Personalization & Asynchronicity to Support Mobile Web Access *

Abstract

The World Wide Web has become a global distributed information system with a rapidly increasing user community. The advent of mobile computing coupled with the recent developments in wireless communications and personal computer technology, has created a new challenging area: mobile information access. Accessing web based information sources is likely to be one of the most important applications of mobile computers. However, the software and protocols associated with the WWW were designed with static hosts in mind. The resource limitations of mobile clients, low bandwidth of the wireless network and frequent disconnections are situations that existing WWW systems are ill equipped to deal with. There is a need to overcome these constraints and provide reliable information access from mobile computers. In this paper we present W^3IQ, a system which supports mobile access to the web using asynchronous collaborative information retrieval techniques. We describe the model of W^3IQ - its architecture, software implementation, its capabilities as an information filtering mechanism. We also present experimental results.

*This work was supported in part by an IBM Faculty Development Award. Support of Lycos who have allowed to access to their internal API is also acknowledged.

1 Introduction

The advent of mobile computing, which provides users ubiquitous access to the sources and services of the networked information infrastructure, has enabled new applications in the areas like tele-medicine, public information services, battlefield awareness and education etc. These applications typically involve accessing multimedia information from heterogeneous sources over a wireless network, using systems originally designed for the wired network. The problems faced by a system designed for a stationary environment when it switches to a mobile environment are well documented in [1]. These include the limited power and memory of the mobile clients and their "doze off" mode of operation, high latency, limited and variable bandwidth, error prone transmission and frequent disconnections of the system. Thus, there is a need to develop middleware which alleviates these problems and seeks to provide the mobile users the same degree of responsiveness and performance as a wired environment.

Retrieving information from the web [19] is now an integral part of most computer user's daily routine. It is perhaps the largest distributed source of information. The current model of web browsing, however, leaves the burden of finding relevant information on the user. The user has to search through the documents and follow the links to find information that s/he needs. Bandwidth, a precious resource in the mobile scenario, is thus wasted by transmitting a lot of useless data across the wireless channel in the browsing process. Given the synchronous nature of the HTTP protocol, a continuous connection is required throughout the information retrieval process and disconnections represent a catastrophic phenomena. Further, web servers have no knowledge about the client machine resources and assume that the client machines can handle the data sent by them. For example, a user may follow a hyper-link to a MPEG video on a machine that does not have the capability of playing MPEG. This results in consumption of available bandwidth and wastage of time for transferring data across the network. Moreover, the mobile client expends battery resources in receiving this data, which it cannot use.

Providing a reasonable level of performance in the face of frequent disconnections, restricted bandwidth and limited resources is a major issue in mobile computing. We refer to the information retrieval from WWW in a mobile environment as "disconnected browsing" [16]. The term "disconnected" is used to subsume what has sometimes been described as "weakly connected" access as well. Our work proposes a software architectural to support disconnected browsing [16] which is based upon the widely accepted model of mobile computing [2]. Two components of the system are W^3IQ and Mowser. Mowser is an active transcoding proxy that allows the mobile user to set his/her viewing preferences, based on the network connection and available resources. It involves transcoding of data depending on its type to suit the network QoS parameters [9] . This paper deals with W^3IQ, whose objective is to manage disconnections and save bandwidth by providing the user the right information through information filtering and collaborative information retrieval techniques. The W^3IQ system provides an interface between different W^3IQ servers in the network to share information and support collaborative filtering methods. From our experimental results, we observe that collaborative information retrieval often pinpoints the information the user is looking for, or at least significantly narrows the search space. W^3IQ thus takes

up the burden of information retrieval on behalf of the user, saving wireless bandwidth and the MH's resources. The idea is to utilize and reuse the information already existing in the system to answer user's queries, analogous to the natural social recommendation process [21]. Thus personalization is used to support web browsing in the mobile scenario.

In the following section, we mention related work in this area done by various research groups and organizations. Section 3 describes the software architecture and functionality of the W^3IQ system. The implementation and experimental results are explained in sections 4 and 5 respectively. Finally, the current and ongoing development of the system is described in section 6 .

2 Related Work

A considerable amount of work has been done in the area of information access from mobile platforms. The Client-Proxy-Server model has begun to feature in many mobile applications to overcome the constraints of limited bandwidth in the mobile computing scenario.

Much of this work concentrates on transcoding. The Mowgli model [10] consists of two mediators, the Mowgli Agent and the Mowgli Proxy located on the mobile host and the mobile-connection host respectively. They use the Mowgli HTTP protocol to communicate with each other, which reduces the number of round-trips between client and server. A specialized transport service, the Mowgli Data Channel Service is used for reliable communication between the mobile-connection host and the mobile host. Mowgli WWW reduces the data transfer over the wireless link in three ways: data compression, caching, and intelligent filtering of the document before sending it to the client. It only performs GIF to JPEG conversion, and large embedded images are not transferred at all to the mobile node.

We have developed Mowser, a proxy based system to support web browsing from mobile clients over wireless networks [9]. It is a proxy agent between the mobile host and the web server, that performs active transcoding of data on both upstream and downstream traffic to present web information to the mobile user according to the QoS parameters set by the user. Active transcoding is defined as modifying the HTTP stream in situ. The process is entirely transparent to the user. This is an improvement over other proxy based systems, which only transcode images on the downstream and are mostly not configurable.

The InfoPad[17], Daedalus[12], and Glomop[11] projects from Berkeley focus on mobile aware wireless information access. Katz suggests [12] that mobile systems must be able to detect the transmission environment and exploit knowledge about its current situation to improve the quality of communications. The authors call it "situation awareness". InfoPad employs a dumb terminal and offloads all functionality from the client to the server. Daedalus and Glomop use dynamic content type "transcoding" or "distillation" of data to utilize bandwidth while transmitting it to a mobile host. Glomop operates a proxy server on a stationary machine, through which the mobile users make their requests. The proxy server retrieves the requested documents and then forwards these documents across the slow

wireless link in a format useful to the mobile user.

Another line of work has sought to support disconnected operation using local caching. Several offline browsers are commercially available. WebWhacker[6] and WebFetcher[8] are offline browsers that download documents into a local disk for later viewing. The user specifies a set of links, similar to the bookmarks file, which the application downloads and caches. The amount of data (files) being downloaded is controlled by limiting the number of links traversed for each URL or by the amount of disk space used. Traveling Software's WebEx [7] and Open Market's OM-Express [13], run a proxy web server on the portable machine to deliver cached documents. After downloading the requested documents, the user sets his/her browser to point to the proxy server running on the mobile host. The proxy server then serves documents from its local cache, without using a network connection. Both provide flexibility in scheduling webpage downloads. However, neither allows compression of the downloaded pages, which would greatly reduce storage constraints. Offline cache, a Netscape Netcaster feature, allows users to download information from an information broadcast channel, save it in a cache on their system's hard drive, and view it at a later date.

The common feature of these systems is that they propose to conserve one scarce resource (connection bandwidth) by using up another scarce resource (disk space). Furthermore, none of the mobile platform's resource limitations is considered in these systems. We believe this is a very narrow view of "disconnected operation".

While not yet used in the context of mobility, the use of "personalization" techniques has recently gained popularity in web searching. Such personalization is typically done using collaborative filtering. Collaborative filters help people make choices based on the opinions of other people. GroupLens [23] is a system for collaborative filtering of netnews to help people find articles they will like in the huge stream of available articles. News reader clients display predicted scores and make it easy for users to rate articles after they read them. Rating servers, called Better Bit Bureaus, gather and disseminate the ratings. The rating servers predict scores based on the heuristic that people who agreed in the past will probably agree again. The entire architecture is open: alternative software for news clients and Better Bit Bureaus can be developed independently and can interoperate with the components developed.

The Referral web [20] is an interactive system for reconstructing, visualizing, and searching social networks on the World-Wide Web. The idea on which Referral web is based is that searching for a piece of information is equivalent to searching the social network for an expert on the topic together with a chain of personal referrals from the searcher to the expert. The network is used to guide the search for people or documents in response to user queries.

Phoaks is a collaborative filtering system that continuously parses, classifies, abstracts, and tallies opinions posted by web users. The three design principles that distinguish Phoaks from other computer mediated collaborative filtering systems are role specialization, reuse, and recontextualization. The design features, system architecture and the Phoaks URL recommendation interface are discussed in [18].

Siteseer [22] is another web page recommender system that uses an individual's bookmarks and the organization of the bookmarks within folders for predicting and recommending relevant pages. It forms dynamically defined virtual communities of interest by looking at each user's folders and bookmarks. It also measures the degree of overlap (such as common URLs) of each folder with other people's folders. It uses URLs as an unique identifier and completely ignores titles. As it is purely collaborative, it does not help the first-time user, since there is no collective experience to leverage until a community is discovered similar to user's tastes.

Fab [15] is a hybrid content based collaborative system. It tries to utilize the advantages of both the content based, as well as collaborative systems. It maintains user profiles based on the content analysis and directly compares these profiles to determine similar users for collaborative recommendation. Its recommendation process has two stages: collector stage, that gathers pages relevant to a small number of topics, and computer clustered groups of interest, which track the changing tastes of users. These pages are then delivered to the user through a selection stage, which allows them to go through the recommendations and rate on their personal interest. If a particular user rates a page high, then this page is sent to other users sharing the same tastes to the collection agent that adapts to his profile.

The Savvy Search[3] meta-search engine is designed to query other search engines likely to return other useful results and by responding to fluctuating load demands on the Web. Savvy Search learns to identify which search engines are most appropriate for particular queries, reasons about resource demands and represents an iterative parallel search strategy as a simple plan.

In summary, there has been a lot of work done independently in the areas of information retrieval and filtering as applied to the web, and transcoding of multimedia data to support mobile access to the web. Our approach to mobile web access integrates collaborative information retrieval and filtering techniques, disconnection management and transcoding. The architecture of this model and the functionality of the system are discussed in the next section.

3 Architecture and Functionality

The software architecture of our system is based on the popular mobile networking model described in [2]. The mobile hosts (MH) are supported by mobile support stations (MSSs), which act as gateways between the wired and wireless network and support mobile hosts within their cell. Our architecture (logically) adds a proxy server to each MSS in the basic MSS-MH model. This functions as the gateway to the Web for the browsers on MHs and supports the "disconnected browsing" functionality. These multiple proxies can be viewed as a distributed multiagent system [5] which cooperates to solve the disconnected browsing problem. Figure 1 illustrates the proposed architecture. Notice that the equation of the proxy agent with the MSS is only logical. Physically, the proxy can be on any host on the wired network, although typically it will be on a host not too many network hops away from the proxy.

As mentioned earlier, our proxy agents have two components. One, called Mowser[9], provides the

transcoding functionality. The other, called W^3IQ, handles disconnection and personalizes the web space to limit the data flow on the wireless network. This latter component is the subject of the paper and is described next.

HTTP is a synchronous protocol – connection has to be maintained from the time the request is made until the response returns. In the mobile scenario, disconnections are frequent and so the existing protocol does not serve us well. We have designed W^3IQ so as to make web access asynchronous. W^3IQ splits browsing into a set of asynchronous operations and maintains the state information while disconnections occur. These operations are transaction like, and provide the functionality to (i) request information, (ii) retrieve the results of the request and (iii) rate the URLs obtained as a result of the request. In addition, there are administrative operations like creating accounts etc. Connection is required only during the execution of these transactions [1] . State information consists of data about the user, his/her query, the obtained URLs and the results of the queries not yet rated by the user. Thus there are allowable points of disconnection in the system, from where the user can reconnect and proceed further. W^3IQ allows the user to specify requests as queries, rather than explicit URLs and takes up the burden of locating the "right" information on behalf of the user.

Broadly, the operation of W^3IQ can be described as follows: The user connects to the proxy, makes a request, and disconnects. The proxy then tries to assemble a set of URLs that would satisfy the request. When the user connects again, the results of his prior request(s) are made available. These results are URLs that the system feels will contain the information the user requested. Users then prioritize the URLs provided by the system by rating them, depending on their perception of the relevance of the URL. The user feedback allows the system to infer a "semantic" match between the user's query terms and the information in the document. This helps the system create a user's profile which is used to control the information he/she would receive in response to future queries. The ranking information, the URLs, and the corresponding query are all stored by the W^3IQ proxy as metadata. Over time, then, the proxy builds a repository of ranked URLs which can be used to answer future queries by this user or other users.

The W^3IQ system allows the proxy to pass the query to other W^3IQ proxies in the network, to garner any information related to the query in their local metadata. This allows all proxies to share information (specifically, ratings in regard to particular query terms) about URLs obtained by one proxy.

The process of evaluating a query proceeds as follows: To answer a user's query, the proxy first looks at its local metadata to see if this user has queried on the same terms before, and if so uses the ranked URLs to respond. If not, it queries its peer proxies about their metadata from all users in an order determined by the user's "closeness" to other proxies. This enables collaborative information retrieval [4]. If this fails, or not enough results have been returned so far, then the proxy passes the query to

[1] The term *transaction* is used here in a looser sense than its strict database interpretation. More specifically, no claim regarding ACID properties is made.

Lycos, and uses its results.

The closeness of a user to a particular proxy (or more specifically, its rating metadata) is determined by the difference in rankings for the URLs returned from a common query. In other words, the difference between the rank given to a URL by a user and its (aggregate) ranking in a proxy is computed and averaged over several URLs. The lesser the difference in rankings, the closer the user to (the rating metadata at) the proxy. A user qualifies for the grouping after rating a minimum number of URLs. Thus the "closer" W^3IQ servers are queried first for a new query by a user. Note that peer servers may be closer to a user that the local server of the user. This provides for "global aggregation" of resources. For a new user, the cache of URLs on the local server is queried first and the peer servers are searched in an arbitrary order.

Reusing previously obtained URLs which were highly ranked by other users with similar interests means that URLs the user likely wants will be returned. The proxy thus minimizes the bandwidth it uses over the mobile link as well as the power it forces MH to consume by sending information which is relevant to the user's query. By coming up with the closeness notion, we try to make our system net friendly. In other words, when a "new" user appears in the system, his query is broadcast to all peers to obtain information. Over time, as the peer information sources best for this user are determined, the broadcast mode is replaced by a selective communication mode to the "chosen" peers.

4 Implementation of W^3IQ

The implementation, a combination of Perl, Tcl/Tk scripts and C++, is split into four modules: filter-user interface, server internals, interfaces with other information resources and the peer to peer protocols with the other W^3IQ servers. The message exchange between the MH and the proxy is designed to be minimal, so that the connection durations between them can be as short as possible.

4.1 Filter-User Interface

The user connects to the system, presents a query and disconnects. At a later time, when the user logs on, he is provided with a list of pending results for queries he submitted earlier. The user may view them then, or at any other time s/he reconnects. The query results are shown till the user rates them. The rated links are stored in the user profile and the system metadata. If the results include those obtained from other users, a "semantic distance" of interest is calculated between the current user and the users from whom the URLs have been retrieved. This is used for future web searches by the user as described earlier. The filter-user interface supports administrative operation modes such as account creation, login process etc.

The system uses several "databases[2]". These include the keyword-User database – which stores

[2]The data is currently stored in the flat file format.

information regarding users who have queried on particular query terms; Metadata database for each user – which contains the keywords of the user's queries and the corresponding URLs with their ratings; and a Distance database for each user – which stores the "distance" of the user with the W^3IQ servers.

4.2 Server Functionality

The server receives the request from the filter, processes it, and responds to it. The request can be to create an account or to log on to the system, query service or to view the pending results and submit ratings. The user-password, the rated links, and the pending results are stored by the server database system. W^3IQ server maintains the state information of a user storing the query keywords, the unrated links and constantly updating the semantic distance of the users with other in the system for every submitted ratings.

4.3 Collaborative Information Retrieval

The information retrieval in the W^3IQ system is collaborative. It is collaborative in the sense that the URLs rated by the users are processed for future queries. The system aggregates the users ratings (recommendations) as inputs and uses them to respond to future queries, thus functioning as a recommender system as well.

A W^3IQ server maintains URLs-keyword database for every user, that contains all the links the user previously rated. These databases are used by the server to respond to another query on the same set of keywords, at a later point of time. Also maintained is a file which contains the semantic distance of the user with the other users.

The implementation of the W^3IQ server follows the following algorithm to obtain links in response to a query.

1. The users links database is queried for the required number of links.

2. If step 1 fails, the distance file (data containing the relationship between the user and the W^3IQ servers) of the user is read and based upon the data, a request is sent to the appropriate server in order till the required number of links are retrieved. If no "close" server exists, the query is broadcast.

3. When a server receives a request from step 2, the list of users in the system who have queried on the particular keyword(s) is extracted and a search is conducted on their links database. If the requested number of links are found, then the results are returned to the querying server.

4. If step 3 fails to accumulate the requested number of links, then the original server tries to query other Web information resources. Currently Lycos is used as a default source to fetch links related to the particular keyword. We have access to Lycos's internal database structure to retrieve URLs

from their URL repositories. This means that for boolean queries we use the same semantics for A 'and' B & A 'or' B as Lycos does.

4.4 Peer to Peer Protocols

The standard client-server model was used to make the W^3IQ servers communicate with one another. Each W^3IQ server listens at a particular port to receive requests from its peers. Every W^3IQ server maintains a list which contains information about the ports of all the servers in the network.

Since the W^3IQ server fetches information from different sources (its peers, Lycos etc), there are cases where users in different servers rate the same link differently depending on the relevance of the URL to them. When the server retrieves these URLs from different sources, there could be duplicate links retrieved from different sources. There needs to be a method to discard redundant information collected. In such cases where duplication of links (multiple occurrences of the same URL with different rating) occur, the links rated by users "closer" in interest to the querying user are given priority over recommendations of other users in system. The W^3IQ server follows the above convention and displays to the user all the links retrieved from different sources after getting rid of the duplicate links (multiple occurrences of the same URL with same rating) collected. W^3IQ server takes the burden of retrieving information away from the user to facilitate disconnected operation. Lycos has exposed their internal server API to us, which W^3IQ uses to directly retrieve raw URL and rating data.

5 Experimental Results

Extensive testing has been done and we have a small user base (students in the group) which use the sytem on a semi-regular basis. We illustrate here some experimental scenarios, and the results that were collected during testing.

1. We show the information retrieval and filtering feature of W^3IQ in figures 2 and 3. User A rates the URLs for the query "mobile and browsing" (figure 2) and submits them. When reconnected later and searching on the same keywords User A gets the links s/he rated earlier in sorted order(figure 3).

2. The collaborative retrieval techniques of the system are displayed. Assume three users A, B and C on three different W^3IQ servers. The user profiles are calculated based on their searches. From the experiment conducted it will be shown that since users A and C are "close" together in terms of interest, user C gets the rated URLs of user A rather than those rated by user B for the same query. This is shown as follows: User A queries on the keywords "mobile and browsing", rates and submits them (figure 3). User B queries on the same keywords, rates the obtained the URLs and submits them (figure 4). Now the user C is queries on the same keywords. The results s/he gets are those of submitted by user A as seen from the figure 5 .

We have made the system available on the web, the system may be accessed from the following URL *http://www.cecs.missouri.edu/~joshi/dbrowse/*.

6 Discussion & Ongoing Work

The paper discusses the W^3IQ system which facilitates disconnected browsing. A software architecture suited for this task, which integrates transcoding, disconnection management and personalization is presented. W^3IQ makes Web access an asynchronous operation, thus providing a facility to manage disconnection. The design and implementation of W^3IQ to facilitate personalization through collaborative retrieval of information is discussed. This process allows the user to specify the information s/he needs, and recieve URLs collected from several sources. The "filtering" process attempts to make these the URLs the user needs, thus avoiding bandwidth on the wireless side that would have been otherwise wasted by the user in trying to search through the web space to retrieve this information. Clearly, we trade off proxy CPU cycles, and perhaps even extra traffic on the wired side to achieve this goal. A brief summary of the experimental results are presented.

In ongoing work, we are extending the system to introduce categorization of search areas. The basis of this extension is that users having similar interests in "sports" may not share similar interests in "literature". This will add more information filtering capacity to the system. Also fuzzy clustering methods to group the users on the system to enable a more reliable and faster information retrieval.

Moreover, the resources available are themselves changing over time. Clearly, the user cannot be expected to know of all possible resources, nor to track them over time. We propose to use recommender agents to facilitate this process and make mobile computing ubiquitous. In [5] , we describe a multiagent system. Each agent/server knows only a part of the knowledge corpus needed to recommend appropriate resources to the user, and must cooperate with other agents/servers to perform its advisory task in a scenario where the number of agents and their capabilities evolve over time. We will leverage this work to create a multiagent recommender system to make distributed information access for mobile computing ubiquitous.

References

[1] Acharya, A, Badrinath, B.R., Imielinski, T. and Navas J. "A WWW-based Location Dependent Information Service for Mobile Clients", *http://cs.rutgers.edu/dataman/papers/loc-dep-mosiac/Overview.html*, 1995.

[2] Badrinath, B.R., A.Acharya and Imielinski, T. "Impact of Mobility on Distributed Computations.", *Operating Systems Review*, 27:15 - 20, 1993.

[3] Adele Howe and Daniel Dreilinger. "SavvySearch: A Meta-Search Engine that Learns which Search Engines to Query", *http://www.cs.colostate.edu/ howe/pubs.html*, January, 1997.

[4] Kavasseri, R. , Keating, T. Wittman, M., Joshi, Anupam and Weerawarana, S., "Web Intelligent Query - Disconnected Web Browsing with Collaborative Techniques", *Proc. First IFCIS Conf. on Cooperative Information Systems.* 1996.

[5] Anupam Joshi, Ramakrishnan, N. and Houstis E.N. "MultiAgent Systems to support Networked Scientific Computing", To appear in *IEEE Internet Computing*, Vol. 3, 1998.

[6] WebWhacker. "WebWhacker Product Description", *http://www.ffg.com/whacker/index.html*, 1995.

[7] WebEx. "WebEx Product Description", *http://www.travsoft.com/products/webex/2.0*, 1997.

[8] WebFetcher. "WebFetcher Product Description", *http://www.ontv.com/webfetcher/*, 1995.

[9] Bharadvaj, Harini, Joshi, A., Auephanwiriyakyl, Sansanee "An Active Transcoding Proxy to Support Mobile Web Access", *Technical Report of the Dept. of CECS, Univ. of Missouri-Columbia*, April, 1998.

[10] M. Liljeberg, M. Kojo and K. Raatikainen. "Enhanced services for world-wide web in mobile wan environment", *http://www.cs.Helsinki.FI/research/mowgli/mowgli-papers.html*, 1996.

[11] A Fox, S.D. Gribble, E.A. Brewer and E. Amir. "Adapting to Network and Client Variability via On-Demand Dynamic Distillation", *http://daedalus.cs.berkeley.edu/index.html/recent_publications*, 1996.

[12] Randy Katz "Adaptation and Mobility in Wireless Information System", *http://daedalus.cs.berkeley.edu/index.html/recent_publications*, 1994.

[13] "OM-Express product description." *http://www.openmarket.com/*, 1995.

[14] A.D. Joseph, A.F. deLespinasse, J.A. Tauber, D.K. Gifford and M.F. Kaashoek, "Rover: A ToolKit for Mobile Information Access", *Proc. 15th Symposium on Operating Systems Principles*, ACM, December 1995.

[15] Balabanovic, M. and Shoham, Y., "Content-Based, Collaborative Recommendation", *Communications of ACM*, Volume 40, pp. 66-72, 1997.

[16] Joshi, A. and Weerawarana, S. and Houstis, E.N "On Disconnected Browsing of Distributed Information", *Proceedings of the Seventh International Workshop on RIDE*, IEEE Press, pp. 101-107, 1997.

[17] InfoPad group, "InfoPad Project Description," *http://infopad.eecs.berkeley.edu/*, Web Page.

[18] Terveen, L.G. and Hill, W.C. and Amento, B. and McDonald, D. and Creter J., "PHOAKS", *http://www.acm.org/sigchi/chi97/proceedings/paper/lgt.htm*, Web Page.

[19] T. Berners-Lee, R. Caillau, J.-F. Groff and B. Pollermann, "World Wide Web: The Information Universe", *Electronic Networking: Research, Applications and Policy*, 2(1):52-58, 1992.

[20] Kautz, H. and Selman, B. and Shah, M., "Combining Social and Collaborative Filtering", *Communications of ACM*, Volume 40, pp. 63-65, 1995.

[21] Resnick Paul and Varian, Hal R., "Recommender Systems", *Communications of ACM*, Volume 40, pp. 56-58, 1997.

[22] Rucker, J. and Polanko, M.J., "Personalized Navigation for the Web", *Communications of ACM*, Volume 40, pp. 73-75, 1997.

[23] Konstan, Joseph A. and Miller, Bradley N. and Maltz, David and Herlocker, Jonathan L. and Gordon, Lee R. and Riedl, John. "Applying Collaborative Filtering to Usenet News", *Communications of ACM*, Volume 40, pp. 77-87, 1997.

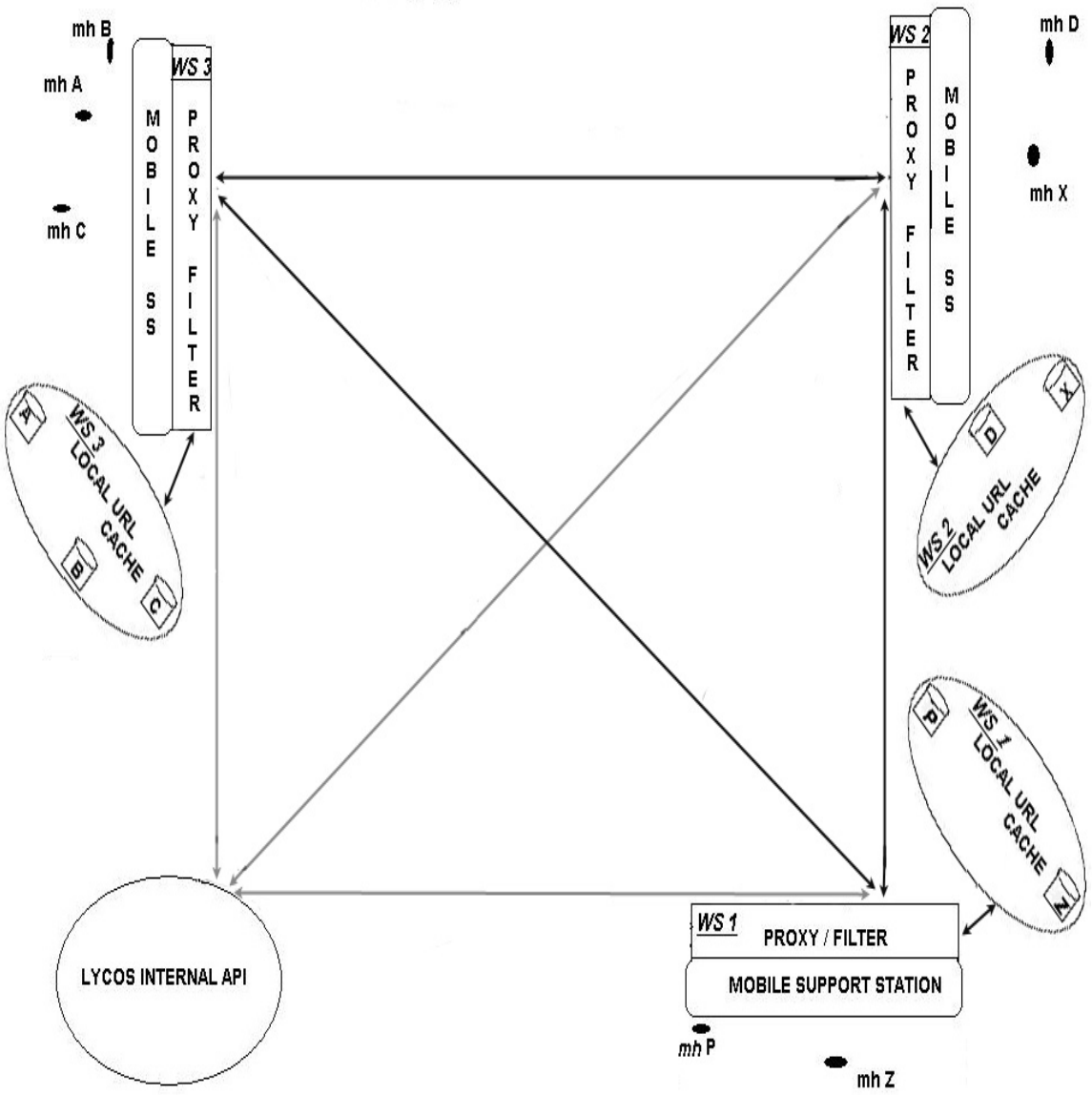

Figure 1: Network Architecture of W^3IQ

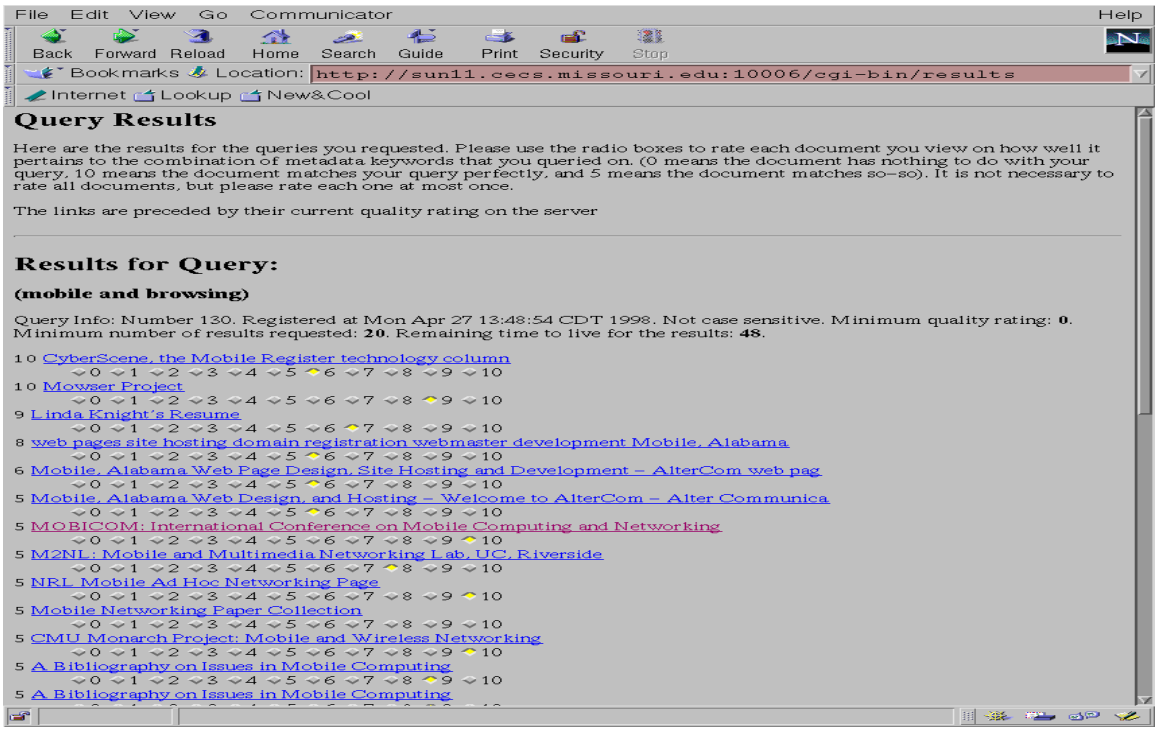

Figure 2: The dynamic HTML page created when the user A wants to see the results from his query.

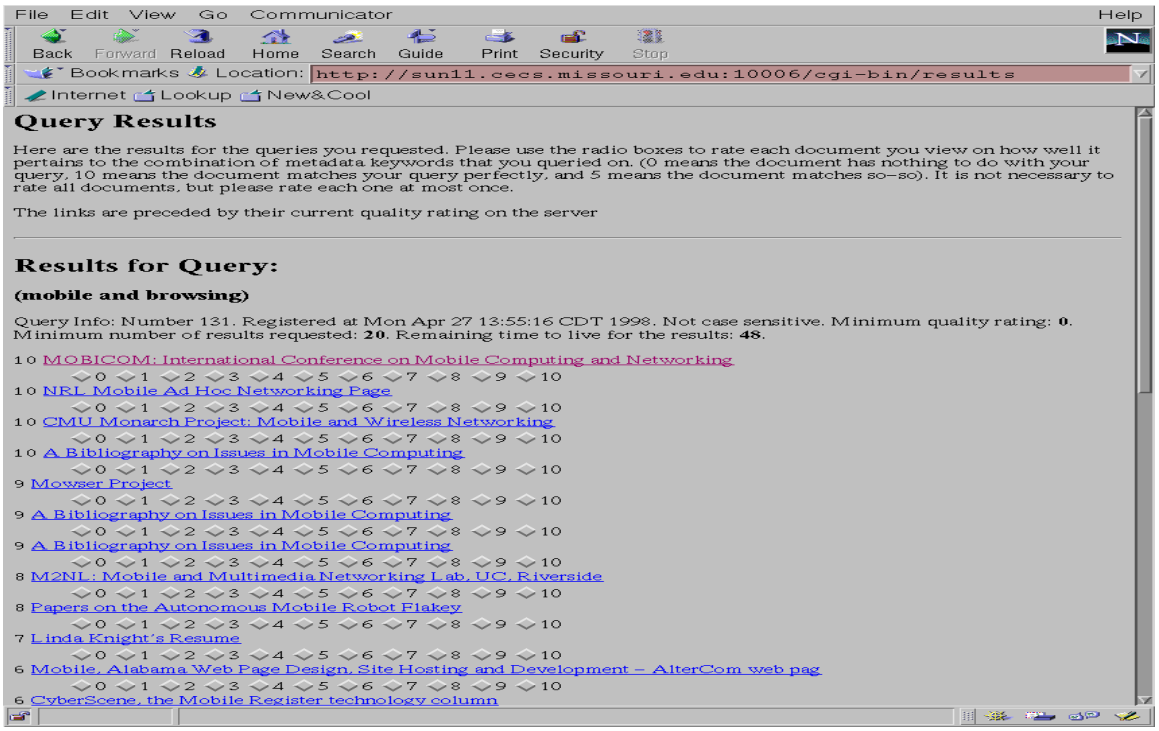

Figure 3: The dynamic HTML page created when the user A, logs back and queries on "mobile and browsing" The URLs rated by him/her earlier are restored.

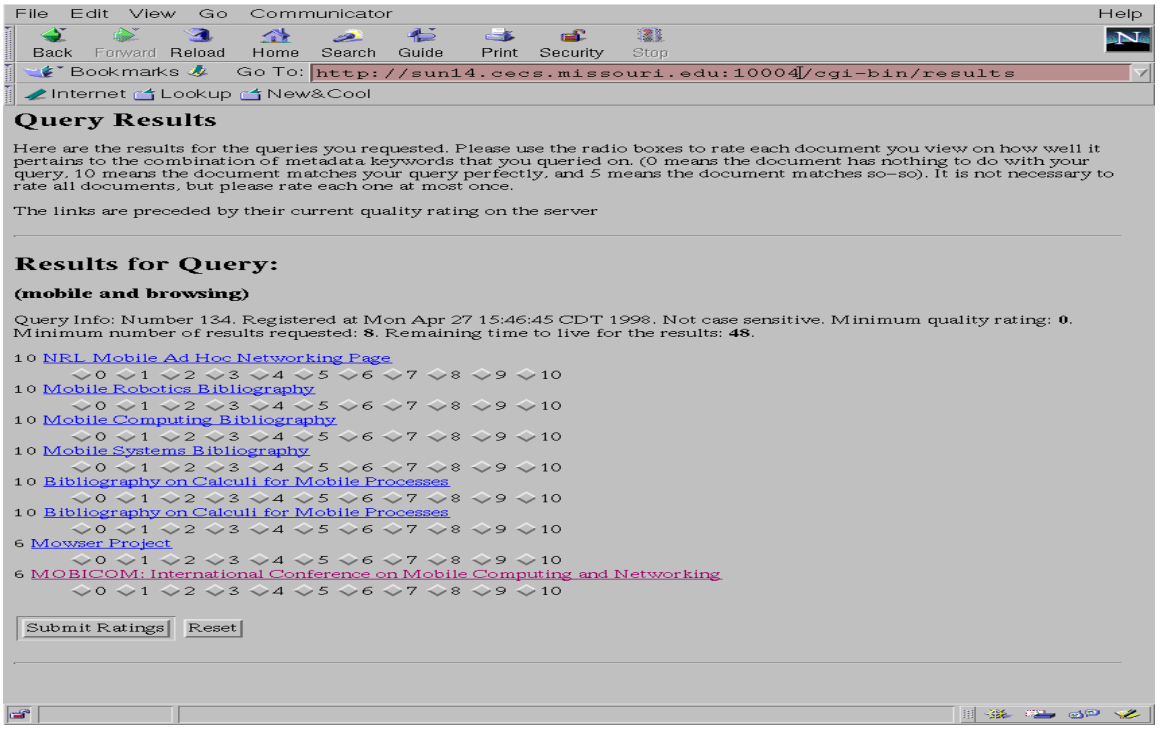

Figure 4: The URLs rated by user B for search query "mobile and browsing".

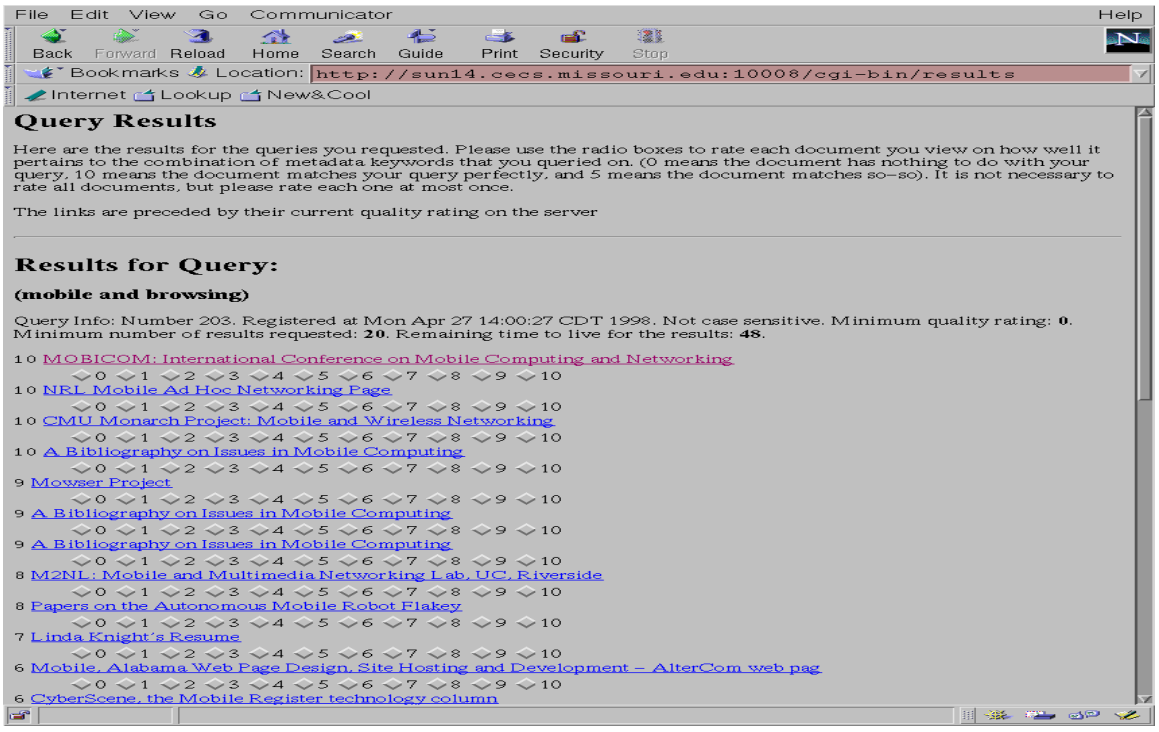

Figure 5: The URLs obtained for user C whens s/he queries on "mobile and browsing". The URLs rated by A are obtained.