An Issue in NewReno After Fast Recovery

Yoshifumi Nishida
nishida@dyyno.com
Definition of Fast Retransmit and Fast Recovery Algorithm

(Step 5 of fast retransmit fast recovery)
When a full ACK arrives after retransmission,
Exit Fast recovery
and cwnd will be:
  1) min (ssthresh, FlightSize + SMSS)
  2) ssthreth

RFC3782 Page 4 line 7:
Full acknowledgements:
If this ACK acknowledges all of the data up to and including "recover", then
the ACK acknowledges all the intermediate segments sent between the original
transmission of the lost segment and the receipt of the third duplicate ACK.
Set cwnd to either (1) min (ssthresh, FlightSize + SMSS) or (2) ssthresh,
where ssthresh is the value set in step 1; this is termed "deflating" the window. ...
... while "FlightSize" in step 5 refers to the amount of data outstanding in step 5,
when Fast Recovery is exited.)
An Issue of This Algorithm

- If we take 1), the cwnd will be $\min (\text{ssthresh}, \text{FlightSize} + \text{SMSS})$.

- This means when the FlightSize = 0, cwnd will be 1 SMSS.

- If we send only 1 packet after first recovery, the ACK might be delayed by the delayed ACK algorithm.
Possible Scenario (A)

When cwnd is small, ACK transmitted after retransmission acks all outstanding packets.
Possible Scenario (B)

When dupacks are transmitted with small advertised window, very small amount of packets are transmitted during recovery.

Sender

Receiver

New Data 1
New Data 2
New Data 3
New Data 4
New Data 5
New Data 6
New Data 7

lost Packet

Retransmit Data 1

DUP ACKS

ACK 7 with win 0
ACK 7 with win 0
ACK 7 with win 0
ACK 7 with win 0
ACK 7 with win 0
ACK 7 with win 60000

ACK for all outstanding Data

Delayed by Delayed ACK Algorithm

Only 1 packet is transmitted

cwnd=6
Other Possible Scenarios

- Similar things can happen by dupack drops or slow receivers
Proposed Solution

- Change algorithm from:
  \[
  \min (\text{ssthresh}, \text{FlightSize} + \text{SMSS})
  \]
- to:
  \[
  \min (\text{ssthresh}, \max(\text{FlightSize}, \text{SMSS}) + \text{SMSS})
  \]

- This ensures that cwnd is always larger than 2 SMSS
ns-2 modification for RFC3782

ns-2.33 seems to be slightly different from RFC3782

NewRenoTcpAgent::recv() in tcp-newreno.cc

Algorithm in red part performs: min (ssthresh, FlightSize + SMSS)
If flightsize = 0, outstanding = 1 and cwnd = 1.
However, cwnd will be increased by recv_newack_helper()

```c
void NewRenoTcpAgent::recv(Packet *pkt, Handler*){
    if (tcph->seqno() > last_ack_) {
        if (tcph->seqno() >= recover_
            || (last_cwnd_action_ != CWND_ACTION_DUPACK)) {
            if (last_cwnd_action_ == CWND_ACTION_DUPACK)
                last_cwnd_action_ = CWND_ACTION_EXITED;
            if (exit_recovery_fix_){
                int outstanding = maxseq_ - tcph->seqno() + 1;
                if (ssthresh_ < outstanding)
                    cwnd_ = ssthresh_;
                else
                    cwnd_ = outstanding;
            }
        }
    }
    firstpartial_ = 0;
    recv_newack_helper(pkt);
```
ns-2 modification for RFC3782 (2)

- ns-2.33 seems to be slightly different from RFC3782
  - In recv_newack_helper(), it calls opencwnd()
    - In opencwnd(), cwnd will be increased by 1 due to slow-start algorithm
      when cwnd = 1, it is always lower than ssthresh
      (ssthresh is never belower than 2)

    ```cpp
    void TcpAgent::opencwnd()
    {
        double increment;
        if (cwnd_ < ssthresh_)
        { /* slow-start (exponential) */
            cwnd_ += 1;
        }
    }
    
    Our modification
    - Do not call opencwnd() when cwnd is set after fast receovey
Simulation Result (1)

- **Network Configuration**

  ![Network Configuration Diagram]

  - Router drops 1 packet when \( \text{cwnd}=4 \)
  - 10Mbps bandwidth
  - 2ms delay
  - Use DelACK
  - DelACK interval: 100ms

- **Simulator:** ns-2.33 (with RFC3782 modification)

- **Simulation Scenario:** Router drops 1 packet when \( \text{cwnd}=4 \)

- **Compare two algorithms:**
  - Original RFC3782: \( \min (\text{ssthresh}, \text{FlightSize} + \text{SMSS}) \)
  - Proposed algorithm: \( \min (\text{ssthresh}, \max(\text{FlightSize}, 1) + \text{SMSS}) \)
This is very rare case. We don’t need to consider.
  -> It does not look very rare case. Even so, we had better avoid problem

ns-2 implementation is correct, we can increase cwnd after fast recovery.
  -> If so, we need to clarify it in RFC.