TCP Friendly Rate Control (TFRC): Protocol Specification RFC3448bis

draft-ietf-dccp-rfc3448bis-03.txt

S. Floyd, M. Handley, J. Padhye, and J. Widmer

Testing and simulations from A. Sathiaseelan

December 2007,

DCCP Working Group

Changes from draft-ietf-dccp-rfc3448bis-02 reported last time:

- Added a line to the pseudocode for reducing the sending rate during idle periods during initial slow-start.
  - This fixes a problem when the sender is in initial slow-start, has an allowed sending rate less than twice the initial sending rate, and has been idle since the nofeedback timer was set. Step (1) of Section 4.4. Problem reported by Arjuna.

- Added a fix so that when datalimited and p = 0, the sender doesn't double the allowed sending rate after each feedback packet. Step (4) of Section 4.3. Problem reported by Arjuna.

Other changes from draft-ietf-dccp-rfc3448bis-02.txt:

- In a data-limited period, instead of setting the receive rate to Infinity, set it to the maximum of (X_recv, values in X_recv_set). Step (4) of Section 4.3.

- Added one line to the pseudocode in Section 4.4 on "Expiration of Nofeedback Timer" so that when the nofeedback timer expires and the sender does not have an RTT sample and has not yet received feedback from the receiver, we also look at whether the sender has been idle during the entire nofeedback interval.

Editing changes from draft-ietf-dccp-rfc3448bis-02:

- General editing from feedback from Colin Perkins.

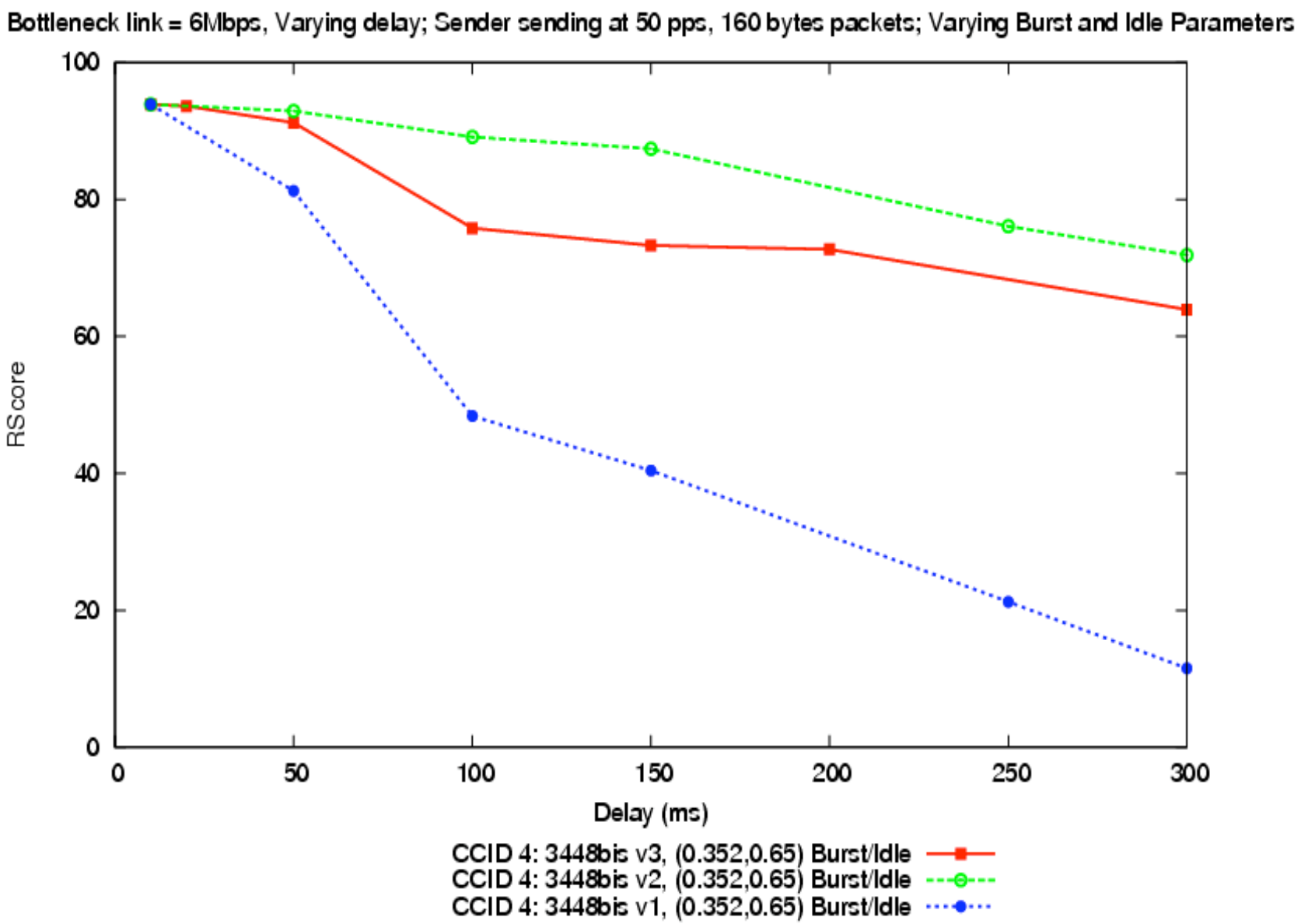
- General editing from feedback from Gerrit Renker. This includes the following:
  - Added a subsection to Section 8 on implementation issues about "Sender Behavior When a Feedback Packet is Received".
  - Moved Section 4.6.1 on "Sending Packets Before their Nominal Send Time" to Section 8 on "Implementation Issues".

- Added a subsection on "Evaluating TFRC's Response to Idle Periods" to the Appendix, encouraging future work on TFRC's responses to idle and data-limited periods.

Data-limited simulation results:

- Simulations for the change in setting the receive rate after a data-limited period. (Thanks for Arjuna.)
- No more problems with data-limited senders.
- The specific problem of increasing the allowed sending rate during data-limited intervals during slow-start is also solved.

Data-limited Simulation Results



Future Work:

- Do we need Congestion-Window-Validation type behaviour during data-limited periods?
  - Or save that for a separate document?

To implement Congestion Window Validation (modified for TFRC):

- To reduce the allowed sending rate during data-limited periods:
  - In Section 4.3, step (4):

  If (data-limited over entire interval) {
  Multiple old entries in X_recv_set by 0.85.
  Maximize X_recv_set;
  }

- This would decay the receive rate to:
  - 85% of its old value after one RTT.
  - 50% of its old value after four RTTs.

- Ready for Working Group Last Call?

Slides from last time:

Reported in previous IETFs:

- Changes from RFC 3448, in draft-ietf-dccp-rfc3448bis-00.txt
- Changes in draft-ietf-dccp-rfc3448bis-01.txt
- Reported for me in March 2007:
  - Changes in draft-ietf-dccp-rfc3448bis-02b.txt, (never submitted).
  - A slide on “things that could be done”.

Changes from draft-ietf-dccp-rfc3448bis-01.txt:

- The initial feedback packet after an idle period.
  - The mechanism for dealing with this has changed.
- Response to idle and data-limited periods.
  - The sender is not limited by the receive rate if the sender has been idle or data-limited for an entire feedback interval.
- Use of unused send credits:
  - The sender may keep unused sent credits up to one RTT.
- Many clarifications and some small changes, listed in the draft.

The initial feedback packet after an idle period:

- The mechanism for dealing with this has changed.
- The new mechanism:
  - Keep X_recv_set, with X_recv from the last two RTTs.
  - If (the entire interval covered by the feedback packet was a data-limited interval)
    - Replace X_recv_set contents by Infinity;
- Older mechanisms in older revisions:
  - If (not the first feedback packet, and not the first feedback packet after a nofeedback timer)
  - If (feedback packet reports Limited Receive Rate or sender has been data-limited over period covered by the last feedback packet)

Response to Idle and Data-Limited Periods:

Protocol	Long idle periods	Long data-limited periods
Standard TCP:	Window -> initial.	No change in window.
TCP with CWV:	Halve window (not below initial cwnd).	Reduce window half way to used window.
Standard TFRC:	Halve rate (not below 1 pkt/64 sec).	Rate limited to twice receive rate.
Revised TFRC:	Halve rate (not below initial rate).	Rate not limited to twice receive rate.

Response to Idle Periods:

- The initial version of RFC3448bis:
  - After a long idle period, the sender doesn't reduce the allowed rate below the initial rate.
  - From RFC4342.

- This is still true.
  - But the mechanisms have changed.

Response to Idle Periods:

- Current pseudocode:
  - If (X_recv < recover_rate, and sender has been idle ever since nofeedback timer was set)
    - Don't use X_recv to reduce sending rate.
- Initial versions of the draft (-00 and -01):
  - The code for dealing with idle or data-limited periods was in response to feedback packets, not in response to the nofeedback timer.
  - If (sender has been idle or data-limited)
- Later versions of the draft (-02c):
  - The code for dealing with idle or data-limited periods was moved to be in response to the nofeedback timer (as it is now).
  - If (X_recv < 4 packets per round-trip time, and sender has been idle since nofeedback timer was set)
    - Don't use X_recv to reduce sending rate.

Response to Data-Limited Periods:

- This draft:
  - Follow Standard TCP, and don't be limited by receive rate during data-limited periods.
  - If (the entire interval covered by the feedback packet was a data-limited interval) {

        Replace X_recv_set contents by Infinity;

- Earlier -00, -01, and -02c revisions:
  - During idle or data-limited periods, do be limited by receive rate, but not below the initial sending rate.
  - If (sender has been idle or data-limited within last two round-trip times)

        min_rate = max(2*X_recv, W_init/R);

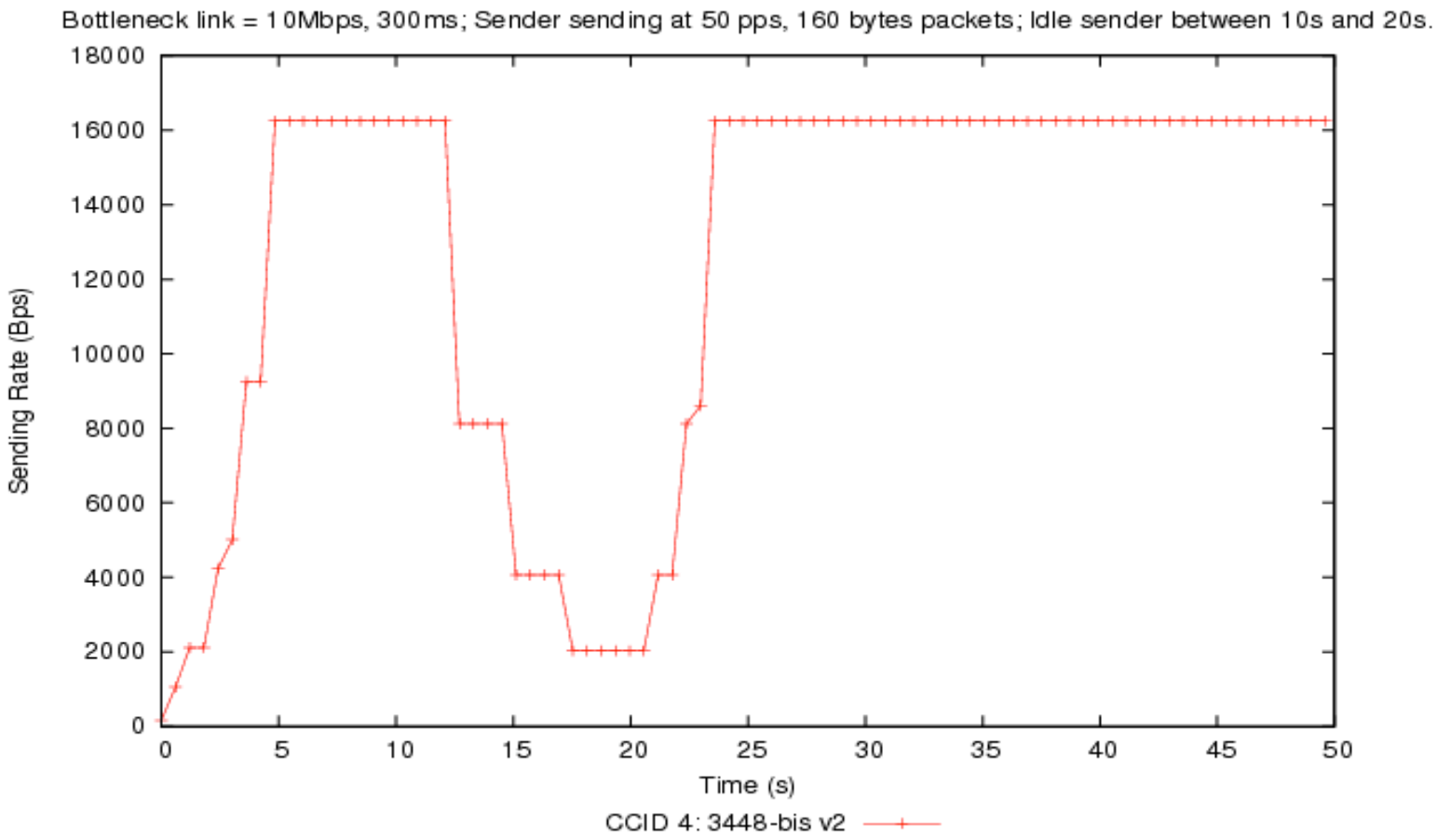
Unused send credits:

- Specified that the sender may maintain unused sent credits up to one RTT.
  - This gives behavior similar to TCP.
  - A TFRC implementation MAY limit bursts to less than one RTT, if desired.

- This was not explicitly addressed in RFC 3448, or in earlier revisions of this draft.

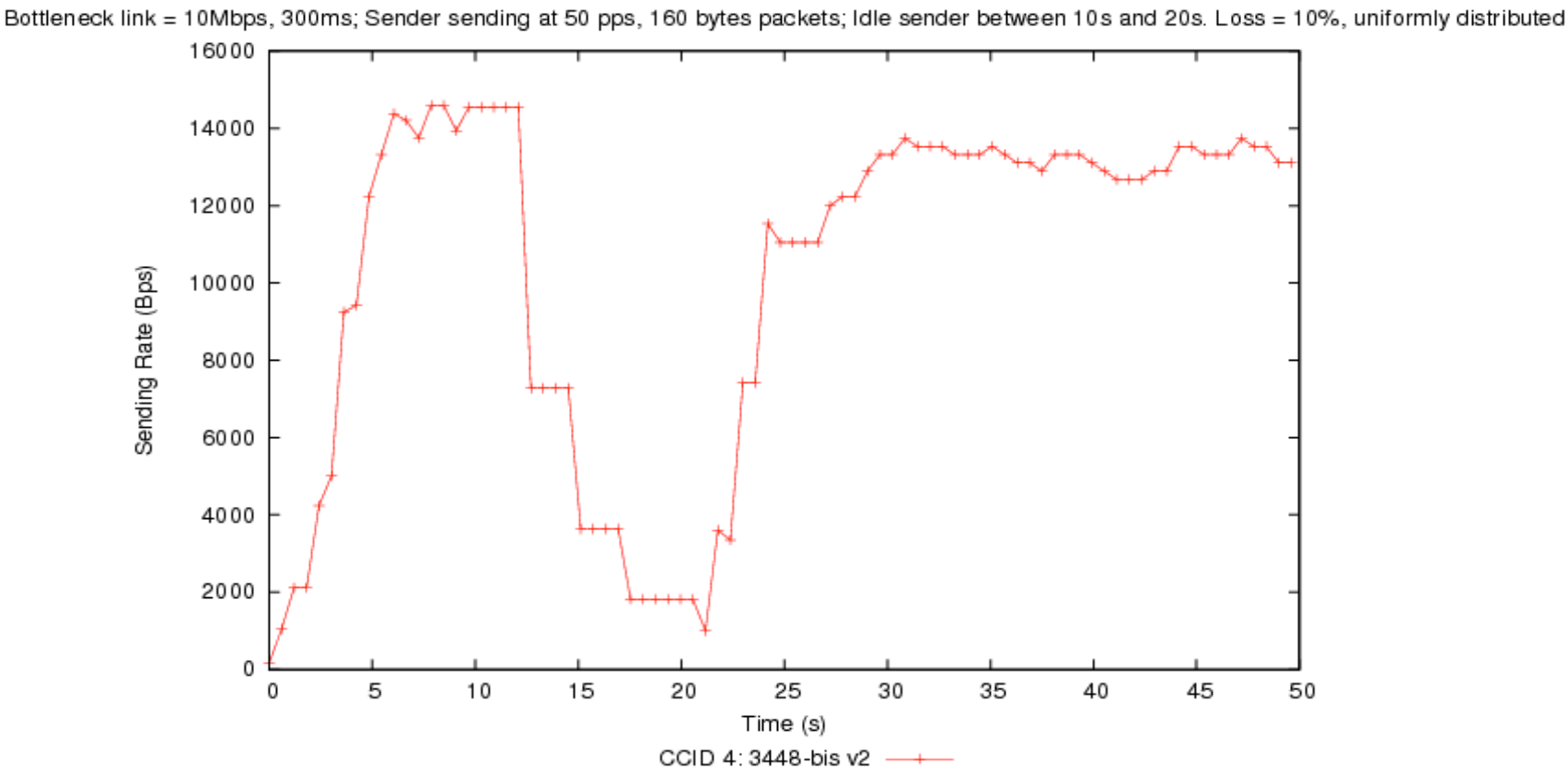
Basic Simulation Results - I

- Long idle period behaviour.
- Sending rate never reduces below recover_rate
- Low receiver rate after idle period and initial startup rectified.



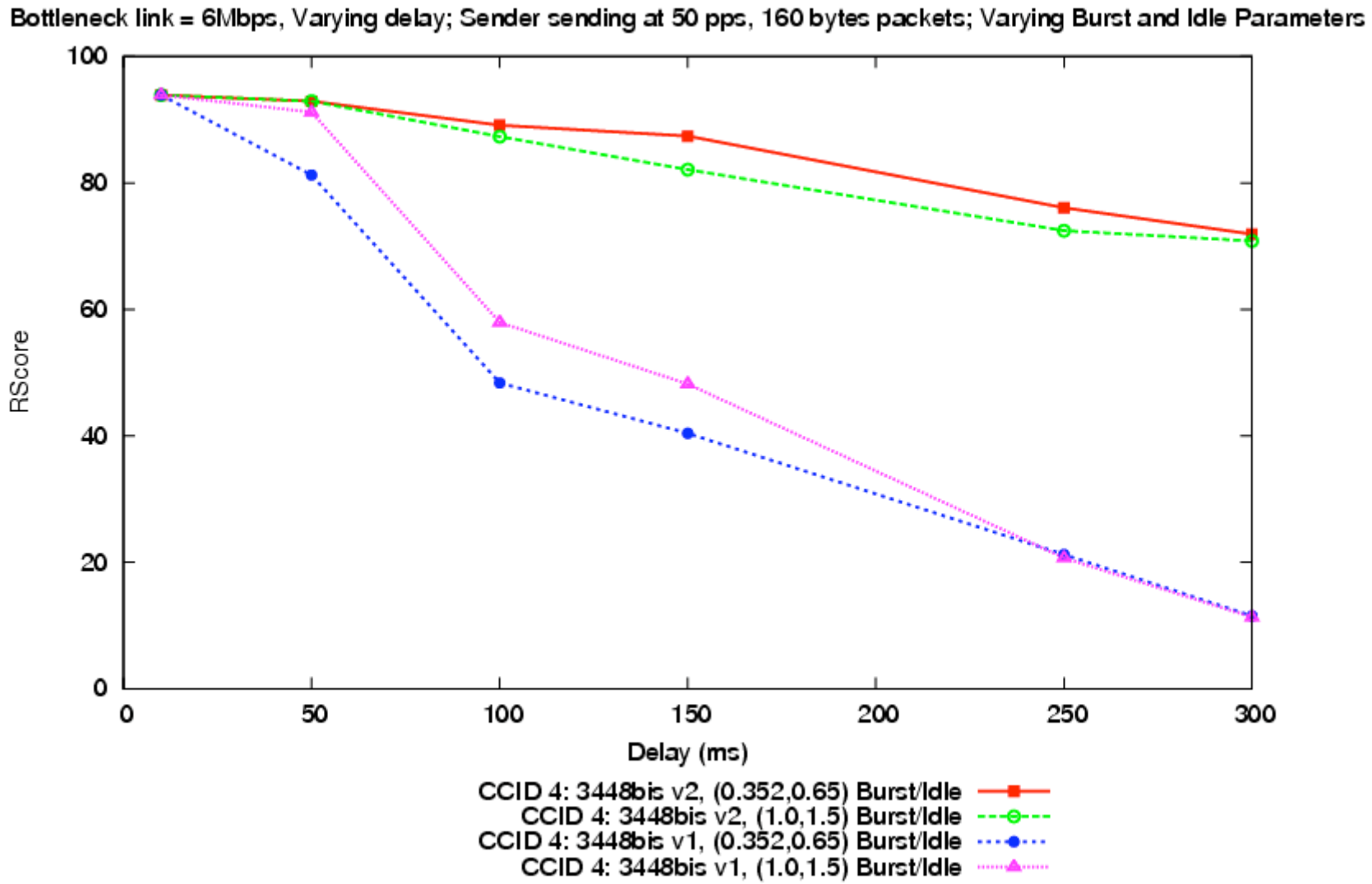
Basic Simulation Results - II

- Long idle period behaviour.
- With loss, the sending rate is limited by the throughput equation after the idle period.



Basic Simulation Results - III

- **Datalimited behaviour**
- Low receiver rate problem rectified.
- 3448-bis now good for bursty traffic : gives high perceived quality.



Change #1 from -02:

- For reducing sending rate during idle periods during initial slow-start.
- Old:

Else if (X_recv < recover_rate, and
sender has been idle ever since nofeedback timer was set)
Timer_limit is not updated;

- New:

Else if (((p>0 && X_recv < recover_rate) or
(p==0 && X < 2 * recover_rate)), and
sender has been idle ever since nofeedback timer was set)
Timer_limit is not updated;

Problem reported by Arjuna,

Change #2 from -02:

- When datalimited and p = 0, the sender still doubles the allowed sending rate after each feedback packet.
- Old code, for when (p==0):

```
Else if (t_now - tld >= R)    // initial slow-start
        X = max(min(2*X, recv_limit), initial_rate);
        tld = t_now;
```

- New code, for when (p==0):

```
Else if (t_now - tld >= R) and
   (sender was not data-limited over entire feedback interval)
        // initial slow-start
        X = max(min(2*X, recv_limit), initial_rate);
        tld = t_now;
```

Problem reported by Arjuna. (Fix not yet tested.)

Future work (in a separate document):

- "Future work could explore alternate responses to using the receive rate during a data-limited period."
  - E.g., more like TCP with Congestion Window Validation.
- At a minimum, we could have more limits on *increasing* the allowed sending rate during a data-limited period.