Efficient Mobile Access to the WWW over GSM

Xavier Delord[1] , *Stéphane Perret*[2], *Andrzej Duda*[1]

[1] LSR-IMAG, Grenoble, France
[2] INRIA, France and University of California, Berkeley, USA

1 Introduction

An increasing number of users access the WWW from small portable mobile hosts connected through different network interfaces supporting mobility: wireless low bandwidth connections over long distances (GSM - 9.6 Kbit/s), wireless medium bandwidth connections over small distances (*waveLAN* - 2 Mbit/s), desk area infrared connections to stationary LANs (*NetBeamIR* - 4 Mbit/s). The connections have different distance coverage, bandwidth, latency, cost, and quality of service (error rate, jitter) and the parameters may vary over time. GSM provides global untethered connectivity thus allowing ubiquitous mobile access to the WWW: *anywhere, anytime*. However, it suffers from relatively slow bandwidth and important cost compared to its wired counterpart. Various client devices can use GSM ranging from PDAs to full-featured laptops. In order to use GSM efficiently, we need system and application support for reducing bandwidth requirements, adapting to hardware variations, and optimizing connection costs. In this paper, we propose an application support for WWW access based on a different paradigm than the previous work: we use mobile agents to delegate all time-consuming operations to the network, in particular, downloading documents and datatype specific distillation of their contents.

In the remainder of this paper, we analyze the problem of mobile access to the WWW (Section 2), we overview the MAP architecture (Section 3), we present Alycta, an application that allows efficient access to WWW from mobile hosts connected via GSM (Section 4), and outline conclusions (Section 5).

2 Mobile Access to the WWW

The current WWW access model is not suitable for mobile hosts. It is based on the client-server paradigm and suffers from many limitations. For example, it is strictly synchronous—the user must wait for the transfer of a document and access another one only when the transfer is completed. The user may experience considerable delays because of overloaded servers or limited network bandwidth. Synchronous access limits user performance if the underlying communication network does not provide sufficient bandwidth, which is the case for example of wireless links such as GSM. Moreover, many accessed documents may be not really useful to the user who realizes it only after the transfer is completed. This results in wasted bandwidth of the network and wasted user time. Finally, this mode of operation is not suitable for applications running on nomadic or mobile computers that may occasionally disconnect, because in the traditional client-server model a client application must remain active when accessing a server. In fact, the client-server model should only be used when a user is effectively interacting with a service. In all other cases, asynchronous execution is preferable.

If we want to access WWW using low-bandwidth wireless links, communication performance is crucial: downloading large volume data objects such as images or video over GSM may be unacceptably long for the user. Previous work in this domain has shown that performance can be improved by using a specialized transport protocol instead of TCP—the useful application bandwidth attains 8.7 Kbit/s [5].

Another solution to the bandwidth bottleneck is based on *distillation*: the quality of media objects is degraded according to the needs of the user before an object is downloaded to a nomadic host. Several projects such as Daedalus [3] and Odyssey [7] have investigated this approach. Distillation reduces the volume of the information to be transported over low bandwidth wireless links, however, this operation is done when the user awaits the data, so its overhead is a part of the user response time. The overall response time perceived by the user is decreased, because transferring smaller media objects takes much less time. If bandwidth is a scarce and expensive resource, which is the case of GSM, the user may want to trade connection costs against the desired content quality: we may accept a lower quality image that needs only 10 sec to download instead of an original one requiring a 100 sec download. The experience of Daedalus shows that distillation may take between 10 to 50 % of the total user response time. At the same time, distillation allows content adaptation to hardware resources, for example, simple PDA devices can only display gray scale images at low resolution, so it is of no use to download high-resolution large sized images.

Both Daedalus and Odyssey propose a communication architecture based on proxies that download requested documents and distill the content. However, when the network is overloaded, which is often the case in Europe, downloading delays become important and increase the overall user response time.

We can overcome these performance problems by changing computing paradigm, so that document downloading and distillation is done off-line. An example of such an approach is Rover that adapts the client-server model to nomadic environment. By using *queued remote procedure calls*, it provides a uniform distributed object architecture for code shipping, object caching, and asynchronous object invocation [4]. Our approach is based on a different paradigm—delegation by means of mobile agents that an application activates to execute tasks on remote nodes. Mobile agents can move from node to node, perform useful operations and report results. The paradigm is intrinsically asynchronous—an application can disconnect after agent activation and retrieve results later on. We have developed an application called Alycta that allows efficient access to WWW from mobile hosts connected via GSM. Alycta uses MAP support for executing mobile agents in Scheme on WWW servers [9, 6].

3 MAP - an Architecture for Mobile Agents

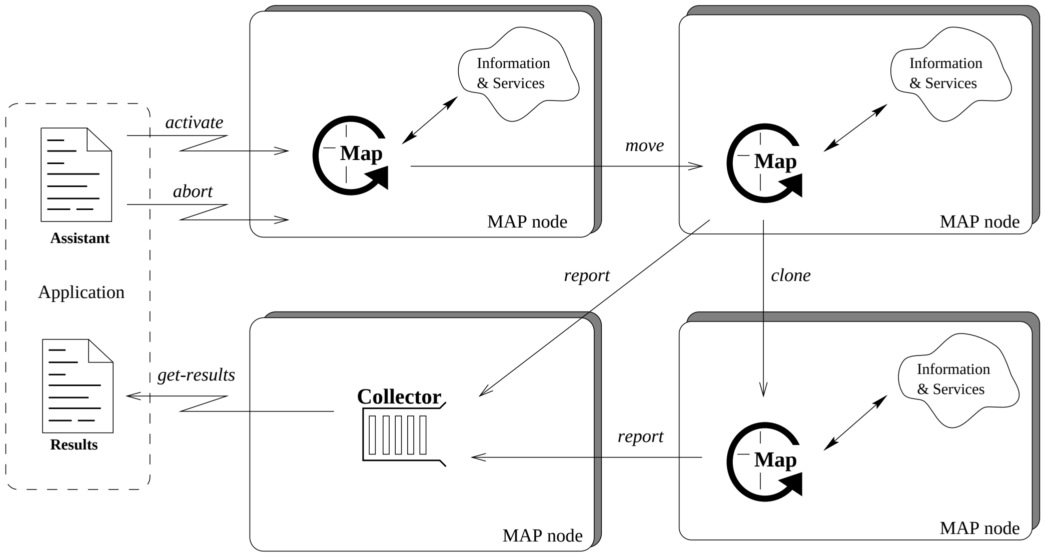

Fig. 1. Principles of MAP

The MAP architecture supports execution of mobile agents that we call *assistants*. Figure 1 presents the principles of the MAP programming model. An application *activates* an assistant on a remote node by providing source code to interpret. A *result collector*, a persistent object unique for an activation, is created on a result node. A *capability* for the collector is returned to the user for later retrieval of results. An interpreter interprets the assistant program and its execution state can be *checkpointed* to a persistent storage. In case of a node failure, assistants are *restored* from the persistent storage automatically. To accomplish its task, an assistant *moves* to a remote node, *clones* other assistants and *reports* results. The primitives execute as atomic actions, so that assistants may eventually progress despite node or communication failures. The application can retrieve results from the collector by means of a *get results* primitive that detects the termination of all assistants associated with a given activation. The primitive returns either partial or complete results depending on the termination status. This asynchronous mode of operation is advantageous because the application needs to connect to the network only for assistants activation or retrieving results. The asynchronous mode can provide important performance gains, because long processing is delegated to agents and performed off-line.

We have implemented the Mobile Assistant Programming model using the WWW framework and the Scheme programming language. [8]. MAP agents may use specialized primitives for accessing and processing HTML document [10]. We have worked on architectural aspects of using mobile agents in a nomadic environment [2].

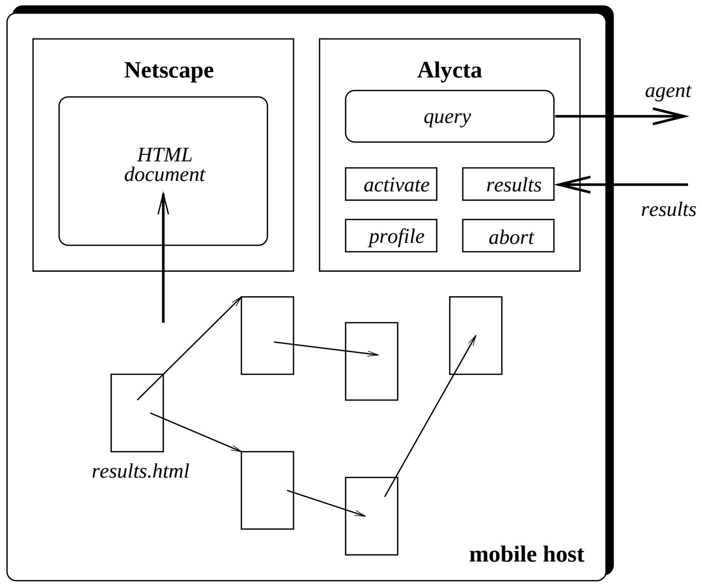

Fig. 2. Principles of Alycta

4 Alycta - Mobile Access to the WWW over GSM

With Alycta, we would like to improve user perceived performance of accessing WWW documents over GSM links. Unlike Daedalus and Odyssey, Alycta delegates downloading documents and degrading their quality to mobile agents in the network that perform the tasks off-line when the application is disconnected from the network. The user response time depends only on the speed of transporting distilled results over the wireless link.

	Original	JPEG Quality 20	JPEG Quality 5
Size (KB)	3	1.9	1.3
Reduction	1	1.6	2.3
Size (KB)	54.6	26.8	11.4
Reduction	1	2.0	4.8

Table 1. Size of images with different quality

Alycta runs on a mobile host and presents an interface in Java that allows the user to enter a query, to specify acceptable quality of large volume data types such as images, sound or video, and to control MAP agents. Figure 2 presents the principles of Alycta. When the user submits a query, the application activates mobile agents on a stationary host using a wireless GSM connection. After this, the user may disconnect. The agents evaluate the query on a search engine such as Altavista and follow hyper-text links starting from the result pages to gather all the relevant documents up to a specified deepness of links. Before being stored in the result collector, each document and all included objects are processed to degrade their quality and reduce volume. When the user connects to the stationary host, the results are transferred to the mobile host over GSM and Alycta reconstructs a local subspace of HTML documents by swizzling links. At this instant, the user can access the documents efficiently, because they are stored locally. Integration of mobile agents with content distillation allows increasing performance, because downloading and distillation are done off-line, so that the user response time is not affected by these operations.

Currently, we use a simple method of distillation - images are converted to JPEG and degraded according to the user requested quality using *ImageMagic* package [1]. The volume is reduced substantially, much more then by a loss-less compression process. Table 1 shows how much the size of images is reduced when converted with different quality (this value corresponds to JPEG quality setting: quality is from 0 (worst) to 100 (best); default is 75). The first image with different quality is presented in Figures 3-5: original (3 KB), degraded with the quality factor of 20 (1.9 KB), and degraded with the quality factor of 5 (1.3 KB). It can be seen that even in the case of the most degraded image, the text is still readable. The quality factor must be controlled by the user to specify to what extent the user wants to trade the transfer time for quality. If an image is illegible, even a short transfer time will not satisfy the user.

Fig. 3. Original image

Fig. 4. Degraded, Quality 20

Fig. 5. Degraded, Quality 5

Alycta has been implemented in Java and we have tested it on a laptop connected to the Internet via a GSM link. We have measured the performance of Alycta for some testing documents and we present in Table 2 the results of the performance measurement. We have measured the time needed for an Alycta user to obtain the results on the mobile host assuming that all results have been downloaded and degraded by mobile agents (this time includes reconstructing the document on the mobile host). The time is compared to the time needed for downloading the original document by Netscape: we have considered two cases—the document is downloaded from a local cache or from its origin site during an overcrowded afternoon. The results show that delegation of downloading and distillation to mobile agents decreases the user response time significantly.

Document	Document size	Alycta	Netscape	Distillation
Quality 5	24.9 KB	37.0 s		8.8 s
Quality 20	39.1 KB	53.0 s		9.2 s
Original, local	119.0 KB		135 s	
Original, remote	119.0 KB		285 s	

Table 2. Measured performance of Alycta

5 Conclusions

In this paper, we have proposed an application support based on mobile agents for WWW access over GSM. To experiment with this approach, we have designed and implemented Alycta, an application that uses MAP mobile agents. Running on a mobile host, Alycta can delegate time-consuming operations such as downloading documents and distillation of their contents to a node in the network.

References

1. Cristy. ImageMagick 4.0, URL: `http://www.wizards.dupont.com/cristy/imagemagick.html`. 1998.
2. A. Duda and S. Perret. Mobile agent architecture for nomadic computing. In *International Conference on Computer Communications*, Cannes, 1997.
3. A. Fox et al. Adapting to network and client variation via on-demand, dynamic distillation. In *ASPLOS-VII*, Boston, 1996.
4. A.D. Joseph et al. Rover: A toolkit for mobile information access. In *Proc. Fifteenth Symposium on Operating Systems Principles*, 1995.
5. J. Kiiskinen, et al. Data channel service for wireless telephone links. *IEEE Bulletin on Operating Systems and Application Environments*, 8(1):3–12, 1996.
6. J. Kiniry and D. Zimmerman. A hands-on look at Java mobile agents. *IEEE Internet Computing*, 1(4), 1997.
7. B. Noble et al. Agile application-aware adaptation for mobility. In *16th ACM Symposium on Operating System Principles*, St. Malo, 1997.
8. S. Perret and A. Duda. Implementation of MAP: A system for mobile assistant programming. In *Proc. IEEE International Conference on Parallel and Distributed Systems*, Tokyo, June 1996.
9. S. Perret and A. Duda. MAP: Mobile assistant programming for large scale communication networks. In *Proc. IEEE International Communications Conference 96*, Dallas, June 1996.
10. S. Perret and A. Duda. Mobile assistant programming for efficient information access on the WWW. In *Proc. Fifth International World-Wide Web Conference*, Paris, May 1996.