Flow Control

TCP Flow Control

Note: *This topic describes the Reno enhancement of classical "Van Jacobson" or Tahoe congestion control. There have been many suggestions for improving this mechanism - see the topic on high-speed TCP variants.*

TCP flow control and window size adjustment is mainly based on two key mechanism: *Slow Start* and *Additive Increase/Multiplicative Decrease* (AIMD), also known as Congestion Avoidance. (RFC 793 and RFC 5681)

Slow Start

To avoid that a starting TCP connection floods the network, a *Slow Start* mechanism was introduced in TCP. This mechanism effectively probes to find the available bandwidth.

In addition to the window advertised by the receiver, a *Congestion Window* (`cwnd=) value is used and the effective window size is the lesser of the two. The starting value of the =cwnd` window is set initially to a value that has been evolving over the years, the TCP Initial Window. After each acknowledgment, the `cwnd` window is increased by one MSS. By this algorithm, the data rate of the sender doubles each round-trip time (RTT) interval (actually, taking into account Delayed ACKs, rate increases by 50% every RTT). For a properly implemented version of TCP this increase continues until:

- the advertised window size is reached
- congestion (packet loss) is detected on the connection.
- there is no traffic waiting to take advantage of an increased window (i.e. cwnd should only grow if it needs to)

When congestion is detected, the TCP flow-control mode is changed from *Slow Start* to *Congestion Avoidance*. Note that some TCP implementations maintain cwnd in units of bytes, while others use units of full-sized segments.

Congestion Avoidance

Once congestion is detected (through timeout and/or duplicate ACKs), the data rate is reduced in order to let the network recover.

Slow Start uses an exponential increase in window size and thus also in data rate. *Congestion Avoidance* uses a linear growth function (additive increase). This is achieved by introducing - in addition to the `cwnd` window - a *slow start threshold* (`ssthresh`).

As long as `cwnd` is less than `ssthresh` , *Slow Start* applies. Once `ssthresh` is reached, `cwnd` is increased by at most one segment per RTT. The `cwnd` window continues to open with this linear rate until a congestion event is detected.

When congestion is detected, `ssthresh` is set to half the `cwnd` (or to be strictly accurate, half the "Flight Size". This distinction is important if the implementation lets cwnd grow beyond `rwnd` (the receiver's declared window)).
`cwnd` is either set to 1 if congestion was signalled by a timeout, forcing the sender to enter *Slow Start*, or to `ssthresh` if congestion was signalled by duplicate ACKs and the *Fast Recovery* algorithm has terminated. In either case, once the sender enters Congestion Avoidance, its rate has been reduced to half the value at the time of congestion. This multiplicative decrease causes the `cwnd` to close exponentially with each detected loss event.

Fast Retransmit

In *Fast Retransmit*, the arrival of three duplicate ACKs is interpreted as packet loss, and retransmission starts before the retransmission timer (RTO) expires.

The missing segment will be retransmitted immediately without going through the normal retransmission queue processing. This improves performance by eliminating delays that would suspend effective data flow on the link.

Fast Recovery

Fast Recovery is used to react quickly to a single packet loss. In Fast recovery, the receipt of 3 duplicate ACKs, while being taken to mean a loss of a segment, does not result in a full Slow Start. This is because obviously later segments got through, and hence congestion is not stopping everything. In fast recovery, ssthresh is set to half of the current send window size, the missing segment is retransmitted (*Fast Retransmit*) and `cwnd` is set to `ssthresh` plus three segments. Each additional duplicate ACK indicates that one segment has left the network at the receiver and `cwnd` is increased by one segment to allow the transmission of another segment if allowed by the new `cwnd` . When an ACK is received for new data, `cwmd` is reset to the `ssthresh` , and TCP enters congestion avoidance mode.

References

- *Congestion Avoidance and Control*, V. Jacobson, Computer Communication Review, vol. 18, no. 4, pp. 314-329, August 1988, ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z
- *TCP Congestion Control*, RFC 5681, M. Allman, V. Paxson, E. Blanton, September 2009
- *Congestion Control in the RFC Series*, RFC 5783, M. Welzl, W. Eddy, February 2010
- *Computing TCP's Retransmission Timer*, RFC 6298, V. Paxson, M. Allman, J. Chu, M. Sargent, June 2011
- *Congestion Control in Linux TCP*, P. Sarolahti, A. Kuznetsov, USENIX Annual Technical Conference 2002, Freenix Track

- *The Great Internet TCP Congestion Control Census*, A. Mishra, X. Sun, A. Jain, S. Pande, R. Joshi, B. Leong, ACM SIGMETRICS, December 2019 (PDF, presentation video, Gordon code)

– Main.UlrichSchmid - 07 Jun 2005
– Main.SimonLeinen - 27 Jan 2006 - 20 Jun 2020