RFC 5681 – Summary

**Introduction:**

RFC 5681 specifies about four of the congestion control algorithms. Apart from explaining about the algorithms, this RFC also defines about how the TCP connection should react after a long ideal time along with the clarification of the issues related to TCP ACK generation.

**Congestion Control Algorithms:**

There are scenarios where the TCP sender tends to be conservative than required. This can be beneficial considering the situation, but the TCP sender should never be aggressive than the allowable limit by the congestion control algorithms. This is obvious because when the TCP sender tends to be conservative, though it might not make use of the total available capacity, it still does the job without any data loss. However, when it tries to be more aggressive, there are good changes of data loss since the TCP sender is trying to send more data than the available bandwidth for the data transfer. Also, the congestion in the network is identified through the loss of data in the below-mentioned algorithms.

**Slow Start and Congestion Avoidance:**

These algorithms to curb the data which is sent to the network. Two variables are added to the TCP for every connection. The minimum of the Congestion Window (cwnd – Sender Side Limit) and the Receivers Advertised Window (rwnd – Receiver Side Limit) helps the TCP determine the value which could be used to limit the data which will be transmitted in the network. The second variable – Slow Start Threshold (ssthresh) is used to choose between one of these two algorithms.

When a new transmission is beginning, TCP uses a slow start algorithm to release the data into the network. The reason behind this is that the network conditions are unknown, and the TCP does not want to send huge chunks of data and congest the network. Hence this algorithm can be used to calculate the available capacity of the network. This algorithm is also used to start the “ACK clock”. The upper bound of the cwnd`s Initial Window (IW) must be set based on the value of SMSS. The ssthresh`s initial value is set to the maximum possible announced window. However, this value should reduce according to the congestion response from the data connection. The slow start algorithm is implemented when Congestion Window < Slow Start Threshold and when the other way is true then congestion avoidance algorithm is applied. If both the values are same, then the sender can choose any one of the two algorithms.

Congestion Window is incremented by TCP by atmost by SMSS bytes in case of slow start algorithm. This increment of cwnd happens whenever an acknowledgement is received. When ssthresh >= cwnd or when congestion occurs in the network, then this algorithm comes to an end. The recommendation here is that the TCP should increment cwnd based on the minimum value of N and SMSS. This also helps in handling the “ACK Division”. Congestion Window is incremented by one full-sized segment per RTT in case of congestion avoidance algorithm. This algorithm runs till any congestion is observed in the network. The recommendation here to increment cwnd based on a number which is equal to the count of the number of bytes that have been acknowledged by ACKs for the new data. It is important to note that cwnd should never increase by more than SMSS bytes per RTT. After resending the dropped data, TCP sender usually starts with the slow start – increases the window till it reaches ssthresh – then followed by congestion avoidance algorithm.

**Fast Retransmit/Fast Recovery**

When a segment is out of place, then the TCP Sender sends a duplicate acknowledgement to inform the sender. The reason for the duplicate ACKs are as follows. Firstly, they can be caused due to dropped segments. Secondly, it can be because of the reordering of the data segment. Moreover, it can be due to the replication of the ACK or the data segment. Once TCP receiver receives the missing data, it should send an ACK immediately to the sender which helps the sender to recover quickly. To repair the loss occurred due to the duplicate acknowledgements, the TCP sender uses fast retransmit algorithm. This works in such a way that if it receives three duplicate ACKs, then TCP retransmits the lost data segment. Once the lost data is transmitted, then the fast recovery algorithm takes over.

**Additional Considerations:**

**Restarting Idle Connections:**

After an extended period of idle time, TCP congestion control algorithms tend to send an inappropriate chunk of traffic through the network. This is because it cannot use ACK clock as all the ACKs would have drained. The recommendation here is that, after an extended period of idle time, slow start algorithm should be used by the TCP which also starts the ACK clock.

**Generating Acknowledgements:**

The following are the recommendations: TCP receiver can delay the acknowledgements which are being sent from them. This process is called delayed acknowledgements. When doing so, the delay should not be excessive. Also, the acknowledgements should be sent immediately when the received segments are out of order, and when the missing data is received, the same process has to be followed to notify the sender.

**Loss Recovery Mechanisms:**

There are possibilities where multiple losses occur in a single window of data. The recommendation here is that the implementors of TCP should implement advanced loss recovery algorithm to take care of this scenario.

**Security Considerations:**

Currently, on receiving an ACK, the size of the cwnd is increased based on a constant number. The recommendation here is that the size of the cwnd should be increased based on the number of acknowledgements received. On successful implementation of this recommendation, we could avoid the congestion collapse and preserve the network stability.