Plug-and-Play – An Enabling Capability for Responsive Space Missions

Thomas Morphopoulos, *Microcosm, Inc.* El Segundo, CA
L. Jane Hansen, *HRP Systems, Inc.,* Torrance, CA
Jon Pollack, *HRP Systems, Inc.,* Torrance, CA
Jim Lyke, *Air Force Research Laboratory Space Vehicles Directorate,* Albuquerque NM
Scott Cannon, *Utah State University,* Logan UT

PLUG-AND-PLAY – AN ENABLING CAPABILITY FOR RESPONSIVE SPACE MISSIONS*

Thomas Morphopoulos, Microcosm Inc.±, El Segundo CA E-mail: tom@smad.com
L. Jane Hansen, HRP Systems, Inc. Torrance, CA E-mail: jhansen@smad.com
Jon Pollack, HRP Systems, Inc. Torrance, CA E-mail: jonp@smad.com
Jim Lyke, Air Force Research Laboratory Space Vehicles Directorate, Albuquerque, NM E-mail: lyke@plk.af.mil
Scott Cannon, Utah State University, Logan UT

ABSTRACT

A self-organizing network concept, leveraging commercial approaches is under development to support responsive space avionics networks. The work is being done in support of an Air Force contract, including the following elements: a network manager (hardware and network medium specific component), mission manager (mission objective specific), and GN&C algorithms for a four state activity (power-on, initialize, nominal GN&C, safe). The current work emphasizes the resource manager, which is responsible for discovering resources as they come on-line. It also manages real-time data descriptions and health/status information for potential consumers of each produced element within the overall network. These mechanisms form a basic system for plug-and-play, in which the components of a system can be rapidly assembled with minimal need to write detailed, low-level code pertaining to the interface of each element. The resulting automation allows system designers to focus on design of higher-level software in an object-oriented fashion, a process that itself might be automated under this concept.

INTRODUCTION

Critical to creating a truly responsive space paradigm is to put in place an environment where spacecraft (and launch vehicles) can be built to inventory. It may not be necessary to have multiple spacecraft in "shrink-wrap" and sitting on a shelf, but key components must be built-up and ready to integrate, based on specific payload and/or mission requirements. Thus, spacecraft capable of supporting responsive missions will need to embrace the PC-based concept of "plug and play", where the user plugs a peripheral (e.g., a mouse) into a PC-resident USB port, triggering an automated process in which the operating system installs the correct driver, configures the system parameters, and then allows the user to begin operations without manual intervention. This paper will describe an architecture approach for avionics with much the same aim, but more ambitiously in a system in which eventually all avionics components can be quickly assembled with minimal human intervention. Several elements of this approach are under active research at the Air Force Research Laboratory (AFRL). For example, one AFRL contract with Microcosm is developing a self-configuring avionics network with emphasis on guidance, navigation, and control (GN&C) components, which is the central focus of the present paper. In this effort, representative GN&C components such as sensors, actuators, and processors (8-bit, 32-bit and 64-bit) will be assembled into a rudimentary ground testbed. More importantly, the testbed will act as a staging area for concepts such as self-organizing interfaces. The network will be extensible, eventually expanded to include multiple spacecraft (docked together) integrated into a single system of systems.

Some of the benefits of the approach discussed in this paper include:

- Reduction in R&D costs by leveraging of COTS hardware/ software and standards
- Reduction production costs by creating a robust, flexible protocol that meets the needs of GN&C and other spacecraft applications
- Increase in spacecraft reliability by eliminating software/hardware configuration errors and providing built-in redundancy and reconfigurability of flight systems
- Increase in system performance through realizing lower avionics and associated interconnect mass.

- Copyright 2004, Microcosm, Inc. Reproduced by the Responsive Space Conference, with permission.

± 401 Coral Circle, El Segundo, CA 90245

This paper is organized as follows. In the following section, we provide examples to illustrate the problem. The next section discusses the desirable attributes of a space avionics network. Next, work on commercial plug-and-play standards is reviewed. Following this, we describe near-term testbed configurations under development at Microcosm and AFRL.

MOTIVATION

Illustrative examples help crystallize some of the more problematic issues in developing avionics to manage a number of diverse components. We first discuss a simple network of sensors, followed by an example of an avionics system.

Simple Network

We consider the example of a payload consisting at first of four simple scalar sensors, such as thermometers or dosimeters, providing a single host interface to a spacecraft through a data handling system (DHS). Several implementations are sketched in Figure 1. The simple method in Figure 1a employs DHS-to-sensor interfaces with point-to-point connections, and implements the host interface the same way. In this example, a fifth copy of the simple sensor must be added to the network, however in the Figure 1a approach, the DHS cannot accommodate it as a fifth point-to-point connection is not available. This limitation does not exist in the multidrop interface shown in Figure 1b. Such interfaces (examples include MIL-STD-1553 and RS-485) are in principle very scalable, and a fifth sensor (or even a 200th sensor) can in principle be readily accommodated. The Figure 1b implementation is not particularly robust, as its single multidrop interface and static, single master are single points of failure. Modern spacecraft typically employ (in the case of MIL-STD-1553) dual redundant busses and one or more backup masters (capable of implementing the DHS function). Even these modulations of the basic theme are constrained by the use of a centralized control approach. One way to eliminate it is shown in Figure 1c. Here, each sensor can connect at several points to either another sensor or to the satellite host. In this case, a dynamic mastering concept could be used, eliminating the need to even design a centralized DHS. This topology is amorphous in the sense that it is not necessary to implement a ring or bussed connection to all elements. It is easy to build into this concept a form of improved robustness by using multiple connections between elements, since only a single connection between each element and the host interface (through any number of intermediaries) is required for successful operation.

Spacecraft Avionics Example

A second example we consider is that of several components to be connected within the spacecraft, as shown in Figure 2. These are GNC components, including reaction wheels, gyros, star tracker, and a GPS system. The objective is to connect to these components to form a coherent GNC system. It is not sufficient to simply connect these elements into a bus structure such as MIL-STD-1553. It is necessary that the components agree upon the roles that they would play inside of the GNC system. Normally, expectations are not so high. No component intelligence is really expected, and significant amounts of painstaking effort go into hand coding detailed software for both the components and the overall application involving the set of components. If somehow, on the other hand, a set of GNC components could "understand" their specific role within a complex system and somehow present a very simplified interface to a user/designer, then integration could be far more rapid. Humans could be freed from worrying about the details of specific mappings of signals, unification of measurement standards within the different devices, compensation, calibration, etc.

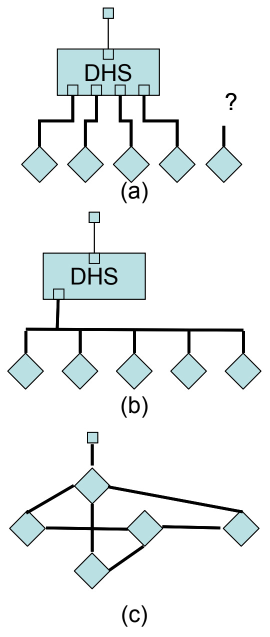

Figure 1. Simple network / data handling system (DHS) configurations. (a) Point-to-point. (b) Multi-drop. (c) Amorphous.

Each of the preceding examples illuminates a different facet of the problem in interfacing complex systems. In the first example, the issues are related to the physical topology. It is necessary for a designer to be concerned over the specific arrangement of components within a system, and usually limited flexibility exists in the latepoint addition of components. In the second example, we strive to automate the organization of components within a complex system, even within a narrowly defined domain such as GNC. We desire an ability to combine components into a unified system rapidly, but in practice get bogged down over many low-level interface details such as the development of device drivers, worrying about the coherence in the applications bundled with each device, and then attempting to write some unifying code in a central computer (likely) that manipulates and extracts information throughout the network of components to produce the function of a GNC system.

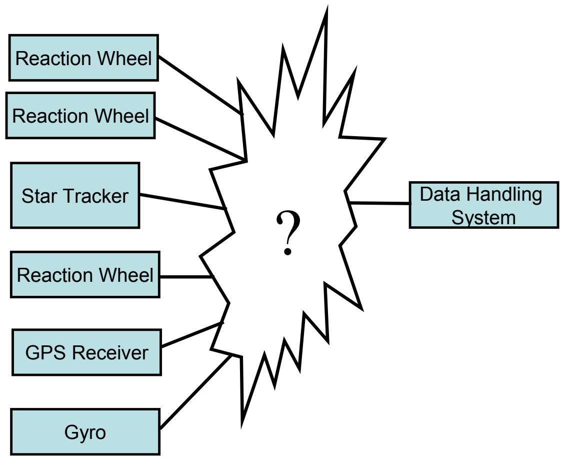

Figure 2. GNC Interface Example

REQUIREMENTS OF A RAPID-RESPONSE AVIONICS NETWORK

In this section, we briefly address the desired requirements for the components of a distributed network and more importantly for the way that the network itself operates. Different names could be applied, and we have chosen the term *appliqué sensor* ("peel-and-stick") *network* for this concept. In fact, we envision the network is not limited to simple sensors, the complex sensors, actuators, processors, and other elements as needed in a complex avionics system.

Machine Negotiabilty of Interfaces. The hallmark of a good plug-and-play approach is to reduce the act of adding hardware to a system to the simple act of plugging or (in the case of some wireless systems) simply bringing components within adequate physical proximity. Providing the ability for intelligence within the network to perform interfacing, complete with the conveying of interface documentation and automatic adjusting between components to ensure compatibility is the highest goal of an avionics infrastructure in responsive space. This is because humans are removed from the loop, and interface control documentation is mechanized.

Distribution. It is important to move away from centralized control, since this concentrates the amount of software and wires in one location, and the number of errors and difficulty in test of software is related non-linearly to code size. Ideally, an appliqué sensor network needs no centralized control, but the functions related to the management of each component of the network as well as the network itself are distributed as tasks performed by the participants of the network. This analogy is Internet-like in the sense that the Internet is itself decentralized: it provides users with a robust catalog of services and information.

Amorphous. It should not be necessary for users to be concerned over the exact shape of the network, no more than consumers worry about the shape of the power grid when they plug a device into a wall socket.

Time and space. Embedded systems in particular require conveyance of precise notions of time and spatial location. A coordinated notion of time is difficult to achieve in distributed systems, particularly if a non-deterministic networking approach is employed. In GNC systems, for example, the precise location of each component within the spacecraft is required, and precision in placement and capture of placement information is counter to the notion of rapid assembly and integration.

Hot-swappable. The nodes of this network must be dynamic. Specifically it should be possible to add or remove components *in situ* without disrupting the operation of the network.

Fault tolerance. In part, the concept of hot swapability itself conveys a certain notion of robustness, since redundant copies of components could be used in a network, and hot swapability covers the possibility of one copy failing. Beyond that, other concepts and fault tolerance such as timeout should permeate the design of an appliqué sensor network for spacecraft at all levels.

Coherence and unity. To the maximum degree possible, the appliqué sensor network should look like a single system, and application design should not necessarily need to know of the existence of a specific physical devise, but should be able to operate on the idea of a virtual service or devise whose actual functions are supplied by one or more real physical devices in that in the network.

In-situ Reconfiguration. The ability to update the software within any portion of the network in situ is an important feature to provide the possibility of rectification, i.e., the correction of errors discovered during integration or operation.

EXISTING APPROACHES

In this section, we consider briefly the body of existing work to develop network standards in plug-and-play. To facilitate this discussion we present an appliqué sensor network reference model, shown in Figure 3. We then consider the number of approaches in physical interconnect and commercial plug-and-play standards.

Reference model.

The Figure 3 network model does not correspond directly to an open system interconnect (OSI) stack. The key set of "actors" in this model are the devices located within the network (forming nodes), the network itself, and a host user interface and applications that are composed to operate upon the network.

For *devices* the *physical interface* is necessarily a low-level interface, which is concerned over the details of specific discrete digital, analog, and power connections. The device may be a thermometer, camera, microscope, mirror, motor, etc. With reconfigurable technologies, it may be possible to reduce the number of custom components required in a physical interface, since in devices such as field programmable gate arrays, the functionality can be software controlled. In any event, the goal of physical interfaces is to convert the very specific signals necessary to control and manage a specific device to a unified representation conveyed by a *driver* layer.

The driver layer then organizes the eclectic universe of potential devices in a way that can be operated upon by the rest of the layers in an appliqué sensor network. A *resource manager* which is also resident on each node is a computation resource that manages both network and driver interfaces.

The *network* physical interface is a small OSI stack that interposes the resource manager and the network itself. The interposer renders the details of how the network is physically implemented as unimportant (to first order) to the way that the resource manager operates. In this case, the physical and the network physical interfaces are simply access portals.

In this reference model, we also include the notion of a *semantic* layer. The semantic layer is intended to bind a higher level of meaning to the access and control of a node. A resource manager can mechanically organize device parameters, units of measure, services available, and other details. The concept of "semantic" is somewhat gray, as a very sophisticated resource manager may blur the boundary between the mechanical catalog of device features and the higher-level interface of these features presented to the local (nodal) and network (global) application layers.

The local (nodal) *application* layer represents software that is bundled with the node.

Two other non-node-specific layers are added to this model. The first is *network application*. In the case of the GNC domain problem (Figure 2), the network application might be "guidance". Guidance requires both observability and controllability defined as a composition of services from these elements. The degree of sophistication in the semantic features in each node can simplify the development of guidance as a global application by eliminating the need for units conversion, for example.

The final level in the reference model is the *user interface*. The user interface is intended to represent in the case of a satellite the command infrastructure and telemetry produced as a result of commands. It represents a high-level request to access one or more the services provided by the network and is not to be confused with network applications. User interfaces are more conducive to scripting approaches as opposed to the control algorithms that might reside in a resource manager.

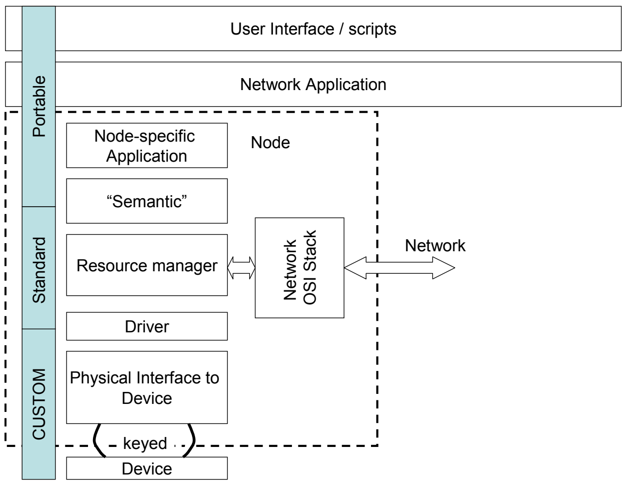

Figure 3. Applique sensor and network model.

COMMERCIAL INTERCONNECTION AND PLUG-AND-PLAY APPROACHES

We consider briefly the significant body of work done to develop physical interconnect and smart network standards. Physical add or connect approaches can be related to the network physical lawyer in the reference model. The networking standards may be reconciled against the nodal part of the reference model combined with the network layer and network application parts of the reference model.

Physical interconnect

Figure 1 illustrated three different topologies based on wired buses in some arrangement between the elements of a network. Here, we identify a variety of ways of achieving physical interconnect between devices. It is important to reinforce that exact physical medium used to network nodes in an appliqué sensor network is not limited to a particular method or even a single method. Terrestrial networking thrives because of the ability to bridge across a variety of standards. Reasons of standardization, performance, economy, or other considerations may drive specific choices.

Point-to-point. Point-to-point is a basic and direct wire connection between pieces of system. Example point to point signaling standards include RS-232, RS-422, Firewire, LVDS.

Multidrop. Multidrop interfaces refer to cases where more than two components can attach to a serial or parallel wiring structure. Examples include MIL-STD-1553 (which is usually implemented as a dual, multidrop bus), VME, PCI, and RS 485 (which is a multidrop version of RS-422).

Switch fabric. The Advent very high-speed serial signaling standards such LVDS motivated development in the 1990s of switch fabric approaches. They are Internet like in the sense that nodes can be connected to hubs, and hubs can be connected to each other. The set of these building blocks can be used to form a network of arbitrary size. Rather than the deterministic transactions conducted on point-to-point and multidrop systems, switch fabric systems tend to rely on packet based communications. Intelligent routing approaches examine the structure of these packets to facilitate moving them from sources to destinations. High-bandwidth links and non-blocking crossbar hubs with large numbers of ports contribute to the construction of extremely high-performance networks. The definition of robust and lightweight messaging interfaces (an example of which includes FPMI)[1] makes switch fabrics especially relevant for embedded systems. Examples of switch fabrics include Infiniband, RapidIO, Myrinet, and Spacewire.

Communications on power. It is possible to superimpose communications on power distribution systems, and examples have been developed for residential (X-10), automotive[2], and even spacecraft applications. These networks function like multidrop networks, and the hostile environment represented by power systems is not conducive to high-speed transfer (e.g., Powerlan[3] operated at 9600 baud). Nevertheless, the simplicity of a power-based network concept has some attraction in the development of simple networks since only a power connection need be supplied between elements of network, thereby dramatically reducing the complexity of the wiring harness.

Wire less. The recent explosion in development of wireless technology has been fueled the IEEE 802 and Bluetooth technology standards, and many other wireless schemes have been examined for distributed sensor networks. Wireless systems are tremendously appealing because they eliminate altogether the need for any physical connections except power (and even that is not essential in rf-powered schemes). Peer-to-peer versions of wireless networks permit the easy access and commute ability between components and communications. Wireless systems are not without their problems. Of greatest concern for spacecraft, for example, is the possibility of interference. One hope in dealing with EMI problems is that in many cases frequency hopping approaches are used, and it seems in principle possible to create a version of a standard in which the "keep out" spectral bands could be defined (i.e., a configurable protocol), in effect providing opportunities to work around interference bands in complex systems.

Free-space optical networks. A compelling technology particularly for high-performance applications is the possibility of simply beaming information between the points of the system. Free-space optical networks, and even light guided networks based on fiber-optic, offer attractive benefits that are not possible in wire-based networks, which include elimination of magnetic interference, simplification of impedance matching problems, and lower loss in high-speed information transfer. Free space optical connections, however, have the obvious problem of lines-of-sight clearances between the parts of the network, which makes this less attractive for flexible applications.

Commercial plug-and-play standards

Although the exact origin of the term "plug-and-play" is unclear, most attribute it to the early work in the PC industry to overcome the problems in interfacing many third party peripheral components to a variety of potential motherboard configurations.

A number of significant advancements have been made in terrestrial plug-and-play systems. In addition to the well-known PC form of plug-and-play (PnP), a number of dynamic network-based approaches have been proposed. Examples include: HART, Universal PnP, IEEE 1451, LonWorks, and JINI. These approaches and key concepts are briefly discussed here. Most of these concepts can be related to Figure 3 reference model.

HART.[4] The Highway Addressable Remote Transceiver (HART) protocol was introduced by

Fischer-Rosemount Ltd. in the 1980's to provide a method of command and control of smart instruments, and it is still in widespread use today for industrial control and measurement. Its physical layer is based on frequency shift key modulation impressed onto a 20mA current loop. One of the first attempts at a universal approach for managing a number of complex sensors, the HART protocol introduced a number of advanced concepts, including a device description language (DDL) long before the introduction of XML. The main disadvantage of HART is its relatively low performance and lack of scalability.

Echelon LonWorks[5]**.** LonWorks (later proposed as EIA 709) is a very popular distributed networking system developed by Echelon. It employs an ad hoc networking system, complete with router components, and supports a variety of interconnect signaling approaches that can presumably be mixed within a complex network. As its resource manager, Echelon employs a proprietary Neuron processor. One limitation with the LonWorks system is the public availability of intellectual property for designing a radiation tolerant form of the Neuron and its network elements.

IEEE 1451[6]. In the mid-1990's, an industrial consortium led by NIST developed a set of smart sensor standards, now available as the IEEE 1451 series. Several key concepts were contributed by these standards. Network capable application processors (NCAPs) represent a form of resource manager in IEEE 1451. Device information is conveyed through a transducer electronic data sheet (TEDS), representing a variant of the DDL approach used in HART. Networking of nodes for the purposes of exchanging configuration information is accomplished through a very simple serial, multidrop standard referred to as Microwire. This configuration-only network bus differs somewhat from the Figure 3 model in that high-performance data connections between nodes would require a separate network interface (not shown). One concern about the IEEE 1451 system is in the rigid definition of TEDS as a memory mapped structure, although some of the 1451.4 working group are considering the extension of the TEDS concept to include XML support.

Universal PnP[7] The UPnP standard actually combines other standards, such as HTML, XML with a focus on network centricity and independence from individual devices and drivers. As such, the lower, custom levels in the Figure 3 model are essentially ignored or left to the implementer. Instead, UPnP focuses on the concepts of discovery, control points, and "eventing" through the use of its simple service discovery protocol (SSDP) . Discovery refers to the process by which the components of a self-organizing network communicate their presence, which uses protocols referred to as "advertisement" and "search". Information for each network device consists of device and service descriptions (in XML). Control points exploit advertised services to implement network applications. "Eventing" refers to a publish-subscribe mechanism used in advertised services by control points.

JINI[8] The Java Intelligent Network Infrastructure (JINI) was developed by Sun to extend the utility of the Java programming language to support device interoperability through an infrastructure referred to as Java Remote Method Invocation (RMI). Instead of the control points used in uPNP, JINI employs "lookup services", which are consulted by nodes within a JINI network. The nodes themselves contribute services, and nodes upon initially joining a JINI network use lookup services both to *advertise* (register) services that they can contribute or to find services that they may need. JINI provides support for the development of distributed applications by supporting *remote events* and *transactions*. These features provide scalability to JINI networks by permitting application code to be amortized over the components of a network. One interesting feature of JINI is the *leasing* concept. JINI grants services using *leases,* which add a certain robustness to the network. For example, in the case of a workstation requiring the use of a printer, if the printer is removed, the lease eventually expires, and it cannot be renewed[9]. One of the drawbacks of JINI is the expectation of the use of the Java programming language, which has been met with some "pushback" from groups such as Microsoft.

Salutation. "Salutation" and "Salutation Lite" are royalty-free standards defined by the non-profit Salutation Consortium[10] to promote open methods of plug-and-play support to heterogeneous components. Like UPnP, Salutation supports registration and discovery mechanisms using salutation managers, which may themselves be distributed about a network. The salutation manager is described as a "service broker"[10], and these "brokers" coordinate on behalf of the variety of devices in the network to provide knowledge and exchange of capabilities. Services are "atomically" composed of functional units, which is the lowest level at which a meaning feature can be defined in the salutation system.

Obje.[11] Obje appears to be an evolution of the JINI approach introduced by PARC to introduce semantics in plug-and-play networks. Whereas most plug-and-play systems provide mechanisms to dynamically add, remove, discover and transact across a network, this

[10]http://www.salutation.org.

infrastructure is devoid of understanding how the various pieces of a network can be used in distributed applications. Drivers and application code must be written by users. Most of the details of Obje are presently clouded by propriety, but two apparent distinguishing characteristics include (1) extensibility of the service/discovery protocols themselves and (2) provision of flexible meta-interfaces between the application and device to enhance code mobility. In fact, Obje claims interoperability with other plug-and-play standards such as UPnP and JINI.

AN INITIAL IMPLEMENTATION FOR AEROSPACE SYSTEMS

A common theme of the previously described plug-and-play systems is the ability to distribute components rapidly and flexibility. In all cases, service discovery and delivery mechanisms are defined to provide a uniform method of automatically integrating these components when introduced. These systems for the most part lack several important elements. First, none of these methods provide a definite synchronization system capable of sub-millisecond precision between components. Second, semantic support is weak, and while not a principal limitation, it does still necessitate the need for potentially protracted software development on each component to supply driver and application content. Finally, the interconnection / functional interface support in these standards is lacking. This limitation is manifested in two ways. First, there is no support for passing large quantities of data (i.e, hundreds of megabits/sec). Most of the standards may use their configuration infrastructure as identically the conduit for inter-component communication in transactions, therefore these flexible but low-performance networks are impractical for high-volume data transport. Second, for additional functionality, such as a discrete signal for synchronization or other specialty needs, the standard do not provide support for describing and registering other signals that might be shared between components. These missing features are precisely those that do not work so cleanly in an Internet-like system, where pieces can be geographically distributed and do not have the need for real-time or other connective conduits.

To study some of the other issues inherent in creating a plug-and-play infrastructure for a single domain (GN&C), we set out to develop our own simple testbed. This testbed would not itself be the final standard, or even necessarily a good interim one, but rather an opportunity to examine and refine the definition of a minimal set of requirements for a universal component plug-and-play system for real-time embedded space and missile systems. Under AFRL support, Microcosm is developing a software-based product that will provide network dependent and independent components, while leveraging COTS networks to the greatest extent possible to create a self-configuring, avionics network, with an emphasis on the GN&C system. The first phase of demonstration is in process and there will be a more extensive system level demonstration later in the year. The first demonstration sets up the infrastructure needed to create the demonstration including the implementation of a CANBus protocol that will facilitate discovery, command/response transactions, and general data delivery. GN&C components, such as single and multiple axis gyros and accelerometers, magnetometers, and LEDs that represent thrusters, have been purchased and are being used in the demonstrations.

There are four primary components to the software product being demonstrated: the mission manager, the resource manager, network manager, and the GN&C application software. The mission manager (MM), demonstrated in the second phase of the development process, is the component of the system that understands the mission objectives, requirements, and success criteria. It is in this software that decisions are made regarding the mission phase and the algorithms that must execute at any given time during a mission. The Resource Manager (RM) understands the resource discovery process and maintains the information regarding data descriptions and system as well as subsystem health and status. A component of the RM is the local resource manager that provides an API (application programming interface) to resident software applications by abstracting the physical activity needed to access the data while providing a mechanism for error handling and reporting. The Network Manager (NM) understands addressing, routing, protocol, and interface associated with the medium over which data is being supplied. This is the component of the software that will be changed as different medium and protocols are introduced. Finally, the GN&C application software understands the required sensor inputs, processing and outputs, the control laws by mission mode, and the actuator distribution and execution functions. Each of the elements is needed to make the overall system operational

The key to implementing a true plug and play capability for GN&C applications is reducing the data type down to its molecular or atomic level. By evaluating each piece of data at the lowest level, the application software is no longer dependent on traditional subsystems or LRUs (line replaceable units) but rather focuses on the primary inputs and outputs that need to be generated. The actual number of input types needed for GN&C can be limited to four: time, rotation measurements, translation measurements, and third body angles. In reality this equates to a finite number of atomic level data elements that need to produced by the various sensors and sensor suites. These include: time stamp, rotation angles (3 components), rotation rates (3), rotation accelerations (3), translation position (3) translation rates (3), translation accelerations (3), earth angle (2 components), earth angle rate (2), sun angle (2), moon angle (2), and star angle(s) (2 components for each star in the field of view). The same is true for the actuator commands, output from the GN&C software. There are commands for requested thruster force (3 components), requested thruster torque (3), wheel momentum, and magnetic torquer duty cycle. The GN&C application software for these demonstrations will focus on these rudimentary input and output components.

During the first demonstration, the focus will be on the resource manager and the GN&C application software. The actual demonstration configuration is shown in Figure 5. The resource manager will provide the guidance for the system during discover, nominal operations, and as components are failed or added to the system. The GN&C application software will implement a "bang-bang" algorithm that will respond to various sensor combinations on the bus and motion of the spacecraft, simulation through the use of a rate table. .

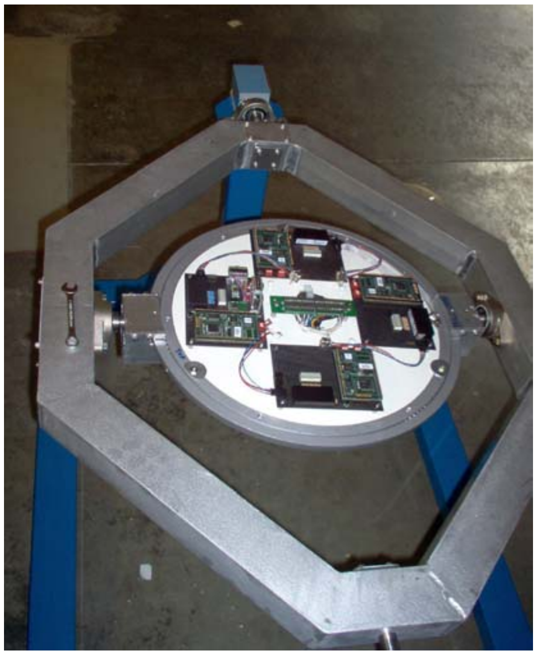

Figure 5 Microcosm Plug and Play Demonstration 1

The demonstration scenario includes initialization, basic operations, motion detection by the GN&C algorithms, sensor failure, and the addition of a new sensor to system. At initialization the RM on the host node will boot and announce to the network(s) that it is available. At this time, if there had been a different RM, an arbitration process would ensue. However, for the demonstration, this will occur at power-on. The RM inventories the nodes on the network(s) as well as the application software that is available. Each of the nodes will announce themselves by requesting a query through the command/response process. For GN&C components, there are three basic elements to each data element: the data, the configuration information, and the health/status information. The data is simply the information the sensor senses. The configuration information includes such things as mounting, alignment, and calibration data, as well as moding for the subsystem itself. The health/status data will include a brief history of the components performance and its current operational status.

The system will function in a basic mode, given a nominal attitude on the rate table. The nominal algorithms will make use of the "best" sensor information available in the system. Next, the rate table will be adjusted to represent the need for an attitude correction. Based on the initial set of sensor data and the motion, the GN&C bang-bang algorithm will request thruster firings. This will be demonstrated through the use of LEDs. When the rate table is re-set, the system will stop requesting thruster activity and return to the nominal state.

The next step of the demonstration will include the arbitrary failure of various sensor(s). At this point the GN&C algorithms will attempt to find alternative producers of the sensor data required to maintain attitude. If adequate sensor information is not available, the GN&C algorithms will command the spacecraft to a safe mode. For this early step of the demonstration, alternate, but less accurate, sensor data will be available. The rate table attitude will again be adjusted to the second position. This time, because the data is less robust than during the initial adjustment, the system will not attempt to fire thrusters. The tolerance on the sensor information will not allow the algorithms to fire. The attitude will be returned to the nominal state.

Next, a higher fidelity sensor (3-axis gyro) will be added to the sensor configuration. The attitude adjustments will again be made and this time, the GN&C algorithms will request that thrusters be fired. When these sensors are failed in the second attitude position, the GN&C algorithms will command the spacecraft to a safe mode.

This demonstration shows the ability of the system to autonomously discover elements on the network, not only at initialization but also as they are added or taken away during regular operations. The configuration data associated with each of the sensors provides the means for the GN&C algorithms to determine the fidelity of its solutions, based on different levels of sensor inputs. The GN&C algorithms will demonstrate the ability of the spacecraft designer to create a set of algorithms that will accommodate redundancy, reconfiguration, and fault-tolerance, all of which can be autonomously executed. The software will be centralized for this demonstration but in the future demonstration(s), the software can be distributed with the mission manager providing orchestration of the entire process.

The implementation of this type of system allows the spacecraft GN&C engineer to easily configure the spacecraft attitude and orbit control systems, and tailor the control logic to any desired hardware suite without needing a brand new software development program for each new spacecraft and new set of sensors and actuators. Thus, the concept as envisioned covers not only rapid prototyping, but also facilitates laboratory testing and last minute component replacement. Each of these will contribute to responsive space.

CONCLUSIONS

For a variety of reasons, an increasing emphasis has been placed on the pace at which space objectives can be achieved. The vision of responsive space will be technology-driven, but a "first-principles" approach is required as opposed to a random or brute-force application of technology. It is important not to confuse, for example, the idea of simply trying to build a system quickly, with the concept of developing the infrastructure to build a system quickly. The terrestrial form of "plug-and-play" technology in consumer PCs is an illustrative example of this dichotomy. Nearly a decade of development was required to refine this approach, but as a result it is possible to introduce a new device (such as a flash-based disk drive) in a matter of seconds.

Avionics, to include electronics hardware, software, and interfaces is central to the drive for responsive space. As the PC industry evolved (and continues to refine) an infrastructure for machine-negotiated interfacing for responsiveness, so too must space avionics be recast for responsiveness. In this paper, we have shown how adaptive avionics can play a role. Specifically, we have discussed a system demonstration of components for a GN&C application that confirm that a level of plug and play can be achieved in an embedded spacecraft architecture.

REFERENCES

[1] Moore, Mike, "Application Program Interface (API) of the Fusion Processor Messaging Interface (FPMI)", unpublished user's manual, draft, February 2002.

[2] http://www.yamar.com.

[3]Powerlan http://www.powerlan-usa.com

[4] Bowden, Romilly, "HART Field Communications Protocol: A Technical Description", publication of Fisher-Rosemount Limited, West Sussex England, June 1996.

[5] Echelon (http://www.echelon.com)

[6] IEEE 1451 (http://ieee1451.nist.gov)

[7] Kastner, Wolfgang and Markus Leupold, "How Dynamic Networks Work: A Short Tutorial on Spontaneous Networks",

[8] JINI http://www.jini.org

[9] Helal, Sumi. "Standards for Service Discover and Delivery," Pervasive Computing (IEEE), 2002.

[10]Pasco, B. "", a white paper of the Salutation Consortium, 6 June 1999.

[11] "Application of Self-Configuring Network for Plug-and-Play Device Control", USAF contract F29601-02-C-0240, 12 November 2002.

RELATED PAPERS

[1] Edwards, W. Keith et.al., "The Case for Recombinant Computing", Xerox PARC TR CSL-01-1, 20 Apr 01.

[2] "Application of Self-Configuring Network for Plug-and-Play Device Control", Contract F29601-

[3] "Transformation Trends", presentation by OSD Office of Force Transformation, 17 October 2003, available at the Office of Force website (http://www.oft.osd.mil).

[4] Hiltzik, Michael. Dealers of Lightning : Xerox PARC and the Dawn of the Computer Age, DIANE Publishing Company, Collingdale, PA 1999.

[5] Boehm, Barry. *Software Cost Estimating*, Prentice-Hall, 1981.

[6] Lyke, J. "Reconfigurable Systems: A Generalization of Computational Strategies for Space Systems", *IEEE 2002 Aerospace Conference*, 2002.