Strategies for Streaming Media Using TFRC
Internet Draft
Document: draft-ietf-dccp-tfrc-media-01.txt
Expires: May 2006

T. Phelan
Sonus Networks
October 2005

Strategies for Streaming Media Applications Using TCP-Friendly Rate Control

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on May 24, 2006.

Copyright Notice

Abstract

This document discusses strategies for using streaming media applications with unreliable congestion-controlled transport protocols such as the Datagram Congestion Control Protocol (DCCP) or the RTP Profile for TCP Friendly Rate Control. Of particular interest is how media streams, which have their own transmit rate requirements, can be adapted to the varying and sometimes conflicting transmit rate requirements of congestion control protocols such as TCP-Friendly Rate Control (TFRC).

Table of Contents

1. Introduction

The canonical streaming media application emits fixed-sized (often small) packets at a regular interval. It relies on the network to deliver the packets to the receiver in roughly the same regular interval. Often, the transmitter operates in a fire-and-forget mode, receiving no indications of packet delivery and never changing its mode of operation. This often holds true even if the packets are encapsulated in the Real-time Transport Protocol (RTP) and the RTP Control Protocol (RTCP) [RFC 3550] is used to get receiver information; it's rare that the RTCP reports trigger changes in the transmitted stream.

The IAB has expressed concerns over the stability of the Internet if these applications become too popular with regard to TCP-based applications [RFC 3714]. They suggest that media applications should monitor their packet loss rate, and abort if they exceed certain thresholds. Unfortunately, up until this threshold is reached, the network, the media applications, and the other applications are all experiencing considerable duress.

TCP-Friendly Rate Control (TFRC, [RFC 3448]) offers an alternative to the [RFC 3714] method. The key differentiator of TFRC, relative to the Additive Increase Multiplicative Decrease (AIMD) method used in TCP and SCTP, is its smooth response to packet loss. TFRC has been implemented as one of the "pluggable" congestion control algorithms for the Datagram Congestion Control Protocol (DCCP, [DCCP] and [CCID3]) and as a profile for RTP [RTP-TFRC].

This document explores issues to consider and strategies to employ when adapting or creating streaming media applications to use transport protocols using TFRC for congestion control. The approach here is one of successive refinement. Strategies are described and their strengths and weaknesses are explored. New strategies are then presented that improve on the previous ones and the process iterates. The intent is to illuminate the issues, rather than to jump to solutions, in order to provide guidance to application designers.

2. TFRC Basics

AIMD congestion control algorithms, such DCCP's CCID2 [CCID2] or TCP's SACK-based control [RFC 3517], use a congestion window (the maximum number of packets or segments in flight) to limit the transmitter. The congestion window is increased by one for each acknowledged packet, or for each window of acknowledged packets, depending on the phase of operation. If any packet is dropped (or ECN-marked [ECN]; for simplicity in the rest of the document assume

that "dropped" equals "dropped or ECN-marked"), the congestion window is halved. This produces a characteristic saw-tooth wave variation in throughput, where the throughput increases linearly up to the network capacity and then drops abruptly (roughly shown in Figure 1).

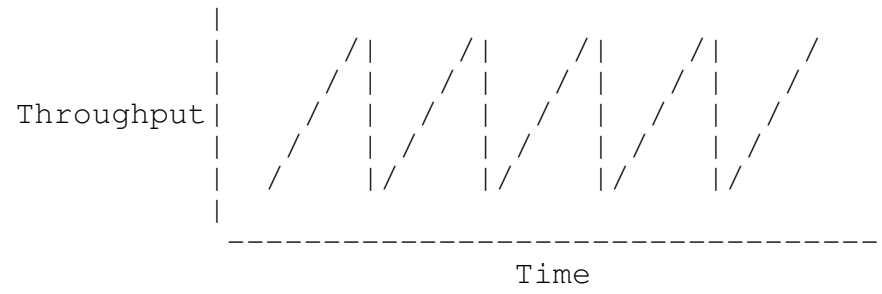

Figure 1: Characteristic throughput for AIMD congestion control.

On the other hand, with TCP-Friendly Rate Control (TFRC), the immediate response to packet drops is less dramatic. To compensate for this TFRC is less aggressive in probing for new capacity after a loss. TFRC uses a version of the TCP throughput equation to compute a maximum transmit rate, taking a weighted history of loss events as input (more weight is given to more recent losses). The characteristic throughput graph for a TFRC connection looks like a flattened sine wave (extremely roughly shown in Figure 2).

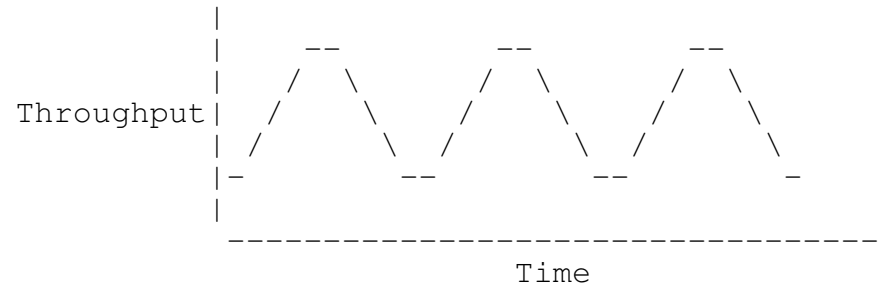

Figure 2: Characteristic throughput for TFRC congestion control.

In addition to this high-level behavior, there are several details of TFRC operation that, at first blush at least, seem at odds with common media stream transmission practices. Some particular considerations are:

- Slow Start -- A connection starts out with a transmission rate of up to four packets per round trip time (RTT). After the first RTT, the rate is doubled each RTT until a packet is lost. At this point the transmission rate is halved and we enter the equation-based phase of operation. It's likely that in many situations the initial transmit rate is slower than the lowest

bit rate encoding of the media. This will require the application to deal with a ramp up period.

- Capacity Probing and Lost Packets -- If the application transmits for some time at the maximum rate that TFRC will allow without packet loss, TFRC will continuously raise the allowed rate until a packet is lost. This means that, in many circumstances, if an application wants to transmit at the maximum possible rate, packet loss will not be an exceptional event, but will happen routinely in the course of probing for more capacity.

- Idleness Penalty -- TFRC follows a "use it or lose it" policy. If the transmitter goes idle for a few RTTs, as it would if, for instance, silence suppression were being used, the transmit rate returns to two packets per RTT, and then doubles every RTT until the previous rate is achieved. This can make restarting after a silence suppression interval problematic.

- Contentment Penalty -- TFRC likes to satisfy greed. If you are transmitting at the maximum allowed rate, TFRC will try to raise that rate. However, if your application is transmitting below the maximum allowed rate, the maximum allowed rate will not be increased higher than twice the current transmit rate, no matter how long it has been since the last increase. This can create problems when attempting to shift to a higher rate encoding, or with video codecs that vary the transmission rate with the amount of movement in the image.

- Packet Rate, not Bit Rate -- TFRC controls the rate that packets may enter the network, not bytes. To respond to a lowered transmit rate you must reduce the packet transmission rate. Making the packets smaller while still keeping the same packet rate will not be effective.

- Smooth Variance of Transmit Rate -- The strength and purpose of TFRC (over AIMD Congestion Control) is that it smoothly decreases the transmission rate in response to recent packet loss events, and smoothly increases the rate in the absence of loss events. This smoothness is somewhat at odds with most media stream encodings, where the transition from one rate to another is often a step function.

3. Streaming Media Applications

While all streaming media applications have some characteristics in common (e.g. data must arrive at the receiver at some minimum rate for reasonable operation), other characteristics (e.g. tolerance of end-to-end delay) vary considerably from application to application. For the purposes of this document, it's useful to divide streaming media applications into three subtypes:

- One-way pre-recorded media
- One-way live media
- Two-way interactive media

The relevant difference, as far as this discussion goes, between recorded and live media is that recorded media can be transmitted as fast as the network allows (assuming adequate buffering at the receiver) -- it could be viewed as a special file transfer operation. Live media can't be transmitted faster than the rate at which it's encoded.

The difference between one-way and two-way media is the sensitivity to delay. For one-way applications, delays from encoding at the sender to playout at the receiver of several or even tens of seconds are acceptable. For two-way applications delays from encoding to playout of as little as 150 to 200 ms are often problematic [XTIME].

While delay sensitivity is most problematic when dealing with two-way conversational applications such as telephony, it is also apparent in nominally one-way applications when certain user interactions are allowed, such as program switching ("channel surfing") or fast forward/skip. Arguably, these user interactions have turned the one-way application into a two-way application -- there just isn't the same sort of data flowing in both directions.

3.1 Stream Switching

The discussion here assumes that media transmitters are able to provide their data in a number of encodings with various bit rate requirements and are able to dynamically change between these encodings with low overhead. It also assumes that switching back and forth between coding rates does not cause excessive user annoyance.

Given the current state of codec art, these are big assumptions. As a practical matter, continuous shifts between higher and lower quality levels can greatly annoy users, much more so than one shift to a lower quality level and then staying there. The algorithms given below indicate methods for returning to higher bandwidth encodings, but, because of the bad user perception of shifting quality, many media applications may choose to never invoke these methods.

Also, the algorithms and results described here hold even if the media sources can only supply media at one rate. Obviously the statements about switching encoding rates don't apply, and an application with only one encoding rate behaves as if it is simultaneously at its minimum and maximum rate.

For convenience in the discussion below, assume that all media streams have two encodings, a high bit rate and a low bit rate, unless otherwise indicated.

3.2 Media Buffers

The strategies below make use of the concept of a media buffer. A media buffer is a first-in-first-out queue of media data. The buffer is filled by some source of data (the encoder or the network) and drained by some sink (the network or the playout device). It provides rate and jitter compensation between the source and the sink.

Media buffer contents are measured in seconds of media play time, not bytes or packets. Media buffers are completely application-level constructs and are separate from transport-layer transmit and receive queues.

3.3 Variable Rate Media Streams

The canonical media codec encodes its media as a constant rate bit stream. As the technology has progressed from its time-division multiplexing roots, this constant rate stream has become not so constant. Voice codecs often employ silence suppression (also called Voice Activity Detection, or VAD), where the stream (in at least one direction) goes totally idle for sometimes several seconds while one side listens to what the other side has to say. When the one side wants to start talking again, the codec resumes sending immediately at its "constant" rate.

Video codecs similarly employ what could be called "stillness" suppression, but is instead called motion compensation. Often these codecs effectively transmit the changes from one video frame to another. When there is little change from frame to frame (as when the background is constant and a talking head is just moving its lips) the amount of information to send is small. When there is a major motion, or change of scene, much more information must be sent. For some codecs, the variation from the minimum rate to the maximum rate can be a factor of ten [MPEG4]. Unlike voice codecs, though, video codecs typically never go completely idle.

These abrupt jumps in transmission rate are problematic for any congestion control algorithm. A basic tenet of all existing algorithms assumes that increases in transmission rate must be gradual and smooth to avoid damaging other connections in the network. In TFRC, the transmission rate in a Round Trip Time (RTT) can never be more than twice the rate actually delivered to the receiver in the previous RTT.

TFRC uses a maximum rate of two packets per RTT after an idle period. This rate might support immediate restart of voice data after a silence period, at least when the RTT is in the suitable range for two-way media. More problematic are the factor of ten variations in some video codecs. In some circumstances, TFRC allows an application

to double its transmit rate over one RTT (assuming no recent packet loss events), but an immediate ten times increase is not possible.

4. Strategies for Streaming Media Applications

This section covers a number of strategies that can be used by streaming media applications. Each strategy is applicable to one or more types of streaming media.

4.1 First Strategy -- One-way Pre-recorded Media

The first strategy is suitable for use with pre-recorded media, and takes advantage of the fact that the data for pre-recorded media can be transferred to the receiver as fast as the network will allow it, assuming that the receiver has sufficient buffer space.

4.1.1 Strategy 1

Assume a recorded program resides on a media server, and the server and its clients are capable of stream switching between two encoding rates, as described in section 3.1.

The client (receiver) implements a media buffer as a playout buffer. This buffer is potentially big enough to hold the entire recording. The playout buffer has three thresholds: a low threshold, a playback start threshold, and a high threshold, in order of increasing size. These values will typically be in the several to tens of seconds range. The buffer is filled by data arriving from the network and drained at the decoding rate necessary to display the data to the user. Figure 3 shows this schematically.

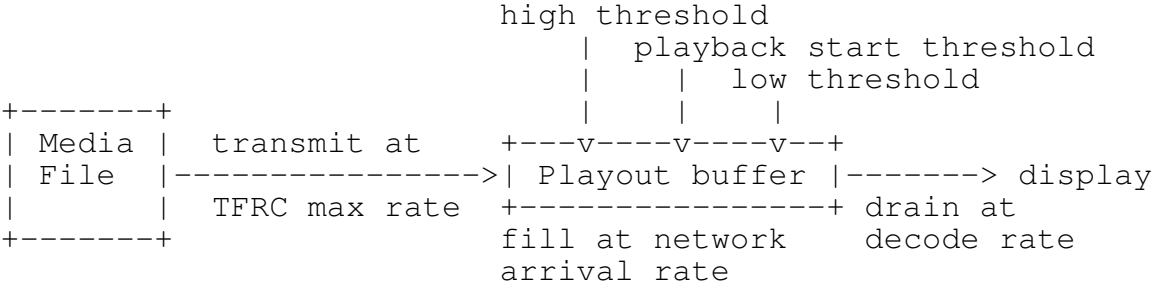

Figure 3: One-way pre-recorded media.

During the connection the server needs to be able to determine the depth of data in the playout buffer. This could be provided by direct feedback from the client to the server, or the server could estimate its depth (e.g. the server knows how much data has been sent, and how much time has passed).

To start the connection, the server begins transmitting data in the high bit rate encoding as fast as TFRC allows. Since TFRC is in slow start, this is probably too slow initially, but eventually the rate should increase to fast enough (assuming sufficient capacity in the network path). As the client receives data from the network it adds it to the playout buffer. Once the buffer depth reaches the playback start threshold, the receiver begins draining the buffer and playing the contents to the user.

If the network has sufficient capacity, TFRC will eventually raise the transmit rate to greater than necessary to keep up with or exceed the decoding rate, the playout buffer will back up as necessary, and the entire program will eventually be transferred.

If the TFRC transmit rate never gets fast enough, or loss events make TFRC drop the rate, the receiver will drain the playout buffer faster than it is filled. When the playout buffer drops below the low threshold the server switches to the low bit rate encoding. Assuming that the network has a bit more capacity than the low bit rate requires, the playout buffer will begin filling again.

When the buffer crosses the high threshold the server may switch back to the high encoding rate. Assuming that the network still doesn't have enough capacity for the high bit rate, the playout buffer will start draining again. When it reaches the low threshold the server switches again to the low bit rate encoding. The server will oscillate back and forth like this until the connection is concluded.

If the network has insufficient capacity to support the low bit rate encoding, the playout buffer will eventually drain completely, and playback will need to be paused until the buffer refills to the playback start level.

Note that, in this scheme, the server doesn't need to explicitly know the rate that TFRC has determined; it simply always sends as fast as TFRC allows (perhaps alternately reading a chunk of data from disk and then blocking on the socket write call until it's transmitted). TFRC shapes the stream to the network's requirements, and the playout buffer feedback allows the server to shape the stream to the application's requirements.

4.1.2 Issues With Strategy 1

The advantage of this strategy is that it provides insurance against an unpredictable future. Since there's no guarantee that a currently supported transmit rate will continue to be supported, the strategy takes what the network is willing to give when it's willing to give it. The data is transferred from the server to the client perhaps faster than is strictly necessary, but once it's there no network problems (or new sources of traffic) can affect the display.

Silence suppression can be used with this strategy, since the transmitter doesn't actually go idle during the silence -- it just gets further ahead. Variable rate video codecs can also function well. Again, the transmitter will get ahead faster during the interpolated frames and fall back during the index frames, but a playout buffer of a few seconds is probably sufficient to mask these variations.

One obvious disadvantage, if the client is a "thin" device, is the large buffer at the client. A subtler disadvantage involves the way TFRC probes the network to determine its capacity. Basically, TFRC does not have an a priori idea of what the network capacity is; it simply gradually increases the transmit rate until packets are lost, then backs down. After a period of time with no losses, the rate is gradually increased again until more packets are lost. Over the long term, the transmit rate will oscillate up and down, with packet loss events occurring at the rate peaks.

This means that packet loss will likely be routine with this strategy. For any given transfer, the number of lost packets is likely to be small, but non-zero. Whether this causes noticeable quality problems depends on the characteristics of the particular codec in use.

4.2 Second Try -- One-way Live Media

With one-way live media you can only transmit the data as fast as it's created, but end-to-end delays of several or tens of seconds are usually acceptable.

4.2.1 Strategy 2

Assume that we have a playout media buffer at the receiver and a transmit media buffer at the sender. The transmit buffer is filled at the encoding rate and drained at the TFRC transmit rate. The playout buffer is filled at the network arrival rate and drained at the decoding rate. The playout buffer has a playback start threshold and the transmit buffer has a switch encoding threshold and a discard data threshold. These thresholds are on the order of several to tens of seconds. Switch encoding is less than discard data, which is less than playback start. Figure 4 shows this schematically.

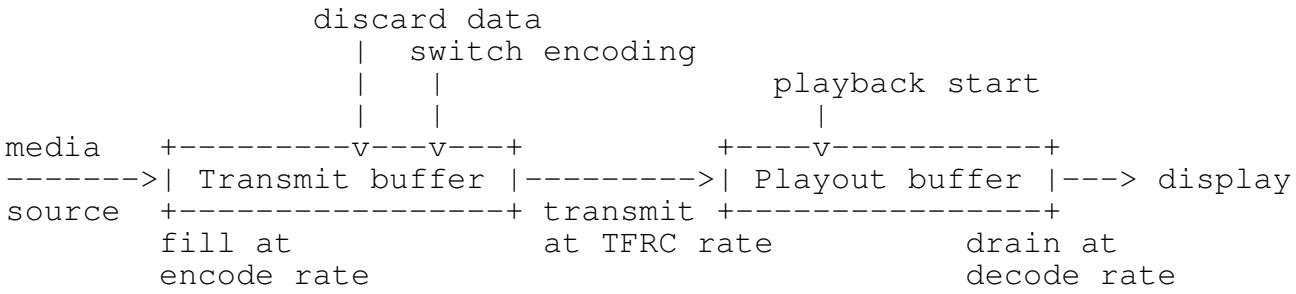

Figure 4: One-way live media.

At the start of the connection, the sender places data into the transmit buffer at the high encoding rate. The buffer is drained at the TFRC transmit rate, which at this point is in slow-start and is probably slower than the encoding rate. This will cause a backup in the transmit buffer. Eventually TFRC will slow-start to a rate slightly above the rate necessary to sustain the encoding rate (assuming the network has sufficient capacity). When this happens the transmit buffer will drain and we'll reach a steady state condition where the transmit buffer is normally empty and we're transmitting at a rate that is probably below the maximum allowed by TFRC.

Meanwhile at the receiver, the playout buffer is filling, and when it reaches the playback start threshold playback will start. After TFRC slow-start is complete and the transmit buffer is drained, this buffer will reach a steady state where packets are arriving from the network at the encoding rate (ignoring jitter) and being drained at the (equal) decoding rate. The depth of the buffer will be the playback start threshold plus the maximum depth of the transmit buffer during slow start.

Now assume that network congestion (packet losses) forces TFRC to drop its rate to below that needed by the high encoding rate. The transmit buffer will begin to fill and the playout buffer will begin to drain. When the transmit buffer reaches the switch encoding threshold, the sender switches to the low encoding rate, and converts all of the data in the transmit buffer to low rate encoding.

Assuming that the network can support the new, lower, rate (and a little more) the transmit buffer will begin to drain and the playout buffer will begin to fill. Eventually the transmit buffer will empty and the playout buffer will be back to its steady state level.

At this point (or perhaps after a slight delay) the sender can switch back to the higher rate encoding. If the new rate can't be sustained the transmit buffer will fill again, and the playout buffer will drain. When the transmit buffer reaches the switch encoding threshold the sender goes back to the lower encoding rate. This

oscillation continues until the stream ends or the network is able to support the high encoding rate for the long term.

If the network can't support the low encoding rate, the transmit buffer will continue to fill (and the playout buffer will continue to drain). When the transmit buffer reaches the discard data threshold, the sender must discard a chunk of data from the transmit buffer for every chunk of data added. Preferably, the discard should happen from the head of the transmit buffer, as these are the stalest data, but the application could make other choices (e.g. discard the earliest silence in the buffer). This discard behavior continues until the transmit buffer falls below the switch encoding threshold. If the playout buffer ever drains completely, the receiver should fill the output with suitable material (e.g. silence or stillness).

Note that this strategy is also suitable for one-way pre-recorded media, as long as the transmit buffer is only filled at the encoding rate, not at the disk read rate.

4.2.2 Issues with Strategy 2

Strategy 2 is fairly effective. There is a limit on the necessary size of the playout buffer at the client, so clients with limited resources can be supported. When silence suppression is used or motion compensation sends interpolated frames, the transmit rate will actually go down, and then must slowly ramp up to return to the maximum rates, but this smoothing can often be masked by a playout buffer of a few seconds.

Also, since strategy 2 limits the transmission rate to the maximum encoding rate, and therefore doesn't try to get every last bit of possible throughput from the network, routine packet loss can be avoided (assuming that there's enough network capacity for the maximum encoding rate).

4.3 One More Time -- Two-way Interactive Media

Two-way interactive media is characterized by its low tolerance for end-to-end delay, usually requiring less than 200 ms for interactive conversation, including jitter buffering at the receiver. Rate adapting buffers will insert too much delay and the slow start period is likely to be noticeable ("Hello" clipping).

This low delay requirement makes using TFRC with variable-rate codecs (codecs using silence suppression or motion compensation) highly problematic. The extra delays imposed by the smooth rate increases mandated by TFRC are unlikely to be tolerated by the interactive applications.

There are further problems with the usual practice in interactive voice applications of using small packets. In voice applications,

the data rate is low enough that waiting to accumulate enough data to fill a large packet adds unacceptable delay. For example, the G.711 codec generates one byte of data every 125 microseconds. To accumulate enough data for a 1480-byte packet, the encoder would need to delay some data by 185 ms, eating up nearly the entire delay budget just for packetization. These considerations can also apply to very low rate video.

The goal of TFRC is fair sharing of a bottleneck, in packets per second, with a TCP application using 1480-byte packets. Applications using smaller packets will receive a fair share of packets per second, but less than a fair of bytes per second. With the packet sizes typically in use in interactive voice applications (e.g., 80 bytes of user data for G.711 with 10 ms packetization), it can be very difficult to achieve useful byte per second rates when in competition with TCP applications.

Further research is needed to resolve these issues. The strategy below can only be applied to constant rate codecs whose data rate is sufficiently large to fill 1480-byte packets within tolerable delay limits.

4.3.1 Strategy 3

To start, the calling party sends an INVITE (loosely using SIP [RFC 3261] terminology) indicating the IP address and port to use for media at its end. Without informing the called user, the called system responds to the INVITE by connecting to the calling party media port. Both end systems then begin exchanging test data, at the (slowly increasing) rate allowed by TFRC. The purpose of this test data is to see what rate the connection can be ramped up to. If a minimum acceptable rate cannot be achieved within some time period, the call is cleared (conceptually, the calling party hears "fast busy" and the called user is never informed of the incoming call). Note that once the rate has ramped up sufficiently for the highest rate codec there's no need to go further.

If an acceptable rate can be achieved (in both directions), the called user is informed of the incoming call. The test data is continued during this period. Once the called user accepts the call, the test data is replaced by real data at the same rate.

If congestion is encountered during the call, TFRC will reduce its allowed sending rate. When that rate falls below the codec currently in use, the sender switches to a lower rate codec, but should pad its transmission out to the allowed TFRC rate. Note that this padding is only necessary if the application wishes to return to the higher encoding rate when possible. If the TFRC rate continues to fall past the lowest rate codec, the sender must discard packets to conform to that rate.

If the network capacity is sufficient to support one of the lower rate codecs, eventually the congestion will clear and TFRC will slowly increase the allowed transmit rate. The application should increase its transmission padding to keep up with the increasing TFRC rate. The application may switch back to the higher rate codec when the TFRC rate reaches a sufficient value.

An application that did not wish to switch back to the higher encoding (perhaps for reasons outlined in section 3.1) would not need to pad its transmission out to the TFRC maximum rate.

Note that the receiver would normally implement a short playout buffer (with playback start on the order of 100 ms) to smooth out jitter in the packet arrival gaps.

4.3.2 Issues with Strategy 3

An obvious issue with strategy 3 is the post-dial call connection delay imposed by the slow-start ramp up. This is perhaps less of an issue for two-way video applications, where post-dial delays of several seconds are accepted practice. For telephony applications, however, post-dial delays significantly greater than a second are a problem, given that users have been conditioned to that behavior by the public telephone network. On the other hand, the four packets per RTT initial transmit rate allowed by DCCP's CCID3 in some circumstance is likely to be sufficient for many telephony applications, and the ramp up will be very quick.

As was stated in section 4.3, this strategy is only suitable for use with constant-rate codecs with fast enough data rates to tolerate using large packets.

5. Security Considerations

There are no security considerations for this document. Security consideration for TFRC and the protocols implementing TFRC are discussed in their defining documents.

6. IANA Considerations

There are no IANA actions required for this document.

7. Thanks

Thanks to the AVT working group, especially Philippe Gentric and Brian Rosen, for comments on the earlier version of this document.

8. Informative References

[DCCP] E. Kohler, M. Handley, S. Floyd, J. Padhye, Datagram Congestion Control Protocol (DCCP), February 2004, draft-ietf-dccp-spec-06.txt, work in progress.

[CCID2] S. Floyd, E. Kohler, Profile for DCCP Congestion Control 2: TCP-Like Congestion Control, February 2004, draft-ietf-dccp-ccid2-05.txt, work in progress.

[CCID3] S. Floyd, E. Kohler, J. Padhye, Profile for DCCP Congestion Control 3: TFRC Congestion Control, February 2004, draft-ietf-dccp-ccid3-04.txt, work in progress.

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

[RFC 3714] S. Floyd, J, Kempf, IAB Concerns Regarding Congestion for Voice Traffic in the Internet, March 2004, RFC 3714.

[RFC 3261] J. Rosenberg, et al, SIP: Session Initiation Protocol, June 2002, RFC 3261

[RFC 3517] E. Blanton, M. Allman, K. Fall, L. Wang, A Conservative Selective Acknowledgment (SACK)-based Loss Recovery Algorithm for TCP, April 2003, RFC 3517

[RFC 3550] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: A Transport Protocol for Real-Time Applications, July 2003, RFC 3550

[XTIME] ITU-T: Series G: Transmission Systems and Media, Digital Systems and Networks, Recommendation G.114, One-way Transmission Time, May 2000

[ECN] K. Ramakrishnan, S. Floyd, D. Black, The Addition of Explicit Congestion Notification (ECN) to IP, September 2001, RFC 3168

[MPEG4] ISO/IEC International Standard 14496 (MPEG-4), Information technology - Coding of audio-visual objects, January 2000

[RTP-TFRC] L. Gharai, RTP Profile for TCP-Friendly Rate Control, October 2004, draft-ietf-avt-tfrc-profile-03.txt, work in progress

9. Author's Address

Tom Phelan
Sonus Networks
5 Carlisle Rd.
Chelmsford, MA USA 01824
Phone: +1-978-614-8456
Email: tphelan@sonusnet.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.