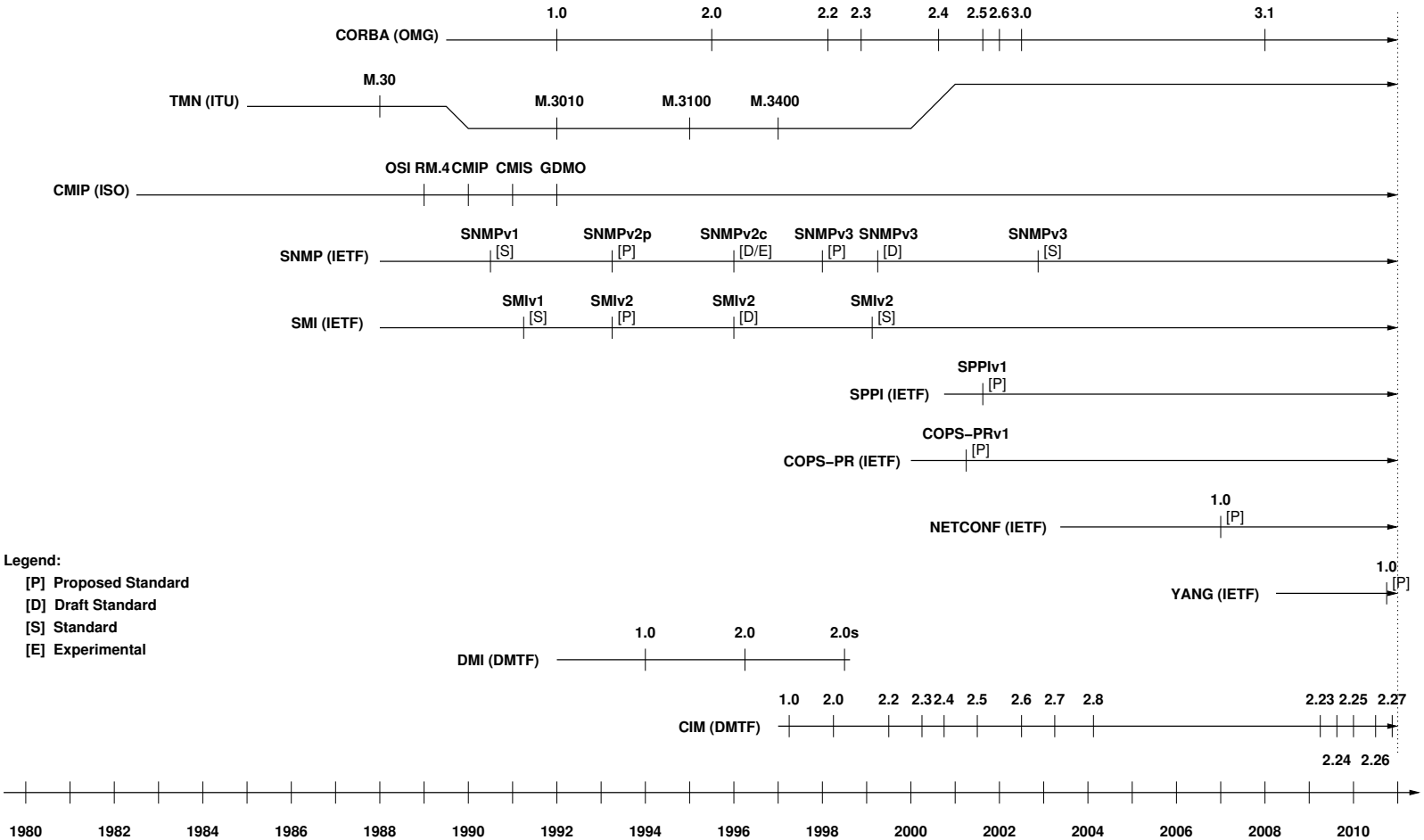
- Since conceptual models can be implemented in different ways, multiple DMs can be derived from a single IM.
- Although IMs and DMs serve different purposes, it is not always possible to decide which detail belongs to an IM and which detail belongs to a DM.
- Similarily, it is sometimes difficult to determine whether an abstraction belongs to an IM or a DM.

IMs and DMs in the Real World

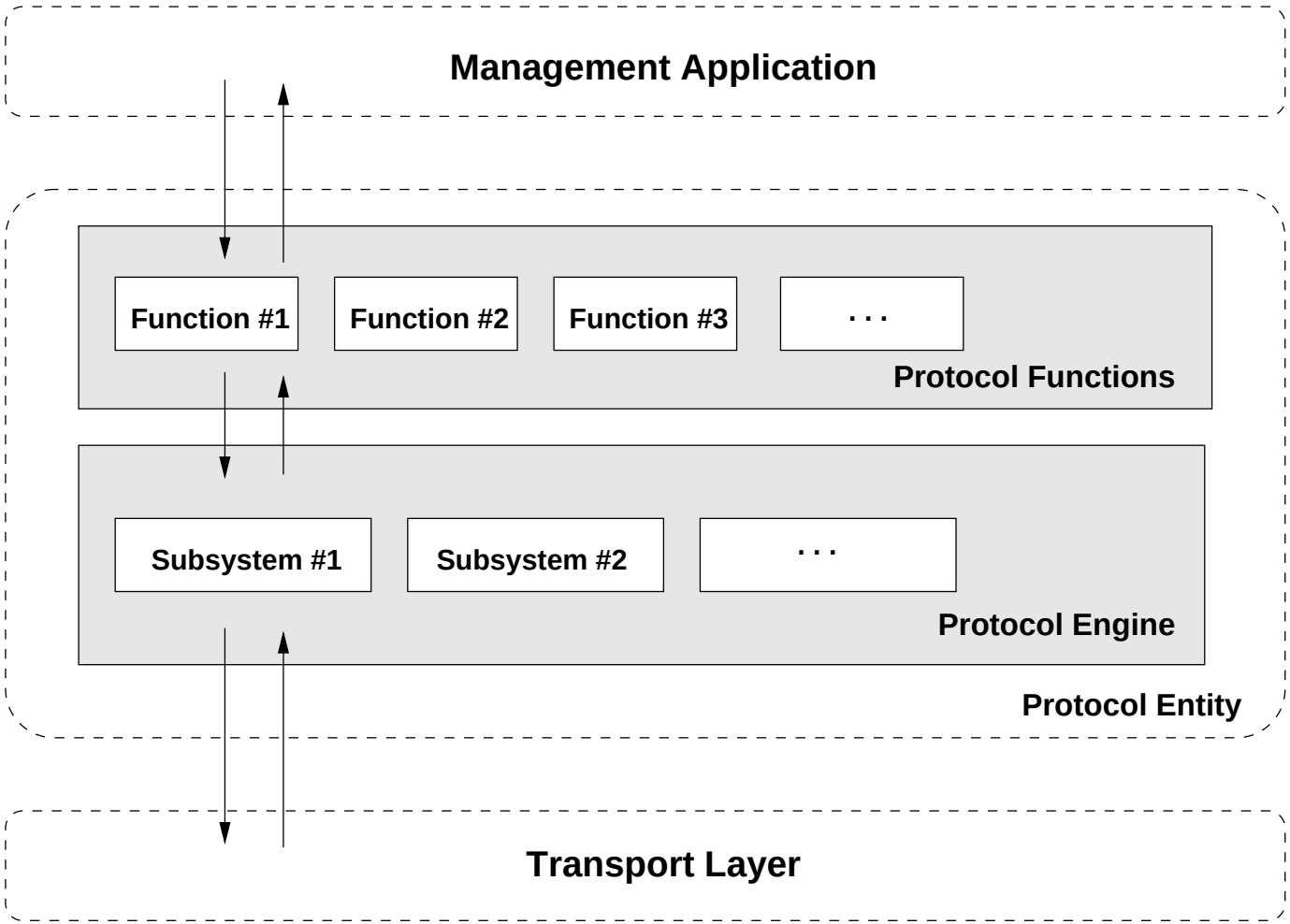


- The Architecture for Differentiated Services (RFC 2475) is an example for an informal definition of the DiffServ information model.
- The DiffServ MIB module (RFC 3289) and the DiffServ PIP module (RFC 3317) are examples of data models conforming to the DiffServ information model.

Network Management Standards



Architectural Concepts



Architectural Concepts

- An protocol *Entity* can be decomposed into a protocol engine and protocol functions.
- A protocol *Engine* is concerned with the handling of protocol messages.
- Protocol *Functions* realize the protocol's functionality.
- *Subsystems* provide a well defined functionality which is accessible through a defined subsystem interface.
- *Models* implement a subsystem interface. There may be one or multiple models that implement the same subsystem interface.

Naming and Addressing

- Naming of Information
  - Identification of a managed object within a naming scope
  - Names may be scoped by a protocol engines (which is identified by an address)
  - Globally unique names may be constructed from the pieces
  - Definition of the naming system is an essential design step
- Addressing of Protocol Engines
  - Use (extended?) transport addresses to identify protocol engines
  - Introduce a new address space for the protocol engines
  - Uniform Resource Identifiers (URIs)

Security and Access Control

- Message-based security
  - \+ Self-contained solution
  - \+ Reduced dependencies on the infrastructure
  - \- Complex specification and verification
  - \- Complex implementation and costs
- Session-based security
  - \+ Leveraging existing work
  - \+ Better integration with a security infrastructure
  - \- Dependency on a security infrastructure
- Explicit access control rules vs. implicit access control vs. no access control

Network Monitoring using SNMP

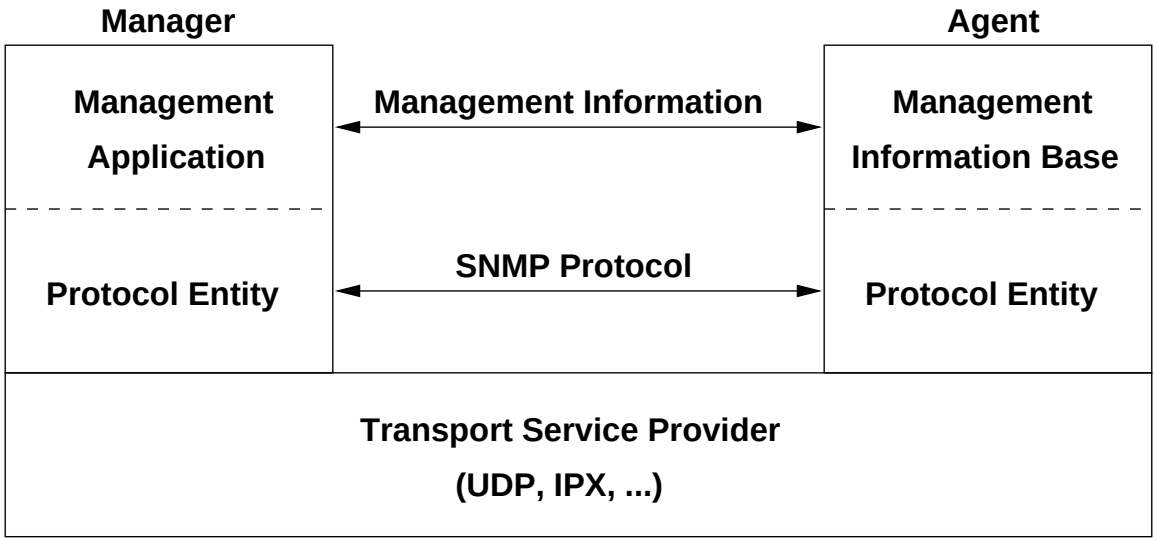
Overview and Evolution of SNMP

- "SNMP" (SNMPv1) circa 1988-1991
  (RFC 1155, 1157, 1212, 1213, 1215)
- "SNMP Security" and "SMP" circa 1991-1992
- "Party-based SNMPv2" (SNMPv2p) circa 1993-1995
  (RFC 1441–1452)
- "Community-based SNMPv2" (SNMPv2c) circa 1996
  (RFC 1901–1907)
- "SNMPv2u" and "SNMPv2*" circa 1996
- "SNMPv3" circa 1998-2002
  (RFC 3411-3418, 2578–2580)

SNMP Design Goals

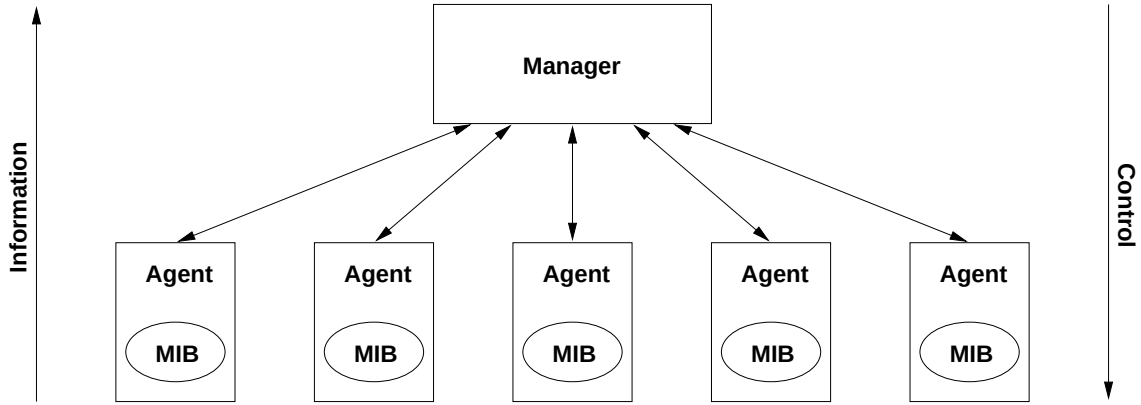
- Minimize the number and complexity of management functions realized by the management agent.
  - Reduce development costs for management agents.
  - Allows to instrument all kind of devices (HUBs as well as CRAYs).
  - Allows to develop management functions over time without the need to continuously change management agents.
- Extensibility through new MIB definitions.
- Independence of existing host or gateway architectures.
- Robustness (assumes only connection-less transport).
- Independence of other network services (e.g., directories).

Manager-Agent Relationship



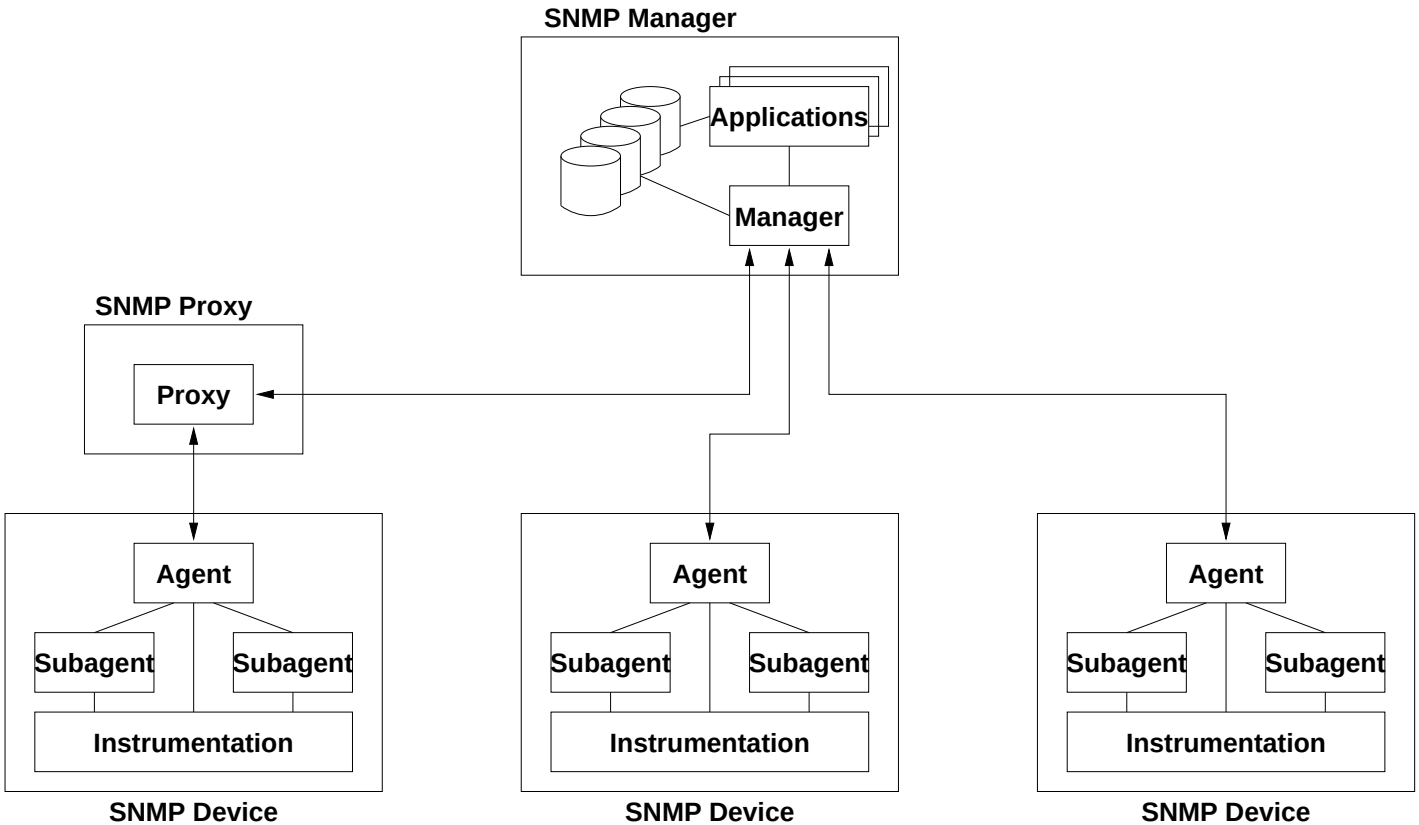
- The SNMP protocol provides primitives to read/write management information maintained by the agent.
- Management applications within the manager process the management information.
- The SNMP protocol usually runs over connection-less transport services.

Manager-Agent Relationship (cont.)



- An SNMP manager periodically polls agents for basic management information.
- SNMP agents can send unsolicited notifications to inform managers of exceptional conditions.
- An SNMP manager can adapt its polling strategy when receiving notifications (trap-directed polling).
- Information usually flows from the agent to the manager while control flows from the manager to the agent.

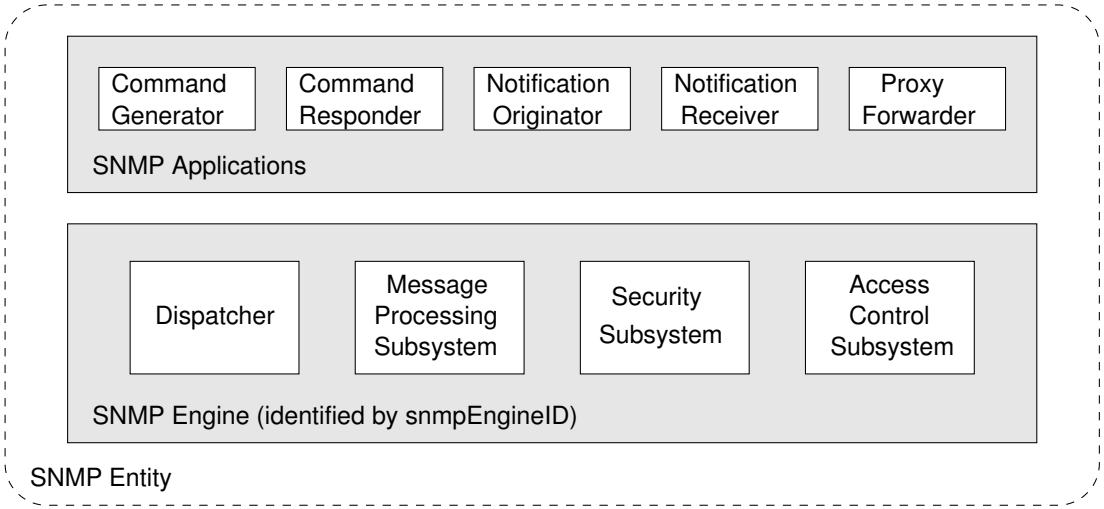
SNMP Deployment Model



SNMP Version 3

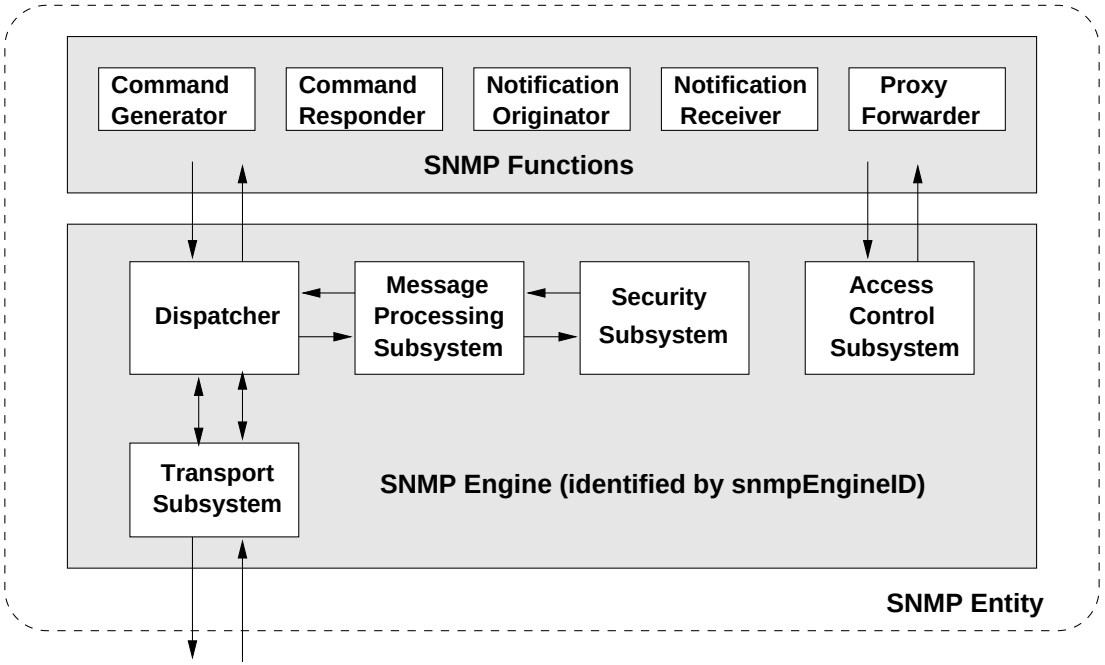
- → Architectural Concepts
- → Protocol Operations
- → Message Format
- → Authentication and Privacy
- → Authorization and Access Control
- → Remote Configuration
- → Status and Limitations

Architectural Model (RFC 3411)



- Fine grained SNMP applications instead of coarse grained agents and managers.
- Exactly on engine per SNMP entity and exactly one dispatcher per SNMP engine.
- Every abstract subsystem may have of one or more concrete models.
- Modularization enables incremental enhancements.

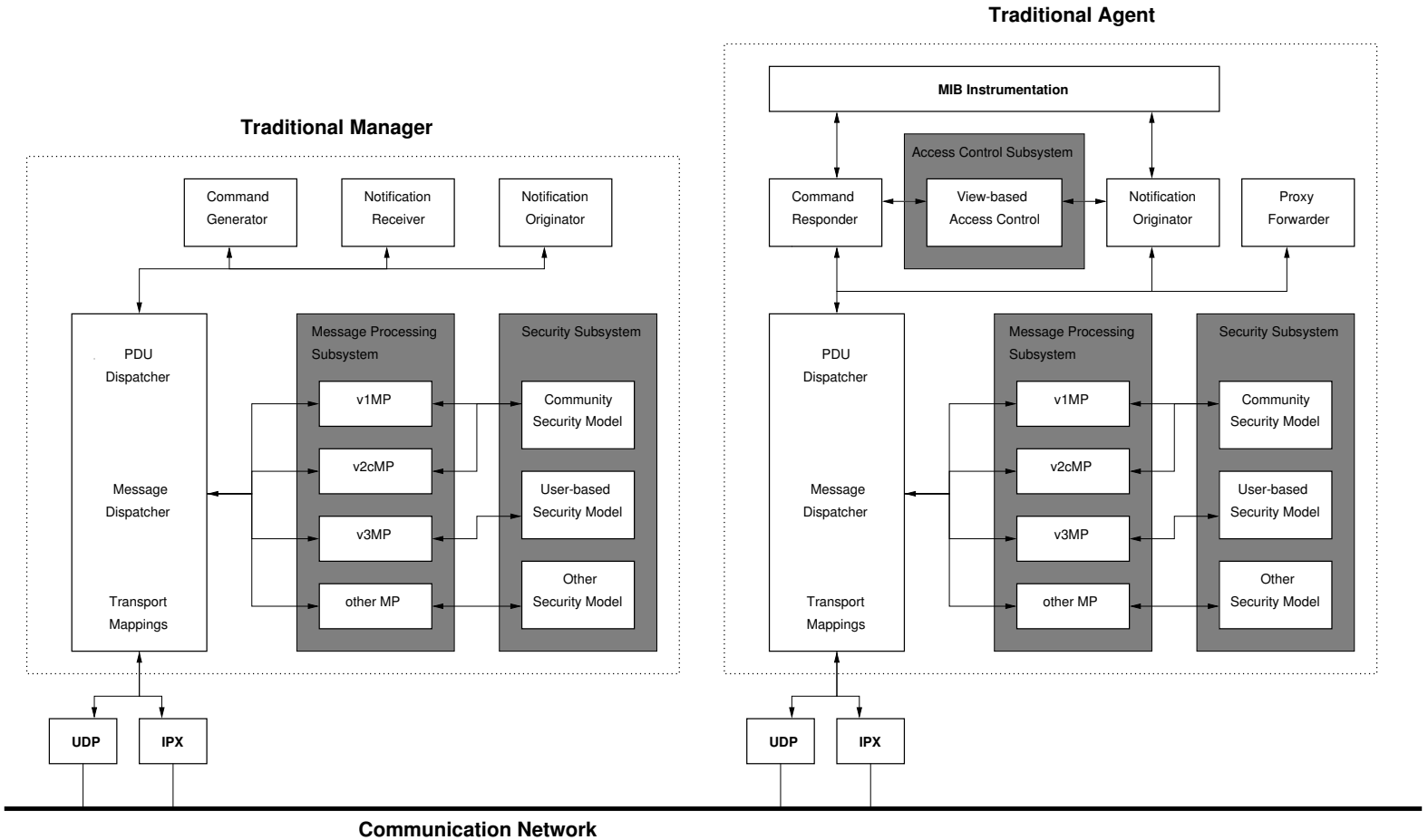
Extended Architectural Model (RFC 5590)



SNMP Contexts

- An context is a collection of management information accessible by an SNMP entity.
  - SNMP entities may have access to multiple contexts.
  - Identical management information may exist in more than one context.
- Within a management domain, a managed object is uniquely identified by:
  1. the identification of the engine within the SNMP entity (e.g., “800007e580e16e1566696f7440”)
  2. the context name within the SNMP entity (e.g., “board1”)
  3. the managed object type (e.g., “IF-MIB.ifDescr”)
  4. the instance identifier (e.g., “1”)

Manager and Agent in the SNMP Architecture



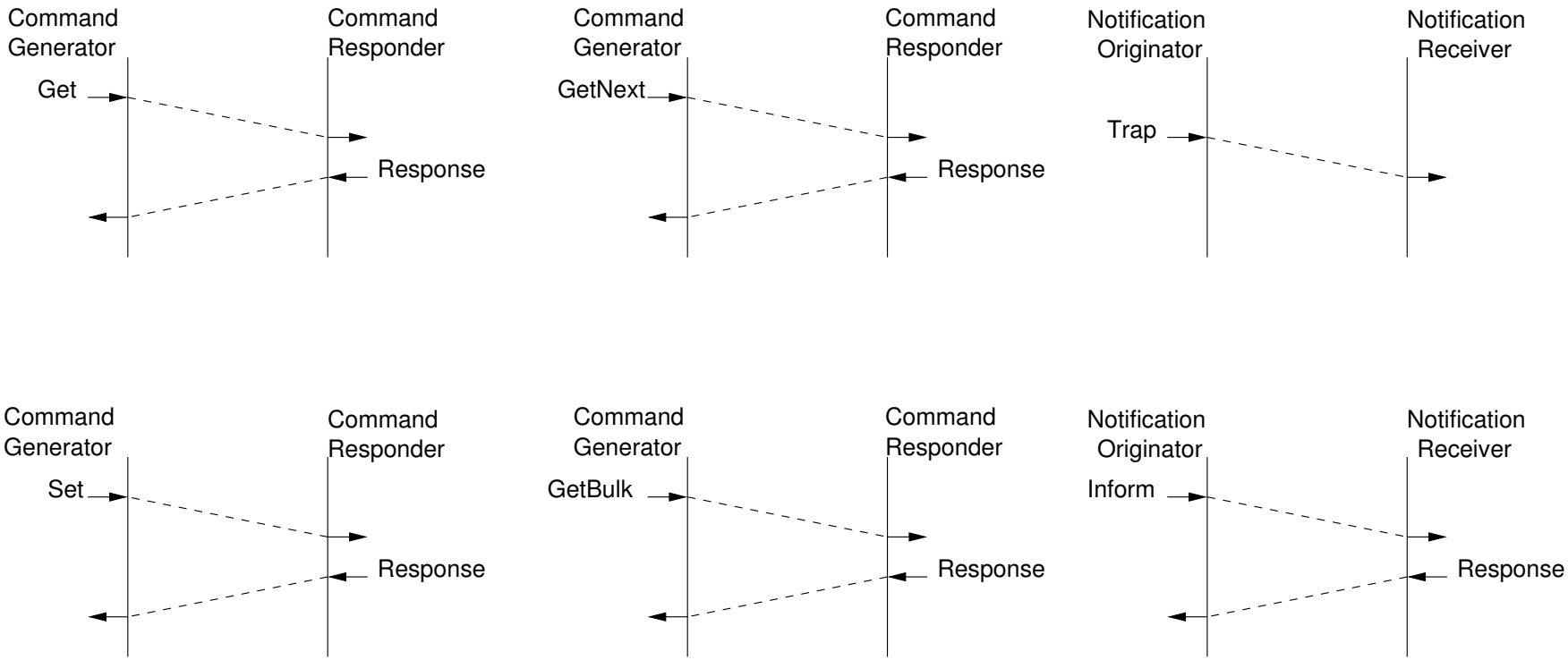
SNMPv3/USM Textual Conventions

- `SnmpEngineID`
  - Unique identification of an SNMP engine within a management domain.
- `SnmpSecurityModel`
  - Identification of a specific security model.
- `SnmpMessageProcessingModel`
  - Identification of a specific message processing model.
  - The message processing model is encoded in the `msgVersion`.

SNMPv3/USM Textual Conventions

- `SnmpSecurityLevel`
  - The security level of a given message (`noAuthNoPriv`, `authNoPriv`, `authPriv`).
  - The security level is encoded in the `msgFlags`.
- `KeyChange`
  - Defines a cryptographic algorithm to change authentication or encryption keys.
  - Does not require encryption.
  - An attacker can "drill forward" once a key is broken.

Protocol Operations (RFC 3416)



- An additional `Report` protocol operation is used internally for error notifications, engine discovery and clock synchronization.

Lexicographic Ordering

- Given are two vectors of natural numbers $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_m)$ with $n \leq m$. We say that x is lexicographically less than y if and only if one of the following conditions is true:

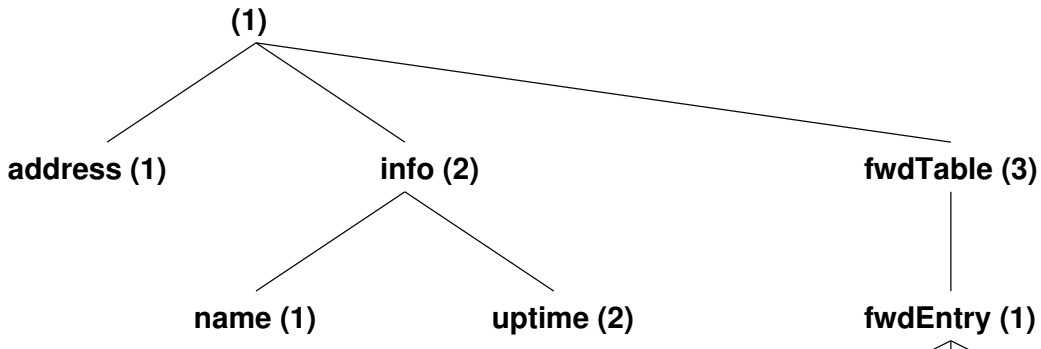
  (a) $x_j = y_j$ for $1 \leq j \leq k$ and $x_k < y_k$ with $k \leq n$ and $k \leq m$

  (b) $x_j = y_j$ for $1 \leq j \leq n$ and $n < m$

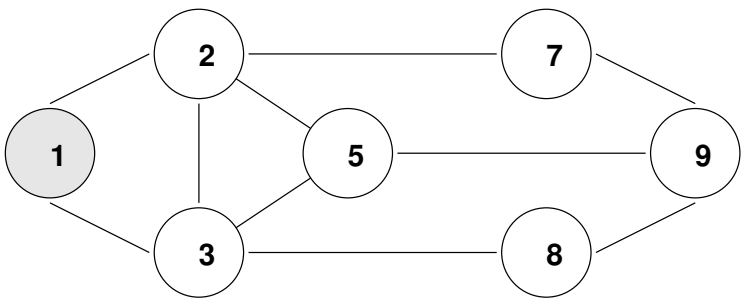
- All OIDs identifying instances can be lexicographically ordered.

- The SNMP protocol operates only on the lexicographically ordered list of MIB instances and not on the OID registration tree or on conceptual tables.

Simple Forwarding Table Example



fwdIndex (1)	fwdDest (2)	fwdNext (3)
1	2	2
2	3	3
3	5	2
4	7	2
5	8	3
6	9	3



Lexicographic Ordering Example

- Lexicographic ordered list of MIB instances:

OID	name	value
1.1.0	address.0	10.1.2.1
1.2.1.0	name.0	"ACME Router"
1.2.2.0	uptime.0	54321
1.3.1.1.1	fwdIndex.1	1
1.3.1.1.2	fwdIndex.2	2
1.3.1.1.3	fwdIndex.3	3
1.3.1.1.4	fwdIndex.4	4
1.3.1.1.5	fwdIndex.5	5
1.3.1.1.6	fwdIndex.6	6
1.3.1.2.1	fwdDest.1	2
1.3.1.2.2	fwdDest.2	3
1.3.1.2.3	fwdDest.3	5
1.3.1.2.4	fwdDest.4	7
1.3.1.2.5	fwdDest.5	8
1.3.1.2.6	fwdDest.6	9
1.3.1.3.1	fwdNext.1	2
1.3.1.3.2	fwdNext.2	3
1.3.1.3.3	fwdNext.3	2
1.3.1.3.4	fwdNext.4	2
1.3.1.3.5	fwdNext.5	3
1.3.1.3.6	fwdNext.6	3

- Conceptual table instances are ordered column by column not row by row.

PDU Processing Errors

- An error response signals the complete failure of the corresponding request.
- An error response contains an *error status* (numeric error code) and an *error index* (position in the variable list where the error occured).
- Error responses contain no useful management information.
- There is only a single error status and error index even if there are multiple errors.
- An error in general implies that none of the actions has taken place during a write operation (as if simultaneous writes).

PDU Error Codes (RFC 3416)

SNMPv3 Error Code	Get/GetNext/GetBulk	Set	Trap/Inform	SNMPv1 Error Code
noError(0)	X	X	X	noError(0)
tooBig(1)	X	X	X	tooBig(1)
noSuchName(2)				noSuchName(2)
badValue(3)				badValue(3)
readOnly(4)				readOnly(4)
genErr(5)	X	X	X	genErr(5)
noAccess(6)		X		noSuchName(2)
wrongType(7)		X		badValue(3)
wrongLength(8)		X		badValue(3)
wrongEncoding(9)		X		badValue(3)
wrongValue(10)		X		badValue(3)
noCreation(11)		X		noSuchName(2)
inconsistentValue(12)		X		badValue(3)
resourceUnavailable(13)		X		genErr(5)
commitFailed(14)		X		genErr(5)
undoFailed(15)		X		genErr(5)
authorizationError(16)	X	X	X	noSuchName(2)
notWritable(17)		X		noSuchName(2)
inconsistentName(18)		X		noSuchName(2)

PDU Processing Exceptions

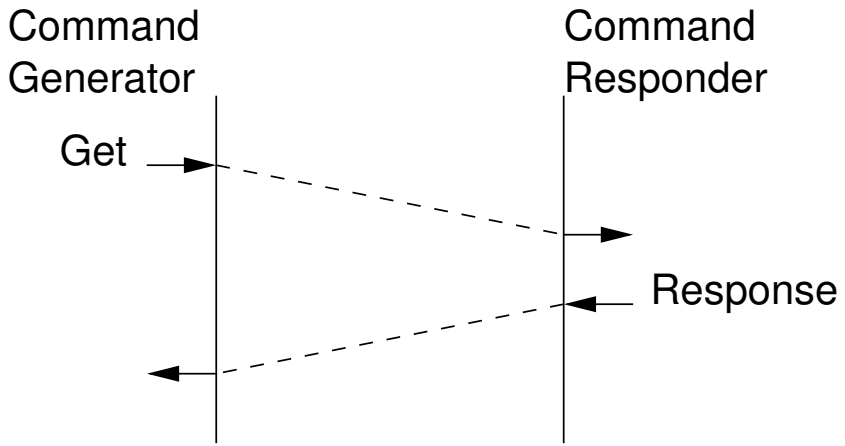
- A response can contain per variable binding exceptions.
- One or more exceptions in a response are not considered to be an error condition of the corresponding request.
- A response with exceptions still contains useful management information.
- Applications receiving response messages
  - must check the error code,
  - must detect exceptions, and
  - they must deal with them gracefully.
- Not all applications get this right...

PDU Exceptions (RFC 3416)

SNMPv3 Exception	Get	GetNext/GetBulk	SNMPv1 Error Status
noSuchObject	X		noSuchName(2)
noSuchInstance	X		noSuchName(2)
endOfMibView		X	noSuchName(2)

- The `noSuchInstance` exceptions indicates that a particular instances does not exist, but that other instances of the object type can exist.
- The `noSuchObject` exception indicates that a certain object type is not available.
- This distinctions allows smart applications to adapt to the capabilities of a particular command responder implementation.

Get Operation (RFC 3416)



- The `Get` operation is used to read one or more MIB variables.
- Possible error codes: `tooBig`, `genErr`
- Possible exceptions: `noSuchObject`, `noSuchInstance`

Example Get Operations

1. `Get(1.1.0)`
   `Response(noError@0, 1.1.0=10.1.2.1)`

2. `Get(1.2.0)`
   `Response(noError@0, 1.2.0=noSuchObject)`

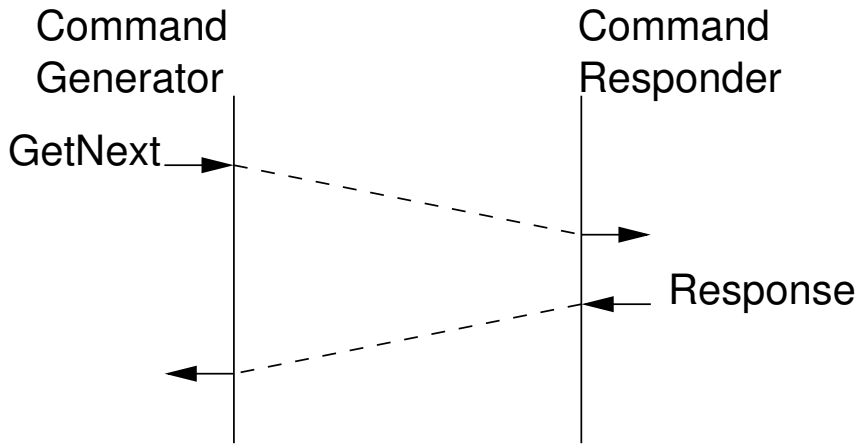
3. `Get(1.1.1)`
   `Response(noError@0, 1.1.1=noSuchInstance)`

4. `Get(1.1.0, 1.2.2.0)`
   `Response(noError@0, 1.1.0=10.1.2.1, 1.2.2.0=54321)`

5. `Get(1.3.1.1.4, 1.3.1.3.4)`
   `Response(noError@0, 1.3.1.1.4=4, 1.3.1.3.4=2)`

6. `Get(1.1.0, 1.1.1)`
   `Response(noError@0, 1.1.0=10.1.2.1, 1.1.1=noSuchInstance)`

GetNext Operation (RFC 3416)



- The `GetNext` operation allows to retrieve the value of the next existing MIB instances in lexicographic order.
- Successive `GetNext` operations can be used to walk the MIB instances without prior knowledge about the MIB structure.
- Possible error codes: `tooBig`, `genErr`
- Possible exceptions: `endOfMibView`

Example GetNext Operations

1. `GetNext(1.1.0)`
   `Response(noError@0, 1.2.1.0="ACME Router")`

2. `GetNext(1.2.1.0)`
   `Response(noError@0, 1.2.2.0=54321)`

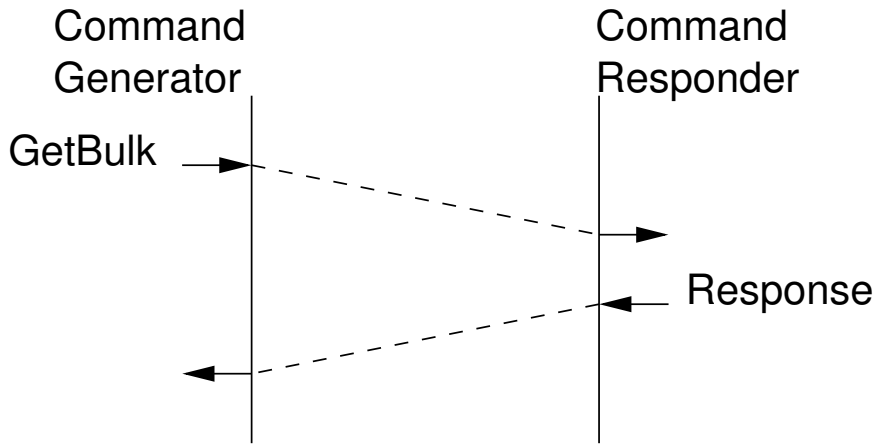
3. `GetNext(1.1)`
   `Response(noError@0, 1.1.0=10.1.2.1)`

4. `GetNext(1.3.1.1.1)`
   `Response(noError@0, 1.3.1.1.2=2)`

5. `GetNext(1.3.1.1.6)`
   `Response(noError@0, 1.3.1.2.1=2)`

6. `GetNext(1.3.1.1.1, 1.3.1.2.1, 1.3.1.3.1)`
   `Response(noError@0, 1.3.1.1.2=2, 1.3.1.2.2=3, 1.3.1.3.2=3)`

GetBulk Operation (RFC 3416)



- The `GetBulk` operation is a generalization of the `GetNext` operation where the agent performs a series of `GetNext` operations internally.
- The `GetBulk` operation like all the other protocol operations operates only on the lexicographically ordered list of MIB instances and does therefore not respect conceptual table boundaries.

GetBulk Operation (RFC 3416)

- GetBulk processing details:
  - The first N elements (`non-repeaters`) of the varbind list will be processed similar to the `GetNext` operation.
  - The remaining R elements of the varbind list are repeatedly processed similar to the `GetNext` operation.
  - The parameter M (`max-repetitions`) defines the upper bound of repetitions.
- The manager usually does not know how to choose a value for `max-repetitions`.
- If `max-repetitions` is too small, the potential gain will be small. If it is too large, there might be a costly overshoot.

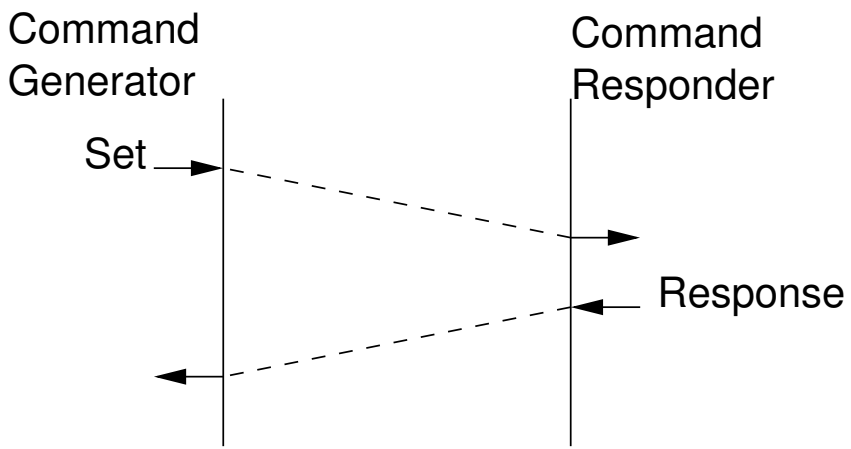
Example GetBulk Operations

1. `GetBulk(non-repeaters=0, max-repetitions=4, 1.1)`
   `Response(noError@0, 1.1.0=10.1.2.1, 1.2.1.0="ACME Router",`
   `1.2.2.0=54321, 1.3.1.1.1=1)`

2. `GetBulk(non-repeaters=1, max-repetitions=2, 1.2.2,`
   `1.3.1.1, 1.3.1.2, 1.3.1.3)`
   `Response(noError@0, 1.2.2.0=54321,`
   `1.3.1.1.1=1, 1.3.1.2.1=2, 1.3.1.3.1=2,`
   `1.3.1.1.2=2, 1.3.1.2.2=3, 1.3.1.3.2=3)`

- The `non-repeaters` are typically used to retrieve a discontinuity indicating scalars, such as `sysUpTime.0`.
- Any ideas for a better GetBulk operation?

Set Operation (RFC 3416)



- The `Set` operation allows to modify a set of MIB instances. The operation is atomic (either all instances are modified or none).
- Possible error codes: `wrongValue`, `wrongEncoding`, `wrongType`, `wrongLength`, `inconsistentValue`, `noAccess`, `notWritable`, `noCreation`, `inconsistentName`, `resourceUnavailable`, `commitFailed`, `undoFailed`

Example Set Operations

1. `Set(1.2.1.0="Moo Router")`
   `Response(noError@0, 1.2.1.0="Moo Router")`

2. `Set(1.1.0="foo.bar.com")`
   `Response(badValue@1, 1.1.0="foo.bar.com")`

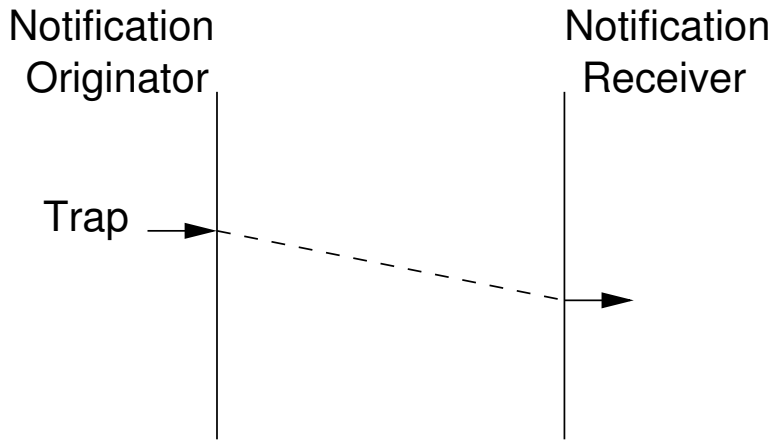
3. `Set(1.1.1=10.2.3.4)`
   `Response(noSuchName@1, 1.1.1=10.2.3.4)`

4. `Set(1.2.1.0="Moo Router", 1.1.0="foo.bar.com")`
   `Response(badValue@2, 1.2.1.0="Moo Router", 1.1.0="foo.bar.co`

5. `Set(1.3.1.1.7=7, 1.3.1.2.7=2, 1.3.1.3.7=3)`
   `Response(noError@0, 1.3.1.1.7=7, 1.3.1.2.7=2, 1.3.1.3.7=3)`

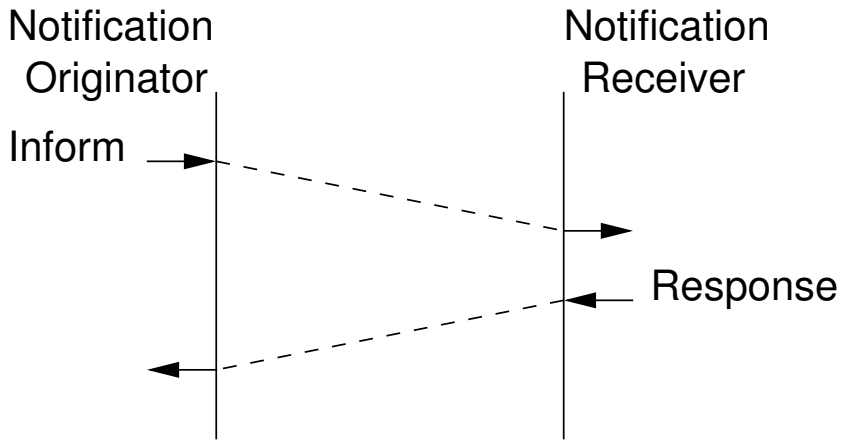
- The error codes `authorizationError` and `readOnly` are not used.
- No support for object type specific error codes.

Trap Operation (RFC 3416)



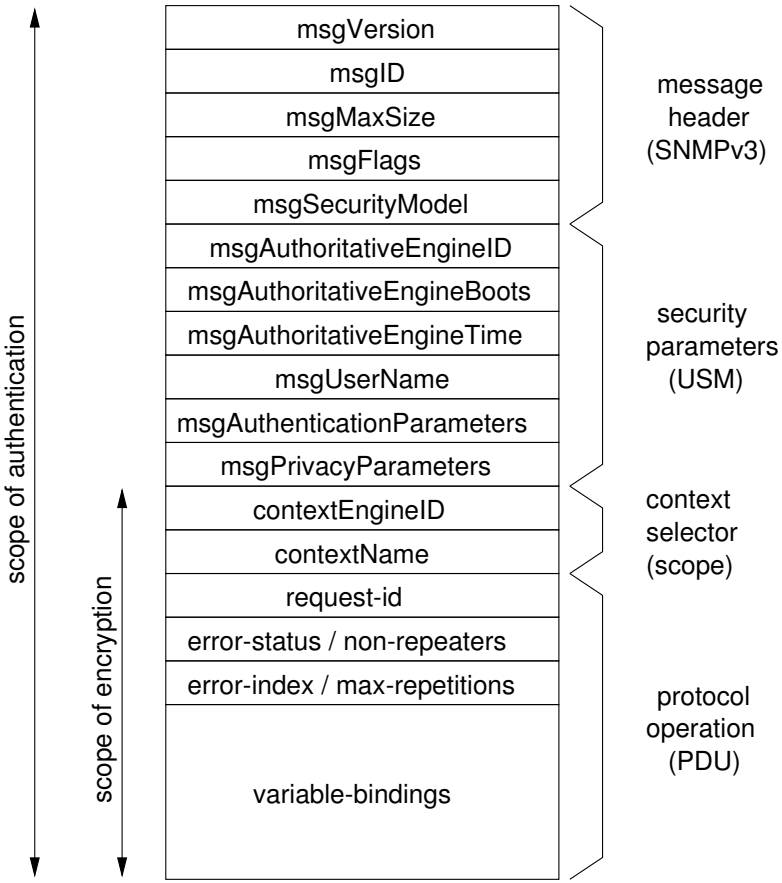
- The `Trap` operation is used to notify a manager of the occurance of an event.
- The `Trap` operation is unconfirmed: The sending agent does not know whether the trap was received and processed by a manager.
- All trap specific information in encoded in the varbind list (`sysUpTime`, `snmpTrapOID`, `snmpTrapEnterprise`).

Inform Operation (RFC 3416)



- The `Inform` operation is a confirmed trap.
- The contents of the varbind list of an `Inform` operation is similar to that of a `Trap` operation.
- The reception of an `Inform` operation is confirmed by a response message from the notification receiver.
- Confirmation indicates that the notification was delivered, not that the notification was understood.

Message Format (RFC 3412, RFC 3414)



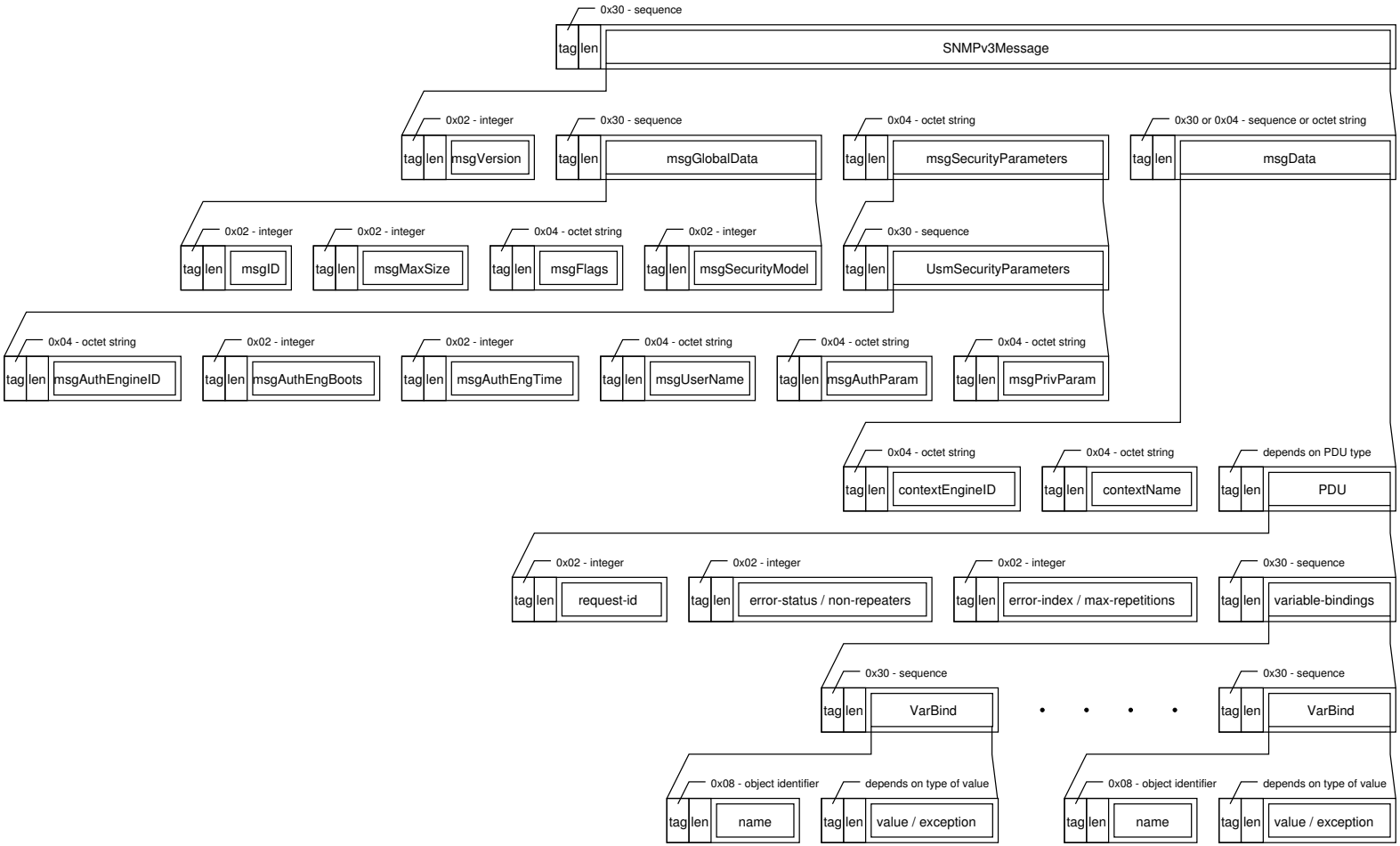
- `msgVersion` identifies the message processing model
- `msgSecurityModel` identifies the security model
- `contextEngineID` and `contextName` determine the context
- protocol operation type (and version) is determined by the tag of the PDU

Classes of Protocol Operations (RFC 3411)

Class	Description
Read	PDUs that retrieve management information.
Write	PDUs which attempt to modify management information.
Response	PDUs which are sent in response to a request.
Notification	PDUs which transmit event notifications.
Internal	PDUs exchanged internally between SNMP engines.
Confirmed	PDUs which cause the receiver to send a response.
Unconfirmed	PDUs which are not acknowledged.

- The processing of a message depends on the class of the embedded protocol operation.
- PDU classes support the introduction of new protocol operations without changes the core specifications.
- However, no indication of the set of protocol operations supported by an SNMP engine implementation.

Encoding of SNMPv3/USM Messages



Security Issues

- The following questions must be answered in order to decide whether an operation should be performed or not:
  1. Is the message specifying an operation authentic?
  2. Who requested the operation to be performed?
  3. What objects are accessed in the operation?
  4. What are the rights of the requester with regard to the objects of the operation?
- 1 and 2 are answered by message security mechanisms (authentication and privacy).
- 3 and 4 are answered by authorization mechanisms (access control).

Security Threats (RFC 3414)

1. Modification of Information
   (Unauthorized modification of in-transit SNMP messages.)
2. Masquerade
   (Unauthorized users attempting to use the identity of authorized users.)
3. Disclosure
   (Protection against eavesdropping on the exchanges between SNMP entities.)
4. Message Stream Modification
   (Re-ordered, delayed or replayed messages to effect unauthorized operations.)

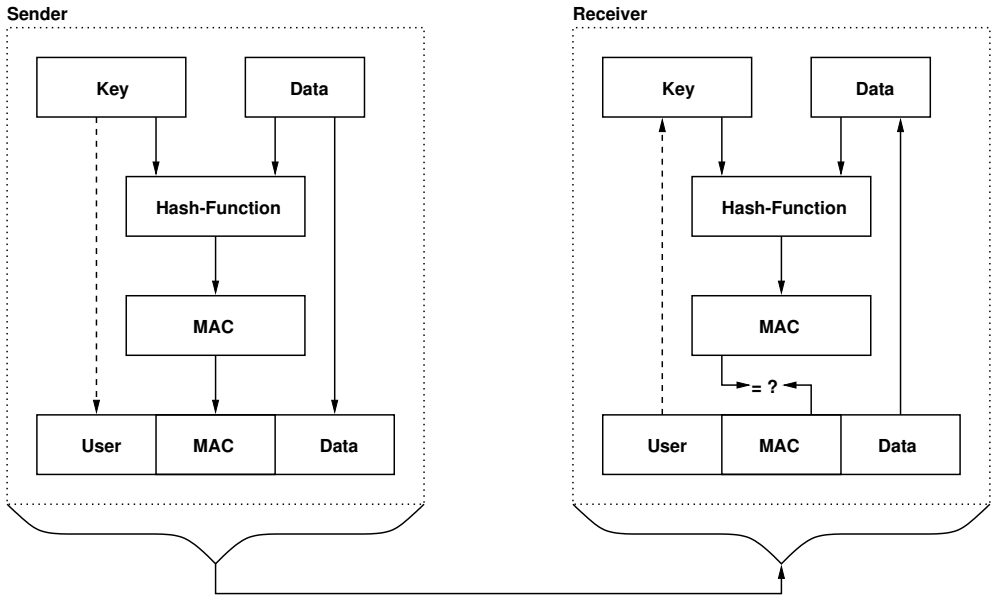
USM Security Services (RFC 3414)

- Data Integrity
  - Data has not been altered or destroyed in an unauthorized manner.
  - Data sequences have not been altered to an extent greater than can occur non-maliciously.
- Data Origin Authentication
  - The claimed identity of the user on whose behalf received data was originated is corroborated.
- Data Confidentiality
  - Information is not made available or disclosed to unauthorized individuals, entities, or processes.

USM Security Services (RFC 3414)

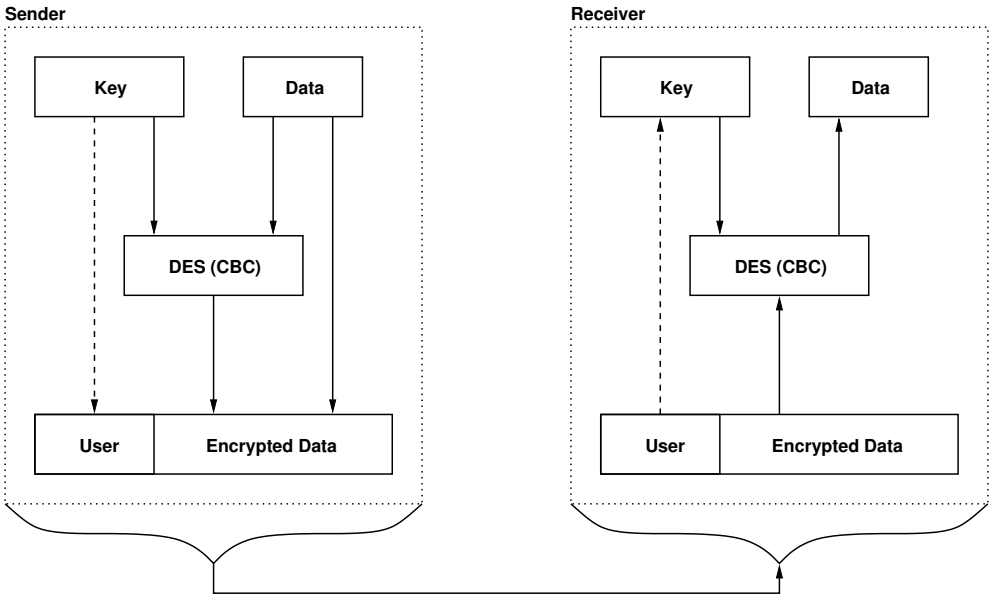
- Message Timeliness and Limited Replay Protection
  - A message whose generation time is outside of a time window is not accepted.
  - Message reordering is not dealt with and can occur in normal conditions too.
- No protection against Denial of Service attacks
  - Too hard of a problem to solve.
- No protection against Traffic Analysis attacks
  - Many management traffic patterns are predictable.
  - Hiding periodic management traffic would be extremly costly.

Data Integrity and Data Origin Authentication



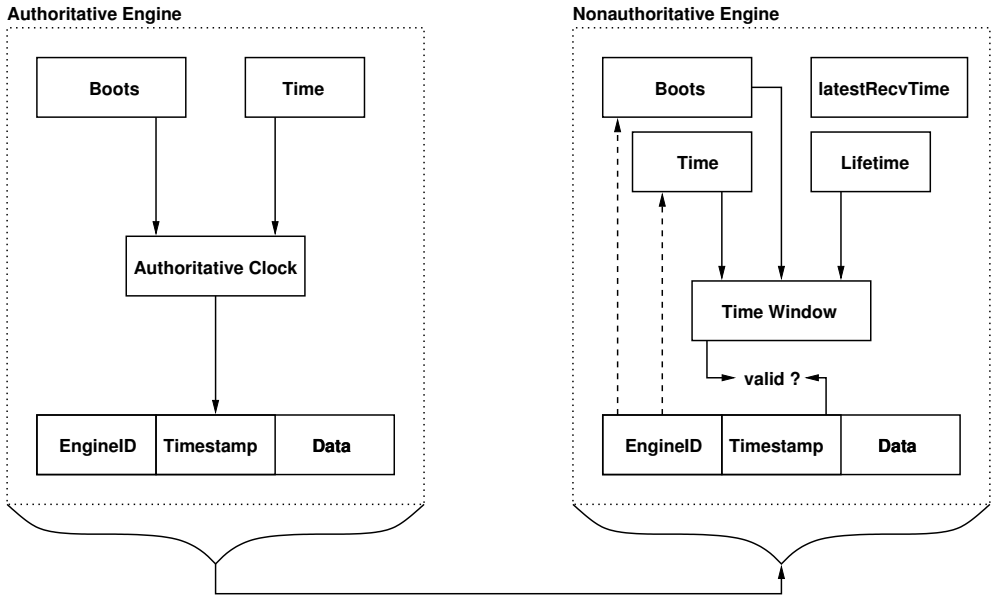
- Cryptographic strong oneway hash functions generate message authentication codes (MACs).
- The MAC ensures integrity, the symmetric key provides for authentication.
- USM currently uses HMAC-MD5-96 or HMAC-SHA-96.
- Other hash functions may be added in the future.

Data Confidentiality



- Optional encryption of the `ScopedPDU` using symmetric but localized keys.
- USM currently uses CBC-DES.
- Other encryption functions may be added in the future.
- Encryption is CPU expensive — use only when needed.

Message Timeliness and Replay Protection



- A non-authoritative engine maintains a notion of the time at the authoritative engine.
- A non-authoritative engine keeps track when the last authentic message was received from a given engine.
- A message is accepted and considered “fresh” if it falls within a time window.

Generating Keys from Passwords

- Algorithmic transformation of a human readable password into a cryptographic key:
  - Produce a string S of length $2^{20} = 1048576$ bytes by repeating the password as many times as necessary.
  - Compute the users key K_U using either $K_U = MD5(S)$ or $K_U = SHA(S)$.
- Slows down naive brute force password attacks.
- No serious barrier for an attacker with a transformed dictionary.

Localized Keys

- Algorithmic transformation of the users key K_U and an engine identification E into a localized key:
  - For a given engine E, compute either
    $K_{UE} = MD5(K_U, E, K_U)$ or
    $K_{UE} = SHA(K_U, E, K_U)$.
- Advantage: A compromised key does not give access to other SNMP engines.
- Very important in environments where devices can easily be stolen or accessed physically by attackers.

Key Changes

- Key change procedure (initiator):
  1. Generate a random value r from a random number generator.
  2. Compute $d = MD5(K_{old}, r)$ or $d = SHA(K_{old}, r)$.
  3. Compute $\delta = d \oplus K_{new}$ and send (δ, r).
- Key change procedure (receiver):
  1. Compute $d = MD5(K_{old}, r)$ or $d = SHA(K_{old}, r)$.
  2. Compute $K_{new} = d \oplus \delta$.
- The receiver computes the correct new key since $d \oplus \delta = d \oplus (d \oplus K_{new}) = K_{new}$.

Key Change Properties

- Key changes must be possible without encryption since encryption is optional.
- An attacker who is able to catch all key updates can calculate the current keys once an old key has been broken.
- Attackers thus get an unlimited amount of time to break keys if they can catch all key change requests.

$\Rightarrow$ Use encryption for key changes if at all possible!

Authoritative Engine

- Either the sender or the receiver of a message is designated the authoritative engine.
- The receiver is authoritative if the message contains a confirmed class PDU.
- The sender is authoritative if the message contains an unconfirmed class PDU.
- The determination whether a message is recent is made relative to the authoritative engine.

Timeliness Check

Authoritative Receiver

1. $\texttt{snmpEngineBoots} = 2^{31} - 1$
2. $\texttt{msgAuthoritativeEngineBoots} \neq \texttt{snmpEngineBoots}$
3. $abs(\texttt{msgAuthoritativeEngineTime} - \texttt{snmpEngineTime}) > 150$ seconds

Non-authoritative Receiver

1. $\texttt{snmpEngineBoots} = 2^{31} - 1$
2. $\texttt{msgAuthoritativeEngineBoots} < \texttt{snmpEngineBoots}$
3. $\texttt{msgAuthoritativeEngineBoots} = \texttt{snmpEngineBoots}$ and $\texttt{msgAuthoritativeEngineTime} < \texttt{snmpEngineTime} - 150$

Clock Synchronization

- For each remote authoritative SNMP engine, an SNMP engine maintains:
  `snmpEngineBoots`, `snmpEngineTime` and `latestReceivedEngineTime`
- Time synchronization only occurs if the message is authentic.
- An update occurs, if at least one of the following conditions is true:
  1. `msgAuthoritativeEngineBoots` $>$ `snmpEngineBoots`
  2. `msgAuthoritativeEngineBoots` $=$ `snmpEngineBoots` and `msgAuthoritativeEngineTime` $>$ `latestReceivedEngineTime`

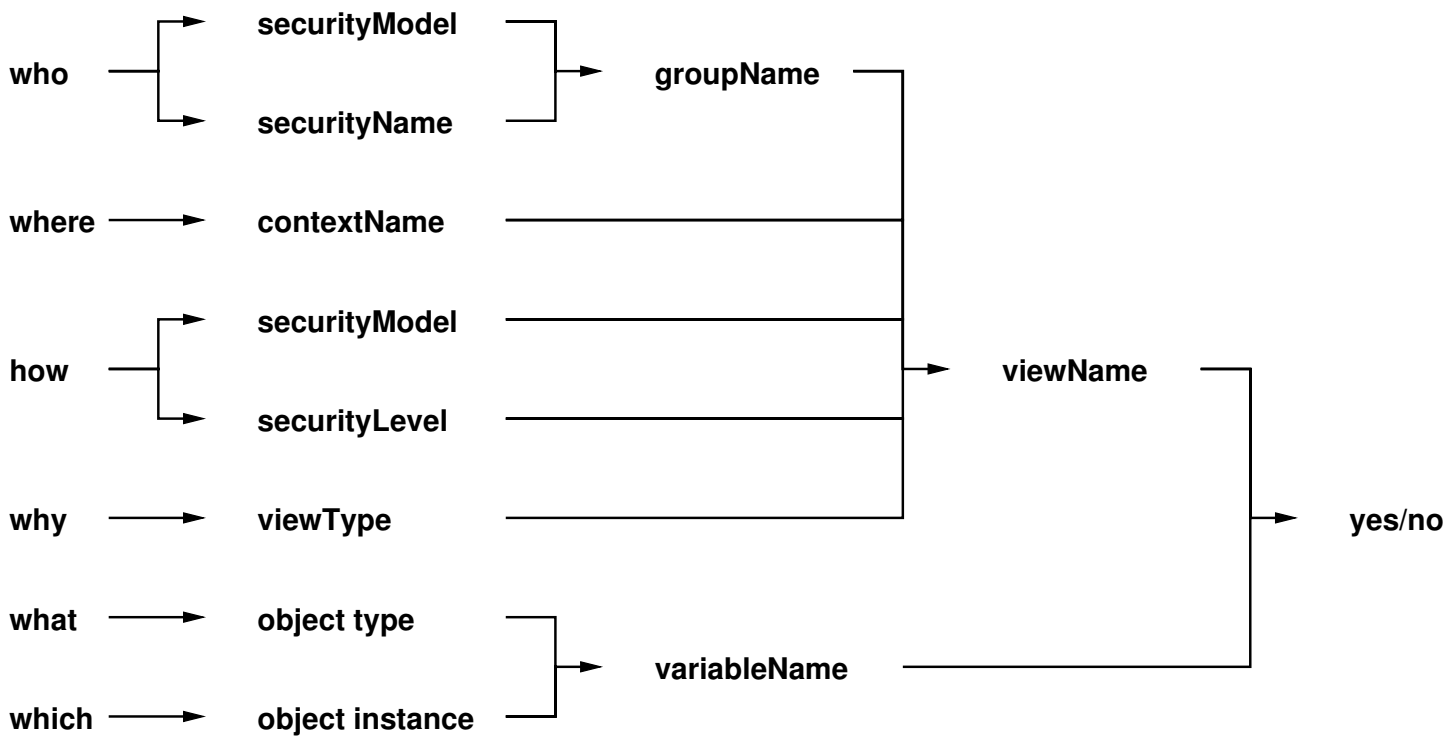
Discovery and Initial Synchronization

- The engine identification is needed to compute localized keys and to keep clock information for authoritative engines.
- An SNMP engine can learn the engine identification by sending a `noAuthNoPriv` request with a zero-length `msgAuthoritativeEngineID`.
- The receiver returns a `Report` PDU with the real `msgAuthoritativeEngineID`.
- Similarly, (initial) clock synchronization happens by sending an authentic request and receiving a `Report` PDU with the authoritative time.

USM MIB (RFC 3414)

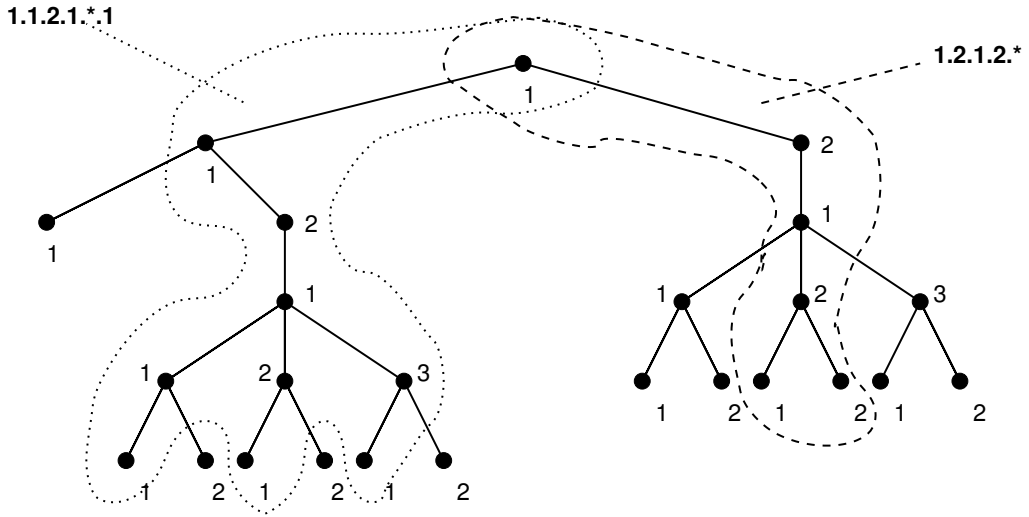
- The `usmUserTable` maps USM user names to `securityNames`.
- New entries may be created by cloning existing entries (together with their keys).
- The `usmUserAuthKeyChange` and `usmUserPrivKeyChange` objects may be used by the security administrator to change the user's keys.
- The `usmUserOwnAuthKeyChange` and `usmUserOwnPrivKeyChange` objects may be used by the user to change his keys.

Authorization and Access Control (RFC 3415)



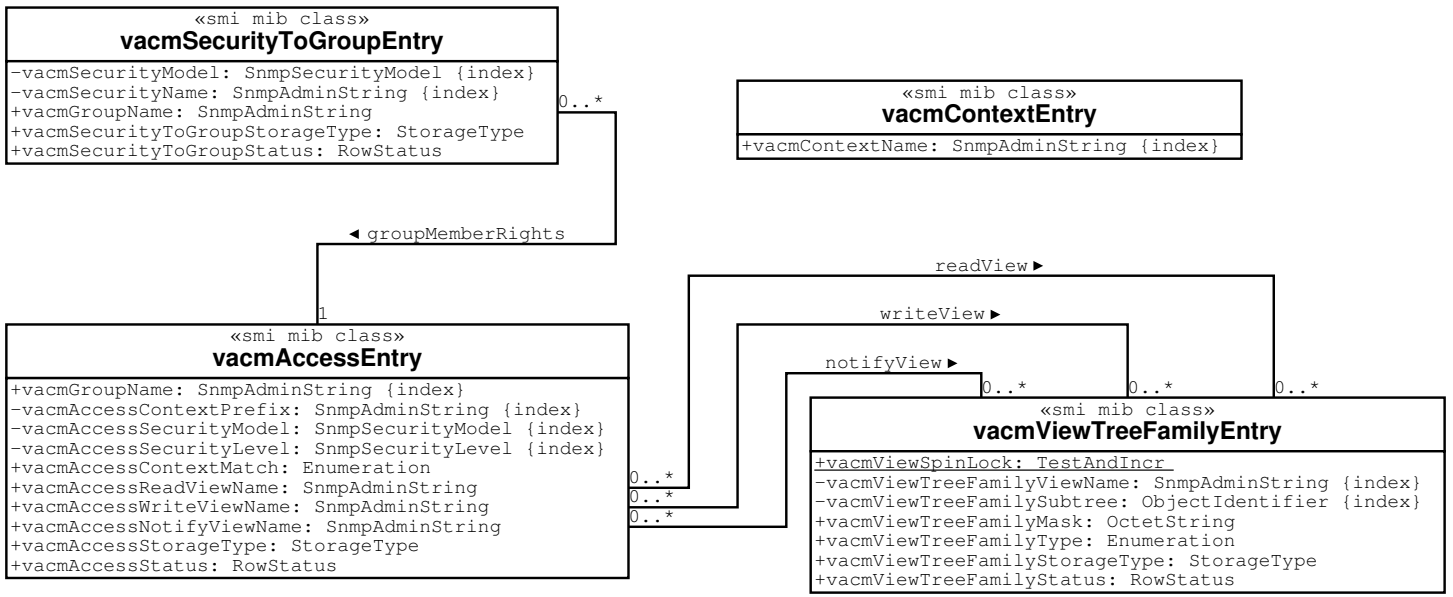
- Three different `securityLevels`: `noAuthNoPriv`, `authNoPriv`, `authPriv`
- A `securityName` is a security model independent name for a principal.

View-based Access Control (RFC 3415)



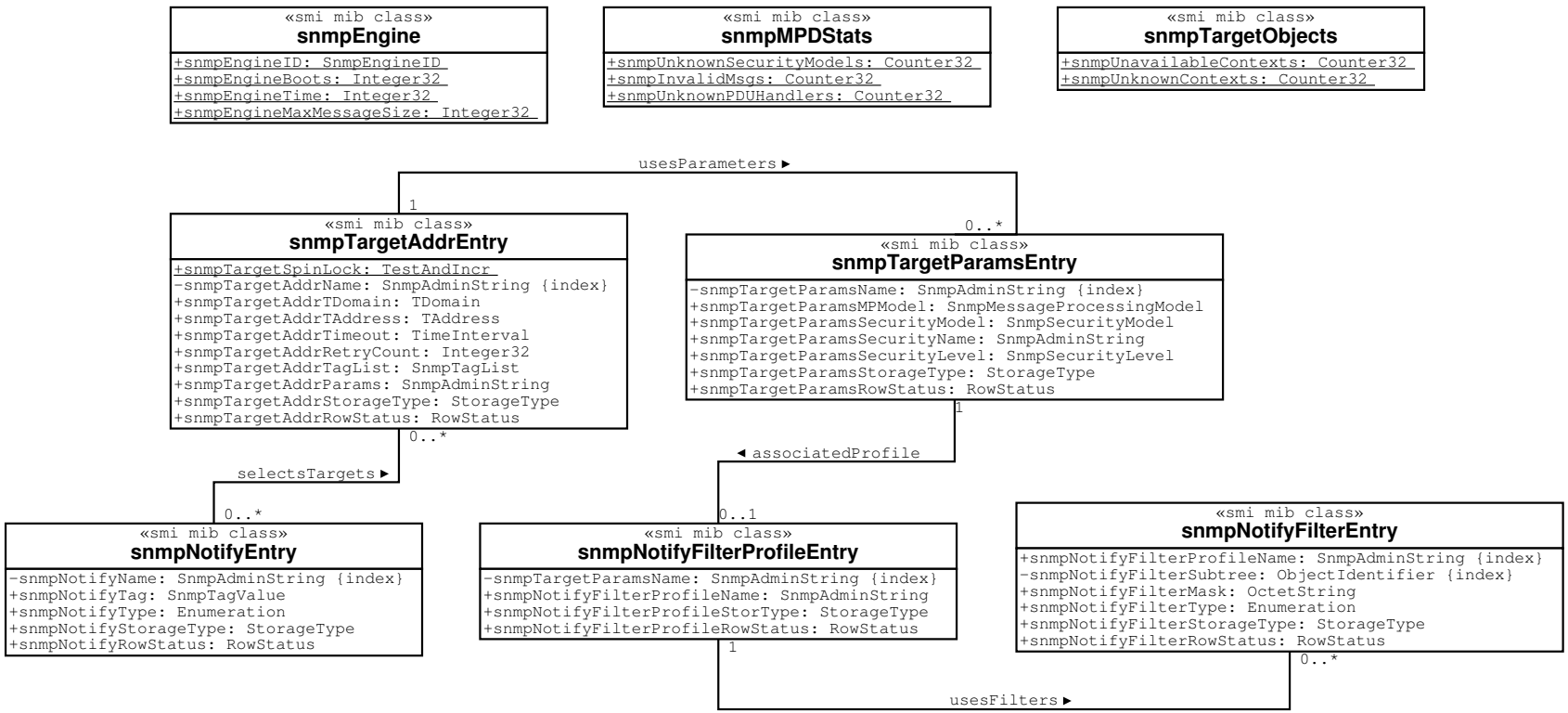
- A *view subtree* is a set of managed object instances with a common OID prefix.
- A *view tree family* is the combination of an OID prefix with a bit mask (wildcarding of OID prefix components).
- A *view* is an ordered set of view tree families.
- Access control rights are defined by a *read view*, *write view* or *notify view*.

View-based Access Control MIB (RFC 3415)



- A security name (with a given security level) can not be a member of multiple groups.
- The `vacmViewTreeFamilyType` can be used to include or exclude a view tree family.
- The context table is kind of degenerated.

Remote Configuration (RFC 3413)



- SNMPv3 defines several MIB modules for remote configuration of SNMP entities.

SNMPv3 Status and Limitations

- Many implementations and products are available.
- Some technology domains (e.g., cable modem industry in the US) require SNMPv3 support.
- However, general deployment happens much slower than originally expected.
- Manual configuration is an error prone and time consuming.
- Lack of integration in deployed AAA systems.
- Remote configuration and key management requires nontrivial applications.
- Work underway to run SNMP over secure transports (SSH / TLS / DTLS / ...)

SNMPv3 Status and Limitations

- Missing extensibility for new base data types (e.g., Unsigned64).
- Missing extensibility for new protocol operations (e.g., `GetRange`).
- Limited flexibility in VACM grouping rules.
- Asymmetries between notification filtering and VACM filtering.
- Strength of USM security (DES versus AES, key change procedure).
- Initial key assignment problematic (no standardized Diffie-Hellman exchange, no integration with other key management systems).

Network Configuration using NETCONF / YANG

41 Network Management Overview

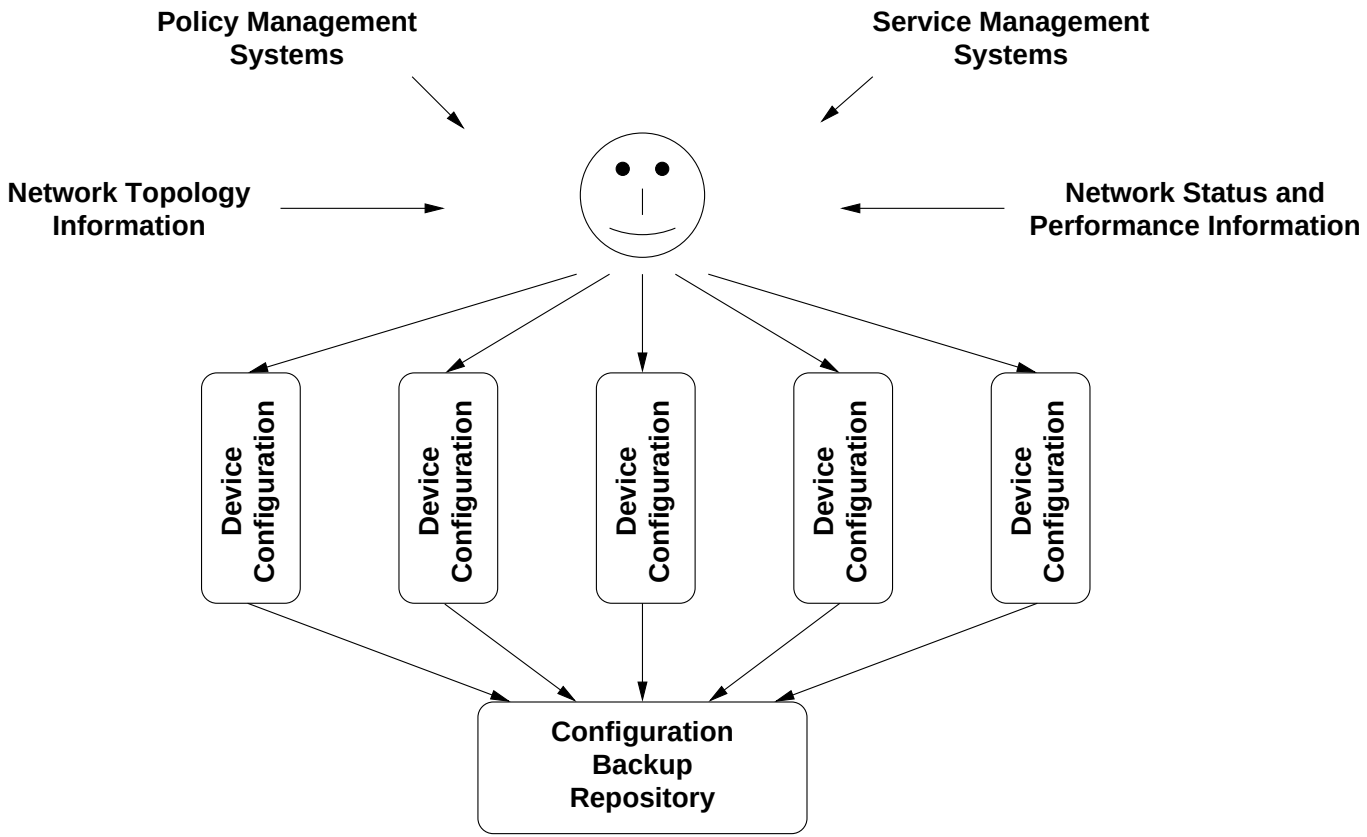
42 Network Monitoring using SNMP

43 Network Configuration using NETCONF / YANG
- Configuration Management Approaches
- NETCONF Protocol Overview
- YANG Data Modeling Overview

44 System Logging Protocol (SYSLOG)

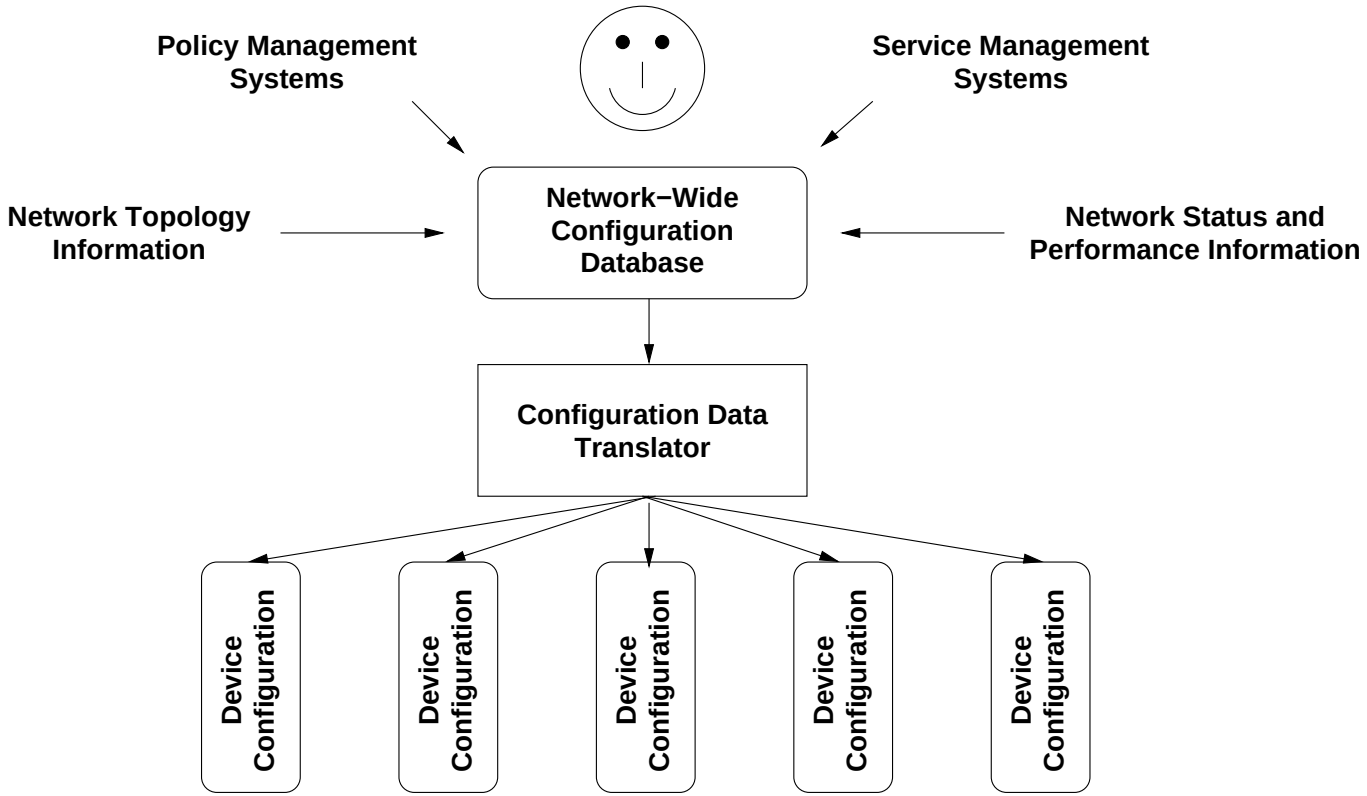
45 Traffic Analysis using NETFLOW / IPFIX

"The Network is the Record" Approach



- Labor intensive, expensive, error prone, widely deployed

"Generate Everything" Approach



- All configuration changes are made (and validated) on the network-wide configuration database and devices are never touched manually

Configuration Management Requirements (part 1)

R1: configuration state vs. operational state

A configuration management protocol must be able to distinguish between configuration state and operational state.

R2: concurrency support

A configuration management protocol must provide primitives to prevent errors due to concurrent configuration changes.

R3: configuration transactions

A configuration management protocol must provide primitives to apply configuration changes to a set of network elements in a robust and transaction-oriented way.

Configuration Management Requirements (part 2)

R4: distribution vs. activation

It is important to distinguish between the distribution of configurations and the activation of a certain configuration.

R5: distinguish multiple configurations

A configuration management protocol must be able to distinguish between several configurations and devices should be able to hold multiple configurations.

R6: persistence of configuration state

A configuration management protocol must be clear about the persistence of configuration changes.

Configuration Management Requirements (part 3)

R7: configuration change events

A configuration management protocol must be able to report configuration change events to help tracing back configuration changes.

R8: configuration dump and restore

A full configuration dump and a full configuration restore are primitive operations frequently used by operators and must be supported appropriately.

Configuration Management Requirements (part 4)

R9: support for standard tools

A configuration management protocol must represent configuration state and operational state in a form which allows operators to use existing comparison, conversion, and versioning tools.

R10: minimize impact of configuration changes

Configurations must be described such that devices can determine a set of operations to bring the devices from a given configuration state to the desired configuration state, minimizing the impact caused by the configuration change itself on networks and systems.

NETCONF IETF Working Group

Milestones

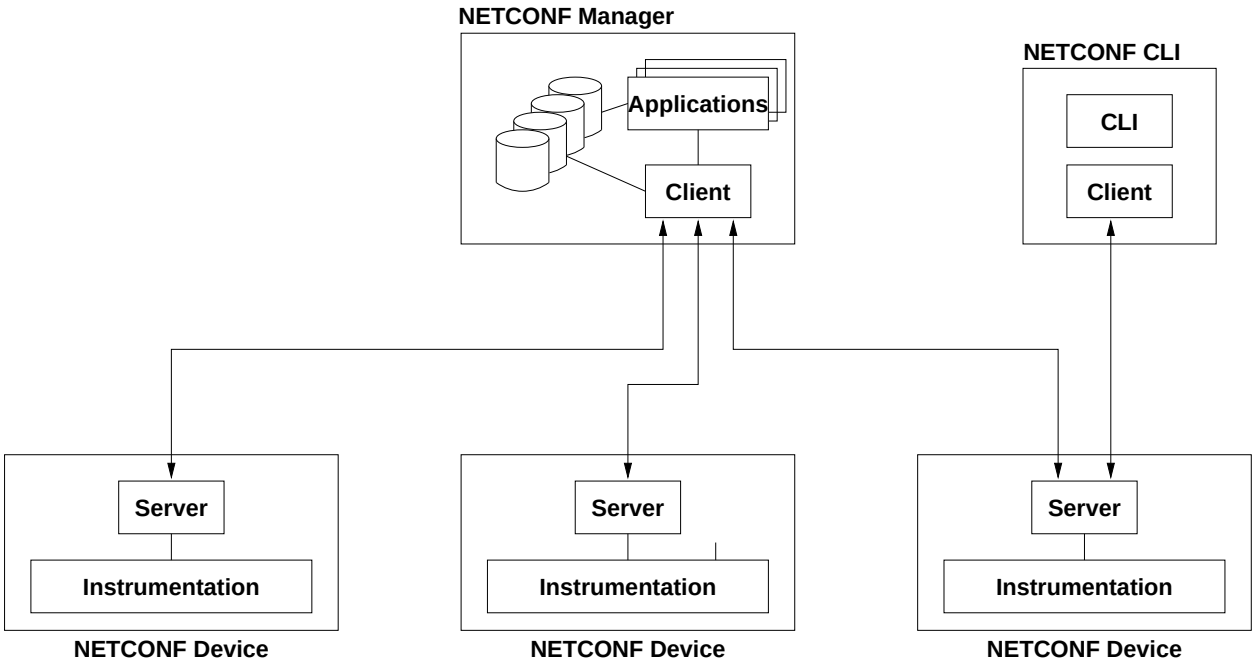
- NETCONF WG chartered in May 2003, core specifications published in December 2006
- Heavily influenced by Juniper's JunoScript
- Core contributors from Juniper Networks and Cisco
- Some design decisions were difficult to take

Status

The NETCONF WG is still active:

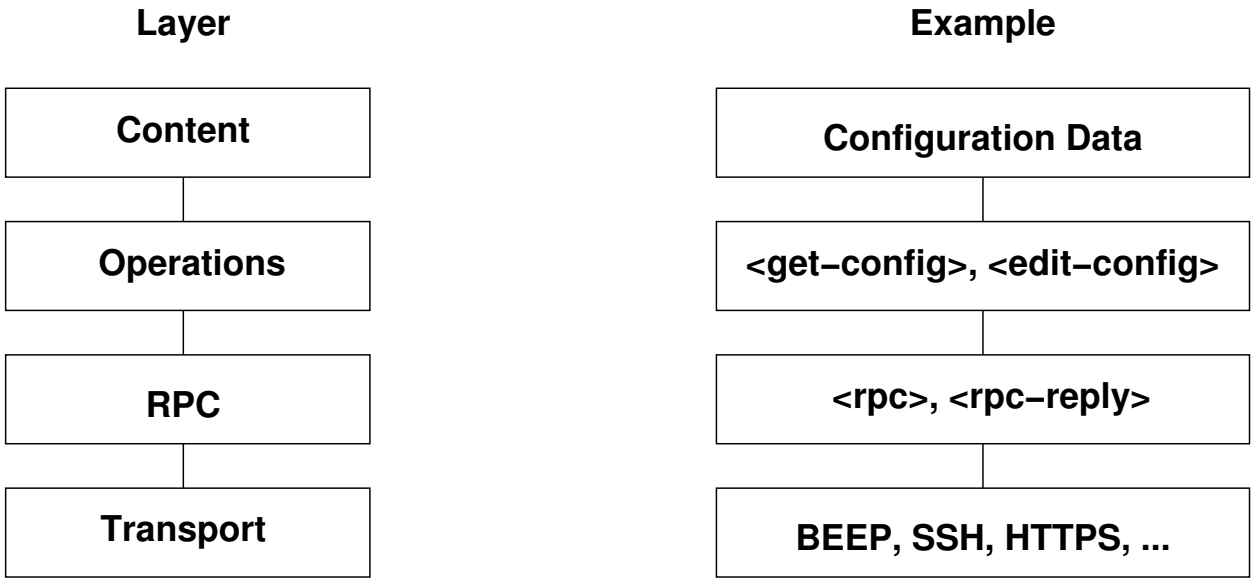
- fine grained locking
- with-defaults capability
- data model for NETCONF monitoring
- revision of the NETCONF core specification

Deployment Model



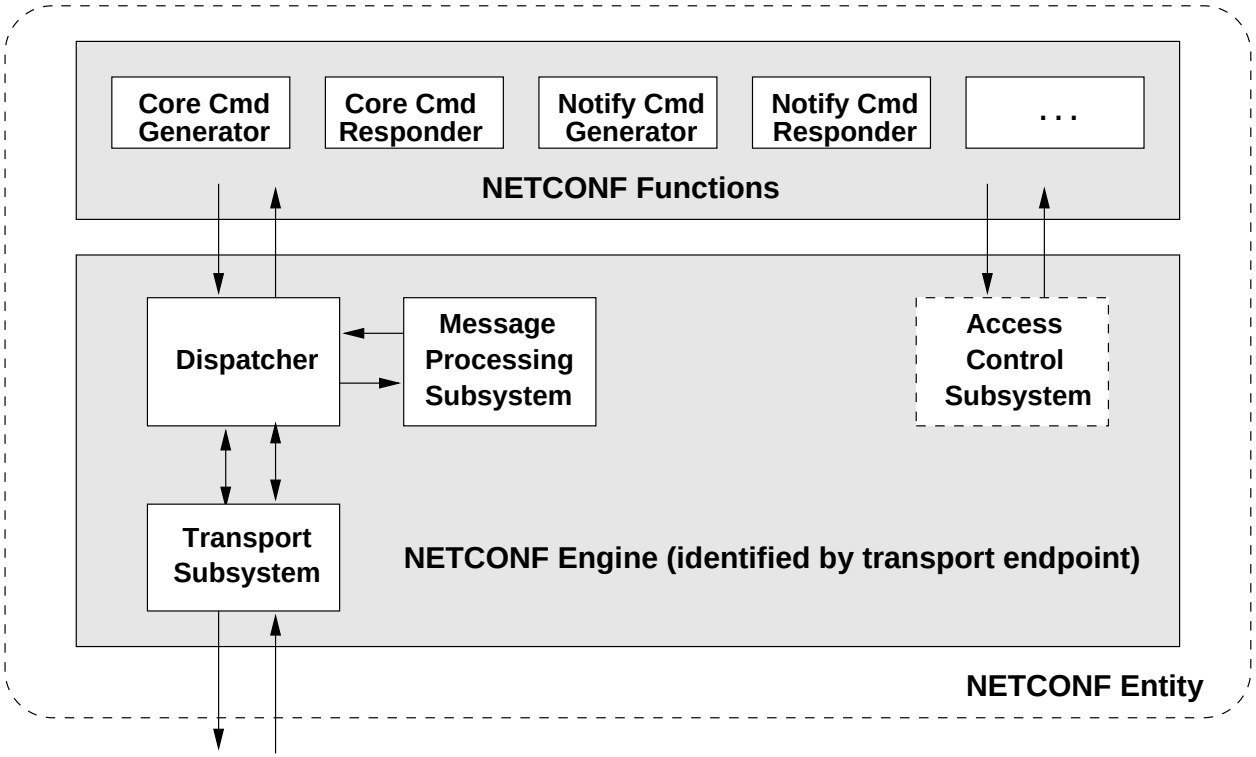
- NETCONF enabled devices include a NETCONF server
- Management applications include a NETCONF client
- Command Line Interfaces (CLIs) can be a wrapped around a NETCONF client

Layering Model (RFC4741)



- Security has to be provided by the transport layer.
- The operations layer provides the primitives to handle configurations.
- The set of operations is supposed to be extensible.

Architectural Model



- Does not exist formally (so take this with some care)
- Loosely based on SNMP architectural concepts

Configuration Datastores

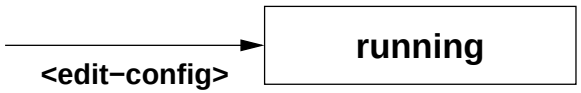
Definition

A configuration datastore is the complete set of configuration information that is required to get a device from its initial default state into a desired operational state.

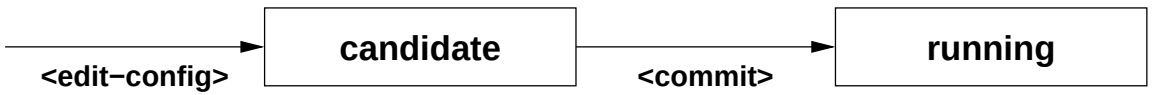
- The `<running>` configuration datastore represents the currently active configuration of a device and is always present.
- The `<startup>` configuration datastore represents the configuration that will be used during the next startup.
- The `<candidate>` configuration datastore represents a configuration that may become a `<running>` configuration through an explicit commit.

Transaction Models

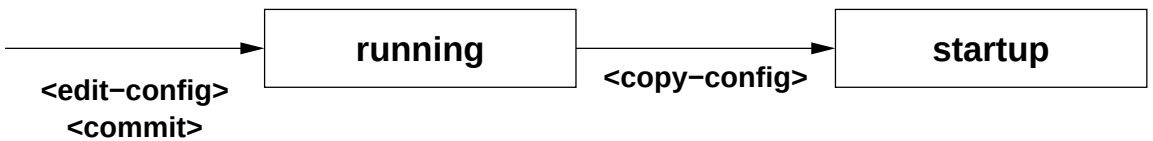
Direct Model



Candidate Model (optional)



Distinct Startup Model (optional)



- Some operations (`edit-config`) may support different error behaviours, including rollback behaviour.

Capability Exchange

Hello

After estabilishing a (secure) transport, both NETCONF protocol engines send a `hello` message to announce their capabilities and the session identifier.

```
A: <hello>
A:   <capabilities>
A:     <capability>
A:       urn:ietf:params:xml:ns:netconf:base:1.0
A:     </capability>
A:     <capability>
A:       urn:ietf:params:xml:ns:netconf:base:1.0#startup
A:     </capability>
A:   </capabilities>
A:   <session-id>4<session-id>
A: </hello>
```

Remote Procedure Calls

RPC protocol

The Remote Procedure Call (RPC) protocol consists of a <rpc/> message followed by an <rpc-reply/> message.

```
M: <rpc message-id="101"
M:      xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
M:   <get-config>
M:     <source>
M:       <running/>
M:     </source>
M:   </get-config>
M: </rpc>
A: <rpc-reply message-id="101"
A:      xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
A:   <data><!-- ...contents here... --></data>
A: </rpc-reply>
```

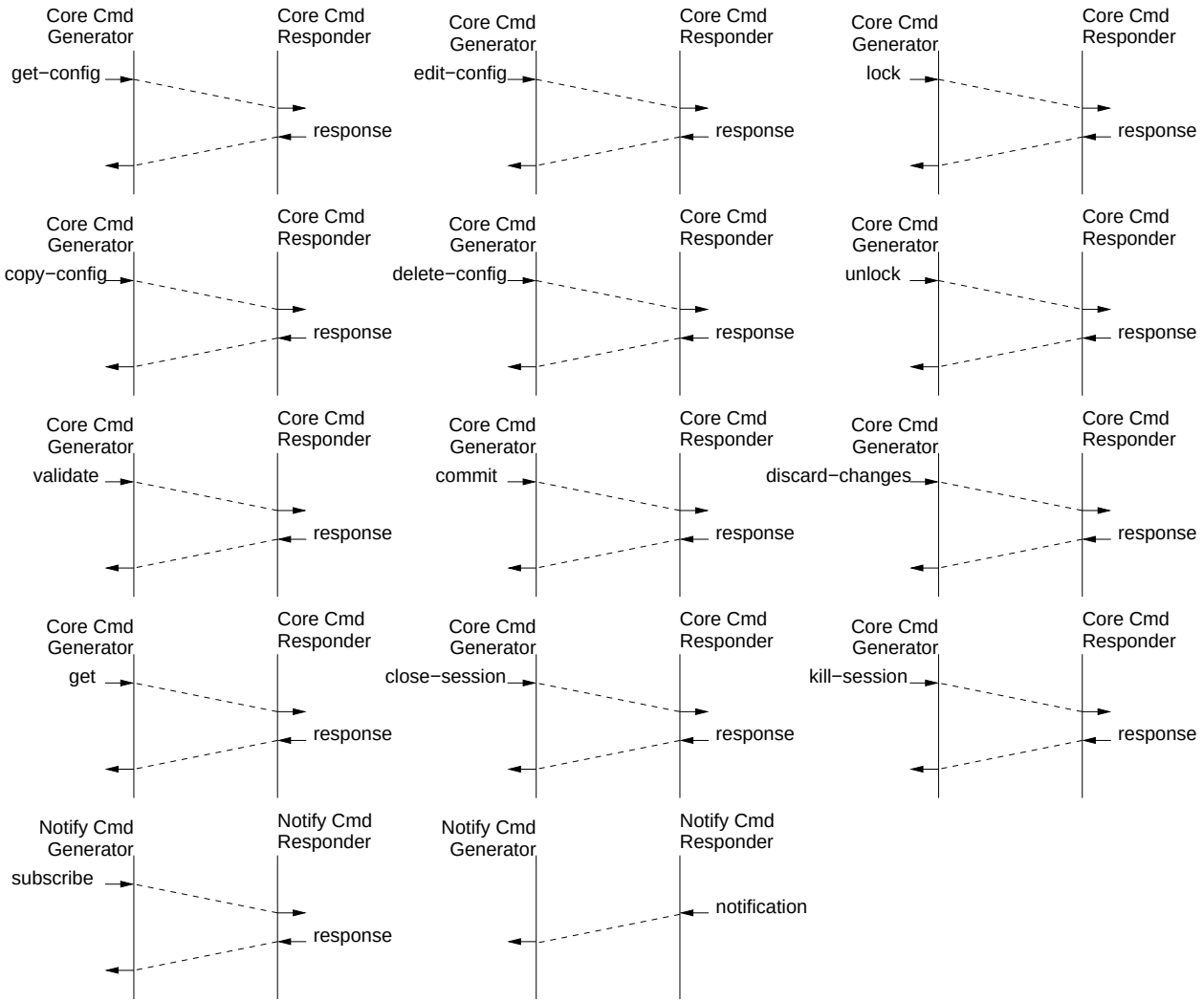
Remote Procedure Calls (cont.)

RPC protocol

RPC failures are indicated by one or more <rpc-error/> elements in the <rpc-reply/> element.

```
M: <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
M:   <get-config><source><running/></source></get-config>
M: </rpc>
A: <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
A:   <rpc-error>
A:     <error-type>rpc</error-type>
A:     <error-tag>missing-attribute</error-tag>
A:     <error-severity>error</error-severity>
A:     <error-info>
A:      <bad-attribute>message-id</bad-attribute>
A:      <bad-element>rpc</bad-element>
A:     </error-info>
A:   </rpc-error>
A: </rpc-reply>
```

NETCONF Operations Overview



NETCONF Operations

- `get-config(source, filter)`
  Retrieve a (filtered subset of a) configuration from the configuration datastore `source`.
- `edit-config(target, default-operation, test-option, error-option, config)`
  Edit the `target` configuration datastore by merging, replacing, creating, or deleting new config elements.
- `copy-config(target, source)`
  Copy the content of the configuration datastore `source` to the configuration datastore `target`.
- `delete-config(target)`
  Delete the named configuration datastore `target`.

NETCONF Operations (cont.)

- `lock(target)`
  Lock the configuration datastore `target`.
- `unlock(target)`
  Unlock the configuration datastore `target`.
- `get(filter)`
  Retrieve (a filtered subset of a) the running configuration and device state information.
- `close-session()`
  Gracefully close the current session.
- kill-session(session)
  Force the termination of the session `session`.

NETCONF Operations (cont.)

- `discard-changes()`
  Revert the candidate configuration datastore to the running configuration (#candidate capability).
- `validate(source)`
  Validate the contents of the configuration datastore `source` (#validate capability).
- `commit(confirmed, confirm-timeout)`
  Commit candidate configuration datastore to the running configuration (#candidate capability).
- `create-subscription(stream, filter, start, stop)`
  Subscribe to a notification `stream` with a given `filter` and the `start` and `stop` times.

Editing Configuration

merge

The configuration data is merged with the configuration at the corresponding level in the configuration datastore.

replace

The configuration data replaces any related configuration in the configuration datastore identified by the target parameter.

create

The configuration data is added to the configuration if and only if the configuration data does not already exist.

delete

The configuration data identified by the element containing this attribute is deleted in the configuration datastore.

Editing Configuration Example

```
M:  <rpc message-id="101"
M:       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
M:    <edit-config>
M:      <target>
M:        <running/>
M:      </target>
M:      <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
M:        <top xmlns="http://example.com/schema/1.2/config">
M:          <interface xc:operation="replace">
M:            <name>Ethernet0/0</name>
M:            <mtu>1500</mtu>
M:            <address>
M:              <name>192.0.2.4</name>
M:              <prefix-length>24</prefix-length>
M:            </address>
M:          </interface>
M:        </top>
M:      </config>
M:    </edit-config>
M:  </rpc>
```

Subtree Filtering

Subtree Filter Expressions

Subtree filter expressions select particular XML subtrees to include in `get` and `get-config` responses.

Namespace Selection

If the 'xmlns' attribute is present, then the filter output will only include elements from the specified namespace.

Attribute Match Expressions

The set of (unqualified or qualified) XML attributes present in any type of filter node form an "attribute match expression" The selected data must have matching values for every attribute of an attribute match expression.

Subtree Filtering (cont.)

Containment Nodes

For each containment node specified in a subtree filter, all data model instances must exactly match the specified namespaces, element hierarchy, and any attribute match expressions.

Selection Nodes

An empty leaf node within a filter is called a "selection node" and it selects the specified subtree(s) and it suppresses the automatic selection of the entire set of sibling nodes in the underlying data model.

Content Match Nodes

A leaf node that contains simple content is called a "content match node" and it selects some or all of its sibling nodes. It represents an exact-match filter on the leaf node element content.

Subtree Filtering Example

```
<filter type="subtree">

   <!-- namespace selection and containment node selection -->
   <t:top xmlns:t="http://example.com/schema/1.2/config">

      <!-- containment node selection -->
      <t:interfaces>

         <!-- containment node selection and attribute match expression -->
         <t:interface t:ifName="eth0">

            <!-- selection node -->
            <t:ifSpeed/>

            <!-- content match node -->
            <t:ifType>Ethernet</t:if-type>

         </t:interface>
      </t:interfaces>
   </t:top>
</filter>
```

NETCONF over SSH

- Motivation: Use an already deployed security protocol, thereby reducing the operational costs associated with key management.
- SSH supports multiple logical channels over one transport layer association.
- For framing purposes, the special end of message marker `]]>]]>` has been introduced.
- NETCONF over SSH has been selected as the mandatory to implement transport for NETCONF.

NETCONF over SSH Example

```
A: <?xml version="1.0" encoding="UTF-8"?>
A: <hello>
A:   <capabilities>
A:     <capability>
A:       urn:ietf:params:xml:ns:netconf:base:1.0
A:     </capability>
A:     <capability>
A:       urn:ietf:params:xml:ns:netconf:base:1.0#startup
A:     </capability>
A:   </capabilities>
A:   <session-id>4<session-id>
A: </hello>
A: ]]>]]>
```

NETCONF over SSH Example

```
M: <?xml version="1.0" encoding="UTF-8"?>
M: <hello>
M:   <capabilities>
M:     <capability>
M:       urn:ietf:params:xml:ns:netconf:base:1.0
M:     </capability>
M:   </capabilities>
M: </hello>
M: ]]>]]>
```

NETCONF over SSH Example

```
M: <?xml version="1.0" encoding="UTF-8"?>
M: <rpc message-id="105" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
M:   <get-config>
M:     <source>
M:       <running/>
M:     </source>
M:     <filter type="subtree">
M:       <config xmlns="http://example.com/schema/1.2/config">
M:         <users/>
M:       </config>
M:     </filter>
M:   </get-config>
M: </rpc>
M: ]]>]]>
```

NETCONF over SSH Example

```
A: <?xml version="1.0" encoding="UTF-8"?>
A: <rpc-reply message-id="105" xmlns="urn:ietf:params:xml:ns:netconf:base:1
A:   <data>
A:     <config xmlns="http://example.com/schema/1.2/config">
A:       <users>
A:         <user><name>root</name><type>superuser</type></user>
A:         <user><name>fred</name><type>admin</type></user>
A:         <user><name>barney</name><type>admin</type></user>
A:       </users>
A:     </config>
A:   </data>
A: </rpc-reply>
A: ]]>]]>
```

BEEP Protocol (RFC 3080)

- BEEP is a generic application protocol kernel for connection-oriented, asynchronous interactions.
- BEEP supports multiple channels, application layer framing and fragmentation.
- BEEP exchange styles:
  - `MSG/RPY`
  - `MSG/ERR`
  - `MSG/ANS`
- Integrates into SASL (RFC 2222) and TLS (RFC 2246) for security.
- Connections can be initiated by both participating peers (no strict client/server roles).

NETCONF over BEEP

- BEEP supports multiple logical channels.
- Every peer can be the initiator of a connection.
- SASL allows to map to existing security infrastructures.
- Framing and fragmentation services provided by BEEP.
- BEEP is currently not widely deployed and there is a lack of operational experience with BEEP in the operator community.
- BEEP is an optional NETCONF transport.

NETCONF over BEEP Example

```
M: MSG 0 1 . 10 48 101
M: Content-Type: application/beep+xml
M: <start number="1">
M:   <profile uri="http://iana.org/beep/netconf" />
M: </start>
M: END
A: RPY 0 1 . 38 87
A: Content-Type: application/beep+xml
A:
A: <profile uri="http://iana.org/beep/netconf" />
A: END
```

NETCONF over BEEP Example

```
A: MSG 1 0 . 0 436
A: Content-Type: application/beep+xml
A:
A: <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
A:   <capabilities>
A:     <capability>
A:       urn:ietf:params:xml:ns:netconf:base:1.0
A:     </capability>
A:     <capability>
A:       urn:ietf:params:xml:ns:netconf:base:1.0#startup
A:     </capability>
A:   </capabilities>
A:   <session-id>4</session-id>
A: </hello>
A: END
M: RPY 1 0 . 0 0
M: END
```

NETCONF over BEEP Example

```
M: MSG 1 42 . 24 344
M: Content-Type: text/xml; charset=utf-8
M:
M: <rpc message-id="105" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
M:   <get-config>
M:     <source>
M:       <running/>
M:     </source>
M:     <filter type="subtree">
M:       <config xmlns="http://example.com/schema/1.2/config">
M:         <users/>
M:       </config>
M:     </filter>
M:   </get-config>
M: </rpc>
M: END
```

NETCONF over BEEP Example

```
A: RPY 1 42 . 24 542
A: Content-Type: text/xml; charset=utf-8
A:
A: <rpc-reply message-id="105" xmlns="urn:ietf:params:xml:ns:netconf:base:1
A:   <data>
A:     <config xmlns="http://example.com/schema/1.2/config">
A:       <users>
A:         <user><name>root</name><type>superuser</type></user>
A:         <user><name>fred</name><type>admin</type></user>
A:         <user><name>barney</name><type>admin</type></user>
A:       </users>
A:     </config>
A:   </data>
A: </rpc-reply>
A: END
```

NETCONF over SOAP/HTTP[S]

- Instead of inventing a special purpose RPC protocol, use existing Web Services standards.
- Pros:
  - more developers / tools available
  - better integration with IT infrastructure
- Cons:
  - base technology not under control of the IETF
  - unneeded complexity
  - interoperability problems (immature technology)
  - HTTP is a bad generic application protocol kernel
- Note: Transport mapping does not map NETCONF operations to SOAP operations!

NETCONF over SOAP/HTTP Example

```
M: POST /netconf HTTP/1.1
M: Host: netconfdevice
M: Content-Type: text/xml; charset=utf-8
M: Accept: application/soap+xml, text/*
M: Cache-Control: no-cache
M: Pragma: no-cache
M: Content-Length: 490
M:
M: <?xml version="1.0" encoding="UTF-8"?>
M: <soapenv:Envelope
M:   xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
M:   <soapenv:Body>
M:     <rpc message-id="101"
M:          xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
M:       <get-config>
M:         <source><running/></source>
M:         <filter type="subtree">
M:           <top xmlns="http://example.com/schema/1.2/config">
M:             <users/>
M:           </top>
M:         </filter>
M:       </get-config>
M:     </rpc>
M:   </soapenv:Body>
M: </soapenv:Envelope>
```

NETCONF over SOAP/HTTP Example

```
A: HTTP/1.1 200 OK
A: Content-Type: application/soap+xml; charset=utf-8
A: Content-Length: 668
A:
A: <?xml version="1.0" encoding="UTF-8"?>
A: <soapenv:Envelope
A:   xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
A:   <soapenv:Body>
A:     <rpc-reply message-id="101"
A:                xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
A:       <data>
A:         <top xmlns="http://example.com/schema/1.2/config">
A:           <users>
A:             <user>
A:               <name>root</name>
A:               <type>superuser</type>
A:               <full-name>Charlie Root</full-name>
A:                 <dept>1</dept>
A:                 <id>1</id>
A:               </company-info>
A:             </user>
A:           </users>
A:         </top>
A:       </data>
A:     </rpc-reply>
A:   </soapenv:Body>
A: </soapenv:Envelope>
```

YANG, YIN, XSD, RELAX NG

YANG's purpose

YANG is an extensible NETCONF data modeling language able to model configuration data, state data, operations, and notifications. YANG definitions directly map to XML content.

YANG vs. YIN

YANG uses a compact SMIng like syntax since readability is highest priority. YIN is an XML version of YANG (lossless roundtrip conversion).

YANG vs. XSD or RELAX NG

YANG can be translated to XML Schema (XSD) and RELAX NG so that existing tools can be utilized.

YANG and IETF NETMOD WG History

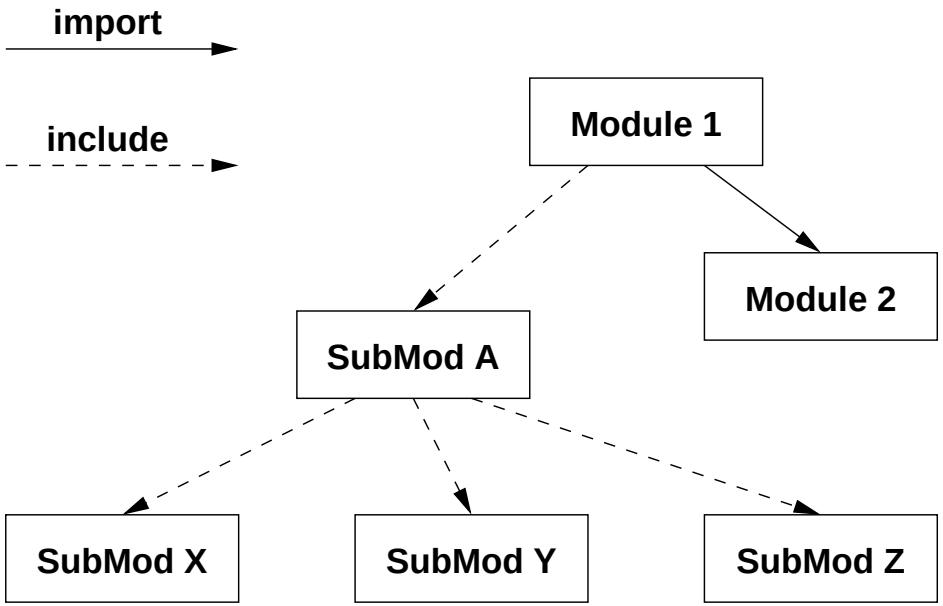
YANG Milestones (pre IETF)

- YANG design team created in Spring 2007
- Three design team meetings (USA, London, Stockholm)
- YANG discussions at the 71st IETF (Vancouver)
- YANG discussions at the 72nd IETF (Philadelphia)

NETMOD Milestones

- Apr. 2008: NETMOD WG chartered
- Aug. 2008: initial YANG, YIN, DSDL, ... documents
- Mar. 2009: submit architecture document to the IESG
- Sep. 2009: submit YANG, YIN, DSDL, ... to the IESG

Modules and submodules



Module

A self-contained collection of YANG definitions.

Submodule

A partial module definition which contributes derived types, groupings, data nodes, RPCs, and notifications to a module.

Module Example

```
module acme-module {
    namespace "http://acme.example.com/module";
    prefix "acme";

    import "yang-types" {
        prefix "yang";
    }
    include "acme-system";

    organization "ACME Inc.";
    contact      "support@acme.example.com";
    description
        "The module for entities implementing the ACME products";

    revision "2007-06-09" {
        description "Initial revision.";
    }
}
```

Built-in Data Types

Category	Types	Restrictions
Integral	{u,}int{8,16,32,64}	range
Decimals	decimal64	range, fraction-digits
String	string	length, pattern
Enumeration	enumeration	enum
Bool and Bits	boolean, bits	
Binary	binary	length
References	leafref	path
References	identityref	base
References	instance-identifier	
Other	empty	

Type system

The data type system is mostly an extension of the SMIng type system, accommodating XML and XSD requirements.

Example: typedef

```
module inet-types {

    namespace "urn:ietf:params:xml:ns:yang:inet-types";
    prefix "inet";

    typedef ipv4-address {
        type string {
            pattern '(([0-1]?[0-9]?[0-9]|2[0-4][0-9]|25[0-5])\.){3}'
                  + '([0-1]?[0-9]?[0-9]|2[0-4][0-9]|25[0-5])'
                  + '(%[\p{N}\p{L}]+)?';
        }
    }

    // ...

    typedef ip-address {
        type union {
            type inet:ipv4-address;
            type inet:ipv6-address;
        }
        description "Represents a version neutral IP address.";
    }
}
```

Leafs, Leaf-lists, Container, Lists

leaf

A `leaf` has one value, no children, one instance.

leaf-list

A `leaf-list` has one value, no children, multiple instances.

container

A `container` has no value, holds related children, has one instance.

list

A `list` has no value, holds related children, has multiple instances, has a key property.

Example: leaf and leaf-list

```
leaf domain {
    type inet:domain-name; // values are typed (type imported)
    mandatory true;        // must exist in a valid configuration
    config true;           // part of the set of configuration objects
    description
        "The host name of this system.";
}

// XML: <domain>example.com</domain>

leaf-list search {
    type inet:domain-name; // imported from the module with prefix inet
    ordered-by user;       // maintain the order given by the user
    description
        "List of domain names to search.";
}

// XML: <search>eng.example.com</search>
// XML: <search>example.com</search>
```

Example: container

```
container system {
    config true;
    leaf hostname {
        type inet:domain-name;
    }
    container resolver {
       leaf domain { /* see above */ }
       leaf-list search { /* see above */ }
       description
            "The configuration of the resolver library.";
    }
}

// XML: <system>
// XML:   <hostname>server.example.com</hostname>
// XML:   <resolver>
// XML:     <domain>example.com</domain>
// XML:     <search>eng.example.com</search>
// XML:     <search>example.com</search>
// XML:   </resolver>
// XML: </system>
```

Example: list

```
    list nameserver {
        key address;
        leaf address {
            type inet:ip-address;
        }
        leaf status {
            type enumeration {
                enum enabled; enum disabled; enum failed;
            }
        }
    }

// XML:   <nameserver>
// XML:     <address>192.0.2.1</address>
// XML:     <status>enabled</status>
// XML:   </nameserver>
// XML:   <nameserver>
// XML:     <address>192.0.2.2</address>
// XML:     <status>failed</status>
// XML:   </nameserver>
```

Augment, Must, When

augment

The `augment` statement can be used to place nodes into an existing hierarchy using the current module's namespace.

must

The `must` statement can be used to express constraints (in the form of XPATH expressions) that must be satisfied by a valid configuration.

when

The `when` statement can be used to define sparse augmentations where nodes are only added when a condition (expressed in the form of an XPATH expression) is true.

Example: augment and presence

```
augment system/resolver {
    container debug {
        presence "enables debugging";
        description
           "This container enables debugging.";
        leaf level {
            type enumeration {
                enum low;
                enum medium;
                enum full;
            }
            default "medium";
            mandatory false;
            description
               "The debugging level; default is medium debug information.";
        }
    }
}

// XML: <system><resolver>
// XML:    <debug/>
// XML: </resolver></system>
```

Example: augment and must

```
augment system/resolver {
    leaf access-timeout {
        type uint32;
        unit "seconds";
        mandatory true;
        description "Maximum time without server response.";
    }
    leaf retry-timer {
        type uint32;
        units "seconds";
        description "Period after which to retry an operation";
        must "$this < ../access-timeout" {
            error-app-tag "retry-timer-invalid";
            error-message "The retry timer must be less "
                        + "than the access timeout";
        }
    }
}
```

Example: augment and when

```
augment system/resolver/nameserver {
    when "status = enabled";
    leaf tx {
        type yang:counter32;
        config false;
    }
    leaf rx {
        type yang:counter32;
        config false;
    }
}

// XML:   <nameserver>
// XML:     <address>192.0.2.1</address>
// XML:     <status>enabled</status>
// XML:     <tx>2345</tx>
// XML:     <rx>1234</rx>
// XML:   </nameserver>
// XML:   <nameserver>
// XML:     <address>192.0.2.2</address>
// XML:     <status>failed</status>
// XML:   </nameserver>
```

Grouping and Choice

grouping

A `grouping` is a reusable collection of nodes. The `grouping` mechanism can be used to emulate structured data types or objects. A `grouping` can be refined when it is used.

choice

A `choice` allows one alternative of the choice to exist. The `choice` mechanism can be used to provide extensibility hooks that can be exploited using `augments`.

- Should a `grouping` be considered a template mechanism or a structured data type mechanism?

Example: grouping

```
grouping target {
    leaf address {
        type inet:ip-address;
        description "Target IP address.";
    }
    leaf port {
        type inet:ip-port;
        description "Target port number.";
    }
}

list nameserver {
    key "address port";
    uses target;
}

// XML: <nameserver>
// XML:   <address>192.0.2.1</address>
// XML:   <port>53</port>
// XML: </nameserver>
```

Example: choice

```
container transfer {
    choice how {
        default interval;
        case interval {
            leaf interval {
                type uint16; default 30; units minutes;
            }
        }
        case daily {
            leaf daily {
                type empty;
            }
            leaf time-of-day {
                type string; units 24-hour-clock; default 1am;
            }
        }
        case manual {
            leaf manual {
                type empty;
            }
        }
    }
}
```

Notification and RPC

notification

The `notification` statement can be used to define the contents of notifications.

rpc

The `rpc` statement can be used to define operations together with their input and output parameters carried over the RPC protocol.

- Should the `rpc` statement be called `operation` since it is used to define operations?
- Should all NETCONF operations be formally defined in YANG?

Example: notification

```
notification nameserver-failure {
    description
       "A failure of a nameserver has been detected and
        the server has been disabled."
    leaf address {
        type leafref {
            path "/system/resolver/nameserver/address";
        }
    }
}

// MSG: <notification>
// MSG:   <eventTime>2008-06-03T18:34:50+02:00</eventTime>
// MSG:   <nameserver-failure>
// MSG:     <address>192.0.2.2</address>
// MSG:   </nameserver-failure>
// MSG: </notification>
```

Example: rpc

```
rpc activate-software-image {
    input {
        leaf image name {
            type string;
        }
    }
    output {
        leaf status {
            type string;
        }
    }
}

// RPC: <rpc xmlns="urn:mumble" message-id="42">
// RPC:   <activate-software-image>
// RPC:     <image-name>image.tgz</image-name>
// RPC:   </activate-software-image>
// RPC: </rpc>
```

Available Tools

pyang

Open source YANG validator and translator written in Python.

yangdump

Closed source YANG validator and translator written in C.

smidump

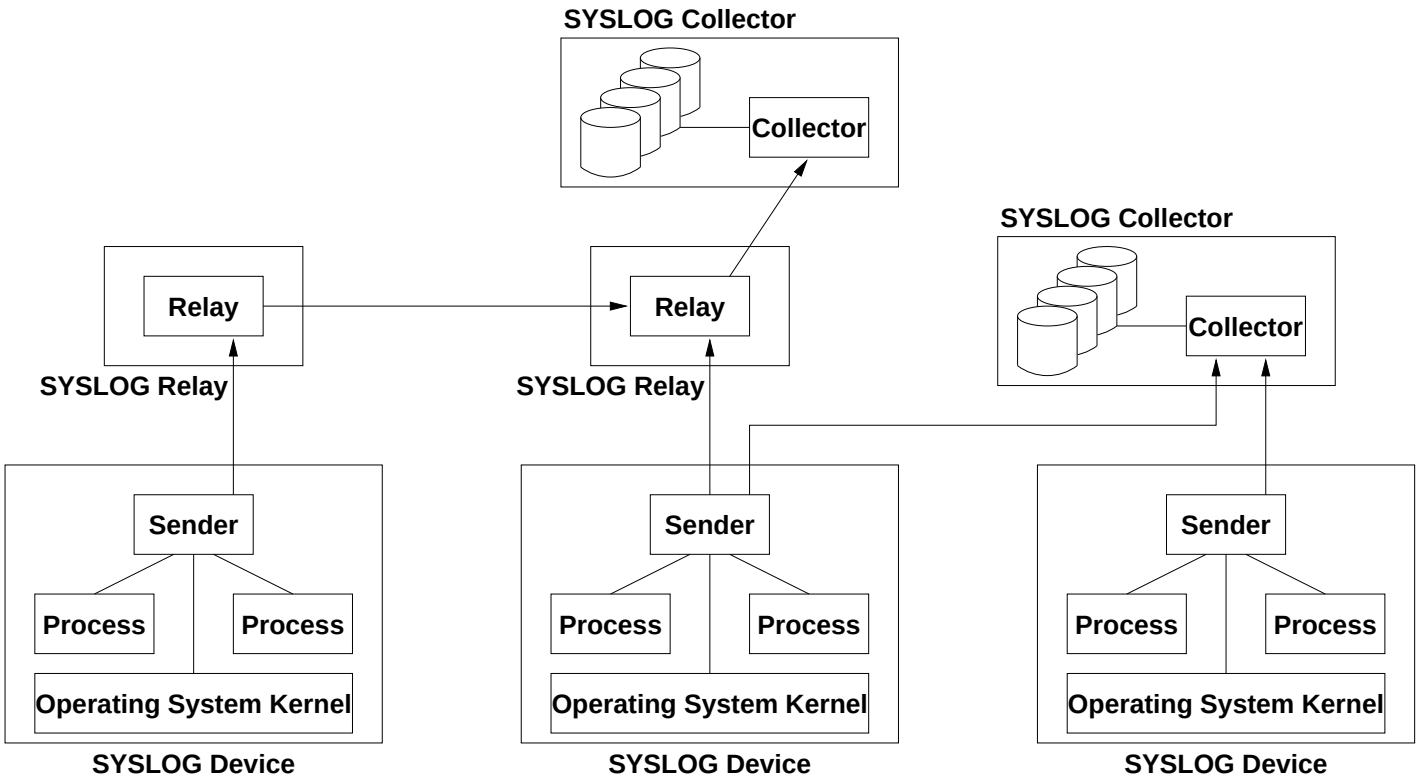
Open source SMI to YANG translator written in C.

emacs

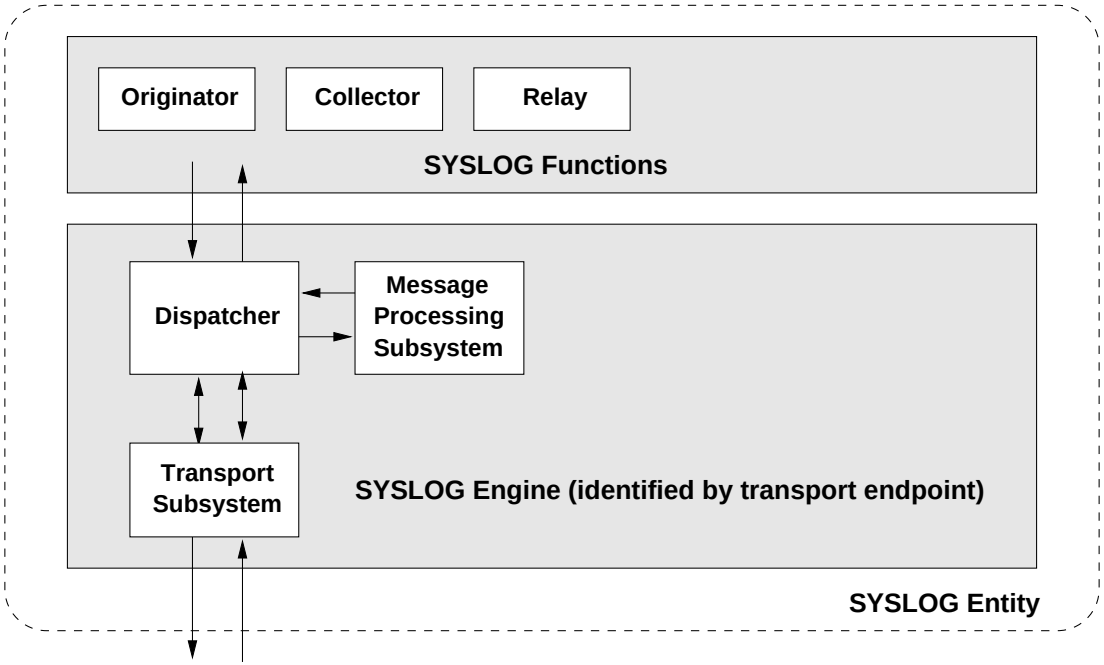
Open source YANG editing mode for the emacs editor.

System Logging Protocol (SYSLOG)

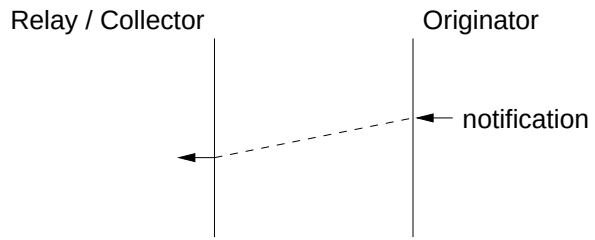
System Logging Protocol (SYSLOG)



System Logging Protocol (SYSLOG)

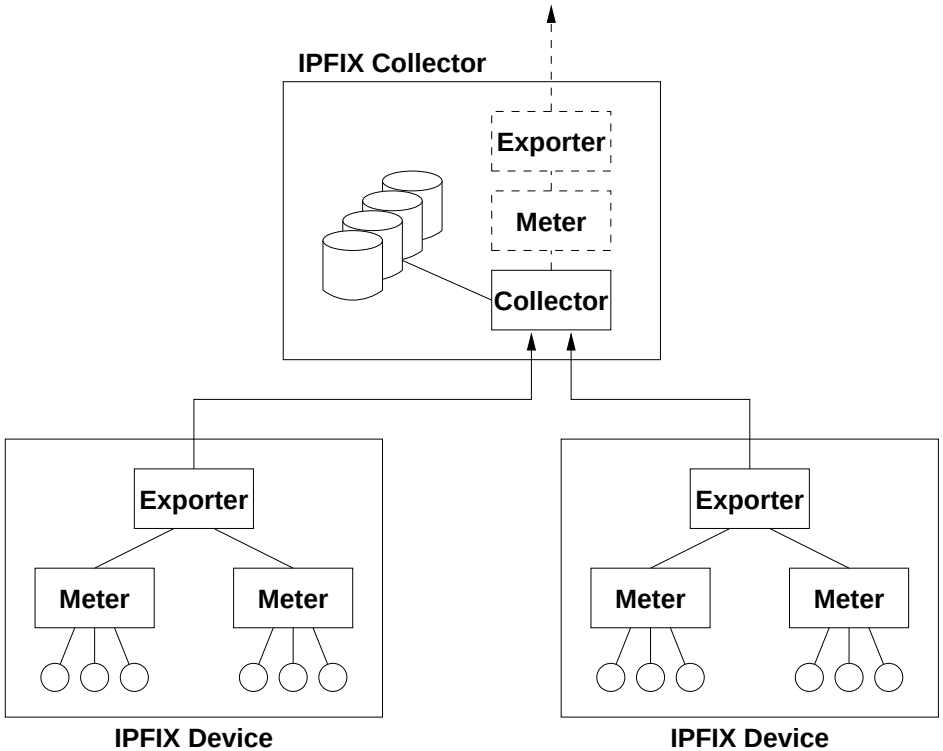


System Logging Protocol (SYSLOG)

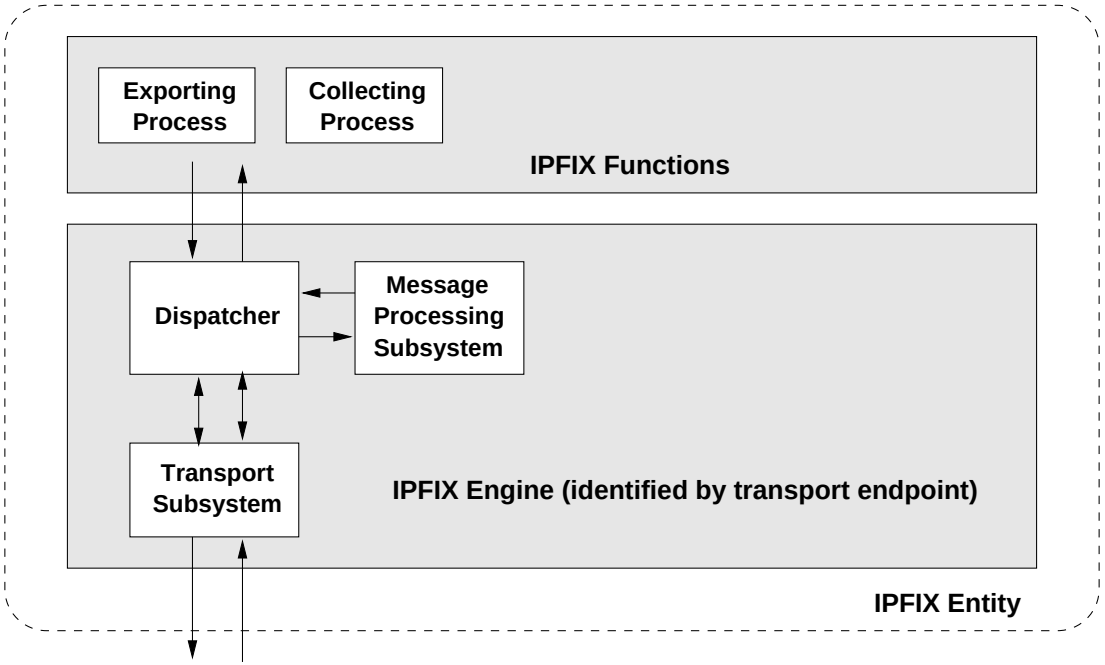


Traffic Analysis using NETFLOW / IPFIX

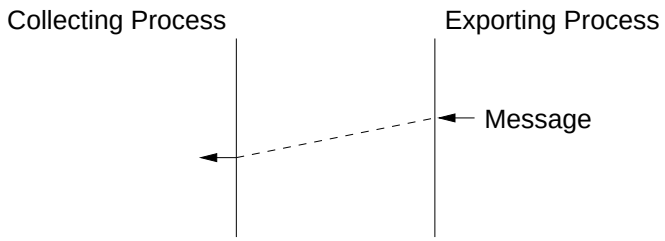
Flow Information Export Protocol (IPFIX)



Flow Information Export Protocol (IPFIX)



Flow Information Export Protocol (IPFIX)



References I

W. Stallings.

SNMP, SNMPv2, SNMPv3, and RMON 1 and 2.
Addison-Wesley, 3 edition, 1999.

D. Zeltserman.

A Practical Guide to SNMPv3 and Network Management.
Prentice Hall, 1999.

U. Blumenthal and B. Wijnen.

Security Features of SNMPv3.
Simple Times, 5(1), December 1997.

J. Case, R. Mundy, D. Partain, and B. Stewart.

Introduction and Applicability Statements for Internet Standard Management Framework.
RFC 3410, SNMP Research, Network Associates Laboratories, Ericsson, December 2002.

D. Harrington, R. Presuhn, and B. Wijnen.

An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks.
RFC 3411, Enterasys Networks, BMC Software, Lucent Technologies, December 2002.

J. Schönwälder.

Handbook of Network and System Administration, chapter Internet Management Protocols, pages 295–328.

Elsevier, November 2007.

L. Sanchez, K. McCloghrie, and J. Saperia.

Requirements for Configuration Management of IP-based Networks.
RFC 3139, Megisto, Cisco, JDS Consultant, June 2001.

References II

J. Schönwälder.

Overview of the 2002 IAB Network Management Workshop.
RFC 3535, International University Bremen, May 2003.

R. Mahajan, D. Wetherall, and T. Anderson.

Understanding BGP Misconfiguration.
In *Proc. SIGCOMM 2002*. ACM, August 2002.

D. Oppenheimer, A. Ganapathi, and D. A. Patterson.

Why do Internet services fail, and what can be done about it?
In *Proc. 4th Usenix Symposium on Internet Technologies and Systems*. Usenix, March 2003.

R. Enns.

NETCONF Configuration Protocol.
RFC 4741, Juniper Networks, December 2006.

M. Wasserman and T. Goddard.

Using the NETCONF Configuration Protocol over Secure SHell (SSH).
RFC 4742, ThingMagic, ICEsoft Technologies, December 2006.

T. Goddard.

Using NETCONF over the Simple Object Access Protocol (SOAP).
RFC 4743, ICEsoft Technologies, December 2006.

E. Lear.

Using the NETCONF Protocol over the Blocks Extensible Exchange Protocol (BEEP).
RFC 4744, Cisco Systems, December 2006.

References III

S. Chisholm and H. Trevino.

NETCONF Event Notifications.
RFC 5277, Nortel, Cisco, July 2008.

M. Badra.

NETCONF over Transport Layer Security (TLS).
RFC 5539, CNRS/LIMOS Laboratory, May 2009.

M. Bjorklund.

YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF).
RFC 6020, Tail-f Systems, October 2010.

J. Schönwälder.

Common YANG Data Types.
RFC 6021, Jacobs University, October 2010.

P. Shafer.

An NETCONF- and NETMOD-based Architecture for Network Management.
Internet Draft (work in progress) <draft-ietf-netmod-yang-arch-02.txt>, Juniper Networks, May 2009.

J. Schönwälder.

Protocol Independent Network Management Data Modeling Languages - Lessons Learned from the SMIng Project.
IEEE Communications Magazine, 46(5):148–153, May 2008.

J. Schönwälder, M. Björklund, and P. Shafer.

Network Configuration Management Using NETCONF and YANG.
IEEE Communications Magazine, 48(9):166–173, September 2010.

References IV

R. Gerhards.

The Syslog Protocol.
RFC 5424, Adiscon GmbH, March 2009.

F. Miao, Y. Ma, and J. Salowey.

Transport Layer Security (TLS) Transport Mapping for Syslog.
RFC 5425, Huawei Technologies, Cisco Systems, March 2009.

A. Okmianski.

Transmission of Syslog Messages over UDP.
RFC 5426, Cisco Systems, March 2009.

G. Sadasivan, N. Brownlee, B. Claise, and J. Quittek.

Architecture for IP Flow Information Export.
RFC 5470, Rohati Systems, CAIDA, University of Auckland, Cisco Systems, NEC, March 2009.

B. Claise.

Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information.
RFC 5101, Cisco Systems, January 2008.

J. Quittek, S. Bryant, B. Claise, P. Aitken, and J. Meyer.

Information Model for IP Flow Information Export.
RFC 5102, NEC, Cisco Systems, PayPal, January 2008.

B. Trammell, E. Boschi, L. Mark, T. Zseby, and A. Wagner.

Specification of the IP Flow Information Export (IPFIX) File Format.
RFC 5655, Hitachi Europe, Fraunhofer IFAM, Fraunhofer FOKUS, ETH Zurich, October 2009.