Designing agents for archie and ftp sessions in Obliq

Cristian Ionitoiu

cristi@cs.utt.ro, “Politehnica” University of Timisoara, Computer Science Department,
2 Vasile Parvan, 1900 Timisoara, Romania.

This extended abstract presents a solution based on mobile agents for archie/ftp sessions on Internet. Due to the increased rate of interactivity, which imposes relatively long connection periods, required by these services, their use is not appropriate for mobile stations. This solution provides mobile user access to these services while keeping a low rate of the connections.

1 Introduction

The agent concept was first conceived in Artificial Intelligence world [92]. The aim of this concept is to solve communication problems between distributed information repositories with similar or different representations of knowledge. Thus, agents are designed both to carry and to exchange information. Recently, researchers from Distributed Systems field have borrowed the agent concept [37] from AI, emphasizing the aspects related to mobility and interaction. Agent-oriented programming is a new emerging paradigm thought as an alternative to client-server programming for certain distributed applications. Agent programming can support all the features of client-server computation, but due to agent asynchronous behavior it is a good tool for disconnected operations for mobile computers. In the case of mobile computers where the system resources are scarce, agents represent a good choice when a long dialogue between the mobile host and the fixed network is needed. The classical solutions, transactions or protocols [68], impose an active connection during the dialogue or repeated connections and disconnections. As an alternative the user working on the mobile host can provide his part of dialogue, then create an agent and use the connection only for the time needed to send the agent. The agent will roam within the static network, will reach the dialogue partner, will execute on it and will eventually come back with the result.

1.1 The archie and ftp services

Such type of scenario is an invoking sessions of `archie` [93] and `ftp` [73] Internet services. First, the user has to invoke an `archie` server for look-up after a certain software package and then he/she must access one of the `ftp` servers provided by the `archie` server to download the software. If the consulted `archie` server does not return an `ftp` server, the user has to invoke another `archie` server. Thus, the user has to know how to use both programs and to interpret correctly `archie`'s output to find the best ftp server which contains the desired software.

The above described scenario is a typical situation that can be solved using mobile agents, which can roam within the fixed network visiting, alternatively, `archie` and `ftp` servers. Moreover, this solutions can improve Internet traffic by imposing some optimization on the agent behaviour, like accessing the nearest `ftp` server and downloading software from different sites using the same agent. The agent approach introduces a uniform view for both services. The operation of this pair of services becomes completely transparent to the user. The user would provide the name of the desired piece of software (and some searching constraints) and after a while he or she will be told by the application that the software has just been downloaded.

It is claimed that all `archie` servers provide the same list of `ftp` sites for a certain piece of software. A test has been made, searching for sites containing the popular `xfig` distribution package. A snapshot of the results provided is given below. From the table one can infer that the previous claim is not true. In fact, an `archie` server provides, usually, `ftp` servers from its neighbourhood. Only a small percentage are remote `ftp` servers. But, even for the same neighbourhood, there is a certain percentage of different `ftp` sites.

site	distinct hosts	entries
archie.belnet.be	5	24
archie.luth.se	25	100
archie.bunyip.com	12	12
archie.ac.il	13	25
archie.wide.ac.jp	22	93

The paper is organised in four parts. After the introduction, a discussion of the main design requirements is given in section 2, followed by a description of the application architecture and of the abstract machine structure, which forms section 3. Section 4 presents conclusions and future work.

2 Design requirements

The design of the run-time system must consider two important aspects, security and portability of the agent execution. The security aspect depends on the type of the application, and because archie and ftp are public services, they have been designed to comply to certain type of operations. Hence an important part of the security is ensured by the services themselves. I decided to design the agent roaming and execution based on an abstract machine. The abstract machine ensures both security because the agent can execute only the instructions of the machine and portability, because the instructions can be implemented in any language. I have chosen the Obliq [14] language for implementation of agents.To avoid forever-roaming agents, the design must provide tools which would help the user to impose constraints on the agent behaviour.

3 Application architecture and the abstract machine

To migrate objects (agents), in Obliq, one needs to be provided specific remote engines, together with the `netobjd` name server, on every site that will eventually be visited by an agent. The application scenario is simple. The user, working on the mobile host, interacts with an interface program. He introduces certain data that both defines his goal and imposes constraints on agent roaming. Then, the interface program generates the abstract machine code and, in the end, an agent. In the end, the user receives either the same agent containing a result or an execution report. The interface program request the user to specify the following items: (1) the name of the wanted software package, (2) the maximal searching and downloading time, (3) the number of `archie` and `ftp` servers allowed to be visited, (4) a specific `archie` server imposed by the user.

The structure of the autonomous agents encompasses the structure of the abstract machine and a small memory. The memory is used to keep intermediate best results during migration from one site to another. The abstract machine has the following state: (a) four dedicated registers hold the user provided information, in the same order given above, (b) `stack`, is the machine's stack, (c) `mem`, is a local memory for holding intermediate results during migration, (d) `AX`, is a set of six general-purpose registers for numbers, (e) `BH`, is set of six general-purpose register for strings. The machine is an object with the above state, a code area and methods for each instruction.

```
let TMachine = {
    state => clone(TState),
    code => [],
    ... (* method definitions *)};
```

The machine has been endowed with several types of instructions: initialisation, stack management, general-purpose, flow control, `archie`/`ftp` server interaction, migration control. In case of server interaction a specific instruction is introduced for each type of server. The migration can be done using addresses from either from registers (`migrater`) or from memory. The implementation of the first migration instructions is given below (the `hop()` function is given in [13]).

```
migrater=> meth(self, reg)
 let sv = net_importEngine("ArchieSearch",self.state.BX[reg]);
 hop(sv,agent,self);
end;
```

4 Conclusions

The paper has presented a solution for adapting interactive archie/ftp sessions for users working on mobile computers. The solution uses agents implemented in Obliq. The present implementation contains most of the concepts present above, less the time management and the advanced user specifications of agent behaviour.