Speeding up TCP's loss recovery

Tail loss probe (TLP)
TCP with forward error correction (TCP-FEC)

Nandita Dukkipati
nanditad@google.com

Linux Plumbers Conference, August 2012.
Motivation

Lossy responses last 10 times longer than lossless ones.

6.1% responses and 10% TCP connections experience losses.

30% losses recovered by TCP's fast recovery, 70% by timeouts.

Our contributions
- PRR: make fast recovery even faster. (Linux 3.2-rc1)
- TLP: convert timeouts to fast recoveries.
- FEC: 0-RTT loss recovery.
Tail loss probe (TLP)

Problem: timeouts are expensive for short transfers
- Timeout recovery is 10-100x longer compared to fast recovery
- Tail losses are the majority
  [A L *] pattern more common than [A L * S * L].

TLP key idea: convert timeouts to fast recovery
- Retransmit last segment in 2.RTT to trigger SACK information and invoke fast recovery
TLP pseudo code

**Probe timeout (PTO):** timer event indicating that an ACK is overdue.

**Schedule probe on transmission of new data in Open state:**
- Either cwnd limited or application limited.
- RTO is farther than PTO.
- FlightSize > 1: schedule PTO in \( \max(2 \cdot SRTT, 10\text{ms}) \).
- FlightSize == 1: PTO is \( \max(2 \cdot SRTT, 1.5 \cdot SRTT + WCDelAckT) \).

**When probe timer fires:**
(a) If a new previously unsent segment exists:
- Transmit new segment.
- FlightSize += SMSS. cwnd remains unchanged.

(b) If no new segment exists:
- Retransmit the last segment.

(c) Reschedule PTO.

**ACK processing:**
- Cancel any existing PTO.
- Reschedule PTO relative to time at which the ACK is received.
TLP experiments results

- 2-way experiment over 10 days: Linux baseline versus TLP.
- 6% avg. reduction in HTTP response latency for image search.
- 10% reduction in RTO retransmissions.
- 0.6% probe overhead.

![Graphs showing improvement in web search, photos, and maps with TLP over Google Instant.]
TLP properties

- Property 1: Unifies recovery regardless of loss position.
- Property 2: Fast recovery of any N-degree tail loss for any sized transaction.

<table>
<thead>
<tr>
<th>loss position</th>
<th>scoreboard after TLP ACKed</th>
<th>mechanism</th>
<th>outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>A A A L</td>
<td>A A A A</td>
<td>TLP loss detection</td>
<td>All repaired</td>
</tr>
<tr>
<td>A A L L</td>
<td>A A L S</td>
<td>Early retransmit</td>
<td>All repaired</td>
</tr>
<tr>
<td>A L L L</td>
<td>A L L S</td>
<td>Early retransmit</td>
<td>All repaired</td>
</tr>
<tr>
<td>L L L L</td>
<td>L L L S</td>
<td>FACK fast recovery</td>
<td>All repaired</td>
</tr>
<tr>
<td>&gt;=5 L</td>
<td>...L S</td>
<td>FACK fast recovery</td>
<td>All repaired</td>
</tr>
</tbody>
</table>
Detecting repaired losses: basic algorithm

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

- Basic idea
  - TLP episode: $N$ consecutive TLP segments for same tail loss.
  - End of TLP episode: ACK above SND.NXT.
  - Expect to receive $N$ TLP dupacks before episode ends

- Algorithm is conservative: cwnd reduction can occur with no loss.
  - Delayed ACK timer.
  - ACK loss.
**TCP with forward error correction (TCP-FEC)**

- **Goal:** reduce tail latency via 0-RTT loss recovery.

- **Key design aspects**
  - FEC is integrated with TCP.
  - Encoding scheme.
  - Signaling of encoded segments.
  - Congestion response.
  - Middlebox considerations.

---

**TCP-FEC example**

- **Client**
  - SYN + FEC option
  - SYNACK + FEC option
  - DATA/ACK + FEC option

- **Server**
  - cwnd=10
  - 1 2 \(\cdots\) 10
  - FEC
  - ACK 9
  - ACK 10

- **Recover**
  - cwnd=7
FEC encoding approach

- Simple XOR based checksum encoding.
- Data encoded in blocks of MSS size bytes.
  - Robustness against repacketizations and variable length pkts.
- Interleaved XOR supports recovery of back-to-back losses.

Loss stats
- 40% losses (10-pkt burst) are single packet losses
- Probability of at most 2 packets lost > 0.5
Signaling FEC information

- **Basic approach**
  - Reuse SEQ number: FEC packet carries sequence# of first encoded byte.
  - New TCP option distinguishes FEC packet from original SEQ.

- **Example FEC option**
  - Option]:
    - **Kind**: 0
    - **Length**: 8
    - **FEC magic**: 16
    - **Flag**: 32
    - **Encoding range**: 40
    - **SEQ**: 10000
    - **MSS**: 1460
    - **Encoding range**: 14600
    - **XOR range**: 10000 - 24600

- **Three flavors of options**:
  - FEC negotiation in SYN/ACK handshake.
  - FEC option in FEC packets.
  - FEC option in every DATA packet.
FEC Acknowledgements

- **Successful recovery**
  - Treated similar to a successful fast retransmit.
  - Sender reduces congestion window like in fast recovery.
  - Loss recovery notification similar to explicit congestion notification (ECN).

- **Failed recovery**
  - Key: FEC packet has information on range of transmitted sequence.
  - Sender is notified of the sequence range that is lost.
  - Sender triggers fast recovery.
Examples of successful and failed recoveries

Successful:
- Client: Recovered packet 10
- Server: Received ACK 9, increased cwnd to 10
- Client: Sent ACK 10

Failed:
- Client: Recovered packet 9
- Server: Received ACK 8
- Client: Sent ACK 8
- Server: Received ACK 8
- Client: Sent ACK 10
- Server: Marked segments 9, 10 as lost
- Server: Recovery ends
### Middleboxes and alternative designs

<table>
<thead>
<tr>
<th>Middlebox issue</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rewrite ISN; preserve unknown options.</td>
<td>Relative sequence numbers.</td>
</tr>
<tr>
<td>Removal of new TCP options.</td>
<td>Negotiate option in handshake; Enable option in every packet carrying data.</td>
</tr>
<tr>
<td>Rewrite ACK number to match state of middlebox.</td>
<td>Retransmit recovered data; suppress DSACK block in ACK.</td>
</tr>
<tr>
<td>Resegmentation (split, coalesce).</td>
<td>Segments with options are OK.</td>
</tr>
<tr>
<td>Buffering OOO segments.</td>
<td>None - no worse than today.</td>
</tr>
<tr>
<td>Normalization: rewrite payloads for previously seen sequence ranges.</td>
<td>(potential: Checksum FEC payload</td>
</tr>
</tbody>
</table>

Reference: [Is it still possible to extend TCP?](https://<URL>)

**Alternative designs**
- No reuse of SEQ numbers: FEC and original have different SEQ.
- Receiver and sender maintain running checksum.
What's next?

- FEC prototype ~1500 LoC.
- Experiments with FEC.
  - Impact on Web page download time.
  - FEC performance in mobile networks.
- Pursue IETF standardization.
- FEC should eventually replace TLP.
- Near term
  - TLP to net-dev.