NOKIA
Connecting People

Advanced Transport Protocols

Computer Networks II course

Helsinki University of Technology

September 24, 2007

Pasi Sarolahti

Nokia Research Center

Outline

- Transport layer functions
- Congestion Control
- TCP Improvements
- Stream Control Transmission Protocol (SCTP)
- Datagram Congestion Control Protocol (DCCP)
- IETF Standardization

Transport layer

[Comer2000]: “The primary duty of the transport layer is to provide communication from one application program to another. Such communication is often called end-to-end. Transport layer MAY regulate flow of information. It MAY also provide reliable transport, ensuring that data arrives without error and in sequence. To do so, transport protocol software divides the stream of data being transmitted to small pieces and passes each packet along with a destination address to the next layer for transmission”

Congestion Control

Congestion Control

- TCP congestion control principles introduced in late 1980s [Jacobson88]
  - Not part of the original transport layer functionality
- Important design factor in today's Internet protocol development
  - Requirement for new transport layer work
- Principles discussed in RFC 2914
  - Preventing congestion collapse
  - Fairness

Congestion Control

Preventing Congestion Collapse

- Collapses in the 1980s motivated creation of congestion control algorithms
- Transmission rate must be reduced when congestion is detected
  - Responsibility of transport layer, i.e., the sending end host
- Packet loss is assumed to be congestion signal
  - No deployed explicit congestion notification scheme
- At most one congestion action / round-trip time
  - Burst of packet losses can be indication of same congestion situation

Congestion Control

Fairness to TCP

- **Transport implementations must be fair to other flows**
  - **Transmission rate should be roughly similar to that of TCP**
- **Components of TCP-friendly congestion control**
  - **Slow-start**
  - **Additive Increase, Multiplicative Decrease (AIMD)**
  - **Retransmission timers relative to round-trip time**
- **Theoretic equation to define upper bound for TCP sending rate [Padhye98, Floyd00]:**

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}(3\sqrt{\frac{3p}{8}})p(1+32p^2)}$$

T = rate (usually in Bps)
s = packet size
R = round-trip time
p = packet loss rate
t_{RTO} = retransmit timeout

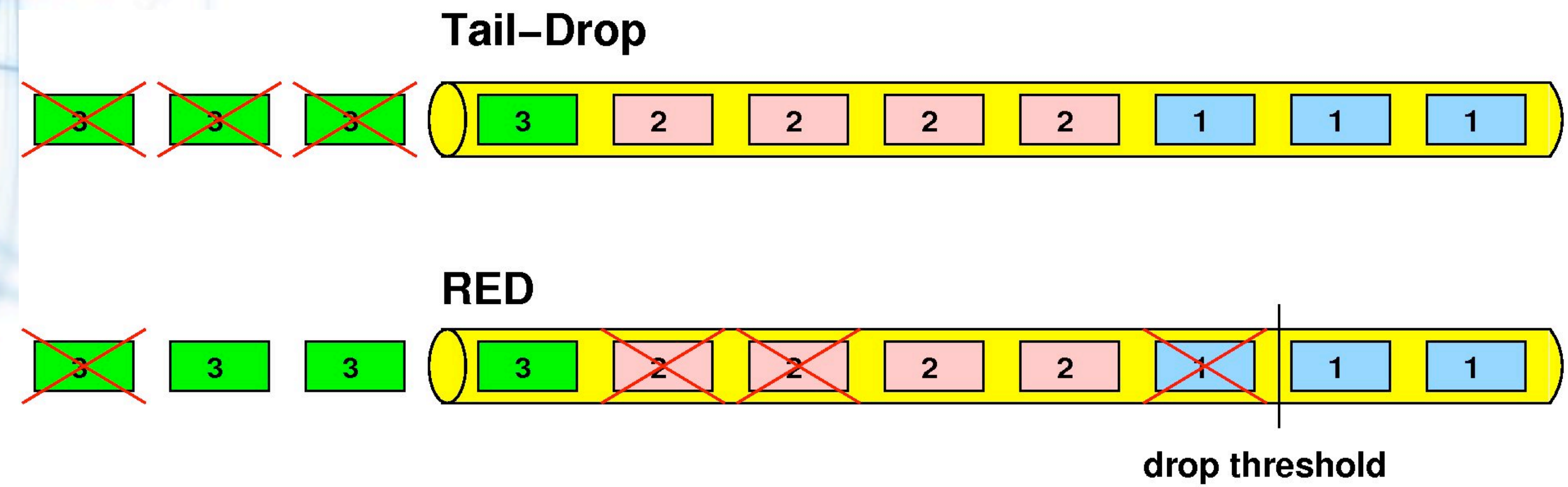
Congestion Control

Active Queue Management (AQM)

- Simple router implementation drops packet when queue is full
  - Lock-out: Sometimes few flows get to dominate most of queue space
  - Queue delay: Long packet queues increase transmission delays
- Active Queue Management marks packets before queue is full
- Random Early Detection (RED) [Floyd93]
  - Mark a packet at probability P when queue length is more than L
    - Typically P is a function of current queue length
  - Marks are distributed more evenly between flows
  - Average queue length remains shorter
- Without Explicit Congestion Notification “mark” == drop packet

Congestion Control

Active Queue Management (AQM)



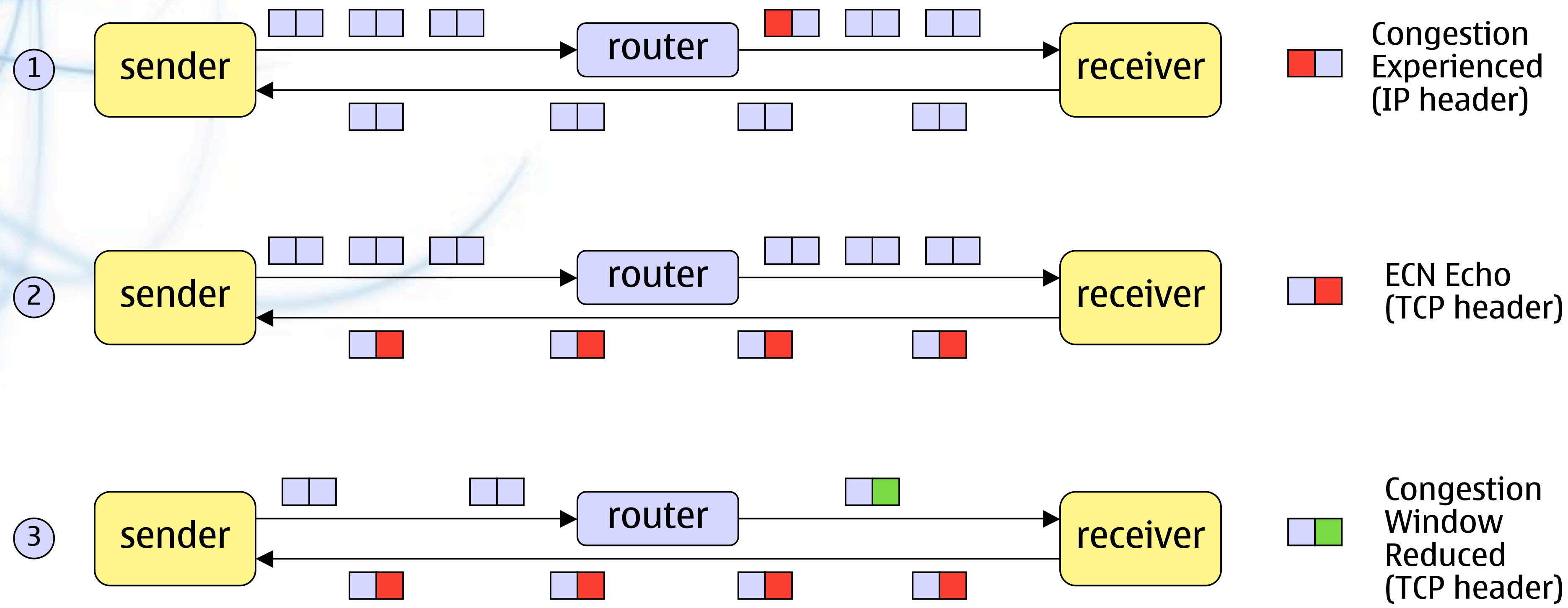
Congestion Control

Explicit Congestion Notification (ECN)

- Sender marks a bit in IP header if transport is ECN capable
- Routers to indicate congestion with a *congestion bit* in *IP header*
  - Used with Active Queue Management
  - Reduces the number of packet losses
- Transport layer receiver echoes congestion notification to sender
  - In transport header
- When receiving notification, sender reduces its transmission rate
- Implemented in many end-hosts, but not too many routers
  - Problem: Some devices in network drop IP packets with ECN bits
- More information in [Floyd94, RFC 3168]

Congestion Control

Explicit Congestion Notification (ECN)



TCP Improvements

Transmission Control Protocol (TCP)

- Reliable
  - Cumulative acknowledgements
  - Fast retransmit / fast recovery
    - Reno [RFC 2581], NewReno [RFC 3782]
  - Retransmission timeouts [RFC 2988]
- Stream-oriented
  - no concept of datagram boundaries
  - ideal for transferring files
  - transferring series of structured messages more difficult

TCP Improvements
Important concepts

- **Congestion Window:** TCP sender variable that tracks how many bytes/packets are allowed to be in transit in the network at a time. Adjusted following the slow-start or congestion avoidance algorithms
- **Round-trip time:** Time from sending packet to receiving acknowledgement for it
- **Retransmission timeout:** Triggers retransmission of unacknowledged segments in slow-start. Timer is reset every time a new acknowledgement arrives at sender
- **Duplicate Acknowledgment:** Triggered by receiver on receiving out of order segment. Potential indication of packet loss

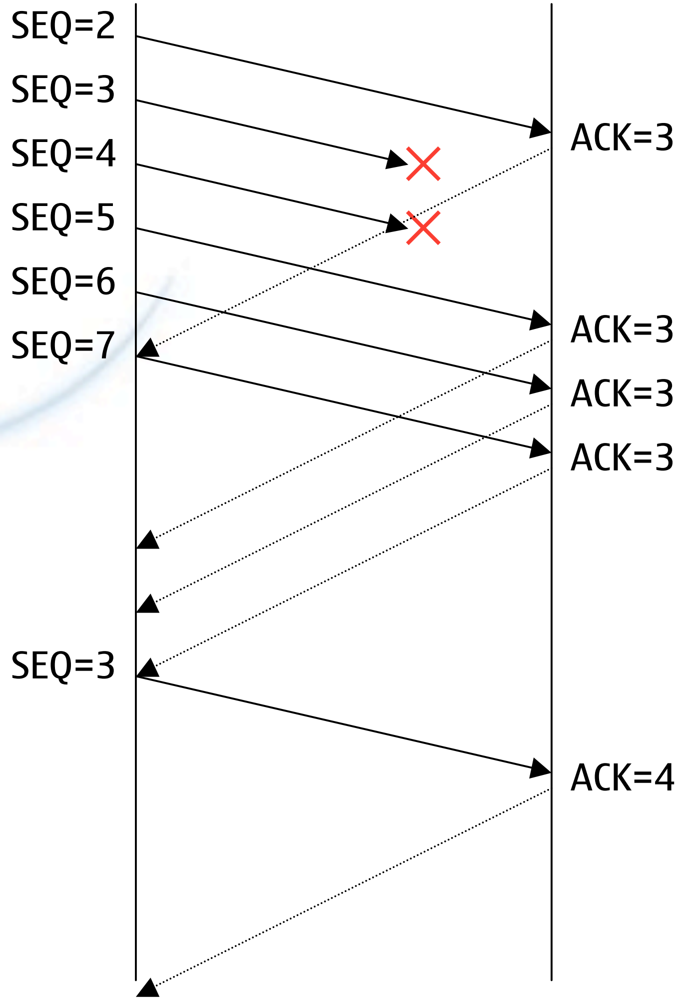
TCP Improvements

Retransmission Mechanism

- TCP receiver acknowledges next sequence number it expects to receive
  - If receiver gets packet out of order it acknowledges same sequence number than earlier
- When sender receives 3 duplicate acknowledgements it considers the first unacknowledged segment lost
  - Congestion response: reduce the congestion window by half
  - Retransmit the first unacknowledged segment
- If no acknowledgements arrive for time *RTO*, sender retransmits the first unacknowledged segment
  - Congestion window is set to 1 segment

TCP Improvements

Retransmission Mechanism



TCP Improvements

Specific Problems with Basic TCP

- Minimal information from cumulative acknowledgements
  - Problems in environments with frequent packet losses (wireless)
- Small window and packet retransmissions
  - May prevent fast retransmit from working
- Retransmission ambiguity -- is ACK for original or retransmit?
  - Hinders the round-trip time measurement
- Unnecessary retransmissions
  - Unnecessary use of bandwidth (sometimes expensive in wireless)

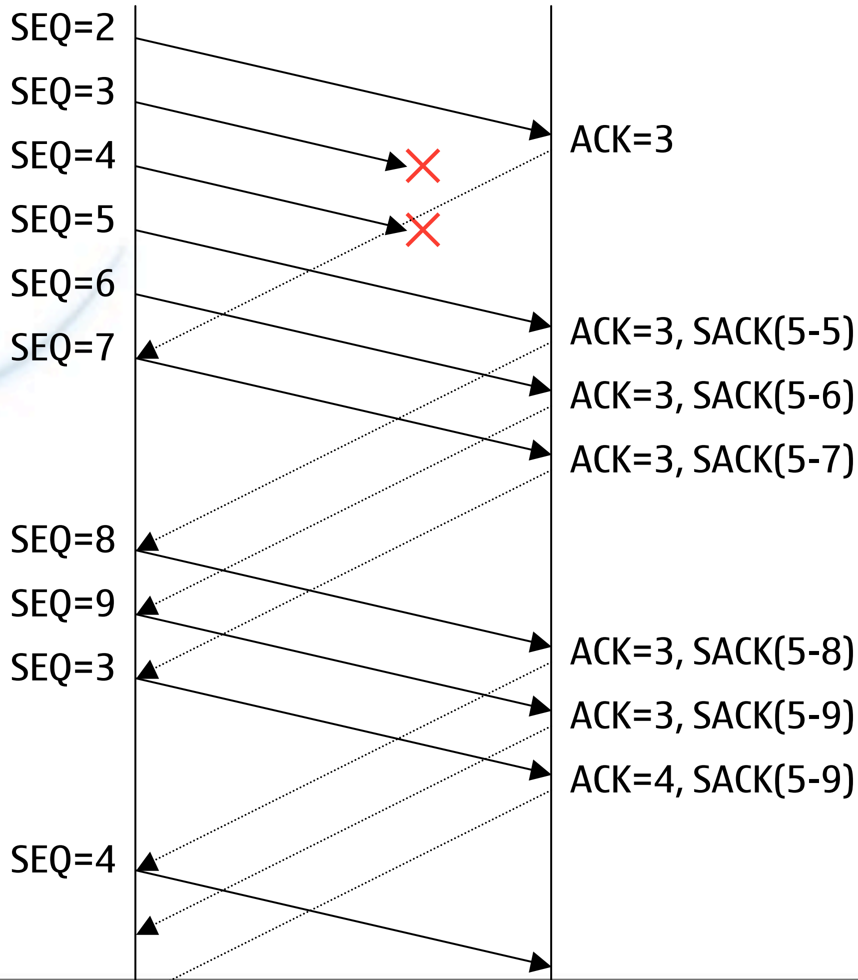
Selective Acknowledgements (SACK)

Problem: Minimal information from cumulative acknowledgements

- Additional information about “holes” in sequence number space
- TCP option that reports discontinuous blocks of received data
- Sender gets better information about which segments are lost
  - Allows more efficient retransmissions
    - Without SACK sender can retransmit only one segment in round-trip time
    - With SACK more retransmissions can be made in a round-trip time
  - Allows more efficient tracking of number of outstanding segments
- SACK option specified in RFC 2018
- SACK-based retransmission algorithm specified in RFC 3517

TCP Improvements

Selective Acknowledgements (SACK)



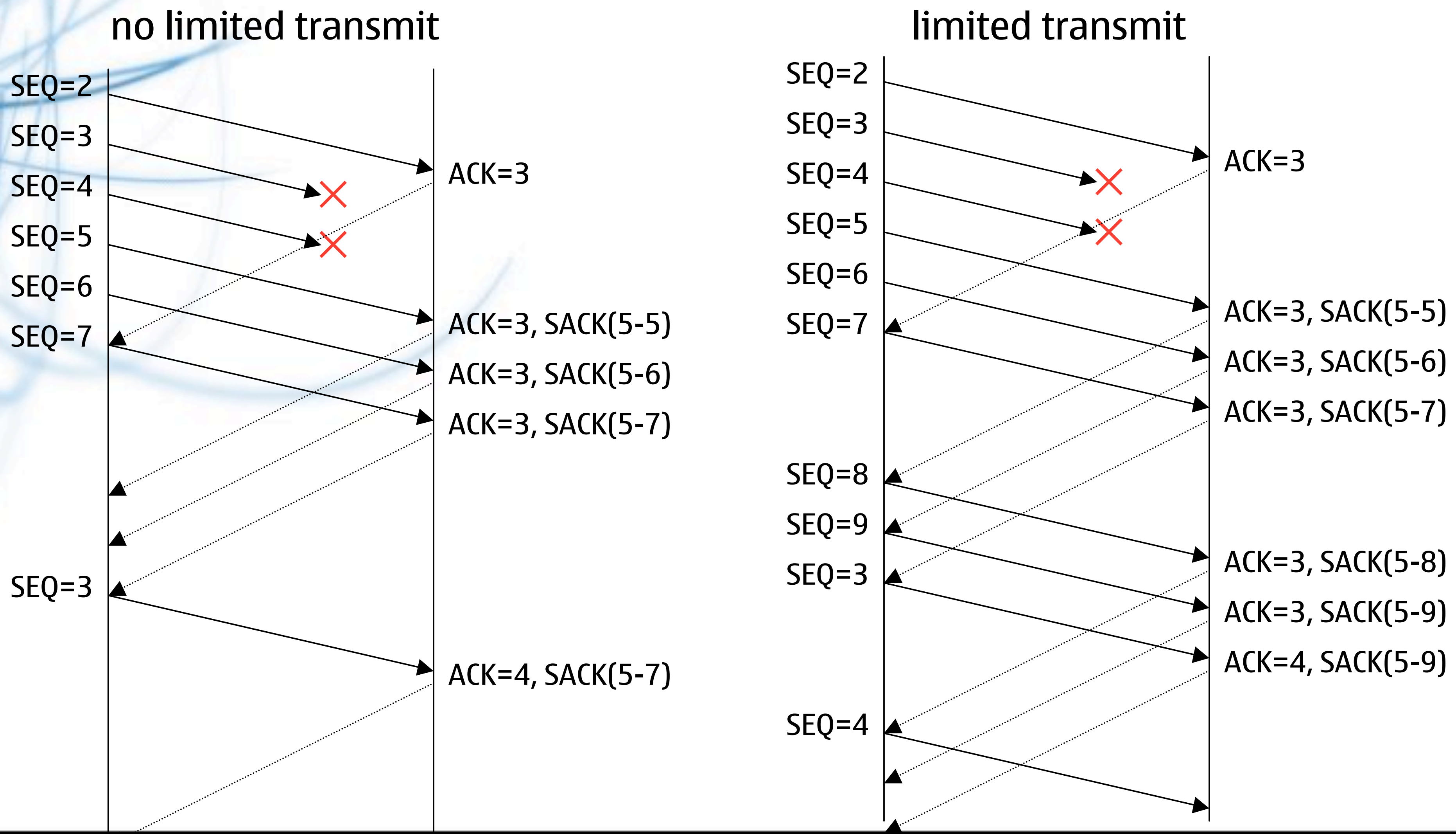
Limited Transmit & Early Retransmit

Problem: Small window and packet retransmissions

- **Fast retransmit requires congestion window of 4 or more**
  - **To allow three duplicate acknowledgements (DupAcks)**
- **Otherwise sender must wait for timeout to retransmit**
  - **Typically takes much longer than fast retransmit**
- **Limited transmit allows sending new data on first two DupAcks [RFC 3042]**
  - **To trigger more duplicate ACKs**
- **Early retransmit allows retransmitting on 1st or 2nd DupAck [draft-allman-tcp-early-rexmt]**
  - **Condition: there are less than 4 outstanding segments**

TCP Improvements

Limited Transmit



TCP Improvements

Timestamps

Problem: Retransmission ambiguity

- Specified in RFC 1323
- TCP option for sender to include timestamp in every packet
- TCP receiver echoes the timestamp back to sender
- Retransmissions have different timestamp than original
  - Allows round-trip time measurement for retransmitted segments
    - Not allowed without timestamps
  - Allows detection of spurious retransmissions [Ludwig00]
- Allows protection against wrapped sequence numbers

TCP Improvements

Detecting Spurious RTOs

Problem: Unnecessary retransmissions

- **Delay spikes on wireless links**
  - **Layer 2 retransmissions**
  - **Hand-off procedures**
- **Unnecessary retransmission timeout has bad effects**
  - **Unnecessary retransmission of several segments**
  - **False slow-start**
- **Eifel: Use TCP timestamps to detect unnecessary retransmission [RFC 4015, Ludwig00]**
- **F-RTO: Detect spurious RTO by changing the sending pattern [RFC 4138, Sarolahti03]**

TCP Improvements
Other TCP Variants

- Binary Increase (BIC) [Xu04]
  - Binary search between maximum and minimum window size
- Vegas [Brakmo95]
  - Congestion avoidance based on delay measurements, etc.
- Westwood [Mascolo01]
  - Estimate bandwidth based on incoming ACKs
- Compound TCP [Tan06]
  - Add a delay-based component to standard congestion window
- Many more...

TCP Improvements
Concluding Remarks

- TCP is not a simple protocol today
  - Number of incremental patches and improvements
- TCP is not perfect but worthy improvements are hard to invent
  - Cost: added complexity
    - more difficult to test
    - possibility of introducing new defects
- Today's protocol stacks include many of the state-of-the art improvements
  - For example, SACK is rather common

Stream Control Transmission Protocol (SCTP)

Stream Control Transmission Protocol (SCTP)

- Specified in RFC 2960
- Reliable
  - Similar retransmission algorithms than with TCP
  - Similar congestion control with TCP
  - SACK supported by default
- Additional features to TCP
  - Preserve message boundaries
  - Support for multiple "streams" in single association (=connection)
  - Support for multi-homing
    - Several IP addresses for an end-host

Stream Control Transmission Protocol (SCTP)

Packet Structure

- **SCTP packet starts with common generic header**
  - **Source and destination ports**
  - **Verification tag**
  - **Checksum**
- **SCTP packets are built from modular units called "*chunks*"**
  - **Type-Length-Value coded**
  - **No strict ordering**
  - **Extensible for new features**
  - **Several chunks can be bundled in single packet**
- **Chunks can consist of number of options**

Stream Control Transmission Protocol (SCTP)

Packet Structure

Note: Packet length limited by Max. Transmit Unit length

TCP	Length
IP Header	*20 bytes*
TCP Header	*20 bytes*
TCP Options	*max. 40 bytes*
Data	*Variable*

SCTP	Length
IP Header	*20 bytes*
SCTP Gen. Hdr	*12 bytes*
chunk #1	*Variable*
chunk #2	*Variable*
chunk #3	*Variable*
chunk #n	*Variable*

Stream Control Transmission Protocol (SCTP)
Basic Chunk Types (1/3)

- *INIT*: Establishing the connection
  - Can contain multiple IPv4/IPv6 addresses per endpoint
  - Can contain multiple parallel streams per endpoint
  - Acknowledgement with *INIT ACK* chunk
- *SACK*: Acknowledging received chunks
  - Similar logic and retransmission algorithms to TCP SACK
- *HEARTBEAT*: “No operation” chunk
  - Testing if connection path works
  - Acknowledged using *HEARTBEAT ACK* chunk

Stream Control Transmission Protocol (SCTP)

Basic Chunk Types (2/3)

- *DATA*: Payload of SCTP transfer
  - Assigned with
    - ***Transport Sequence Number (TSN)***
    - ***Stream Identifier***
    - ***Payload Protocol Identifier***
  - Long messages can be fragmented over several chunks

Stream Control Transmission Protocol (SCTP)

Basic Chunk Types (3/3)

- *ABORT*: Immediately close association (typically due to error)
  - Should contain one or more error cause definitions
- *SHUTDOWN*: Close down a connection
  - Acknowledged with *SHUTDOWN ACK*
- *ERROR*: Report an error condition
  - Must be included with ABORT chunk to report fatal error
  - Can be included with other chunks to report non-fatal error

Stream Control Transmission Protocol (SCTP)

Multiple Streams

- Possible problem in TCP: Head-of-Line blocking
  - Consequence of requirement for strict ordering of data
  - Lost segment prevents delivery of following segments
- Some streams of data need to be delivered in order
- Individual streams can be delivered in free order
- Assign data to number of streams
  - Messages in single stream are delivered in order to the upper layers
  - Messages in different streams have independent ordering

Stream Control Transmission Protocol (SCTP)

Multi-homing

- **INIT chunk can include alternative IPv4 or IPv6 addresses**
  - **One of the addresses is indicated as *primary***
- **If the primary IP address fails, SCTP endpoint can use one of the alternative addresses**
  - **HEARTBEAT chunk is used to test a connection path**
  - **Number of connection paths: nr. sender IPs * nr. receiver IPs**
    - **Some combinations may not be possible (e.g., IPv4 to IPv6)**
- **Separate congestion control for each address pair**
  - **Not possible with TCP + layer 3 multi-homing/mobility protocols**

Stream Control Transmission Protocol (SCTP)

Chunk Authentication

- *AUTH* chunk: authenticate one or more other chunks in packet [RFC 4895]
  - Requires additional INIT parameters
    - RANDOM: random number shared between endpoints
    - CHUNKS: chunk types that must always be authenticated
    - HMAC-ALGO: Algorithm(s) to be used for authentication (SHA-1 must be supported)
  - Cryptographic keys can be pre-configured or shared by some other mechanism

Stream Control Transmission Protocol (SCTP)

Dynamic Address Update

- Specified extension in RFC 5061
- Originally the multi-homed address set was specified in the beginning of association
  - Not possible to change later during the association
- *ASCONF* chunk: Allow changing of address set during association
  - “Add IP” option: Add new IP addresses to the set
  - “Delete IP” option: Delete IP addresses from the set
- ASCONF chunks required to be authenticated with AUTH chunk
- Could enable “transport layer mobility”
  - However: SCTP has not been designed to replace mobility protocols

Stream Control Transmission Protocol (SCTP)

Partial Reliability

- **RFC 3758**
- **Default: SCTP retransmits until connection times out and aborts**
- **Option to specify upper limit to the number of retransmissions**
- **For traffic that has timing constraints**
- **Partial Reliability support should be indicated in INIT/INIT-ACK**
- **If supported by both ends**
    - **New chunk *FORWARD TSN*: indicates that receiver can move cumulative ACK point forward even if not receiving the segment**

Stream Control Transmission Protocol (SCTP)

Extended Socket API

- SCTP has several advanced features over TCP or UDP
  - Traditional socket API is not sufficient to control all features
- Extended socket API for controlling advanced features [draft-ietf-tsvwg-sctpsocket]
  - Basic features can be used also with the traditional API
- Additional features in extended socket API
  - Controlling multi-homing
    - For example, bind() for multiple addresses
  - Additional context data with sendmsg()
    - Stream identifier, Payload protocol identifier, ...

Stream Control Transmission Protocol (SCTP)

Concluding Remarks

- Many advanced features over TCP
- Especially suitable for signalling traffic
  - Used for signaling in telephone networks
  - Not in other wide-scale use
- Challenges with middleboxes
  - Network Address Translators (NATs) and firewalls do not recognize new protocol
  - Flexible chunk structure can be problematic
  - Multi-homing is particularly problematic with NATs
- Implemented in Linux + some other Unix systems

Datagram Congestion Control Protocol (DCCP)

Datagram Congestion Control Protocol (DCCP)

- Problem: UDP does not implement congestion control
  - Increased volumes of real-time traffic ⇨ congestion collapse?
- DCCP: Unreliable datagram-oriented protocol [RFC 4340]
- Connection-oriented
  - ⇨Requires connection state machine
- Congestion control
  - ⇨Requires acknowledgement mechanism, sequence numbers
- Negotiable features & options
  - ECN support, Checksum coverage, congestion control parameters

Datagram Congestion Control Protocol (DCCP)

Some DCCP Features

- Partial checksums
  - Many target applications prefer corrupted data to no data at all
  - Checksum coverage parameter: how much of the payload is covered
  - IP pseudoheader + DCCP header + options are always covered
- Service codes
  - Separate the two traditional purposes of transport ports
    - multiplexing
    - service identification
  - Enables connecting application using different ports

Datagram Congestion Control Protocol (DCCP)

Congestion Control Profiles

- **DCCP is capable of negotiating the congestion control profile**
  - **Identified with *Congestion Control Identifier (CCID)***
  - **End-hosts to agree based on the application requirements**
- **The initiator indicates the desired CCIDs in *DCCP-Request* packet**
  - **The responder either accepts one of the CCIDs or rejects connection**
  - **Different hosts may support different sets of CCIDs**
  - **Possibility to specify new CCIDs for different purposes**
- **Currently specified CCIDs:**
  - **CCID 2: "TCP-like Congestion Control" [RFC 4341]**
  - **CCID 3: "TCP-Friendly Rate Control" [RFC 4342, RFC 3448, Floyd00]**
  - **New profiles for better performance in progress**

Datagram Congestion Control Protocol (DCCP)

CCID-2: TCP-like Congestion Control

- **Behaviour similar to TCP congestion control**
  - **Window-based**
  - **Acknowledgement-clocked**
  - **For apps that tolerate abrupt changes of AIMD congestion control**
- **Differences to TCP**
  - **Acknowledgements need to be acknowledged**
    - **Also acknowledgements are congestion controlled**
    - **Allow receiver to clean up *acknowledgement vector* state**
  - **DCCP congestion control operates on packets, not on bytes**
  - **No retransmissions, some concepts need to be adapted (like RTO)**

Datagram Congestion Control Protocol (DCCP)

Acknowledgement Vectors

- **Used by CCID 2**
- **DCCP's "selective acknowledgement" system**
- **Receiver holds cumulative acknowledgement point**
- **Receiver collects information on which packets are lost, received or ECN-marked**
- **ACK vector grows every time a new packet arrives at receiver**
  - **DCCP-Ack packet communicates the vector to the sender**

Datagram Congestion Control Protocol (DCCP)

Acknowledgement Vectors

- **Example: cumulative ACK = 100**
  - **0 = received; 1 = received + ECN; 3 = not received**

	0	0	3	0	0	3	1	0	0	5
Pkt number	100		99		95-98		94		88-93	

(Length of vector: 5 * 4 = 20 bytes)

Packet 100 received
Packet 99 not received
Packets 95, 96, 97, 98 received
Packet 94 received but ECN marked
Packets 88, 89, 90, 91, 92, 93 received

Datagram Congestion Control Protocol (DCCP)

CCID-3: TCP-Friendly Rate Control (TFRC)

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}\left(3\sqrt{\frac{3p}{8}}\right)p(1+32p^2)}$$

($t_{RTO} = 4R$)

- TFRC is based on the TCP throughput equation
- Responds slower to the changes in path conditions
- Rate-clocked
  - No congestion window, algorithm adjusts the transmit rate in Bytes per sec
    - based on nominal packet size (s), loss event rate (p) & round-trip time (R)
  - Rate of incoming acknowledgements can be lower (e.g., 1 ACK / RTT)
  - Sender implementation needs additional timer to trigger packet sending
- Loss intervals option
  - Receiver indicates lengths of loss periods and lossless periods
    - Not detailed per-packet information as in ACK vectors
    - Loss period: period of time less than 1 RTT with one or more packet losses
  - Sender uses the intervals to calculate the loss event rate

Datagram Congestion Control Protocol (DCCP)

CCID-3: TCP-Friendly Rate Control (TFRC)

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}\left(3\sqrt{\frac{3p}{8}}\right)p(1+32p^2)}$$

- **What if packet loss rate (p) is 0?**
  - **For example in the beginning of connection...**
- **Double the transmission rate from the previous calculation**
  - **Similar to TCP slow start**
- **Initial transmit rate is 1 packet/second**
  - **until there is the first round-trip time measurement (R)**

Datagram Congestion Control Protocol (DCCP)

TFRC Enhancements

- Simulations have shown that TFRC does not work well for VoIP apps
  - With small packets TFRC performance is not very good
- Small-packet variant for TFRC (TFRC-SP) [RFC 4828]
  - Use true packet size *true_s* instead of the nominal size *s*
  - Include header length in packet size calculation
  - Limit minimum transmission interval to 10 ms
    - Discourage transmission of extremely small packets
  - Loss period length calculated in number of packets
  - To become CCID 4
- Some analysis of the differences of TFRC and TFRC-SP in [Kohler06]

Datagram Congestion Control Protocol (DCCP)

Concluding Remarks

- DCCP is intended for replacing UDP for long-lived non-reliable flows
  - Voice-over-IP, videoconferencing, games
- For short-lived flows congestion control is not important
  - No need of connection establishment, etc. overhead for DNS
- Not at all simple protocol compared to UDP
- There are prototype implementations, for example in Linux
- Currently NATs and firewalls do not support DCCP
- No much real-world experience yet
  - How does TFRC work over real links (for example wireless)?
  - How does TFRC work for real applications?

IETF Standardization

- Consists of tens of working groups, for example:
  - TSVWG: congestion control and SCTP maintenance/extensions
  - TCPM: TCP maintenance/extensions
  - DCCP: DCCP maintenance/extensions
  - http://tools.ietf.org/wg/dccp
    - (replace "dccp" with any other WG name)
- Documents are prepared in "Internet Drafts"
- When finished, Internet-Drafts become stable RFCs
- Discussion conducted on mailing lists
- Meetings 3 times a year

Thank You!

References (Books & Research papers)

[Brakmo95] L. Brakmo and L. Peterson. "*TCP Vegas: End to End Congestion Avoidance on a Global Internet*", IEEE Journal on Selected Areas in Communications, 13(8), pages 1465-1480, October 1995.

[Comer00] D. Comer. *"Internetworking with TCP/IP: Principles, Protocols, and Architectures"*, 4th ed. Prentice-Hall, 2000.

[Floyd93] S. Floyd and V. Jacobson. "*Random Early Detection Gateways for Congestion Avoidance*". IEEE/ACM Transactions on Networking, 1(4), p. 397-413, August 1993.

[Floyd94] S. Floyd. "*TCP and Explicit Congestion Notification*". ACM Computer Communication Review, 24(5), pages 10-23, October 1994.

[Floyd00] S. Floyd, M. Handley, J. Padhye, and J. Widmer. "*Equation-Based Congestion Control for Unicast Applications*", In Proc. of ACM SIGCOMM 2000, pages 43-56, August 2000.

[Jacobson88] V. Jacobson. "*Congestion Avoidance and Control*". In Proc. of ACM SIGCOMM '88, pages 314-329, August 1988.

[Kohler06] E. Kohler, M. Handley, and S. Floyd. "*Designing DCCP: Congestion Control Without Reliability*", In Proc. of ACM SIGCOMM 2006, pages 27-38, September 2006.

References (Books & Research papers)

[Ludwig00] R. Ludwig and R.H. Katz. "*The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions*", ACM Computer Communication Review, 30(1), pp. 30-36, January 2000.

[Mascolo01] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, and R. Wang. "*TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links*", Proc. of ACM MOBICOM 2001, pages 287-297, July 2001.

[Padhye98] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. "*Modeling TCP Throughput: A Simple Model and its Empirical Validation*", Proc. of ACM SIGCOMM '98, pages 303-314, Vancouver, Canada, September 1998.

[Sarolahti03] P. Sarolahti, M. Kojo, and K. Raatikainen. "*F-RTO: an Enhanced Recovery Algorithm for TCP Retransmission Timeouts*", ACM Computer Communication Review, 33(2), pages 51-63, April 2003.

[Tan06] K. Tan, J. Song, Q. Zhang, M. Sridharan. "*A Compound TCP Approach for High-Speed and Long Distance Networks*", Proc. of IEEE Infocom 2006, pages 1-12, April 2006.

[Xu04] L. Xu, K. Harfoush, and I. Rhee. "*Binary Increase Congestion Control for Fast Long-Distance Networks*", Proc. of IEEE Infocom 2004, March 2004.

References (RFCs)

[RFC 1323] V. Jacobson, R. Braden, and D. Borman. *"TCP Extensions for High Performance"*, May 1992.

[RFC 2018] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. "*TCP Selective Acknowledgement Options*", Proposed Standard, October 1996.

[RFC 2581] M. Allman, V. Paxson, and W. Stevens. "*TCP Congestion Control*", Proposed Standard, April 1999.

[RFC 3782] S. Floyd, T, Henderson, and A. Gurtov. "*The NewReno Modification to TCP's Fast Recovery Algorithm*", Proposed Standard, April 2004.

[RFC 2914] S. Floyd. "*Congestion Control Principles*", Best Current Practice, September 2000.

[RFC 2988] V. Paxson and M. Allman. "*Computing TCP's Retransmission Timer*", Proposed Standard, November 2000.

[RFC 2960] R. Stewart, et. al. "*Stream Control Transmission Protocol*", Proposed Standard, October 2000.

References (RFCs)

[RFC 3042] M. Allman, H. Balakrishnan, and S. Floyd. "*Enhancing TCP's Loss Recovery Using Limited Transmit*", Proposed Standard, January 2001.

[RFC 3168] K. Ramakrishnan, S. Floyd, and D. Black. "*The Addition of Explicit Congestion Notification (ECN) to IP*", Proposed Standard, September 2001.

[RFC 3448] M. Handley, S. Floyd, J. Padhye, and J. Widmer. "*TCP Friendly Rate Control (TFRC): Protocol Specification*", Proposed Standard, January 2003.

[RFC 3517] E. Blanton, M. Allman, K. Fall, and L. Wang. "*A Conservative Selective Acknowledgement (SACK)-based Loss Recovery Algorithm for TCP*", Proposed Standard, April 2003.

[RFC 3522] R. Ludwig, M. Meyer. "*The Eifel Detection Algorithm for TCP*", Experimental, April 2003.

[RFC 3758] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, P. Conrad. "*Stream Control Transmission Protocol (SCTP) Partial Reliability Extension*", Proposed Standard, May 2004.

[RFC 4138] P. Sarolahti, M. Kojo. "*Forward RTO-Recovery (F-RTO): An Algorithm for Detecting Spurious Retransmission Timeouts with TCP and the Stream Control Transmission Protocol (SCTP)*", Experimental, August 2005.

References (RFCs)

[RFC 4340] E. Kohler, M. Handley, and S. Floyd. "*Datagram Congestion Control Protocol (DCCP)*", Proposed Standard, March 2006.

[RFC 4341] S. Floyd and E. Kohler. "*Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control*", Proposed Standard, March 2006.

[RFC 4342] S. Floyd, E. Kohler, and J. Padhye. "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC)", Proposed Standard, March 2006.

[RFC 4828] S. Floyd and E. Kohler. "*TCP Friendly Rate Control (TFRC): The Small-Packet (SP) Variant*", Experimental, April 2007.

[RFC 4895] M. Tuexen, R. Stewart, P. Lei, and E. Rescorla. "*Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)*", Proposed Standard, August 2007.

[RFC 5061] R. Stewart, Q. Xie, M. Tuexen, S. Maruyama, and M. Kozuka. "*Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration*", Proposed Standard, September 2007

References (Internet-Drafts)

[draft-allman-tcp-early-rexmt] M. Allman, K. Avrachenkov, U. Ayesta, and J. Blanton. "*Early Retransmit for TCP and SCTP*", Internet-Draft "draft-allman-tcp-early-rexmt-05", June 2007. Work in progress.

[draft-ietf-tsvwg-sctpsocket] R. Stewart, Q. Xie, L. Yarroll, K. Poon, and M. Tuexen. "Sockets API Extensions for Stream Control Transmission Protocol (SCTP)", Internet-Draft "draft-ietf-tsvwg-sctpsocket-15", July 2007. Work in progress.