Improving the Robustness of TCP to Non-Congestion Events

Presented by:
Sally Floyd
floyd@acm.org

For the Authors:
Sumitha Bhandarkar
A. L. Narasimha Reddy
{sumitha,reddy}@ee.tamu.edu
Outline

• Problem Statement
• Proposed Solution
  – Modifications to TCP
  – Choice of \[ \]
  – Other details
• Evaluation
  – Wireless networks with channel errors
  – Wired networks with packet reordering
• Conclusions
Problem Statement

- TCP’s behavior: On receipt of three dupacks retransmit the packet and reduce cwnd by half.
- Caveat: Not all 3-dupack events are due to congestion (Ex: channel errors in wireless networks, reordering etc.)
- Result: Sub-optimal performance in networks with non-negligible non-congestion events.
Problem Statement / Proposed Solution

Congestion Response
Delay Timer Cancelled

End-to-End RTT

No Retransmission or Window Reduction

Packet Delayed Causing Reordering

Congestion Response Delay Timer Cancelled
Proposed Solution

• TCP’s 3-dupack mechanism a heuristic
  – Allows mild reordering
  – Time to revisit this heuristic in new networks
• Proposal: Change this delay to one window (RTT)
• Allows enough time for underlying mechanisms to recover from non-congestion events.
• Essentially a tradeoff between wrongly inferring congestion and promptness of response to congestion.
Proposed Solution
(Modifications to TCP)

• Delay triggering of congestion response algorithms by \( t \) during congestion avoidance phase.
• During the delay \( t \), send one new packet for every dupack (similar to limited transmit algorithm)
• If cumulative acknowledgment received before the delay timer \( t \) expires, cancel congestion response
• Else, trigger fast retransmit/recovery.
Proposed Solution
(Choice of □)

• Should be large enough to recover from non-congestion event.
  – For the wireless network, should be at least equal to the round trip time of the wireless portion of the network.
  – For the case of reordering, no fixed lower bound.

• Should be small enough to avoid expensive RTO

• Suggested value: one RTT (end-to-end)
Proposed Solution (Other Details)

• Can be implemented based on a timer or by changing the dupthresh.

• During times of congestion, the required buffer size at receiver is twice that of unmodified TCP
  – availability of buffer space ensures maximum benefit
  – lack of buffers causes no harm

• During the delay the sender is ack-clocked, uses limited transmit
  – during non-congestion events, packets continue to be sent
  – during congestion, sending rate is at best the same as when the first dupack was received
Wireless Channel Errors -- Topology Explanation for next slide (To be removed from final version)
Wireless Channel Errors

Throughput Vs Channel Error Rate

Throughput (Mbps)

Channel Error Rate (%)

TCP-SACK
TCP-DCR
Wireless Channel Errors (cont.)

- TCP-SACK reduces sending rate for channel errors
- Result: Degraded performance as channel error rate increases
- Other concerns:
  - When available wireless bandwidth is large, TCP-SACK cannot utilize it well
  - When wireless delay is large, it takes larger time to recover from window reduction, degradation in performance more drastic
Packet Reordering -- Topology
Explanation for next slide
(To be removed from final version)
Packet Reordering

Throughput Vs Percentage of Packets Delayed

Throughput (Mbps) vs Percentage of Packets Delayed for TCP-SACK and TCP-DCR.
Packet Reordering (cont.)

- TCP-SACK wrongly infers delayed packet as congestion
- Result: Degraded performance in networks with non-negligible packet reordering
- Other concerns:
  - Requirement of near in-order delivery imposes limitations on new routing schemes.
Congestion Only -- Fairness (Topology)  
Explanation for next slide  
(To be removed from final version)
Congestion Only -- Fairness

Throughput Vs Bottleneck Link Congestion Drop Rate

Average per-Flow Throughput (Mbps)

Bottleneck Link Congestion Drop Rate (%)
Congestion Only -- Fairness (cont.)

- During the delay period TCP-DCR is still ack-clocked
- Limited Transmit is used during the delay period
- Overall protocol behavior is still AIMD

Overall performance of TCP-DCR similar to competing TCP-SACK flows for different congestion rates
Congestion Only -- Sudden Changes in Traffic
Explanation for next slide
(To be removed from final version)
Congestion Only -- Sudden Changes in Traffic

Response of TCP-SACK to Sudden Changes in Network Traffic

Response of TCP-DCR to Sudden Changes in Network Traffic

Aggregate Throughput (Mbps) (in 1 sec bins)

Time (Seconds)
Congestion Only -- Sudden Changes in Traffic (cont.)

- TCP-DCR relinquishes and reclaims bandwidth in similar fashion to TCP-SACK
Channel Errors and Congestion -- Topology
(To be removed from final version)

Same as “Congestion Only -- Fairness”
Channel Errors and Congestion

Throughput Vs Channel Error Rate with Congestion in the Network

Average Per-flow Throughput (Mbps)

Channel Error Rate (%)
Channel Errors and Congestion (cont.)

• For both TCP-SACK and TCP-DCR, $T = \frac{1}{p}$
• In case of TCP-SACK, $p = (\text{congestion loss rate} + \text{channel error rate})$
• In case of TCP-DCR, $p = \text{congestion loss rate}$
• For lower congestion rate, competing TCP-DCR flows get better throughput
• As congestion increases, difference in throughput between TCP-DCR and TCP-SACK flows decreases
Conclusions

• Significant performance improvement with non-congestion events
• Similar to unmodified versions of TCP in the absence of non-congestion events
• Simple to implement
  – Linux implementation - less than 10 lines of code changed
• Unified solution, handling multiple issues
For Further Details….

• “TCP-DCR: Making TCP Robust to Non-Congestion Losses”

• “TCP-DCR: A Novel Protocol for Tolerating Wireless Channel Errors”