**SIM21 : Implementing Hoe’s Modification to Slow Start**

**Team :**

*16co213 ---Lokesh Manideep*

*16co220 ---Sai Pradyumna*

*16co225 ---Prasanth Sagar Kottakota*

**Breif : ​**

It proposes changes to improve the startup behavior of the congestion control scheme.

Changes proposed include using an estimated value instead of default value for the slow start threshold at start-up, and modifying the Fast Retransmit algorithm.

**Differences of hoes modification to tcp and normal tcp**

|  |  |
| --- | --- |
| **Normal tcp** | **Hoes modifed tcp** |
| Congestion control :  1)Slow start  2)fast retransmit and fast recovery  **Slow start -**  Starts with a congestion window of one segment and exponentially increases the congestion window . When the congestion window hits the ‘ssthresh’ the sender continues to increase the congestion window(cnwd) linearly,probing the network capacity as it becomes available.  In this scheme,the choice of threshhold , which is an estimation of the equilibrium operating point ie, packet leaves the network as a sender puts a packet into the network,is key to the performance of algorithm.  **Fast retransmit and fast recovery -**  The algorithm helps in recovering the lost packet iff in single packet loss.  Where as, in case of multiple packet loss ,fast recovery does’nt have significant effect.  THe issues for time consuming loss recovery is  1)the initial arbitrary value of ‘ssthresh’ , which allows the sender to clock out a large surge of packets in exponential slow-start mode leading to multiple packet losses (ie as cwnd might be low at start the packets may be lost as window size is small )  2) The failure of the Fast Retransmit algorithm to recover the lost packets. The result is that the sender must wait for the long retransmission timeout, which drastically reduces the performance during the start-up period. | In this modification, we propose changes , making better use of existing information so that we can take advantage of the queue.  **Better initial value of ssthresh**  We see that initial value of ssthresh is critical. One way to avoid the large surge of packet that leads to multiple packet losses is to pick a better ssthresh than the arbitrary initial value  A lower ssthersh would allow cwnd to open exponentially upto ssthresh and then linearly (reduces overfeeding capacity) , but the sender would prematurely switch to the additive increase mode, and the performance would suffer. Thers wont be any packet loss but sender would take long tme to transfer.  So it is advised to nd an estimate of the threshold at which the sender is closely approaching the full network capacity and thus should slow down and probe the remaining capacity.  **Method suggested :**  bandwidth delay product.  Working :  Band width approximation :  Datapackets are sent closely, so they arrive at the receiver at the rate of narrow bandwidth link(we can assume without loss of packets),If ACK ‘s arrive at the sender with approximately same spacing,  We can calculate approximate bandwidth.  Delay approximation :  The round-trip delay can be approximated by timing one segment, reading the timestamp upon sending the segment  and reading the timestamp again once the ACK for the segment is received. We can then calculate the bandwidth delay product.  We set ssthresh to the byte-equivalent of  the above product. |
| In the cases, where false fast retransmits  can occur, where retransmission of a segment that was not lost , generates a duplicate ACK. and thus leads to false fast retransmission occurs, we call it as false fast retransmission , since the algorithm is activated even though there is no segment loss. The false retransmit mistakenly forces the sender to go into the less aggressive linear mode, when there is really no congestion. | To deal with these issues, there are two simple approaches:  (1) curtail the surge of packets that lead to multiple packet losses  (2) change the Fast Retransmit algorithm so that it recovers from multiple losses of packets in the same window and thus reduces the need to wait for the retransmit  timeouts. The rst approach can be implemented by fnding a better initial value of ssthresh , and the second requires a more aggressive Fast Retransmit algorithm to recover lost  segments. |

Algorithm :

Below we can find an algorithm, where when we get 3 duplicate ACK’s we generally do

(a) ssthresh value reduces to half of the current window size.

(b) set cwnd= ssthresh

(c) start with congestion avoidance phase

but,here

We make a different algorithm as shown in fig

Fast retransmit phase

Setup

(i) set ssthresh = 1/2 congestion window

congestion window = 1 segment

Snd\_nxt = snd\_una

(ii) set snd\_high = snd\_max

Save\_cwnd = ssthresh + 1 segment

Retransmit

(iii) send using slow start algorithm

(iv) allow congestion window to increase as long as

congestion window <= save\_cwnd

(v) do not start a new fast retransmit phase

(vi) upon receiving 2 duplicate ACK’s, send out 1 new

packet beyond snd\_high

The sender to be known the packet loss before actual timeout(in this way we can save time that we wait for the original timeout), then it will start a retransmission due to time out case (it is better use full for multiple packet losses) .

