The ATL Postmaster: A System for Agent Collaboration and Information Dissemination

Jennifer Kay, Julius Etzl, Goutham Rao, and Jon Thies
Lockheed Martin Advanced Technology Laboratories
1 Federal Street
Camden, New Jersey 08102
+1 (609) 338-2014
{jkay, jetzl, grao, jthies}@atl.lmco.com

1. ABSTRACT

Most of the mobile agent work at Lockheed Martin Advanced Technology Laboratories (ATL) uses agents for information retrieval and dissemination. We have found that our two dominant sources of cost are bandwidth and services (e.g. database accesses). We designed the ATL Postmaster system to reduce these costs. The Postmaster system sits on top of other mobile agents systems and provides tools for agent collaboration and information dissemination.

There is a Postmaster resident on every machine that agents may visit. When an agent jumps to a new machine, it checks in with that machine's Postmaster. The agent gives the Postmaster its itinerary, and the Postmaster returns a possibly altered itinerary. A Postmaster may choose to reroute an agent if it knows a less-expensive alternate location the agent can visit to collect the same information. The Postmaster also provides a means for agents to share information with each other.

Autonomous Agents 98 Minneapolis MN USA

1.1 Keywords

Mobile Agents, Multi-Agent Communication Coordination and Collabora-tion, Knowledge Acquisition and Accumulation, Agent Architectures, ATL Postmaster.

2. INTRODUCTION

Most of the mobile agent work being performed at Lockheed Martin's Advanced Technology Laboratories (ATL) involves using agents for information retrieval. Agents are launched from one "home" machine, visit a series of servers to post or collect data, and send or bring this data back to the home machine.

For example, one of our major applications for agent systems is the Domain Adaptive Information System (DAIS) [1][7]. In DAIS, the servers manage large databases, and the agents visit multiple servers to make queries of those databases. DAIS was designed to be used by the military in environments where bandwidth can be extremely limited.

There are two sources of cost in this model. First, the cost of the actual services. In DAIS this is in terms of cycles on the servers, however it could equally be a charge levied for each database access. Second, the cost of the bandwidth required to transmit the agents and their data between machines.

In an ideal world, machines always have cycles available, services are free, and networks are always constructed of high-speed, high-bandwidth links. Unfortunately, even as technology improves, we often seem to be moving further away from this ideal. The more cycles we have the more we need, and the more bandwidth we get the more we use.

We wanted to build an agent collaboration and information dissemination system that reduces our costs by enabling us to do three things:

1. Reduce the number of duplicate requests sent to a particular server
2. Reduce the amount of duplicate data transmitted across the low-bandwidth lines
3. Provide the option for automatic updates, which would repeat a request every so often, without unnecessary transmission overhead if there was no new data to report.

We also wanted to build this system so that it is general enough to be applicable to any mobile agent system that met certain minimum requirements.

The result is the ATL Postmaster system. The Postmaster sits on top of an existing agent system and provides it with an agent collaboration and information dissemination layer.

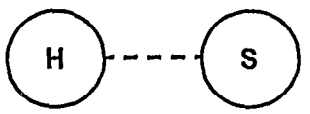
Each machine on a network has its own Postmaster. Agents arriving at a machine check in with the Postmaster before starting work, and check in again right before they leave the machine. The Postmaster examines the agent's plans, and may redirect the agent to another machine to collect the same data. Agents are redirected when the data to be collected is already cached on another machine (cheaper services) or when another machine is connected by a higher bandwidth link to the home machine (cheaper communication). The Postmaster can also use this check-in period to piggyback data on another agent that is heading in the appropriate direction.

While we are not aware of similar research in the agent domain, data replication methods are used in mobile computing to improve availability, performance, and reliability. File systems such as Bayou [5], Coda [4], and Rumor [3] enable users to read and update replicated files, even though this could cause conflicts with other copies of those files on the other side of a low-bandwidth or disconnected link.

3. POSTMASTER FEATURES

3.1 Automatic Updates

Often an agent wants to collect data from a server, and then return later to see if the resulting data has changed in any way. Consider the scenario shown in Figure 1. An agent is based on machine H (its "home" machine) and jumps to machine S to request services. The resulting data is to be transported back to machine H.



Machine S (a server) is connected to the home machine, H, by a very low-bandwidth link. An agent on H wishes to use a service on S to produce data, and bring the results back to H. The Postmaster system enables periodic updates of a service request. Instead of having to send an agent from H to A to perform the query again, the system can be set up so that the request is repeated at A, and results are returned to H as available. This reduces the amount of traffic sent across the low-bandwidth connection.

Figure 1. Periodic updates reduce costs due to bandwidth.

In the DAIS system, for example, the various databases are continuously being updated. New information may arrive and be entered into the database, and the same query, performed at five minute intervals, may return vastly different results.

In this situation, the primary concern is bandwidth consumption. If we know in advance that we will need periodic updates, we would prefer to avoid any unnecessary agent jumps, and duplicate data transmission.

There are two ways that the Postmaster can determine that new data will be returned as the result of a service request. The first is by periodically performing that request, and comparing the results with the previous request. The second method requires the agent adding new data to notify the Postmaster as the data is added, so that the Postmaster is immediately aware of the new data. The underlying agent system provides a method for determining whether data is new. Our initial trial uses a database with monotonically increasing index numbers to make this test simple.

3.1.1 Periodic Updates

When an agent makes a request for data from a server, it can also ask the Postmaster to arrange for the same request to be performed every so many minutes. If new data is returned as a result of the request, the Postmaster arranges for this to be sent to the destination specified by the agent. If no new data is returned, no action is taken.

By using the Postmaster, only new data is transmitted to the destination machine. The overhead of transmitting the agent and duplicate data across the network is eliminated.

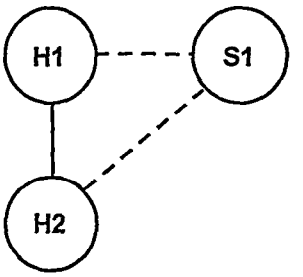
3.1.2 Immediate Updates

If data only appears at the server as the result of a delivery by some other agent, there is a second update option. If the other agent notifies the Postmaster that certain data is being given to the server, it may be possible for the Postmaster to check whether that data would be appropriate to forward on to the destination machine. In this situation data can be forwarded immediately. This saves bandwidth in the same way that the "periodic update" method does. In addition, it reduces the number of requests made to the server.

3.2 Redirection

The two costs in our system we are trying to reduce through the Postmaster are costs due to bandwidth constraints, and the cost of services. The Postmaster's ability to forward updates primarily reduces the bandwidth used, with potential reduction in service requests under the "immediate update" option.

Redirection focuses on both bandwidth and cost of service issues. Consider the scenario depicted in Figure 2. In this scenario, we have two "home machines", H1 and H2, and



Two "home machines", H1 and H2, and one server, S1. H1 and H2 are both located on a high-speed network. Other connections are across very-low-bandwidth networks. Agent A1 (whose "home" is H1) jumps to S1 and makes a request. Agent A2 (whose home is H2) later jumps to S1 and makes the same request. A2 is told by the Postmaster to retrieve the results of the request at H1. This reduces the number of costly server requests as well as the amount of traffic across the low- bandwidth link.

Figure 2. Agent Redirection reduces costly service requests and data transmission across low-bandwidth links.

one server, S1. H1 and H2 are connected by a high-speed link, and all other connections are across very-low-bandwidth links. Agent A1 (whose "home" is H1) jumps to S1 and makes a request. Agent A2 (whose home is H2) later jumps to S1 and makes the same request.

Because A1 registered its request with the Postmaster, and A2 checks in with the Postmaster before making the service request, the Postmaster can redirect A2 to H1 to collect the data. This reduces the number of costly server requests as well as the amount of traffic across the low-bandwidth link.

Agents actually inform each Postmaster of all of their current plans, so the redirection feature is more powerful than it may seem from the preceding example. Suppose that the system depicted in Figure 2 is augmented with two additional servers, S2 and S3, which are connected to all of the other machines by low-bandwidth links (see Figure 3).

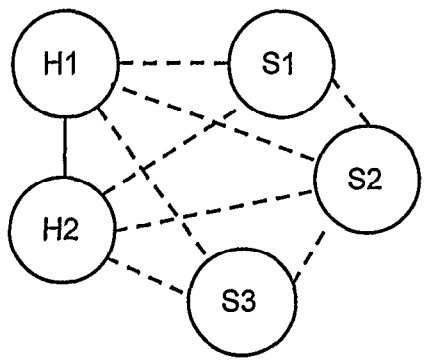


Figure 3. The network from Figure 2 has been extended to include two additional servers. The Postmaster system enables a more complex redirection to take place (see text for details).

We use the term "itinerary" to refer to the list of machines to visit, the corresponding tasks to perform at each machine, and the machines these results are destined for (this is similar, but not identical, to the use of the term introduced by the developers of the Concordia [2] mobile agent system).

Suppose Agent A3 has the itinerary shown in Figure 4: visit machine S1, and perform task T1 (a request for service from S1), then visit machine S2, and perform task T1, finally, visit machine S3 and perform task T2. The destination field on each row of the itinerary tells us that A3 will be sending the data back to H1. In addition, When A3 finishes, S1, S2, and S3 will all know the results of the tasks A3 performed should have all been returned to H1.

Next, suppose that after A3 performs its work, A4 is sent out with the itinerary shown in Figure 4. Our Postmaster will recognize the results of the first two lines of A4's itinerary can be found at H1. Rather than send A4 to S1 and S2, the Postmaster sends a new agent to H1 to collect the results that have been cached, and changes A4's itinerary so it only has to visit S3. If it is crucial to A4 that it get the absolutely most up-to-date data, then A4 can still visit S1 and S2 and make a request for any additional data available.

Itinerary: Agent A3		
Machine	Task	Destination
S1	T1	H1
S2	T1	H1
S3	T2	H1

Itinerary: Agent A4		
Machine	Task	Destination
S2	T1	H2
S1	T1	H2
S3	T1	H2

Figure 4. The itinerary for Agents A3 and A4. An itinerary consists of a list of machines to be visited, corresponding tasks to be performed at those machines, and the machine to which the results should be sent.

3.3 Sharing Information Tokens

We use the word "token" [6] to refer to a small chunk of information that might be of interest to an agent.

3.3.1 Leaving tokens with a Postmaster

An agent can leave an information token with a Postmaster to distribute to other agents that meet some particular specification. This specification can be as general as all agents or as specific as a single agent. When agents that match the specification check in with the Postmaster, the Postmaster gives the agent any appropriate tokens.

3.3.2 Sending Tokens to Another Postmaster

An agent can also ask a Postmaster to pass a particular token (with a corresponding agent specification) on to another Postmaster running on another machine. The first Postmaster arranges for the transmission of that token to the remote Postmaster.

4. SYSTEM REQUIREMENTS

There are certain basic features that an agent system must provide in order to be applicable to the ATL Postmaster system. A system that meets these requirements is said to be "Postmaster Compliant."

4.1 An Agent System Exists

It is important to emphasize that we are not trying to build a new system for mobile agents. Instead, we are trying to provide a package that will extend an existing agent system to incorporate information dissemination and agent collaboration. The Postmaster system does not provide any of the basic mobile agent technologies such as mechanisms to move the agents between machines, start agents running, start up new agents, etc.

4.2 Agents Understand Itineraries

As we discussed in Section 3.2, the agent must have some internal representation of its itinerary, which specifies where it will go, what it will do there, and where the results from that action will be brought. While the agent's itinerary does not have to be in the same form as the one used by the

Postmaster, the agent must be able to generate such an itinerary to give to the Postmaster, and must be able to adjust its plans based on an updated itinerary returned to it by the Postmaster.

4.3 An Agent's Itinerary May Be Reordered

The underlying agent system must not depend on an agent performing the tasks in its itinerary in a fixed order. As was demonstrated in Section 3.2, the Postmaster may split up a single agent into two agents which perform in parallel. There are no guarantees made about the order in which the itinerary is followed. If it is important to maintain an order for a given agent system that is sending data back to a collection agent, it is possible and fairly simple for the collection agent to oversee that the results were reordered as necessary to match the original itinerary. But if the values of the results are dependent on visiting machines in a particular order, the Postmaster system is not appropriate.

4.4 Items On An Agents Itinerary Must Be Independent

In order to enable the Postmaster to split an itinerary between multiple agents, none of the tasks on the itinerary can depend on the results of other tasks on the itinerary

4.5 There Is A Need For A Postmaster

The Postmaster system is a system to enable and enhance agent collaboration. There is a certain amount of overhead associated with the post office system. If bandwidth is unlimited, services are cheap, and agents never need to share information, the Postmaster will add unnecessary overhead, and should not be used.

5. TECHNICAL DESIGN

5.1 Basics

There is a Postmaster running on each machine that the agents can jump to. As soon as an agent jumps to a new machine, it checks in with the Postmaster on that machine. The agent gives the Postmaster its itinerary, and the Postmaster returns a possibly altered itinerary. Each Postmaster maintains four post office "bulletin boards": for-sale, wanted, most wanted, and most important. It uses these bulletin boards to keep track of the work that the various agents are performing, so that it can reroute agents as appropriate.

5.1.1 The For-Sale Bulletin Board

When an agent checks in with the Postmaster, the Postmaster makes note of the tasks and corresponding machines on the agent's itinerary on the for-sale bulletin board. When another agent comes by, the Postmaster can compare the new agent's intended tasks with those that it has recorded as already done on the for-sale bulletin board. If any of the new agent's tasks match something on the for-sale bulletin board, the Postmaster can choose to reroute the agent to the machine where the data is cached.

5.1.2 The Wanted Bulletin Board

When an agent wishes to receives updates for a particular query as the results change, he requests that the Postmaster put a message on the wanted bulletin board. The agent also specifies the frequency with which the server should be checked, and the destination machine. Periodically, at the frequency requested by the agent, the Postmaster starts up a new agent to perform the appropriate query. If any new results are found, they are transmitted to the Postmaster at the destination machine, who hands them to the query managing agent on that machine.

5.1.3 The Most Wanted Bulletin Board

Sometimes it is critical to receive an update about a change as soon as it happens. In this situation, the agent requests that the Postmaster put a message on the most wanted bulletin board. Every time the Postmaster is notified about new data, it compares this data with the data on the most wanted bulletin board. If there is a match, the data is returned to the Postmaster on the destination machine, who gives it to the query managing agent on that machine.

Because there is no guarantee that all new data will be filtered through the Postmaster, the Postmaster periodically starts up an agent to perform the appropriate query in the same way it does for the wanted bulletin board.

5.1.4 The Most Important Bulletin Board

All of the bulletin boards described so far contain only metadata, that is, they contain the query that was performed, or a query that an agent wants information about, but do not contain the results of any queries. In contrast, the most important bulletin board contains both metadata and data.

If an agent deems information to be of immediate importance to other agents, it can choose to put that information on the most important bulletin board. This information can take two forms:

Query and result. In this case, some (or all) of the results of a query are listed along with the query itself.

Agent type and information tokens. In this form, there are tokens that are on the bulletin board.

5.2 Postmaster/Agent Interface Commands

The ATL Postmaster is written in the Java language. In designing the Postmaster, we have endeavored to maintain a separation between the underlying agent system and the Postmaster system. Our goal is for the Postmaster system to be general enough to apply to a variety of agent systems.

On the other hand, the preceding description of the Postmaster makes it clear that it needs to have some specific knowledge about the task to be performed. For example, in order to determine whether a particular task has been performed before, the Postmaster needs to be able to compare two tasks and determine whether they are the same.

We get around this problem as follows. The Postmaster contains two Java interface classes that must be implemented by the agent system programmer. The first, P-IN, consists of calls that an agent can make to the Postmaster. This includes, for example, the initial call that the agent makes upon arriving at a machine. The Postmaster side of this call is the same for any agent

system. The Postmaster accepts an itinerary, compares it with its data, and returns a potentially altered itinerary. On the agent side, the agent's plans need to be formed into a Postmaster itinerary, and when a potentially changed itinerary is returned as the result of the call, the agent must be able to alter its plans accordingly.

The second Java interface class, P-OUT, consists of calls that the Postmaster can make to the agent system. For example, the Postmaster sometimes needs to create a new agent (when it wants to split an agent and redirect it to do part of its itinerary at another machine, see Section 3.2). The system designer who wishes to use the Postmaster system must provide a new-agent method that accepts an itinerary, and returns an agent that will follow that itinerary.

6. CONCLUSION

The ATL Postmaster is a system for cooperative information retrieval and agent collaboration. It provides an agent programmer with a set of useful features that sit on top of the underlying agents system, giving it the potential to be more efficient, and to reduce the cost of bandwidth and services.

Preliminary tests of the Postmaster in our laboratory have demonstrated a reduction in resource and network use. In addition, the initial Postmaster system has been integrated into a concept demonstration for the DARPA sponsored Small Unit Operations CyberAngels program. In the first CyberAngels demonstration, the Postmaster was used for automatic forwarding of threat warnings to individual soldiers from disparate distributed data sources, even when those data sources contained inconsistent data.

7. ACKNOWLEDGMENTS

The authors would like to thank Ken Whitebread, Martin Hofmann, Russ Lentini and Maria Ebling for their help and advice.

8. REFERENCES

[1] Hofmann, Martin O.; McGovern, Amy; and Whitebread, Kenneth R, "Mobile Agents on the Digital Battlefield," Second International Conference on Autonomous Agents (Agents '98), Minneapolis/St. Paul, May 10-13, 1998.

[2] Mitsubishi Electric ITA, Horizon Systems Laboratory, "Concordia: An Infrastructure for Collaborating Mobile Agents," First International Workshop on Mobile Agents 97, Berlin, Germany April 1997.

[3] Reiher, Peter; Popek, Gerald; Gunter, Michial; Salomone, John; and Ratner, David, "Peer-to-Peer Reconciliation Based Replication for Mobile Computers. *European Conference on Object Oriented Programming '96 Second Workshop on Mobility and Replication*, June 1996.

[4] Satyanarayanan, Mahadev, Kistler, James J., Kumar, Puneet, Okasaki, Maria E., Siegel, Ellen H., and Steere, David C., "Coda: A Highly Available File System for a Distributed Workstation Environment," *IEEE Transactions on Computers*, vol. 39, number 4, April 1990.

[5] Terry, Douglas B.; Theimer, Marvin M.; Petersen, Karin; Demers, Alan J.; Spreitzer, Mike J.; and Hauser, Carl H., "Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System," *Operating Systems Review*, vol. 29, number 5, December 1995.

[6] Utility Patent Application 80AT3213, Lockheed Martin Corporation, U.S. Patent Office.

[7] Whitebread, Kenneth R. and Jameson, Steve, "Information Discovery in High-Volume, Frequently Changing Data," IEEE Expert, vol. 10, number 5, October 1995.