Congestion Avoidance and Control

Van Jacobson, Michael J. Karels
Paper review by: **Ehsun Daroodi**
810181050

The flow on a TCP connection should obey a 'conservation of packets' principle. By conservation of packets we mean that a connection in equilibrium (running stably with a full window of data in transit), a new packet isn't put into the network until an old packet leaves. In this way, sender uses acks as a clock to strobe new packets into network (self-clocking).

To start the clock we use a slow-start algorithm to gradually increase the amount of data in-transit. This algorithm increments the TCP window size (cwnd) whenever an ack is received. When starting or restarting after a loss, cwnd is set to one.

Any protocol that expects to survive heavy load should use a good round trip time (RTT) estimator. We need it to compute the retransmit timeout interval, rto, for the next packet sent. We account for RTT variations in computing rto to avoid retransmitting of some packets that delayed in transit. We use an exponential backoff when a packet has to be retransmitted more than once.

A 'congestion avoidance' strategy, have two components: The network must be able to signal the transport endpoints that congestion is occurring. And the endpoints must have a policy that decreases utilization when signal is received and increases it if the signal isn't received. The signal can be a packet time out due to a loss. A source controls load in a window-based protocol. On congestion, it uses a multiplicative decrease of the window size. On no congestion, the best increase policy is to make small and constant changes to window size.

Gateway uses a 'congestion detection' algorithm to send signals to the endpoints to reduce their traffic on congestion. Since congestion grows exponentially, detecting it early is important.

Slow-start and congestion avoidance algorithms conflict with each other in someway; so senders use a combined slow-start with congestion avoidance algorithm, instead.