Scalable Delivery of Web Pages Using Cyclic Best-Effort Multicast *

Kevin C. Almeroth
University of California
Santa Barbara, CA 93106-5110
almeroth@cs.ucsb.edu

Mostafa H. Ammar, Zongming Fei
Georgia Institute of Technology
Atlanta, GA 30332-0280
{ammar,fei}@cc.gatech.edu

Abstract

The World Wide Web (WWW) has gained tremendously in popularity over the last several years. In this work we explore the use of UDP, best-effort multicast as a delivery option. Reliability is achieved through repetitive, cyclic transmission of a requested page. This solution is expected to be most efficient when used for highly requested pages. We view this cyclic multicast technique as a delivery option that can be integrated with the traditional reliable unicast and recently proposed reliable multicast options. We first describe the architecture of an integrated web server employing all three delivery options. We then describe the cyclic multicast technique and consider the various procedures needed for its successful operation. We characterize the gains in performance achieved by our proposal through an extensive performance analysis and simulation of our technique by itself, and when integrated with the other delivery options. We also describe our experience with an implementation of a prototype cyclic multicast server and its performance over the Multicast Backbone (MBone).

1 Introduction

The web has gained tremendously in popularity over the last several years. Web administrators are struggling to upgrade their servers to handle the huge volumes of requests. One approach to handling the large volume of requests has been to simply buy more hardware. This solution is neither cost effective nor scalable, and will cease to be an option as the WWW continues its growth. Other solutions have been investigated including improvements in the HTTP protocol, the use of transparent server replication[1] and caching of hot pages[2]. At issue is how WWW pages can be delivered to increasing numbers of users given limited server capacity and network bandwidth.

Another technique that can be used in conjunction with caching and replication to improve scalability is multicast delivery. One specific approach that closely addresses the problem of end-to-end scalability has been proposed in our earlier work[3]. The proposed approach uses a reliable multicast protocol[4] to transmit popular pages to groups of users who make the same page request. Requests reaching the server for the same page can sometimes be aggregated and responded to by one multicast out of the server. Using multicast means the server needs to make fewer page retrievals and the network needs to carry less traffic.

In this work we explore the use of UDP multicast as an alternate delivery option. Reliability is achieved through repetitive, cyclic transmission of a requested page. This solution is expected to be most efficient when used for highly requested pages, typically a popular site's top level page. Current servers can find that they are always in the process of transmitting such pages. Our approach thus accepts this fact and continuously multicasts the page without needing requests to drive such transmissions. UDP multicast has two important advantages over reliable multicast. First, aside from the initial request, there is no need for users to communicate with the server. This eliminates the overhead associated with a feedback channel, packet acknowledgments and retransmissions. Heavyweight HTTP over TCP connections can be closed and resources used for receiving additional requests. Second, a significant portion of the Internet already supports UDP multicast. The Multicast Backbone (MBone)[5] has been successfully used for the wide-area, scalable delivery of real-time audio and video.

The concept of cyclic, multicast delivery has been used in other contexts primarily to improve the scalability of a service. Examples of such work are: 1) the studies on the performance of cyclic multicast in the context of teletext systems (an early technology designed to deliver information over TV channels)[6], 2) the work on the Datacycle Architecture as a means to provide scalable access to a large database[7], and 3) the use of cyclic delivery over a broadcast media to provide a Broadcast Disk[8].

*This work is supported in part by research grants from IBM, Intel, and NSF under contract number NCR-9628379

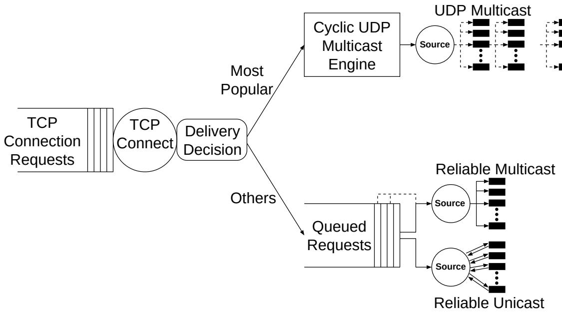

Figure 1: Integrated WWW architecture.

2 Integrated WWW Architecture

Figure 1 shows the architecture of a WWW server capable of delivering pages using cyclic multicast, reliable multicast, and reliable unicast. Requests for TCP connections arrive from page requesters and are queued until the server can process them. When the server establishes the connection, the user transmits the page request. At this point, the server decides which protocol will be used to serve the request. The decision is based mainly on the popularity of the requested page. Extremely popular pages are served via cyclic multicast. Moderately popular pages are served using reliable multicast. Other pages are served in the traditional way using unicast TCP connections. The decision on which pages to serve via cyclic multicast can be user controlled (through commands sent as part of the request) or can be driven by information that the server maintains in a dynamic or static fashion expressing the popularity of certain pages at the current time.

A typical web server would be limited as to how many simultaneous outgoing streams it can support. We envision reserving a certain number of those streams for the cyclic multicast engine's use. In a well-designed system these reserved streams would be utilized well (see Section 5).

3 Cyclic Multicast

The cyclic multicast engine is most effectively used to deliver a site's most popular and heavily requested pages. The engine is capable of delivering multiple pages simultaneously using multiple multicast groups. We only describe here the engine's delivery of a single page. This operation is replicated for every page designated for delivery via cyclic multicast.

The cyclic multicast delivery scheme is summarized below. Each step is further described in the following subsections.

1. The page to be delivered including all embedded files is divided into a number of *chunks*.
2. All chunks in a page are sequentially transmitted from the server to the group of receivers using an agreed-upon multicast address. One transmission of all chunks in a page constitutes one *cycle*.
3. Receivers join the appropriate multicast group and remain a member until an error-free copy of each chunk has been received. If a chunk is not received the receiver remains part of the group until the missed chunk is re-transmitted in a subsequent cycle and correctly received.
4. The server continues cyclic transmission as long as it believes there is at least one requester trying to receive the page. This *stopping condition* is estimated using a formula based on our analysis described in Section 4.

3.1 Chunking

The division of a page into chunks helps the receiver re-construct the WWW page as it is delivered. Chunks can then be further subdivided into one or more data packets. Chunking can range in complexity from simple page segmentation to options that include page structure information in each chunk. The simplest form of chunking, page segmentation, creates chunks of uniform length independent of page content. An alternate form of chunking uses HTML tags (where they exist) and embedded file markers to break a page into displayable groups of chunks. An image, table, list, or other HTML object will only be displayed when all chunks in the group have been received.

In addition to chunking, Forward Error Correction (FEC)[9] can be used to re-construct lost chunks. In an FEC protocol, data is transmitted with redundancy across chunks so that if some chunks are lost, redundancy in other chunks can be used to reconstruct the original data[10].

3.2 Multicast Address Determination

The multicast group address is chosen using one of two methods. When the first request for a particular page is received, the server chooses a random address. As requests are made, the server informs requesters *explicitly* via the TCP connection used to make the request. The second option uses a hash function to convert the page's Universal Resource Locator (URL) to a multicast group. The same hash function is used by both the server and all receivers. For this option, a requester need only make a request if, after joining the multicast group, it discovers the server is not transmitting the page. However, this gives no information to the server about how many requests there are. A compromise solution would have receivers immediately join a multicast group (thereby achieving very fast response if the server is actually transmitting) and then actually make the request to the server. Additional merits and limitations of these methods are discussed extensively in [3]. In either case, the TCP connection can be closed quickly and the resources used to handle other incoming requests.

3.3 Requester Operation

The initial challenge for requesters is determining how the server will deliver a page. If the server informs receivers explicitly (over the TCP connection used to make the request), the task is quite simple. However, if no such exchange is used then the requester must be prepared to listen on the open TCP connection, on a multicast address used by a reliable protocol, and on another address for cyclic multicast. Listening on three network connections is not a resource intensive task but it does add complexity to browser operation.

If delivery is via cyclic multicast, requesters will start to receive chunks as soon as the server starts transmitting them. If the server is already transmitting, the receiver will see data immediately after joining the multicast group. If an error occurs or a chunk is missed, the receiver must wait until the chunk is re-transmitted in the next cycle. If the browser is capable, the page may be displayed in non-contiguous segments. After receiving all chunks, each requester will leave the multicast group. In response, routers will prune the receiver from the multicast tree, thereby preventing the delivery of unwanted packets.

3.4 Transmission Duration

If all receivers have been satisfied and there are no outstanding requests, the cyclic multicast server should stop transmitting. However, without explicit feedback, the server cannot know for certain if there are still any receivers. Our solution is to use the arrival time of all requests and an estimate of the network error probability to estimate, with high certainty, the number of cycles needed to satisfy all receivers. A formula for this estimate and mechanisms for estimating error conditions are given in the next section.

4 Cyclic Multicast Analysis

We now present an analysis that is aimed at understanding and comparing the performance of cyclic multicast with reliable unicast. The analysis also helps us develop an estimator for the number of cycles needed to achieve a high probability of satisfying all receivers in a cyclic multicast system.

We consider the transmission of a web page that is broken down into C equal-size chunks. We assume that, in both the unicast and cyclic multicast cases, transmissions out of the server are in packets with each packet containing one chunk. In our analysis we assume that there are K receivers who make requests for the same page at the same time. The probability that any chunk (or packet) is received correctly is q_c and we assume that the loss is independent from one chunk to the next and among the K receivers. The probability that a chunk is lost in our analysis covers all end-to-end conditions including errors and loss due to congestion.

For reliable unicast retransmission we assume a best-case selective reject ARQ protocol which transmits only lost chunks. We also ignore the overhead of connection and disconnection. We thus have that the number of chunks (or packets) that need to be transmitted in order for all K receivers to receive all chunks of the page is

$$C_u = \frac{K * C}{q_c} \qquad (1)$$

For the cyclic multicast system we assume that the server will continue to cycle through the chunks until the probability that all K receivers have received all chunks is greater than or equal to some *certainty threshold*, β. Let γ be the number of cycles required to achieve this certainty threshold. The number of chunks transmitted will be[1] $C_m = C * \gamma$.

Recall that for the purposes of this analysis we assume that all K receivers make their request at the same time and, therefore, they will all be waiting just before the beginning of the transmission of the first cycle. In order to determine γ we use an analysis technique similar to [11]. A discrete time Markov chain is used to represent the progress of the system. The state of the system is the number of receivers that have received a particular chunk. Given that i receivers have received a particular chunk at the end of cycle t, the probability that $i + r$ receivers receive the chunk at the end of cycle $t + 1$ is given by:

$$a_{(i,i+r)} = \binom{K-i}{r} (q_c)^r * (1-q_c)^{(K-i-r)} \qquad (2)$$

Figure 2 shows a representation of the Markov chain for one chunk. The transitions between states are given in terms of Equation 2, and the end state is achieved when all receivers have received the chunk.

Let $P(n,t)$ denote the probability that n receivers have received a particular chunk by the end of cycle t. $P(n,t)$ is given by

$$P(n,t) = \sum_{i=0}^{n} a_{(i,n)} * P(i,t-1) \qquad (3)$$

For $P(n,t)$ the following initial conditions apply:

$$P(0,0) = 1 \text{ and } P(i,0) = 0 \text{ for } i > 0 \qquad (4)$$

[1] We only consider stopping at cycle boundaries and therefore only consider integer values for γ.

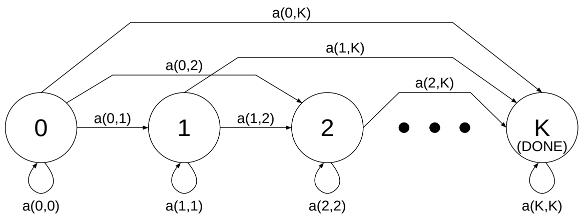

Figure 2: Markov chain for satisfied receivers.

Let $P_{DONE}(K, C, t)$ represent the probability that all K receivers receive all C chunks by the end of cycle t. Because we assumed independence of loss we have:

$$P_{DONE}(K, C, t) = [P(K, t)]^C \quad (5)$$

The minimum number of cycles, γ, required to meet the certainty threshold, β, for delivery to all receivers is determined by computing $P_{DONE}(K, C, t)$ with increasing values of t. γ is the smallest value of t such that $P_{DONE}(K, C, t)$ is greater than β.

4.1 Numerical Examples

Figure 3 shows the number of chunk transmissions needed to satisfy different numbers of receivers (x-axis) with $q_c = 0.95$ for both the cyclic multicast and the reliable unicast approaches. For the cyclic multicast system we use $\beta = 99.9\%$. In this case, 5 receivers is the "break-even" point where cyclic multicast transmits fewer chunks than reliable unicast.

Figure 4 shows for $K = 50$ the cumulative probability that all receivers are satisfied by the end of each cycle for four different chunk loss probability $(1 - q_c)$. As expected, the lower the probability of error, the fewer cycles required to achieve a high certainty that all receivers are satisfied. Even at high loss probabilities the number of cycles is quite reasonable. From Figure 4, even with a loss probability of 20% the cyclic multicast server need only send the page 8 times.

At this point, let us briefly consider the cost of unicast vs multicast delivery. We have just shown that

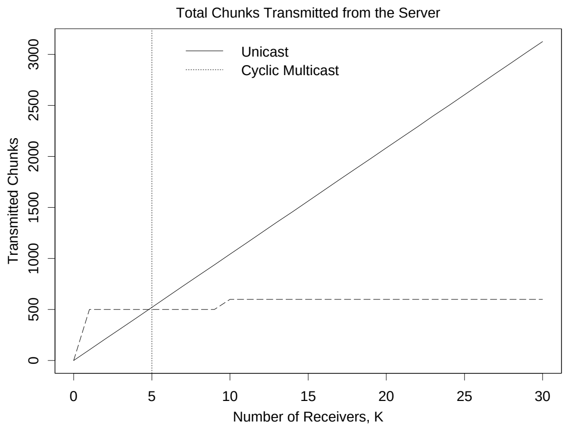

Figure 3: Chunks needed to complete a transmission.

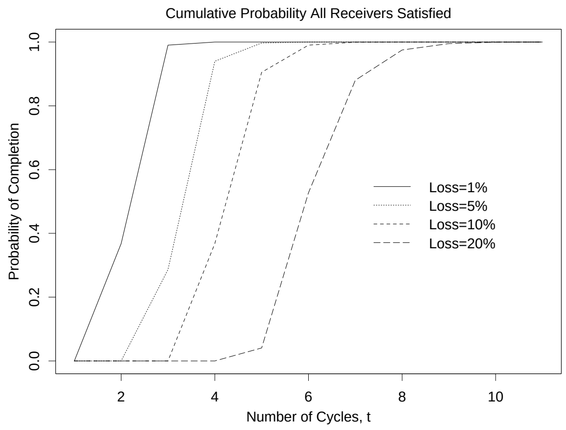

Figure 4: Cycles needed to complete a transmission.

cyclic multicast reduces the number of chunks transmitted from the server. However, a chunk sent to a multicast group will consume more network hops than a chunk sent to one receiver via unicast. Determining an estimate of network resources consumed by unicast versus multicast is beyond the scope of this paper but is discussed in other work[12]. Clearly, though, the amount of bandwidth consumed in multicasting a page to N receivers is at most equal to that required to unicast the page to the same N receivers.

4.2 Determining When to Stop

Based on the analysis results in this section we propose a heuristic scheme to estimate the number of transmission cycles needed to satisfy all receivers. Our objective is to determine a *stopping condition* that would allow a cyclic multicast server to stop transmitting when there are no more receivers. One of the main differences between the model used for our analysis and a real system is that in a real system, requests continue to arrive after the initial set of requests. The procedure is as follows:

1. At the end of each cycle determine how many new requests were made during the cycle.

2. Compute (using the analysis provided earlier) the number of cycles needed to satisfy these new requests for the given certainty threshold, β.

3. Take the maximum of the number computed in Step 2 and the value computed during the previous cycle decremented by one (to account for the cycle just completed).

4. If the maximum computed in Step 3 is 0 then stop transmission of the page. Otherwise, transmit and repeat starting with Step 1.

The computation in the analysis (needed in Step 2) requires an estimate of the packet loss rate. An exact loss value will be difficult to determine because losses in the Internet can vary over a wide range for any given receiver. Furthermore, losses will typically vary among different receivers in a group. However, we expect to make a rough estimate based on a feedback scheme similar to that used by the Real-Time Control Protocol (RTCP)[13]. We should be able to estimate a value fairly accurately for each receiver. These values can then be combined using a pessimistic estimate and the assumption that receivers with very high loss, i.e. greater than 20%, will most likely have to request a page by unicast. One difference between RTCP and our error estimation protocol will be that each receiver's estimate will be sent via unicast back to the server using a random interval based on the number of receivers in the multicast group.

Note that the procedure described above does not guarantee that all receivers will have been satisfied when the server stops transmission. However, it can insure that the probability of that event is high. In the unlikely case where the server stops transmitting even though there are still unsatisfied receivers, they can request that multicast transmission be re-started or can request that the server use reliable unicast.

5 Simulation of Cyclic Multicast

In this section we present the results of simulations for an integrated web server that includes cyclic multicast, reliable multicast and reliable unicast. Our objective is to provide a first order understanding of the response time experienced by clients using such a server and how that compares with their experience using a unicast-only server.

5.1 Simulation Model

The server in our simulation has a maximum number of simultaneous outgoing streams that it can support. We assume that from the point of view of a server, streams carrying cyclic multicast, reliable multicast and reliable unicast are equivalent and interchangeable.

We assume a large number of potential clients that make requests according to a Poisson process and that all requested pages are of the same size[2]. Each server will have a number of pages L that it can serve. We assume that the probability that a request for a particular page follows a Zipf distribution[14]. That is, if we label the pages in decreasing order of popularity, the probability that a particular request is for page i is given by κ/i where κ is the normalization constant $[\sum_i^L 1/i]^{-1}$.

[2]Although this assumption deviates significantly from the actual workloads experienced by Web servers it simplifies matters for our purposes and allows us to focus on the problem of interest; namely the interaction of the different types of delivery mechanisms within an integrated server.

Parameters	Values Tested
Duration	10000 events
Request Rate	4.5 to 50 req per sec
Page Selection	Zipf Distribution
Total Pages	5, 100, 1000, 10000
Size of Page	50K (100 chunks)
Server Streams	5, 15, 20, 25, 35
Packet Loss	1% to 50%
Certainty	90% to 99.99%

Table 1: The simulation parameters.

We use the time to transmit a page in chunks (i.e., the time for transmitting one cycle) as our time unit. We further assume there is no propagation delay in making a request, or in sending chunks from the server. Requests are queued and wait if the server does not have an available stream.

Reliable unicast transmissions out of the server use a selective reject ARQ protocol as described in Section 4. The operation of the two multicast protocols is as follows. For reliable multicast, initial transmissions are made to the entire group and missed packets are re-transmitted on an individual basis in a selective reject manner. For cyclic multicast, the first request for a particular page starts the cyclic multicast engine. The engine uses the procedure described in Section 4.3 to calculate when to stop cycling through the page's chunks based on a defined certainty threshold. Table 1 summarizes the list of simulation parameters and the range of values tested.

5.2 Performance Measures

In our simulations, the most important performance measure is *response time* defined as the time it takes to completely receive a page, measured in cycles. A cycle in the multicast case is simply the time to transmit all chunks in a page. For unicast, a cycle is the time to transmit all chunks to a single user assuming no packet loss.

Although, the focus of the simulation is to compare response time, the number of packets transmitted is still important. However, our simulated systems almost always operate above the breakeven number of receivers described in Section 4. Another important performance measure is the number of receivers per multicast group. This measures the success of aggregating requesters. The more receivers per group, the better scalability characteristics of the WWW server.

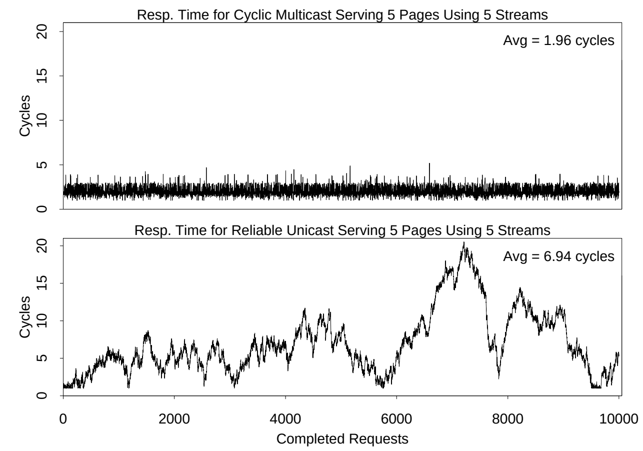

Figure 5: Comparison of response times.

5.3 Simulation Results

5.3.1 Cyclic Multicast vs. Reliable Unicast

Figure 5 shows the response time for each of 10000 simulated and completed requests. The systems being compared are cyclic multicast and reliable unicast. Both servers can deliver up to five streams simultaneously and both offer five different WWW pages. The probability of packet loss is 5%. The request arrival rate for this graph is 4.76 requests per second which is just below the maximum rate that the unicast server can handle. This maximum rate can computed using Little's Theorem and is as follows:

$$Request\_Rate_{Max} = \frac{Server\_Streams}{Service\_Time} \quad (6)$$

$$= \frac{q_c * Server\_Streams}{(K-1) * Receiver\_Rate} \quad (7)$$

A request rate any greater than the max rate would cause the unicast server to be unstable, i.e., it would receive requests faster than it could service them.

5.3.2 Performance of Integrated Servers

Figure 6 shows results for a system combining cyclic multicast and reliable unicast to satisfy user requests. The server has the capacity to support 35 simultaneous streams, five of which are reserved for the cyclic multicast of the most popular five pages. The request arrival rate is 40 requests per time unit, and the number of pages supported by the system is 1000. Of the 1000 pages, only the most popular five are serviced by the cyclic multicast engine. The chunk loss probability in this system is set to 5%. The top graph of Figure 6 shows the response time for each request completed via the cyclic multicast engine. The average response time for these requests is 1.96 cycles. The second graph from the top shows that response time for unicast pages is relatively well behaved with an average of 2.48 cycles. The third graph from the top shows the number of streams in use right after each request is satisfied. The system is operating at near capacity with all channels, both unicast and multicast, almost always fully utilized. The bottom graph shows the number of multicast group members for cyclic multicast. Multicast, with an average group size of 5.0, has several receivers in a group while unicast will only ever have one receiver per stream.

Figure 7 shows how performance improves by adding the ability to use reliable multicast. All but one of the parameters are the same as in Figure 6. Instead of 40 requests per time unit, the request rate was raised to 50 requests per time unit. Reliable multicast is used for requests made for pages 6 through 1000 when there are multiple requests for the same page in the queue at the time a request is serviced.

5.3.3 How Many Cyclic Multicast Pages?

We now look at the issue of deciding how many pages to deliver via cyclic multicast. In Figure 8 we show simulation results where we vary the total number of pages at the server, and the number of pages delivered using cyclic multicast. Results are given for a system that includes only cyclic multicast and reliable unicast (left column) and all three delivery options (right column). The graphs plot response time as a function of the number of pages designated for cyclic multicast. The systems have a limit of 35 streams with one stream reserved for each page designated for cyclic multicast. The request arrival rate is 50 requests per unit time and the packet loss probability is 5%.

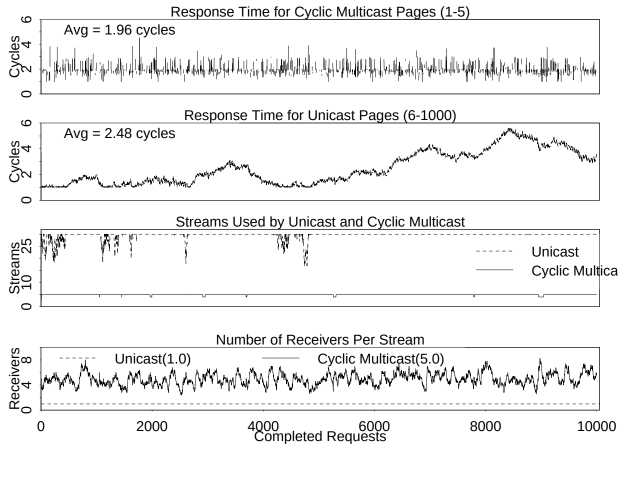

Figure 6: Performance of unicast vs. cyclic multicast.

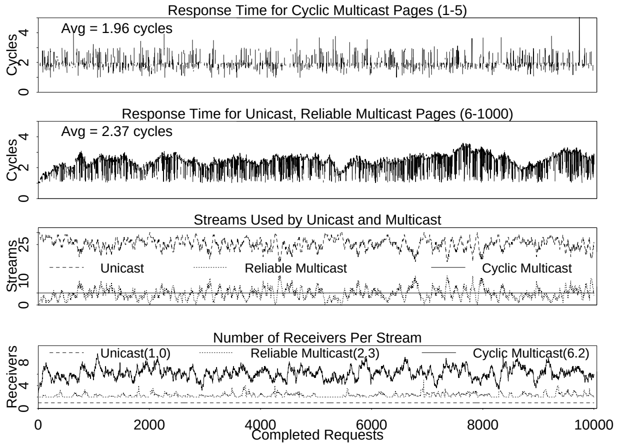

Figure 7: Performance for a combined system.

In general there is an optimum number of pages that should be designated for cyclic multicast. Designating more pages increases the overall average response time. This is because for each designated page there is one less stream available to the other pages, so unless this designated page is popular enough to utilize its allocated stream, system resources are wasted.

In comparing systems with and without reliable multicast (the left and right columns of the figure, respectively), we note that the addition of reliable multicast has several effects. First, as seen earlier it reduces the overall average response time. Second, it tends to increase the optimum number of pages that should be designated for cyclic multicast. Lastly, it reduces the response time penalty for not designating the optimum number of cyclic multicast pages. This is because streams not reserved for cyclic multicast can be utilized more efficiently when the reliable multicast option is available.

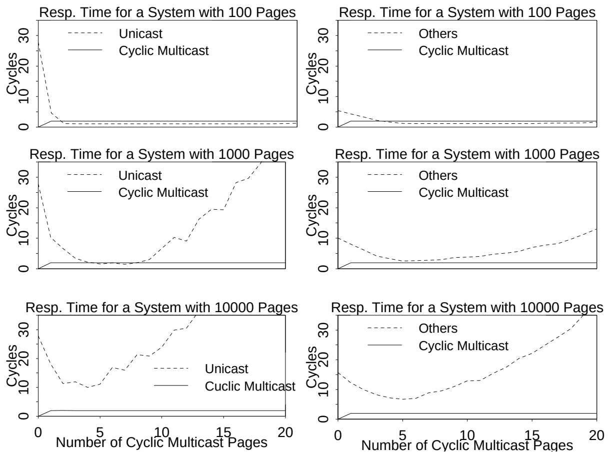

Figure 8: Varying pages sent via cyclic multicast.

6 Cyclic Multicast Implementation

We have implemented a prototype cyclic multicast engine in order to judge the practical feasibility of our proposed approach. Here we report on some preliminary measurements of performance from experiments using our prototype. We measure two components of user-perceived response time. The first is *reception time*, the time between when a receiver gets the first chunk of a page until all chunks have been received. The second component is the *first-packet time* defined as the time from when the user issues a request until the first chunk is received. For these two measures we are interested in the performance of the cyclic multicast prototype and how it compares with a typical unicast web server.

Our experiments were conducted over the MBone with a server at Georgia Tech and clients located at Georgia Tech (GT), the University of California at Los Angeles (UCLA), and the University of Massachusetts at Amherst (UMASS). The clients all request the same page which requires 50 chunks each of which is 512 bytes. Figure 9 shows the reception time at the three clients for different transmission rates. The reception time decreases quickly as the sending rate increases up to 128 Kbps. Between 128 Kbps and 480 Kbps the time to receive remains relatively flat. This version of cyclic multicast does not implement congestion control, and so as the transmission rate increases, chunk loss starts to increase. At 512 Kbps and above, the transmission rate increases to the point where congestion negatively affect reception time. As a performance benchmark, Figure 9 also includes the reception time for a page delivered via reliable unicast (actually HTTP over TCP) for an unloaded and loaded web server. Both the client and the server were at Georgia Tech in the unicast experiments. It should be noted that the cyclic multicast server reception times will be unaffected for differently loaded servers.

In the second set of experiments we measure the *first-packet time*. One client was placed at UCLA with the server at GT. For reliable unicast this time includes setting up the TCP connection, making the page request, and then waiting to receive the first chunk. The first line in Table 2 shows that on average (over 100 experiments) it takes 264 msecs to receive the first chunk.

For cyclic multicast, join time depends on whether the page is already being transmitted and whether there are any branches of the multicast tree close to the new receiver. In the worst case, the page is not being transmitted and there are no other receivers. In this scenario, the user will request the page via a TCP

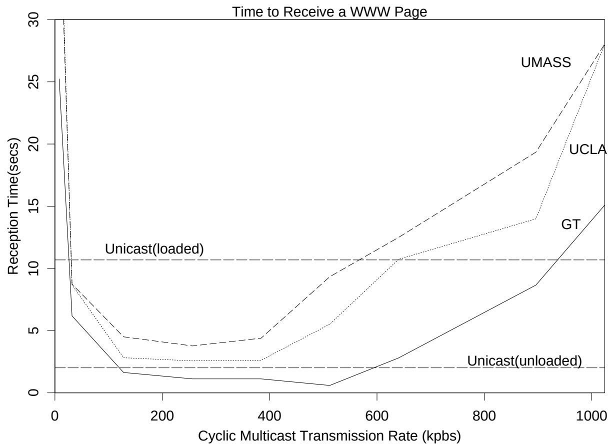

Figure 9: Observed time to receive WWW pages.

	Response Time	
	Avg	Std. Dev.
TCP (reliable unicast)	264 ms	135 ms
Multicast: one receiver	241 ms	57 ms
Multicast: recvr at UCLA	149 ms	68 ms

Table 2: Time from request to first packet.

connection while at the same time joining the appropriate multicast group. The time to receive the first chunk will be the maximum of (1) the time it takes for the server to receive the request and begin sending the page (comparable with the TCP join time value above), and (2) the time for the receiver to join the multicast group. The two cyclic multicast entries in Table 2 assume the page is being transmitted and measures the time to join the multicast group. The first entry measures the case where there are no receivers and the multicast tree must be constructed between UCLA and GT. The entry considers the case where there is another group member close by. We conclude here that the multicast join time will tend to be, in the worst case, on the same order as the TCP first-packet time. Our experiments with the prototype confirm that cyclic multicast of a page can drastically improve response time, especially for loaded servers.

7 Concluding Remarks

In this paper we examine the possibility of using cyclic best effort (UDP) multicast to deliver a WWW site's most frequently requested pages. Our proposed protocol uses UDP multicast with reliability achieved through repetitive, cyclic multicast transmission of a requested page. With this technique, aside from the initial request, there is no need for users to communicate with the server. Our analysis and simulation results demonstrate the performance gains possible through the use of this technique. Furthermore, our experience with a prototype implementation shows the practical feasibility of the approach. Our results also show that ideally, servers should integrate all three delivery options (cyclic and reliable multicast and reliable unicast) for best performance and flexibility.

References

[1] E. D. Katz, M. Butler, and R. McGrath, "A scalable HTTP server: the NCSA prototype," *Computer Networks and ISDN Systems*, vol. 27, pp. 155–164, 1994.

[2] H. Braun and K. Claffy, "Web traffic characterization: an assessment of the impact of caching documents from NCSA's web server," *Computer Networks and ISDN Systems*, vol. 28, pp. 37–51, Dec 1995.

[3] R. Clark and M. Ammar, "Providing scalable web service using multicast delivery," in *IEEE Workshop on Services in Distributed and Networked Environments*, (Whistler, Canada), June 1995.

[4] R. Talpade and M. Ammar, "Single connectin emulation: An architecture for providing a reliable multicast transport service," in *IEEE Conference on Distributed Computing Systems*, 1995.

[5] S. Casner, *Frequently Asked Questions (FAQ) on the Multicast Backbone (MBone)*. ISI, December 1994. Available from ftp://ftp.isi.edu/mbone/faq.txt.

[6] M. Ammar and J. Wong, "On the optimality of cyclic transmission in teletext systems," *IEEE Transactions on Communications*, vol. 35, pp. 68–73, January 1987.

[7] G. Herman, G. Gopal, K. Lee, and A. Weinrib, "The datacycle architecture for very high throughput database systems," in *ACM SIGMOD*, 1987.

[8] S. Acharya, M. Franklin, and S. Zdonik, "Dissemination based data delivery using broadcast disks," *IEEE Personal Communications*, vol. 2, Dec 1995.

[9] C. Huitema, "The case for packet level FCE," in *Proceedings of IFIP 5th International Workshop on Protocols for High Speed Networks*, (Sophia Antipolis, France), October 1996.

[10] J. Nonnenmacher, E. Biersack, and D. Towsley, "Parity-based loss recovery for reliable multicast transmission," Tech. Rep. 97-17, Department of Computer Science, Univ of Massachusetts, March 1997.

[11] M. Ammar, "Probabilistic multicast: Generalizing the multicast paradigm to improve scalability," in *Infocom '94*, (Toronto, Canada), June 1994.

[12] K. Almeroth and M. Ammar, "Multicast group behavior in the internet's multicast backbone (MBone)," *IEEE Communications*, June 1997.

[13] H. Schulzrinne, S. Casner, R. Frederick, and J. V., *RTP: A Transport Protocol for Real-Time Applications*, rfc 1889 ed., January 1996.

[14] G. Zipf, *Human Behavior and the Principle of Least Effort*. Reading, MA: Addison-Wesley, 1949.