NASA/TM—2014-216628

Power, Avionics and Software Communication Network Architecture

William D. Ivancic, Obed S. Sands, Casey J. Bakula, Daniel R. Oldham, Ted Wright, and Martin A. Bradish
Glenn Research Center, Cleveland, Ohio

Joseph M. Klebau
Science Applications International Corporation (SAIC), Cleveland, Ohio

February 2014

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include creating custom thesauri, building customized databases, organizing and publishing research results.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at *http://www.sti.nasa.gov*

- E-mail your question to *help@sti.nasa.gov*

- Fax your question to the NASA STI Information Desk at 443–757–5803

- Phone the NASA STI Information Desk at 443–757–5802

- Write to:
  STI Information Desk
  NASA Center for AeroSpace Information
  7115 Standard Drive
  Hanover, MD 21076–1320

NASA/TM—2014-216628

Power, Avionics and Software Communication Network Architecture

William D. Ivancic, Obed S. Sands, Casey J. Bakula, Daniel R. Oldham, Ted Wright, and Martin A. Bradish
Glenn Research Center, Cleveland, Ohio

Joseph M. Klebau
Science Applications International Corporation (SAIC), Cleveland, Ohio

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

February 2014

Level of Review: This material has been technically reviewed by technical management.

Available from

NASA Center for Aerospace Information
7115 Standard Drive
Hanover, MD 21076–1320

National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312

Available electronically at http://www.sti.nasa.gov

Power, Avionics and Software Communication Network Architecture

William D. Ivancic, Obed S. Sands, Casey J. Bakula, Daniel R. Oldham, Ted Wright, and Martin A. Bradish
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Joseph M. Klebau
Science Applications International Corporation (SAIC)
Cleveland, Ohio 44135

Abstract

This document describes the communication architecture for the Power, Avionics and Software (PAS) 1.0 subsystem for the Advanced Extravehicular Mobility Unit (AEMU). The following systems are described in detail: Caution Warning and Control System, Informatics, Storage, Video, Audio, Communication, and Monitoring Test and Validation. This document also provides some background as well as the purpose and goals of the PAS subsystem being developed at Glenn Research Center (GRC).

Keywords: Communication Architecture, Protocols, Telemetry, Avionics

Contents

1 Background **2**

2 Documentation **3**

3 Terminology **3**

4 Communication Architecture Design Philosophy **4**

5 Switches and Controls **5**

6 Telemetry **6**
6.1 Bits on the Wire . . . 6
6.2 Google Protocol Buffers . . . 6
6.3 Messages . . . 7
6.4 Software . . . 8

7 AEMU Communication Architecture **8**
7.1 Internal Bus . . . 8
7.2 Radio Networks . . . 9
7.3 Discovery and Registration . . . 10
7.4 Naming and Addressing . . . 10

8 Subsystems **11**
8.1 Caution Warning and Control and Power Management and Distribution . . . 11
8.1.1 Description . . . 11
8.1.2 Implementation . . . 11
8.1.3 Operations . . . 12
8.1.4 Research . . . 13
8.2 Audio Processing . . . 13
8.2.1 Description . . . 13
8.2.2 Implementation . . . 13
8.2.3 Operations . . . 14
8.2.4 Inputs . . . 15
8.2.5 Output . . . 15
8.2.6 Research . . . 16
8.3 Communication . . . 16
8.3.1 Radio . . . 17
8.3.2 Description . . . 17
8.3.3 Operations . . . 17
8.3.4 Research . . . 17
8.3.5 Routing . . . 18
8.3.6 Description . . . 18
8.3.7 Operations . . . 18
8.3.8 Research . . . 18
8.3.9 Time Synchronization . . . 19
8.3.10 Navigation . . . 19
8.4 Informatics . . . 19
8.4.1 Implementation . . . 19
8.5 Storage . . . 20
8.6 Video . . . 20
8.7 Monitoring, Test and Validation . . . 20

9 Intersystem Communications **20**
9.1 Design Philosophy . . . 20
9.2 Messages . . . 21
9.2.1 Implemented Messages . . . 21
9.2.2 Future Work / Desired Functionality . . . 22

10 Summary **22**

Appendix A Example Message **24**

Appendix B Google Protocol Buffers AEMU.proto file **25**

1. Background

The current Extravehicular Mobility Unit (EMU) suites (space suits) are still using technology from the 1970's and 1980's, particularly regarding the communications system. These systems work well, and there is always a reluctance to change space systems. There is nearly always the desire to be backwards compatible with proven systems, to reduce risk. Unfortunately, this backwards compatibility often occurs to the detriment of new technologies infusion. Ironically, infusing new technologies may actually reduce cost while dramatically improving capability.

In order to infuse new technologies into the EMUs and prepare for future National Aeronautics and Space Administration (NASA) missions, NASA is currently developing an Advanced Extravehicular Mobility Unit (AEMU)[1]. This is being performed within the Advanced EVA Systems Development Project under the Advanced Exploration Systems (AES) Program. A key part of this development is the spacesuit Primary Life Support Subsystem (PLSS) technology unit for long-duration microgravity or planetary missions and vacuum or low-pressure environments. NASA GRCs role is to develop the PAS subsystem.

The PAS subsystem being developed at GRC supports the AEMU program at NASA's Johnson Space Center (JSC). GRC's role is to develop a prototype suite avionics subsystem and to research new technologies for the AEMU. The results will be used to refine requirements for the AEMU as well as to potentially integrate new technologies into the AEMU.

2. Documentation

In order to reduce cost and increase productivity, the PAS Communication Architecture group utilized modern documentation and collaboration tools. Within much of NASA, the current widely practiced method of handling issues, requirements and general documentation is to utilize spreadsheets and pass various versions around with one person in control. A "most-up-to-date" version is often kept on a collaborative server such as EMC Corporation's eRooms or Microsoft's SharePoint®. A problem with that practice is that the documents on those servers are rarely the current working documents. Rather, there are three or more working documents being edited, and when changes are made, nobody knows who made them, when or why. Thus the PAS Communication Architecture group adopted techniques used by software programmers including use of an issue tracker and wiki. This ensured all documents are in one place and up-to-date with most of the material directly on the wiki and with any critical issues documented in the issue tracker. Use of revision control (version control) has been implemented for all documentation, not just software. Users then know what the latest documents are as well as what revisions have been made with logged authors and modification times. The same can be said regarding issues and general documentation residing on the wiki.

3. Terminology

To aid in discussion within this document it is useful to develop and define some terminology specific to our concepts of operations.

Caution and Warning Events

Our preferred caution and warning definitions are below. We prefer these to past definitions as they align with industry practices, such as terminology used in industrial plants and power plants. These definitions were altered in the Intersystem Communications Section in order to maintain consistency with terminology currently in use on the International Space Station (ISS) and previously used on Space Transportation System (STS) a.k.a. Shuttle.

Alert Single Tone for minor events such as "Text Message Received" or when a suits operational configuration has changed.

Caution Take notice, but don't take action (usually flashing lights in the power plant).

Warning Take Action Now. (These take precedence over Cautions).

Data Types

There are four basic data types listed below. Note, there are no command types or response types. Rather, we use events and telemetry. The interaction between subsystems is based entirely on a publish/subscribe concept that greatly simplifies code and reduces the amount of state that one needs to maintain [1].

[1] A spacesuit that provides environmental protection, mobility, life support, and communications for astronauts performing an Extravehicular Activity (EVA)

Application Specific (e.g., file transfer)

Event - a message type indicating change of state that may require some action. Change of state includes items such as caution and warning signals, audio push button position changes, and call group selections.

Streaming - real-time streaming applications such as audio and video

Telemetry - system and subsystem status, configuration and measurements

Call Group (i.e. Voice Loop) A group of voice reception devices that all listen on a common "channel". For Voice-Over-IP (VOIP), the channel may be a multicast address and port. For simple Radio Frequency (RF) communications, that channel may be a particular frequency and time slot or spreading code. For illustration purposes, assume we have three AEMUs. Then, some examples of call groups for AEMU may be:

- All three AEMUs
- A single AEMU to Home and Mission Control
- All three AEMUs, plus Home and Mission Control
- Home and Mission Control

Home Home is where the crew members return to after their EVA (e.g. ISS, Habitat, Base Camp, Tier-1 Relay in the Network Diagram - Figure 4)

Multi-Homed Network A node connected to two or more Wide Area Network (WAN) networks such that data has multiple potential paths in which it can flow to it's final destination. For example the AEMU the communication system utilizes two separate radio networks and is thus multi-homed.

Telemetry Often telemetry refers to ALL data coming from a spacecraft to the ground. For the AEMU, telemetry is just system and subsystem and assembly status, configuration and performance measurement data (e.g. suit internal pressure, suit internal temperature, power supply voltage, power supply current, radio transmission rate). All other data flows such as file transfers, and streaming video and audio are considered data flow and not telemetry.

4. Communication Architecture Design Philosophy

The goal is to develop a flexible, extensible, testable, and cost effective architecture. The design is not just for use on the ISS or Orion, but for future planetary exploration science missions' terrestrial sorties. There is also a great desire to be able to readily infuse new technology.

PAS Assemblies have been developed, for the most part, independently. This was done for a number of reasons. First and foremost, Caution, Warning and Control System (CWCS) is a critical system that is being developed with a direct path to flight in mind. Thus, there is a desire to keep both the hardware and software as simple and reliable as possible while maintaining power and mass requirements. The audio processing and communication assemblies both have significant research elements. Allowing these assemblies to be developed independently enables these research elements to to be developed on parallel paths. The informatics system is highly desirable for AEMU sorties involving scientific research and terrestrial exploration. As such, it is considered a long-term development.

Currently, each of the subsystems has its own processing unit. A future AEMU would likely have only two or three processors. The CWCS would likely maintain its own processor while the rest of the systems might share one. The communications and audio processing system may be combined having a separate processing unit. Informatics may have its own high-end processor and software as this unit is likely to be deployed last and updated often. This should not be a problem as Informatics is a "low-critical subsystem" 2.

NASA's current mission and budget profile drives the AEMU development to occur in a phased, evolutionary manner. One could anticipate that the CWCS, the audio, and the communications assemblies would be developed prior to informatics. This is particularly true since informatics provides its greatest return on investment for exploration sorties rather than EVA maneuvers around International Space Station or the Orion spacecraft.

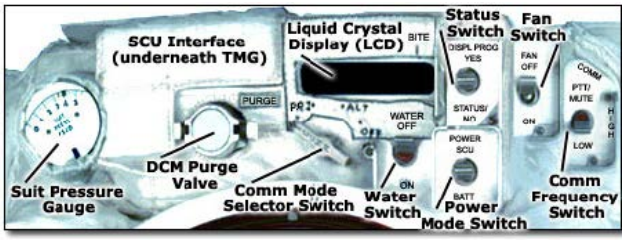

Figure 1: Power, Avionics and Control Software Critical Systems

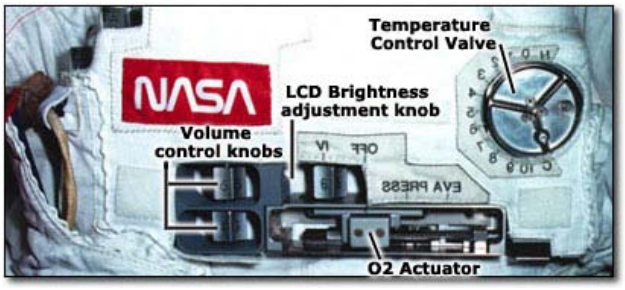

Figure 2: Power, Avionics and Control Software Critical Systems

Due to limited resources (funds, people, and time), existing software technologies and techniques have been adopted such as using Google Protocol Buffers for telemetry, ZeroMQ® for messaging, and Wireshark® for monitoring activity across the Gigabit Ethernet (GigE) and WAN busses.

There is also a great desire to learn from other's experience. For example: the AEMU sortie operations are very similar to soldier operations as are many of the requirement and features. Both the army operations and the AEMU need separate call groups and multi-homed radio systems. The Soldier Radio Waveform project already has these capabilities [2]. The AEMU is also quite similar to Intelligent Transportation Services (ITS) [3] vehicles with some systems being critical for safety of life and others being desired features. In many ways, ITS has more stringent requirements as security is a major concern.

5. Switches and Controls

On the current EVA suits, the Display and Control Module (DCM) is where the EVA crewmember interacts with the PLSS. It contains displays and controls for the operation of the EMU and is attached to the front of the upper torso of the EMU . In order to read the displays at this location, the crewmember uses mirrors (attached to the arms of the EMU) to reflect and read the information, which is written backward on the front of the suit. Sensor readings are provided to the crew and read on the top of the DCM on the Liquid Crystal Display (LCD) [4] - see figure 1.

For the prototype AEMU, external controls such as switches and knobs are associated with individual assemblies. For example, the Push-To-Talk (PTT) and the volume control switch are associated with the audio assembly. The call group knob (rotary switch), similar to that shown for Temperature control in figure 2, is associated with the communication assembly. In this instance, there are no commands either from other subsystems or external to the EMU to set the volume. It is performed by the audio assembly via the volume switch. However, there is a desire to display the volume settings. As such, a telemetry message containing volume information is periodically published by the audio assembly making it available to other subsystems or remote systems that desire the information. In a similar manner, the Communication system creates a telemetry message indicating the current call group setting (knob position).

6. Telemetry

Our desire is to develop a flexible telemetry format that does not required predefined large fixed-field data structures to be sent with each transmission. As such, we send only the relevant telemetry that two subsystems need to communicate.

We investigated fixed ID/Data pairs such that each piece of telemetry is identified by a Flag indicating if the next field value is an ID or data. The ID value would indicate what the data represents (e.g. voltage, pressure, Bit-error-rate, etc, ...). The data value would represent the value corresponding to the ID (e.g., a Boolean value, Integer value, etc...). Upon further investigation, we decided to utilize Google Protocol Buffers (GPB) (or simply "protocol buffers") [5] because it is a standardized format similar to what we were developing, and because it generates source code for the protocol encoding and decoding tasks. The telemetry format is shown in figure 3. Protocol buffers can be run over User Datagram Protocol (UDP) if so desired, but currently all messaging is implemented using ZeroMQ (Transmission Control Protocol (TCP) transport mode) for ease of development in our prototype subsystem - see section 9. Protocol Buffers uses a key/value pair for each entry. Many of these key/value pairs utilize *Variants*, a variable length method of serializing integers using as little as one byte. This method allows for very efficient bits-on-the-wire encoding. In our deployment, the only required key/value pairs are *Timestamp*, *Sender* (e.g, CWCS, Informatics, Audio, COMM) and *Type*. All other key/value pairs are optional (depending on the required Type field), enabling great flexibility.

6.1. Bits on the Wire

In addition to the protocol buffer payload, the packet wire format contains zero to three null (`0x00`) padding bytes appended to the end to make the total ZeroMQ data size a multiple of 32 bits. This is needed to preserve compatibility with older ARM microprocessors that do not have native instructions for byte alignment. The protocol buffer payload is prefixed by a 16 bit field that stores the length of the payload in bytes. A 16 bit Cyclical Redundancy Check (CRC) is computed over the combined length, payload and pad bytes and is added before the length field. The CRC and length are stored in network byte order. Thus we have:

1. A 2 byte CRC calculated over of the rest of the packet (network byte order)
2. 2 bytes which are the length of the encoded protocol buffer (network byte order)
3. A variable length encoded protocol buffer payload
4. 0 to 3 pad bytes (`0x00`) to make the total packet length a multiple of 4

6.2. Google Protocol Buffers

Use of GPB greatly simplifies coding as all header information and associated data structure are automatically generated from the GPB ".proto" file. The following excerpt is taken directly from the protocol buffers web pages:

> "Protocol buffers were initially developed at Google to deal with an index server request/response protocol. Prior to protocol buffers, there was a format for requests and responses that used hand marshalling/unmarshalling of requests and responses, and that supported a number of versions of the protocol. This resulted in some very ugly code. Explicitly formatted protocols also complicated the rollout of new protocol versions, because developers had to make sure that all servers between the originator of the request and the actual server handling the request understood the new protocol before they could flip a switch to start using the new protocol.
>
> Protocol buffers were designed to solve many of these problems:
>
> - New fields could be easily introduced, and intermediate servers that didn't need to inspect the data could simply parse it and pass through the data without needing to know about all the fields.
> - Formats were more self-describing, and could be dealt with from a variety of languages (C++, Java, etc.)
>
> As the system evolved, it acquired a number of other features and uses:

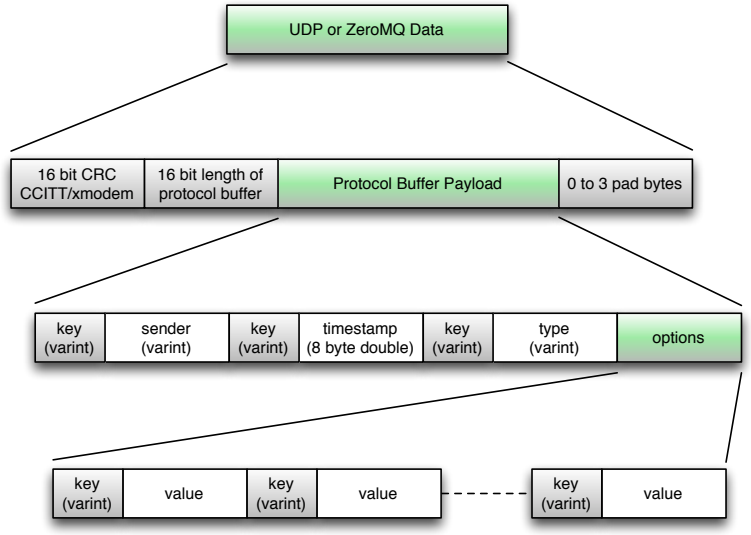

Figure 3: Protocol Buffer Telemetry Format

- Automatically-generated serialization and deserialization code avoided the need for hand parsing.
- In addition to being used for short-lived Remote Procedure Call (RPC) requests, people started to use protocol buffers as a handy self-describing format for storing data persistently (for example, in Bigtable).
- Server RPC interfaces started to be declared as part of protocol files, with the protocol compiler generating stub classes that users could override with actual implementations of the server's interface.

Protocol buffers are now Google's lingua franca for data at time of writing, there are 48,162 different message types defined in the Google code tree across 12,183 .proto files. They're used both in RPC systems and for persistent storage of data in a variety of storage systems."

Since protocol buffers are so widely used, additional tools have been created by third party developers. PAS 1.0 uses the following protocol buffer related software:

- nanobp is a plain C language implementation of Google's Protocol Buffers data format and is used instead of the C++ language default implementation. It is targeted at 32 bit microcontrollers, but is also fit for other embedded systems with tight (2-10 kB Read Only Memory (ROM), <1 kB Random Access Memory (RAM) memory constraints. [6]
- protobuf-wireshark is an auto-generated Wireshark dissector plugins for GPB messages. [7][5]

6.3. Messages

Since the AEMU PAS 1.0-subsystem is a prototype used to develop the necessary requirements and raise the technology readiness level closer to a flight implementation, we concentrated on developing the required messages and human readable code and were not overly concerned with the efficiency of bits-on-the-wire. To speed development, PAS 1.0 uses the ZeroMQ [8] messaging library to avoid the complications of writing reliable publish/subscribe networking code. We feel it is far more valuable in this prototype effort to concentrate on the Messages (what needs to be communicated between subsystems and AEMUs) rather than the Messaging implementation.

The following summary of ZeroMQ features are taken from the ZeroMQ web site and user guide:

- Moves data between networked subsystems regardless of its format

- Faster than UDP , for clustered products and supercomputing
- Has excellent support for the publish/subscribe network architecture
- Carries messages across inproc, Interprocess Communication (IPC) , UDP , and multicast
- Connect N-to-N via fanout, publish/subscribe, pipeline, request/reply
- Asynchronous I/O for scalable multicore message-passing apps
- Large and active open source community.
- Supported by 30+ languages including C, C++, Java, Python
- Implemented on several operating system, including Linux, Windows, OS X

A Publish/Subscribe network architecture is a best practice for reliable information exchange between loosely coupled networked assemblies, such as the AEMU. While it is possible to create a simple system based on UDP broadcasts, past experience has shown that making it reliable and fixing all the corner cases can become complicated. Building distributed software and getting code talking to code can be difficult and expensive. The open source ZeroMQ socket library makes coding a basic publish/subscribe system trivially easy, requiring about a dozen lines of code. ZeroMQ is a library that is linked to code, so unlike most other "middleware" solutions, it does not require a separate daemon process or broker software. It was originally developed for high frequency trading applications on Wall Street where efficiency is paramount.

6.4. Software

In the PAS 1.0 interfacing effort, there is a great desire to develop assembly software for reuse. Using Google Protocol Buffers and ZeroMQ enable this. With protocol buffers one can easily add key/value pairs without affecting existing code. Using ZeroMQ one can easily switch transport mechanisms as it can use different protocols depending on the peers location (TCP, Pragmatic General Multicast (PGM) multicast, IPC, inproc shared memory), [9] which is extremely importantly for our code reuse. One can easily switch from an Internet Protocol (IP) routing to IPC or in process shared memory. Thus, if the bus structure changes from GigE to Peripheral Component Interconnect (PCI) bus or some other bus, one simply needs to change the transport type in ZeroMQ code and not much else (e.g., in the italicized line below, tcp become PGM for multicast and IPC for interprocess communication).

*subSockets.connect(**tcp**://localhost:%d' %pubPort)*

Software was developed in a manner that allows each assembly to draw upon a common library of messages This enables an assembly's software to remain intact even if the bus architecture changes.

Originally we were going to implement a common set of Application Program Interface (API)s. However using ZeroMQ and protocol buffers, the messaging was so simple, that there was no need for actual API development. We were able to go from message construct development to integration, test and validation in approximately 2 months with communication between four unique assemblies.

7. AEMU Communication Architecture

Figure 4 shows the AEMU communication architecture block diagram. The AEMU has three networks.: The local area networks, the low-rate, critical RF communication network, and a high-rate, wide-band RF communication network. Figure 4 also shows the basic addressing scheme selected for both the GigE and WANs i.e., the radio networks. All networks in this architecture are addressable via Internet Protocol version 4 (IPv4) and Internet Protocol version 6 (IPv6) addressing. The preferred choice is to use IPv6 addressing as all future technology developments related to Internet protocols are likely to occur using IPv6 address space.

7.1. Internal Bus

All the assemblies are connected via a GigE bus. GigE was selected to handle large amounts of data generated from the video assembly as well as ease of subsystem integration. This ensures sufficient available bandwidth across the bus for all assemblies, but more specifically, to handle multiple streams of high definition video. Furthermore,

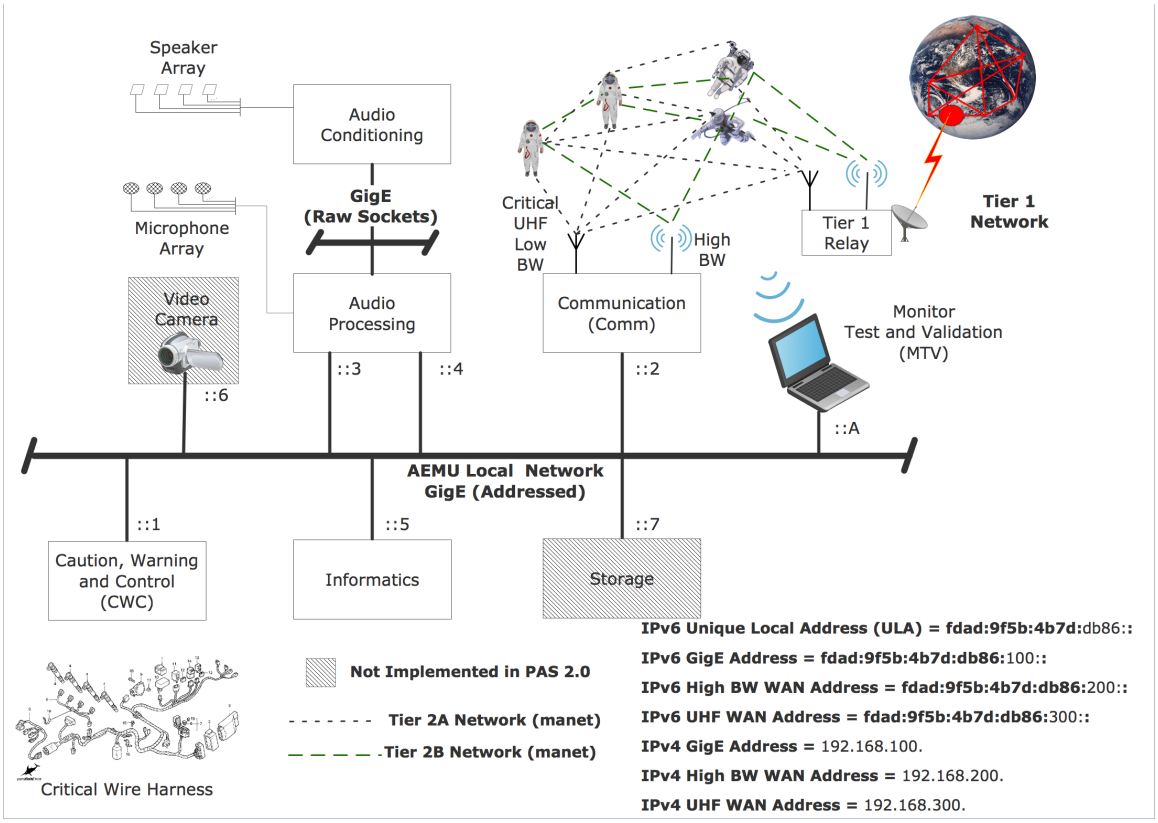

Figure 4: Power, Avionics and Control Communication Architecture

software and drivers are readily available in most operating system to easily move data across an Ethernet bus. No custom driver software needs to be developed, saving time, energy and money.

For the local area network the onboard network, each subsystem has an address unique to the local network. Although unique to the local network, this address space is identical for each AEMU. All routes off the suit send information to the communication subsystem. The communication subsystem is the default route to external systems. Likewise, all information coming from external sources would be received by the communications subsystem and routed to the appropriate subsystem.

7.2. Radio Networks

There are two separate RF networks: a low-rate UHF radio network and a high-rate, wide-band radio network. Each radio network has its own address space and is on its own subnet. The critical communication RF network is a low-rate network over a UHF radio system. Information transferred over the system includes voice and critical telemetry. The wide-band communication radio network may carry voice, critical telemetry, video, and noncritical data such as scientific data taken during a terrestrial sortie. There are many possible ways to utilize these networks. The first is to always have audio communication and critical telemetry flow over the UHF radio and all other data to flow over the wide-band RF system. This, however, precludes the use of the wide-band system for critical communications. As such, in the event that the UHF system fails, one would be unable to take advantage of the wide-band system even if it is working. A second possibility would be to have all communication flow over the wide-band system with the critical system in standby and only active when a wide-band first system fails. However, it is been shown that the wide-band system conductivity will fail often so this is probably not a valid operating mode [10]. The third possibility and probably the most reliable would be to have both radios operating simultaneously with voice and critical telemetry flowing over the UHF system and all other communications flowing over the wide-band system with an option to enable critical communications to flow over the wide-band system if the UHF radio system were to fail.

This may introduce undesired complexity. A detailed study of such an architecture would have to be completed to prove or disprove this complexity assertion.

The UHF radio network has a much further range than the wide-band radio network. As currently envisioned, all radios in this network are expected to be within range of each other, that is, there is no expectation of using these radio nodes in a multi-hopped Mobile Ad Hoc Network (MANET) network. However, using a multi-hopped MANET network over the UHF radios could greatly extend the range of critical communications - perhaps with little additional complexity since MANET routing is already designed into the communication system for the wide-band network.

The range of wide-band radios is limited. In order to maintain connectivity between wide-band radio nodes and HOME, each node is designed as a MANET forwarding agent to enable a multi-hoped communication and extend the overall range of the wide-band network. It is certainly possible and perhaps advantageous to actually combine the UHF and wide-band radio networks into a single MANET network or to have two separate MANET networks: the critical radio network and the wide-band radio network.

It is important to note that although the system described here is for the AEMU, the multi-homed radio network most certainly could be used by other systems such as a group of cooperative robots or rovers. Developing a single multi-use communication system would allow space agencies to reduce costs and improve reliability of the overall communication system as well as distribute costs across various groups such as the robotics missions and the human exploration missions.

7.3. Discovery and Registration

In order to simplify network configurations and improve reliability by avoiding mis-configurations, it is highly desirable to have some form of automated system and service discovery and registration [11][12]. Furthermore, it is important that no single point of failure exists in the system. Therefore, all services should be designed to operate without a centralized server. This is very similar to the desired operation in a military edge network. In fact, many of the techniques used in military edge networks should be applicable, yet simpler to apply, since network security issues are quite limited in a space-base network versus a terrestrial military network. Since space networks, particularly the AEMU communication networks, are relatively small, scaling is a non-issue, which makes deployments much simpler than for large terrestrial networks. For example, techniques similar to distributed host tables are reasonable for networks of tens of nodes but may be unreasonable for millions of nodes.

A registration process consists of mapping services on a system such as an AEMU, robot, rover or HOME with the system ID; mapping the system ID to network addresses; and distributing this mapping. For a multi-homed system, a particular service may be reachable via multiple paths. The discovery process consists of locating systems and determining what services that system offers and/or desires.

There are many server-less technologies and services being developed. Many are directed at challenged networks such as communication within a remote village, or disaster relief and recovery where existing infrastructure does not or no longer exists [13]. Development of server-less applications is also taking place for security reasons as your data and messages are never stored on any server, so they are safe from data breaches and prying eyes [14]. This is of great concern for some corporate network as well as individuals per recent public revelations regarding government activities [15] and data mining activities of major Internet service providers such as Facebook®, Google®, Yahoo® and others. Regardless of why these technologies are being developed, they are quite applicable to space-based networks.

7.4. Naming and Addressing

In order to facilitate discovery and registration, a well thought out naming and addressing scheme is required. Naming is currently a major research topic in the Internet Research Task Force (IRTF), Information Centric Networking Research Group (ICNRG). Distributing and manipulating named information is a major application in the Internet today. Furthermore, Point-to-Point (P2P) and Content Delivery Network (CDN) are promoting a communication model of accessing data by name, regardless of location. Our network of AEMUs and rovers is an example of a P2P network. A good example of naming and addressing is presented in Day's book,"Patterns in Network Architecture [16]." A phone number use to be an actual physical address attached to an endpoint. Each portion of the phone number indicated a region of the country or city. Today, phone number is simply a phone service identifier indicating the identity of a particular endpoint. When one powers on their cell phone, it registers the phone number as a service request which gets mapped to a particular address. That address may be a 4G cellular network address at one point

in time and then IP address when the phone is attached to a Wi-Fi network. Thus, the phone number is no longer a physical address, but rather a service ID associated with a unique endpoint.

Since each AEMU in the architecture shown in figure 4 has identical internal addressing, in order to uniquely identify an AEMU, each AEMU has to have a unique name. As of September 2013 for PAS-1.0, that unique name was the suit identification (i.e suit ID). However, since different crewmembers can use the same suit, it may be more appropriate to be an name uniquely associated with the crewmember - similar to how a smart card embedded with a digital security certificate is associated with a individual. Regardless, all applications and services (e.g. voice communications, file transfers, video, medical telemetry) provided by any AEMU should be associated with the unique name, not any point-of-attachment addresses [16]. The application to point-of-attachment mapping occurs as part of the discovery and registration process and reregistration process where reregistration occurs when one perceives change in network attachment. This facilitates auto configuration as well as mobility (i.e. continuous communication when moving across various subnetworks).

8. Subsystems

There are four major subsystems being developed under PAS 1.0. They are: (a) the Caution Warning and Control System, (b) Informatics, (c) Communications, and (d) Audio. Also included is a Measurement, Test and Validation (MTV). For PAS 1.0, the Video system and the Storage system will not be developed. These systems are being considered for future implementation. Each of the systems will be described in detail in the following sections. *Note that, for PAS 1.0, much the communication architecture described above has not been implemented for this phase of the development effort.*

8.1. Caution Warning and Control and Power Management and Distribution

8.1.1. Description

The Caution, Warning and Control (CWC) includes the subassemblies for the PLSS control. The CWC Assembly monitors the data channels coming to it from the sensors within the PLSS, and detects out-of-range and anomalous readings. It then reports these caution and warning items to the crewmember as text on a small display and as tones through the same speakers in the helmet as those associated with the inbound voice audio i.e., there is only one set of speakers in PAS subsystem and all caution and warning tones run through the audio process subassembly.

The CWC Assembly is assigned a Criticality-1S designation, since it is monitoring a life-critical hardware item and is critical for warning the crew of depleted life support resources or safety hazards due to malfunction of the life support system. Any software on the CWC Assembly would be software Class-A as defined in NPR 7150.2A [17].

*The Power, Management and Distribution (PMAD)*subsystem distributes power to the avionics, PLSS, and tools of the EVA system. It includes the battery and recharging interfaces, as well as battery health monitoring. It is currently envisioned that most powered devices will derive power from a single suit power source, as opposed to separate batteries for separate devices.

Since the PMAD Assembly provides power to safety critical systems, the hardware and avionics are considered Criticality-1, and any software in the PMAD Assembly is Class-A.

The CWC subsystem monitors and controls the Power Management and Distribution subsystem. For control purposes PMAD is considered part of the CWCS. The allocation of critical systems is shown in Figure 5.

8.1.2. Implementation

The CWCS is the primary focus of the development effort regarding form, fit and function at a pre-Preliminary Design Review (PDR) level of maturity for PAS 1.0 [18]. The CWCS is designed to fit in a box approximately 10.25 inches (in.) long $\times$ 6.25 in. deep $\times$ 3.8 in. high, as shown in Figure 6.

When fully implemented, the CWCS will provide the following functions:

- Monitor the status of PLSS life-support functions (e.g. consumables and component status) for display to the EVA crewmember
- Provide PLSS fault detection to alert the crewmember of off-nominal conditions and perform corrective actions

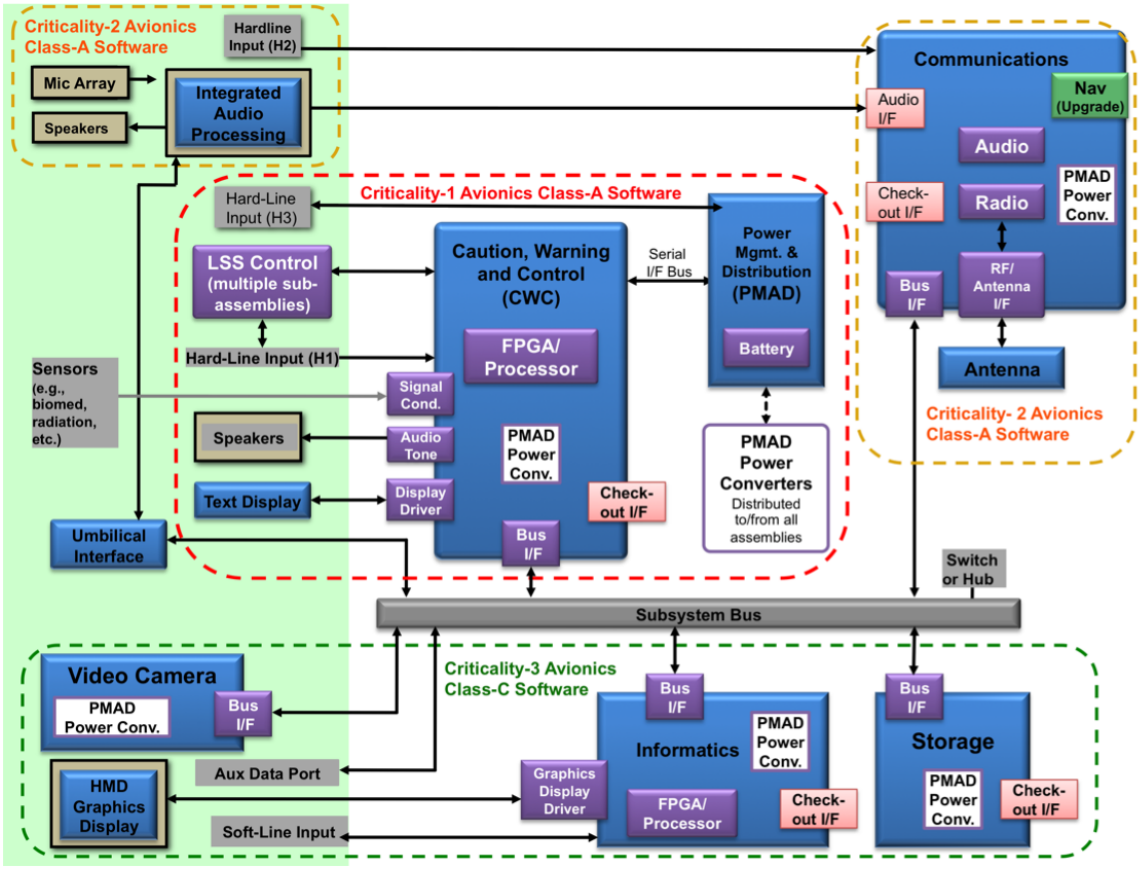

Figure 5: Power, Avionics and Control Software Critical Systems

- Electronically control the following PLSS components:
  - the Suit Water Membrane Evaporator (SWME), which provides heat exchange for the thermal control loop by vaporizing water (H2O) across a water membrane wall
  - a fan which circulates gases through the oxygen ventilation loop
  - the Temperature Control Valve (TCV), which is a diverter valve that adjusts the amount of cooling flow to the crewmember, either manually by the crewmember or automatically via a control algorithm
  - the Rapid Cycle Amine (RCA) swing-bed, which provides carbon dioxide (CO2) scrubbing and humidity removal of the ventilation loop
  - a pump which circulates H2O through the liquid cooling loop to provide crewmember cooling
- Monitor the Primary Oxygen Regulator (POR), which provides suit pressure regulation at multiple pressure set points of oxygen (O2) delivered from the primary O2 system
- Monitor the Secondary Oxygen Regulator (SOR), which provides suit pressure regulation at a single pressure set point of O2 delivered from the secondary O2 system

8.1.3. Operations

In an operational setting, the CWCS will not receive commands from off suit or from any other EVA subsystems. It only sends asynchronous Events (e.g., set point changes) or periodic Telemetry (e.g., health and status information).

Figure 6: PAS CWCS Packaging Illustration

For the development system, one may wish to send messages to the CWCS to configure the type of telemetry being sent and the frequency in which various telemetry messages are sent.

One critical hardware item that belongs to CWCS is the Caution and Warning Acknowledgement switch (ACK). ACK is used to control caution and warning tones. The messages CWCS sends to Audio indicate cautions or warnings include data showing whether or not these have been acknowledged. A lookup table or some other mechanism will be used to determine how long an acknowledgement is valid. It may be that for emergency warnings, one may wish to clear the acknowledgement after some period and force the crew member to re-acknowledge cautions and particularly warnings.

CWCS publishes Caution and Warning Report messages. These report message include a Caution or Warning time stamp, a Caution or Warning type, a flag indicating whether the message is a caution or warning, and a flag indicating whether this Caution or Warning has been acknowledged - see Section 9.2, Messages.

8.1.4. Research

The CWCS is an engineering development prototype effort. Information obtained from this development effort will be documented for use in developing requirements for future AEMUs. In addition, the AEMU contractors may decide to utilize various internal architectures, and techniques demonstrated here.

8.2. Audio Processing

8.2.1. Description

The Audio Subassembly provides for the handling and control of the audio signals retrieved from the integrated audio processing subassembly located in the helmet and delivers the audio to the communications subsystem or informatics. Likewise, the Audio Subsystem processes audio received from the communications subsystem and delivers that to the integrated audio processing subassembly. The Audio subsystem also is required to mix up to four separate inbound audio streams for the Communication subsystem along with caution and warning tones and deliver that to the integrated audio processor speaker subassembly.

The Audio Subsystem interfaces to the Communications subsystem through a GigE bus that is separate from the subsystem GigE bus and delivers microphone array signals prepared to be processed by the Audio Subsystems. Audio data is also sent from the Communication assembly to this assembly to be delivered to the speakers.

8.2.2. Implementation

Initial implementation was accomplished using an embedded processor development board using an Avnet Spartan-6/OMAP Co-Processing Development Kit for the outbound voice channel (voice off-suit) and a netbook computer to process the four inbound channels. The GStreamer open source software was used to encode and decode audio streams [19]. This was done simply to have a working audio system available for inter-subsystem message communication testing to test message passing between subsystems as there was approximately two months available from conception

to integration testing. Four additional netbooks were used to emulate four external audio sources thereby providing four inbound channels. The operating systems on both the embedded processor board and the netbooks was Linux Ubuntu. This allowed all software to be common between the embedded processor and the netbooks. Note, this is not the configuration for audio processing that would be deployed. It was done simply to enable testing of messaging (i.e. telemetry and events). Details of this implementation testing is available in the PAS Subsystem Interface Test Report [20].

8.2.3. Operations

There are three switches associated with the Audio system: (a) Mode, (b) Push-to-Talk, and (c) Volume. There is an additional switch owned by CWCS that directly affects the Audio system processing: the Caution and Warning Acknowledgement switch (ACK).

Mode (MODE) - The MODE switch is set for either PTT, Voice Operated Switch (VOX) or Open Microphone (Open-Mic).

Push-to-Talk (PTT) - If the MODE is set to PTT and the PTT switch is *not* set, no voice data will flow off suit. If the MODE is set to PTT and the PTT switch *is* set, voice data should flow to the appropriate call group(s) as selected by the Communication subsystem Call Group rotary switch.

Volume (VOLUME) - The Volume Switch increases or decreases the overall volume of the combined audio heard by the crew member. This is implemented to increase slowly while in the UP position and decrease slowly while in DOWN position. In PAS 1.0, individual in-bound channel volume is controlled via commands from Informatics.

Caution and Warning Acknowledgement (ACK) - ACK is used to control caution and warning tones. CWCS sends messages to Audio indicating caution or warning and whether or not these have been acknowledged. This switch will silence those tones. If the caution or warning tones have not been acknowledge, Audio plays the tones according to information found in a lookup table. That information includes: enunciation, tone frequency, and tone repletion period (with zero meaning continuous).

Audio modes include PTT and VOX, and OPEN MIC. You can effectively achieve a MUTE mode by selecting PTT mode on the mode switch and not pushing the PTT button. OPEN MIC is the default. If MODE is set to VOX, only audio that contains voice will flow off suit. In Open-Mic is selected, all audio is sent off suit even if nothing is being said (no voice activation). Thus, the receiving system will hear everything, including just breathing and background noise, assuming this is not filtered by some type of audio processing.

Regarding voice generated on-suit, Audio will send Informatics an audio stream in Linear Pulse Code Modulation (LPCM) encoding format. Informatics can store this for archiving purposes or use it for voice recognition commanding and voice notes. Audio will be sent off-suit according to the position of the corresponding audio switches and the call group switch or some type of indicator. A possible call group is "Local Only" such that no transmission goes off suit. For safety reasons, this may not be desirable or one may want to be able to over-ride this remotely from HOME or Mission Control.

Since all audio is being placed on the internal bus for the Informatics and Communication subsystem to pick up, both the MODE and PTT switches could belong to the Communication subsystem rather than Audio. This may actually simplify processing as no special telemetry messages indicating switch positions have need to be sent from Audio to COMM. However, the current approach is to keep these with Audio because giving the radio assembly a set of switches related to voice blurs the line between radio and Audio functionality.

For PAS 1.0, the Audio subsystem will locally store all locally generated audio data for one sortie's data (for PAS 1.0 the assumption shall be that one sortie is up to eight hours in length). In future AEMUs implemented with a separate Storage system, audio storage may occur in the storage subsystem rather than locally.

It is possible that there is currently sufficient processing power within the Audio subsystem to perform voice commanding. However, that function is currently be delegated to Informatics. Thus, all voice is transmitted to Informatics regardless of any switch positions. This is necessary in order for Informatics to always hear and interpret voice commands as well as for taking audio notes. A good example of current voice recognition software is Dragon

Dictation®. Dragon Dictation allows a soft switch and an audio command. To turn on voice commands one can use the GUI switch or say "Wake up". To turn off voice commands or dictation one can also use the GUI switch or say "Go to sleep". Informatics running similar software would need to receive all audio to process the "Wake Up" or "Go to Sleep" commands.

8.2.4. Inputs

For PAS 1.0, the Audio subsystem will obtain inbound channel gain settings for up to four channels from Informatics or the Monitor, Test and Validation system.

In order to set the proper audio gains, the Audio subsystem needs static pressure measurements from the suit helmet. Those measurements come from CWCS. For PAS 1.0 these will be periodically transmitted and repeated at a rate sufficient to maintain steady state gain.

For PAS 1.0, the caution and warning tones will be generated within the Audio subsystem along with associated aural annunciations. Notifications come from CWCS via messages sent across the GigE bus. One of four warning tone types (advisory/alert, caution/ warning and emergency) is associated with each warning type. The warning types are:

Information - tones used to direct crew attention to messaging of an informational nature (i.e. buddy warnings and text messaging)

Status - tones used to identify failures during suit checkout (i.e. failed leak check) and other messages of a status nature

Alert - tones used to identify successful progress during a suit checkout (i.e. start leak check & successful leak check)

Warning - tones uses to identify serious system issues upon which a crew member needs to take immediate action (including faults during an EVA)

The following Caution or Warning examples are taken from the NASA GRC EVA PAS Subsystem Architecture Data Book:

- Radiation High
- Oxygen Low
- CO_2 High
- Water Low
- Suit Pressure Out of Limit
- Battery Low
- Vent Fan
- Thermal Pump
- H_2O Heater
- LCG Cooling Valve
- Glove Heater

8.2.5. Output

The Audio subsystem will provide various health and status messages. Included in these messages will be switch settings and gain settings for each inbound channel and the local voice channel. Other health and status messages that may be useful are current and available storage, power consumption, or other measurable hardware related parameters.

For PAS 1.0, detailed system configuration settings are assumed to be available from a configuration file.

As stated earlier, the Audio subsystem will provide voice messages to Informatics for use in voice recognition and audio notes. In future implementations, each message may be tagged with the following information: *Time Stamp*, *Voice Activity Detection (VAD)*, *Gain*, *MODE* setting and *PTT* and might include a payload consisting of a nominal 10 ms of LPCM audio. These packets are transmitted as unicast UDP packets from Audio to Informatics.

Outbound streaming voice data from Audio to Communication shall be unicast. Communication shall re-transmit to other external nodes according to route tables and call group settings.

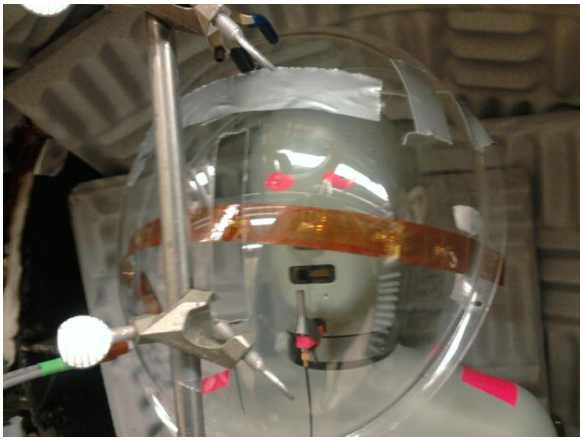

Figure 7: Microphone Array on Flexible Polyimide Circuit Board

For our initial PAS 1.0 interface tests performed in the August of 2014, audio distribution was performed quite differently as described in the test report [20]. GStreamer [19] audio streams were sent between external systems (e.g. Home and other AEMUs) using notebooks to emulate external systems. All GStreamer packet were sent unicast and channel (source) separation was performed using a different port number for each source.

8.2.6. Research

To date, the main research activities have been directed at eliminating the communications helmet (a.k.a. "Snoopy cap") while still maintaining acceptable speech quality. Research includes speech quality and intelligibility tests and determining audio processing, microphone array and placement options. For example: figure 7 shows microphone array on flexible polyimide circuit board. This is a prototype and the actual microphones would be much smaller.

Research ongoing in Digital Signal Processing solutions for audio processing focusing on temporal/spatial processing includes:

- Adaptive beamforming
- Linear beamforming
- Structure borne vibration suppression
- Echo cancellation

Implementation and processing technologies being investigated and deployed include:

- PGA, DSP processors
- MEMS microphones, Electret microphones
- Analog, digital array interfaces

8.3. Communication

The Communication Subsystem contains the baseband processing and formatting of the digitized data that is transmitted and received, including digitized voice. The Communication subsystem performs the routing of information on and off the AEMU as well as configuration control of the radios.

The Communication Assembly is assigned a 2R Criticality, assuming that if the voice communications are lost during an EVA, the mission (the EVA) would have to be terminated on that day. If communications could not be repaired, the mission would be seriously jeopardized. Software for the Communication assembly (and Navigation if included as an upgrade) is assumed to be Class-A based on requirements for similar manned systems, and as defined in NPR 7150.2.

8.3.1. Radio

8.3.2. Description

Due to the often conflicting requirements of high data rates and reliable radio coverage, a dual-band radio architecture is proposed for AEMU suit systems. This radio would use a wide-band, high data rate RF interface with enough throughput to handle simultaneous voice, telemetry, and video flows, as well as a lower data rate RF interface that can carry mission essential voice and telemetry. The wide-band interface would be used amongst all radios while they are in range of each other, Since the range of wide-band networks is relatively short, the low-rate interface will be utilized when the wide-band interface fails. Non-essential data flows originating from the suit will be queued up locally on the suit and transmitted when the wide-band network is once again available. This concept allows for each of the two interfaces to be designed separately according to each set of requirements for both coverage and high data rates. [10]

8.3.3. Operations

For PAS 1.0, neither a dual-band radio nor MANET routing were demonstrated. Rather, a single wide-band 802.11 radio broadcast network was utilized (i.e. all radios are within range). The radios were configured for "ad-hoc mode" rather than "infrastructure mode" so that there is no single point of failure (access point failure).

Operationally, there is nothing to do relative to the radios except turn them on or off. There will eventually be health and status data that will be placed into telemetry messages.

One potential operational consideration is to have the wide-band radio shut down to save power when battery power falls below some threshold. This could be accomplished either via a specific command from the CWCS or by monitoring the CWCS telemetry power status and taking appropriate action. The later allows reduced software complexity in both the CWCS and Communication system while allowing both units to evolve independently.

8.3.4. Research

In the Fall of 2011, the NASA GRC participated in the Desert Research and Technology Studies Desert Research and Technology Studies (DRATS) field experiments held near Flagstaff, Arizona. A Dual-Band Radio Communication System was demonstrated during this outing. The EVA radio system was designed to transport both voice and telemetry data through a mobile ad hoc wireless network and employed a dual-band radio configuration [10]. Some key characteristics of this system include:

- Dual-band radio configuration. One radio is used for high data rate communication and the other radio is used for contingencies to provide improved coverage and extended operational range.
- Intelligent switching between wireless networks with two different capabilities. The switching algorithm utilized the expected transmission count for the wireless links in order to select the appropriate wireless network.
- Self-healing network. Alternative data paths are determined either within the same network, or by transitioning to an alternative independent wireless network with different capabilities and characteristics.
- Simultaneous data and voice communication. The wireless communication system uses Media Access Control (MAC) layer traffic control and open source Speex-based Voice over Internet Protocol (VoIP) technology.

The significant research challenge lies in the implementation of this wireless interface on a hardware platform certified for operation beyond the Low-Earth Orbit (LEO) environment. Currently, a flight-grade chipset that implements a commercial Wireless Local Area Network (WLAN) interface does not exist. The questions that need to be answered regarding this include what will the costs be for the design and fabrication of an Application Specific Integrated Circuit (ASIC) chipset and which wireless interface standard will be most suitable for the wide variety of spacecraft proximity communications applications.

Research is ongoing into development of a new lower data rate RF radio waveform to replace the current Space-to-Space EMU Radio Space-to-Space EMU Radio (SSER) [21]. The current SSER waveform cannot support new technologies like packetized IP data and its telemetry format is fixed. The current SSER has heritage to the 1970's and is designed for ease in integration with the 1553 Bus onboard the spacecraft. Research involves modernizing this waveform to ease integration with a routed packet network. Some of the issues that need to be addressed are

the design of a MAC protocol that maintains the reliability of the SSCS and also meets the requirements for ad-hoc routing, supporting packetized IP, and increasing the robustness of the physical layer by considering things like short/long-term channel estimation feedback, adaptive modulation and coding, and dynamic spectrum access, which are challenging problems in a decentralized radio network. Another desirable feature of this interface is the ability to re-configure the waveform on orbit, so there are also open research issues regarding how to build a flight grade software-defined radio with a low enough Size, Weight and Power (SWaP) footprint that it may be used on the suit. This will require hardware components with capabilities beyond those used in the software-defined radios developed for the Mars program.

8.3.5. Routing

8.3.6. Description

For general AEMU development, routing should be able to accommodate the generic network scenario shown in figure 4 and described in DRATS field experiments report [10]. In general, one should be able to accommodate a multi-homed, dual-band radio with policy-base routing such that only critical voice and telemetry flows over the Ultra High Frequency (UHF) reliable radio network whereas high-rate science data and video flow over the high-rate radio network. Furthermore, the radios should allow for multi-hop routing (routing through multiple suits). Much of this has already been demonstrated and new technologies and techniques are being produced at a rapid pace [22] [3].

8.3.7. Operations

For PAS 1.0, the IP addressing scheme is provided in figure 4. IPv6 addresses should be used wherever possible as all new commercial Internet technology is expected to utilize IPv6 addressing. This is particularly true for mobile ad hoc networks.

There are currently no plans to implement Delay/Disruption/Disconnection Tolerant Networking (DTN) on the AEMU, and there is currently no concept of operations indicating a need within the AEMU. DTN from HOME to wherever could be considered, but there is no demonstrable need for multi-hop store, carry and forward regarding any AEMU scenarios - particularly if no terrestrial EVA s are expected in the foreseeable future.

All IP packets are forwarded on-suit or off-suit according to the forwarding (route) tables. Application that utilize IP addressing should operate without any issues assuming sufficient bandwidth.

There will be a handful of pre-defined Call Groups (a.k.a. voice loops) that the crew members can switch between during a sortie. Communication will translate these loops into IP addresses and transmit the packets accordingly. Those voice loops will be defined by mission control and can be modified at any time. Currently, the plan is to have Call Groups defined in a configuration file. Where the file resides is To Be Determined (TBD). For PAS 1.0, that file will be stored in the Communication system.

Potential call groups include:

- Local ONLY - No voice leaves the suit. This is for voice commands and audio message stamping of experimental data.
- Broadcast - All parties involved including other AEMU's HOME, and mission control
- Proximity Crew Only (voice between AEMU crew only...However, this may not be a permitted mode of operations.)
- Mission Ops Only (voice between the crew member and mission ops, which excludes other AEMU crew)

Audio sends its outbound streaming voice data to Communication via unicast IP packets. Communication shall re-transmit to other external nodes according to route tables and call group settings.

Caution and warning report messages from CWCS shall be published (transmitted) off-suit to the CWCS caution and warning subscribers. These message should come directly from CWCS . The report message will include a Caution or Warning time stamp, a Caution or Warning type, a flag to discriminate between caution or warning, and a flag indicating if this Caution or Warning has been acknowledged.

8.3.8. Research

Technology related to multi-home radios and routing is likely to move ahead within the commercial arena at a far greater pace and with far greater investment than NASA can provide. Such technology already exists in smartphones

and is likely to become standard in vehicle-to-vehicle and vehicle-to-infrastructure communication. Thus, no new research is currently planned related to multi-home dual-band radio wireless and network interface standards.

8.3.9. Time Synchronization

In PAS 1.0, the Communication subsystem is responsible for supporting time synchronization with the other subsystems. The current implementation is running Network Time Protocol (NTP) [23] on all subsystems. As timing requirements do not call for high precision, NTP should be sufficient.

During the PAS Interface Testing of August 2013, initial NTP testing showed NTP worked, but not without issues. NTP operates under the assumption that higher tiered clock data is available to the lower-tiered clocks for long periods of time [23]. Over time, the NTP slave devices slowly begin to "trust" the more stable master clocks based on their observed performance, and the slave devices will eventually begin to more aggressively synchronize with these clocks. Since the suit electronics are only powered up for several hours at a time, this process is too long and involved for this application. The behavior of NTP can be and was modified to suit the needs of PAS, but most of these modifications simply disabled a lot of the advanced functionality of NTP. Periodic, forced synchronizations with the Communication timestamp might be sufficient to maintain clock synchronization between the PAS assemblies over the duration of an EVA, and the same is true between Communication and its higher-tier clock (HOME, Mission Control, etc.).

8.3.10. Navigation

Navigation research related to AEMU development and EVA's is mainly concerned with terrestrial navigation for both safety and to be able to mark locations during science EVAs (i.e. where and when was a particular sample taken).

In preparation for the Constellation Program, NASA was interested in finding new methods of surface navigation to allow astronauts to navigate on the lunar surface. The interest is still there, but in more general terms than terrestrial navigation.

In support of the Vision for Space Exploration, the NASA GRC developed the Lunar Extra-Vehicular Activity Crewmember Location Determination System and performed testing at the Desert Research and Technology Studies event in 2009. A significant amount of sensor data was recorded during nine tests performed with six test subjects. [24]

Technology related to terrestrial navigation is likely to move ahead within the commercial and military arena at a far greater pace and with far greater investment than NASA can provide. Thus, no new research is currently planned as no terrestrial EVAs are expected any time soon.

8.4. Informatics

The Informatics assembly is a computerized system aiding crewmembers in their exploration missions. The operational goal is to increase EVA crewmember productivity and effectiveness and decrease the amount of mission support provided by ground-based controllers.

Functionally, the Informatics assembly provides information to be displayed for real-time navigation and tracking, along with productivity and work enhancement aids such as task procedures, checklists, and schedule coordination. It also displays situational awareness information, such as physiological data and consumable usage.

The Informatics subsystem performs only non-critical functions and therefore is assigned a Criticality 3 designation. Associated software is Class C. Failures within these will not result in a loss of life nor loss of mission. If this subsystem is down, the crew could continue to perform an EVA, but might not be as productive.

8.4.1. Implementation

For the tabletop breadboard, Informatics was implemented using a Beaglebone Black development board (`http://beagleboard.org/Products/BeagleBone+Black`). Some of the salient characteristics are:

- TI OMAP processor
- Linux Operating System
- 1 GHz ARM 7 architecture
- 512 MB RAM
- PowerVR 3D GPU
- File system on micro-SD card (16 GB currently)

8.5. Storage

A separate storage unit is not part of the Power, Avionic and Software 1.0 implementation. In PAS 1.0 all subsystems are responsible for providing their own storage.

The Storage Assembly is a non-volatile mass storage for the entire PAS Subsystem. This assembly is considered a secondary storage unit and not directly accessible by each of the assembly's processors; rather, each assembly will access it through their input/output channels via the subsystem bus. The Storage Assembly is considered to be all of the associated components that retain the digital data for use by the user and data stored by the PAS Subsystem for archiving.

The storage subsystem is assigned a Criticality 3 designation and associated software is assumed to be Class C. Failures within these will not result in a loss of life nor loss of mission. If these systems were down, the crew could continue to perform an EVA, but might not be as productive.

8.6. Video

The Video Camera is a self-contained unit. It is interfaced independently and directly to the PAS Subsystem data bus; therefore, the video data travels on the same bus as other the other PAS data. This interface requires physical, link, network, and transport layer Open System Interconnection (OSI) bus functions. Video imaging is the suit's highest data rate source, dwarfing most other suit data by a factor of 10 or more. The lens and sensor for the camera is expected to be located on either side of helmet with fixed field of view (FOV). Storage for video is expected to take place in Informatics for PAS 1.0 implementation. The Video Camera assembly is assumed to be a low criticality device.

For the PAS Interface Testing during August of 2013 [20], a commercial-off-the-shelf high definition webcam, a SANYO® VCC-HD4600, was used simply to load the GigE bus as much as possible. No storage was performed.

8.7. Monitoring, Test and Validation

The MTV system primarily monitors the GigE subsystem bus and the wireless links and creates the necessary signals to exercises various subsystems (e.g., subsystem configurations, multiple VOIP streams). The Validation system displays telemetry in a manner that can be used for debugging bus activity as well as validate test commands. The Test portion of MTV is a packet generation tool to exercise various subsystem commands over the GigE bus. This is particularly useful for debugging one subsystem independently of another.

For the PAS Interface Testing during August of 2013 [20], the MTV was actually distributed over a number of systems. Informatics implemented a number of event messages that would normally be sourced by other subsystems in order to test the Informatics display capabilities. Likewise Audio implemented four input streams using four netbooks. A separate netbook was used to monitor the Ethernet bus using Wireshark® software and for capturing and playing video from the HD Camera.

9. Intersystem Communications

9.1. Design Philosophy

Subsystem communications is performed via a Publish/Subscribe mechanism using the ZeroMQ software library. ZeroMQ also supports a Request/Response mechanism, but it was avoided because it requires maintaining additional software state and is more prone to race conditions, deadlock and other software complications.

Note, throughout this document that the term "command" is intentionally avoided as "command" often has a connotation of request/response communication.

We are using only ZeroMQ publish and subscribe sockets to move data packets between computers. The format of the data packets is defined using GPB in a common AEMU.proto file. Each message type has specific parameters associated with it.

The following are some important notes on ZeroMQ publish/subscribe messages (for a full explanation see: `http://zguide.zeromq.org/page:all`).:

- In theory with MQ sockets, it does not matter which end connects and which end binds. However, in practice there are undocumented differences. Generally, bind the PUB and connect the SUB.

Table 1: Required Fields

sender	integer	A number identifying the sender (1 for CWCS , 4 for Informatics, etc)
timestamp	double	The time the message was sent (Unix time plus fractional part, seconds since January 1, 1970)
type	integer	A number identifying the message type (which determines the additional fields in the message)

- When using PUB-SUB sockets, one does not know precisely when a subscriber starts to get messages. Even if you start a subscriber, wait a while, and then start the publisher, the subscriber will always miss the first messages that the publisher sends. This is because as the subscriber connects to the publisher (something that takes a small but non-zero time), the publisher may already be sending messages out. This is known as the "slow joiner" symptom.
- When using UDP sockets, a publisher does not put anything on the wire until someone subscribes. If you use multicast (e.g. PGM or Encapsulated Pragmatic General Multicast (EPGM)) the publisher will immediately put messages on the wire for anyone to receive.

9.2. Messages

Each message has the structure shown in Figure 3. The required fields are shown in Table 1. These required fields may be followed by various parameters, some which must be present and some which are optional depending on the particular message.

The following are the message types taken from the AEMU.proto file - see Appendix B. They are split into Implemented Messages and Future Work / Future Considerations. The latter are place holders for functionality that we currently believe requires a message for some controlling source and are thus, generally EVENT type messages. Often, the need for these messages depends on which subsystem owns and controls buttons and switches. If particular control buttons or switches belong to the subsystem, there is no need for configuration messages to be transmitted between subsystems. However telemetry "Status" messages could still be used by Informatics or CWCS to validate switch settings.

9.2.1. Implemented Messages

The following messages were implemented and tested during the Intersystem Communication Testing during August of 2013 [20]:

General Messages

EVENT_TEXT_MESSAGE
EVENT_BUDDY_INFO

CWCS Messages

TELEMETRY_ CWCS_STATUS
TELEMETRY_CWCS_CONSUMABLE_PO2
TELEMETRY_CWCS_CONSUMABLE_SO2
TELEMETRY_CWCS_CONSUMABLE_BATT
TELEMETRY_CWCS_CONSUMABLE_H2O
TELEMETRY_CWCS_CAUTIONWARNING
TELEMETRY_CWCS_PHYSIO
TELEMETRY_CWCS_BASICS
TELEMETRY_CWCS_TWO_LINE_DISPLAY

COMM Messages

TELEMETRY_COMM_STATUS

Audio Messages

TELEMETRY_AUDIO_STATUS
TELEMETRY_AUDIO_STATUS_OUTBOUND
TELEMETRY_AUDIO_STATUS_INBOUND
EVENT_PLAY_AUDIO
EVENT_AUDIO_SET_VOLUME_CHANNELS
EVENT_AUDIO_SET_VOLUME_TONES

INFO Messages

TELEMETRY_INFO_STATUS

An example, the details of and implementation code for the message EVENT_BUDDY_INFO is provided in Appendix A

9.2.2. Future Work / Desired Functionality

The following messages are for are functionality we believe need to exist, but are currently unimplemented:

COMM Messages

EVENT_COMM_CONFIG_REBOOT (Deprecated)
EVENT_COMM_CONFIG_UPDATE
EVENT_COMM_CONFIG_CRITICAL_RADIO
EVENT_COMM_CONFIG_HIGH_RATE_RADIO
EVENT_COMM_CONFIG_CALL_GROUP_RADIO

System and subsystem reboot message were considered. However, if a subsystem goes down, it probably cannot receive a reboot message. Rather, one would probably have to reboot via some type of power cycle. Thus, the notion of rebooting subsystems appears to be valid, but how to actually accomplish this has yet to be resolved.

The Communication system configuration consists of routing information and radio configuration. The radio configuration could include data rates, modulation formats, media access types (e.g. time-division-multiplexed or frequency-division-multiplexed) and perhaps a variety of waveforms. Who controls these waveforms and configurations is TBD. The initial thought is that this is a Mission Operations Center (MOC) function.

Call group configuration may be controlled by the MOC. Or, the MOC may configure the possible call groups with the crewmember controlling which call group they are participating in via a switch on the radio or via voice-commanding.

10. Summary

Some of the noteworthy items related to the PAS 1.0 Communication Architecture include the following:

A distributed architecture was used for two main reasons: to allow separation of software and hardware criticality and to allow subsystems to develop independent of each other. This was particularly useful for separation of research and technology development in the Audio and Communication subsystems while allowing the CWCS to be implemented with a nearer-term path to flight goal.

Use of a gigabit Ethernet bus enabled use of common off-the-shelf drivers and hardware greatly simplified subsystem communication integration.

Use of Google Protocol Buffers greatly simplified message structures while allowing flexibility and expandability in the various messages.

Use of ZeroMQ simplified communication between subsystems as we could concentrate on the message structure rather than the messaging implementation. Furthermore, ZeroMQ allows us to move from UDP transport to IPC or in-process shared memory without changing the message structure.

References

[1] K. Birman, T. Joseph, Exploiting virtual synchrony in distributed systems, Vol. 21, ACM, 1987.

[2] E. S. Division, Soldier Radio Waveform (SRW) 1.1.1 Waveform Design Specification (WDS), US NAVY SPAWAR SYSTEMS CENTER, North Charleston, SC 29419-9022, itt document 8245546 cdrl b001 revision e Edition (July 2012).

[3] etsi.org, Intelligent transportations system [cited 2013].
URL http://www.etsi.org/technologies-clusters/technologies/intelligent-transport

[4] NASA. Console handbook - extravehicular activity systems [online].

[5] Google protocol buffers.
URL https://developers.google.com/protocol-buffers/docs/overview

[6] P. Aimonen, Nanopb - protocol buffers with small code size.
URL http://koti.kapsi.fi/jpa/nanopb/

[7] D. A. Joseph, D. Chap, Auto-generate wireshark/ethereal dissector plugins for protocol buffer messages.
URL http://code.google.com/p/protobuf-wireshark/

[8] The intelligent transport layer [cited 13-July-2013].
URL http://www.zeromq.org/

[9] A. Dworak, M. Sobczak, F. Ehm, W. Sliwinski, P. Charrue, Middleware trends and market leaders 2011, Tech. rep. (2011).

[10] A. J. Swank, C. J. Bakula, Eva radio drats 2011 report.

[11] S. Cheshire, M. Krochmal, Multicast DNS, RFC 6762 (Proposed Standard) (Feb. 2013).
URL http://www.ietf.org/rfc/rfc6762.txt

[12] S. Cheshire, M. Krochmal, DNS-Based Service Discovery, RFC 6763 (Proposed Standard) (Feb. 2013).
URL http://www.ietf.org/rfc/rfc6763.txt

[13] Serval project - wiki [online] (2013 December) [cited December 2013].

[14] Bittorrent chat [online] (2013 December) [cited December 2013].

[15] The nsa files [online] (2013 December) [cited December 2013].

[16] J. Day, Patterns in network architecture: a return to fundamentals, Prentice Hall, 2007.

[17] O. of the Chief Engineer, Nasa software engineering requirements NPR7150.2A.

[18] Caution, warning, and control subsystem (cwcs) pre-delivery functional test plan and procedures, Tech. Rep. PAS-018, NASA Glenn Research Center (April 2013).

[19] gstreamer.freedesktop.org, Gstreamer open source multimedia framework [cited 2013].
URL http://gstreamer.freedesktop.org/

[20] W. Ivancic, O. S. Sands, C. J. Bakula, D. Oldham, T. Wright, M. Bradish, J. Klebau, Power avionics and software interface test report - september 2013, NASA Technical Memoradum TBD, NASA Glenn Research Center (February 2014).

[21] National Aeronautics and Space Administration,, Lyndon B. Johnson Space Center, Houston, Texas, Extravehicular Activities Space-To-Space Communication System Training Workbook (JSC-36345) (March 2000).

[22] U. D. of Transportation, Connected vehicle research [cited 2013].
URL http://www.its.dot.gov/connected_vehicle/connected_vehicle.htm

[23] D. Mills, J. Martin, J. Burbank, W. Kasch, Network Time Protocol Version 4: Protocol and Algorithms Specification, RFC 5905 (Proposed Standard) (Jun. 2010).
URL http://www.ietf.org/rfc/rfc5905.txt

[24] D. Chelmins, O. S. Sands, A. Swank, Data analysis techniques for a lunar surface navigation system testbed.

Appendix A. Example Message

The following is an example of the C language protocol buffer related code needed to send a typical message, the EVENT_BUDDY_INFO message.

This asynchronous message is intended to be sent off suit and should be subscribed to by all other interested parties - particularly other crew members. The intent is to allow crew members to see each others system status. In an operational system EVENT _BUDDY_INFO would be generated by CWCS . For the Interoperability Communication Testing of August 2013, this message was created by the Informatics system operating in the capacity of a message generator portion of the Test, Validation and Monitoring MOC.

For PAS 1.0, Informatics is the only subsystem that subscribes to EVENT_BUDDY_INFO. Upon receiving EVENT_BUDDY_INFO message, Informatics updates its displays and to sends an EVENT_PLAY_AUDIO message to Audio. This allows the Audio subsystem to play the proper caution and warning tones.

```
 AEMU_Message msg;
memset(&msg, 0, sizeof(AEMU_Message)); /* default all fields to false */

/* required fields */
msg.sender = AEMU_Message_Source_INFO;
msg.timestamp = timeNow();
msg.type = AEMU_Message_Type_EVENT_BUDDY_INFO;

/* fields needed for EVENT_BUDDY_INFO */
/* with nanopb, .has_XXX = true is needed for active fields tagged as optional */
msg.has_suitID = true;
msg.suitID = 4; /* int32, identifier to tell assets apart */
msg.has_po2TimeLeft = true;
msg.po2TimeLeft = 350; /* int32, primary oxygen time remaining in seconds */
msg.has_so2TimeLeft = true;
msg.so2TimeLeft = 5000; /* int32, secondary oxygen time remaining in seconds */
msg.has_battTimeLeft = true;
msg.battTimeLeft = 750; /* int32, battery time remaining in seconds */
msg.has_h2oTimeLeft = true;
msg.h2oTimeLeft = 4958; /* int42, water time remaining in seconds */
msg.has_latitude = true;
msg.latitude =  41.41029; /* float, suit position latitude in decimal degrees North */
msg.has_longitude = true;
msg.longitude = -81.86486; /* float, suit position longitude in decimal degrees East
    */
msg.has_altitude = true;
msg.altitude = 818.0; /* float suit position altitude in feet above sea level */

/* 0 to 7 caution and warning reports to send to buddies */
msg.cwReport_count = 1; /* number of caution and warning reports in this message */
msg.cwReport[0].has_savedTimestamp = true;
msg.cwReport[0].savedTimestamp = 13857834.9; /* double, time stamp of the caution or
    warning */
msg.cwReport[0].has_warningType = true;
msg.cwReport[0].warningType = AEMU_Message_WarningType_PO2_LOW;
msg.cwReport[0].has_isCaution = true;
msg.cwReport[0].isCaution = false; /* bool, true means caution, false means warning */
msg.cwReport[0].has_isAcked = true;
msg.cwReport[0].isAcked = false; /* bool, true means acknowledged, false means not
    acknowledged */
```

Appendix B. Google Protocol Buffers AEMU.proto file

```
package AEMU;

message Message {
  // required fields are kept to a minimum (3) because they must appear in every
      message

  enum Source {
    GEN = 0;          // general messages not associated with a subsystem
    CWCS = 1;         // Caution and Warning Control System
    COMM = 2;
    AUDIO = 3;
    INFO = 4;         // Informatics
    VIDEO = 5;        // placeholder
    STORAGE = 6;      // placeholder
    MTV = 9;          // monitor, test, validataion
  }
  required Source sender = 1;     // appears in every message

  required double timestamp = 2; // appears in every message; the time the message was
      sent [Unix time plus fractional part, seconds since January 1, 1970]

  enum Type {
    // GEN types range from 0-399 ------------------

    NOP = 0;                      // "No Operation"
    EVENT_TEXT_MESSAGE = 1;
    EVENT_BUDDY_INFO = 2;
    EVENT_LOCATION = 3;
    EVENT_DIRECTION = 4;

    // CWCS types range from 400-799 ---------------

    TELEMETRY_CWCS_STATUS = 400;

    TELEMETRY_CWCS_CONSUMABLE_PO2 = 401;
    TELEMETRY_CWCS_CONSUMABLE_SO2 = 402;
    TELEMETRY_CWCS_CONSUMABLE_BATT = 403;
    TELEMETRY_CWCS_CONSUMABLE_H2O = 404;

    TELEMETRY_CWCS_CAUTIONWARNING = 405;

    TELEMETRY_CWCS_PHYSIO = 406;

    TELEMETRY_CWCS_BASICS = 407;

    TELEMETRY_CWCS_TWO_LINE_DISPLAY = 408;

    // COMM types range from 800-1199 --------------

    TELEMETRY_COMM_STATUS = 800;
    EVENT_COMM_CONFIG_REBOOT = 801;
    EVENT_COMM_CONFIG_UPDATE = 802;
    EVENT_COMM_CONFIG_CRITICAL_RADIO = 803;
    EVENT_COMM_CONFIG_HIGH_RATE_RADIO = 804;
```

```
    EVENT_COMM_CONFIG_CALL_GROUP = 805;

    // AUDIO types range from 1200-1599 ---------------

    TELEMETRY_AUDIO_STATUS = 1200;
    TELEMETRY_AUDIO_STATUS_INBOUND = 1201;
    TELEMETRY_AUDIO_STATUS_OUTBOUND =1202;
    EVENT_PLAY_AUDIO = 1203;
    EVENT_AUDIO_SET_VOLUME_CHANNELS = 1204;
    EVENT_AUDIO_SET_VOLUME_TONES = 1205;

    // INFO types range from 1600-1999 ----------------

    TELEMETRY_INFO_STATUS = 1600;
  }
  required Type type = 3;     // appears in every message; determines which optional
      fields should be present

  // additional optional fields depend on type ------------------------------------

  // GEN types range from 10-399

  optional string broadcastMessage = 10; // used by type TEXT_MESSAGE_BROADCAST

  optional int32 genNodeID = 11;                    // used by type BUDDY_INFO_BROADCAST
      and TELEMETRY_COMM_STATUS; unique system ID
  optional int32 po2TimeLeft = 12;                  // used by type BUDDY_INFO_BROADCAST
      ; primary oxygen time left [seconds]
  optional int32 so2TimeLeft = 13;                  // used by type BUDDY_INFO_BROADCAST
      ; secondary oxygen time left [seconds]
  optional int32 battTimeLeft = 14;                 // used by type BUDDY_INFO_BROADCAST
      ; battery time left [seconds]
  optional int32 h2oTimeLeft = 15;                  // used by type BUDDY_INFO_BROADCAST
      ; water time left [seconds]
  optional float latitude = 16;                     // used by type BUDDY_INFO_BROADCAST
      and EVENT_LOCATION; position [decimal degrees, negative for South]
  optional float longitude = 17;                    // used by type BUDDY_INFO_BROADCAST
      and EVENT_LOCATION; position [decimal degrees, negative for West]
  optional float altitude = 18;                     // used by type BUDDY_INFO_BROADCAST
      and EVENT_LOCATION; position [meters above sea level]
  message CautionAndWarningReport {
    optional double savedTimestamp = 1;
    optional WarningType warningType = 2;
    optional bool isCaution = 3;
    optional bool isAcked = 4;
  }
  repeated CautionAndWarningReport cwReport = 19; // used by type BUDDY_INFO_BROADCAST

  optional float heading = 20;                      // used by type EVENT_DIRECTION;
      degrees from north (east is positive)

  // CWC starts at 400 - 799;

  enum Status {
    UNKNOWN = 0;
    NOMINAL = 1;
```

```
  // ...
}
optional Status status = 400; // used by type TELEMETRY_CWCS_STATUS and
    TELEMETRY_COMM_STATUS and TELEMETRY_AUDIO_STATUS and TELEMETRY_INFO_STATUS

optional int32 timeRemaining = 401;  // used by type TELEMETRY_CWCS_CONSUMABLE_PO2
    and TELEMETRY_CWCS_CONSUMABLE_SO2 and TELEMETRY_CWCS_CONSUMABLE_BATT and
    TELEMETRY_CWCS_CONSUMABLE_H2O; [seconds]
optional float currentPrimary = 402; // used by type TELEMETRY_CWCS_CONSUMABLE_PO2
    and TELEMETRY_CWCS_CONSUMABLE_SO2 and TELEMETRY_CWCS_CONSUMABLE_BATT and
    TELEMETRY_CWCS_CONSUMABLE_H2O; current amount
optional float maxPrimary = 403;     // used by type TELEMETRY_CWCS_CONSUMABLE_PO2
    and TELEMETRY_CWCS_CONSUMABLE_SO2 and TELEMETRY_CWCS_CONSUMABLE_BATT and
    TELEMETRY_CWCS_CONSUMABLE_H2O; maximum amount
optional float critPrimary = 404;    // used by type TELEMETRY_CWCS_CONSUMABLE_PO2
    and TELEMETRY_CWCS_CONSUMABLE_SO2 and TELEMETRY_CWCS_CONSUMABLE_BATT and
    TELEMETRY_CWCS_CONSUMABLE_H2O; critical value

enum WarningType {
  GENERIC = 0;
  PO2_LOW = 1;
  SO2_LOW = 2;
  BATT_LOW = 3;
  H2O_LOW = 4;
}
optional WarningType warningType = 405; // used by type
    TELEMETRY_CWCS_CAUTIONWARNING
optional bool isCaution = 406;          // used by type
    TELEMETRY_CWCS_CAUTIONWARNING
optional bool isAcked = 407;            // used by type
    TELEMETRY_CWCS_CAUTIONWARNING
optional string warningDetails = 408;   // used by type
    TELEMETRY_CWCS_CAUTIONWARNING

optional float metabolicRate = 409;     // used by type TELEMETRY_CWCS_PHYSIO; in
    BTU/hour
optional float heartRate = 410;         // used by type TELEMETRY_CWCS_PHYSIO; in
    beats/minute

enum CwcsMode {
  MODE_MANUAL = 0;
  MODE_EVA_43 = 2;
  MODE_EVA_60 = 3;
  MODE_EVA_82 = 4;
}
optional CwcsMode cwcsMode = 412;        // used by type TELEMETRY_CWCS_BASICS
optional float evaTime = 411;            // used by type TELEMETRY_CWCS_BASICS; time
    since start of EVA [seconds]
optional float suitPressure = 413;       // used by type TELEMETRY_CWCS_BASICS; [psi
    ]
optional float feedwaterPressure = 414;  // used by type TELEMETRY_CWCS_BASICS; [psi
    ]
optional float waterTemperature = 415;   // used by type TELEMETRY_CWCS_BASICS; [
    degrees F]
optional float batteryVoltage = 416;     // used by type TELEMETRY_CWCS_BASICS; [
    Volts DC]
```

```
  optional float batteryCurrent = 417;      // used by type TELEMETRY_CWCS_BASICS; [
      Amps]

  optional string twoLineDisplay1 = 418;    // used by type
      TELEMETRY_CWCS_TWO_LINE_DISPLAY
  optional string twoLineDisplay2 = 419;    // used by type
      TELEMETRY_CWCS_TWO_LINE_DISPLAY

  // COMM starts at 800

  optional int32  commOperationalStatus = 801;          // used by type
      TELEMETRY_COMM_STATUS; everything within range
  optional string commFirmware = 802;                   // used by type
      TELEMETRY_COMM_STATUS; Version Info
  optional string commSoftware = 803;                   // used by type
      TELEMETRY_COMM_STATUS; Version Info
  optional string commWaveform = 804;                   // used by type
      TELEMETRY_COMM_STATUS; Version Info
  optional bool   commExtTimeSync = 805;                // used by type
      TELEMETRY_COMM_STATUS; COMM time synced to external source YES = 1, NO = 0
  optional int32  commSysHardwareFailure = 806;         // used by type
      TELEMETRY_COMM_STATUS; Zero, on failure. All other values are failure codes
  optional bool   commCriticalRadioOn = 812;            // used by type
      TELEMETRY_COMM_STATUS; ON =1, OFF = 0
  optional bool   commCriticalRadioSync =813;           // used by type
      TELEMETRY_COMM_STATUS; 1 = Synced, 0 = Loss of Sync
  optional bool   commCriticalRadioTracking =814;       // used by type
      TELEMETRY_COMM_STATUS; 1 = Locked, 0 = Loss of Lock
  optional int32  commCriticalRadioSignalStrength = 815; // used by type
      TELEMETRY_COMM_STATUS
  optional int32  commCriticalRadioDataRate = 816;      // used by type
      TELEMETRY_COMM_STATUS
  optional float  commCriticalRadioBerRate = 817;       // used by type
      TELEMETRY_COMM_STATUS
  optional float  commCriticalRadioSN = 818;            // used by type
      TELEMETRY_COMM_STATUS
  optional bool   commHighRateRadioOn = 822;            // used by type
      TELEMETRY_COMM_STATUS; ON =1, OFF = 0
  optional bool   commHighRateRadioSync =823;           // used by type
      TELEMETRY_COMM_STATUS; 1 = Synced, 0 = Loss of Sync
  optional bool   commHighRateRadioTracking =824;       // used by type
      TELEMETRY_COMM_STATUS; 1 = Locked, 0 = Loss of Lock
  optional int32  commHighRateSignalStrength = 825;     // used by type
      TELEMETRY_COMM_STATUS
  optional int32  commHighRateDataRate = 826;           // used by type
      TELEMETRY_COMM_STATUS
  optional float  commHighRateBerRate = 827;            // used by type
      TELEMETRY_COMM_STATUS
  optional float  commHighRateSN = 828;                 // used by type
      TELEMETRY_COMM_STATUS; perhaps should be Eb/No rather than S/N
  optional bool   commHighRateHardwareFailure = 829;    // used by type
      TELEMETRY_COMM_STATUS; Failure = 1

  // Routing Info
  enum CallGroup {
    LOCAL_ONLY = 0;         // No voice leaves suit
```

```
  BROADCAST = 1;          // All parties including mission ops, HOME and other AEMUs
  PROXIMITY = 2;          // All parties not including mission ops (e.g., HOME and
      other AEMUs)
  MISSION_OPS_ONLY = 3; // Peer session with Mission Ops, no other parties
}
optional CallGroup commCallGroup = 850;     // used by type
    EVENT_COMM_CONFIG_CALL_GROUP

optional string commConfigUpdate = 852;     // used by type EVENT_COMM_CONFIG_UPDATE;
     configuration file to load

optional bool commSetCritalRadioOn = 853;  // used by type
    EVENT_COMM_CONFIG_CRITICAL_RADIO; true enables radio, false disables radio

optional bool commSetHighRateRadioOn = 854; // used by type
    EVENT_COMM_CONFIG_HIGH_RATE_RADIO; true enables radio, false disables radio

// AUDIO starts at 1200;

optional int32 audioOperationStatus = 1201; // used by type TELEMETRY_AUDIO_STATUS;
    zero means everything within range, other numbers have other meanings
optional string audioFirmware = 1202;        // used by type TELEMETRY_AUDIO_STATUS;
    Version Info
optional string audioSoftware = 1203;        // used by type TELEMETRY_AUDIO_STATUS;
    Version Info
optional int32 audioHardwareFailure = 1204; // used by type TELEMETRY_AUDIO_STATUS;
    Zero, on failure, all other values are failure codes

enum AudioMode {
  PTT = 0;       // PTT switch is operational and take precidance
  VOX = 1;       // Voice is sent relative to Voice Activation Detection (VAD)
  OPEN_MIC = 2; // Voice is sent regardless of VAD (silence or breathing or whatever
      )
}
optional AudioMode audioMode = 1212; // used by type TELEMETRY_AUDIO_STATUS_OUTBOUND
optional float audioGain = 1210;      // used by type TELEMETRY_AUDIO_STATUS_OUTBOUND
    ; pressure compensated microphone gain [dB]
optional bool  audioPTT = 1213;       // used by type TELEMETRY_AUDIO_STATUS_OUTBOUND
    ; OFF = 0, ON = 1
optional float audioVolume = 1214;    // used by type TELEMETRY_AUDIO_STATUS_OUTBOUND
    ; 0 - 10 for gstreamer
optional float audioOVDC = 1215;      // used by type TELEMETRY_AUDIO_STATUS_OUTBOUND
    ; output voice duty cycle

optional float audioVolumeChannel0 = 1220; // used by type
    TELEMETRY_AUDIO_STATUS_INBOUND
optional bool  audioMuteChannel0 = 1221;   // used by type
    TELEMETRY_AUDIO_STATUS_INBOUND
optional float audioVolumeChannel1 = 1222; // used by type
    TELEMETRY_AUDIO_STATUS_INBOUND
optional bool  audioMuteChannel1 = 1223;   // used by type
    TELEMETRY_AUDIO_STATUS_INBOUND
optional float audioVolumeChannel2 = 1224; // used by type
    TELEMETRY_AUDIO_STATUS_INBOUND
optional bool  audioMuteChannel2 = 1225;   // used by type
    TELEMETRY_AUDIO_STATUS_INBOUND
```

```
  optional float audioVolumeChannel3 = 1226; // used by type
      TELEMETRY_AUDIO_STATUS_INBOUND
  optional bool  audioMuteChannel3 = 1227;   // used by type
      TELEMETRY_AUDIO_STATUS_INBOUND
  optional float audioVolumeChannel4 = 1228; // used by type
      TELEMETRY_AUDIO_STATUS_INBOUND
  optional bool  audioMuteChannel4 = 1229;   // used by type
      TELEMETRY_AUDIO_STATUS_INBOUND
  optional float audioVolumeTones = 1250;    // used by type
      TELEMETRY_AUDIO_STATUS_INBOUND
  optional bool  audioMuteTones = 1251;      // used by type
      TELEMETRY_AUDIO_STATUS_INBOUND

  optional float audioSetVolumeChannel0 = 1230; // used by type
      EVENT_AUDIO_SET_VOLUME_CHANNELS
  optional bool  audioSetMuteChannel0 = 1231;   // used by type
      EVENT_AUDIO_SET_VOLUME_CHANNELS
  optional float audioSetVolumeChannel1 = 1232; // used by type
      EVENT_AUDIO_SET_VOLUME_CHANNELS
  optional bool  audioSetMuteChannel1 = 1233;   // used by type
      EVENT_AUDIO_SET_VOLUME_CHANNELS
  optional float audioSetVolumeChannel2 = 1234; // used by type
      EVENT_AUDIO_SET_VOLUME_CHANNELS
  optional bool  audioSetMuteChannel2 = 1235;   // used by type
      EVENT_AUDIO_SET_VOLUME_CHANNELS
  optional float audioSetVolumeChannel3 = 1236; // used by type
      EVENT_AUDIO_SET_VOLUME_CHANNELS
  optional bool  audioSetMuteChannel3 = 1237;   // used by type
      EVENT_AUDIO_SET_VOLUME_CHANNELS
  optional float audioSetVolumeChannel4 = 1238; // used by type
      EVENT_AUDIO_SET_VOLUME_CHANNELS
  optional bool  audioSetMuteChannel4 = 1239;   // used by type
      EVENT_AUDIO_SET_VOLUME_CHANNELS

  optional float audioSetVolumeTones = 1252;    // used by type
      EVENT_AUDIO_SET_VOLUME_TONES
  optional bool  audioSetMuteTones = 1253;      // used by type
      EVENT_AUDIO_SET_VOLUME_TONES; probably never used, not allowed.

  // if only one sounds can play at a time, higher numbers have higher priority
  enum AudioPlaySound {
    AUDIO_TONE_INFORMATION = 0;
    AUDIO_TONE_STATUS = 1;
    AUDIO_TONE_ALERT = 2;
    AUDIO_TONE_WARNING = 3;
  }
  optional AudioPlaySound audioPlaySound = 1260; // used by type EVENT_PLAY_AUDIO
  optional bool audioPlayStart = 1261;           // used by type EVENT_PLAY_AUDIO
}
```

List of Acronyms

AEMU Advanced Extravehicular Mobility Unit

AES Advanced Exploration Systems

ASIC Application Specific Integrated Circuit

API Application Program Interface

CDN Content Delivery Network

CRC Cyclical Redundancy Check

CWC Caution, Warning and Control

CWCS Caution, Warning and Control System

DCM Display and Control Module

DRATS Desert Research and Technology Studies

DTN Delay/Disruption/Disconnection Tolerant Networking

EMU Extravehicular Mobility Unit

EPGM Encapsulated Pragmatic General Multicast

EVA Extravehicular Activity

GigE Gigabit Ethernet

GPB Google Protocol Buffers

GRC Glenn Research Center

ICNRG Information Centric Networking Research Group

IRTF Internet Research Task Force

ISS International Space Station

IP Internet Protocol

IPC Interprocess Communication

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

ITS Intelligent Transportation Services

JSC Johnson Space Center

LCD Liquid Crystal Display

LEO Low-Earth Orbit

LPCM Linear Pulse Code Modulation

MAC Media Access Control

MANET Mobile Ad Hoc Network

MOC Mission Operations Center

MTV Measurement, Test and Validation

NASA National Aeronautics and Space Administration

NTP Network Time Protocol

OSI Open System Interconnection

P2P Point-to-Point

PCI Peripheral Component Interconnect

PAS Power, Avionics and Software

PDR Preliminary Design Review

PGM Pragmatic General Multicast

PLSS Primary Life Support Subsystem

PMAD Power, Management and Distribution

PTT Push-To-Talk

RAM Random Access Memory

RF Radio Frequency

ROM Read Only Memory

RPC Remote Procedure Call

SSER Space-to-Space EMU Radio

STS Space Transportation System

SWaP Size, Weight and Power

TBD To Be Determined

TCP Transmission Control Protocol

UDP User Datagram Protocol

UHF Ultra High Frequency

VAD Voice Activity Detection

VOX Voice Operated Switch

VOIP Voice-Over-IP

WAN Wide Area Network

WLAN Wireless Local Area Network