Computer Networks

Lecture on

Transport Control Protocol

Plan of This Lecture

- Main TCP features
- Header
- Connection establishment & termination
- Sliding window & retransmission
- State diagram
- Congestion control
- Timers

Main TCP Features

- Reliable transport ≠ reliable processing by terminals
- Connection-oriented
- Full duplex stream flows – TCP messages are called segments
- Congestion control
- Priority messaging
- Complicated

Application – reliable transfer of massive data

Transports majority of Internet traffic more than 90%

Common Methods for Data Structure Synchronisation

Where is the start and the end of the data structure in the byte stream?

1. End of line character

   Messages are human readable

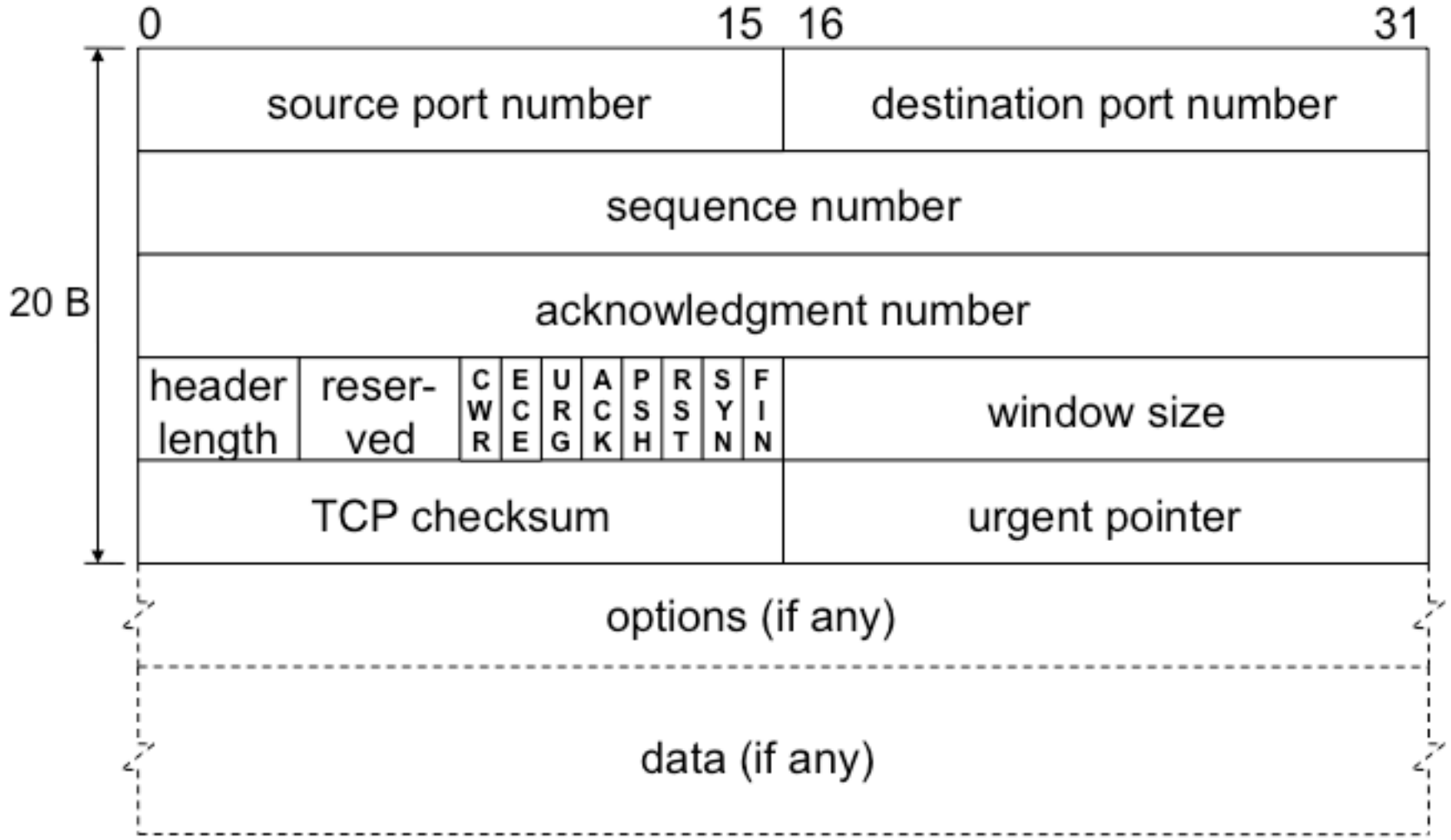
   Easy to program, debug, analyse

2. <Length, Type, Value> or <Type, Length, Value> tuples

   A magic number can by added at the end or at the beginning

   It helps to discover programmer errors

Header


0
15
16
31
source port number
destination port number
sequence number
acknowledgment number
20 B
header length
reser-ved
CWR
ECE
URG
ACK
PSH
RST
SYN
FIN
window size
TCP checksum
urgent pointer
options (if any)
data (if any)

Port number – points a process on the terminal device

Sequence number – indexes bytes in a data stream

Acknowledgement number – index of the first expected byte

Checksum – of all bytes of the TCP segment and IP meta-header (IP addresses)

- Calculated in ones' complement numeric representation
  (value 0 has two representations: all 1s and all 0s)

Flags

- CWR – Congestion Window Reduced
- ECE – Explicit Congestion Notification Echo
- URG – urgent data in the segment
- ACK – acknowledgement number is carried
- PSH – push the data in the stream from the buffers to be sent
- RST – reset the connection
- SYN – start / synchronize the connection
- FIN – finish the connection

Window size – what is the size of reception window? / how many bytes can be sent without ACK?

Urgent pointer – Where the urgent data starts in the segment

Network hardware interfaces can support TCP, e.g.

- checksum offloading
- segmentation offloading input & output

TCP Options

Options – allow TCP to evolve backward and forward compatible
are encoded by using a Type Length Value format

Some of them are negotiated during connection establishment (SYNs)
SYN segment proposes ACK segment accepts or ignores

- Maximum Segment Size negotiated during SYNs
  - Transmission efficiency ↑ if MSS == max transfer unit of the path

- Selective Acknowledgements negotiated during SYNs
  - Transmission efficiency ↑ if there are radio links on the path

- Fast Open Cookie
  - Authenticate server or client
  - Used for secure data transfer during SYNs

TCP Extensions for High Performance RFC 1323

- TCP Window Scale Option
  - header contains *snd.wnd >> S*
  - *S* is the scaling factor ($0 \le S \le 14$) negotiated during SYNs

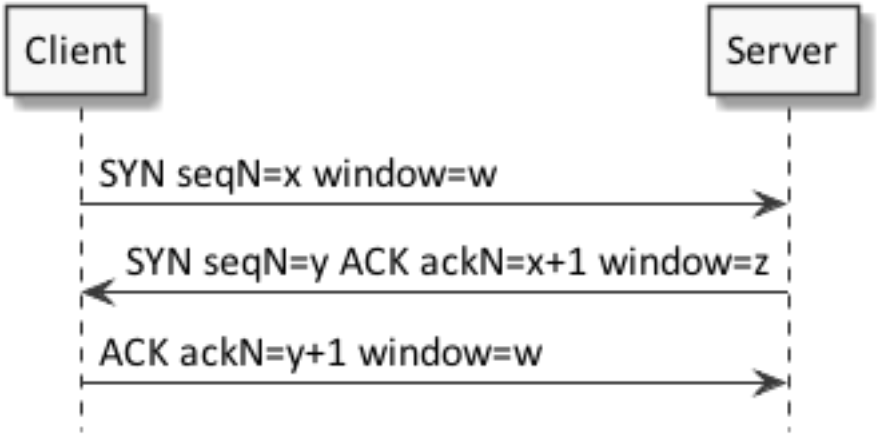
	no Window Scale Option	with Window Scale Option
max receive buffer	2^{16} i.e. 64 KB	2^{30} i.e. 1GB
max throughput for 1 ms RTT	524 Mbps	8590 Gbps
max throughput for 500 ms RTT	1.05 Mbps	17 Gbps

- Round-Trip Time Measurement – extensions to support timestamps
  - to be used with large windows

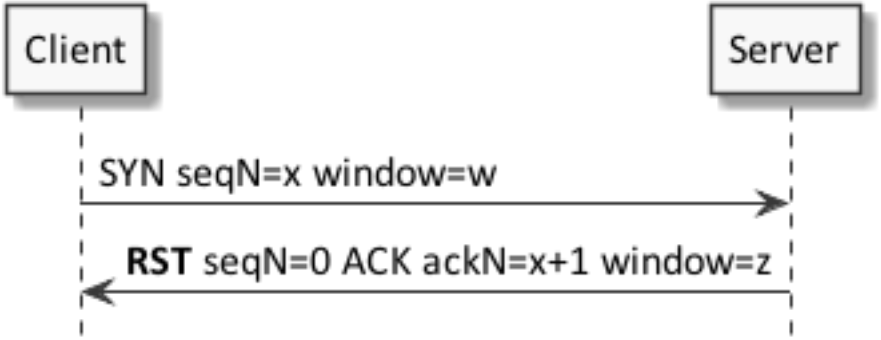
- Protect Against Wrapped Sequence Numbers – adding a delayed duplicate packet
  - to be used with large windows
  - otherwise it limits max throughput – SN cannot wrap before 2 * *Maximum Segment Lifetime*
    MSL is assumed to be 2 min RFC 793

Connection Establishment

Successful connection



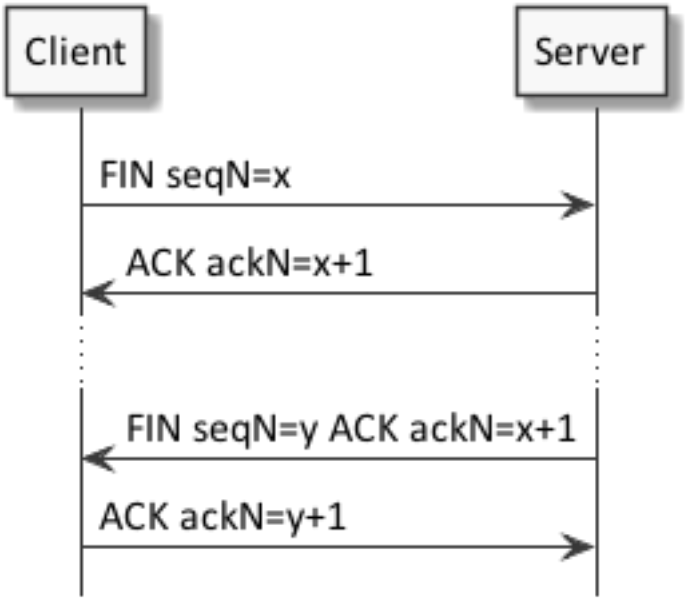
Connection refused



seqN – random initial sequence number
ackN – number of the awaited byte
SYN can be retransmited

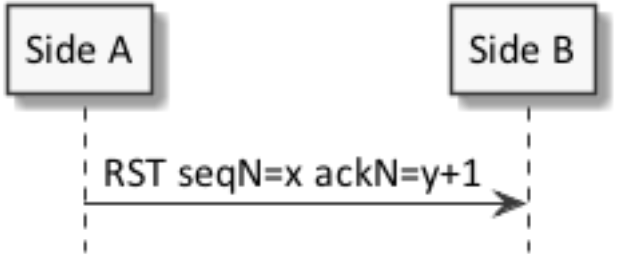
Connection Termination

Graceful termination of both streams



FIN can be retransmitted

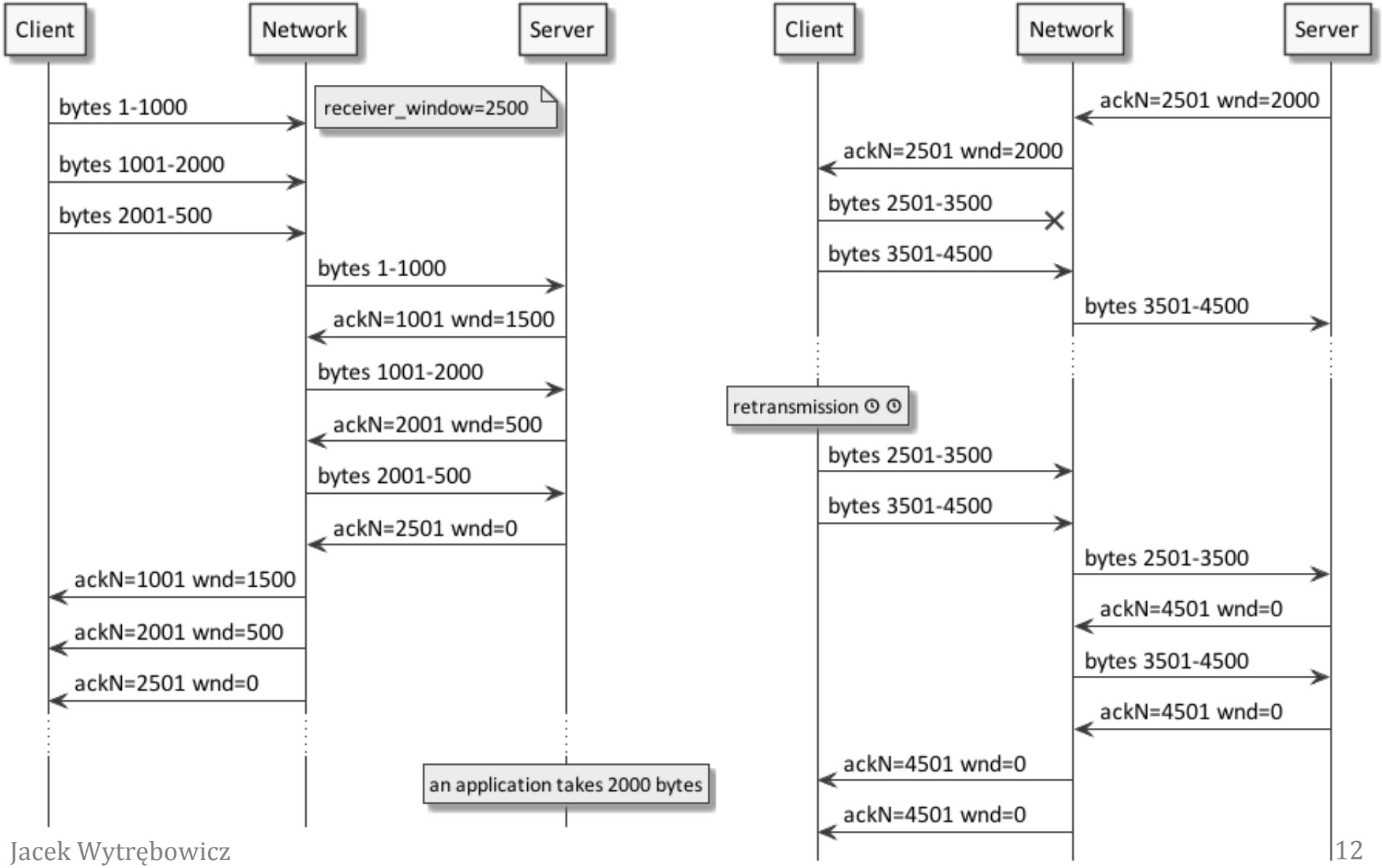
Abrupt termination



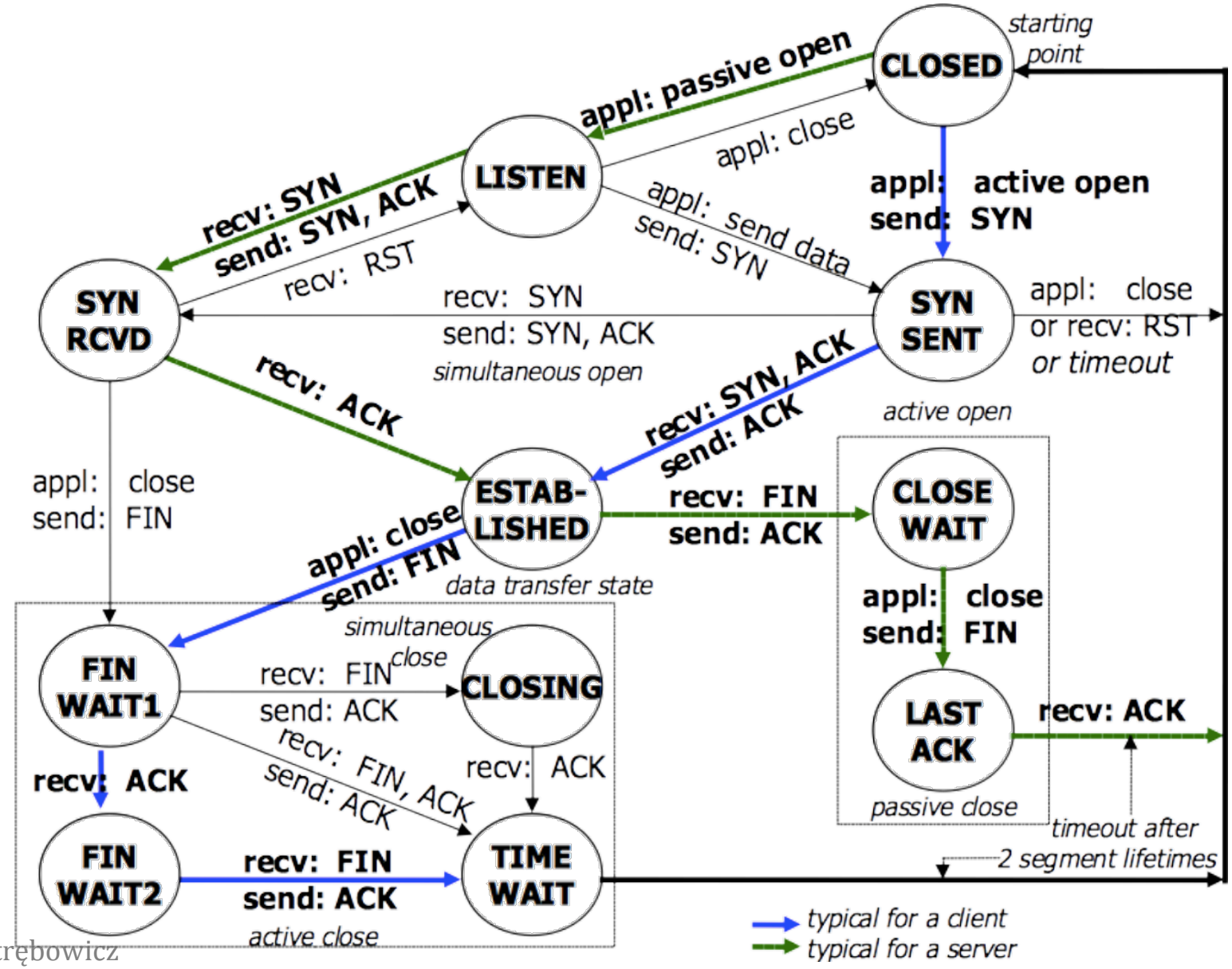
RST is neither acknowledged nor retransmitted

If one side closes a connection , the other side can still transmit data

Sliding Window & Retransmission



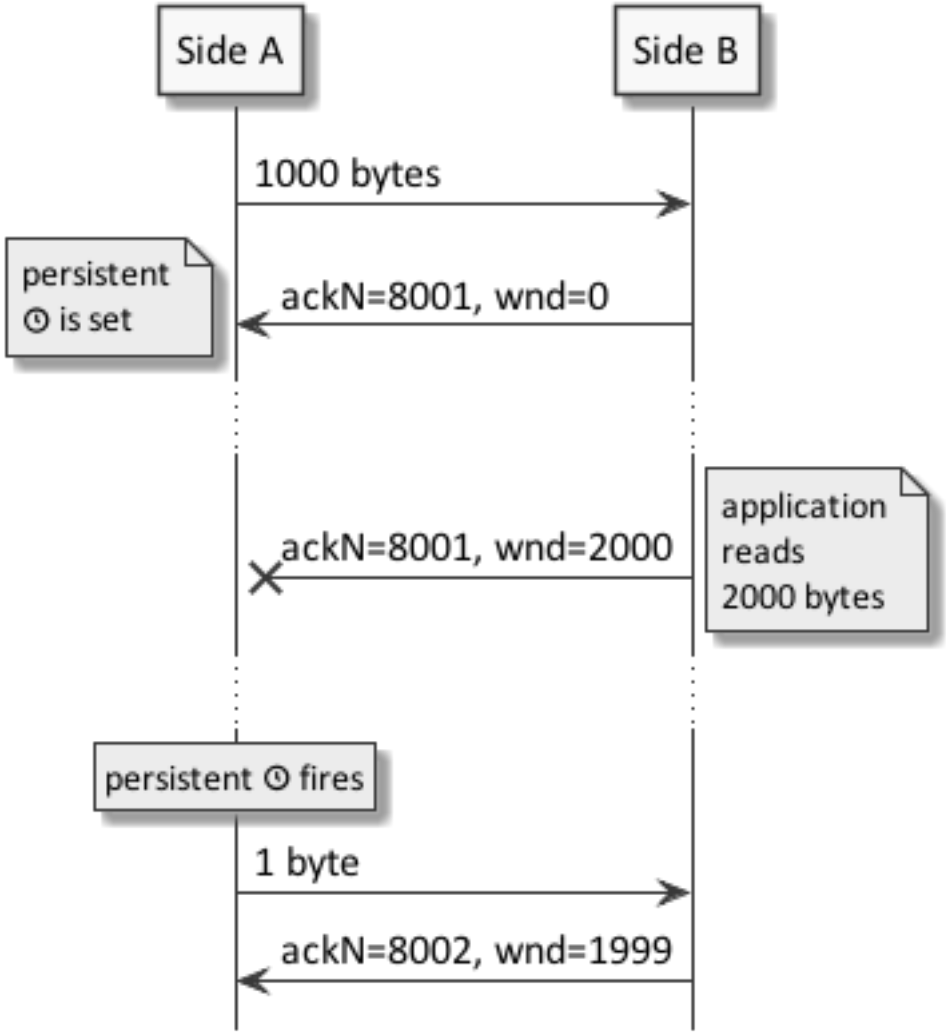
State Diagram



Timers

- Retransmission timer
  - Set for every segment sent
- Silence after connection timer TIMEWAIT 2 * TTL ~ 30s
  - To eliminate the risk of adding a delayed segment from previous connection
  - Slowdowns opening subsequent connection between the same peers!
    - Do not close & open for pipeline processing
  - Makes problems on busy TCP servers – freeing memory is delayed
- Persistent timer
  - Set when window size changes to 0
  - If it fires TCP tries to send 1 byte of data
  - Protect the connection against a deadlock
- Live & inactivity timers
  - Allow for discovery of connection loss – while no data is transmitted
  - Protect against half-open connection – while other side is shut off

Persistent Timer



Retransmission Timeout

If expires too early ⇒ bandwidth is wasted
by retransmitting segments that have been received

If expires too late ⇒ application suffers from delays
sender is idle waiting for the expiration of its retransmission timeout

- Timestamp of each data segment – inefficient for small data segments
- Round-trip-time (RTT) is to measure
  the delay between the transmission of a data segment and the reception of its ACK
  - RTT differs between TCP connections
  - RTT may change during the lifetime of a single connection
    - Adaptive algorithm

      $RTT = \alpha * OLD\_RTT + (1-\alpha) * NEW\_SAMPLE$

- Timeout value
  - Initial (after SYN) 3 seconds
  - Set for every sent segment
  - $TIMEOUT = \beta * RTT$ suggested $\beta = 2$
    - several optimizations exist

Retransmission Clock – Problems

- RTT counting is difficult
  - data segment can be retransmitted
  - ACK can be lost
  - ACK can be cumulative
    - no direct correspondence between sent segment and ACK
    - ACK number can differs from sent segments number
  - RTT measurement (timestamps) is effective for large windows – TCP option

- How to count timeout after retransmission?
  - Ignoring samples after a retransmission is insufficient!
  - NEW_TIMEOUT = γ * TIMEOUT
    - suggested γ = 2 aka. exponential backoff
      or a table for subsequent retransmissions
  - After ACK use RTT: TIMEOUT = β * RTT
  - Max timeout = 60 s next connection is closed

ACK & Retransmission Strategies

- Delayed ACK
- Fast retransmit
- Selective repeat

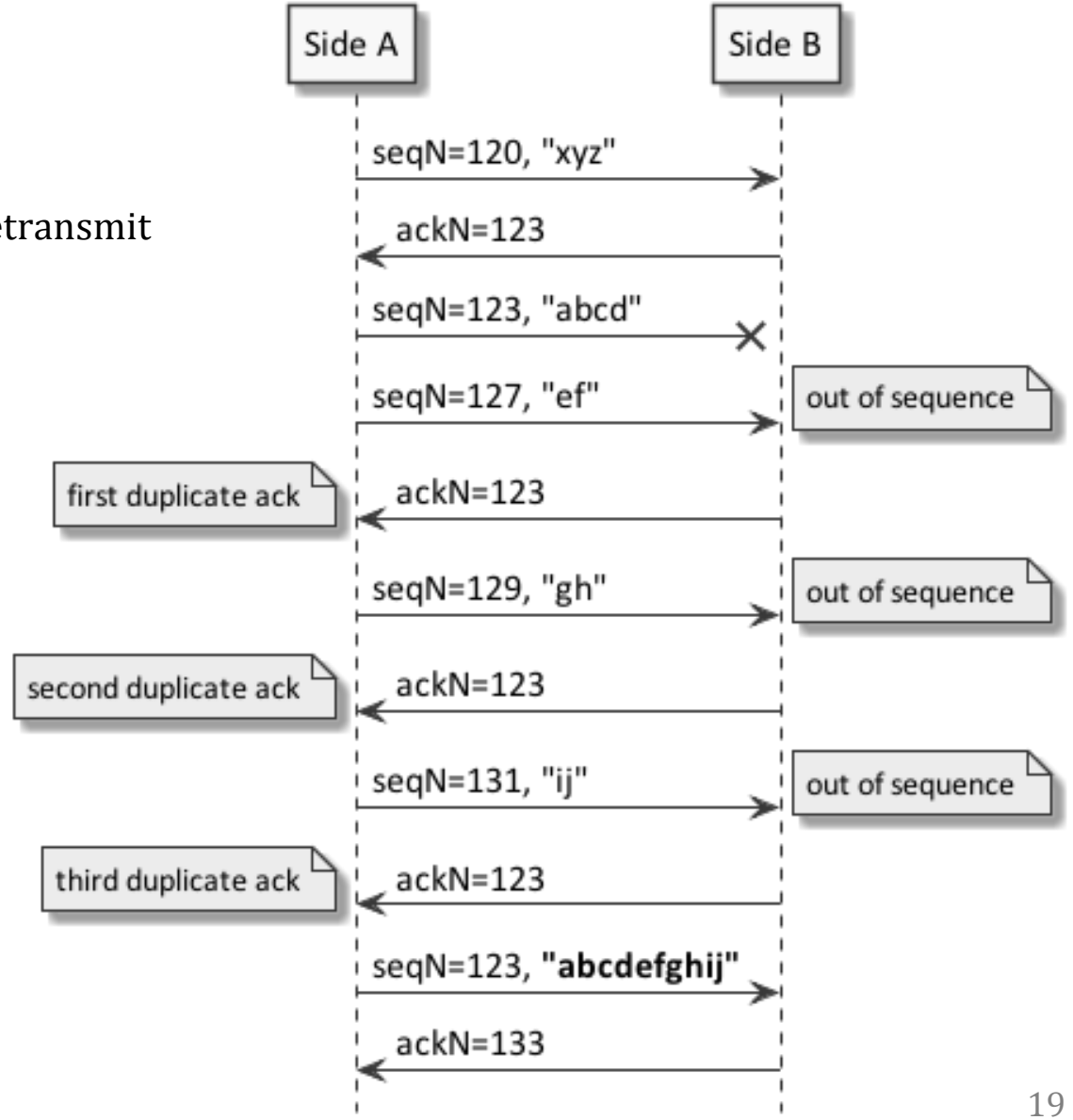
Delayed ACK strategy

- Piggybacking is used whenever possible
- ACK every 1 s – if no losses ⇒ less packets in the network
- ACK immediately – there are losses or reordering

Fast Retransmit

- Isolated segment loss often occurs
- Duplicate ACKs signal them
- Third duplicate ACK triggers fast retransmit
  - no waiting for the timeout

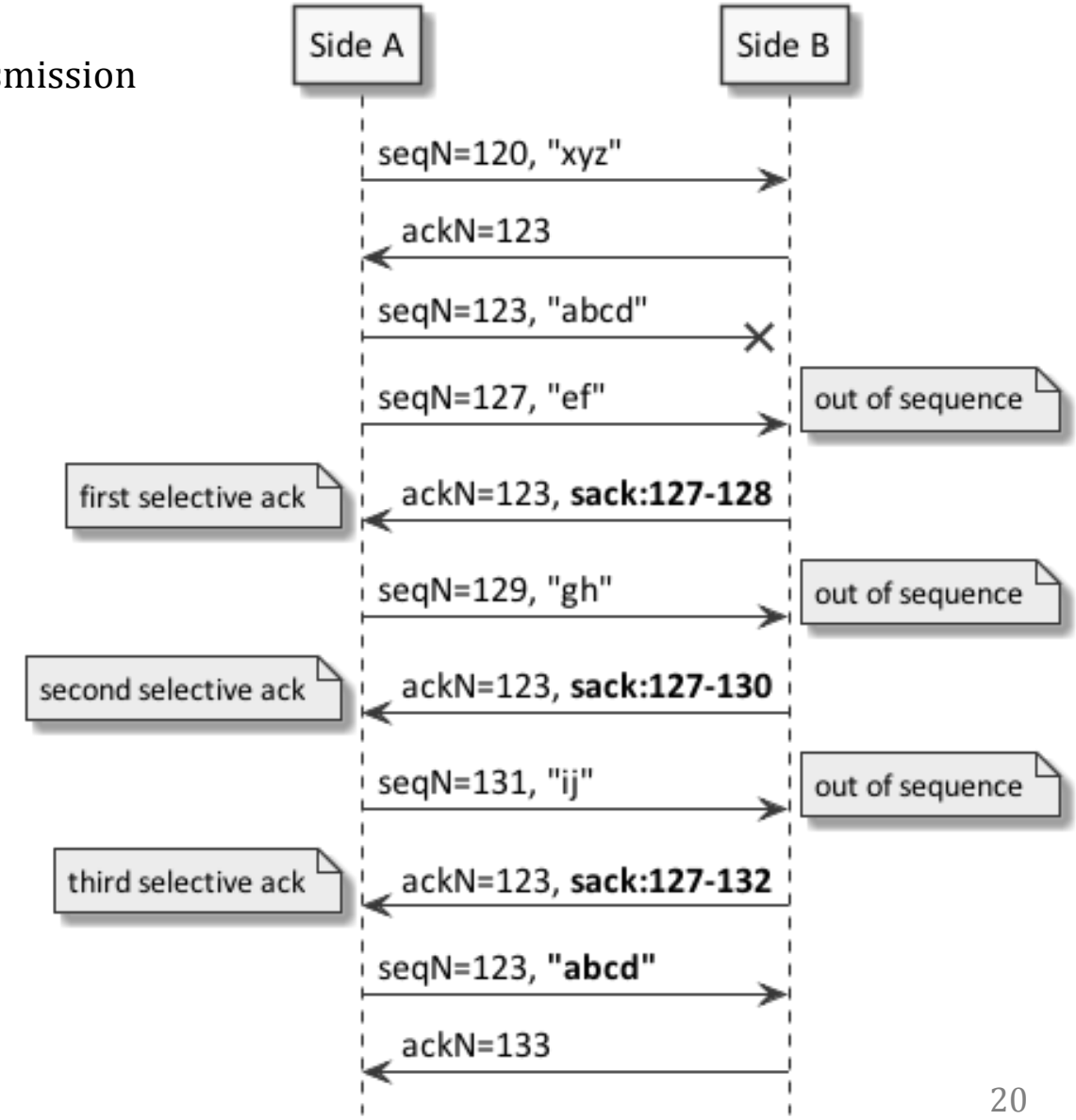
- Inefficient if
  - losses are not isolated
  - or the windows are small



Selective Repeat

- Third duplicate SACK triggers retransmission
  - no waiting for the timeout
- Efficient if
  - losses are not isolated
  - and the windows are small

Recommended for radio links



Congestion Control

Aim – avoiding link saturation
and thus packet drops and unnecessary retransmissions

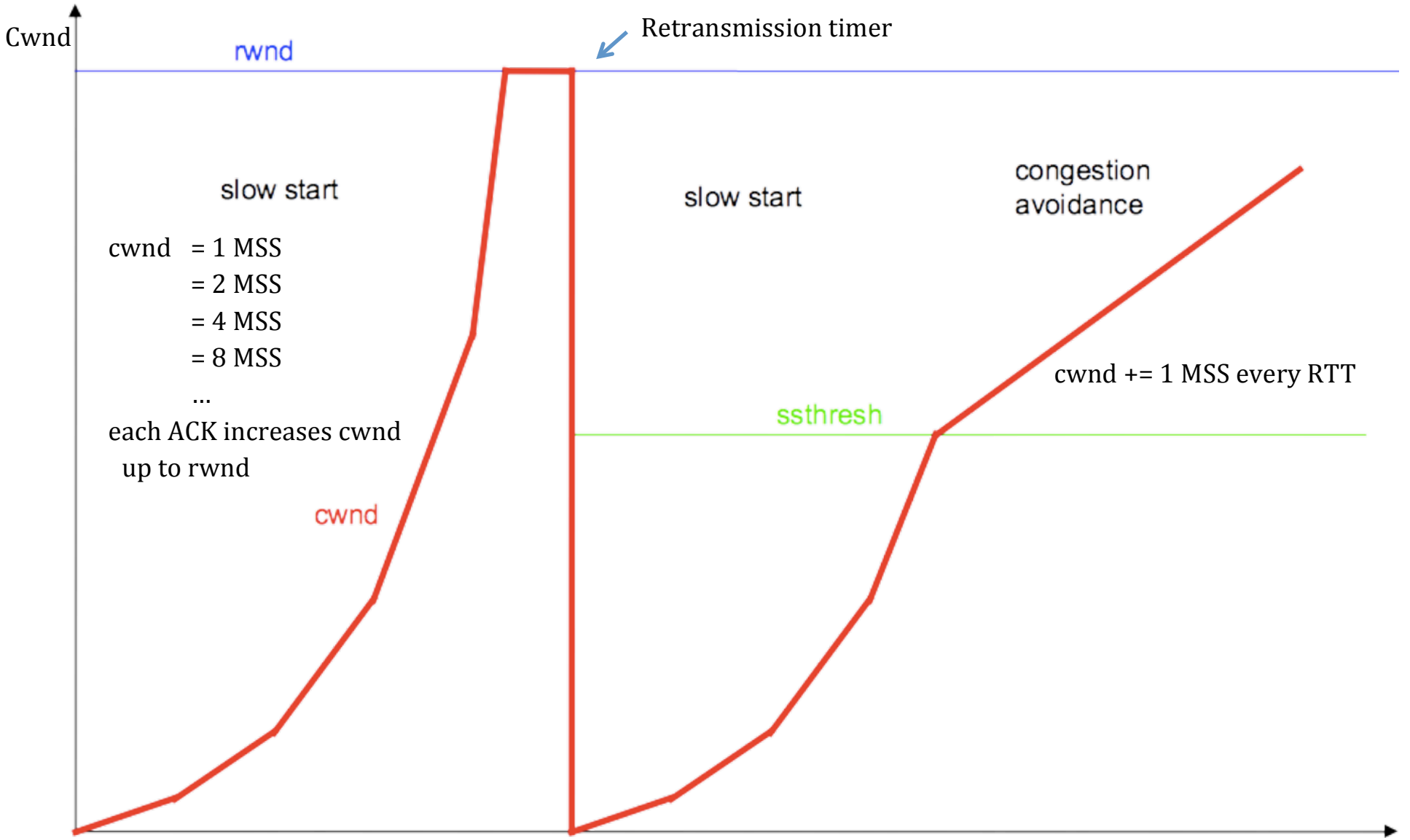
A segment loss ⇒ an indication of congestion

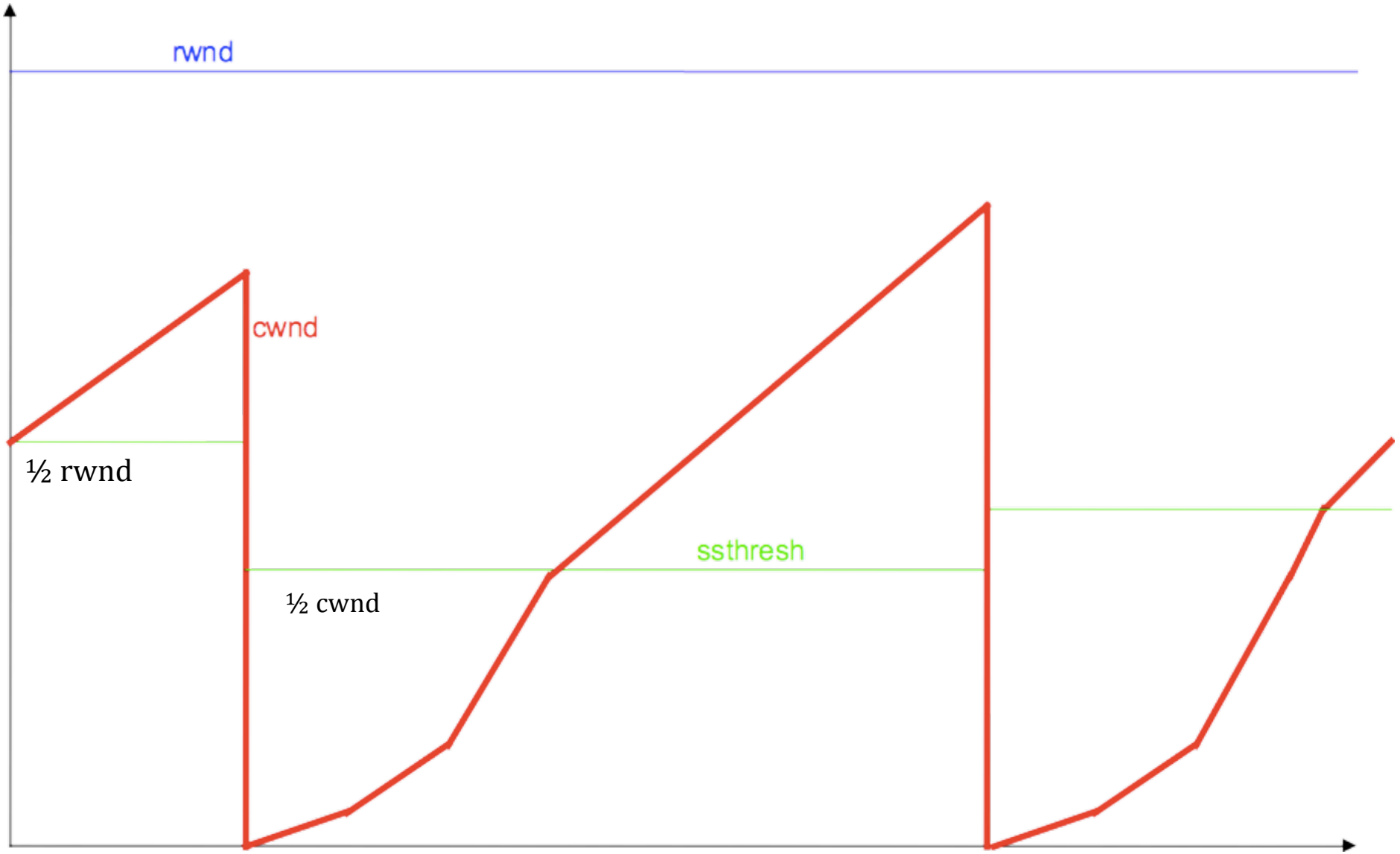
- 2 phases:
  - slow-start
  - congestion avoidance

- Additional features
  - fast retransmit
  - fast recovery

TCP cannot send data faster than window/RTT
- Congestion window (cwnd) controls sending data rate

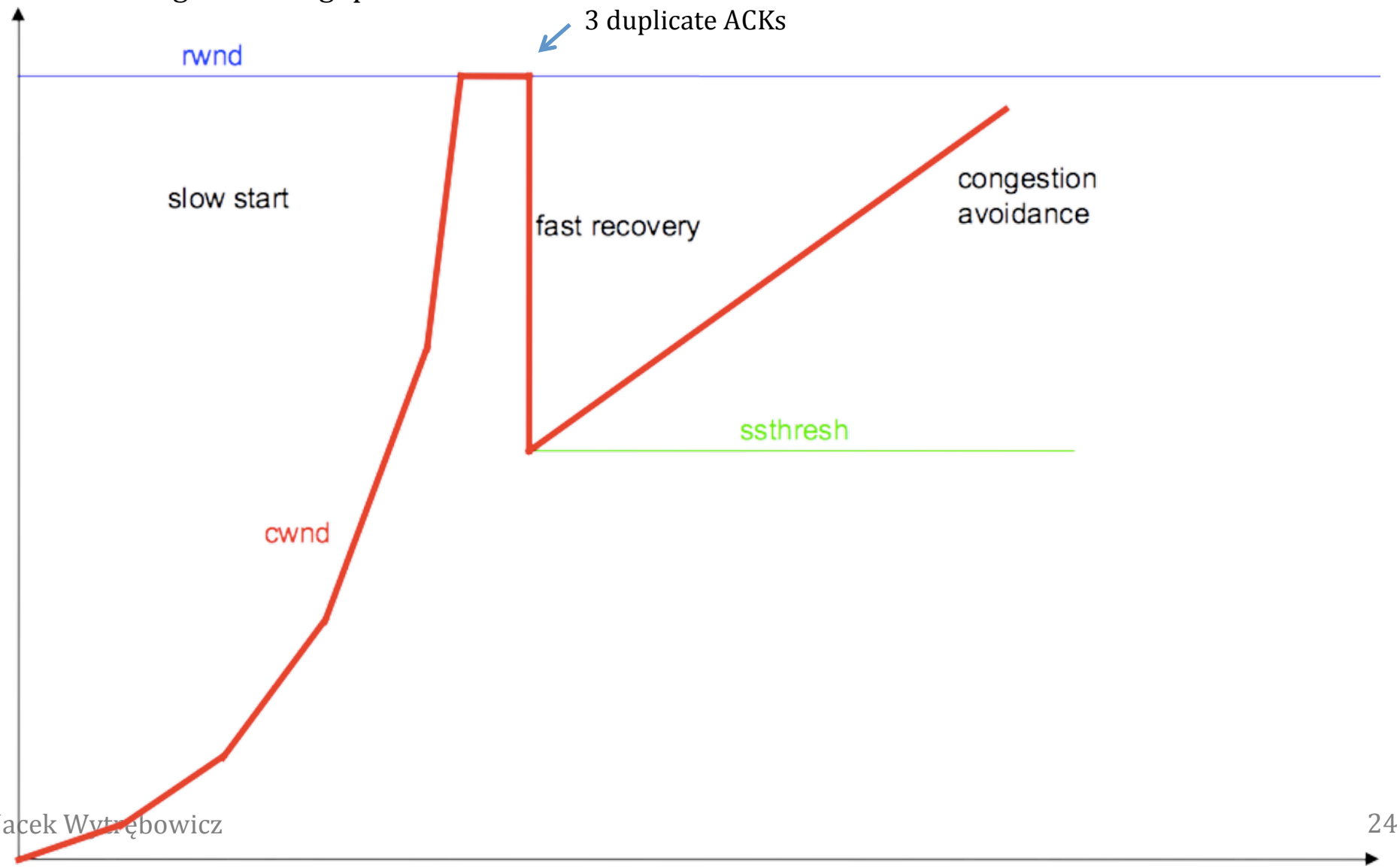
Slow Start & Congestion Avoidance




rwnd
cwnd
½ rwnd
ssthresh
½ cwnd

Fast Recovery

Gives higher throughput



Explicit Congestion Notification

- Extensions in IP and TCP headers (RFC3168, RFC4774)
- IP: two less significant bits of the DiffServ field:
  - 00: Non ECN-Capable Transport
  - 10: ECN Capable Transport
  - 01: ECN Capable Transport
  - 11: Congestion Encountered
- TCP: two reserved bits:
  - ECN-Echo
  - Congestion Window Reduced

Evolution of Congestion Control

A 100 different variants of the TCP congestion control scheme were proposed

The most popular & implemented in popular OSs

- NewReno
- TCPVegas
- CUBIC
- Compound TCP
- TCP BBR

Some OSs (e.g. Linux) allow to select one of them

New TCP congestion control schemes will continue to appear

Summary

- Main TCP features
- Methods for data structure synchronisation
- Header
  - TCP options
- Connection establishment & termination
- Sliding window & retransmission
- State diagram
- Timers
  - Retransmission clock – problems
- ACK & retransmission strategies
  - Fast retransmit
  - Selective repeat
- Congestion control
  - Slow start& congestion avoidance
    - Fast recovery
  - Explicit congestion notification
  - Evolution of congestion control

Questions

1. What are main features of TCP protocol?
2. When TCP is a good choice for an application?
3. Can we use TCP for multicast transfer and why?
4. Can an application logically close TCP connection in one direction?
5. The sequence number in TCP, are they used for byte numbering or for packet numbering? What is the reason?
6. What for is the “sequence number” field of TCP PDU?
7. What for is the “acknowledgement number” field of TCP PDU?
8. What for is the “window size” field of TCP PDU?
9. What for is the “urgent pointer” field of TCP PDU?
10. What for is the “header length” field of TCP PDU?
11. What are the flag bits in the TCP PDU?
12. Can we use PUSH flag in TCP connection to synchronize data structures?
13. How does Explicit Congestion Notification work on IP networks?
14. How does TCP avoid network congestions?
15. Explain why 3-way handshake must be applied in TCP. Why 2-way handshake is insufficient?
16. What does happen when the acknowledgement of the FIN message is lost at the end of TCP connection?

17. Draw message sequence chart while the two communication sides simultaneously send SYN.
18. How does TCP set the retransmission timer? How is calculated the timer value?
19. Why do we need the persistent timer?
20. What is the purpose of the TCP live / inactivity timers?
21. Why do we need the silence-after-connection timer?
22. Why are there active and passive openings of TCP connection?
23. List the names of TCP states the automaton go through during active open.
24. List the names of TCP states the automaton go through during passive open.
25. Why TCP allows for delayed acknowledgement?
26. Explain the principle of the slow start and congestion avoidance mechanisms.
27. Explain the principle of the fast retransmit and recovery mechanisms.
28. Consider a transport that supports window of one hundred 1250 bytes segments. What is the maximum bandwidth that this protocol can achieve if the round-trip-time is set to one second?