SOIS Application Support Layer

SpaceWire/SOIS meeting

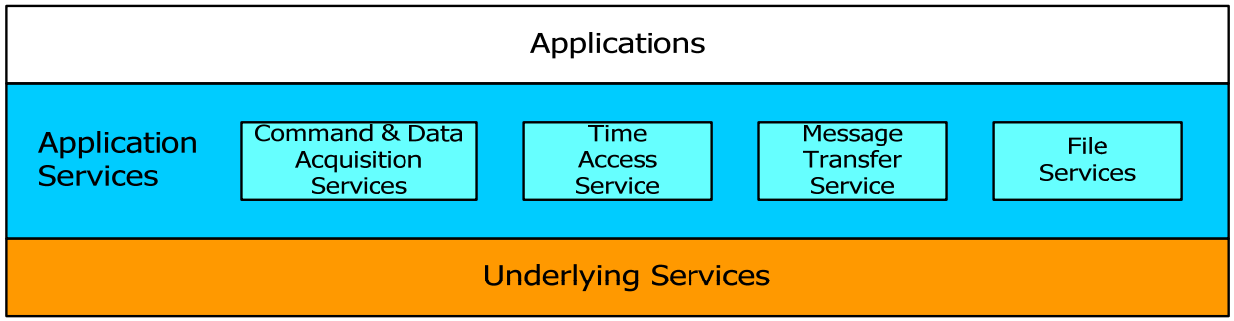
Estec April 07

Stuart Fowell SciSys UK Limited

SOIS Application Support Layer

- SOIS Application Support Layer provides common services required by applications on any processing node of the spacecraft
- They isolate the applications from the underlying topology and communications architecture of the spacecraft
- These services rely upon SOIS Subnetwork Services to provide abstract communication services, that are themselves mapped onto individual onboard buses, mesh networks and LANs

Services of the SOIS App. Support Layer

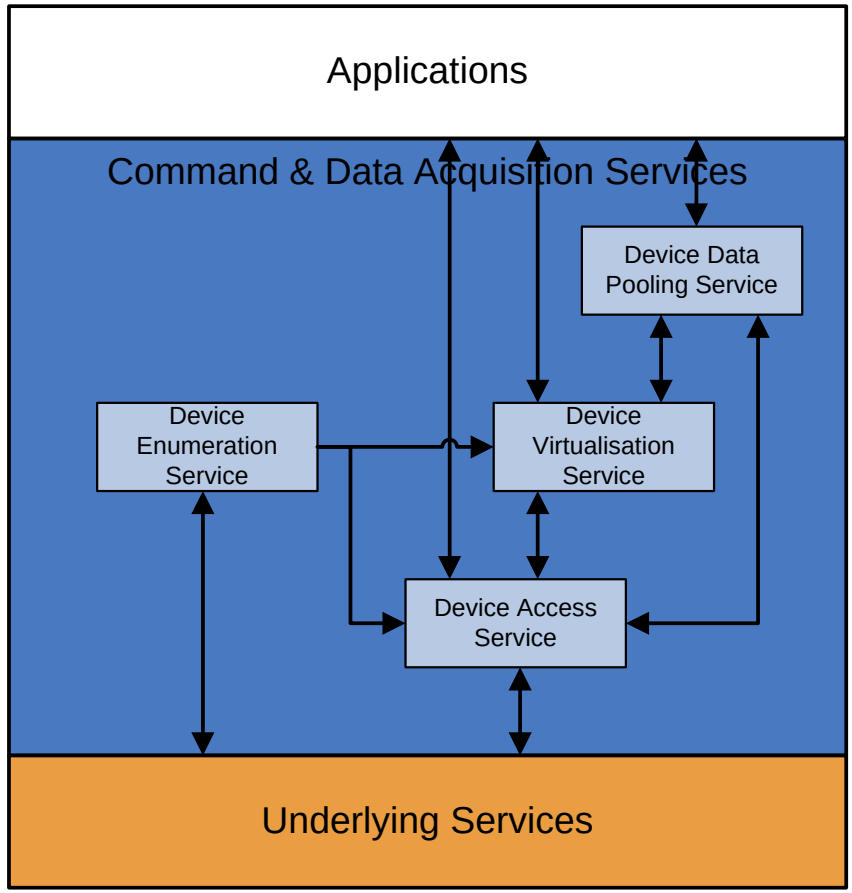


- **Command and Data Acquisition Services** – commanding and data acquisition by applications for devices, i.e. transducers and simple instruments, independent of their locations
- **Message Transfer Service** – enables applications hosted onboard a spacecraft to communication with each other using asynchronous, ad-hoc, discrete messages with a bounded latency, including multicast and broadcast, independent of their locations
- **File Services** – access by applications to, management of, and transfer of files within onboard file stores
- **Time Access Service** – access for applications to the onboard time with bounded accuracy independent of their locations

Command and Data Acquisition Services

- Provides a low overhead access method for spacecraft hardware devices such as sensors and actuators, regardless of location

- Split into a number of capability sets:
- **Device Access Service** – basic reading from and writing to devices regardless of location
- **Device Virtualisation Service** – provides reference to virtual, i.e. generic, image of a physical device
- **Device Data Pooling Service** – maintains an image of the states of a number of devices
- **Device Enumeration Service** – manages plug-and-play of devices



Device Access Service

- Provides standard interface between service users and hardware devices
- Basic device read and write capability

- Service user is isolated from the physical location or the detailed knowledge of the electrical interface
  - Be it accessed via direct IO, analogue, digital, pulsed etc, or across a bus or across a network
- Service user must still know the format of and is responsible for correctly composing commands written to and interpreting data read from the device

- Can be used directly be applications or basis for more capable services, e.g. engineering unit conversions on raw data or monitoring services

- Open Issues:
  - Service is based on request/response. In DisCo project, asynchronous generation of data from devices is also addressed. Does the standard cover this?
  - No Device Access Protocol has been specified where a proxy Device Access Service implementation must be used to access a device (because of no direct subnetwork access)

Device Virtualisation Service

- Provides standard interface to virtual, i.e. generic, image of a physical device
- Service user interacts with virtual image of the physical device and service handles translation of commands to the virtual image into commands to the physical device, and vice versa for data

- Allows for application to be implemented to interact with “standard” devices, with the service providing the translation into particular devices
- Replacement of a particular device type only requires modification to the service and not the application

- Class hierarchy of devices
  - Starting point for class hierarchy is the ETSI/ECSS SSDHI Standard

- Open Issues:
  - Standardisation is still at an early stage

Device Data Pooling Service

- Maintains an image of the states of a number of devices
- Service user can access the state of a device in the pool without having to generate explicit data acquisition from the actual device
- Service periodically samples the devices at predetermined sampling rate or caches state from devices that asynchronously generate data
- Provides guarantees on maximum age of each parameter in pool and that software image is accurate

- Open Issues:
  - None

Device Enumeration Service

- Responsible for management of plug-and-play devices
- Handling of “discovered” devices:
  - Determine capabilities, e.g. device data sheet
  - Initial configuration of device
  - Management of other Command and Data Acquisition Services’ MIBs
    - e.g. update Device Access and Virtualisation Services’ MIBs to allocate Device Identifier and configure device class and access mechanism (underlying service, address and QoS)

- Open Issues:
  - Standardisation is still at an early
    - See Plug-and-Play slides

Message Transfer Service

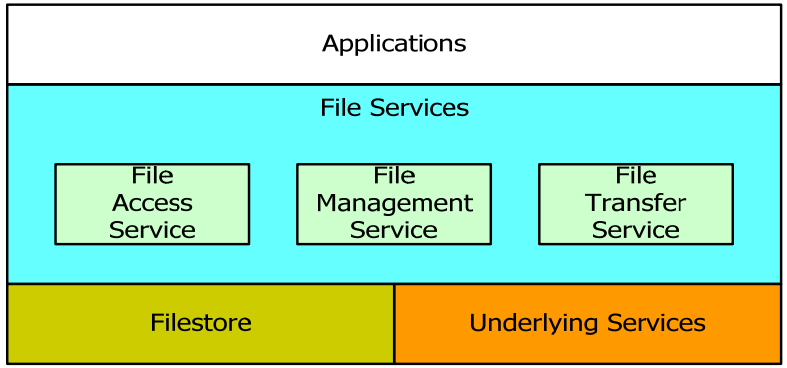
- Enables applications to communicate with each other using asynchronous, ad-hoc, discrete messaging with a bounded latency, including multicast and broadcast
  - Provides basis for variety of communication frameworks, e.g. distributed PUS, Real-Time/Embedded CORBA

- Open Issues
  - Likely to be a Magenta Book, documenting best practise on how to implement the SIS Asynchronous Messaging Service (AMS) onboard in the SOIS architecture

File Services

- Used to access, manage and transfer files
- Files could contain any type of data, e.g. telemetry, commands and command sequences, software updates, imagery and other science observations

File Services



- Basic concept is files reside in a **File Store**, that consists of:
  - **Mass Memories** in which files reside
  - Associated **File System** providing functionality for managing the files

- Split into a number of capability sets:
- **File Access Service** – allows access to files and portions of their contents
- **File Management Service** – allows manipulating existing files on local or remote file stores (onboard the same spacecraft)
- **File Transfer Service** – allows a service user to transfer files between file stores (onboard the same spacecraft)

File Access Service

- Provides access to files and portions of their contents, in a file store regardless of its location
- Primitives include
  - Open, Close file
  - Read, Write, Append, Insert, Remove
  - Find first, find last, find next

- Open Issues:
  - No File Access Protocol is specified. Should it be?

File Management Service

- Provides manipulation of existing files in a file store regardless of location
- Primitives include:
  - Make, destroy, rename, lock, unlock directory
  - Create file in, delete file from, delete all types of files from directory
  - Copy file, concatenate file
  - Rename, lock, unlock file
  - Find file
  - List directory contents
  - Operate specific algorithm on file

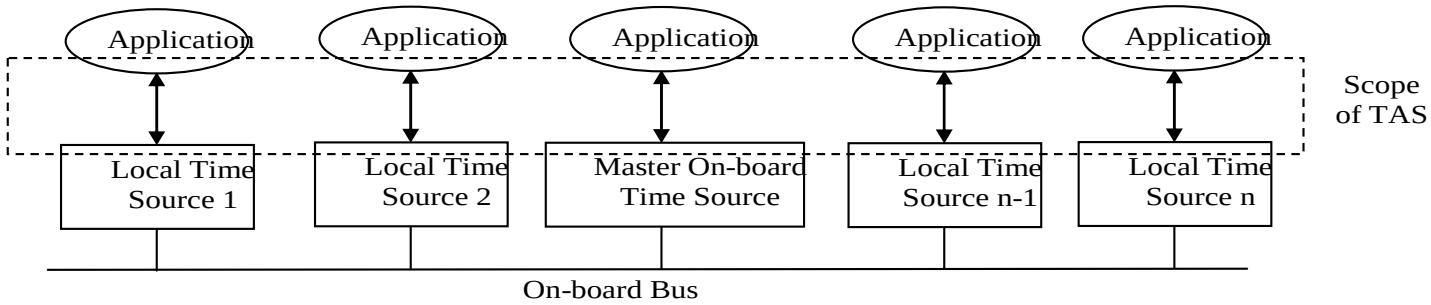
- Open Issues:
  - No File Management Protocol is specified. Should it be?

File Transfer Service

- Allows transfer of files between file stores on the same spacecraft
- Primitives include:
  - Initiate transmission of files between file stores
  - Write produced data into file in local or remote file store
  - Initiate file store operations on local and remote file stores
  - Receive events related to operation of current transactions
  - Request status information related to current transactions
  - Send or receive messages associated with current transactions
  - Suspend, resume, or cancel transmission of current transaction
- Does not include transfer of files between file stores residing on different nearby spacecraft or between spacecraft and ground – this is provided by CFDP

- Open Issues:
  - No File Transfer Protocol is specified, though CFDP is suggested. Is this appropriate?

Time Access Service



- Provides users with a consistent interface to a local time source that is correlated to some centrally maintained master on-board time source
- **Wall-clock** – ability to read the time on demand
- **Alarm-clock** – request notification at a particular time
- **Metronome** – periodic notifications with a specified interface and starting at a particular time
- Not concerned with mechanism used to correlate time between time sources as diverse implementation on different spacecraft

- Open Issues:
  - None

App. Support Layer - Services Required from the Underlying Layers

- Required Underlying Services:
  - Packet Service
  - Memory/Register Read/Write Service
  - Time Correlation/Distribution Service
  - Device Discovery Service

- Generic requirements for each service:
  - Unique identification of each node's access point to access method, i.e. address
  - Subnetwork QoS presented to users

- For asynchronous sub-networks:
  - Segmentation and Prioritised delivery of PDUs

SOIS Application Level Addressing

- Application Layer Services provide abstraction from underlying communications
  - Entities addressed using Identifiers
    - E.g. Applications, devices, files etc.
  - Allows reconfigurations without affecting applications
    - E.g. during integration subnetwork addressing changes, direct I/O changes
    - E.g. during fault handling, swap over of underlying communications bus, replacement of devices, restart of applications on different PM

- Identifiers map onto (amongst other things):
  - Subnetwork service to be used to access them
  - Associated parameters:
    - Subnetwork-specific Address
    - Subnetwork service-specific QoS (class and priority?)

Plug-and-Play within SOIS

- SOIS for static configuration of Spacecraft well defined
  - Of course, still being mapped onto specific busses, etc.
- Next stage is dynamic configuration – “Plug-and-play”

- Three stages:
  - Identify requirements
    - Driven by use cases
  - Review equivalent standards and implementations
    - Comparisons of USB 2.0, IEEE 1451, 1-wire
    - SpaceWire P-n-P prototyping
  - Define generic architecture
    - Device Discovery, Enumeration and management of Device Virtualisation & Access Services

Plug-and-Play – Use Cases

- Initial suggestions:
  - Spacecraft Integration & Test
    - EGSE connection using wireless technologies
  - Activation of redundant devices in response to faults
    - FDIR simply powers up replacement
    - Reconfiguration happens automatically (bottom-up), rather than hierarchical (top-down)
  - Rapid Spacecraft assembly
    - Reduce/eliminate need for aspects of Spacecraft database?
  - Manned space
    - True ***plug***-and-play?
  - Any others?

- Status is ongoing
  - Will be used to extract generic architecture

Plug-and-Play – Review equivalents

- Candidates
  - USB 2.0
  - IEEE 1451
  - 1-wire
  - SpaceWire P-n-P prototyping
  - Any others?

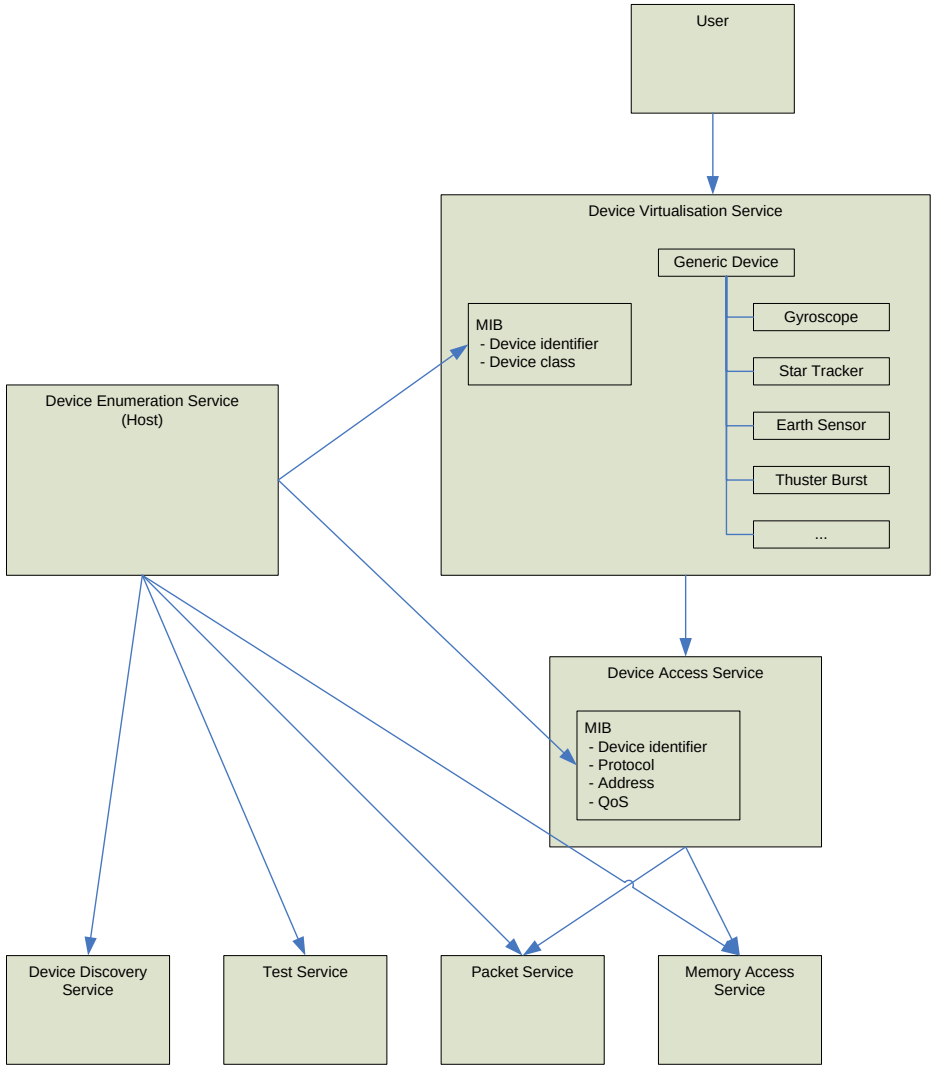
- Status is ongoing
  - Will produce comparison of capabilities
  - Compare against requirements identified from use cases

Plug-and-Play – Generic Architecture

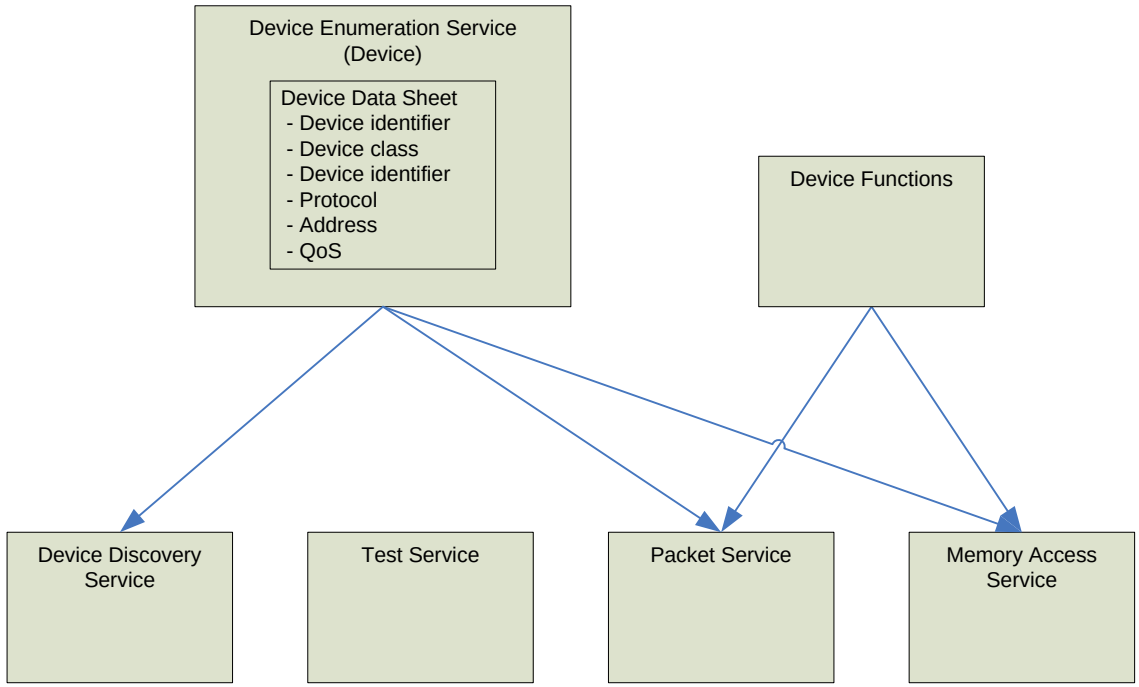
- Heart of the mechanism is the Device Enumeration Service
  - uses Device Discovery Service to detect new devices
  - uses Test Service to ensure operational
  - uses Memory Access or Packet Service to read Device Data Sheet from device
  - updates Device Virtualisation and Device Access Service's MIBs with information read from Device Data Sheet
- Device Virtualisation Service
  - provides class hierarchy of devices
  - class-specific interfaces for collections of devices
- Device Access Service
  - provides abstraction from access mechanism to device
- Device Discovery Service
  - provides mechanism to detect new devices

- This approach allows for both static or dynamic systems to be deployed :
  - dynamic system, by which the Device Enumeration Service detects devices, which describe themselves, and the MIBs are populated on this basis
  - static system, i.e. miss out Device Enumeration and Device Discovery services, with MIBs statically populated or directly manipulated by system applications, e.g. FDIR.

Plug-and-Play – Generic Architecture



Plug-and-Play – Generic Architecture



Plug-and-Play – Generic Architecture