Design of a Distributed Ground Support System for Small Satellites

R.M. Barry
Electronic Systems Laboratory, Department of Electric and Electronic Engineering, University of Stellenbosch, South Africa, 7600, email: richardbarry@bigfoot.com, Tel +27-21-8084329

P.J. Bakkes
Department of Electric and Electronic Engineering, University of Stellenbosch, South Africa, 7600, email: bakkes@ing.sun.ac.za, Tel +27-21-808-4335

Abstract. On February 23, 1999 South Africa's first satellite SUNSAT was launched. The satellite was designed and built entirely at the University of Stellenbosch, a project spanning 8 years. Up to 1999 the bulk of the development work was done on the satellite itself, with time constraints limiting the work done on the ground support system. This led to a situation where the ground station was developed in the two months prior to the launch and shortly afterwards. The system was developed with the specific purpose of supporting only SUNSAT. The final product was a ground support system consisting of a single OSCAR class ground station located at the University of Stellenbosch. The system was very effective in supporting SUNSAT, but the implementation made expansion and reuse of the system very difficult. The station requires a high degree of human operation and automating the system would require big changes in the software used. This article looks at the design of a new generation of ground support system, for use at the University of Stellenbosch's satellite program on future missions. Distributed architectures are used to enable components of the system to be physically located where resources are best utilised. The design also allows for increasing levels of automation to be introduced.

Introduction

The SUNSAT small satellite program at the University of Stellenbosch reached a climax on February 23rd, 1999 when SUNSAT was put in orbit by a Delta II rocket. SUNSAT was mainly an academic project and the culmination of research done over many postgraduate projects. The great majority of the work done on the project was on the development of the satellite, and in comparison less work was done on the ground support eventually needed to support the satellite. The eventual ground support system was developed in the two months leading up to the launch and evolved thereafter. Because of the time constraint on the development of the ground station software, the most reliable option in development was to reuse as much as possible of the flight software. The result was ground station software developed to support the SUNSAT mission reliably, but with little room for reuse and expansion in terms of multiple ground stations. Automation was limited and operator interaction was required for communication with the satellite.

The need for development of new software to support future small satellite missions was clear. The new software had to be based on an architecture that would allow future expansion as needed. A distributed system would not only allow expansion

and automation, but also the use of multiple ground stations in the future. In the following sections the current ground station software will first be reviewed. A new software architecture based on distributed services is then proposed and finally recommendations are made on further improvements and future work.

Review of the SUNSAT Ground Station Software

The SUNSAT ground station consists of the following components (see figure 1):

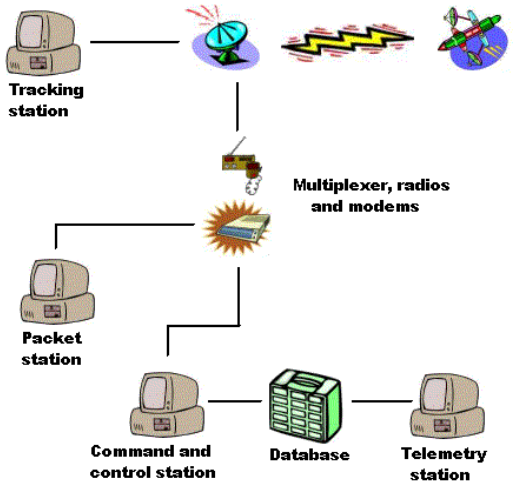

Figure 1: Ground Station Layout

Tracking Station

Interfaces with the antennae controller hardware to track the satellite as it passes overhead. Orbit prediction software is used to calculate the pass times.

Multiplexer, Radios and Modems

Amateur radios are used to communicate with the satellite. A multiplexer is used to route the analogue data between different combinations of modems and radios.

Command and Control Station

Commands the satellite and centralises the control of the other components of the ground station. The control of the station requires a high degree of human intervention. The software was mostly developed in Modula-2 and runs on the MS-DOS platform.

Database

Stores the data acquired from the communication with the satellite. A wide range of data, mostly telemetry data but also status and error information as well as scientific data, is stored in the relational database.

Telemetry Station

Allows for the extraction and processing of the data from the repository as well as exporting the data to third party software.

Packet Station

Provides an interface to the ground segment of the PACSAT protocol suite. Facilities provided by PACSAT satellites, such as the Packet Bulletin Board System (PBBS) and file uploading, can be accessed from here.

Data Flow

The flow of the data from the satellite through the ground station can be seen in figure 2.

The data arrives via the AX.25 radio link at the command and control station where it is handled by the layered protocol stack and stored on the local file system in the form of Whole Orbit Data (WOD) files by

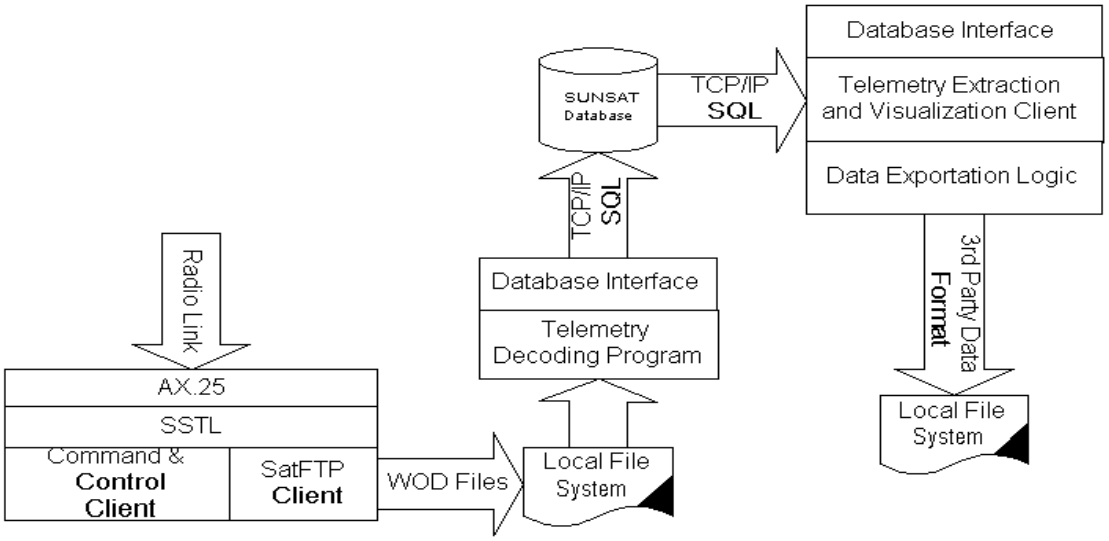

Figure 2: Ground Station Data Flow

the SatFTP Client. The WOD files are then input into the telemetry decoding program and stored in the SUNSAT database via TCP/IP network connection to the relational database. The data can then be extracted using the telemetry extraction and visualisation client and stored in a form readable to third party analysis tools.

In the analysis of the system to determine where changes are to be made to obtain the goals of distribution and automation the following areas where identified:

Command and Control Client

The fact that the client was developed in Modula-2 severely limits the possibility of automation as well as distributing the implemented facilities as separate services, mainly because of a lack of network support and multitasking in the operating system. A solution would be to redevelop the needed functions as individual services distributed on the network. This would make reuse and automation easier to implement. The use of a more modern programming language, like Java, would make this redevelopment less time consuming.

Telemetry Decoding Program

The telemetry decoding connects to the database through a network connection and can easily be included in an automated system. It could be changed to accept input from a client over the network and therefore be deployed as a service on the network.

In the next section a new model will be proposed for the ground support system.

Unified Information Model

The main aim of small satellite missions is the collection of data from its various payloads and systems. The ground support system therefore has as its goals to efficiently and reliably receive, store and present the data from the satellite, as well as efficient control of the satellite. The idea of a unified information model is developed in [1]. The unified information model proposes a software architecture with portable software and portable data.

The use of Java as development language will enable faster development and make the software portable for use on any platform. Using the Extensible Markup Language (XML) as data description language makes the data fully portable and separates the data from the data handling. Standard data handling tools can therefore be developed and reused for different data sets across different missions. A services based architecture enables expansion and reuse of services independently from the clients that use them. The services based architecture has the following benefits:

- The creation of a standard link between network systems the expansion of services and the addition of future systems.
- Increased flexibility and redundancy which leads to greater reliability and scalability.
- The physical distribution of services to where resources are best utilised.
- Automation can be introduced in increasing levels by adding services to facilitate automation.

XML

XML [2] is a meta-language used to define other languages, like it's predecessor Standard Generalised Markup Language (SGML). Unlike HTML or most other markup languages, XML does not specify the tag set or the grammar for a language. The tag set is the set of tags that have a meaning in the specific language, while the grammar defines the correct use of the language's tags. XML allows the definition of the content of the data in a variety of ways, as long as the definition conforms to the general required structure.

Some of the advantages of XML are that it is human legible and reasonably simple, that it is easy to use and create and that it has support for a wide area of applications. Using XML to describe data gives the data a high degree of portability, separating the data from the applications using the data. Separating the data itself from the presentation of the data makes it possible to present the same data in different ways on different platforms. It therefore brings the type of portability to data that Java brings to software.

XML-RPC

XML-RPC is a protocol based on XML, used to make remote procedure calls. Because the remote procedure calls are formatted in XML, the protocol is language, platform and application independent. This gives it an advantage over language specific protocols like Java's RMI. It is also simple and lightweight, which gives it an advantage in simple implementations over more complicated middleware architectures like CORBA and DCOM.

Services Based Architecture

At the base of the services based architecture is the viewpoint of the ground support system and satellite or satellites used to attain a set of goals. The ground support system can be seen as consisting of a group of services designed to provide the functionality needed to manage the satellite mission[1,3]. A certain set of these services is mission specific and is ground-based as well as flight-based. There is also a set of standard services used by every mission. These services form the core of the ground support system and are shared and re-used in different missions. They fall into mainly two groups, namely

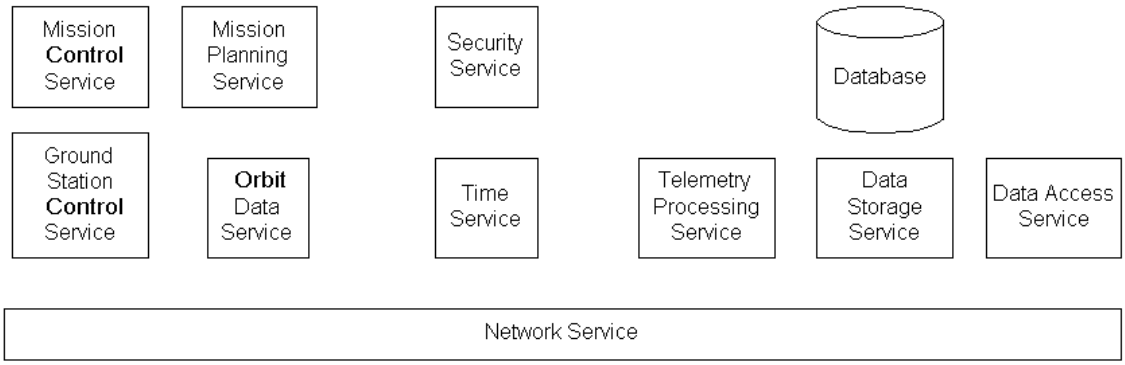

Figure 3: Services Based Architecture

a group providing application type functionality (like orbit prediction) and another for data acquisition, processing and storage.

Services needed to provide the functionality needed for future satellite projects are presented in figure 3 and will be discussed further in this section.

Network Service

The network service provides a network bus for components to function as a distributed system with total transparency. In other words the components are developed without the need to know where they will be distributed, or that they will be distributed at all. This allows developers with the needed scientific expertise to develop components without the need to learn complex network programming. The basis of communication in the network service is XML-RPC.

Data Storage Service

The core the data processing layer of the ground support system is the database. The data storage service provides the functionality to store data in the database. In an automated system this will typically be done by the telemetry processing service. Security is provided by the DMBS and prevents unauthorised users from entering data into the database. The data storage service is therefore an XML-RPC interface to the database.

Data Access Service

The data access service provides an XML-RPC interface to the database data. Data presentation clients receive the data in XML format and can display the data as needed without the database server having to format the data according to the platform or application used. For example presenting the data on the web becomes trivial with a Java applet acting as an XML-RPC client displaying the data.

Orbit Data Service

The orbit data service provides orbit information for a number of different satellites. Time of contact, position and visibility is some of the data provided by the service. The satellite data returned could be for the default ground station location and current time, or a query can be made for a specific location and time.

Time Service

The time service is used to ensure that the distributed network is time synchronised. The Network Time Protocol (NTP) is a protocol specifically used for time synchronisation and many clients and servers are freely available.

Security Service

Security on the distributed system is provided on an individual service basis. Access to services like the data storage service has to be restricted to users authorised to use them to protect the system and data from malicious damage. Currently security is implemented in each service that requires it, but it can be changed to a central security service where users are first authenticated and then given access to authorised services. Users are given a key by the security service and then use this key to access the other services. The services then authenticate the user according to the key from the security service.

Telemetry Processing Service

This service provides processing of raw telemetry data received from a satellite. It will typically process the raw data and store it in the database using the data storage service. The telemetry storage and forward mechanism for each mission will probably not be the same, so the telemetry processing service will be mission specific. A new processing component will have to be developed for every mission.

Ground Station Control Service

The ground station control service provides distributed control over a specific ground station. This includes control over all the hardware like radios and antennas and the tracking involved. Deploying multiple ground stations means duplicating and adapting this service to provide a standard interface for controlling each ground station. Standardisation of the interface to these services makes distributed control of multiple ground stations possible and allows a higher level of automation using these services to be included. As an example an operator uses the mission control service to command a specific satellite. The mission control service then automatically schedules the use of the correct ground station control services to communicate with the satellite when it next comes into range.

Mission Control Service

The mission control service provides the functionality to manage all components and services used for control the specific mission. It provides a front-end to send commands to the satellite and receive data from the satellite. It uses the ground station control service to communicate with the satellite. As automation is introduced it could schedule the use of the correct ground station automatically by using the orbit data service to see which ground station will next be in contact with the satellite.

Mission Planning Service

This service provides a transparent direct interface to the satellite and experiments. Automation of the ground system can be seen as a second layer of services on top of the services providing basic functionality. These basic services can be used together to control a satellite mission manually, like the manual control of the current SUNSAT ground station reviewed earlier.

The automation services layer then automates the use of the basic services by providing a higher level of abstraction for the user to work with. This enables not-technical users (like a scientist doing an experiment) to use the system without the need for detailed training on the use of the system. The mission planning service provides this an interface to these automated services so that high level mission planning can be done. Automation algorithms then break down the tasks ordered by the user into tasks performed by the basic services and schedule these tasks to be performed when necessary.

Network Service Implementation

The network service was implemented as a set of XML-RPC servers. A Java implementation was created with the help of Hannes Wallnofer's[4] Java XML-RPC library. Each server is a basic web server accepting XML-RPC requests. The requests are passed to handlers registered at the server and the server then returns the result. Any Java class can be registered as a handler, thereby exposing all public methods as remote procedures. By registering itself as an XML-RPC handler, the server makes remote configuration via XML-RPC available through its public configuration methods. Configuration data is stored locally by each server in XML format, making the use of standard XML tools for working with the configuration data possible.

Security in the current implementation is handled on the service level by each service requiring security. By using Java Secure Socket Extensions (JSSE) the communication between the client and server can be encrypted to provide secure communication. Future implementations could centralise security by using an authentication server where users are first authenticated and then given access to services they are authorised to.

Orbit Data Service Implementation

As an example of the integration of a COTS product the implementation of the orbit data service will be discussed. The service was implemented by providing an XML-RPC front-end to Predict, an orbit propagator developed by John A. Magliacane[5] (see figure 4), available under the GNU General Public License. The orbit data service handler uses Predict's existing network interface to retrieve data requested by the XML-RPC client. A sample client showing the position and footprint of a satellite was implemented to demonstrate the idea of separate development of the service and the clients (see figure 5).

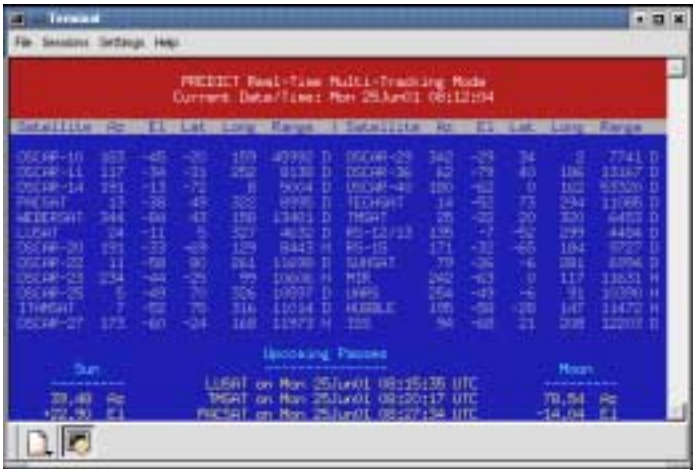

Figure 4: Predict Orbit Propagator

Using a COTS product like Predict in stead of implementing complex orbit propagator algorithms saved a great deal of time in the system development.

Deploying the software as a service on the network and separating the data display from the computation separates the development of clients using the data from the service development. Clients can therefore be developed and changed as needed independent from the service providing the data.

Figure 5: Satellite Tracking Client

Conclusions and Future Work

The services architecture provides a platform for the development and evolution of a new generation of ground support at the University of Stellenbosch. The reuse of components and integration of commercial off the shelf (COTS) products will bring along cost and time savings in the development and planning of future missions. Expanding to multiple ground stations gives the opportunity to build up a global ground support network of services available for use by multiple missions and organisations.

Future work to be done on the system includes the completion of the proposed implementation and the evolution of the proposed services as needed. The new architecture also opens up the opportunity for research into automation of the ground support and autonomous decision making systems.

References

[1] B. van der Merwe, "Micro-satellite Data Handling: a Unified Information Model", Masters Thesis, University of Stellenbosch, Department of Electric and Electronic Engineering, March 2001

[2] T. Bray, J. Paoli, C.M. Sperberg-McQueen and E. Maler, "Extensible Markup Language (XML) 1.0", Recommendation, W3C, October 2000

[3] W. Tai and D. Sweetnam, "A Mission Operations Architecture for the 21st Century", Proceedings of the 4th International Symposium on Space Mission Operations and Ground Data Systems, Munich, Germany, September 1996.

[4] Hannes Wallnofer, "XmlRpc-Java", http://helma.at/hannes/xmlrpc, 2001

[5] John A. Magliacane, "Predict – A Satellite Tracking/Orbital Prediction Program", http://www.qsl.net/kd2bd/predict.html, 2000

Biographical Information

Pieter J. Bakkes received his PhD in 1996 from the University of Sellenbosch, South Africa. He is currently a member of the faculty at the Department of Electronic Engineering at the University of Stellenbosch and he specialises in hardware design.

Richard M. Barry received his BSc (Computer Science, Physics) (1998) and BEng (Electric and Electronic Engineering) (2000) from the University of Stellenbosch, South Africa. He is

currently completing his MSc at the University of Stellenbosch. His main fields of study include distributed systems, network programming and their application to satellite systems.