IN2140:
Introduction to Operating Systems and Data Communication



Transport Protocols in the Internet

Introduction to TCP

TCP - Transmission Control Protocol

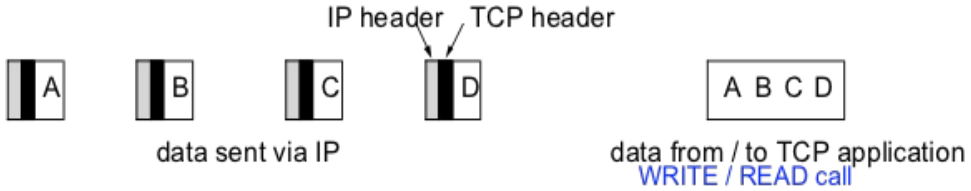
- TCP is the main transport protocol of the Internet
- TCP provides
  - Reliable end-to-end byte stream over an unreliable network service
- Motivation: network with connectionless service
  - Packets and messages may be
    - duplicated, in wrong order, faulty
    - i.e., with such service only, each 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 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

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

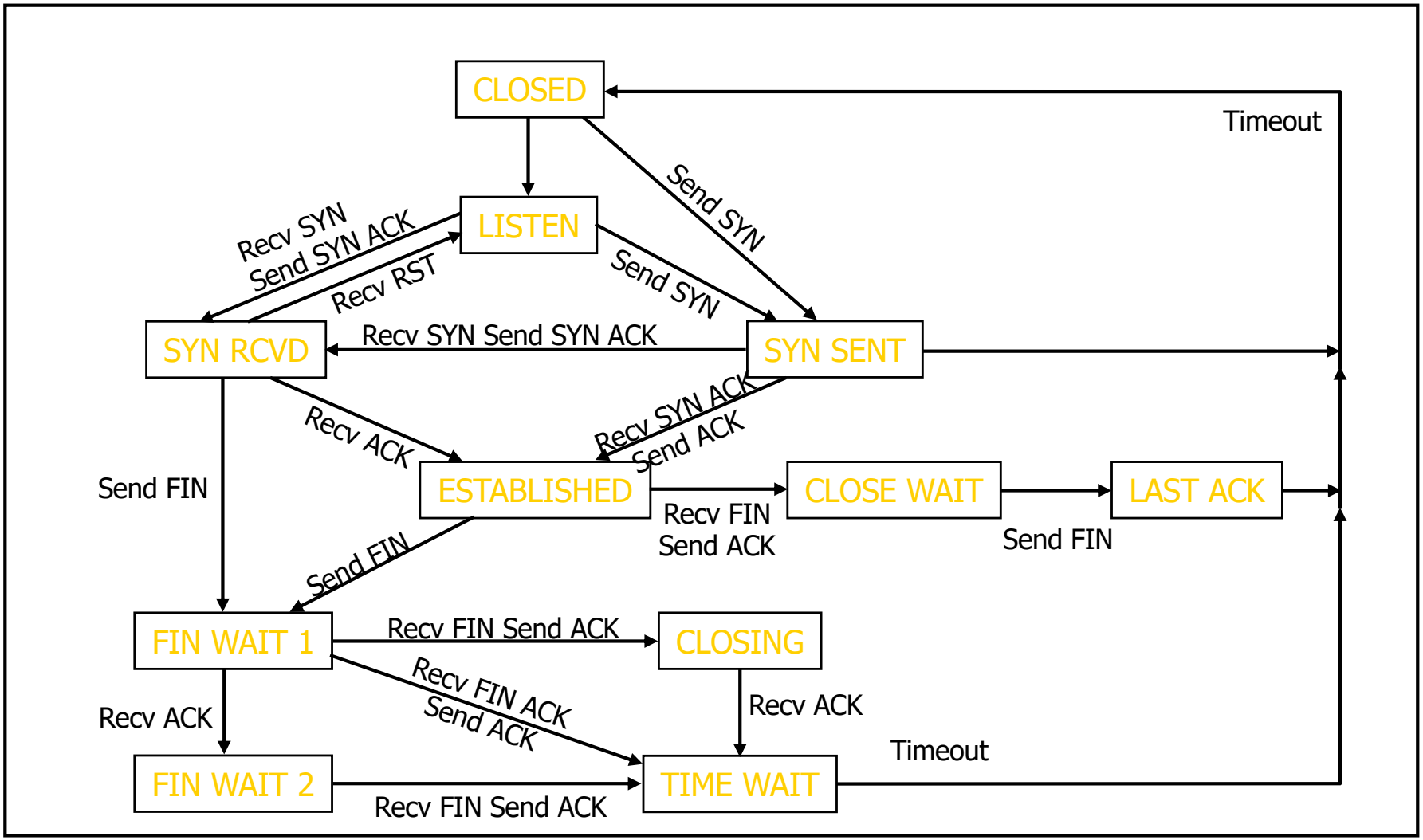


  - For transmission the sequential data stream is
    - Divided into segments
    - Delayed if necessary (to collect data)

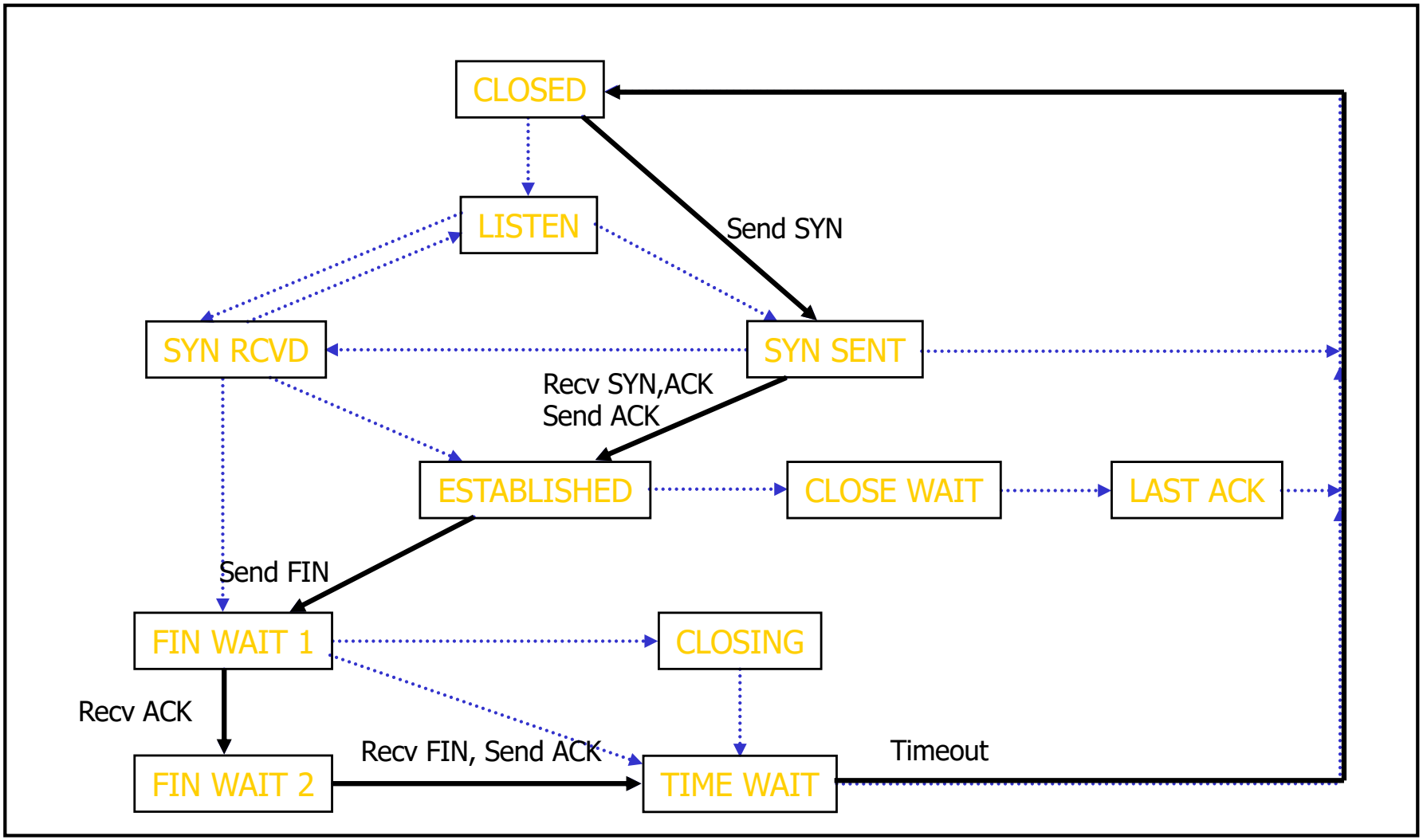
TCP Characteristics

- 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

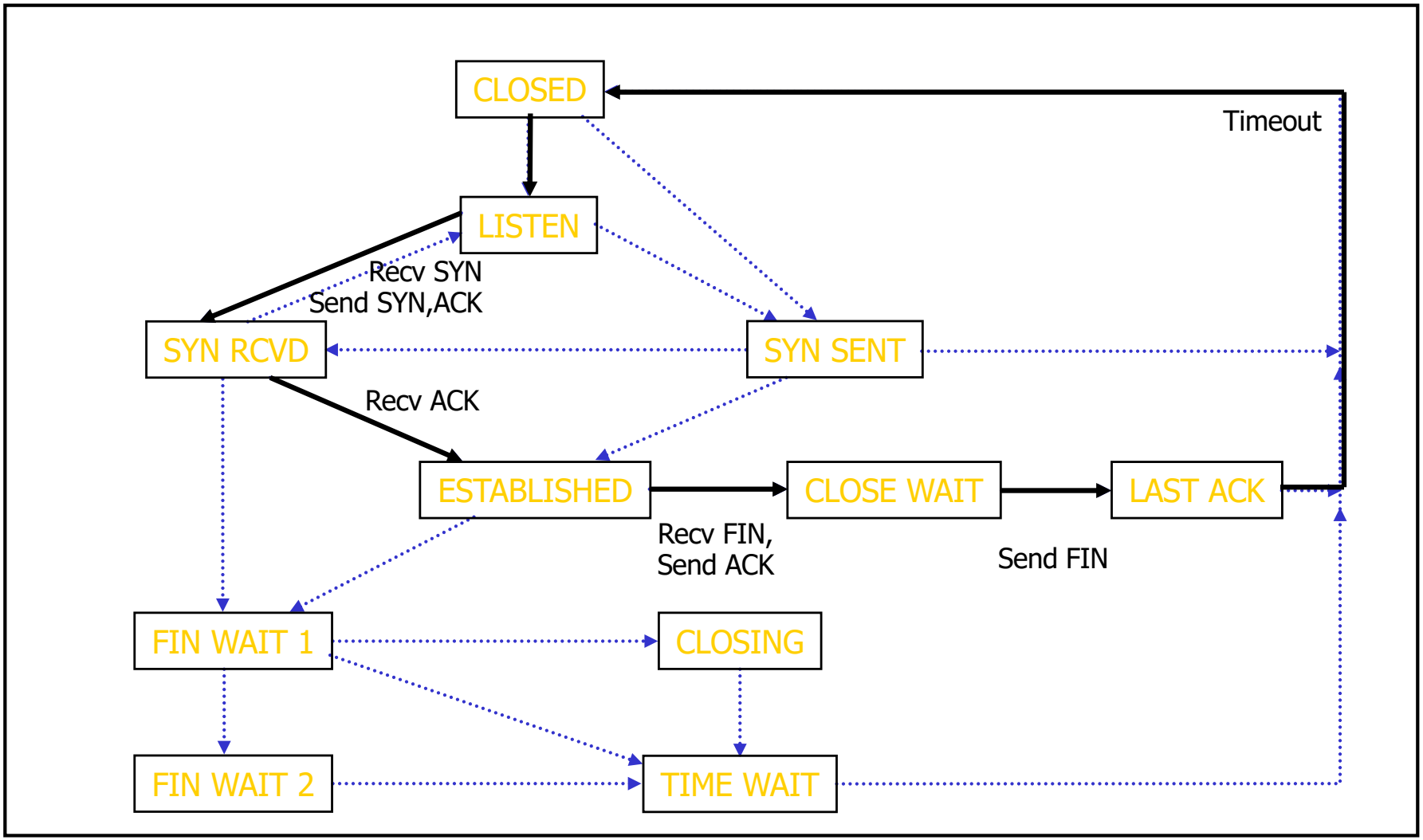
TCP State machine



Typical State Sequence of a TCP Client

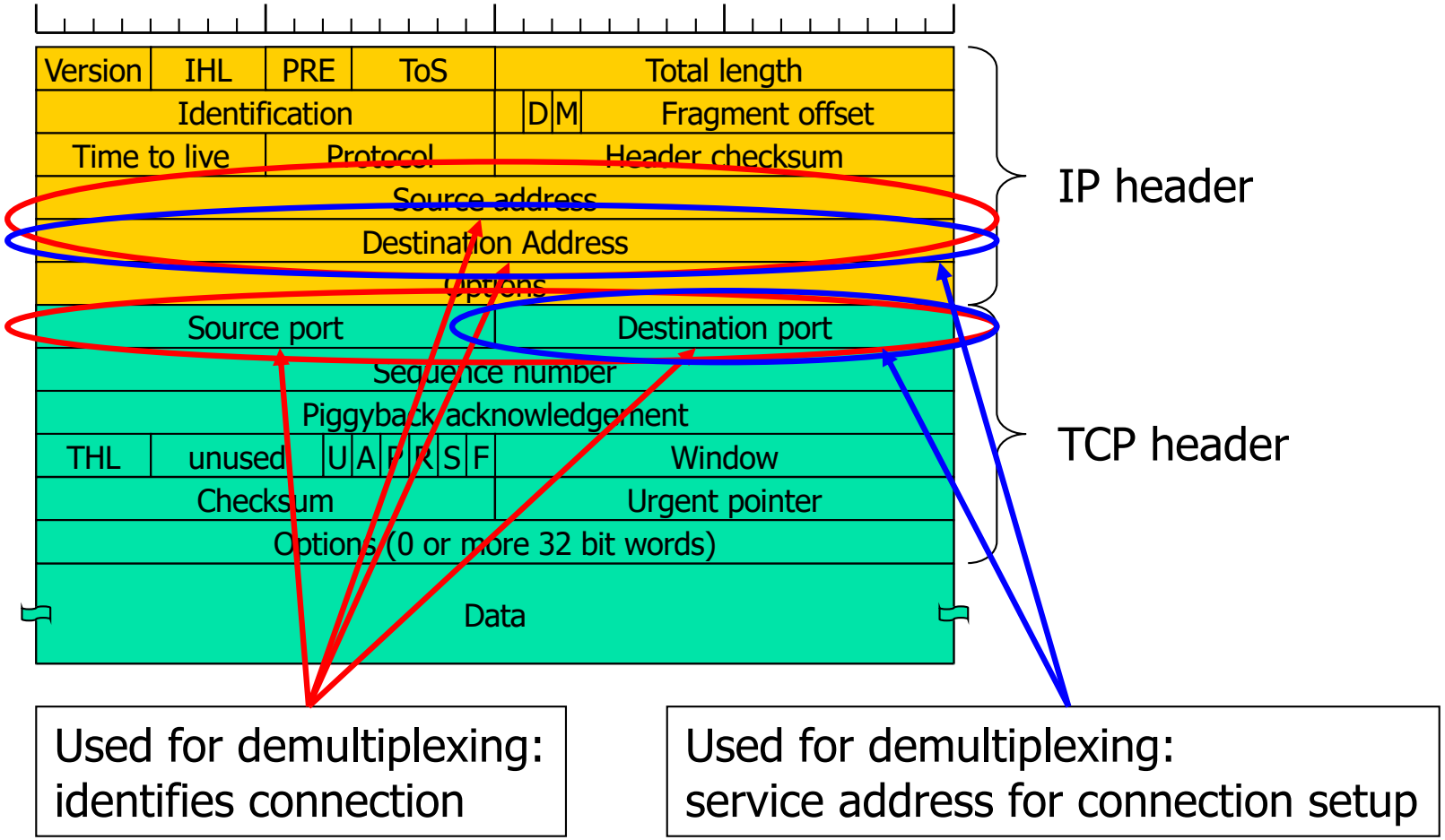


Typical State Sequence of a TCP Server



TCP: Message Format

- TCP/IP Header Format



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 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

- everything on the Web including video-on-demand (http)
- File transfer (scp, ftp)
- Interactive terminal (ssh, telnet)
- E-mail (smtp, imap)
- X-Windows

IN2140:
Introduction to Operating Systems and Data Communication



Transport layer

Terminology

Terminology: bandwidth (1/2)

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

Traditional, "real" definition!

"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"

Terminology: 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, ...

Terminology: delay

- Propagation delay - time to transfer a bit
- Latency = *propagation delay*
  + *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
- End-to-end delay – time a message needs from a sending entity to a receiving entity
  - end-to-end delay on the application layer includes TCP retransmissions
  - end-to-end delay on the transport layer does not
- Round-trip time (RTT) - time a messages needs to go from sender to receiver and back

IN2140:
Introduction to Operating Systems and Data Communication



Transport layer

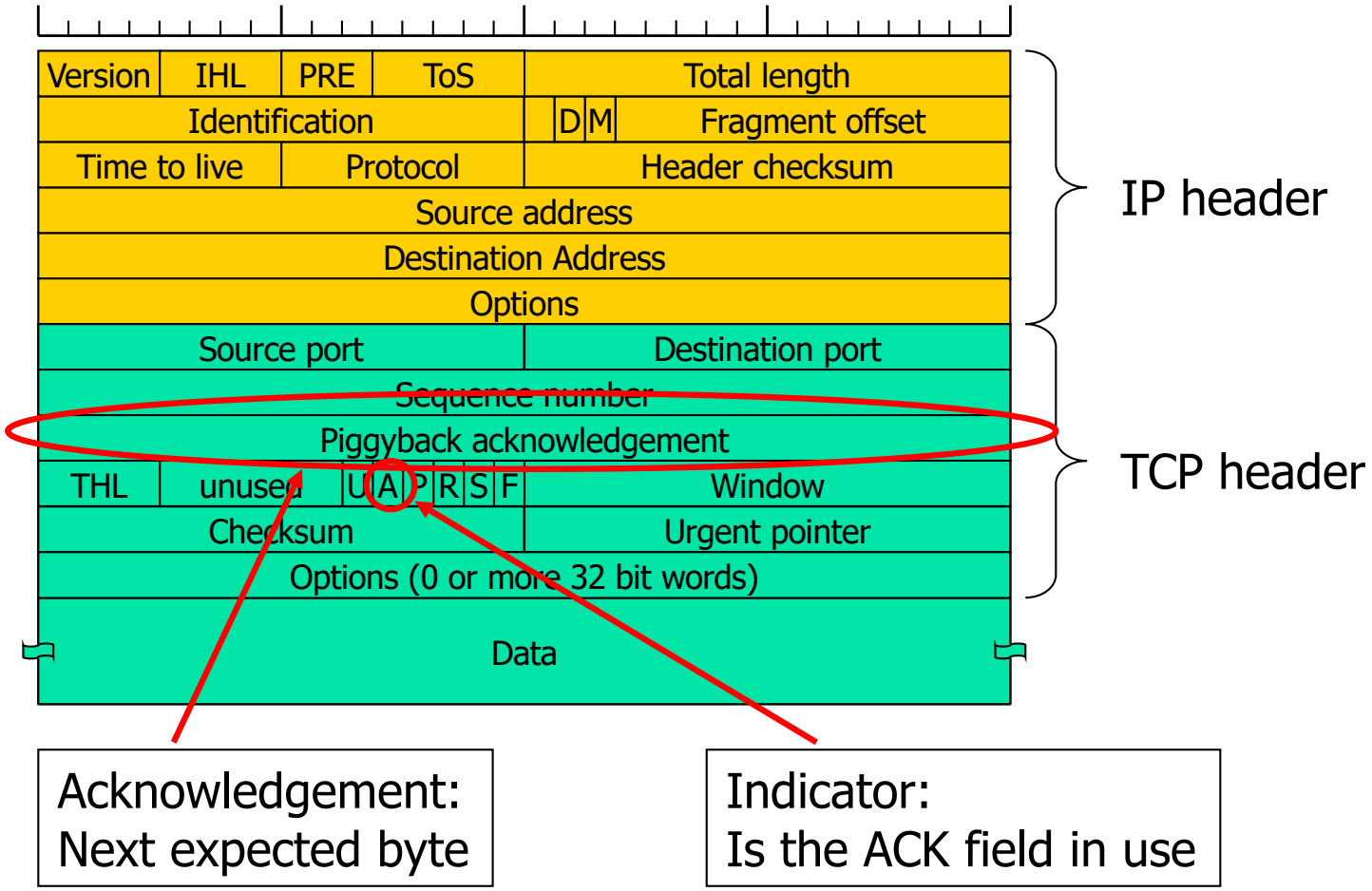
TCP reliability

TCP Reliability

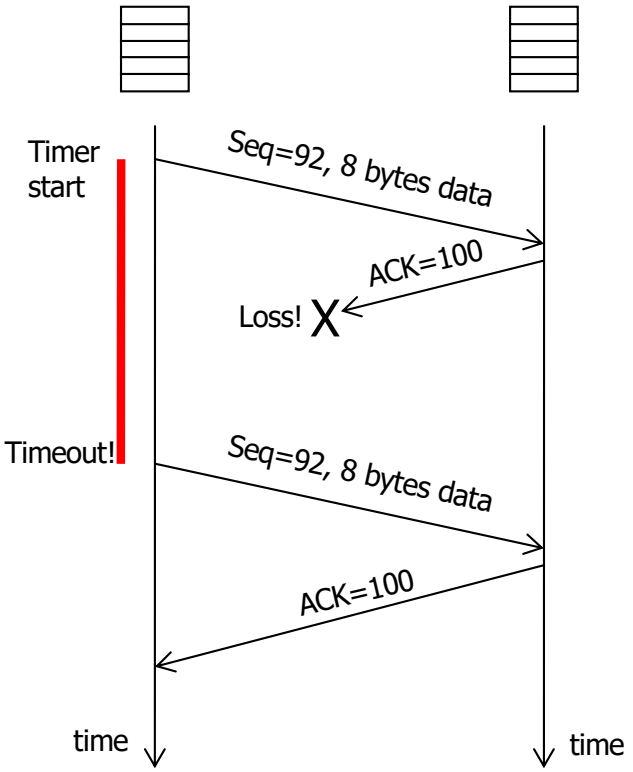
- TCP/IP has only two tools for ensuring reliable transfer:
  - detecting duplicate byte ranges
  - retransmitting data that has not arrived and not arrived correctly

TCP Reliability

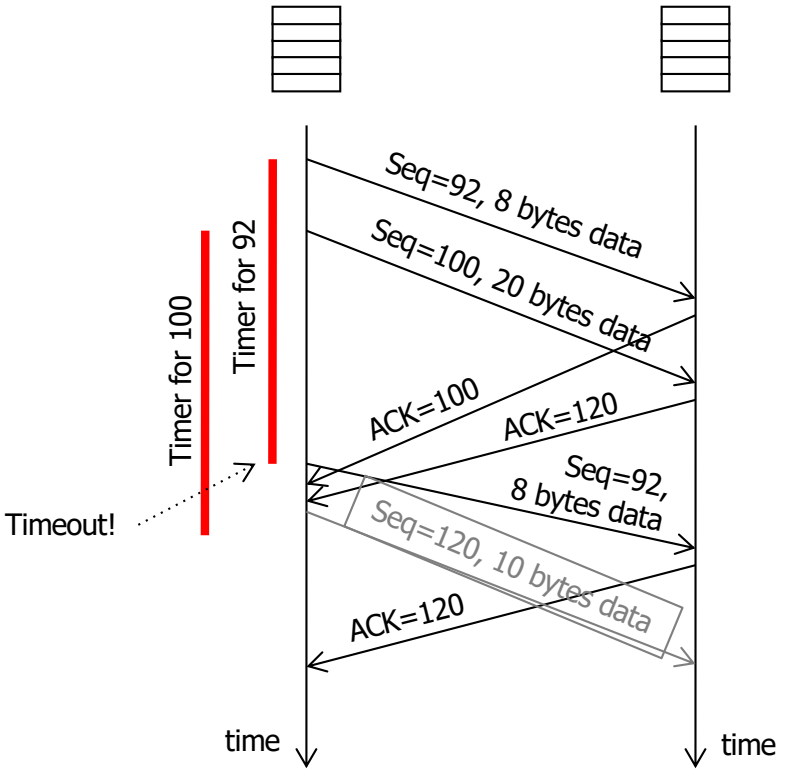
- TCP/IP Retransmits segments until they are acknowledged
- ACK field is used for that



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)

$$x = 0.1$$

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

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

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

TCP standard says 1 here, but lower-bound with 1 second. Variations exist...

IN2140:
Introduction to Operating Systems and Data Communication

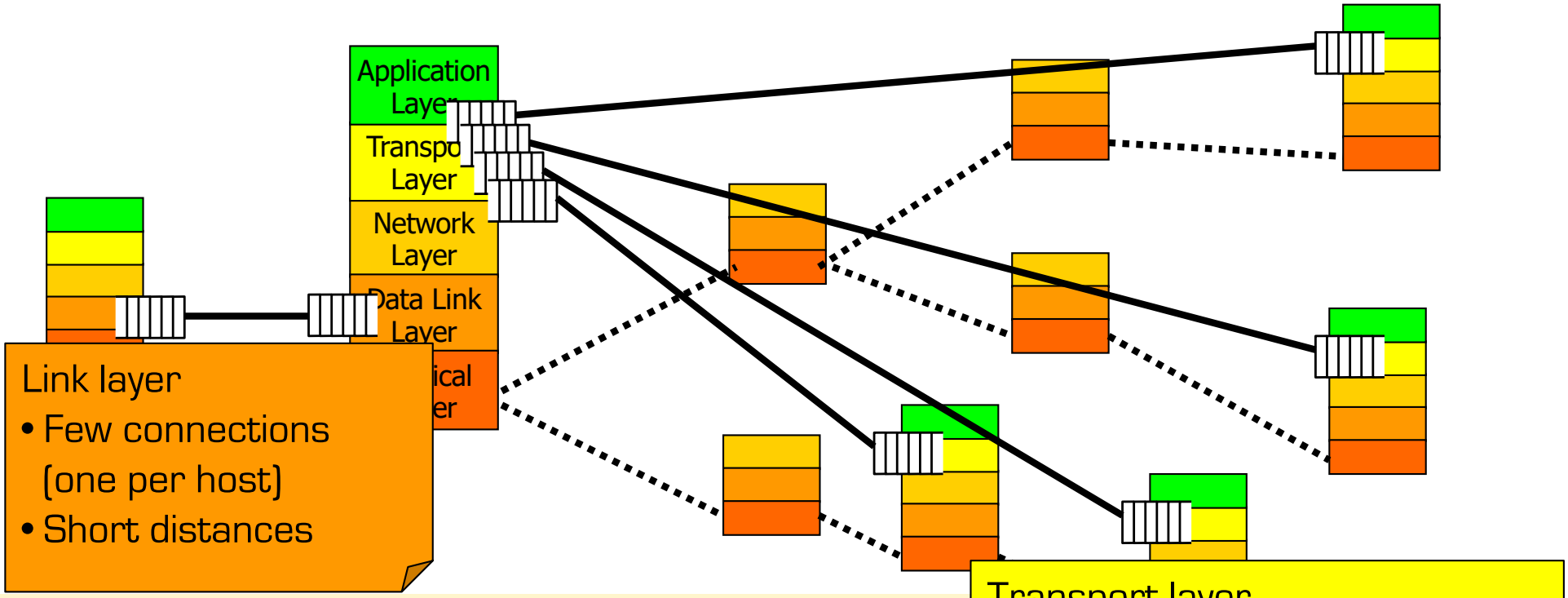


Transport layer

Flow Control

Flow Control on Transport Layer

Fast sender shall not flood slow receiver



Flow control mechanisms:

- Stop-and-wait
- Sliding window
  - Go-back-N / no buffer allocation
  - Selective repeat / static buffer allocation
  - Credit mechanism / dynamic buffer allocation

Flow control and error treatment

Problem

- sender can send faster than receiver can receive

Without flow control

- receiver loses packets despite error-free transmission

With flow control

- sender can adapt to receiver's abilities by feedback

Comment

- error control and flow control are usually interlinked
- rate control
  - controls sending speed as well
  - but defines sequencing of send operations
  - whereas flow control defines conditions for next send operation

© Ralf Steinmetz, Technische Universität Darmstadt

IN2140:
Introduction to Operating Systems and Data Communication



Transport layer

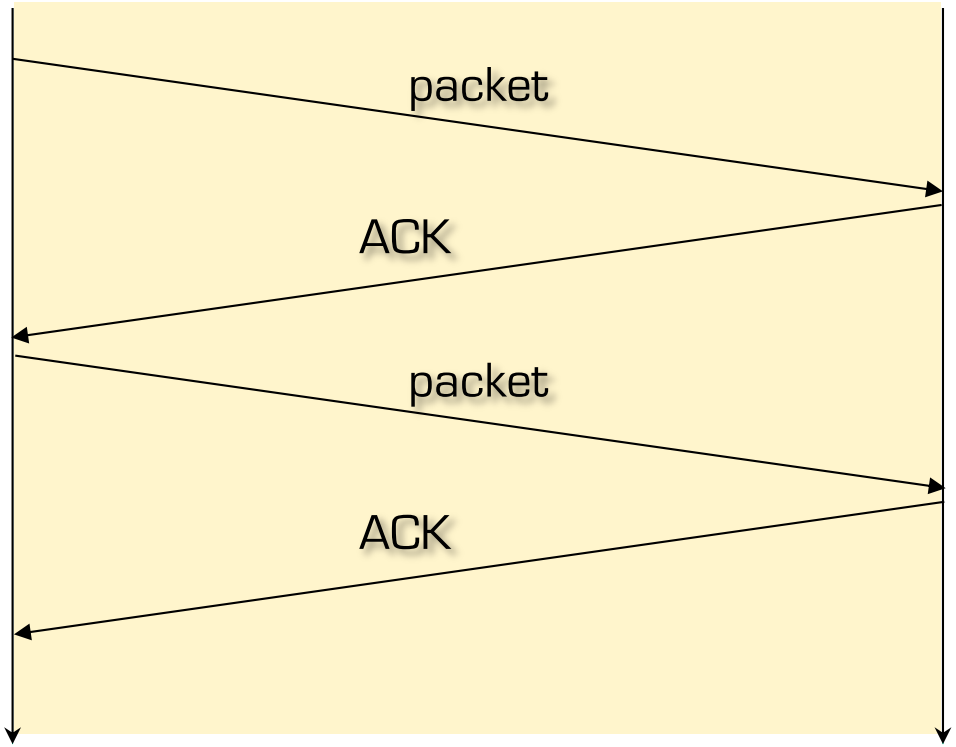
Flow Control: stop-and-wait

Flow control: Stop-and-Wait





sender receiver



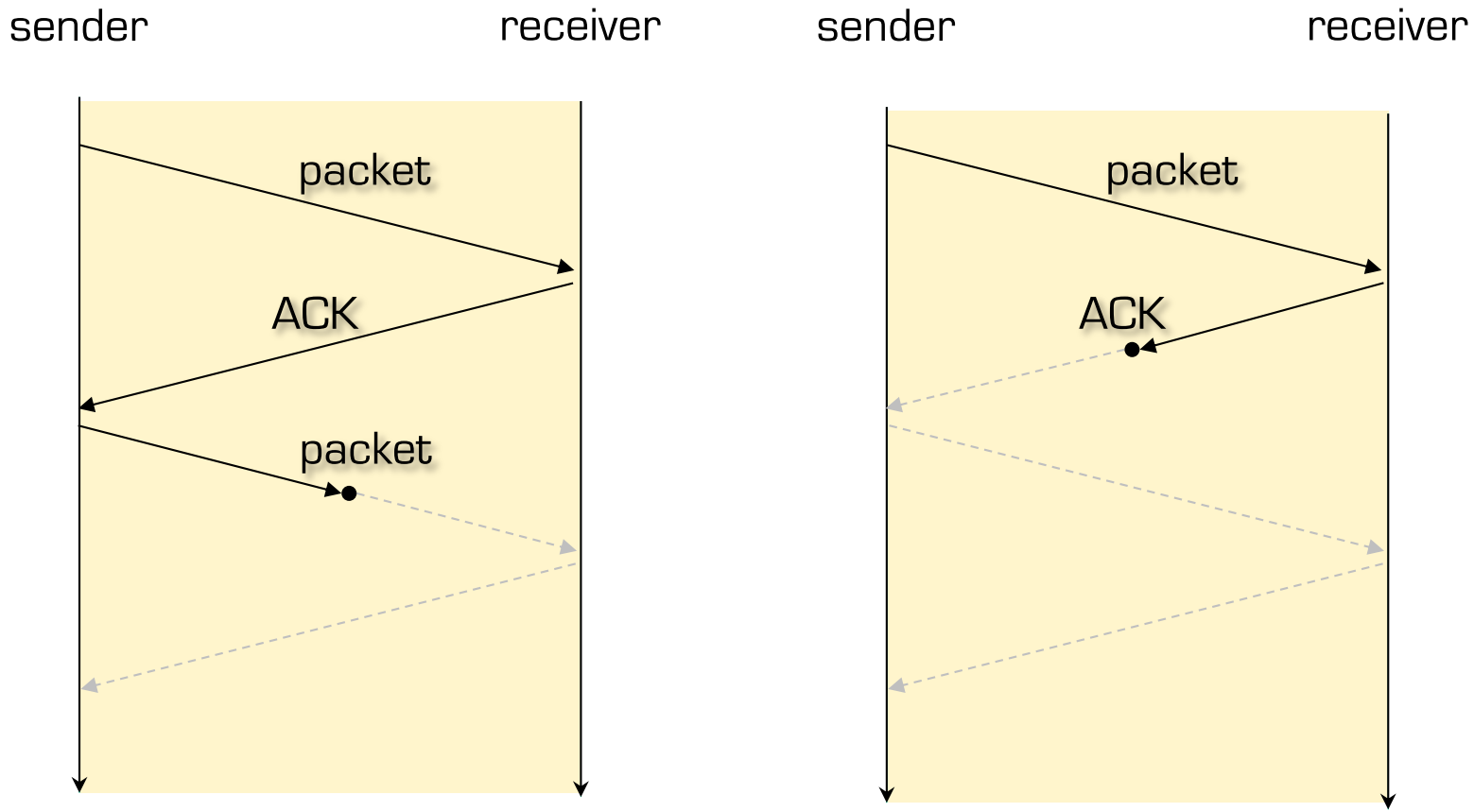
Needed:

(A) **a receiving buffer**

(B) **bi-directional communication**

Flow control: Stop-and-Wait

Packets can get lost



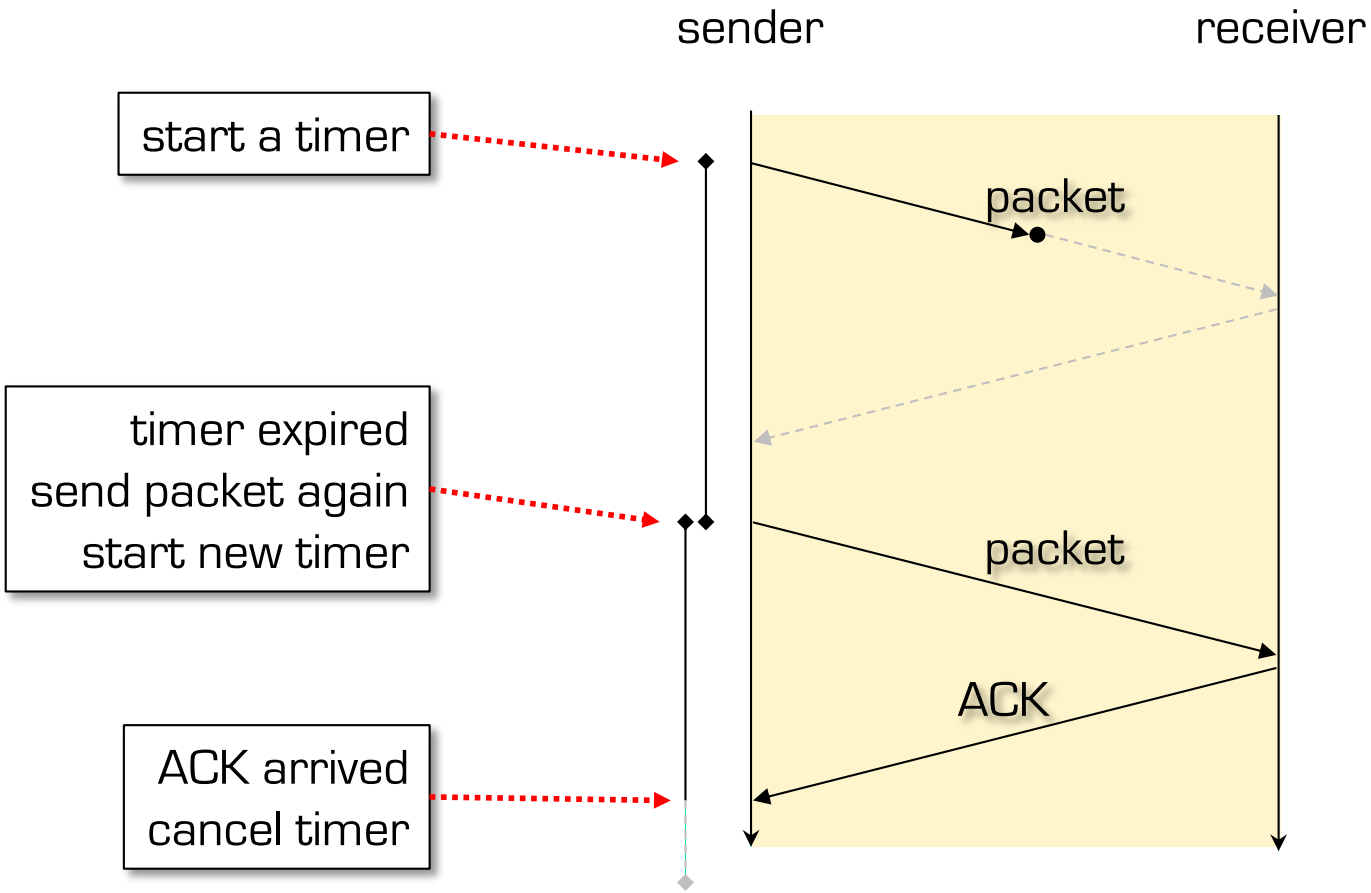
Features required by Stop-and-Wait:

(a) fix the loss

(b) distinguish packet loss and ACK loss

Flow control: Stop-and-Wait

(a) fix the loss

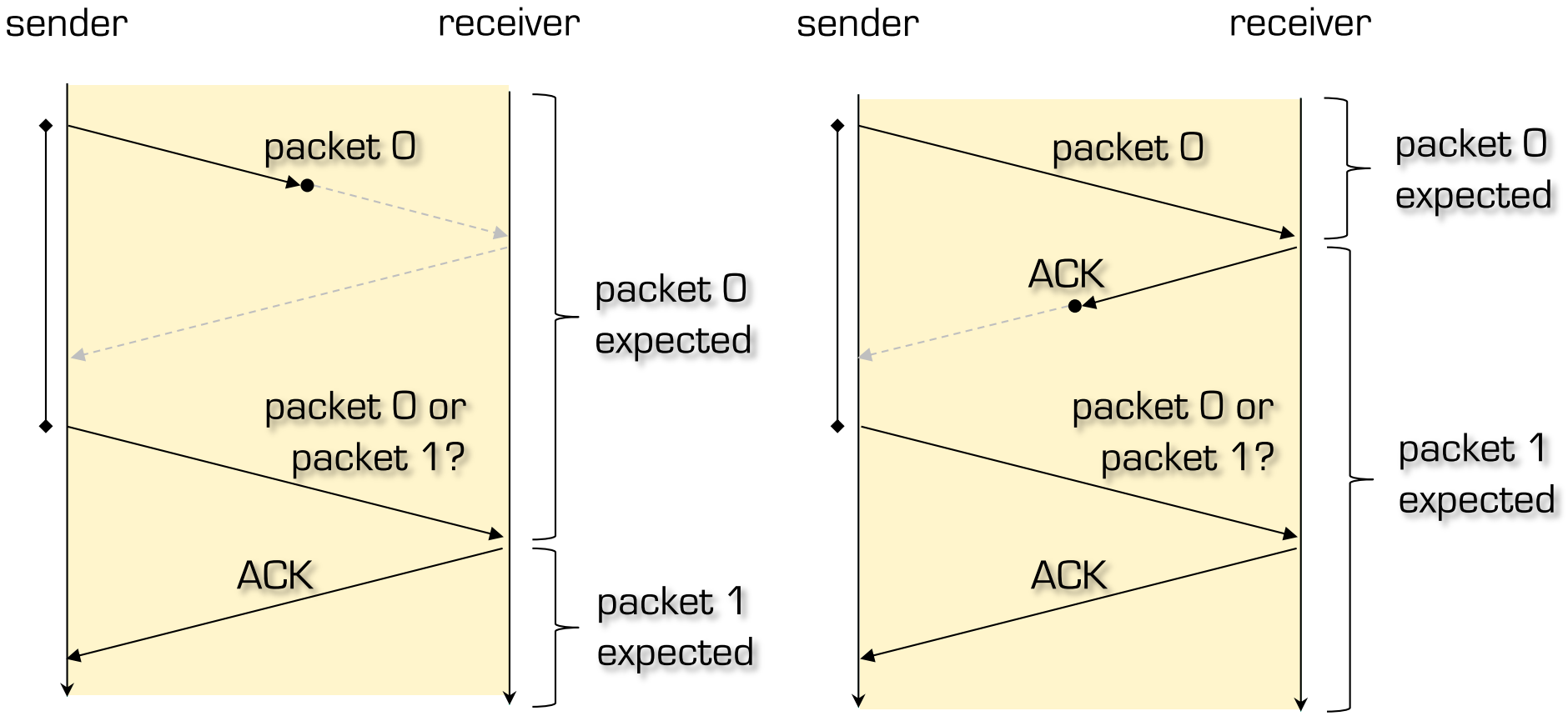


Timeout interval:

- Too short: unnecessary sending of packets
- Too long: unnecessary long wait in case of error

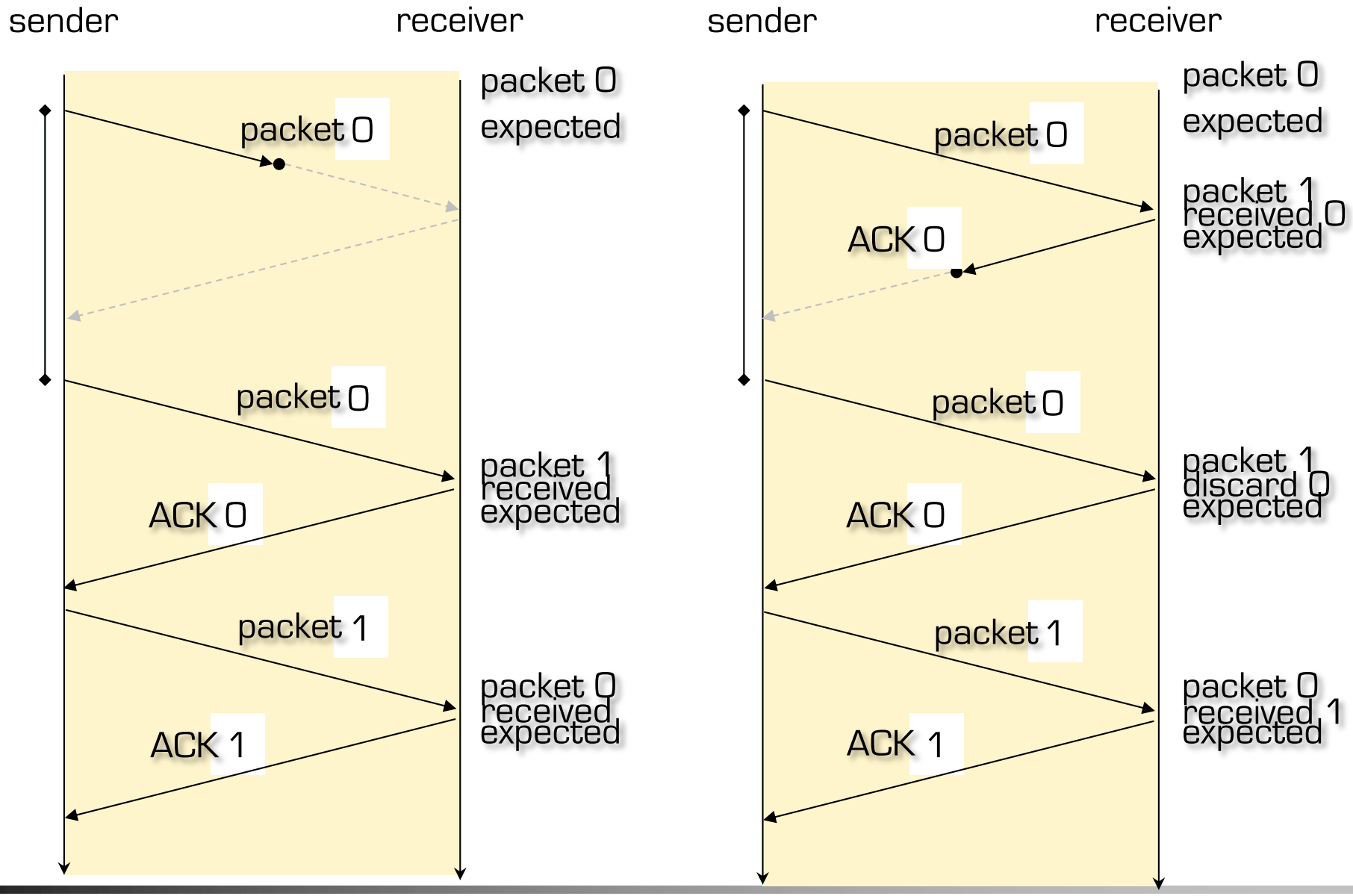
Flow control: Stop-and-Wait

(b) distinguish packet loss and ACK loss



Flow control: Stop-and-Wait

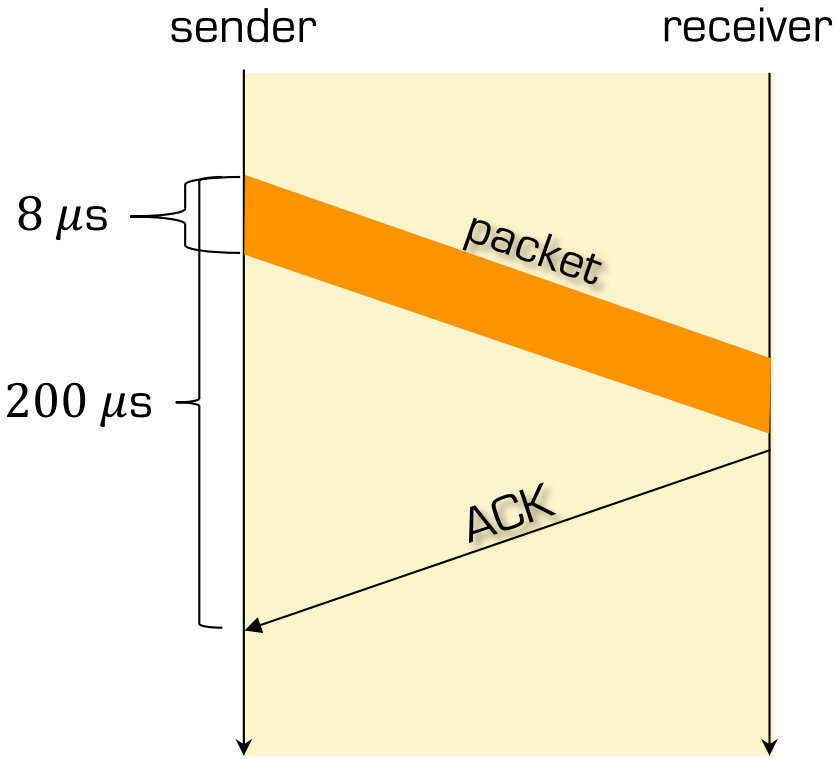
Distinguish cases by sequence number: single bit with value 0 and 1



Flow control: Stop-and-Wait

Stop-and-Wait

- sender can never send new packet before ACK or timeout
- connection is idle most of the time
- poor **throughput**



Gigabit Ethernet channel

- transmission rate: 1 Gbps
- roundtrip delay 200 μs (2*0.1 ms)
- packet size: 8000 bit
- in comparison
  → ACK is short and negligible

this means

- sending takes 8000 bit / 1.000.000.000 bps = 8 μs
- sender is blocked for 192 μs of 200 μs

→ Channel utilization 4%

IN2140:
Introduction to Operating Systems and Data Communication



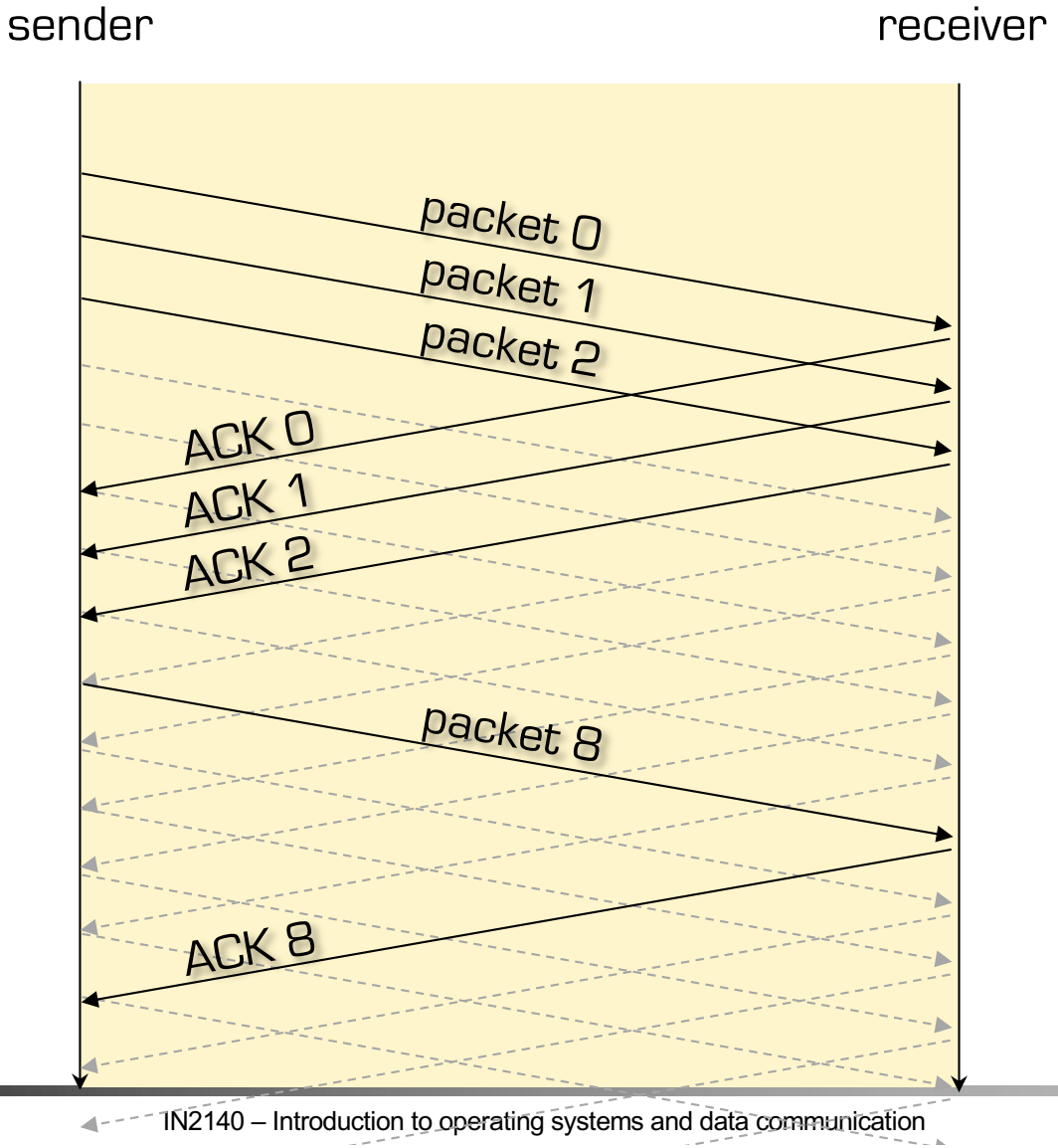
Transport layer

Flow Control: sliding window

Flow control: Sliding Window

Improve utilization: pipelining

Flow control: **sliding window** mechanism



Sliding Window: Concept

1st goal of link flow control:
receiver buffer must not overflow



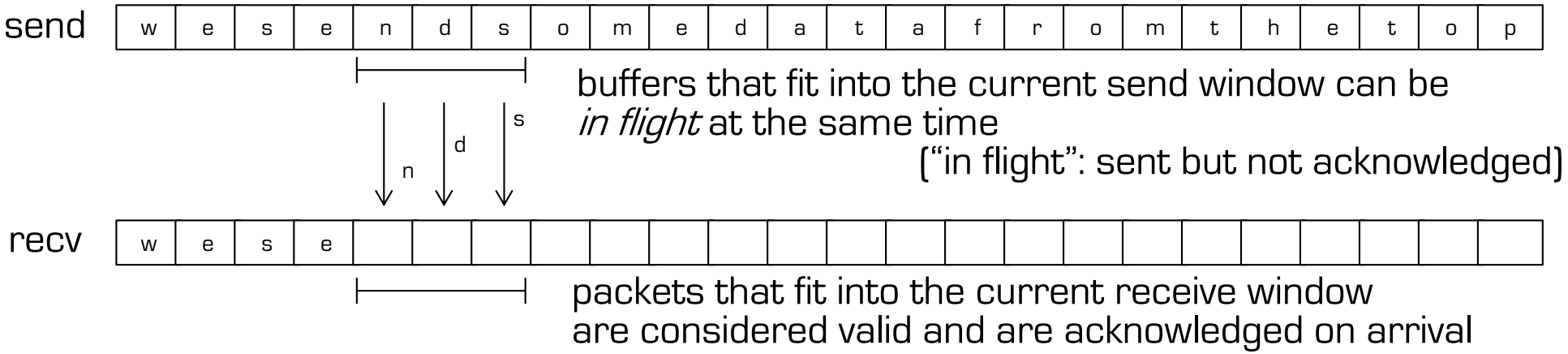
Assumptions:
each buffer can contain one packet

Two windows per communication relationship
Sender Window (or Send Window)
- packets that were sent but not yet acknowledged

Receiver Window (or Receive Window)
- packets that can be accepted

packets are identified by sequence numbers

but the packets keep coming and these numbers will wrap eventually



Sliding Window: Concept

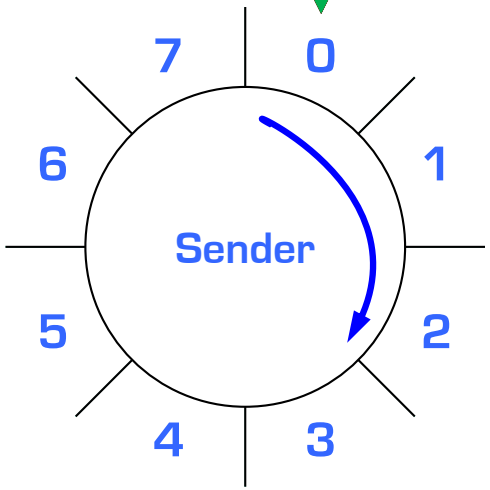
Sender side

Example:
SeqNo: [0..7]
Window Size (**WS**): 3

2 variables

lower bound — LB — 0

upper bound — UB — 0



- LB: oldest seqno that is still unconfirmed
- UB: next seqno to be sent
- if LB==UB :
  nothing is in flight
  sender is idle

advancing sender seqno (modulo 8)
- LB: when ACK is received
- UB: when sending a packet

Sliding Window: Concept

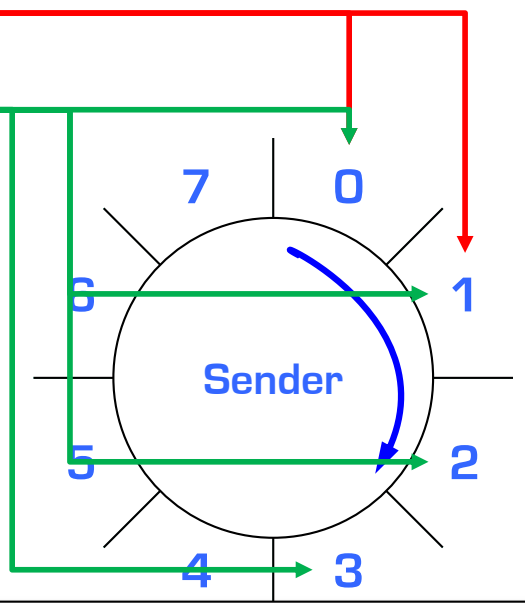
Sender side

Example:
SeqNo: [0..7]
Window Size (**WS**): 3

2 variables

lower bound — LB — 1

upper bound — UB — 3



- LB: oldest seqno that is still unconfirmed
- UB: next seqno to be sent
- if LB==UB :
  nothing is in flight
  sender is idle

increase UB (modulo 8 !)
when sending a packet

increase LB (modulo 8 !)
when receiving an ACK

send packet(0) packet(1) packet(2) WS=3

recv ACK(0): WS=2

Sliding Window: Concept

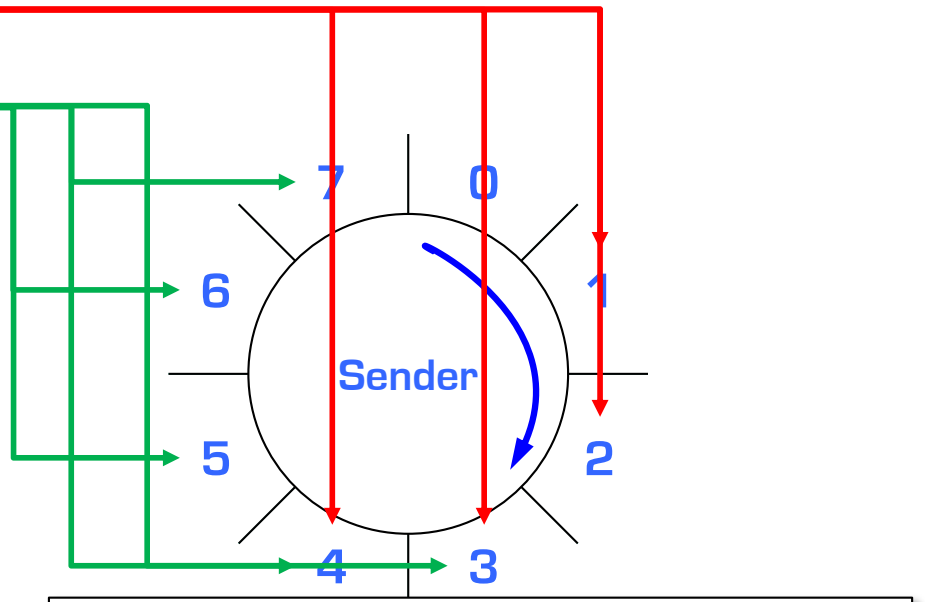
Sender side

Example:
SeqNo: [0..7]
Window Size (**WS**): 3

2 variables

lower bound — LB — 4

upper bound — UB — 7



- LB: oldest seqno that is still unconfirmed
- UB: next seqno to be sent
- if LB==UB :
  nothing is in flight
  sender is idle

increase UB (modulo 8 !)
when sending a packet

increase LB (modulo 8 !)
when receiving an ACK

send packet(0) packet(1) packet(2) WS=3

recv ACK(0): WS=2

send packet(3) WS=3

recv ACK(1), ACK(2), ACK(3) WS=0

send packet(4) packet(5) packet(6) WS=3

Sliding Window: Concept

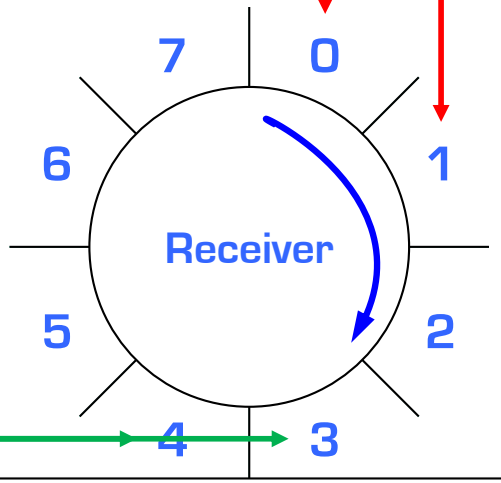
Receiver side

Example:
SeqNo: [0..7]
Window Size (**WS**): 3

2 variables

lower bound — LB — 1

upper bound — UB — 4



- LB: lowest valid seqno that should be ACKed
- UB: lowest invalid seqno
- if UB==LB: buffer full
- if UB==(LB+3)%8: nothing to ACK

increase LB (modulo 8 !)
when receiving a packet

increase UB (modulo 8 !)
when sending an ACK

recv packet(0) ... WS=2

send ACK(0) WS=3

Sliding Window: Concept

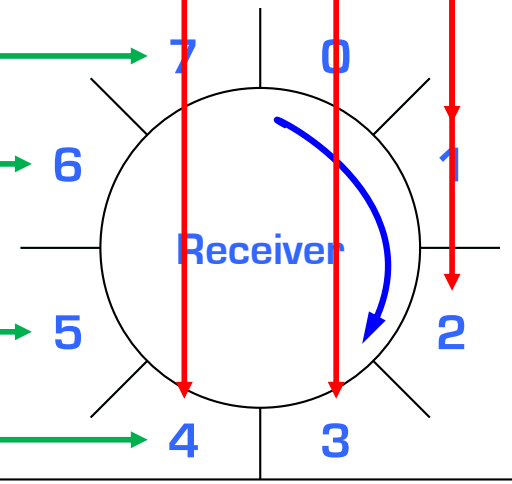
Receiver side

Example:
SeqNo: [0..7]
Window Size (**WS**): 3

2 variables

lower bound — LB — 4

upper bound — UB — 7



- LB: lowest valid seqno that should be ACKed
- UB: lowest invalid seqno
- if UB==LB: buffer full
- if UB==(LB+3)%8: nothing to ACK

increase LB (modulo 8 !)
when receiving a packet

increase UB (modulo 8 !)
when sending an ACK

recv packet(0) ... WS=2

send ACK(0) WS=3

recv packet(1) packet(2) packet(3) WS=0

send ACK(1) ACK(2) ACK(3) WS=3



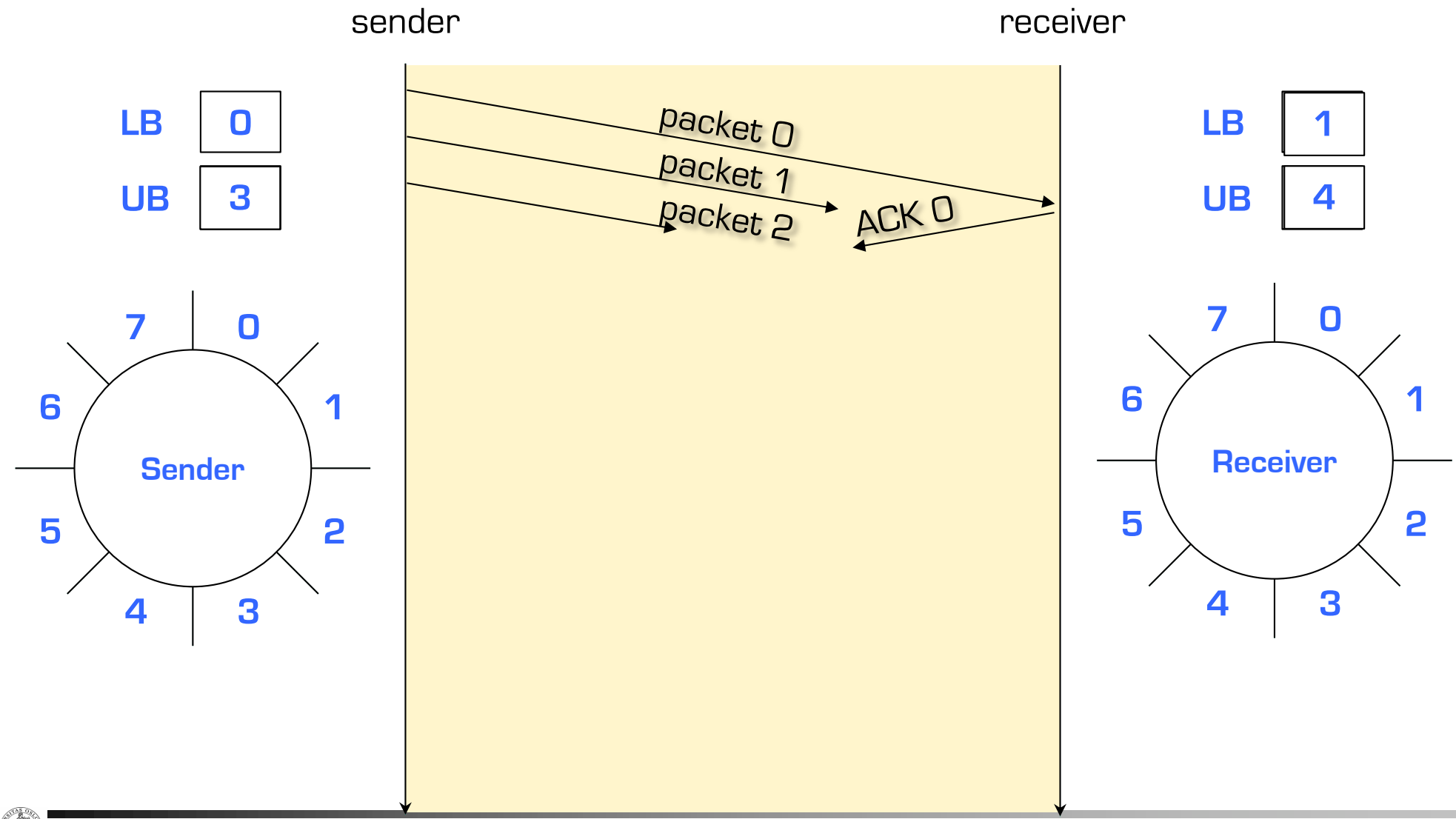
Transport layer

Flow Control: cumulative ACK

Sliding Window: Concept

Feature: Cumulative ACK

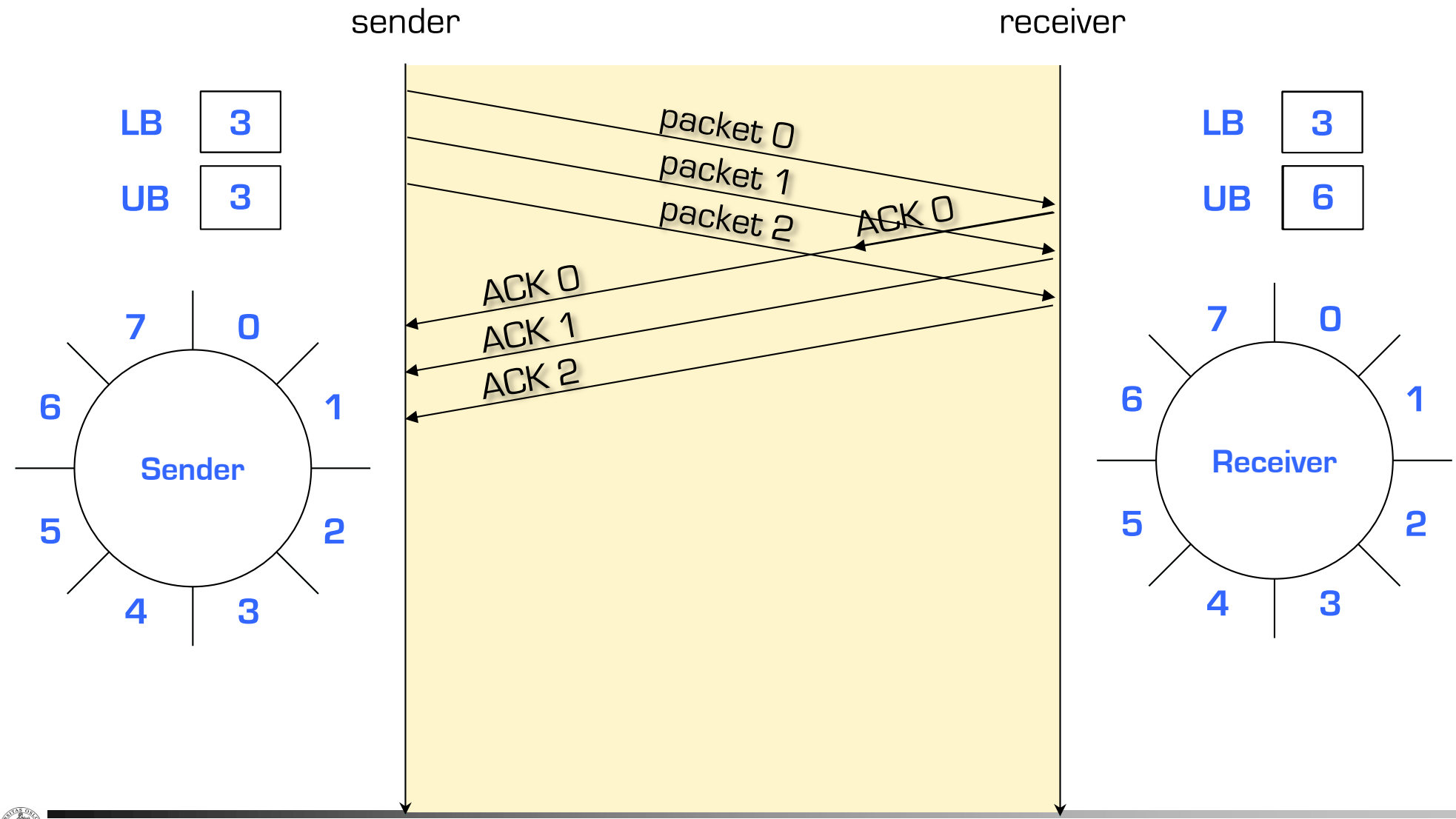
Example:
SeqNo: [0..7]
Window Size (**WS**): 3



Sliding Window: Concept

Feature: Cumulative ACK

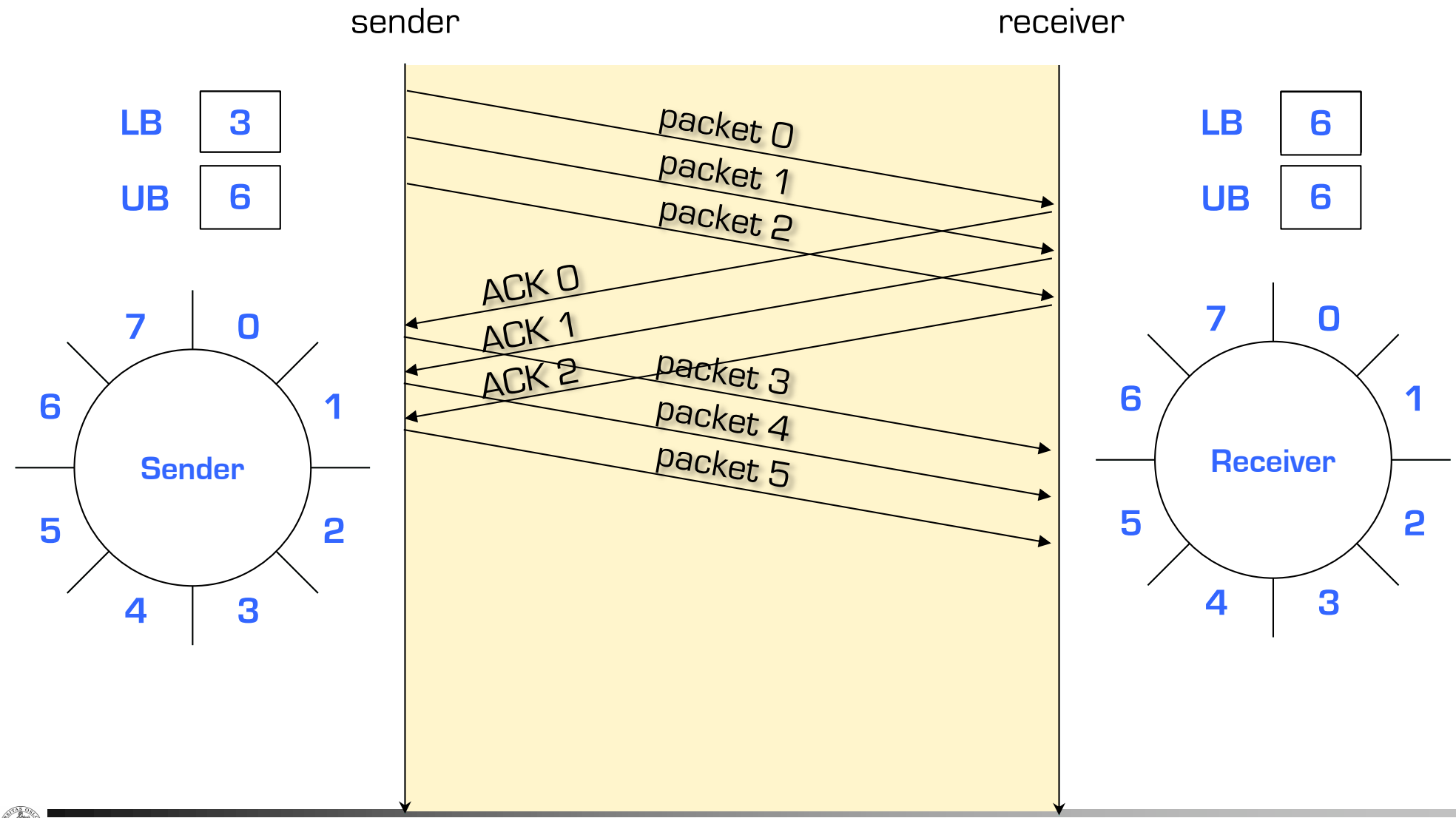
Example:
SeqNo: [0..7]
Window Size (**WS**): 3



Sliding Window: Concept

Feature: Cumulative ACK

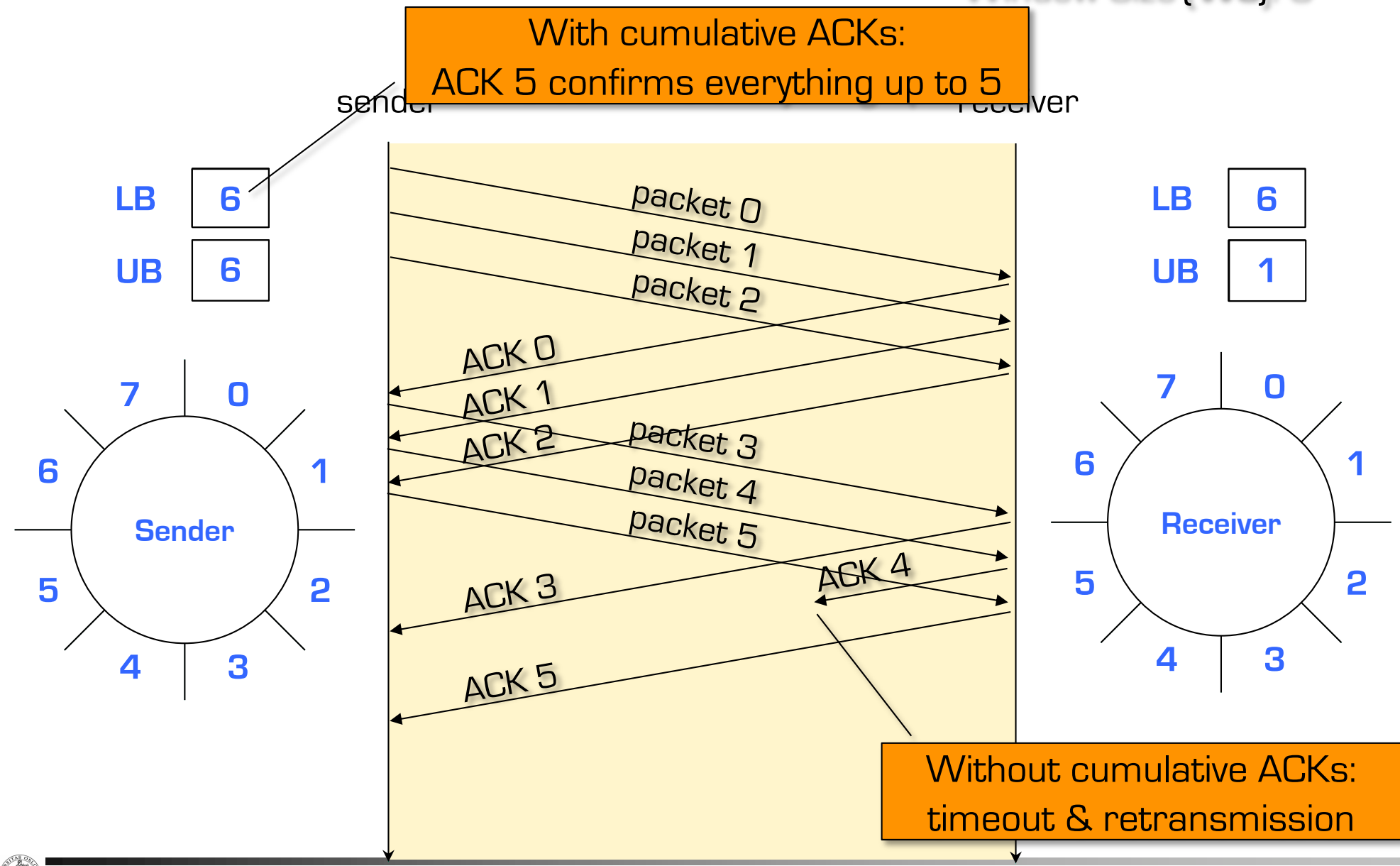
Example:
SeqNo: [0..7]
Window Size (**WS**): 3



Sliding Window: Concept

Feature: Cumulative ACK

Example:
SeqNo: [0..7]
Window Size (**WS**): 3



Sliding Window

Stored packets at the sender

- the sender must store all unacknowledged packets and be able to retransmit
- maximum number defined by sender's window size (here 3)
- the packets not yet acknowledged by the receiver

Stored packets at the receiver

- not **necessary** to store any packets
  - receive buffer must hold at least one packet, but can be passed to next layer right away
- not **useful** to store more than one receiver window size

ACK sent by receiver only if the packet

- **has been** identified as being correct
- **can be** transmitted correctly to the application
  - *correct* includes "in the right order"
  - *correct* includes usually: free from detected bit errors

IN2140:
Introduction to Operating Systems and Data Communication

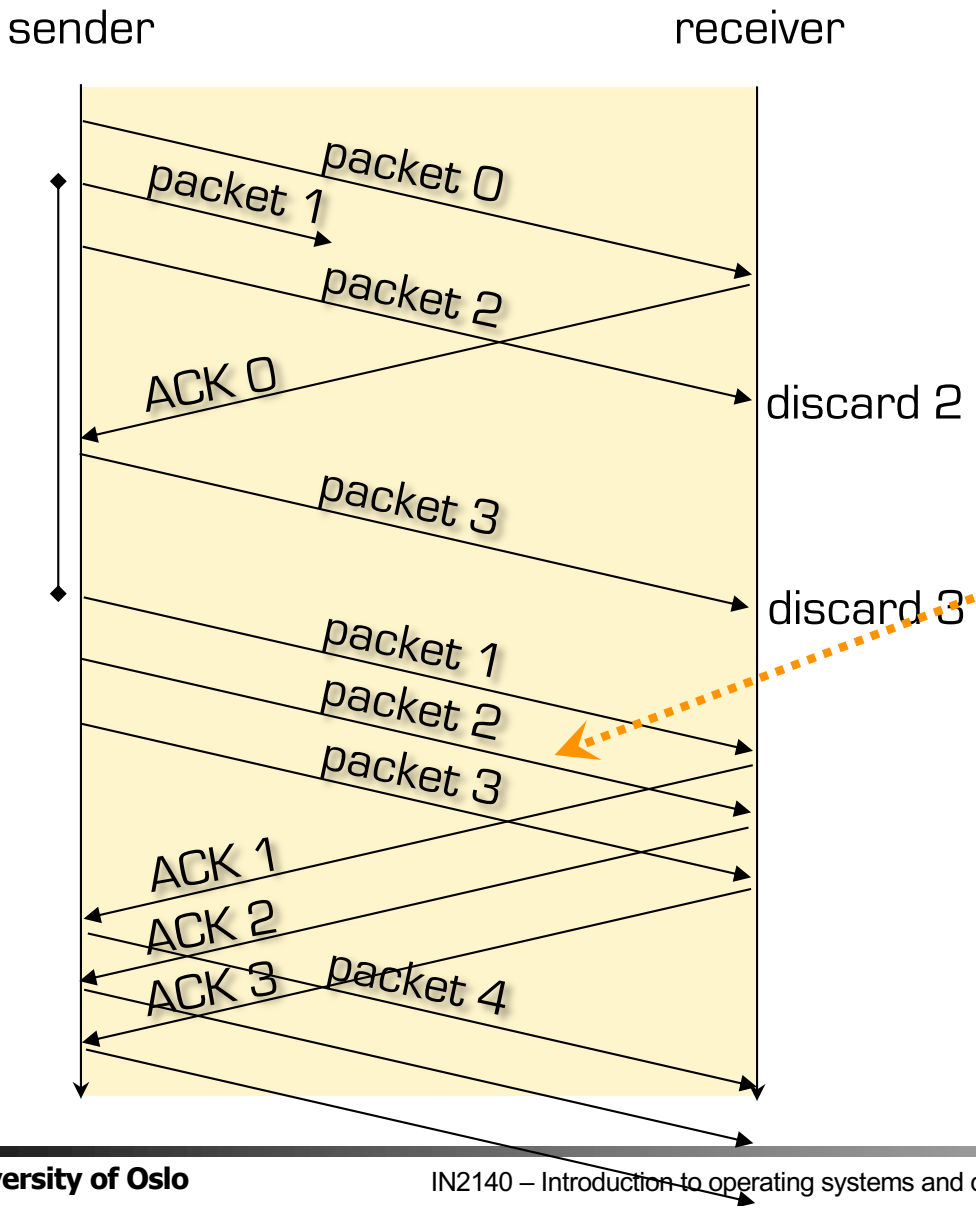


Transport layer

Flow Control: retransmission

Sliding Window

Retransmission approaches



This approach is called

Go-Back-N

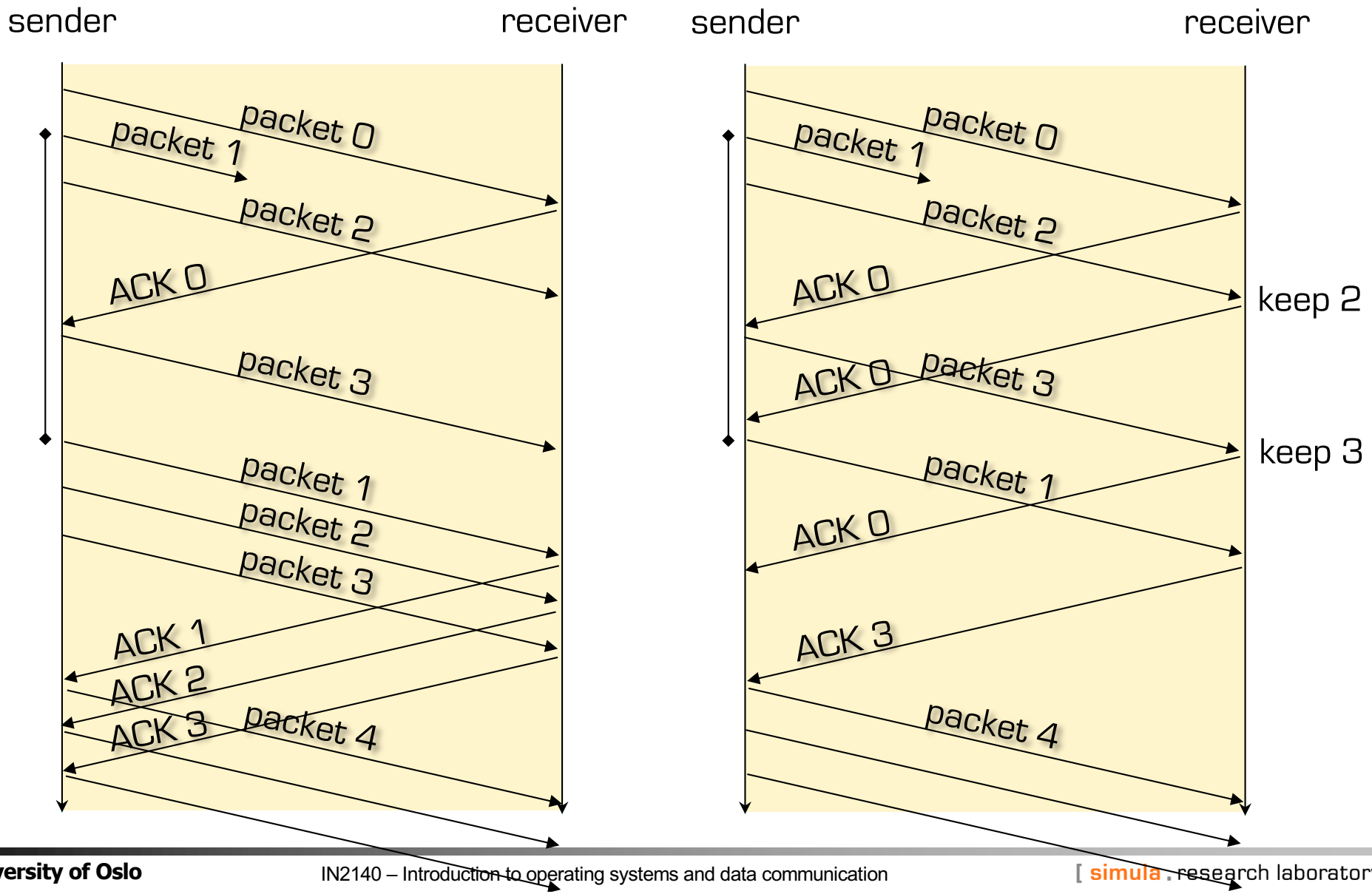
Receiver accepts packets only in order

Advantages:

- receiver only needs buffer for 1 packet
- if the network usually loses >1 packets in a row, this fixes the problem quickly

Sliding Window

Retransmission approaches



Sliding Window

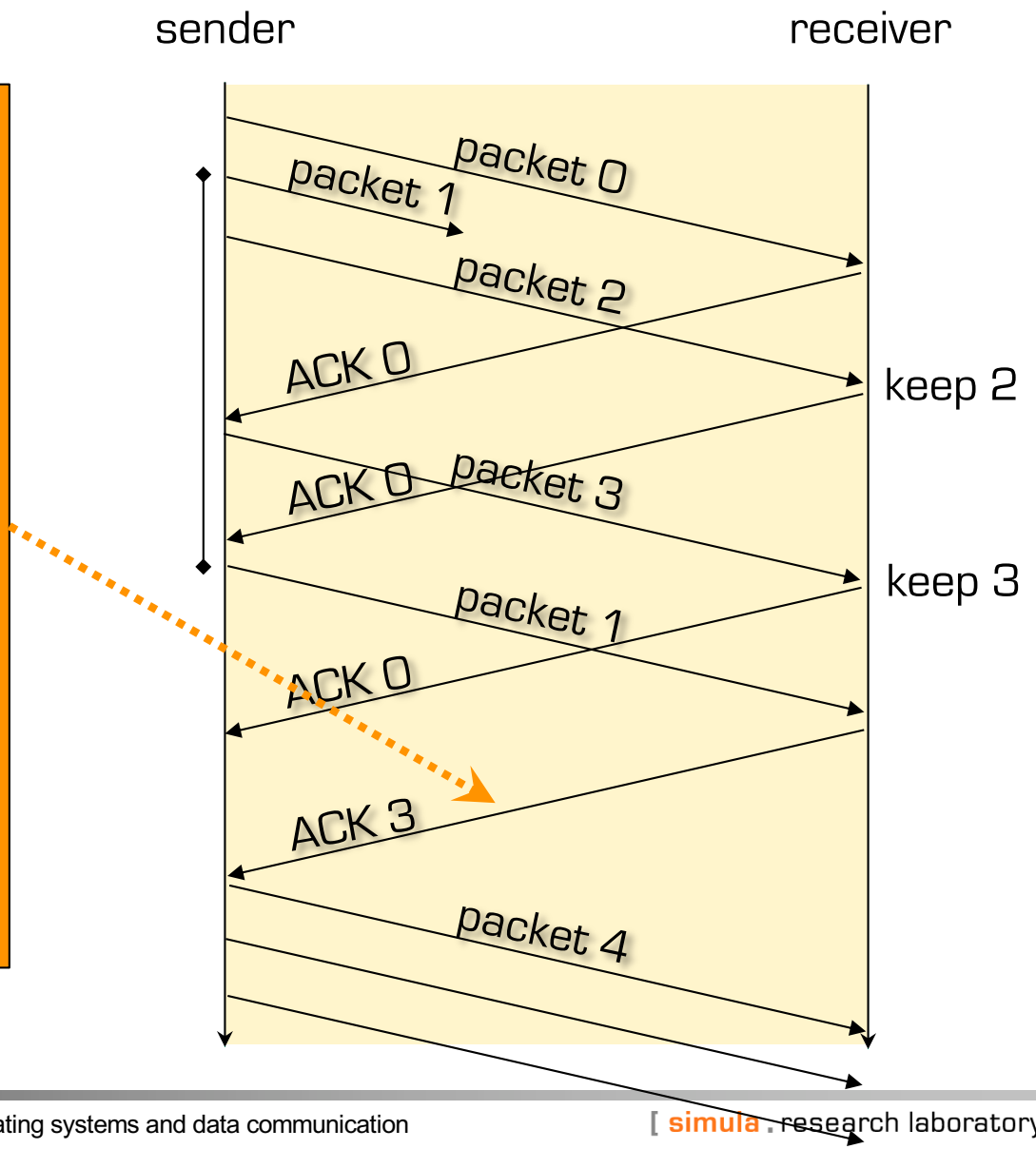
Retransmission approaches

This approach is called
Selective Repeat

Receiver buffers packets that arrive out of order but in window

Advantages:

- less traffic
- if the network usually loses only 1 packet in a row, this uses bandwidth better



IN2140:
Introduction to Operating Systems and Data Communication

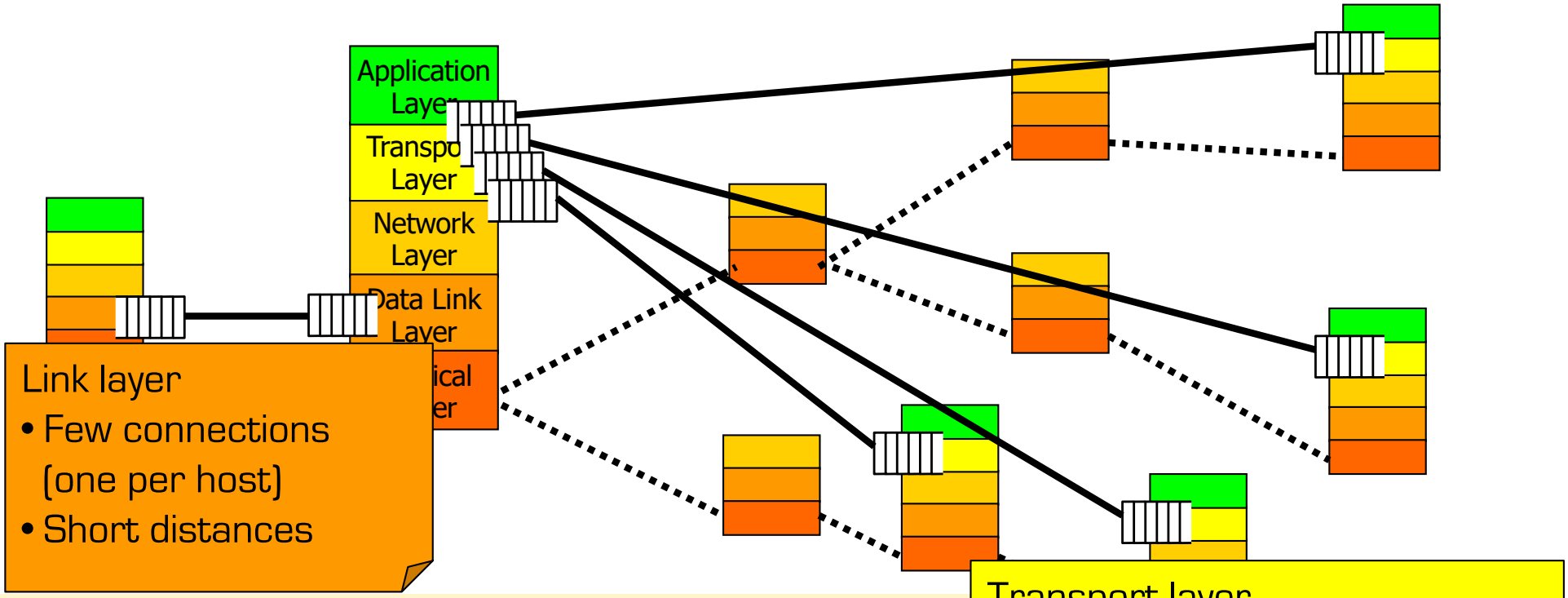


Transport layer

Flow Control

Flow Control on Transport Layer

Fast sender shall not flood slow receiver



Flow control mechanisms:

- Stop-and-wait
- Sliding window
  - Go-back-N / no buffer allocation
  - Selective repeat / static buffer allocation
  - Credit mechanism / dynamic buffer allocation

IN2140:
Introduction to Operating Systems and Data Communication

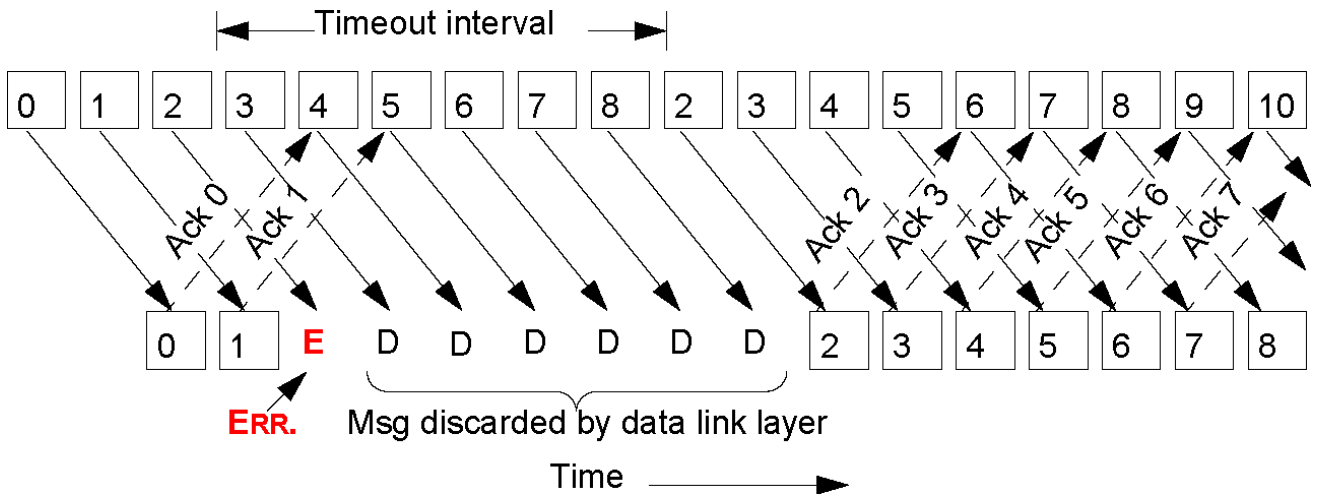


Transport layer

Flow Control:
sequence number space

Sliding Window: Go-Back-N

Example: sender: error detection by timeout



© Ralf Steinmetz, Technische Universität Darmstadt

Sliding Window: Go-Back-N

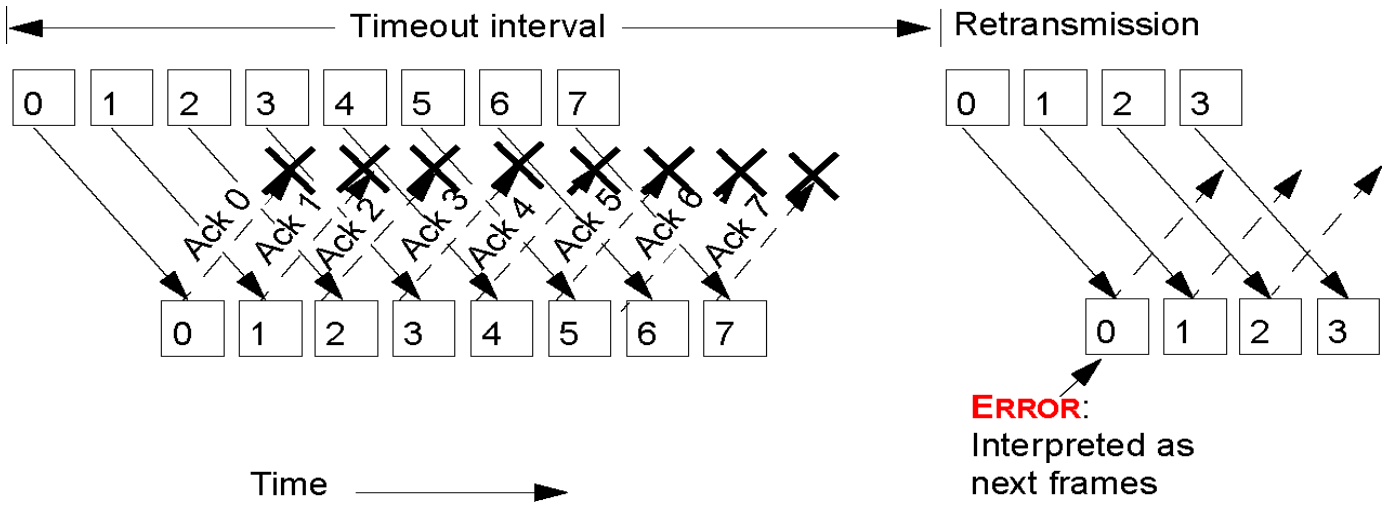
Correlation between

- window size and
- number of possible sequence numbers

→ at least *max. window size* ***strictly less than*** *range of sequence numbers*

Example for incorrect window size:

- amount of sequence numbers 8
- window size 8
- all ACKs lost



Sliding Window: Go-Back-N

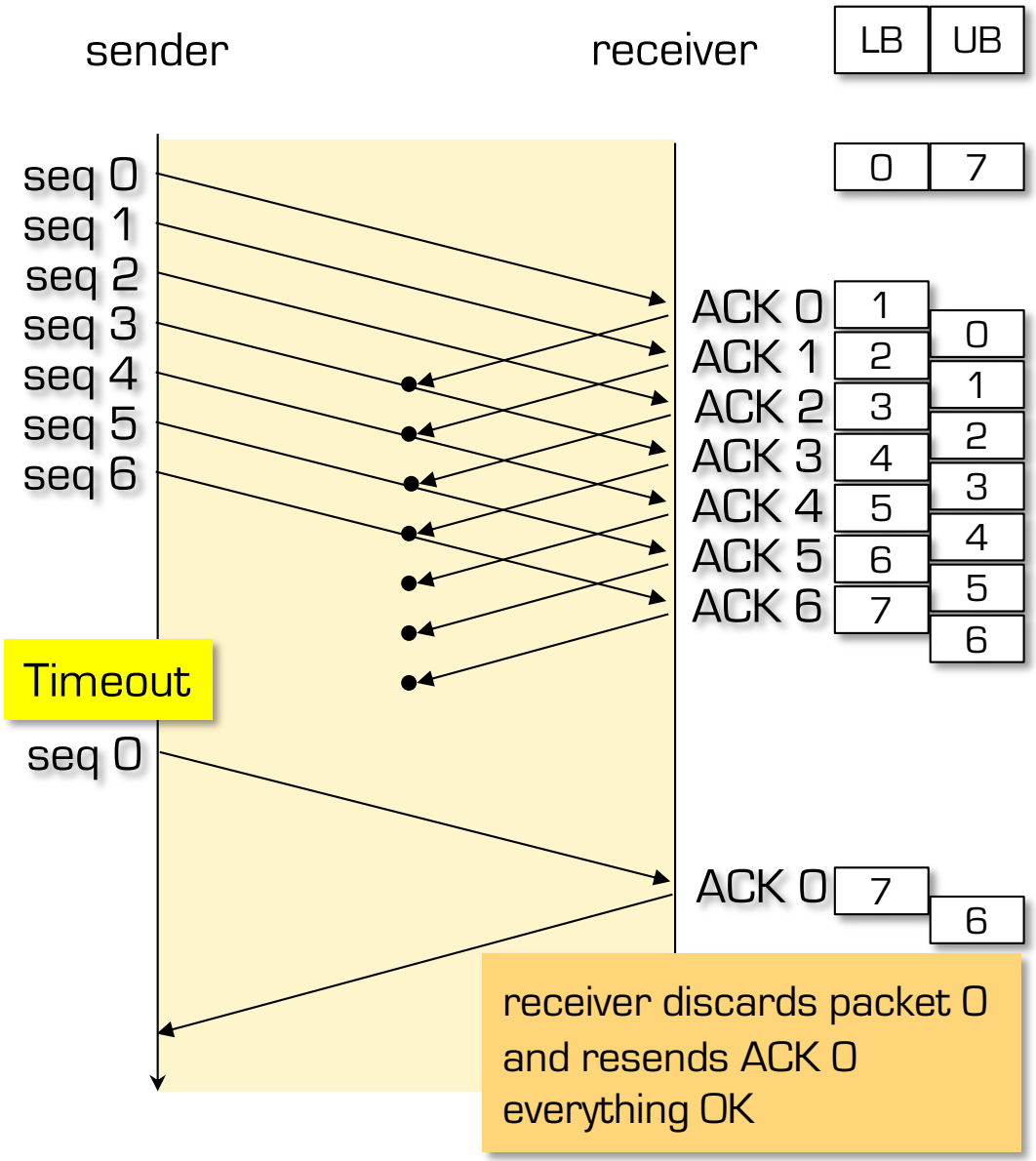
Correlation between

- window size and
- number of possible sequence numbers

→ at least *max. window size* ***strictly less than*** *range of sequence numbers*

Example for correct window size:

- amount of sequence numbers 8
- window size 7
- all ACKs lost



Sliding Window: Go-Back-N

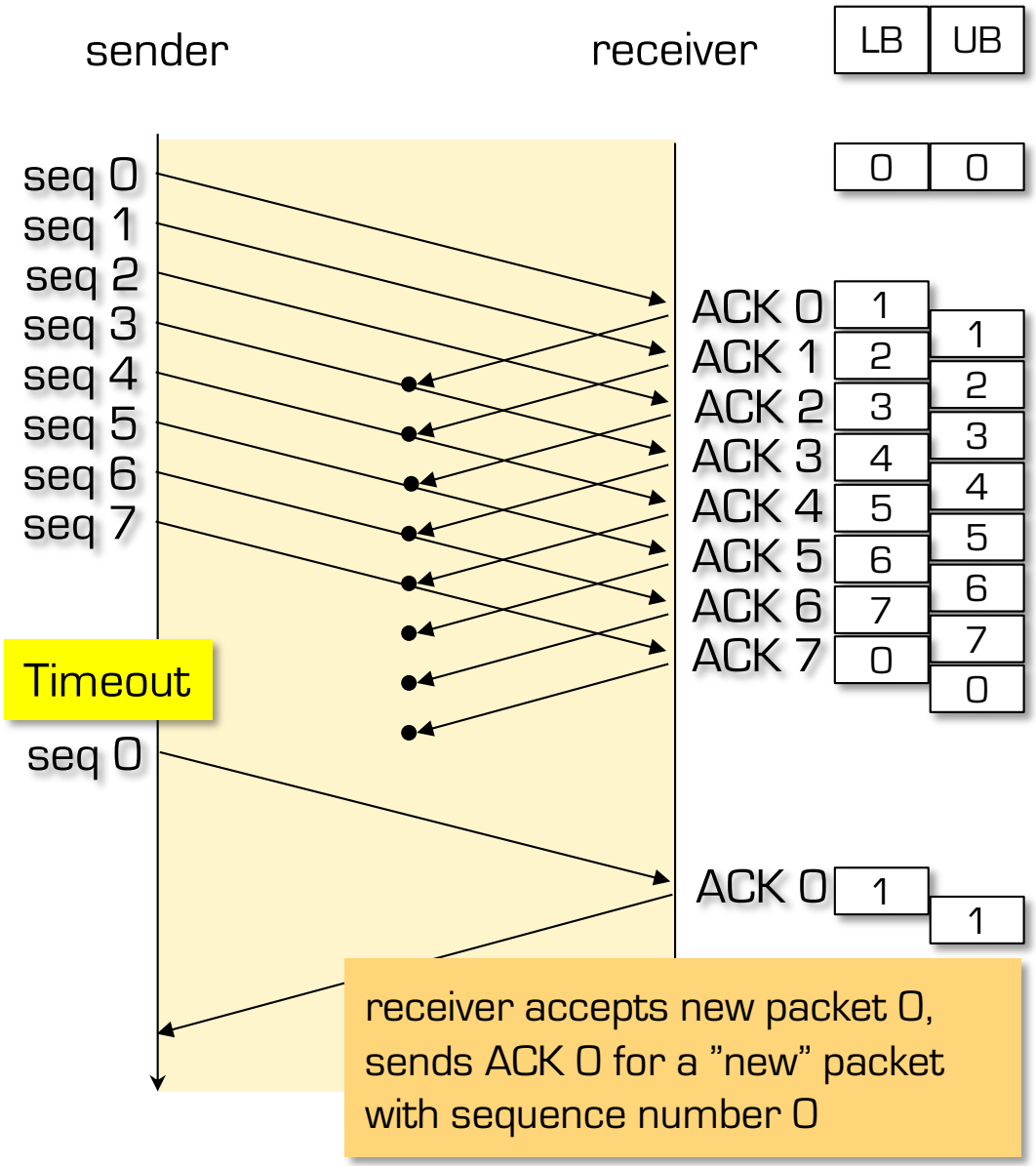
Correlation between

- window size and
- number of possible sequence numbers

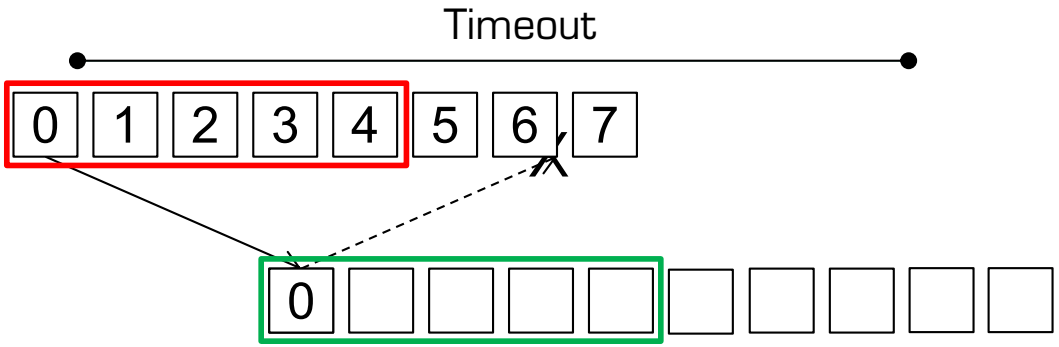
→ at least *max. window size* ***strictly less than*** *range of sequence numbers*

Example for incorrect window size:

- amount of sequence numbers 8
- window size 8
- all ACKs lost



Sliding Window: Selective Repeat



Correlation between

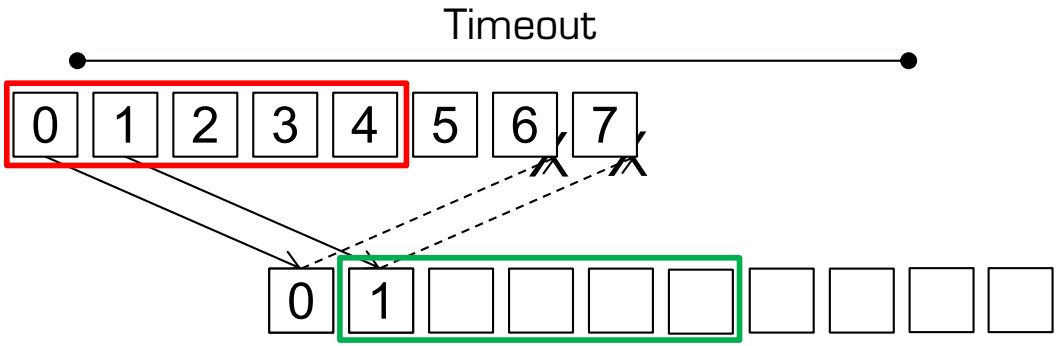
- window size and
- number of possible sequence numbers

→ max. window size <= 1/2 range of sequence numbers

Example for **incorrect** window size:

- amount of sequence numbers 8
- window size 5
- **all ACKs are lost**, and the packet that has been lost last is the first one to arrive at the receiver again

Sliding Window: Selective Repeat



Correlation between

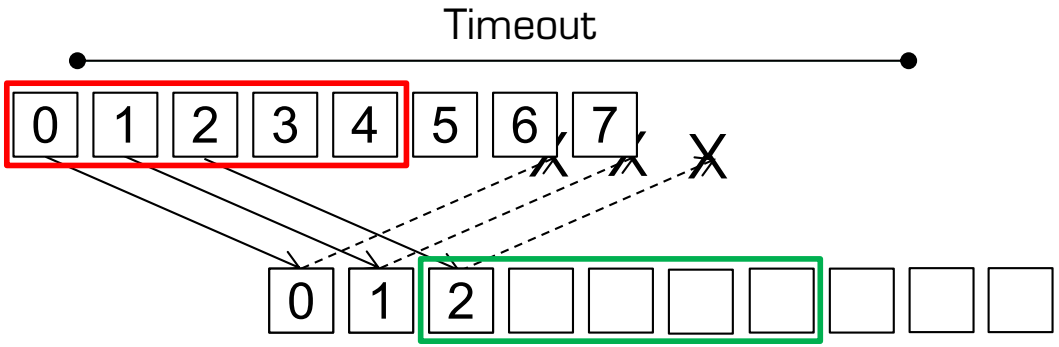
- window size and
- number of possible sequence numbers

→ max. window size <= 1/2 range of sequence numbers

Example for **incorrect** window size:

- amount of sequence numbers 8
- window size 5
- **all ACKs are lost**, and the packet that has been lost last is the first one to arrive at the receiver again

Sliding Window: Selective Repeat



Correlation between

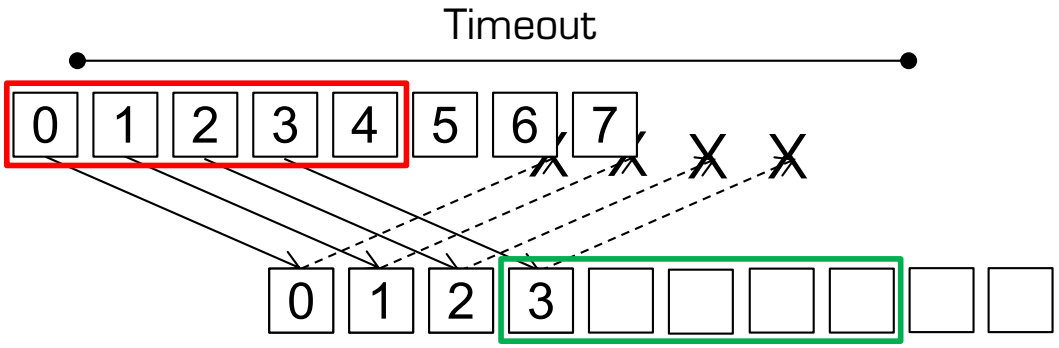
- window size and
- number of possible sequence numbers

→ max. window size <= 1/2 range of sequence numbers

Example for **incorrect** window size:

- amount of sequence numbers 8
- window size 5
- **all ACKs are lost**, and the packet that has been lost last is the first one to arrive at the receiver again

Sliding Window: Selective Repeat



Correlation between

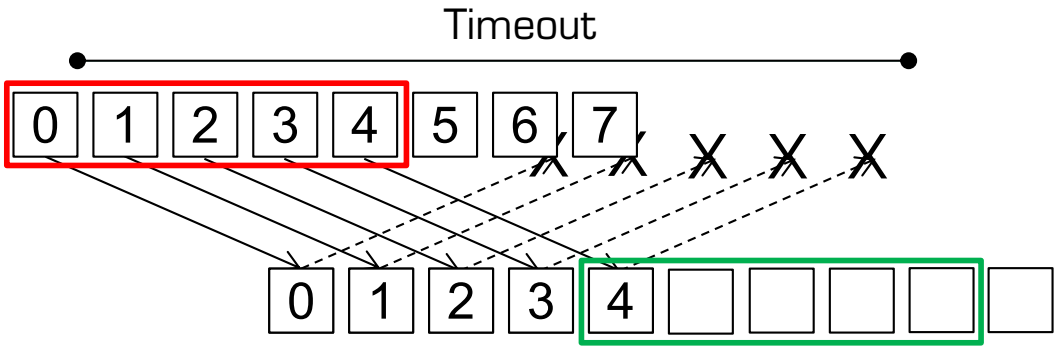
- window size and
- number of possible sequence numbers

→ max. window size <= 1/2 range of sequence numbers

Example for **incorrect** window size:

- amount of sequence numbers 8
- window size 5
- **all ACKs are lost**, and the packet that has been lost last is the first one to arrive at the receiver again

Sliding Window: Selective Repeat



no ACKs arrived at the sender
must wait for timeout

Correlation between

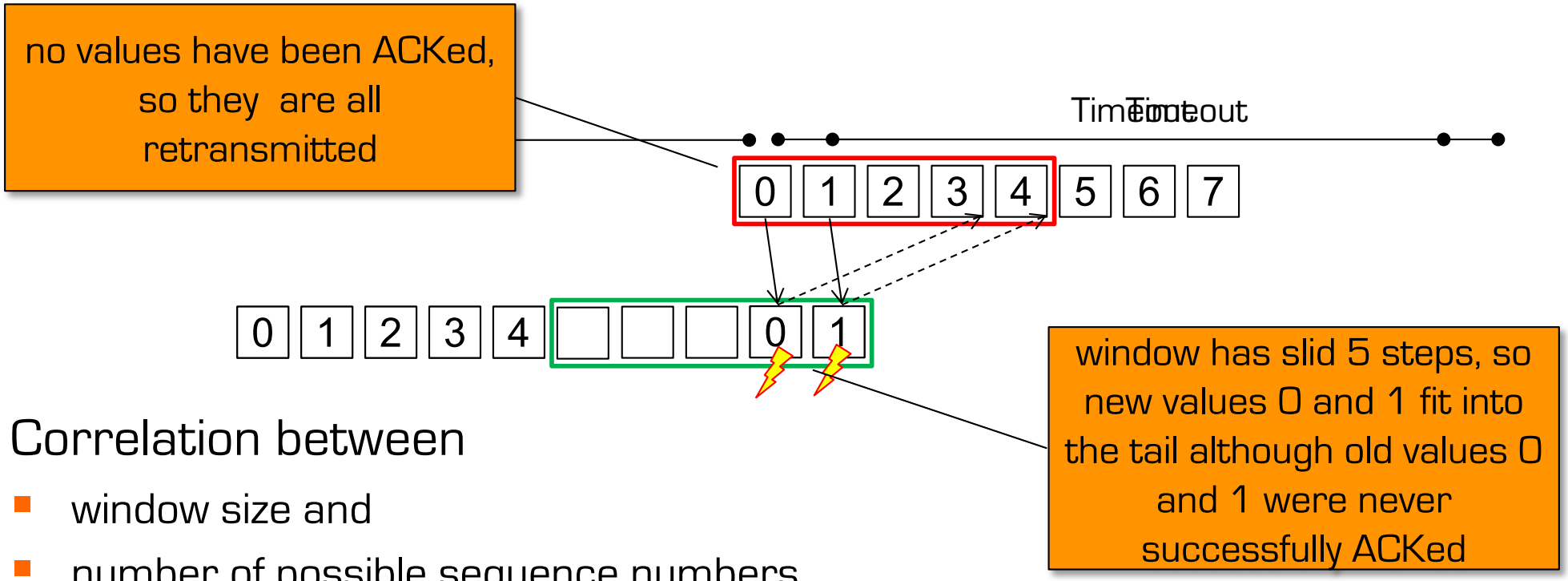
- window size and
- number of possible sequence numbers

→ max. window size <= 1/2 range of sequence numbers

Example for **incorrect** window size:

- amount of sequence numbers 8
- window size 5
- **all ACKs are lost**, and the packet that has been lost last is the first one to arrive at the receiver again

Sliding Window: Selective Repeat



Correlation between

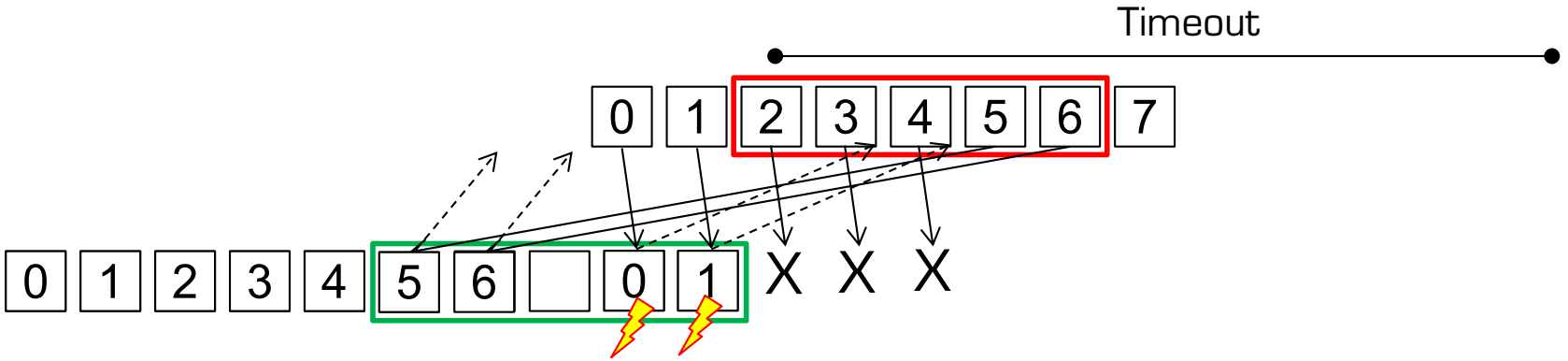
- window size and
- number of possible sequence numbers

→ max. window size <= 1/2 range of sequence numbers

Example for **incorrect** window size:

- amount of sequence numbers 8
- window size 5
- **all ACKs are lost**, and the packet that has been lost last is the first one to arrive at the receiver again

Sliding Window: Selective Repeat



Correlation between

- window size and
- number of possible sequence numbers

→ max. window size <= 1/2 range of sequence numbers

Example for **incorrect** window size:

- amount of sequence numbers 8
- window size 5
- **all ACKs are lost**, and the packet that has been lost last is the first one to arrive at the receiver again

Go-back-N vs Selective Repeat – a clear choice?

- Burst errors are rare
  - Selective Repeat repeats exactly the missing/bad packet
  - Go-back-N discards lots of safely arrived packets

- Burst errors are frequent
  - Selective Repeat waits for a timeout for every packet separately
  - Go-back-N recovers quickly after the first timeouts

- Error-free transmission
  - Selective Repeat senders must wait after sending ½ sequence number space packets
  - Go-back-N senders must wait after sending sequence number space-1 packets

IN2140:
Introduction to Operating Systems and Data Communication



Transport layer

Flow Control:
credit mechanism

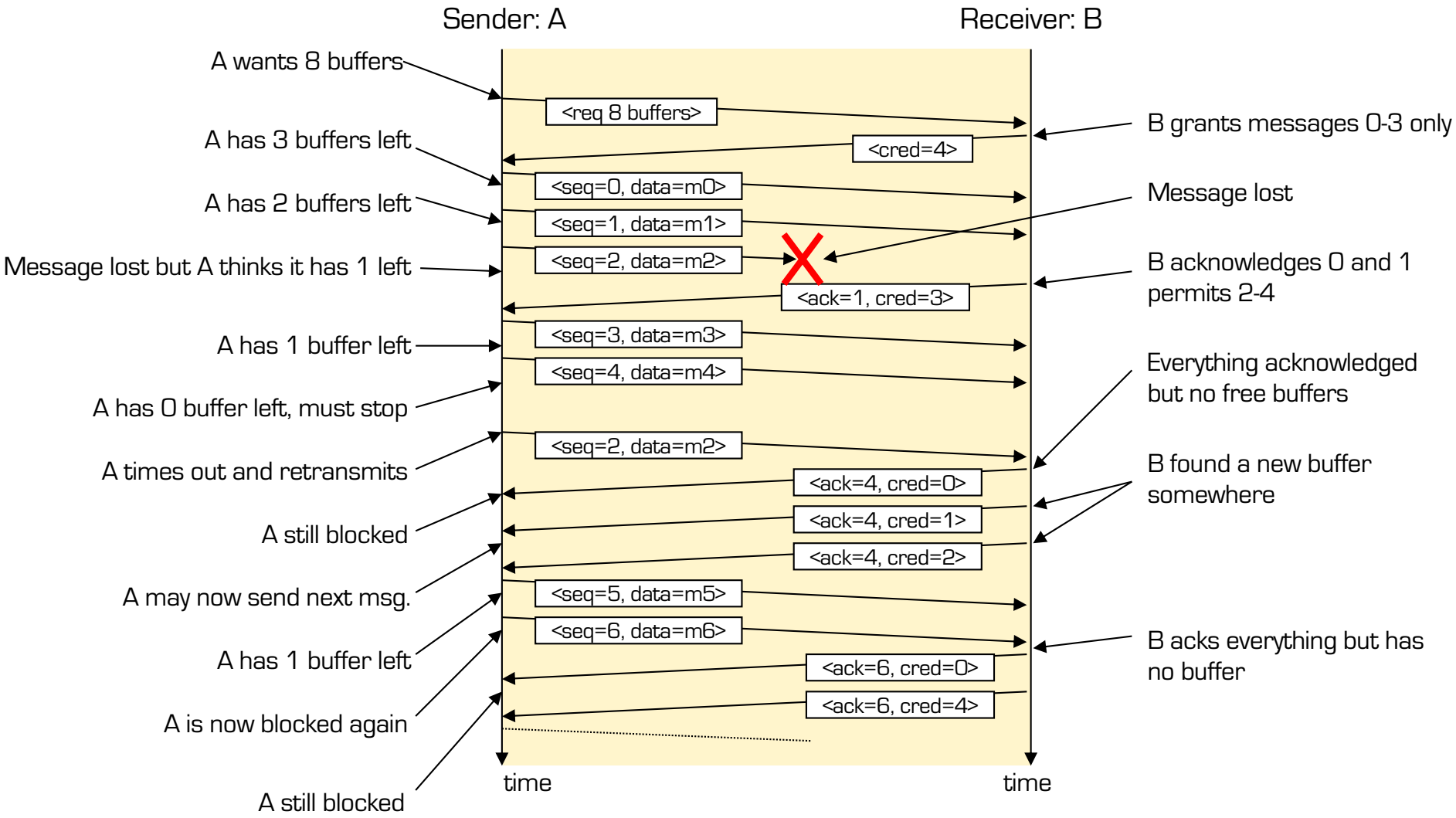
Credit Mechanism

- 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



IN2140:
Introduction to Operating Systems and Data Communication



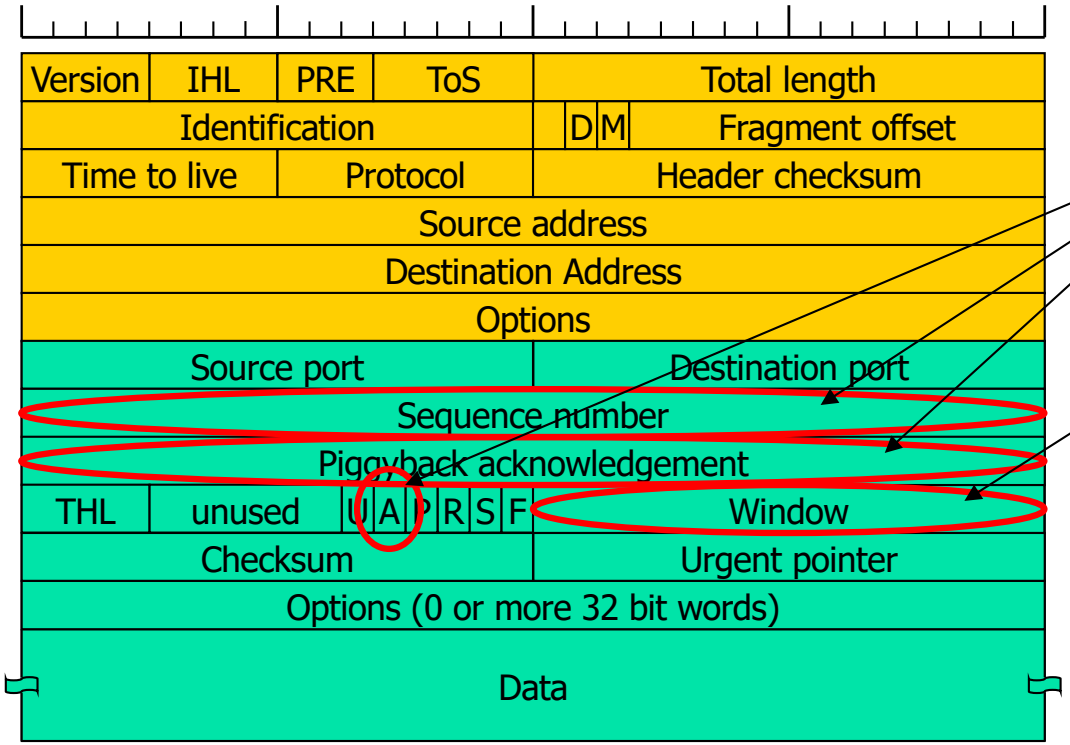
Transport layer

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
- `sequence number` up to 2^{32} *bytes*
- `window` up to 2^{16} *units*
  - *originally:* units = bytes
  - *today:* Window Scale Option, multiply `window` with scale factor 2^N bytes, with $0 \leq N \leq 14$

- 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

TCP Flow Control

