Congestion Avoidance Notes
Sai Anirudh Kondaveeti

I. CONGESTION AVOIDANCE AND CONTROL

‘Congestion collapse’ is the state of network where the network is delivering packets at its capacity, but most of it is useless as most of the packets are duplicates (caused by retransmissions). These ‘congestion collapses’ started happening around 1886 and this paper was published in 1888 to avoid this problem. This algorithm was built on the ideas introduced by the DECbit where they tried to avoid congestion by updating a bit to signal congestion. But this approach was not implementable as it involved changing functionality of routers. So a more general way to detect congestion through estimation of packet losses is suggested. These changes were just required at the end host. Also this is the first implementation of congestion control on Unix kernel(BSD 4.3). This implementation is one of the key reasons for the wide adoption of this algorithm. The other reason for its wide adoption is the dire need of a solution to avoid ‘congestion collapse’.

II. CONSERVATION OF PACKETS

The condition that “A new packet isn’t put into the network until an old packet leaves”, ensure that a collapse will be avoided. This principle only fails in the following ways:

- The connection doesn’t get to equilibrium, or
- A sender injects a new packet before an old packet has exited, or
- The equilibrium can’t be reached because of resource limits along the path.

It is to be noted that the older version of TCP wasn’t following this principle. It was just dumping a large window of packets onto the network. This led to overflowing of buffers causing persistent retransmissions.

III. ALGORITHMIC IDEAS

The main contributions of this paper are:

- Slow start algorithm to get to the equilibrium
- Retransmission timeout estimation using rtt variance so as to maintain the equilibrium.
- Exponential back-off which describes how the retransmits be spaced when a packet has to be retransmitted more than once. This technique helps in maintaining equilibrium.
- Congestion avoidance which describes how the congestion window has to increase when there is no congestion and how the congestion window has to decrease in case of congestion.

These ideas were very fundamental and almost every Internet connected computer in the world ran a slightly modified version of this algorithm between 1988 and 2004.

IV. FIXING THIS LARGE WINDOW PROBLEM

Instead of using a large window size at the start of transmission we need to find the right window size at the current network conditions. To do that we start our window size by 1 and increase it by 1 on every ACK. By following this rule our window size gets doubled every RTT. This is an exponential increase in size of window over time but still it is considered slow compared to a fixed large window size. This increase in window size stops when we detect a packet loss through a timeout. Whether the bottleneck link capacity is 56 Kbps or 10 Mbps, we get to the optimal window size very fast as it is exponential increase.

V. RETRANSMISSION TIMEOUT

Estimating the retransmission timeout is essential. If we estimate it to a large value and retransmit the packet late then we are holding up the delivery of data. We are holding up the data because TCP provides an in-order reliable bytestream. If we estimate it too small and retransmit packets then we are hogging up the network resources which will lead to congestion collapse.

In this paper a new method for retransmission timeout was suggested using rtt variance. The previous approaches were not estimating the rtt variance and were using a constant factor beta over mean RTT to estimate retransmission timeout. Here the beta accounted to rtt variance. This led to wrong estimation of retransmit timeout interval. In this paper they suggested a faster (Optimizing RTT mean and variance calculation using integer arithmetic instead of floating point) and accurate way to estimate variance which led to setting a better retransmission timeout. This retransmission timeout is adaptive as it tracks the mean RTT and variance of RTT more closely (compare Figure 6 and 5). This led to avoiding unnecessary retransmits and maintain the network at equilibrium.

VI. EXPONENTIAL BACK-OFF

Another scenario which previous TCP implementations didn’t deal correctly is the scenario when the retransmitted packet is lost. The loss of a retransmitted packet means the estimates could be way off. So this problem is dealt by doubling the retransmission timeout until an ACK is received. Once the ACK is received the timeout estimation logic is reset and restarted. In practice there is an upper bound on the maximum size of retransmission timeout (typically 60 seconds). This exponential back-off nature of retransmission timeout can be explained by the analogy that the network is a linear system. If a linear system is unstable (about to have congestion), stability is achieved through exponential damping (exponential timer back-off) to its primary excitation (senders, traffic sources).
VII. CONGESTION AVOIDANCE

After the slow start which gets to the optimal window size we need to cautiously explore for more bandwidth or decrease our bandwidth depending on network load. This congestion avoidance phase allows different flows to be in equilibrium and consequently allows them to converge to their fair share. In the steady state the window size is increased by $1/cwnd$ on every ACK. This can also be seen as increasing the window size by 1 packet every RTT (additive increase). In case of packet loss during this increase the window size is decremented by a factor of 2 (multiplicative decrease).

VIII. WHY ADDITIVE INCREASE MULTIPLICATIVE DECREASE (AIMD)?

This is the right combination because this algorithm converges all the flows to an equilibrium (Chiu-Jain plots). The reasoning behind multiplicative decrease is that the queue lengths in the router increase exponentially which can be countered by multiplicative decrease in windows size. By doing this multiplicative decrease the queues will stop growing and consequently drain. The paper doesn’t explain the reasoning behind additive increase but most probably the reason could be that the alternative which is multiplicative increase leads to congestion.

IX. GATEWAY CONGESTION CONTROL

There is a need for congestion control algorithms at gateways as the end hosts can’t ensure fairness at a granularity less than a second. Also it is not expected of the every user to use the congestion control algorithm which gives rise to some user unfairly using other user’s share of bandwidth. To mitigate these problems we need to introduce some intelligence in gateways. These problems are tackled in the next lecture through protocols like XCP and WFQ.

X. OVERALL SUMMARY OF THE PAPER

The paper successfully demonstrates a new congestion control algorithm which achieves better effective bandwidth (figure 11).