Space Networking- Interacting in Space Communication Networks

Sanda Mandutianu

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109-8099

Abstract— Space missions always required reliable transfer of data between space and ground. With the advent of multi-spacecraft missions, the requirements extend to inter-spacecraft telecommunications. The network of telecommunication nodes has to provide communication relay and navigation support for a variety of science missions. From this point of view any spacecraft will be considered as just another communication node. One of the communication characteristics is that the communication nodes may be dynamically making and breaking links between them. The existing communication architectures use the ground stations as intermediate or even terminal nodes. An important technological problem is to configure the nodes so that they could establish the links whenever they are needed and also to provide them with the capability to track one another so that they could set up the communication channel between themselves. It is not always feasible to define in advance all the possible routes, in which case the route construction has to be done in real time for each specific configuration. There is an ongoing exchange of information and negotiation among the spacecraft and the ground, and possibly between the spacecraft themselves. The space telecommunication constraints, such as intermittent connectivity, limited signal strength and noise make data loss more serious than in ground networks. On the other hand, communication at a lower level of abstraction increases the probability of frequent exchanges, including associated data transfers.

We address the issue of enabling computational technologies addressing both the requirements of communicating at a higher level of abstraction between heterogeneous ground and on-board systems and the requirements of space communication protocol standards. Higher level protocol layers are presented in the context of a distributed interactive multi-agent architecture simulating flight and ground software systems controlling and transmitting information in a space telecommunication network. The resulting agent protocol include higher level protocols for inter-agent communication at the knowledge or application level used to negotiate goals, as well as lower end protocols for space mission data transmissions. The asynchronous behaviors present in network management mean that complex interactions can occur. Multi-layered control architectures based on interacting agents seem appropriate, allowing for logical decomposition of the decision-making processes, reflecting the inherent distributed nature of this type of systems and support the need for intelligent control and robustness.

This paper presents the use of a multi-agent system, emphasizing the interaction aspects of computational components and allowing for sharing higher-level information structures between components. This architectural approach allows for encapsulation of the main mission software functions and transparently ensures their interoperability.

TABLE OF CONTENTS

1. Introduction

A critical challenge for space missions either orbiting the Earth or exploring other parts of the solar system is to transmit research and mission control information effectively and efficiently. In a complex system of satellites and ground tracking systems, there are some generic telecommunication problems that have to be addressed.

Existing and near-term constellations (e.g. GPS, Iridium, ICO) communicate bilaterally with their ground stations. Each satellite serves as a "bent pipe" between the sending and receiving ground stations. Managing constellations of bent-pipe satellites is relatively straightforward, because each satellite can be treated as an independent resource. In the next generation, the satellites will be interlinked in satellite networks. Some technological problems have to be solved, especially providing one satellite with the facility to acquire and track another reliably so that it can set up the telecommunication channel between them. Space-borne experiments are currently in progress (e.g. SILEX) to show that this can be done. Where the satellites are in a Geostationary Earth Orbit (GEO) then all possible routes can be constructed in advance. However, where the satellite network is at lower altitudes (i.e. in Low Earth Orbit - LEO) or Medium Earth Orbit (MEO), then the route construction has to be done in real time for each message. Additional problems add to this dynamic communication scenario, by exchanges (i.e. satellites or nodes) that are joining or leaving the network. The network has to remain functional while these changes happen.

The existing communication architectures use the ground stations as intermediate or even terminal nodes. An important technological problem is to configure the satellites so that they can establish the link whenever it is needed and also to provide them with the capability to track one another so that they could set up the communication channel between themselves. It is not always feasible to define in advance all the possible routes, in which case the route construction has to be done in real time for each specific configuration. There is an ongoing exchange of information and negotiation between the spacecraft and the ground, and possibly between the spacecraft themselves. Such systems have to manage the sequencing of the spacecraft data transmissions, the scheduling of the ground tracking systems, the analysis of the quality of a communication link, etc. They have to detect changes in the environment, and also estimate the availability of the transmission resources. A communication network in such a domain must be highly configurable and also be able to act autonomously in the absence of direct user control.

The objective of the effort described by this paper has been to create a flexible and robust software system for constellations of satellites. We address the problems of communicating between nodes, so that connections can be opportunistically established based on knowledge of the available resources in the network of satellites. Supporting successive topology reconfiguration requiring coordination of different aspects of network functionality from a more global perspective will be also addressed.

2. Multi-agent Communication System

There appears to be increasing interest in flattening control systems such that a community of cooperating agents can coordinate their activities to achieve overall decision making processes via negotiation. It seems appropriate for multi-layered approach to decision-making processes.

Telecommunication systems are inherently distributed. Their management system requires the ability to deal with run-time exceptions and remain robust under such circumstances. As more services are required, greater volume of information needs to be exchanged and processed and partial system failures require rapid reconfiguration. Satellite network management is becoming increasingly difficult to handle by global centralized mechanisms. Using distributed localized expert problem solvers can help automate the management process.

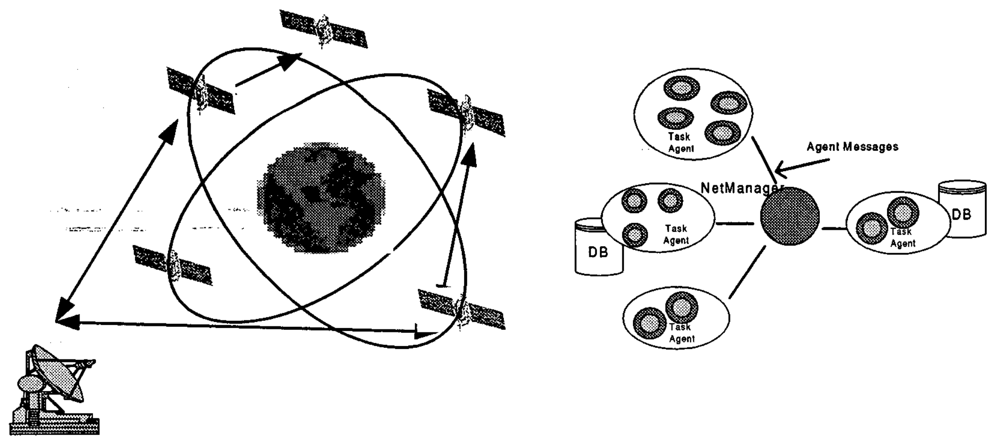

Fig. 1 Constellation of satellites as a multi-agent system

Routing in a communications network refers to the task of propagating a transmission flow from its source node to its destination node. The objective of a routing procedure is to ensure optimal utilization of the network. For a network node to be able to make an optimal routing decision, as dictated by the relevant performance criteria, it requires not only up-to-date and accurate knowledge of the state of the network, but also an accurate prediction of the network dynamics during propagation of the transmission flow through the network. However, this is impossible unless the routing algorithm operates in real time, in an opportunistic manner. An alternative to calculating the routes in real time is to select routes from a pre-defined set of possible routes. Anyway, in real time routing the adopted. The onboard automatic telecommunication link analysis determines the channel capacity and sets up the communication link. Establishing an optimal path involves besides evaluating an individual link, also taking into account the general instantaneous configuration of the network.

The asynchronous behaviors present in network management mean that complex interactions can occur. It is very difficult to encode all the possible scenarios, rather, robust techniques are required to encode the interacting factors and adapt to changes. Using multi-agent systems enables achieving multiple goals and managing multiple sensory input (Fig.1).

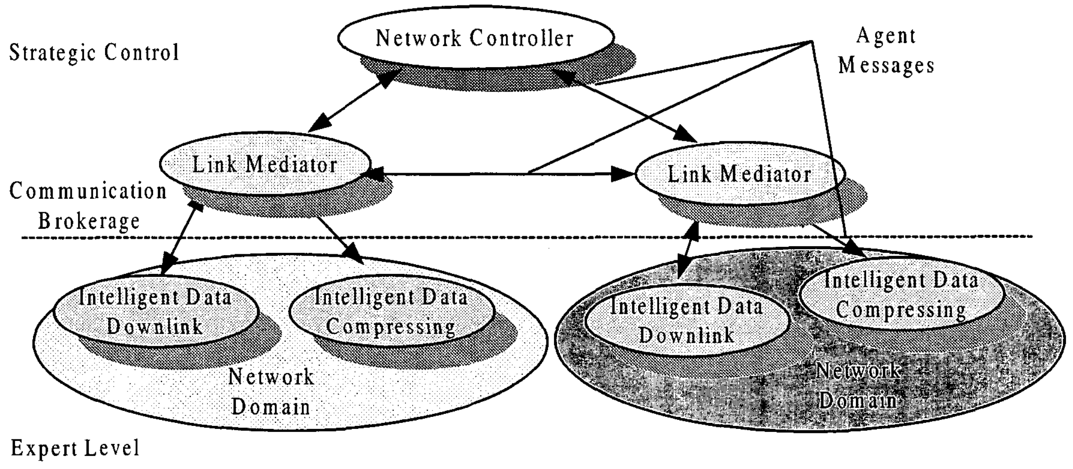

Fig.2 Decision making layers in a multi-agent system

transmission path can change step-wise. For example, the network nodes along the path might determine the route to be followed by consecutively specifying the next node to the route. The route of communication is dynamically altered after any change that could affect the route, for instance link failures, topology update or congestion.

If dynamic routing is driven by science opportunities, when such opportunities arise, for instance as a result of recognizing relevant features in the data being collected, or as a consequence of an adverse natural phenomenon, an optimal route has to be determined in real time. Traditionally this was done ground-based human operators. Sometimes there is no time for such interactions, or the use of the bandwidth became prohibitive. On-board, automatic solutions have to be

A multi-layered control architecture based on interacting agents seems appropriate to approach telecommunication management problems in a network of satellites (Fig.2). Each layer is defined to achieve the control of the telecommunication processes to a certain level of competence. This approach allows for logical decomposition of the decision-making processes, reflecting the inherent distributed aspects of this and supports the need for intelligent control, robustness and concurrency.

The interactions appear between heterogeneous agents, based on a shared collection of common knowledge (ontology) and using a generic agent communication language. The agent language defines the communication protocols as well. The agents can cooperate and achieve common tasks.

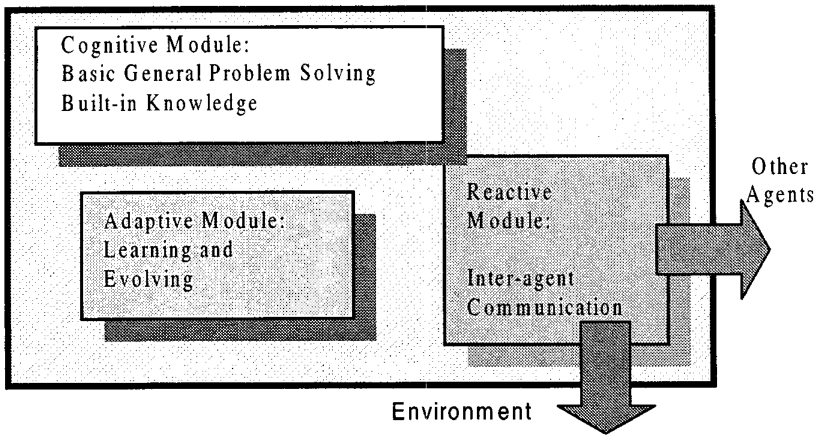

Figure 3. Agent Internal Structure

Any agent is comprised of a general problem solving mechanism and more specialized problem oriented capabilities (Fig.3). The general problem solving mechanism consists of a production system engine, which has the role of interpreting the rules describing the agent behavior. The rules represent the domain knowledge of the agent. Since no single set of rules is appropriate in all situations, the agent starts with a given set of rules, and evolves other, based on the interactions with the environment and with the other agents. This way, the population of problem solvers can adapt themselves to dynamically changing situations [11].

An agent is considered as an entity described in terms of common sense modalities such as beliefs, commitments, intentions, etc. This intuitive approach has its benefits for complex systems such as distributed systems, robots, etc., where simpler, mechanistic descriptions might not exist. The agent might also have a repository of recipes (plans or rules) which specify the course of actions that has to be followed by the agent to achieve its intentions. The beliefs are updated from observations of the environment and the effect of the interactions with other agents.

The agent behavior is expressed in a declarative agent definition language, which is used by an interpreter, which acts as a controller of the agent actions. The agent execution cycle consists of the following steps: processing new messages, determining which rules are applicable to the current situation, executing the actions specified by these rules, and possibly, planning new actions [10].

To achieve this flexibility and openness, the following new technologies have been used:

- **Agent Technology**. The use of agent technology provides a high degree of decentralization of capabilities, one of the keys to system scalability and extensibility. It also provides support for application encapsulation at the agent level, offering a higher level of software abstractions.
- **Domain Models** (Ontologies). Domain models give a declarative, uniform description of the meaning of the information, independently of the underlying syntactic representation or the conceptual models of information bases. Domain models increase the accessibility of information by allowing multiple ontologies belonging to different user groups.
- **Communication Brokerage**. Specialized link mediation agents, or brokers, support the flow of information. Brokered systems are also able to cope more quickly with a rapidly fluctuating population of agents, where agents can come and go and capabilities may change dynamically.
- **Distributed Decision-Making**. Besides following their own goals, the agents can act collectively by communicating and cooperating with each other. The agents use an agent communication language (ACL) as support of communicating goals; they can negotiate and coordinate. Problem solving performed by agents is a form of distributed problem solving and it involves coordination, negotiation and communication. In order for the agents to solve common problems coherently, they must share information about their activities, preferably via explicit communication.

3. Space Communications

Space communications face specific architectural challenges, due to extreme distances and the propagation delays that may have significant impact on real-time communications. Additionally, the spacecraft have limited power resources and they have to optimize the available bandwidth.

Space communication protocols have to provide reliable delivery of spacecraft command and telemetry messages between on-board computers and ground computers possibly involving unreliable space data transmission paths. Such protocols [15] have to cope with intermittent space/ground connectivity due to limited visibility from ground stations and contention among missions for contact time. These protocols have to provide interoperability across space missions and between data systems and the broader ground network environment.

Although the Internet protocol suite (e.g., TCP, UDP, IP, FTP) provides many functions needed for space communications, they were developed by assuming non-stop connectivity, limited data corruption, balanced bi-directional telecommunication links, etc. The SCPS supplement current protocols with the necessary layers for deep space communications.

Some of the requirements for enhanced space protocols has been identified as follows [15]:

- routing within satellite constellations;
- different routing treatments for different messages;
- routing algorithms must be able to accommodate dynamic topologies, select different algorithms based on network, connectivity, and circumstance;
- point-to-point, multicast, and broadcast;
- precedence (priority) and precedence-based congestion management;
- signaling of network conditions to upper-layer protocols.

4. Agent Interactions and Protocols

The inter-agent communication protocol builds on the Internet and space protocols layered structures. It defines a higher level for communicating at the application or domain level. It supports agent autonomous activities, peer-to-peer communication, higher level interoperability and benefit from the lower level signaling of the space protocols. Signals include indications of network congestion, network corruption and link outages, so that the agents can adapt their strategies.

In order to achieve *coordination* in multi-agent systems the agents might have to negotiate, they have to exchange information, i.e. they need to communicate. In multi-agent systems, the possible solutions for the communication range from no communication at all, ad hoc agent communication languages, and standard agent communication languages. If the agent communication language (ACL) is primitive, requests, commands and complex intentions cannot be expressed. If an ad hoc language is used, this makes the inter-operation non-trivial, and sometimes impossible.

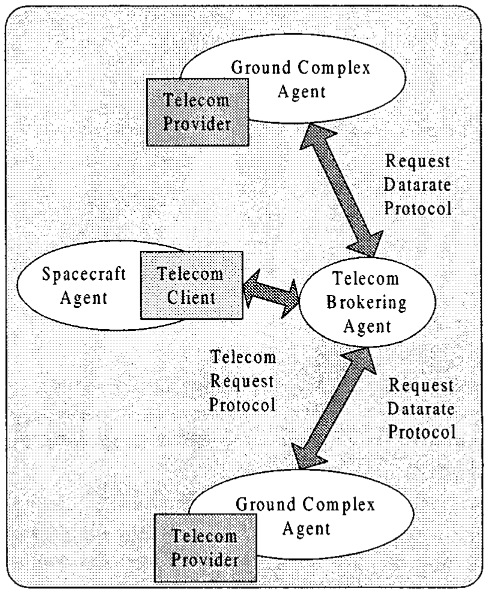

Fig.4 Interactions Protocols

The de-facto agent communication language, used in the existing multiagent systems is KQML (Knowledge Query and Manipulation Language) [4]. KQML is both a language and a protocol. It can be viewed as being comprised of three layers: a content layer, a message layer and a communication layer. The content layer is the actual content of the message, in a particular representation language. KQML can carry any representation language, including logic languages, ASCII strings, etc. The communicative actions such as asking for information, making something true about the environment, passing information to another agent, passing information to all agents are expressed using KQML [4] performatives: ask, achieve, tell, broadcast, etc.

A KQML message conceptually consists of a performative, associated arguments including the real content of the message and a set of optional arguments describing the content in a manner that is independent of the content language. For example, a query about the availability of a particular antenna might be represented as:

(ask-all :content (AVAILABLE (?antenna))
        :ontology MISSION-MODEL)

The message and communication layers encode features describing the lower level communication parameters, such as the identity of the sender and the recipient, a unique identifier associated with the communication process, and the kinds of interactions one can have with the KQML-speaking agent. This way, the sender can attach a type of the communicative interaction, known as a performative, such as an assertion, a query, or a command (tell, ask, achieve).

The following are representative performatives:

Basic responses: error, sorry;
Basic query : ask-if, ask-one, ask-all;
Multi-Response Query: stream-about, stream-all;
Basic effector: achieve, unachieve;
Generator: standby, ready, next, rest, discard;
Capability-Definition: advertise;
Notification: subscribe;
Networking: register, unregister, forward, broadcast, transport-address;
Facilitation: broker-one, broker-all, recommend-one, recommend-all, recruit-one, recruit-all.

The agents have the ability to participate in more than one interaction with another agent at the same time. The structure to manage separate conversations is the protocol. The *protocol* provides additional structured contextual information to disambiguate the meaning of the message. In these conversations, an agent can play different roles depending on the context of the tasks or subjects (Fig. 4). A role is the defined pattern of communications of an agent when implementing a protocol. An agent can assume several roles, when appropriate, depending on the protocol.

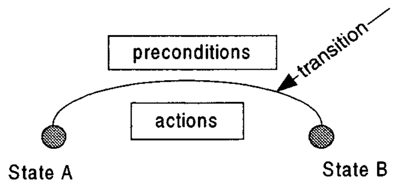

Fig. 5 Interaction Protocol State Transition

A protocol can be described by a *protocol state diagram* (Fig. 5). Transitions in the state of a protocol represent changes in the state of a protocol, i.e. communications between agents. Defining a transition translates into creating a template for a KQML message that will be sent and received by the agents implementing the roles. The various fields in a KQML message will then be used by the message sending and handling rules in the appropriate agents.

Inter-agent communication is regulated by intercommunication protocols. An example of a representative protocol for requesting a telecom communication process is given in Fig.6.

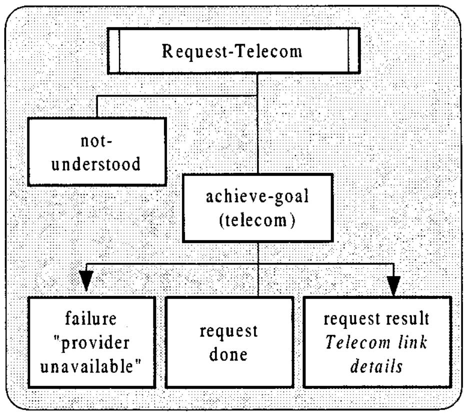
Request-Telecom
not-understood
achieve-goal (telecom)
failure "provider unavailable"
request done
request result *Telecom link details*

Fig.6 Request Telecom Protocol

We chose to use AgentBuilder [1], an integrated toolkit for constructing intelligent software agents. The agents have built-in capabilities for autonomous operation, monitoring their environments, reasoning, and communicating with other agents and users. The interactions between agents are defined by agent protocols. AgentBuilder, using specific protocol generation tools can automatically generate the protocols. The generated rules have to be further completed with more specific information. The protocols are implemented as behavioral rules, which encode the preconditions and actions, associated with state transitions in the state transition diagram of a protocol (see example below).

```
[Telecom Provider Role: Rule defining the reply
  for "Telecom Request" protocol]
(%incomingProtocolMessage.protocol
      EQUALS "Telecom Request" )
(%incomingProtocolMessage.contentType
      EQUALS DateRateRequest)
(%incomingProtocolMessage.performative
                    EQUALS "tell")
(%incomingProtocolMessage.ontology
EQUALS "Telecom Ontology")
(%incomingProtocolMessage.inReplyTo
EQUALS "datarate request")
DO SystemOutPrintln
(Concat ("Received datarate quote from",
  %incomingProtocolMessage.sender))
DO SystemOutPrintln (Concat
   ("The datarate is" , ConvertToString
     (%incomingProtocolMessage.content.price)))
```

Interacting at the goal level.

The interactions, as defined by KQML and FIPA (Foundation for Intelligent Physical Agents) [5] ACL are explicitly defined at the level of actions. To allow for interactions to happen at the level of goal and commitments, rather than explicit commands, the goals are embedded in the content language. Thus, an agent can communicate its objectives, and the recipient agent can do the reasoning and planning accordingly. A simple descriptive language for goals has been defined and has been used as the chosen content language (see example below).

```
(request
  :sender I
  :receiver j
  :content (achieve-goal (at (target 12 84)) camera-x)))
  :ontology "Telecom"
  :reply-with commitment-to-achieve-at)
```

The achieve-goal actions trigger the behavioral rules associated with accomplishing the given goal. The agent reasons and selects the best plan of actions according to its beliefs, its current commitments and the given environment state. It determines how to react if a goal fails, and informs the goal issuer about the reasons of failure. The negotiations are also achieved by the exchange of goals between agents.

This approach provides a distributed solution to the decision-making processes, in which entities interact to achieve the global task of control. The main characteristics of the agents are that they are functionally extensible and they are cooperative. Depending on their roles, they are able to correlate information and events or perform re-active or pro-active actions to prevent or correct a problem.

The protocol for requesting goal achievement is defined for all the agents in the system, which can either be in the position of a manager or a goal-achiever. The interaction protocol for goal achievement conforms to the FIPA-contract-net Protocol in that it adds rejection and confirmation communicative acts [5].

5. Multi-agent Interoperability

The agents are heterogeneous and distributed. This means that they reside on different platforms and have different domains of expertise. In the current implementation, the interoperability is achieved by the system infrastructure rather than by the agents themselves. Agent interoperability is realized at two levels: knowledge level and interoperability mechanisms represented by the current

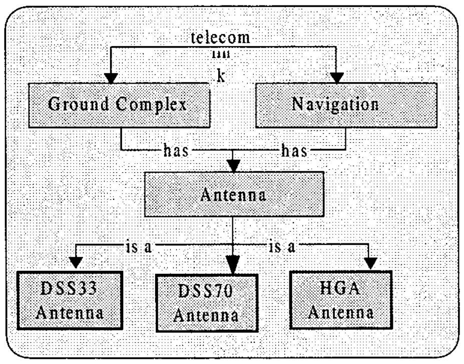

Fig. 7 A Semantic Network for Telecom Ontology

distributed objects middleware (i.e. CORBA, RMI).

At the knowledge level the agents share their knowledge.

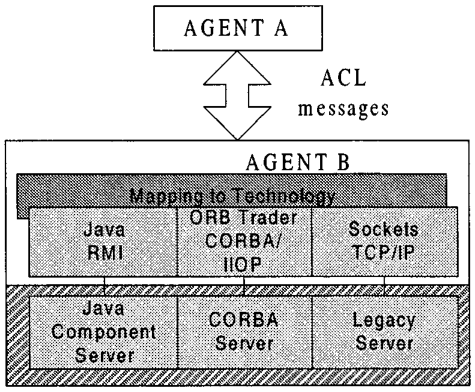

Fig. 8 Enabling Technology Mapping for Interoperability

They do this by interacting at run-time, based on a common set of models, grouped in a shared ontology. The substrate of this process is the concept of virtual knowledge; each agent must comply with the knowledge base abstraction when it communicates with other agents. This technology allows the agents to interact at the knowledge level, independently of the format in which the information is encoded. The ontology represents the formal description of the model of the application domain. The model specifies the object in the domain, operations on them and their relationship (Fig.7). For a given application, several ontologies might be defined. For instance, an ontology for navigation and control, a telecom ontology and so forth. We have experimented with a telecom ontology. Some possible elements in a telecom ontology include: antenna, Ground complex, navigation, and telecom link (the instant communication channel established between the spacecraft and the ground). Ontology can be represented as a semantic network as shown in Fig. 7, and has been implemented as a library of shared objects.

At the lower level of exchanging data structures and invoking remote methods across disparate platforms such enabling technologies as CORBA or RMI have been under consideration (Fig. 8). These standards and protocols are considered as a good substrate on which the higher-level communication levels offered by the agent communication language, can be built. AgentBuilder uses the distributed object infrastructure offered by RMI (Remote Method Invocation use for Java environments).

In Fig. 9 we give a more detailed example of two agents, including beliefs, capabilities, commitments, etc. The agents communicate over a given ontology, the Telecom ontology that contains the necessary meta-knowledge about the application domain.

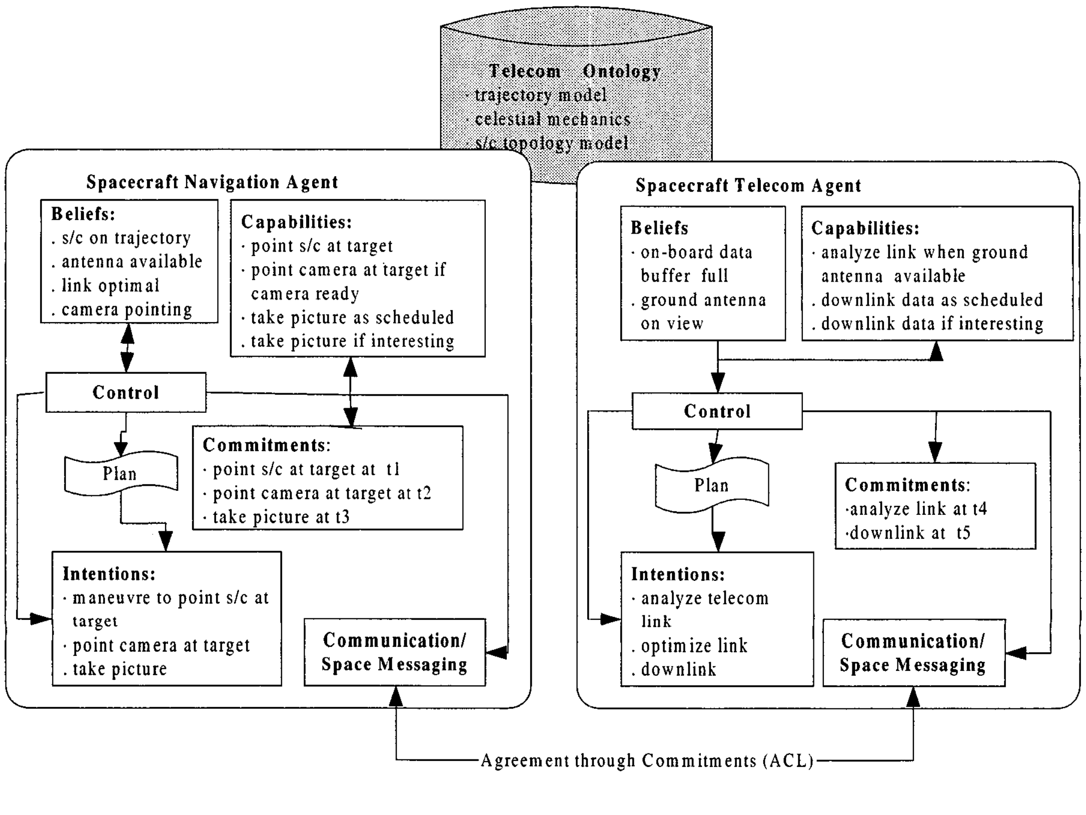

Fig. 9 Navigation/Telecom Cooperating Agents

6. Future Work

The two main goals of this work have been the integration of applications into a common mission framework, and mission independent generalization of applications. There are also several other ways that the agent-based software engineering might prove to be beneficial to mission operations:

- flexibility by providing advice and making independent decisions;
- robustness by distributing decision making processes;

- efficiency by achieving tasks in parallel;
- dynamically optimizing the services by negotiation and coordination,
- fault tolerance in the sense that it can recover from most exceptions (ex: when a link that is part of the connection can no longer be provided, an alternative link is automatically set up);
- notification to the task managers (users or agents);
- interoperability across heterogeneous languages and platforms;
- integration of legacy systems.

There have been several agent-based approaches to system control and in particular to spacecraft control or mission operations. We can cite the Remote Agent Experiment for automatically spacecraft controlling and commanding NASA DS1 mission [2], or multi-agent approaches such as multi-operation support operations, and handling malfunctions in the Reaction Control System (RCS) of NASA's space shuttle [7]. Some of them solve the problem using one single agent, such as for DS1, or in the case of a multi-agent approach have no explicit model for inter-agent communication or application integration.

There is some other multi-agent approaches offering more elaborate models for cooperation between agents such as in [16]. Supporting agent mobility for code mobility, agent migration and agent cloning, provides for scalability and increased fault tolerance. There should be better ways to mitigate the two quite different approaches, the communicative intelligent agent and the mobile agents [9].

We consider that the experiments with a given infrastructure for building multi-agent systems, will help to better understand fundamental issues such as the nature and content of space mission models, how to use them and how to express them in such a way that they might be re-used. By conceptually considering the mission system as a distributed process, the agent-based engineering offers a promising perspective to define the system components at a higher conceptual level, ultimately shortening the development process and offering support for an advanced problem solving framework.

Further work will focus on defining more elaborate mission ontology, refining the communicative models to allow for complex agent interactions such as negotiation, and investigate the agent mobility.

7. References

1. [AgentBuilder 1998]: User's Guide, Reticular Systems, Inc., 1998.
2. [Bernard 1998] Bernard, D.E., et al.: Design of the Remote Agent Experiment for Spacecraft Autonomy, Proc. of the IEEE Aerospace Conference, Aspen, 1998.
3. [Bond 1988]: Bond, A., H., Gasser, L., eds.: Readings in Distributed Artificial Intelligence, Morgan Kaufmann Publishers, San Mateo, CA 1998.
4. [Finin 1992] Finin, T., Weber, J., Wiederhold, G., Genesereth, M., Fritzson, R., McGuire, J., McKay, D., Shapiro, S., Pelavin, R., Beck, C.: Specification of the KQML Agent Communication Language (Official Document of the DARPA Knowledge Sharing Initiative's External Interfaces Working Group), Technical Report 92-04, Enterprise Integration Technologies, Inc., Menlo Park, California, 1992.
5. [FIPA, 1998] FIPA Specification, http://www.fipa.org/spec/fipa98.html
6. [Genesereth 1997] Genesereth, M., R.: An Agent-Based Framework for Interoperability. In Software Agents, J.M. Bradshaw, AAAI Press/MIT Press, 1997.
7. [Ingrand 1996] Ingrand, F.F., Georgeff, M.P., Rao, A.: An Architecture for Real-Time Reasoning and System Control, http://www.laas.fr/~felix/publis/ieee-exp92/ieee-diagl2h.html
8. [Jennings 1998] Jennings, N.R., Sycara, K., Woolridge, M.: A Roadmap of Agent Research and Development. In Autonomous Agents and Multi-Agent Systems, 1, 275-306 , Kluwer Academic Publishers, Boston, 1998.
9. [Labrou 1999] Labrou, Y., Finin, T., Peng, Y.: The Interoperability Problem: Mobile Agents and Agent Communication Languages. In Proceedings of HICCS 32, the 32th Hawaii International Conference on System Sciences, Jan. 1999.
10. [Mandutianu 1999A] Mandutianu, S., Cooperative Intelligent Agents for Mission Support. In Proceedings of IEEE Aerospace Conference, March 1999.
11. [Mandutianu 1999B] Mandutianu, S., Stoica, A.,: An Evolvable Multi-Agent Approach to Space Operation Engineering. In Proceedings of the International Conference on Multi-Agent Systems, July 1999.
12. [Rao 1995] Rao, A., Georgeff, M.: BDI Agents: From Theory to Practice. In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, June 1995.
13. [Siewert 1996] Siewert, S.: A Distributed Operations Automation Testbed to Evaluate System Support for Autonomy and Operator Interaction Protocols. In Proc. of 4th International Symposium on Space Mission Operations and Ground Data Systems, ESA, Forum der Technik, Munich, Germany, September

1996.(http://www-sgc.colorado.edu/people/siewerts/intl_space_ops_96/SO96_6_07.html

14. [Shoham 1995] Shoham, Y.: CSLI Agent-oriented Programming Project: Applying software agents to software communication, integration and HCI (CSLI home page), Stanford University, Center for the Study of Language and Information, 1995.
15. [SPCS 1998] http://www.spcs.org.
16. [Tambe 1995] Tambe, M., Johnson, W.,L., Jones, R.,M., Ross, F., Laird, J., E., Rosenbloom, P.,S., Schwamb, K.: Intelligent Agents for Interactive Simulation Environments, AI Magazine, Spring 1995.

Sanda Mandutianu is a senior member of the technical staff at Jet Propulsion Laboratory, California. She holds an MS in Physics from University of Bucharest, Romania. She has been working and leading projects in Artificial Intelligence since 1983 in various fields such as knowledge representation, cognitive science, natural language understanding, distributed computing and their applications. Her current interests are cooperative software agents, agent communication languages, distributed spacecraft and spacecraft autonomy.