Forwarders vs. centralized server: an evaluation of two approaches for locating mobile agents

Sara Alouf, Fabrice Huet, Philippe Nain

Forwarders vs. Centralized Server: An Evaluation of Two Approaches for Locating Mobile Agents

Sara Alouf,[1] Fabrice Huet,[1,2] and Philippe Nain[1]
[1] INRIA Sophia Antipolis, France [2] CNRS, I3S and UNS

1 Introduction

The Internet has allowed the creation of huge amounts of data located on many sites. Performing complex operations on some data requires that the data be transferred first to the machine on which the operations are to be executed, which may require a non-negligible amount of bandwidth and may seriously limit performance if it is the bottleneck. However, instead of moving the data to the code, it is possible to move the code to the data, and perform all the operations locally. This simple idea has led to a new paradigm called *code-mobility*: a mobile object – sometimes called an agent – is given a list of destinations and a series of operations to perform on each one of them. The agent will visit all of the destinations, perform the requested operations and possibly pass the result on to another object. Any mobility mechanism must first provide a way to migrate code from one host to another. It must also ensure that any communication following a migration will not be impaired by it, namely that two objects should still be able to communicate even if one of them has migrated. Such a mechanism is referred to as a *location* mechanism since it often relies on the knowledge of the location of the objects to ensure communications. Two location mechanisms are widely used: the first one uses a centralized server whereas the second one relies on special objects called *forwarders*.

This paper evaluates and compares the performance of an existing implementation of these approaches in terms of cost of communication in presence of migration. Based on a Markov chain analysis, we will construct and solve two mathematical models, one for each mechanism and will use them to evaluate the cost of location. For the purpose of validation, we have developed for each mechanism a benchmark that uses *ProActive* [2], a Java library that provides all the necessary primitives for code mobility. Experiments conducted on a LAN and on a MAN have validated both models and have shown that the location server always performs better than the forwarders. Using our analytical models we will nevertheless identify situations where the opposite conclusion holds. However, under most operational conditions location servers will perform better than forwarders.

2 Forwarders

On leaving a host, a process leaves a special reference, called a *forwarding reference* which points toward its next location. As the system runs, *chains of forwarders* are built. A consequence of this mechanism is that a caller does usually not know the location of the callee. A special built-in mechanism called short-cutting allows the update of the address as soon as a communication takes place. When a forwarded message reaches a mobile object, the latter sends its new location to the caller. Thus, any forthcoming request will not go through the existing forwarders. To keep the same semantic as with a static program (i.e. with no mobile object) communications through a chain of forwarders should be synchronous, i.e. the caller stays blocked during the communication time. We can now describe the protocol in use:

- A migrating agent leaves a forwarder on the current site;
- This forwarder is linked to the agent on the remote host;
- No communication can occur if the agent is moving;
- When receiving a message, the forwarder sends it to the next hop (possibly the agent);
- Any successful communication places the agent one hop away of the caller.

3 Centralized Server

An alternative to using forwarders for locating a mobile object is to use a *location server* which keeps track of the location of mobile objects in a database. We will assume that sources and agents know the location of this server. Each time an agent migrates, it informs the server of its new location. Whenever the source wants to reach the agent, it sends a message to the last known location of the agent; if this communication fails, then the source sends a `location request` to the server. We can now describe the protocol used by sources and agents to communicate with the server:

- The Mobile Agent: **Step 1:** Migrates; **Step 2:** Sends its new location to the server.
- The Source: **Step 1:** Issues a message to the agent with the recorded location. Upon failure goes to Step 2; **Step 2:** Queries the server to have the current location of the agent; **Step 3:** Re-issues the message to the agent with the location given by the server. Upon failure, goes to Step 2.

At the server, `update location requests` have priority over `location requests`. In addition, before sending the position of an object to a source, the server checks whether its queue contains an `update location request` coming from this object; if yes, then the `location request` is sent back to the queue (effective if the service policy is non-preemptive). If the queue contains several requests sent by the same agent then the newest is served and the others are destroyed.

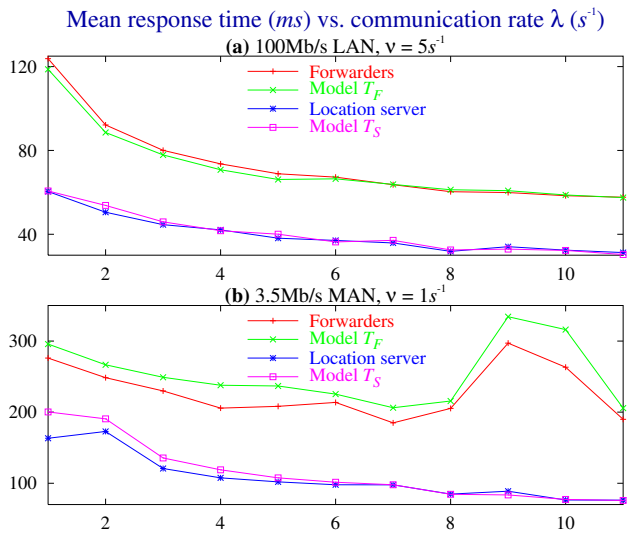

Figure 1: Validation with experiments.

Table 1: Percentiles of the relative error

		25%	50%	75%	90%	95%
Forwarding mechanism	LAN	1.2	3.9	8.5	13.2	14.2
	MAN	5.4	9.3	14.3	20.7	23.0
Centralized approach	LAN	1.4	3.4	10.2	13.8	15.5
	MAN	5.5	11.4	19.0	26.3	30.3

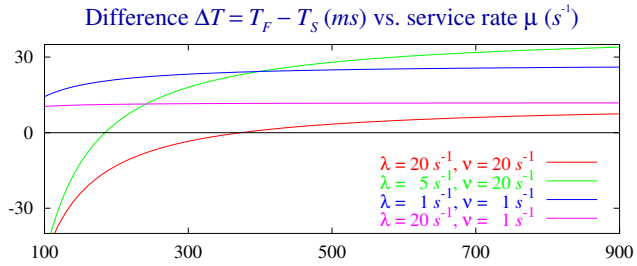

Figure 2: Difference between response times ΔT.

4 A Markovian analysis

For each mechanism, we have developped a model based on a Markov chain analysis [1] taking all processes into account, mainly, idle times for a source and for an agent, migration durations, traveling times of messages in the network and service times (centralized approach only). All these processes were assumed to be exponential random variables and independent from each other. As a result, we derived the expected time needed by the source to contact the agent, referred to as the communication time and noted by T_F (resp. T_S) for the forwarding (resp. centralized) mechanism.

5 Validation via experiments

In order to validate the theory developed in Section 4, we have conducted extensive experiments on a LAN and a MAN. The testbed was composed of a single mobile object and of a single source. Idle times for the source are exponentially distributed with rate λ (i.e. λ is the communication rate when the source is idling) and inter-migration times of the agent are exponentially distributed with rate ν. Parameters λ and ν can be modified from one session to another. All the other model parameters are system-dependent and cannot be changed. All benchmarks were written using *ProActive* [2]. The network was composed of five machines: Pentium II and Pentium III machines running Linux (2.2.18) and Sun SPARC running SunOS interconnected with a 100Mb/s switched LAN or a 3.5Mb/s MAN.

Figure 1 reports the experimental and theoretical communication times obtained for a LAN (upper graph) and for a MAN (lower graph). The graphs display the communication time as a function of the communication rate λ for the migration rate $\nu = 5$ (LAN case) and $\nu = 1$ (MAN case). The server performs better than the forwarders since it achieves a smaller communication time.

For each network configuration (LAN or MAN), the performance of the models over all experiments ($\lambda = 1 \dots 11 s^{-1}$ and $\nu = 1 \dots 6 s^{-1}$) are collected in Table 1 as percentiles of the relative error between experimental values and analytical results. Results in Table 1 indicate that both theoretical models behave fairly well especially when the underlying network is a LAN, which validates both models. Furthermore, since in the experiments most of the assumptions under which the models where constructed are not verified, we can say that the models are robust against real conditions.

6 Comparing both approaches

As mentioned in Section 5, many parameters are network-dependent so that an experimental comparison of both approaches is necessarily limited to a few scenarios. No such limitations occur when comparing both approaches by using the theoretical results obtained in Section 4. We will focus on the expected response time given by each approach and, more precisely, on their difference $\Delta T = T_F - T_S$.

Figure 2 displays ΔT as a function of the server processing speed for different values of λ and ν. The aim of this comparison is to study the behavior of our system when the server is on a slow machine or under a heavy load (typically the case of several sources and several agents). For a low migration rate ($\nu = 1 s^{-1}$) the server always performs better. However, when the mobile object has a high migration rate ($\nu = 20$) a slow/loaded server will act as a bottleneck. In this case, the forwarders yield the best performance.

References

[1] S. Alouf, F. Huet, and P. Nain. Forwarders vs. centralized server: An evaluation of two approaches for locating mobile agents. *Performance Evaluation, (Proc. of Performance '02, Rome, Italy)*, 49(1-4):299–319, Sept. 2002.

[2] ProActive. INRIA, 1999. http://www-sop.inria.fr/oasis/ProActive.