

# Mobisaic: An Information System for a Mobile Wireless Computing Environment

Geoffrey M. Voelker and Brian N. Bershad  
Department of Computer Science and Engineering  
University of Washington

September 19, 1994

## Abstract

Mobisaic is a World Wide Web information system designed to serve users in a mobile wireless computing environment. Mobisaic extends the Web by allowing documents to both refer and react to potentially changing contextual information, such as current location in the wireless network. Mobisaic relies on client-side processing of HTML documents that support two new concepts: *Dynamic Uniform Resource Locators (URL)* and *Active Documents*. A dynamic URL is one whose results depend upon the state of the user's mobile context at the time it is resolved. An active document is one that automatically updates its contents in response to arbitrary changes in a specified environment, such as a user's mobile context. This paper describes the design of Mobisaic, the mechanism it uses for representing a user's mobile context, and the extensions made to the syntax and function of Uniform Resource Locators and HyperText Markup Language documents to support mobility.

## 1 Introduction

This paper describes Mobisaic, a system that uses the World Wide Web [Berners-Lee et al. 92] to enable information browsing in a mobile computing environment. Information browsing is an ideal mobile application because it allows users to interact with their environment as they work within it, places minimal requirements on a user-input device, and cannot be handled with large on-board caching. With mobile information browsing, users can discover who and what is in their immediate surroundings, whether it is colleagues at a business meeting, speakers at a presentation, projects in a lab, or displays in a museum. A mobile information system can also allow users to execute general queries that incorporate information from their environment, such as finding the nearest cafe or the nearest bus stop that will take them to a specific location. Users of the World Wide Web rely on client browsers to access information servers on the Internet. With Web clients, users browse documents written in the HyperText Markup Language (HTML) by traversing hypertext links, called Uniform Resource Locators (URLs), that load documents or invoke programs on the information servers.

Mobisaic extends standard client browsers in two ways. First, Mobisaic allows clients to insert dynamic information, such as a user's mobile context, into an HTML link before it is resolved by a WWW server. This is done with a *dynamic URL*, which is a hypertext link that incorporate references to dynamic environment variables which are resolved when the link is traversed. Second, Mobisaic supports *active documents*, which include references to dynamic environment variables, but which automatically present and update information for the user as the information the document contains changes or otherwise becomes invalid. The update is done by the client browser, which is notified when the dynamic information changes. Active documents allow browsers to react to changes without the user's involvement.

Dynamic information in Mobisaic is represented within a dynamic environment [Schilit et al. 93B]. Just as shells provide environment variables to customize applications running on Unix (e.g., \$DISPLAY), dynamic environments provide variables which can be used to customize mobile applications. For example,

an HTML document might include a reference to the **Location** dynamic environment variable in order to determine a user's current location.

## 1.1 Related Work

Researchers at Xerox PARC have broadly introduced the idea of context-aware applications [Schilit et al. 93A]. They initially proposed the notion of dynamic environments as a means of representing and disseminating information from a user's mobile context throughout the system. Mobisaic is essentially an application of those ideas to WWW browsing. Projects at DEC WRL and MIT have moved the World Wide Web client interface to a mobile device, although they do not appear to incorporate information from the user's mobile computing context into the system.

## 1.2 Paper Outline

The rest of this paper is organized as follows. Section 2 gives an overview of the World Wide Web system, and describes the extensions to the system used for incorporating a user's mobile computing context into the WWW. Sections 3 and 4 describe dynamic URLs and active documents. Section 5 discusses how Mobisaic can be useful in the desktop environment as well as the mobile environment. Section 6 describes the implementation of Mobisaic. Section 7 summarizes.

# 2 System Overview

This section first provides a high-level overview of the World Wide Web information system. It then describes the extensions used for incorporating a user's mobile computing environment into the system.

## 2.1 The World Wide Web

The three main components of a World Wide Web (WWW) information system are documents, clients, and information servers. The user interacts with the system using a Web client, which lets the user name and load documents from servers for viewing. Web clients typically support a number of different document types, such as ftp, netnews, and Hypertext Markup Language (HTML), and support connections with a variety of information servers, such as ftp daemons, news servers, and Hypertext Transport Protocol (HTTP) daemons. Documents are typically files on a server referenced by Uniform Resource Locators (URLs), and they can contain a variety of information types such as ASCII text formatted according to HTML directives, embedded pictures, audio and video clips, as well as embedded URLs that are used as hypertext links to other documents. URLs can also name programs on HTTP daemons that, when executed, produce an HTML document as output.

## 2.2 Extensions for a mobile WWW

In its current form, the Web infrastructure cannot easily accommodate mobile clients because information is either statically expressed in HTML documents, or because all non-static information must be explicitly entered through form-based interfaces that run on the client. Moreover, there is no support for automatically updating a document when it, or the reason for displaying it, changes. Our extensions to Web infrastructure include:

- a server that maintains dynamic environment variables within a client-specific domain,
- an asynchronous callback mechanism to notify Web clients when a user's dynamic computing environment changes,
- a syntax for referencing dynamic information in URLs and documents.

## Representing mobile computing contexts

Mobisaic uses dynamic environments to represent a user's mobile computing context. The basic unit in a dynamic environment is the dynamic environment variable, which is conceptually similar to a standard Unix shell environment variable: dynamic environment variables have a name and a value, which can be accessed and changed to customize applications to the user's mobile computing environment just as standard shell environment variables customize applications launched from the shell. However, unlike shell environment variables which are associated with a login process, dynamic environment variables are associated with users and locations, and have indefinite lifetimes. Applications on the network with sufficient privilege can access and change dynamic environment variables, and whatever changes they make can be seen by other applications with sufficient access privileges.

## Notification of changes in mobile computing contexts

*Active documents* allow environmental changes to be reflected in the information displayed to the user because a Mobisaic client can automatically react to changes in dynamic environment variables. If the information in an active document that the client is displaying becomes invalid then the client can be notified of the change so that it can display a more relevant document. Notifications contain the name of the variable that changed and its new value. For example, say that the user changes cell locations in the wireless environment. The wireless communications system that is monitoring the user's location can *publish* the new location by updating the **Location** variable in the user's dynamic environment. If the user were displaying a document that was sensitive to location, the client would have subscribed to the **Location** variable, and would receive a notification informing it of the change. At this point the client could take action in response to the change, such as loading a new document that relates to the user's new location.

## Syntax and Scope

A Mobisaic client relies on a syntax for referencing dynamic environment variables within dynamic URLs and active documents. The syntax supported by Mobisaic is of the form  $\$(environment.variable)$ , where *environment* and *variable* denote a dynamic environment and dynamic environment variable, respectively. For example,  $\$(bershad.Location)$  would reference the name of Brian's current location in the wireless network. Mobisaic also supports a shorthand notation for referencing variables in the user's dynamic environment. If the reference doesn't contain the name of an environment, then Mobisaic assumes that the variable denoted is a reference to the environment associated with the user. Thus,  $\$(Location)$  refers to the **Location** environment variable in the user's dynamic environment.

Mobisaic supports recursive references to dynamic environment variables, so that environment or variable names can themselves be references to dynamic environment variables. For example, given that locations have dynamic environments associated with them,  $\$(\$(bershad.Location).Printer)$  would reference the **Printer** variable in the environment associated with Brian's current location.

## 3 Using Dynamic URLs

Dynamic URLs allow a single URL to return different documents or execute different commands depending upon the state of the user's dynamic environment at the time the query is executed. A URL is dynamic if it includes a dynamic environment variable. For example, in our department we have written HTML documents describing ourselves and the places in which we work. The name space of these documents on our server is well structured, enabling the dynamic URL `http://www/offices/$(Location).html` to return the document describing the user's current location. Another example would be a Web server that has a program **busroute** which takes a starting location, a destination, and a time, and returns an HTML document detailing how to get to the closest bus stop on the shortest bus route to the destination. A dynamic URL to find the bus route to the Space Needle in Seattle would appear as:

`http://www/.../mobisaic/busroute?$(Location)?SpaceNeedle?$(Time.TIME).` <sup>1</sup>

### 3.1 Resolving Dynamic URLs

When a user selects a dynamic URL in a document, the client browser is responsible for resolving all references to dynamic environment variables within the URL. The client obtains the values of dynamic environment variables from the appropriate dynamic environment and replaces the references with the values as strings. When all variable references have been resolved, the result is a standard URL which the client then sends to the server. For example, if a user were in office 225 and executed the location description dynamic query in the previous section, the client would resolve the **Location** dynamic environment variable in the user's context and substitute `http://www.cs.washington.edu/offices/225.html` for the reference.

## 4 Active Documents

Active documents are HyperText Markup Language documents that enable the Web client to automatically react to changes in a user's mobile computing context. In this way the client is able to update the information being displayed without the user having to navigate the Web. It places less of a burden on the user to search for information by placing more of a burden on the author to organize it. This tradeoff is possible in a mobile environment because users who roam the wireless network are quite likely to be interested about who and what is in their immediate surroundings, and the information in a user's mobile context is enough to enable a Web client to do the searching on the user's behalf. In this way, it changes the way users interact with the Web: they spend less time searching for information because the client presents it to them as they interact with their surroundings.

This section describes how to write active documents and how the client handles them when they are being viewed by the user. To illustrate these processes, it also describes a set of pages collectively called the WhereAmI active document. The WhereAmI document is a guide for visitors to our department. Each page in it corresponds to a room in our wireless network, and contains a brief description of the room, links to the home pages describing the occupants of the room, and links to pages describing any projects housed in the room. When a user selects this active document, the client loads the page corresponding to the user's location. Then, as the user changes rooms, the client automatically discards the page describing the old room and replaces it with the one describing the new room. Each page in the WhereAmI active document can be loaded using the first dynamic URL from the previous section:

`http://www/offices/$(Location).html`

Authors write active documents just like they write standard HTML documents, with one addition. They must place a *subscribe* command in the document which lists the dynamic environment variables that the client must subscribe to when it loads the document. In effect, the variables listed in the subscribe command are the elements of a user's mobile context that, when they change value, invalidate the information in the document. The new values of the variables also provide the information necessary for the client to determine which document to load in place of the current one.

A subscribe command is embedded in an HTML comment line <sup>2</sup> and has the following form:

```
<!-- (subscribe to variable,action variable,action ... ) -->
```

*Variable* is a standard reference to a dynamic environment variable. When the client loads the document and parses the subscribe command, it subscribes to each variable it finds in the command.

*Action* is the action the client takes when it receives a notification that the variable has changed value. It can have one of the following four values:

---

<sup>1</sup>The question marks in the query are standard HTML syntax denoting the arguments that are passed to the program invoked on the server.

<sup>2</sup>By having the *subscribe* command embedded in a comment, active documents remain backwards compatible and can be loaded by standard Web clients that do not support the features of Mobisaic.

reload Re-execute the URL that loaded the current document. If the URL is dynamic, the references to dynamic environment variables are resolved again.

load Execute a new URL and load the document in the same window.

spawn Execute a new URL and load the document in a new window.

close Close the current window.

The action also implicitly specifies how the client interprets the new value of the variable. The client ignores the new value of the changed variable for the **reload** and **close** actions, but it interprets the new value of the variable as a URL for the **load** and **spawn** actions.

For the WhereAmI active document, each page has the following subscribe command:

```
<!-- (subscribe to $(Location),reload) -->
```

The commands tells the client that, when it loads the page, it should subscribe to the **Location** variable in the user's dynamic environment, and that, when this variable changes value, it should re-execute the URL. In this case, we can tell the client to ignore the value of the variable because the dynamic URL already incorporates it.

## 5 Mobisaic in the Desktop Environment

Although Mobisaic was originally designed for use in a mobile computing environment, it can also be useful in the desktop environment. There are a number of information sources in the WWW that produce information periodically, and it is straightforward to write documents in Mobisaic that tap into these sources. Some documents that have already been written include simple cron scripts that, early in the morning, publish the URLs for the Dr. Fun and Dilbert comic pages to a list of interested users running Mobisaic. The published URLs spawn new windows showing the contents of the pages. When users come in to work in the mornings, the daily comic pages are already showing on their screens.

As another example, we use active documents together with electronic mail to implement a distributed, recommendation-based "hot list" of new and interesting pages. Our local version of Mosaic has been modified to make it easy to forward URLs to others in our department using email, so now it is common practice for people to forward onto friends URLs that they have discovered and find interesting. The email messages generated by Mosaic have a special mail tag, *X-URL*, that contains the URL being forwarded. A user's incoming mail filter finds these tags and publish the URLs in them to the user's Web client. The result is that, when users have a URL forwarded to them, a window automatically appears on their screen displaying the document referenced by the URL.

A third application uses active documents to display and update stock quotes. A background filter monitors a stock quote information source, and, in a dynamic environment for stock prices, publishes the latest values of the stocks it monitors as they change. Documents displaying the information for a given stock subscribe to the dynamic environment variable associated with the stock in the stock dynamic environment. When users view a document for a stock, the document will update itself as new stock values are published in the dynamic environment variable for stocks.

## 6 Implementation

This section discusses our implementation of Mobisaic and the changes we applied to a standard Web client to use them.

### 6.1 Dynamic Environments

We have implemented dynamic environments using a network-based publish and subscribe paradigm [Oki et al. 93]. A dynamic environment is maintained by a subscription-based server to which applications broadcast

queries and from which applications receive multicast notifications. We use the zephyr notification system [DellaFera et al. 88] to implement the publish and subscribe facilities. Zephyr allows programs to subscribe to any number of message classes and class instances to which other programs can address and send messages. Since multiple programs can subscribe to the same message class and class instance, a message to a class and instance will be multicast to all subscribing programs. A dynamic environment server is a zephyr client that listens to a specific zephyr message class for dynamic environments and a class instance that is the name of the environment it serves. Applications that interact with dynamic environments send zephyr messages with the dynamic environment message class, the name of the environment as the class instance, and the name of the dynamic environment variable as the recipient.

We chose to use zephyr because it provides network transparency, automatic subscriber-based routing, authentication, asynchronous notification, and the ability to run redundant zephyr clients supporting redundant instances of the same dynamic environment. (We presently do not take advantage of zephyr's authentication system.) Zephyr is not without drawbacks, however. For example, it operates only within relatively small administrative domains, such as a department or campus. It cannot distribute information quickly to many hosts, which can become a problem in the current system during heavy load. (This bottleneck and a possible solution to it are discussed in [Schilit and Theimer 94].) Fortunately, a C library interface hides the use of zephyr from Web clients, so changing to a new transport should be relatively easy.

## 6.2 Client Modifications

Any Web browser can be modified to support dynamic URLs and active documents provided that it supports an interface for loading and reloading documents, spawning and closing windows, and the ability to add an asynchronous input descriptor to its set of inputs. A client library handles all communication with dynamic environments, and parses dynamic environment variable references. Filters, supplied by the library, are applied to the input and output communication paths. The filter on the output stream resolves references to dynamic environment variables embedded in dynamic queries, and the filter on the input stream subscribes the client to dynamic environment variables embedded in active documents. If the client supports an internal interface for document and window manipulation, then it can be directly linked with the Mobisaic client library. For those clients that only support an external interface or do not have an interface for adding asynchronous input descriptors, we have a wrapper program that handles dynamic environments and controls the Web client as a child process.

Our prototype Mobisaic uses the X Mosaic [Andreessen and Bina 94] client linked with the Mobisaic client library running on desktop machines. The X Mosaic client has a relatively clean internal interface for loading documents and spawning windows, so most of the code changes were to add the file descriptor for the dynamic environment communication channel to the list of input descriptors, the callback to handle asynchronous input on the descriptor, and calls to the Mobisaic library filters on the input and output communication paths. We intend to eventually acquire Windows-based subnotebooks with wireless communication devices and location-sensors, so we anticipate making similar changes to WinMosaic.

## 7 Summary

This paper has described a World Wide Web information system called Mobisaic that investigates information browsing in a mobile wireless computing environment. Dynamic URLs allow a single URL to return different documents or execute different commands depending upon the values of the embedded variables at the time the URL is selected by the user. Active documents have references to dynamic environment variables to which the Web client subscribes when it loads the document. By subscribing to the variables, the client receives notifications when they change value, implicitly denoting that the information currently contained in the document has become invalid or no longer relevant to the user's mobile computing context. Minimal modifications to Web clients and the HTML syntax are required to support the features of Mobisaic. Web servers need no modifications whatsoever. A library hides the details of the communication

mechanism and provides filters for parsing and resolving dynamic environment variable references which makes it straightforward to modify a Web client to take advantage of the features of Mobisaic.

## 8 Acknowledgements

We would like to thank Marc Fiuczynski, Ed Lazowska, Hank Levy, Stefan Savage, Terri Watson, and John Zahorjan for their suggestions on Mobisaic. Bill Schilit at Xerox Parc has been incredibly helpful in discussing context-aware applications. Finally, special thanks are due to Xerox PARC for supplying us with the software and hardware that began our experiment in mobile computing.

## References

- [Andreessen and Bina 94] Marc Andreessen and Eric Bina. “NCSA Mosaic: A Global Hypermedia System” In *Internet Research*, 4(1):7–17, Spring 1994.
- [Berners-Lee et al. 92] Tim Berners-Lee, Robert Cailliau, Jean-Francois Groff, and Bernard Pollermann. “World-Wide Web: The Information Universe” In *Electronic Networking: Research, Applications, and Policy*, 2(1): 52–58, Spring 1992.
- [DellaFera et al. 88] C. Anthony DellaFera, Mark W. Eichen, Robert S. French, David C. Jedinsky, John T. Kohl, and William E. Sommerfeld. “The Zephyr Notification Service.” In *Proceedings of the USENIX 1988 Winter Conference*, Winter 1988.
- [Oki et al. 93] Brian Oki, Manfred Pfluegl, Alex Siegel, and Dale Skeen. “The Information Bus — An Architecture For Extensible Distributed Systems” In *Proceedings of the Fourteenth ACM Symposium on Operating System Principles*, December 1993.
- [Schilit and Theimer 94] Bill N. Schilit and Marvin M. Theimer. “Disseminating Active Map Information to Mobile Hosts” To appear in *IEEE Network*, September, 1994.
- [Schilit et al. 93A] Bill N. Schilit, Norman Adams, Rich Gold, Michael Tso, and Roy Want. “The ParcTab mobile computing system.” In *Proceedings of the Fourth Workshop on Workstation Operating Systems*, pp. 34–39, October 1993.
- [Schilit et al. 93B] Bill N. Schilit, Marvin M. Theimer, and Brent B. Welch. “Customizing mobile applications.” In *Proceedings of the USENIX Symposium on Mobile & Location-Independent Computing*, pp. 129–139, August 1993.