Proceedings of the IEEE 21st International Conference on Advanced Information Networking and Applications (AINA-07), Niagara Falls, Canada, May 21-23, 2007, pp.687-694

Symbiotic Audio Communication on Interactive Transport

Javed I. Khan and Olufunke I. Olaleye
Networking and Media Communications Research Laboratories
Department of Computer Science, Kent State University
Kent, OH 44242, javed|oolaleye@cs.kent.edu

Abstract

Congestion control for compressed audio is a nontrivial pursuit. Audio perception is highly susceptible to disturbance in temporal quality. We recently proposed and implemented a novel symbiotic perceptual audio streaming mechanism, which receives fast feedback from network and responds to impending congestion with advanced auditory perception based adaptation techniques and supports near flawless sound. The design is based on recently proposed iTCP. It enables a new principle where encoder trade-off fidelity for accurate audio temporal quality using fast network event feedback. We have tested it live over the Internet. In this paper, we share the performance of this system and report observed improvements in time-bounded streaming audio.

1 Introduction

Audio including voice and music has become a major contributor to the Internet traffic. Besides the growth of VOIP, many household applications such as IM®, Skype®, today regularly provide voice service. Audio perception is highly susceptible to disturbance in temporal quality. The audio quality degrades when the packets face congestion. This is because almost all currently deployed schemes for congestion response are based on delaying packets at various network points, or packet dropping (prioritization, admission control etc). However, these introduce time distortion in the transport pathway. This is the last thing an audio stream seeks. Mere use of UDP does not solve the problem [7]. This only circumvents the window-buffer clogging caused by late packets in TCP. Network itself does not get any relief as application continues generating packets at full force. Inside network one lost packet with critical field may mean loss of a long sequence of UDP data for the application. Loss and delay processes inside network do not discriminate between UDP or TCP datagrams.

It seems applications have to be included in congestion control for time-sensitive traffic. Audio algorithms are known that can minimize the impact of current packet delays and losses. These algorithms can adapt the size of the playout buffer [1], the coding rate [4], the amount of forward error correction and packet path diversity [8] in order to maximize the audio/voice quality. However, a major problem faced by them is to receive any feedback from the network about congestions. Recently introduced TCP-friendly [2] approach use RTP based end-to-end measurement to help some applications at-least to be indirectly aware of network delay. But, for audio the time constant in such measurement are often too slow.

A recently proposed scheme of iTCP [6] now provides transport interactivity needed for such fast feedback. iTCP makes a subset of its internal events/states accessible. It uses a decoupling formalism called *Transientware* (T-ware) that application-level components can selectively invoke when subscribed events occur at the transport layer.

Based, on iTCP, in this paper we present an alternate approach to audio congestion control which, instead of relying on packet *delaying and dropping principles,* employs *information shaping*. In this approach an application (the actual source) receives direct feedback from the network end-point (from transport such as iTCP) and dynamically adjusts the very generation of data. We call it network/application symbiosis. This also eliminated the TCP buffer-clogging. The principle is useful for *elastic-traffic*- where generated data volume can be traded-off for temporal quality. Data forms intended for perceptual consumption including audio and video seem to offer such elastic property.

The resulting streaming scheme is similar in spirit to the TCP-friendly approaches. But, instead of end-to-end measurement the network state is received directly from local transport end-point. Thus the response becomes much faster and effective. A network architecture such as iTCP that is interactive[1] and transparent can open up a new paradigm of building applications (and systems), which are genuinely network friendly. Generally more

[1] The original TCP proposal (RFC007, RFC793) did call for interactivity, but subsequent implementations ignored it.

sophisticated and targeted solutions based on new principles (such as information shaping) can be designed at the application layer, thus drastically reducing the complexity of network layers.

In this paper, we present this **symbiotic audio streaming** system. Previously we have demonstrated a similar approach on video streaming [5]. We have implemented and tested the symbiotic audio system on the FreeBSD iTCP systems. This paper reports the performance. As we will show, the overall solution is not only intuitive and simple, but also, surprisingly effective when compared to many other recently proposed schemes, which have involved much more complex system/network layer reorganization.

2 iTCP: interactive TCP

2.1 Model Architecture

Recently proposed *Interactive Transparent Networking* [5,6] framework enables network applications to receive instant notification about the service state information from various network layers on which it critically relies. Briefly, the framework provides a means to application programs running in the upper layers to subscribe to a selected set of transport events (details in [5,6]). By subscribing, an application program (optionally) declares that it wishes to be notified through signaling when certain events (state transitions) occur at a specified protocol. Once they are notified, subscribers can also pull up the required service state information when available, perform the actual action by components called *Transientware Modules* (or *T-ware),* which run at the application layer. The *T-ware* can use transport state parameters to determine precise adaptive action plan for the application (for example calculate a new generation schedule that guarantees certain delivery time bound of the traffic). Figure 1 shows the general architecture of TCP Interactive (iTCP) -- a TCP which can provide such event feedback. Empowered with iTCP the design of a symbiotic system involves two parts (a) an enhanced elastic traffic encoder with dynamic generation rate control ability and (b) a symbiosis throttling system which glues the elastic traffic encoder with the network events described next.

3 Symbiosis Throttling System

Its key task is to specify the target rate for the application based on the event interrupt. It accepts the event feedback provided by the underlying interactive transport layer, and generates a corresponding rate value feedback for rate formation capable applications. Based on the specific audio instance and a tolerance level on its quality; the system should still be able to generate audio, however, with lesser audio quality based on precise quality/delay tradeoff boundaries of the audio. During normal operation, network bandwidth is readily available for the generation data rate of application. When a loss event is detected (e.g., timeout), the transport bandwidth retracts to some smaller effective value due to window resizing. The underlying cause might be a packet loss or a congestive delay deep inside the network. In response to the loss/time-out event (ζ=1), we let the subscriber adjust its generation rate to a lower generation state; the *frugal* state. Below we show a generation rate retraction logic which is similar to TCP.

3.1 Analysis of Symbiotic Throttling Model

Let the target bitrate at a normal state generation be b_{max}, the minimum acceptable rate for application be b_{min}. and b be the computed target rate at frugal state. Assuming the loss is detected at t_{loss}, the window rate decreases below the generation rate, the sender buffer builds up and results to delay. In an effort to reduce delay, we show a throttling system similar to TCP (exponential back off and additive – increase and a piecewise step of b(t).

Based on the tolerance level of the quality, we define a ratio called *rate retraction ratio*.

$$\rho = \frac{b}{b_{\max}} \qquad \text{-- (1)}$$

For symbiosis with the underlying TCP, we define a running generation threshold function as following:

$$b_T(t) = \frac{1}{2} b(t-1) \quad when \quad \xi = 1$$
$$= b_T(t-1) \quad otherwise \qquad \text{-- (2)}$$

The subscriber retracts and adjusts its rate to half its current size when a loss event occurs. The running control generation function b(t) is given by:

$$b(t) = \rho . b_{\max} \ when \ \xi = 1$$
$$= 2 \, . \, b(t-1) \ when \ b(t) \geq \frac{1}{2} b_T(t-1) \qquad \text{-- (3)}$$
$$= \min[b_{\max} \, , \, b(t-1) + 1] \ when \ b(t-1) \geq b_T(t-1)$$

The control generation function performs exponential-backoff and additive increase within the limits given by generation parameter ρ and normal mode target bitrate $b_{\max}$. The system enters a frugal state when the loss event occurs (ξ) and stays in the frugal state until the control (target bitrate) recovers to the normal target bitrate. The time of recovery is called $t_{recovery}$ and it is pre-defined.

3.2 Elastic Audio Rate Controller

Figure 2 shows the modified rate control design. Adaptive output audio stream in the encoder is achieved through the rate retraction ratio ρ applied to the target bitrate.

At the start point, the target bits/granule is determined. Each frame contains two granules. The target bits per granule is derived from bitrate, sampling frequency and side information and header.

$$T(t) = ((b(t).1152/samp\_freq) - sideinf - header)/2 = ((\frac{1}{2}(b(t).1152/samp\_freq) - sideinf - header)/2 \quad when \quad \xi = 1 \quad \text{-- (4)}$$

sideinf and *header* are the information for conveying information about the elementary stream data contained in the packet data. *samp-freq* is the frequency at which the audio is sampled. *T(t)*, target bits is the allocated bits determined for a granule before encoding. When a time-out event is detected, *T(t)* is recalculated to reflect the change in network state. *max-bits* is the needed bits based on the perceptual energy, which is the summation of a fraction of the target bits per granule and the current reservoir size. *max-bit* is referred to as *estimated target buffer fullness (ETBF)* in this paper.

$$ETBF = 0.9 * T(t) + R_s \quad \text{-- (5)}$$

As the target number of bits decreases, the *estimated target buffer fullness (ETBF)* also decreases. The reservoir size, R_s is updated after each frame is encoded. A is the actual number of bits encoded per granule

$$R_s = T - A \quad \text{-- (6)}$$

The noise allocation block in figure 2 uses the output of the psychoacoustic model "signal to mask" ratio (ratio of threshold to energy) to determine how to allocate the number of code available for quantization of subband signals, thus minimizing the audibility of the quantization noise.

It uses two nested iteration loops, distortion and rate control loop to find the optimum gain and scalefactors for a given bit-rate and output from the psycho-acoustic model in an analysis-by-synthesis way. It iteratively varies the quantizer in an orderly way, counts the number of Huffman code bit required to code a frame and calculates the resulting noise.

The outer iteration loop controls the masking conditions of all scalefactor bands. It computes the best scalefactor and global gain through the Quant-compare in figure 2. The first step of the outer loop is the inner iteration loop (rate control loop) that is responsible for quantization of the audio signal energy $ix_{(i)}$. The quantization is computed with the given scalefactor to determine if the overall bit sum is less than the available bits to encode a frame. It outputs the number of bits that Huffman code length table use to code information.

$$ix_{(i)} = \left(\frac{(\ |xr_{(i)}|\ )^{0.75}}{\sqrt[4]{2^{global\_gain}}} - 0.0946\right) \quad \text{-- (7)}$$

The *global gain* is summation of *qquant* (counter for the quantizer step size) and *quantanf,* system-constant for all the scaling operations of the encoder and the offset to achieve the correct output with decoding process. The distortion *xfsf(sb)* within each scalefactor band in this quantization is then computed

$$xfsf(sb) = \sum_{lbl(sb)}^{i=ibl(sb)+bw(sb)-1} \frac{(|xr_{(i)}| - ix_{(i)}^{4/3} * \sqrt[4]{2^{qquant + quantf}})^2}{bandwidith\ (sb)} \quad \text{-- (8)}$$

The *lbl(sb)* is the number of the component representing the lowest frequency in a scalefactor and *bw(sb)* is the number of component within this band. The *bandwidth(sb)* is number of scalefactor band (window type). The computed distortion in the quantization *xfsf(sb)* of a particular scalefactor band j is compared with the allowed distortion *xmin(sb)*

$$xmin(sb) - xfsf(sb) \text{ of scalefactor band j} < 0) \quad \text{-- (9)}$$

$$xr_{(i)} = xr_{(i)} * ifqstep \quad \text{-- (10)}$$

$$x\_\min(sb) = ratio\ (sb) * \frac{en\ (sb)}{bw\ (sb)} \quad \text{-- (11)}$$

The $ratio\ (sb)$ is ratio of threshold to energy in each scalefactor band and *en(sb)* is the energy in each scalefactor band.

$$ifqstep = 2^{(0.5\ *\ (1+scalefac\ -scale))} \quad \text{-- (12)}$$

All the values of the scalefactor bands that have a distortion that exceeds the allowed distortion are amplified by a factor of *ifqstep* . It is then requantized by the adaptive quantization step size. The *scalefac* and *scale* in eq. 12 are referenced from a table.

When a scalefactor band is amplified, it makes the next quantization to use more bits for that band. Thus, more bits are used to encode frames in that band but with a less quantization.

It quantizes the new scalefactor, global gain and computes the distortion. It iteratively uses different scalefactor and global gain until the best quantization with allowed distortion to sharpen the noise is achieved by the quant-compare.

The best quantized values are coded using Huffman code and other side information into bit streams. With the influence of the rate retraction ratio ρ and quantization, the final audio output stream is adaptive.

3.3 Overall Symbiotic System

The elastic audio encoder (MPEG-1 layer 3 encoder) now runs as the normal application. The symbiosis throttling system runs as two T-wares. The first one is invoked upon a loss-event. It estimates the parameters required to execute the model by probing iTCP as needed and finally it provides the application (elastic encoder) the rate parameters, as required to operate in symbiosis. The other T-ware is invoked when the system returns to normal state. Figures 3a and 3b show the pseudo code for the signal handler T-ware and recovery handler T-ware. No chance is needed in receiver side.

4 Experiment and Performance Analysis

We used a real implementation of iTCP and the MPEG layer 3 Symbiotic Encoder. The results presented were obtained from its live performance experiment conducted over the real audio delivery sessions over the Internet.

4.1 Experiment Setup

The experiment required running the symbiotic encoder equipped with iTCP transport protocol and player on remote host. The symbiotic encoder runs on FreeBSD system. This experiment describes the performance of an ISO/IEC 11172-3 [3] audio stream encoded with base frame rate of 128kbps.

Figure 4 illustrates deployment setup. The audio encoder runs on an iTCP machine which feeds audio stream to the player. To create some forced congestion in the experiment, we interrupt the connection between the player and the encoder for 3 seconds. The interrupt creates congestion bursts. A three-second burst usually triggers 1 to 2 retransmit timer out events depending on the player's location. We ran the player on Linux/Windows platform. We repeated the experiment for two audio sound qualities to a listening ear: high quality music (HighQmusic) and speech mixed with music (SpeechMusic). The player and encoder units were enhanced to collect detail frame arrival and delivery measurements.

4.2 Impact on Audio Frame Delay

For a detailed comparison we performed several sets of experiment. These are: iEXP, iOFF and Classic modes. In the iEXP, (exponential backoff) mode, we used a predetermined *rate retraction ratio* (ρ = 0.50) and multiplied it with current bit rate to calculate the frugal state bit rate, we also used a predefined recovery time $t_{recovery}$ of 4.0 seconds

As a base case, we also repeated the experiment with the same congestion schedule in classical mode, where the interactive and symbiotic rate adaptation features were turned-off and the entire system ran in classical TCP mode.

We also repeated the experiments in another mode called iOFF, for overhead estimation. The mode is similar to classic TCP. No symbiosis is performed. But the event subscription mechanism remains active. This will be explained later.

In all the iTCP enabled runs (iEXP and iOFF), the encoder subscribes with iTCP for the retransmission timer out event.

In the experiment, we took frame-wise detail event trace of the first 800 up to 1200 frames of the audio at both sending and receiving ends. For a given discard threshold time in the receiving end, we also traced which frame was successfully received or not at the receiving player. As explained earlier, we traced one transport aware cases iEXP and two transport unaware cases (iOFF and Classic)

Now, we show the dramatic impact of iTCP's interactivity-based symbiosis. In figure 5, we plot the delay experienced by the audio frames in terms of frame arrival time at the player for the three modes mentioned above. In addition, we also show the ideal expected frame delivery time-Expected in the figure-based on linear generation rate. As can be seen iTCP, (iEXP) outperformed classical TCP; after the congestion burst, the unaware cases (Classic and iOFF) fell behind. The delay built up and it could hardly recover. This is evident by the step jumps in the delay line. The TCP aware cases also suffered some step buildup, but it was much smaller and it could recover after few seconds due to the rate retraction.

4.3 Impact of Jitter on Audio

Assuming during congestion, packet $j, j = 0, 1, 2 \ldots n$ arrives at the receiver node at time tj. If the expected arrival time of packet j is ej under a perfect network condition, we define the referential jitter *refJitter(j)*, experienced by packet j, j to be the difference between tj and ej. Where $refjitter(j) = tj - ej$. The value of ej can be calculated by applying the best-case scenario of the flow assuming perfect throughput of the end -to-end network path and zero packet loss.

The *refjitter(j)* is negative when a packet arrives early and positive when late. Packets that arrive early can be buffered and played by player when its turn comes for

playing while the player will pause and wait for packet that arrive late.

Figure 6 shows the corresponding reference jitter experienced by the audio samples. We took the difference between the expected ideal arrival time and the actual arrival time for each frame.

Observation of the plots indicated the step jumps in the iEXP runs were generally smaller than those in TCP-classic runs since the frames that faced congestion suffered less delay. As shown the TCP-interactive drastically reduced the jittery behavior.

4.4 Symbiotic Rate Control

Figure 7 depicts the symbiotic frame rate encoding that occurred due to the joint rate specification at the rate control logic at the symbiosis unit and in the encoder for each frame. In each plot of figure 7, we see the target bit rate, the retraction ratio as specified by the symbiosis controller and the resulting outgoing actual frame rate generated by the encoder. The timer out events reported by iTCP resulted in the symbiosis unit to modify the rate according to the exponential backoff symbiotic rule.Table 1, presents the frame delay and acceptance ratio comparison. The table shows the performance for three choices of delay tolerance, d = 2, 4, and 6 seconds. For each value of d we traced the three running modes (iEXP, iOFF and Classic), recorded the average delay in seconds that each frame has experienced and the frame acceptance ratio at the receiving player nodes. It can be clearly noticed that iTCP/aware (iEXP) modes achieved low delays and high acceptance percentages while the unaware/classic modes suffered from higher delays and lower acceptance percentages. Clearly iTCP's T-ware mechanism allowed the application to use sophisticated techniques to control the temporal qualities.

Table 2 provides the overall stream compression due to symbiotic adaptation for the entire stream (iEXP cases), as compared to the normal non-symbiotic cases (iOFF and Classic cases) thereby reducing the overall delivered bits about 80-90%. The listening ears barely notice the drop in quality during frugal mode.

4.5 Observation at Application Level

An application receives and delivers uncompressed frames. The performance metric this end-system uses is the temporal and simultaneous masking effect between the transmitted and the reproduced uncompressed audio frames. The underlying MPEG-1 layer 3 transport protocol and the network layer TCP together provide the transport. The specific compression, windowing etc. and other detail mechanisms are external techniques to the end systems.

It is interesting to note that the size of the audio files generated by of iEXP compared to the iOFF and Classical TCP indicated a tremendous reduction. The iEXP mode achieved the tradeoff, though to the listening ear shows no difference in quality due to the masking effect. Fundamentally, what iTCP has offered is a qualitatively (as opposed to the quantitative improvements offered by any unaware solution) new empowering mechanism, where the catastrophic frame delay can be traded off for acceptable reduction in quality.

4.6 Interactivity Overhead

The dramatic advantage in application level performance came at a cost since the event tracking mechanism added some overhead. We were also curious to find out the overhead of the event mechanism. To track the overhead, we recorded the total data transmission time under the three conditions (iEXP, iOFF and Classic). The middle bar of figure 8 plots the transport time for the exponential backoff interactive mode where we activate both event delivery and symbiosis. To observe the overhead of the event service in the iOFF mode, we used the iTCP implementation; however, we stopped the symbiotic reduction so the transport layer handled the same amount of data. As expected the overall transmission time increased in all three cases. However, in the third column (Classic mode) run, we turned off the interactive service altogether and thus we saved its overhead and lost its benefit. As can be seen, the slight increase in the event delivery overhead was vastly offset by the application level technique.

5 Conclusions

Readers can listen to all audio used in the above experiments in [9]. In this paper, we have presented a case of audio rate symbiosis mechanism in line with current advances in TCP friendly systems. The approach exposes the overall advantage of network 'friendly' applications. However, it also departs significantly from the mainstream TCP friendly systems that have been suggested recently in two senses. Firstly, it does not add any new major component in the network software structure, it is simple and effective. The proposed interactivity does not alter any dynamics in the network side such as queuing.

Nevertheless, the approach adds slight complexity in the network layer. The augmentation of the notification feature increases the normal mode delay of TCP slightly; although it can be widely surpassed by the gain made at the application layer. There are profound and fundamental advantages of event based interactivity. The event based instant knowledge of the network

states enable sophisticated and efficient solutions to various network impairment problems-which cannot be otherwise realized via classical closed layer design.

The throttling model is novel. Yet, an improved (and perhaps optimizing) future model can easily be incorporated just by replacing the T-ware modules-without any change in network/transport layer.

6 References

[1] Catherine Boutremans, Jean-Yve L Boudec, "Adaptive Joint Playout Buffer and FEC adjustment for the Internet Telephony," 2003.

[2] Handley M., Floyd S., Pahdye J., and Widmer J., "TCP Friendly Rate Control (TFRC): Protocol

[3] ISO/IEC 11172-3. Information technology -- Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s -- Part 3:

[4] Johnny Matta, Christine Pepin, Khosrow Lashkari, Ravi Jain, "A source and channel rate adaptation algorithm for AMR in VoIP using E-model," June 2003.

[5] Javed I. Khan and Raid Y. Zaghal, Symbiotic Rate Adaptation for Time Sensitive Elastic Traffic with Interactive Transport, Journal of Computer Networks, Elsevier Science Volume 51, Issue 1, 17 January 2007, Pages 239-257.

[6] Javed I. Khan and Raid Y. Zaghal, Interactive Transparent Networking: Protocol Meta-modeling based on EFSM, Journal of Computer Communications, Elsevier Science, Volume 29, Issue 17, 8 November 2006, Pages 3536-3552

[7] Wenyu Jiang, Hening Schulzrinne, "VoIP: Comparision and optimatization of packet loss repair methods in VoIP perceived quality under bursty loss," May 2002

[8] Yi J. Liang, Eckehard G. Steinbach, Bernd Girod, "Voice over IP: Real-time voice communication over the Internet using packet path diversity," October 2001.

[9] Olufunke I. Olaleye and Javed I. Khan , Symbiotic Audio Communication on Interactive Transport, Technical Report 2007-02-01, Internetworking and Media Communications Research Laboratories, Kent State University, At http://medianet.kent.edu/ technicalreports.html, February, 2007.

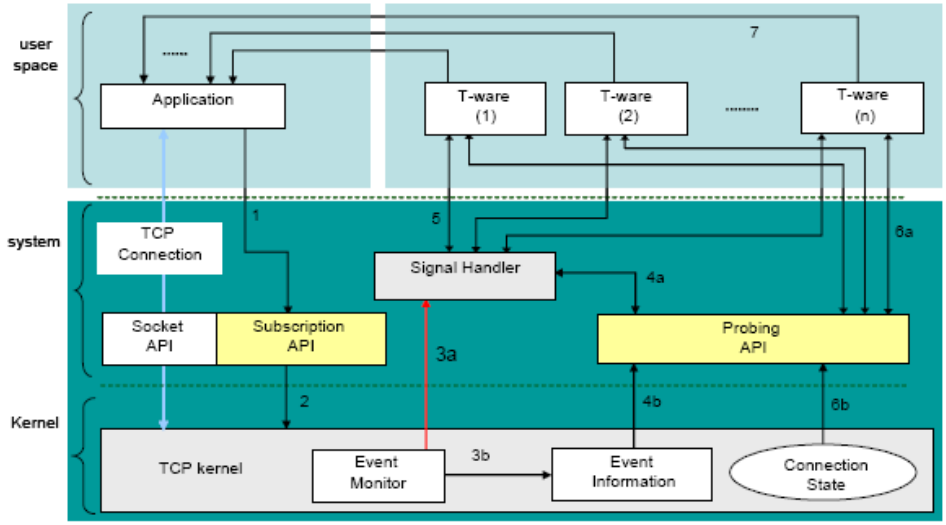

Figure 1. The iTCP extension and API.

Table 2 Percentage of Total Bits delivered to each node

	HighQmusic		Speech & Music	
	Target Bits	**Actual Bits**	**Target Bits**	**Actual Bits**
iEXP	0.900	0.900	0.812	0.805
iOFF	1.000	1.000	1.000	1.000
Classic	1.000	1.000	1.000	1.000

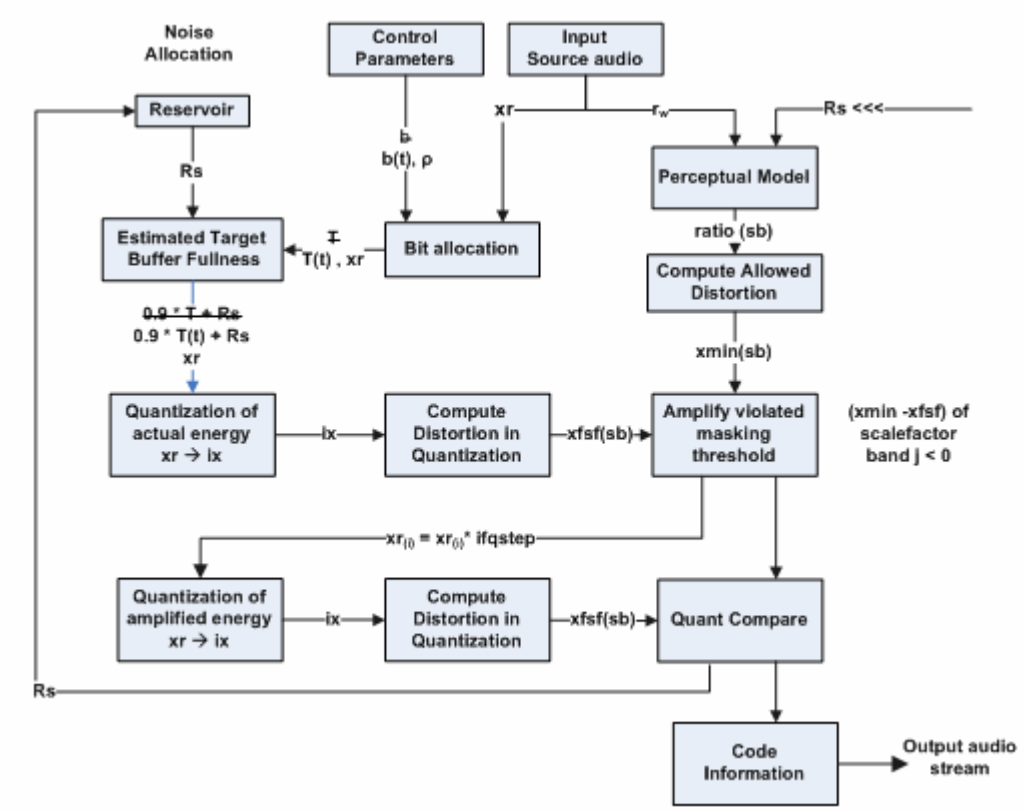

Figure 2. Rate Control of Encoder

```
1:   SignalHandler (signum) {
2:      struct evtSubInfo * handInfo;
3:      if ((signum = = SIGIO) && (EVENT) {
4:         s = GetSockid();
5:         ProbeEvtInfo(s, handInfo);
6:         StartRecover();
7:         if  (! (child = fork( ) ) ) {
8:            execl (handInfo → handler,
                                  retraction_ratio);
9:            exit ( 0 );
10:        } // end if
11:     } // end if
12:  } // end SignalHandler
```

(a)

```
1:   RecoveryHandler (signum){
2:      if  (signum ==  SIGALRM) {
3:         waitTimecount++;
4:         if ( waitTimecount && !rateOK &&
                        (waitTime > T_recovery)){
5:               ratefile = fopen("rate.par", "w");
6:               fwrite(originalRate, ratefile);
7:               rateOK = 1;
8:            }//end if
9:         }//end if
10:  }//end RecoveryHandler
```

(b)

Figure 3a & 3b. Pseudo code of the Signal Handler and the Recovery Handler T-wares

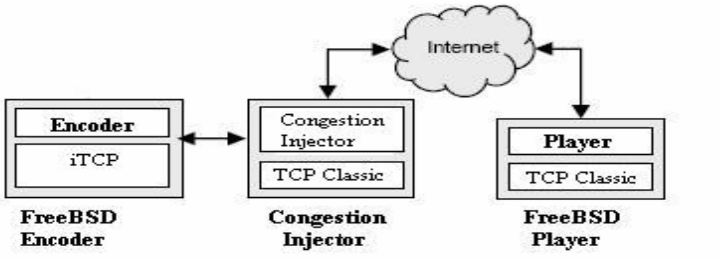

Figure 4. Experiment setup

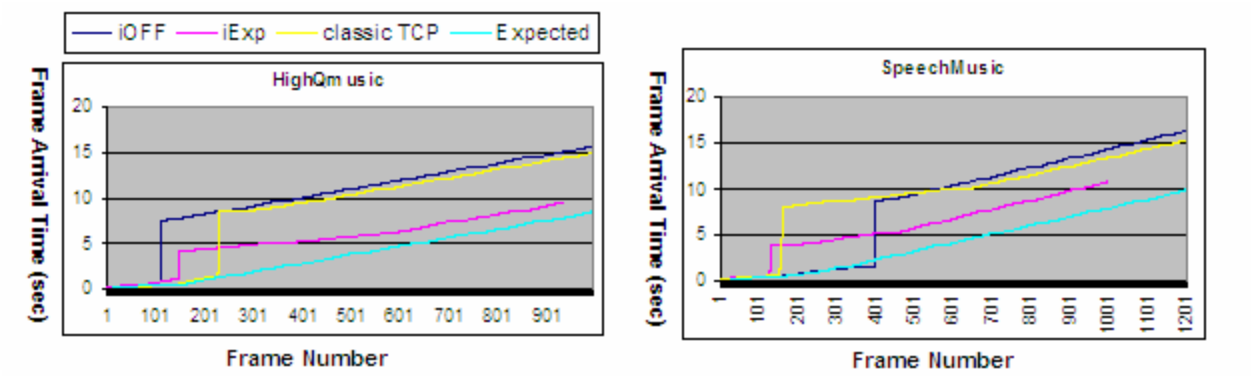

Figure 5. Frame arrival delay on the two audio qualities

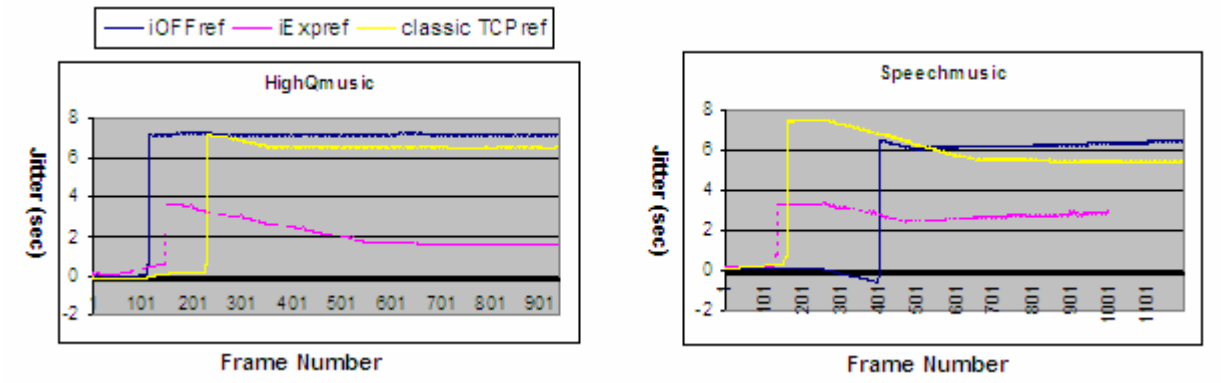

Figure 6. Referential Jitter on the two audio qualities

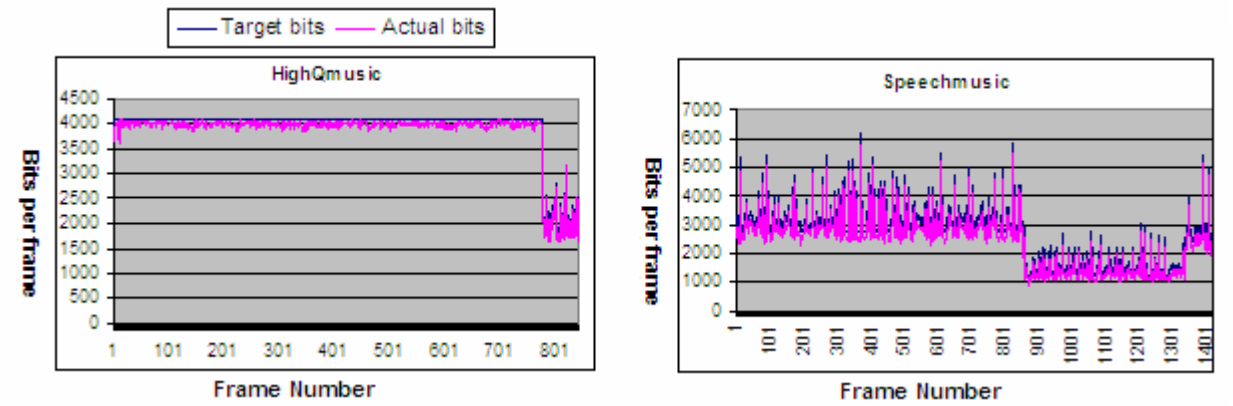

Figure 7. Symbiotic Rate Reduction on the two audio qualities

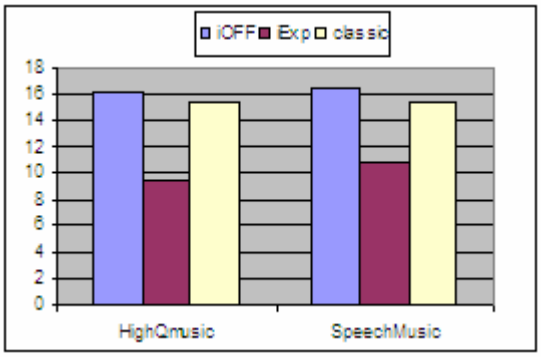

Figure 8. Overhead of the interactivity service

Table 1. Percentage of total bits delivered for each mode

mode		HighQmusic		Speechmusic	
		Average delay	Accept ratio	Average delay	Accept ratio
d =2	iEXP	-1.091	0.869	-1.554	0.922
	iOFF	4.451	0.484	2.211	0.598
	Classic	3.167	0.539	3.274	0.662
d =4	iEXP	-3.091	0.899	-3.553	0.929
	iOFF	2.451	0.417	0.211	0.656
	Classic	1.167	0.609	1.274	0.606
d =6	iEXP	-5.091	0.917	-5.55	0.933
	iOFF	0.451	0.609	-1.789	0.698
	Classic	-0.833	0.661	-0.726	0.652