Analysis of TCP congestion control protocols

Introduction:

The protocols in transport layer provides host to host communication. There are two important protocols in transport layer. One is TCP which provides connection-oriented transmissions, the other one is UDP which provides connectionless transmissions. In TCP the connection should be established using three-way handshake where as in UDP, connection is not established between client and server. Transmission over TCP is reliable as it guarantees the delivery of data to destination whereas the delivery of data is not guaranteed in UDP. TCP provides some mechanisms like flow control, congestion control which are not provided by UDP. In TCP, the packets at destination arrive in order whereas in UDP, it is not guaranteed. UDP also has some advantages as it is faster and efficient compared to TCP. TCP is used by HTTP, HTTPS, SMTP etc. whereas UDP is used by DNS, DHCP, SNMP etc. TCP has many variants like TCP Cubic, TCP Reno, TCP Vegas. In this report we will compare different variants of TCPs by changing bandwidths, delays and losses. We will use Netem tool to change different network parameters.

TCP variants:

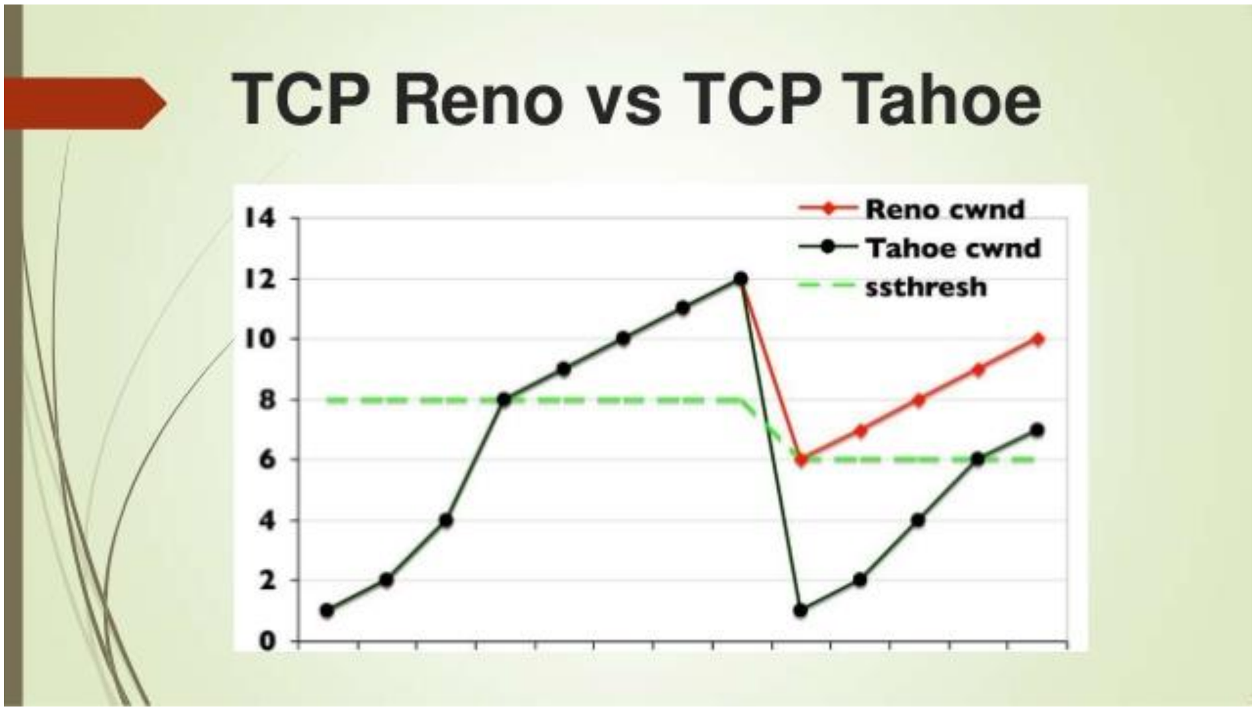
There are many variants of TCP like TCP Vegas, TCP Reno, TCP Cubic etc. let us discuss briefly about their features and when we should use them.

TCP Tahoe

Before TCP Tahoe, TCP used go back n model to avoid congestion in transmission**.** TCP Tahoe consist of slow start phase, congestion avoidance phase and fast retransmit phase. In this we will have threshold called ssthres. We start with congestion window size of 1. It is in slow start phase initially. In slow start phase, for every acknowledgement received it increases the size of congestion window by 1. Initial size is 1, it receives the acknowledgement and increases congestion window size to 2. In next RTT as it receives 2 acknowledgements, the window size will become 4 and it increases similarly. When it is in slow start phase the size of congestion window grows exponentially. Once it reaches threshold it will go to congestion avoidance phase where it will increase the congestion window size by 1 for every RTT. When we are in congestion avoidance phase, packet loss indicates congestion and the current window size falls to 1. This phase is called fast retransmit. Now for the second iteration the value of threshold is set to half of the size of congestion window at which packet loss occurred. TCP Tahoe deals packet loss and 3 duplicate acks in the similar way. Tahoe uses additive increase and multiplicative decrease algorithm to set the size of the congestion window.

Problems

It will take a complete timeout to detect a packet loss. Every time a packet is lost, it waits for timeout and then the pipeline will be emptied.



TCP congestion window vs RTT graph

TCP Reno

In TCP Reno, we will have 4 phases called slow start phase, congestion avoidance phase, fast recovery phase and fast retransmit phase. In the slow start phase the size of congestion window increases exponentially similar to Tahoe. In the congestion avoidance phase, the window size will be increased by 1 for every RTT. TCP Reno deals timeout and 3 duplicate acks in different ways. It will behave similar to Tahoe in the case of timeout. If 3 duplicate acks arrive in congestion avoidance phase it switches to fast recovery phase. The value of threshold is set to half the size of congestion window. In fast recovery current window size is changed to (window size)/2 +3. If duplicate acks are still coming, it will increase the congestion window size by 1 for every duplicate ack. When next new ack comes, the size of congestion window decreases to threshold unlike Tahoe where it will drop to 1.

Problems

TCP Reno is helpful only when one packet is lost. When multiple losses occur, it works similar to TCP Tahoe. It assumes that there is no packet loss in the fast recovery phase. If window size is very less and a loss occurs, we will not receive duplicate acks and we should wait for the timeout. To overcome these problems and deal with multiple losses, TCP New Reno is evolved.

TCP New Reno is efficient when compared to TCP Tahoe and Reno. When duplicate ACKs trigger a retransmission for a lost packet, TCP New Reno remembers the highest packet sent from the window (‘recover’ variable). If recover is greater than packet number which is acknowledged, the acknowledgement is partial ack otherwise it is new ack. If the acknowledgement received is partial ack, then stay in fast recovery else exit fast recovery.

TCP Vegas

The variants of TCP studied above detects congestion based on packet loss. TCP Vegas is congestion avoidance algorithm. It detects congestion at an early stage based on increasing Round-Trip Time (RTT) values of the packets in the connection unlike other flavors such as Reno, New Reno, etc., which detect congestion only after it has actually happened via packet loss.

TCP source maintains RTT known as baseRTT. Expected size of window is (congestion window size/baseRTT). TCP Vegas maintains two thresholds, i.e. alpha and beta.

If (expected – actual) is less than alpha, the size of congestion window is increased. If it is more than beta, size of congestion window is decreased.

It can detect packet losses much faster than Tahoe and retransmits them. More than half of time outs of Reno are prevented by Vegas as it detects and retransmits more than one lost packet before the timeout occurs.

TCP Cubic

It is an optimized congestion control algorithm used when the network is of high bandwidth and high latency. The window size in Cubic is Cubic function of time elapsed since the last congestion event. As it is Cubic function, the graph will have convex and concave portions. In the concave part, the window size quickly ramps up to the size before the last congestion event. In the convex part, it increases slowly at first and then increases rapidly. Cubic spends more time on plateau between convex and concave parts where the network stabilizes. Unlike other TCP variants, Cubic window size depends only on the last congestion event.

Setup:

In this project we are working with 3 machines. Machine 2 acts as server whereas machine 1 and machine 3 are clients. Machine 1 sends a file to machine 2 and machine 2 sends it to machine 3. Machine 2 will not wait until the entire file is received. As soon as it receives data from machine 1 it transmits to machine 3. While sending data, machine 2 changes network parameters. We are using netem tool to modify the network parameters like loss, bandwidth and delay.

Results:

No changes in network

Size	TCP Cubic	TCP Vegas	TCP Reno
1MB	0.3852	0.7866	0.3827
10MB	8.41	8.4310	8.7392
50MB	46.45	44.2004	43.8557

Delay 10ms

Size	TCP Cubic	TCP Vegas	TCP Reno
1MB	0.7966	0.7865	0.7958
10MB	8.8218	8.8291	8.8055
50MB	47.4964	44.3022	44.1687

Delay 100ms

Size	TCP Cubic	TCP Vegas	TCP Reno
1MB	1.1825	2.7971	0.7701
10MB	11.7007	14.4536	8.4000
50MB	68.488	64.5862	44.1587

Delay 1000ms

Size	TCP Cubic	TCP Vegas	TCP Reno
1MB	10.54	14.1801	16.3727
10MB	82.73	141.9261	135.2289
50MB	335.677	411.1534	386.1748

Loss 1%

Size	TCP Cubic	TCP Vegas	TCP Reno
1MB	0.3970	0.4005	0.3853
10MB	8.4744	8.4844	8.3966
50MB	43.9047	44.7698	45.4696

Loss 5%

Size	TCP Cubic	TCP Vegas	TCP Reno
1MB	0.8329	0.3934	0.3826
10MB	9.5075	10.3529	8.8483
50MB	68.7385	67.7720	61.4900

Loss 10%

Size	TCP Cubic	TCP Vegas	TCP Reno
1MB	1.3150	0.6800	2.6671
10MB	24.8221	17.8782	13.2908
50MB	168.4872	151.6790	139.6114

Bandwidth 1MBps

Size	TCP Cubic	TCP Vegas	TCP Reno
1MB	1.1141	1.1196	0.5806
10MB	12.7579	12.2754	12.2478
50MB	60.4357	62.2836	60.6091

Bandwidth 2MBps

Size	TCP Cubic	TCP Vegas	TCP Reno
1MB	0.5663	1.2267	0.5042
10MB	11.1936	11.1287	11.1935
50MB	55.1077	55.1831	54.4429

Bandwidth 5MBps

Size	TCP Cubic	TCP Vegas	TCP Reno
1MB	0.5197	1.4189	1.5067
10MB	11.5557	11.1563	11.2806
50MB	55.8022	55.6088	55.3054

Analysis of the obtained results:

No Congestion in Network

- When there is no congestion in the network all the congestion control protocols (TCP Reno, TCP Cubic, TCP Vegas) performed similarly

Delay in Network

- With the added delay in the network TCP Cubic performance > TCP Reno > TCP Vegas. This difference in performance is only observed when the delay is high and file size is large. When the delay in the network is low then TCP Reno is performing better than TCP Cubic
- When the delay is high overall performance of all the three protocols decreased

Loss in Network

- When there is loss in the network TCP Reno performance > TCP Cubic performance > TCP Vegas. This behavior is observed with all tested file sizes but the performance difference is significant only with high loss network

Modification of Network Bandwidth

- The tested three congestion protocols performed similarly when the bandwidth of the network is changed and performance of all the three protocols decreased when the bandwidth is low
- The performance increases with increase in bandwidth and reaches saturation after some bandwidth threshold.

- TCP Vegas performed poorly when compared to other two in all the congestion cases but it is expected of TCP Vegas because it is more of a congestion avoidance algorithm rather than a congestion control one.

Remarks

- The above results may vary based on congestion in local network.

Group members

2016A7PS0142H	I.Keerthan
2016A7PS0104H	E.Anish Reddy
2016A7PS0034H	S.L.Ruthvik Reddy
2016A7PS0079H	U.Sai surya