Low delay MP3 Internet Telephony

RICHARD AKESTER
Department of Computer Science
University College London
Gower Street, London, WC1E 6BT.
United Kingdom

Abstract: - The use of MP3 as a compression codec for Internet Telephony has become feasible in view of the latest developments in PC hardware and broadband Internet capacity. In this context, this paper examines the implementation details behind such an Internet telephone, measuring delay in the end hosts. A new approach to detecting and correcting audio skew is proposed and implemented, focusing on the accuracy of measurements and on the algorithm's effect on the audio experience of the listener. The algorithms presented are shown to successfully remove audio skew, thus reducing delay and loss and hence improving audio quality. The feasibility of MP3 as a codec for Internet telephony is shown to be feasible and to work well within the required delay bounds.

Key-words: - MP3, Internet Telephony, VoIP, Audio Skew, Delay.

1 Introduction

Internet Telephony (or Voice over Internet Protocol – VoIP) is gradually gaining acceptance, especially with the business community. Based on an In-Stat/MDR survey of its Technology Adoption panel, by the end of 2002 roughly 2% of US firms were using some form of IP Telephony. The same research group forecasts growth to 19% of US firms by 2007 [12].

This adoption is often based on integration between tradition telephone networks and IP networks through gateways, enabling users to make calls from or to a regular telephone, rather than a PC. Since these traditional telephone networks use "toll-quality" 64kbps connections, there seems no need to use higher quality audio quality over the Internet segment of a call connection. Indeed, when VoIP was introduced most users had very limited Internet bandwidth and very high compression audio codecs were developed to enable such users to make Internet telephone calls.

Even today when broadband connections are more common, and if both sender and receiver in a telephone call have a 512kbps broadband connection, the software available still attempts to compress the audio data as much as possible, providing a low quality service to the user.

Telephone quality audio is sampled at 8000 samples/second capturing a frequency range of 4kHz. The precision of each sample is 8 bit. With these sampling parameters, any compression codec is limited to these frequency and noise bounds.

PCs have been capable of CD quality audio capture for a long time now. This provides 44100 samples/second at 16-bit precision, and is typically used to record music. The compression format MP3 (MPEG-1 Audio Layer 3) [1] is the most common compression format for these recording parameters, but usually produces an output bandwidth in the region 128kbps. Recently, PCs have reached a speed where they can encode MP3 audio in real-time, and (more importantly for VoIP) can encode and decode two streams simultaneously in real-time.

Even though 128kbps is a high bandwidth (in telephony terms), it is feasible to carry a full-duplex voice conversation compressed with MP3 over a broadband connection, and still have capacity to spare. The result is rich sound, capturing a wide frequency range at high precision.

The problem with MP3 VoIP is the same as with standard VoIP services. The Internet is a best-effort network that does not guarantee delay, jitter (delay variation between packets) or reliability. VoIP has limits on the amount of delay and loss it can tolerate for an interactive telephone call.

Packet loss can reduce audio reception quality, but repair techniques can compensate. With 20ms of speech samples per frame, waveform substitution can help to make speech intelligible even with loss rates up to 20% [3].

For real-time audio applications to maintain interactivity, the round trip delay should not exceed 400ms [2]. Adaptive audio applications compensate for network jitter (delay variance) by buffering, the size of the buffer depending on the current level of jitter. For interactivity the size of the buffer may be limited. If the worst-case jitter is greater than the maximum size of the buffer some breaks in the media stream will be expected. Combining the average delay and delay variance, and comparing this with the interactivity bound can gain an indication as to whether the network performance will be suitable for realtime traffic and an adaptive application.

With this delay bound in mind, it is important to assess the MP3 codec delay to determine the feasibility of using this codec for telephony. The MP3 packetization delay for 1152 samples (a standard MP3 audio frame) at 44100 samples/second is 26.122ms. Therefore, the encoding delay must be less than 26ms in order to be realtime (the LAME [9] encoder used can achieve this on a PII 266MHz at highest quality encoding). For decoding, the MPG123 [10] decoder is used. Decoding is easier than encoding, and therefore faster. Furthermore the decoded data can be played immediately. Therefore, there is no reason why the choice of MP3 as the codec should prevent the telephony interactive delay bounds to be exceeded.

Another lesser-known but potentially more severe cause of delay in Internet Telephony is from the end-hosts in the form of audio clock skew [4,5]. Audio skew is an effect caused by lack of synchronization between the sender and receiver audio clocks. It is common for clocks to have a discrepancy of a few percent (equivalent to a variation of approximately one second every minute).

This is different from system clock skew [6,7] since the ultimate timing source and destination is the audio producer at the sender and the audio consumer at the receiver. For example, if, under ideal conditions, the sender is sending packets with 1152 samples of data per packet, then 1152 samples on the local audio clock (not the system clock) should elapse between packet arrival times.

Even a small clock discrepancy will, over time, result in under-run or over-run of a receiver's buffers, resulting in either the erosion of the initial buffer followed by audible discontinuities caused by network jitter, or a gradual increase in the receiver's delay until a buffer limit is reached at which point data is periodically discarded.

To *detect* audio skew, the receiver has to measure the mean packet arrival time, according to its own audio clock. To *correct* audio skew, the receiver must adjust its audio clock to match the estimate of the sender's clock.

Previous work in the field [8] has concluded that it is not possible to use MP3 as a codec for VoIP due to high delay. However, it is likely that these delays are due to a large audio output buffer (a consequence of audio skew) and are not due to using the MP3 codec.

Hence, overcoming audio skew is key to enabling MP3 as a telephony codec. The remainder of this paper examines how audio clock skew can be measured and corrected, and presents the results and conclusions of experiments to measure the delay during an MP3 telephony session.

2 Audio Skew Detection and Correction

2.1 Audio Clock Accuracy

Reading the current audio clock value from the hardware is not trivial. Although the internal clock is usually rated in MHz the best granularity that is available is to the nearest sample. As will be seen below, even this accuracy is usually not available even with the latest hardware.

Communication between the audio card and the main processor is achieved by using an interrupt. The audio card sends interrupts to the processor periodically whenever a certain number of samples have been processed. The number of samples is again specified in advance.

It is possible to use the interrupts to measure the audio clock, but the granularity is limited to the maximum

frequency that interrupts can be handled by the system. The maximum recommended interrupt rate is every 1ms. At this rate the pressure of servicing interrupts starts to cause a noticeable system load. At a clock rate of 48kHz, 1ms represents 48 clock ticks. Measuring to the nearest millisecond is not very accurate, given that granularity 48 times better can potentially be achieved. It is also inadvisable to cause so much system load from the interrupts.

A better method of discovering the current audio clock value would be to query the card when a value is required. This would not be so frequent (once per packet arrival) and would give more accurate information (potentially a clock value to the nearest sample).

A way to query the audio card for the current clock values is by using either DMA (Direct Memory Access) for ISA (Industry Standard Architecture) or "Bus Mastering" for PCI (Peripheral Component Interconnect) architectures. DMA is general hardware (not on the audio card) that controls transfers from main system memory to the audio card without the main processor's intervention. The DMA can be stopped, the current buffer position obtained, and then restarted again. The price to be paid is that the DMA has to be stopped before the value can be read. However, most audio cards cache a few samples and can tolerate a small interruption with no discontinuity in the output. It is possible that too many queries to the DMA would cause a noticeable effect as the audio card is starved of access to the main memory.

A better solution arises from newer PCI cards, which use a technique called "Bus Mastering" to access main memory. In effect, the DMA hardware is built into the audio card itself, with the advantage that the Bus Master position can be obtained from PCI query functions without stopping memory transfers. Thus, there is no danger of too many requests causing audio interruptions.

The accuracy of the buffer position obtained from a PCI audio card is often only to the current block of samples. A typical example is a 128 sample block size, which (at a 48kHz sample rate) gives a 2.7ms clock granularity. Deriving the current sample position from the current block position must be done with care to minimize error.

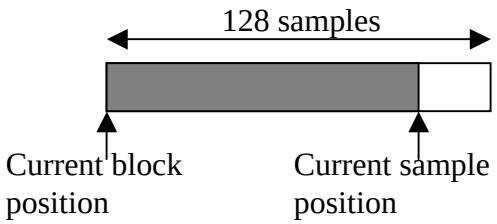


Figure 1: deviation between block and sample position

Figure 1 shows that a better estimate of the current sample position can be obtained by adding 64 (in this case) onto the current block position. This halves the error and hence doubles the accuracy.

2.2 Measuring The Average Packet Arrival Rate

Each time a packet arrives the receiver can check the current local audio clock, and can compare the current clock value with the previous clock value to obtain the amount of time that has elapsed between packet arrivals.

This value is not very reliable, however, since it is affected by network jitter. To remove the effects of jitter, the average of many inter-arrival times must be calculated.

A simple approach of keeping a record of the sum and the number of items and calculating the mean average does not scale since the sum quickly becomes big causing storage overflow and calculation problems. To avoid this a running average can be kept.

Eg. If a_n is inter-arrival time n, and the average of the first two inter-arrival times, $av_2 = (a_1 + a_2) / 2$

Then the average of three inter-arrival times, av_3

$= (a_1 + a_2 + a_3) / 3$

$= (av_2 * 2 + a_3) / 3$

$= 2/3 * av_2 + 1/3 * a_3$

Generalizing for n values, $av_n = (n-1)/n * av_{n-1} + 1/n * a_n$

This can be re-written as: $av_n = av_{n-1} + (a_n – av_{n-1}) / n$

This calculation does not require the sum to be kept, and scales until the number of inter-arrival times, n, becomes too large to store. At this point, the average changes so little that new values have virtually no impact on the average, so when the number of inter-arrival times becomes sufficiently large, it is safe to assume that the average has stabilized.

High jitter slows the convergence to the average value, so an initial buffer is necessary to absorb any underflows during the "training" period (until the average stabilizes).

2.3 Correcting the Audio Skew

To correct the audio skew, it is necessary to change the rate of audio consumption to match the rate at which packets are arriving. This involves dynamically adapting the audio sample rate.

It is not generally possible to change the clock rate on the fly while it is running, since the audio hardware requires recalibration.

It is possible, however, to use software to convert data from one sample-rate to another. Such software uses interpolation (preferably of order at least quadratic) to convert from one sample base to another. A sample-rate converter does not require both the input sample rate and the output sample rate. Rather, the ratio of the two is enough for the algorithm to perform the conversion. This ratio (between the input and output rates) is valuable because it is the same as the ratio between the sender clock and the receiver clock and additionally it is independent of unit.

It is important to calculate the output from the sample rate converter carefully. For example, when converting from 44.1kHz to 48kHz and feeding 1152 samples into the converter an output of 1253 (1152 x 48 / 44.1) is not acceptable. The result would be a steadily decreasing queue length. The converter must keep state and output either 1253 or 1254 samples in order to maintain 48000 samples per second output.

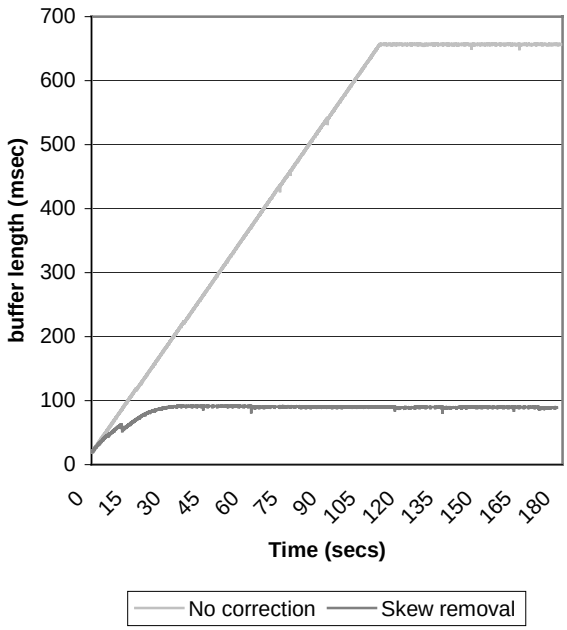
The sender clock is contained in the Realtime Transport Protocol (RTP) [11] "timestamp" field. When the RTP payload type is "MPEG Audio", the timestamp in an RTP packet gives the current sender clock time (with a 90kHz accuracy). The difference between these timestamps gives the inter-departure time of the packets, according to the sender clock.

The ratio of the inter-arrival time and the average inter-departure time can now be fed to the sample rate converter, and the resulting output can be sent to the audio device.

To test the algorithm, an experiment is created over a part of the Internet (within a national network) from an academic network onto a commercial network, with a broadband connection at 512kbps downlink and 256kbps uplink speed and an 802.11g 54Mbps wireless LAN as the last hop. An MP3 transmitter sent voice packets for 3 minutes, and a receiver played the packets while displaying buffer length statistics. The link is full duplex, with a sender and receiver at both ends. The receiver at one end showed overflow, while the receiver at the other end showed underflow. This is to be expected, since one audio clock will be slower than the other. The overflowing receiver is used to collect statistics, as displayed in figure 2.

The top line in the figure shows the default case (no attempt by the application to control skew). The buffer length increases linearly for about 2 minutes before overflowing (with a delay of 2/3sec), at which point data is lost periodically. The bottom line shows the result of trying to match the input and output speeds. There is an initial increase in the buffer length while the algorithm adapts to the input rate, after which the delay remains constant at about 100ms.

Figure 2: Effect of skew removal on buffer length



2.4 Converging the Buffer Length to the Worst-Case Jitter

Figure 2 demonstrates that the skew correction is able to match the incoming packet rate but the buffer length becomes unnecessarily high (many times higher than the worst-case jitter) as the sample-rate stabilizes. For telephony where delay is limited, this is a problem. The buffer could be corrected by throwing away packets, but this will reduce quality substantially. A better way is to over-correct the sample-rate to converge the buffer length onto the maximum jitter value (making sure that the sample rate is adjusted slowly enough to make the change inaudible).

The difference between the current buffer length and the required buffer length ("max_jitter") is an important quantity when correcting the buffer length. The amount of history used when finding the max_jitter has to be chosen so that old jitter events are not "forgotten". A value of 20 seconds is chosen in these experiments. The amount of correction is thus:

```
correction = buffer_length – max_jitter;
```

This correction value can be positive (the buffer is too long) or negative (the buffer is too short), and can potentially be of large magnitude (+ or – the maximum buffer size).

In fact, the "current buffer length" is not easy to obtain when network jitter is causing the buffer length to oscillate. Since a variation of max_jitter would occur in the future it is prudent to keep track of the minimum buffer length (over recent history) and to try to bring this in line with the current max_jitter. The amount of history used in these experiments is chosen to be 2 seconds worth. This value is chosen as, in this case, it is preferable to "forget" quite quickly about old values.

Having obtained a measure of how much correction is required, a method is needed to translate this into a small correction to the current sample-rate converter that will not be audible.

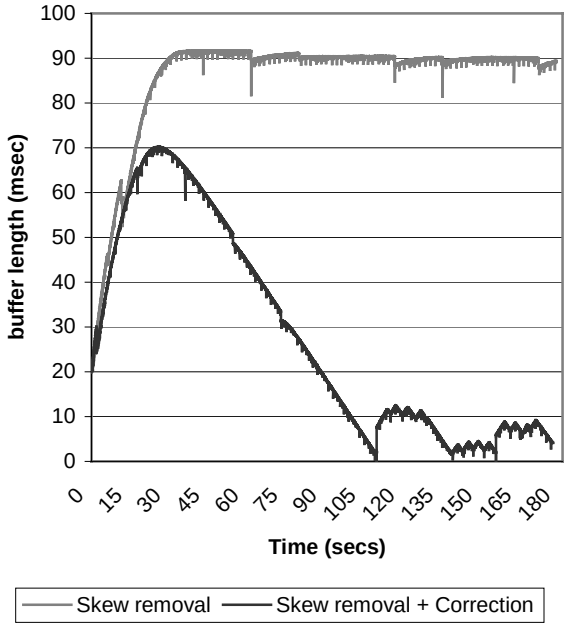
By experimentation it is found that only variations of 1 sample unit are possible without being audible. Corrections are thus calculated as follows:

```
If(min_buffer_length > max_jitter)
correction = +1; else correction = -1;
```

If the number of samples to be consumed is "num_samples" then the buffer length can be corrected by feeding num_samples into a sample rate converter and requesting (num_samples – correction) samples to be returned.

Figure 3 shows the effect of the buffer correction algorithm using the same experimental setup as used before. The top line is the result of the detection algorithm, as shown in figure 2. The bottom line shows the effect as the correction algorithm attempts to adjust the buffer length to the current max_jitter level. The buffer length initially rises as before but starts to reverse and tends towards a much lower delay level (~10ms) in line with the current max_jitter. As max_jitter varies, so the buffer length converges onto it, from ~10ms max_jitter drops to ~3ms, and then rises to ~8ms.

Figure 3: Effect of skew correction on buffer length



3. Conclusions

The results show that the proposed detection and correction algorithms are having the desired effect, improving the quality (reducing delay) of streamed MP3 audio.

The implementation of the MP3 Internet telephone was actually used to conduct telephone conversations and

the subjective impression of the service was that it was very usable, delay being hardly noticeable. This was an important achievement, given that earlier work had stated that it was not feasible to use MP3 for Internet telephony.

When the Internet was congested (less than 128kbps was available), the receiver initially had a buffer underrun, but quickly reduced the sample-rate resulting in slightly lower pitch audio but without discontinuities. The bottleneck router in the Internet became a point of delay, however, and as the router buffers overflowed, a source of periodic loss. The service was still usable, but it would have been preferable for the sender to compress MP3 with a lower bandwidth in these conditions.

Future work involves testing the implementation over more diverse Internet connections and reviewing the parameters used in the program. Specifically the amount of history to take into account when calculating the max_jitter and min_buffer_lengths. Are the current values optimal under all conditions? In addition, the addition or subtraction of 1 to the local_audio_clock_delta to adapt the buffer level to the current max_jitter may not be universally applicable. A fraction of min_buffer_length – max_jitter may produce better results.

References:

[1] ISO/IEC, JTC1/SC29/WG11 MPEG, “Information Technology — Coding of Moving Pictures and Associated Audio for Digital Storage Media at up about 1.5 Mbit/s — Part 3: Audio”, IS11172-3, 1992 (“MPEG-1”).

[2] Brady, P. T. “Effects of Tranmission Delay on Conversational Behaviour on Echo-Free Telephone Circuits.” Bell System Technical Journal, pp115-134, January 1971

[3] Hardman, V., Sasse, M.A., Handley, M., Watson, A., “Reliable Audio for Use over the Internet”, Proc. INET’95.

[4] Orion Hodson, Colin Perkins, and Vicky Hardman, “Skew Detection and Compensation for Internet Audio Applications.” Proceedings of the IEEE International Conference on Multimedia and Expo, New York, July 2000.

[5] Richard Akester, and Stephen Hailes “A New Audio Skew Detection and Correction Algorithm.” Proceedings of the IEEE International Conference on Multimedia and Expo, Lausanne, September 2002.

[6] Sue Moon, Paul Skelly, and Don Towsley, “Estimation and removal of clock skew from network delay measurements.” Proceedings of the Conference on Computer Communica-tions (IEEE INFOCOM), New York, March 1999.

[7] Vern Paxson, “On calibrating measurements of packet transit times.” Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems, pages 11–21, Madison, Wisconsin, June 1998.

[8] Sooyeon Kim, JeongKeun Lee, Tae Wan You, Kyoungae Kim, and Yanghee Choi, "Hat: A High-quality Audio Conferencing Tool using mp3 Codec," INET 2002, Washington, DC, USA, June 2002

[9] The LAME project: http://www.mp3dev.org/mp3

[10] Mpg123 homepage: http://www.mpg123.org

[11] RTP: A transport protocol for real-time applications. RFC 1889.

[12]http://www.instat.com/infoalert.asp?Volname=Volume%20%23%2025