Volume 6, Issue 1, January 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Congestion Avoidance in TCP

Amogh Santosh Pai Raiturkar
Student, Department of Computer Engineering, PCCE
Goa University, Goa, India

Abstract— This paper is an exploratory survey of Congestion Avoidance in TCP. In addition to the Congestion Avoidance Mechanisms as well as standard Congestion Avoidance Algorithms used in common software implementations of TCP, this paper also describes some of the more common Traffic Shaping techniques developed by researchers over the years. By studying the congestion control avoidances in TCP implementation software and network hardware we can better comprehend the performance issues of packet switched networks and in particular, the public Internet.

Keywords— Congestion Avoidance, TCP, Mechanisms, Algorithms, Traffic Shaping, Networks.

I. INTRODUCTION

Congestion control is defined as when the user of the network collectively demands for more resources than the network has to offer. In general, good routing and good buffer management can help to overcome or rather alleviate the problems by spreading the sessions over the subnet resources. In case of TCP network, once the congestion happens, TCP will control the congestion. TCP repeatedly increases the load to find the point at which congestion occurs and then backs off from that point. Congestion can also occur when data arrives on big pipe and gets sent out on smaller pipe. Another case where congestion can occur is when multiple input streams arrive at the destination viz. router whose output capacity is less than sum of inputs. To deal with this lost packet, Congestion avoidance algorithm is used. In this algorithm, there are two indications of packet loss; a timeout occurring and the receipt of duplicate ACKs. Hence, loss of packet signals the congestion somewhere in network between source and destination. In case of congestion avoidance mechanism, it predicts when the congestion will occur and accordingly reduce the rate at which hosts sends the data just before the packets start being discarded. All these algorithms and mechanisms will be judged according to several categories viz. Performance, Fairness, Compatibility with current technology and Complexity. TCP congestion control dissolve congestion in bottleneck link by reducing window sizes. This is mainly done to solve an implicit global convex optimization, where source rates are primal variables updated at sources and congestion measures are dual variables updated at the links.

II. CONGESTION AVOIDANCE MECHANISMS

There are three methods which are used for congestion avoidance mechanisms. They are:

A. *DEC Bit*

Here each router monitors the load which it experiences and explicitly notifies the end nodes when congestion is about to occur. This notification is implemented by setting a binary congestion bit in the packets that flow through the router. The destination host then copies the congestion bit into ACK and sends it back to the source and accordingly, the source adjusts its sending rate so to avoid congestion. A single congestion bit is added to the packet header. A router sets this bit in a packet if average queue length is greater than or equal to 1 at times when packet arrives. Using queue length of 1 as the trigger for setting the congestion bit is a tradeoff between significant queuing and increased idle time that is it optimizes the power function. The source then maintains a congestion window and watches to see what fraction of the last window‘s worth of packet resulted in bit being set. If less than 50% of the packets had bit set, then source increases its congestion window by one packet. If it’s 50% or more then source decreases the congestion window to 0.875 times the previous value. The value 50% was chosen as the threshold based on analysis to peak of power curve.

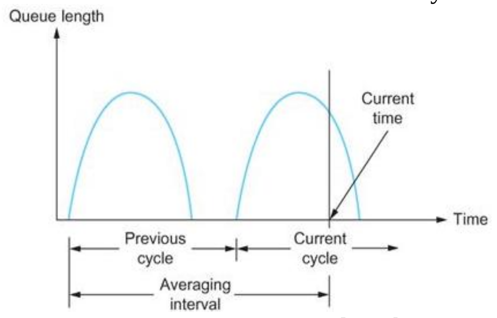

Fig. 1 Comparing average queue length at a router

B. Random Early Detection (RED)

It is similar to DEC bit scheme as in each router is programmed to monitor its own queue length, and when it detects that congestion is imminent, to notify the source to adjust its congestion window. RED is designed to be used in conjunction with TCP, which detects the congestion by means of timeouts. The router however drops a few packets before it has exhausted its buffer space completely so as to cause the source to slow down. The RED algorithm uses technique called early random drop that defines the details of how to monitor the queue length and when to drop a packet. The queue length is measured every time new packet arrives at the gateway. RED has two queue lengths threshold viz. Minimum Threshold and Maximum Threshold. If average queue length is less than equal to Minimum threshold, packet is queued; else we drop the arriving packet. Both the threshold methods are implemented by using FIFO queue.

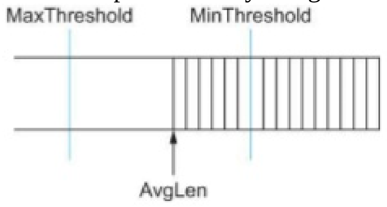

Fig. 2 RED thresholds on FIFO queue

C. Source-based Congestion Avoidance

The general idea of these techniques is to watch for some sign from the network that some router's queue is building up and that congestion will happen soon if nothing is done about it. The congestion window normally increases in TCP, but every two round-trip delays the algorithm checks to see if the current RTT is greater than the average of the minimum and maximum RTT. If it is, then algorithm decreases the congestion window by one-eighth. The window is adjusted once every two round-trip delays based on the product which is calculated as:

(Current Window – Old window) X (Current RTT – Old RTT)

If result is 0 or negative, source increases the window by one maximum packet size else the source decreases the window size by one-eighth.

III. PROPOSED ALGORITHMS

There are four types of algorithms that are used for congestion Avoidance in TCP. They are:

A. Slow Start Algorithm

TCP starts a connection with the sender injecting multiple segments into the network, up to the window size specified by the receiver. It operates by observing that the rate at which new packets should be injected into the network is the rate at which the ACK are returned by another end. It defines window called as "congestion window (cwnd)" for a new connection with size one segment. Every time an ACK is received, this window is increased by 1. The sender now starts by transmitting one segment and waiting for its acknowledgment. When that ACK is received, the cwnd is incremented from one to two, and two segments can be sent. When each of those two segments is acknowledged, cwnd is increased to four. This provides exponential growth. At some point the capacity of the internet can be reached and and intermediate router will start discarding the packets. This is the time the sender knows that its congestion window (cwnd) has gotten too large.

B. Congestion Avoidance Algorithm

The algorithm begins by initializing cwnd to one segment and ssthresh to 65535 bytes. When congestion occurs, one-half of the current window size is saved in ssthresh. Additionally, if the congestion is indicated by timeout, cwnd is set to one segment. When new data is acknowledged by other end, cwnd is increased but the way it increases depends on whether TCP is performing slow start or congestion avoidance algorithm. If cwnd is less than or equal to ssthresh, TCP is in slow start or else it is performing congestion avoidance algorithm. Slow start continues until TCP is halfway to where congestion occurred and then congestion avoidance algorithm takes over.

C. Fast Retransmit Algorithm

In this algorithm, TCP may generate an immediate acknowledgement when an out of order segment is received. This duplicate ACK shouldn't be delayed. The objective of duplicate ACK is to let other end know that a segment was received out of order, and to tell it what sequence number it is expected. Since TCP doesn't know whether duplicate ACK is caused by lost segment or just reordering of it, it waits for number of duplicate ACKs to be received. TCP then performs retransmission of what appears to be the missing segment, without waiting for a retransmission timer to expire.

D. Fast Recovery Algorithm

When duplicate ACK is received, the sender is aware that segment is only missed. For missing segment, Fast Retransmit algorithm is done and then congestion avoidance is performed. This is known as Fast Recovery algorithm. It is an improvement that allows high throughput especially for large windows. The fast retransmit and fast recovery algorithms are implemented together. When third duplicate ACK in a row is received, ssthresh is set to one-half the current congestion window, and missing segment is retransmitted with cwnd set to ssthresh plus 3 times the segment size. However, each time another duplicate ACK arrives, cwnd is incremented by segment size which inflates the congestion window and packet is transmitted. When the next ACK arrives that acknowledges new data, cwnd is set to ssthresh and ACK should acknowledge all the intermediate segments sent between the lost packet and receipt of the first duplicate ACK.

IV. TRAFFIC SHAPING TECHNIQUES

Traffic Shaping controls the rate at which packets are sent. They are mostly used in ATM Integrated Services Networks. The sender and carrier both negotiate traffic pattern. There are 2 traffic shaping algorithms:-.

A. Leaky Bucket Algorithm

This Algorithm is used to control rate in a network. It is implemented as a single-server queue with service time constant. If bucket overflow, packet is discarded. Operation is shown below.

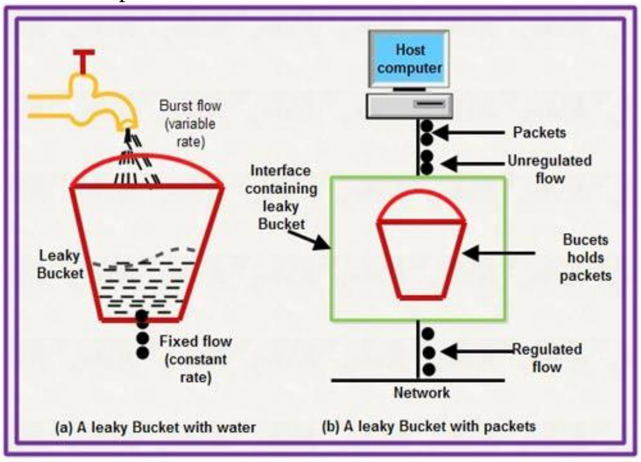
Burst flow (variable rate)
Leaky Bucket
Fixed flow (constant rate)
Host computer
Packets
Unregulated flow
Interface containing leaky Bucket
Bucets holds packets
Regulated flow
Network
(a) A leaky Bucket with water
(b) A leaky Bucket with packets

Fig. 3 Leaky Bucket Algorithm

The leaky bucket algorithm enforces a constant output rate and doesn’t do anything when input is idle. The host injects one packet per clock tick onto the network. This results in uniform flow of packets, smoothing bursts and reducing congestion. When packets are of same size, one packet per tick is okay. However, in case of variable length it allows fixed number of bytes per tick. Here, packets are sent at average rate and tokens are not saved.

B. Token Bucket Algorithm

The Token Bucket Algorithm allows the output to vary depending on the size of the burst. The bucket holds token at once together. To transmit a packet, the host must capture and destroy one token. These tokens are generated by a clock at the rate of one token every t seconds. Idle hosts can capture and save up tokens in order to send larger bursts. Unlike Leaky Bucket Algorithm, this algorithm doesn’t discard packet but rather discards token. The packets can only be transmitted if there are enough tokens to cover length in bytes.

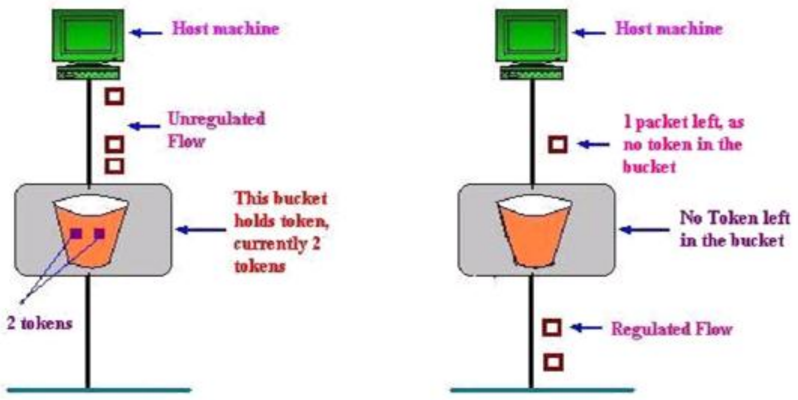
Host machine
Unregulated Flow
This bucket holds token, currently 2 tokens
2 tokens
Host machine
1 packet left, as no token in the bucket
No Token left in the bucket
Regulated Flow

Fig. 4 Token Bucket Algorithm

V. CONCLUSION

As seen in this paper, all various TCP congestion avoidance mechanisms and algorithms were studied and analyzed properly. It is seen that Congestion Avoidance is directly proportional to the performance of any network. From above mechanisms and algorithms, congestion can be controlled or rather avoided in TCP but nevertheless there are some more algorithms too at the same time being considered in research arena which will give better time complexity and better results than above all those described. The overall advantage of this paper is to examine and understand the working principle of these entire algorithm and its advantages.

REFERENCES

[1] Z.Chen and M.M Ali, "The Performance of TCP Congestion Control Algorithms over high speed transmission links", vol.3,pp.1371-1374

[2] Jon Postel, " Transmission Control Protocol, September 1981", RFC 793.

[3] Sally Floyd and Van Jacobson, " Random Early Detection Gateways for Congestion Avoidance" IEEE/ ACM Transactions on Networking

[4] Van Jacobson, " Congestion Avoidance and Control. Computer Communications Review", Volume 18 number 4, pp. 314-329, August 1988.

[5] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", January 1997, RFC 2001..

[6] Amit Aggarwal, Stefan Savage, and Thomas Anderson. Understanding the Performance of TCP Pacing, March 30, 2000, IEEE InfoCom 2000.

[7] Hamed Vahdet Nejad, Mohammad Hossien Yaghmaee, Hamid Tabatabaee, "Fuzzy TCP: Optimizing TCP Congestion Control", IEEE, 2006.

[8] Lisong Xu, Khaled Harfoush, and Injong Rhee: "Binary Increase Congestion Control for Fast, Long Distance Networks", 2003.