# Mobi*Scape*: WWW Browsing under Disconnected and Semi-Connected Operation[*]

Carlos Baquero[†] Victor Fonte Francisco Moura Rui Oliveira

*Departamento de Informática / INESC*
*Universidade do Minho*
*Braga - Portugal*

**Abstract**

Although WWW browsers such as Netscape already offer some support for operation under low bandwidth connections, this is still unsuited for mobile environments. In contrast to fixed networks, here connection time is usually expensive and thus clients cannot afford long sessions.

Mobi*Scape* uses the proxy interface present on modern navigation tools. It intercepts the HTTP data flow, compresses it, and applies caching, hoarding and prefetching policies to both the mobile host and its support station. The Support Station proxy keeps a given subset of remote documents in local store and up-to-date; occasionally it will prefetch. The Mobile Host proxy will hoard according to its profile, trying to minimize the need for a connection. The system also maximizes the use of the cache by using dynamic caching policies.

No modifications were made to the existing HTTP server, proxies or browsers.

## 1 Introduction

The WWW information browsing and interchange network supported by the HTTP protocol is a multi-party distributed application with multiple servers and clients. As usual for distributed applications, a fixed communication network is assumed. On fixed networks bandwidth is high and communication costs are typically low or even free, from the user's perspective. In contrast, mobile communication links tend to have low bandwidth and high communication costs.

[*]Contact: {*cbm,vff,fsm,rco*}*@di.uminho.pt* or *http://www.di.uminho.pt/~*{*cbm,vff,fsm,rco*}.

[†]The author is supported by JNICT-PRAXIS XXI grant BD/3123/94.

A mobile user connecting from a distant location by a dial-up link cannot afford to wait for a long HTML document that is being fetched from a slow HTTP server or link. It would be desirable to prefetch that document into a host in the logging fixed network so that in a subsequent connection it can be promptly handled to the mobile host. However, current caching policies are unsuited for this purpose as they are based on the locality of references among different users and do not keep "prefetch profiles".

Another important issue concerns the support of disconnected operation. A mobile host working autonomously should be allowed to use locally cached documents. The set of hoarded documents can be established by a profile that defines the caching strategy for disconnected operation support. Typical hoardable documents are clusters of text pages such as FAQ, manuals and tutorials.

In this paper we propose a model (along with an implementation) for mobile WWW browsing. It copes with the above shortcomings in order to achieve greater efficiency both in connected or semi-connected operation, and to allow disconnected operation.

In the next section we describe the overall model of the system, followed by an analysis of the proposed caching scheme. We then briefly describe Mobi*Scape*'s working model and give some insight into the current implementation. After comparing this work with others', in section 6 we conclude with a discussion of its contributions and remaining open issues.

## 2 The Mobi*Scape* Model

The system consists of a fixed host and a mobile one, typically a *Notebook* or a *Personal Digital Assistant*. The fixed host has a permanent Internet connection and plays the role of Support Station (SS). The system relies on the SS essentially for its reliable Internet connection and storage capabilities.

Given a reasonable connection to the SS, the Mobile Host (MH) uses the SS as a front-door to the WEB. The main idea is then to provide a suitable caching scheme for WWW documents, both in the MH and in the SS, in order to minimize the periods of connection between these machines. Basically, the goal is to have a document cache at the SS side to reduce wait periods caused by fetching of remote documents and a MH cache to allow disconnected operation.

The overall system is depicted in figure 1. The HTTP data flow between the WWW browser (in the MH) and the HTTP server/proxy is intercepted twice, by the MH proxy and by the SS proxy, respectively. The system is fairly symmetric due to the same kind of processing at both ends.

Without caching the two proxies would simply forward the request/reply data

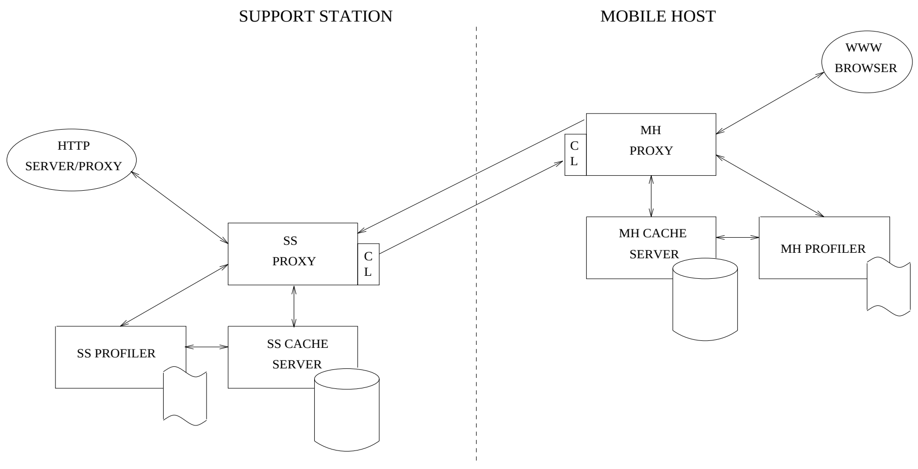

Figure 1: The Mobi*Scape* Model

flow introducing a compression/decompression layer in the reply way. This layer, depending on the MH/SS link characteristics, can be switched on or off.

Both proxies are connected to a cache server and incoming requests are forwarded only after a cache miss. The cache servers employ a *Least Recently Used* rejection policy, and serve the proxy dynamic requests as well as the requests of a stand-alone profiler. The profilers follow a user-defined script of “should-always-be-in-cache” documents.

## 3 Caching in Mobi*Scape*

Although the existing HTTP server/proxies and WWW browsers already support caching, this seems unsuited for mobile access purposes. In fact, current caching policies are (1) entirely LRU based and (2) cache buffers are shared by all users. This has the major problem of not assuring the presence in the SS cache of a user’s important document when he or she connects to the SS.

Mobi*Scape* uses two distinct policies in order to guarantee the caching needs of individual users. Firstly, it supports the user’s dynamic interaction by means of a cache that stores every newly received document not yet present in cache. Secondly, the user may specify a set of important documents that he or she would like to have permanently in cache, thus determining a profiling policy.

Profilers run in background following a script containing URL addresses and for each URL its recycling period. Because there is no way of knowing that a certain document has been changed (it is impossible to have cache invalidation mechanisms in the WWW) the user must specify, based on his or her knowledge of the relevance of the document, the appropriate periods of re-fetch for each document.

Profiling is present both in the SS and in the MH, though its use is slightly different in each machine. The SS's profiling script indicates the documents the user wants nearby while in the MH's are those the user wants always at hand. Due to the SS storage capabilities and “cheap” permanent Internet connection its profiling script is likely to be large. On the contrary, in the MH the user is confined to a (much) smaller profiling script due to storage limitations and, mainly, because hoarding and cache updating may depend on an expensive and slow link. As a result, the MH's profiling script is a subset of the SS's script.

This two-tiered policy fits especially well in a mobile environment; disconnected operation is possible through the local cache and the knowledge of the SS profiling gives the user a measure of the cost of fetching locally unavailable documents. Some of these entail downloading from the SS, while others have to be fetched from their original sites.

## 4 Working Model and Implementation

Mobi*Scape*'s working model is quite simple. Every request from the WWW browser is intercepted by the MH proxy. If the requested document is found locally it is returned to the browser. Otherwise the request is forwarded to the SS proxy where the very same procedure is followed. If the document is found in the dynamic cache its reference count is incremented whereas in a cache miss the proxy updates its dynamic cache. In the SS the profiler periodically checks for recycling times in order to update its profiling cache. In the MH, profiling can only work during periods of connectivity, where its behaviour is the same of its SS counterpart. Due to MH unreliable connections special care must be taken when updating the MH caches: an old version of a document is only expunged after the complete reception of the new version.

One of our major concerns was to avoid any modification of the existing HTTP server/proxies and WWW browsers. This implementation only requires a HTTP server with proxy interface and proxying support in the WWW browser.

The system stays quite modular. Currently the proxy, the cache server and the profiler are three independent Unix processes with the following functionality:

**Proxies** The connections between the WWW browser and the HTTP server/proxy exist for the time of a request/reply operation. Following this philosophy, the SS and MH proxies are instantiated every time a request is made and they finish after the reply. The WWW browser sends its requests to a local port serviced by the MH proxy which forwards them, if necessary, to a SS port serviced by the SS proxy, which in turn connects to the HTTP server/proxy.

**Cache Servers** The cache server manages a multi-level cache buffer. Depending on a document's tag it stores incoming documents in one of its levels. These tags identify whether the document was fetched dynamically or by profiling. Cache lookups take place at every level. The management of the cache is based on a hashing algorithm. Since a cache server has a state and may take a fairly long setup time, it cannot be part of the proxy.

**Profilers** The profilers read their configuration script and periodically request documents. Requests are done through the Mobi*Scape* proxies and are tagged as coming from a profiler. This allows their correct treatment by the cache servers. Before requesting the whole document, profilers firstly request the document's header and check it against the cached one. Depending on the match the document is fetched or skipped. Requests may bypass the cache servers using the existing HTTP tag "Pragma: no-cache".

Mobi*Scape* is written in C. It is being used by the authors in a mobile setting consisting of Linux-based notebooks and a dial-up PPP connection to the university's Ethernet. A source and binary package will soon be made publicly available through the authors home-pages.

## 5 Related Work

The use of the WWW information system under a mobile/nomadic setting has only been addressed by the research community fairly recently. The two main topics are the introduction of location dependent information and the development of hoarding and prefetching techniques. The MOBISAIC[6] and the DATAMAN[3] projects propose interesting solutions for the introduction of location dependency, but pay less attention to caching issues. Another project proposes an implementation of dynamic documents using them to deal with different bandwidths[5]. Its implementation is based on a modified NCSA web client. The caching policy is non-profiled and equivalent to the Netscape browser cache. Nevertheless this work already addresses the possibility of using the proxy interface for local caching purposes.

## 6 Discussion

Two different though related issues remain open for discussion. Firstly, it is possible to augment the functionality of profilers to allow them to do some prefetching on their own. That is, profilers could, by themselves, prefetch documents following the links they find in the user's profiled documents. This is not difficult to implement, but the point is which links should be followed? Should the profilers blindly descend the web graph? Documents such as images and sound could probably be skipped, but otherwise where should they stop?

This leads to another open issue: what would be the impact of Mobi*Scape* if users were not careful when specifying their profiles, or if fully dynamic profiling was used? Clearly, cached documents would progressively become obsolete, thereby reducing much of the value of what is now one of the best and largest distributed systems. Our opinion is that while ATM and FDDI networks do not proliferate, system administrators need to rely on users conscience or on old disk quota policies to avoid excessive local WWW mirroring.

On the other hand, the caching scheme of Mobi*Scape* can be interesting even outside a mobile setting. Having a personally tailored cache of documents can be appealing for any desktop user. We hope that the experience gained by the use of Mobi*Scape* and its evolution will help to clarify these issues.

## 7 Acknowledgments

We thank Eric Jul and the two anonymous referees for their useful comments on an early version of this paper. A special thanks to our motivating colleagues who are willing to have their private Mobi*Scape* profiling cache.

## References

[1] HTML: Hypertext Markup Language. http://info.cern.ch/hypertext/WWW/MarkUP/MarkUP.html.

[2] HTTP: A Protocol for Networked Information. http://info.cern.ch/hypertext/WWW/Protocols/HTTP/HTTP2.html.

[3] Arup Acharya, Tomasz Imielinski, and B. Badrinath. Dataman project: Towards a mosaic-like location dependent information service for mobile clients. Technical report, Department of Computer Science, Rutgers University, New Brunswick, NJ 08903, USA, 1994.

[4] Joel Bartlett. W4 - The Wireless World Wide Web. In *IEEE Workshop on Mobile Systems and Applications*. Dec 94.

[5] Franz Kaashoek, Tom Pinckney, and Joshua Tauber. Dynamic documents: Mobile wireless access to the WWW. In *IEEE Workshop on Mobile Systems and Applications*. Dec 94.

[6] Geoffrey Voelker and Brian Bershad. Mobisaic: An information system for a mobile wireless computing environmen. In *IEEE Workshop on Mobile Systems and Applications*. Dec 94.