The Adaptive Network Architecture for Formations of Heterogeneous Spacecraft

Dipa Suri, Lockheed Martin Advanced Technology Center
Adam Howell, Lockheed Martin Advanced Technology Center
3251 Hanover Street
Palo Alto, Ca. 94304

Abstract **Many future earth and space science missions, such as the Magnetospheric Multiscale Mission and Terrestrial Planet Finder, have been proposed that require a high level of coordination between multiple heterogeneous sensors and spacecraft to achieve their mission objectives. On-board autonomy and distributed processing are additional key technologies that can improve the fidelity and quality of the missions' data products while reducing the associated ground support. This paper will present the development of an agent-based software architecture for real-time embedded systems that provides a flexible, reconfigurable framework to support these technologies.**

A. INTRODUCTION

Satellite formations are a key element of the Earth-Sun System enterprise's strategic plan for future space and earth science missions. Among the important themes proposed by the Advanced Information Systems Technology (AIST) program is the deployment of missions comprised of multiple autonomous spacecraft that use a distributed, agent-based architecture. Furthermore, it is prudent for implementations of such architectures to leverage mature terrestrial standards, e.g. TCP/IP, to reduce development and validation costs. The potential benefits of these resulting systems are multifold; autonomy leads not only to flexibility and system robustness, but more importantly it can reduce the life-cycle cost for both the space and ground elements.

To help realize this vision, we are developing the Adaptive Network Architecture (ANA) composed of a set of heterogeneous software agents that interact and collaborate through message-based communication. The ANA is fundamentally responsible for ensuring that mission objectives are met by autonomously responding in real-time to both environmental events and ground user requests for (i) the optimal allocation of computing and communication resources; (ii) instrument reconfiguration as part of either current mission needs or fault management; and (iii) distributed science processing and data aggregation. The goal is to provide an integrated solution for all spacecraft formation operations that would allow ground users to operate a distributed aperture sensor or cluster as if it were a single instrument.

Section B will present an overview of the software architecture, common agent functionality, and communication schemes composing the ANA. A detailed description of the various agents comprised in the ANA will be provided in Section C, followed by an example mission implementation in Section D. Finally, a discussion of future work and some concluding remarks will be presented in Sections E and F, respectively.

B. ARCHITECTURE OVERVIEW

The ANA is based upon the concept of *software agents*. Many diverse definitions for software agents can be found in the literature [1], however we will define an agent to be a software entity that has the characteristics of *communication*, *autonomy*, and *collaboration*. The ability to communicate is relatively self-explanatory, while the characteristics of autonomy and collaboration require clarification. By autonomy, we mean that the agent is able to make independent decisions in support of its internal goals without direct operator intervention (although this does not necessarily apply to *all* decisions). Collaboration means that each agent must be able to interact and negotiate with other agents in order to achieve its goals.

The ANA distributes the responsibility for different aspects of a science mission between several component agent types. A schematic of the ANA agents and supporting services is shown in Fig. 1. Each agent type shown has its own tasks to perform, while more complex processes are achieved through interactions and collaborations of multiple agents. The ANA provides additional flexibility by allowing different configurations of agents to be instantiated at system initialization or runtime, depending on the desired functionality. By default, the Executive and Computing Agents are singletons required on all hosts. However, subsets of the other agents can be created depending on the specific host's hardware, mission objectives, and available computing resources. A typical system running the ANA would consist of (i) multiple hosts and (ii) multiple instruments/payloads either on a single or multiple spacecraft.

C. AGENT ANATOMY

The characteristics described above cover the basic functionality that we require from each agent, however the anatomy of an agent can be broken down into three main components: inter-agent communication, basic agent functions, and the role/responsibilities of the specific agent types.

C.1 Inter-Agent Communication

As mentioned above, a message-based scheme is used for inter-agent communication. The Common Object Request Broker Architecture (CORBA) [2] standard is an inherent part of the ANA communication structure to enable transparent cross-platform interoperability (network type,

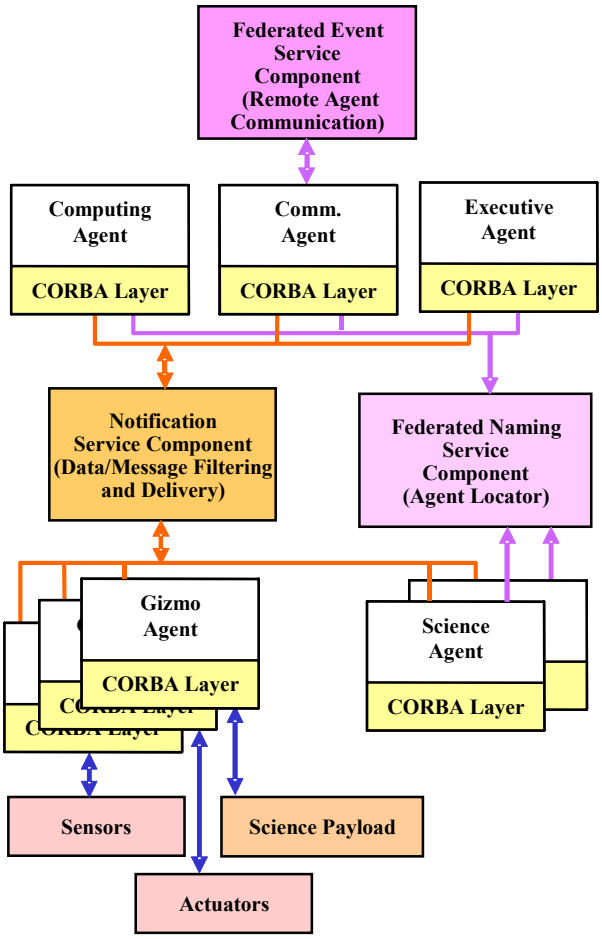

Fig. 1: Schematic of the Adaptive Network Architecture

programming language, operating system) and leverage existing terrestrial standards. The ANA is built using the open-source ADAPTIVE Communications Environment (ACE) and The ACE ORB (TAO) [3]. ACE/TAO is an ideal choice for several salient reasons:

- Rich use of software design patterns makes it both robust and flexible
- Real-time extensions for many standard CORBA Services
- A world-wide user community that continually infuses it with improvements

It should be noted that while we have chosen C++ as our implementation language, ACE/TAO directly supports programming in C/C++ and Java (ZEN).

The ANA message format is based on the standard developed by the Foundation for Intelligent Physical Agents (FIPA) for Agent Communication Languages (ACL) [4]. The description of an ANA Message in CORBA Interface Definition Language (IDL) is shown in Table 1. A subset of the message performatives and message fields were chosen for simplicity, however it can easily be extended to support the full FIPA specification. The first three fields are the names of the sender, receiver, and the receiver of any replies. Next is the timestamp of message transmission, along with a globally unique conversation identifier that allows agents to relate a message to a specific conversation. The header of the message is a performative that describes the type of message, and the last field is a sequence of the message payload data. The particular meaning of a given message is specific to each agent type's inner language, and naturally two agents must both comprehend a common language to converse.

Two CORBA services - the Notification Service and an extended Event Service - are currently used by the ANA to provide messaging support. The use of these services depends on whether the conversing agents are collocated i.e. on a single host or distributed i.e. resident on multiple hosts on a spacecraft or multiple spacecrafts. Local inter-agent communication relies on the Notification Service to deliver ANA Messages via structured events [5]. The Notification service was chosen because of its flexible event structure and advanced filtering, however it does not currently provide an inherent means of passing events between agents located on different processors. The ANA uses the TAO Real Time Event (RTEC) Service to handle such remote communications. More specifically, a federated RTEC is implemented to transparently connect the ANA agents that are not collocated. Although the RTEC can be distributed through federation, the simplicity of the real-time events constrains the event filtering possibilities. A schematic representation of the communication architecture is also shown in Fig. 1.

TAO's primary developer, the Distributed Object Computing (DOC) group – now part of the Institute of Software Integrated Systems (ISIS) at Vanderbilt University - is our team member for this research, and is currently enhancing the TAO Notification Service to be federated.

Table 1: ANA Message IDL

```
// Subset of FIPA Message Performatives
enum Performative
{
        REQUEST,
        INFORM,
        AGREE,
        REFUSE,
        NOTUNDERSTOOD
};

// Modified FIPA ACL message structure
struct Message
{
        string sender;
        string receiver;
        string reply_to;
        long time_stamp;
        string conversation_id;

        Performative header;
        sequence<any> content;
};
```

With this added capability, the RTEC will soon be phased out, resulting in a simpler, more robust architecture.

C.2 Base Agent Functionality

Each agent supports a basic level of functionality common across all agent types, such as messaging, health reporting, and telemetry handling. These fundamental capabilities are provided for each specific agent through inheritance from a parent 'BaseAgent' class.

To give insight into the common functionality support, the BaseAgent interface definition in CORBA IDL is shown in Table 2. First, all agents are identified by an agent type and a unique name, which describes the specific agent, host processor, and physical location where the agent is running. During instantiation, each agent binds itself with the Naming Service that currently runs on the ground station. This allows agents to locate each other based on the naming scheme we have designed.

Table 2: ANA BaseAgent IDL

```
// Agent types
enum AgentClasses
{
        EXEC, // Executive
        COMM, // Communication
        COMP, // Computing
        GIZMO, // Generic Gizmo
        SCIENCE,
        USERINTERFACE
};

// Agent states
enum StateType
{
        AWAKE,
        DOZING
};

// Heartbeat information
struct HeartBeat
{
        string sender;
        StateType CurrentState;
};

// Standardized agent interface
interface BaseAgent
{
        readonly attribute AgentClasses AgentType;
        attribute string AgentName;

        readonly attribute StateType ActivityState;
        //Request agent becomes dormant
        boolean Doze();
        //Request agent becomes active
        void Wakeup();

        attribute float HeartBeatRate;

        //Process incoming message
        void ProcessMessage();
};
```

All agents can be in one of two states, dozing or awake. While in the dozing state, the agent's acceptance of messages and most internal processes are suspended. Generally, all agents are instantiated in the dozing state and must be explicitly commanded to awake. This capability of 'job control' for an agent is provided by the Doze() and Wakeup() interface calls.

All agents must report their health via heartbeats to a designated agent at the specified HeartBeatRate. A heartbeat message has an INFORM performative with the 'HeartBeat' structure as content. Once an agent is instantiated and sent an initial Wakeup() call, the heartbeat messages are sent via the Notification Service independent of the agent's subsequent operational state.

C.3 Agent Roles and Responsibilities

The roles and responsibilities for different aspects of a distributed science mission are divided between several types of component agents. The agent types currently implemented are further subdivided into a subset that run on a spacecraft (the "space" set) consisting of the Executive, Communication, Science, Computing, and Gizmo Agents, and another subset that resides on a ground station (the "ground" set) consisting of a User Interface Agent. Each agent type will be briefly described below.

C.3.1 Executive Agent

An Executive agent manages all agents on a single host, including creation, state change, and health monitoring. Continuing enhancements to the Executive Agent will include the management of faults and system security policies.

C.3.2 Communication Agent

The Communication Agent manages various aspects of communication on a given host. This will entail management of the communication hardware, communication services, as well as the types and frequency of transmitted data. To reduce system complexity, we have designed the Communication Agent to be solely responsible for routing telemetry to the ground. All agents in the space set format their own unique telemetry as part of their heartbeat processing, then transmit it to the Communication Agent as an INFORM message. The individual messages are re-assembled by the Communication Agent into a single telemetry stream before being sent to the User Interface agent.

The telemetry system is designed to be extremely flexible. As is typical of spacecraft, each telemetry item is codified with a unique identifier – its measurand ID. Based on the operational environment, at any given time, each agent is free to format and transmit selected items with no constraint on either the number of items or the length of each item. Thus the contents of the telemetry stream are dynamically variable.

C.3.3 *Science Agent*

Each Science Agent contains the logic associated with achieving a specific science objective, and the numerical processing generating the desired data products. The data processing is constructed as a set of concurrent streams that apply a sequential series of algorithms to the input data. The output data products can then be redirected to other Science Agents or User Interface Agents as specified in the mission logic.

C.3.4 *Computing Agent*

One agent type that directly supports the Science Agents' tasks is the Computing Agent. The Computing Agent manages the computing resources, in terms of CPU utilization, memory usage, and network throughput, for a for the science processing conducted on a single processor. This management is used to ensure efficient resource utilization and real-time delivery of data products. Currently, the Computing Agent performs only resource monitoring, however a purely distributed negotiation protocol for requesting remote computing resources has been implemented to handle the resource allocation.

C.3.5 *Gizmo Agent*

The Gizmo Agent, was developed to manage interaction and conflict resolution for accessing "negotiable" physical devices, such as payload sensors. The Gizmo Agent is an abstract parent class that presents a publish/subscribe protocol for other agents to subscribe to data updates based on data type, sampling rate, subscription duration, and priority. Conflicting subscriptions are resolved by a priority based scheduler. Direct interaction with each type of physical device will be handled by a specific agent class that inherits from the 'Gizmo Agent' base class, and contains the device-specific interface code. A concrete realization of a Gizmo agent has been implemented for a 'Smart' Camera commercially developed by Carnegie Mellon University (CMU) [6].

C.3.6 *User Interface Agent*

Finally, the User Interface Agent provides a unified means for both autonomous interaction and manual ground user interaction with the space set ANA agents. At present, it provides visibility into those agents that have registered with the Naming Service and for the processing, display, and recording of telemetry transmitted from the space segment.

D. EXAMPLE SCIENCE MISSION: GAMMA RAY BURST DETECTION

Although each agent has a particular role to play in the ANA, they seldom work in isolation. Interaction between multiple agents is necessary to complete complex, higher level tasks. This section will illustrate some of these interactions within the context of an example science mission involving the detection of gamma ray bursts. For brevity, this description focuses on the data acquisition process involving the Science and Gizmo Agents; however the full mission includes distributed onboard processing of images (e.g. compression) and the telemetry downlink.

The goal of the example science mission is to detect and capture images of a transient gamma ray burst event. The physical system consists of a single smart camera as a payload sensor, a single processing host, and a ground station. A schematic of the experimental system is shown in Fig. 2.

The smart camera can operate in two mutually exclusive modes: a color tracking mode and an image capture mode. In the color tracking mode, the smart camera's onboard processor finds the centroid of all pixels within an *apriori* specified RGB color range at a rate of 17Hz. Single frames can be captured by the camera at a slower rate of 1Hz in the image capture mode.

Given our hardware configuration, the mission objectives are best served by switching between the two camera modes to ensure

- Rapid detection of a burst (via the color tracking mode)
- Increased data collection during a burst (via the image capture mode)

The logic for this mission can be captured in the interaction diagram shown in Fig. 3.

After initialization, the Science Agent subscribes to the Smart Camera Agent (a child of the Gizmo Agent class) to receive color tracking data at the maximum sampling rate. The Smart Camera Agent receives the SUBSCRIBE message, schedules the subscription request, and sets the current camera mode to the appropriate mode. As the smart camera takes samples, they are encapsulated in INFORM messages that are sent to the subscribing Science Agent. Once a burst is detected (i.e. a nontrivial centroid is sent to the Science Agent), the Science Agent changes its subscription by requesting a single image frame by sending a new SUBSCRIBE message with high priority. The Gizmo Agent then reschedules the highest priority request (i.e. the image capture subscription), and sets the new mode of the camera. Once the image is captured, it is also sent to the Science Agent via an INFORM message for further processing and

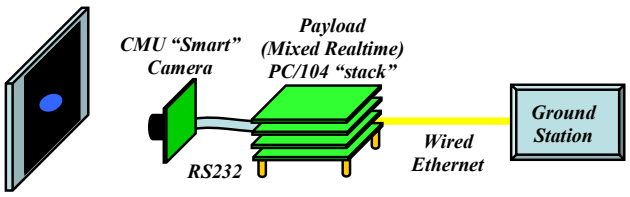

Fig. 2: Schematic of the experimental setup for the example science mission

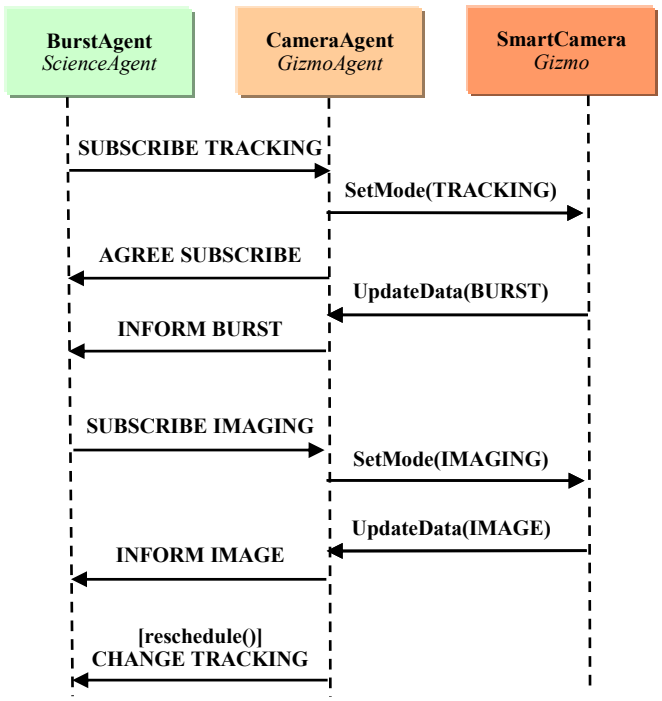

Fig. 3: Science Mission Interaction Diagram

downlink to the ground station. Subsequently, the Gizmo Agent reschedules the subscriptions again, returning to the color tracking mode. This cycle repeats throughout the duration of the mission.

E. FUTURE WORK

The near-term target of the ANA is deployment and verification on a hardware testbed containing a set of heterogeneous computing platforms with heterogeneous connectivity termed the Formation Computing Environment (FCE). The FCE is an integral part of our Distributed Systems Laboratory (DSL). Other assets that will come into play for system demonstration are two classes of air-bearing robots termed picoBots and microBots to simulate small spacecraft and their formations, shown in Fig. 4. At present, we are developing and verifying the ANA on two different

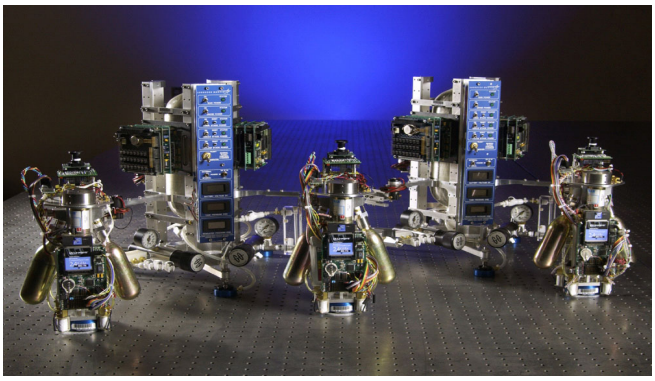

Fig. 4: Robotic assets of the Distributed Systems Laboratory

embedded PC/104 processing hosts (i.e. stacks) – an X-86 Pentium III and PowerPC (MIP405) PC/104, as well as a simulated ground station – a X-86 Pentium IV workstation. In order to satisfy hard real-time constraints of functions such as spacecraft GNC, we run the ANA under the WindRiver Tornado/VxWorks operating system on the Pentium III processor. For science processing, the PowerPC processor runs a small footprint GNU/Linux OS with RTAI real-time extensions for both "soft" real-time and non real-time processing. The ground station runs Windows. Fig. 5 is a schematic showing the processor types, their roles, and the end-to-end connectivity. Both the Pentium III and PowerPC platforms can be mounted on a microBot to represent the spacecraft and payload subsystems respectively. The workstation represents ground mission control. Multiple picoBots and microBots will be configured to represent formations of heterogeneous spacecraft.

Since the inception of Phase I of this program, we have made significant progress in the design and the implementation of the infrastructure, particularly with regard to the inter-agent communication. Extensive work yet remains to provide the agents with the full capacity to exhibit intelligent, adaptive behavior in order to meet mission objectives. While such development will continue to evolve incrementally over the next several years, much of the 1 year Phase II will be devoted to incorporating additional functionality for fault detection and recovery, improved resource utilization, and mission planning. The space set agents will, under the direction of the Executive Agent, detect and respond to faults via autonomous macro (at the spacecraft formation level) and micro (on a subsystem or lower level) reconfiguration. Further additions of a Mission Operations Agent and a Science Operations Agent will greatly enhance the computational and autonomous decision making capacity of the system as a whole. Initial verification of the latter will take place by infusing these agents into the ground set. With continued development, transfer of this capability to the space set will lead to a progressively more autonomous system.

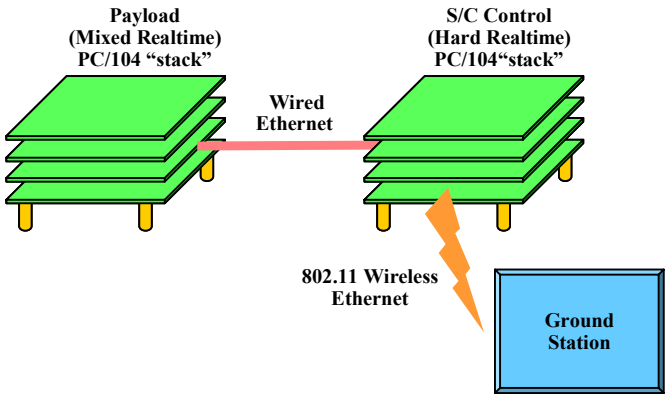

Fig. 5: Schematic of the Formation Computing Environment including processor type, role, and connectivity

For increased flexibility, we will transition from using the CORBA Naming Service to the CORBA Trading Service, (akin to a "yellow pages") as the Agent locator. This will free us from an unnecessarily constrained scheme of finding agents via name only. All agents will publish known capabilities in addition to names as a means of determining location. In addition, the Trading Service provides the capability to automatically notify subscribed agents of changes to its data base resulting from either departing or newly joined agents.

To help complete the full ANA suite, a Navigation Agent will be added to the space set. This will handle the Guidance Navigation and Control functions of a spacecraft as well as the higher-level constellation formation and maintenance.

F. CONCLUSIONS

Agent technology is now an established paradigm for developing large complex software systems across many varied domains, where autonomy and intelligence are common central important themes. The ANA follows in the tradition of multi-agent systems such as OASIS [7] deployed and tested at the Sydney Airport, and the Remote Agent Experiment flown on the NASA Deep Space 1 mission [8]. In extending the capabilities of these predecessor systems, it will be well poised to support and handle not only complex space based science missions of heterogeneous makeup, but also other large distributed systems that comprise missions such as NASA's Space Exploration Initiative.

We have successfully built the ANA over mature terrestrial standards that will provide the same seamless operational functionality to future missions with components distributed both in space and ground. Improvements to these standards as a result of this technology development are already beginning to emerge e.g. performance optimizations and a significantly reduced footprint make TAO even more feasible for use in resource constrained real-time embedded systems. This work has provided similar impetus for extending the capability of the CORBA Notification Service to operate as a federation with multiple options to configure the federation based on available resources and mission specific needs.

We look forward to verifying the versatility and usefulness of this technology on future earth, space, and planetary missions. We are confident that it will come to represent an important component of systems constructed to meet the objectives of these missions.

ACKNOWLEDGMENT

We are greatly indebted to the Advanced Information Systems Technology (AIST) program of the NASA Earth-Sun System Technology Office (ESTO) whose sponsorship has been instrumental in the development of this technology.

REFERENCES

[1] N. Jennings, M. Wooldridge, Agent Technology: Foundations, Applications, and Markets, Springer-Verlag, 1997

[2] http://www.omg.org/gettingstarted/corbafaq.htm

[3] www.cs.wustl.edu/~schmidt/TAO-users.html

[4] http://www.fipa.org/specs/fipa00061/

[5] http://www.omg.org/technology/documents/formal/notification_service.htm

[6] http://www-2.cs.cmu.edu/~cmucam/home.html

[7] M. Georgeff, A. Rao, "Rational Software Agents: From Theory to Practice", Agent Technology: Foundations, Applications, and Markets, pp. 139-160, Springer-Verlag, 1997

[8] N. Muscettola, et. al., "Validating the DS1 Remote Agent Experiment", Proceedings of the 5th International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS-99).