PCap Analysis

Dhaval Bagal

October 20, 2021

1 Parsing captured packets

Every packet in the internet has a specific structure. Knowing this structure, you can write your own parser to parse pcap files. E.g: for tcp, the tcp segment starts from the 34th byte where 34th and 35th bytes denote source port, 36th and 36th byte denote destination port. However, this parsing isn't that simple as it appears to be. When packets are formed they are formed using C constructs, so your parser should understand these constructs if your parser is written in any language except C itself. E.g: python provides the **struct** library to unpack bytes from the captured packet. TCP uses big-endian format, so for python's struct library the format field should specify the format starting with > denoting big-endian format.

2 Grouping packets based on TCP flows

During the second step in the TCP handshake, the sender receives a SYN-ACK packet from the receiver. This is essentially used to identify connections in a packet trace.

On traversing the packet list, a packet with SYN-ACK flag set implies a connection is being setup. You can register the end-point addresses of this connection.

Now traverse the packet list again and for every packet find the connection addresses pair to which the packet source and destination addresses match. The packet belongs to the connection to which its addresses are matched.

3 Determining throughput

The empirical throughput for a connection can be found by simply summing up the payload sizes for all the packets and dividing it by the total number of packets sent.

4 Calculating loss rate

Number of lost packets is equivalent to the number of retransmissions in the tcp flow. Loss rate can then be defined as a ratio of number of lost packets to the total number of packets sent.

5 Estimating average round trip time

According to Karne's algorithm, RTT estimation is not done for retransmitted packets. Thus if the packets are indexed by seq number and ack number, then both the indexings should have packets with unique ack number and seq number.

For every unique ack number, we find a unique seq number which is one less than the ack number.

6 Estimating congestion window sizes

Congestion window is the amount of data TCP can send into the network before receiving an ACK.

Since seq numbers and ack numbers are byte indexed, we can find congestion window by simply subtracting the last received ack from the latest seq number sent. This gives us the size of the unacknowledged bytes which is the congestion window size.

7 Retransmissions

There are two types of retransmissions - timeout based and triple duplicate ack based.

The triple dup ack based retransmissions are easy to figure out. You just have to find the number of packets with every ack number . If this count is greater than 3, then it's a triple duplicate ack induced retransmission.

Timeout based retransmissions are difficult to figure out based on only the packet trace. However, we can find the number of timeout based retransmissions by simply subtracting the triple-dup-ack based retransmissions from the total number of retransmissions.

8 Reassembling HTTP request and response

For pipelined HTTP, reassembling request and response is not that straightforward. Pipelined HTTPs maintain session ids to map responses to the request.

However, the non-pipelined versions are fairly straightforward to reassemble. All responses belong to the latest request before them. Request and responses in non-pipelined versions of HTTP are sequential.