Analysis of Different Congestion Control Formats for TCP/IP Networks

Gesu Thakur
Research Scholar at Singhania University, Pacheri Bari (Jhunjhunu), Rajasthan

Abstract: Network congestion is biggest challenge of this era. There are so many methods for controlling congestion for TCP connections. In this paper I discuss the some of these formats. The Slow Start algorithm, treat the network as a end to end packet transition so we can only detect congestion is through packet loss and changes in round trip time, or throughput. The last two, Random Early Detection and Explicit Congestion Notification depend on the gateways to provide indications of congestion.
Keywords: TCP/IP, Connection, congestion, packet, transmission, retransmission, throughput, RED, ECN, ACK

Introduction

"Network congestion is the situation in which an increase in data transmissions results in a proportionately smaller increase, or even a reduction, in throughput."
We can control the congestion on network by the help of various congestion control algorithms. There are following criteria by which we can evaluate the performance of algorithms:
Network Performance
This is the very important criteria to set an algorithm as good. It can be measure by maximum throughput and minimum retransmission as compared to other algorithms.
Network strength
The good algorithm is judged by how well connections share the resources with other connections and whether there are any biases towards connections with certain characteristics such as burstiness.
Network Complexity
The algorithm is judged by the complexity of implementation. Algorithms with lower overhead are preferred.
Network Compatibility
We can judge an algorithm the interaction level with current scenario of technology.

We compare algorithms which use changes in performance, i.e., packet loss, increase in round trip time, change in throughput, to detect congestion. These algorithms have the advantage that they require only a new implementation of TCP and do not involve changing the network, as opposed to the algorithms discussed in the second section which require changing gateways and possibly adding fields to IP packets.
There are two types of formats we use to categorized algorithms:
1. Source based format
1.2 Slow Start algorithm
This algorithm developed by Jacobson and Karels. Before this there was no congestion control specified mechanism for TCP. This method works on principle for preservation of packets in which that the packets are entering the network at the same rate that they are exiting with a full window of packets in transit. A connection in this state is said to be in stable. If all connections are in stability, congestion collapse is unlikely. The authors identified three ways for packet preservation to be violated:
1. The connection never becomes stable.
2. After old one exit, a source sends a new packet
3. Congestion in the network prevents a connection from reaching stable.

TCP having a self clocking system it means the source first receives a ACK form old packet then it send a new packet and the rate at which the source receives ACKs is the same rate at which the destination receives packets. So the rate at which the source sends matches the rate of transmission over the slowest part of the connection.

1.2.1 Algorithm

A slow-start algorithm was developed to avoid failure of connection. This algorithm added a

congestion window. The minimum of the congestion window and the destination window is used when sending packets. Congestion window size is set to as one packet. The congestion window is then increased by one packet upon the receipt of an ACK. This would bring the size of the congestion window to that of the destination window in RTT $\log_2 W$ time, where RTT is the round-trip-time and W is the destination window size in packets.

Violation would occur if the retransmit time is too short, making the source retransmits a packet that has not been received and is not lost. What is needed is a good way to estimate RTT:

Err = Latest_RTT_Sample - RTT_Estimate

RTT_Estimate = RTT_Estimate + g*Err

Where g is a 'gain' ($0 < g < 1$) which is related to the variance. This can be done quickly with integer arithmetic. This is an improvement over the previous method which used a constant to account for variance.

Congestion avoidance takes care of violation. By the help of lost packets we can identify congestion on the network. The authors state that the probability of a packet being lost due to transit is very unusual. In addition, because of the improved round trip timer, it is a safe guess that a timeout is due to network congestion. An additive increase / multiplicative decrease policy was used to avoid congestion.

According to [8]this format suffers from oscillations when the network is overloaded. Because the window size is increased until a packet is dropped to indicate congestion, the bottleneck node is kept at maximum capacity. The window size oscillates between the maximum window size allowable by the bottleneck and half that size on timeouts. This leads to long queuing delays and high delay variation.

Wang, et. al., also point out that this format is biased toward connections with fewer hops. The additive increase / multiplicative decrease algorithm forces, eventually, the window sizes for all connections to be equal. However, this would take much iteration, in which time, the shorter connections, with shorter round trip times, would increase faster.

2. Gateway based format

There was a problem with source based (end to end) congestion control formats is that the presence of congestion is detected through the effects of congestion like packet loss, increased round trip time, changes in the throughput gradient, overflowing queues etc. There can also be a problem with fairness and non-compliant sources. To solve this problem we can place Gateways. The gateway knows how congested it is and can notify sources explicitly, either by marking a congestion bit, or by dropping packets. The main drawback to marking packets with a congestion bit, as opposed to simply dropping them, is that TCP makes no provision for it currently. Floyd in [3]states that some have proposed sending Source Quench packets as ECN messages. Source Quench messages have been criticized as consuming network bandwidth in a congested network making the problem worse.

2.1 Random Early Detection(RED)

One method for gateways to notify the source of congestion is to drop packets. This is done automatically when the queue is full. The default algorithm is when the queue is full dropping the any new packets. This is called Tail Drop. Another algorithm is when the queue is full and a new packet arrives, one packet is randomly chosen from the queue to be dropped. Floyd [4]mentions these formats in their paper. The drawback to Tail Drop and Random Drop gateways is that it drops packets from many connections and causes them to decrease their windows at the same time resulting in a loss of throughput.

Early Random Drop gateways are a slight improvement over Tail Drop and Random Drop in that they drop incoming packets with a fixed probability whenever the queue size exceeds a certain threshold. Floyd notes that none of these algorithms handled misbehaving connections well.

21.1 Algorithm

Floyd [4]proposes a new method called Random Early Detection (RED) gateways. In this method, once the average queue is above a certain threshold the packets are dropped (or marked) with a certain probability related to the queue size. To calculate the average queue size the algorithm uses an exponentially weighted moving average:

$avg = (1-w_q)avg + w_q * Queue\_Size$

The author goes into detail describing how to determine the upper and lower bounds for w_q.
The probability to drop a packet, p_b, varies linearly from 0 to max_p as the average queue length varies from the minimum threshold, minth, to the maximum threshold, maxth. The chance that a packet is dropped is also related to the size of the packet. The probability to drop an individual packet, p_a, increases as the number of packets since the last dropped packet, count, increases:
$p_b = max_b(avg\text{-}min_{th})/(max_{th}\text{-}min_{th})$
$p_b = p_b$*Packet_Size/Max_Packet_Size
$p_a = p_b/(1\text{-}count*p_b)$
In this algorithm as the congestion increases, more packets are dropped. Larger packets are more likely to be dropped than smaller packets which use less resources.

2.2. Explicit Congestion Notification

In [3]the author considers RED gateways that produce explicit congestion notification in the form of a message to the source or by marking a congestion notification bit on the packet. New guidelines are proposed by Floyd for the response of the source to an ECN. Because an ECN is not an indication of queue overflow it would be undesirable for the source to treat an ECN as it would a dropped packet and start TCP's conservative congestion control mechanism. The guidelines are as follows:

- The response to ECN should, over time, be similar to that of a lost packet.
- The instant response should be much less conservative than the response to a dropped packet
- The receipt of a single ECN should set off a response.
- The source should respond to an ECN only once per RTT.
- The source should follow the existing rules for sending packets in response to acks.
- The source should slow-start and retransmit on a dropped packet.

2.2.1 Algorithm

The algorithm implemented in the simulations in [3]is as follows. When a source receives an ECN message and no responses to congestion have been made in the last round trip time, both the congestion window and the slow start threshold are halved. Note that since no packet was lost the source shouldn't slow start.
If the source receives three duplicate ACKs indicating a lost packet and it has not responded to congestion in the last round trip time, it should follow the Fast Retransmit and Fast Recovery procedures. The source shouldn't respond to an ECN or another set of duplicate ACKs until all packets which were outstanding at the time it first responded to congestion have been acked.
If the source has responded to an ECN and soon after receives three duplicate acks it should retransmit the dropped packet but it should not change the congestion window or the slow start threshold because that has already been done on the receipt of the ECN.
In the both the LAN and the WAN scenarios, the throughput of bulk data transfers was high and the delay of telnet packets was low for the ECN capable RED. The ECN capable RED shows much improvement over the other formats when telnet delay is compared.

Floyd identifies two potential problems with their format: non-compliant sources and the loss of ECN messages. The problem of a non-compliant source is a hazard for any congestion control algorithm. If there can be a source which ignores ECN messages, there could also be a source that does not respond to packet drops. With a congestion control format that uses packet drops to control congestion, any source interested in maximizing throughput cannot ignore packet drops, however. The author states non-compliant connections can cause problems in non ECN environments as well as in ECN environments. With regards to ECN message loss, since the RED gateway continually sets ECN bits while congestion persists the loss of an ECN message will not fundamentally affect the algorithm
One major hurdle to the application of this algorithm to TCP is the incremental deployment of ECN capable gateways and sources. One proposed solution is to provide two bits in the header to indicate ECN compliance and the presence of congestion. This can also be done with one bit, where "off" represents ECN capability and "on" would represent either no ECN capability or congestion notification. When a gateway marks a packet with the bit "off" it simply switches the bit

"on." If the gateway wants to mark a packet with the bit "on" it simply discards it. Notice that the one bit format would not work for two way traffic where data packets travel in one direction and acks in the other. If a congested node sets the ECN bit for one packet, as the ack returns to that node, it will be discarded.

Conclusion

We have described different algorithms for congestion control for TCP. The first of these were improvements upon the implementation of previous technology. With these source based formats, it is easy for put into practice a new format and uses it without having to worry about others who may not be using the same execution. Whereas with the gateway based formats, global standardization is needed to implement new formats. Furthermore, it would seem that algorithms which work on gateways would require more overhead, per connection, than the source based format, by considering how many computations need to be done per packet for a multi-hop connection. However, algorithms which operate at the location of the congestion, i.e., gateway based, should provide better feedback than algorithms which have to estimate the degree of congestion from performance.

While improvements may still be made to the source based formats, it would seem that the greatest improvement to congestion control would come from gateway based formats, or a combination there of, such as [4]. The likelihood of such formats being implemented is low due to the problem related to converting existing tools.

References:

1. Lawrence S. Brakmo and Sean W. O'Malley, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," in SIGCOMM '94 Conference on Communications Architectures and Protocols, (London, United Kingdom), pp. 24-35, Oct. 1994.
2. Douglas E. Comer, Internetworking with TCP/IP Volume I, Englewood Cliffs, New Jersey: Prentice Hall, 1991.
3. Sally Floyd, "TCP and Explicit Congestion Notification," ACM Computer Communication Review, vol 24, pp. 8-23, Oct. 1995.
4. Sally Floyd and Van Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Transactions on Networking, vol. 1, pp. 397--413, Aug. 1993.
5. Van Jacobson, "Congestion Avoidance and Control," ACM Computer Communication Review, vol. 18, pp. 314--329, Aug. 1988. Proceedings of the Sigcomm '88 Symposiumin Stanford, CA, August, 1988.
6. Raj Jain, "A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks," ACM Computer Communication Review, vol. 19, pp. 56-71, Oct 1989.
7. Zheng Wang and Jon Crowcroft, "Eliminating Periodic Packet Losses in the 4.3-Tahoe BSD TCP Congestion Control Algorithm," ACM Computer Communication Review, vol. 22, pp. 9--16, Apr. 1992.
   This paper describes the DUAL congestion control mechanism. This format modifies the Slow Start format to eliminate oscillatory behavior.
8. Zheng Wang and Jon Crowcroft, "A New Congestion Control Format: Slow Start and Search (Tri-S)," ACM Computer Communication Review, vol. 21, pp 32-43, Jan 1991