TCP-Friendly Rate Control with Token Bucket for VoIP Congestion Control

Miguel Maldonado, Salman Abdul Baset, and Henning Schulzrinne
Department of Computer Science
Columbia University
{mam2136, salman, hgs}@cs.columbia.edu
October 17, 2005

Abstract

TCP Friendly Rate Control (TFRC) is a congestion control algorithm that provides a smooth transmission rate for real-time network applications. TFRC refrains from halving the sending rate on every packet drop, instead it is adjusted as a function of the loss rate during a single round trip time. TFRC has been proven to be fair when competing with TCP flows over congested links, but it lacks quality-of-service parameters to improve the performance of real-time traffic. A problem with TFRC is that it uses additive increase to adjust the sending rate during periods with no congestion. This leads to short term congestion that can degrade the quality of voice applications.

We propose two changes to TFRC that improve the performance of VoIP applications. Our implementation, TFRC with Token Bucket (TFRC-TB), uses discrete calculated bit rates based on audio codec bandwidth usage to increase the sending rate. Also, it uses a token bucket to control the sending rate during congestion periods. We have used ns2, the network simulator, to compare our implementation to TFRC in a wide range of network conditions. Our results suggest that TFRC-TB can provide a quality of service (QoS) mechanism to voice applications while competing fairly with other traffic over congested links.

I. INTRODUCTION

Current Internet stability depends on the end-to-end congestion control provided by TCP which is based on an Additive Increase Multiplicative Decrease (AIMD) algorithm. This mechanism allows flows to share bandwidth fairly over congested links. Data transfer application, such as HTTP, FTP and SMTP, greatly benefit from the bandwidth sharing and are not affected by the drastic rate changes. Multimedia applications, by contrast, require a consistent transmission rate, since abrupt rate changes can cause noticeable degradation in user-perceived quality. The widespread use of Internet media applications such as audio players, IP-telephony, video conferencing and other real-time applications can significantly increase the amount of non-TCP traffic over the Internet. Since most of this traffic has no congestion control algorithms, and given the best-effort nature of the current Internet it can lead to TCP traffic starvation or even congestion collapse [1].

Researchers have developed TCP-friendly protocols to prevent such a collapse. Protocols are considered TCP friendly when their long term throughput does not exceed the throughput of a TCP flow under similar network conditions. This is accomplished by modeling the TCP throughput as an equation based on measurable network parameters such a round-trip time (RTT) and packet loss. Senders then use this equation to modify their sending rate. The three main efforts in this area are TCP-Friendly Rate Control (TFRC) [2], Datagram Congestion Control Protocol (DCCP) [3], and the development of rate-adaptive audio codecs [4]. For the rest of our study, we will focus only on TFRC and will use TFRC and TCP-friendly interchangeably.

TFRC is a receiver-based congestion control mechanism designed for unicast flows operating in an Internet environment and competing with TCP traffic [5]. TFRC is designed to be reasonably fair when competing for bandwidth with TCP flows; its sending rate is generally within a factor of two of a TCP flow under similar network conditions. In order to have a relatively smooth sending rate, TFRC flows respond slowly to changes in available bandwidth. The TFRC [2] protocol works as follows:

- The receiver measures the loss event rate and returns it to the sender.
- The sender uses this feedback messages to calculate the RTT
- The sender uses the RTT and the loss event rate to calculate the transmit rate.
- The sender adjusts its sending rate accordingly.

The TFRC throughput equation is based on a simplified version of the throughput equation for Reno TCP [2]:

$$\lambda = \frac{s}{RTT * \left(\sqrt{2 * \frac{p}{3}} + \left(12 * \sqrt{3 * \frac{p}{8}} * P * (1 + 32 * p^2)\right)\right)}$$

where, λ is the transmission rate in bytes/sec, s is the the packet size in bytes, and p is the loss event rate.

TFRC uses the loss event rate to model a protocol that reduces its data window once per congestion notification. The loss event rate is measured in terms of loss intervals, spanning the number of packet between consecutive loss events, which are averaged with decaying weights [6]. Also, TFRC is not a complete transport protocol, therefore, it requires to be deployed together with transport protocols such as UDP, or DCCP [2]. Finally, TFRC is designed for applications that use a fixed packet size and vary their sending rate in response to a congestion notification.

TFRC has been designed to compete fairly with TCP, but little attention has been given to the quality of service (QoS) for real-time applications [7]. A major limitation of TFRC is that it is still inherently an AIMD mechanism. This leads to short term congestion that can degrade the quality of voice applications. In voice traffic, small increments in available bandwidth do not affect the overall quality of the communication. Our implementation, TFRC with Token Bucket (TFRC-TB), focuses on providing a QoS mechanism for VoIP applications. It uses discrete bit rates based on calculated audio codec bandwidth usage to control the sending rate. This allows voice applications to manage the communication quality, since depending on the available bandwidth the sender and receiver can utilize the appropriate voice encoding. The selected encoding rate must be less than the calculated fair rate of the TFRC equation in order to maintain fairness with competing traffic. TFRC-TB uses a token bucket to store the difference between the selected encoding and the sending rate calculated from the congestion equation. These tokens can be used during a congestion period. Token buckets are widely utilized to smooth bursty traffic by forcing a constant transmission rate.

Our work focuses on the believe that the use of discrete sending rates and the token bucket strategy will result in a more stable sending rate, that results in a smoother traffic pattern. Also, we will verify that TFRC-TB still competes fairly with TCP traffic when sharing a common bottleneck link. This idea was partially motivated by the Internet Architecture Board (IAB) discussion of voice traffic congestion control [6], in which it is assured that VoIP applications that use end-to-end congestion control, and that have codecs that can adapt to the bit rate received by the client can be successfully deployed in the current best effort environment. Also, TFRC-TB was influenced by the voice transmission system [7] (Fig. 1), discussed in Section II-A

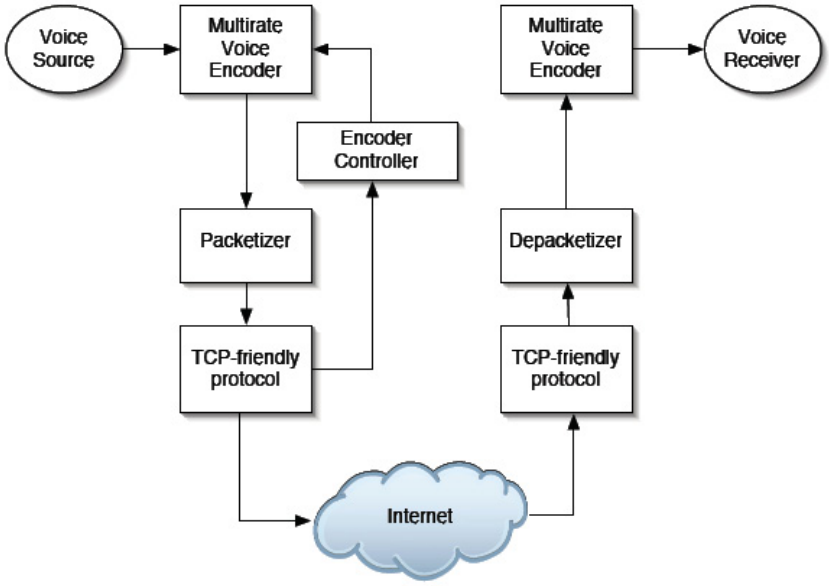

Fig. 1. Voice transmission system [7]

II. Related Work

Below, we summarize related work for improving TFRC.

A. TCP-Friendly Transmission of Voice over IP

Francesco Beritelli, Giuseppe Ruggeri, and Giovanni Schembra addressed the problem of TCP-friendly algorithms for real-time applications which are more concerned with QoS than fairness with TCP [7]. They analyzed the problem of transmitting voice using TCP-friendly protocols and proposed a VoIP architecture for improving the subjective quality of the transmissions. Their voice transmission system (Fig. 1), has three main components, the TCP-friendly algorithm, the voice encoder and the encoder controller.

The voice encoder receives the input from the voice source and uses a multi-rate encoder that adapts to the bandwidth imposed by the TCP-friendly rate control mechanism [7]. The voice encoder sends the voice frame to the packetizer which prepares the packet for the TCP-friendly rate controller. The encoder controller uses the information from the TFRC algorithm to select the appropriate voice encoding. Since for voice applications a high loss rate may not produce acceptable quality for voice communication, the authors proposed a change for TCP-friendly protocols to take into consideration the RTT delay variation. When the delay variation surpasses a decision threshold, the application cuts its rate in half and goes into a slow-start phase. Their experiment focused on a set of voice sources sharing a common bottleneck link. Their results showed an increase

in perceived user quality on sets of sources, which consists of 10 to 40 unique voice flows over a single link of capacity 150 kbit/s, simulated n ns-2 [7].

B. TFRC for Voice

Sally Floyd and Eddie Kohler are currently working in a variant of TFRC for VoIP that provides fairness to applications that send small packets. They argue [6] that it is acceptable for VoIP flows to assume that network limitations are in bytes per second, which measure congestion in terms of the available bandwidth, instead of the more common measurement of packet per second which relates to routing functions such as header processing and packet forwarding. They proposed the following changes to the basic TFRC protocol:

- set the nominal packet size to 1460 bytes,
- reduce the allowed transmit rate to account for the packet header,
- impose a minimal interval between packets of 10 ms.

These changes not only help the flow to share the available bandwidth in bytes per second, but also discourage application from using extremely small packet sizes.

A faster restart is also introduced to improve voice application responsiveness after idle periods. Faster restart allows a idle flow to quadruple its sending rate in every congestion free RTT up to their previously achieved transmission rate [8]. For example, we will consider a G.711 VoIP call, which has a sending rate of 12 kB/s, and a estimated RTT of 0.15 seconds. The sending rate account for the audio packet size and the headers of the IP, UDP and RTP protocols, which are commonly used for the audio packets [7]. During the silent periods the source sending rate will be around 1 kB/s, since the application still sends a packet every couple of RTT to maintain network state information. Using the faster restart and, assuming no congestion is detected during the restart phase, the flow requires only 2 RTTs to achieve the previous stable sending rate of 12 kB/s, current implementations of TCP and TFRC would require 4 RTTs.

III. EXPERIMENTAL EVALUATION

We tested TFRC-TB rate control extensively using the ns-2.28 [9] network simulator. We used the dumbbell topology for our experiments (Fig. 2). This topology allowed us to evaluate the effect of different congestion mechanisms over a shared

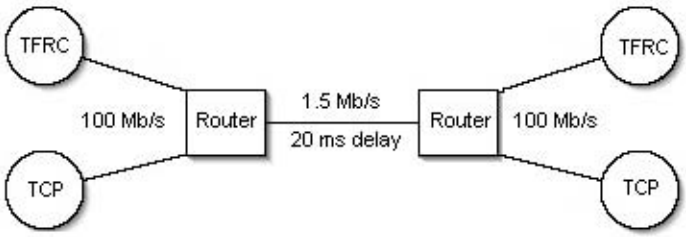

Fig. 2. Dumbbell Topology for experiment simulation.

bottleneck link [10]. We focused on measuring the fairness between competing flows and the variance of the transmission rate for individual flows. We tested our implementation under a wide variety of traffic mixes that not only highlight the properties of the mechanism, but also attempt to simulate realistic network environments. All of our simulations use a common bottleneck link of 1.5 Mb/s and 20 ms delay using Random Early Detection (RED) queue management. To simulate realistic voice traffic, our TFRC-TB rate corresponds to the rate generated by the G.711 (64 kb/s), G.726 (32 kb/s), G.728 (16 kb/s), G.729 (10 kb/s) and G.723 (6.4 kb/s) codecs. The transmission rate for each codec is calculated by adding size of the header of the IP, UDP and RTP protocols to the audio packets.

IV. SIMULATION RESULTS

In order to prove that our TFRC-TB implementation can be safely deployed in Internet, we must demonstrate that it competes fairly with other network traffic when sharing a common bottleneck link. Also, we need to analyze the rate variance of TFRC-TB in comparison to TFRC during realistic network traffic scenarios. Finally, we investigate the effect of idle periods on the transmission rates for voice traffic.

A. Fairness

Our initial test focuses on the fairness of TFRC-TB when competing with TCP traffic over a shared link. We conducted tests over a wide range of network conditions and will only present a summary of our work. These tests consist of n TCP and n TFRC flows sharing a common bottleneck link. We calculate the mean throughput for each flow, this value is normalized in relation to the fair share per flow of the link bandwidth. The results (Fig. 3) , show that our TFRC-TB implementation is

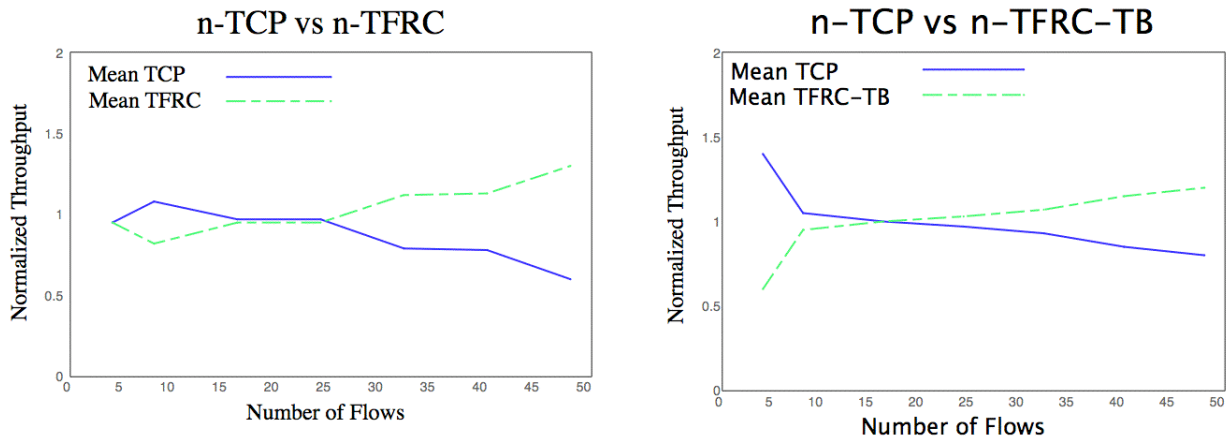

Fig. 3. Normalized mean throughput of TCP flows competing with TFRC or TFRC-TB flows over shared bottleneck link.

comparable in fairness to TFRC. Furthermore, it highlights that, when a small number of TFRC-TB flows are present on the congested link, TCP gets more than its fair share of the throughput. This is because TFRC-TB uses the discrete bit rate of the available codecs to control transmission rate. Our implementation does not transmits faster than the sending rate of its best quality codec. This is an acceptable limitation for voice application since we have no necessity to transmit faster than this. The results also show a limitation of TCP-friendly protocols, since they are slow to respond to changes in link bandwidth on periods of high congestion when it uses more that its fair share of the available bandwidth. This is a characteristic of the TCP-friendly protocol, since the behavior for TFRC-TB when the sending rate is less than the lowest encoding is solely dictated by TFRC. Upon closer inspection of the traffic shape (Fig. 4) , it shows that our TFRC-TB provides a more stable

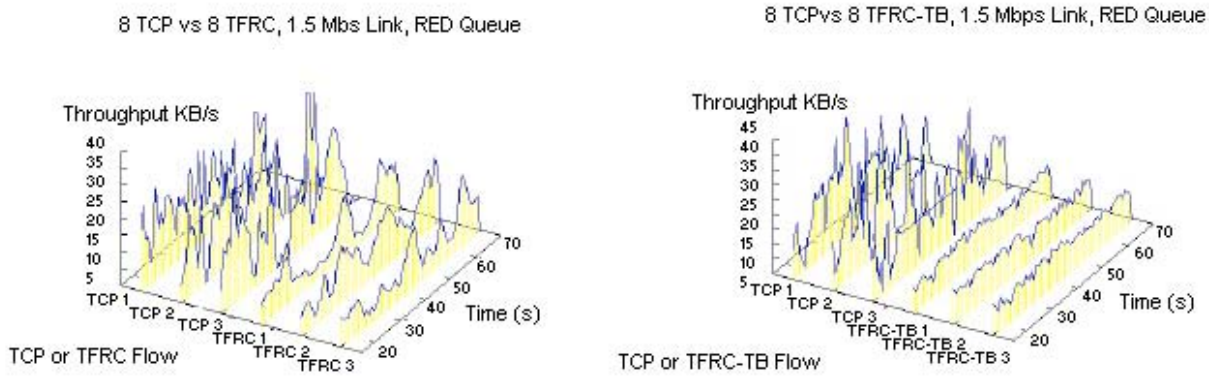

Fig. 4. Sample set of TCP flows competing with TFRC or TFRC-TB throughput from Figure 3 as a function of time.

sending rate when compared to TFRC, but also the TCP flows receive more bandwidth with TFRC-TB than when competing with TFRC.

B. Coefficient of Variation

For the next simulation scenario, we want to test the variance in the sending rate for a single TFRC-TB flow and a single TFRC flow under similar network conditions. In order to quantify the variance of a flow we will use the coefficient of variation [5], which is the standard deviation divided by the mean. It can be defined using the following function for the sending rate R over an interval of time as follows:

$$R_{\delta,F(t)} = \frac{\text{packet size} * \text{number of packets send between } t \text{ and } t+\delta}{\delta}$$

In this simulation scenario we are looking to model the effect of competing web traffic on the smoothness [5] of our TFRCTB implementation. We define smooth traffic as one that is able to hold a consistent transmission rate for a determine period of

time. Our simulation consist of a long-lived TCP flow, a long-lived TFRC-TB or TFRC flow and a series of UDP on and off heavy-tailed distribution flows. The UDP flows have a mean on period of one second with a sending rate of 200 Kb/s and a mean off period of two seconds. The simulation ran for 2500 seconds and averages five runs, each set of runs used between 50 and 100 on and off connections.

Our test results (Fig. 5) showed that TFRC and TFRC-TB have a similar coefficient of variation. The coefficient of variation

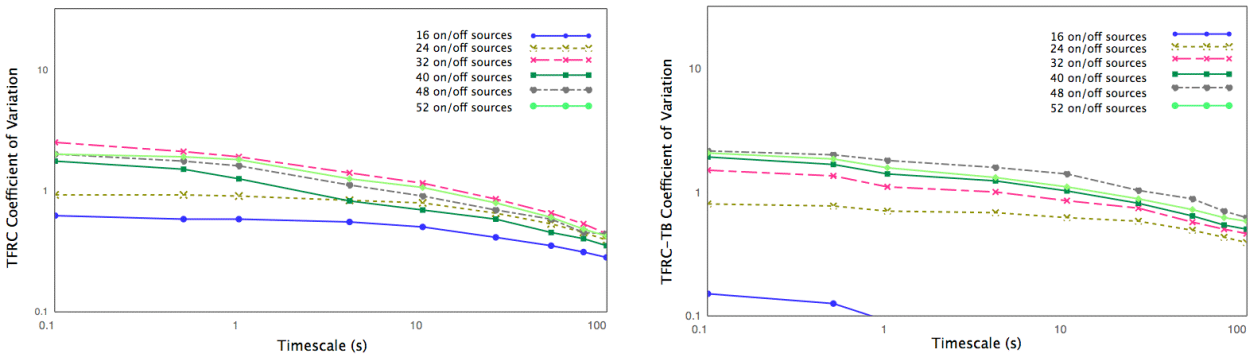

Fig. 5. Coefficient of Variation of TRFC and TFRC-TB when competing with web traffic.

of the TFRC-TB simulations for 16 on and off source is very close to zero because of the maximum sending rate limitation discussed in Section IV-A. The other difference that can be observed is that for TFRC-TB the coefficient of variation remains more stable across the various time-scales. We believe this is caused by a smaller quantity of larger magnitude rate changes from the discrete audio rates implemented in TFRC-TB. We had expected TFRC-TB to have a smaller coefficient of variation that TFRC, but we are satisfied with the results since a typical TCP connection under similar scenarios can have a coefficient of variation four times this value [5].

C. Impact of Voice Activity Detection

The final aspect of our implementation studied is the responsiveness of TCP-friendly rate control to VoIP silence periods. Our investigation so far has focused on the fairness and variance of TCP-friendly rate control using long lived flows with constant senders. This model does not represent the typical voice communication where senders have similar mean ON and OFF periods. This a major concern for voice application, since it not acceptable for users to open and close a connection before resuming the voice call. Also, current networking practices do not allow a idle sender to maintain an active sending rate. S. Floyd and E. Kohler [8] discussed that it is reasonable for an application that contributes to transient congestion, by remaining idle during silence periods, to improve it responsiveness after silence periods by a faster rate ramp up. They propose a faster restart mechanism that allows a idle VoIP connection to quadruple their sending rate every congestion free RTT, until the last achieved rate.

For the next set of simulations we use a long lived TFRC or TFRC-TB flow with 10 second talk spurts, and 10 second silent periods. Since we wanted to focus on the impact of voice activity detection and the connection restart mechanism, not on the actual quality of the voice communication these value are sufficient for our experiment. We also use 25 on and off sources, as defined for the simulations in Section IV-B, for background traffic. We compared both TFRC and TFRC-TB using a TCP like slow-start mechanism (Fig 6), and using the fast restart algorithm (Fig 7). The fast restart allows the flows to

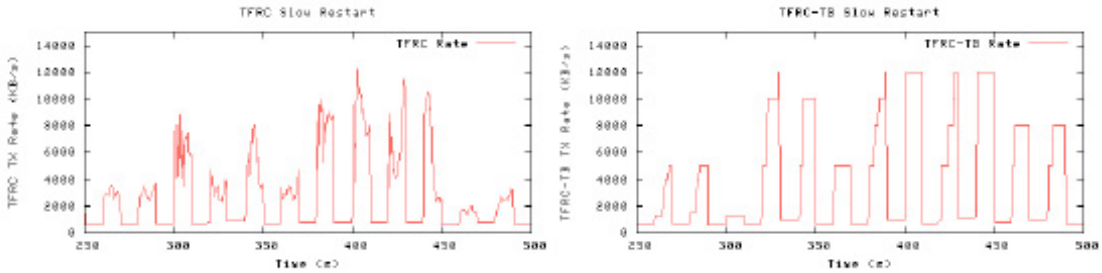

Fig. 6. Voice over TFRC or TFRC-TB with no fast restart.

resume their previous sending rate in a very consistent manner, but on some cases it causes congestion in the middle of the

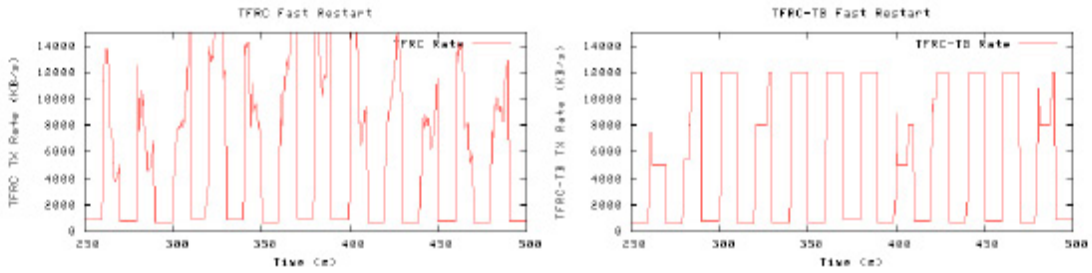

Fig. 7. Voice over TFRC or TFRC-TB with fast restart.

talk spurt. This is especially highlighted on the TFRC-TB graph of (Fig 7), where the transmission rate is reduced during voice activity periods. On the other hand, the slow-start mechanism has more variance in the transmission rate after the idle periods, but it seldomly significantly reduce the transmission rate during the length of the talkspurt. These experiments do not provide conclusive evidence that any of the mechanism is better than the other.

V. CONCLUSION

In this paper we have proposed a QoS mechanism for voice transmission using TCP-friendly rate control. Our algorithm, TFRC-TB, uses discrete sending bit rates and a token bucket to provide QoS parameter to voice traffic. We extensively tested our implementation under the ns2 simulator. TFRC-TB is able to maintain fairness when competing with TCP traffic under a wide range of network conditions. It has demonstrated that it is able to successfully use the discrete sending rates to maintain a more constant transmission rate. This proves that it is possible to develop QoS aware algorithms that can be implemented at the end host without requiring further changes in the current network architecture and that is safe to be deployed in best effort networks.

VI. FUTURE WORK

Currently TFRC-TB has only been simulated under ns2. Thus an actual implementation will be a natural step in the development of TFRC-TB voice applications. Another interesting aspect that we have yet to test is user perception of the proposed QoS mechanism. Currently, it is assumed that higher bit rates correspond to better quality as was the case in the older codecs. This is no longer the case with the wide range of audio codecs that are currently available. In order to improve TFRC-TB responsiveness to calculated available bandwidth, the TFRC protocol headers can be extended to help the sender and the receiver agree on the codec. The transmission rate after idle periods for voice applications is another area for future research.

REFERENCES

[1] J. Widmer, R. Denda, and M. Mauve, “A survey on tcp-friendly congestion control,” *IEEE Network Magazine*, vol. 15, no. 3, pp. 28–37, May 2001.
[2] M. Handley, S. Floyd, , J. Padhye, and J. Widmer, “Tcp friendly rate congestion control (tfrc): protocol specification,” RFC 3448, June 2002.
[3] S. Floyd, E. Kohler, and M. Handley. (2005, Mar.) Datagram congestion control protocol (dccp). Internet draft. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-dccp-spec-11.txt
[4] J. Sjoberg, M. Westerlund, A. Lakaniemi, and Q. Xie, “Real-time transport protocol (rtp) payload format and file storage format for the adaptive multi-rate (amr) and adaptive multi-rate wideband (amr-wb) audio codecs,” RFC 3267, Jan. 2003.
[5] M. Handley, S. Floyd, , J. Padhye, and J. Widmer, “Equation-based congestion control for unicast applications: the extended version,” in *Special Interest Group on Data Communications (SIGCOMM ’00)*, Stockholm, Sweden, Aug. 2000.
[6] S. Floyd and J. Kempf, “Iab concerns regarding congestion control for voice traffic in the internet,” RFC 3714, Mar. 2004.
[7] F. Beritelli, G. Ruggeri, and G. Schembra, “Tcp-friendly transmission of voice over ip,” in *International Conference on Communication (ICC)*, New York, NY, Apr. 2002.
[8] S. Floyd and E. Kohler. (2005, July) Tcp friendly rate control (tfrc) for voice: Voip variant. Internet draft. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-dccp-tfrc-voip-02.txt
[9] The network simulator ns-2. [Online]. Available: http://www.isi.edu/nsman/ns
[10] A. Bitorika, M. Robin, and M. Huggard, “A framework for evaluationg active queue management schemes,” Department of Computer Science, Trinity College, Dublin, Ireland, Tech. Rep., July 2003.