UNIVERSITETET I OSLO

[simula . research laboratory]

Transport Layer

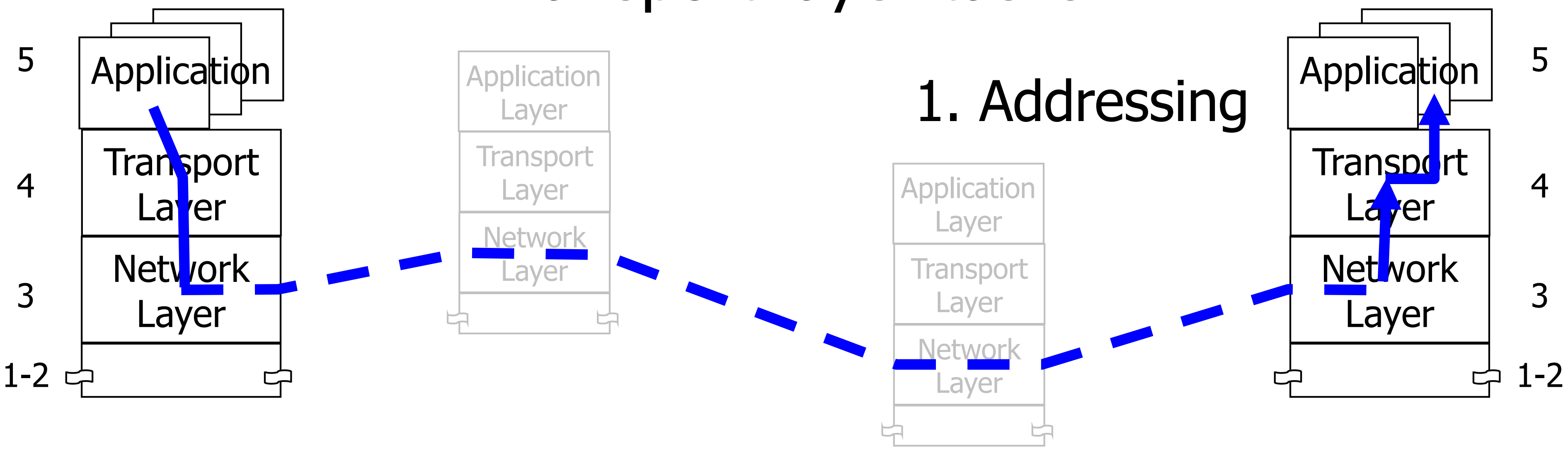
INF3190

Foreleser: Carsten Griwodz
Email: griff@ifi.uio.no

(Including slides from Michael Welzl)

Transport Layer Function

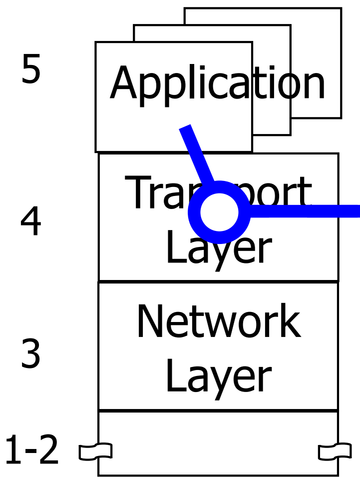
Transport layer tasks



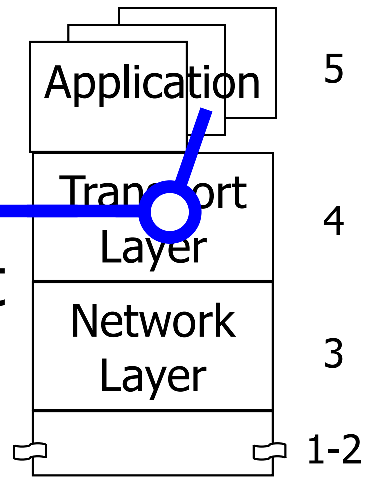
simula

Transport Layer Function

Transport layer tasks

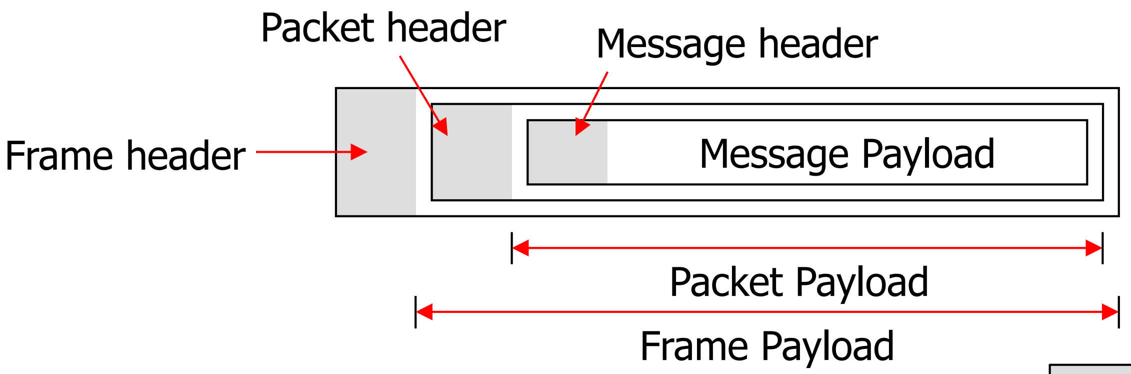


1. Addressing
2. End-to-end connection management
3. Transparent data transfer between end points
4. Quality of service
   - Error recovery
   - Reliability
   - Flow control
   - Congestion control



Transport Service: Terminology

- Nesting of messages, packets, and frames



Layer	Data Unit
Transport	Message
Network	Packet
Data link	Frame
Physical	Bit/symbol (bitstream)

TCP/IP Message

ISO TPDU (transport protocol data unit)

TCP name for Message Payload: **Segment**

UDP name for Message: **Datagram**

Internet terminology

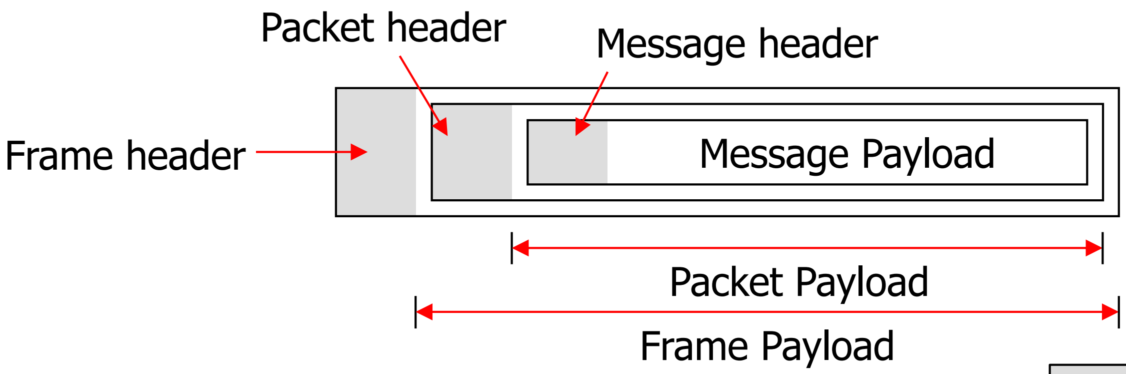
- TCP segment could be divided across multiple IP packets (fragmentation)
  - hence different words used

- In practice, this is inefficient and not often done

- hence normally ***message*** = ***packet***
  - and in everyday work, although obviously not the same: packet ≈ message or packet ≈ segment
  - distinction from context

Transport Service: Terminology

- Nesting of messages, packets, and frames



Layer	Data Unit
Transport	Message, **Packet**
Network	Packet
Data link	Frame
Physical	Bit/symbol (bitstream)

TCP/IP Message

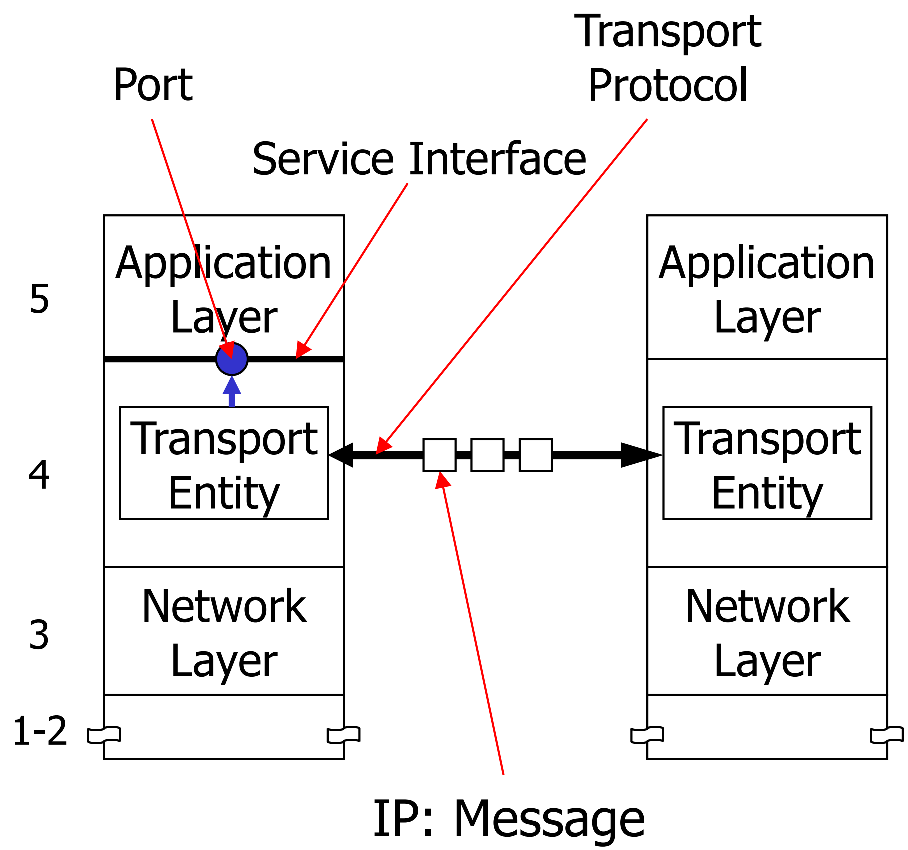
ISO TPDU (transport protocol data unit)

TCP name for Message Payload: **Segment**

UDP name for Message: **Datagram**

Transport Service

- Connection oriented service
  - 3 phases
    - connection set-up
    - data transfer
    - disconnect
- Connectionless service
  - Transfer of independent messages
- Realization: transport entity
  - Software and/or hardware
  - Software part usually contained within the kernel (process, library)



TCP/IP	1. Port 2. IP address !!
ISO	TSAP (transport service access point)

Ambiguities: bandwidth (1/2)

Traditional, "real" definition!

Mirriam-Webster online (http://www.m-w.com):

- Main Entry: band*width, Pronunciation: 'band-"width
  Function: noun, Date: circa 1937
  - 1 : a range within a band of wavelengths, frequencies, or energies; especially : a range of radio frequencies which is occupied by a modulated carrier wave, which is assigned to a service, or over which a device can operate
  - 2 : the capacity for data transfer of an electronic communications system <graphics consume more bandwidth than text does>; especially : the maximum data transfer rate of such a system

"Information rate"

- Unit: definition 1 - "Hz", definition 2 - "bit/s" (bps)

- Common terminology in computer network context:
  How many bits/sec can be transferred = "how thick is the pipe"

Ambiguities: bandwidth (2/2)

- Various wooly “bandwidth“ terms
  - Nominal bandwidth: Bandwidth of a link when there is no traffic
  - Available bandwidth: (Nominal bandwidth - traffic) ... during a specific interval
  - Bottleneck bandwidth: smallest nominal bandwidth along a path, but sometimes also smallest available bandwidth along a path

- Throughput: bandwidth seen by the receiver

- Goodput: bandwidth seen by the receiving application
  - e.g. TCP: goodput != throughput
  - Difference is not only message headers - more required to ensure reliability, ordering, flow control, congestion control, ...

Ambiguities: delay

- Latency - time to transfer an "empty" message
  - also: “propagation delay”
  - limit: speed of light!

- End2end delay = *latency* + *msg_length / bottleneck bandwidth*
  + queuing delay
  - just a rough measure; e.g., processing delay can also play a role, esp. in core routers (CPU = scarce resource!)

- Jitter - delay fluctuations, very critical for most real-time applications

- Round-trip time (RTT) - time a messages needs to go from sender to receiver and back

More ambiguities

- mbit / mb / Mb / Mbit ... ?
- Latency: sometimes end2end-delay
- link: physical connection between one or more hosts or routers, or link between IP routers (may consist of multiple physical links!)
- capacity: often physical capacity, but different if you talk about TCP
- In general:
  make sure you know which layer you are talking about!

Transport Service

- Similar services of
  - Network layer and transport layer
  - Why 2 Layers?

- Network service
  - Not to be self-governed or influenced by the user
  - Independent from application & user
    - enables compatibility between applications
  - Provides for example
    - "only" connection oriented communications
    - or "only" unreliable data transfer

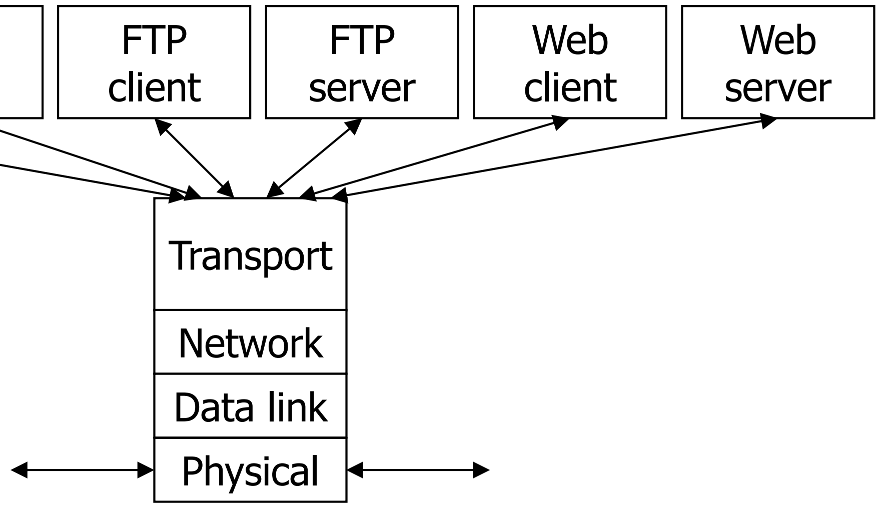
- Transport service
  - To improve the network services that
    users and higher layers want to get from the network layer, e.g.
    - reliable service
    - necessary time guarantees

Transport Service

- Transport protocols of TCP/IP protocols
  - Services provided implicitly (ISO protocols offer more choice)

	UDP	DCCP	TCP	SCTP
Connection-oriented service		X	X	X
Connectionless service	X			
Ordered			X	X
Partially Ordered				X
Unordered	X	X		X
Reliable			X	X
Partially Reliable				X
Unreliable	X	X		X
With congestion control		X	X	X
Without congestion control	X			
Multicast support	X	X		
Multihoming support				X

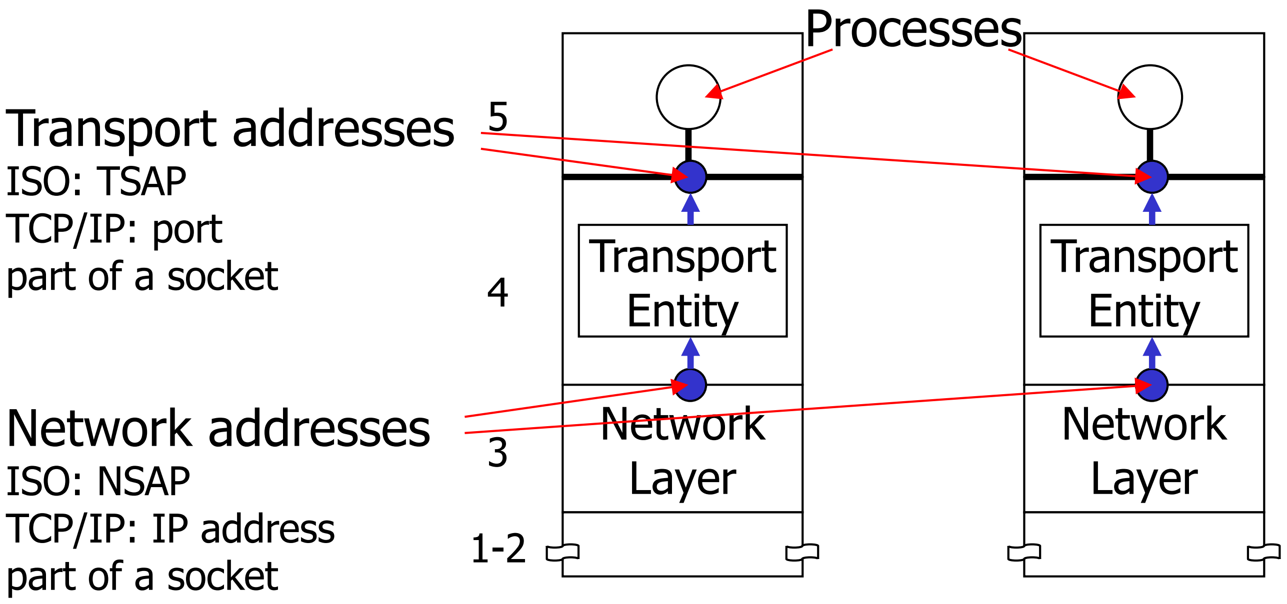
Addressing at the Transport Layer



- Applications …
  - … require communication
  - … communicate
    - locally by interprocess communication
    - between systems via transport services
- Transport layer
  - Interprocess communication via communication networks
- Network layer (including Internet Protocol, IP)
  - Enables endsystem-to-endsystem communication
  - Not application to application

Addressing at the Transport Layer

- Transport address different from network address
  - Sender process must address receiver process
  - Receiver process can be approached by the sender process



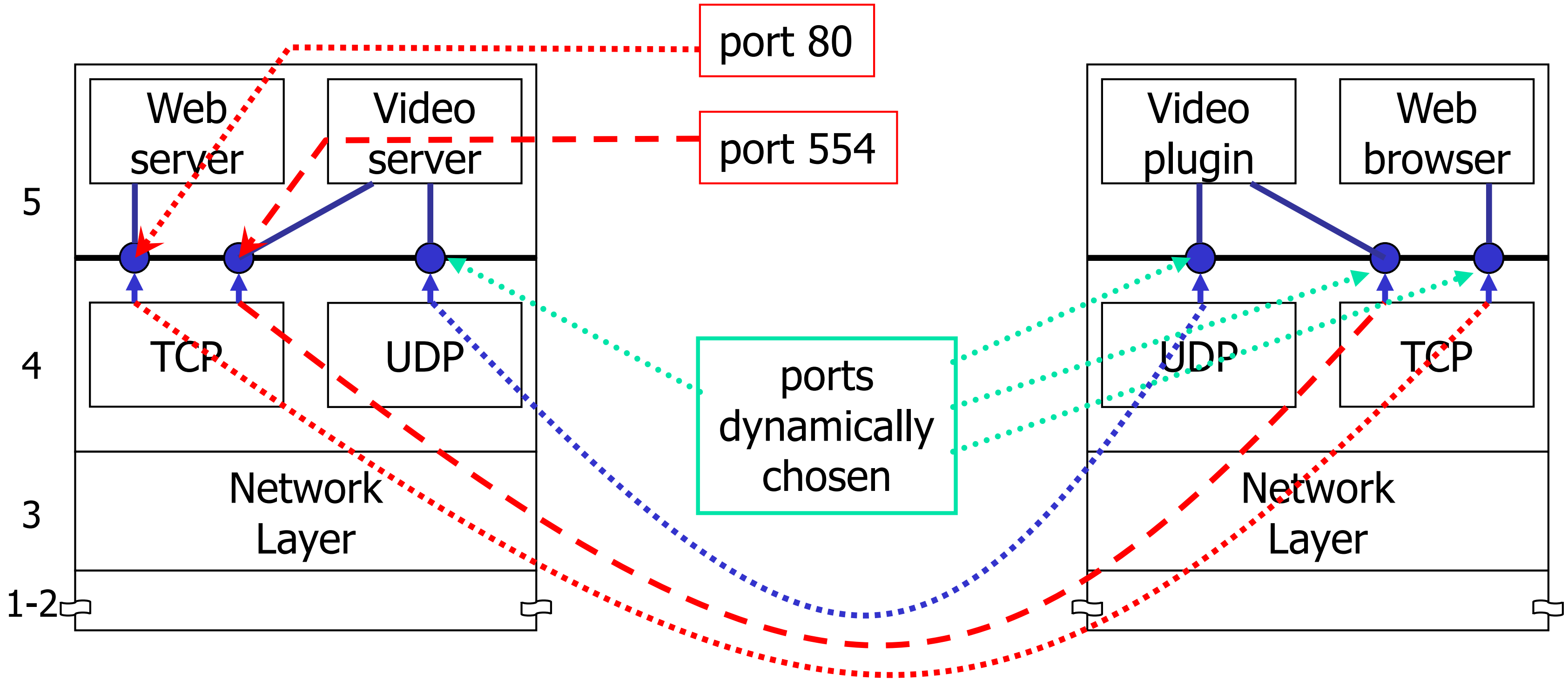
Addressing Services at the Transport Layer

Port Number	Service	Service name
22	SSH	Secure Shell
25	SMTP	Simple Mail Transfer Protocol
53	DOMAIN	Domain Name Systems
80	HTTP	Hypertext Transport Protocol
110	POP3	Post Office Protocol, version 3
123	NTP	Network Time Protocol
143	IMAP	Internet Message Access Protocol

- TCP and UDP have different assignments
  - The table shows some examples for TCP (read /etc/services for more)

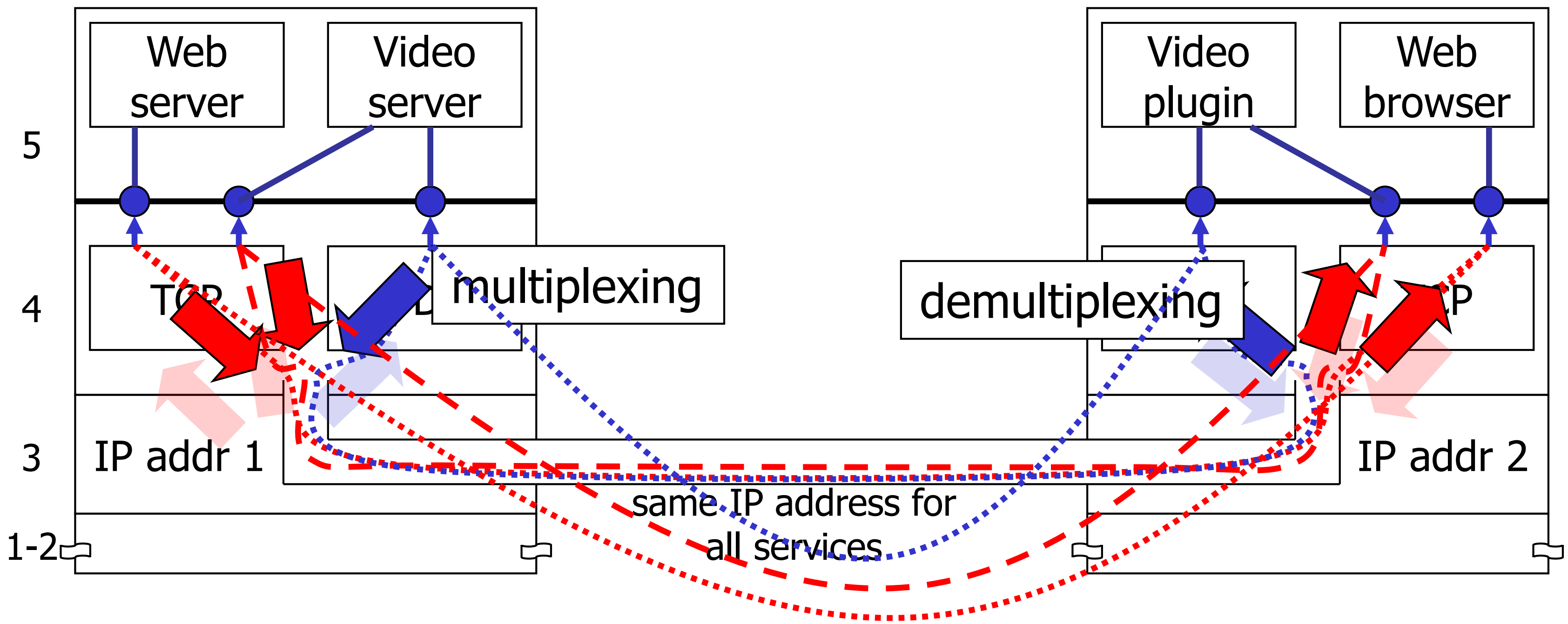
Multiplexing task of the Transport Layer

- Multiplexing and demultiplexing task of the transport layer
- Example: accessing a web page with video element
  - Three protocols used (minor simplification)
    - HTTP for web page
    - RTSP for video control
    - RTP for video data



Multiplexing task of the Transport Layer

- Multiplexing and demultiplexing task of the transport layer
- Example: accessing a web page with video element
  - Three protocols used (minor simplification)
    - HTTP for web page
    - RTSP for video control
    - RTP for video data



Transport Protocols

UDP

UDP - User Datagram Protocol

- History: IEN 88 (1979), RFC 768 (1980), STD 6

- UDP is a simple transport protocol
  - Unreliable
  - Connectionless
  - Message-oriented

- UDP is mostly IP with short transport header
  - De-/multiplexing
  - Source and destination port
  - Ports allow for dispatching of messages to receiver process

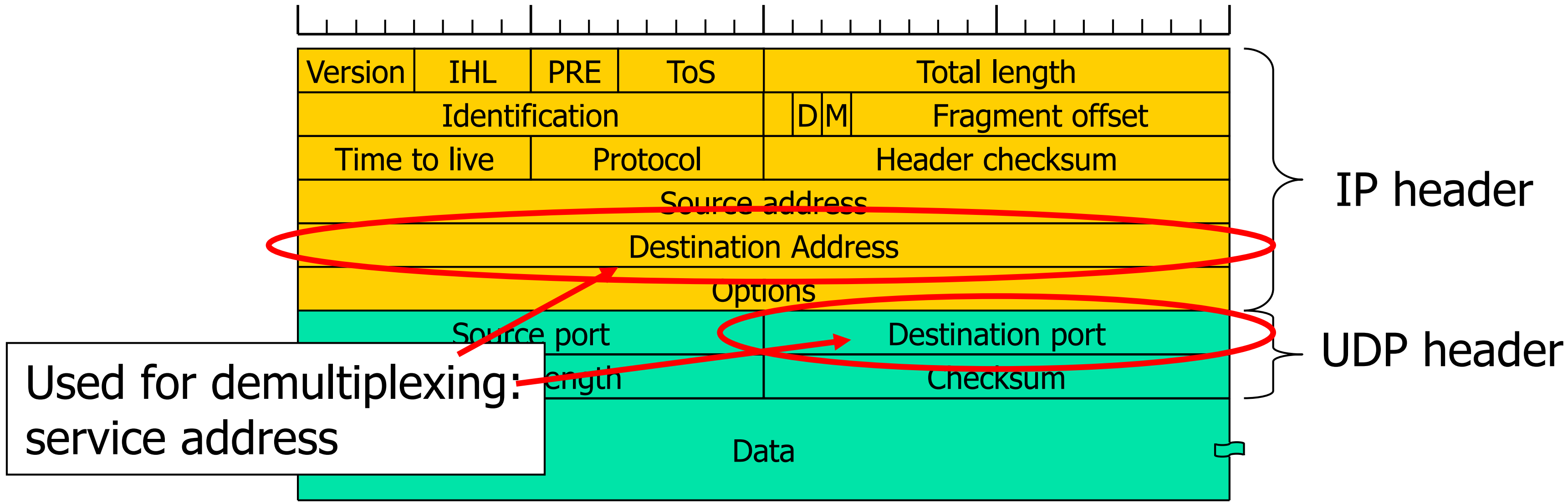
UDP Characteristics

- No flow control
  - Application may transmit as fast as it can / wants and its network card permits
  - Does not care about the network's capacity

- No error control or retransmission
  - No guarantee about packet sequencing
  - Packet delivery to receiver not ensured
  - Possibility of duplicated packets

- May be used with broadcast / multicasting and streaming

UDP: Message Format



- Source port
  - **Optional**
  - 16 bit sender identification
  - Response may be sent there
- Destination port
  - Receiver identification
- Packet length
  - In byte (including UDP header)
  - Minimum: 8 (byte)
    i.e. header without data
- Checksum
  - **Optional** in IPv4
  - Checksum of header and data for error detection

UDP: Message Format – Checksum

- Purpose
  - Error detection (header and data)

- UDP checksum includes
  1. UDP header (checksum field initially set to 0)
  2. Data
  3. Pseudoheader
     - Part of IP header
       - source IP address
       - destination IP address
       - Protocol
       - length of (UDP) data
     - Allows to detect misdelivered UDP messages

Source address		
Destination Address		
00000000	Protocol=17	UDP segment length

- Use of checksum optional
  - i.e., if checksum contains only "0"s, it is not used
    - Transmit 0xFFFF if calculated checksum is 0

UDP: Ranges of Application

- Suitable
  - For simple client-server interactions, i.e. typically
    - 1 request packet from client to server
    - 1 response packet from server to client
  - When delay is worse than packet loss and duplication
    - Video conferencing
    - IP telephony
    - Gaming

- Used by e.g.
  - DNS: Domain Name Service [1]
  - NTP: Network Time Protocol [1]
  - SNMP: Simple Network Management Protocol
  - BOOTP: Bootstrap protocol
  - TFTP: Trivial File Transfer Protocol
  - NFS: Network File System [1]
  - RTP: Real-time Transport Protocol [1]

[1] can also be used with TCP

UDP and UDP Lite

UDP header

Source port	Destination port
Packet length	Checksum

- UDP checksum:
  - If null: packets are not dropped even if address and port are wrong
  - If not null: packets are dropped even if application can tolerate errors in dat

UDP Lite header

Source port	Destination port
Checksum coverage	Checksum

- UDP Lite (length := checksum coverage)
  - advantage: e.g. video codecs can cope with bit errors, but UDP drops whole packet
  - critical: app‘s depending on UDP Lite can depend on lower layers
  - usefulness: often, link layers do not hand over corrupt data
- UDP Lite is a separate transport protocol
- OK to drop packet length: IP has a "total length" field

Transport Protocols

TCP

TCP - Transmission Control Protocol

- TCP is the main transport protocol of the Internet

- History: IEN 112 (1979), RFC 793 (1981), STD 7

- Motivation: network with connectionless service
  - Packets and messages may be
    - duplicated, in wrong order, faulty
    - i.e., with such service only, each application would have to provide recovery
      - error detection and correction
  - Network or service can
    - impose packet length
    - define additional requirements to optimize data transmission
    - i.e., application would have to be adapted

- TCP provides
  - Reliable end-to-end byte stream over an unreliable network service

What is TCP?

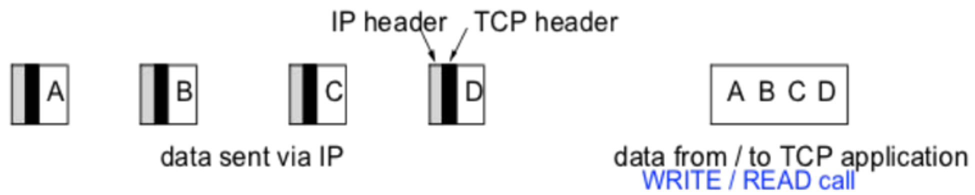
Transport protocol specification	~~Transport protocol implementation~~

- TCP specifies
  - Data and control information formats
  - Procedures for
    - flow control
    - error detection and correction
    - connect and disconnect
  - As a primary abstraction
    - a connection
    - not just the relationships of ports (as a queue, like UDP)

- TCP does not specify
  - The interface to the application (sockets, streams)
  - Interfaces are specified separately: e.g. Berkeley Socket API, WINSOCK

TCP Characteristics

- Data stream oriented
  - TCP transfers serial byte stream
  - Maintains sequential order
- Unstructured byte stream
  - Application often has to transmit more structured data
  - TCP does not support such groupings into (higher) structures within byte stream
- Buffered data transmission
  - Byte stream not message stream: message boundaries are not preserved
    - no way for receiver to detect the unit(s) in which data were written

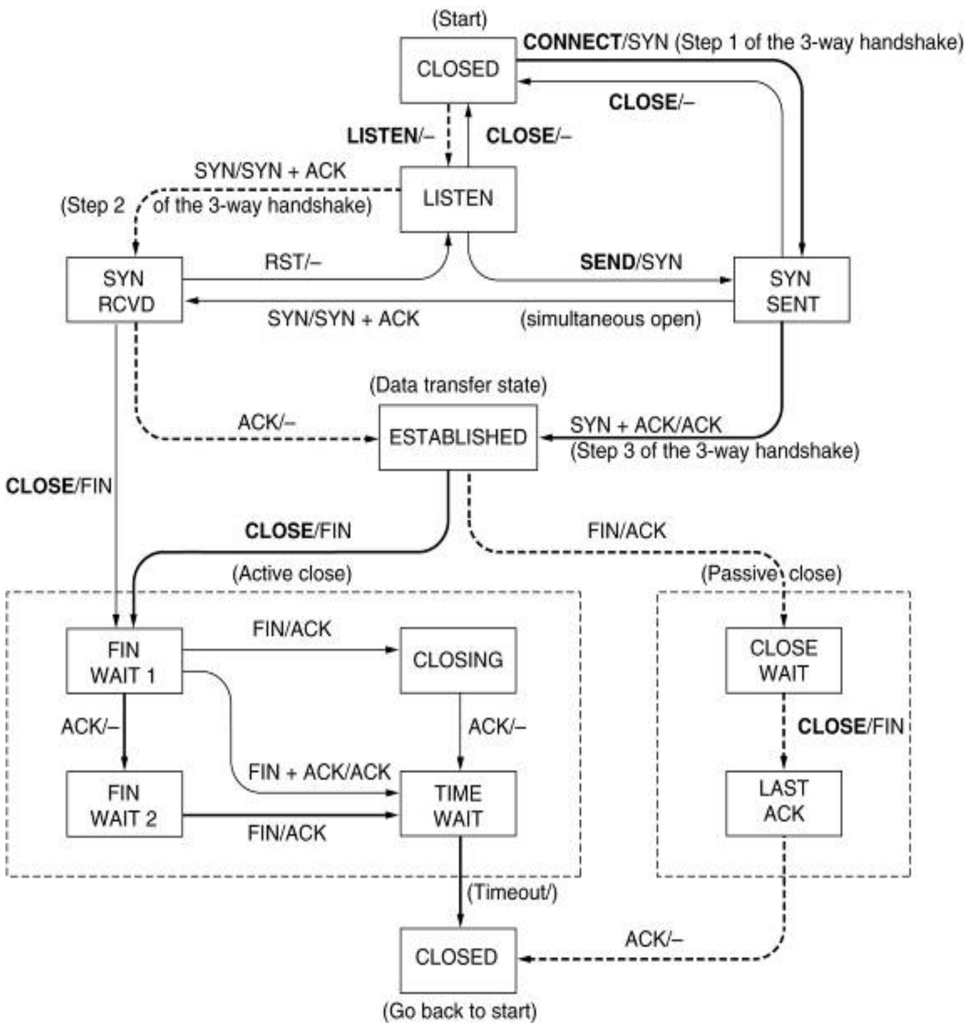


  - For transmission the sequential data stream is
    - Divided into segments
    - Delayed if necessary (to collect data)

TCP Characteristics

- Virtual connection
  - Connection established between communication parties before data transmission
- Two-way communications (fully duplex)
  - Data may be transmitted simultaneously in both directions over a TCP connection
- Point-to-point
  - Each connection has exactly two endpoints

TCP Connection Management



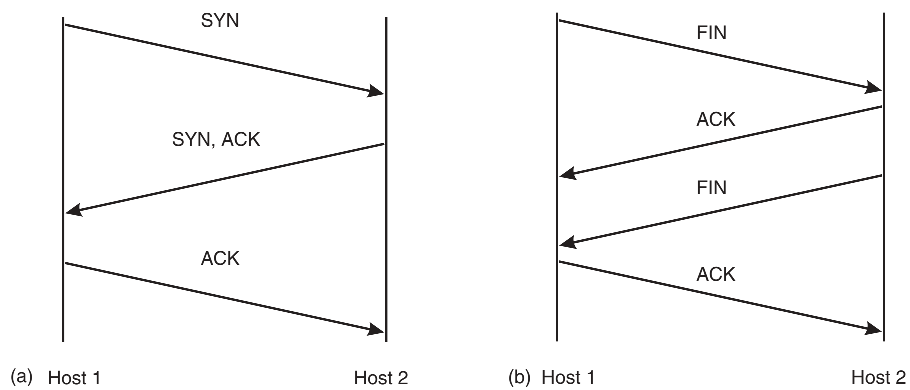
heavy solid line:
normal path for a client

heavy dashed line:
normal path for a server

Light lines:
unusual events

Connection setup

teardown



Some servers go from ESTABLISHED to LISTEN via RST/-
e.g. Apache etc.

TCP Characteristics

- Virtual connection
  - Connection established between communication parties before data transmission
- Two-way communications (fully duplex)
  - Data may be transmitted simultaneously in both directions over a TCP connection
- Point-to-point
  - Each connection has exactly two endpoints
- Reliable
  - Fully ordered, fully reliable
    - Sequence maintained
    - No data loss, no duplicates, no modified data

TCP Characteristics

- Error detection
  - Through checksum

- Piggybacking
  - Control information and data can be transmitted within the same segment

- Urgent flag
  - Send and transfer data to application immediately
    - example <Crtl C>
      arrival interrupts receiver's application
  - Deliver to receiver's application before data that was sent earlier

TCP Characteristics

- No broadcast
  - No possibility to address all applications
  - With connect, however, not necessarily sensible
- No multicasting
  - Group addressing not possible
- No QoS parameters
  - Not suited for different media characteristics
- No real-time support
  - No correct treatment / communications of audio or video possible
  - E.g. no forward error correction

TCP in Use & Application Areas

Benefits of TCP

- Reliable data transmission
  - Efficient data transmission despite complexity
  - Can be used with LAN and WAN for
    - low data rates (e.g. interactive terminal) and
    - high data rates (e.g. file transfer)

Disadvantages when compared with UDP

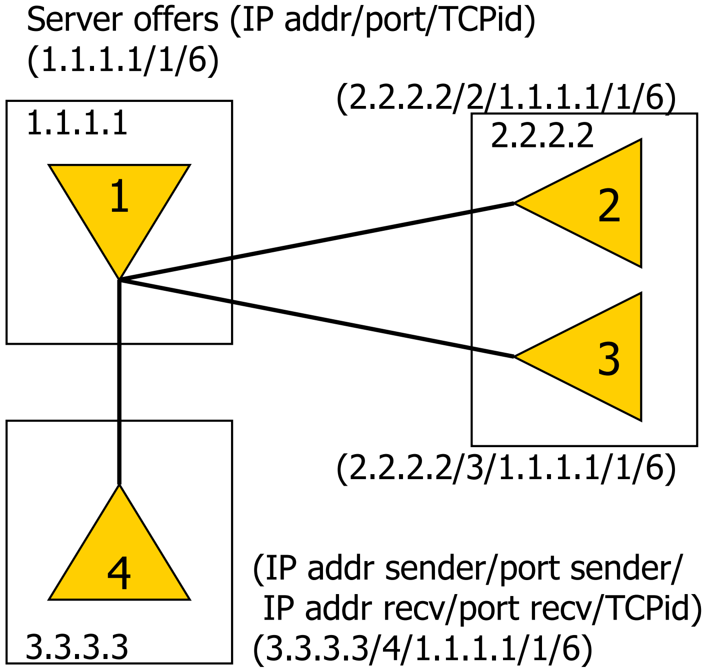
- Higher resource requirements
  - buffering, status information, timer usage
- Connection set-up and disconnect necessary
  - even with short data transmissions

Applications

- File transfer (FTP)
- Interactive terminal (Telnet)
- E-mail (SMTP)
- X-Windows

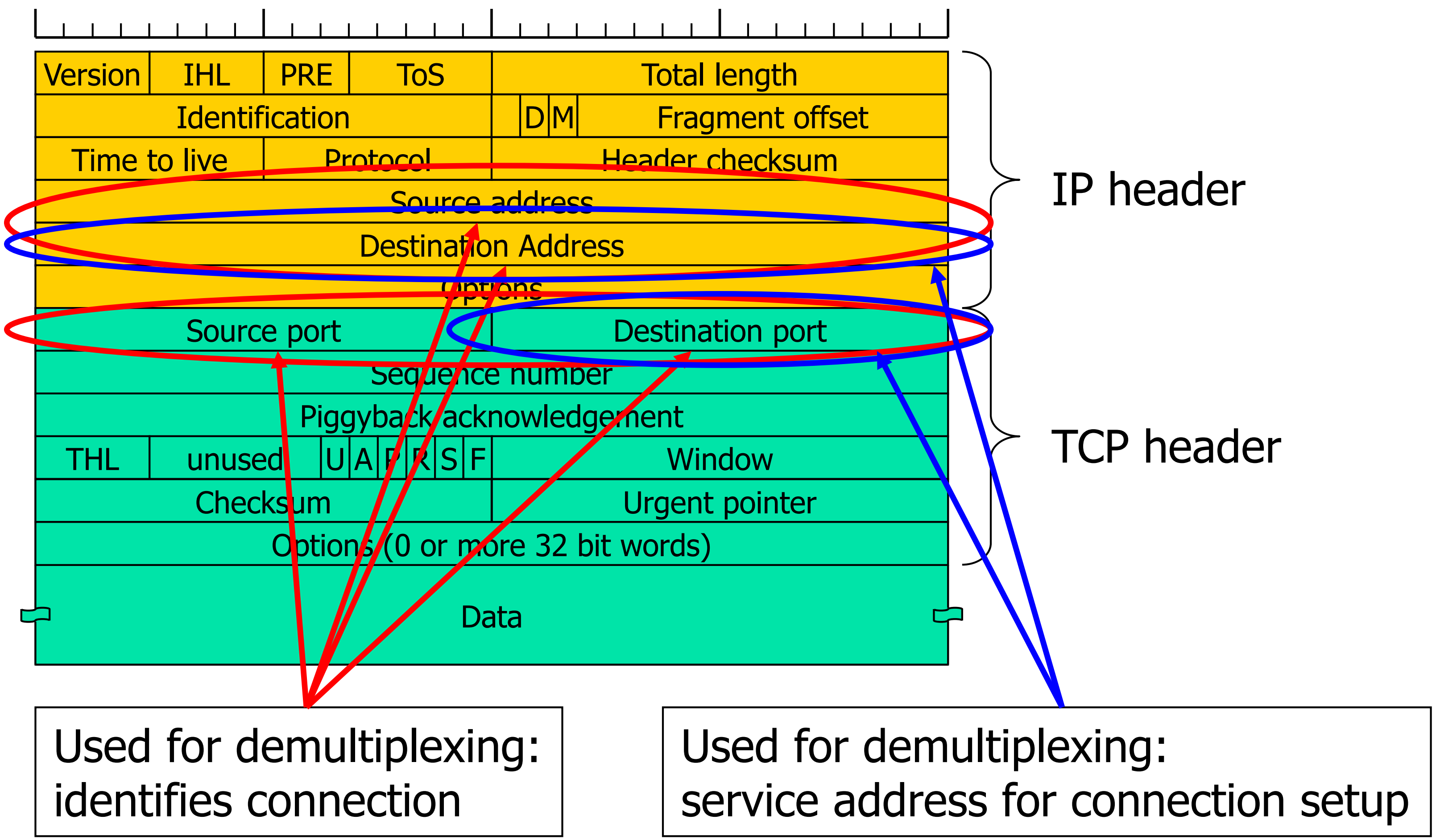
Connection – Addressing

- TCP *service* obtained via service endpoints on sender and receiver
  - Typically socket
  - Socket number consists of a ***3-tuple***
    - IP address of host and
    - 16-bit local number (port)
    - TCP protocol identifier
- Transport Service Access Point
  - Port
- TCP *connection* is clearly defined by a ***5-tuple*** consisting of
  - IP address of sender and receiver
  - Port address of sender and receiver
  - TCP protocol identifier
- Applications can use the same local ports for several connections
  - But only when the remote side is different



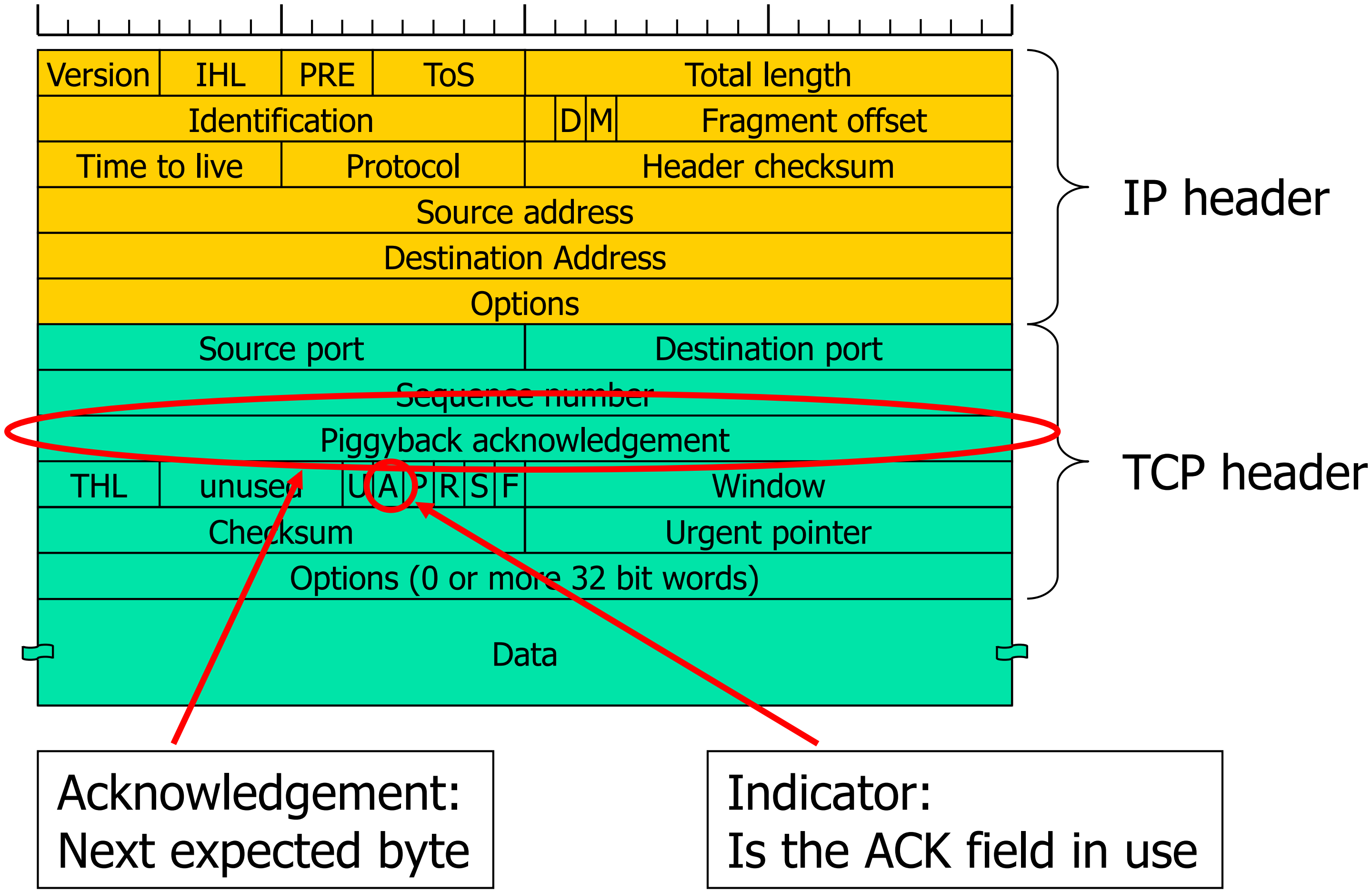
TCP: Message Format

- TCP/IP Header Format



TCP Reliability

- TCP/IP Retransmits segments until they are acknowledged
- For that is used the ACK field



Why count bytes in TCP?

- IP fragmentation exists

- IP fragments vs TCP segments
  - Small packets have large header overhead

- Path MTU Discovery: determine the largest packet that does not get fragmented
  - originally (RFC 1191, 1990): start large, reduce upon reception of ICMP message ➔ black hole problem if ICMP messages are filtered
  - now (RFC 4821, 2007): start small, increase as long as transport layer ACKs arrive ➔ transport protocol dependent

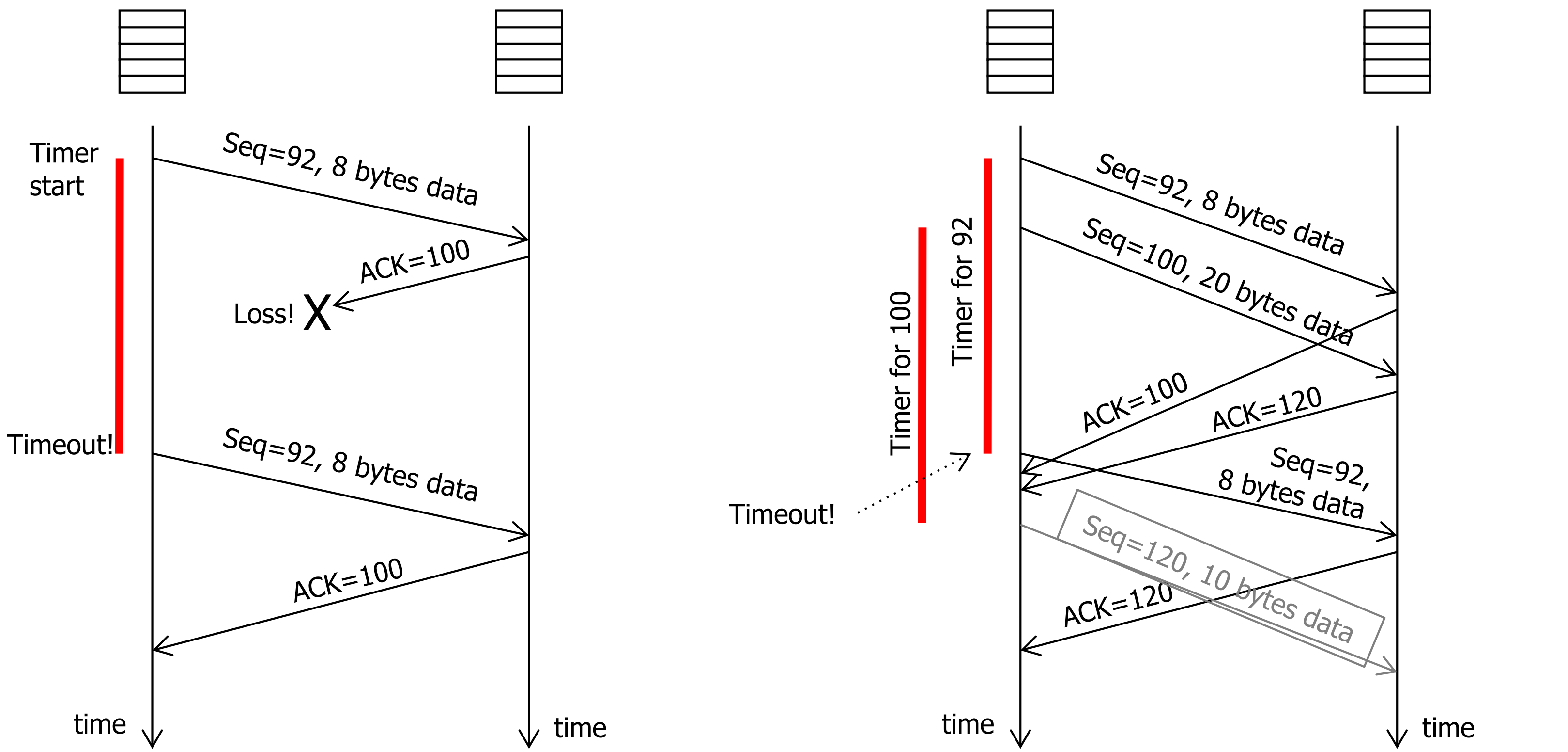
- Network layer function with transport layer dependencies

Error control: Acknowledgements

ACK generation according to RFC 1122 and RFC 2581

Event	TCP Receiver Action
Segments arrive in order No missing segments Previous segment was ACK'd	Use Delayed ACK. Wait for up to 500ms before sending ACK. Don't send if following segment arrives during waiting time.
Segments arrive in order No missing segments Previous segment wasn't ACK'd yet	Send one (cumulative) ACK immediately.
Segment arrives out of order. Segment number is higher than expected.	Send an ACK for the last in-order segment immediately. This ACK contains the sequence number of the first byte of the missing part.
Segment arrives that fills a gap or part of a gap.	Send an ACK immediately. The ACK contains the sequence number of the first byte that is now missing.

TCP ACK and Reliability



Timeout retransmission **Spurious retransmission**

In modern TCP implementations,
a lot is done to suppress spurious retransmissions.
The reason is that timeouts have another meaning: congestion

TCP Round trip time and timeout

- How to set the retransmission timer?
  - Must be longer than round-trip time (RTT)
  - Too short: many spurious retransmissions
  - Too long: slow reaction, flow control buffer fills up
- Idea
  - Estimated RTT
    - Use a SampleRTT: Measure time from segment sending and ACK received
    - Compute EstimatedRTT as a floating average
    - Increase the EstimatedRTT with a safety margin that is big when SampleRTT changes a lot
      - (because fewer timeouts are considered better)

$$x = 0.1$$

$$Deviation = (1 - x) * Deviation + x * |SampleRTT - EstimatedRTT|$$

$$EstimatedRTT = (1 - x) * EstimatedRTT + x * SampleRTT$$

$$Timeout = 3 * EstimatedRTT + 4 * Deviation$$

Not only 1 RTT because we have also "Fast Retransmission" for fixing packet loss, explained with "Congestion"

RTO calculation

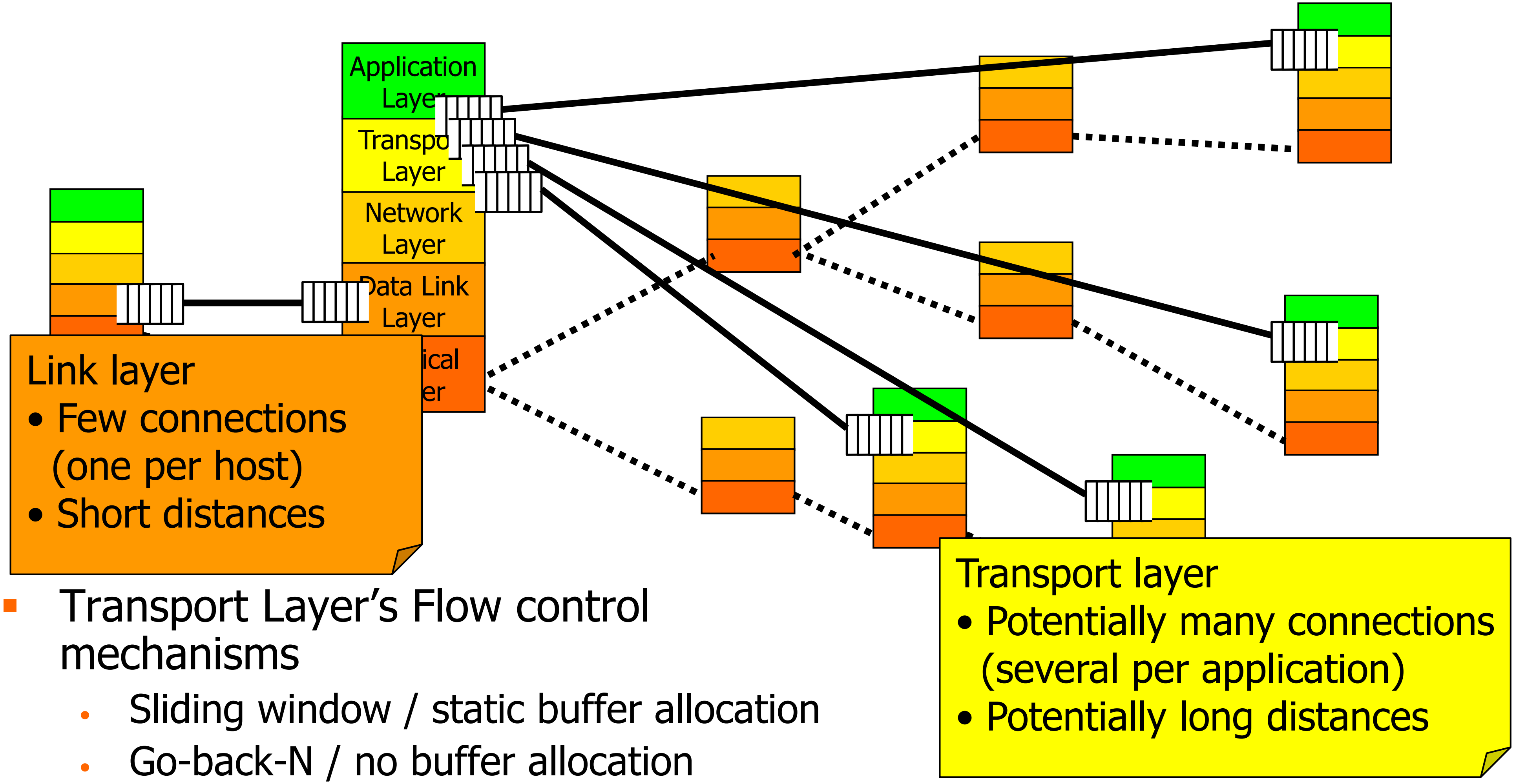
- Problem: retransmission ambiguity
  - Segment #1 sent, no ACK received → segment #1 retransmitted
  - Incoming ACK #2: cannot distinguish whether original or retransmitted segment #1 was ACKed
  - Thus, cannot reliably calculate RTO!
- Solution 1 [Karn/Partridge]: ignore RTT values from retransmits
  - Problem: RTT calculation especially important when loss occurs; sampling theorem suggests that RTT samples should be taken more often
- Solution 2: Timestamps option
  - Sender writes current time into packet header (option)
  - Receiver sends this value back
  - At sender, when ACK arrives, RTT = (current time) - (value carried in option)
  - Problems: additional header space; facilitates NAT detection

Transport layer

Flow Control

Flow Control on Transport Layer

Fast sender shall not flood slow receiver



- Transport Layer's Flow control mechanisms
  - Sliding window / static buffer allocation
  - Go-back-N / no buffer allocation
  - Credit mechanism / dynamic buffer allocation

Sliding Window / Static Buffer Allocation

- Flow control
  - Sliding window - mechanism with window size *w*

- Buffer reservation
  - Receiver reserves *2*w* sequence numbers per duplex connection

- Characteristics
  - Receiver can accept all packets
  - Buffer requirement proportional to number of connections
  - Poor buffer utilization for low throughput connections

- i.e.

  ⇒ Good for traffic with high throughput
    - (e.g. data transfer)

  ⇒ Poor for bursty traffic with low throughput
    - (e.g. interactive applications)

Stop-and-wait?
Really bad idea due to round-trip-times

Go-back-N / No Buffer Allocation

- Flow control
  - Go-back-N (or no flow control)
- Buffer reservation
  - Receivers do not reserve buffers
  - Buffers allocated out of shared buffer space upon arrival of packets
  - Packets are discarded if there are no buffers available
  - Sender maintains packet buffer until ACK arrives from receiver
- Characteristics
  - Optimized storage utilization
  - Possibly high rate of ignored packets during high traffic load
- i.e.

  ⇒ Good for bursty traffic with low throughput

  ⇒ Poor for traffic with high throughput

Credit Mechanism

- Flow control
  - Credit mechanism

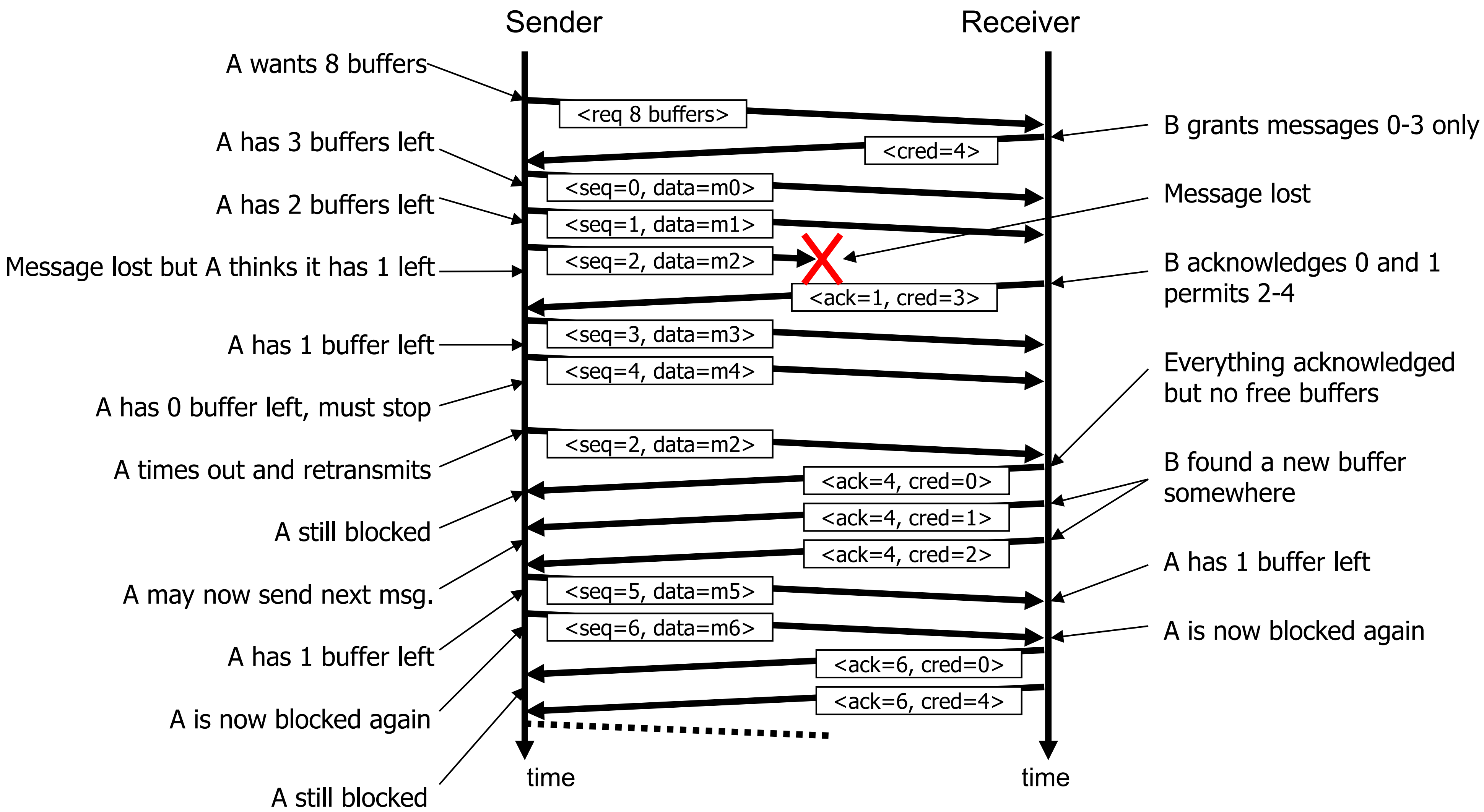
- Buffer reservation
  - Receiver allocates buffers dynamically for the connections
  - Allocation depends on the actual situation

- Principle
  - Sender requests required buffer amount
  - Receiver reserves as many buffers as the actual situation permits
  - Receiver returns ACKs and buffer-credits separately
    - ACK: confirmation only (does not imply buffer release)
    - CREDIT: buffer allocation
  - Sender is blocked when all credits are used up

Credit Mechanism

- Dynamic adjustment to
  - Buffer situation
  - Number of open connections
  - Type of connections
    - high throughput: many buffers
    - low throughput: few buffers

Credit Mechanism

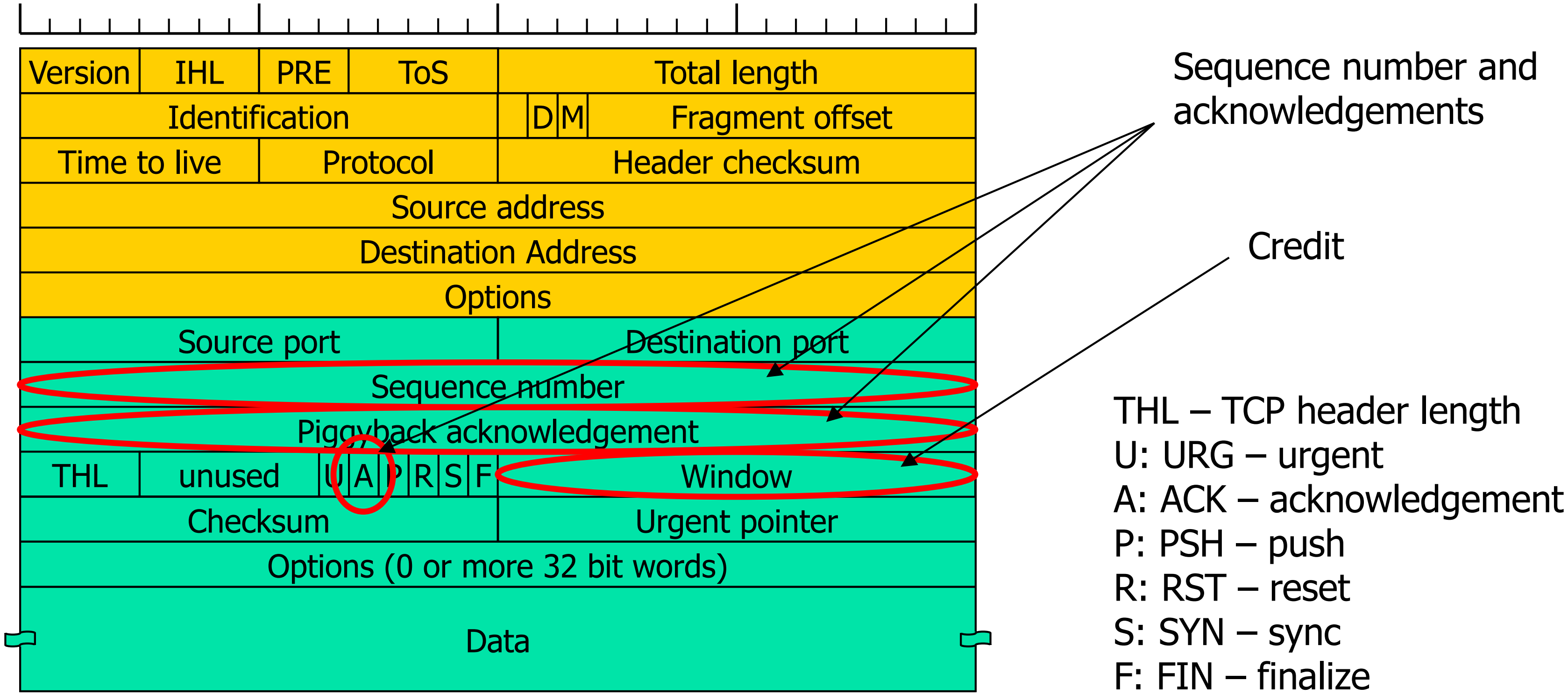


Transport layer

Flow Control: TCP

TCP Flow Control

- Variable window sizes (credit mechanism)
  - Implementation
    - The actual window size is also transmitted with each acknowledgement
    - Permits dynamic adjustment to existing buffer



TCP Flow Control

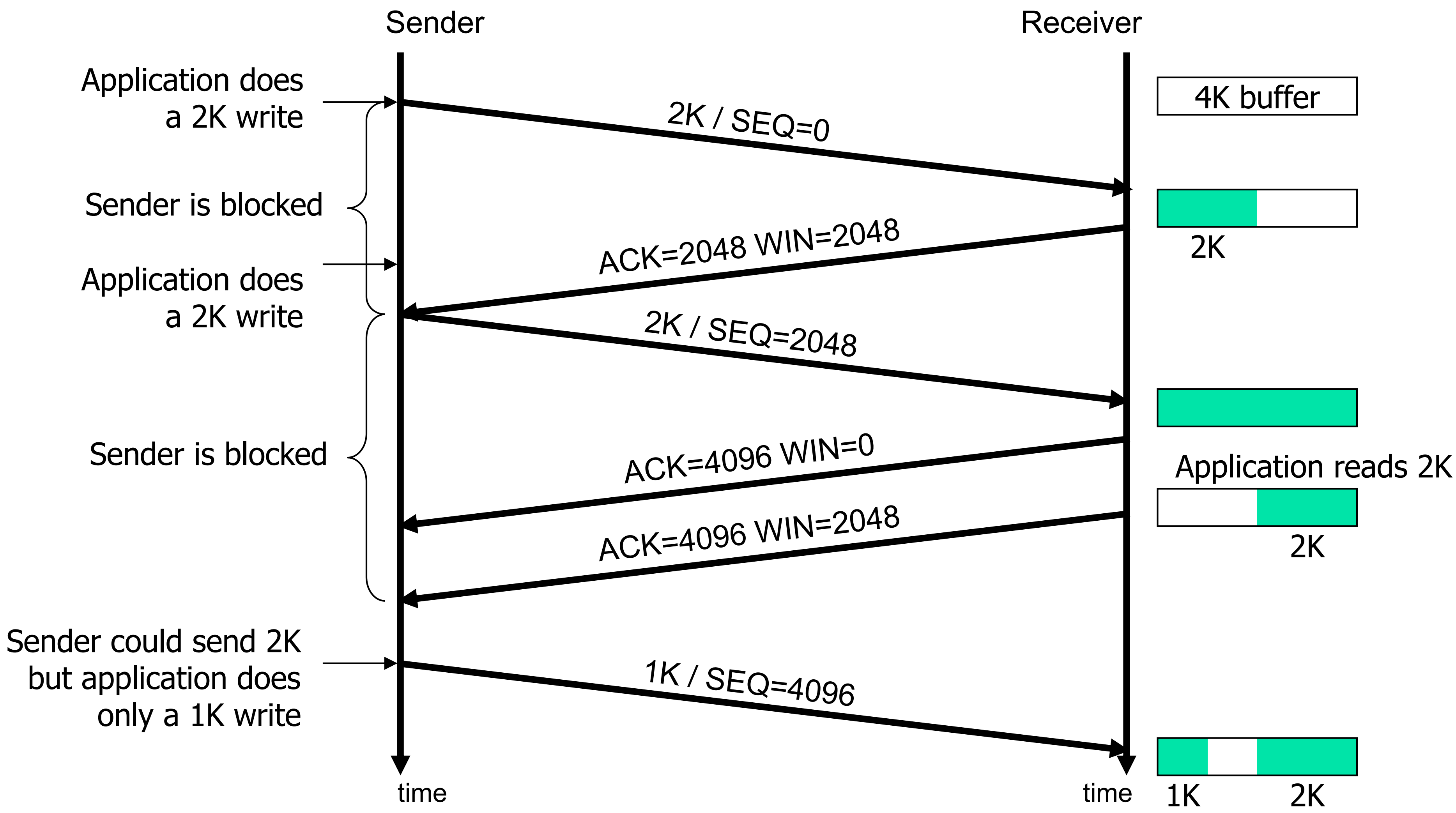
- "Sliding window" mechanism
  - Sequence number space is limited
  - No buffers longer than 2^{32} possible
  - No credit larger than 2^{16} possible

but 2^{16} of what?

  - Acknowledgement and sequence number
    - Acknowledgments refer to *byte* positions
    - Sequence numbers refer to the *byte* position of a TCP connection
  - Positive acknowledgement
    - *Cumulative acknowledgements*
      - Byte position in the data stream up to which all data has been received correctly
      - Reduces overhead and adds tolerance to ACK loss

  - 2^{16} bytes limit the max. throughput in long fat pipes
    - Window Scale Option - RFC 1323
    - Changes only meaning of `window` (the 16-bit credit field)
    - Count in 2^N bytes, with $0 \leq N \leq 14$

TCP Flow Control

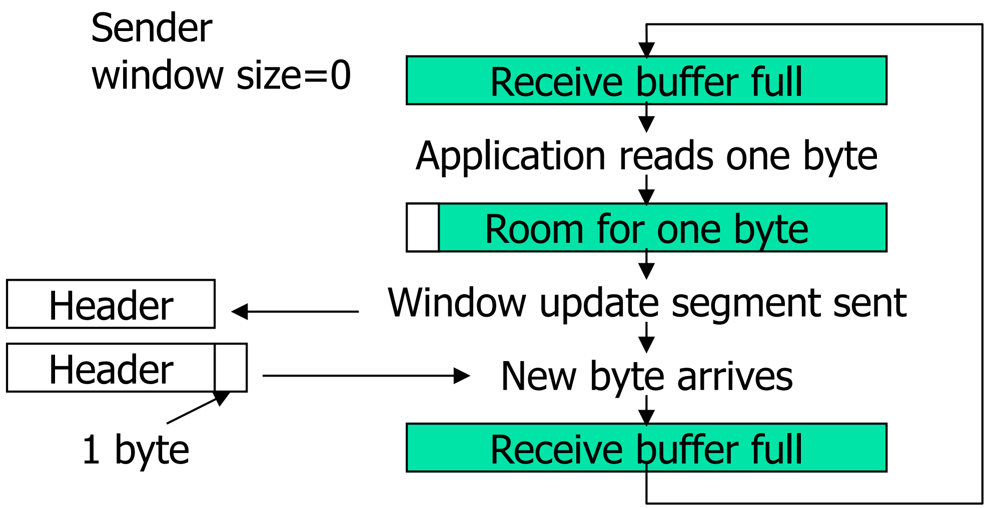


TCP Flow Control: Special Cases

- Optimization for low throughput rate
  - Problem
    - Telnet (and ssh) connection to interactive editor reacting on every keystroke
    - 1 character typed requires up to 162 byte
      - Data: 20 bytes TCP header, 20 bytes IP header, 1 byte payload
      - ACK: 20 bytes TCP header, 20 bytes IP header
      - Echo: 20 bytes TCP header, 20 bytes IP header, 1 byte payload
      - ACK: 20 bytes TCP header, 20 bytes IP header
  - Approach often used
    - Delay acknowledgment and window update by 500 ms (hoping for more data)

- Nagle's algorithm, 1984
  - Algorithm
    - Send first byte immediately
    - Keep on buffering bytes until first byte has been acknowledged
    - Then send all buffered characters in one TCP segment and start buffering again
  - Comment
    - Effect at e.g. X-windows: jerky pointer movements
    - Nagle can be good and bad - choose depending on your needs

TCP Flow Control: Special Cases



- Silly window syndrome (Clark, 1982)
  - Problem
    - Data on sending side arrives in large blocks
    - But receiving side reads data one byte at a time
  - Clark's solution
    - Prevent receiver from sending window update (new credit) for 1 byte
    - Certain amount of space must be available in order to send window update
  - min(X,Y)
    X = maximum segment size announced during connection establishment
    Y = buffer / 2