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

Kevin C. Almeroth
Mostafa H. Ammar
Zongming Fei

Networking and Telecommunications Group
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
{kevin,ammar,fei}@cc.gatech.edu
(404)894-3292(office)
(404)894-0272(FAX)

May 1, 1997

Abstract

The World Wide Web (WWW) has gained tremendously in popularity over the last several years. Solutions to the problem of overloaded web servers have included buying more hardware, the use of transparent server replication and mirroring, and caching of hot pages. Another technique that can be used in conjunction with these other solutions is the use of multicast for the delivery of web pages. 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).

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

1 Introduction

The World Wide Web (WWW) 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. Further adding to the problem is the trend of WWW sites to offer more complex pages including flashy media like sound, graphics, pictures, and video. One straightforward 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[1], the use of transparent server replication[2] and caching of hot pages[3]. 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 the scalability of web access is the use of multicast for the delivery of web pages. One specific approach that closely addresses the problem of end-to-end scalability has been proposed in our earlier work[4]. The proposed approach uses a reliable multicast protocol[5] to transmit popular pages to groups of users who make the same page request. In this approach 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 fewer copies of the page.

In this work we explore the use of UDP multicast[6] 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, for example, CNN Interactive at http://cnn.com. 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 and this support is likely to expand in the future as wide scale multicast deployment efforts bear fruit.[1]. The Multicast Backbone (MBone)[7] has been successfully used for the wide-area, scalable delivery of real-time audio and video.

[1] See the IP Multicast Initiative at www.ipmulticast.com

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)[8, 9], 2) the work on the Datacycle Architecture as a means to provide scalable access to a large database[10], and 3) the use of cyclic delivery over a broadcast media to provide a Broadcast Disk[11].

We view this UDP cyclic multicast delivery as a third delivery option that can be integrated with the reliable unicast and multicast options explored earlier. We first describe in Section 2 how these three delivery options may be integrated in a web server. Next we focus in Section 3 on the cyclic UDP multicast technique and consider the various procedures needed for its successful operation. In Section 4 we analyze the basic cyclic multicast protocol and compare its performance to that of the use of reliable unicast (e.g., HTTP over TCP). In Section 5 we provide the results from an extensive simulation of an integrated server combining UDP multicast, reliable multicast and reliable unicast. We next describe in Section 6 our experience with an implementation of a cyclic UDP multicast server and its performance over the multicast backbone. The paper is concluded in Section 7.

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).

The use of a multicast protocol may not be appropriate at all times. In these cases, the user can specify that reliable unicast be used instead. This may happen if a user does not have multicast

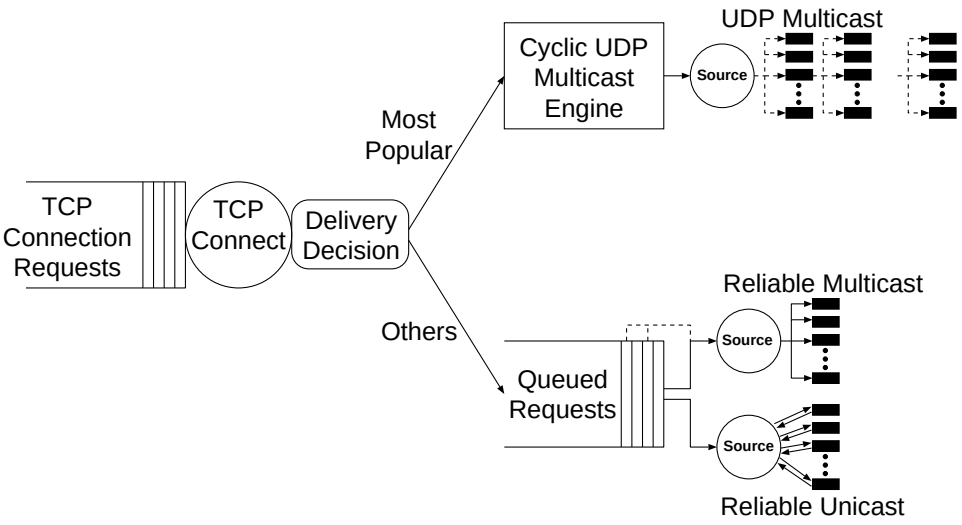

Figure 1: Integrated WWW architecture.

capability or has not been able to successfully receive a page via multicast. Servicing of requests using reliable multicast in conjunction with reliable unicast is described in [4]. The operation of the cyclic multicast component of this server is described next.

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. A single transmission of each and every chunk constitutes one *cycle*.
3. Receivers join the appropriate multicast group and remain a member until an error-free copy of all chunks 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. This is especially useful when chunks are received out of order due to network conditions including loss or congestion. 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 WWW page object will only be displayed when all chunks in the group have been received.

In addition to chunking, Forward Error Correction (FEC)[12, 13] 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[14].

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 [4]. In either case, the TCP connection can be closed soon after the request is made 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 numbered 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. Chunks or groups of chunks are displayed as soon as possible. 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[15].

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 that might cause a chunk not to be received 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 (this would tend to favor the reliable unicast scheme in our analysis). 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

$$C_u = \frac{K * C}{q_c} \tag{1}$$

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[2] $C_m = C * \gamma$.

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 [16]. 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:

$$a_{(i,i+r)} = \left(\begin{array}{c} K - i \\ r \end{array} \right) (q_c)^r * (1 - q_c)^{(K-i-r)} \tag{2}$$

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

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

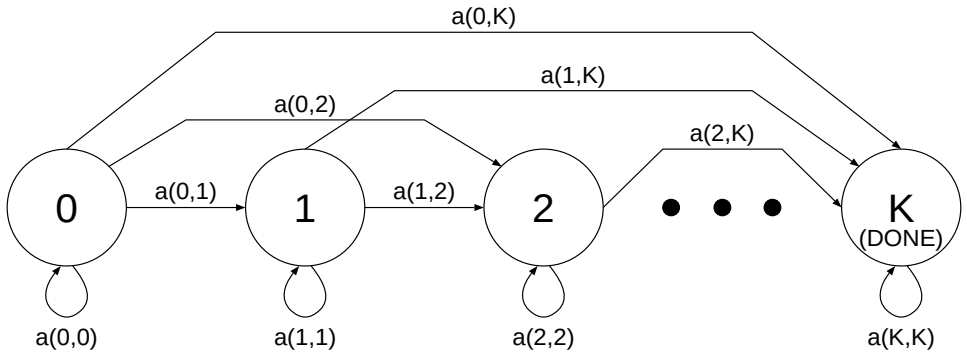

Figure 2: Markov chain representing satisfaction of multicast receivers.

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

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

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

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 among the receivers we have:

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

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. W can then compute the number of chunks transmissions needed as $C_m = C * \gamma$.

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 $(1 - q_c)$. Recall that one cycle represents a

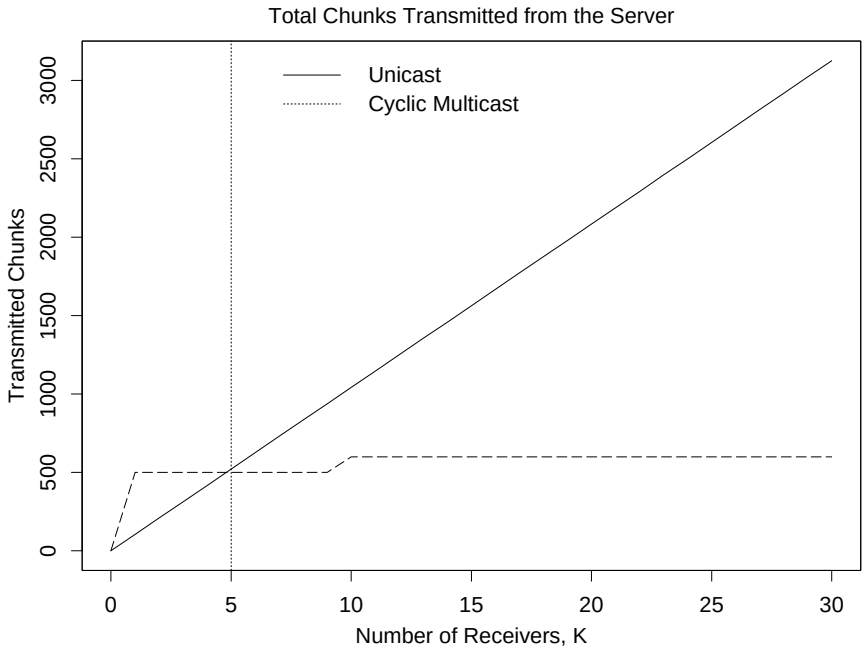

Figure 3: Number of chunks needed to complete a transmission.

single transmission of all chunks for a particular page. 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. Consider that for 50 receivers, a unicast server will need to send the page 50 times plus additional transmissions to recover from loss. 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 sending a unicast chunk versus sending a multicast chunk. We have just shown that 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 estimates of network resources consumed by unicast versus multicast is beyond the scope of this paper but results based on data collected from MBone groups are given in [17]. Clearly, though, the amount of bandwidth consumed in multicasting a page to N receivers is no more (and could be considerably less) that that required to unicast the page to the same N receivers.

Figure 5 shows the minimum number of multicast group members necessary for a cyclic multicast server to transmit fewer packets than a reliable unicast server given the packet loss probabilities $(1 - q_c)$ represented on the x-axis. At lower packet loss rates, cyclic multicast requires significantly

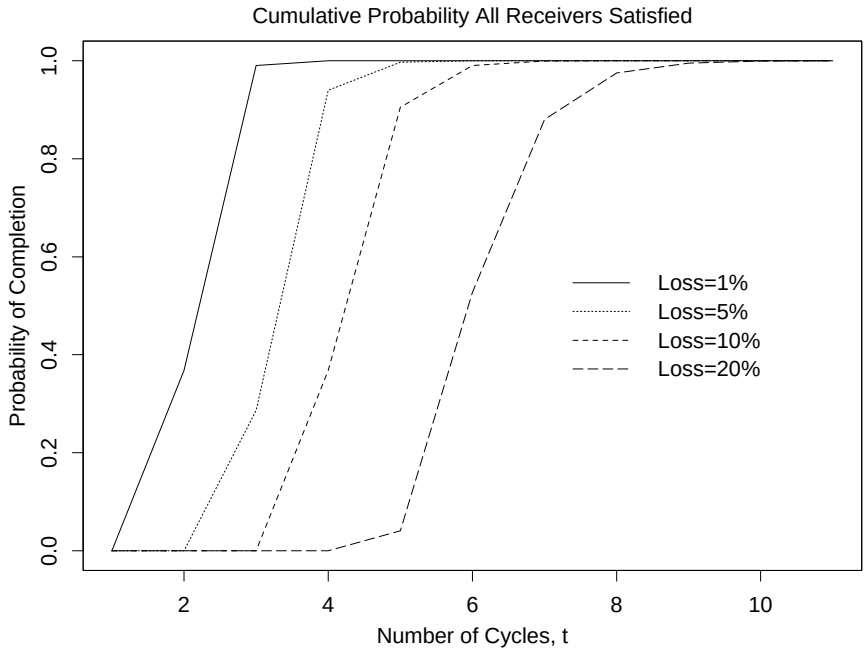

Figure 4: Number of cycles needed to complete a transmission.

fewer cycles and packets to satisfy receivers. This makes cyclic multicast much better at low loss rates. As the probability of loss increases so does the number of cycles and the minimum number of multicast receivers to achieve the break-even point.

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 a page at the point when no more requests for the page are made. One of the main differences between the model used for our analysis and the operation of a real system is that in a real system, requests will typically 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).

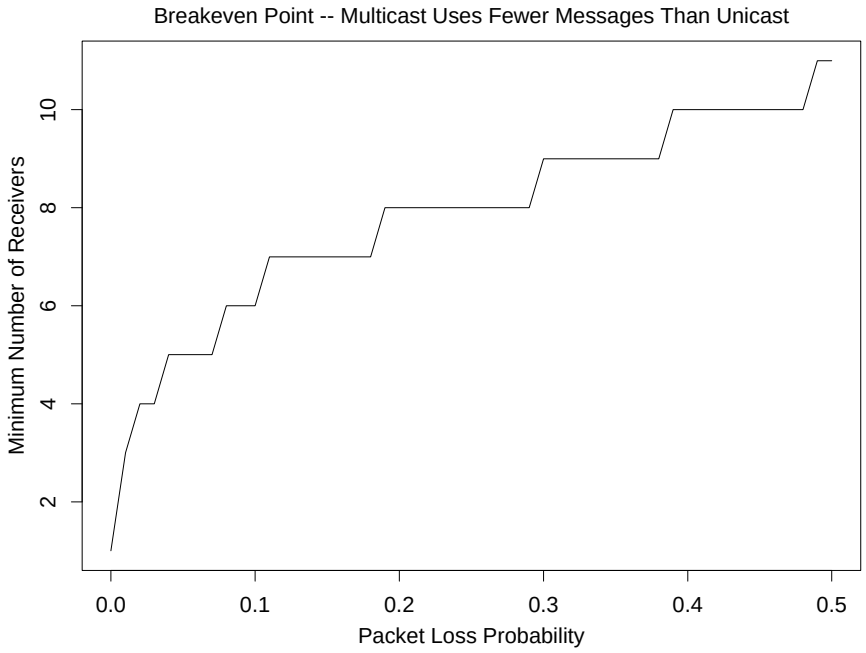

Figure 5: Breakeven point for multicast.

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

For example, Consider a server that received 10 requests for a particular page during the last cycle, and computed that five cycles were required to satisfy these 10 requests. At the end of the current cycle the server receives only one new request for the same page and it computes that complete reception for this new request would require two cycles. The server will continue to transmit for four more cycles: the maximum of 4 (5 minus 1 for the cycle that just ended) and 2. If, in the next cycle, no more requests are received the server will decrement this new value until it reaches zero. At this point the server believes all receivers have been satisfied and will stop transmitting.

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 may be in bursts, and may vary widely 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)[18]. 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. Experience with the prototype will be used to refine our estimation technique.

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 provide delivery using reliable unicast.

5 Simulated Performance of Servers Using Cyclic Multicast

In this section we present the results of simulations of the performance of 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[3]. Each server will have a number of pages L that it can serve. We assume that the probability that a request is for a particular page follows a Zipf distribution[19]. 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 $[\sum_i^L 1/i]^{-1}$.

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.

[3] 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.

Simulation Parameter	Range of Values Tested
Simulation Duration	10000 events
Request Arrival Rate	4.5 to 50 requests per second
Page Selection	Zipf Distribution
Number of Pages	5, 100, 1000, 10000
Size of Page	50K divided into 100 chunks
Number of Server Streams	5, 15, 20, 25, 35
Packet Loss Probability	1% to 50% (Nominal value = 5%)
Estimator Certainty	90%, 99%, 99.9%, 99.99% (Nominal value = 99.9%)

Table 1: The simulation parameters, and the nominal value for each.

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. In the unlikely case where the server stops transmitting even though there are still unsatisfied receivers, they will issue another request (after a timeout) that will cause the multicast transmission to be re-started.

Table 1 summarizes the list of simulation parameters and shows the ranges and nominal values we used in our simulation experiments.

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.

Other useful performance measures are less important or less interesting but worth briefly mentioning. 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.

Our results also show other useful performance measures. First, the number of receivers per multicast group measures the success of aggregating requesters. The more receivers per group,

the better scalability characteristics of the WWW server. Second, there is a low probability that the server will stop transmitting a WWW page even though there are still receivers, causing non-delivery for receivers who are still waiting. This probability can be kept low by increasing the certainty threshold that our stopping algorithm works with but the tradeoff is the possibility of excess resource utilization if there are no receivers[4].

5.3 Simulation Results

5.3.1 Cyclic Multicast vs. Reliable Unicast for Popular Pages

Figure 6 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:

$$Max\_Request\_Rate = \frac{Server\_Streams}{Service\_Time} \quad (6)$$

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

For the simulation parameters used in Figure 6 the maximum request rate is computed to be 4.8 requests per second. A request rate any greater than this would cause the unicast server to be unstable, i.e., it would receive requests faster than it could service them. The average time is almost 7 cycles and the maximum is more than 20. For the multicast system, the average number of cycles required to satisfy all receivers is 1.96 and no receiver experiences a wait time greater than 5 cycles.

Figure 7 shows the performance of the same type of systems and parameters as the systems shown in Figure 6, except that the request arrival rate is increased to 20 requests per second. The cyclic multicast system has nearly identical performance. Because there are still only five pages and five streams, additional requesters do not require any additional resources. Furthermore, the only limitation on the scalability of such a system is the processing of connection requests. If no

[4] Multicast routing algorithms in the Internet will prune the multicast tree to zero links but resources in the server are still allocated to the transmission of the WWW page.

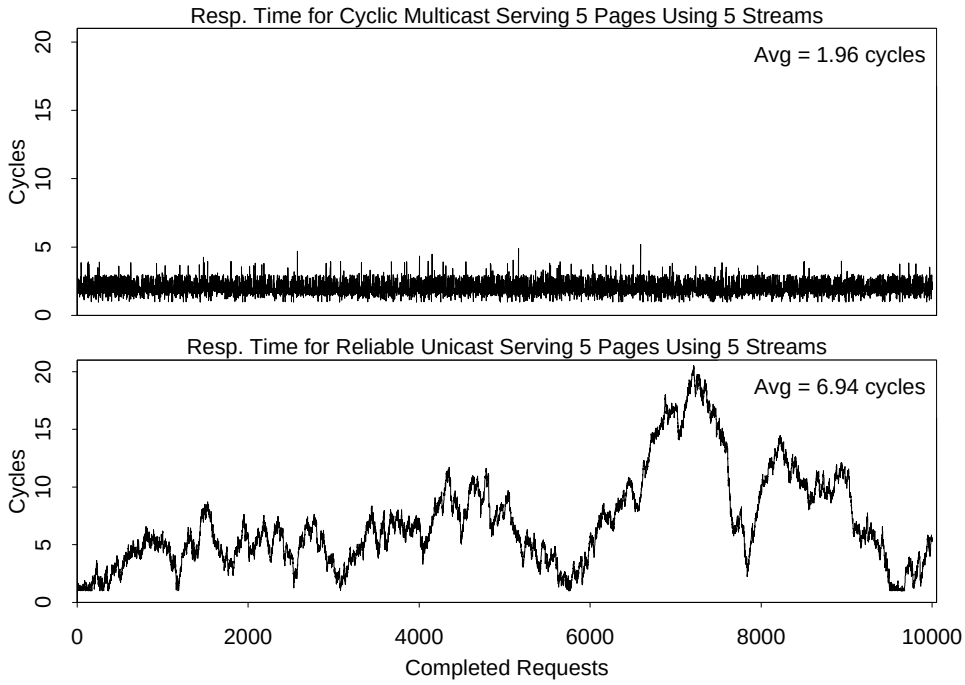

Figure 6: The effect of changing the certainty threshold.

such mechanism is used and receivers simply join a multicast group without contacting the server, the server could support any reasonably large number of receivers. However, reliable unicast does not scale at all. The second and third graphs in Figure 7 show that even three and four times as much capacity in terms of more streams is insufficient to achieve stability. Only when 25 streams are available can the unicast server meet demand.

5.3.2 Effect of the Certainty Threshold

Figure 8 shows the affect of varying the certainty threshold on the same cyclic multicast system used for Figure 6 with the request arrival rate being varied. The graph on the left shows the percentage of receivers that are unsatisfied because the cyclic multicast server had estimated there were no more receivers and stopped transmitting the respective page. The graph shows that when the certainty threshold is set to 99.99% there were no unsatisfied receivers observed during the simulation, but the tradeoff is that the server is unlikely to ever stop transmitting the WWW page, especially at high request rates. The graph on the right in Figure 8 shows the average number of streams used to satisfy receivers of the five pages being cyclicly multicast. A value of 5.0 means that all streams for all five pages were being used constantly throughout the simulation. Less than 5.0 means the server stopped transmitting for some period of time. The results show that only

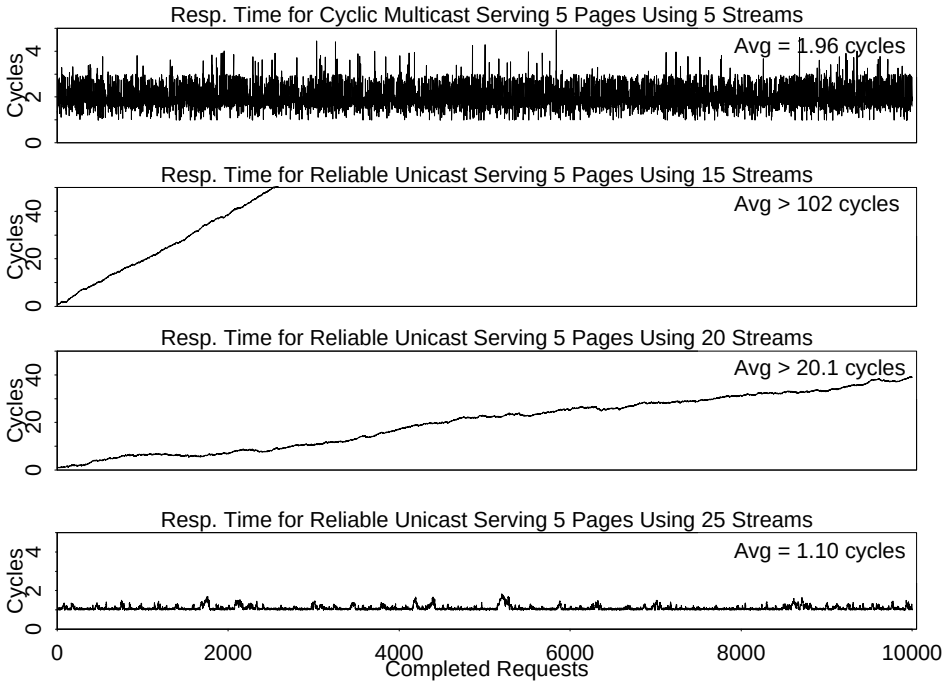

Figure 7: Comparison of cyclic multicast and different sized unicast systems.

for low request arrival rates and low certainty threshold does the server not use all the channels. Furthermore, the results show that increasing the certainty threshold does not significantly increase the amount of resources used, especially at high request arrival rates. If request rates are sufficiently high that there is almost always a new request in each cycle which means that the server never stops transmitting.

5.3.3 Performance of Integrated Servers

Figure 9 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 9 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,

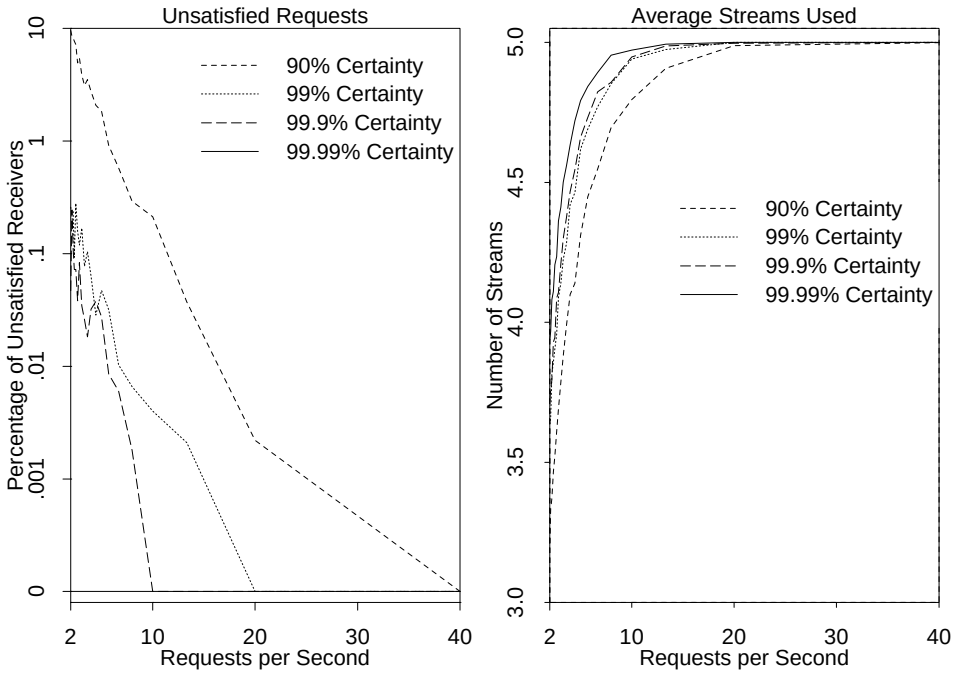

Figure 8: Comparison of estimator certainty and server workload.

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 10 shows how the performance improves by adding the ability to serve requests using reliable multicast. All but one of the parameters are the same as in Figure 9. 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. Note how the response time for pages 6-1000 in the second graph from the top improves with the response time maximum now hovering around 3 cycles compared to 6 cycles in the system without reliable multicast (shown in Figure 9). The third graph from the top in Figure 10 shows 5 streams are used for cyclic multicast, a few streams (around 5) used for reliable multicast, and the rest are used to deliver pages via unicast. The bottom graph shows that the best scalability is achieved for cyclic multicast which services the most popular page requests. Reliable multicast streams have an average group size of 2.3.

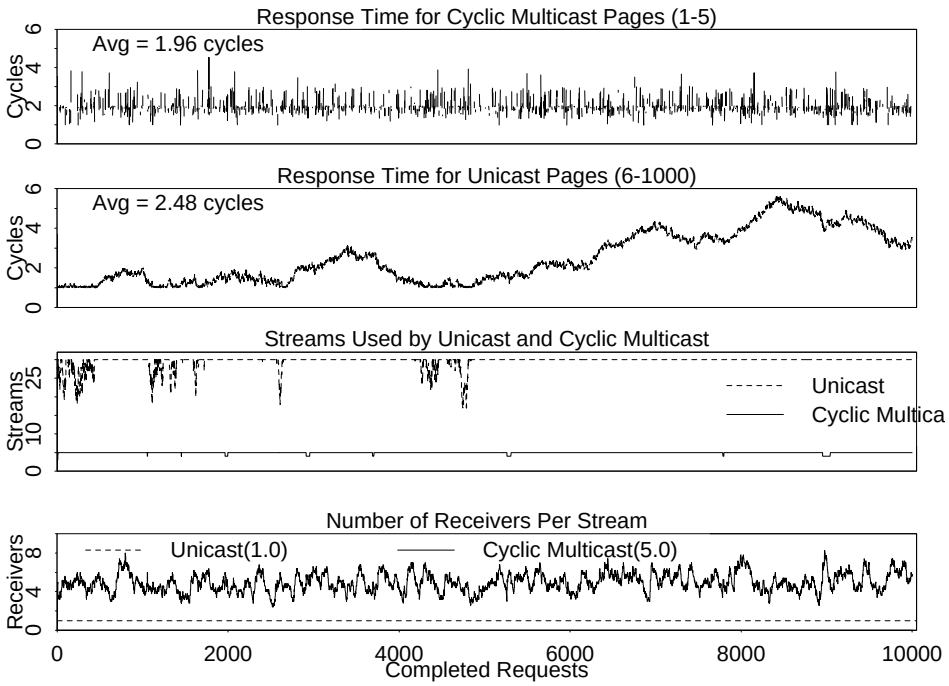

Figure 9: Server performance using unicast and cyclic multicast.

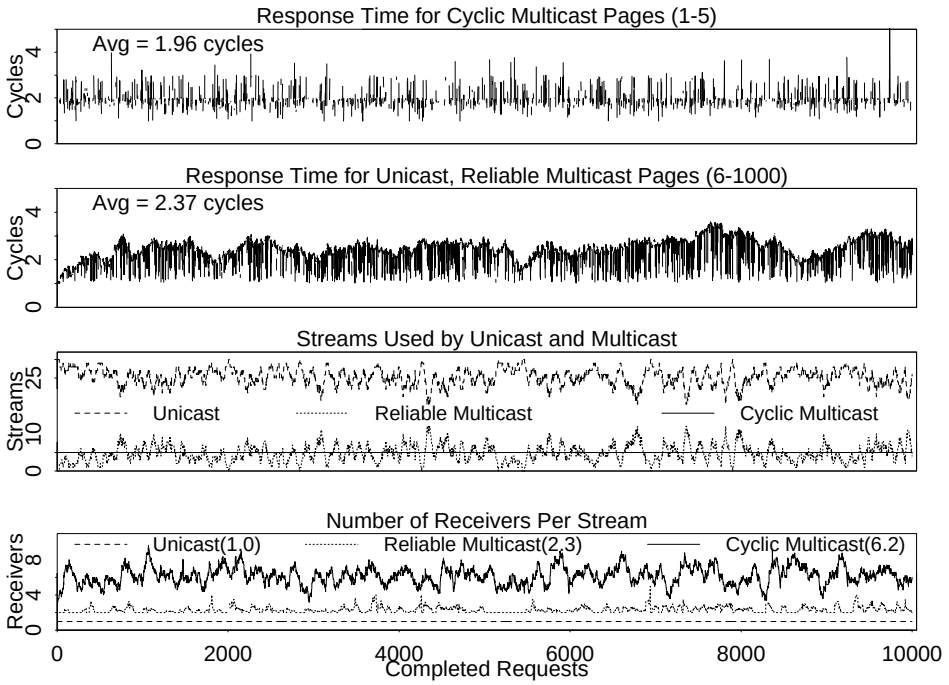

Figure 10: Performance using unicast, reliable multicast, and cyclic multicast.

5.3.4 How Many Cyclic Multicast Pages?

We now look at the issue of deciding how many pages to deliver via cyclic multicast. In Figure 11 we show simulation results where we vary the total number of pages in the WWW server library, and the number of pages delivered using cyclic multicast. Results are also given for a system that includes only cyclic multicast and reliable unicast (left column) and then all three delivery options (right column). The graphs plot the average response time as a function of the number of pages designated for cyclic multicast. The systems have a limit of 35 outgoing 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%.

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.

Note that as the total number of pages available in the server increases (going top to bottom in the figure), the optimum number of pages that should be designated for cyclic multicast decreases This is of course very dependent on the page request distribution and the results will vary if the distribution is less or more skewed than the Zipf distribution we are using.

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.

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.

In the experiments reported here we measure two components of user-perceived response time. The first is the *reception time* defined as the time between when the receiver receives the first chunk of a page until it successfully receives all the chunks that make up the page. The second

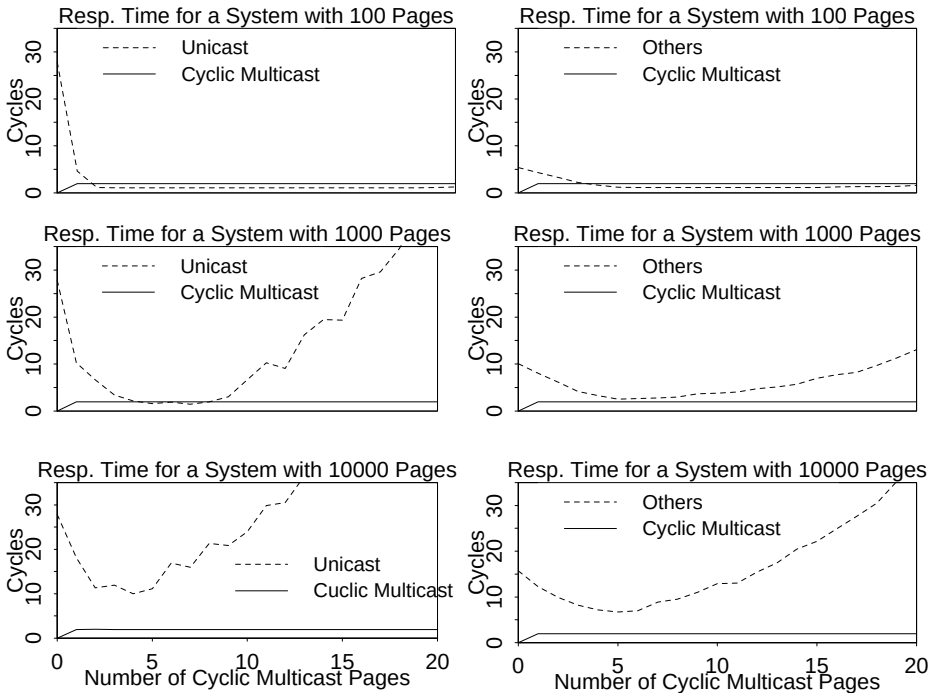

Figure 11: Varying multicast pages in cyclic multicast and unicast system.

component is the *first-packet time* defined as the time from when the user issues a request for a page until it receives the first chunk from the server. Note that our previously defined response time is the sum of the two measures we define above. 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 with the unicast web server were conducted using the Apache 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 long.

Figure 12 shows the reception time at the three clients for different cyclic multicast server transmission rates. The reception time for cyclic multicast 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 around 512 kbps and above, the transmission rate increases to the point where congestion starts to negatively affect reception time.

As a performance benchmark, Figure 12 also includes the reception time experienced when the same page is delivered via reliable unicast (actually HTTP over TCP) for a loaded web server

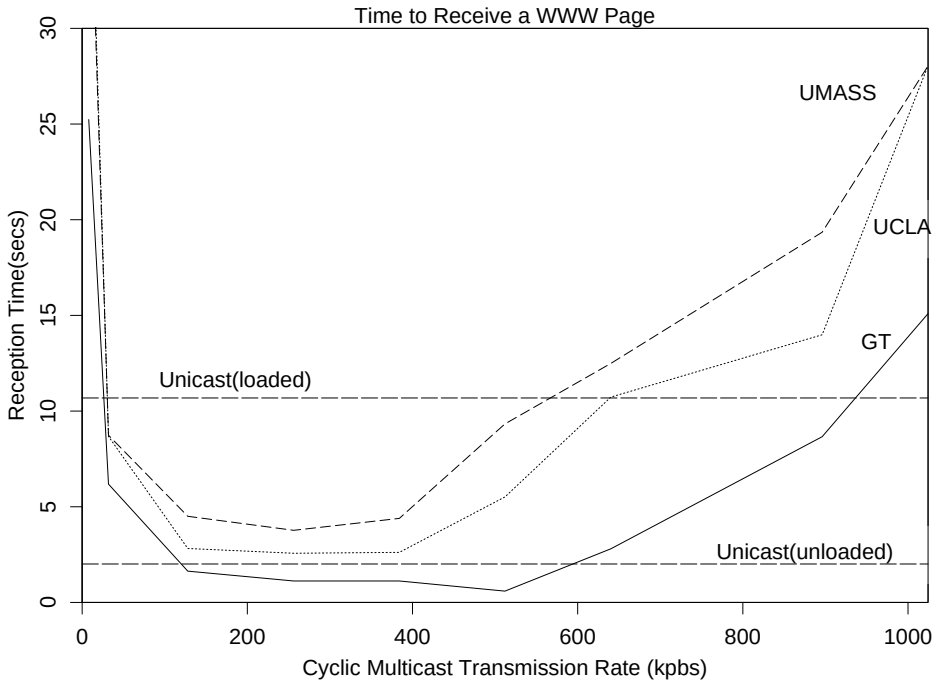

Figure 12: Time to receive page for cyclic multicast and reliable unicast.

and an unloaded web server. Both the client and the server were at Georgia Tech in the unicast experiments which tends to favor the unicast server. The loaded server was sent back-to-back requests for the same page under consideration. It should be noted that the cyclic multicast server reception times will be unaffected by loading the server with requests for the same page that is being multicast. The transmission rate was not directly controlled in the unicast experiments (of course TCP congestion control was in effect for those transmissions).

In the second set of experiments we measure the *first-packet time* defined above. For the second set of experiments 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 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

	Request to First Packet	
	Average	Std. Deviation
TCP (reliable unicast)	264 msecs	135 msecs
Cyclic Multicast – one new receiver	241 msecs	57 msecs
Cyclic Multicast – receiver at UCLA	149 msecs	68 msecs

Table 2: Time from request to reception of first packet.

above), and (2) the time for the receiver to join the multicast group. Because (1) is measured above we focused on measuring (2) The two cyclic multicast entries in Table 2 assume a particular page is already being transmitted by the server and only measures the time to join the multicast group. The first entry measures the case where there is no other receiver (at UCLA) and the multicast tree must be constructed between UCLA and GT. The second multicast entry in the table considers the case where there is already a group member on the same subnet at UCLA. We conclude here that the multicast join time will tend to be, in the worst case, on the same order as the TCP (unicast) join time.

Our experiments with the prototype, therefore, confirm that cyclic multicast of a page can drastically improve response time (reception time + join time) when a server is loaded with requests for the page.

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.

Wide scale deployment of such servers requires wide scale availability of multicast support within the Internet. Current industrial efforts including the IP Multicast Initiative are likely to make this a reality in the near future.

Future work in this area would include a more detailed simulation with more realistic trace-driven workloads and a more extensive experimentation with our prototype.

References

[1] T. Berners-Lee, R. Fielding, and H. Nielson, "Hypertext transfer protocol – HTTP/1.0," Tech. Rep. RFC 1945, IETF RFC, May 1996.

[2] 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.

[3] 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.

[4] 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. (to appear in *Computer Networks and ISDN Systems*).

[5] R. Talpade and M. Ammar, "Single connectin emulation: An architecture for providing a reliable multicast transport service," in *Proceedings of the Int. Conference on Distributed Comuting Systems*, IEEE, 1995.

[6] S. Deering and D. Cheriton, "Multicast routing in datagram internetworks and extended LANs," *ACM Transactions on Computer Systems*, pp. 85–111, May 1990.

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

[8] 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.

[9] M. Ammar and J. Wong, "Design of teletext broadcast cycles," *Performance Evaluation*, vol. 5, pp. 235–242, November 1985.

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

[11] S. Acharya, M. Franklin, and S. Zdonik, "Dissemination-based data delivery using broadcast disks," *IEEE Personal Communications*, vol. 2, pp. 50–60, December 1995.

[12] J. Metzner, "An improved broadcast retransmission protocol," *IEEE Transactions on Communications*, vol. 32, pp. 679–683, June 1984.

[13] 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.

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

[15] W. Fenner, "Internet group management protocol, version 2," Tech. Rep. draft-ietf-idmr-igmp-v2-*.txt, Internet Engineering Task Force (IETF), January 1997.

[16] M. Ammar, "Probabilistic multicast: Generalizing the multicast paradigm to improve scalability," in *Proceedings of Infocom '94*, (Toronto, Canada), pp. 848–855, June 1994.

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

[18] H. Schulzrinne, S. Casner, R. Frederick, and J. V., "Rtp: A transport protocol for real-time applications," Tech. Rep. RFC 1889, Internet Engineering Task Force, January 1996.

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