Chapter 3

Analysis of two agent location mechanisms in a mobile environment

This chapter is devoted to the performance evaluation and comparison of two approaches for locating an agent 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. This chapter builds on Markov chain analysis to evaluate the cost of communication in the presence of migration. It undertakes validation of the proposed models by comparing theoretical results first with simulations and then with experimental results. Our research revealed that neither approach is uniformly best than the other as their respective performance depends on the system parameters. The analysis developed in this chapter can be used to select the best scheme based on its average response time. Designers and programmers can benefit from this research by implementing the best mechanism which minimizes communication times between components of the application at hand.

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

Note: Part of the material presented in this chapter is published in [13, 14, 15].

3.1 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 of the operations locally. This simple idea has led to a new paradigm called *code-mobility* [117]. In this paradigm, a mobile object – sometimes called an agent – is given a list of destinations and a series of operations to perform at 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. searching for the lowest fare on many different sites) to data mining [33]. Mobility can be implemented as a service provided by an operating system [92]; 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 [5, 19, 60, 74, 84, 101, 121].

Any mobility mechanism must first provide a way to migrate code from one host to another. It must also ensure that any communication between objects will not be impaired by migration, 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 mobile 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 [85] or mobile as in wireless [102] and ad-hoc networks [25, 42, 69], or more recently in peer-to-peer networking [18, 38, 76, 113].

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 agents, whereas the second one relies on special objects – called *forwarders* – whose role is to forward a message to the agents. A more careful description of these approaches will be given later on.

Mobility raises several concerns, among them security [32, 71] (of both the mobile agent and the host sheltering it) and performance issues [20, 55]. In [55] the complexity of using forwarding addresses is extensively studied whereas [20] addresses fault-tolerance properties in forwarder-based mechanisms. In this chapter we will only focus on performance issues and, more specifically, on the cost of communication in the presence of migration. To the best of our knowledge, this is the first time that such an analysis is performed. In [84] 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 [20].

In this work we develop Markovian models of the forwarders and of the location server as implemented in *ProActive* [101], a Java library that provides all of 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 chapter is organized as follows: preliminary definitions and notation are introduced in Section 3.2; the forwarding mechanism is presented and evaluated in Section 3.3 and the centralized server mechanism is investigated in Section 3.4. Simulations and experimental results are reported in Section 3.5 as well as a theoretical comparison between both approaches. Extension of our work to more general cases are discussed in Section 3.6.

3.2 Definitions and Notation

In this section we introduce several random variables (RVs) that we will use to construct our models. Throughout the chapter a mobile object will indifferently be called a mobile agent or simply an agent. The following notation is related ti a given source-agent pair.

The ith 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 (ith) *communication time* – is a scheme-dependent quantity that will be defined later 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 jth migration of the mobile agent occurs at time $m_j > 0$ and it requires the mobile agent p_j units of times to reach its new location. During a migration period the agent is unreachable. The mobile agent 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 agent does not migrate in $[0, m_1)$ (see Figure 3.1).

Both the source and the agent exhibit a two state process: the source alternates between an "idle" state and a "communicating" state, and the agent alternates between an "idle" state and a "migrating" state.

The following assumptions will be enforced throughout the chapter:

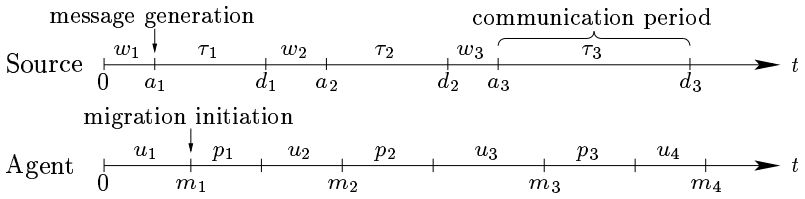

Figure 3.1: A time diagram including all RVs relative to the source and to the agent

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.

3.3 The forwarders

3.3.1 Description

Forwarding techniques were first introduced in distributed operating systems like DEMOS/MP [99] 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. Upon a later migration, a new forwarding reference, pointing toward the new location, is created, and the previously created forwarding reference is now pointing to the new one. Thus, as the system runs, *chains of forwarders* are built (as illustrated in Figure 3.2, step 1). 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 agent, 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.

An illustration of the short-cutting feature is given in Figure 3.2: 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 maintain the same semantic as that of a static program (i.e. with no mobile agent) one has to introduce constraints. In particular, communications through a chain of forwarders should be synchronous, i.e. the caller remains blocked during the communication phase. With these assumptions we can now describe the protocol in use:

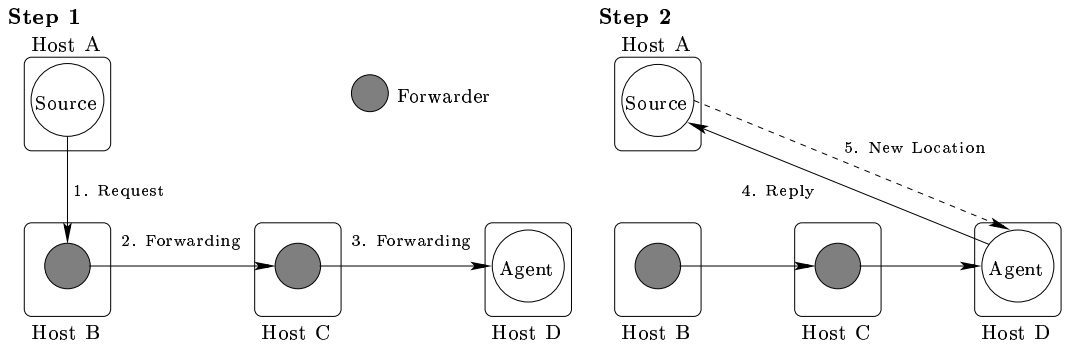

Figure 3.2: The short-cutting feature in the forwarding mechanism

- Upon migration, a mobile agent leaves a forwarder on the current site;
- This forwarder is linked to the mobile agent on the remote host;
- No communication can occur while the mobile agent is migrating;
- Every forwarder is exclusive to the agent that had created it;
- When receiving a message, the forwarder sends it to the next hop (possibly the mobile agent);
- Any successful communication places the mobile agent one hop away from the caller (e.g. after Step 2 in Figure 3.2 the agent is located one hop away from the source).

The above protocol is implemented in various Java libraries (*MOA* [84], *ProActive* [101] and *Voyager* [121]) except for the short-cutting feature that is triggered by the agent at the end of the message processing.

3.3.2 A Markovian analysis of the forwarders

In this section we will evaluate the performance (response time and number of forwarders in Sections 3.3.2.1 and 3.3.2.2 respectively) of the mechanism introduced in Section 3.3.1 through a Markovian analysis. We will assume that a mobile agent 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 3.2). 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 if the number of hosts is infinite. Observe that a forwarder is used only once by a given source, because of the short-cutting that takes place at the end of a successful communication.

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 dynamics is not affected by other objects; there will be as many chains as there are source-agent pairs. 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 does not place any assumption on the number of objects (source or agent) in the application.

From the description given in Section 3.3.1, it can be seen that at any time the system is in one of the following states (see Figure 3.3):

- 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 has 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

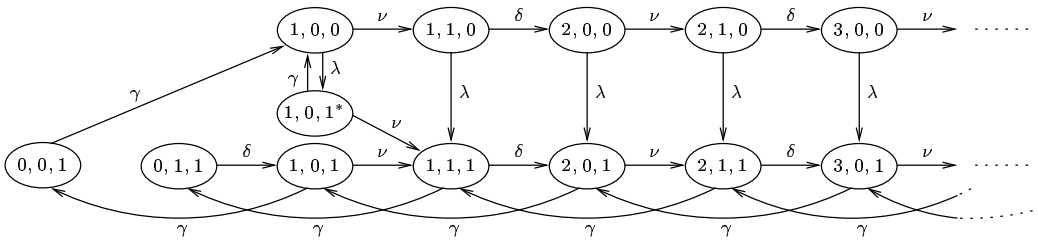

Figure 3.3: System states and transition rates in the forwarding mechanism

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 3.2,

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 3.3: 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 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).

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

[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 3.5.

unique strictly positive solution of the Chapman–Kolmogorov (C-K) equations [75]

$$
\begin{aligned}
(\lambda+\nu)\,p_{1,0,0} &= \gamma\,(p_{0,0,1}+p_{1,0,1^*}) && && (3.1)\\
(\lambda+\nu)\,p_{i,0,0} &= \delta\,p_{i-1,1,0} && i=2,3,\ldots && (3.2)\\
(\lambda+\delta)\,p_{i,1,0} &= \nu\,p_{i,0,0} && i=1,2,\ldots && (3.3)\\
\delta\,p_{0,1,1} &= \gamma\,p_{1,1,1} && && (3.4)\\
(\delta+\gamma)\,p_{1,1,1} &= \nu\,(p_{1,0,1}+p_{1,0,1^*})+\lambda\,p_{1,1,0}+\gamma\,p_{2,1,1} && && (3.5)\\
(\delta+\gamma)\,p_{i,1,1} &= \nu\,p_{i,0,1}+\lambda\,p_{i,1,0}+\gamma\,p_{i+1,1,1} && i=2,3,\ldots && (3.6)\\
(\nu+\gamma)\,p_{1,0,1^*} &= \lambda\,p_{1,0,0} && && (3.7)\\
p_{0,0,1} &= p_{1,0,1} && && (3.8)\\
(\nu+\gamma)\,p_{1,0,1} &= \delta\,p_{0,1,1}+\gamma\,p_{2,0,1} && && (3.9)\\
(\nu+\gamma)\,p_{i,0,1} &= \delta\,p_{i-1,1,1}+\lambda\,p_{i,0,0}+\gamma\,p_{i+1,0,1} && i=2,3,\ldots && (3.10)
\end{aligned}
$$

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

We will not try to solve Equations (3.1)–(3.10) 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 $|z|\le 1$

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

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 $|x|\le 1, |y|\le 1$ and $|z|\le 1$

$$
\begin{aligned}
F(x,y,z) &:= \sum_{(i,j,k)\in\mathcal{E}} z^i x^j y^k p_{i,j,k}\\
&= \sum_{i=1}^{\infty} z^i p_{i,0,0}+x\sum_{i=1}^{\infty} z^i p_{i,1,0}+y\sum_{i=0}^{\infty} z^i p_{i,0,1}+xy\sum_{i=0}^{\infty} z^i p_{i,1,1}+zy^2\,p_{1,0,1^*}\\
&= f(z)+x\,g(z)+y\,h(z)+xy\,k(z)+zy^2\,p_{1,0,1^*}
\end{aligned}
$$

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

Proposition 3.3.1 (z-transform of the Markov process)

The Markov process depicted in Figure 3.3 is ergodic if and only if

$$
\frac{1}{\gamma}<\frac{1}{\nu}+\frac{1}{\delta}. \qquad (3.11)
$$

In steady-state, the z-transform $F(x,y,z)$ is given by

$$f(z) = \frac{z\,\gamma(\lambda+\delta)(\lambda+\nu)(\gamma+\nu)}{\nu(\nu+\lambda+\gamma)\,a(z)}\,p_{1,0.1} \tag{3.12}$$

$$g(z) = \frac{z\,\gamma(\lambda+\nu)(\gamma+\nu)}{(\nu+\lambda+\gamma)\,a(z)}\,p_{1,0,1} \tag{3.13}$$

$$h(z) = -\frac{z^2\,\delta\gamma}{b(z)}\,p_{0,1,1} + \left[\frac{\delta(\nu-\gamma)z^2 - \gamma(\nu+\delta)z + \gamma^2}{b(z)} - \frac{z^3\delta\gamma[\lambda\,a(z) + (\gamma+\nu)(\lambda\gamma + (\lambda+\delta)(\lambda+\nu))]}{(\nu+\lambda+\gamma)\,a(z)\,b(z)}\right]p_{1,0,1} \tag{3.14}$$

$$k(z) = \frac{\gamma\,(\gamma - z\,(\gamma+\nu))}{b(z)}\,p_{0,1,1} - \frac{z^2\gamma(\lambda+\nu)(\gamma+\nu)(\lambda\gamma + (\lambda+\delta)(\lambda+\nu))}{(\nu+\lambda+\gamma)\,a(z)\,b(z)}\,p_{1,0,1} \tag{3.15}$$

$$p_{1,0,1^*} = \frac{\lambda\gamma}{\nu\,(\nu+\lambda+\gamma)}\,p_{1,0,1} \tag{3.16}$$

$$p_{0,1,1} = \frac{1 - \delta\nu/(\gamma(\delta+\nu))}{1 + [1 - \delta\nu/(\gamma(\delta+\nu)) + (\lambda+\nu)(\gamma+\nu)(\gamma+\lambda)/(\lambda\nu(\nu+\lambda+\gamma))]\,c(z_0)} \tag{3.17}$$

$$p_{1,0,1} = \frac{[1 - \delta\nu/(\gamma(\delta+\nu))]c(z_0)}{1 + [1 - \delta\nu/(\gamma(\delta+\nu)) + (\lambda+\nu)(\gamma+\nu)(\gamma+\lambda)/(\lambda\nu(\nu+\lambda+\gamma))]\,c(z_0)} \tag{3.18}$$

for $|z| \le 1$, with

$$\begin{aligned}
a(z) &:= -\delta\nu z + (\lambda+\delta)(\lambda+\nu),\\
b(z) &:= \delta\nu z^2 - \gamma(\gamma+\nu+\delta)z + \gamma^2,\\
c(z) &:= \frac{(\gamma - z\,(\gamma+\nu))\,(\nu+\lambda+\gamma)\,a(z)}{((\lambda+\nu)(\gamma+\nu)(\lambda\gamma + (\lambda+\delta)(\lambda+\nu))\,z^2)},\\
z_0 &= \gamma\left(\gamma+\nu+\delta - \sqrt{(\gamma+\nu+\delta)^2 - 4\nu\delta}\right)/(2\nu\delta).
\end{aligned}$$

♦

Proof. We first derive (3.12) and (3.13). From (3.2) and (3.3) we obtain

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

for $i \ge 1$, so that

$$f(z) = \left[\frac{z\,(\lambda+\delta)(\lambda+\nu)}{(\lambda+\delta)(\lambda+\nu) - z\,\delta\nu}\right]p_{1,0,0}, \qquad g(z) = \left[\frac{z\,\nu(\lambda+\nu)}{(\lambda+\delta)(\lambda+\nu) - z\,\delta\nu}\right]p_{1,0,0}. \tag{3.19}$$

From (3.1) and (3.7)–(3.8) we find

$$p_{1,0,0} = \frac{\gamma(\gamma+\nu)}{\nu(\nu+\lambda+\gamma)}\, p_{1,0,1}. \tag{3.20}$$

Introducing (3.20) into (3.19) gives (3.12) and (3.13). On the other hand, we find (3.16) from (3.20) and (3.7).

We now address the computation of $h(z)$ and $k(z)$. From (3.4)–(3.6) and (3.8)–(3.10) we find

$$[z(\gamma+\nu)-\gamma]\, h(z) = z\nu[p_{1,0,1} - zp_{1,0,1^*}] - \gamma[p_{1,0,1} + z^2 p_{1,0,1^*}] + z\lambda f(z) + z^2\delta k(z) \tag{3.21}$$

$$[z(\gamma+\delta)-\gamma]\, k(z) = \gamma(z-1)p_{0,1,1} + z\lambda g(z) + z\nu h(z) - z\nu[p_{1,0,1} - zp_{1,0,1^*}]. \tag{3.22}$$

Substituting $f(z)$, $g(z)$ and $p_{1,0,1^*}$ into (3.21)–(3.22) for their values found in (3.12), (3.13) and (3.16), respectively, defines a linear system of two equations in the unknowns $h(z)$ and $k(z)$. Solving formally for this system of equations yields (3.14) and (3.15). Observe from (3.14) and (3.15) 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 $F(1,1,1) = 1 = f(1) + g(1) + h(1) + k(1) + p_{1,0,1^*}$. With the help of equations (3.16) and (3.12)–(3.15) we see that the normalizing condition will be satisfied iff.

$$p_{0,1,1} + \frac{(\lambda+\nu)(\gamma+\nu)(\gamma+\lambda)}{\lambda\nu(\nu+\lambda+\gamma)}\, p_{1,0,1} = \frac{\delta\nu}{\delta+\nu}\left(\frac{1}{\nu}+\frac{1}{\delta}-\frac{1}{\gamma}\right)(1-p_{1,0,1}). \tag{3.23}$$

We conclude from (3.23) that the condition

$$\frac{1}{\gamma} < \frac{1}{\nu}+\frac{1}{\delta}, \text{ or equivalently } 1 - \frac{\delta\nu}{\gamma(\delta+\nu)} > 0,$$

is necessary for the Markov process to be stable. Indeed, if $\frac{1}{\gamma} = \frac{1}{\nu}+\frac{1}{\delta}$ then (3.23) 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 (3.23) 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 (3.11) 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 (3.14) (resp. in (3.15)) vanishes at this point. This gives rise to two relations

between $p_{0,1,1}$ and $p_{1,0,1}$ that are actually identical and given by (using (3.15))

$$p_{1,0,1} = c(z_0)\, p_{0,1,1}, \quad \text{with } c(z) = \frac{(\gamma - z\,(\gamma+\nu))\,(\nu+\lambda+\gamma)\,a(z)}{(\lambda+\nu)(\gamma+\nu)(\lambda\gamma+(\lambda+\delta)(\lambda+\nu))\,z^2}. \tag{3.24}$$

Solving for the system of linear equations (3.23) and (3.24) in the unknowns $p_{0,1,1}$ and $p_{1,0,1}$ yields (3.17) and (3.18), which completes the calculation of $F(x,y,z)$. Notice that $p_{0,1,1}$ and $p_{1,0,1}$ are strictly positive due to (3.11) 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 $F(x,y,z)$, as given in Proposition 3.3.1, is analytic in $|z| < 1$ for any value of x and y, continuous in $|z| \leq 1$ for any value of x and y and satisfies the condition $F(1,1,1) = 1$ if (3.11) holds. We can actually find $\epsilon > 0$ such that $F(x,y,z)$ is analytic in $|z| < 1+\epsilon$ for any value of x and y (if z_1 and z_2 denote the zeros of $b(z)$ and $a(z)$, respectively, in $|z| > 1$, then $\epsilon = \min(|z_1|, |z_2|) - 1$). Therefore, we may invoke a classical result on z-transform [81] to conclude that (3.11) is the stability condition and that $F(x,y,z)$ is the z-transform of the stationary probabilities. Note that $\frac{1}{\gamma} < \frac{1}{\nu} + \frac{1}{\delta}$ is not only a necessary condition for stability, but also a sufficient condition since, when it holds, one can find a unique strictly positive and normalized solution to the C-K equations, that is given by the coefficients of the z-transform $F(x,y,z)$. ■

3.3.2.1 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 agent, to which we must add the time it takes to the mobile agent 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 agent after exactly one hop then there is no need for the mobile agent to send its position to the source since the source knows it; in this case, the communication terminates once the message has reached the mobile agent, 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 3.3 we find that

$$q_F(1,0,1^*) = \frac{\lambda}{\lambda+\nu} \tag{3.25}$$

$$q_F(i,0,1) = \frac{\lambda(\lambda+\delta)}{\delta\nu}\,r^i \quad i = 2,3,\ldots \tag{3.26}$$

$$q_F(i,1,1) = \frac{\lambda}{\delta}\,r^i \quad i = 1,2,\ldots \tag{3.27}$$

where $r := \dfrac{\delta\nu}{(\lambda+\delta)(\lambda+\nu)} < 1$. By using (3.25)–(3.27) we may rewrite T_F as follows

$$T_F = \frac{\lambda}{\lambda+\nu}\,T_{1,0,1^*} + \frac{\lambda(\lambda+\delta)}{\delta\nu}\,\sum_{i=2}^{\infty} r^i\,T_{i,0,1} + \frac{\lambda}{\delta}\,\sum_{i=1}^{\infty} r^i\,T_{i,1,1}. \tag{3.28}$$

With the definitions $G(z) := \sum_{i=0}^{\infty} z^i\,T_{i,0,1}$ and $H(z) := \sum_{i=0}^{\infty} z^i\,T_{i,1,1}$, (3.28) becomes

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

We now need 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 3.3:

$$T_{1,0,1^*} = \frac{1}{\nu+\gamma} + \frac{\nu}{\nu+\gamma}\,T_{1,1,1}$$

$$T_{0,0,1} = \frac{1}{\gamma}$$

$$T_{i,0,1} = \frac{1}{\nu+\gamma} + \frac{\nu}{\nu+\gamma}\,T_{i,1,1} + \frac{\gamma}{\nu+\gamma}\,T_{i-1,0,1} \qquad i = 1,2,\ldots$$

$$T_{0,1,1} = \frac{1}{\delta} + T_{1,0,1}$$

$$T_{i,1,1} = \frac{1}{\delta+\gamma} + \frac{\delta}{\delta+\gamma}\,T_{i+1,0,1} + \frac{\gamma}{\delta+\gamma}\,T_{i-1,1,1} \quad i = 1,2,\ldots$$

which yields

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

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

$$G(z) = \frac{1}{D(z)}\left(\frac{(\delta+\nu)\,z}{1-z} + \frac{\nu}{\gamma}(1-z)(\gamma z - \delta) + \gamma z + \nu(\gamma z - \delta)\,zT_{0,1,1}\right) \tag{3.29}$$
$$H(z) = \frac{1}{D(z)}\left(\frac{(\delta+\nu)\,z}{1-z} + (\gamma+\delta)\,z - (\gamma^2\,z^2 - \gamma(\gamma+\nu)\,z + \delta\nu)T_{0,1,1}\right) \tag{3.30}$$

where we have defined

$$D(z) := (z-1)\left(\gamma^2\,z^2 - \gamma(\gamma+\nu+\delta)\,z + \delta\nu\right). \tag{3.31}$$

Using (3.29)–(3.31) and after some algebraic manipulations, we find (use $T_{1,0,1^*} = T_{0,1,1} - 1/\delta - 1/(\gamma+\nu)$)

$$\begin{aligned} T_F \;=\; & \frac{1}{\alpha(\lambda)}\Bigg[\left((\lambda+\delta)(\lambda+\nu) - \gamma\nu\right)T_{0,1,1} \\ & - \frac{(\lambda+\delta)(\lambda+\nu+\delta)}{\delta} - \frac{(\lambda+\delta)(\lambda+\nu)(\lambda(\lambda+\nu)+\nu(\gamma+\nu)) + \delta\gamma\nu\lambda}{\lambda\,(\lambda+\nu)(\gamma+\nu)}\Bigg] \end{aligned} \tag{3.32}$$

where $\alpha(\lambda) := (\lambda+\delta)(\lambda+\nu) - \gamma(\lambda+\nu+\delta)$. It remains to identify the constant $T_{0,1,1}$ in (3.32). 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 (3.32) 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 (3.32) 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

$$T_F = \begin{cases} \dfrac{1}{\alpha(\lambda)}\left[((\lambda+\delta)(\lambda+\nu)-\gamma\nu)\,T_{0,1,1} - \dfrac{(\lambda+\delta)(\lambda+\nu+\delta)}{\delta}\right. & \\ \qquad \left. - \dfrac{(\lambda+\delta)(\lambda+\nu)(\lambda(\lambda+\nu)+\nu(\gamma+\nu))+\delta\gamma\nu\lambda}{\lambda(\lambda+\nu)(\gamma+\nu)}\right] & \text{for } \lambda \neq \lambda_0 \\ \dfrac{(2\lambda+\nu+\delta)(2\gamma(\gamma+\nu)(\lambda+\nu+\delta)+\delta\lambda(\lambda+\nu)((\gamma+\nu)\,T_{0,1,1}-1))}{\lambda\delta(2\lambda+\nu+\delta-\gamma)(\lambda+\nu)(\gamma+\nu)} & \\ \qquad - \dfrac{(2\lambda+\nu)(\lambda+\delta)(\lambda+\nu-\gamma(\gamma+\nu)\,T_{0,1,1})+\gamma\nu\delta}{\lambda(2\lambda+\nu+\delta-\gamma)(\lambda+\nu)(\gamma+\nu)} & \text{for } \lambda = \lambda_0 \end{cases} \tag{3.33}$$

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

3.3.2.2 The expected number of forwarders

In this section we compute the expected number of forwarders in steady-state between the agent and the source. Let $q(i)$ be the probability that the agent is located $i \geq 1$ hops away from the source, which corresponds to a situation where there are exactly $i-1$ forwarders in the system. Clearly,

$$q(i) = p_{i,0,0} + p_{i,1,0} + p_{i,0,1} + p_{i,1,1} + \mathbf{1}\{i=1\}p_{1,0,1^*}, \qquad i = 1, 2, \ldots$$

and the expected number N_s of forwarders (the subscript s refers to "source", which is the starting point of the count of the number of forwarders) is given by

$$\begin{aligned} N_s &= \sum_{i=1}^{\infty}(i-1)\,q(i) = \sum_{i=1}^{\infty} i\,q(i) - \sum_{i=1}^{\infty} q(i) \\ &= f'(1) + g'(1) + h'(1) + k'(1) + p_{1,0,1^*} - (1 - p_{0,0,1} - p_{0,1,1}) \end{aligned} \tag{3.34}$$

where $\phi'(1)$ denotes the derivative of $\phi(z)$ at point $z=1$. From (3.12) and (3.13) we find

$$f'(1) + g'(1) = \frac{\gamma(\lambda+\delta)(\lambda+\nu)^2(\gamma+\nu)}{\lambda^2\nu(\nu+\lambda+\gamma)(\lambda+\delta+\nu)}\,p_{1,0,1} \tag{3.35}$$

whereas the relation

$$\begin{aligned} h'(1) + k'(1) &= \frac{\delta\nu(\delta\gamma+\nu^2) - (\nu^2\delta(\nu-\gamma)+\gamma(\nu+\delta)(\delta\gamma-\nu^2))p_{1,0,1}}{\nu(\delta+\nu)(\gamma(\delta+\nu)-\delta\nu)} \\ &\quad + \frac{\delta\gamma(\gamma+\nu)(\delta\lambda(\nu-\gamma)+\nu(\lambda+\nu)(\lambda+\delta))}{\lambda^2(\nu+\lambda+\gamma)(\lambda+\nu+\delta)(\gamma(\delta+\nu)-\delta\nu)}p_{1,0,1} \end{aligned} \tag{3.36}$$

follows from (3.14) and (3.15). Combining now (3.34), (3.35) and (3.36) with the expressions found for $p_{0,0,1}$, $p_{1,0,1^*}$ and $p_{0,1,1}$ (in (3.8), (3.16) and (3.23), respectively), we find

$$N_s = \frac{\delta^2(\gamma^2 - \gamma\nu + \nu^2)(1 - p_{1.0.1})}{\gamma(\delta+\nu)(\gamma(\delta+\nu) - \delta\nu)} + \left(\frac{\nu\gamma(\gamma+\lambda)(\gamma+\nu)}{\lambda^2(\nu+\lambda+\gamma)} - \frac{\gamma^2-\nu^2}{\nu}\right)\frac{\delta\, p_{1,0,1}}{\gamma(\delta+\nu)-\delta\nu} \quad (3.37)$$

where $p_{1,0,1}$ is given in (3.18). As shown in [20] N_s can be used to evaluate the fault-tolerance of the protocol. This issue is not addressed here.

N_s represents the expected number of forwarders through which a message would go on its way to reach the agent *if* the agent *does not* migrate in the meanwhile. We will compute now the expected number of forwarders, denoted by N, through which a message has to go in order to reach the agent. A similar analysis to the one conducted in Section 3.3.2.1 has to be done. Note that we should have $N > N_s$ as the number of forwarders between the message and the agent may increase over time.

When a communication starts, a message is generated 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 $i \geq 1$. If the number of forwarders is n at the beginning of a communication, this number decreases as the message travels towards the agent but it can increase if the agent migrates in the meanwhile. Define $N_{i,j,k}$ with $i \geq 1$, $j = 0,1$, $k = 1$ or with $(i,j,k) = (1,0,1^*)$, as the expected number of forwarders given that a message was generated when the system was in state (i,j,k) just after the generation of the message.

The expected number of forwarders N is given by

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

where $q_F(i,j,k)$ denotes the probability of reaching state (i,j,k) given that the process was initially in state $(1,0,0)$ (probability introduced in Section 3.3.2.1). It actually represents the probability that a communication starts when the system moves from state $(i,j,0)$ to state (i,j,k). The expressions of $q_F(i,j,k)$ are given in (3.25)–(3.27). Using them, we may rewrite N as follows

$$N = \frac{\lambda}{\lambda+\nu} N_{1,0,1^*} + \frac{\lambda(\lambda+\delta)}{\delta\nu} \sum_{i=2}^{\infty} r^i\, N_{i,0,1} + \frac{\lambda}{\delta} \sum_{i=1}^{\infty} r^i\, N_{i,1,1}. \quad (3.38)$$

With the definitions $N_0(z) := \sum_{i=0}^{\infty} z^i\, N_{i,0,1}$ and $N_1(z) := \sum_{i=0}^{\infty} z^i\, N_{i,1,1}$, (3.38) becomes

$$N = \frac{\lambda}{\lambda+\nu} N_{1,0,1^*} + \frac{\lambda(\lambda+\delta)}{\delta\nu}\,(N_0(r) - r\, N_{1,0,1} - N_{0,0,1}) + \frac{\lambda}{\delta}\,(N_1(r) - N_{0,1,1}). \quad (3.39)$$

It remains to determine the generating functions $N_0(z)$ and $N_1(z)$ 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 3.3:

$$N_{1,0,1^*} = \frac{\nu}{\nu+\gamma} N_{1,1,1} \tag{3.40}$$

$$N_{0,0,1} = 0 \tag{3.41}$$

$$N_{1,0,1} = \frac{\nu}{\nu+\gamma} N_{1,1,1} + \frac{\gamma}{\nu+\gamma} N_{0,0,1} \tag{3.42}$$

$$N_{i,0,1} = \frac{\nu}{\nu+\gamma} N_{i,1,1} + \frac{\gamma}{\nu+\gamma} (1 + N_{i-1,0,1}) \qquad i = 2, 3, \ldots \tag{3.43}$$

$$N_{0,1,1} = N_{1,0,1} \tag{3.44}$$

$$N_{i,1,1} = \frac{\delta}{\delta+\gamma} N_{i+1,0,1} + \frac{\gamma}{\delta+\gamma} (1 + N_{i-1,1,1}) \quad i = 1, 2, \ldots. \tag{3.45}$$

Notice from Equations (3.40)–(3.42) and (3.44) that

$$N_{1,0,1^*} = N_{1,0,1} = N_{0,1,1}. \tag{3.46}$$

After some algebraic manipulations on Equations (3.40)–(3.45), we obtain the following linear system of equations in the unknowns $N_0(z)$ and $N_1(z)$

$$\begin{aligned}
&(\nu + \gamma\,(1-z))\, N_0(z) - \nu\, N_1(z) = \frac{\gamma z^2}{1-z} - \nu N_{1,0,1} \\
&-\delta\, N_0(z) + (\delta + \gamma\,(1-z))\, z N_1(z) = \frac{\gamma z^2}{1-z} + \gamma z\, N_{1,0,1}.
\end{aligned}$$

Solving for $N_0(z)$ and $N_1(z)$ gives

$$N_0(z) = \frac{1}{D(z)} \left(\frac{\gamma\, z^2}{z-1} \Big(\gamma z^2 - (\gamma+\delta) z - \nu \Big) + \nu(\gamma z - \delta)\, z N_{1,0,1} \right) \tag{3.47}$$

$$N_1(z) = \frac{1}{D(z)} \left(\frac{\gamma\, z^2}{z-1} \Big(\gamma z - (\gamma+\delta+\nu) \Big) - (\gamma^2\, z^2 - \gamma(\gamma+\nu)\, z + \delta\nu) N_{1,0,1} \right) \tag{3.48}$$

where $D(z)$ is given in (3.31). Using (3.47)–(3.48) together with (3.46) and (3.31), (3.39) can be rewritten as follows:

$$N = \frac{((\lambda+\delta)(\lambda+\nu) - \gamma\nu) N_{1,0,1} - \nu\gamma(\lambda+\delta)/\lambda}{(\lambda+\delta)(\lambda+\nu) - \gamma(\lambda+\nu+\delta)} \tag{3.49}$$

where the denominator is nothing but the polynomial $\alpha(\lambda)$ introduced in the previous section. To identify the constant $N_{1,0,1}$ in (3.49), we proceed as in Section 3.3.2.1. In order for N to be well-defined for all non-negative values of λ, the numerator in (3.49) must vanish when $\lambda = \lambda_0 = 1/2 \times \left(\gamma - \nu - \delta + \sqrt{(\gamma+\nu+\delta)^2 - 4\delta\nu} \right)$, which is the only positive zero

of the denominator $\alpha(\lambda)$ (see Section 3.3.2.1). It is readily seen that $N_{1,0,1} = \nu/\lambda_0$ (Note: use the relation $(\lambda_0 + \delta)(\lambda_0 + \nu) = \gamma(\lambda_0 + \nu + \delta)$). Finally

$$N = \begin{cases} \dfrac{((\lambda+\delta)(\lambda+\nu) - \gamma\nu)\nu/\lambda_0 - \nu\gamma(\lambda+\delta)/\lambda}{(\lambda+\delta)(\lambda+\nu) - \gamma(\lambda+\nu+\delta)} & \text{for } \lambda \neq \lambda_0, \\ \dfrac{\nu(2\lambda_0^2 + (\nu+\delta)\lambda_0 + \gamma\delta)}{\lambda_0^2(2\lambda_0 + \nu + \delta - \gamma)} & \text{for } \lambda = \lambda_0, \end{cases} \tag{3.50}$$

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

Both expected numbers computed in this section (i.e. N_s and N) are expected to be increasing functions of δ and ν and decreasing functions of λ and γ which is illustrated in Figure 3.4. Looking at the upper graphs in Figure 3.4 in which the evolution of both N and N_s are plotted against λ (upper graph at the left) and ν (upper graph at the right), we can verify that $N > N_s$. The crosses in each graph indicate the value of N (the expected number of forwarders that a message encounters) corresponding to the same values of the model parameters: $\lambda = 1, \nu = 10, \delta = 20, \gamma = 50$. In Figure 3.4, the plots for $\delta = 30s^{-1}, \delta = 40s^{-1}$ (lower graph at the left) and $\gamma = 9s^{-1}$ (lower graph at the right) depict the behavior of our model when the ergodicity condition is close to being violated. In that situation, the expected number of forwarders grows to infinity.

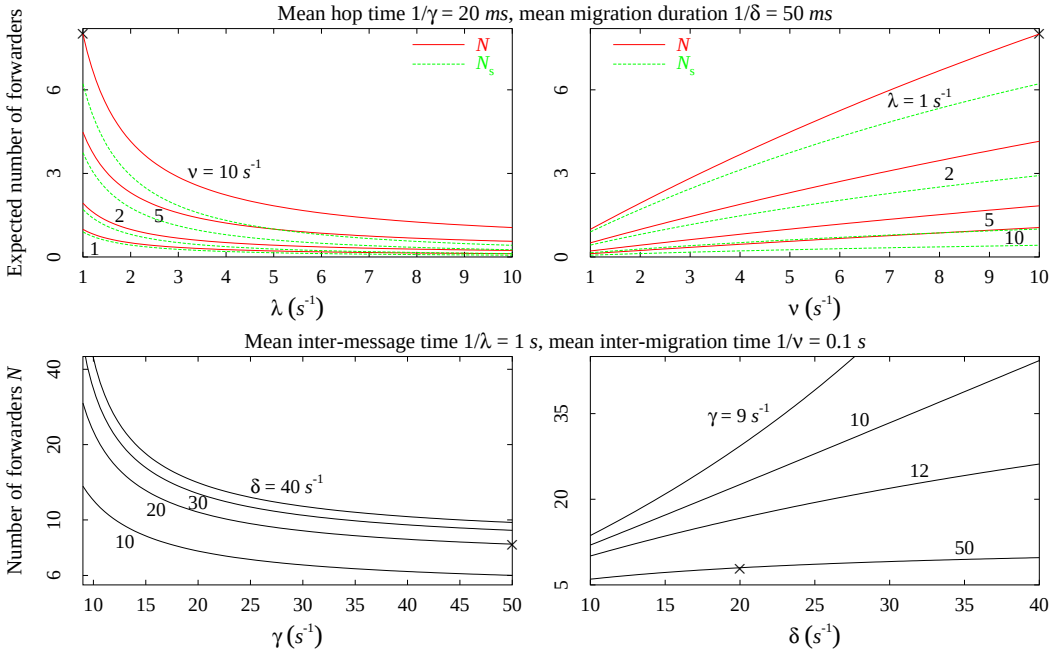

Figure 3.4: The expected number of forwarders

3.4 Centralized Server

3.4.1 Description

An alternative to the forwarders approach for locating a mobile agent is to use a *location server*. Such a server keeps track of the location of mobile agents in a database. Servers like this are widely used in the Internet. For instance, the Domain Name Server [85] uses a hierarchically organized servers to associate a location (IP address) to a 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 the location server is simple: each time a mobile agent migrates, it informs the server of its new location. Whenever the source wants to reach the mobile agent, it sends a message to the last known location of the mobile agent; 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 agent are only updated when needed. We now give a careful description of the protocol used by the source and the mobile agent to communicate with the server:

- The Mobile Agent

  Step 1: Performs the migration;

  Step 2: Sends its new location to the server.

- The Source

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

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

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

The above protocol is implemented in various Java libraries (*MOA* [84] and *ProActive* [101]). An illustration of this approach is given in Figure 3.5. In Figures 3.5(b)-(c) the chronological order of the events is indicated by the numbers $1, 2, \ldots$. In Figure 3.5(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.

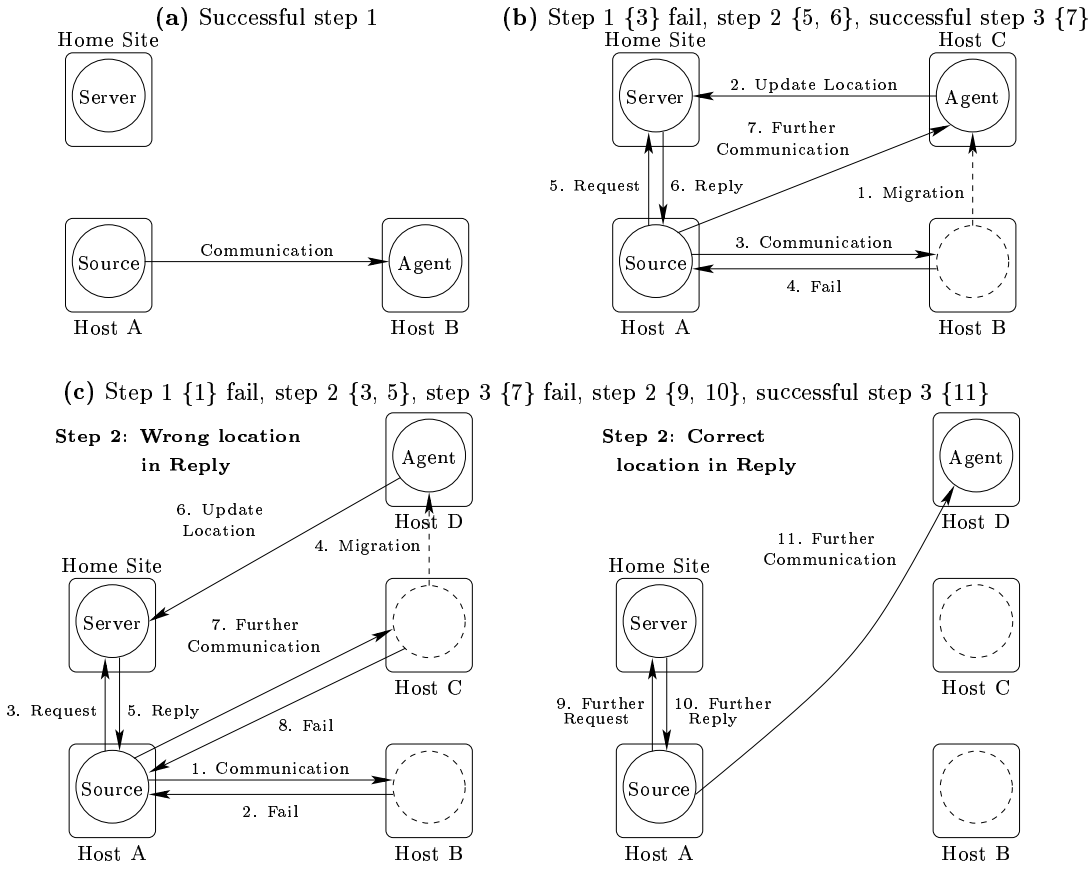

Figure 3.5: Some possible scenarios in the centralized approach from the source point of view

Figure 3.5(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* [101], 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 source-agent pair possesses its own queue. The server cyclically polls these different queues, serving one request per queue in each cycle. Within each queue, there is a priority scheduling mechanism that gives priority to `update requests` over `location requests`. If a queue contains several `update requests` sent by the same mobile agent 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 request` under processing cannot be preempted by a more recent one, which may harm the performance of the protocol[2]. For-

[2]Notice that only the queue that is attended by the server may have two `update requests`, all other

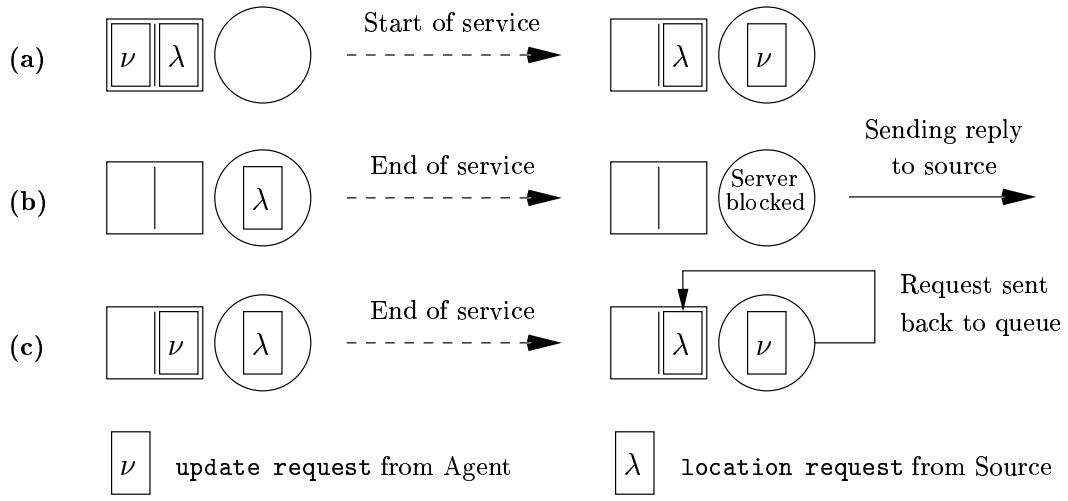

Figure 3.6: Details on the service policy at the server

tunately, 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 request` coming from the mobile agent; if this is the case, then the `location request` is sent back to the queue; otherwise, it sends the position of this mobile agent (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.

Figure 3.6 illustrates some scenarios at the server. In Figure 3.6(a), two requests are queued in the buffer. The `update request` is served with priority even though the `location request` is older. In Figure 3.6(b), the server is blocked after serving a `location request`, until the reply reaches the source. Figure 3.6(c) illustrates one inconvenient of using a Java library. Since the service policy is non-preemptive, a `location request` that is already served, is sent back to the queue to be served again whenever an `update request` is waiting in the buffer.

Observe that the server delivers a customized service to each type of queries: for `update 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 request`.

3.4.2 A Markovian analysis of the server

An exact modeling of the centralized approach would consist of modeling the system as

ones having at most one single pending `update request`.

a nonexhaustive cyclic server with vacations and finite buffers at the queues. 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 sources and/or agents. This queue may have at most a single `location request` and two `update requests`. By arguing that it is highly unlikely that an arriving `update request` finds an `update request` (from the same agent) being processed, we will assume that there can be at most one pending `update request` at the queue or, in other words, that an `update request` under processing is preempted by a more recent one (cf. Section 3.4.1). The latter restriction can easily been removed at the expanse of enlarging the state-space (which would yield a 29.6% increase in the number of states).

We assume that the set of assumptions **A1** holds (cf. Section 3.2). Note, however, that in this context p_j will represent the sum of the jth traveling time of the agent to its new host and of the travel time of the associated `update request` to the server site. We further assume that the traveling times between the source and the presumed location of the mobile agent (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 is only one source-agent pair in an application, μ would be the server processing speed). All of 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 3.7. 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 3.7.

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 request` or `location request`) in the queue, with $i_l \in \{0, \lambda, \nu\}$ the type of the message that occupies the lth position in the queue ($l = 1, 2$). By convention, $i_l = 0$ (resp. $i_l = \lambda$, $i_l = \nu$) indicates that the lth position is not occupied (resp. the lth position is occupied by a `location request`, the lth position is occupied by an `update request`). Recall that `update requests` have non-preemptive priority over a `location request` and that an arriving `update request` that finds one `update 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 mobile agent: $j = 1$ (resp. $j = 1^*$) will indicate that the mobile agent is migrating and that the source knows (resp. does not know) the location of the host that the mobile agent is leaving; similarly, $j = 0$ (resp. $j = 0^*$) will indicate that the mobile agent 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

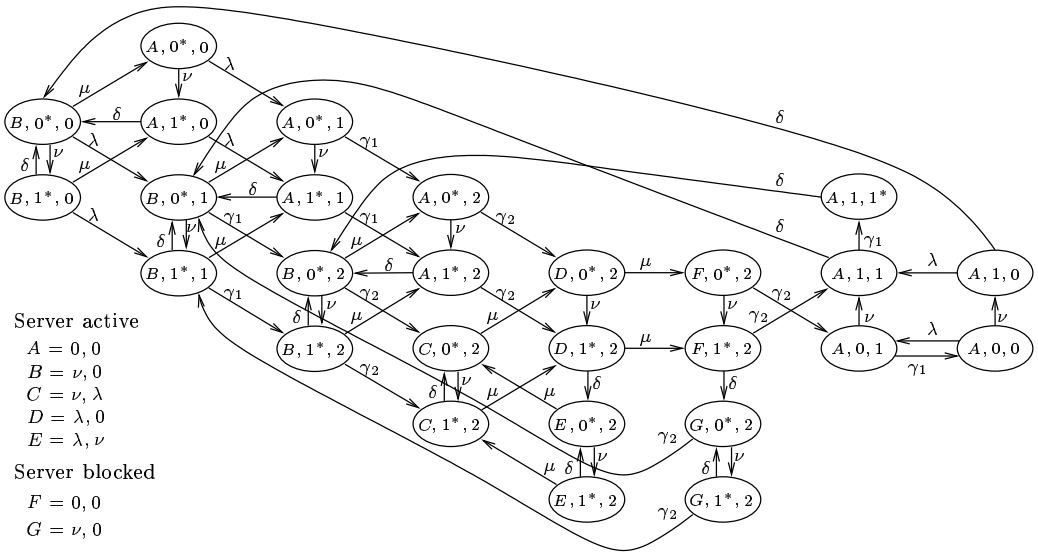

Figure 3.7: System states and transition rates in the centralized approach

mobile agent, $k = 1^*$ if the message sent by the source has reached a site that the mobile agent 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 mobile agent 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 3.2). 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 mobile agent returned by the server (which may or may not be the current location).

Under the above description/assumptions, the process depicted in Figure 3.7 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 3.7 ($\mathcal{F}$ 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 easily be identified from Figure 3.7), then we know (see e.g. [75]) that $\mathbf{p}$ is the unique solution of the system of linear equations $\mathbf{p} \cdot \mathbf{Q} = 0$, $\mathbf{p} \cdot \mathbf{1} = 1$, which can be solved by using a routine numerical procedure.

3.4.2.1 The expected communication time

We are interested in finding the expected communication time (see Section 3.4.2), 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 mobile agent and terminates when the latter is reached. It is seen from Figure 3.7 that a message may only be generated when the system is in 6 distinct states – the 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

$$T_S = \sum_{j=0,1} q_S(A, j, 1)\, T_{A,j,1} + q_S(A, j^*, 1)\, T_{A,j^*,1} + q_S(B, j^*, 1)\, T_{B,j^*,1} \tag{3.51}$$

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 3.7 it is easily seen that

$$q_S(A, 0, 1) = \frac{\lambda}{\lambda + \nu} \tag{3.52}$$

$$q_S(A, 1, 1) = \frac{\nu\lambda}{(\lambda + \delta)(\lambda + \nu)} \tag{3.53}$$

We will next 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 $w \in \mathcal{V} \cup \{(A, 0^*, 0), (A, 1^*, 0), (B, 0^*, 0), (B, 1^*, 0)\}$. We see from Figure 3.7 that

$$q_S(v) = \frac{\delta\nu}{(\lambda + \delta)(\lambda + \nu)}\, P((B, 0^*, 0); v) \tag{3.54}$$

for $v \in \mathcal{V}$. We now need to compute $P((B, 0^*, 0); v)$ for $v \in \mathcal{V}$. The following linear relations readily derive from Figure 3.7:

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

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

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

$$
\begin{bmatrix}
\lambda+\nu & -\nu & 0 & 0 \\
0 & \lambda+\delta & -\delta & 0 \\
-\mu & 0 & \lambda+\nu+\mu & -\nu \\
0 & -\mu & -\delta & \lambda+\delta+\mu
\end{bmatrix}
\begin{bmatrix}
p((A,0^*,0);v) \\ p((A,1^*,0);v) \\ p((B,0^*,0);v) \\ p((B,1^*,0);v)
\end{bmatrix}
= \lambda
\begin{bmatrix}
p((A,0^*,1);v) \\ p((A,1^*,1);v) \\ p((B,0^*,1);v) \\ p((B,1^*,1);v)
\end{bmatrix}
\tag{3.55}
$$

for all $v \in \mathcal{V}$. All terms in the r.h.s. of (3.55) are either 0 or 1, as shown in Table 3.1 (rows 2–5). From rows 2–5 in Table 3.1 and (3.55), we can compute $p((B,0^*,0),v)$ for all $v \in \mathcal{V}$ (see last row of Table 3.1).

Table 3.1: Values of the probabilities, with $K := (\lambda+\nu+\mu)(\lambda+\delta+\nu)$

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

Substituting the resulting values of $p((B,0^*,0);v)$ into (3.54) yields

$$
q_S(A,0^*,1) = \frac{\nu\mu\delta}{(\lambda+\nu)(\lambda+\nu+\mu)(\lambda+\delta+\nu)} \tag{3.56}
$$

$$
q_S(A,1^*,1) = \frac{\nu^2\mu\delta(2\lambda+\delta+\mu+\nu)}{(\lambda+\delta)(\lambda+\nu)(\lambda+\nu+\mu)(\lambda+\delta+\nu)(\mu+\lambda+\delta)} \tag{3.57}
$$

$$
q_S(B,0^*,1) = \frac{\nu\delta}{(\lambda+\nu+\mu)(\lambda+\delta+\nu)} \tag{3.58}
$$

$$
q_S(B,1^*,1) = \frac{\nu^2\delta}{(\lambda+\nu+\mu)(\lambda+\delta+\nu)(\mu+\lambda+\delta)}. \tag{3.59}
$$

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 Theorem 3.3.3 pages 113-114 in [89])

$$T_{A,0,0} = 0, \qquad \sum_{\mathbf{i},j,k} q_{(\mathbf{i}',j',k'),(\mathbf{i},j,k)} T_{\mathbf{i},j,k} = -1 \quad \text{for } (\mathbf{i}', j', k') \neq (A, 0, 0) \tag{3.60}$$

where $q_{(\mathbf{i}',j',k'),(\mathbf{i},j,k)}$ are the elements of the generator matrix $\mathbf{Q}$. In order to simplify (3.60), 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 (3.60) then reads

$$\mathbf{M}_{A,0,0} \cdot \mathbf{T} = -\mathbf{1}. \tag{3.61}$$

Solving (3.61) and using (3.52),(3.53) and (3.56)–(3.59), we finally find T_S.

3.5 Validation and comparison

3.5.1 Validation through simulations

The theory developed in Sections 3.3 and 3.4 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 3.3 and 3.4 against results obtained from event-driven simulations of both schemes. In the simulations, we have considered a single agent and a single source. We have run several simulations that explore the effects of violations of one assumption at a time and the violations of all of the assumptions. 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 at hand, we have collected measurements of the travel times, the migration duration and the service times on a LAN and a MAN and fitted the resulting data to well-known distributions. Table 3.2 summarizes our findings. Except for the service times which are approximately constant, all other (network-dependent) parameters are well represented by Weibull distributions. The distribution of the communication rate λ (inverse of the average idle time for the source) and the migration rate ν (inverse of the average idle time 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 (in the case of idle times and service times) or Weibull (in the case of migration durations and travel times). Last, we simulate the systems where all random variables are non-exponential. In these runs, the only assumptions not violated are the ones concerning the independence of the processes at hand.

Table 3.2: Distribution fits for the model parameters

Random variable (ms)	100Mb/s switched LAN	7Mb/s MAN
Forwarders		
migration durations	Weibull shape 4, scale 100	Weibull shape 2.5, scale 392
travel times	Weibull shape 11, scale 23	Weibull shape 6, scale 88
Server		
migration durations	Weibull shape 2.5, scale 75.5	Weibull shape 3, scale 1010
travel times source-agent	Weibull shape 1.8, scale 9.7	Weibull shape 10, scale 28.6
travel times source-server	Weibull shape 1.8, scale 14.7	Weibull shape 1.8, scale 93
service times	$\approx$ constant	$\approx$ constant

Table 3.3: Sample mean and percentiles of the relative error provided by the models. Default values: $\lambda = 1, \nu = 10, \delta = 11$ (forwarders), $\delta = 15$ (server), $\gamma = 45, \gamma_1 = 115, \gamma_2 = 75, \mu = 2325$

Simulations	Mean	25	50	75	90	95
Forwarders						
deterministic source idle times$\lambda \in [1, 10]$	7.7	6.5	8.8	9.6	10.3	10.5
deterministic agent idle times$\nu \in [1, 10]$	1.6	0.3	1.4	2.4	3.5	4.2
Weibull migration times $\delta \in [8, 25]$ shape 4	8.3	4.3	8.2	10.4	14.7	16.3
Weibull travel times $\gamma \in [33.8, 69.8]$ shape 11	2.7	1.0	2.2	3.9	6.4	7.1
All together $\lambda, \nu \in \{1, 3, 5, 7, 9\}$	14.1	4.9	10.0	20.6	32.4	38.3
Server						
deterministic source idle times$\lambda \in [1, 10]$	14.9	15.1	16.3	17.1	17.1	17.2
deterministic agent idle times $\nu \in [1, 10]$	2.4	2.2	2.2	2.6	4.1	4.8
Weibull migration times $\delta \in [12, 19]$ shape 2.5	12.3	11.7	12.2	13.5	14.8	15.5
Weibull travel times $\gamma_1 \in [90, 131]$ shape 1.8	1.3	0.6	1.5	1.8	2.1	2.3
Weibull travel times $\gamma_2 \in [56, 94]$ shape 1.8	0.9	0.3	0.6	1.3	2.2	2.7
deterministic service times $\mu \in [500, 2500]$	1.5	0.6	1.5	2.2	2.9	3.5
All together $\lambda, \nu \in \{1, 3, 5, 7, 9\}$	14.1	6.1	10.5	17.0	30.0	40.1

Table 3.3 reports the sample mean and the percentiles of the relative error (expressed in percentage) between simulated results and theoretical expected response times as predicted by both models. The values in Table 3.3 are obtained after 3000 seconds of simulation (the steady-state is reached after 1000 seconds of simulation run, see [64] for more details on the convergence issue). 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 row 6 (resp. rows 12 and 13) in Table 3.3).

Neither model is sensitive to the distribution of agent idle times. 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

row 4 (resp. row 10) in Table 3.3). The model of the location server is very robust against deterministic service times. The largest error observed during the simulations is 3.7%.

However, the models are more sensitive to the distribution of the migration durations and the source idle times. Nevertheless, when simulating the forwarders scheme the mean relative error is 7.7% (resp. 8.3%) for the case that the source idle times are deterministic (resp. the migration durations are Weibull) (see column 2 in Table 3.3 row 3 (resp. row 5)). When simulating the centralized approach, we observe almost the same relative error when the source idle times are deterministic (resp. the migration durations are Weibull), the largest error being 17.2% (resp. 16%).

When all of 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 14.1% in both models (see rows 7 and 15 in Table 3.3). The robustness of the models against correlated processes will be addressed in the next section.

3.5.2 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 [101] which provided us with all of the necessary mobile primitives. The network was composed of various Pentium II and Pentium III machines running Linux (2.2.18) and Sun SPARC machines 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 [59] in all experiments.

For each approach (forwarders and location server), we executed a single source-agent pair on the testbed. Source idle times 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 experiment to another. All of the other model parameters $(\delta, \gamma, \gamma_1, \gamma_2, \mu)$ are system-dependent and cannot be changed. Hence, among all of 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 3.2)). Indeed, migration durations and communications latencies were found to have Weibull distributions both on a LAN and a MAN whereas service times were approximately constant. Furthermore, assumptions on the independence of these processes are not valid under real conditions. Migration durations and travel times are particularly correlated in real life.

Figures 3.8 and 3.9 report both the experimental and theoretical expected response times as well as the expected number of forwarders obtained for a LAN and for a MAN. Graphs on the left display the expected response time (upper graphs) or the expected number of forwarders (lower graph) 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 ($\nu = 1, 5, 10s^{-1}$); similarly,

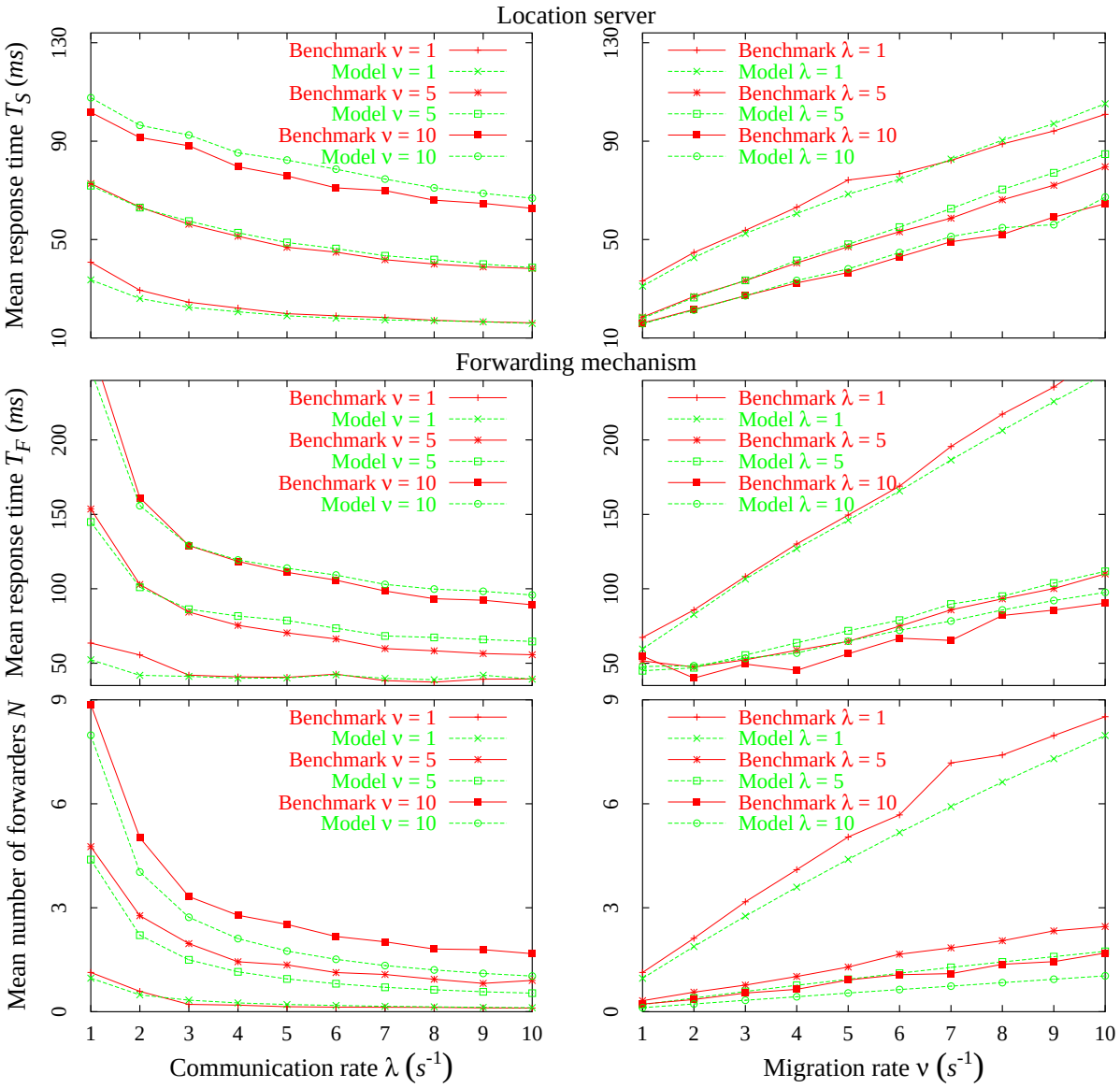

Figure 3.8: Validation with experiments on a 100Mb/s LAN

graphs on the right display the expected response time (upper graphs) or the expected number of forwarders (lower graph) 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 ($\lambda = 1, 5, 10s^{-1}$). For each value of the pair (λ, ν), the empirical values of δ and γ (resp. $\delta, \gamma_1, \gamma_2$ and μ) were substituted into the expressions of T_F and N (resp. T_S) given in (3.33) and (3.50) (resp. (3.51)) when the forwarding mechanism (resp. the centralized mechanism) was used. Observe that analytical and experimental response times are very close to each other across almost all experiments. The average number of forwarders obtained in the experiments is not as well predicted by the model via (3.50), especially on a MAN. It appears that the analytical values underestimate the actual average number of forwarders. The gap between

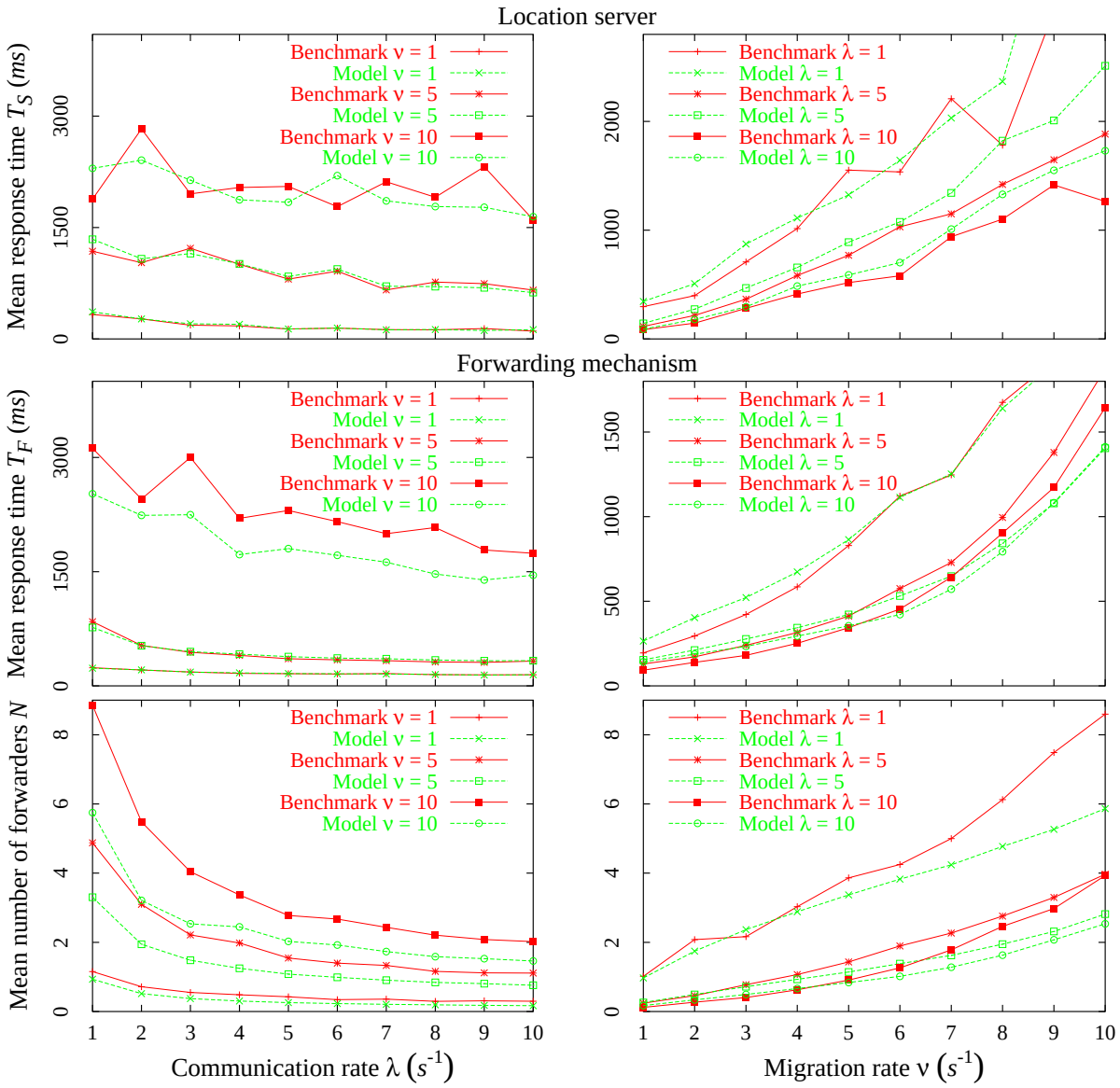

Figure 3.9: Validation with experiments on a 7Mb/s MAN

analytical and experimental values increases when the communication rate λ decreases. For small migration rates $\nu < 5$, there is almost no gap (i.e. almost no absolute error).

For each network configuration (LAN or MAN), the performance of the models are collected in Tables 3.4–3.5. In Table 3.4, we report the sample mean (column 2) and order statistics (25th, median, 75th, 95th and 99th percentiles in columns 3–7) of the *absolute* error (or gap) between analytical N and experimental results. In Table 3.5, we report the sample mean (column 2) and order statistics (25th, median, 75th, 90th and 95th percentiles in columns 3–7) of the relative error between analytical T_F or T_S and experimental results.

We have chosen to report statistics on the absolute error (or gap) $|N_{exp} - N|$ in Table

Table 3.4: Sample mean and percentiles of the absolute error on the average number of forwarders in the forwarding mechanism

Experiments	Mean	25	50	75	95	99
100Mb/s LAN	0.4172	0.1764	0.3914	0.6281	0.8336	1.2336
7Mb/s MAN	0.8360	0.1280	0.4202	1.3881	2.6996	5.2652
Overall	0.6247	0.1467	0.4045	0.7371	1.9469	4.4891

Table 3.5: Sample mean and percentiles of the relative error on the communication time in both mechanisms

Experiments	Mean	25	50	75	90	95
Forwarders						
100Mb/s LAN	7.3	2.1	5.7	10.8	17.0	20.3
7Mb/s MAN	12.1	2.3	7.8	19.0	25.9	35.0
Overall	9.7	2.2	7.1	15.4	22.1	26.3
Server						
100Mb/s LAN	4.6	2.3	4.3	6.2	8.5	10.4
7Mb/s MAN	13.9	6.2	11.3	21.5	28.3	33.0
Overall	9.6	3.7	6.5	13.4	23.4	28.3

3.4, rather than reporting statistics on the relative error $|N_{exp} - N|/N_{exp}$, as we believe that the former are more insightful. For instance, in experiments over a LAN, the plots corresponding to $\nu = 1$ (refer to the lowest graph at the left in Figure 3.8) are almost identical. The maximum gap between the plots is 0.16558 which is an excellent result; however, for the same gap $N_{exp} = 1.13393$ yielding 14.6% of relative error which does not sound very good. In experiments over a LAN, the absolute error between experimental and analytical mean number of forwarders is relatively low, the highest gap being 1.26 (see row 2 in Table 3.4). This is not the case in experiments over a MAN where the maximal gap is 5.38164. Still, the average gap is 0.836 and in 75% of the experiments the gap is under 1.3881 (see row 3 in Table 3.4).

Results in Table 3.5 indicate that the theoretical models behave fairly well. Their overall performance are very close to each other (see rows 5 and 9 in Table 3.5): the sample mean of the relative error is 9.7% (resp. 9.6%) for the model of the forwarders (resp. of the server) (see column 2 in Table 3.5) and in 90% of the experiments using the forwarders (resp. the location server) the relative error is below 22.1% (resp. 23.4%) (see column 6 in Table 3.5). However, both models are more robust when applied on a LAN (see rows 3 and 7 in Table 3.5) than when applied on a MAN (see rows 4 and 8 in Table 3.5).

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. For better viewing, the empirical mean response

time returned by both mechanisms is the only variable plotted in Figure 3.10. Graphs on the left display the empirical mean response time as a function of the communication

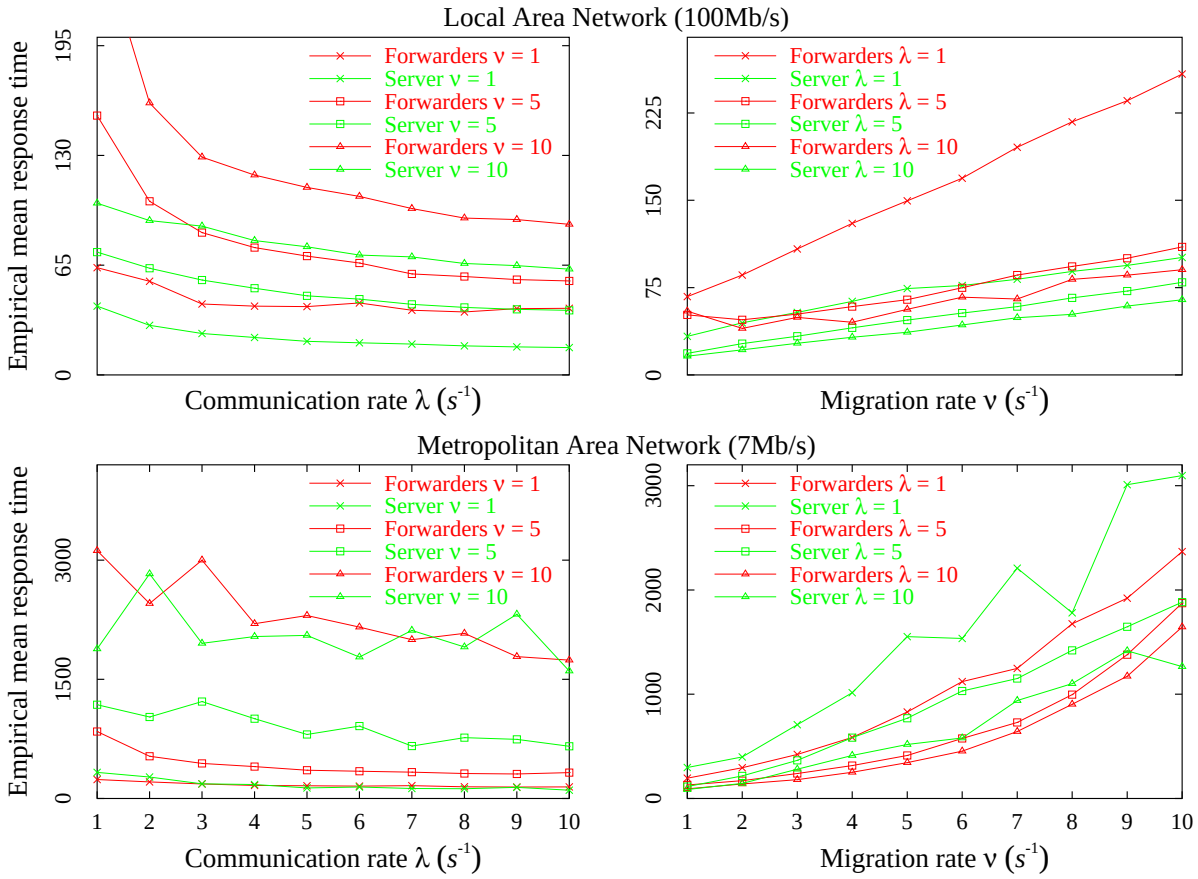

Figure 3.10: Experimental results obtained on a LAN and a MAN

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 empirical mean 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 ($\lambda = 1, 5, 10$). Looking at the upper pair of graphs in Figure 3.10 (experiments on a LAN), it appears that the location server scheme has a lower response time when compared to the forwarders scheme. Unexpectedly, we observe the opposite in the lower pair of graphs (experiments on a MAN), where the forwarders scheme achieves the smallest communication time in most of the experiments. 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.

Remark 3.5.1 *We would like to point out that the ergodicity condition was always satisfied in our experiments. Actually, this will always be the case, in practice, due to the fact that the migration of an agent over the network is achieved by sending several messages from*

the old site to the new one. Sending only one of these messages is expected to take $1/\gamma$ *seconds, while the whole migration process accounts for* $1/\delta$ *seconds. Thus, we always have* $1/\delta > 1/\gamma$.

3.5.3 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 limitation occurs when comparing both approaches by using the theoretical results obtained in Sections 3.3 and 3.4. We will focus on the expected response time given by each approach and, more precisely, on their difference ΔT, namely,

$$\Delta T = T_F - T_S \tag{3.62}$$

where T_F and T_S are given in (3.33) and (3.51), respectively. Several cases have been investigated corresponding to different values of the model parameters. Unless otherwise mentioned, the parameters have the values listed in Table 3.6. Each entry in Table 3.6 is the

Table 3.6: Values used for theoretical comparison

Network	$\delta(s^{-1})$	$\gamma(s^{-1})$	$\gamma_1(s^{-1})$	$\gamma_2(s^{-1})$	$\mu(s^{-1})$
100Mb/s LAN	10.9	45.6	115.6	76.3	2325
7Mb/s MAN	1.6	12.3	36.7	12.1	938

average value obtained over all experiments reported in Section 3.5.2. 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 3.11 displays the sign of ΔT in all of the cases that we have investigated.

We first consider LAN and MAN conditions (refer to rows 2 and 3 in Table 3.6 for self-contained information) and we vary the communication and migration rate from 1 to 50. The sign of ΔT in these cases is shown in Figure 3.11(a). Under LAN conditions, the line $\Delta T = 0$ corresponds to very high migration rates ($\nu > 30$). Above this line, the forwarders mechanism performs better than the centralized mechanism, and below this line, the opposite statement holds. Under MAN conditions, the line $\Delta T = 0$ corresponds to very low migration rates ($\nu < 2.5$) and to high communication rates ($\lambda > 10$). Below this line, the centralized mechanism performs better than the forwarders mechanism, and above it, the opposite statement holds. Observe that in our experiments, we investigated the region $1 \leq \lambda 10$ and $1 \leq \nu 10$ and we have found that, over a LAN, the centralized technique is faster, and over a MAN, the forwarders achieve lower communications time. Our conclusions, drawn from the experiments, agree with the study of the sign of ΔT. However, what we could

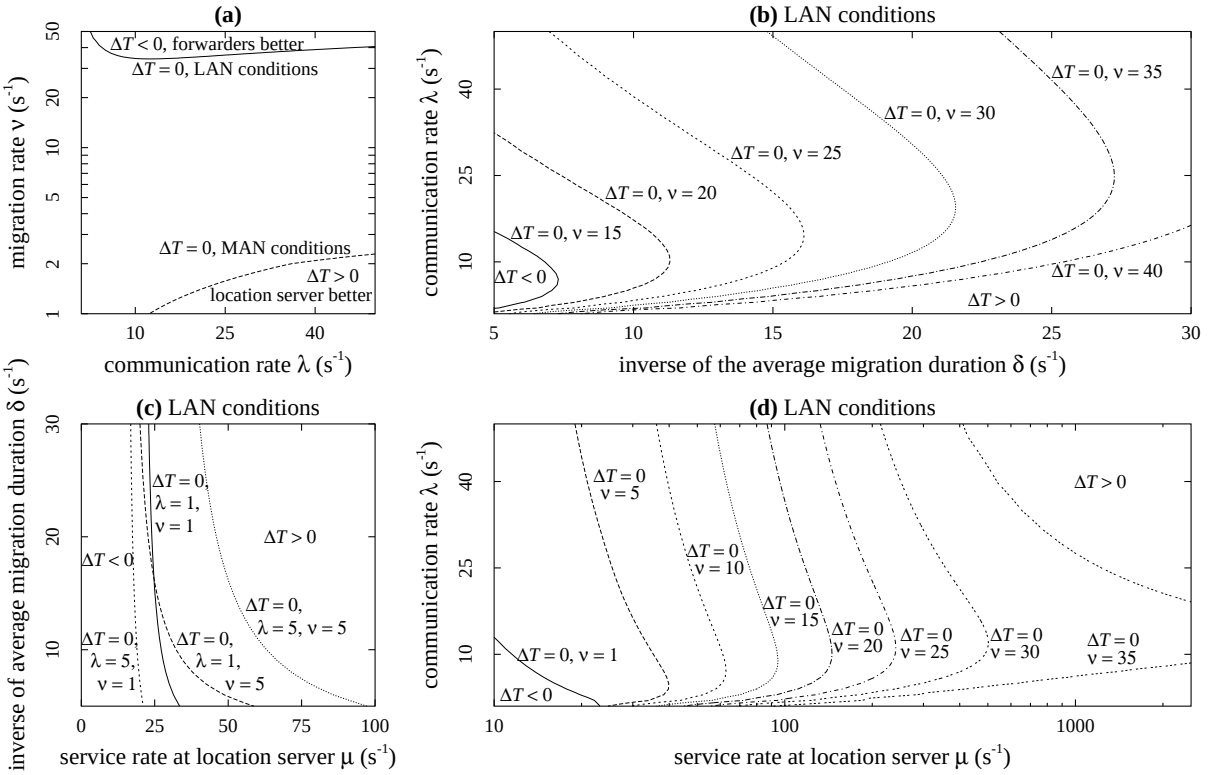

Figure 3.11: Sign of the difference between response times $\Delta T = T_F - T_S$

not foresee in the experiments is that for some migration rates (e.g. $\nu = 35$) the forwarders scheme is better only for intermediate values of the communication rate ($\lambda = 9 \ldots 19$ for instance). This is also observed in Figures 3.11(b) and (d) for fixed values of δ and μ. A closer look at this phenomenon revealed that, for intermediate values of λ, the difference ΔT is slightly negative. In other words, the performance of the mechanisms considered here is almost the same for this choice of parameters. This observation suggests that it would be preferable to draw the lines $|\Delta T| = \epsilon$, where ϵ is small, instead of drawing the line $\Delta T = 0$. Then, for $\Delta T > \epsilon$, the location server is preferable; for $\Delta T < \epsilon$, the forwarders are preferable; and for $|\Delta T| \leq \epsilon$ either mechanism can be chosen.

In Figure 3.11(c), we observe that for extremely low service rate (μ less than $16 s^{-1}$), the forwarders technique becomes better than the centralized technique. The frontier between the regions where one approach is better than the other, i.e. the line $\Delta T = 0$, depends on the values of λ and ν. Surprisingly enough, increasing the communication rate while keeping the migration rate unchanged 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 source-agent pair, the operational conditions are far away from these

frontiers ($\mu = 2325$ in our experiments over a LAN). However, when there are multiple sources and/or agents, the buffer at the server is completely partitioned between them, so that each pair would have only a fraction of the server processing speed (which is simply $\mu = 2325$). We have seen from our simulations (reported in Section 3.6), that the service rate per queue depends on parameters λ and ν. For instance, we have seen that for $\lambda = 1$ and $\nu = 1$ there should be around 100 source-agent pairs in order to have a service rate per queue as low as 30, whereas 60 pairs are enough to achieve the same service rate per queue when $\lambda = 10$ and $\nu = 1$. It is only for a large number of source-agent pairs that the performance of the server degrades till the point where forwarders may perform better.

Figure 3.11(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 3.6). 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 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 3.4.1). 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.

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.

3.6 Extension to the case of multiple source-agent pairs

The model developed in Section 3.4.2 is precise in the case of a single source-agent pair. But it is just an approximation in the case of applications with several sources and/or agents. In this section, we will first give some guidelines on a precise modeling of systems with multiple objects, and show next how one can use our model in such systems (approximate solutions only).

In applications with several sources and/or agents, there is a single server attending multiple queues. Each queue can have up to two requests, a single `update request` coming from an agent and a single `location request` from a source trying to communicate with the agent. There are as many queues as there are source-agent pairs. The server switches from queue to queue in a cyclic way, serving a unique request in each queue. Such a server is well modeled by a single-server polling model with cyclic non-exhaustive service. In order

to obtain accurate results, the fact that the server stays blocked while sending the replies to the sources must be taken into account. In such cases, the server is unavailable for work even though customers could be waiting. To model this, there are two possible choices: either adopting a server with vacation [115] as illustrated in Figure 3.12(a), or considering the "block" times as making part of the service times of `location requests`, in which case it would be better to have two different service rates (μ_1 for the `update requests` and μ_2 for the `location requests`) (see illustration in Figure 3.12(b)). Such polling models are not easy to analyze. An alternative (and precise) model is presented next.

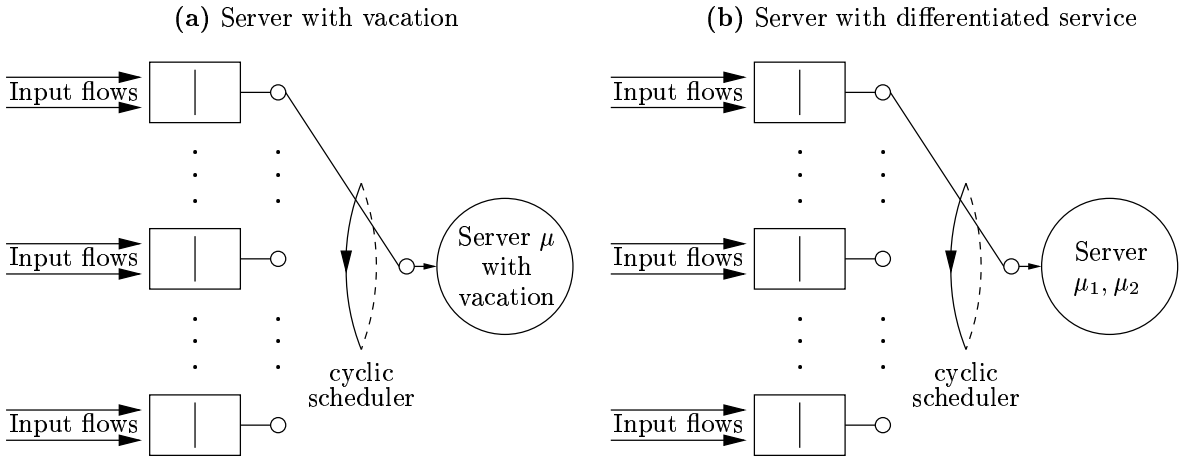

Figure 3.12: Possible choices for modeling the original system

A vacation model describes a single-server queue in which the server can be unavailable for work (away on "vacation") even though customers are waiting [115]. In the polling model illustrated in 3.12(a), the time that the server spends away from any particular queue, serving the other queues or waiting for the end of a "block" period, can be viewed as a vacation from that queue. Adoption of this viewpoint greatly simplifies the analysis of the polling model. The decomposition of the multiple queues system into multiple single queue subsystems is illustrated in Figure 3.13.

Figure 3.13(a) illustrates the real system having a cyclic server with vacation. Figure 3.13(b) illustrates the equivalent system having independent queues, each with one server with vacation. Note that the moments of the vacations are not the same in both systems. In the original system (at the left), the vacations stand for the time in which the server is blocked, whereas in the equivalent system (at the right), the vacations stand for the time the server is away from any particular queue ("block" time + service times of other queues). In both systems, there are two input flows in each queue, one resulting from the `update requests` sent by a mobile agent, and the other resulting from the `location requests` sent by a source. The arrival rates into queue i ($i = 1, \ldots, n$ where n is the total number of queues) are then $\nu_i\delta_i/(\nu_i+\delta_i)$ (`update requests` from agent i) and e_i (`location requests` from source i). The rate e_i and the first two moments of the vacation period can hardly be

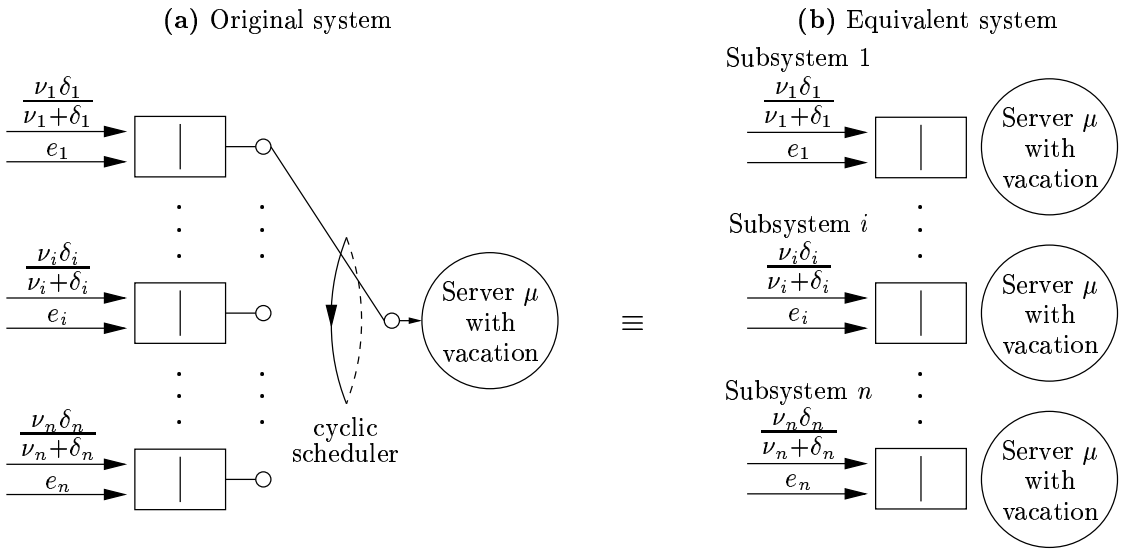

Figure 3.13: Decomposition of the cyclic-service system into n independent single queue subsystems

computed.

The model developed in Section 3.4.2 can be used if one is able to find the equivalence shown in Figure 3.14. At the left of Figure 3.14, the subsystem with one queue and one server with vacation is drawn (the same subsystem as in Figure 3.13(b)). At the right of Figure 3.14, an equivalent system having a server with reduced speed μ_i is illustrated. The

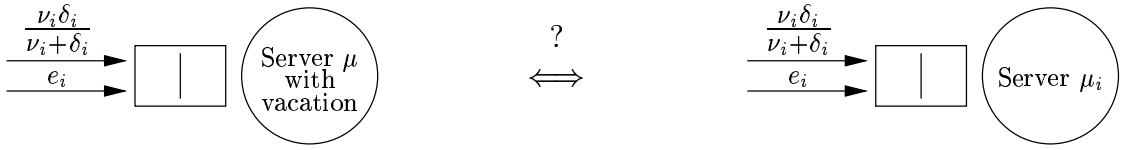

Figure 3.14: Equivalence between two systems, one having a server with vacation (service rate μ) and the other having a server with service rate μ_i

key point of this equivalence resides in the computation of the processing speed μ_i of queue i in the equivalent system. We will necessarily have $\mu_i \leq \mu$ to compensate the vacation periods in the system at the left of Figure 3.14. The equality $\mu_i = \mu$ is obtained when there are a single source-agent pair in the system. Observe that, if there are just a few sources and/or agents, there is likely to be just one active (non-empty) queue most of the time, due to multiplexing. In such cases, it is sufficient to consider $\mu_i \approx \mu$. But what about the general case?

The equivalent systems drawn in Figure 3.14 are assumed to behave identically concerning waiting times and response times. Let $T_{vacation}$ (resp. $T_{reduced}$) denote the response time of the system with vacation (resp. the system with reduced speed). The time $T_{vacation}$

depends on parameters $\nu_i, \delta_i, e_i, \mu$ and the first two moments of the vacation periods. The time $T_{reduced}$ depends on parameters ν_i, δ_i, e_i and μ_i which is to be calculated. Writing $T_{vacation} = T_{reduced}$ yields the computation of μ_i in terms of $\nu_i, \delta_i, e_i, \mu$ and the first two moments of the vacation periods. Unfortunately, it is quite hard to express e_i and the first two moments of the vacation periods, and not much can be done without these parameters.

One may infer the value of μ_i by measuring, at the source, the response time of the server. If the source records the sending time st of a `location request` and the reception time rt of its reply, it can measure the response time as $rt - st - RTT$, where RTT is the round-trip time of a message through the network. The service rate is roughly $1/(rt-st-RTT)$. For systems with a single source-agent pair, we have $1/(rt-st-RTT) \lesssim \mu = \mu_i$. This approximation works only in the special case of high speed networks (LAN for instance). Recall that whenever the server sends a reply to a source, it remains blocked until the reply reaches its destination (synchronous communications between objects). Therefore, when the ratio $\frac{travel\ time}{service\ time}$ increases the approximation gets worse and worse since the block times are considered to be service times.

We already know that over a MAN the forwarders scheme performs better. A single server with multiple queues is definitely slower than a server with a single queue. It is evident that the forwarders will achieve better results in applications with multiple objects. We will therefore concentrate on LAN networks, and determine up to which utilization of the server the approximation holds. Note that it is possible to determine which mechanism achieves the lowest response time by substituting μ_i into ΔT as given by (3.62) (refer to Figure 3.11(d) for more details).

We have carried out four sets of simulations with multiple objects. In each set of simulations, all of the RVs were distributed exponentially. This way, one can precisely measure the goodness of the approximation made in the inference of μ_i. All source-agent pairs had the same migration and communication rates, $\nu_i = \nu$ and $\lambda_i = \lambda$ for $i = 1, \ldots, n$ where n denotes the number of source-agent pairs considered in a particular simulation. In each set of simulations, n was held fixed during a simulation run, and its value ranged from 1 to 100 over all simulations in the same set (the total number of simulations is thus 400). Over all 400 simulations, the travel times, the service times and the migration durations were i.i.d. RVs with rates γ_1, γ_2, μ and δ respectively. The values retained for the latter parameters correspond to a LAN condition (refer to row 2 in Table 3.6 for self-contained information). Finally, each set of simulations is characterized by a single value for the pair (λ, ν). The values retained are (1,1), (1,10), (10,1) and (10,10).

For each simulation, we have computed the expected communication time T_S using (3.51) together with the approximation of μ_i, and measured the utilization of the multiqueue server. The average response time returned by the simulations together with the theoretical T_S are plotted against the number of source-agent pairs n in Figure 3.15. (The analysis curves are not smooth because μ_i is inferred from measurements.) We have also plotted, for each set of simulations, the utilization of the server and the relative error between the simulated result and the analytical one. Figure 3.15 contains eight graphs, two of which

pertain to the same set of simulations. Observe how in all four sets of simulations, the analytical T_S is larger than the simulated result. This is expected since the inferred μ_i (cf. Figure 3.14) is lower than the true one, and the gap between both values increases as the number of queues n increases.

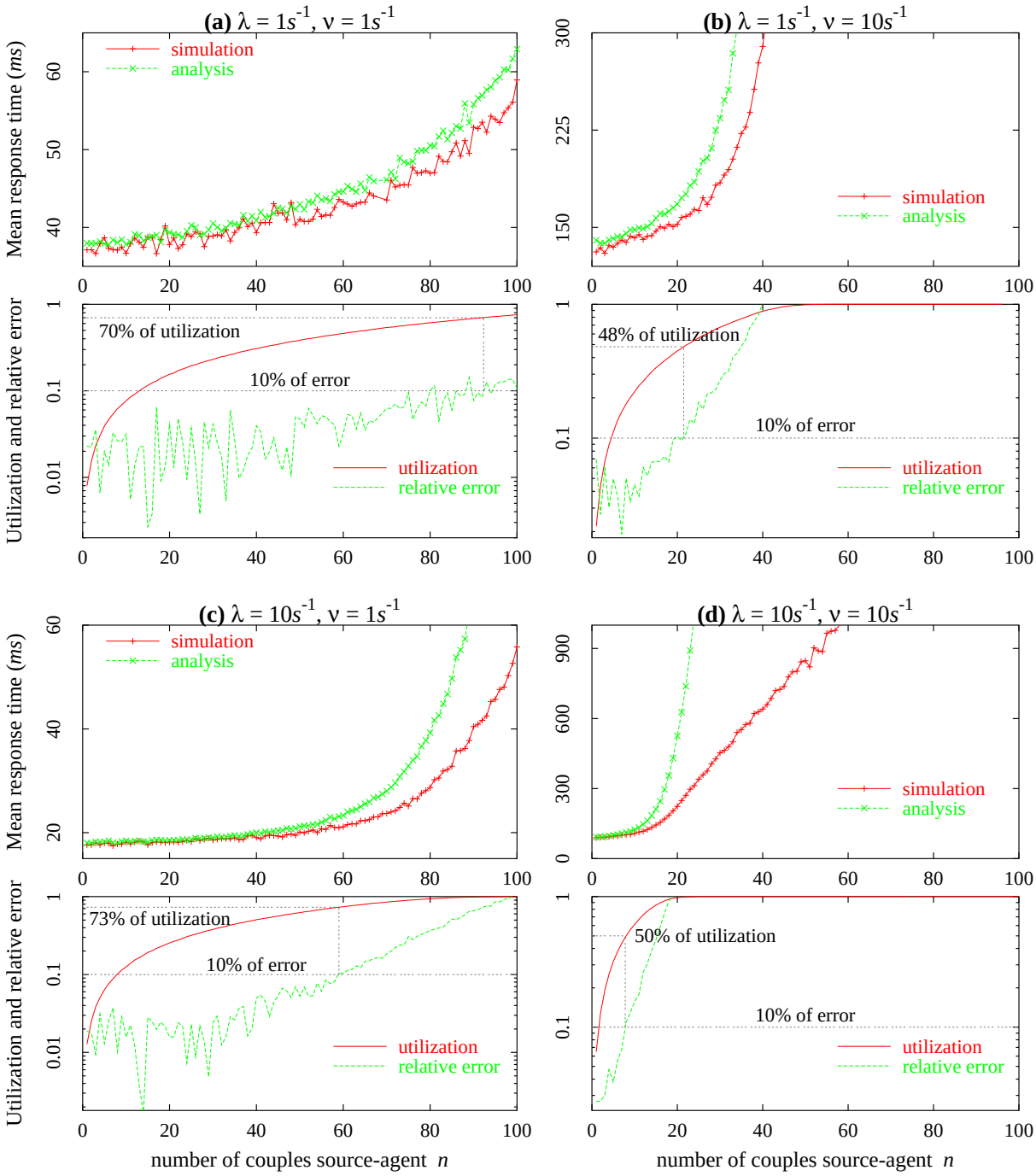

Figure 3.15: Simulated and analytical response times, utilization of the server and relative error between both response times

We have pointed out in Section 3.5.3 that the performance of the location server mechanism degrades rapidly when the migration rate ν increases. This is even more true in the case of multiple objects, especially as the migration rate increases for each agent (all agents have the same migration rate in the simulations). The reader is invited to compare the results shown in Figures 3.15(a) and (b) and the ones in Figures 3.15(c) and (d). The approximation on μ_i yields a smaller error on the response time when the migration rate ν is smaller, for the same utilization of the server and the same communication rate λ. On the other hand, changing the communication rate λ from 1 to 10 while keeping the migration rate ν unchanged does not have a big effect on the quality of the approximation (see Figures 3.15(a) and (c) and Figures. 3.15(b) and (d)). A quantitative comparison of the quality of the approximation in the four sets of simulations is reported in Table 3.7.

Table 3.7 reads as follows: with a tolerance of 10% of error (resp. 15% of error) on the response time, the approximate model can be used as long as the server utilization is below 0.73 (resp. 0.81), which is attained when there are 59 (resp. 66) source-agent pairs, given that $\lambda = 10$ and $\nu = 1$ (see row 5 in Table 3.7). Note that in the set of simulations

Table 3.7: Utilization and number of source-agent pairs yielding 10% and 15% of error

	10% of error		15% of error	
Simulations set	Utilization	Number of pairs	Utilization	Number of pairs
$\lambda = 1, \nu = 1$	0.70	92	N. A. (> 0.77)	N. A. (> 100)
$\lambda = 1, \nu = 10$	0.48	21	0.55	25
$\lambda = 10, \nu = 1$	0.73	59	0.81	66
$\lambda = 10, \nu = 10$	0.50	8	0.61	10

where $\lambda = 1$ and $\nu = 1$, the maximal error obtained was 14.3%. Thus the utilization and the number of pairs which yield 15% of error on the response time are not available, but we know that for such error, the utilization of the server must be higher than 0.77 and that there must be more than 100 source-agent pairs (see row 3 in Table 3.7).

To conclude this section, we can say that in all cases, the approximate model returns fair prediction of the expected communication time as long as the utilization does not exceed 50%.

3.7 Conclusion

We have proposed simple Markovian analytical models for evaluating the performance of two approaches for locating mobile agents. One approach uses forwarders to enable communication between a source and a mobile agent; 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 of 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. 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.