Multicast Protocols for Scalable On-Demand Download*

Niklas Carlsson Derek L. Eager
Department of Computer Science
University of Saskatchewan
Saskatoon, SK S7N 5C9, Canada
carlsson@cs.usask.ca, eager@cs.usask.ca

Mary K. Vernon
Computer Science Department
University of Wisconsin-Madison
Madison, WI 53706, USA
vernon@cs.wisc.edu

Abstract

Previous scalable protocols for downloading large, popular files from a single server include batching and cyclic multicast. With batching, clients wait to begin receiving a requested file until the beginning of its next multicast transmission, which collectively serves all of the waiting clients that have accumulated up to that point. With cyclic multicast, the file data is cyclically transmitted on a multicast channel. Clients can begin listening to the channel at an arbitrary point in time, and continue listening until all of the file data has been received.

This paper first develops lower bounds on the average and maximum client delay for completely downloading a file, as functions of the average server bandwidth used to serve requests for that file, for systems with homogeneous clients. The results show that neither cyclic multicast nor batching consistently yields performance close to optimal. New hybrid download protocols are proposed that achieve within 15% of the optimal maximum delay and 20% of the optimal average delay in homogeneous systems.

For heterogeneous systems in which clients have widely-varying achievable reception rates, an additional design question concerns the use of high rate transmissions, which can decrease delay for clients that can receive at such rates, in addition to low rate transmissions that can be received by all clients. A new scalable download protocol for such systems is proposed, and its performance is compared to that of alternative protocols as well as to new lower bounds on maximum client delay. The new protocol achieves within 25% of the optimal maximum client delay in all scenarios considered.

Keywords: Scalable download, multicast protocols, required server bandwidth

1. Introduction

Large, popular files can be efficiently delivered from a single server system to potentially large numbers of concurrent clients using *scalable download* protocols based on multicast (IP or application-level) or broadcast. Existing scalable download protocols include *batching* [11, 24] and *cyclic multicast* [5, 17]. With batching, clients wait to begin receiving a requested file until the beginning of its next multicast (or broadcast) transmission, which collectively serves all of the waiting clients that have accumulated up to that point. With cyclic multicast, the file data is cyclically transmitted on a multicast channel (e.g., a multicast group) which clients begin listening to at an arbitrary point in time, and continue listening to until all of the file data has been received.

Note that with batching, clients that request a file while a file multicast is in progress do not begin receiving the data currently being transmitted, but instead wait until the beginning of the next multicast. This strategy has the advantage of providing in-order data delivery, but the disadvantage of not fully utilizing the potential sharing of multicast transmissions. With cyclic multicast, in contrast, clients can begin receiving file data immediately. However, transmissions are not limited to times when there are (or are likely to be) relatively large numbers of listeners, as with batching.

---

* To appear in *Performance Evaluation*. This work was partially supported by the Natural Sciences and Engineering Research Council of Canada, and by the National Science Foundation under grants ANI-0117810, CNS-0435437 and EIA-0127857.

This paper considers the problem of devising protocols that minimize the average or maximum client delay for downloading a single file, as a function of the average server bandwidth used for delivery of that file. An equivalent problem is to minimize the average server bandwidth required to achieve a given average or maximum client delay, and sometimes we adopt this equivalent perspective instead. Although we do not explicitly consider delivery of multiple files, note that use of a download protocol that minimizes the average server bandwidth for delivery of each file will minimize the average total required server bandwidth for delivering all files, as well.

We focus first on systems with homogeneous clients that have identical reception rate constraints and develop lower bounds on the average and maximum client delay for downloading a file, as functions of the average server bandwidth used for delivering that file. We define optimized batching and cyclic multicast protocols, and find that each of these protocols is significantly suboptimal over some region of the system design space. For example, the cyclic multicast protocol provides near-optimal maximum client delay when the client reception bandwidth is low relative to the file request rate, but can have maximum client delay up to 80% higher than optimal otherwise. An optimized batching protocol provides near-optimal average client delay when the client reception bandwidth is high relative to the file request rate, but can have average client delay up to 50% higher than optimal otherwise. Motivated by these results, Section 5 develops new practical hybrid protocols that largely close these gaps. The new protocols achieve within 15% of the optimal maximum delay and 20% of the optimal average delay, in homogeneous systems.

We then consider protocols for delivery of a file to heterogeneous clients that have widely varying achievable reception rates. In this context, achieving efficient delivery as well as lower delay for higher rate clients requires use of multiple multicast channels. Each client listens to the number of channels corresponding to its achievable reception rate. The key challenge is to achieve a close-to-optimal compromise between high rate transmissions (in aggregate, over all channels used for a file), which enable lower delays for clients that can receive at such rates, and low rate transmissions that allow maximal sharing. A protocol for delivery to heterogeneous clients is proposed that yields maximum client delays that are within 25% of optimal in the scenarios considered.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 defines and analyzes the optimized batching and cyclic multicast protocols. In this section, as in the subsequent two sections, we assume homogeneous clients. Lower bounds on the average and maximum client delay for downloading a single file, for given average server bandwidth usage (or, equivalently, on the average server bandwidth required to achieve a given average or maximum client delay) are derived in Section 4. Section 5 develops new scalable download protocols that achieve close to optimal performance. Protocols for delivery to heterogeneous clients are developed and evaluated in Section 6. Conclusions are presented in Section 7.

2. Related Work

Considerable prior work has concerned the problem of scheduling one or more broadcast channels that serve a collection of small, fixed length objects using a batching approach [11, 24]. The main problem considered is that of determining which object should be transmitted on the channel (or channels) at each point in time, so as to minimize the average client delay. Both *push based* [6, 14, 1] and *pull based* [11, 24, 4] protocols have been proposed. Hybrid approaches that combine push and pull are also possible [2, 19]. Push based protocols determine a transmission schedule based only on average object access frequencies, in which case a periodic delivery schedule is optimal [6]. Pull based protocols assume knowledge of the currently outstanding client requests. Candidate scheduling policies for determining the object to transmit next include *first come first serve (FCFS)*, *most requests first (MRF)*, and *longest wait first (LWF)* [11, 24]. The criteria used by the former two policies are combined in the *RxW* policy, proposed by Aksoy and Franklin [4]. This policy uses the product of the number of pending requests (R) for each object and the longest waiting time of these pending requests (W), when deciding which object to transmit next. Other work has investigated batching protocols for streaming of video rather than download [3, 10, 21], including for example an earlier proposal of a policy very similar to *RxW* [3].

In contrast to the previous work on scalable download using batching, we consider download of large files and protocols in which new clients can begin listening to an on-going multicast rather than waiting until the beginning of the next multicast. Furthermore, we consider contexts in which the total server bandwidth devoted to file download is somewhat elastic, and thus consider the download protocol for a given file that will minimize the average or maximum client delay for a given *average* bandwidth used for delivery of the file. We note that in a given server setting, the best parameterization of the near-optimal protocol will depend on the current server load and the actual upper bound on total server bandwidth.

Prior work on scalable download of large files from a single server has focused on cyclic multicast, in which a file's data is cyclically transmitted on a multicast/broadcast channel [15, 5, 7, 22, 9, 17, 8]. Each requesting client can begin listening to the channel at an arbitrary point in time, and continues listening until all of the file data has been received. This prior work has focused on the performance benefits that cyclic delivery offers in comparison to unicast delivery, the accommodation of packet loss through use of erasure coding, and support for heterogeneous clients. Erasure coding enables a client to recover from packet loss simply by continuing to listen to the channel until an amount of erasure-coded data equal to the size of the requested file (or possibly slightly greater, depending on the encoding scheme) has been successfully received, at which point the file can be reconstructed [16, 9, 18]. Heterogeneous clients can be supported through the delivery of file data on multiple channels. Each client listens to the subset of channels appropriate to its achievable reception rate. By careful selection of the order in which data blocks are transmitted on each channel [7, 8], or use of erasure codes with long "stretch factors" [18], receptions of the same data block on different channels can be reduced or eliminated. In contrast to this prior work on cyclic multicast, we focus on the performance comparison between batching and cyclic multicast, and the design of hybrid protocols that combine elements of both approaches to achieve superior performance.

There has been some prior work on hybrid protocols that combine batching and cyclic multicast, specifically the work by Wolf et al. [23]. The authors find that their proposed hybrid algorithms yield better performance than pure batching protocols. We similarly find hybrid protocols to yield better performance. However, the focus in the work by Wolf et al. is on delivery of digital products using otherwise unused bandwidth in a broadcast television system. They assume a fixed schedule of broadcast channel availability and fixed delivery deadlines with associated delivery payments. In contrast, we assume complete flexibility in when transmissions occur, and develop protocols that achieve near-optimal average or maximum client delay as a function of the average required server bandwidth.

3. Baseline Protocols

This section defines and analyzes simple "baseline" batching and cyclic multicast protocols for delivery of a single file, assuming homogeneous clients. The metrics of interest are the average client delay (i.e., download time), the maximum client delay in cases where such a maximum exists, and the average server bandwidth used for the file data multicasts. It is assumed throughout the paper that each requesting client receives the entire file; i.e., clients never balk while waiting for service to begin or after having received only a portion of the file. Our analysis and protocols are compatible with erasure-coded data. Each client is assumed to have successfully received the file once it has listened to multicasts of an amount of data L (termed the "file size" in the following, although with packet loss and erasure coding, L may exceed the true file size). Poisson request arrivals are assumed unless otherwise specified. Generalizations are discussed in some cases. We note that Poisson arrivals can be expected for independent requests from large numbers of clients. Furthermore, multicast delivery protocols that have high performance for Poisson arrivals, have even better performance under the more bursty arrival processes that are typically found in contexts where client requests are not independent [12].

3.1 Batching

Consider first batching protocols in which the server periodically multicasts the file to those clients that have requested it since it was last multicast. Any client whose request arrives while a multicast is in progress, simply waits until the next multicast begins.

Table 1: Notation

Symbol	Definition
λ	File request rate
L	File size
b	Maximum sustainable client reception rate
r	Transmission rate on a multicast channel ($r \le b$)
B	Average server bandwidth
A	Average client delay (time from file request, until file is completely received)
D	Maximum client delay
Δ, n, f	Batching delay parameters

Perhaps the simplest batching protocol is to begin a new multicast of the file every t time units for some constant t. However, this protocol has the disadvantage that multicasts may sometimes serve no or only a few clients.

Two optimized batching protocols are considered here. The first, termed *batching/constant batching delay* (*batching/cbd*), achieves the minimum average server bandwidth for a given maximum client delay, or equivalently the minimum value of maximum client delay for a given average server bandwidth, over the class of batching protocols as defined above. Letting T denote the time at which some file multicast begins and a denote the duration of the time interval from T until the next request arrival, the server will begin the next multicast at time $T+a+\Delta$, where Δ is a parameter of the protocol. Thus, using the notation defined in Table 1, the average time between file multicasts is $\Delta+1/\lambda$, the average server bandwidth is $L/(\Delta+1/\lambda)$, and the maximum client delay is Δ plus L/r (the file transmission time). With respect to the average client delay, note that the client whose request arrival triggers the scheduling of a new multicast experiences the maximum waiting time Δ until the multicast begins. All clients whose requests arrive during the batching delay Δ will share reception of this multicast. On average, there will be $\lambda\Delta$ such clients, and the average waiting time until the multicast begins for such a client will be $\Delta/2$. In summary,[1]

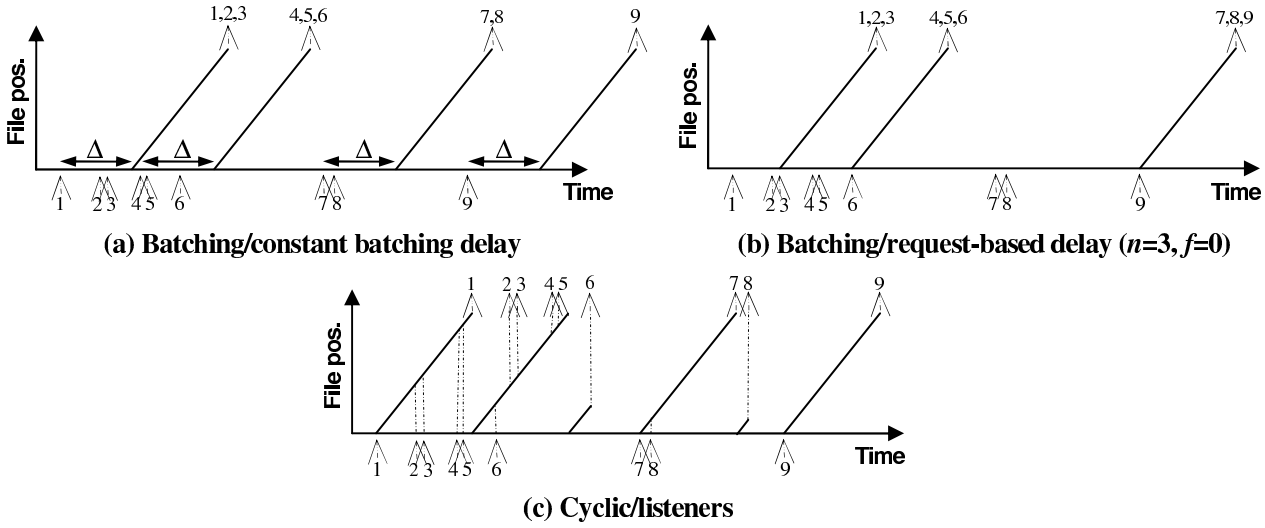
$$B_{b/cbd} = \frac{L}{\Delta + 1/\lambda}; \qquad A_{b/cbd} = \frac{\Delta(1+\lambda\Delta/2)}{1+\lambda\Delta} + L/r; \qquad D_{b/cbd} = \Delta + L/r.$$

The second optimized batching protocol, termed *batching/request-based delay* (*batching/rbd*), achieves the minimum value of average client delay for a given average server bandwidth, over the class of batching protocols as defined above.[2] The basic idea is to make the batching delay some integral number of request inter-arrival times. To make it possible to achieve arbitrary average server bandwidth values, the protocol is defined such that the server waits for $n+1$ requests for a fraction f of its multicasts, and for n requests for the remaining fraction $1-f$, where n and f are protocol parameters (integer $n \ge 1$, $0 \le f < 1$).[3] Thus, the average time between file multicasts is $(n+f)/\lambda$, and the average server bandwidth is $L/((n+f)/\lambda)$. The average client delay can be derived from the fact that each multicast serves n clients plus with probability f one additional client, and the i'th last of these clients experiences an average waiting time until the multicast begins of $(i-1)/\lambda$. Note that the maximum client delay is unbounded with this protocol. Thus,

[1] In the non-Poisson case, assuming request interarrival times are independent and identically distributed (IID), these performance metrics can be obtained by calculating conditional expectations. For example, note that $1/\lambda$ in the bandwidth expression can be replaced with the expected time from after the initiation of a transmission until the next request, conditioned on the fact that there was a request arrival time Δ in the past.

[2] This can be established formally using an argument similar to that used for the lower bound on average server bandwidth in Section 4.1.

[3] When arrivals are Poisson, inter-arrival times are memoryless, and the method by which the server determines when to wait for n versus $n+1$ arrivals (for fixed f) has no impact on average server bandwidth usage or average delay.



(a) Batching/constant batching delay

(b) Batching/request-based delay (n=3, f=0)

(c) Cyclic/listeners

Figure 1: Operation of the Baseline Protocols for an Example Request Sequence

$$B_{b/rbd} = \frac{L}{(n+f)/\lambda}; \qquad A_{b/rbd} = \frac{\frac{n(n-1)}{2\lambda} + f\frac{n}{\lambda}}{n+f} + L/r = \frac{n(n+2f-1)}{2\lambda(n+f)} + L/r; \qquad D_{b/rbd} \text{ is unbounded.}$$

Note that for both of these batching protocols, the value of the multicast transmission rate r that minimizes average and maximum client delay is equal to the maximum sustainable client reception rate b.

Figure 1 illustrates the operations of these two batching protocols, as well as the cyclic multicast protocol discussed in the next section, for an example sequence of requests. Requests are numbered and the arrival times and service completion times of the requests are indicated by the arrows at the bottom and top of each subfigure, respectively. The solid, slanted lines denote multicast transmissions, each of which, in the case of the batching protocols, delivers the entire file. For the *batching/cbd* protocol, the batching delays (each of duration Δ) are indicated with double arrows along the horizontal (time) axis.

3.2 Cyclic Multicast

Perhaps the simplest cyclic multicast protocol is to continually multicast file data at a fixed rate r (cycling back to the beginning of the file when the end is reached) on a single multicast channel, regardless of whether or not there are any clients listening. Here we consider a more efficient cyclic multicast protocol, *cyclic/listeners (cyclic/l)*, that assumes that the server can determine whether there is at least one client with an unfulfilled request for the file, and transmit only if there is. Since the server transmits whenever there is at least one client, the delay experienced by each client is just the file transmission time, L/r. The average server bandwidth can be derived by noting that there will be at least one client listening on the multicast channel at an arbitrary point in time T, if and only if at least one request for the file was made during the time interval $[T-L/r, T]$, and that the probability of at least one request arrival during an interval of duration L/r is $1-e^{-\lambda L/r}$ for Poisson arrivals at rate λ.[4] This yields

$$B_{c/l} = r\left(1-e^{-\lambda L/r}\right); \qquad A_{c/l} = D_{c/l} = L/r\,.$$

Note that the transmission rate r is the only protocol parameter, and by itself determines the tradeoff between server bandwidth usage, and client delay.

[4] Note that the performance of this protocol can be analyzed for any arrival process for which it is possible to compute the probability of there being at least one request arrival during a randomly chosen time period of duration L/r.

4. Lower Bounds

Making the same assumptions as in Section 3 of homogeneous clients, full-file delivery, and Poisson client request arrivals, this section derives fundamental performance limits for scalable download protocols. These limits depend on the maximum sustainable client reception rate. Note that for batching protocols, for example, if the server transmission rate is increased the batching delay can be increased without increasing the total client delay, thus providing a longer period over which aggregation of requests can occur and more efficient use of server bandwidth. Section 4.1 considers the limiting case in which clients can receive data at arbitrarily high rate, for which there is a previously derived bound on maximum delay [20]. Section 4.2 considers the realistic case in which there is an upper bound b on client reception rate.

4.1 Unconstrained Client Reception Rate

Consider first the maximum client delay, and the average server bandwidth required to achieve that delay. From Tan et al. [20],[5]

$$B \geq \frac{L}{D+1/\lambda} \quad \Leftrightarrow \quad D \geq L/B - 1/\lambda .$$

This bound is achieved in the limit, as the server transmission rate tends to infinity, by a protocol in which the server multicasts the file to all waiting clients whenever the waiting time of the client that has been waiting the longest reaches D.

Consider now the problem of optimizing for average client delay. At each point in time an optimal protocol able to transmit at infinite rate would either not transmit any data, or would transmit the entire file. To see this, suppose that some portion of the file is transmitted at an earlier point in time than the remainder of the file. Since client requests might arrive between when the first portion of the file is transmitted and when the remainder is transmitted, it would be more efficient to wait and transmit the first portion at the same time as the remainder. Optimizing for average client delay requires determining the spacings between infinite rate full file transmissions that are optimal for this metric. With Poisson arrivals and an on-line optimal protocol, (1) file transmissions occur only on request arrivals, and (2) each multicast must serve either n or $n+1$ clients for some integer $n \geq 1$. With respect to this latter property, consider a scenario in which the file is multicast to n waiting clients on one occasion and to $n+k$ clients for $k \geq 2$ on another. A lower average delay could be achieved, with the same average spacing between transmissions, by delaying the first multicast until there are $n+1$ waiting clients, and making the second multicast at the request arrival instant of the $n+k-1^{th}$ client instead of the $n+k^{th}$.

Thus, a lower bound on the average server bandwidth B required to achieve a given average client delay A can be derived by finding an integer $n \geq 1$, and value f $(0 \leq f < 1)$, such that

$$A = \frac{\frac{n(n-1)}{2\lambda} + f\frac{n}{\lambda}}{n+f} = \frac{n(n+2f-1)}{2\lambda(n+f)},$$

in which case

$$B \geq \frac{L}{(n+f)/\lambda} .$$

Equivalently, to determine a lower bound on the average delay A that can be achieved with average server bandwidth B, let $n = \max[1, \lfloor \lambda L/B \rfloor]$, and $f = \max[0, \lambda L/B - n]$. Then,

---

[5] As with the bandwidth expression for *batching/cbd* in Section 3.1, for the case of non-Poisson request arrivals with IID request interarrival times the $1/\lambda$ term can be replaced by the appropriate conditional expectation. Further note that a bandwidth lower bound can be obtained for any process such that this quantity can be bounded from above, as has been noted in the scalable streaming context [13].

$$A \geq \frac{n(n+2f-1)}{2\lambda(n+f)}.$$

Note that for $B < \lambda L$ (the bandwidth required for unicast delivery), the optimal protocols for minimizing the average delay A and the maximum delay D are different, and thus the lower bounds on A and D cannot be achieved simultaneously. In fact, for all $B < \lambda L$ the optimal protocol for average delay has unbounded maximum delay. If $\lambda L/B$ is an integer greater than one, the lower bound on A is exactly half the lower bound on D; otherwise, it is somewhat greater than half. In particular, as B tends to λL, the ratio of the lower bounds on A and D tends to one.

4.2 Constrained Client Reception Rate

Assume now that clients have a finite maximum sustainable reception rate b. In this case, both the maximum and average delay must be at least L/b. To achieve the minimal values $D = A = L/b$, each client must receive the file at maximum rate starting immediately upon its request. The *cyclic/l* protocol defined in Section 3.2 achieves the lowest possible server bandwidth usage in this case, as the transmission rate of the server is (only) b whenever there is at least one active client, and zero otherwise. Thus, for $D = A = L/b$, we have the bound $B \geq b\left(1 - e^{-\lambda L/b}\right)$.

More generally, for a specified maximum delay $D \geq L/b$, the average server bandwidth is minimized by the *send as late as possible* (*slp*) protocol, in which the server cyclically multicasts file data at rate b whenever there is at least one active client that has no "slack" (i.e., for which transmission can no longer be postponed). Such a client must receive data continuously at rate b until it has received the entire file, if it is to avoid exceeding the delay bound. Note that although this protocol is optimal for maximum delay, it requires that the server maintain information on the remaining service requirements and request completion times of all outstanding requests. Furthermore, the *slp* protocol can result in extremely fragmented transmission schedules. This motivates simpler and more practical near-optimal protocols such as that devised in Section 5.1.

An accurate approximation for the average server bandwidth with the *slp* protocol is given by

$$B_{slp} \approx \left(\frac{(e^{\lambda L/b} - 1)/\lambda + D - L/b}{e^{\lambda L/b}/\lambda + D - L/b} \right) \frac{L}{D}.$$

Here the L/D factor approximates the average server bandwidth usage over those periods of time during which there is at least one active client (i.e., client with an outstanding request). The factor in brackets approximates the fraction of time that this condition holds. This fraction is equal to the average duration of a period during which there is at least one active client, divided by the sum of this average duration and the average request inter-arrival time ($1/\lambda$). The average duration of a period during which there is at least one active client is approximated by the average duration of an $M/G/\infty$ busy period with arrival rate λ and service time L/b, as given by $(e^{\lambda L/b} - 1)/\lambda$, plus the duration of the delay after the arrival of a request to a system with no active clients until the server must begin transmitting ($D - L/b$). Note that a corresponding approximation for the minimum achievable maximum delay, for given average server bandwidth, can be obtained by solving for D in the above approximation.

Exhaustive comparisons against simulation results indicate that the above approximation is very accurate, with relative errors under 4%, and thus we use the approximation rather than simulation values in the remainder of the paper.[6] Figure 2 summarizes the validation results, showing contours of equal error over a two dimensional space. Negative and positive errors correspond to underestimations and overestimations of the true values as obtained from simulation, respectively. Without loss of generality, the unit of data volume is chosen to be the file, and the unit of time is chosen to be the time required to download the file at the maximum sustainable client

[6] All of our simulations make the same system and workload assumptions as the analytic models (including the assumption of Poisson arrivals). Note that where we have both simulation and analytic results, the purpose of the simulation is to assess the accuracy of the approximations made in the analysis, and not for validation of the system or workload assumptions.

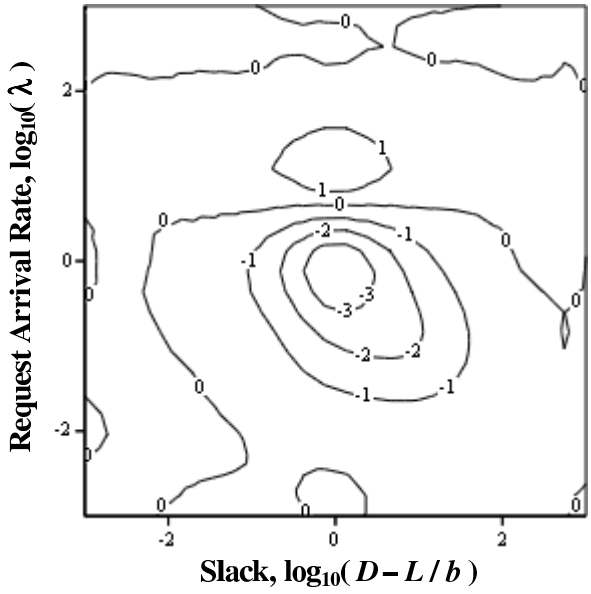


Figure 2: Lower Bound Approximation (% relative error contours; unit of data volume is the file, unit of time is the time required to download the file at maximum rate: i.e., *L*=1, *b*=1)

reception rate. With these choices of units, L and b are each equal to one. The only two remaining parameters are λ and D. The logarithm of the arrival rate λ is used on the vertical axis of the contour plot, covering six orders of magnitude of arrival rates, while six orders of magnitude of "slack" are covered on the horizontal axis using the logarithm of $D-L/b$. As can be seen directly from the approximation, this expression is exact for the boundary cases of $\lambda \to 0$ (minimum λ), $\lambda \to \infty$ (maximum λ), $D \to \infty$ (maximum D), $L \to 0$ (minimum L), $b \to \infty$ (maximum b), and $D = L/b$ (minimum D, or maximum L, or minimum b), holding the other parameters fixed in each case. For example, note that for $b \to \infty$ the approximation reduces to $L/(D+1/\lambda)$, and for $D = L/b$ the approximation reduces to $b\left(1-e^{-\lambda L/b}\right)$.

The optimal scalable download protocol for *average* delay, under a reception rate constraint, appears to be very difficult to determine in general. However, we can derive a lower bound as follows. As noted previously, for $A=L/b$ the optimal protocol is *cyclic/l* as defined in Section 3.2, with $r = b$. Furthermore, a variant of cyclic multicast in which the server sometimes or always waits until a second request arrival before beginning transmission will also be optimal, for values of average delay and bandwidth that can be achieved by this protocol, since each unit of additional channel idle time is achieved by delaying the minimum possible number of clients (only one). Letting f denote the fraction of idle periods in which channel transmission does not begin until a second request arrives, the server bandwidth and average delay under this *cyclic/wait for second, listeners* (*cyclic/w2,l*) protocol are given by

$$B_{c/w2,l} = b\frac{\left(e^{\lambda L/b}-1\right)/\lambda}{\left(e^{\lambda L/b}-1\right)/\lambda+(1+f)/\lambda} = b\frac{e^{\lambda L/b}-1}{e^{\lambda L/b}+f}; \qquad A_{c/w2,l} = \frac{f/\lambda}{\lambda\left((e^{\lambda L/b}-1)/\lambda+(1+f)/\lambda\right)}+L/b = \frac{f/\lambda}{e^{\lambda L/b}+f}+L/b.$$

Note here that $(e^{\lambda L/b}-1)/\lambda$ is the average duration of an $M/G/\infty$ busy period with arrival rate λ and service time L/b, and $(1+f)/\lambda$ is the average duration of a channel idle period. For server bandwidth values B that can be achieved with this protocol, we have (from solving for f in terms of B and then substituting into the average delay expression),

$$A \geq \max\left[0, \frac{\left(e^{\lambda L/b} - 1\right)b/B - e^{\lambda L/b}}{\lambda\left(e^{\lambda L/b} - 1\right)b/B}\right] + L/b\,.$$

Equivalently, to determine the lower bound on the average server bandwidth B that can be achieved with average delay A, solving for f in terms of A and substituting into the average server bandwidth equation yields

$$B \geq \max\left[0, b\left(1 - e^{-\lambda L/b}\right)\frac{1/\lambda - (A - L/b)}{1/\lambda}\right].$$

Values of B that are smaller (or values of A that are larger) than those achieved for $f = 1$ are not achievable by the *cyclic/w2,l* protocol, because in this protocol each idle period always ends no later than the time of the second request arrival. However, the above bounds are valid (although unachievable) for those smaller values of B (and larger values of A) that can be obtained by substituting values greater than one for the parameter f in the above expressions. The bounds are valid in this case because even for $f > 1$, these expressions still assume that the minimum number of clients is delayed (i.e., only one) before the server begins transmission. The bounds are unachievable since the average duration of this delay is assumed to be f/λ, which for $f > 1$ is greater than the average delay until the second request arrival.

A second lower bound on average delay can be derived as follows. First, note that in an optimal protocol, data transmission will always occur at rate b, since: (1) each client can receive at rate at most b, and (2) the average delay cannot increase when a period of length l between request completions during which the transmission rate is less than b, is replaced by an idle period followed by a period of transmission at rate b (of combined length l and equal total server bandwidth usage).

Suppose now that each request arrival that occurs during a busy period is shifted earlier, so that it occurs at a multiple (possibly zero) of $L/(2b)$ from the start of the busy period. As a result of this shifting, requests arriving during a busy period will have greater likelihood of completing service before the busy period ends, for a fixed busy period duration. Therefore, average delay cannot increase. It is now possible to determine the optimal protocol, assuming this shift of request arrivals, based on the following three observations: (1) by the same arguments as in Section 4.1, in the optimal protocol each idle period must end once n, or $n+1$ with some probability f, requests have been accumulated, for some integer $n \geq 1$ and $0 \leq f < 1$; (2) each busy period must end on a request completion, and therefore in the optimal protocol be of total length equal to a multiple (at least two) of $L/(2b)$; and (3) since the state of the system at each multiple of $L/(2b)$ within a busy period is entirely captured by the number of request arrivals that occurred within the previous $L/(2b)$ (all of whose respective clients have been listening to the channel for exactly time $L/(2b)$, owing to the shifting), there is an integer threshold $k \geq 1$ such that if the number of such arrivals is less than k, the server will stop transmitting in the optimal protocol (thus ending the busy period), and otherwise it will not. Note that these observations uniquely specify the operation of the optimal protocol, by establishing the criteria used for determining when to start a transmission, specifying the possible instances when a transmission can be completed, and for each of these time instances specifying the criteria used to determine if the transmission should be stopped.

Given values for the parameters n, f, and k, the average server bandwidth and the average client delay with this (unrealizable) *shifted arrivals* (*sa*) protocol are given by

$$B_{sa} = b\frac{(1+1/p)L/(2b)}{(1+1/p)L/(2b) + \left(n + f - \sum_{i=0}^{k-1} i\frac{p_i}{p}\right)\Big/\lambda}, \quad A_{sa} = \frac{L}{b} + \frac{\sum_{i=0}^{k-1}\frac{p_i}{p}\left[i(n-i)\frac{1}{\lambda} + \frac{(n-i)(n-i-1)}{2\lambda} + \frac{n}{\lambda}f\right]}{\lambda\left((1+1/p)L/(2b) + \left(n + f - \sum_{i=0}^{k-1} i\frac{p_i}{p}\right)\Big/\lambda\right)},$$

where $p_i = \frac{1}{i!}(\lambda L/(2b))^i e^{-\lambda L/(2b)}$ is the probability of i request arrivals in time $L/(2b)$, and $p = \sum_{i=0}^{k-1} p_i$ is the probability of a busy period ending when its duration reaches a multiple of $L/(2b)$ (and at least L/b). B_{sa} is given by the ratio of the average duration of a busy period to the sum of the average durations of a busy period and an

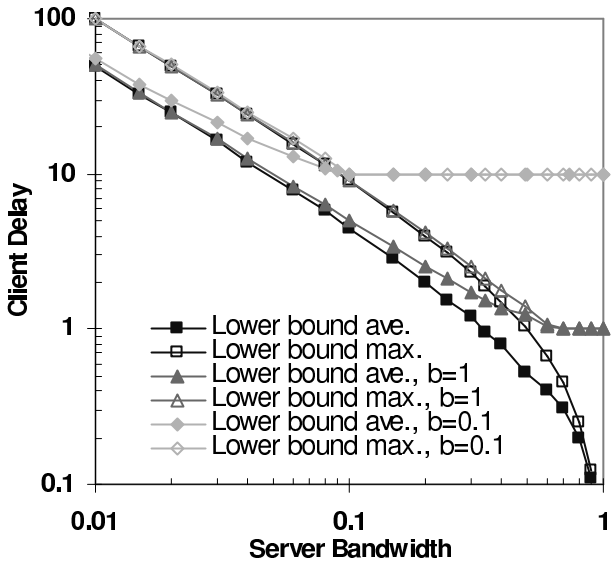


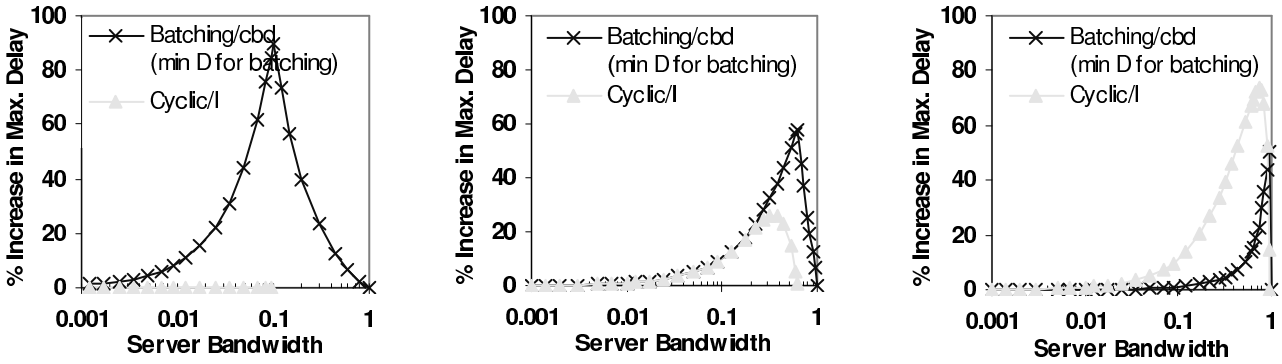
Figure 3: Lower Bounds on Client Delay (unit of data volume is the file, unit of time is the average time between requests: i.e., L=1, λ=1)

idle period, times the transmission rate b. Note here that when the busy period ends owing to having $i < k$ request arrivals during the previous $L/(2b)$, the average duration of the idle period will be $(n+f-i)/\lambda$, since only $n-i$ (or $n+1-i$) new requests need be received to obtain a total of n (or $n+1$) unsatisfied requests. A_{sa} is equal to the total expected idle time incurred by those clients making requests during a busy period and the following idle period, divided by the expected number of such requests, plus the time required to download the file data (L/b). The optimal n, f, and k values for a particular server bandwidth or average client delay can be found numerically, so as to obtain a lower bound on average delay or server bandwidth, respectively. This bound can then be combined with the corresponding bound from the *cyclic/w2,1* protocol analysis, to yield a single lower bound, by taking the maximum of the two.

4.3 Lower Bound Comparisons

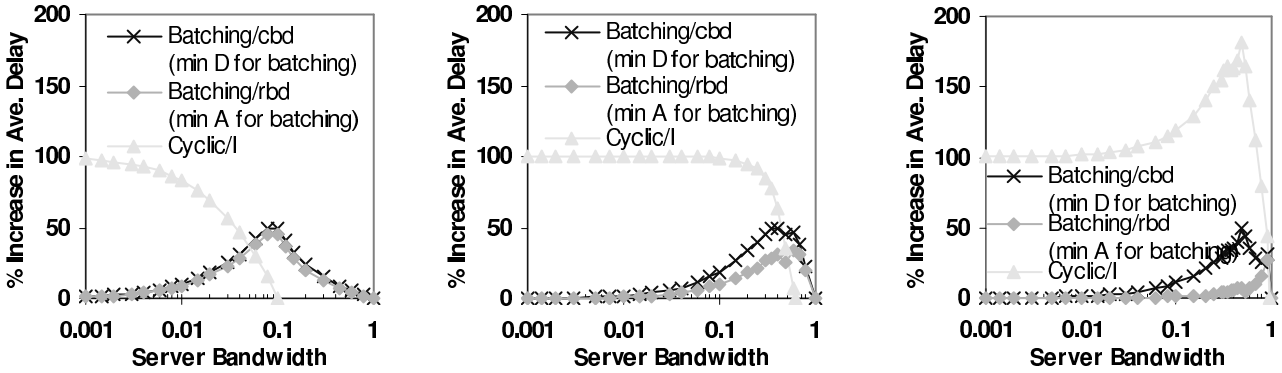
Figure 3 shows the lower bounds on average and maximum client delay for the case of unconstrained client reception rates and for $b = 1$ and $b = 0.1$. Without loss of generality, the unit of data volume is chosen to be the file and the unit of time is chosen to be the average time between requests. With these choices of units, $L = 1$, $\lambda = 1$, client delay is expressed as a normalized value in units of the average time between requests, average server bandwidth is expressed as a normalized value in units of file transmissions per average time between requests, and the maximum sustainable client reception rate is expressed as a normalized value in units of file receptions per average time between requests. These units are used in all figures comparing homogenous client protocols (Sections 4 and 5). Note that the average server bandwidth B in these units can be interpreted as the fraction of the average bandwidth required for unicast delivery, so the region of interest in the design of scalable multicast protocols corresponds to values of B considerably less than one.

Although our choice of data volume and time units correctly reflects the fact that it is server bandwidth and client reception rate *relative* to request rate and file size that determines performance, some care is required in interpreting the resulting figures. Consider, for example, Figure 3, and a scenario in which the client request rate decreases for fixed average server bandwidth (when expressed in unnormalized units). With our chosen units λ remains equal to one in this scenario (since the unit of time is the average time between requests), but B (expressed in units of file transmissions per average time between requests) increases proportionally to the decrease in the client request rate. Thus, in Figure 3, the increasing value of the normalized server bandwidth B



(a) $b = 0.1$ (b) $b = 1.0$ (c) $b = 10.0$

Figure 4: Maximum Delay with Baseline Protocols Relative to Lower Bound ($L = 1, \lambda = 1$)



(a) $b = 0.1$ (b) $b = 1.0$ (c) $b = 10.0$

Figure 5: Average Delay with Baseline Protocols Relative to Lower Bound ($L = 1, \lambda = 1$)

as one moves from left to right on the horizontal axis can correspond to increasing server bandwidth (with a fixed client request rate) or decreasing client request rate (with a fixed server bandwidth). Similar considerations apply with respect to the normalized maximum sustainable client reception rate b.

Perhaps the main observation from Figure 3 is that client reception rate constraints can strongly impact the achievable performance, although this impact diminishes as the value of the normalized average server bandwidth B decreases. Note also that the difference between the average and maximum delay bounds decreases with increasing server bandwidth. The point where these bounds become identical is the point at which each client experiences only the minimum delay of L/b.

Figure 4 plots the percentage increases in the maximum client delay for the baseline batching and cyclic multicast protocols in comparison to the lower bound, for three different values of client reception rate. Figure 5 plots the corresponding percentage increases in average client delay for the baseline protocols. The system measures are expressed in the same normalized units as in Figure 3. Note that the average server bandwidth with *cyclic/l* cannot exceed b times the fraction of time that there is at least one active client, and thus the rightmost point of each *cyclic/l* curve is for server bandwidth of less than one.

Figures 4 and 5 show that the batching protocols are close to optimal for small (normalized) server bandwidths, when many requests are accumulated before the next transmission takes place, and for server bandwidths approaching one, when most clients are served individually with minimal delay of L/b. Batching can be significantly suboptimal for intermediate server bandwidth values, however, particularly for maximum client delay (for example, in Figure 4(a), $b = 0.1$ and B between 0.05 and 0.2). Note also that the overall relative performance of batching degrades as the maximum sustainable client reception rate decreases, since in this case

the required duration of a multicast increases, and with the batching protocols new clients are not able to begin listening to a multicast after it has commenced.

In contrast, the performance of *cyclic/l* improves for decreasing client reception rate. However, *cyclic/l* is substantially suboptimal for average client delay over most of the parameter space, and for maximum delay when the client reception rate is high and the server bandwidth is also high although not approaching one (i.e., in Figure 4(c), b = 10.0 and B between 0.4 and 0.9). Note that for small and intermediate server bandwidths, *cyclic/l* is close to optimal for maximum client delay, but since the optimal average client delay is approximately half the optimal maximum client delay in this case, the average client delay with *cyclic/l* is about 100% higher than optimal.

5. Near-Optimal Protocols

Figures 4 and 5 suggest that there is substantial room for improvement over the baseline batching and cyclic multicast protocols, since for each of maximum and average client delay there is a region of the parameter space over which each protocol is substantially suboptimal. The main weakness of the batching protocols is that clients that make requests while a multicast is already in progress do not listen to this multicast. All clients receive the file data "in-order", waiting until the beginning of the next multicast before beginning their downloads. With the baseline cyclic multicast protocol, on the other hand, clients can begin receiving data at arbitrary points in time within an on-going multicast. Since the server transmits whenever there is at least one active client, however, there will be periods over which transmissions serve relatively few clients.

Clearly, an improved protocol should allow clients to begin listening to an on-going multicast at the times of their requests, but should also allow server transmissions to be delayed so as to increase the actual or expected number of clients each serves. It is straightforward to apply a batching-like rule for deciding when a cyclic multicast transmission should commence; the key to devising a near-optimal protocol is determining the conditions under which a multicast should be continued, or terminated. Section 5.1 develops and analyzes new protocols that focus on improving maximum client delay, while Section 5.2 develops and analyzes protocols whose focus is improved average client delay. As in Sections 3 and 4, we assume homogeneous clients, full-file delivery, and Poisson arrivals. Section 5.3 relaxes the Poisson assumption, and considers the worst-case performance of the protocols under arbitrary arrival patterns.

5.1 Protocols Minimizing Maximum Delay

We consider first a simple hybrid of batching and cyclic multicast termed here *cyclic/constant delay, listeners* (*cyclic/cd,l*), in which a cyclic multicast is initiated only after a batching delay (as in the *batching/cbd* protocol from Section 3.1), and is terminated when there are no remaining clients with outstanding requests (as in the *cyclic/l* protocol). With batching delay parameter Δ and transmission rate r ($r \le b$), the average duration of a channel busy period is given by $(e^{\lambda L/r} - 1)/\lambda$, and the average duration of an idle period is given by $1/\lambda + \Delta$. This yields

$$B_{c/cd,l} = r\frac{e^{\lambda L/r} - 1}{e^{\lambda L/r} + \lambda\Delta}; \qquad A_{c/cd,l} = \frac{\Delta(1+\lambda\Delta/2)}{e^{\lambda L/r} + \lambda\Delta} + L/r; \qquad D_{c/cd,l} = \Delta + L/r.$$

The operation of the *cyclic/cd,l* protocol, as well as that of the other protocols developed in this section, is illustrated in Figure 6 for the same example pattern of request arrivals as in Figure 1.

For $D_{c/cd,l} > L/b$, there are multiple combinations of Δ and r that yield the same maximum client delay. Optimal settings that minimize server bandwidth can be found numerically. Interestingly, $r = b$ is often not optimal. Since a cyclic multicast is continued as long as there is at least one listening client, channel busy periods may have long durations. Under such conditions, it may be possible to reduce server bandwidth usage while keeping the maximum delay fixed by reducing both r and Δ. In particular, note that for $\lambda \to \infty$, the channel is always busy, and thus the optimal r is the minimum possible r (the file size L divided by the maximum delay) and the optimum Δ is zero.

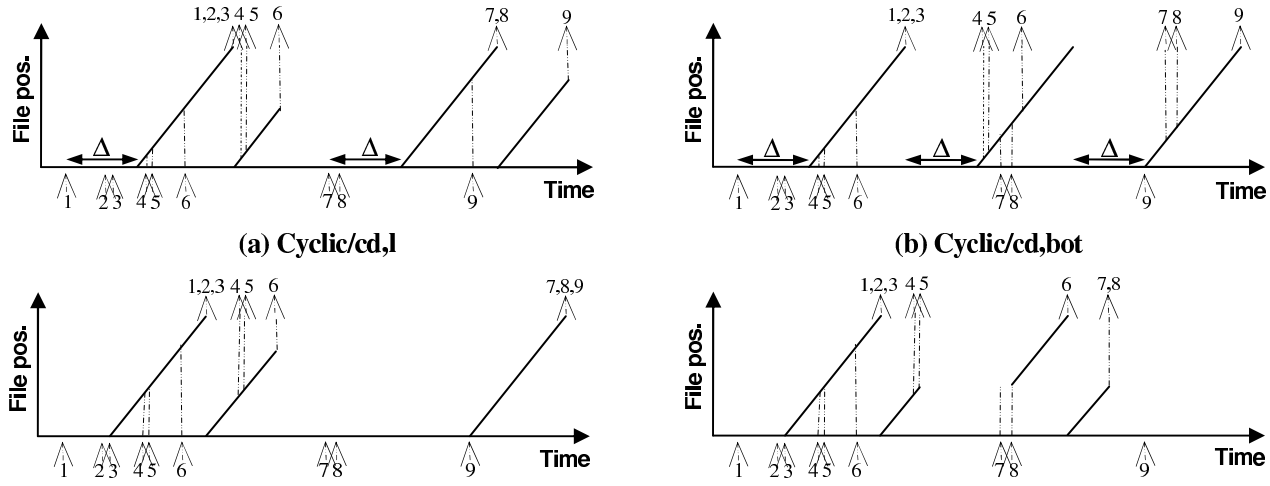
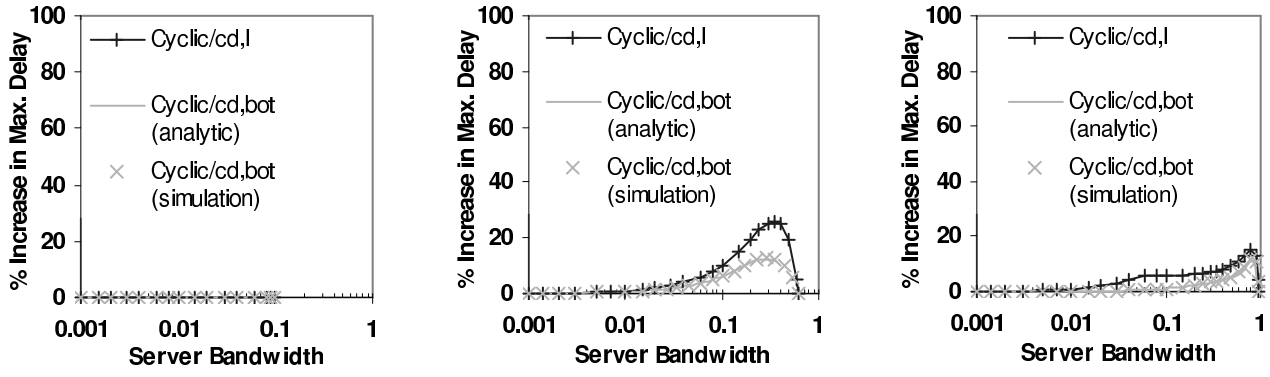


Figure 6: Examples Scenarios for Improved Protocols

A better hybrid protocol, termed here *cyclic/constant delay, bounded on-time (cyclic/cd,bot)*, can be devised by using a better policy for when to stop transmitting. The key observation is that the duration of a multicast transmission can be limited to at most L/r without impact on the maximum client delay. As in the *cyclic/cd,l* protocol, a cyclic multicast is initiated only after a batching delay Δ, but the multicast is terminated after at most a duration L/r, allowing the server to be idle for a new batching delay Δ that impacts only the clients whose requests arrived after the multicast began, if any. Any clients whose requests arrive during a multicast will receive part of the file during the multicast that is in progress and the rest of the file during the next multicast one batching delay Δ later, thus guaranteeing a maximum client delay of $\Delta + L/r$. A multicast is terminated before duration L/r when a client completes reception of the file and there are no remaining listeners, an event that will occur if no new client has arrived since before the previous multicast terminated. Note that the relatively simple operation of this protocol, illustrated in Figure 6(b), is in contrast to that of *slp*, for which the transmission schedule and service of any particular client can be extremely fragmented. The optimal value for r with *cyclic/cd,bot* is the maximum possible (b), and thus this parameter setting is used in our experiments.

Accurate approximations for the average server bandwidth usage and average client delay with the *cyclic/cd,bot* protocol can be derived as follows. First, we distinguish two types of channel busy periods. Channel busy periods such that at least one request arrival occurred during the preceding idle period are termed "type 1" busy periods, and will have the maximum duration L/r. The remaining busy periods are termed "type 2" busy periods. A type 2 busy period will have duration equal to L/r if there is at least one request arrival during this period. If there are no such arrivals, the duration will equal the maximum, over all clients whose requests arrived during the preceding busy period, of the amount of data that the client has yet to receive, divided by r.

We make the approximation that the rate at which a type 2 busy period ends when prior to its maximum duration L/r (i.e., the system empties) is constant. Denoting this rate by α, the probability that a type 2 busy period is of duration less than L/r (also equal to the probability that the system empties during this busy period), is then given by $1-e^{-\alpha L/r}$, and the average duration of a type 2 busy period is given by $(1-e^{-\alpha L/r})/\alpha$. Note that the duration of a type 2 busy period of less than maximum duration depends only on the duration of the previous busy period and the points at which request arrivals occurred during this previous period. In light of this observation, we suppose that α is independent of Δ, and calculate α for a system with $\Delta \to 0$. Consider, for $\Delta \to 0$, the average total duration of a type 1 busy period and the following type 2 busy periods up to when the system next empties (following which there is the next type 1 busy period). This quantity is equal to the average duration of an $M/G/\infty$ busy period with arrival rate λ and service time L/r, as given by $(e^{\lambda L/r}-1)/\lambda$. This quantity is also equal



(a) b = 0.1 **(b) b = 1.0** **(c) b = 10.0**

Figure 7: Maximum Delay with Improved Protocols Relative to Lower Bound (L = 1, λ = 1)

to the probability that the total duration is greater than L/r (equal to $1-e^{-\lambda L/r}$) times the average total duration conditioned on being greater than L/r (equal to $L/r+1/\alpha$), plus the probability that the total duration is equal to L/r (equal to $e^{-\lambda L/r}$) times L/r, yielding

$$\left(e^{\lambda L/r}-1\right)/\lambda = \left(1-e^{-\lambda L/r}\right)\left(L/r+1/\alpha\right)+\left(e^{-\lambda L/r}\right)L/r \quad \Rightarrow \quad \alpha = \frac{1-e^{-\lambda L/r}}{\left(e^{\lambda L/r}-1\right)/\lambda - L/r}.$$

Let $p_{emptied}$ denote the probability that at the beginning of a randomly chosen idle period the system had emptied; i.e., there were no clients with unsatisfied requests. Let p_{type1} denote the probability that a randomly chosen busy period is of type 1. These two probabilities can be obtained by solving the following two equations, the first of which applies $p_{emptied}$ to the idle period preceding a randomly chosen busy period, and the second of which applies p_{type1} to the busy period preceding a randomly chosen idle period:

$$p_{type1} = p_{emptied} + \left(1-p_{emptied}\right)\left(1-e^{-\lambda\Delta}\right); \qquad p_{emptied} = p_{type1}e^{-\lambda L/r} + \left(1-p_{type1}\right)\left(1-e^{-\alpha L/r}\right).$$

The average duration of a channel busy period is given by $p_{type1}L/r+\left(1-p_{type1}\right)\left(1-e^{-\alpha L/r}\right)/\alpha$ and the average duration of an idle period by $p_{emptied}/\lambda+\Delta$, yielding

$$B_{c/cd,bot} = r\frac{p_{type1}L/r+\left(1-p_{type1}\right)\left(1-e^{-\alpha L/r}\right)/\alpha}{p_{type1}L/r+\left(1-p_{type1}\right)\left(1-e^{-\alpha L/r}\right)/\alpha + p_{emptied}/\lambda+\Delta};$$

$$A_{c/cd,bot} = \frac{\Delta\left(p_{emptied}+\lambda\Delta/2\right)+\left(1-p_{emptied}\right)\dfrac{(\lambda L/r)}{1-e^{-\lambda L/r}}\Delta}{\lambda\left(p_{type1}L/r+(1-p_{type1})(1-e^{-\alpha L/r})/\alpha + p_{emptied}/\lambda+\Delta\right)}+L/r; \qquad D_{c/cd,bot} = \Delta+L/r.$$

The derivation of the first term in the numerator of the equation for average delay is similar to the corresponding term in the average delay equations for *batching/cbd* and *cyclic/cd,l*, except that the batching delay was triggered by a new request arrival (which then experiences the maximum waiting time Δ) only in the case when the system has emptied (with probability $p_{emptied}$). The second term in the numerator is the probability that at the beginning of a randomly chosen idle period the system had not emptied (i.e., that the idle period results from the limit of L/r on the duration of a multicast), times the average number of clients still active at the beginning of such an idle period all of whom must wait until the next multicast to complete their service, times the duration Δ of this wait. The average number of clients active at the beginning of such an idle period is equal to the average number of arrivals during the preceding busy period (of length L/r), conditioned on there being at least one such arrival.

The results in Figure 7 show that the *cyclic/cd,bot* protocol performs close to optimal (within 15% in all cases). The figure also illustrates the high accuracy of the approximate analysis. In addition, Figure 7 illustrates that

even the simple hybrid *cyclic/cd,l* protocol can yield good performance (within 30% of optimal in all cases), although note that the results shown for this protocol are with optimal parameter settings. An advantage of *cyclic/cd,bot* is that it has just one parameter (Δ), which is chosen based on the desired trade-off between maximum delay and bandwidth usage. Since *cyclic/cd,bot* is relatively simple and outperforms *cyclic/cd,l*, the performance of *cyclic/cd,l* with alternative (suboptimal) parameter settings is not explored here.

5.2 Protocols Minimizing Average Delay

Again, we begin with a simple hybrid of batching and cyclic multicast in which a cyclic multicast is initiated only after a batching delay, in this case of the same form as in the *batching/rbd* protocol from Section 3.1, and terminated when there are no active clients (as in the *cyclic/l* protocol). The average server bandwidth and average/maximum client delay achieved with this *cyclic/request-based delay, listeners (cyclic/rbd,l)* protocol, with batching delay parameters n and f (integer $n \geq 1$, $0 \leq f < 1$), and transmission rate r ($r \leq b$), are given by

$$B_{c/rbd,l} = r\frac{e^{\lambda L/r}-1}{e^{\lambda L/r}-1+(n+f)}; \qquad A_{c/rbd,l} = \frac{n(n+2f-1)/(2\lambda)}{e^{\lambda L/r}-1+(n+f)} + L/r; \qquad D_{c/rbd,l} \text{ is unbounded.}$$

These expressions are derived using the average duration of a channel busy period $(e^{\lambda L/r}-1)/\lambda$ and the average duration of an idle period $(n+f)/\lambda$. As with the *cyclic/cd,l* protocol, $r = b$ is not necessarily optimal, and parameter settings that optimize for average delay are found numerically.

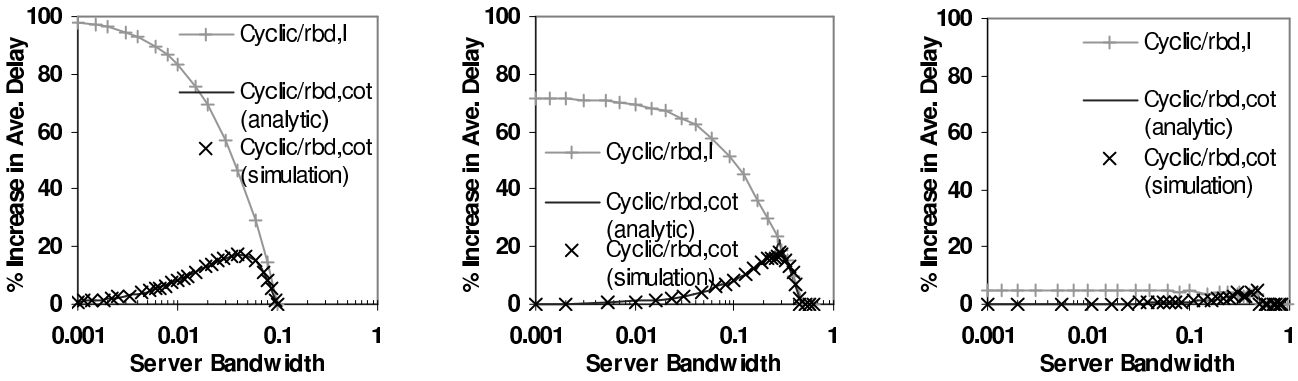
The key to designing a better protocol is, as before, determining a better policy for when to stop transmitting. If the total time each client spent receiving data from the channel was exponentially distributed (rather than of constant duration L/r), then the optimal policy for average delay would be for the server to continue its cyclic multicast whenever there is at least n (or n+1 for some fraction of busy periods f) clients with unfulfilled requests. In the (actual) case of constant service times, however, the objective of achieving consistently good sharing of multicasts has to be balanced by consideration of the required remaining service time of the active clients. For example, if a client has only a small amount of additional data that it needs to receive for its download to complete, then continuing the cyclic multicast may be optimal with respect to the average delay metric regardless of the number of other active clients.

In the protocol that we propose here, termed *cyclic/request-based delay, controlled on-time* (*cyclic/rbd,cot*), these factors are roughly balanced by distinguishing between clients whose requests were made prior to the beginning of a busy period, and clients whose requests were made during it. The server continues its cyclic multicast at least until all of the former clients complete their downloads (time L/r), after which transmission continues only as long as the number of clients with unfulfilled requests is at least max[n-1,1], where n is the same as the batching delay parameter that is used, together with the parameter f, to control the initiation of transmission after an idle period. Empirically, the optimal r is equal to b for this protocol.

Note that for n = 1 or 2, this protocol is identical to the *cyclic/rbd,l* protocol with $r = b$, the analysis of which was given above. Although an exact analysis of this protocol for $n \geq 3$ appears to be quite difficult, an accurate approximate analysis has been developed. This approximate analysis constrains the duration of a busy period to be a multiple of L/b, yielding the following approximations for server bandwidth usage and average client delay (for $n \geq 3$):

$$B_{c/rbd,cot} = b\frac{(1/p)L/b}{(1/p)L/b + \left(n+f-\sum_{i=0}^{n-2} i\frac{p_i}{p}\right)\Big/\lambda}, \qquad A_{c/rbd,cot} = \frac{L}{b} + \frac{\sum_{i=0}^{n-2}\frac{p_i}{p}\left[i(n-i)\frac{1}{\lambda} + \frac{(n-i)(n-i-1)}{2\lambda} + \frac{n}{\lambda}f\right]}{\lambda\left((1/p)L/b + \left(n+f-\sum_{i=0}^{n-2} i\frac{p_i}{p}\right)\Big/\lambda\right)},$$

where $p_i = \frac{1}{i!}(\lambda L/b)^i e^{-\lambda L/b}$ and $p = \sum_{i=0}^{n-2} p_i$. $D_{c/rbd,cot}$ is unbounded. The derivations of these expressions are analogous to those for the *shifted arrivals* protocol in Section 4.2.



(a) $b = 0.1$ **(b) $b = 1.0$** **(c) $b = 10.0$**

Figure 8: Average Delay with Improved Protocols Relative to Lower Bound ($L = 1, \lambda = 1$)

The results in Figure 8 show that the *cyclic/rbd,cot* protocol yields performance close to optimal, with an average delay within 20% of the lower bound in all cases considered. Note also that our lower bound on average delay is achievable only for high server bandwidth (low delay), specifically the region in which the *cyclic/w2,l* protocol operates, so performance is even closer to optimal than these results would suggest. Also shown in the figure is the high accuracy of the approximate analysis. Finally, the figure shows that the simple hybrid *cyclic/rbd,l* protocol yields good performance across the server bandwidth range of most interest only for high client reception rates (i.e., rates such that the probability of a client request arrival during the time required to download the file is very low).

5.3 Worst-Case Performance

This section relaxes the Poisson arrival assumption and considers the worst-case performance of the protocols under arbitrary request arrival patterns. Specifically, of interest is the worst-case average server bandwidth usage and average client delay, as functions of the protocol parameters and the average request rate λ. Our results are summarized in Table 2. We do not consider the maximum client delay, since for each protocol either the maximum delay is independent of the request arrival pattern, or it is unbounded under Poisson arrivals and can therefore be no worse with some other arrival process. Note that achieving these worst-case results often requires the arrival pattern to be pessimistically tuned according to the values of the protocol parameters, and that the worst-case average bandwidth usage and the worst-case average client delay cannot usually be achieved with the same arrival pattern.

Consider first the average client delay. For *cyclic/l*, the client delay (and thus the average client delay) is always L/r. For *batching/cbd*, *cyclic/cd,l*, and *cyclic/cd,bot*, the average client delay can be at most the maximum client delay, and this is achieved when all request arrivals occur in batches (of arbitrary size) with each batch arriving when there are no previous clients with outstanding requests. For *batching/rbd*, *cyclic/rbd,l*, and *cyclic/rbd,cot*, consider first the case of $f = 0$. With all three protocols, note that the average client delay cannot exceed $(n-1)/\lambda+L/r$, since in that case the average number of clients waiting for a multicast transmission to begin would (from Little's Law) exceed $n-1$, whereas in each protocol there can never be more than $n-1$ waiting clients. An arrival pattern for which this average client delay is achieved is as follows. Immediately after the end of a multicast transmission, a batch of $n-1$ requests arrives. Following this batch arrival a long delay ensues of deterministic duration $((n-1+m)/\lambda) - L/r$, for $m \to \infty$, followed by a batch arrival with m requests. This initiates a new multicast transmission of duration L/r. It is straightforward to verify that the average arrival rate with this request pattern is λ and that the average client delay tends to $(n-1)/\lambda+L/r$ as $m \to \infty$. For $f > 0$, the worst-case average delay depends on the precise policy by which the server determines whether to wait until n requests have accumulated, or to wait until $n+1$ requests have accumulated, prior to beginning a new multicast, rather than just the fraction f of occasions that it waits for $n+1$. We give here the highest possible worst-case average delay over

Table 2: Summary of Worst-Case Performance ($\partial_{f>0} = 1$ if $f > 0$ and 0 otherwise)

Protocol	Parameters	Average Client Delay	Average Server Bandwidth
Batching/cbd	Δ, r	$\Delta + L/r$	$\min[L/\Delta, \lambda L]$
Batching/rbd	n, f, r	$(n-1+\partial_{f>0})/\lambda + L/r$	$\lambda L/(n+f)$
Cyclic/l	r	L/r	$\min[r, \lambda L]$
Cyclic/cd,l	Δ, r	$\Delta + L/r$	$\min[r, \lambda L]$
Cyclic/cd,bot	Δ, r	$\Delta + L/r$	$\min[L/(\Delta + L/r), \lambda L]$
Cyclic/rbd,l	n, f, r	$(n-1+\partial_{f>0})/\lambda + L/r$	$\min[r, \lambda L]$
Cyclic/rbd,cot	n, f, r	$(n-1+\partial_{f>0})/\lambda + L/r$	$\min[r, \lambda L/\max[n-1,1]]$

all such policies, which can be achieved, for example, by a policy that makes the choice probabilistically. By the same argument as used above for the case of $f = 0$, the average client delay cannot exceed $n/\lambda + L/r$. An arrival pattern for which this average client delay is achieved is similar to that used above, but with a batch size of n rather than $n-1$, and (whenever the server chooses to wait for $n+1$ arrivals and thus a new transmission does not start immediately) a delay of duration $((n+fm)/\lambda - L/r)/f$, for $m \to \infty$, followed by a batch arrival with m requests.

Consider now the average server bandwidth. For *batching/rbd,* the average bandwidth depends only on the average arrival rate, rather than the specific arrival pattern, since every n^{th} (or $n+1^{st}$) request arrival causes a new transmission of the file that only the clients making those n (or $n+1$) requests receive. Thus, the worst-case average bandwidth usage for this protocol is the same as the average bandwidth usage for Poisson arrivals. For *batching/cbd*, if $\lambda \le 1/\Delta$ then request arrivals can be spaced such that no arrivals occur simultaneously and no arrivals occur during a batching delay, yielding a worst-case bandwidth usage equal to the unicast bandwidth usage of λL. For $\lambda \ge 1/\Delta$, batched arrivals with deterministic spacing of Δ between the batches yield the worst-case bandwidth usage of L/Δ. Thus, the worst-case bandwidth usage is $\min[L/\Delta, \lambda L]$. For *cyclic/l*, if $\lambda \le r/L$ the worst-case bandwidth usage is achieved when the spacing between consecutive arrivals is always at least L/r, yielding a bandwidth usage of λL. For $\lambda \ge r/L$, transmission can be continuous, giving a bandwidth usage of r. Thus, the worst-case bandwidth usage is $\min[r, \lambda L]$. The same worst-case bandwidth usage is achieved with *cyclic/cd,l*, and *cyclic/rbd,l*. For $\lambda \ge r/L$, transmission can be continuous, and for $\lambda \le r/L$ a bandwidth usage of λL is achieved when the fraction of arrivals that occur during busy periods approaches one, and the spacing between consecutive busy-period arrivals is of deterministic duration infinitesimally less than L/r. Similarly, for *cyclic/rbd,cot*, if $\lambda \le (r/L)(\max[n-1,1])$ the worst-case bandwidth usage is achieved when the fraction of arrivals that occur during busy periods approaches one, and busy period arrivals occur in batches of size $\max[n-1,1]$ with spacing between consecutive batches of deterministic duration infinitesimally less than L/r, yielding a bandwidth usage of $\lambda L/\max[n-1,1]$. For $\lambda \ge (r/L)(\max[n-1,1])$, arrivals can be spaced such that transmission is continuous, giving a bandwidth usage of r. Thus, the worst-case bandwidth usage is $\min[r, \lambda L/\max[n-1,1]]$. Finally, for *cyclic/cd,bot*, if $\lambda \le 1/(\Delta + L/r)$ then request arrivals can be spaced such that no arrivals occur simultaneously or during a batching delay or channel busy period, yielding a worst-case bandwidth usage of λL. For $\lambda \ge 1/(\Delta + L/r)$, arrivals can be spaced such that the system never empties, giving a bandwidth usage of $L/(\Delta + L/r)$. Thus, the worst-case bandwidth usage is $\min[L/(\Delta + L/r), \lambda L]$.

6. Heterogeneous Clients

In this section we relax our homogeneity assumption and consider the case in which there are multiple classes of clients with differing associated maximum delays (Section 6.1) and achievable reception rates (Section 6.2). Section 6.1 also supposes that the amount of data a client needs to receive from a channel may be class-specific. This scenario is relevant to the protocols developed in Section 6.2, in which file data blocks are delivered on multiple channels and each client listens to the subset of channels appropriate to its achievable reception rate.

Throughout this section only maximum client delay is considered, although our results can also yield insight for the case in which average client delay is the metric of most interest.

6.1 Class-Specific Service Requirement and Maximum Delay

Here we assume that clients of class i have maximum delay D_i and need to receive an amount of file data L_i from a single shared channel. All clients have a common reception rate constraint b. As in the case of homogeneous clients, the *slp* protocol is optimal and thus its average bandwidth usage provides a lower bound on that achievable with any protocol. Section 6.1.1 generalizes the approximation for this lower bound that was given in Section 4.2, to this heterogeneous context. As motivated again by the complexity of *slp*, Section 6.1.2 extends the simpler and near-optimal *cyclic/cd,bot* protocol given in Section 5.1, so as to accommodate heterogeneous clients, and compares its performance to that of *slp*.

6.1.1 Lower Bound (slp) Bandwidth Approximation

A key observation used to generalize our lower bound approximation is that with *slp*, the presence or absence of requests from "high slack" clients (i.e., clients of classes j such that D_j is large relative to L_j/b), will have relatively little impact on the server bandwidth usage during periods with one or more active "low slack" clients. Exploiting this observation, the classes are ordered in non-increasing order of L_i/D_i, and the average server bandwidth usage of *slp*, with the assumed client heterogeneity, is written as

$$B_{slp} = \sum_{i=1}^{C} (P_i - P_{i-1})\,\beta_i ,$$

where C denotes the number of customer classes, P_i denotes the (cumulative) probability that there is at least one client from classes 1 through i with an outstanding request (with P_0 defined as 0), and β_i denotes the average server bandwidth usage over those periods of time during which there is at least one client from class i with an outstanding request but none from any class indexed lower than i.

An approximation for the probability P_i can be obtained using a similar approach as was used for the corresponding quantity in the approximation for homogeneous clients. P_i is equal to the average duration of a period during which there is at least one client from classes 1 through i with an outstanding request, divided by the sum of this average duration and the average request inter-arrival time for this set of classes ($1/\sum_{k=1}^{i}\lambda_k$, where λ_k denotes the rate of requests from class k clients). The average duration of a period during which there is at least one client from classes 1 through i with an outstanding request is approximated by the average duration of an $M/G/\infty$ busy period with arrival rate $\sum_{k=1}^{i}\lambda_k$ and average service time $\sum_{j=1}^{i}\frac{\lambda_j}{\sum_{k=1}^{i}\lambda_k}L_j/b$, as given by $(e^{\sum_{j=1}^{i}\lambda_j L_j/b}-1)/\sum_{k=1}^{i}\lambda_k$, plus the average duration of the delay after the arrival of a request to an idle system, until the server must begin transmitting ($\sum_{j=1}^{i}\frac{\lambda_j}{\sum_{k=1}^{i}\lambda_k}(D_j - L_j/b)$), yielding

$$P_i \approx \frac{e^{\sum_{j=1}^{i}\lambda_j L_j/b} - 1 + \sum_{j=1}^{i}\lambda_j\left(D_j - L_j/b\right)}{e^{\sum_{j=1}^{i}\lambda_j L_j/b} + \sum_{j=1}^{i}\lambda_j\left(D_j - L_j/b\right)}.$$

The average bandwidth usage β_i is approximated as L_i reduced by the average amount of data x_i received by a class i client while there is at least one active client from a lower indexed class, divided by the portion of the time D_i during which no such lower indexed client is active:

$$\beta_i \approx \frac{L_i - x_i}{D_i(1 - P_{i-1})}.$$

Defining β_ave_i by

$$\beta_ave_i = \sum_{j=1}^{i-1} \beta_j \left(P_j - P_{j-1}\right) / P_{i-1} ,$$

the quantity x_i is computed using

$$x_i \approx \beta_i (D_i P_{i-1}) + (\beta_ave_i - \beta_i) E_i ,$$

where E_i denotes the average portion of the time D_i during which a class i client receives data from the channel at the higher average rate equal to β_ave_i, owing to the presence of requests from lower indexed classes, rather than at the lower rate β_i. A simple approximation for E_i would be $D_i P_{i-1}$, but this would neglect the impact of variability in the portion of time t_i that there is at least one active client from a lower indexed class, during the period over which a particular class i client is active. In particular, there is a maximum length of time during which a class i client can receive data at the higher average rate, without accumulating an amount of data exceeding L_i. Noting that t_i is at most D_i, we capture the first-order impact of variability by assuming a truncated exponential distribution for t_i, with rate parameter φ_i such that the average of the distribution is $D_i P_{i-1}$:

$$\frac{1}{\varphi_i} - \left(D_i e^{-\varphi_i D_i}\right) / \left(1 - e^{-\varphi_i D_i}\right) = D_i P_{i-1} .$$

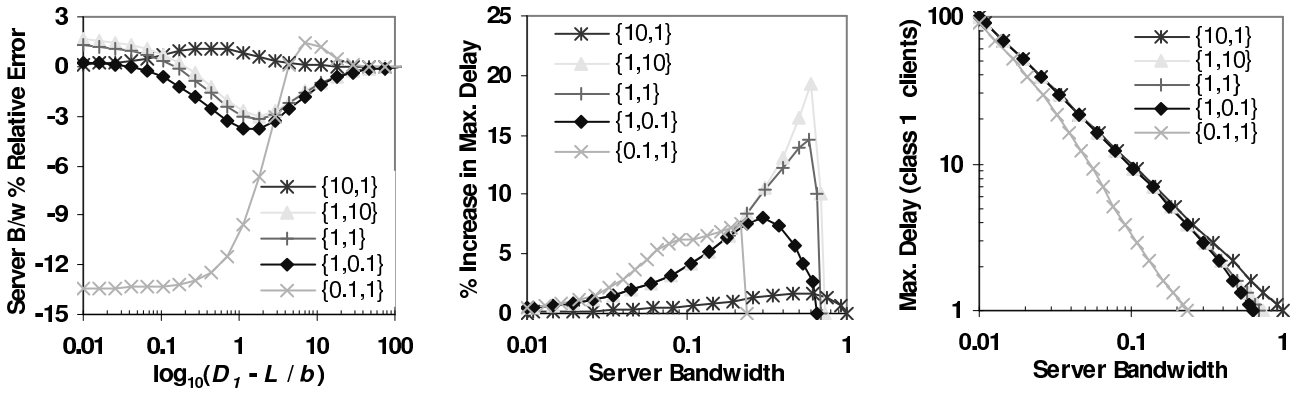
During the portion of time when a class i client is receiving data at the higher average rate β_ave_i, the rate at which additional data is received (i.e., beyond that which would otherwise be received) is given by $\beta_ave_i - \beta_i$. Since at most L_i data can be received in total during this time period, the average additional amount of data that can be received owing to reception at the higher average rate is upper bounded by $L_i - E_i \beta_i$. We approximate the maximum length of time during which a class i client can receive data at the higher average rate, without accumulating an amount of data exceeding L_i, by $t_max_i = \min[D_i, (L_i - E_i \beta_i)/(\beta_ave_i - \beta_i)]$. An approximation for E_i is then obtained as

$$E_i \approx \left(\frac{\left(1 - e^{-\varphi_i t_max_i}\right) / \varphi_i - t_max_i \left(e^{-\varphi_i t_max_i}\right)}{1 - e^{-\varphi_i D_i}} \right) + t_max_i \left(\frac{e^{-\varphi_i t_max_i} - e^{-\varphi_i D_i}}{1 - e^{-\varphi_i D_i}} \right),$$

where the first term is the probability that t_i does not exceed t_max_i times its expected value in this case, and the second term is t_max_i times the probability that t_i exceeds t_max_i.

The above analysis results in a system of non-linear equations that can easily be solved numerically, beginning with the quantities for class 1 and proceeding to those for successively higher indexed classes. Although the analysis might seem complex, simpler variants were found to have substantially poorer accuracy. Note also that for the case in which the client classes have identical L_i and D_i, the analysis yields identical bandwidths β_i, and the bound reduces to that given earlier for homogeneous clients.

Sample validation results comparing the analysis against simulations of the *slp* protocol are presented in Figure 9(a). In the scenarios considered in this figure there are two client classes, with $L_1 = L_2 = 1$ and $D_2 = 5D_1$. The maximum sustainable client reception rate b is fixed at one. Five combinations of request rates $\{\lambda_1, \lambda_2\}$ are considered, and the percent relative error in the average server bandwidth usage computed using the approximate analysis is plotted against the slack $(D - L/b)$ of the low slack clients (class 1), for each request rate combination. Additional experiments included a full factorial experiment for two class systems, and an experiment in which a large number of randomly generated systems with 3-6 classes were tested. In all these experiments no case was found in which the absolute relative error exceeded 20%.



(a) Accuracy of Lower Bound Approximation **(b) Maximum Delay with *Cyclic/cd,bot* Relative to Lower Bound** **(c) Relative Impact of Low Slack vs. High Slack Clients**

Figure 9: Impact of Class-Specific Maximum Delays ($L = 1, D_2 = 5\,D_1$, varying arrival rates $\{\lambda_1, \lambda_2\}$)

6.1.2 Extension of Cyclic/cd,bot

The *cyclic/cd,bot* protocol is extended to accommodate heterogeneous clients as follows. The duration of each multicast transmission is limited to at most the maximum value of L_i/r over all classes i that have active clients at the beginning of the transmission. As before, if the last active client completes reception of the file and there are no more listeners, the transmission is terminated early. The delay Δ becomes variable, now being dependent on which classes have clients with outstanding requests. At the beginning of each delay period, it is initialized to the minimum value of $D_i - L_i/r$ over all classes i that have active clients. If a client of some other class j arrives during the delay period, and the time remaining in the delay period exceeds $D_j - L_j/r$, the length of the delay period must be reduced accordingly. As before, each client obtains the entire file either in a single busy period, or in two busy periods separated by an idle period, and the optimal r is equal to b.

Representative simulation results comparing performance with the extended *cyclic/cd,bot* protocol to the lower bound defined by the optimal *slp* protocol are presented in Figure 9(b). (The analytic approximation from Section 6.1.1 is not used here, as the differences from optimality of *cyclic/cd,bot* are not sufficiently greater than the errors in the approximation.) As in Figure 9(a), there are two client classes with $L_1 = L_2 = 1$ and $D_2 = 5D_1$, the client reception rate b is fixed at one, and five combinations of request rates $\{\lambda_1, \lambda_2\}$ are considered. As the figure illustrates, the achieved performance is reasonably close to optimal.

Figure 9(c) shows the maximum delay for class 1 clients (the maximum delay for class 2 clients is five times greater) as a function of server bandwidth for the *cyclic/cd,bot* protocol, for the same scenarios as previously considered. Noting that the curves can be separated into three groups based only on the request rate of the low slack clients, the main observation from this figure is the minimal impact of the request rate of the "high slack" clients on system performance.

6.2 Class-Specific Reception Rates

Suppose now that class i clients have a class-specific maximum sustainable client reception rate b_i as well as maximum delay D_i, but common $L_i = L$. Section 6.2.1 presents an algorithm for computing a lower bound on the required average server bandwidth. In Section 6.2.2, scalable download protocols for this context are proposed and their performance evaluated.

6.2.1 Heterogeneous Lower Bound

The *slp* protocol can be suboptimal when there is heterogeneity in client reception rates. For example, consider a scenario in which two clients request the file at approximately the same time, one with a relatively high reception rate and a relatively low maximum delay and one with a low reception rate and a high maximum delay, and in which no other requests arrive until these two clients have completed reception. With *slp*, the server will delay beginning transmission for as long as possible, and then, if it is the high rate client that has no slack at this point,

Table 3: Notation for Heterogeneous Lower Bound Algorithm

Symbol	Definition
K	Length of request sequence, with requests indexed from 1 to K in order of request deadline
$c(j)$	The class of the request j client
T_j^A	Arrival time of request j
T_j^D	Deadline of request j ($T_j^A + D_{c(j)}$)
$T_{j,i}$	Time from the arrival of request i until the deadline of request j ($T_j^D - T_i^A$)
x_j	Amount of data received by the request j client, from transmissions not received by any client with an earlier request deadline
$x_{j,i}$	Amount of data received by the request j client, from transmissions not received by any client with an earlier request deadline, that is also received by the request i client ($j < i \le K$)
$y_{j,i}$	Sum of $x_{k,i}$ for $1 \le k \le j$
B_j	Total amount of data transmitted to serve requests 1 through j

$B_0^{hlb} = 0,\ y_{0,i}^{hlb} = 0 \ \ 1 \le i \le K$
for j=1 to K
  $x_j^{hlb} = L - y_{j-1,j}^{hlb}$
  $B_j^{hlb} = B_{j-1}^{hlb} + x_j^{hlb}$
  for i=j+1 to K
    if $T_i^A < T_j^D$ then
      $x_{j,i}^{hlb} = \min\left\{x_j^{hlb},\ L - y_{j-1,i}^{hlb},\ b_{c(i)}T_{j,i} - y_{j-1,i}^{hlb},\ b_{c(i)}T_{j,j}\right\}$
    else
      $x_{j,i}^{hlb} = 0$
    $y_{j,i}^{hlb} = y_{j-1,i}^{hlb} + x_{j,i}^{hlb}$
  end for
end for

Figure 10: Heterogeneous Lower Bound Algorithm

begin transmitting at an aggregate rate equal to the rate of the high rate client. However, in this case greater sharing of the server transmissions, and thus lower server bandwidth usage, could be achieved by starting transmission earlier, at the low rate.

Using the notation in Table 3, Figure 10 presents an algorithm that yields a lower bound on the server bandwidth required to serve a given sequence of request arrivals[7]. The algorithm considers each request j in order of request deadline; i.e., the time by which the associated client must have completed reception of the file so as not to exceed the maximum delay for its respective class. The quantity x_j^{hlb} approximates (in a manner allowing a lower bound on server bandwidth to be computed) the amount of additional data (not received by earlier clients) that the server would need to transmit to enable the request j client to meet its deadline. This quantity is computed as $L - y_{j-1,j}^{hlb}$, where $y_{j-1,j}^{hlb}$ is the total over all earlier requests k of an optimistic estimate $x_{k,j}^{hlb}$ of the portion of x_k^{hlb} that the request j client could have shared reception of. A proof that $B_j^{hlb} = \sum_{k=1}^{j} x_k^{hlb}$ is a lower bound on the total server bandwidth required to serve requests 1 through j is given in the Appendix. In the case that all classes share a common maximum sustainable client reception rate, the lower bound is tight and gives the bandwidth used by *slp*. With heterogeneous client reception rates, the bound may be unachievable.

6.2.2 Protocols

Perhaps the simplest protocol for serving clients with heterogeneous reception rates is to dedicate a separate channel to each class. Any of the scalable download protocols from Section 5 can be utilized on each channel, with transmission rate chosen not to exceed the maximum sustainable reception rate of the respective clients. The disadvantage of this *separate channels* protocol is that there is no sharing of server transmissions among clients of different classes.

A second approach, termed here *shared cyclic/listeners (s-cyclic/l)*, extends the *cyclic/l* protocol from Section 3.2 to this heterogeneous client context. The client classes are indexed in decreasing order of their associated

[7] The algorithm as presented in Figure 10 has complexity $O(K^2)$, but can easily be implemented in a more efficient manner in which only requests i whose time in system overlaps with that of request j are explicitly considered in the inner loop.

maximum delays, aggregating any classes with equal maximum delays into a single class. A channel is created for each class, with the transmission rate on channel 1 chosen as L/D_1 and the rate on channel i for $i > 1$ chosen as $L/D_i - L/D_{i-1}$. Class i clients listen to channels 1 through i.[8] The server cyclically multicasts file data on each channel, whenever at least one client is listening. We assume here (as well as for the remaining protocols discussed in this section) that through careful selection of the order in which data blocks are transmitted on each channel [7, 8], and/or use of erasure codes with long "stretch factors", a client listening to multiple channels will nonetheless never receive the same data twice. The average server bandwidth usage on each channel i may be derived in a similar fashion as for the *cyclic/l* protocol, yielding

$$B_{sc/l} = (L/D_1)\left(1 - e^{-\sum_{j=1}^{C}\lambda_j D_j}\right) + \sum_{i=2}^{C}(L/D_i - L/D_{i-1})\left(1 - e^{-\sum_{j=i}^{C}\lambda_j D_j}\right).$$

This protocol achieves sharing of server transmissions among clients of different classes, but as with the *cyclic/l* protocol there will be periods over which transmissions on a channel serve relatively few clients.

The near-optimal protocols for delivery to homogeneous clients that were proposed in Section 5 have the characteristic that whenever the server transmits, it is at the maximum client reception rate b. Intuitively, for fixed maximum or average client delay, transmitting at the maximum rate allows a greater delay before beginning any particular transmission, and thus a greater opportunity for batching. In contrast, note that in the *s-cyclic/l* protocol, clients of each class i receive server transmissions that are at an aggregate rate equal to the *minimum* rate required to complete their downloads within time D_i. The key to devising an improved protocol is to achieve a good compromise between use of higher aggregate rates, which permit better batching opportunities for the clients that can receive at those rates, and low aggregate rates that maximize the sharing of server transmissions among clients of different classes.

We define a family of protocols that enables such a compromise, as follows. The client classes are indexed in non-decreasing order of their reception rates. A channel is created for each client class, with the transmission rate r_i on channel i chosen as $b_i - \sum_{j=1}^{i-1} r_j$.[9] Class i clients receive an amount of data l_i^j on each channel j, for $1 \le j \le i$, as determined by the protocol, such that $L = \sum_{j=1}^{i} l_i^j$. Server transmissions on each channel follow a protocol such as the extended *cyclic/cd,bot* protocol from Section 6.1.

Within this family, two extremes can be identified. At one extreme, clients receive the maximum amount of data possible on the lower-numbered channels, thus maximizing the sharing of transmissions among clients of different classes. Specifically, class i clients receive an amount of data $l_i^j = \min[L - \sum_{k=1}^{j-1} l_i^k, r_j D_i]$ on each channel j, $1 \le j \le i$.[10] At the other extreme, batching opportunities for class i clients are maximized by equalizing their slack on each channel. In this case, $l_i^j = (r_j/b_i)L$ for each channel j, $1 \le j \le i$. Simulation results have shown that neither of these protocols yields uniformly better performance than the other, and that the performance differences between them can be quite significant.

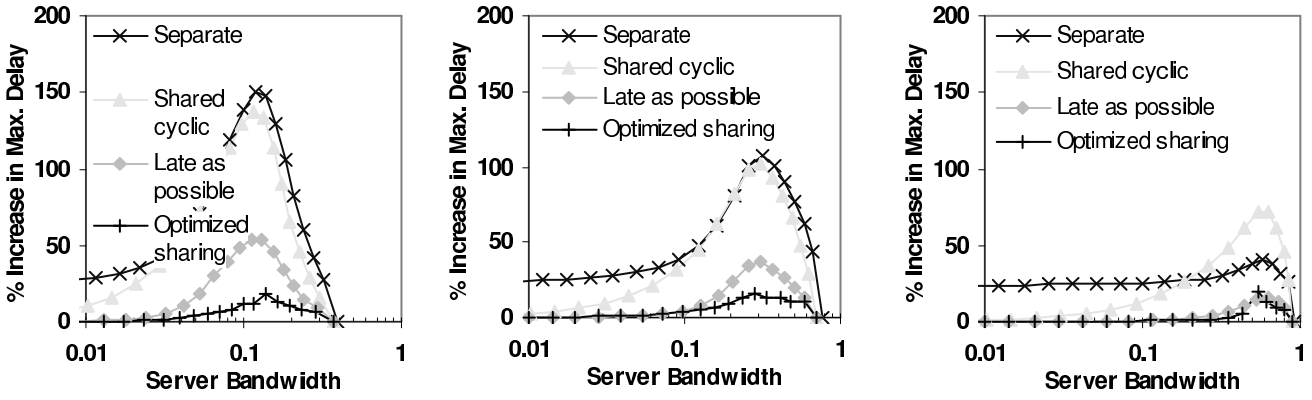
The best intermediate strategy can be closely approximated by a protocol termed here *optimized sharing*, in which the l_i^j values are chosen to be approximately optimal. With C classes, the number of free parameters in the optimization problem is $C(C-1)/2$. For each candidate allocation, the approximate lower bound analysis from Section 6.1 can be used to estimate the average server bandwidth with that allocation. With a small number of

---

[8] Alternatively, a large number of channels may be employed, with the server transmitting on each at the same low rate r. Class i clients would then listen to channels 1 through k_i, where $k_i = \lceil L/(rD_i) \rceil$.

[9] Note that if $b_i = b_{i-1}$, then the rate r_i is computed as 0. Channel i will then not be used, but for convenience of indexing we retain it.

[10] If this rule results in class i clients retrieving no data from channel i, then channel i can effectively be aggregated with channel i+1.



(a) 80% b=0.2; 10% b=1; 10% b=5 **(b) equal split among b = 0.2, 1, 5** **(c) 10% b = 0.2; 10% b=1; 80% b=5**

Figure 11: Maximum Delay with Heterogeneous Client Protocols Relative to Lower Bound (L = 1, λ = 1, D values inversely proportional to maximum achievable reception rates)

classes, as in the experiments whose results are presented here, L can be discretized and exhaustive search employed, for example, to find an allocation that results in the minimum predicted average server bandwidth.

Note that with all of the above protocols, the amount of data received on each channel by a client is statically determined according to the client's class. We also consider the extension to heterogeneous clients of the *slp* protocol, in which a client's use of each channel is dynamically determined. The client classes are indexed in non-decreasing order of their associated maximum sustainable reception rates. The server transmits at aggregate rate b_i whenever there is at least one client from class i that has no slack, and there is no such client from a class indexed higher than i. Channels are defined (as in the previous protocol family, for example), such that a class j client can receive at rate min[b_i, b_j] whenever the server is transmitting at aggregate rate b_i.

Figure 11 shows representative performance results, using the heterogeneous lower bound algorithm from Section 6.2.1 to provide a baseline for comparison. For the *separate channels* and *optimized sharing* protocols, the optimal *slp* protocol is used on each channel, although as illustrated in Figure 9(b) use of the more practical *cyclic/cd,bot* protocol would not greatly impact the results. For the heterogeneous client *slp* protocol and for *optimized sharing*, simulation is used to obtain the results shown (although as noted previously, the approximate lower bound analysis is used in *optimized sharing* to determine the data allocation), while for *separate channels* and *s-cyclic/l*, the results are from analysis. In the scenarios considered in this figure there are 3 client classes with respective reception rates of 0.2, 1, and 5, and D values such that $b_iD_i = b_jD_j$ for all classes i, j. The total request arrival rate is (without loss of generality) fixed at one, and the different parts of the figure correspond to different choices for the division of the total request rate among the classes.

The principal observations from this figure are: (1) the *separate channels* protocol yields poor performance, even in this scenario with greatly differing client reception rates; (2) the *s-cyclic/l* protocol can yield performance as poor, or worse than, *separate channels* (note, however, that the protocol does relatively better when the classes are more similar); (3) the *optimized sharing* protocol yields substantially better performance than *separate channels* and *s-cyclic/l*, and never worse (and sometimes significantly better) than the heterogeneous client *slp* protocol; and (4) the *optimized sharing* protocol does not appear to leave much room for performance improvement, achieving within 25% of the lower bound on maximum client delay in all scenarios considered.

7. Conclusions

This paper has considered the problem of using scalable multicast protocols to support on-demand download of large files from a single server to potentially large numbers of clients. Lower bounds were developed that indicate the best achievable performance. An optimized cyclic multicast protocol and two batching protocols optimized for average and maximum client delay were found to have significantly suboptimal performance over particular regions of the system design space, motivating the development of new hybrid protocols.

In the case of homogeneous clients, the best of the new practical protocols that focus on improving maximum client delay (*cyclic/cd,bot*) yielded results within 15% of optimal, in all scenarios considered. Similarly, the best of the new protocols designed to improve average client delay (*cyclic/rbd,cot*) yielded results within 20% of optimal. Both these protocols allow clients to begin listening to an on-going multicast if one is in progress at the times of their requests. Both protocols also achieve efficient batching of clients through use of a batching delay prior to the start of each multicast transmission and by limiting the transmission duration.

With the objective of minimizing the maximum client delay, *cyclic/cd,bot* uses a batching delay of fixed duration, and terminates each multicast transmission after delivering the full file or when a client completes reception of the file and there are no remaining listeners. In contrast, with the objective of minimizing the average client delay *cyclic/rbd,cot* initiates each new multicast transmission only when the number of waiting clients reaches some minimum value. The multicast is terminated when the clients that were waiting at the beginning of the multicast have completed reception, and the number of newly arrived clients still listening to the multicast drops below some minimum value.

For heterogeneous clients, the proposed *optimized sharing* protocol achieved within 25% of the optimal maximum client delay, in all scenarios considered. This protocol uses multiple channels to deliver the file data, and an analytic model to estimate the optimal amount of data that each class of clients should retrieve from each channel. An interesting observation is that *optimized sharing* can substantially outperform *send as late as possible*, which is optimal in the homogenous environment.

The work in this paper has focused on protocols for delivery from a single server. On-going work concerns the problem of devising optimal and near-optimal delivery protocols when files are replicated at multiple servers.

8. References

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, "Broadcast Disks: Data Management for Asymmetric Communication Environments", *Proc. ACM SIGMOD '95*, San Jose, CA, May 1995, pp. 199--210.

[2] S. Acharya, M. Franklin, and S. Zdonik, "Balancing Push and Pull for Data Broadcast", *Proc. ACM SIGMOD '97*, Tucson, AZ, May 1997, pp. 183--194.

[3] C. Aggarwal, J. Wolf, and P. Yu, "On Optimal Batching Policies for Video-on-Demand Storage Servers", *Proc ICMCS '96*, Hiroshima, Japan, June 1996, pp. 253--258.

[4] D. Aksoy, and M. Franklin, "RxW: A Scheduling Approach for Large-Scale On-Demand Data Broadcast", *IEEE Trans. on Networking 7*, 6 (Dec. 1999), pp. 846--860.

[5] K. V. Almeroth, M. H. Ammar, and Z. Fei, "Scalable Delivery of Web Pages Using Cyclic Best-Effort (UDP) Multicast", *Proc. IEEE Infocom '98*, San Francisco, CA, Mar. 1998, pp. 1214--1221.

[6] M. H. Ammar, and J. W. Wong. "On the Optimality of Cyclic Transmission in Teletext Systems", *IEEE Trans. on Communications 35*, 1 (Jan. 1987), pp. 68--73.

[7] S. Bhattacharyya, J. F. Kurose, D. Towsley, and R. Nagarajan, "Efficient Rate-Controlled Bulk Data Transfer using Multiple Multicast Groups", *Proc. IEEE Infocom '98*, San Francisco, CA, Apr. 1998, pp. 1172--1179.

[8] Y. Birk, D. Crupnicoff, "A Multicast Transmission Schedule for Scalable Multi-Rate Distribution of Bulk Data using Non-Scalable Erasure-Correcting Codes", *Proc. IEEE Infocom '03*, San Francisco, CA, Mar./Apr. 2003, pp. 1033--1043.

[9] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data", *Proc. SIGCOMM '98*, Vancouver, BC, Canada, Sept. 1998, pp 56--67.

[10] A. Dan , D. Sitaram, and P. Shahabuddin, "Scheduling Policies for an On-Demand Video Server with Batching", *Proc. ACM Multimedia '94*, San Francisco, CA, Oct. 1994, pp.15--23.

[11] H. D. Dykeman, M. H. Ammar, and J.W. Wong, "Scheduling Algorithms for Videotex Systems under Broadcast Delivery", *Proc. ICC '86*, Toronto, ON, Canada, June 1986, pp. 1847--1851.

[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, pp. 206--215.

[13] D. L. Eager, M. K. Vernon, and J. Zahorjan, "Minimizing Bandwidth Requirements for On-Demand Data Delivery", *IEEE Trans. on Knowledge and Data Engineering 13*, 5 (Sept./Oct. 2001), pp. 742--757.

[14] S. Hameed, and N. H. Vaidya, "Log-Time Algorithms for Scheduling Single and Multiple Channel Data Broadcast", *Proc. ACM/IEEE MobiCom '97*, Budapest, Hungary, Sept. 1997, pp. 90--99.

[15] L. Rizzo and L. Vicisano, "A Reliable Multicast Data Distribution Protocol Based on Software FEC Techniques", *Proc. HPCS '97*, Chalkidiki, Greece, June 1997, pp. 115--124.

[16] L. Rizzo, "Effective Erasure Codes for Reliable Computer Communication Protocols", *ACM Computer Communication Review 27*, 2 (Apr. 1997), pp. 24--36.

[17] S. Rost, J. Byers, and A. Bestavros, "The Cyclone Server Architecture: Streamlining Delivery of Popular Content", *Proc. WCW '01*, Boston, MA, June 2001, pp. 147--163.

[18] A. Shokrollahi, "Raptor Codes", *Technical Report DF2003-06-001*, Digital Fountain Inc., 2003.

[19] K. Stathatos, N. Roussopoulos, and J. Baras, "Adaptive Data Broadcast in Hybrid Networks", *Proc. VLDB '97*, Athens, Greece, Sept. 1997, pp. 326--335.

[20] 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, pp. 387--410.

[21] A. K. Tsiolis, and M. K. Vernon, "Group Guaranteed Channel Capacity in Multimedia Storage Servers", *Proc. ACM Sigmetrics '97*, Seattle, WA, June 1997, pp. 285--297.

[22] L. Vicisano, L. Rizzo, and J. Crowcroft, "TCP-like Congestion Control for Layered Video Multicast Data Transfer", *Proc. IEEE Infocom '98*, San Francisco, CA, Apr. 1998, pp. 996--1003.

[23] J. L. Wolf, M. S. Squillante, J. Turek, P. S. Yu, and J. Sethuraman, "Scheduling Algorithms for the Broadcast Delivery of Digital Products", *IEEE Trans. on Knowledge and Data Engineering 13*, 5 (Sept./Oct. 2001), pp. 721--741.

[24] J. W. Wong, "Broadcast Delivery", *Proc. of the IEEE 76*, 12 (Dec. 1988), pp. 1566--1577.

Appendix

This appendix provides a proof of the *heterogeneous lower bound* computed using the algorithm in Figure 10; i.e., that $B_j^{hlb} \le B_j$ for any realizable protocol. We actually prove a stronger result, by considering a more general algorithm in which the expression giving x_j^{hlb} in Figure 10 is replaced by $L - y_{j-1,j}^{hlb} - \varepsilon_j$, where the ε_j, $1 \le j \le K$, can be chosen to be any values such that $L - y_{j-1,j}^{hlb} \ge \varepsilon_j \ge 0$.

The proof that this more general algorithm yields a lower bound on B_j uses strong induction on j. As each client receives an amount of data equal to the file size L, in the case of just a single request $B_1^{hlb} = L - \varepsilon_1 \le L = B_1$, thus establishing the induction basis. Now, assume that $B_j^{hlb} \le B_j$, $B_{j-1}^{hlb} \le B_{j-1}$, …, $B_1^{hlb} \le B_1$, for some $j \ge 1$. We show that $B_{j+1}^{hlb} \le B_{j+1}$ by establishing that

$$B_k^{hlb} + L - y_{k,j+1}^{hlb} \le B_k + L - y_{k,j+1} \tag{B.1}$$

for $k = 1, 2, \ldots, j$. Note that for $k = j$, relation (B.1) implies that

$$B_{j+1}^{hlb} = B_j^{hlb} + x_{j+1}^{hlb} = B_j^{hlb} + \left(L - y_{j,j+1}^{hlb} - \varepsilon_{j+1}\right) \le B_j^{hlb} + L - y_{j,j+1}^{hlb} \le B_j + L - y_{j,j+1} = B_j + x_{j+1} = B_{j+1}.$$

We show relation (B.1) by strong induction on k. For $k = 1$, since $B_1^{hlb} = L - \varepsilon_1$, $B_1 = L_1$, $y_{1,j+1}^{hlb} = x_{1,j+1}^{hlb}$, and $y_{1,j+1} = x_{1,j+1}$, relation (B.1) is equivalent to $x_{1,j+1} \le x_{1,j+1}^{hlb} + \varepsilon_1$. If $T_{j+1}^A \ge T_1^D$, $x_{1,j+1} = x_{1,j+1}^{hlb} = 0$ and the relation holds. Otherwise, using the expression giving $x_{j,i}^{hlb}$ in Figure 10, $x_{1,j+1} \le x_{1,j+1}^{hlb} + \varepsilon_1$ is equivalent to

$$x_{1,j+1} \le \min\{L - \varepsilon_1,\ L,\ b_{c(j+1)}T_{1,j+1},\ b_{c(j+1)}T_{1,1}\} + \varepsilon_1.$$

Since the amount of data received by the request 1 client from transmissions also received by the request j+1 client can be at most L, and at most $b_{c(j+1)}$ times the period over which such transmissions can occur, this establishes the induction basis.

Suppose now that relation (B.1) holds for k, k-1, …, 1, for some k such that $j > k \geq 1$, and consider the relation for k+1. If $T^A_{j+1} \geq T^D_{k+1}$, then $y^{hlb}_{k+1,j+1} = y_{k+1,j+1} = 0$, and the relation holds since from the inductive hypothesis on the main claim, $B^{hlb}_{k+1} \leq B_{k+1}$ for $k < j$. There are four cases to consider when $T^A_{j+1} < T^D_{k+1}$, based on which term in the expression giving $x^{hlb}_{j,i}$ in Figure 10 yields the minimum (i.e., whether $x^{hlb}_{k+1,j+1}$ is equal to x^{hlb}_{k+1}, $L - y^{hlb}_{k,j+1}$, $b_{c(j+1)}T_{k+1,j+1} - y^{hlb}_{k,j+1}$, or $b_{c(j+1)}T_{k+1,k+1}$).

***Case 1*:** $x^{hlb}_{k+1,j+1} = x^{hlb}_{k+1}$

Since relation (B.1) holds for k from the inductive hypothesis,

$$\begin{aligned} B^{hlb}_{k+1} + L - y^{hlb}_{k+1,j+1} &= \left(B^{hlb}_k + x^{hlb}_{k+1}\right) + L - \left(y^{hlb}_{k,j+1} + x^{hlb}_{k+1,j+1}\right) = B^{hlb}_k + L - y^{hlb}_{k,j+1} \\ &\leq B_k + L - y_{k,j+1} \leq B_k + L - y_{k,j+1} + \left(x_{k+1} - x_{k+1,j+1}\right) = B_{k+1} + L - y_{k+1,j+1} \end{aligned}$$

which establishes relation (B.1) for k+1 for this case.

***Case 2*:** $x^{hlb}_{k+1,j+1} = L - y^{hlb}_{k,j+1}$

Since from the inductive hypothesis on the main claim, $B^{hlb}_{k+1} \leq B_{k+1}$ for $k < j$,

$$B^{hlb}_{k+1} + L - y^{hlb}_{k+1,j+1} = B^{hlb}_{k+1} + L - \left(y^{hlb}_{k,j+1} + x^{hlb}_{k+1,j+1}\right) = B^{hlb}_{k+1} + L - \left(y^{hlb}_{k,j+1} + \left(L - y^{hlb}_{k,j+1}\right)\right) = B^{hlb}_{k+1} \leq B_{k+1} \leq B_{k+1} + L - y_{k+1,j+1}$$

establishing relation (B.1) for k+1 for this case.

***Case 3*:** $x^{hlb}_{k+1,j+1} = b_{c(j+1)}T_{k+1,j+1} - y^{hlb}_{k,j+1}$

From the inductive hypothesis on the main claim, $B^{hlb}_{k+1} \leq B_{k+1}$ for $k < j$. Also, since $T_{k+1,j+1}$ is the time from the arrival of request j+1 until the deadline of request k+1, and $y_{k+1,j+1}$ is the total amount of data received by the request j+1 client from transmissions also received by at least one other client, with request indexed at most k+1, we must have $y_{k+1,j+1} \leq b_{c(j+1)}T_{k+1,j+1}$. Therefore,

$$\begin{aligned} B^{hlb}_{k+1} + L - y^{hlb}_{k+1,j+1} &= B^{hlb}_{k+1} + L - \left(y^{hlb}_{k,j+1} + x^{hlb}_{k+1,j+1}\right) = B^{hlb}_{k+1} + L - \left(y^{hlb}_{k,j+1} + \left(b_{c(j+1)}T_{k+1,j+1} - y^{hlb}_{k,j+1}\right)\right) = B^{hlb}_{k+1} + L - b_{c(j+1)}T_{k+1,j+1} \\ &\leq B_{k+1} + L - b_{c(j+1)}T_{k+1,j+1} \leq B_{k+1} + L - y_{k+1,j+1} \end{aligned}$$

establishing relation (B.1) for k+1 for this case.

***Case 4*:** $x^{hlb}_{k+1,j+1} = b_{c(j+1)}T_{k+1,k+1}$

This case is divided into sub-cases depending on the other requests, if any, whose deadlines fall between the arrival time and the deadline of request k+1.

***Case 4.1*:** $x^{hlb}_{k+1,j+1} = b_{c(j+1)}T_{k+1,k+1}$, and there is no request i ($i \leq k$) such that $T^A_{k+1} < T^D_i \leq T^D_{k+1}$.

Since there are no clients with earlier request deadlines that are able to share the transmissions required for request k+1, $B_{k+1} = B_k + L$. From this fact together with $x^{hlb}_{k+1} \leq L$, $y_{k+1,j+1} - y_{k,j+1} = x_{k+1,j+1} \leq b_{c(j+1)}T_{k+1,k+1}$, and since relation (B.1) holds for k from the inductive hypothesis, we have

$$\begin{aligned} B^{hlb}_{k+1} + L - y^{hlb}_{k+1,j+1} &= \left(B^{hlb}_k + x^{hlb}_{k+1}\right) + L - \left(y^{hlb}_{k,j+1} + x^{hlb}_{k+1,j+1}\right) = \left(B^{hlb}_k + L - y^{hlb}_{k,j+1}\right) + \left(x^{hlb}_{k+1} - b_{c(j+1)}T_{k+1,k+1}\right) \\ &\leq \left(B_k + L - y_{k,j+1}\right) + \left(L - b_{c(j+1)}T_{k+1,k+1}\right) = \left(B_k + L\right) + L - \left(y_{k,j+1} + b_{c(j+1)}T_{k+1,k+1}\right) \leq B_{k+1} + L - y_{k+1,j+1} \end{aligned}$$

which establishes relation (B.1) for k+1 for this case.

***Case 4.2*:** $x^{hlb}_{k+1,j+1} = b_{c(j+1)}T_{k+1,k+1}$, and there is at least one request i ($i \le k$) such that $T^A_{k+1} < T^D_i \le T^D_{k+1}$ and such that $x^{hlb}_{i,j+1} = L - y^{hlb}_{i-1,j+1}$.

This case cannot occur since $x^{hlb}_{i,j+1} = L - y^{hlb}_{i-1,j+1}$ would imply that $x^{hlb}_{k+1,j+1} = 0$, in contradiction to the assumption that $x^{hlb}_{k+1,j+1} = b_{c(j+1)}T_{k+1,k+1}$.

***Case 4.3*:** $x^{hlb}_{k+1,j+1} = b_{c(j+1)}T_{k+1,k+1}$, and there is at least one request i ($i \le k$) such that $T^A_{k+1} < T^D_i \le T^D_{k+1}$ and such that $x^{hlb}_{i,j+1} = b_{c(j+1)}T_{i,j+1} - y^{hlb}_{i-1,j+1}$.

Since $x^{hlb}_{k+1,j+1} = b_{c(j+1)}T_{k+1,k+1}$ and $y^{hlb}_{k,j+1} \ge y^{hlb}_{i,j+1}$, we have $y^{hlb}_{k+1,j+1} = y^{hlb}_{k,j+1} + x^{hlb}_{k+1,j+1} \ge y^{hlb}_{i,j+1} + b_{c(j+1)}T_{k+1,k+1}$. Since $x^{hlb}_{i,j+1} = b_{c(j+1)}T_{i,j+1} - y^{hlb}_{i-1,j+1}$, and therefore $y^{hlb}_{i,j+1} = y^{hlb}_{i-1,j+1} + x^{hlb}_{i,j+1} = b_{c(j+1)}T_{i,j+1}$, this yields $y^{hlb}_{k+1,j+1} \ge b_{c(j+1)}T_{i,j+1} + b_{c(j+1)}T_{k+1,k+1}$. Using $T_{k+1,k+1} > T_{k+1,j+1} - T_{i,j+1}$, and the fact that $b_{c(j+1)}T_{k+1,j+1} \ge y_{k+1,j+1}$, this implies that $y^{hlb}_{k+1,j+1} > y_{k+1,j+1}$. Together with the inductive hypothesis on the main claim, $B^{hlb}_{k+1} \le B_{k+1}$ for $k < j$, this yields

$$B^{hlb}_{k+1} + L - y^{hlb}_{k+1,j+1} \le B_{k+1} + L - y^{hlb}_{k+1,i+1} < B_{k+1} + L - y_{k+1,j+1}$$

which establishes relation (B.1) for k+1 for this case.

***Case 4.4*:** $x^{hlb}_{k+1,j+1} = b_{c(j+1)}T_{k+1,k+1}$, and all requests i ($i \le k$) such that $T^A_{k+1} < T^D_i \le T^D_{k+1}$ (of which there is at least one), are such that $x^{hlb}_{i,j+1} = x^{hlb}_i$.

Let $n > 0$ denote the number of such requests, indexed k+1-n through k. Given that $x^{hlb}_{i,j+1} = x^{hlb}_i$ for each such request i,

$$B^{hlb}_{k+1} + L - y^{hlb}_{k+1,j+1} = \left(B^{hlb}_{k-n} + \sum_{i=k+1-n}^{k+1} x^{hlb}_i\right) + L - \left(y^{hlb}_{k-n,j+1} + \sum_{i=k+1-n}^{k+1} x^{hlb}_{i,j+1}\right) = B^{hlb}_{k-n} + L - y^{hlb}_{k-n,j+1} + \left(x^{hlb}_{k+1} - x^{hlb}_{k+1,j+1}\right).$$

Since relation (B.1) holds for k-n from the inductive hypothesis, this implies

$$B^{hlb}_{k+1} + L - y^{hlb}_{k+1,j+1} \le B_{k-n} + L - y_{k-n,j+1} + \left(x^{hlb}_{k+1} - x^{hlb}_{k+1,j+1}\right).$$

Using $x^{hlb}_{k+1,j+1} = b_{c(j+1)}T_{k+1,k+1}$ and the fact that $x^{hlb}_{k+1} \le L$ yields

$$B^{hlb}_{k+1} + L - y^{hlb}_{k+1,j+1} \le B_{k-n} + L - y_{k-n,j+1} + \left(L - b_{c(j+1)}T_{k+1,k+1}\right).$$

Consider now the total amount of data that the request j+1 client receives from transmissions also received by at least one of the clients with requests indexed k+1-n through k+1, but not received by any client with an earlier request deadline, i.e., $\sum_{i=k+1-n}^{k+1} x_{i,j+1}$. The portion of this data received after the arrival of request k+1 is upper bounded by $b_{c(j+1)}T_{k+1,k+1}$. The portion of this data received prior to the arrival of request k+1 is upper bounded by $\sum_{i=k+1-n}^{k+1} x_i - L$, since $\sum_{i=k+1-n}^{k+1} x_i$ gives the amount of data received by the clients with requests indexed k+1-n through k+1, from transmissions not received by any client with an earlier request deadline, and at least L of this data must be transmitted after the arrival of request k+1 so as to serve this request. (Note that all of the data received by the request k+1 client, must be from transmissions not received by any client with a request deadline earlier than that of request k+1-n, since such deadlines occur prior to the arrival of request k+1.) Thus,

$\sum_{i=k+1-n}^{k+1} x_{i,j+1} \le b_{c(j+1)} T_{k+1,k+1} + \sum_{i=k+1-n}^{k+1} x_i - L$, or $L - b_{c(j+1)} T_{k+1,k+1} \le \sum_{i=k+1-n}^{k+1} x_i - \sum_{i=k+1-n}^{k+1} x_{i,j+1}$, yielding, when applied with the previous relation,

$$B_{k+1}^{hlb} + L - y_{k+1,j+1}^{hlb} \le B_{k-n} + L - y_{k-n,j+1} + \left(\sum_{i=k+1-n}^{k+1} x_i - \sum_{i=k+1-n}^{k+1} x_{i,j+1} \right) = B_{k+1} + L - y_{k+1,j+1}$$

and establishing relation (B.1) for k+1 for this case.

Case 4.5: $x_{k+1,j+1}^{hlb} = b_{c(j+1)} T_{k+1,k+1}$, and all requests i ($i \le k$) such that $T_{k+1}^A < T_i^D \le T_{k+1}^D$ are such that either $x_{i,j+1}^{hlb} = b_{c(j+1)} T_{i,i}$ or $x_{i,j+1}^{hlb} = x_i^{hlb}$, with at least one having $x_{i,j+1}^{hlb} = b_{c(j+1)} T_{i,i}$.

Define two requests indexed p, q ($p < q$, and thus $T_p^D \le T_q^D$) as overlapping if $T_q^A < T_p^D$. Define "(in)directly overlapping" as the transitive closure of the overlapping relation. Let U denote the set of requests that are (in)directly overlapping with request j+1 when considering only request j+1 and those requests i such that $i \le k+1$ and $x_{i,j+1}^{hlb} = b_{c(j+1)} T_{i,i}$. Note that $|U| \ge 2$, since by the assumptions of this case, request k+1 is in U as is at least one other request. Let the index of the request in U with the earliest arrival time be denoted by e. Note that if $T_e^A \le T_{j+1}^A$, then, since $B_{k+1}^{hlb} \le B_{k+1}$ for $k < j$ from the inductive hypothesis on the main claim,

$$B_{k+1}^{hlb} + L - y_{k+1,j+1}^{hlb} \le B_{k+1}^{hlb} + L - b_{c(j+1)} T_{k+1,j+1} \le B_{k+1} + L - b_{c(j+1)} T_{k+1,i+1} \le B_{k+1} + L - y_{k+1,j+1},$$

which would establish relation (B.1) for k+1 for this case. Assume in the following that $T_e^A > T_{j+1}^A$.

Let V denote the set of requests i ($i \le k$) such that $T_e^A < T_i^D \le T_{k+1}^D$ and such that $x_{i,j+1}^{hlb} \ne b_{c(j+1)} T_{i,i}$. No request $i \in V$ can have $x_{i,j+1}^{hlb} = L - y_{i-1,j+1}^{hlb}$, by the same reasoning as used for case 4.2 above. Also, if for at least one request $i \in V$, $x_{i,j+1}^{hlb} = b_{c(j+1)} T_{i,j+1} - y_{i-1,j+1}^{hlb}$, then relation (B.1) is established for k+1 for this case using similar reasoning as used for case 4.3 above, i.e., from $y_{i,j+1}^{hlb} = y_{i-1,j+1}^{hlb} + x_{i,j+1}^{hlb} = b_{c(j+1)} T_{i,j+1}$ and $y_{k+1,j+1}^{hlb} > b_{c(j+1)} T_{k+1,j+1} \ge y_{k+1,j+1}$. Thus, in the following, assume that $x_{i,j+1}^{hlb} = x_i^{hlb}$ for each request $i \in V$.

Let B_U denote the total amount of data in the transmissions received by one or more of the set U clients. Note that these transmissions would be sufficient for serving a shorter request stream including *only* the requests in the set U. Therefore, from the inductive hypothesis on the main claim, B_U is lower bounded by the total amount of transmitted data that would be computed by the (more general, with the ε_j) heterogeneous lower bound algorithm, when applied to this reduced request stream. Denote the values computed for the reduced request stream, and the ε_j values used in this computation, with the superscript "*". We claim that it is possible to choose the ε_j^* values such that for each request in the reduced stream, i.e. each request $i \in U$, $x_i^{hlb*} = x_i^{hlb}$. To see this, note that for the request $i_1 \in U$ with the earliest deadline (and thus the first request in the reduced request stream), $\varepsilon_{i_1}^*$ can be chosen as $L - x_{i_1}^{hlb}$. For the request $i_2 \in U$ with the next earliest deadline, note that $y_{i_1,i_2}^{hlb} \ge y_{i_1,i_2}^{hlb*}$, since the presence of requests i in the full request stream with deadlines prior to that of i_1, and the resulting nonnegative x_{i,i_2}^{hlb} values, cannot decrease y_{i_1,i_2}^{hlb}. Similarly, requests i intermediate between i_1 and i_2 in the full request stream contribute nonnegative x_{i,i_2}^{hlb} values, and thus $y_{i_2-1,i_2}^{hlb} \ge y_{i_1,i_2}^{hlb*}$, implying that $\varepsilon_{i_2}^*$ can be chosen as $y_{i_2-1,i_2}^{hlb} - y_{i_1,i_2}^{hlb*} + \varepsilon_{i_2}$. Similarly for the other requests in U; in general, the $\varepsilon_{i_l}^*$ values can be chosen in order of request deadline, such that $\varepsilon_{i_l}^* = y_{i_l-1,i_l}^{hlb} - y_{i_{l-1},i_l}^{hlb*} + \varepsilon_{i_l}$. Thus, $\sum_{i \in U} x_i^{hlb} = \sum_{i \in U} x_i^{hlb*} \le B_U$.

Let $m \geq 2$ denote $|U|+|V|$. From $x_{i,j+1}^{hlb} = x_i^{hlb}$ for each request $i \in V$, we have

$$B_{k+1}^{hlb} + L - y_{k+1,j+1}^{hlb} = \left(B_{k+1-m}^{hlb} + \sum_{i \in U} x_i^{hlb} + \sum_{i \in V} x_i^{hlb} \right) + L - \left(y_{k+1-m,j+1}^{hlb} + \sum_{i \in U} x_{i,j+1}^{hlb} + \sum_{i \in V} x_{i,j+1}^{hlb} \right)$$

$$= B_{k+1-m}^{hlb} + L - y_{k+1-m,j+1}^{hlb} + \left(\sum_{i \in U} x_i^{hlb} - \sum_{i \in U} x_{i,j+1}^{hlb} \right)$$

which implies, since relation (B.1) holds for k+1-m from the inductive hypothesis,

$$B_{k+1}^{hlb} + L - y_{k+1,j+1}^{hlb} \leq B_{k+1-m} + L - y_{k+1-m,j+1} + \left(\sum_{i \in U} x_i^{hlb} - \sum_{i \in U} x_{i,j+1}^{hlb} \right).$$

Since $x_{i,j+1}^{hlb} = b_{c(j+1)} T_{i,i}$ for each request $i \in U$, $b_{c(j+1)} T_{k+1,e} \leq \sum_{i \in U} x_{i,j+1}^{hlb}$. Together with $\sum_{i \in U} x_i^{hlb} \leq B_U$, this yields

$$B_{k+1}^{hlb} + L - y_{k+1,j+1}^{hlb} \leq B_{k+1-m} + L - y_{k+1-m,j+1} + \left(B_U - b_{c(j+1)} T_{k+1,e} \right).$$

Consider now the total amount of data that the request j+1 client receives from transmissions also received by at least one of the clients with requests indexed k+2-m through k+1, but not received by a client with an earlier request deadline, i.e., $\sum_{i=k+2-m}^{k+1} x_{i,j+1}$. The portion of this data received after the arrival of request e is upper bounded by $b_{c(j+1)} T_{k+1,e}$. The portion of this data received prior to the arrival of request e is upper bounded by $\sum_{i=k+2-m}^{k+1} x_i - B_U$, since $\sum_{i=k+2-m}^{k+1} x_i$ gives the amount of data received by the clients with requests indexed k+2-m through k+1, from transmissions not received by a client with an earlier request deadline, and at least B_U of this data is transmitted after the arrival of request e, as it is received by one or more set U clients. (Note that all of the data received by set U clients, must be from transmissions not received by any client with a request deadline earlier than that of request k+2-m, since such deadlines occur prior to the arrival of any of the set U clients.) Thus, $\sum_{i=k+2-m}^{k+1} x_{i,j+1} \leq b_{c(j+1)} T_{k+1,e} + \sum_{i=k+2-m}^{k+1} x_i - B_U$, or $B_U - b_{c(j+1)} T_{k+1,e} \leq \sum_{i=k+2-m}^{k+1} x_i - \sum_{i=k+2-m}^{k+1} x_{i,j+1}$, yielding, when applied with the previous relation,

$$B_{k+1}^{hlb} + L - y_{k+1,j+1}^{hlb} \leq B_{k+1-m} + L - y_{k+1-m,j+1} + \left(\sum_{i=k+2-m}^{k+1} x_i - \sum_{i=k+2-m}^{k+1} x_{i,j+1} \right) = B_{k+1} + L - y_{k+1,j+1}$$

which establishes relation (B.1) for k+1 for this case.

As the above cases are mutually exhaustive, relation (B.1) is established, and thus also the main claim.