Understanding TCP

The concept and ideas behind it

No header bit definitions
No DoS protection stuff

What is TCP

- Transmission Control Protocol
  - Defined in RFC793 (in 1981!)
  - Based on „*A Protocol for Packet Network Intercommunication*" by Vinton G. Cerf, Robert E. Kahn (in 1974)
  - Updated over the years by a large number of additional RFC's

- TCP is the primary protocol on the Internet

- That is what I will talk about today

Purpose of TCP

- Provide a reliable data channel
  - It tries hard to deliver the data
  - And tells the application if it can‘t
- Sequential and in-order data stream
  - It ensures that A is delivered before B
- Over a lossy and ‚dumb‘ network (IP)
  - The Internet everywhere and anytime

Smart vs. Dumb (1)

- Two network types exist

- Smart network with dumb terminals
  - Terminal is just a presentation device
  - All the logic and data handling is in the network
  - Centralized approach
    - Everything has to be implemented and prepared in the network

  - Examples:
    - Telephony network
    - Compuserve, AOL, MSN, Minitel

Smart vs. Dumb (2)

- Dumb network with smart terminals
  - Terminal is also doing data handling

  - The network is just a dumb packet transporter
    - Stateless to any packet flows
    - Network is usage agnostic
    - Every packet is just a packet like all the others

  - Decentralized approach
    - The terminal has to implement the data handling itself
    - End to end principle

  - Examples:
    - Internet
    - X.21 network (partially stateful)

Dumb network (1)

- The terminal doesn‘t know anything about the network
  - No idea on the speed and bandwidth
  - No idea on the delays and round trip times
  - Absolutely nothing!

- The network is a black box

- TCP has to discover everything by itself
  - Through observing the network

Dumb network (2)

- IP packets can get lost at any time
  - Queue overflows in switches and routers
  - Bit errors or collisions on Layer 2
  - Lost link, broken line, …
  - Anything

- Lost packets are not reported!

- Packet loss comes with these properties
  - Single packet is lost
  - A whole number (burst) of packets is lost
  - Packets are reordered (B before A)
  - No packets make it through

Transmission Control Protocol

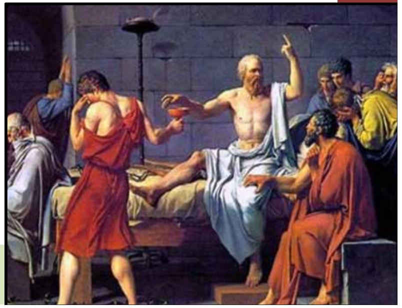
- It‘s the job of TCP to hide all these problems
  - User and application don‘t have to care
  - Avoid re-inventing the wheel for every application

- TCP hides a lot of complexity as you will find out

TCP overview

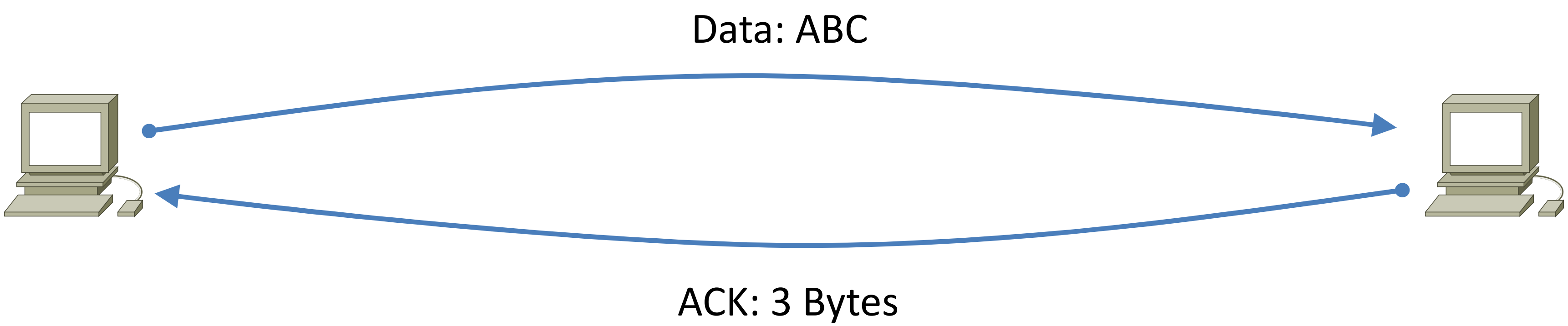
- TCP consists of a few primary mechanisms
  - Acknowledgement system
  - Loss detection system
  - Loss recovery and retransmit system
  - Bandwidth & congestion control
  - Timeouts

- More detail on each in the next slides



Acknowledgement system (1)

- The remote terminal must tell when it received data from us
- It has to send an acknowledgement („I got the data“)



Acknowledgement system (2)

- Sequence space numbering in each direction
  - So that both terminals know where they are
  - TCP header contains two fields
    - Start sequence number of this packet
    - Acknowledgement sequence number of the latest (in-order) received packet

- It takes a full RTT for us to know whether our data packet was received
  - And it takes longer to find out that it got lost

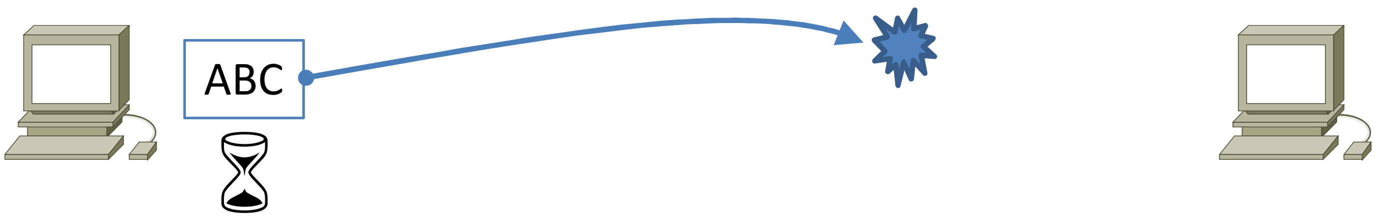
Loss detection system (1)

- How do we find out that the data packet was lost?

- Two methods exist
  - See next slides

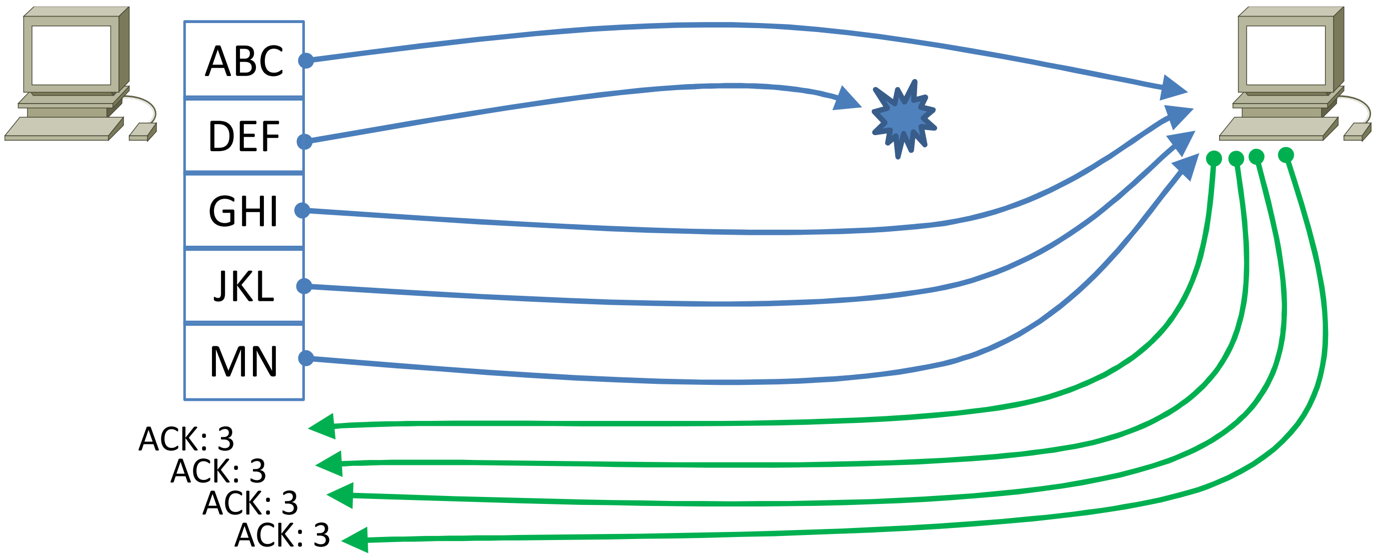
Loss detection system (2)

- Whenever we send a data packet we start a timer
  - When it expires we can assume the packet got lost
  - The data packet may have made it but the ACK got lost…
  - The timer is dynamically adjusted based on the measured RTT



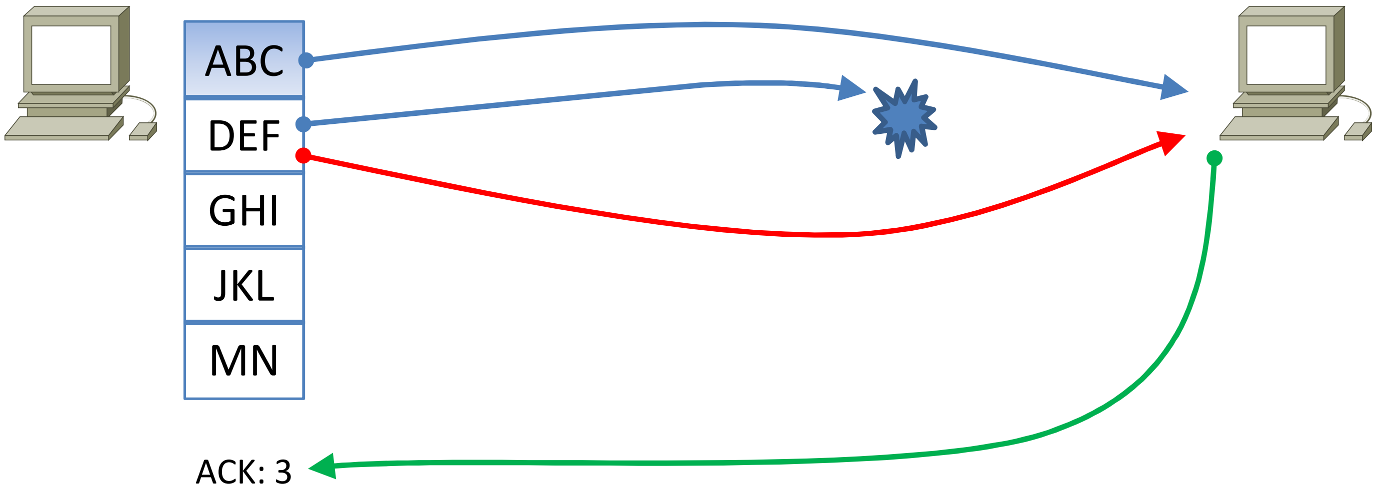
Loss detection system (3)

- Four ACK's with the same ACK number
  - We only get an ACK when a packet was received
  - We can assume the data packet at the ACK number got lost
    - May have been severe reordering as well…



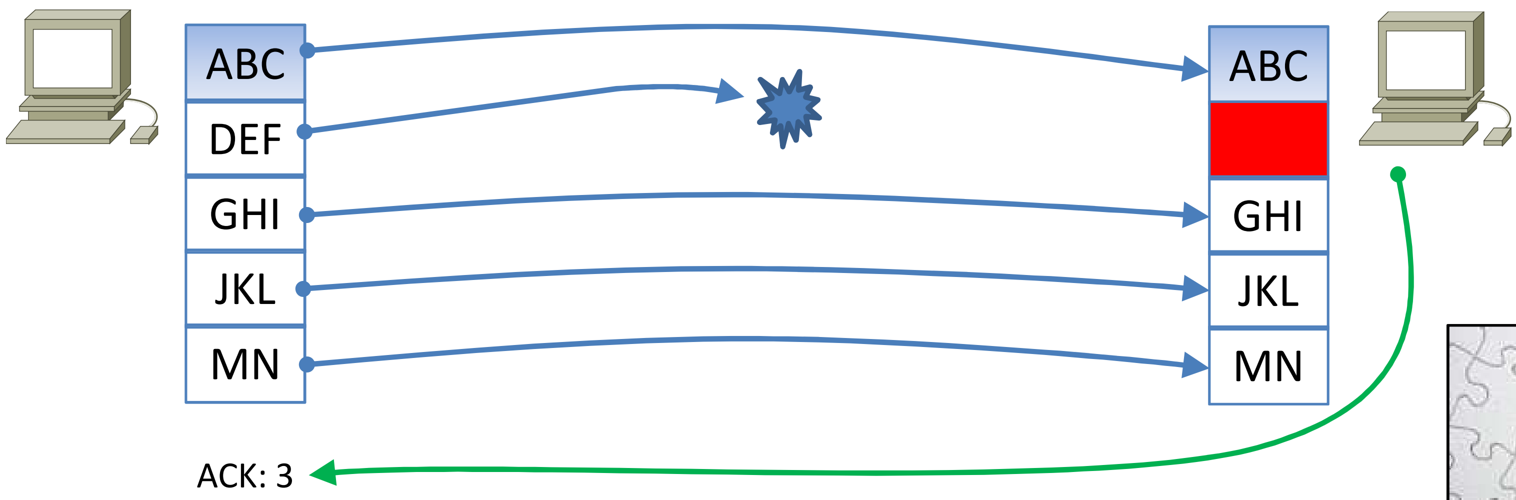
Loss recovery and retransmit system (1)

- The sender keeps a copy of the data it has sent
  - Until it is acknowledged
  - Called a send buffer
- When a data packet is lost, it can be sent again



Loss recovery and retransmit system (2)

- The receiver also has a buffer for incoming data
  - To store the data until the application reads it
  - To hold data when a packet before it got lost (or reordered)



Bandwidth & congestion control (1)

- TCP can‘t just blast out the data packets at maximum speed
  - Overflows buffers in switches and routers
  - Many packet losses
  - There are other TCP terminals too
  - No idea how fast the network is all the way to the receiver

Bandwidth & congestion control (2)

- We need something that ensures
  - Fairness for multiple TCP senders
  - Careful capacity probing
  - Conservation principle (overall efficiency)

- Measuring the ACK‘s gives two feedbacks
  - Packet loss
  - Change in RTT
  - Both are delayed feedbacks (at least 1 RTT)

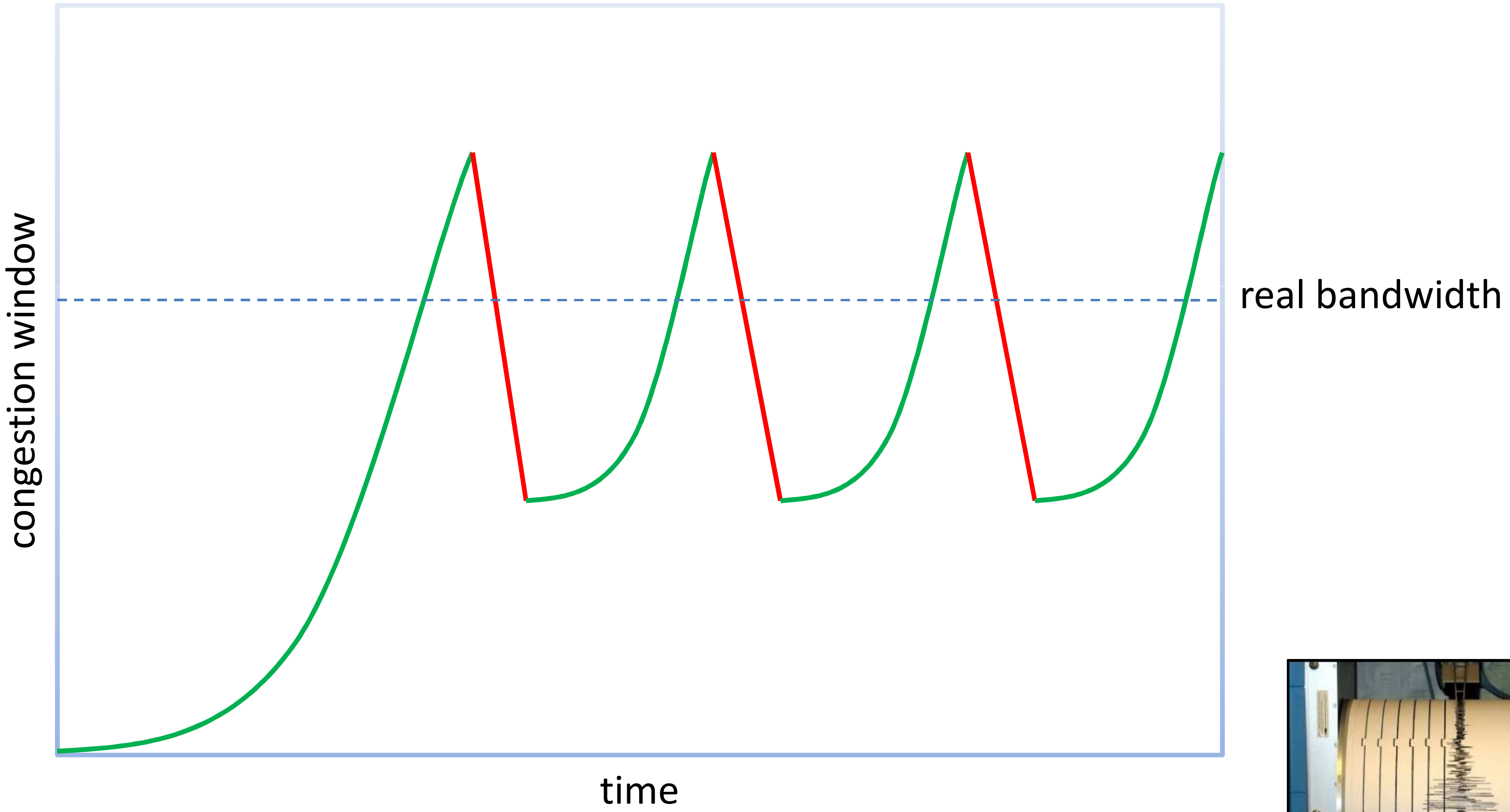
Bandwidth & congestion control (3)

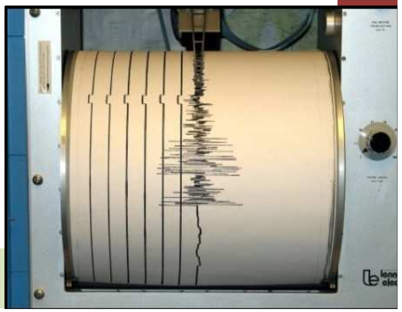
- Congestion window
  - To control how fast TCP can send new data
  - Limits the amount of unacknowledged data (inflight)

- AIMD algorithm
  - Additive increase
    - For every received ACK two new packets are sent
    - Exponential growth
  - Multiplicative decrease
    - On a lost packet the window is reduced to 50%

Bandwidth & congestion control (4)

- Graph of AIMD





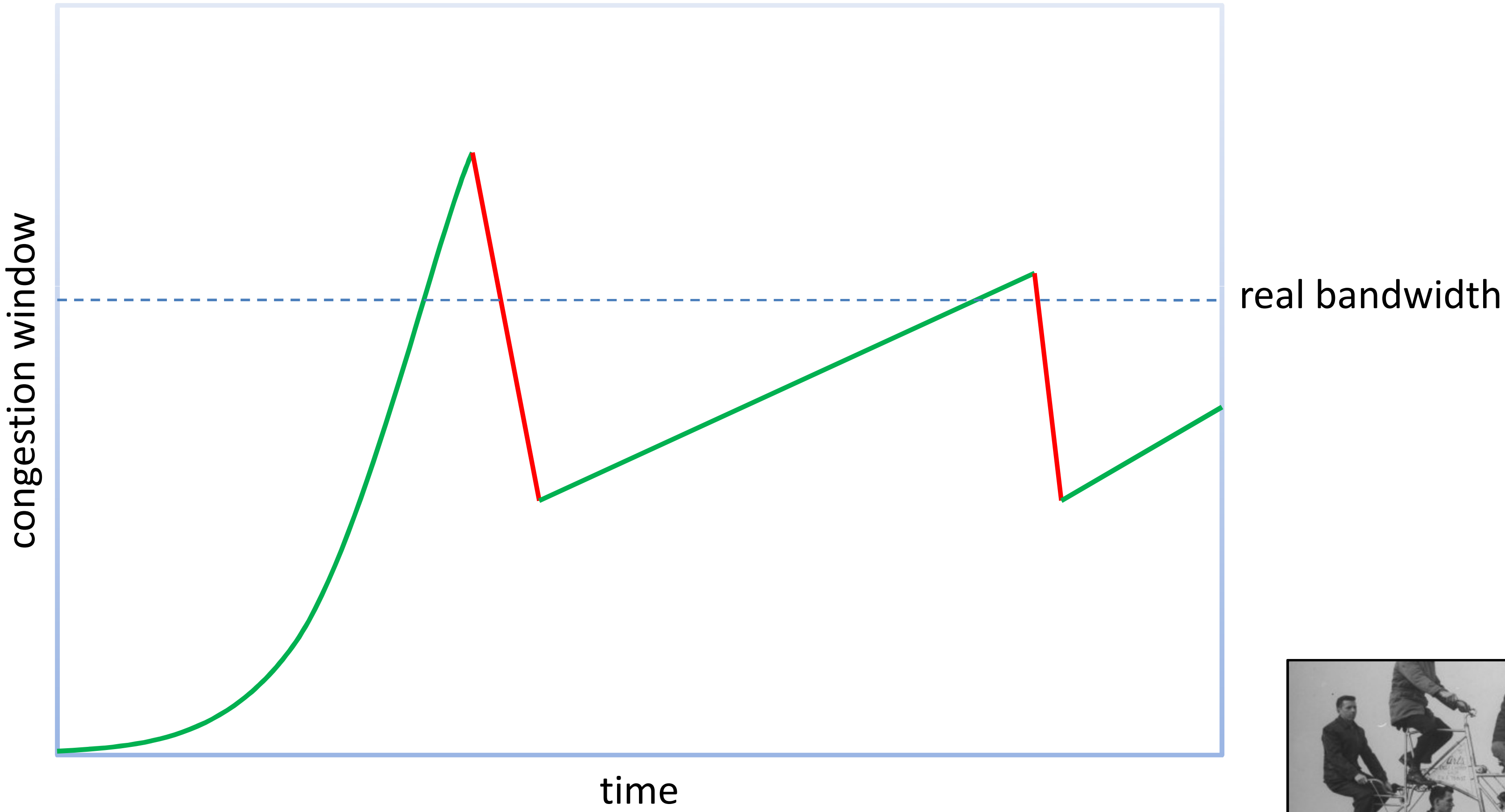
Bandwidth & congestion control (5)

- Using only AIMD is inefficient
  - Sawtooth effect
  - We want better congestion avoidance

- TCP has two send modes
  - Slow start (probing phase)
    - Additive increase
  - Congestion avoidance
    - One additional packet per full RTT

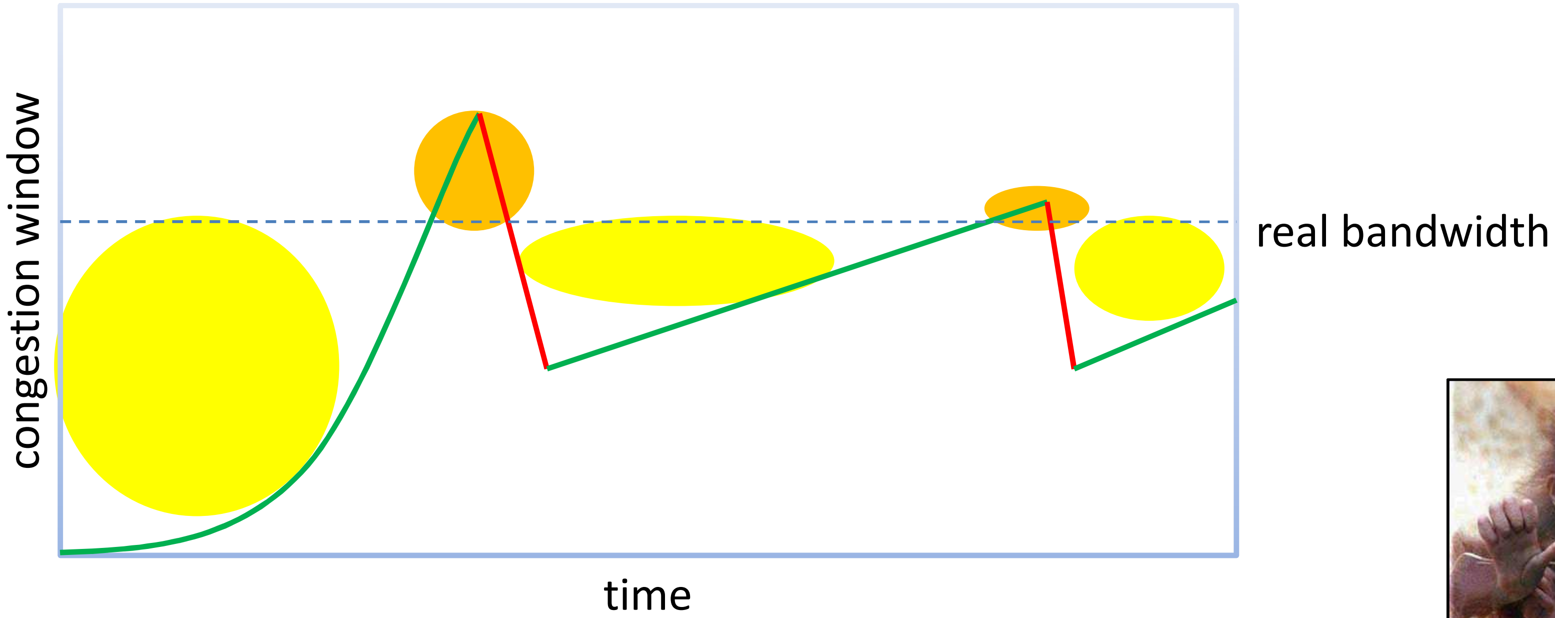
Bandwidth & congestion control (6)

- Graph of slow start and congestion avoidance



Bandwidth & congestion control (7)

- Low RTT scales much faster
  - Faster reaction times
  - Unfairness when low and high RTT transfer share the same link
- Throughput vs. goodput



Timeouts

- TCP tries to be realiable but can‘t guarantee to transfer all data
  - Network disconnect
  - Receiver crashed…

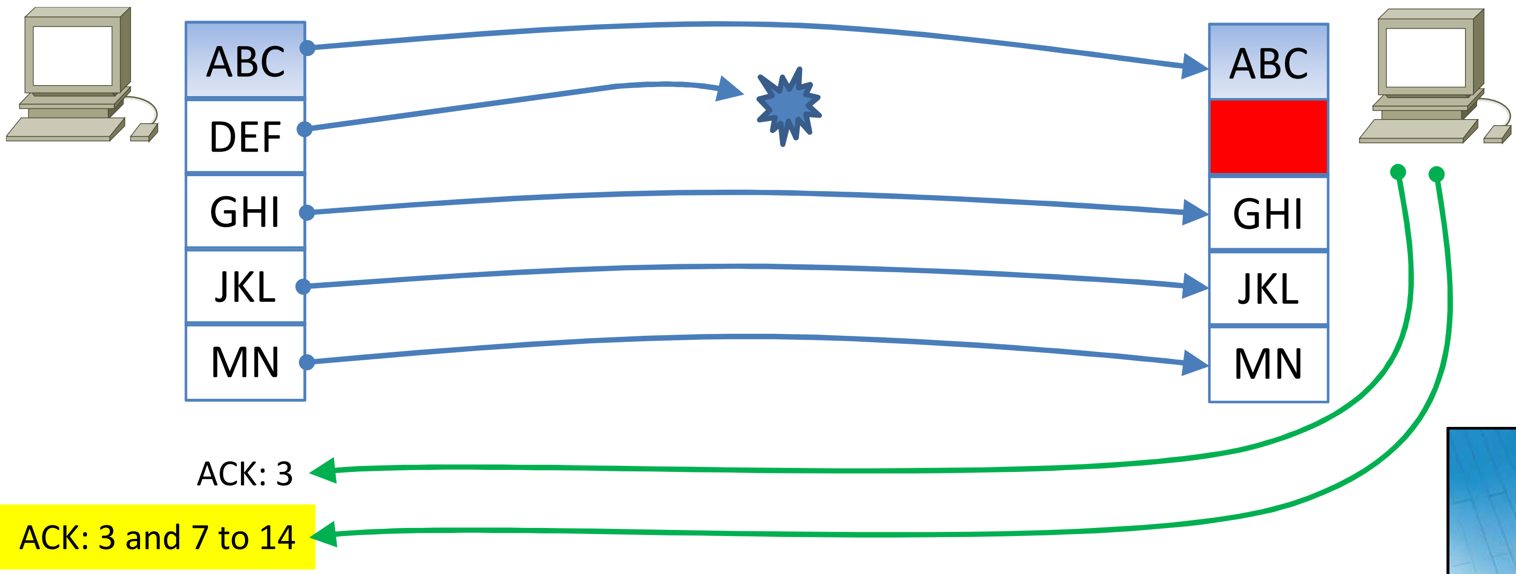
- It has to know when to give up
  - TCP tries to send the data again
  - Each time the interval increases
  - Until there is only little hope
    - After approx. 42 minutes

TCP improvements (1)

- Delayed acknowledgements
  - To reduce the ACK traffic and number of packets
- Nagle algorithm
  - Only have one packet in flight
  - For interactive applications (telnet/ssh)
- Timestamps
  - Improved RTT measurement
- SYN cookies
  - Avoid state tracking for incoming connections
- ECN
  - Explicit congestion notification (by router)

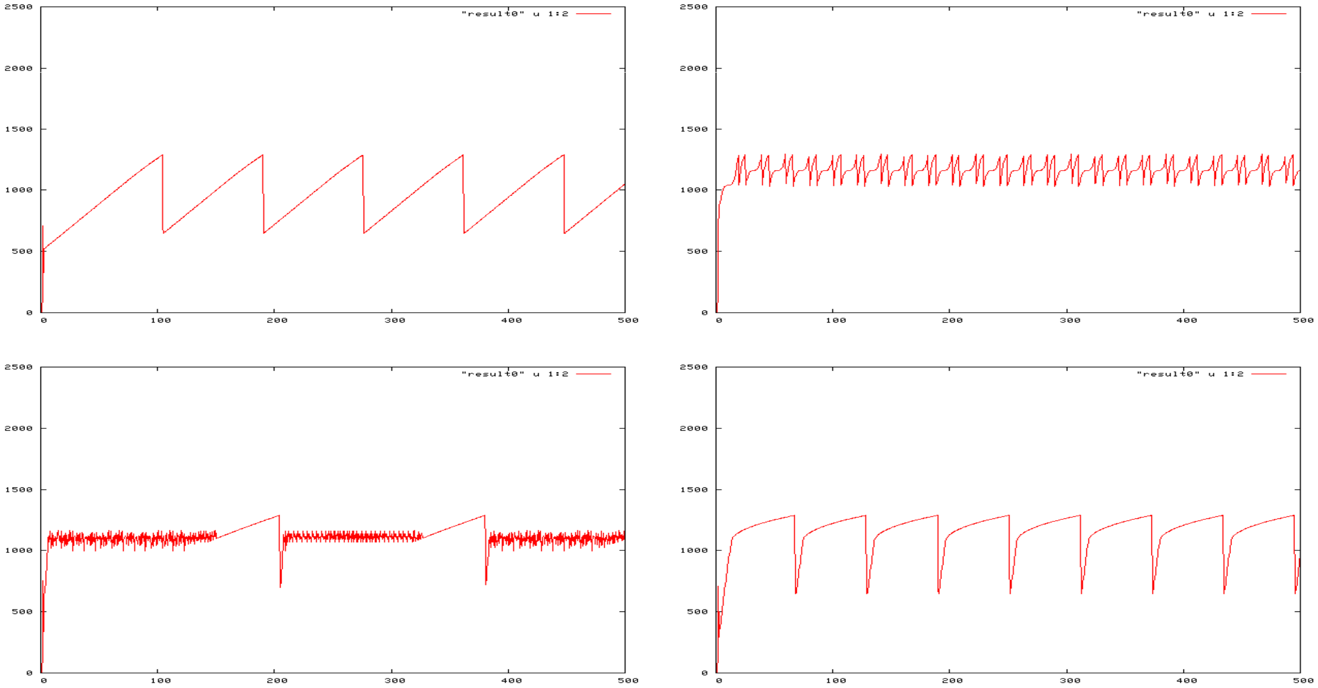
TCP improvements (2)

- SACK
  - Selective Acknowledgement
  - Reports which data is received after a lost one
  - Better loss recovery algorithms



TCP improvements (3)

- Better congestion control algorithms
  - Linux uses „CUBIC“
  - Windows 7 uses „Compound TCP“
  - Some more proposed



New Reno, CUBIC

Compound, Illinois

Tuning TCP

- Socket buffer sizing
- Enable window sizing
- Enable timestamps
- Enable SACK

Delay * Bandwidth product

- Defines how much bandwidth can be used
  - Send buffer keeps data for retransmit
    - Send buffer limits how much data can be inflight
  - Receive buffer limits how much data can be received until the application reads the data

	10ms	100ms	200ms
10Mbit	0.02MB	0.2MB	0.3MB
100Mbit	0.2MB	1.2MB	2.5MB
1Gbit	1.2MB	13MB	25MB

Tuning the network for TCP

- Active queue management
  - RED (random early detection)
    - Drop packets before the queue is full
    - Drop only one packet of any concurrent TCP connection (statistically)

- Properly sized interface buffers
  - Means large buffers
  - Delay before loss

Questions?

- Don‘t hesitate to contact me!
- Thank you for your attention
- I‘m available as a consultant and network engineer who can look at your situation in detail
    - Email: oppermann@networx.ch