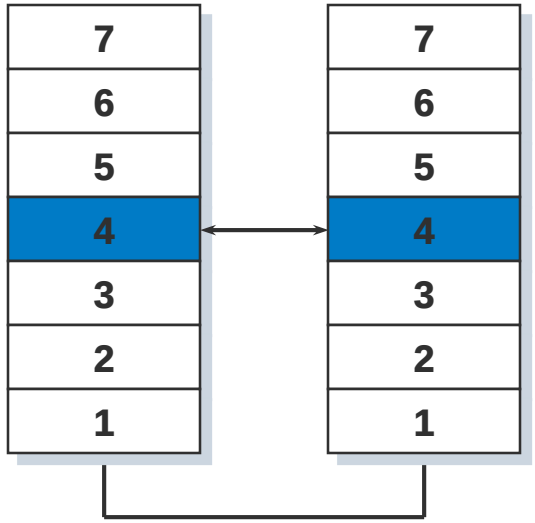
Freie Universität Berlin

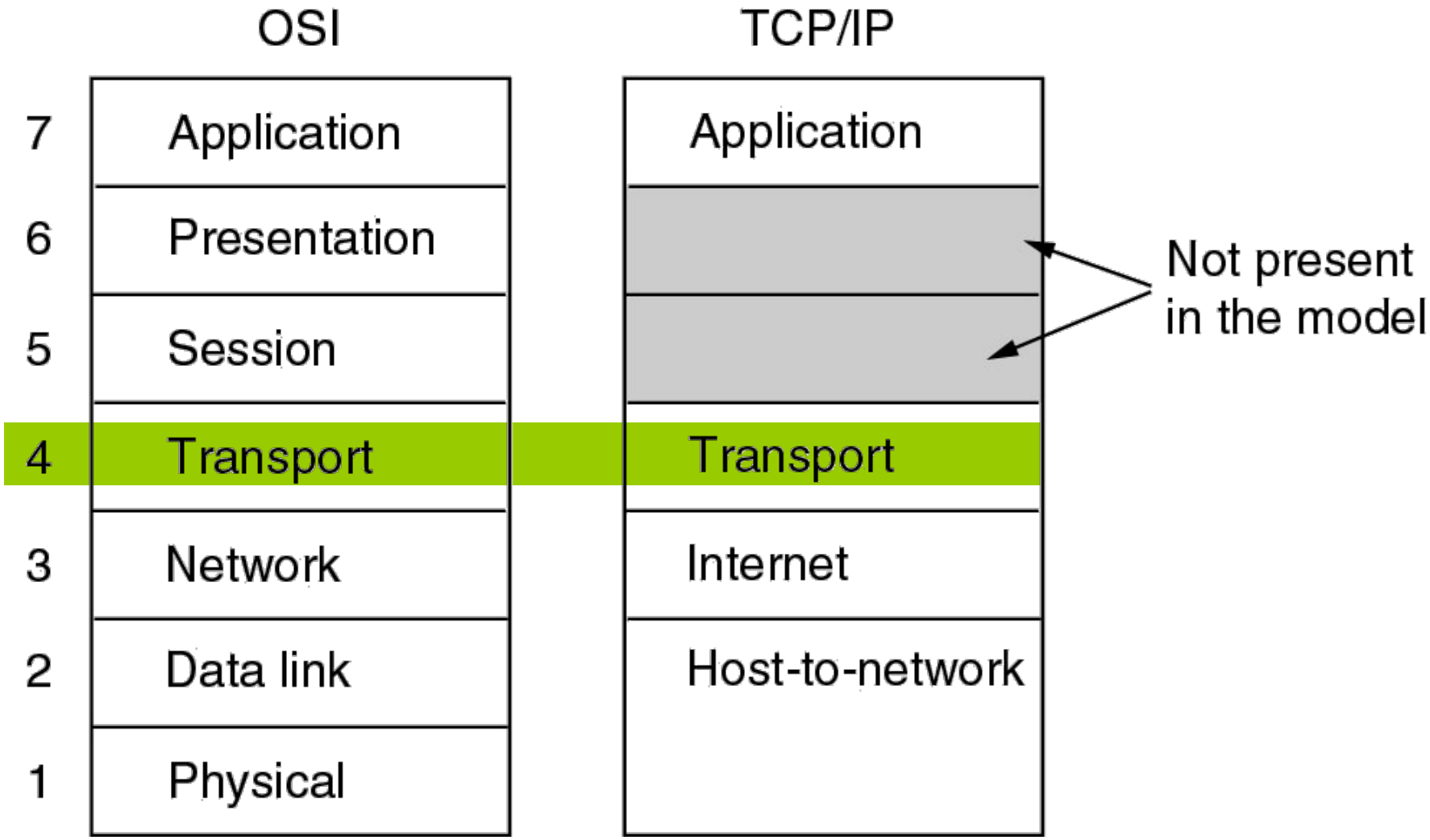
TI III: Operating Systems & Computer Networks
Transport Layer

Prof. Dr.-Ing. Jochen Schiller
Computer Systems & Telematics
Freie Universität Berlin, Germany



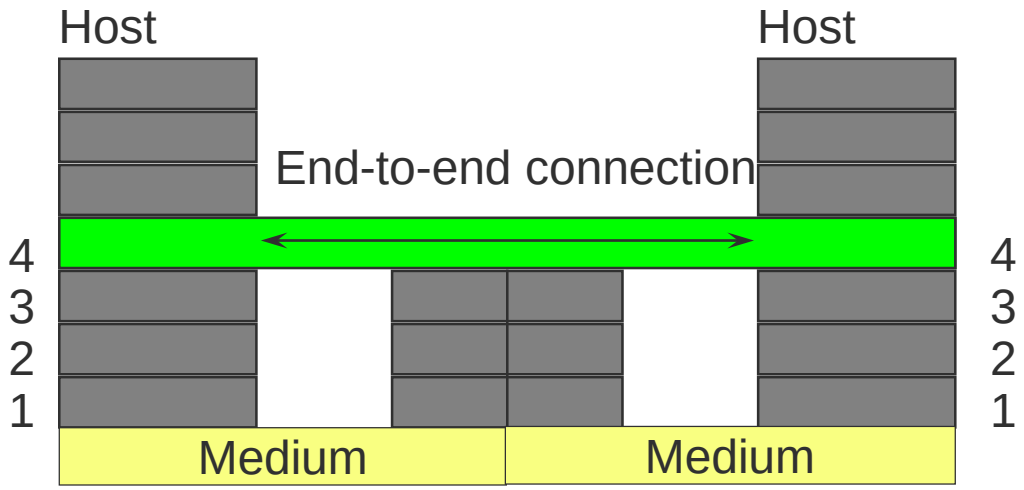
Content

Transport Layer



Tasks of Transport Layer

End-to-end connection: From process to process (not node-to-node)



Insulation of higher layers from technology, structure and impairments of lower layers, e.g., packet loss
Transparent transmission of user data
Support of Quality of Service (QoS)
- Not widely deployed in the classical Internet

Independent addressing of processes, i.e., independent of Layer 3
- Exception: The Internet Socket (IP-address + port)

Services of Transport Protocols

Services provided to upper layers:

- Connection-less and connection-oriented transport service
  - Connection management (setup and teardown) necessary as auxiliary service

- Reliable or unreliable transport
  - In-order delivery
  - Reliability, i.e., all packets

- Congestion control – be a good citizen in the network

- Demultiplexing, i.e., support of several transport endpoints in a single host

- Support different interaction models
  - Byte stream, messages
  - Remote Procedure Calls (RPC)

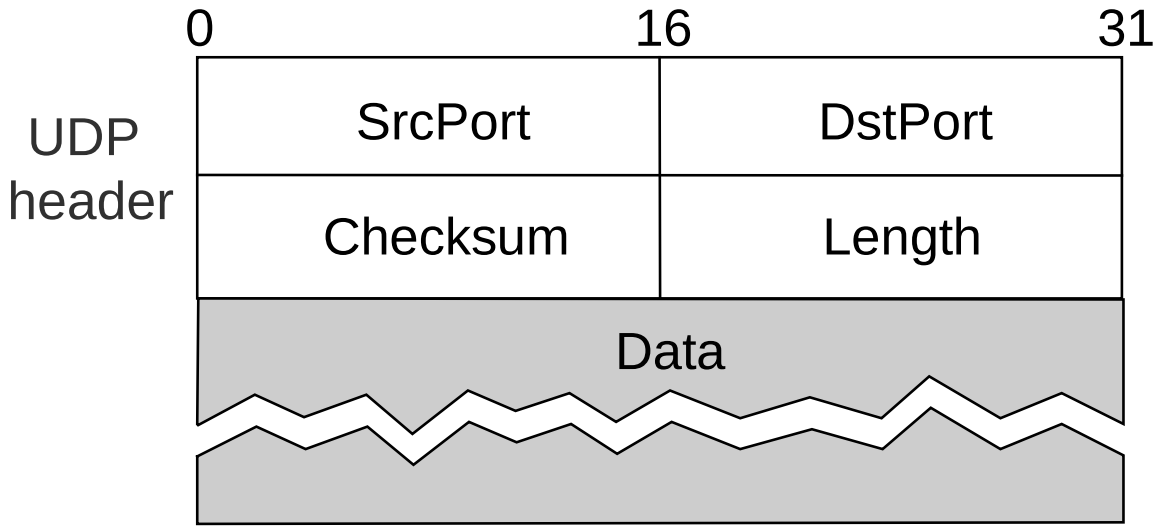
Example: User Datagram Protocol (UDP)

Unreliable, datagram protocol

(De)multiplex several data flows onto IP layer and back to applications

Ensures packet's correctness

- Checksum of pseudoheader (IP source/destination, protocol ID and length of data), UDP header and data



Addressing / Demultiplexing

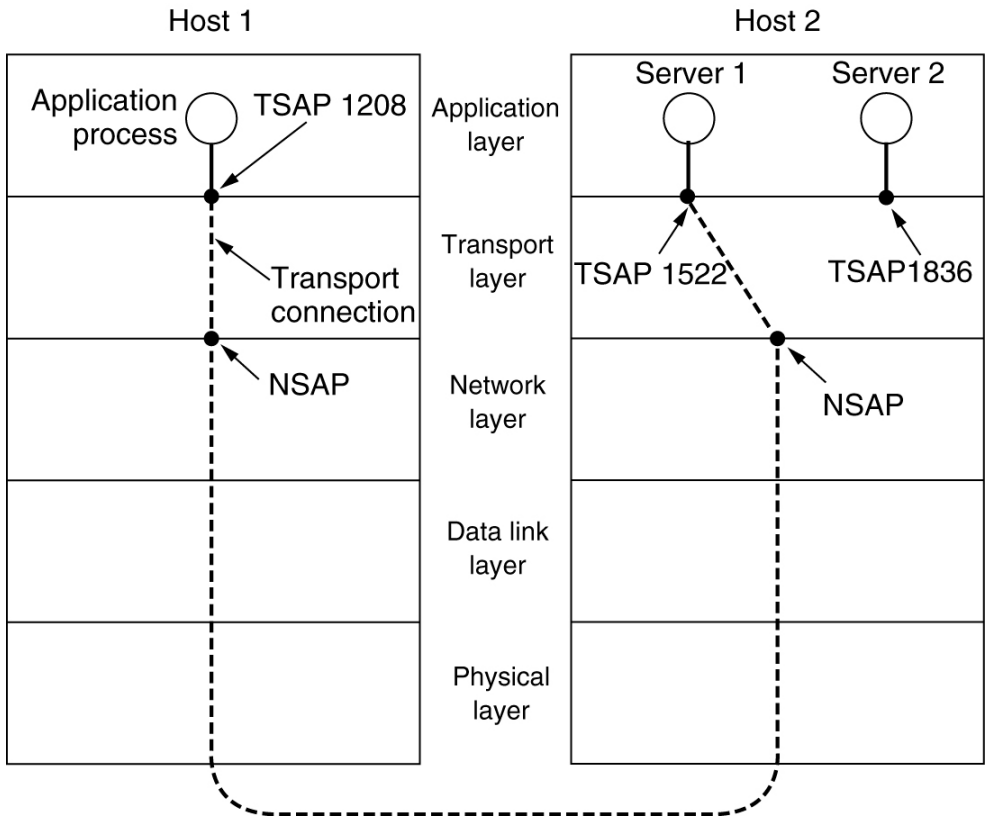
Provide multiple service access points (SAPs) to multiplex several applications

- SAPs can identify connections or data flows

Example: Port numbers as Transport SAP identifiers in TCP/UDP

- Dynamically allocated
- Predefined for “well-known” services, e.g., port 80 for HTTP/Web server
- Privileges required to bind to certain ports

TCP/UDP connection is thus identified by four tuple (known as socket pair):



[Source Port, Source IP Address, Destination Port, Dest. IP Address]

TCP – Some Well-known Ports

Many applications choose TCP/UDP as transport protocol

Correct port must be used to communicate with respective application on server side:

- 13: Day time
- 20: FTP data
- 25: SMTP
  (Simple Mail Transfer Protocol)
- 53: DNS
  (Domain Name Server)
- 80: HTTP
  (Hyper Text Transfer Protocol)
- 119: NNTP
  (Network News Transfer
  Protocol)

```
> telnet walapai 13
Trying 129.13.3.121...
Connected to walapai.
Escape character is '^]'.
Sun Jan  21 16:57:19 2007
Connection closed by foreign host
```

```
> telnet mailhost 25
Trying 129.13.3.161...
Connected to mailhost .
Escape character is '^]'.
220 mailhost ESMTP Sendmail 8.8.5/8.8.5; Sun,
21 Jan 2007 17:02:51 +0200
HELP
214-This is Sendmail version 8.8.5
214-Topics:
214-    HELO    EHLO    MAIL    RCPT    DATA
214-    RSET    NOOP    QUIT    HELP    VRFY
214-    EXPN    VERB    ETRN    DSN
214-For more info use "HELP <topic>".
...
214 End of HELP info
```

Questions & Tasks

- What does the transport layer see from the underlying technology, media, intermediate systems?
- Why is it difficult to offer QoS at the transport layer?
- How can congestions happen in the underlying network?
- Why having UDP – it is as unreliable as IP…?

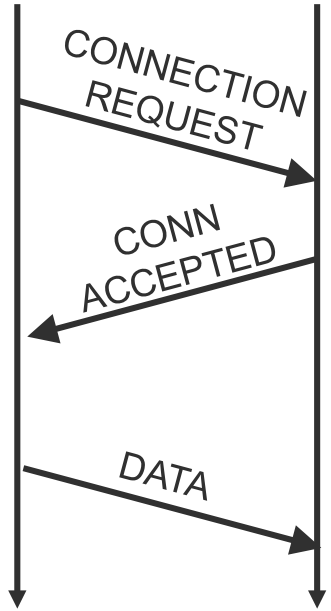
Connection Establishment

How to establish a joint context and a connection between sender and receiver?

- Only relevant in end-systems (not for routers), network layer (IP) assumed to be connection-less

Naïve solution:

- Sender sends
  - CONNECTION REQUEST
- Receiver answers with
  - CONNECTION ACCEPTED
- Sender proceeds once that
  - message is received



Failure of Naïve Solution

Naïve solution fails in realistic networks

- In which packets can be lost, stored/reordered, and duplicated

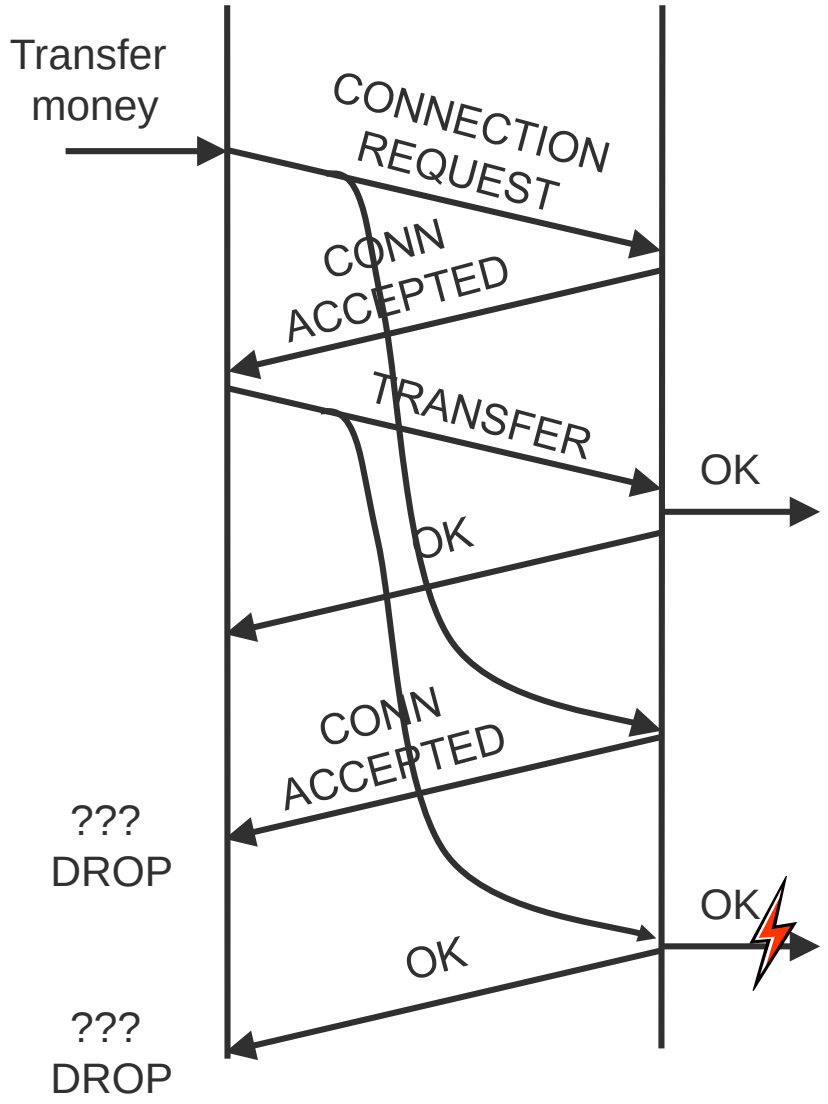
Example failure scenario: All packets are duplicated and delayed

- Due to congestion, errors, re-routing, ...

Result: Two independent transactions performed, while only one was intended

- Similar to replay attack

Problematic are delayed duplicates!



More Sophisticated Solution

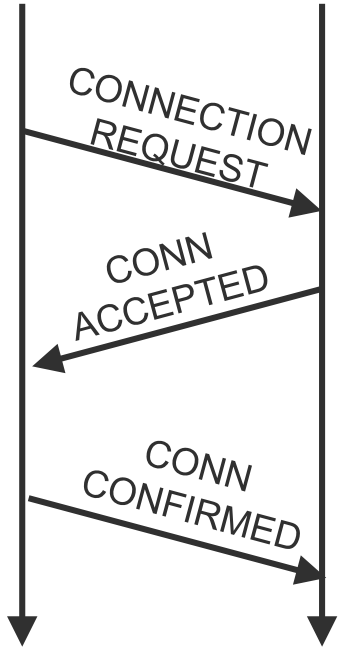
Idea: Add additional handshake

- Sender has to re-confirm to receiver that it actually wants to set up this connection

Add third message to connection setup phase:

Three-way handshake

- This third message can already carry data for efficiency (piggy backing)



Connection Setup – Further Issues

Terminology for TCP:

- SYN (synchronize) packet – connection setup
- SYN/ACK packet – Connection accepted
  - Previous sequence number is acknowledged; new sequence number from receiver is proposed
- ACK packet – Connection confirmation
  - Combined with DATA

Sequence numbers used for:

- Identification of duplicate connection setup messages
- Acknowledgement of following data packets

Crashing or malicious nodes may leave connections half open, i.e., not reply to SYN/ACK

- Tie up some resources (kernel-space memory)
- Resources need to be freed after timeout
- Possible attack: SYN-Flooding

Connection Release

Goal: Release connection when both peers have agreed that they have received all data and have nothing more to say

- Both sides must have invoked a “Close”-like service primitive

Problem:

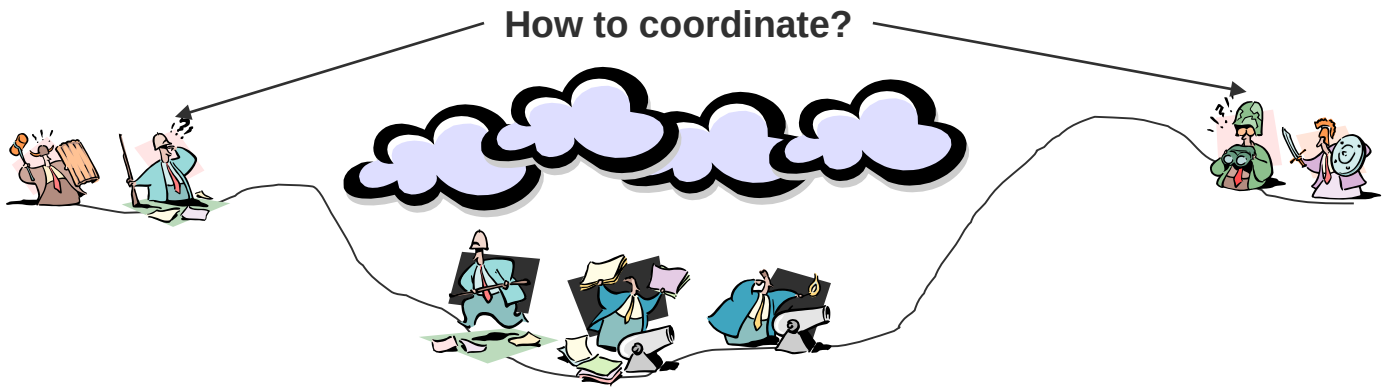
- Given that packets may be lost, how to acknowledge reliably that no further communication is required
- ACKs would require to be ACKed, which would require to be ACKed...

Analogy: Two army problem (coordinated attack)

- Two armies form up for an attack against each other
- One army is split into two parts that must attack together
- Communication via messengers who can be captured

Which rules shall commanders use to agree on attack date?

Provably unsolvable if messages can be lost

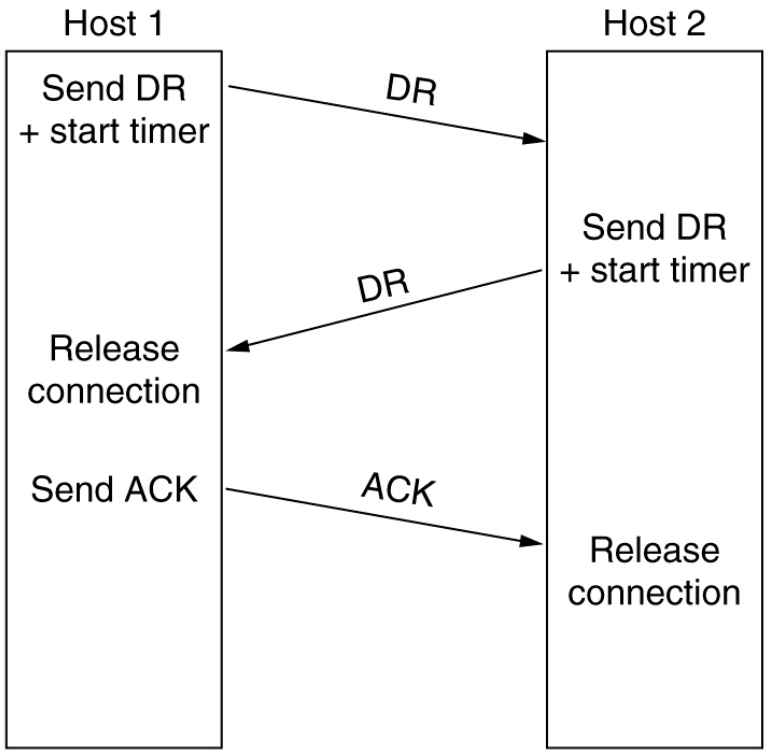


Connection Release in Practice

Take some risks when releasing a connection

Usual approach: Three-way handshake again

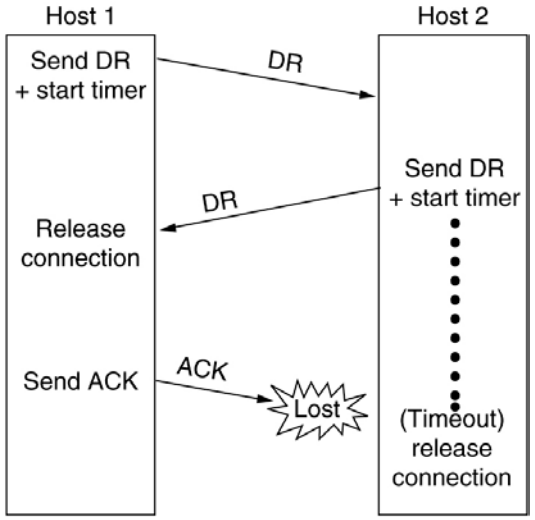
- Send disconnect request (DR)
- Set timer
- Wait for DR from peer
- Acknowledge DR
- Possibly retry
- Possibly time out



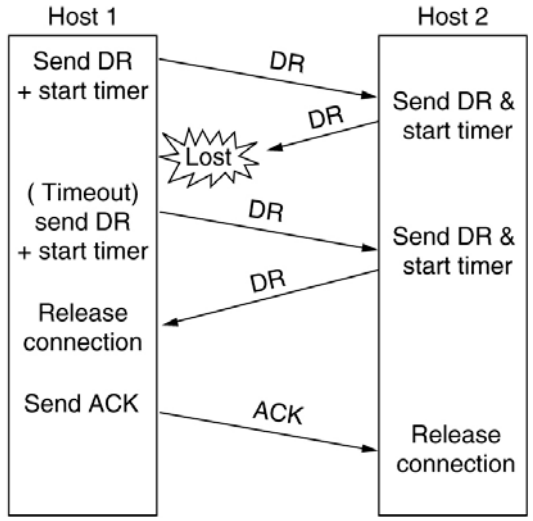
Problems for Connection Release

Problem cases for connection release with three-way handshake:

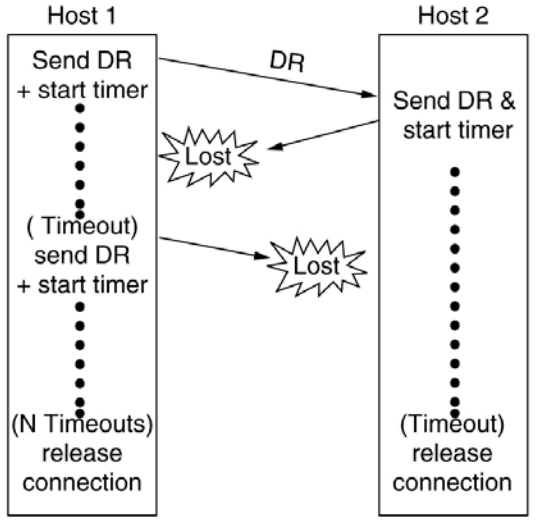
Lost ACK solved by (optimistic) timer in Host 2



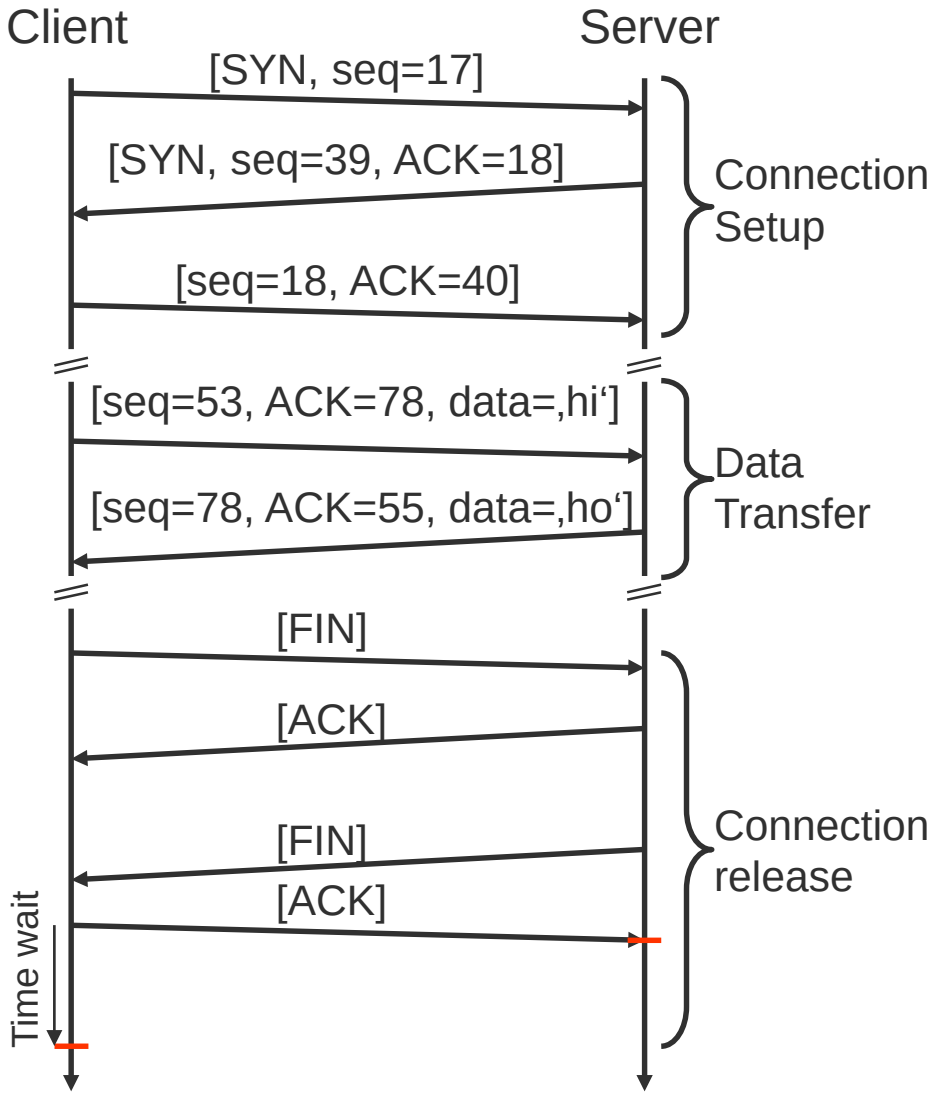
Lost second DR solved by retransmission of first DR



Timer solves (optimistically) case when 2nd DR and ACK are lost



Example: TCP Setup, Transmission, Release



Connection setup:

- Three-way handshake
- Negotiation of window size, sequence numbers, TCP options

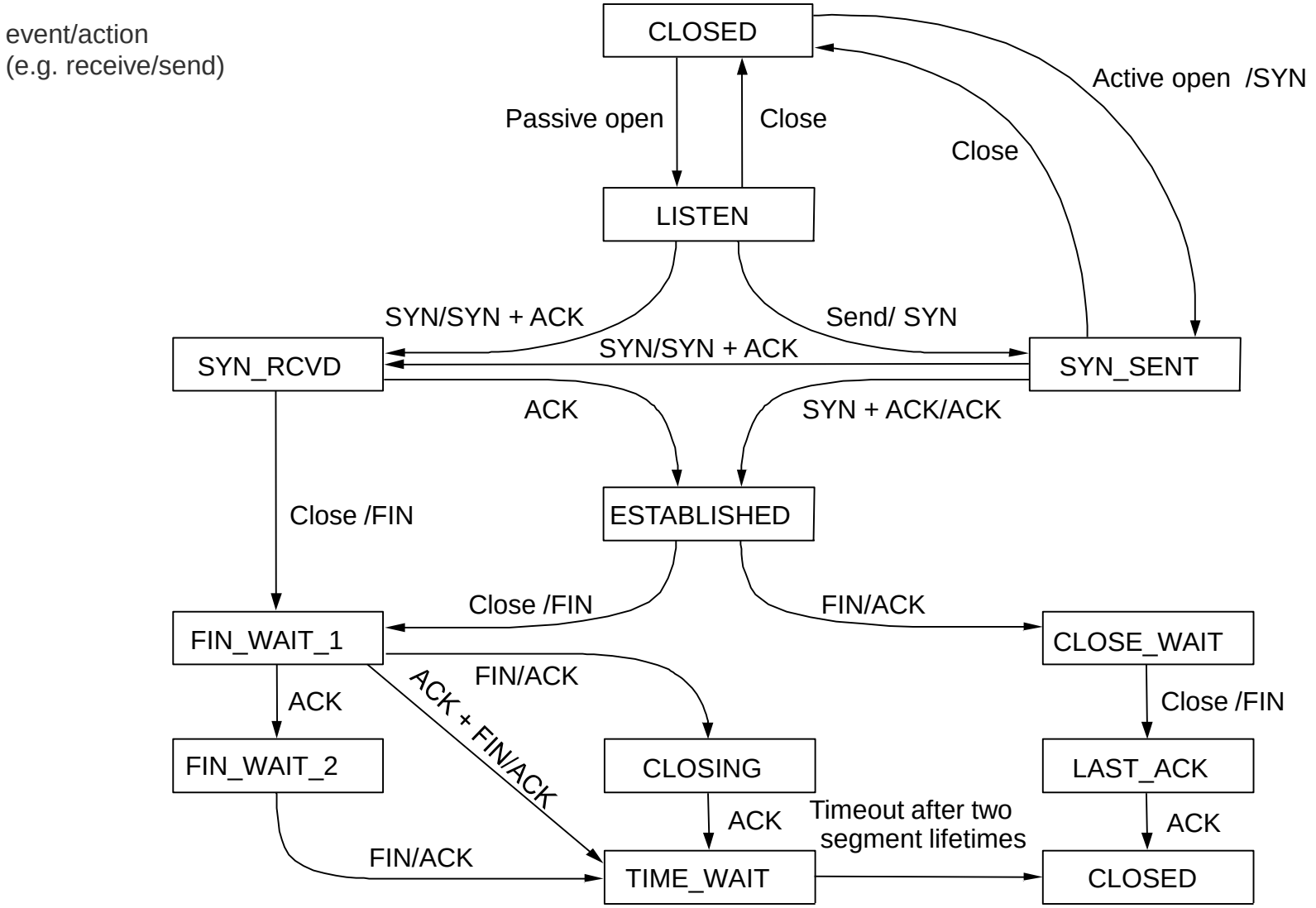
Data transfer:

- Piggybacked acknowledgements

Connection release:

- Confirmed
- Resources on client-side released after time-wait (frozen reference)
  - E.g. 30 s, 1 min, 2 min

TCP State Transition Diagram



Example

```
PS Z:\> Get-NetTCPConnection -LocalAddress 160.45.114.36
```

LocalAddress	LocalPort	RemoteAddress	RemotePort	State	AppliedSetting	OwningProcess
160.45.114.36	61291	130.133.170.130	59531	Established	Internet	7776
160.45.114.36	61289	130.133.170.200	59531	Established	Internet	7776
160.45.114.36	58783	130.133.170.200	59532	Established	Internet	7776
160.45.114.36	58500	160.45.41.101	49159	Established	Internet	2012
160.45.114.36	58424	160.45.41.36	8194	Established	Internet	2232
160.45.114.36	58366	91.190.217.52	12350	Established	Internet	8768
160.45.114.36	58363	64.4.23.168	40048	Established	Internet	8768
160.45.114.36	58362	160.45.41.142	445	Established	Internet	4
160.45.114.36	58329	160.45.41.91	445	Established	Internet	4
160.45.114.36	56972	130.133.170.200	59531	Established	Internet	7776
160.45.114.36	53168	13.107.3.128	443	Established	Internet	8768
160.45.114.36	53166	160.45.114.28	139	TimeWait		0
160.45.114.36	53157	160.45.41.90	80	TimeWait		0
160.45.114.36	53155	52.109.120.20	443	TimeWait		0
160.45.114.36	52582	65.52.108.74	443	Established	Internet	8768
160.45.114.36	51739	65.52.139.168	443	Established	Internet	8768
160.45.114.36	51507	160.45.41.90	10123	Established	Internet	3500
160.45.114.36	51345	13.92.210.230	443	Established	Internet	7596
160.45.114.36	50956	160.45.41.17	49687	Established	Internet	2012
160.45.114.36	49740	160.45.114.36	8194	Established	Internet	2192
160.45.114.36	49704	160.45.114.36	8194	Established	Internet	2192
160.45.114.36	8194	160.45.114.36	49704	Established	Internet	2232
160.45.114.36	8194	160.45.114.36	49740	Established	Internet	2232
160.45.114.36	3985	0.0.0.0	0	Listen		8768
160.45.114.36	443	0.0.0.0	0	Listen		8768
160.45.114.36	139	0.0.0.0	0	Listen		4
160.45.114.36	80	0.0.0.0	0	Listen		8768

Questions & Tasks

- Why using a three-way handshake?
- What is piggy backing?
- Why does TCP use sequence numbers? What do these numbers count?
- Why can’t we guarantee the release of a connection? How is the problem “solved” in real systems?
- What can happen if the system freezes the resources too long or too short?

Flow Control in TCP

Recall: Flow control serves to prevent a fast sender from overrunning a slow receiver

➢Similar issue in link and transport layer

Additional problems in transport layer flow control:

- Many connections, need to adapt the amount of buffer per connection dynamically
  - Instead of simply allocating fixed amount of buffer space per outgoing link
- Unlike link layer frames, transport layer PDUs can differ widely in size
- Network's packet buffering capability clouds the picture
  - Need to estimate how many packets are currently in transit

Flow Control Buffer Allocation

To support outstanding packets, sender either has to
- ... rely on receiver to process packets as they come in
  - ➢ Out-of-order delivery, no applicable to all protocols, e.g. TCP
- ... assume that receiver has sufficient buffer space available

More buffer allows for more outstanding packets
- ➢ Necessary to obtain highly efficient transmission
  - See bandwidth-delay product

How does sender have buffer assurance?
- Sender can request buffer space
- Receiver can tell sender about available buffer space
  - ➢ For sliding window protocols: Set size of sender's send window

Flow Control Permits and Acknowledgements

Two separate mechanisms:
- Permits
  - "Receiver has buffer space, go ahead and send more data"
  - ➢ Flow control
- Acknowledgements
  - "Receiver has received certain packets"
  - ➢ Error control

Can be combined with dynamically changing buffer space at receiver
- Due to different speed with which application actually retrieves received data from transport layer
- ➢ Example: TCP combines ACKs with sequence numbers

➢ Combination of mechanisms has implications:
- ➢ TCP cannot distinguish between packet loss at receiver and in transit

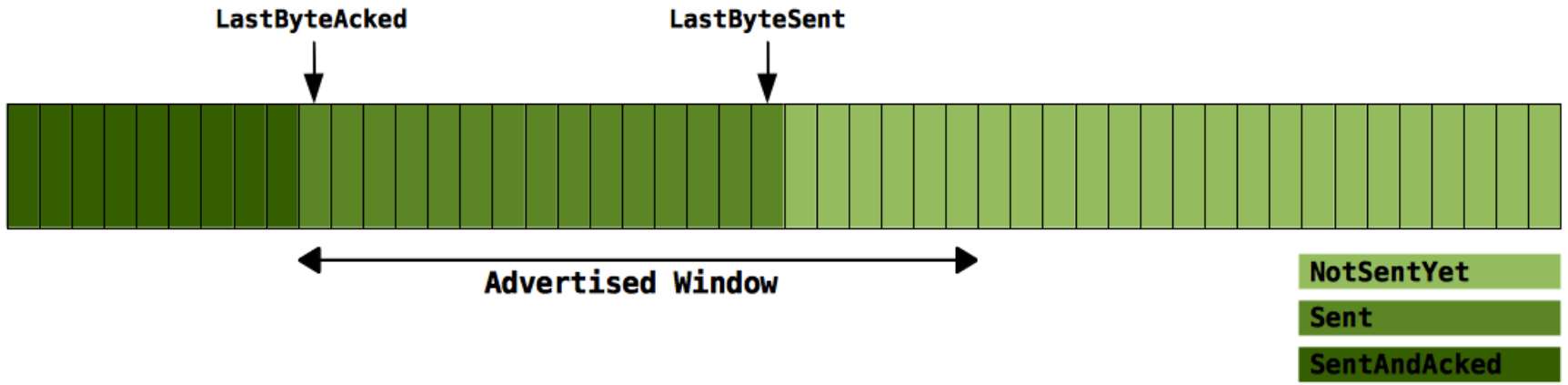
TCP Sender

Sent bytes need to be saved until they are acked

- When timer expires before a byte has been acked it is resent

Each time a packet is acked a new window size is advertised

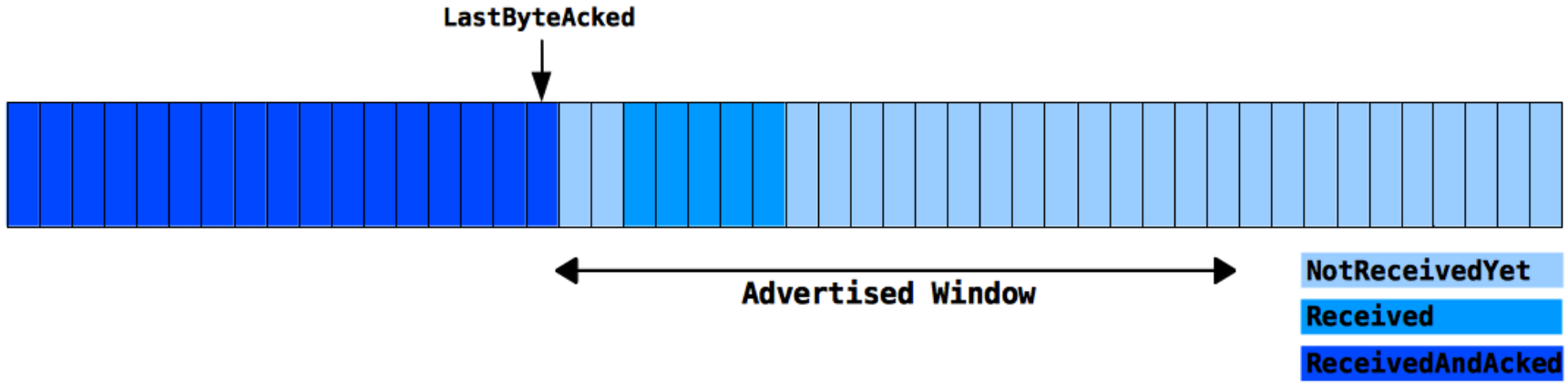
- The window is moved to the right



TCP Receiver

Application expects the data it receives to be in the correct order

➢When bytes arrive out of order, the receiver must keep space in the buffer free (gaps)



Timeout Computations

Timeouts protect against lost packets

Timeouts should reflect round-trip time (RTT) between sender and receiver

- Problem: RTTs can be highly variable
  - Range over several orders of magnitude

➢ Dynamic measurements/adaptation of RTTs

Simple approach: Keep a running average of RTTs

- Computed by an autoregressive model:

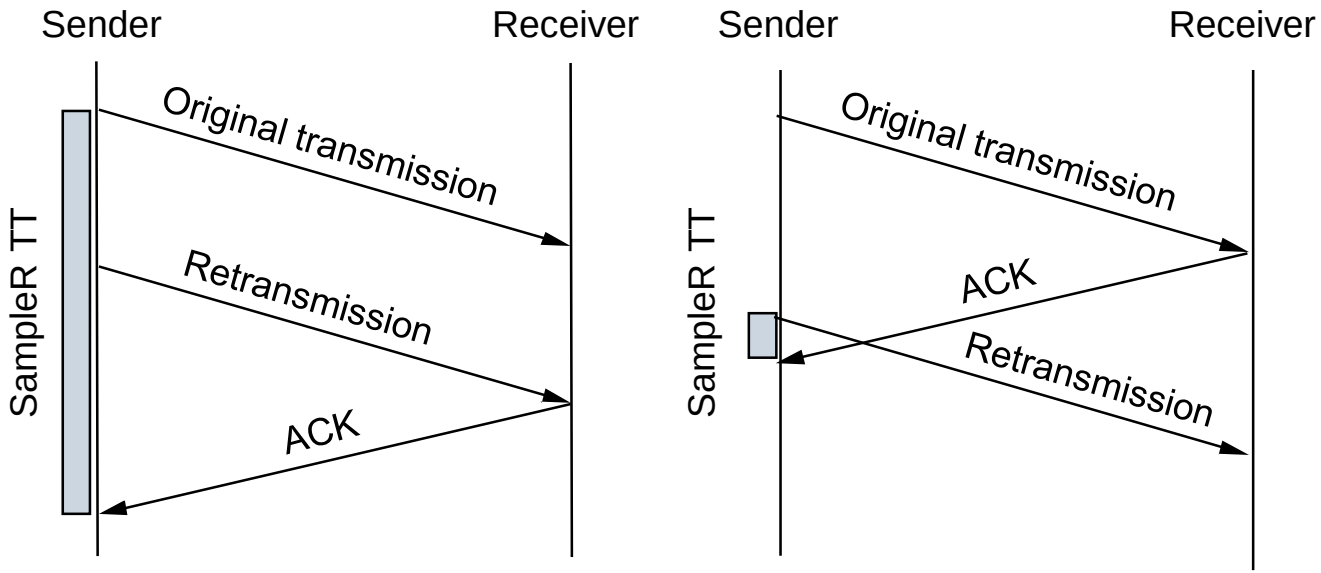
$$\text{EstimatedRTT}_n = \alpha\ \text{EstimatedRTT}_{n-1} + (1-\alpha)\ \text{RTTSample}_n$$

  - Parameter α smoothes estimation (α = 0.8, …, 0.9)
- (Conservative) timeout choice: 2 * EstimatedRTT

Problems with Timeout Computations

Simple algorithm cannot obtain correct RTT samples if packets have been retransmitted
- ACKs refer to data/sequence numbers, not to individual packets

Two examples:



➢Solutions:
- Karn/Partridge algorithm: Do not take RTT samples for retransmitted packets
- Jacobsen/Karels algorithm: Also consider variance of RTT

Timer and Packet Loss

Reaction to packet loss:

After packet loss is detected by timeout, transmission speed needs to slow down

Basic idea: Use successively larger timeout values

➢ ***Exponential backoff:*** Double timeout value for each additional retransmission
- Multiplicative factor for exponential backoff is reset upon ACK arrival
- Reset connections if maximal timeout value (given by number of retries) is exceeded

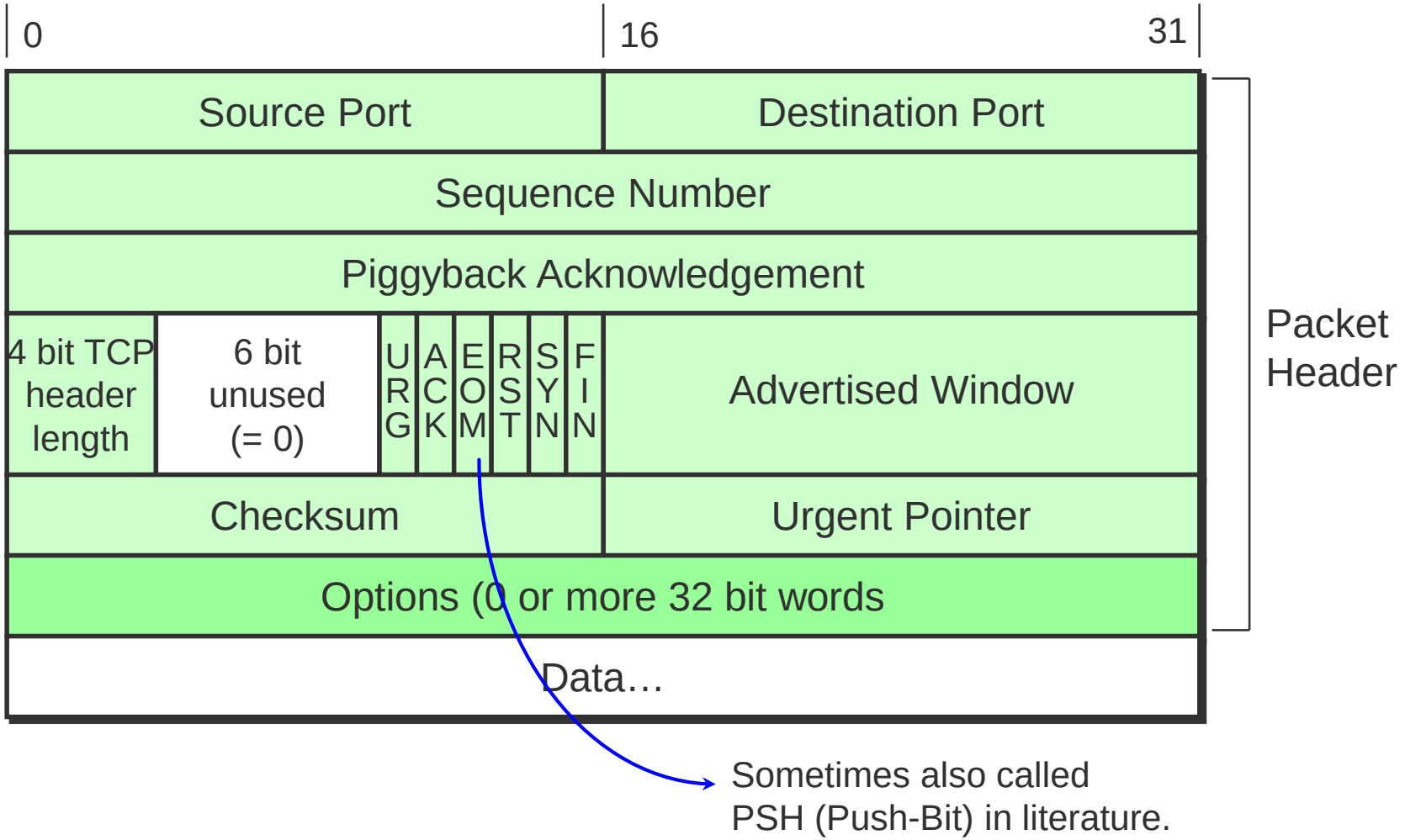
TCP Fairness & TCP Friendliness

TCP Fairness:
- Adjust dynamically to available bandwidth
- Fairly share bandwidth among all connections
  - If n connections share a given bottleneck link, each connection obtains 1/n of its bandwidth (in the long run)

Interaction with other protocols:
- Bottleneck bandwidth also depends on load of other protocols
  - Example: UDP, which is has no built-in congestion control
- UDP traffic can potentially squeeze out TCP traffic

Other transport protocols should be *TCP friendly*:
- They should not consume more bandwidth than a TCP connection in a comparable situation
- UDP is *not* TCP friendly
  - Workarounds using queuing and dropping techniques in routers
  - Alternatives are available but little used up to now, e.g. Datagram Congestion Control Protocol (DCCP)

TCP Packet Header



TCP – Summary

TCP provides a reliable byte stream using

- Connection management – three-way handshake for setup and teardown
- Error control via Go-Back-N or Selective Repeat (depending on version)
- Flow control using advertised receiver window
- Congestion control using exponential backoff, AIMD, slow start, congestion threshold (see literature)

TCP semantics/parameters are quite subtle

- Non-trivial step from unreliable datagram service to efficient reliable byte stream
- Interaction of TCP with other layers is more complicated than it looks because of hidden, implicit assumptions
  - Example: Packet loss is not an indication of congestion in wireless networks
- Many little details and extensions are not discussed here

Conclusion

Transport protocols can be anything from trivial to highly complex, depending on the purpose they serve

They determine to a large degree the dynamics of a network and – in particular – its stability
- It is trivial to build TCP protocols “faster”, but they (or the network) are less stable

Interdependencies of various mechanisms in a transport protocols can be very subtle with big consequences
- Examples: Fairness, coexistence of different TCP flavors, ...

More in Telematics
- SCTP, DCCP, MP-TCP… still ongoing research!

Content

Questions & Tasks

- What is the difference between flow, error, and congestion control?
- Why is it problematic to combine the mechanisms e.g. in TCP?
- What is the impact of the bandwidth-delay product on flow control?
- What contributes to the RTT? Why is it not that simple to calculate a correct time-out?
- TCP fairness sounds fine. But what happens in case of short-lived connections (e.g. web requests)?
- Why should protocols be “TCP friendly”? What happens if they are not?