Communications for Integrated Modular Avionics

Richard L. Alena
Richard.L.Alena@nasa.gov
John P. Ossenfort IV, SAIC
Kenneth I. Laws, QSS
Andre Goforth
NASA Ames Research Center
Moffett Field, CA 94035
Fernando Figueroa, NASA Stennis Space Center

Abstract—The aerospace industry has been adopting avionics architectures to take advantage of advances in computer engineering. Integrated Modular Avionics (IMA), as described in ARINC 653, distributes functional modules into a robust configuration interconnected with a "virtual backplane" data communications network. Each avionics module's function is defined in software compliant with the APEX Application Program Interface. The Avionics Full-Duplex Ethernet (AFDX) network replaces the point-to-point connections used in previous distributed systems with "virtual links". This network creates a command and data path between avionics modules with the software and network defining the active virtual links over an integrated physical network. In the event of failures, the software and network can perform complex reconfigurations very quickly, resulting in a very robust system.

In this paper, suitable architectures, standards and conceptual designs for IMA computational modules and the virtual backplane are defined and analyzed for applicability to spacecraft. The AFDX network standard is examined in detail and compared with IEEE 802.3 Ethernet. A reference design for the "Ancillary Sensor Network" (ASN) is outlined based on the IEEE 1451 "Standard for a Smart Transducer Interface for Sensors and Actuators" using real-time operating systems, time deterministic AFDX and wireless LAN technology. Strategies for flight test and operational data collection related to Systems Health Management are developed, facilitating vehicle ground processing. Finally, a laboratory evaluation defines performance metrics and test protocols and summarizes the results of AFDX network tests, allowing identification of design issues and determination of ASN subsystem scalability, from a few to potentially thousands of smart and legacy sensors.[1][2]

TABLE OF CONTENTS

1. INTRODUCTION

Recent advances in software architecture and data communications can benefit modern aerospace avionics systems by increasing capability and improving reliability. In particular, network-oriented standards offer high bandwidth and flexible support for modern devices and data types. Specifically, advances in computational capability, software interfaces and time deterministic networks can improve system architecture and facilitate software development for modern aerospace avionics systems. The ARINC 653 "Avionics Application Software Standard Interface" operating system specification incorporates partitions for separating critical and non-critical functions. Its Integrated Modular Avionics (IMA) concept relies on functional isolation between operating system partitions to limit propagating failure modes within avionics software; and to simplify software validation and verification (V&V). IMA replaces point-to-point cabling with a "virtual backplane" data communications network. The network connects software-configurable computing modules that can adapt to changes in operating modes or respond to an avionics system fault. There is a potential path between any of these modules, with the software and network defining the active Virtual Links to support effective partitioning. In the event of failures, the system can quickly reconfigure its software functions (in pre-determined ways), resulting in a very robust system.

[2] IEEEAC Paper #1230, Version 1.3, Updated December 27, 2006

IMA requires robust yet flexible data communications. This paper focuses on the Airbus Avionics Full-Duplex Ethernet (AFDX) and ARINC 664 Aircraft Data Network Part 7 time-deterministic network Standards in order to determine completeness of the Standards, maturity of commercial products, acceptance and adoption by the aerospace industry, and appropriate role on-board spacecraft. AFDX uses the physical layer widely used in the computer industry as specified in the IEEE 802.3 Switched Ethernet Standard. This is a "star" topology, with the switch forming the center, routing data only to specified destinations and thereby minimizing network contention. The AFDX switch enforces timing and policies, limiting fault propagation and ensuring time determinism.

Dual-redundant AFDX networks are used for reliability. The physical medium is generally shielded twisted pair copper cables, although optical fiber can also be used. Bandwidth is 10 Mbits per second (Mbps), 100 Mbps, and even 1000 Mbps. AFDX creates *Virtual Links* (VLs), which are point-to-multipoint time-division multiplexed connections. Also, time-synchronous delivery of critical data streams can be guaranteed, with a timing jitter of at most 0.5 msec for any VL. AFDX uses the concept of *Bandwidth Allocation Gap (BAG)* to distribute the data stream into timed packets, with the BAG value determining the time interval between packets on a given VL.

Primary benefits of AFDX are a reduction of cable weight and volume, an increase in bandwidth and data volume, and greater flexibility in software design and V&V. AFDX complements ARINC 653 software architectures, serving as the virtual backplane. AFDX provides compatibility with network-oriented data communication formats. AFDX can support additional sensors for flight test and monitoring capability upgrades. The Ancillary Sensor Network is proposed for such use, to support up to 10,000 sensors at high data rates and resolutions, supporting smart sensors compliant with the IEEE 1451 Standard, simplifying calibration and maintenance.

The Ancillary Sensor Network (ASN) was prototyped using up to four end stations. A total of 120 VLs were used to move large amounts of data in a time-deterministic manner suitable for sensor-data acquisition. Packet-to-packet "bandwidth allocation gap" (BAG) timing was measured for every packet generated by the traffic generator according to recipes that simulated various target functions. Average packet-to-packet timing for each VL was determined, along with standard deviation and minimum and maximum packet-to-packet timing intervals. These were compared to the AFDX jitter specification of 0.5 msec maximum. Lost, missing, or delayed packets were detected and reported.

AFDX is a key technology for installing time-deterministic networks aboard aerospace vehicles. A survey of aerospace use showed that Airbus and Boeing are both adopting AFDX and ARINC 653 for flight-critical roles aboard the next generation of commercial aircraft. Although space-qualified parts are not currently available, use of AFDX in certain roles aboard spacecraft can simplify cabling, leading to a significant reduction in cable weight and volume. Other significant advantages include simplified software design and network management, ease of reconfiguration and extension, and compliance with industry network standards.

The team from Ames Research Center has significant experience in spacecraft data systems and communications, providing key analysis for the Space Station Freedom Command and Data Handling System. In 1995, the Wireless Network Experiment aboard Shuttle and Mir led to the adoption of wireless network technology for the International Space Station (ISS). Subsequently, the team developed the Databus Analysis Tool, a MIL-STD 1553B data-bus monitor space-qualified during STS 85 and used for ISS Control Momentum Gyro checkout during STS 92.

2. INTEGRATED MODULAR AVIONICS

The aerospace industry has been moving from "distributed" and "federated" avionics architectures to "modular" architectures, and is adopting other advances in computer engineering such as networks and Internet Protocol (IP). These advances promise to increase the performance and reliability of avionics systems while simplifying the development and certification of flight software and avionics hardware.

The International Space Station (ISS) exemplifies a distributed architecture. The ISS avionics system is built with a large number of relatively small Orbital Replaceable Units (ORUs)—identical in concept to aircraft Line-Replaceable Units (LRUs)—interconnected with multiple MIL-STD 1553B data buses.[1] The units typically have very specific functions that do not change over time (although a unit's operating mode within a redundant configuration may change). An example of federated architecture is the Boeing 767 aircraft, which has dedicated LRUs responsible for specific functions such as engine control interconnected by ARINC 429 data buses.

Such architectures have several drawbacks. The multiplicity of specific modules requires a large and complex development effort, plus the maintenance of a similar multiplicity of spare parts. Each component on a synchronous data bus must interoperate with all the other components, so software changes in one may require corresponding changes in the others. Experience has shown such systems to be slow as well as difficult to develop and to upgrade. On the plus side, the architecture can be fairly robust. Each module can be designed to function independently despite failures at other points in the system.

The Boeing 777 Aircraft Information Management System (AIMS) exemplifies integrated avionics, with design starting in the mid 1990s. Most of the computation takes

place in large cabinets of processors. Data communication is via point-to-point data buses such as ARINC 429 or ARINC 629, augmented by fiber optic cables for display and maintenance interfaces.[2] LRUs and data communication cables are duplicated for reliability. The processing boards have dedicated functions (again in redundant configurations), but may aggregate functions formerly distributed among several LRUs in the older designs. This saves weight and power, and may simplify software development and integration. However, this architecture requires long cable runs for interconnecting distant LRUs that increase weight and may introduce reliability issues.

Integrated Modular Avionics (IMA) replaces the point-to-point cabling with a "virtual backplane" data communications network.[3] The network connects software-configurable LRUs that can adapt to changes in network functioning or operating modes. There is a potential path between any of the LRUs, with the software and network defining the active Virtual Links in real-time. In the event of failures, the system can quickly reconfigure, resulting in a very robust system. The ARINC 653 Standard "Avionics Application Software Standard Interface" describes an application program interface and operating system for producing a flight-critical avionics system that partitions critical and non-critical functions so that they cannot interfere with each other.[4] Not only does this simplify software design and implementation, it allows more flexibility in software V&V. It is the result of years of need by airframe manufacturers and their suppliers to have a practical way to build avionics systems that may be certified for human-rated or safety-critical applications within a reasonable cost, and to allow upgrades to their equipment so as to stay competitive and profitable. The first part of the standard refers to a specific Application Program Interface (API) called APEX, implemented in real-time operating systems (RTOSs) used for avionics control systems. The second part extends the APEX API by specifying the file system, port data structures, and scheduler, while the third part deals with APEX test compliance. XML configuration files are used to define the system hardware and software configuration.

One of the benefits of this ARINC 653 Standard is that once flight certification is achieved, it is possible to add software such as health monitoring or mission support functions within new ARINC 653 partitions that may not be certified to the same level as flight critical functions. This provides an improved level of flexibility and ease of V&V to avionics software development. Currently, certain software changes in a platform running flight or safety-critical software can produce unintended fault modes due to ineffective isolation between software components. Proper adherence to ARINC 653 architecture and design guidelines helps guarantee this isolation, thereby allowing certification by component. Certain effects, such as software execution timing changes, can still propagate faults into isolated partitions, but methods such as communication-link partitioning can help limit these effects. This software standard and its associated certification processes allows each component to be certified to the level of criticality associated with that component, producing major cost and schedule savings.

ARINC 653 supports Avionics Full Duplex Ethernet (AFDX) by means of Queuing and Sampling (Q&S) Ports. The APEX API defines the calls to such ports, and the use of an AFDX data communications system nicely complements the logical constructs for Q&S ports. The concept of a Queuing Port is similar to a First In-First Out (FIFO) buffer, with transmit and receive timing similar to TCP/IP streams. With AFDX providing bounded-time latency, the delay in stream reception is also bounded. However, the FIFO buffer, whose size is set as a parameter in the AFDX configuration, will produce some variable delay as it is filled or emptied at different rates. See Figure 1 for a graphical description of Q&S ports. A sampling port, by contrast, overwrites the previous value in a single packet buffer and updates a "freshness" parameter with a timestamp indicating when the data was written. This buffer can then be read by the receiving application either once or multiple times. This mode is very useful for reading sensor data where the exact time of the sensor value is of critical importance. Most ports used in ARINC 653 flight implementations are sampling ports.

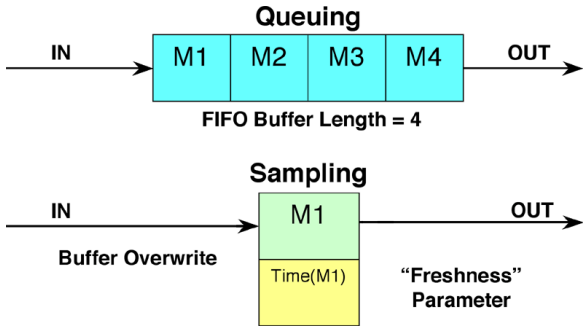


Figure 1. Queuing and Sampling Ports

3. NETWORKS FOR AEROSPACE

Flight-critical avionics systems maintain flight-control loops by reading critical sensors such as inertial measurement units, computing current deviations from desired trajectory, and outputting needed corrections to flight-control surfaces and thrust-vector actuators. This must be done at a certain consistent rate to maintain full control of the flight vehicle. Missing corrections or erroneous corrections to control elements can result in loss of flight control, and in catastrophic hazards such as vehicle airframe overstress and breakup. Typical ascent-vehicle flight-control loops run at the 10 msec rate. Support functions can run at lower rates, resulting in a collection of "rate groups" usually at time frames of 10 msec, 100 msec and 1000 msec. Therefore, a primary measure of data-communication utility for flight control is the ability to maintain rate-monotonic scheduling for all messages. Alternative figures of merit are total message bandwidth, message error rates, and fault containment for propagating fault modes.

Two communication standards—ARINC 429 and MIL-STD 1553B—have dominated commercial and military aviation. ARINC 429 connects LRUs aboard the Boeing 737 and other civilian aircraft using distributed avionics architectures. The MIL-STD 1553B is used in most military aircraft for flight-critical control and control of various mission systems. Both are half-duplex communication standards. ARINC 429 connects LRUs via a point-to-point cabling scheme; MIL-STD 1553B connects multiple devices via a common bus. Both are currently used in production aircraft and spacecraft, but deficiencies in performance for modern aircraft has led to adoption of extended and modified versions of these standards.

Data communications standards usually address specific layers of communication, as formalized in the Open Systems Interconnection Basic Reference Model. We will use a simplified version of this seven-layer stack, reducing it to four layers. At the bottom is the physical layer (PHY), which is how the data is represented in physical reality: as electrical signals or optical impulses, the type of modulation and interconnection, and any other characteristics specific to signaling such as data rate, clocking scheme, etc.

Next is the Media Access Control (MAC) layer—also called the data link layer—specifying the logical mechanisms for transmitter and receiver control, access to the medium (using either time-division, frequency-division, or code-division multiplexing), byte and word interpretation, and any error-checking or correction functions.

The next higher layer is the protocol (PROT) layer, which specifies the logical construct of "messages," message sequencing, and acknowledgement. The fourth layer is the Application Layer, where specific software functions running on the computational platforms can define their own methods for assuring data integrity and managing communication assets and redundancy.

Ethernet 802.3 MAC allows each transceiver on the network to initiate message transfer on its own. This is the key characteristic separating networks from data buses: networks generally provide peer-to-peer asynchronous access rather than master-initiated access and message transfer. Ethernet is an asynchronous MAC with each transceiver truly independent and the network using Carrier Sense Multiple-Access/Collision Detect (CSMA/CD) for arbitrating access. Therefore, in cases of high network load (with many transceivers trying to send messages at the same time), the probability of collision is high and the latency for a given message to be transferred can be unbounded. That is, it may take a long time for a given transceiver to be able to access the physical layer to send its message, due to constant interference or collisions with messages being sent by other transceivers. This unbounded message latency makes Ethernet unusable for time-critical control loops. Such a network is termed an *802.3-compliant* network.

As the Ethernet standard evolved, it moved from a common bus-type cable fabric (Thick and Thin Ethernet) to a star topology with a hub in the center. Each transceiver was connected to an independent hub port and now all collisions and interference could be mediated in the hub. However, this simply moved the cause of the media contention from the cable to the hub and did not solve the unbounded message-latency problem.

The emergence of the switching hub—which buffered and directed each message to the correct port based on its MAC address—resulted in significant reduction of media contention and much better network performance. A lightly loaded switched Ethernet does in fact produce predictable latency, provided that each transceiver exercises restraint in its use of the network media. This results in a *profiled* network.

This concept of limiting each transceiver's media access is the central one behind time-deterministic Ethernet. AFDX times packet and message initiation for traffic shaping, limiting when and at what rate any transceiver transmits on the AFDX network. This produces bounded latency for message delivery, and produces a *time-deterministic* network. [5] AFDX is really a standardized method for creating a good profiled network.

Ethernet emerged as the first high-performance local area network (LAN) standard for microcomputers in the 1980s. It was developed by DEC, Intel, and Xerox, replacing point-to-point serial links and telephone modems. The IEEE 802 family of standards describes the Ethernet PHY and MAC layers and their relationship to Internet Protocol (IP). IEEE 802.1 describes overall architecture, bridging, and network-management functions. IEEE 802.2 describes logical link control for Ethernet, while 802.3 through 802.17 define variations of PHY and MAC-layer technology such as switched Ethernet and wireless Ethernet.

IEEE 802.3, "Carrier Sense Multi Access/Collision Detect (CSMA/CD) Access Method and Physical Layer Specification," is the most relevant standard, being the primary one for switched Ethernet and AFDX.[6] The original 802 specifications defined shared-medium, peer-to-peer data packet communication methods for multiple end stations, with broadcast modes for network management. It allowed LAN subnets to be bridged together into much larger networks, including large hierarchical networks.

The Ethernet PHY layer is a differential 10-volt, Manchester-encoded, self-clocking signal carried over a transmission line. Coaxial cable or shielded twisted pairs are commonly used for cable segments of up to 1 Km in length. Conventional 802.3 Ethernet uses the CSMA/CD access method: an end station listens for network traffic and transmits when the network is clear, but then backs off and retransmits in the event of a collision (when two end stations transmit simultaneously).

An Ethernet 802.3 frame is a data structure for containing a variable-length data field within a packet. Up to 1518 bytes can be contained in one frame (with contents defined at the MAC layer). The MAC layer also delimits the frame, addresses the packets (48 bit), transfers data and detects errors, performing Logical Link Control (LLC). Basic error rates specified in the standards are important, being 8×10^{-8} per byte for wired end stations, with undetected errors specified at a 5×10^{-14} rate per byte of data.

The use of a star topology and a central switch—which detects the destination address and sends the packet to only that port—can significantly decrease the chances for data collision, resulting in best performance with minimal packet latency. Note however, that IEEE 802.3 is not time-deterministic. Under certain circumstances, such as high network load, successful packet transmission cannot be guaranteed within a given time period (packet latency). A packet can be delayed indefinitely by repeated collisions within the network. In fact, there is an uncovered failure mode affecting the entire network: when an end station starts jabbering uncontrollably and therefore saturates the network, preventing other needed data from being transferred. For this reason, Ethernet has not been considered for real-time, safety-critical use.

However, the widespread use of Ethernet, its performance, and its peer-to-peer structure are attractive for use in the aerospace industry. There have been several attempts to define Ethernet for use aboard vehicles, such as the ARINC 664 Standard, "Aircraft Data Network," which is comprised of eight parts.[7] This standard was developed to enable multiple networks optimized for different purposes to co-exist aboard a vehicle, interface well with Airline Operational Networks, be compliant with a large range of commercial standards and implementations, and standardize Conformance testing at the system and subsystem level.

Part 1 of the standard is "Systems Concepts and Overview," followed by Ethernet and Internet Protocol specifications in Parts 2 to 6. This ARINC standard is an implementation of the IEEE 802 Standards adapted for aircraft, containing much of the same material but in greater detail and specific to aircraft technology. ARINC 664 Part 7, "Avionics Full-Duplex Ethernet (AFDX) Network," specifies a certain time-deterministic method applicable to real-time, safety-critical command and control.[8]

Airbus commissioned the study and development of network-based data communication standards for commercial aviation, resulting in the Airbus Standard for AFDX. [9] Subsequently, ARINC developed the 664 Part 7 standard around similar concepts for Boeing and other US companies. There are minor differences between the two. IEEE 802.3 interface methods allow the use of standard Ethernet hardware for development by implementing a compliant PHY layer. However, AFDX modifies the MAC layer extensively, resulting in AFDX frames that look similar to Ethernet frames but with certain fields containing different parameters. See Figure 2 for a definition of an AFDX frame.

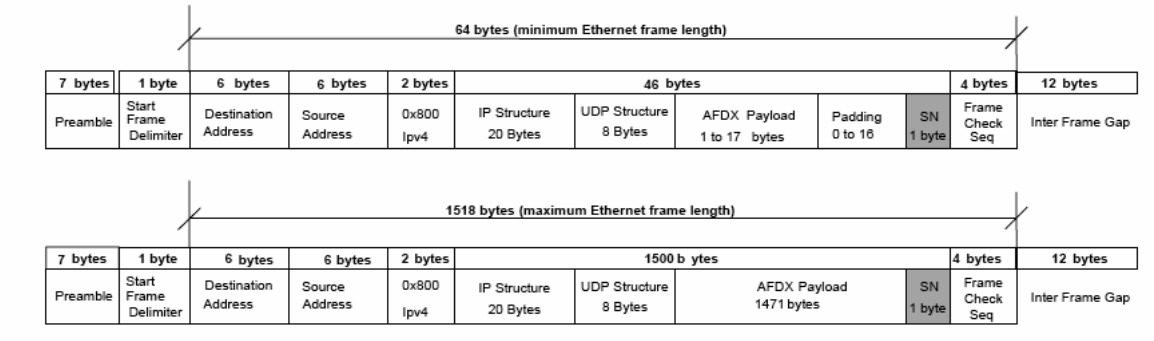


Figure 2. AFDX Frame Definition

AFDX places certain requirements on the 802.3 PHY interface: the interface must transmit even with the cable disconnected, and must support the full 100-Mbps rate in full-duplex mode. This is to prevent transient cable disconnects from backing data up through the network, and to support the full data rate. Ethernet can be used without a protocol in a raw-packet transfer mode, but data-transfer reliability requires a protocol and AFDX specifies the use of either Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) in all cases.

Key properties of an AFDX network are Virtual Links (VL), Bandwidth Allocation Gap (BAG), Maximum Frame Size (LMax), Sequence Number (SN), data Integrity Management (IM), and Redundancy Management (RM). The Virtual Link (VL) provides a logical-addressing structure within the network, simulating a virtual set of ARINC 429 point-to-multipoint data links. That is, each source and multiple receivers create a permanently defined virtual circuit with guaranteed delivery of data, even though the physical and logical transport is over a star-topology, peer-to-peer network fabric.

The VL concept is central to AFDX in that each VL has a specific timing and bandwidth allocation on the network. That is, timing and traffic shaping is done on a per-VL basis. The VL is specified in both the MAC address of the end station and in the specific IP addressing at the protocol level. One of the first major differences a user observes on an AFDX network—as compared to an Ethernet network—is that a given end station has multiple MAC addresses (each corresponding to a different VL on the interface) as well as multiple IP addresses (also designating different VL numbers) on the same end-station port. In contrast, most Ethernet ports have fixed MAC and IP addresses for each interface port.

The AFDX traffic-shaping function regulates the timing of packet generation on the network for each VL and also the portion of total network bandwidth used by the given VL. The key mechanism is the Bandwidth Allocation Gap (BAG), which is the specified time interval between successive packets (or frames) for a given VL. That is, instead of simply transmitting successive packets at the maximum rate possible (like Ethernet), an AFDX end station will regulate its transmission of packets to one for each specified BAG interval, thereby providing traffic at a constant and deterministic rate. Aggregations of carefully timed and bandwidth-limited transmitting end stations should ensure that the entire AFDX produces and preserves time-determinism.

The allowed AFDX BAG values are 1, 2, 4, 8, 16, 32, 64, and 128 milliseconds (msec). In order to properly allocate bandwidth, the different BAGs are given maximum data-payload sizes that are proportional to the BAG rates. That is, the 1 msec BAG is typically allowed only 64 bytes per frame of data, while the 128 msec group is allowed the maximum 1518 bytes per frame. A network traffic timeline for several VLs with differing BAGs is shown in Figure 3.

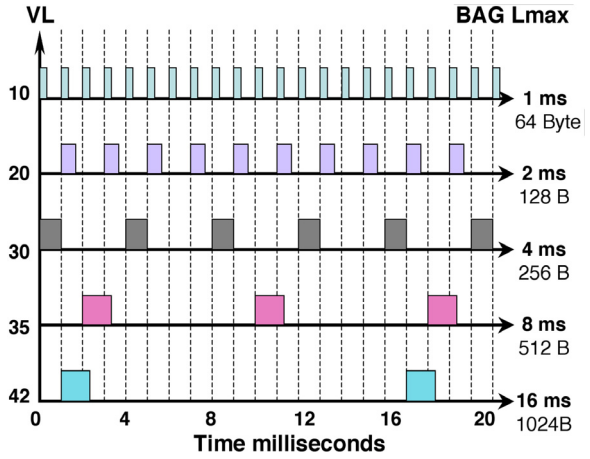


Figure 3. Virtual-Link Bandwidth Allocation Gap

By definition, a given VL is sourced by only one transmitting end station but can have multiple receivers. This makes the virtual topology of AFDX closely resemble ARINC 429 and MIL-STD 1553B physical topologies. In fact, there is a formal mapping of ARINC 429 to AFDX and a similar mapping can be done for MIL-STD 1553B and similar protocols.[10] This simplifies porting of legacy software code to AFDX environments and can benefit flight-certification efforts. Any AFDX end-station port interface can source many VLs and can also receive other VLs, as needed by the application. Note how the many

physical cables needed for ARINC 429 are replaced by a single network cable plant containing the same circuits, but now implemented in time-division multiplexed VLs—saving much weight, volume, and wiring complexity.

The receiver maintains data integrity by checking that each packet arrives within the specified time interval (BAG) for a given VL and its SN matches the next expected value. Frames for a given VL must be transmitted and received in order, for the SN to stay sequential and out-of-order packets are rejected. Furthermore, the receiver checks data integrity using cyclical redundancy check (CRC) error detection. The implication is that packet retries and acknowledgements are not the primary mechanism for ensuring data integrity. Rather, very low packet loss—with data integrity and timing determined by SN and CRC methods—is needed for time determinism. Errors are rejected rather than corrected because timing conformance is as important as data integrity in real-time control systems, an inherent design tradeoff.

Other data integrity methods can be instituted at the application layer, but the AFDX network will provide a sequential stream of data at the defined time intervals with a very high degree of error detection. Network errors look like missing data for AFDX, but look like delayed data for Ethernet. The Integrity Management (IM) parameter determines whether the receiver uses the SN or ignores SN errors. One might notice that data integrity and temporal integrity are a design tradeoff in data communications; one can improve data integrity by acknowledgement or redundancy, but at the expense of complicating the timing (and vice versa).

The protocol (PROT) layer is very important in maintaining data integrity. ARINC 664 Part 7 also defines the AFDX network protocol layers based on TCP/IP and User Datagram Protocol (UDP). Ethernet simply forms and transmits packets, but leaves the data accounting, and therefore data integrity to the PROT layer, usually implemented as TCP/IP for most networks to be compatible with the Internet.

TCP/IP requires acknowledgement packets to be sent by the receiving station for packets successfully received. TCP/IP also tracks packet sequence and retransmits packets lost in the network, allowing out-of-order transmission and receipt of packets as necessary. All of this acknowledgement and retransmission takes time and bandwidth, resulting in temporal non-determinism for Ethernet and TCP/IP networks. The TCP/IP protocol also arranges packets in the correct order prior to passing them to the application layer. Therefore, a missing packet can interject a significant delay, since all data is delayed until successful receipt of a retransmitted packet. On a heavily loaded network, packet loss can be significant and packet delay and retransmission can produce significant and highly variable latency. Furthermore, the use of TCP/IP "windows" where a single acknowledgement suffices to cover a number of transmitted packets (usually contained in a specified buffer size) can also disrupt temporal behavior.

AFDX solves the potential problems by using UDP, which is similar to broadcasting packets. Although ARINC 664 Part 7 specifies either TCP/IP or UDP, in practice only UDP is used. UDP requires no protocol acknowledgement and therefore limits network load by eliminating acknowledgement packets, meshing nicely with SN and BAG concepts. AFDX thus relies on the MAC frame delivery mechanism, using BAG to limit network bandwidth and SN to track data integrity.

Packet loss can be a significant performance issue for AFDX. The Study Team investigated mechanisms for packet loss, since this would be a major problem for critical applications. By its nature, Ethernet 802.3 is asynchronous, with each end station able to transmit at any time. There is no global time synchronization mechanism. Ethernet 802.3 relies on the switch to prevent significant packet loss. However, Ethernet 802.3 allows packet loss. Any given switch has limited buffering capability, and in the event of overload will drop packets. The use of TCP/IP prevents packet loss from becoming data loss by retransmitting the lost packets. UDP, by contrast, simply allows dropped packets to become data loss and never retransmits missing packets. AFDX using UDP has a similar policy; missing packets translate to missing data and therefore missing sensor values or missing control parameters when applied to a flight-critical avionics system.

The primary mechanism for packet loss is collisions. In an Ethernet 802.3 network, this loss is in the switch. Since each end station-to-switch connection is full-duplex, packets cannot be lost on that network segment. For most avionics use, network utilization is so far below capacity that overload in end station-to-switch segments is unlikely. Recall that AFDX purposely limits network utilization by staggering all packets by BAG. However, an end station with many VLs can experience a coincidence of all VLs, therefore resulting in many frames being transmitted or received concurrently. For example, in Figure 3 above, VL 10 will interfere with VL 20 every other frame and with VL 30 every fourth frame. If all these VLs originate from the same end station, the end station scheduler can stagger the frame transmission and avoid contention.

While the timing for a single VL can be maintained easily, when many end stations are used to form a real AFDX network the AFDX switch is responsible for solving contention caused by collisions. In this case, the collisions are between simultaneous packets issued from multiple, asynchronous end stations. The network switch will buffer the simultaneous packets and send them to the destinations after a brief queuing delay, thus introducing time jitter into the AFDX network. The maximum timing jitter is defined and enforced for each BAG rate for every VL in the network, to maintain maximum bounded latency for all network traffic. The maximum jitter is equal to half the minimum BAG, or at most 0.5 msec. If the switch or end station cannot handle many concurrent VLs due to buffer limitations, there will be packet loss.

The AFDX Switch is a critical component of an AFDX network, providing the MAC-layer routing function from one end station to another. It is also responsible for AFDX policy enforcement for addressing and timing. Most Ethernet switches are learning types that determine a destination MAC address—of the end station connected to each switch port—using the Logical Link Control Discovery Protocol. Once the first Ethernet packet is successfully sent through the switch port, that MAC addresses is stored in the switch address buffer. This process implies that the first time a packet is routed through the switch, the discovery process and switch learning time will delay that packet for an indeterminate interval. This cannot be tolerated in a time deterministic network, so the AFDX switch contains a static address-switching table configured during initialization and remaining constant during operation. This means that the network cannot be physically reconfigured during operation, unlike Ethernet. However, reconfiguration can be easily achieved by updating the switch table explicitly.

The AFDX switch validates addresses and routes data to the correct switch port using the static configuration table. It also enforces BAG timing by monitoring the bandwidth utilization and packet timing of a specific VL. If a specific VL is malfunctioning by jabbering and jamming the network, the switch will block those faulty packets from going any further. The AFDX switch is therefore a key element in containing fault propagation in an AFDX network, by detecting and isolating babbling end stations. Of course, there would be a total loss of the VLs involved in such policy violations. Therefore, data integrity is maintained at the PHY layer using cyclical redundancy check (CRC) for each frame (providing excellent error detection) and Sequence Numbers (SN) for VLs to track data packets.

AFDX also defines the use of dual-redundant networks. Redundancy management is performed on a VL basis, using the Redundancy Management (RM) parameter. If set, both AFDX ports transmit the same packet at the same time and both networks provide the same packets to two independent receiver ports. The algorithm for accepting data is to accept the first received packet. In the case of unequal transit times through the switch, the first packet wins—which favors better time determinism by reducing jitter. By specifying redundancy on a per VL basis, other non-time-critical data could be transmitted on either port, useful for single avionics modules that are only in close proximity to one network's physical route through the vehicle.

4. MISSION COMPUTER DESIGN

The "Mission Computer" is a concept for an on-board general-purpose computer for spacecraft mission support functions. The use of an ARINC 653-compliant real-time operating system (RTOS) would be ideal for the Mission Computer (MC), allowing multiple applications to have access to all flight vehicle data and state information. This data exchange would be one way: the Mission Computer can read the vehicle parameter information but cannot alter it. The high-priority partition would be used to run the real time tasks—such as reading the AFDX network, buffering, and writing the entire data stream to the Flight Data Recorder (FDR) and for performing telemetry and communication-link protocol formatting.

The central concept is to separate the flight-critical functions from mission-critical functions by running them on different platforms. The MC would be able to host application programs needed for certain specific phases of flight. The use of ARINC 653 would allow transient applications, to be loaded only when needed. Allowing dynamic use of programs enables very effective use of the Mission Computer functional capability. Of course, each of these "dynamic" programs would be fully certified as appropriate for its function prior to running it aboard spacecraft. The use of ARINC 653 allows each program to be validated and verified independently since the RTOS provides adequate partitioning. The AFDX network ensures that data communications can be conducted in a time-deterministic manner. Compliance with the ARINC 653 guidelines on functional separation and AFDX network usage further supports robust partitioning.

A reference software architecture for the Mission Computer can be based on ARINC 653, taking advantage of fault containment provided by the partitions. The tau 0, time-priority partition runs the main data-handling function, which receives flight-data parameters, processes them, and stores them in the FDR, built as a large array of flash memory. In addition, it selects parameters for downlink, manages the multiple parallel communication resources, and provides the main interface to space-based network communications. It can host ancillary data acquisition functions, storing these parameters in the FDR and even down linking them upon request. It could act as a command arbitration interface, determining which specific communication link had priority—important for supporting

concurrent legacy, network-compliant and proximity communication links. It could host machine-vision applications for automated rendezvous and docking and even Vehicle System Management (VSM) functions, exclusive of critical failover.

A classic example of advanced VSM function would be the reconfiguration of a critical system after a failover. The automatic failover may be suboptimal, therefore, a subsequent reconfiguration could optimize against subsequent additional failures. It could also optimize resource use such as power, acting as an on-board overall power manager. Such functions could be performed on the ground, as in current practice, or could be migrated to the MC later to provide more autonomy for future missions. The flight critical computer (FCC) would reject any inappropriate input from the Mission Computer, providing the final safety check (as appropriate for flight control functions).

ARINC 653 defines queuing and sampling ports in the APEX API, and AFDX extends this concept to time-deterministic network operation. Queuing ports are very similar to TCP/IP streaming sockets, except that AFDX allows definition of a fixed queuing first-in-first-out (FIFO) buffer size, preventing most buffer overruns and underflows. Generally, the queuing FIFO size is specified as the number (not bytes) of AFDX packets in the FIFO, also leading to more control over the potential length of time delays caused by the FIFO buffer backing up. Proper setting of internal RTOS and AFDX parameters can ensure that the queuing FIFO buffers will not experience problems.

The ARINC 653 sampling ports have no buffer at all, but do indicate parameter freshness through the use of a timestamp. In this case, a read or write of a sampling port is the equivalent of reading a sensor connected directly to the computer: each read will read the last sensor value communicated on the network. Multiple reads will read the same parameter, but can use the freshness parameter to identify stale data. The design of AFDX, requiring predefinition of VLs, sampling and queuing ports, and other runtime parameters can eliminate many of the buffer-management issues involved in Ethernet networks.

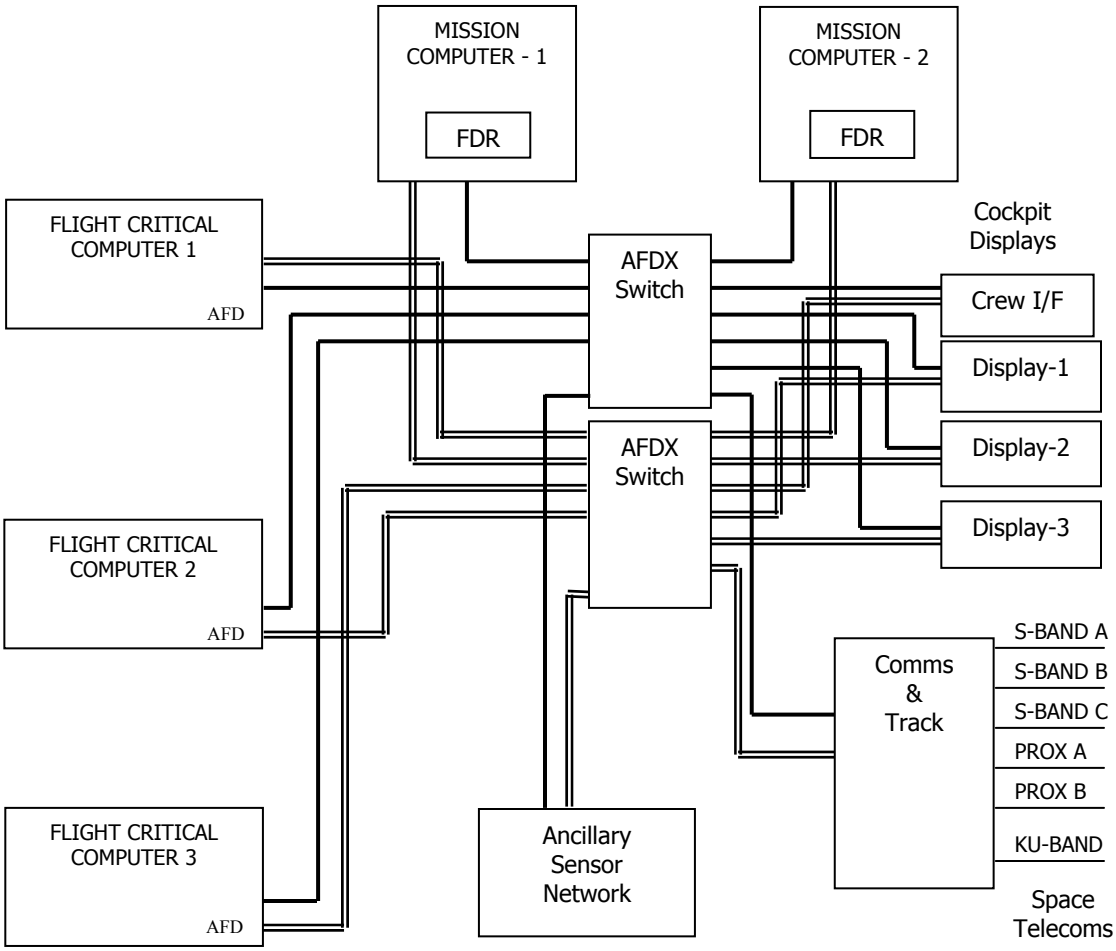


Figure 4. Mission Computer Network

Ancillary Sensor Network Design

A simple AFDX network design example will help identify the relevant issues for AFDX configuration and design. The detailed design of the Ancillary Sensor Network (ASN) highlights the advantages of a network approach for adding sensors to a spacecraft for flight test or vehicle health management. This can help meet the development flight instrumentation need for flight tests. Many times during a vehicle's operational lifetime, it may be desirable to add some new sensing capability to an existing design. For example, the Orbiter has the "Orbiter Instrumentation Bus" used for flight-test and ancillary data collection. Subsequent to the loss of Columbia, a wireless wing-leading-edge impact-detection system was installed on the Orbiter to detect any foam impact during ascent or any micrometeorite impacts during orbit.

We will assume that the Ancillary Sensor Network needs to be single-fault tolerant and provides time-synchronous delivery of sensor data to the Mission Computer. In compliance with standard data communications practice, one network runs down the right side of the vehicle and one network runs down the left side of the vehicle. The goal is to allow new "smart sensors" to be added to the ASN in a plug-and-play fashion with minimal reconfiguration required. Smart sensors can be new sensor modules or "virtual sensors" which use legacy sensors with additional processing in software providing ease of configuration, identification and data validation. The smart-sensor modules include network-connected interfaces to multiple sensors adhering to IEEE 1451.1 standards for data, information, and knowledge management across networks [11]; transducer electronic data sheets (TEDS)—IEEE 1451.2—and health-related information as specified in Health Electronic Data Sheets (HEDS). [12] Use of AFDX sampling ports rather than Ethernet will allow setting up different time-synchronous rate groups for sampling the sensor data, with time stamping of each data point.

The use of smart sensors (virtual-using legacy sensors, or physical-with embedded processing) provides a number of benefits. Smart sensors provide configuration data and information via TEDS and HEDS, which describe details regarding sensor operational state, calibration status, serial number and other meta data relevant to validating sensor values in critical systems. [13] An analogous approach could define component electronic data sheets (CEDS), to include (virtual) smart components such as valves, tanks, etc.; also communicating data and information on the network bus. [14] Smart sensors can use unified data interfaces such as AFDX, with greatly simplified connections, emulating plug-and-play functions. This makes adding sensors for flight test or operational health assessment far simpler than older methods. Software can use TEDS and HEDS templates to quickly add a new sensor to a data acquisition loop, for example. Finally, advances in fabrication techniques are possible using a number of microscopic sensing elements, with the sensed value representing an average of the valid sensing elements. This has the potential of significantly improving calibration and sensor operation by providing redundancy and computational functions at the sensor module level. A bad or inaccurate sensing element could be voted out of a valid configuration, increasing robustness and increasing calibration intervals. [15]

The overall ASN design is shown in Figure 6 consisting of a dedicated pair of AFDX switches for ASN use that connect to the main AFDX Mission Computer switches. This example provides four smart-sensor ports for the spacecraft. AFDX will support mixes of single and dual-redundant network traffic.

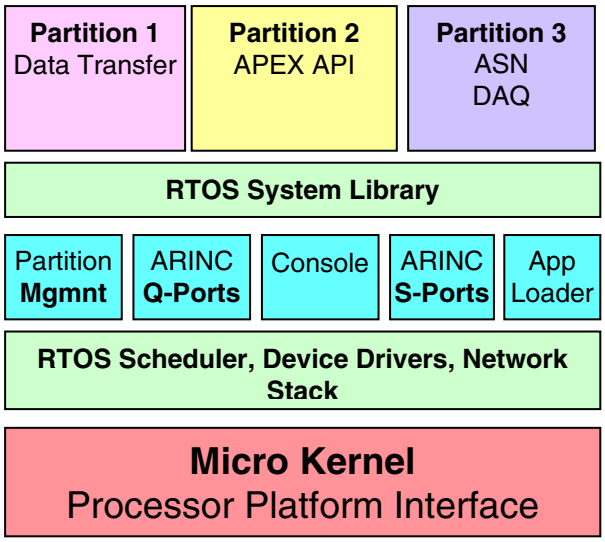


Figure 5. ASN Software Architecture

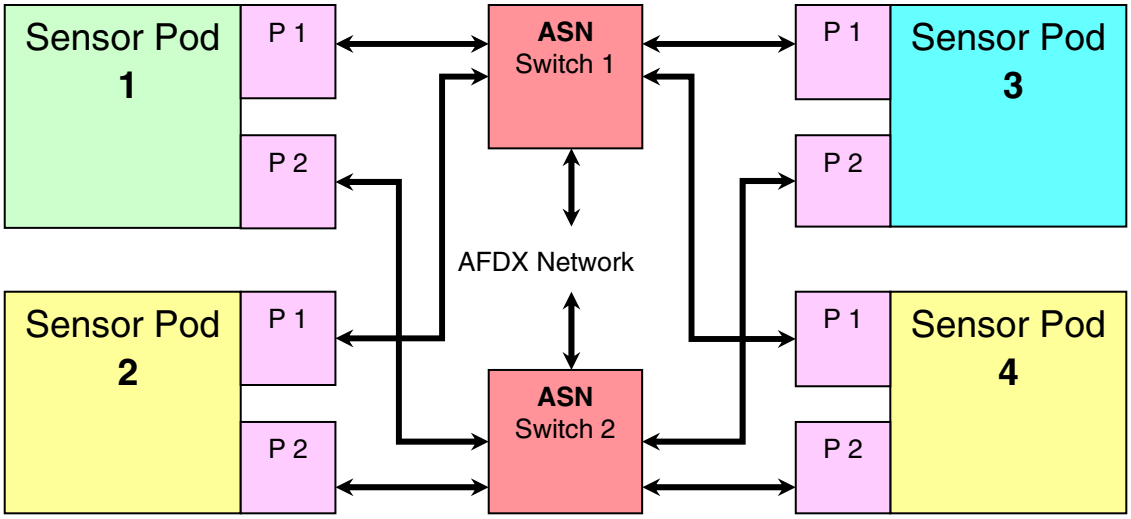


Figure 6. Ancillary Sensor Network Diagram

In software, the ASN network manager runs in a separate ARINC 653 partition. The applications that might use the data are run in another partition and communicate with each other via shared memory. The ASN manager performs the functions of polling the ASN network, discovering and reading the data from the various modules. Each module is assigned a certain set of 1-msec, 8-msec, 32-msec, and 128-msec rate groups.

The AFDX network is designed and scheduled to accommodate a full set of potential modules. In a real spacecraft, most of these "slots" would not be used. If that module or data slot were missing, no packets would flow; the manager would place nulls in the corresponding shared memory array. The array would be completely refreshed each major cycle.

AFDX internal sampling ports would be an alternative method of communicating data from the ASN manager to the sensor application programs. The manager would also put the parameters into the telemetry stream if desired, and also write them to the FDR. Figure 5 shows a representative architecture for the ASN using the ARINC 653 software standard running in the Mission Computer. The only new software module would be the ASN data acquisition (DAQ) module, running in a separate partition to guard against software coding errors (or runtime errors related to the network software) in this module from affecting functions like FDR or telemetry.

The table below shows the number of sensor-pod modules, data channels, VLs, and other parameters for this design example. Our Lab Evaluation will attempt to duplicate this ASN design for validation. We define just four sensor pods, each containing eight channels per rate group, for a total of 32 channels per pod. This gives a total of 128 channels, 32 at each rate group. This is a small sensor system. Maximum bandwidth and other system parameters are summarized at the bottom. This example uses 30% of the total available bandwidth on a 100-Mbps network. There is much capacity left, and the design could scale up a factor of two—resulting in an ASN that would support up to eight sensor pods, each hosting multiple sensors at high rate and resolution.

Table 1. Ancillary Sensor Network Configuration

ASN Channels	Sensor Rate Groups (BAG value)			
Pod ID	1 msec	8 msec	32 msec	128 msec
Pod 1 (VL#)	0-7	8-15	16-23	24-31
Pod 2 (VL#)	32-39	40-47	48-55	56-63
Pod 3 (VL#)	64-71	72-79	80-87	88-95
Pod 4 (VL#)	96-103	104-111	112-119	120-127
Data Size (bytes)	64	128	256	512
Time/channel (µsec)	0.84	1.48	2.76	5.32
Total Time (µsec)	26.88	47.36	88.32	170.24
Bandwidth %	21.504	4.736	2.208	1.064

Each sensor pod can host many individual smart sensors or conventional sensors, given the rather large message size available for data transfer. Even the smallest packet size (64 bytes) associated with the 1 msec rate group would support up to 16 sensors for each channel providing 32 bits of data (Integer32) at each sample point. Each Pod supports 8 channels per rate group. Using this assumption, this ASN design would support a total of 7680 sensors.

Certain sensor pods could actually be sensor networks or webs in practice, resulting in a scalable hierarchical system. This approach would allow a given sensor pod to be implemented as a wireless network access point or bridge,

now capable of acquiring up to 1920 sensors at any one of the four rate groups. Such a cluster architecture has significant advantages in terms of connection routing efficiency, functional redundancy and fault containment.

The hardware and RTOS platform needs to be validated by extensive testing of the hardware for environmental tolerance, particularly to space radiation effects such as single event upset (SEU) and total dose effects.

Any flight use of AFDX will require a full validation of the AFDX hardware interface and the associated software. This is best done in the context of the platform-validation effort with the data-communication hardware and software being major platform elements tested. Test protocols from this paper's Laboratory Evaluation can be used as the basis for AFDX flight certification.

The adoption of AFDX for high-speed data handling for telemetry, communications, display-data transfer, and flight data-recording functions fills an important need for more bandwidth, time determinism, and flexibility in spacecraft data communications. The use of an AFDX Ancillary Sensor Network (ASN) provides an opportunity for adoption of AFDX with virtually zero risk—only the non-critical ASN would be affected in case of technology performance shortfalls. Concurrent adoption of smart-sensor technology (IEEE 1451) can significantly improve the types of sensors available and their reliability and cost. The AFDX would provide flexible capability to add sensors for vehicle health monitoring that would provide sufficient bandwidth, ease of interface and component design, and considerable simplification of software design, implementation and V&V.

5. LAB EVALUATION

The AFDX ASN laboratory evaluation measured the performance of commercial AFDX products to determine conformance to the Airbus AFDX and ARINC 664 Standards using a representative ASN configuration. The evaluation focused on time determinism and network throughput in the packet domain. Statistical analysis was used to quantify the capability of ADFX to provide time-deterministic messaging during testing over a broad range of operating configurations. The network was deliberately overloaded and the resultant error modes observed.

A key requirement for the evaluation was the ability to generate a rich set of test vectors containing different network packets with various BAG time rates, packet data sizes, and number of VLs. A second key requirement was the ability to capture and analyze the packet traffic of a fully loaded network at the full network speed of 100 Mbps. Setting up the evaluation protocols to conform to current aerospace practice has required consultation with experts on AFDX. They have indicated that all AFDX implementations use UDP as the chosen protocol, given that it can provide better time determinism by not requiring acknowledgement packets. Therefore, all testing was performed using the UDP Protocol.

The laboratory evaluation supported multiple software tools for generating AFDX network traffic, including the ability to inject specific errors. The AFDX monitor and associated software produced packet-capture datasets with hardware time stamps accurate to better than 1 microsecond. The traffic must stress the net to expose AFDX weaknesses and any product implementation errors and timing jitter. Do many high-rate (1 ms) packets cause more stress than high-rate packets interspersed with large, low-rate data payload packets? How does congestion overloading of the switching hub (via multiple simultaneous AFDX traffic generators) affect packet timing jitter? Can the overloaded switch—which adds latency to packets subject to network contention—produce violations of the maximum packet jitter? What are the throughput and configuration issues associated with larger, more representative ASN networks?

The Study Team also developed software to analyze the data produced by the AFDX testbed. Loading the network generated thousands of packets per second, each generating a detailed packet-capture record. Analysis of timing jitter required software to read this data and perform conformance analyses. To answer questions posed in the task request, the Study Team developed network performance metrics based on jitter statistics (e.g., minimum, average, maximum, and standard deviation), number of packets and total bytes per second (representative of network throughput), type of errors encountered under stress conditions and the sensitivity of each rate group to timing variations. Test runs were long enough to generate statistically meaningful data.

The packets were captured and the data set analyzed for absolute timing, timing jitter, VL consistency, and errors. Deviations were identified and analyzed before proceeding to the next testbed configuration. Our data analysis tracked SNs for each VL, independently flagging missing packets to understand the mechanisms of packet loss in AFDX networks. Key metrics used to characterize AFDX performance are BAG timing variables (such as average packet-to-packet time), their standard deviations, and the minimum and maximum values observed for each VL under test. Additional metrics included any packet errors observed, the skew between two redundant channels, and the total sustained bandwidth for each VL and for the AFDX network as a whole.

At full load, the AFDX network produced thousands of packets per second, producing many thousands and even millions of packets for analysis, producing good timing statistics. Since each run could include tens of thousands of data capture records—each packet record having more than 40 data fields—this examination of the data had to be automated. The analysis process required sorting the full packet capture dataset by VL and also by channel for

redundant data sets. A key requirement was to identify any missing or dropped packets.

The analysis program Panorama V was used to validate the data sets and to perform statistical analysis of the captured packet data to produce BAG timing, jitter timing, and redundant-channel timing statistics. The team chose to use Panorama, a "personal computer" database/spreadsheet application from ProVUE Development (www.provue.com) because of its excellent tools for exploratory data analysis. Panorama includes a powerful scripting language for both database manipulation and interface development. It also offers exceptional data processing speed. Using its database features, the captured records could be easily parsed, augmented, sorted, selected, summarized, and displayed. Using the scripting language, the team captured these manipulations as screening routines for validating data. Subtle anomalies buried in the data could thus be brought to attention, enhancing the team's understanding of the generating processes and preventing errors from appearing in the statistics.

Switch Latency Data Analysis

The first task of the laboratory evaluation was to characterize the switch. The evaluation used commercial Ethernet switches (Catalyst 2900) from Cisco. Aerospace AFDX product vendors recommend using such switches for testing, since actual AFDX switches are very expensive, only built to custom order, and may block the errors and other phenomenon we were trying to measure. Theoretically, the switch should present fixed time latency for network traffic if lightly loaded, with a simple constant timing delay for each packet. With testbed timing accuracy better than 1 µsec, we were able to confirm that hypothesis showing that latency scaled with packet length, validating the test setup and accuracy of each packet monitor. It should be mentioned that an AFDX switch is statically configured, with each VL and BAG defined prior to use. Furthermore, there is to be a maximum of 100 µsec of packet delay, and buffer sizes need to be scaled to the AFDX network to prevent packet loss.

A Cisco Catalyst 2900 was statically configured to support the current test setup. The Cisco Catalyst 2900 allows the user to “tie” two or more switch ports together with the creation of virtual LANs (VLANs), isolating AFDX traffic by statically configuring the switch to reduce collisions. This is very similar to the predefined AFDX switch configuration table. Collisions are prevented and packet loss is minimized since traffic is steered only to ports active in the test. This is important, because we wish to observe packet loss due to AFDX end-station malfunction, difficulties in VL scheduling, traffic overload, and other phenomena that require the switch to work properly for AFDX emulation.

The following tests were used to quantify Cisco 2900 switch latency. The reference end station system was used with the Cisco switch in one segment and a direct connection in the other segment. In this configuration, Channel A-to-Channel B skew represents absolute switch latency.

The Cisco switch latency scales with packet size and not with VL or BAG values. A full packet (1518 bytes) incurs a latency of about 132 µsec, while a 512-byte packet incurs roughly one-third that value, or 52 µsec. At the low end, a 64-byte packet requires 16 µsec, which is not quite linear. The Cisco switch is about 30% slower than the AFDX switch specification allows.

The Cisco switch uses an 8 MB buffer global to all ports. This will introduce an aggregate of 640 msec of delay for all packets being buffered in the switch. It will also be able to accommodate at least 4000 packets before overrun and packet loss. The basic difference between various switches is the total amount of latency they can introduce into a network and also the point (in terms of stored packets) at which they begin to drop packets. The Cisco is able to handle a significantly larger number of packets in its buffer than other switches, resulting in much longer delays prior to packet loss.

The switch should learn the MAC addresses of end stations connected to each port and then use them to switch the Ethernet frames to the correct port. Our analysis revealed that the switch was broadcasting to all ports, and that simultaneous packet delivery to multiple ports (even ports not involved in the VL data transfer) was resulting in extra “collisions” leading to packet delay and possible loss. In this mode, all packet traffic was affecting all active switch ports. Proper operation would have the packet traffic switched to only those switch ports that were active in the VL, resulting

Table 2. Cisco Switch Latency Data Analysis Summary

AIM-1 Port 1 & 2 transmit direct and through Cisco switch to AIM-2 Port 1 and 2

VL	BAG msec	Pkt Size bytes	BAG Ave msec	BAG SD µsec	BAG Min msec	BAG Max msec	Skew µsec	BW Mbps	Errors
10	128	1518	128.00	0.29	128.000	128.001	132	0.1	0
11	64	1518	64.00	0.21	64.000	64.001	132	0.2	0
12	8	512	8.00	0.11	8.000	8.001	52	0.53	0
13	1	64	1.00	0.04	1.000	1.001	16.2	0.67	0
14	1	64	1.00	0.05	1.000	1.001	16.2	0.67	0
15	1	64	1.00	0.04	1.000	1.001	16.2	0.67	0
							Totals	2.84	0

in lower traffic rates at each port. This switch works in a broadcast mode until it is able to fill out its switch table; then it starts buffering packets and routing them to the correct ports based on MAC addresses. Further study revealed that most 802.3 switches learn the MAC address of connected end stations by capturing the MAC address of each transmitting station. This works fine for Ethernet, where all ports have a unique MAC address and also transmit at some time, allowing this algorithm to reliably fill out the switch table.

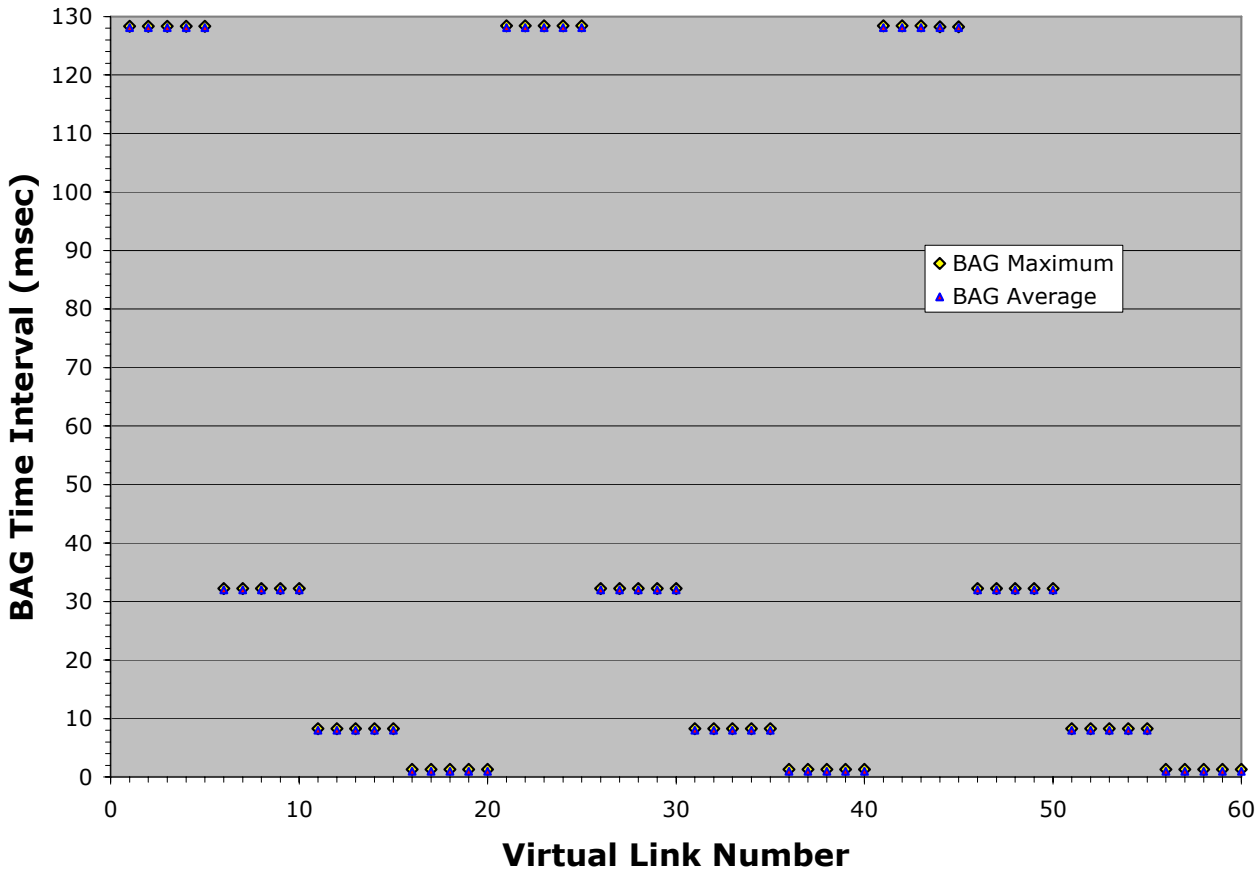
Each end station was connected to the switch using a full-duplex cable connection, which eliminated any chance of collisions on this network segment, but packet collisions could still occur internal to the switch. For example, when two end stations send a packet to the same receiving end station at the same time, the two packets must be staggered by delaying one until the other has been successfully received. Ethernet 802.3 switches have sufficient buffer capability and data transfer speed to be able to do this for at least a half-dozen packets per port. Moreover, the switch should use the MAC address to steer each packet to its destination port and no others, reducing contention.

Ancillary Sensor Network Test Vectors

The following test vectors were used to incrementally build and test an AFDX configuration to simulate the Ancillary Sensor Network (ASN). Traffic patterns that could be produced by sets of sensors at the prescribed rate groups were generated, with each test vector adding more sensor channels and loading the network more fully. Two end-stations running traffic generation and packet capture software were setup and were designated AIM-1 and AIM-2. The tests were run unidirectional, from AIM-1 to AIM-2 in a dual redundant configuration using the Cisco 2900 switch. The first test vector (TV32) used 60 VLs and the second test vector was run with 120 VLs to determine the effects on packet timing caused by more collisions in the switch due to increased network load.

Test Vector 32 (TV32), set up 15 VLs each at the 128-msec, 32-msec, 8-msec and 1-msec BAG values, for a total of 60 VLs. The packet sizes were 512, 256, 128 and 64 bytes respectively. This test was run unidirectional in dual-redundant mode. The following chart shows the data for Test Vector 32 on a scatter plot showing average measured packet BAG values and maximum BAG values. All values match up, indicating conformance to the AFDX jitter

Chart 1. TV32: 60 VL ASN Test



specification. For these charts, the blue triangle (measured packet BAG average value) should overlay the yellow diamond (measured packet BAG maximum value) completely if the VL is conforming to the AFDX jitter specification. If the yellow diamond is showing, this VL has at least one packet that exceeds the 0.5 µsec jitter specification. This allows easy recognition of which VLs contain jitter in excess of the specification.

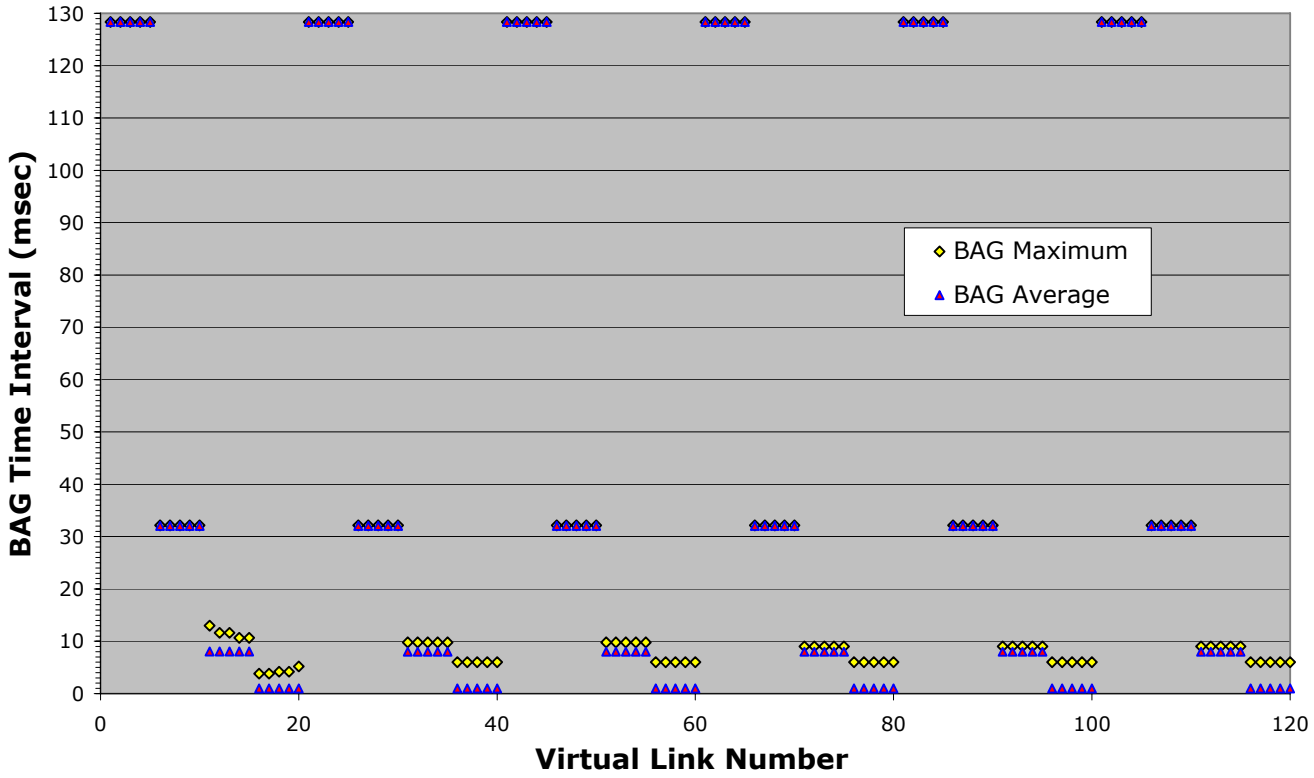
While the average values are still acceptable (with only a very small increase), the standard deviations have increased significantly and the maximum delay values push out to 450 µsec, near the jitter limit. The average of all the standard deviations in this set is 60 µsec, as compared to under one µsec for simple loopback configurations. Certain cycles, where many packets are issued simultaneously, are starting to experience switch congestion at a network load factor of 14%. The packet delay in this case is about 156 µsec, but the corresponding switch delay is a maximum of 1890 µsec, enough to start causing missed packets in high-rate groups, which was not observed with TV32. Skew between packets from Channel A and Channel B was always small and within specifications.

Test Vector 33 (TV33) expanded the number of VLs to 120, again by repeating the rate groups. This was done to demonstrate the effects of network throughput on packet timing. The chart below shows the average and maximum BAG values for this TV 33 test run. Average BAG timing increased by 0.3 msec for all the 128-msec VLs, with standard deviations nearly four times those of TV32. The standard deviations did not exceed the jitter threshold, but the maximum BAG value exceeded the jitter threshold of 0.5 msec many times as shown in Chart 2. Certain VLs of the high-rate BAGs are affected badly, with many 8-msec groups taking 12 and even 13 msec at certain times. The 1-msec BAG VLs can sometimes experience up to 4 msec of delay. Packet-issue delay would be double that of TV32, or 312 µsec, but maximum switch delay is an astounding 3780 µsec (3.8 msec), which has to be spread out between all active VLs, resulting in the occasional delayed packet. Network throughput in this case was 28 percent of total capacity.

The interpretation of these results centers upon understanding the maximum possible collision rate and consequent delay through the Cisco 2900 switch. Up to 640 milliseconds of delay is possible through this switch without dropping packets. With about 240 packets being generated sequentially by the two end stations simultaneously, delay through the switch is almost certainly the dominant factor—particularly since the packet-issue delay by each end station is much less by at least an order of magnitude. This interpretation is bolstered by the observation that BAG averages for TV33 are all longer than nominal. All jitter timing violations are positive, therefore caused by additional delay.

The exact interleave of packets through the switch and the resulting packet delay value was very dependent upon the

Chart 2. TV33: 120 VL ASN Test



relative phase of the network traffic from each port. Pressing a GUI button starts packet traffic from each AIM system; so all VLs start at the same time from that AIM unit but start asynchronously between AIM-1 and AIM-2 systems. There may also be minor timing differences between Channel A and Channel B, which may have an effect on collisions. Therefore, collisions in the switch resulting in packet delay will be different depending upon the phase difference between transmitting end stations. The switch must reconcile highly variable contention as phase differences between transmitting end stations varies. The worst case collision rate occurs when many VLs all transmit at the same time, necessitating buffering of most packets.

Collisions in the switch—which leads directly to variable latency for packets traversing the switch, increasing the jitter and disturbing AFDX network timing is the primary design issue for AFDX networks. If all end stations generate packets like clockwork, the only real source of jitter is in the switch. It is the central component that absorbs the timing uncertainties by correctly distributing the variable latencies between the packets in order to prevent BAG timing violations on all VLs. It is clear that switch delay (in cases where many packets are issued simultaneously, and buffered in the switch) is the main reason this test configuration—representative of a full-up ASN—has timing problems. A faster switch would help mitigate this timing problem. The Cisco switches are slower than the AFDX specification, which is 100 μsec maximum per packet. The Cisco switches exceed this by about 30%. This example fully demonstrates the importance of switch design, even in an AFDX network loaded to only 28 percent. This ASN would comply with the AFDX specification and ARINC 664 P7 if the delay through the switch were less. An AFDX-compliant switch actually regulates total packet delay to prevent such issues. Therefore, an ASN design similar to this would be feasible with the proper AFDX switch.

6. CONCLUSIONS

The Aerospace Industry has developed a series of new standards and components that would be suitable for high-bandwidth data acquisition and information processing. There are significant advantages to adopting ARINC 653 and AFDX communication standards for use aboard spacecraft. Our ASN design study has revealed some important lessons regarding IMA design:

- AFDX technology is mature enough to apply to specific data-handling functions aboard spacecraft, particularly to telemetry, data display, and flight data recording. Component certification is needed.
- Integrated Modular Avionics (IMA) architecture is a strong approach for improving software design, with functional partitioning resulting in improved validation and verification. It requires high-function LRUs and enhanced data communications, but delivers better fault isolation and configuration management. It is potentially sensitive to timing variations due to partition swapping.
- The ARINC 653 "Avionics Application Software Standard Interface" application program interface (API) and partitioned operating system is being adopted for flight control and should simplify software design and validation and verification.
- AFDX is well suited for IMA, providing good integration with ARINC 653 Queuing and Sampling Ports and the use of XML system-configuration methods.
- AFDX and similar network standards can dramatically reduce the number, weight, and volume of cable interconnects required for a spacecraft.
- AFDX can provide much higher data throughput than most aircraft control data buses. It can provide reliable time determinism and fault tolerance suitable for mission-critical applications.
- AFDX network functionality of the Ancillary Sensor Network was well established in laboratory evaluation, providing significant risk reduction for adoption of AFDX for Flight Test Instrumentation.
- Laboratory evaluation of ASN AFDX configurations showed that they could scale well to about 30% of full 100 Mbps transmit rate before high standard deviation of BAG timing and violations of the AFDX jitter specification became apparent.
- The major factor for AFDX timing violations and packet loss was overload of network switch buffers caused by too many packets arriving simultaneously. Staggering the packets from each end station can mitigate this loss.
- The AFDX network switch is a critical component and characterization of switch function and latency and the effect of collisions and packet loss on AFDX networks was determined.
- A real AFDX network prevents timing variations and packet losses by defining network VL timing characteristics and packet length during the design phase. Careful selection of switch buffer size can then eliminate most loss and delay exceeding the jitter specification.

REFERENCES

[1] ILC Data Device Corp., "Mil-Std 1553 Designer's Guide," 1995.

[2] G. Norris, M. Wagner, *Boeing 777*, 1996.

[3] R. Black, M. Fletcher, "Next-Generation Space Avionics: A Highly Reliable Layered System Implementation," Honeywell Intl., Glendale, AZ.

[4] ARINC 653 Standard, "Avionics Application Software Standard Interface," 2006.

[5] Creative Electronic Systems SA, "CES White Paper on AFDX," 2003.

[6] IEEE 802.3 Standard, "IEEE Standard for Local and Metropolitan Area Networks, Part 3: Carrier sense multiple access with Collision detection (CSMA/CD) access method and physical layer specifications," 2005.

[7] ARINC 664 Standard, "Aircraft Data Network," 2002.

[8] ARINC 664 Standard, "Aircraft Data Network," 2002.

[9] Airbus France, "AFDX End System Detailed Functional Specification," 2003.

[10] ARINC 664 Part 7 Standard, "Aircraft Data Network Part 7. Avionics Full Duplex Switched Ethernet ((AFDX) Network," Appendix B, 2005.

[11] IEEE Std 1451.1-1999, "IEEE Standard for a Smart Transducer Interface for Sensors and Actuators – Network Capable Application Processor Information Model," Instrumentation and Measurement Society, TC-9, Institute of Electrical and Electronic Engineers, New York, NY 10016-5997, SH94767, April 18, 2000.

[12] IEEE Std 1451.2-1997, "IEEE Standard for a Smart Transducer Interface for Sensors and Actuators – Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) formats," Instrumentation and Measurement Society, TC-9, Institute of Electrical and Electronic Engineers, New York, NY 10016-5997, SH94566, September 25, 1998.

[13] J. Schmalzel, F. Figueroa, J. Morris, S. Mandayam, and R. Polikar, "An Architecture for Intelligent Systems Based on Smart Sensors," IEEE Transactions on Instrumentation and Measurement, Vol. 54, No. 4, August 2005, pp. 1612-1616.

[14] F. Figueroa and J. Schmalzel, "Rocket Testing and Integrated System Health Management", *Condition Monitoring and Control for Intelligent Manufacturing,* pp. 373-392, Eds. L. Wang and R. Gao, Springer Series in Advanced Manufacturing, Springer Verlag, UK, 2006

[15] F. Figueroa, R. Holland, J. Schmalzel, D. Duncavage, A. Crocker, R. Alena, "ISHM Implementation for Constellation Systems", AIAA Conference 2005.

BIOGRAPHY

Richard L. Alena *is a Computer Engineer in the Intelligent Systems Division at NASA Ames. Mr. Alena was the co-lead for the Advanced Diagnostic Systems for International Space Station (ISS) Project, developing model-based diagnostic tools for space operations. He was the chief architect of a flight experiment conducted aboard Shuttle and Mir using laptop computers, personal digital assistants and servers in a wireless network for the ISS. He was also the technical lead for the Databus Analysis Tool for International Space Station on-orbit diagnosis. He was group lead for Intelligent Mobile Technologies, developing planetary exploration systems for field simulations. Mr. Alena holds an M.S. in Electrical Engineering and Computer Science from the University of California, Berkeley. He is the winner of the NASA Silver Snoopy Award in 2002, a NASA Group Achievement Award in 1998 for his work on the ISS Phase 1 Program Team and a Space Flight Awareness Award in 1997.*

John Ossenfort *is an employee of SAIC at NASA Ames Research Center, currently working on several projects including the Advanced Diagnostic and Prognostic Testbed (ADAPT) and the Systems Autonomy for Vehicles and Habitats (SAVH) project. In the past, he has been responsible for administration of multiple lab networks and participated in several field simulations, assisting in all aspects of wired and wireless network design, deployment, troubleshooting and maintenance. John has a dual BA degree in Anthropology and East Asian Studies from Washington University in St. Louis.*

Kenneth I. Laws *is a computer scientist with QSS Group, Inc., providing support to the Intelligent Systems Division at NASA Ames. At USC, he developed texture energy measures for image texture classification. After seven years at SRI International's Artificial Intelligence Center, Dr. Laws served for two years as NSF's Program Director for Robotics and Machine Intelligence. He moderated the AIList Arpanet discussion list (at SRI), then founded Computists International and published the Computists' Communique for ten years.*

André Goforth *is a Computer Engineer in the Intelligent Systems Division at NASA Ames. Mr. Goforth has twenty-five years of experience in application of computer technologies to NASA missions. He has worked on a number of intelligent systems projects ranging from supercomputing systems to avionics and flight software and integrated health management systems for ground and flight systems. He worked on the development of processing requirements for NASA Ames' Numerical Aerodynamic Simulator (NAS) supercomputer computing and communication requirements; the prototyping of high speed links for the NAS networking backbone; avionics requirements for the support of telerobotic operations for NASA's Flight Telerobotic Servicer flight article; development of the Space-Borne VLSI Multi-processor System (SVMS) processor, a VLSI-based parallel processor for support of artificial intelligence (AI) applications. He led the research and development for use of parallel Ada technologies for AI applications and was part of the Ada 9X effort sponsored by the DoD for which he received a DoD Certificate of Appreciation for Valuable Contributions. Recently, he served as the project manager for the Crew Exploration Vehicle Integrated Systems Health Management (CEV ISHM) project at Ames and received a NASA Group Achievement Award for his contributions in the development of the Advanced Diagnostics and Prognostics Testbed (ADAPT) at Ames. Mr. Goforth is a member of IEEE and has served on program and organization committees of IEEE Hot Chips and Compcon conferences. In addition, he has served as NASA's representative to the W3C and OMG standards organizations.*

Fernando Figueroa *holds a B.S. Degree in Mechanical Engineering from the University of Hartford, CT, and M.S. and Ph.D. degrees in Mechanical Engineering from Penn State University. In 1988, He joined the faculty of Tulane University, New Orleans; and was on the faculty of the University of New Brunswick, Canada, from 1995 to 1997. He joined NASA Stennis Space Center in 2000. His areas of interest encompass development of Integrated System Health Management (ISHM) capability for ground and space systems, Intelligent Systems, Intelligent Sensors, Networked Distributed Intelligence, Robotics, Sensors and Instrumentation, Controls. He has led multi-center, multi-entity, large-scale projects focused on implementation of ISHM capability on rocket engine test stands, the International Space Station and future components of NASA's Constellation Program.*