Using TCP Selective Acknowledgement (SACK) Information to Determine Duplicate Acknowledgements for Loss Recovery Initiation

<draft-ietf-tcpm-sack-recovery-entry-01>

Ilpo Järvinen

Markku Kojo

IETF-77, Anaheim March 23, 2010

An Alternative Algorithm to Trigger Fast Retransmit

- Use SACK information to determine the out-of-order segments successfully arrived at the receiver, instead of simply counting dupACKs
- More timely triggering of Fast Retransmit in case of
  - ACK losses
  - ACK reordering
  - Delayed ACKs are in use (tend to conceal the first dupACK)
- Reduces the risk of false Fast Retransmits due to
  - Segment duplication
  - Out-of-window segments
- Also allows Limited Transmit for each full segment that has left the network
  - keeps ACK clock running more accurately

Current Progress

- Changes from draft-jarvinen-tcpm-sack-recovery-entry-01
  - Added resetting dupack counter as Step 3 of the algorithm
  - Added discussion on how adapted dupack counter is managed vs. traditional dupack counter
  - Completed security considerations by adding discussion on SACK splitting attacks
  - Clarifications based on feedback and general editing
- Changes from draft-ietf-tcpm-sack-recovery-entry-00
  - Redefined IsLost() to be less stricter
    - Now requires > SMSS * (DupThresh – 1) to be SACKed
    - Original IsLost() of RFC 3517 requires at least DupThresh * SMSS octets to be SACKed
  - Explicitly mention setting RecoveryPoint when entering recovery
  - Improved examples and general editing

Next Steps

- Document basically ready
- Currently planning to merge this document together with an update of RFC 3517

THANK YOU!

Backup Slides

Background

- Like with RFC 2581 (and bis), entry to recovery in RFC 3517 is based on duplicate ACKs
- SACK blocks provide more redundancy for the purpose of determining how much have been received than dupACK counter
- SACK based methods are mentioned here and there briefly
  - E.g., ackcc I-D
  - But not specified anywhere
- This I-D borrows from
  - RFC 3517
  - Linux TCP implementation
  - Forward Acknowledgment (FACK)
    - FACK different in how "holes" are counted

The Algorithm

Upon the receipt of an ACK containing SACK information:

1. If not in loss recovery, goto Step 2. Else, continue the ongoing loss recovery
2. Update scoreboard via Update () [RFC3517]
3. If ACK is cumulative ACK, reset dupACK counter
4. If new in-window SACK information arrived, count ACK as dupACK
5. If IsLost(SND.UNA) == FALSE AND less than DupThresh dupACKs arrived

   5A. Invoke optional Limited Transmit:

   Run SetPipe ()

   If cwnd – Pipe >= 1 SMSS

   If unsent data available AND rwnd allows

   Transmit as many MSS-sized segments of previously unsent data as allowed by cwnd and Pipe

   Else

   5B. Invoke Fast Retransmit and Fast Recovery

   - Continue as specified in Fast Rexmit & Fast Recovery Algorithm, e.g., RFC 3517

Potential Issues

1. One of the SACKed segments is small
   - A variant of the next case but can happen also with Nagle (thus more significant)
   - Solution: modified IsLost() in Step 5 of the algorithm to take care of this case by requiring that more than SMSS * (DupThresh – 1) to be SACKed, instead of the original requirement of having DupThresh*SMSS octets to be SACKed
     - Robust against ACK losses
     - Not problem, if the sender is packet boundary aware
2. A TCP sender sending small segments (Nagle disabled)
   - IsLost (SND.UNA) in Step 5 may fail to detect the need for loss recovery in time (on 3rd dupack) as not enough (DupThresh*SMSS + 1) octets have been SACKed
     - Packet boundary aware calculation in IsLost() calculation is immune
   - Solved by addition of Steps 3&4 and the latter condition of Step 5
     - Effectively a fallback to an adapted dupACK based algorithm
3. SACK capability misbehavior - negotiates SACK but does not send them
   - Requires RTO (No problem as SACK-based loss recovery won't work either)
4. Non-compatibility with non-SACK based Loss Recovery
   - SHOULD not be used with non-SACK based fast recovery (e.g., NewReno) as such algorithm will count late dupACKs during fast recovery as extra