TCP Friendly Rate Control (TFRC): Protocol Specification
RFC3448bis

draft-ietf-dccp-rfc3448bis-02.txt
S. Floyd, M. Handley, J. Padhye, and J. Widmer
Testing and simulations from A. Sathiaeelan

July 2007,
DCCP Working Group
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:
  - 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:

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Long idle periods</th>
<th>Long data-limited periods</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Standard TCP:</strong></td>
<td>Window -&gt; initial.</td>
<td>No change in window.</td>
</tr>
<tr>
<td><strong>TCP with CWV:</strong></td>
<td>Halve window</td>
<td>Reduce window half way</td>
</tr>
<tr>
<td></td>
<td>(not below initial cwnd).</td>
<td>to used window.</td>
</tr>
<tr>
<td><strong>Standard TFRC:</strong></td>
<td>Halve rate</td>
<td>Rate limited to</td>
</tr>
<tr>
<td></td>
<td>(not below 1 pkt/64 sec).</td>
<td>twice receive rate.</td>
</tr>
<tr>
<td><strong>Revised TFRC:</strong></td>
<td>Halve rate</td>
<td>Rate not limited to</td>
</tr>
<tr>
<td></td>
<td>(not below initial rate).</td>
<td>twice receive rate.</td>
</tr>
</tbody>
</table>
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 \((X_{recv} < recover\_rate, \text{ and sender has been idle ever since nofeedback timer was set})\)
    • 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 \((X_{recv} < 4 \text{ packets per round-trip time, and sender has been idle since nofeedback timer was set})\)
    • 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)
    \[ \text{min} \_\text{rate} = \max(2 \times X_{\text{recv}}, W_{\text{init}}/R); \]
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.

Bottleneck link = 10Mbps, 300ms; Sender sending at 50 pps, 160 byte packets; Idle sender between 10s and 20s. Loss = 10%, uniformly distributed.
Basic Simulation Results - III

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

Bottleneck link = 6Mbps, Varying delay; Sender sending at 50 pps, 160 bytes packets; Varying Burst and Idle Parameters
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 ($p==0$):
  
  ```c
  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 ($p==0$):
  
  ```c
  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.