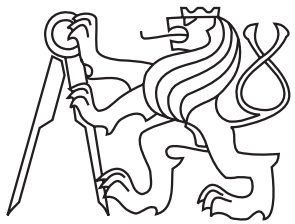
Transport layer. TCP, UDP.

Ing. Yelena Trofimova

Department of Computer Systems
Faculty of Information Technology
Czech Technical University in Prague

Computer networks, BIE-PSI
SS 2020/21, Lecture 6

`https://courses.fit.cvut.cz/BIE-PSI/`



Content

- transport layer
  - layer services
  - protocols
  - establishment and termination of connection
  - flow control, congestion control
- TCP
- UDP
- RTP, RTCP

Transport layer services

- transport layer is a boundary between
  - applications (software)
    and
  - networks (technology)
- ensures transparent data transfer between end-users
  - reliability
  - flow control
  - data segmentation
  - correction of errors

Addressing

- service must have an address
  - TSAP – Transport Service Access Point
- predefined TSAPs ("well-known")
  - eg. port 80 for http
- dynamically allocated – needs special software
  - portmapper
    - registers TSAP for services
    - eg. BitTorrent

Connection establishment I

- trivial solution:

  CONNECTION REQUEST $- - - >$

  $< - - -$ CONNECTION ACCEPTED

  problem: duplicated CR packet
- improvement:
  - unique identifier for each session
  - limited lifetime of packet

Connection establishment II

- three-way handshake:

CONNECTION REQUEST $- - - >$
(seq=x)

$< - - -$ ACKNOWLEDGE
(seq=y, ack=x)

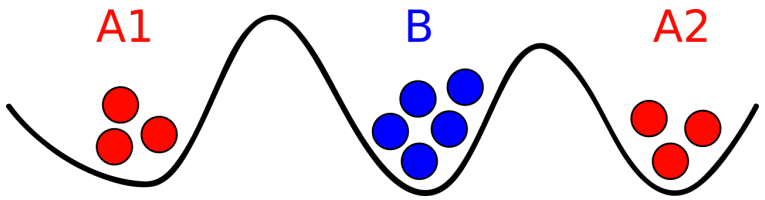
DATA $- - - >$
(seq=x+1, ack=y)

Connection termination

- asymmetric termination
  - one user terminates the connection (example: landline phone call)
  - possible data loss when the other user sent data before receiving termination message
- symmetric termination
  - both sides must agree
  - there is no fully reliable solution if communication is not secured (Two Armies Problem / Coordinated Attack Problem / Two Generals' Problem)

Two Armies Problem

A will be winner if A1 and A2 start attack at one moment.



- A1 sends the message, but it is not clear if the message passes the enemy area
- A2 sends the acknowledgment, but it can be also lost
- A1 should send the acknowledgment of received acknowledgment
- It s still not clear if they agreed: another acknowledgment?
- Prove of non-existence of solution is by contradiction: let n be the number of messages enough for the protocol to work correctly. If the last message is lost: sender of the last message will attack and the other side will not – contradiction.

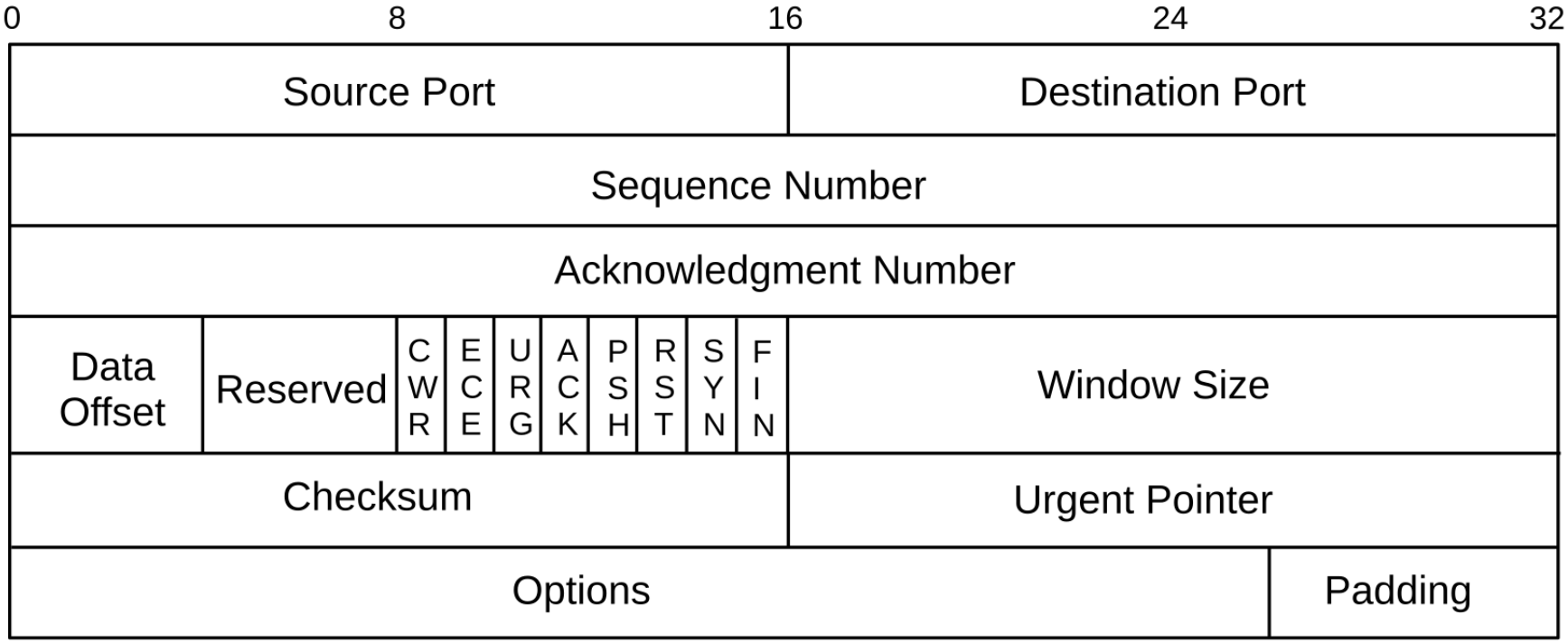
Flow control and congestion control

- similar principle as at the link layer
  - detection of errors
  - elimination of duplicated packets (sequence numbers)
  - limited number of unacknowledged packets
  - sliding window
- congestion control
  - prevents from overwhelming the network

TCP

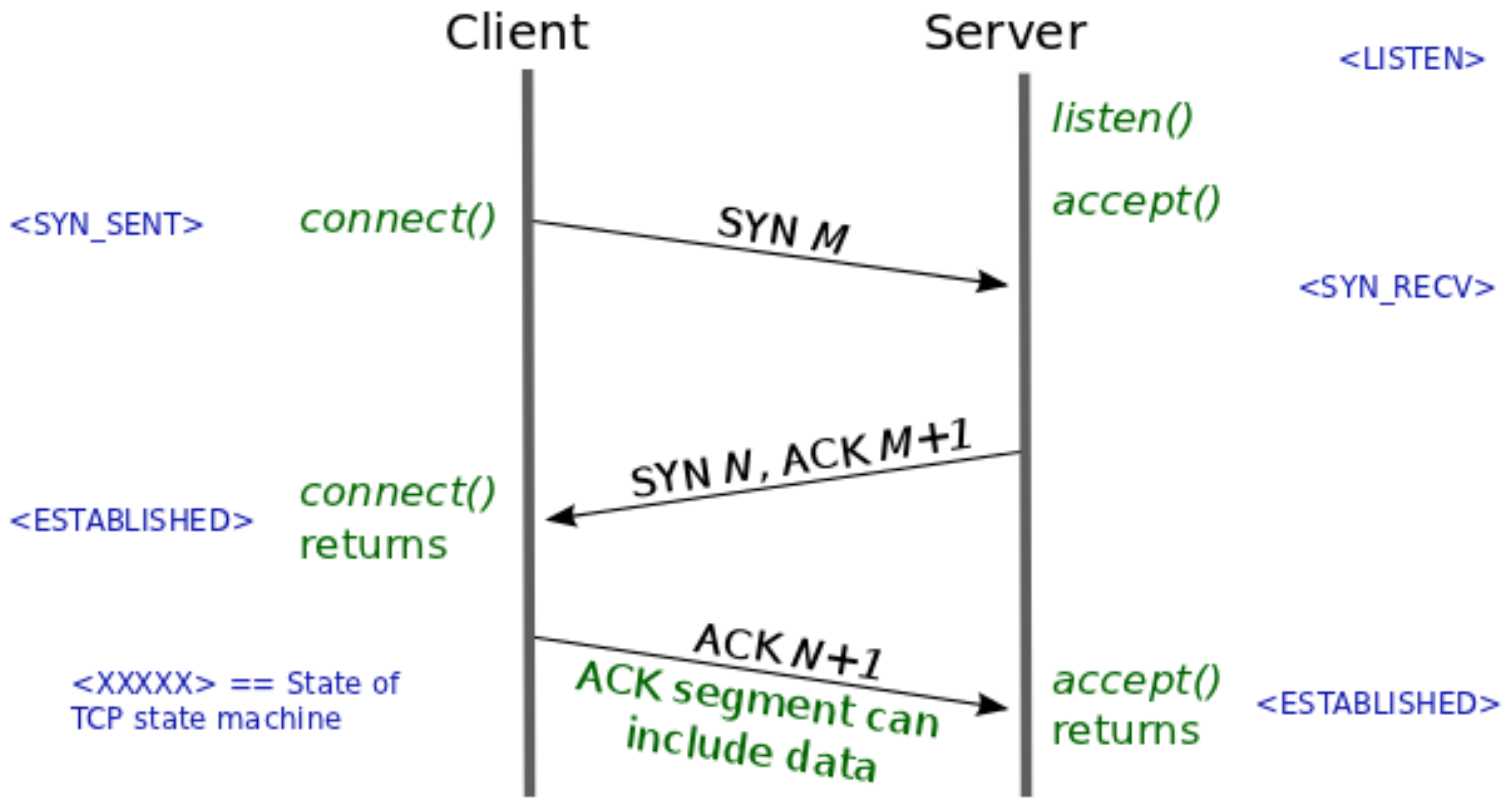
- Transmission Control Protocol
- service on layer 4
  - connection-oriented
  - guarantees error free data transmission
  - duplex
- many implementations – various improvements
  - Reno
  - Tahoe
  - Vegas
  - New Reno, CUBIC and many others

TCP header



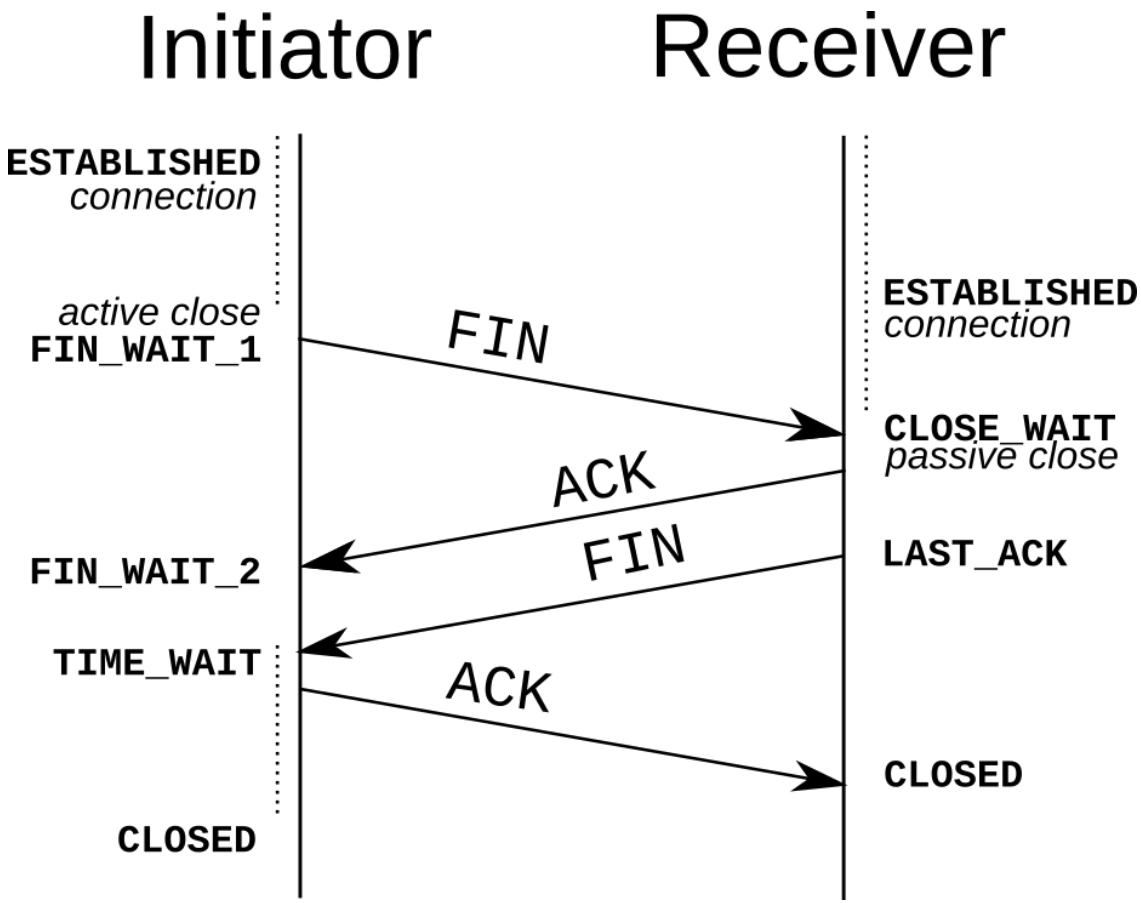
Source: https://intronetworks.cs.luc.edu/current/html/tcp.html

TCP connection establishment



Source: https://lwn.net/Articles/508865/

TCP connection termination



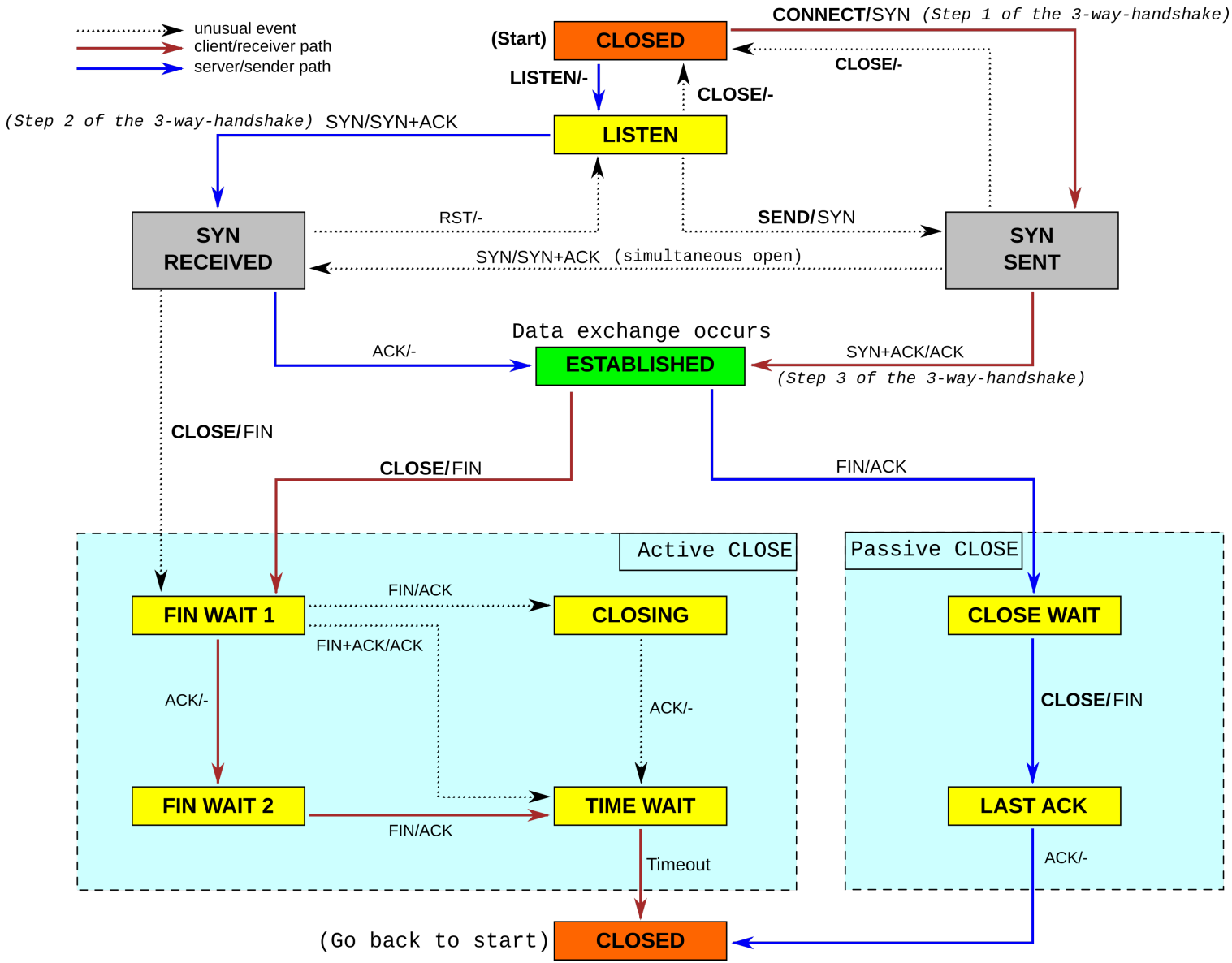
Source: Wikipedia

Note: after second message (ACK from receiver) connection is half-closed and receiver can still send data to initiator.

TCP properties

- data delivery guarantee:
  - checksum
  - detection of duplicate packets
  - retransmission
  - correct ordering
  - timeout
- sliding window
- congestion detection

TCP state diagram (source: Wikipedia)



Sliding window

- receiver reserves buffer – window for receiving data
- receiver specifies the current window size in the reply
  - sender must not send more data than the size of the window, then waits for acknowledgment
  - window size is decreased if the receiver does not manage to process the data
  - the window and its shrinking are used for flow control

Acknowledgment I

- receiver sends ACK, where is the sequence number of the byte, which is expected
- this confirms that all the previous bytes has been received
- receiver does not need to send acknowledgment individually for all packets

Timeout (on the sender side):

- sender has no ACK up to expected time
- unacknowledged data will be retransmitted

Acknowledgment II

Duplicate ACK

- if a packet does not arrive, but the following does, receiver repeats the last ACK
- it will be used as an indicator that the packet might have been lost

Timeout (on receiver side)

- no packet arrived up to expected time
- send again the last ACK
- max. 3 retransmissions of ACK

Window size

- 16 bits $\Rightarrow$ max 64kB
- problem for fast links or great delay
  - bandwidth $*$ delay
  - example: 1Gbps link:
    - sending of 64kB takes 0.5 ms
    - for delay of 50 ms link is used only for 1% of its capacity
- solution: window scale option
  - a maximum value of 14 may be used for the scale factor $\Rightarrow$ 30 bits for window size

Congestion control variables

- MSS (Maximum Segment Size) – maximum size of the packet to be sent, defined by the receiver
- SSTHRESH (Slow-start Threshold) – limit for likely congestion
  - estimate how many unconfirmed data can be sent
  - is multiples of MSS
- CWND (Congestion Window) – sender window
  - how many unconfirmed data the sender sends

Congestion Control I

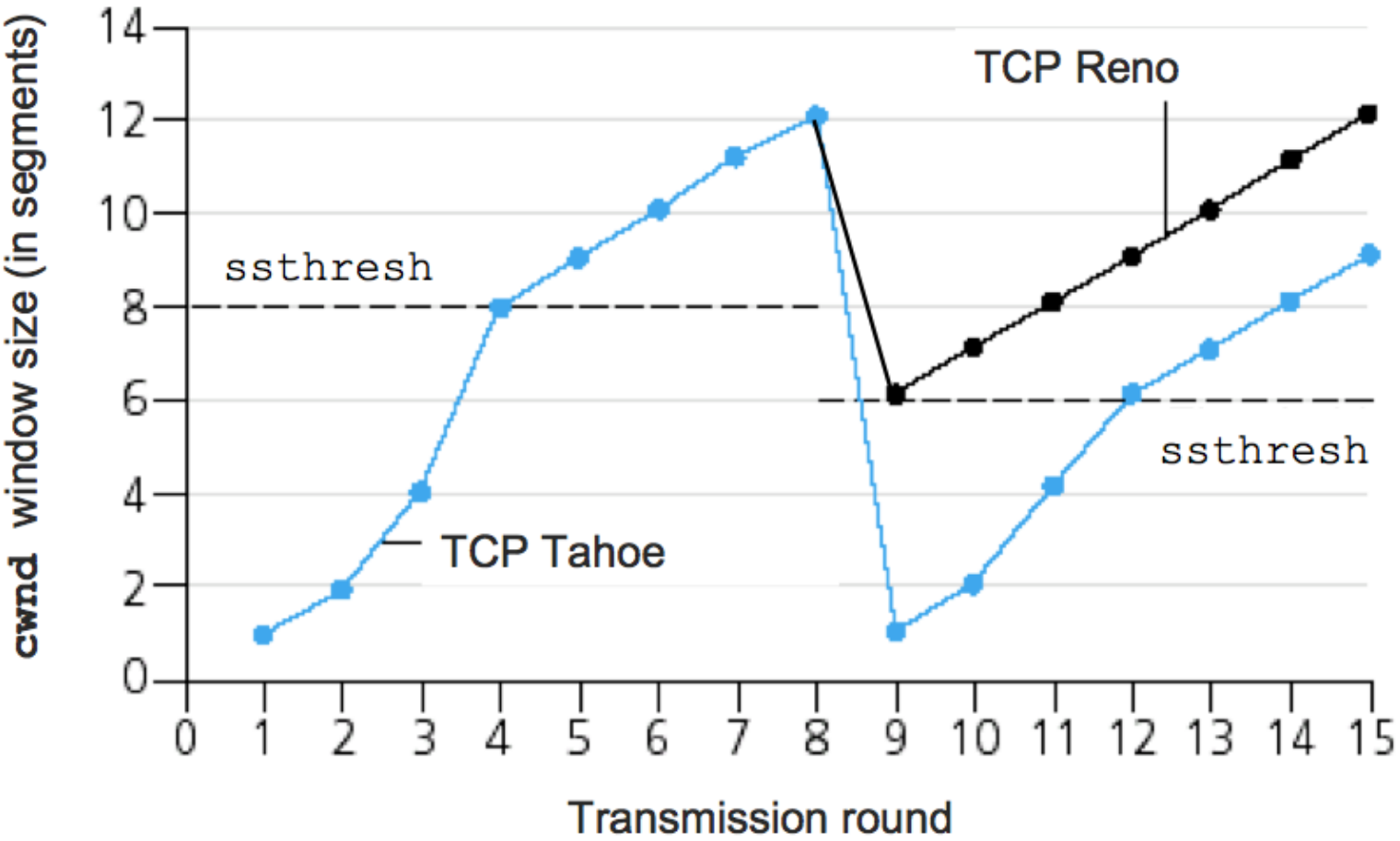
- limit on size of unacknowledged bytes
- slow start: initial value of CWND is 1
- congestion window size is increased 2x: grows exponentially
- after reaching SSTHRESH size is grown linearly

Congestion Control II

Congestion:

- packet or its acknowledgment is lost
- if ACK did not come at all (timeout):
  - transmission is repeated from the last acknowledged packet
  - slow start is repeated
- is duplicate ACK comes 3x:
  - "fast retransmit" and "fast recovery" is activated
  - lost packt is sent again, but not the following ones ("selective retransmit")
  - CWND is decreased by half

Evolution of TCP's congestion window (Tahoe and Reno)



Source: Kurose & Ross

TCP usage

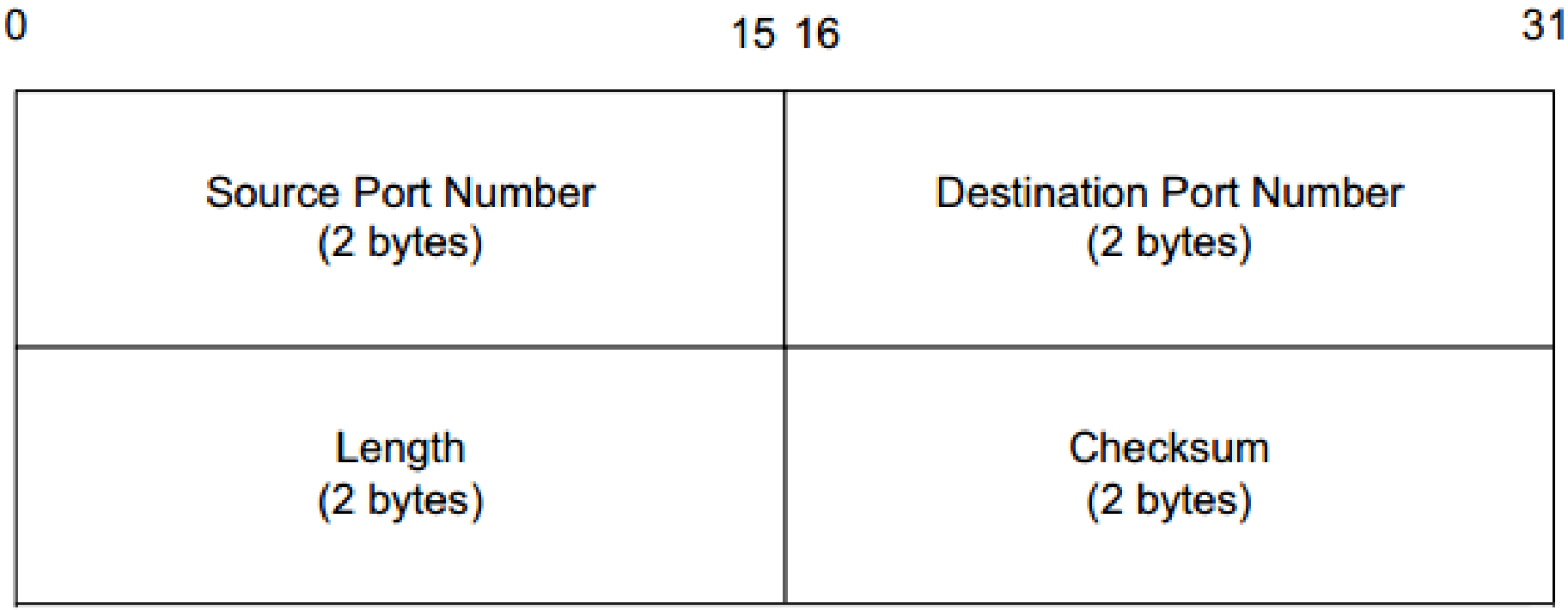
- TCP is good for applications which need reliable data channel
- TCP is not good for small blocks of data – high overhead (time and data)
- TCP is not suitable for real-time applications (VOIP, streaming)
  - the packet is delivered at all costs – unacceptable delays
- TCP is not suitable for embedded-systems (too complex)

UDP

User Datagram Protocol

- service on layer 4
- connection-less
- UDP datagram may be lost
- uses ports for multiplexing as TCP
- maximum 64 kB of data (avoid fragmentation in IP)
- example: DNS

UDP header


0
15 16
31
Source Port Number
(2 bytes)
Destination Port Number
(2 bytes)
Length
(2 bytes)
Checksum
(2 bytes)

UDP usage

- small blocks of data, when:
  - packet losses are not be critical (eg. DNS)
  - TCP overhead if not acceptable (eg. NTP)
- real-time applications

RTP

Real-time Transport Protocol

- streaming data between endpoints in real time
- header contains "timestamp"
  - time from the beginning of the stream
  - the unit depends on the application
  - allows to interpret the received block
- implemented mostly over UDP
- RTP usage:
  - any multimedia formats (H.264, MPEG-4, MJPEG, MPEG, ...)
  - videoconferencing
  - data streaming
  - IP Telephony

RTCP

RTP Control Protocol

- provides statistics and control information for an RTP session
- does not transport any media data itself
- used to control quality of service