haodongj@andrew.cmu.edu
yunchens@andrew.cmu.edu

Reno Analysis

Transfer time for a 20MB file: 103 seconds

Congestion window vs. time:

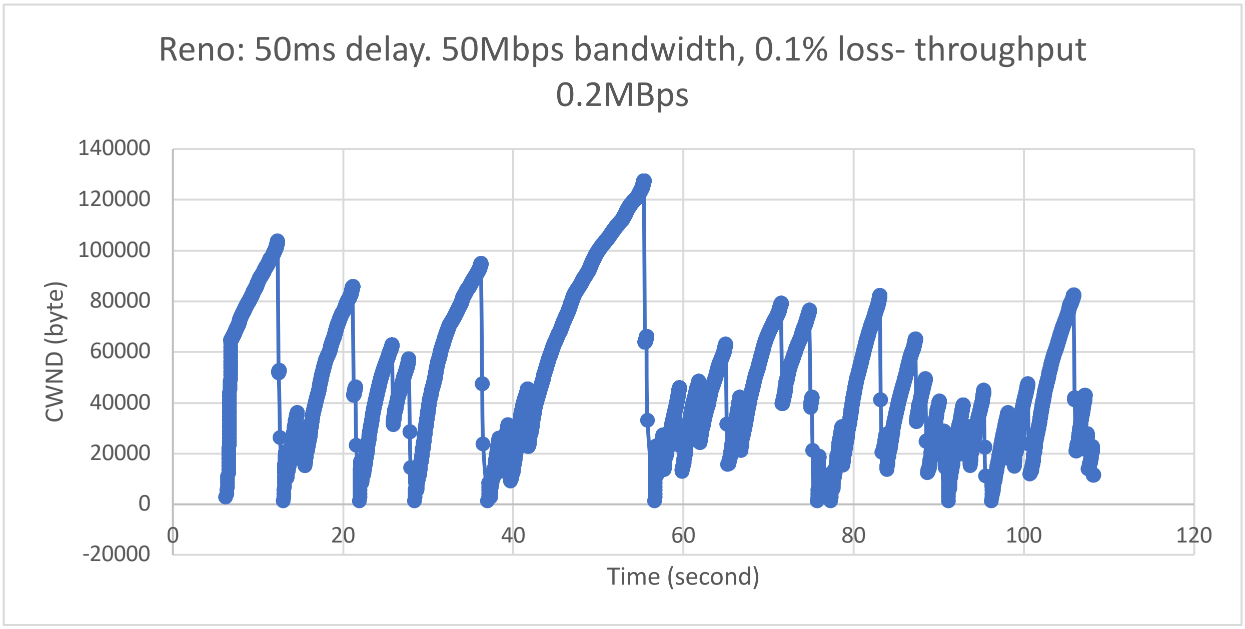

Figure 1

Discussion:

As the graph shows, the original Reno only increases its congestion window in a linear fashion that takes too long to reach the capacity of modern network channel.

At the same time, every time it times out it assumes network congestion happened and the punishment is to heavy, meaning the congestion window backs off too much. As shown in Figure 1 at around second 71, the window was halved. Those two features combined dictated that Reno can hardly utilize the full capacity of a modern network link, despite the fact that the loss rate is very low.

Algorithm Proposal

We propose an upgraded version of Reno, Banana.

Banana shares the same states and CWND control mechanism with Reno but they are different mainly in three ways.

First, every timeout would change the state to SLOW_START but the increase rate at SLOW_START state is substantially higher than Reno. Second, during congestion avoidance, Banana increases its window ten times faster than original Reno in an attempt to reach link capacity faster. Finally, whenever packet loss happens, the congestion window only decreases by one fifteenth, alleviating the punishment.

The quick growth during SLOW_START phase is to ensure that Banana is still fair because when timeout happens, which indicates a high likelihood of network congestion, it returns back to SLOW_START. Such behavior ensures that all competing flows start equal whenever congestion happens. Figure 2 below shows the state machine of Banana.

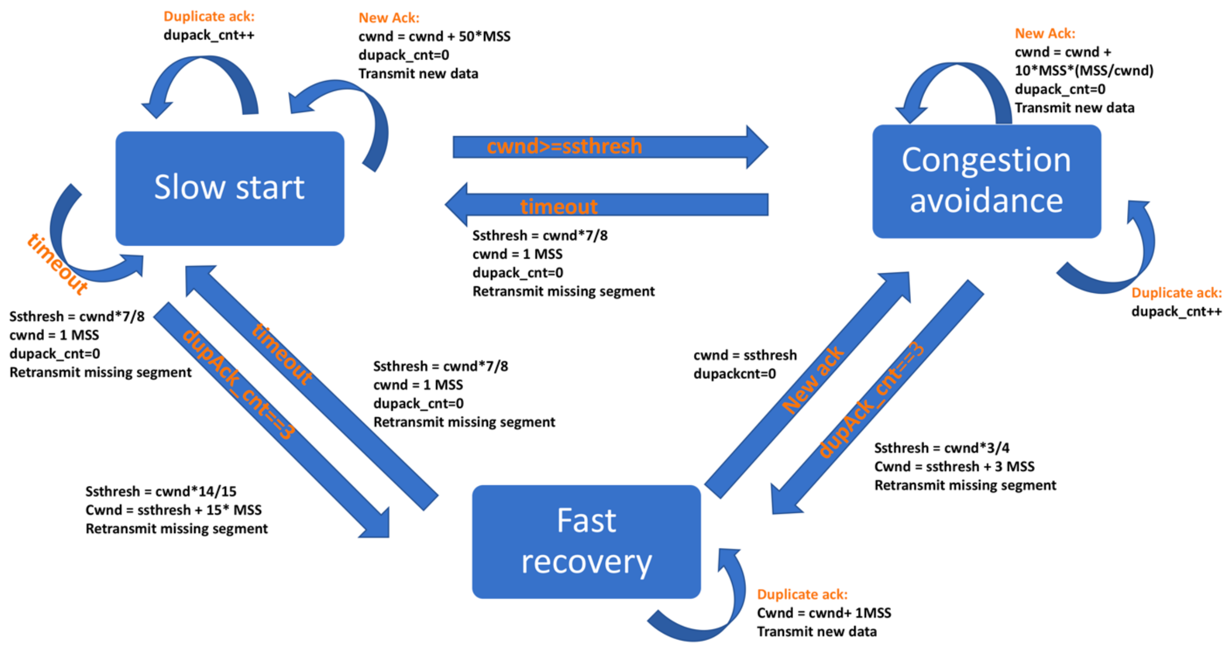

Figure 2

Evaluation: Banana

Transfer time for a 20MB file (single flow): 54 seconds

Transfer time for a 20MB file (two flows): 90 seconds on average

JFI: 0.9994009721714491

CWND change during transmission:

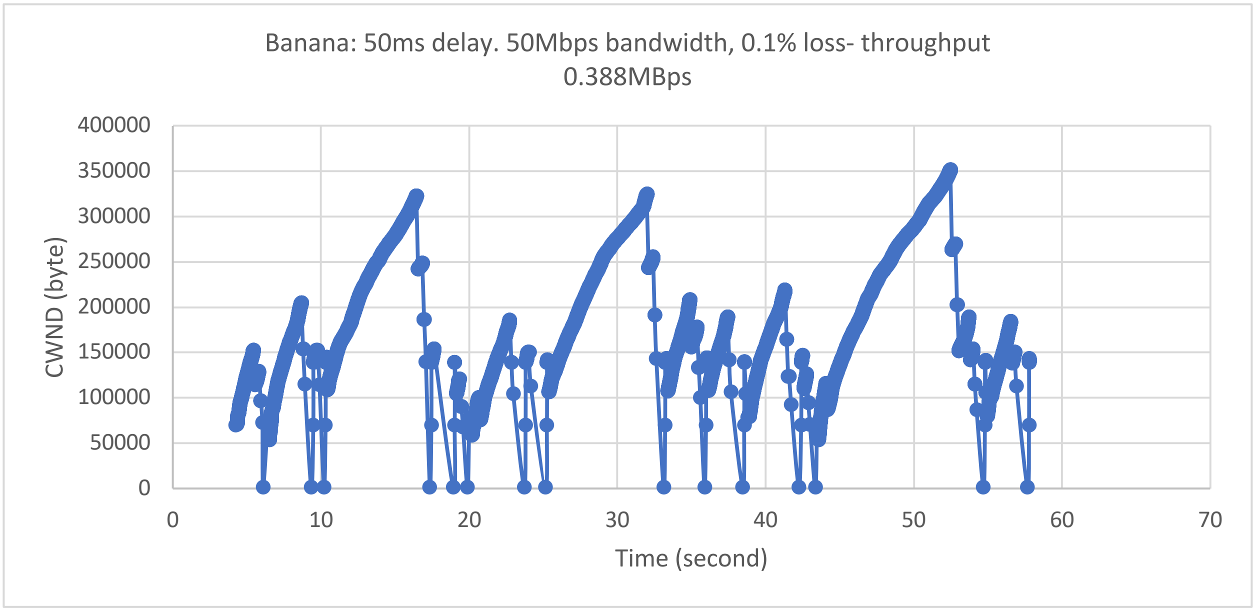

Figure 3

Discussion:

The Banana achieved average throughput of 0.388MBps compared to original Reno which had only 0.2MBps, we can see an increase of 94%. At the same time, Banana kept the JFI high at a fair level.

In Figure 3, we can see that the sender can return to congestion avoidance after a timeout much faster than original reno and even at CONGESTION_AVOIDANCE phase, the congestion window increases faster than original reno to approach network capacity faster. At time 35 second, a DupAck happened and the CWND was preserved to three fourths of what it was before to preserve the CWND size. Both changes combined, the average throughput was increased substantially.