Scalable On-Demand Media Streaming with Packet Loss Recovery

Anirban Mahanti, Derek L. Eager, Mary K. Vernon, and David Sundaram-Stukel

***Abstract*— Previous scalable on-demand streaming protocols do not allow clients to recover from packet loss. This paper develops new protocols that (1) have a tunably short latency for the client to begin playing the media, (2) allow heterogeneous clients to recover lost packets without jitter as long as each client's *cumulative* loss rate is within a tunable threshold, and (3) assume a tunable upper bound on the transmission rate to each client that can be as small as a fraction (e.g., 25%) greater than the media play rate. Models are developed to compute the minimum required server bandwidth for a given loss rate and playback latency. The results of the models are used to develop the new protocols and assess their performance. The new protocols, Reliable Periodic Broadcast (RPB) and Reliable Bandwidth Skimming (RBS), are simple to implement and achieve nearly the best possible scalability and efficiency for a given set of client characteristics and desirable/feasible media quality. Furthermore, the results show that the new reliable protocols that transmit to each client at only twice the media play rate have similar performance to previous protocols that require clients to receive at many times the play rate.**

1 Introduction

An important problem for a number of existing and future Internet applications is that of delivering streaming media, on-demand and in a *scalable* and *reliable* manner, to potentially large numbers of concurrent clients that receive the data over lossy and possibly heterogeneous channels. This problem has been addressed effectively for bulk data downloads [29, 33, 5], such as large software updates. However, solutions that are both scalable and reliable do not currently exist for streaming media content, as would be required in applications such as video-on-demand.

The *digital fountain* [5] approach [29, 33, 5] is designed for downloading bulk data over channels that have significant packet loss, including IP multicast on the Internet, satellite transmission, and wireless transmission channels. The approach uses an *erasure code* to construct a stream of packets from a bulk data file, such that a receiver can reconstruct the file from any subset of the packets of size equal to or just slightly (e.g., 5%) greater than the number of packets in the source data. Thus, each client can begin listening to the (multicast) stream at a time of their own choosing, and simply keep listening until the required number of packets have been correctly received. The method is fully scalable because the required server transmission bandwidth is equal to a single transmission stream, independent of the number of clients actively acquiring the data. The method is efficient because (1) no feedback channels are required for clients to recover lost packets, (2) the amount of data each client needs to receive in order to obtain the full object is (nearly) minimal, and (3) with an efficient erasure code [5], the amount of processing time required for a client to reconstruct the original data is small.

The digital fountain described above has high client start-up latency when applied to on-demand streaming media, since in general a client is unable to reconstruct the portion of the object that is needed to begin playing the media until after receiving most or all of the data needed to reconstruct the entire object. Furthermore, other recent approaches for reliable *live* or *scheduled* broadcast delivery of streaming media [35, 13, 7, 21, 36, 23, 4, 20, 32, 9] do not address the problem of providing scalable *on-demand* delivery.

Recently proposed protocols for scalable on-demand media streaming, such as *periodic broadcast* protocols [34, 1, 17, 19, 14, 24, 16], *patching* [8, 18, 6, 15, 30], and *bandwidth skimming* [12], do not address the issue of providing reliable delivery over lossy channels. When applying scalable streaming methods in environments where packet loss is relatively rare, local error concealment (e.g., interpolating lost video frames from frames that are received correctly) may be adequate. However, in environments that have more frequent or bursty packet loss, local error concealment is inadequate for many applications [27].

It is not straightforward to extend previous scalable streaming protocols to include redundant data for recovering lost packets. In particular, patching, bandwidth skimming, and nearly all periodic broadcast protocols are designed such that each delivered media packet is needed (nearly) immediately for playback by at least one of the clients that receive the packet. In this case, the use of erasure codes or packet retransmissions for recovering lost packets is severely restricted. In a few previous periodic broadcast protocols each media segment is received entirely before it is needed for playback (e.g., [19, 16]). In this case, each segment could be encoded using an erasure code for packet loss recovery. However, the protocols do not provide an efficient method for recovering in the case where packet loss is higher than anticipated in the segment encoding and transmission rate. Furthermore, all previous protocols that deliver each segment entirely before it is needed for playback require the clients to receive *all* of the segments of

A . Mahanti and D. Eager are with the Dept. of Computer Science, University of Saskatchewan, Saskatoon, SK S7N 5A9, Canada (email: mahanti@cs.usask.ca; eager@cs.usask.ca).
M. Vernon and D. Sundaram-Stukel are with the Computer Sciences Dept., University of Wisconsin, Madison, WI 53706-1685, USA (email: vernon@cs.wisc.edu; sundaram@cs.wisc.edu).

the requested object simultaneously (at an aggregate rate equal to 5-10 times the media play rate[1] if low start-up delay is to be achieved). Such aggregate data rates may not be feasible in environments such as the Internet, since they may create unacceptably high packet loss rates for a given desired media quality and transmission path. A high aggregate data rate to the clients also implies a large client buffer requirement.

This paper develops new Reliable Periodic Broadcast (RPB) and Reliable Bandwidth Skimming (RBS) protocols for *reliable* on-demand streaming over lossy and possibly heterogeneous channels. The new protocols (1) have a tunable short latency for the client to begin playing the media, (2) allow each client to recover lost packets without jitter as long as the client's *cumulative* loss rate is within a tunable threshold, and (3) have a tunable upper bound on the transmission rate to each client. This third parameter can be as small as a fraction (e.g., 25%) greater than the media play rate. Thus, the new RPB protocols extend the state-of-the-art in periodic broadcast protocols that do not recover from packet loss, since all previous periodic broadcast protocols require an aggregate transmission rate to each client of two or more times the media play rate.

First, models are developed that quantify the minimum required server bandwidth for a given average client packet loss rate and maximum latency to begin playback. The models are then used to guide the design and assess the performance of the proposed new protocols. The new protocols are relatively simple to implement and achieve nearly the best possible scalability and efficiency for a given set of client characteristics and desirable/feasible media quality. We further propose using the new protocols to allow clients who arrive late to a live or scheduled multicast to request the earlier media content that they missed, with only a very modest increase in server bandwidth.

The RPB protocols are based on a new approach that enables efficient transmission of redundant data to clients with heterogeneous packet loss rates, and requires each client to simultaneously listen to only a small subset of the server streams delivering the media file. These protocols have the minimum required server bandwidth for the set of tunable protocol parameter values. A key question addressed is how well the RPB protocols scale compared with previous protocols that require the clients to listen to all segment transmissions simultaneously. Performance comparisons in Section 4 show that the scalability of the new RPB protocols is similar to the scalability of these previous protocols. Also, like all previous periodic broadcast protocols, the new RPB protocols require no real-time client feedback to the server, and support arbitrarily high request rates for the media using server bandwidth that grows only logarithmically with the ratio of media play duration to the start-up latency.

The new RBS protocols require limited feedback from the clients (i.e., the initial request from each client that initiates a new stream), and are not as efficient as the RPB protocols with respect to the amount of redundant data that is received by a client. However, like the previous bandwidth skimming protocols, the RBS protocols have minimal start-up latency for client playback, fully support client interactive requests, and require server bandwidth that grows only logarithmically with the client request rate.

The remainder of the paper is organized as follows. Section 2 reviews previous periodic broadcast and bandwidth skimming protocols, and previously developed lower bounds on the required server bandwidth for each of these two classes of protocols. Section 3 compares alternative packet loss recovery strategies, both qualitatively, and through the derivation of simple lower bounds on the required server bandwidth. Sections 4 and 5 develop the proposed new RPB and RBS protocols, respectively, and provide qualitative and quantitative assessments of the new protocols. Conclusions are presented in Section 6.

2 Background

The new protocols proposed in this paper build on the previously developed periodic broadcast and bandwidth skimming techniques, that are briefly reviewed in Sections 2.1 and 2.2, respectively. Section 2.3 reviews a previous analytic model [14, 3, 10] that provides a lower bound on the server bandwidth required for each class of protocol. This model serves as a basis for the new lower bounds that are developed in Section 3. Finally, Section 2.4 summarizes the goals of the new protocols developed in Sections 4 and 5 of the paper.

2.1 Periodic Broadcast Protocols

Periodic broadcast protocols were first proposed by Viswanathan and Imielinski in the form called *Pyramid Broadcasting* [34]. These protocols divide a media object into K segments with relative lengths $l_1, l_2, ..., l_K$, each of which is broadcast periodically according to a fixed schedule. In most protocols, each segment is transmitted at a rate equal to or greater than the media play rate, so that the client can play each segment, including the first segment, as it arrives. Clients must be capable of receiving multiple segments concurrently, at a total rate exceeding the media play rate, and of buffering data received ahead of when it is needed. With appropriate design of the segment sizes, channel transmission rates, and segment broadcast schedule, the required server bandwidth increases only *logarithmically* with the ratio of the total media play duration to the maximum latency for initiating playback.

Here we review the previous periodic broadcast protocols that are most relevant to this paper, namely (1) the *Skyscraper* protocol that has the lowest total transmission rate to each client, equal to twice the media play rate, and (2) *Harmonic Broadcasts* and other recent protocols in which clients receive each segment entirely before playing the segment. For each of the latter protocols, clients must be able to receive all segment transmissions simultaneously.

[1] "Media play rate" is defined as the rate the transmitted data is consumed during playback.

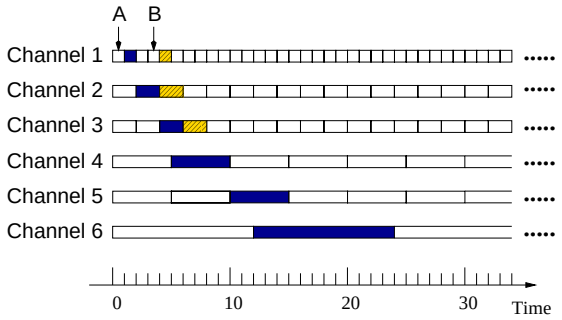


Fig. 1: Skyscraper Broadcasts ($K = 6$)

The *Skyscraper* protocol [17] divides the media file into K segments with relative sizes $1, 2, 2, 5, 5, 12, 12, 25, 25, \ldots$. Each segment is repeatedly broadcast at the media play rate on its own channel, as depicted in Figure 1 for $K = 6$. A client selecting the object at any point in time obtains a schedule for tuning in to each channel, starting at the beginning of the next segment 1 broadcast on channel 1. For example, client A in Figure 1 obtains the schedule of shaded segment broadcasts. A client can begin playback once it begins to receive segment 1. The segment sizes and broadcast schedule ensure that whichever segment 1 broadcast is received, the client begins to receive each other media segment at or before the time it is needed for playback. Since segments are increasing in size, clients that initially listen to different segment 1 broadcasts may listen to the same broadcasts of later segments. For example, client B listens to the same broadcasts of segments 4 through 6 as client A. As in other periodic broadcast protocols, the small first segment permits low start-up latencies while the larger later segments keep the total number of channels needed for the broadcast small.

Skyscraper systems have the key property that clients listen to at most two play rate channels concurrently. Clients must have buffer space equal to a fraction $(l_K - 1)/\sum l_k$ of the media object. Furthermore, the maximum time a client waits to begin receiving a media stream of duration T is $T/\sum_{k=1}^{K} l_k$; e.g., if $K = 10$, the maximum start-up latency is $0.007T$. Other periodic broadcast protocols use different segment size progressions, channel transmission rates, broadcast schedules, and number of channels the client listens to concurrently, to achieve different client buffer requirements and start-up delays for a given server transmission bandwidth.

The *Harmonic Broadcasting* [19] protocol divides a media object into K segments of equal size, and each segment k is broadcast repeatedly on its own channel at rate $1/k$ times the media play rate. Clients listen to all K channels concurrently. In the protocol as originally described, a client begins playback as soon as it begins to receive the first segment, but this does not guarantee jitter-free playback [24]. A simple and efficient correction that guarantees jitter-free playback, is for clients to begin receiving data on all channels immediately (without waiting for a segment boundary), and to start playback as soon as the reception of the first segment is complete. This strategy ensures that each segment is completely received prior to beginning its playout. Harmonic Broadcasting and its variants [24], as well as protocols that follow a hybrid approach [25], achieve lower start-up latency for a given server transmission bandwidth than Skyscraper Broadcasts, but require that clients be able to receive at much higher aggregate transmission rates (i.e., clients must be able to receive on all channels simultaneously).

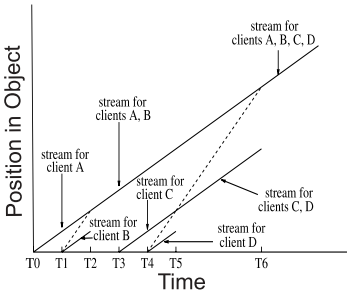
In recent work, Hu [16] derives the segment sizes and transmission rates that minimize server bandwidth for a given client start-up delay assuming (1) a fixed number of segments, (2) each segment is completely received prior to beginning its playout, and (3) each client can receive *all segments concurrently*. No algorithm is presented for determining segment sizes and transmission rates in other cases. For the special case that clients can listen to at most two play rate streams, Hu briefly suggests a Fibonacci series segment size progression with unspecified holes in the transmission schedule. Hu also briefly discusses how periodic broadcast protocols can support a limited form of client fast-forward requests.

In the Skyscraper Broadcast protocol clients need to play a segment as it is being received, which considerably restricts what can be done with respect to packet loss recovery. The periodic broadcast protocols in which each client receives each segment prior to playing the segment can be adapted to use erasure codes to recover from packet loss. Issues that need to be addressed in adapting these protocols include how to accommodate clients with heterogeneous cumulative loss rates, and how to efficiently recover in the case where packet loss is higher than anticipated in the segment encoding. This paper addresses these issues in the context of new high performance protocols that allow clients to listen to a small subset of the server streams delivering the media file.

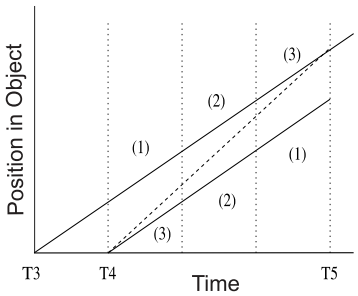
2.2 Bandwidth Skimming Protocols

Bandwidth skimming protocols [12] initiate a new multicast transmission of the media object for each new client request. In the simplest case, each client also listens to the closest earlier stream that is still active [10, 11], so that its own stream can terminate after transmitting the data that was missed in the earlier stream, as illustrated in Figure 2(a). In the figure, clients A through D request the media object at times T0, T1, T3, and T4, respectively. At time T4, client D listens to the stream that starts at T4 as well as the stream that was initiated for client C at time T3. At time T5, the stream for client D can be terminated, and clients C and D are "merged". When clients merge, they begin listening to the closest earlier stream that is still active, and so on.

An important feature of the bandwidth skimming protocols is that the hierarchical merging illustrated in Figure 2(a) can be implemented when the aggregate transmission rate to the client is less than twice the media play rate. For example, in one version of the protocol (called



(a) Hierarchical Multicast Stream Merging



(b) Partition

Fig. 2: Bandwidth Skimming Example

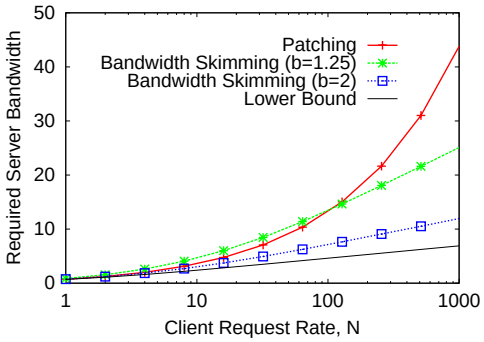


Fig. 3: Required Server Bandwidth

Partition in [12]), each stream is transmitted at the play rate, but on k channels, where k is a parameter of the protocol. Each channel carries $1/k$ of the stream data using a deterministic fine-grained interleaving of the data packets on the channels. Figure 2(b) illustrates how client D merges with client C when each stream is transmitted on three channels (i.e., $k = 3$), each at rate 0.33 times the media play rate, and client D can listen to at most four channels for a maximum client data rate of 1.33 times the media play rate. Client D goes through three distinct periods between arriving at time T4 and merging with client C at time T5. During each period, client D listens to the number of channels from each stream indicated in parentheses near the stream. For example, in the first period, client D listens to one of the channels of client C's stream. In the second period, client D has already received the data that will be delivered on one of the channels transmitting the stream it initiated, so client D listens to two channels of its own stream and two channels of client C's stream.

In bandwidth skimming systems typically one client listening to a stream needs to play out each packet soon after it is received. As in Skyscraper systems, this considerably restricts what can be done with respect to providing packet loss recovery. However, in contrast to both optimized patching and periodic broadcast, bandwidth skimming naturally allows each client to start at an arbitrary point in the media stream, and thus the protocol is easily adapted to support client interactive requests including general "fast forward" requests [10].

Results in [12, 10], illustrated in Figure 3, show that the server transmission bandwidth used for the bandwidth skimming protocols increases only logarithmically (with a small constant factor) as the client request rate increases. As shown in Figure 3, bandwidth skimming with client data rate b only 1.25 times the media play rate yields similar or (for high request rates) substantially better performance than the optimized patching technique [6, 15], which requires client data rate equal to twice the media play rate. Furthermore, the required server bandwidth for bandwidth skimming with $b = 2$ compares favorably with the required bandwidth for Skyscraper systems that have a low start-up delay, shown in Figure 4. Bandwidth skimming with $b = 2$ also achieves nearly the lower bound on required server bandwidth for any protocol that provides immediate service to each client, shown in Figure 3 for comparison purposes. The lower bound is reviewed next.

Table 1: Notation for Scalability Bounds

Symbol	Definition
λ	average client request rate for a media object
T	media object playback duration
N	average number of requests for an object that arrive during a period of length T ($N = \lambda T$), or equivalently, the average number of clients concurrently playing the object
d	maximum start-up delay for object playback, as a fraction of total object duration
B	required server bandwidth (in units of the object playback bit rate)
p	packet loss probability

2.3 Maximum Achievable Scalability

In [10] a tight lower bound is derived on the required server bandwidth[2] for *any* protocol that provides immediate on-demand streaming of multimedia content, with no packet loss recovery. This bound assumes Poisson request arrivals, as measured, for example, in [2]. Assuming the notation in Table 1, the lower bound $B_{min}^{d=0}$ (the superscript indicating that this is for *immediate-service* on-demand streaming) is as follows:

$$B_{min}^{d=0} = \int_0^T \frac{dx}{x + \frac{1}{\lambda}} = \ln(N+1). \qquad (1)$$

The bound was derived by considering a small portion of the object at some arbitrary time offset x. For an arbitrary client request that arrives at time t, this portion of the object must be delivered no later than time $t + x$. If the portion is multicast at time $t + x$, then (at best) those clients that request the object between time t and $t+x$, can receive the same multicast. Assuming Poisson arrivals, the average time from $t+x$ until the next request for the object is $1/\lambda$. Therefore, the minimum frequency of multicasts of the portion at time offset x is $1/(x + 1/\lambda)$, which yields the above bound. Note that the required server bandwidth for delivery of multiple objects can be derived by summing the bandwidth required for each object, weighted by the relative playback rate.

[2]The "required server bandwidth" for an object is defined as the average server bandwidth used by the specified protocol. This bandwidth is measured in units of the media play rate.

As described in [10], the Poisson arrival assumption can be relaxed to cover a wide class of arrival processes (including those with heavy-tailed interarrival time distributions), yielding a similar analytic result with difference bounded by a constant independent of λ. Note also that, as illustrated in [12], the Poisson arrival assumption yields conservative server bandwidth estimates for heavy-tailed interarrival time distributions, since greater burstiness in the arrival process causes clients to be merged more quickly.

Corresponding bounds for systems with various classes of interactive requests have been derived in [31].

The bound in equation (1) can be extended by adding a start-up delay d to the minimum time between multicasts of the portion at position x (i.e., add d to the denominator of the integrated function). For broadcast schemes that provide bounded start-up delay for arbitrarily high (i.e., essentially infinite) client arrival rate, this yields the following lower bound on required server bandwidth, B^d_{min} (the superscript indicating that this is for on-demand streaming with *bounded-delay* d), as shown in [14, 3]:

$$B^d_{min} = \int_0^T \frac{dx}{x+d} = \ln(\frac{T}{d} + 1). \qquad (2)$$

2.4 Goals of the New Streaming Protocols

The key goals of the new delivery protocols developed in Sections 4 and 5 are:

- Convenient: Clients can begin playing the requested media content after a tunably small start-up latency.
- Tolerant: The protocol should tolerate clients with heterogeneous packet loss rates and transmission path data rates.
- Reliable: Clients that have a packet loss rate up to a specified value should be able to reconstruct each media packet prior to its play point.
- Efficient: The protocol should require minimal client feedback, and, if possible, the total amount of data each client receives should be minimal.
- High Quality Media: The protocol should have a tunable parameter that specifies the maximum total transmission rate to a client, expressed as a multiple, perhaps not much larger than 1.0, of the media play rate.
- Scalable: The protocol should require close to the minimum possible server bandwidth for any protocol that satisfies the above goals.

To quantify the goals for designing new protocols that satisfy the above requirements, we consider a simple "piecewise download" approach in which the media file is divided into K equal-size segments, each of which is delivered as a digital fountain. A client downloads one segment at a time, playing the previously downloaded segment while receiving the next segment. Assuming a maximum cumulative loss probability for any client equal to p, and perfect decode efficiency,[3] the transmission rate on each fountain is equal to $1/(1\text{-}p)$ times the media play rate. If the loss rate for a given segment is less than p, the client begins receiving the next segment earlier than anticipated. As long as the cumulative loss rate at any point after receiving the first segment is not higher than p, the client will be able to recover lost packets and play the media without jitter.

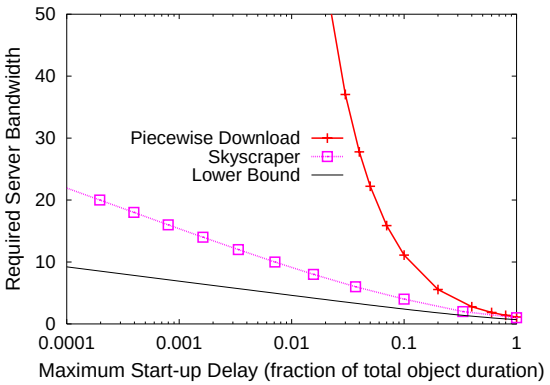


Fig. 4: Server Bandwidth for Protocols with $d > 0$

Figure 4 shows the required server bandwidth as a function of the maximum client start-up delay for two delivery approaches that use existing techniques and have maximum transmission rate to each client no more than two times the media play rate: Skyscraper Broadcasts, and the "piecewise download" approach with loss rate p equal to 10%. Server bandwidth is measured in units of the media play rate. For comparison purposes, the figure also shows the lower bound for any protocol that guarantees the given maximum start-up delay, reviewed in Section 2.3. A key question addressed in this paper is whether new protocols can be devised that share (or perhaps even improve on) the scalability and relatively low required client reception bandwidth of Skyscraper Broadcasts or bandwidth skimming, while simultaneously employing efficient packet loss recovery for heterogeneous clients, as in a digital fountain approach.

3 Loss Recovery Strategies

This section compares three basic strategies for packet loss recovery for on-demand media streaming, namely: unicast retransmission of lost packets, multicast retransmission of lost packets, and multicast transmission of redundant data that is computed using erasure codes. Server-based recovery is assumed; at the cost of additional infrastructure and complexity, distributed recovery architectures are also possible and have been extensively explored [20].

The conclusion drawn from the comparisons below is consistent with conclusions drawn for the reliable scheduled (i.e., not on-demand) multicast setting [23]. That is, erasure codes provide a better solution. However, the anal-

[3] *Decode efficiency* is defined as the ratio of the number of packets that must be received in order to reconstruct a segment to the number of packets in the reconstructed segment. The codes described in [5] have a decode efficiency of about 1.05. For simplicity, we assume unless otherwise noted that decode efficiency is equal to 1.0 and that the time to decode an erasure-coded segment is negligible. As discussed in Section 4.2 in the context of the proposed new RPB protocols, these assumptions are easily relaxed.

ysis quantifies the benefits of erasure codes with respect to the server load imposed by error recovery for scalable on-demand streaming. In the process, new fundamental lower bounds on required server bandwidth are derived that will be applied later in the paper.

3.1 Qualitative Discussion

The three basic approaches can be compared qualitatively along at least three dimensions, namely: implementation complexity, start-up delay, and scalability. Regarding implementation complexity, the use of erasure codes entails the overhead of encoding and decoding the data, but approaches based on retransmission must handle unpredictable retransmission requests and feedback implosion (e.g., [20]).

The *start-up delay* is defined as the time from when a client requests a media object, until the client can begin playback. A retransmission-based approach must have sufficient start-up delay to allow for clients to request retransmissions and for retransmitted data to be successfully received prior to its play point. In an approach based on erasure codes, media data is coded/decoded in coarse grained *blocks*. Further, these blocks may be interleaved during transmission to reduce sensitivity to burst losses. The start-up delay must be sufficient to allow time for each entire block to be received and decoded at the client. In either case, the start-up delay requirement is additive with other start-up delay components, such as that required to deal with network jitter. One might expect total start-up delays to be of similar magnitude in both approaches, but the delay may be more easily estimated for an approach that uses erasure codes and a given interleaved transmission schedule.

Scalability concerns how the server bandwidth required to reliably deliver an object on-demand must increase as a function of the object request rate and client packet loss rates. For multicast streaming, different clients experience different packet losses, and the alternative packet loss recovery schemes differ with respect to effective sharing of redundant data transmissions, and thus with respect to server bandwidth requirements. Multicast retransmissions require the server to resend only one copy of data that multiple clients have not received, but other clients may receive more data than they need to recover from their own losses. Multicast transmission of redundant data computed using erasure codes permits a single redundant packet to repair different losses for different clients, which leads to the qualitative notion that this approach may be more scalable than the retransmission-based approaches.

3.2 Quantitative Scalability Bounds

The lower bounds on required server bandwidth for any protocol that provides reliable on-demand streaming using a specific packet loss recovery strategy build on the bounds for no packet loss recovery reviewed in Section 2.3. The simplest lower bound is for the case that erasure codes are used to recover from packet loss. In this case, each client must receive an amount of (source and redundant) data that is at least equal to the size of the object. If the average client packet loss probability is p, in the best case each client has average packet loss probability equal to p, and in this case the server must transmit an amount of data per unit of time (on average) that is a factor of $\frac{1}{1-p}$ greater than when packet loss recovery is not performed (since on average $1-p$ of the packets will be received). Thus, the lower bound for immediate service using erasure codes is:

$$B^{d=0}_{min,erasure} = \frac{1}{1-p}\ln(N+1), \tag{3}$$

and the lower bound for on-demand streaming with bounded delay is

$$B^{d}_{min,erasure} = \frac{1}{1-p}\ln(\frac{T}{d}+1). \tag{4}$$

The above bounds are used to evaluate the proposed new protocols in Sections 4 and 5.

Bounds for retransmission strategies are derived here for the case of immediate service. For unicast retransmission of lost packets, if the object playback duration is T minutes, the average amount of data that is retransmitted per client, measured in playback minutes, is equal to $\frac{pT}{1-p}$. Given a client request rate of λ, the server bandwidth required just for the retransmitted data is $\lambda\frac{pT}{1-p}$. Using $N=\lambda T$ and the minimal server bandwidth needed for the original packet transmissions from the bound in equation (1), yields the following lower bound, $B^{d=0}_{min,ur}$ (the second subscript indicating that this is with use of unicast retransmissions):

$$B^{d=0}_{min,ur} = \ln(N+1) + \frac{p}{1-p}N. \tag{5}$$

The lower bound on required server bandwidth in the case of multicast retransmissions of lost packets is derived in Appendix A. The bound is not obtained in closed form, but can be solved numerically for particular values of N and p. The bound for multicast retransmissions assumes that each packet is lost with independent probability p, although generalizations are possible.

The above bounds for unicast retransmissions of lost packets and for multicast transmission of redundant data using erasure codes can be generalized for non-Poisson request arrival processes as described in [10] for the previous bound in equation (1). The bound for multicast retransmissions is more complex, and at present a generalization for non-Poisson arrival processes has not been formulated.

3.3 Numerical Results

Figure 5 presents the immediate service bounds derived above as functions of the normalized client request rate N, for 15% average packet loss probability (p). Smaller but similar and still significant differences in the bounds are observed for lower values of p.

Results in Sections 4 and 5 will show that the lower bounds for packet loss recovery using erasure codes can be

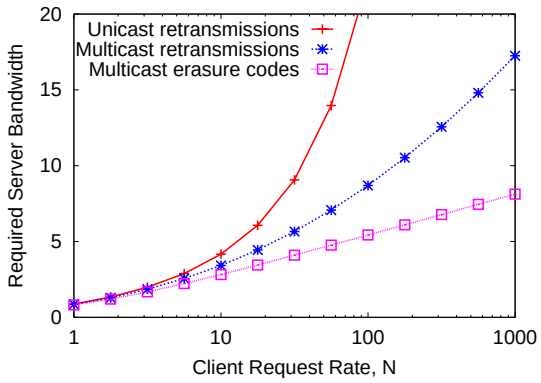


Fig. 5: B_{min} for Immediate Service & Packet Loss Recovery (15% Packet Loss)

Table 2: Parameters of Optimized PB and RPB

Symbol	Definition
K	total number of segments
r	segment transmission rate, in units of the object playback bit rate
s	assumed maximum number of streams that clients can listen to concurrently ($s \leq K$)
b	assumed maximum aggregate transmission rate to a client, in units of the object play rate ($b = s \times r > 1$)
l_k	length (i.e., playback duration) of the k^{th} segment, relative to the length of segment 1

closely approached by actual streaming protocols. The results in Figure 5 show that even a "perfect" retransmission-based recovery strategy would require more server bandwidth. Given the implementation difficulties for retransmission based approaches, the new reliable broadcast protocols developed next use the erasure coding strategy.

4 Reliable Periodic Broadcast

In this section we develop optimized broadcast protocols that (1) assume a maximum aggregate transmission rate to any given client that is a tunable parameter, b, which may be a small fraction greater than the media play rate, and (2) enable clients with heterogeneous loss probability to recover from packet loss. The notation used in developing the RPB protocols is given in Table 2.

To deal with the fundamental challenges involved in designing such protocols, we proceed in four stages. Section 4.1 develops a family of Optimized Periodic Broadcast (Optimized PB) protocols that do not support packet loss recovery, but have the maximum possible segment size increases for a given segment streaming rate (r) and maximum number of streams that clients can listen to concurrently (s), under the constraint that clients receive each segment entirely before playing the beginning of the segment. These protocols thus have the minimum possible start-up delay for a given total server bandwidth, or the minimum possible server bandwidth for a given start-up delay, under the stated constraints. Like the Skyscraper protocol, each segment is transmitted on its own equal-rate channel. However, since each segment is downloaded before playing the segment, each segment transmission rate (r), including the first segment transmission rate, can be lower than the media play rate. We can thus explore the minimum server bandwidth as a function of client start-up delay when the aggregate transmission rate to the client ($b = s \times r$) is less than twice the media play rate. Performance of the Optimized PB protocols with $b = 2$ times the media play rate, for various values of r and s, is compared against that for the previous Skyscraper protocol.

Section 4.2 extends the Optimized PB protocols to create a family of basic Reliable Periodic Broadcast (Basic RPB) protocols. These protocols transmit data that has been encoded using erasure codes, enabling each client to reconstruct each segment k prior to the time the segment needs to be played, assuming that the cumulative loss at the end of receiving each segment is not greater than a tunable parameter p which has the same value for each segment. The Basic RPB protocols have maximum segment size increases for the given values of r, s, and p. Appendix B provides an asymptotic analysis of the new Basic RPB protocols which shows that the required server bandwidth can approach the lower bound provided in equation (4) for reliable delivery using erasure codes. The analysis also suggests that the protocols can approach the minimum possible required server bandwidth under any client bandwidth constraint.

Section 4.3 generalizes the Basic RPB protocols to allow each segment to have a different associated cumulative loss protection. These RPB protocols address the impact of bursty packet losses on reliable reconstruction of earlier segments, and allow the estimation of cumulative packet loss rate to be less conservative for later segments.

Section 4.4 addresses heterogeneous client packet loss probabilities and jitter-free delivery to clients with packet loss rate greater than the specified upper bound.

4.1 Optimized PB Protocols

The proposed new family of Optimized PB protocols assumes that (1) packet loss in any given transmission path can be adequately addressed using local error concealment, and (2) each segment must be entirely received before the beginning of the segment is played. Each segment is repeatedly transmitted on its own multicast channel, as illustrated in Figure 6 for $K = 6$, $r = 1$ and $s = 2$. As in Figure 1, time advances from left to right, the sequence of rectangles for each channel indicates the sequence of transmissions of the associated segment, and the shaded regions show the reception period on each channel for an example client. In these protocols, clients generally obtain each complete segment from listening to portions of two consecutive transmissions of that segment. In particular, a client arriving at an arbitrary point in time immediately begins listening to the first s channels transmitting segments 1

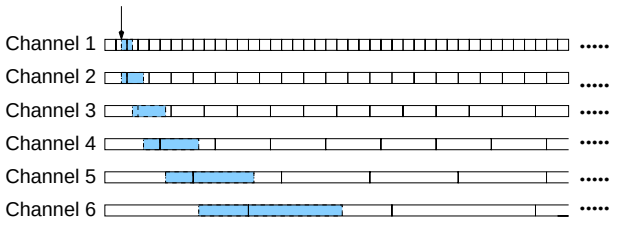


Fig. 6: Example Optimized PB Protocol ($K = 6$, $r = 1$, $s = 2$)

through s, each for a period of time equal to the time it takes to transmit the respective segment. Once the first segment is fully received, the client begins playback and also begins receiving segment $s+1$. The segment sizes are designed so that the client will receive each entire segment just in time to begin playing the segment.

To derive the maximum possible segment size increases, first consider the case that each segment is delivered at the object playback rate[4] (i.e., $r = 1$) and multicast join operations have zero latency. In this case, if s is the specified maximum number of segment transmissions a client can simultaneously listen to, then each segment k, $1 < k \leq s$, has maximum length equal to the time to receive segment 1 plus the time to play segments 1 through $k-1$. For segment $k > s$, the client will begin receiving segment k at the time that segment $k-s$ is just received and starts playing. The client must finish receiving segment k by the time that segment $k-1$ finishes playing. Thus, measuring segment lengths relative to the first segment length, we have $l_1 = 1$ and the following maximum sizes for segments $k > 1$:

$$l_k = \{l_1 + \sum_{j=1}^{k-1} l_j,\ 1 < k \leq s;\ \sum_{j=k-s}^{k-1} l_j,\ k > s\}. \qquad (6)$$

For a given number of server streams, K, used to multicast the object, the total server bandwidth used is $B = r \times K$ and the client start-up delay is equal to $\frac{T}{r \sum l_k}$.

Figure 6 illustrates the segment sizes and a transmission schedule for the Optimized PB protocol with parameters $K = 6$, $r = 1$, and $s = 2$. Note that (1) any alignment of transmissions between any two channels is valid, (2) two clients arriving at different points in time generally tune into each segment multicast stream at different times, (3) if $r = 1$ and $s = 2$, the segment length progression is the Fibonacci series, and (4) for any Optimized PB parameter settings, the segment transmission schedule has no holes.

Generalizing the Optimized PB protocols for segment transmission rate $r \leq 1$ enables the maximum total transmission rate to each client, $b = s \times r > 1$, to be less than twice the minimum transmission rate required for real-time playback. As noted previously, this allows more of the achievable transmission rate to a client to be used for delivering higher quality media content. Furthermore, as will be illustrated below, for any given value of b and total server transmission bandwidth (B), a lower value of r yields a lower start-up delay (but a larger number of server multicast transmission streams, $K = B/r$).

Noting that the time to download each segment is equal to $1/r$ times the segment length, the maximum relative segments sizes are easily generalized for $r \leq 1$, as follows:

$$\frac{1}{r} l_k = \{\frac{1}{r} l_1 + \sum_{j=1}^{k-1} l_j,\ 1 < k \leq s;\ \sum_{j=k-s}^{k-1} l_j,\ k > s\}. \qquad (7)$$

Figure 7(a) shows the required server bandwidth as a function of start-up delay for two Optimized PB protocols with $b = 2$, as well as the required server bandwidth versus the average and maximum start-up delay for the Skyscraper system which assumes $b = 2$. Maximum start-up delay is twice the average start-up delay in Skyscraper systems. The curves show that the performance of the Optimized PB systems is competitive with the Skyscraper system, even though each segment is completely received before the beginning of the segment is played.

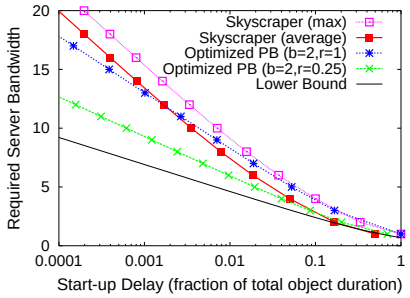
Figure 7(a) shows that the performance of the Optimized PB systems improves for lower segment transmission rate, $r = b/s$. This performance improvement is explored further in Figure 7(b), which compares the segment multicast frequency for different Optimized PB systems that assume the same client bandwidth (b) and approximately equal start-up delay (d), but different segment transmission rate (r).

The lower bound curve shown in the figure is derived from the fact that any periodic broadcast protocol must multicast the portion of the object at position x with frequency at least $1/(x+d)$. For a finite number of segments, all data in a segment is multicast with the same frequency, yet only the data at the beginning of the segment is needed for playback at the time the segment download is complete. Lowering r implies that the number of segments will be correspondingly increased, and thus each segment size is reduced. This implies that a larger fraction of the segment is received "just in time" and the multicast frequencies more closely track the lower bound curve. (A gap remains with respect to the lower bound curve even for $r = 0.02$, due to the limited aggregate transmission rate to each client (i.e., $b = 2$).)

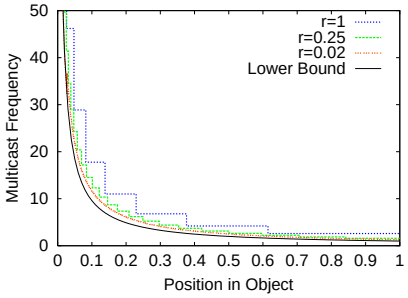
The Optimized PB segment sizes are easily generalized for non-negligible latency to join an on-going multicast. In this case, a conservative estimate of the latency to join the multicast transmission is added to the left-hand side of equation (7) and to the time to download the first segment on the right-hand side of the equation for $k \leq s$.

Required client buffer space for an Optimized PB protocol can be derived by summing the amounts by which the buffer fills during the download of the initial segment, and during each segment play time for which the download rate exceeds the play rate. Expressed as a fraction of the total

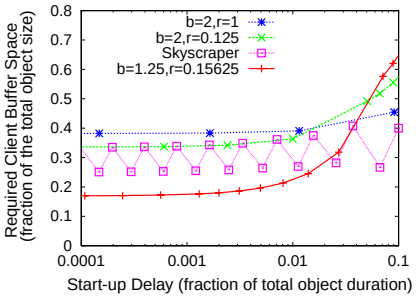
[4]For simplicity, the development of the segment sizes assumes constant bit rate content or a fully smoothed variable bit rate (VBR) stream with the requisite additional start-up delay (if any). The segment sizes can be further optimized for VBR content that isn't (fully) smoothed. In this case each segment can be fully smoothed and delivered in a constant bit rate stream, but calculation of the optimized segment transmission rates and sizes is substantially more complex.



(a) Required Server Bandwidth



(b) Segment Multicast Frequency ($b = 2$, $d \approx 0.0043T$)



(c) Client Buffer Requirements

Fig. 7: Optimized PB Protocol Performance

object size, the required client buffer space is given by

$$\frac{(sl_1 + (b-1)\sum_{k=1}^{K-s} l_k + \sum_{k=K-s+1}^{K-s+\lfloor \frac{b-1}{r} \rfloor} ((K-k)r - 1)l_k)}{\sum_{k=1}^{K} l_k}. \quad (8)$$

Client buffer requirements for Optimized PB protocols with various values of b and r are shown in Figure 7(c). The client buffer requirements for Skyscraper systems are shown for comparison.[5]

4.2 Basic Reliable Periodic Broadcast

We next consider modifying the Optimized PB protocols to enable *efficient* recovery from packet losses in the transmission path to each client. This section generalizes the protocols for the (idealized) case that the "cumulative packet loss" at the end of receiving each segment is never greater than a tunable parameter p.[6] Sections 4.3 and 4.4 will provide a further optimization and generalizations for clients that have packet loss greater than the assumed upper bound.

A key insight, supported by Figure 6, is that the Optimized PB protocol can be extended to enable efficient recovery from packet loss if *each segment* is delivered by an approximation of a digital fountain. That is, each segment is encoded using an erasure code with a large stretch factor [5], and the resulting encoded packets are transmitted on the channel. A client can listen to each channel until it has correctly received the number of packets required to reconstruct the respective segment, and then listen to the next segment that must be received. Assuming perfect decode efficiency and maximum packet loss equal to p, the client will receive the requisite number of packets for segment k by time $1/(1-p) \times l_k/r$. Letting $a = 1/(1-p)$ and assuming segment decode time is negligible, the maximum relative segment sizes are as follows:

$$a \times \frac{l_k}{r} = \{a \times \frac{l_1}{r} + \sum_{j=1}^{k-1} l_j,\ 1 < k \le s;\ \sum_{j=k-s}^{k-1} l_j,\ k > s\}. \quad (9)$$

Figure 8 illustrates the portion of each multicast stream listened to by a sample client for the Optimized RPB protocol with parameters $K = 6$, $r = 1$, $s = 2$, and $p = 0.1$, assuming that the client experiences a packet loss rate of exactly p on each stream. Note that, in contrast to Figure 6, there are no segment boundaries indicated in the figure, since each channel carries a stream of encoded packets each of which is essentially equivalent with respect to its usefulness to a client. Note also that the listening time on each channel can increase or decrease depending on the actual packet loss experienced. In particular, lower packet loss (and thus reduced listening time) for an earlier segment permits tolerance of greater packet loss (increased listening time) for a later segment, as long as the cumulative loss does not exceed p.

For a given number of server streams, the (deterministic) start-up delay is equal to $\frac{aT}{r\sum l_k}$. The segment sizes given in equation (9) can be modified to account for non-negligible segment decode times by adding the decode time for segment k to the left-hand side of the equation and by adding the decode time for segment 1 to the download time for segment 1 on the right-hand side. Imperfect decode efficiency can be accounted for by letting a equal the actual

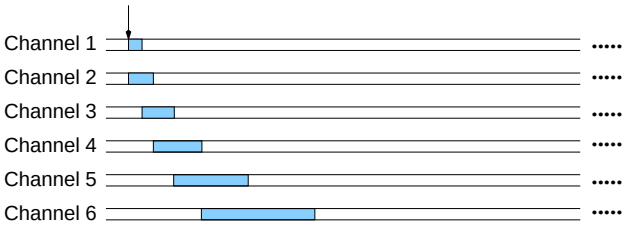
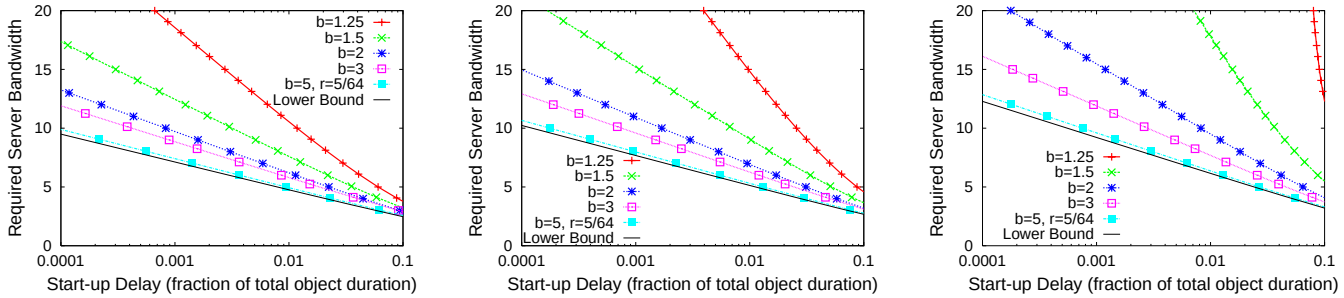


Fig. 8: Example Optimized RPB Protocol ($K = 6$, $r = 1$, $s = 2$, 10% Packet Loss)

[5] The Skyscraper curve is non-monotonic because each odd-numbered segment has the same size as the previous even-numbered segment.

[6] By "cumulative loss" at the end of receiving segment k, we intuitively mean the average loss over segment k and the "preceding segments". Since multiple segments are received in parallel, making this notion precise requires some care. When obtaining numerical results that require a more precise definition (specifically, for Figure 11), we define as the "preceding" segments, those that must be received prior to the segment of interest and that use the "same unit" of client bandwidth when segment receptions are completed in-order; i.e., for segment k, segments $k-s$, $k-2s$, ..., would be the preceding segments that are considered.



(a) 3% Packet Loss (b) 10% Packet Loss (c) 25% Packet Loss

Fig. 9: Performance of the Basic RPB Protocols ($a = \frac{1}{1-p}$, $r = b/8$ unless otherwise specified)

decode efficiency times $1/(1-p)$. For example if $p = 0.2$ and decode efficiency is 1.05, $a = 1.05 \times 1.25 = 1.3125$. Segment sizes may be capped at some maximum value so as to control client buffer size and decoding time requirements. In the following performance results we assume all segment sizes are as given by equation (9).

Figures 9 & 10 provide numerical results for the required server bandwidth (in units of the object playback bit rate) for delivery of a single object, as a function of the client start-up delay. Decode efficiency is assumed to be 1.0 and decode time is assumed to be negligible in these figures; required server bandwidth will increase for larger values of either of these parameters. Figure 9 shows the impact of b and p on the required server bandwidth as a function of the start-up delay. As shown in the figure, if the maximum cumulative packet loss (p) is 10% or less, start-up delay equal to 1% of T is feasible even if the aggregate transmission rate to each client (b) is only 1.25 times the media play rate. As b increases and (for fixed b) r decreases, lower target start-up delays become more feasible (even for higher values of p), and the server bandwidth required by the Optimized PB protocol approaches the lower bound on the required server bandwidth for recovering from loss rate equal to p, given in equation (4). Recall that the lower bound assumes unlimited aggregate transmission rate to each client.

Figure 10 illustrates the impact of r on required server bandwidth. The maximum transmission rate to a client is fixed at $b = 2$. Thus, as r decreases the maximum number of streams a client listens to (s) increases. As shown in the figure, performance improves as r decreases, although with diminishing returns. For example, the benefit for $r < 0.25$ may not be worth the cost of managing the additional multicast streams. On the other hand, decreasing r yields shorter segments which reduces the decode time per segment. These factors need to be considered when selecting the value of r for a given implementation of the protocol.

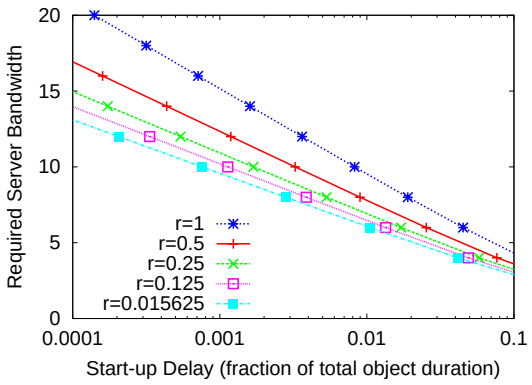


Fig. 10: Impact of Streaming Rate on RPB Performance ($b = 2$, $a = \frac{1}{1-p}$, 10% Packet Loss)

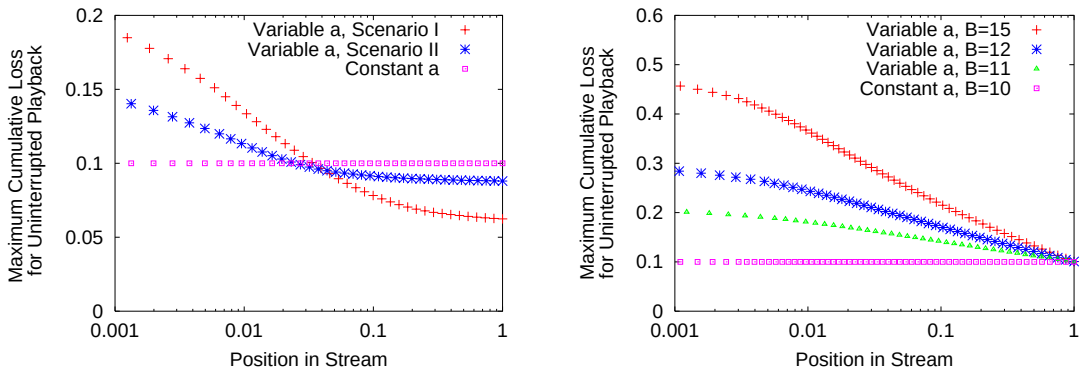
4.3 RPB Protocols for Bursty Packet Loss

In the Basic RPB protocols, each client that observes cumulative packet loss rate less than or equal to p at the end of receiving each segment will be able to recover from packet loss and play the media object without interruption. Furthermore, if a given client observes cumulative packet loss rate less than p at the end of receiving a given segment, that segment can be reconstructed prior to its playout point. In this case, the client can begin listening to later segments earlier than anticipated. This "work-ahead" allows the client to tolerate a higher loss rate than p for a later segment, with no interruption in playback if the cumulative loss probability at the end of receiving the later segment is still less than or equal to p. Permitting clients to begin reception of each segment at an arbitrary point in time enables this useful work-ahead capability.

As discussed in extensive previous work (e.g., [26] and the references therein), packet loss is typically quite bursty. Spikes in the packet loss rate imply that earlier segments in the Basic RPB system require a higher level of loss protection than later segments.

A greater level of protection for earlier segments can be accomplished by letting each segment have a different value a_k that determines the default time the client will listen to the stream for segment k. For a given average loss rate p and $a = 1/(1-p)$ times the decode efficiency, earlier segments will have a_k larger than a and later segments will have a_k lower than a. The RPB segment sizes for the specified values of a_k are easily derived as follows:

$$a_k \times \frac{l_k}{r} = \{a_1 \times \frac{l_1}{r} + \sum_{j=1}^{k-1} l_j,\ 1 < k \leq s;\ \sum_{j=k-s}^{k-1} l_j,\ k > s\}. \quad (10)$$



(a) Fixed Server Bandwidth ($B = 10$)

(b) Fixed Overall Loss Protection

Fig. 11: Loss Tolerance Using Variable a $(b = 2, r = 0.25, d = 0.00017T)$

The above specification for RBP systems can be modified to include non-negligible multicast join or segment decode times, as discussed for the Basic RPB protocols.

For a given environment with specified values of b, r, and a_k, the required server bandwidth ($B = K \times r$) can be plotted as a function of start-up delay, to support server provisioning decisions.

Figure 11(a) shows that a variety of skews in the loss protection can be achieved with a fixed server bandwidth and start-up delay. That is, for $B = 10$, $b = 2$, $s = 8$, and $d = 0.0017T$, the figure shows the cumulative loss protection (at the end of each segment) provided with a_k equal for all segments, and the loss protection provided in two example scenarios with variable a_k where higher protection is provided to the initial segments by trading off protection for the latter segments. Alternatively, higher loss protection can be achieved for the initial segments with a fixed overall cumulative loss protection and start-up delay by modest increases in server bandwidth, as illustrated in Figure 11(b). In any case, a careful analysis of the traffic characteristics in the implementation environment is needed to properly tailor the loss protection. Note also that in many environments (including the Internet and wireless networks) only a small to moderate skew in the protection may be desired, because more significant spikes in the loss rate indicate that the total transmission rate for the media object should be reduced in order to alleviate congestion in the transmission network. This issue is discussed next.

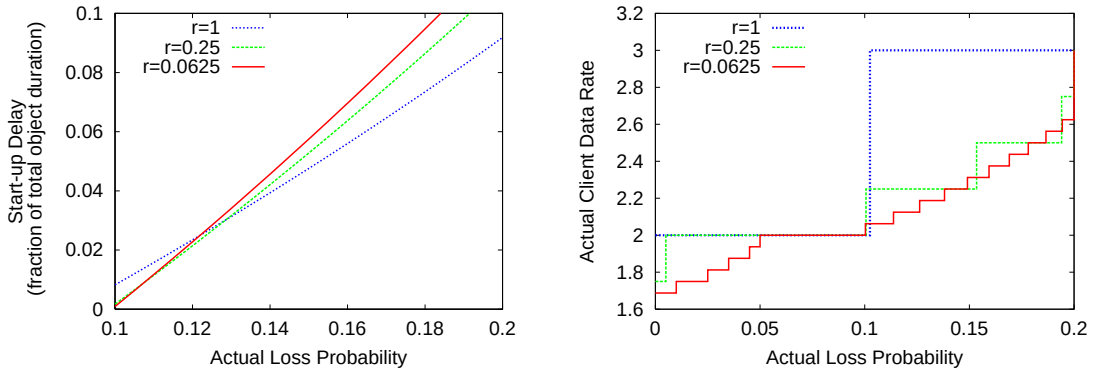
4.4 Client Heterogeneity

The RPB family of protocols so far defined assume that all clients have the same maximum aggregate transmission rate (equal to b times the media playback rate), and the same maximum cumulative loss probability at the end of receiving each segment. This section addresses the issues of heterogeneous client transmission path bandwidths and loss rates.

For congestion control and/or to keep packet loss rate within a reasonable range for a given client, one of two approaches might be employed, based on use of "simulcast" or a layered media encoding [20], respectively. In the simulcast approach, several different versions of the object encoded for different bit rates are each delivered using the RPB protocol. If the client may dynamically switch between the different versions based on observed changes in the transmission channel, the RPB protocol for each version should have the same parameter values (i.e., b, r, K, and a_k) so that interruption in playback is minimized when version changes are made. Alternatively, the media object might be compressed using a layered encoding scheme, and the RPB protocol might be applied to each encoded layer. In the simplest case, the parameters of the RPB protocol would be the same for each layer. More complex schemes that allow each layer to have different parameters, for example to implement greater loss protection for the base layer, can also be designed such that the time to receive each segment k is the same for each layer. In any case, a client with a given aggregate transmission bandwidth (perhaps experimentally determined) subscribes to the appropriate number of layers, possibly dynamically in response to observed changes in the transmission channel.

For a given simulcast version or set of layers, modest heterogeneity in client capabilities can be accommodated using the tradeoffs among start-up delay, client data rate, and loss rate that are supported by the RPB protocol. Precise policies and protocols for exploiting these tradeoffs are context dependent and are left for future work, but we outline the basic ideas in the following two examples.

Figure 12(a) shows how a client with a somewhat higher packet loss rate than specified in an RPB protocol can achieve full packet loss recovery and uninterrupted playback by adopting a higher start-up delay. In this case, the protocol is designed for client bandwidth $b = 2$, fixed cumulative packet loss threshold $p = 10\%$, and three different values of the segment streaming rate r. Start-up delay needed for jitter-free playback is plotted versus the actual maximum cumulative loss observed by a given client. The very low start-up delay for actual loss probability equal to 10% is the start-up delay that the protocol is designed for. However, an individual client that observes cumulative packet loss closer to 15% can achieve jitter-free playback if its start-up delay is slightly more than 0.05T, or 5% of



(a) Start-up Delay vs. Loss Probability

(b) Client Data Rate vs. Loss Probability

Fig. 12: RPB Performance for Heterogeneous Clients ($B = 10, b = 2, p = 0.1$)

the object playback duration. One possible technique by which this tradeoff could be exploited would involve the client monitoring the packet loss rate prior to playing the first segment (i.e., while receiving the first s segments), and delaying playback if the cumulative loss rate is greater than p. The values of $p = 0.1$ and actual loss rate equal to 15% are only illustrative examples. The key point here is that an individual client with a somewhat higher packet loss rate than assumed in the protocol can select an increased start-up delay instead of dropping to a lower quality simulcast version or fewer layers.

As a second example, Figure 12(b) shows that if a client has higher aggregate transmission rate than specified in the design of the RPB protocol, the client can achieve uninterrupted playback with a higher loss rate than specified in the protocol design, by listening to more streams concurrently than specified in the RPB protocol. The figure illustrates this for three RPB protocols, each designed for $B = 10$, $b = 2$, and $p = 0.1$. Each protocol has a different segment streaming rate r and associated $s = b/r$. If the actual loss probability observed by an individual client is a value greater than 0.1, the client can achieve uninterrupted playback if the client can receive at the actual transmission rate (greater than 2) given in the curve for the RPB protocol employed to deliver the object. Each step in the curve indicates that the client must listen to one additional stream concurrently to achieve uninterrupted playback. Also note that a slightly larger start-up delay will also be required to ensure that the initial s segments are received in time for playback if these segments have higher than the anticipated loss probability. Conversely, a client that observes lower than anticipated loss probability can in some cases listen to fewer than s streams for uninterrupted playback, as shown in the figure for loss probability less than 0.1.

In the context of the above mechanisms for accommodating heterogeneous clients, the server can also use mechanisms such as redundant transmission paths [22] to reduce the cumulative loss probability to a given client. If, in spite of such mechanisms, the cumulative loss rate observed by a given client exceeds the loss that can be tolerated, the only alternative is to switch to a lower quality media stream (by subscribing to fewer layers or a different version). If that fails, interruption in playback is unavoidable.

5 Reliable Bandwidth Skimming

The Reliable Bandwidth Skimming (RBS) protocols that are developed in this section have at least two advantages compared to the RPB protocols. First, server bandwidth decreases when the client request rate decreases. Second, they support more general (but not zero delay) "fast forward" and "skip ahead" client requests. However, the RBS protocols are somewhat less efficient than the RPB protocols with respect to the amount of data received by each client, and the RBS protocols may be less able to tolerate bursty packet loss without interruption in playback. Furthermore, servers that use the RBS protocols require client requests in order to initiate a new transmission of the media stream, whereas servers that use the RPB protocols can simply announce the current channel broadcast schedules (e.g., on a Web page).

The RBS protocols divide the media stream into segments that have fixed duration as short as possible for the required loss protection. Each segment is encoded using an erasure code. The server transmits $1/(1-p)$ encoded packets for each segment using a rate 1.0 *primary* stream and a rate $p/(1-p)$ *secondary* stream, where p is the specified maximum packet loss rate for each segment. The secondary stream is offset in time by the segment duration to provide some protection against burst losses. The client begins playing the media after the first segment has been received on both streams, and decoded. Assuming the packet loss rate for each segment is not greater than p, each subsequent segment is received and decoded prior to its play point.

The primary stream (and the corresponding "redundant" secondary stream) can be merged with earlier primary (and corresponding secondary) streams in the same way that merging is accomplished in the original bandwidth skimming protocols. For example, if the Partition stream merging method described in Section 2.2 is used, the primary and secondary streams are each transmitted on their own k channels, with fine grained interleaving of the respective

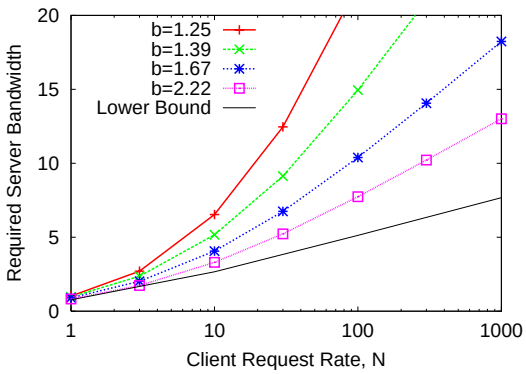


Fig. 13: RBS Performance (10% Packet Loss)

packets on the channels. Clients receive $k+1$ primary and $k+1$ secondary substreams, yielding a total client transmission rate $b = (1+1/k)(1+p/(1-p))$, in units of the media play rate. Merging the streams is the key extension to previously proposed schemes for reliable live or scheduled broadcasts, which also involve establishing separate streams of redundant data [9, 32].

Various extensions to RBS for tolerating heterogeneous client capabilities are possible. For example, feedback to the server reporting high loss rate could trigger the server to create one or more new channels in a secondary stream for transmitting more redundant data to provide increased protection against packet loss. Further specification of methods for accommodating heterogeneous clients in RBS systems is left for future work.

Figure 13 illustrates the performance of reliable bandwidth skimming protocols for maximum loss rate equal to 10%, and various values for b, the achievable client data rate.[7] For comparison purposes, the graph also shows the lower bound from equation (3), which assumes an unlimited client data rate. The principal observations from this figure are that the RBS protocols adapt to varying client request rate, and in light of the assumed client data rates, yield performance reasonably close to the lower bound.

Comparing Figures 13 and 9(b), it can be seen that when the RPB protocol is designed for a given client data rate (e.g., $b = 2$) and a given low start-up delay (e.g., $d = 0.05T$), then the required server bandwidth is fixed (e.g., equal to 8) for all client arrival rates. The RBS protocol with a similar client data rate (e.g., $b = 2.22$) has required server bandwidth that is higher for high client request rates (e.g., $N > 100$) but also somewhat lower for lower client request rates. This illustrates the property that the RBS protocols automatically decrease required server bandwidth as client request rate decreases.

We have recently implemented a prototype system to experiment with on-demand streaming protocols. This prototype implements bandwidth skimming (without packet loss recovery) with aggregate client transmission rate (b) equal to 2, and is installed in the eTeach system at the University of Wisconsin, which handles 1500 client requests per day when classes are in session. Clients request the videos and frequently pause, resume, fast-forward, rewind, and jump to special markers in the content, at arbitrary points in time [2]. The implementation has demonstrated that bandwidth skimming (1) is simple to implement, particularly for the closest target (CT) variant of the protocol [11], (2) is easily used to support client interactive requests, and (3) can be designed with almost no client feedback for effecting the stream merging. Extensions to provide packet loss recovery using the RBS protocol are in progress.

[7] These results are obtained from simulations under Poisson request arrivals, although as reported in [12], qualitatively very similar results are obtained for a heavy-tailed distribution of interrequest times modeled by a Pareto distribution. All results in the figure have 95% confidence intervals that are within 5% of the reported values.

6 Conclusions

This paper has developed and evaluated two classes of scalable and reliable protocols for on-demand delivery of streaming media, namely Reliable Periodic Broadcast (RPB) and Reliable Bandwidth Skimming (RBS). A key property of the protocols is that the upper bound on transmission rate to each client can be set to a small factor larger than the media play rate. Another key property, particularly for the RPB protocols, is that redundant data is transmitted efficiently to clients with bursty losses and heterogeneous loss rates.

The evaluation of the protocols relied in part on simple lower bounds on required server bandwidth for any protocol that provides uninterrupted playback when the average packet loss rate is bounded by p. One of the bounds assumes the server provides immediate service to each client; the other assumes the server serves an unlimited number of clients and a specified maximum client start-up delay. Each new class of protocols nearly achieves the applicable lower bound, and thus achieves nearly the best possible scalability. Furthermore, the required server bandwidths are low enough that either the RPB or the RBS protocols could be used to allow clients that arrive late to a live or scheduled (reliable) broadcast to retrieve the portion of the broadcast that they missed with only a very modest additional burden on server bandwidth.

On-going research includes experimental evaluation of the new RPB and RBS protocols, developing congestion control strategies for RPB and RBS systems, developing protocols for VBR content that has not been fully smoothed, developing RPB systems that have different loss protection for different layers in a layered media stream, and quantifying required network bandwidth for RPB and RBS systems.

Acknowledgments

We would like to thank John Zahorjan for early technical discussions on this topic. We also thank Jussara Almeida, Paul Barford, and the anonymous referees for comments that improved the paper presentation. This work was partially supported by the National Sciences and Engineering

Research Council of Canada under Grant OGP-0000264 and by the National Science Foundation under Grant CCR 9975044.

References

[1] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "A Permutation Based Pyramid Broadcasting Scheme for Video-On-Demand Systems", *Proc. IEEE ICMCS '96*, Hiroshima, Japan, June 1996.

[2] J. M. Almeida, J. Krueger, D. L. Eager, and M. K. Vernon, "Analysis of Educational Media Server Workloads", *Proc. NOSSDAV '01*, Port Jefferson, NY, June 2001.

[3] Y. Birk and R. Mondri, "Tailored Transmissions for Efficient Near-Video-On-Demand Service", *Proc. IEEE ICMCS '99*, Florence, Italy, June 1999.

[4] J. C. Bolot, S. Parisis, and D. Towsley, "Adaptive FEC-Based Error Control for Internet Telephony", *Proc. IEEE Infocom '99*, New York, NY, March 1999.

[5] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data", *Proc. ACM Sigcomm '98*, Vancouver, Canada, Sept. 1998.

[6] Y. Cai, K. A. Hua, and K. Vu, "Optimizing Patching Performance", *Proc. MMCN '99*, San Jose, CA, Jan. 1999.

[7] G. Carle and E. W. Biersack, "Survey of Error Recovery Techniques for IP-based Audio-Visual Multicast Applications", *IEEE Network*, Vol. 11, No. 6, Nov./Dec. 1997.

[8] S. W. Carter and D. D. E. Long, "Improving Video-on-Demand Server Efficiency Through Stream Tapping", *Proc. ICCCN '97*, Las Vegas, Sept. 1997.

[9] P. A. Chou, A. E. Mohr, A. Wang, and S. Mehrotra, "FEC and Pseudo-ARQ for Receiver-driven Layered Multicast of Audio and Video", *Proc. IEEE Data Compression Conf.*, Snowbird, UT, March 2000.

[10] D. L. Eager, M. K. Vernon, and J. Zahorjan, "Minimizing Bandwidth Requirements for On-Demand Data Delivery", *IEEE Trans. on Knowledge and Data Engineering*, Vol. 13, No. 5, Sept./Oct. 2001.

[11] D. L. Eager, M. K. Vernon, and J. Zahorjan, "Optimal and Efficient Merging Schedules for Video-on-Demand Servers", *Proc. ACM Multimedia '99*, Orlando, FL, Nov. 1999.

[12] D. L. Eager, M. K. Vernon, and J. Zahorjan, "Bandwidth Skimming: A Technique for Cost-Effective Video-on-Demand", *Proc. MMCN '00*, San Jose, CA, Jan. 2000.

[13] S. Floyd, V. Jacobson, C. G. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", *Proc. ACM Sigcomm '95*, Cambridge, MA, Aug. 1995.

[14] L. Gao, J. Kurose, and D. Towsley, "Efficient Schemes for Broadcasting Popular Videos", *Proc. NOSSDAV '98*, Cambridge, UK, July 1998.

[15] L. Gao and D. Towsley, "Supplying Instantaneous Video-on-Demand Services Using Controlled Multicast", *Proc. IEEE ICMCS '99*, Florence, Italy, June 1999.

[16] A. Hu, "Video-on-Demand Broadcasting Protocols: A Comprehensive Study", *Proc. IEEE Infocom '01*, Anchorage, AK, April 2001.

[17] K. A. Hua and S. Sheu, "Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems", *Proc. ACM Sigcomm '97*, Cannes, France, Sept. 1997.

[18] K. A. Hua, Y. Cai, and S. Sheu, "Patching: A Multicast Technique for True Video-On-Demand Services", *Proc. ACM Multimedia '98*, Bristol, U.K., Sept. 1998.

[19] L. Juhn and L. Tseng, "Fast Data Broadcasting and Receiving Scheme for Popular Video Service", *IEEE Trans. on Broadcasting*, Vol. 44, No. 1, March 1998.

[20] X. Li, M. H. Ammar, and S. Paul, "Video Multicast over the Internet", *IEEE Network*, Vol. 13, No. 2, March/April 1999.

[21] X. Li, S. Paul, P. Pancha, and M. Ammar, "Layered Video Multicast with Retransmission (LVMR): Evaluation of Error Recovery Schemes", *Proc. NOSSDAV '97*, St. Louis, MO, May 1997.

[22] Y. J. Liang, E. Steinbach, and B. Girod, "Multi-stream Voice Transmission over the Internet Using Path Diversity", *Proc. ACM Multimedia '01*, Ottawa, Canada, Sept./Oct. 2001.

[23] J. Nonnenmacher, E. W. Biersack, and D. Towsley, "Parity-Based Loss Recovery for Reliable Multicast Transmission", *IEEE/ACM Trans. on Networking*, Vol. 6, No. 4, Aug. 1998.

[24] J.-F. Paris, S. W. Carter, and D. D. E. Long, "Efficient Broadcasting Protocols for Video on Demand", *Proc. MASCOTS '98*, Montreal, Canada, July 1998.

[25] J.-F. Paris, S. W. Carter, and D. D. E. Long, "A Hybrid Broadcasting Protocol for Video On Demand", *Proc. MMCN '99*, San Jose, CA, Jan. 1999.

[26] V. Paxson, "End-to-End Internet Packet Dynamics", *IEEE/ACM Trans. on Networking*, Vol. 12, No. 5, June 1999.

[27] C. Perkins, O. Hodson, and V. Hardman, "A Survey of Packet Loss Recovery Techniques for Streaming Audio", *IEEE Network*, Vol. 12, No. 5, Sept./Oct. 1998.

[28] L. Rizzo, "Effective Erasure Codes for Reliable Computer Communication Protocols", *Computer Communication Review*, Vol. 27, No. 2, April 1997.

[29] L. Rizzo and L. Vicisano, "A Reliable Multicast Data Distribution Protocol Based on Software FEC Techniques", *Proc. HPCS '97*, Chalkidiki, Greece, June 1997.

[30] S. Sen, L. Gao, J. Rexford, and D. Towsley, "Optimal Patching Schemes for Efficient Multimedia Streaming", *Proc. NOSSDAV '99*, Basking Ridge, NJ, June 1999.

[31] H. Tan, D. L. Eager, and M. K. Vernon, "Delimiting the Range of Effectiveness of Scalable On-Demand Streaming", *Proc. Performance '02*, Rome, Italy, Sept. 2002.

[32] W. Tan and A. Zakhor, "Multicast Transmission of Scalable Video using Receiver-driven Hierarchical FEC", *Packet Video Workshop*, New York, NY, April 1999.

[33] L. Vicisano, L. Rizzo, and J. Crowcroft, "TCP-like Congestion Control for Layered Video Multicast Data Transfer", *Proc. IEEE Infocom '98*, San Francisco, CA, April 1998.

[34] S. Viswanathan and T. Imielinski, "Metropolitan Area Video-on-Demand Service using Pyramid Broadcasting", *Multimedia Systems*, Vol. 4, No. 4, Aug. 1996.

[35] R. Yavatkar, J. Griffoen, and M. Sudan, "A Reliable Dissemination Protocol for Interactive Collaborative Applications", *Proc. ACM Multimedia '95*, San Francisco, CA, Nov. 1995.

[36] X. R. Xu, A. C. Myers, H. Zhang, and R. Yavatkar, "Resilient Multicast Support for Continuous-Media Applications", *Proc. NOSSDAV '97*, St. Louis, MO, May 1997.

A Multicast Retransmissions

This appendix derives a lower bound on the required server bandwidth when retransmissions are multicast, under the assumption that packet losses at each client are independent and occur with probability p. In an actual system there may be some correlation among the packet losses experienced at different clients due to shared links in the transmission paths to these clients. Simple models of plausible correlation structures may be analyzed using a similar approach as described below for independent packet losses.

As in the analysis for the case of no packet loss reviewed in Section 2.3, consider a small portion of the object at some arbitrary time offset x. For an arbitrary client request that arrives at time t, this portion of the object must be delivered no later than time $t+x$. Thus, there is a "sharing window" of duration at most x over which a multicast of this portion may be fruitfully received by new clients.

A lower bound on the required server bandwidth can be obtained by (a) assuming that closely spaced retransmissions experience the same (rather than correlated) loss probability, and (b) neglecting the impact of the time required for retransmissions on the size of the sharing window and the scheduling of multicasts that are not retransmissions. In the absence of precisely simultaneous client

requests, there is only one client for which any particular multicast of the portion at time offset x is "just in time". Under the assumption of uncorrelated loss probabilities, the number of transmissions required to achieve successful delivery to this client is equal to n with probability $p^{(n-1)}(1-p)$, and has average value $\frac{1}{1-p}$. The probability s that some other client that is listening to these transmissions successfully receives the data is given by

$$s = \sum_{n=1}^{\infty} p^{(n-1)}(1-p)(1-p^n) = \frac{1}{1+p}. \quad (11)$$

If one of these clients does *not* successfully receive the data, another "fresh" multicast must be scheduled. The required server bandwidth is minimized if this new multicast is scheduled so as to achieve "just in time" delivery to the earliest such client.

Thus, each fresh multicast of the portion at time offset x incurs on average $\frac{1}{1-p}$ transmissions. Furthermore, not all clients listening to these transmissions may successfully receive the data, and so the minimal average frequency with which these multicasts must be scheduled is increased in comparison to the case of no packet loss. This minimal frequency is identical to the average throughput of a system in which "customers" (representing fresh multicasts and their associated sharing windows) arrive and reside in the system for constant time duration x. Arrivals occur at rate λ when there are no customers present in the system, at rate $\lambda * (1-s)$ when there is one customer present in the system, at rate $\lambda * (1-s)^2$ when there are two customers present in the system, and so on.

The average throughput in this system is identical to that in a similar system, but with exponentially distributed customer residence times of mean duration x. The average throughput can therefore be computed numerically as the solution of an infinite state one-dimensional Markov chain with transition rate from state i to $i+1$ of $\lambda * (1-s)^i$ and from state $i+1$ to i of $\frac{i+1}{x}$, for all $i \geq 0$. A lower bound on required server bandwidth can then be (numerically) obtained by dividing the object into arbitrarily small portions, and summing over all portions, the size of the portion times $\frac{1}{1-p}$ times the throughput computed from the Markov chain analysis for the corresponding time offset x.

B Asymptotic Analysis of RPB

The results presented in Figure 9 suggest that as the parameter b of the RPB protocols is increased and (for fixed b) the parameter r is decreased, start-up delay approaches that of the lower bound. The following asymptotic analysis of the segment size progression supports this result.

Fix b, and let r grow small and K/s grow large, so that each segment becomes infinitesimally small in length and delivery rate, and so that the start-up delay approaches zero. Further, let $l(x)$ denote the length of the $x\frac{s}{b}$'th segment ($b + \frac{b}{s} \leq x \leq K\frac{b}{s}$). Equation (9) and $r = \frac{b}{s}$ yield

$$l(x) = l_{x\frac{s}{b}} = \frac{1}{a} \sum_{j=(x-b)\frac{s}{b}}^{x\frac{s}{b}-1} l_j \frac{b}{s}. \quad (12)$$

Asymptotically the sum can be written as an integral:

$$l(x) = \frac{1}{a} \int_{x-b}^{x} l(y)dy. \quad (13)$$

As can be verified by substitution, the solution to (13) is

$$l(x) = ue^{vx} \quad (14)$$

where u and v are independent of x, and v is given by

$$v = \frac{1-e^{-vb}}{a}. \quad (15)$$

Thus, asymptotically $l(x) = e^{vb}l(x-b)$, yielding $l_k = e^{vb}l_{k-s}$. Using this equality to substitute for the left-hand side of equation (9) for $k > s$, we get that the sum of the lengths of s consecutive segments $js, js+1, ..., (j+1)s-1$ is asymptotically $\frac{a}{r}e^{vb}l_{js}$. Further, the sum of the lengths of the next set of consecutive s segments $(j+1)s, (j+1)s+1, ..., (j+2)s-1$, is greater by a factor of $e^{vb} > 1$. From this geometric progression we can conclude that if the first segment is normalized to length 1, then the sum of the lengths of all K segments is asymptotically $\mathcal{O}(e^{vb\frac{K}{s}})$. Thus, since the start-up delay d is equal to a times the length of the first segment, and using $b = sr$, we have that d is $\mathcal{O}(ae^{-vrK})$. Solving for the server bandwidth rK yields a required server bandwidth of

$$\frac{1}{v}ln(\frac{1}{d}) + lower\ order\ terms \quad (16)$$

For large b, $\frac{1}{v}$ tends to a. If a is chosen equal to $\frac{1}{1-p}$ (the smallest a for which full recovery from loss can be possible without extra delay at the clients, given loss rate p), we obtain a required server bandwidth that is asymptotically the same as the lower bound given in equation (4).

Interestingly, for $a = 1$ (i.e., the no packet loss case), and general b, we get $v = 1 - e^{-vb}$, implying that $\frac{1}{v}$ is identical to the constant $\eta_{b,\epsilon}$ defined in [10]. This implies that the asymptotic required server bandwidth for general b (and small r), in this no packet loss case, is identical to the conjectured asymptotic lower bound on required server bandwidth for general b from [10].