COMPUTER NETWORKS

ASSIGNMENT 2

SUBMITTED TO:	**Dr Umar Farooq**
SUBMITTED BY:	**MUAWIZ UMAR**
Roll no:	**359607**

CE-42-A

DATE: 28/11/2022

DEPARTMENT OF COMPUTER AND SOFTWARE ENGINEERING

TCP Taho and TCP Reno perform well for usual network conditions. However, their congestion avoidance mechanisms result in noticeably degraded performance for high-speed networks.

Q. 1) Can you calculate the time needed to achieve a data rate of 1 Gbps, using TCP Reno, where RTT is 2000 ms, current congestion window (CWND) size is 1 MSS, and last timeout had occurred at CWND = 128 MSS.

The findings from Q.1 are sufficient to indicate the performance degradation faced by TCP in high-speed networks. Therefore, several alternatives have been developed.

Threshold occurs due to time out. CWND becomes half the size in case of TCP reno congestion algorithm.

ssthread = cwnd/2 = 64

as we know,

1 MSS = 1460 byte

Cwnd = 1 MSS

RTT = 2000ms

Sending Rate S = W x MSS / RTT

=>W * (1460/2000 * 10) = 125,000 byte/s

=> (125000 * 2000 * 10-3)/1460 = 171 MSS

=> (14 + 99) *(2000*10-3)

Time required to achieve 171 MSS=226s

****

Q. 2) Can you compare and criticize the following TCP versions for their performance in high-bandwidth networks, wireless/loss-prone networks, and high-delay networks (impact of RTT on performance)

Introduction:

In wired and wireless networks, TCP is the most popular transport protocol. Because mobile nodes regularly change the topology in mobile ad hoc networks, there are considerable packet losses and a decline in network performance. This is because TCP is unable to differentiate between route failure and network congestion.

Start slowly: TCP must first determine the amount of available bandwidth in order to transfer data quickly. Performing this will increase the TCP connection's throughput dramatically reduce, since if the buffer is full, the intermediate routers must queue or discard the packets. Congestion window, abbreviated as cwnd, is a new parameter added by the slow start mechanism that regulates the pace at which packets are transmitted.

Each time a fresh TCP connection establishes, cwnd's initial value is set to a number that is no larger than two segments and no less than the maximum segment size (MSS). The cwnd is expanded by one segment each time an ACK segment is obtained. In this manner, the cwnd is extended to two and two data segments are provided in response to an ACK that acknowledges the first packet. The cwnd is incremented to four after ACKs for these two segments come. It gives the cwnd parameter an exponential rise. The minimum of the receiver's advertised window and the TCP sender's own value of cwnd are the maximums that can be sent. TCP stays in this slow star phase with exponential growth if cwnd value is less than or equal to slow start threshold (ssthresh).

Congestion Avoidance: Congestion Avoidance: If packets are dropped, the TCP algorithm uses congestion avoidance to prevent further packet losses. Congestion avoidance is done via TCP. whenever cwnd exceeds ssthresh. Every round-trip in the congestion avoidance phase results in an increase in the cwnd of 1 full-sized segment (RTT). Until congestion is felt, traffic is avoided.

Congestion can be detected in two ways:
1) Receipt of 3duplicate acknowledgment
2) Due to time timeout.

FAST TCP and TCP Vegas

QUICK AQM Active Queue Management Scalable TCP, where TCP stands for Transmission Control Protocol.

An algorithm for TCP that avoids congestion on long-distance, high-latency networks. The average transmitting rate is dependent on the likelihood of a loss in most existing congestion management algorithms, which detect congestion and slow down when they notice missed packets. This has two shortcomings.

First, to maintain high data rates, low loss probabilities are necessary. TCP Reno requires extremely low loss probability, but even more recent congestion avoidance algorithms, like H-TCP, demand loss rates that are lower than those offered by most wireless wide area networks. Additionally, although delay is a more comprehensive measure of congestion than packet loss, which only gives a single bit of information.

One of the better variations is TCP Vegas. It employs triple duplicate acknowledgments, which invariably need retransmission of the packet. Additionally, Vegas has unveiled a brand-new retransmission system for dropped packets. To speed up the process of identifying packet losses, it also calculates the round-trip transmissions and examines each transmission's duration to re-transmit them if it takes longer than anticipated. Additionally, it uses improved slow start and congestion avoidance algorithms to discover the best window before suffering losses. Along with the ability to quickly retransmit and recover, this variation also detects repeated packet losses. However, it also has the downside of lacking a rerouting method. Even so, it still outperforms the alternatives listed above.

A FAST TCP flow aims to ensure an even distribution of packets in network queues. By calculating the difference between the observed round-trip time (RTT) and the base RTT, which is the round-trip time when there is no queuing, one may estimate the number of packets in queues. The smallest observed RTT for the connection is believed to be the base RTT. The transmitting rate increases when too few packets are queued, whereas it decreases when there are too many packets queued. It is a direct ancestor of TCP Vegas in this regard.

When the number of packets stored is too few or too many, TCP Vegas adjusts the rate differently than FAST TCP. Regardless of how close to the goal rate the existing rate is, TCP Vegas adjusts it by a constant amount. When the system is further from equilibrium, FAST TCP takes greater steps; when it is closer to equilibrium, it takes smaller steps. This increases both the stability and the speed of convergence.

TCP CUBIC and TCP WESTWOOD

One of the newest and most effective TCP variations is CUBIC. Given that it is currently Linux's default TCP algorithm, its effectiveness is evident. Less aggressive and methodical CUBIC, where the congestion window is a cubic function of the number of packet losses since the last one. Along with the ability to quickly retransmit and recover, this variation also detects repeated packet losses. When it comes to adjusting the congestion window, TCP CUBIC has an advantage over the competition.

TCP Westwood (TCPW) is a sender-side-only enhancement to TCP New Reno that is designed to manage dynamic load, high bandwidth-delay product routes (big pipes), and pipes that may leak packets because of transmission faults or other problems (dynamic pipes).
The main novel concept of TCP Westwood is to monitor the pace of returning ACKs at the sender side to assess the bandwidth consumed by the connection. Compared to TCP Reno, this enables a quicker recovery of the connection. When using wireless lines, the TCP Westwood mechanism is especially useful because conventional congestion algorithms sometimes mistake intermittent losses brought on by radio channel issues for congestion and thereby reduce the TCP window without necessity.

To adjust more accurately the Slow, Start Threshold (ssthresh) and Congestion Window congestion management settings, TCP Westwood mines the ACK stream for information (cwin). While a loss is detected or when TCP Westwood is in its "Agile Probing" phase, a suggested change to the well-known slow start phase, the sender updates ssthresh and cwin using an estimated "Eligible Rate." In addition, a technique known as Persistent Non-Congestion Detection (PNCD) has been developed to identify persistent lack of congestion and trigger an Agile Probing phase to quickly use enormous dynamic bandwidth.

Throughput	Congestion control algorithm (sender)	latency	loss
2.35 Gb/s	Cubic	<1ms	0%
195 Mb/s	Reno	140ms	0%
347 Mb/s	Cubic	140ms	0%
344 Mb/s	Westwood	140ms	0%
340 Mb/s	BBR	140ms	0%
1.13 Mb/s	Reno	140ms	1.5% (sender > receiver)
1.23 Mb/s	Cubic	140ms	1.5% (sender > receiver)
2.46 Mb/s	Westwood	140ms	1.5% (sender > receiver)
160 Mb/s	BBR	140ms	1.5% (sender > receiver)
0.65 Mb/s	Reno	140ms	3% (sender > receiver)
0.78 Mb/s	Cubic	140ms	3% (sender > receiver)
0.97 Mb/s	Westwood	140ms	3% (sender > receiver)
132 Mb/s	BBR	140ms	3% (sender > receiver)

CONCLUSION:

There is no version that can completely solve the congestion issue, as may be inferred from the information about the many variations.

Because queueing delay is used by both FAST TCP and TCP Vegas as a congestion indicator, their window updating algorithms depend on propagation delay, which is calculated using a metric called baseRTT. The shortest round-trip time (RTT) to date is the metric used to describe it. Because the routers' queues might occasionally remain full, base RTT estimation of the actual propagation delay may be off, leading to unfairness and excessive queue variation.

It can be concluded from the information about different variants that there is none which can overcome the problem of congestion. All have their own pros and cons however, looking at all the parameters’ Vegas and newly CUBIC provides more efficiency and give better results.