Proceeding of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space: i-SAIRAS 2003, NARA, Japan, May 19-23, 2003

Command and Control of a Cluster of Satellites

Paul Zetocha
Air Force Research Laboratory
AFRL/VSSW
Kirtland AFB NM, 87117
Paul.Zetocha@Kirtland.af.mil

Abstract

There is an increasing desire in many organizations, including NASA and the DoD, to use constellations or fleets of autonomous spacecraft working together to accomplish complex mission objectives. At the Air Force Research Laboratory's (AFRL) Space Vehicles Directorate we are investigating and developing architectures for commanding and controlling a cluster of cooperating satellites. For many space missions, large monolithic satellites are required to meet mission requirements. In many cases this results in costly satellites which are more complex, more susceptible to failure, and which have performance characteristics that are less than optimal due to realistic physical size limitations. Recently various organizations have begun to explore how distributed clusters of cooperating satellites can replace their larger monolithic counterparts resulting in an overall cost reduction, enhanced mission performance, and increased system fault tolerance. Large clusters of satellites flying in formation are required to have some level of on-board autonomy in order to: fly within specified tolerance levels; perform collision avoidance; address fault detection, isolation, and resolution (FDIR); share knowledge; and plan and schedule activities. In addition, from an operations standpoint, commanding and controlling a large cluster of satellites can be very burdensome for ground operators. At AFRL we are addressing these issues by development of an on-board *Cluster Management* system which will, in essence, provide the capability to treat a cluster of satellites as a single virtual satellite. A systems level approach is being taken, therefore from a ground perspective the ground control station must also be able to treat the cluster as a virtual satellite. [1]

This paper will describe our Cluster Management system, which is the intelligent entity that is responsible for making cluster level decisions. It is this piece of software that enables a satellite cluster to function as a *virtual* satellite. The cluster manager functionality can be broken down into the following five areas:

- Command and control
- Cluster data management
- Formation flying
- Fault management
- On-board Planning

This paper will contain a detailed description of the Cluster Manager architecture along with its various modules.

1. Introduction

For many missions the payload performance of an individual satellite has a limitation which is a function of the physical size of the satellite. As an example, for a surveillance type mission, resolution capability is a function of antenna size with a larger antenna providing better resolution. For obvious reasons it is not feasible to have an antenna which is on the order of hundreds of meters across. A method to overcome this limitation is to have a cluster of cooperating satellites which would in effect result in a single *virtual* satellite with a large virtual aperture that is much larger than could feasibly be hosted on a single satellite. The end result of this would be a surveillance system with much higher resolution capability. Many challenges exist in implementing such a system including formation flying, distributed payload processing, fault management, and on-board cluster control. In order to maintain optimum mission performance the cluster of satellites must be able to fly in formation relative to one another. This includes tight position control with even tighter position knowledge. Tight position knowledge may be needed because timing may be very critical in order to process payload data accurately. In addition, some form of executive cluster control is also necessary in order to handle issues such as redundancy; load balancing; fault detection, isolation, and resolution (FDIR); and resource optimization. [2]

The high level architecture for the Cluster Manager prototype under development is depicted in figure 1.

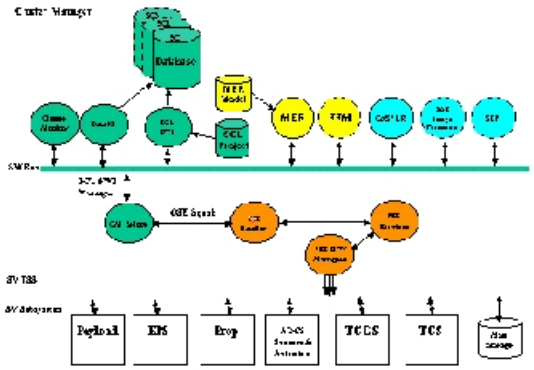

Figure 1. Cluster Manager Architecture

The command and control portion of the cluster manager is implemented using a combination of the Spacecraft Command language (SCL) from Interface and Control Systems, CASPER which was developed by NASA/JPL, and TeamAgent which was develope d by Princeton Satellite Systems. Intersatellite communication and communication between the cluster manager and the ground is implemented using SCL. CASPER is used to break high level commands down into lower level commands and to help plan implementation of complex tasks. SCL, CASPER and TeamAgent are all used to generate commands for other spacecraft with the specific commands encoded as CCSDS packets.

A sophisticated cluster data management system is needed because the cluster must be able to exchange state of health information for all the satellites in the cluster. The cluster manager must be able to provide telemetry data to the ground for all of the satellites in the cluster. The SCL database is used to keep track of all necessary data. Data is exchanged amongst satellites on a periodic basis with varying periods which are dependent on both importance and change frequency.

The formation flying part of the cluster manager is responsible for maintaining the cluster formation and for reconfigur ing the cluster whenever necessary. The algorithms necessary for formation flying are implemented as agents within the TeamAgent architecture. The inputs needed by the agents are provided by both other agents and by telemetry points which are captured in the SCL database. The outputs from the formation flying segment of the cluster manager can be either tasking to other vehicles in the cluster or to other modules on the resident satellite.

The cluster manager will be responsible for identifying and handling cluster level faults. Cluster level faults are those faults which require action from the cluster in order to be managed. An example of a cluster level fault is a failure of the intersatellite link in one of the satellites. In this case, though the spacecraft in question may still be able to function as an individual satellite, it is no longer able to participate as a member of the cluster and the cluster manager must compensate for this fact. Cluster level fault management will be implemented using a combination of TeamAgent, the SCL expert system, and a model-based reasoning system.

CASPER is an autonomous on-board planning tool that takes as input high level goals which are to be achieved and then develops sequences of spacecraft activities that will meet those goals. To perform reasoning CASPER contains knowledge of spacecraft state, subsystem models, and operational constraints.

2. Command and Control

For a large cluster it is not efficient or cost effective to command satellites on an individual basis. A more efficient method is to send commands to the on-board cluster manager and then have the cluster manager either parse the command string and forward the command(s) to the appropriate satellite(s) or to make some intelligent decision as to the most appropriate action. In our architecture two types of ground commanding are possible. The first type involves sending up a sequence of commands which are intended for specific satellites. With this scenario, one possible implementation method is to use a delimiter to separate commands with each command possessing fields containing the satellite number, command name, value, duration, and time. This description is simplified to illustrate the point and neglects fields such as a CRC field for error checking. This command string would be sent to the on-board cluster manager where the cluster manager would then parse the command string and send commands to the appropriate satellites. [3]

A second type of commanding involves commands which are sent to the cluster without specific indication as to which satellites in the cluster will ultimately be effected. A hypothetical example might be to issue a command to "observe region x at time y". The cluster manager, based on the status of the satellites at time y, will then determine the appropriate course of action to be taken. This type of commanding requires more on-board intelligence than in the first scenario and has a higher level of risk. Because of the higher level of risk, safeguards need to be put in place to ensure no adverse conditions arise.

To command the cluster of satellites a common frequency is assumed with a spacecraft ID used to denote what commands are destined for which satellites. The satellites are assumed to be flying in close enough formation so that they are all within the same beamwidth. All satellites receive the command but not all will process that command. The operation is somewhat analogous to a TCP/IP broadcast system.

Telemetry decommutation on the ground requires being able to parse telemetry from multiple satellites. For a TDM-based telemetry system this scales nicely from how traditional Time Division Multiplexing (TDM)-based systems operate. Telemetry from different satellites simply gets associated with specified frames and frame locations. For CCSDS-based systems packets can contain a field which identifies where they originated.

As the core of our ground system we are baselining the *Spacecraft Command Language* (SCL) from Interface and Control Systems. SCL is a Commercial-off-the-Shelf (COTS) software package which contains an expert system and a command scripting language. It was designed to operate both on-board a satellite and on the ground. This makes it an ideal environment for developing a prototype which contains the cluster commanding and monitoring capability described earlier. Expert system rules can be developed and migrated from ground to space as appropriate. Using the rule-based expert system a fault tree for known anomalous conditions can be developed.

3. Cluster Data Management

Cluster data management is needed because the cluster must be able to provide state of health information for all the satellites in the cluster. In addition, it must keep track of data, such as relative position a nd velocity, needed to control the cluster. The Cluster Manager must be able to provide any telemetry data requested by the ground for any of the satellites in the cluster. The SCL database is used to keep track of all necessary data. Data is provided b y the various satellites in the cluster to the cluster manager on a periodic basis with the update frequency dependent on both telemetry point importance and change frequency The Cluster Manager maintains a database for each satellite in the cluster and e ach database will be periodically updated with the values of all the spacecraft mnemonics. In addition to the spacecraft telemetry points each database will also have a number of derived cluster level telemetry points. Of particular importance to our pro totype are the following telemetry points:

- Relative position
- Relative velocity
- Absolute position
- Absolute velocity
- Attitude quaternion
- System time
- Spacecraft mode
- Fuel level
- Reference trajectory
- Sensor states

Accurate knowledge of the particular mode that a satellite is in is critical in order to make correct decisions. As an example, it is counter-productive to perform a maneuver, or any other vehicle maintenance procedure, while a particular satellite is performing a surveillance task. In addition, to properly assess the health and status of a satellite in an autonomous fashion, it is imperative that the mode of the satellite be known. The health and status of a vehicle is virtually always determined by the values of its telemetry mnemonics. A large number of the telemetry parameters have values which are a function of the space vehicle mode. Attempting to assess the state of health of a vehicle without knowing what mode it is in is fruitless. The Cluster Manager will have limited capability to

schedule resources on-board to ensure both mission success and optimization of resources.

The Cluster Manager will also have knowledge of the status of sensors across the cluster. This would include both payload sensors and attitude sensors. To ensure that sensors are operating properly, their associated parameters can be monitored with the values limit checked. If payload sensors are malfunctioning, the outputs coming from those particular sensors can be ignored or compensated for. Another possible solution is to switch to a backup unit. In tasking individual satellites to perform a mission the sensor states are accounted for.

4. Formation Flying

The formation flying portion of the cluster manager is responsible for maintaining the cluster formation, r econfiguring the cluster whenever necessary, and performing collision avoidance. The algorithms necessary for formation flying are implemented as ObjectAgent agents. The inputs needed by the agents are sometimes provided by other agents and sometimes obta ined from the SCL database. The outputs from the formation flying segment of the cluster manager are satellite taskings either to the satellite generating the command(s) or to other satellites within the cluster.

ObjectAgent is an agent-based messaging system and is depicted in figure 2. [4]

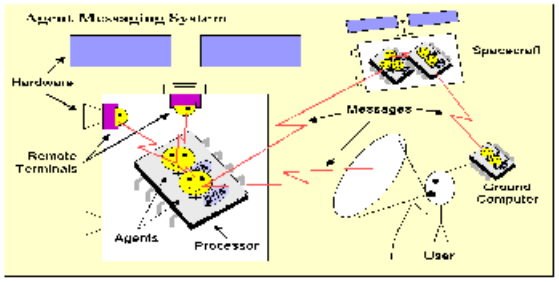

Figure 2. ObjectAgent System

The ObjectAgent system is an agent-based real-time architecture for distributed, autonomous control. [5] Control systems are decomposed into agents, each of which is a multi-threaded process. Agents are used to implement all of the software functionality and communicate via a flexible messaging architecture. Each message has a content field written in natural language that is used to identify the purpose of the message and its contents. Agents may be loaded at any time and have the capability to configure themselves when launched, which simplifies the process of updating flight software and removes the complexity associated with software patches. They will automatically seek out other agents who can provide them with the inputs they need. Decision-making and fault detection and recovery capabilities are also built-in at all levels.

Agents in ObjectAgent can be used at all levels of software functionality because there is no set level of complexity for an agent. For instance, a designer could choose to implement the entire flight software as a single agent, he could use one agent for the software related to each subsystem, or he could use multiple agents for each subsystem etc. ObjectAgent agents are composed of skills. The skills that an agent possesses determine its complexity and functionality. However, all agents have some basics skills to ensure that they can communicate. In addition, agents have self-knowledge and they can explain their functioning and purpose to other agents and users. Agent communication takes place solely through messages, there is no shared memory between agents. This ensures that agents can work together even when they are not located on the same processor.

The TeamAgent system applies ObjectAgent to the problem of controlling multiple cooperative satellites. [6] Thus, both systems are well suited for use in the command and control of satellite clusters. TeamAgent enables agent-based multi-satellite systems to fulfill complex mission objectives by autonomously making high- and low-level decisions based on the information available to any and/or all agents in the satellite system. The required spacecraft functions for the multiple spacecraft missions have been identified and the use of software agents and multi-agent based organizations to satisfy these functions have been demonstrated. Simulations of multi-agent systems for multiple satellites have been developed using TeamAgent to illustrate collision avoidance and reconfiguration for a four-satellite constellation. Agents were used to monitor for collisions, reconfigure the fleet, optimize fuel usage across the cluster during reconfiguration, and develop a fuel-optimal maneuver for reconfiguration.

5. Fault Management

The cluster manager will be responsible for identifying and handling cluster level faults. Cluster level faults are those faults which require action from the cluster in order to be managed. An example of a cluster level fault is a failure of the intersatellite link in one of the satellites. In this case, though the spacecraft in question may still be able to function as an individual satellite, it is no longer able to participate as a member of the cluster and the cluster manager must compensate for this fact.

Fault management within our architecture is largely handled at two levels. At one level we employ the SCL real-time expert system. At the other level we use a model-based reasoning system.

Rule-based expert systems by design are centralized systems where an inference engine makes decisions based on the contents of a particular knowledge base. This scenario works fine for a single satellite, however attempting to run an expert system, with a single inference engine, over a distributed knowledge base is not practical. By design an inference engine is made to operate on a single cohesive knowledge base.

This does not however preclude the use of expert systems with a distributed cluster of satellites. One approach to overcome this is to have a hierarchy of rule-based expert systems. Each satellite in the cluster can have its own expert system with a knowledge base composed of information about itself. A global knowledge base can be used for cluster level decision making and hence Cluster Management functionality. The rules within this knowledge base would remain fairly constant, and developed a-priori, and would in most cases require operator input for modification. The facts would be dynamic and would be obtained from the various satellites in the cluster. These facts would contain information regarding the status of the particular satellites. The facts in this knowledge base could easily be obtained from the expert systems on-board the individual satellites. The knowledge base on-board the Cluster Manager satellite is periodically updated via the intersatellite link with the update frequency dependent on mnemonic importance and change frequency

A model-based reasoner functions by building a model of some subset of the satellite. Given a set of inputs into the model a theoretical set of outputs will be generated. These theoretical outputs are compared with the observed outputs and any discrepancy would indicate anomalous behavior somewhere in the system. Isolation of the fault can be determined by a number of methods but most of these center around isolating each component and observing it's behavior on the overall system.

6. On-board Planning

At the core of our on-board replanning system is CASPER which was developed by NASA/JPL. [7] CASPER is the on-board instantiation of the more well known ground based planning system known as ASPEN (Automated Scheduling and Planning Environment). CASPER works by incorporating spacecraft models, constraints, and objectives into the planning system. CASPER then generates a sequence of activities which will achieve the desired objectives. Inputs into the planning system would include data such as the mission profile, projected AFSCN and X-band contacts, projected target locations, and current and projected spacecraft State-of-Health (SOH). All activity plans generated by ASPEN are sent to SCL where they in turn are encoded as CCSDS messages. In order to provide consistency between flight and ground planning systems we are using ASPEN will at several levels. At a coarse top level it will be used to generate a one year mission profile. At the next level it will be used to develop two week plans which will be given to the primary operations center. At the finest level it will be used to generate plans over a 24 hour period. This data will then be used by the mission operations center to schedule the desired spacecraft activities with the Air Force Satellite Control Network (AFSCN). Figure 4 provides an example of an ASPEN display. This figure essentially highlights resources along the vertical axis vs. use of those resources along the horizontal axis.

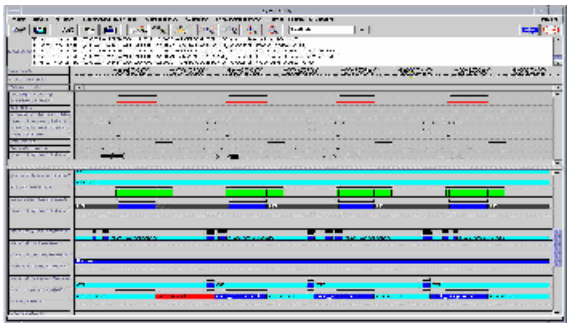

Figure 3: ASPEN

7. Conclusion

Work has been underway on our Cluster Management system for a little over two years and is progressi ng well. The core framework for the system has been developed and ported to the ENEA OSE real -time operating system running on a cluster of PowerPC 750 single board computers within the AFRL/VSSW Distributed Architecture Simulation Laboratory at Kirtland AFB. CASPER, SCL and our model -based reasoning system have also been ported to OSE and are in the process of being integrated. The formation flying system has been baselined and is operating within the TeamAgent architecture under the Linux operating sys tem. It is also currently being ported to OSE. Upon integration of the various components the Cluster Manager will then be tested and validated in a realistic environment. Following successful testing a flight demonstration is planned.

8. References

[1] Wyatt J., Sherwood R., Sue M., Szijjarto J., "Flight Validation of On-Demand Operations: The Deep Space One Beacon Monitor Operations Experiment", *Proceedings of the 5th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, ESTEC, The Netherlands, June 1999.

[2] "TechSat 21: Advanced Research and Technology Enabling Distributed Satellite Systems", *Overview Briefing of TechSat 21*, http://www.vs.afrl.af.mil/vsd/techsat21.

[3] Zetocha, P., "Satellite Cluster Command and Control", *Proceedings of the IEEE Aerospace 2000 Conference*, Big Sky MT, Mar 2000.

[4] Paluszek M., Thomas S., Sullivan W., Surka D., "ObjectAgent System Architecture for Autonomous Spacecraft", *Phase I SBIR Final Report*, May 1999.

[5] Zetocha P., Self L., Wainwright R., Burns R., Brito M., Surka D.; *Command and Control of a Cluster of Satellite, ;*Article in IEEE Intelligent Systems Journal, December 2000.

[6] Surka, D. M., M. C. Brito, and C. G. Harvey, "Development of the Real-Time ObjectAgent Flight Software Architecture for Distributed Satellite Systems," Presented at *IEEE Aerospace Conference*, Big Sky, Montana, March 20001.

[7] S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins, D. Tran , "ASPEN - Automating Space Mission Operations using Automated Planning and Scheduling," *SpaceOps 2000*, Toulouse, France, June 2000.

Acknowledgment

Much of the cluster management research described here is being accomplished by Princeton Satellite Systems and Interface and Control Systems under Small Business Innovative Research contracts and managed by the Space Vehicles Directorate of AFRL. The distributed architecture simulation laboratory is a resource of AFRL/VSSW and is located on Kirtland AFB in Albuquerque New Mexico.

Paul Zetocha is a Computer Engineer and is currently the lead for the Intelligent Satellite Systems Group of the Air Force Research Laboratory's Space Vehicles Directorate. For the past eight years he has been involved, both as Program Manager and through in-house development, with over a dozen programs related to spacecraft autonomy. He is also the chairman of the AIAA Intelligent Systems Technical Committee. Mr. Zetocha has received M.S. degrees in both Electrical Engineering and Computer Science from the University of New Mexico with an emphasis in the areas of signal processing and artificial intelligence respectively.