



UiO : **Faculty of Mathematics and Natural Sciences**

University of Oslo



Department of Informatics

Networks and Distributed Systems (ND) group

INF 3190

New Internet Standards



Michael Welzl

Overview: a biased sample...

- Based on: “what goes on ‘under the hood’ today/tomorrow, with the things many of us are using?”
 - but also related to own research



- ... and some not-so-novel background needed!

Stream Control Transmission Protocol (SCTP)

- TCP, UDP do not satisfy all application needs
- SCTP evolved from, and is used for, IP telephony signaling
 - Like TCP: reliable, full-duplex connections
 - Unlike TCP and UDP: new delivery options that are particularly desirable for telephony signaling and multimedia applications
- TCP + features
 - Congestion control similar; some optional TCP mechanisms mandatory
 - Two basic types of enhancements: 1) Performance; 2) Robustness

SCTP services: SoA TCP + extras

Services/Features	SCTP	TCP	UDP
• Full-duplex data transmission	yes	yes	yes
• Connection-oriented	yes	yes	no
• Reliable data transfer	yes	yes	no
• Unreliable data transfer	yes	no	yes
• Partially reliable data transfer	yes	no	no
• Ordered data delivery	yes	yes	no
• Unordered data delivery	yes	no	yes
• Flow and Congestion Control	yes	yes	no
• ECN support	yes	yes	no
• Selective acks	yes	yes	no
• Preservation of message boundaries (ALF)	yes	no	yes
• PMTUD	yes	yes	no
• Application data fragmentation	yes	yes	no
• Multistreaming	yes	no	no
• Multihoming	yes	no	no
• Protection against SYN flooding attack	yes	no	n/a

Packet format

- Unlike TCP, SCTP provides message-oriented data delivery service
 - key enabler for performance enhancements
- **Common header**; three basic functions:
 - Source and destination ports together with the IP addresses
 - Verification tag
 - Checksum: **CRC-32 instead of Adler-32**
- followed by **one or more chunks**
 - chunk header that identifies length, type, and any special flags
 - concatenated building blocks containing either control or data information
 - control chunks transfer information needed for association (connection) functionality and data chunks carry application layer data.
 - Current spec: 14 different Control Chunks for association establishment, termination, ACK, destination failure recovery, ECN, and error reporting
- Packet can contain several different chunk types
- SCTP is extensible

Application Level Framing (ALF)

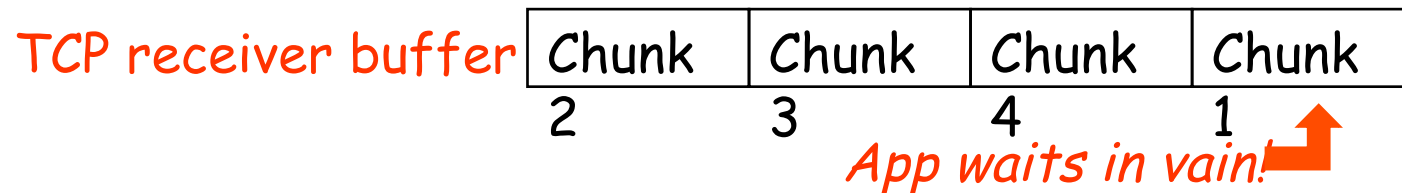
- Concept applied in RTP and SCTP
 - Byte stream (TCP) inefficient when packets are lost
 - Application may want logical data units (“chunks”)



- 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

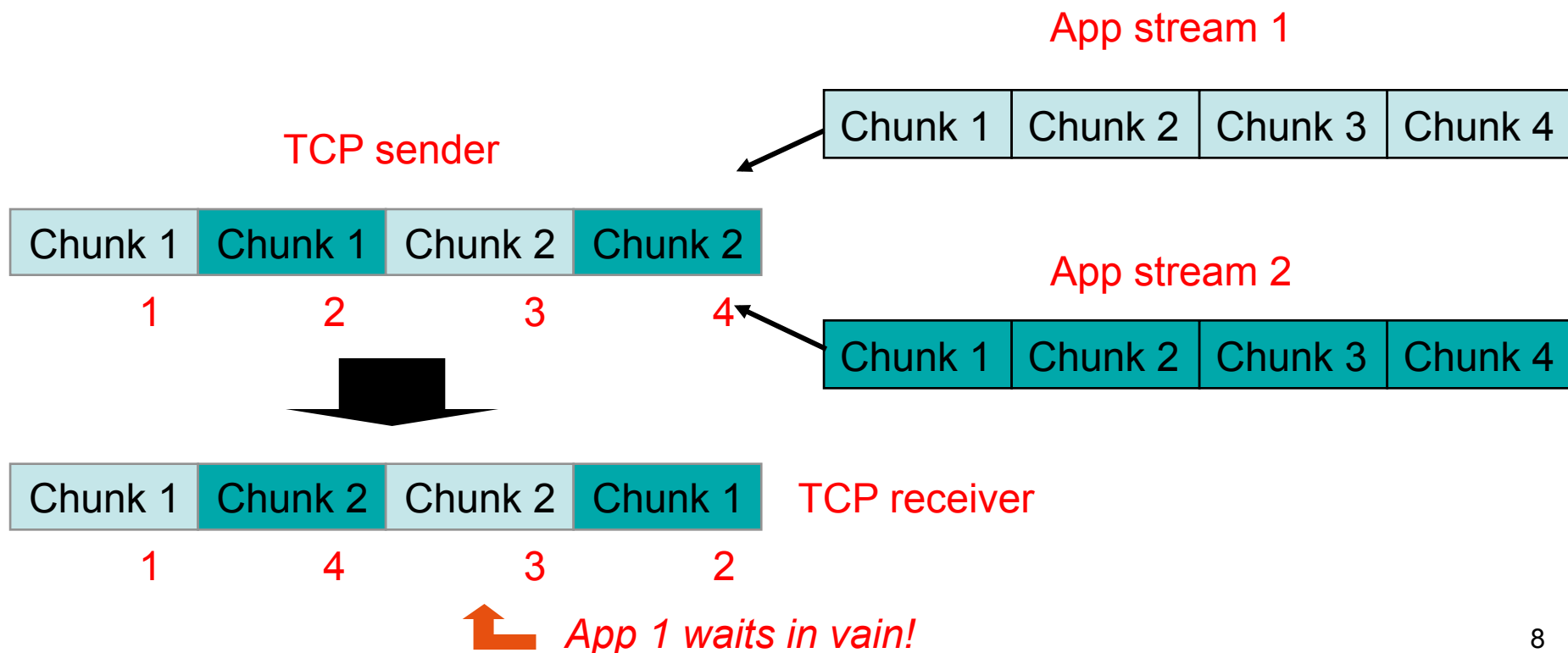
- Decoupling of reliable and ordered delivery
 - Unordered delivery: eliminate **Head-Of-Line (HOL)** blocking delay



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

Multiple Data Streams

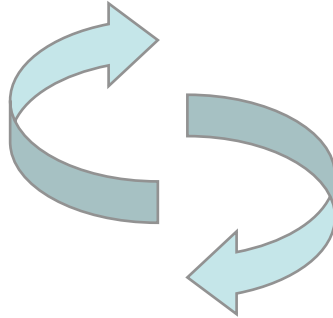
- Application may use multiple logical data streams
 - e.g. pictures in a web browser
- Common solution: multiple TCP connections
 - separate flow / congestion control, overhead (connection setup/teardown, ..)



Multihoming

- ...at transport layer! (i.e. transparent for apps, such as FTP)
- TCP connection \Leftrightarrow SCTP association
 - 2 IP addresses, 2 port numbers \Leftrightarrow 2 sets of IP addresses, 2 port numbers
- Primary goal: robustness
 - automatically switch hosts upon failure
 - eliminates effect of long routing reconvergence time
 - Now also CMT (Concurrent Multipath Transport) as with MPTCP; more later...
- TCP: no “keepalive” messages when connection idle
- SCTP monitors reachability via ACKs of data chunks and heartbeat chunks

The Internet-deployment vicious circle



Application developer:

- wants to max. revenue
- Use new protocol: effort (-\$) unless the protocol works everywhere (maybe ++\$)

OS developer, middlebox designer / maintainer:

- wants to max. revenue
- Support new protocol alone: effort or risk (-\$) unless the protocol is beneficially used by applications (maybe ++\$)

Consequences of this vicious circle

- Efforts to make everything look like standard TCP
 - MPTCP, Minion (partially in Apple)
- Efforts to build whole protocols over UDP
 - LEDBAT (BitTorrent), QUIC (Google), RTMFP (Adobe), Skype (proprietary), etc.
- Possible solution: change interface to transport layer (let applications specify service, not protocol)
 - ongoing efforts in the IETF (TAPS WG)

Google's SPDY

(basis for HTTP/2.0, many similarities)

- Goal: reduce page load time
- Reuses HTTP semantics
 - But changes how data is written to the network (e.g. no ascii protocol!)
 - retains all features including cookies, Content-Encoding negotiations etc.
- Universal encryption
 - SPDY is negotiated over SSL/TLS, thus operates exclusively over a secure channel

SPDY /2

- Header compression
- Server Push/Hint
 - Servers could proactively push resources to clients (e.g. scripts and images that will be required)
 - Or can send hints advising clients to pre-fetch content
- Content prioritization
 - Client can specify the preferred order in which resources should be transferred

SPDY Multiplexing

- Multiplexing

- Persistent connection as in HTTP/1.1
Reason: allow TCP to increase its window
(most web flows terminate in slow start)
- But: in HTTP/1.1, sequence determined by client
- Client does not know which requests take long
(e.g. database lookups, ..); can cause **HOL delay !**
- SPDY multiplexes frames onto the TCP connection
 - TCP can still cause RTT-timescale HOL blocking delay
 - Google's solution: **QUIC / UDP...**

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 “ $\frac{1}{2}$ TCP”, tries to send as much as possible over least congested path
 - Least congested = largest window achieved; hence, increase in proportion to window

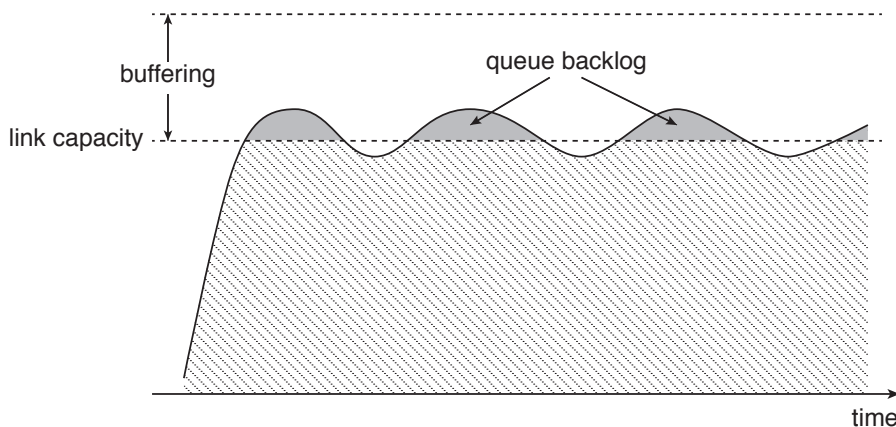


MPTCP /2

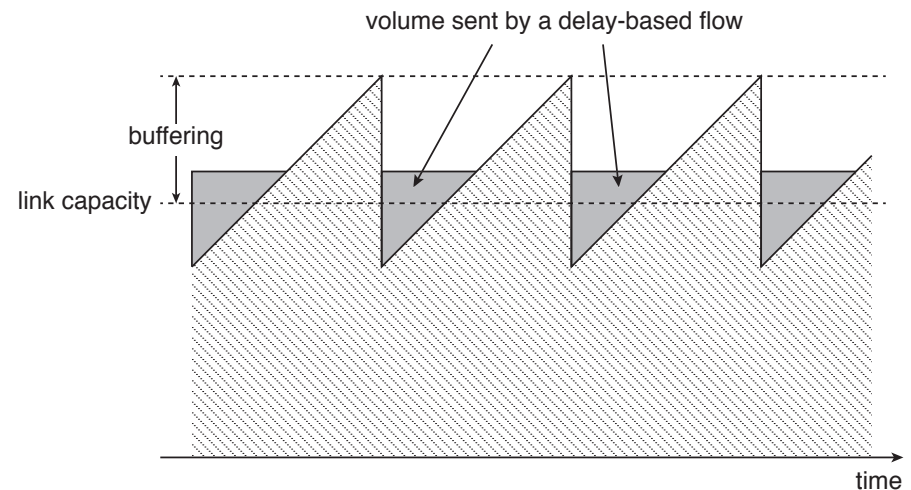
- Moving traffic away from congested links achieves “resource pooling”
 - A web server connected to two 100 Mbit/s links behaves roughly as if it had one 200 Mbit/s link
 - Only one host needs to be multi-homed
- Issues
 - Must look like TCP to work everywhere
Minimal on-the-wire changes: new TCP option
 - Parallel paths can cause reordering → delay in handing over data to application on receiver side

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.

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