

# Mobile Agents for Monitoring Distributed Systems

Delbert Hart  
University of Alabama in  
Huntsville  
Huntsville, AL 35899 USA  
+1 256 824 5160  
dhart@cs.uah.edu

Mihail Tudoreanu  
Washington University in  
St. Louis  
St. Louis, MO, 63130 USA  
+1 314 935 7536  
renu@cs.wustl.edu

Eileen Kraemer  
University of Georgia  
Athens, GA, 30606 USA  
+1 706 542 5799  
eileen@cs.uga.edu

Mobile agents can be employed to help a system adapt to diverse conditions and provide functionality that would otherwise be cumbersome and perhaps in-feasible. The benefits of mobile agents though are not without their own costs in performance and complexity. This paper gives an overview of our work in determining when mobile agents are appropriate for monitoring distributed applications. The high degree of variability resulting from the interaction between the users, applications, and the computing environment makes succinct descriptions of when mobile agents are appropriate very difficult. Our approach has been to identify the basic classes of variables in the evaluation formula and then to perform empirical tests to identify which aspects of the various components has the greatest impact. These results will then be used to identify guidelines as to when mobile agents are advantageous for the monitoring of distributed systems.

## 1. INTRODUCTION

The increasing complexity of software heavily burdens the reliability and performance as well as the development, maintenance, and even usage of the software. One way to overcome this complexity is to use monitoring, which through software visualization can create either real-time or offline representations of the computation in a form that can be easier to understand. Monitoring can also be used in conjunction with automated tools to adaptively tune performance. To deal with the growing complexity of software though monitoring systems need to have greater flexibility to support a variety of user needs for a wide range of applications and environments.

Monitoring distributed applications is a challenging task. To be effective online monitoring needs to adapt to the changing needs of the users while minimizing the effect on the application. The desire to limit the perturbation of monitoring is especially noticeable in long-lived and distributed computations. Further, the non-determinism and lack of a global clock in distributed applications make the monitor task more difficult.

The use of mobile agents has emerged as a technique for adaptively working with distributed systems. The use of mobile agents for monitoring has a number of advantages: 1) Responsiveness - agents are able to react locally to conditions at the application processes. 2) Transience - the ability to deploy agents on demand helps to minimize the overall cost of monitoring. 3) Customization - agents may be encoded at run-time; and thus can make use of application-specific information, permitting efficient solutions. 4) Mobility - the ability to migrate between processes makes agents well suited to distributed applications in which properties are not necessarily bound to a process.

PathFinder [?] (Figure ??) uses an exploratory visualization approach, which addresses the size and complexity of distributed systems by engaging the user as an active partner who guides the data collection and visual representation. The functionality of PathFinder is provided via modules (Figure ??) that are dynamically loaded at runtime. This ability to easily load and unload agent modules allows the use of mobile agent systems optimized for a particular role. A specialized agent system has less overhead than a single all purpose agent system. The disadvantage of utilizing multiple specialized agent systems is the complexity of supporting each of them. To mitigate this complexity we developed a basic mobile code model from which the agent systems are derived.

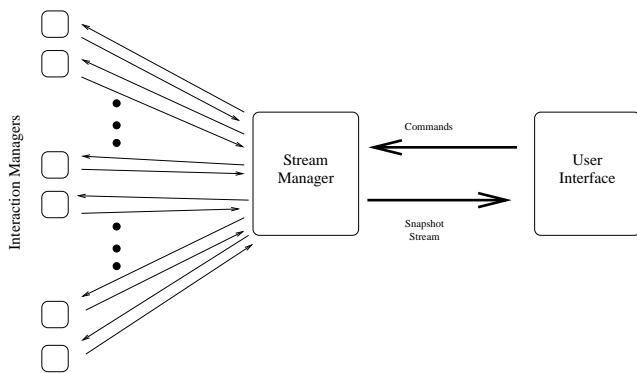
The agent model[?] has two kinds of entities: *agents* and *milieux* (agent servers). The distinguishing feature of this model is that it is specialized to be embedded within a monitoring library. This constrains the milieu to having to possibly share the thread of control with the application and utilizing an abstract asynchronous form of communication with other milieux. Agents access information about the application via specialized agents called *avatar* agents and *event* agents, providing synchronous and asynchronous access respectively. The milieu provides services that allow an agent to execute inside of it, to interact with other agents, and to move to other milieux. A milieu does not provide explicit support for inter-milieu communication, other than supporting agent migration.

## 2. EVALUATION CRITERIA

There is a wide range of factors that affect the performance of a distributed monitoring system. A significant factor is what application is being monitored. Different applications have different characteristics in terms of how the state evolves over time, how the processes communicate with each other, how the processes interact with other entities

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AGENTS'01, May 28-June 1, 2001, Montréal, Quebec, Canada.  
Copyright 2001 ACM 1-58113-326-X/01/0005 ...\$5.00.



**Figure 1: PathFinder's architecture.** The User Interface sends a sequence of commands to the Stream Manager who coordinates actions at the Interaction Managers. The Stream Manager also collates the local snapshots from the Interaction Managers into a sequence of snapshots that the User Interface creates visualizations from.

(such as users or other systems). Some distributed applications have tight constraints on the amount of resources, such as hard real-time deadlines, which affect how the monitoring system can be used.

Another factor that affects a distributed monitoring systems is the demands placed on it by a user. Each user has their own agenda and uses the monitoring system differently to accomplishing their agenda. The computational environment in which the application runs can have a significant impact on performance. A simple example is the communication model employed, assumptions valid in a FIFO lossless network differ from those in a lossy network.

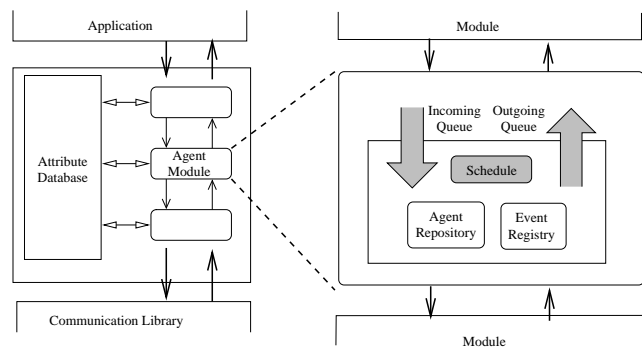
Trade-offs are associated with the use of every agent system, and the performance of one may not necessarily reflect the performance of all. One aspect of our work is to distinguish what characteristics apply to all mobile agents systems, and which are limited to a particular implementation. Within a particular agent system there is a large number of ways in which to accomplish a task. Evaluating these different mobile agent strategies will provide insight into how agents can be most effectively employed.

The last factor that we consider is how mobile agent monitoring compares to non-agent solutions. Mobile agents are not a silver-bullet and any integration needs to combine agent and non-agent solutions in order to best utilize the strengths of each.

### 3. EVALUATING MONITORING AGENTS

This section presents an example of how we have been evaluating mobile agents. For an initial application we chose one in which a token moves among processes. The problem of discovering a resource in a network is similar to finding a particular property in a distributed application[?] and seemed a good candidate for an application in which mobile agents would prove advantageous. The user in this scenario is interested in performing two tasks: 1) finding the token and 2) tracking its movement.

In order to evaluate different agent strategies we encoded them as a set of agent blueprints[?]. PathFinder's ability to utilize multiple agent modules (Figure ??) allows differ-



**Figure 2: An agent module can be one of many modules installed in the Interaction Manager to provide monitoring and steering related services.** The agent module provides the bridge between the application and the milieu.

ent agent systems to be used concurrently. Although this introduces additional perturbation it allows the direct comparison of different agent systems to each other. The difficulty of running the tests serially is that the conditions in a distributed system change with every execution due to the non-determinism. Agent modules supporting agents written in Perl[?] and JESS[?] have been developed.

Similarly, agent modules can be compared to the functioning of modules consisting of compiled C++ code. The C++ modules will execute more efficiently at the local processes, but lack the flexibility in filtering of the agent modules.

Embedding mobile agents within a monitoring system provides a new degree of flexibility in the tasks that can be performed and how they are done. Judicious choices must be made about when to use mobile agents, what type of mobile agents, and how they are coordinated. We have begun empirical testing of these monitoring agents in order to gain insight into the general principles that should guide these choices.

### 4. REFERENCES

- [1] E. J. Friedman-Hill. *Jess, The Java Expert System Shell*. Sandia National Laboratories, Livermore, CA, Mar. 1998. Version 4.0.
- [2] D. Hart, E. Kraemer, and G.-C. Roman. Consistency considerations in the interactive steering of computations. *International Journal of Parallel and Distributed Systems and Networks*, to appear, 1999.
- [3] D. R. Hart. *Exploratory Visualization of Distributed Systems*. PhD thesis, Washington University in St. Louis, August 2000.
- [4] K. Jun, L. Boloni, K. Palacz, and D. C. Marinescu. Agent-based resource discovery. In *9th Heterogeneous Computing Workshop*, Cancun, Mexico, May 2000. IEEE.
- [5] B. Kohn, E. Kraemer, D. Hart, and D. Miller. A sensory-motor approach to the specification of monitoring and steering interactions with distributed systems. In *Conference on Parallel and Distributed Computing*, Las Vegas, NV, Nov 7-9 2000. IASTED.
- [6] R. Schwartz and L. Wall. *Programming Perl*. O'Reilly and Associates, 1994.