1) SS-13: Using an 0S1 Process Control Standard for Monitor and Control

W. Randy I leuser, Richard J.. Chen and Michael 11. Stockett
Monitor & Control Technology Group
Jet Propulsion Laboratory
California Institute of' 1 'echnology
4800 Oak Grove 1 Drive
Pasadena, CA 91109

Abstract

The flexibility and robustness of a monitor and control (M&C) system arc a direct mull of the underlying inter-processor communications arclli[cc.lure.. A new architecture for M&C at the. Deep Spare Communications Complexes (DSCCs) has been developed based on the Manufacturing Message Specification (MMS) process control standard of the Open System Interconnection (OSI) suite of protocols. This architec ture has been tested both in a laboratory environment and under operational conditions at the Deep Space Network (DSN) experimental station (DSS-13).

'J"hc1)SS-13 experience in the application of OSI standards to support M&C has been extremely successful. MMS meets the functional needs of the station and provides a level of flexibility 'and responsiveness previously unknown in that environment. The architecture is robust enough (o meet CLI1 rent operational needs and flexible enough to provide a migration path for new subsystems. This paper will describe the architecture of the DSS-13 M&C system, discuss how MMS was used and the requirements this imposed on other parts of the system, and provide results from systems and operational testing at 1) SS-13.

Introduction

The Deep Space Network (DSN) operates the DSS-13 facility at the Goldstone Deep Space Communications Complex as a research site to conduct experiments in new tracking techniques, communication technologies and instrumentation. in response to expanding demand on station resources generated by the new 34-meter Beam Waveguide Antenna, a new monitor and control system was developed and installed based on 0S1 protocols [1]. As an experim ental facility, the 1) SS-13 site is constantly changing. New instrumentation is moved to the station and tested on a weekly basis. The procedures to support observations or spacecraft activities change daily. However, 1) SS-13 is also used to routinely for radio astronomy observations and the High Resolution Microwave Survey (HRMS). The Station Monitor and Control system was designed to support the constantly changing conditions at the station, while providing reliable support for operational experiments.

Requirements

The basic requirements for Station Monitor and Control wcrcdc.rived from the requirements for monitor and control of the DSN operational facilities. The operational facilities arc a collection of computer automated antennas and support equipment which arc linked together with a Local Area Network (LAN). The station is operated from a set of central control consoles which support the following requirements:

1) Allocation of station resources to provide assignment of subsystems in support of station activities.

2) Support data files arc. received at monitor and control and redistributed to subsystems as required. ,

3) Operator directives entered on monitor and control support the configuration and operation of subsystems.

4) Subsystem health and performance is displayed on monitor and control.

5) Subsystem event or alarm conditions arc generated and reported to monitor and control.

6) Monitor data is exchanged between monitor and control and the active subsystems based on negotiated interface agreements.

in addition 10 these. basic requirements, the constantly changing environment at 1) SS-13 created the following requirements.

Copyright ©1993 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

1) The Station Monitor and Control system must support the addition of a new subsystem without modification to the Software.

2) The Station Monitor and Control system must support modification of displays and construction of new displays without new software deliveries.

The additional requirements reflect concerns for flexibility and low life cycle costs. The design of Station Monitor and Control makes extensive use of commercial software products with the tools that provide the required flexibility.

Station Monitor and Control Architecture

The 0S1 based monitor and control architecture enables operation of DSS - 13 through a single operator workstation. 0S1 protocols arc used to link the operator workstation and the various stat ion subsystems across the LAN. Commercial software packages based on the OSI Manufacturing Message Specification (MMS) protocol were employed .to support all interprocessor communications, A commercial Supervisory Control and Data Acquisition (SCADA) software. package was incorporated to support most of the operator interface unctions through a graphical user interface (GUI). The extensive application of commercial software packages made it possible to build and install the system in one year at substantial cost savings [?].

There are two aspects to the OSI based monitor and control architecture. The first is the communications architecture which employs all seven layers specified in the OSI Basic Reference Model. Support for the communications architecture resides on all of the station computers. The second aspect involves the structure of the program (or programs) that run on each of the station computers. This structure is called the software architecture and is tightly coupled to the computer hardware, the operating system and the computer language employed. Since 98% of the software for Station Monitor and Control was purchased, much of the software architecture was set by the vendors. This discussion is therefore limited to those components used to integrate the commercial products into a working system. Both aspects of the architecture are depicted in Figure 1 and the station network configuration is seen in Figure ?.,

Station Monitor and Control Implementation

The Station Monitor and Control system was initially implemented on an Intel 80386, 33 MHz computer running the Interactive UNIX operating system. The computer platform was provided to the project and selection of the software components was governed by the platform. The selection of the UNIX operating system was made to provide a flexible multi-tasking environment. The platform and operating system dire. ctc.d the selection of the specific MM S/OSI software package. In addition, the platform and operating system led to the selection of a Supervisory Control and Data Acquisition (SCADA) software package called Dexterity. The Dexterity product met all of the basic requirements to support the opc.rater interface and provided an Application Program Interface (API) to support software development. The software development effort was limited to providing a link between Dexterity and the MMS network services.

The MMS software package provided a set of libraries and sample code to build h4MS service calls. Using this package and the Dexterity API, the following elements were developed to support the MMS/OSI network services:

- The **MMS-Dexterity** Network Server provides all of the network services for Station Monitor and Control. The network server is a program that runs concurrent to Dexterity under the UNIX operating system and by UNIX convention is called a daemon.

- The **Dexterity Interface Program** is used to initiate the MMS-Dexterity network server to take action. The program is initiate.d to run when an operator uses the mouse. or keyboard to trigger some network action.

- The **MMS-Dexterity Timer Program** runs concurrent to the. network serve.r and provides a heartbeat to initiate polling of subsystem status data.

The key and most complex element of the software. architecture is the MMS-Dexterity Network Serve.r. The network server is started simultaneously with the Dexterity product and is table driven so changes to the network can be accommodated without changing the network software. The network server establishes the environment to communicate across the LAN by reading a table with all of the network address

Copyright ©1993 by the American Institute of Aeronautics and Astronautics, Inc. Aiirip]ltsrc.serve.(i,

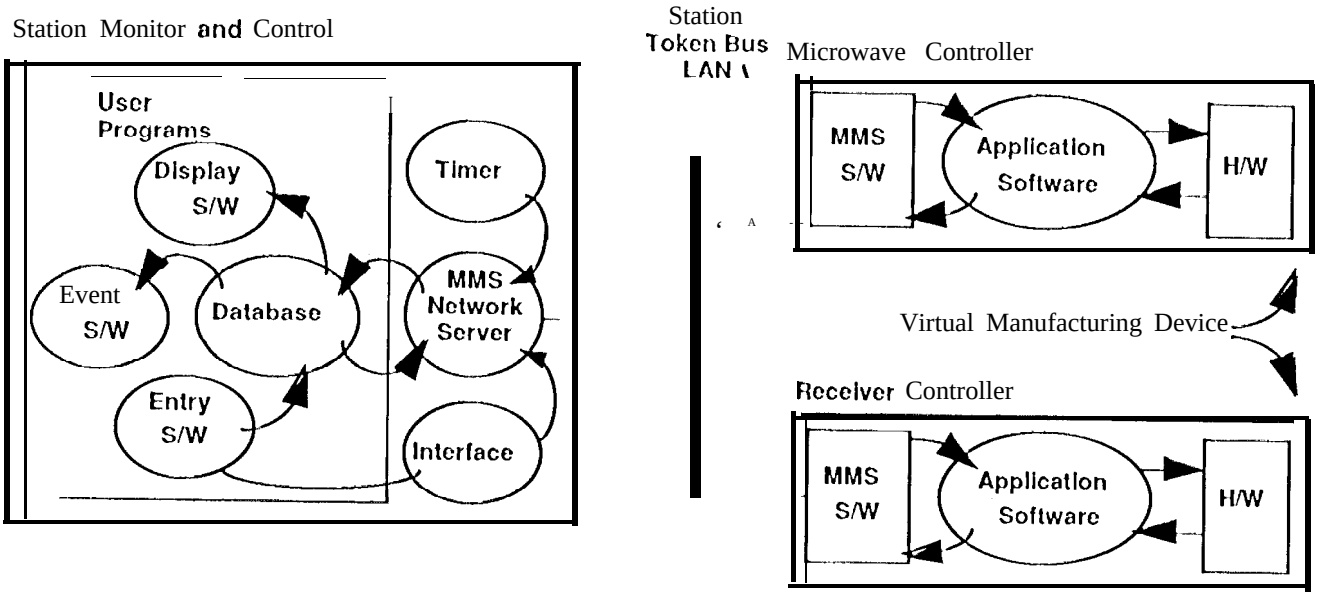

Figure 1. Sotfware Architecture for DSS-13.

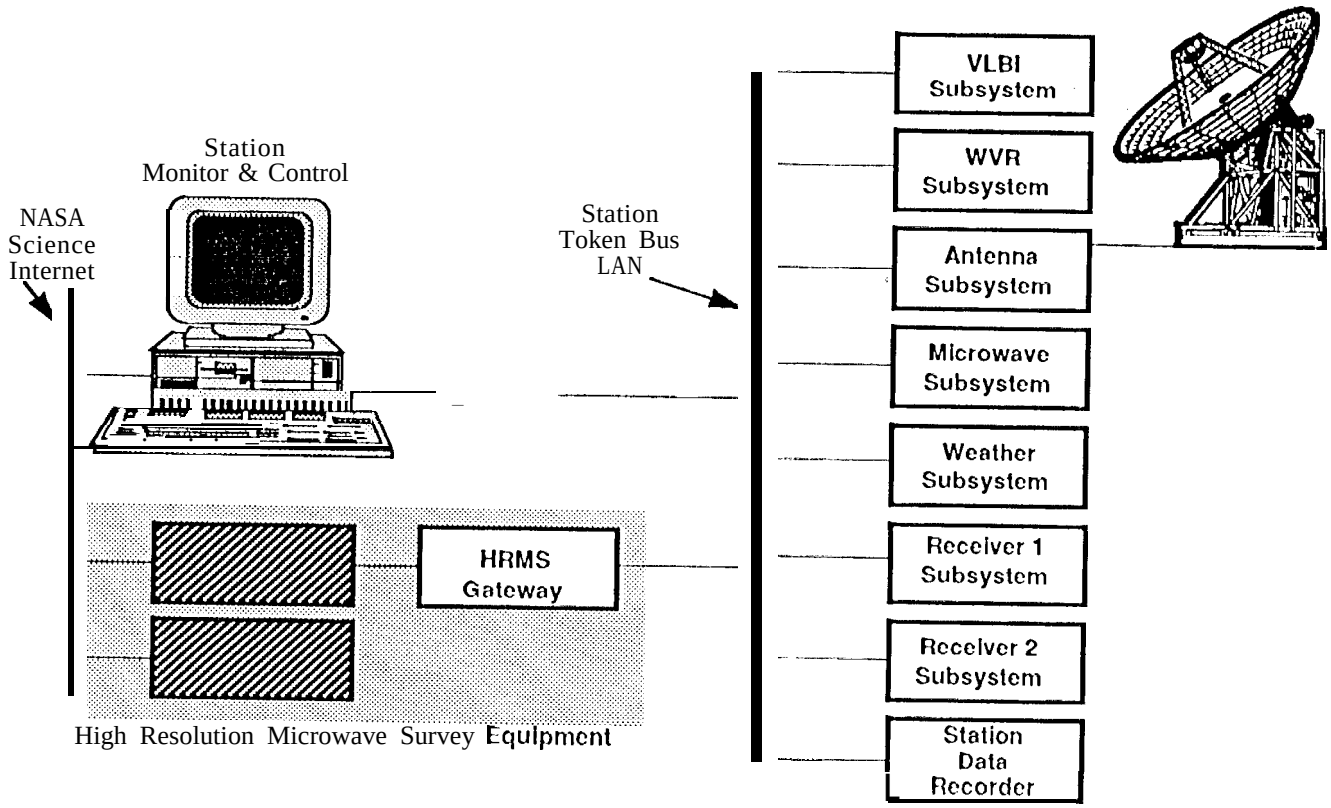

Figure 2. **Network configuration** for DSS-13.

Copyright ©1993 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

information. Next, the network server reads a set of tables that identify all of the control and performance parameters used to operate the subsystems. Finally, the network serve.r establishes communications with the 1 Dexterity database.

Once initialized, the network serve.r is ready to support all of the, network communication sei vices required 10 cy)c.rate the complex from the Station Monitor and Control workstation. Stat ion pei sonal operate the complex through a graphical user interface (GUI) managed by the Dexterit y software. The Dexterity software supports operator actions through the keyboard or a mouse. The I Dexterity database services the user screens. In addition, the. Dexterity product provides a set of tools 10 build and manage the GUI screens. The basic operation of the station is accomplished as follows:

1) Station resources are allocated to support station activities by establishing a connection between Station Monitor and Control and the selected subsystems. Resources arc released by concluding the connection. The process of establishing and releasing a connection is accomplished using the mouse..

2) Subsystem health and performance is read by, or reported to Station Monitor and Control using the MMS variable read service. or the information report service. Changes in subsystem status and performance arc echoed on Station Monitor and Control using the same MMS services.

3) Operators control a subsystem by entering configuration parameters through the GUI into the Dexterity database. The operator action triggers the Dexterity Interface Program to execute and pass the selected parameters to the Dexterity Network Server. '1 The net work server extracts the selected data from the Dexterity database and writes the data across the network to the subsystem.

4) Support data files arc received by Station Monitor and Control from JI 'L across the NASA Science Internet. Operators distribute support files to individual subsyslc.ills using the MMS file transfer services. The operator uses a Dexterity screen to enter the file name to be transferred, the destination file name and the destinat ion subsystem.

5) Subsystem event or alarm conditions are supported in two ways. Both methods combine the MMS information report service with the Dexterity alarm service. In the first method, the subsystem is programmed to report data based on changing conditions. The subsystem software engine.cr can report on change, report on critical condition and/or report on return 10 nominal condition. The Dexterity database is then configured to alarm the operator when a reported data element goes outside its limits. In the second method, the subsystem is programmed to report a text message to Station Monitor and Control. The conditions that warrant a text message arc left to the discretion of the subsystem software engineer. The text message is sent to Station Monitor and Control using the MMS information report service. The Dexterity database is configured lo display the.sc. text messages as events or alarm as a function of the message type.

6) Monitor data is exchanged between directly between subsystems based on negotiated interface agreements using the MMS read variable service or the MMS information report service.

Subsystem integration

All of the subsystems at DSS-13 arc based on the. Intel 80x86 computer and the DOS operating system. Personal computers and the, DOS operating system are frequently used by experimental subsystems because they arc. inexpensive and simple to program, DOS is a "single, user" operating system and supports execution of only one program at a time. The subsystem program is strut.turcd to establish the. subsystem configuration and status on execution, and to run in a continuous loop until an operator interruption triggers termination.

The key element in the application of Mh4S is the abstract model called the Virtual Manufacturing Device (VMD). The VMD model is used (o describe the externally visible. characteristics of a real device. Software modules which manipulate a real device are called VMD objects. These objects can be manipulated using MMS services such as context management, variable. acce ss, domain management, semaphore management and event manageme nt services. State changes detected in

Copyright ©1993 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

the real device anti defined in the VMD model can trigger MMS services. For the implementation at 11 SS-13, a minimum of effort was required to make the subsystems appear as VMDs.

'1'0 minimize the impact of the monitor anti control effort on subsystem development, software routines were. developed to provide network integration with a minimum of impact to the subsystem code. The impact 10 subsystem code is reflected in the Jmu(io-code below:

```
main()
(
  dsn_init_mms(); /* routine called to
                     set up lilt network
                     environment */

  set_up_subsystem_environment();

  while( not_interrupted_by. operator )
  {
      monitor- subsystem_operations();
      execute_subsystem_commands();

      dsn_comm_server();
                     /* routine called to'
                     check for network
                     messages    */
  )

  set_subsystem_in_safe_shut_down_
  conditions();

  dsn_terminate_mms();
                     /* routine called to
                     terminate the network
                     environment */
  exit();
)
```

A summary of the MMS services and routines developed for the subsystems and the Station Monitor and Control workstation is provided below.

MMS Context Management Services

The routine *dsn_init_chan()* is used by a client subsystem to initiate communications with a serve.r.

The routine *dsn_conclude()* is used by a client subsystem to close an association with a server subsystem.

The routine *dsn_listen()* is used by a server subsystem to open a communication resource. The communication resources for each subsystem, commonly referred to as channels arc established when the subsystem initiates operation. A subsystem that operates as a server will allocate listen channels for clients establish communications. The dsn_listen() routine. is provided to permit a subsystem to allocate additional channels as serve.r channels while. operating. The *dsn_stop_listen()* is used to close a channel in the listen mode.

The *dsn_abort()* is used by a client to terminate a client-server association without regard for pending rc.quests. This service is intended for use under special conditions and not to be used in place of the dsn_conclude() service.

MMS File Management Services

The routine *dsn_copy_file()* is used by a client subsystem to copy a file. from a server subsystem.

The routine *dsn_send_file()* is used by a client subsystem to invoke the MMS obtain file service in which a server subsystem copies a file from a client subsystem.

The routine *dsn_delete_file()* is used by a client subsystem to delete a file, from a server subsystem,

MMS Variable Services

The routine *create_dsn_mms_types()* is used to establish a variable as a network object. Subsystem implementors use this routine to register variables that will be managed across the network. The registration process identifies the address of the variable, its type and size to the subsystem network software..

The routine *reg_write_var_indic_func()* is used to identify a function to be called by the server, When a specific named variable is written across the network. For example, a server subsystem could have a physical switch that is set according to the value of a named variable. By design, the switch must be reset to its new condition on change of the named variable.. The server subsystem software engineer can write. a function that affects the physical change of the switch and use. the reg_write_var_indic_func() routine to associate the function with the **named** variable. When the write indication is received for the associated named variable, the function is called.

Copyright ©1993 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved

'J 'he routine *reg_ write var confr_ func() is used 10* identify a function to be Called by the client, when a specific named variable. is written acr 0ss the network and the write confirmation is received by the client. Extending the example above., a client could have a sequence of actions to perform that arc dependent of the server subsystem switch position. The client subsystem software engineer can write. a function that affects action that can only occur after the switch reset has been affected. The client function can be associated with the named variable write confirmation using the rcg.w'rite._v ar_corlfr.fllllco routine. When the, write confirmation is received for the associated named variable, the function is called.

'I'llcrotllirlcr cg-read-v~~r *indic_ func()* is used to identify a function to be called by the server, when a specific named variable is read across the network. For example, the status of some element **of a server** subsystem is reflected by a named variable. The subsystem software engineer can write a function to update the status and associate that function with the specific namCd variable using the reg_ read. var_ indic_ func() routine. Whcrl'the read indication is received, the function will be called and the read response will reflect [he. latest status.

The *reg_ read_ var_ confr_ func()* is used to identify a function to be called by the client, when a specific named variable is read across the network and the confirmation is received by the client. Extending the example. above., the client subsystem software engineer will write a function to take action on the status returned by the server subsystem. That function is associated with the specific name.d variable using the reg. read_ var_ confr_ func() routine. When the read confirmation is received by the client, the function is called.

The *reg_ info report_ func()* is used 10 identify a function to-be called by the client, **when a** specific named **variable** is reported by a server using the *dsn_ send_ info_ rpt()* service.

Summary

The Station Monitor and Control system has been operating on a daily basis at DSS-13 since December, 1992. The operation personnel have de veloped an extensive knowledge and understanding of the system. The station personnel perform routine configuration managemen t, develop new tools and new displays as conditions changes.

The implement ation of 1) SS-13 Station Monitor and Control has demonstrated that the MMS/OSI protocols can be used to support interprocessor communications in a spacecraft tracking station. The integration of a commercial user interface package with an MMS network server has created a powerful, flexible 1001 to support the. constantly changing requirements at DSS-13. Most of the cost savings derived from the MMS/OSI approach comes from the application of commercial products. The software developed in this activity to support the MMS network services on Station Monitor and Control is less that 20,000 lines of code. The commercial software products represent more that 750,000 lines of code. The functional capabilities provided through the combination of the.sc two technologies offer great potential for cost savings.

The integration of the two commercial products required three. months with an additional three months of comprehensive testing bc.fore dc.livery to the station. We have. recently intc.grate.d a different SCADA product with the same MMS product on a Sun SPARC 2. Though the API for the ncw SCADA product was different, the. process only required two months. The table driven approach to the network interfaces and the. SCADA products provide a flexible environment for station personal to respond 10 changing conditions.

The MMS/OSI communications architecture **also** provides a flexible environment to changing requirements. The Virtual Manufacturing Device model for subsystems provides a single interface to network services. When the. High Resolution Microwave Survey project discovered problems with their interface to the antenna equipment, the implementation of a network gateway was a straight forward process, requiring no modifications 10 the network.

The most difficult part of the effort was upgrading the individual subsystems to enable them to operate over the I, AN. Because the decision to proceed with an integrated, centralized M&C system was made after the subsysten is began development, the subsystem s were, in essence, retrofitted to support the ncw architecture. Fifty percen t of the Station Monitor & Control effort was spent modifying the subsystems (o include a LAN interface, addressing problems **which occurred** due **to** the

Copyright ©1993 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

limitations of the DOS operating system chosen by the subsystems, and integrating a variety of commercial packages to support subsystem-specific problems.

The DSS-13 Architecture is now serving as a baseline for testing automation concepts. The open, flexible architecture has greatly reduced integration time. DSS-13 is serving as an operational testbed for the Link Monitor & Control Operator Assistant, an Artificial Intelligence-based tool which automates configuration, calibration, test, and Operation of ground station equipment [3].

Acknowledgments

The work described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

Refer-cnccs

[1] W. Randy Heuser, "An OSI Architecture for the Deep Spat.c. Network," TDA Progress Report 42-109, Jet Propulsion Laboratory, Pasadena, CA, March 30, 1992. '

[2] W. Randy Heuser and Lynne P. Cooper, "An OSI Architecture for the DSN," SpaceOps '92., Pasadena, CA, November 1992.

[3] Lorrine F. Lee and Lynne P. Cooper, "Operations Automation Using the Link Monitor & Control Opc.rater Assistant," to be published in the Proceedings of AIAA Computing in Aerospace '9, San Diego, CA, October 1992

Copyright ©1993 by the American Institute of Aeronautics and Astronautics, inc.. All rights reserved.