Loss Tolerant TCP (LT-TCP): Implementation and Evaluation

Koushik Kar
Rensselaer Polytechnic Institute

Bishwaroop Ganguly
MIT Lincoln Labs

Nathan Hourt
Rensselaer Polytechnic Institute
Outline

❖ Motivation

❖ LT-TCP overview

❖ Performance experiments and results

❖ Ongoing efforts and Future directions

❖ Short demo
LT-TCP: History & Acknowledgements

- Protocol proposed in 2007; ns-2 simulation study

- Linux kernel implementation effort since 2011
  ★ Joint effort between RPI and MIT Lincoln Labs

- Key collaborators:
  ★ Shiv Kalyanaraman (RPI; now at IBM), K.K. Ramakrishnan (AT&T)
  ★ Vijay Subramanian, Vicky Sharma, Brian Molnar, Buster Holzbauer, Nico Sayavedra, Jeff Wright, Jay Chamberlain, Kevin Battle (RPI students)
TCP under Lossy Conditions

- **Observations:**
  - Drastic falloff in performance with PER
  - Performance very bad for high loss, delay:
    - 5%+ loss rate
    - 100 ms+ RTT

- **Causes:**
  - TCP can not distinguish between congestion loss and link loss
  - Backs off on each loss
  - Recovers from link losses through retransmissions

*TCP-SACK Degradation with increased erasure rate and RTT (i.i.d. erasure probabilities. 10 MB/s capacity, one flow)*
How to fix TCP?

- We have proposed Loss Tolerant TCP (LT-TCP)
- Key ideas:
  - Use Explicit Congestion Notification (ECN)
    - TCP-like congestion control algorithm, but only responsive to ECN, not arbitrary losses
  - Use Forward Error Correction (FEC) to correct for erasures
    - Proactive FEC (PFEC): sent pre-emptively to minimize recovery latency
    - Reactive FEC (RFEC): sent later as required (i.e. PFEC proves insufficient)
    - Use loss estimation for FEC provisioning
  - Separation of reliability and congestion control
    - The reliability mechanism (FEC provisioning) can be viewed as “sitting above” the window control mechanism
- We have implemented LT-TCP as a peer to TCP in the Linux kernel
Key Considerations for Robust Transport

- Robust to difficult (e.g. lossy, long delay, bandwidth-limited) networks
  - MANET, Airborne, SATCOM
- Performs in stable networks
  - Internet, high-rate links
  - Match TCP performance
- Minimal reprogramming complexity for applications
  - Low effort level for reprogramming of TCP applications
  - Minimum of network knowledge required from programmer
- End-to-end
  - Minimize support from internal network components
- Implemented in the kernel
## Related Work

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Uses ECN</td>
<td>Modifications to TCP window control</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ad hoc TCP (ATCP) (2001)</td>
<td>Uses ECN</td>
<td>Thin layer between TCP and IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCP Westwood (2001)</td>
<td>Send-side b/w estimation from ACK return rate</td>
<td>Largely similar to TCP Reno</td>
<td></td>
<td></td>
<td>Loss rate based window adaptation</td>
<td>In kernel</td>
</tr>
<tr>
<td>TCP+ adaptive FEC (2004)</td>
<td>Proactive and reactive FEC</td>
<td>Adds a redundancy layer on TCP</td>
<td></td>
<td></td>
<td>Loss estimate based FEC provisioning</td>
<td></td>
</tr>
<tr>
<td>RFC 5740 (NORM) (2009)</td>
<td>Mainly reactive FEC, proactive optional</td>
<td>Congestion control options</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coded TCP (CTCP) (2012)</td>
<td>RTT Estimation</td>
<td>Proactive and reactive FEC</td>
<td>Alternative congestion control</td>
<td>Loss estimate based FEC provisioning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LT-TCP (2013)</td>
<td>Uses ECN</td>
<td>Proactive and reactive FEC</td>
<td>Behaves as TCP-SACK at zero loss rates</td>
<td>Loss estimate based FEC provisioning</td>
<td>Research Implementation</td>
<td></td>
</tr>
</tbody>
</table>
LT-TCP: Proactive and Reactive FEC

Properties:

★ Data encoded in blocks
  ◦ Erasure coding used

★ Data + PFEC sent in the initial transmission

★ Received data + PFEC + RFEC used to recover original data
  ◦ Block recoverable as long as the number of packets (Data or PFEC/RFEC) received is no less than the number of data packets in block

★ Receiver feedback used to compute loss estimate
  ◦ Used to determine how much PFEC, RFEC should be sent
LT-TCP Components

Application Data

MSS Adaptation

Granulated Window Size

Block Parameters (n,k)

FEC Computation

Scheduler
Determine type of packet to send

Loss Estimate/Block Loss Feedback

Reactive FEC

Proactive FEC

Data Packets

Block
LT-TCP Testbed

- **lttcp1** – Gentoo 1.12.14
  - Kernel 2.6.26.5

- **lttcp2** – Fedora 18
  - Kernel 3.6.10

- **lttcp3** – Gentoo 1.12.14
  - Kernel 2.6.26.5

All nodes: 3.6 GHz, 2 GB RAM
All interfaces are GbE
Overview: Set of 10MB file transfer results over the same testbed for three transport protocols

- TCP-SACK
- LT-TCP
- NORM

Parameters

- Packet erasure rate (correlated, uncorrelated)

Configuration

- No congestion
- NORM protocol was parameterized with line rate of testbed
NORM Details

- Transport protocol for both multicast and unicast proposed and implemented by Naval Research Laboratory (NRL)
  - Provides robust performance in the presence of packet losses
  - Implemented as user-space code
  - Can be called as a library or in “proxy” mode; we used library
  - Download: src-norm-1.5b1.tgz; Site: http://downloads.pf.itd.nrl.navy.mil/norm/
    - Used normFileSend.cpp, normFileRecv.cpp applications

Summary

- Uses FEC to repair errors, FEC also sent proactively in implementation
- Has some form of congestion control (not used here)
- Leverages user-supplied information for flow control

At high loss rates, TCP-SACK performance is extremely poor/crashes; NORM is a better performance comparison candidate
## Performance under Correlated Losses

<table>
<thead>
<tr>
<th>Erasure Rate [(E_{\text{Uncorr}})]</th>
<th>LT-TCP</th>
<th>NORM</th>
<th>TCP-SACK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Uncorrelated</td>
<td>E=2</td>
<td>E=5</td>
</tr>
<tr>
<td>0% [N/A]</td>
<td>.17</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>5% [1.05]</td>
<td>.46</td>
<td>.45</td>
<td>.44</td>
</tr>
<tr>
<td>10% [1.18]</td>
<td>.62</td>
<td>.68</td>
<td>.68</td>
</tr>
<tr>
<td>20% [1.25]</td>
<td>1.18</td>
<td>1.12</td>
<td>1.25</td>
</tr>
</tbody>
</table>

*Transfer time results for 10MB file transfer (seconds)*

*Averaged of completed trials; some did not complete*
SATCOM Configuration Testbed

![Diagram of SATCOM Configuration Testbed](image)

- **Ittcp1** – Gentoo 1.12.14, Kernel 2.6.26.5
- **Ittcp2** – Fedora 18, Kernel 3.6.10
- **Ittcp3** – Gentoo 1.12.14, Kernel 2.6.26.5

- **GbE Switch Data Network**
  - 125ms delay

- **Control Network**
- **Loss process (via NetEm)**

---

Rensselaer
# Performance under Long Delays

<table>
<thead>
<tr>
<th>Erasure Rate</th>
<th>LT-TCP</th>
<th>NORM</th>
<th>TCP-SACK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Uncorrelated</td>
<td>E=2</td>
<td>E=5</td>
</tr>
<tr>
<td>0%</td>
<td>3.24</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>5%</td>
<td>3.75</td>
<td>3.69</td>
<td>3.44</td>
</tr>
<tr>
<td>10%</td>
<td>3.74</td>
<td>3.74</td>
<td>3.75</td>
</tr>
<tr>
<td>20%</td>
<td>4.36</td>
<td>4.30</td>
<td>4.17</td>
</tr>
<tr>
<td>30%</td>
<td>4.80</td>
<td>4.68</td>
<td>4.62</td>
</tr>
<tr>
<td>40%</td>
<td>4.81</td>
<td>4.92</td>
<td>5.04</td>
</tr>
<tr>
<td>50%</td>
<td>5.83</td>
<td>5.81</td>
<td>5.93</td>
</tr>
</tbody>
</table>

Transfer time results for 10MB file transfer (seconds)

*Average of completed trials; some did not complete
Summary and Directions

- LT-TCP implementation/evaluation summary
  - Familiar socket programming model
  - File transfer performance robust to loss rate, loss correlation
  - File transfer performance robust to long RTT
  - Comparisons to TCP-SACK, NORM (plan to do CTCP soon)

- Ongoing efforts and future directions
  - Completion of portability upgrade
  - Testing of ECN reaction code
  - Exploration of alternate congestion control techniques
  - Integration with applications and performance testing

- Demo
  - Image (file) transfer comparison between TCP and LT-TCP
Thank you!

Questions?