Dynamic Documents: Scalable Access to the WWW

Tom Pinckney and Joshua A. Tauber
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139, USA

{pinckney, josh}@lcs.mit.edu

1 Introduction

Information retrieval will be one of the driving applications of scalable computing. Popular information retrieval tools need to be adaptable to many environments and platforms in order to allow users to have information at their fingertips. Recently, the World-Wide Web (WWW) and NCSA Mosaic have become extremely popular information retrieval tools . As currently designed, Mosaic's fixed policies and fixed interfaces prevent it from easily incorporating new applications or running on computing platforms differing from the high-performance, uniprocessor workstations connected to high-bandwidth networks for which WWW clients are generally. For example, wireless mobile computers are quickly overwhelmed by the data included in many multi-media WWW documents.

We propose to allow documents to set their own policies by using information that a WWW client provides about the client hardware and user preferences. The single set of policies tuned to the high-performance, uniprocessor workstation-environment would then no longer constrain the use of WWW clients in other environments. For example, Mosaic currently automatically displays all HTML documents it receives in response to a query. In the environment for which Mosaic was developed this policy performs adequately. However, on another platforms–for instance, a mobile, wirelessly-networked computer–this display policy could easily overload the network and use all the limited screen space.

2 Dynamic Documents

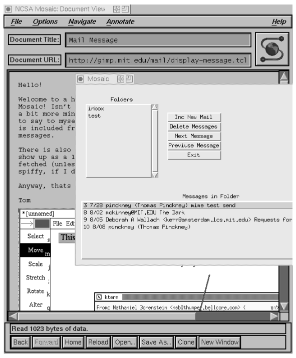
We propose *dynamic documents* to addrress the variable resource capabilities of multi-scale computing. Dynamic documents are programs executed by WWW clients in order to generate the actual information displayed to the user. Execution of a dynamic document may result in any action related to the document. Other documents may be fetched, new information may be generated, or information local to the client may be accessed. Safety is guaranteed by using an interpreter that executes only safe commands. We have implemented dynamic documents by storing documents as Tcl scripts and adding a new MIME content type. We modified the NCSA Mosaic client to run dynamic documents it retrieves through a Tcl interpreter, displaying the resulting document. In addition, our client hides the latency of wireless networks by caching and prefetching. We experienced a 40% hit rate with a global caching policy. The caching and prefetching can be made more "intelligent" by using dynamic documents to control the policies.

Documents can be customized at the client depending on available resources. For example, if the screen size is small, the script can select some core subset of information to display to the user. If the bandwidth is limited, the fetching of large multi-media objects can be delayed and placed in the background while the user views more accessible information.

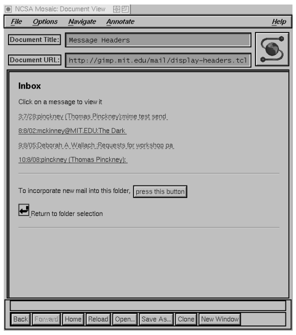
Dynamic documents function by means of callbacks. A callback is a procedure provided by the dynamic document to be invoked by the client when a particular condition is met. For example, when the client needs to decide what information to display to the user, a callback may be invoked. This callback is then responsible for producing HTML that the client displays. Our caching and prefetching additions use document supplied callbacks so that the document itself may decide when and what to prefetch and how to handle cache-coherency.

To add functionality to the client, we must define the primitives available to be invoked by dynamic documents. These primitives define the ways in which dynamic documents find out about and influence the rest of the world. While many primitives are possible, we have decided only to address the issues of flexibility in information retrieval, information display, and interface

This work was done jointly with and supervised by M. Frans Kaashoek. It was supported in part by the Advanced Research Projects Agency under contract N00014-94-1-0985, by an NSF National Young Investigator Award, and by grants from IBM and AT&T. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. government.



(a)



(b)

Figure 1. (a) With a large screen and fast network, the browser displays the messages with text and graphics, and has an elaborate user interface in a separate window. (b) The browser adapts itself to less screen space and network bandwidth.

processing. Thus, we provide functions to fetch information from HTTP servers, to query parameters of the local computer, and to perform user-interface actions, as well as functions to control the document cache.

3 An Adaptive Email Browser

To investigate the potential of dynamic documents, we have built an adaptive email browser in which email is accessed through a Mosaic client. Dynamic documents generate the browser and views of email messages dynamically. In addition, dynamic documents application specific caching and prefetching strategies. This application did not require any changes to the adaptable WWW client itself.

The adaptive email browser is organized as follows. The browser is a set of dynamic documents that is retrieved from an HHTP server when the user opens the URL for the browser. The dynamic documents then run on the client to generate the browser. The browser's interface runs entirely on the client so button clicks and typing do not necessarily result in messages to the server. Environment variables (e.g., screen size and type of network) determine which parts of email messages the browser retrieves. At the client, the browser then generates an HTML page for the message to display. This setup moves the application interface from the server to the client dynamically. As a result, we reduce the load on the server, we reduce the network traffic between the client and server, and we improve interactive performance.

The browser adapts to the screen size and to the network speed of the platform on which it runs. Depending on the screen size, it either displays a menu of URLs that point to individual mail folders and messages, or it generates a more elaborate Tk interface in a second window that is much more user friendly. In addition, messages adapt to screen size and network capabilities. For example, if the screen space is limited or the network speed slow, the browser generates links to pieces of the message rather than actually fetching and displaying the data completely.

Figures 1a and 1b illustrate how the browser adapts to different environments. Figure 1a shows the browser for a platform with a big screen and a fast network. In this configuration, the browser displays multiple windows and displays the message with graphics and text. Figure 1b shows how the browser adapts to having less screen space and network bandwidth. It has only one window with a more primitive interface. The important point about these figures is that the browser and mail message adapt by using dynamic documents and environment parameters. No changes are needed to the adaptable Mosaic client.

4 Conclusions

Dynamic document are an approach to allow WWW clients to adapt easily to different computing environments; they contain programs, which are run at the client and generate the document to be displayed. We have used application-specific information with dynamic documents to customize information display and interface processing in a scalable manner. With dynamic documents we can guarentee the correctness of a cache while increasing its hit rate and performing intellegent prefetching. We have used dynamic documents to implement an adaptable mail browser that gracefully adapts to multi-scaled computing environments.

References

[1] F. Kaashoek, T. Pinckney, and J. Tauber. Dynamic documents: Extensibility and adaptability in the WWW. In *Proceedings of the Second International WWW Conference '94: Mosaic and the Web*, Chicago, IL, October 1994.

[2] F. Kaashoek, T. Pinckney, and J. Tauber. Dynamic documents: Mobile wireless access to the WWW. In *Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, U.S., 1994.