# An Issue in NewReno After Fast Recovery

Yoshifumi Nishida
nishida@dyyno.com

# RFC3782 Definition

- 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:

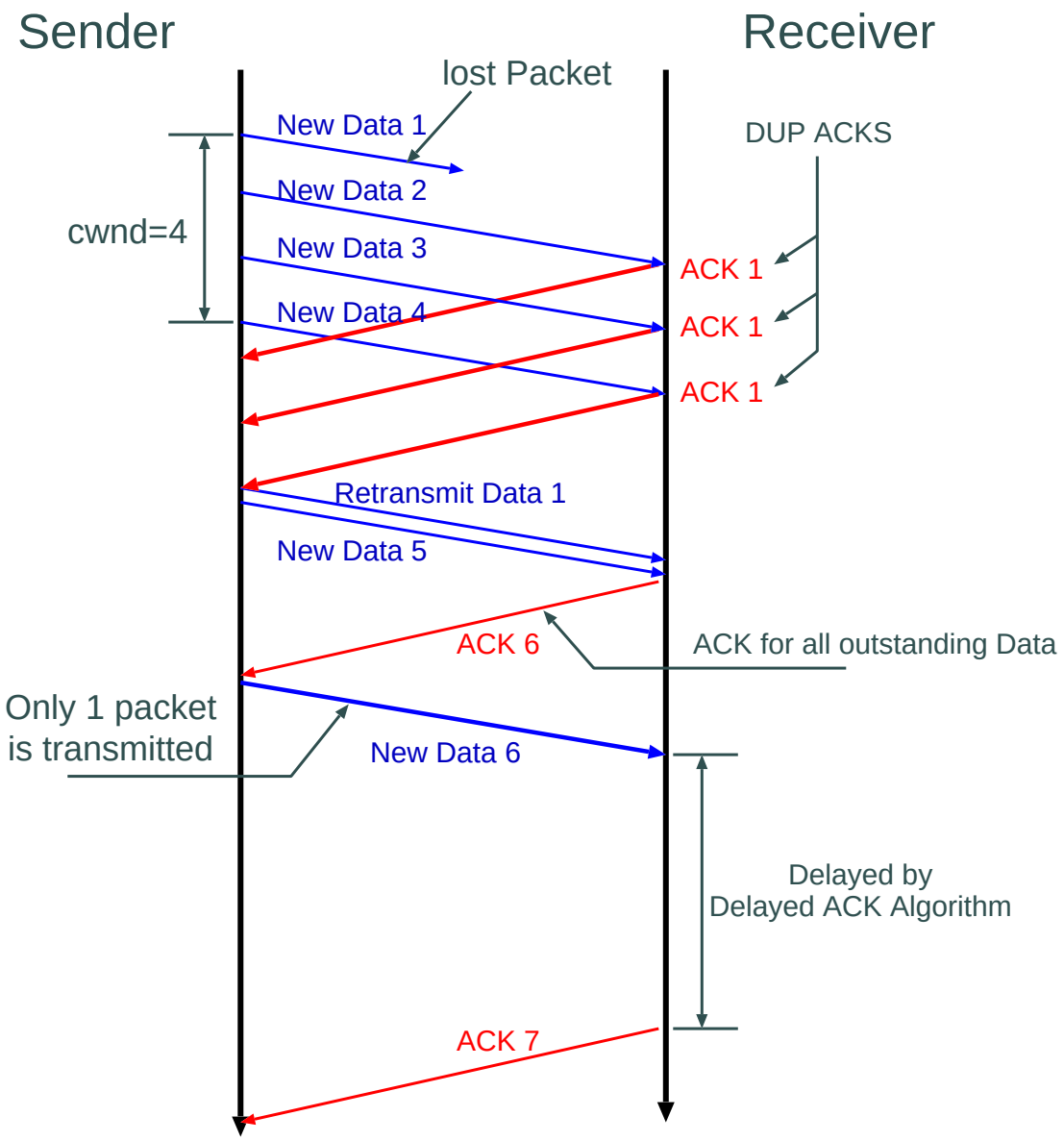
If this ACK acknowledges all of the data up to and including "recover", then the ACK acknowledges all the intermediate segments sent between the original transmission of the lost segment and the receipt of the third duplicate ACK. Set cwnd to either (1) min (ssthresh, FlightSize + SMSS) or (2) ssthresh, 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 (ssthresh, FlightSize + 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 delayed ACK algorithm.

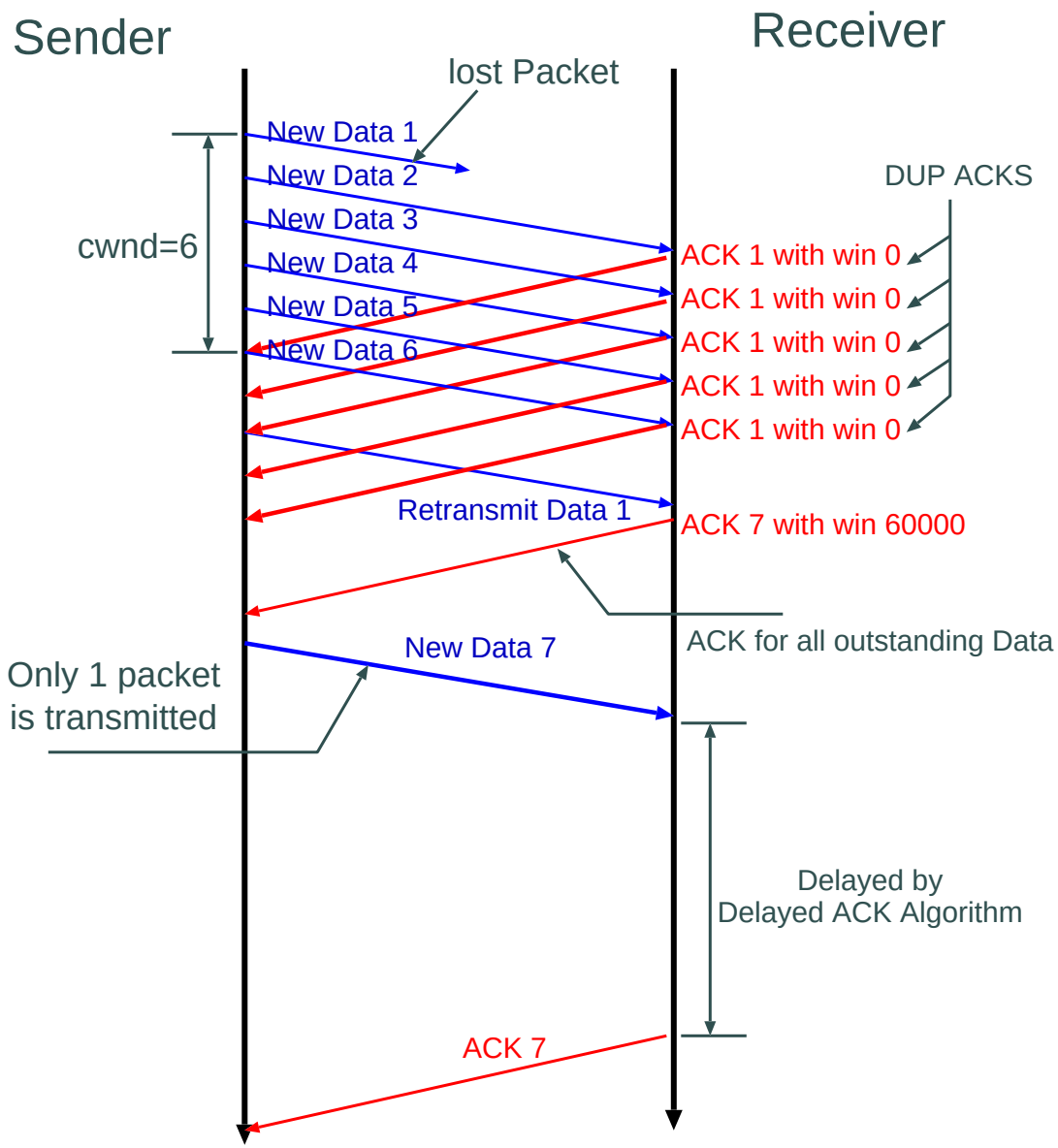
# Possible Scenario (A)

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



# Other Possible Scenarios

- Similar things can happen by dupack drops or slow receivers

# Proposed Solution

- Chage algorithm from:

  min (ssthresh, FlightSize + SMSS)

  to:

  min (ssthresh, max(FlightSize, SMSS) + SMSS)

- This ensures that cwnd is always larger than 2 SMSS

# ns-2 modification for RFC3782 (1)

- 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 flighsize = 0, outstanding = 1 and cwnd = 1.
      - However, cwnd will be increased by recv_newack_helper()

```
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)

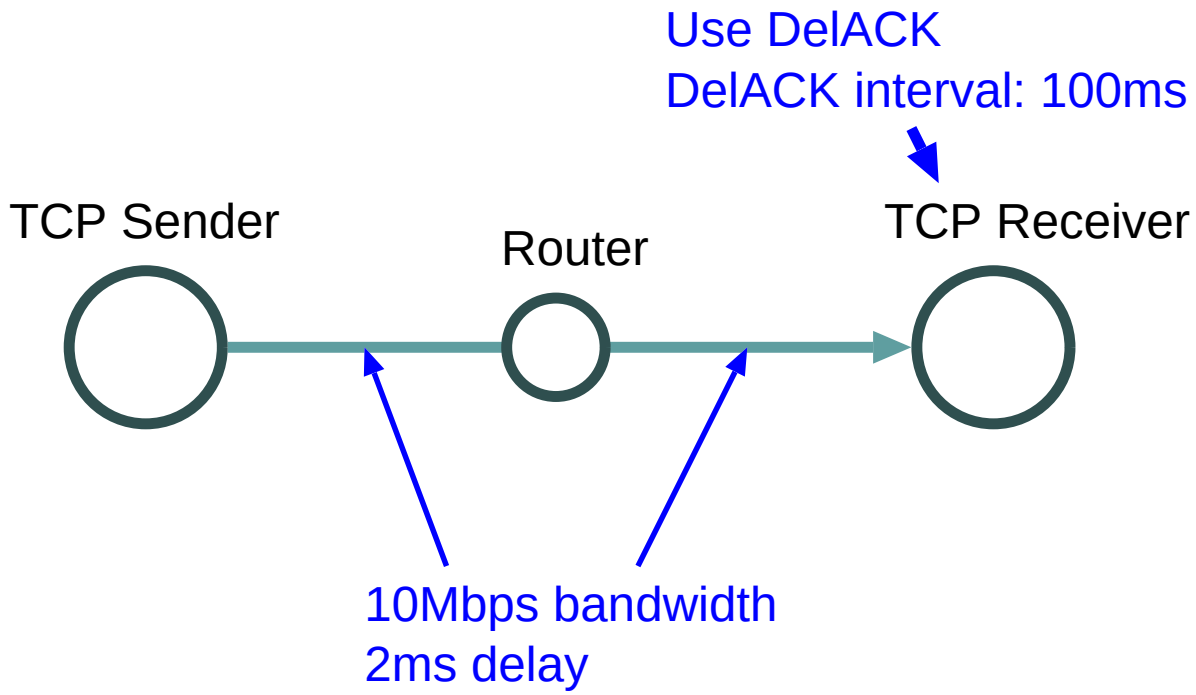
- ns-2.33 seems to be slightly different from RFC3782
  - In recv_newack_helper(), it calls opencwnd()
    - In opencwnd(), cwnd will be increased by 1 due to slow-start algorithm
      when cwnd = 1, it is always lower than ssthresh
      (ssthresh is never belower than 2)

```
void TcpAgent::opencwnd()
{
        double increment;
        if (cwnd_ < ssthresh_) {
                /* slow-start (exponential) */
                cwnd_ += 1;
        } else {
                  :
```

- Our modification
  - Do not call opencwnd() when cwnd is set after fast receovery

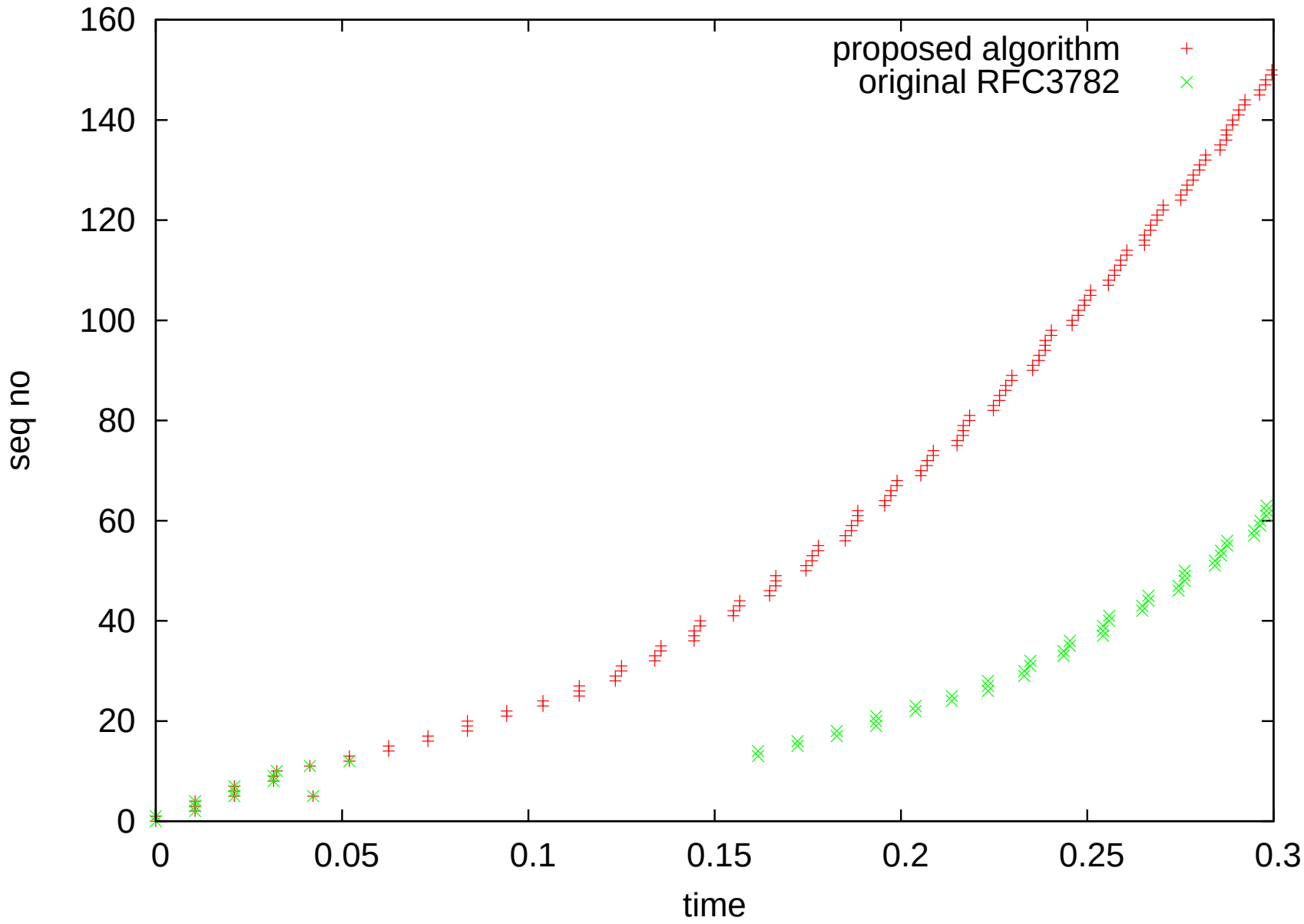
# Simulation Result (1)

- Network Configuration




- Simulator: ns-2.33 (with RFC3782 modification)
- Simuation Scenario: Router drops 1 packet when cwnd=4
- Compare two algorithms:
  - Orignal RFC3782: min (ssthresh, FlightSize + SMSS )
  - Proposed algorithm: min (ssthresh, max( FlightSize, 1 ) + SMSS )

# Simulation Result (2)



# Discussion

- 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 receovery.
  - -> If so, we need to clarify it in RFC.