Internet Engineering Task Force
INTERNET-DRAFT
draft-ietf-dccp-tfrc-voip-01.ps
Expires: August 2005

Sally Floyd
ICIR
Eddie Kohler
UCLA
21 February 2005

TCP Friendly Rate Control (TFRC) for Voice: VoIP Variant and Faster Restart

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 adds a VoIP variant to TFRC. TFRC was intended for applications that use a fixed packet size, and was designed to be reasonably fair when competing for bandwidth with TCP connections using the same packet size. The VoIP variant of TFRC is designed for applications that send small packets, where the design goal is to achieve the same bandwidth in bps as a TCP flow using 1500-byte data packets. The VoIP variant of TFRC enforces a Min Interval of 10 ms between data packets, to prevent a single flow from sending small packets arbitrarily frequently. This document also 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.

TO BE DELETED BY THE RFC EDITOR UPON PUBLICATION:

Changes from draft-ietf-dccp-tfrc-voip-00.txt

* Added more simulations.
* Added a Related Work section.

Table of Contents

1. Conventions 4
2. VoIP Variant Introduction 4
3. VoIP Variant Congestion Control 5
4. VoIP Variant Discussion 5
4.1. The TCP Throughput Equation 6
4.2. Accounting for Header Size 6
4.3. The VoIP Min Interval 6
5. Faster Restart Introduction 7
6. Faster Restart Congestion Control 8
6.1. Entering and Leaving Idle Periods 8
6.2. Feedback Packets 9
7. Faster Restart Discussion. 9
8. Simulations of the VoIP Variant of TFRC 10
8.1. Packet Dropping Behavior at Routers 10
9. Simulations of Faster Restart 12
10. Implementation Issues 12
11. Security Considerations. 12
12. IANA Considerations 13
13. Thanks 13
Normative References 13
Informative References 13
Authors' Addresses 13
14. Related Work on VoIP Variants of TFRC 14
Full Copyright Statement 14
Intellectual Property 14

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. VoIP Variant Introduction

This document specifies a VoIP variant for TCP-friendly rate control (TFRC) [RFC 3448]. TFRC was designed to be reasonably fair when competing for bandwidth with TCP flows, but to avoid the abrupt changes in the sending rate characteristic of TCP's congestion control mechanisms. TFRC is intended for applications such as streaming media applications where a relatively smooth sending rate is of importance.

The VoIP variant is intended for flows that need to send frequent small packets. Conventional TFRC measures loss rates by estimating the loss event ratio as described in [RFC 3448], without considering packet size. This has consequences for the rate a TFRC flow can achieve when sharing a bottleneck with large-packet TCP flows. In particular, a low-bandwidth, small-packet TFRC flow sharing a bottleneck with high-bandwidth, large-packet TCP flows may be forced to slow down, even though the application's nominal rate in bytes per second is less than the rate achieved by the TCP flows. Intuitively, this would be "fair" only if the network limitation was in packets per second (such as a routing lookup), rather than bytes per second (such as link bandwidth). Conventional wisdom is that many of the network limitations in today's Internet are in bytes per second, even though the network limitations of the future might move back towards limitations in packets per second.

The VoIP variant of TFRC described here will better support applications that do not want their sending rates in bytes per second to suffer from their use of small packets. This variant is restricted to applications that send packets no more than once every 10 ms (the Min Interval). Given this restriction, the VoIP variant effectively calculates the TFRC fair rate as if the bottleneck restriction was in bytes per second. Applications using the VoIP variant of TFRC could have a fixed packet size, or could vary their packet size in response to congestion.

The VoIP variant of TFRC is motivated by the approach in [RFC 3714], which argues that it is acceptable for VoIP flows to assume that the network limitation is in bytes per second (Bps) rather in packets per second (pps), and to have the allowed drop rates for the VoIP flow be determined by the drop rates experienced by a TCP flow with 1500-byte packets and the same sending rate in Bps as the VoIP flow. [RFC 3714] states the following:

> "While the ideal would be to have a transport protocol that is able to detect whether the bottleneck links along the path are limited in Bps or in pps, and to respond appropriately when the limitation is in pps, such an ideal is hard to achieve. We would not want to delay the deployment of congestion control for telephony traffic until such an ideal could be accomplished. In addition, we note that the current TCP congestion control mechanisms are themselves not very effective in an environment where there is a limitation along the reverse path in pps. While the TCP mechanisms do provide an incentive to use large data packets, TCP does not include any effective congestion control mechanisms for the stream of small acknowledgement packets on the reverse path. Given the arguments above, it seems acceptable to us to assume a network limitation in Bps rather than in pps in considering the minimum sending rate of telephony traffic."

Translating the discussion in [RFC 3714] to the congestion control mechanisms of TFRC, it seems acceptable to standardize a variant of TFRC that allows VoIP flows sending small packets to achieve a rough fairness with TCP flows in terms of the sending rate in Bps, rather than in terms of the sending rate in pps. This is accomplished by a simple two-line modification at the TFRC sender, as described below. No changes are required at the TFRC receiver.

However, because the bottlenecks in the network in fact can include limitations in pps as well as in Bps, we pay special attention to the potential dangers of encouraging a large deployment of best-effort traffic in the Internet consisting entirely of small packets. This is discussed in more detail in Section 4.3. In addition, as again discussed in Section 4.3, the VoIP variant of TFRC includes the limitation of the Min Interval between packets of 10 ms.

3. VoIP Variant Congestion Control

TFRC uses the TCP throughput equation given in Section 3.1 of [RFC 3448], which gives the allowed sending rate X in bytes per second as a function of the loss event rate, packet size, and round-trip time. [RFC 3448] specifies that the packet size s used in the throughput equation should be the packet size used by the application, or the estimated mean packet size if there are variations in the packet size depending on the data. This gives rough fairness with TCP flows using the same packet size.

The VoIP variant changes this behavior in the following ways.

- The nominal packet size s is set to 1460 bytes. Following [RFC 3714], this provides a goal of fairness, in terms of the sending rate in bytes per second, with a TCP flow with 1460 bytes of application data per packet.
- The allowed transmit rate X in bytes per second is reduced by a factor that accounts for packet header size. This gives the application some incentive, beyond the Min Interval, not to use unnecessarily small packets. In particular, we introduce a new parameter H, which represents the expected size in bytes of network and transport headers to be used on the TFRC connection's packets. This is used to reduce the allowed transmit rate X as follows:

  X <- X * s_true / (s_true + H),

  where s_true is the true average data packet size for the connection in bytes, excluding the transport and network headers.

  The H parameter is set to the constant 40 bytes. Thus, if the VoIP TFRC application used 40-byte data segments, the allowed transmit rate X would be halved to account for the fact that half of the sending rate would be used by the headers. Section 4.2 justifies this definition. However, a connection using the VoIP variant MAY instead use a more precise estimate of H, based on the actual network and transport headers to be used on the connection's packets. For example, a DCCP connection [DCCP] over IPv4, where data packets use the DCCP-Data packet type, and there are no IP or DCCP options, could set H to 20 + 12 = 32 bytes.
- Finally, the VoIP variant of TFRC enforces a Min Interval between packets of 10 ms. A flow that wished to exceed this Min Interval MUST use the conventional TFRC equations, rather than the VoIP variant. The motivation for this is discussed below.

4. VoIP Variant Discussion

4.1. The TCP Throughput Equation

The VoIP variant of TFRC uses the TCP throughput equation given in [RFC 3448]. As shown in Table 1 of [RFC 3714], for high packet drop rates, this throughput equation gives rough fairness with most aggressive possible current TCP: a SACK TCP flow using timestamps and ECN.

4.2. Accounting for Header Size

[RFC 3714] makes the optimistic assumption that the limitation of the network is in bandwidth in bytes per second (Bps), and not in CPU cycles or in packets per second (pps). However, some attention must be paid to the load in pps as well as to the load in Bps. Even aside from the Min Interval, the VoIP variant of TFRC gives the application some incentive to use fewer but larger packets, when larger packets would suffice, by including the bandwidth used by the packet header in the allowed sending rate.

As an example, a sender using 120-byte packets needs a TCP-friendly rate of 128 Kbps to send 96 Kbps of application data. This is because the TCP-friendly rate is reduced by a factor of s_true/(s_true + H) = 120/160, to account for the effect of packet headers. If the sender suddenly switched to 40-byte data segments, the allowed sending rate would reduce to 64 Kbps of application data; and one-byte data segments would reduce the allowed sending rate to 3.12 Kbps of application data. (In fact, the Min Interval would prevent senders from achieving these rates, since applications using the VoIP variant cannot send more than 100 packets per second.)

The VoIP variant assumes 40 bytes for the header size, although the header could be larger (due to IP options, IPv6, IP tunnels, and the like) or smaller (due to header compression) on the wire, because using the exact header size in bytes would have little additional benefit. The VoIP variant's use of an assumed 40-byte header is sufficient to get a rough estimate of the throughput, and to give the application some incentive not to use unnecessarily-many small packets. Because we are only aiming at rough fairness, and at a rough incentive for applications, the use of a 40-byte header in the calculations of the header bandwidth seems sufficient.

4.3. The VoIP Min Interval

The header size calculation provides an incentive for applications to use fewer, but larger, packets. Another incentive is that when the path limitation is in pps, the application using more small packets is likely to receive more packet drops, and to have to reduce its sending rate accordingly. That is, if the congestion is in terms of pps, then the flow sending more pps will receive more congestion indications, and have to adjust its sending rate accordingly. However, the increased congestion caused by the use of small packets in an environment limited by pps is experienced not only by the flow using the small packets, but by all of the competing traffic on that congested link. These incentives are therefore insufficient to provide sufficient protection for pps network limitations.

The VoIP variant for TFRC, then, includes a Min Interval of 10 ms. This provides additional restrictions on the use of unnecessarily many small packets.

One justification for the Min Interval is the practical one that the applications that currently want to send small packets are the VoIP applications that send at most one packet every 10 ms, so this restriction does not affect current traffic. A second justification is that there is no pressing need for best-effort traffic in the current Internet to send small packets more frequently than once every 10 ms (rather than taking the 10 ms delay at the sender, and merging the two small packets into one larger one). This 10 ms delay for merging small packets is likely to be dominated by the network propagation, transmission, and queueing delays of best-effort traffic in the current Internet. As a result, our judgement would be that the benefit to the user of having less than 10 ms between packets is outweighed by the benefit to the network of avoiding unnecessarily many small packets.

The Min Interval causes the VoIP variant of TFRC not to support applications sending small packets very frequently. Consider a TFRC flow with a fixed packet size of 100 bytes, but with a variable sending rate and a fairly uncongested path. When this flow was sending at most 100 pps, it would be able to use the VoIP variant of TFRC. If the flow wished to increase its sending rate to more than 100 pps, but to keep the same packet size, it would no longer be able to achieve this with the VoIP variant to TFRC, and would have to swich to the default TFRC, receiving a dramatic, discontinuous decrease in its allowed sending rate. This seems not only acceptable, but desirable for the global Internet.

What is to prevent flows from opening multiple connections, each with a 10 ms Min Interval, and thereby getting around the limitation of the Min Interval? Obviously, there is nothing to prevent flows from doing this, just as there is currently nothing to prevent flows from using UDP, or from opening multiple parallel TCP connections, or from using their own congestion control mechanism. Of course, implementations or middleboxes are also free to limit the number of parallel TFRC connections opened to the same destination in times of congestion, if that seems desirable. And flows that open multiple parallel connections are subject to the inconveniences of reordering and the like. But even without a mechanism to prevent flows from subverting the Min Interval by opening multiple parallel connections, it seems useful to include the Min Interval in the VoIP variant of TFRC.

5. Faster Restart 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 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. Incurring slow start on every switch between parties may cause perceived performance to seriously degrade. 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.

6. 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.

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

T_idle
: The time the connection went idle.

X_fast_max
: The rate at which to turn off faster restart; 0 if not in faster restart. Initially 0.

X_active_recv
: The rate at which packets were received in the last active sending period. An active sending period is a period in which the sender was neither idle nor in fast restart. It is initialized to 0 until there has been an active sending period.

T_active_recv
: The most recent time in an active sending period.

Several previously existing state variables are also particularly important, as follows.

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.

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

6.1. Entering and Leaving Idle Periods

When the application has nothing to send (an idle period is entered), TFRC sets T_idle := now.

When the application has something to send, TFRC uses the following code to determine whether it is leaving an idle period, and if so, how the sending rate should be adjusted. The code will use Faster Restart up to the full last fair rate after an idle period of 10 minutes or less; will not use Fast Testart after an idle period of 30 minutes or more; and interpolates between these extremes after idle periods between 10 and 30 minutes.

```
If (now - T_idle) > max(R, 1 / max(X_calc, s/t_mbi)),
   /* If idle for <= 10 minutes, end fast start at the
      full last fair rate; if idle for >= 30 minutes,
      don't do fast start; in between, interpolate. */
   delta_T := now - T_active_recv
   F := (30 min - min(max(delta_T, 10 min), 30 min)) / 20 min
   /* Initialize X_fast_max to a fraction of the last active
      rate */
   X_fast_max := F*X_active_recv
   /* Alter the cached X_recv so we start out between 4
      and 8 packets/RTT */
   X_recv := max(2*s/R, X_recv)
```

6.2. Feedback Packets

The core of the Faster Restart algorithm is a replacement for the 4th step of Section 4.3, Sender behavior when a feedback packet is received, of [RFC 3448], as follows.

```
To update X when you receive a feedback packet
----------------------------------------------
If (2*X_recv < X_fast_max) and the feedback packet
      indicates a loss or mark,
   /* Stop faster restart at the first sign of congestion */
   X_fast_max := 0,
   X_recv := X_recv/2.
If p > 0,
   Calculate X_calc using the TCP throughput equation.
   If (2*X_recv < X_fast_max),
      /* Faster restart case */
      X := max(min(X_calc, 4*X_recv), s/t_mbi).
   Else
      X_fast_max := 0,  /* Stop faster restart */
      X := max(min(X_calc, 2*X_recv), s/t_mbi).
Else
   If (t_now - tld >= R)
      X := max(min(2*X, 2*X_recv), s/R);
      tld := now.
```

7. 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.

8. Simulations of the VoIP Variant of TFRC

VoIP mode for TFRC has been added to the NS simulator, and is illustrated in the validation test "./test-all-friendly" in the directory tcl/tests.

8.1. Packet Dropping Behavior at Routers

The default TFRC, without the VoIP variant, was designed for rough fairness with TCP, for TFRC and TCP flows with the same packet size, and experiencing the same packet drop rate. When the issue of fairness between flows with different packets sizes is addressed, it matters whether the packet drop rates experienced by the flows is related to the packet size. That is, are small VoIP packets just as likely to be dropped as large TCP packets, or are the smaller packets less likely to be dropped [WBL04]? And what is the relationship between the packet-dropping behavior of the path, and the loss event measurements of TFRC?

In our simulations of TCP flows competing with a VoIP TFRC flow with smaller packets, in a scenario with a congested router with a DropTail queue, the VoIP TCP flow receives more than its fair share in bytes per second. This is the case even for a scenario where the TCP flows are the most aggressive, with SACK TCP, timestamps, and ECN.

As expected, the packet dropping behavior can be varied by varying the Active Queue Management mechanism in the router. When the routers use RED in packet mode, where each *packet* has the same probability of being dropped, the TFRC and TCP flows receive roughly the same packet drop rate. In contrast, when the routers use RED in byte mode, where each *byte* has the same probability of being dropped, the TFRC flow sees a much smaller packet drop rate than the TCP flows, in simulations with moderate levels of congestion.

N	TCP DropRate	TCP Throughput	TFRC DropRate	TFRC Throughput
10	0.010	0.71	0.010	0.14
25	0.067	0.61	0.063	0.32
50	0.180	0.37	0.162	0.57
75	0.257	0.18	0.261	0.75
100	0.295	0.15	0.407	0.82
150	0.370	0.13	0.571	0.83

Table 1: Simulation Results with Drop-Tail Queues.

Table 1 above shows the results of simulations with N TCP flows, with unlimited data to send and 1460-byte packets, competing against N VoIP TFRC flows with 100 packets per second of application data and 200-byte data packets. N ranges from 10 to 150, with a congested link of 10 Mbps, and each simulation is run for 100 seconds. Each row of the table gives the result of a single simulation, giving the packet drop rate for the TCP and TFRC flows, and the fraction of the link bandwidth used by the N TCP and the N TFRC flows respectively. The simulations in the table above use Drop-Tail queues. When N is small, congestion is low, and each VoIP TFRC flow is limited by the maximum data rate of 160 Kbps from the application. In these cases, the TCP flows receive considerably more bandwidth than the TFRC flows. When N is large, driving the packet drop rate up to 25-50%, the TFRC flows receive significantly more bandwidth than the TCP flows. In each of these simulations, the TCP and TFRC flows receive somewhat comparable packet drop rates. The SACK TCP connections in these simulations use the default parameters in the NS simulator, with Limited Transmit, and a minimum RTO of 200 ms. Adding timestamps to the TCP connection didn't change the results appreciably.

N	TCP DropRate	TCP Throughput	TFRC DropRate	TFRC Throughput
10	0.009	0.73	0.008	0.14
25	0.051	0.60	0.037	0.33
50	0.166	0.38	0.183	0.56
75	0.218	0.16	0.231	0.79
100	0.251	0.12	0.392	0.87
150	0.326	0.10	0.550	0.87

Table 2: Simulation Results with RED Queues in Packet Mode.

Table 2 shows that the simulation results for RED queues in packet mode, where each packet is dropped with the same probability, are roughly comparble to those with Drop-Tail Queues. Tables 1 and 2 suggest that the TCP response function used in TFRC could be modified to give more realistic values for TCP throughput at higher packet drop rates. We ran these simulations using ECN and TCP timestamps, with little change in the overall results.

N	TCP DropRate	TCP Throughput	TFRC DropRate	TFRC Throughput
10	0.009	0.72	0.008	0.14
25	0.049	0.60	0.020	0.33
50	0.138	0.29	0.031	0.65
75	0.181	0.12	0.168	0.83
100	0.242	0.11	0.356	0.85
150	0.323	0.09	0.536	0.87

Table 3: Simulation Results with RED Queues in Byte Mode.

Table 3 shows simulation results for RED queues in byte mode, where the router takes the packet size into account in deciding whether or not to drop a packet. Again, for higher values of N, the VoIP TFRC flow receives more than its share of the link bandwidth.

The goal of the VoIP variant of TFRC has been for the TCP flows and the VoIP TFRC flows to have rough fairness in the sending rate in bps, in a scenario where each packet receives roughly the same probability of being dropped. In a scenario where large packets are more likely to be dropped than small packets, or where flows with a bursty sending rate are more likely to have packets dropped than are flows with a smooth sending rate, flows using the VoIP variant of TFRC could receive more bandwidth than competing TCP flows.

Although the VoIP variant of TFRC doesn't require that applications are limited by a maximum sending rate, in fact VoIP flows do have such a limitation. As illustrated in the simulations by Tom Phelan, this complicates the issue of exploring the fairness between TCP and VoIP TFRC flows.

In addition, for VoIP TFRC flows with a maximum sending rate of 96 Kbps, or with a smaller maximum sending rate, VoIP TFRC only reduces the sending rate of these flows when the packet drop rate is fairly high. In this regime, the performance of TFRC is very much determined by the accuracy of the TCP response function in representing the actual sending rate of a TCP connection. In this regime of high packet drop rates, the performance of the TCP connection is very much affected by the TCP algorithm (e.g., SACK or not), by the use of timestamps and/or of ECN, by the minimum RTO, by the use or not of Limited Transmit, and the like. Thus, for simulations in this regime, there are many parameters to consider. It is good to insure that simulations exploring fairness include the exploration of fairness with the most aggressive TCP mechanisms conformance with the current standards, that is, SACK TCP using timestamps, ECN, Limited Transmit, and a minimum RTO of 100-200 ms.

9. Simulations of Faster Restart

TBA

10. Implementation Issues

TBA

11. Security Considerations

TBA

12. IANA Considerations

There are no IANA considerations in this document.

13. Thanks

We thank Tom Phelan for discussions of the VoIP variant of TFRC and for his paper exploring the fairness between TCP and VoIP TFRC flows. We also 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.

[WBL04] J. Widmer, C. Boutremans, and Jean-Yves Le Boudec. Congestion Control for Flows with Variable Packet Size, Technical Report.

Authors’ Addresses

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

Eddie Kohler <kohler@cs.ucla.edu>
4531C Boelter Hall
UCLA Computer Science Department

Los Angeles, CA 90095
USA

14. Related Work on VoIP Variants of TFRC

Other proposals for variants of TFRC for application with variable packet sizes include [WBL04]. [WBL04] argues that adapting TFRC for variable packet sizes by just using the packet size of a reference TCP flow in the TFRC equation would not suffice in the high-packet-loss regime; such a modified TFRC would have a strong bias in favor of smaller packets, because multiple lost packets in a single round-trip time would be aggregated into a single packet loss. (We note that the VoIP Variant proposed in our document differs from the straw proposal discussed in [WBL04] in that in our VoIP Variant, the allowed bandwidth includes the bandwidth used by packet headers; and a minimum interval of 10 ms between packets is enforced.) [WBL04] proposes modifying the loss measurement process to account for the bias in favor of smaller packets.

[WBL04] produces both a byte-mode and a packet-mode variant of the TFRC transport protocol, for connections over paths with per-byte and per-packet dropping respectively. We would argue that in the absence of transport-level mechanisms for determining whether the packet dropping in the network is per-packet, per-byte, or somewhere in between, a single TFRC implementation is needed that gives roughly acceptable behavior, to the connection and to the network as a whole, on paths with both per-byte and per-packet dropping (and on paths with multiple congested routers, some with per-byte dropping mechanisms, some with per-packet dropping mechanisms, and some with dropping mechanisms that lie somewhere between per-byte and per-packet).

Full Copyright Statement

Intellectual Property