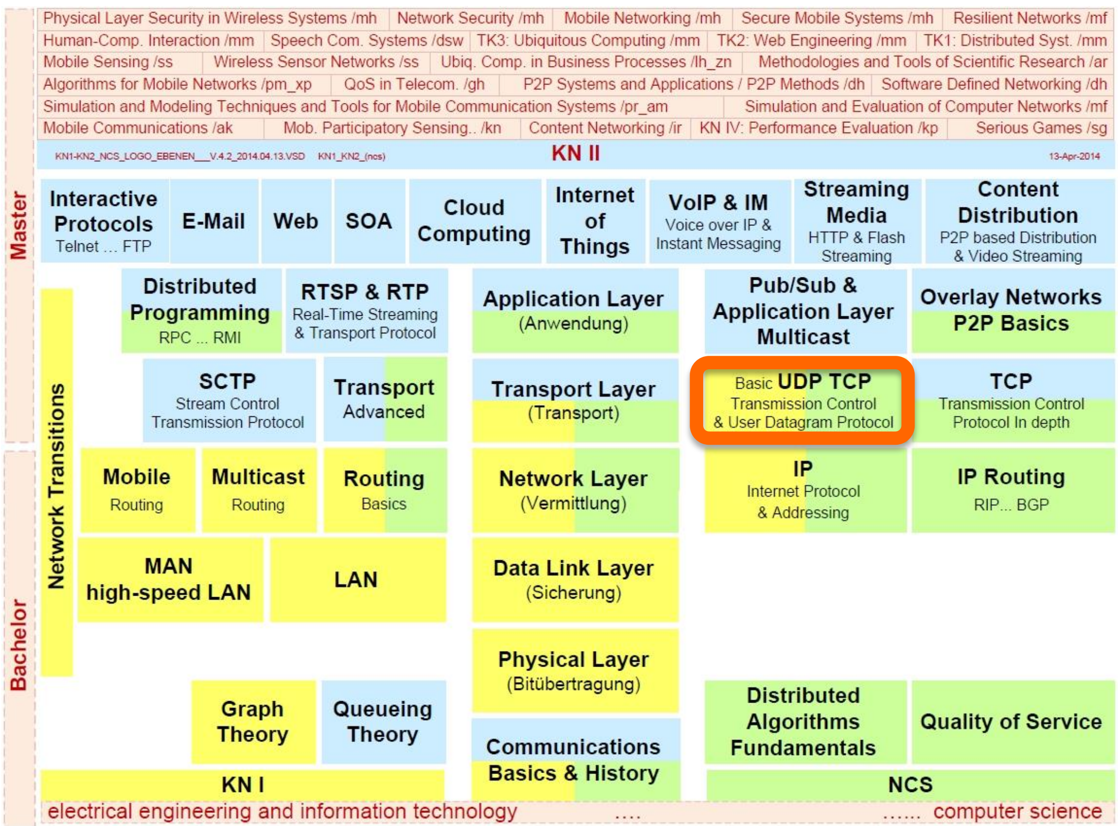
Communication Networks I

Transport Layer Protocols in General
UDP – TCP (Basics)



Prof. Dr.-Ing. **Ralf Steinmetz**
KOM - Multimedia Communications Lab

Overview

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Application layer

- communication between applications required
- applications communicate
  - locally by interprocess communication
  - between systems via TRANSPORT SERVICES

Transport layer

- interprocess end-to-end communication via communication networks

Internet protocol IP

- enables only end system - to - end system communication

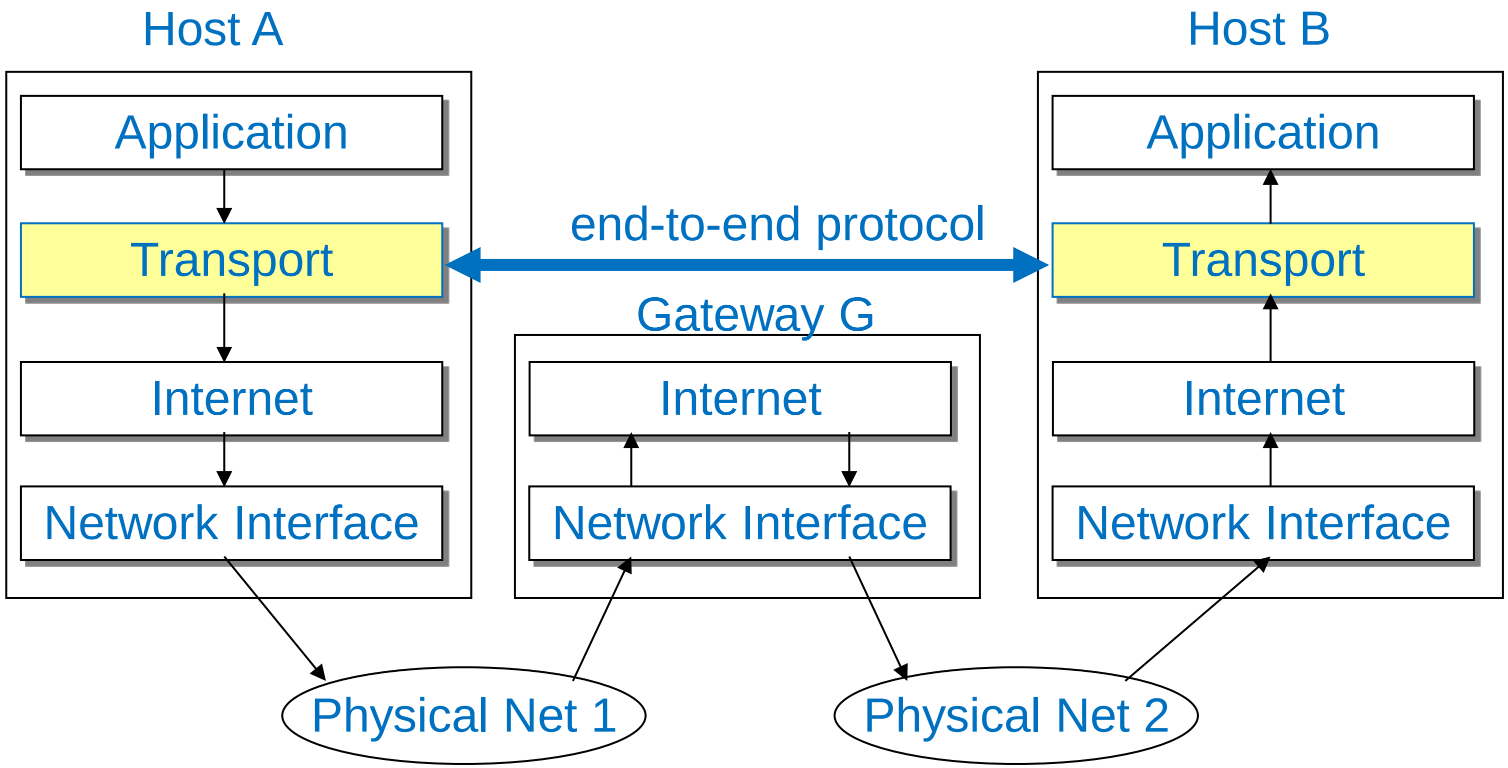
Internet

- UDP = User Datagram Protocol
- TCP = Transmission Control Protocol

ISO-OSI

- are practically irrelevant today
- but show overall design space

Internet Transport Layer (In General & Addressing)



Lowest level end-to-end protocol

- header generated by sender is interpreted only by destination
- routers / gateways view transport header as part of the payload

Adds extra functionality to the best effort packet delivery service provided by IP

- makes up for shortcomings of core network

Some Functions of Transport Protocols

Multiplexing/demultiplexing data for multiple applications
- uses “port’’ abstraction

Connection establishment
- logical end-to-end connection

Error control
- hides unreliability of network layer from applications
- some types of errors:
  - corruption, loss, duplication, reordering

End-to-end flow control
- to avoid flooding the receiver

Congestion control
- to avoid flooding the network

TECHNISCHE UNIVERSITÄT DARMSTADT

3 types of identifiers:

- names, addresses and routes

"The NAME of a resource indicates

- WHAT we seek,

an ADDRESS indicates

- WHERE it is, and

a ROUTE tells

- HOW TO GET THERE"
    - [Shoch 78]:

Address identifies

- type of service or application (destination of communication)

A process at destination host is ultimate destination for a message
→ why not addressing the process using [destination IP, remote process id]?

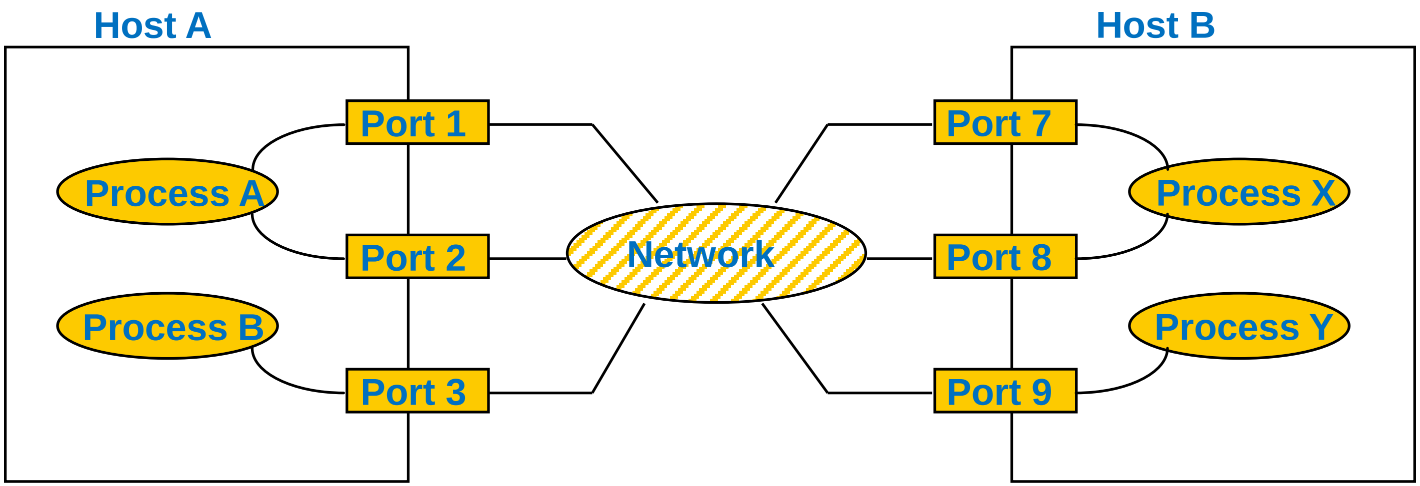
- processes are generated/terminated dynamically
  - i.e. process number rarely known
- relationship (Service,Process) is not one to one and is not fixed
  - 1 process can supply multiple services
  - various processes can provide same service
- not possible to replace processes receiving datagrams without informing all senders (e.g. on reboot)
- need to identify destination
  - based on the implemented function
  - without knowing the process that implements it

Port: Addressing Concept

In order to communicate, the sender needs to know

- Network layer (IP) address of receiving host
- Port number of receiving service

Possible communication scenarios:



- **In general**
  - insert network layer address and ports of sender and receiver in all packets to allow for complex addressing / multiplexing

Port: Addressing Concept

Each machine is imagined to have a set of abstract destination points called "protocol ports"

- identified by a positive integer
- local operating system provides interface mechanism
  - that processes use to specify a port or access it
- ports are in general buffered

To communicate with a foreign port

sender needs to know

- IP address of the destination device and
- protocol port number of the destination within that device

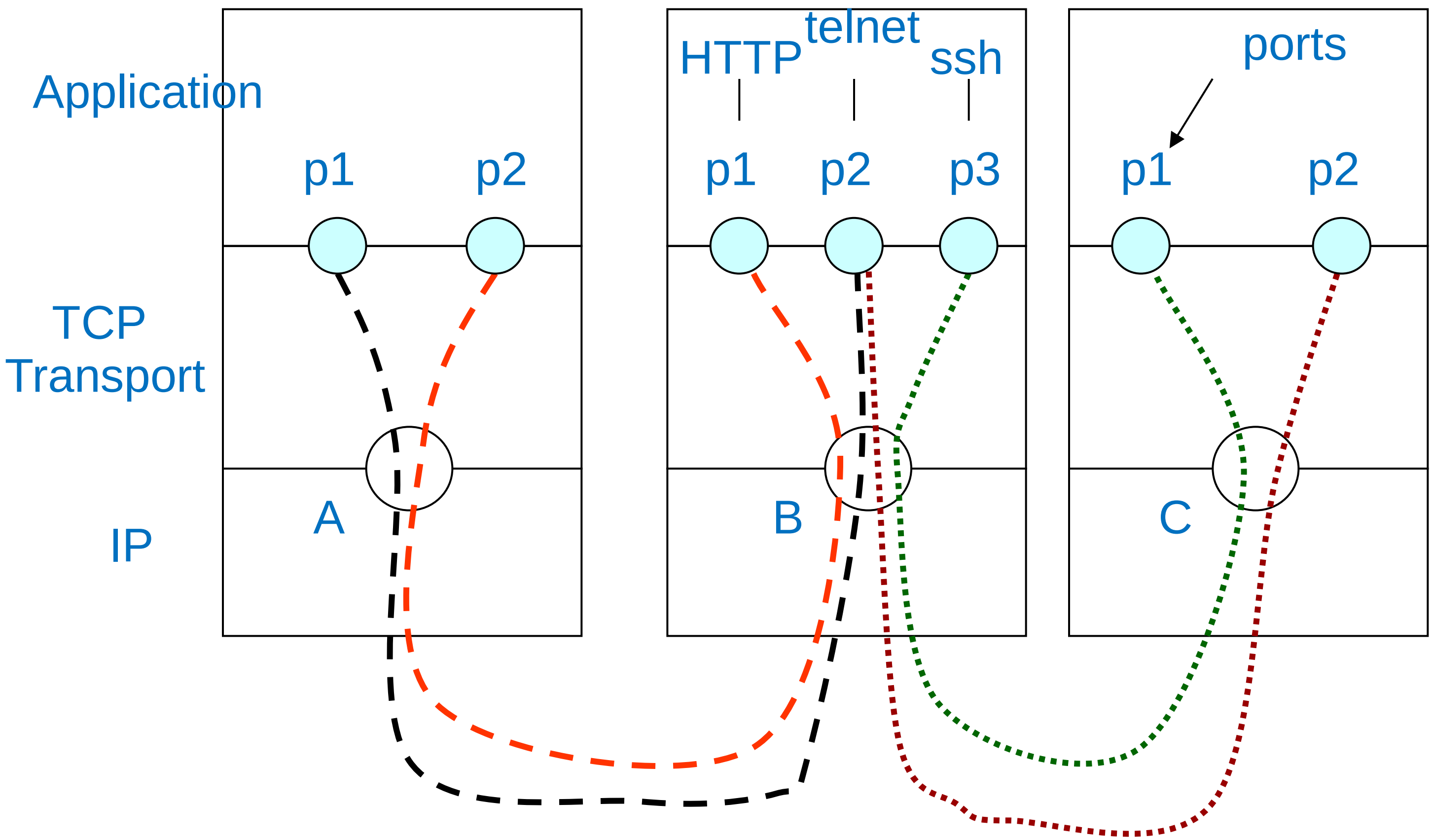
➔ Concept of an abstract communication endpoint:

- Port

e.g. Use of Transport Layer Port Number

In TCP, a data stream is identified by a set of identifiers (numbers):

- Source Address, Destination Address,
- Source Port, Destination Port
- (and protocol identifier, here for TCP)



Reserved Port Numbers

Decimal	Keyword	UNIX Keyword	Description
0			Reserved
1	TCPMUX		TCP Multiplex
5	RJE		Remote Job Entry
7	ECHO	echo	ECHO
9	DISCARD	discard	Discard
11	USERS	systat	Active Users
13	DAYTIME	daytime	Daytime
15		netstat	Network status program
17	QOUTE	qotd	Quote of the Day
19	CHARGEN	chargen	Character Generator
20	*FTP-DATA*	*FTP-DATA*	*FILE TRANSFER PROTOCOL (DATA)*
21	*FTP*	*FTP*	*FILE TRANSFER PROTOCOL*
23	*TELNET*	*TELNET*	*TERMINAL CONNECTIONS*
25	*SMTP*	*SMTP*	*SIMPLE MAIL TRANSFER PROTOCOL*
37	TIME	time	TIME
42	NAMESERVER	name	Host Name Server

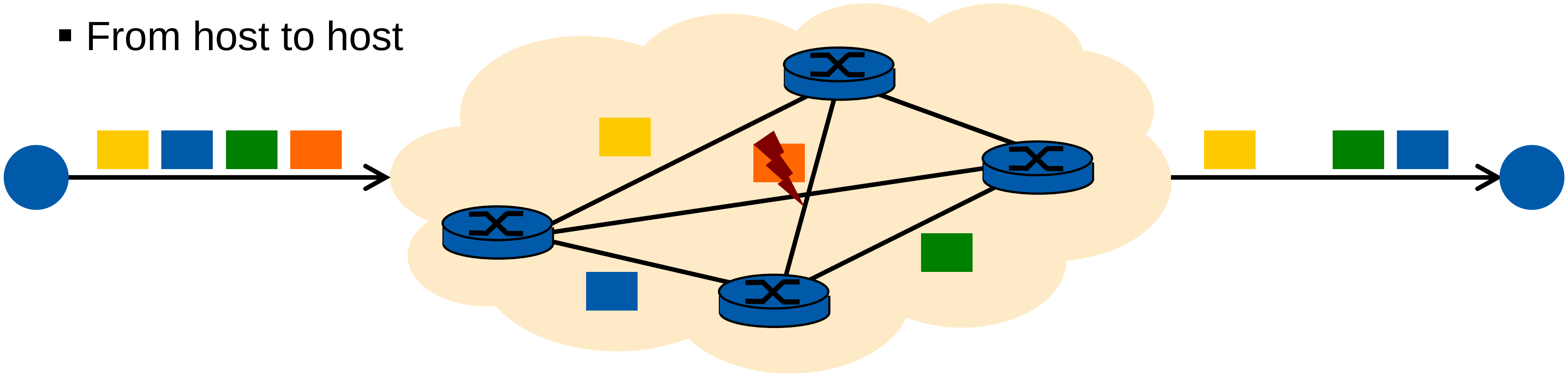
- TCP and UDP have their own assignments
  - this table shows some examples for TCP

Reserved Port Numbers

Decimal	Keyword	UNIX Keyword	Description
43	NICNAME	whois	Who is
53	DOMAIN	nameserver	Domain Name Server
77		rje	any private rje service
79	FINGER	finger	Finger
80	*HTTP*	*HTTP*	*WORLD WIDE WEB*
101	HOSTNAME	hostname	NIC Host Name Server
102	ISO-TSAP	iso-tsap	ISO TSAP
103	X400	x400	X.400 Mail Service
104	X400-SND	x400-snd	X.400 Mail Sending
110	*POP3*	*POP3*	*REMOTE EMAIL ACCESS*
111	SUN RPC	sunrpc	SUN Remote Procedure Call
113	AUTH	auth	Authentication Service
117	UUCP-PATH	uucp-path	UUCP Path Services
119	NNTP	nntp	USENET News Transfer Protocol
129	PWDGEN		Password Generator Protocol
139	NETBIOS-SSN		NETBIOS Session Protocol
160-1023	Reserved		

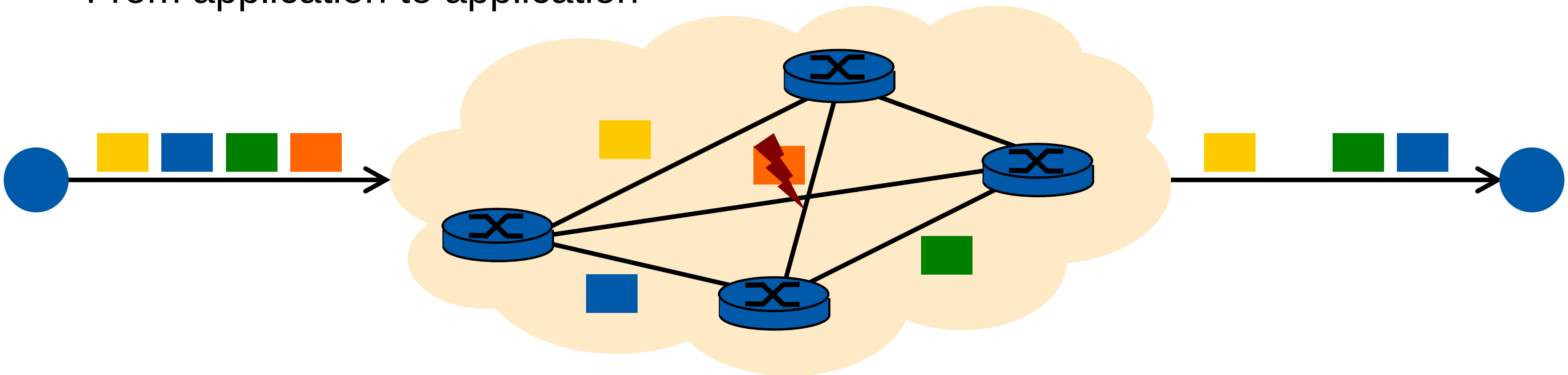
Internet layer offers best effort packet delivery

- From host to host



UDP offers best effort message delivery

- From application to application



UDP – User Datagram Protocol

Specification:
- RFC 768

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

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

Characteristics
- No flow control
  - application may transmit
    - as fast as it can/wants to and
    - as fast as the network permits
- 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 / multicast and streaming

UDP: Message Format

Sender port
- 16 bit sender identification
  - is optional (if not used: 0000000000000000)
  - use: response may be sent there

Receiver port
- Receiver identification

Packet length
- In bytes (including UDP header)
- Minimum: 8 (byte), i.e., header without data

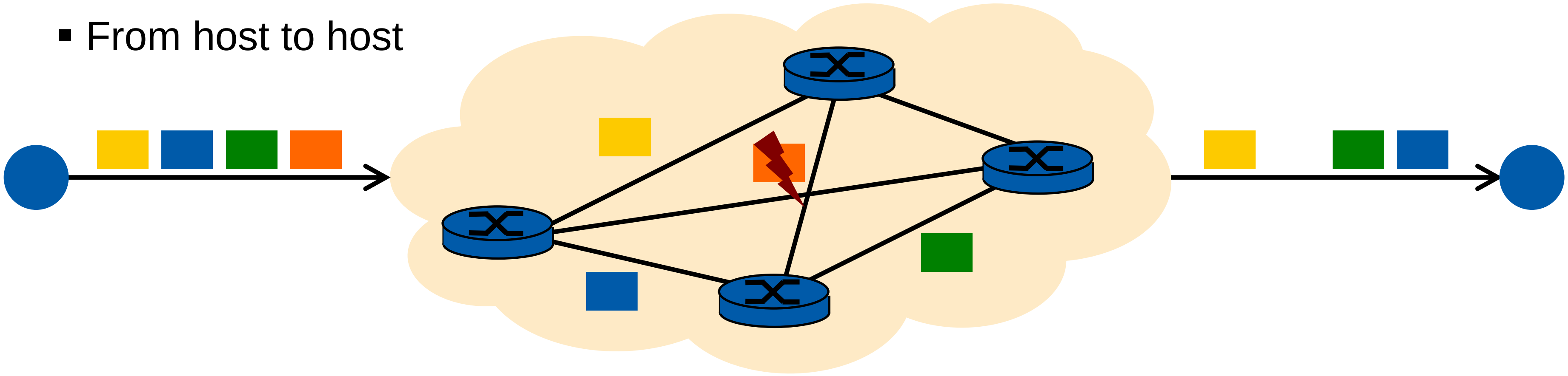
Checksum
- Of header and data for error detection
- Use of checksum optional

0	16 31
Sender Port	Receiver Port
Packet Length	Checksum
Data	
⋮	

UDP Header

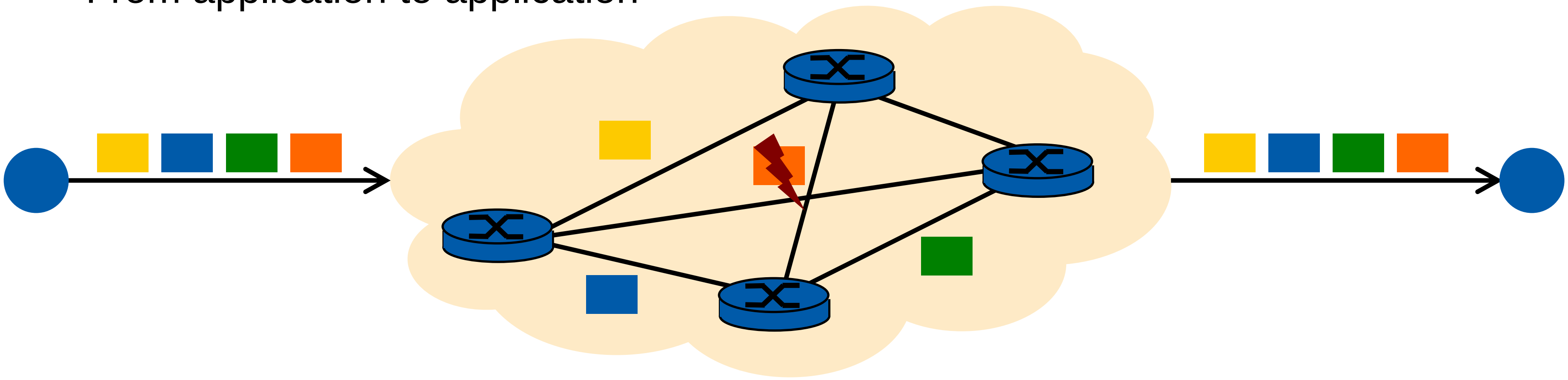
Internet layer offers best effort packet delivery

- From host to host



TCP offers reliable byte stream

- From application to application



Motivation: network layer provides unreliable connectionless service

- Packets and messages may be
  - duplicated
  - delivered in wrong order
  - faulty
- Given such an unreliable service, each application would have to implement error detection and correction separately
- Network or service can
  - impose packet length
  - define additional requirements to optimize data transmission
  - i.e., each application would have to be adapted separately
  - ➔ do not reinvent the wheel for every application

➔TCP is the Internet transport protocol providing reliable end-to end byte stream over an unreliable internetwork

Specification

- RFC 793 - Transmission Control Protocol: originally
- RFC 1122 and RFC 1323: errors corrected, enhancements implemented

TCP in Use & Application Areas

Each machine supporting TCP has a TCP transport entity composed of

- Library procedure
- User process
- Part of kernel

TCP transport entity manages

- TCP streams
- Interfaces to IP layer

TCP transport entity at sending side

- Accepts user data streams for local processes
- Splits them into pieces <= 64 KB, typically 1460 bytes
  (to fit into single Ethernet frame with IP and TCP headers)
- Sends each piece as separate IP datagram

TCP transport entity at receiving side

- Gets TCP data from datagram received at host
- Reconstructs original byte streams

TCP in Use & Application Areas

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 must ensure reliability

- IP layer doesn’t guarantee that datagram will be delivered properly / in order
  - TCP must handle this, e.g. timeout and retransmit / reorder
    → i.e. 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
  - (up to 8 Mbps on 10 Mbps Ethernet)
- Can be used with LAN and WAN for
  - Low data rates (e.g., interactive terminal)
  - High data rates (e.g., file transfer)

Disadvantages compared to UDP

- Higher resource requirements
  - Buffering
  - Status information
  - Timer usage
- Connection set-up and disconnect necessary (even in case of short data transmissions)

Applications

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

Some Missing Characteristics

No broadcast

- No possibility to address all applications at the same time with a single message

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 (FEC)

4.2 TCP Features, Packet Header and Connection Management

Reliable bidirectional in-order byte stream
- Socket: SOCK_STREAM

Connections established & torn down

Multiplexing/ demultiplexing
- Ports at both ends

Error control
- Users see correct, ordered byte sequences

End-to-end flow control
- Avoid overwhelming the machines at either end

Congestion avoidance
- Avoid creating traffic jams within network

TCP Header:

| 0	| 16 31 |
Source Port	Dest. Port
Sequence Number	
Acknowledgment Number (Ack. No.)	
HL/RESV/Flags	Advertised Win.
Checksum	Urgent Pointer
Options..	

TCP Connection Setup

Q: Why is connection setup necessary?

A: Mainly to agree on starting sequence numbers

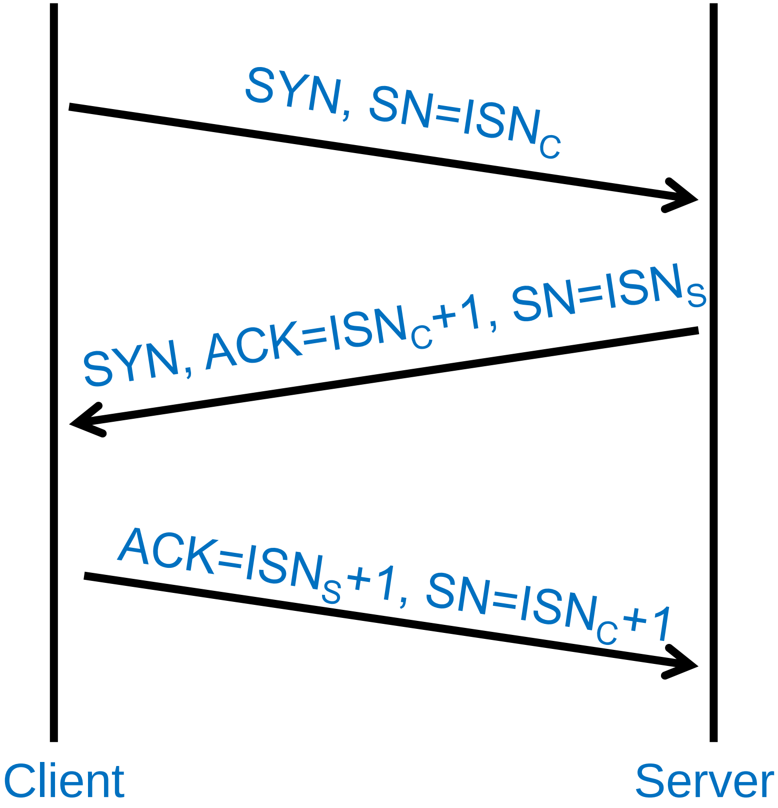
- Starting sequence number is randomly chosen
- Reason: to reduce the chance that sequence numbers of old and new connections overlap

TCP connection setup: 3-way handshake

1) Client sends message with
   - SYN flag set
   - Sequence number (SN) field containing Client Initial Sequence Number (ISN_C)

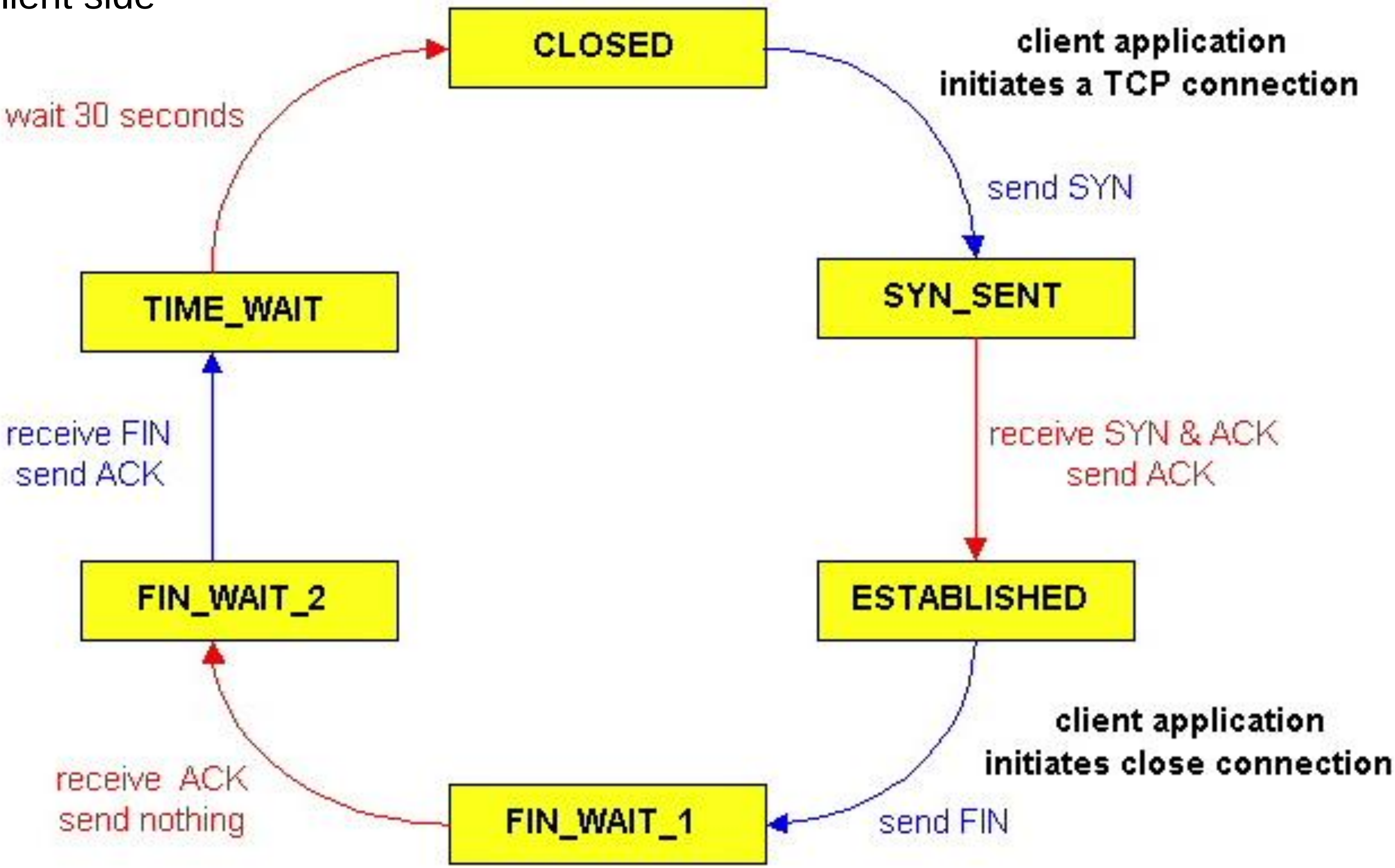
2) Server sends message with
   - SYN and ACK flags set
   - Acknowledgment field containing ISN_C+1
   - Sequence number field containing Server Initial Sequence Number (ISN_S)

3) Client send message with
   - ACK flag set
   - Acknowledgment field containing ISN_S+1
   - Sequence number field containing ISN_C+1



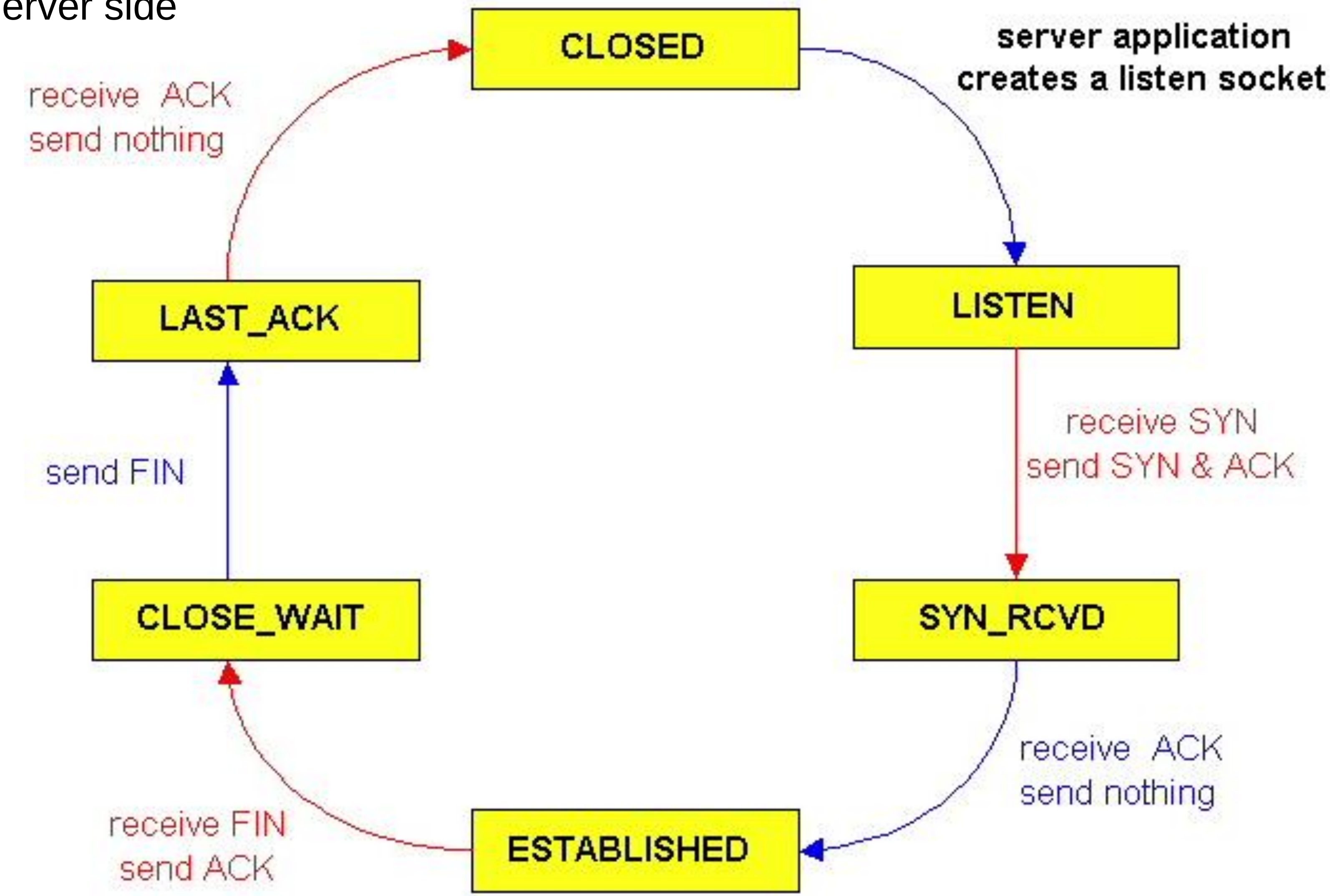
TCP connection lifecycle state diagram

- Client side



TCP connection lifecycle state diagram

- Server side



TECHNISCHE UNIVERSITÄT DARMSTADT

Sliding window protocol

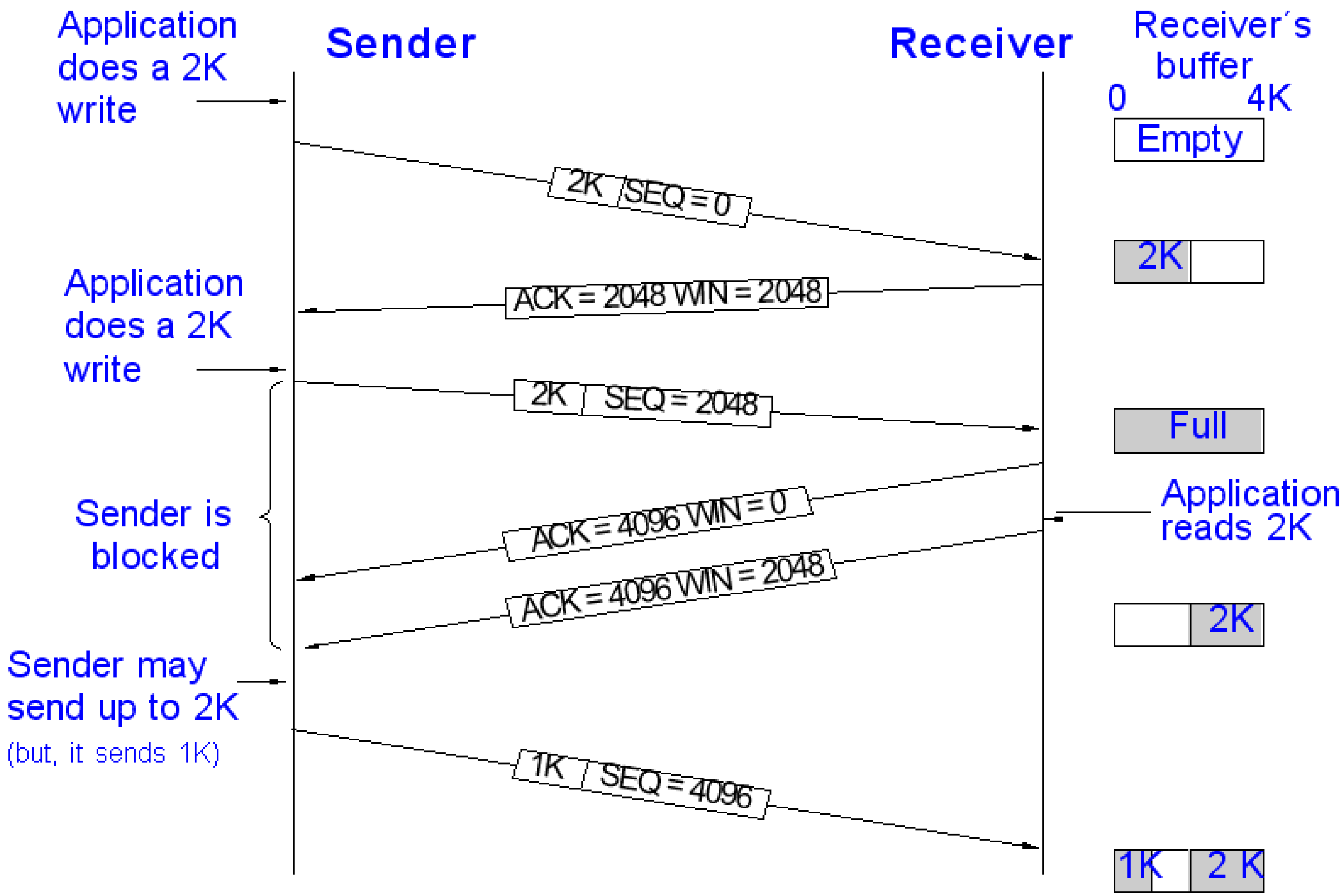
- for window size n
  → sender can send up to n bytes without receiving an acknowledgement
- when the data is acknowledged
  then the window slides forward

Window size determines

- how much unacknowledged data the sender can send

But there is one more detail …

e.g.



Application does a 2K write
Sender
Receiver
Receiver´s buffer
0
4K
Empty
2K SEQ = 0
2K
ACK = 2048 WIN = 2048
Application does a 2K write
2K SEQ = 2048
Full
Sender is blocked
ACK = 4096 WIN = 0
Application reads 2K
ACK = 4096 WIN = 2048
2K
Sender may send up to 2K
(but, it sends 1K)
1K SEQ = 4096
1K
2 K

Ongoing Communication

TECHNISCHE UNIVERSITÄT DARMSTADT

Bidirectional Communication

- each side acts as sender & receiver

- every message
  - contains acknowledgement of received sequence
    - even if no new data has been received
  - advertises window size
    - size of its receiving window
  - contains sent sequence number
    - even if no new data is being sent

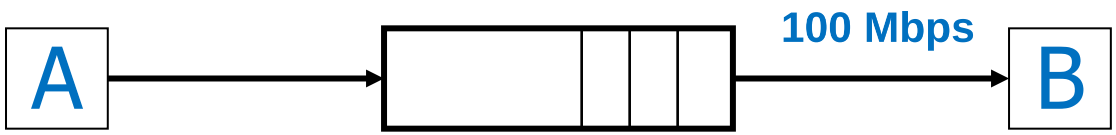
Three Congestion Control Problems

Adjusting to bottleneck bandwidth

Adjusting to variations in bandwidth

Sharing bandwidth between flows

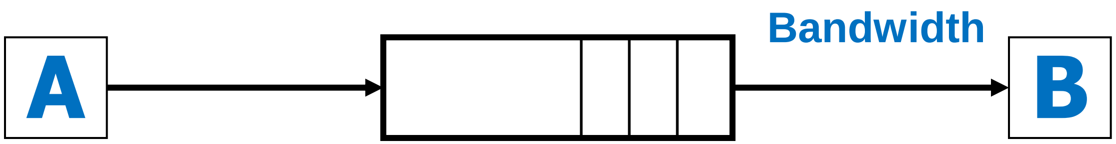
Single Flow, Fixed Bandwidth



Adjust rate to match bottleneck bandwidth

- without any a priori knowledge
- could be gigabit link, could be a modem

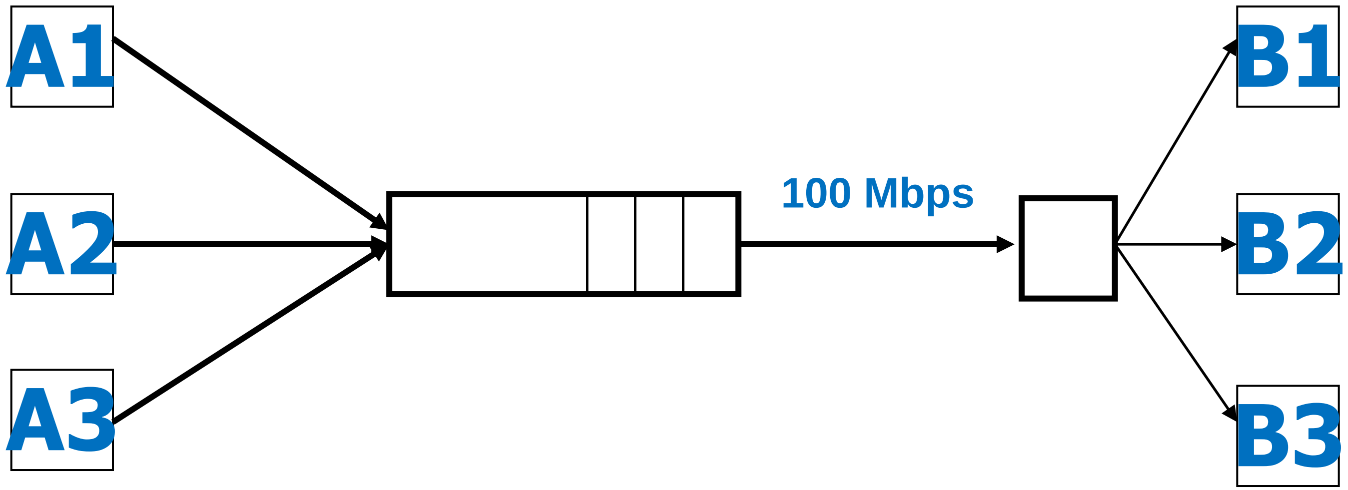
Single Flow, Varying Bandwidth



Adjust rate to match instantaneous bandwidth

Bottleneck can change because of a routing change

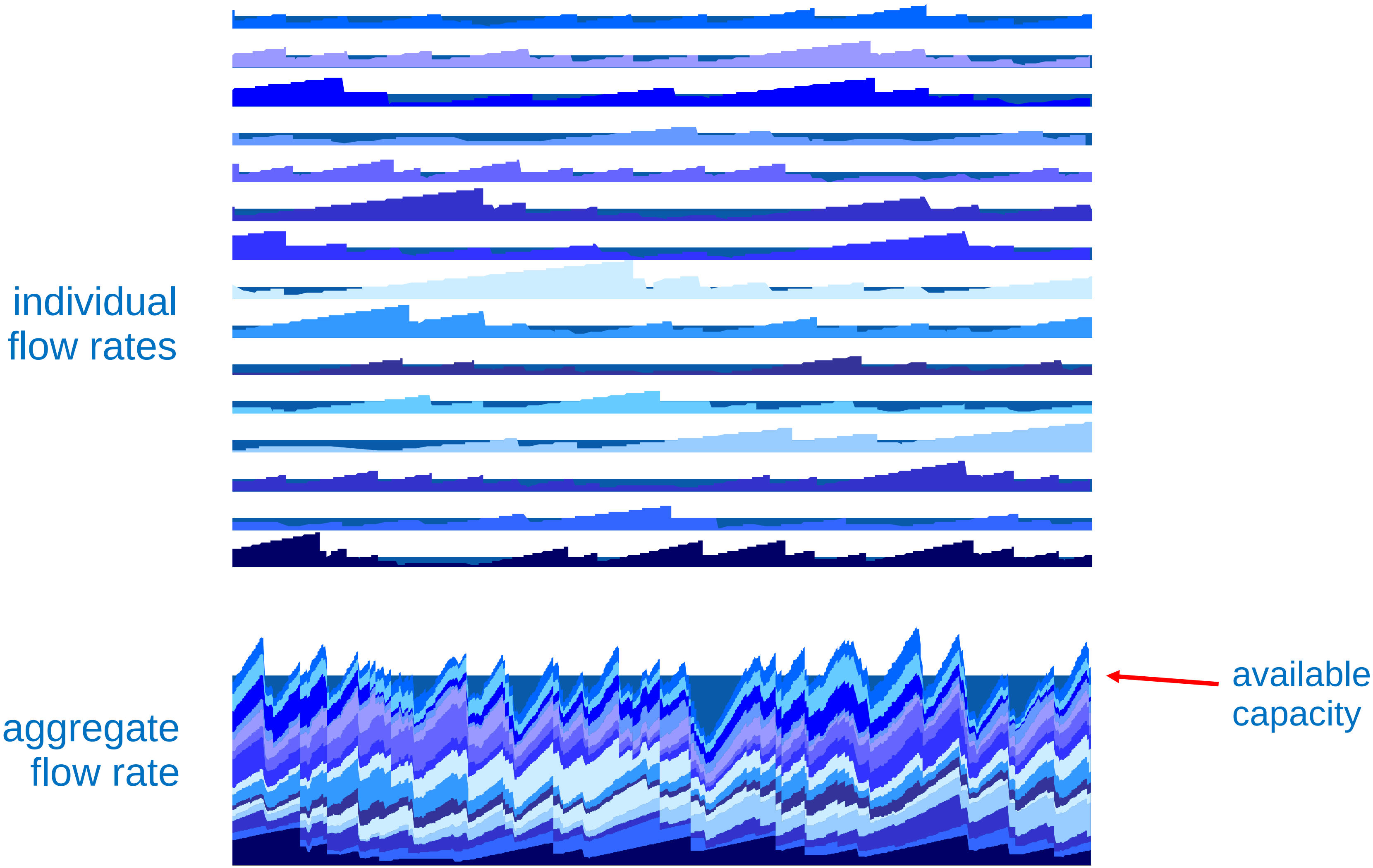
Multiple Flows



(two) issues:

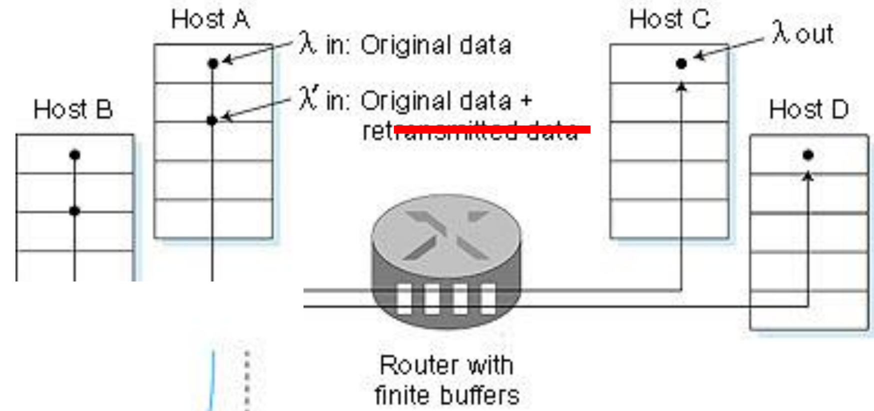
- Adjust total sending rate to match bottleneck bandwidth
- Allocation of bandwidth between flows

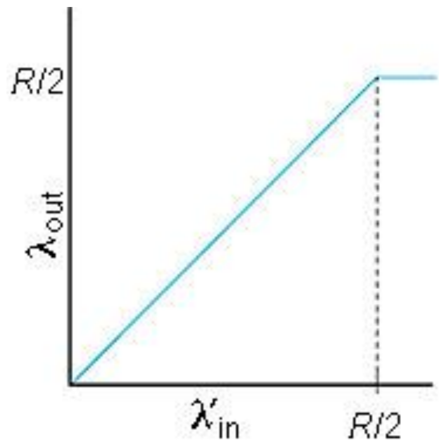
Multiple Flows and how TCP shares capacity



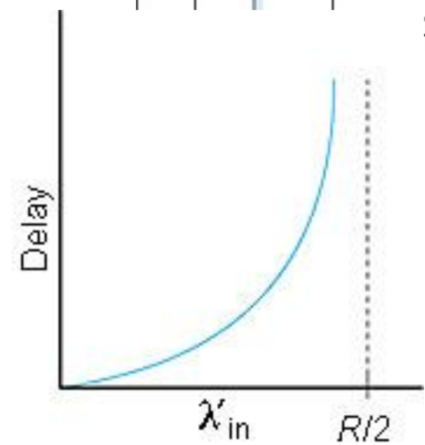
Source Damon Wischik, UCL

Why is Congestion Bad - Revised?

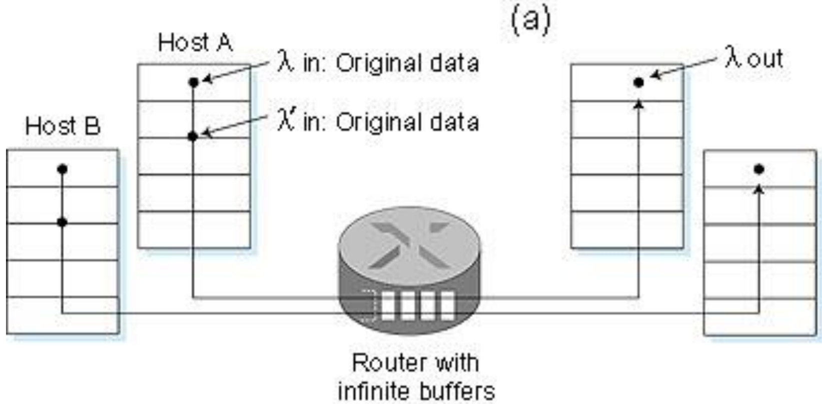




(a)



(b)



From Kurose & Ross

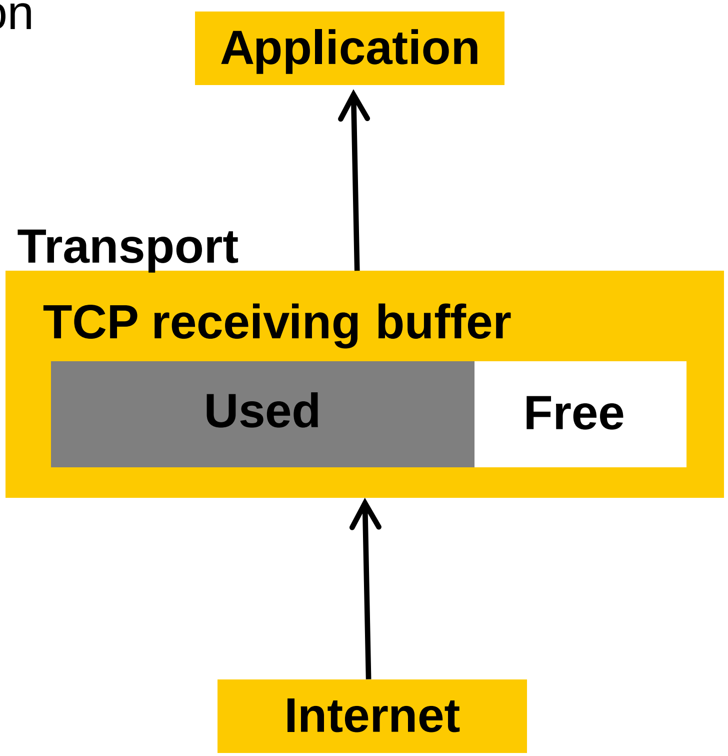
Flow control required to avoid flooding receiver

- Receiver allocates buffer space for receiving messages

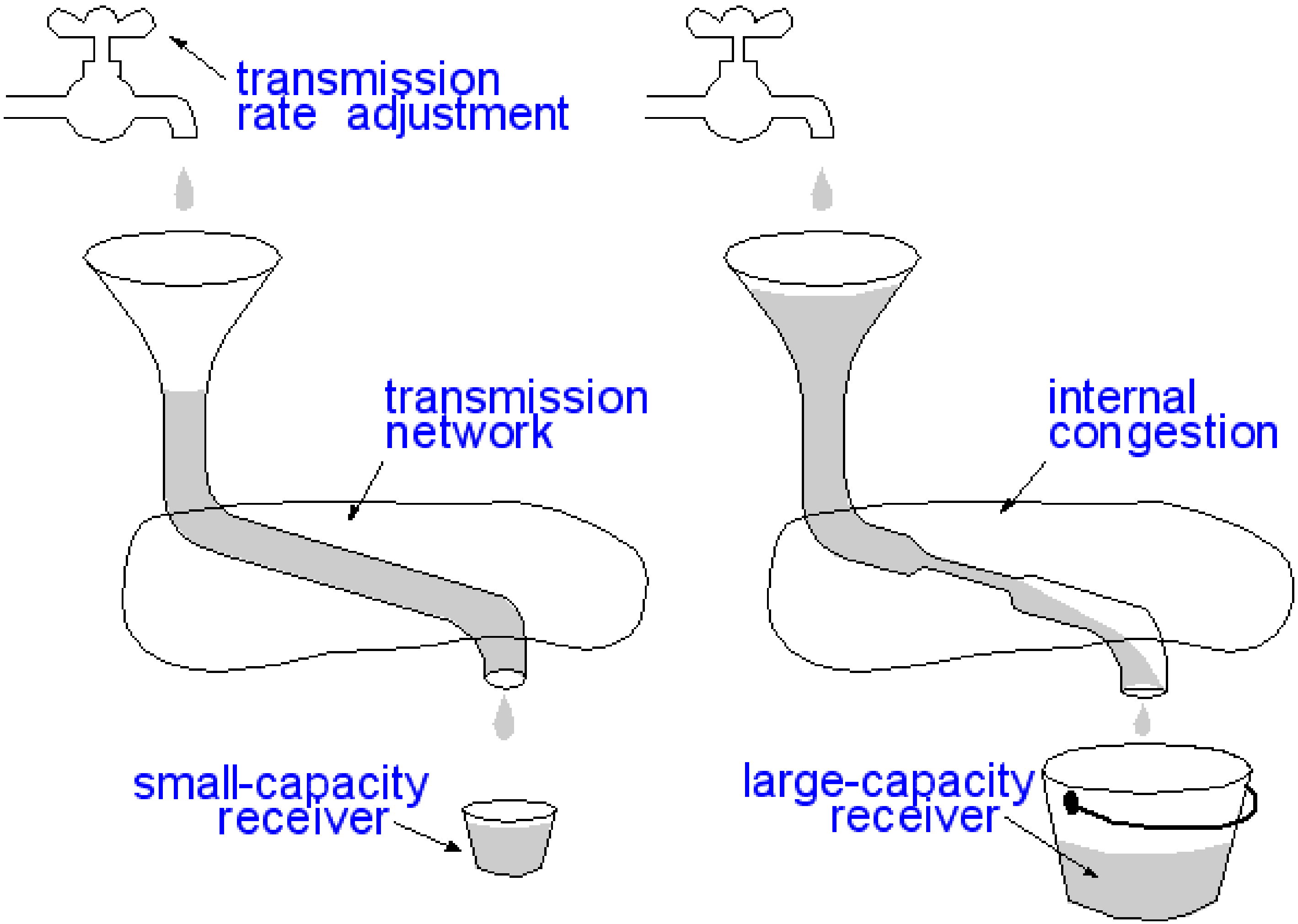
- Buffer becomes available when
  - Data is acknowledged and
  - Data is read from buffer by receiving application

- But: application may be slower than network
  - E.g. high load situations

→ Receiving buffer may become full
  - How to react?



Flow Control & Congestion Control



Controlled by
Window: advertised window awnd

Controlled by
Window: congestion window cwnd

Phases of Congestion Control

Phase 1: Slow start (getting to equilibrium)

- want to find this extremely fast and wasting time

Phase 2: Congestion Avoidance

- additive increase
  - gradually probing for additional bandwidth
- multiplicative decrease
  - decreasing cwnd upon loss/timeout

Initialization

Congestion Window (cwnd)

- Initial value is 1 MSS (=maximum segment size)
  - counted as bytes

Slow-start threshold Value (ss_thresh)

- Initial value is advertised window size

i.e. phase 1:

- slow start (cwnd < ss_thresh)

i.e. phase 2:

- congestion avoidance (cwnd >= ss_thresh)

Phase 1: TCP Slow Start

Goal:

- to discover roughly the proper sending rate quickly

Whenever

- starting traffic on a new connection, or whenever
- increasing traffic after congestion was experienced:

➔ initialize cwnd =1

each time a segment is acknowledged,

➔ increment cwnd by one (cwnd++)

Continue until

- reach ss_thresh
- packet loss

Slow Start Illustration

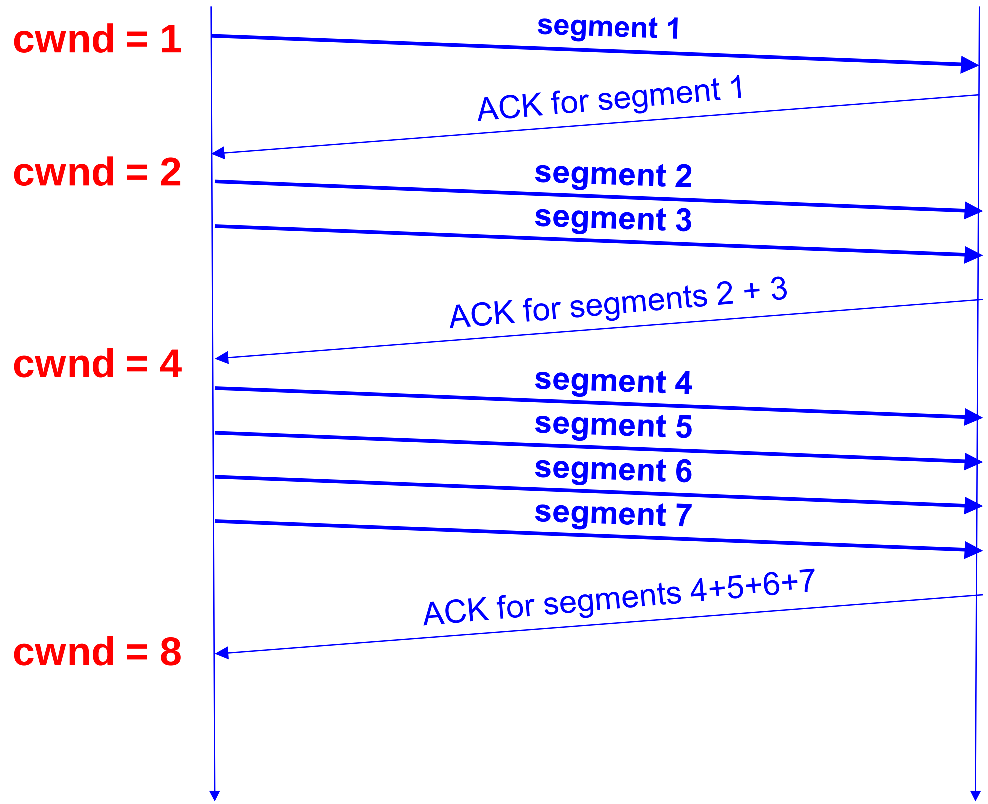
congestion window size grows very rapidly

TCP slows down the increase of cwnd

- when cwnd >= ss_thresh

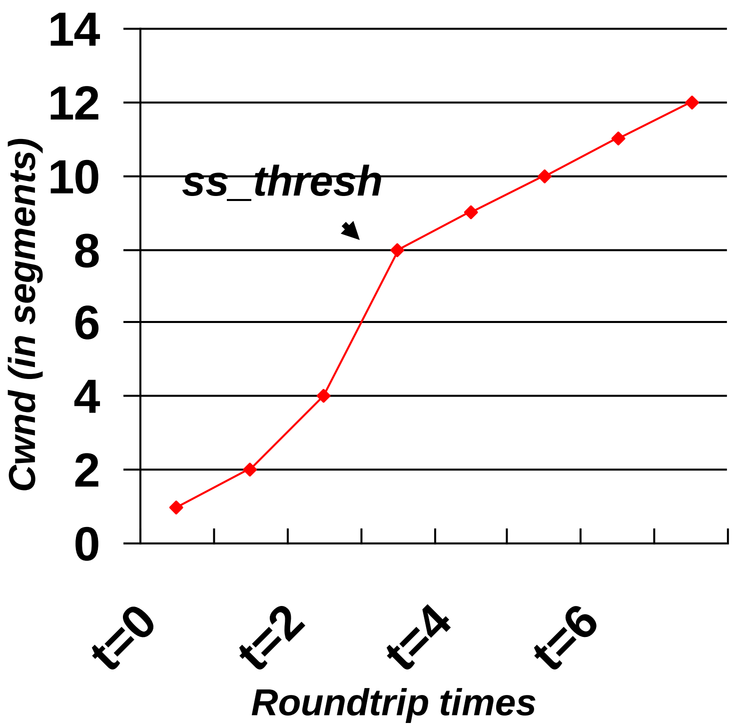
Observe:

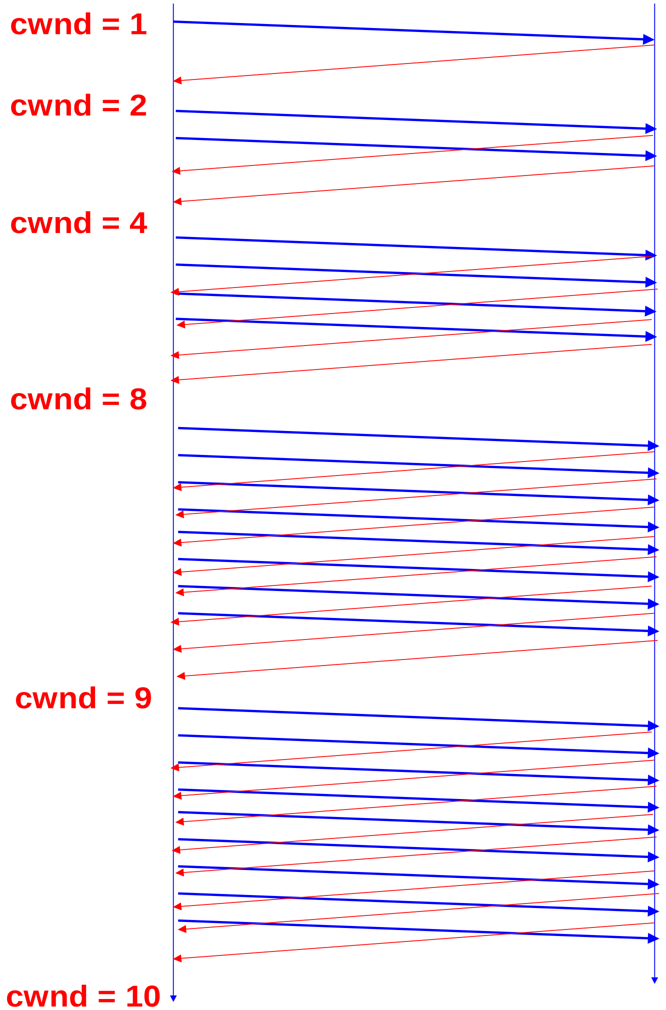
- each ACK generates 2 packets
- slow start increases rate exponentially
  - (doubled every RTT)



Example of Slow Start + Congestion Avoidance

Assume that ss_thresh = 8





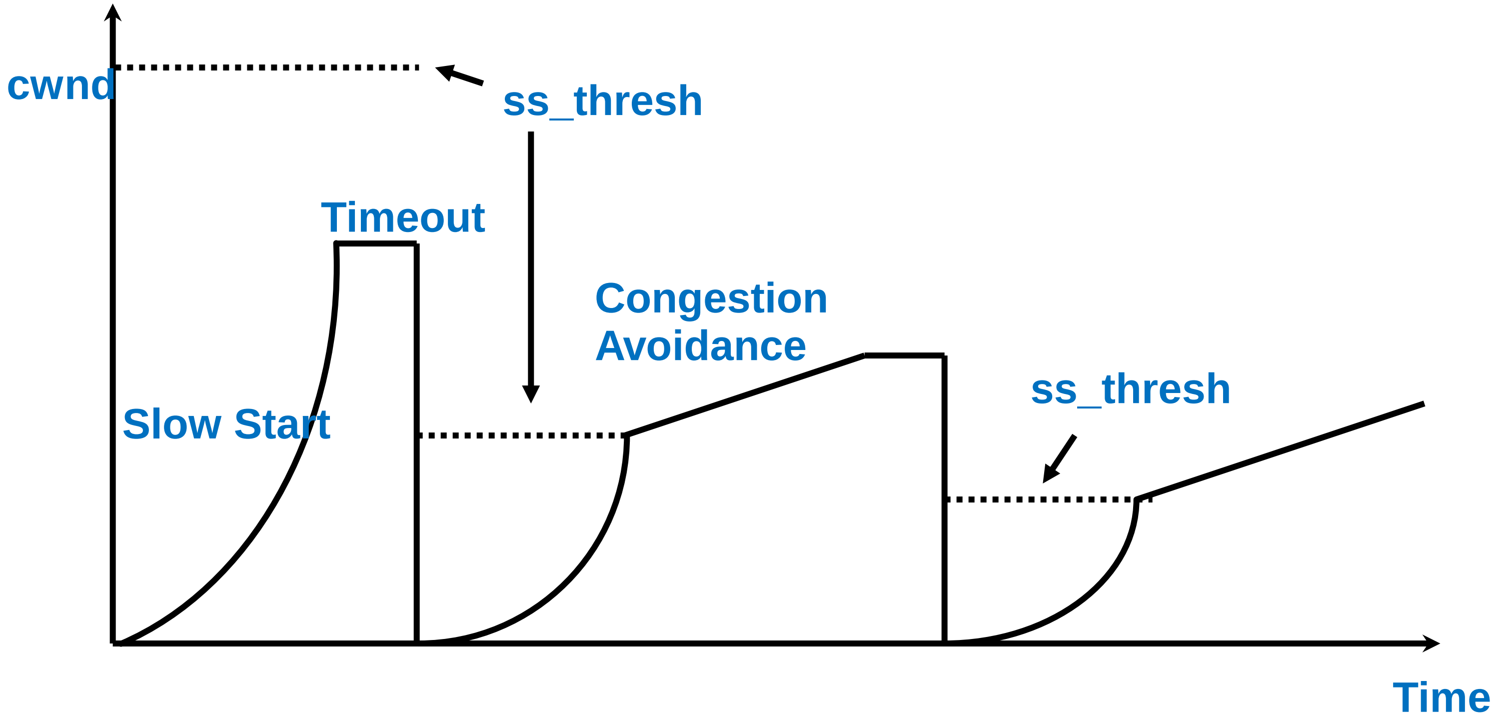
Congestion Avoidance: Multiplicative Decrease

Timeout = congestion

Each time when congestion occurs,

- ss_thresh is set to 50%
  of the current size of the congestion window:
  - ss_thresh = cwnd / 2
- cwnd is reset to one:
  - cwnd = 1
- and
  slow-start is entered

TCP illustrated



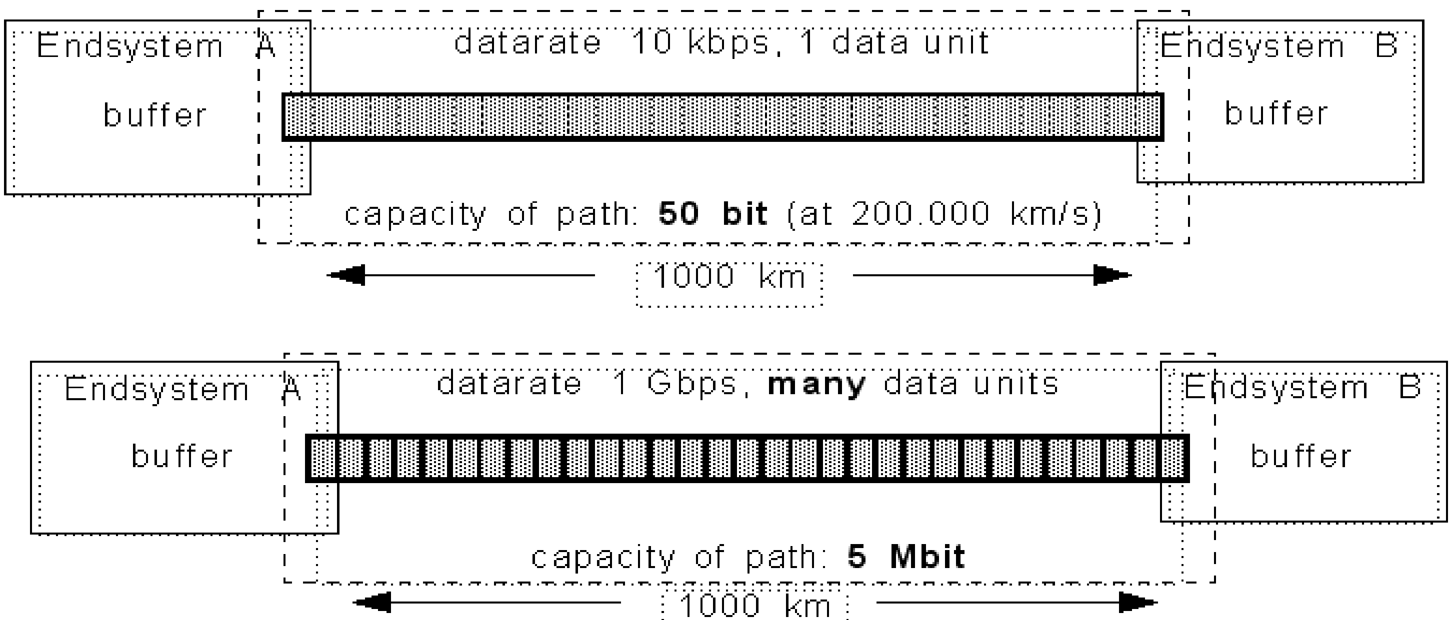
Motivation

- networks and applications have changed

Networks

- higher data rates
- also farther distances (e.g. also via satellite)
- networks for data storage

$$\text{Data amount} = \frac{\text{Data rate} \times \text{Distance}}{\text{Velocity of Propagation}}$$

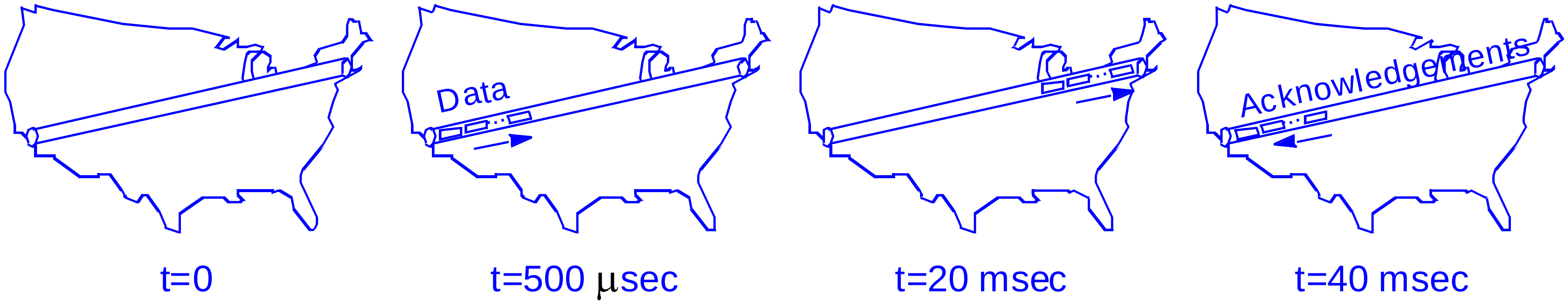


Further Development of Transport Protocols

TECHNISCHE UNIVERSITÄT DARMSTADT

Bandwidth-Delay Product increases
- bandwidth [bits/sec] * round-trip delay [sec]

Useful parameter for network performance analysis
- Capacity of pipe from sender to receiver and back (in bits)



Example:
- Transmission from San Diego to Boston
  - sending 64 KB burst (receiver buffer 64 KB), link: 1 Gbps
  - one-way propagation delay (speed-of-light in fiber): 20 msec
- Bandwidth-delay product: 40 million bit
- i.e.: sender would have to transmit burst of 40 million bits to keep pipe busy till ACK

Receiver window must be >= bandwidth-delay product
- for good performance