

* A Message Transfer Service for Space Applications

Peter Shames

JPL/CalTech, 4800 Oak Grove, MS 301-265, Pasadena, CA 91109 USA
Tel. +1 (818) 354-5740, Fax +1 (818) 393-1333, e-mail: Peter.Shames@jpl.nasa.gov

ABSTRACT

Spacecraft applications that interact with on-board sensors, effectors, and major components have traditionally used private interfaces which are tightly bound to the interface details of the data links connecting these sub-systems. The latest spacecraft, which often include several powerful processors running real time operating systems, high-speed on-board networks, and intelligent peripherals, can support a more layered networked environment. Use of standard interfaces and networked elements is expected to yield reusable software and hardware components. Within CCSDS Panel 1K we have been developing such on-board interface standards. This paper will present a Message Transfer Service that defines a common API for use within a single spacecraft or among spacecraft flying in a constellation.

CCSDS PANEL 1K INTRODUCTION

The Consultative Committee on Space Data Systems (CCSDS) is an international volunteer organization of space agencies and industrial associates interested in mutually developing standard data handling techniques to support space research, including space science and applications. It has, over the years, published a series of standard recommendations for space communications, starting with physical layer modulation and coding, and including link layer and reliable data delivery over near Earth and Deep Space communication distances. See <http://ccsds.org> for details of the entire program and standards suite. Recent efforts have been focussed on providing Internet-like connectivity for mission operating in deep space and on providing multi-point connectivity among in-situ elements and spacecraft flying in constellations or other formations. Since missions are becoming increasingly interdependent, with one agency's rover using another's orbiting spacecraft to communicate to Earth through even a third agency's ground station, interoperability and cross-support, the foundations of CCSDS agreements over the years, have become even more important.

CCSDS Panel 1K is specifically focused on defining standard interfaces that are suitable for use on board spacecraft. The context of the Spacecraft Onboard Interface (SOIF) is the communications bus (network) onboard the spacecraft, and the elements connected to such busses [SOIF]. It does not necessarily include ground, spacelink, or special onboard interfaces where they are required, but instead seeks to define interface approaches to meet the broadest set of common instrument, sensor, and effector requirements. The scope includes all types of devices from large instruments to small engineering sensors like a temperature sensor, and all types of spacecraft, from micro and small to large and/or manned, from LEO to deep space. See Figure 1, SOIF Context.

The intent of SOIF is to define several levels of interfaces, which can be adopted, as desired, by compliant missions. These interface standards, at the physical, network, and applications layers provide services that permit device and application level portability, and thus promote reuse, risk reduction, and cost savings. Where many current missions develop applications that interact with devices directly at the data link layer, SOIF recommends use of a "convergence layer" that isolates applications from the vagaries of individual data links. This permits

* This work was performed at and under the direction of the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration. Copyright © 2002 by the California Institute of Technology. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. Released to AIAA to publish in all forms

application portability and also supports the exchange of one data link for another (perhaps of higher capacity) without requiring their re-implementation. This paper describes one of these proposed interface, a Message Transfer Service (MTS) suitable for use in space applications.

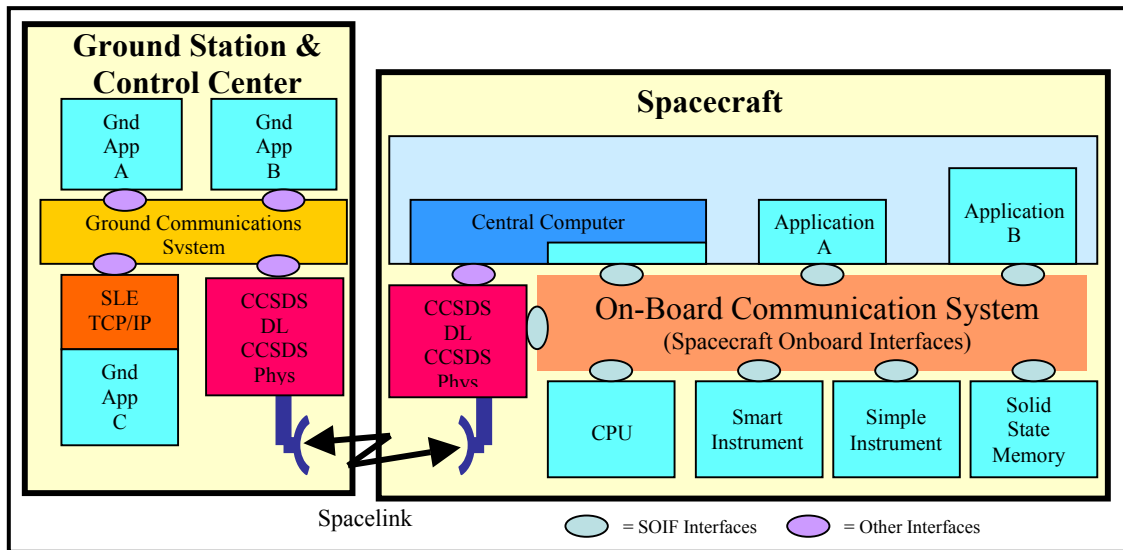


Figure 1: SOIF Context

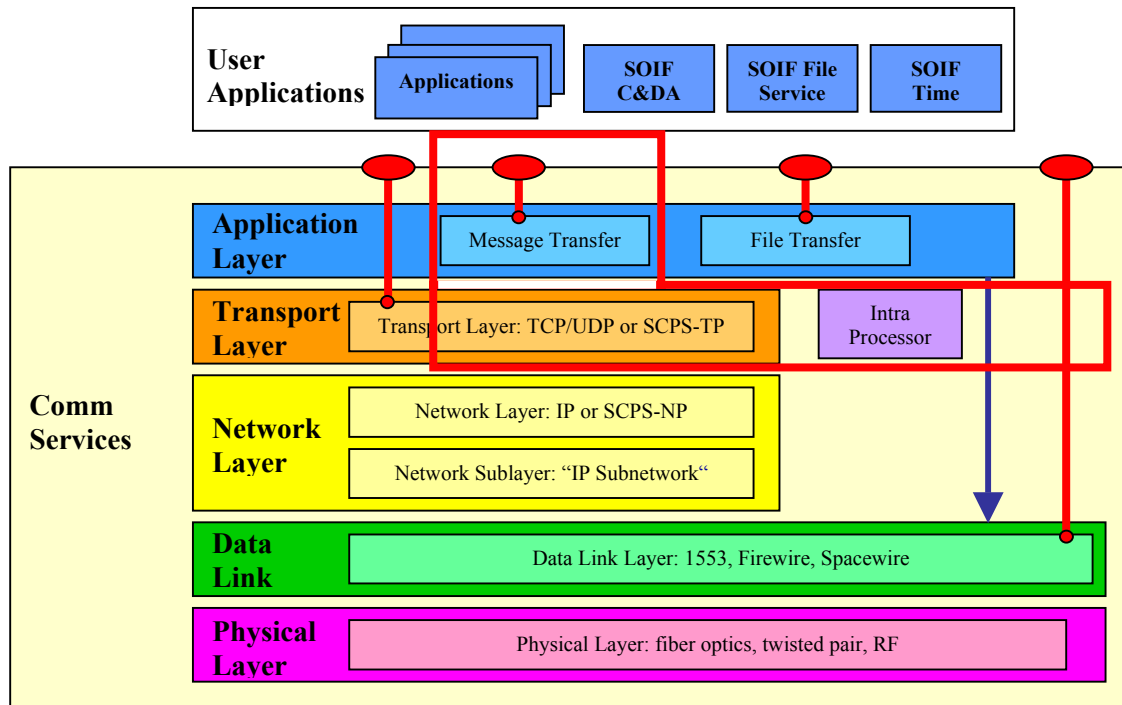
SOIF APPLICATION SERVICES

The SOIF Reference Model [SREF] assumes that spacecraft applications will have to operate over a variety of different underlying buss capabilities and topologies. These busses are chosen based upon a number of factors: performance, reliability, latency, robustness, fail-safe behavior, redundancy, mass, and power. Some spacecraft are “single string” with a single computer and buss and no inherent redundancy. Others use multiple busses, each of which may be block redundant and have dual paths. Many spacecraft applications need not concern themselves with this underlying physical complexity, their concern is really the analysis of spacecraft trajectory and attitude, the planning of observations and the processing of resulting information, or the management of data and transmission of results to end users. Such applications need these underlying communications services to work, but need not concern themselves with the details of how this is carried out.

Another observation about modern spacecraft is that many of them are now quite advanced in computing power and functionality. They may include several powerful processors running real time operating systems, mass memories that emulate disk storage, high speed on-board networks, intelligent peripherals, and can support a quite complete networked environment. The implementation model chosen for the SOIF convergence layer is to adopt the TCP/IP suite, where that is possible, as a direct means of providing both high connectivity, robustness, and handling traffic flow across a variety link protocols. See Figure 2 for a view of the implementation approach.

The same functionality in TCP/IP that provides routing across multiple busses within a single spacecraft can also support routing of data among multiple spacecraft. This will be feasible where the assumptions re continuous connectivity, low latency, and interactivity inherent in the Internet protocols, and embedded in their design, are not violated. This becomes interesting for providing communications among several spacecraft that are flying in formation or among several spacecraft in a constellation. In the latter case continuous connectivity may be problematic due to orbital geometries, but as long as the contact periods are reasonably long the benefits of buying a network communications solution instead of building one are expected to hold. For these operating environments a suitable space link protocol like Proximity-1 [PROX] may be used below TCP/IP.

SOIF defines service interfaces for user applications at a couple of different levels, depending upon the application requirements. At the highest level are message transfer and file transfer services. File transfer is expected to be performed using the CCSDS File Delivery Protocol (CFDP), and it is anticipated that this will be used where bulk data is to be transmitted over intermittently available long latency links [CFDP]. Message transfer is the other high



NOTE: For simplicity no security protocol is shown

Figure 2: SOIF Implementation Model

level service and it is expected to be used on-board a single spacecraft or among nearby spacecraft where the connectivity is nearly continuous and the round trip light time (RTLTL) latencies are modest (<0.25 sec). In addition to providing both message and file transfer interfaces, SOIF also permits direct use of the TCP/IP socket interface and even of low level data link interfaces.

The use of a single message transfer service interface hides network/interface differences from higher application functions. When this standard interface is used, the application can use all the services provided by the protocol, otherwise the applications either need to supply the service themselves (e.g. reliability, routing, multi-link transparent transport), or do without the service. Furthermore, without using this standard interface, when the underlying bus changes, then the effects will ripple up into the applications. Some applications, particularly those that directly interface with low level sensors or effectors, might require direct link level access, either due to latency or the need for direct control of some critical interface such as a pyro device or propulsion control valve. However, for most applications the use of these higher level interfaces will provide significant benefits in terms of portability, functionality, and isolation from changes at the physical level.

The team that has been developing this specification includes representatives from several different space agencies and supporting organizations. It includes people with backgrounds that include experience with spacecraft flight system and ground system design and development, development and integration of flight and ground hardware, and design and development of distributed data systems. The Messaging Special Interest Group (SIG) that has

developed the API described in the rest of this paper used a spiral development approach that included production of three independent prototype implementations. What is reported here is the result this prototyping, the analysis of the results, and the integrated effort of these three teams.

MESSAGE TRANSFER SERVICE DEFINITION

A set of common functions for Message Transfer and Level of Service (LoS) management that sit above the transport layer in the communications stack and provide a service API for mediating the transfer of data between processes in a distributed system.

Assumptions:

- The processes run on a single system or multiple systems that may be heterogeneous
- The network can be LAN, WAN (Wire or RF) and Internet
- The Transport Layer functionality may be provided by TCP/IP or SCPS, by a direct Data Link, or by efficient mechanisms such as shared memory, pipes, or other IPC

The SOIF Message Transfer service has the following requirements:

- Transfer of information of arbitrary structure between processes over point to point or multi-point connection
- Control and specification of the level of service and service delivery mechanisms (timeliness, reliability, throughput, determinism, reporting, policies, transfer services)
- Responsive control, monitoring and reporting of the service and/or connections
- Application portability and interoperability across heterogeneous implementations

And the following high level implementation goals:

- Low overhead
- Low latency
- Small “footprint”
- Thread – safe

These implementation goals are aimed at producing an interface that provides the required functionality and yet is suitable for use within a spacecraft environment where resources, though more abundant than in the past, are still very much constrained.

MESSAGE TRANSFER SERVICE FUNCTIONALITY

The SOIF Message Transfer Service (MTS) is intended to provide a high functionality set of service for the transfer of messages of arbitrary structure among processes that reside on one or more “local” processors. The interface is intended to isolate application processes from the need to deal with any of the vagaries of underlying transport mechanisms, and yet provide means for applications to specify the level of service required to meet their needs.

We have defined a simple, yet rich, interface that provides “enough” functionality to meet a broad range of real time and non real time application requirements. The interface supports both peer to peer and client server interaction models, and provides primitives that can be used to construct basic publish / subscribe interaction pattern if such is desired. The basic elements of this include: name and address resolution, connection establishment, synchronous and asynchronous message transfer mechanisms, and synchronization mechanisms for cooperating processes. Mechanisms are also provided to enable discovery of available level of service (LoS), to

select LoS, and for error reporting and service monitoring suitable for operation in a space environment. See Figure 3 for an example of how the SOIF MTS may be used within a single spacecraft.

Prior to initiating this design effort a study was made of a number of different existing message transfer interfaces to determine if any of them were directly suitable for adoption. Nine different APIs were studied, and while some of them had a number of characteristics that were relevant none of the identified candidates appeared to be directly applicable [SAPI]. Most of the messaging APIs have been designed to operate in a transaction oriented environment, where primary emphasis is on reliability and guaranteed delivery. Timeliness and throughput are typically secondary considerations and implementation “footprint” is also not high on the list. Many of these interfaces used a store and forward approach, which introduces queuing and other delays. For our applications the situation is almost reversed, throughput, timeliness, and a small implementation footprint are primary.

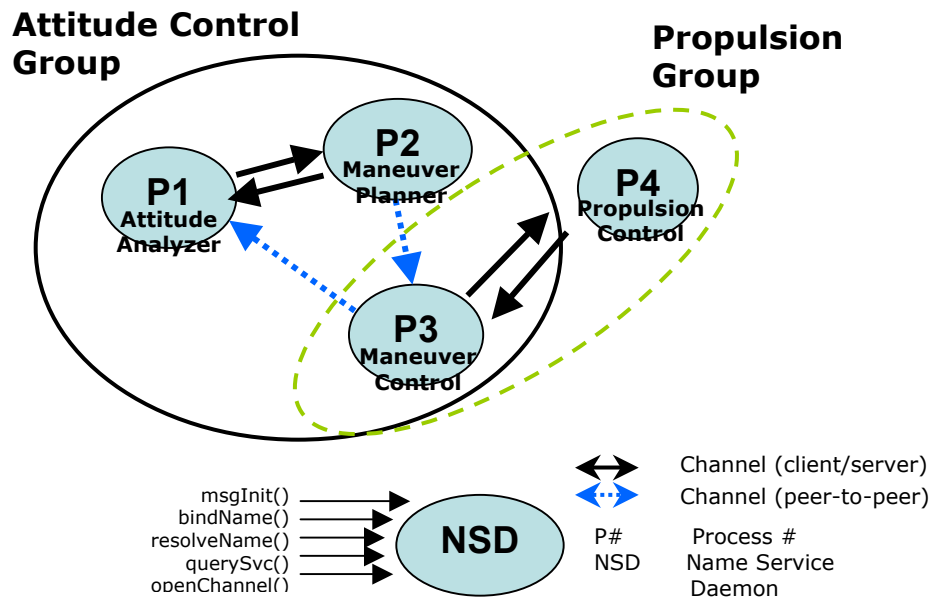


Figure 3: Example of MTS Use

The Message Transfer Service includes several groups of functions that provide different kinds of services to applications. At a high level these include:

Communication Establishment Mechanisms

- Service initiation / termination, authentication
- Name space lookup / management

Process - Process Communication

- Send/receive (two-sided), put/get (one-sided), blocking/non-blocking
- Multicast / Broadcast - specified set or promiscuous recipients
- Publish/subscribe - managed delivery of data
- Callback - event notification and handling

Level of Service

- Service & connection negotiation, policy management
- Priority/scheduling/resource management
- Reliability, determinism, ordering, timeliness

Control

- Message buffer management (make/free/get/set/init)
- Error Handling, monitoring, signaling, reporting
- Group management (joingroup/leavegroup)

We drew from our requirements study, analyses of available interfaces, and from team member experience in selecting the functions that we believed were highest priority within the interface. The desire to meet requirements for real time performance, low latency and low timing jitter, and the assumption that we would primarily be operating in a connected, “interactive” environment, directed an implementation approach that excluded use of store and forward queues. We included a number of LoS capabilities, but further analysis of the functions which are expected to be most useful in a spacecraft environment, vs. their implementation costs and overhead, suggests that these may not be required. Many of the LoS requirements are typically handled directly as part of the system engineering of the communications system as a whole. As a result we elected not to expose these interfaces within the API itself. This choice will be reviewed again during the next design cycle.

MTS PROTOTYPE ACTIVITIES

Three different teams developed independent prototypes that exhibited the functionality called for in the draft specification. The initial prototyping efforts that were carried out to explore feasibility, functionality, performance and footprint did not include all of the defined functions. Excluded were publish/subscribe, multi-cast and group functions, and all of the LoS elements except for selection of reliable/unreliable communication modes. Both blocking and non-blocking communication approaches were included [PROT].

The three prototype teams took rather different approaches to providing the functionality, but developed remarkably similar APIs which exhibited nearly identical performance, except in the “footprint” of the implementation. All of the prototypes were developed on two or more PCs running Linux or FreeBSD, using Ethernet for the data link. An agreed series of tests were run, using different interaction patterns and message lengths to represent typical S/C operating environments.

The team from Logica developed an API based upon some work that they had initially done for a prototype SOIF Command and Data Acquisition (C&DA) interface. They had initially set out to demonstrate how to provide a C&DA function for simple devices, and in doing so derived a need for a message passing interface. This interface, based directly on use of UDP over a standard sockets, formed the basis of their MTS prototype. This approach provided the highest performance and the smallest footprint.

Science Systems Ltd developed their API based upon work that they had done in developing a minimal CORBA ORB for space use [CORB]. The functions included in the General InterORB Protocol (GIOP) layer are very similar to the capabilities defined in our specification. This team did a comparison of their GIOP implementation to the agreed specifications and developed a simple prototype using a CORBA IDL interface that performed the agreed tests. GIOP is itself built over TCP sockets. This approach provided the second smallest footprint, but had the slowest performance due to the added layers of protocol overhead and the use of TCP.

The JPL team implemented an API that borrowed elements from the Message Passing Interface (MPI) real time specification, which appeared to be the best of those in the study. The implementation approach chosen was to use the Asynchronous Communications Environment (ACE) toolkit, which provides high functionality and a wide variety of programming patterns [ACE]. This implementation produced the second fastest performance, but the worst footprint by a factor of more than one hundred because of the construction of the ACE libraries.

The results of these prototype activities were extensively analyzed by the members of the Messaging SIG during the spring 2002 CCSDS meetings and the three different APIs were compared. While there were distinct

differences among the three APIs they turned out to be more similar than different. The Messaging SIG developed an integrated API that drew from the results of these prototype efforts. This has been further refined during subsequent telecons and is now ready for the second phase of prototyping and validation.

DRAFT MESSAGE TRANSFER SERVICE API

The draft MTS API only includes the basic functions for communication that we identified in the prototyping efforts. This was done to bound the scope of the effort in order to prove out essential functionality. The selection of included functionality is based on the understanding that many of the issues re Level of Service (LoS) will be handled during the system engineering of the S/C communications system. Many of the implementation choices that are made during S/C design place bounds and constraints upon what can be achieved re latency, throughput, timing jitter, reliability, and overhead. For example, use of the well known IEEE 1553 data link offers easy guarantees on timing jitter and reliability, but at the cost of throughput and overhead for large data transfers [1553]. Conversely, use of Firewire in asynchronous mode (IEEE 1394) [FIRE], Spacewire (variant of IEEE 1355) [SPWR], or even Ethernet can provide much higher throughput, but at the cost of greater uncertainties in timing jitter.

The selection of one or more busses, and their characteristics, are engineering decisions that constrain the possible performance and service envelopes. Rather than complicate the interface at this time with functions that attempt to provide control over LoS we have chosen instead to provide a simple set/query interface. This permits a process to identify which level of service it is capable of supporting and allows its peer to query this and select a compatible mode. Level of service can be viewed as a matrix that includes various dimensions: reliable / unreliable, acknowledged / unacknowledged, complete / best efforts, and interactive / store and forward. One dimension of quality of service may also include concepts like message priority, starting time, and deadline for transmission. We are only addressing reliable / unreliable LoS within the interface at this time.

Service Establishment

- InitializeMsgService – initialize the message service, identify the process to the daemon
- FinalizeMsgService – close the message service, remove association of process
- ResolveName – resolve name of peer entity to a process ID handle
- BindName – identify process to daemon and create process handle

Level of Service

- queryLoSCapability – request Level of Service (LoS) matrix for process handle
- setLoSCapability – establish available LoS matrix for process handle

Communication Mechanisms

- openChannel – open bi-directional channel between two processes, return channel handle
- closeChannel – close channel between two processes
- sendMessage – send a process-process message on channel, block on response
- sendMessagePoll – send a message on channel, poll for completion
- sendMessageAsynch – send a message on channel, identify handler for callback
- receiveMessage – receive a message on channel, block on completion
- receiveMessagePoll - receive a message on channel, poll for completion
- receiveMessageAsynch - receive a message on channel, identify handler for callback
- eventLoop – query communication status, used as polling function for I/O completion

The full specification of the API, including both syntax and semantics, is too voluminous to document in this paper. Please contact the author if you are interested in the entire specification or in getting a copy of the reference

implementation that is now being developed. The only LoS functionality included in this next prototype will be reliable / unreliable, which maps to a selection between TCP and UDP. The prototype will also include a mapping to an efficient shared memory queue for high performance intra-processor communication.

NEXT STEPS

This design and specification activity was initiated within the CCSDS Panel 1K organization and we expect to continue to work within that venue. This means that the ultimate definition of this interface will include inputs from a number of different sources, thus helping to ensure that it is suitable for use in a broad range of environments. The next round of prototype efforts is expected to include two or more teams and it will also include, at that point, interoperability testing between the independent implementations. The evolved specification will include an API, with syntax and semantics, and specifications of control message formats and protocol actions.

It is likely that these prototypes of the API will be further validated by exploring their use with two or more different message structure approaches that are presently in use or being developed. At a minimum these are expected to include CCSDS packets (including the ESA Packet utilization Standard (PUS) variant) [PUS], the JPL Mission Data System (MDS) goal nets and state reports [MDS], and possibly some message structures based upon XML schema [XML]. These further explorations are expected to validate the functionality of the interface in what are representative S/C operating scenarios.

Driven by a desire to have a common message transfer service that can operate in “connected” environments, where an interactive communication style is possible, and in intermittently connected environments, where a delay tolerant style is required, we will re-evaluate the API spec. Many spacecraft, even in near Earth or constellation environments, experience intermittent disconnection of their peer communication links due to purely physical events such as orbital geometry, planetary occlusion, or observational activities that require pointing changes. In Deep Space communication delays may be due to any of these events, but are primarily due to the long RTLTs that characterize communications between planets. In order to handle these events it is at least necessary to ensure that the protocol stack appropriately deals with the outages.

It is clear that introducing some sort of store and forward capability will help in dealing with these link outages and still provide reliable, acknowledged, though not particularly timely, communication. One of the additional studies will explore what sorts of underlying data transfer mechanisms might be appropriate for these operating environments and how to integrate them into the service interface. Our initial analysis suggests that it may be possible to utilize the CCSDS File Delivery Protocol (CFDP), in either acknowledged or unacknowledged modes, as required to provide this quality of transfer service. The LoS matrix would include a new row for delay tolerant communication, which would distinguish reliable and unreliable modes.

SUMMARY

At this point the SOIF Messaging SIG has developed a high level agreement on requirements, goals, capabilities, and functionality of a message transfer service that is suitable for space use. We have created three different prototypes to evaluate functionality, performance, footprint, and ease of use. We prototyped three potential approaches, compared the results of these approaches, and identified commonalities and differences. The draft Phase 1 API, which is based upon the results of this prototype and evaluation activity, is now undergoing review and the next round of prototype activities is underway. The results of this will be further verified by interoperability testing and validated by applying it in real world S/C operating scenarios. Extensions to the service to support both interactive and delay tolerant LoS will be explored.

We expect that the development of a common and agreed Message Transfer Service interface will be a useful addition to the spacecraft on-board interface (SOIF) communications approach. It is expected to permit

development of applications that can be ported across different spacecraft, operating platforms, and underlying communications stacks. These communications stacks at a minimum will include the TCP/IP protocol suite, direct interfaces to low level data links, and high performance intra-processor interfaces such as shared memory queues, pipes, or other IPC mechanisms. Adoption of these new spacecraft interfaces is expected to permit development of re-usable software elements and of re-usable hardware components.

ACKNOWLEDGEMENTS

The service interface described in the paper is the work of a team of individuals called the Messaging SIG. Primary contributors to the SIG were the three prototype teams, which were: Logica/BNSC/UK (Nick Achilleos, Ewan Carr, Gavin Kenny, Gary Lay), Science Systems Ltd/BNSC/UK (Stuart Fowell), NASA/JPL (Chia-lan Hwang, Laverne Hall, Peter Shames). Other contributors to the SIG working meetings and discussions included Ed Greville/NASA/GSFC, Norm Lamarra/NASA/JPL, Stuart Mills/U Dundee, Nick Rouquette/NASA/JPL, and Joe Smith/NASA/JPL. The rest of the CCSDS Panel 1K, including particularly Patrick Planke/ESA/ESTEC, Chris Plummer/ESA/ESTEC, Steve Wells/U Dundee, Rick Schnurr/NASA/GSFC, and Don Dominic/BNSC/Logica also participated in the review of the work of this SIG. The author chaired the SIG.

REFERENCES

- [SOIF] CCSDS SOIF Home Page, <ftp://ftp.estec.esa.nl/pub/ws/wsd/ccsds/ccsdsoif/index.htm>
- [SREF] CCSDS SOIF Reference Model, 3rd SOIF Workshop,
<ftp://ftp.estec.esa.nl/pub/ws/wsd/ccsds/ccsdsoif/Meetings/Third%20meeting/shames2.ppt>
- [PROX] CCSDS Proximity Space Link Protocol, CCSDS Red Book 211.0-R-3, Jan 2002, <http://ccsds.org/>
- [CFDP] CCSDS File Delivery Protocol (CFDP), CCSDS 727.0-B-1, Blue Book, Jan 2002, <http://ccsds.org/>
- [SAPI] Message Layer Study and Report, 5th SOIF Workshop,
ftp://ftp.estec.esa.nl/pub/ws/wsd/ccsds/ccsdsoif/Meetings/Fifth%20meeting/SOIF_Msg_Layer.ppt
- [PROT] Message Service Prototype plan, Messaging SIG,
http://groups.yahoo.com/group/SOIFpanel/files/SIG-5%20Message%20Layer/SOIF_Msg_Proto_Apprch-17Jan02.ppt
- [CORB] Object Management Group, <http://www.omg.org>
- [ACE] Adaptive Communications Environment (ACE), <http://www.cs.wustl.edu/~schmidt/ACE.html>
- [1553] MIL-STD-1553 Description, <http://www.1553.com/1553interp.htm>
- [FIRE] IEEE-1394 Firewire Description, <http://www.1394ta.org/Technology/About/faq.htm>
- [SPWR] SpaceWire home site, <http://www.estec.esa.nl/tech/spacewire/>
- [PUS] Telemetry and Telecommand Packet Utilization, Draft 10, Apr 2002, ECSS-E-70-41
- [MDS] Mission Data System (MDS), <http://x2000.jpl.nasa.gov/flash/technology/mds.html>
- [XML] W3C Recommendation - Extensible Markup Language (XML) 1.0 (Second Edition, 6 October 2000), <http://www.w3.org/TR/REC-xml>