TCP RTO Estimation

\[ \text{RTO} = \text{weighted avg. of historical RTTs} + \text{safety factor} \]

\[ \text{Estimated RTT} (\hat{R}) \]

\[ \text{Deviation of RTT} (\hat{\Delta}) \]

\[ \hat{R}(t) = \alpha \hat{R}(t-1) + (1-\alpha) m(t) \]

\[ \alpha = 0.8 \]

\[ \hat{R}(t-1) = \alpha \hat{R}(t-2) + (1-\alpha) m(t-1) \]

\[ \hat{R}(t) = \alpha \left[ \alpha \hat{R}(t-2) + (1-\alpha) m(t-1) \right] + (1-\alpha) m(t) \]

\[ \hat{R}(t) = \alpha^2 \hat{R}(t-2) + \alpha (1-\alpha) m(t-1) + (1-\alpha) m(t) \]

\[ = 0.64 \hat{R}(t-2) + 0.16 m(t-1) + 0.2 m(t) \]
EWMA ⇒ Exponential weighted moving average.

Deviation of RTT

\[ \hat{\Delta}(t) = \alpha \hat{\Delta}(t-1) + (1-\alpha) \hat{\hat{R}}(t) - m(t) \]

smoothen the \( \hat{\Delta}(t) \) as well.

\( \hat{\Delta}(t) \) is the EWMA of the deviation of RTT.

All RTT history up until time \( (t-2) \) has a weight of 64%.
\[ RTO(t) = \hat{R}(t) + 4 \hat{\Delta}(t) \]

Retransmit timeout estimation over time.

When estimating \( \hat{R}(t) \), ignore dupACKs, and any stale ACKs that is left of Tx's window base.

③ Packets to Bytes

In reality, TCP operates as a byte stream.

MSS : maximum segment size
If Rx’s buffer fills up, it’s sad because congestion happened right at the entrance of the app^n layer.

Rx TCP includes how much space is available in its socket buffer. \( \rightarrow B \) bytes. \( \rightarrow \) ACK includes value B.

When Tx TCP gets Ack, it decides to send: \( \min \frac{3}{2}B, CW_{\max} \)

\( \rightarrow \) The CW does not change...... CW keeps following the protocol.
How is capacity divided between Alice & Bob?

If capacity = 100

<table>
<thead>
<tr>
<th>CWA</th>
<th>CWB</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>0</td>
</tr>
<tr>
<td>80</td>
<td>20</td>
</tr>
<tr>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td>65</td>
<td>32.5</td>
</tr>
<tr>
<td>50</td>
<td>17.5</td>
</tr>
</tbody>
</table>

CWA + CWB = Capacity

\[ x + y = k \]

Alice's CW = CWB

ideal operating point

Bob's CW

TCP Fairness

CNN

Fox

Flow #1

Flow #2

CWA

CWB

Alice

Bob

0

10

17.5

30

40

60

10

17.5
TCP RED (Random Early Drop) → Cross layer Hints

- Hack/workaround for the network layer to give feedback to TCP about its queue size/congestion.
- RED → Network layer drops a random data packet from each flow, so that the router queue does not fill up above some threshold (say 60%).
TCP over wireless

BER: Bit error rate

Wireless $\sim 1$ bit in error every $10^{4-6}$ bits $\Rightarrow \frac{1}{100}$ pkts corrupt

Wired $\sim 10^{-12}$ $\Rightarrow \frac{1}{\text{million}}$ pkt corrupt
TCP must differentiate between wireless losses & congestion losses (at router) because:

If congestion loss

then reduce CW

else if wireless loss

then simply retransmit & don't change CW.
Split TCP.

TCP snoop because link layer is snooping into TCP header so that it can retransmit lost packets.
End of Transport / TCP.