Self-Evaluating Space and Robotic Agents

Dr. Ronald Marsh
Computer Science Department
University of North Dakota
Grand Forks, ND 58201
Voice: 701-777-4013
Fax: 701-777-3330
Email: rmarsh@cs.und.edu

Amarender Challa
Computer Science Department
University of North Dakota
Grand Forks, ND 58201
Voice: 701-777-4107
Fax: 701-777-3330
Email: challa@cs.und.edu

Abstract

Our simulation consists of three parts: a satellite, a broker and a ground station. The communication is through sockets using TCP/IP protocols. All messages sent by satellites are received by the broker and inserted into a queue and are later forwarded to a ground station for processing. When the server acknowledges the receipt of a message by a ground station, the message is removed from the queue. As the number of satellites increases the queue size increases and to process the increased number of messages we require additional ground station(s). In this paper we discuss the method used to determine when we should add or remove ground stations from the system to ensure efficiency of operation and the timely delivery of messages.

1. Introduction

Our goal for this work was to develop a metric that would allow agents in an agent-oriented architecture to be self-aware regarding their workload and to be able to predict the near-term growth or decay of that workload. To evaluate our metric we modeled the environment where an Earth observing satellite (a type one agent) acquires images and sends the images to a ground station (a type two agent) that archives the data. While the satellite continuously sends messages, the ground station only receives them when the satellite and ground station are in a direct line of sight of each other. We have made the assumption that the environment is very dynamic and that the orbit and number of satellites is not fixed. Neither is the location nor the number of ground stations fixed. Any ground station is capable of receiving data from any satellite as long as they are in line-of-sight of each other. Additionally, we have no a priori information regarding the

amount of data traffic that may occur at any given time. There are times when no satellites are in view and the ground station is idle and times when several satellites are simultaneously in view and the ground station is overloaded and not capable of the task.

Specifically, the goal was to develop a metric that would allow the ground station to be self-aware regarding its current workload as well as the trend in the near-term growth or decay of the workload to predict if additional ground stations were needed or if any existing additional ground stations could be deleted from the system.

This paper is organized as follows. In section 2, we provide a description of cooperating agents. In section 3, we discuss our simulation and present our results. Our conclusion is given in section 4.

2. Background

For efficient management and development of our system, we used the concept of multi agents where independent software agents interact with each other to achieve common goals. According to Jennings (1998), an agent is a computer system, situated in some environment, which is capable of flexible autonomous action in order to meet its design objectives. This definition implies that an agent must be able to react to events that occur in its environment in order to maintain/achieve its goals. Further, along with being reactive, an agent must also be proactive. That is, it must be able to take the initiative and be opportunistic when necessary in order to meet its stated goals. Finally, agent communication has been shown to be an important factor in coordinating efficient group behavior in agents (Iba et al, 1997, Iba, 1998, and Yager, 1997). An agent must be social. It must be able to interact with other agents and with users to coordinate to meet common goals, and to negotiate to resolve conflicting goals. When dealing with real time systems (as in our case), the flexible and responsive nature of agents in a multi agent system is ideal. For example, military training simulations incorporate agents to represent enemy forces. These “enemy agents” react to the changing environment and they coordinate and negotiate to meet the goal of attacking or responding to an attack from the other forces taking part in the simulation.

Cooperating agents in a multi-agent system exist within a framework in which they can communicate, coordinate and negotiate to meet their goals. We call this framework the agent architecture. Several models for agent communication have been developed: agent to agent, agent broker, and agent matchmaker (Decker et al, 1996). In an agent to agent model, each agent knows the name of any other agent with which it might need to communicate. This is a completely distributed model in that there is no central coordinator among the agents. The latter two models both fall into a category of agent facilitators, in which a special agent is tasked with finding agents to fulfill services required by requestor agents. An agent broker is an agent that facilitates communication among other agents. In this model, each agent registers with a broker and advertises the services that it can perform on behalf of other agents. When an agent requests a service from the broker, the broker passes the request along to the agent that provides the

requested service. If no such agent exists, that is, if an agent has not advertised the requested service to the broker the broker responds to the requesting agent with a message indicating so. An agent matchmaker works in much the same way as an agent broker. However, when a request for service is made to a matchmaker, the matchmaker agent passes along a reference to the agent that can provide the service. That is, the matchmaker puts together, or matches, agents that can work together. Once this match is made, the agents can communicate with each other directly without the use of the matchmaker.

We use the concept of mobile agents (Harrison, 1995 and White, 1997), which are also robust and flexible. Once a user has created an agent, it can run autonomously and asynchronously without intervention from the user. Mobile agents provide a reliable transport between a client and server without necessitating a reliable underlying communication medium. They can also react autonomously to changes in their execution environment, and are therefore more flexible in their operation.

3. Simulation and Experimental Results

In this section we will discuss the satellite simulation (SATSIM) environment that we designed for evaluating the effectiveness of message transmission and receipt by agents. SATSIM was designed to operate as a standalone simulation model and allows the user to configure the environment to generate different amounts of data allowing the analysis of different possible situations.

SATSIM was written in the C++ language and uses the SOLID (Interference Detection Library Copyright © 1997, 1998 Gino van den Bergen) library for determining when objects are within line of sight. All agent interactions are through sockets using the TCP/IP protocol. SATSIM has two primary components, a server (our broker) that creates sockets and waits for connections from the agents and the agents. Depending on the command-line arguments used when initiating them, agents are either satellites or ground stations. Ground station parameters include the name of the machine hosting the broker to connect to, the latitude and longitude of the ground station, and a unique ID. Satellite parameters include the name of the machine hosting the broker to connect to, the initial latitude and longitude, the altitude, and a unique ID. Once the simulation is started, satellites continuously send messages to ground stations; however, ground stations can only receive the messages if they are in line-of-sight. Figure 1 shows the environment with one ground station and two satellites. The red line indicates which agents are in-sight of each other.

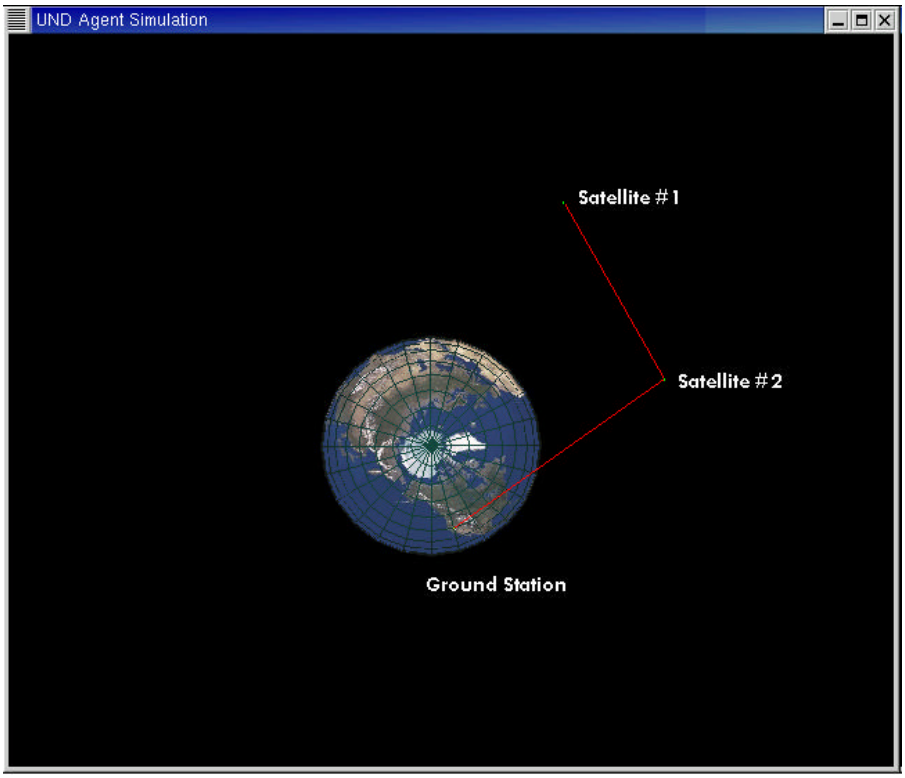

Figure 1: Simulation environment.

The major problem with a real-time simulation, such as this one, is the non-independence of the agents. Agent independence was obtained by placing the different components on different machines. This, however, resulted in the loss of messages. In part due to collisions on the network, but more importantly due to the arrival of simultaneous messages at the broker. We drastically reduced this symptom by via simple queuing theory. All satellite messages received by the broker are stored in a FIFO queue. The messages are then passed on to a ground station for processing.

Tables 1, 2, and 3 show how the message processing time and number of satellites affect the queue size. The simulation ran until the satellites sent a specific total number of messages. Once the desired number of messages was sent, the simulation stopped and the data compiled. The ground station assumed a constant (column 4) message processing time. Each satellite generated messages at a rate dependent on its altitude.

Table 1: Simulation with 2 satellites and 1 ground station.

% of messages received by server	% of messages processed by ground	% of messages in queue	Processing time for messages (msec)
99.6	99.6	0	0
99.7	91.02	8.7	0.20
99.65	59.17	40.47	0.40
99.5	41.15	58.35	0.60

Table 2: Simulation with 4 satellites and 1 ground station.

% of messages received by server	% of messages processed by ground	% of messages in queue	Processing time for messages (msec)
98.27	98.27	0	0
97.37	49.05	48.32	0.20
97.85	30.15	67.7	0.40
97.85	23.12	74.7	0.60

Table 3: Simulation with 8 satellites and 1 ground station.

% of messages received by server	% of messages processed by ground	% of messages in queue	Processing time for messages (msec)
97.82	97.82	0	0
97.7	32.97	64.2	0.20
97.8	19.17	78.85	0.40
98.34	15.42	82.9	0.60

As can be seen from the results, a small percentage of messages are still lost. However, we believe this to be an artifact of our network (collisions) and that this issue could be easily resolved in a free-space network environment. Hence, we ignore these lost packets. The tables also show that for a given processing time, as the number of satellites increases, number of messages sent to the broker at a particular instant of time increases, this in turn increases the queue size in the broker.

Two of the goals of the environment modeled are to acquire/receive as much data as possible from the satellites and to model a highly dynamic environment where none or many satellites may be in view at any given time. Obviously, we could simply assume a very large queue in the broker or we could simply add new ground stations to help process the additional messages stored in the queue. However, another goal was to be efficient, therefore, a key issue was if and when to add additional ground stations and if and when to remove the additional ground stations. The approach we decided to investigate was to monitor the time-sampled growth of the queue and to use this slope to determine when to create or destroy ground stations. Figure 2 and 3 depict the system evolving over time.

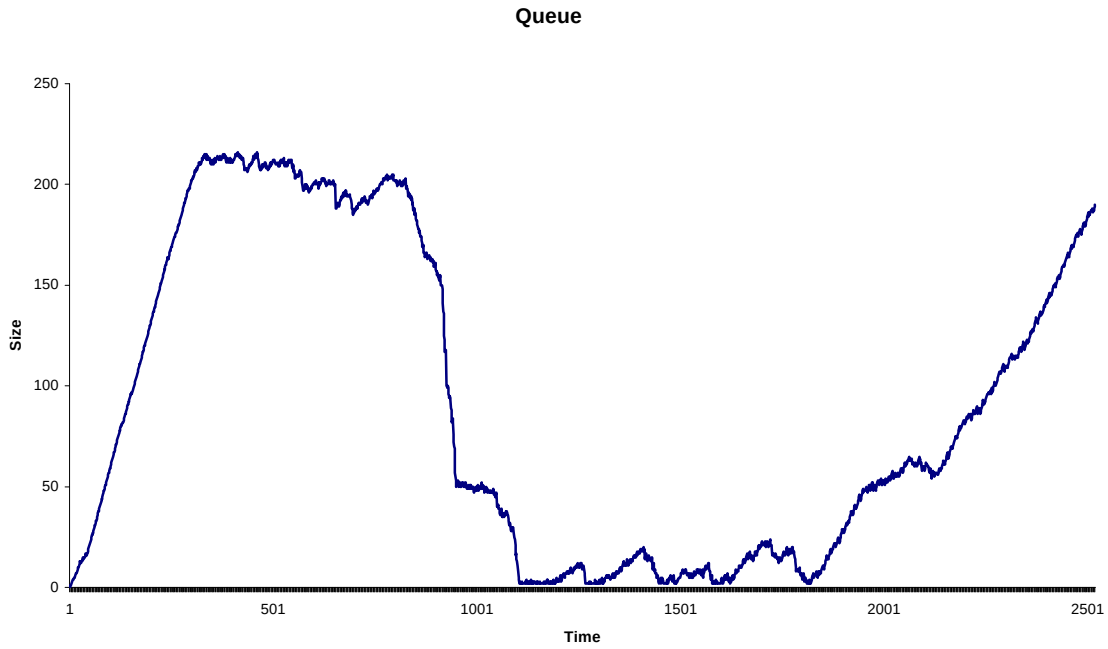

Figure 2: Queue size.

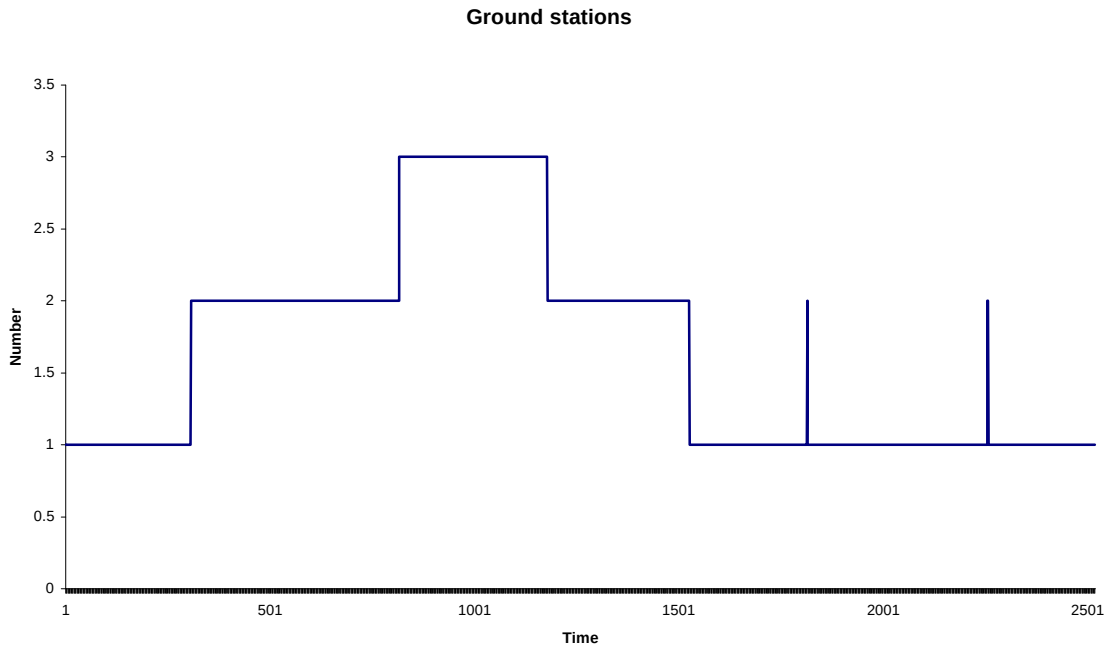

Figure 3: Number of ground stations.

Figure 2 depicts the growth and decay of the queue (driven by the number of satellites in view of the ground station(s) at any given time). Figure 3 shows the number of ground stations present at any given time. Initially, we have one ground station that is receiving very few messages, however, as more satellites come into view the queue grows rapidly.

Eventually the satellites move out of view, the message traffic reduces, and the ground station gets a chance to "catch up" by processing the messages in the queue. However, this scenario only occurs under ideal conditions. In most of our simulations the satellites come back into view before the ground station processed the remaining messages. Over a short period of time the size of the queue becomes extremely large.

To constrain the queue size we determined that we could easily decide when new ground stations should be added by monitoring a time-averaged growth of the queue. Using the data presented we determined that a slope less than or equal to 45^o was desired. Thus, whenever the slope was greater than 45^o a new ground station was added by having the broker send a message to a ground station instructing it to spawn another. We used a similar method to remove unnecessary ground stations, when the slope decreased beyond 15^o we removed a ground station by having the broker send a "kill" message to a ground station instructing it to remove itself from the system. This ensured that the system was functioning at an optimum efficiency level. This behavior is shown in figure 3 where we added 2 ground stations due to the sudden "appearance" of several satellites and removed 2 ground stations as the satellites moved beyond the horizon.

4. Conclusion and Future work

Using a real-time simulation of a distributed agent architecture that is highly dynamic in nature we have developed a simple metric to determine when additional agents will be required to manage the expected message traffic and when we can remove the additional agents to reduce the system overhead.

Several issues have yet to be fully answered. These include the length of the time window for averaging the queue growth and the slope values used for triggering action. How dependent these parameters are on the specifics of the environment have yet to be answered and will be the focus of our future work.

This work was supported by a grant (F49620-00-1-0302) from the Air Force Office of Scientific Research.

References:

Decker, K., Williamson, M., & Sycara, K. (1996). Matchmaking and Brokering. *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96).*

Harrison, C. G., Chessm, D. M., & Kershenbaum, A. (1995). *Mobile Agents: are they a good idea?*. Research Report, IBM Research Division.

Iba, H., Nozoe, T., & Ueda, K. (1997). Evolving Communication Agents based on

Genetic Programming. *Proceedings of 1997 IEEE International Conference on Evolutionary Computation*, pp 297-302.

Iba, H. (1998). Evolutionary learning of communication agents. *Journal of Information Sciences, vol.10,* pp181-205.

Jennings, N. R., Sycara, K., & Wooldbridge, M. (1998). A roadmap of agent research and development. *Autonomous Agents and Multi Agent Systems, 1,* 275-306. Kluwer Academic Publishers, Boston.

White, J. E. (1997). Mobile Agents. *Software Agents,* pp 437-472. AAAI press and The MIT Press.

Yager, R. (1997). Protocol for Negotiations among Multiple Intelligent Agents. *Consensus under Fuzziness*, pp 165-174. Kluwer Academic Press.