Providing Scalable Web Service Using Multicast Delivery*

Russell J. Clark
Mostafa H. Ammar

GIT-CC-95-03

January 14, 1995

Abstract

The recent growth in use of the World-Wide Web in the Internet has caused a significant increase in the demand placed on Web servers. This increased load results in noticeably longer response times for users. We propose an approach to using multicast in the delivery of Web resources that reduces the load on servers as well as the networks that connect them. We analyze the issues involved in using multicast in the Web, especially those related to routing and addressing. We also discuss the design and implementation of a system based on the existing WWW client and server architecture and the multicast support provided within IP.

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
rjc@cc.gatech.edu

*This research is supported by a grant from the National Science Foundation (NCR-9305115) and ARPA contract N00174-93-K-0105.

1 Introduction

The number of World-Wide Web users in the Internet has grown dramatically in recent months. During the period from September to October, 1994, the amount of WWW traffic on the NSFNET grew by thirty-one percent [18]. With this increased popularity has come a significant increase in the demand placed on popular Web servers. This increased demand has caused noticeable increases in response time experienced by users.

There is significant ongoing research related to the performance of WWW systems. The most popular approach to addressing the server load problem is to introduce replicated caching servers at the most popular Web sites. Claffy and Braun have analyzed the benefits of caching in the Mosaic server at NCSA [8]. Padmanabhan and Mogul have proposed mechanisms to reduce the latency associated with HTTP communications by removing the overhead of creating a new connection for every HTTP request [20]. Sedayao studied the Mosaic related traffic on a large corporate network and presented several issues to consider when deploying Mosaic [23]. Pitkow and Recker have developed a dynamic caching algorithm for Web servers that is based on a model of human memory [22].

In this paper we consider a technique that utilizes server-to-clients multicast communication to reduce the load placed on the server. At typical servers, requests for the most frequently accessed (or "hot") pages represent a large portion of requests received (see section 3). In our proposed scheme requests arriving for a particular hot page are grouped and the response is transmitted using a multicast connection from the server to the clients requesting the page. At heavily loaded servers, it is expected that within any given short period of time, many requests for the same hot page are likely to be made. The server's multicast response, therefore, has the potential of allowing the server to satisfy many requests with a single transmission; and consequently reducing the server load. Multicasting can also potentially reduce the bandwidth required to transmit the response traffic. This paper considers the design and implementation of such a multicast response system with particular emphasis on the possibility of implementing such a system within the context of the Internet and the existing World Wide Web architecture.

Our work is inspired by earlier research into the performance of Broadcast Delivery Information Systems (BDIS) [2] and broadcast delivery in Videotex and Teletext systems [1, 3, 25]. Measurement of our BDIS implementation showed significant improvement in the number of clients that a server can support for the same level of quality of service.

Another system that uses multicast to distribute resources in the Internet is the Muse mechanism for delivering USENET News over the MBONE [17]. This system uses multicast protocols over the Internet to provide the transfer of articles between News servers. The system is incorporated into the current communica-

tion environment so that when News articles are missed by one server during the multicast transmission the traditional point-to-point mechanism is used to recover the information. Other proposals for the use of multicast delivery of information include the Boston Community Information System used to deliver articles from the New York Times [14] and the DataCycle architecture [15] used to provide general database access.

The provision of multicast support within the Internet has been largely driven by the requirements of video conferencing and distribution applications. Our multicast delivery Web application provides a substantially different context which is representative of a class of multicast applications. One contribution of this paper is a first step in the evaluation of existing Internet multicast support in the context of this new class of applications.

In the next section we give a brief description of the current World-Wide Web architecture and discuss the performance problem we are addressing. In Section 3 we describe the access patterns that we observed on the Web servers at Georgia Tech. The skewed characteristics of this access pattern are key to making our proposed multicast approach beneficial. In Section 4 we present our proposed architecture for delivering Web resources to clients using multicast communication. In Section 5 we present a detailed discussion of the various design issues that need to be considered for our system. In Section 6 we discuss the status of our implementation and Section 7 contains some concluding remarks.

2 Current Web Architecture

The World-Wide Web consists of a collection of servers connected to the Internet that provide a wide variety of data resources. Users access the Web through client programs such as NCSA's Mosaic, NetScape, and the LineMode browser from CERN. Figure 1 shows a typical configuration where clients can access servers located on different physical networks. In the Web, typical requests include the access or update of files on a server host or the execution of a program. By far the most predominant access request is for the read-only retrieval of Web pages.

Web resources are identified by their Universal Resource Locators (URLs) [6] which identify a server and a path to the file, or page, representing the resource being requested. Web clients typically support several different protocols or access methods for communicating with servers. The most common access method for current web resources is the HyperText Transfer Protocol (HTTP) [5]. In our work we only consider resources accessed using HTTP.

HTTP is a connection-oriented transaction protocol for accessing and manipulating documents in hypertext as well as other formats. HTTP is designed to utilize a reliable, connection-oriented transport protocol (TCP) for the transfer of infor-

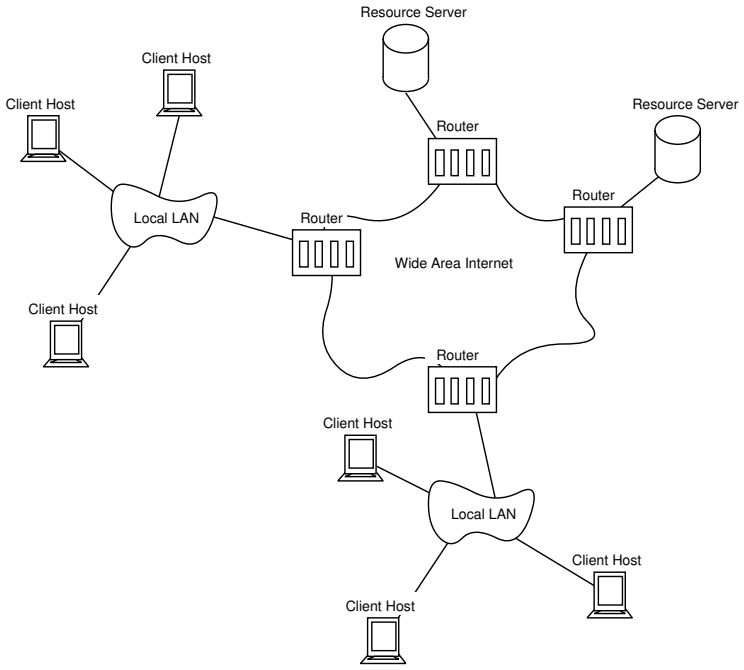

Figure 1: WEB clients in the Internet

mation between Web systems. Each HTTP connection is built on top of a single TCP connection. Both the page request and the response are carried over the same HTTP/TCP connection. This communication is inherently point-to-point and each request by a client requires that a new reliable connection is setup to the server. Given this architecture, servers are designed to handle every user request individually.

3 Server Request Statistics

The effectiveness of the multicast response approach we propose is dependent on the skewed access characteristics of WWW servers. It has been observed that the bulk of accesses to Web servers are requests to retrieve one of a small percentage of the files on each server. These files comprise the set of so called hot pages for the server.

Figure 2 demonstrates this kind of behavior for the server *www.gatech.edu* at the Georgia Institute of Technology. These statistics are taken from the access activity from September 11, 1994 through January 5, 1995. For this 118 day time period there were 2,519,609 accesses to the server using 51,580 distinct requests. The x-axis indicates the number of distinct requests, listed in increasing order of popularity. The y-axis indicates the total number of accesses accounted for by the requests. Of the 51,580 unique requests to the server, the first 43,842 (85%) account for just

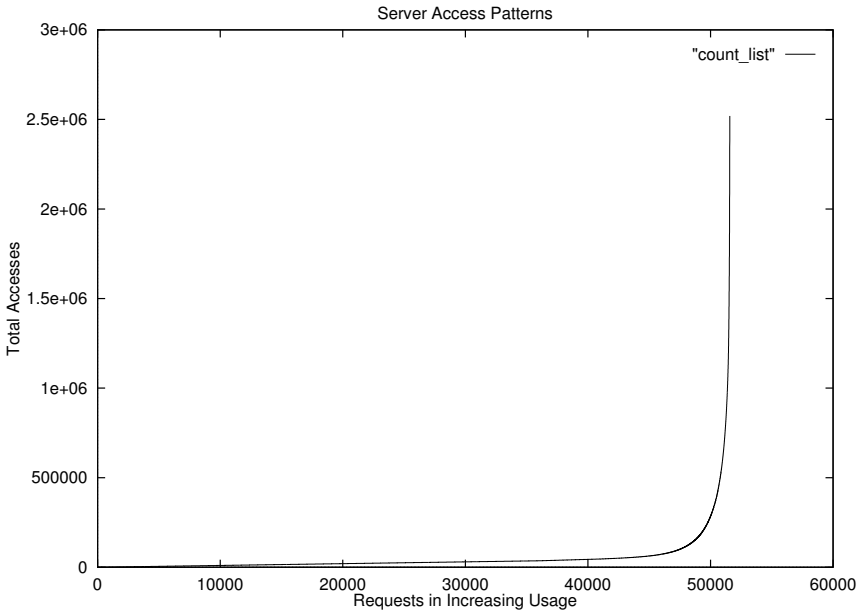

Figure 2: Access Patterns for *www.gatech.edu*

Pages		Accesses	
Number	Percentage	Number	Percentage
26	0.05	691532	27.45
52	0.10	974901	38.69
258	0.50	1623548	64.44
516	1.00	1872243	74.31
2580	5.0	2350769	93.30
5159	10.0	2441494	96.90
7738	15.0	2463724	97.78
10317	20.0	2472943	98.15

Table 1: Request Percentages for *www.gatech.edu*

55,885 (2.2%) of the accesses. The first 49,000 (95%) account for 168,840 (6.7%). It is the remaining 2580 (5%) most popular requests that account for 93.3% of the server accesses.

Some data values for the most popular pages are given in Table 1. The first line indicates that just the top 26 requests alone account for 27.45% of the server accesses over this period. The top 0.50% of the total requests account for nearly two-thirds of the total accesses. We have observed similar skewed request patterns on two other popular servers located on the Georgia Tech network. Similarly, Claffy and Braun describe results showing that the top 25 documents (.27%) on the NCSA Mosaic server account for 59% of the document requests [8].

A closer inspection of the top 100 requests for this server reveals that all but four

were simple *HTTP GET* requests to retrieve a file[1]. This means that for 96 of the top 100 requests, the server loaded and transmitted the same exact file for every access issuing one of these requests. On popular servers it is likely that there will be multiple clients concurrently submitting the same request. It is this duplication of effort by Web servers that we alleviate in our multicast approach.

Period	Mean	Max	Var	Sdev
Overall	14.95	171.00	313.13	17.70
Weekdays	17.78	171.00	368.04	19.18
Weekdays 10am-5pm	29.14	170.00	541.09	23.26
Weekdays 4pm-5pm	32.85	153.00	608.23	24.66
For just November (the busiest month)				
Weekdays 10am-5pm	33.88	134.00	402.25	20.06
Weekdays 4pm-5pm	38.79	125.00	403.84	20.10

Table 2: Requests Per Minute for *www.gatech.edu*

It is important to emphasize that we use the Georgia Tech server as an example for the request distribution only. We do not consider this server to be heavily loaded. The request rates for this server are presented in Table 2. With an average of 14.95 requests/minute over the entire sample period, it is unlikely that many requests for the same page will arrive within a short period (say 10 seconds) and the benefits of grouping requests and multicasting cannot be realized. Other WWW servers (e.g. www.ncsa.uiuc.edu, info.cern.ch) are much more heavily loaded and multicast delivery can be used to advantage there. If, however, the activity on the Georgia Tech server were to grow by an order of magnitude (something that can easily occur within the next year), then multicast response can be beneficial. Even at its current level of activity, the Georgia Tech server experiences heavy usage times where the peak access rate can easily be an order of magnitude (171 requests/minute) above the average rate. Multicast response can be beneficial at those times as well.

4 Proposed Architecture

In this section we present an overview of our proposed multicast delivery architecture and summarize the behavior of clients and servers in the architecture. A detailed discussion of the design issues is presented in Section 5.

[1]The other four were *cgi-bin* requests that return the output of a program run at the time of the request. None of the top 100 requests were *PUT* requests or any other type that submitted data to the server as opposed to retrieving files from the server.

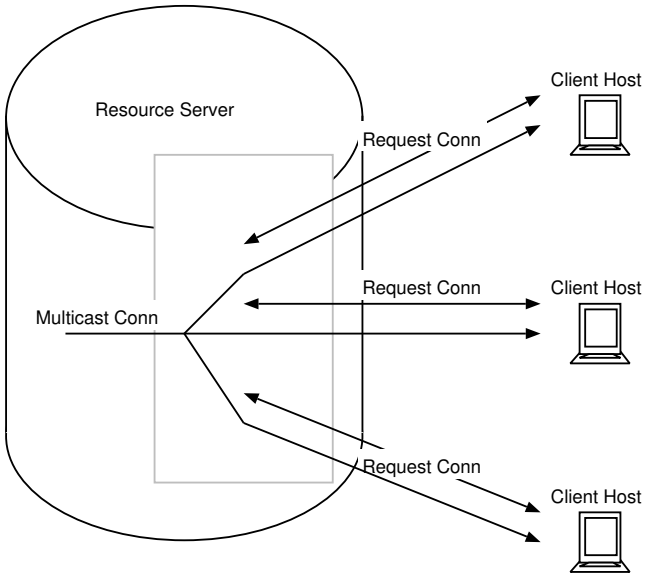

Figure 3: Multicast Communication Architecture

4.1 Request and Response Connections

In our proposed architecture requests reach the server in the same way as the existing WWW system, i.e., via an HTTP connection. Pages are classified by the server as either "hot" or "cold". Responses to requests for cold pages are handled in the same manner as before, i.e., the response is sent back using the same HTTP connection that carried the request.

For hot pages, clients who requested the same hot page are grouped (in a manner described later) into a single multicast group and a single multicast response connection is initiated from the server to this multicast group. The response is then transmitted over the connection. This architecture is depicted in Figure 3.

In the same way that an HTTP connection is built on top of a TCP connection, the multicast response connection needs to be built on top of a reliable multicast transport protocol. For this we use a protocol that we developed in our group [24]. This protocol is, in turn, built around the IP multicast capabilities supported in the Internet Multicast Backbone (MBONE) [7]. In such an environment, the multicast group formed by the clients requesting the same page needs to be identified by a single multicast IP address and all the clients need to explicitly join that group. We discuss how this is accomplished shortly.

When designing new features for an application used as widely as the Web, it is important to maintain compatibility with current systems. For the Web, this means that current clients must be allowed to communicate with new multicast capable servers and new multicast capable clients must operate with current servers. We have designed our multicast extensions to the Web service with the goal of sup-

porting the coexistence of different Web systems. For example, we allow clients to declare (in their request message) whether they are multicast capable. If not, their responses are sent via the request HTTP connection. To communicate with non-multicast capable servers, clients (even multicast capable ones) must allow for the reception of responses on the HTTP connection used to send the request, as well as a multicast response connection.

4.2 Client Request Processing and Response Reception

The multicast capable client performs the following steps when requesting a page from a Web server.

1. *Determine group for request:* The client determines the multicast group address a server would use to send this page if it were sent via multicast. At this point, the client does not know for certain that the requested page will be sent via multicast since it does not know whether the page is a hot page. Also the client could be communicating with a non-multicast capable server.

2. *Join group:* The client begins listening to the previously determined group. In a BSD socket implementation, this is done by creating a socket for the multicast address and doing a *listen()* on that socket. This step should also involve notifying the local router of the requested group membership. For example, the client could send an IGMP Host-Report message to the router [10].

3. *Send query to server:* The client sends the request to the server. This is done using the current connection request model of HTTP. Along with the request message the client sends an indication that it will support a multicast response for this query.

4. *Listen for response:* The response could come as a unicast response over the request connection or multicast on the new connection. If no response arrives within an expected timeout the client should resubmit the request with an indication that multicast should not be used. This will cause the server to reply to this second request over the unicast channel. This special retry procedure, which we expect to be a rare occurrence, is needed to overcome problems, as discussed later, that may be encountered in multicasting a response.

5. *Close Connections and Leave Group:* Once the response is received the client can close both the request and multicast connections. The client also stops listening to the multicast address.

4.3 Server Request Processing and Response Transmission

The multicast capable server performs the following steps when processing a request.

1. *Receive request:* The server listens for connection requests and receives the HTTP request once a connection is established.
2. *Schedule request:* The server determines when to respond to the request. The scheduling of response transmissions is discussed later.
3. *Transmit request:* If this is a request for a hot page then a new connection is established with the multicast address used to satisfy this request. The page is then sent using this new connection. If the request is not a GET for a hot page then the request is satisfied via the current HTTP approach of unicasting the page to the client over the request connection.

5 Design Issues

Here we present a detailed discussion of several issues in the design of a WWW multicast delivery system. We specifically consider the issues brought up in the overview discussion in Section 4.

5.1 Multicast Addressing

In our proposed multicast response mechanism each response is directed to a specific multicast IP address. Clients expecting the response need to add the multicast IP address to the list of addresses they recognize in order to receive the multicast response. In addition to loading a multicast group address, clients also need to listen to the proper port number in order to receive the multicast connection requests.

The problem here is how to get all the clients requesting the same page and the server to agree on the same multicast address. For this we use an *address mapping* function that maps a page's URL into a multicast address. All multicast clients and servers use the same function and agreement on the same address is, therefore, possible. The selection of an appropriate function is important and we discuss this below.

The ideal address mapping function has the following properties:

1. it provides a one-to-one mapping of URL's to multicast addresses.

2. the domain of the mapping function forms a reserved set of multicast addresses not to be used by other applications.

3. it is a simple mapping procedure that is not time-consuming.

This ideal scheme will insure that each URL is mapped into a unique multicast address. Therefore, multicast communication with clients requesting this URL cannot be confused with other URLs or multicast applications and vice versa.

The current multicast IP environment allows for a total of 2^{28} multicast addresses and is, therefore, not favorable to the ideal mapping scheme as described above. It is easy to envision a time when the number of WWW pages exceeds 2^{28}. Also mapping each URL (essentially a string of ASCII characters) into a 28 bit stream in a one-to-one fashion will require complex procedures that can be time-consuming[2].

We, therefore, adopt a mapping function that provides a many-to-one mapping, thus relaxing our first ideal property. We design this mapping such that URLs on different servers are mapped to distinct multicast addresses, but URLs on the same server may be mapped into the same address. This is accomplished by using a hierarchical addressing scheme with part of the address uniquely defining a server (the first part of the URL, e.g. www.gatech.edu) and the rest of the address describing the remainder of the URL. The effect of this many-to-one mapping is that clients may get pages they did not request in which case they can easily discard such responses. It is important to emphasize that a server that gets requests for two different URLs that happen to map to the same IP address will still be able to determine that these are two distinct requests.

5.1.1 Proposed Addressing Scheme

In our current implementation we are using a hierarchical addressing scheme for assigning multicast addresses to WWW pages from the current IP class D address space. This scheme is shown in Figure 4. The first four bits indicate that this is a class D multicast address. The next twelve bits are used to indicate that this address is from the range of addresses reserved for WWW use. Each value in this field reserves 2^{16} addresses for WWW use.

The *Server Address* field is a number from 0-255 that is assigned based on the server that provides the page associated with this address. Each popular server will be assigned a number to use in this field. This number will be known by the server and should be available to all clients that request files from the server. One way to deliver this address to the clients is to include it in the DNS [19] information for the server. When a client sends a request it will generally query the DNS to obtain

[2]The next generation IP protocol (IPv6) [16] may have a larger multicast address space and may allow greater flexibility in multicast address assignment.

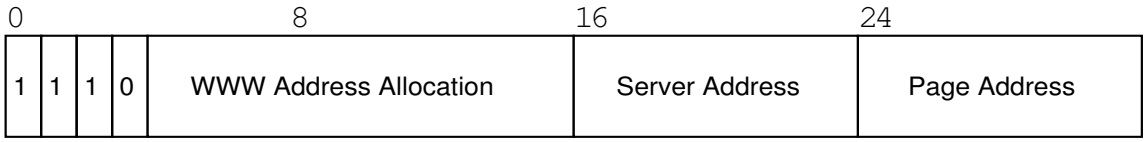

Figure 4: WWW Multicast Address Scheme

an IP address for the server. At the same time it could obtain the multicast address assigned to that server so it would know which address to listen on for the reply.

The *Page Address* field is a number from 0-255 that is assigned based on the pathname of the page being accessed. This pathname is equivalent to the *url-path* portion of the Uniform Resource Locator (URL) used to access this page [6]. This character string is hashed into an eight bit number to generate the address. In our current implementation we use a simple additive hash function that calculates the sum of the ASCII values modulo eight.

This scheme reserves 256 addresses to each of 256 different servers. Given the relatively small number of high demand servers that currently require multicast capability, we expect that an assignment of of one or two blocks of this size would be sufficient for near-term use. Based on our analysis of server access distributions we feel that 256 addresses should be sufficient for covering the hot pages at a server.

5.2 Request Scheduling at the Server

The current popular HTTP server implementations from NCSA and CERN are designed to handle requests using a single request per connection. When a connection request arrives, a new process is created via *fork()* to handle the server request associated with this single connection. In this design there is no mechanism for a shared queue of page requests. The server performs queueing of connection attempts, not resource requests.

In order to benefit from multicast shared response in a resource server it is necessary to use a central *request* queue. This queue provides a rendezvous point for matching common requests. Our proposed server implementation, presented in Figure 5, includes such a request queue.

As requests arrive they are queued in a FCFS order [3]. An arriving request is checked to see if it is for a hot page (a page that will be responded to by multicast). If so, the current request queue is checked to see if there are other pending requests for the same page, in which case the newly arriving request is dropped. Otherwise, the newly arriving request is queued for service. Requests for cold pages are

[3]Other queuing disciplines are possible and may even be desirable. An analysis of some possibilities is presented in [13]. We plan to consider these scheduling disciplines in future work.

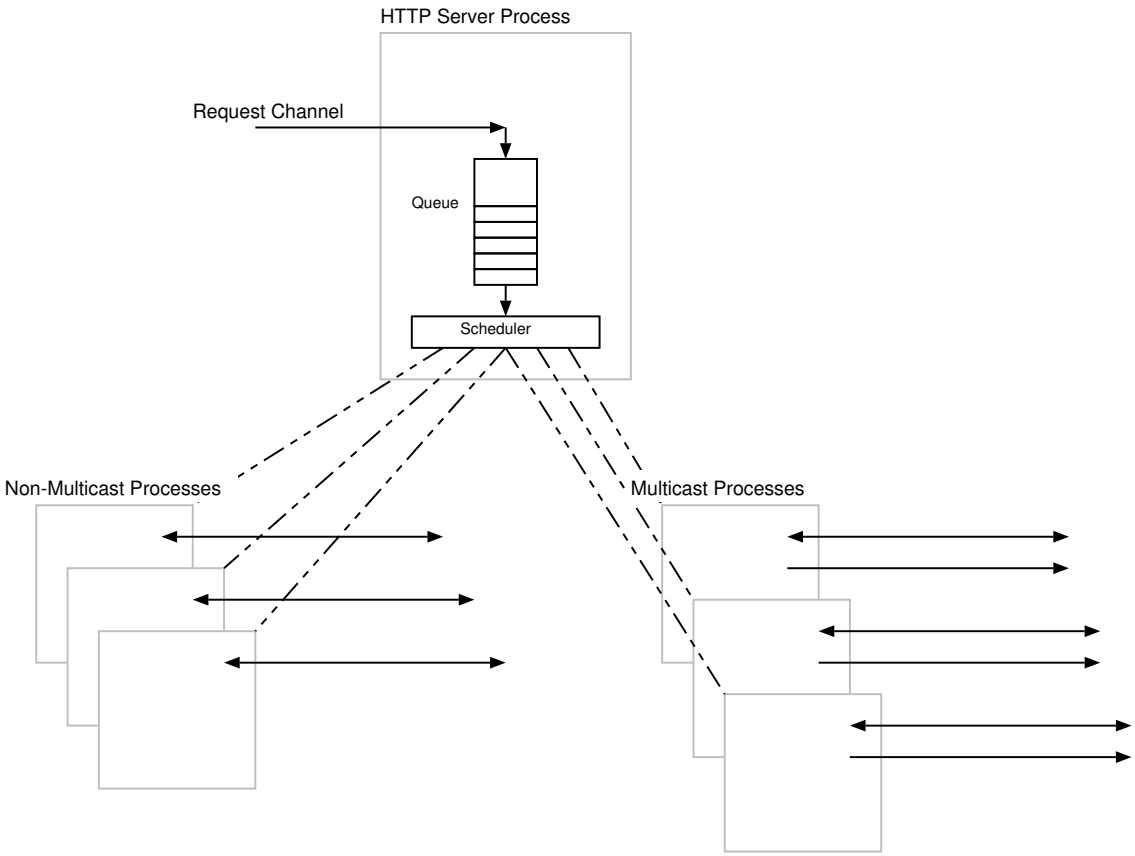

Figure 5: Server Implementation Architecture

automatically placed at the end of the queue even if they duplicate existing requests. Note that each request in the queue for a hot page may represent multiple clients waiting for a response. Our implementation maintains a list of the clients associated with each hot request. This queueing discipline has been analyzed in [3].

When a request arrives at the front of the queue and is ready for service the server checks to see if this request is eligible for a multicast response. If the response will not be made via multicast a regular service process is created to handle this request in the same manner as traditional implementations. If a response could be made via multicast, a multicast service process is created to handle the request. A page that is eligible for multicast but with only one requesting client is handled by the regular *point-to-point* service process.

The multicast service process creates a multicast connection with all of the clients that requested the resource using the multicast address derived from the URL name. This new connection is established by creating a new socket and then calling the *connect()* system call. The reliable multicast protocol is our working prototype of the Single Connection Emulation (SCE) architecture described in [24].

5.3 Interaction with Multicast Routing

Bandwidth Consumption: By multicasting responses we are able to satisfy multiple client requests with a single transmission from the server. This clearly saves on server resources and allows it to handle a heavier request load. Multicasting can also potentially save on the network bandwidth required to satisfy the clients requests. Whether bandwidth savings are possible is very much dependent on the characteristics of the multicast routing protocols being used.

In order to minimize bandwidth cost it is important that the multicast traffic is only delivered to networks that have interested clients. A multicast routing protocol proposed for the Internet and in use in the MBONE is a variation of reverse path forwarding [9] known as the Distance Vector Multicast Routing Protocol (DVMRP) [12, 21]. In its simplest version this protocol would broadcast all multicast messages and rely on end routers to discard packets if they know of no end-users that should receive them. Clearly, this form of multicast is wasteful of bandwidth. Unless there are interested clients on every network, multicast responses carried over this form of routing scheme will consume more network bandwidth than a sequence of individual point-to-point responses taking shortest paths from the server to each individual client. A variation of the DVMRP scheme allows for more efficient routing by pruning the parts of the network that should not receive packets destined to particular multicast addresses. Other proposed multicast routing protocols for the Internet include Core Based Trees [4] and Protocol Independent Multicasting (PIM) [11]. These protocols promise more bandwidth-efficient multicast routing.

Time to Join: Another issue to consider with multicast routing protocols is the time lag between the point where a host requests to join a group and the point where it is able to receive messages sent to the group. Clients join multicast groups by issuing requests to network routers. Propagation of this information across the network eventually results in the particular client's host machine receiving multicast messages destined to the group. How this information propagates and how long it takes is a function of the particular routing protocol in use. With DVMRP and its variants clients will not experience significant delays in joining multicast groups.

PIM operates in two modes. In its *dense mode* it behaves similarly to DVMRP. The *sparse-mode* PIM operation is a receiver-initiated protocol. In this approach, multicast traffic is not forwarded onto a network until some receiving station explicitly requests transmission of the traffic. A receiver makes this request by contacting a rendezvous point designated for the group it wishes to join. With this approach it can take longer for a client to begin receiving traffic for a group because of the additional delay in contacting the rendezvous point. This delay could be critical for a multicast based Web server since the client must be able to receive

group messages quickly in order to realize an improvement in the response time over a point-to-point response. More analysis of PIM characteristics is warranted in order to investigate this aspect of its performance and how it interacts with our multicast response scheme.

Recall (Section 4.2, Step 4) that multicast capable clients that do not receive a response on the first try, retransmit their request with a point-to-point retransmission requirement. This will take care of situations when a client does not receive a multicast response because it did not join the multicast group in time.

Multicast Scope Control: With current MBONE multicast routing (based on DVMRP), each multicast message is scope controlled by limiting the number of hops (or routers) it can go over. The particular scope of a multicast message is determined by the distance to the farthest destination in the multicast group. Having a larger than necessary scope for a multicast message can cause significant bandwidth wastage. It is, therefore, important in our proposed scheme for the server to be able to determine the appropriate scope for each multicast response message.

One way to handle this is for the server to maintain a list of hopcounts to various destination networks in the Internet. When the server is ready to transmit a page via multicast it will then reference this list to determine the "distance" to each of the clients currently requesting the page. Another approach would be to guess the distance of a client from the remaining value of the incoming IP datagram Time-to-live field.

By setting the scope to the distance of the farthest client the server should reach all the intended clients. Because of routing fluctuations this will not always guarantee that all clients are reached. As always, clients will be able to resend a request and ask for a unicast reply in the event that multicast replies are not reaching them.

5.4 Multicast Connection Semantics

Our multicast response is based on the use of a connection-oriented multicast transport protocol that we developed [24]. For such a protocol it may not always be necessary or even appropriate for all group members to be active for the entire duration of a connection. There are situations where it makes sense to allow participants to join the connection late or to leave early. Given this situation, it becomes necessary to specify the semantics or success criteria for what is required in a multicast connection. Our prototype protocol allows the application (its user) to specify the semantics of the connection.

For our multicast HTTP server we use a reliable connection that supports the semantics we refer to as "at least n or timeout". When the server sends a page it

will try to reach all of the clients that have recently requested that page. The server will attempt to connect to *at least* those clients. However, if one or more of the known clients is no longer accessible there must be a mechanism for allowing the server to proceed with responses to the rest of the group. We propose that the server use a timer to indicate when it is acceptable to end the connection establishment phase and proceed with the transmission to the currently connected clients.

Another aspect of the multicast approach is that there may be additional clients that are in the process of requesting a resource while the server is setting up the multicast connection. In this case, these clients whose requests have not yet reached the server should also be allowed to join the multicast response connection[4]. In this scenario it is possible that more than n clients will become connected and the server should allow this. It is important to emphasize that clients can only join during the connection establishment phase. Once data transfer is started the set of connected clients is only allowed to shrink, not grow.

5.5 Determining Hot Pages

In our server architecture it is necessary to determine when a page is "hot" enough to warrant multicast delivery. The challenge is to identify situations where the use of multicast will result in an overall reduction in the cost of delivering a page to all the requesting clients. We define the cost of delivering a single page to a single client via point-to-point delivery as C_p. The cost of delivering a single page to k clients via point-to-multipoint delivery is $C_m(k)$[5]. A page i, with average number of clients per multicast X_i, should be identified as hot when $C_m(k) < X_i C_p$.

We can identify the various aspects of the system that contribute to the costs $C_m(k)$ and C_p. The cost of delivering Web pages is incurred by the server, the network , and the clients. In the server, components of both $C_m(k)$ and C_p include parsing the client's request, loading the file from disk, and transmitting the file on the network. The additional costs for C_m are grouping the common requests in the queue, determining the destination multicast group, and opening the multicast reply channel. The savings when using multicast comes from performing the common tasks which contribute to the values of both $C_m(k)$ and C_p just one time. In general, the larger a file is, the more these common tasks will cost and the more benefit will be attained from using multicast delivery.

In the network the cost of delivering a page is measured in the bandwidth consumed on each segment that carries the data. For C_p this cost is relatively constant for each segment but, again, it must be repeated every time a page is transmitted. As

[4]This has the interesting effect of allowing the client to receive a reply before it finishes submitting the request.

[5]It is reasonable to expect that the cost of a multicast response will consist of a fixed cost component and a cost component that varies with the number of clients.

discussed in Section 5.3 there may be some additional bandwidth cost in $C_m(k)$ for bandwidth used when pages are delivered to networks where there are no clients requesting them. The cost benefit for multicast in the network is the potential reduction in bandwidth used, particularly at the points closest to the popular Web servers.

At the clients the common costs are in formulating the request, transmitting the request, and receiving the reply. Additional components of the cost in $C_m(k)$ are in determining which multicast group to join, joining that group, and then listening for a response on the multicast connection. We expect that this cost will be overcome in improved response time if a client typically accesses popular servers. When the client is accessing a less popular server, that server will not have an assigned multicast address range and the client will not need to join the multicast group.

In most cases, the set of hot pages on a server may change over time. These changes should be made as access patterns alter the X_i's for the various pages and as $C_m(k)$ and C_p change based on variations to the system and network load and configuration.

Our work in progress will aim at estimating the costs $C_m(k)$ and C_p through a combination of measurement, simulation and analysis.

6 Implementation Status

We are developing a system based on version 3.0 of the HTTP server and LineMode client from CERN. We are using the LineMode client because of the ability to use it in batch mode during experimental benchmarks of the system. It is important to note that we are still processing the large size graphic and audio files as well as HTML pages typically found on a Web server.

Our implementation is being developed on the Sun OS release 3.1.U1 operating system for Sun Sparc systems. These systems have been modified to support IP multicast and the multicast transport protocol described in [24]. This transport protocol provides a reliable point-to-multipoint implementation of the TCP protocol.

We have made significant modifications to the HTTP server code to implement the request queue architecture depicted in Figure 3. In addition, we have added support for the detection of hot pages and the creation and use of multicast connections. We have added support in the client-server options negotiation for clients to indicate that they can accept multicast replies. This option negotiation is done as part of the request sent by the clients.

In the client implementation we are also adding support for joining the multicast

group and listening for replies on both the unicast and multicast channels. The client will also include the retry mechanism described in Section 4.2 for recovering from missed multicast responses.

We plan to test the functionality and performance of our system using scripted clients running over a large number of machines in a workstation cluster. We will report on the results of those experiments in a later document[6].

7 Concluding Remarks

The phenomenal growth in popularity of the World-Wide Web is placing a significant demand on Web servers and the networks that connect them. Meeting this demand will require novel approaches to the design of network distribution services. We have developed a Web page distribution architecture that addresses the problem of overloaded Web resources through a shared response mechanism. In this architecture, multicast distribution is used to deliver Web pages to multiple clients at the same time. A key to this approach is that it takes advantage of the skewed access patterns of Web servers where a small percentage of hot pages account for a major portion of the requests handled. By delivering these hot pages via multicast we can potentially reduce both server and network loading by eliminating the need to transmit the same page several times over a short period.

We are currently in the process of completing and testing an implementation of the design presented in this paper. It should, perhaps, be emphasized that while our discussion of the design issues may have seemed complex at times, our implementation incorporating the final results of this analysis will be of similar complexity/simplicity to the existing Web server and client software.

Multicast delivery of Web pages is an instance of an application requiring the use of reliable multicast. It is also an application in which the multicast source (the server) knows the identities of the multicast destinations (the clients). In those respects it is very different from the video conferencing and distribution applications that have been the driving force behind development of multicast technology in the Internet. Our analysis of the design issues relating to our application has yielded some interesting insights that are likely to carry over to other applications. These include: 1) the need for flexible multicast connection semantics, 2) the extent of interaction between multicast routing and application requirements, and 3) the importance of allowing for multicast address allocation mechanisms similar to those used for unicast addressing.

[6]We plan to make our implementation available to others. Readers who are interested in obtaining a copy should contact one of the authors.

References

[1] M.H. Ammar. Teletext-like information delivery using broadcast polling. *Computer Networks and ISDN Systems*, 12(2):107–115, 1987.

[2] M.H. Ammar and H. Kim. Prototyping a broadcast delivery information system. Technical Report GIT-ICS-90/16, Georgia Institute of Technology, May 1990.

[3] M.H. Ammar and J.W. Wong. Response time performance of videotex systems. *IEEE Journal on Selected Areas in Communications*, 4(7):1174–1180, October 1986.

[4] T. Ballardie, Paul Francis, and J. Crowcroft. Core based trees (CBT) an architecture for scalable multicast routing. In *ACM SIGCOMM '93*, pages 85–95. ACM, September 1993.

[5] T. Berners-Lee. Hypertext transfer protocol. *Internet Draft*, November 1993.

[6] T. Berners-Lee, L. Masinter, and M. McCahill. Uniform resource locators (URL). *Internet Draft*, September 1994.

[7] S. Casner. Frequently asked questions (faq) on the multicast backbone. *file://venera.isi.edu/mbone/faq.txt*, May 1993.

[8] K.C. Claffy and H. Braun. Web traffic charecterization: An assessment of the impact of caching documents from NCSA's web server. In *Second World Wide Web Conference '94*. http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings, October 1994.

[9] Y.K. Dalal and R.M. Metcalfe. Reverse path forwarding of broadcast packets. *Communications of the ACM*, 21:1040–1048, December 1978.

[10] S. Deering. Host extensions for IP multicasting. *RFC 1112*, August 1989.

[11] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei. Protocol independent multicast(PIM): Motivation and architecture. *Internet Draft*, October 1994.

[12] S.E. Deering and D.R. Cheriton. Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.

[13] H.D. Dykeman, J.W. Wong, and M.H. Ammar. Scheduling algorithms for videotex systems under broadcast delivery. In *IEEE International Conference on Communications '86*, pages 1847–1851. IEEE, June 1986.

[14] D.K. Gifford. Polychannel systems for mass digital communications. *Communications of the ACM*, 33(2):141–151, February 1990.

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

[16] R. Hinden. Internet protocol, version 6 (IPv6) specification. *Internet Draft*, October 1994.

[17] K. Lidl. Drinking from the firehose: Multicast USENET news. In *USENIX Winter '94*, pages 33–45. USENIX Association Press, January 1994.

[18] Merit. Nsfnet traffic distribution highlights. *file://NIC.MERIT.EDU/nsfnet/statistics/1994/nsf-9410.highlights*, October 1994.

[19] P. Mockapetris. Domain names - concepts and facilities. *RFC 1034*, November 1987.

[20] V.N. Padmanabhan and J.C. Mogul. Improving http latency. In *Second World Wide Web Conference '94*. http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings, October 1994.

[21] C. Partridge, D. Waitzman, and S. Deering. Distance vector multicast routing protocol. *RFC 1075*, November 1988.

[22] J.E. Pitkow and M.M. Recker. A simple yet robust caching algorithm based on dynamic access patterns. In *Second World Wide Web Conference '94*. http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings, October 1994.

[23] J. Sedayao. "Mosaic will kill my network!" - studying network traffic patterns of mosaic use. In *Second World Wide Web Conference '94*. http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings, October 1994.

[24] R. Talpade and Mostafa H. Ammar. Single connection emulation (SCE): An architecture for providing a reliable multicast transport service. Technical Report GIT-CC-94/47, College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280, October 1994.

[25] J.W. Wong and M.H. Ammar. Analysis of broadcast delivery in a videotex system. *IEEE Transactions on Computers*, 34(9):863–866, September 1985.