Jambalaya:

Using Multicast for Blind Distributed Web Searching and Advertising

Julio C. Navas*and **Haym Hirsh**
Computer Science Department
Rutgers, The State University
Piscataway, NJ 08855
{navas,hirsh}@cs.rutgers.edu

November 30, 1998

Abstract

Currently, the communication protocols for the World Wide Web are based on a form of unicasting called the Transport Control Protocol (TCP). One of the assumptions of unicasting, which is point-to-point only, is that the address of the remote end-point is known in advance. This assumption, however, severely constrains the design of web information systems by forcing them to accept certain compromises, such as being highly centralized. This paper proposes the use of multicast as an underlying technology on which more robust web applications can be based. By using multicast, web information systems gain flexibility and shed the constraints imposed by unicasting. We also describe Jambalaya, a web information system that is based on the multicast protocol and demonstrates these ideas in two ways: by allowing users to perform sender-blind broadcasting of information requests, and by allowing information servers to perform sender-blind advertising of their most-often requested information.

1 Introduction

Today's web information systems are based on the unicast point-to-point model of internetwork communications. The main limiting factor of unicasting is that the end-point of any connection must be known a priori. This places severe restrictions on the design and implementation of web information systems. Basing the web information systems on *multicast*, which can be a more powerful and general model of internetwork communications, can greatly increase the flexibility of the web as a whole because multicast allows for *sender-blind communications*. By blind, we mean that senders of information do not need to know a priori the identities, locations, or number of receivers.

*Julio C. Navas was supported in part by DARPA under contract numbers DAAH04-95-1-0596 and DAAG55-97-1-0322, NSF grant numbers CCR 95-09620, IRIS 95-09816 and Sponsors of WINLAB.

One limitation of unicasting is that it forces systems designed for finding information on the Internet to be centralized. Unicasting requires that point-to-point connections be made. Since this means that the address of the end-point must be known a priori, it is infeasible to connect directly to every information server endpoint to gather knowledge regarding its content and to obtain whatever relevant information it may have. Instead, information about the content of these servers is typically sequestered at a central repository at a well-known location, such as at a web search engine, thus creating islands of meta-information. By their very nature, however, these islands are points-of-failures and bottlenecks. Also, simply the process of gathering all of this meta-information places undue strains on the web servers and the intervening network links since the entire document corpus must be retrieved in order to extract its nuggets of meta-information. We argue that these problems are inherent in today's web architecture foundation, and that the adoption of a new foundation would allow for completely distributed web systems.

For example, web applications such as emergent *push* technologies do not inherently require a point-to-point communications medium and, instead, would benefit from a multi-point model. However, at present such technologies need to have a priori knowledge of the end-point of each information recipient, thereby requiring recipients to first communicate with the sender and declare their intention to receive the pushed data. Ideally, a true push technology would operate in an uncoordinated manner similar to radio, which permits a sender to blindly transmit information and those users who wish to receive the data need only tune to the correct channel. This is what we propose in this paper.

More specifically, we proposing using multicast [6] [5], which is a many-to-many protocol, as the basis for web information services, to address many of these problems. By using multicast, senders and receivers do not need to know the identity or location of each other ahead of time. Users gain the ability to interact with many web systems at the same time. New web information systems can be added or deleted without having to inform or reconfigure the rest of the web.

This paper argues for using multicast as the basis for web communication services and describes an example web information system, called Jambalaya, that embodies these ideas. By using multicast, Jambalaya is able to distinguish itself from current web information systems by offering additional capabilities. Using Jambalaya, a user can broadcast a request for information without knowing the identities of possible information providers, and, moreover, Jambalaya allows users to make requests targeted to local or global sources of information. Jambalaya also allows information servers to "advertise", for example, by broadcasting popular information that has received many requests, thereby becoming active entities that can promote their information.

The rest of the paper is organized as follows. Section 2 discusses background information such as motivations for this work and multicast protocols. Section 3 talks about our goals for using multicast in web protocols. Section 4.1 gives an overview of the Jambalaya system. Section 4.2 discusses some implementation issues. Section 5 delves into related work. Section 6 discusses future research directions for using multicast for the web. Finally, Section 7 presents some concluding remarks.

2 Background

2.1 Motivation

The rise in popularity of the World Wide Web (WWW) has brought a torrent of electronically published material in the form of web pages. This is in addition to the information contained in archives accessible through pre-web systems such as FTP or WAIS. The rapid growth of the volume and diversity of data, however, overwhelm the average person's ability to find a desired piece of information by simple browsing alone. In response, various research and commercial search tools, such as Harvest [3], Archie [2], and web search engines like Yahoo! [18], have appeared in order to facilitate the search for information. Some more recent tools, such as Marimba [14], Pointcast [17], and web publishing [11] [1], use a *push* methodology reminiscent of radio or television to send to subscribers a packaged set of information. But like all unicasting-based information systems, all of these technologies are constrained by the need to be point-to-point.

The most common tool used in the Internet today are web search engines as epitomized by AltaVista, Hotbot, and Yahoo!. However, current search engine technology has several limitations. For example, users must know the Internet address of any search engine that they might want to use.[1] Also, each search engine represents a point of failure. Since different commercial search engines attempt to differentiate themselves by the type or organization of the information that they carry, it is not always easy or possible to simply switch to another engine when the preferred one fails. Finally, for the most part, search engines store only Internet-wide knowledge; there is no way to query only local sources.

In order to search the Internet for information to index, search engines rely on *web crawlers*. These crawlers iterate through a set of known pages and follow all links on these pages to discover new pages. This method can be problematic for several reasons. Because of the ever-increasing growth of the volume and diversity of data, this method already struggles to keep pace and maintain a current web index. Web crawlers also place a heavy load burden on servers and the network, because of the manner in which they gather information. Oftentimes, many related and inter-linked web pages are located on the same server. By following links to discover new pages, crawlers will access the same web server several times in rapid succession causing congestion at that server. Also, congestion on the intervening network links is caused by the retrieval of the whole document corpus so that it can be indexed. Moreover, much of the information is discarded by many crawlers — for example, some crawlers retain only the HTML anchors from a web page.

As a result of the large effort involved in cataloging the web, the indexes actually used for user queries are only updated every once in a long time. Even though it is possible for a web author to submit a link to be cataloged, the submitted link may still not appear for a long time. Even worse, the author would have to submit his link to several different search engines in order to ensure as much distribution for his link as possible. Given the long times between web index updates and given the fast pace that information is added or deleted from the Internet, web indices typically are rife with dangling links or missing pages.

[1] Even so-called "meta-search engines" or "Para-Sites", such as MetaCrawler [15] suffer this limitation — users must know the location of the meta-search engine, and the meta-search engine is restricted to a fixed set of search engines about which it knows.

The multiplicity of different interface designs of Internet search engines is also problematic. Users now need to learn the foibles of each new search system's interface. Para-Sites, such as the MetaCrawler [15], attempt to homogenize the search engines by providing one interface to many search engines. However, new search engines appear frequently and para-sites use only a small preselected subset of them.

Web page authors have little or no control over the representation of their page by search engines. For instance, it is the administrator or some automated program, not the author, who decides such things as what categories a document will fall under and what query keywords will access that document. Also, the description of a document that today's search engines offer can often be just the first few lines of text from the document. Unless the author premeditatively constructed the document such that the first few lines act as a summary, this scheme can fail to give a meaningful description.

Recently, new commercial and research tools have emerged that use the Internet to continually send to subscribed users a prepackaged set of information. Such so-called *push* technology is similar in concept to the programming channels available on radio or television. Today's push technology, however, is really an *extended pull* since a user must first signal the central server his desire to receive the packaged set of information. Current push technology, such as Pointcast, can place a heavy load on the Internet because it uses a separate TCP connection between itself and each subscribed user. This uses network bandwidth inefficiently since the same information needs to be sent multiple times.

2.2 Multicast Description

Multicast is a many-to-many connectionless internetwork protocol which provides an efficient method of transmitting a datagram from a sender to a dynamic group of receivers. Like unicasting, multicast is part of current and future Internet protocols. This group of receivers is defined to be a set of zero or more hosts which are identified collectively by a single Internet Protocol (IP) address. The multicast protocol is responsible for transporting the data packets efficiently through the network by, essentially, creating a shortest path routing tree from the sender to the set of receivers. Note that only one copy of each data packet travels along each link of the routing tree. However, as in the point-to-point unicast datagram service, the basic transmission medium is considered unreliable and the internetwork only guarantees to give a "best-effort" to deliver the datagrams intact and in the correct sequence to all members of the destination group. Several research efforts [12] have proposed improvements to the basic multicast protocol so that it guarantees reliable transport of data packets.

Membership in a multicast group is unrestricted and dynamic. Any host in any location may join any number of multicast groups. Conversely, once a host joins a group, it may leave the group at its discretion. There are no restrictions on the number of members that a group may have. Multicast groups are considered to be *open* in that a sending host need not be a member of the group to which it is sending. Note that at no point in time does a sender know the individual identities of the receiving hosts.

A sender may make distinctions between local and global receivers. The multicast protocol allows a sender to specify a *range* for datagrams. Range is defined as the number of routers a datagram may traverse before being dropped. In this manner, only receivers within the range will receive the sent datagram. Furthermore, multicast traffic can be con-

Multicast Thresholds	
Range	Range
Host	0
Subnet	1
Site	32
Region	64
Continent	128
Unrestricted	255

Table 1: Multicast *thresholds* which prevent datagrams with a smaller range from leaving an administrative domain.

strained to reach only those within a certain administrative domain. Several standard range *thresholds* exist that, by convention, limit a multicast's reach to a certain range of sites. By the varying the range, a sender specifically states whether the multicast should pass beyond the boundary of the domain. The standard range values for different domain thresholds are shown in table 1.

Recent trends in multicast protocol development, such as Protocol Independent Multicast (PIM) [8] and the next generation of the Internet Protocol (IPv6) [7] [9], lead to greater multicast transmission efficiency and more flexibility for a sender. PIM specifies a network-friendly method of setting-up and maintaining the routing paths through which multicast datagrams travel. Using PIM Sparse-Mode, connection to a core-based multicast routing tree is receiver-initiated. This lightens the burden on the network for transporting and initiating multicasts. IPv6 differentiates between local and global multicasts by encoding within the multicast group address itself whether the group is local or global. Also, IPv6 allows the root of a local multicast to be different from the location of the sender. By allowing the root to be elsewhere, IPv6 allows a user to send a multicast only to those group members near an arbitrary point in the Internet. What we are proposing is consistent with this.

3 Goals

The ultimate purpose of using multicast instead of unicast for web services would be to increase the flexibility of the World-Wide-Web for both clients and servers. The flexibility goals that we hope to accomplish and some examples of the functionalities that we would like are as follows:

Sender Blind Communications

Senders should not need to have a priori knowledge of the identity, location, or number of receivers. Users should be able to interact with all servers at once.

Distributed Systems - Not Centralized

Information systems should be completely distributed, with no point-of-failures. Users should be able to add or delete web servers at will without coordination. All web servers should be able to respond to information queries. As an extreme case, each

web page would listen and respond to queries. Any information changes should be immediately reflected.

Local vs. Global Queries
Users and advertisers should be able to make distinctions between local or global sources of information.

Efficient Network Use
Users should be able to send out a single request and have the network transport that request to all qualifying servers. All messages, whether pushed or pulled, should travel the shortest path from the sender to all receivers with only one copy of the message traveling down common paths. By having each information server be able to respond to requests for information about their content, there is no need for the gathering of web page meta-information to a single site. Hence, there is no need for web crawlers or the submission of web pages. This reduces the load on intervening networks and servers.

Total Control to Authors
Authors should have full control over the representation of their work. This includes how their work is categorized, what keywords it responds to, and the description of the document.

4 Jambalaya

4.1 Overview

Jambalaya[2] leverages the potential of multicasting to create a true *blind distributed search paradigm* that breaks with the current tradition of centralized search and push models of web-based information access. It demonstrates these ideas in two ways. First, users can perform sender-blind broadcasting of information requests. Second, information servers can perform sender-blind advertising of their most-often requested information. The key insight is that, through the use of multicast, Jambalaya operates in such a way that clients and servers do not need to know the identity or location of each other. Users can take advantage of Jambalaya's diffusely distributed nature by being able to choose to query/advertise either locally or globally. Web authors are empowered by having greater control over the way their documents are presented in answer to user queries.

The Jambalaya system is composed of two main components: the client interface and the meta-information server (MIS) (See Figure 1). While this involves the use of new software at the client and server sides, the Jambalaya system is constructed using the JAVA language so that users can continue to use the Internet browsing tools that they are already familiar with.

[2]Jambalaya is a thick, rich, hearty stew served in the city of New Orleans and the surrounding bayous. It is concocted from the various vegetables and meats that happen to be available when the stew is made. As such, though the basic recipe is the same, each locale's version comes out different.

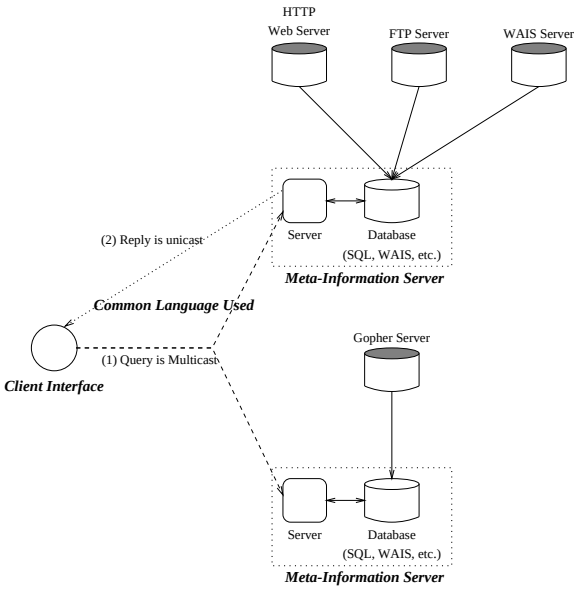

Figure 1: Querying Using the Jambalaya Information Search System

For all information searches, Jambalaya provides a single graphical user interface which executes at the user's computer. The client interface uses the multicast protocol to efficiently transport the query from the client to all of the meta-information servers. The client sends the query *blindly* to the group of all MISs (where the network assumes the responsibility of correctly routing the queries to all MIS servers). The interface allows a user to make distinctions between local and global information by utilizing multicast's range option. Specifying a range for a query gives the user more control over where the information can come from and who will answer any query (See Table 1). For example, a user (whose computer is connected to the Rutgers University computer network) who wishes to query only those information sources within Rutgers itself would specify the range for the query to be within the "site" only. The interface even allows the user to perform an expanding-ring search for the *nearest* information source with relevant information. (In the near future, with the imminent wide-spread use of IPv6, a user will be able to specify a different root for a query. For example, a user at Rutgers University who wishes to query information sources local to U.C. Berkeley would specify "berkeley.edu" as the root and specify the query range to be constrained within the "site" only.) The various responses from the MISs are gathered at the client interface and displayed to the user.[3]

Clearly, Jambalaya, as the broadcaster of information requests, and MISs, as the recipient of information requests, must be able to communicate in a common language. Although our approach does not require any specific communication language (only that it be one that is common to all participants), for the purposes of Jambalaya we developed a common language that is called the Item protocol that allows a user to specify the type of word-based queries

[3]Currently, no ordering is imposed on the responses from the MISs and they are displayed in the order in which they are received. However, if MISs included some confidence measure or relevance ranking along with the information they return about each document the client interface could potentially use this information to order the documents.

common to most web search engines (see Section 4.2.4). By specifying a protocol between Jambalaya and the MISs it becomes possible to add new services with the sole requirement that they, too, understand the Item protocol, thereby allowing for easy future expansion of the system.

One role of the MIS is to respond to user's queries with useful information. Since the client interface uses multicast to transport the query, all MISs within the user-specified range will receive the query. Once the MIS receives the query, it bears the responsibility of deciding whether it contains information relevant to the query. If it determines that it does possess relevant information, then the MIS will return directly to the user a list of the documents which satisfy the query. If no relevant documents exist, then the MIS will silently ignore the query and return no information to the user. In this manner, the user will only receive replies from MISs which have self-selectively determined that they have information relevant to the query.

This structure for communicating between Jambalaya and MISs also allows information servers to not be co-located with MISs. Each MIS need only have information about the details of the information that it can provide, either from some single information server that it represents, or from a collection of servers that it serves as a gateway to. Since the web data server interfaces can be diverse (i.e., FTP, WAIS, Archie), the MIS should translate the user's query into the specific format for each server. In this manner, the MIS can be paired with any web server (which, for that matter, can themselves be other MISs). It also allows an MIS to serve as a "front end" to legacy/exotic systems. Structuring things in this way requires a user to only understand one query format.

However, in our work thus far, MISs are assumed to be co-located with the information that they represent. Having the MIS be co-located with the information allows the author of the document to be able to specify how the document will be represented to clients. For example, it makes it possible for an author to specify a document's title, description, and the keywords that describe it. Also note that the "dangling links" problem, which is typical of centralised search engines, is less likely to exist here because Jambalaya's distributed nature allows the MISs to be co-located with the information sources and, thus, be continually up-to-date.

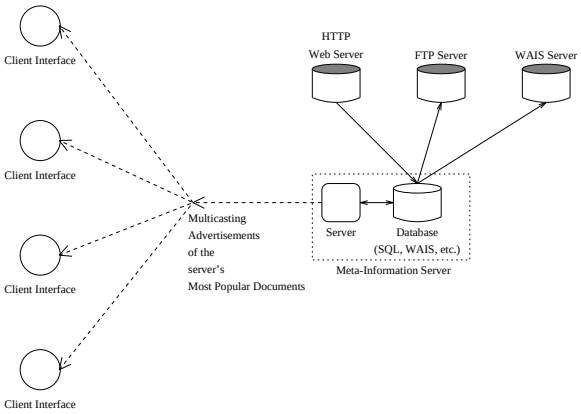

Figure 2: A Meta-Information Server advertising its popular documents using Multicast

Jambalaya also exploits multicast by allowing MISs to actively promote their documents

by efficiently advertising them to all users at the same time. Instead of waiting for a user to search for a popular document through a query and then accessing it, an MIS can, instead, advertise the availability of popular documents. For example, it can do this by multicasting information about the document (such as its title, description, and URL) to the multicast group of all Jambalaya clients. Note that since the group address for all Jambalaya clients differs from the group address of all MISs (See Figure 2), there is no confusion between client broadcasting of information requests and server broadcasting of information advertisements.

Users can decide whether they wish to *tune-in* (or not) to the advertisement multicast group. In this manner, users are not invaded by unwanted document advertisements and their privacy from intrusion is preserved. Also, because of the nature of multicast, a sender to a multicast group does not know (and can not know) the identities or even the number of receivers. Hence, the anonymity of users who decide to receive advertisements is preserved. Even when users decide to receive advertisements, filters can be used to weed-out unwanted advertisements and only let through those in which the user would be interested. Finally, knowledge of such popular information can also make more intelligent web caching possible, where information objects that appear particularly popular can be added to a cache before even seeing a user's request for it, thereby having the potential to decrease user latency.

4.2 Implementation

4.2.1 Client Query Window

The Client Query Window is the main user interface component (See Figure 3). This interface is implemented using JAVA [4] and allows a user to enter a query and then transmit it through the network. Note that, in addition to the user-specified query, the client will also include its host and port number so that MISs are able to unicast responses directly to it.

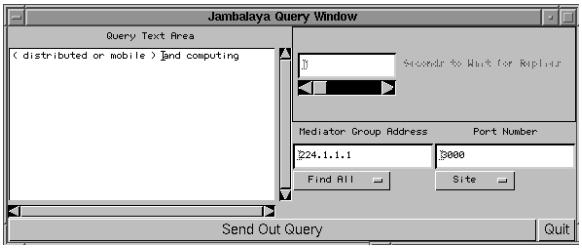

Figure 3: The Jambalaya Client Query Window

The user can set several parameters. These parameters include:

Multicast Address This designates the MIS group address.

Port Number This designates the port number to use in order to contact the MISs.

Number of Responses The user can choose to receive responses from all relevant MISs or to only receive a response from the nearest relevant MIS. The former option sends out a single query to the set range while the latter option will perform an expanding ring search for the nearest relevant server. To do this, it will first transmit the query

with a range of one, wait for a user-specified period of time, then increase the range and retransmit the query until a response is received or the maximum user-specified range is reached.

Time to Wait The user can specify the amount of time to wait for a response to be returned.

Range This menu controls the multicast range for the query. The user is able to choose one of the standard range settings (See Table 1).

4.2.2 Response Window

The Response Window shows the results of a single user query. Multiple queries will cause multiple response windows to appear — one for each query. The window will show for each response that document's title, description, and URL. Selecting one of the items in the window will cause the document to be downloaded, and displayed if Jambalaya is being executed within a web browser.

4.2.3 Advertisement Window

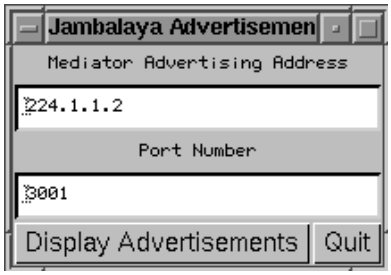

Figure 4: The Advertisement Window

The Advertisement Window allows the user to view document advertisements multicast from the MISs (See Figure 4). This window has the following features:

Multicast Address This designates the multicast group address used to receive advertisements.

Port Number This designates the port number to use in order to receive the advertisements.

View Advertisements Button Users need to press this button in order to view the advertisements. The current implementation follows a policy of not forcing users to view advertisements. Pressing this button will cause a Response Window to appear. As the advertisements are received, they are listed in the response window.

Item Component	Prefix	Suffix
Item	<item>	</item>
Query	<query>	</query>
Host Name	<host>	</host>
Port Number	<port>	</port>
URL	<url>	</url>
Title	<title>	</title>
Description	<desc>	</desc>

Table 2: Item Protocol components and their corresponding prefixes and suffixes.

4.2.4 Item Protocol

The Item protocol breaks down a message into discrete components. For instance a user query would be sent out as one item. An agent response, however, may contain several items in it — one item for each document that satisfies the user's query. The current version of the user interface can understand items that consist of up to six components: query, host name, port number, URL, title, and description. Any item components that are not part of this list will be ignored. In this way, the system can be expanded without having to retrofit the entire system.

Each item and each component of an item is delineated by an HTML-like prefix and suffix. Each item in a message is preceded by "<item>" and followed by "</item>". These two strings denote the boundaries of the information in an item. Only data found between these two strings are part of the item. Similarly, each item component has similar prefixes and suffixes. For instance, the query item component has a "<query>" prefix and a "</query>" suffix. Anything found inside of these two strings is considered part of the query. This allows a query to consist of text, or images, or anything the user may require. Table 2 shows the list of item prefixes and suffixes.

4.2.5 Meta-Information Server and Database

The Meta-Information Server (MIS) is the main component of the Jambalaya System. The MIS is actually composed of two separate processes: an Internet server and a meta-information database (See figure 1).

The operation of the MIS is fairly straightforward. The MIS will listen to the all-MIS multicast group address for queries sent from the Jambalaya client interface. When a query arrives at the MIS, the Internet server components will parse the query and create a new query in the language specific to the local meta-information database (SQL, EBL, etc). It uses the new query to see if the database contains any information about documents relevant to the query. If any documents are relevant to the query, the Internet server component will return a description of each document directly back to the user. Each document description contains a title, a short description, and the document's URL.

The MISs are able to track which documents are actually retrieved by users. When responding to a query, the MIS will change the document's URL so that it will redirect any

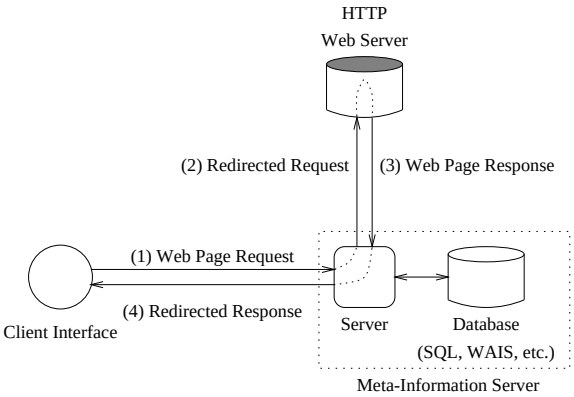

Figure 5: Meta-Information Server permutes URLs so that all web page requests must first go through itself.

document accesses through the MIS. When the user selects one of the MIS's responses in order to display its contents, the redirected URL causes a connection to be made to the MIS which sent that query response and not the actual web server (See Figure 5). The MIS will record the fact that the document was accessed and then contact the appropriate web server itself, download the document, and pass it on to the user. Note that this process is transparent to the user. In this manner, the MIS can keep track of which pages were actually accessed as a result of its advertising or responses to queries. As a result, they are able analyze the tracking data to discover relationships. For example, the current implementation is able to distinguish between those documents which are frequently accessed (popular) or infrequently accessed. It will then periodically advertise the top ten documents that are the most popular.

5 Related Work

One of the first attempts to address the inefficiencies of the current web architecture is the Harvest Project [3]. The Harvest system attempts to more productively use the Internet servers and network bandwidth by concentrating on improving individual system components. For instance, Harvest web crawlers are designed to be executed at the information providers site, thus reducing the network load. However, while efficiency gains were realized, by not breaking with the current web architecture, the Harvest project only addresses a subset of the problems tackled by this paper.

The Inktomi project at Berkeley [13] attempts to ameliorate the point-of-failure problem by creating a distributed search engine which executes on several computer hosts at the same time. Thus, the loss of a single part of the engine does not cause the engine as a whole to fail. However, there is still only a single point of entry to the system.

Para-Sites, such as the MetaCrawler [15], attempt to homogenize the search engines by providing one interface to many search engines. However, the information that a user seeks may not exist in the small preselected subset of engines that the Para-Site uses. Para-Sites, therefore, are at best a short-term solution to the problem of locating the addresses of search engines.

Recently, some of the larger search engines, such as Yahoo!, have attempted to provide

some information based on location. However, this information is typically available only at the granularity of a city or small country. In the few instances where a finer granularity is possible, such as using postal zip-codes, the returned information was found to be incomplete. For example, the zip-code 08855 failed to report the presence of the Rutgers University Busch Campus. Furthermore, using a large centralized system to search for local information is counter-intuitive and suffers from the same problems common to large centralized systems.

Two new research web publishing systems from Rutgers University [11] and Brown University [1] both attempt to improve upon the push model by, among other things, allowing the end-users to request changes in the order that the packaged information will be sent. However, these systems only exist as theoretical concepts or as simulations.

6 Future Directions

Currently, the user interface will gather all of the responses from the MISs and simply concatenate the document lists in the order in which they were received. However, future efforts will concentrate on organizing the responses such as by title, author, or server of origin. One can also imagine the use of machine learning techniques to learn which MISs have provided particularly pertinent information in the past. Preferential ranking could then be given to the responses from these MISs when the user interface determines the order in which to display the query responses.

Leveraging the use of the multicast range option as a tool to distinguish between local and global sources of information relies on a network-centric view of the world. An information source is considered local if the number of intervening subnets is small. However, the layperson intuitively thinks of the world from a geographic-centric view-point and in this case the multicast range option is not as useful because the distance between two subnets can vary greatly. In order to bring the concept of local vs. global information into the layperson's geographic-centered view of the world, a combination of geographic routing [10] and multicast needs to be used. Geographic routing allows a message to be transported through the network to users in an arbitrary geographic region. In this manner, a user could retrieve information from servers that are geographically close to him without having to worry about network distances. Future research will investigate pairing Jambalaya with geographic routing.

7 Concluding Remarks

Basing web information services on multicast greatly increases the flexibility of web information systems and allows for *sender-blind communications* such as querying and advertising. By blind, we mean that clients do not need to know a priori the addresses of information servers and, likewise, advertisers do not need a priori knowledge of the recipient addresses. The use of multicast assures a minimal disruption of the underlying network while still reaching all intended receivers. Clients can send out a single query and have the network transport that query to all qualifying servers. Furthermore, using multicast's range option, clients and advertisers can make distinctions between local and global information. A benefit of using

multicast is that clients and servers can be added randomly without having to inform or reconfigure the rest of the system.

Since multicast can deliver a query to all servers at once, centralized information systems are no longer needed. Hence, using multicast communication protocols allows for completely distributed systems. All web servers could respond to information queries. By being distributed and pushing the capability of responding to queries to the very same people who publish the documents, the web is better able to keep up with the ever-increasing growth of the WWW content. The ability to control when and how to respond to user queries empowers authors by giving them the ability to individually choose how to represent themselves. This control also allows web authors to quickly update their local servers so that any document content changes are reflected promptly to user queries.

Being distributed and using multicast leads to efficient network usage. For instance, the Internet as a whole is helped by minimizing the network and server loads, bottlenecks, and points-of-failures associated with today's centralized search engines and web crawlers. Also, advertising messages travel the shortest path from the sender to all receivers with only one message traveling down common paths.

Jambalaya, an example prototype web information system which is based on multicast, was also described. Jambalaya provides one interface for all web searching. The Jambalaya system components then translate the user query into the appropriate format for their own meta-information database. This system design makes possible one query front-end but many database back-ends, thereby allowing system administrators to choose the appropriate database to handle queries about their web content. This design also allows the integration of legacy systems in a less painful manner. The current system advertises a server's ten most popular documents, but other criteria could be used. Also, when advertising or responding to queries, only the description is sent instead of the whole document. This allows for use in both high-speed and low-speed Internet connections.

Acknowledgements

The authors wish to thank Tomasz Imielinski for many helpful discussions and other valuable contributions to this work.

References

[1] S. Acharya, M. Franklin and S. Zdonik, *Balancing Push and Pull for Data Broadcast*, Proceedings of the ACM SIGMOD, May 1997.

[2] *Archie*, http:// www.bunyip.com/ products/ archie/ archie.html

[3] C. Mic Bowman, Peter B. Danzig, Darren R. Hardy, Udi Manber and Michael F. Schwartz. *The Harvest Information Discovery and Access System.* Computer Networks and ISDN Systems 28 (1995) pp. 119-125.

[4] Mary Campione and Kathy Walrath, *The Java Tutorial: Object-Oriented Programming for the Internet*, Addison-Wesley, ISBN 0-201-63454-6, 1996.

[5] S. Deering. *Host Extensions for IP Multicasting*. STD 5, RFC 1112, Stanford University, August 1989.

[6] S. Deering. *Multicast Routing in a Datagram Internetwork*, Ph.D. Thesis, Stanford University, December 1991

[7] Deering, S., and R. Hinden, Editors, *Internet Protocol, Version 6 (IPv6) Specification*, RFC 1883, Xerox PARC, Ipsilon Networks, December 1995.

[8] Deborah Estrin et al., *Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification*, Internet Draft, March 1997. Work in progress.

[9] Hinden, R., and S. Deering, Editors, *IP Version 6 Addressing Architecture*, RFC 1884, Ipsilon Networks, Xerox PARC, December 1995.

[10] J. Navas and T. Imielinski, *Geographic Addressing and Routing*, Proc. of the Third ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom'97), Budapest, Hungary, September 1997.

[11] T. Imielinski, S. Viswanathan, B. R. Badrinath, *Energy Efficient Indexing on Air*, Proc. ACM SIGMOD Conference, May 1994.

[12] B.N. Levine and J.J. Garcia-Luna-Aceves. *A Comparison of Reliable Multicast Protocols*. ACM Multimedia Systems, August 1998.

[13] *The Inktomi Technology Behind HotBot: A White Paper*, http:// www.inktomi.com/ CoupClustWhitePap.html

[14] *Marimba, Inc.*, http:// www.marimba.com/

[15] *MetaCrawler*, http:// www.metacrawler.com/

[16] *New Media Watch*, http:// www.mediametrix.com/ interact_mmnewmedia.htm, March 1998.

[17] *Pointcast, Inc.*, http:// www.pointcast.com/

[18] *Yahoo!*, http:// www.yahoo.com/