

NetChaser:

Agent Support for Personal Mobility

ANTONELLA DI STEFANO AND CORRADO SANTORO

University of Catania

The advent of mobile telecommunication systems has brought the term *user mobility* to next-generation telecommunication network research and design. User mobility implies two things: *terminal mobility*—the ability of the network to locate a mobile terminal—and *personal mobility*—the ability of users to access defined services from any terminal in the network, while maintaining their personal environment settings.

Many researchers are working on new telecommunication technologies to support both terminal and personal mobility; however, the increasing popularity of laptops and PDAs necessitates the introduction of user mobility to existing wide telematic environments, such as the Internet. Several research projects, such as Mobile IP,¹ are dealing with *terminal mobility*, but there are few exploring *personal mobility*. Although some browsers, like Netscape Navigator, already support a kind of personal mobility called *roaming user access*,² it is not transparent; each time the user changes working terminal, the installed browser must be reconfigured and restarted.

NETCHASER

NetChaser is a mobile-agent-based infrastructure for supporting personal mobility in accessing Internet services. Written entirely in Java, the prototype of the system is currently running within the Autonomous Remote Cooperating Agents (<http://osweb.iit.unict.it/ARCA>) framework developed at the University of Catania. NetChaser's mobile agents form a wrapper layer between the applications (Internet clients or servers) and the network and assist users by following them when they change working terminals. The system uses the Internet browser alone as the user's front end. It exploits DHTML and Java capabilities to transparently add the functionalities required for the client machine to operate with the agent-based layer.

As agents are often used to allow interoperability between existing heterogeneous systems,³ in NetChaser they cooperate with applications that are already running. In addition, the system uses agent mobility to track user movement, which allows sessions to be suspended and then resumed from another terminal. NetChaser exploits the agents' intrinsic characteristics, such as autonomy, activeness, and proactiveness,⁴ to assist users in accessing information services. By attempting to predict users' future actions, agents can help provide more effective and personalized service.

NetChaser is an agent-based infrastructure for supporting personal mobility in accessing Internet information services. It exploits agents' ability to assist users by following them when they change working terminals.

REQUIREMENTS ANALYSIS

To support personal mobility in the Internet environment, any solution must meet certain requirements:

- *Adaptability.* The new infrastructure must cooperate with existing servers without requiring modifications to them. The Internet community cannot accept any solution requiring software patches for the network daemons already running (and in wide use).
- *Transparency.* Workable solutions cannot force users to change operational modes or perform any software management activity. The infrastructure must interoperate with existing applications or, if needed, provide users with transparent access to new applications.
- *Heterogeneity.* The solution must be “seamless” rather than platform-specific.

Further, a basic requirement for a *mobile* infrastructure solution is *accessibility*. For example, let us call the user’s personal computer the *home station* and any other machine he or she utilizes the *host station*. The home station’s environment configuration must be available to the user while working at any host station.

Standard Internet services are generally accessed by client applications like Netscape Navigator and Internet Explorer. Two different approaches are possible for allowing a mobile user to access these applications from a host station:

- The application can be automatically reconfigured by the system with the user’s personal settings.
- A new application program, offering the same interface used in the home station, can be downloaded from application servers or the home station itself.

These solutions both meet our initial requirements as they work with unmodified Internet servers, involve operations that are transparent to the user, and are not platform-specific. We use both of these approaches in NetChaser.

PROPOSED APPROACH

NetChaser is composed of *wrapper layers*, added to each machine in the distributed system between the client or server and the network. The layers are composed of a set of cooperating agents, some of which are also mobile. We chose this architecture because

agents implemented using a platform-independent language (for example, Java, Perl, or Tcl) can be easily integrated into existing systems on any machine.⁴

Agents’ cooperative behavior and autonomy can be used to distribute tasks between the client and server wrapper layers, which better utilizes network resources and reduces latency. Their active and proactive behaviors let agents adapt to variable network conditions as they attempt to consistently provide users with better service. Furthermore, they can “learn” user actions to assist in accessing information services.

Our approach requires adopting suitable servers to handle working files and user profile settings: *file servers* allow users to access working files from host stations, and *user servers* act as profile repositories and perform user authentication. At least one user server is necessary for each Internet domain. In a manner similar to the way the sendmail daemon delivers e-mail messages, the NetChaser location protocol identifies the user server from the domain name.⁵ Thus, a user profile can be accessed with a URL-style naming scheme, such as `netchaser:user-name@domain-name`. In such an environment, the user profile must include at least the following information:

- user identity
- Web bookmarks
- welcome Web page
- preferred downloadable applications and addresses of the relative application servers
- e-mail settings (personal address, POP/SMTP server addresses, signature, and so on)
- file server addresses and relative account names

WORKING SCHEME

Several types of servers comprise the NetChaser system model:

- *Information servers* handle services, such as e-mail, Web, and FTP.
- *Proxy servers* bridge user workstations and information servers.
- *User servers* store user profiles.
- *File servers* store users’ working files.
- *Application servers* act as repositories for applications automatically downloaded to host stations.

Figure 1 shows the NetChaser architecture. User assistants, user profile managers, and file server managers are static management agents; HTTP, Mail, and FTP assistants are mobile service agents.

User assistants act as interfaces between the

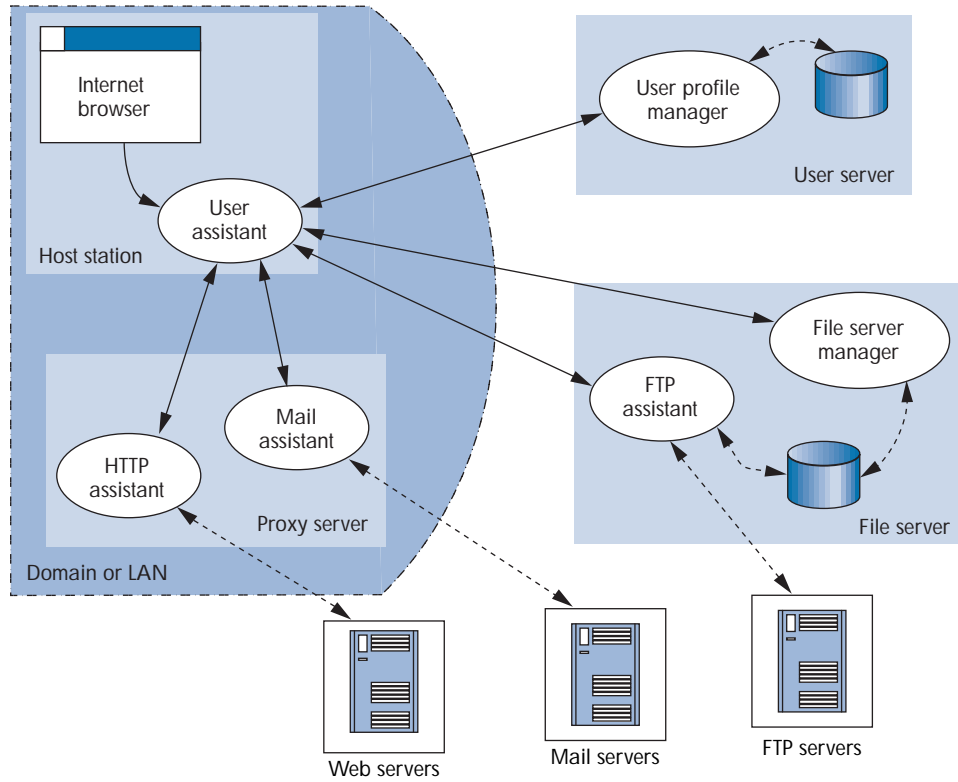


Figure 1. NetChaser architecture. The user assistant is the main interface agent and is a static management agent, as are user profile managers and file server managers. HTTP, Mail, and FTP assistants are mobile service agents.

browser and the Internet, performing login processes, managing user profile downloads and uploads, and aiding the user in accessing information services. A user assistant is always active on each host station, while the other agents are activated after the user logs in to NetChaser. When the login request is issued, the user assistant analyzes the submitted URL and executes the authentication protocol by contacting the given domain's user server. It downloads the user profile from this server, configures the user's working environment, and then NetChaser presents a welcome Web page (see Figure 2). This page contains user-related information and a control panel for various activities, such as launching applications and controlling Web browsing.

A different service assistant agent exists for each kind of service. The HTTP assistant and mail assistant both run on the proxy server nearest to the host station and maintain the working session status. Because they frequently interact with the user assistant, the links between these agents must be fast and reliable. Therefore, the proxy server these agents run on should be located in the same local network as the host station. On the other hand, to

reduce file transfer times, FTP assistants should run on the file server where the required file is to be downloaded or uploaded.

ACCESSING SERVICES

After login, the user assistant and the service-specific assistants mediate access to Internet information services. They work together to provide more personalized access to the service.

Browsing Assistant

In NetChaser, Web browsing requires a double-proxy architecture. Each HTTP request from the browser in the host station passes through the user assistant, which routes the request to the HTTP assistant, which in turn forwards it to the addressed Web server. The HTTP assistant thus helps Web browsing by maintaining control of HTTP traffic. Its autonomy allows the agent to automatically download Web pages linked to the page the user is reading and suggest which links to follow. As Lieberman shows,⁶ this approach drastically reduces network latency in navigating hyperlinks.

The HTTP assistant's primary role, however, is to

support personal mobility. It maintains all information on Web browsing session status, including a history of recently accessed URLs, cached copies of visited Web pages, and copies of cookies. When a user logs in to the NetChaser system from a new host station, the HTTP assistant moves to a proxy server closer to the new host station, taking with it Web browsing session status (see Figure 3). This allows the user to pick up a session from where he or she left off. Cookies maintain the working session on Web servers, and the history enables use of the *back* and *forward* control panel buttons. Furthermore, the page cache continues to speed access to recently viewed pages in spite of the user's change in terminals.

The NetChaser model presents two main advantages over Netscape Navigator's roaming access support. Not only is it more transparent for the user because it does not require manual reconfiguration of the host station's browser, but the HTTP assistant also provides better service, performing context-sensitive assistance to the user's browsing activity. The architecture does, however, suggest performance and scalability problems. The double message-forwarding technique could degrade performance during Web browsing, for example. However, degradation is insignificant since the user assistant runs on the same machine as the browser, and the proxy server is located near the host station. We have observed negligible differences between browsing with NetChaser and browsing directly.

With regard to scalability, two main problems could arise with resource availability on the proxy servers. First, as the number of HTTP assistants increases, the total session status size—especially the page cache size—could reach several megabytes. Our solution is to share one large page cache between all the HTTP assistants on a machine. Each of these agents maintains only an index of which pages the user accesses. When an HTTP assistant migrates to a new proxy server it brings its index, and then just the pages missing from the new shared cache are transferred as well. Another possible resource availability problem could arise from inactive HTTP assistants' consuming operating system

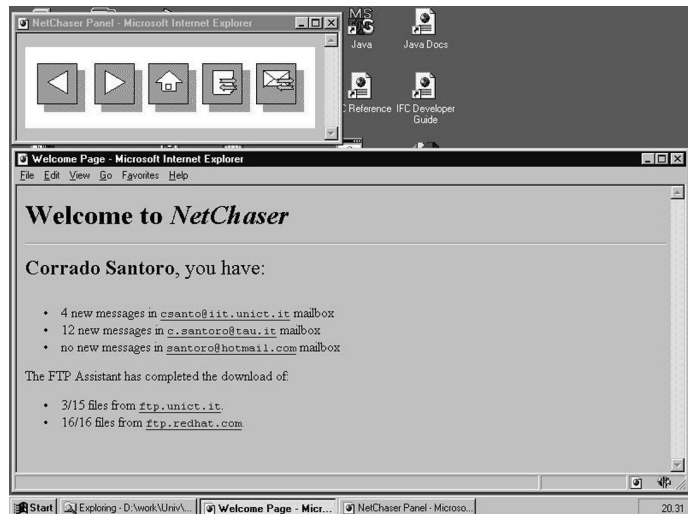


Figure 2. The NetChaser welcome page displays the number of new messages and gives a link to each one. It also indicates what files have been downloaded by the FTP assistant and lets the user access details about the downloading process.

resources, including entries in the process table, memory, and swap space. We confront this problem using a *least-active-agent removing* algorithm. When OS resources are about to run out, the agent that has been inactive longest is destroyed; only a small part of the browsing session status (that is, history, cook-

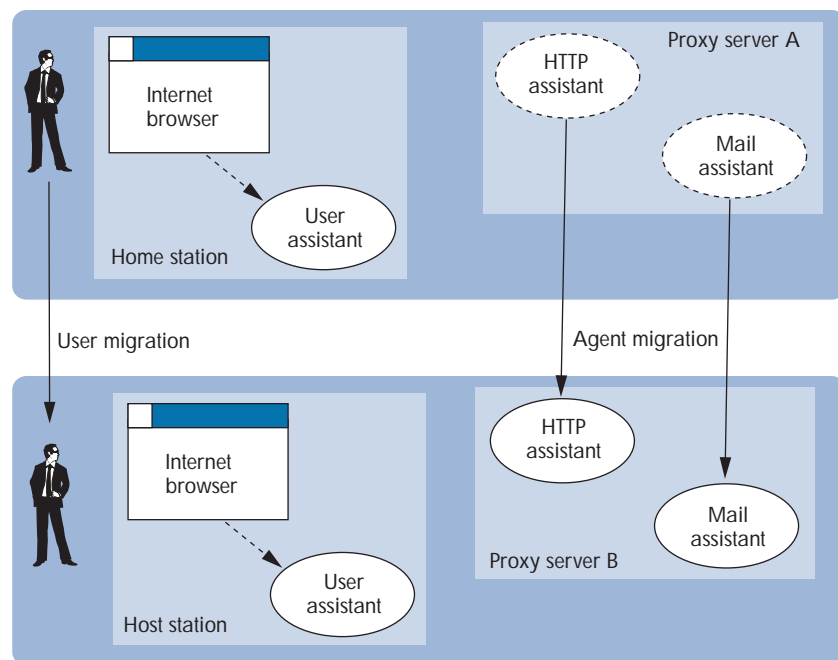


Figure 3. Migration. Assistant agents follow a user who migrates to another host station on a different subnet.

ies, and cache index) is sent to the user server, which updates the relevant user profile. When the user appears at another host station, the newly created HTTP assistant downloads the saved information from the user profile and resumes the Web session.

E-mail Assistant

E-mail client programs typically maintain one or more mailbox files on the local hard disk containing the messages sent to, or read from, the mail

Since the mail assistant knows the user profile's settings, it can manage multiple users' mailboxes.

server. Thus, the local station and the mail server have complementary information: The former maintains the downloaded messages (old and new) and the sent ones, while the latter keeps (at least) the incoming mail not yet read by the user. To support personal mobility, the NetChaser mail server is a single repository—accessible from any station in the network—for all messages.

NetChaser provides the user with an ad hoc mail reader, called NetMailer, which has a user interface similar to many existing mail readers. NetMailer is written entirely in Java and runs on any Java-enabled browser. The application is automatically downloaded and started on the host machine when the user clicks on the control panel's "e-mail" button, and provides a graphical user interface for reading, composing, and sending e-mail. It cooperates with the NetChaser mail assistant to help the user during e-mail operations. Since the mail assistant knows the user profile's mail settings, it can manage multiple users' mailboxes—retrieving both new and old messages to allow users to browse just like a traditional e-mail reader application.

Sliding download window. To improve efficiency and reduce wait times in message retrieval, we have designed the *sliding download window* algorithm. After the login phase, the mail assistant transfers a fixed number of new messages—the (configurable) window size—to a private cache. Only the message text is retrieved at first; attachments are downloaded if the user explicitly requests them. The mail assistant transfers cached messages to NetMailer,

which makes them available to the user. As the user reads the loaded messages, the window adds messages one at a time, and, each time a message is transferred to NetMailer, the mail assistant retrieves a new message from the mail server. Once again, this technique aims to predict the user's future actions by making information available even before the user issues the request.

The same algorithm is started when the user selects a new folder or when the user scrolls up the current folder. This algorithm allows an e-mail session status to be defined by the sliding window's position in each folder as well as the messages cached by the mail assistant. The user can thus resume e-mail activity after changing workstations, when the mail assistant is transferred to the new proxy server.

Context-sensitive help. The mail assistant cooperates with NetMailer to profile message relevance for the user by analyzing how the user handles each message (reading immediately; leaving unread, but not deleting; deleting after reading; deleting without reading; and so on). The mail assistant assigns each message an interest score based on the contents of the From and Subject fields. For example, if the user typically reads messages from a particular person or containing particular subject keywords before others, the mail assistant assigns a high score to these messages. When it downloads new mail, the agent lists the "most relevant" messages first. The profiling information the mail assistant generates becomes a part of the user profile, which is reused (and updated) each time the user changes host stations.

FTP Assistant

NetChaser allows mobile users to access FTP services and work with files in their storage areas on the file servers.

FTP services. To access FTP services, the user assistant contacts FTP assistants on each file server specified in the user profile. When the user issues a download request from the browser, the user assistant prompts three choices:

- Download the file to the local workstation to view it immediately.
- Save the file on one of the user file servers to view it at a later date.
- Use both of the previous options.

The FTP assistant is not involved in the first option. The user gains access to the downloaded files while

at the current host station, but not after moving to another terminal. Once the user terminates the working session, the user assistant automatically removes the files and aborts any open FTP session.

In the second case, the user chooses a file server on which to store the file, and the corresponding FTP assistant is contacted to start the downloading process, which will continue even if the user migrates. From the new host station, the user can still open the FTP panel and check download status via the progress bar. If a network failure interrupts a file transfer, the FTP assistant performs an automatic retry using incremental downloading, if available at the FTP server. Fatal download errors are reported to the user via e-mail.

In the last option, the FTP assistant performs as described in the second case. It also redirects network traffic to the user assistant, which stores the file on the host station. Automatic retries on both sides ensure reliable downloading at both the file server and the host station.

Working files. The system's file manager application, NetFiler, controls user access to working files stored on the file servers. It not only enables typical file management operations, but also allows files to be processed using host station applications, such as Microsoft Word, Excel, and so on.

All file management operations require cooperation between NetFiler, the user assistant on the host station, and a set of specialized agents, called file server managers, which reside on the file servers. When a user wishes to work locally with a remote file, NetFiler copies the file to the host station's hard disk. To maintain consistency between remote and local copies, the user assistant periodically checks for modifications to the file and, if necessary, sends the updated information to the file server manager. To this end, we are evaluating two different approaches:

- using a logging/replay mechanism of the operations made on the files,^{7,8} and
- checksumming the file by blocks and updating only the blocks modified.

These operations are made transparently; the user will always access a consistent version of the file from any host machine. To preserve the file's integrity, the remote updating of the file server is atomic: Modifications become persistent only when all the updated information is sent to the file server manager.

CONCLUSION

With NetChaser we want to help introduce personal mobility to wide environments. Through its use of software agents that can cooperate with existing applications, NetChaser is particularly suitable for integration into existing distributed systems. This system further demonstrates how agent technology is not only applicable as a middle layer for interoperability, but also as an infrastructure for providing personalized assistance to users. Our future work will deal with the intelligence of the involved agents. We will explore developing inference engines to enable the agents to adapt to the variable conditions of distributed environments, such as network and systems performances or users' needs. ■

REFERENCES

1. C.E. Perkins, "Mobile Networking Through Mobile IP," *IEEE Internet Computing*, Vol. 2, No. 1, Jan.-Feb. 1998, pp. 58-69; also available online at <http://computer.org/internet/v2n1/perkins.htm>.
2. "Manually Implementing Roaming Access," Netscape technical article available online at http://help.netscape.com/products/client/communicator/manual_roaming2.html.
3. N.R. Jennings and M.J. Wooldridge, *Agent Technology: Foundation, Applications, and Markets*, Springer-Verlag, Heidelberg, Germany, Feb. 1998.
4. M.J. Wooldridge, *Multiagent Systems*, G. Weiss, ed., MIT Press, Cambridge, Mass., Apr. 1999.
5. D.E. Comer, *Internetworking with TCP/IP*, Prentice Hall, Englewood Cliffs, N.J., 1991.
6. H. Lieberman, "Letizia: An Agent that Assists Web Browsing," *Proc. Int'l Joint Conf. on Artificial Intelligence*, ACM SIGMOD, New York, Aug. 1995, pp. 924-929.
7. A.D. Joseph, A.J. Tauber, and M.F. Kaashoek, "Mobile Computing with the Rover Toolkit," *IEEE Trans. Computers*, vol. 46, no. 3, Mar. 1997.
8. L. Satyanarayanan and K.K. Kistler, "Disconnected Operation in the Coda File System," *ACM Trans. Computer Systems*, Vol. 10, No. 1, Feb. 1992, pp. 3-25.

Antonella Di Stefano is associate professor of operating systems at the engineering faculty of the University of Catania, Italy. Her research interests are in distributed systems, and presently focus on process control networks, distributed real-time systems, and mobile agents.

Corrado Santoro is a PhD student in computer science engineering at the University of Catania. He is currently a consultant, and his research interests include mobile agents, distributed operating systems, and distributed process control.

Readers can contact the authors at {adistefa, csanto}@iit.unict.it.