CCSDS
The Consultative Committee for Space Data Systems

Report Concerning Space Data System Standards

SPACECRAFT ONBOARD INTERFACE SERVICES

INFORMATIONAL REPORT

CCSDS 850.0-G-2

GREEN BOOK
December 2013

The Consultative Committee for Space Data Systems

Report Concerning Space Data System Standards

SPACECRAFT ONBOARD INTERFACE SERVICES

INFORMATIONAL REPORT

CCSDS 850.0-G-2

GREEN BOOK
December 2013

AUTHORITY

Issue:	Informational Report, Issue 2
Date:	December 2013
Location:	Washington, DC, USA

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and represents the consensus technical agreement of the participating CCSDS Member Agencies. The procedure for review and authorization of CCSDS documents is detailed in *Organization and Processes for the Consultative Committee for Space Data Systems*, and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the address below.

This document is published and maintained by:

CCSDS Secretariat
Space Communications and Navigation Office, 7L70
Space Operations Mission Directorate
NASA Headquarters
Washington, DC 20546-0001, USA

FOREWORD

This document is a CCSDS Informational Report to assist readers in understanding the Spacecraft Onboard Interface Services (SOIS) documentation. It has been prepared by the Consultative Committee for Space Data Systems (CCSDS). The concepts described herein are the baseline concepts for the CCSDS standardisation activities in respect of communication services and generic support services to be used in the flight segment of spacecraft systems.

This Report describes the challenges posed by spacecraft onboard interfaces, details the service architecture of the SOIS services, and elaborates on the goals and expected benefits of the of key SOIS services. It is intended to serve as a reference for both service users and service implementers in order to maximise the potential of standardised onboard interfaces with respect to re-use, interoperability, and inter-agency cross support.

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Report is therefore subject to CCSDS document management and change control procedures, which are defined in *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-3). Current versions of CCSDS documents are maintained at the CCSDS Web site:

http://www.ccsds.org/

Questions relating to the contents or status of this document should be addressed to the CCSDS Secretariat at the address indicated on page i.

DOCUMENT CONTROL

Document	**Title**	**Date**	**Status**
CCSDS 850.0-G-1	Spacecraft Onboard Interface Services, Informational Report, Issue 1	June 2007	Original issue, superseded
CCSDS 850.0-G-2	Spacecraft Onboard Interface Services, Informational Report, Issue 2	December 2013	Current issue

CONTENTS

CONTENTS (continued)

CONTENTS (continued)

EXECUTIVE SUMMARY

The CCSDS Spacecraft Onboard Interface Services (SOIS) Area has developed a layered set of communications services for flight avionics. This set of services is intended to cover the majority of onboard communications requirements. The services have been divided into those to be provided over the onboard communications media—the so-called Subnetwork Layer services—and those supporting onboard applications—the Application Support Layer services.

The SOIS Area does not directly specify protocols; rather it defines the services to be supported by the underlying protocols and in doing so eases the changes required when an existing protocol is substituted for an alternative. In the case of ESA, the SOIS services are used to drive the development of all data link protocols developed under the ECSS standardisation authority, for example MIL-STD-1553B, CAN bus, and SpaceWire.

In order to locate the SOIS services within the onboard hardware and software environment, a reference communications architecture has also been established. In developing this architecture care has been taken to ensure that existing implementations can take benefit from the SOIS services without major change. For example, the AFRL SPA architecture implements proprietary protocols for subnetwork access, but the device virtualization service is compatible with that defined by SOIS. Similarly the NASA GSFC cFE, a message based implementation, benefits from both the device virtualization service and the subnetwork services and protocols specified for SpaceWire.

The use of Electronic Data Sheets to specify device interface details, although driven by the SOIS architecture, is not SOIS dependent, and all avionics implementations will greatly benefit from precise, machine-readable specifications. Such specifications open up many possibilities for directly derived data products such as spacecraft database import and data driven interface software.

In summary, SOIS provides a set of well-structured communications services located within a reference architecture. An implementer may choose to apply all services or only a selection relevant to a particular implementation approach. In much the same way, the reference architecture can be used to guide an implementer in developing a layered approach, but it is by no means mandatory to take benefit from the SOIS service specifications.

1 INTRODUCTION

1.1 PURPOSE

The purpose of this document is to describe the concept and supporting rationale for the Spacecraft Onboard Interface Services (SOIS) developed by the Consultative Committee for Space Data Systems (CCSDS). This document:

– provides an introduction and overview of the SOIS services concept upon which the detailed CCSDS SOIS recommendations are based;

– summarises the specific individual service recommendations and supplies the supporting rationale.

This document is a CCSDS Informational Report and is therefore not to be taken as a CCSDS Recommended Standard.

1.2 SCOPE

This document:

– describes the rationale and approach of CCSDS SOIS standardisation;

– establishes the SOIS concepts and architecture (including the addressing strategy);

– provides an overview of the SOIS services;

– provides examples of the deployment of SOIS services and protocols.

The basic context of SOIS services is that of a single spacecraft within a single mission. Communications between elements outside of a single spacecraft and communications between multiple spacecraft falls outside the scope of SOIS. However, other CCSDS services exist that fulfil these external interfacing requirements, and the SOIS services are designed to be compatible with these.

The case of application of wireless local area networking in swarms of spacecraft or proximal landed elements is addressed separately (reference [13]).

1.3 APPLICABILITY

The SOIS standardised services are intended to be applicable to all classes of civil missions, including scientific and commercial spacecraft, and manned and un-manned systems. These standardized services may apply to military missions, although military security requirements have not been considered in their specification.

1.4 RATIONALE

CCSDS has enjoyed a great deal of success in standardisation of interfaces between spacecraft and ground systems and has managed to extend this success to areas such as lander-to-orbiter interfaces. Although CCSDS's authority derives from the requirement for interoperability between national space agencies, the primary benefit has been in cost and risk reduction internal to the agencies and to the individual missions. This manifests itself in:

– reuse of mission infrastructure;

– sharing of resources between missions and agencies;

– reuse of mission hardware and software;

– ready availability of space-qualified components and subsystems;

– accumulated knowledge base within the agencies;

– reuse of standard electrical ground support equipment;

– extensive validation of the operation and completeness of the standards.

In general, spacecraft interface development is based on unique designs which are specified and implemented on a project-by-project basis. Any reuse of these interfaces is usually a by-product of reuse of the whole spacecraft bus, with data handling interfaces having no self-sustaining level of reuse. While individual developers may have limited proprietary standards, these are generally closed and require significant adaptation across missions, particularly those involving interorganisational cross support.

Similarly, there exists little interface standardisation which can be used by individual equipment and instrument providers. While it is true that there are a limited number of physical interfaces applicable for use in the space environment, the services and access to these interfaces vary considerably between implementations.

At the international level there have so far been very few significant attempts at standardizing spacecraft onboard interfaces, and consequently incompatible interfacing solutions have evolved. Typically, the interfacing solutions that have been developed for spacecraft are tied to the peculiarities of spacecraft buses, real-time operating systems, and existing flight software approaches, and bear very little resemblance to the 'plug-and-play' interfaces used to integrate computing devices in modern terrestrial systems.

The result is that a multitude of solutions are in place, with each mission either inheriting past solutions or developing new ones. However, an increase in the number and complexity of international missions and the cost of developing state-of-the-art high-speed data interfaces has led to significant impetus for pushing missions in the direction of using standards within and across programs.

CCSDS is perfectly placed to develop standards for agency adoption because:

– it can call on a multi-agency expertise base;

– it can offer global cost and risk reduction by nurturing suppliers on an international basis;

– it has the influence at mission and agency level to promote standards adoption.

Within CCSDS, the SOIS area has been charged with addressing the issue. Its solution lies in the development of a suite of open recommendations involving the complete spacecraft. The goal of the CCSDS SOIS standardisation activity is therefore to develop standards that will improve both the process of spacecraft development and integration as well as the quality of the finished product, and at the same time facilitate the adoption of promising new hardware and software technologies supporting international onboard interface interoperability.

The SOIS approach is to standardise the interfaces between items of spacecraft equipment by specifying well-defined standard service interfaces and protocols which allow standardised access to sensors, actuators, and generic spacecraft functions, allowing spacecraft applications to be developed independently of the mechanisms that provide these services. Applications are thus insulated from the specifics of a particular spacecraft implementation and may be reused across different spacecraft platforms with little regard of implementation details.

Service interface standardisation allows hardware interfaces to be accessed by flight software such that core spacecraft software may be reused on different underlying communications infrastructures with little or no change. The standard services could be implemented using a standard Application Programming Interface (API) that would enable portability and re-use of application software, and of service implementations.

The definition of the services makes no assumption about the implementation of the services in hardware or software or a mixture of both. In addition, SOIS aims to promote interoperability between software and hardware devices operating on various spacecraft communication buses. There are several benefits of this approach:

– as long as the subnetwork services remain stable, software and hardware may evolve independently;

– developers of core spacecraft software can rely on a standard set of services on which to base their design;

– requirements definition activities are reduced as direct reference may be made to the CCSDS Recommended Standards;

– a standard test suite may be used during qualification;

– costs are reduced by adhering to a single solution;

– risk is reduced through amortisation of development and testing across mission cost and time bases;

– subsystem and payload portability across missions is enabled;

– the possibility for reuse of both interfaces and core spacecraft software and the scope for further standardisation activities are significantly increased.

The SOIS services,

- in conjunction with protocol specifications, allow portability of equipment across spacecraft that use the same data links;
- in conjunction with standard APIs, allow source code portability of application software across spacecraft that use different data links;
- in conjunction with subnetwork protocol conversion, as implemented by the Transfer Layer, allow portability of equipment across spacecraft that use different data links;
- in conjunction with Network and Transport Layer standardisation, support interoperability between equipment onboard spacecraft via a number of data links.

1.5 APPROACH

The process for SOIS standardisation is progressively:

- to identify and articulate a standard set of services which application software or higher-layer services can use to communicate between onboard components over a single data link;
- to provide standard mappings between service provision and various underlying data link communications media, recognizing that implementation of services is link-dependent;
- to provide a framework to allow various qualities of service to be supported over any underlying data link;
- to develop protocols in support of the various SOIS services;
- to promote the development of standard APIs implementing the services, thus promoting further software reuse.

The current SOIS activity is limited to service definition. The existence of standard services is not, in and of itself, sufficient to enable interoperability between data systems or to allow complete portability of application software. It is, however, a necessary condition for the definition of protocols to enable interoperability; it allows application software reuse at a semantic level; and, again, it is a necessary condition for the definition of APIs that will promote complete application software portability.

The definition of protocols and APIs cannot begin without the establishment of SOIS standard services. It is envisaged that, with the publication of these services, API and protocol development will take place in CCSDS Member Agencies, in industrial partners, and within CCSDS itself.

1.6 TERMS AND DEFINITIONS

With respect to service and protocol definition, SOIS, in general, uses terms and definitions defined within the ISO Open Systems Interconnect model defined in reference [2]. The following definitions are provided:

best effort: Not guaranteeing packet delivery.

data link: Connection between onboard subnetwork locations for the purpose of transmitting and receiving data.

NOTE – In this Informational Report the term data link refers to the onboard subnetwork should not be confused with the CCSDS space-to-ground Data Link Layer for Space Communications Protocols.

data system: An addressable entity situated in a subnet which hosts an instance of the subnetwork protocols, subnetwork services, and subnetwork users. The subnetwork users are uniquely identifiable in a subnetwork by a combination of data system address and a protocol ID. A data system is typically a computer or a device.

data system address: An identifier which uniquely identifies a data system in a subnetwork. The data system address may be referred to as a Destination Address or a Source Address depending on the context of its invocation at the subnetwork service interface. A data system may have more than one data system address.

device: Any physical device onboard a spacecraft that can be commanded and/or data acquired from by an onboard application across a subnetwork.

device abstraction control procedure, DACP: The control procedure that provides the abstraction of a device-specific access protocol to a functional interface. This may involve e.g., the application of calibrations to raw values provided by the device or combination of multiple raw values to determine a derived value in SI units.

device-specific access protocol, DAP: The protocol that makes use of a subnetwork service to command and/or acquire data from a device. This is specific to each device as no standardisation of access protocols at the device level exists.

dictionary of terms, DoT: Ontology of terms used to describe engineering profiles of data in functional interfaces in electronic data sheets.

electronic data sheet, EDS: Electronic description of a device’s functional interface, device-specific access protocol, and, optionally, device abstraction control procedure.

engineering profile: A collection of attributes used to define the meaning of an item of data used in operations and parameters. An engineering profile may include attributes defining semantic type (behaviour or meaning) or attributes of that define syntactic type, such as the number of bits in a variable and the encoding of integers.

functional interface: The command and data acquisition operations provided by a device, isolated from the access protocols. This may or may not involve abstraction from raw values provided by a device (see device abstraction control procedure).

heterogeneous network: A network that uses more than one underlying communications protocol, e.g., part SpaceWire and part MIL-STD-1553B.

packet: Delimited octet-aligned data unit.

priority: Identification of the transmit precedence of an SDU relative to other SDUs.

protocol data unit; PDU: A unit of data specified in a protocol and consisting of protocol control information and possibly user data.

protocol ID: An identifier which uniquely identifies a SOIS subnetwork user within a data system.

quality of service; QoS: The ability of a communication system to provide predictable and differentiated services. Quality of service for a communication service may be characterised in terms of important features relevant to that communications service, for example: reliability, transmission rate, effective bandwidth and latency, error rate.

reliability: A QoS parameter indicating whether or not a protocol function will attempt to acknowledge the successful receipt of a packet and possibly retry sending a PDU if no acknowledgment is received by the sender.

semantic type: A descriptive attribute for an item of an onboard device or software component that helps users to identify the behaviour and meaning (semantics) for that item. Users use the semantic type to distinguish the use and purpose of different items.

service class: A category of service on a subnetwork distinguished by its QoS.

service data unit; SDU: A unit of data passed into or out of a service interface.

subnetwork: An abstraction of a collection of equipment and physical media, such as a local area network or a data bus, which forms and autonomous whole and can be used to interconnect real systems for the purpose of data transfer.

syntactic type: A category of information structure conforming to a specific set of rules for combining words and other elements. Syntax refers to the spelling and grammar of an item. Each item may define its own syntactical rules that control which words may be used, which combinations of words are meaningful, and what punctuation is necessary. The syntactic type of an item is the identification of the rules for that type of item. A syntactic type uses attributes of an engineering profile that describe the representation of numbers in bits, such as encoding, and width in bits. The syntactic types often correspond to familiar primitive types in programming languages, such as 32-bit float, or 16-bit unsigned integer.

time criticality: Necessity to deliver a packet within a certain period of time or treat preferentially with respect to other packets.

user: An entity that makes use of a SOIS service.

virtual device: A virtual version of a single physical device, exposing an idealised interface with a structured syntax and a simplified semantics and thus hiding the operation of the real device.

1.7 REFERENCES

The following documents are referenced in the text of this Report. At the time of publication, the editions indicated were valid. All documents are subject to revision, and users of this Report are encouraged to investigate the possibility of applying the most recent editions of the documents indicated below. The CCSDS Secretariat maintains a register of currently valid CCSDS publications.

[1] *Organization and Processes for the Consultative Committee for Space Data Systems*. Issue 3. CCSDS Record (Yellow Book), CCSDS A02.1-Y-3. Washington, D.C.: CCSDS, July 2011.

[2] *Information Technology—Open Systems Interconnection—Basic Reference Model: The Basic Model*. 2nd ed. International Standard, ISO/IEC 7498-1:1994. Geneva: ISO, 1994.

[3] *Space Packet Protocol*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 133.0-B-1. Washington, D.C.: CCSDS, September 2003.

[4] *Encapsulation Service*. Issue 2. Recommendation for Space Data System Standards (Blue Book), CCSDS 133.1-B-2. Washington, D.C.: CCSDS, October 2009.

[5] J. Postel. *Internet Protocol*. STD 5. Reston, Virginia: ISOC, September 1981.

[6] J. Postel. *Transmission Control Protocol*. STD 7. Reston, Virginia: ISOC, September 1981.

[7] *CCSDS File Delivery Protocol (CFDP)*. Issue 4. Recommendation for Space Data System Standards (Blue Book), CCSDS 727.0-B-4. Washington, D.C.: CCSDS, January 2007.

[8] *Asynchronous Message Service*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 735.1-B-1. Washington, D.C.: CCSDS, September 2011.

[9] V. Cerf, et al. *Delay-Tolerant Networking Architecture*. RFC 4838. Reston, Virginia: ISOC, April 2007.

[10] K. Scott and S. Burleigh. *Bundle Protocol Specification*. RFC 5050. Reston, Virginia: ISOC, November 2007.

[11] M. Ramadas, S. Burleigh, and S. Farrell. *Licklider Transmission Protocol—Specification*. RFC 5326. Reston, Virginia: ISOC, September 2008.

[12] *The Application of CCSDS Protocols to Secure Systems*. Issue 2. Report Concerning Space Data System Standards (Green Book), CCSDS 350.0-G-2. Washington, D.C.: CCSDS, January 2006.

[13] *Wireless Network Communications Overview for Space Mission Operations*. Issue 1. Report Concerning Space Data System Standards (Green Book), CCSDS 880.0-G-1. Washington, D.C.: CCSDS, December 2010.

[14] *Satellite Earth Stations and Systems (SES); European Co-operation for Space Standardization (ECSS); Satellite Software Data Handling Interfaces (SSDHI)*. ETSI EN 301 927 V1.1.1 (2003-02). Sophia-Antipolis: ETSI, 2002.

[15] *Space Engineering—Ground Systems and Operations—Telemetry and Telecommand Packet Utilization*. ECSS-E-70-41A. Noordwijk, The Netherlands: ECSS Secretariat, January 2003.

2 SOIS CONCEPTS AND ARCHITECTURE

2.1 CONCEPTS

On any given spacecraft, several types of data subnetworks may be used between specific data systems. The actual type of subnetwork used is determined by the required characteristics of the interaction between those entities. These may typically be categorised as:

– **Multidrop Buses** providing connection to a central bus master and a number of slaves (e.g., MIL-STD-1553B), or (occasionally) peer-to-peer (e.g., TTP/c); a shared resource that must be highly managed. Communication is generally asymmetrical with only one transmitter on at a time and often involves low-level read and write access to slaves. The central control of bus traffic results in a highly stochastic traffic profile well suited to applications requiring bounded communications timing.

– **Point-to-point serial interfaces** used for instrument connection, possibly for bulk data transfer but also combined with instrument control and gathering sensor readings or controlling spacecraft equipment. Again, these interfaces usually operate in a master/slave mode.

– **Homogeneous Networks** used on larger infrastructures where data systems have generally equal computing power and have a diversity of communication requirements. Communication is on a peer-to-peer basis with a level of variability in delay due to resource queuing.

Onboard applications should not be concerned with the nature of these subnetworks, and so the SOIS concept aims to provide a solution by recommending that applications interact only with a well-defined set of standard onboard data services. In addition, given the disparity of functionality supported by different Data Link Layers, the SOIS subnetwork services provide a common interface and a convergence of common services to the upper-layer applications for communicating over any single data link. Together, the application and subnetwork services provide a standard means to communicate between virtually all spacecraft components.

Figure 2-1 shows a layered view of the recommended services and their associated access points. Users, i.e., **Mission-Specific Applications,** are the mission-dependent applications that make use of the SOIS-defined services. The **Transfer Layer** provides Transport- and Network-layer services based on existing protocols either defined or adopted by CCSDS (e.g., IP and Space Packet Protocol). In many cases the Transfer Layer will not be required. It is required, however, where multiple data links may be accessed by a spacecraft node or where routing across multiple data links is needed. There are also benefits in adopting some Transfer Layer functionality to provide data link-independent addressing. The **Subnetwork Layer** provides access to the data link medium and provides a set of SOIS-defined services to be mapped over the subnetwork defined by that medium. Mappings can be made to an extensible set of various Data Link Layer protocols, e.g., MIL-STD-1553B, SpaceWire, and wireless protocols, but is out of scope of SOIS.

Network management services, while part of the architecture, are for future development and will be discussed only briefly in this document. Network management aspects are implemented for all services and functions and will be accessed by a variety of methods. SOIS makes no recommendations concerning these access methods. However, it is incumbent on SOIS to detail the Management Information Bases (MIBs) for all of its recommendations.

Details of the services, the architecture, and associated protocols are given in subsequent subsections.

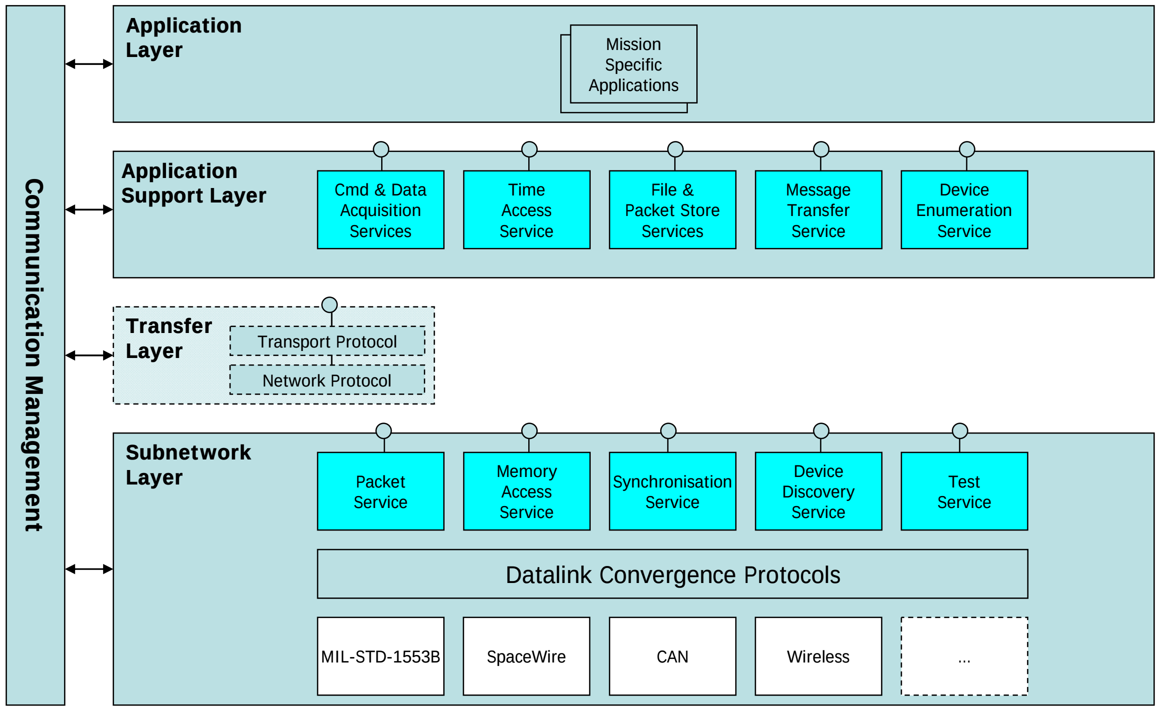

Figure 2-1: SOIS Reference Communications Architecture

It should be noted that all service access points, whether they be at Subnetwork, Transfer, or Application Support Layer, are accessible by users, i.e., mission-specific applications.

2.2 ARCHITECTURAL CONSIDERATIONS

In arriving at the overall SOIS architecture a number of fundamental issues must be tackled:

– Spacecraft do not uniformly use a single underlying data communications medium; instead, a single spacecraft will most likely implement one or more buses and point-to-point links. Where there are multiple buses and point-to-point links they are likely not to be combined into a single heterogeneous network with routing between them, because of resource limitations and users’ real-time requirements.

– The choice of underlying data communications media, although limited, will vary across spacecraft and depend on mission needs.

– Communicating devices are often asymmetrical in regard to their capabilities. Typically, a spacecraft design will use a limited number of processors having sufficient resources to implement a full protocol stack, but the communications will also require access to sensors and actuators that have little or no computing capabilities.

2.3 MANAGEMENT CONCEPTS

SOIS conforms to the established consensus within CCSDS regarding management concepts.

A Management Information Base (MIB) description will be mandatory for inclusion in any protocol specification claiming to implement SOIS Services and will include parameters, databases, and actions necessary to inform operation of the protocols. The method of access to the MIB by the management system is undefined and may be a combination of preconfigured code, local configuration, or remote management via management protocol and local agent.

Management functions within the SOIS architecture configure the SOIS services with respect to quality of service, e.g., reserving resources and allocating a channel identifier for those resources, and also informing user applications about attributes of the configured service, e.g., informing the user applications about the existence of a channel and the resources which are reserved for it.

2.4 IMPLEMENTATION CONFORMANCE

An Implementation Conformance Statement (ICS) proforma is recommended for all SOIS Service specifications. Any implementation claiming conformance to a SOIS service should need to provide a completed ICS.

Although, SOIS does not specify any protocols to be implemented to provide a SOIS service, it is recommended that a Protocol Implementation Conformance Statement (PICS) Proforma be included in any protocol specification claiming to implement SOIS Services. Any implementation of such a protocol claiming conformance to a SOIS Service should provide a completed PICS for the protocol in addition to the ICS for the SOIS Service.

2.5 SERVICE INTERFACE PROVISION

All services are defined in terms of abstract service primitives. The concrete instantiation, i.e., implementation, of these service primitives is outside the scope of the SOIS standards. Examples may include:

– direct API, blocking, or call-back;

– software bus, using for example service request and response message exchange.

2.6 COMMANDING AND ACQUIRING DATA FROM ONBOARD DEVICES

2.6.1 INTRODUCTION

The SOIS Command and Data Acquisition Services provide the ability for onboard applications to command and acquire data from onboard devices across subnetworks, whilst being isolated from the protocols associated with the particular subnetworks.

Applications can interface to the functionality directly provided by a physical device or with a higher level abstraction of the physical device, known as a *virtual device*. Such device virtualisation is described in the following subsections.

2.6.2 DEVICE VIRTUALISATION

2.6.2.1 General Concept

Device Virtualisation provides, as its name suggests, a virtual version of a single physical device. That virtual version hides the operation of the real device and exposes an idealised interface with a structured syntax and a simplified semantics; the semantics of a virtual device are based on *command* and *acquire* operations.

– A **command** operation modifies a single specified parameter in a device, where a parameter may have complex type, e.g., a set of fields. The operation will be self-contained with no unexpected side-effects.

– An **acquire** operation returns the current value of a specified parameter from a device, where a parameter may have complex type, e.g., a set of fields. Again, the operation will be self-contained with no side-effects.

Each virtual device is therefore defined by:

– the list of command operations which are valid for the device:

  Each command operation is defined by the device parameter being commanded and an identifier. Each parameter, in turn, has an identifier and an associated engineering profile.

– the list of acquire operations which are valid for the device:

  Each acquire operation is defined by the device parameter being acquired and an identifier. The parameter has its own identifier and an associated engineering profile.

These lists of operations together make up the provided **Functional Interface** for a device.

Engineering profiles are essential for defining the meaning of operations and must be uniquely related to human-readable descriptions and physical quantities, allowing, for example, tool-driven reasoning on a SOIS-based system or selection of a device to make use of in a mission. This task is accomplished by the use of a **Dictionary of Terms** (DoT), or ontology, used to specify all valid terms that can compose an *engineering profile*. Engineering profiles can be used in electronic data sheets to describe data in the functional interfaces provided by virtual devices, and in the required functional interfaces of applications. (See section 5 for more information of DoTs.)

These Functional Interfaces can be described by an **Electronic Data Sheet** (EDS) to permit automatic code generation for tooling support (design time adaptivity) or online application configuration (run-time adaptivity). The EDS must describe the available operations using their identifiers and types. (See section 5 for more information upon EDS.)

2.6.2.2 Layering of Device Protocols

Virtualisation of a device from its functional interface to the 'on-the-wire' protocol to the device is achieved through the layered architecture of protocols that together provide the interface to the function provided by the device. A layered architecture is illustrated in figure 2-2.

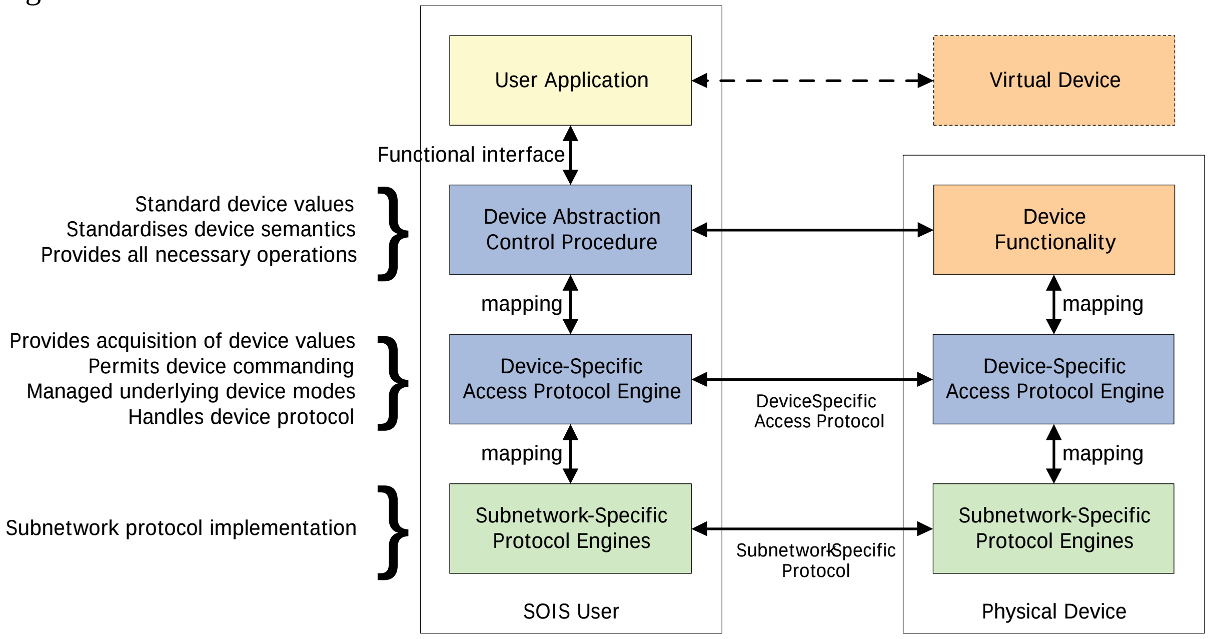
User Application
Virtual Device
Functional interface
Standard device values
Standardises device semantics
Provides all necessary operations
Device Abstraction Control Procedure
Device Functionality
mapping
mapping
Provides acquisition of device values
Permits device commanding
Managed underlying device modes
Handles device protocol
Device-Specific Access Protocol Engine
DeviceSpecific Access Protocol
Device-Specific Access Protocol Engine
mapping
mapping
Subnetwork protocol implementation
Subnetwork-Specific Protocol Engines
SubnetworkSpecific Protocol
Subnetwork-Specific Protocol Engines
SOIS User
Physical Device

Figure 2-2: Layering of Protocols and Procedures to Provide Device Virtualisation

The purpose of each layer is:

– Subnetwork protocol—standardised, subnetwork-specific protocols. This layer transfers data to and from the device, implementing any Quality-of-Service requirements such as timeliness or reliability.

– Device-specific Access Protocol (DAP)—provision of the device-specific interface for each device and hiding the device access method, through the mapping of the device-specific interface onto the subnetwork services by a protocol engine that uses one or more subnetwork services' primitives. This layer may be a simple one-to-one mapping of device-specific command and acquisition operations to subnetwork service primitives.

– Device Abstraction Control Procedure (DACP)—provision of the functional interface for each virtual device, hiding the physical device specifics through the mapping of functional interface onto the device-specific interface by a set of control procedures that use one or more device-specific type conversions and interface primitives. DACPs may, for example, contain:

  a) state machines using multiple device accesses to provide a function, using the DAP;

  b) reflexive semantic-syntactic type conversions, for example using calibration curves to convert between raw values and engineering units;

  The DACP may be a simple one-to-one mapping of functional to device-specific command and acquisition operations. However, it will become more complex when the functional interface does not directly match the device-specific interface, e.g., through compliance with a generic functional interface (see the next subsection on standardisation).

The DACP and DAP are provided by the Command and Data Acquisition Services. Further information is provided in 3.2.

2.6.2.3 Standardisation of Device Virtualisation

With device virtualisation, standardising the functional interface for device types into a generic device image is possible (where there are common functions across devices of the same or similar type). This *generic functional interface* should allow replacement of physical devices of the same type without affecting the applications using them. It would be useful to include a mechanism by which device-specific extensions to the generic functional interface may be introduced, so as not to prevent innovation as new devices are designed with new capabilities that specific missions wish to exploit. This may only be possible if the differing physical characteristics of the device do affect usage by the application or can be quantified, such as being provided as a constant value within the functional interface so that the application can include this as a constant term in its algorithm obtained from the functional interface. Of course, it is important to be clear that device virtualisation is possible without standardisation of generic functional interfaces.

Device virtualisation allows application portability without requiring online required/provided functional interface matching, permitting true design-time adaptivity with static applications. For a virtual device to be standard it must implement the complete functional interface for one or more device types; it may supplement those operations with additional, device-specific operations, if required.

Such an approach relies upon a single **common** DoT. (See section 5 for more information of a common DoT.)

The best manner of specifying generic functional interfaces is through the specification of a **standard EDS** for each device type using the common DoT, listing the invariant functional interface for a given device type. Such an approach is currently out of scope of the CCSDS, though there is nothing to stop individual agencies standardising functional interfaces.

2.7 PLUG-AND-PLAY OF SPACECRAFT DEVICES

2.7.1 INTRODUCTION

SOIS takes a narrow definition of plug-and-play of devices, encompassing design-time activities as well as ‘run-time’ or ‘operation-time’ activities, but excluding dynamic discovery of device capabilities at run-time.

The benefits of plug-and-play are:

– **Interoperability**. Permitting application portability and hardware interoperability promotes reuse and lowers both development and operating costs. Plug-and-play should isolate software application and hardware device, permitting flexibility and innovation in both.

– **Adaptability**. Plug-and-play should introduce the ability for systems to adapt to change, such as changes in devices or software. This adaptability is not necessarily an attribute of the implemented space system, but may instead be associated with the development process. Adaptability promotes reuse and permits system change late in the development process or during operation.

– **Agility**. The characteristics of plug-and-play should promote shorter development times, assisting design, implementation, integration, and testing.

2.7.2 USE CASES

Four scenarios serve to illustrate the use of plug-and-play:

– **Rapid spacecraft development**. Plug-and-play techniques are used to assist the development of a spacecraft. The resulting space system is expected to be static during operations with no requirement to dynamically detect or configure devices. In this case the plug-and-play adaptivity is in the development tool chain.

– **Automated integration and test**. Assisted by Ground Support Equipment (GSE), i.e., test tools and simulated devices, device detection techniques are used to assist integration, checking for integration problems, and to assist and automate testing. In this case the dynamic aspects of plug-and-play are being used, but the adaptivity can be in the GSE rather than the spacecraft.

– **Dynamic fault recovery**. The device detection capabilities of plug-and-play can be used to assist FDIR, verifying the presence of devices in a standard way. Should a fault occur, recovery can also be assisted by plug-and-play techniques, detecting a powered-on redundant device and automating the reconfiguration of the subnetwork in response. This provides a very limited form of adaptivity in the onboard software.

– **Dynamic Device Migration.** Some systems may be dynamic during operation, such as those that utilise wireless data links and may appear, disappear, or move; or those that are physically modified with human or robotic interaction, again causing the subnetwork topology to change. In these cases, plug-and-play device discovery and network configuration are utilised to provide a level of onboard adaptivity that can accommodate the expected changes to the subnetwork.

2.7.3 DEVICE DISCOVERY AND SUBNETWORK CONFIGURATION

Where run-time adaptivity is necessary, subnetwork **Device Discovery Services** (DDS) identify connected devices and receive notifications of device additions/removals to/from the subnetwork. DDS:

– discovers the initial subnetwork topology, including current devices on subnetwork;

– detects any changes to subnetwork topology, including addition or removal of devices through failure or controlled disconnection (including power down);

– informs subnetwork management with device discovery information;

– provides notification of device changes (addition/removal).

In response to device discovery, the subnetwork is responsible for configuring the subnetwork-related features of devices, as well as other network resources such as routers. This must be guided by some policy to control, for example, address assignment. Such a policy will typically be mission specific.

2.7.4 DEVICE ENUMERATION

Device enumeration assists the onboard reconfiguration functions such as mode management or fault detection, isolation and recovery regarding the notification of changes in the spacecraft configuration, and execution of the needed operations to adjust the onboard software to the new configuration. While the reconfiguration strategy is the responsibility of the user applications, device enumeration relates to the detecting the addition or removal of devices and the reconfiguration of the other SOIS services used to access these devices. In addition, device enumeration supports the spacecraft integration and test by managing the verification of the correct configuration of the onboard devices with respect to the spacecraft desired configuration or by easing the insertion/removal of device simulators in replacement of the real equipment.

Device enumeration is provided by the Device Enumeration Service (DES), one of the Command and Data Acquisition Services. Further information is provided in 3.2.

The basic identification information for the devices installed in the system (known as device metadata) are retrieved by the Subnetwork DDS and used by the DES to assign a (system-wide, unique) virtual device identifier to each device so that they can be accessed by the user applications using the appropriate functional interface, DACP, DAP, and subnetwork services (namely the Subnetwork Memory Access or Packet Service).

2.7.5 IMPACTS ON VALIDATION & VERIFICATION

The use of the Subnetwork DDS and DES introduces additional functionality dealing with potentially a high number of configuration combinations. However, not all the possible configurations are used by a given subsystem of a specific spacecraft. This aspect needs to be considered in the validation and verification process at different levels of the development.

– Device Discovery and Enumeration mechanisms need to be validated independently from the possible network/device configurations at the software component level.

– Validation is needed in the context of the spacecraft where the software components are deployed covering the configurations actually supported.

– System integration and validation can then be streamlined by using standard test suites to verify the discovery and enumeration at subsystem level.

– Subsystem simulators can be more easily used during system integration.

2.8 NAMING AND ADDRESSING

SOIS requires a consistent and integrated approach to a number of addressing regimes:

– the addressing of devices and, for each service, an abstract, device independent representation of data link service parameters (e.g., memory addresses, parameter names);

– translation of addresses by the SOIS application functions and device-dependent drivers;

– the addressing of data link service users by data system, service, and user entity;

– the addressing of application service users by data system, service, and user entity.

The particular challenge of the SOIS asymmetrical communications model is in ensuring that connection-oriented transactions remain possible given the disjoint addressing capabilities of the two ends of a transaction. This is typified by a low-level device having no cognisance of the addressing to an Application Support Layer entity, requiring that address translation occur through the system and that it be reversible, such that return data is sent to the correct data requestor.

The scope of addressing in the SOIS layers is:

– Application Support Layer—abstract application, storage and device entity identification (may be achieved through disjoint identifier sets, i.e., one per entity, or globally across all entities), and translation between abstract entity identification and spacecraft network addressing;

– Transfer Layer—global addressing of data systems, routing to multiple subnetworks, selection of subnetwork, conversion between spacecraft network address, and data link-dependent address;

– Subnetwork Layer—data link-dependent subnetwork addressing.

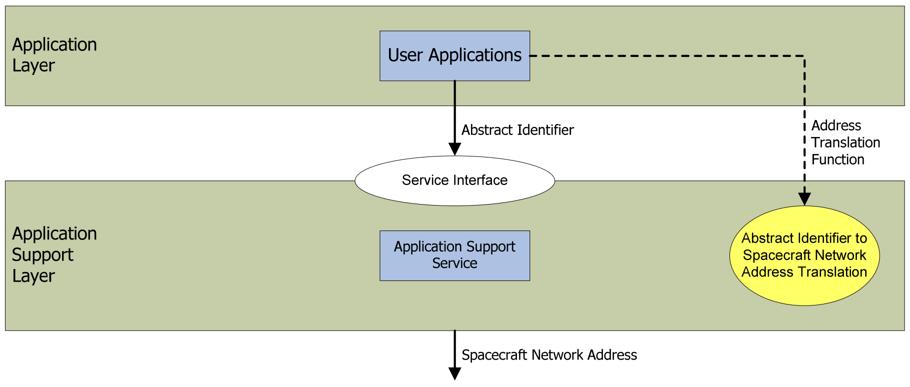
Application Layer
User Applications
Abstract Identifier
Address Translation Function
Service Interface
Application Support Layer
Application Support Service
Abstract Identifier to Spacecraft Network Address Translation
Spacecraft Network Address

Figure 2-3: Application Support Layer Addressing

Figure 2-3 shows the Application Support Layer addressing architecture, using a generic Application Support Service as an example. Here, the user application identifies objects to be interacted with via the application support services by an abstract identifier (e.g., device identifier ‘reaction wheel 1’ in the case of the Command and Data Acquisition Services). A function common to all Application Support Layer services is then to convert that abstract address to the concrete spacecraft network address required by the underlying Transfer Layer. This function may also be directly accessible to user applications where they access the Transfer Layer directly.

NOTE – The service interface of this conversion function is not standardised.

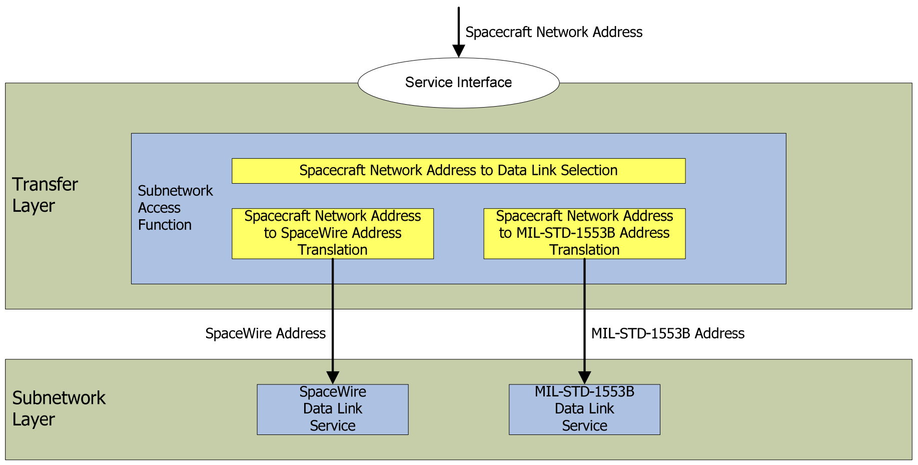
Spacecraft Network Address
Service Interface
Transfer Layer
Subnetwork Access Function
Spacecraft Network Address to Data Link Selection
Spacecraft Network Address to SpaceWire Address Translation
Spacecraft Network Address to MIL-STD-1553B Address Translation
SpaceWire Address
MIL-STD-1553B Address
Subnetwork Layer
SpaceWire Data Link Service
MIL-STD-1553B Data Link Service

Figure 2-4: Subnetwork Layer Addressing

In the SOIS addressing model, the Transfer Layer uses a spacecraft network address, which is global within a spacecraft, to select the data link to be used and to derive a data link-dependent subnetwork address which is used to traverse the subnetwork. This function is shown in figure 2-4 provided as part of the subnetwork access functions supporting the subnetwork services.

It should be noted the subnetwork address is required only for the Subnetwork Packet Service (where it forms part of the Packet Destination Service Access Point (PDSAP) along with user entity identification), the Subnetwork Memory Access Service (where it is the Destination Address), and the Subnetwork Test Service (where it is the Test Address for non-self test operation). The Synchronisation and Device Discovery Services do not require Subnetwork Address translation (though a reverse translation may possibly be performed to convert subnetwork-specific addresses to Spacecraft Network Addresses).

Figure 2-5 shows the envisaged most common case where full network protocol functionality (including routing and relaying) is not deemed necessary for intra-spacecraft communications. In this case the Application Support Layer services use the spacecraft network directly. The figure also shows an application directly using subnetwork services. In this case the Application Support Layer may still provide abstract to subnetwork address translation to the application.

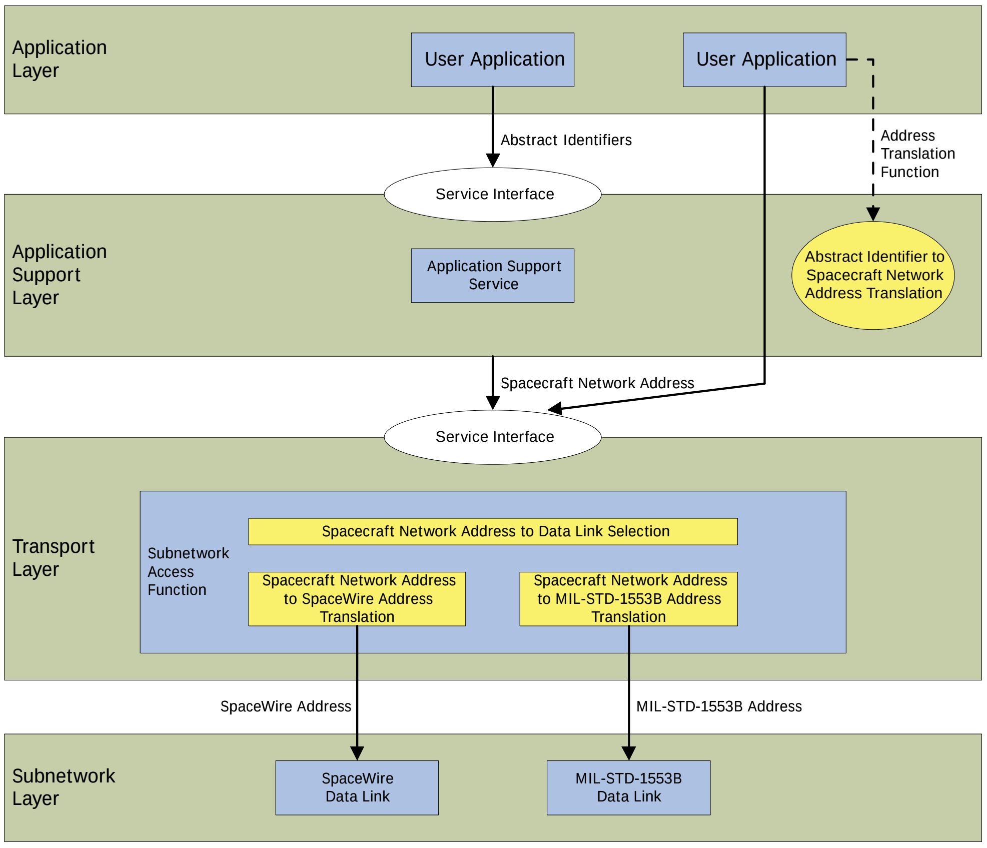
Application Layer
User Application
User Application
Abstract Identifiers
Address Translation Function
Service Interface
Application Support Layer
Application Support Service
Abstract Identifier to Spacecraft Network Address Translation
Spacecraft Network Address
Service Interface
Transport Layer
Subnetwork Access Function
Spacecraft Network Address to Data Link Selection
Spacecraft Network Address to SpaceWire Address Translation
Spacecraft Network Address to MIL-STD-1553B Address Translation
SpaceWire Address
MIL-STD-1553B Address
Subnetwork Layer
SpaceWire Data Link
MIL-STD-1553B Data Link

Figure 2-5: Direct Use of Spacecraft Network Addressing by the Application Support Layer

There are a number of translations and selections in the protocol stack illustrated in figure 2-5; abstract identifier to spacecraft network address translation, the spacecraft network address to data link selection and associated data link address translation. Such translations and selections may be determined by QoS parameters which may be generated in the application or application support layers. The QoS parameters may include reliability, timeliness, or service specific (e.g., time synchronization accuracy) values.

Figure 2-6 shows the use of UDP/IP in the Transfer Layer operating over the SOIS packet service.

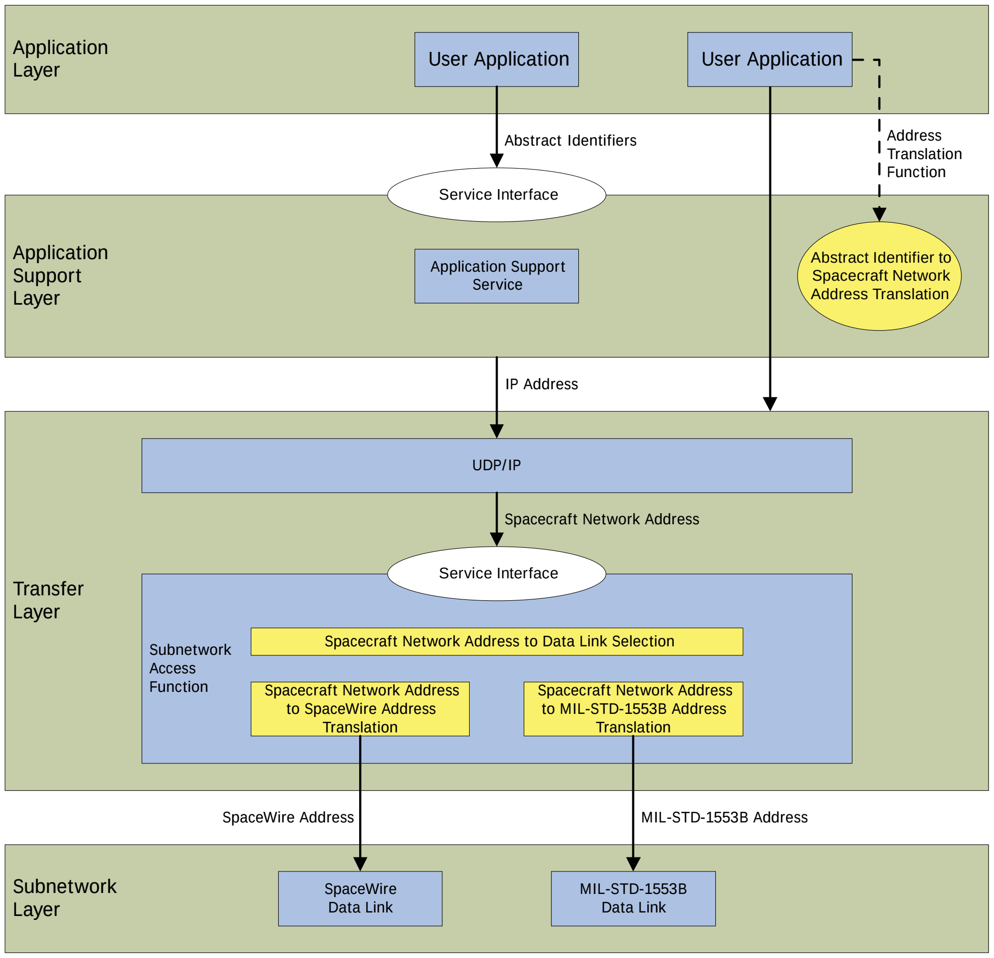
Application Layer
User Application
User Application
Abstract Identifiers
Address Translation Function
Service Interface
Application Support Layer
Application Support Service
Abstract Identifier to Spacecraft Network Address Translation
IP Address
UDP/IP
Spacecraft Network Address
Service Interface
Transfer Layer
Subnetwork Access Function
Spacecraft Network Address to Data Link Selection
Spacecraft Network Address to SpaceWire Address Translation
Spacecraft Network Address to MIL-STD-1553B Address Translation
SpaceWire Address
MIL-STD-1553B Address
Subnetwork Layer
SpaceWire Data Link
MIL-STD-1553B Data Link

Figure 2-6: Incorporation of IP for Global Addressing, Routing, and Relaying

Figure 2-7 reviews the addressing architecture in the context of device virtualization, in order to explain the differences in address sets. An application may communicate directly with any of the layers in the figure, using the address set that is appropriate to that layer.

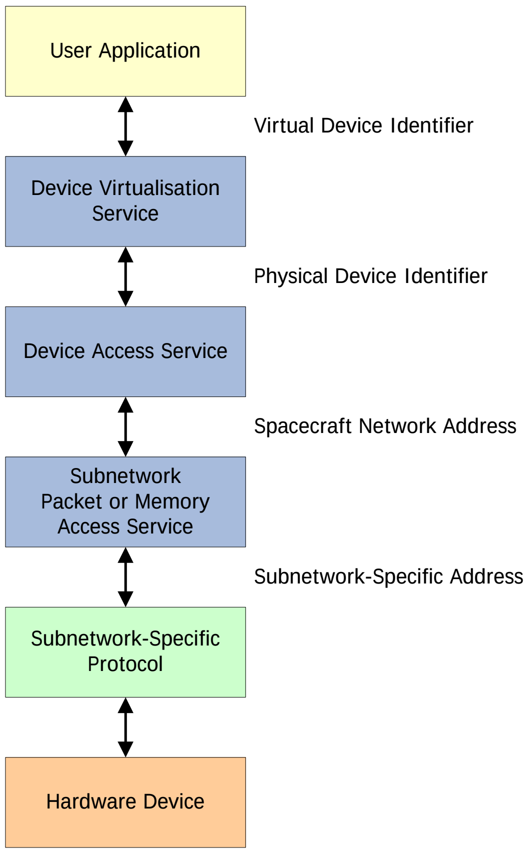
User Application
Virtual Device Identifier
Device Virtualisation Service
Physical Device Identifier
Device Access Service
Spacecraft Network Address
Subnetwork Packet or Memory Access Service
Subnetwork-Specific Address
Subnetwork-Specific Protocol
Hardware Device

Figure 2-7: Addressing for Device Virtualization

The **virtual device identifiers** presented by the Device Virtualization Service are a subset of the total set of addressable components onboard the spacecraft. Figure 2-4 indicates the other addressable components that may be present, such as file stores, packet stores, and other nodes, which are not represented in device virtualization.

The virtual device identifiers do not necessarily map one-to-one to the hardware devices. A redundancy-control application could alter the mapping between virtual devices and real devices in the Device Virtualization Service.

The physical device identifiers used by the Device Access Service map one-to-one to the devices onboard the spacecraft. If a device has more than one network adapter, then the Device Access Service would represent each of its network adapters with a separate physical device identifier. The representation of devices in the Device Access Service is an exact model of the devices actually present on the vehicle. Optionally, an implementation of SOIS may use a subset of spacecraft network addresses as the physical device identifiers, but this equality is not required for compliance with SOIS. For example, the physical device identifiers could be integers, while the spacecraft network addresses could be IP addresses.

The **spacecraft network addresses** are a set of mutually unique addresses that span the spacecraft, mapping one-to-one to the addressable entities onboard the spacecraft. This mapping is not necessarily one-to-one with the network adapters, because some onboard computers may present multiple addresses through a single network adapter. Figure 2-6 shows an implementation of spacecraft network addresses using IP addresses.

The **subnetwork-specific addresses** are the addresses that are appropriate to the various network communication technologies that may be present onboard a spacecraft. There is a separate set of addresses for each such subnetwork.

2.9 SOIS SERVICE AND PROTOCOL ARCHITECTURE

2.9.1 OVERVIEW

This subsection describes the resulting SOIS Reference Communications Architecture, in terms of services and protocols.

2.9.2 GENERAL

SOIS services, as shown earlier, exist at three service interfaces:

– an Application Support Layer service interface;

– a Transfer Layer service interface;

– a Subnetwork Layer service interface.

2.9.3 APPLICATION SUPPORT LAYER SERVICES

The Application Support Layer services provide a number of capabilities which are commonly required onboard a spacecraft and which need not be limited to communications. The Application Support Layer services make use of the Subnetwork Layer services either locally to a data system or remotely over a network. The services are defined in terms of protocols, procedures, protocol data units, and a Management Information Base (MIB). The Application Support Layer services identified are:

– Command and Data Acquisition—typically used to access spacecraft hardware devices such as sensors and actuators:

  • Device Access—a device-specific interface providing direct access to a device using *command* and *acquire* operations with no interpretation of the data (i.e., raw) associated with these operations, and the side-effects of each operation;

  • Device Virtualisation—adaptation of the parameters and semantics of the device-specific interface provided by Device Access to match the functional interface provided by the virtual device (this functional interface may be standardised into a generic form);

  - Device Data Pooling—maintaining an image of the states of a number of devices' values (either virtual or raw);

- Time Access—providing access to a local time source;
- Message Transfer—providing application-to-application message exchange;
- File and Packet Store Services—providing access to the spacecraft storage system;
- Device Enumeration—providing support for dynamic spacecraft configuration.

SOIS Application Support Layer services may be provided across a spacecraft network consisting of a number of heterogeneous or homogeneous SOIS subnetworks in conjunction with a Network Layer such as the Space Packet or IP protocol. In cases where time criticality does not allow this approach, the SOIS Application Support Layer services may directly use SOIS Subnetwork Layer services over a single subnetwork.

The command and acquire operations offered by the DAS must access the device using the (sub-)network Packet and Memory Access Services using a DAP, which for local devices would in effect be a device driver providing access to a local memory map. To do this, DAS must manage states or modes associated both with the DAP, and with the device itself.

The DVS must adapt the parameters and semantics of this interface to match the functional interface provided by the virtual device using the DACP. The DACP typically involves:

- the conversion of values, from device-specific units to the International System of Units (SI);
- the adaptation of device semantics, to present the simplified semantics associated with a virtual device;
- the provision or emulation of necessary device functions which are not provided by the underlying physical device.

Depending on the semantics of the physical device, when simplifying operations the DVS may need to manage aspects of device state or modes, such as device acquisitions steps (begin ADC conversion, wait for ADC conversion, acquire result). Application Support Layer services are more fully addressed in section 3 of this document.

2.9.4 TRANSFER LAYER SERVICES

At present, the Transfer Layer is assumed to be composed of extant CCSDS-recognised protocols and services. Examples of these are:

- TCP/UDP/IP;
- Space Packet Protocol.

Transfer layer functionality may be present to provide access to multiple subnetworks managed as a single network. This may be provided as part of the above Transfer Layer protocols or via a mission-dependent solution.

Typically two functions are provided:

– routing of packets between the subnetworks that make up the network;

– addressing of data systems attached to the subnetworks of the spacecraft.

If there is no requirement for packets to be routed between data systems on different subnetworks, or if there is indeed an explicit requirement not to allow this, then a Transfer Layer may not be required.

If a consistent addressing model of all data systems attached to the subnetworks of the spacecraft is required an address translation function from a spacecraft network address to a subnetwork and associated subnetwork address should be deployed (see 2.8 for more details).

2.9.5 SUBNETWORK SERVICES

The SOIS Subnetwork provides a set of SOIS-defined services which support upper-layer Application Support and Transfer Layer entities. The subnetwork services which are provided are independent of the underlying data link in that the service primitives and associated parameters are the same regardless of the underlying data link. Multiple data links may be available to a user entity and, in this case, multiple instances of the subnetwork service, one for each data link, would be available to the user entity. Data link selection is achieved by selection of a service instance by the entity making use of subnetwork services.

The services provided by the underlying data link need to be matched to those required by the subnetwork service, and this may require the provision of convergence functions. Convergence functions add the necessary functionality to that inherently provided by the data link. The work needed to map a particular set of services to a data link depends on the services provided by the underlying data link. In some cases the inherent data link service will be equivalent to that required by the subnetwork, in which case no convergence function will be required. In others, the inherent data link service will not match that required by the subnetwork, in which case convergence functions will be required.

The services identified at the Subnetwork Layer are:

– Memory Access (memory location read/write, includes read/modify/write)—providing direct access to device memory;

– Synchronisation—providing spacecraft time and event synchronisation;

– Packet—providing packet delivery over a single subnetwork;

– Device Discovery—providing dynamic device recognition;

– Test Service—providing establishment of subnetwork functionality and availability.

SOIS will define a standard subnetwork service interface for each of the services outlined above.

For each SOIS-compliant data link there will need to be a mapping of subnetwork services to actual data link implementation, including the provision of convergence functions where required. This mapping is left to dedicated groups and possibly SOIS (where no dedicated group is available). CCSDS may adopt as recommendations the mappings performed by dedicated groups. An example of such a dedicated group is the ECSS SpaceWire Working Group.

The SOIS subnetwork provides access to the subnetwork medium (e.g., cable) and, as such, does not rely on any underlying services.

2.9.6 DEPLOYMENT SCHEMES

The following examples illustrate the SOIS approach.

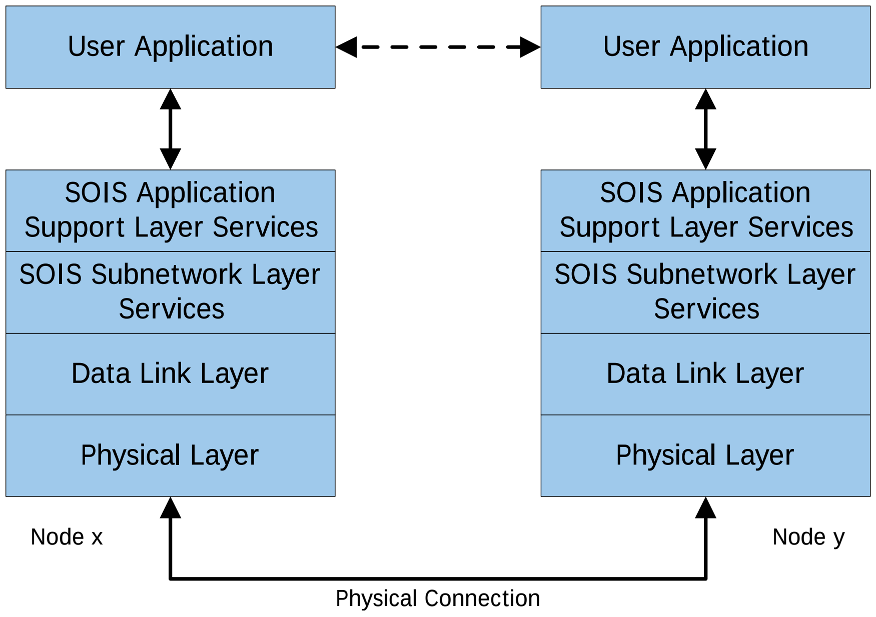
User Application
User Application
SOIS Application Support Layer Services
SOIS Subnetwork Layer Services
Data Link Layer
Physical Layer
SOIS Application Support Layer Services
SOIS Subnetwork Layer Services
Data Link Layer
Physical Layer
Node x
Node y
Physical Connection

Figure 2-8: Symmetrical Communication

In figure 2-8 it is assumed that all communications are supported by intelligent data systems able to implement a full SOIS protocol stack. This scenario is typically used for communication between application software supported by a processor and associated resources, for example for message exchange using the SOIS message transfer service.

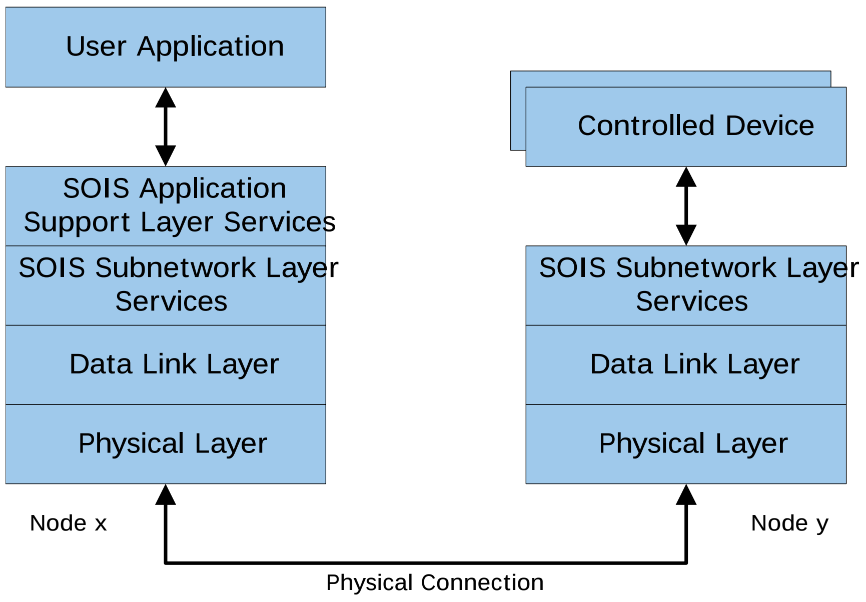

Figure 2-9: Application Layer Asymmetrical Communication

In figure 2-9 it is assumed that the controlled device is directly connected to a standard spacecraft Physical and Data Link Layer and has sufficient capability to implement SOIS-defined Subnetwork Layer services. Currently, the only SOIS subnetwork service which provides symmetrical communication is the Packet service.

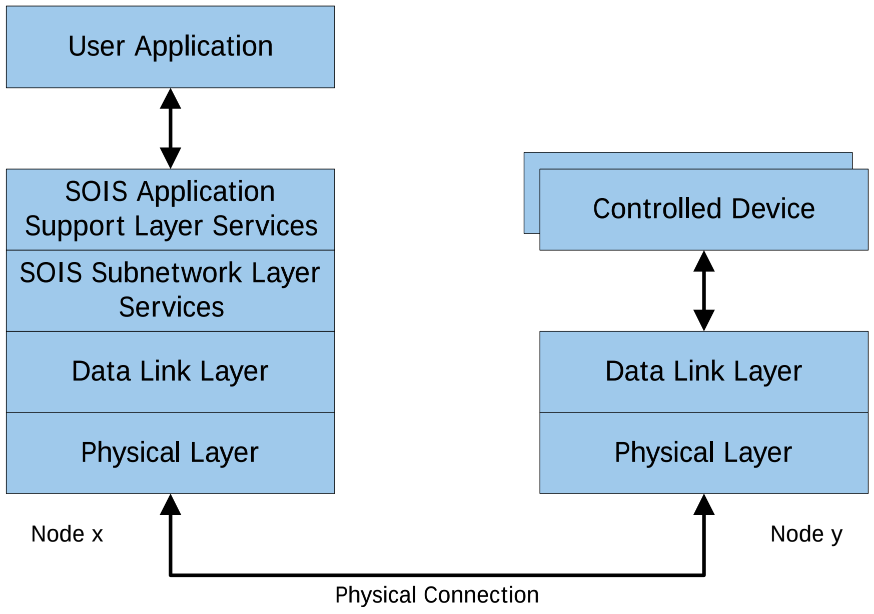

Figure 2-10: Subnetwork Asymmetric Communication

In figure 2-10 it is assumed that the controlled device is connected directly to a data link and has no capability to implement the SOIS protocol stack. This is typical of sensors and actuators where the interface provides only for reading and writing register values. This example requires an asymmetric method of communication whereby the SOIS capabilities resident in the controlling data system must take full responsibility for the controlled device. Unlike a conventional peer-to-peer data communications scenario, the service is provided at only one service interface, although service provision involves more than one data system. The SOIS subnetwork services which operate in this mode are the Synchronisation Service, the Memory Access Service, the Test Service, and the Device Discovery Service.

2.10 DEVICE VIRTUALISATION VIEW OF SOIS ARCHITECTURE

This subsection describes how the SOIS Reference Communications Architecture provides device virtualisation by providing a walk-through of how the SOIS services interact together to provide device virtualisation, as illustrated in figure 2-11.

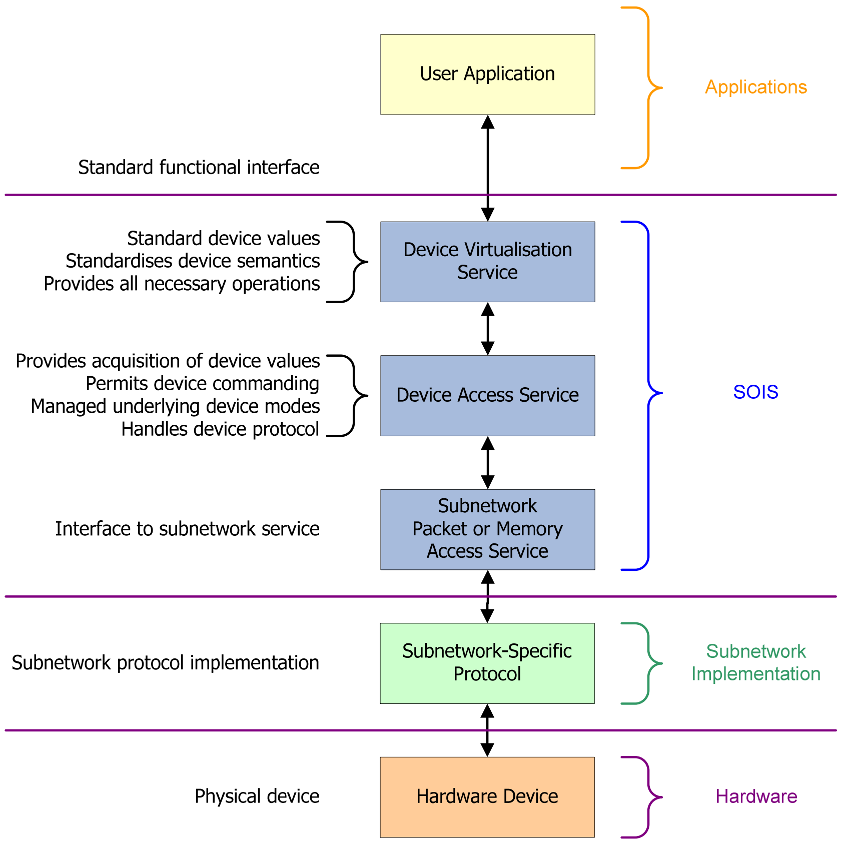
User Application
Applications
Standard functional interface
Standard device values
Standardises device semantics
Provides all necessary operations
Device Virtualisation Service
Provides acquisition of device values
Permits device commanding
Managed underlying device modes
Handles device protocol
Device Access Service
SOIS
Interface to subnetwork service
Subnetwork Packet or Memory Access Service
Subnetwork protocol implementation
Subnetwork-Specific Protocol
Subnetwork Implementation
Physical device
Hardware Device
Hardware

Figure 2-11: Device Virtualisation View of SOIS Architecture

a) The subnetwork protocol transfers data from the user's data system to the device, providing a QoS. The Subnetwork Service provides a consistent service interface to this subnetwork protocol.

b) The DAS accesses the functionality provided by the device by implementing the DAP, permitting commanding of the device and acquiring device values as well as management of underlying device modes.

c) Finally, the DVS provides the virtual, possibly generic, functional interface of the device, through mapping to the device-specific interface by the DACP. This for example provides conversion of values between device-specific representations and standardised representations from the DoT and through sequences of device-specific commands and value acquisitions to affect a standardised command or acquire a standardised value composed from a number of device-specific values.

2.11 DEVICE PLUG-AND-PLAY VIEW OF SOIS ARCHITECTURE

This subsection describes how the SOIS Reference Communications Architecture provides plug-and-play of spacecraft devices by providing a walk-through of how the SOIS services interact together to provide device plug-and-play, as illustrated in figure 2-12.

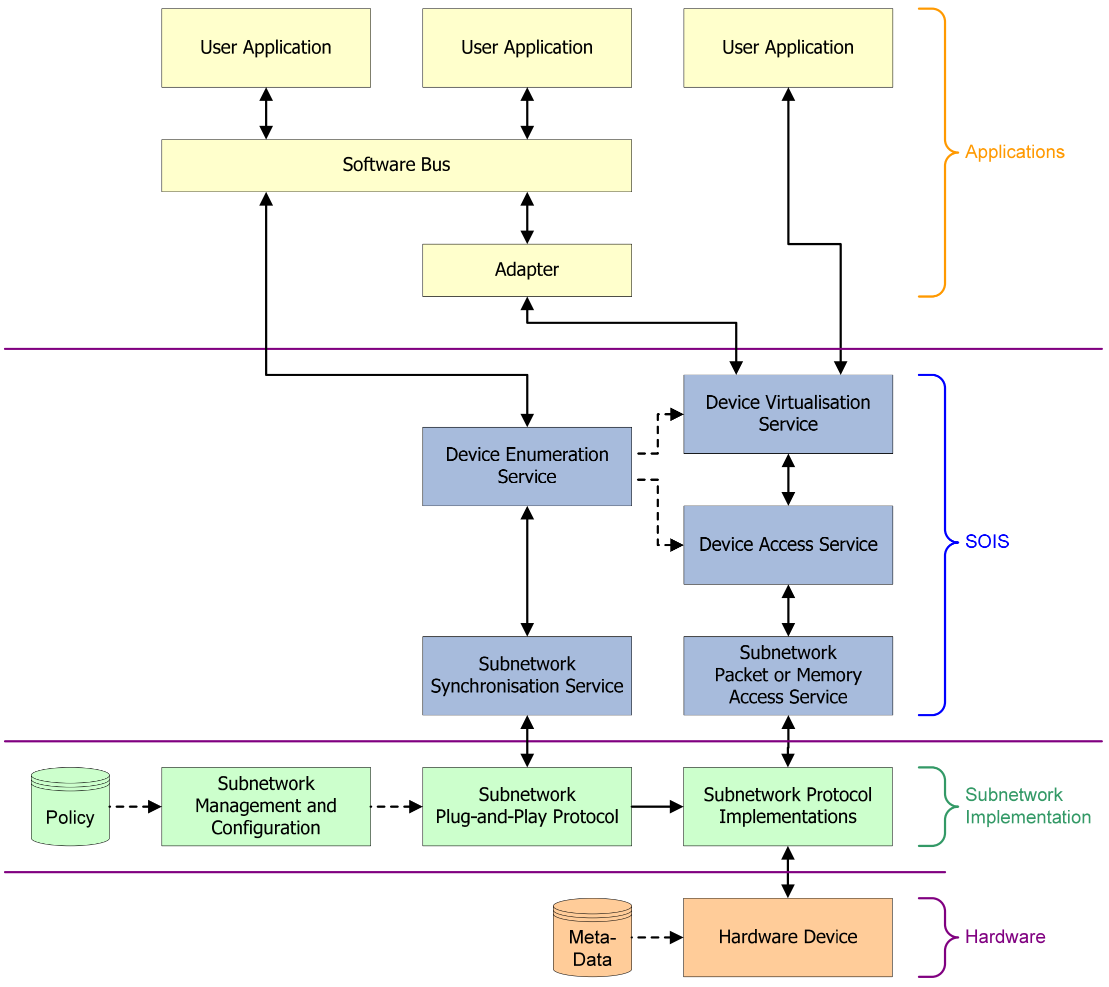
User Application
User Application
User Application
Software Bus
Adapter
Applications
Device Virtualisation Service
Device Enumeration Service
Device Access Service
SOIS
Subnetwork Synchronisation Service
Subnetwork Packet or Memory Access Service
Policy
Subnetwork Management and Configuration
Subnetwork Plug-and-Play Protocol
Subnetwork Protocol Implementations
Subnetwork Implementation
Meta-Data
Hardware Device
Hardware

Figure 2-12: Device Plug-and-Play View of SOIS Architecture

There are three levels of plug-and-play available, on a device-by-device basis (depending upon device capabilities):

a) No plug-and-play. Device IDs and spacecraft network addresses are hard coded. Spacecraft network and subnetwork addresses are pre-allocated. DVS and DAS are hard coded with necessary DACPs and DAPs with well-known virtual and physical device identifiers. DDS and DES are not used. This may be the case for dumb devices that provide no device discovery capability.

b) Device Capability Verification. Application uses DDS to check expected device matches reality—checking that device's metadata, e.g., serial number, acquired by DDS matches the expected one. This may be used, for example, for fault management. DES not used. This may be used by fault management to verify that the expected discoverable devices are accessible.

c) Device Discovery. DES is used to dynamically discover devices of known device type and then configure DVS and DAS to provide access to them (including device identifier allocation and association with DACP, DAP and spacecraft network address); applications are notified of the device ID and associated metadata, e.g., serial number.

This architecture highlights the ability for virtual devices to appear as components on a software bus. In figure 2-12 above, an adapter is shown, presenting the DVS interface for a virtual device in a form suitable for the software bus. Typically for a SOIS-compliant architecture a software bus would utilise the Message Transfer Service (MTS) for communication between components.

This architecture requires device metadata but could in addition include the ability to use an EDS hosted either on the device or in some EDS repository to permit runtime adaptivity. However, this is out of scope of SOIS.

Where design-time adaptivity is used, the EDS becomes part of the design flow, aiding or automating the generation of DVS and DAS. (See section 5 for more information on EDS.)

2.12 SOIS COMPLIANCE

The SOIS initiative defines standard services at the Subnetwork Layer upon which upper-layer applications may rely. The subnetwork services are independent of the underlying hardware and thus insulate upper layers from the mechanisms of the subnetwork protocols. Further standardisation is provided by defining a common set of Application Support Layer services upon which spacecraft-dependent applications may be built.

Ideally, any SOIS-compliant protocol specification should implement the full set of either Subnetwork or Application Support Layer services, but, in reality, protocol specifiers must have the freedom to select only those services which are required and to omit others in accordance with protocol capabilities. In addition, implementers of these protocols have the freedom to select those services pertinent to their requirements. A SOIS-compliant implementation is therefore compliant to a protocol specification which is itself compliant to SOIS services.

To aid this selection process while still allowing implementers to claim SOIS compliance, the SOIS-compliant protocols shall incorporate Protocol Implementation Conformance Statement (PICS) proformae indicating mandatory and optional elements in support of the SOIS services. These proformae are required to be completed by any developer claiming compliance to SOIS services. The resulting completed form is the Protocol Implementation Conformance Statement for that implementation.

SOIS will endeavour to restrict options to a bare minimum in the interests of promoting interoperability at both protocol and service levels. However, because of the varying capabilities of interfaces and the varying requirements of applications, some profiling will be defined, and appropriate conformance-statement proformae shall be included in any SOIS-compliant protocol definition.

3 SOIS APPLICATION SUPPORT LAYER SERVICES

3.1 INTRODUCTION

This section describes the suite of SOIS Application Support Layer services which are designed to provide common services required by applications on any processing data system of the spacecraft. They isolate the applications from the underlying topology and communications architecture of the spacecraft. These services then rely upon services provided by the underlying SOIS layers to carry the services' protocols over the spacecraft communications architecture.

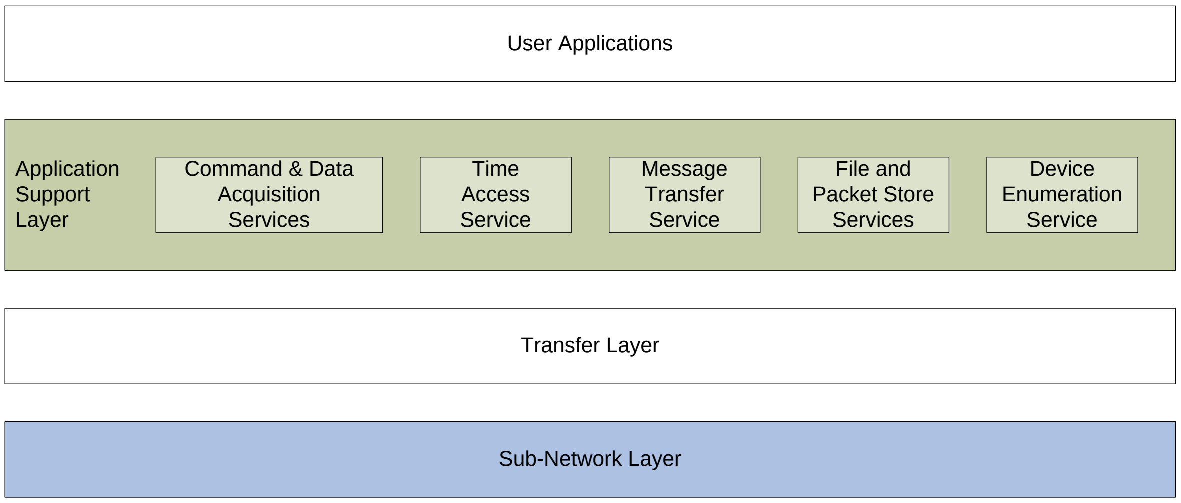
User Applications
Application Support Layer
Command & Data Acquisition Services
Time Access Service
Message Transfer Service
File and Packet Store Services
Device Enumeration Service
Transfer Layer
Sub-Network Layer

Figure 3-1: SOIS Application Support Layer Services

The suite of SOIS Application Support Layer services is as follows:

– Command and Data Acquisition Services (CDAS)—commanding and data acquisition by applications for devices (e.g., transducers and simple instruments) independent of their locations;

– Time Access Service (TAS)—access for applications to the onboard time with bounded accuracy independent of their locations;

– Message Transfer Service (MTS)—communication between applications hosted onboard a spacecraft using asynchronous, ad-hoc, discrete messaging with a bounded latency, including multicast and broadcast, independent of their locations;

– File and Packet Store Services (FPSS)—access by applications to, management of, and transfer of files and packets within a (nominal) global onboard store;

– DES—support for dynamic spacecraft configuration.

This section provides an overview of each of the Application Support Layer services (a complete description of each of these services can be found in the relevant CCSDS documents).

3.2 COMMAND AND DATA ACQUISITION SERVICES

3.2.1 GENERAL

The Command and Data Acquisition Services (CDAS) are used to provide a low-overhead access method for spacecraft hardware devices such as sensors and actuators, regardless of location.

The CDAS are split into a number of capability sets, each served by a distinct service:

- DAS—(Device Dependent Driver) providing basic reading from and writing to devices regardless of location;
- DVS—(Standard Device Driver) providing reference to a device using a virtual, i.e., generic, image of a physical device;
- Device Data Pooling Services (DDPS)—maintaining an image of the states of a number of devices' values.

The relationship between the services is shown in figure 3-2.

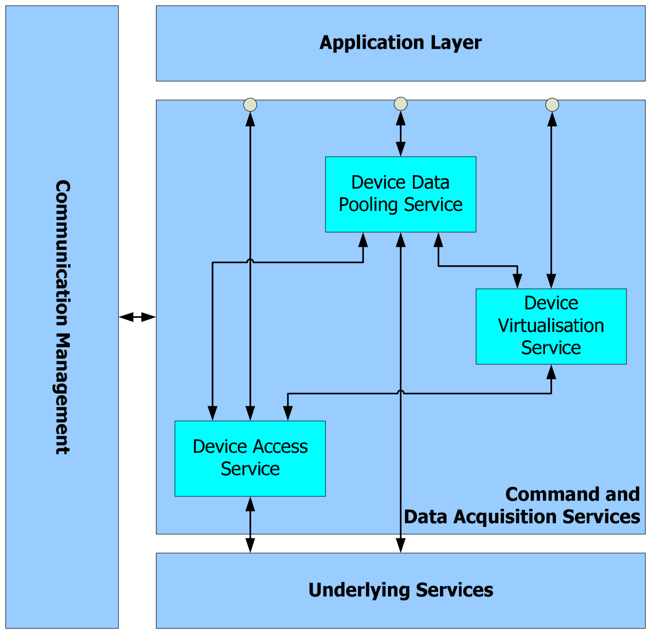
Application Layer
Communication Management
Device Data Pooling Service
Device Virtualisation Service
Device Access Service
Command and Data Acquisition Services
Underlying Services

Figure 3-2: Relationship between the Different SOIS Command and Data Acquisition Services

Each device is identified by the user's using an device identifier. The DVS uses a logical device identifier mapped to a physical device identifier used by DAS, which is responsible for accessing the device. DDPS uses either logical or physical device identifiers. The DAS maps the physical device identifier onto an access method and appropriate addressing scheme, e.g., a subnetwork service, a DAP and subnetwork service access point such as an address.

A device type is an abstraction of specific device implementations into a generic view of groups of devices with a common, i.e., generic function. The DAS provides access to a specific device, including the particulars of the function of that device. It may be thought of as a Device Dependent Driver. The DVS provides access to device types, with the implementation of the service mapping the generic function of the device type onto the specific devices, i.e., by using the DAS. It may be thought of as a Device Independent Driver.

3.2.2 DEVICE ACCESS SERVICE

3.2.2.1 Overview

The DAS provides a very basic device data acquisition and commanding capability that can be used directly by software applications, or can be used as the basis for more capable services, such as those performing engineering unit conversions on raw data, or monitoring services. The service user is isolated from the physical location or the detailed knowledge of the device's electrical interface and access method (be it accessed via direct IO; analogue, digital, pulsed, etc., or a protocol running across a subnetwork or a full network). It may be thought of as a **Device Dependent Driver**.

The benefit of the service is that the service user is no longer concerned with the details of the location of the sensor, its physical interface, or how it is accessed. As a result, configuration changes involving a change in the physical location of a device or changes to its electrical interface do not require changes to the application software using that device.

Although isolated from the details of device location and interface type, the service user must still know the format of the value, corresponding to each command, written to and the format of the value, corresponding to each data item, read from the device, and the user remains responsible for correctly composing and interpreting those formats.

3.2.2.2 Functions Performed

The DAS provides **Device-specific Access Protocol (DAP)** for each supported device. Each DAP provides the following functions:

– **Acquire value from device**: to acquire a value (i.e., data) from a device, a service user provides a physical device identifier and a value identifier. The service resolves the physical device identifier in order to determine the device location and the DAP through which it is accessed. The service uses the DAP to acquire the identified value from the device.

  Certain devices asynchronously emit values that are stored by the device's DAP. To acquire such a value, a service user provides a physical device identifier and a value identifier, which the DAP resolves in order to determine the stored value. The DAP and thence the service then returns the most recently stored value. Optionally, the

DAP and thence the service may also emit an indication to a service user when an asynchronously emitted value from a device is acquired by the service.

– **Command a device**: to command a device, a service user provides a physical device identifier and a value identifier, together with the command value to be written. The service resolves the physical device identifier in order to determine the DAP by which it is commanded. The service uses the DAP to command the device and return a response if generated by the device (as some devices do not generate a response to a command).

  The **DAP** maps information associated with the request onto functionality of the protocol, which in turn uses an underlying transport (UT) service's service access point, e.g., destination address and QoS parameters. There are three categories of DAP, based on the underlying transport service used:

  • Packet-based DAP. A protocol engine on the user's data system exchanges packets using an underlying Packet Service with a protocol engine on the device, which in turn interacts with the device's actual functionality.

  • Memory Access-based DAP. A protocol engine on the user's data system determines memory locations to access (read or write) on the device using an underlying Memory Access Service. On the device, the memory accesses drive a protocol engine that interacts with the device's actual functionality, e.g., through simply status and command registers or with functionality directly triggered by 'reads' or 'writes'.

    A special case is where a local device is accessed, e.g., using memory-mapped I/O. Here the underlying Memory Access Service is the local device driver providing the memory-mapped I/O.

The logical relationship between the service, the Device-specific Access Protocols and the UT service access points is illustrated in figure 3-3. This also illustrates how DAS accesses devices using underlying packet or memory access services (including direct accesses using a local driver).

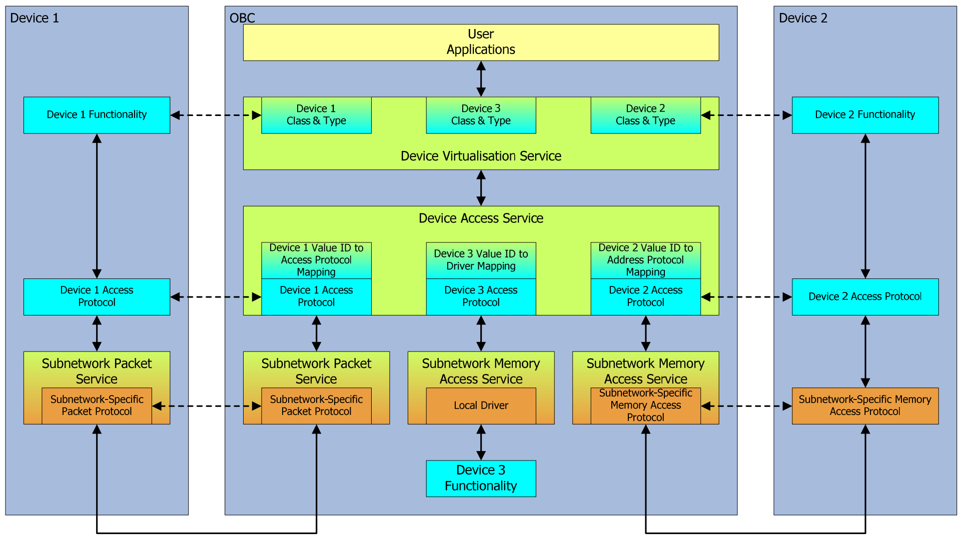
Device 1
OBC
Device 2
User Applications
Device 1 Functionality
Device 1 Class & Type
Device 3 Class & Type
Device 2 Class & Type
Device 2 Functionality
Device Virtualisation Service
Device Access Service
Device 1 Value ID to Access Protocol Mapping
Device 3 Value ID to Driver Mapping
Device 2 Value ID to Address Protocol Mapping
Device 1 Access Protocol
Device 1 Access Protocol
Device 3 Access Protocol
Device 2 Access Protocol
Device 2 Access Protocol
Subnetwork Packet Service
Subnetwork-Specific Packet Protocol
Subnetwork Packet Service
Subnetwork-Specific Packet Protocol
Subnetwork Memory Access Service
Local Driver
Subnetwork Memory Access Service
Subnetwork-Specific Memory Access Protocol
Subnetwork-Specific Memory Access Protocol
Device 3 Functionality

Figure 3-3: Relationship between DAS, DAPs, and Underlying Service Access Points

3.2.2.3 Requirements for Underlying Services

The access mechanisms themselves are provided by underlying services. They depend upon the different devices, but typically take the form of one of the following:

– packet;

– memory/register read/write.

Where the same access mechanism may be used to access different devices, the underlying service must provide a unique address for the device.

Where there is a common resource used for accessing multiple devices, e.g., an ADC multiplexer or a subnetwork, **contention** for the resource may occur. To support the real-time properties required by onboard applications, the underlying services must provide **bounded worst-case access times** and **prioritised access**.

3.2.3 DEVICE VIRTUALISATION SERVICE

3.2.3.1 Overview

The DVS permits the service user to refer to a device using a virtual, i.e., generic, image of a physical device. The service user interacts with the virtual image of the physical device and the DVS handles the translation of commands on the virtual image into commands to the physical device, and vice versa for data. A simple example of virtualisation is use of the virtual image like a disk drive for a physical device like flash memory. The user does not need to know the physical access mechanism of a flash memory and accesses it by sending commands to access a disk; such commands get translated into appropriate accesses to a flash memory. It may be thought of as a **Standard Device Driver**.

The benefit of using this service comes from the service user's being isolated from the physical characteristics of the device so that a class of devices can be commanded in the same manner, by interacting with the virtual device. The functions incorporated into a DVS implementation for a given device would typically exist in an implementation not employing SOIS, usually in the form of a library of functions for handling the device, converting acquired values, etc. The explicit specification of a device virtualisation service permits greater abstraction, standardisation, interoperability, and portability.

3.2.3.2 Functions Performed

The DVS provides **Device Abstraction Control Procedure (DACP)** for each supported device. The functions performed by each DACP are dependent upon the virtual devices to which they provide functional interfaces. However, they fall into two broad categories:

– Commanding: to command a device, a service user provides a virtual device identifier, command identifier, and command parameters. The service initiates the command and returns a status indicating the outcome of the command.

– Data Acquisition: to acquire data from a device, a service user provides a virtual device identifier and a data identifier. The service typically initiates the acquisition of the identified data from the device and returns the data.

3.2.3.3 Requirements for Underlying Services

The DVS typically uses the DAS to command and acquire data from a device.

3.2.4 DEVICE DATA POOLING SERVICE

3.2.4.1 Overview

The Device Data Pooling Service (DDPS) maintains an image of the states of a number of devices' values. A service user can access the state of a device's value in data pool without having to generate an explicit data acquisition request for the real device. The DDPS will periodically acquire data from the devices at a determined sampling rate or cache data from devices that asynchronously generate samples. The DDPS may be implemented on top of a DAS and/or DVS implementation, providing for example raw or engineering values respectively.

The benefit of using this service comes from the avoidance of the repetition of multiple users performing the same data acquisitions from devices, thereby reducing traffic on the underlying (sub-)network.

3.2.4.2 Functions Performed

The DDPS provides **Device-specific Access Protocol (DAP)** for each supported device. Each DAP provides the following functions:

– Acquire data from a device and store in data pool. This can take two forms:

  • DDPS periodically acquires data from device;

  • device asynchronously emits data to DDPS.

  Optionally, an implementation can synchronise acquisitions to the subnetwork, using the Subnetwork Synchronisation Service to periodically trigger the acquisitions.

  Optionally, an implementation can notify a service user when an acquisition has taken place so the service user can synchronise reading a value and subsequently processing it with the acquisition itself.

– Read value from data pool; allow service users to read values from the data pool.

– Optionally, add, remove, start and stop acquisition orders if the system requires dynamic management of acquisitions.

3.2.4.3 Requirements for Underlying Services

The DDPS uses the DAS and/or DVS to acquire data (including asynchronously) from a device.

Optionally, the DDPS uses the Subnetwork Synchronisation Service to synchronise acquisitions to the subnetwork.

3.3 TIME ACCESS SERVICE

3.3.1 GENERAL

The Time Access Service (TAS) provides service users with a consistent interface to a local time source, which typically is correlated to some centrally maintained master onboard time source by some mission-defined underlying service. The time values provided by this service might typically be used by the application to schedule some operation, such as the acquisition of an image, or to time stamp locally generated telemetry data.

The need to provide a local, correlated time source in onboard data systems is common to all spacecraft that have more than one processing data system connected to a subnetwork or a network. A typical architectural scenario is shown in figure 3-4. It should be noted the TAS is concerned only with providing the interface to the local time source. Time distribution is addressed by the Subnetwork Synchronisation Service described in 4.2.3.

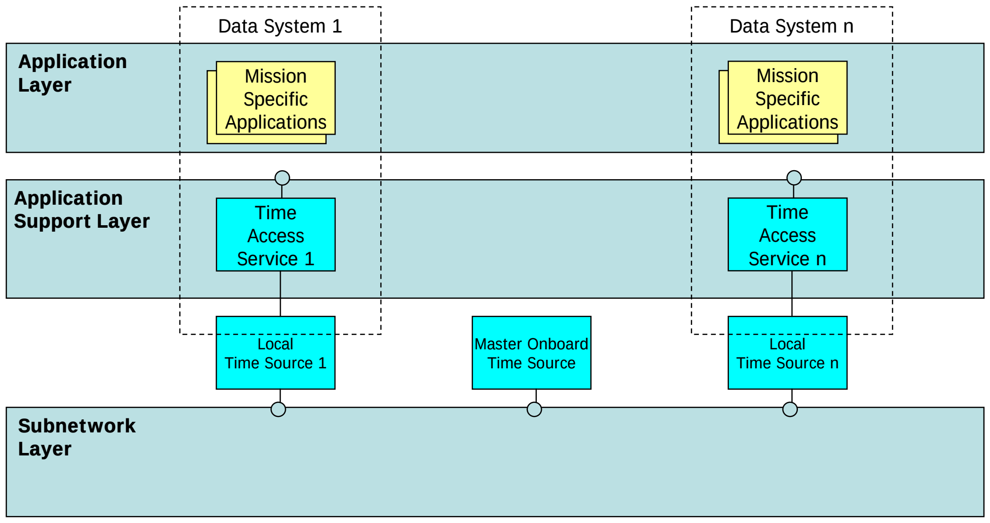

Figure 3-4: Time Access and Synchronisation Services Usage Scenario

In this architecture the local time sources are typically free-running hardware counters accumulating seconds and sub-seconds of elapsed time. Each of these counters is driven by its own oscillator, and the absolute frequency and frequency stability of each oscillator are different in each data system. The master onboard time source, the reference for onboard time for all onboard mission operations, is usually a similar free-running counter driven by an oscillator with precise absolute frequency and high stability. The value provided by this time source is usually called the Mission Elapsed Time (MET).

The TAS defines a standard interface between applications hosted on each data system and the local time source for that data system (the scope of TAS is indicated by the dashed boxes in figure 3-4).

The benefit is that all service users have a uniform interface to the local time source, regardless of their location on the spacecraft, and have no need to access local hardware directly. This simplifies the development of applications and allows them to be relocated if necessary and reused in other missions.

3.3.2 FUNCTIONS PERFORMED

The basic capability provided by the TAS is:

– ‘wall clock’ capability, which enables the application to read the time on demand.

Two optional extensions that reflect common requirements for onboard software systems are also defined:

– ‘alarm clock’ capability, which enables the application to request notification at a particular time;

– ‘metronome’ capability, which enables the application to request periodic notifications with a specified interval and starting at a particular time.

3.3.3 REQUIREMENTS FOR UNDERLYING SERVICES

The TAS requires access to a local time source. Where the spacecraft has more than one data system connected to a subnetwork or a network, correlation between the local time source and a master onboard time source may be required. It is assumed that this will be provided by an implementation-specific solution.

3.4 MESSAGE TRANSFER SERVICE

3.4.1 GENERAL

The Message Transfer Service (MTS) enables applications hosted onboard a spacecraft to communicate with each other using asynchronous, ad-hoc, discrete messaging with a bounded latency, including querying, multicast (‘publish’) and broadcast (‘announce’), independent of their locations. The MTS can be used for direct exchange of information or synchronisation between applications or as the basis of higher-level services, e.g., application frameworks. It is a goal of MTS to be efficient and predictable so as to support modelling of onboard software task and communication scheduling.

It should be noted each service user must be uniquely identified by an ‘MTS Node Identifier’. The transfer of messages between service users may be determined by QoS parameters that, in conjunction with the location of the service users, will determine the underlying service used and its associated QoS parameters.

3.4.2 FUNCTIONS PERFORMED

The MTS offers the following functions:

– send a discrete message to another application at a particular priority;

– receive the next queued discrete message: this is the next queued message waiting to be received, the next message being determined in priority order, FIFO with a priority level;

– send a query message to another application at a particular priority and receive a reply message back.

Two optional extensions are also defined:

– multicast a discrete message to all applications within a defined group, i.e., the publish-subscribe pattern;

– broadcast a discrete message to all applications, also known as announce.

The Asynchronous Message Service (AMS) (reference [8]) provides a general service definition that, when mapped onto a predictable Real-Time Operating System (RTOS) and communication services, meets the general and QoS requirements for MTS. Therefore MTS adopts the AMS service interface and recommends a profile of the AMS protocol specification providing predictable, prioritised delivery of messages, mapped onto an appropriate underlying communication service for exchange of AMS PDUs.

3.4.3 REQUIREMENTS FOR UNDERLYING SERVICES

Where the sending and receiving applications reside on different processors, the underlying services provide a service to **transfer discrete messages** between MTS implementations on the different processors. The underlying service must provide for uniquely identifying the source and destination processors, e.g., **addresses**. To support the real-time properties required by onboard applications, the underlying services must provide **bounded worst-case delivery times** and **prioritised delivery** of the discrete messages.

The MTS must provide a **queuing mechanism** for the messages awaiting reception by a receiving application. This mechanism must deliver the messages in priority order, FIFO within a priority level. Such a mechanism may be provided by an underlying RTOS.

Where the sending and receiving applications reside on the same processors, the message should be placed into the queuing mechanism for the messages awaiting reception by a receiving application. It is an MTS implementation decision how this is achieved.

3.5 FILE AND PACKET STORE SERVICES

3.5.1 GENERAL

The File and Packet Store Services (FPSS) are for use by service users to access, manage, and transfer within a spacecraft files and packets that could contain any type of data, including telemetry, commands, and command sequences, software updates, imagery, and other science observations. In addition to the general SOIS goal of more reusable applications and tolerance for changes in the spacecraft hardware configuration, use of the FPSS will make it easier to control access and management of shared hardware resources (e.g., mass memories).

A basic concept of this service is the definition of a file or a packet store. A file is a named data set residing in a file store. The packet store may or may not be aware of, and its actions informed by, the contents of a packet, e.g., fields within the packet header. A file or a packet store comprises:

– a memory element in which files or packets reside; this may be implemented in, e.g., local processor memory, hard disk stores, or a dedicated mass memory subsystem;

– an associated file or packet store system providing functionality for managing the files or packets.

It should be noted no assumption is made about persistence of files or packets in the file or packet store or any file or packet replication strategies.

As indicated in figure 3-5, the FPSS comprises the following categories of service:

– File Access Service (FAS)—allows access to files and portions of their contents in a file store;

– File Management Service (FMS)—allows manipulation of existing files in a file store;

– Packet Store Access Service (PSAS)—allows storage, retrieval, and deletion of packets in a packet store;

– Packet Store Management Service (PSMS)—allows creation and deletion of packet stores.

The File Store may consist of the following:

– Local File System and Remote Data Storage—the file system functionality is local to the SOIS File Services so that it resides upon the user’s data system. The data for the file system is held remotely in data storage, e.g., a simple mass memory, with some Remote Block Storage Protocol used by the file system to access the data storage across a (sub-)network using the Subnetwork Packet or Memory Access Service.

– Remote File System and Data Storage—the file system functionality is remote to the SOIS File Services so that a Network File Access Protocol is used to access the file system, e.g., on an advanced mass memory, across a (sub-)network using the

Subnetwork Packet Service. The data storage used by the file system is local so that accesses are internal to that data system.

– Remote File System and Remote Data Storage—the file system functionality is remote to the SOIS File Services so that a Network File Access Protocol is used to access the file system, e.g., on another OBC, across a (sub-)network using a Packet Service. The data for the file system is held remotely in data storage, e.g., a simple mass memory, with a Remote Block Storage Protocol used by the file system to access the data storage across a (sub-)network using the Subnetwork Packet or Memory Access Service. The (sub-)network across which the Network File Access Protocol is used may be different from the (sub-)network across which the Remote Block Storage Protocol is used.

These are illustrated in figure 3-5.

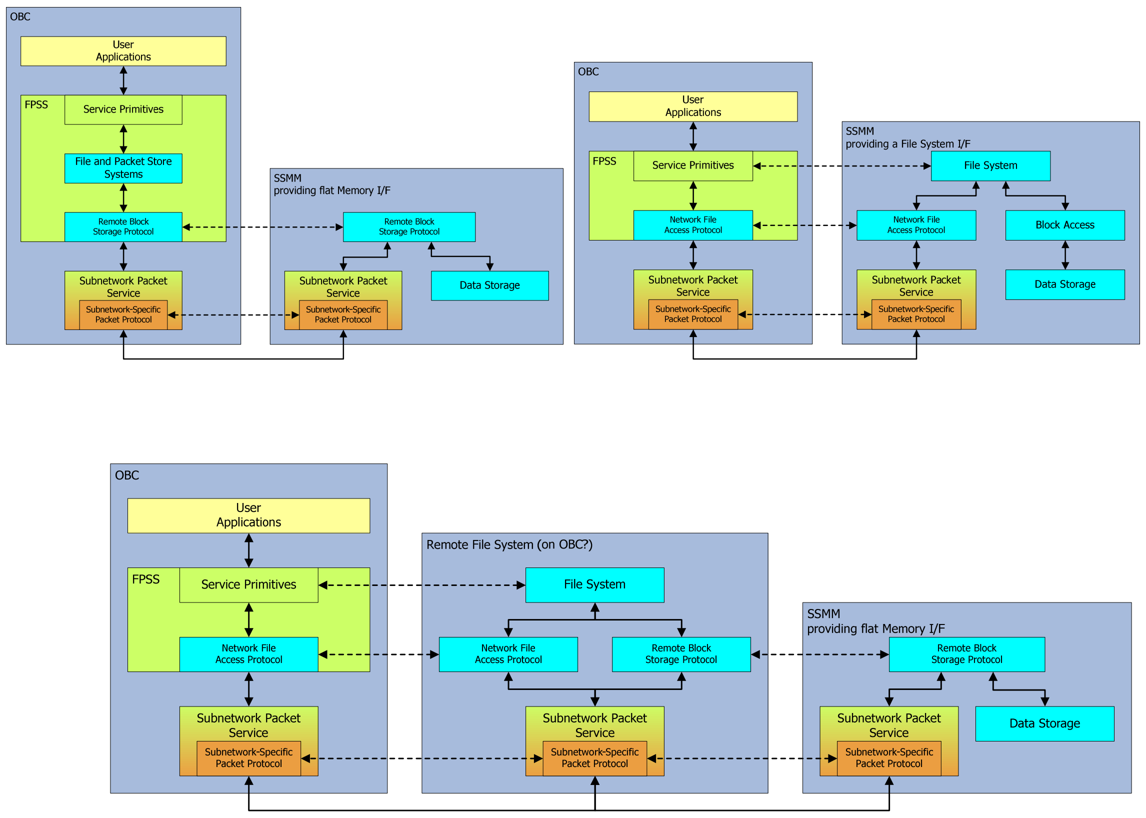
OBC
User Applications
FPSS
Service Primitives
File and Packet Store Systems
Remote Block Storage Protocol
Subnetwork Packet Service
Subnetwork-Specific Packet Protocol
SSMM providing flat Memory I/F
Remote Block Storage Protocol
Data Storage
Subnetwork Packet Service
Subnetwork-Specific Packet Protocol
OBC
User Applications
FPSS
Service Primitives
Network File Access Protocol
Subnetwork Packet Service
Subnetwork-Specific Packet Protocol
SSMM providing a File System I/F
File System
Network File Access Protocol
Block Access
Data Storage
Subnetwork Packet Service
Subnetwork-Specific Packet Protocol
OBC
User Applications
FPSS
Service Primitives
Network File Access Protocol
Subnetwork Packet Service
Subnetwork-Specific Packet Protocol
Remote File System (on OBC?)
File System
Network File Access Protocol
Remote Block Storage Protocol
Subnetwork Packet Service
Subnetwork-Specific Packet Protocol
SSMM providing flat Memory I/F
Remote Block Storage Protocol
Subnetwork Packet Service
Subnetwork-Specific Packet Protocol
Data Storage

Figure 3-5: Alternative File and Packet Store Services Architectures

Each of these distinct services and required underlying services is described in more detail in the following subsections.

3.5.2 FILE ACCESS SERVICE

3.5.2.1 Overview

The FAS provides for the service user to access files and portions of their contents in a file store regardless of its location; i.e., the accessed files can reside in a file store on the spacecraft.

3.5.2.2 Functions Performed

The FAS provides the following capabilities:

– open file;

– close file;

– read from file;

– write to file.

3.5.2.3 Requirements for Underlying Services

Where the file store is located on a different data system from that of the service user, the FAS requires a Network File Access Protocol, carried for example by a subnetwork packet service, to access a Minimum File Store capability, as defined in 3.5.3.3, resident on the remote data system.

Where the file store is located on the same data system to the service user, the FAS requires the provision of a Minimum File Store capability as defined in 3.5.3.3.

3.5.2.4 Minimum File Store Capability

To enable interoperability, the FAS requires a file store to provide the following minimum set of capabilities to be used to access, manage, and transfer files:

– list contents of directory;

– create file;

– open a file;

– close a file;

– read from file;

– write to file;

– delete file;

– move file;

– copy file.

Optionally, the file store may also provide the following capabilities:

– create directory;

– remove directory;

– rename directory;

– get current directory;

– lock and unlock files;

– move current user position within file;

– get file status.

The file system shall maintain the following attributes to a file:

– name;

– creation time;

– last write time;

– lock status;

– file size.

In an FAS implementation, the service primitives must then be mapped to and from the actual file system available. This approach allows complete independence from the technology used to implement the file store. Of course, the way in which this mapping is performed is implementation specific.

3.5.3 FILE MANAGEMENT SERVICE

3.5.3.1 Overview

The FMS allows service users to manipulate existing files in a file store regardless of location; i.e., the accessed files can reside in a file store on the spacecraft.

3.5.3.2 Functions Performed

The FMS provides the following capabilities:

– list directory contents;

– create file;

– delete file;
– copy file;
– move file.

Optionally, the FMS may provide the following capabilities:

– create directory;
– get current directory;
– change directory;
– delete directory;
– rename directory;
– lock file;
– unlock file;
– find file;
– seek within file;
– get file status.

3.5.3.3 Requirements for Underlying Services

Where the file store is located on a different data system from that of the service user, the FMS requires a Network File Access Protocol, carried for example by a subnetwork packet service, to access a Minimum File Store capability, as defined in 3.5.3.3, resident on the remote data system.

Where the file store is located on the same data system as the service user, the FMS requires the provision of a Minimum File Store capability, as defined in 3.5.3.3.

3.5.4 PACKET STORE ACCESS SERVICE

3.5.4.1 Overview

The Packet Store Access Service (PSAS) provides for the service user to store, retrieve and delete packets in a packet store regardless of its location; i.e., the accessed packets can reside in packet stores local or remote to the service user on the spacecraft.

A stored packet is, at a minimum level of complexity, a delimited data structure. Depending on the packet type, various structural elements of the packet may be accessible to the PSAS. These include packet identifiers, protocol identifiers or routing information. Examples of such packets are CCSDS packets, IPv4 and IPv6 packets, or CCSDS encapsulation packets.

Packets can be time-stamped with the time the packet was stored, a user-supplied time, or a time embedded within the packet.

A packet store can be organised as a bounded FIFO, an unbounded FIFO, or a random access store. Packet selection criteria are based on time-stamps associated with the packet or, where available, other fields within the packet header, or user-defined structures within the packet.

3.5.4.2 Functions Performed

The PSAS provides the following capabilities:

– get packet stores information;

– clear a packet store;

– write packets to a packet store;

– read packets from a packet store;

– move position in packet store;

– free packets from a packet store;

– report status of a packet store.

Optionally, the PSAS may provide the following capabilities:

– dump packets from a packet store to ground using the spacecraft telemetry system;

– select packets in a packet store;

– read selected packets in a packet store;

– free selected packets in a packet store;

– dump selected packets from a packet store to ground using the spacecraft telemetry system.

3.5.4.3 Requirements for Underlying Services

Where the packet store is located on a different data system from that of the service user, the PSAS requires a Network File Access Protocol, carried for example by a subnetwork packet service, to access a Minimum Packet Store capability, as defined in 3.5.4.4, resident on the remote data system.

Where the packet store is located on the same data system to the service user, the PSAS requires the provision of a Minimum Packet Store capability, as defined in 3.5.4.4.

3.5.4.4 Minimum Packet Store Capability

To enable interoperability, the PSAS requires a packet store to provide the following minimum set of capabilities to be used to store, retrieve, and delete packets:

– reset—delete all packets in a packet store;

– write—write *n* packets to the packet store;

– read—read the next *n* packets from the packet store;

– move by—move the user's next packet pointer backwards or forwards in the packet store;

– free—free the oldest *n* packets in the packet store;

– status report—report the status of the packet store.

Optionally, the packet store may provide the following capabilities:

– dump packets to ground using the spacecraft telemetry system;

– selective seek—create a new set of packets in the packet store meeting the specified criteria;

– selective read—read the next packets in a set in the packet store selected by the selective seek capability;

– selective free—free the oldest packets in a set in the packet store selected by the selective seek capability;

– selective dump—dump to ground using the spacecraft telemetry system the next packets in a set in the packet store selected by the selective seek capability.

3.5.5 PACKET STORE MANAGEMENT SERVICE

3.5.5.1 Overview

The PSMS allows the service user to create and delete packet stores, regardless of that packet store's location; i.e., it can be local or remote with respect to the service user on the same spacecraft.

3.5.5.2 Functions Performed

The PSMS provides the following capabilities:

– make a packet store;

– remove a packet store.

3.5.5.3 Requirements for Underlying Services

Where the packet store is located on a different data system from that of the service user, the PSAS require a Network File Access Protocol, carried for example by a subnetwork packet service, to access a Minimum Packet Store capability, as defined in 3.5.4.4, resident on the remote data system.

Where the packet store is located on the same data system to the service user, the PSAS requires the provision of a Minimum Packet Store capability, as defined in 3.5.4.4.

3.5.5.4 Minimum Packet Store Capability

To enable interoperability, the PSMS requires a packet store to provide the following minimum set of capabilities to be used to make and remove packet stores:

– make—make a packet store;

– remove—remove a packet store.

3.6 DEVICE ENUMERATION SERVICE

3.6.1 GENERAL

The DES provides management and user-notification of added or removed devices from a spacecraft. Management of added devices consists of assigning a (system-wide unique) virtual device identifier and verifying that the functions and configuration of the discovered device match the ones required by the system. Management of removed devices consists of revoking the virtual device identifier so that the user cannot access anymore the functions provided by the device.

The DES also provides a query function, allowing user applications to obtain the logical or physical identifiers of devices that match specified criteria, e.g., general criteria such as device type or specific criteria such as a serial number.

3.6.2 FUNCTIONS PERFORMED

The DES provides the following capabilities:

– Management of existing devices, through the following mechanisms:

  a) Enumeration of all devices.

  b) Enumeration of devices which match a criterion. This is optional.

– Management and user notification of added devices. This is optional and achieved through either or both the following mechanisms:

a) ‘Bottom-up’. Discovery of added devices and their associated metadata is provided by the Device Discovery Service (DDS) within the different subnetworks, and this is notified to the DES.

b) ‘Top-down’. Service users (higher level services or applications) configure the DES with an added device through manipulating the MIB.

– Management and user notification of removed devices. This is optional and achieved through either or both the following mechanisms:

a) ‘Bottom-up’. Discovery of removed devices is provided by the DDS within the different subnetworks and this is notified to the DES.

b) ‘Top-down’. Service users (higher level services or applications) configure the DES with a removed device through manipulating the MIB.

3.6.3 REQUIREMENTS FOR UNDERLYING SERVICES

For optional dynamic discovery, a DDS is required for each supported subnetwork type, and control of the MIBs of DVS and DAS is also expected.

4 SOIS SUBNETWORK SERVICES

4.1 INTRODUCTION

The SOIS subnetwork is decomposed into a number of layers and sublayers as illustrated in figure 4-1. The uppermost sublayer exposes a standard set of SOIS services to upper-layer applications. These services are supported by two underlying mechanisms:

– Where the capabilities provided by the underlying Physical and Data Link Layers already support the required SOIS service, a direct mapping of the data link service to the SOIS subnetwork is performed.

– Where the capabilities of the underlying Physical and Data Link Layers do not support the required SOIS service, additional functionally must be provided by the SOIS convergence sublayer in order to achieve the required SOIS subnetwork service. The convergence sublayer may require the provision of a protocol to provide the required functionality. While it is tempting to assume that a single protocol could be developed to operate over all types of Data Link Layers, in reality the requirements of the convergence sublayer protocol will be highly dependent on the diverse services provided by individual data link protocols.

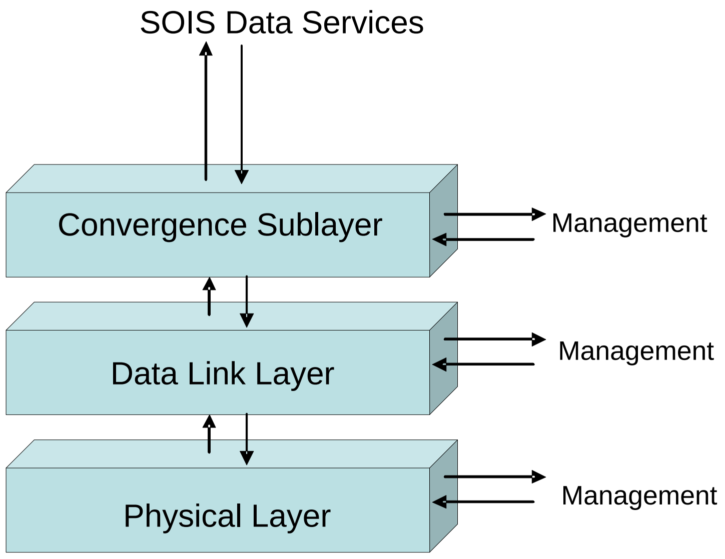

Figure 4-1: SOIS Subnetwork Decomposition

It should be noted that it may not be feasible, or desirable, to provide all SOIS subnetwork services over all Physical and Data Link Layer combinations. For example, a simple sensor interface is not likely to require or support packet or time distribution. To support such variation in capability, the SOIS services are defined independently of each other.

To provide a set of services that are feasible and realistic to support, account must be taken of implementation constraints and variability in design choice. For example:

- Priority of transfer may be supported by multiple queues at the Data Link Layer, with each queue assigned a certain priority. It could equally be implemented using a single queue at the Data Link Layer and assigning priority at a high layer of protocol. Alternatively, priority may be on a first-come-first-served basis.
- Data transfer may be guaranteed by the Data Link Layer protocol; alternatively, the Data Link Layer may provide a best-effort service which may be optionally improved by Transfer Layer retransmission protocols. Furthermore, for time-critical data transfers it may be undesirable to perform any retransmission, although retransmission may still be utilised with a bounded time after which the data is considered useless and retries are abandoned.

To account for this variability in implementation, the SOIS services include optional parameters as part of the service primitive. It is be up to the implementer to decide which optional parameters will be used.

Controlling the detailed effect of these parameters on the operation of subnetwork functions is a management function, as is the task of informing subnetwork users of the characteristics associated with each parameter. For example, it could be a management function to reserve resources in the subnetwork and assign a channel number to these resources, and then to inform the user of the subnetwork service as to the resources reserved and the channel number required to access them.

The following set of services has been selected for SOIS subnetwork standardisation:

- Packet;
- Memory Access;
- Synchronisation;
- Device Discovery;
- Test.

Directly accessed devices, e.g., using memory mapped I/O, are modelled as a subnetwork with the standard subnetwork services, primarily memory access, test, and device discovery, still being provided.

4.2 SOIS SUBNETWORK SERVICE DESCRIPTIONS

4.2.1 PACKET SERVICE

The SOIS packet service supports the transfer of data packets over a single bus/subnetwork while presenting a consistent, uniform interface to the service users. It enables the multiplexing of multiple network protocols with a range of QoS support over underlying data links. A variety of qualities of service are available, including priority, resource reservation, and reliability.

An implementation of the packet service must provide functions that are necessary to transfer data across a single bus/subnetwork. Required functionality of the underlying Data Link Layer may vary according to the requested QoS. For example, an implementation of this service over MIL-STD-1553B requires the following functions:

– subnetwork address translation, which translates the data destination address provided by the service user into a MIL-STD-1553B bus terminal address;

– protocol identification, which ensures that incoming data are directed to the appropriate entity in the Transfer or Application Support layers;

– segmentation, which breaks large data units into segments of data that can be transferred using a sequence of MIL-STD-1553B messages, each with a maximum size of thirty-two sixteen-bit data words.

By contrast, the implementation of this service to carry, for instance, IP packets reliably over a SpaceWire network requires the following functions:

– protocol multiplexing, if other types of protocols and services, such as SCPS packets or DAS PDUs, share the bus;

– address translation, which translates the data destination address provided by the service user into a data system address that is recognised on the SpaceWire network;

– resource reservation, which assures that the packet has the bandwidth or slot allocation required;

– retry function, such as LLC, which guarantees delivery or sends back status to the service user if delivery could not be completed;

– redundancy function, which provides an alternate path on the data link if the primary path is not available or healthy.

A major function of the packet service, and an essential one in time-critical installations, is to provide either prioritisation of PDUs or resource reservation.

The send interface to the packet service accepts the SDU to be transmitted along with parameters related to addressing and quality of service. Together, these parameters supply all the information needed for transporting the PDU, across a specific bus/LAN, to its destination with a given QoS.

Service class encompasses retry and resource reservation in a single QoS identifier that specifies to the packet service how an SDU is to be handled. There are four principal service classes supported: Best Effort, Assured, Reserved, and Guaranteed.

The Best Effort service class provides for non-reserved, try-once communication. It makes no promises about the time of delivery, the network bandwidth available, or the error rate of the traffic. Several priority levels can be provided for Best Effort traffic. Traffic with a higher priority level is treated preferentially compared to traffic with a lower priority level.

The Assured service class provides for non-reserved communication with retries. It tries to ensure that the traffic arrives at the intended destination. If the data does not arrive safely at the destination then it is resent. To support this, the destination acknowledges the receipt of Assured traffic. For Assured traffic the same priority levels as those for Best Effort traffic may be provided.

The Reserved service class provides for best-effort communication over a resource-reserved logical link. It is able to ensure the time of delivery and the network bandwidth available through the use of dedicated channelisation but makes no promises about the error rate of the traffic. Traffic may be lost but if it does arrive at the destination it will do so in a timely manner.

The Guaranteed service class provides for resource-reserved communications with retries. It is able to ensure the time of delivery, the network bandwidth available, and tries to ensure that the traffic arrives at the intended destination without error.

All of these traffic classes may be used in any combination in the same SOIS system.

The protocol multiplexing function supports the multiplexing and de-multiplexing of the different network protocols over the underlying data link links. Examples of protocols to be supported are IPv4, IPv6, and SCPS-NP. Abstract protocol identification is translated to subnetwork-specific capabilities in the Data Link Layer.

4.2.2 MEMORY ACCESS SERVICE

The SOIS memory access service provides the capability to read or write data from or to a memory or register location in a device. Data can be read/written one word at a time, or as a block of words that are loaded into contiguous memory locations on the target device.

This is an asymmetric service that is provided to the service user by functions in the subnetwork. The service can read data from or write data to devices that are:

– directly connected to the data system, e.g., through a serial digital interface or a SpaceWire link;

– directly connected to a bus that is attached to the data system, e.g., a MIL-STD-1553B data bus.

The service user must know the details of the physical port and/or channel to which the device is connected, and the memory/register mapping of the device where data is to be read written. The service provides an abstract hierarchy of device, memory identifier, and memory address.

The memory access service also provides for an atomic read/modify/write capability.

Quality of service aspects supported by this service relate to reporting the results of memory access operations and for verification of data prior to writing to memory.

4.2.3 SYNCHRONISATION SERVICE

The SOIS synchronisation service provides the capability to inform the subnetwork user of events within the subnetwork. The event may be an asynchronous event, e.g., a notification of a change in subnetwork configuration, or a synchronous event such as a time notification.

The service can be solicited by the user where, for instance, the instantaneous time is requested or can be unsolicited, e.g., where a network reconfiguration event is notified or where a regular time signal is produced. It may be necessary for the subnetwork user to register for reception of some synchronisation events via the SOIS management functions.

4.2.4 DEVICE DISCOVERY SERVICE

The SOIS device discovery service provides the capability to detect devices becoming active following a change in the hardware configuration of the spacecraft. This may occur when a cold redundant device is powered on, for example. This service discovers devices that are:

– directly connected to the data system, e.g., through an analogue or digital interface;

– directly connected to a bus that is attached to the data system, e.g., a MIL-STD-1553B data bus.

Device discovery may also be initiated by a user entity requesting that the subnetwork be scanned for devices present.

Also discovered with a device is its associated device metadata, describing the discovered device. The device metadata should include type characteristics, e.g., star tracker, reaction wheel, thermistor, or mass memory. It may also include specific device information, including identification of the manufacturer, the series, model, variant, part number, and unique device serial number.

4.2.5 TEST SERVICE

The Test Service is intended to be used for checking data system functionality and connectivity of the subnetwork. The service is used to check operation of the subnetwork aspects of the local data system as well as subnetwork connectivity to other data systems. The return parameters may be dependent on the inherent capabilities of the Data Link Layer protocol. As a minimum the service should return a go/no-go status, but this may be augmented by error codes, bit rate selection, prime/redundant media active, etc. The service does not indicate the correct operation of other subnetwork services but allows for the reporting of subnetwork-specific status.

4.3 SUBNETWORK FUNCTIONS

4.3.1 GENERAL

Subnetwork functions combine to provide subnetwork services by implementing a common set of functionality, described in the following subsections. Not all buses/subnetworks provide the full set of required Subnetwork functionality. Where necessary, the Convergence layer provides the missing functionality for each of the buses/subnetworks.

The identified functions which can provide the capability to support the full range of SOIS subnetwork services, either inherently by the data link or by the addition of convergence functions, are:

– redundancy;

– integrity;

– retry;

– segmentation;

– resource reservation;

– prioritisation;

– sequence preservation;

– protocol multiplexing.

In addition some functions are necessary to support specific services. These are:

– memory access;

– synchronisation;

– device discovery;

– test.

Figure 4-2 shows the Data Link Layer general purpose convergence functions required with reference to examples of data link types.

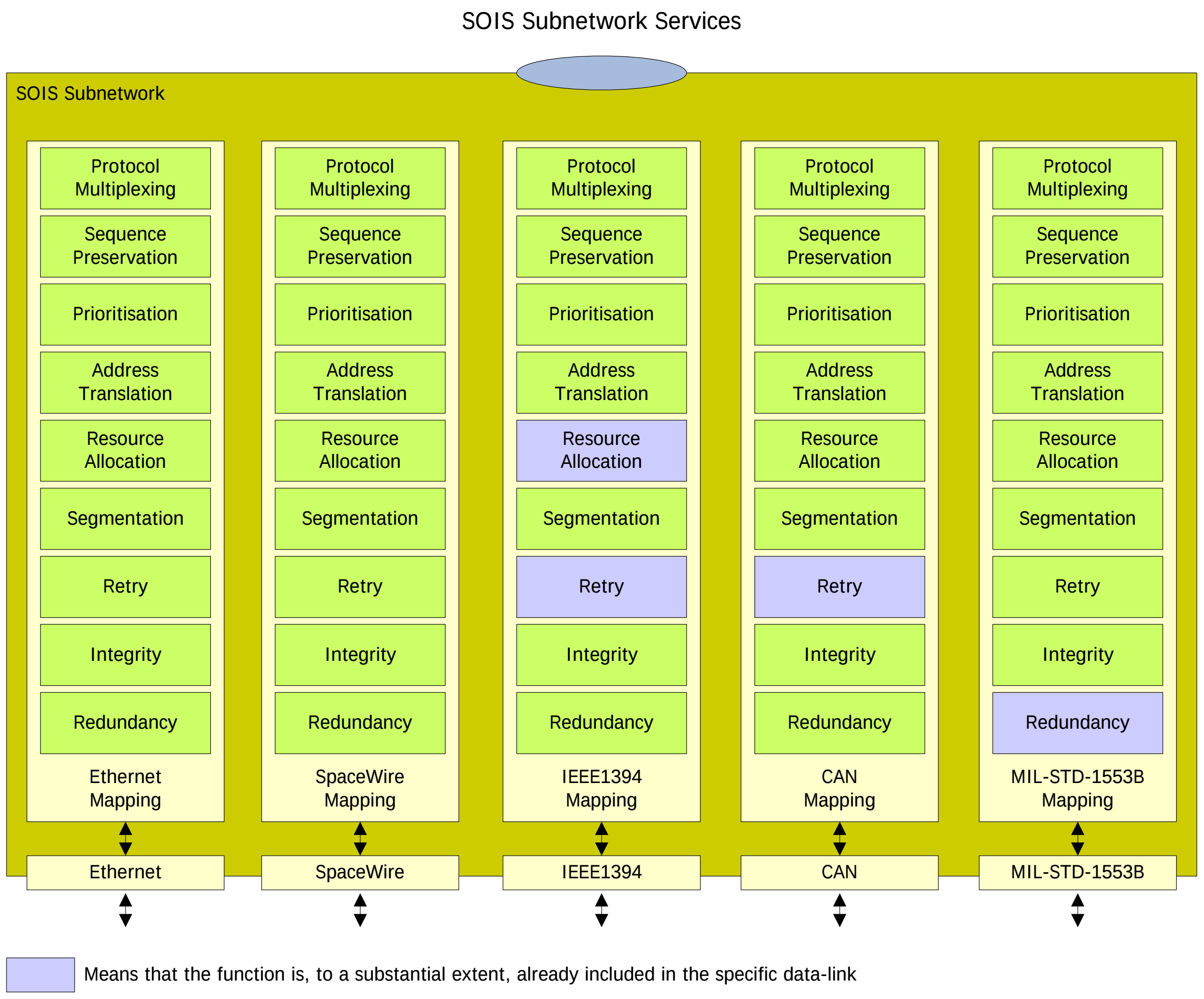

Figure 4-2: SOIS Data Link Convergence Functions

Each data link comprises a quality of service domain in which the QoS requirements are reconciled amongst all SOIS services and users supported by that data link. Functions within the subnetwork may be applied across all services even though these functions are not observable at the service interface. For instance, prioritisation and resource reservation are explicitly evident and may be selected on a per-SDU basis at the Packet Service interface. In order not to impact on Packet Service provision, other services that are provided in the subnetwork may be assigned priority or resource reservation levels dedicated to that service. In addition, for all services, prioritisation or resource reservation may be allocated on a per service user basis by management.

SOIS subnetworks do not support priority levels within a reserved resource. This would introduce a level of complexity, particularly with regard to sequence preservation, that is regarded as unnecessary in the SOIS environment. For this reason, a single Criticality QoS parameter is used which may be interpreted as either a priority or as a reserved resource depending on the capability of the subnetwork. It should also be noted that prioritisation or resource reservation may be selected within the subnetwork by reference to, for instance, the identity of the service user (i.e., the Service Access Point). This relationship would be established via subnetwork management. It should be noted segmentation is used in the general sense as is prevalent within CCSDS, to divide large SDUs into smaller units suitable for transmission over the subnetwork, rather than in the specific sense used by TCP to indicate stream segmentation.

4.3.2 CONVERGENCE FUNCTIONS

4.3.2.1 Redundancy Function

The redundancy model adopted by SOIS is that of equivalent data links that provide alternative paths from a source end-point to a destination end-point on a single subnetwork. The architecture supports autonomous switching between equivalent data links. Non-autonomous switching of paths can be done by an application specifying the bus/link/path to be used (using different destination addresses) or reconfiguring routing tables used to select which link is used for a particular address. Applications may control autonomous redundancy and link-level retry by using a management parameter associated with the transmit service class. It should be noted that system management policy might uniformly dictate a redundancy policy which applications must use.

Link equivalence requires two independent paths to a destination. These redundant paths may be used in one of three ways:

a) sending data over both paths at the same time;

b) sending over the prime link and then if there is a failure using the redundant link (often used for MIL-STD-1553B bus);

c) sending over either link, and then if there is failure of one link using the redundant link for all traffic.

The link redundancy function is bus/subnetwork specific. Users of the subnetwork service may not select the redundancy function on a data-unit-by-data-unit basis. However, use of the redundancy function may be defined on a user-by-user basis by management.

4.3.2.2 Integrity Function

The integrity function delivers data units without errors. That is to say, it discards data units within which errors are detected. Other functions (e.g., retry) may or may not be informed of instances of data discard depending on implementation.

4.3.2.3 Retry Function

The link-level retry function provides a mechanism for resending PDUs that are not received at the other end of the data link, either through data unit loss or data unit discard due to errors. The retry mechanism itself is subnetwork specific. Consideration should be given to any QoS parameters when setting the time-out and retry values, i.e., packets that have bounded latency requirements. If multiple copies of the same PDU arrive at the destination, only one copy of an SDU will be delivered to the user. The retry function is necessary to provide the formal quality of service of ‘completeness’.

4.3.2.4 Segmentation Function

Segmentation is needed if the underlying data link cannot support the maximum SDU size in a single packet on the data link. Segmentation may also be necessary to reduce latency caused by large data units in the data link queues. It is the data link’s responsibility to segment the PDUs if necessary and to reassemble them at the other end of the data link to reform the original PDUs before they are passed to the user of the Subnetwork Layer service. It should be noted, as all services are provided without error (via the integrity function), loss of a segment will result in loss of the whole packet unless retransmission is implemented at the segment level in the subnetwork. The segmentation function is bus/subnetwork specific.

4.3.2.5 Resource Reservation Function

Resource reservation assigns data link resources to subnetwork traffic. It is able to ensure the time of delivery and the network bandwidth available through the use of dedicated channelisation. Although this function is only explicitly invoked by the Packet Service, PDUs generated in support of other services will need to conform to channelisation if it is present in support of the Packet Service. The channelisation for these services is controlled by management and may be on a per service or per SAP basis. By this management mechanism, all services may make use of the resource reservation function even if it is not invoked by explicit service primitive parameters,

Implementation of the resource reservation function is data link specific.

4.3.2.6 Prioritisation Function

The Prioritisation function queues incoming SDUs for transmission in priority order, that is to say, highest priority first, with FIFO ordering of SDUs within the same priority level. It is informed by the Criticality parameter in the subnetwork data service request.

4.3.2.7 Sequence Preservation

Sequence Preservation guarantees the formal ‘in sequence’ quality of service. Typically, this means the implementation of a sequence auditing mechanism (typically a counter) and discarding PDUs that are received out of sequence.

Alternatively it is also possible to buffer PDUs and to wait for missing PDUs when a sequence gap is detected. This mechanism is typically implemented in conjunction with the retry function when providing Assured and Guaranteed Packet Service classes. It introduces variable delays into SDU delivery which may be undesirable.

4.3.2.8 Protocol Multiplexing Function

The protocol multiplexing function allows multiple Network or higher-layer entities to access Subnetwork Layer services. This is achieved via a protocol identification capability specific to the subnetwork. This capability is used to direct service indications to the appropriate Network or higher-layer entities.

4.3.3 SERVICE SPECIFIC FUNCTIONS

4.3.3.1 Memory Access Function

An implementation of the memory access service must provide functions to write data to or read data from locally connected devices as well as to provide the QoS aspects. These functions are specific to the actual device interfaces that are used.

4.3.3.2 Synchronisation Function

An implementation of the synchronisation service requires a synchronisation function that will depend on the nature of internal events in the subnetwork that can be conveyed to the subnetwork user. In the case of a time notification and depending on required accuracy, the function may be implemented by any combination of time messages in the subnetwork, dedicated timing infrastructure in the subnetwork, and locally maintained time sources inside the data systems of the subnetwork.

4.3.3.3 Device Discovery Function

An implementation of the device discovery service must provide functions to detect subnetwork/device configurations, e.g., by detecting subnetwork events or by periodically scanning for attached devices. These functions are specific to the subnetwork to which devices may be attached.

4.3.3.4 Test Function

The test function implements the test service. This can range from a simple data system present or not present test to more complex diagnostics.

5 ELECTRONIC DATASHEETS AND DICTIONARY OF TERMS

5.1 OVERVIEW

Electronic Data Sheets are essentially a machine readable form of a paper Interface Control Document (ICD) of a device, software component, or protocol. The intent is to provide a standardised way of defining information such that it can be used to automate many of the tasks that are currently performed manually. This includes device selection by spacecraft designers, perhaps from an industry catalogue, automatic generation of software or hardware interfaces and protocol implementations, and direct import into spacecraft-related databases that describe the content of specific spacecraft or missions. This intent applies equally to flight systems and ground systems. Figure 5-1 shows a very simplified view of how device and software EDSes can support different mission development and operations functions across the entire mission life cycle, from mission definition to on-orbit maintenance. A SOIS EDS is a source of inputs to all of these functions, but as discussed in the following paragraphs it will be defined in the narrow context of a device utilizing the SOIS stack up to the Device Virtualization Service.

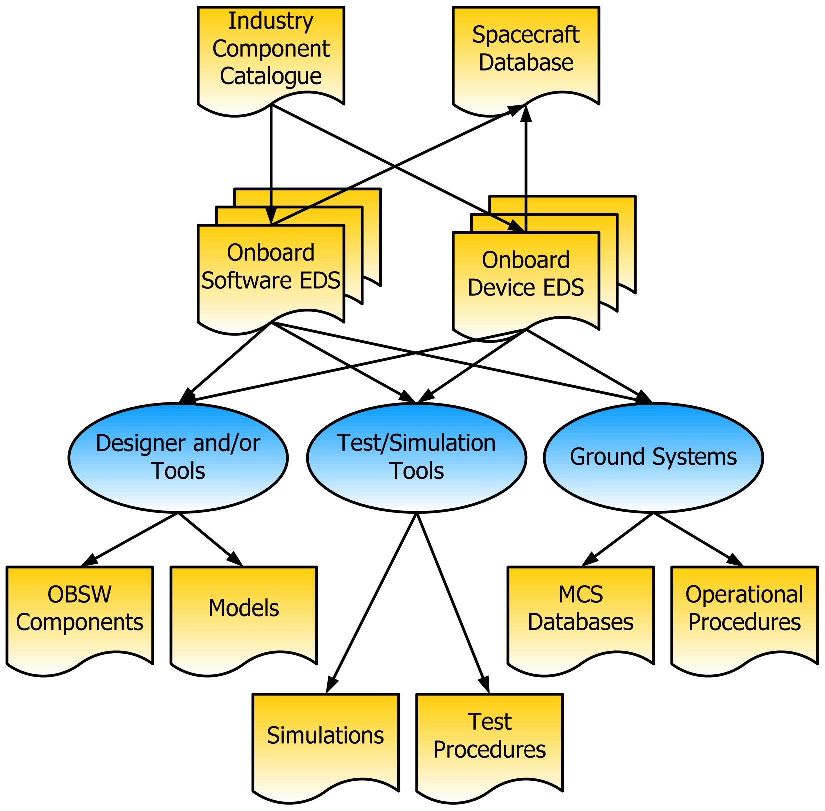

Figure 5-1: Mission Use of EDS

5.2 SOIS EDS DESCRIPTION

The process to develop the SOIS EDS is still in formulation but it is expected that the SOIS Area will create a DoT and a tightly defined SOIS EDS schema expressed in an eXtensible

Markup Language (XML). To support the DoT and schema developments several prototype EDSes need to be created for a sample of devices such as star trackers, GPS units, and reaction wheels. The relationship of the DoT, generic DVS, and device-specific EDS are shown in figure 5-2.

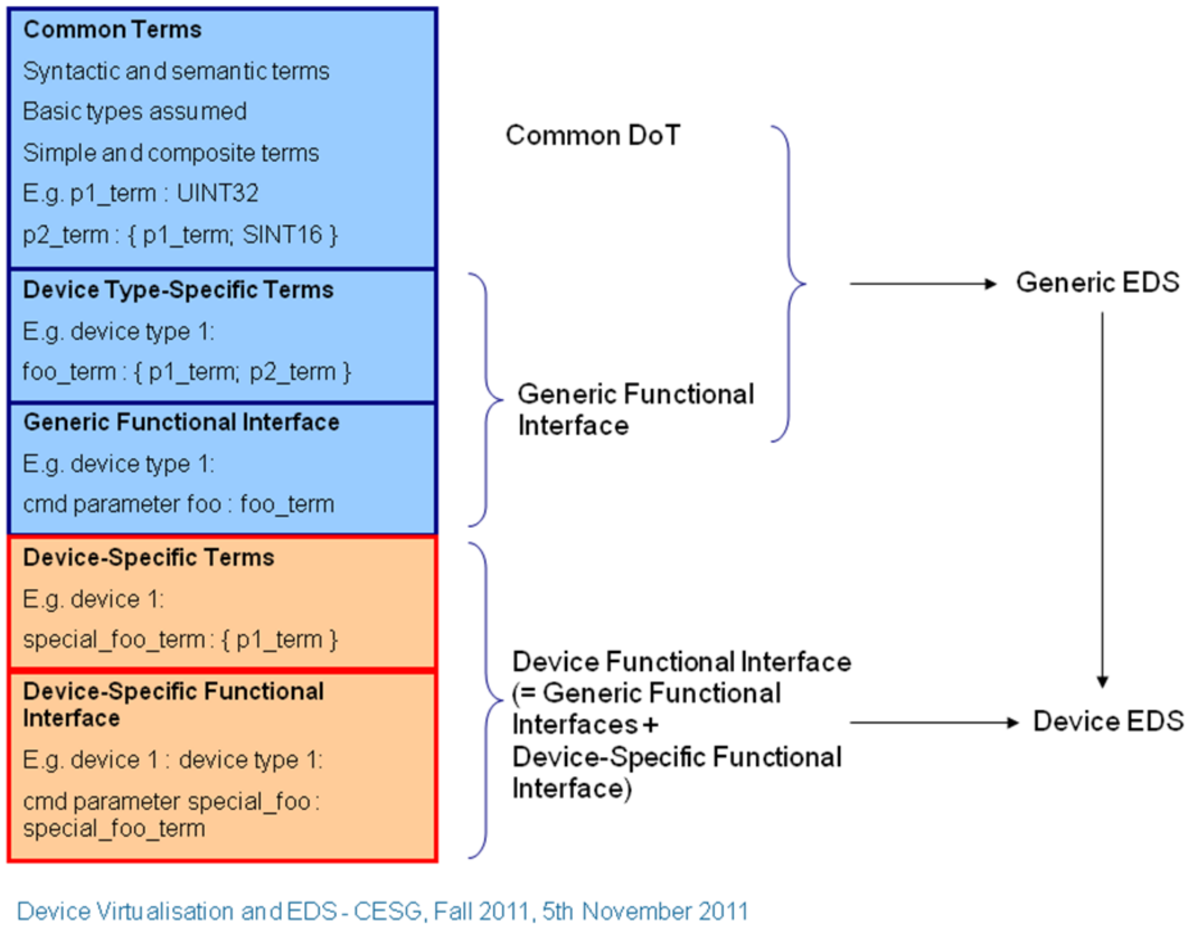
Common Terms
Syntactic and semantic terms
Basic types assumed
Simple and composite terms
E.g. p1_term : UINT32
p2_term : { p1_term; SINT16 }
Device Type-Specific Terms
E.g. device type 1:
foo_term : { p1_term; p2_term }
Generic Functional Interface
E.g. device type 1:
cmd parameter foo : foo_term
Device-Specific Terms
E.g. device 1:
special_foo_term : { p1_term }
Device-Specific Functional Interface
E.g. device 1 : device type 1:
cmd parameter special_foo : special_foo_term
Common DoT
Generic Functional Interface
Generic EDS
Device Functional Interface (= Generic Functional Interfaces + Device-Specific Functional Interface)
Device EDS
Device Virtualisation and EDS - CESG, Fall 2011, 5th November 2011

Figure 5-2: DoT and EDS Relationships

A full EDS completely describes the operation of the DAS and DVS services, such that it could replace a paper datasheet entirely. Such an EDS permits all, or portions of, DAS and DVS to be generated for a given device. This generation could be human- or tool-driven. The EDS encompasses the DVS interface description, described above, which can be used to ensure application portability. An EDS contains three distinct aspects:

– **Interface Descriptions**. The EDS must describe all of the operations that are possible for this device. For example, a reaction wheel device should provide an operation to command the torque; a sun sensor should provide an operation to acquire the angle to the sun. Thus it describes what the valid DVS and DAS command and acquisition identifiers are.

– **Protocol and Procedure Descriptions**. An EDS must also describe how the operations which form part of the interface are mapped onto the underlying services.

For DAS this mapping must include the data formats and interactions which form the Device-specific Access Protocol. For DVS this is likely to include conversions from device-specific data types to standard data types with engineering units as well as any device interactions required to provide the functional interface, together forming the Device Abstraction Control Procedure.

– **Documentation**. Rather than relying on an additional paper datasheet, a SOIS EDS may also contain documentation. This documentation may be extracted from the electronic datasheet using suitable tools to produce the equivalent of a paper datasheet, with descriptive text, diagrams, etc. By including documentation in the EDS there is a single source for information about the device, ensuring consistency, and descriptive documentation elements can be associated with interface and implementation descriptions allowing connections to be formed between the human-readable and machine-readable elements of the EDS.

A complete datasheet contains sufficient information to permit the necessary implementations for DVS and DAS to be generated using only datasheet information. With sufficiently capable tools this generation could be entirely automatic. It is not required for a single EDS to completely define all aspects of a device. Partial datasheets may be used to, for example, accompany manually generated implementations to support tooling, or permit the generation of DAS only.

To promote interoperability, the SOIS EDS standard is aligned with two existing, or currently evolving, EDS standards:

– The portion of the EDS describing the DVS interface aligns well with the evolving Space Plug-and-Play Avionics (SPA) standards developed by the US AFRL and proposed for AIAA standardisation. It is intended to fully align the SOIS EDS with this work such that the SOIS EDS is a strict superset of the SPA xTEDS EDS format. This alignment is aided by the choice of XML as the representation for the SOIS EDS.

– The portion of the EDS describing the function of DAS and DVS is functionally aligned with the IEEE 1451.0 TEDS standard. The SOIS EDS is, again, a strict functional superset of the IEEE 1451.0 TEDS such that a TEDS could be machine translated to a SOIS EDS.

In order to be able to express protocols, to allow description of the DAP, the EDS permits description of machine representations of types, and mappings to constructs such as packets (for Packet Service use) and memory spaces (for Memory Access Service use). Machine types can be captured in the DoT to allow reuse between EDS instances. As a DAP is unlikely to be stateless, the EDS must be able to express protocol states and state progression rules, including features such as timeouts. The same expressions are used to describe device states or modes for both DVS and DAS implementations.

The logical positions of the EDS technology alignment with xTEDS and IEEE 1451.0 TEDS are shown in figure 5-3.

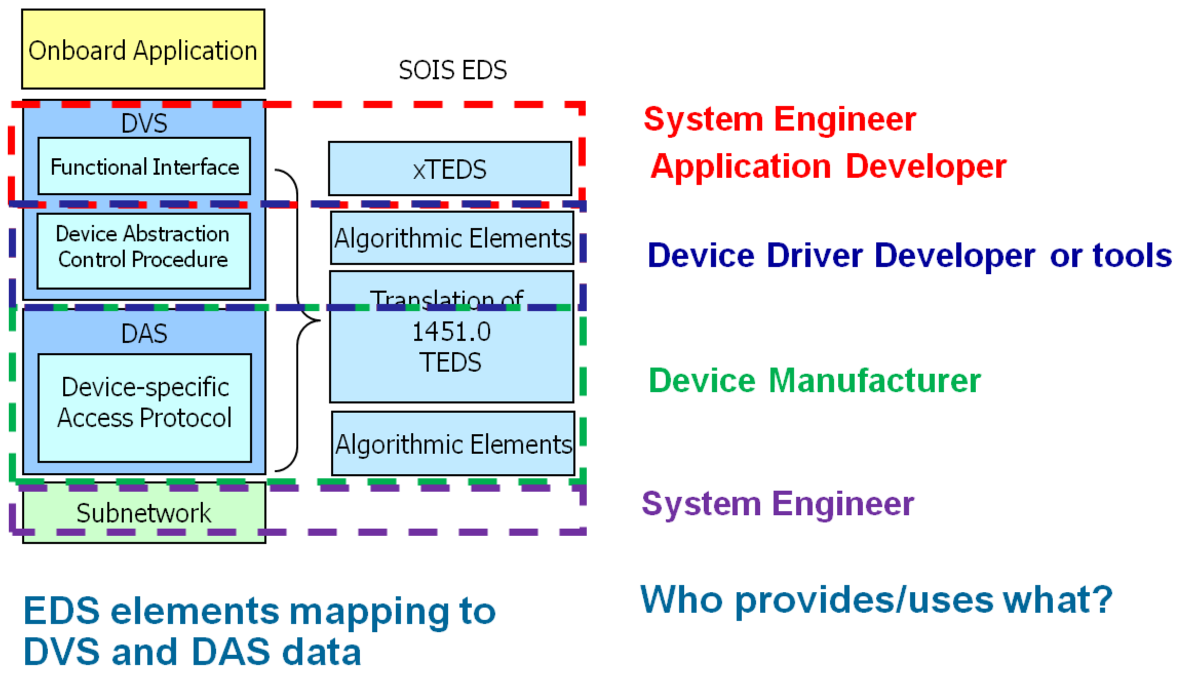

Figure 5-3: Alignment of EDS Technology with xTEDS and IEEE 1451.0 TEDS

5.3 DICTIONARY OF TERMS

Engineering profiles are essential for defining the meaning of operations and must be uniquely related to human-readable descriptions and physical quantities. This task is accomplished by the use of a Dictionary of Terms (DoT), or ontology, used to specify all valid terms that can describe *engineering profiles*. Engineering profiles can be used in an EDS to describe data in the functional interfaces provided by virtual devices and in the required functional interfaces of applications.

Engineering profiles are expressed as a list of values of semantic properties. The primary set of semantic properties is the following:

– unit of measure, for numbers that represent physical magnitudes;

– purpose, to distinguish measurements from set-points, among other things;

– syntactic type, such as 8-bit unsigned two's-complement integer;

– array length, such as three for a vector in non-relativistic space, which tells the number of items that are aggregated in an array to represent a multidimensional concept;

– reference frame, such as device, ECI, vehicle, and others;

– to-frame, which states the reference frame of the result of a transformation;

- coordinate type, which identifies the meanings of the coordinates in a concept represented by an array of coordinates; the array length is unnecessary for many coordinate types;
- subject, which identifies an object for which the data item is a property; the subject property is unnecessary when the to-frame or the reference frame is the device frame.

An engineering profile is represented by a list of specific values for some subset of the list above. For example, a quaternion produced by a star tracker could have an engineering profile specified by *reference frame = J2000*, *to frame = device*, and *coordinate type = quaternion*; the unit of measure, array length, and subject are inapplicable; and the purpose remains to be specified. The purpose might be specified as set-point when an attitude quaternion is specified as a set-point in a command to an attitude control system.

If an engineering profile is used frequently enough to merit having its own name, then it can be defined as a semantic type in the dictionary of terms. For example, some reaction wheels can be commanded by specifying a torque in SI units; its semantic type could be named *WheelTorqueSI*, representing the engineering profile, (*referenceFrame = device*, *axis = x*, *unit = Nm*, *quantityKind = torque*).

When an engineering profile includes syntactic attributes, such as encoding and number of bits, then we can say that it has been bound to syntax. Commonly used syntactic types may have names, such as "*INT16*" or "*FLOAT64*" in the DoT. The list above and the specific names of semantic properties may differ between this example and the actual dictionary of terms.

Because an engineering profile relates the data items to engineering concepts, a designer can use this to adapt the data items in an interface of a device to an appropriate application for consuming the data or commanding the device. For example, a GPS receiver's functional interface presents an accurate time stamp, so a designer can match that part of the interface to requirements elsewhere in a system where there is a need to receive accurate time.

5.4 STANDARDISATION OF ASPECTS OF DEVICE INTERFACES

As mentioned in 2.6, with device virtualisation, standardising the functional interface for device types[1] into a generic device image is possible (where there are common functions across devices of the same or similar type), known as a *generic functional interface.*

Such an approach relies upon a single **common** DoT.

The management procedure for the common DoT is defined in the corresponding CCSDS standard and is designed to be:

- universally accessible;

[1] The standardisation of device types is outside the scope of CCSDS. Such a standardisation may be based on those identified in the ETSI study (reference [14]).

- easy to update with new terms as necessary;[2]
- under the strict control of a management or editing authority to ensure terms are unambiguous and unique.

Furthermore, each standard virtual device and device type must be uniquely identifiable. The approach to this is defined in the CCSDS standard on Electronic Data Sheets, using both of the following mechanisms as appropriate for standard and ad hoc definitions.

- Use of centrally assigned organisation (vendor) identifiers together with organisation-assigned product identifiers to create a unique identifier. This is the approach used for PCI and USB and would require a single authority for ID assignment such as CCSDS SANA. This mechanism is appropriate for standard definitions.
- Use of algorithmically generated unique (or practically unique) identifiers such as the standard UUID. This removes the need for a naming authority but requires a larger numeric ID space to provide reasonable assurance of uniqueness (UUIDs are 128 bits). This mechanism is appropriate for ad hoc definitions.

[2] Part of the process of adding a new term to the dictionary is a search to determine that there is not already a term present that will provide the same semantics. Every new term comes with the cost that an interface that uses a new term is incompatible with older applications.

6 SECURITY

6.1 SECURITY BACKGROUND

The SOIS services are intended for use with protocols that operate solely within the confines of an onboard subnet. It is therefore assumed that SOIS services operate in an isolated environment which is protected from external threats. Any external communication is assumed to be protected by services associated with the relevant space-link protocols. The specification of such security services is out of scope of the SOIS Recommended Practices.

6.2 SECURITY CONCERNS

At the time of writing there are no identified security concerns. If confidentiality of data is required within a spacecraft it is assumed it is applied at the application layer. (For more information regarding the choice of service and where it can be implemented, see reference [12].)

6.3 POTENTIAL THREATS AND ATTACK SCENARIOS

Potential threats and attack scenario typically derive from external communication and are therefore not the direct concern of the SOIS services, which make the assumption that the services operate within a safe and secure environment. It is assumed that all applications executing within the spacecraft have been thoroughly tested and cleared for use by the mission implementer. Confidentiality of applications may be provided by Application Layer mechanisms or by specific implementation methods such as time and space partitioning. Such methods are outside the scope of SOIS.

6.4 CONSEQUENCES OF NOT APPLYING SECURITY

The security services are out of scope of this document and should be applied at layers above or below those specified in the SOIS Recommended Practices. If confidentiality is not implemented, science data or other parameters transmitted within the spacecraft might be visible to other applications resident within the spacecraft resulting in disclosure of sensitive or private information.

7 USE OF SOIS

7.1 OVERVIEW

This section presents recommended best practise on the use of SOIS for certain spacecraft system architectural issues.

7.2 SPACE LINK ACCESS

7.2.1 SPACE PACKET ROUTING BETWEEN THE SPACE LINK AND THE ONBOARD NETWORK

To receive a Space Packet (reference [3]) from the Space Link, there are two approaches the Monitoring and Control (M&C) interface of the Telemetry (TM)/Telecommand (TC) equipment may implement:

– Asynchronous Notification—Upon receiving the Space Packet, the TM/TC equipment can immediately, asynchronously route the packet over the Subnetwork Packet Service to an onboard application. This is the preferred option.

– Polled—The onboard application periodically polls the M&C interface of the TM/TC equipment to determine if a packet has been received and is buffered in the TM/TC equipment. The result may itself be the packet; otherwise the onboard application can then read the packet from the TM/TC equipment.

To send a Space Packet (reference [3]) across the Space Link, the onboard application should send the Space Packet over the Subnetwork Packet Service to the TM/TC equipment where a routing function will send it over the Space Link. Thus the onboard network can be treated as part of a single mission network, incorporating the ground network and the space links with the TM/TC equipment acting as a space packet router. The routing function is configured using commands sent using the DVS and DAS, e.g., to set the virtual channel upon which particular Space Packet are to be sent.

There are alternative architectures where an advanced packet store may itself send one or more packets directly to the TM/TC equipment, rather than an onboard application’s first retrieving them from the packet store.

The chosen approach should depend upon the manner in which the subnetwork is managed. Polling, e.g., used on MIL-STD-1553B, allows for predictable communications but can waste bandwidth. Asynchronous notification, e.g., used on vanilla SpaceWire and IP-based networks, is more reactive but can suffer from blocking or congestion.

The recommended SOIS architecture to enable onboard applications to control spacecraft TM/TC equipment over a subnetwork is to treat the TM/TC equipment as a virtualised device, as illustrated in figure 7-1. This consists of the following items:

- a packet protocol for communicating over the subnetwork between the TM/TC equipment and the OnBoard Computer (OBC) upon which the onboard application resides;
- a Subnetwork Packet Service implementation that uses the subnetwork-specific packet protocol to exchange packets between the TM/TC equipment and the OBC;
- a TM/TC equipment-specific access protocol that it used to provide a M&C interface to the TM/TC equipment;
- a Device Access Service implementation that maps data acquisitions and commands for the logical device identifier assigned to the TM/TC equipment to the TM/TC equipment-specific access protocol;
- a Device Virtualisation Service implementation that provides a TM/TC equipment device type mapped to the particulars of the M&C interface of the TM/TC equipment used on the spacecraft.

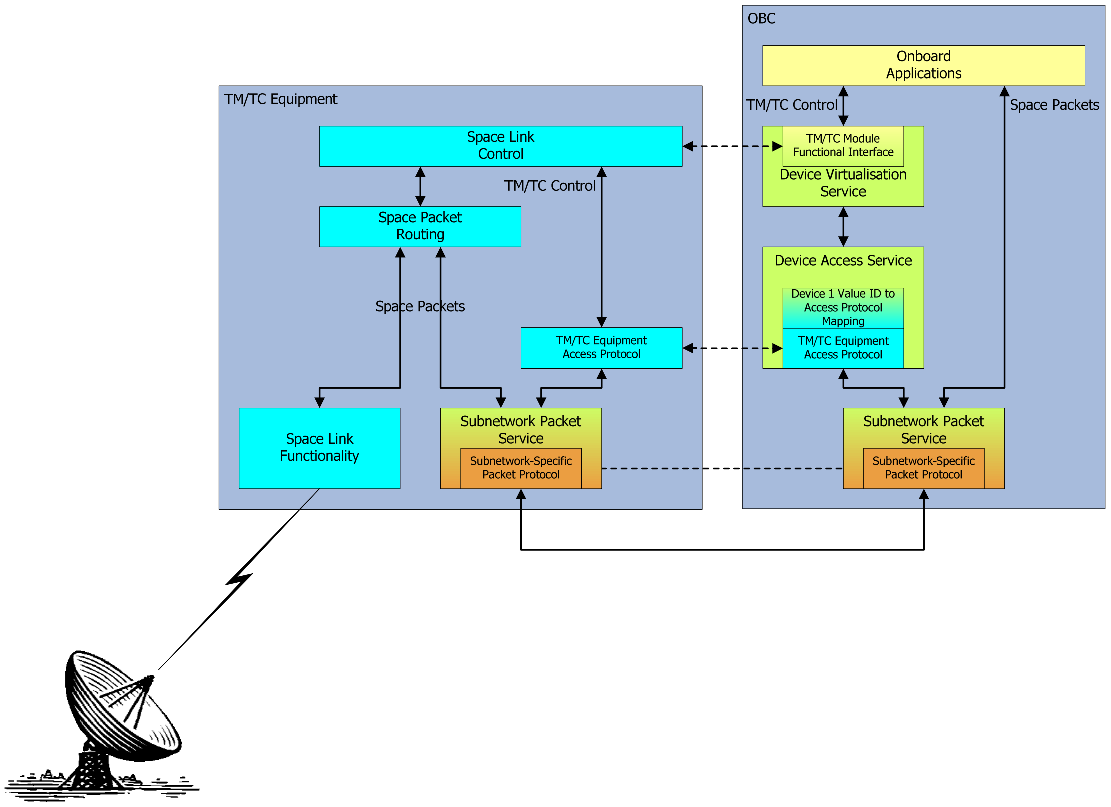
OBC
Onboard Applications
TM/TC Control
Space Packets
TM/TC Equipment
Space Link Control
TM/TC Module Functional Interface
Device Virtualisation Service
TM/TC Control
Space Packet Routing
Device Access Service
Device 1 Value ID to Access Protocol Mapping
Space Packets
TM/TC Equipment Access Protocol
TM/TC Equipment Access Protocol
Space Link Functionality
Subnetwork Packet Service
Subnetwork-Specific Packet Protocol
Subnetwork Packet Service
Subnetwork-Specific Packet Protocol

Figure 7-1: Recommended SOIS Architecture to Access TM/TC Equipment

7.2.2 IMPLEMENTING SPACE INTERNETWORKING PROTOCOLS

This subsection identifies recommendations on the use of the SOIS architecture to implement certain standard and/or typical space internetworking protocols:

- Packet-based message protocol, e.g., ECSS PUS (reference [15])—The architecture defined in 7.2.1 is recommended to receive telecommand packets and to send telemetry packets.
- CFDP (reference [7])—The architecture defined in 7.2.1 is recommended to send and receive CFDP PDUs to and from TM/TC equipment. Where Extended Procedures or the Store and Forward overlay are deployed, the CFDP implementation is responsible for handling multiple TM/TC equipment for different Space Links. CFDP also requires an onboard file system; it is recommended to use the SOIS File Access and Management Services for this purpose. This has the added advantage that onboard applications can use the same onboard file system as CFDP but directly (and therefore more efficiently) using the SOIS File Access and Management Services.
- DTN (reference [9])—The architecture defined in 7.2.1 is recommended to send and receive BP (reference [10]) or LTP (reference [11]) PDUs to and from TM/TC equipment. The BP implementation is responsible for handling multiple TM/TC equipment for different Space Links.

Where multiple space internetworking protocols are deployed upon a spacecraft, it is expected that the CCSDS Encapsulation Service (reference [4]) be used with the associated Protocol Identifier field used to de-multiplex the PDUs of the different space internetworking protocols for the different protocol implementations at the OBC.

7.3 AGGREGATION OF COMPONENTS VIA A REMOTE INTERFACE UNIT

A spacecraft onboard system can be generalized as a group of functions implemented by a respective group of avionics boxes interconnected via an onboard network. Typically there exists a ‘catch-all’ avionics box that collects group of functions often defined as ‘dumb’ sensors/actuators. Within some NASA spacecraft, many of these ‘dumb’ devices are defined by Transducer Electronic Data Sheets (TEDS) per IEEE-1451.

The IEEE-1451 TEDS specify the access service for commanding and gathering data for these devices (equivalent to the SOIS DAS interface), as well as how to interpret the raw sensor data into engineering units (equivalent to the SOIS DVS interface).

IEEE-1451 supports the aggregation of devices through a Smart Transducer Interface Module (STIM), which is functionally equivalent to a Remote Interface Unit (RIU) (also known by many other names). The STIM bridges the low-level sensor/actuator devices interface to the onboard network interface. This example would mean that the data gathering and translation function are performed in the STIM and the results provided over the network in the processed form via a packet. In this example, the STIM provides a new TEDS to the Network Capable Application Processor (NCAP). The STIM ‘hides’ the low-level details of the ‘dumb’ devices’ TEDS.

The system may also be constructed so that the ‘dumb’ devices’ TEDS may be ‘tunnelled’ through the STIM RIU. This would mean that the ‘smarts’ for controlling and translating the ‘dumb’ device’s data is in the NCAP (sometimes called Command & Data Handling [C&DH] for space systems) and not the STIM. The STIM in this example is just the concentrator or attachment point for communications of the ‘dumb’ devices back to the NCAP. The STIM TEDS in this example would consist of the multiplexing function of the individual TEDS of the ‘dumb’ devices and bridging them over the network (similar to bridging PCI over a serial bus, i.e., PCI express). This would consist of a ‘shim’ to the STIM TEDS to support the individual ‘dumb’ devices’ TEDS multiplexing function that the STIM provides.

It should be noted that the ‘dumb’ device in the previous examples can be replaced with a smart device and the process would be the same.

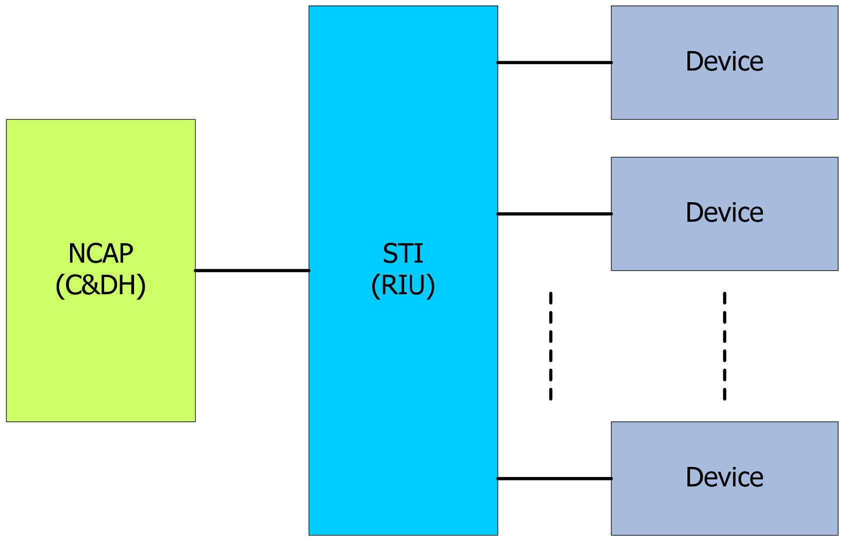

Figure 7-2: Aggregation of Components by IEEE-1451 via a Remote Interface Unit

Even though the IEEE-1451 TEDS is not in the SOIS Electronics Data Sheet (EDS) format, translation to the SOIS format is straightforward.

This provides an architecture for adoption by SOIS by which ‘dumb’ devices can be aggregated by an RIU, supported by a SOIS EDS for each device and for the RIU.

The EDS for each device contains the following:

– functional interface of the device;

– DAP.

The Functional Interfaces of the aggregated devices are ultimately offered to an onboard user via the RIU. However, the DAPs of the aggregated devices are specific to the interfaces between the RIU and the dumb devices. The DAP to an onboard user offered by the RIU for accessing the aggregated devices is specific to the RIU (it may be a simple tunnelling mechanism allowing for simple configuration for accessing different aggregated devices). Therefore the SOIS EDS of the RIU is composed of the following:

– functional interfaces of the RIU itself and the aggregated ‘dumb’ devices;

– DAP of the RIU, including how to access the RIU to access the aggregated ‘dumb’ devices.

7.4 ONBOARD SYSTEM AND SOFTWARE ARCHITECTURES

7.4.1 OVERVIEW

This subsection provides some example system and software architectures showing different SOIS deployment architectures.

7.4.2 CENTRAL SYSTEM AND SOFTWARE ARCHITECTURE

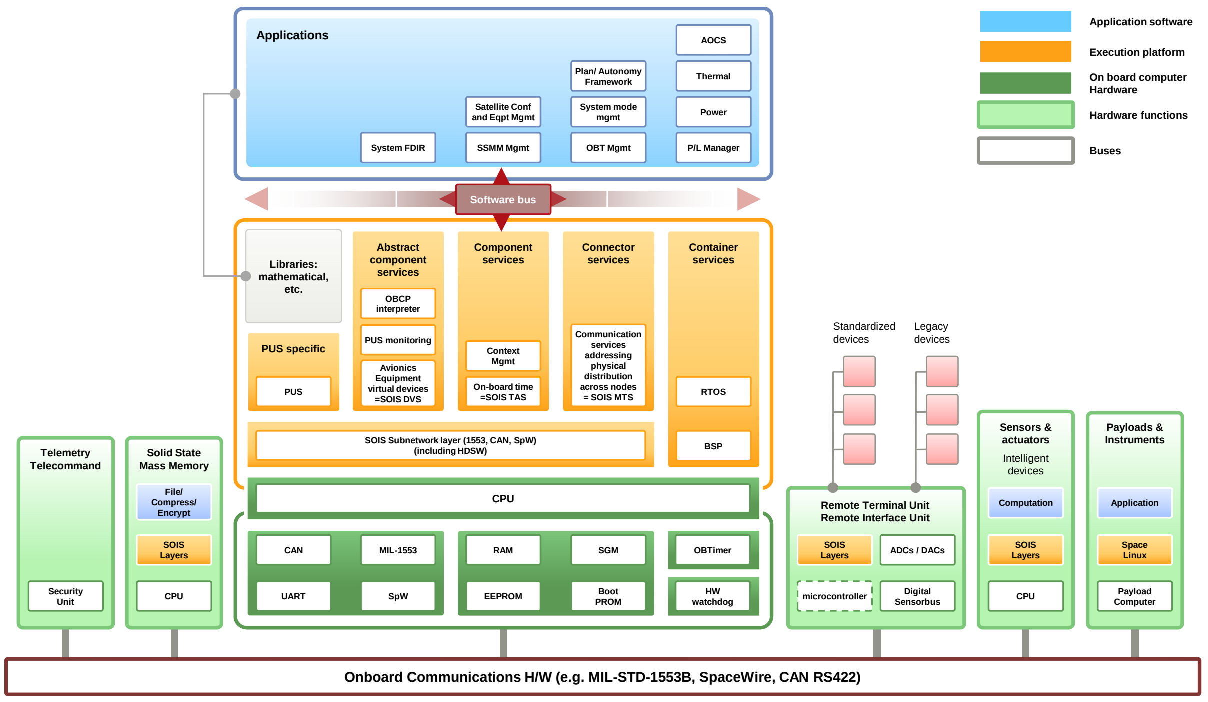

Figure 7-3: SAVOIR Avionics Reference Architecture

The Space AVionics Open Interface Architecture (SAVOIR) group is an initiative, between European space agencies and space industry at prime and supplier level, that studies ongoing initiatives in both agencies and industry towards the vision of an avionics development that maximises reuse and standardisation. SAVOIR Advisory Group (SAG) is established to steer the work plan of the technical discussions. SAVOIR-Fair Architecture and Interface Reference Elaboration (FAIRE) is a specific subgroup of the SAG. It allows more detailed technical discussions on specific architectural topics, provides recommendations to R&D activities, and provides the SAG with a synthesis of their results.

It is commonly acknowledged within SAG and the related subgroups the importance of SOIS communication architecture. The advantage of adopting SOIS subnetwork services among all intelligent equipment, and the full set of SOIS services within the central OBC, comes from a standardisation of the Execution Platform services, allowing management architecture variability factors involving Avionics, Network, and Processor Module.

Regarding OBC Execution Platform, the SOIS services are used as follows:

- MTS: infrastructure for message passing between the entities of the application layer of the local computer or between different processing units (connector services);
- FPSS: basic services to manage solid state mass memories and to store and retrieve software context (component services);
- TAS: basic services to manage the onboard time (component services);
- CDAS: to provide access to the onboard devices (abstract component services);
- Subnetwork Services: low-level data link-independent communication services.

The SAVOIR-Sensors and Actuators Functional Interfaces (SAFI) subgroup foresees using the SOIS concepts related CDAS and DES to standardise the functional interfaces of Attitude and Orbit Control System (AOCS)/Guidance Navigation and Control (GNC) equipment as well as taking advantage from the introduction of the Electronic Data Sheets as part of the software design process.

7.4.3 DECENTRALISED SYSTEM AND SOFTWARE ARCHITECTURE

7.4.3.1 Space Plug-and-play Avionics

The Space Plug-and-play Avionics (SPA) used by Air Force Research Laboratory was developed to minimize the time from specification of a space mission to launch. SPA uses extensible Transducer Electronic Data Sheets (xTEDS) during design and operation of space vehicles to guide the formation of sessions between components. SOIS Electronic Data Sheets are supersets of xTEDS. The SPA network software that handles xTEDS will be extended to accommodate the SOIS EDS. The DoT for SOIS EDS will provide SPA components with metadata that can assure reliable formation of sessions when components are reused in different combinations across multiple missions. SPA hardware designers will use SOIS EDS to generate the device-dependent functions which appear inside the ‘Device’ box in figure 7-4. Figure 7-4 describes an architecture for interoperability between SOIS and SPA, which will enable adaptive use of SPA components in a stable onboard computing environment.

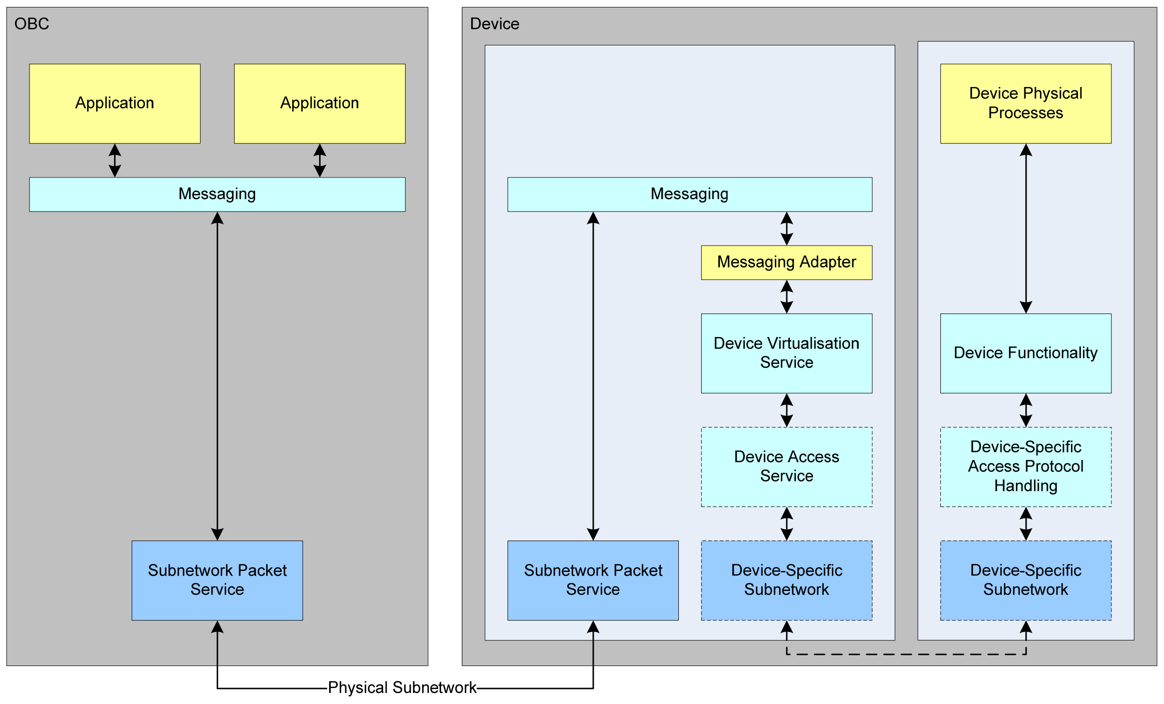
OBC
Application
Application
Messaging
Subnetwork Packet Service
Device
Messaging
Messaging Adapter
Device Virtualisation Service
Device Access Service
Subnetwork Packet Service
Device-Specific Subnetwork
Device Physical Processes
Device Functionality
Device-Specific Access Protocol Handling
Device-Specific Subnetwork
Physical Subnetwork

Figure 7-4: Intelligent Device Deployment for SOIS Plug-and-Play Architecture

7.4.3.2 Core Flight Executive

The core Flight Executive (cFE) used by NASA Goddard Space Flight Center provides a framework for software and software artefact reuse to minimize non-recurring engineering tasks. Similar to SOIS, the cFE uses a layered software architecture and includes several services that parallel those specified in SOIS. Unlike SOIS, the cFE is based on a software component architectural pattern with a software bus providing the primary data exchange mechanism between software components. Hardware devices are connected to software application components via a software driver that connects the hardware to the software bus. The software driver contains a mix of the SOIS DVS and DAP functionality. This approach allows hardware devices to appear to the software applications as just another software component with a high level of virtualization and portability in a distributed system.

Of the defined SOIS services there is potential to align interfaces as circled red in figure 7-5 below. With the exception of the DES and DDS the cFE currently has a fully specified ‘C’ language API. Changes to align the API with SOIS are possible as the cFE evolves outside its original robotic spacecraft domain. The DES and DDS are currently being evaluated for inclusion in the cFE framework.

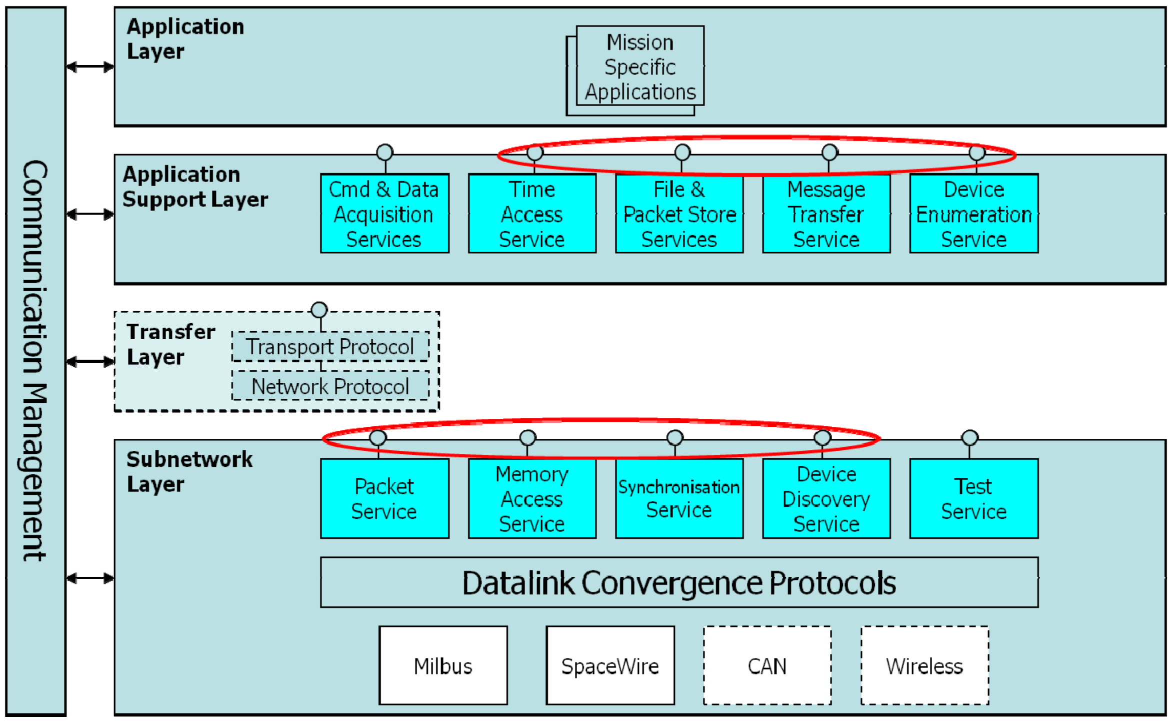

Figure 7-5: cFE Software Framework to Mapping to SOIS

In the area of Electronic Data Sheets, the cFE designers expect to fully utilize the DoT and schemas as developed by CCSDS for flight software development, data exchange definitions, and ground system databases. cFE EDS use would align well with the EDS use cases and the descriptions in the EDS section of this document.

ANNEX A

ACRONYMS AND ABBREVIATIONS

This annex identifies and defines the acronyms and abbreviations used in this Report.

AMS	Asynchronous Message Service
API	application programming interface
BP	Bundle Protocol
CCSDS	Consultative Committee for Space Data Systems
CDAS	Command and Data Acquisition Services
CFDP	CCSDS File Delivery Protocol
DACP	Device Abstraction Control Procedure
DAP	Device-specific Access Protocol
DAS	Device Access Service
DDPS	Device Data Pooling Service
DES	Device Enumeration Service
DoT	dictionary of terms
DVS	Device Virtualisation Service
EDS	electronic data sheet
EGSE	electrical ground support equipment
FAS	File Access Service
FIFO	first in, first out
FMS	File Management Service
FPGA	field programmable gate array
FPSS	File and Packet Store Services
ISO	International Standards Organisation
LAN	local area network

LLC	logical link control
LTP	Licklider Transmission Protocol
MIB	management information base
MTS	Message Transfer Service
NP	Networking Protocol
OBC	onboard computer
OSI	Open Systems Interconnection
PDU	protocol data unit
QoS	quality of service
PSAS	Packet Store Access Service
PSMS	Packet Store Management Service
RMAP	Remote Memory Access Protocol
RTOS	real-time operating system
SAP	service access point
SDU	service data unit
SOIS	Spacecraft Onboard Interface Services
TAS	Time Access Service
TC	telecommand
TCP	Transmission Control Protocol
TM	telemetry
UDP	User Datagram Protocol
XML	eXtensible Markup Language