Tail drops

TCP recovers tail drops in two ways
1. Fast: send more new data to trigger FR (limited-transmit)
2. Slow: timeout

For Web traffic the situation is terrible
1. Often no new data to "probe"
2. Timeout is slow and has collateral damage
   a. RTO is not seasoned yet
   b. Retransmit & slow-start from cwnd of 1
3. Tail drops are very common
   a. 70% losses on Google.com are recovered by timeout

Idea: within 1-2 RTTs, retransmit the last packet to trigger FR
TLP example

Client

1

2.RTT

loss probe: 10

ACK: 5, SACK: 10

fast retrans: 6

ACK: 10

cwnd == 7

Server

cwnd == 10

ACK: 5

ACK: 10
When to send TLP?

- TLP is scheduled only if PTO < RTO.
  
  \[ PTO = \max(2 \times \text{SRTT}, 10 \text{ms}) \]
  
  \[ = \max(2 \times \text{SRTT}, 1.5 \times \text{SRTT} + \text{WCDelAckT}) \]

- Experimenting with
  - Extend RTO to always send TLP
  - Only send TLP if PTO < RTO - SRTT
Corner case: sender with 1 packet in flight

Won't react to the single drop repaired by TLP

Solution 1: make one packet like N>1 packets
  ● Retransmits only the last byte
  ● What if the sender only send 1 byte?

Solution 2: react to the later DUPACKs by spurious TLP
  ● Complex to get right

Solution 3: don't do TLP in this case

Solution 3: just ignore it
<table>
<thead>
<tr>
<th></th>
<th>ER</th>
<th>TLP</th>
<th>RTO-restart</th>
<th>F-RTO</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scenarios</strong></td>
<td>#dupacks &lt; dupthresh</td>
<td>Tail drops</td>
<td>Tail drops</td>
<td>Timeout</td>
</tr>
<tr>
<td><strong>Idea</strong></td>
<td>Smaller dupthresh</td>
<td>Send last or new packet before RTO</td>
<td>offsetting timeout by sndbuf q delay</td>
<td>send new data on timeout</td>
</tr>
<tr>
<td><strong>Pros</strong></td>
<td>2RTT recovery time</td>
<td>3RTT recovery time</td>
<td>Shorter timeout</td>
<td>Avoid spurious timeout setting cwnd to 1</td>
</tr>
<tr>
<td><strong>Implementation Complexity</strong></td>
<td>Small - medium</td>
<td>Medium</td>
<td>Small?</td>
<td>Large</td>
</tr>
<tr>
<td>need SACK</td>
<td>no</td>
<td>yes. FACK.</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Linux default</td>
<td>Linux?</td>
<td>?</td>
<td>Linux default, FreeBSD</td>
</tr>
</tbody>
</table>
WG adoption

- Work in progress
  - Experiment with different PTOs and probes
    - A parity packet (FEC)
  - Upstream to Linux
  - A research paper
  - Merge ER, F-RTO, TLP together?

- Enough interests for WG adoption?
Detecting TLP repaired losses

- **Problem**: congestion control not invoked if TLP repairs loss and the only loss is last segment.

- **Approach 1**: Count DUPACKs for TLP
  - TLP episode: N consecutive TLP segments for same tail loss.
  - End of TLP episode: ACK above SND.NXT.
  - No loss: sender receives N TLP dupacks before episode ends.
  - Loss: sender recvs <N TLP dupacks.

- **Approach 2**: Restrict TLP retransmission to 1-byte.

- We are experimenting both
Relating TLP to RTO Restart draft

● TLP and RTO Restart are philosophically not coherent.

● View-point of TLP
  ○ Try fast recovery as far as possible, use RTO as last resort.
  ○ Push RTO farther away to be always able to schedule a TLP.
  ○ A spurious probe is less risky than a spurious RTO.

● View-point of RTO Restart
  ○ Make RTOs more "tight" while being RFC-compliant.

● Difference in scope
  ○ RTO Restart used when #outstanding segments <= 3.
  ○ TLP used only for SACK enabled connections.