TCP CONGESTION CONTROL

Networking

MARCH 7, 2020

AMERICAN UNIVERISTY OF BEIRUT
Dr. Jad Matta

TCP Congestion Control

The internet has become the fastest growing technology of all time. So far, the internet is still chugging along, but a good question to ask is “Will it continue to do so?”

TCP is easily the most widely used protocol in the transport layer on the internet (Ex. HTTP, TELNET, and SMTP), plays an integral role in determining overall network performance.

TCP Flow Control

One of TCP’s primary functions is to properly match the transmission rate of the sender to that of the receiver and the network.

TCP’s 16-bit window field is used by the receiver to tell the sender how many bytes of data the receiver is willing to accept. Since the window field is limited to a maximum of 16 bits, this provides for a maximum window size of 65,535 bytes.

The windows size advertised by the receiver tells the sender how much data, starting from the current position in the TCP data byte stream can be sent without waiting for further acknowledgments. As data is sent by the sender and then acknowledged by the receiver, the window slides forward to cover more data in the byte stream. This concept is known as “sliding window”.

Retransmissions, Timeouts and Duplicate Acknowledgements

TCP is relegated to rely mostly upon implicit signals it learns from the network and remote host. TCP must make an educated guess as to the state of the network and trust the information from the remote host in order to control the rate of data flow.

A sender’s implicit knowledge of network conditions may be achieved through the user of a timer. For each TCP segment sent the sender expects to receive an acknowledgement within some period of time otherwise an error in the form of a timer expiring signals that something is wrong.

Somewhere in the end-to-end path of a TCP connection a segment can be lost along the way. Often this is due to congestion in network routers where excess packets must

be dropped. TCP not only must correct for this situation, but it can also learn something about network conditions from it.

Fundamental to the timeout and retransmission strategy of TCP is the measurement of the round-trip-time between two communicating TCP hosts. The round-trip time may vary during the TCP connection as network traffic patterns fluctuate and as routes become available or unavailable.

TCP keeps track of when data is sent and at what time acknowledgements covering those sent bytes are returned. TCP uses this information to calculate an estimate of round trip time. As packets are sent and acknowledged, TCP adjusts its round-trip time estimate and uses this information to come up with a reasonable timeout value for packets sent. If acknowledgements return quickly, the round-trip time is short and the retransmission timer is thus set to a lower value. This allows TCP to quickly retransmit data when network response time is good, alleviating the need for a long delay between the occasional lost segment.

If a TCP data segment is lost in the network, a receiver will never even know it was once sent. However, the sender is waiting for an acknowledgement for that segment to return. In one case, if an acknowledgement doesn't return, the sender's retransmission timer expires which causes a retransmission of the segment. If however the sender has sent at least one additional segment after the one that was lost and that later segment is received correctly, the receiver does not send an acknowledgement for the later, out of order segment.

The receiver cannot acknowledgement out of order data; it must acknowledge the last contiguous byte it has received in the byte stream prior to the lost segment. In this case, the receiver will send an acknowledgement indicating the last contiguous byte it has received. If that last contiguous byte was already acknowledged, we call this a duplicate ACK. The reception of duplicate ACKs can implicitly tell the sender that a segment may have been lost or delayed. The sender knows this because the receiver only generates a duplicate ACK when it receives other out of order segments.

Standard TCP Congestion Control Algorithms

The four algorithms are:

1. Slow Start
2. Congestion Avoidance
3. Fast Retransmit
4. Fast Recovery

Slow Start

Slow Start, a requirement for TCP software implementations is a mechanism used by the sender to control the transmission rate, otherwise known as sender-based flow control. This is accomplished through the return rate of acknowledgements from the receiver. In other words, the rate of acknowledgements returned by the receiver determine the rate at which the sender can transmit data.

Congestion Avoidance

During the initial data transfer phase of a TCP connection the slow start algorithm is used. However, there may be a point during slow start that the network is forced to drop one or more packets due to overload or congestion. If this happens, congestion avoidance is used to slow the transmission rate.

In the congestion avoidance algorithm a retransmission timer expiring or the reception of duplicate ACKs can implicitly signal the sender that a network congestion situation is occurring. The sender immediately sets its transmission window to one half of the current window size, but at least two segments. If congestion was indicated by a timeout, the congestion window is reset to one segment, which automatically puts the sender into slow start mode. If congestion was indicated by duplicate ACKs, the Fast retransmit and fast recovery algorithms are invoked

Fast Retransmit

When a duplicate ACK is received, the sender does not know if it is because a TCP segment was lost or simply that a segment was delayed and received out of order at the receiver. If the receiver can re-order segments, it should not be long before the receiver sends the latest expected acknowledgment. Typically no more than one or two duplicate

ACKs should be received when simple out of order conditions exists. If however more than two duplicate ACKs are received by the sender, it is a string indication that at least one segment has been lost. The TCP sender will assume enough time has lapsed for all segments to be properly re-ordered by the fast that the receiver has enough time to send three duplicate ACKs.

Fast Recovery

Since the Fast Retransmit algorithm is used when duplicate ACKs are being received, the TCP sender has implicit knowledge that there is data still flowing to the receiver. The reason is because duplicate ACKs can only be generated when a segment is received. This is a strong indication that serious network congestion may not exist and that the lost segment was a rare event. So instead of reducing the flow of data abruptly by going all the way into slow start, the sender only enters Congestion Avoidance mode.

Rather than start at a window of one segment as in Slow Start mode, the sender resumes transmission with a larger window, incrementing as if in Congestion Avoidance mode. This allows for higher throughput under the condition of only moderate congestion.