Using Autonomous Agents to Improve Efficiency and Robustness in Slow, Unreliable Networks

Daria Chacón, Benjamin Bell, and John McCormick

Lockheed Martin Advanced Technology Laboratories
1 Federal Street
Camden, NJ 08102
[dchacon, bbell, jmccormi]@atl.lmco.com

Problem Description

A number of environmental factors commonly present in military and particularly tactical military contexts affect the ability of applications to operate robustly and efficiently:

- Machines hosting agents may shut down due to damage or power loss
- Hostile forces may disrupt communication links
- Communication links may be exceedingly slow, since some signals are transmitted over tactical radio links with maximum data rates as low as 4.8 kbps
- Tactical processors and network services may be overloaded, impeding the completion of agent tasks.

Mobile agents can significantly reduce load on networks in environments where bandwidth is limited and only a small percentage of the information available in the network is needed by an application. (Kotz et. al. 2000). However, to exploit this capability in military tactical networks, which are particularly fragile and dynamic, agents must be highly robust and fault-tolerant.

In this abstract, we present a brief case study that illustrates problems common to military environments, and we describe ideas for improving agents' decision-making capabilities to enable them to operate efficiently and robustly in the face of degraded network conditions.

Background

Lockheed Martin's Advanced Technologies Laboratory (ATL) has over five years of experience in developing and deploying mobile agents in military applications. Due to the fragility of tactical networks and the dynamic nature of military field operation centers, ATL has often encountered the need for robust, efficient agent operation in the face of catastrophic network failures and poor network performance.

EMAA: The Extendable Mobile Agent Architecture

ATL has deployed over fifteen agent applications built on our Extendable Mobile Agent Architecture (EMAA), a flexible, Java-based mobile agent development architecture. EMAA provides simple, standardized mechanisms for an agent to migrate from one computing node to another and to use resources at that node. At EMAA's core lies a Dock that provides an execution environment for agents, handles incoming and outgoing agent transfers, and controls agent access to services.

EMAA allows users to define agents, services, and events. Under the EMAA framework, agents are built from small, easily reused tasks that combine to meet a user's goal. An agent's tasks are structured within an itinerary. Agents may be mobile, and typically use stationary services. Services may implement connections to external systems (e.g., databases or legacy applications), provide complex functionality, or carry out other functions, but they are not primary actors. Goal-oriented and directed activity is generally held to be the function of agents. Both agents and services may send and receive events.

In the following subsections we discuss two EMAA features which support robust agent operation under undesirable conditions.

Itineraries. EMAA itineraries are composed of states (each of which may contain one or more tasks) and transitions between states; they are structured as finite state machines. An EMAA agent contains internal memory that encodes data dependencies among tasks: the output of one task may be used as input to another task, etc. Itineraries employ some decision logic to determine whether and where to execute states. Some states must be executed on a specific machine, and others must be executed on any machine where the resources needed to support execution of the encapsulated task (or tasks) exist.

Reliable Event Messaging. To address the need for remote control and execution monitoring of mobile agents, ATL developed the Event Transceiver Server (ETS), an event handler residing on each dock with registered listeners for agents. The ETS provides reliable event messaging for mobile objects, both sending and receiving. It is invaluable for remotely or locally controlling, monitoring, and retasking agents.

DAIS: A Case Study and Problem Illustration

The Domain Adaptive Information System (DAIS) employs mobile agents for information discovery and dissemination in a military intelligence (MI) network. Counter Intelligence/Human Intelligence (CI/HUMINT) operations gather information from prisoners of war, cooperating civilians in the secured rear area, and long-range surveillance teams inserted into the enemy's territory. Effective CI/HUMINT operations are essential to the protection of friendly forces and to successful operation in hostile arenas.

Until recently, CI/HUMINT operations depended on hand-delivered paper reports, telephone messages, and transcription. Costly delays of several hours often occurred between the collection of information and its integration into planning processes. DAIS was developed to improve the efficiency and speed of military tactical operations.

DAIS illustrates a number of problems pertinent to a discussion of robust agent autonomy. Communications among many tactical elements were carried over tactical radios, which often moved in and out of router range. Communication connectivity to remote teams was intermittent, and links provided painfully slow maximum data rates, ranging from 4.8 to 64 kbps depending on range and hardware. Processing power on command elements varied widely, from low-powered laptops to powerful servers.

DAIS employed several pragmatic strategies to mitigate the problem of unreliable network connectivity. Before sending an agent over a communication link, the Dock verified that the destination host was reachable, thus averting costly timeouts. If a connection was unavailable, an agent could request the Dock to notify it when the connection became available again. If the agent was composed of independent tasks, it could split its itinerary to create a second agent assigned to complete the non-machine-dependent tasks. After completing their tasks, both agents reunited to combine and deliver results.

Despite these efforts, DAIS did not address the challenges of handling machine failures, avoiding slow communication links, and overloaded processors. Some approaches taken in DAIS to mitigate network problems break down for complex agents with itineraries composed of interdependent tasks. The lessons we learned in deploying DAIS and other military agent applications help us articulate solutions to the challenges of robust agent functionality in distributed, unreliable networks.

Approaches to Increasing Agent Robustness and Efficiency

Intelligent Mobility

We define intelligent mobility as the ability of an agent to adaptively choose an execution path in real time, considering both the relative priorities of different execution paths and the condition of the network. An effective implementation of intelligent mobility allows an agent to avoid slower communication links and processors, and react to communication failures by choosing an alternative host or execution path.

As an EMAA agent implements its itinerary, it executes the initial state and follows one of possibly several valid transition paths to a next state. A state may be bound to a set of host constraints or a set of resource constraints; if host constraints are given, the state must run on one of the specified machines. If resource constraints are given for a state, the agent must discover the set of machines at which all of the required resources are present. The agent is supplied with transition logic that may evaluate valid transitions based on host availability, transition utility, and transition feasibility/efficiency.

Host availability. If no host is available to support a given transition, the transition logic should eliminate it from consideration.

Utility. The transition logic can obtain the relative utility of the transition (utility describes the fact that one transition may be inherently preferable to another; e.g., one data source may be more frequently updated than others), which may have been calculated or assigned at runtime.

Feasibility/Efficiency. The transition logic may determine the feasibility and efficiency of following the transition based on environmental conditions. Variables that can affect the feasibility and efficiency of a transition include:

- Cost of using a service
- Reliability of service
- Reliability of communication link to the service
- Cost (speed) of using communication link

An agent may factor the first two variables into the feasibility/efficiency calculation if the agent architecture employs a resource control strategy that permits services to impose usage costs to prevent thoughtless or unintentional overuse. The resource control strategy also requires services to publish self-descriptive information including failure rates (if applicable), usage penalties, etc. This is similar to market-based resource allocation schemes such as that proposed in Bredin et. al, 2000. The Dock or a designated link information server may retain statistics and status data on communication links, including mean time to failure, mean failure duration, average throughput, and link availability status. This allows an agent to consider the last two variables listed above. An agent's transition logic might weigh any combination of these four variables to assess feasibility and efficiency of each transition.

For each transition, the transition logic may then consider utility together with feasibility/efficiency to determine the transition's overall effectiveness, where effectiveness is accomplishing the agent's overall objective. Transition logic will generally be agent-specific or application-specific, since the relative importance of efficiency to utility in transition selection will vary among applications and agents.

Employment of Failure Recovery Strategies

Agents in tactical networks must robustly react to inevitable machine failures. EMAA agents contain hooks for reaction policies to be executed in special situations including migration, remotely initiated halt, or catastrophic errors that inhibit execution of the itinerary. In case of a "soft" machine or Dock failure (a non-instantaneous failure which affords applications time to close politely), an individual agent select and employ and appropriate recovery strategy. The Dock will notify all hosted agents and servers that a shutdown is imminent, and agents may react accordingly. An agent must depend upon other agents or components for help in handling a "hard" failure (an immediate, unexpected system shutdown).

Automatic Restart Upon Machine Availability. In case of either a hard or soft failure, an agent may register a request with the Dock to restart the agent as soon as the machine becomes available again. This is appropriate for agents that can be restarted without restoration of state.

Migration to a Different Machine. In the case of soft failure, an agent may evaluate other execution options for the task to be executed, perhaps moving to another machine that offers services needed to complete task execution.

Reinstantiation by Monitoring Entity. An agent remotely monitored by another agent, a service, or an application may be tracked and reinstantiated by the monitoring entity upon failure (the monitoring entity assumes a catastrophic failure when it fails to receive status events within a desired time interval). This is appropriate for agents that need not preserve full state data (for example, unprocessed query results). Because EMAA's Distributed Event Messaging System allows task-level monitoring, it is possible to restart the agent after the last successfully executed task.

Restoration of Full, Checkpointed State. Some agents may require their full state preserved upon restoration. An agent collecting a time-tagged process snapshot, for example, could not be reinstantiated without losing unrecoverable, time-sensitive data. The agent must checkpoint its state data with either a local, persistent store or a remote store.[1] If it is monitored remotely by another entity, or a local persistent entity, that entity can restart the agent with full state information. This solution should be used sparingly, since it may tax storage resources or increase bandwidth usage, reducing the benefit added by agent mobility.

Summary

We have presented the intelligent mobility concept as a means for agents to improve execution efficiency and exploit alternatives in case of resource unavailability. We suggest failure recovery strategies to improve robustness of overall application performance in the face of machine failures. Figure 1 illustrates the specific problems that our proposed approaches seek to overcome.

Common Problems	Proposed Solutions
Reaction to temporary host failure	Failure handling strategies
Temporary link failure detection	Failure handling strategies
Temporary link failure handling	Intelligent Mobility
Slow comms link avoidance	Intelligent Mobility
Slow processor avoidance	Intelligent Mobility

Figure 1. Intelligent mobility and failure handling strategies are complementary approaches to addressing problems common to tactical networks.

[1]Explicit checkpointing (periodic storage of state) for itinerary-based agents employing weak mobility has been described by Silva, Batista, and Silva (2000). The design of the EMAA agent itinerary lends itself well to implicit checkpointing between state executions, although that feature has not yet been fully implemented.

Ensuring reliable agent operation in a fragile distributed environment poses many challenges not yet addressed. For instance, interrelated failures are likely to occur on tactical networks, since damage to one machine is often accompanied by damage to other machines on the network. If permanent or semi-permanent link or host failures occur, services and agents may need to be entirely replicated elsewhere on the network. Additional problems may arise when a machine thought to have permanently failed becomes available after having been replicated elsewhere.

Despite the remaining challenges, mobile agent architectures offer inherent advantages in coping with unreliable network environments. An agent requires only a brief connection to migrate, whereas client-server solutions generally require a sustained connection over unreliable links. Each EMAA agent operates within its own thread, so a single failure poses no threat to other independent agents or interacting systems. Because an individual agent can be persistent, it can wait and retry communication links, or choose a different execution strategy. Itinerary-based agents are inherently amenable to high-level checkpointing, since an agent's itinerary is naturally divided into distinct execution states. Building on these advantages by adding to agents the ability to evaluate choices in reaction to unexpected conditions in their environment will add greatly to the robustness and efficiency of agent applications.

Acknowledgements

Work on DAIS was supported by DARPA Contract N66001-95-C-8636. We thank Dr. Susan McGrath for her initial direction on this effort, and Dr. Martin Hofmann for his helpful review of this abstract.

References

Bredin, J., Maheswaran, R., Imer, C., Basar, T., Kotz, D., and Rus, D. 2000. Game-Theoretic Formulation of Multi-Agent Resource Allocation. In *Proceedings of the Fourth International Conference on Autonomous Agents*, 349-356.

Hofmann, M.O., McGovern, A., Whitebread, K.R. 1998. Mobile Agents on the Digital Battlefield, In *Proceedings Second International Conference on Autonomous Agents (Agents '98)*, 219-225, Minneapolis/St.Paul, MN.

Kotz, D., Jiang, G., Gray, R., Cybenko, G., and Peterson, R. 2000. Performance Analysis of Mobile Agents for Filtering Data Streams on Wireless Networks. ACM Workshop on Modeling, Simulation, and Analysis of Wireless and Mobile Systems.

McGrath, S., Chac6n, D., Whitebread, K. 2000, Intelligent Mobile Agents in the Military Domain. In *Proceedings of the Autonomous Agents 200 Workshop on Agents in Industry*, Barcelona, Spain.

Silva, L., Batista, V., Silva, J. 2000, Fault-Tolerant Execution of Mobile Agents., In *Proceedings of the International Conference on Dependable Systems and Networks.*