An Experimental Evaluation of Voice Quality over the Datagram Congestion Control Protocol

Horia Vlad Balan[*]
International University Bremen
h.balan@iu-bremen.de

Lars Eggert[*]
Nokia Research Center
lars.eggert@nokia.com

Saverio Niccolini
NEC Network Laboratories
niccolini@netlab.nec.de

Marcus Brunner
NEC Network Laboratories
brunner@netlab.nec.de

Abstract— **Most Internet telephony applications currently use either TCP or UDP to carry their voice-over-IP (VoIP) traffic. This choice can be problematic, because TCP is not well suited for interactive traffic and UDP is unresponsive to congestion. The IETF has recently standardized the new Datagram Congestion Control Protocol (DCCP). DCCP has been designed to carry media traffic and is congestion-controlled. This paper experimentally evaluates the voice quality that Internet telephony calls achieve over prototype implementations of basic DCCP and several DCCP variants, under different network conditions and with different codecs. It finds that the currently-specified DCCP variants perform less well than expected when compared to UDP and TCP. Based on an analysis of these results, the paper suggests several improvements to DCCP and experimentally validates that a prototype implementation of these modifications can significantly increase voice quality.**

I. Introduction

Voice-over-IP (VoIP) Internet telephony has become very popular. Most Internet telephony applications, including SIP-based services [1] or peer-to-peer applications such as Skype [2], currently use either TCP or UDP. The use of either of these two protocols for voice traffic can be problematic. TCP transmits data reliably, which is unnecessary for interactive data that loses relevance for the receiver the longer it is delayed. TCP retransmissions of already-stale data can hence delay transmission of more current audio, increasing delays and reducing voice call quality. UDP does not transmit reliably and hence does not suffer from this problem. UDP, however, is not responsive to network congestion and can hence negatively impact competing traffic when congestion occurs.

The Datagram Congestion Control Protocol (DCCP) [3] provides a framework for unreliable but congestion-controlled data transmission [4]. DCCP's Congestion Control ID 3 (CCID3) profile [5], based on TCP-Friendly Rate Control (TFRC) [6], is specifically designed for the transmission of interactive data. Extensions to TFRC to improve operation when applications generate small packets [7] and to enable faster slow-start-restart after idle periods [8] both attempt to further improve transmission of interactive traffic. Section II gives an overview of DCCP and its variants.

This paper experimentally analyzes the achievable voice quality when a prototype implementation of DCCP that has been extended to implement these DCCP variants carries interactive voice traffic. It also compares the voice quality achieved over DCCP against that over UDP and TCP in a number of network scenarios with different round-trip times and packet loss rates. Additional results are available [9]. Section III describes the details of the prototype implementation and the experimental setup, parameters and metrics. Section IV presents the detailed results of the experimental analysis.

The main finding of these experiments is that DCCP's congestion control mechanisms can significantly reduce voice quality, compared to both UDP and TCP. It is expected that voice quality over the congestion-controlled DCCP decreases to some degree, compared to congestion-unresponsive UDP. An unexpected finding is that some variants of DCCP can also result in voice quality that is poorer than that achieved with TCP. Section V discusses these findings and describes their causes. DCCP variants are still under active design and one goal of this research is to aid the ongoing efforts of the IETF working group.

Based on this discussion, the paper describes several improvements to DCCP's currently-specified mechanisms. One contribution of this paper is an implementation of these improvements in the KAME DCCP prototype [10]. Based on this implementation, Section VI experimentally compares the modified DCCP against the currently-specified variants, as well as UDP and TCP. It finds that voice quality of the modified DCCP is significantly better and approaches the quality achievable with UDP in some scenarios.

Finally, Section VII concludes this paper with a brief discussion of future work, such as a continued experimental evaluation of the impact the various DCCP variants have on competing traffic and vice versa.

A number of prior efforts have investigated how DCCP compares to other transport protocols [11]–[13]. However, these investigations have used simulations or older DCCP implementations and have focused on high-bitrate video traffic. Neither has investigated interactive, bursty, lower-bitrate voice traffic in an experimental setup with a DCCP prototype implementation that conforms to the current specifications.

II. DCCP Overview

The Datagram Congestion Control Protocol (DCCP) [4] provides a framework for congestion-controlled but unreliable

[*] Horia Vlad Balan and Lars Eggert were with NEC Europe Ltd., Network Laboratories in Heidelberg, Germany during the duration of this research.

data transmission. Within this framework, different Congestion Control Identifiers (CCIDs) implement different TCP-friendly congestion control profiles. CCID2 [14] specifies a profile similar to TCP's Additive Increase/Multiplicative Decrease (AIMD) congestion control mechanisms. CCID3 [5] provides rate-controlled congestion control, based on TCP-Friendly Rate Control (TFRC) [6], that is better suited for the transmission of interactive data.

TFRC [6] is a congestion control mechanism that controls the sending rate based on a stochastic Markov model for TCP-Reno [15], which has a closed-form expression under a specific set of assumptions. The mathematical model behind TFRC has some limitations, such as the assumption of unlimited offered load or modeling all transmissions as maximum-sized segments. DCCP's CCID3 [5] is based on TFRC but incorporates several improvements.

The development and specification of DCCP and TFRC is still ongoing in the IETF. One focus of the current IETF work is on variants of the existing mechanisms that are better suited for bursty interactive traffic. The TFRC *small packet* variant [7] aims to better support low-bitrate and small-packet traffic. It is motivated by the observation that the original TFRC model assumes transmission of MTU-sized segments. Applications that transmit short segments, such as voice audio, achieve a significantly lower throughput under TFRC. The *small packet* variant of TFRC allows senders that emit small segments in such a way that they achieve roughly the same bandwidth as a TFRC sender that uses large segments. The TFRC *faster restart* mechanism [8] is another proposed extension to TFRC aimed at improving the transmission of bursty traffic. It is motivated by the assumption that slow-start-restart after (brief) idle periods may be safely accelerated based on the knowledge of past transmission rates.

The experimental evaluation in Section IV hence focuses on measuring voice quality of VoIP calls over the *small packet* variant of TFRC and a combination of the *small packet* and *faster restart* variants of TFRC. However, it also includes basic TFRC, i.e., DCCP CCID3, as well as UDP and TCP as baselines in the measurements, to compare the optimized TFRC variants against the basic variants as well as other transport protocols.

III. Experimental Setup

During each experimental run, a sender generates a random voice call, encodes it into audio frames and transmits them across a network topology using a transport protocol. The receiver measures the voice quality of the call. This section describes the details of each of these steps, i.e., call generation, encoding, transmission, network setup, playout and voice quality measurement.

A. Call Generation

The first step of an experimental run is the generation of a random voice call at the sender. Note that although voice calls typically transmit audio in both directions, transmission in the two directions is logically independent. This experiment hence only transmits voice in one direction.

TABLE I
Codec parameters and characteristics [20].

Codec	Audio Bandwidth [kbps]	Sample Period [ms]	Frame Size [Bytes]	Frames/ Packet	Data Bandwidth [kbps]
G.711	64	20	160	1	95.2
G.729	8	10	10	2	39.2

The sender generates a random voice call by interleaving talkspurts and pauses with lengths generated by a decaying exponential distribution according to the model given in [16], which has an average duration of 1 second per talkspurt and 1.5 seconds per pause. The voice audio for each talkspurt is extracted out of the beginning of a speech recording.

Each call consists of a precomputed concatenation of 100 talkspurt/pause cycles, leading to an average call length of 250 seconds, which the sender then encodes and transmits.

B. Voice Encoding

Once the sender has generated random audio for a call, it encodes this audio into data frames for transmission across the network. The experiment uses two different configurations of the Speex codec [17] for this purpose. The configuration used for a given call is one of the parameters of the experiment.

The first Speex codec configuration emits a frame stream that is similar to that of the G.711 codec [18], the second one emits a frame stream that is similar to the G.729 codec [19]. Both configurations enable voice activity detection, i.e., stop emitting audio frames during periods of silence. Table I shows the main characteristics of the two Speex configurations.

Note that the voice quality metric used to compare the results does *not* depend on the payload of the audio frames but merely on the packet arrival sequence and timings at the receiver (see Section III-F). Configuring Speex to resemble G.711 and G.729 enables the use of published coefficients (Table III) to determine audio quality impairments, as described in Section III-F.

C. Data Transmission

Encoding the voice call generates a stream of audio frames. The sender transmits these across the network using either UDP, TCP with the Nagle algorithm disabled, TFRC (DCCP CCID3), the TFRC *small packet* variant (TFRC SP), or a combination of the TFRC *faster restart* and *small packet* variants (TFRC SP+FR). The choice of transport protocol is one parameter of the experiment.

An enhanced reimplementation of the original `ttcp` tool performs transport of audio frames using different transport protocols. It reads framed data from the codec and transmits it over one of the various transport protocols to a `ttcp` peer on the receiving host. The tool also produces a time-sequence trace of received audio frames, which is used as the basis for determining voice quality at the receiver.

The sender and receiver run version 5.4 of the FreeBSD operating system with a recent KAME snapshot that includes Yoshifumi Nishida's prototype DCCP implementation [10], using an outbound queue of five packets. Note that this DCCP implementation is not fully conformant to the published specification. For this experiment, we implemented the major features of the *faster restart* and *small packet* variants of DCCP, as far as they are relevant to the investigated scenarios. We also implemented modifications to these variants that further improve voice quality, as described in Section V and evaluated in Section VI.

D. Network Setup

During each experimental run, the `ttcp` tool transmits a single voice call to a corresponding `ttcp` instance on a receiver. These transmissions occur across a network path with varying loss and delay characteristics, emulated by a DummyNet router [21]. All machines used in this experiments have identical hardware. All links in this network use Gigabit Ethernet. The DummyNet router uses a FIFO queue of 50 packets.

The path delay and loss rates at the DummyNet router are two parameters of the experiment. Path delay is increased from 0 to 400 ms. Loss rates vary from 0.01% to 10%.

E. Playout

The `ttcp` tool on the receiver generates a time-sequence trace of incoming audio frames. Compared to the sending sequence, audio frames may be reordered, dropped or arrive with different inter-frame delays. Audio applications use playout buffers to mitigate some of these effects and increase audio quality. Arriving frames enter a playout buffer and are delayed for some amount of time to achieve a similar end-to-end delay for most frames within a talkspurt.

Many different playout buffer management algorithms exist [22]. Because the goal of this experiment is to investigate the impact of different transport protocols on audio quality – and *not* the impact of different playout algorithms – the experiment uses an offline, dynamic-programming-based algorithm based on [23] to compute an upper bound of the impairment caused by delay, based on the number of packets dropped in the playout buffer. The algorithm deduces the best possible playout sequence, i.e., the one that leads to the highest possible audio quality for the received voice frames. This establishes an upper bound on the achievable voice quality and eliminates the effects of different playout algorithms.

The logical result of the playout algorithm is yet another time-sequence trace of audio frame playout times. This second time-sequence trace is the input to the voice quality metric described in III-F. Note that for this experiment, the computation of voice quality scores has been included in the playout management algorithm, to reduce processing times.

F. Voice Quality Metric

One common metric for the perceived quality of a voice call is the Mean Opinion Score (MOS) [24], which is used in subjective quality evaluation tests and ranges from 1 ("poor") to 5 ("best"). The E-Model [25] is an online analysis method generating a score that can be translated to a MOS score.

TABLE II
CORRELATION OF R SCORES, MOS SCORES AND VOICE QUALITY [26].

R Score	MOS Score	Perceived Quality
90 – 100	4.34 – 4.50	Best
80 – 90	4.03 – 4.34	High
70 – 80	3.60 – 4.03	Medium
60 – 70	3.10 – 3.60	Low
50 – 60	2.58 – 3.10	Poor

TABLE III
COEFFICIENTS FOR QUALITY IMPAIRMENT DUE TO LOSS [26], [27].

Codec	Frames/Packet	λ_1	λ_2	λ_3
G.711	1	0	30.00	15
G.729	2	10	47.82	18

The E-Model defines a quality factor called the *R score* as $R = R_0 - I_s - I_e - I_d + A$, where R_0 captures the effects of noise, I_s those of impairment occurring simultaneously with the voice signal, I_e those of impairment caused by loss, I_d those of impairment caused by delay and A compensates for the above impairments under certain conditions. The MOS score corresponding to the R score is obtained by $MOS = 1 + 0.035R + 7{\times}10^{-6}R{\times}(R\text{-}60){\times}(100\text{-}R)$. Table II shows the mapping of R factors to MOS scores together with an approximation of the user-perceived quality.

Delay and loss, i.e., the factors I_d and I_e, are the main factors affecting voice quality in a packet-switched network. A reduction of the E-Model for these environments [26] is: $R = 94.2 - I_e - I_d$.

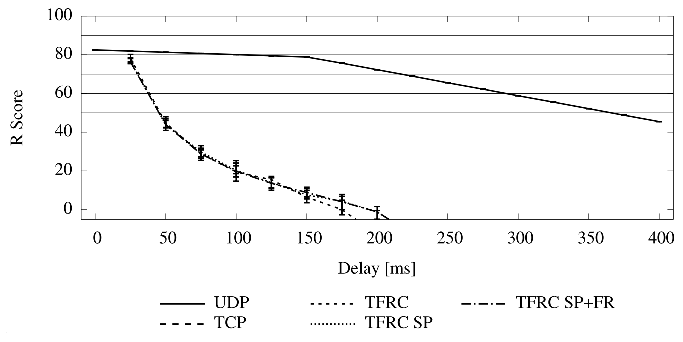
Experiments have shown that the loss factor can be approximated by $I_e = \lambda_1 + \lambda_2 \ln(1 + \lambda_3 e)$. Here, λ_1 quantifies the voice quality degradation due to the codec, λ_2 and λ_3 quantify the degradation due to loss and e is the overall packet loss rate. Values for the λ parameters have been determined through simulations of different loss conditions for different codecs [26], [27]. Table III shows the values for the G.711 and G.729 codecs used for these experiments.

The impact of delay can be similarly modeled [26] as $I_d = 0.024d + 0.11(d\text{-}177.3)I(d\text{-}177.3)$, where $I(x)$ is a unity step function and d is the total end-to-end delay.

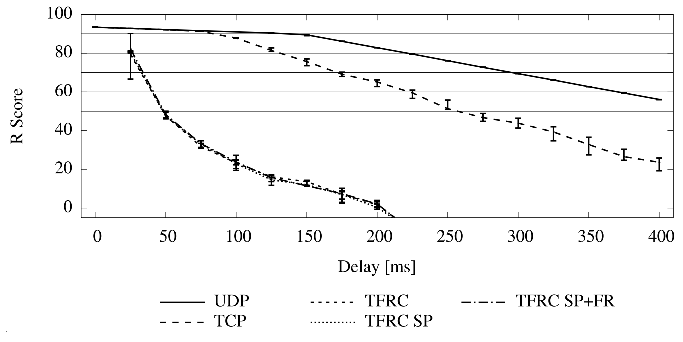
Based on these observations, Section IV presents R scores based on the time-sequence playout traces taken for voice calls encoded with different codecs over different transport protocols and transmitted over paths with different loss and delay characteristics.

IV. EXPERIMENTAL EVALUATION

This experimental evaluation investigates voice quality for UDP, TCP and three variants of DCCP: basic TFRC (termed simply "TFRC"), the *small packet* variant of TFRC (termed "TFRC SP") and a combination of the *small packet* and *faster restart* variants (termed "TFRC SP+FR"). Section IV-A evaluates voice quality in a network scenario with varying

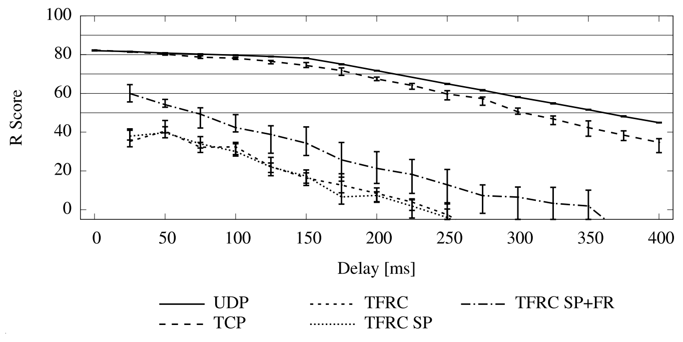


(a) Results for G.729.

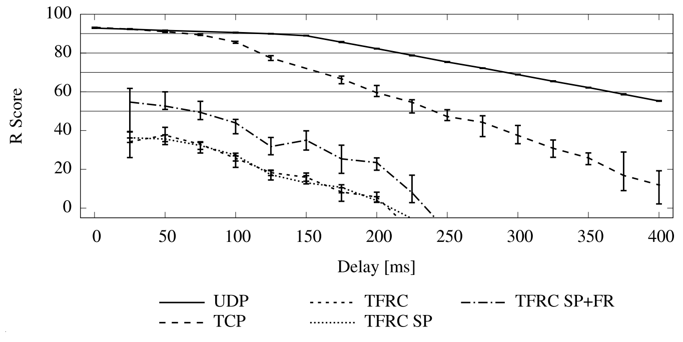


(b) Results for G.711.

Fig. 1. Median R scores over 15 runs with corresponding interquartile gaps as error bars, for five different transport protocols and varying one-way delays from 0 to 400 ms and no losses, using G.729 and G.711.



(a) Results for G.729.



(b) Results for G.711.

Fig. 2. Median R scores over 15 runs with corresponding interquartile gaps as error bars, for five different transport protocols and varying one-way delays from 0 to 400 ms and a fixed loss rate of 0.1%, using G.729 and G.711.

delay and no loss, Section IV-B evaluates voice quality in a network scenario with varying delay and a fixed loss rate and Section IV-C evaluates voice quality in a network scenario with fixed delay and a varying loss rate.

A. Varying Delay, No Loss

In the first experiment, one-way path delay varies from 0 to 400 ms. No packet loss rate is configured. Figure 1(a) shows the experimental results for G.729 and Figure 1(b) shows the corresponding results for G.711. These figures – and all others in this paper – graph median R scores based on 15 runs, together with the corresponding interquartile gaps as error bars. All figures also include vertical lines at R scores that correspond to the user-perceived voice qualities in Table II.

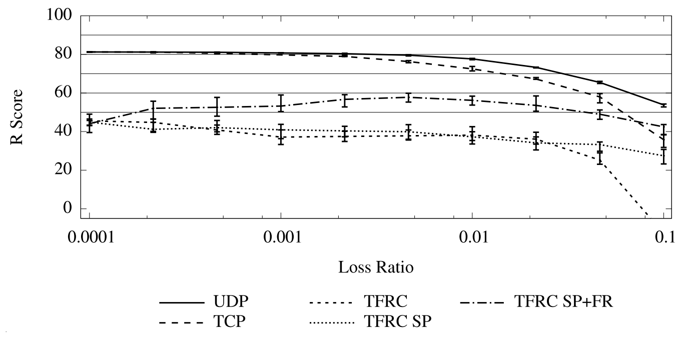
The R scores for UDP in this scenario illustrate the maximally achievable voice quality over G.711 and G.729. Voice quality over UDP using G.729 is roughly 10 points below that with G.711 at low delays, reflecting the difference in data rates generated by the two codecs. Also, note that this difference diminishes with increasing delays. With a delay of 400 ms, for example, R score for G.711 and G.729 are very similar. This illustrates how the growing impairment due to delay offsets the initial advantage of G.711.

The main result of this experiment is that voice quality over *any* variant of TFRC is significantly worse than over either UDP or TCP. It quickly drops below an R score of 50, i.e., "poor" perceived voice quality, when the one-way delay exceeds 50 ms. UDP only reaches a quality this low when the delay exceeds 375–425 ms, depending on the codec. Even TCP can sustain a better voice quality up to delays of 250 ms or higher, depending on the codec.

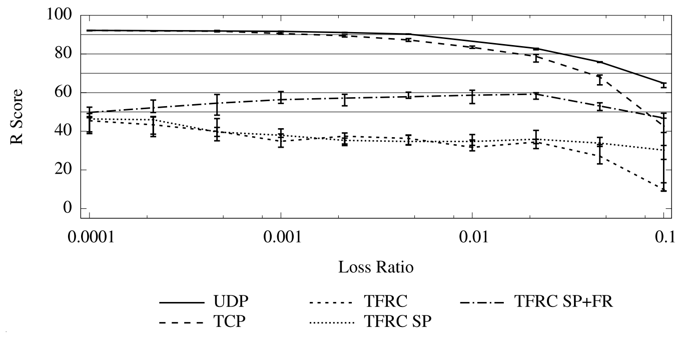
This result is due to the behavior of the TFRC feedback mechanism that reports the receive rate. After an idle period, the first data packet that reaches the receiver triggers a reply that contains the reported receive rate, which the receiver computes by dividing the number of bytes received since the last receive rate report by the time since that receive report was sent – which *includes* the idle period. Consequently, depending on the length of the idle period, the receive rate report significantly understates the actual receive rate before the connection went idle. This causes the sender to resume transmission at a much lower rate than before the idle period.

A second observation of this experiment is that voice quality over basic TFRC and TFRC SP is identical. This is surprising, because TFRC SP is allowed to inject as many small packets in the network as desired, as long as the aggregate bandwidth use is equivalent to that of basic TFRC using maximum-sized packets. This should theoretically increase voice quality. The reason this does not occur is that the miscalculated send rate – as described in the previous paragraph – is low enough that it prohibits TFRC SP from injecting these additional packets.

A third result is that voice quality for TFRC SP and TFRC SP+FR is identical. TFRC SP and TFRC SP+FR only differ in their operation after slow-start. TFRC variants exit slow-start when they detect that the first packet loss, which does never occur in this scenario.



(a) Results for G.729.



(b) Results for G.711.

Fig. 4. Median R scores over 15 runs with corresponding interquartile gaps as error bars, for five different transport protocols and varying loss rates from 0.01% to 10% and a fixed one-way delay of 50 ms, using G.729 and G.711.

A fourth observation is that voice quality over UDP and TCP is exactly the same, as long as the codec does not exceed the send rate allowed by TCP's initial window. This is the case for the low-bandwidth G.729, as illustrated by Figure 1(a), where the graphs for UDP and TCP overlay. It is not the case for the higher-bandwidth G.711, as illustrated by Figure 1(b), where voice quality for TCP degrades at delays exceeding 100 ms. The reason for this difference is that FreeBSD 5.4 enables the TCP *large initial window* extension [28] by default, which results in an effective minimal send rate that is larger than the data rate generated by G.729, but not larger than that of G.711. Consequently, TCP congestion control can delay G.711 frames, resulting in the voice quality difference illustrated in Figure 1(b).

Finally, note that due to stability issues with the prototype DCCP implementation in network configurations with very low delays, no experimental results are available for the TFRC variants at 0 ms configured delay.

B. Varying Delay, Fixed Loss

The second series of experiments repeats the first, but adds a small fixed loss rate of 0.1% on the path. This allows investigation of TFRC performance after the slow-start phase. Again, one-way delay varies from 0 to 400 ms. Figure 2(a) shows the experimental results for G.729 and Figure 2(b) shows the corresponding results for G.711.

The main result is that even though the performance of the TFRC variants has increased compared to the previous experiment, it is still significantly worse than either UDP or TCP. For basic TFRC, this is because the rate calculation uses the actual packet size used when computing the permitted send rate, which is orders of magnitude smaller than the maximum-sized packets that the math in the underlying TCP model assumes. This results in an permitted send rate that is significantly below that of TCP.

TFRC SP is supposed to mitigate this known limitation of basic TFRC by changing the TFRC rate computation to take actual packet size into account. One surprising result of this experiment is that the use of TFRC SP does not significantly improve voice quality with either G.729 or G.711.

This result is again due to the problem described in Section IV-A, i.e., miscalculation of the receive rate after idle periods. The difference in voice quality in this scenario, compared to the previous one, is due to the small loss rate present. Unlike in the previous scenario, these losses cause the TFRC variants to exit slow-start and enter congestion avoidance, where their congestion control behavior is different. The moderately better performance of TFRC SP+FR in this scenario, which was identical to the other TFRC variants in the previous scenarios, illustrates this difference. TFRC SP+FR quadruples the send rate during slow-start-restart, i.e., it reaches a sufficient send rate to sustain the data rate of the codecs more quickly, increasing voice quality.

Note that the performance of basic TFRC and TFRC SP is still very similar. This is again because that the miscalculated send rate prohibits TFRC SP from injecting additional packets.

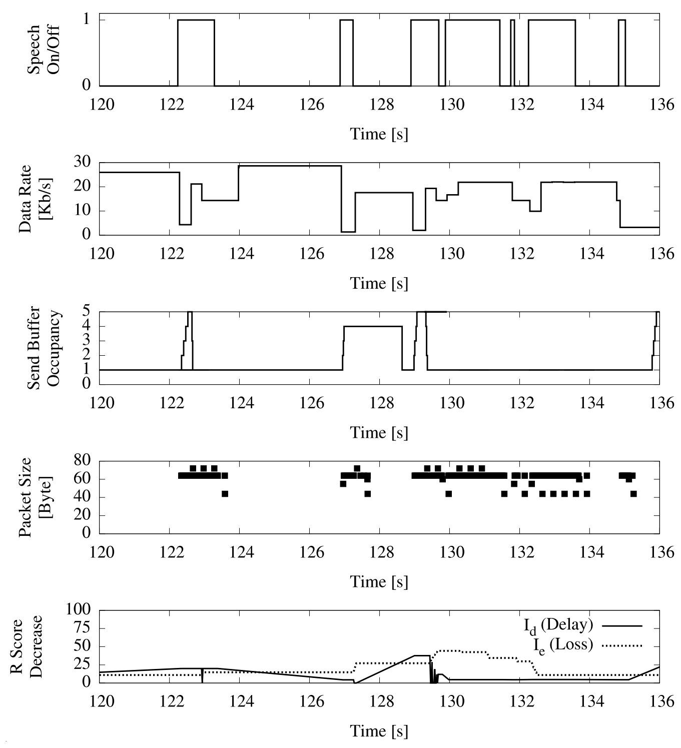
A final result of this scenario is that a lower-bitrate codec, such as G.729, can maintain a voice quality over TCP that is reasonably close to that of UDP. Voice quality with a higher-bandwidth codec, such as G.711, however, decreases quickly beneath that of the lower-bandwidth codec.

C. Fixed Delay, Varying Loss

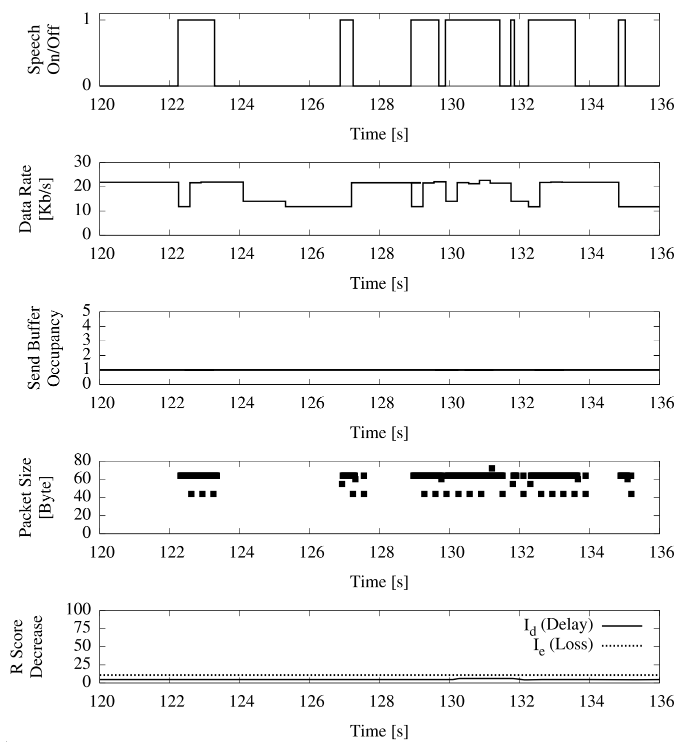
The third series of experiments investigate the impact of losses on the voice quality achievable over the different transport protocols. Here, one-way delay is fixed at 50 ms and loss rates vary from 0.01% to 10%. Figure 4(a) shows the experimental results for G.729 and Figure 4(b) shows the corresponding results for G.711.

The main result of this scenario is that all TFRC variants perform worse than TCP in almost all cases. TFRC only starts to outperform TCP when loss rates approach 10%, but voice quality in those cases is still poor (R score $\leq$ 50). This is surprising, because TCP needs to retransmit lost packets, increasing the playout delay, while DCCP does not.

One reason for this result is that the FreeBSD TCP stack enables the TCP *selective acknowledgments* (SACK) [29] and *limited transmit* [30] extensions by default. These extensions improve TCP operation over lossy paths and with small windows, which are both present in this scenario. A second reason for this result is that voice quality impairment for TCP is only due to playout delay increases due to retransmissions and for DCCP it is only due to losses. With a 50 ms path delay, impairment due to loss may have a greater impact on voice



(a) Behavior with TFRC SP+FR.



(b) Behavior with TFRC SP+FR+MD.

Fig. 3. Voice activity, data rate after encoding, buffer occupancy at the sender, packet transmission details and transient voice quality (from top to bottom) of the same 16-second call excerpt encoded with G.729 and transmitted with both TFRC SP+FR (left) and TFRC SP+FR+MD (right) over a network path with 0.01% loss rate and a one-way delay of 150 ms.

quality than impairment due to increases in playout delay. Additional experiments are needed to validate this hypothesis.

As in the previous scenarios, voice quality of all TFRC variants is significantly below those of UDP and TCP, even at very low loss rates of 0.01%. TFRC SP+FR is the best TFRC variant; the performance of basic TFRC and TFRC SP is again very similar. This is again mostly due to the miscalculation of the send rate, as described in the previous sections.

V. Discussion

Section IV experimentally analyzed the voice quality that is achievable over basic TFRC, TFRC SP, TFRC SP+FR as well as UDP and TCP. The main finding of this experimental analysis is that none of the currently proposed TFRC variants result in voice qualities that approach that of UDP. More importantly, they all perform significantly worse than TCP even under losses, where TCP's need for retransmissions can introduce significant additional playout delays or cause the playout algorithm to skip received audio frames.

These results are due to a number of shortcomings of the currently proposed TFRC variants: miscalculation of the permitted send rate after idle periods, issues during slow-start-restart and miscalculation of the first loss interval.

Miscalculation of the permitted send rate after an idle period occurs, because the receiver computes the receive rate by dividing the number of bytes received since the last receive rate report by the time since that report was sent. After an idle period, this time *includes* the idle period itself. Consequently, depending on the length of the idle period, the receive rate report significantly understates the actual receive rate before the idle period. This causes the sender to resume transmission at a much lower rate than necessary.

The second issue with the currently proposed TFRC variants affects their slow-start-restart behavior after idle periods. The initial window during slow-start-restart is as low as eight small packets per RTT for TFRC SP+FR. TCP, on the other hand, starts at the byte rate described in [28]. Because TFRC restarts at a much lower rate, it requires more round-trips to reach the same rate as TCP. TFRC is further restricted by maintaining its window in terms of packets rather than bytes; TCP can inject a comparatively larger number of (small) packets per window.

The third issue with the current TFRC variants affects the method for initializing the loss history after the first loss event [6]. TFRC uses the receive rate over the last RTT to approximate the sending rate, and uses that to initialize the loss history. One issue with this approach is that the receive rate can be miscalculated, as described above. A second issue is that for interactive, bursty traffic, if little traffic was received during the last RTT, the resulting send rate is also too low.

Based on this analysis, this paper proposed a modification to the combined *small packet* and *faster restart* variant of TFRC, termed TFRC SP+FR+MD. TFRC SP+FR+MD makes two changes compared to the original proposals.

First, it maintains a minimum send rate of eight packets per RTT. This is in the spirit of TFRC and TFRC SP, because it uses bandwidth similarly to TCP with full-sized packets (and less bandwidth than TCP with large initial windows).

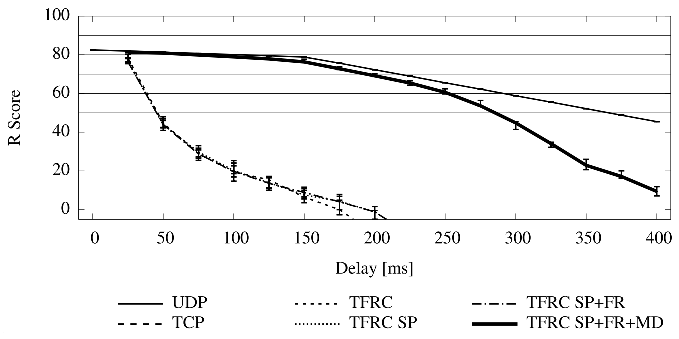
Second, TFRC SP+FR+MD adds an additional *post_idle* state to the TFRC state machine. When sending the first packet while in the *idle* state, the sender changes to the new *post_idle* state and remains in it until receiving the first feedback packet, after which it enters the *not_idle* state. Because the receive rate in this first feedback packet may understate the true receive rate (as described above), it is discarded and does not enter into the send rate calculation. Send rate after the idle period is the minimal rate of 8 packets/RTT. Note that although the receive rate information in the first feedback packet during *post_idle* is ignored, information about loss events *is* acted on.

Figure 3 illustrates the impact that the different send rate calculation schemes have. The same 16-second voice call excerpt is encoded with G.729 and then transmitted with both TFRC SP+FR and TFRC SP+FR+MD. Figure 3(a) on the left and in Figure 3(b) on the right show voice activity, data rate after encoding, buffer occupancy at the sender, packet transmission details and transient voice quality (from top to bottom) for TFRC SP+FR and TFRC SP+FR+MD, respectively. Transmission occurs over a network path with 0.01% loss rate and a one-way delay of 150 ms.

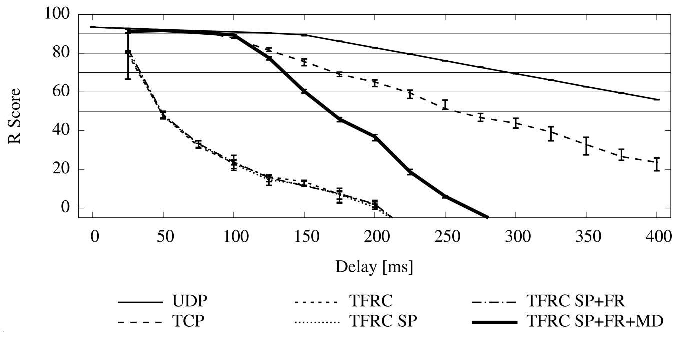
During the 16-second excerpt shown in Figure 3, both TFRC variants are in congestion avoidance. For TFRC SP+FR, Figure 3(a) illustrates one example of voice quality impairment. When a talkspurt starts around second 127, the send rate drops sharply, due to the reception of miscalculated receive rate report. This causes the send buffer to fill up, which in turn causes additional queuing delays. This becomes evident in the increase in the delay impairment I_d. After the send buffer is full (with five packet queued), further packets are dropped at the sender, and the loss impairment I_e starts increasing.

Figure 3(b) illustrates the exact same sequence for TFRC SP+FR+MD. Here, the send rate does not drop near zero – both because it is bounded at 8 packets/RTT and because the miscalculated receive rate feedback is ignored – and transmission occurs without losses or delays, resulting in no voice quality impairment.

Even when these issues are corrected – either with TFRC SP+FR+MD or an equivalent that may be designed in the relevant IETF working group – one underlying issue remains. TFRC is based on a mathematical model of TCP Reno under simplifying conditions. TCP Reno is not a modern TCP. A modern TCP implementation with large initial windows, selective acknowledgments and limited transmit extensions is significantly more efficient than TCP Reno, and also somewhat more aggressive. Proposed experimental extensions, such as *appropriate byte counting* [31], may further improve TCP transmission behavior. Because the goal of TFRC is to use bandwidth approximately in the same way as TCP, using a modern TCP instead of TCP Reno should be a safe approach to improving TFRC performance. Finally, TFRC suffers from the same limitations as TCP when transmitting over paths with non-congestion-based losses, and can benefit from techniques similar to those that have been proposed to improve TCP performance over such paths.



(a) Results for G.729.



(b) Results for G.711.

Fig. 5. Median R scores over 15 runs with corresponding interquartile gaps as error bars, for six different transport protocols and varying one-way delays from 0 to 400 ms and no losses, using G.729 and G.711.

The TFRC SP+FR+MD variant will not increase the sending rate after an idle period until after receiving the second feedback packet. For increased performance, a future implementation might take advantage of the loss interval option [5] provided in the first feedback packet in order to infer that no path congestion has been detected and the doubling of the send rate can take place right away.

VI. Experimental Validation of a Modified TFRC

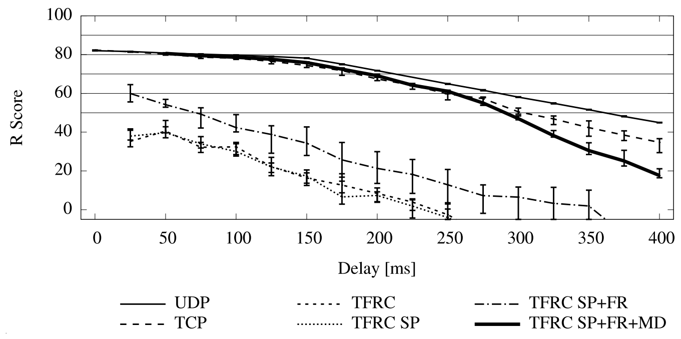
This section repeats the experiments from Section IV in order to evaluate the voice quality that a TFRC SP+FR+MD prototype implementation achieves in the same scenarios.

The voice quality of TFRC SP+FR+MD is illustrated with a thicker solid line in all graphs below. Voice quality over the other TFRC variants, UDP and TCP is included for comparison but remains unchanged from Section IV.

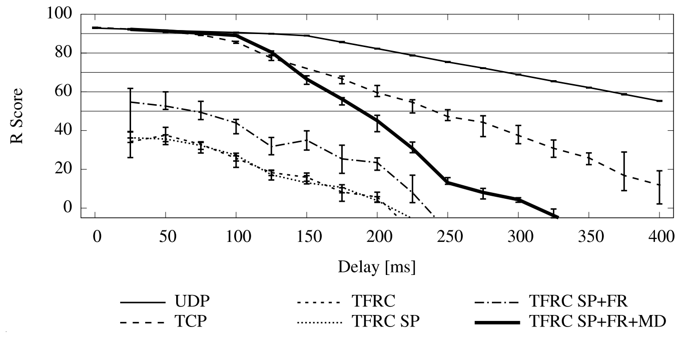
Section VI-A evaluates voice quality in a network scenario with varying delay and no loss, Section VI-B evaluates voice quality in a network scenario with varying delay and a fixed loss rate and Section VI-C evaluates voice quality in a network scenario with fixed delay and a varying loss rate.

A. Varying Delay, No Loss

In this scenario, one-way path delay varies from 0 to 400 ms. No packet loss rate is configured. Figure 5(a) shows

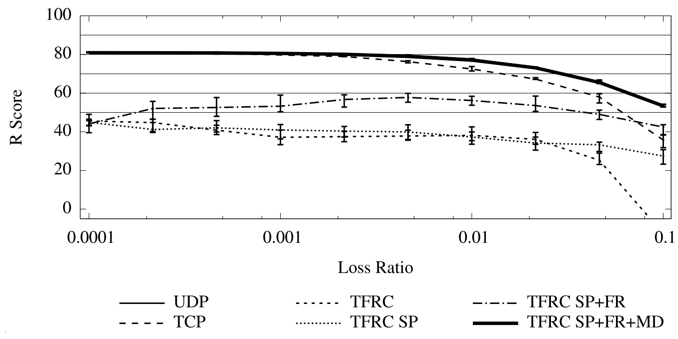


(a) Results for G.729.

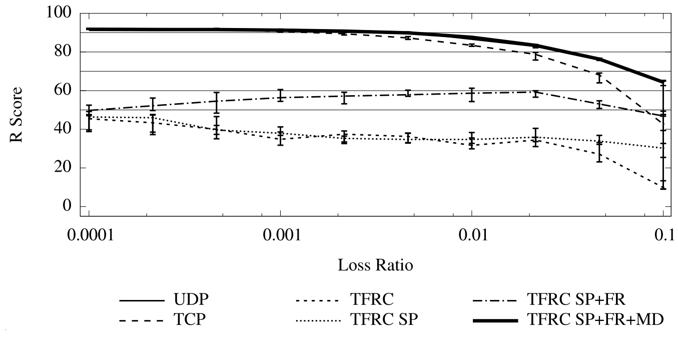


(b) Results for G.711.

Fig. 6. Median R scores over 15 runs with corresponding interquartile gaps as error bars, for six different transport protocols and varying one-way delays from 0 to 400 ms and a fixed loss rate of 0.1%, using G.729 and G.711.



(a) Results for G.729.



(b) Results for G.711.

Fig. 7. Median R scores over 15 runs with corresponding interquartile gaps as error bars, for five different transport protocols and varying loss rates from 0.01% to 10% and a fixed one-way delay of 50 ms, using G.729 and G.711.

the experimental results for G.729 and Figure 5(b) shows the corresponding results for G.711.

Voice quality with TFRC SP+FR+MD is significantly higher than that achieved over the other TFRC variants. Up to delays of 100 ms, it is identical to that of UDP for both G.729 and G.711. For G.729, it remains close to that of UDP up to 250 ms delay. For G.711, it quickly degrades when delays exceed 100 ms. This is due to feedback arriving less and less frequently with increasing path delays. For the higher-bandwidth G.711 codec, this effect is more pronounced.

Note, however, that TCP still outperforms TFRC SP+FR+MD in this scenario. Because there are no losses, TCP voice quality is identical to that over UDP for the lower-bitrate G.729 codec and moderately worse for higher-bitrate G.711 codec, due to delays introduced by restart after idle periods.

B. Varying Delay, Fixed Loss

The second series of experiments repeats the first, but adds a small fixed loss rate of 0.1% on the path. This allows investigation of behavior after the slow-start phase. Again, one-way delay varies from 0 to 400 ms. Figure 6(a) shows the experimental results for G.729 and Figure 6(b) shows the corresponding results for G.711.

TFRC SP+FR+MD voice quality again is significantly above that of other TFRC variants. It is also slightly better than TFRC SP+FR+MD in the previous scenario, especially when path delays exceed 250 ms. This is because in slow-start, the send rate doubles, but in congestion avoidance – after the first loss has occurred – it is allowed to quadruple, due to the use of *faster restart*.

As in the previous experiment, TCP still outperforms TFRC SP+FR+MD. However, the difference in voice quality between TCP and TFRC SP+FR+MD has diminished compared to the previous case, especially at delays above 200 ms, where the occasional retransmissions start to have a noticeable impact on the R score, independent of the codec.

C. Fixed Delay, Varying Loss

The final series of experiments investigate the impact of losses on the voice quality of TFRC SP+FR+MD. Here, one-way delay is fixed at 50 ms and loss rates vary from 0.01% to 10%. Figure 7(a) shows the experimental results for G.729 and Figure 7(b) shows the corresponding results for G.711.

In this scenario, TFRC SP+FR+MD again outperforms the other TFRC variants, as before, but also outperforms TCP. Voice quality is identical to that over UDP for both G.729 and G.711. This is the desired TFRC behavior: congestion-controlled transmission at a voice quality similar to UDP.

VII. Conclusion and Future Work

This paper has experimentally analyzed the voice quality that is achievable over basic TFRC, TFRC SP, TFRC SP+FR as well as UDP and TCP. The main finding of this experimental analysis is that none of the currently proposed TFRC variants result in voice qualities that approach those over UDP.

More importantly, they all perform significantly worse than TCP even under losses, where TCP's need for retransmissions can introduce significant additional playout delays.

The paper has analyzed the reasons for this behavior, which is due to several issues with the currently specified TFRC mechanisms. Based on this analysis, the paper described an improved TFRC variant and implemented in the KAME DCCP prototype implementation. It then experimentally compares the modified DCCP against the currently-specified variants as well as UDP and TCP. It finds that voice quality of the modified DCCP is significantly better and approaches that achievable with UDP in some scenarios. The DCCP working group in the IETF has incorporated some of TFRC SP+FR+MD's improvements into the specification.

However, a modern TCP implementation that enables extensions such as *selective acknowledgments*, *limited transmit* and *appropriate byte counting* can sustain voice calls at similar or even better quality, at least in the scenarios investigated in this paper. These extensions make TCP significantly more efficient than the TCP Reno variant that underlies TFRC's analytical model of TCP behavior. TFRC is consequently less aggressive than modern TCPs.

A future continuation of this research will hence investigate how TFRC's bandwidth use and sending behavior can be aligned more closely to that of modern TCPs. Because this has the potential to make TFRC more aggressive, it is important to investigate fairness on competing traffic, especially along congested paths. An initial investigation has already been started [9]; but the preliminary results could not be included in this paper due to space restrictions. Finally, other recent work has shown that, depending on the codec, some audio frames are more important to maintain a high voice quality than others [32]. These results can have impact on send buffer management and potentially also router queue management.

Acknowledgment

The authors thank Sally Floyd, Eddie Kohler, Yoshifumi Nishida, Arjuna Sathiaseelan and Jürgen Schönwälder for their comments on this research. Lars Eggert and Marcus Brunner were partly supported by *Ambient Networks*, a research project funded by the European Commission under its Sixth Framework Program.

References

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," RFC 3261, June 2002.

[2] S. Guha, N. Daswani, and R. Jain, "An Experimental Study of the Skype Peer-to-Peer VoIP System," in *Proc. International Workshop on Peer-to-Peer Systems (IPTPS)*, Santa Barbara, CA, USA, Feb. 2006.

[3] E. Kohler, S. Floyd, and M. Handley, "Designing DCCP: Congestion Control Without Reliability," in *Proc. ACM SIGCOMM*, Pisa, Italy, Sept. 2006.

[4] E. Kohler, M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)," RFC 4340, Mar. 2006.

[5] S. Floyd, E. Kohler, and J. Padhye, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC)," RFC 4342, Mar. 2006.

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

[7] S. Floyd and E. Kohler, "TCP Friendly Rate Control (TFRC): the Small-Packet (SP) Variant," Internet Engineering Task Force, Internet-Draft draft-ietf-dccp-tfrc-voip-02, July 2005, work in progress.

[8] E. Kohler and S. Floyd, "Faster Restart for TCP Friendly Rate Control (TFRC)," Internet Engineering Task Force, Internet-Draft draft-ietf-dccp-tfrc-faster-restart-00, July 2005, work in progress.

[9] V. Balan, "An Experimental Evaluation of Voice-over-IP Quality over the Datagram Congestion Control Protocol," MS Thesis, International University Bremen, Germany, June 2006.

[10] Y. Nishida, "A Preliminary DCCP Implementation on BSD," 2006, http://www.jp.nishida.org/dccp/.

[11] F. Nivor, "Experimental Study of DCCP for Multimedia Applications," in *Proc. ACM Conference on Emerging Network Experiments and Technologies (Student Workshop)*, Toulouse, France, 2005, pp. 272–273.

[12] C. Xu, J. Liu, and C. Zhao, "Performance Analysis of Transmitting H.263 over DCCP," in *Proc. IEEE Workshop on VLSI Design and Video Technology*, May 2005, pp. 328–331.

[13] S. Takeuchi, H. Koga, K. Iida, Y. Kadobayashi, and S. Yamaguchi, "Performance Evaluations of DCCP for Bursty Traffic in Real-time Applications," in *Proc. Symposium on Applications and the Internet (SAINT)*, Jan. 2005, pp. 142–149.

[14] S. Floyd and E. Kohler, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control," RFC 4341, Mar. 2006.

[15] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP Reno Performance: A Simple Model and its Empirical Validation," *IEEE/ACM Transactions on Networking*, vol. 8, no. 2, pp. 133–145, 2000.

[16] K. Sriram and W. Whitt, "Characterizing Superposition Arrival Processes in Packet Multiplexers for Voice and Data," *IEEE Journal on Selected Areas in Communication (JSAC)*, vol. 4, no. 6, pp. 833–846, 1986.

[17] G. Herlein, J.-M. Valin, S. Morlat, R. Hardiman, and P. Kerr, "RTP Payload Format for the Speex Codec," Internet Engineering Task Force, Internet-Draft draft-ietf-avt-rtp-speex-00, Oct. 2005, work in progress.

[18] ITU-T, "Pulse code modulation (PCM) of voice frequencies," ITU-T Recommendation G.711, Nov. 1988.

[19] ITU-T, "Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear-prediction (CS-ACELP)," ITU-T Recommendation G.729, Mar. 1996.

[20] Newport Networks, "VoIP Bandwidth Calculation," *White Paper*, 2005.

[21] L. Rizzo, "Dummynet: A Simple Approach to the Evaluation of Network Protocols," *ACM Computer Communication Review (CCR)*, vol. 27, no. 1, pp. 31–41, 1997.

[22] R. Ramjee, J. F. Kurose, D. F. Towsley, and H. Schulzrinne, "Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Networks," in *Proc. IEEE INFOCOM*, Toronto, Canada, 1994, pp. 680–688.

[23] S. B. Moon, J. Kurose, and D. Towsley, "Packet Audio Playout Delay Adjustment: Performance Bounds and Algorithms," *Multimedia Systems*, vol. 6, no. 1, pp. 17–28, 1998.

[24] ITU-T, "Methods for subjective determination of transmission quality," ITU-T Recommendation P.800, Aug. 1996.

[25] ITU-T, "The E-model, a computational model for use in transmission planning," ITU-T Recommendation G.107, Mar. 2005.

[26] R. G. Cole and J. H. Rosenbluth, "Voice over IP Performance Monitoring," *ACM Computer Communication Review (CCR)*, vol. 31, no. 2, pp. 9–24, 2001.

[27] L. Ding and R. A. Goubran, "Speech Quality Prediction in VoIP Using the Extended E-Model," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM)*, San Francisco, CA, USA, Dec. 2003, pp. 3974–3978.

[28] M. Allman, S. Floyd, and C. Partridge, "Increasing TCP's Initial Window," RFC 3390, Oct. 2002.

[29] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgement Options," RFC 2018, Oct. 1996.

[30] M. Allman, H. Balakrishnan, and S. Floyd, "Enhancing TCP's Loss Recovery Using Limited Transmit," RFC 3042, Jan. 2001.

[31] M. Allman, "TCP Congestion Control with Appropriate Byte Counting (ABC)," RFC 3465, Feb. 2003.

[32] C. Hoene, S. Wiethölter, and A. Wolisz, "Calculation of Speech Quality by Aggregating the Impacts of Individual Frame Losses," in *Proc. International Workshop on Quality of Service (IWQoS)*, Passau, Germany, June 2005.