Part 4: Multimedia Protocols

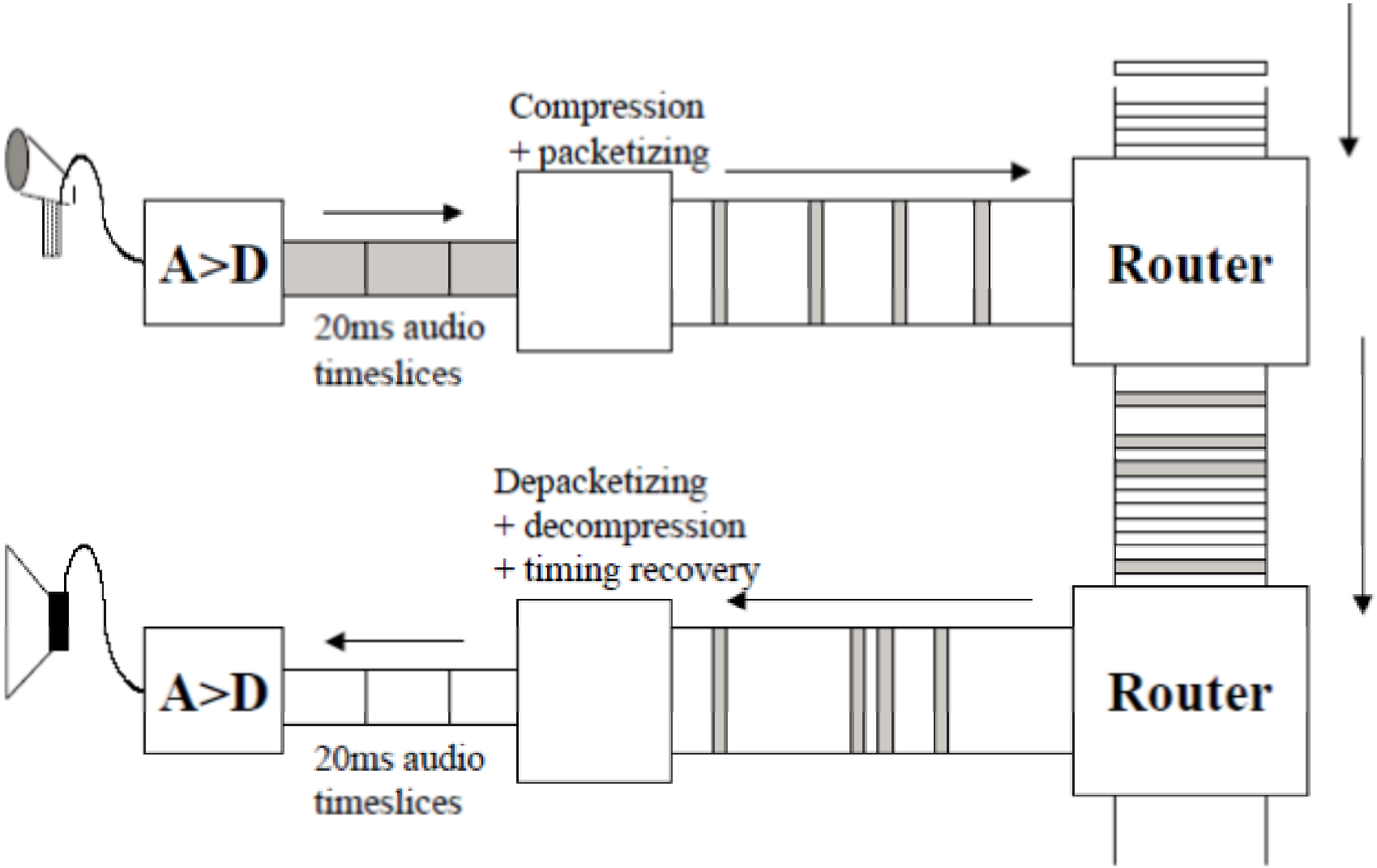
- Real-time Transport Protocol (RTP)
- Resource Reservation Protocol (RSVP)
- Stream Control Transmission Protocol (SCTP)
- AAA: RADIUS and Diameter
- Multi-resource Allocation (MRA)
- Network Management
- Message Broker (RabbitMQ)
- Virtual Private Networks (VPN)
- Content Distribution Networks (CDN)
- Software-defined Networks (SDN)

Part 4: Multimedia Protocol Components

- Real-time Transport Protocol (RTP)
  - Basic Operations and Goals
  - Terms and Definitions
  - Mechanisms and Packet Format
  - Payload Types and Usage Scenarios
  - Media Packetization Schemes
  - Real-time Transport Control Protocol (RTCP)
  - Message Types
  - Statistics Collection
  - Payload Types

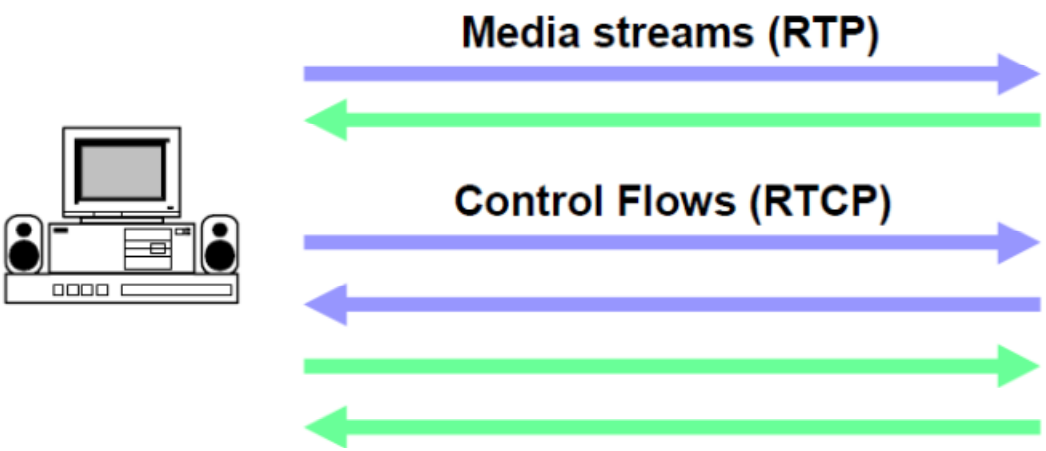
Partial thanks to Jörg Ott, Aalto University, Finland

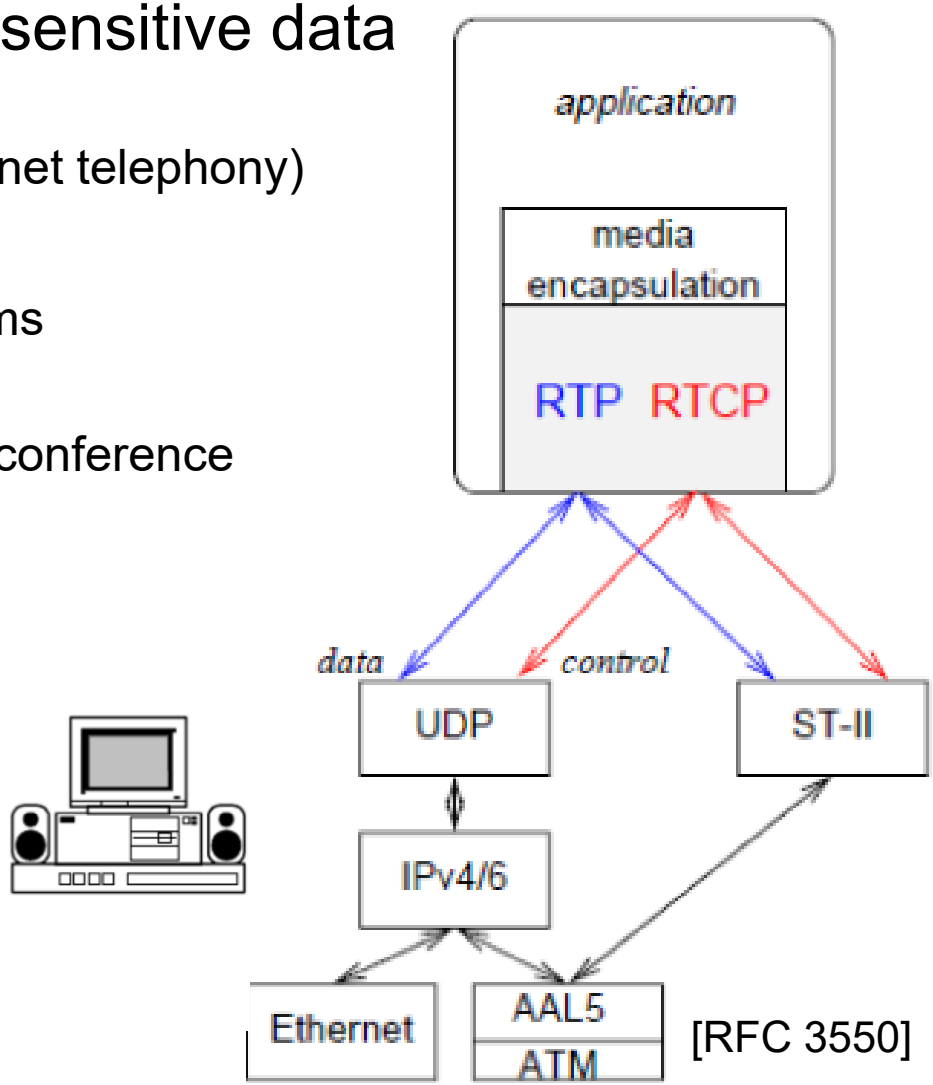
Internet Real-time Traffic



RTP Operation

- End-to-end delivery for temporally sensitive data
  - Unicast and multicast
  - Initiated by ITU H.323 (conferencing, Internet telephony)
- RTP Functionality
  - Framing for audio/video information streams
  - Reserve intra- and inter-stream timing
  - Mechanisms for awareness of others in a conference
  - RTP sessions





[RFC 3550]

RTP Goals and Outcome

- Flexible: provide mechanism, don't dictate algorithms
- Protocol-neutral: UDP/IP, ST-II, IPX, ATM-AALx, . . .
- Secure: support for encryption, possibly authentication
- Standard RTP packet header
  - Independent of payload type
  - Possibly seconded by payload header
- Mechanisms
  - Detect packet loss, cope with reordering
    - Sequence number per media stream
  - Determine variations in transmission delays
    - Media specific time stamp (*e.g.*, 8 kHz for PCM audio)
    - Allows receiver to adapt playout point for continuous replay
  - Source identification
    - Possibly mixed from several sources
  - Payload type identifier

ST-II: Internet Stream Protocol II
IPX: Internetwork Packet eXchange
PCM: Pulse Code Modulation

RTP Characteristics

- Data source and payload type identification
- Data sequencing
- Timing and cross-media synchronization
- Delivery monitoring
  - User information, tools for fault diagnosis, feedback on the quality of data distribution
- Providing tools for handling network and host Heterogeneity

- RTP does not provide for
  - Network resource reservations
  - Quality-of-Service (QoS) guarantees
  - Reliable data delivery
  - Multicast address alloction

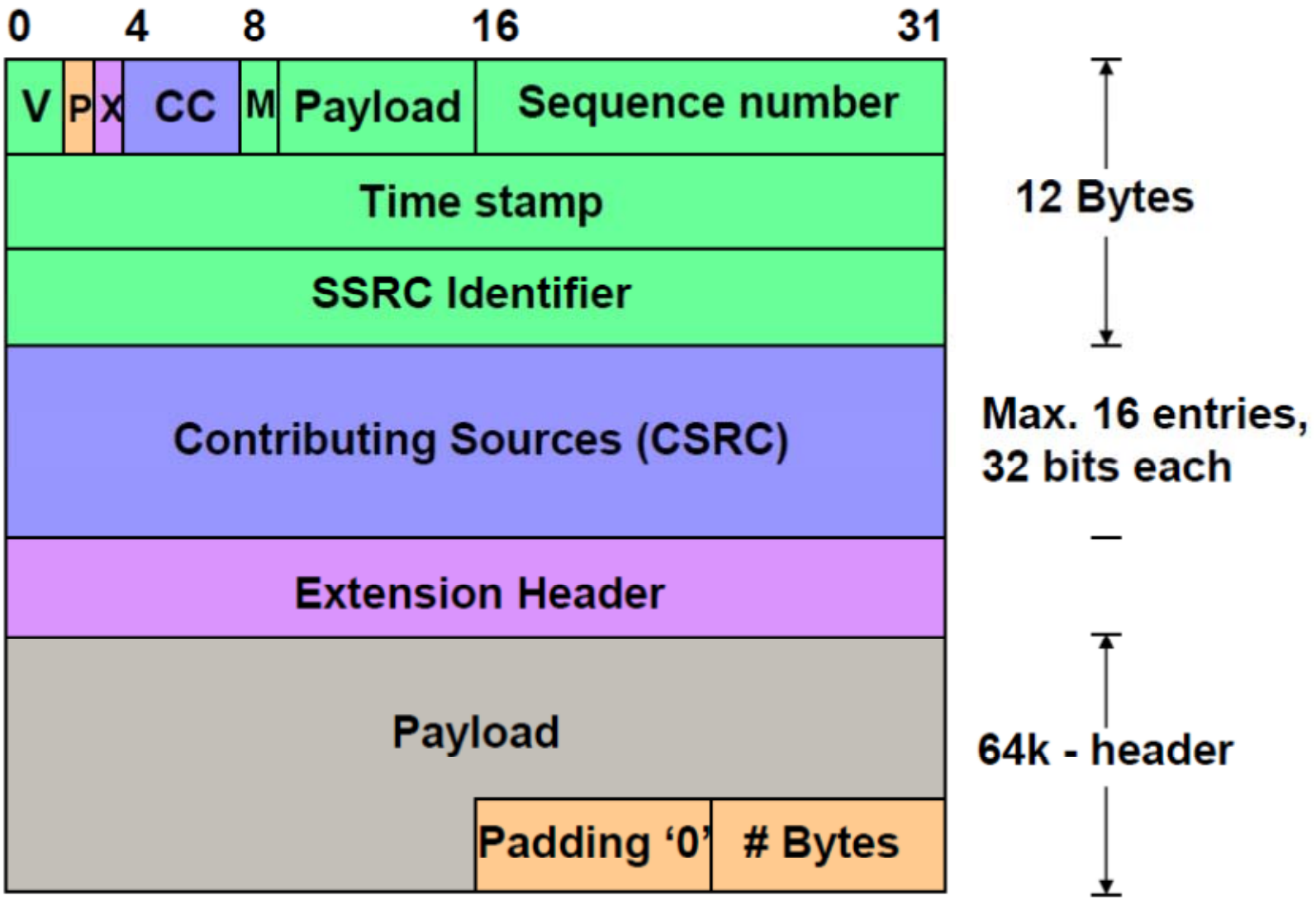
RTP Terms and Definitions

- RTP session
  - Defined by a pair of transport addresses
  - Network address + { port(RTP) must be even# ; port(RTCP) next higher # }
- RTP payload
  - Data transported in an RTP packet
- SSRC
  - Synchronization SouRCe, identifies a unique RTP packet sender
  - The packet IP source address does not identify the sender, which means users can move and transcoders can be employed
- CSRC
  - Contributing SouRCe. If streams are mixed, the CSRC identifies those sources, which were mixed together

RTP Mechanisms

- Real-Time Transport Protocol (RTP) = data + control
  - Data: timing, loss detection, content labeling, talkspurts, encryption
  - Control: RTCP periodic in T
- Receivers constantly measure transmission quality
  - Delay, jitter, packet loss
- Regular control information exchange between senders and receivers
  - Feedback to sender (receiver report)
  - Feed forward to recipients (sender report)
- Allows applications to adapt to current QoS
- Overhead limited to a small fraction (default: 5% max.) of total bandwidth per RTP session
  - Members estimate number of participants
  - Adapt their own transmission rate
- Obtaining sufficient capacity: outside of RTP

RTP Packet Format (1)



RTP Packet Format (2)

- V: Version, version 2 defined in RFC 1889
- P: Padding in Byte as indicated in last byte
- X: eXtension bit, extension header is present
- Extension header: single additional header (TLV coded)
- CC: CSRC count in terms of number of contributing sources
- CSRC: Contributing SouRCes, which sources have been “mixed” to produce this packet’s contents
- M: Marker bit to marks semantic boundaries in media stream
- Payload type indicates packet content type
- Sequence number of the packet in the media stream strictly monotonically increasing
- Timestamp indicates the instant when the packet contents was sampled (measured to media-specific clock)
- SSRC: Synchronization SouRCe identification of packet origin

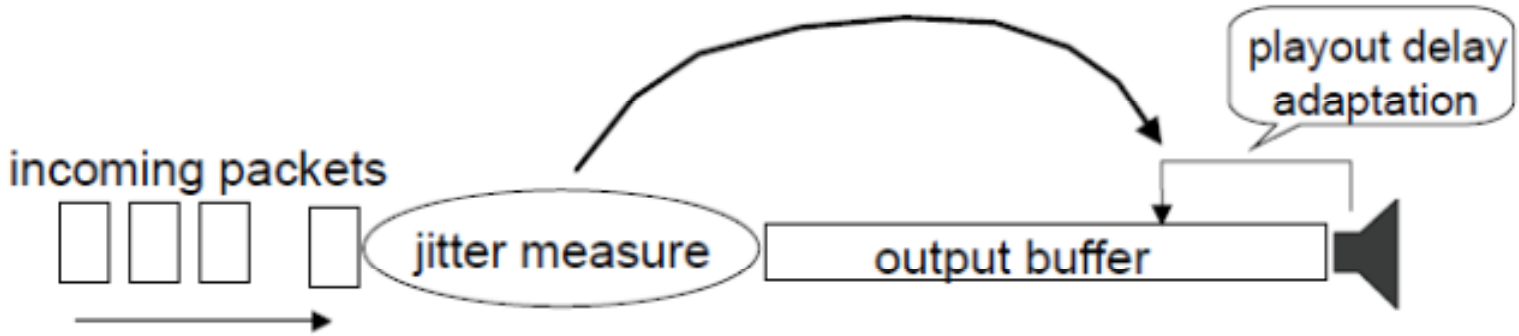
RTP Payload Types

- 7 bit payload type identifier
  - Some numbers statically assigned
  - Dynamic payload types identifiers for extensions
  - Mapping to be defined outside of RTP
    - Control protocol, *e.g.*, SDP “a=rtpmap:”

- Payload formats defined for many audio/video encodings
  - Conferencing profile document RFC 3551
    - Audio: G.711, G.722, G.723.1, G.728, GSM, CD, DVI, …
  - In codec-specific RFCs
    - Audio: Redundant Audio, MP-3, ...
    - Video: JPEG, H.261, MPEG-1, MPEG-2, H.263, H.263+, BT.656
    - Others: DTMF, text, SONET, ...
  - Generic formats
    - Generic FEC, multiplexing

RTP Usage Scenario (1)

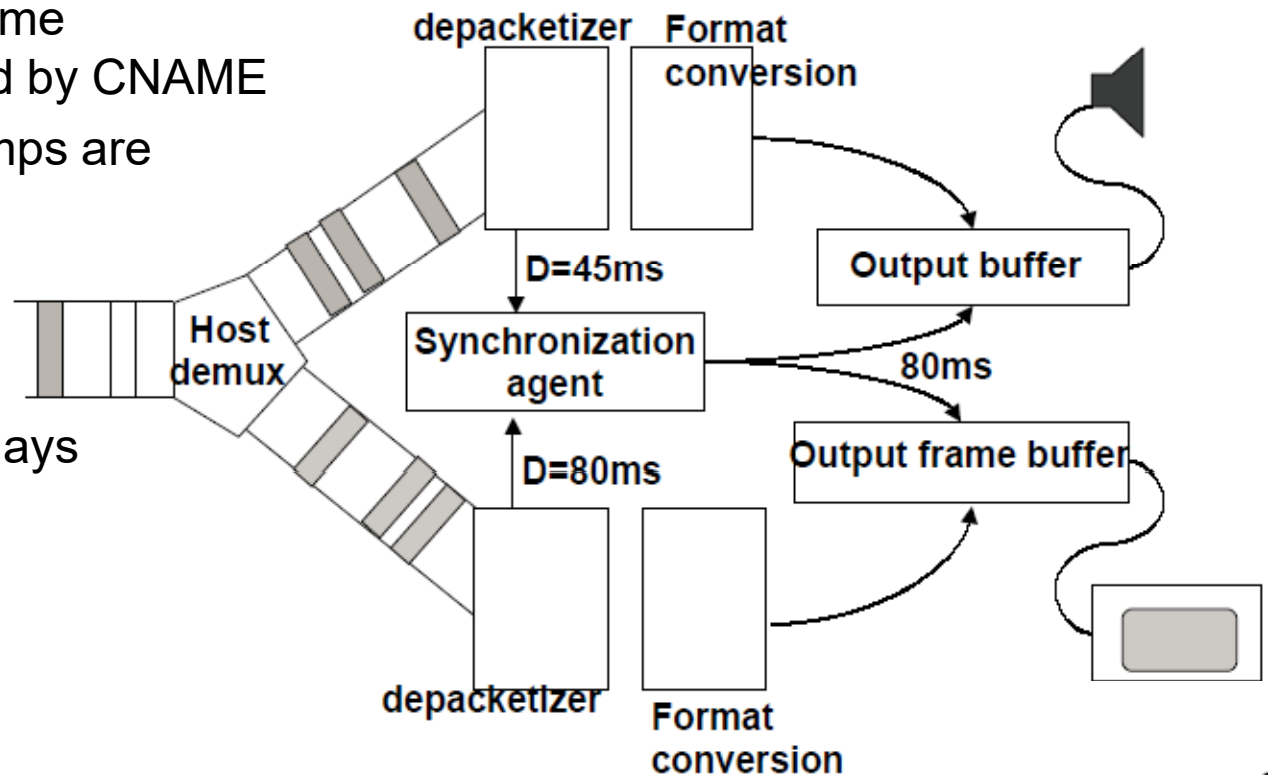
- Example 1: An audio conference session
  - Each packet carries
    - A SSRC
    - A sequence number for data ordering and loss detection
    - A timestamp for inter-packet gap calculation



  - Question: how long to hold the data before playing out?
  - Play-out delay must be long enough so that once a talk spurt starts play-out, all data in the spurt will (most likely) be available in time
    wait time = delay jitter = longest-delay - shortest delay
  - Wait for at least the jitter time long before playing the first packet in a talk spurt
  - Play out the whole talk spurt at the medium's clock

RTP Usage Scenario (2)

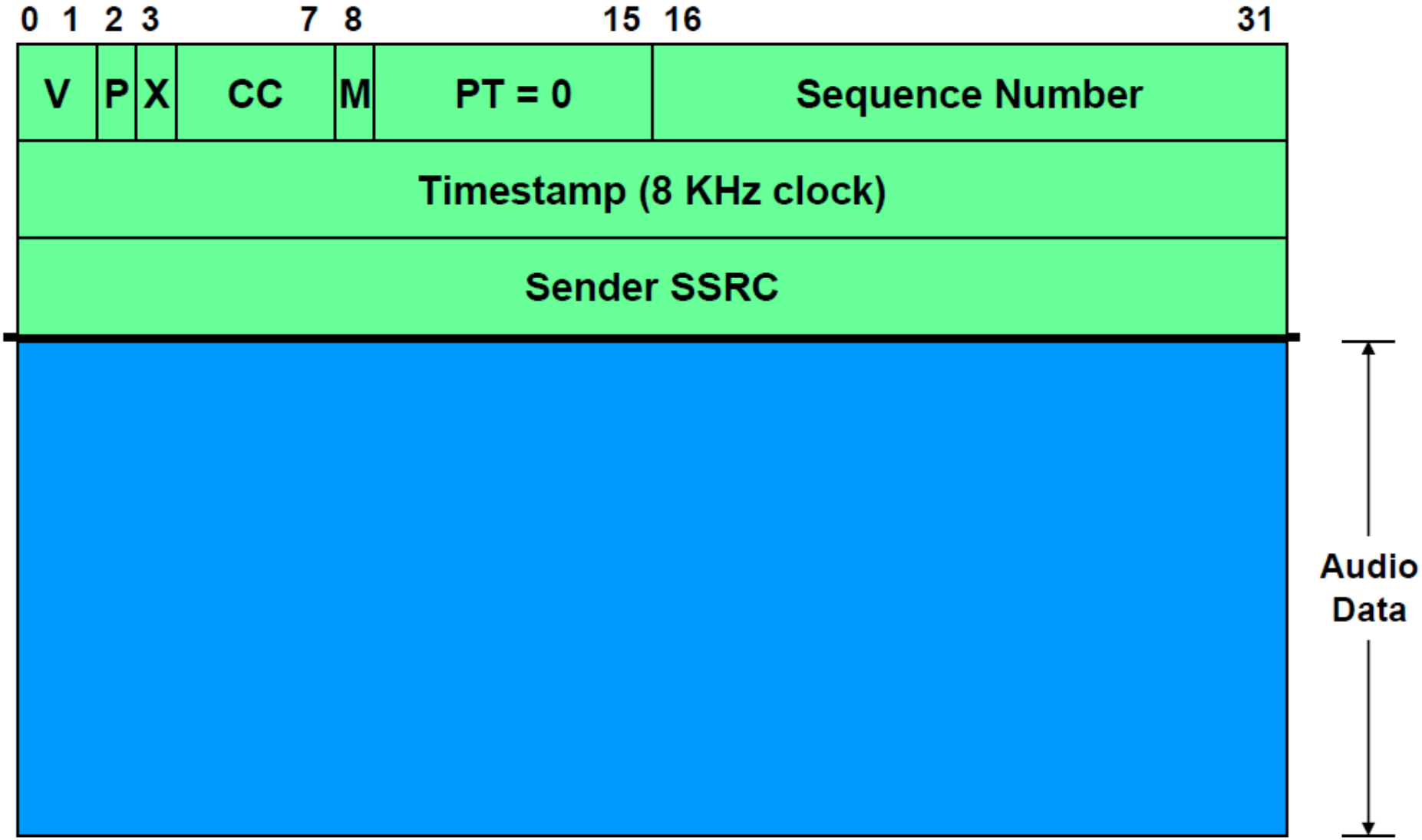
- Example 2: An audio & video conference application
  - Each medium is transmitted in a separate RTP session
  - Provides flexibility for participants to receive one stream only
  - RTCP provides cross-media clock mapping information
    - Sessions from the same participant associated by CNAME
    - Once media timestamps are related to real-time, inter-stream synchronization can be performed by adjusting play-out delays



Media Packetization Schemes (1)

- Packetization schemes are designed to meet needs of different types of media streams. Scheme used in an RTP session is agreed upon by senders & receivers through non-RTP means.
- General principle
- Payload specific additional header (if needed)
- Followed by media data
  - Packetized and formatted in a well-defined way, trivial ones as within RFC 3551
  - RFC 2029, 2032, 2035, 2038, 2190, 2198, 2250, 2343, 2429, 2431, 2435, 2658, 2733, 2793, 2833, 2862, and many further ones
  - Guidelines for writing packet formats: RFC 2736
- Functionality
  - Enable transmission across a packet network
  - Allow for semantics-based fragmentation
  - Provide additional information to simplify processing and decoding at the recipient
  - Maximize possibility of independent decoding of individual packets

Audio over RTP: PCM

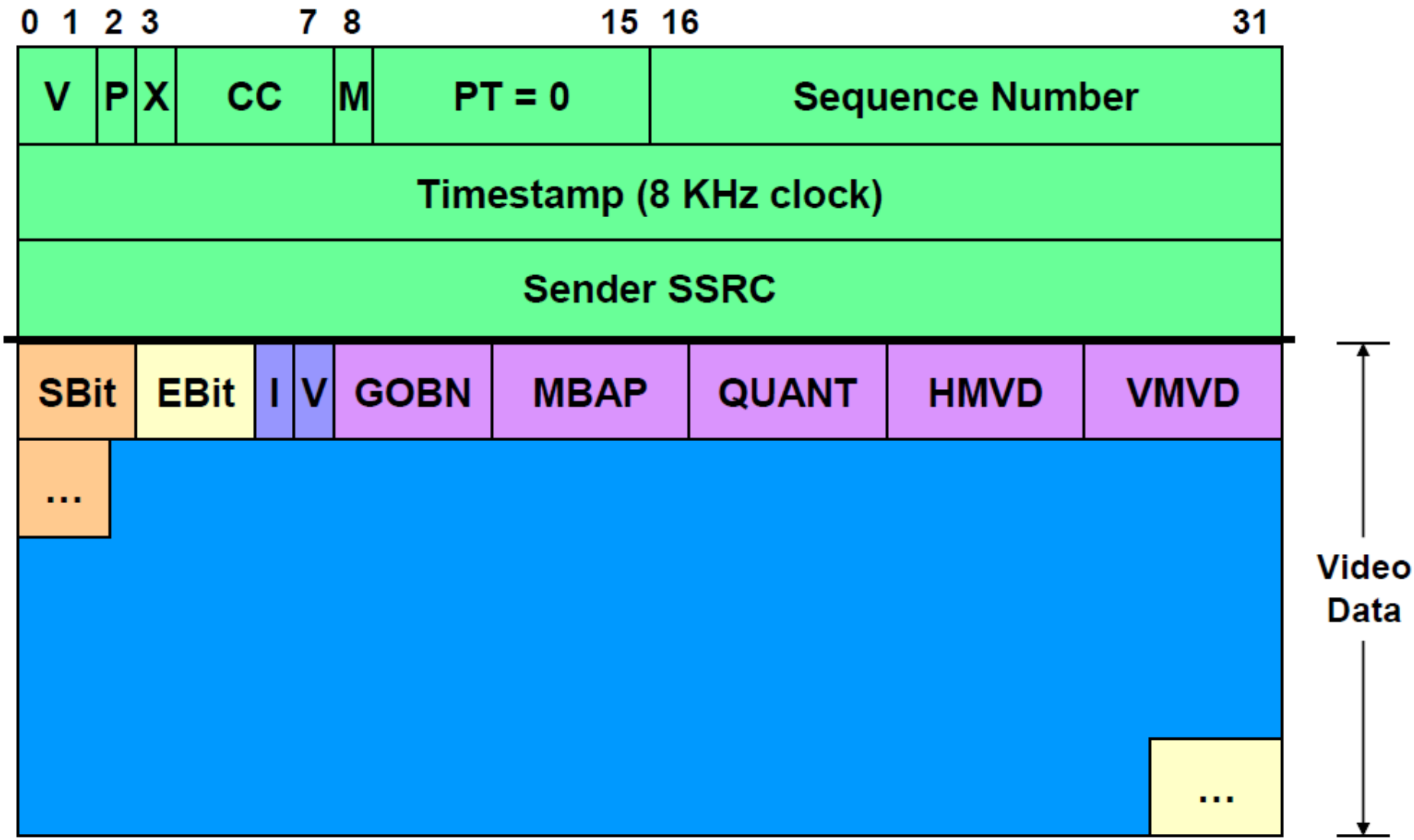


Video over RTP: H.261 (1)

- Additional payload-specific header preceeds payload
- To avoid expensive bit shifting operations
  - Indicate number of invalid bits in first (SBit) and last (EBit) octet of payload
- Indicate Intra encoding (I bit)
- Indicate the presence of motion vector data (V bit)
- Carry further H.261 header information to enable decoding in the presence of packet losses

- Further mechanisms for video conferencing
- FIR: Full Intra Request
  - Ask sender to send a full intra encoded picture
- NACK: Negative Acknowledgement
  - Indicate specific packet loss to sender

Video over RTP: H.261 (2)

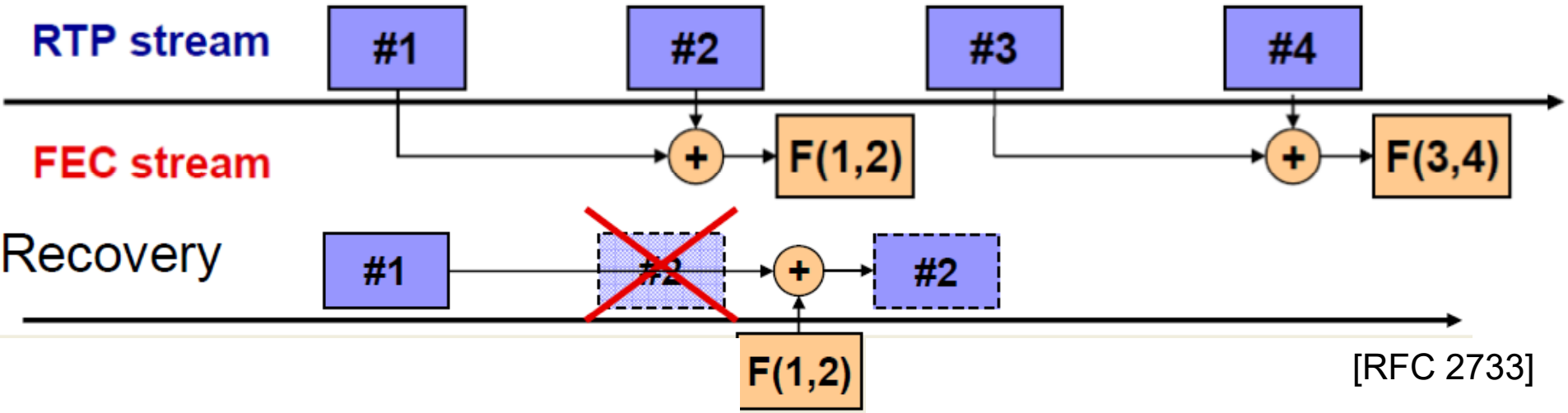


Media Packetization Schemes (2)

- Error-resilience for real-time media
- Input: Observation on packet loss characteristics
- Generic mechanisms (RFC 2354)
  - Retransmissions in special cases only (e.g. with no interactivity!)
    - Interleaving
  - Forward Error Correction (FEC)
    - Media-dependent vs. media-independent
    - Generic FEC: RFC 2733
- Feedback loops for senders
  - Based upon generic and specific RTCP messages
  - Adapt transmission rate, coding scheme, error control

RTP FEC

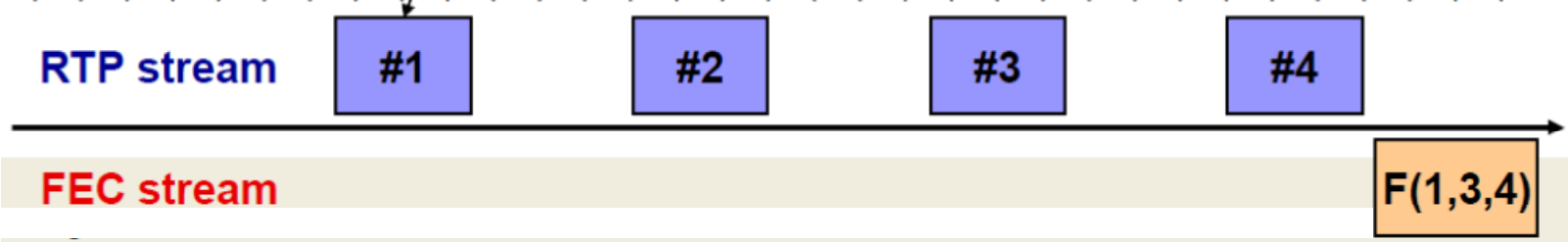
- Forward Error Correction (FEC) scheme for RTP packets
  - Media-independent, flexible FEC that can be enhanced
- Simple XOR-based (parity) FEC
  - P_fec = P1 XOR P2 XOR P3 XOR … XOR Pn
    - Allows reconstruction of any single missing packets of P1, …, Pn, P_fec
- RTP FEC stream transmitted independently of RTP stream
  - Separate transport address (IP address, port number)
  - Different SSRC



[RFC 2733]

RTP Parity FEC Packet Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|    FMT   |       PT      |          length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    SSRC of packet sender                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    SSRC of media source                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             SN base   = 1     |        length recovery        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|E| PT recovery |              mask  = 10110000 00000000 00000000 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         TS recovery                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                                                               :
:         XOR of Payloads indicated by SN Base and mask         :
:                                                               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

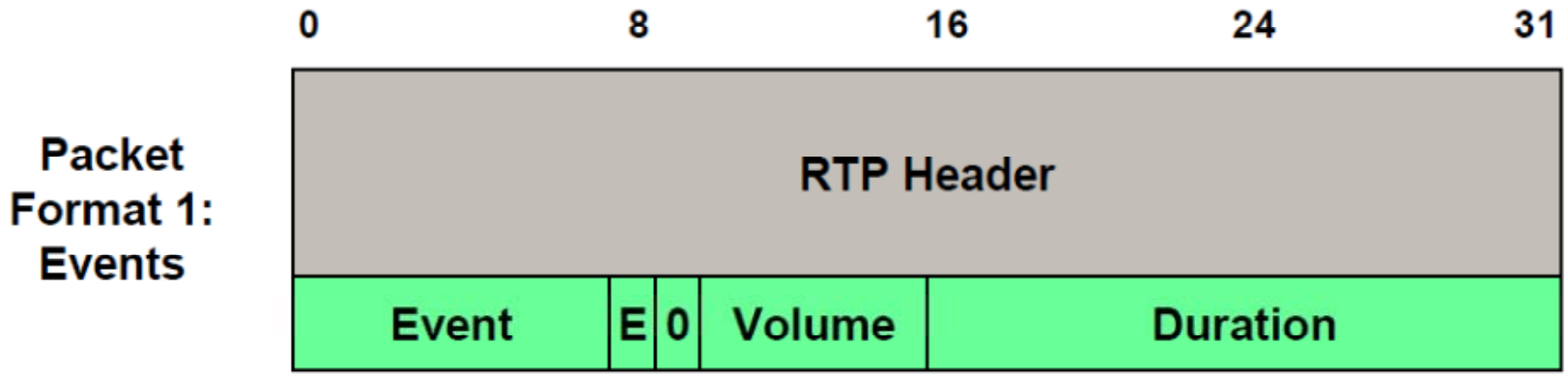


DTMF over RTP (1)

- DTMF digits, telephony tones, and telephony signals
  - Two payload formats
  - 8 kHz clock by default
  - Audio redundancy coding for reliability

DMTF: Discrete Tone Multi Frequency Signaling

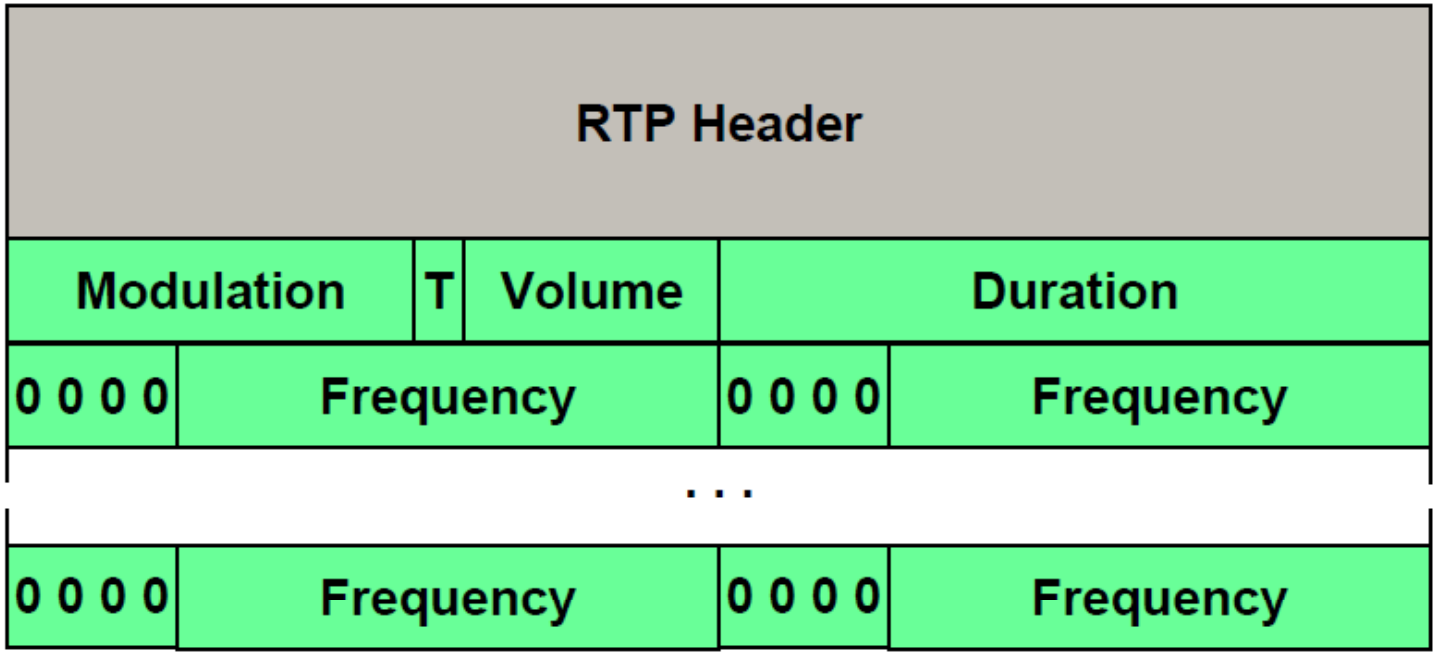
- Format 1: reference pre-defined events
  - 0 - 9 * # A - D (Hook) Flash
  - Modem and fax tones
  - Telephony signals and line events
    - Dial tones, busy, ringing, congestion, on/off hook, …
  - Trunk events specified through identifier (8-bit value), volume, duration



DTMF over RTP (2)

- Format 2: specify tones by frequency
  - One, two, or three frequencies
  - Addition, modulation
  - On/off periods, duration
  - Specified through modulation, n x frequency, volume

Packet Format 2: Tones



Real-time Transport Control Protocol (RTCP)

- RTP allows only for one type of message: payload data

- Other messages in a session control the flow and quality of data and allow for the recipient to send feedback to the source(s)
- RTCP offers five message types
  - RTCP Sender Report (SR) messages are sent be all data senders to provide the relationship between media timestamps and real-time.
  - RTCP Receiver Report (RR) messages are sent by all receivers provide feedback to the senders on reception quality.
  - RTCP Source Description (SDES) messages are sent by all session members
  - RTCP BYE message sent from source to shut-down a session
  - RTCP APP application-specific message defines a packet for an application that wants to use new applications (not defined within RTCP)
- All messages can be combined into compound packets
  - Their rate depends and may vary

[RFC 3550, 3605]

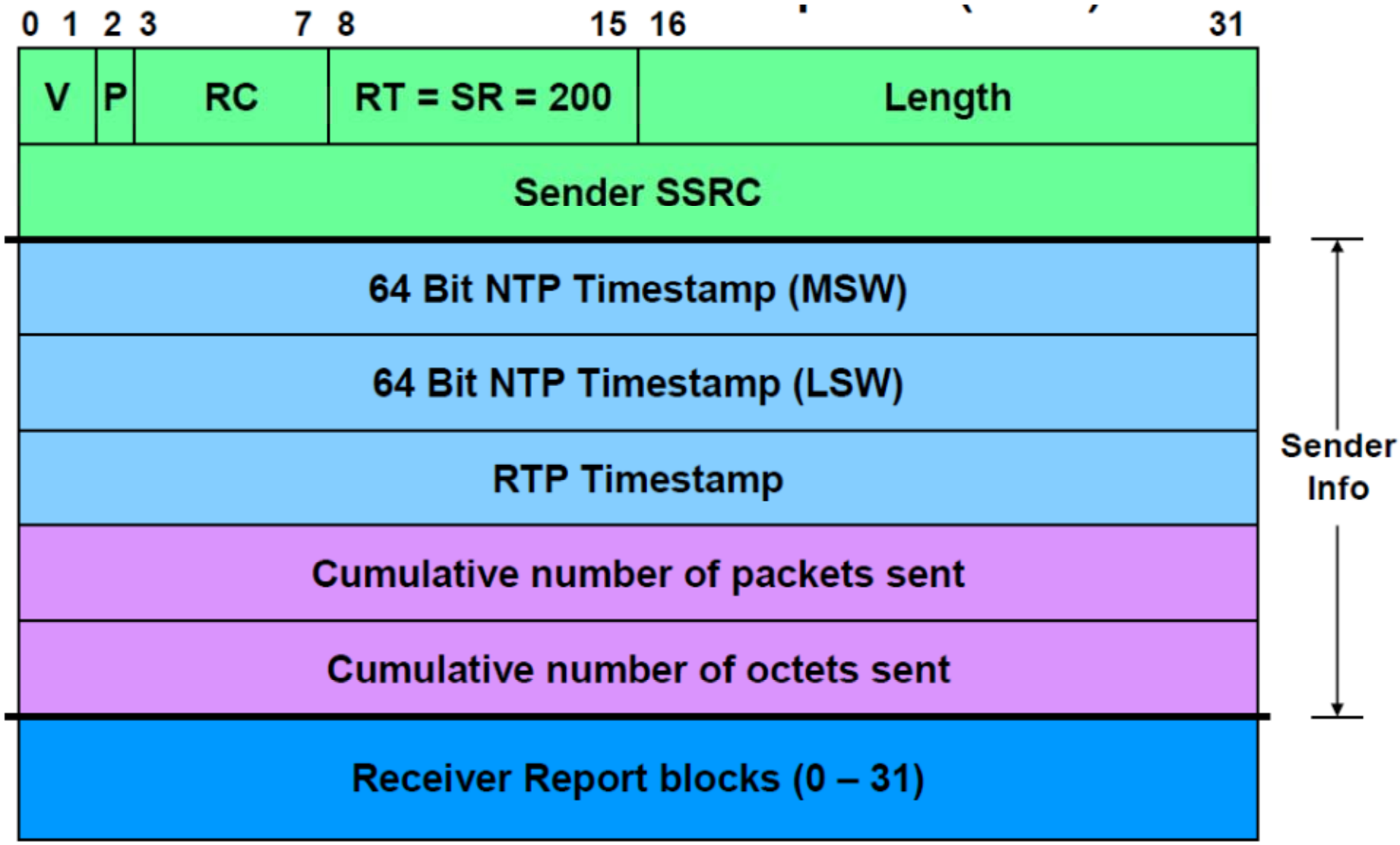
RTCP Sender Report (1)

- The sender report is sent periodically by active senders in a conference to report transmission and reception statistics for all RTP packets sent during that interval.

- Enables cross-media stream synchronization
  - Relate stream-specific RTP time stamp to wall clock time
  - NTP timestamp + RTP timestamp
  - Playout adjustment to be performed by the receivers
- Provide data point for RTT measurement
  - NTP timestamp
- Provide feed forward about data transmitted
  - Transmit sender's packet and byte count
  - Enable receiver to do proper loss calculation
- Include Receiver Reports for the sender as well

NTP: Network Time Protocol
RTT: Round-trip Time

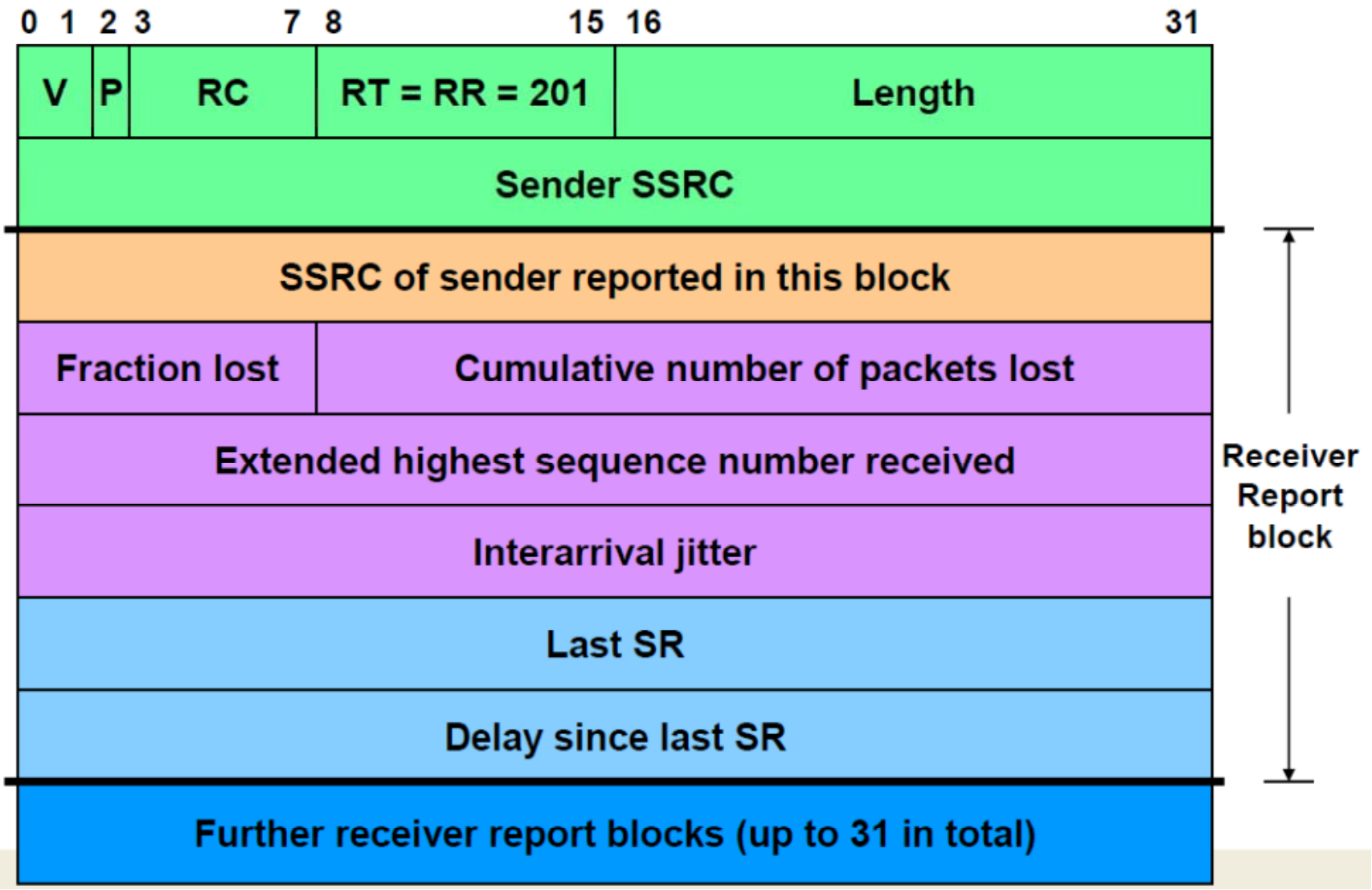
RTCP Sender Report (2)



RTPC Receiver Report (1)

- The receiver report is for passive participants, those that do not send RTP packets. The report informs the sender and other receivers about Quality-of-Service.
- Feedbacks timing for RTT estimation
  - Sender Report (SR) Timestamp
    - Middle 32 bits taken from the last SR's NTP timestamp
  - Delay since last SR
    - Local delay at receiver between receiver SR and sending the RR block
    - Measured in units of 1 / 65556 seconds
- Provides per-sender reception statistics
  - Total number of packets lost and fraction of packets lost (in units of 1/256)
  - Highest sequence number received so far
  - Jitter of received packets
- Enables adaptive sender behavior
  - Adjust codecs, codec parameters, transmission rate and the like

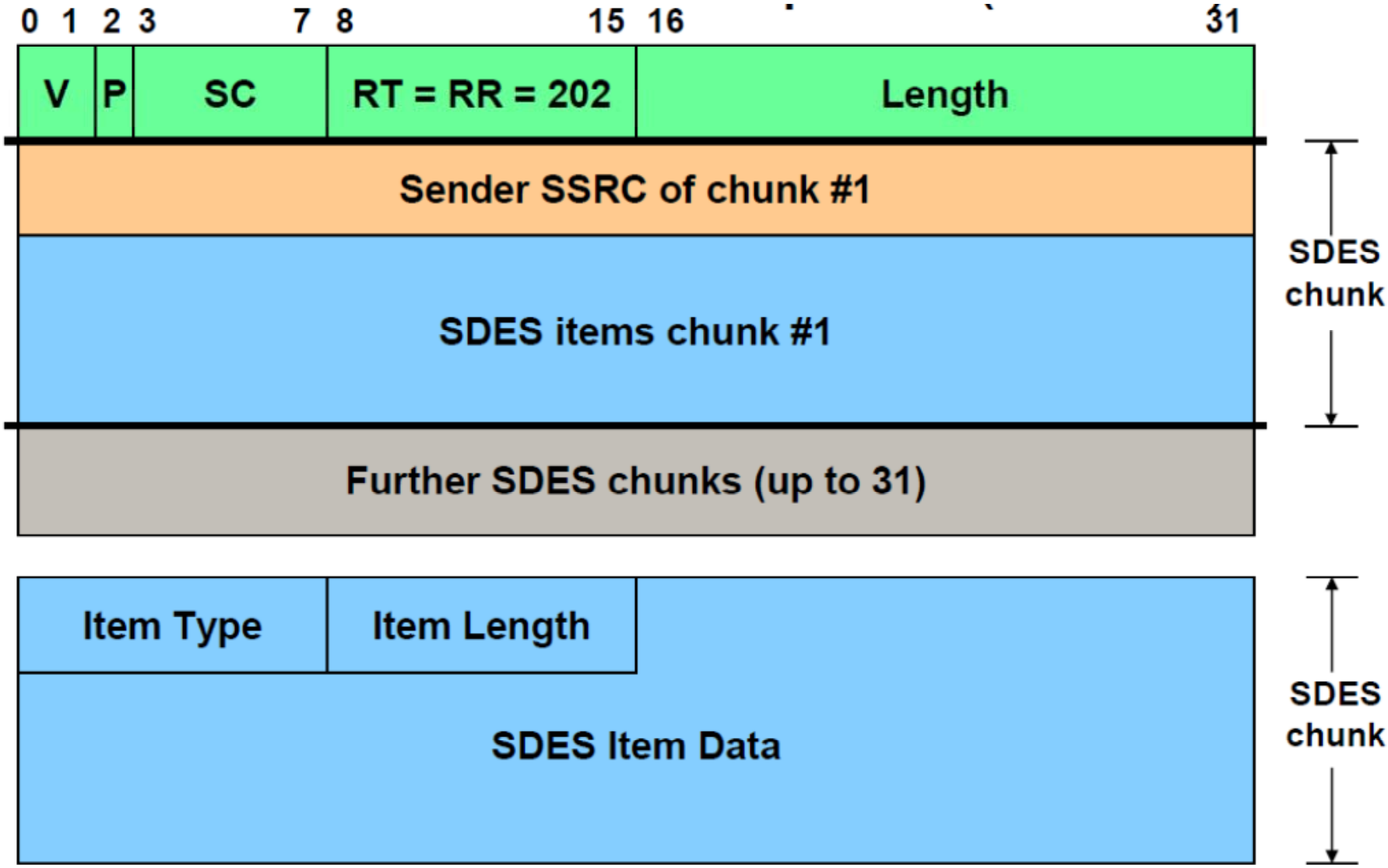
RTPC Receiver Report (2)



RTCP Source Description (1)

- The source periodically sends a Source DEScription (SDES) message to give additional information about itself. This information can be the name, e-mail address, telephone number, and address of the owner or controller of the source.
- Persistent Identification of an endpoint: Canonical Name (CNAME)
  - CNAME: globally unique identifier (id@host) – Mandatory!
  - Binding across RTP sessions
  - Identification across changes in SSRC in an RTP session (name, email, location, ...)
- Providing additional information about an endpoint
  - NAME: Name of user (or system)
  - EMAIL: mailto: address
  - PHONE: phone number
  - LOC: location (no format defined)
  - TOOL: software client in use
  - NOTE: brief to other participants (e.g. “on the phone”)
  - PRIV: private extensions

RTCP Source Description (2)

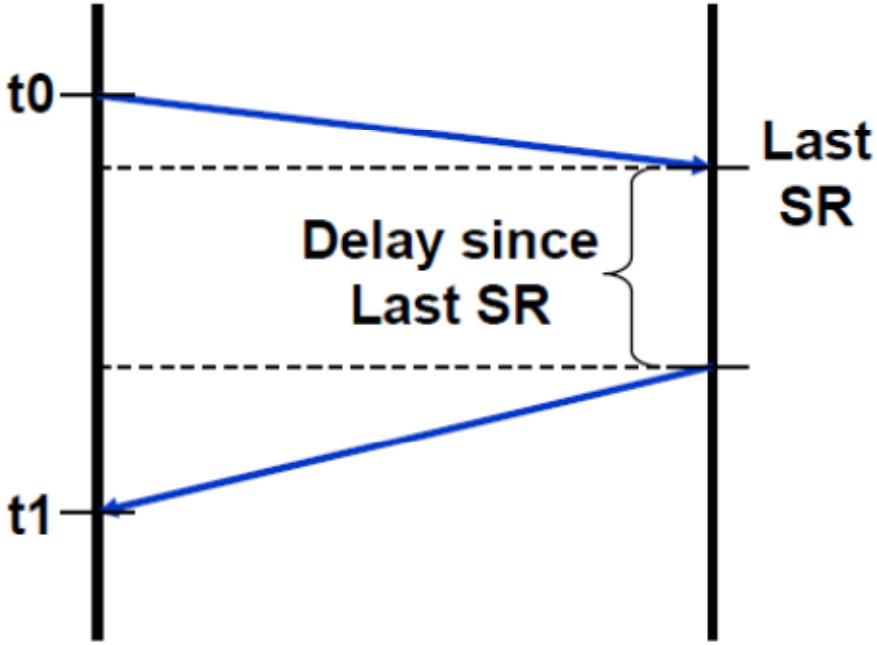


RTCP Statistics Collection (Sender)

- Round-Trip Time (sender only)
  - Derived from time stamps in RR
  - Simple formula

$$RTT = t1 - t0 - DSL\_SR$$

  - RTT may be asymmetric!



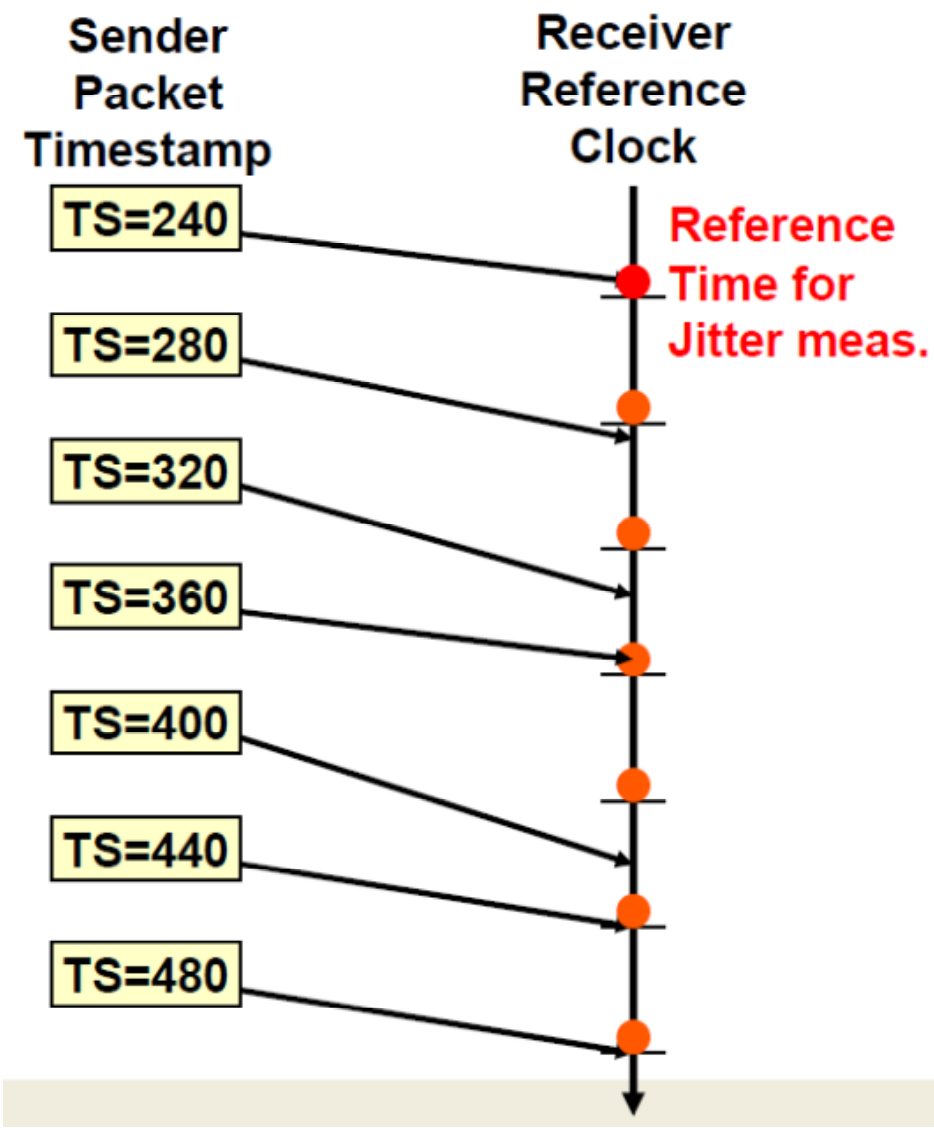
- Byte count
- Packet count

RTCP Statistics Collection (Receiver)

- Packet Loss
  - Calculated from gaps in sequence number space
    - First (lowest packet sequence number) received
  - Expected number of packets = current – lowest
  - Received number of packets
    - Count duplicates, out-of-order, and late packets as received!
  - Absolute # of lost packets = expected – received
    - May be negative!
  - Fraction of lost packet
    - Loss since last SR or RR packet was sent
  - Loss of all packets not detected!
- Extended highest sequence number received (32 bits)
- Time of last SR reception
- Jitter

RTCP Interarrival Jitter Estimation

- Receiver measures in time units of the media clock
- Relates it to local real-time clock
- Initialized through first packet received
- Derives expected reception time
- Calculates deviation D upon packet reception
- Sampled for each packet
- Jitter derived for each peer of successively received packets
  - Ordering is not relevant
- Weighing function:
  $J = J' + (D - J') / 16$



RTCP Payload Type Overview (1)

- RFC 3551 Collection of simple packetization formats (formerly RFC 1890)
- RFC 2029 Sun CellB Video encoding
- RFC 2032 H.261 video
- RFC 2435 JPEG video (was RFC 2035)
- RFC 2250 MPEG-1/MPEG-2 video (was RFC 2038)
- RFC 2190 H.263 video (historic)
- RFC 2343 Bundled MPEG
- RFC 2429 H.263+ video & video redundancy support
- RFC 2431 BT.656 video
- RFC 2658 PureVoice audio
- RFC 2793, 4103 Text conversation
- RFC 2833 DTMF, telephony tones, and telephony signals
- RFC 2862 Real-time Pointers
- RFC 3016 MPEG-4 Audio/visual streams
- RFC 3047 G.722.1 audio
- RFC 3119 Loss-tolerant format for MP3

RTCP Payload Type Overview (2)

- RFC 3189 DV video
- RFC 3190 12-bit DAT and 20-/24-bit linear audio
- RFC 3267 Adaptive Multirate (AMR) audio
- RFC 3389 Comfort noise
- RFC 3497 SMPTE 292M video
- RFC 3557 ETSI Distributed speech recognition (ES 201 108)
- RFC 3558 Enhanced variable rate codecs and selectable mode vocoders
- RFC 3640 MPEG-4 elementary streams
- RFC 3952 Low Bit Rate Codec (iLBC) Speech
- RFC 3984 H.264 Video
- RFC 4040 64 kbit/s Transparent Call
- RFC 4060 Distributed speech recognition encoding (ES 202 050/211/212)
- RFC 4175 Uncompressed Video
- RFC 4184 AC-3 Audio
- Many more to come … revisions to many of the above

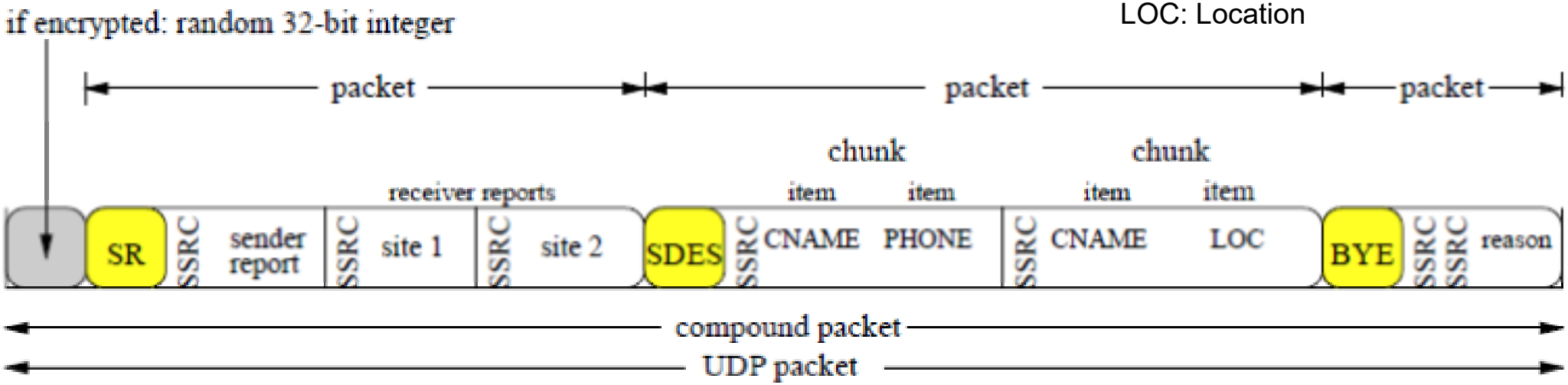
RTP Specs and Recommendations

- RFC 3550 Base specification (formerly RFC 1889)
- RFC 3551 RTP Profile for Audio and Video Conference with minimal control
- RFC 2198 Redundant (Audio) coding
- RFC 2508 RTP header compression for low-speed links
- RFC 2733 Generic FEC
- RFC 2736 Guidelines for writers of RTP payload specifications
- RFC 2762 Group membership sampling (“timer reconsideration”)
- RFC 3095 Robust header compression for RTP (among others)
- RFC 3096 Requirements for robust IP/UDP/RTP header compression
- RFC 3158 RTP testing strategies
- RFC 3242 Link-layer assisted profile for IP/UDP/RTP header compression
- RFC 3243 Requirements & assumptions for 0-byte IP/UDP/RTP header comp.
- RFC 3409 Lower-layer guidelines for robust IP/UDP/RTP header compression
- RFC 3545 Enhanced compressed RTP (CRTP) for high-delay links
- RFC 3555 MIME registrations of RTP payloads
- RFC 3611 RTCP XR extension
- RFC 3711 Secure RTP (SRTP)

Security Support in RTP

- Support of confidentiality through encryption
  - Use profile to specify security services and algorithms
  - DES in CBC mode as the default encryption algorithm

DES: Data Encryption Standard
CBC: Chain Block Cypher
LOC: Location



- May define additional payload types for encrypted encodings
  - Expect support for authentication and message integrity from lower layers.

RTP Implementations

tool	who	media	RSVP	adaptive
NeVoT	GMD Fokus	audio	yes	not yet
vic	LBNL	video	no	no
vat	LBNL	audio	no	no
rat	UCL	audio	no	no
Rendezvous	INRIA	A/V	no	yes
NetMeeting	Microsoft	A/V	no	no
IP/TV	Cisco	A/V	no	no
RM G2	Real	A/V	no	yes

http://www.cs.columbia.edu/~hgs/rtp/

Part 4: Multimedia Protocol Components

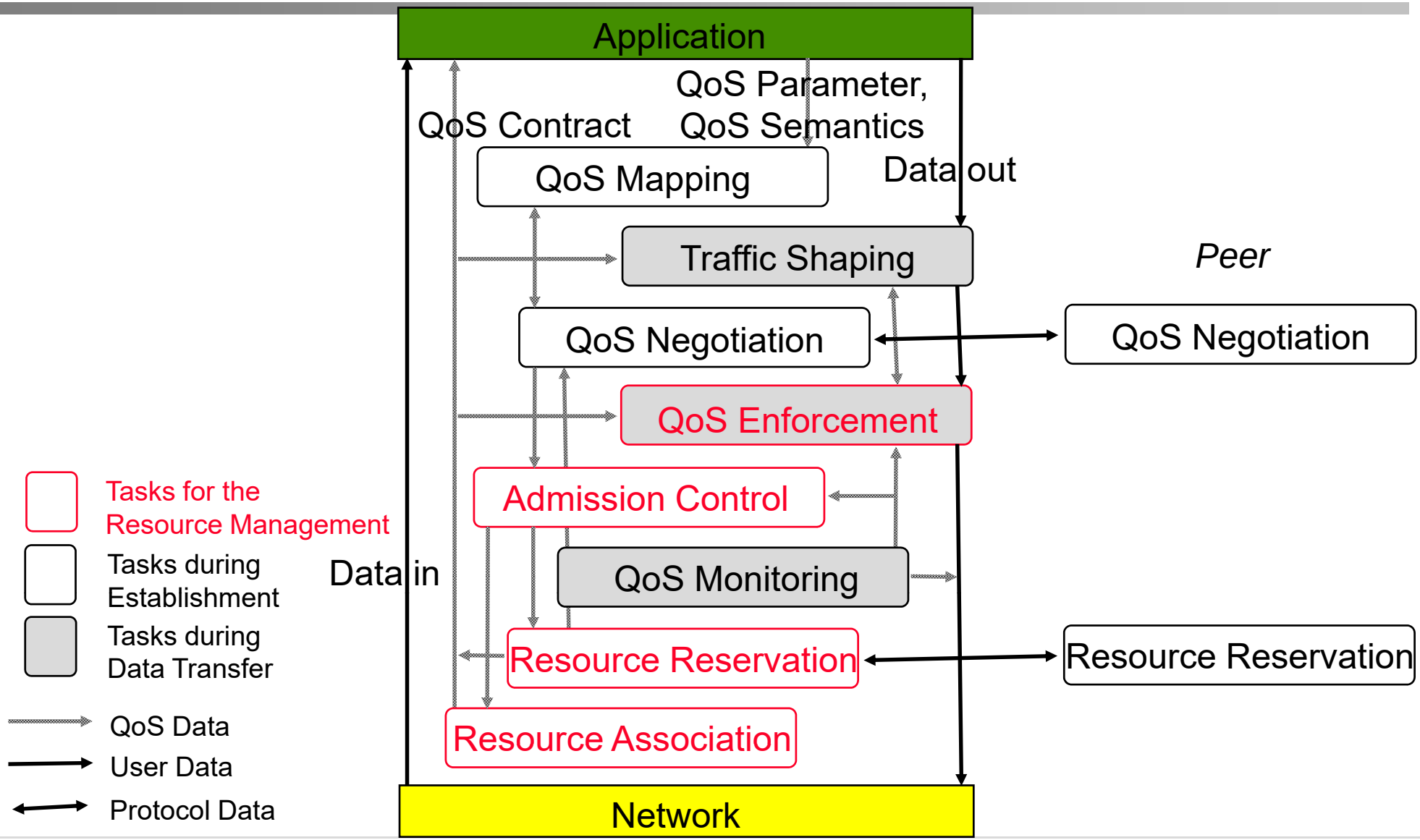
- Resource Reservation Protocol (RSVP)
  - Basic Concepts and Alternatives
  - Resource Management
  - RSVP
  - Comparison

Distinction of Resources

- Two orthogonal distinctions are useful:
  - Active and passive resources,
  - End-system (local) and network resources (global).
- Active resources are schedulable, *e.g.*, CPU, network.
- Passive resources do not process data and are not schedulable, *e.g.*, memory buffers.

- End-system resources have to be identified for a certain flow of data and associated to this flow.
  - Resource Association by local mechanisms.
- Network resources have to be identified for a certain flow of data and reserved for this flow.
  - Resource Reservation by peer-to-peer protocols.

Resource Management (1)



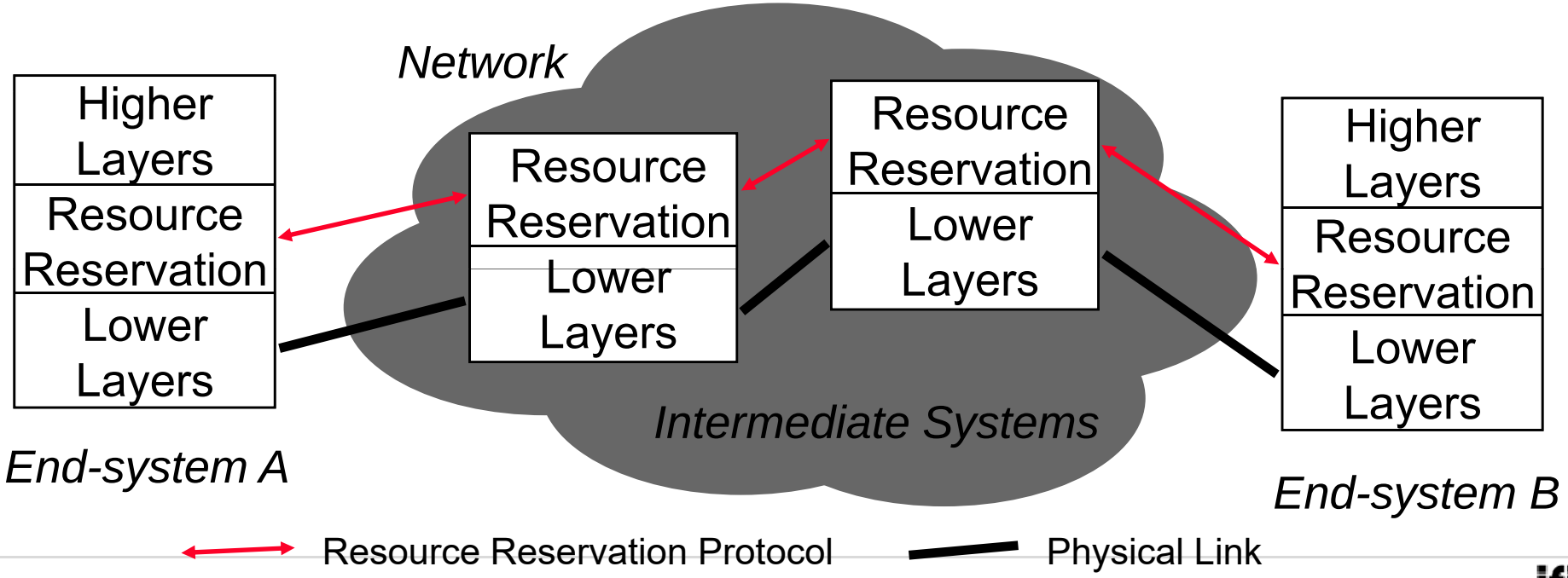
Reservation Strategies

- Prerequisites:
  - Applications have to predict their traffic shapes.
    - Simple for continuous flow of data (cf. leaky bucket).
    - Difficult for real-time flow of data (bursty behavior).

- Pessimistic approach:
  - Handling of worst-case situations while assuming that the application does not exceed the agreed upon QoS contract.
    - Avoiding conflicts between applications.
    - Poor utilization of resources, but deterministic guarantees.
- Optimistic approach:
  - Reservations are done for the average case + x% optionally.
    - Conflicts are possible, violation of QoS contracts for a time period.
    - Better resource utilization, but statistical guarantees.

Resource Reservation Protocols

- Resource reservation encompasses a link-by-link communication between participating end-systems and involved intermediate systems.
  - Propagation of considered QoS to end- and intermediate systems and distribution and collections of reservations.

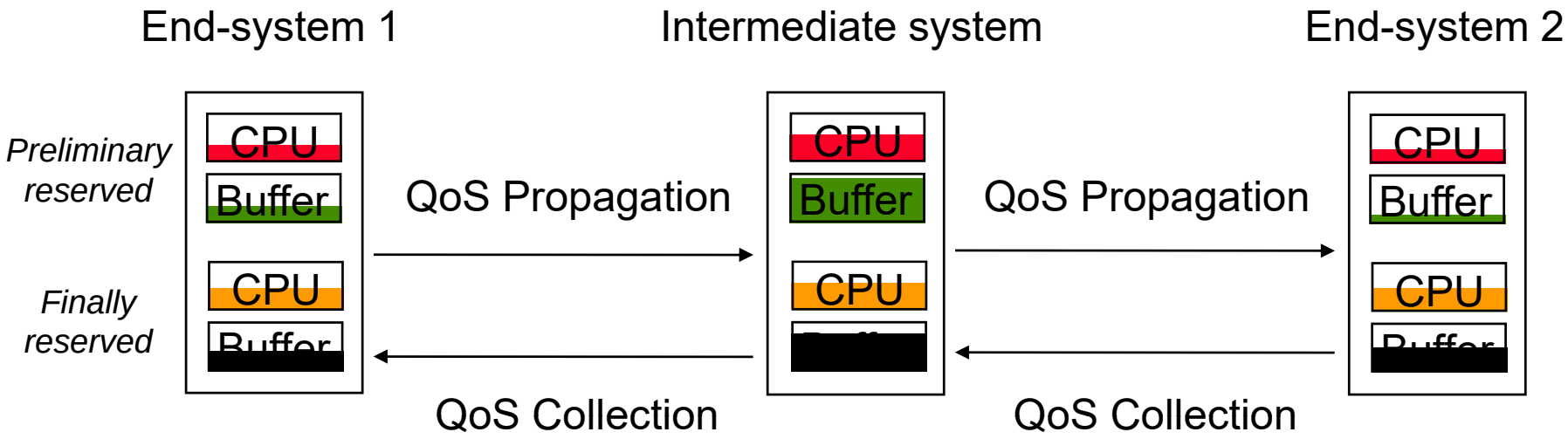


Example

End-system 1:
maxPDUsize = 2048 byte
Throughput = 2 Mbit/s
(≈ 2 PDU/100 ms)

Intermediate system:
maxPDUsize = 1024 byte
Throughput = 2 Mbit/s
(≈ 4 PDU/100 ms)

End-system 2:
maxPDUsize = 2048 byte
Throughput = 1 Mbit/s
(≈ 1 PDU/100 ms)



End-system 1:
CPU-Util = 10%
Buffer = 4 PDU
maxPDUsize = 1024 byte
Throughput = 1 Mbit/s

Intermediate system:
CPU-Util = 5%
Buffer = 10 PDU
maxPDUsize = 1024 byte
Throughput = 1 Mbit/s

End-system 2:
CPU-Util = 25%
Buffer = 4 PDU
maxPDUsize = 1024 byte
Throughput = 1 Mbit/s

Resource Reservation Protocol (RSVP)

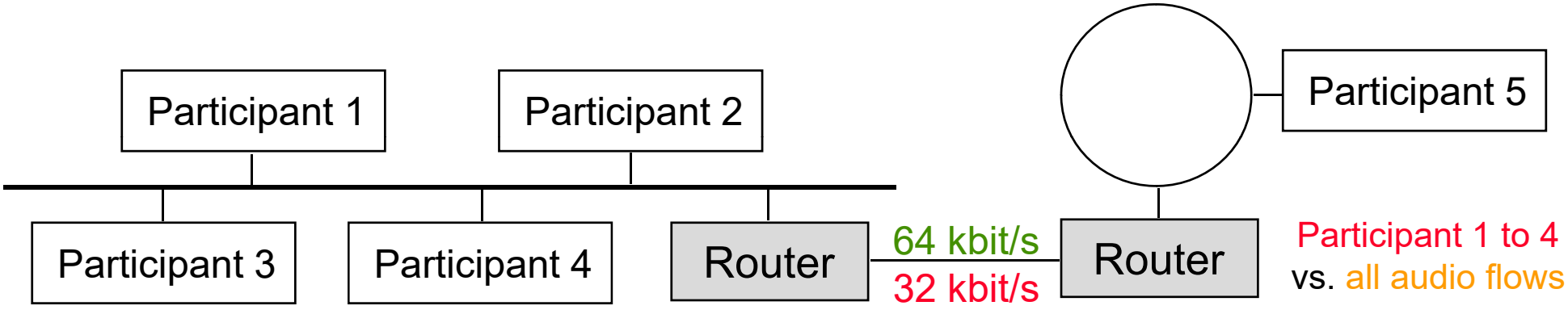
- RSVP allows for the reservation of resources.
  - First specifications Xerox Parc and MIT (1993), IETF (1994).
  - Its basic implementation is based on the Internet.
  - User data transfer is processed by IP.
  - RSVP reserves resources along a path, that has been determined by a routing protocol.
  - IP datagrams follow this route (IP is connectionless).
- RSVP supports receiver-initiated, uni-directional, multicast connections.
- Separation of reservation from packet filtering.
  - FlowSpec and FilterSpec.
- Reduction of control protocol overhead.

[RFC 2205]

RSVP – Reservation Styles

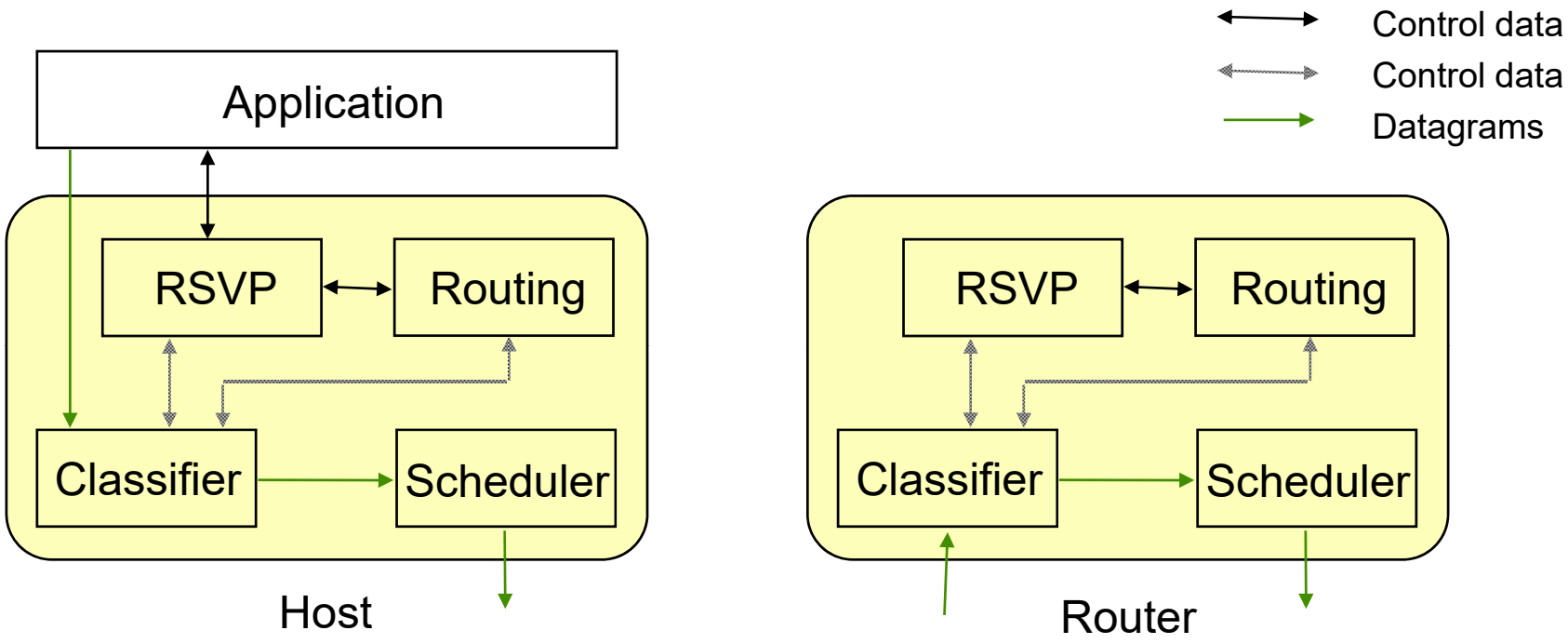
- Flexibility in reservation handling:
  - Fixed Filters: Resources are reserved for a particular flow.
  - Shared Explicit: Resources are reserved for several specifically dedicated flows at once, allowing that these may share these resources.
  - Wildcard Filter: Reserve resources for a general type of flow, without specifying the flow directly. Sharing of resources is allowed always for this type of flows.

- Example: Audio-Conference



RSVP – Architecture

- Required methods for sufficient use of RSVP include:
  - Admission control per node (end and intermediate system).
  - Classifier to determine datagrams using reservations.
  - Scheduler to assign resources for sending datagrams.



RSVP – Messages

- Resources are reserved for flows:
  - Unidirectional flows identified by destination IP address, and, optionally, by a destination port. Refinements incorporate the flow label field.
  - FlowSpecs are passed along the route.
- Reservation messages are sent “backwards”.
  - Receiver-oriented reservation mode.
  - Application-oriented announcements of new flows.
- Path messages are used to identify the route.
  - Sender-driven selection of routes to be used for the flow.
  - They identify the flow and announce possible reservation expectations for intermediate systems.

RSVP – Message Format (1)

- IP-encapsulated RSVP message with IP Next Header value 46.
- Version 2, Flags unused, RSVP Type (path, reserva-tion, 2 error indication, 2 tear-down)

Byte				
1 - 4	Vers	Flags	RSVP Type	Checksum
5 - 8	Message Length			Reserved
9 - 12	Message Identifier			
13 - 16	Reserved	More Fragments	Fragment Offset	
17 - n	Body of RSVP Message (RSVP Object)			

bit 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8

RSVP – Message Format (2)

- RSVP object format:

Class	Type
– SESSION	IPv6 session (destination of flow).
– RSVP_HOP	IPv6 previous or next hop address.
– TIME_VALUE	Frequency of path or reservation refreshes.
– STYLE	Reservation style.
– FLOWSPEC	Requiring controlled delay.
	Requiring predictive QoS.
	Requiring guaranteed QoS.

Byte			
1 - 4	Object Length	Class No.	Class Type
5 - m	Object Content		

bit 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8

RSVP – PATH Message Format

Byte							
1 - 4	2	0	Type: 1	Checksum			RSVP plus Object
5 - 8	Message Length: 76			0			
9 - 12	Message Identifier: 0x1234567						
13 - 16	0		Fragment Offset: 0				
17 - 20	Session Object Length: 16			Class: 1	Type: 2		SESSION
21 - 28	Destination Address						
29 - 32	0		Flags	Destination Port			
33 - 36	Hop Object Length: 16			Class: 3	Type: 2		RSVP_ HOP
37 - 44	Last Hop Address						
45 - 48	Logical Interface Handle for Last Hop						
49 - 52	Time Object Length: 12			Class: 5	Type: 1		TIME_ VALUE
53 - 56	Refresh Period (milliseconds)						
57 - 60	Maximum Refresh Period (milliseconds)						
61 - 64	Sender Object Length: 16			Class: 11	Type: 3		SENDER_ TEMPLATE
65 - 72	Sender´s Address						
73 - 76	0		Flow Label Sender will Use				

bit 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8

RSVP – RESERVE Message Format

Byte					
1 - 4	2	0	Type: 1	Checksum	
5 - 8	Message Length: 106+n			0	
9 - 12	Message Identifier: 0x1234567				
13 - 16	0		Fragment Offset: 0		
17 - 76	SESSION and RSVP_HOP and TIME_VALUE				
77 - 80	Style Object Length: 8			Class: 8	Type: 1
81 - 84	Style ID: 2		Style Option Vector: 0x00000A		
85 - 88	FlowSpec Object Length: 8++			Class: 3	Type: 2
89 - 92+n	FlowSpec Object				
93+n - 98+n	Filterspec Object Length: 16			Class: 10	Type: 3
99+n - 102+n	Source Address				
103+n - 106+n	0		Flow Label		

RSVP plus Object

(cf. PATH message)

STYLE

FLOWSPEC

FILTER_ SPEC

bit 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8

Comparison of Reservation Protocols

Criteria	CSZ	FP	RCAP	RSVP	SRP	ST-II	UNI 3.1
Interface	yes	yes	yes	no	yes	req.	yes
Protocol	no	yes	yes	yes	yes	yes	yes
Traffic Control	yes	req.	yes	no	yes	no	yes
Resource Mgt.	req.	req.	yes	no	no	no	req.
Connect. Type	1:1 sx	1:1 sx	1:1 sx	m:n sx	1:1 sx	1:1 dx 1:n sx	1:n sx
QoS Parameter							
Throughput	sp, gu	sp, gu	sp, gu	–	sp, gu	–	sp, gu
Delay	sp	sp	sp, gu	–	sp, gu	–	sp, gu
Reliability	–	–	sp, gu	–	sp, gu	–	sp, gu
Jitter	–	–	sp, gu	–	–	–	sp, gu
Guarantees							
Deterministic	yes	yes	yes	no	yes	no	yes
Stochastic	no	no	yes	no	no	no	yes
Best effort	yes	yes	yes	yes	yes	yes	yes
Other	predicted	no	max. Jitter	no	no	no	no

dx: duplex gu: guaranteed no: not existent req.: required sp: specifyable sx: simplex yes: existent

Network Support for Reservation

- Bandwidth is a critical QoS parameter to be reserved.

Network/ Access Modus	Throughput [Mbit/s]	Delay	Delay Jitter	Multicast	Product Availability	Bandwith Reservation
Ethernet	10	random	N/A	⌛	⌛	no
ISO Ethernet	10+6	fixed, < 1ms	0	–	–	N/A
Token Ring	16	config., < 20 ms	max	⌛	⌛	yes/ext.
Fast Ethernet	100	random	N/A	⌛	⌛	yes
FDDI	2*100	TTRT	max	⌛	⌛	yes
FDDI-II	n*6	fixed, < 1ms	0	⌛	⌛	yes
DQDB QA access	b	random	N/A	⌛	⌛	yes
DQDB PA access	b	fixed	0	–	(⌛)	yes
N-ISDN	n*0.064	fixed, < 10 ms	0	–	⌛	yes
B-ISDN / ATM	b	bounded, < 10ms	max	(⌛)	⌛	yes

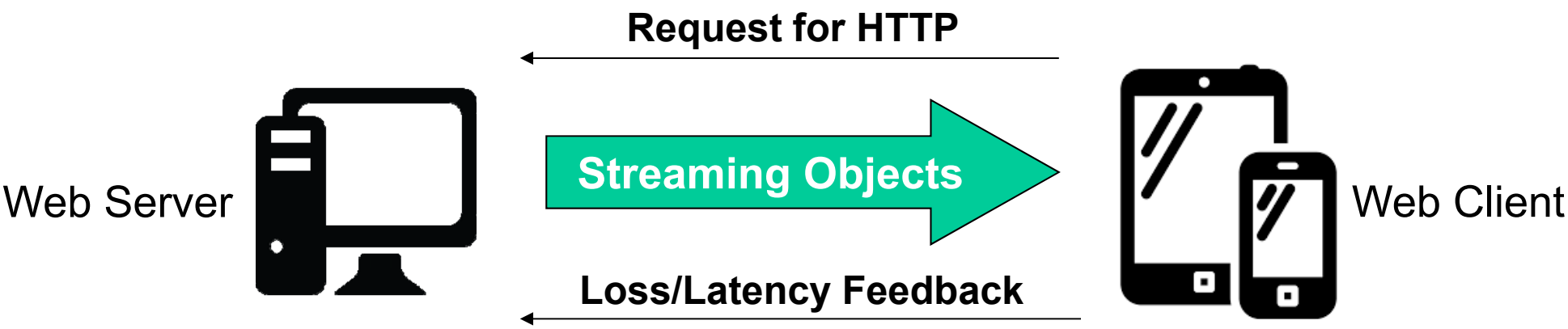
b: one of the following figures: 25, 45, 155, 622; –: no; ⌛: yes; N/A: not available

Part 4: Multimedia Protocol Components

- Stream Control Transmission Protocol (SCTP)
  - Streaming
  - SCTP Intention and Embedding
  - TCP vs. SCTP Comparison
  - Feature List and Details
  - RFCs

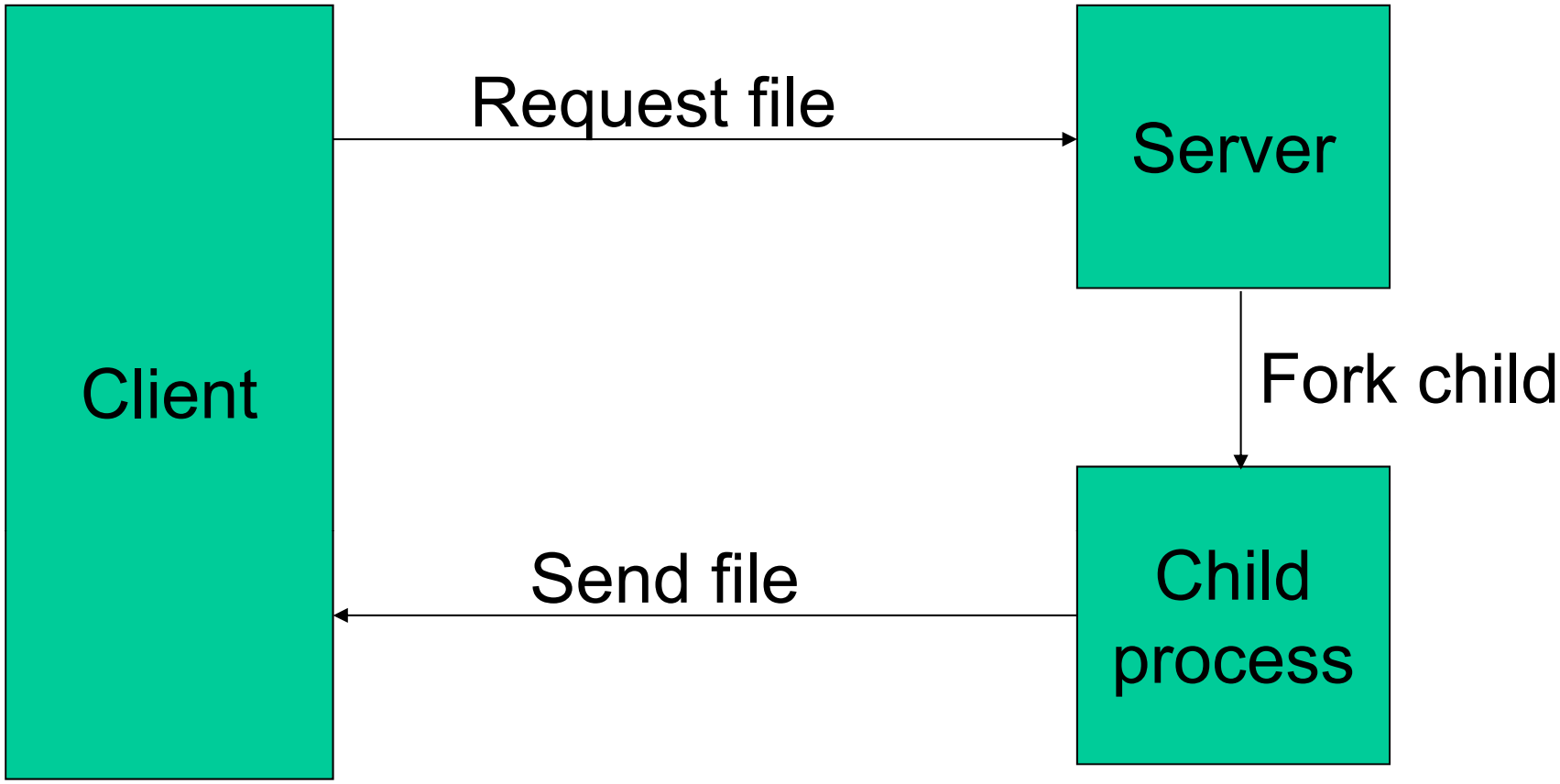
Streaming in the Internet

- Streaming: To transmit audio, video, and other multi-media content over the Internet or other networks
  - Similarities between streaming stored audio/video and streaming live audio/video: Both sensitive to delay and neither can accept retransmissions
- Typically, a local buffer is filled with data, which is used to be played out. Upon depletion the buffer is continuously refilled
  - Web server stores HTML (Hyper Text Mark-up Language) files
  - Streaming media reside on a dedicated streaming server
  - Link is “clicked”, browser reads HTML, and player/plug-in takes over



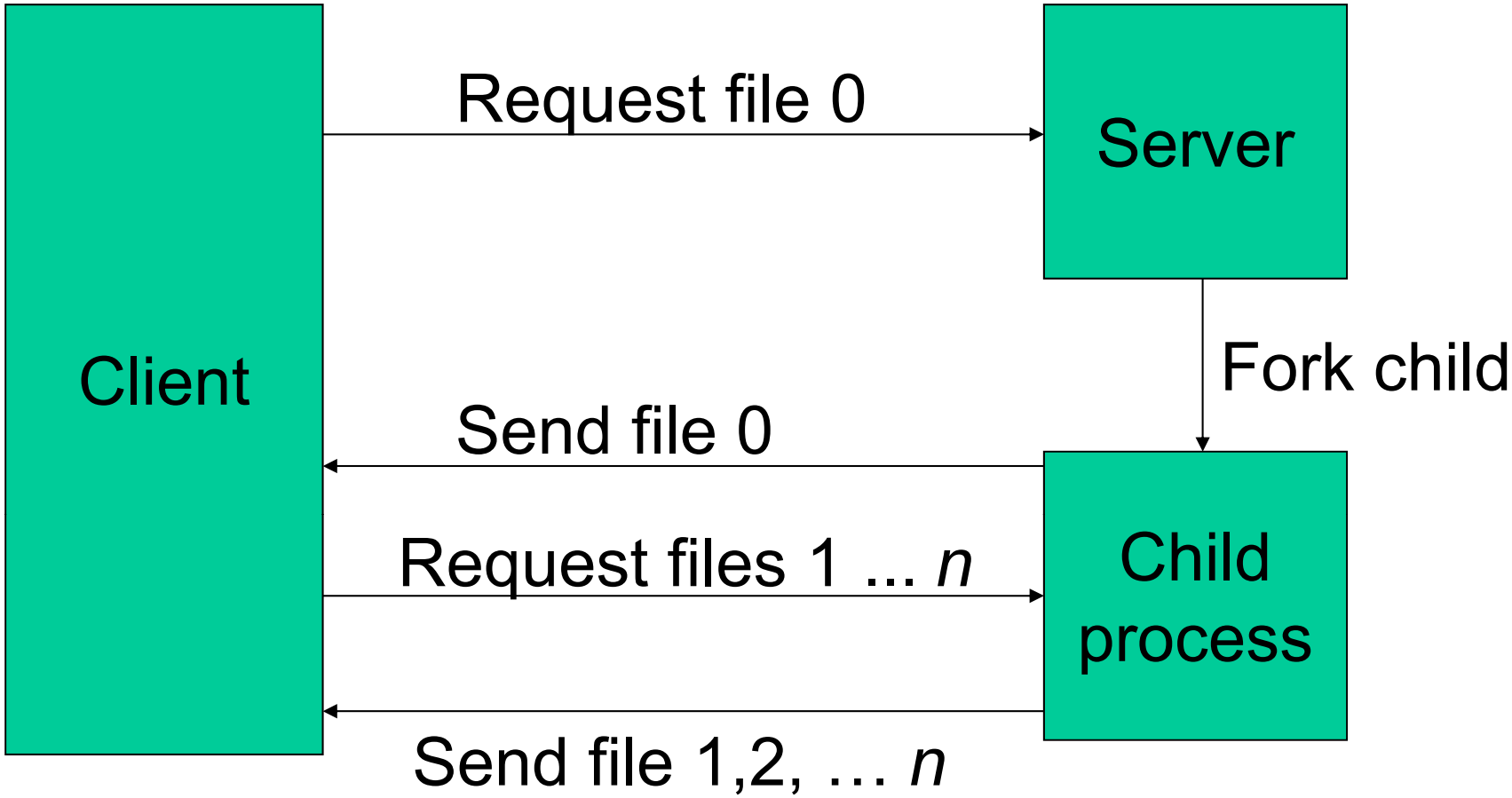
HTTP-based Server Architecture (1)

- Single File Transfer (Both TCP and SCTP are similar)



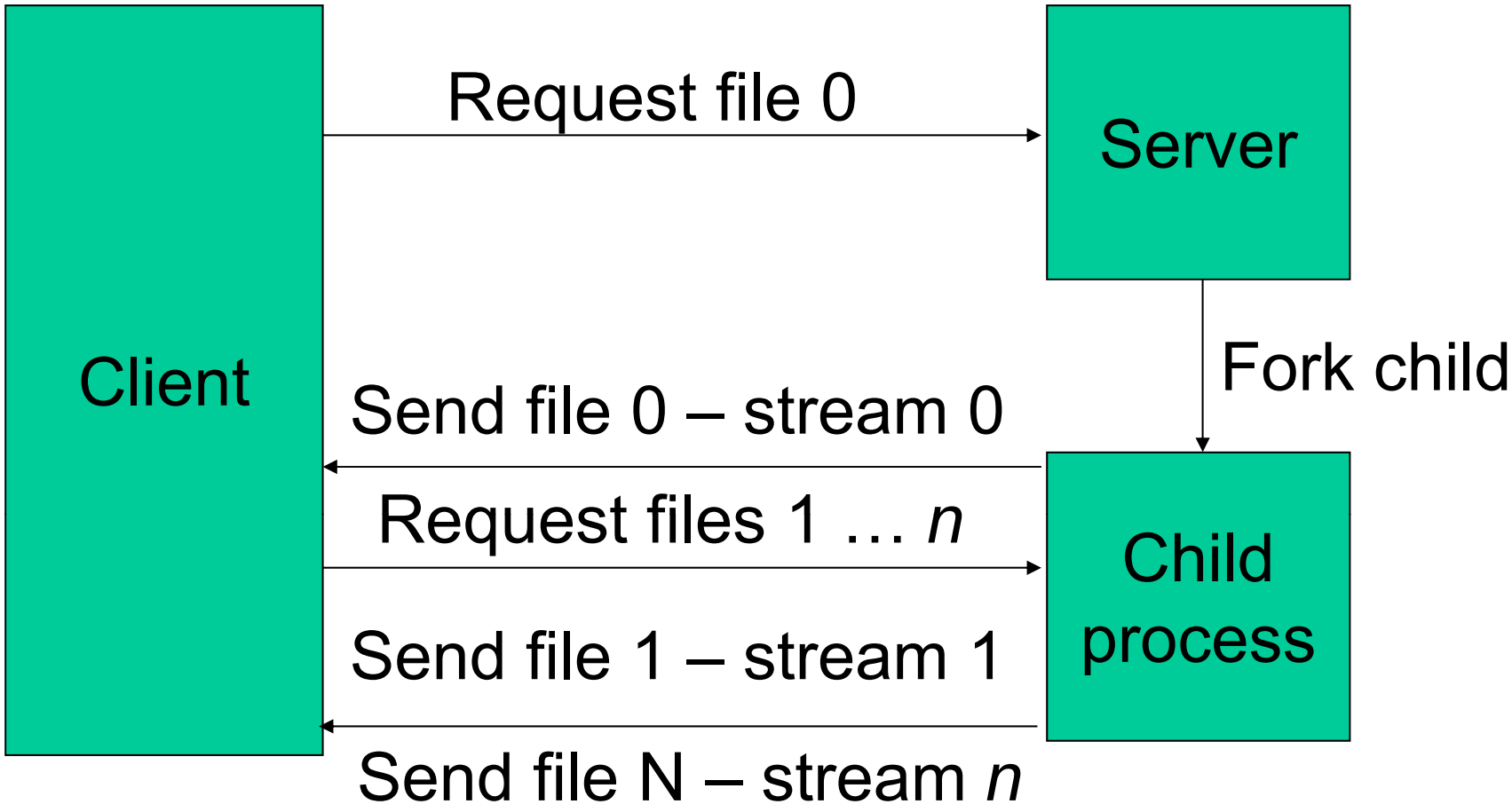
HTTP-based Server Architecture (2)

- Multiple File Transfer (Embedded files) via TCP



HTTP-based Server Architecture (3)

- Multiple Files Transfer (Embedded Files) via SCTP

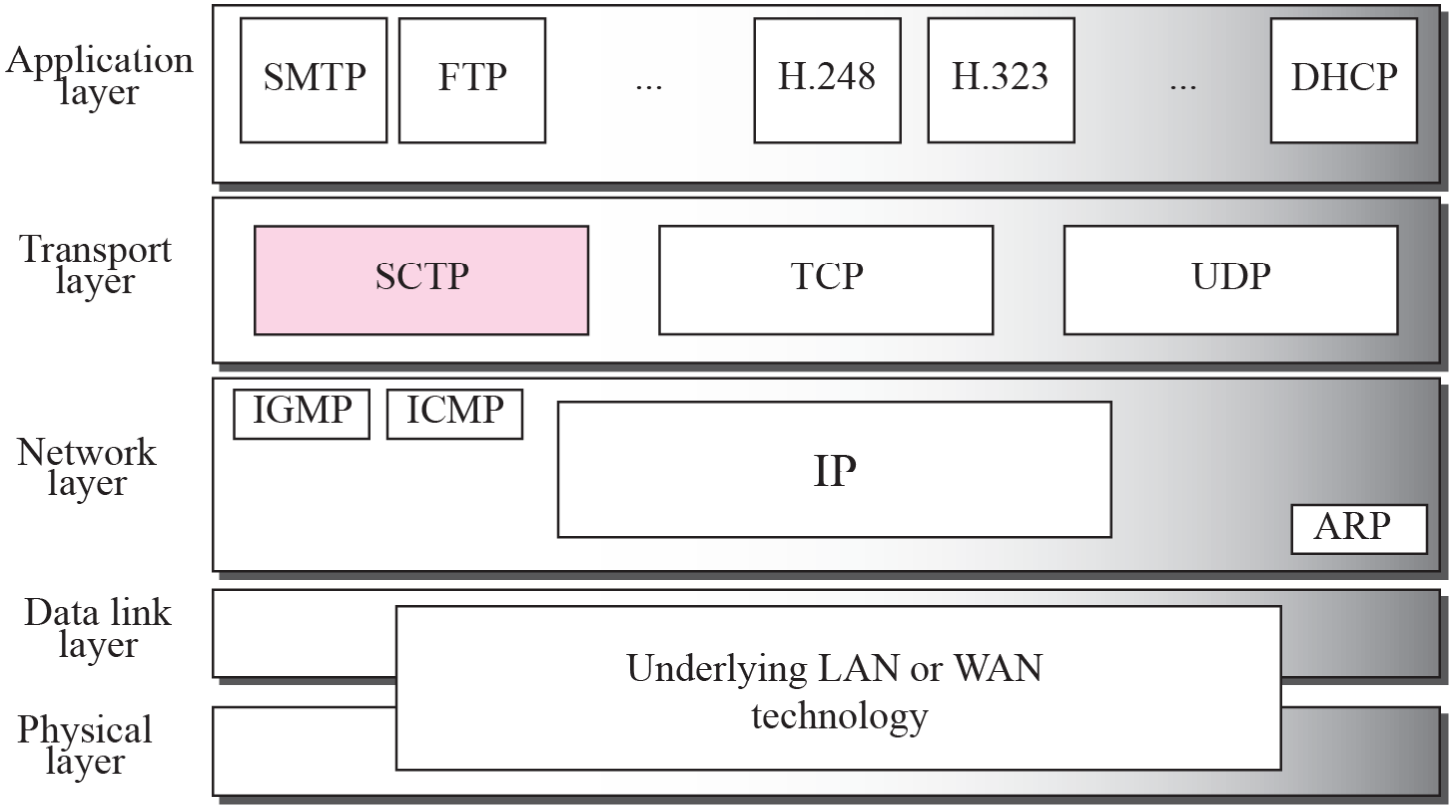


Extracting Streaming Demands

- Many applications need a reliable message delivery
  - Reliably done by delineating a TCP (Transmission Control Protocol) Byte stream
- TCP provides both strict-ordering and reliability
  - Many applications may not need both at the same time
  - HTTP (Hyper Text Transport Protocol) defines one such protocol as it does not require strict-order of delivery, when fetching embedded links.
- Goal
  - Upon the transmission of multiple embedded files a reliable file transfer for each separate file is demanded
  - Partial ordering for packets of each file is essential
  - Total ordering amongst all packets across those multiple files is not needed
- Problem
  - TCP provides more than this
  - TCP shows an unnecessary overhead
- Solution: SCTP (Stream Control Transport Protocol) [RFC 4960]

SCTP Intention and Embedding

- SCTP's original intention
  - To support PSTN (Public Switched Telephone Network) signaling messages over IP (Internet Protocol) Networks
- Stream Control Transmission Protocol is a transport layer protocol



General Transport Layer Protocol Comparison

- UDP: Message-oriented, unreliable
- TCP: Byte-oriented, reliable
- SCTP: Message-oriented, reliable

- UDP: Connection-less
- TCP: Connections
- SCTP: Concept of associations (connection-oriented)
  - Each association can have multiple streams

- UDP: No sequence numbers
- TCP: Byte numbering
- SCTP: Data chunk is numbered using a Transmission Sequence Number (TSN)
  - To distinguish between different streams an SI (Stream Identifier) is used
  - To distinguish between different data chunks belonging to the same stream SSNs (Stream Sequence Number) are used

TCP to SCTP Comparison

- Control information
  - Part of the header
- Data
  - One entity in a TCP segment
- Options
  - Part of the header
- Mandatory parts of header
  - 20 Byte (sequence number in header, SYN and FIN need to consume one sequence number)
- Checksum
  - 16 bit
- Association identifier
  - None

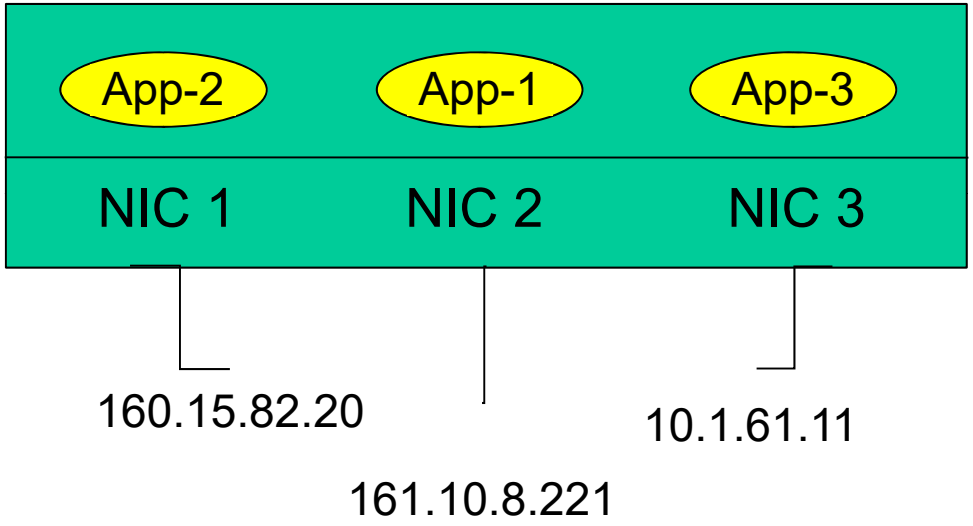
- Control information
  - Several types of control chunks
- Data
  - Several data chunks in a packet
- Options
  - Handled by defining new chunk types
- Mandatory parts of header
  - 12 Byte (TSN in data chunk's header and Acknowledgement number as well as window size part of control chunk
- Checksum
  - 32 bit
- Association identifier
  - Verification tag for multi-homing

SCTP Transport Address and End-point

- Transport protocols define a transport layer header
  - De-multiplexes data coming to a host to the correct application
- TCP and UDP
  - Binding to a “port” forming the de-multiplexing method for data
- SCTP
  - Application typically opens an SCTP socket and binds one address, a set of addresses, or all addresses to that socket
- SCTP also defines the same byte positions in its transport header for the two 16 bit port fields
  - SCTP port and an IP address is the “SCTP Transport Address”
  - IP address in an SCTP Transport Address, MUST be a routeable unicast address
  - I.e. multicast and broadcast addresses are invalid
- SCTP endpoint is logical end of a SCTP transport protocol
  - Communicating party
  - Endpoints may have more than one IP address, but it always has one and only one port number

SCTP End-point Use

- SCTP endpoint is represented as a list of SCTP transport addresses with the same port:
  - Endpoint = [10.1.4.2, 10.1.5.3 : 80]
- An SCTP transport address can only be bound to one single SCTP endpoint
- Example:
  - Application-1 has bound one IP address 161.10.8.221 of the host with port 2223.
  - If a new Application-2 is started, it may bind [160.15.82.20 : 2223] or [10.1.61.11 : 2223] or even [160.15.82.20, 10.1.61.11 : 2223]
  - The new application will ***not*** be able to bind the existing SCTP transport address that Application-1 has bound i.e: [161.10.8.221 : 2223]



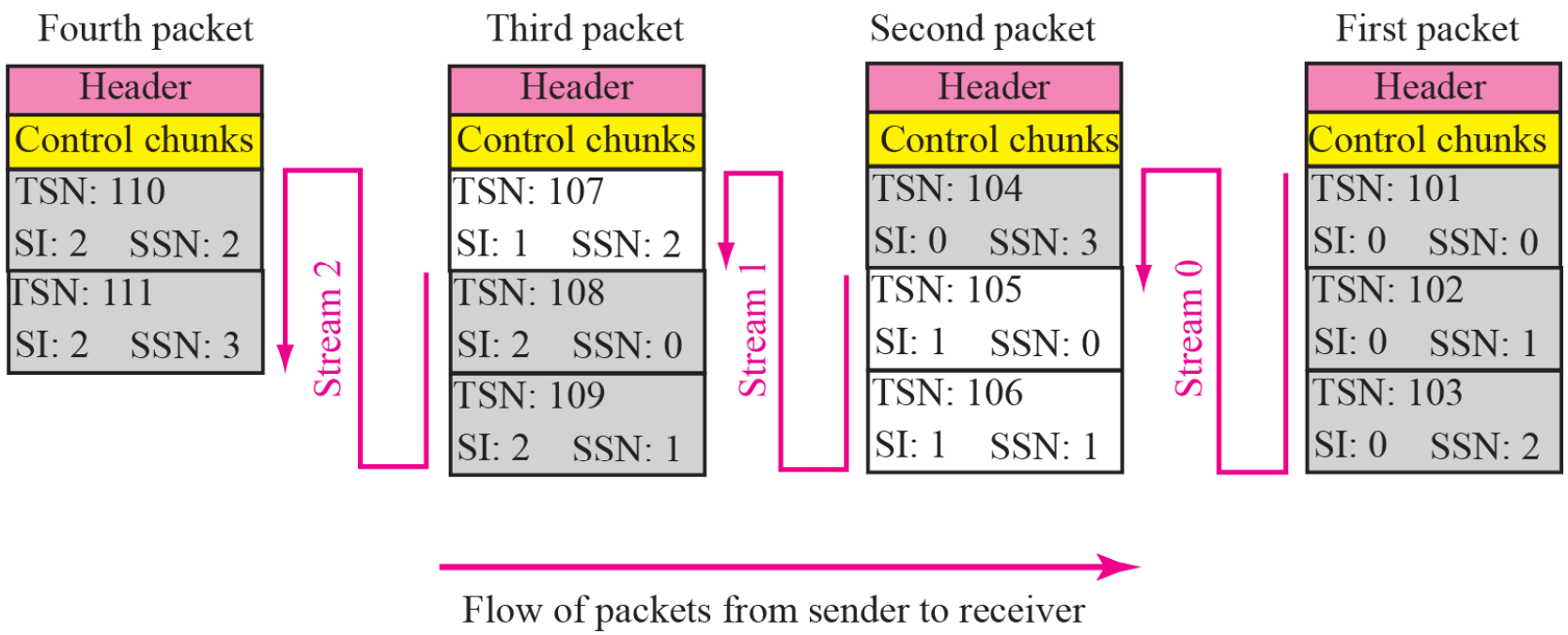
SCTP Associations and Streams

- Connection-oriented protocols require a setup procedure
  - To establish this state, both sides go through a specific set of exchanges
    - TCP uses a 3-way handshake (SYN, SYN/ACK, ACK) – “Connection”
    - SCTP uses a 4-way handshake – “Association” for reliable data transfer
- An SCTP association can be represented as a pair of SCTP endpoints:
  - assoc = { [10.1.61.11 : 2223], [161.10.8.221, 120.1.1.5 : 80]}
  - An SCTP endpoint may have multiple associations
  - Only one association may be established between any two SCTP endpoints
- SCTP stream
  - Messages are sent within a stream identified by a Stream Identifier (SI)
  - Messages can be ordered or un-ordered
    - Each ordered message sent within a stream is also assigned a Stream Sequence Number (SSN)
    - Unordered messages have no SSN and are always delivered

Data Chunks and Streams

- Data chunks are identified by 3 identifiers: TSN, SI, and SSN.
  - TSN is a cumulative number identifying the association
  - SI defines the stream
  - SSN defines the chunk in a stream

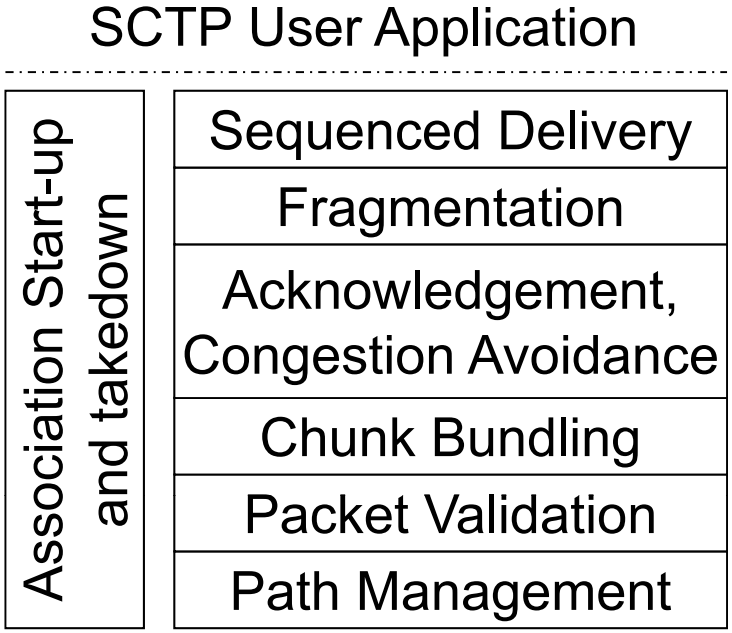
TSN: Transmission Sequence Number
SI: Stream Identifier
SSN: Stream Sequence Number



- Acknowledgment numbers are used to acknowledge data chunks
  - Control chunks are acknowledged by other control chunks

SCTP Feature List

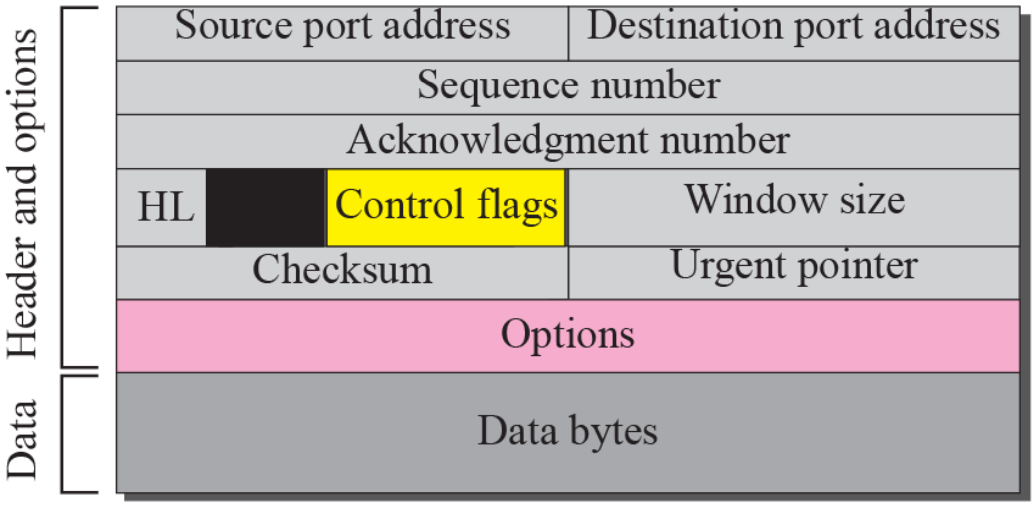
- Message-oriented nature and association set-up
- Partially ordered data transfer (Chunks)
- Reliable data transfer
- No Head-of-line blocking
- Multi-streaming
- Multi-homing
- Four-way handshake association initialization
- Chunk bundling
- Fragmentation
- Packet validation
- Path and peer monitoring
- Takedown



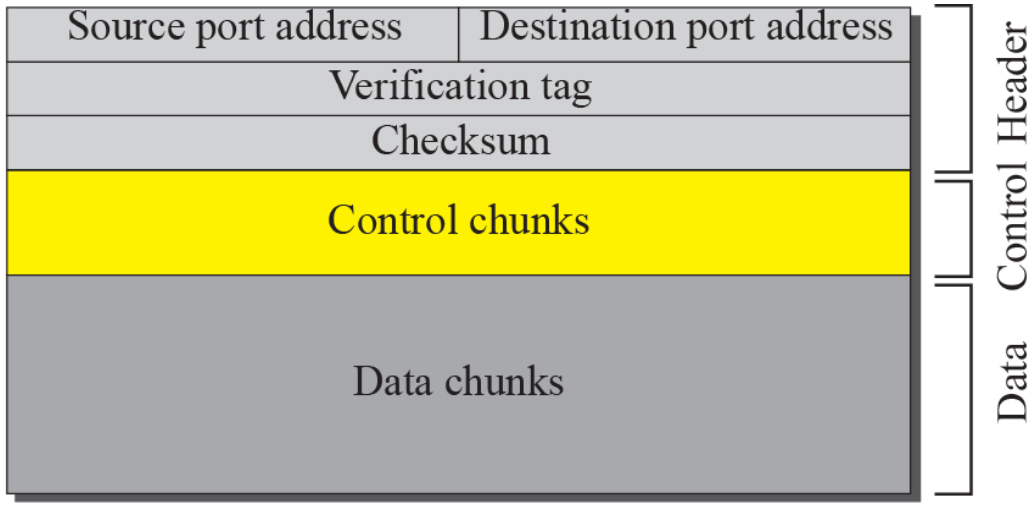
Functional View: SCTP Transport Services

[RFC 2960]

TCP vs. SCTP Packet



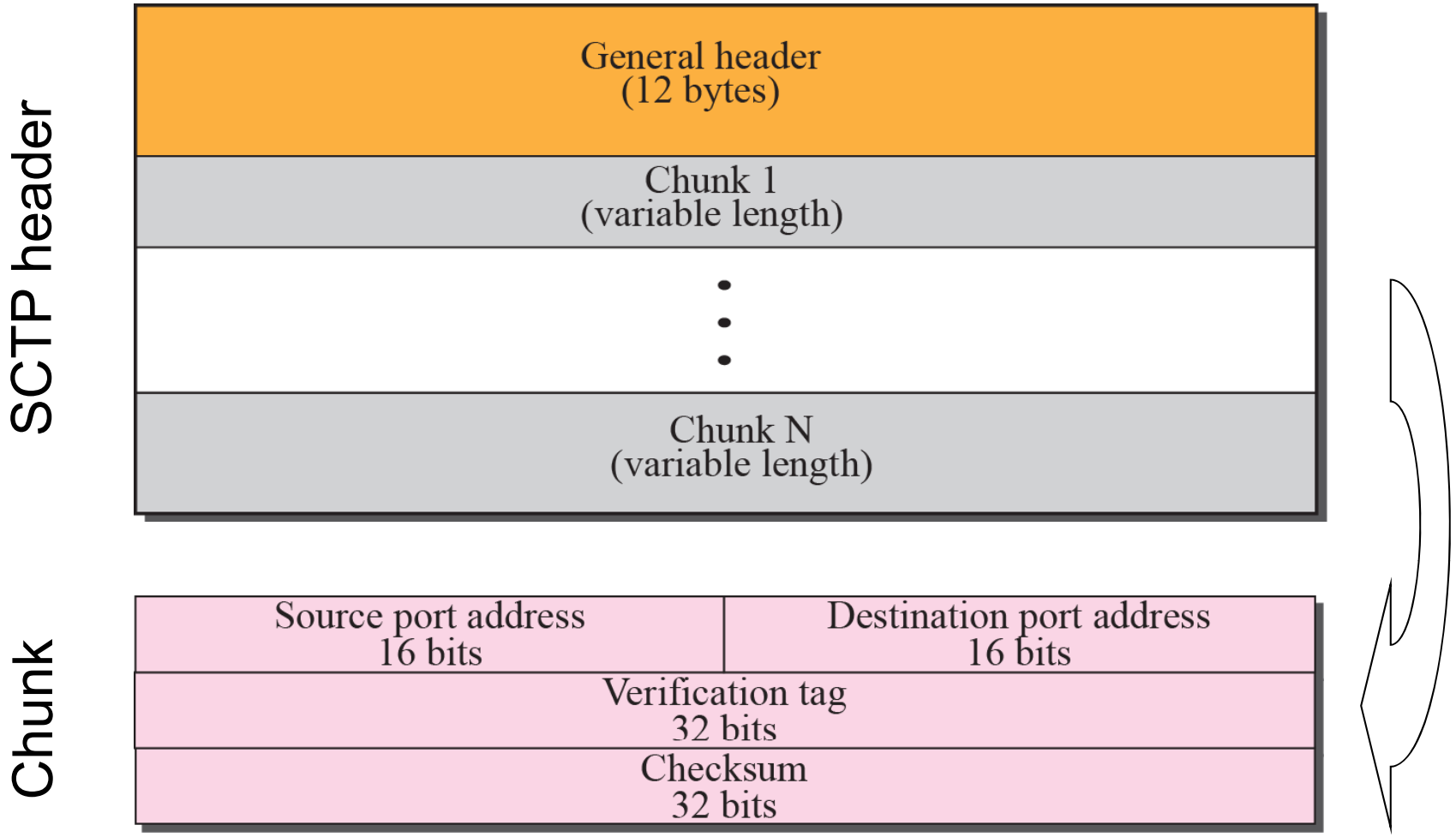
A segment in TCP



A packet in SCTP

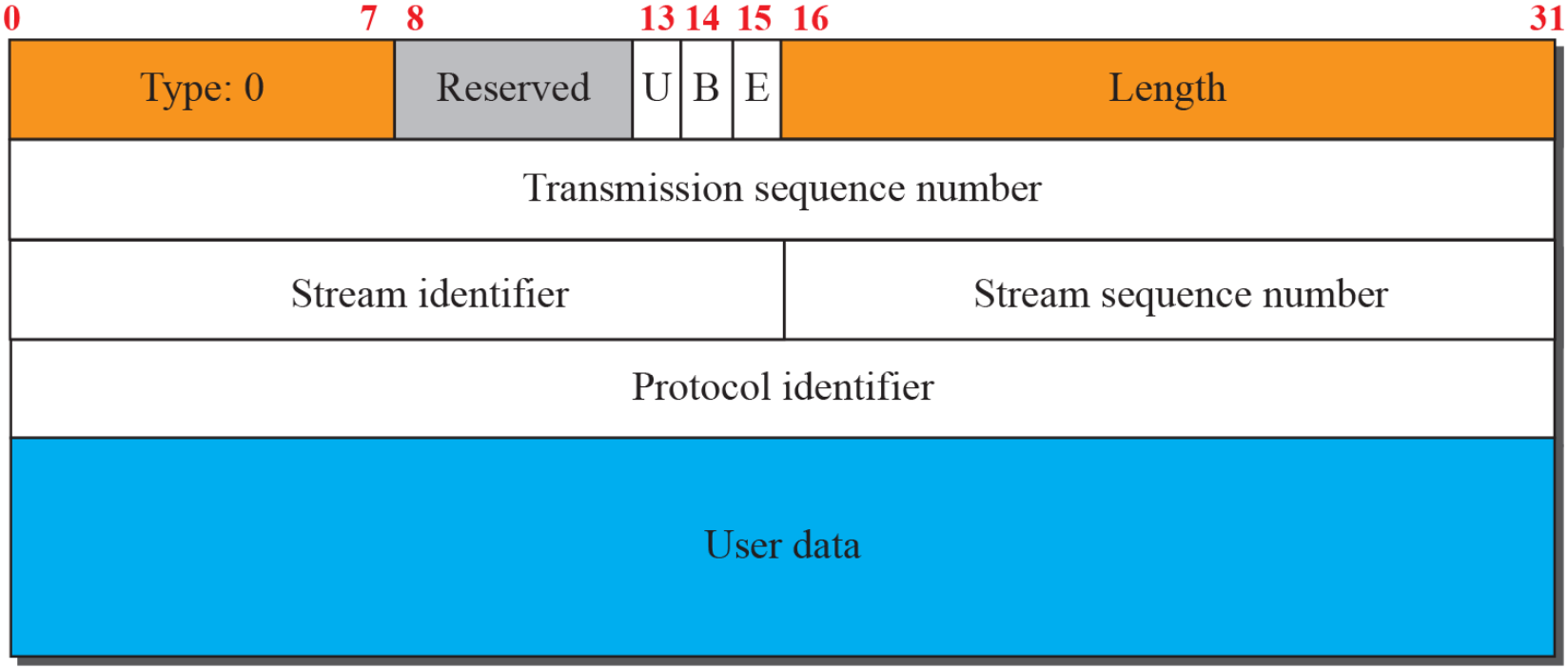
SCTP Header and Chunks

- In an SCTP packet, control chunks come before data chunks



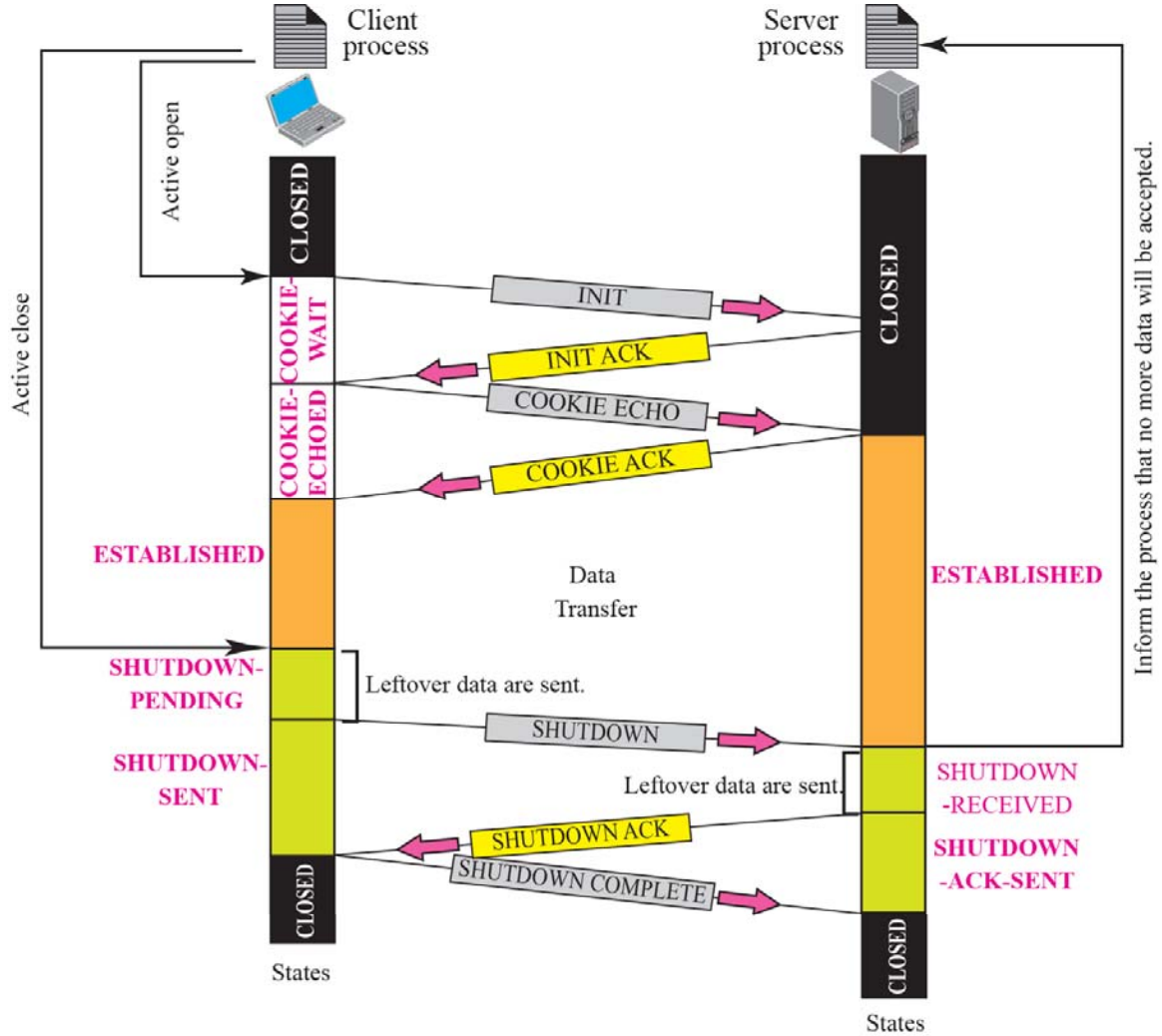
Data Chunk

- Data chunk cannot carry data belonging to more than one message
  - Message can be split into several chunks.
- Data field of the data chunk must carry at least one byte of data, which means the value of length field cannot be less than 17.



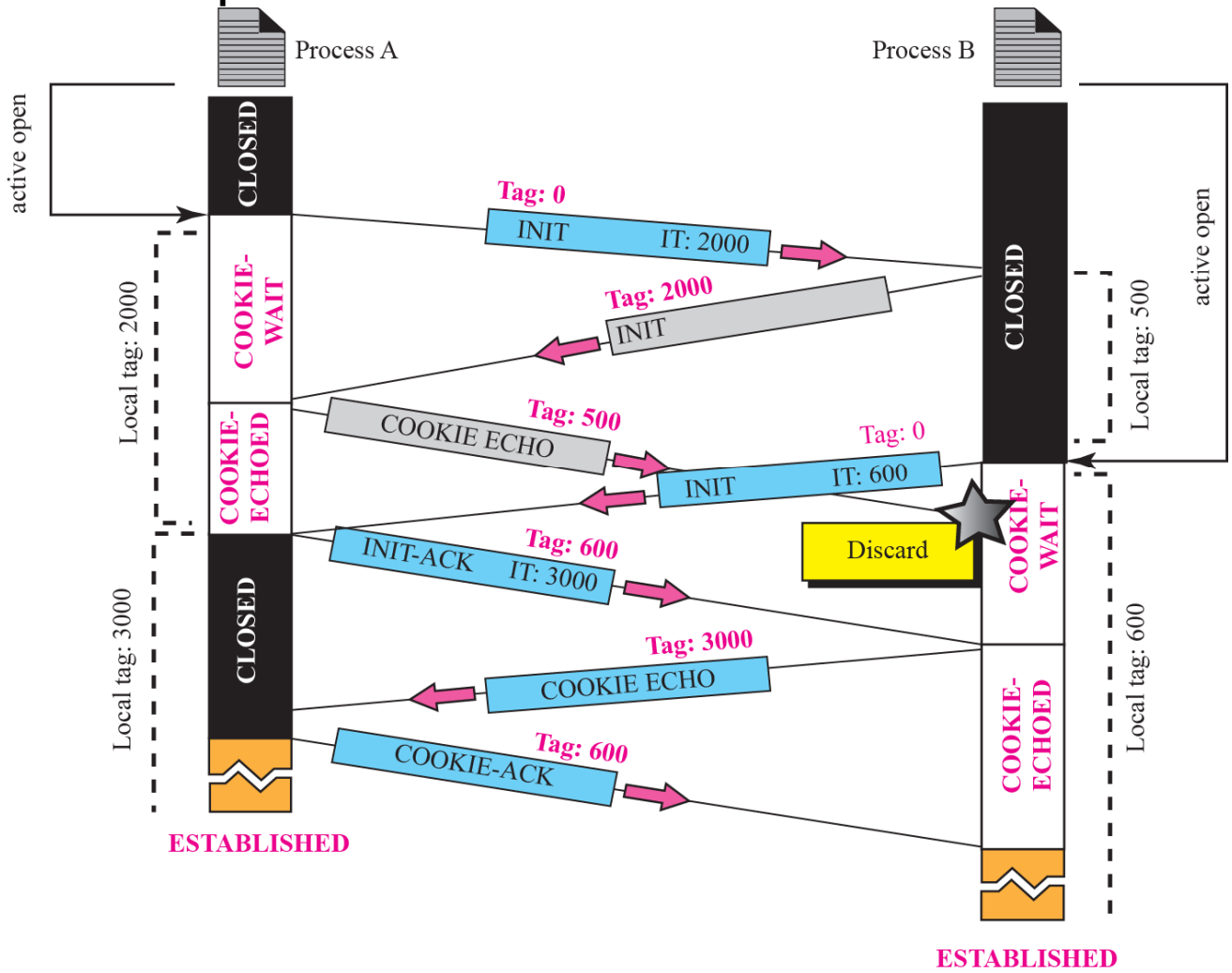
Connection/Association Set-up (1)

- Common Scenario of State



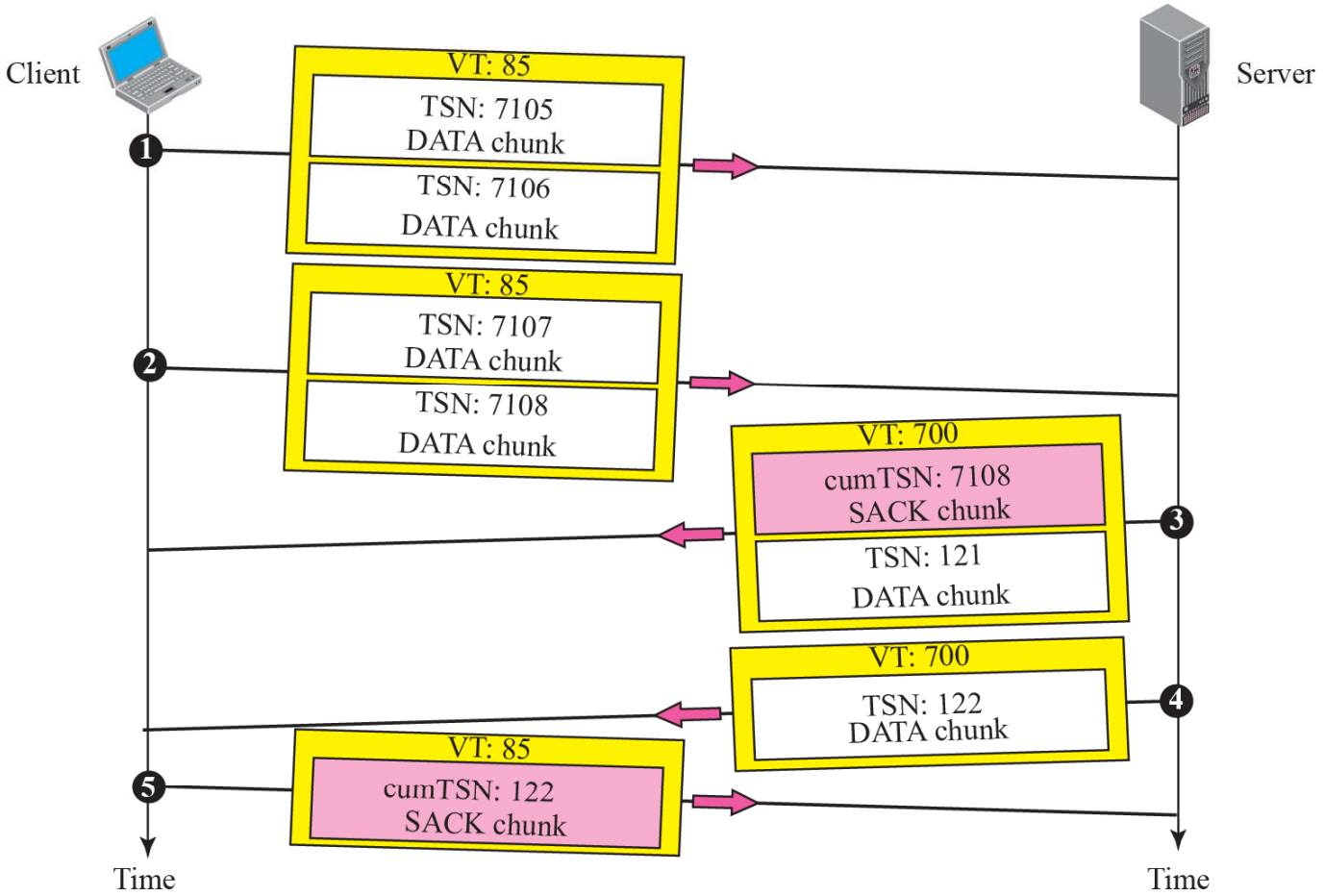
Connection/Association Set-up (2)

- Simultaneous open



Simple Data Transfer

- Acknowledgment in SCTP defines the cumulative TSN, the TSN of the last data chunk received in order

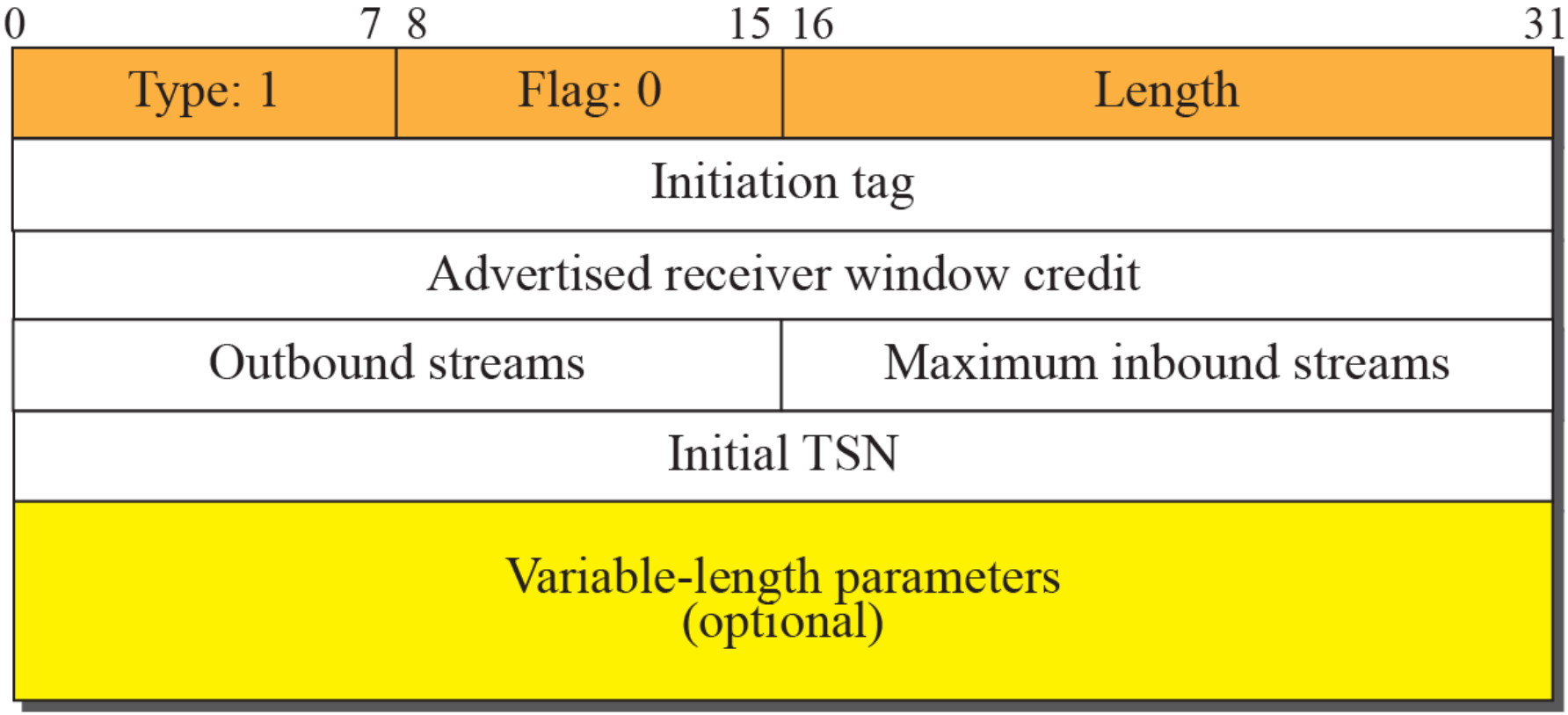


Data and Control Chunks Overview

Type	*Chunk*	*Description*
0	DATA	User data
1	INIT	Sets up an association
2	INIT ACK	Acknowledges INIT chunk
3	SACK	Selective acknowledgment
4	HEARTBEAT	Probes the peer for liveliness
5	HEARTBEAT ACK	Acknowledges HEARTBEAT chunk
6	ABORT	Abort an association
7	SHUTDOWN	Terminates an association
8	SHUTDOWN ACK	Acknowledges SHUTDOWN chunk
9	ERROR	Reports errors without shutting down
10	COOKIE ECHO	Third packet in association establishment
11	COOKIE ACK	Acknowledges COOKIE ECHO chunk
14	SHUTDOWN COMPLETE	Third packet in association termination
192	FORWARD TSN	For adjusting cumulating TSN

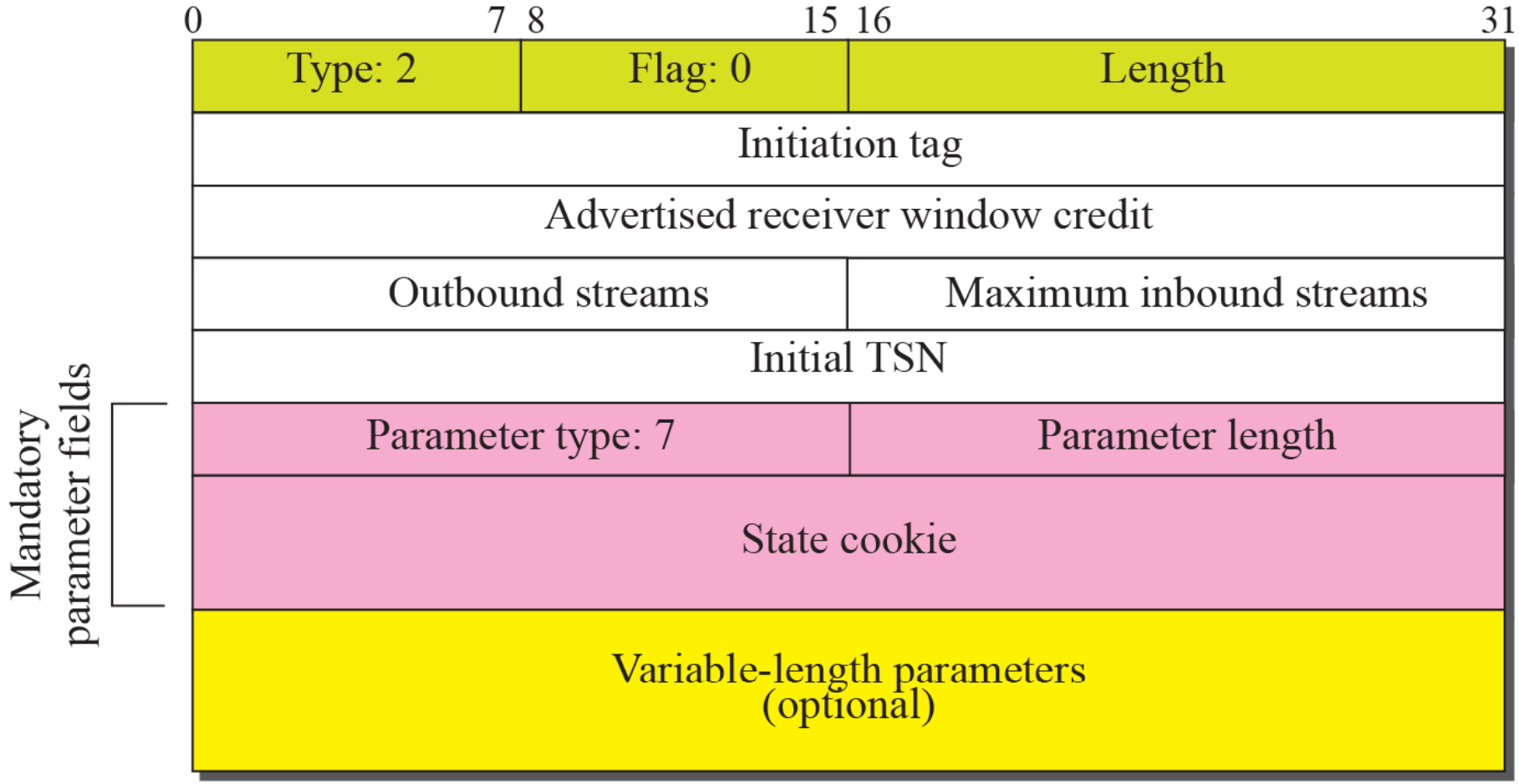
INIT Chunk

- No other chunk can be carried in a packet that carries an INIT chunk



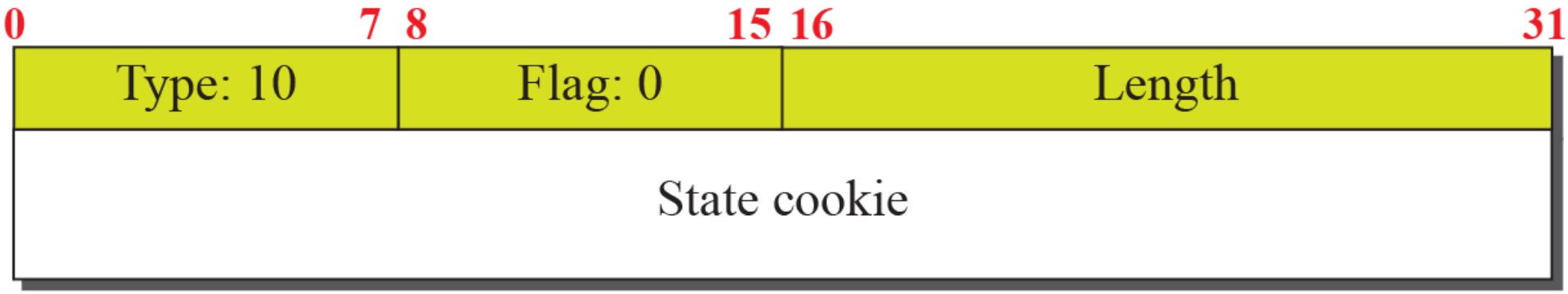
INIT ACK Chunk

- No other chunk can be carried in a packet that carries an INIT ACK chunk

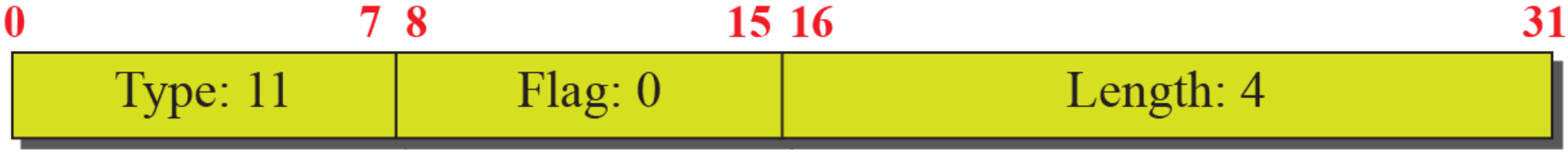


Other Chunk Types

- COOKIE ECHO Chunk

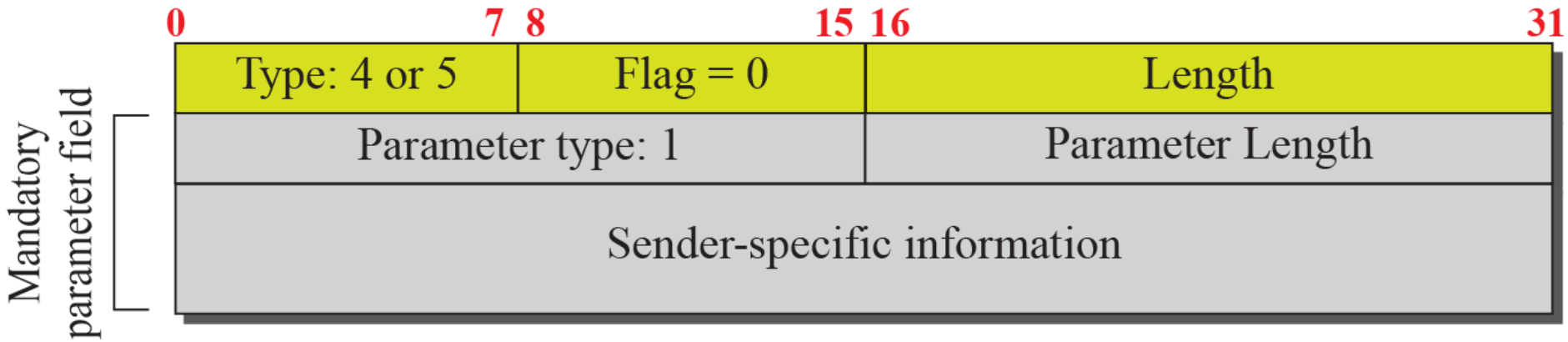


- COOKIE ACK



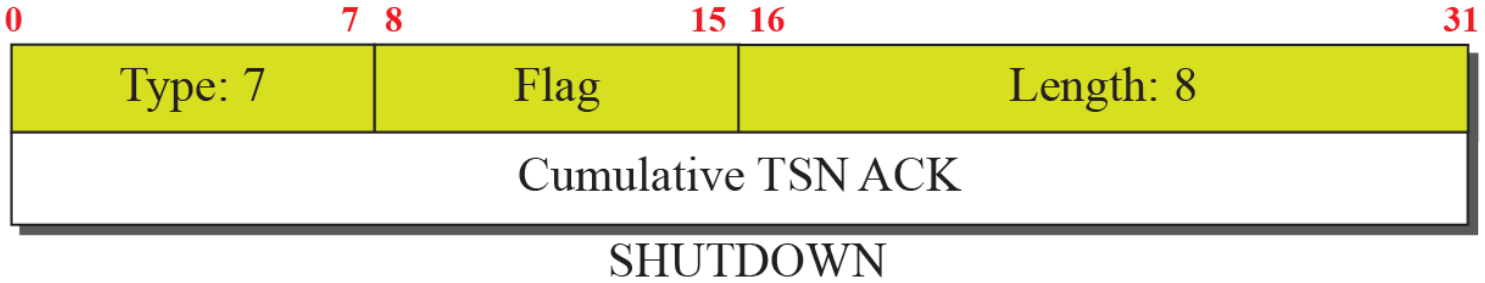
HEARTBEAT and HEARTBEAT ACK Chunk

- An endpoint should send a heartbeat chunk to its peer endpoint to probe the reachability of a particular destination transport address defined in the present association
- The Sender-specific Heartbeat Info field should normally include information about the sender‘s current time, when this heartbeat chunk is sent and the destination transport address to which this heartbeat is sent to

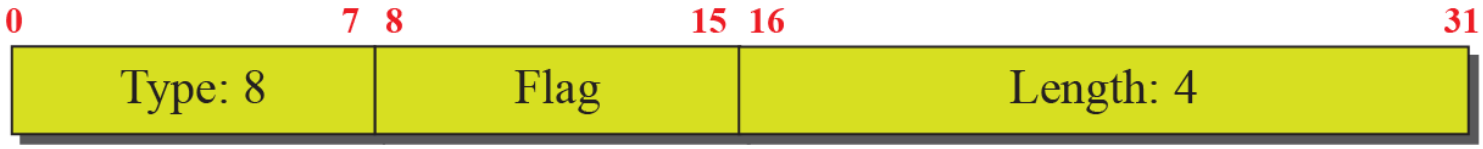


- Used to periodically probe the condition of an association

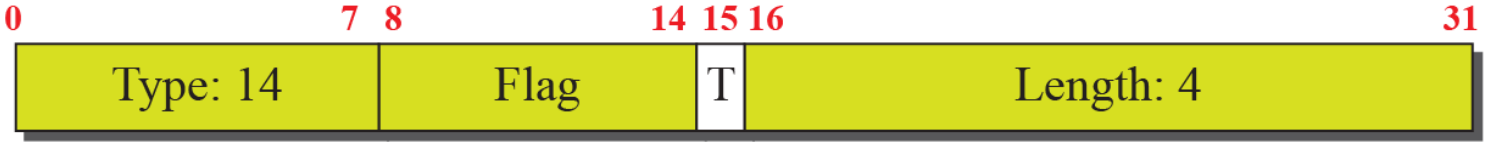
SHUTDOWN and ERROR Chunks



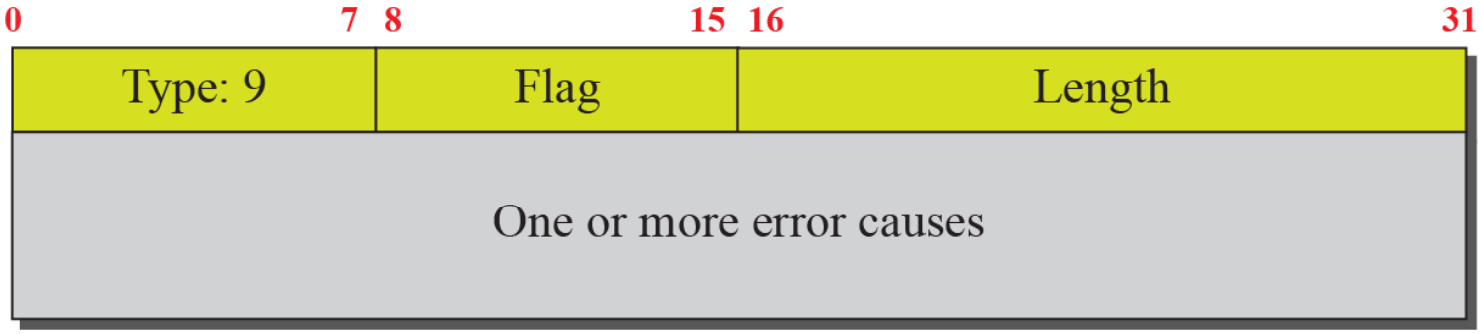
SHUTDOWN



SHUTDOWN ACK

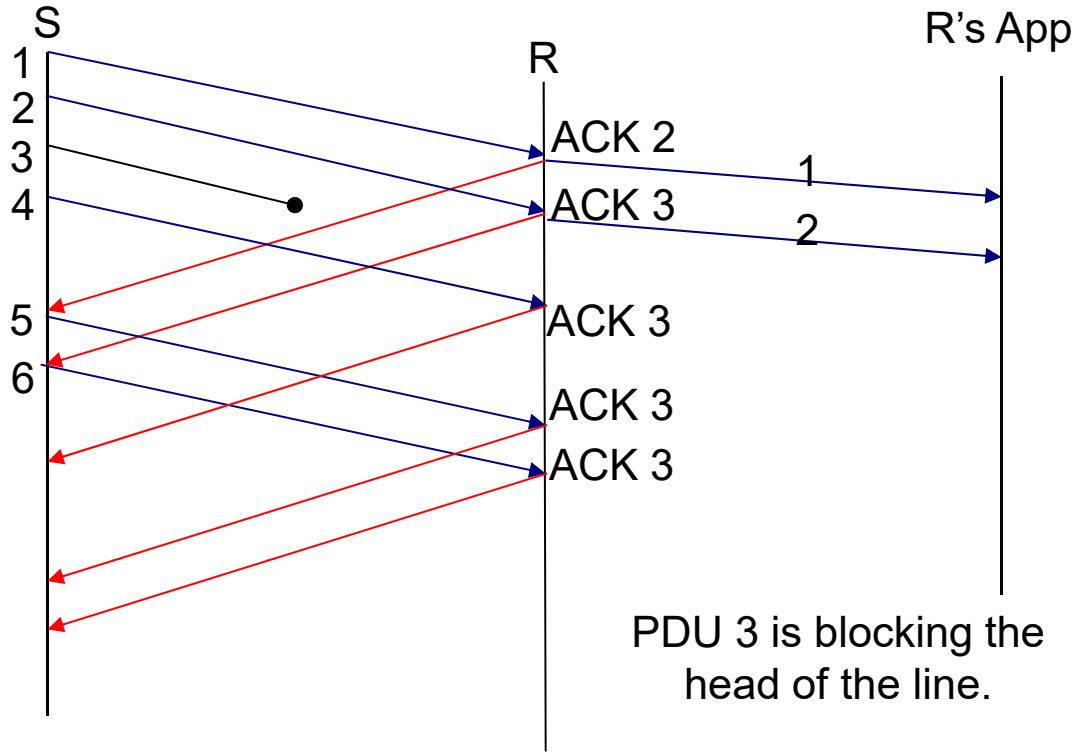


SHUTDOWN COMPLETE



Head-of-line Blocking

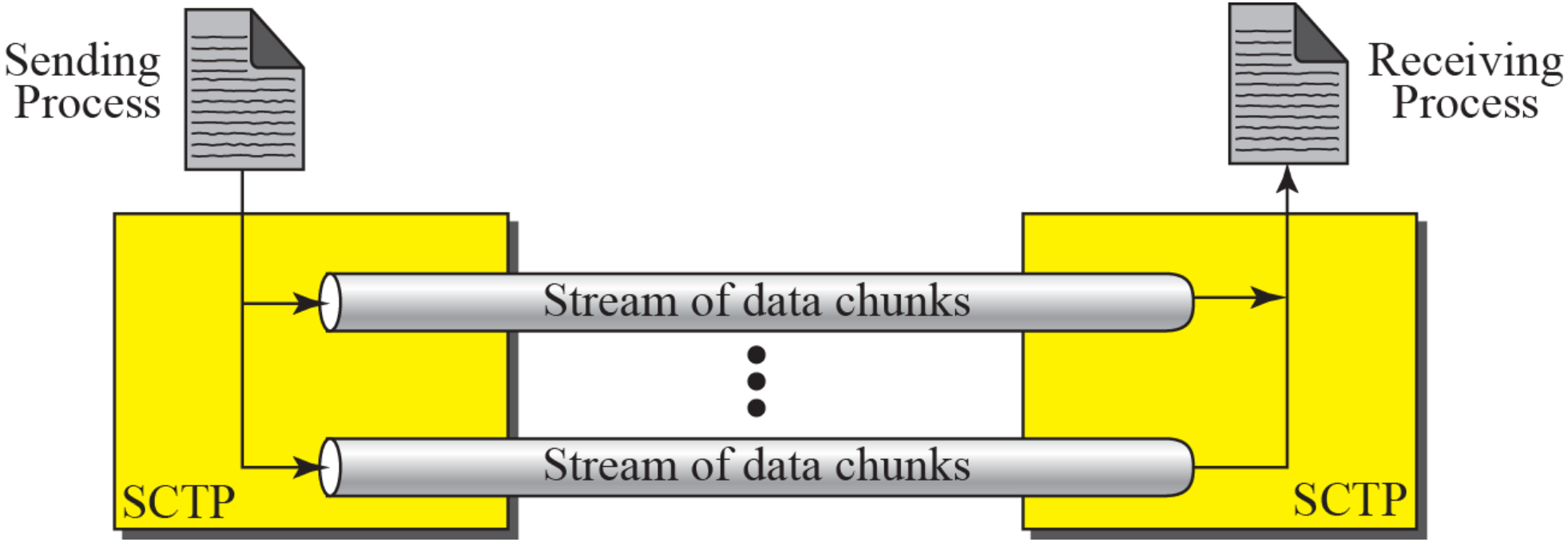
- TCP provides a single data stream
  - When a segment is lost, subsequent segments must wait to be processed.
  - Problem for some applications, such as telephony or streaming



- SCTP provides multiple independent streams per association

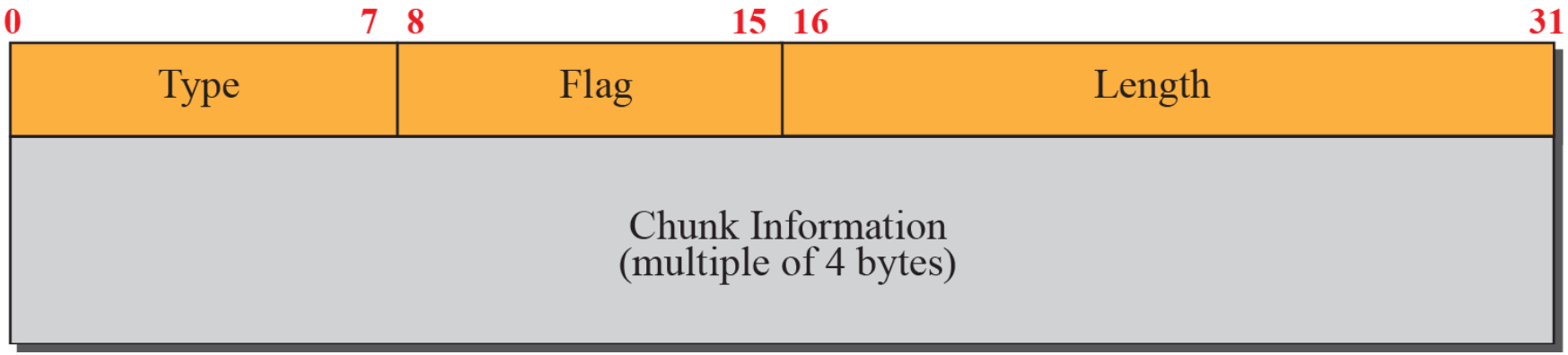
Multi-streaming (1)

- An association in SCTP can involve multiple streams
  - Sequence of messages maintained per stream
  - If one of the streams is blocked, other streams can still deliver their data
  - Reduces unnecessary head-of-line blocking



Multi-streaming (2)

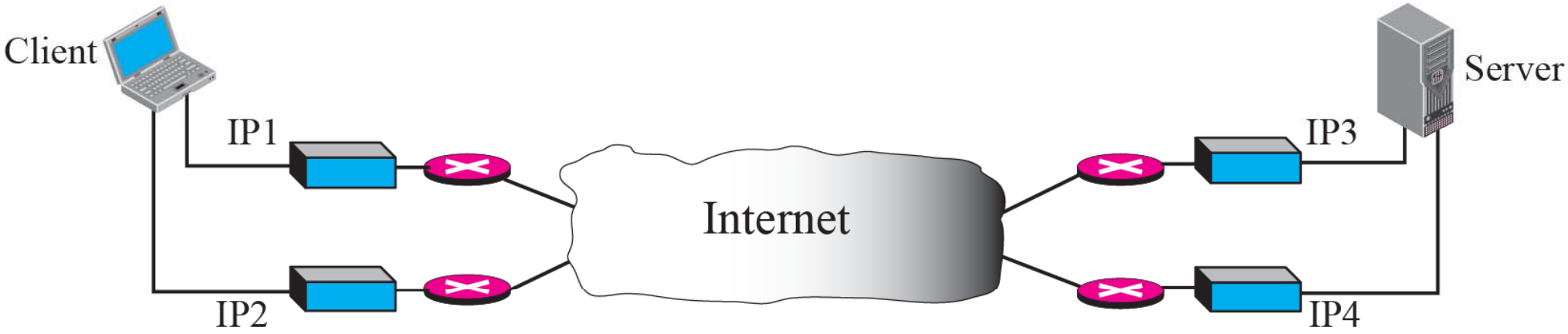
- Chunks need to terminate on a 32 bit (4 Byte) boundary



- Number of padding bytes is not included in the value of the length field

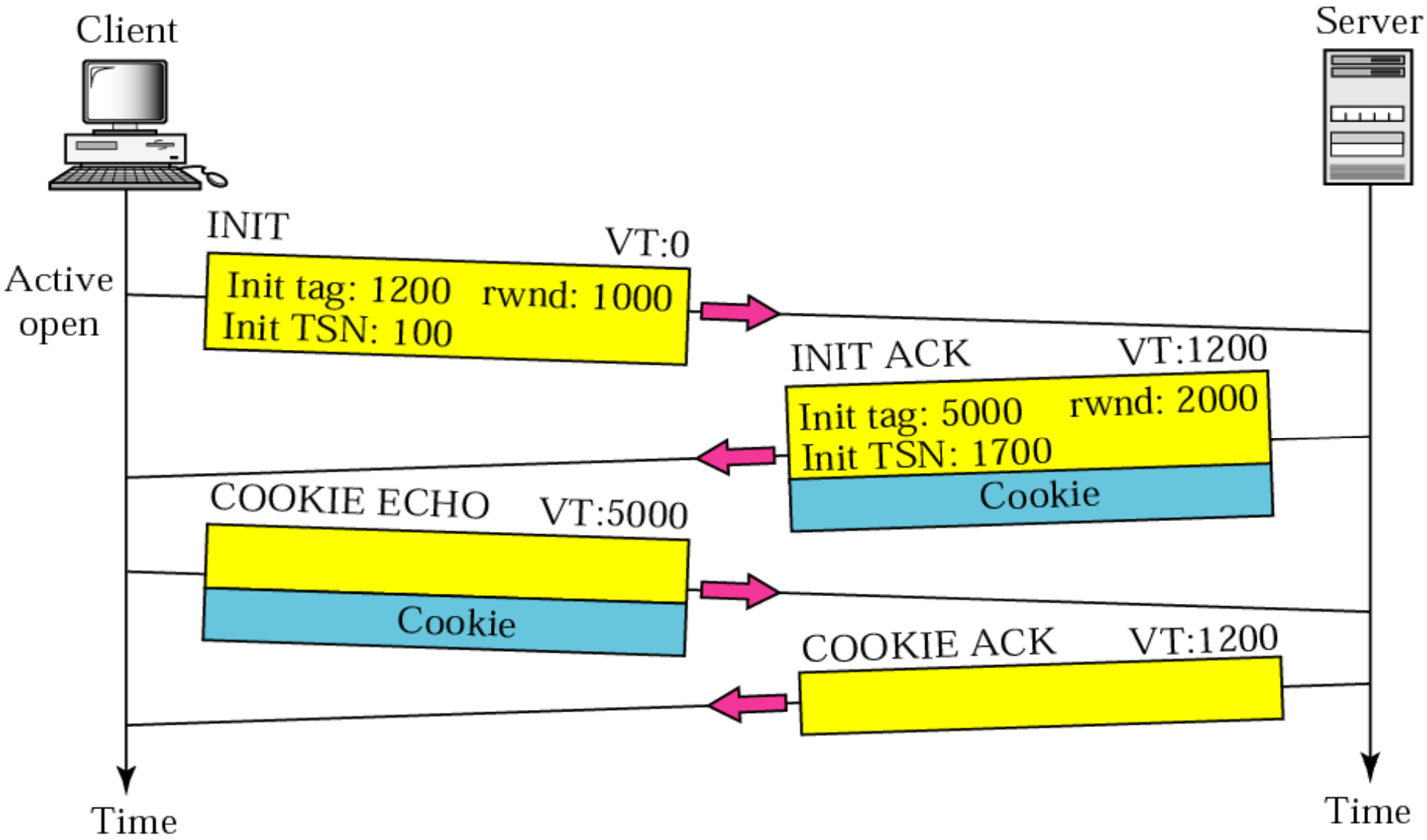
Multi-homing

- SCTP associations allow multiple IP addresses for each end



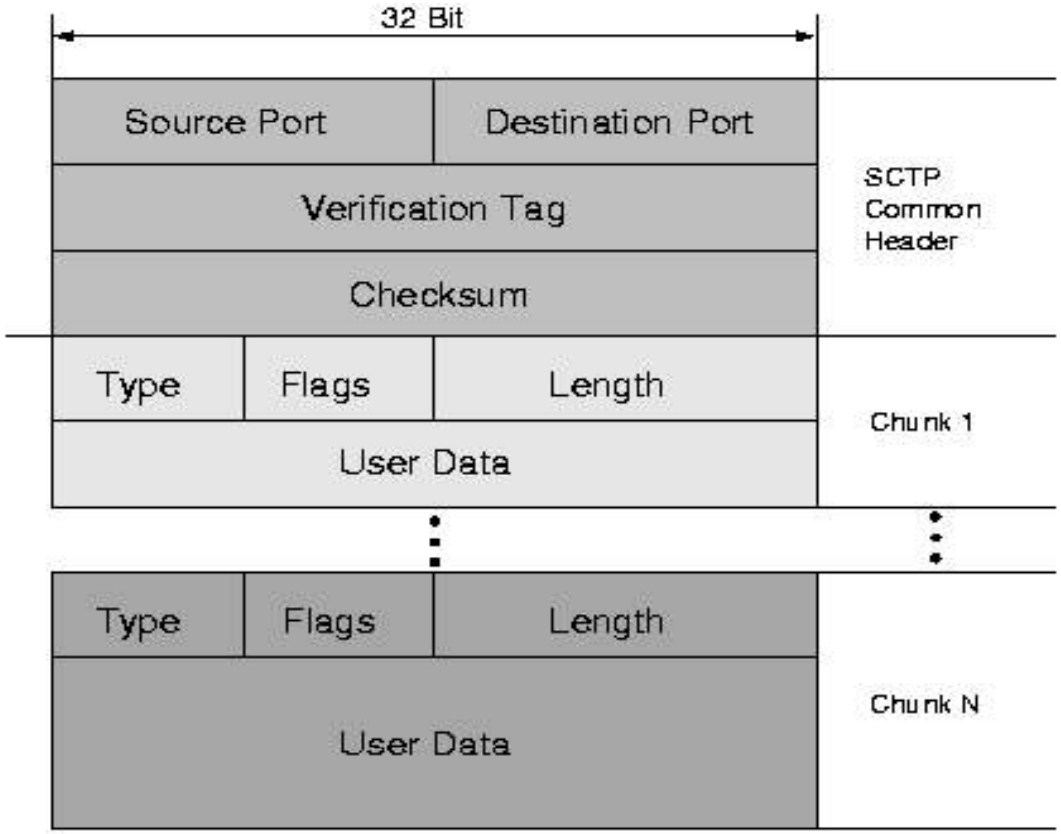
- SCTP does not allow load sharing between different paths
- Multi-homing it used only for fault tolerance purposes

Four-way Handshake Association Initialization



Chunk Bundling

- SCTP optionally bundles multiple user messages into a single SCTP packet. Each chunk may contain either user data or SCTP control information.

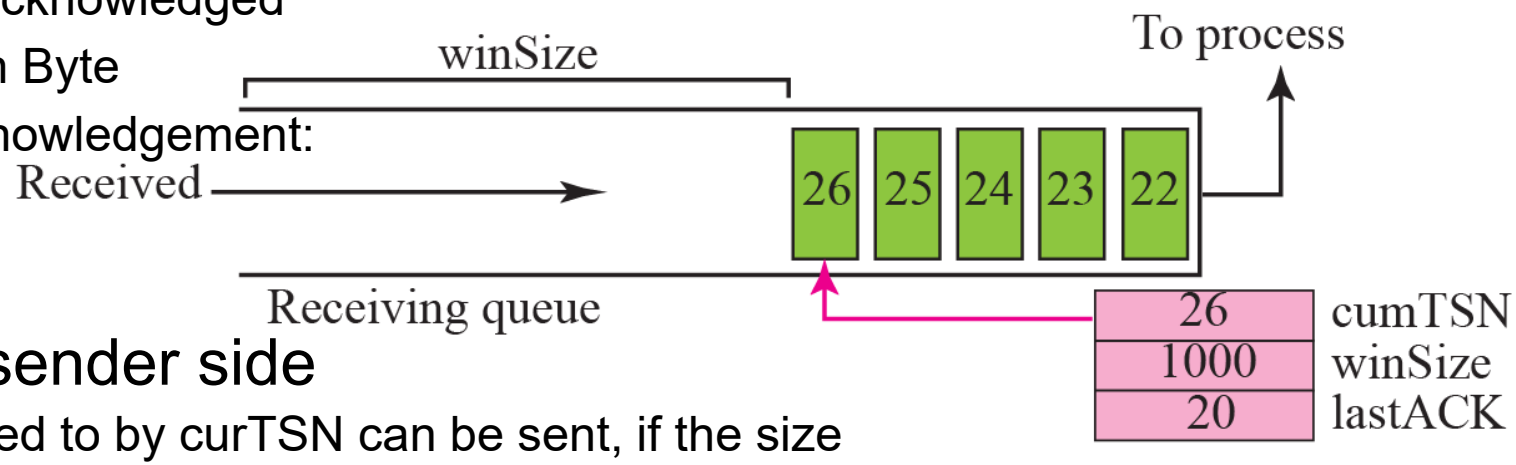


Fragmentation

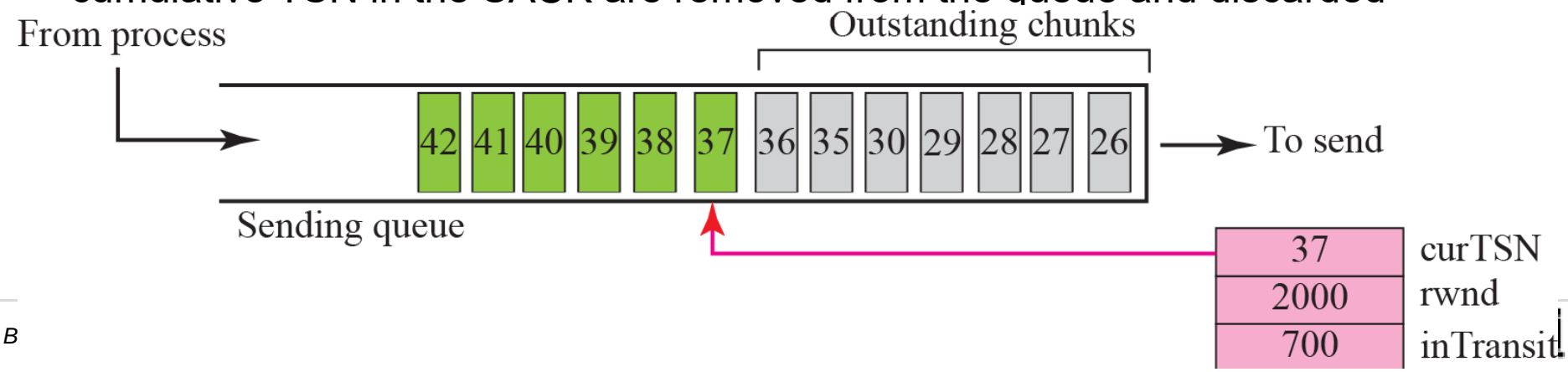
- SCTP preserves boundaries of messages from process to process when creating a DATA chunk from a message
- When needed, SCTP fragments user messages to ensure that the SCTP packet passed to the lower layer conforms to the path MTU (Maximum Transmission Unit)
  - All header chunks carry the same SI, SSN, payload protocol ID, and U flag
  - On receipt, fragments are reassembled into complete messages before being passed to the SCTP user
  - In SCTP fragmentation each fragment carries a different TSN

Path and Peer Monitoring (1)

- Flow control receiver side
  - Sent but not acknowledged
  - rwnd, cwnd: in Byte
  - TSN and Acknowledgement: in chunks



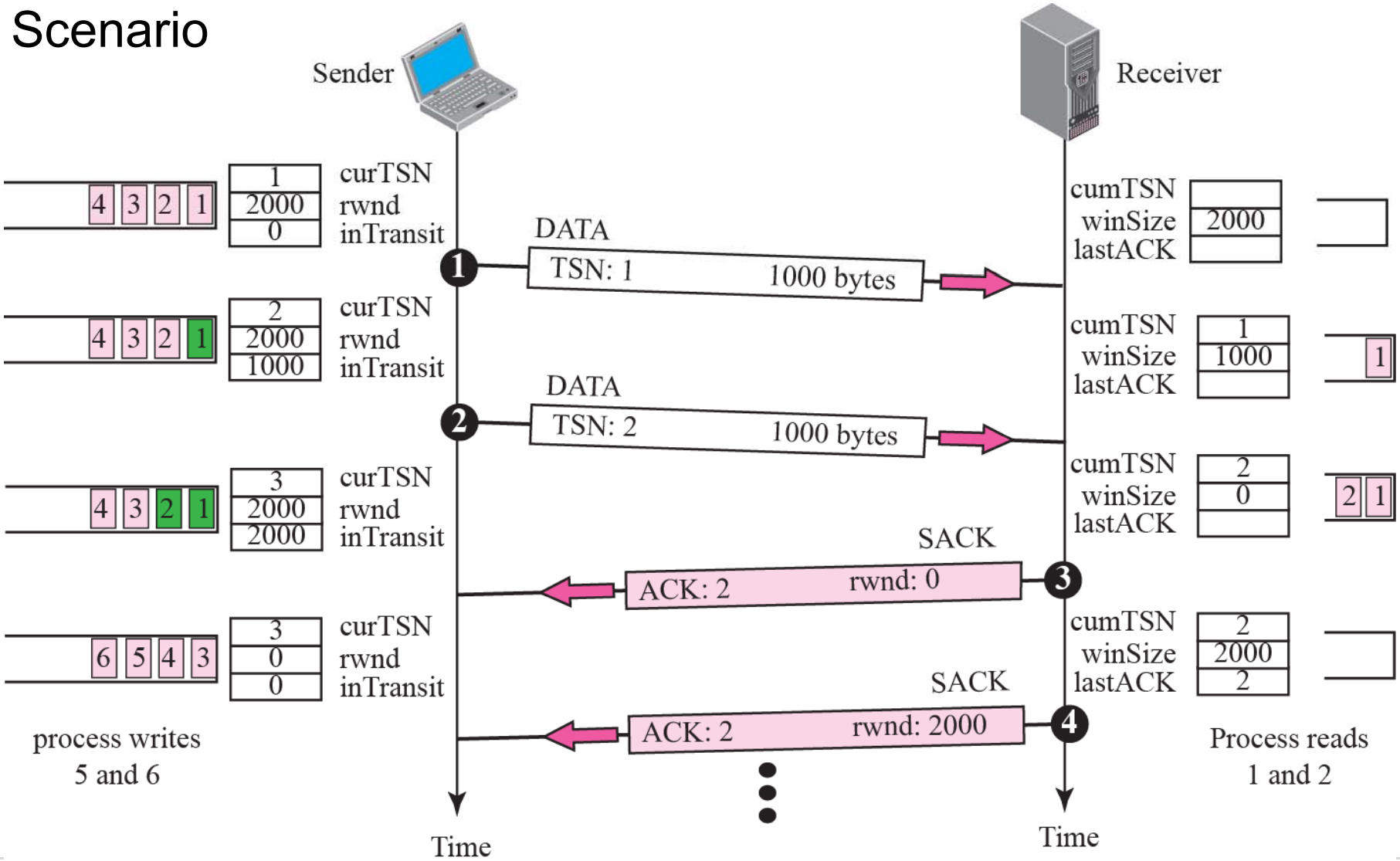
- Flow control sender side
  - A chunk pointed to by curTSN can be sent, if the size of data is less than or equal to the quantity (rwnd-inTransit)
  - When a SACK is received, chunks with a TSN less than or equal to the cumulative TSN in the SACK are removed from the queue and discarded



© 2020 B

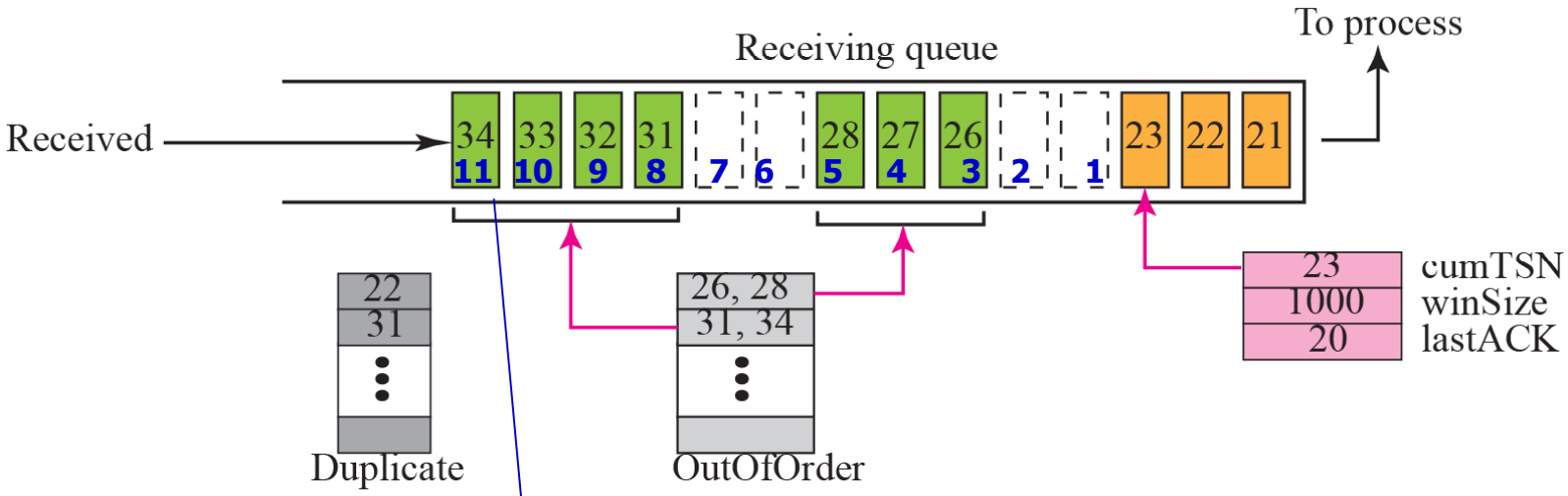
Path and Peer Monitoring (2)

- Scenario



Path and Peer Monitoring (3)

- Error control receiver side



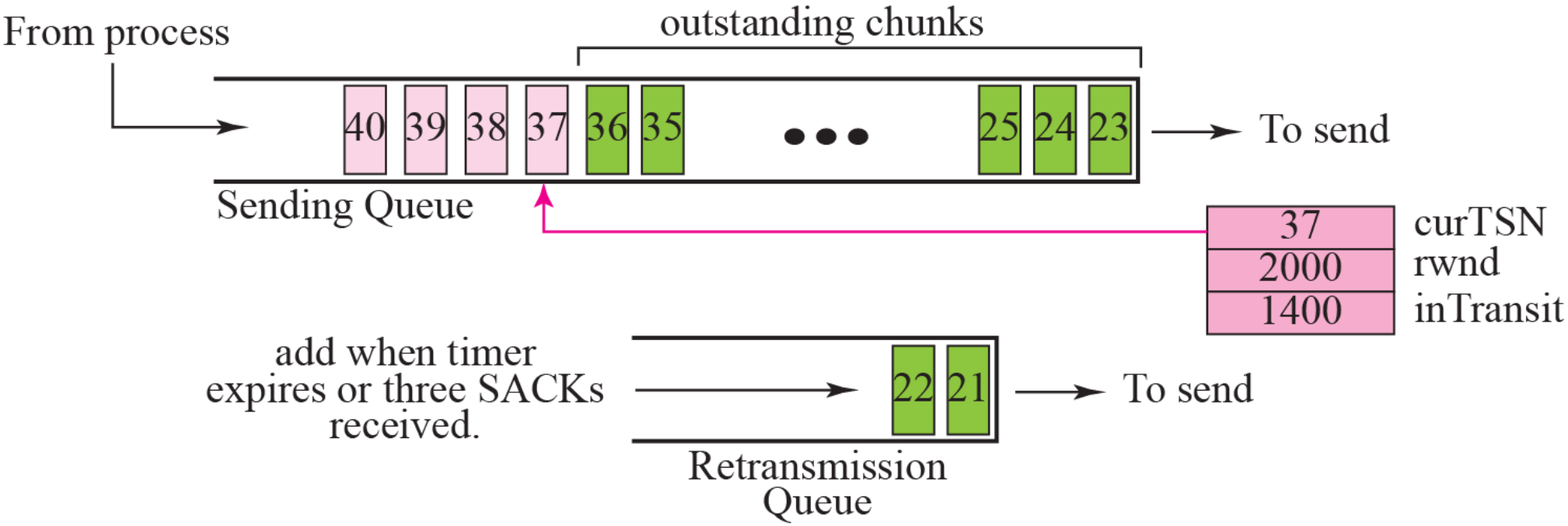
SACK chunk

Type: 3	Flag: 0	Length: 32
Cumulative TSN: 23		
Advertised receiver window credit: 1000		
Number of gap ACK blocks: 2	Number of duplicates: 2	
Gap ACK block #1 start: 3	Gap ACK block #1 end: 5	
Gap ACK block #2 start: 8	Gap ACK block #2 end: 11	
Duplicate TSN: 22		
Duplicate TSN: 31		

Numbers are relative to cumTSN

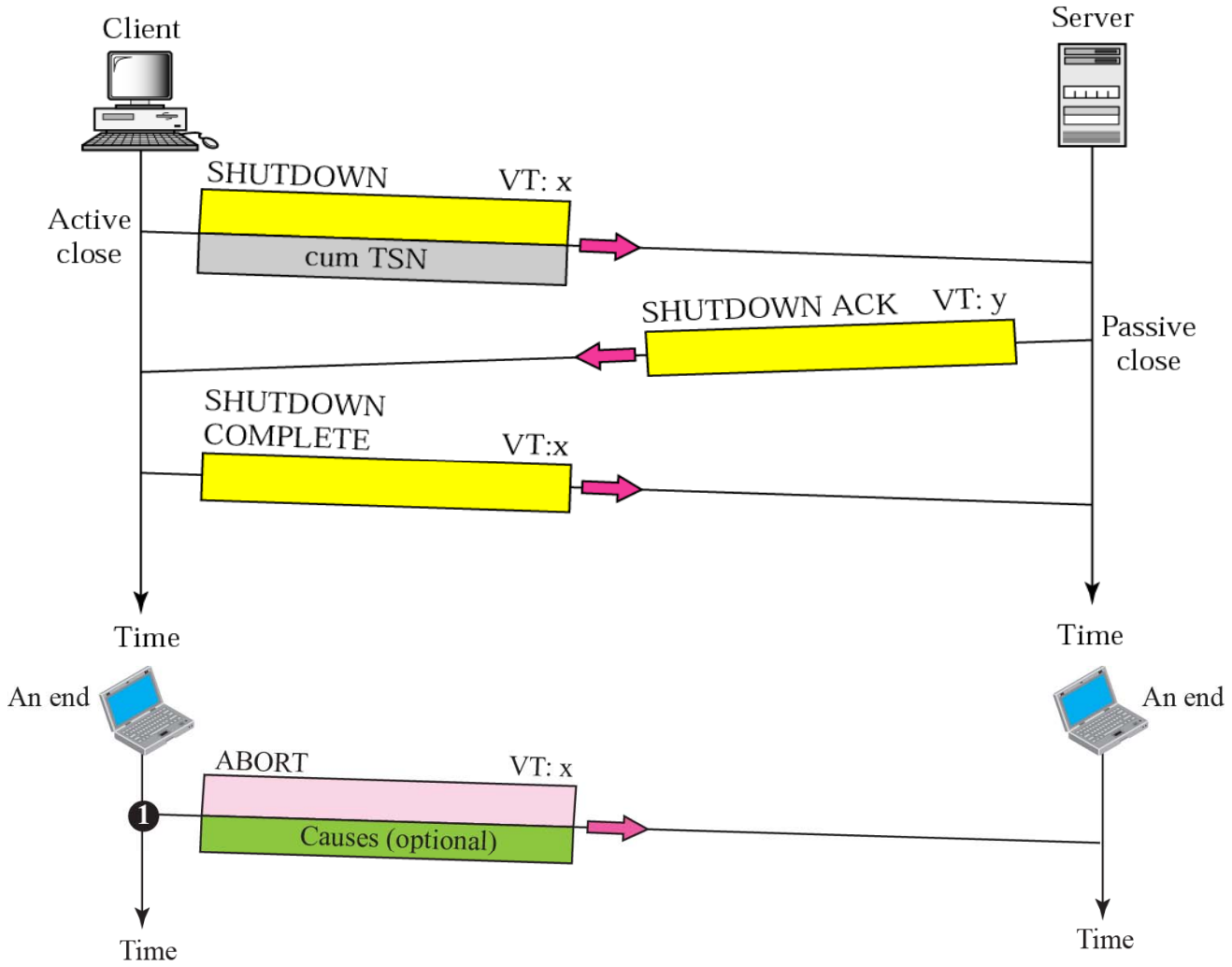
Path and Peer Monitoring (4)

- Error control sender side
  - Assume 100 bytes per chunk
  - Chunks in retransmission queue have priority



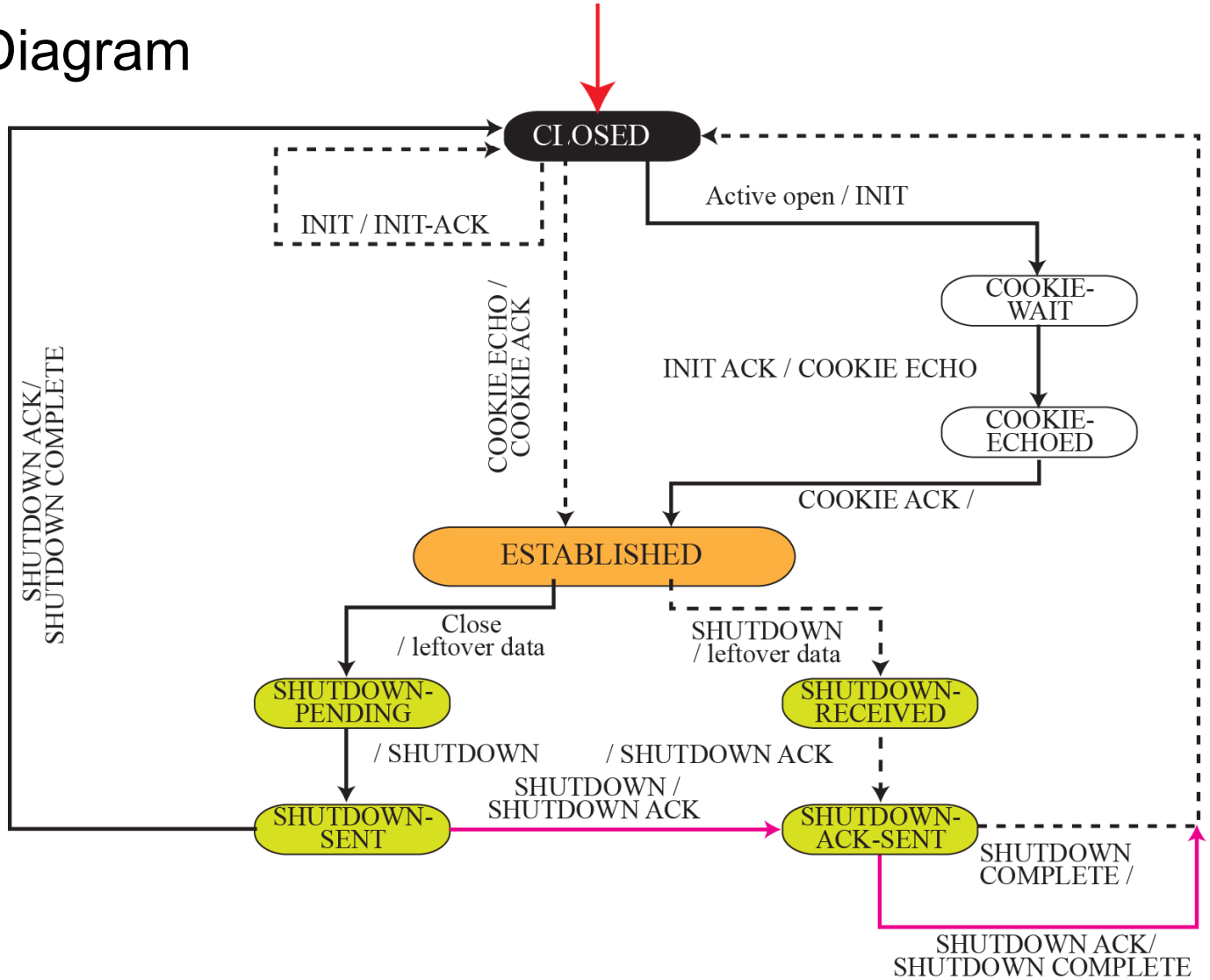
Takedown (1)

- SCTP does not allow “half-closed” associations



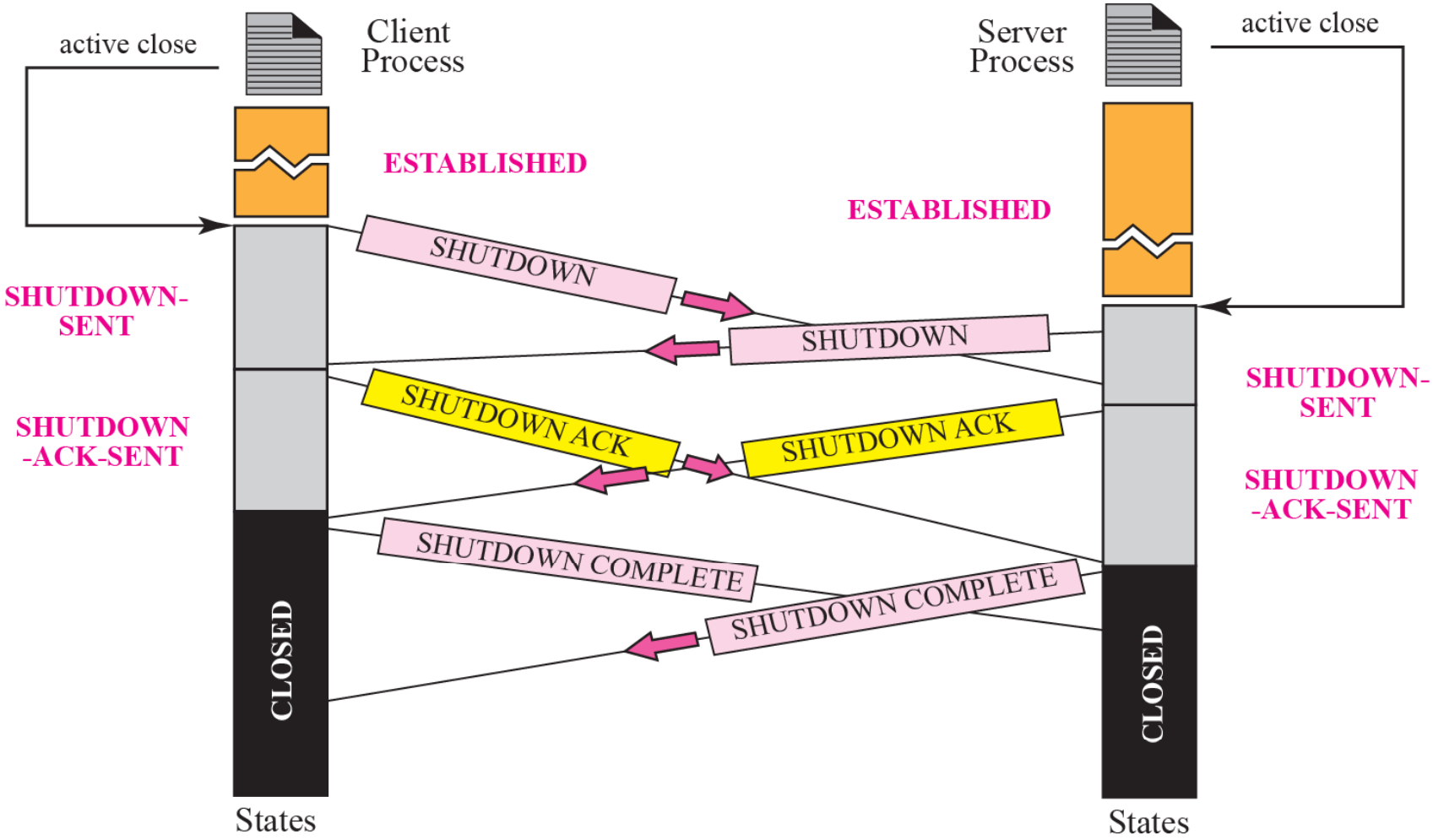
Takedown (2)

- State Diagram



Takedown (3)

- Simultaneous CLOSE



RFCs

- RFC 2960 – Stream Control Transmission Protocol
- RFC 3257 - SCTP Applicability Statement
- RFC 3286 - An introduction to SCTP
- RFC 3309 – SCTP Checksum Change
- RFC 3436 – Transport Layer Security over SCTP
- RFC 3758 – SCTP Partial Reliability Extension

Part 4: Multimedia Protocol Components

- AAA: RADIUS and Diameter
  - AAA and Security
  - Policy-based Management
  - AAA Architecture
  - RADIUS and Diameter

AAA and Security (1)

- Authentication:
  - Process of identifying *who* a user is. Typically, users prove to the system who they are by valid user name/valid password.

- Authorization:
  - Process of identifying *what* a user can do. For example, after login, an authorized user may try to issue commands which are permitted. Some systems, merge authorization and authentication into a single process.

- Accounting:
  - Process of measuring resources a user has consumed. Accounting measures the amount of system time a user has used, or the amount of data a user has sent and received.

→ Authentication, Authorization, Accounting: AAA

AAA and Security (2)

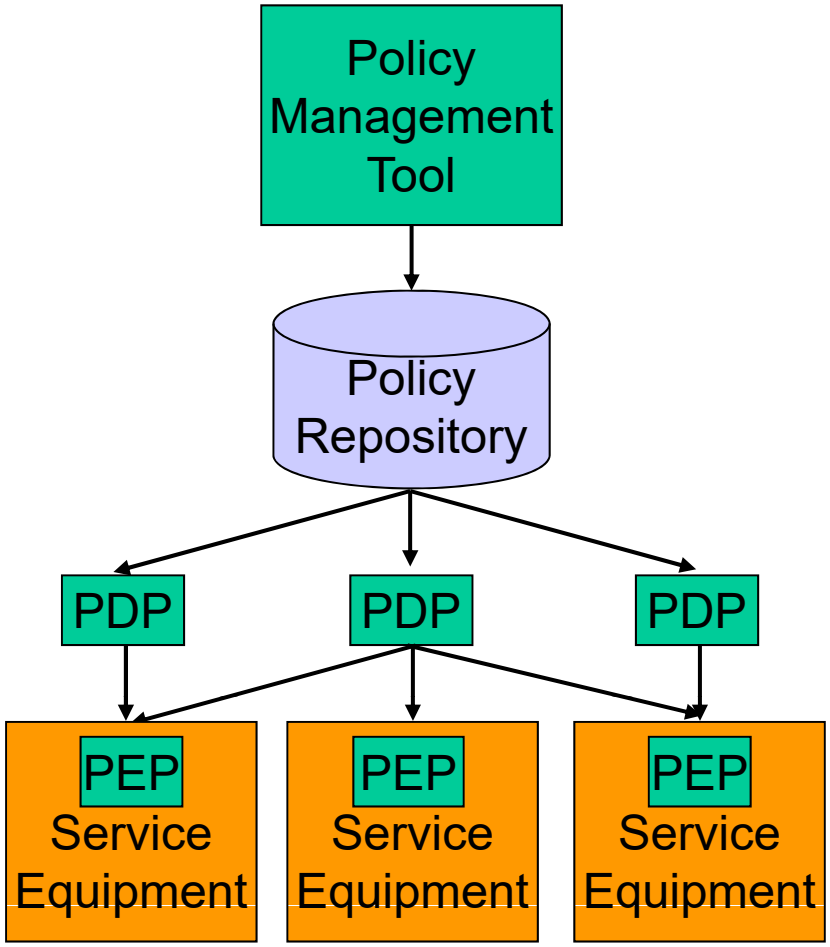
- AAA is important for effective network management and security:
  - Access of network via Network Access Server (NAS), Communication, Remote Access, or Terminal Servers
  - Control who is allowed to connect to the network
  - Control what users are allowed to do

  - Provisioning at the point of network entry, *e.g.*, dial-in users
  - Accounting of utilized resources:
    - At access (NAS)
    - Within network
    - Along communication paths

Policy-based Management Architecture

- Policy Decision Points (PDP):
  - Take decisions, evaluate policies based on observed data
- Policy Enforcement Point (PEP):
  - Transform decisions into configuration

- A systematic view:
  - Enforcement by Service Equipment
  - Entities providing authentication, authorization, accounting, auditing, charging services described by policies
- An operational view.
  - Support of mobile devices and customers



Policy-based Accounting (1)

- Based on and fitting into AAA architecture
- Reference model for describing interactions between:
  - Different metering, accounting, charging processes
  - Their configuration via policies
- Accounting policies describe rules for:
  - Generation, transport, and storage of accounting data in a standardized representation of configuration information for different elements of the accounting infrastructure
- Considers intra- and inter-domain scenarios
- Push and Pull models for obtaining accounting policies [RFC 3334]

Policy-based Accounting (2)

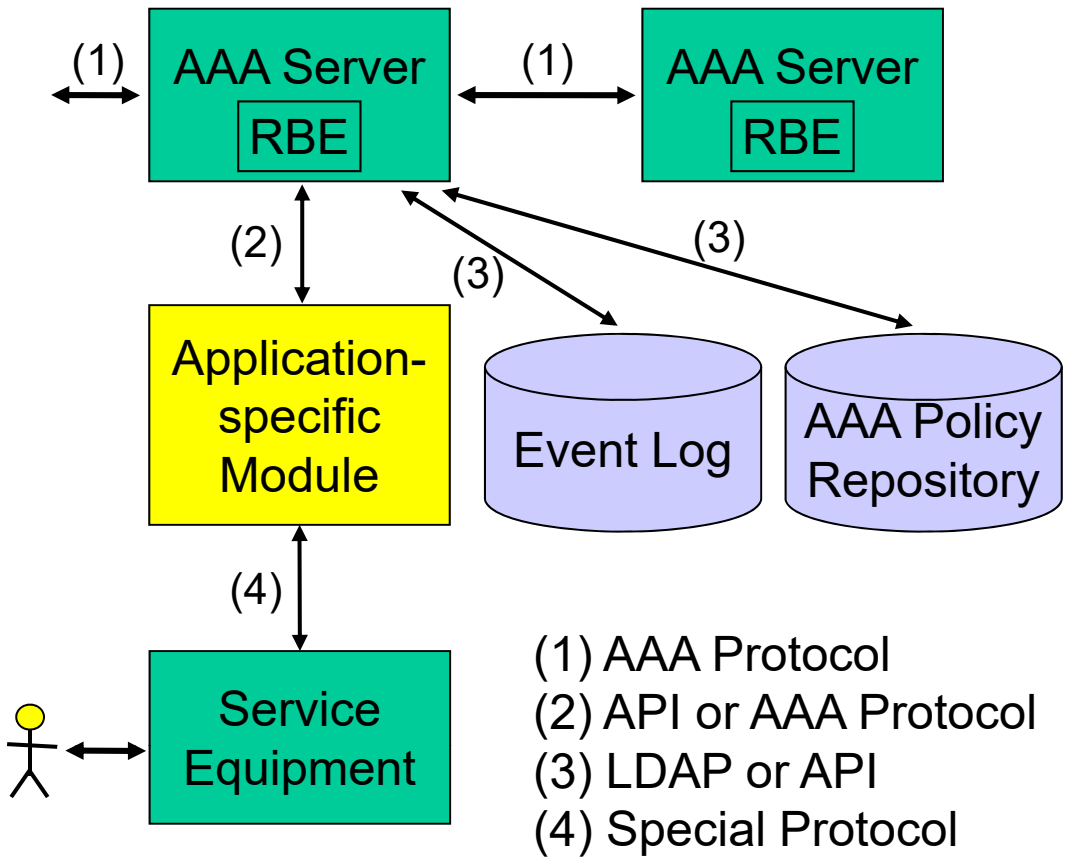
- Policies:
  - Policy rule conditions: Customer ID, IP address, Time of day, Service class, Accounting type
  - Policy rule actions: Accounting record type and destination, Report interval, Flow granularity, Meter accuracy

- Accounting services:
  - Integrated accounting: accounting seen as part of provisioned service, can collect information by directly exploiting service-specific entities like signaling servers
  - Discrete accounting: accounting is a separate service on its own, and it is not coupled with a specific service

- Examples:
  - Mobile roaming, DiffServ, NeTraMet, NetFlow configuration

[RFC 3334]

AAA Architecture and Weaknesses



AAA Authentication, Authorization, Accounting
LDAP Light-weight Directory Access Protocol
RBE Rule-based Engine

- Policy decision and policy enforcement not separated:
  - AAA Server decides on authorization, but enforces accounting
- Difficult enhancements:
  - Enforcement located in the AAA server or the Application-specific module
- AAA applied to transport, but not content, charging, and auditing
- QoS support not provided

[RFC 2903]

A^x Model: Levels and Partitions

- Horizontal levels:
  - Internet connectivity
  - Transport
  - Application
  - Content
- Vertical partitioning:
  - Control path (signaling)
  - Data path (payload)

Level	***Control Path***	***Data Path***
Content	RTSP	news, streams
Application	HTTP, H.245, SIP	video conf, IP telephony, Java applets
Transport	RSVP, RTCP, ICMP	TCP, UDP, RTP
Connectivity	DHCP	Sonet/SDH, DWDM

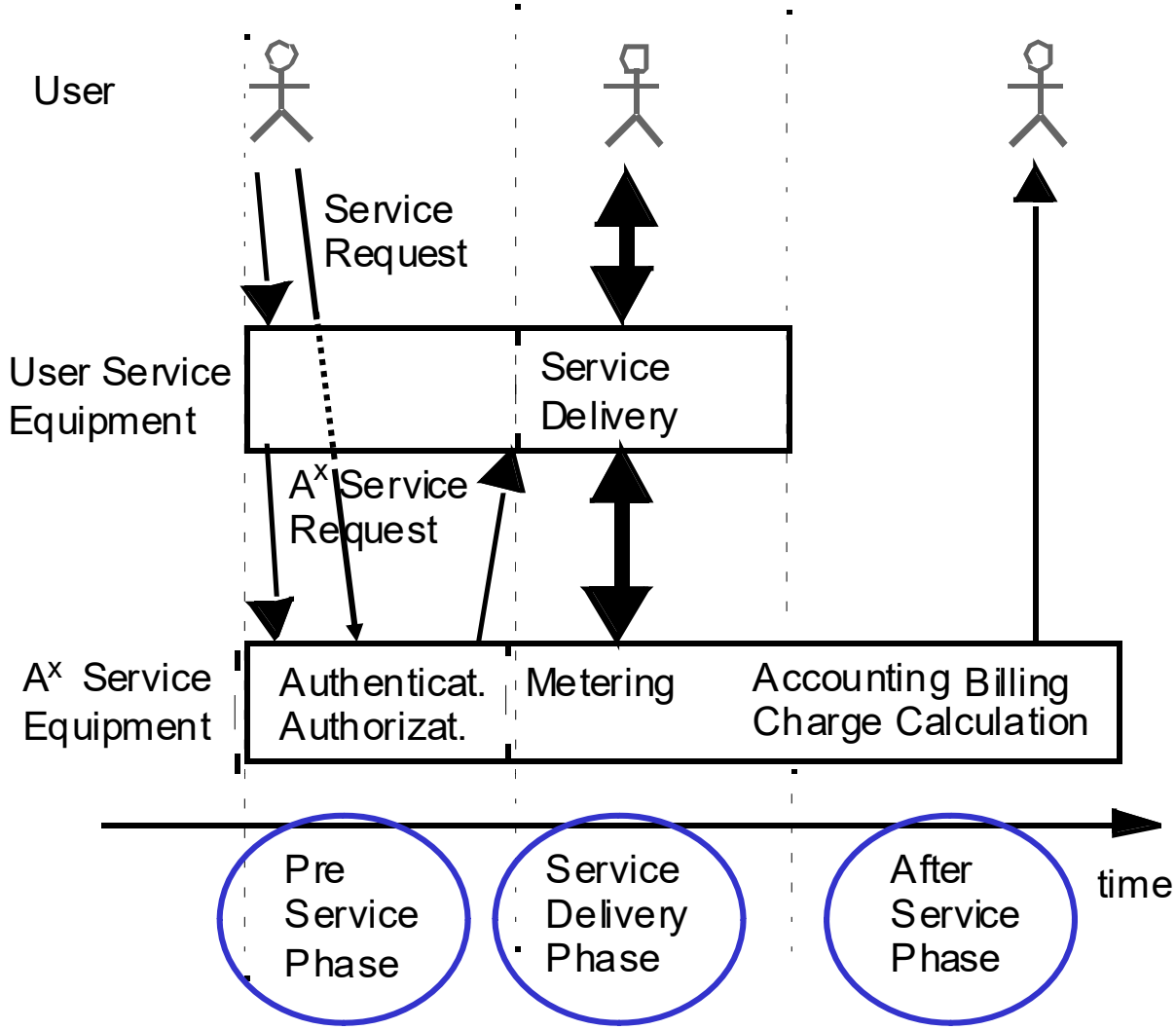
→ Horizontal structure leads to service classes with similar characteristics and similar A^x requirements.

A^x Model: Service Interactions

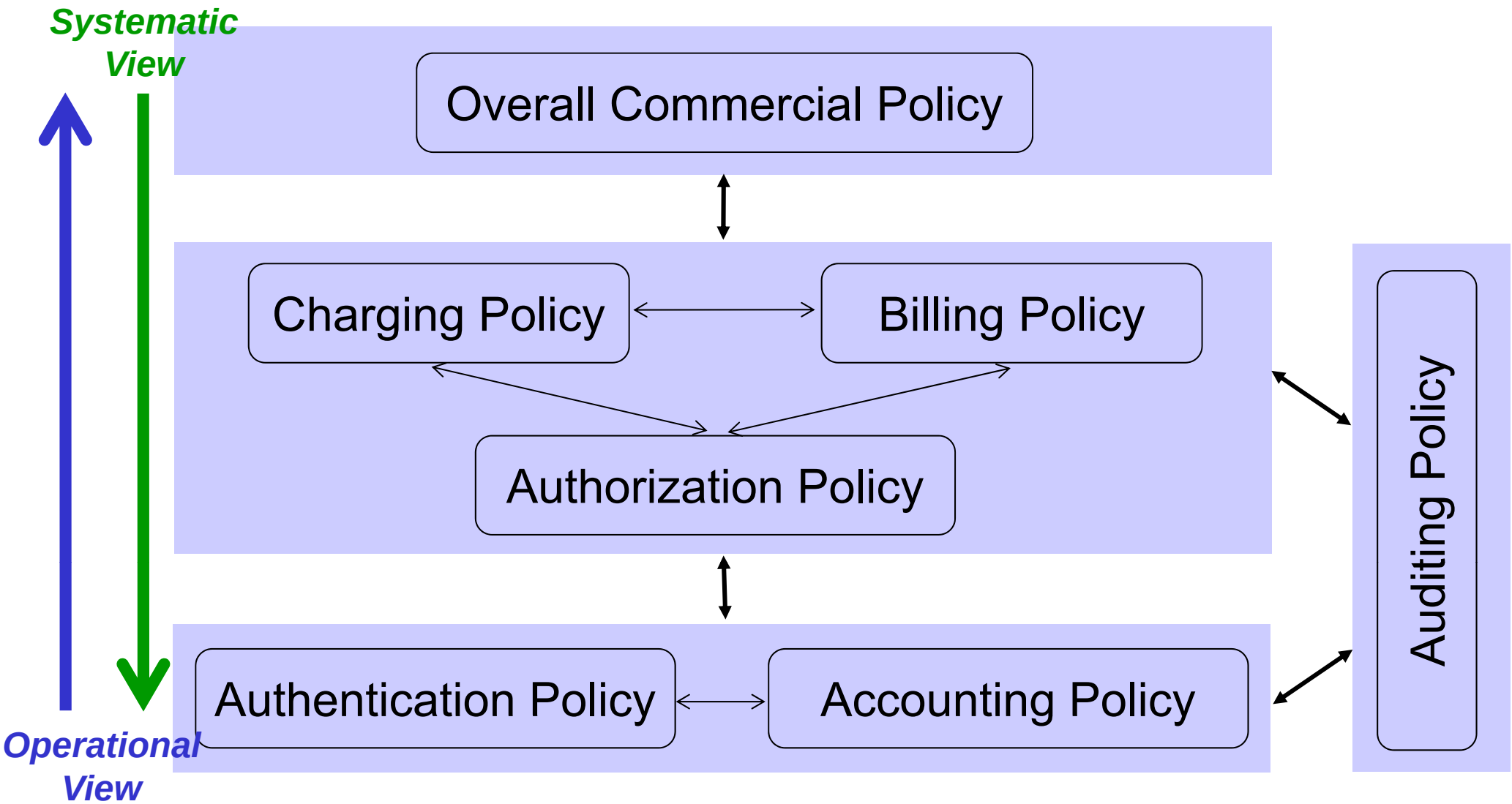
- Objective:
  - Support of multiple user services with configuration requirements
  - Generic A^x services

⇒ Logical separation

- Sequence of action:
  - Phases



Policy Model of the A^x Architecture



RADIUS (1)

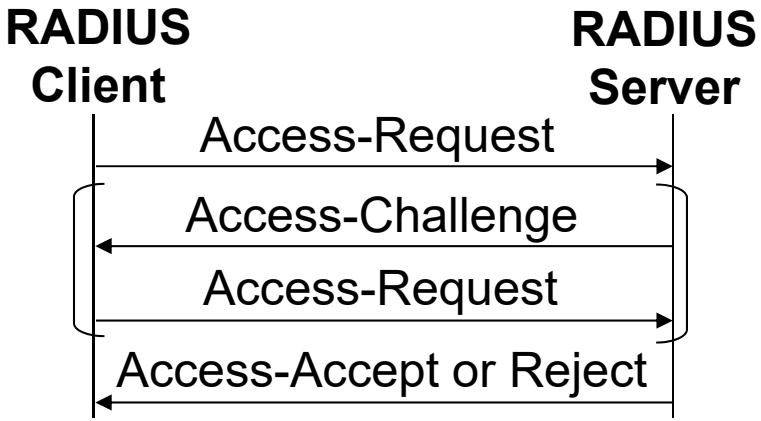
- RADIUS: Remote Authentication Dial In User Service
- Supports:
  - Authentication, authorization, and accounting mechanisms
- Client-server model, server can act as a client (proxy)
- Connectionless protocol based on UDP
- Uses attribute-value pairs:
  - Providing the ability to create vendor-specific extensions
- Applies the hop-to-hop security model
- Supports PAP and CHAP authentication via PPP
- Wide-spread deployment

PAP: Password Authentication Protocol
CHAP: Challenge Handshake Authentication Protocol
PPP: Point-to-point Protocol

[RFC 2865]

RADIUS (2)

- Messages for Access control (AA):
  - Access-Request
  - Access-Accept
  - Access-Reject
  - Access-Challenge



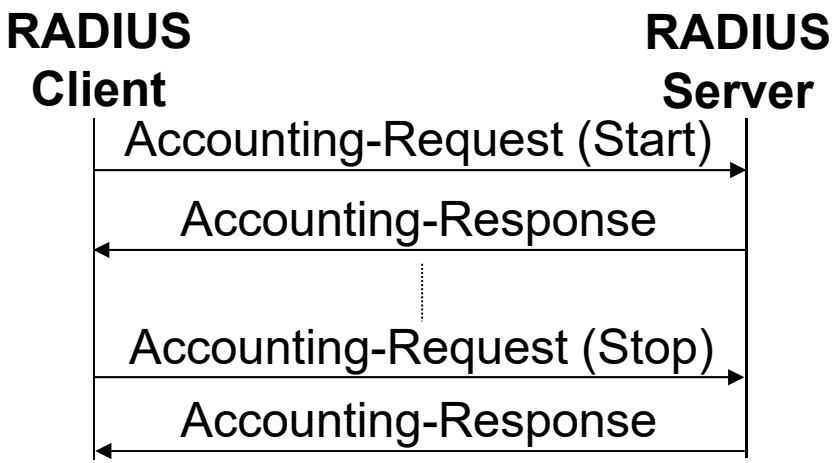
- Attribute Value Pairs
  - Value Types: Integer, Text String, IP Address, Date, Binary
  - Message size: Header size (12 bytes) + NrOfAttributes(0 ..N) * Attribute (3..255 bytes) with max 256 different attributes

Attribute Number	Attribute Length	Attribute Value

[RFC 2865]

RADIUS Accounting

- Messages for accounting:
  - Accounting-Request
    - Start
    - Stop
    - Interim-Update
  - Accounting-Response



- Accounting-specific attributes:
  - Acct-Session-Time
  - Acct-Input-Octets, Acct-Output-Octets
  - Acct-Input-Packets, Acct-Output-Packets
  - Acct-Session-ID
  - Acct-Terminate-Cause

[RFC 2866]

Security of RADIUS

- Client and server hold a shared secret
- Hop-to-hop security model
- Packet authentication based on MD5
- User password hiding based on the shared secret and the MD5 hash algorithm

- Vulnerabilities:
  - Attacks to get the shared secret
  - Weak request authentication
  - User password encryption

- It is recommended to use RADIUS over IPSec

Diameter (1)

- Diameter is another AAA protocol providing minimum requirements as defined in RFC 2989
- Diameter designed to overcome the deficiencies of RADIUS:
  - Different access technologies
  - Distributed security model
  - Multi-domain roaming scenarios
  - Server-initiated messages
  - Better error handling and reporting
  - Capability negotiation
  - Peer discovery and configuration
- Consists of session-oriented base protocol and different extensions (*e.g.*, for accounting)
- Reliable transport provided by use of
  - TCP
  - SCTP

[RFC 3588]

Diameter (2)

- Attribute-value pairs (AVPs) for data exchange
  - Extended data formats for AVP data types
  - Standardized set of AVPs is defined, but it is flexible for further extensions.
  - AVP format:

AVP Code	AVP Flags	AVP Length	Vendor-ID (optional)	Data

- Security support:
  - Hop-by-hop and end-to-end security
  - Support of IPSec and TLS
  - Auditability, data-object security mechanisms
- Messages for accounting:
  - Accounting-Request
  - Accounting-Answer
- Accounting Record types:
  - Event record
  - Start record, interim record, stop record

[RFC 3588]

Part 4: Multimedia Protocol Components

- Multi-resource Allocation (MRA)
  - Introduction and Motivation
  - Problem and Approach
  - Related Work
  - Metrics and Questionnaire
  - Simulations
  - Technical Approach
  - Conclusions

Thanks to Patrick Poullie, University of Zürich

Introduction

- Data centers apply virtualization to
  - *Homogenize* the hardware of nodes
  - *Isolate* users
  - *Multiplex* varying workloads efficiently

- Clouds are a special case of data centers, where resources are deployed by Virtual Machines (VM)
  - On-demand computing power for users
    - End-users and companies
  - Omnipresent
    - Deployed by all major and many minor companies
    - Private cloud computing also attractive

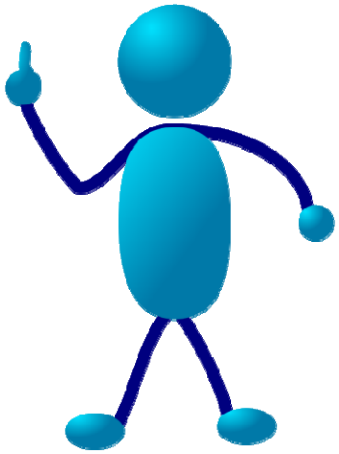
Illustration

User u_g executes arbitrary workloads

User u_r executes jobs consisting of tasks running inside VEs

Motivation

- Fairness problems in data centers have been pointed out frequently over the last years
  - Noisy neighbor problem causes low QoE



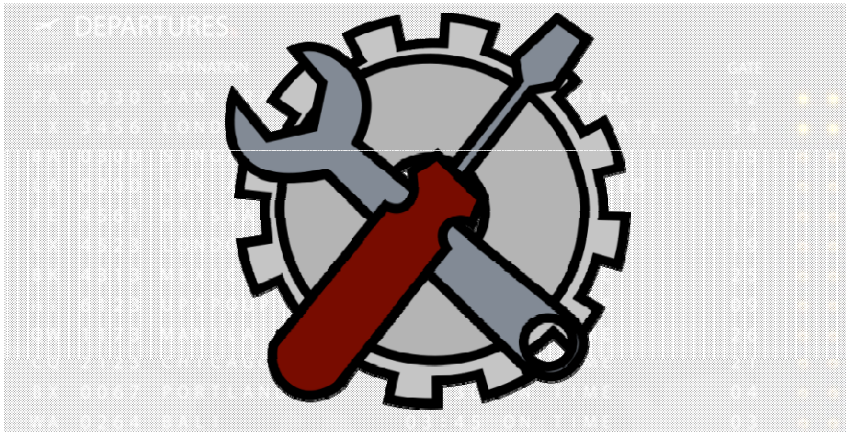
- Complex allocation problems
  - Multiple heterogeneous resources, e.g., CPU, RAM, Disk I/O, network access
  - Partitioned over multiple hosts
  - Consumed by multiple VMs per user
  - With varying demands over time
  - No access to utility functions

- QoS and QoE may differ greatly among users

Problem

- Private data centers/clouds do not prescribe resource allocation by SLAs
  - Fairness is an illusive concept
  - What is fair?
  - How to achieve good QoE for all users?

- Scheduling impels the traditional solution approach
  - Determine next VM to start
  - Insufficient when VMs run over long periods or change utilization
  - How to establish fairness and align QoS?

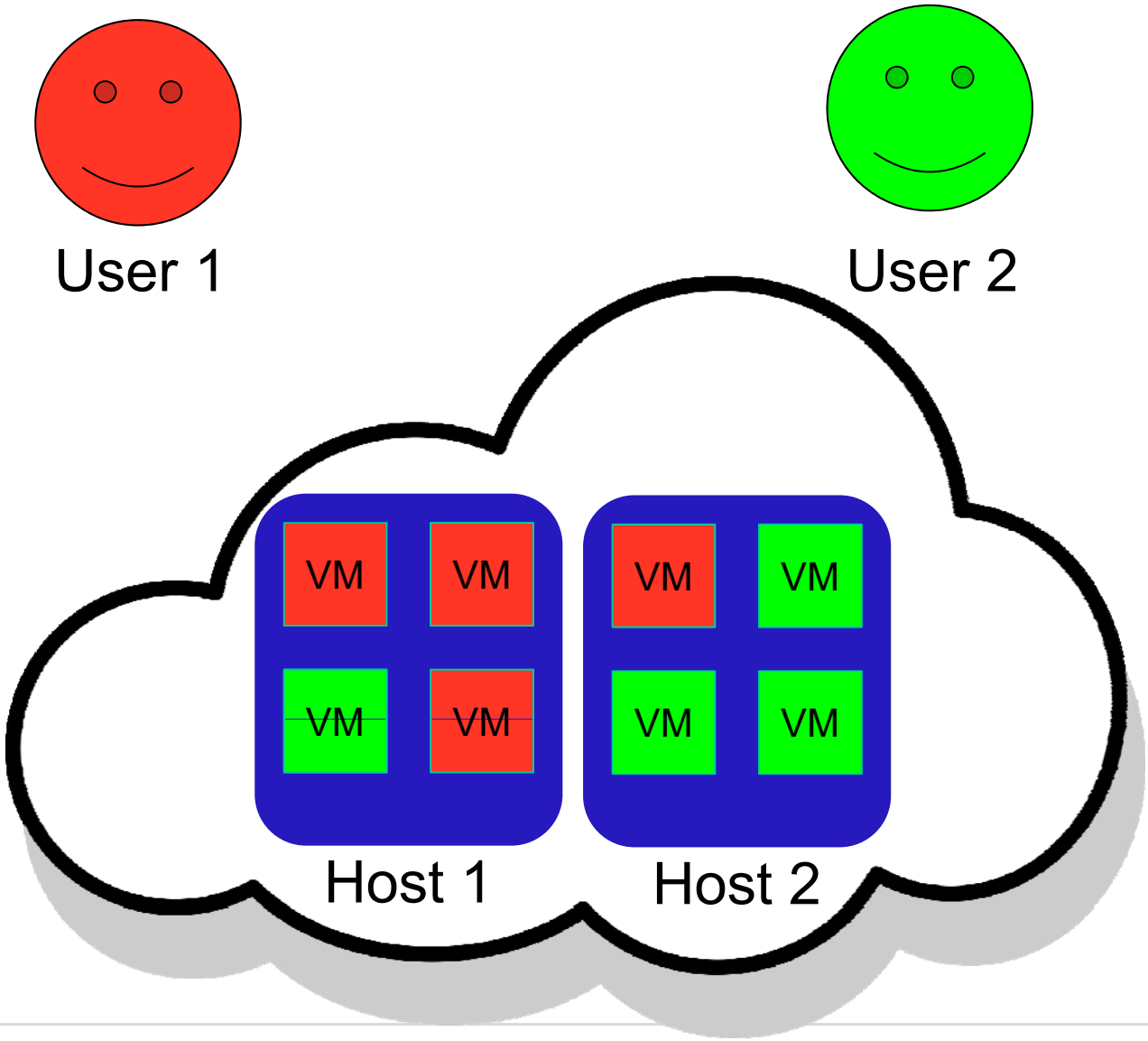


Approach

- Prioritize users inversely to their greediness
  - Greediness to be defined based on utilized resources
    - Avoids assumptions on utility functions (QoS-QoE dependencies)
    - Moves problem from defining fairness to defining greediness

- Handicap VMs of heavy users and prioritize VMs of light users
  - Hypervisor changes priorities during VM runtime
  - Prototypically implemented
  - VMs of light users receive higher QoS

Illustration



Related Work on Multi-resource Data Center Fairness

- Dominant Resource Fairness (DRF)
  - Simple assumptions on utility functions applied to scheduling
- Bottleneck Based Fairness (BBF)
  - Allocate every user at least equal share of one bottleneck
  - Not applicable across nodes as no metric defined
- Priority functions $\mathbb{N}_{\geq 0}^{m} \rightarrow \mathbb{N}_{\geq 0}$ to map *m*-resource-utilization to priority
  - Do not account for VM configuration
- Reciprocal Resource Fairness
  - Multiple complex mechanisms to enforce asset fairness in coalition of users

Utility Functions

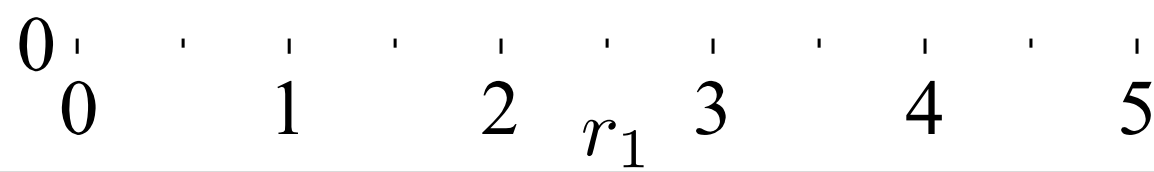
- Used in economics to model how resources are to be allocated to users
- Also used to model data center multi-resource allocation

  – Leontief

$$lt(x_1, x_2, \ldots, x_{\mathfrak{r}}) = \min\left\{\frac{x_1}{v_1}, \frac{x_2}{v_2}, \ldots, \frac{x}{v}\right.$$

  – Cobb-Douglas

$$cd(x_1, x_2, \ldots, x_{\mathfrak{r}}) = \prod_{j=1}^{\mathfrak{r}} x_j^{v_j}$$



Utility functions: QoS != QoE

- Resources a VM receives = QoS
- Performance of a VM = QoE
- To investigate dependency
  - Change virtual and physical resources a VM has access to
  - Measure QoE/performance by benchmark scores
  - Measure utilization of other resources
    - Via qemu process
    - RAM utilization measured via proportional set size

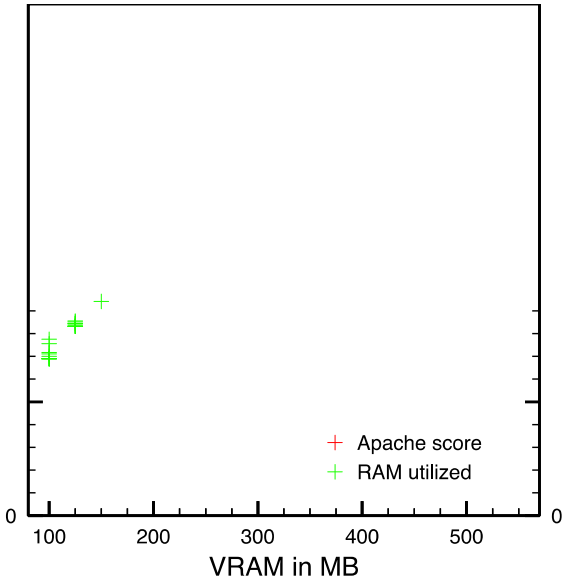
- Host: 2.5GHz AMD Opteron 6180 SE processor with 24 cores and 6 and 10 MB of level 2 and 3 cache, respectively, and 64 GB of ECC DDR3 M with 1333 Mhz
- VM and host have a x86-64 architecture and run Ubuntu 14.04.2 LTS, Trusty Tahr

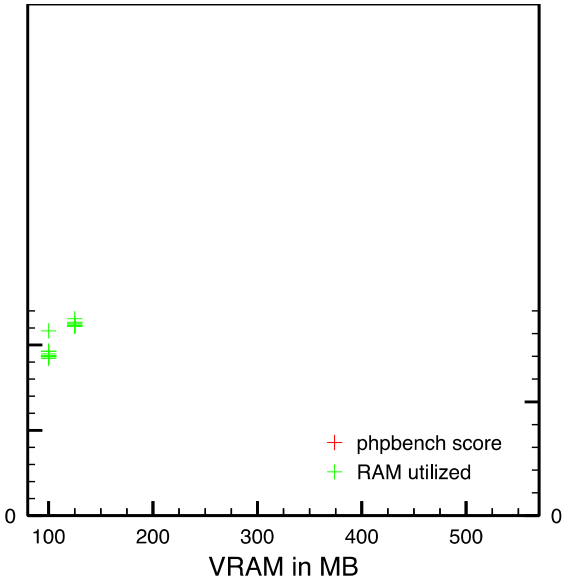
Benchmarks

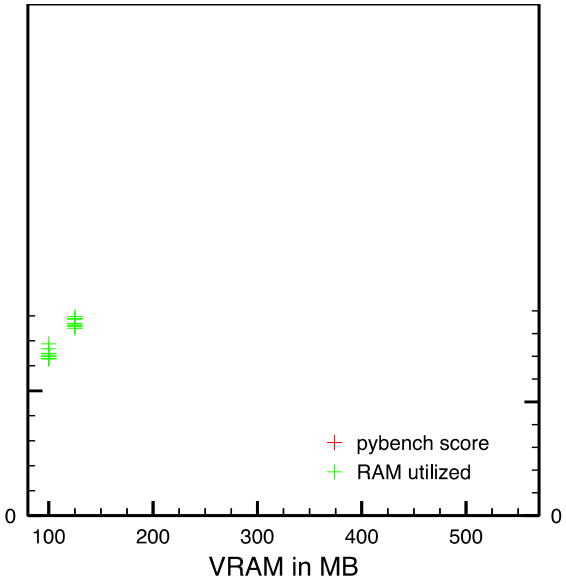
- Benchmarks represent a broad range of applications
- Apache
  - Globally most popular webserver software
- NGINX
  - Globally second most popular webserver software
- AIO-stress
  - VM's permanent storage often cached in host's RAM
  - Permanent storage I/O (hard disk or solid state drive) throughput
- Pybench
  - One of the most popular scripting languages
- PHPbench
  - Older websites may be based on PHP
- 7zip
  - Popular compression tool

RAM Effect on QoE (1)

- VM utilizes more RAM, if available, while QoE/performance increases only marginally or not at all

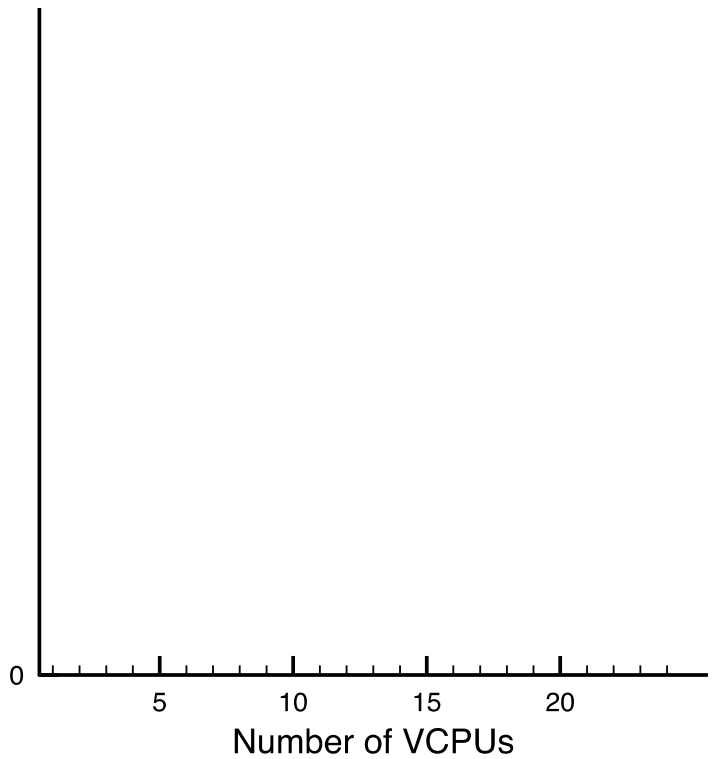


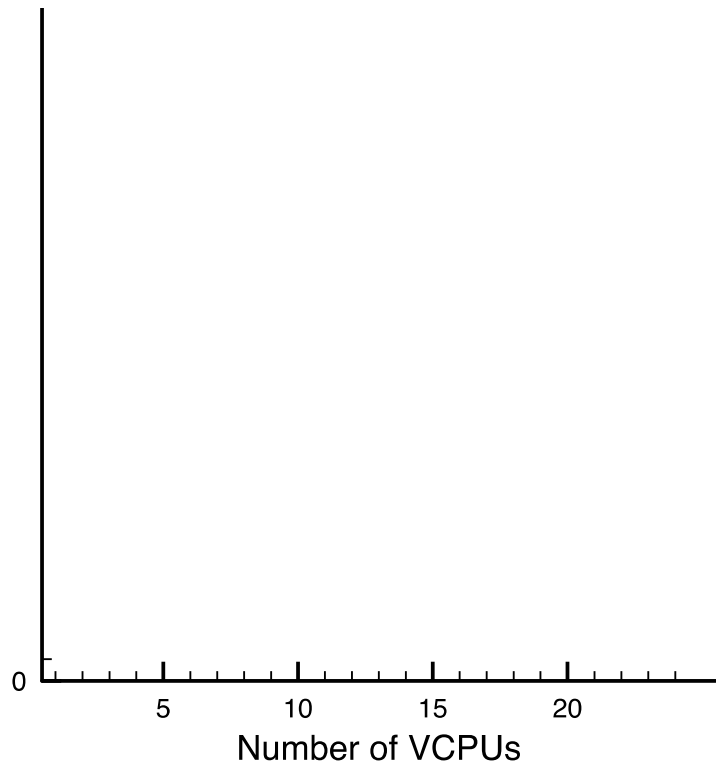




RAM Effect on QoE (2)

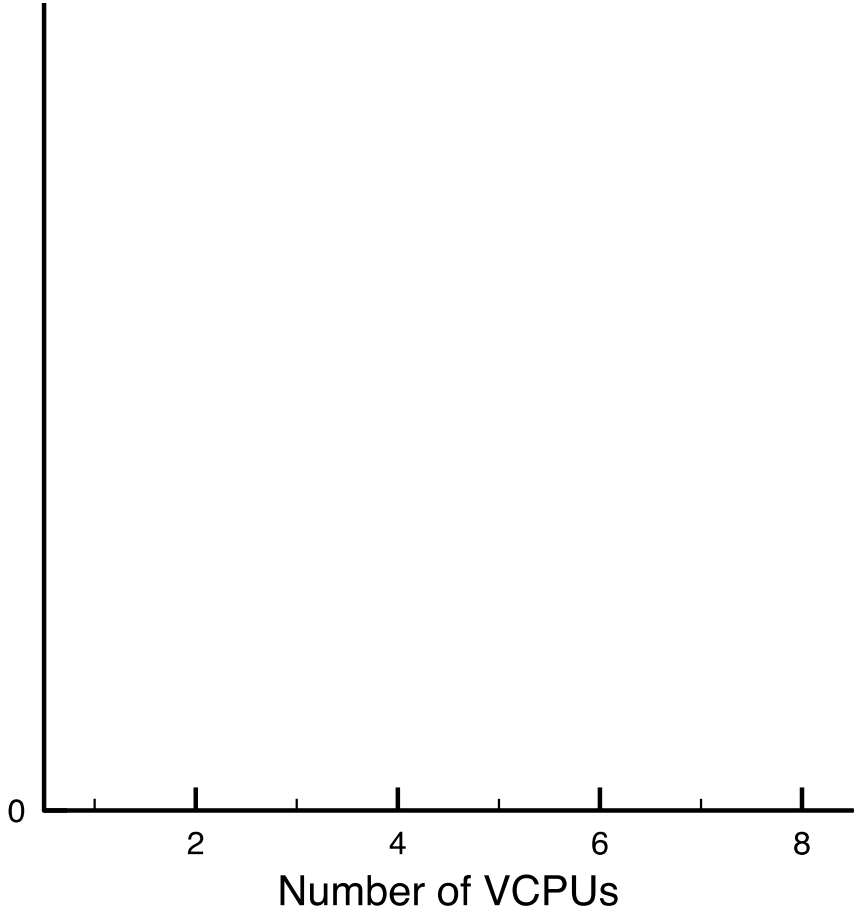
- VM performance/QoE differs only slightly despite great difference in (V)RAM
- VM with 9GB VRAM utilizes significantly more RAM than VM with 1GB VM is able to utilize

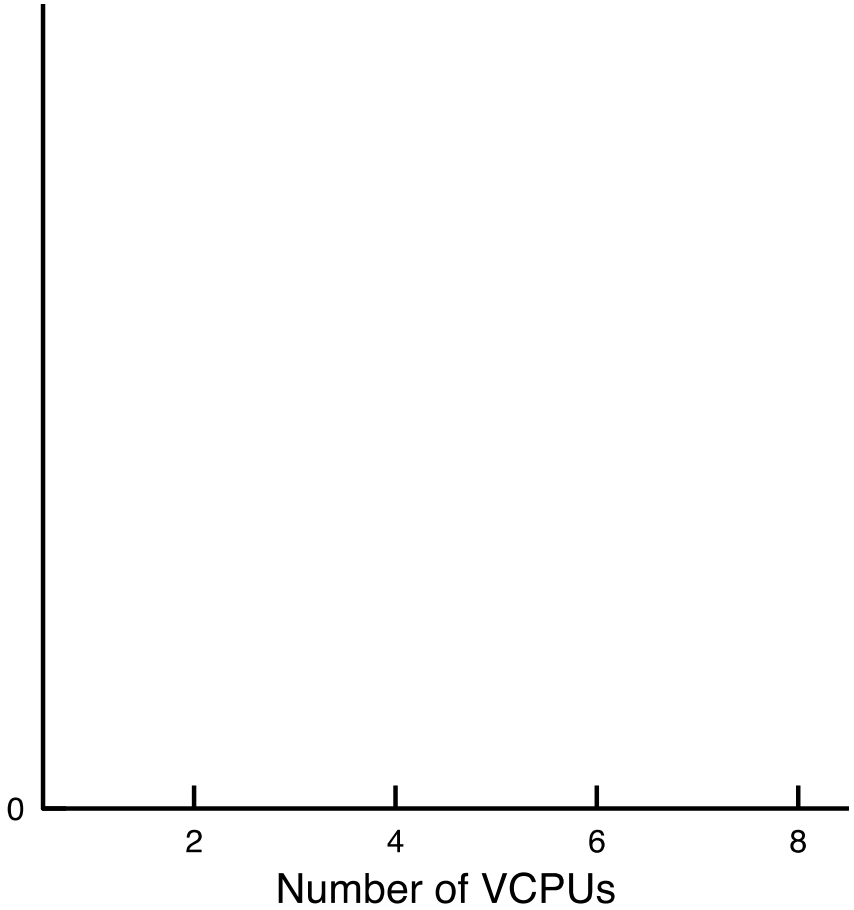




VCPUs Effect on QoE (1)

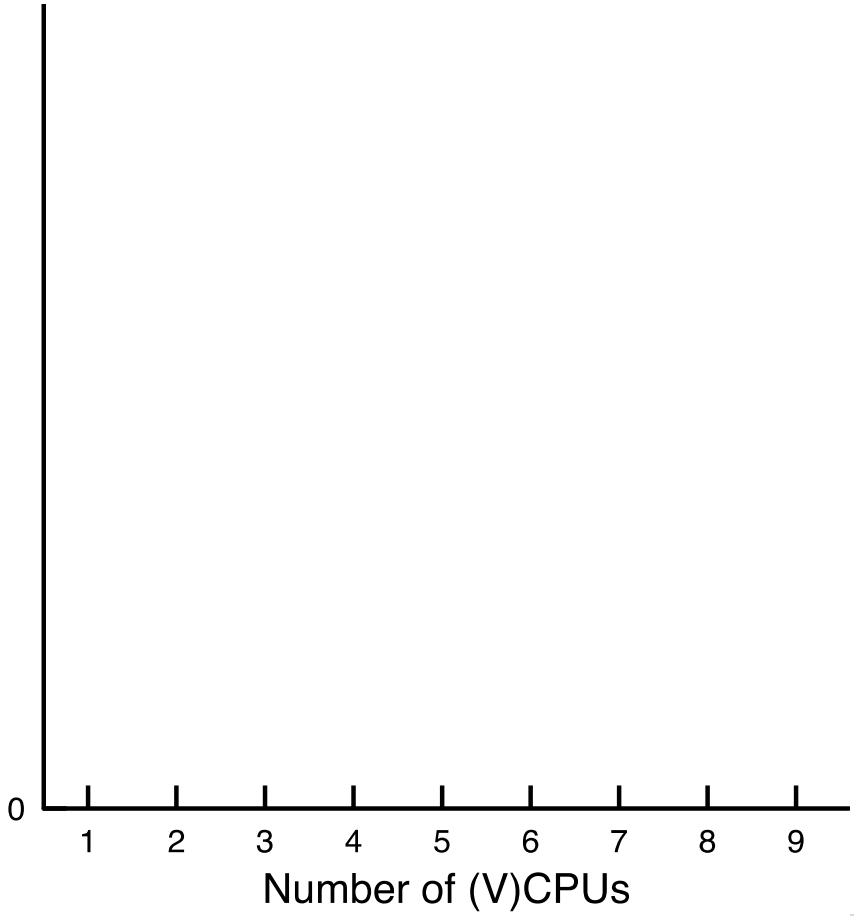
- More VCPUs are beneficial for single-threaded workloads
  - when host is under stress

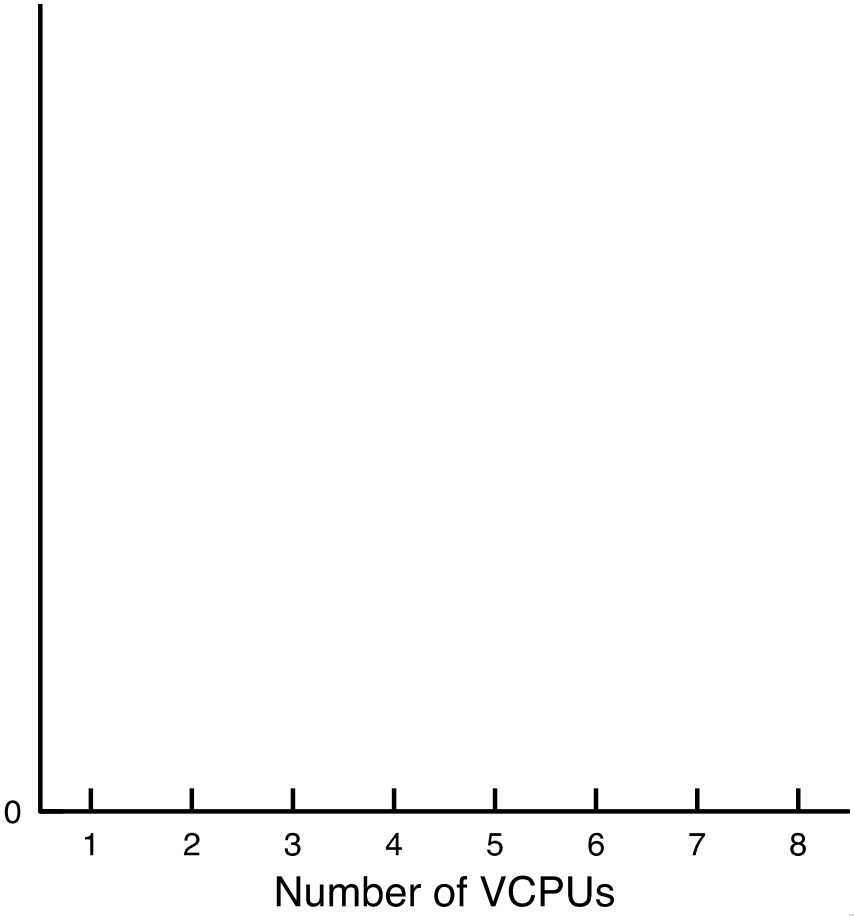




VCPUs Effect on QoE (2)

- More VCPUs harm multi-threaded workloads





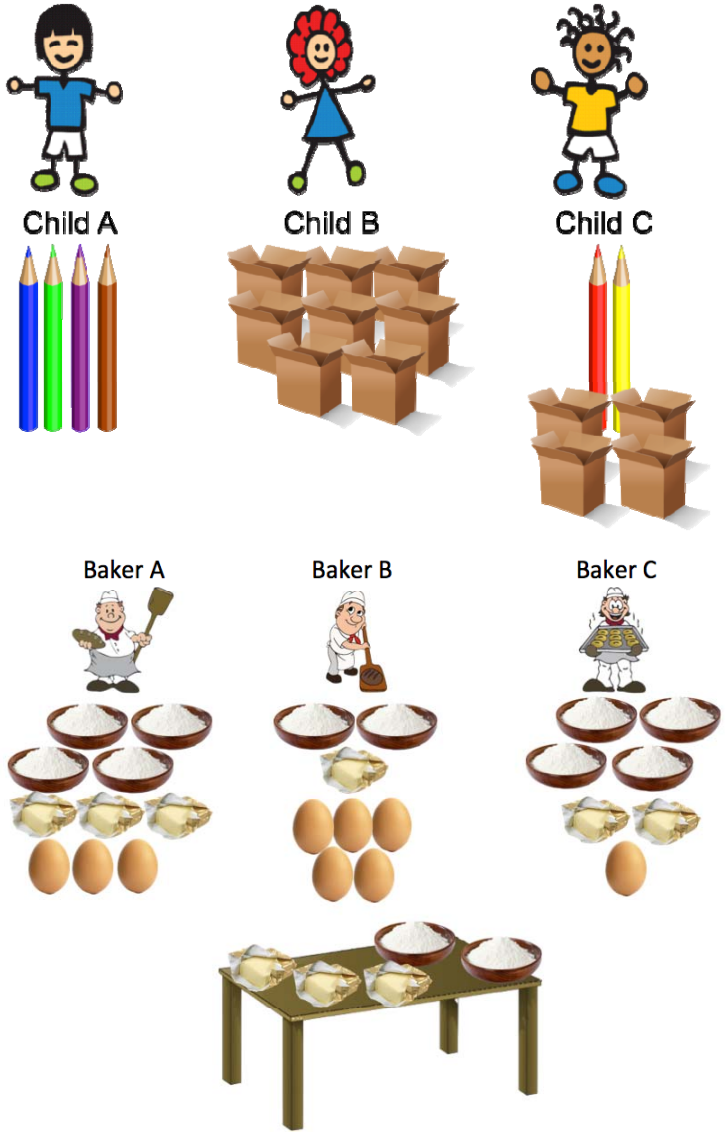
Lessons Learned

- No significant effect of VRAM on VM performance
  - Even if VRAM greatly constrains the utilization of RAM
  - If lack of RAM affects performance, it is not in proportion to the lack
- Lack of RAM bandwidth has significant effect as shown by BA
- Additional VCPUs often decrease performance of multi-threaded workloads
  - single-core benchmarks profit from additional VCPUs, when host is stressed
- VCPUs may increase RAM usage, but not necessarily
- Assumptions on utility functions made in literature are too simplistic
  - Assumptions should be avoided
  - Hard to define relationship between QoS and QoE
- How to define fairness without utility functions?
  - By max-min fairness!
  - But how to compare bundles?

Questionnaire

- Goals
  - Gain insights on an intuitive understanding of fairness
  - Evaluate existing heaviness metrics
  - Justify empirically GM as heaviness metric

- No particular target group
- No information about utility functions
- Intuitive and real-life scenarios
- 604 questionnaires completed out of 702 starts



Metrics in Question

- Resource supplies are normalized to 1
- a_{ij} denotes what consumer c_j is allocated of resource r_i
- Price $P(c_j) : \sum_{r_i \in R} a_{ij}$

y $\left(\quad {}_j \right) \quad \sum_{r_i \in R} \left(\quad {}_{ij} \quad \sum_{c_l \in C} \quad {}_{il} \right)$

$${}_{j \in} \left(\quad \left\lfloor \quad {}_{l \in} \quad \right\rfloor \right)$$

$$\left(\quad {}_j \right) \qquad {}_{j \in} \qquad {}_j$$

Greediness Metric (1)

- Example

Greediness Metric (2)

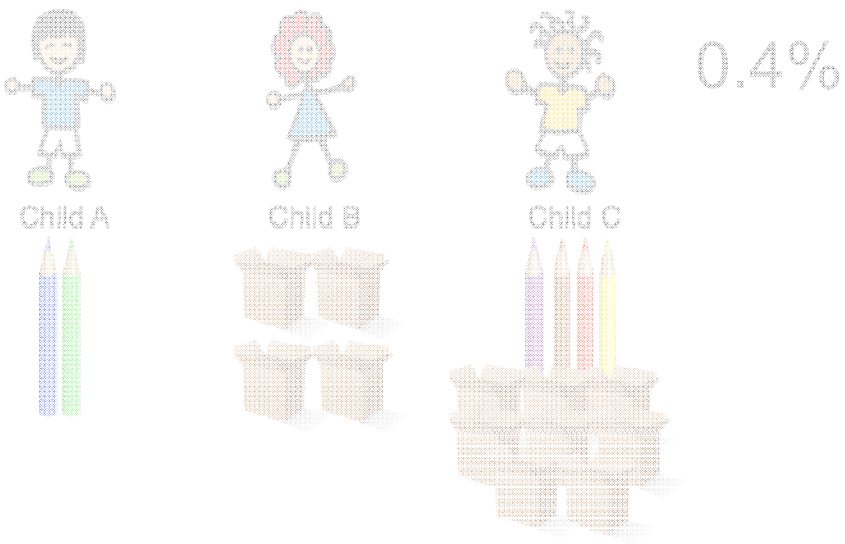
- If all consumers are equal $e_{ij} := \frac{1}{|C|}$ for all $c_j \in C$
  - In clouds, e is determined by VMs' virtual resources

- $g(c_j) := \sum_{r_i \in R} e_{ij} + (a_{ij} - e_{ij}) \cdot$

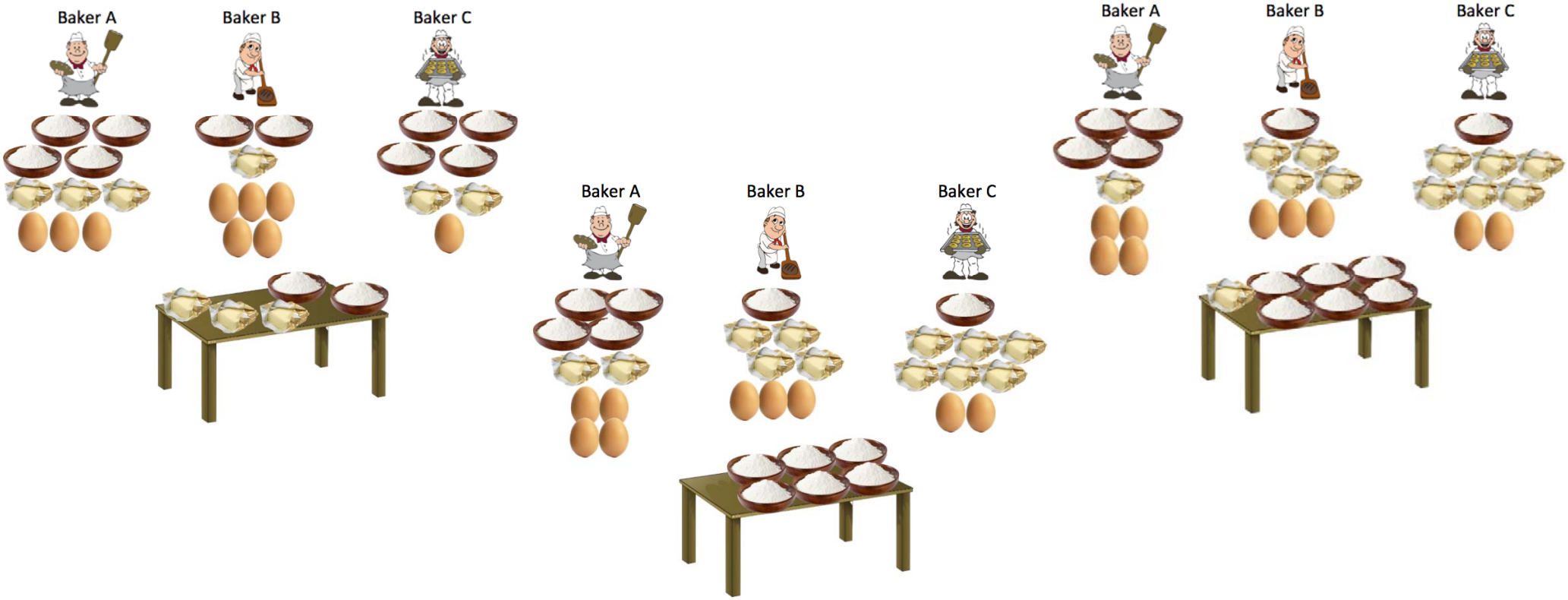
$$\begin{cases} 1 & \text{if } a_{ij} \geq e_{ij}, \\ \overbrace{\min\left(1, \frac{\sum_{c_k \in C} \max\left(0, a_{ik} - e_{ik}\right)}{\sum_{c_k \in C} \max\left(0, e_{ik} - a_{ik}\right)}\right)}^{\text{Dynamic Ceding Factor (DCF)}} & \text{else.} \end{cases}$$

overcommit factors

Leontief Scenario

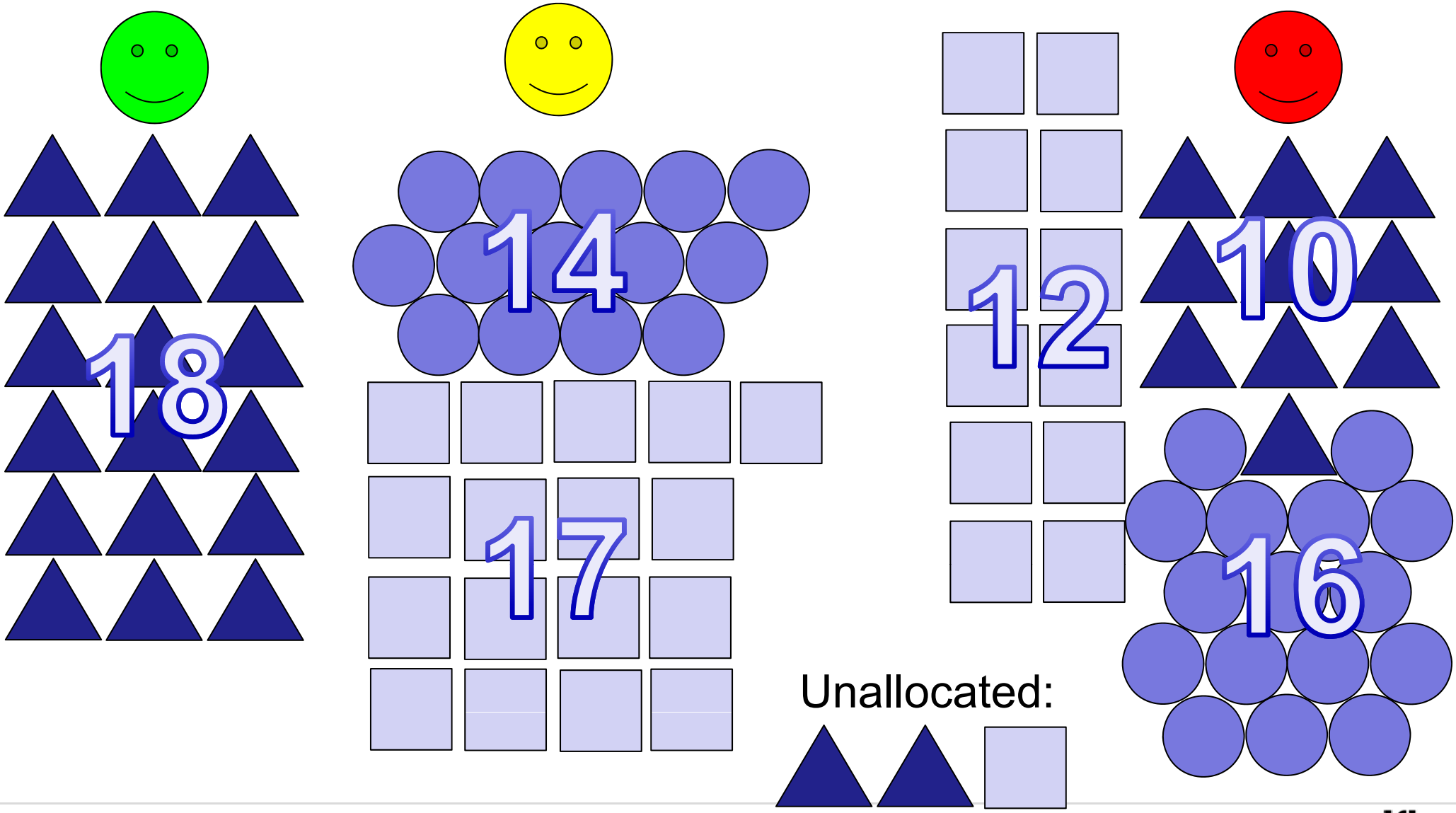


Self-serving Scenarios



Scenario 1	
52.7%	B,A,C
38.0%	A,B,C
6.5%	A,C,B

Distinct Scenario



Simulator

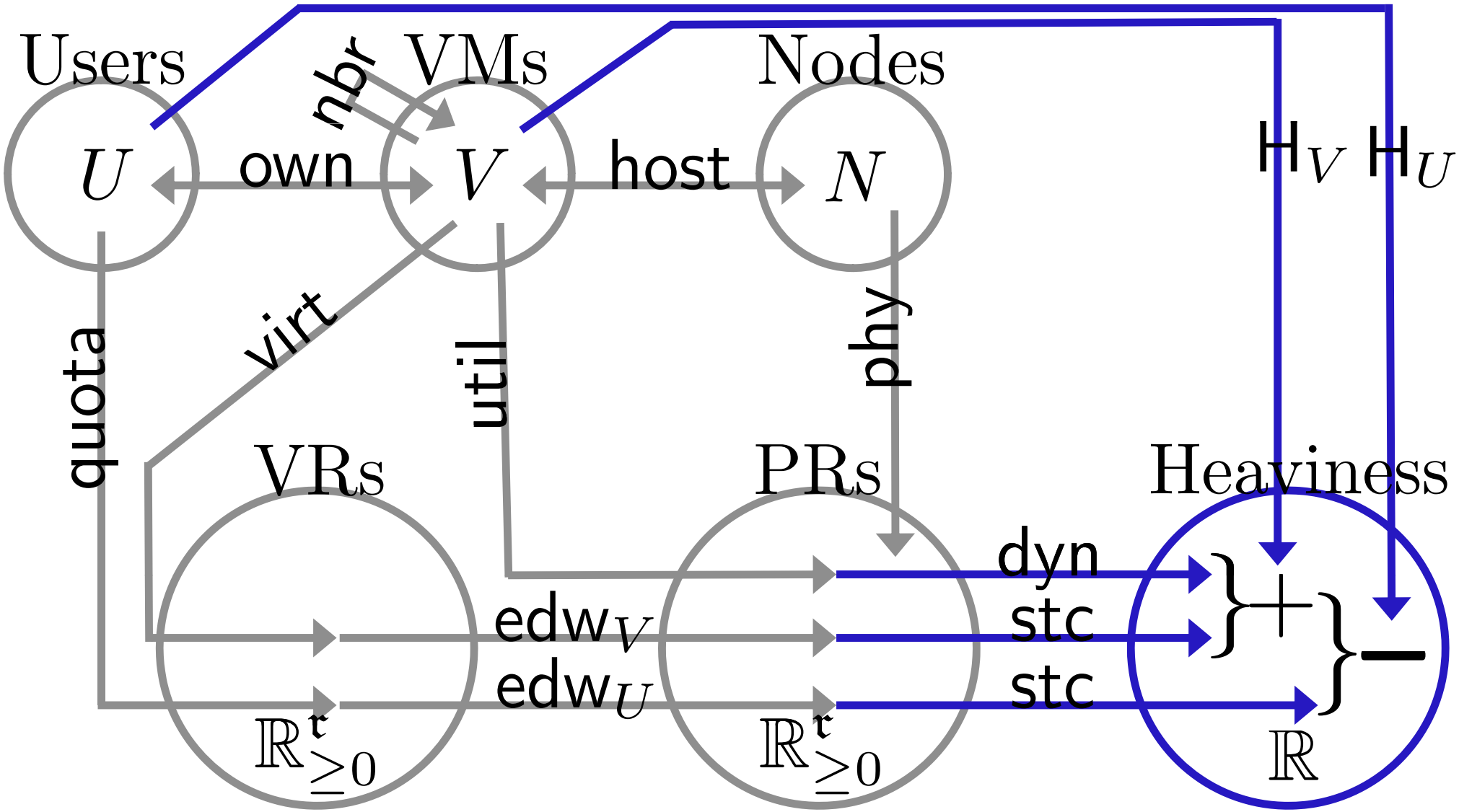
- Heaviness of a user is sum of heaviness of the user's VMs
  - Heaviness defined by the following metrics $\mathsf{Met} := \{\mathsf{A}, \mathsf{D}, \mathsf{G}^0, \mathsf{G}^{0.5}, \mathsf{G}^1\}$
  - Setup is a set of users, nodes and VMs
- There is a fixed number of resources
- Each VM has a requirement vector over these resources
  - That is, the ration in which the VM is allocated these resources
- The simulator allocates resources to VMs such that
  - Max-min fairness among heaviness of users (global) **XOR** VMs (local) is enforced
  - The allocation is Pareto-efficient
- The allocation is a **snapshot**
  - No change over time
- VM that is owned by user u_a and runs on node n_x is denoted by v_x^a

Simulator

- Heaviness of a user is sum of heaviness of the user's VMs
  - Heaviness defined by the following metrics $\mathsf{Met} := \{\mathsf{A}, \mathsf{D}, \mathsf{G}^0, \mathsf{G}^{0.5}, \mathsf{G}^1\}$
  - Setup is a set of users, nodes and VMs
- There is a fixed number of resources
- Each VM has a requirement vector over these resources
  - That is, the ration in which the VM is allocated these resources
- The simulator allocates resources to VMs such that
  - Max-min fairness among heaviness of users (global) **XOR** VMs (local) is enforced
  - The allocation is Pareto-efficient
- The allocation is a **snapshot**
- Fairness is determined by different metrics:

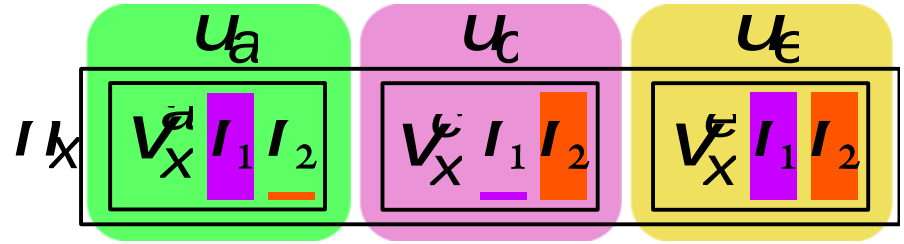
$$\mathbb{R}^{\mathfrak{r}\times\mathfrak{u}} \xrightarrow{\mathsf{M}\in\mathsf{Met}} \mathbb{R}^{\mathfrak{u}} \xrightarrow{\text{Jain's Index}} [0,1].$$

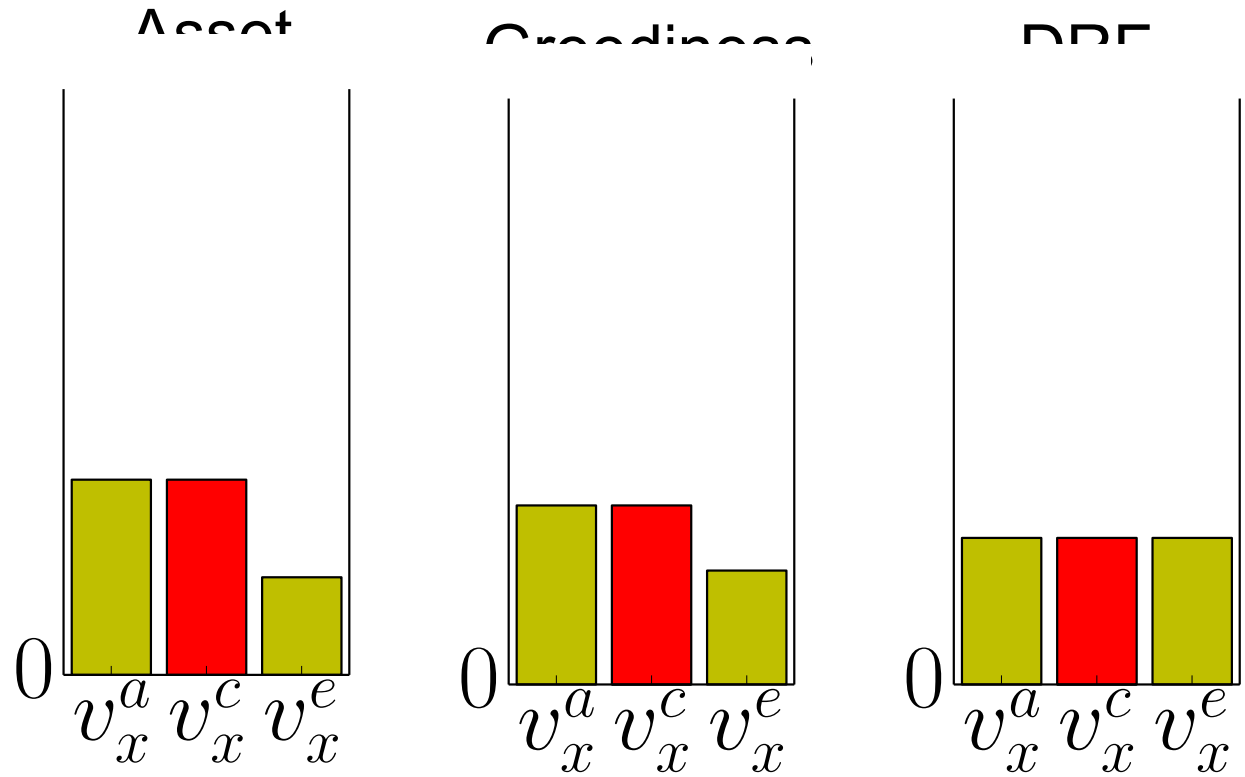
Illustration



Setup 1

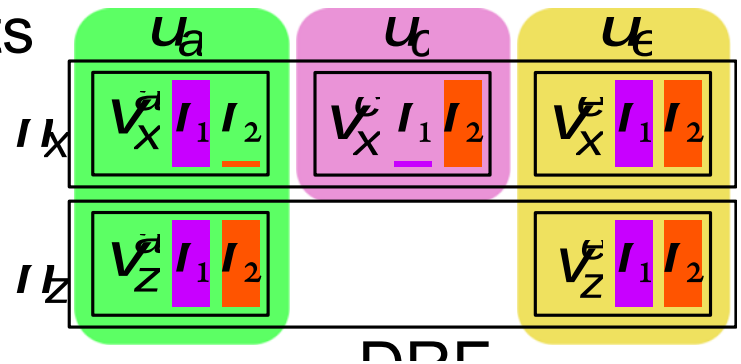
- Scope irrelevant
- Greediness allocation is between the Asser and DRF allocation

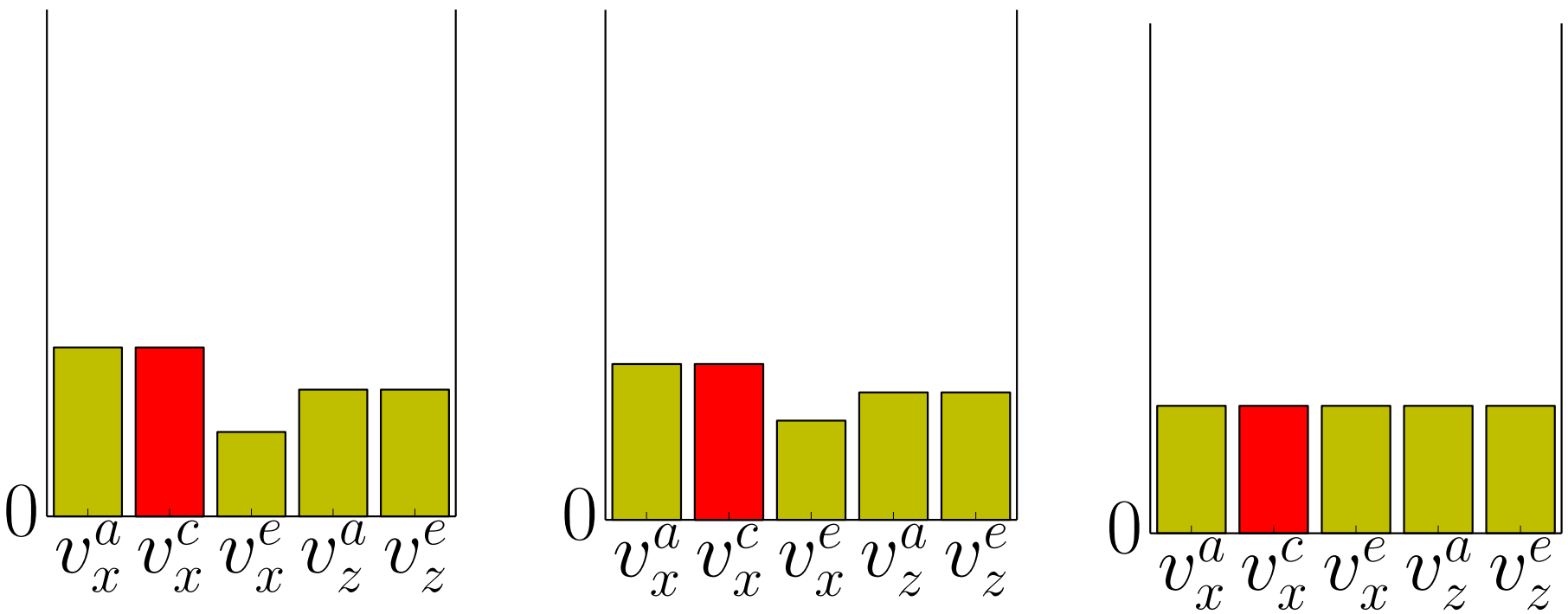




Setup 2: Local Policies

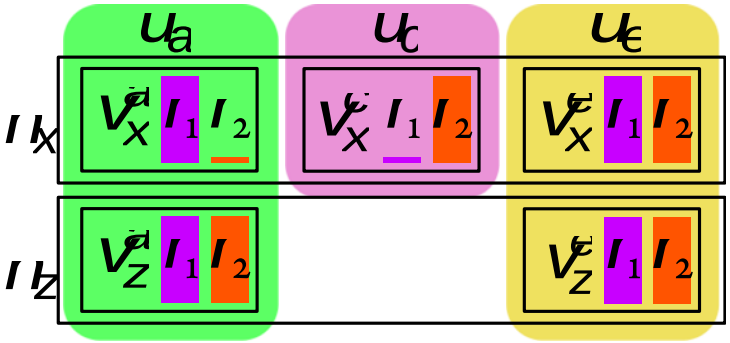
- Setup 1 is extended by a node that hosts two VMs that request both resources equally

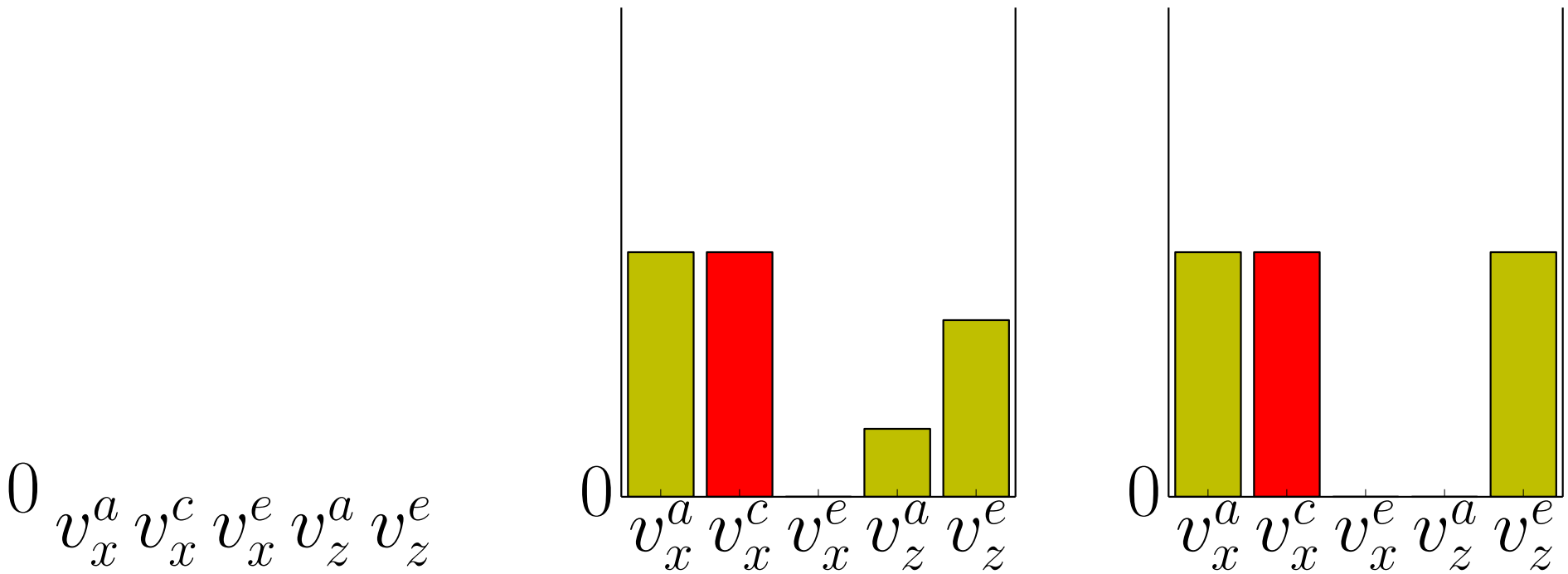




Setup 2: Global Policies

- Greediness allocation is between the Asset and DRF allocation





Setup 3: Incentives

- All seven VMs utilize resources
  in ratio (2,1)

- Indices encode configuration

- Illustrated allocation is GM-fair
  - All VMs have same greediness

VMs

Incentives

- All seven VMs utilize resources
  in ratio (2,1)

- Indices encode configuration

- Illustrated allocation is GM-fair
  - All VMs have same greediness

VMs

Lessons Learned

- Fairness has to be tackled on a per User (not per VM) basis and this task has to be coordinated among nodes
- Greediness allocations are between the DRF and Asset allocations
- Contrary to DRF and Asset fairness, Greediness fairness rewards correct configuration of VMs

GM-fairness and Scheduling

- GM-fairness also applicable to VM scheduling

- State-of-the-art of fair data center resource allocation is DRF and
  - Provides sharing incentive
  - Is Pareto efficient
  - Is strategy proof
  - Ensures envy freeness

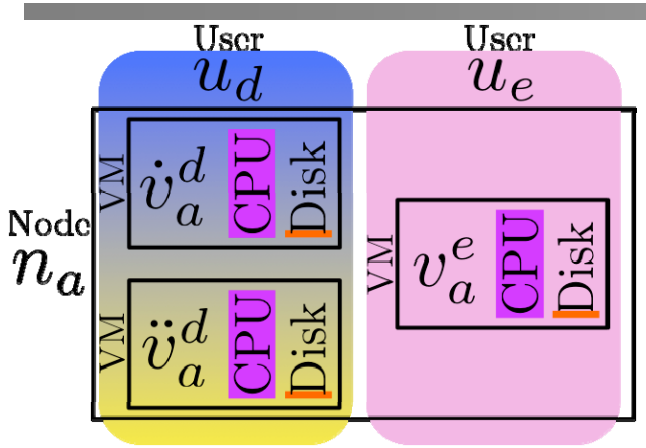
- Under the assumptions of DRF (Dominant Resource Fairness), GM-fairness also achieves these properties

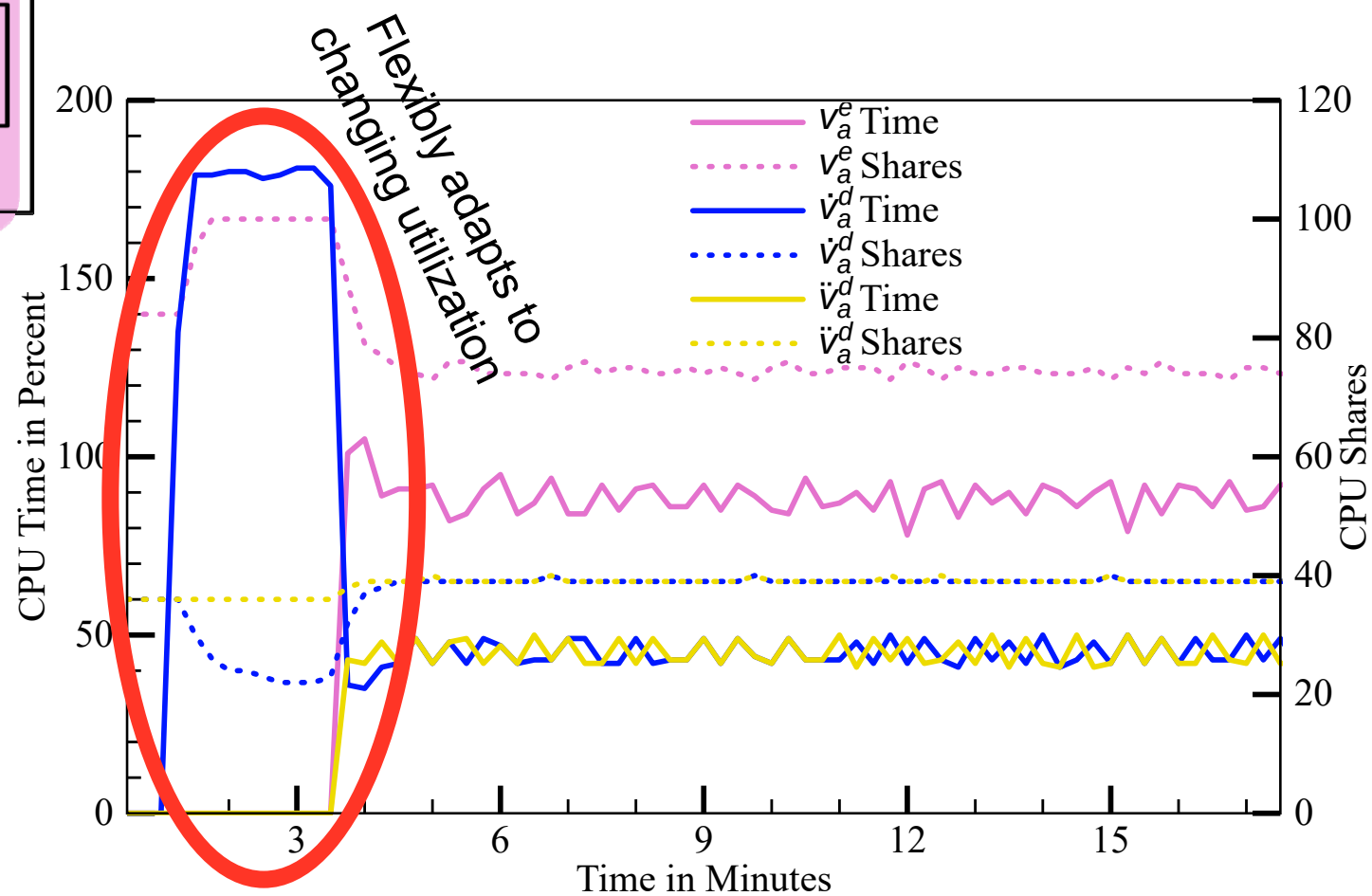
OpenStack VM Prioritization

- OpenStack and KVM resource monitoring and control
  - Fairness enforcement during runtime technical feasible

- OpenStack framework
  - Nova-compute extension enforces fairness according to different heaviness metrics

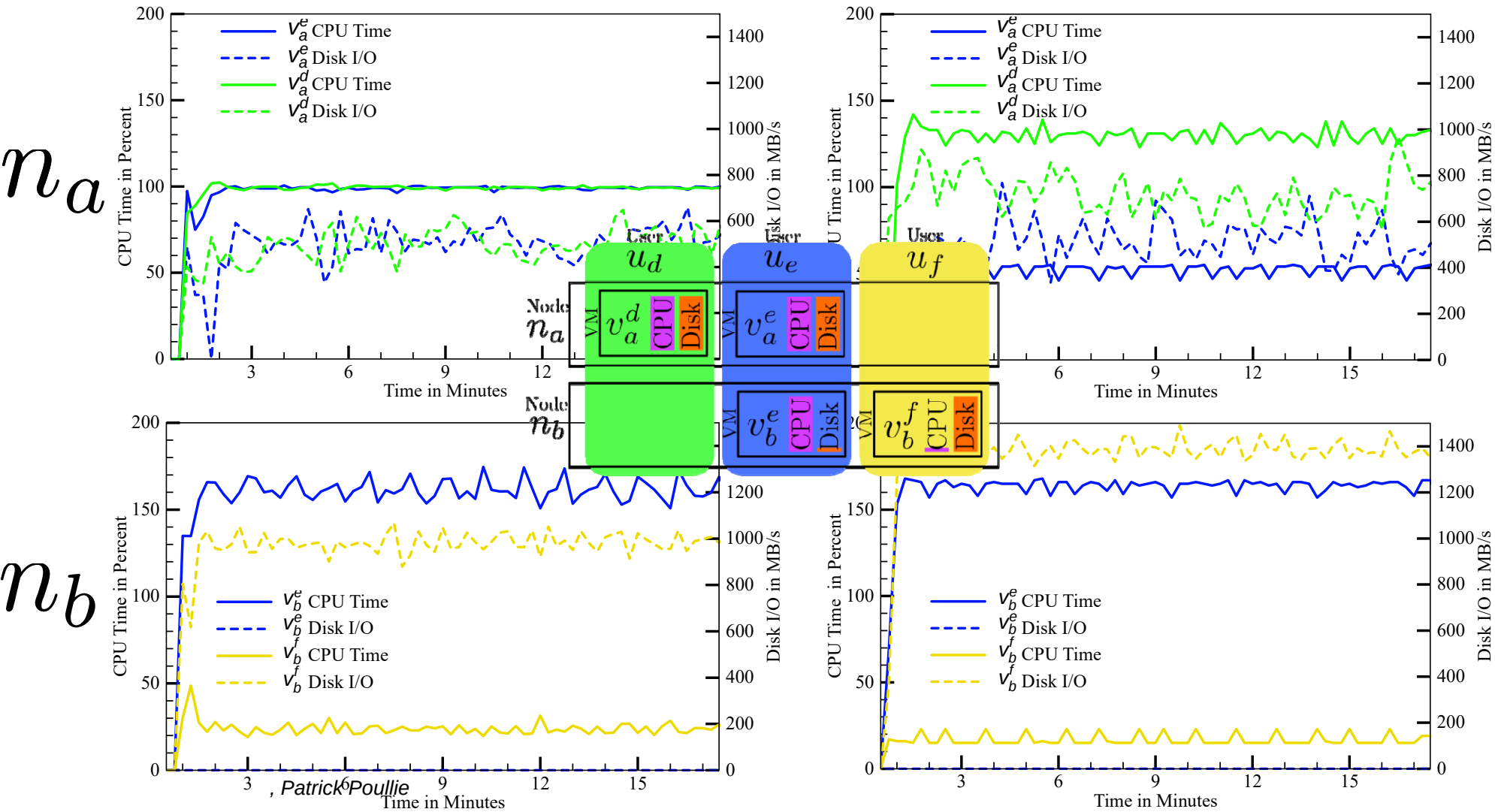
Practical Evaluation (1)





Practical Evaluation (2)

Without nova-fairness With nova-fairness



Key Take Aways

- Greediness Metric (GM)
  - GM captures an intuitive understanding of fairness
    - Without access to utility functions
    - Perfectly complies with questionnaire results
  - GM provides incentive to configure VMs correctly
  - GM allocation lay between Asset and DRF allocations
  - GM fairness successfully implemented by OpenStack service

- Runtime prioritization instead of scheduling
  - Runtime prioritization technically feasible and more effective
  - Resource dependencies too complex and fluctuant to make assumptions on utility functions
  - Allows to better control QoS and QoE

Cloud Flat Rates

- Users pay monthly price for a fixed quota
  - Quota can be increased by higher payment
  - Instantiated VMs are limited by this quota
- Nova-fairness runs in the background
  - Heavy users receive more resources
    - Receive lots of “best effort” QoE
  - Light users receive resources with high priority
    - Reliable QoE
  - Heavy users do not impede light users
- Much simpler charging scheme than per-hour-per-VM scheme offered by today’s commercial cloud providers

Conclusions

- What is (data center) fairness?
  - Greediness Metric allows to define an intuitive understanding of cloud applicable fairness
    - Without access to utility functions, as complex QoS-QoE dependences exist
    - provides incentive to configure VMs correctly
  - DRF is not intuitively fair
  - Light users receive reliable QoE, heavy users best effort QoE

- How to establish fairness?
  - By runtime prioritization
    - technically feasible
  - Allows to better control resources than scheduling

Part 4: Multimedia Protocol Components

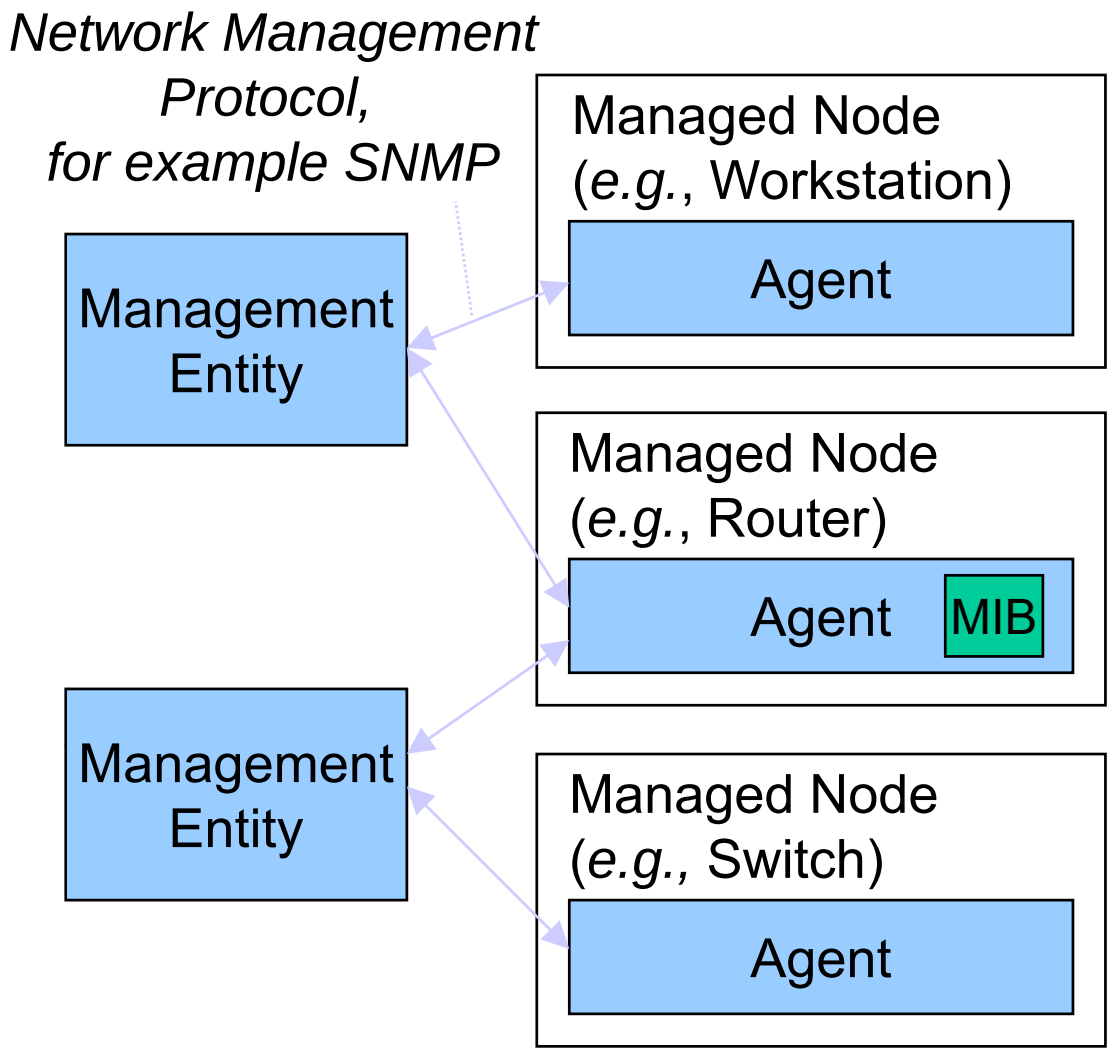
- Network Management
  - Areas
  - Components
  - Internet network management
  - ASN.1
  - SNMP

Network Management Areas

- Fault Management
  - Recording, detection, and reaction on errors
- Configuration Management
  - Detection and configuration of existing, accessible devices
- Accounting Management
  - Specification, recording, and control of resources and the access to them
- Performance Management
  - Determination, measurements, reporting, analysis, and control of performance, *e.g.*, throughput, load, delay, or error rates achieved
- Security Management
  - Access control to resources based on pre-determined policies
  - Key management and distribution

- These areas are termed FCAPS.

Network Management Components (1)



MIB: Management Information Base

Network Management Components (2)

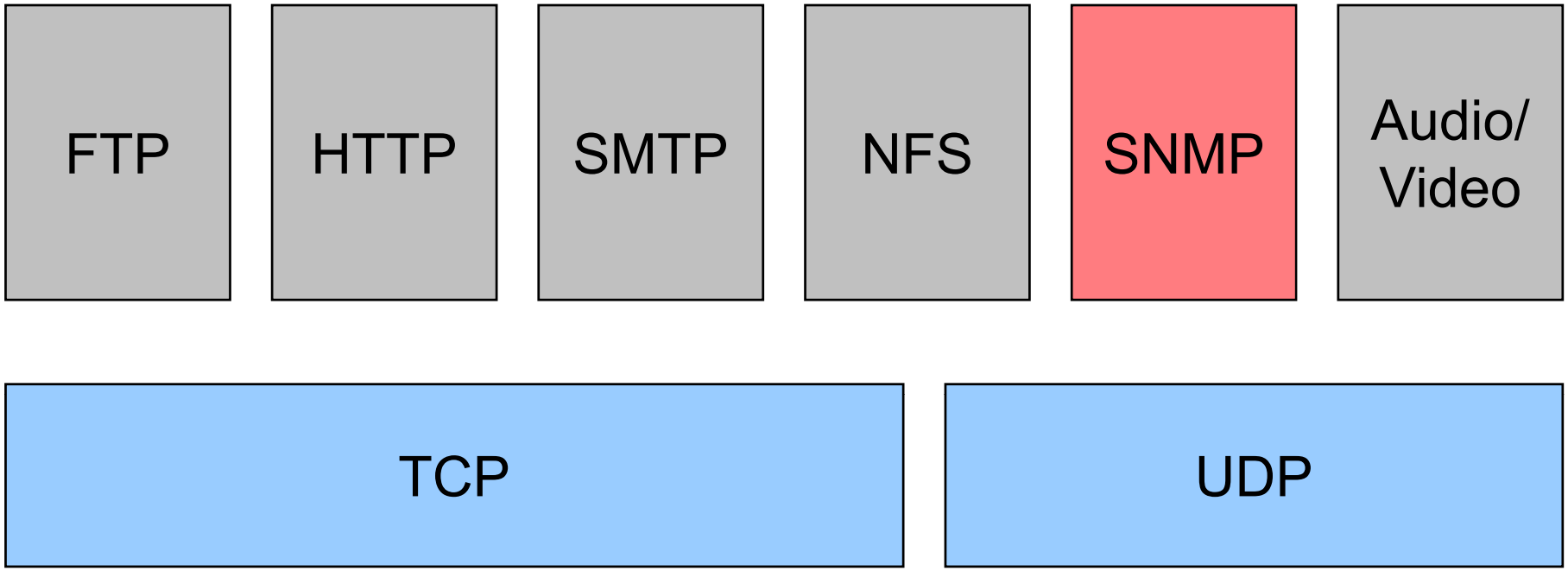
- Management Entity
  - Interactive application typically located at a central management station
  - To collect, handle, analyze, and report of management information
- Managed Node
  - Software or hardware components within the administered „network“, *e.g.*, end-system, router, or modem
  - Managed objects
    - Hardware or software objects within a node covering relevant configuration parameters and further informationen
    - Management Information Base (MIB) contains a collection of relevant configuration parameters and related information in a uniquely addressed form
  - Agent
    - Prozess, der mit der Management-Instanz über ein Netzmanagement-Protokoll kommuniziert und lokale Aktionen auf dem Gerät ausführt
- Network management protocol
  - Required for state reports and information of events

Internet Network Management

- Modular architecture
  - Network management objects (MIB object)
    - Collection of objects in terms of a virtual information space
    - Collocation into MIB modules
    - Examples: Counter of data packets discared, software version number
  - Data description language (Structure of Management Information, SMI)
    - Definition of data types and object model
    - Language to describe MIB objects
    - Based on Abstract Syntax Notation One (ASN.1)
    - Definition of additional SMI-specific data types
  - Network management protocol
    - Exchange of information and commands
    - Example: Simple Network Management Protocol (SNMP)
  - Extensions for security support:
    - Encryption, authentication, authorization
    - Example: Simple Network Management Protocol v3 (SNMPv3)

Localization of Network Management

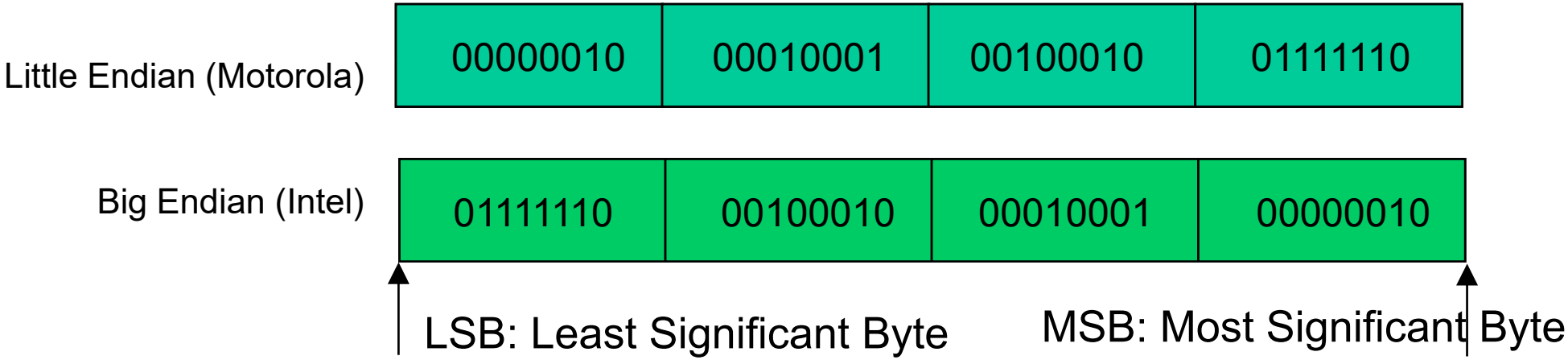
- SNMP:
  Simple Network Management Protocol (RFC 1157, May 1990)
  - SNMPv3 (RFC 3414, December 2002)



[RFC 1157]

ASN.1

- Data types need to be represented in a standardized manner to enable its open transport across network boundaries
- Sample: integer representation: 34.677.374



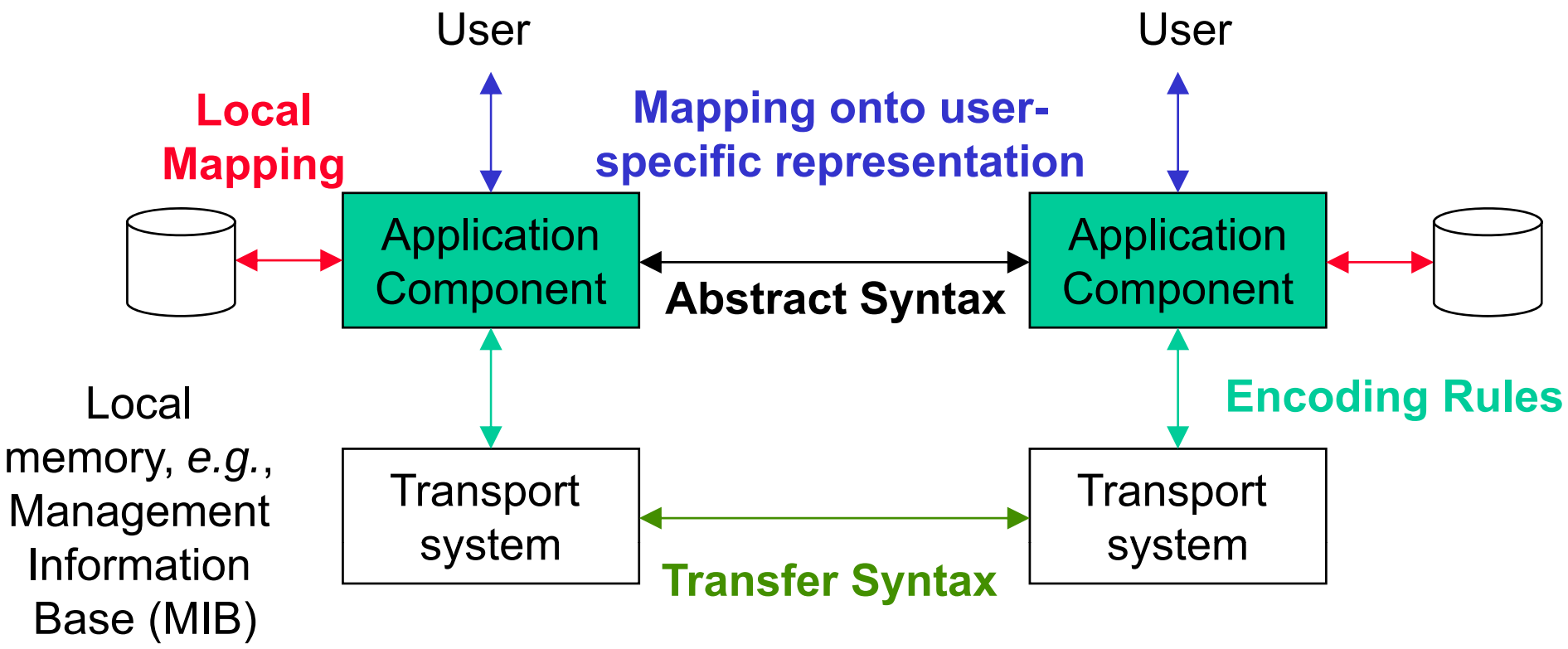
- ASN.1 includes
  - Data type description language
    - Subset used for SNMP objects (abstract syntax)
  - Basic Encoding Rules (BER)
    - Determine the mapping of data types onto the bits transferred (transfer syntax)

Abstract and Transfer Syntax (1)

- Abstract Syntax
  - Set of type definitions for data objectc, *e.g.*,
    - ASN.1
    - XDR (eXternal Data Representation) Language
  - to describe the generic dats structure independent of the encoding

- Transfer syntax
  - Concrete representation of data in transit
  - Defined by a set of encoding rules, *e.g.*,
    - Basic Encoding Rules (BER) for ASN.1 or
    - XDR rules (Remote Procedure Call)

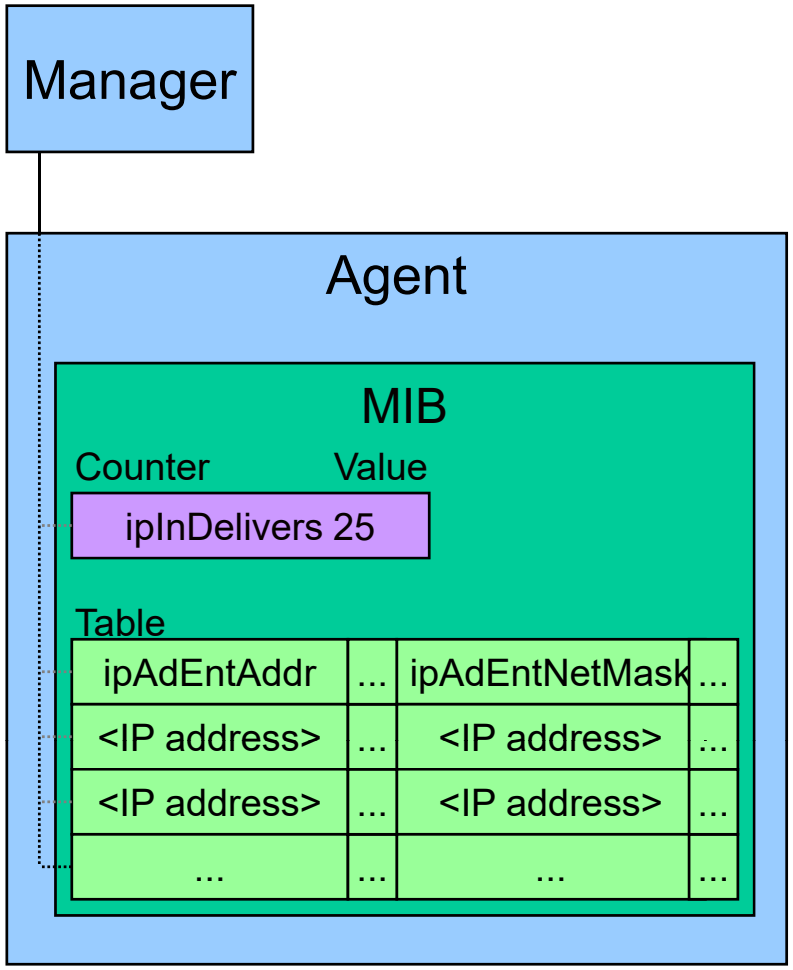
Abstract and Transfer Syntax (2)

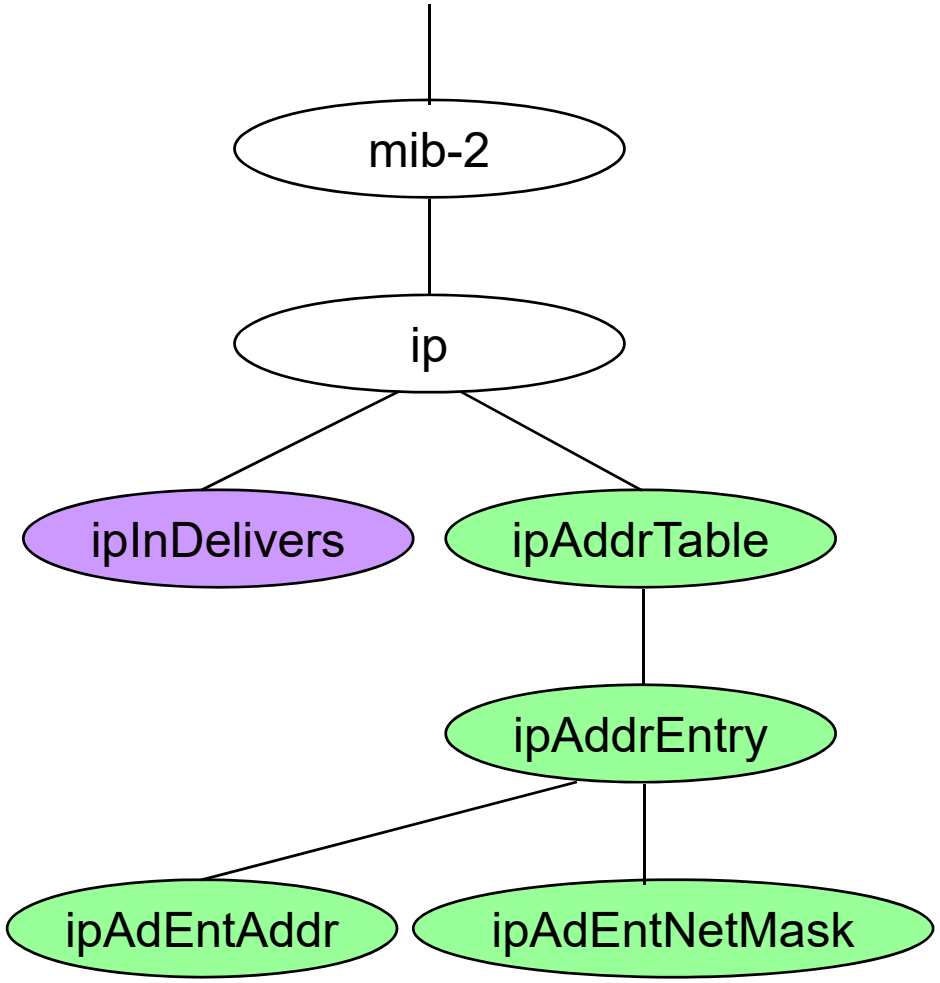


ASN.1 Types and the SNMP Instance

- Basic types (SNMP/SMI)
  - BOOLEAN
  - INTEGER
  - ENUMERATED
    - Enumeration of objects
  - OBJECT IDENTIFIER
    - Set of values for an object
  - BIT STRING
    - Sequenz of 0 or more bits
  - OCTET STRING
    - Sequenz of 0 or more bytes
  - String types
    - *E.g.*, Numeric, Printable, General, Graphics
  - REAL
- Additional types
  - SEQUENCE
    - Fixed and ordered list of data types
  - SEQUENCE OF
    - Fixed and ordered list of elements of the same data type
  - SET
    - Fixed and ordered list of types, including optional ones
  - SET OF
    - Fixed and ordered list of types of elementes of a single types, including optional ones
  - CHOICE
    - List of alternative types

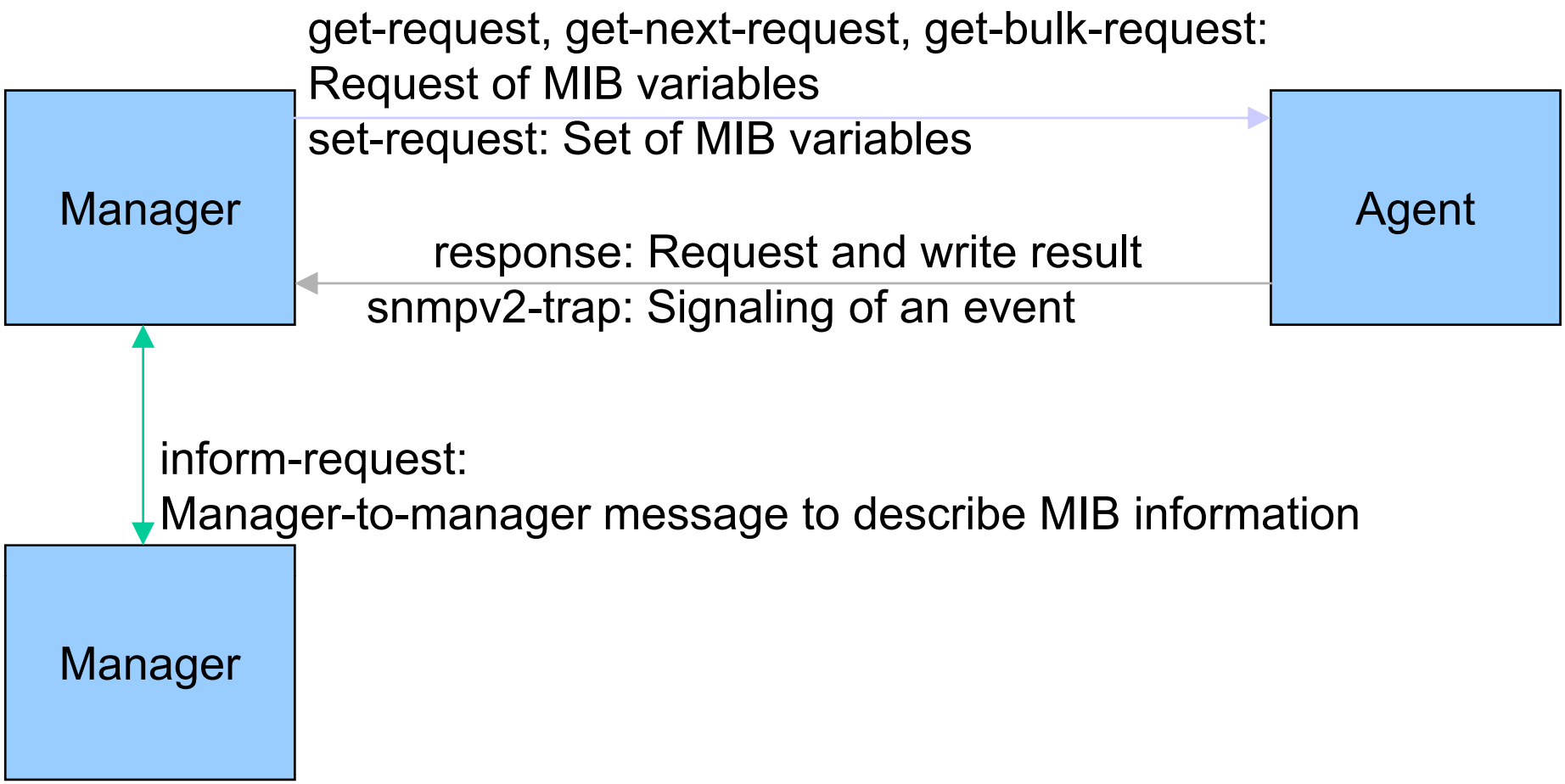
Management Information Base (MIB)





[RFC 3418]

SNMP — Message Exchange



SNMP — PDU Types

GetRequest, GetNextRequest, SetRequest, SNMPv2-Trap, InformRequest

PDU Typ	Request ID	0	0	Variable

Response

PDU Typ	Request ID	Error Status	Error Index	Variable

GetBulkRequest

PDU Typ	Request ID	non repeaters	max. repetitions	Variable

Name 1	Value 1	...	...	Name n	Value n

- SNMP runs over UDP (Port 161, 162 used for traps)
- Application responsible for packet loss and corruption!

SNMPv3

- Extension of SNMP by supporting security functions

- User Based Security Model (USM)
  - Authentication via Hashed Message Authentication Code (HMAC)
  - Privacy via encryption (DES)
  - Playback prevention via nonces
  - External key management

- View-Based Access Control Model
  - Access control by different access restrictions for MIB and operations for different managers

[RFC 2574, 3410, 3414]

Part 4: Multimedia Protocol Components

- Message Broker (RabbitMQ)
  - Introduction and Motivation
  - Message Broker Definition
  - High-Level Process & Benefits
  - RabbitMQ & Concepts
  - Use Cases
  - Demos

Thanks to Guilherme Sperb Machado, Swisscom AG

Introduction and Motivation

- Scalability is not just about technology
- Overuse of synchronous calls
- Inappropriate uses of databases
- Faulty applications affecting others applications
- Lack of caching
- No learning from history
- Avoid relying on vertical scale

Message Broker Definition

- Entity which accepts, stores, and forwards binary blobs of data (messages)

- Messages can include any kind of information, *e.g.*, about
  - Status
  - Events
  - Changes

- Acts as an intermediary platform

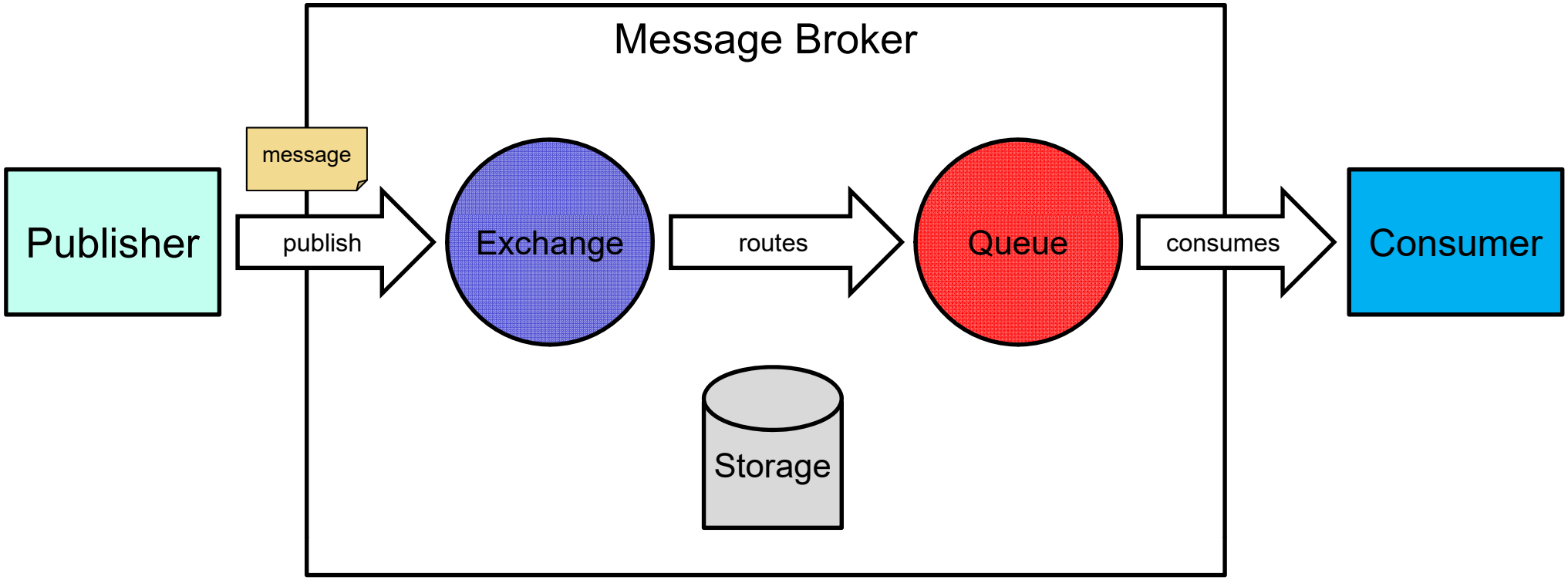
RabbitMQ

Apache Kafka

Amazon SQS

NATS

High-Level Process (1)



High-Level Process (2)

- Applications push messages to a message broker
- Message broker's software stores the message until receiving application connects and takes the message off the queue
- The receiving application then treats the message and ends the message life-cycle, or forwards the message again to the message broker

Benefits (1)

- Decoupled systems
  - More specialized applications
  - Less applications directly depending of each other
- Specialized applications receiving/generating events
  - Normal events
  - Failure events
- Scalability
  - Millions of messages per second
  - Traffic spikes are easier to handle
- Resiliency
  - The whole system can still be operational even if part of the backend applications are not operational

Benefits (2)

- Redundancy
  - If applications are not taking messages off the queue, the message is persisted to disk and it's not lost
  - Messages can be re-processed later
- Transactional
  - Applications signalize that successfully processed the message before it is completely removed from the queue
- Monitoring
  - The whole message broker system can be monitored (e.g., queue size, amount of messages, message rates)

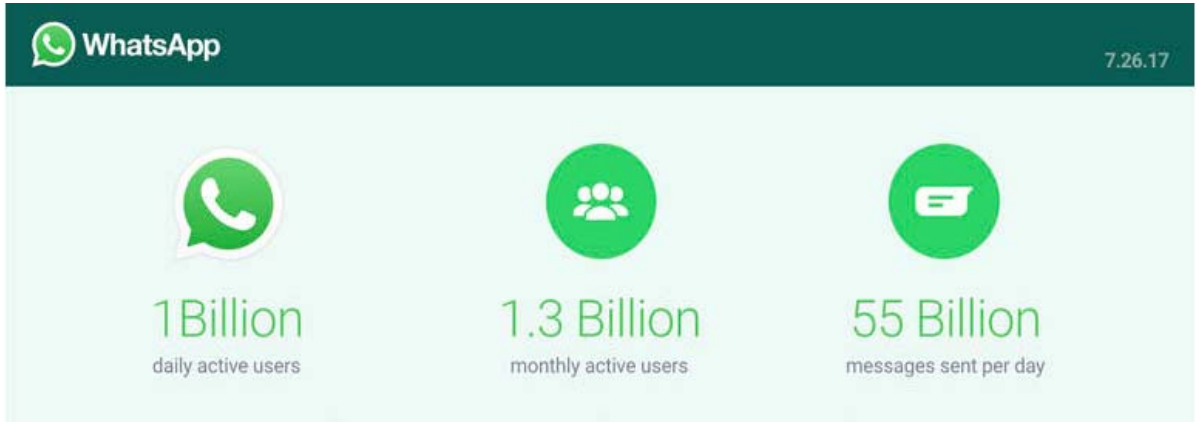
Protocol Standards

- MQTT (Message Queue Telemetry Transport)
  - Simple protocol (binary), focused on publish-subscribe
  - Resource constrained devices (low bandwidth, low latency)
- STOMP (Simple/Streaming Text Oriented Messaging Protocol)
  - Text-based (analog to HTTP), simple implementation, publish-subscribe
  - Extensible, but hard to port code between STOMP brokers
- **AMQP (Advanced Message Queuing Protocol)**
  - Designed as an open replacement for proprietary protocols
  - Advanced features, e.g., flexible routing, topic-based publish-subscribe messaging
  - Focus on reliability and interoperability

RabbitMQ

- Open-source message-broker software
  - RabbitMQ speaks **AMQP**, MQTT, and STOMP
- Enterprise edition
- Cluster-oriented
- 1 million msg/sec on Google Compute Engine
  - 30 RabbitMQ RAM nodes
  - 1 RabbitMQ Disc node
  - 1 RabbitMQ Stats node
  - Each with 8 vCPUs and 30GB of RAM

  ~ 86 billion messages per day



AMQP: Advanced Message Queuing Protocol (ISO Standard)
MQTT: Message Queue Telemetry Transport (OASIS and ISO standard)
STOMP: Streaming Text Oriented Messaging Protocol

RabbitMQ

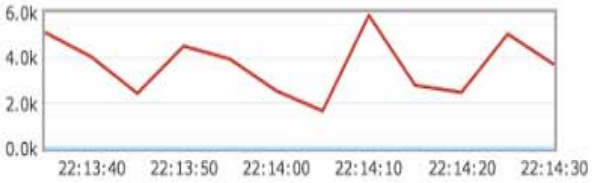
User: queue
RabbitMQ 3.2.3, Erlang R15B01
Log out

Overview | Connections | Channels | Exchanges | Queues | Admin

Overview

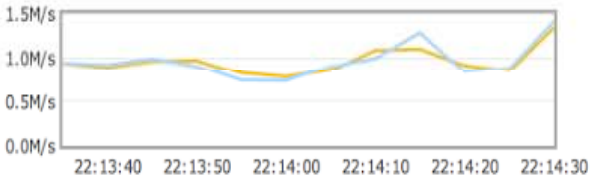
Totals

Queued messages (chart: last minute) (?)



Ready	2,343 msg
Unacknowledged	0 msg
Total	2,343 msg

Message rates (chart: last minute) (?)



Publish	1,345,531/s
Deliver (noack)	1,413,840/s

Global counts (?)

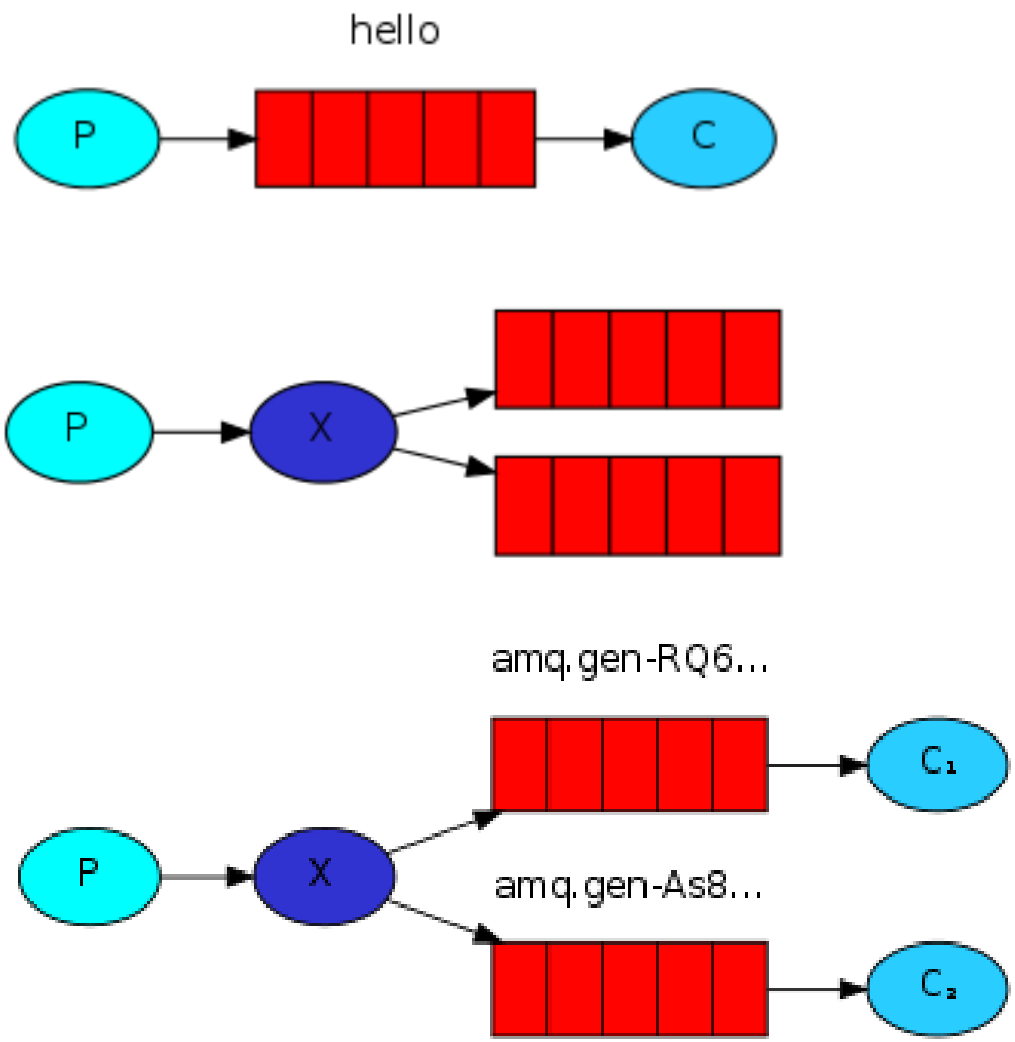
Connections: 12690 | Channels: 12690 | Exchanges: 194 | Queues: 186 | Consumers: 10304

Nodes

Name	File descriptors (?)	Socket descriptors (?)	Erlang processes	Memory	Disk space	Uptime	Type
rabbit@b-rabbitmq-queue-1te2	445 262144 available	395 235837 available	4594 1048576 available	252MB 12GB high watermark	6.2GB 48MB low watermark	1h 33m	RAM

RabbitMQ: Concepts (AMQP)

- Producer
- Consumer
- Queue
- Message
- Exchanges
  - Types: direct, topic, fanout, headers
- Bindings
- RPC



Producer and Consumer

- A *producer* is an application that sends messages
- *Producer* can only send messages to an *exchange*

- A *consumer* is an application that receives messages

- Example:
  - Payment → Message
  - Application that collects payments from users → Producer
  - Application that process payments, by analyzing them, storing to the database, and sends an email with a confirmation → Consumer

Queue

- A *queue* is a buffer entity that stores messages

- Attributes
  - Name
  - Durable
    - Queue survives a broker restart (declarative)
  - Exclusive
    - Used by only one connection and the queue will be deleted when that connection closes
  - Auto-delete
    - Queue that has at least one consumer is deleted when last consumer unsubscribes
  - Arguments (optional)
    - *E.g.*, message TTL and queue length limit.

Temporary Queue

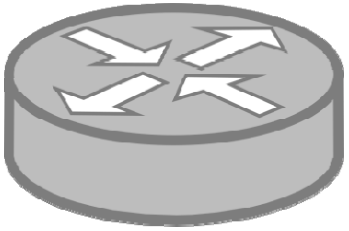
- Giving a queue a name is important when
  - Producers and Consumers share an specific queue
- Depending on the use case, it is not necessary to have a queue always active
  - Example: if an empty/fresh queue is required for a consumer when it connects
  - Solution: a temporary queue associated to the consumer lifecycle
- Queues can have a random name (or the broker chooses one)

Message

- Message acknowledgements
  - basic.deliver, basic.get-ok, basic.ack
- Rejection
- Re-queue
- Routing key

Exchanges

- An *exchange* is a *message* router
  - Receive *messages* from *producers*
  - Based on some rules, pushes them to *queues*



- When a message arrives to an exchange...
  - Should it be appended to a particular queue?
  - Should it be appended to many queues?
  - Or should it get discarded?

- Rules for such behavior are defined by *exchange type*

Exchange Type: *direct*

- Delivers messages to queues based on the message routing key
- Ideal for the unicast routing of messages

Exchange Type: *fanout*

- Route *messages* to all of the *queues* that are bound to it
- Routing key is ignored
- Ideal for the broadcast of messages
  - E.g., *messages* transmitted to all *queues* attached to such *exchange*
  - Thus, meaning that all applications (consumers) receive a particular message

Exchange Type: *topic* (1)

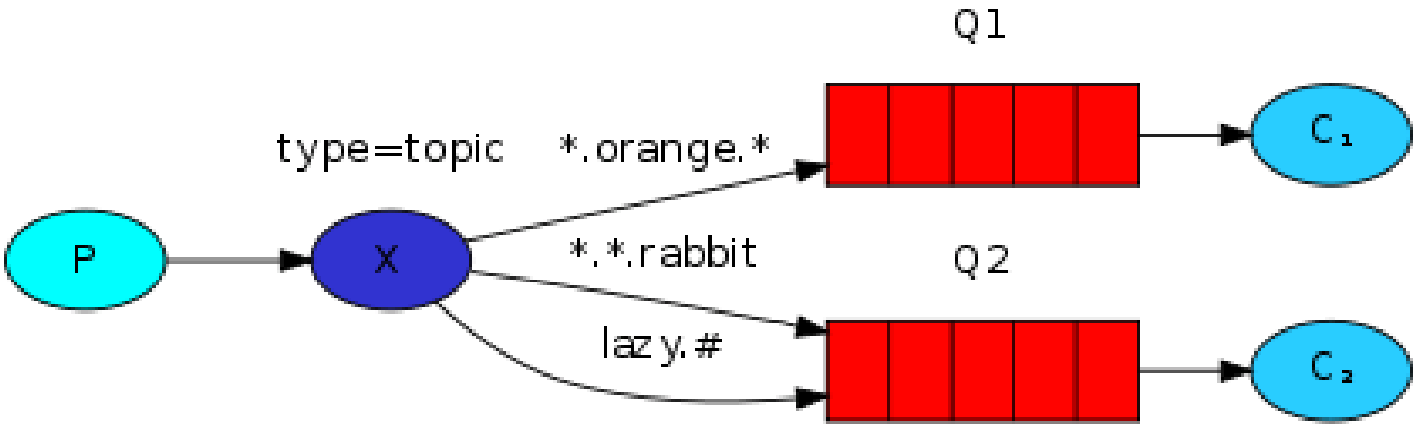
- Exchanges that forward messages based on a *topic*

- Routing key (message)
  - Must be a list of words, delimited by dots
  - Limit: 255 bytes
  - Words can be anything, but usually they specify roles/behavior inherent to the message

- Binding
  - Must also be in the same format as the routing key
  - However, it allows to be described in a <u>pattern</u> (*****'s and **#**'s)
  - The message is forwarded to a *queue* if a message's *routing key* matches such specific *binding key pattern*

Exchange Type: *topic* (2)

- Binding key pattern:
  - * (star) can substitute for exactly one word
  - # (hash) can substitute for zero or more words



- Me
  - E.g., `quick.orange.rabbit, lazy.orange.elephant, quick.orange.fox`
- What about `lazy.orange.old.elephant` ?

Default Exchange

- Direct exchange with no name (empty string), pre-declared by the broker
  - Every created queue is automatically bound to it with a routing key which is the same as the queue name

- Example
  - Application declares queue ”paid_orders”
  - A *default exchange* with no name is created by the broker
  - Broker binds the *default exchange* to ”paid_orders” queue, with ”paid_orders” as the routing key

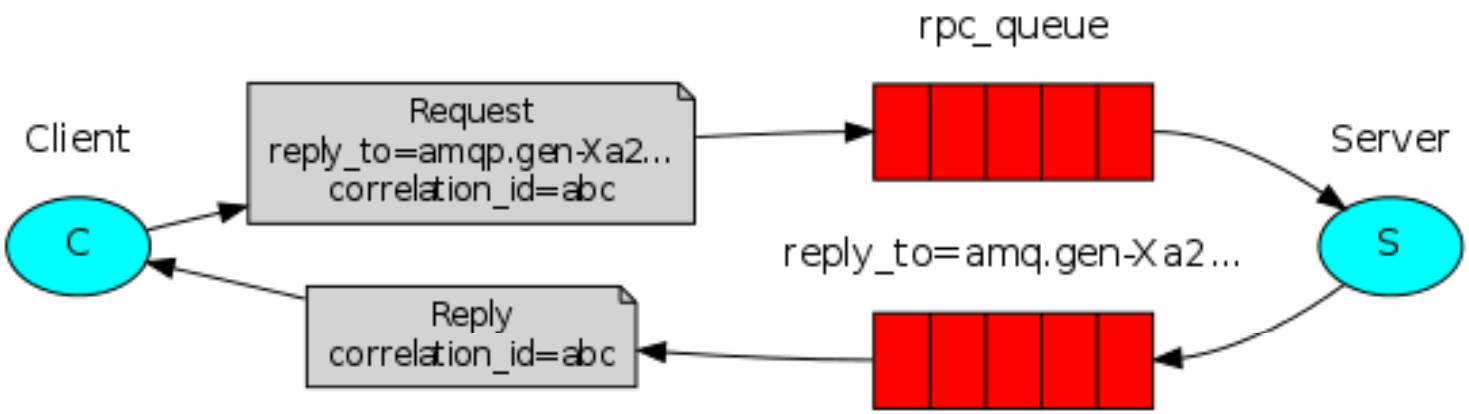
- Always an exchange delivers messages to queues
  - Even if an exchange is ”invisible”

Binding

- Relationship between message and a *queue*
  - Rules that *exchanges* use to route messages into *queues*
  - ”the *queue* is interested in messages from this *exchange*”

- Routing key (optional)
  - Depends on *exchange types*
  - E.g., *fanout* exchanges simply ignored its value

RPC: Callback Queue (1)



- Decoupled *queues* for requests and responses
  - Callback pattern

- Useful for:
  - APIs that dispatch requests to more specialized servers
  - APIs that wait until a response is available

RPC: Callback Queue (2)

- Example:
  - **API Gateway** for payments: `/pay/visa`, `/pay/mastercard`
  - **API Gateway** has the role to only dispatches the API calls to more specialized systems
  - Waits for an answer (from specialized payment applications) to give back to the customer

- Advantage:
  - Possibility to scale in different layers: load balancers, API Gateway applications, and specialized payment applications

AMPQ: Declaring Entities

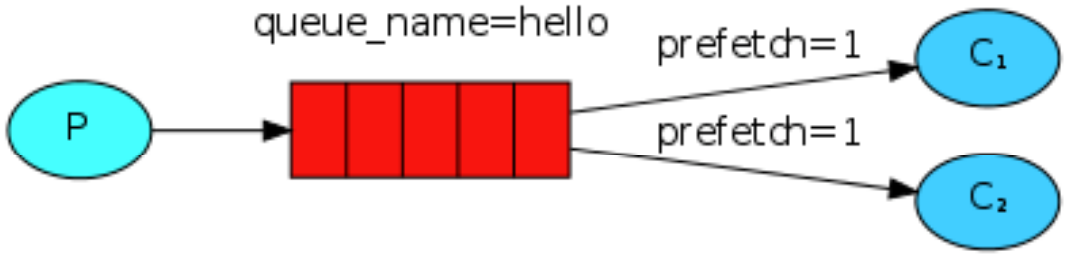
- Programmable protocol
  - Protocol operations are able to, e.g., declare queues and exchanges, define bindings between them, and subscribe to queues
- Entities and routing are primarily **defined by applications**, not a broker administrator
  - Freedom to application developers
  - …but requires to be aware of potential *declaration conflicts*, e.g., queues declared twice with different configurations
- Declaration conflicts
  - *Indication of misconfiguration*

Queue and Message Durability

- Unexpected crashes or disruptions might occur in the Broker (RabbitMQ) cluster
- Properties (durable and delivery modes) can be set to persist queue configuration and messages to, *e.g.*, disk

Fair Dispatch

- Some consumers can get busy, some others not
  - How to tell the broker to distribute the messages with a fair policy?



- By default, RabbitMQ doesn't look at the number of unacknowledged messages for a consumer

- Solution:
  - Set ”prefetch_count” attribute on the ”basic_qos” of a consumer

Message Acknowledgement (1)

- Deliver Acknowledgement
  - Consumer acknowledges that a message had been received, processed and that the broker can delete it
  - Delivery tags

- Publisher Confirms
  - Confirms that a message was published to the broker

Message Acknowledgement (2)

- Manual acknowledgements can be deactivated
  - However, messages can be lost
- Manual acknowledgements can be forgotten
  - Messages will be redelivered, over and over again
  - Serious consequences (e.g., RAM consumption increase on the broker)

RabbitMQ: Simulator Demo

- www.tryrabbitmq.com
- github.com/RabbitMQSimulator/RabbitMQSimulator

Use Cases

- Order System
  - Specialized application processing different types of orders
  - Specialized applications processing different types of payments
  - Failing payments should be forwarded to applications that differently handle the ”failure reason”

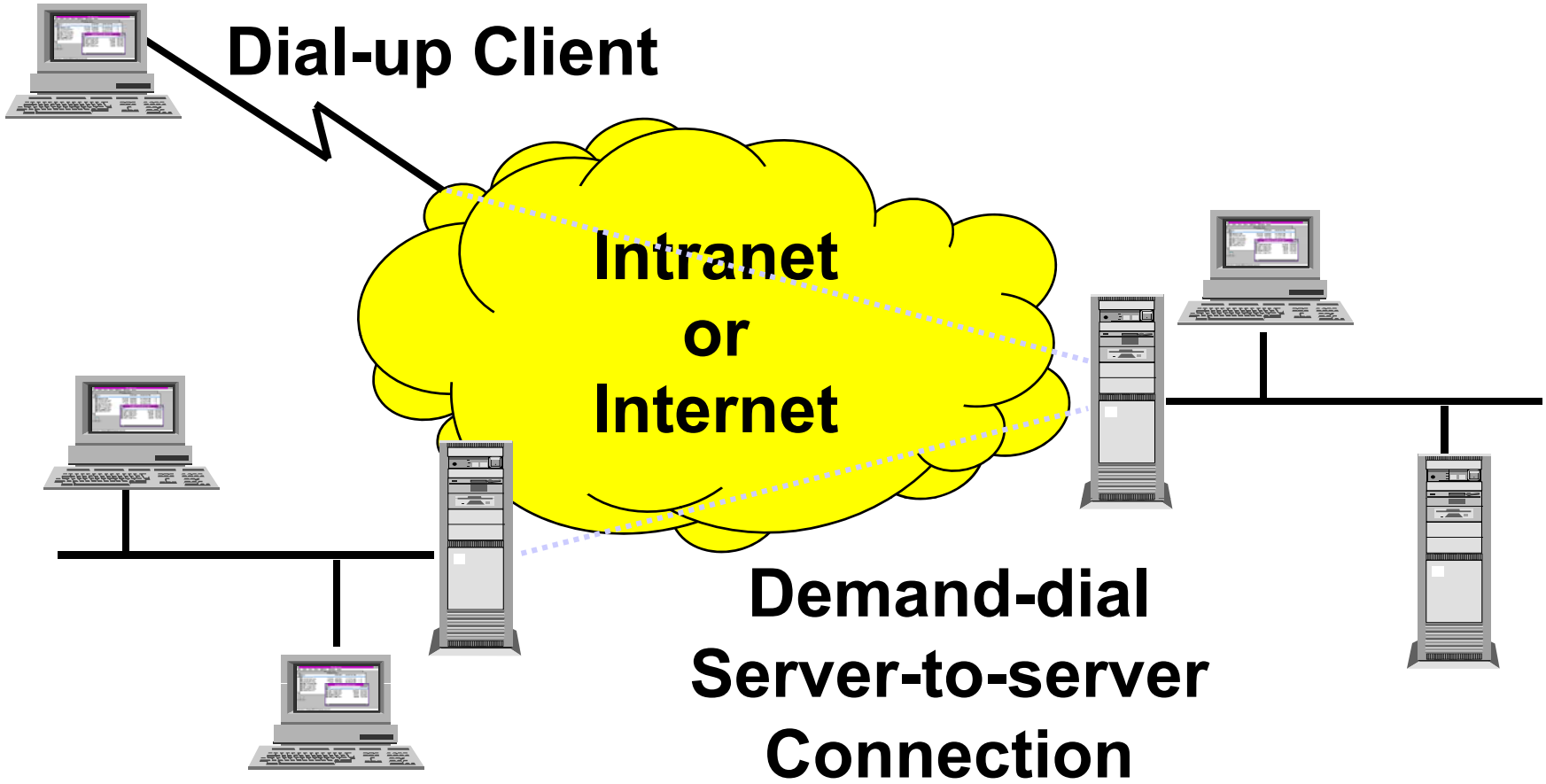
- Cloud Storage System
  - Uploads and Downloads
  - File operations
    - Create, Rename, Delete, Move files
  - Thumbnails

Part 4: Multimedia Protocol Components

- Virtual Private Networks
  - Technology Principle
  - Protocols
  - Security and QoS

Networking in a Virtual Private Area

- Idea: Create a private network via tunneling and/or encryption over the Intranet or Internet



Definition

- Virtual Private Networks (VPN) define private communications between remote locations based on public carrier networks:

  "A private network constructed over a shared infrastructure".

  – Typically employs some combination of encryption, digital certificates, strong user authentication, and access control

- Virtual
  – An artificial object, not really existing, simulated
- Private
  – Separate and distinct environments
- Network
  – Collection of communicating devices

Types and Technologies

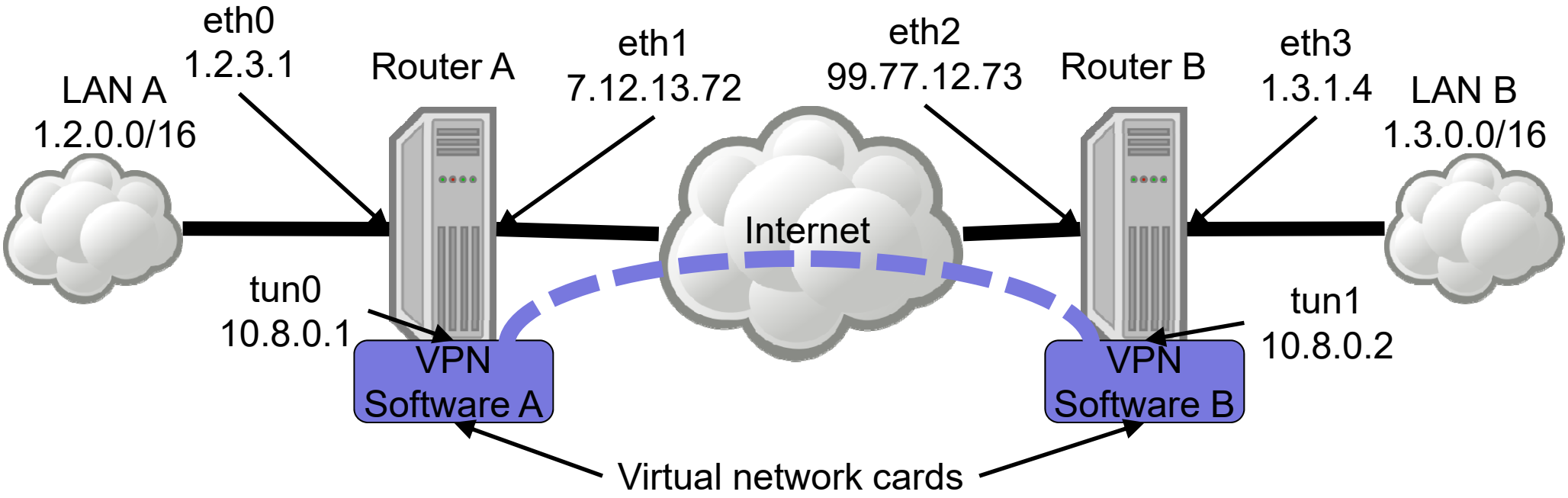
- VPN types
  - Intranet VPN: Enterprise network with access control
  - Extranet VPN: Combined enterprise and customer network with access control
  - Access/Dial-up VPN: Remote access to enterprise data via service provider points-of-presence (POP)

- Technology alternatives include
  - IP
  - Frame Relay
  - ATM
  - MPLS

Internet Protocol VPN

- Internet Protocol VPN (IP-VPN)
  - Utilization of IP
  - Basically, configuration of end-points required
  - The underlying Internet provides routing, packet forwarding, and packet transmission
    (no path configuration required as for ATM or Frame Relay)
  - Similar WAN and LAN communication
  - Out-sourced networking to the provider
    - Provider places routers within the enterprise ‘s domain
  - In-sourced networking
    - Provider interconnects privately owned routers
  - Policy support in edge routers, explicit security support

Connecting IP LANs via the Internet (1)



Destination	Gateway	Interface
1.2.0.0/16	-	eth0
10.8.0.0/24	-	tun0
1.3.0.0/16	10.8.0.2	tun0
default	7.12.13.1	eth1

Destination	Gateway	Interface
1.2.0.0/16	10.8.0.1	tun1
10.8.0.0/24	-	tun1
1.3.0.0/16	-	eth3
default	99.77.12.1	eth2

Connecting IP LANs via the Internet (2)

- Assume a host in LAN A sends packet P to address 1.3.0.14
- P reaches Router A, because it is LAN A's standard gateway
- Router A determines tun0 as outgoing interface by RTL
- Packet send via tun0 and thus is input to VPN Software A
- VPN Software A sends new packet P+ to 99.77.12.73:XX
  - P+'s payload is entire packet P but encrypted
- Router A determines eth1 as outgoing interface for P+ by RTL
- P+ reaches Router B (on eth2), where VPN Software B is listening on port XX
- VPN Software B decrypts P+ and returns P on tun1
- Router B determines eth3 as output interface for P by RTL
- P reaches destination

RTL: Routing Table Lookup

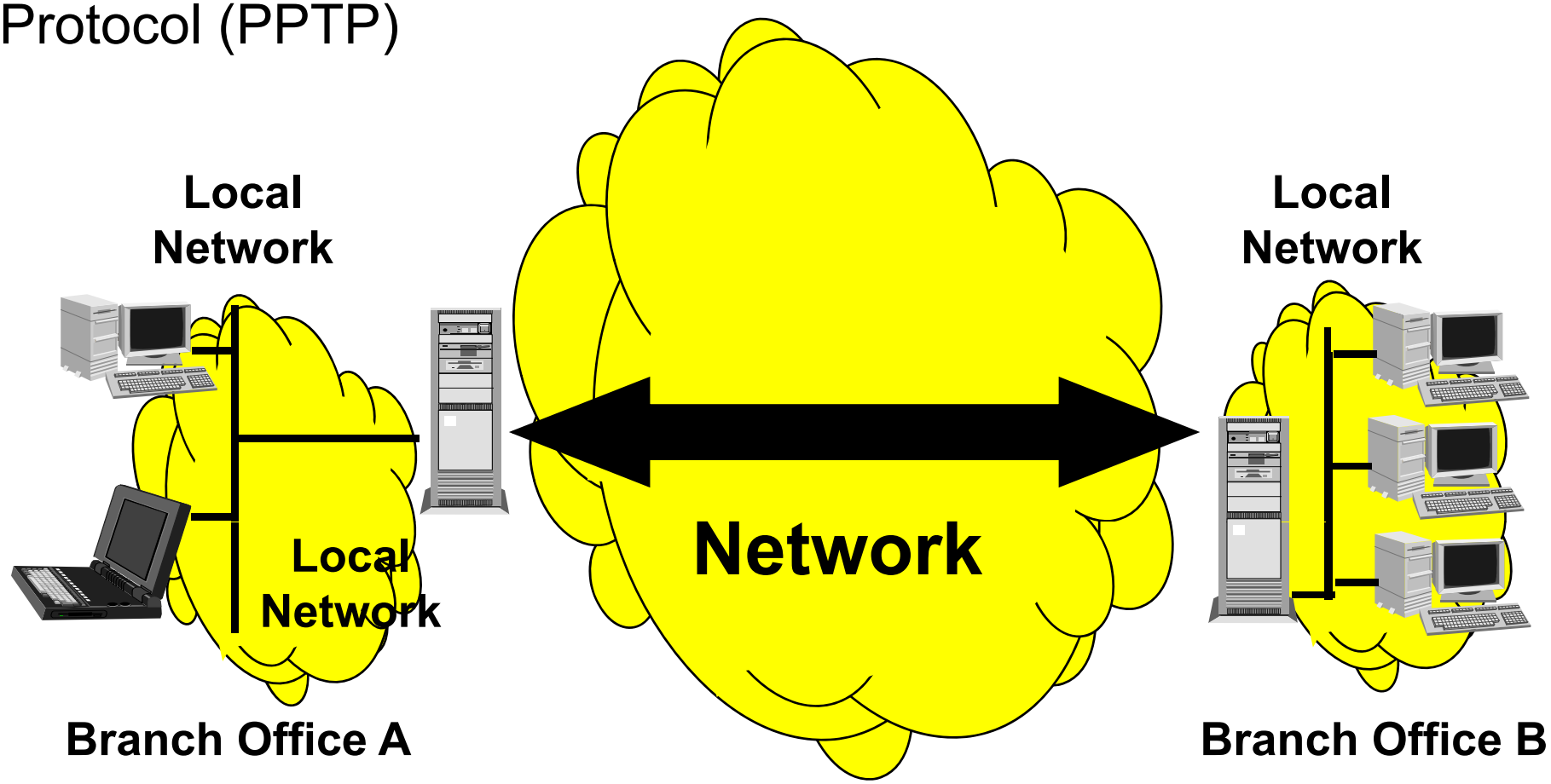
Specialized Protocols

- Point-to-point Tunneling Protocol (PPTP) [RFC 2637]
  - PPTP Forum (US Robotics, Ascend, 3Com, Telematics...)
- Combination of Point-to-point Protocol (PPP) and IP
  - Standard PPP dial up frames encapsulated in IP
  - PPP is extensible and includes an authenticated dial up standard
  - 128 bit encryption

- IETF draft on Layer 2 Tunneling Protocol (L2TP) [RFC 2661]
  - L2TP = PPTP + L2F (Layer 2 Forwarding)
- Combination: L2TP, IPSec, and Smart Cards

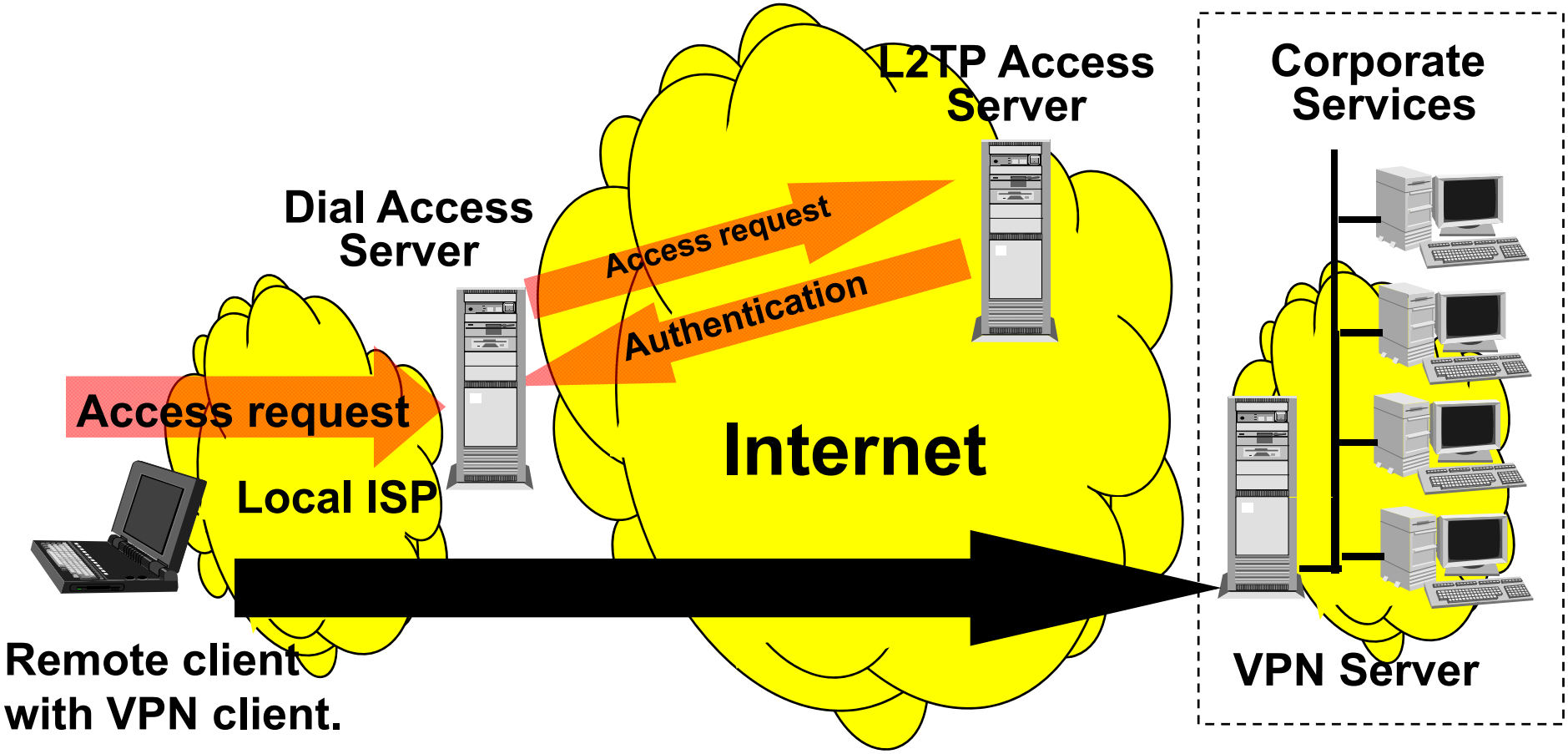
Intranet VPN

- Tunnel construction: Generic Routing Encapsulation (GRE), Layer 2 Tunneling Protocol (L2TP), Point-to-point Tunneling Protocol (PPTP)



Access/Dial-up VPN

- Dial Access Servers: Accounting, Authentication, and Authorization (AAA) by RADIUS, TACACS



VPN via DNS

- IP packets (or potentially any content) can be “encoded” into URLs
  - Request for encodedIPpacket.my_DNS_server.ch
- DNS server being responsible for my_DNS_server.ch decodes packet from URLs and forwards it
- It returns replies to these packet as a DNS response
- As soon as a client can resolve DNS requests the tunnel works
  - DNS requests are almost always allowed
    - Often even at hotspots before payment ;)
  - Provider forward DNS request to their server
    - This server contacts my_DNS_server.ch

- Usually it is tunneled via Layer 2 or 3, via DNS is a tunnel on the application layer!

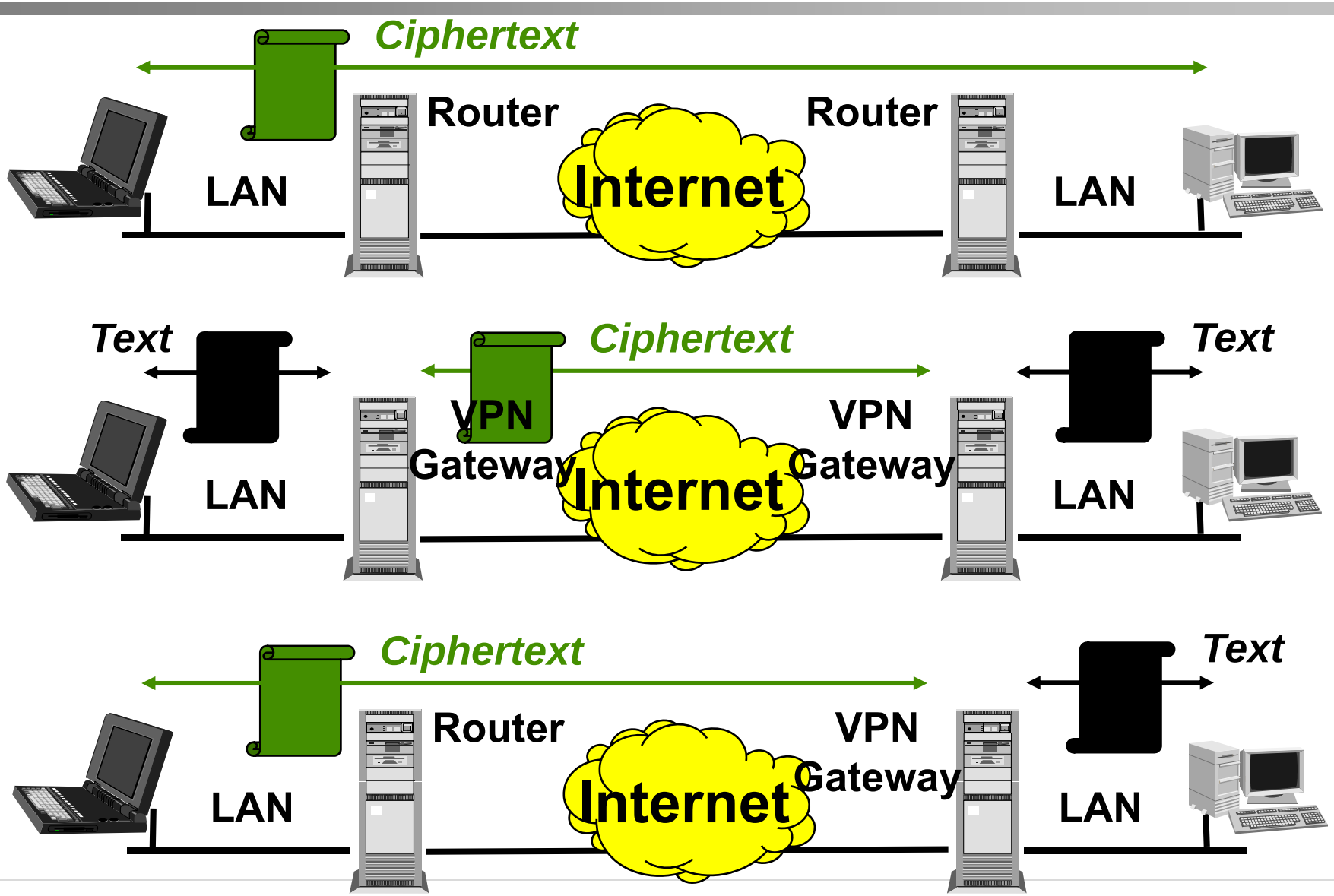
Virtual Network Interfaces

- Virtual Network Interface
  - Abstract representation of a computer network interface that may or may not correspond directly to a physical network interface.
- Operations
  - Packets send via these interfaces are not put on a medium, but input to the virtualization functionality
  - Virtual Network Interface generates (possibly encrypted) packets to produce the payload for new IP packet, which is addressed to the VPN instance y on another router Y
    - The packet is send via a physical network interface
    - The VPN instances y (and x) “normally” listen on a respective port
  - Received packets are “decoded” and the content is returned via the virtual network interface
  - Within the Internet only Virtual Network Interface-generated (“decrypted”) packets are visible

Handling Security-related Matters

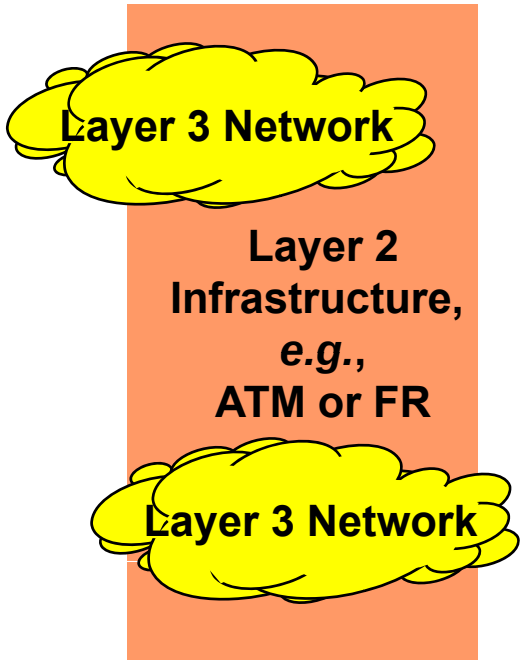
- Define access rules based on policies
  - Gateway-to-gateway and user-to-gateway
  - Centrally manage/monitor client authentication
- Minimize network access points into an enterprise
  - Network access regulated through Network Access Server
  - PPTP access server is the single entry point to LAN
- In case of tunneling utilize encryption
  - IPSec for IP tunnels, *e.g.*, 3DES
- Firewall support
  - Filter all requests at PPTP access server
    - Eliminate hacker attacks on server
  - Works alongside Internet standard firewalls

Secured Tunnels in VPNs



QoS and Management in VPNs

- Some applications require QoS
  - IP VPNs as the Internet itself lack generic QoS support
- Link layer VPNs for some/none QoS guarantees
  - Asynchronous Transfer Mode (ATM)
  - Frame Relay
  - VPs (Virtual Paths) for aggregated QoS guarantees
- VPN Management Systems
  - Scaling issues (constructing, adding)
  - Provisioning and SLA definitions



Part 4: Multimedia Protocol Components

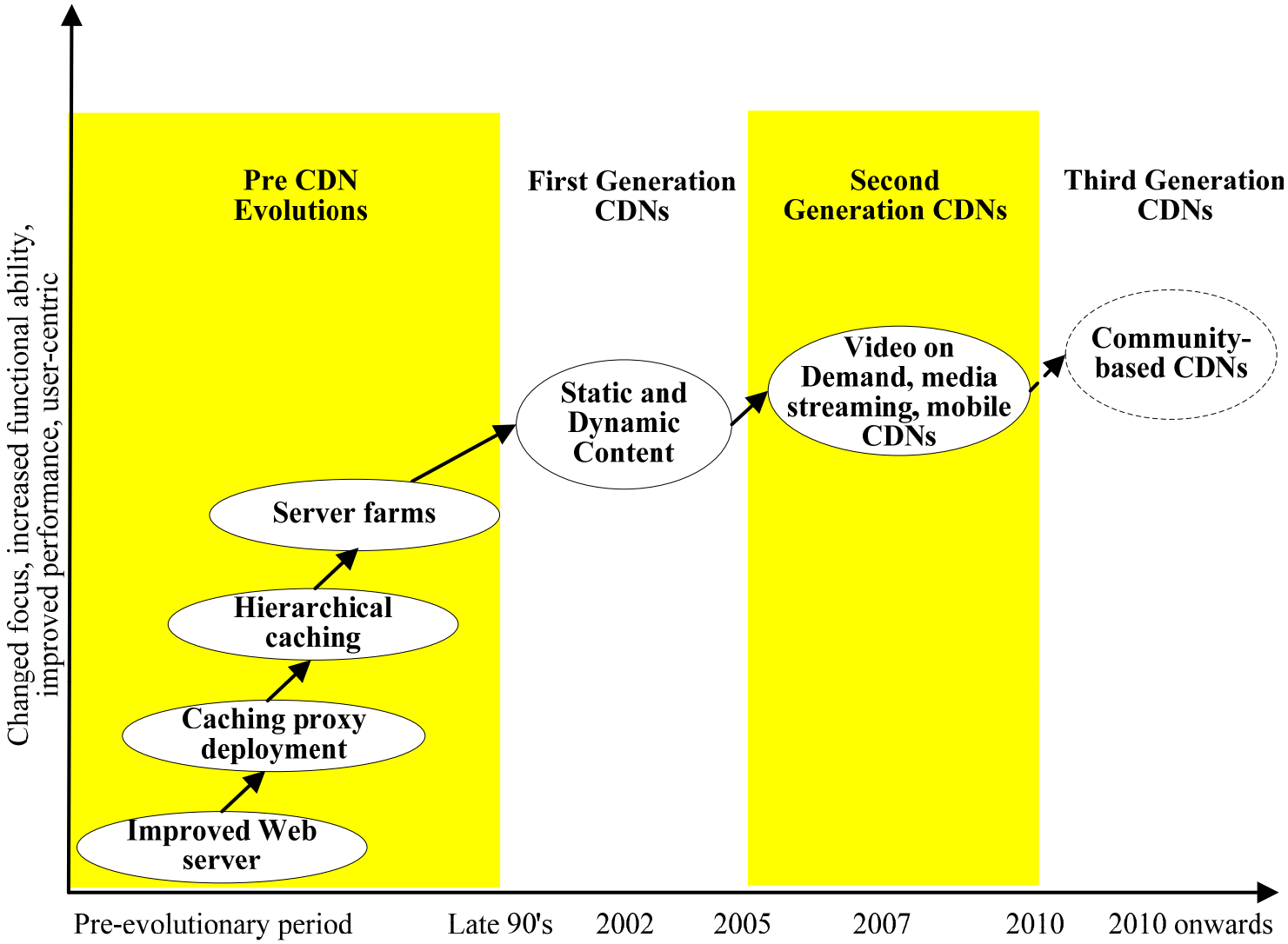
- Content Distribution Networks (CDN)
  - Development Goals and Evolution
  - Architecture and Characteristics
  - Technology
  - Taxonomy
  - Models, Limitations, Directions

More than partial thanks to Rajkumar Buyya and Mukaddim Pathan, University of Melbourne, Australia

Development Goals

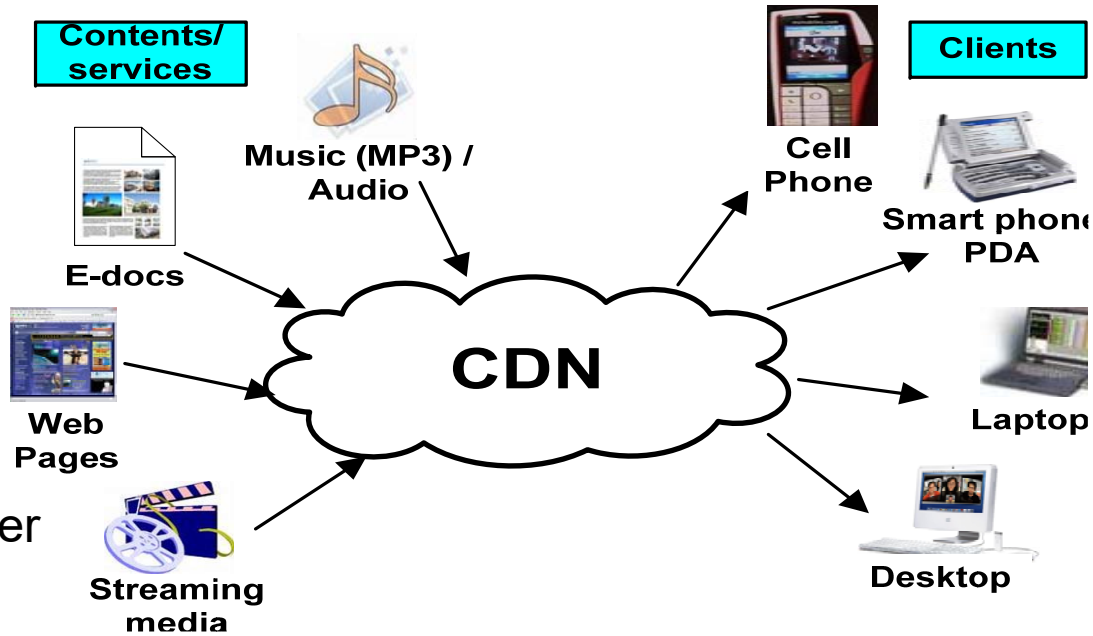
- Content Delivery Networks (CDNs) emerged due to
  - Internet service degradation and QoS management problems
  - Value-added Services (VAS) provisioning
  - Data privacy concerns
  - Handling server loads in a distributed manner
- Organizational solution
  - Moving the content to the “edge” of the Internet, close to end-users
- Technical alternatives
  - Increased bandwidth
  - Web caching
  - Web pre-fetching
- CDN advantages
  - Reduced server loads
  - Distributed network traffic
  - Reduced latency

CDN Evolution

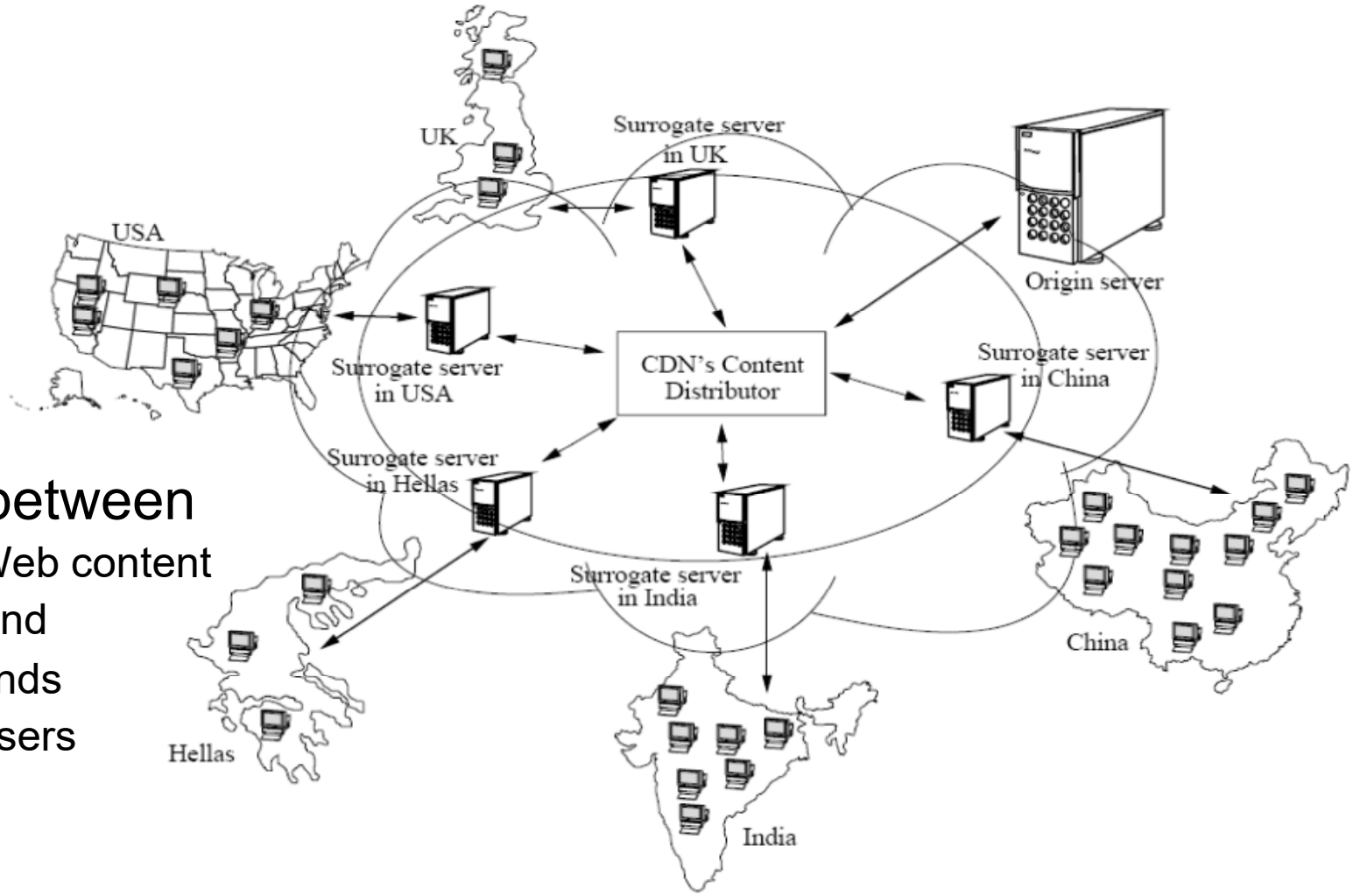


CDN-supported Content and Services

- Static content
  - Static HTML pages, images, documents, software patches
- Streaming media
  - Audio, real-time video
  - User Generated Video (UGV)
- Content services
  - Directory, e-commerce, file transfer
- Sources of content
  - Large enterprises, Web service providers, media companies, news broadcasters
- Customers
  - Media/Internet advertisement companies, data centers, ISPs, online music retail, mobile operators, consumer electronics manufacturers, other carrier companies
- User interaction
  - Cell phone, smart phone/PDA, laptop, desktop



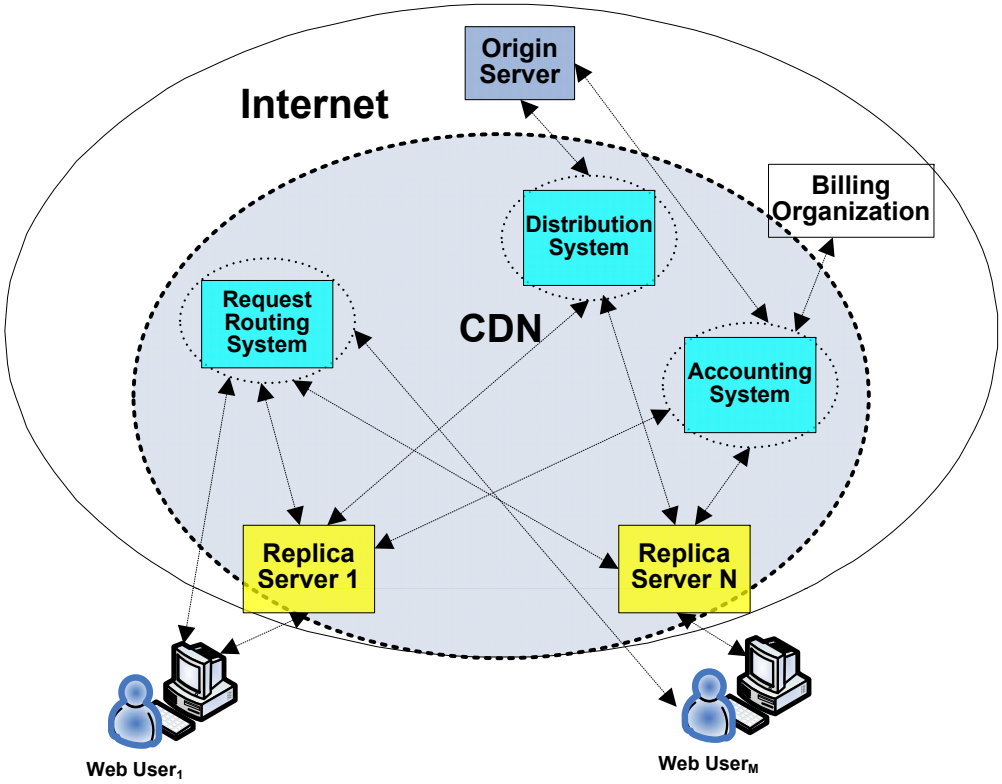
Overview and Trade-off



- Balancing between
  - Costs for Web content providers and
  - QoS demands from end-users

CDN Architecture

- Content delivery component
  - Origin server and a set of edge servers (surrogates) to replicate content
- Request routing component
  - Direct user requests to edge servers
  - Interacting with distribution component to keep an up-to-date view of content
- Content distribution component
  - Moves content from the origin to edge servers and ensures consistency
- Accounting component
  - Maintains logs of client accesses and records usage of servers
  - Assists in traffic reporting and usage-based charging



CDN Characteristics and Trends

- Scalability
  - Ability to expand to handle new and large amounts of data, users, transactions
  - Requires capabilities for dynamic provisioning and high quality content delivery, with low operational cost
    - Future trend: Content providers, end-users will pay to get high quality content
- Security
  - Protection of content against unauthorized access and modification
  - Requiring physical, network, software, data, and procedural security
    - Future trend: Reducing business interruption by combating against DDoS attacks and other malicious activities
- Reliability and Performance
  - Service availability, handling possible outages, and supporting improved end-user experience
  - Requires a fault-tolerant network with appropriate load balancing
    - Future trend: distributed content location, cache consistency and routing mechanisms

Technology Comparison

Features	CDNs	Data Grids	Distributed Databases	P2P Networks
Category	A collection of networked computers spanning the Internet	Data intensive computing environment	Locally organized collection of data distributed across multiple physical locations	Information retrieval network formed by ad-hoc aggregation of resources
Constitution	Distribution of cache servers to the edge of the Internet	Formation of a VO of participating institutions	Federation or splitting of existing database(s)	Collaboration among peers
Main goal	Reducing Web latency during content delivery	Performance gain through data distribution by pre-staging, optimal source selection, and high speed data movement	Integration of existing databases and replication of database fragments in a transparent manner	File sharing among peers
Integrity	Integrity between caches	Integrity between data grid replicas	Integrity between multiple DBs	N/A
Consistency	Strong cache consistency between replicated content	Weak consistency between data grid replicas	Strong database consistency between distributed DBs	Weak consistency between cached content
Autonomy	None	Autonomous participants	Autonomous DDB sites	Autonomous peers
Operational Activities	Content caching	Seamless analysis, collaboration, and maintenance of data across organizational and regional Boundaries	Query processing, optimization, and Management	Locating or caching content, encrypting, retrieving, decrypting, and verifying content
Administration	Individual companies. proprietary in nature	Institutions who cooperate on some shared goals	Single authoritative entity	Self-interested end users/peers

Commercial CDNs (1)

And many more!

Commercial CDNs (2)

CDN Name	Service Type	Coverage	Products and Solutions
Akamai http://www.akamai.com	Provides CDN service, including streaming	Covers 85% of the market. 25,000 servers in 900 networks in 69 countries. It handles 20% of total Internet traffic today	Edge Platform for handling static as well as dynamic content, Edge Control for managing applications, and Network Operations Control Center (NOCC)
EdgeStream http://www.edgestream.com	Provides disrupted video streaming applications over the public Internet	Provides video streaming over consumer cable or ADSL modem connections around the globe, even over paths that have 20 router hops between server and end user	EdgeStream video on-demand and IPTV Streaming software for video streaming
Limelight Networks http://www.limelightnetworks.com	Provides distributed on-demand and live delivery of video, music, games and download	Edge servers located in 72 locations around the world	Limelight ContentEdge for distributed content delivery via HTTP, Limelight MediaEdge Streaming for distributed video and music delivery via streaming, and Limelight Custom CDN for custom distributed delivery solutions
Mirror Image Http://www.mirror-image.com	Provides content delivery, streaming media, Web computing and reporting services	Edge servers located in 22 countries	Global Content Caching, Extensible Rules Engine (XRE), Video On-Demand, and Live Webcasting

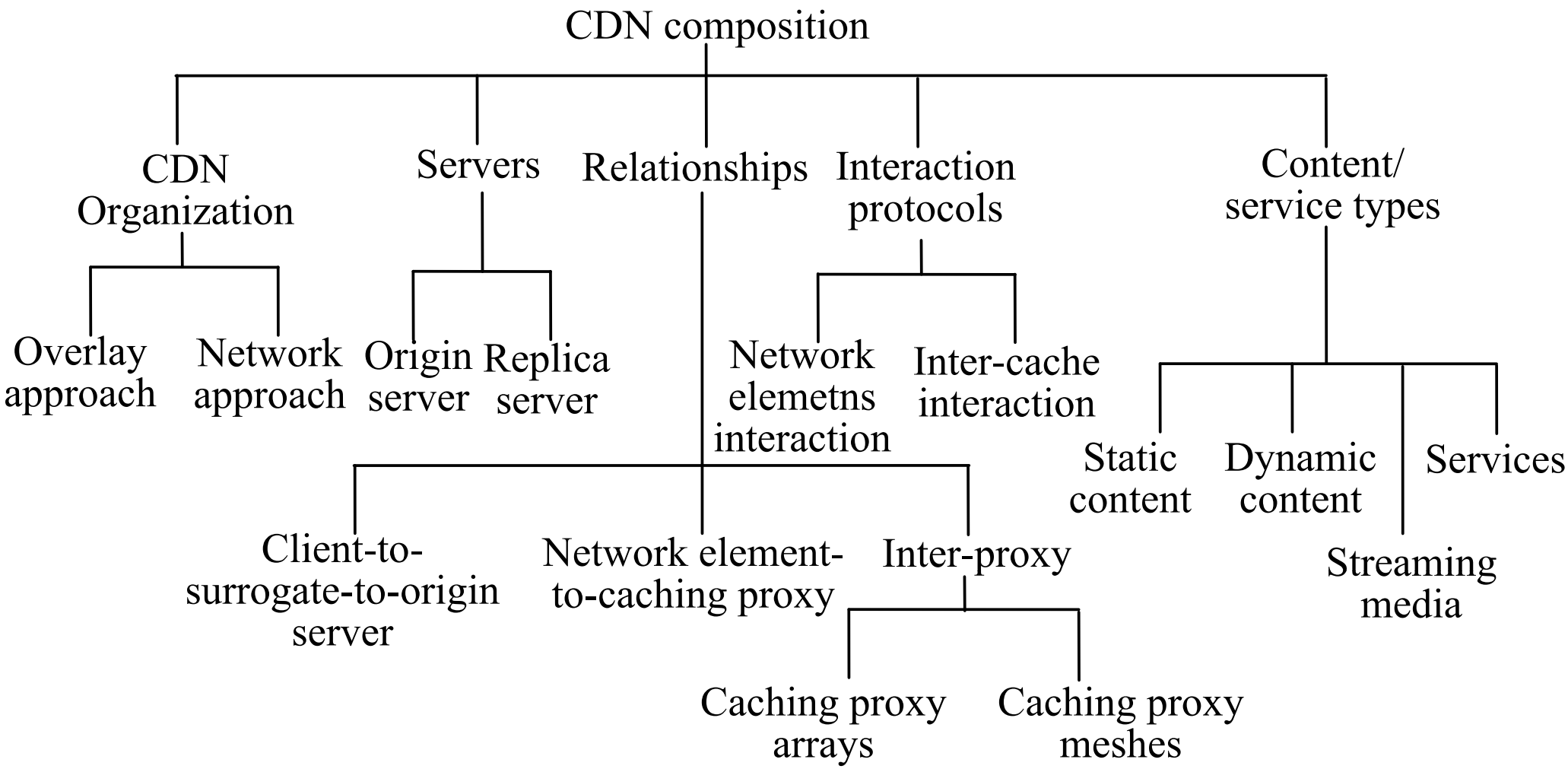
CDN Taxonomy

Issues for CDN Taxonomy
- CDN composition
- Content distribution and management
- Request-routing
- Performance measurement

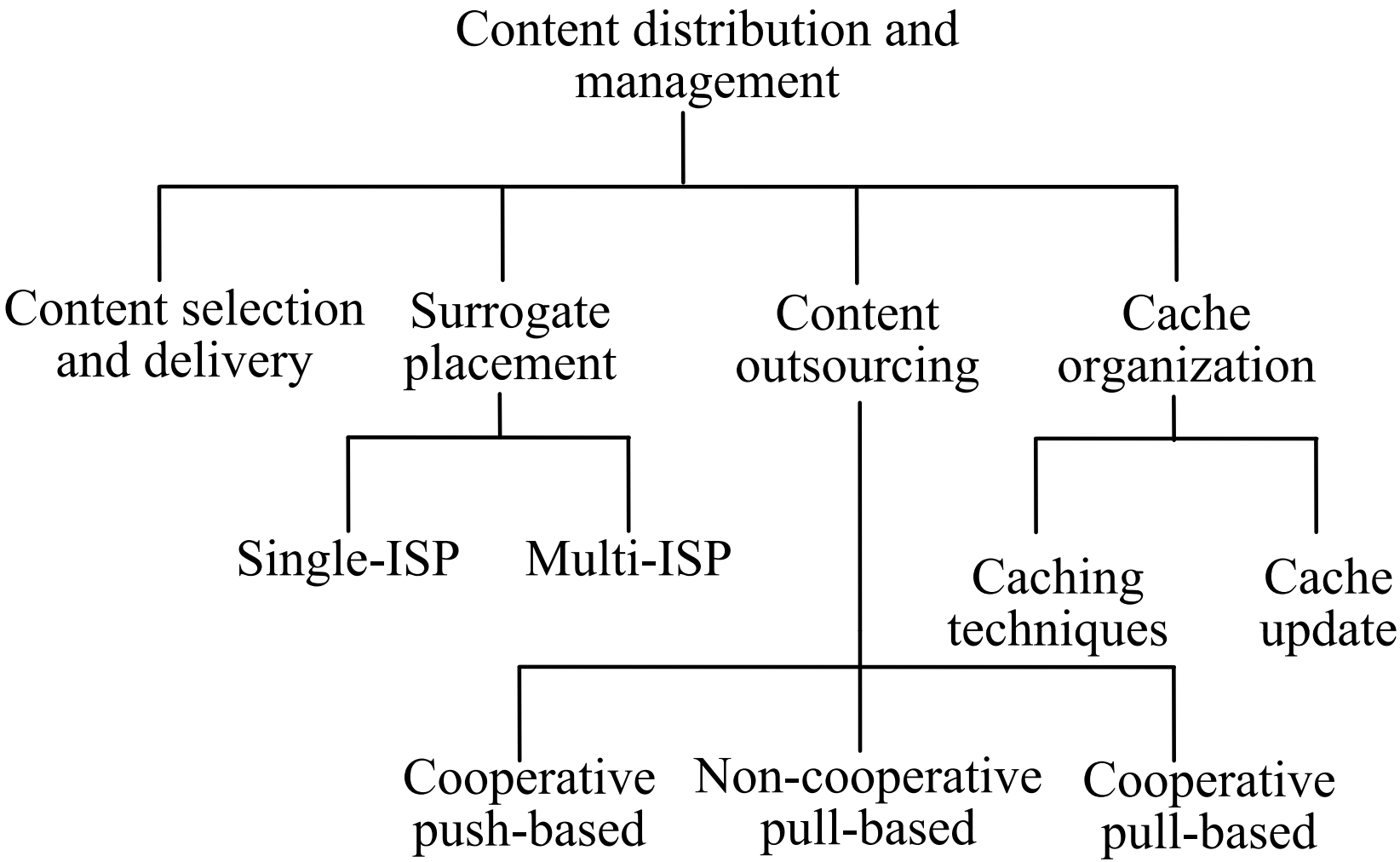
- Core issues
  - What is required for a harmonious CDN composition?
  - How to perform effective content distribution and management?
  - How to route client requests to appropriate CDN nodes?
  - How to measure a CDN's performance?

- Additional issues (not discussed here)
  - How to handle wide-area failures in a CDN?
  - How to ensure security in a wide-area CDN system?
  - How to achieve wide-area application hosting?

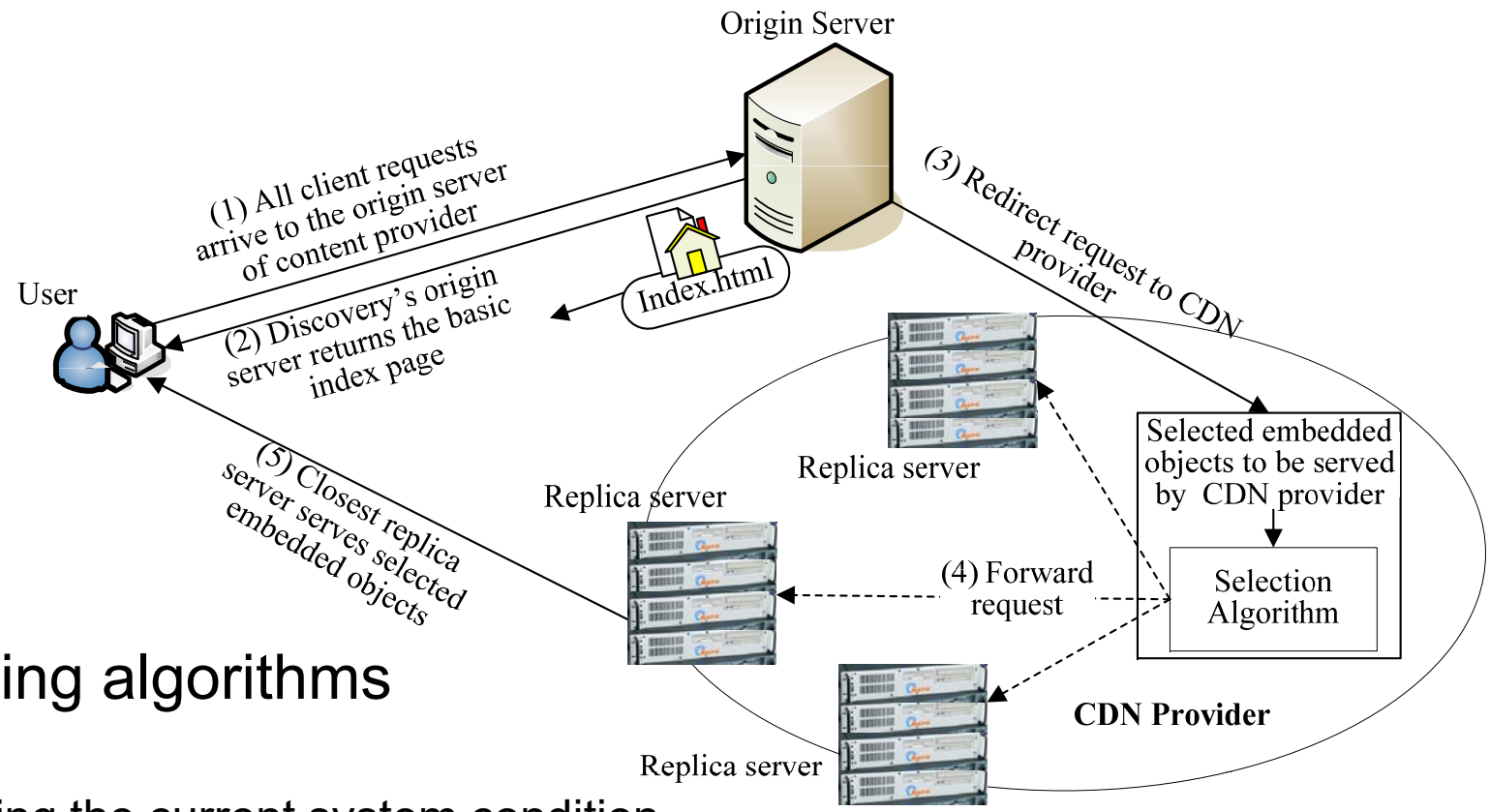
CDN Composition



Content Distribution and Management

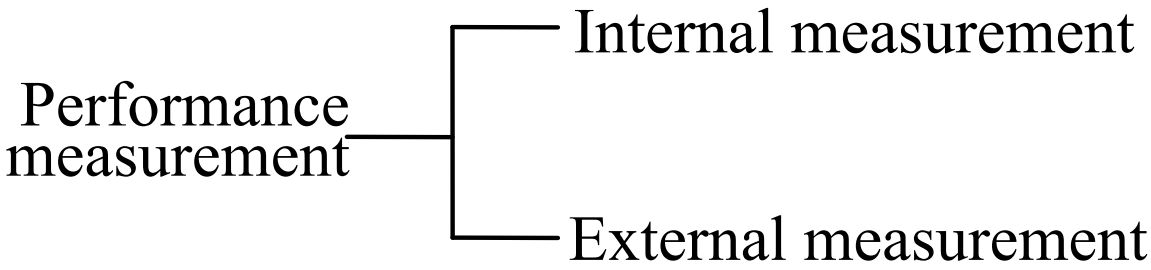


Request Routing



- Request routing algorithms
  - Adaptive
    - Considering the current system condition to select a cache server for content delivery
  - Non-adaptive
    - Use of heuristics to select a cache server rather than to consider current system condition

Performance Measurement



- Cache hit ratio
  - Ratio of the number of cached content versus total content requested
- Reserved bandwidth
  - Measure of the bandwidth used by the origin server
- Latency
  - User perceived response time
- Surrogate server utilization
  - Fraction of time during which the surrogate servers remain busy
- Reliability
  - Packet-loss measurements

SDNs Improving CDNs

- How to improve the delivery of content through SDN?

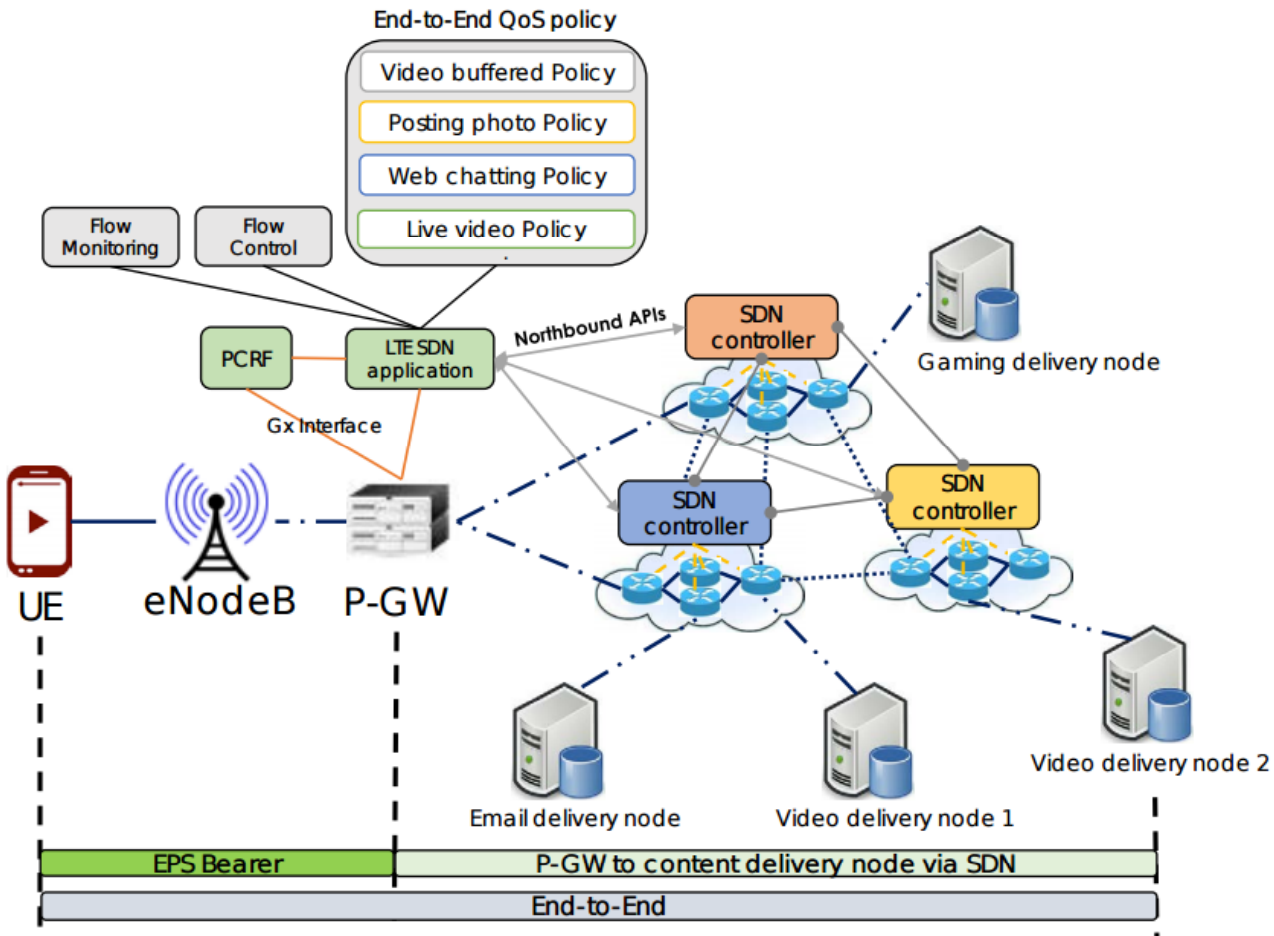


Figure source

Key advantages:

- SDN Application Awareness
- WAN Routing
- End-to-end QoS monitoring and flow control
- SDN and NFV (Network Function Virtualization)

CDN Models

Features	Typical CDN models		Advanced models for CDN peering		
	Conventional CDNs	**P2P-based CDNs**	**Peering CDNs**	**Brokering-based**	**QoS-driven (customized) brokering-based**
Nature of Content Delivery	Based on Web server Collaboration	Based on peering and content availability	Based on CDN Internetworking/ Peering	Based on CDN Performance	Based on user defined QoS (Customized)
Responsibility for effective content delivery	CDN Provider	Peers/Users	Primary CDN Provider	Content Provider	Content Provider
Entities in agreement	CDN-Content Provider	No real Agreement (Self-interested users)	CDN-Content Provider, CDN-CDN	CDN-Content Provider	CDN-Content Provider
Agreement nature	Static	N/A	Short-term or long-term	Policy-based	Dynamic
Scalability	Limited	High	High	High	High
Cooperation with external CDNs	No	No	Yes	Yes	Yes
Cooperation between CDNs	No	No	Yes	No, CDNs work in Parallel	No, CDNs work in parallel
Cooperation between users	No	Yes	No	No	No

Current Limitations of CDNs

- Commercial CDNs sign Service Level Agreements (SLA) with customers
  - Objective is to provide global competitive services satisfying QoS requirements
  - Over-provision or harness external resources on-demand
- A “Flash Crowd” causes adverse business impact
  - Very often CDNs can not provide QoS to end-users requests
  - SLA violation to end-up costing the provider
- Existing CDNs are proprietary in nature
  - Each has expensive closed delivery network
  - No simultaneous usage of the resources of multiple CDNs
- CDN expenses to have led to consolidation in the market

Emerging CDN Directions

- Unified content network
  - Integration of “Content Delivery/Distribution” and “Content Services” domains
  - Content Service Network (CSN) as a “service” distribution channel for VAS
- Dynamic content delivery
  - On-demand content generation using Web applications based on end-user request specifications (*e.g.*, scripts, animations, DHTML, XML)
  - Edge computing, context-aware data caching, content replication, content blind data caching, proprietary solutions (*e.g.*, EdgeSuite, IBM WebSphere edge)
- Web services hosting
  - Use of XML parsing, Java serialization, reflection copy, clone copy
  - Application Delivery Network (ADN) to host .NET and J2EE applications
  - Capacity Provisioning Network (CPN) for trading cache capacities
- Service-oriented Architecture
  - Content management is expected to be motivated by user preferences
  - User personalization based on data mining

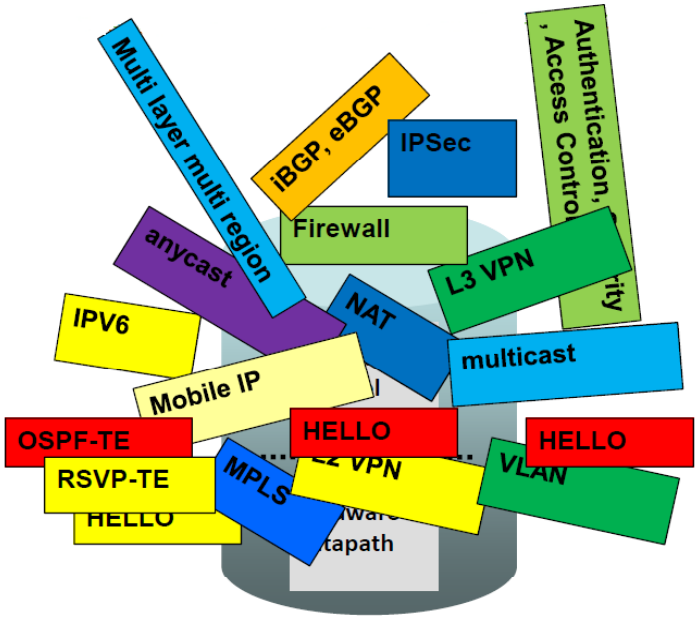
Part 4: Multimedia Protocol Components

- Software-defined Networking (SDN)
  - Introduction
  - SDN Basic Concepts and Definitions
  - OpenFlow
  - Applications and Examples

Thanks to Bruno Rodrigues, University of Zürich

Internet Limitations

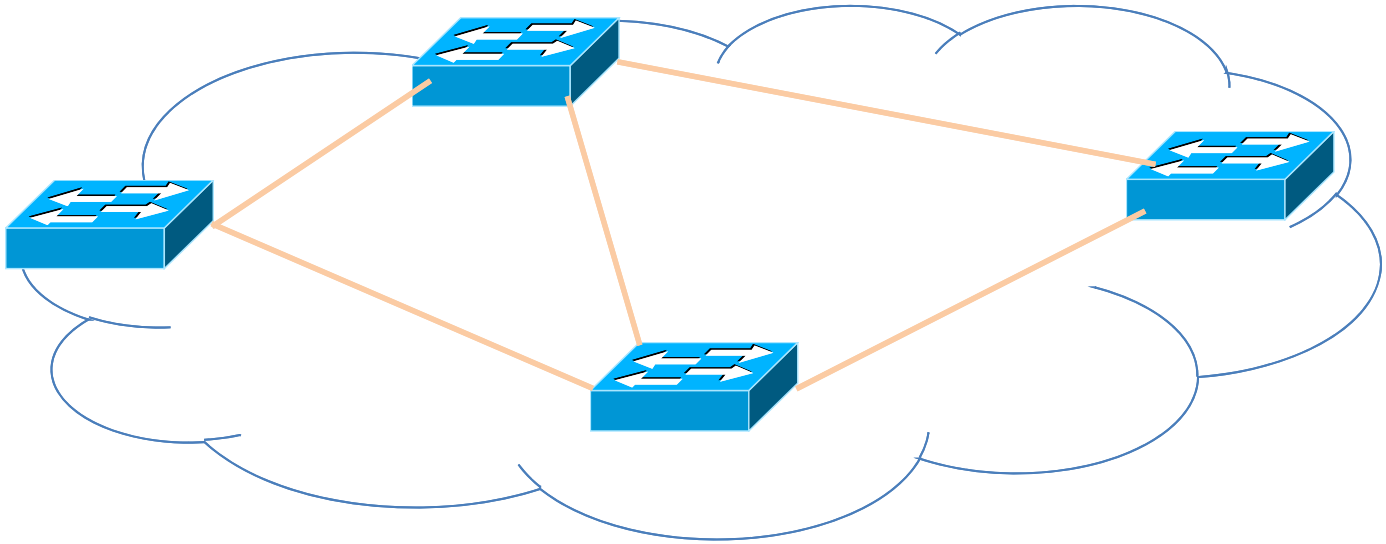
- Evolution over the time through many “patches”.
  - Subnets
  - Network Address Translation (NAT)
  - Mobile IP, Secure IP (IPSec)
  - Domain Name System (DNS)
- Inside the network infrastructure.
  - Hardware and software lock-in
    - Vendor-specific APIs
  - Slow protocol standardization
  - Few people can innovate:
    - Vendors write the code
    - Slow introduction of new features
- The network “ossification” impacts on performance, security, reliability, cost.



Rojas, M. A. T. (2014). SDN Basics Course – Ericsson.
Rexford, J. (2012). Software-Defined Networking Course. Source.

Traditional Networking (1)

Data plane:
Packet forwarding

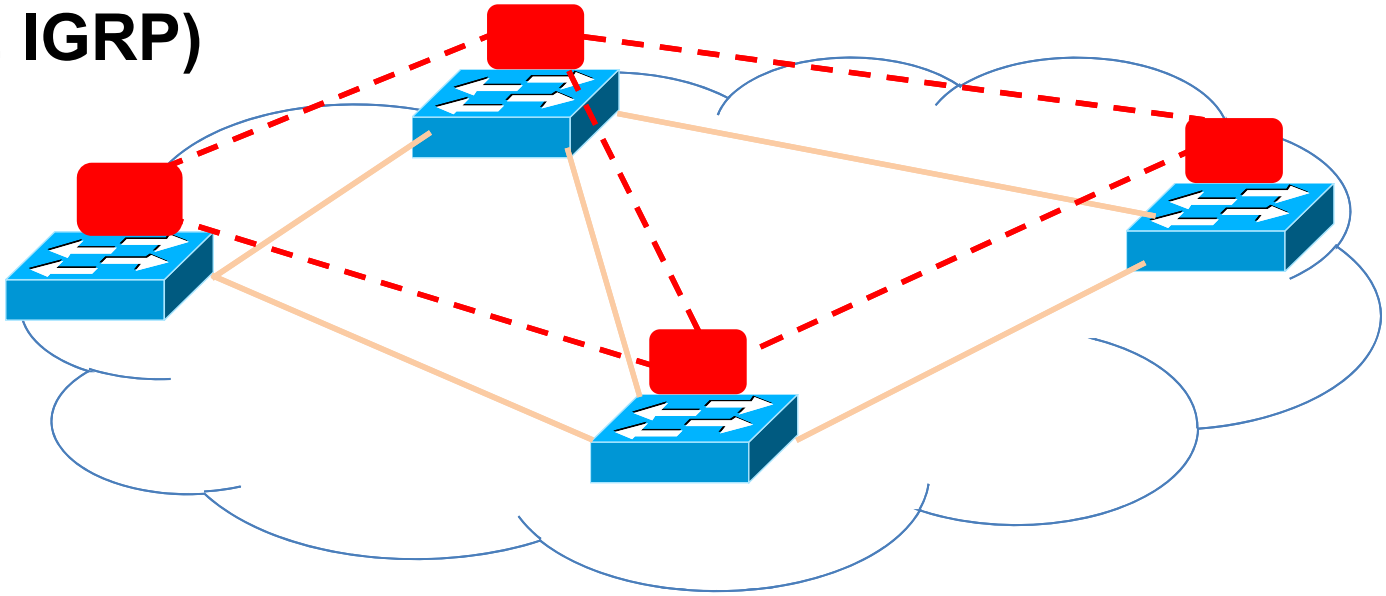


Forward, filter, buffer, mark, rate-limit, and measure packets

Rexford, J. (2012). Software-Defined Networking Course. Source.

Traditional Networking (2)

Control plane:
Distributed algorithms (BGP, OSPF, IGRP)

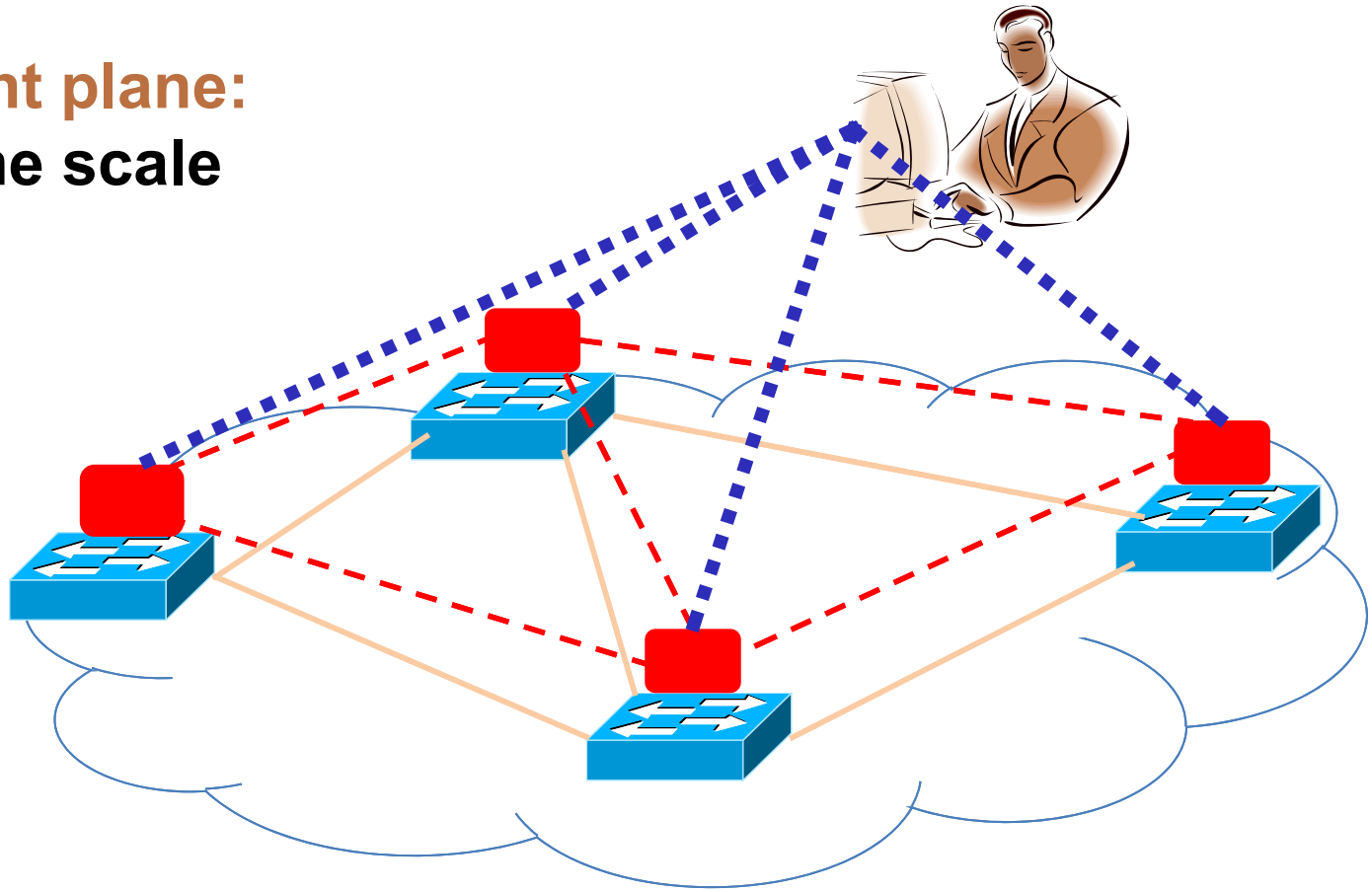


Track topology changes, compute routes, install forwarding rules

Rexford, J. (2012). Software-Defined Networking Course. Source.

Traditional Networking (3)

Management plane:
Human time scale

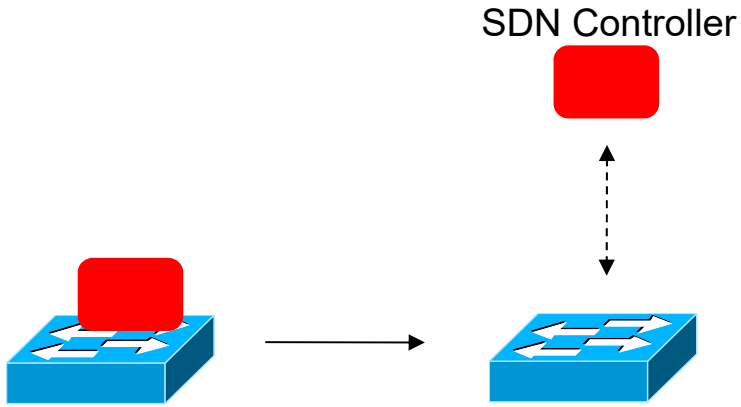


Collect measurements and configure the equipment

Rexford, J. (2012). Software-Defined Networking Course. Source.

Rethinking the Network Design

- Software-Defined Networking
  - Enable network programmability
- Simpler management
  - No need to “invert” control-plane operations
- Easier interoperability
  - Compatibility only in “wire” protocols
- Simpler, cheaper equipment
  - Minimal software/standardized software
- Enable innovation at much faster pace!
  - Less dependence on vendors and standards, open-source software

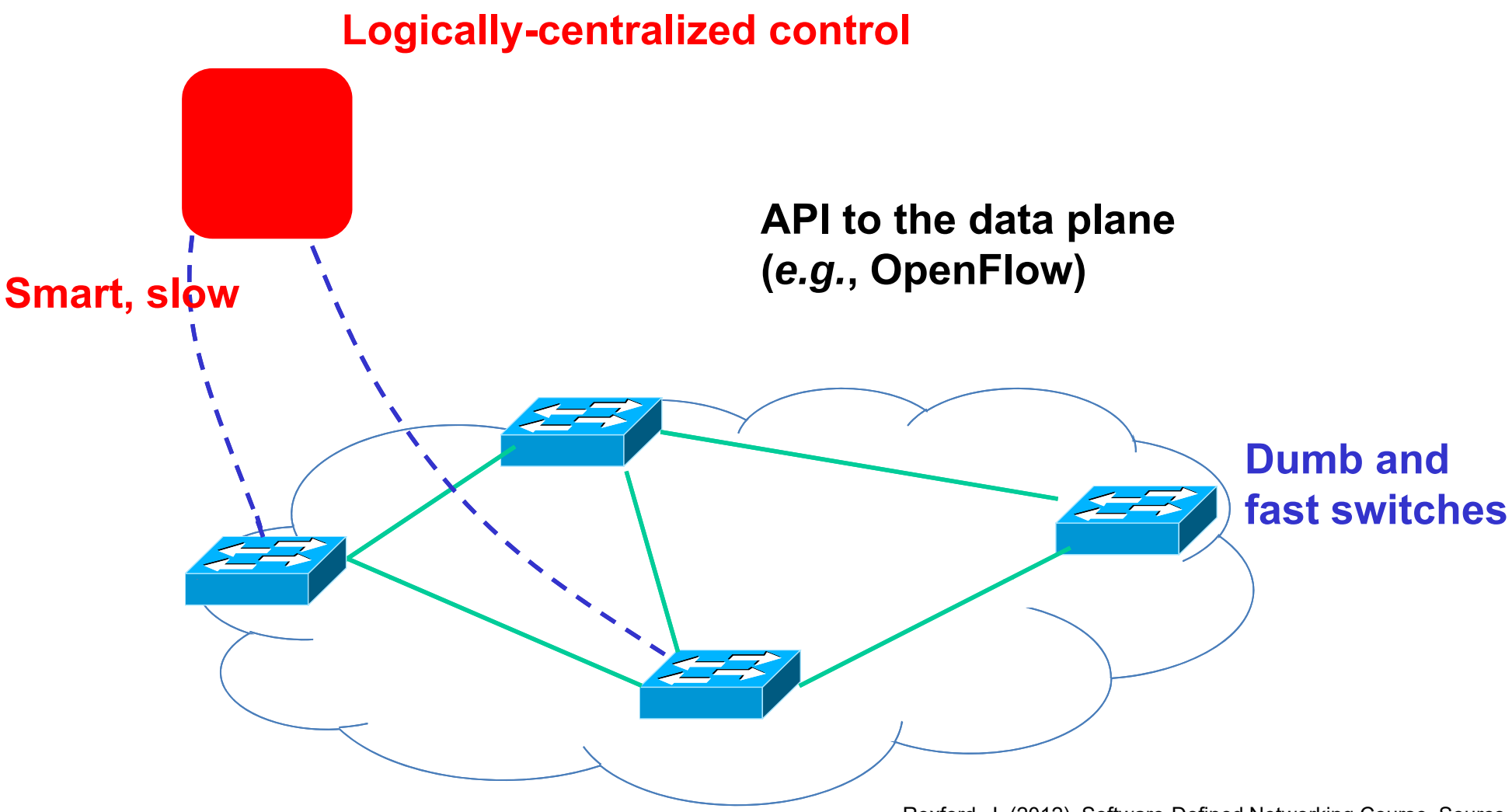


Data and control planes in the same device

Data and control planes in different device

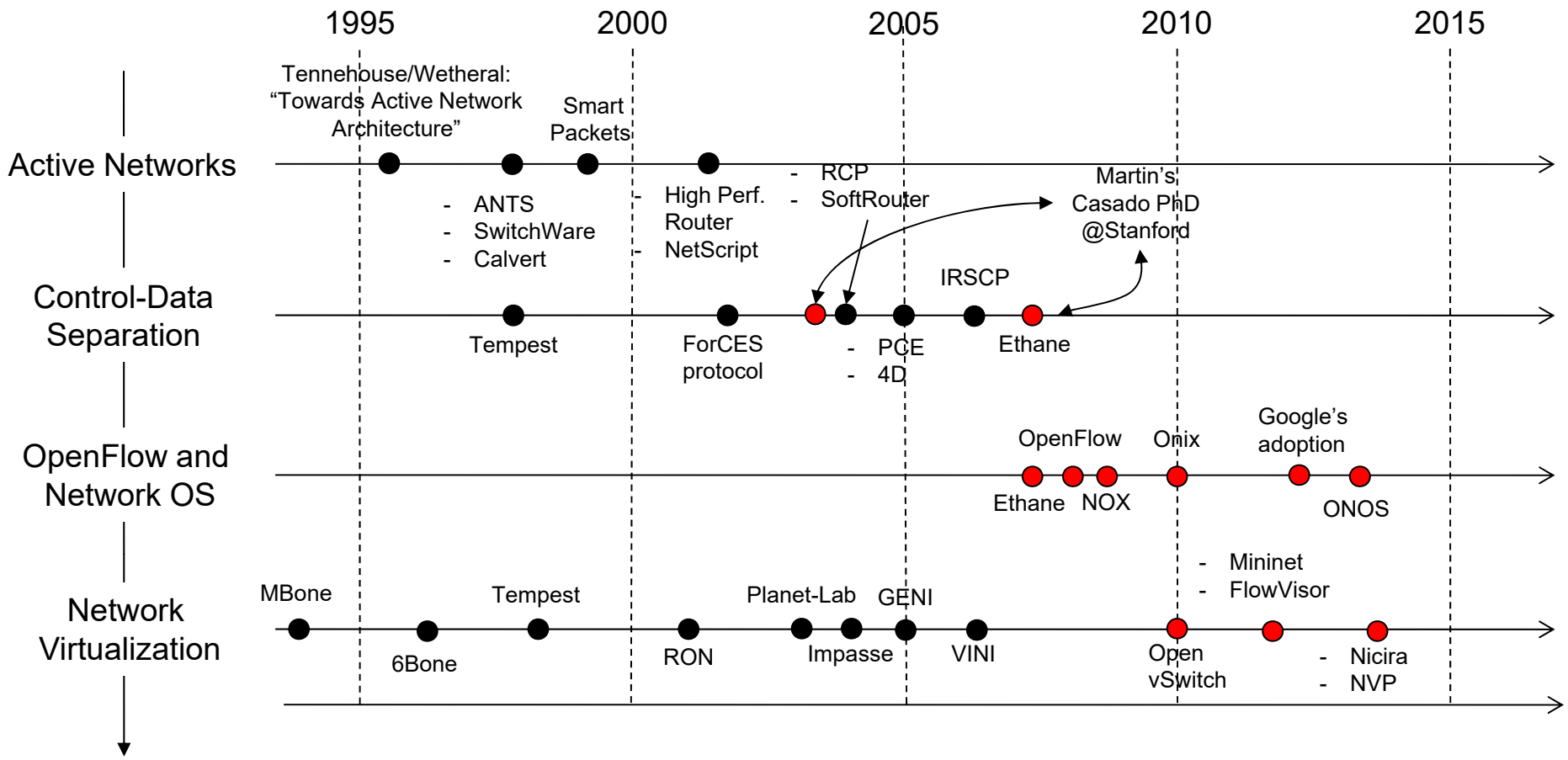
Rexford, J. (2012). Software-Defined Networking Course. Source.

Software-defined Networking (SDN)



Rexford, J. (2012). Software-Defined Networking Course. Source.

The Road to SDN



Feamster, N. *et al.* (2014). The Road to SDN

SDN Definitions

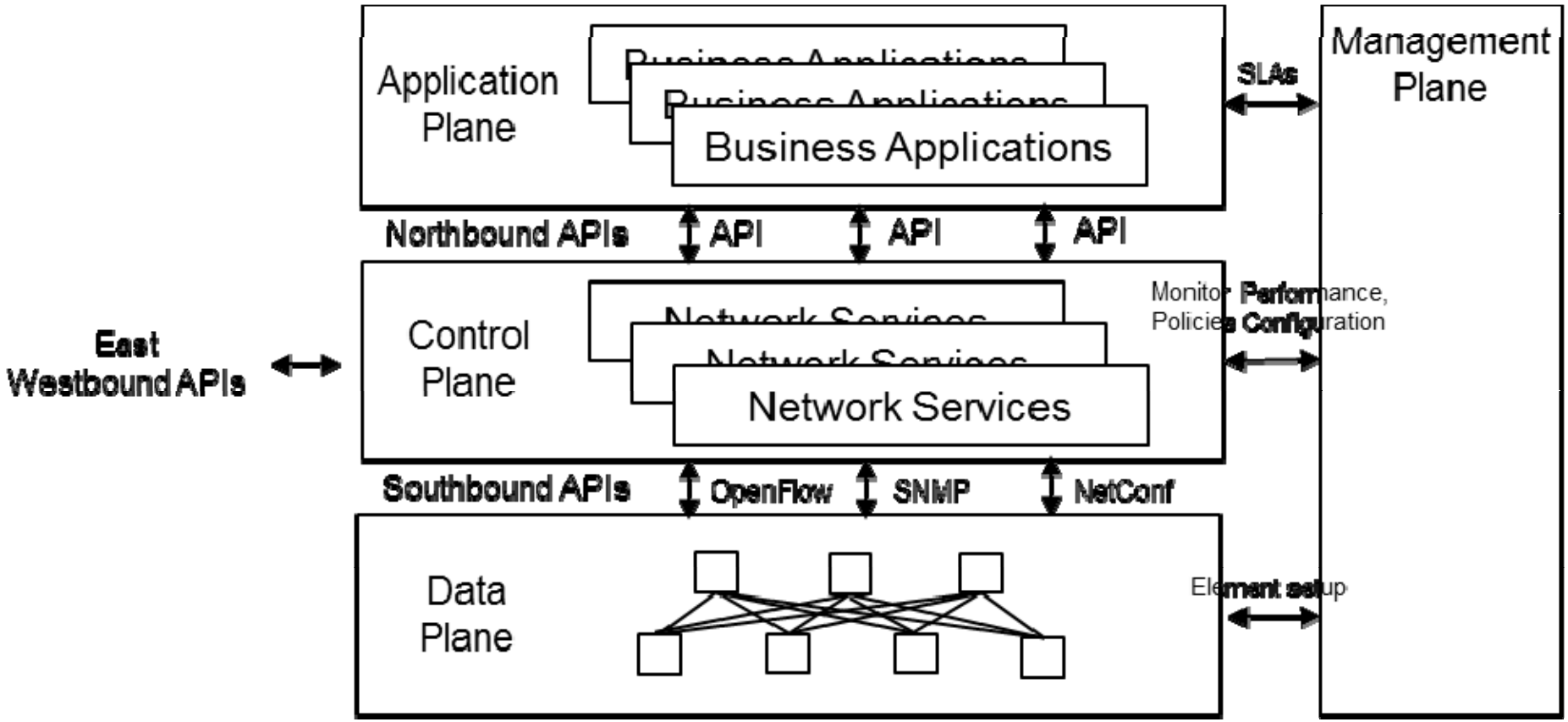
- "SDN has two defining characteristics: first, it separates the control plane (which decides how to handle the traffic) from the data plane (which forwards traffic according to decisions that the control plane makes). Second, it consolidates the control plane, so that a single software controls multiple data plane elements"

  Feamster *et al.* (2014), The Road to SDN.

- "A dynamic, manageable, cost-effective, and adaptable architecture. It decouples the network control and forwarding functions enabling the network control to become directly programmable." Open Networking Foundation (ONF).

- "SDN centralizes control of the network by separating the control logic to off-device computer resources" SDXcentral.

SDN Architecture Overview



Open Networking Foundation. SDN Architecture Overview. Source.

Architecture Characteristics

- Programmable: control is decoupled from the data plane
- Agile: control abstraction enable administrators to dynamically adjust the network
- Centrally managed: intelligence is logically centralized in the control plane devices that maintain a global view of the network
- Programmatically configured: managers can configure, manage, secure and optimize network resources via dynamic, automated SDN programs
- Open standards-based and vendor-neutral: instructions are provided by SDN controllers instead of multiple, vendor-specific devices and protocols

Open Networking Foundation. SDN Architecture Overview. Source.

Architecture Elements

- SDN Applications
  - Programs that explicitly, directly, and programmatically communicate their network requirements and desired behavior to the SDN Controller via Northbound interfaces (*e.g.*, firewall, load balancer, deep packet inspection).
- SDN Controller
  - Logically centralized entity in charge of (i) translating requirements from SDN Applications down to the data plane devices and (ii) providing an abstract view of the network to SDN Applications.
- NBI, SBI, East/Westbound Interfaces
  - Northbound interfaces (NBI): communication app ←→ controller
  - Southbound interfaces (SBI): communication controller ←→ devices
  - East/Westbound interfaces: communication controller ←→ controller
- Network Devices
  - Devices which exposes visibility and non-contended control over its advertised forwarding and data processing capabilities.

Open Networking Foundation. SDN Architecture Overview. Source.

Examples of Controllers

C++/Linux
Nicira/VMware

Python
Nicira/VMware

Java
Rice University

Java
Stanford

Ryu
Python
NTT

TREMA
Ruby/C
NEC

Java
Big Switch

Flowvisor

Java
On.Lab

Java
Linux Foundation

Java
Linux Foundation

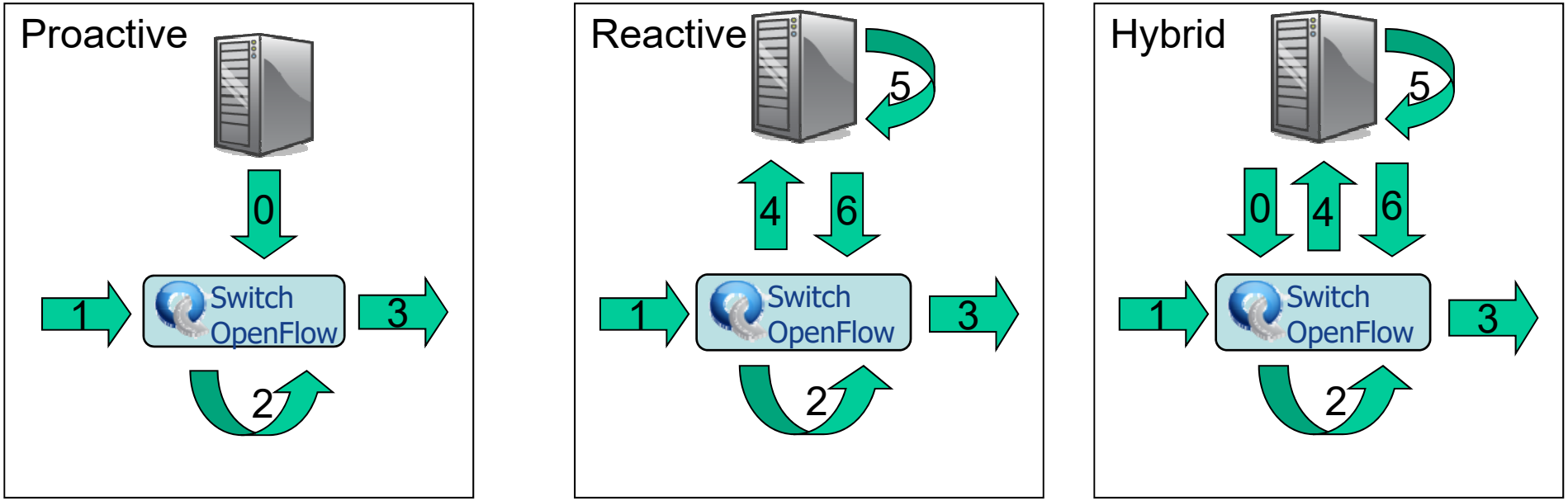
Most used in academy
for quick prototyping

Most used in industry
for production environments
(*e.g.*, cloud environment)

Controller Operation Modes (1)

- Proactive: flows are installed beforehand
  - Controller define flows in advance
  - Reduce overhead in the controller to define and create flows
  - Requires a high number of pre-defined rules
  - Loss of connectivity with the controller does not affect the network functioning
- Reactive: flows are created on demand
  - First packet of each flow is sent to the controller, which define and install a flow
  - Every new flow has a latency to define and install flow rules
  - Efficient usage of flow-tables
  - Loss of connectivity with the controller affects drastically the network functioning
- Hybrid: flows are created beforehand and adjusted on demand
  - Combination of proactive and reactive operation modes

Controller Operation Modes (2)



0 - Controller install flows

1 - Packet arrives

2 - switch check rules in flow-table

3 - packet is forwarded in case of matching or dropped

4 – packet is sent to the controller

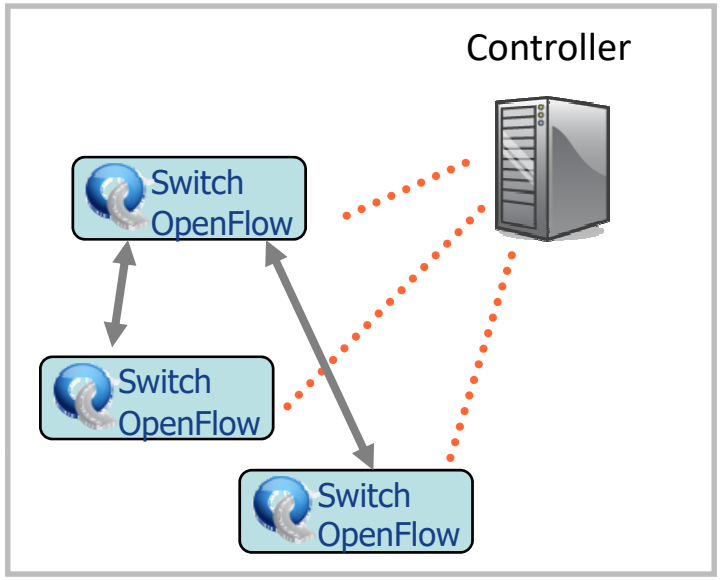
5 – controller define action

6 – controller install flows

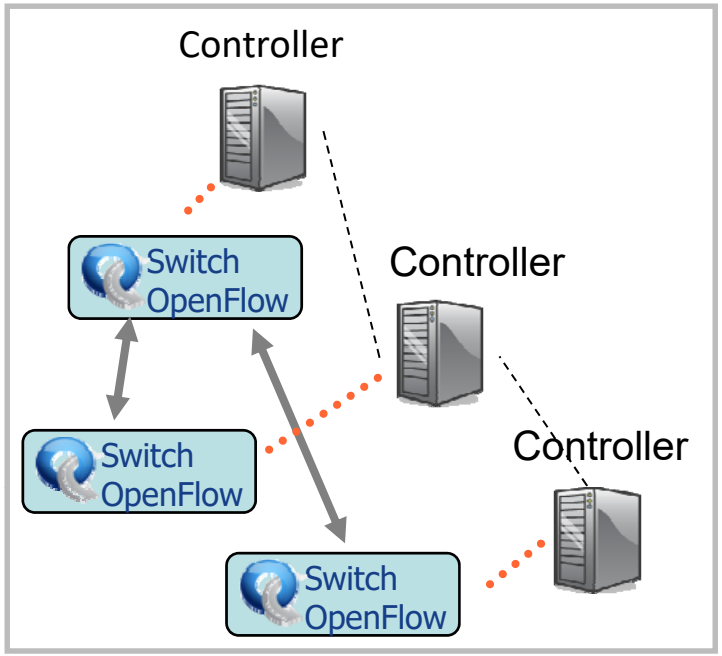
Rojas, M. A. T. (2014). SDN Basics Course – Ericsson.

Controller Placement (1)

- Controller hierarchy and placement.
  - Need to ensure low latency between controller(s) and node(s)
  - Multiple controllers to increase fault-tolerance, reliability



Single controller

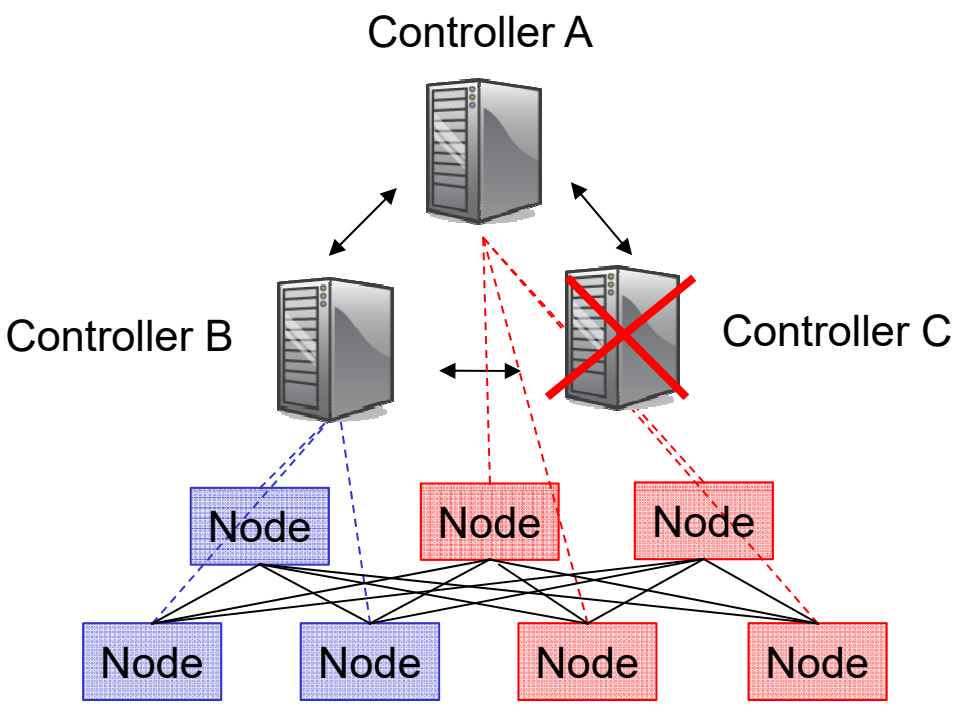


Multiple controllers

Rojas, M. A. T. (2014). SDN Basics Course – Ericsson.

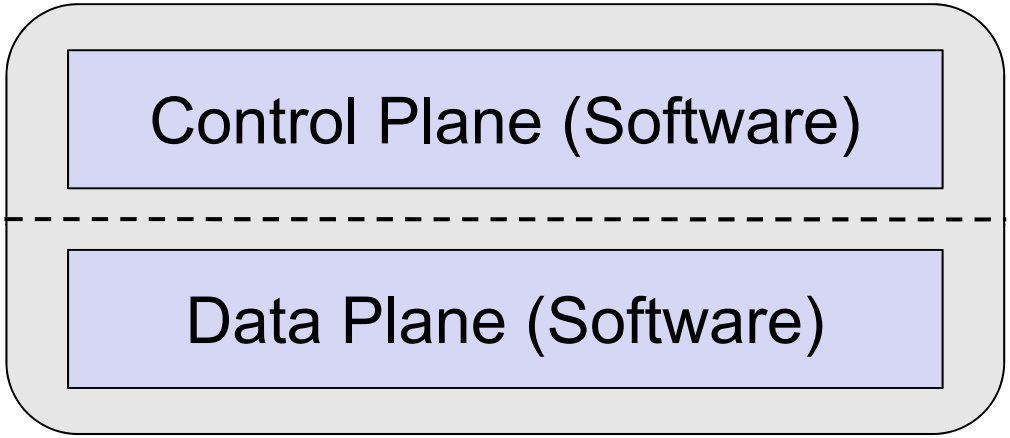
Controller Placement (2)

- Example
  - Redundancy to increase network reliability. If controller C fails, Controller A assume the control over its nodes.



Network Device (1)

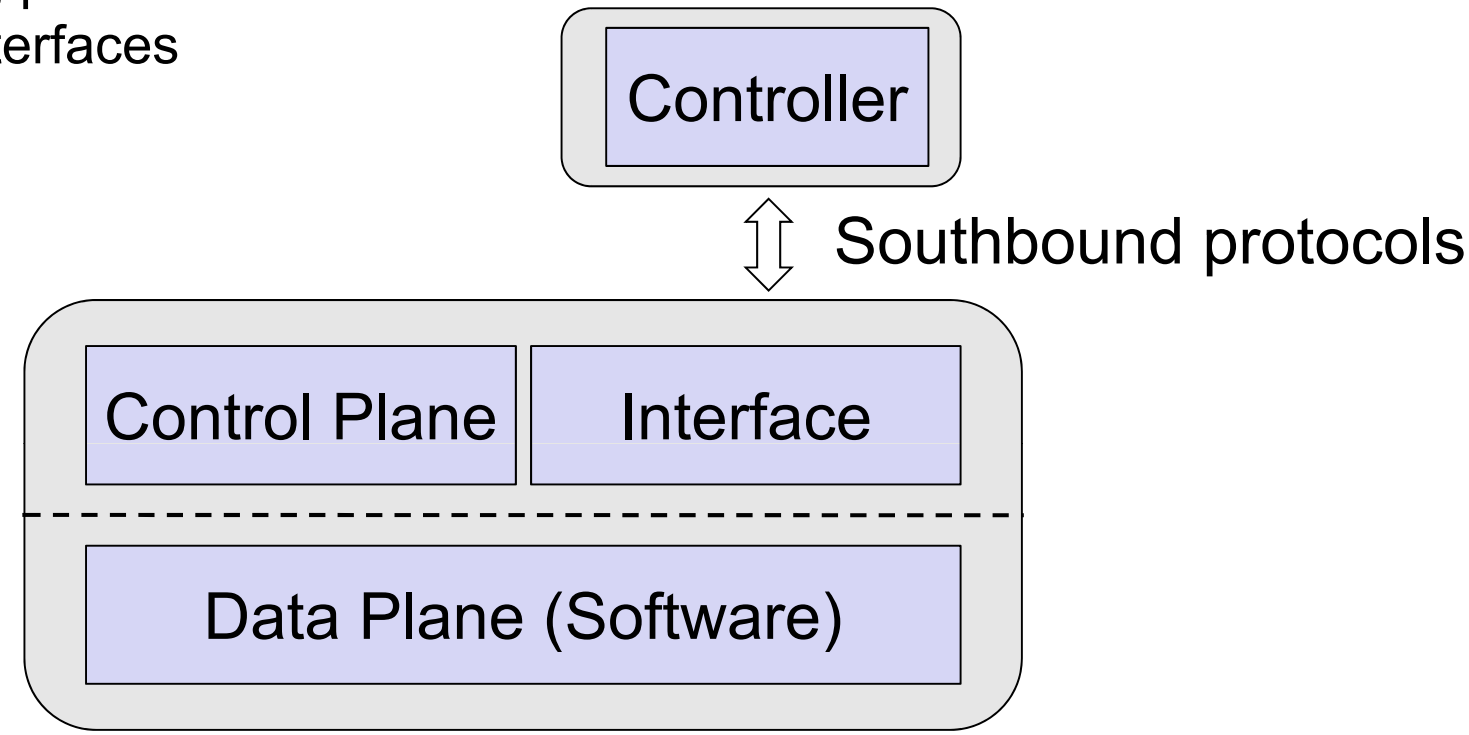
- Standard network device
  - Control plane:
    Responsible for network control operations, decision on routes and exchange of information with other switches (resources and connections)
  - Data plane:
    Forwarding packets to the physical interfaces



Rojas, M. A. T. (2014). SDN Basics Course – Ericsson.

Network Device (2)

- OpenFlow network device
  - Control plane:
    Responsibility to define operations and routes is abstracted to controller
  - Data plane:
    Forwarding packets to physical interfaces



Rojas, M. A. T. (2014). SDN Basics Course – Ericsson.

OpenFlow Protocol

- Is an open standard communication protocol between the control and data planes

- Its structure consists of:

  – A flexible structure to define flows characteristics using (or not) wildcards, e.g., a flow can be identified by a set of characteristics or just by source/destination IP addresses.

  – Multiple flow tables to match flows and enforce actions. Devices contain 1 to N flow tables to match pre-defined flow rules (set of characteristics) and enforce actions such as forward to a set of ports or drop.

  – Set of counters to obtain usage statistics. Counters are maintained for each flow table, flow entry, port, queue, group, group bucket, and so on.

Mackeon *et al.* (2008).OpenFlow: Enabling Innovation in Campus Networks

OpenFlow Benefits

- Key benefits of OpenFlow vs. traditional network management protocols (NetConf, SNMP/MIBs, or proprietary interfaces)
  - Programmability instead of configurability
    - Instead of configure pre-defined actions of a network device it is possible to program its behavior at the SDN controller and enforce via OpenFlow
  - Flexibility to control how packets are forwarded by setting rules in flow-tables
    - Matching rules can be specific to single type of packet/application or broad to match different type of flows using wildcards
  - Network automation and integration.
    - SDN reduce network management complexity enabling the development of tools to automate different network functions and integrate different network applications
  - Open source
    - No proprietary solutions (*e.g.*, Cisco).

Mackeon *et al.* (2008).OpenFlow: Enabling Innovation in Campus Networks

OpenFlow Switch

- Traditional Devices
  - Routers and switches uses the RIB (Routing Information Base) and FIB (Forwarding Information Base)
  - They execute complex algorithms in the RIB to choose the best route in the FIB
  - RIB is usually implemented in software while FIB is implemented in hardware

- OpenFlow Devices
  - The controller executes complex algorithm to choose the best route to forward packets, installing rules in flow-tables of OpenFlow-enabled devices
  - Flow-tables are built using a special kind of memory: TCAM
  - Flow-rules are stored in the TCAM (Flow-tables) providing information to the FIB

TCAM: Ternary Content Addressable Memory

Main OpenFlow Messages (1)

Message	Type	Description
Hello	Controller -> Switch	Following the TCP handshake, the controller sends its version number to the switch
Hello	Switch -> Controller	The switch replies with its supported version number
FeatureRequest	Controller -> Switch	The controller asks to see which features are available
FeatureResponse	Switch -> Controller	The switch replies with a list of ports, port speeds, and supported tables and actions
SetConfig	Controller -> Switch	In this case, the controller asks the switch to set configuration parameters

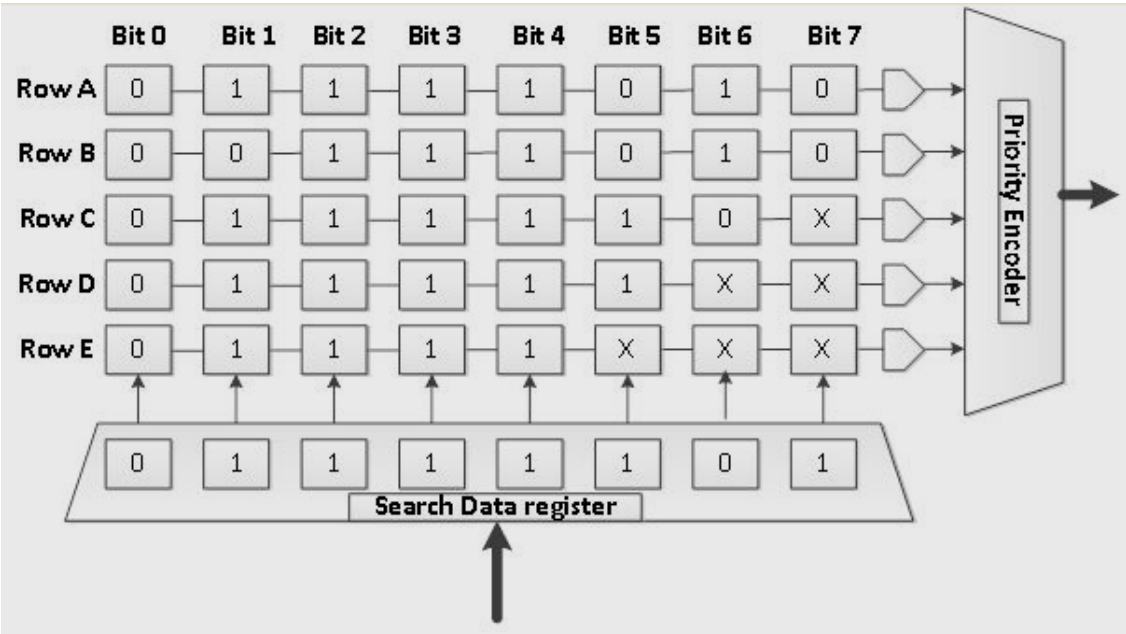
Main OpenFlow Messages (2)

Message	Type	Description
PacketOut	Controller -> Switch	Controller send a packet out one or more switch ports.
FlowMod	Controller -> Switch	Instructs a switch to add a particular flow to its flow table.
FlowRemoved	Switch -> Controller	the switch sends to the controller when a flow entry in a flow table is removed.
PacketIn	Switch -> Controller	a packet was received and it didn't match any entry in the switch's flow table, causing the packet to be sent to the controller.

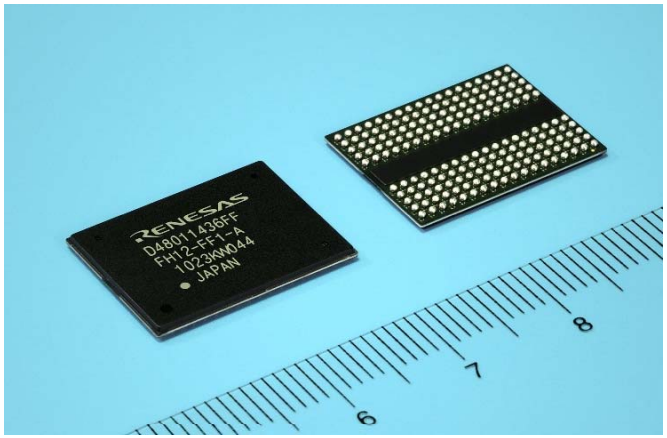
TCAM (1)

- TCAM supports the storage values {0,1 and X}, in which X can assume 0 or 1 (wildcard)

- TCAM is an expensive memory
  - The most expensive component in a switch



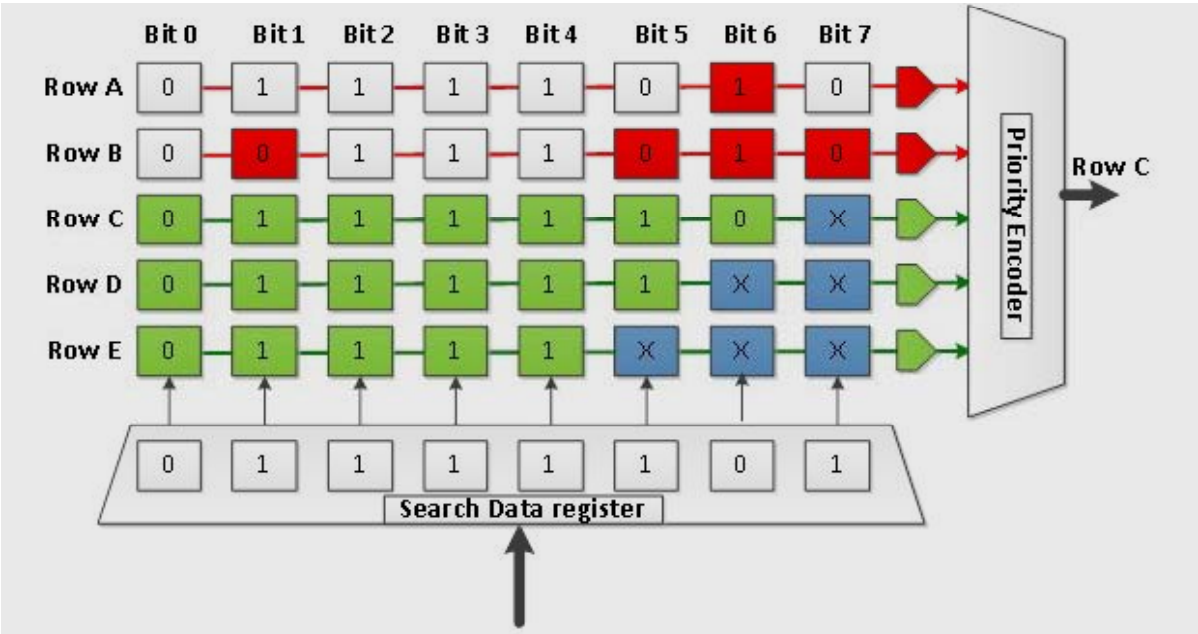
Example: Renesas TCAM chip (2010)



TCAM (2)

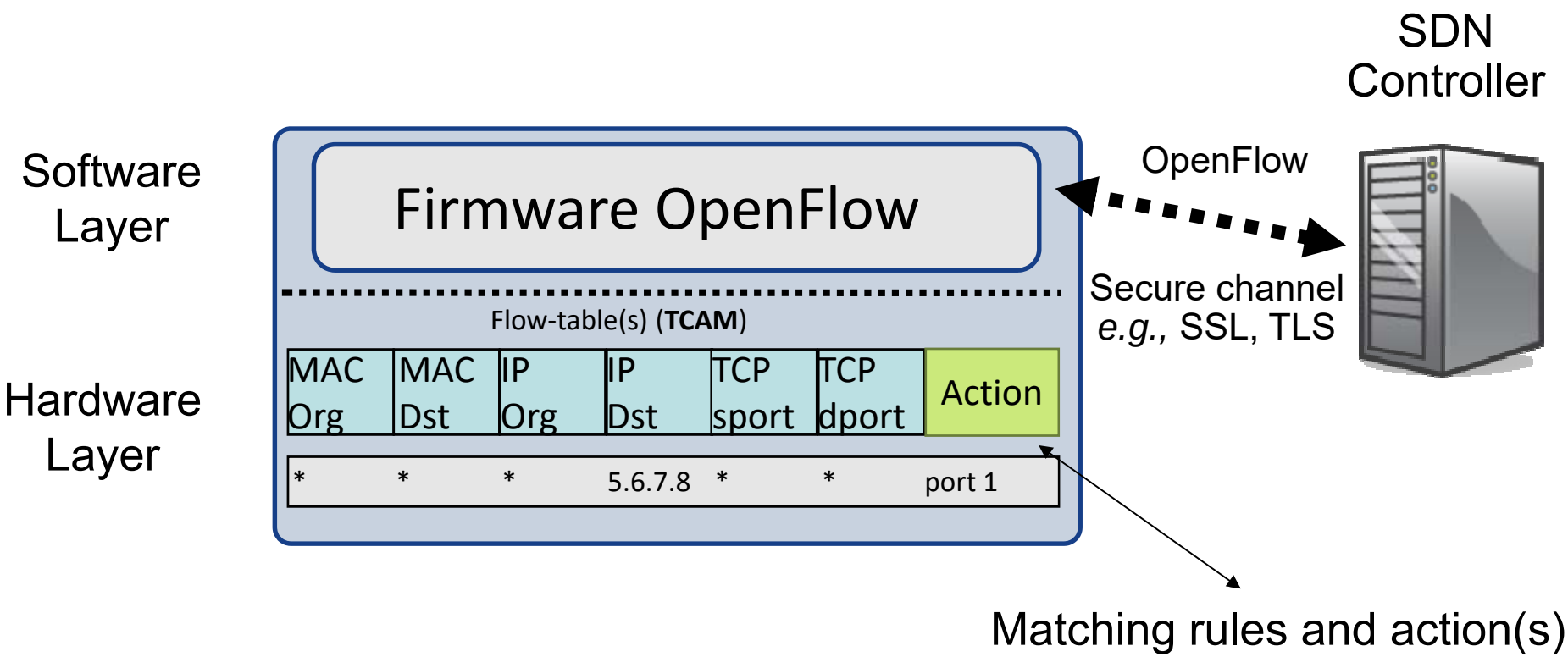
- TCAM is a specialized type of high-speed memory
  - Searches its entire contents in a single clock cycle

First matching rule



Takes 1 clock cycle

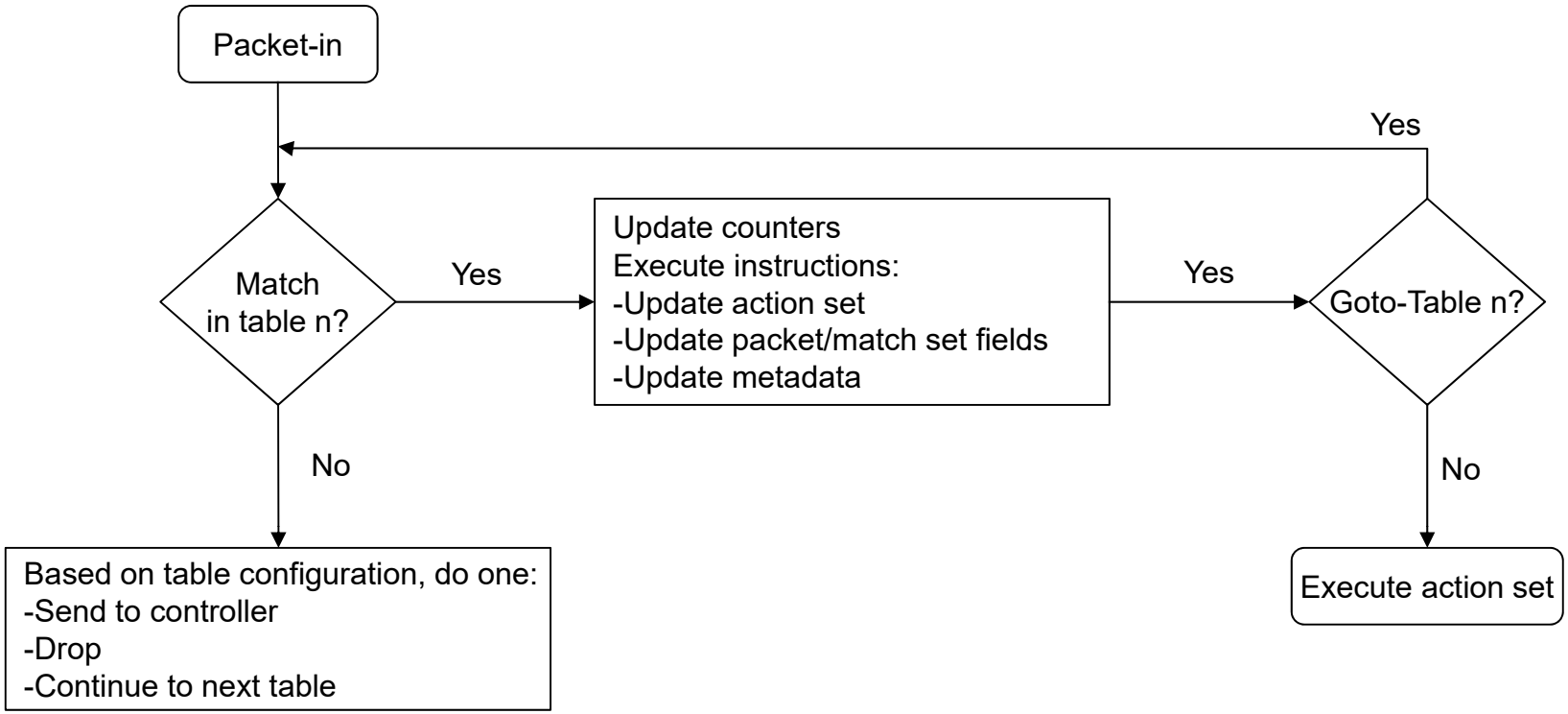
OpenFlow Switch



TCAM: Ternary Content Addressable Memory
SSL: Secure Socket Layer
TLS: Transport Layer Security

OpenFlow Matching

- Packet flowchart through an OpenFlow switch



OpenFlow Matching Rules

Rule	Action	Statistics

Packets and bytes counter

1. Forward packet(s) to port(s)
2. Encapsulate and forward to controller
3. Discard packet
4. Forward to normal processing pipeline

Port Src	MAC Src	MAC Dst	Eth Type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport

Matching Examples (1)

Switching: if matches the destination MAC, then forward to port 6

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port 6

Flow forwarding: if the flow matches, then forward to port 6

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
port3	00:20..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port 6

Firewall: drop all packets on port 22

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	*	*	*	22	drop

Matching Examples (2)

Routing: if matches the IP destination, then forward to port 6

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	5.6.7.8	*	*	*	Port 6

VLANs commutation: forward the tagged packet to different ports

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f..	*	vlan1	*	*	*	*	*	Port 6, Port 7, Port 9

OpenFlow Counters

- Counters are maintained per-table, per-flow, per-port and per queue. Are implemented in software and maintained by polling hardware counters with more limited ranges.

Counter	Bits
Per Table	
Active Entries	32
Packet Lookups	64
Packet Matches	64
Per Flow	
Received Packets	64
Received Bytes	64
Duration (seconds)	32
Duration (nanoseconds)	32
Per Port	
Received Packets	64
Transmitted Packets	64
Received Bytes	64
Transmitted Bytes	64
Receive Drops	64
Transmit Drops	64
Receive Errors	64
Transmit Errors	64
Receive Frame Alignment Errors	64
Receive Overrun Errors	64
Receive CRC Errors	64
Collisions	64
Per Queue	
Transmit Packets	64
Transmit Bytes	64
Transmit Overrun Errors	64

QoS Monitoring and QoS Enforcing

Polling can be done in a “smart way”, *e.g.*, adaptive monitoring schemes to avoid control traffic congestion

OpenFlow Protocol Evolution

2009

1.0

2011

1.1

Multiple table
- Group
- Not popular – incompatible with 1.0

2012

1.2

Extensible match support
- Controller role change mechanism

Refactor capabilities negotiation
- More flexible table miss support
- Per flow meters
- Auxiliary connections

2013

1.3

More extensible wire protocol
- Flow monitoring
- Eviction
- Vacancy events
- Bundles

2013

1.4

Egress tables
- Packet type aware pipeline
- Extensible flow entry stats
- Meter action
- Port properties for pipeline fields

2014

1.5

OpenFlow-enabled (v1.3) Devices (1)

Brocade MLX series

NEC IP8800

Ciena Coredirector

Juniper MX series

Netgear 7324

TPLink WR1043ND

HP 8200 zl

Pronto 3240/3290

OpenFlow-enabled (v1.3) Devices (2)

Cisco Nexus 3548 Switch

Ericsson Router 8801

Cisco Catalyst 6k

IBM G8264

Research Challenges (1)

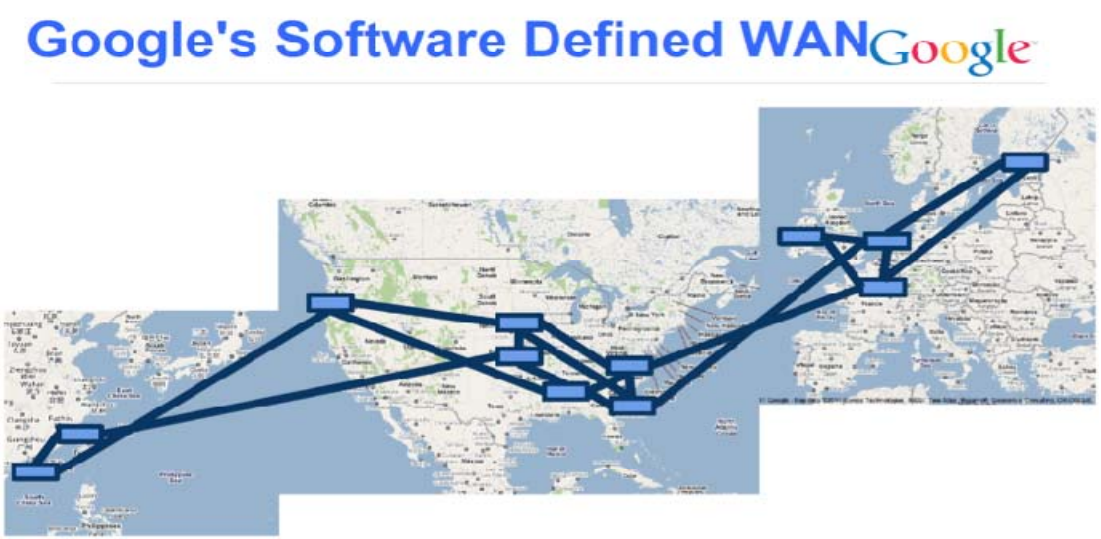
- Security
  - Controller logic security, redundancy and fault tolerance
  - Robust policy framework to ensure actions of the controller and its applications
  - Forensics and Remediation
- Energy Efficiency
  - Development of traffic-aware applications
  - Smart use of TCAM (power hungry)
- Development of applications based on the central network view
- Traffic engineering
  - Intra-domain, inter-domain, optimizations based on policies, orchestration and theory of control
- Controller optimization
  - Ensure high performance and reliability requirements with the lowest possible number of controllers

Research Challenges (2)

- Management of the SDN environment
  - Enforcement of rules and consistency
- Programmability and Debugging
  - Development of automated testing features
- Controller scalability
- Management in heterogeneous networks
  - OpenFlow networks x traditional networks
- Integration with others technologies
  - Such as NFV (Network Function Virtualization)
- Performance

Google WAN Managed by SDN

- WAN-intensive apps provided by Google
  - Data centers deployed across the world
  - Serve content with geographic locality
  - Replicate content for fault tolerance
  - Need a network to connect these data centers to one another
    - Not on the public Internet
    - Cost effective network for high volume traffic

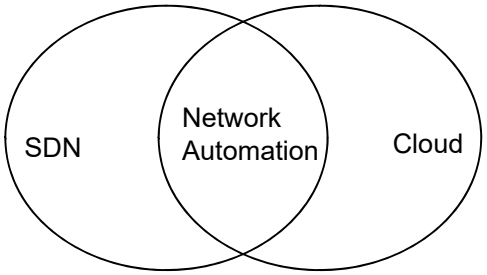


SDN and Cloud Networking (1)

- A quick reminder on clouds and networks:

  "*Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction".*

  P. Mell and T. Grance. The NIST definition of cloud computing, 2009.
- Enables ubiquitous, convenient and on-demand access to resources, such a networks, servers, storage, apps, and services
- Characteristics:
  - Rapidly provisioning and releasing of resources
  - Minimal management effort and service provider interaction.
- Some SDN benefits:
  - Service provisioning speed and agility
  - Network flexibility and holistic management
  - Better and more granular security
  - Efficiency and lower operating expenses



SDN and Cloud Networking (2)

- OpenStack (Cloud platform) + OpenDaylight (SDN platform)

- OpenStack provides support to OpenFlow controllers in its networking module (Neutron), which can be implemented to provision/release and manage virtual networks

SDN and Cloud Networking (3)

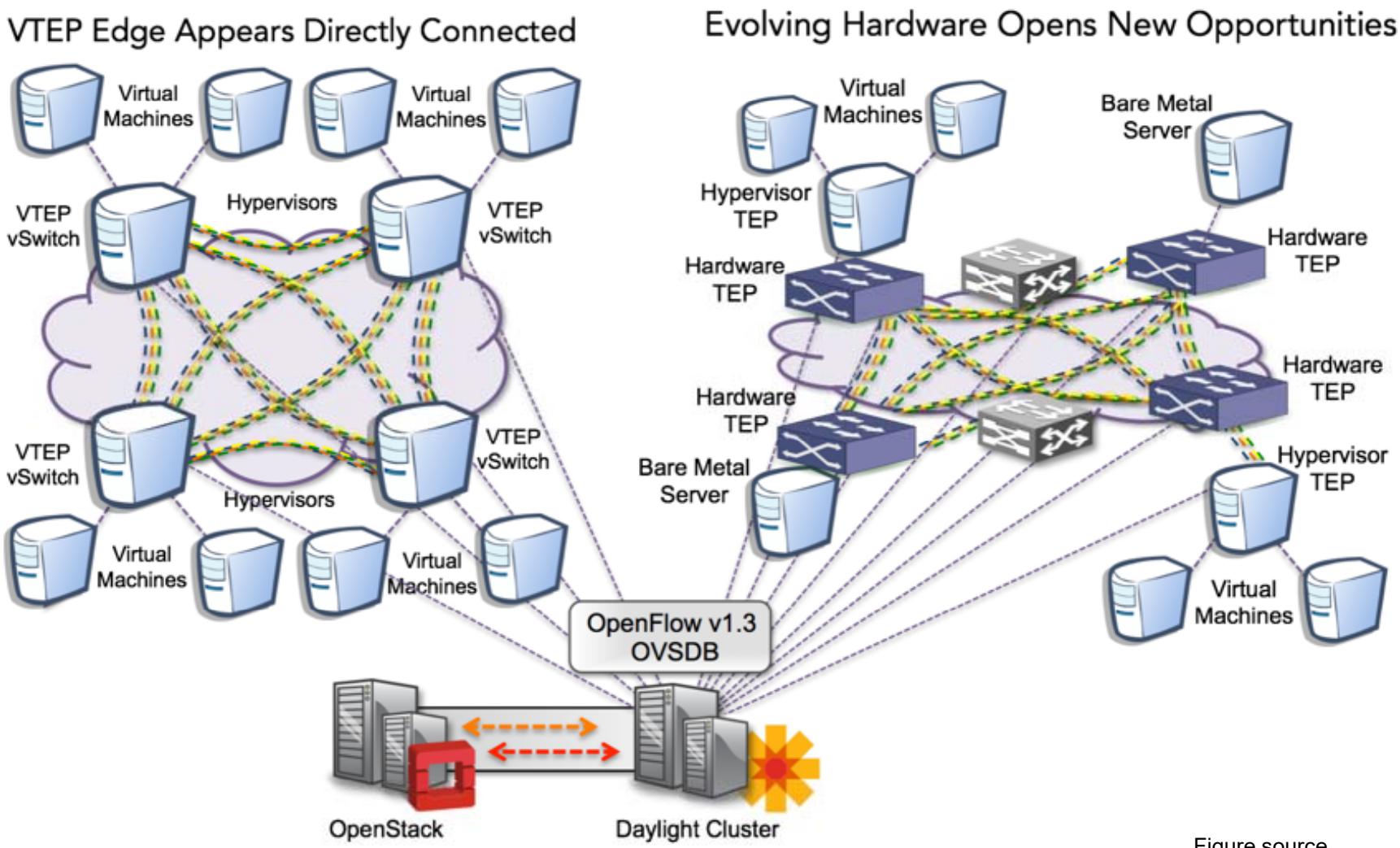


Figure source

Key SDN Characteristics

- The Good
  - Network-wide visibility
  - Direct control over switches
  - Simple data plane abstraction

- The Bad
  - Low-level programming interface
  - Functionality tied to hardware
  - Explicit resource control

- The Ugly
  - Non-modular, non-compositional
  - Programmers face challenging distributed programming problems

Rexford, J. (2014). TCP Performance Monitoring (and Pyretic background). Source

References (1)

- T. Koistinen: Protocol overview: RTP and RTCP; https://www.netlab.tkk.fi/opetus/s38130/k99/presentations/4.pdf
- L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala: *RSVP: A New Resource ReSerVation Protocol;* IEEE Network, Vol. 7, No. 5, September 1993, pp 8 – 18
- B. A. Forouzan: TCP/IP Protocol Suite, McGraw Hill Companies, 2010
- C. Rensing, H. Hasan, M. Karsten, B. Stiller: A*AA: A Survey and Policy-based Architecture and Framework;* IEEE Network Magazine, Vol. 16, No. 6, December 2002.
- H.-G. Hegering, S. Abeck, B, Neumair: *Integrated Management of Networked Systems,* Morgan Kaufmann, San Francisco, California, U.S.A., 1999
- W. Stallings: *SNMP, SNMPv2, and CMIP: The Practical Guide to Network Management Standards.* Addison-Wesley, Reading, Massachusetts, U.S.A., 1993.
- G. Rosenbaum, W. Lau, S. Jha: *Recent Directions in Virtual Private Network Solutions;* 11th IEEE International Conference on Networks 2003, September 28 - October 1, 2003, Sydney, NSW, Australia, pp 217-223
- R. Buyya, M. Pathan: *Content Delivery Networks: Overlay Networks for Scaling and Enhancing the Web;* 16th International Conference on Advanced Computing and Communications, ADCOM 2008, December 14-17, 2008

References (2)

- H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson: RTP: *A Transport Protocol for Real-Time Applications;* IETF RFC 3350, July 2003
- R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin: *Resource ReSerVation Protocol (RSVP);* IETF RFC 2205, September 1997
- R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson: *Stream Control Transmission Protocol;* IETF RFC 2960, October 2000
- R. Stewart (Edts.): *Stream Control Transmission Protocol;* IETF RFC 4960, Sept. 2007
- C. de Laat, G. Gross, L. Gommans, J. Vollbrecht, D. Spence: *Generic AAA Architecture;* IETF RFC 2903, August 2000
- J. Case, R. Mundy, D. Partain, B. Stewart: *Introduction to Version 3 of the Internet-standard Network Management Framework;* IETF RFC 2570, April 1999
- D. Harrington, R. Presuhn, B. Wijnen: *An Architecture for Describing SNMP Management Frameworks;* IETF RFC 2571, April 1999
- L. Andersson, T. Madsen: *Provider Provisioned Virtual Private Network (VPN) Terminology;* IETF RFC 4026, March 2005
- M. Oskar van Deventer: *ETSI Standards for CDNs;* International Broadcasting Convention IBC 2011, September 9-11, 2011, Amsterdam, The Netherlands

References (3)

- N. McKeown et al.: *OpenFlow: Enabling Innovation in Campus Networks,* ACM SIGCOMM Computer Communication Review, Vol. 38, No. 2, 2008, pp. 69-74.
- M. D. D. Moreira et al.: *Internet do futuro: Um novo horizonte,* Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2009, pp. 1-59.
- J. Rexford: *Software-Defined Networking Course.* Teaching slides 2012.
- Feamster, Nick, Jennifer Rexford, Ellen Zegura: *The Road to SDN,* IEEE Queue, Vol. 11, No. 12, 2013, p 20.
- Aiko Pras, Jürgen Schönwälder, Mark Burgess, Olivier Festor, Gregorio Martínez Pérez, Rolf Stadler, Burkhard Stiller: *Key Research Challenges in Network Management,* IEEE Communications Magazine; Vol. 45. No. 10, 2007, pp. 104-110.
- Open Networking Foundation: *SDN Architecture Overview;.* 2014.
- D. Kreutz, Fernando M. V. Ramos, Paulo Esteves Veríssimo, Christian Esteve Rothenberg, Siamak Azodolmolky, Steve Uhlig: *Software-defined Networking: A Comprehensive Survey.* Proceedings of the IEEE, Vol. 103, No. 1, 2015, pp. 14-76.
- M. A. T. Rojas: *SDN Basics Course*, Ericsson 2014.

References (4)

- Wojciech Burakowski,Andrzej Beben, Hans van den Berg,Joost W. Bosman, Gerhard Hasslinger, Attila Kertesz, Steven Latre, Rob D. van der Mei, Tamas Pflanzner, Patrick Poullie, Maciej Sosnowski, Bart Spinnewyn, and Burkhard Stiller. Traffic Management for Cloud Federation. In: Rob D. van der Mei, Hans van den Berg, and Ivan Ganchev (Editors), Autonomous Control for a Reliable Internet of Services: Methods, Models, Approaches, Techniques, Algorithms and Tools. Springer, Heidelberg, Germany, 2017.

- Patrick Poullie, Thomas Bocek, and Burkhard Stiller. A Survey of the State- of-the-Art in Fair Multi-resource Allocations for Data Centers. To appear in IEEE Transactions on Network and Service Management, TNSM ’17, 2017.

- Patrick Poullie, Stephan Mannhart, and Burkhard Stiller. Virtual Machine Priority Adaption to Enforce Fairness Among Cloud Users. 12th International Conference on Network and Service Management, CNSM’16, Montreal, Canada, October 2016.

- Patrick Poullie and Burkhard Stiller.Cloud Flat Rates Enabled via Fair Multi-resource Consumption. 10th International Conference on Autonomous Infrastructure, Management and Security, AIMS’16, Vol. 9701 of Lecture Notes in Computer Science, pp 30–44, Munich, Germany, June 2016.

- Patrick Poullie and Burkhard Stiller. The Design and Evaluation of a Heaviness Metric for Cloud Fairness and Correct Virtual Machine Configurations. 13th Conference on the Economics of Grids, Clouds, Systems, and Services, GECON’16, Vol. 10382 of Lecture Notes in Computer Science, pp 173–178, Athens, Greece, September 2016.

References (5)

[1] https://content.pivotal.io/blog/rabbitmq-hits-one-million-messages-per-second-on-google-compute-engine

[2] http://www.zdnet.com/article/whatsapp-now-one-billion-people-send-55-billion-messages-per-day/

[3] https://www.rabbitmq.com/tutorials/

[4] Jorge Luzuriaga, Pablo Boronat, Miguel Perez, Carlos Calafate, Juan-Carlos Cano, and Pietro Manzoni: *A comparative evaluation of AMQP and MQTT protocols over unstable and mobile networks*. IEEE Consumer Communications and Networking Conference (CCNC), 2015, Las Vegas, NV, vol. 12. DOI: 10.1109/CCNC.2015.7158101

[5] AMPQ Protocol Specification, Version 0.9.1, 13th November 2008. Available at: http://www.amqp.org/specification/0-9-1/amqp-org-download

[6] S. Vinoski: *Advanced Message Queuing Protocol*. IEEE Internet Computing, vol. 10, no. 6, pp. 87-89, Nov-Dec. 2006. DOI: 10.1109/MIC.2006.116

[7] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash: *Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications*. IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2347-2376, 2015. DOI: 10.1109/COMST.2015.2444095