Internet Engineering Task Force
INTERNET-DRAFT
draft-ietf-dccp-tfrc-faster-restart-00.ps
Expires: January 2006

Eddie Kohler
UCLA
Sally Floyd
ICIR
21 July 2005

Faster Restart for TCP Friendly Rate Control (TFRC)

Status of this Memo

Copyright Notice

Abstract

TCP-Friendly Rate Control (TFRC) is a congestion control mechanism for unicast flows operating in a best-effort Internet environment [RFC 3448]. This document introduces Faster Restart, an optional mechanism for safely improving the behavior of interactive flows that use TFRC. Faster Restart is proposed for use with both the default TFRC and with the VoIP variant of TFRC.

Table of Contents

1. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119].

2. Introduction

In any RTT, a TFRC flow may not send more than twice X_recv, the amount that was received in the previous RTT. The TFRC nofeedback timer reduces this number by half during each nofeedback timer interval (at least four RTT) in which no feedback is received. The effect of this is that applications must slow start after going idle for any significant length of time, in the absence of mechanisms such as Quick-Start [JFAS05].

This behavior is safe, though conservative, for best-effort traffic in the network. A silent application stops receiving feedback about current network conditions, and thus should not be able to send at an arbitrary rate. But this behavior can damage the perceived performance of interactive applications such as voice. Connections for interactive telephony and conference applications, for example, will usually have one party active at a time, with seamless switching between active parties. A slow start on every switch between parties may seriously degrade perceived performance. Some of the strategies suggested for coping with this problem, such as sending padding data during application idle periods, might have worse effects on the network than simply switching onto the desired rate with no slow start.

There is some justification for somewhat accelerating the slow start process after idle periods (as opposed to at the beginning of a connection). A connection that fairly achieves a sending rate of X has proved, at least, that some path between the endpoints can support that rate. The path might change, due to endpoint reset or routing adjustments; or many new connections might start up, significantly reducing the application’s fair rate. However, it seems reasonable to allow an application to contribute to transient congestion in times of change, in return for improving application responsiveness after idle periods.

This document suggests a relatively simple approach to this problem. Some protocols using TFRC [CCID 3 PROFILE] already specify that the allowed sending rate is never reduced below the RFC-3390 sending rate of four packets per RTT during an idle period. Faster Restart specifies that the allowed sending rate is never reduced below eight packets per RTT, for small packets. In addition, because flows already have some (possibly old) information about the path, Faster Restart allows flows to quadruple their sending rate in every congestion-free RTT, instead of doubling, up to the previously achieved rate. Any congestion event stops this faster restart and switches TFRC into congestion avoidance.

3. Faster Restart Congestion Control

DRAFT DRAFT DRAFT

A connection goes "idle" when the application has nothing to send for at least a nofeedback interval (as least four round-trip times). However, when Faster Restart is used, the transport layer MUST send a "ping" packet every several round trip times, to continue getting RTT samples and some idea of the loss event rate.

The Faster Restart mechanism refers to several existing TFRC state variables, including:

R The RTT estimate; kept current during any idle periods as described above.

X The current allowed sending rate in bytes per second.

p The recent loss event rate.

X_recv
: The rate at which the receiver estimates that data was received since the last feedback report was sent. Note that this includes "ping" packets sent during idle periods (above) as well as application packets.

Faster Restart also introduces two new state variables to TFRC, as follows.

X_active_recv
: The receiver's estimated receive reported during a recent active sending period. An active sending period is a period in which the sender was neither idle nor in faster restart. It is initialized to 0 until there has been an active sending period.

T_active_recv
: The time at which X_active_recv was measured. It is initialized to the connection's start time.

Other variables have values as described in [RFC 3448].

3.1. Feedback Packets

The Faster Restart algorithm replaces for the 4th step of Section 4.3, "Sender behavior when a feedback packet is received", of [RFC 3448]. The replacement code has two goals:

1. It keeps track of the active receive rate, X_active_recv. This parameter models the connection's most relevant loss- and mark-free transmit rate, and represents an upper bound on the rate achievable through faster restart. Thus, X_active_recv is increased as the connection achieves higher congestion-free transmit rates, and reduced on congestion feedback, to prevent inappropriate faster restart until a new stable active rate is achieved. Specifically, on congestion feedback at low rates, the sender sets X_active_recv to X_recv/2; this allows limited faster restart up to a likely-safe rate, and lowers the likelihood that badly-timed transient congestion will wholly cripple the faster restart mechanism.
2. It adjusts the receive rate, X_recv, more aggressively during faster restart periods, up to the limit of X_active_recv.

The code works in three phases. The first phase determines X_fast_max, the adjusted rate at which faster restart should stop. Full faster restart up to X_active_recv should be allowed for short idle periods, but more conservative behavior should prevail after longer idle periods. Thus, if 10 minutes or less have elapsed since the last active-period measurement (T_active_recv), the code sets X_fast_max to the full value of X_active_recv. If 30 minutes or more have elapsed, X_fast_max is set to 0. Linear interpolation is used between these extremes.

The second phase adjusts X_active_recv based on the feedback packet's contents and the value of X_fast_max.

Finally, the third phase sets X based on X_fast_max, X_recv, and X_calc, the calculated send rate. Several temporary variables are used, namely X_fast_max, delta_T, F, and X_recv_limit.

```
To update X when you receive a feedback packet
----------------------------------------------
/* First phase.  Calculate X_fast_max */
/* If idle for <= 10 minutes, end faster restart at the
   full last fair rate; if idle for >= 30 minutes,
   don't do faster restart; in between, interpolate. */
delta_T := now - T_active_recv,
F := (30 min - min(max(delta_T, 10 min), 30 min)) / 20 min,
X_fast_max := F*X_active_recv.

/* Second phase.  Update X_active_recv */
If the feedback packet corresponds to an active period
      and does not indicate a loss or mark, then
   If X_recv >= X_fast_max, then
      X_active_recv := X_fast_max := X_recv,
      T_active_recv := current time.
Else if X_recv < X_fast_max and the feedback packet
      DOES indicate a loss or mark,
   X_active_recv := X_fast_max := X_recv/2,
   T_active_recv := current time.

/* Third phase.  Calculate X */
If p > 0,
   Calculate X_calc using the TCP throughput equation.
   X_recv_limit := 2*X_recv.
   If X_recv_limit < X_fast_max,
      X_recv_limit := min(4*X_recv, X_fast_max).
   X := max(min(X_calc, X_recv_limit), s/t_mbi).
Else
   If (t_now - tld >= R)
      X := max(min(2*X, 2*X_recv), s/R);
      tld := now.
```

4. Faster Restart Discussion

TCP has historically dealt with idleness either by keeping cwnd entirely open ("immediate start") or by entering slow start, as recommended in RFC 2581. The first option is too liberal, the second too conservative. Clearly a short idle period is not a new connection: recent evidence shows that the connection could fairly sustain some rate. However, longer idle periods are more problematic, and idle periods of hours would seem to require slow start. RFC 2861 [RFC 2861], which is fairly widely implemented [MAF04], gives a moderate mechanism for TCP, where the congestion window is halved for every round-trip time that the sender has remained idle, and the window in re-opened in slow-start when the idle period is over.

Faster Restart should be acceptable for TFRC if its worst-case scenario is acceptable. Realistic worst-case scenarios might include the following scenarios:

- The path changes and the old rate isn't acceptable on the new path. RTTs are shorter on the new path too, so Faster Restart clobbers other connections for multiple RTTs, not just one.

- Two (or more) connections enter Faster Restart simultaneously. The packet drop rate can be twice as bad, for one RTT, than if they had slow-started after their idle periods.
- In addition to connections Fast-Restarting, there are short TCP or DCCP connections starting and stopping all the time, with initial windows of three or four packets. There are also TCP connections with short quiescent periods (web browsing sessions using HTTP 1.1). The audio and video connections have idle periods. And the available bandwidth might vary over time, because of bandwidth used by higher-priority traffic (routing traffic, and diffserv). All of this is happening at once, so the aggregate arrival rate naturally varies from one RTT to the next. And the congested link is an access link, not a backbone link, so the level of statistical multiplexing is not high enough to make everything just look like lovely white noise.

Further analysis is required to analyze the effects of these scenarios.

We note that Faster Restart in VoIP TFRC is considerably more restrained that Faster Restart in the default TFRC; in VoIP TFRC, the sender is restricted to sending at most one packet every Min Interval. Similarly, Faster Restart in the default TFRC is more restrained that Faster Restart would be if added to TCP; TFRC is controlled of a sending rate, while TCP is controlled by a window, and could send in a very bursty pattern, in the absence of rate-based pacing.

5. Simulations of Faster Restart

TBA

6. Implementation Issues

TBA

7. Security Considerations

TBA

8. IANA Considerations

There are no IANA considerations in this document.

9. Thanks

We thank the DCCP Working Group for feedback and discussions.

Normative References

[RFC 2119] S. Bradner. Key Words For Use in RFCs to Indicate Requirement Levels. RFC 2119.

[RFC 2434] T. Narten and H. Alvestrand. Guidelines for Writing an IANA Considerations Section in RFCs. RFC 2434.

[RFC 2581] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. RFC 2581.

[RFC 3448] M. Handley, S. Floyd, J. Padhye, and J. Widmer, TCP Friendly Rate Control (TFRC): Protocol Specification, RFC 3448, Proposed Standard, January 2003.

Informative References

[CCID 3 PROFILE] S. Floyd, E. Kohler, and J. Padhye. Profile for DCCP Congestion Control ID 3: TFRC Congestion Control. draft-ietf-dccp-ccid3-06.txt, work in progress, October 2004.

[DCCP] E. Kohler, M. Handley, and S. Floyd. Datagram Congestion Control Protocol, draft-ietf-dccp-spec-08.txt, work in progress, October 2004.

[JFAS05] A. Jain, S. Floyd, M. Allman, and P. Sarolahti. Quick-Start for TCP and IP. Internet-draft draft-amit-quick-start-04.txt, work in progress, February 2004.

[MAF04] A. Medina, M. Allman, and A. Floyd, Measuring the Evolution of Transport Protocols in the Internet, May 2004, URL "http://www.icir.org/tbit/".

[P04] T. Phelan, TFRC with Self-Limiting Sources, October 2004. URL "http://www.phelan-4.com/dccp/".

[RFC 2861] M. Handley, J. Padhye, and S. Floyd. TCP Congestion Window Validation. RFC 2861, June 2000.

[RFC 3714] S. Floyd and J. Kempf, Editors. IAB Concerns Regarding Congestion Control for Voice Traffic in the Internet. RFC 3714.

Authors' Addresses

Eddie Kohler <kohler@cs.ucla.edu>
4531C Boelter Hall
UCLA Computer Science Department
Los Angeles, CA 90095
USA

Sally Floyd <floyd@icir.org>
ICSI Center for Internet Research
1947 Center Street, Suite 600
Berkeley, CA 94704
USA

Full Copyright Statement

Intellectual Property