A Satellite Data Model for the AFRL Responsive Space Initiative

Kenneth Sundberg, Scott Cannon, Todd Hospodarsky
Utah State University
Logan, UT

Don Fronterhouse, Jim Lyke
Air Force Research Laboratory
Albuquerque, NM

Abstract

The Air Force Research Laboratory's Responsive Space Testbed initiative is focused on the rapid deployment of tactical satellites. To support this goal we have developed the Satellite Data Model (SDM).

Critical to the success of the Satellite Data Model are the autoconfiguration abilities of the SDM. The SDM system allows devices and applications can be developed independently of each other. When the need of a satellite is conceived, components appropriate to that need are selected and added into the system. The SDM then manages the necessary initialization and discovery so that the components can work correctly in concert to form a functioning satellite.

1 Introduction

The Satellite Data Model (SDM) project is supported by the Air Force Research Laboratory as a critical component to the Responsive Space Testbed initiative. This initiative is intended to support the 6 day satellite – implementing conception to launch of a small tactical satellite within 6 days.

Within this very short window of 6 days, there will be insufficient time to develop even a single line of software code. The SDM was designed as a middleware layer to support auto-configuration of pre-written applications and provide a completely generalized interface between applications and devices or sensors.

The 6-day satellite is envisioned to consist of both a high-speed data network and a low-power network. To standardize this network, three communications system standards have been developed: SPA-U, SPA-E, and SPA-S. SPA-U is based on an enhanced USB protocol. SPA-S is a high-speed SpaceWire variation. SPA-E is high-speed Ethernet.

The SDM is based upon a variation of the publish/subscribe distributed systems model. Utilizing the SPA-U communications link (an enhancement of USB), devices are auto-detected by the SDM system when connected or powered up. Each device is supported by a small micro-power controller capable of SPA-U communications and messaging with the SDM.

Upon detection, each device is queried by the SDM for a complete description of its data producing capabilities, controls, and data message formats. To support this very complex and complete description, an XML description language was developed called xTEDS as an extension of the IEEE 1451 TEDS specification.

In addition, any applications that are data producers initiated also register their xTEDS descriptions with the SDM. As such, the SDM maintains a registration of all device and application capabilities available within a system. When a device is powered-down or an application is suspended or terminated, the SDM automatically inactivates the associated xTEDS.

When an application needs data, it simply queries the SDM for registered producers of related data products, chooses the supplier most appropriate to its needs, and then subscribes to receive that data. To support such a general query, both a query language and a common or standardized data definition or dictionary was implemented.

A subscription to data products is passed by the SDM to the device controller, which can then initiate data messaging to the consumer. A device can have a direct peer-to-peer interface to a high-speed network, utilize a local SDM processor as high-speed network proxy, or use the SPA-U network for messaging.

As such, the SDM allows applications to be developed and tested prior to specific knowledge of network configuration, data device specifics or interfaces, or interfaces to its own data consumers. By providing a general mechanism to allow devices and applications to self-configure, the time between mission conception and launch can be significantly reduced. There is currently a wide consortium of device and application developers working on SDM compatible products.

1.1 An Example

As an example suppose there is to be a satellite launch in six days using the SDM system. Along with the necessary flight systems we would like to include a few experiments. One of these experiments must be kept at a constant temperature.

Along with the design of our experiment we have written an SDM application, this application acts as a thermostat. It monitors the temperature of the experiment, and uses a heating element to adjust the temperature to keep it within the required range.

We also have a selection of temperature gauges and heating elements from a variety of vendors. We select a gauge based on range, accuracy and economics. Likewise we select a heating element considering energy output, accuracy, responsiveness, and economics. One very important consideration is that we are free to select a different device as our needs and circumstances change. This is similar to Cotterell's system of interchangeable hardware eBlocks [1], though larger in scope and complexity.

We install our application and connect our gauge and heating element to the SDM system. The gauge and heating element are then placed in a physically appropriate place. The devices then participate in a self discovery and configuration process, becoming available to the system. Part of this process includes registering a description of the devices capabilities known as the xTEDS document. This placement information is also recorded so that it can be used by the flight software.

When our application starts it knows that it needs temperature data and a means to manipulate that value. The application requests a data source, and a temperature actuator. The SDM responds with the xTEDS descriptions of the temperature gauges and heating elements along with their network locations.

Our application then requests a floating point representation of temperature in degrees Celsius, the SDM converts this request into the format required by the device and in turn converts the data produced by the device into this requested format. The application never needs to know the specific device description, only its relevant capabilities.

When our application detects that the temperature has fallen too far it requests that the temperature actuator increases its energy output. Again this request is made in a generic way and interpreted by the SDM into the specific format required by the device. The request is routed to the appropriate heating element and the results are seen by the application in the temperature data.

2 System Overview

The SDM system consists of both core components that are present on every system, and interchangeable components such as applications and devices that are selected on a mission by mission basis. figure 1 contains a graphical depiction of the SDM system.

2.1 SDM Core Components

The core SDM components provide the environment for SDM applications and devices. From the perspective of applications and devices these components form a single unit, the SDM system.

2.1.1 Data Manager

The Data Manager acts as a resource broker. It is the repository of the descriptions of data sources available in the system. The Data Manager is also responsible for keeping track of the network locations of the available resources. When data consumers query the Data Manager for data sources with specific attributes, the Data Manager responds with a list of appropriate data sources.

The Data Manager is central to the initialization and discovery. The Data Manager exists at a known network location, this allows applications and devices to register their data requirements and capabilities. The capabilities of applications and devices are registered as xTEDS documents that are added to the Data Managers xTEDS library. The Data Manager then matches capabilities

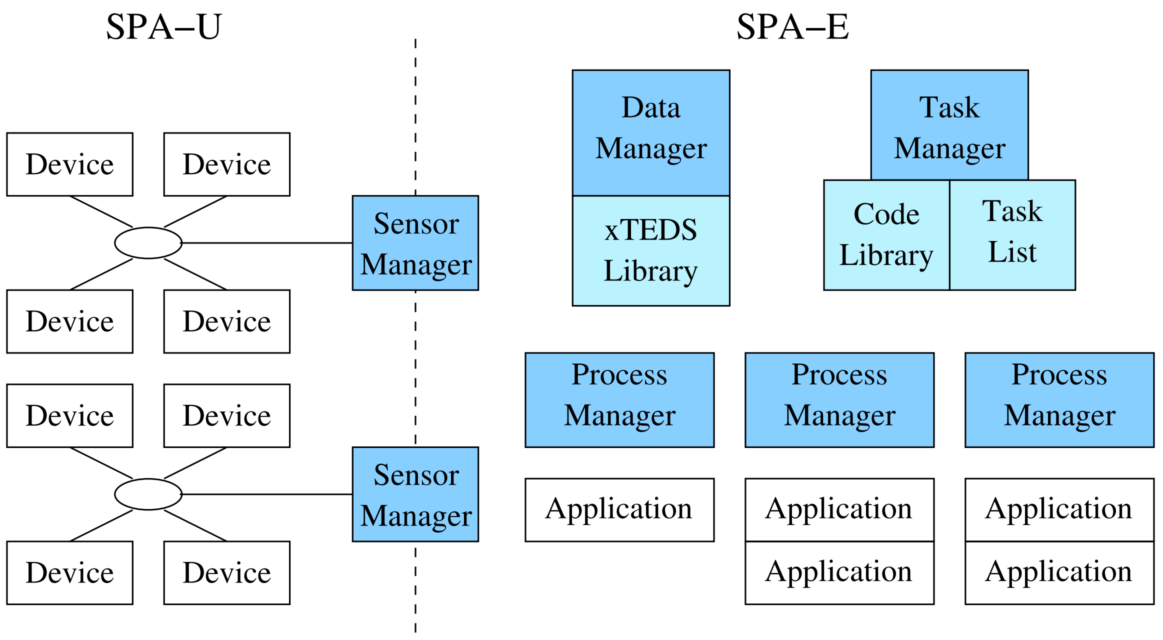

Figure 1: SDM system overview with core components highlighted.

to requirements and communicates the match and involved network locations to the participants. The Data Manager is however only used during initialization and discovery. Once a data consumer has selected an appropriate data source from the list provided by the Data Manager all communications occur peer to peer.

2.1.2 Task Manager

The Task Manager has two responsibilities. First, the Task Manager maintains a library of applications that can be run by the SDM system. Second, the Task Manager coordinates the efforts of the Process Managers to insure that tasks are run. This coordination includes load balancing and fault-tolerance.

2.1.3 Process Manager

A Process Manager is responsible for a single processing node on the SDM network. It registers the nodes capabilities with the Task Manager. The Task Manager responds with an application that should be run on the processing node, along with the code if necessary. The Process Manager executes the task. It monitors all of its executing tasks so that the system can be informed of failures.

2.1.4 Sensor Manager

A Sensor Manager is responsible for a SPA-U network. The SPA-U network consists of a modified USB bus and any devices that will be attached to the network. The Sensor Manager monitors when devices are added to and removed from its network. It also acts as a proxy between the SPA-U and SPA-E networks and protocols.

2.2 SDM Applications

There are two subsets of SDM applications based on whether they are data consumers or data producers. These two sets are not mutually exclusive, it is very possible to have an application that consumes one data set, operates on it, and produces a second set.

2.2.1 Data and Service Providers

Data or service providing applications must register their capabilities with the Data Manager. After registering with the Data Manager, service and data requests will be sent to the application. These requests contain all of the network information to send the appropriate responses back to the subscriber on a peer to peer basis. Provider applications are responsible for managing all of the subscription information required for this task.

2.2.2 Data and Service Consumers

Data consumers request data by querying the Data Manager for a data source with attributes matching criteria that the application specifies. The Data Manager responds with a list of matches, from which the application chooses. The application informs the Data Manager of its selection. This request is then forwarded on to the data provider after the Data Manager has append the network location of the consumer. The provider then starts the requested peer to peer communications.

2.3 SDM Devices

SDM Devices can be both sensors and actuators. They have an xTEDS document that describes there capabilities and messages.

2.3.1 Lightweight Devices

Lightweight devices are envisioned to be the most common type of SDM device. A lightweight device only implements the SPA-U interface and relies on a Sensor Manager to handle interactions with the rest of the SDM system.

2.3.2 Heavyweight Devices

Heavyweight devices have their own dedicated connection to the SPA-E network. They may either not use or only partially utilize the Sensor Manager.

2.3.3 Legacy Devices

Since the SPA-U interface is simple, it is very feasible to write a simple microcontroller to make legacy devices compatible with the SDM system.

3 Auto-configuration

At the heart of the Responsive Satellite is a philosophy of self configuration. In order to meet the tight time constraints for design of a satellite it is critical that the components are designed in such a way that they are interoperable and in many cases interchangeable. This extends to both hardware and software within the SDM system.

3.1 Initialization

Each component of the Satellite will be turned on in an indeterminate order. The SDM imposes a partial ordering on the initialization. This partial ordering is shown in figure 2. The initialization process is an ongoing one. In the interests of power management various nodes and devices may be turned off and on at different intervals through a flight. As they rejoin the SDM network the initialization of that component reoccurs.

3.1.1 Core Components

The core SDM components initialize before any of the other components. First each core component performs any start-up routines that are independent of other components. The Data Manager has no start-up routines that depend on any other components and so after this first step the Data Manager is fully initialized.

The Sensor Manager and Task Manager both require that the Data Manager be initialized prior to there initialization being complete as both may post xTEDS. The Sensor Manager and Task Manager both periodically send ready messages to the Data Manager. When the Data Manager is fully initialized it responds to these ready messages with another ready message. Having thus confirmed that the Data Manager has completed initialization both the Task and Sensor Managers proceed and complete their initialization.

After the Sensor Manager has initialized it begins monitoring the SPA-U network. When a new device is detected it requests that the device initialize and send its xTEDS which the Sensor Manager then forwards to the Data Manager.

The Process Managers require that the Task Manager be fully initialized to function properly. To insure this, like the Task and Sensor Managers, each Process Manager periodically sends a ready message to the Task Manager. When the Task Manager has been initialized it responds to the message.

3.1.2 Devices

SDM devices connect to the system on a modified USB bus. Each USB bus is managed by a Sensor Manager. When a device is connected the Sensor Manager is notified. The Sensor Manager then requests that the device run its initialization routine and then transmit it's xTEDS. The Sensor Manager also notes the network

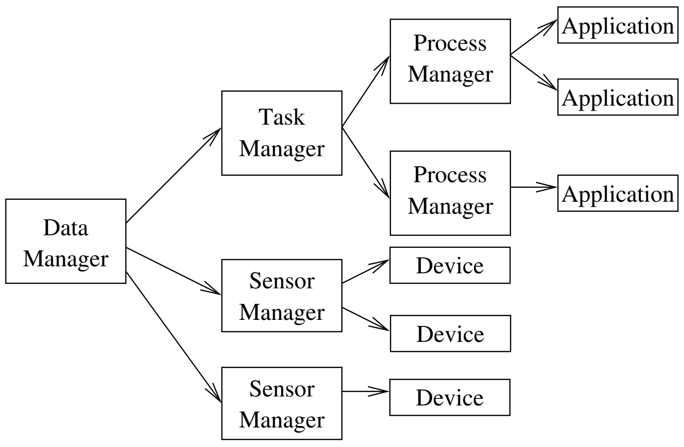

Figure 2: SDM initialization partial ordering.

location of the device which it passes along with the xTEDS to the Data Manager. The Data Manager looks up a physical location based on the SPA-U network location of a device and appends location qualifiers to the xTEDS of the device. This description of the device's capabilities is then registered with the Data Manager.

3.1.3 Applications

Applications are initialized in three ways in the SDM system. First the configuration file includes a list of applications to be started during initialization. The Task Manager will start these applications as soon as an appropriate Process Manager registers. Second running applications can post new tasks, again the Task Manager will start these new applications on an available Process Manager. Finally, if the application is a data provider the Data Manager will post the application to the Task Manager when appropriate.

3.2 Discovery

Discovery in the SDM system is attribute based. The meaning and names of these attributes is defined in a Common Data Dictionary. An application looking for particular capabilities and attributes makes a query of the Data Manager. The Data Manager looks through the xTEDS library to find any applicable matches. When a match is found a brief synopsis of the match called a message definition is built from the relevant portions of the matching xTEDS. This message definition is then passed back to the requesting application along with the logical address of the application or device with the matching xTEDS. In the case of multiple matches, one message definition is sent back for each match. Then the application selects the source to use and begins peer-to-peer communication with that source. This process is shown in figure 3.

3.3 Failure Recovery

When a device fails, its presence is no longer detected by the Sensor Manager monitoring the SPA-U network that the device was associated with. When a failure is detected the Sensor Manager cancels the xTEDS of the device and recovers any resources that the device was utilizing.

Likewise if an application experiences an abnormal termination condition, such as a segmentation fault, the Process Manager detects this. Upon detection the Process Manager cancels the xTEDS of the application and recovers any resources used.

The failure of a Sensor or Process Manager is detected by a timeout. All of the core components periodically send out heartbeat messages, when a heartbeat fails to arrive the component is assumed to have failed. When this failure is detected all of the xTEDS that were associated with the Sensor or Process Manager are canceled.

A canceled xTEDS no longer matches any requests and so will not be found during the discovery phase.

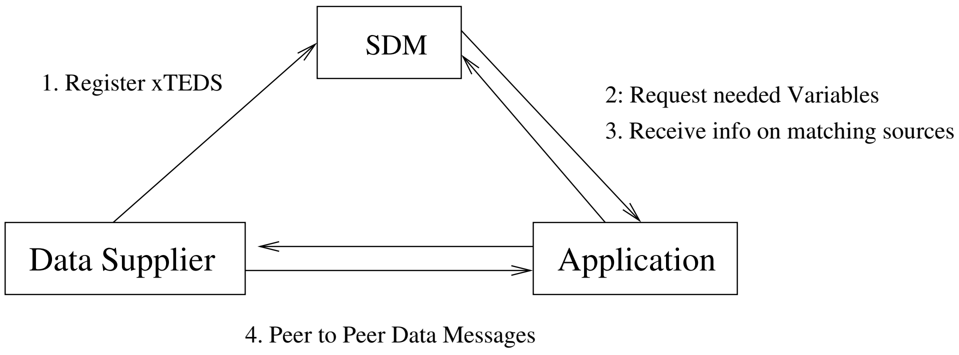
SDM
1. Register xTEDS
2: Request needed Variables
3. Receive info on matching sources
Data Supplier
Application
4. Peer to Peer Data Messages

Figure 3: SDM discovery and operation.

Further applications can ask to be informed of any cancellations. An application on being informed of the failure of one of its data or service providers can begin looking for alternatives using the normal discovery process.

3.4 Control

Other than the initialization, discovery, and failure recovery phases of the SDM system all communication is peer to peer. This means that tight control loops can be formed, this is critical for many systems most notably the flight systems of the satellite.

4 Data Description

4.1 xTEDS

xTEDS are an extension of the IEEE 1451 TEDS [3, 4]. They provide a description of the capabilities of a data or service provider.

4.1.1 Device Description

The first section of an xTEDS document describes the device. This description includes such things as a name, id number, and manufacturer. It can also include a ASCII text description of the device.

4.1.2 Variable Descriptions

In the variable section each data variable is defined and described. The variable description includes format, array length for vector variables, and a name. Often included is a data kind as defined in the Common Data Dictionary, such as temperature, units, precision among other qualities. As with devices an ASCII text description of the variable may be included.

4.1.3 Message Descriptions

Data messages descriptions define the output of a device. Each output message is listed in the xTEDS along with references to every variable in the data message.

4.1.4 Command Descriptions

Commands are a description of the messages used to control and manipulate a device. Many commands may result in data being produced, in which case they will also be part of the description of a service in the xTEDS. Devices will often include commands to set data rates and change operational modes.

4.1.5 Service Descriptions

Service descriptions bind a command message to a data message. Services act similar to subroutines in that they expect an input message, the command, and generate one or more output messages, the data message.

4.2 Common Data Dictionary

The Common Data Dictionary is intended as an ontology for SDM systems. It is a developing work that defines the different kinds of information and formats that can be available. Both devices and applications are to be designed in reference to the Common Data Dictionary, so that the meaning of these fields remains universal.

4.3 Physical Location

When a device registers its xTEDS through the Sensor Manager, a section describing the physical location of the device is append to the registered xTEDS. This allows data consumers to use location attributes to find data and service providers.

5 Message Protocols

The SDM system consists of two types of networks and two protocols. The different SDM protocols are variants on the publish/subscribe model [2]. SPA-E forms the main network that all components participate on in some fashion. SPA-U forms smaller device networks that participate in SPA-E via a proxy.

5.1 SPA-U

The SPA-U protocol joins the Sensor Manager and SDM devices. The protocol is simple and easily implemented, this allows for legacy devices to be integrated into an SDM system with minimal difficulty. This protocol currently is used on a modified USB 1.1 network.

5.2 SPA-E

SPA-E is the main messaging protocol for the SDM system. In the current prototypes it is run over an ethernet network, but it is anticipated that spacewire may be used in the future.

6 SDM API

There are many common operations among SDM applications and the core SDM components. In an effort to simplify the production of SDM code a library of classes has been built to handle common SDM tasks. This library continues to expand with further development of the SDM system.

6.1 Message Classes

The message classes serve as a layer of abstraction between SDM applications and the SPA-E protocol. Each class corresponds to one of the SPA-E messages, that way as changes have been made and will continue to be made to the SPA-E protocol or in the future the transport network, no changes will be required in previously written SDM applications.

6.2 Message Manipulator

The Message Manipulator class provides generic marshalling and unmarshalling capabilities. The class is guided by a message definition, a summary of the portions of a xTEDS relevant to one particular message. An application can then set and get values within a data or command message by using the variable name, the Message Manipulator determines the offset and performs the required marshalling and unmarshalling steps.

6.3 Subscription Manager

The Subscription Manager was written to speed development of applications that are intended as data providers. In the SDM system a data provider is expected to keep track of all subscription information and to multicast data to all interested providers. The Subscription Manager handles these details.

6.4 Message Manager

SPA-E was originally envisioned to use UDP as a transport mechanism. The Message Manager was written to easily handle the receipt and queuing of asynchronous messages.

7 Conclusion

The Satellite Data Model is a developing standard for rapid integration of hardware and software components in small satellites. Its attribute based discovery and autoconfiguration allows devices and applications to be fully developed and tested independently of each other and out of the context of a satellite mission. In addition there is a developing library of shared code to help speed the development of SDM applications. This leads to the ability to more easily develop SDM applications and devices, and with the growing body of such applications and devices to rapidly deploy a satellite.

References

[1] Susan Cotterell, Frank Vahid, Walid Najjar, and Harry Hsieh. First results with eblocks: Embedded systems building blocks. *CODES+ISSS*, October 2003.

[2] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, June 2003.

[3] Kang Lee. A synopsis of the IEEE P1451 - Standards for Smart Transducer Communication. National Institue of Standards and Technology, 1999.

[4] Kang Lee and Richard D. Schneeman. Distributed measurement and control based on the IEEE 1451 Smart Transducer Interface Standards. *IEEE Transactions on Instrumentation and Measurement*, 49(3):621–627, June 2000.