TCP and QoS

Markus Peuhkuri

2001-11-13

Lecture topics

- Why QoS for TCP?
- TCP and quality needs
- Effect of QoS mechanisms on TCP
- Explicit Congestion Notification

Chapters from book: none

TCP is elastic . . .

- TCP is provides reliable byte stream
  - window-based flow control
  - AIMD (Adaptive Increase, Multiplicative Decrease) controlled congestion window
    - probes available bandwidth
    - controls number of packets in network
- Throughput depends on
  - round trip time
  - packet loss rate

  Throughput (in segments) is something like

  $$B < \min\left(\frac{W}{RTT}, \frac{1}{RTT\sqrt{p}}\right) \quad (1)$$

  B number of segments in time unit, RTT round trip time, W window size in segments. Equation 1 is only approximate for steady-state and does not hold for large packet losses or during slow-start.

  Number of congestion signals (dropped packets or multiple acks) depends on size of a flow and for each signal rate is halved. This results two multiplicative effects and $1/\sqrt{p}$ term [5, footnote 6 on pages 4–5].

. . . users are not

- TCP can live with packet rate of 0.01 Hz (<15 bit/s)
- Applications need some minimum bandwidth to maintain their fidelity
- Interactive applications need minimum response time

Interactive applications

- Interactive applications must follow 0.1/1/10 rule [7, p. 135]

  0.1 s immediate feedback to user, to show user that action command (mouse click, key press, voice) was received.

  1 s user's workflow is not interrupted if task completes within one second. If task is not completed, then there should be at least an indicator to show user that task is proceeding with some estimate of time to complete.

  10 s user losses ones attention to task in question if task does not complete within 10 seconds. User either interrupts task or switches to an another task.

- Acceptable performance needs
  - small delay
  - sufficient bandwidth

  Transferring 7 KiB document within 1 s using HTTP

	(a)	(b)
minimum link speed	64 kbit/s	10 Mbit/s
1-way delay	4,4 ms	198 ms

  TCP connection is established but it is not closed. Maximum segment size of 1500 bytes is assumed and slow start with initial congestion window of two segments. End system processing time for each segment is assumed to be 0.1 ms in average including time to process and deliver HTTP request and response. No segments are lost.

  The major difference between those two cases is caused by packet serialisation time which totals to 0.98 s in case (a) and 0.006 s in case (b). Average network utilisation is 61 kbit/s in both cases (as equal segments as transmitted).

- Interactive TCP applications do need some performance assurance
- Available bandwidth is the most important
- Delay should not vary too much or exceed some limit

Partially interactive applications

- Application is controlled by a human operator, but transfer takes place in background
  - FTP transfers
  - email MUA-MTA (mail user agent – mail transfer agent) communications
- Much longer transfer times are tolerated
- Some upper limit for total transfer time, depends on application: few minutes for email, half an hour for FTP

Non-interactive communications

- Communication between end hosts
- Each ongoing communication takes some resources from end system
- Application timeouts [1]

  FTP server command timeout of 5 minutes

  SMTP per command timeout 2–10 minutes (depending on command), per session timeout of an hour

  DNS default timeout 5 seconds with exponential back-off (up to one minute)

- There may be "user behind" waiting for email, for example
- Even longer times tolerated

How loss and delay affect

- Both decrease throughput
- Loss rate

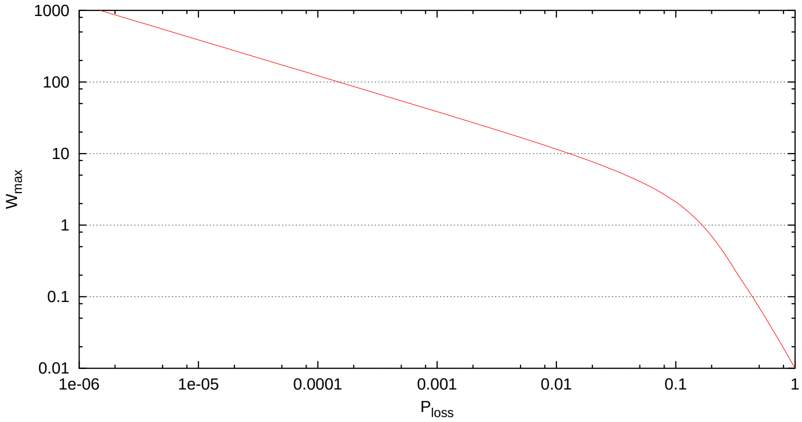
  $<$ **1 %** very little effect

  $>$ **20 %** throughput very low
- Delay
  - at longer delays window size is limiting factor
- One approximation of bandwidth

$$B \approx \min\left(\frac{W}{RTT}, \frac{1}{RTT\sqrt{\frac{2p}{3}} + T_0 \min\left(1, 3\sqrt{\frac{3p}{4}}\right) p(1+32p^2)}\right) \tag{2}$$

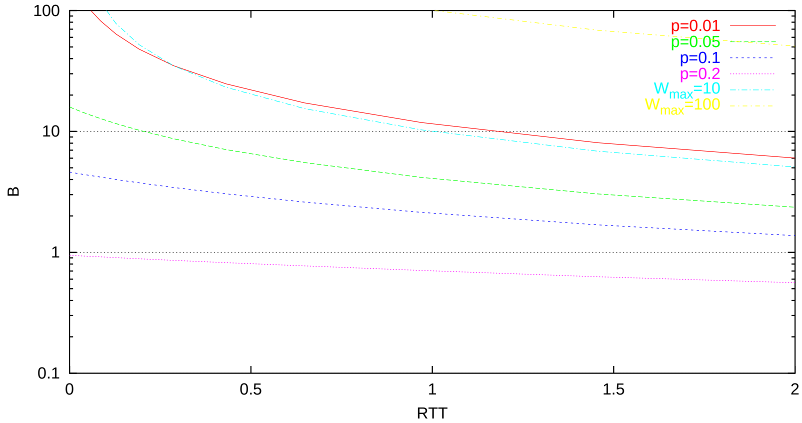
[8]

Loss and maximum window

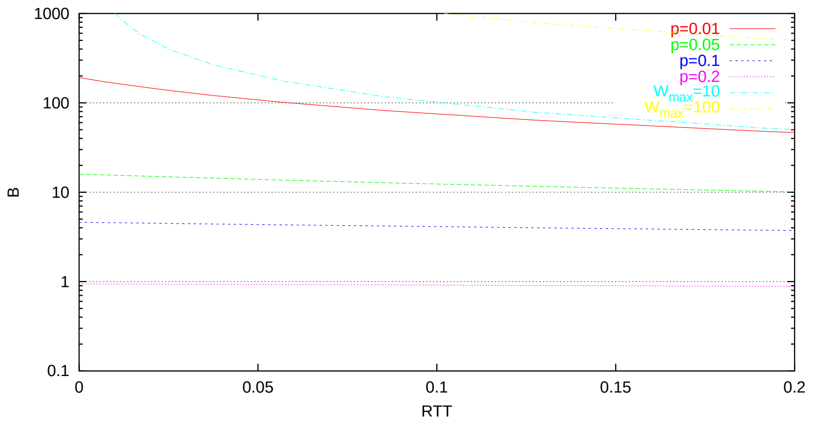


$RTT = 1$

Throughput: loss and round trip time



Throughput: loss and round trip time

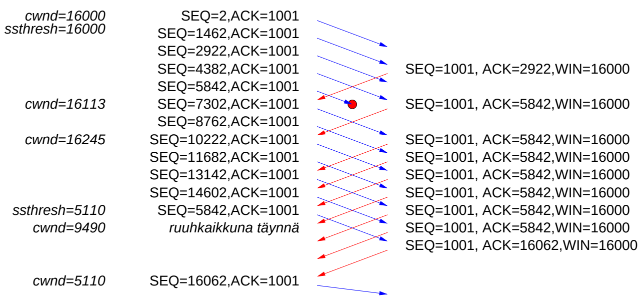


Fast Retransmit / Fast Recovery

If a segment is lost
⇒ gap in byte sequence

- Receiver acks last byte of *continuous*?sequence, if receives bytes after hole
  ⇒ segment loss is identified by 3 duplicates
- Retransmit Timeout if window size small, because there are no segments in flight to trigger duplicate acks

1. $sstresh = \max\{FlightSize/2, 2SMSS\}$
2. retransmit of lost, $cwnd = sstresh + 3SMSS$
3. for each duplicate received $cwnd = cwnd + SMSS$
4. continue sending if $cwnd$ and receiver window allows
5. set $cwnd = sstresh$ when new data is acknowledged



TCP and QoS marking

- TCP bursty by design
- For each ack, available window is sent
- Burst may exceed allowed *burst size*
  ⇒ Tail of bursts gets marked out-profile
  ⇒ Loss of multiple segments
  ⇒ Possibly Retransmit Timeout

Partial ACK

- With a large $FlightSize$ there may be several holes
  ⇒ a new fast recovery for each hole
  ⇒ $cwnd$ reduced for each, maybe too much
- Reduce only once for each $FlightSize$
- Still problems identifying which segment(s) to resend

Selective Acknowledgement

- Helps to identify lost segments [6]
- Use agreed on SYN-segments with *TCP Sack-Permitted Option*
- In case of loss, receiver sends ACK (as normal), and a partial list (TCP maximum option size of 40 bytes allows 4 blocks; 3 if Timestamp option is used) of some segments received
- First block includes SACK relevant for this ACK
- Receiver *may drop* some data that is SACKed but not ACKed
- D-SACK (Duplicate SACK [4]) reports duplicates received
  ⇒ info about spurious retransmits

Elephants and mices

Elephant A flow which lasts for a long time and has many bytes in it

- terminal sessions
- usenet news server-server traffic
- database synchronisation
  ⇒ steady-state communication

Mice Short-lived flow with only few segments

- HTTP requests
- DNS (on top of UDP)
  ⇒ only in slow-start phase
  ⇒ does not react to congestion control, unfairness
- majority of flows; and of bytes

Should connections be limited?

- Limit number of TCP connections at link
  - guarantee minimum bandwidth
  - with high loss, goodput rate low
- Limit maximum flow speed
  - if a flow takes a great partition of link capacity, slow it
  - charge only high-bandwidth flows
  - go around by using multiple TCP flows
    ⇒ limit by host, network…
- Report congestion to edge routers to enforce policing for misbehaving flows
- Some utility, however problems with
  1. scalability
  2. fairness
  3. accuracy

Active queue management

- RED (Random Early Detection) drops packet even if queue not full
  - statistical dropping
  - aim to avoid synchronisation of flows
- Packet drop crude *signal* of congestion
  - delivered packet has better utility than dropped
    - has already used some network resources
    - better fidelity without retransmissions
    - TCP must wait for multiple packets before it can distinguish between reorder and drop
- Some better indication needed

Signals of Congestion

IP ICMP Source Quench

- router sends if its resources are exhausted [3]
- sender *must* limit transmission rate; for TCP react as
  - retransmission timeout had occured[2], or
  - cut congestion window into half as in Fast Retransmit
- adds traffic to congested network
  ⇒ needs rate limiting
- fast feedback: less than RTT

Packet networks DECbit[12]

- a bit is set by average queue size
- if more than half of packets have bit set
  ⇒ decrease congestion window multiplicatively, otherwise increase additively

Frame Relay FECN/BECN

- based on virtual channels, set up by signalling or network management
- FECN set if packet experienced congestion in transit
- BECN set if congestion in reverse direction

IP Explicit Congestion Notification

- Possibility to indicate congestion in network by marking packets[11]
  ⇒ no traffic added
- Internet routes asymmetric
  ⇒ recipient must echo; needs support in both end systems
  ⇒ delay of RTT
- If transport protocol is *ECN capable*, it may set ECT bit ECN Capable Transport
- If router *would drop* and *ECT is set*
  ⇒ router sets CE bit (Congestion Experienced)
- Transport protocol *must* react if packet had *been drop*

  TCP reduce congestion window

  VoIP reduce sending rate; possibly using higher compression rates

  multicast select stream with lower rate, if there is one available (as should be because ECN is activated)

- Redefined ToS field

0 1 2 3 4 5	6 7
DS field, DSCP	ECN field

- ECN codepoints

0	0	Not-ECT
0	1	ECT(1) (ECN-Capable Transport)
1	0	ECT(0)
1	1	CE (Congestion Experienced)

  Older specification [10] used ECT bit (bit 6) to indicate ECT and if congestion was experienced, CE bit (bit 7) was set. Value “01” (currently ECT(1)) was not defined.

  Use of two different ECT values can serve 1-bit nonce to protect end systems from misbehaving network elements.

TCP and ECN

0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

Data Offset	*Reserved*	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window

- Use negotiated at connection setup

  1. connection initiator sets CWR (Congestion Window Reduced) and ECE (ECN-echo) bits
  2. recipient replies with CWR clear and ECE set
  3. connection may use CE codepoint

  Note that use of ECN differs from other TCP extension using options. This may result some problems with non-compliant firewalls and end systems.

- Congestion signalled once for RTT

  1. receives TCP segment with CE bit set, sets ECE bit for all TCP segment it sends
  2. receives TCP segment with ECE bit set, reduces congestion window and sets CWR bit; ignores ECE until next RTT
  3. receives TCP segment with CWR bit set, stops setting ECE bit

Possible problems in ECN

- Unresponsive hosts
  - host may report it honours ECN
    ⇒ packet not dropped but marked
  - ignores CE, does not reduce rate
  - host can behave badly without ECN by increasing sending rate with FEC (Forward Error Correction)
- Feedback delay
  - asymmetric routing; a router may not see other direction
- IP tunnels; IPSec
  - DS byte “volatile” (not covered by AH or ESP headers)
  - in tunnel mode IPSec outer header discarded at end of tunnel
  - should ECN or DiffServ codepoints be copied to inner header?
  - depends on situation
    ⇒ ECN Tunnel attribute for IPSec SA (Security Association)
- TCP specification

> If an incoming segment has a security level, or compartment, or precedence which does not exactly match the level, and compartment, and precedence requested for the connection, a reset is sent and connection goes to the CLOSED state. The reset takes its sequence number from the ACK field of the incoming segment. [9, p. 37]

- problems with both DiffServ and ECN
- only few implementations check for those
  ⇒ TCP updated in RFC2873 to ignore precedence [13]

Summary

- While elastic, TCP need some QoS
- Bursty losses bad; especially with small window
- ECM provides gentler signal of congestion

References

[1] R. Braden and Ed. Requirements for Internet Hosts - Application and Support. Request for Comments RFC 1123, Internet Engineering Task Force, October 1989. (Internet Standard) (Updates RFC0822) (Updated by RFC2181) (Also STD0003). URL:`http://www.ietf.org/rfc/rfc1123.txt`.

[2] R. Braden and Ed. Requirements for Internet Hosts - Communication Layers. Request for Comments RFC 1122, Internet Engineering Task Force, October 1989. (Internet Standard) (Also STD0003). URL:`http://www.ietf.org/rfc/rfc1122.txt`.

[3] R.T. Braden and J. Postel. Requirements for Internet gateways. Request for Comments RFC 1009, Internet Engineering Task Force, June 1987. (Historic) (Obsoleted by RFC1812) (Obsoletes RFC0985). URL:`http://www.ietf.org/rfc/rfc1009.txt`.

[4] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky. An Extension to the Selective Acknowledgement (SACK) Option for TCP. Request for Comments RFC 2883, Internet Engineering Task Force, July 2000. (Internet Proposed Standard). URL:`http://www.ietf.org/rfc/rfc2883.txt`.

[5] V. Jacobson. Congestion avoidance and control. In *Proceedings of the ACM SIGCOMM Conference*, pages 314–329, August 1988.

[6] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgement Options. Request for Comments RFC 2018, Internet Engineering Task Force, October 1996. (Internet Proposed Standard) (Obsoletes RFC1072). URL:`http://www.ietf.org/rfc/rfc2018.txt`.

[7] Jakob Nielsen. *Usability Engineering*. AP Professional, Boston, 1993.

[8] Jitedra Padhye, Victor Firoiu, Don Towsley, and Jim Krusoe. Modeling TCP throughput: A simple model and its empirical validation. In *ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 303–314, Vancouver, CA, 1998. URL:`ftp://gaia.cs.umass.edu/pub/Padhye-Firoiu98-TCP-throughput-TR.ps`.

[9] J. Postel. Transmission Control Protocol. Request for Comments RFC 793, Internet Engineering Task Force, September 1981. (Internet Standard) (Updated by RFC3168) (Also STD0007). URL:`http://www.ietf.org/rfc/rfc793.txt`.

[10] K. Ramakrishnan and S. Floyd. A Proposal to add Explicit Congestion Notification (ECN) to IP. Request for Comments RFC 2481, Internet Engineering Task Force, January 1999. (Experimental) (Obsoleted by RFC3168). URL:`http://www.ietf.org/rfc/rfc2481.txt`.

[11] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. Request for Comments RFC 3168, Internet Engineering Task Force, September 2001. (Internet Proposed Standard) (Updates RFC2474, RFC2401, RFC0793) (Obsoletes RFC2481). URL:`http://www.ietf.org/rfc/rfc3168.txt`.

[12] K. Ramakrishnan and R. Jain. A binary feeback scheme for congestion avoidance in computer networks. *ACM Transactions on Computer Systems*, (2):158–181, May 1990.

[13] X. Xiao, A. Hannan, V. Paxson, and E. Crabbe. TCP Processing of the IPv4 Precedence Field. Request for Comments RFC 2873, Internet Engineering Task Force, June 2000. (Internet Proposed Standard). URL:`http://www.ietf.org/rfc/rfc2873.txt`.