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

Sara Alouf
salouf@sophia.inria.fr
INRIA

Fabrice Huet
fhuet@sophia.inria.fr
INRIA - CNRS - I3S - UNSA

Philippe Nain
nain@sophia.inria.fr
INRIA

2004 Route des Lucioles, B.P. 93, 06902 Sophia Antipolis Cedex, France

Abstract

This paper evaluates and compares the performance of two approaches for locating an agent (e.g. a code) in a mobile agent environment. The first approach dynamically creates a chain of forwarders to locate a moving agent whereas the second one relies on a centralized server to perform this task. Based on a Markov chain analysis, we compute the performance of each scheme (time to reach an agent, number of forwarders) and compare them first with simulations and second with experimental results obtained by using *ProActive*, a Java library. Depending on the system parameters we identify the best scheme and observe that within a LAN the server yields the best performance whereas the forwarders yield the best performance within a MAN.

Keywords: Mobile code, migration, centralized server, forwarders, Markov chain.

1 Introduction

The Internet has allowed the creation of huge amounts of data located on many different machines. Performing complex operations on some data requires that the data be transferred first to the machine on which the operations are to be executed. This transfer 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* [27]. In this paradigm, 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 mobile object will visit all of the destinations, perform the requested operations and possibly pass the result on to another object. Any machine willing to receive an agent must provide an agent-platform which is a placeholder where the agent is executed.

Code mobility has recently received a lot of attention because of its wide application to fields ranging from e-commerce (e.g. looking for the lowest fare on many different sites) to data mining [7]. Mobility can be implemented as a service provided by an operating system [22]; however this severely limits its usefulness in a heterogeneous environment such as the Internet. Another solution is to use a library which provides an application with all of the necessary features [1, 3, 12, 15, 19, 24, 28].

Any mobility mechanism must first provide a way to migrate code from one host to another. It must also ensure that any communication posterior to 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 *routing* mechanism or even as a *location* mechanism since it often relies on the knowledge of the location of the objects to ensure communications. Location problems are not exclusive to code mobility. They can be encountered under different forms in many different networking areas, where the object to be located can either be fixed [20] or mobile as in wireless [25] and ad-hoc networks [5, 9, 13], or more recently in peer-to-peer networking that can be seen as a communal sharing of computer resources without a centralized authority [2, 8, 17, 26].

In the more specific setting of code mobility, two location mechanisms are widely used: the first one uses a centralized (location) server which keeps track of the location of mobile objects, whereas the second one relies on special objects – called *forwarders* – whose role is to forward a message to the mobile object. A more careful description of these approaches will be given later on.

Mobility raises several concerns, among them security [6, 14] (of both the mobile agent and the host sheltering it) and performance issues [4, 10]. In [10] the complexity of using forwarding addresses is extensively studied whereas [4] addresses fault-tolerance properties in forwarder-based mechanisms. In this paper we will only focus on performance issues and, more specifically, on the cost of communication in presence of migration. To the best of our knowledge, this is the first time that such an analysis is performed. In [19] the authors only give intuitive criteria on how to select the proper location scheme under certain circumstances. Our analysis can be used to select the best scheme based on its response time. It also allows us to compute the average number of forwarders, which is useful to study the fault-tolerance of forwarding schemes [4].

In this work we develop Markovian models of the forwarders and of the location server as implemented in *ProActive* [24], a Java library that provides all the necessary primitives for code mobility. Closed-form expressions for various performance measures are derived, including the time needed to reach an agent and the mean number of forwarders. These expressions are in turn used to evaluate the cost of location under various network conditions and for different applications. For the purpose of validation, we have developed for each mechanism both an event-driven simulator and a benchmark that uses *ProActive*. Simulations and experiments conducted on a LAN and on a MAN have validated both models and have shown that no scheme performs uniformly better than the other, thereby justifying the present research.

The paper is organized as follows: preliminary definitions and notation are introduced in Section 2; the forwarding mechanism is presented and evaluated in Section 3 and the centralized server mechanism is investigated in Section 4. Simulated and experimental results are reported in Section 5 as well as a theoretical comparison between both approaches. Concluding remarks are given in Section 6.

2 Definitions and Notation

In this section we introduce several random variables (rvs) that we will use to construct our models. Throughout the paper a mobile object will indifferently be called a mobile agent or simply an object or an agent.

The i-th message is sent by the source to the agent at time a_i and the communication is over at time $d_i := a_i + \tau_i$. The rv τ_i – referred to as the (i-th) *communication time* – is a scheme-dependent quantity that will be defined later on for each mechanism (forwarders and centralized server). In the time-interval (d_i, a_{i+1}) no message is generated by the source. Let $w_{i+1} := a_{i+1} - d_i$ be the length of this time-interval and assume that $w_1 := a_1$ and that no message is generated in $[0, a_1)$.

The j-th migration of the mobile occurs at time $m_j > 0$ and it requires the mobile p_j units of times to reach its new location. During a migration period the agent is unreachable. The mobile then spends u_{j+1} units of time at its jth location, time during which it can be reached by a message, and then initiates a new migration. We set $u_1 := m_1$ and assume that the mobile does not migrate in $[0, m_1)$ (see Figure 1).

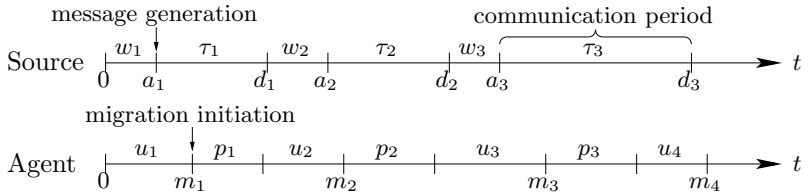

Figure 1: A time diagram including all rvs relative to the source and to the agent

The following assumptions will be enforced throughout the paper:

A1 The *input* sequences $\{w_i, i \geq 1\}$, $\{p_j, j \geq 1\}$ and $\{u_k, k \geq 1\}$ are assumed to be mutually independent *renewal* sequences of rvs such that w_i, p_j and u_k are exponentially distributed rvs with parameters $\lambda > 0$, $\delta > 0$ and $\nu > 0$, respectively.

3 The forwarders

3.1 Description

Forwarding techniques were first introduced in distributed operating systems like DEMOS/MP [23] for finding mobile processes. The mechanism is straightforward: on leaving a host/machine, 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 not usually 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 communicates its new location to the caller. As a result, all subsequent requests issued by this caller will not go through the existing forwarders – which are shortcut.

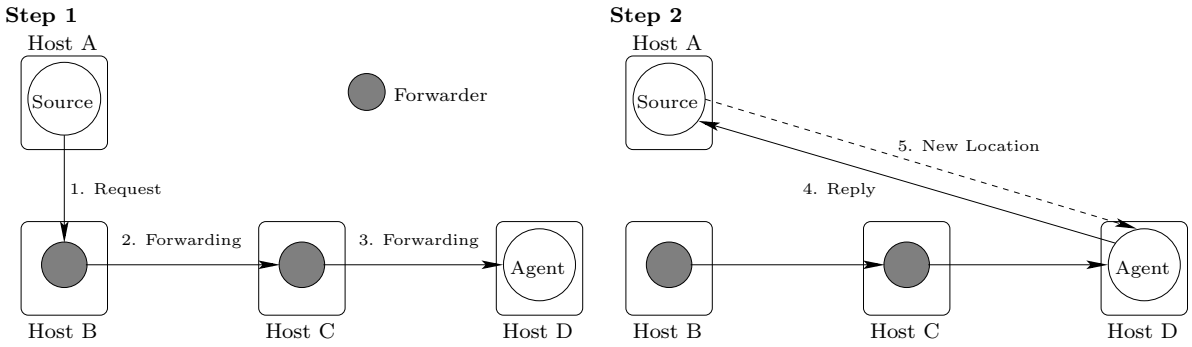

Figure 2: The short-cutting feature in the forwarding mechanism.

An illustration of the short-cutting feature is given in Figure 2: a message is sent by the source to the last known location of the agent (Host B). Since the agent is no longer at this location, the message is then forwarded to the host that was next visited by the agent (Host C). Again, the agent has already moved when the message reaches Host C and the message is forwarded to the next visited host (Host D) where the agent is finally located. A location message is then sent by the agent (located at host D) to the source and the next message will be sent by the source to Host D.

In order to keep the same semantic as with a static program (i.e. with no mobile object) one has to introduce constraints. Mainly, communications through a chain of forwarders should be synchronous, i.e. the caller stays blocked during the communication time. With these assumptions we can now describe the protocol in use:

- Upon migration, a mobile object leaves a forwarder on the current site;
- This forwarder is linked to the mobile object on the remote host;
- No communication can occur when the object is migrating;
- When receiving a message, the forwarder sends it to the next hop (possibly the mobile object);
- Any successful communication places the mobile object one hop away of the caller (e.g. after Step 2 in Figure 2 the agent is located one hop away from the source).

The above protocol is implemented in various Java libraries (*MOA* [19], *ProActive* [24] and *Voyager* [28]) except for the short-cutting feature that is triggered by the agent at the end of the message processing.

3.2 A Markovian analysis of the forwarders

In this section we will evaluate the performance (number of forwarders in Section 3.2.1 and response time 3.2.2) of the mechanism introduced in Section 3.1 through a Markovian analysis. We will assume that a mobile object does not return to a previously visited site where there is still an active forwarder, i.e. it does not migrate twice (or more) to a particular site between two consecutive epochs d_i (see Section 2). Hence, there can be no loops within a chain. It is clear that under these conditions the length of a chain can extend to infinity.

A forwarding mechanism is well represented by the chains of forwarders that it produces. In an application a chain of forwarders connects a single source to a single agent and its dynamic is not affected by other objects; there will be as many chains as there are pairs source-agent. It suffices then to study the behavior of one chain to evaluate the performance of the forwarders approach. To study the dynamics of a chain, one should take into account the state of a chain and the states of the source and the agent at its endpoints. Notice that this doesn't place any assumption on the number of objects (source or agent) in the application.

From the description given in Section 3.1, it can be seen that at any time the system is in one of the following states (see Figure 3):

- States $(i, 0, 0)$, $i \geq 1$, indicate that the agent is available (i.e. not migrating) and located i hops away from the source, and that no message is traveling;
- State $(1, 0, 1^*)$ indicates that the agent is available and located one hop away from a message, and the latter has never been through any forwarder;
- State $(1, 0, 1)$ indicates that the agent is available and located one hop away from a message, the latter having already gone through at least one forwarder;
- States $(i, 0, 1)$, $(i \geq 2)$ indicate that a message is traveling, the agent is available and located i hops away from the message;
- States $(i, 1, k)$, $i \geq 1$, indicate that the agent is migrating and that before the initiation of its migration it was located i hops away from the source if no message is traveling $(k = 0)$ or it was located i hops away from the message if a message is traveling $(k = 1)$;
- State $(0, 1, 1)$ indicates that a message that has gone through at least one forwarder and the agent are at the same location but that the agent is migrating (i.e. the agent has initiated a migration just before the arrival of the message);
- State $(0, 0, 1)$ indicates that the message has reached the agent after having traveled through at least one forwarder and that the agent is currently communicating its new position to the source.

State $(1, 0, 1^*)$ has been introduced to take into account the fact that if a new message reaches the agent after exactly one hop and that the agent has not initiated a migration before the arrival of the message then the cycle is over and the source can transmit a new message; otherwise, if a message reaches the agent after having gone through at least one forwarder then the agent will have to communicate its location to the source once the message will have reached it.

Under the enforced assumption

A2 The traveling time of a message from one host (possibly the source) to the next one (possibly the agent) is an exponential rv with parameter $\gamma > 0$. The successive traveling times are assumed to be mutually independent and independent of the input sequences $\{w_i\}_i$, $\{p_i\}_i$ and $\{u_i\}_i$ introduced in Section 2,

and assumption **A1**[1], it is easily seen that the sojourn time in each state is exponentially distributed and that any state can be reached from any other state in a finite number of steps. In other words, the process defined above is an irreducible Markov process on the state-space $\mathcal{E} := \{(0, 0, 1), (0, 1, 1), (1, 0, 1^*), (i, j, k), i \geq 1, j, k = 0, 1\}$.

The transition rates are indicated in Figure 3: from state $(1, 0, 0)$ the process may jump to state $(1, 0, 1^*)$ with rate λ (new message generated) or to state $(1, 1, 0)$ with rate ν (migration of the agent); from state $(1, 0, 1^*)$ the process may move to state $(1, 1, 1)$ with rate ν (migration) or to state $(1, 0, 0)$ with rate γ (message has reached the agent, cycle is over); from state $(i, 0, 0)$ with $i \geq 2$ the process may jump to state $(i, 1, 0)$ with rate ν (migration) or to state $(i, 0, 1)$ with rate λ (new message generated); from state $(i, 1, 1)$ with $i \geq 1$ the process can move to state $(i + 1, 0, 1)$ with rate δ (end of migration) or to state $(i - 1, 1, 1)$ with rate γ (message has reached next host on its route); from state $(i, 1, 0)$ with $i \geq 1$ the process can jump to state $(i + 1, 0, 0)$ with rate δ (end of migration period) or to state $(i + 1, 0, 1)$ with rate λ (new message generated); from state $(i, 0, 1)$ with $i \geq 1$ the process can move to

[1]Assumptions **A1** and **A2** are mainly made for sake of mathematical tractability. We have however observed in our experiments that our models are fairly robust to deviations from these assumptions – see Section 5.

state $(i-1,0,1)$ with rate γ (message has reached next host) or to state $(i,1,1)$ with rate ν (migration); from state $(0,1,1)$ the process may only move to state $(1,0,1)$ with rate δ (end of migration); finally, from state $(0,0,1)$ the process may only move to state $(1,0,0)$ with rate γ (location message sent by agent has reached the source; cycle over).

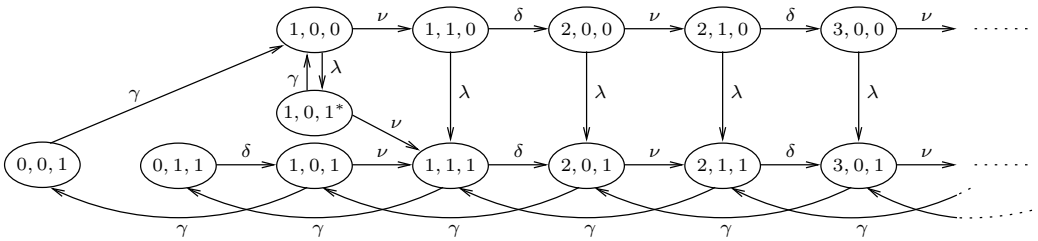

Figure 3: System states and transition rates in the forwarding mechanism.

Let $p_{i,j,k}$ be the stationary probability that the process is in state $(i,j,k) \in \mathcal{E}$. If the Markov process is ergodic then the stationary probabilities $\{p_{i,j,k}, (i,j,k) \in \mathcal{E}\}$ are the unique strictly positive solution of the Chapman-Kolmogorov (C-K) equations [16]

$$
\begin{aligned}
(\lambda+\nu)\, p_{1,0,0} &= \gamma\,(p_{0,0,1} + p_{1,0,1^*}) && && (1)\\
(\lambda+\nu)\, p_{i,0,0} &= \delta\, p_{i-1,1,0} && i = 2,3,\ldots && (2)\\
(\lambda+\delta)\, p_{i,1,0} &= \nu\, p_{i,0,0} && i = 1,2,\ldots && (3)\\
\delta\, p_{0,1,1} &= \gamma\, p_{1,1,1} && && (4)\\
(\delta+\gamma)\, p_{1,1,1} &= \nu\,(p_{1,0,1} + p_{1,0,1^*}) + \lambda\, p_{1,1,0} + \gamma\, p_{2,1,1} && && (5)\\
(\delta+\gamma)\, p_{i,1,1} &= \nu\, p_{i,0,1} + \lambda\, p_{i,1,0} + \gamma\, p_{i+1,1,1} && i = 2,3,\ldots && (6)\\
(\nu+\gamma)\, p_{1,0,1^*} &= \lambda\, p_{1,0,0} && && (7)\\
p_{0,0,1} &= p_{1,0,1} && && (8)\\
(\nu+\gamma)\, p_{1,0,1} &= \delta\, p_{0,1,1} + \gamma\, p_{2,0,1} && && (9)\\
(\nu+\gamma)\, p_{i,0,1} &= \delta\, p_{i-1,1,1} + \lambda\, p_{i,0,0} + \gamma\, p_{i+1,0,1} && i = 2,3,\ldots. && (10)
\end{aligned}
$$

such that $\sum_{(i,j,k)\in\mathcal{E}} p_{i,j,k} = 1$.

We will not try to solve equations (1)-(10) explicitly. Instead, we will use the standard z-transform approach to characterize the ergodicity condition and the invariant measure of the Markov process. To this end, define for $|z| \le 1$

$$
\begin{aligned}
f(z) &:= \sum_{i=1}^{\infty} z^i p_{i,0,0} & \qquad g(z) &:= \sum_{i=1}^{\infty} z^i p_{i,1,0}\\
h(z) &:= \sum_{i=0}^{\infty} z^i p_{i,0,1} & \qquad k(z) &:= \sum_{i=0}^{\infty} z^i p_{i,1,1}
\end{aligned}
$$

the z-transform of the stationary probabilities $\{p_{i,0,1}\}_{i\ge 0}$, $\{p_{i,1,0}\}_{i\ge 1}$, $\{p_{i,1,1}\}_{i\ge 0}$ and $\{p_{i,0,0}\}_{i\ge 1}$, respectively. Last, introduce for $|z| \le 1$

$$
\begin{aligned}
F(z) &:= p_{1,0,1^*} + p_{0,0,1} + p_{0,1,1} + \sum_{i=1}^{\infty} (p_{i,0,0} + p_{i,1,0} + p_{i,0,1} + p_{i,1,1})\, z^i\\
&= p_{1,0,1^*} + f(z) + g(z) + h(z) + k(z)
\end{aligned}
$$

the z-transform of the stationary probabilities $\{p_{i,j,k},\ (i,j,k) \in \mathcal{E}\}$. $F(z)$ is determined in the following proposition:

Proposition 3.1 (z-transform of the Markov process)

The Markov process depicted in Figure 3 is ergodic if and only if

$$\frac{1}{\gamma} < \frac{1}{\nu} + \frac{1}{\delta}. \tag{11}$$

Under (11), the z-transform $F(z)$ is given by

$$f(z) = \frac{z\,\gamma(\lambda+\delta)(\lambda+\nu)(\gamma+\nu)}{\nu(\nu+\lambda+\gamma)\,a(z)}\,p_{1,0.1} \tag{12}$$

$$g(z) = \frac{z\,\gamma(\lambda+\nu)(\gamma+\nu)}{(\nu+\lambda+\gamma)\,a(z)}\,p_{1,0,1} \tag{13}$$

$$h(z) = -\frac{z^2\,\delta\gamma}{b(z)}\,p_{0,1,1} + \left[\frac{\delta(\nu-\gamma)z^2-\gamma(\nu+\delta)z+\gamma^2}{b(z)} - \frac{z^3\delta\gamma[\lambda\,a(z)+(\gamma+\nu)(\lambda\gamma+(\lambda+\delta)(\lambda+\nu))]}{(\nu+\lambda+\gamma)\,a(z)\,b(z)}\right]p_{1,0,1} \tag{14}$$

$$k(z) = \frac{\gamma\,(\gamma-z\,(\gamma+\nu))}{b(z)}\,p_{0,1,1} - \frac{\gamma\,(\lambda+\nu)(\gamma+\nu)(\lambda\gamma+(\lambda+\delta)(\lambda+\nu))}{(\nu+\lambda+\gamma)\,a(z)\,b(z)}\,z^2\,p_{1,0,1} \tag{15}$$

$$p_{1,0,1^*} = \frac{\lambda\gamma}{\nu\,(\nu+\lambda+\gamma)}\,p_{1,0,1} \tag{16}$$

for $|z|\le 1$, with

$$\begin{aligned} a(z) &:= -\delta\nu z + (\lambda+\delta)(\lambda+\nu) \\ b(z) &:= \delta\nu z^2 - \gamma(\gamma+\nu+\delta)z + \gamma^2. \end{aligned}$$

The constants $p_{0,1,1}$ and $p_{1,0,1}$ are given in (48) and (49). ◇

The proof of Proposition 3.1 is given in Appendix A.

3.2.1 The expected number of forwarders

In this section we compute the expected number of forwarders in steady-state between the agent and the message (resp. the source if there is no message). Let $q(i)$ be the probability that the agent is located $i \ge 1$ hops away from a message (resp. from the source if there is no message), which corresponds to a situation where there are exactly $i-1$ forwarders in the system. Clearly,

$$\begin{aligned} q(1) &= p_{1,0,0} + p_{1,1,0} + p_{1,0,1} + p_{1,1,1} + p_{1,0,1^*} \\ q(i) &= p_{i,0,0} + p_{i,1,0} + p_{i,0,1} + p_{i,1,1}, \qquad i = 2,3,\ldots \end{aligned}$$

and the mean number N of forwarders is given by

$$\begin{aligned} N &= \sum_{i=1}^{\infty}(i-1)\,q(i) \\ &= \sum_{i=1}^{\infty} i\,q(i) - \sum_{i=1}^{\infty} q(i) \\ &= f'(1)+g'(1)+h'(1)+k'(1)+p_{1,0,1^*}-(1-p_{0,0,1}-p_{0,1,1}) \end{aligned} \tag{17}$$

where $\phi'(1)$ denotes the derivative of $\phi(z)$ at point $z=1$.

From (13) and (12) we find

$$f'(1)+g'(1) = \frac{\gamma(\lambda+\delta)(\lambda+\nu)^2(\gamma+\nu)}{\lambda^2\nu(\nu+\lambda+\gamma)(\lambda+\delta+\nu)}\,p_{1,0,1} \tag{18}$$

whereas the relation

$$h'(1)+k'(1) = \frac{\delta\nu(\delta\gamma+\nu^2)-(\nu^2\delta(\nu-\gamma)+\gamma(\nu+\delta)(\delta\gamma-\nu^2))p_{1,0,1}}{\nu(\delta+\nu)(\gamma(\delta+\nu)-\delta\nu)} + \frac{\delta\gamma(\gamma+\nu)(\delta\lambda(\nu-\gamma)+\nu(\lambda+\nu)(\lambda+\delta))}{\lambda^2(\nu+\lambda+\gamma)(\lambda+\nu+\delta)(\gamma(\delta+\nu)-\delta\nu)}p_{1,0,1} \tag{19}$$

follows from (14) and (15). Combining now (17), (18) and (19) with the expressions found for $p_{0,0,1}$, $p_{1,0,1^*}$ and $p_{0,1,1}$ (in (8), (16) and (45), respectively), we find

$$N = \frac{\delta^2(\gamma^2-\gamma\nu+\nu^2)(1-p_{1.0.1})}{\gamma(\delta+\nu)(\gamma(\delta+\nu)-\delta\nu)} + \left(\frac{\nu\gamma(\gamma+\lambda)(\gamma+\nu)}{\lambda^2(\nu+\lambda+\gamma)} - \frac{\gamma^2-\nu^2}{\nu}\right)\frac{\delta\,p_{1,0,1}}{\gamma(\delta+\nu)-\delta\nu}$$

where $p_{1,0,1}$ is given in (49). The average number of forwarders is expected to grow when δ or ν increases and it should back off when λ or γ increases which is illustrated in Figure 4. The crosses in each graph correspond to the same values of the model parameters: $\lambda=1, \nu=10, \delta=20, \gamma=50$. As shown in [4] N can be used to evaluate the fault-tolerance of the protocol. This issue is not addressed here.

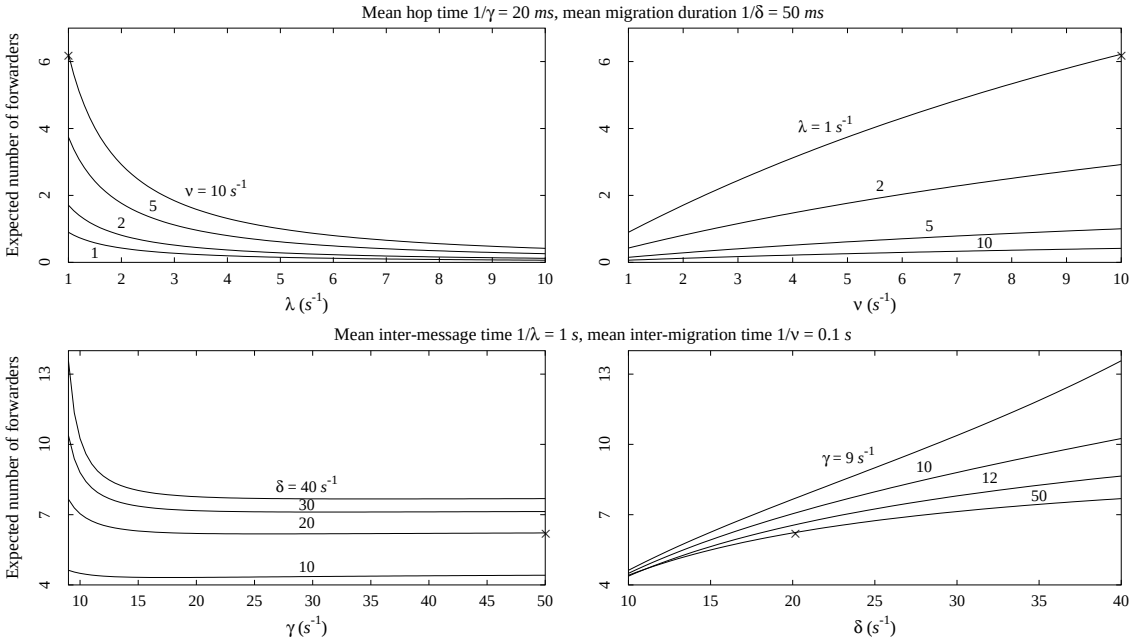

Figure 4: The expected number of forwarders

3.2.2 The expected communication time

In this section we determine the expected communication time. The communication time is the time needed to a message to reach the mobile object, to which we must add the time it takes to the mobile object to send its new position to the source in case the message has to go through at least one forwarder. If the message reaches the mobile object after exactly one hop then there is no need to the mobile object to send its position to the source since the source knows it; in this case, the communication terminates once the message has reached the object, thereby justifying the definition of the communication time given above.

At the end of a communication the agent is not migrating, the source idles and it is one hop away from the agent. This corresponds to state $(1, 0, 0)$. At this time, the source stays idle for an exponentially distributed duration with mean $1/\lambda$. After this idling period a new communication is initiated and the system can be in any one of the following states: $(1, 0, 1^*)$, $(i, 0, 1)$ for $i \geq 2$, or $(i, 1, 1)$ for $i \geq 1$. Define $T_{i,j,k}$ with $i \geq 1$, $j = 0, 1$, $k = 1$ or with $(i, j, k) = (1, 0, 1^*)$, as the expected communication time given that a message was generated when the system was in state (i, j, k) just after the generation of the message.

The expected communication time T_F (the subscript F refers to "Forwarders") is given by

$$T_F = q_F(1, 0, 1^*)\, T_{(1,0,1^*)} + \sum_{i=2}^{\infty} q_F(i, 0, 1)\, T_{(i,0,1)} + \sum_{i=1}^{\infty} q_F(i, 1, 1)\, T_{(i,1,1)}$$

where $q_F(i, j, k)$ denotes the probability of reaching state (i, j, k) given that the process initiated in state $(1, 0, 0)$. It actually represents the probability that a communication starts when the system moves from state $(i, j, 0)$ to state (i, j, k). With the help of Figure 3 we find that

$$q_F(1, 0, 1^*) = \frac{\lambda}{\lambda + \nu} \tag{20}$$

$$q_F(i, 0, 1) = \frac{\lambda(\lambda + \delta)}{\delta\nu}\, r^i \quad i = 2, 3, \ldots \tag{21}$$

$$q_F(i, 1, 1) = \frac{\lambda}{\delta}\, r^i \quad i = 1, 2, \ldots \tag{22}$$

where $r := \frac{\delta\nu}{(\lambda+\delta)(\lambda+\nu)} < 1$. By using (20)-(22) we may rewrite T_F as follows

$$T_F = \frac{\lambda}{\lambda+\nu}\, T_{1,0,1^*} + \frac{\lambda(\lambda+\delta)}{\delta\nu} \sum_{i=2}^{\infty} r^i\, T_{i,0,1} + \frac{\lambda}{\delta} \sum_{i=1}^{\infty} r^i\, T_{i,1,1}. \tag{23}$$

With the definitions $G(z) := \sum_{i=0}^{\infty} z^i\, T_{i,0,1}$ and $H(z) := \sum_{i=0}^{\infty} z^i\, T_{i,1,1}$, (23) becomes

$$T_F = \frac{\lambda}{\lambda+\nu} T_{1,0,1^*} + \frac{\lambda(\lambda+\delta)}{\delta\nu}\, (G(r) - r\, T_{1,0,1} - T_{0,0,1}) + \frac{\lambda}{\delta}\, (H(r) - T_{0,1,1})\,.$$

It remains to determine the generating functions $G(z)$ and $H(z)$ at $z = r$. To this end we will use the following recursive equations that follow from the Markovian description of the protocol displayed in Figure 3:

$$\begin{aligned}
T_{1,0,1^*} &= \frac{1}{\nu+\gamma} + \frac{\nu}{\nu+\gamma}\, T_{1,1,1} \\
T_{0,0,1} &= \frac{1}{\gamma} \\
T_{i,0,1} &= \frac{1}{\nu+\gamma} + \frac{\nu}{\nu+\gamma}\, T_{i,1,1} + \frac{\gamma}{\nu+\gamma}\, T_{i-1.0,1} \qquad i = 1, 2, \ldots \\
T_{0,1,1} &= \frac{1}{\delta} + T_{1,0,1} \\
T_{i,1,1} &= \frac{1}{\delta+\gamma} + \frac{\delta}{\delta+\gamma}\, T_{i+1,0,1} + \frac{\gamma}{\delta+\gamma}\, T_{i-1,1,1} \quad i = 1, 2, \ldots
\end{aligned}$$

which yields

$$\begin{aligned}
(\nu + \gamma\,(1-z))\, G(z) - \nu\, H(z) &= \frac{1}{1-z} + \frac{\nu}{\gamma} - \nu T_{0,1,1} \\
-\delta\, G(z) + (\delta + \gamma\,(1-z))\, zH(z) &= \frac{z}{1-z} - \frac{\delta}{\gamma} + \gamma z\, T_{0,1,1}.
\end{aligned}$$

Solving for $G(z)$ and $H(z)$ gives

$$G(z) = \frac{1}{D(z)} \left(\frac{(\delta+\nu)\, z}{1-z} + \frac{\nu}{\gamma}(1-z)(\gamma z - \delta) + \gamma z + \nu(\gamma z - \delta)\, z T_{0,1,1} \right) \tag{24}$$

$$H(z) = \frac{1}{D(z)} \left(\frac{(\delta+\nu)\, z}{1-z} + (\gamma+\delta)\, z - (\gamma^2\, z^2 - \gamma(\gamma+\nu)\, z + \delta\nu) T_{0,1,1} \right) \tag{25}$$

where we have defined

$$D(z) := (z-1)\left(\gamma^2\, z^2 - \gamma(\gamma+\nu+\delta)\, z + \delta\nu\right). \tag{26}$$

Using (24)-(26) and after some algebraic manipulations, we find (use $T_{1,0,1^*} = T_{0,1,1} - 1/\delta - 1/(\gamma+\nu)$)

$$T_F = \frac{1}{\alpha(\lambda)} \left[((\lambda+\delta)(\lambda+\nu) - \gamma\nu)\, T_{0,1,1} - \frac{(\lambda+\delta)(\lambda+\nu+\delta)}{\delta} - \frac{(\lambda+\delta)(\lambda+\nu)(\lambda(\lambda+\nu) + \nu(\gamma+\nu)) + \delta\gamma\nu\lambda}{\lambda\,(\lambda+\nu)(\gamma+\nu)} \right] \tag{27}$$

where

$$\alpha(\lambda) := (\lambda+\delta)(\lambda+\nu) - \gamma(\lambda+\nu+\delta).$$

It remains to identify the constant $T_{0,1,1}$ in (27). This can be done by noticing that $\alpha(\lambda)$ has a single zero $\lambda = \lambda_0$ in $[0, \infty)$ given by

$$\lambda_0 = \frac{\gamma - \nu - \delta + \sqrt{(\gamma+\nu+\delta)^2 - 4\delta\nu}}{2}.$$

In order for T_F to be well-defined for all non-negative values of λ, the coefficient of $1/\alpha(\lambda)$ in (27) must vanish when $\lambda = \lambda_0$, which gives us an extra relation from which we can determine $T_{0,1,1}$. Using the identity $(\lambda_0+\delta)(\lambda_0+\nu) = \gamma(\lambda_0+\nu+\delta)$ and setting the coefficient of $1/\alpha(\lambda)$ in (27) to 0 when $\lambda = \lambda_0$, gives

$$T_{0,1,1} = \frac{\gamma(\lambda_0+\nu+\delta) + \delta\,\lambda_0}{\gamma\delta\lambda_0}.$$

Finally

$$T_F = \begin{cases} \dfrac{1}{\alpha(\lambda)}\left[((\lambda+\delta)(\lambda+\nu)-\gamma\nu)\,T_{0,1,1} - \dfrac{(\lambda+\delta)(\lambda+\nu+\delta)}{\delta}\right. & \\ \qquad \left. - \dfrac{(\lambda+\delta)(\lambda+\nu)(\lambda(\lambda+\nu)+\nu(\gamma+\nu))+\delta\gamma\nu\lambda}{\lambda(\lambda+\nu)(\gamma+\nu)}\right] & \text{for } \lambda \neq \lambda_0 \\ \dfrac{(2\lambda+\nu+\delta)(2\gamma(\gamma+\nu)(\lambda+\nu+\delta)+\delta\lambda(\lambda+\nu)((\gamma+\nu)\,T_{0,1,1}-1))}{\lambda\delta(2\lambda+\nu+\delta-\gamma)(\lambda+\nu)(\gamma+\nu)} & \\ \qquad - \dfrac{(2\lambda+\nu)(\lambda+\delta)(\lambda+\nu-\gamma(\gamma+\nu)\,T_{0,1,1})+\gamma\nu\delta}{\lambda(2\lambda+\nu+\delta-\gamma)(\lambda+\nu)(\gamma+\nu)} & \text{for } \lambda = \lambda_0 \end{cases} \tag{28}$$

where the latter relation is obtained after a routine application of l'Hopital's rule.

4 Centralized Server

4.1 Description

An alternative to the forwarders approach for locating a mobile object is to use a *location server*. Such a server keeps track of the location of mobile objects in a database. Servers like this are widely used in the Internet. For instance, the Domain Name Server [20] uses a hierarchically organized servers to associate location (IP address) to symbolic name. For sake of simplicity, we will consider here a single *centralized* server, although many different schemes can be conceived to improve speed and reliability. We will also assume that each object (source or mobile agent) knows the location of this server.

The idea behind location server is simple: each time a mobile object migrates, it informs the server of its new location. Whenever the source wants to reach the mobile, it sends a message to the last known location of the mobile; if this communication fails, then the source sends a `location request` to the server. This solution is often referred to as a lazy solution since the references to a mobile object are only updated when needed. We now give a careful description of the protocol used by the source and the mobile object to communicate with the server:

- The Mobile Object

  Step 1: Performs the migration;

  Step 2: Sends its new location to the server.

- The Source

  Step 1: Issues a message to the mobile object with the recorded location. Upon failure goes to Step 2;

  Step 2: Queries the server to have the current location of the mobile object;

  Step 3: Issues a message to the mobile object with the location provided by the server. Upon failure, returns to Step 2.

The above protocol is implemented in various Java libraries (*MOA* [19] and *ProActive* [24]). An illustration of this approach is given in Figure 5. In Figures 5(b)-(c) the chronological order of the events is indicated by the numbers $1, 2, \ldots$. In Figure 5(b) a migration (events no. 1 & 2) takes place before a message has been sent by the source. When the source finally sends a message (event no. 3) to the agent at Host B (last known position of the agent), it receives a communication error from Host B (event no. 4). The source then asks the home site to give it the location of the agent (event no. 5). Once the source knows the new location of the agent (event no. 6), it sends a copy (event no. 7) of the original message to Host C (i.e. to the agent). The communication is successful and the source becomes idle.

Figure 5(c) displays a more complicated situation. In this case, the server does not give the correct location of the agent to the source since the agent has initiated a migration in the meantime. As a result, the source will have to send a second location request to the server (event no. 9) before finally being able to reach the agent (event no. 11).

Regarding the service policy, there exist several different schemes that can be implemented. In *ProActive* [24],

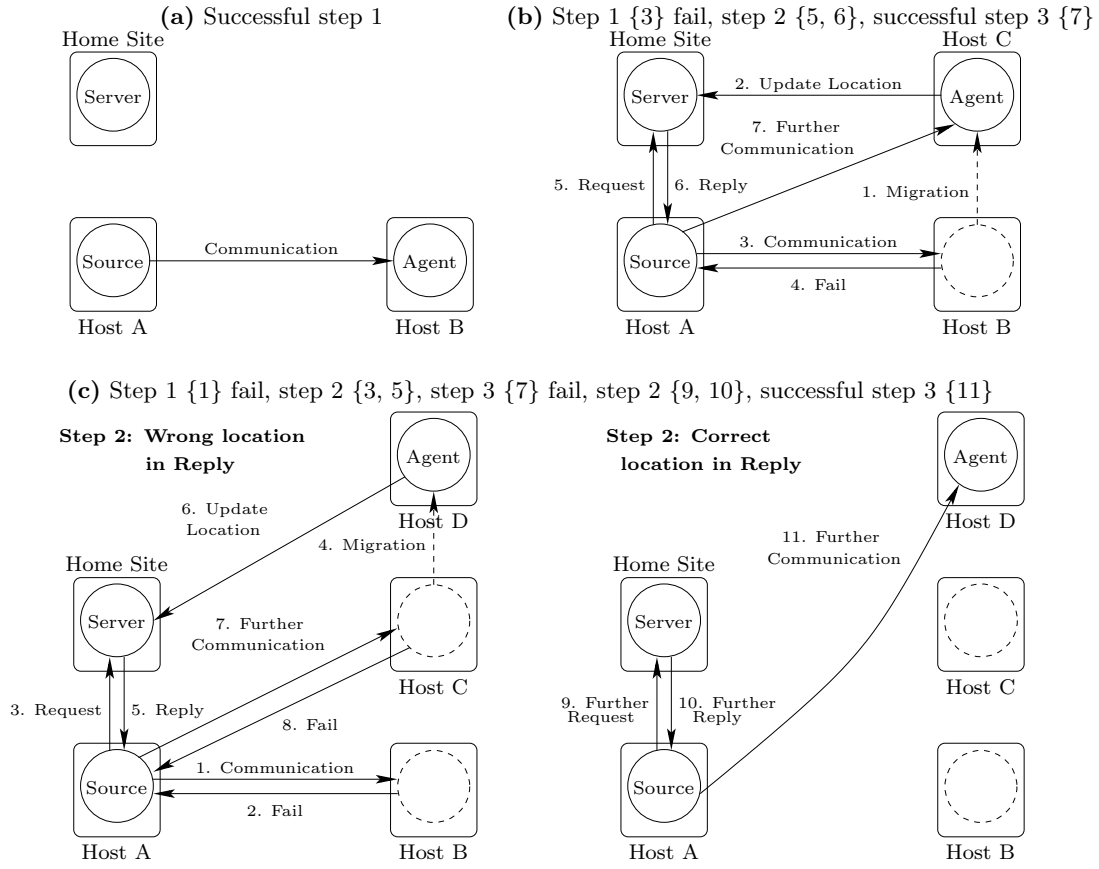

Figure 5: Some possible scenarios in the centralized approach from the source point of view.

the performance of the server has been optimized so that it does not act as a bottleneck. The buffer at the server is completely partitioned in the sense that each pair source-agent possesses its own queue. The server polls these different queues and attends the queue having the oldest query. Within each queue, there is a priority scheduling mechanism that gives priority to to `update location requests` over `location requests`. If a queue contains several `update location requests` sent by the same object then only the newest is processed and the others are destroyed. Note that the service policy is non-preemptive since *ProActive* is a Java library and that the execution of a Java method cannot be stopped. As a consequence an `update location request` under processing cannot be preempted by a more recent one, which may harm the performance of the protocol[2]. Fortunately, it has been observed that this event is very rare in practice. Another consequence of this scheme is that upon serving a `location request`, the server has to check whether the corresponding queue contains an `update location request` coming from the object; if this is the case, then the `location request` is sent back to the queue; otherwise, it sends the position of this object (as found in the database) to the source. Finally, communications in *ProActive* between the server and the different sources are synchronous. Therefore, a source that has sent a `location request` cannot perform any other task before it gets a reply from the server, which implies there is at most one pending `location request` per source at the server.

Observe that the server delivers a customized service to each type of queries: for `update location requests` it just updates the agent location in a database, whereas for `location requests` it scans the database for the current agent location and further scans the queue for any `update location request`.

[2]Notice that only the queue that is attended by the server may have two `update location requests`, all other ones having at most one single pending `update location request`.

4.2 A Markovian analysis of the server

An exact modeling of the centralized approach would consist in modeling the system as a polling server with vacations. In this section, we develop an approximate model that considers a single queue – thereby a single source communicating with a single agent – where the processing speed of the server is reduced to account for the contention due to the potential presence of several agents/sources. This queue may have at most a single `location request` and two `update location requests`. By arguing that it is highly unlikely that a coming `update location request` finds an `update location request` (from the same agent) being processed, we will assume that there can be at most one pending `update location request` at the queue or, in other words, that an `update location request` under processing is preempted by a more recent one (cf. Section 4.1). The latter restriction can easily been removed at the expanse of enlarging the state-space (which would yield a 35% increase in the number of states).

We assume that the set of assumptions **A1** holds (cf. Section 2). Note, however, that in this context p_j will represent the sum of the j-th traveling time of the agent to its new host and of the travel time of the associated `update location request` to the server site. We further assume that the traveling times between the source and the presumed location of the mobile object (resp. between the source and the location server) are i.i.d. exponentially distributed rvs with parameter $\gamma_1 > 0$ (resp $\gamma_2 > 0$). Finally, we assume that the service times – regardless of the query type – are i.i.d. exponentially distributed rvs with parameter $\mu > 0$ (notice that if there are only one source and one agent in an application μ would be the server processing speed). All these rvs are assumed to be mutually independent.

We model the system behavior by a finite-state Markov process whose transition diagram and rates are given in Figure 6. A state has the representation $(\mathbf{i}, j, k)$ with $\mathbf{i} \in \{A, B, \ldots, G\}$, $j \in \{0, 0^*, 1, 1^*\}$ and $k \in \{0, 1, 1^*, 2\}$, where the 2-dimensional vectors $A, B, \ldots, G$ are defined in Figure 6.

More precisely, the vector $\mathbf{i} = (i_1, i_2)$ represents the state of a queue at the server, namely, the type of the message (`update location request` or `location request`) in the queue, with $i_l \in \{0, \lambda, \nu\}$ the type of the message that occupies the l-th position in the queue ($l = 1, 2$). By convention, $i_l = 0$ (resp. $i_l = \lambda$, $i_l = \nu$) indicates that the l-th position is not occupied (resp. the l-th position is occupied by a `location request`, the l-th position is occupied by an `update location request`). Recall that `update location requests` have non-preemptive priority over a `location request` and that an arriving `update location request` that finds one `update location request` will destroy it at once.

The component $j \in \{0, 0^*, 1, 1^*\}$ in the state description $(\mathbf{i}, j, k)$ represents the state of the object: $j = 1$ (resp. $j = 1^*$) will indicate that the object is migrating and that the source knows (resp. does not know) the location of the host that the object is leaving; similarly, $j = 0$ (resp. $j = 0^*$) will indicate that the object is not migrating and that the source knows (resp. does not know) its location.

Finally, the component $k \in \{0, 1, 1^*, 2\}$ in the state description $(\mathbf{i}, j, k)$ represents the state of the source: $k = 0$ if the source has no activity, $k = 1$ if it has sent a message to the object, $k = 1^*$ if the message sent by the source has reached a site that the object is about to leave, and $k = 2$ if it has sent a `location request` to the server. The latter only occurs if the presumed location of the active object is no longer valid.

The system enters state $(A, 0, 0)$ just after the end of a communication (defined as the instant when a message has reached the agent). It remains in that state for an exponentially distributed duration with mean $1/\lambda$, and then a new message is generated by the source. The time that elapses between the generation of a new message by the source and the next visit to state $(A, 0, 0)$ is the *communication time* (i.e. quantities τ_i's introduced in Section 2). In other words, the successive communication times $\{\tau_i\}_i$ are initialized when k goes from 0 to 1, and are stopped when k goes from 1 to 0. Each time a communication fails, a request is issued to the server and k switches from 1 to 2. As soon as the server replies, k switches back to 1 and the message is re-issued by the source to the location of the object returned by the server (which may or may not be the current location).

Under the above description/assumptions, the process depicted in Figure 6 is an irreducible finite-state Markov process on the state-space $\mathcal{F}$, where $\mathcal{F}$ is the set of all states indicated in Figure 6 ($\mathcal{F}$ contains 27 elements). Let $\mathbf{p} = \{p_{\mathbf{i},j,k},\ (\mathbf{i}, j, k) \in \mathcal{F}\}$ be the stationary probability of this Markov process ($\mathbf{p}$ exists since an irreducible finite-state Markov process is ergodic). If $\mathbf{Q}$ denotes the infinitesimal generator of this Markov process (whose elements can

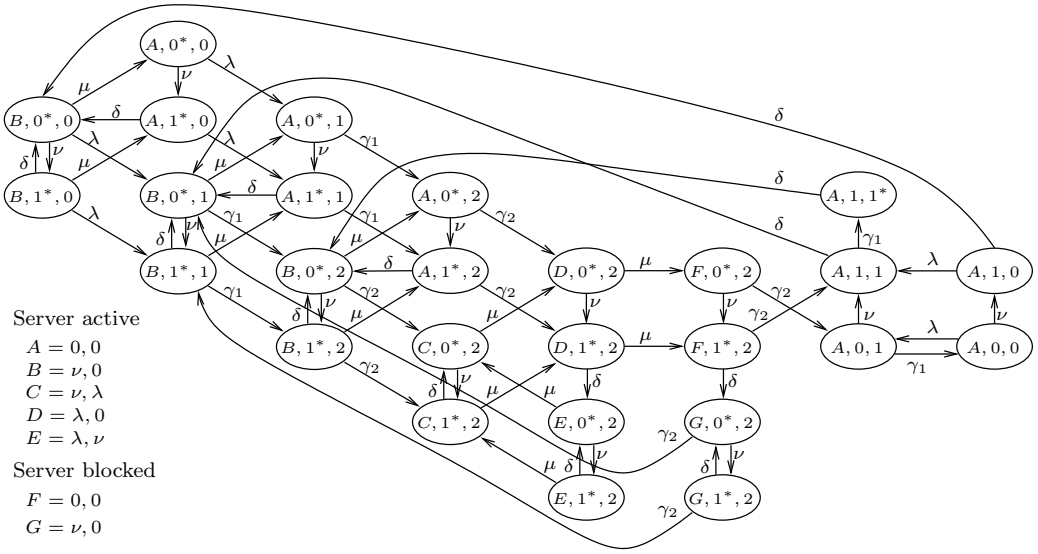

Figure 6: System states and transition rates in the centralized approach.

easily be identified from Figure 6), then we know (see e.g. [16]) that $\mathbf{p}$ is the unique solution of the system of linear equations $\mathbf{p}.\mathbf{Q} = 0$, $\mathbf{p}.\mathbf{1} = 1$, which can be solved by using a routine numerical procedure.

4.2.1 The expected communication time

We are interested in finding the expected communication time (see Section 4.2), denoted as T_S (the subscript S refers to "Server"). Recall that a communication begins when the source – after an idling period with mean $1/\lambda$ – sends a message to the active object and terminates when the object is reached. It is seen from Fig. 6 that a message may only be generated when the system is in 6 distinct states – all states where $k = 0$. As soon as $k = 1$ a message is generated. Hence, a communication may start only when the system is in one of the following states: $(A, 0, 1)$, $(A, 1, 1)$, $(A, 0^*, 1)$, $(A, 1^*, 1)$, $(B, 0^*, 1)$ or $(B, 1^*, 1)$.

Let $T_{\mathbf{i},j,k}$ denote the expected time to hit state $(A, 0, 0)$ starting from state $(\mathbf{i}, j, k)$. The expected response time T_S of the system is given by

$$\begin{aligned} T_S = {} & q_S(A,0,1)\, T_{A,0,1} + q_S(A,0^*,1)\, T_{A,0^*,1} + q_S(B,0^*,1)\, T_{B,0^*,1} \\ + {} & q_S(A,1,1)\, T_{A,1,1} + q_S(A,1^*,1)\, T_{A,1^*,1} + q_S(B,1^*,1)\, T_{B,1^*,1} \end{aligned} \tag{29}$$

where $q_S(\mathbf{i}, j, 1)$ denotes the probability that the communication is initiated when the system is in state $(\mathbf{i}, j, 1)$. With Figure 6 it is easily seen that

$$q_S(A,0,1) = \frac{\lambda}{\lambda+\nu} \tag{30}$$

$$q_S(A,1,1) = \frac{\nu\lambda}{(\lambda+\delta)(\lambda+\nu)} \tag{31}$$

$$q_S(A,0^*,1) = \frac{\nu\mu\delta}{(\lambda+\nu)(\lambda+\nu+\mu)(\lambda+\delta+\nu)} \tag{32}$$

$$q_S(A,1^*,1) = \frac{\nu^2\mu\delta(2\lambda+\delta+\mu+\nu)}{(\lambda+\delta)(\lambda+\nu)(\lambda+\nu+\mu)(\lambda+\delta+\nu)(\mu+\lambda+\delta)} \tag{33}$$

$$q_S(B,0^*,1) = \frac{\nu\delta}{(\lambda+\nu+\mu)(\lambda+\delta+\nu)} \tag{34}$$

$$q_S(B,1^*,1) = \frac{\nu^2\delta}{(\lambda+\nu+\mu)(\lambda+\delta+\nu)(\mu+\lambda+\delta)}. \tag{35}$$

The derivation of the last four identities is given in Appendix B.

It remains to compute the hitting times $T_{\mathbf{i},j,k}$. They are easily identified from the infinitesimal generator matrix $\mathbf{Q}$ as follows (see Thm 3.3.3 pp. 113-114 in [21])

$$\begin{aligned} T_{A,0,0} &= 0 \\ \sum_{\mathbf{i},j,k} q_{(\mathbf{i}',j',k'),(\mathbf{i},j,k)} T_{\mathbf{i},j,k} &= -1 \quad \text{for } (\mathbf{i}',j',k') \neq (A,0,0) \end{aligned} \tag{36}$$

where $q_{(\mathbf{i}',j',k'),(\mathbf{i},j,k)}$ are the elements of the generator matrix $\mathbf{Q}$.

In order to simplify (36), let us introduce $\mathbf{M}_{A,0,0}$ as the minor of the matrix $\mathbf{Q}$ obtained by removing the row and the column corresponding to state $(A,0,0)$. Let $\mathbf{T} = \{T_{\mathbf{i},j,k},\ (\mathbf{i},j,k) \in \mathcal{F} - \{(A,0,0)\}\}$ be the vector of hitting times, except $T_{A,0,0}$ (which is null). Equation (36) then reads

$$\mathbf{M}_{A,0,0} \cdot \mathbf{T} = -\mathbf{1}. \tag{37}$$

Solving for (37) and using (30)-(35) we finally find T_S.

5 Validation and comparison

5.1 Validation through simulations

The theory developed in Sections 3 and 4 relies on several assumptions. 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) were all assumed to be exponential rvs and independent from each other.

In this section we undertake validation of the Markovian models presented in Sections 3 and 4 against results obtained from event-driven simulators of both schemes. In the simulations, we have considered a single agent and a single source. We have run each simulator several times having one assumption violated at a time, and one time having all assumptions violated. This way we can observe the robustness of the corresponding model and the impact of each assumption on its performance.

In order to test realistic distributions for the rvs in hand we have collected measurements on a LAN and a MAN and fitted the resulting data to well-known distributions. Table 1 summarizes our findings. Except for the service times which are approximately constant, all other (network-dependent) parameters are well represented by a Weibull distribution. The distribution of the communication rate λ (inverse of the average idle times for the source) and the migration rate ν (inverse of the average idle times for the agent) depends only on the application. To test the robustness of the models against each assumption, we have run simulations where all random variables are exponential except one whose distribution is either deterministic (case of idle times and service times) or Weibull (case of migration durations and travel times). At last, we have run the simulators having all random variables being non-exponential. In these runs, the only assumptions not violated are the ones concerning the independence of the processes in hand.

	100Mb/s switched LAN	7Mb/s MAN
migration durations (forwarders)	Weibull shape 4, scale 47	Weibull shape 2.5, scale 280
migration durations (server)	Weibull shape 2.5, scale 73	Weibull shape 3, scale 640
travel times (forwarders)	Weibull shape 11, scale 49	Weibull shape 6, scale 9
travel times source-agent (server)	Weibull shape 1.8, scale 25	Weibull shape 10, scale 15
travel times source-server (server)	Weibull shape 1.8, scale 17	Weibull shape 1.8, scale 25
service times	$\approx$ constant	$\approx$ constant

Table 1: Distribution fits for the model parameters.

Table 2 reports the sample mean and the percentiles of the relative error (expressed in percent) between simulated results and theoretical expected response times as predicted by both models. It appears that both models are very

robust against Weibull travel times. In 95% of the simulations conducted, the relative error on the communication time stays under 7.1% (resp. 2.3% and 2.7%) in the forwarders mechanism (resp. centralized mechanism) (see line 4 (resp. lines 9 and 10) in Table 2).

Both models are not sensitive to the distribution of idle times for the agent. In 95% of the simulations conducted, the relative error on the communication time stays within 4.2% (resp. 4.8%) of the simulated value in the forwarders mechanism (resp. centralized mechanism) (see line 2 (resp. line 7) in Table 2). The model of the location server is very robust against deterministic service times. The largest error observed during the simulations is equal to 3.7%.

However, the models are more sensitive to the distribution of the migration durations and the idle times for source. Nevertheless, when simulating the forwarders scheme the mean relative error is equal to 7.7% (resp. 8.3%) in case the idle times for source are deterministic (resp. the migration durations are Weibull) (see 1st column in Table 2 line 1 (resp. line 3)). When simulating the centralized approach, we observe almost the same relative error in case the idle times for source are deterministic (resp. the migration durations are Weibull), the largest error being 17.2% (resp. 16%).

When all the assumptions concerning the distribution of the rvs are violated, the performance of the models is fair. In half of the simulations the relative error on the response time is less than 10.5% and its mean is equal to 14.1% in both models (see lines 5 and 12 in Table 2). The robustness of the models against correlated processes will be addressed in the next section.

5.2 Validation through experiments

In order to further validate the models, we have conducted extensive experiments on a LAN and a MAN. All benchmarks were written using the *ProActive* library [24] which provided us with all the necessary mobile primitives. The network was composed of various 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 7Mb/s MAN (four machines were used overall). We used Java 1.2.2 Green threads [11] in all experiments.

For each approach (forwarders and centralized server), the testbed was composed of a single mobile object and 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 session. All the other model parameters $(\delta, \gamma, \gamma_1, \gamma_2, \mu)$ are system-dependent and cannot be changed. Hence, among all the assumptions that we made to construct the models, only 2 assumptions are not violated (the ones concerning the *input* sequences $\{w_i, i \geq 1\}$ and $\{u_k, k \geq 1\}$ (see Section 2)). Indeed, migration durations, communications latencies and service times were found to have Weibull distributions both on a LAN and a MAN. Further, assumptions on the independence of these processes are not likely to be valid under real conditions. Migration durations and communication latencies are particularly correlated in

		Mean	25%	50%	75%	90%	95%
Forwarders	deterministic idle times for source $\lambda \in [1, 10]$	7.7	6.5	8.8	9.6	10.3	10.5
default values:	deterministic idle times for agent $\nu \in [1, 10]$	1.6	0.3	1.4	2.4	3.5	4.2
$\lambda = 1, \nu = 10$	Weibull migration durations $\delta \in [8, 25]$ (shape 4)	8.3	4.3	8.2	10.4	14.7	16.3
$\delta = 11$	Weibull travel times $\gamma \in [33.8, 69.8]$ (shape 11)	2.7	1.0	2.2	3.9	6.4	7.1
$\gamma = 45$	All together $\lambda, \nu \in \{1, 3, 5, 7, 9\}$	14.1	4.9	10.0	20.6	32.4	38.3
Server	deterministic idle times for source $\lambda \in [1, 10]$	14.9	15.1	16.3	17.1	17.1	17.2
default values:	deterministic idle times for agent $\nu \in [1, 10]$	2.4	2.2	2.2	2.6	4.1	4.8
$\lambda = 1, \nu = 10$	Weibull migration durations $\delta \in [12, 19]$ (shape 2.5)	12.3	11.7	12.2	13.5	14.8	15.5
$\delta = 15$	Weibull travel times $\gamma_1 \in [90.0, 130.8]$ (shape 1.8)	1.3	0.6	1.5	1.8	2.1	2.3
$\gamma_2 = 75$	Weibull travel times $\gamma_2 \in [56.2, 93.7]$ (shape 1.8)	0.9	0.3	0.6	1.3	2.2	2.7
$\gamma_2 = 75$	deterministic service times $\mu \in [500, 2500]$	1.5	0.6	1.5	2.2	2.9	3.5
$\mu = 2325$	All together $\lambda, \nu \in \{1, 3, 5, 7, 9\}$	14.1	6.1	10.5	17.0	30.0	40.1

Table 2: Sample mean and percentiles of the relative error provided by the models.

		Mean	25%	50%	75%	90%	95%
Forwarding mechanism	LAN (100Mb/s)	9.7	2.2	6.4	12.7	24.8	34.4
	MAN (7Mb/s)	12.1	2.3	7.8	19.0	25.9	35.0
	overall	10.9	2.3	7.3	17.0	25.1	34.8
Centralized approach	LAN (100Mb/s)	4.6	2.3	4.3	6.2	8.5	10.4
	MAN (7Mb/s)	16.0	5.8	11.1	21.0	34.3	37.0
	overall	10.8	3.8	6.5	12.7	23.5	34.5

Table 3: Sample mean and percentiles of the relative error in both mechanisms

real life.

Figure 7 reports both the experimental and theoretical expected response times obtained for a LAN (4 upper graphs) and for a MAN (4 lower graphs). Graphs on the left display the expected response time as a function of the communication rate λ for λ ranging from $1s^{-1}$ to $10s^{-1}$, for three different values of the migration rate ν ($\nu = 1, 5, 10$); similarly, graphs on the right display the expected response time as a function of the migration rate ν for ν ranging from $1s^{-1}$ to $10s^{-1}$, for three different values of the communication rate λ ($\lambda = 1, 5, 10$). For each value of the pair (λ, ν), the empirical values of δ and γ (resp. $\delta, \gamma_1, \gamma_2$ and μ) were plugged into the expression of T_F (resp. T_S) given in (28) (resp. (29)) in case the forwarding mechanism (resp. the centralized mechanism) was used. Observe that analytical and experimental response times are very close to each other over almost all experiments. For each network configuration (LAN or MAN), the performance of the models are collected in Table 3 in means of order statistics and the sample mean (1st column) of the relative error between analytical results and experimental values. The second column in Table 3 gives the 25^{th} percentile of the relative error. The third column gives the median value, the fourth column gives the 75^{th} percentile, etc.

Results in Table 3 indicate that the theoretical models behave fairly well. Their overall performance are very close to each other: the sample mean of the relative error is equal to 10.9% (resp. 10.8%) for the model of the forwarders (resp. of the server) (see 1st column in Table 3) and in 95% of the experiments using the forwarders (resp. the location server) the relative error is below 34.8% (resp. 34.5%) (see last column in Table 3). However, both models are more robust when applied on a LAN (see lines 1 and 4 in Table 3) than when applied on a MAN (see lines 2 and 5 in Table 3).
The model of the server has been validated for one source-agent pair. We still need to perform experiments including multiple sources and/or agents in order to complete the validation of the model. Beside validation, we can use the experimental results to compare the performance of the location mechanisms. Looking at the 4 upper graphs in Figure 7 (experiments on a LAN), it appears that the location server scheme has a lower response time if compared to the forwarders scheme. Unexpectedly, we observe the opposite in the 4 lower graphs in Figure 7 (experiments on a MAN), where the forwarders scheme achieves the smallest communication time. It looks like the location server scheme performs the best only within high speed networks. When communication latencies increases (and subsequently the migration durations), the forwarding technique surpasses the centralized technique in performance.

5.3 A theoretical comparison of both approaches

As already mentioned, many parameters are network-dependent so that an experimental comparison of both the forwarder approach and the centralized server approach is necessarily limited to a few scenarios. No such limitations occur when comparing both approaches by using the theoretical results obtained in Sections 3 and 4. We will focus on the expected response time given by each approach and, more precisely, on their difference ΔT

$$\Delta T = T_F - T_S \tag{38}$$

where T_F and T_S are given in (28) and (29), respectively. Four cases have been investigated corresponding to different values of the model parameters. Unless otherwise mentioned, the parameters have the values listed in Table 4. Each entry in Table 4 is the average value obtained over all experiments conducted on a LAN and on a MAN. In each case we have identified regions where $\Delta T = 0$, which corresponds to situations where both schemes yield the same expected response (communication) times. When $\Delta T > 0$ (resp. $\Delta T < 0$) the location server (resp. forwarding) scheme gives the best performance. Figure 8 displays the sign of ΔT in all four cases that we have investigated.

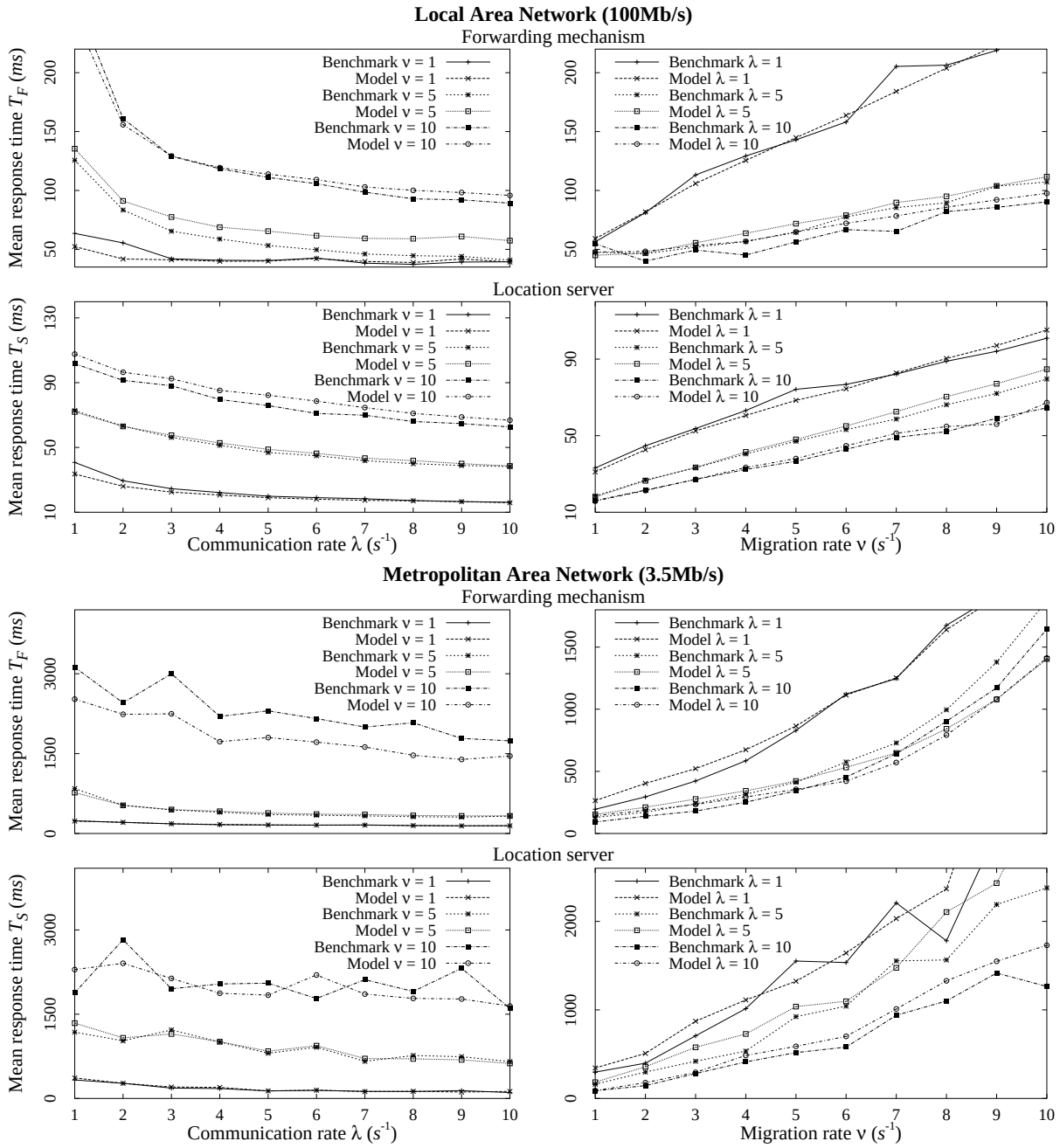

Figure 7: Validation with experiments on a LAN and a MAN

	$\delta(s^{-1})$	$\gamma(s^{-1})$	$\gamma_1(s^{-1})$	$\gamma_2(s^{-1})$	$\mu(s^{-1})$
LAN	10.9	45.6	115.6	76.3	2325
MAN	1.6	12.3	36.7	12.1	938

Table 4: Values used for theoretical comparison

Figure 8(a) shows that under LAN conditions (refer to 1st line in Table 4 for self-contained information), the centralized technique performs the best almost everywhere and that under MAN conditions (refer to line 2 in Table 4 for self-contained information), the forwarders technique performs the best almost everywhere. This is what we have observed in Section 5.2. However, what we could not foresee in the experiments is that for some migration rates

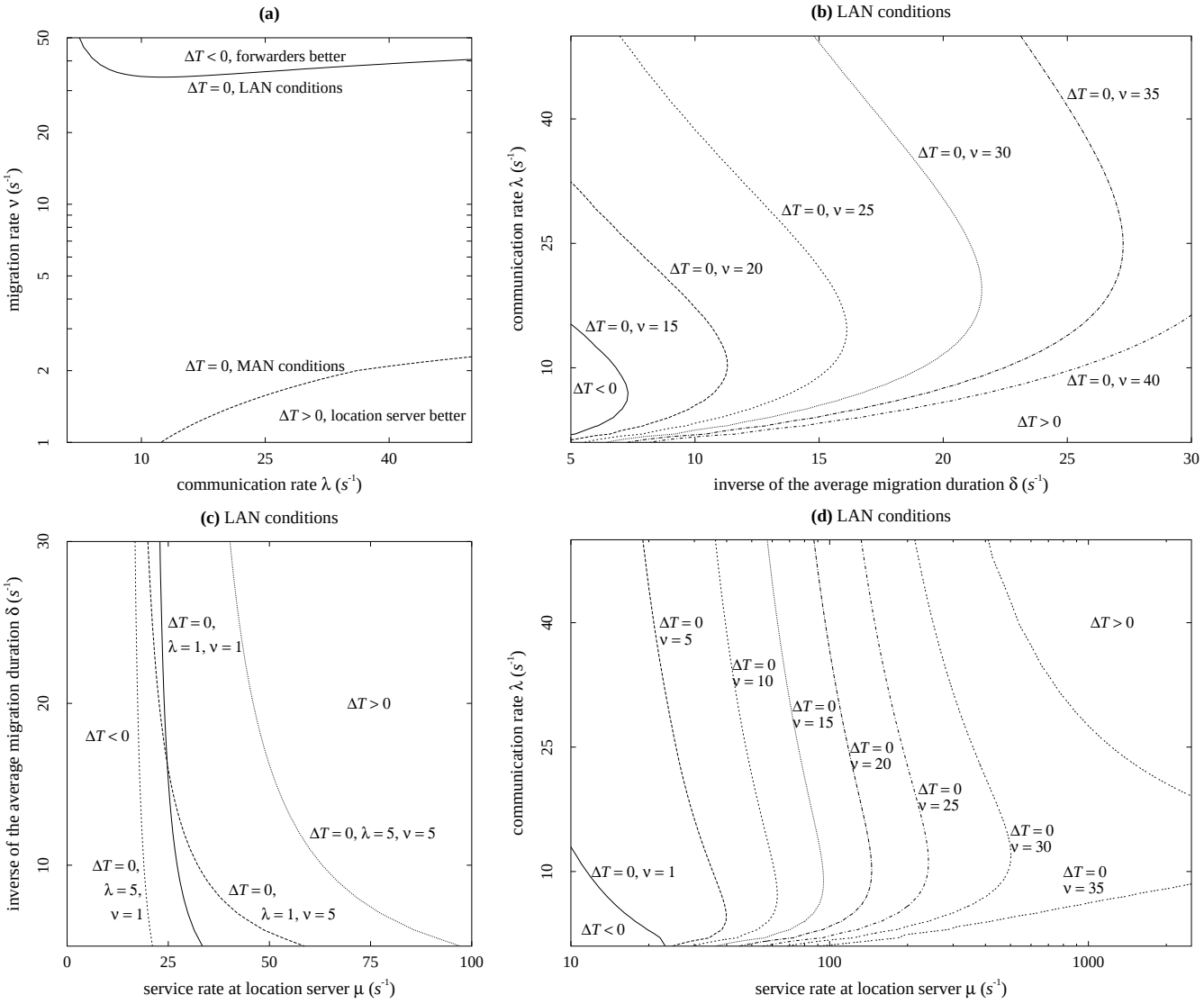

Figure 8: Sign of the difference between response times $\Delta T = T_F - T_S$.

(e.g. $\nu = 35$) the forwarders scheme is better only for intermediate values of the communication rate ($\lambda = 9 \ldots 19$ for instance).

From Figure 8(b) we can say that for each value of δ ($1/\delta$ is the expected duration of a migration) there exist values of the migration rate ν for which the forwarders scheme is better only for intermediate values for the communication rate λ. For instance, if $\delta = 10$ and $\nu = 20$, the centralized scheme outperforms the forwarders scheme for $\lambda \in [1, 5] \cap [18, 50]$ whereas the opposite conclusion holds when $\lambda \in [6, 17]$. This result holds under LAN conditions (travel times and service rate are given in Table 4).

In Figure 8(c), the communication latencies correspond to a 100Mb/s switched LAN. We observe that for extremely low service rate (μ less than $16s^{-1}$), the forwarders technique becomes better than the centralized technique. The frontier between the regions where each approach is better, i.e. the line $\Delta T = 0$, depends on the values of λ and ν. Surprisingly enough, increasing the communication rate while keeping unchanged the migration rate does not have the same effect on this frontier: for $\nu = 1$, an increase in λ from 1 to 5 shifts the line $\Delta T = 0$ to the left (lower service rate) whereas the same increase in λ but for $\nu = 5$ shifts the frontier to the right (higher service rate). (A shift to the right enlarges the region where forwarders are better and a shift to the left enlarges the region where

the server is better.) Notice that for applications generating a single couple source-agent, the operational conditions are far away from these frontiers ($\mu = 2325$ in our LAN experiments). However, when there are multiple sources and/or agents, the buffer at the server is completely partitioned between them, so that each couple would have only a fraction of the server processing speed (which is simply $\mu = 2325$). There should be around 80 active queues at the server in order to have service rates per queue as low as 30. It is then, and only then, that the performance of the server degrades till the point where forwarders may perform better.

Figure 8(d) displays the frontiers $\Delta T = 0$ for several values of the migration rate ($\nu \in \{1, 5, 10, 15, 20, 25, 30, 35\}$) and for $\mu \in [10, 2500]$ and $\lambda \in [1, 50]$; the travel times and the migration durations correspond to a LAN (values in Table 4). When ν increases the line $\Delta T = 0$ shifts to the right (higher service rate) enlarging the region where forwarders are better. Actually, as the migration rate increases, the performance of both approaches degrades: in the forwarders approach, the forwarding chain gets longer, thereby yielding larger communication times; in the centralized approach, the effect of such an increase is threefold. First, the probability that the recorded location at the source is no longer valid increases. Second, the probability of having a wrong location in the server's reply increases as well. Third, there will be much more `update location requests` generated. Since these requests have the highest priority, the service of `location requests` coming from the source will be delayed in time (see Section 4.1). Each of these effects will increase the communication time. As a result, when ν increases, the performance of the server degrades more rapidly than the performance of the forwarders. Notice that here also we observe the same effect as the one observed in Figures 8(a) and 8(b): for some values of the migration rate ν the forwarders perform better only for intermediate values of the communication rate λ. For instance, if $\mu = 2325$ and $\nu = 35$, the forwarders are better if $\lambda = 9 \ldots 19$.

To conclude this section, we would like to stress the fact that selecting the best location scheme is not as intuitive as it could look in the first place... Our study of the sign of ΔT has pointed out several unexpected effects when the value of some parameter is changed.

6 Concluding remarks

We have proposed simple Markovian analytical models for evaluating the performance of two approaches for locating mobile objects. One approach uses forwarders to enable communication between a source and a mobile object; in the other approach communications are ensured by a centralized server. Our models have been validated through simulations and extensive experiments on a LAN and on a MAN. In all the experiments that we have conducted, we have observed that the server yields the best performance on a LAN and the forwarders are more efficient on a MAN. Using the theoretical expected response times of both schemes, we have identified the best location scheme under a wide variety of conditions. Concerning future work, the obvious next step is to complete the validation of the model of the location server against applications with several sources/agents. In the case where our model does not scale well to large number of objects (sources and agents) we will consider developing an exact model that consists in seeing the server as a vacation server, polling among the different queues.

At last, note that our contribution in the field of code mobility is twofold. First, we have identified the best location scheme depending on the network conditions. Our conclusions are not that intuitive and they were made possible thanks to our rigorous approach. Second, we have shown that modeling such mechanisms is possible using rather simple techniques, thereby leaving the door open to the performance analysis of similar schemes.

A Proof of Proposition 3.1

We first derive (12) and (13). From (2) and (3) we obtain

$$\begin{aligned} p_{i,0,0} &= \left(\frac{\delta\nu}{(\lambda+\delta)(\lambda+\nu)}\right)^{i-1} p_{1,0,0} \\ p_{i,1,0} &= \left(\frac{\delta\nu}{(\lambda+\delta)(\lambda+\nu)}\right)^{i-1} p_{1,1,0} \end{aligned}$$

for $i \geq 1$, so that

$$\begin{aligned} f(z) &= \left[\frac{z\,(\lambda+\delta)(\lambda+\nu)}{(\lambda+\delta)(\lambda+\nu)-z\,\delta\nu}\right] p_{1,0,0} \\ g(z) &= \left[\frac{z\,(\lambda+\delta)(\lambda+\nu)}{(\lambda+\delta)(\lambda+\nu)-z\,\delta\nu}\right] p_{1,1,0}. \end{aligned}$$

Expressing $p_{1,1,0}$ in terms of $p_{1,0,0}$ by using (3) and $p_{1,0,0}$ in terms of $p_{1,0,1^*}$ and $p_{0,0,1}$ ($= p_{1,0,1}$ – see (8)) by using (1), we may rewrite $g(z)$ and $f(z)$ as follows:

$$f(z) = \frac{z\,\gamma(\lambda+\delta)}{(\lambda+\delta)(\lambda+\nu)-z\,\delta\nu}\,(p_{1,0,1}+p_{1,0,1^*}). \tag{39}$$

$$g(z) = \frac{z\,\gamma\nu}{(\lambda+\delta)(\lambda+\nu)-z\,\delta\nu}\,(p_{1,0,1}+p_{1,0,1^*}) \tag{40}$$

By noticing that (use (1), (7) and (8))

$$p_{1,0,1}+p_{1,0,1^*} = \frac{(\lambda+\nu)\,(\gamma+\nu)}{\lambda\gamma}\,p_{1,0,1^*} = \frac{(\lambda+\nu)\,(\gamma+\nu)}{\nu\,(\nu+\lambda+\gamma)}\,p_{1,0,1} \tag{41}$$

and plugging these values into (39)-(40) we find (12) and (13). Note in passing that the last identity in (41) gives

$$p_{1,0,1^*} = \frac{\lambda\gamma}{\nu\,(\nu+\lambda+\gamma)}\,p_{1,0,1} \tag{42}$$

which is nothing but (16).

We now address the computation of $h(z)$ and $k(z)$. From (4)-(10) we find

$$[z(\gamma+\nu)-\gamma]\,h(z) = z\nu[p_{1,0,1}-zp_{1,0,1^*}]-\gamma[p_{1,0,1}+z^2p_{1,0,1^*}]+z\lambda f(z)+z^2\delta k(z) \tag{43}$$

$$[z(\gamma+\delta)-\gamma]\,k(z) = \gamma(z-1)p_{0,1,1}+z\lambda g(z)+z\nu h(z)-z\nu[p_{1,0,1}-zp_{1,0,1^*}]. \tag{44}$$

Substituting $f(z)$ and $g(z)$ in (43)-(44) for their values found in (39) and (40), respectively, defines a linear system of two equations in the unknowns $h(z)$ and $k(z)$. Solving formally for this system of equations and using (42) yields (14) and (15). Observe from (14) and (15) that $h(z)$ and $k(z)$ are well-defined (i.e. analytic) for $|z| \leq 1$ as long as $b(z)$ does not vanish for $|z| \leq 1$. We will come back in a short while on this analyticity issue.

For the time being, let us consider the normalizing condition $F(1) = 1 = p_{1,0,1^*} + f(1) + g(1) + h(1) + k(1)$. With the help of equations (42) and (12)-(15) we see that the normalizing condition will be satisfied iff.

$$p_{0,1,1}+\frac{(\lambda+\nu)(\gamma+\nu)(\gamma+\lambda)}{\lambda\nu(\nu+\lambda+\gamma)}\,p_{1,0,1} = \frac{\delta\nu}{\delta+\nu}\left(\frac{1}{\nu}+\frac{1}{\delta}-\frac{1}{\gamma}\right)(1-p_{1,0,1}). \tag{45}$$

We conclude from (45) that the condition

$$\frac{1}{\gamma} < \frac{1}{\nu}+\frac{1}{\delta} \tag{46}$$

is necessary for the Markov process to be stable. Indeed, if $\frac{1}{\gamma} = \frac{1}{\nu}+\frac{1}{\delta}$ then (45) only holds if $p_{0,1,1} = p_{1,0,1} = 0$ and the Markov process is not ergodic on the state-space $\mathcal{E}$; if $\frac{1}{\gamma} > \frac{1}{\nu}+\frac{1}{\delta}$ then (45) cannot hold if $p_{0,1,1}, p_{1,0,1} > 0$ and again the Markov process is not ergodic on $\mathcal{E}$.

We now come back to the analyticity of $h(z)$ and $k(z)$ in the unit disk. It is easily seen that under condition (46) the polynomial $b(z)$ has exactly one zero $z = z_0$ in $|z| \leq 1$ given by

$$z_0 := \gamma\,\frac{\gamma+\nu+\delta-\sqrt{(\gamma+\nu+\delta)^2-4\nu\delta}}{2\nu\delta}$$

so that $h(z)$ (resp. $k(z)$) will be well-defined (i.e. analytic) at $z = z_0$ if the coefficient of $1/b(z)$ in (14) (resp. in (15)) vanishes at this point. This gives rise to two relations between $p_{0,1,1}$ and $p_{1,0,1}$. The reader will check that these two relations are actually identical and given by (using (15))

$$p_{1,0,1} = \frac{(\gamma-z_0\,(\gamma+\nu))\,(\nu+\lambda+\gamma)\,a(z_0)}{(\lambda+\nu)(\gamma+\nu)(\lambda\gamma+(\lambda+\delta)(\lambda+\nu))\,z_0^2}\,p_{0,1,1} \tag{47}$$

From (45) and (47) we get

$$p_{0,1,1} = \frac{\frac{\delta\nu}{\delta+\nu}\left(\frac{1}{\nu}+\frac{1}{\delta}-\frac{1}{\gamma}\right)}{1+\left[\frac{\delta\nu}{\delta+\nu}\left(\frac{1}{\nu}+\frac{1}{\delta}-\frac{1}{\gamma}\right)+\frac{(\lambda+\nu)(\gamma+\nu)(\gamma+\lambda)}{\lambda\nu(\nu+\lambda+\gamma)}\right]\left[\frac{(\gamma-z_0(\gamma+\nu))(\nu+\lambda+\gamma)a(z_0)}{(\lambda+\nu)(\gamma+\nu)(\lambda\gamma+(\lambda+\delta)(\lambda+\nu))z_0^2}\right]} \quad (48)$$

$$p_{1,0,1} = \frac{\frac{\delta\nu}{\delta+\nu}\left(\frac{1}{\nu}+\frac{1}{\delta}-\frac{1}{\gamma}\right)\left(\frac{(\gamma-z_0(\gamma+\nu))(\nu+\lambda+\gamma)a(z_0)}{(\lambda+\nu)(\gamma+\nu)(\lambda\gamma+(\lambda+\delta)(\lambda+\nu))z_0^2}\right)}{1+\left[\frac{\delta\nu}{\delta+\nu}\left(\frac{1}{\nu}+\frac{1}{\delta}-\frac{1}{\gamma}\right)+\frac{(\lambda+\nu)(\gamma+\nu)(\gamma+\lambda)}{\lambda\nu(\nu+\lambda+\gamma)}\right]\left[\frac{(\gamma-z_0(\gamma+\nu))(\nu+\lambda+\gamma)a(z_0)}{(\lambda+\nu)(\gamma+\nu)(\lambda\gamma+(\lambda+\delta)(\lambda+\nu))z_0^2}\right]} \quad (49)$$

which completes the calculation of $F(z)$. Notice that $p_{0,1,1}$ and $p_{1,0,1}$ are strictly positive due to (46) and to the fact that $b(\gamma/(\gamma+\nu)) < 0$ which in turn implies that $z_0 < \gamma/(\gamma+\nu)$.

In summary, we have shown that $F(z)$, as given in Proposition 3.1, is analytic in $|z| < 1$, continuous in $|z| \leq 1$ and satisfies the condition $F(1) = 1$ if (46) holds. We can actually find $\epsilon > 0$ such that $F(z)$ is analytic in $|z| < 1+\epsilon$ (if z_1 and z_2 denote the zeros of $b(z)$ and $a(z)$, respectively, in $|z| > 1$, then $\epsilon = \min(|z_1|, |z_2|) - 1$) we may invoke a classical result on z-transform [18] to conclude that (46) is the stability condition and that $F(z)$ is the z-transform of the stationary probabilities.

B Derivation of equations (32)-(35)

We want to compute $q_S(v)$, the probability that a transmission is initiated in state $v \in \mathcal{V}$, where

$$\mathcal{V} := \{(A,0^*,1),(A,1^*,1),(B,0^*,1),(B,1^*,1)\}.$$

Let $p(w;v)$ be the probability that a communication is initiated in state $v \in \mathcal{V}$ given that the system is in state $w \in \{(A,0^*,0),(A,1^*,0),(B,0^*,0),(B,1^*,0)\}$.

We see from Figure 6 that

$$q_S(v) = \frac{\delta\nu}{(\lambda+\delta)(\lambda+\nu)}\,P((B,0^*,0);v) \quad (50)$$

for $v \in \mathcal{V}$. It remains to compute $P((B,0^*,0);v)$ for $v \in \mathcal{V}$.

The following linear relations readily derive from Figure 6:

$$\begin{aligned}
p((A,0^*,0);v) &= \frac{\lambda}{\lambda+\nu}\,p((A,0^*,1);v) + \frac{\nu}{\lambda+\nu}\,p((A,1^*,0);v)\\
p((A,1^*,0);v) &= \frac{\lambda}{\lambda+\delta}\,p((A,1^*,1);v) + \frac{\delta}{\lambda+\delta}\,p((B,0^*,0);v)\\
p((B,0^*,0);v) &= \frac{\lambda}{\lambda+\nu+\mu}\,p((B,0^*,1);v) + \frac{\nu}{\lambda+\nu+\mu}\,p((B,1^*,0);v) + \frac{\mu}{\lambda+\nu+\mu}\,p((A,0^*,0);v)\\
p((B,1^*,0);v) &= \frac{\lambda}{\lambda+\delta+\mu}\,p((B,1^*,1);v) + \frac{\delta}{\lambda+\delta+\mu}\,p((B,0^*,0);v) + \frac{\mu}{\lambda+\delta+\mu}\,p((A,1^*,0);v)
\end{aligned}$$

for all $v \in \mathcal{V}$. Rearranging these equations give, in matrix form,

$$\begin{bmatrix} (\lambda+\nu) & -\nu & 0 & 0\\ 0 & (\lambda+\delta) & -\delta & 0\\ -\mu & 0 & (\lambda+\nu+\mu) & -\nu\\ 0 & -\mu & -\delta & (\lambda+\delta+\mu) \end{bmatrix} \begin{bmatrix} p((A,0^*,0);v)\\ p((A,1^*,0);v)\\ p((B,0^*,0);v)\\ p((B,1^*,0);v) \end{bmatrix} = \lambda \begin{bmatrix} p((A,0^*,1);v)\\ p((A,1^*,1);v)\\ p((B,0^*,1);v)\\ p((B,1^*,1);v) \end{bmatrix} \quad (51)$$

for all $v \in \mathcal{V}$. All terms in the r.h.s. of (51) are either equal to 0 or to 1, as shown in Table 5 below (lines 1-4). From lines 1-4 in Table 5 and (51), we can compute $p((B,0^*,0),v)$ for all $v \in \mathcal{V}$ (see last line of Table 5). It now remains to plug these values of $p((B,0^*,0);v)$ into (50) to get (32)-(35).

v	$(A,0^*,1)$	$(A,1^*,1)$	$(B,0^*,1)$	$(B,1^*,1)$
$p((A,0^*,1);v)$	1	0	0	0
$p((A,1^*,1);v)$	0	1	0	0
$p((B,0^*,1);v)$	0	0	1	0
$p((B,1^*,1);v)$	0	0	0	1
$p((B,0^*,0);v)$	$\frac{(\lambda+\delta)\mu}{K}$	$\frac{(2\lambda+\delta+\mu+\nu)\nu\mu}{(\mu+\lambda+\delta)K}$	$\frac{(\lambda+\nu)(\lambda+\delta)}{K}$	$\frac{(\lambda+\nu)(\lambda+\delta)\nu}{(\mu+\lambda+\delta)K}$

Table 5: (with $K := (\lambda+\nu+\mu)(\lambda+\delta+\nu)$)

References

[1] Aglets Software Development Kit. IBM, 1999. http://www.trl.ibm.com/aglets/download.html.

[2] A. Bakker, E. Amade, G. Ballintijn, I. Kuz, P. Verkaik, I. van der Wijk, M. van Steen, and A. S. Tanenbaum. The globe distribution network. In *Proc. of USENIX 2000 (FREENIX Track), San Diego, California*, pages 141–152, June 2000.

[3] F. Baude, D. Caromel, F. Huet, and J. Vayssière. Objets actifs mobiles et communicants. *Technique et Science Informatique*, 21(6), 2002.

[4] J. Baumann. *Control algorithms for mobile agents.* PhD thesis, University of Stuttgart, IPVR Department, 1999.

[5] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. of MobiCom'98, Dallas, Texas*, pages 85–97. ACM Press, October 1998.

[6] D. M. Chess. Security issues in mobile code system. In *Lecture Notes in Computer Science 1419*, pages 1–14. Springer-Verlag, 1998.

[7] D. M. Chess, C. G. Harrison, and A. Kershenbaum. Mobile agents: Are they a good idea? Technical report, IBM T.J. Watson Research Division, March 1995.

[8] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Lecture Notes in Computer Science 2009*, pages 46–66. Springer-Verlag, 2001.

[9] S. R. Das, C. E. Perkins, and E. M. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In *Proc. of INFOCOM 2000, Tel-Aviv, Israel*, volume 1, pages 3–12, March 2000.

[10] R. J. Fowler. The complexity of using forwarding addresses for decentralized object finding. In *Proc. of PODC'86, New York, New York*, pages 108–120. ACM Press, August 1986.

[11] J. Gosling, B. Joy, and G. Steele. *The Java Language Specification.* GOTOP Information Inc., 1996.

[12] R. S. Gray. Agent Tcl: A flexible and secure mobile agent system. In *Proc. of the 4th Annual Tcl/Tk Workshop, Monterey, California*, pages 9–23, July 1996.

[13] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Scenario-based performance analysis of routing protocols for mobile ad-hoc networks. In *Proc. of MobiCom'99, Seattle, Washington*, pages 195–206. ACM Press, August 1999.

[14] G. Karjoth, D. B. Lange, and M. Oshima. A security model for Aglets. *IEEE Internet Computing*, 1(4):68–77, July-August 1997.

[15] J. Kiniry and D. Zimmerman. A hands-on look at Java mobile agents. *IEEE Internet Computing*, 1(4):21–30, July-August 1997.

[16] L. Kleinrock. *Queueing Systems: Theory*, volume 1. John Wiley and Sons, 1975.

[17] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Proc. of ASPLOS 2000, Boston, Massachusetts*, pages 190–201, November 2000.

[18] V. A. Malyshev. An analytical method in the theory of two-dimensional positive random walks. *Mathematicheskii Zhurnal*, 13(6):1314–1329, 1972.

[19] D. S. Milojičić, W. LaForge, and D. Chauhan. Mobile Objects and Agents (MOA). In *Proc. of USENIX COOTS'98, Santa Fe, New Mexico*, April 1998.

[20] P. Mockapetris and K. J. Dunlap. Development of the Domain Name System. In *Proc. of SIGCOMM'88, Stanford, California*, pages 123–133. ACM Press, August 1988.

[21] J. Norris. *Markov chains*. Cambridge University Press, 1997.

[22] M. Nuttall. A brief summary of systems providing process or object migration facilities. *Operating Systems Review*, 28(4):64–80, October 1994.

[23] M. L. Powell and B. P. Miller. Process migration in DEMOS/MP. In *Proc. of SOSP'83, Bretton Woods, New Hampshire*, October 1983. Published as Operating Systems Review, 17(5):110-119.

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

[25] S. Ramanathan and M. Steenstrup. A survey of routing techniques for mobile communications networks. *Mobile Networks and Applications*, 1(2):89–104, October 1996.

[26] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of SIGCOMM'01, San Diego, California*, pages 149–160. ACM Press, August 2001.

[27] T. Thorn. Programming languages for mobile code. *ACM Computing Surveys*, 29(3):213–239, Sept. 1997.

[28] Voyager. ObjectSpace, Inc., 1999. http://www.objectspace.com.