Streaming of Multimedia with Reduced Start-Up Delay

V. Weerackody, G. Schuller, and H.-L. Lou
Bell Laboratories, Lucent Technologies, 600 Mountain Avenue, Murray Hill, NJ 07974.
Phone: 908 582-3773. Fax: 908 582-7308. email: {vijitha,schuller,lou}@research.bell-labs.com

Abstract— **High quality multimedia streaming applications over the Internet require very low packet loss rates. The Internet is characterized by long bursts of packet losses and delays. A large receive buffer can be used to mitigate the effects of packet losses and delays. However, a large receive buffer introduces a large delay in the playback of a packet. This large delay could be annoying at the start of a program or during switch over to another channel in a multi-channel broadcast. In this paper we introduce a separate low-delay tuning stream to address this start-up problem. In the steady state, this tuning stream is synchronized appropriately with the high-delay steady state stream to give an enhanced composite signal.**

I. Introduction

Applications that employ multimedia streaming over the Internet are growing rapidly. Internet distribution of radio and television programs, multicasting news events, and delivery of audio and video clips are some examples of these multimedia streaming applications. When network traffic is high the Internet gives rise to large and variable packet delays and packet losses occur in large bursts. It is reported in [1] that the burst length of packets can be several seconds long. High quality multimedia applications typically require very low packet loss rates, for example 10^{-4} or better of random packet losses. At the receiver, it is necessary to buffer the received packets over a reasonably long time to overcome the large variable delays of these received packets. Also, packet losses may be reduced by forward error correction coding across the packets or using retransmission techniques. Buffering of received packets as well as packet error correction techniques introduce delays in the recovered packets. However, since streaming applications are used in mostly non-interactive environments the end-to-end delay is not usually crucial in such cases.

A large buffer is desirable at the receiver because it helps to reduce the packet losses due to large transmission delays and also facilitates better error correction across packets. However, buffering of a large number of packets is not attractive during the initial start-up period of the program or during the switch-over from one program to another in a multi-program broadcast system. This is because in order to maintain a steady average receive buffer size, playback of packets can be done only once the receive buffer is reasonably filled. A start-up delay of a second or more is annoying in a high-quality multimedia transmission system, especially, in the case when a switch from one program to another takes place. In this paper we address this start-up problem by introducing an additional low-delay stream, or a tuning stream, that is useful during the start-up stage. Because of the tuning stream the regular steady state stream can be subjected to a large delay at the receiver and thereby reducing the packet loss rate for the steady state stream. The tuning stream is a low bit rate source layer obtained from a scalable source coder [2] or a multiple description source coder [3] and in the steady state may be combined with the steady state stream to enhance the quality of the steady state. In [4] the startup problem arising from large interleaver delays in fading situations was addressed. In [5] delays caused by the playout buffer are dealt with, but not issues related with startup delay.

II. Start-Up Problem

Consider a source transmitting packets $P(n)$ at periodic intervals nT, where n is an integer and T is the duration between successive packets. For some audio coders T is typically of the order of 30 ms. The Internet gives rise to variable packet delays and Figure 1(a) shows the packet arrival times at the receive buffer and Figure 1(b) shows the packets at the playback end. The receive buffer accumulates the packets over a certain period of time and sends them to the playback end for source decoding. Note that, for convenience, we are denoting the time axis in terms of discrete steps of T. Also, assume a broadcast type transmission where the packets are sent even before the start time n_s. Referring to Figure 1(a) suppose the program is switched on at n_s, then, buffering of packets at the receiver begins with packet $P(n_s - n_d)$, which is the packet sent at $(n_s - n_d)$ with n_d being the transmission delay. Note that although the packets are sent at regular intervals of T the arrival times are variable and some packets may get lost. Also, since the packets may take different routes, they may arrive out of sequence. For illustration purposes let us assume that $P(n_s - n_d)$ is the packet with the lowest sequence number to arrive for $n \geq n_s$. In order to reduce the effects of packet delays and losses the receiver accumulates the packets over a large receive buffer. Figure 1(b) depicts the packets delivered to the playback side with the first received packet, $P(n_s - n_d)$, delivered at $(n_s + n_b)$, where n_b is the delay at the receive buffer. Since the packets are sent at regular intervals of T, the playback end delivers packets sequentially at intervals of T.

For this case we see that n_b is the tuning delay and $(n_b + n_d)$ is the end-to-end delay given by the transmission delay and the buffering delay at the receiver. For streaming applications the end-to-end delay is not critical; however, the tuning delay is an important parameter as discussed earlier. In the case of packet losses, for re-transmission schemes to be effective, n_b should be of the order of several seconds. Since the end-to-end delay is $(n_b + n_d)$ those packets whose transmission delay is more than the above will be lost. In the next section we introduce a separate stream to address the tuning delay problem.

III. Tuning Stream

Figure 2 depicts the packet arrivals and the playback end of the buffer for the tuning stream represented by packets $Q(n)$. At switch on at n_s the receiver starts buffering $Q(n_s - n_{d'})$, where $n_{d'}$ is the transmission delay of $Q(n_s - n_{d'})$. This packet is sent to the playback buffer at $(n_s + n_{b'})$ where $n_{b'}$ is now the tuning delay. Since $n_{b'} < n_b$ the packet losses in the tuning stream are more than the steady state stream, however, playback can commence at an earlier time $(n_s + n_{b'})$.

At a later time, $(n_s + n_b)$, the steady state packets $P(n_s - n_d)$ will be available at the playback end together with $Q(n_s - n_{d'} + n_b - n_{b'})$ from the tuning stream. For continuity of the source signal both these should correspond to the same source signal segment. Suppose $S(n)$ is the source signal corresponding to $P(n)$, then it is clear that $Q(n)$ corresponds to $S(n - \delta)$ with $\delta = (n_d - n_{d'} + n_b - n_{b'})$. Since $P(n)$ and $Q(n)$ are sent at the same time and we may assume $n_d \approx n_{d'}$. It is seen that the tuning stream represents a signal that is a δ-delayed version of the signal that represents the steady state stream. At $(n_s + n_b)$, depending upon the source coders, it is possible to combine the tuning and the steady state streams to enhance the quality of the decoded signal.

For bandwidth efficiency the tuning stream is usually a lower rate signal. This may be derived from a multiple description source coder [3] where the tuning and the steady state streams have two different descriptors of the signal which can reproduce the signal individually or when combined adds quality to the composite signal. For example for an Internet audio broadcast system we may use a multiple description coder to obtain 16 kb/s for the tuning stream and 48 kb/s for the steady state stream with the buffer delays $n_{b'}$=500 ms and n_b=2 s. In this case, at the transmitter, the tuning signal will be delayed by 1.5 s with respect to the steady state signal. At the receiver, after an initial tuning delay of 500 ms the 16 kb/s audio signal is played back. The 48 kb/s steady state signal will be available 2 s after switch on and at this point the 48 kb/s may be combined with the 16 kb/s tuning signal. Note that the packet losses in the tuning stream could be high, therefore, the 48 kb/s steady state signal should be able to reproduce the original signal with sufficient quality.

In the more general case consider K separate streams of the source signal at the n^{th} instant, $S(n)$, represented by $P_1(n), P_2(n), ..., P_K(n)$, with rates $R_1, R_2, ..., R_K$. It is desirable to have $R_1 < R_2 < ... < R_K$. Denote the tuning delay for $S_k(n)$ by

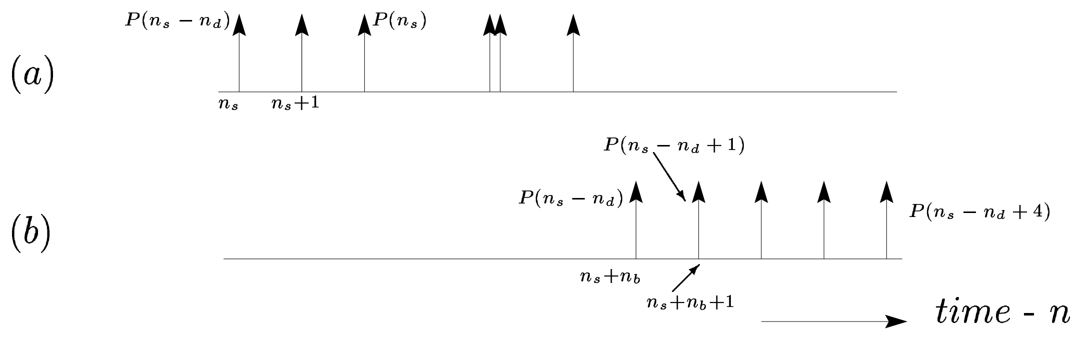

Fig. 1. (a) Arrival times for the packets $P(n)$ at the receive buffer. n_s - program switch on; n_d - transmission delay of $P(n_s - n_d)$. (b) Packets at the playback end of buffer. $(n_s + n_b)$ - start playback buffer.

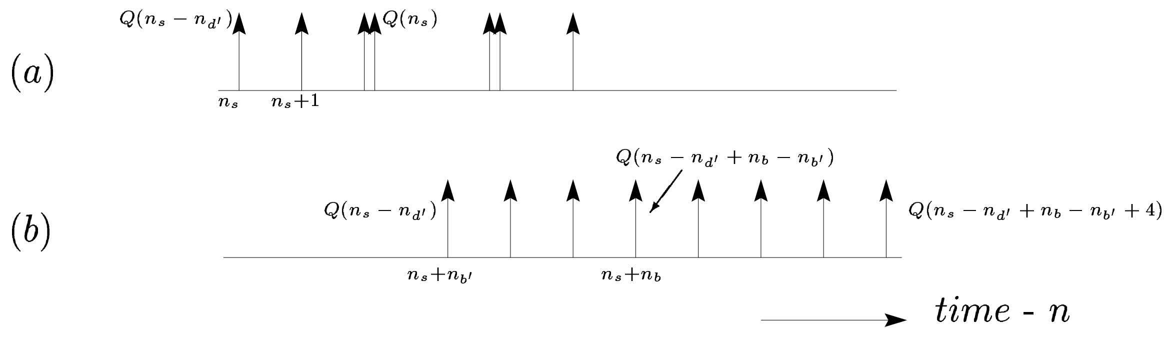

Fig. 2. (a) Arrival times for the packets $Q(n)$ at the receive buffer for the tuning stream. n_s - program switch on; $n_{d'}$ - transmission delay of $Q(n_s - n_{d'})$. (b) Packets at the playback end of buffer. $(n_s + n_{b'})$ - start playback for tuning channel.

n_{bk}, where $n_{b1} < n_{b2} < .... < n_{bK}$. It is desirable to encode the source so that the individual streams, $P_k(n), k = 1, 2, ..., K$, are independently decodable. In this case the receiver first plays back $P_1(n)$ and this is followed by $P_2(n), P_3(n), .., P_K(n)$, in a gradual manner. Since $R_k > R_{k-1}$ and $n_{bk} > n_{b(k-1)}$, during the transient stages the quality of the signal is gradually increased because of the higher source rate and the lower packet losses. This is also because more streams can be combined for increasing values of k. The transmitted packet at the n^{th} instant consists of $P_1(n-\delta_1), P_2(n-\delta_2), ..., P_K(n-\delta_K)$, where $\delta_1 = (n_{bK} - n_{b1}), \delta_2 = (n_{bK} - n_{b2}), \delta_3 = (n_{bK} - n_{b3}), ..., \delta_K = 0$. At n_s, the switch-on time, the receiver begins to accumulate the packet consisting of $P_k(n_s - n_d - \delta_k), k = 1, 2, ..., K$, where n_d represents the transmission delay in the Internet for this packet. At $(n_s + n_{b1})$, after a tuning delay of n_{b1}, $P_1(n_s - n_d - \delta_1)$ is ready for playback. The next stream, $P_2(n_s - n_d - \delta_2)$, will be available at $(n_s + n_{b2})$ and the corresponding value of P_1 at this time instant is $P_1(n_s - n_d - \delta_1 + (n_{b2} - n_{b1}))$. Using the above values for δ_1 and δ_2 we see that the two streams available at this instant at the playback end of the buffer are $P_1(n_s - n_d - n_{bK} + n_{b2})$ and $P_2(n_s - n_d - n_{bK} + n_{b2})$. Note that δ_1 and δ_2 are chosen so that the two streams are synchronized such that there is no relative delay between them, therefore, they may be combined to provide an enhanced signal. Similarly, at $(n_s + n_{bk})$, the k streams $P_1(n_s - n_d - n_{bK} + n_{bk}), P_2(n_s - n_d - n_{bK} + n_{bk}), ..., P_k(n_s - n_d - n_{bK} + n_{bk})$, are available at the playback buffer and they maybe combined to enhance the quality of the signal. Finally, the steady state signal is given at $(n_s + n_{bK})$. Note that for the k^{th} stream the end-to-end delay (including the transmitter delay of δ_k) is $(n_{bK} + n_d)$ and the transmission and buffering delay is $(n_{bk} + n_d)$. The packets from the k^{th} stream should arrive at the receive buffer no later than $(n_{bk} + n_d)$.

IV. Conclusion

In this letter we addressed the problem of large tuning delays associated with large receive buffers in streaming multimedia applications. The source signal is separated to several streams and each stream is sent with a different delay. These delays are determined by the buffering delay at the receiver. The stream with the lowest buffering delay is played back first and this is followed by the other streams so that the quality during the tuning process increases gradually.

References

[1] M. Borella, "Measurement and Interpretation of Internet Packet Loss," *Journal of Communications and Networking*, vol. 2, pp. 93–102, June 2000.

[2] J. Herre, E. Allamanche, K. Brandenburg, M. Dietz, B. Teichmann, B. Grill, A. Jin, T. Moriya, N. Iwakami, T. Norimatsu, M. Tsushima, and T. Ishikawa, "The integrated filterbank based scalable MPEG-4 audio coder," in *105th AES Convention*, (San Francisco, CA, Preprint 4810), Sept. 1998.

[3] V. A. Vaishampayan, "Design of Multiple Description Scalar Quantizers," *IEEE Trans. Inform. Theory*, vol. 39, pp. 821–834, May 1993.

[4] V. Weerackody, H.-L. Lou, and Z. Sayeed, "A code division multiplexing scheme for satellite digital audio broadcasting," *IEEE JSAC:Wireless Communications*, pp. pp. 1985–1998, Nov. 1999.

[5] C. J. Sreenan, J.-C. Chen, P. Agrawal, and B. Narendan, "Delay reduction techniques for playout buffering," *IEEE Transactions on Multimedia*, vol. 2, June 2000.