Evolution and Maintenance of Proxy Networks for Location Transparent Mobile Agents and Formal Representation By Graph Transformation Rules

Masahito Kurihara[a] **and Masanobu Numazawa**[b]

[a] *Hokkaido Institute of Technology*
Maeda 7-15, Teine Ward, Sapporo, 006-8585 Japan
Tel: Tel: +81-11-681-2161 ext 455, Fax: +81-11-681-3622, E-mail: kurihara@hit.ac.jp

[b]*Otaru University of Commerce*
Midori 3-5, Otaru, 047-8501 Japan
Tel: +81-134-27-5385, Fax: +81-134-27-5385, E-mail: numazawa@res.otaru-uc.ac.jp

Abstract

Mobile agent technology has been the subject of much attention in the last few years, mainly due to the proliferation of distributed software technologies combined with the distributed AI research field. In this paper, we present a design of communication networks of agents that cooperate with each other for forwarding messages to the specific mobile agent in order to make the overall system location transparent. In order to make the material accessible to general intelligent systems researchers, we present the general ideas abstractly in terms of the graph theory. In particular, a proxy network is defined as a directed acyclic graph satisfying some structural conditions. It turns out that the definition ensures some kind of reliability of the network, in the sense that as long as at most one proxy agent is abnormal, there always exists a communication path, from every proxy agent to the target agent, without passing through the abnormal proxy.

As the basis for the implementation of this scheme, an appropriate initial proxy network is specified and the dynamic nature of the network is represented by a set of graph transformation rules. It is shown that those rules are sound, in the sense that all the graphs created from the initial proxy network by zero or more applications of the rules are guaranteed to be proxy networks. Finally, we will discuss some implementation issues.

Keywords:

graph transformation; agent; location transparency

Introduction

Mobile agent technology[1] has been the subject of much attention in the last few years, mainly due to the proliferation of distributed software technologies combined with the distributed AI research field and its potential applicability[2] , particularly in future intelligent telecommunication networks[3] . Currently, it is beginning to make the transition from research lab to mainstream software development practice. However, one of the problems that should be solved includes a requirement for "openness," meaning that the underlying infrastructure (the mobile agent systems) should be open to dynamic modification, in order to adapt to dynamic, evolutional nature of the Internet and network applications[4] .

In particular, location transparency of mobile agents is one of the functionalities required for such a dynamic nature. Suppose that two or more mobile agents are communicating to each other for some kind of cooperation. In this circumstance, it is desired that the communications will not result in failure even after some agents have moved to other places without any notice to other agents. Agent systems that satisfy this condition is said to be *location transparent.*

Rather than fixed, built-in mechanisms for location transparency, the framework of typical mobile agent systems adopts a mechanism based on dynamic evolution and maintenance of communication networks of user-defined proxy agents that cooperate with each other for forwarding messages to the specific target agent. This is particularly useful when the underlying mobile agent system is not location transparent, or when one wants to make the system open to dynamic customization and modification for location transparency.

In this paper, we present a new network construction scheme in this framework, and describe the evolution and maintenance of the networks that are reliable, efficient, and simple to implement. More precisely, a *proxy network* is defined graph-theoretically as a directed acyclic graph such that

(1) there exists a unique vertex with no outgoing edges,

(2) there exists a unique vertex with exactly one outgoing edge, and

(3) the remaining vertexes have exactly two outgoing edges.

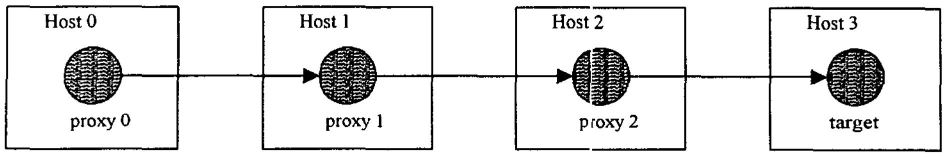

Figure 1- Simple communication path for forwarding messages

The vertex mentioned in the first condition corresponds to the target agent and all the other vertexes correspond to proxy agents. It turns out that this definition ensures some kind of reliability of the network, in the sense that as long as at most one proxy agent is abnormal, there always exists a communication path, from every proxy agent to the target agent, without passing through the abnormal proxy.

As the basis for the implementation of this scheme, an appropriate initial proxy network is specified and the law of the evolution and maintenance of proxy networks is formalized as a set of graph transformation rules. It is shown that those rules are sound, in the sense that all the graphs created from the initial proxy network by zero or more applications of the rules are guaranteed to be proxy networks, thus ensuring reliable location transparent communications among mobile agents. This scheme has been implemented in Java RMI and incorporated into Telepathy, a mobile agent system developed by the authors.

After briefly reviewing mobile agent systems and location transparency in the next section, we will formally present the definition of proxy networks and their properties in Section 3. Then in Section 4 we will present the evolution and maintenance of proxy networks, based on the initial proxy network and a graph transformation system, and discuss its soundness. The last section summarizes our work and briefly discusses some implementation issues.

Approach and Methods

Mobile agent systems and location transparency

Abstraction of software systems is a useful tool for making them easier to write, maintain and reuse. The concept of mobile agents takes the abstraction one step further than in current objected oriented programming to make it suitable for structuring distributed programs. Conceptually, a *mobile agent* is a software component that can autonomously move from place to place in a computer network. A *mobile agent system* is a platform for this technology and works as a kind of middleware on top of conventional operating systems. It is argued that the use of this technology results in distributed software that is robust to system failures (due to unreliable communication lines or devices) and intuitive to write in a well-structured manner[5] .

A general overview of the ideas and technologies behind mobile agents can be found in[1] . Several mobile agent systems have been developed, such as Telescript[6] , Agent Tcl[7] , and Aglets[8] .

A mobile agent system is *location transparent* if the communications between agents will be always maintained even after one or both of the agents have moved to other host computers without any notice to other agents. This means that agents can communicate with each other without caring about the location of other agents. This simplifies the structure of the communication programs significantly. Clearly, implementation of the location transparency requires some effective schemes for pursuing and accessing the current location of agents. At least the following three schemes have been identified[9] .

Logging: Before moving, the agents leave in the agent server the trail information containing the next location to which it is about to move.

Brute Force: The system searches for the location of the target agent by sending an appropriate query to every agent server.

Registration: The system keeps the locations of all agents in a unique directory server, updating the information each time an agent makes a move.

Clearly, the selection of a specific implementation scheme would greatly affects the performance and reliability of the overall systems, and the best choice would depend on the application domain.

There is a scheme of location transparency that is open to modification and customization by the application programmers. Such a scheme is based on dynamic creation of auxiliary, application-level agents called *proxies*. When an agent (let us call this agent the *target*) is about to make a move, it creates an agent whose only purpose is to work as a substitute for the target at the current place, receiving future messages for the target and forwarding them to the target wherever it may be. By assigning to the proxy the same agent identifiers as the target, the communications is kept location transparent, meaning that it causes no trouble and no special information update to other agents participating in the communications.

The program code for the target and proxies may be developed or customized by application programmers in a relatively straightforward manner based on object-oriented programming tools. The basic behavior of the system is to connect the target with the newly-created proxy each time the target makes a move. This process is repeated each time the target changes its place, yielding a path from the oldest proxy to the current target along a sequence of host computers. Messages received by the proxies are forwarded to the target along this path. Figure 1 illustrates the situation after the target has made three moves.

This scheme is simple and effective but has at least two problems. The first problem is its reliability. If there exists a proxy that is abnormal or does not work effectively, then the messages would not delivered to the target.

Another problem is its performance. The length of the communication path is equal to the number of the moves the target has made, so the path will get longer and longer as the target moves.

In the next section, we will present a new scheme that is still simple but effectively solves the two problems.

Design of Proxy Networks

In this section we will formally present the definition of proxy networks and their properties that characterize their reliability. We start with the following formal definition, which is a graph-theoretical abstraction of the message-forwarding networks we propose in this paper.

Definition 1 A proxy network is a finite, simple, directed acyclic graph (DAG) $G=(V, E)$ that satisfy the following three conditions. (The vertexes of V are called *agents*, and the directed edges of E are called *links*. By definition, a *simple* graph contains no parallel edges, which connect the same start and end vertexes; and an *acyclic* graph contains no circuits.)

1. There exists a unique agent (called the *target*) with no outgoing links. (The agents other than the target are called *proxies*.)
2. There exists a unique proxy (called the *special proxy*) with exactly one outgoing link. The link should be connected to the target.
3. The remaining proxies (called *normal proxies*) have exactly two outgoing links.

Some instances of proxy networks are shown in Figure 3.

The following theorem characterizes the reliability (or robustness) of proxy networks. Informally speaking, even if some proxy w is abnormal or does not work effectively, the messages received by any other proxies v can be safely delivered to the target without passing through w.

Theorem 1 For all pairs of distinct proxies v and w, $v \neq w$, there exists a path from v to the target without passing through w.

Proof. Start from v and follow an appropriate path as follows. At normal proxies, follow a link (chosen from two outgoing links) whose end vertex is not w. (The simplicity of the graph ensures the existence of such a link.) In this way you never visit w. This process can be repeated as long as you are at a normal proxy. However, since the network is acyclic and finite, you cannot repeat this process indefinitely. This means that at some time you will reach either the special proxy or the target.

If you are at the target, then you are done. Otherwise, you are at the special proxy, and then you can follow the unique outgoing link connected to the target. (Proof End)

Note that this proof constructively suggests a most efficient algorithm for choosing a path to the target: simply follow any link whose end vertex is not w.

Evolution and Maintenance of Proxy Networks

As long as the mobile agent stays at the birth place, there exists no proxy network in the mobile agent system. The first time the agent has moved to another place, the *initial proxy network* $G_0=(V, E)$ is created, where $V=\{t, s\}$ consists of the target t staying at the current place and the special proxy s remaining at the birth place; and $E=\{(s, t)\}$ consists of a link connecting s and t. See Figure 2.

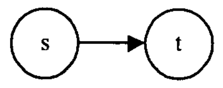

Figure 2 - The initial proxy network G_0

Every time the agent moves to another place, the network should be dynamically modified so that the resultant network maintains the properties required in Definition 1. Let us call this dynamic nature of modification the *evolution* (if V is extended by a new element) or the *maintenance* (if V is unchanged), and represent it by a graph transformation system.

The transformation system is defined as a set of graph transformation rules given in Definition 2, where we will use the notation as follows. We write $G \rightarrow G'$ if the graph G' can be obtained from G by a single application of a graph transformation rule. When a proxy network is literally denoted by $G=(V, E)$, we use the letters t and s to denote its target and the special proxy, respectively. The symbol + denotes the disjoint set union, meaning that $A+B$ implicitly requires that the sets A and B have no common elements. The set difference is denoted by the symbol $-$, i.e., $A - B = A \cap \overline{B}$.

Definition 2 The *proxy network transformation system* consists of the following four graph transformation rules. See Figure 3.

Move to a new place:

$(V, E) \rightarrow (V + \{u\}, E + \{(t, u), (s, u)\})$

This rule is applied when the agent (the current target) moves to a new place. The agent u is the target of the new network, while the target t of the current network will become the special proxy with the unique outgoing link to u attached, and the current special proxy s will become a normal proxy with a new link to u attached as the second outgoing link.

Move to the special proxy:

$(V, E) \rightarrow (V, E - \{(s, t)\} + \{(t, s)\})$

This rule is applied when the agent moves back to the place where the special proxy exists. The new network is formed by reversing the direction of the link (s, t). The special proxy s will become a new target, and the current target t will become the special proxy.

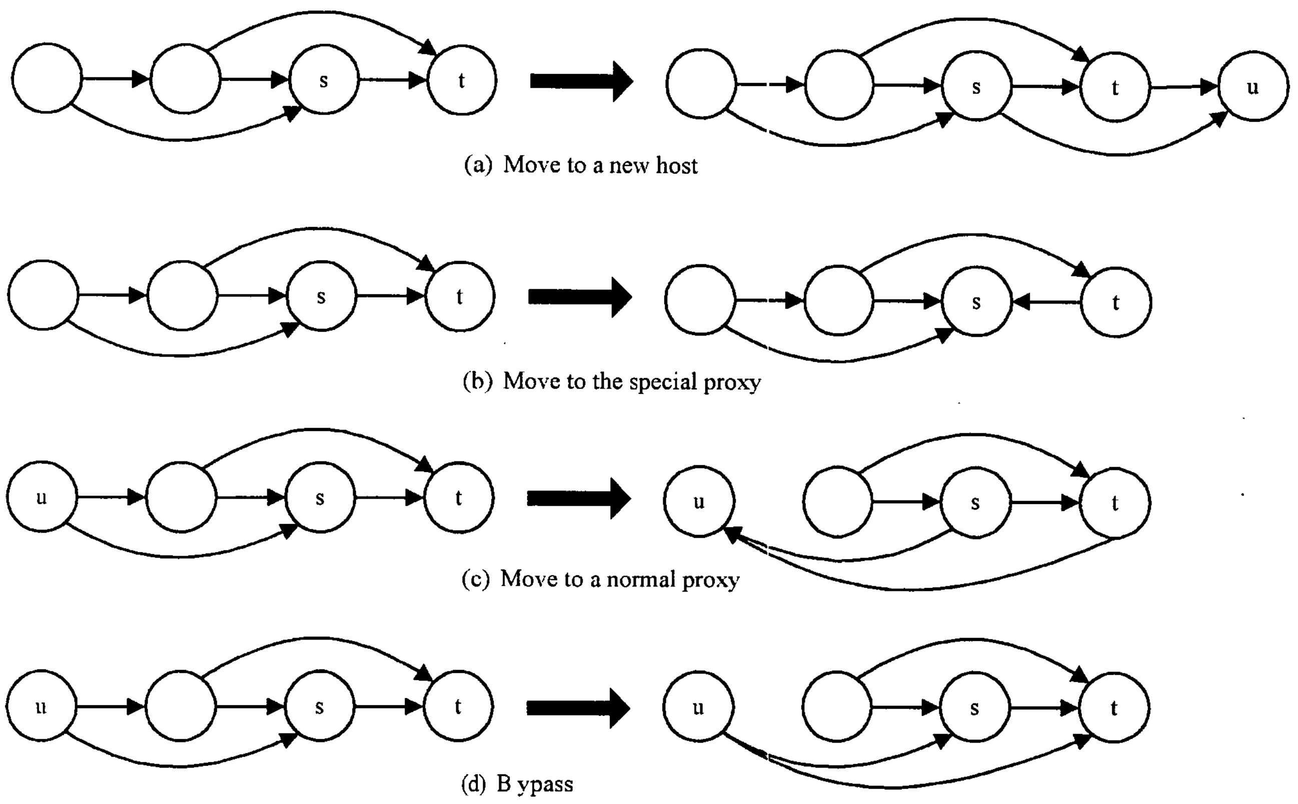

Figure 3 - The proxy network transformation system

Move to a normal proxy:

$(V, E) \rightarrow (V, E-\{(u, x), (u, y)\}+\{(t, u), (s, u)\})$ where $u \in V-\{t, s\}$, and $\{(u, x), (u, y)\}$ denotes the set of links outgoing from u.

This rule is applied in almost the same situation as the previous one except that it is a normal proxy that should exist at the place where the agent has moved. In the new network, the normal proxy u will become the target, with all the (two) outgoing links removed; and the agents t and s will become the special proxy and a normal proxy, respectively.

Bypass:

$(V, E) \rightarrow (V, E-\{(u, v)\} +\{(u, w)\})$ if $v \neq w$ and w is reachable from v.

The link (u, v) is replaced by a *bypass* link (u, w). This rule is useful for increasing the efficiency of the network, because it makes some paths to the target shorter, at the small cost of local modification of links. In practice, we can take the target t as w, as in the logging schemes in many existing systems. More precisely, we apply this rule just after a proxy u has received a message and forwarded it along a path to the target t. Actually, we can take all the proxies on this path as u, and all the links (u, v) contained in this path are replaced by the direct links (u, t) to the target, by application of this rule to each u.

Also, this rule may be applied when the performance of a proxy v has been significantly degraded or when the proxy is about to be deleted. In this case, the link (u, v) in the subpath $u \rightarrow v \rightarrow w$ will be replaced by (u, w).

We write $G_0 \rightarrow^* G$ if there exists a sequence $G_0 \rightarrow G_1 \rightarrow \ldots \rightarrow G_n = G$, $n \geq 0$, meaning that the graph G is obtained from the initial proxy network G_0 by zero or more applications of rules of the proxy network transformation system. The following theorem ensures the *soundness* of the transformation system, in the sense that the properties required for the proxy networks in Definition 1 are always maintained as long as they are transformed by this system.

Theorem 2 If $G_0 \rightarrow^* G$, then G is a proxy network.

***Proof* (*Sketch*).** In order to prove that a graph is a proxy network, you have to show the finiteness, simplicity, and acyclicity of the graph together with the three conditions in Definition 1.

The proof is based on induction. For the base case, we can easily verify that the initial proxy network G_0 is a proxy network. (The third condition for the normal proxies is vacuously true.) For the induction step, we show that if G is a proxy network and $G \rightarrow G'$ then G' is also a proxy network. This can be proved by simple case analysis for each graph transformation rule. (Proof End)

Discussion

Let us briefly discuss some implementation issues here.

Our scheme would fail if two hosts would be failing. However, our results can be easily improved. Actually, it is not hard to extend the proxy network so that the messages are forwarded to the target even if there are two failed hosts. (We just require normal proxies to have three outgoing links and introduce another kind of special proxy with two outgoing links.) Nevertheless, we think that the current version of the proxy network is useful enough, because if we can assume that such failure events occur independently with probability p, the probability of two hosts failing is the square of p, which is practically very small, assuming p is small enough.

It makes sense to think about the case where many agents roam the network. In this respect, we should distinguish abstract design (as presented in this paper) from its implementation. It is a good idea to have a single configurable proxy manager, rather than multiple copies of proxies, in an actual implementation. This paper never excludes this idea. In our formal framework of the system, we have never required that agents should be copied. This kind of issue is a general problem of implementation of data structure, particularly when the structure is defined in a mathematical framework. In the implementation, we often have a choice of how to retain multiple mathematical entities: retain them as copies or let a part of their structure be shared in order to save the memory space? The practical system should be implemented such that agents are not copied but share as much structure as possible, when there exist many agents in the system. For example, if the agents are implemented in an object-oriented way as instances of the same class (which is often the case), the class files can be shared at each host.

It is an important issue how to determine the failed hosts. In this paper, however, we take the position that this issue is a general problem common to virtually all the communication networks and not specific to the proposed scheme. Thus we would rather consider this problem an implementation issue which should be resolved for each network environment and application.

Conclusion

We have presented the general idea of new communication networks for location transparent mobile agents. Specifically, we have presented the formal definition of proxy networks and shown their reliability in Theorem 1. Then we have formalized the evolution and maintenance of proxy networks as the proxy network transformation system, and proved its soundness in Theorem 2. Note that Theorem 1 describes a *static* property of the network, while Theorem 2 clarifies the *dynamic* nature.

References

[1] Cockayne, W. R. and Zyda, M. 1998. *Mobile Agents.* Manning Publications.

[2] Johansen, D. 1998. Mobile agent applicability. In Proc. 2nd Intern. Workshop on Mobile Agents, *Lecture Notes in Computer Science* 1477: 80—98.

[3] Hayzelden, A.L.G. and Bigham, J. 1999. Future communication networks using software agents, In A.L.G. Hayzelden eds. *Software Agents for Future Communication Systems*, Springer-Verlag: 1-57.

[4] Minar, N. and Kramer, K.H. 1999. Cooperating mobile agents for dynamic network routing, In A.L.G. Hayzelden eds. *Software Agents for Future Communication Systems*, Springer-Verlag: 287-304.

[5] Appleby, S. and Steward, S. 1999. Mobile software agents for control in telecommunication networks. In A.L.G. Hayzelden eds. *Software Agents for Future Communication Systems*. Springer-Verlag: 270-286.

[6] White J. E. 1997. Telescript technology: mobile agents. In Bradshow, J.M. ed. *Software Agents*: MIT press.

[7] Kotz, D. 1997. Mobile agents for mobile Internet computing. *IEEE Internet Computing*. 1, 4: 58-67.

[8] Lange, D.B. and Oshima, M. 1998. *Programming and Deploying Java Mobile Agents with Aglets.* Addison-Wesley

[9] Aridor, Y. and Oshima, M. 1998. Infrastructure for mobile agents: requirements and design, In Proc. 2nd Intern. Workshop on Mobile Agents, *Lecture Notes in Computer Science* 1477: 38-49.