On Disconnected Browsing of Distributed Information*

Anupam Joshi
Department of Computer Engineering and Computer Science
University of Missouri
Columbia, MO 65211
joshi@ece.missouri.edu

Sanjiva Weerawarana and Elias N. Houstis
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907-1398
E–mail:{saw,enh}@cs.purdue.edu

Abstract

The software and protocols associated with information browsing systems are largely designed with static access points and wired networks in mind, HTTP and the Web are a case in point. Static hosts are connected to wired, high bandwidth networks, and are capable of transmitting and receiving large amounts of data without significant delays. As such, the size and format of the data files being received by the browser/client has never been a concern. However, this causes problems when information access is desired on mobile hosts (MH), since data transmission over a wireless network is much slower than on a wired network. Mobile computers are also relatively resource-poor, compared to their desktop counterparts. This fact is ignored by HTTP servers, and large data files are transmitted to computers that cannot properly display them. Also, mobile computers operate in constantly changing network environments. It is possible for a mobile computer to become temporarily disconnected from a network when it changes base stations or goes out of range of a base station. A mobile host may also doze off *to preserve battery power and thus be disconnected. The information browsing system and protocols assocaiated with mobile computers should thus be able to tolerate the fault of temporary disconnection. This work focuses on addressing these problems in the context of web browsing from a mobile host. The current model of Web browsing is inherently sequential, and wasteful of bandwidth. This paper investigates an efficient model for browsing and describes the design of a smart Web browsing application which performs transactions based on the user's available resources and manages disconnection.*

*This work was supported in part by NSF awards ASC 9404859 and a grant from the Intel Corporation.

1. Introduction

Recent years have seen a significant increase in interest generated by mobile computing. With the advent of the World Wide Web[3], browsing (surfing) the web is becoming an increasingly common activity for computer users. The current model of browsing the web, however, is sequential, and one that leaves most of the burden of finding the relevant information on the user and his computer. In other words, a user is expected to know starting points, and then search through documents and follow links to find the information s/he needs. Clearly, this means that while the user is searching for information, a network connection needs to be constantly maintained. Moreover, a lot of (potentially) useless information is transmitted over the (wireless) network, wasting precious bandwidth. This mode of information access is clearly not suited for web browsing from a mobile platform. Mobile platforms, for one, are connected over wireless links. Even the best of such networks provides far lower bandwidth than run of the mill wired networks. Wireless LANs typically provide a total bandwidth of 2Mbps, and throughputs in the hundreds of Kbps range. Wide area wireless networks (mostly cellular) operate at tens of Kbps. Moreover, unlike wired networks, disconnections are a frequent phenomenon in wireless networks. These occur while a host is between base-stations, or when it falls in a "radio-shadow" area. Disconnections in the mobile environment can also be elective. In other words, the mobile host may chose to "switch off" certain part(s) of its functionality in order to preserve battery power. We refer the reader to [5] for details of the mobile computing scenario. A related problem is the capability of the mobile platform. Web servers do not take into consideration the network connection between them and their clients. Rather, the server merely returns whatever document was asked for. Furthermore, users are routinely creating multimedia-rich pages. In fact, so heavily is multimedia used that often "text only" versions of these pages have to be made available for users accessing them across WANs. Wireless networks do not have the data transmission speeds of wired networks, so downloading large files over wireless networks, even local area ones, consumes a longer time. Not only does it take longer to download large files, but more battery power is consumed, reducing the time the mobile computer can be used. Web servers also have no notion of the resources available at the client's end, and blindly assume that the client is capable of properly displaying the data that it receives. For example, a user may follow a hyperlink to a sound file on an X terminal which is not capable of playing sound. The server will blindly supply this data to the client, consuming time and network bandwidth to deliver a sound file that cannot be used. Such problems are especially acute on mobile computers, which are not as resource-rich as their desktop counterparts. Text only pages, while helpful in this context, force an unnecessary all or none choice on the user.

While we have articulated these problems in the context of web, they generally relate to information access from mobile platforms to distributed data. The Internet protocols are primarily designed for static networks where changes are rare and disconnections catastrophic. HTTP is a typical such protocol, and has no knowledge about the state of the system, including the underlying network. HTTP/1.0, the version of the protocol used most widely today, establishes a new TCP connection for every GET request. The three way handshake involved in establishing the TCP connection can be a significant overhead on a wireless link. HTTP/1.1 has adderessed this problem, and the proposed HTTP-NG[19], presently being discussed, will go farther by having some notion of the context of the request.

The attempt in this work is to add some intelligence to the present http client server system in a manner that will be interoperable with HTTP-NG. Providing a reasonable level of performance in the face of frequent disconnections, restricted bandwidth and limited resources is a major issue in mobile computing. In context of information access, we refer to this problem as *disconnected browsing*. Note that our use of this term is in a much broader sense than in some commercial products. In these products, the only concern is to be able to operate in a mode where predictable disconnections from the network occur in an otherwise high bandwidth connection. An argument can be made that these bandwidth and resource related problems are transient, and eventually wireless network speeds as well as the resources available on mobile platforms will increase. That may well be the case, yet the speeds on wired networks and the resources (memory, CPU speed, etc.) on static hosts will also increase. Existing (and novel) applications will evolve and be dependent on these enhanced capabilities. Thus while the absolute performance measures of mobile systems will undoubtedly improve, what we call the *bandwidth gap* and *resource gap* will remain. In this paper, we describe a software architecture to support disconnected browsing, and present preliminary implementations of two component systems called *Mowser* and *WebIQ*.

2. Related Work

There is little existing work in the area of web browsing from mobile platforms. There is little existing work in the area of web browsing from mobile platforms. The two major ongoing efforts in this area are MOWGLI and Glomop. Mowgli[12] is a comprehensive project to design Mobile Office Workstations using GSM Links (hence the acronym). One of the efforts in this direction is a web browser. This is accomplished by putting a proxy on the support station, and an agent on the mobile host. Mowgli Web browser's primary strengths is its use of a single TCP connection to handle multiple GETs over the wireless link. This reduces the massive overhead of establishing a new TCP connection for multiple get requests within a document, which requires a three way handshake. It also caches the incoming documents at the proxy end. Glomop[6] has an approach similar to the mowser component[9] of our system. Glomop scales the images down to a smaller size before transmitting them to the mobile user. This is an instance of what the authors call type specific distillation. This reduces the bandwidth requirement between the mobile platform and static host, and allows for faster transmission times. It also allows the user to re-request parts of the distilled document at greater resolution. However, it does not have a notion of asynchronous operation. Another initial effort was the Teleweb project[17] by Schilit et. al., which proposed a reactive architecture to control the costs of information access. Faced with a GET request, the proxy would either get the document, or depending on the network connection, defer the request till a better connection could be obtained. The user could override the proxy's decision. In addition, it cached documents at the proxy end.

The two mutually distinct areas of related research are offline browsing and dynamic browsing. The first is the perusal of information available on the WWW, while being disconnected from the network. Most current work in this area involves downloading entire documents to the mobile platform and caching them for later browsing. We believe this approach does not suit a resource-poor platform as it poses additional burdens in the form of larger disk requirements. Dynamic browsing, on the other hand involves integrating location information to the query being sent. This information is used to retrieve information relevant to the user's current location. While these projects suggest interesting uses for the WWW, they are based on additions to the current protocols, and do not address the problem of viewing the information that is currently available on the WWW, from a resource-poor mobile platform that is frequently disconnected.

Several offline browsers are commercially available. WebWhacker[21] , and Grab-a-Site[7] are offline browsers that download documents into a local disk for later viewing. WebWatch[20] is a similar product, that in addition, periodically looks for changes on the cached sites. Milktruck[13], another offline browser, runs a proxy web server on the portable machine to deliver the cached documents. The common feature of these products 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 are considered in these systems. We believe this is a very narrow view of "disconnected operation".

Mobisaic[18] is a system that extends the current WWW protocols to encode location information in the URL. It uses a "dynamic environment" introduced by Xerox PARC[16] to deduce the mobile user's current location and create a "dynamic URL". Dynamic documents[10] are another example of mobile access to the WWW. In this case, the burden of processing location dependent information is placed on the mobile computer, thereby using more of its resources. Another location-dependent information service is described in [1], with a description of a building navigator application. These applications propose extensions to the current WWW and its protocols, in order to deliver information relevant to a mobile platform (i.e. location information). However, they do not deal with the issues of disconnectivity, and resource-scarcity of mobile platforms.

Some prior work, in unrelated contexts, has been done on searching for information on the Internet. Locating specific information in the Internet is becoming more difficult due to the its explosive growth and diversity. Many search engines and research papers have attempted to address this problem. However, these works for the most view this as a pure information retrieval problem. The reader is likely familiar with several "crawler" based search engines such as Yahoo, AltaVista, Lycos, InfoSeek, etc. etc.

3. Software Architecture

Our proposed software architecture builds on the widely accepted model for mobile computing [2]. In essence, it assumes that the wired network is a high bandwidth pipe. Failures of either the network or host machines on this part of the infrastructure are rare and catastrophic events. The mobile hosts, on the other hand, can frequently disconnect, either electively to preserve resources, or otherwise due to a dropping of the wireless connection. The mobile hosts (MH) are supported by mobile support stations (MSSs) which act as gateways between the wired and wireless networks. Each MSS supports MHs within its cell, similar to how cellular phones are organized. MHs within a cell can directly communicate with each other, otherwise they must communicate through their MSS, which in turn talks to other hosts and MSSs on the wired network.

Our architecture adds a proxy server on each MSS to the basic MSS-MH model. This functions as the gateway to the Web for browsers running on MHs. All information requests will be routed through this proxy, which will operate in a manner transparent to the user. The MHs will be able to still run the user's favorite Web browser, as long as it is forms capable and allows the provision of a proxy mechanism. The proxy will implement the new model we are proposing for disconnected browsing. Essentially, the objective is to provide some state information which will enable us to handle the constraints imposed by mobility. In other words, we are adding some intelligence to the network, so that it adapts to changing resources available. A more general vision of such a proxy, suited to computational rather than information access tasks, can be found in our prior work[4]

Running a proxy server allows us to push all the intelligence into this unit, and also have it maintain needed state information. This allows us to use existing IP based systems as is, and allows retention of popular tools and software such as Web browsers and servers. In other words, legacy stateless protocols effectively become statefull. For disconnected browsing, the proxy has two distinct but related functionalities, which we call Media Transformation and Asynchronous operation respectively. Mowser[9] performs media data transformation on the documents being served to the browser on the MH, changing their granularity to better reflect the available bandwidth and resources. For example, it will notice that the image being sent to an MH with an 8 bit color display has 24 bit data, and will transform it so as to not make the user wait an inordinate time after the request due to the low bandwidth available. The functionality represented by Asynchronous operation is broader, and is implemented as the WebIQ[11] proxy. HTTP at the moment is a synchronous protocol. In other words, once a request has been made, the connection is maintained till such a time as the response returns. In the mobile scenario, disconnections are frequent, and so the existing protocol does not serve us well. We instead use WebIQ to make the process asynchronous. The user connects to the proxy, makes the request, and disconnects. When the user connects next, the result of his prior request(s) is available. S/he can then fetch this, and disconnect again. The user can connect again to provide the system feedback about the usefulness of this information. Thus the system manages disconnection. Another aspect of Asynchronous Operation is to allow the user to specify requests as queries, rather than explicit URLs. In other words, the burden of locating appropriate information is taken away from the user to facilitate asynchronicity. In the synchronous mode, the user has to explicitly follow links to locate the right information, which, as mentioned earlier, is wasteful of bandwidth. To obtain the results of the queries, the WebIQ proxy first looks at its own cache, then queries its peer proxies, and only as a last recourse uses brute force search engines (Yahoo, Lycos etc.) available on the Internet. So our state information consists of data about the QOS from the network (bandwidth, latency), capabilities of the hardware in the MH, and the preferences and information requests of the user.

Proxy servers also have peer to peer protocols to exchange state information as MHs travel from the cell of one MSS to another. Essentially, our method employs an active join, passive leave technique[2] to initiate transfer of information. This means that when an MH decides (based on, say, signal beacons) that it should switch its MSS, it sends a message to the new MSS signaling a join with its own id and the id of its previous MSS. The new MSS can then request the profile of this MH/User, in terms of the information required by Mowser and WebIQ, from the old MSS. The other advantage of such peer protocols is that information about the state of the world obtained by one peer can be shared by the others. In fact, we view these multiple proxies as a distributed multiagent system which cooperates to solve certain problems.

The proposed software architecture allows us to address research issues such as

- The information needed by a user is spread across heterogeneous and geographically distributed systems. To expect the user to provide the location where the information resides is clearly not correct. Further, the response to a query from the user could involve aggregating information obtained from a variety of sources.

- The current models of information retrieval assume that the browsing software is operational after a request for information has been made. This is clearly not justifiable in the mobile scenario, where disconnections are a fact of life.

- While wired networks are capable of sending and receiving large amounts of data without significant delay, this is clearly not the case in the mobile scenario. In addition to network bandwidth limitations and variability, mobile hosts also have relatively less resources than their static counterparts. These include smaller screen resolution and color capability, and limited or no sound capability.

Most existing approaches to disconnected browsing do not deal with these problems. They limit themselves to essentially caching information locally so as to be able to continue operation even if the network connection goes down. Our approach addresses these issues by augmenting information at each site with ancillary structures storing metadata about the state of the network, the capabilities of the MH and the preferences of the user. The cooperative retrieval process[15] can then use a combination of knowledge–rich and knowledge–poor techniques. In our asynchronous model, the system "dynamically" creates Web pages based on the client requests by locating and retrieving information from across the network. In this sense, our architecture can be viewed as a system to develop multilevel models of hypertext [14] that are appropriate for disconnected browsing. We use the next two sections to further describe the Mowser & WebIQ preliminary implementations.

4. Mowser

Mowser (Mobile Browser) allows a mobile user to set his or her viewing preferences, based on the network connection and available resources. Each mobile host's preferences are known to the proxy server. They are stored according to the mobile host's IP address, allowing the MH to access them from anywhere. The only requirement placed on the browser is that it be able to support proxy servers.

4.1 Servers

Logically, the functionality of Mowser can be split along the lines of a preferences server and a caching server. The preferences server has two functions, to get the preferences for a MH at the start of a browsing session, and to update them whenever the user requests so. A browsing session is started on a MH by the Web browser contacting a preference server and saving preferences. These preferences may also be updated anytime during the browsing session. Such changes could reflect a change in network connection or availability of different resources during different sessions. Once the preferences are stored, the proxy server starts the browsing session by loading the starting URL defined in the preferences. All HTTP *GET* & *POST* requests from that point are handled by the proxy server. If a file requested by the MH does not meet the requirements described in the preferences, the proxy server can modify the file before sending it to the MH. The proxy server identifies the file type by means of its MIME type header. Depending on the type of the file, different conversions can be performed on it to meet the MH preferences. If a file does not meet the size requirement for its type, the proxy server will not send it to the MH. Files that cannot be properly displayed by the MH will not be sent either. This granularity reduction is easy to achieve in some cases; for example, image files can be compressed by sacrificing quality without sacrificing semantics. In particular, the size of the image, and the number of colors can be reduced, resulting in a smaller image that conveys the same information. Whenever a file is modified by the proxy server, a URL will be provided to retrieve the original unmodified file, if the user so requires. Figure 1 illustrates the conversion done by Mowser on the homepage of one of the authors. The current implementation handles images, with ongoing work adding in other data types such as video.

Figure 1. Original and Converted pages using Mowser

5. WebIQ

In our asynchronous model, we view Web access as a sequence of small, atomic transactions. The connection between the MH and the proxy system need only be maintained during such transactions; there is no need for a continuous connection. These transactions involve retrieving information and providing metadata about the information. The WebIQ proxy minimizes the bandwidth it uses over the mobile link as well as the power it forces the MH to consume. This is done by weeding out information which has a low probability of interest using user profiles. User profiles are thus important in determining what information will be useful. Users explicitly rate the URLs provided by the system, based on how useful they were. This gives the user control over the kind of information he or she would like retrieved, and allows the system to infer the user's profile. From another perspective, the user feedback allows the system to infer a *semantic* match between the user's query and the information in the document. Oft–repeated actions of users, e.g. fetching the stock market URLs every morning before 9 am, should be detected and automated. This would enable the MH to receive data during periods of low activity, saving time and CPU cycles.

WebIQ[11] uses existing information resources in a cooperative manner to find information. Each proxy will maintain a ranking of its peers, as well as external information brokers, based on the volume of information requested, quality value required for the results, and quality of previous responses etc [8]. Such information can be used to guide a query to a particular search engine, based on past information, or to a proxy which might be caching relevant URLs in response to a query by another user. Reusing previously obtained (and cached) URLs which were ranked highly by the user prevents a brute force keyword based search for every query. It also allows our metadata to indirectly capture the semantic content of the URL. In general, the proxy will contain the notion of user groups and use these to direct queries. Specifically, let us say that a user asks for URLs related to *football*. When rating the returned URLs, the rankings will be different depending on whether the user was European/Asian or American, since this word means different things to these two groups. Which rankings should a system use when another user queries for information related to football. Obviously, knowing what group the new user belongs to can help the proxy return appropriate cached URLs. This notion of groups and sharing information brings up the question of security and privacy. While we are interested in these issues, they are beyond the scope of the work discussed here.

6. Conclusion and Ongoing Work

In this paper, we present techniques that facilitate disconnected browsing – browsing the web from wirelessly connected, resource limited hosts typical of mobile systems. A software architecture suited for this task in particular, and for mobile information access in general, is described. Two components of this architecture are presented in detail, and their preliminary implementations described. One, called Mowser, handles the automatic transformation for multimedia data from the server to a form that the client can handle. The other, WebIQ, makes Web access an asynchronous operation which is relatively oblivious to disconnections. It also reduces the amount of useless (from the user's point of view) information that needs to be transmitted on the wireless link.

Our ongoing work has three major thrusts. The first relates to software improvement. We are trying to rewrite our software in Java/Jeeves and Oracle NCA to make it platform independent. While at present this adds a performance penalty, it should not be so in the near future with the availability of native Java compilers for many machines. Along the same lines, we are trying to convert the file system based database used by WebIQ to use a commercial system such as Oracle. We are also enhancing the various media types that Mowser can handle and convert to bandwidth saving formats. An interesting question arises in this context in terms of the active components of the web, such as applets. There is at present no characterization of the compute capabilities that an applet needs, nor its functionality. As such, there is no easy way to filter out applets that are not suited for a mobile host. We are presently engaged in trying to come up with ways of characterising these. The second thrust is to implement the peer–peer protocols for these systems. While we have defined these, the implementation is still not complete. Finally, we are currently deploying these systems internally and testing them with users. We would like to make them available more widely and conduct detailed studies of their utility in terms of ease of use, as well as the tradeoffs involved in our design choices.

References

[1] A. Acharya, B. Badrinath, T. Imielinski, and J. Navas. *A WWW-based Location-Dependent Information Service for Mobile Clients.* http://www.caip.rutgers.edu/ navas/loc_dep_mosaic/Overview.html, 1995.

[2] B. Badrinath, A. Acharya, and T. Imielinski. Impact of Mobility on Distributed Computations. *Op. Sys. Rev.*, 27:15–20, 1993.

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

[4] T. Drashansky, S. Weerawarana, A. Joshi, R. WeeraSinghe, and E. Houstis. Software architecture of ubiquitous scientific computing environments. *ACM - Baltzer Mobile Networks and Nomadic Applications*, 1(4), 1996.

[5] G. Forman and J. Zahorjan. The Challenges of Mobile Computing. *IEEE Computer*, 27:38–47, 1994.

[6] Glomop. *Glomop description.* http://HTTP.CS.Berkeley.EDU/ fox/glomop/, 1995.

[7] Grab-a-Site. *Grab-a-Site product description.* http://www.bluesquirrel.com/store/116p.htm, 1995.

[8] A. Joshi. To learn or not to Learn .... In *Lecture Notes in AI: Proc. IJCAI '95 Workshop on Learning and Adaptation in Multiagent Systems.* Springer–Verlag, 1995.

[9] A. Joshi, R. Weerasinghe, S. McDermott, B. Tan, G. Bernhardt, and S. Weerawarana. Mowser: Mobile platforms and web browsers. Bulletin of the IEEE Technical Committee on Operating Systems and Application Environments, 1996. Vol 8, no. 1.

[10] F. Kaashoek, T. Pinckney, and J. Tauber. Dynamic documents: Mobile wireless access to the www. Workshop on Mobile Computing Systems and Applications, December 1994.

[11] R. Kavasseri, T. Keating, M. Wittman, A. Joshi, and S. Weerawarana. Web intelligent query - disconnected web browsing using cooperative techniques. In *Proceedings of the First Intl. Conf. on Cooperative Information Systems*, pages 167–174. IEEE Press, 1996.

[12] M. Liljeberg, T. Alanko, M. Kojo, H. Laamanen, and K. Raatikainen. Optimizing world-wide web for weakly connected mobile workstations: An indirect approach. In *Proc. 2nd International Workshop on Services in Distributed and Networked Environments*, June 1995.

[13] Milktruck. *Milktruck delivery description.* http://www.milktruck.com/cpsisv/index.htm, 1995.

[14] J. Myfield and C. Nicholas. Snitch: Augmenting hypertextdocuments with a semantic net. *Intl. Journal of Intelligent and Cooperative Information Systems*, 2:335–351, 1993.

[15] T. Oates, M. Nagendraprasad, and V. Lesser. Cooperative Information Gathering: A Distributed Problem Solving Approach. Technical Report TR-94-66, UMASS, 1994.

[16] B. Schilit, N. Adams, R. Gold, M. Tso, and R. Want. The parctab mobile computing system. pages 34–39. Proceedings of the Fourth Workshop on Workstation Operating Systems, October 1993.

[17] B. Schilit, F. Douglas, D. Kristol, P. Krzyzanowski, J. Sienicki, and J. Trotter. Teleweb: Loosely connected access to the world wide web. *Computer Networks and ISDN Systems*, 28(711):1431, 1996.

[18] G. Voelker and B. Bershad. Mobisaic: An information system for a mobile wireless computing environment. Workshop on Mobile Computing Systems and Applications, December 1994.

[19] W3 Consortium. Http-ng problem statement. http://www.w3.org/, 1995.

[20] WebWatch. *WebWatch product description.* http://www.specter.com/, 1996.

[21] WebWhacker. *WebWhacker product description.* http://www.ffg.com/whacker.html, 1995.