UiO : Faculty of Mathematics and Natural Sciences
University of Oslo

Department of Informatics
Networks and Distributed Systems (ND) group

INF 3190

Summary lecture

Michael Welzl

Pensum: forelesninger!

- "Pensum defineres av forelesningene og gruppetimene og består av forelesningsnotatene, oppgavesettene, oppgaveregningene og annet materiale som blir tilgjengelig på nettsidene i løpet av semesteret. Det er altså ingen lærebok, men for de som er interessert kan følgende anbefales som ekstralitteratur:"

- First in list: Tanenbaum
  - What was talked about in lectures but is not covered in this book?
  - What follows are examples, it's not a complete list!

Path MTU Discovery: only old version in Tanenbaum (RFC 1191)

- (IP) fragmentation = inefficient
  - But small packets have large header overhead
- Path MTU Discovery: determine the largest packet that does not get fragmented
  - originally (RFC 1191, 1990): start large, reduce upon reception of ICMP message ➔ black hole problem if ICMP messages are filtered
  - now (RFC 4821, 2007): start small, increase as long as transport layer ACKs arrive ➔ transport protocol dependent
- Network layer function with transport layer dependencies

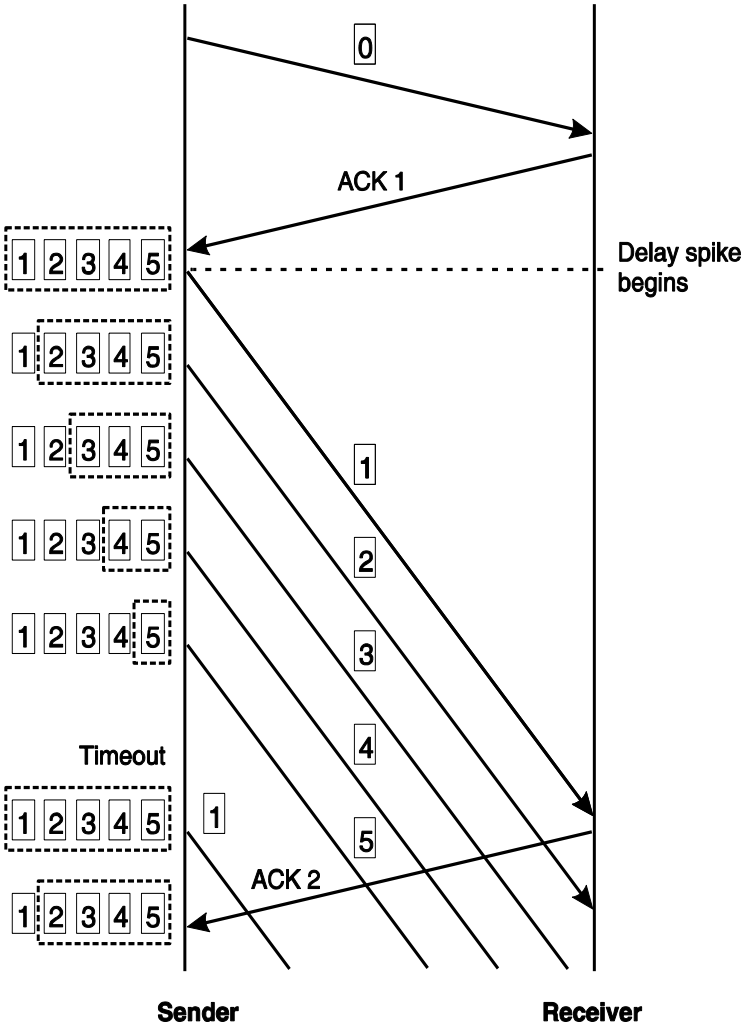
Some mechanisms in modern TCP

- e.g. NewReno only very briefly mentioned

- Appropriate Byte Counting (ABC):
  - Increasing in Congestion Avoidance mode: common implementation (e.g. Jan'05 FreeBSD code): cwnd += SMSS*SMSS/cwnd for every ACK (same as cwnd += 1/cwnd if we count segments)
    - Problem: e.g. cwnd = 2: 2 + 1/2 + 1/ (2+1/2)) = 2+0.5+0.4 = 2.9 thus, cannot send a new packet after 1 RTT
    - Worse with delayed ACKs (cwnd = 2.5)
    - Even worse with ACKs for less than 1 segment (consider 1000 1-byte ACKs) → too aggressive!
  - As name suggests, ABC counts bytes; works fine in Congestion Avoidance, somewhat limited in Slow Start (else can create "micro-bursts")

TCP: Spurious timeouts

- Possible occurrence in e.g. wireless scenarios (handover): sudden delay spike
- Can lead to timeout
  → slow start
  - But: underlying assumption: "pipe empty" is wrong! ("spurious timeout")
  - Old incoming ACK after timeout should be used to undo the error
- Several methods proposed
  Examples:
  - Eifel Algorithm: use timestamps option to check: timestamp in ACK < time of timeout?
  - DSACK: duplicate arrived
  - F-RTO: check for ACKs that shouldn't arrive after Slow Start



TCP: Appropriate Byte Counting

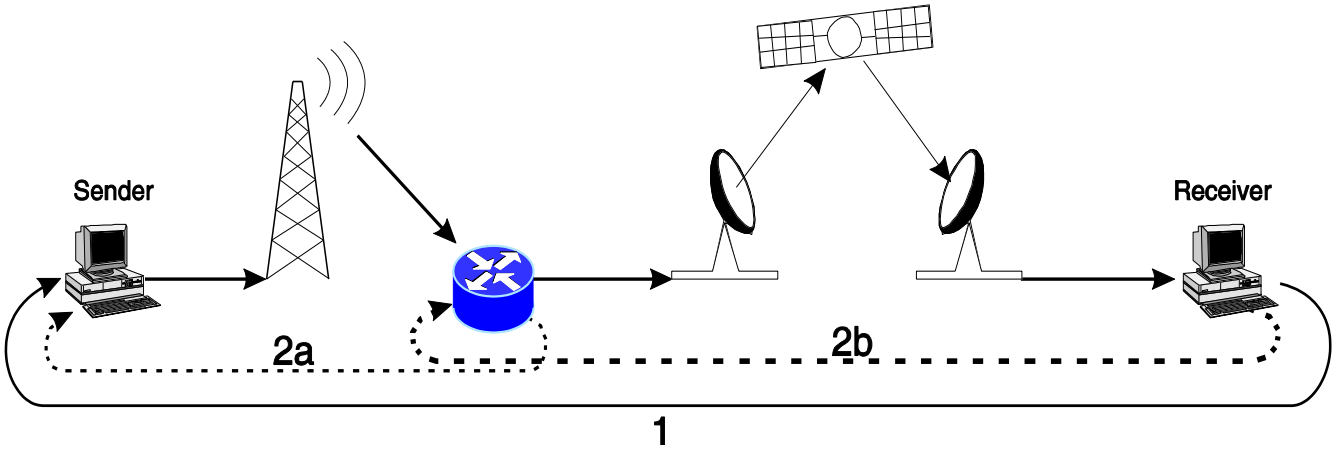
- Increasing in Congestion Avoidance mode: common implementation (e.g. Jan'05 FreeBSD code): cwnd += SMSS*SMSS/cwnd for every ACK (same as cwnd += 1/cwnd if we count segments)
  - Problem: e.g. cwnd = 2: 2 + 1/2 + 1/ (2+1/2)) = 2+0.5+0.4 = 2.9
    thus, cannot send a new packet after 1 RTT
  - Worse with delayed ACKs (cwnd = 2.5)
  - Even worse with ACKs for less than 1 segment (consider 1000 1-byte ACKs)
    → too aggressive!

- Solution: Appropriate Byte Counting (ABC)
  - Maintain bytes_acked variable; send segment when threshold exceeded
  - Works in Congestion Avoidance; but what about Slow Start?
    - Here, ABC + delayed ACKs means that the rate increases in 2*SMSS steps
    - If a series of ACKs are dropped, this could be a significant burst ("micro-burstiness"); thus, limit of 2*SMSS per ACK recommended

TCP over Satellite and PEPs

- Satellites combine several problems
  - Long delay
  - High capacity
  - Wireless (but usually not noisy (for TCP) because of link layer FEC)
  - Can be asymmetric (e.g. direct satellite downlink, 56k modem uplink)

- Thus, TCP over satellite is a major research topic
  - Transparent improvements ("Performance Enhancing Proxies") common
  - Figure: split connection approach: 2a / 2b instead of control loop 1
  - Many possibilities - e.g. Snoop TCP: monitor + buffer; in case of loss, suppress DupACKs and retransmit from local buffer



MPTCP

- Many hosts are nowadays multihomed
  - Smartphones (WiFi + 3G), data centers
  - Why not use both connections at once?

- Cannot know where bottleneck is
  - If it is shared by the two connections, they should appear (be as aggressive) as only one connection
  - MPTCP changes congestion avoidance “increase” parameter to “divide” aggression accordingly
    - but instead of being “½ TCP”, tries to send as much as possible over least congested path
    - Least congested = largest window achieved; hence, increase in proportion to window

Some mechanisms in modern browsers

- WebRTC / rtcweb
  - Direct UDP communication between browsers (p2p)
    - Better latency & bandwidth, important for interactive communication (video, audio, online games, …)
    - Data channel (e.g. control data in games, file transfers, ..) uses SCTP in userspace (in browser)
    - Between browsers, there are many middleboxes; several tricks played (ICE / STUN / TURN protocols)
  - Javascript API lets web designer control “peer connections”
  - Congestion control to be defined; requirements:
    - Avoid queue: react to delay, yet interoperate with TCP
    - Detect shared bottlenecks, combine controls of flows
- SPDY / HTTP/2.0
  - Universal encryption, header compression, multi-streaming, ..

Application Level Framing (ALF)

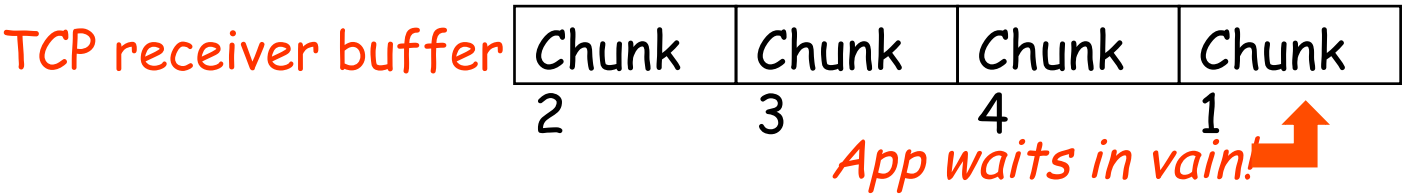
- Concept applied in RTP and SCTP
  - Byte stream (TCP) inefficient when packets are lost
  - Application may want logical data units (“chunks“)

- ALF: app chooses packet size = chunk size
  packet 2 lost: no unnecessary data in packet 1,
  use chunks 3 and 4 before retrans. 2 arrives

- 1 ADU (Application Data Unit) = multiple chunks
  => ALF still more efficient!

Unordered delivery & multistreaming Concept applied in SCTP, SPDY

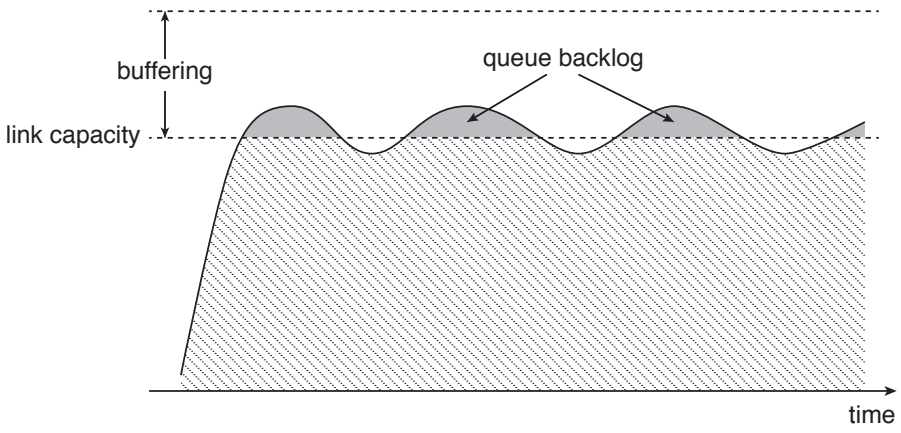
- Decoupling of reliable and ordered delivery
  - Unordered delivery: eliminate Head-Of-Line (HOL) blocking delay



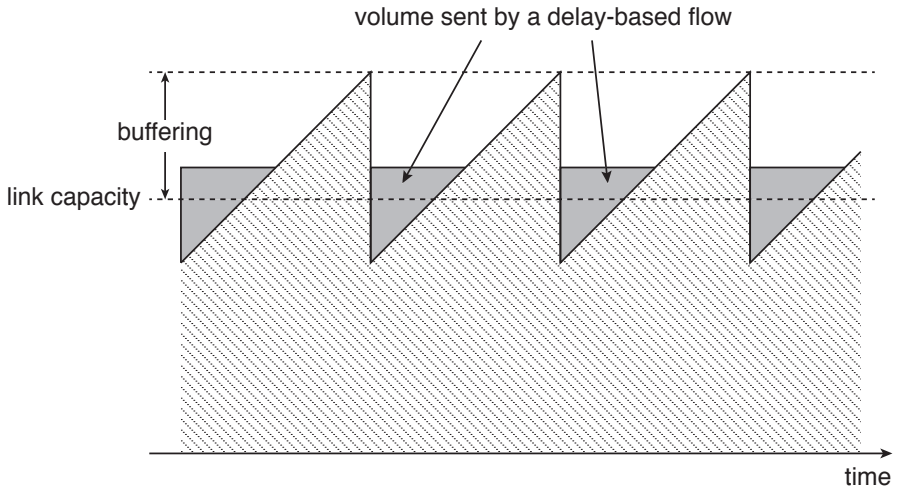
- Support for multiple data streams (per-stream ordered delivery)
  - Stream sequence number (SSN) preserves order *within* streams
  - no order preserved *between* streams

LEDBAT

- Try to send when others don't
  - For low-priority traffic that should not get in the way of other applications
  - Growing (assumption: queuing) delay = early congestion signal
  - Possible benefit: low delay
  - Encapsulation (how to embed in existing protocols) not (yet?) defined; implemented over UDP in BitTorrent



(a) Sending rate of a delay-based flow: ideal case.



(b) Coexistence of an LBE flow with a non-LBE one.